ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA, INFORMATICA E TELECOMUNICAZIONI

OTTIMIZZAZIONE DI SISTEMI WAKE-UP RADIO PER APPLICAZIONI RFID ATTRAVERSO L'UTILIZZO DI MICROCONTROLLORI ULTRA-LOW POWER

Elaborato in

Elettronica dei Sistemi Digitali

Relatore:
Prof.**Aldo Romani**

Presentato da:
Michele Monti

Correlatore:
Dott.**Matteo Pizzotti**

Anno Accademico 2015/2016

Sessione III

A mio padre

“ Transmitter! Oh!
Picking up something good.
Hey, radio head!
The sound...
...of a brand-new world. ”

Talking Heads - Radio Head

Ringraziamenti:

A Livia e alla mia famiglia.
Al Professor Aldo Romani,
al Dottor Matteo Pizzotti,
al Dottor Davide Fabbri,
per l’insostituibile aiuto.

Introduzione

La progressiva e sempre più rapida evoluzione dei sistemi a microprocessore, accompagnata da un incremento dell'efficienza energetica che ne ha drasticamente ridotto i consumi, ha reso possibile negli ultimi anni una diffusione sempre più pervasiva dei dispositivi elettronici nell'ambito della sensoristica e del monitoraggio remoto.
Inoltre, lo sviluppo di tecnologie avanzate ha consentito l'affermazione di tecniche in grado di catturare piccole quantità di energia presente nell'ambiente e convertirla in energia elettrica.
La combinazione di questi due aspetti ha posto le basi per lo sviluppo di tecniche di Energy Harvesting, grazie alle quali è oggi possibile progettare e realizzare sistemi complessi sempre più indipendenti dal punto di vista energetico. La possibilità di attingere ad un certo tipo di energia da sorgenti naturali presenti nello stesso luogo in cui è impiegato un dispositivo remoto rappresenta un punto di svolta che permette di eliminare i caratteristici vincoli determinati dalla necessità di disporre di linee di alimentazione o batterie.
Questo scenario ha dato impulso alla progettazione di microcontrollori Ultra-Low Power rivolti all'ottimizzazione del consumo di energia.
Oggetto di studio di questa tesi di laurea è la realizzazione di un firmware dedicato alla sezione logica di controllo di una Wake-up Radio implementata su tag RFID, realizzata attraverso l'utilizzo di microcontrollore Ultra-Low Power.
Per la realizzazione del sistema, si è deciso di utilizzare il microcontrollore Apollo prodotto da Ambiq Micro.
Tale dispositivo, basandosi sull'architettura Subthreshold Power Optimized Technology (SPOT) patentata da Ambiq, utilizza transistor polarizzati in regione di sottosoglia, presentando i migliori valori di efficienza dichiarati sul mercato.
All'interno del primo capitolo si presenterà il sistema, approfondendo la tematica dell'Energy Harvesting e mostrando nel dettaglio le principali tecnologie adottate nel progetto.
Il secondo capitolo fornirà una descrizione del microcontrollore utilizzato, illustrando il funzionamento delle varie periferiche impiegate.
Nel terzo capitolo sarà illustrato il firmware sviluppato, assieme ad una descrizione

delle scelte progettuali adottate, per poi presentarne la validazione funzionale ed energetica.
Nel quarto capitolo saranno infine riportate le conclusioni.

Indice

Capitolo 1

Il Sistema – Wake-Up Radio

In questo elaborato è illustrata l'implementazione del firmware per un modulo Wake-up Radio costituente un Tag RFID dotato di una sezione di energy harvesting in grado di ricavare energia dal segnale radio ad esso indirizzato ed utilizzarla per svolgere le operazioni di ricezione, elaborazione e risposta.

Con il termine Wake-up Radio si indica un dispositivo di ricezione in grado di monitorare il canale di trasmissione, in modo da disattivarsi in assenza di attività di comunicazione ed attivarsi automaticamente al presentarsi di un segnale. La realizzazione di tali apparati consente di mettere in atto tecniche volte alla minimizzazione del consumo energetico, fattore altamente limitante per le applicazioni implementate attraverso l'utilizzo di microcontrollori.

Il modulo, posto all'interno di una rete di sensori wireless (WSN), viene interrogato via radio da un sistema di monitoraggio che trasmette all'interno di una trama seriale UART, modulata con codifica OOK su portante UHF a 900 MHz, un codice identificativo del tag interpellato.

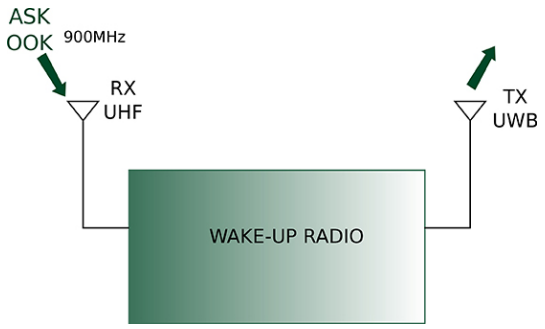

Figura 1.1: Wake-up Radio - Black Box

Quest'ultimo deve avere capacità di indirizzamento: una volta attivato, deve quindi verificare attraverso l'identificativo ricevuto di essere il nodo selezionato e, in caso affermativo, avviare la risposta contenente i dati di localizzazione del tag.

All'interno della rete di sensori wireless è presente, da specifica, un numero minimo di 15 tag, interrogati dalla stazione di monitoraggio ciclicamente con una periodicità di un secondo per l'intero gruppo. Sempre da specifica, è richiesta una precisione di localizzazione di un centimetro e la garanzia di funzionamento ad almeno 10 metri di distanza tra stazione e tag.
É quindi di massima importanza l'ottimizzazione dei consumi in modo da poter garantire lo svolgimento delle operazioni richieste evitando il minimo spreco di energia.

Le operazioni che la Wake-up Radio deve eseguire sul segnale ricevuto sono raggruppabili in tre macroblocchi:
Il primo relativo alle procedure di energy harvesting necessarie per estrarre dal segnale l'energia da rendere disponibile al blocco logico.
Il secondo relativo al raddrizzamento del segnale ricevuto. Tramite questo sarà estratta la trama uart da decodificare.
Infine il terzo blocco, relativo alla logica di addressing (decodifica e confronto) della trama UART e risposta tramite backscattering.

