ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA, INFORMATICA E TELECOMUNICAZIONI

TITOLO DELL'ELABORATO

CARATTERIZZAZIONE ENERGETICA DI SISTEMI A MICROCONTROLLORE BASATI SU TECNOLOGIA FRAM

Elaborato in:

ELETTRONICA DEI SISTEMI DIGITALI

Relatore

Prof. Aldo Romani

Presentata da

Fabio Corelli

Correlatore

Dott. Matteo Filippi

I Appello - Sessione II

Anno Accademico 2013/2014

Parole Chiave:

- *Energy Harvesting*
- *Microcontrollori*
- *Memoria FRAM*
- *Ultra – Low Power*

INDICE

Introduzione

Il progresso nel campo della tecnologia elettronica, nel corso degli anni, ha portato a sviluppare dispositivi elettronici in grado di alimentarsi attraverso energie rinnovabili ed "infinite", provenienti dall'ambiente esterno, svincolandosi dalle tradizionali fonti energetiche, come il petrolio, l'uranio e il carbone. Ci si sta orientando, quindi, verso il campo dell'Energy Harvesting.

L' Energy Harvesting (tradotto come energia raccolta) è un processo mediante il quale minuscole quantità di energia presenti nell'ambiente (sotto forma di vento, onde, luce e vibrazioni) vengono trasformate in energia elettrica la quale viene poi immagazzinata in appositi dispositivi accumulatori come condensatori o piccole batterie ricaricabili. A causa, però, delle bassissime potenze messe a disposizione dall'ambiente, che vanno da qualche nano-watts a qualche centinaia di milli-watts per centimetro cubo, questa forma di energia non ha mai trovato spazio nei precedenti sistemi embedded fino a quando il progresso tecnologico non lo ha permesso, accrescendo sia l'efficacia energetica dei componenti elettronici, tra i quali i microcontrollori, riducendone sensibilmente il loro consumo, sia la qualità di conversione dei trasduttori, i quali hanno una maggior capacità di catturare sensibili quantità di energia dall'ambiente per poi trasformarla in energia elettrica.

I microcontrollori, come detto in precedenza, hanno un ruolo fondamentale nei sistemi integrati moderni. Il loro bassissimo consumo diventa la caratteristica più importante per far sì che essi siano impiegati con successo nelle applicazioni embedded di ultima generazione. Si tratta di consumi estremamente bassi, tanto da far durare una batteria fino a 15 anche 20 anni. L'imperativo di basso consumo è, dunque, un obiettivo obbligato per i produttori di microcontrollori che devono soddisfare le richieste sempre più stringenti degli sviluppatori in termini di consumo di potenza dei sistemi, rimanendo competitivi nei confronti degli altri produttori. [1]

L'obbiettivo di questa tesi è proprio quello di caratterizzare dal punto di vista energetico un microcontrollore per applicazioni "Ultra – Low Power" basato su tecnologia FRAM, valutandone i relativi consumi di potenza nelle diverse modalità di funzionamento.

Nella prima fase ci sarà uno studio dei microcontrollori, evidenziando ciò che li costituisce. Successivamente, verrà descritto con attenzione il microcontrollore MSP430FR5969 oggetto di studio e misure, mettendone in risalto l'architettura interna, la tecnologia di memoria FRAM e i modi di funzionamento Ultra-Low Power. Poi si descriverà il sistema di misura utilizzato durante l'attività e si mostreranno le misure effettuate e i dati raccolti, al fine di caratterizzare energicamente il microcontrollore.

1. Microcontrollori

In elettronica digitale, quando si parla di microcontrollori (*fig_1.1*) o **MCU** (**M**icro **C**ontroller **U**nit) si fa riferimento a sistemi elettronici a microprocessore completo, integrati su singolo chip, progettati per ottenere la massima autosufficienza di funzionamento e il miglior rapporto prezzo – prestazioni delle applicazioni specifiche (special purpose) di controllo digitale sulle quali generalmente vengono utilizzati.

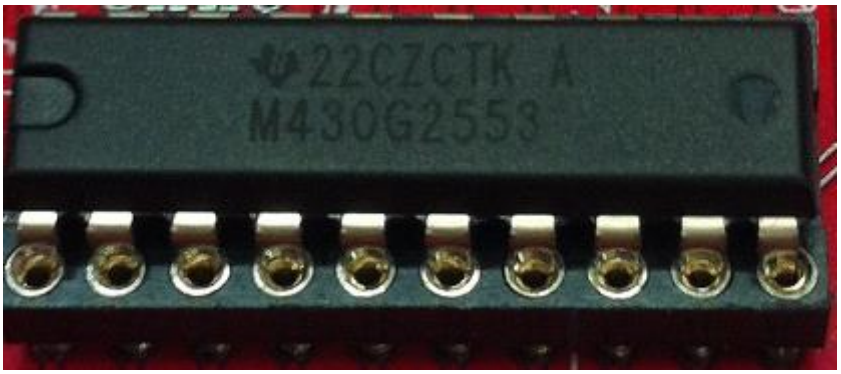
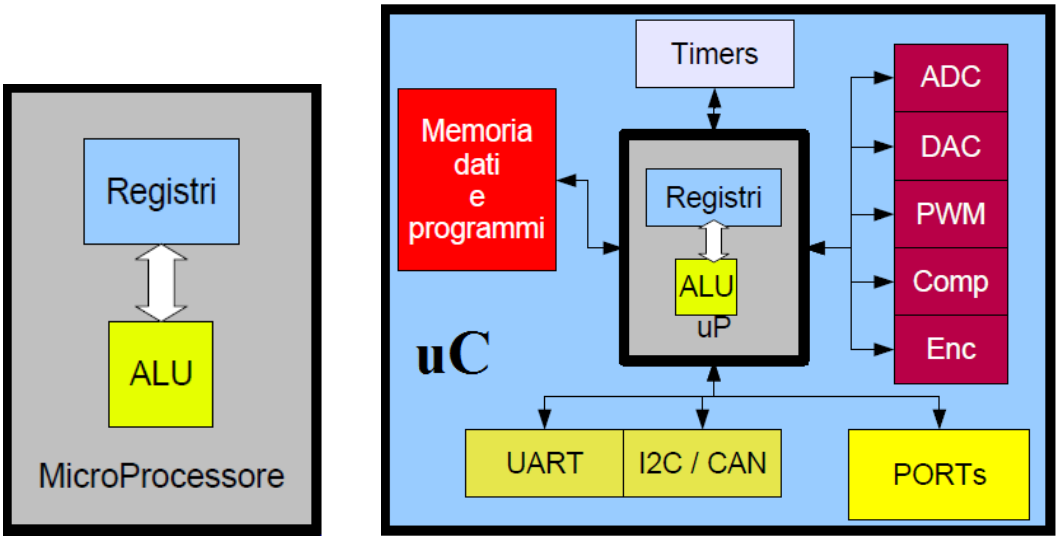


fig_1.1 – Esempio di microcontrollore; nello specifico microcontrollore MSP430G2553 della TI

A differenza dei microprocessori, che oggi giorno sono l'implementazione più comune di una CPU e che per poter operare hanno bisogno di numerose integrazioni esterne quali memoria, periferiche di I/O, oscillatore di clock, i microcontrollori raccolgono tutti gli elementi utili all'interno di un unico circuito integrato, rendendo superfluo il bisogno di aggiungere altri componenti esterni per poter funzionare (*fig_1.2*).



a) *b)*

fig_1.2 – Schema a blocchi dell'architettura del microprocessore a) e microcontrollore b) [11]

Provvisti di CPU, memoria volatile RAM e non volatile ROM, interfacce standard di I/Os fra le quali I^2C e SPI, i microcontrollori permettono di interagire direttamente con il mondo esterno tramite un programma specifico residente nella propria memoria interna[3] e grazie soprattutto alle periferiche integrate, dispongono all'interno di convertitori ADC e DAC, timer, comparatori e PWM particolarmente utili in sistemi embedded come antifurti, strumenti di misurazione, trasmettitori/ricevitori e carica batterie.

1.1 ARCHITETTURA DEI µC

Con architettura di un microcontrollore, *fig_1.2 b)*, si intende la connessione logica dei dispositivi che lo compongono e ne permettono il funzionamento. Si possono distinguere due tipologie di architetture: l'architettura secondo il modello di Von Neumann, mostrata in *fig_1.3*, dove in un unico spazio di memoria sono condivisi i dati relativi al programma e il set di istruzioni del programma, e l'architettura Harvard, in *fig_1.4*, nella quale il programma e i dati relativi allo stesso sono allocati in due spazi di memoria distinti.

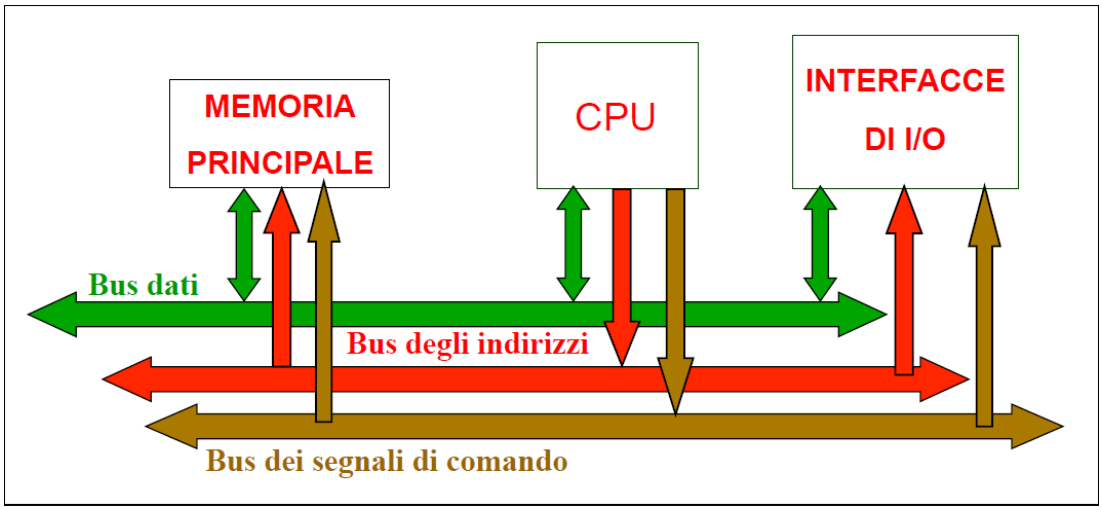


fig_1.3 – Modello di Von Neumann [13]

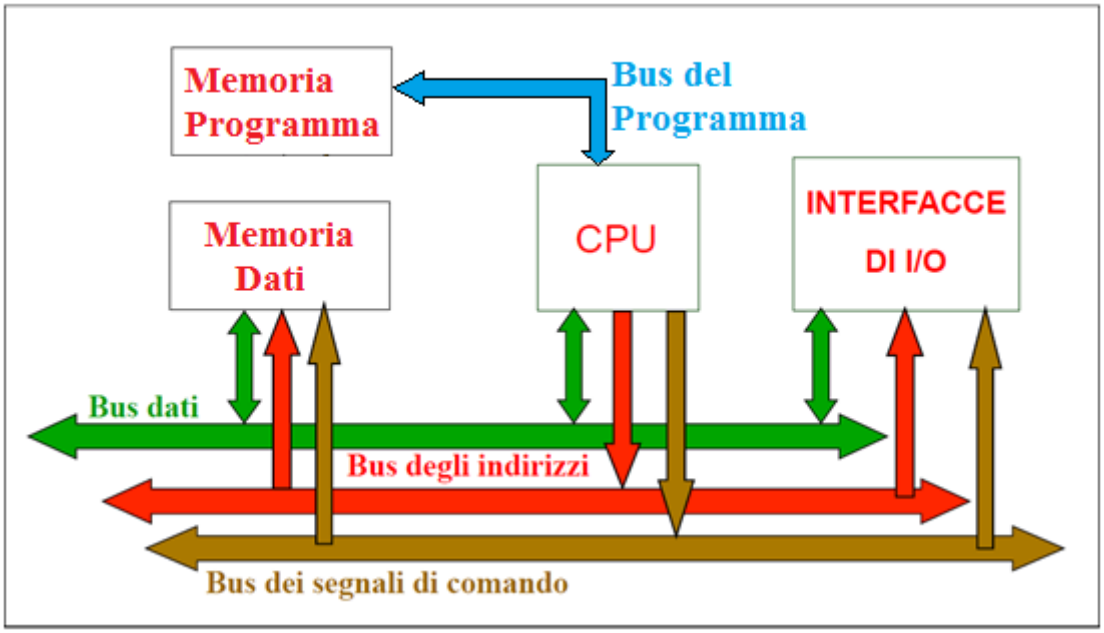


fig_1.4 – Modello di Harvard [13]

Il fatto di avere un'architettura secondo il modello di Von Neumann può chiaramente non essere una soluzione ottimale, soprattutto per la gestione dei dati, vista la singola presenza di un'unità di memoria. Al contrario di come sembra, questo tipo di scelta risulta essere ottimale per minimizzare la complessità della CPU. Minor complessità della CPU vuol dire minor utilizzo della circuiteria interna; di conseguenza, si ha la possibilità, di poter ridurre anche i consumi. L'architettura di Harvard, invece, ha sì una maggior complessità della CPU, ma è in grado di consentire a quest'ultima di raggiungere elevate velocità di esecuzione.

1.1.1 LA CPU

L'unità di elaborazione centrale ha il compito di gestire i dati memorizzati in memoria, prelevarli, elaborarli e restituirne il risultato.

La struttura interna della CPU, mostrata in *fig_1.5*, è composta da due blocchi principali: il **Datapath**, che contiene tutte le unità di elaborazione ed i registri necessari per l'esecuzione delle istruzioni della CPU e l'**Unità di Controllo**, una particolare rete sequenziale sincrona che invia un insieme di segnali di controllo ad Datapath specificandogli l'esecuzione di una determinata operazione [13].

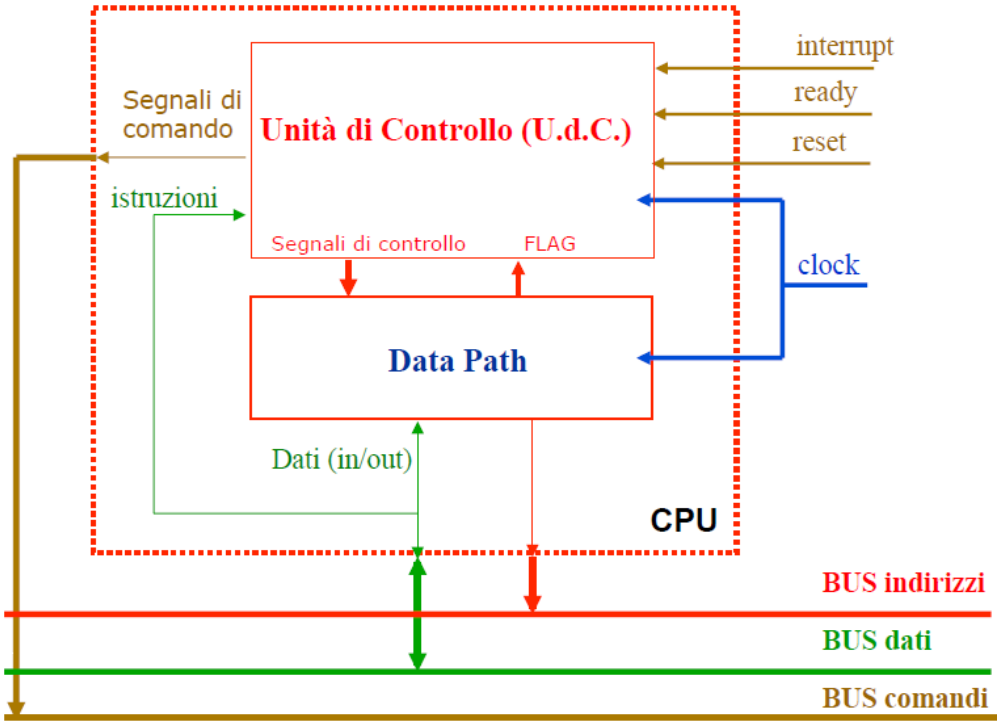


fig_1.5 – Schema a blocchi della CPU

Di seguito, in *fig_1.6*, si mostra il contenuto del Datapath della CPU.

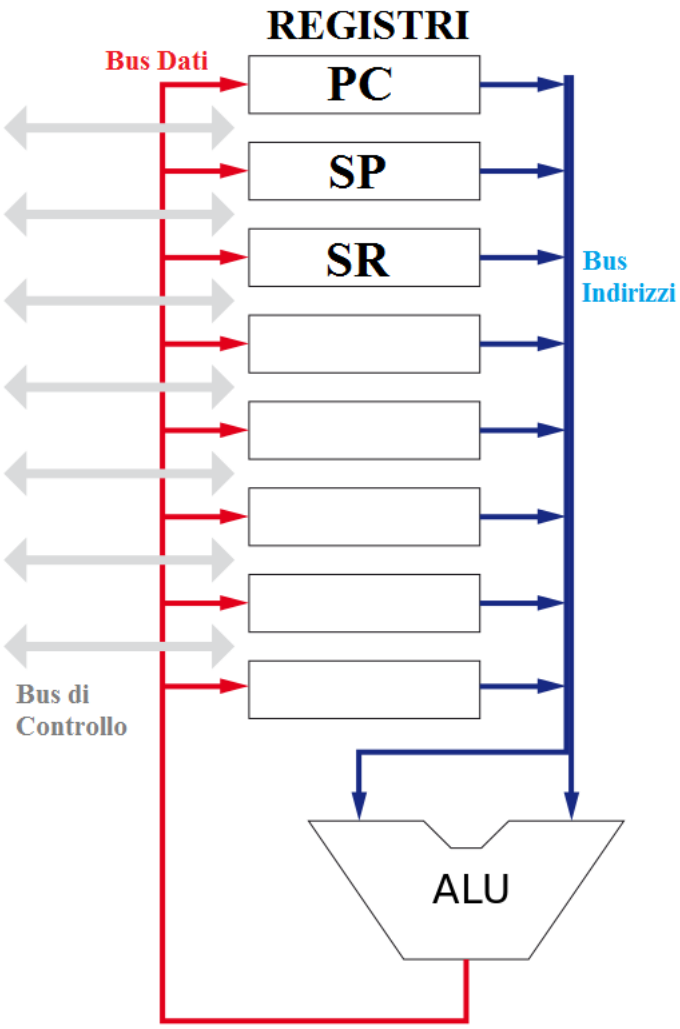


fig_1.6 – Datapath

Ricapitolando, gli elementi principali che costituiscono una CPU sono:

- l'unità di controllo, il cui compito è quello di gestire le operazioni necessarie per l'esecuzione di una o più istruzioni;
- l'unità aritmetico logica (ALU), che si occupa di eseguire le principali operazioni di tipo aritmetico e logico;
- Set di registri ad elevata velocità di accesso, di cui alcuni ad uso dedicato e altri di uso generico (general purpose). Il numero totale dei registri dipende dalla complessità della CPU.

Per esempio, alcuni dei registri dedicati sono:

- **Program Counter (PC):** ha la funzione di contenere l'indirizzo di memoria della successiva istruzione da eseguire;
- **Stack Pointer (SP):** contiene l'indirizzo della locazione di memoria occupata dall'ultimo dato inserito nello stack (top of stack). Lo stack è una struttura dati della memoria principale, gestita dalla CPU con la logica LIFO (Last Input – First Output), utilizzata principalmente per la gestione delle routine di interrupt, ovvero interruzioni temporanee del programma ad opera di eventi esterni;
- **Status Register (SR):** è un insieme di bit, denominati flag, che determinano particolari stati della CPU.