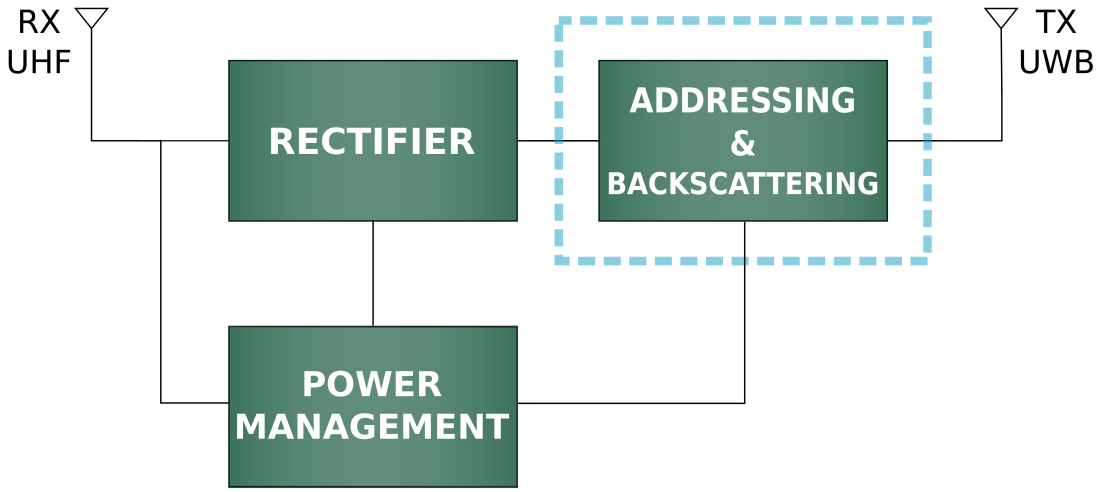

Figura 1.2: Wake-up Radio - Funzioni elementari

In questo contesto, la scelta di utilizzare l'UART per le operazioni di indirizzamento e decodifica rappresenta un approccio innovativo che consente di utilizzare i moduli hardware del microcontrollore in maniera particolarmente efficiente, in modo da ottimizzare il consumo di potenza.
L'ultimo blocco funzionale è quello di competenza del microcontrollore ed è sulla progettazione del software di gestione di questo che verterà il progetto.

1.1 Energy Harvesting

Con il termine energy harvesting si identificano tecniche in grado di utilizzare fonti energetiche non convenzionali per alimentare dispositivi elettronici in contesti sprovvisti di sorgenti tradizionali, eliminando inoltre la necessità di collegamenti elettrici e batterie.
Un sistema dedicato ai processi di energy harvesting, detto harvester, si occupa generalmente di convertire piccole quantità di energia di diversa natura in energia elettrica, immagazzinarla in un accumulatore - che ad esempio può essere un condensatore oppure una batteria a stato solido - e gestire efficacemente l'energia accumulata, occupandosi inoltre della protezione dei dispositivi di accumulo.

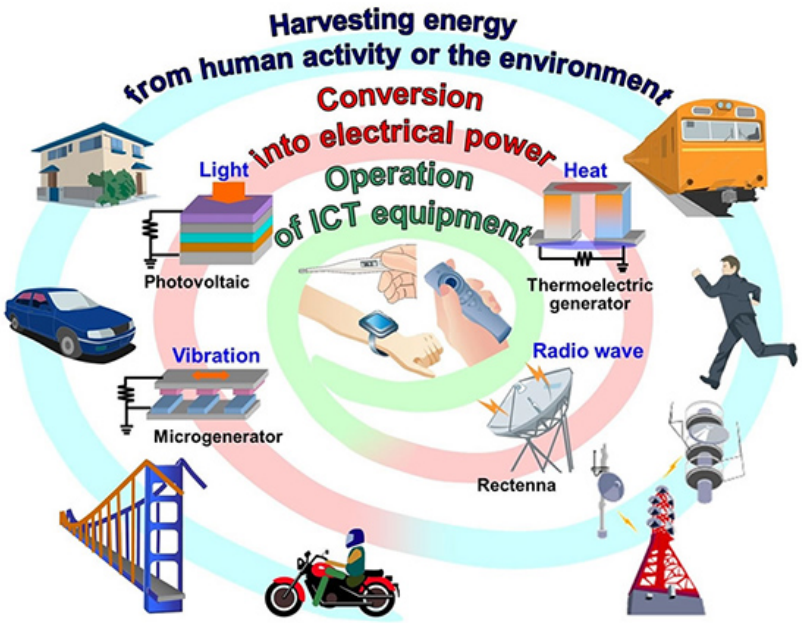

Figura 1.3: Panoramica delle principali tipologie di Energy Harvesting *(http://www.fujitsu.com/global/Images/20101209-01al.jpg)*

Da un certo punto di vista, l'energia ottenuta con tali tecniche può essere considerata energia a costo nullo, ma la convenienza di tali soluzioni è da ricercare soprattutto nella possibilità di evitare l'utilizzo di costose batterie e linee di alimentazione, rendendo i sistemi maggiormente efficienti ed indipendenti.
Le fonti di energia disponibili nell'ambiente di applicazione dei dispositivi possono essere di varia natura. Da fonti luminose è infatti possibile estrarre energia utilizzando celle fotovoltaiche. Elementi piezoelettrici rendono possibile convertire energia meccanica da vibrazione, pressione e movimento. Nel caso di gradienti termici è possibile utilizzare generatori termoelettrici (TEG) o piroelettrici.
L'energia trasportata da onde elettromagnetiche può essere inoltre raccolta utilizzan-

do rectenne sia per frequenze nel campo del visibile come nel caso delle nantenne sia a radiofrequenza come nel caso di dispositivi RFID. Esistono infine dispositivi in grado di estrarre energia prodotta biochimicamente per dispositivi biomedicali impiantabili.

1.2 Tecnologia RFID

La tecnologia RFID (Radio-Frequency IDentification) rende possibile l'utilizzo di campi elettromagnetici per l'identificazione di oggetti, l'acquisizione automatica di dati e la loro memorizzazione attraverso l'utilizzo di particolari transponder, detti tag.

Un sistema RFID è costituito da tre elementi fondamentali:

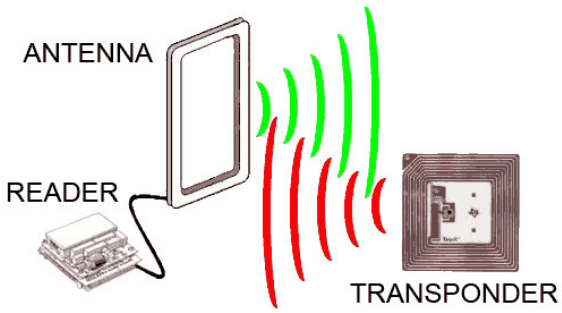

Figura 1.4: Sistema RFID *(© 2016 AHG, Inc. - http://www.ahg.com/business-mobile-apps-blog/mobile-asset-tracking-technologies.html)*

- un dispositivo di lettura e/o scrittura, detto lettore, composto da una ricetrasmittente ed un'antenna
- un insieme di tag, ognuno dei quali è composto da un microchip dedicato alla gestione della logica e dotato di supporto di memorizzazione dei dati, opzionalmente una batteria ed infine un substrato fisico che funge da supporto.
- un sistema di gestione dei dati.

Il lettore si occupa di generare ed inviare un segnale radio in grado di attivare uno o più tag. Una volta interrogato, il transponder è in grado di verificare tramite confronto col proprio codice identificativo se il segnale ricevuto è diretto a lui e, in caso affermativo, procedere con l'invio al lettore dei dati memorizzati.

Il tag è generalmente contenuto in supporti di vario tipo, quali capsule di vetro, piccoli bottoni plastici oppure etichette adesive, in modo da poter essere applicato

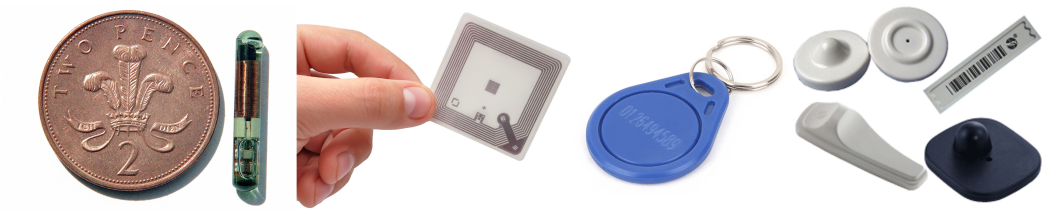

Figura 1.5: Principali tipologie di tag RFID *(fonte Google images)*

ai diversi tipi di oggetto per cui è necessario effettuare una localizzazione, una identificazione o una raccolta di dati.
In base al tipo di alimentazione, tali tag sono detti attivi se alimentati da una batteria, passivi qualora l'alimentazione del circuito interno fosse ricavata automaticamente dal campo elettromagnetico generato dal lettore, semi-passivi se dotati di batteria utilizzabile unicamente per l'alimentazione del chip ma non per alimentare il trasmettitore, oppure semi-attivi se dotati di batteria che alimenta sia il chip sia il trasmettitore ma attivata solo in corrispondenza dell'interrogazione da parte del ricevitore in modo da poter ottimizzare il consumo della carica.
Un altro tipo di classificazione è fatta in base al tipo di accoppiamento tramite il quale avviene la comunicazione ed il trasferimento di potenza tra tag e lettore.
Il tipo di accoppiamento incide direttamente sulla robustezza del collegamento e quindi sulla massima distanza tra tag e lettore. I tipi di accoppiamento più comuni sono l'accoppiamento induttivo, l'accoppiamento magnetico, quello capacitivo e l'accoppiamento in backscattering.
Attraverso l'accoppiamento induttivo il lettore alimenta il tag attraverso un'antenna che genera un campo magnetico. Tale campo magnetico determina per induzione una corrente all'interno di una spira presente sul tag, in maniera del tutto simile a quanto accade all'interno degli avvolgimenti di un comune trasformatore. Questo tipo di tag è generalmente di tipo passivo o semi-passivo, e determina distanze di funzionamento equiparabili alle dimensioni dell'antenna, tipicamente tra i 10 e i 40 centimetri.
L'accoppiamento magnetico è un tipo di accoppiamento a corto raggio, utilizzato tipicamente per la lettura di smart card, simile all'induttivo, in cui lettore e tag si comportano come due avvolgimenti di un trasformatore. La principale differenza tra i due tipi di accoppiamento citati sta nel fatto che in questo caso l'avvolgimento del lettore è realizzato con un nucleo di ferrite circolare o a forma di U sul quale sono poste le spire. Nello specifico, questo tipo di accoppiamento è molto robusto e consente un trasferimento di potenza ottimale, in grado di alimentare chip complessi,

ma la distanza tra lettore e tag deve essere inferiore al centimetro.
Così come l'induttivo, l'accoppiamento capacitivo è un altro tipo di accoppiamento a corto raggio. In questo caso l'antenna è sostituita da elettrodi. Lettore e tag sono dotati di piastre conduttive che assieme formano un condensatore quando poste parallelamente tra loro senza venire in contatto. Anche in questo caso l'accoppiamento è molto robusto e può garantire la corretta alimentazione di circuiti complessi.
L'accoppiamento in backscattering offre una soluzione elegante per la realizzazione di tag RFID sprovvisti di batteria. Questo tipo di accoppiamento si basa sulla retrodiffusione delle onde radio trasmesse dal lettore. Le onde sono riflesse verso la sorgente per trasportare il segnale mantenendone la frequenza ma modificandone il contenuto informativo. È importante notare che solo la parte di segnale viene riflessa alla sorgente, mentre il contenuto di potenza viene utilizzato dal tag per ricavare l'energia necessaria al funzionamento della circuiteria interna al tag.
Altro aspetto caratterizzante è la capacità di memorizzazione ed elaborazione di dati. I tag più semplici sono in grado di memorizzare un singolo bit ed i sistemi basati su questa tipologia possono unicamente riconoscere la presenza o l'assenza del dispositivo interrogato, senza possibilità di riconoscimento specifico dell'oggetto "etichettato". Aumentando la capacità di memorizzazione, ottenendo tag in grado di memorizzare fino a kilobyte di dati, è possibile implementare funzionalità più elaborate. Ad una maggiore capacità di memorizzazione corrisponde però un aumento dei consumi e del costo.
In base alle frequenze di lavoro è poi possibile distinguere diverse tipologie di tag, molte delle quali normate da standard ISO.
Si vuole infine sottolineare che l'estrema flessibilità della tecnologia RFID rende questo tipo di dispositivi soggetto a numerose e frequenti evoluzioni, aprendo il campo a nuovi scenari e rendendo quindi possibile la realizzazione di nuove tipologie e classificazioni oltre a quelle citate.

1.3 Comunicazione UART (Universal Asinchronous Receiver/Transmitter)

I dispositivi UART (Universal Asynchronous Receiver/Transmitter) sono apparati hardware dedicati alla trasmissione seriale asincrona, generalmente implementati su circuito integrato e inseriti all'interno dei microcontrollori, che permettono di configurare il formato dei dati trasmessi e la velocità di comunicazione.
Tali dispositivi sono utilizzati in congiunzione con vari tipi di standard di comunicazione, come ad esempio il TIA RS-232, e si occupano di veicolare la trasmissione di

dati attraverso la porta seriale di computer e periferiche. I byte di dati da trasmettere sono posti in ingresso all'UART di trasmissione, che si occupa di trasmetterne i singoli bit in maniera sequenziale attraverso uno shift register, suddividendoli precedentemente in trame. A lato ricevitore, un secondo UART si occupa di riassemblare i bit e ricomporre le trame trasmesse.
La comunicazione attraverso UART può essere di tre tipi: simplex se unidirezionale, full duplex se entrambi i dispositivi ricevono e trasmettono contemporaneamente oppure half duplex se i dispositivi possono sia ricevere che trasmettere a turno.
Il flusso di dati è asincrono, dal momento che non è presente un segnale di clock comune a trasmettitore e ricevitore, e per l'allineamento ed il controllo della comunicazione i dati sono organizzati in trame all'interno delle quali sono presenti dei bit aggiuntivi (start bit e stop bit per identificare inizio e fine trasmissione, bit di parità per la rilevazione degli errori). In *figura 1.4* è riportato un esempio di trama UART.