Ogni CPU è caratterizzata dai seguenti parametri:

- Parallelismo dell'ALU, ovvero il numero di bit che l'ALU riesce a processare;
- Frequenza di clock, ossia il numero di operazioni elementari per secondo che la CPU è in grado di eseguire;
- Spazio di indirizzamento in memoria, ovvero la dimensione della memoria nella quale risiedono programma e dati;
- Parallelismo o ampiezza del Bus Dati, ovvero il numero di segnali di cui è composto un BUS.

1.1.2 MEMORIA

In un microcontrollore la memoria ha un ruolo fondamentale per il funzionamento del dispositivo in quanto

permette di memorizzare il programma che la CPU deve eseguire e la configurazione delle periferiche. A seconda dell'architettura, che sia di Von Neumann o di Harvard, il microcontrollore possiede una o due memorie: nel primo caso è fornito di una memoria non volatile dove risiedono sia il programma sia i dati, mentre nel secondo caso dispone di una memoria non volatile dove risiede il programma ed una volatile dove vengono memorizzati i dati. Nei più recenti microcontrollori, progettati la gran parte secondo il modello Von Neumann, è solito utilizzare come memoria non volatile la FLASH, in quanto:

- mantiene salvati programma e dati indefinitamente anche senza la tensione di alimentazione presente;
- permette la lettura dei dati in tempi brevi in modo da non rallentare il microcontrollore;
- può essere scritta, modificata e cancellata senza l'ausilio di particolari apparecchiature esterne;
- è possibile integrarla in modo semplice nella struttura del microcontrollore, consentendo, di conseguenza, di ridurre i costi di quest'ultimo.

Come memoria volatile, invece, si è soliti utilizzare una RAM, nella quale vengono memorizzati i dati relativi al programma che il microcontrollore deve eseguire. Hanno il vantaggio di essere molto veloci, in quanto permettono di accedere a qualunque indirizzo di memoria con lo stesso tempo di accesso [3], ma per la loro natura volatile, perdono i dati contenuti allo spegnimento dell'alimentazione del sistema.

1.1.3 OSCILLATORE

L'oscillatore è un circuito che genera una frequenza di clock, ovvero il numero di transizioni tra due livelli logici "0" e "1" che la CPU è in grado di eseguire in un secondo, senza l'ausilio di un segnale di ingresso. Da esso dipende, perciò, la velocità con cui un microcontrollore legge, processa ed esegue il programma contenuto in memoria. Negli MCU le sorgenti di clock possono essere classificate come interne od esterne:

- Interne: la velocità di clock è direttamente generata dal microcontrollore. All'interno di esso risiedono, solitamente, due oscillatori: uno a bassa frequenza, utilizzato per lo più per contenere i consumi del dispositivo, ed uno ad elevata frequenza, impiegato quando si ha la necessità di eseguire istruzioni il più velocemente possibile trascurando, di conseguenza, i consumi.
- Esterne: per generare la frequenza di clock vengono impiegati circuiti esterni, come oscillatori al quarzo (quando si richiedono precisione e frequenze elevate), generatori di onda quadra esterni (quando lo stesso clock deve raggiungere più dispositivi) o risonatori RC (quando non si hanno elevate esigenze di precisione e stabilità ma si punta sulla semplicità e sull'economicità del sistema).

1.1.4 PERIFERICHE

Le periferiche sono il punto di contatto tra il mondo esterno ed il microcontrollore. Grazie ad esse, quest'ultimo ha avuto un'esponenziale crescita nel mercato

dell'elettronica, diventando il dispositivo principale dei sistemi integrati di tutti i giorni. Tra le periferiche più comuni che i microcontrollori possiedono, si hanno:

- **Porte di I/O** utilizzate per portare, dall'interno all'esterno o viceversa, dati contenuti nella memoria del dispositivo. Una volta inizializzate vengono viste dalla CPU come una locazione di memoria che può essere letta o scritta [10].

  Solitamente una porta di I/O è composta da una serie di pin configurabili individualmente, attraverso degli appostiti registri, o come input (ai quali è possibile attivare o meno una resistenza di pull-up) utilizzati principalmente per il collegamento a circuiti di interrupt o di un ADC, oppure come output per il collegamento di led o di altre periferiche come ad esempio i timer o i moduli di clock interni.
- **Moduli di comunicazione** che permettono al microcontrollore di dialogare con altri sistemi integrati. Il più comune è il sistema SPI, ma in alcuni microcontrollori è possibile avere anche sistemi basati su interfaccia I^2C e UART.
- **Periferiche analogiche** come ADC (solitamente a 10 o 12 bit), comparatori, amplificatori operazionali
- **Periferiche digitali** come controllori di display LCD, modulatori PWM o timer programmabili. Questi ultimi, in particolare, vengono utilizzati per introdurre dei ritardi nel programma del microcontrollore oppure per generare degli interrupt. Nei più recenti microcontrollori è stato introdotto un timer di controllo, chiamato Watchdog Timer, il cui compito è quello di monitorare l'esatta esecuzione del programma e nel caso si verifichino blocchi o guasti farlo ripartire.

2. Low Power Contest

Nel mercato odierno, i sistemi embedded che ci circondano sono sempre di più e diventa difficile privarsi delle tante comodità che essi offrono. Ma la voglia di svincolarsi dalle fonti non rinnovabili per sostenere il funzionamento di questi sistemi e sfondare nel campo dell'Energy Harvesting ha portato alla costante ricerca di dispositivi microcontrollori con consumi estremamente bassi, in modo da poter conservare l'energia di una batteria per decine di anni. Con questa idea in testa, diverse case costruttrici di sistemi integrati ha dato vita ad una vera e propria sfida, un Low Power Contest, tra microcontrollori simili basata sul consumo energetico.

Tra queste, Texas Instruments e Microchip sono quelle che più si "sfidano". Infatti, nell'ApplicationNote 1267A [18] della Microchip si mettono a confronto i prodotti di punta della Microchip, i PICF24 XLP, e quelli della TI, gli MSP430F. In particolare viene riportata una tabella, mostrata in *fig_2.1*, che relaziona i consumi di corrente:

Parameter	Device							
	PIC16LF72X	PIC18F46K20	PIC18LF46J11	PIC16LF193X	PIC18LF14K50	PIC24F16KA102	Atmel® ATmega168P/328P	TI MSP430F21X1/ MSP430F21X2/ MSP430F22X2/4
Deep Sleep (nA)	—	—	13	—	—	20	—	—
Sleep (nA)	20	100	54	60	24	25	100(1)	100
WDT (nA)	500	500	830	500	450	420	4200(1)	300-700
32 kHz Oscillator/RTCC (nA)	600	500	820	600	790	520	800	700
I/O Port Leakage (nA)	±5	±5	±200(2)	±50	±5	±50	±1000(1,2)	±50(2)
1 MHz Run (µA)	110	300	272	150	170	195	300	200-270
Minimum VDD	1.8	1.8	2	1.8	1.8	1.8	1.8	1.8

fig_2.1 – Low Power Contest tra Microchip e TI [18]

Come si può notare, la Microchip dichiara dei consumi estremamente bassi comparati a quelli della TI. Ad esempio, 25nA del PIC24F contro i 100nA del MSP430 in modalità sleep e 195µA del PIC24F contro i 200-270µA del MSP430. Da questo confronto, dunque, si evince che la sfida è nettamente a favore del dispositivo della Microchip.

TI, però, non è stata a guardare e ha prontamente risposto al confronto fatto dalla Microchip pubblicando un documento [19] nel quale vengono risaltate alcune incongruenze nelle modalità di operare della Microchip, reputando i consumi dichiarati in *fig_2.1*, non veritieri. Per citarne un esempio[1], la Microchip, nella modalità di funzionamento DeepSleep, dichiara che la RAM viene privata dell'alimentazione; in questo modo, come ben si sa, i dati contenuti in essa vengono persi. Questo vuol dire che i dati devo essere scritti nella Flash del PIC qualora si vuol far funzionare il programma. Questo tipo di operazione richiede, nei PIC24F XLP, circa 10mA ogni 2ms per scrivere ogni blocco di 96 Byte [19]. Si intuisce allora che non si potranno mai avere i 20nA dichiarati da Microchip. Questa

[1] Per approfondimenti si rimanda il lettore ai documenti, in Bibliografia, [18] e [19].

sfida mette in risalto sempre più la voglia da parte delle case costruttrici di avere sul mercato il dispositivo con i consumi minori.

Alla luce di questa gara, prima di implementare il sistema di misura, dunque, si è voluta fare una ricerca tra i microcontrollori messi a disposizione sul mercato da alcune delle più importanti case costruttrici nel settore, fra le quali Freescale, Atmel e le già citate Texas Instruments e Microchip, al fine di trovare quello con i più bassi consumi energetici. La scelta si è basata solo ed esclusivamente sui dati riportati nei datasheet dei componenti presi in considerazione, senza effettuare alcuna prova sperimentale per verificare l'effettivo consumo dichiarato; operazione che invece verrà eseguita per il solo microcontrollore che da datasheet dichiara i minori consumi. Bisogna precisare che di ogni casa costruttrice è stato preso il prodotto "di punta" per quanto riguarda i consumi.

I microcontrollori considerati sono:

- MC9S08RG60 della Freescale;
- PIC24FJ128GA310 della Microchip;
- ATxmega64D3 della Atmel;
- MSP430FR5969 della Texas Instruments;

Si riporta, ora, in *Tab_2* un riepilogo che mette a confronto le diverse modalità di funzionamento e il relativo consumo dei microcontrollori presi in considerazione in questa fase; così facendo si potrà facilmente capire quale dispositivo ha il minor consumo energetico.

	Operatating Modes	DVcc [V]	Icc [µA]
MC9S08RG60	Active Mode (1MHz)	3.0	500
	Standby Mode	3.0	0.5
	Sleep Mode	3.0	0.1
PIC24FJ128GA310	Active Mode (1MHz)	3.3	150
	Standby Mode	3.3	0.34
	Sleep Mode	3.3	0.04
ATxmega64D3	Active Mode (2MHz)	3.0	114
	Standby Mode	3.0	1.2
	Sleep Mode	3.0	0.07
MSP430FR5969	Active Mode (1MHz)	3.0	110
	Standby Mode	3.0	0.4
	Sleep Mode	3.0	0.02

Tab_2 – Confronto tra microcontrollori del consumo energetico

Dalla tabella si nota che tutti i microcontrollori hanno consumi estremamente ridotti, ma tra questi è il dispositivo prodotto dalla Texas Instruments che riesce ad ottenere quelli più bassi.

2.1 IL MICROCONTROLLORE MSP430FR5969

Fin dagli anni ’90, la Texas Instruments ha cercato di affrontare le richieste dei sistemi integrati alimentati a batteria, introducendo sul mercato i microcontrollori MSP430 (*fig_2.2*). Questi ultimi nascono da un gruppo di progettazione della Texas Instruments di Freising, a nord di Monaco di Baviera e fin da subito hanno trovato spazio nelle applicazioni emergenti alimentate a batteria. [10]

fig_2.2 – Microcontrollore MSP430 [8]

Per avere un basso consumo di potenza elettrica, che oramai ha come punto di riferimento il passaggio dal microampere al nano ampere, si devono mettere in atto varie strategie, tra cui quella di realizzare microcontrollori aventi un'architettura (CPU, memoria, periferiche, moduli di clock) molto efficiente. Tenendo conto di questi fattori, di recente Texas Instruments ha introdotto sul mercato la serie di microcontrollori MSP430FRxxxx basati su tecnologia FRAM, con i quali riesce a garantire consumi ultraridotti, e che quindi si prestano in modo eccellente per le applicazioni embedded alimentate a batteria.
Durante il periodo di tesi si è deciso di caratterizzare il microcontrollore MSP430FR5969, in quanto è il dispositivo di punta attualmente sul mercato che garantisce affidabilità, elevate prestazioni e velocità con consumi estremamente bassi sia in modalità attiva ma soprattutto durante il periodo di inattività, il tutto a dei prezzi molto convenienti.

Il microcontrollore MSP430FR5969 Ultra-Low Power (ULP), in *fig_2.3*, combina in modo univoco una memoria integrata FRAM

e una architettura di sistema ultra-low power, consentendo di ottenere elevate prestazioni con basse richieste energetiche. Nel corso del capitolo si cercherà, allora, di spiegare come mai l'architettura di questo integrato è così efficiente, esponendo le innovazioni tecnologiche che la Texas Instruments ha introdotto nel mercato dei microcontrollori, in particolare l'utilizzo di una memoria FRAM, vero e proprio cambiamento in termini di consumi e prestazioni.

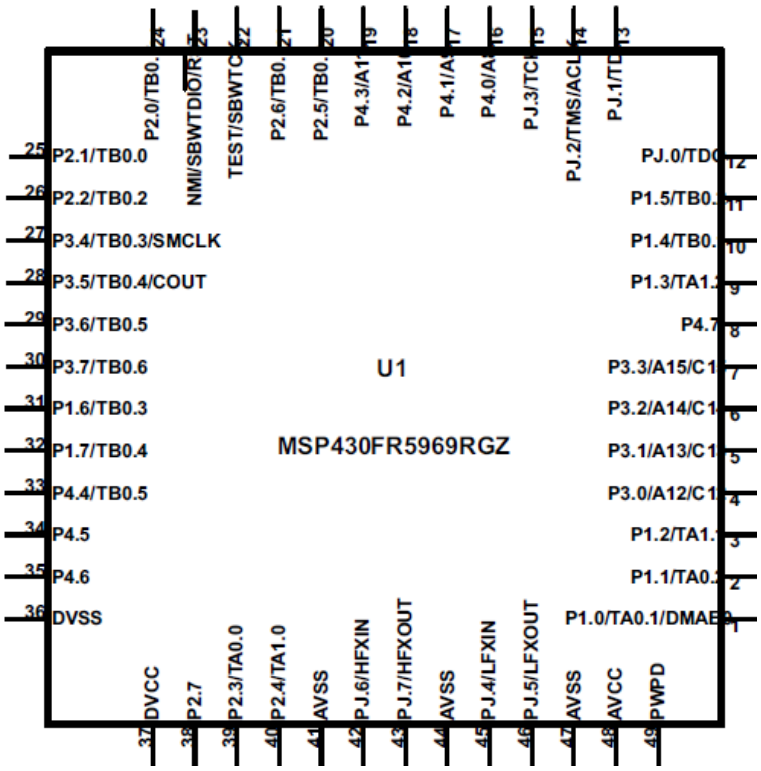


fig_2.3 – Configurazione dei pin del microcontrollore MSP430FR5969 [8]

2.1.1 ARCHITETTURA

Il dispositivo MSP430FR5969 dispone di un'architettura RISC a 16 bit secondo il modello di Von Neumann, il cui schema a blocchi è mostrato in *fig_2.4*, dove è possibile notare la presenza di un Master Data Bus (MDB) ed un bus

d'indirizzo Master Address Bus (MAB) attraverso i quali la CPU può dialogare con le tante periferiche integrate on chip, come cinque porte di I/O a 8 bit, diversi moduli di timer, tra qui il watchdog timer, un modulo di clock, nel quale sono presenti tre diversi sistemi di clock interni e due esterni, un banco di memoria FRAM a 64KB e uno SRAM a 2KB.

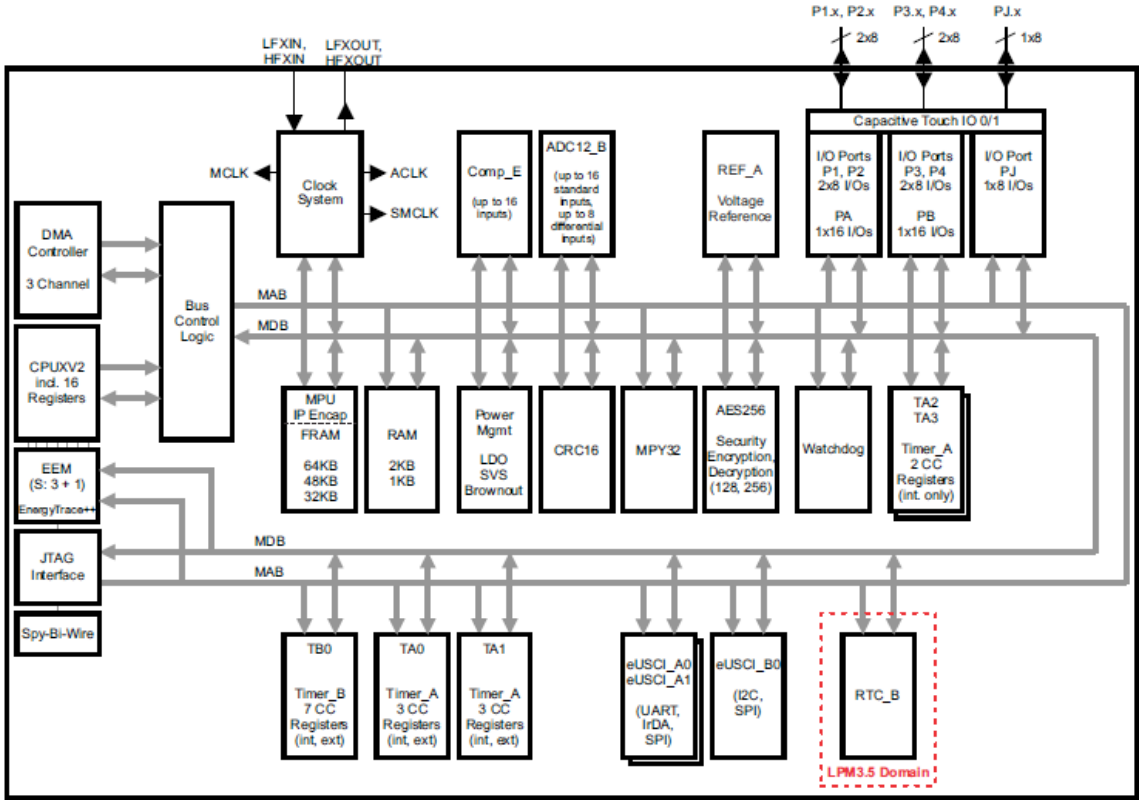


fig_2.4 – Schema a blocchi dell'architettura del microcontrollore MSP430FR5969 [8]

Il Datapath della CPU, in *fig_2.5*, dispone di 16 registri interni, citati da R0 a R15.

I registri da R4 a R15 sono per un utilizzo general purpose, mentre i restanti ad uso specifico. R0 rappresenta il Program Counter, R1 lo Stack Pointer, R3 il Costant Generator, ovvero un generatore di costanti (0,1,2,4,8) che

consente di diminuire la dimensione di un'istruzione, e R2 lo Status Register.