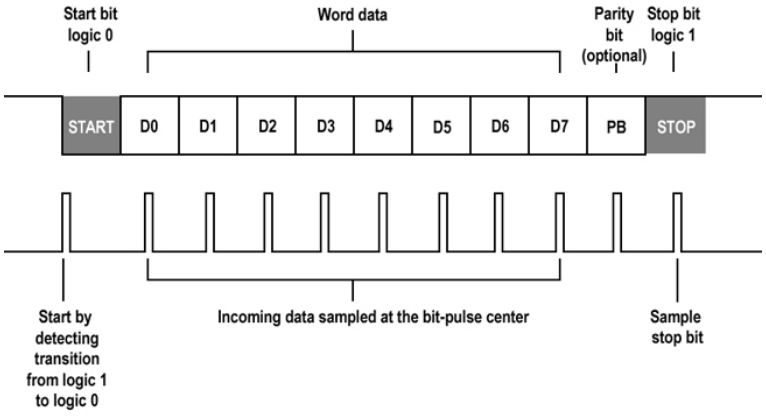

Figura 1.6: Struttura trama UART *(https://electricimp.com/docs/resources/uart/)*

Ogni trama inizia con uno start bit che segnala al ricevitore l'invio dei dati. È poi presente un numero di bit variabile seguito opzionalmente da un bit di parità ed infine uno o più bit di stop, corrispondenti ad un livello logico alto imposto sulla linea.
Non essendo presente una sincronizzazione, è di fondamentale importanza la fase di configurazione della comunicazione, attraverso la quale deve essere garantita compatibilità tra trasmissione e ricezione per quanto riguarda la baud-rate BR, il numero di bit contenuti in ogni trama e la presenza o meno del bit di parità. Nello stato di attesa, la linea è quindi posta a livello logico alto. Alla rilevazione dello start bit (che può avvenire campionando attraverso dei cicli di polling sul livello oppure abilitando un interrupt attivato sul fronte in discesa sulla linea) ha inizio la trasmissione vera e propria. Tramite questo bit il trasmettitore porta la linea a livello logico basso

per un periodo pari al tempo di bit (Tbit=1/BR), poi inizia l'invio dei dati seriali. Il ricevitore, una volta rilevato lo start-bit, inizia il campionamento alla baud-rate prefissata, allineandosi al trasmettitore, che intanto procede alla trasmissione della trama restante.
Tra i vantaggi di questo tipo di comunicazione si ha la semplicità di implementazione dovuta al limitatissimo numero di linee necessarie, dal momento che non è presente una linea per il segnale di clock. Sono infatti necessari unicamente due fili, per una comunicazione di tipo half-duplex (linea TX/RX e massa), o tre fili, per una comunicazione full-duplex (linea TX, linea RX e massa).
D'altro canto, dal momento che sia ricevitore che trasmettitore sono dotati di oscillatori separati (che quindi non potranno fornire frequenze esattamente uguali), il numero massimo di bit contenuti in una trama è superiormente limitato poiché, all'aumentare del numero di bit, l'eventuale sfasamento tra i clock potrebbe determinare un disallineamento tra ricezione e trasmissione.

Queste caratteristiche rendono l'utilizzo dell'interfaccia UART ottimale per la decodifica dell'indirizzo del TAG.
L'indirizzamento è infatti per sua natura asincrono, e consta di una sola linea dati che può essere connessa alla linea di ricezione dell'UART. Inoltre, la presenza di moduli hardware dedicati all'interno dei microcontrollori per la gestione di questo protocollo rende conveniente il suo utilizzo anche da un punto di vista energetico.

1.4 Funzionamento del sistema

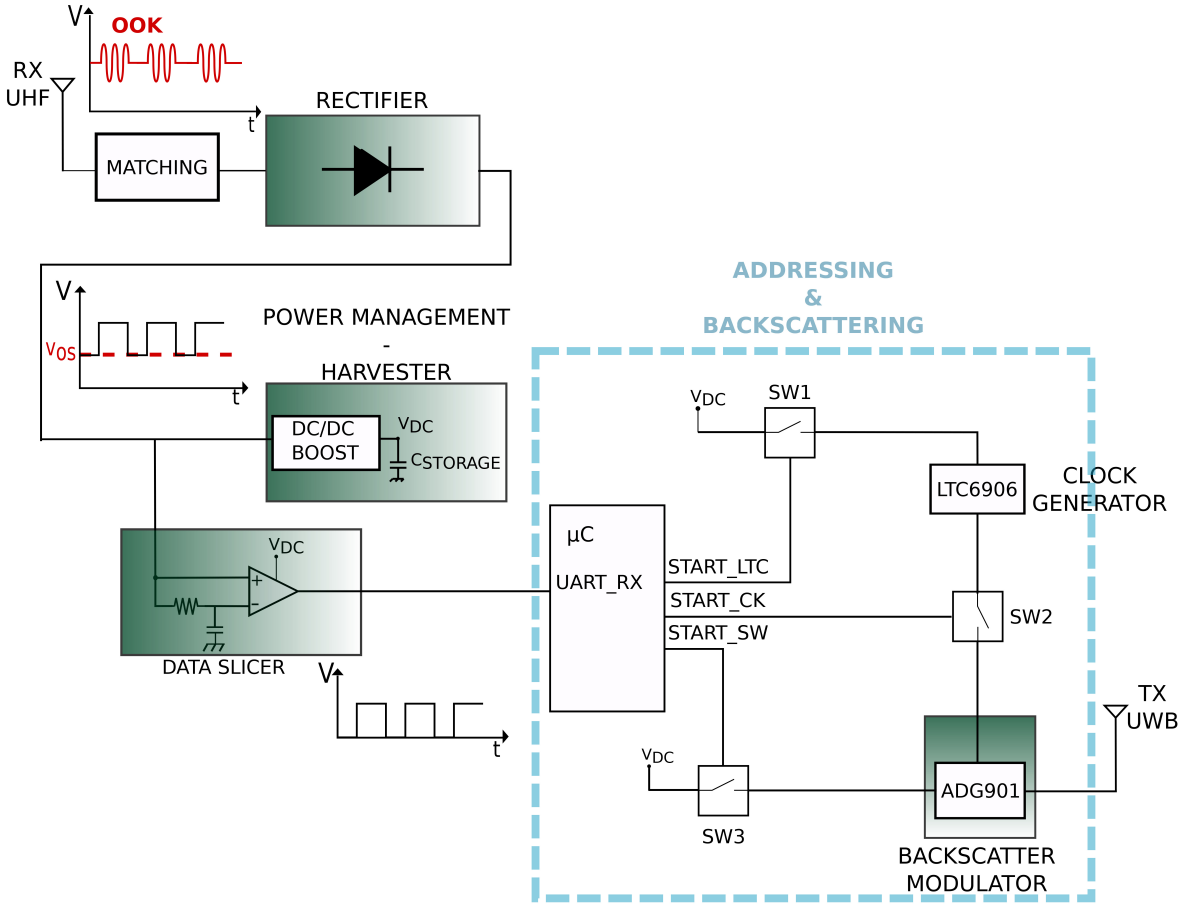

Figura 1.7: Schema a blocchi Wake-up Radio

Il sistema, illustrato in *Figura 1.5*, è composto dalle seguenti sezioni:

- Una sezione di ricezione del segnale, composta da un'antenna UHF seguita da una una rete di matching deputata a massimizzare la potenza ricevuta dall'antenna, realizzando un adattamento tra l'impedenza di quest'ultima e quella del circuito a valle ed un raddrizzatore in grado di demodulare il segnale ricevuto e digitalizzarlo, ricavandone la trama seriale UART;
- Una sezione di energy harvesting composta da un convertitore DC/DC Boost e da una capacità di storage nella quale sarà stoccata l'energia che alimenterà il sistema;
- Una sezione di condizionamento del segnale, rappresentata dal Data Slicer, che si occuperà di eliminare l'offset e adattare il segnale in logica TTL;
- Una sezione dedicata all'addressing della trama ricevuta e alla generazione del segnale di backscattering, il cui elemento principale è un microcontrollore ULP.

Tale microcontrollore è incaricato di attivarsi sul fronte del segnale in ingresso UART_RX, acquisire la trama, effettuare le operazioni di confronto e, in caso di indirizzamento positivo, rispondere con la trasmissione attraverso la rete deputata al backscattering di un segnale PWM su antenna UWB, tornando successivamente in uno stato di inattività.

In corrispondenza della ricezione di un segnale UART, successivamente al risveglio del tag, possono presentarsi quindi due situazioni: una in cui all'interno della trama UART è contenuto il codice identificativo del tag risvegliato oppure una in cui all'interno della trama UART è contenuto un codice identificativo diverso da quello del tag (poichè la richiesta da parte del lettore è diretta ad un altro tag).
Nel primo caso il microcontrollore si occupa di avviare il processo di risposta attivando la rete dedicata al backscattering, disattivandola a fine trasmissione e tornando successivamente in uno stato di inattività
Nel secondo caso il microcontrollore ritorna immediatamente inattivo
La rete per il backscattering, da attivare una volta che il tag interpellato ha verificato l'interrogazione confrontando il codice inviato dalla stazione con il proprio identificativo, è composta da una serie di dispositivi interconnessi tramite interruttori comandati dalle uscite del microcontrollore.
Un primo switch analogico SW1 è utilizzato per fornire l'alimentazione al generatore di clock LTC6906. Questo dispositivo, una volta alimentato, provvede a generare il segnale di clock del modulatore backscatter che sarà stabile dopo un tempo T_{START} massimo di 600 μs.
Gli switch SW2 ed SW3 devono quindi essere attivati con un ritardo di ($<$600 μs) per fornire allo switch ad alta frequenza ADG901 il segnale di clock e l'alimentazione per avviare la generazione del segnale di backscattering.
Il tempo necessario per la trasmissione della risposta completa è di circa 4ms, trascorsi i quali tutti gli switch vanno spenti in modo da disattivare la rete.

L'introduzione di questi switch è necessaria al fine di minimizzare il consumo di potenza nella fase di attesa tra un indirizzamento ed il successivo, disattivando il maggior numero di periferiche possibile e mantenendo attivi solo i moduli necessari a rendere il microcontrollore reattivo nei confronti dell'interrogazione ricevuta tramite UART.
Nei capitoli successivi sono illustrate le caratteristiche del microcontrollore Apollo ed è presentata la realizzazione del firmware progettato in funzione di queste.

Capitolo 2

Il microcontrollore Ambiq Micro - Apollo512-KBR

Il microcontrollore scelto fa parte della famiglia di microcontrollori di ultima generazione altamente integrati APOLLO, prodotti da Ambiq Micro, basati sul processore a 32 bit ARM-Cortex-M4F e dotati di architettura Subthreshold Power Optimized Technology (SPOT). Tale tecnologia, utilizzando transistor polarizzati in regione di sottosoglia, garantisce consumi ridotti ad oggi ineguagliati dagli altri dispositivi presenti sul mercato.

Figura 2.1: Ambiq Micro - Apollo MCU *(http://ambiqmicro.com/)*

Il processore M4F presenta inoltre diverse caratteristiche fondamentali, tra cui il mapping di tutte le periferiche in una singola architettura di memoria da 4 GB, la possibilità di gestire le quattro modalità di funzionamento low-power – Active, Sleep, Deep-Sleep e Power-Off - e la presenza di un avanzato sistema di gestione degli interrupt e degli eventi.

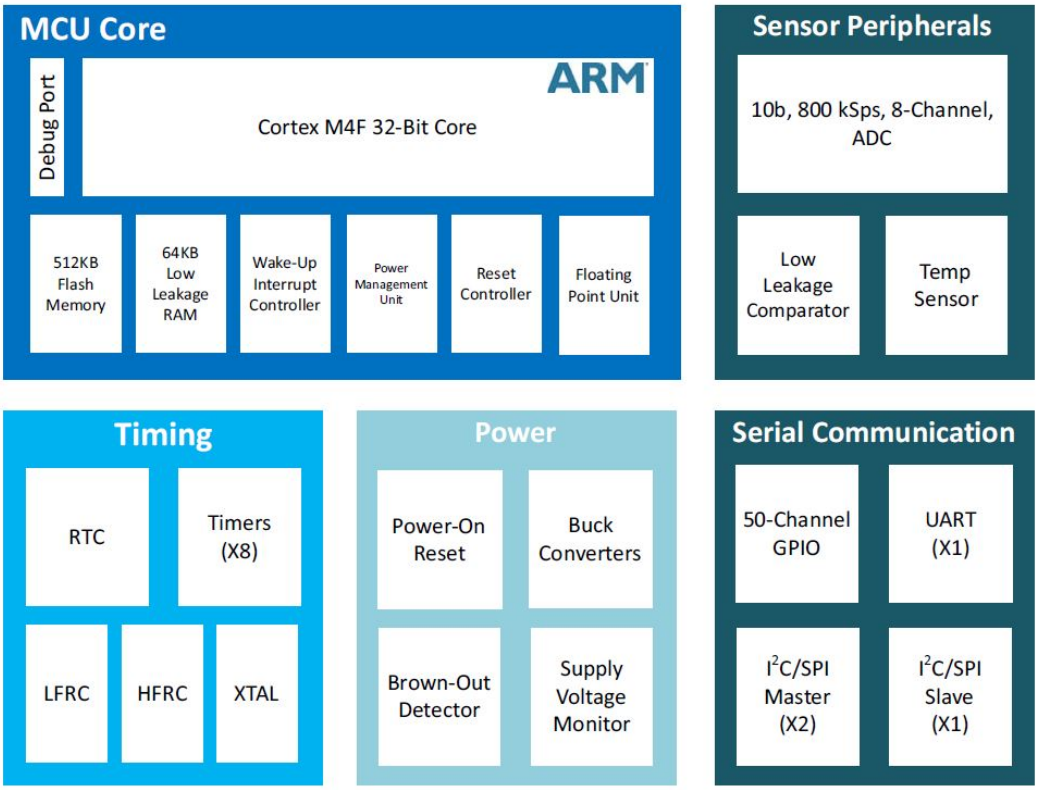

Figura 2.2: Architettura di sistema microcontrollore Apollo *(Apollo MCU Datasheet Rev. 0.9)*

Nello specifico, si è scelto il modello Apollo 512-KBR. Si tratta di un dispositivo a 64 pin con package di tipo BGA. È dotato di 50 ingressi/uscite GPIO, tre sorgenti interne di clock indipen-denti, un orologio in tempo reale (RTC), quattro coppie di timer-contatori a 16 bit asincroni -accoppiabili per ottenere contatori a 32 bit-, un set di periferiche di comunicazione seriale che supportano i protocolli I2C, SPI e UART, un analog-to-digital converter ultra-low power ad approssimazioni successive per il monitoraggio di temperatura, tensioni e fino ad 8 segnali provenienti da sensori esterni, una memoria Flash da 512kB e una RAM a basse perdite da 64kB in cui possono essere memorizzati dati e codice.

I dati sui consumi di corrente forniti dal costruttore sono riportati in*figura 2.3* in relazione alla tensione di alimentazione, alla modalità di funzionamento, alle sorgenti di clock attive e alle dimensioni della ram in cui sono mantenuti i dati.