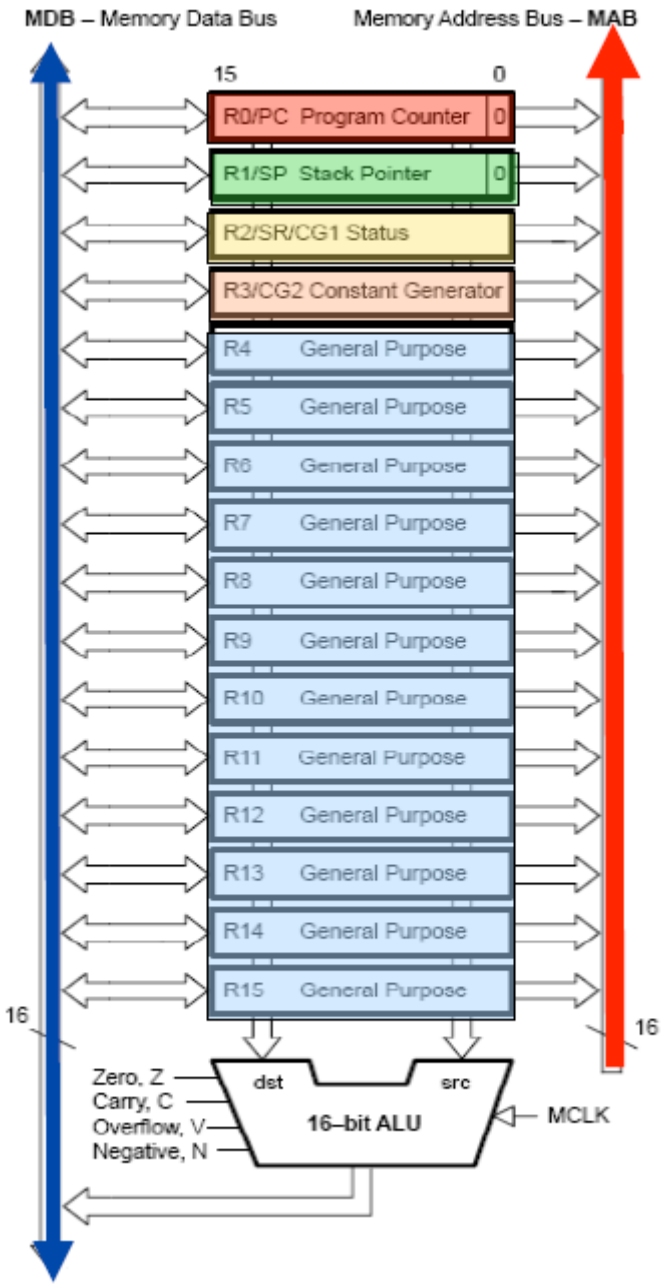


fig_2.5 – Datapath MSP430 [13]

In particolare, quest'ultimo registro ha una fondamentale importanza in quanto, oltre a possedere i flags classici di uno status register, come Z, N, C, dispone anche dei bit relativi alle modalità di basso consumo, che verranno

mostrati nel dettaglio nella sezione 2.1.4 relativo alle modalità di funzionamento.

2.1.2 MEMORIA FRAM

La memoria FRAM (Ferroelectric Random Access Memory) è una memoria con un processo costruttivo simile alla memoria DRAM: consiste, infatti, in una griglia di condensatori di accumulo ed una circuiteria associata per permettere la lettura e la scrittura dei dati nelle celle di memorizzazione. L'informazione è dunque associata alla carica accumulata nel condensatore; se questa carica viene persa, anche l'informazione è perduta. A differenza delle memorie DRAM, che sono volatili ed utilizzano condensatori allo stato solido, le memorie FRAM sono non-volatili, in quanto sfruttano, tra le armature del condensatore, un materiale ferroelettrico del quale viene variato lo stato di polarizzazione.

Questo aspetto è di fondamentale importanza se si prende in considerazione il differente comportamento di un materiale ferroelettrico rispetto ad un comune materiale dielettrico. Nel materiale dielettrico, applicando un campo elettrico esterno, le cariche elettriche presenti in esso assumono una configurazione diversa da quella iniziale, fino a formare un dipolo elettrico, per poi tornare nelle posizioni originali non appena il campo esterno termina d'esistere.
Nei materiali ferroelettrici, invece, è già presente una polarizzazione naturale, che dipende dal reticolo cristallino del materiale utilizzato; non appena viene applicato un campo esterno sufficientemente intenso, la polarizzazione può essere opportunamente direzionata e

rimane tale, o comunque varia di poco, qualora il campo esterno cessi (*fig_2.6*).

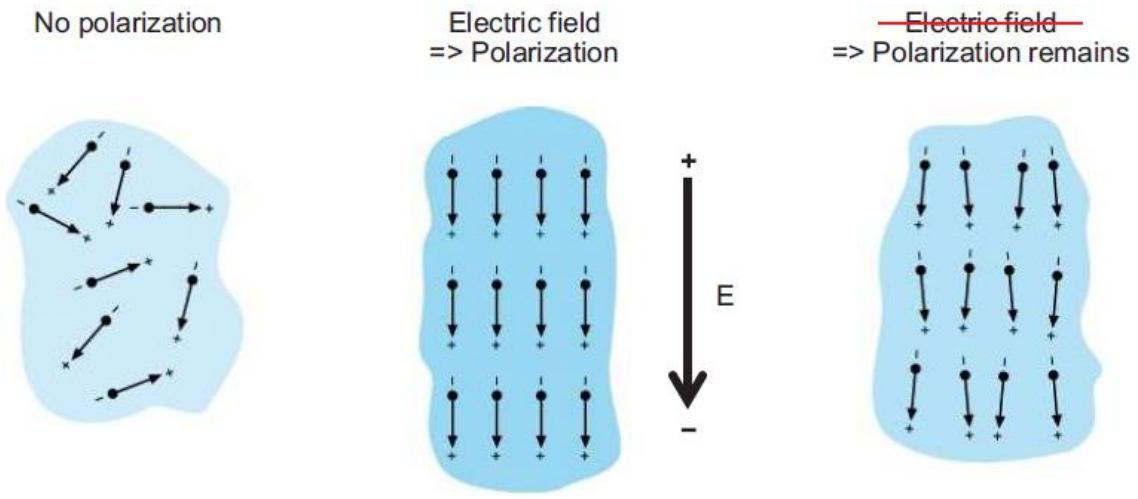


fig_2.6 – Polarizzazione in un materiale ferroelettrico [17]

Comunemente, come materiale ferroelettrico, viene utilizzato un PZT. Il piombo-zirconato di titanio (PZT) è un composto inorganico intermetallico il cui reticolo cristallino, in *fig_2.7*, associato alla formula $Pb[Zr_xTi_{1-x}]O_3$, è caratterizzato dall'avere al centro alternativamente un atomo di Titanio (Ti) o Zirconio (Zi).

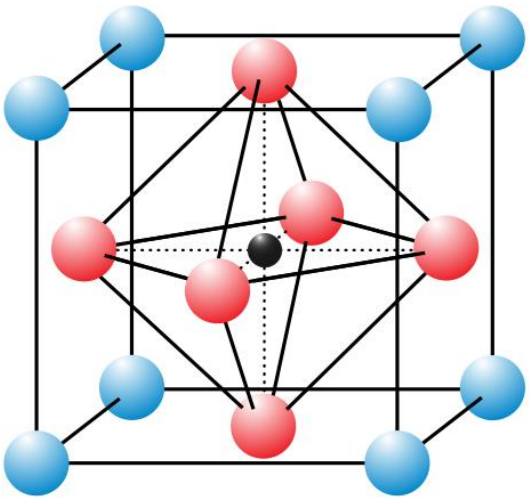
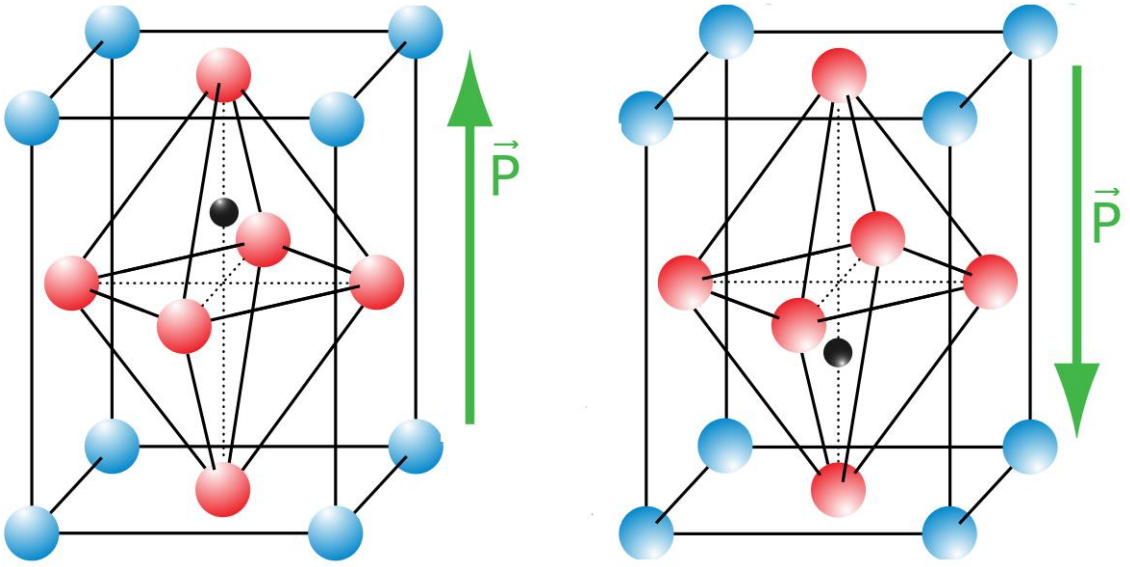


fig_2.7 – Reticolo cristallino del PZT [3]

In *fig_2.8*, viene mostrato come la presenza di un campo elettrico esterno sposti l'atomo centrale del reticolo verso una nuova posizione stabile. A seconda che il campo sia positivo o negativo, l'atomo si sposterà verso l'alto del reticolo o verso il basso, polarizzando il cristallo, rispettivamente, positivamente o negativamente.



a) *b)*

fig_2.8 – Polarizzazione elettrica di un materiale ferroelettrico. Nel caso a) si applica un campo elettrico positivo e la polarizzazione risulta positiva. Viceversa nel caso b) [3]

Quindi, la polarizzazione del materiale ferroelettrico, comporta la presenza di dipoli elettrici non nulli nel dielettrico del condensatore di accumulo.
In *fig_2.9* viene riportato il loop di isteresi, che mostra la relazione tra la carica accumulata ai capi del condensatore ferroelettrico e la tensione applicata ai capi di quest'ultimo.

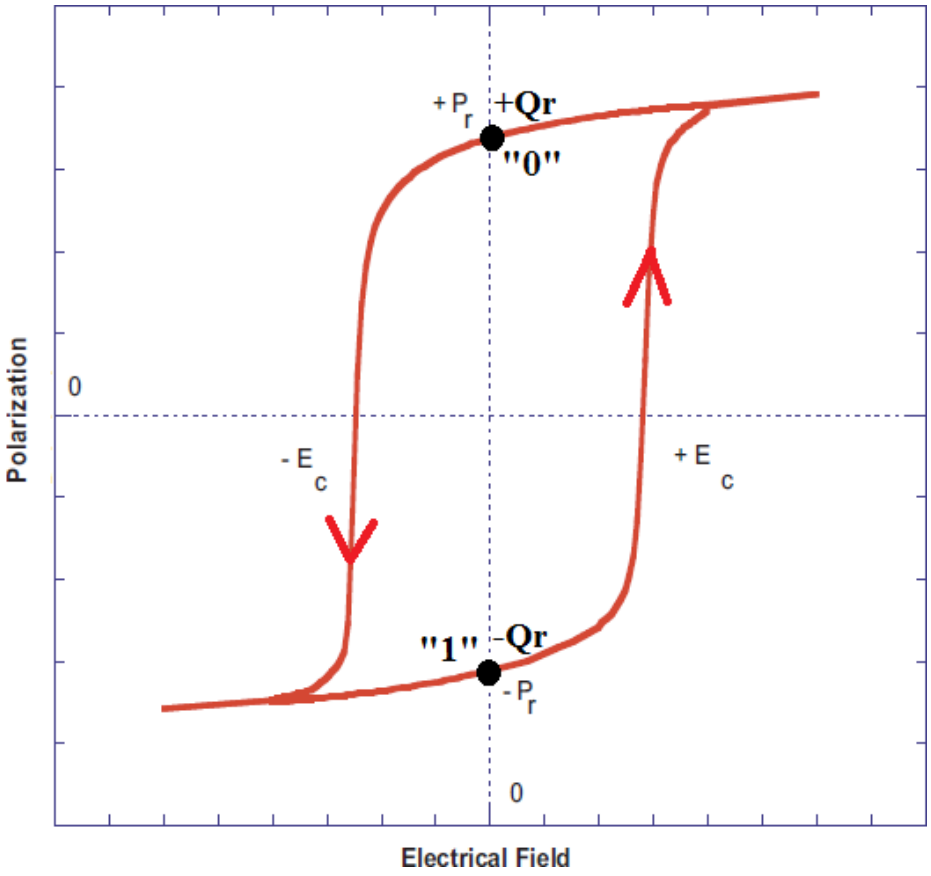


fig_2.9 – Loop di isteresi della polarizzazione del condensatore ferroelettrico [17]

Dal grafico si intuisce che la carica accumulata sul condensatore [+Qr] o [-Qr] si conserva anche in assenza di campo elettrico, il che si traduce in una permanenza del bit logico "0" o "1" anche una volta annullata la tensione ai capi del condensatore. Per passare da uno stato stabile all'altro (ad esempio "0" → "1") è necessario applicare esternamente al capacitore una tensione inversa, in modo tale che anche la polarizzazione del materiale ferroelettrico tra le armature risulti invertita.

2.1.2.1 Scrittura e lettura di una cella FRAM

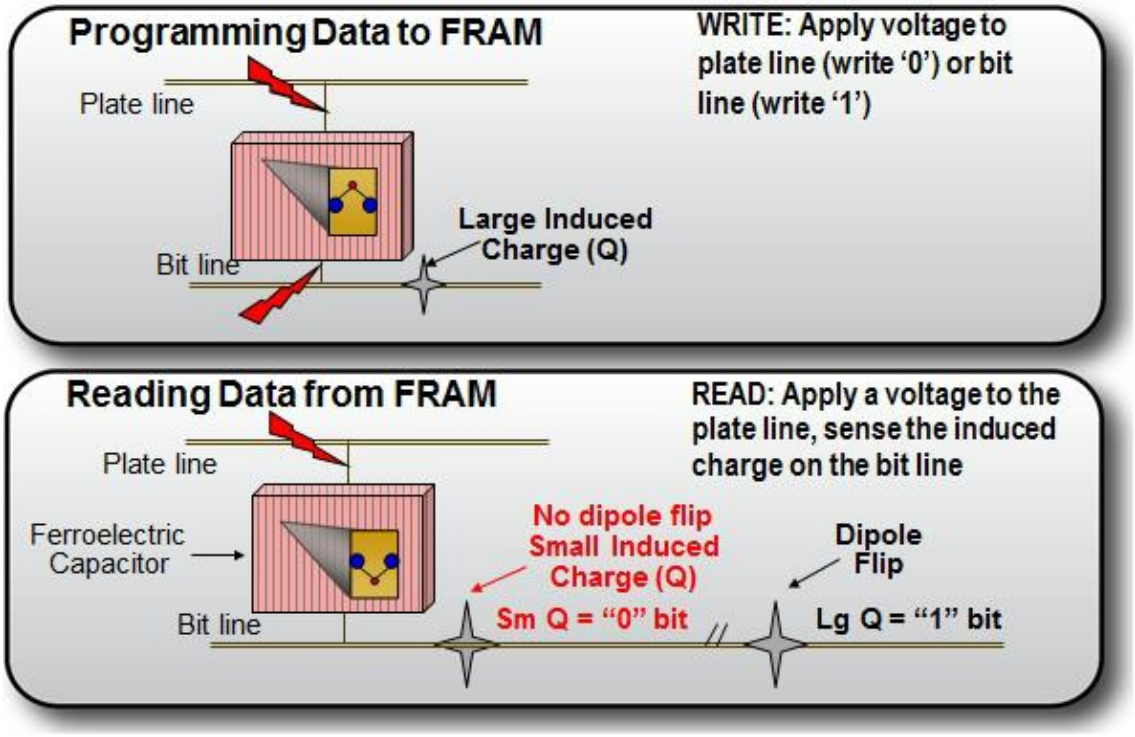


fig_2.10 – Scrittura e lettura della cella FRAM [16]

Come si può notare dalla *fig_2.10*, il condensatore ferroelettrico è connesso alle estremità alla linea di Plateline (PL), grazie alla quale si scrive/legge la singola cella, e alla linea di Bitline (BL).

Scrivere un "1" o uno "0" all'interno della memoria richiede la sola polarizzazione del cristallo in una delle due direzioni; per farlo si applica un voltaggio, quindi un campo elettrico, ai capi del materiale ferroelettrico in fase di programmazione. In particolare, qualora si voglia memorizzare un valore logico alto, si applica una tensione (V_{CC} circa 1.5V) di pari ampiezza sia sulla BL che sulla PL. Se invece si vuole salvare il valore logico basso si mantiene la BL a 0V mentre sulla PL si applica un impulso di tensione.

In realtà, però, è necessaria anche un'altra condizione per eseguire la scrittura/lettura della cella. Infatti, quest'ultima, oltre alla presenza del condensatore, prevede anche un transistore, tipicamente nMOS, collegato ad un'altra linea, chiamata Wordline (WL) che permette l'accesso dei livelli di tensione della linea BL al condensatore (*fig_2.11*).

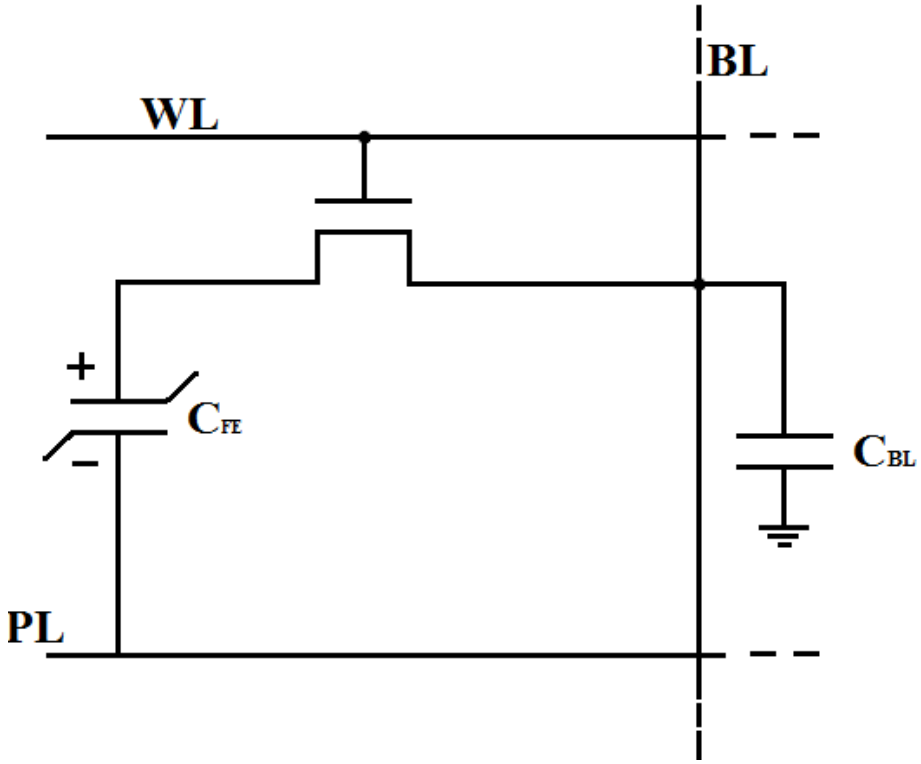


fig_2.11 – Cella di memoria FRAM

L'operazione di lettura del dato memorizzato all'interno della cella, invece, avviene mediante un sense amplifier, che confronta la tensione sulla BL a cui è connesso il condensatore e una tensione di riferimento. Dopo aver rimosso il condensatore dallo stato di isolamento, viene applicata una tensione sulla PL, come durante la fase di scrittura. A questo punto, il condensatore, a seconda dello stato logico memorizzato, possiede una carica differente, e quindi la BL a cui è connesso, possiederà due differenti

valori (V_H→"1", V_L→"0"). Il sense amplifier, proseguendo, porta la BL a V_{CC} o 0V a seconda che la tensione precedentemente rilevata sia V_H o V_L. Procedendo in questo modo, si deduce come l'operazione di lettura porta alla perdita del dato memorizzato. Proprio per questo, la fase di lettura termina sempre con un "re-store" dello stato logico a cui era il condensatore prima dell'operazione.

2.1.2.2 Vantaggi ed innovazione

Sebbene il mercato dei sistemi embedded si basa ancora molto sulle memorie Flash, l'introduzione nei microcontrollori di una memoria FRAM ha portato sicuramente moltissimi vantaggi. Soprattutto al giorno d'oggi dove si richiedono dispositivi in grado di lavorare ad elevate velocità e per lunghi periodi di tempo, il tutto limitando i consumi energetici ed i costi di produzione.

La principale innovazione è che la memoria FRAM si presenta come una "**memoria unificata**" (*fig_2.12*). Sostanzialmente, in un unico spazio di memoria vengono allocati sia il programma che i dati, ma con la possibilità di partizionare in modo flessibile la memoria stessa a seconda delle esigenze dell'applicazione. Infatti, è permesso, via software, decidere se dedicare più spazio per memorizzare i dati piuttosto che per il programma.

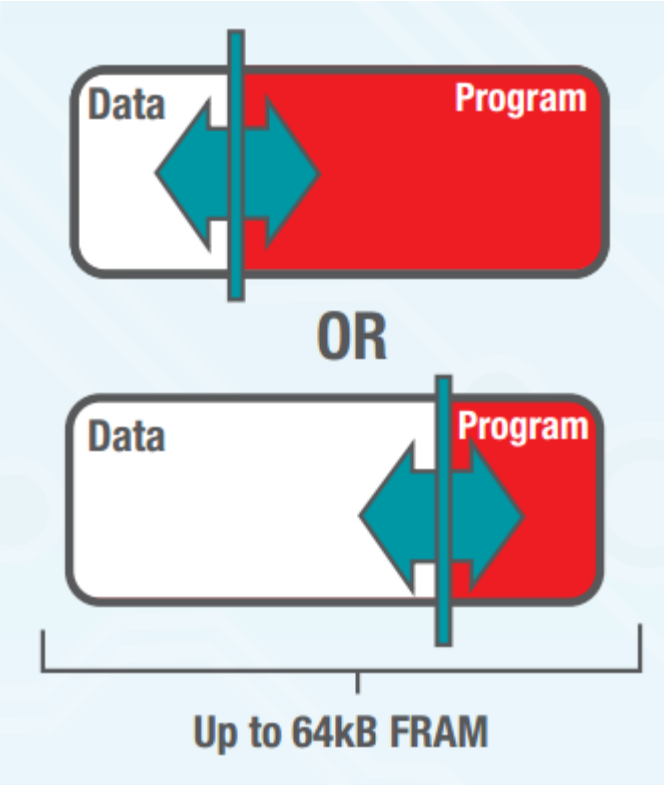


fig_2.12 – Memoria unificata [8]

La memoria FRAM, quindi, si può vedere come un'unica memoria che combina la velocità, i bassi consumi, la resistenza ai cicli di scrittura e la flessibilità di una SRAM e l'affidabilità e stabilità di una Flash.
Infatti, i principali vantaggi (*fig_2.13*) introdotti con questa memoria non-volatile sono:

- **Velocità:** per quanto riguarda il ciclo di lettura è paragonabile con quella di una Flash. Il ciclo di scrittura, invece, risulta essere circa 100volte più veloce, in quanto si basa, come detto in precedenza, sul movimento meccanico degli atomi reagendo ad un campo elettrico esterno; movimento, dunque, estremamente veloce [3]. Nelle Flash, invece, si deve utilizzare una pompa di carico on-chip che richiede un tempo notevole, per raggiungere il potenziale necessario (qualche ms), oltre al tempo necessario per

collocare il giusto quantitativo di carica elettrica sui floating gate dell'array.
Rispetto ad una DRAM, invece, i tempi sia di lettura sia di scrittura si possono considerare simili.

- **Consumo:** il principale vantaggio della memoria FRAM rispetto alle DRAM o Flash è proprio quello di avere consumi energetici molto bassi. I consumi che si intendono sono soprattutto quello dovuti alla scrittura e lettura delle celle di memoria. Nelle DRAM la carica immagazzinata nei condensatori di accumulo svanisce in breve tempo disperdendosi attraverso il non perfetto strato di isolamento e il transistore di accesso. Per questo motivo richiedono il "refresh" della cella, ovvero una continua lettura e riscrittura di quest'ultima. Questa operazione deve essere eseguita svariate volte al secondo e ciò richiede una presenza continua dell'alimentazione, il che comporta indubbiamente degli elevati consumi di potenza. Nelle FRAM, invece, è necessario applicare una tensione solamente al momento della scrittura o della lettura della cella. Riguardo alle memorie Flash, invece, si hanno consumi minori soprattutto durante la fase di scrittura. Infatti, una cella Flash per essere scritta ha bisogno di una alimentazione di supporto di circa 5V mentre una cella FRAM necessita di una tensione 3 volte più bassa (circa 1.5V).
  Per questo le memorie FRAM hanno un consumo circa 1/3 più basso rispetto alle Flash.

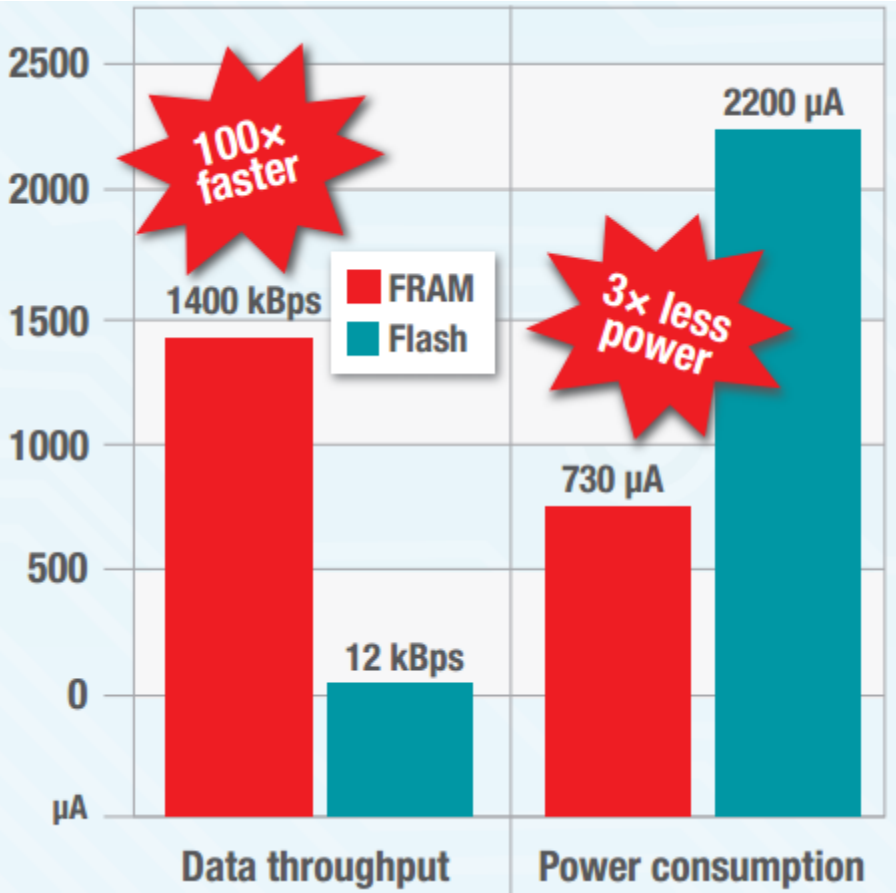


fig_2.13 – Velocità e Consumi: differenza tra memoria FRAM e Flash [8]

- **Durata dei cicli di scrittura:** un altro importante elemento che caratterizza una memoria è dato dal numero di cicli di scrittura che la memoria riesce a sostenere. Di conseguenza questo si ripercuote anche sulla vita del sistema integrato. Per quanto riguarda una memoria Flash, il numero di scritture che questa supporta, seppur molto alto (circa 10^4 cicli di scrittura), non è illimitato, come nelle memorie DRAM e SRAM [17]. Le memorie FRAM, al contrario, offrono la possibilità di eseguire 100 trilioni di cicli di scrittura (10^{14}) [17], come mostrato in *fig_2.14*, garantendo una durata praticamente illimitata della memoria.

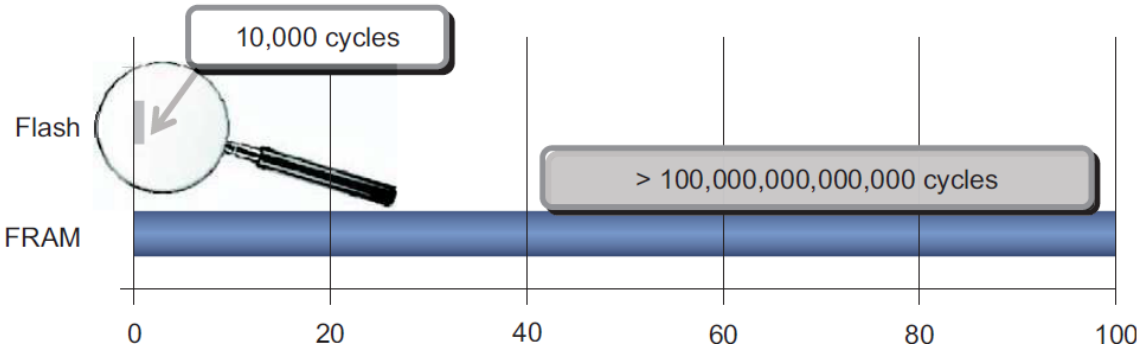


fig_2.14 – Confronto sulla durata dei cicli di scrittura tra la memoria Flash e la memoria FRAM [17]

Questo vantaggio si ripercuote indubbiamente anche sulla vita di un sistema integrato. Come mostrato in *fig_2.15*, se ad esempio si esegue un ciclo di scrittura ogni secondo, il sistema a memoria Flash ha un tempo di vita di circa tre ore, mentre quello basato su memoria FRAM raggiungerebbe, teoricamente, un tempo di vita superiore ai tre milioni di anni.

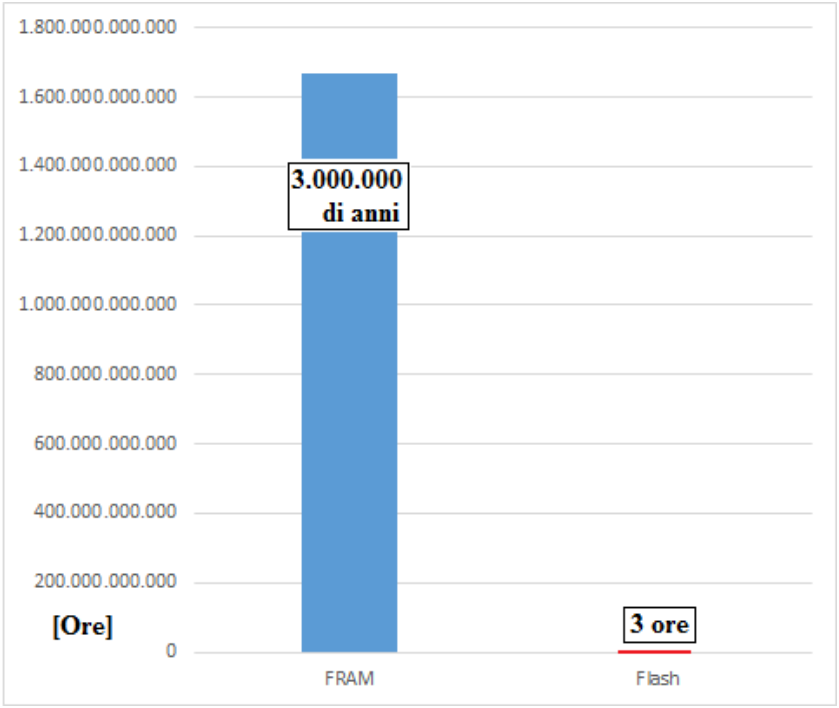


fig_2.15 – Confronto sul tempo di vita di sistemi basati su memoria FRAM e Flash

2.1.3 SISTEMA DI CLOCK

Come accennato nella sezione 1.1.3, un ruolo fondamentale per il funzionamento dei microcontrollori è dovuto alla frequenza di clock generata dagli oscillatori, la quale scandisce le istruzioni interne che la CPU deve svolgere.

Nei sistemi Ultra Low power, spesso si hanno esigenze molto contrastanti da soddisfare. Infatti, da una parte si vuole una velocità di elaborazione molto rapida (frequenza elevata), ma dall'altra ottenere consumi estremamente ridotti (bassa frequenza di clock). Quindi, nei più recenti microcontrollori, tra cui quello preso in esame, si è introdotto, al fine di ottenere un giusto compromesso, un sistema di clock denominato **Unified Clock System** (UCS). L'architettura di questo sistema nel microcontrollore MSP430FR5969, mostrata in *fig_2.16*, è stata progettata, oltre chiaramente ad essere molto efficiente dal punto di vista energetico, per essere semplice e molto flessibile per l'utente. Il sistema prevede 3 oscillatori interni (VLO, DCO e MODOSC), 2 esterni (LFXT e HFXT) e 3 uscite (ACLK, MCLK e SMCLK).

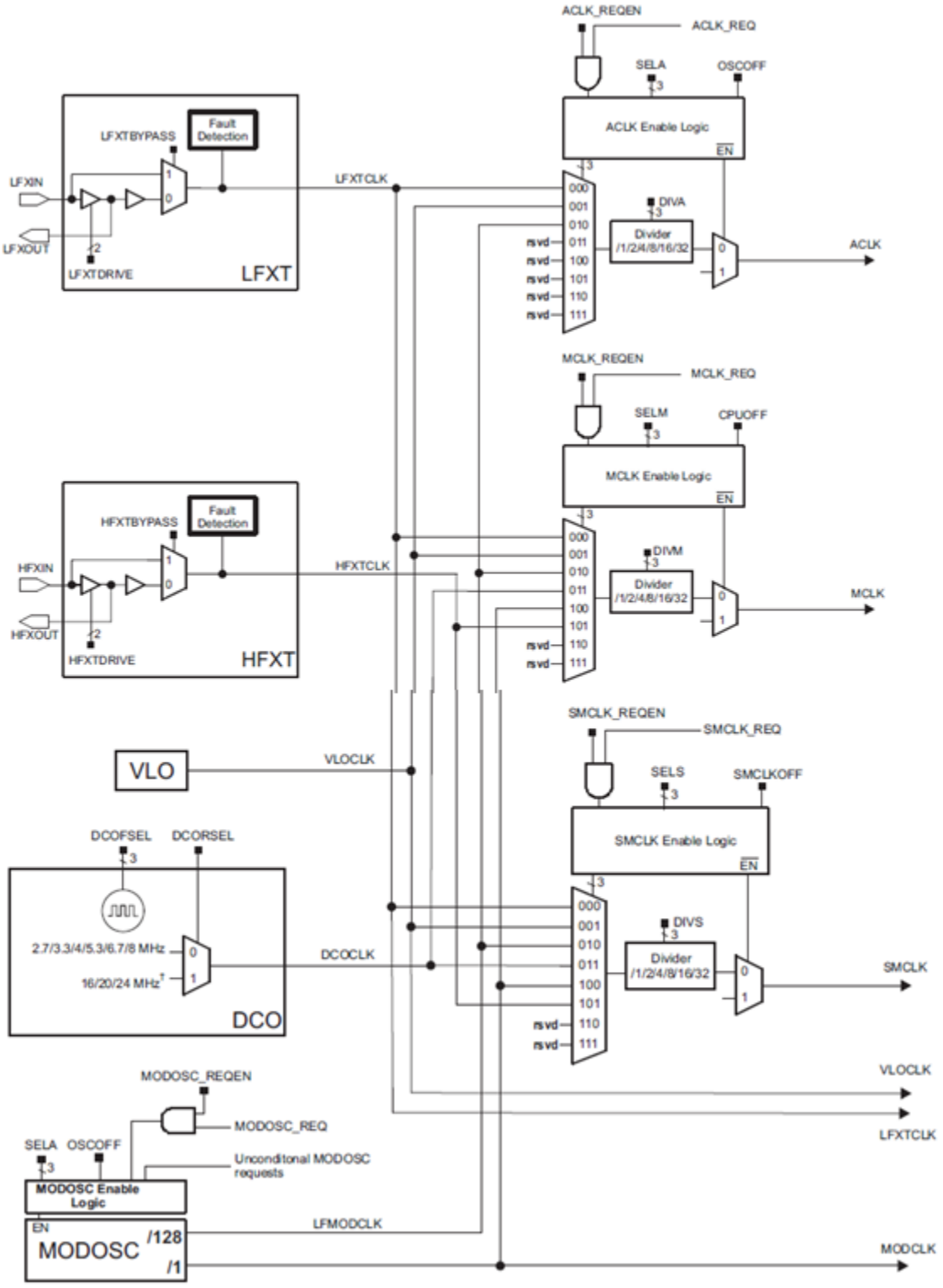


fig_2.16 – Unified Clock System: schema a blocchi [8]

Il sistema di clock, come mostrato nella figura sopra, presenta le seguenti sorgenti di clock:

- Very Low Oscillator (VLO): è un oscillatore, interno al microcontrollore, con un valore tipico di frequenza di 9.4KHz in un range di 6KHz – 14KHz. Seppur non molto preciso visto l'ampia fascia di funzionamento, ha il notevole vantaggio di avere un consumo estremamente ridotto. Infatti, in una modalità di funzionamento Ultra Low power del microcontrollore in cui questo oscillatore rimane attivo, è possibile avere consumi nominali del micro di 400nA;
- Digital Controlled Oscillator (DCO): anch'esso è un oscillatore interno, ma a differenza di quello precedentemente descritto, ha la possibilità di raggiungere ben più elevate frequenze (fino ai 24MHz nel MSP430FR5969). Inoltre ha la caratteristica di poter variare dinamicamente, attraverso dei registri dedicati modificabili in fase di programmazione, la sua frequenza di funzionamento (solitamente 1MHz, 4MHz, 8MHz, 12MHz, 16MHz e 24MHz). Ovviamente, lavorando a più elevate frequenze, questo oscillatore porta il microcontrollore ad avere consumi maggiori rispetto al caso precedente (ad esempio ad 1MHz il consumo è di circa 110μA);
- Module Oscillator (MODOSC): come i precedenti è un oscillatore interno che fornisce tipicamente una frequenza di 5MHz. Viene solitamente impiegato come sorgente di clock negli ADC integrati del microcontrollore.
- LFXT: è un oscillatore a bassa frequenza pilotabile o mediante sorgenti di clock esterne a 50KHz o meno, o con oscillatori al quarzo esterni a 32KHz.
- HFXT: è sempre un oscillatore esterno, ma ad alta frequenza. Infatti, possono essere utilizzati

oscillatori esterni con un range di frequenza 4MHz – 24MHz.

Il sistema di clock, vista la sua flessibilità, permette di impostare le sorgenti di clock appena descritte su ognuna delle tre uscite potendo così gestire anche i consumi. In particolare, queste ultime sono:

- ACLK (Auxiliary Clock), utilizzata solitamente per le sorgenti VLO e LFXT a bassa frequenza;
- MCLK (Master Clock), adoperata normalmente per dettare le istruzioni della CPU;
- SMCLK (Sub System Master Clock), impiegata per la gestione delle periferiche.

Si vuole ulteriormente sottolineare, infine, che la presenza di diverse tipologie di sorgenti di clock, soprattutto quelle interne, e la notevole flessibilità del sistema UCS, permettono, al programmatore del microcontrollore, di selezionare il miglior equilibrio tra prestazioni e consumi di energia. Questo rappresenta, quindi, un notevole vantaggio per i sistemi embedded di ultima generazione.