Symbol	Parameter	Test Conditions[a,b]	VDD*	Typ	Unit
I_{RUNF}	Flash program run current	Executing CoreMark from internal flash memory, HFRC = 24 MHz, all peripherals disabled, buck converters enabled	3.3 V	35	µA/MHz
			1.8 V	51	µA/MHz
I_{RUNF}	Flash program run current	Executing CoreMark from internal flash memory, HFRC = 24 MHz, all peripherals disabled, buck converters disabled	3.3 V	79	µA/MHz
			1.8 V	78	µA/MHz
I_{SLEEP}	Sleep mode current	WFI instruction with SLEEPDEEP = 0, buck converters enabled	3.3 V	57	µA
			1.8 V	56	µA
I_{SLEEP}	Sleep mode current	WFI instruction with SLEEPDEEP = 0, buck converters disabled	3.3 V	63	µA
			1.8 V	57	µA
$I_{DEEPSLEEP}$	Deep Sleep mode current	WFI instruction with SLEEPDEEP = 1	3.3 V	143	nA
			1.8 V	120	nA
$I_{DEEPSLEEP}$	Deep Sleep mode current with RTC active, clocked from 32.768 kHz crystal	WFI instruction with SLEEPDEEP = 1	3.3 V	419	nA
			1.8 V	TBD	nA
$I_{DEEPSLEEP}$ with 8 KB of RAM	Deep Sleep mode current with 8 KB of RAM retained	WFI instruction with SLEEPDEEP = 1	3.3 V	193	nA
			1.8 V	170	nA
$I_{DEEPSLEEP}$ with 16 KB of RAM	Deep Sleep mode current with 16 KB of RAM retained	WFI instruction with SLEEPDEEP = 1	3.3 V	243	nA
			1.8 V	220	nA
$I_{DEEPSLEEP}$ with 32 KB of RAM	Deep Sleep mode current with 32 KB of RAM retained	WFI instruction with SLEEPDEEP = 1	3.3 V	293	nA
			1.8 V	270	nA
$I_{DEEPSLEEP}$ with 64 KB of RAM	Deep Sleep mode current with 64 KB of RAM retained	WFI instruction with SLEEPDEEP = 1	3.3 V	343	nA
			1.8 V	320	nA

a. Core clock (HCLK) is 24 MHz for each parameter unless otherwise noted.
b. All values measured at 25°C

Figura 2.3: Consumi di corrente microcontrollore Apollo *(Apollo MCU Datasheet Rev. 0.9)*

Le correnti dichiarate vanno da un massimo di 79uA/MHz in modalità Active con periferiche e buck converter disattivati, ad un minimo di 120nA in modalità Deep Sleep senza ritenzione di dati in memoria.
Il microcontrollore contiene all'interno due convertitori DC-DC Buck ottimizzati per applicazioni low-power, caratterizzati da bassissimi valori di corrente di quiescenza e dotati di switch di potenza integrati. É quindi necessario inserire solamente un induttore ed un condensatore esterni per ogni convertitore per la generazione ad alta efficienza dell'alimentazione di CPU, SRAM, memoria Flash, blocchi logici e logica digitale. Questo permette al core del microcontrollore di lavorare a tensioni inferiori di 1,8V, riducendo i consumi. In *figura 2.4* è riportato lo schema fornito dal data sheet contenente la configurazione circuitale da implementare.

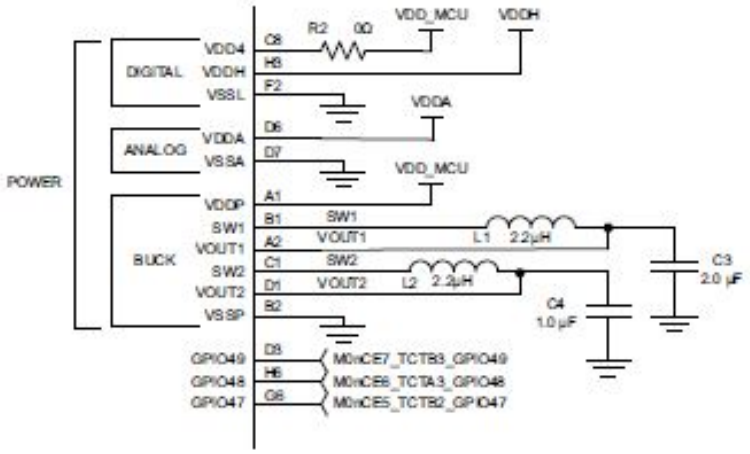

Figura 2.4: Schema di set-up dei convertitori Buck *(Apollo MCU Datasheet Rev. 0.9)*

2.1 Unità Power Management

L'unità di power management (PMU) è una macchina a stati finiti che controlla le transizioni tra le varie modalità di funzionamento del microcontrollore. Passando dalla modalità Active alla modalità Deep Sleep, questa unità gestisce lo spegnimento del regolatore di tensione del microcontrollore e, in associazione col controller degli interrupt di wake-up -Wake-Up Interrupt Controller-, la ritenzione degli stati dei registri.
Una volta entrato in modalità Deep Sleep, la PMU, in congiunzione con il Wake-Up Interrupt Controller, resta in attesa di un evento di wakeup. Quando tale evento si presenta, la PMU avvia il processo di riattivazione delle alimentazioni riabilitando il regolatore di tensione presente nel chip e procede al ripristino del registro di stato della CPU. Nel momento in cui tutti gli stati sono ripristinati, l'M4F ritorna nella modalità Active.

Symbol	Parameter	Test Conditions	Min	Typ	Max	Unit
Buck mode						
$T_{RUN\_TO\_SLEEP}$	Run to Sleep mode transition time	HCLK frequency = 24 MHz	-	200	240	ns
$T_{RUN\_TO\_DEEPSLEEP}$	Run to Deep Sleep mode transition time	HCLK frequency = 24 MHz	-	3.2	3.7	µs
$T_{SLEEP\_TO\_RUN}$	Sleep to Run mode transition time		-	450	575	ns
$T_{DEEPSLEEP\_TO\_RUN}$	Deep Sleep to Run mode transition time		-	37.5	43	µs
LDO mode						
$T_{RUN\_TO\_SLEEP}$	Run to Sleep mode transition time	HCLK frequency = 24 MHz	-	200	240	ns
$T_{RUN\_TO\_DEEPSLEEP}$	Run to Deep Sleep mode transition time	HCLK frequency = 24 MHz	-	3.2	3.7	µs
$T_{SLEEP\_TO\_RUN}$	Sleep to Run mode transition time		-	450	575	ns
$T_{DEEPSLEEP\_TO\_RUN}$	Deep-Sleep to Run mode transition time		-	13.5	15.5	µs

Figura 2.5: Tempi di transizione tra modalità operative microcontrollore Apollo *(Apollo MCU Datasheet Rev. 0.9)*

In*figura 2.5* sono riportate i tempi di transizione fra le modalità operative.
I valori fanno riferimento a due modalità di funzionamento del regolatore di tensione, la modalità Buck e la modalitàO LDO. Nella prima sono attivi sia i convertitori Buck che i convertitori lineari a basso rumore LDO, nella seconda, è possibile disattivare i convertitori Buck quando si entra in modalità Deep Sleep, garantendo il minor consumo energetico con correnti di quiescenza nell'ordine dei nA.

Di seguito sono illustrate le modalità operative nel dettaglio.