2.1.4 MODALITA' DI FUNZIONAMENTO

Più volte durante l'elaborato è stato affermato che un microcontrollore è un dispositivo il cui compito è svolgere una serie di istruzioni (programma) contenute nella sua memoria al fine di pilotare un sistema integrato specifico. Nel mercato odierno sono sempre più le applicazioni dove le operazioni sono compiute molto velocemente, tanto che la CPU alle volte si trova ad attendere, senza far nulla, le istruzioni successive da eseguire. È evidente, essendo la CPU la maggior causa di consumo in un sistema a

microcontrollore, che non è conveniente mantenerla attiva mentre non deve fare nulla; soprattutto nelle recenti applicazioni a batteria, dove si punta ad eliminare tutti i consumi possibili al fine di allungare la vita del sistema. Risulta, allora, più ragionevole spegnere la CPU, mandandola in uno stato detto di "Sleep", durante tutto il tempo di inattività. In questo modo sarà possibile limitare la corrente assorbita dalla CPU e quindi minimizzare anche l'energia utilizzata dal sistema.

Il microcontrollore MSP430FR5969 è stato ottimizzato nella sua architettura per poter lavorare in sette differenti modalità a basso e sono state nominate LPM0, LPM1, LPM2, LPM3, LPM4, LPM3.5 e LPM4.5, rispettivamente in ordine crescente di quanto è il risparmio energetico.
Le diverse modalità di funzionamento sono configurabili settando alcuni dei bit presenti nello Status Register (*fig_2.17*) della CPU.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C
rw-0							rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Bit	Description	
V	Overflow bit. This bit is set when the result of an arithmetic operation overflows the signed-variable range.	
	ADD(.B),ADDC(.B)	Set when: Positive + Positive = Negative Negative + Negative = Positive Otherwise reset
	SUB(.B),SUBC(.B),CMP(.B)	Set when: Positive – Negative = Negative Negative – Positive = Positive Otherwise reset
SCG1	System clock generator 1. When set, turns off the SMCLK.	
SCG0	System clock generator 0. When set, turns off the DCO dc generator, if DCOCLK is not used for MCLK or SMCLK.	
OSCOFF	Oscillator Off. When set, turns off the LFXT1 crystal oscillator, when LFXT1CLK is not use for MCLK or SMCLK.	
CPUOFF	CPU off. When set, turns off the CPU.	
GIE	General interrupt enable. When set, enables maskable interrupts. When reset, all maskable interrupts are disabled.	
N	Negative bit. Set when the result of a byte or word operation is negative and cleared when the result is not negative. Word operation: N is set to the value of bit 15 of the result. Byte operation: N is set to the value of bit 7 of the result.	
Z	Zero bit. Set when the result of a byte or word operation is 0 and cleared when the result is not 0.	
C	Carry bit. Set when the result of a byte or word operation produced a carry and cleared when no carry occurred.	

fig_2.17 – Status Register MSP430

In particolare, i 4 bit che permettono il passaggio da una modalità ad un'altra sono:

- **CPUOFF:** quando settato spegne la CPU;
- **OSCOFF:** quando settato disabilita l'oscillatore LFXT per l'uscita ACLK;
- **SCG0:** quando settato disabilita l'oscillatore DCO se questo non è utilizzato per il MCLK o SMCLK;
- **SCG1:** quando settato spegne il SMCLK.

In *fig_2.18 e fig_2.19* sono riportate una rappresentazione grafica ed una tabella delle varie modalità di funzionamento con le relative combinazioni dei bit dello Status Register.

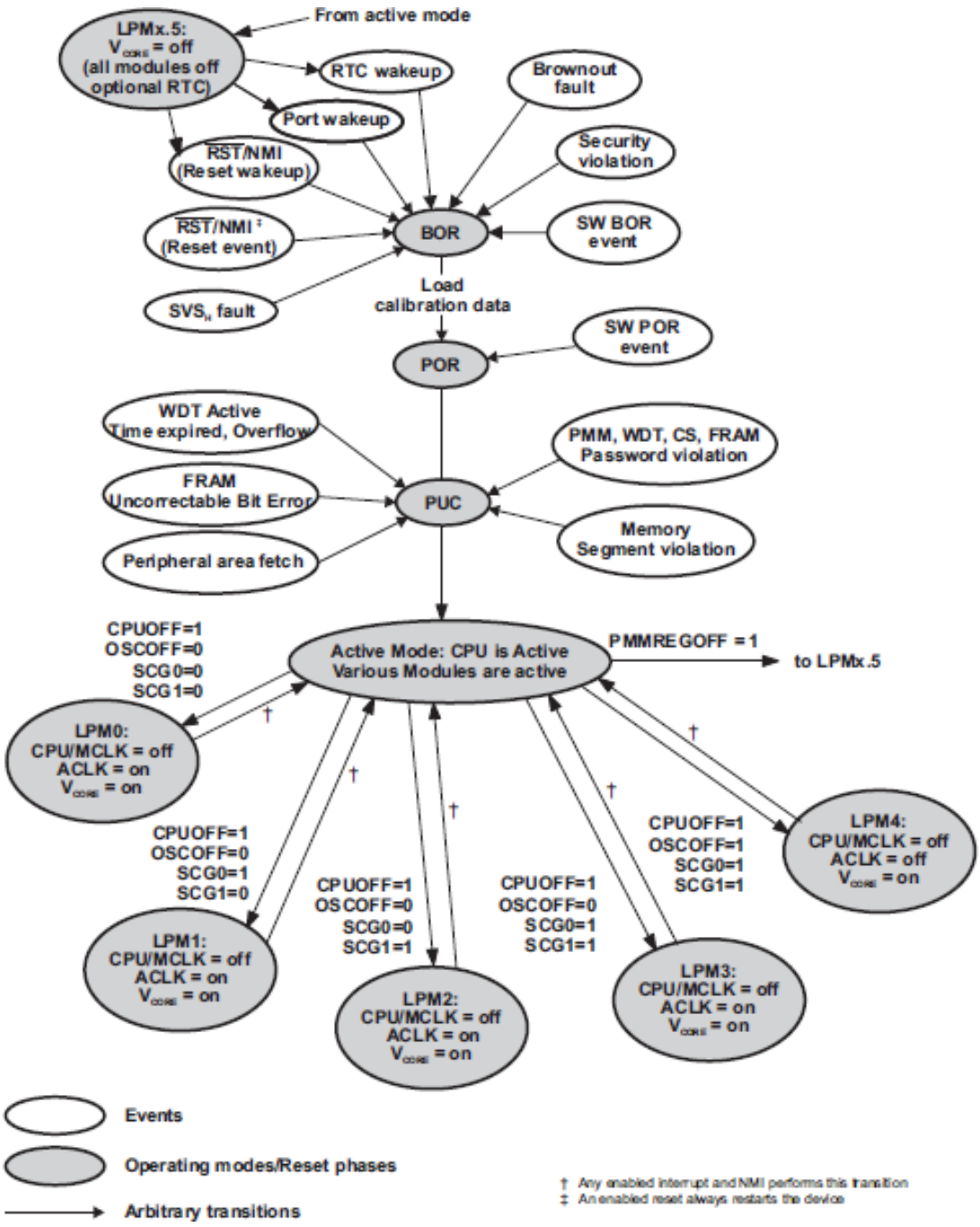


fig_2.18 – Rappresentazione grafica delle varie modalità di funzionamento e relative configurazioni [8]

SCG1[1]	SCG0	OSCOFF[1]	CPUOFF[1]	Mode	CPU and Clocks Status[2]
0	0	0	0	Active	CPU, MCLK are active. ACLK is active. SMCLK optionally active (SMCLKOFF = 0). DCO is enabled if sources ACLK, MCLK, or SMCLK (SMCLKOFF = 0). DCO bias is enabled if DCO is enabled or DCO sources MCLK or SMCLK (SMCLKOFF = 0).
0	0	0	1	LPM0	CPU, MCLK are disabled. ACLK is active. SMCLK optionally active (SMCLKOFF = 0). DCO is enabled if sources ACLK or SMCLK (SMCLKOFF = 0). DCO bias is enabled if DCO is enabled or DCO sources MCLK or SMCLK (SMCLKOFF = 0).
0	1	0	1	LPM1	CPU, MCLK are disabled. ACLK is active. SMCLK optionally active (SMCLKOFF = 0). DCO is enabled if sources ACLK or SMCLK (SMCLKOFF = 0). DCO bias is enabled if DCO is enabled or DCO sources MCLK or SMCLK (SMCLKOFF = 0).
1	0	0	1	LPM2	CPU, MCLK are disabled. ACLK is active. SMCLK is disabled.
1	1	0	1	LPM3	CPU, MCLK are disabled. ACLK is active. SMCLK is disabled.
1	1	1	1	LPM4	CPU and all clocks are disabled.
1	1	1	1	LPM3.5	When PMMREGOFF = 1, regulator is disabled. No memory retention. In this mode, RTC operation is possible when configured properly. See the *RTC* module for further details.
1	1	1	1	LPM4.5	When PMMREGOFF = 1, regulator is disabled. No memory retention. In this mode, all clock sources are disabled; that is, no RTC operation is possible.

(1) This bit is automatically reset when exiting low-power modes.
(2) The low-power modes and, hence, the system clocks can be affected by the clock request system.

fig_2.19 – Tabella riassuntiva delle varie modalità di funzionamento e relative configurazioni [8]

Osservando attentamente la tabella in *fig_2.19* si può notare che settando i bit dello Status Register si vanno a disattivare, oltre alla CPU, anche le sorgenti di clock. Infatti, anche queste ultime sono una principale fonte di consumo.

Di seguito, in *fig_2.20*, *fig_2.21*, *fig_2.22* e *fig_2.23*, si riportano i consumi relativi alle varie modalità dichiarati nel datasheet del microcontrollore in esame:

PARAMETER	EXECUTION MEMORY	V_{CC}	FREQUENCY ($f_{MCLK} = f_{SMCLK}$)										UNIT
			1 MHz 0 wait states (NWAITSx = 0)		4 MHz 0 wait states (NWAITSx = 0)		8 MHz 0 wait states (NWAITSx = 0)		12 MHz 1 wait states (NWAITSx = 1)		16 MHz 1 wait states (NWAITSx = 1)		
			TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	
$I_{AM, FRAM\_UNI}$ (Unified memory)(3)	FRAM	3.0 V	210		640		1220		1475		1845		µA
$I_{AM, FRAM}$(0%)(4) (5)	FRAM 0% cache hit ratio	3.0 V	370		1280		2510		2080		2650		µA
$I_{AM, FRAM}$(50%)(4) (5)	FRAM 50% cache hit ratio	3.0 V	240		745		1440		1575		1990		µA
$I_{AM, FRAM}$(66%)(4) (5)	FRAM 66% cache hit ratio	3.0 V	200		560		1070		1300		1620		µA
$I_{AM, FRAM}$(75%)(4) (5)	FRAM 75% cache hit ratio	3.0 V	170	255	480		890	1085	1155	1310	1420	1620	µA
$I_{AM, FRAM}$(100%(4) (5)	FRAM 100% cache hit ratio	3.0 V	110		235		420		640		730		µA
$I_{AM, RAM}$ (6)	RAM	3.0 V	130		320		585		890		1070		µA
$I_{AM, RAM\ only}$ (7) (5)	RAM	3.0 V	100	180	290		555		860		1040	1300	µA

fig_2.20 – Modalità attiva [8]

PARAMETER	V_{CC}	FREQUENCY (f_{SMCLK})										UNIT
		1 MHz		4 MHz		8 MHz		12 MHz		16 MHz		
		TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	
I_{LPM0}	2.2 V	70		95		150		250		215		µA
	3.0 V	80	115	105		160		260		225	260	
I_{LPM1}	2.2 V	35		60		115		215		180		µA
	3.0 V	35	60	60		115		215		180	205	

fig_2.21 – Modalità LPM0 e LPM1 [8]

PARAMETER		V_{CC}	-40 °C		25 °C		60 °C		85°C		UNIT
			TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	
$I_{LPM2,XT12}$	Low-power mode 2, 12-pF crystal(2) (3) (4)	2.2 V	0.5		0.9		2.2		6.1		µA
		3.0 V	0.5		0.9	1.8	2.2		6.1	17	
$I_{LPM2,XT3.7}$	Low-power mode 2, 3.7-pF cyrstal(2) (5) (4)	2.2 V	0.5		0.9		2.2		6.0		µA
		3.0 V	0.5		0.9		2.2		6.0		
$I_{LPM2,VLO}$	Low-power mode 2, VLO, includes SVS(6)	2.2 V	0.3		0.7		1.9		5.8		µA
		3.0 V	0.3		0.7	1.6	1.9		5.8	16.7	
$I_{LPM3,XT12}$	Low-power mode 3, 12-pF crystal, includes SVS(2) (3) (7)	2.2 V	0.5		0.6		0.9		1.85		µA
		3.0 V	0.5		0.6	0.9	0.9		1.85	4.9	
$I_{LPM3,XT3.7}$	Low-power mode 3, 3.7pF cyrstal, excludes SVS(2) (5) (8) (also refer to Figure 5-2)	2.2 V	0.4		0.5		0.8		1.7		µA
		3.0 V	0.4		0.5		0.8		1.7		
$I_{LPM3,VLO}$	Low-power mode 3, VLO, excludes SVS(9)	2.2 V	0.3		0.4		0.7		1.6		µA
		3.0 V	0.3		0.4	0.7	0.7		1.6	4.7	
$I_{LPM4,SVS}$	Low-power mode 4, includes SVS(10) (also refer to Figure 5-3)	2.2 V	0.4		0.5		0.8		1.7		µA
		3.0 V	0.4		0.5	0.8	0.8		1.7	4.8	
I_{LPM4}	Low-power mode 4, excludes SVS(11)	2.2 V	0.2		0.3		0.6		1.5		µA
		3.0 V	0.2		0.3	0.6	0.6		1.5	4.6	

fig_2.22 – Modalità LPM2, LPM3 e LPM4 [8]

PARAMETER		V_{CC}	-40 °C		25 °C		60 °C		85°C		UNIT
			TYP	MAX	TYP	MAX	TYP	MAX	TYP	MAX	
$I_{LPM3.5,XT12}$	Low-power mode 3.5, 12-pF crystal, includes SVS(2)(3)(4)	2.2 V	0.4		0.45		0.5		0.7		µA
		3.0 V	0.4		0.45	0.7	0.5		0.7	1.2	
$I_{LPM3.5,XT3.7}$	Low-power mode 3.5, 3.7-pF cyrstal, excludes SVS(2)(5)(6) (also refer to Figure 5-4)	2.2 V	0.2		0.25		0.3		0.45		µA
		3.0 V	0.2		0.25		0.3		0.5		
$I_{LPM4.5,SVS}$	Low-power mode 4.5, includes SVS(7) (also refer to Figure 5-5)	2.2 V	0.2		0.2		0.2		0.3		µA
		3.0 V	0.2		0.2	0.4	0.2		0.3	0.55	
$I_{LPM4.5}$	Low-power mode 4.5, excludes SVS(8) (also refer to Figure 5-5)	2.2 V	0.02		0.02		0.02		0.08		µA
		3.0 V	0.02		0.02		0.02		0.08	0.35	

fig_2.23 – Modalità Ultra-Low Power LPM3.5 e LPM4.5 [8]

Dalle tabelle di *fig_2.20* e *fig_2.21* si può notare la dipendenza della corrente dalla frequenza. Infatti, più è elevata la velocità di elaborazione, più i consumi energetici si alzano. È possibile osservare, inoltre, la sostanziale differenza di consumi tra le modalità descritte nelle prime due tabelle e quelle nelle successive due. Questo perché nelle modalità attiva, LPM0 e LPM1 sono ancora attive le sorgenti ad elevata frequenza (MCLK e SMCLK). Nelle modalità LPM2 e LPM3 è possibile avere la sola sorgente a bassa frequenza (ACLK), cosa che non si ha, invece, nella modalità LPM4, LPM3.5 e LPM4.5. Questo rappresenta un notevole vantaggio, in quanto si ha la possibilità di gestire timer, contatori, etc… pur mantenendo bassissimi i consumi (solo 0.4µA nella modalità LPM3, paragonabile quasi con la modalità LPM4).
Confrontando le varie tabelle, le modalità con il minor consumo sono la LPM3.5 e LPM4.5. Queste modalità .5 differiscono dalle altre perché vanno a disattivare, oltre ai 4 bit dello Status Register, il Power Management Module (PMM). All'interno di questo modulo, mostrato in *fig_2.24*, si preleva la tensione di alimentazione primaria del micro (DV_{CC}) e, mediante un regolatore di tensione (LDO) interno, si genera una tensione secondaria (V_{CORE}) più bassa rispetto alla prima, con la quale vengono solitamente alimentate la

CPU, le memorie e le periferiche digitali (con l'alimentazione primaria si alimentano le porte I/O e le periferiche analogiche).

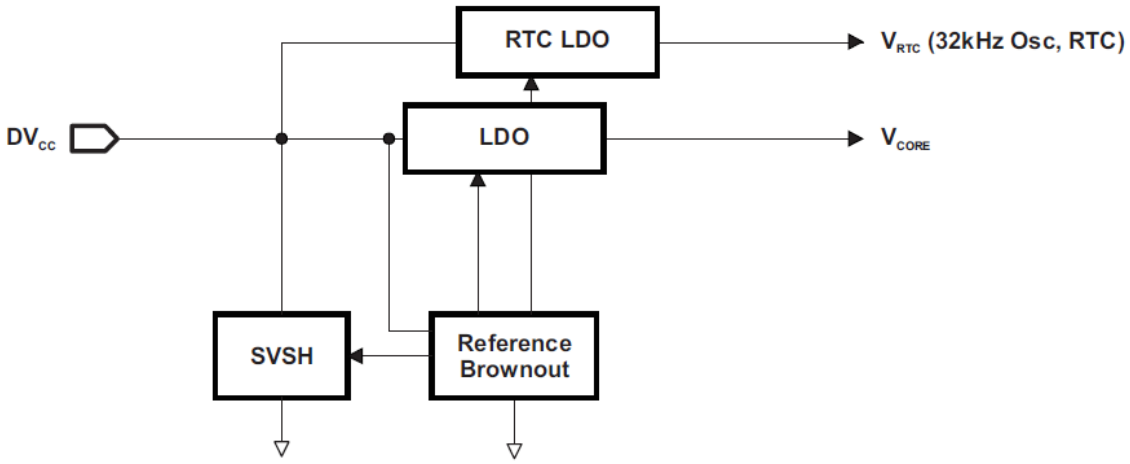


fig_2.24 – Schema a blocchi del PMM [8]

Le modalità .5, sebbene hanno il vantaggio di avere consumi estremamente ridotti, dall'altro portano ad alcuni svantaggi, tra i quali: spegnere tutti i moduli connessi al dispositivo, perdere i dati contenuti nella SRAM e perdere il contenuto dei registri delle periferiche e della CPU. Inoltre, al risveglio da una modalità .5 si ha il completo reset della CPU. Questo porta inevitabilmente a dover re-inizializzare tutte le porte, registri e periferiche prima di poter eseguire il programma. Di conseguenza, si avranno tempi di risveglio più lunghi rispetto alle altre modalità low power (si rimanda al datasheet del dispositivo per maggiori dettagli consultabile sul sito www.ti.com). Per questi motivi, queste modalità LPMx.5 vengono utilizzate per applicazioni in cui si rimane in questi stati a bassissimo consumo per molto tempo. A volte, quindi, risultano più vantaggiose le modalità LPM3, con la possibilità di utilizzare l'oscillatore a basso consumo VLO

e quindi gestire diverse periferiche sia interne (timer, PWM) che esterne, e LPM4 come stato di sleep.