2.1.1 Active Mode

In Active mode il processore è alimentato, i clock (FCLK, HCLK) sono attivi e le istruzioni sono eseguite. In questa modalità l'M4F esige l'accensione e l'abilitazione del clock per ogni dispositivo collegato al bus di sistema per il normale accesso. Per passare da questa modalità ad una qualsiasi delle modalità low-power viene eseguita dal processore una determinata sequenza di istruzioni ed in base al valore assunto da specifici bit contenuti all'interno del System Control Register (SCR) è selezionata una specifica modalità operativa. Nello specifico, una volta impostato il valore dell'SCR, il programma può accedere agli stati low-power attraverso i seguenti metodi

- Esecuzione di un'istruzione di Wait-For-Interrupt (WFI);
- Settaggio del bit SLEEPONEXIT del registro SCR, in modo da ritornare automaticamente nello stato di "sleep" al termine delle routine di servizio degli interrupt (ISR)

Una volta entrato in modalità low-power, nel caso in cui tutte le condizioni necessarie siano soddisfatte, il processore rimane in questo stato fintanto che un evento non determini il ritorno in Active Mode. Gli eventi che possono determinare un ritorno in Active Mode sono:

- Un reset;
- Un interrupt abilitato ricevuto dall'Interrupt Controller (NVIC);
- Un evento di debug ricevuto dalla Debug Access Port (DAP).

2.1.2 Sleep Mode

In questa modalità il processore è alimentato come nel caso precedente ma le sorgenti di clock (HCLK, FCLK) sono disattivate. L'alimentazione è applicata al processore in modo da poterlo attivare immediatamente al verificarsi di un evento di wakeup e renderlo in grado di eseguire le istruzioni.

2.1.3 Deep-Sleep Mode

Nella modalità Deep Sleep il processore entra in modalità State Retention Power Gating (SRPG), dove l'alimentazione principale è rimossa ma i flip-flop mantengono il loro stato. I clock sono inattivi e le sorgenti di clock possono essere disattivate. Per semplificare la rimozione dell'alimentazione ed entrare in modalità SRPG, il processore effettua una procedura di handshake con il Wake-up Interrupt Controller e con l'unità di power management per impostare le condizioni di wakeup.

2.2 Modulo per la generazione del clock

Il modulo per la generazione del clock fornisce tutte le sorgenti di clock necessarie al microcontrollore. Questi clock sono derivati da uno dei tre oscillatori interni presenti: un oscillatore al quarzo ad alta precisione (XT), un oscillatore RC a bassa potenza con frequenza nominale pari a 1kHz (LFRC) ed un oscillatore RC ad alta frequenza con valore nominale pari a 24MHz (HFRC). Ognuno di questi può essere sostituito con un clock esterno per effettuare test e misurazioni

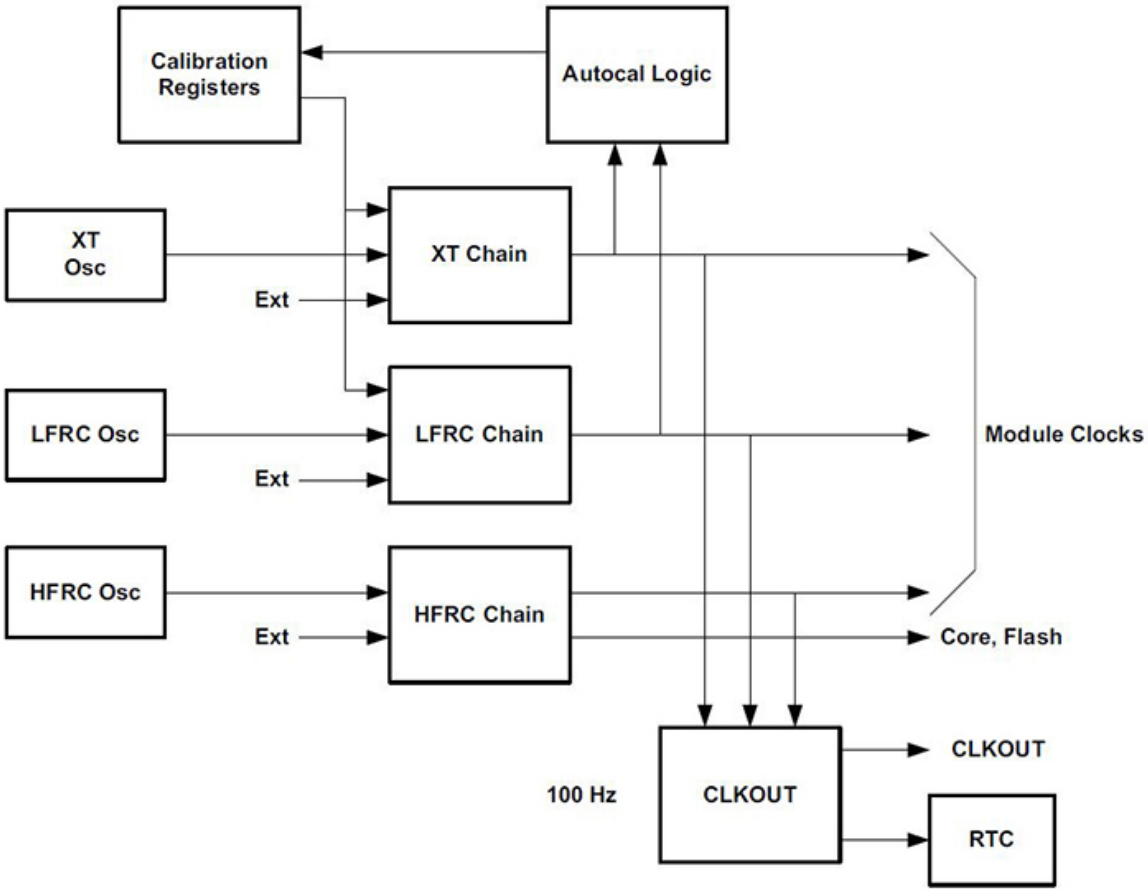

Figura 2.6: Schema a blocchi generatore di clock e real-time clock *(Apollo MCU Datasheet Rev. 0.9)*

È possibile assegnare ad un pin un segnale di clock CLKOUT in modo da poter pilotare periferiche esterne. CLOCKOUT pilota anche il modulo di clock real time (RTC) utilizzato per generare ora e data. Il generatore di clock è in grado di controllare automaticamente l'abilitazione degli oscillatori. In tal modo questi ultimi sono alimentati solo quando richiesto dalle periferiche che li utilizzano, così da minimizzare il consumo di potenza senza la necessità di intervento da parte del software.

2.2.1 High Precision XT Oscillator (XT)

L'oscillatore a cristallo XT è accordato ad un quarzo da 32 kHz esterno ed ha una frequenza nominale di 32,768kHz. È utilizzato quando la precisione a breve termine è di estrema rilevanza. Dal momento che un cristallo esterno consuma potenza in maniera significativa, questo oscillatore è attivato solo quando utilizzato da un modulo interno.

2.2.2 Low Frequency RC Oscillator (LFRC)

Questo oscillatore, con una frequenza nominale di 1024 Hz, è utilizzato quando l'accuratezza del valore di frequenza non è di fondamentale importanza. Fornisce il clock necessario ad alcune macchine a stati di base ed è sempre attivato. Si tratta dell'oscillatore meno dispendioso in termini di potenza.