Ovviamente sta al progettista programmare efficientemente il microcontrollore in base alla specifica applicazione su cui è integrato. Ma l'affidabilità e flessibilità del dispositivo appena descritto, offre la possibilità di poter svariare tra molteplici configurazioni in modo da ottenere il risultato desiderato sempre tenendo a mente l'imperativo Ultra - Low Power.

3. Sistema di misura

Per poter caratterizzare energicamente un sistema a microcontrollore è bene strutturare un banco di misura che permetta il raggiungimento dell'obbiettivo. A tal proposito, per effettuare le misure durante l'attività di tesi, si è utilizzato il sistema mostrato in *fig_3.1*, formato da:

- MSP - EXP430FR5969 LaunchPad™ Development Kit, ovvero una Evaluation Board per il microcontrollore MSP430FR5969;
- PC con installato il software IAR Embedded Workbench per poter scrivere il programma di funzionamento (in linguaggio C) ed eseguire il Download e Debug dello stesso sul microcontrollore sopra citato;
- Multimetro digitale Agilent34401A, impostato come amperometro, per misurare la corrente di alimentazione erogata dal microcontrollore;
- Oscilloscopio Tektronix TDS2014 per valutare le frequenze di clock degli oscillatori interni del microcontrollore in esame e verificare la veridicità di quanto dichiarato nel datasheet;
- Alimentatore TTi EL302T per impostare la giusta tensione di alimentazione del microcontrollore;

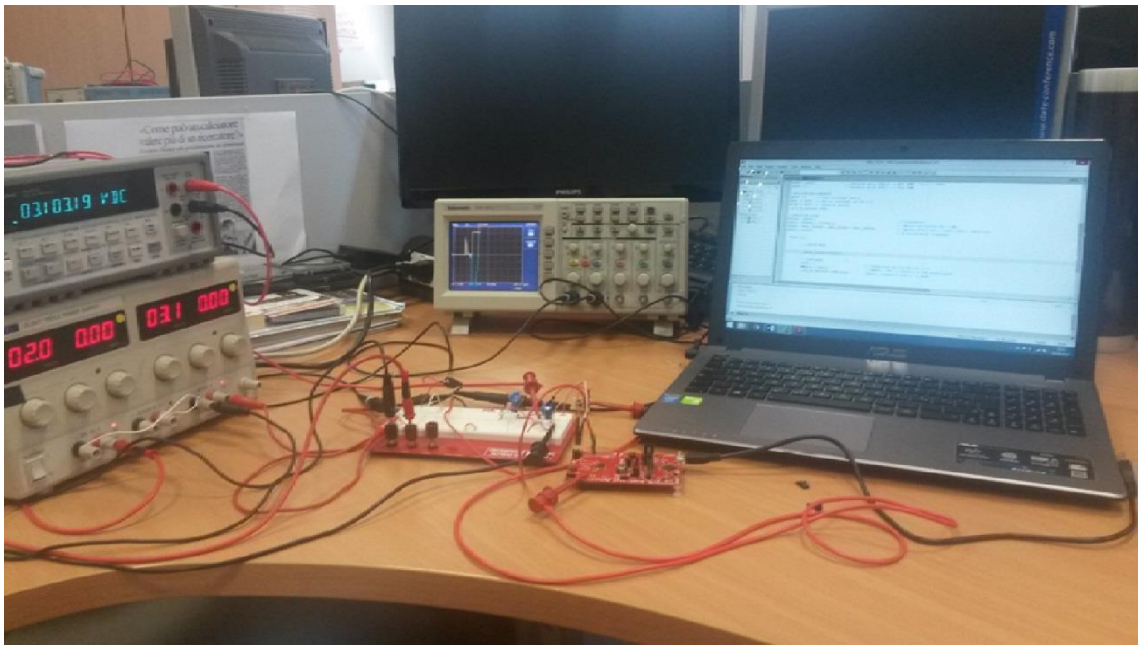


fig_3.1 – Banco di misura

4. Misure

In questo capitolo si andranno a descrivere tutte le misure effettuate durante il periodo di tesi. In particolare, questo passaggio dell'attività si è articolato di 2 fasi:

- Fase 1: valutare sperimentalmente la veridicità dei consumi di corrente, nelle differenti modalità, dichiarati nel datasheet del microcontrollore;
- Fase 2: effettuare una caratterizzazione energetica del dispositivo mentre quest'ultimo esegue un certo di numero di istruzioni e poi entra in modalità di basso consumo.

4.1 FASE 1

Durante questa fase sono state effettuate, come detto in precedenza, tutte le misure necessarie per verificare quanto viene riportato dai datasheet e dalle guide del dispositivo in modo da avere dati sperimentali, e quindi non solo teorici.

I principali registri utilizzati sono:

- **PxDIR:** con questo registro si configurano le porte di I/O come input, valore 0, o come output, valore 1;
- **PxREN:** attraverso questo registro è possibile avere, qualora il pin sia settato come ingresso, abilitato (1) o disabilitato (0) un resistore;
- **PxOUT:** è un registro di scrittura dei dati, ovvero, qualora il bit sia a 1 il pin della porta sarà collegata a Vcc, viceversa qualora il bit sia 0; questo nel caso di pin impostati come output. Nel caso di pin configurati come input, e qualora fosse

abilitato il registro precedente, è possibile scegliere se il resistore sarà di tipo pull-up o pull-down;

- **PxIE:** con questo registro si abilitano sul pin gli interrupt;
- **PxIFG:** dopo che si è verificato un interrupt, il valore del pin che l'ha generato viene memorizzato in questo registro. Prima di poter servire una nuova interruzione, allora, bisognerà azzerare il contenuto di questo registro;
- **CSCTL0:** è il registro dedicato alla password del sistema di clock. Scrivendo la password corretta tutti i registri relativi al sistema di clock possono essere scritti;
- **CSCTL1:** è il registro dedicato alla sorgente di clock DCO. Configurando opportunamente i bit si possono scegliere varie frequenze di click in un range 1MHz – 24MHz;
- **CSCTL2:** questo registro permette di associare le sorgenti di clock alle uscite (ACLK, MCLK e SMCLK);
- **CSCTL3:** serve per dividere il valore di frequenza selezionato;
- **PMMCTL0:** permette di disattivare il modulo PMM in modo da poter entrare in modalità .5;
- **PM5CTL0:** questo risulta essere uno dei registri più importanti del microcontrollore. Possiede il flag LOCKLPM5, impostato via hardware a 1 (ovvero ogni volta che il microcontrollore viene resettato questo bit si porta sempre a livello alto) che blocca tutte le porte di I/O imponendo loro lo stato di alta impedenza, ossia i pin risultano essere disattivati. Questo, quindi, vieta il loro utilizzo per qualsiasi cosa, come gestione degli interrupt, accensione di

led, pressione di pulsanti, etc. La prima operazione da eseguire nel codice di programmazione è dunque quella di "azzerare" questo registro negando il flag sopra citato. In questo modo sarà possibile configurare i pin come desiderato utilizzando i registri dedicati alle porte specificati in precedenza;

- **__bis_SR_register(LPMx_bits):** mediante questa istruzione si vanno a settare i bit dello Status Register che mandano il microcontrollore nelle modalità di basso consumo;
- **__bic_SR_register_on_exit(LPMx_bits):** questa istruzione, al contrario di quella sopra, pulisce i flag dello Status Register in modo da far uscire il microcontrollore dalla modalità di basso consumo a cui era configurato.

Di seguito si riporta lo schema a blocchi del circuito impiegato (*fig_4.1*).

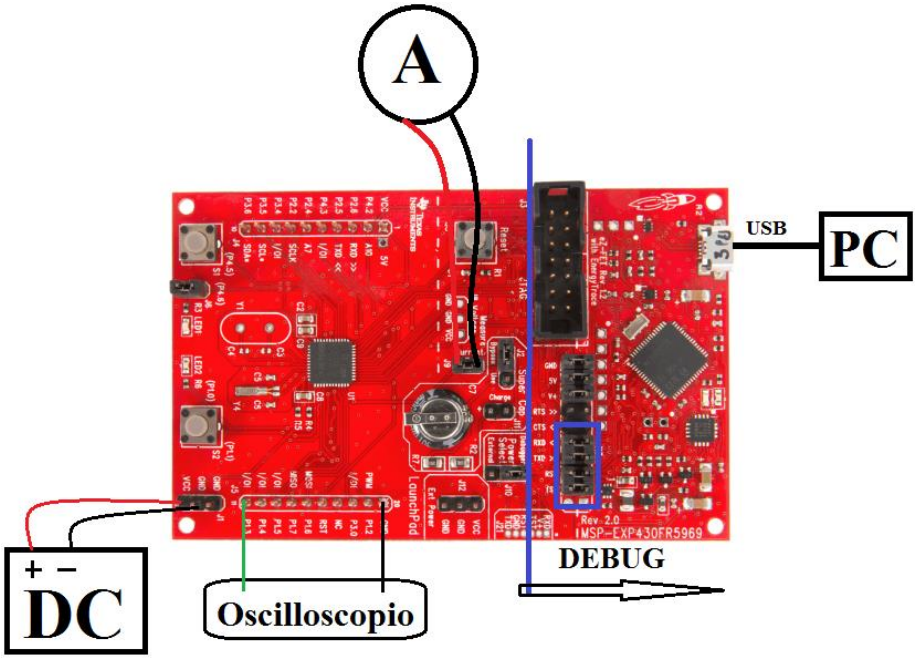


fig_4.1 – Sistema di misura utilizzato nella prima fase

Prima di riportare i valori delle misure ottenute, si vogliono fare alcune precisazioni riguardo la modalità di svolgimento di questa fase. Innanzitutto si sono riprodotte le condizioni di misura descritte nel datasheet impostando correttamente i registri elencati in precedenza, in particolare:

- ✓ Porte di I/O impostate come input con i resistori attivati in modalità pull-down;
- ✓ Sorgenti di clock: ACLK→LFXT o VLO, MCLK=SMCLK→DCO;

Dopodiché si effettua il Debug & Download del programma sul microcontrollore, mediante connessione USB e sezione Debug (*la si può notare dalla fig_4.1*). In seguito si disconnette l'alimentazione fornita al micro dalla porta USB (in quanto questa è pari a 3.6V), si rimuovono i jumper (rettangolo blu della sezione Debug) in modo da disconnettere tra loro microcontrollore e circuito di Debug e si eroga, attraverso un alimentatore esterno, la giusta tensione di alimentazione del dispositivo pari a 3V.

Di seguito si riportano le misure effettuate: in particolare in Tab_4.1 e fig_4.2 le misure sono relative all'utilizzo, come sorgente a bassa frequenza sull'uscita ACLK, dell'oscillatore esterno LFXT, mentre in Tab_4.2 e fig_4.3 le valutazioni si riferiscono all'utilizzo, sull'uscita ACLK, della sorgente interna VLO; in entrambi i casi, inoltre, viene impostata come sorgente ad alta frequenza l'oscillatore interno DCO.

Operating Modes	Voltage	FREQUENZA DCO				
		1MHz	4MHz	8MHz	12MHz	Unit
Active Mode	3.0 V	157	291,13	496,84	804,3	µA
LPM0	3.0 V	112,9	135,47	184,26	266,4	µA
LPM1	3.0 V	72	96	148,26	243,04	µA

Tab_4.1 – Consumi di corrente in relazione alla frequenza:

ACLK→LFXT=32KHz, MCLK=SMCLK→DCO

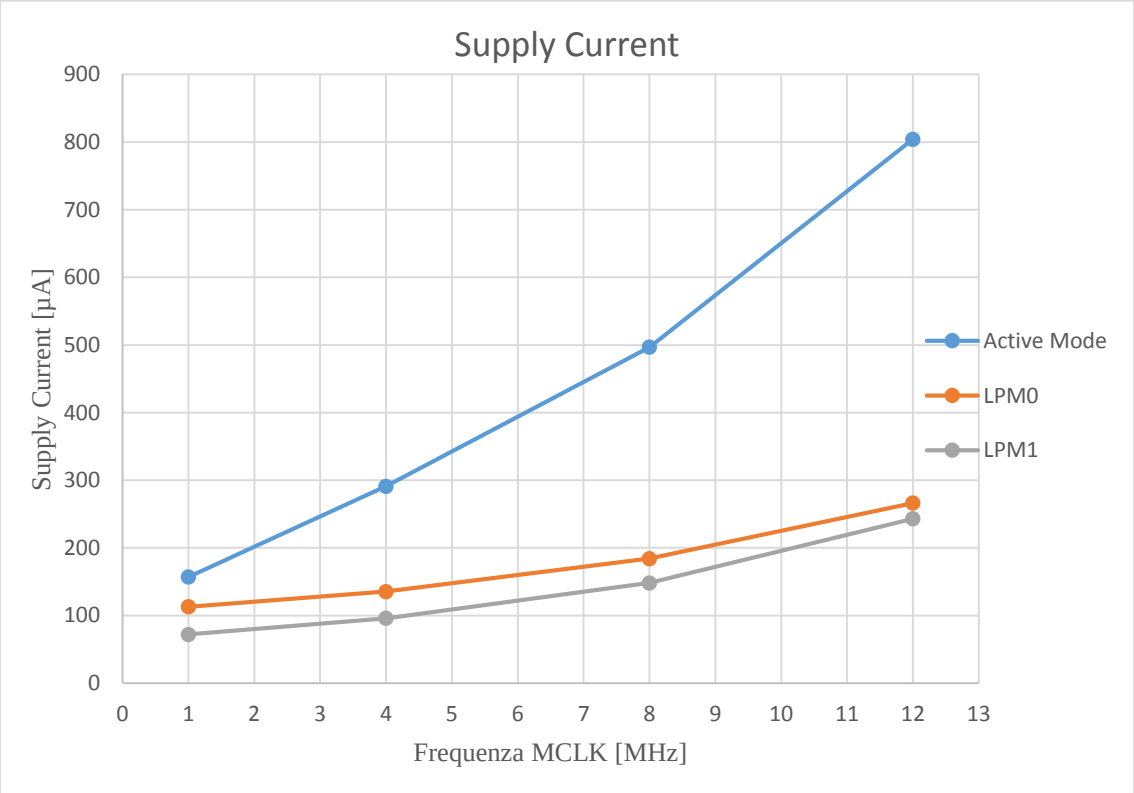


fig_4.2 – Relazione grafica tra la frequenza e la corrente:

ACLK→LFXT=32KHz, MCLK=SMCLK→DCO

Operating Modes	Voltage	FREQUENZA DCO				
		1MHz	4MHz	8MHz	12MHz	Unit
Active Mode	3.0 V	115	260	472	764	µA
LPM0	3.0 V	76	100	115	246	µA
LPM1	3.0 V	35	59	112	205	µA

Tab_4.2 – Consumi di corrente in relazione alla frequenza:

ACLK→VLO=9.4KHz, MCLK=SMCLK→DCO

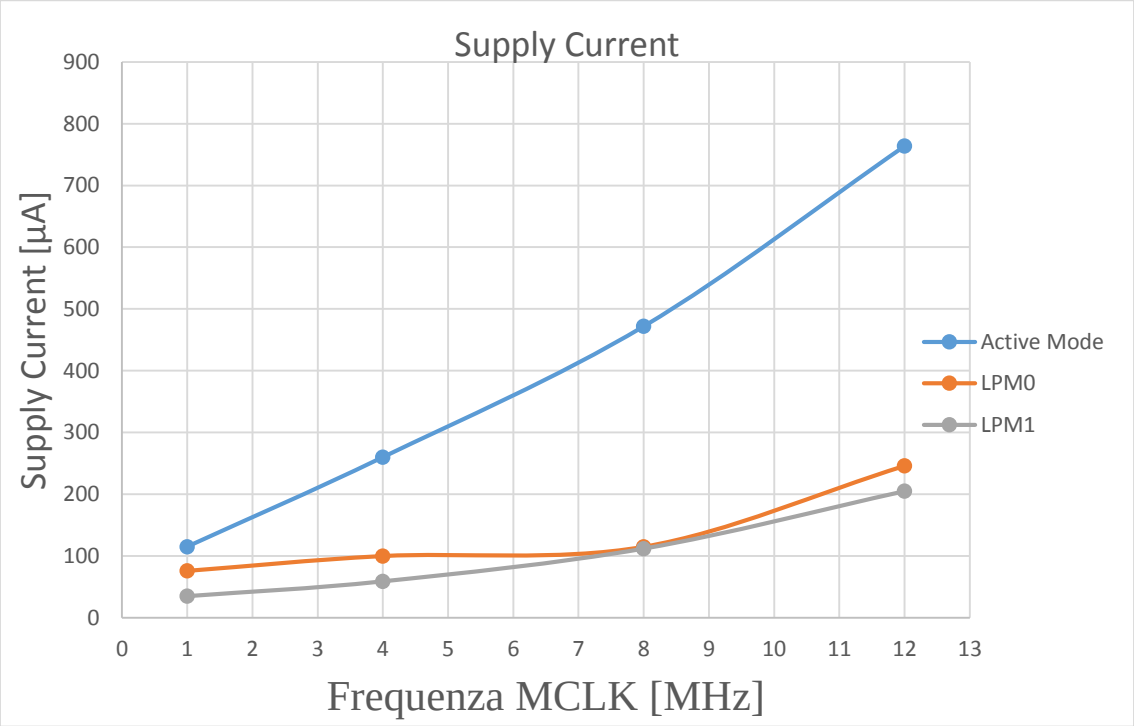


fig_4.3 – Relazione grafica tra la frequenza e la corrente:

ACLK→VLO=9.4KHz, MCLK=SMCLK→DCO

Da queste due tabelle e grafici si può chiaramente notare la dipendenza della corrente dalla frequenza, in particolare con un andamento pressoché lineare, confermando la nota relazione espressa in equazione (1):

$$I_{media} = \frac{P}{V_{DD}} = C_L \cdot V_{DD} \cdot f_{CK} \qquad (1)$$

Infine, in Tab_4.3, si riporta un confronto tra i consumi ricavati sperimentalmente e quelli dichiarati nel datasheet nelle varie modalità di funzionamento del microcontrollore:

	Voltage	Corrente misurata [µA] ACLK=LFXT MCLK =1MHz	Corrente misurata [µA] ACLK=VLO MCLK =1MHz	Corrente Datasheet [µA] ACLK=LFXT MCLK =1MHz	Corrente Datasheet [µA] ACLK=VLO MCLK==1MHz
Active Mode	3.0 V	157	115	110	Non riportata
LPM0	3.0 V	112,9	76	80	Non riportata
LPM1	3.0 V	72	35	35	Non riportata
LPM2	3.0 V	1,1	0,74	0,9	0,7
LPM3	3.0 V	0,8	0,41	0,6	0,4
LPM4	3.0 V	0,32	0,32	0,3	0,3
LPM3.5	3.0 V	0,29	0,26	0,25	Non riportata
LPM4.5	3.0 V	0,02	0,02	0,02	0,02

Tab_4.3 – Confronto tra corrente misurata e corrente dichiarata nel datasheet nelle varie modalità di funzionamento

È interessante osservare come i consumi valutati utilizzando la sorgente LFXT si discostano rispetto a quelli dichiarati nel datasheet, in alcune modalità anche di 30-40 µA. Invece, si hanno consumi pressochè identici qualora si utilizzi l'oscillatore interno VLO. Sicuramente un dato da una parte inaspettato, in quanto pur riproducendo tutte le condizioni di misura espresse nel datasheet non si sono raggiunti pienamente i valori dichiarati. D'altra parte, però, si ha una riconferma che utilizzare una frequenza inferiore porta ad avere consumi energetici più bassi.

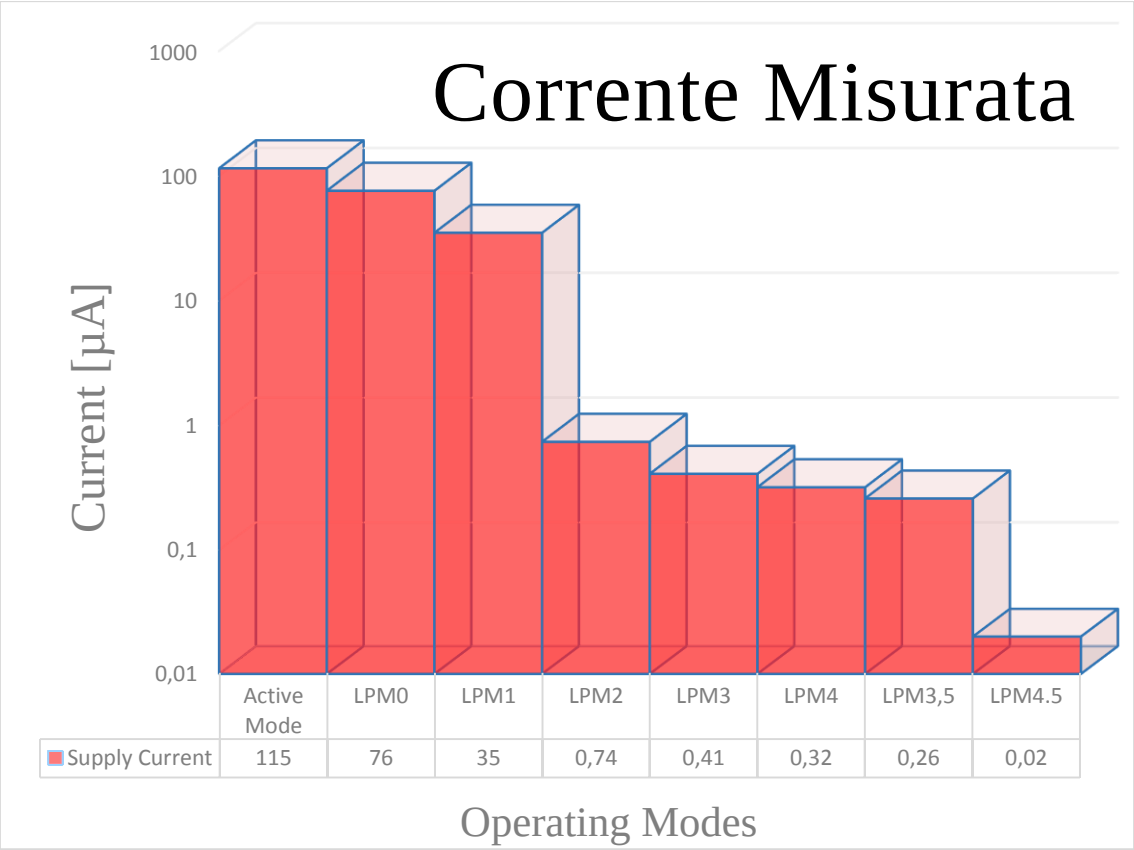


fig_4.4 – Consumo di corrente media nelle varie modalità

Inoltre, in *fig_4.4* si può osservare la netta differenza di consumi tra le modalità Active Mode, LPM0 e LPM1, che hanno attiva la sorgente ad alta frequenza DCO e quella a bassa frequenza VLO rispetto alle altre dove la sorgente di clock è fornita solo dall'oscillatore interno VLO (LPM2 e LPM3) o addirittura non vi è alcuna sorgente attiva (LPM4, LPM3.5 e LPM4.5).

Infine, si riporta il codice del programma utilizzato durante questa prima fase:

```
//Codice1: attraverso questo codice si programma il
microcontrollore configurando le porte e il sistema di
clock come specificato nel datasheet e poi si fa lavorare
il microcontrollore nelle diverse modalità al fine di
rilevarne i consumi
```

```
#include "msp430fr5969.h"
int main(void) {
WDTCTL = WDTPW | WDTHOLD;     // disattivo il watchdog timer

//Inizializzazione porte

PM5CTL0 &= ~LOCKLPM5;      // Sblocca I/O da High Impedance

PADIR = 0x0000;           // Pin PORT_A impostati come input
PBDIR = 0x0000;           // Pin PORT_B impostati come input
PJDIR = 0x00;             // Pin PORT_J impostati come input
PAREN = 0xFFFF;           // Resistori PORT_A abilitati
PBREN = 0xFFFF;           // Resistori PORT_B abilitati
PJREN = 0xFF;             // Resistori PORT_J abilitati
PAOUT = 0x0000;           // Resistori PORT_A --> PULL DOWN
PBOUT = 0x0000;           // Resistori PORT_B --> PULL DOWN
PJOUT = 0x00;             // Resistori PORT_J --> PULL DOWN

//SPECIFICHE CLOCK
CSCTL0 = 0xA500;                    // CS password
CSCTL1 = DCORSEL + DCOFSEL_0;       // Imposto oscillatore
                                    // DCO = 1MHz

CSCTL2 = SELA__VLOCLK + SELM__DCOCLK + SELS__DCOCLK;
// Imposto ACLK = VLO = 9.4KHz e MCLK = SMCLK =DCO

CSCTL3 = 0x0000;              // No divisione di frequenza

  while (1){

          // ACTIVE MODE
          __delay_cycles(1000000);

          // SLEEP MODE
          PMMCTL0 = 0xA510;  // PMMREG = OFF
          __bis_SR_register (LPM4_bits);  //Entra in
                              //modalità LPM4.5 --> SLEEP
  }
}
```

4.2 FASE 2

In questa seconda fase, sulla base anche delle misure effettuate in precedenza, si è voluto effettuare un bilancio energetico del microcontrollore mentre quest'ultimo esegue un certo numero di istruzioni ad una certa frequenza e poi entra in modalità di basso consumo. Quando si parla di bilancio energetico ci si riferisce alla potenza consumata dal microcontrollore per eseguire le sue operazioni nell'unità di tempo:

$$Energia = V\ x\ I\ x\ Tempo \qquad (2)$$

Più volte durante l'elaborato è stato espresso che lavorare a bassa frequenza vuol dire minori consumi; la controprova si nota anche dai dati ricavati nella fase 1. Tuttavia, nel caso in cui il dispositivo non abbia la necessità di rimanere attivo per tutto il tempo, operare a frequenze più elevate non sempre comporta maggiori consumi. Infatti, si può riscrivere l'equazione (2) come:

$$Energia\ Totale = Energia_{Attiva} + Energia_{Sleep}$$
$$= I_{Attiva} x V x Tempo_{Attiva} + I_{Sleep} x V x Tempo_{Sleep} \qquad (3)$$

Effettivamente, l'energia consumata è da valutare in base al tempo in cui il microcontrollore risulta attivo e il tempo in cui quest'ultimo rimane in stato di sleep: se si sta attivi per tempi ridotti e poi si permane in sleep per un intervallo temporale molto lungo, lavorare ad una frequenza elevata può essere un vantaggio. Infatti, il consumo di energia dinamico per eseguire il medesimo programma non dipende dalla frequenza di funzionamento. Tuttavia, le componenti di potenza statica influiscono per la durata di esecuzione del programma.

Dunque, per effettuare il bilancio energetico, espresso nell'equazione (3) e verificare quale fosse il miglior compromesso tra velocità di clock e consumi si è proceduto nel seguente modo [2]: su un intervallo temporale di 120 secondi, si è fatto lavorare il microcontrollore per un certo numero di istruzioni nulle, precisamente 1000, eseguite a differenti frequenze di elaborazione e poi è stato mandato in sleep. In questo modo, conoscendo la frequenza di lavoro ed il numero di istruzioni da eseguire si è calcolato matematicamente quanto tempo rimanesse attivo il microcontrollore. Moltiplicando poi questo tempo per la tensione di alimentazione del dispositivo e la corrente di consumo media a quella determinata frequenza, si è valutata l'energia spesa in modalità attiva.

Inoltre, conoscendo la corrente media in modalità sleep, moltiplicata sempre per la tensione di alimentazione e per il restante tempo in cui il microcontrollore rimaneva spento, si ricava anche l'energia in modalità sleep. Sommando queste due energie ricavate si trova quella spesa in totale dal microcontrollore nei 120 secondi.

Di seguito, in *Tab_4.4*, si riportano i dati ricavati:

Frequenza DCO	Corrente Attiva [µA]	Corrente Sleep [µA]	Tempo Attivo [µs]	Tempo Sleep [s]	Energia Attiva [nJ]	Energia x istruzione [nJ]	Energia Sleep [µJ]	Energia Totale [µJ]
1MHz	116	0,021	1000	119,999	348	0,348	7,55993	7,9079
4MHz	260,65	0,021	250	119,99975	195,48	0,19548	7,55998	7,7554
8MHz	473,55	0,021	125	119,999875	177,58	0,17758	7,559992	7,7375
12MHz	727,4	0,021	85	119,999915	185,48	0,18548	7,559994	7,7454

Tab_4.4 – Bilancio energetico del microcontrollore in un intervallo temporale di 120 secondi

[2] Il sistema di misura utilizzato è quello mostrato in *fig_4.1*.

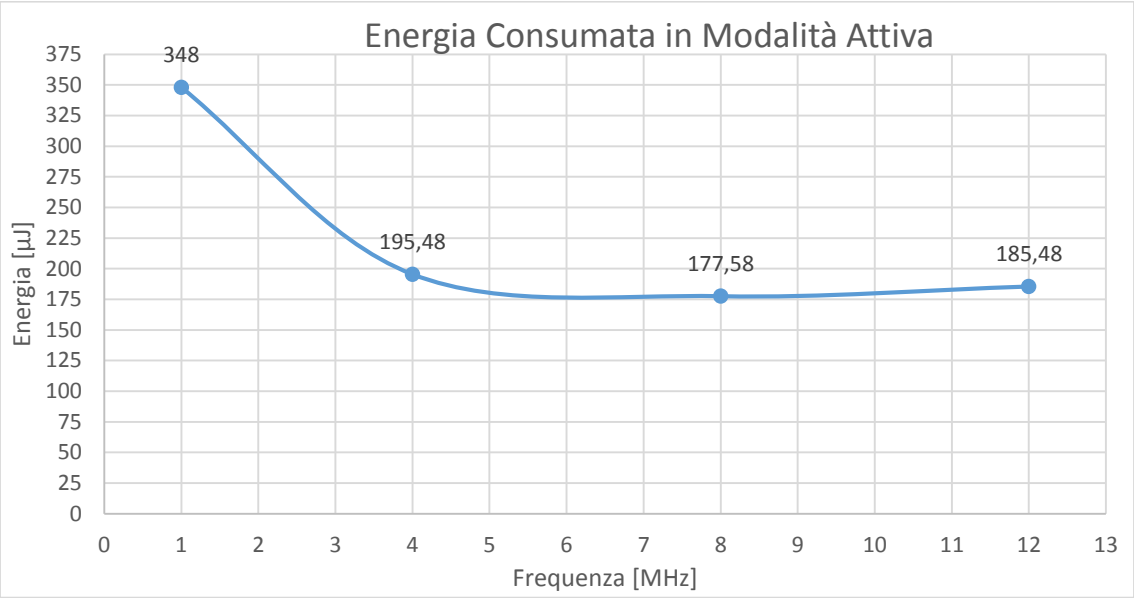


fig_4.5 – Energia consumata in modalità attiva

Si può notare che i minori consumi si hanno a 8MHz. Quindi c'è effettivamente un compromesso tra la velocità di elaborazione del microcontrollore e il consumo di energia totale, anche se il trend, in *fig_4.5*, mostra un consumo di energia pressoché costante a partire da 4 MHz in su.

Operare in questo modo, però, non è del tutto corretto, in quanto non sono stati presi in considerazione alcuni fattori che possono incidere sul consumo totale del microcontrollore. Uno su tutti, ad esempio, è il consumo che si ha durante la transizione da modalità attiva ad una modalità a basso consumo, o viceversa. In più possono esserci anche altri consumi che il procedere come visto in precedenza non porta a considerare.

Ci si è voluti approcciare, allora, ad un nuovo sistema di misura che in qualche modo potesse tener conto anche dei consumi non presi in considerazione in precedenza.

L'idea è stata quella di sfruttare l'equazione caratteristica di un condensatore (4)

$$i_C = C\frac{dV}{dt} \qquad (4)$$

per calcolare la corrente media di consumo del microcontrollore, operante per un certo periodo di tempo in modalità attiva e per il restante in una modalità a basso consumo.

Il nuovo sistema di misura è riportato di seguito, in *fig_4.6*, e prevede:

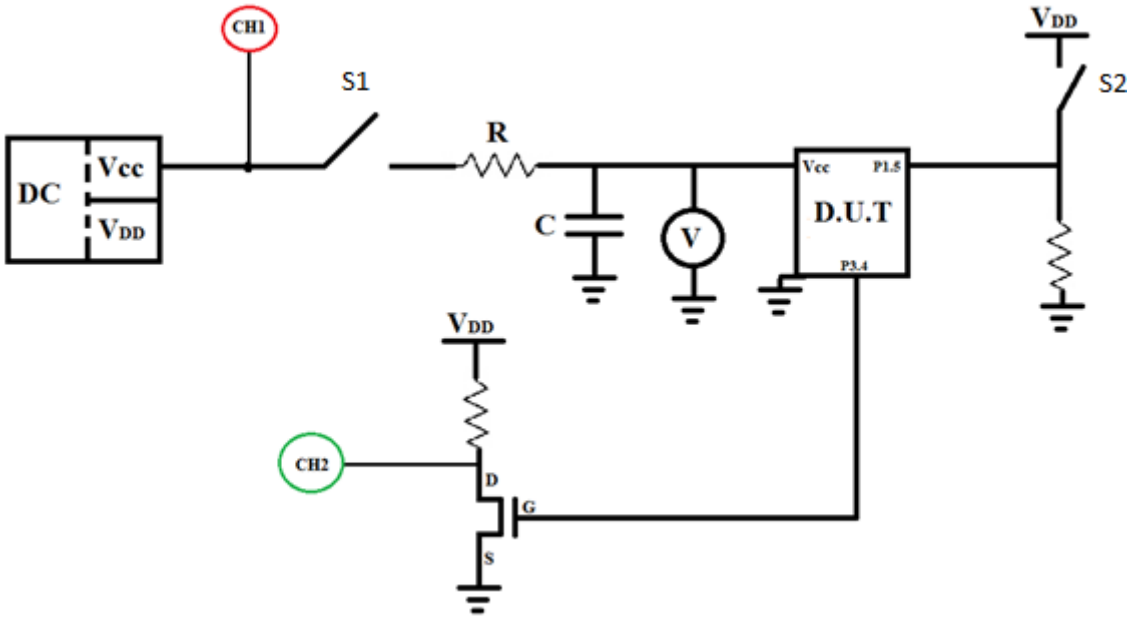


fig_4.6 – Sistema di misura Fase 2

- D.U.T (Device Under Test): non è altro che il blocco riferito al microcontrollore in esame;
- Condensatore: attraverso di esso si alimenta il microcontrollore;
- Voltmetro: per misurare la tensione ai capi del condensatore;
- DC: alimentatore con due differenti sorgenti (VCC=3.1V per caricare il condensatore e V_{DD}=2.2V);

- Interruttori S1 (per scollegare l'alimentatore al condensatore) e S2 (per la gestione dell'interrupt di risveglio del microcontrollore);
- nMOS: per la gestione della visualizzazione, sull'oscilloscopio, del tempo in cui il microcontrollore passa da una modalità all'altra.
- Oscilloscopio: CH1 per visualizzare l'istante in cui far partire la misura e CH2 per valutare il periodo durante il quale il microcontrollore è in modalità attiva.

Sostanzialmente, il modo di procedere è il seguente: si carica il condensatore ad una tensione di 3.1V (essendo l'interruttore S1 inizialmente chiuso anche sul CH1 è presente la stessa tensione). Nell'istante in cui si apre l'interruttore S1, il CH1 si porta a 0V: in questo modo si ha l'esatto istante in cui inizia la misura. Il microcontrollore inizialmente parte da uno stato a basso consumo. Mediante l'interruttore S2, collegato al Pin1.5 inizializzato come input con un resistore di pull-down, si risveglia il microcontrollore che entrerà in modalità attiva dove svolge 4 milioni di istruzioni ad una frequenza di 8MHz (questa scelta è frutto delle misurazioni effettuate nell'approccio precedente in quanto risulta essere la frequenza alla quale il microcontrollore, quando lavora in modalità attiva, ha il minor consumo energetico a parità di istruzioni da eseguire); poi nuovamente rientra nella modalità a risparmio energetico. Durante queste transizioni il Pin3.4 (inizializzato come output collegato a Vcc → 1 logico) cambia valore:

- Transizione da Modalità Attiva a LPM → da "1" a "0";
- Transizione da LPM a Modalità Attiva → da "0" a "1".

Essendo collegato in uscita un nMOS (ai cui capi è connessa una sonda dell'oscilloscopio, CH2) è possibile vedere i passaggi da una modalità all'altra. In questo modo si saprà per certo il tempo durante il quale il microcontrollore sarà in modalità attiva. Mentre il DUT lavora, il condensatore si scaricherà. Quando la tensione ai suoi capi, valutata attraverso il voltmetro posto in parallelo, raggiungerà il valore di 2.9V, si arresta la misura (in pratica viene spinto il tasto STOP sull'oscilloscopio). Conoscendo dunque tutti i parametri dalla misura sperimentale, si può calcolare la corrente media attraverso l'equazione (3).

Di seguito, in *fig_4.7*, si potrà apprezzare un andamento temporale del funzionamento della misura, mettendo in evidenza le forme d'onda presenti sui canali dell'oscilloscopio.

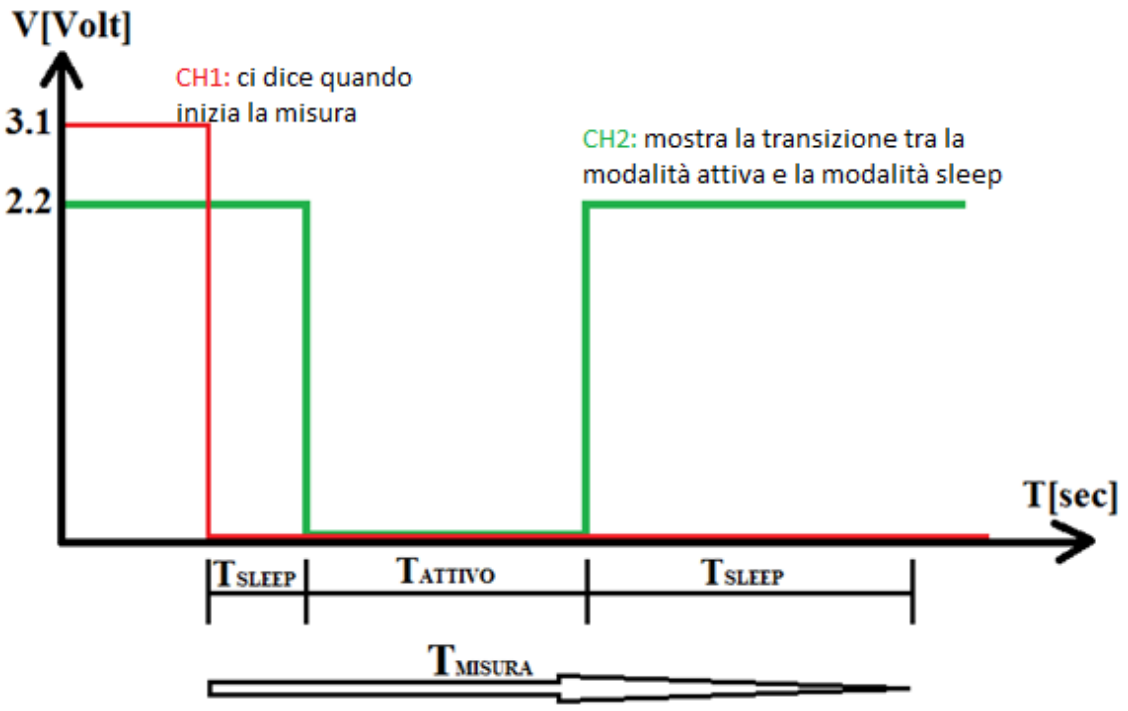


fig_4.7 – Andamento temporale della misura

Si riportano ora, in *Tab_4.5*, i risultati ottenuti attraverso questo secondo approccio di misura:

Modalità di funzionamento	dV=Vi-Vf [mV]	dt=Tmisura [sec]	Tattivo [sec]	C[µF]	Ic[µA]	Energia [µJ]
LPM0	199	25,7	0,498712	20400	157,961	12178,793
LPM1	198	34,9	0,49881	20400	115,736	12117,559
LPM2	197	31,21	0,498744	1413,3	8,92086	835,260
LPM3	198	26,4	0,498804	1302,1	9,76575	773,447
LPM4	197	28,98	0,498856	1302,1	8,8514	769,541
LPM4.5	199	22,94	0,498942	1200,6	10,415	716,758

Tab_4.5 – Bilancio energetico valutato sperimentalmente

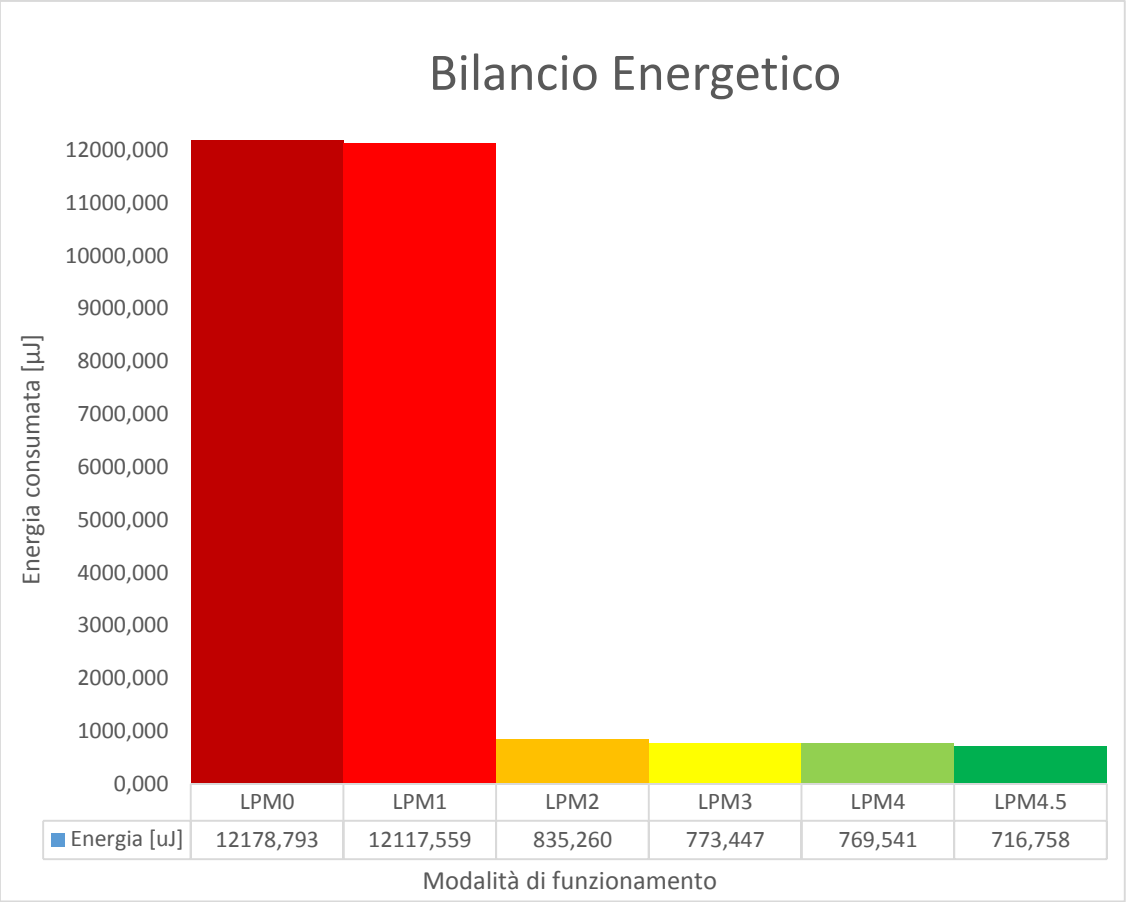


fig_4.8 – Confronto fra i consumi energetici nelle varie modalità di funzionamento

Come si può notare dalla *fig_4.8*, in questo caso di misura nel quale il microcontrollore permane in modalità sleep per un lungo periodo rispetto a quello in cui è in funzionamento attivo, la modalità LPM4.5 risulta essere la più vantaggiosa. Tuttavia, non sempre la modalità .5 ha la conseguenza del minor consumo energetico. Tutto dipende dalla particolare applicazione che il microcontrollore deve gestire. Si può notare, infatti, che non c'è molta differenza di consumo energetico tra le modalità LPM4.5, LPM4 e LPM3. Come affermato nel capitolo 2.1.4, alle volte è quindi più ragionevole lavorare in modalità LPM3 o LPM4 in quanto si hanno certi vantaggi che la LPM4.5 non ha, come il poter utilizzare la sorgente VLO (per timer, contatori, interrupt) e il non dover consumare elevata energia durante il risveglio, in quanto non risulta necessaria una re-inizializzazione di tutti i registri.

Come prova finale, si vuole fare un confronto tra la corrente media calcolata sperimentalmente e quella calcolata attraverso il primo approccio di misura avuto durante la seconda fase, mediante l'equazione (5):

$$I_{Teorica} = \frac{I_{attiva} x T_{attiva} + I_{LPM} x T_{LPM}}{T_{totale}} \qquad (5)$$

Di seguito, in *Tab_4.6*, si riporta una tabella riassuntiva con il confronto tra i due approcci di misura:

Modalità	Tempo Attivo [s]	Tempo Sleep [s]	Tempo Totale [s]	Corrente Attiva [µA]	Corrente LPM [µA]	Ic teorica [µA]	Ic misurata [µA]
LPM0	0,498712	25,201288	25,7	473,55	151,2	157,4552456	157,961
LPM1	0,49881	34,40119	34,9	473,55	110,2	115,3931981	115,736
LPM2	0,498744	30,711256	31,21	473,55	0,74	8,295628024	8,92086
LPM3	0,498804	25,901196	26,4	473,55	0,41	9,349550173	9,76575
LPM4	0,498856	28,481144	28,98	473,55	0,32	8,466087815	8,8514
LPM4.5	0,498942	22,441058	22,94	473,55	0,02	10,31921557	10,41496

Modalità di funzionamento	Energia teorica [µJ]	Energia misurata[µJ]
LPM0	12139,799	12178,793
LPM1	12081,668	12117,559
LPM2	776,720	835,260
LPM3	740,484	773,447
LPM4	736,042	769,541
LPM4.5	710,168	716,758

Tab_4.6 – Confronto tra i bilanci energetici

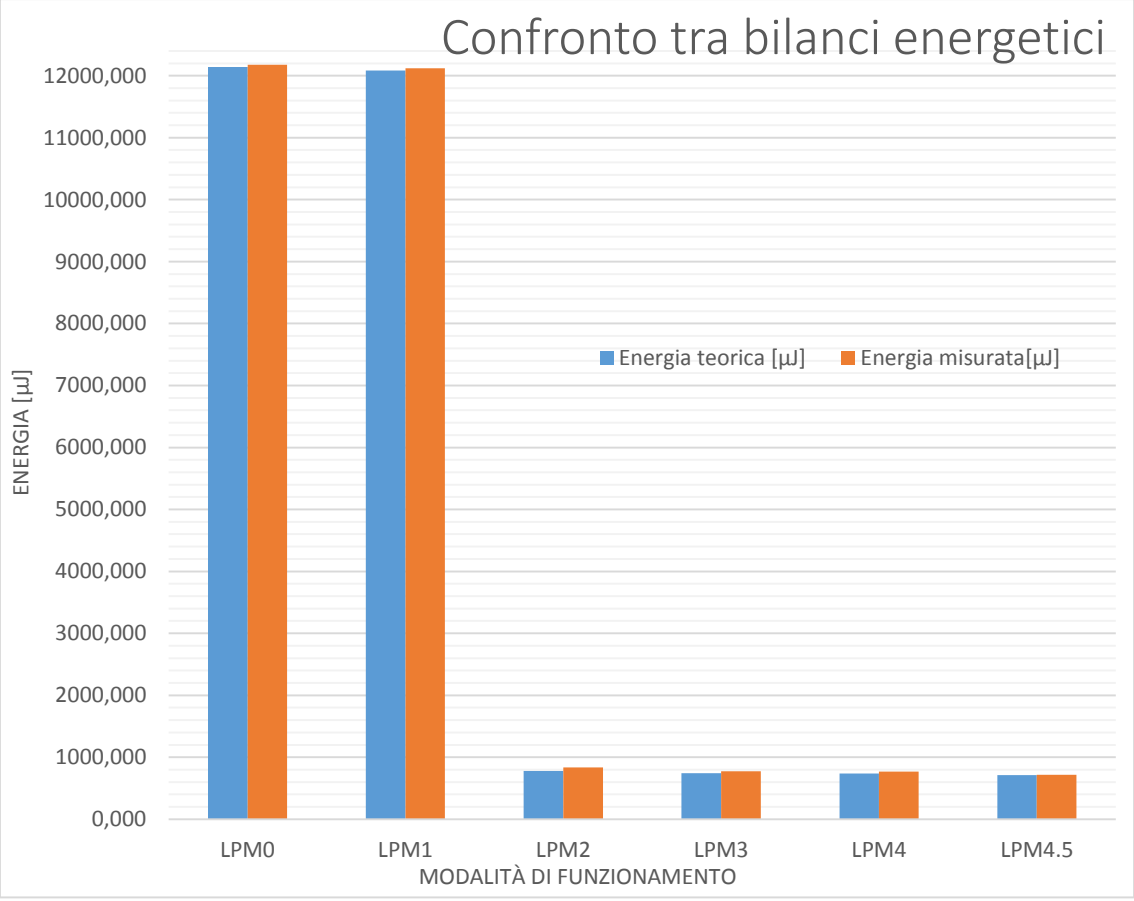


fig_4.9 – Confronto tra i bilanci energetici

Chiaramente, è possibile notare, in *fig_4.9*, che la misura sperimentale ha consumi, seppur minimi, più elevati in quanto tiene conto anche di quei fattori, citati più volte durante il paragrafo, che la misura teorica non prevedeva.

In ultimo, si riporta il codice del programma utilizzato durante questa seconda fase:

```
// Codice2: Attraverso questo codice il microcontrollore viene programmato per svolgere 4 milioni di istruzioni (modalità attiva) e poi viene mandato in modalità di risparmio energetico. Per risvegliare il microcontrollore dalla modalità LPM si utilizza una routine di interrupt associata alla pressione di un pulsante esterno collegato al dispositivo attraverso una porta di I/O.

#include "msp430fr5969.h"

int main(void) {
WDTCTL = WDTPW | WDTHOLD;          // disattivo il watchdog timer


//Inizializzazione porte

PM5CTL0 &= ~LOCKLPM5;              // Sblocca I/O da High Impedance

PADIR = 0x0000;           // Pin PORT_A impostati come input
PBDIR = 0x0010;           // Pin PORT_B impostati come input
                          // P3.4 impostato come output
PJDIR = 0x00;             // Pin PORT_J impostati come input
PAREN = 0xFFFF;           // Resistori PORT_A abilitati
PBREN = 0xFFFF;           // Resistori PORT_B abilitati
PJREN = 0xFF;             // Resistori PORT_J abilitati
PAOUT = 0x0000;           // Resistori PORT_A --> PULL DOWN
PBOUT = 0x0010;           // Resistori PORT_B --> PULL DOWN
                          // P3.4 impostato come out a Vcc
PJOUT = 0x00;             // Resistori PORT_J --> PULL DOWN

//ABILITAZIONE INTERRUPT
PAIFG = 0;                // Clear Interrupt PORT_A
PAIE  |= BIT5;            // Abilita interrupt sul PIN P1.5
P1IES |= BIT5;            // Fronte di discesa
__bis_SR_register (GIE); // Abilita interrupt generali
```

```
//SPECIFICHE CLOCK
CSCTL0 = 0xA500;                        // CS password
CSCTL1 = DCORSEL + DCOFSEL_3;           // Imposto oscillatore
                                        // DCO = 8MHz

CSCTL2 = SELA__VLOCLK + SELM__DCOCLK + SELS__DCOCLK;
// Imposto ACLK = VLO = 9.4KHz e MCLK = SMCLK =DCO

CSCTL3 = 0x0000;                    // No divisione di frequenza



  while (1){

          // ACTIVE MODE
          __delay_cycles(4000000);

          // SLEEP MODE
          P3OUT = 0;                // Transizione del bit P3.4
                                    // da 1-->0
          PMMCTL0 = 0xA510;   // PMMREG = OFF
          __bis_SR_register (LPM4_bits);  //Entra in
                                    //modalità LPM4.5 --> SLEEP
  }
}

/******************* ISR ****************************/
//Routine    di    interrupt    per    il    risveglio    del
microcontrollore

#pragma vector = PORT1_VECTOR
__interrupt void P1_ISR(void){

      __bic_SR_register_on_exit (LPM1_bits);
      P1IFG &= ~BIT5;               // clear interrupt flag
      P3OUT |= BIT4;                // Transizione del bit P3.4
                                    // da 0-->1
}
```

5. Conclusioni

In questo elaborato di tesi si è affrontato in primo luogo uno studio dei microcontrollori, concentrandosi sull'architettura, che ne fa del dispositivo un sistema integrato su singolo chip a tutti gli effetti, vista la presenza di periferiche, moduli di clock e soprattutto memoria.
Proprio quest'ultima, oltre chiaramente alla CPU, è la parte fondamentale del dispositivo, in quanto in essa vengono mantenuti il programma e le istruzioni che il sistema deve eseguire. Avere quindi una memoria affidabile, flessibile e molto efficiente, sia come velocità di scrittura e lettura sia come consumi, è una delle esigenze alla base dello sviluppo dei sistemi Harvesting.
La risposta a questo bisogno, come espresso nell'elaborato, è data dai microcontrollori basati su tecnologia FRAM. Questa risulta essere la memoria che fa al caso dei sistemi embedded alimentati a batteria garantendo un'elevata flessibilità nella sua gestione, bassissimi consumi in relazione alle altre memorie presenti sul mercato e soprattutto una durata ai cicli di scrittura praticamente infinita.
Non solo la memoria, però, porta i vantaggi di avere bassi consumi ma anche le varie modalità di funzionamento a cui il microcontrollore può lavorare fan sì che esso sia un fondamento per i sistemi integrati.
Durante l'attività è stato possibile studiare e prendere confidenza con le modalità operative a basso consumo, fino ad effettuare sperimentalmente un bilancio energetico per ogni modalità per capire in quale situazione il

microcontrollore potesse portare effettivamente ad avere i minor consumi garantendo comunque una certa funzionalità del sistema. Si è partiti allora con un sistema di misura che permettesse di quantificare il consumo di corrente in ciascuna modalità di funzionamento, riproducendo le condizioni di lavoro dichiarate nel datasheet del microcontrollore in esame e confrontando i valori sperimentali con quelli dichiarati. In questa fase si è potuta apprezzare una concordanza quasi netta su tutti i valori misurati e dichiarati. Dopodiché, attraverso un secondo sistema di misura è stato possibile effettuare un analisi dei consumi più accurata, simulando un sistema il cui compito prevedeva lo svolgimento di un certo numero di istruzioni ad una certa frequenza e poi entrare in modalità a risparmio energetico. Questo modo di operare permette di effettuare innumerevoli prove e situazioni differenti al fine di trovare la giusta combinazione per avere un sistema con i minor consumi di energia. In particolare, dalle misure effettuate per l'applicazione specifica della fase 2 è stato valutato che il comportamento che portava ai minori consumi fosse quello di entrare in modalità LPM4.5 una volta terminate le operazioni da svolgere.

È chiaro che con consumi così bassi nelle varie modalità LPM è possibile implementare diversi dispositivi ultra-low power. Infatti, un possibile sviluppo futuro potrebbe essere quello di progettare sistemi alimentati a batteria basati su microcontrollore MSP430FR5969, come sensori di fumo, termostati, applicazioni mediche, applicazioni elettroniche portatili dove il basso consumo del microcontrollore permettono un notevole risparmio energetico e di conseguenza una durata di vita della batteria di 15-20 anni.

Bibliografia

[1] www.elettronicanews.it/articoli.html

[2] www.agienergia.it

[3] it.wikipedia.org

[4] it.rs-online.com

[5] it.emcelettronica.com/low-power-contest-microcontrollori

[6] www.microchip.com: Datasheet PIC24FJ128GA310

[7] www.freescale.com: Datasheet MC9S08RG60

[8] www.ti.com: Datasheet e User Guide MSP430FR5969

[9] www.atmel.com: Datasheet ATxmega64D3

[10] www.laurtec.it/tutorial/programmare-gli-msp430/135-msp430-001-msp430

[11] www.dei.unipd.it/~ieeesb/PIC/PresentazionePIC_01

[12] Tesi di Laurea triennale "Nuove tecnologie per l'implementazione di memorie non volatili", PierFrancesco Ranaldo, Scuola di Ingegneria e Architettura, Università di Bologna

[13] Appunti del corso "Calcolatori Elettronici", Università di Bologna- didattica.arces.unibo.it/mod/resource/view.php?id=427

[14] www.keysight.com - Datasheet del multimetro digitale Agilent34401A

[15] Appunti del corso "Elettronica dei sistemi digitali", Università di Bologna – www-micro.deis.unibo.it/~romani/Dida01/lezioni/micro.pdf

[16] http://dkc1.digikey.com/jp/ja/tod/texasinstruments/msp430fr57xx-fram-mcu_noaudio/msp430fr57xx-fram-mcu_noaudio.html

[17] www.ti.com/lit/wp/slaa502/slaa502.pdf

[18] ww1.microchip.com/downloads/en/AppNotes/01267a.pdf

[19] www.ti.com/lit/wp/slay015/slay015.pdf

Ringraziamenti

Per primi vorrei ringraziare il Professor Aldo Romani e il Dottor Matteo Filippi per la loro immensa disponibilità a seguirmi durante questo percorso di tesi, aiutandomi sempre nei momenti di maggior difficoltà.

Vorrei ringraziare la mia famiglia, che mi ha sempre sostenuto durante questi anni di Università, credendo in me. Un ringraziamento va anche alla cartoleria "La Gommina" di Muccioli Rossella per non avermi fatto mai mancare le slide e i libri su cui studiare durante i tre anni.

Un GRAZIE va a Camilla, che in questi lunghi anni mi ha sempre supportato, mi è stata vicino, mi ha reso felice nei momenti difficili e mi ha saputo dare la forza di continuare credendo sempre in me e nelle mie potenzialità.

Un grazie indubbiamente va alla "Banda del DAIII LE CARTEEE" Zava, Gro, Sibo, Fede, Bart, Dosio, Valmo e Mazzo con i quali ho passato i più bei momenti universitari, non si vedeva mai l'ora di arrivare in pausa pranzo per smaraffare senza fine!

Infine, un grazie anche a tutti gli amici che mi hanno sostenuto durante tutti questi anni del mio percorso universitario.