2.2.3 High Frequency RC Oscillator (HFRC)

L'oscillatore ad alta frequenza presenta una frequenza nominale di 24 MHz ed è utilizzato per fornire tutti i clock ad alta frequenza necessari all'interno del microcontrollore ed il clock di sistema del processore ARM. Dal momento che i valori di cycle-to-cycle jitter sono tipicamente importanti per questo oscillatore e una precisione assoluta non è genericamente richiesta, per il segnale generato non è previsto nessuna funzione di calibrazione. L'HFRC è attivato unicamente quando richiesto da un modulo interno. Quando il processore entra in modalità Sleep o Deep Sleep, l'oscillatore viene automaticamente spento, a meno che non sia utilizzato da uno degli altri moduli. Alla riattivazione dell'oscillatore la frequenza del segnale generato non è istantaneamente stabile, perciò sono presenti porte logiche interne di interfaccia che lo rendono disponibile solo una volta stabilizzato.

Address(s)	Register Name	Description
0x40004000	CALXT	XT Oscillator Control
0x40004004	CALRC	RC Oscillator Control
0x40004008	ACALCTR	Autocalibration Counter
0x4000400C	OCTRL	Oscillator Control
0x40004010	CLKOUT	CLKOUT Frequency Select
0x40004014	CLKKEY	Key Register for Clock Control Register
0x40004018	CCTRL	HFRC Clock Control
0x4000401C	STATUS	Clock Generator Status
0x40004020	HFADJ	HFRC Adjustment
0x40004024	HFVAL	HFADJ readback
0x40004028	CLOCKEN	Clock Enable Status
0x4000402C	UARTEN	UART Enable
0x40004100	INTEN	CLKGEN Interrupt Register: Enable
0x40004104	INTSTAT	CLKGEN Interrupt Register: Status
0x40004108	INTCLR	CLKGEN Interrupt Register: Clear
0x4000410C	INTSET	CLKGEN Interrupt Register: Set

Figura 2.7: Mapping registri generatore di clock *(Apollo MCU Datasheet Rev. 0.9)*

2.3 Modulo per la configurazione dei GPIO

Il modulo di configurazione dei General Purpose I/O e dei pin del microcontrollore Apollo supervisiona le connessioni di fino a 50 pin analogico-digitali. Ogni pin può essere connesso ad una varietà di segnali di interfaccia e il modulo GPIO gestisce la selezione tra ingressi e uscite ed il controllo di ogni pin; inoltre può essere configurato per funzionare come general purpose input e/o output e per diversi tipi di funzioni esterne. Ogni GPIO può essere configurato per generare un interrupt al presentarsi di un fronte in ingresso.

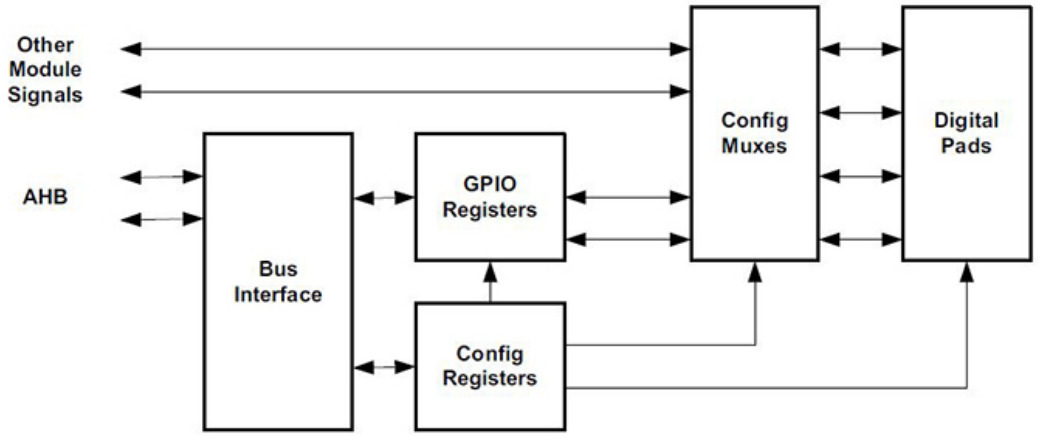

Figura 2.8: Schema a blocchi modulo GPIO *(Apollo MCU Datasheet Rev. 0.9)*

La configurazione della funzione di ogni pin è impostata tramite i registri PADREGy (con y compreso tra A ed M). Per poter scrivere nei registri PADREGy deve essere inserito il valore esadecimale 0x73 nel registro PADKEY. Attraverso il campo PADnFNCSEL è possibile selezionare fino ad otto segnali utilizzabili per ogni pin. Le funzioni a cui è possibile assegnare ogni pin sono raggruppate in moduli, ognuno identificato da un colore nelle tabelle riportate in *figura 2.9* e *figura 2.10* .

Pad	PADnFNCSEL							
	0	1	2	3	4	5	6	7
0	SLSCL [I]	SLSCK [I]	UARTTX [O]	GPIO0	M0SCK_LB	M1SCK_LB	M0SCL_LB	M1SCL_LB
1	SLSDA [S]	SLMISO [O]	UARTRX [I]	GPIO1	M0MISO_LB	M1MISO_LB	M0SDA_LB	M1SDA_LB
2	SLWIR3 [S]	SLMOSI [I]	CLKOUT	GPIO2	M0MOSI_LB	M1MOSI_LB	M0WIR3_LB	M1WIR3_LB
3*	TRIG0 [I]	SLnCE [O]	M1nCE4	GPIO3	M0nCE_LB	M1nCE_LB		
4*	TRIG1 [I]	SLINT [O]	M0nCE5	GPIO4	SLINTGP_LB	SWO [O]	CLKOUT	
5	M0SCL [S]	M0SCK[O]	UARTS [O]	GPIO5	M0SCK_LB		M0SCL_LB	
6	M0SDA [S]	M0MISO [I]	UACTS [I]	GPIO6	SLMISO_LB		SLSDA_LB	
7	M0WIR3 [S]	M0MOSI [O]	CLKOUT	GPIO7			SLWIR3_LB	
8	M1SCL [S]	M1SCK [O]	M0nCE4	GPIO8		M1SCK_LB		M1SCL_LB
9	M1SDA [S]	M1MISO [I]	M0nCE5	GPIO9		SLMISO_LB		SLSDA_LB
10	M1WIR3 [S]	M1MOSI [O]	M0nCE6	GPIO10				SLWIR3_LB
11**	RESERVED	M0nCE0	CLKOUT	GPIO11				
12	ADC0 [A]	M1nCE0	TCTA0	GPIO12				
13	ADC1 [A]	M1nCE1	TCTB0	GPIO13			SWO [O]	
14	ADC2 [A]	M1nCE2	UARTTX [O]	GPIO14				
15	ADC3 [A]	M1nCE3	UARTRX [I]	GPIO15				
16	ADCREF [A]	M0nCE4	TRIG2 [I]	GPIO16				
17	CMPAD0 [A]	M0nCE1	TRIG3 [I]	GPIO17				
18	CMPAD1 [A]	M0nCE2	TCTA1	GPIO18				
19	CMPRF0 [A]	M0nCE3	TCTB1	GPIO19				
20	SWDCK [I]	M1nCE5	TCTA2	GPIO20				
21	SWDIO [S]	M1nCE6	TCTB2	GPIO21				
22	UARTTX [O]	M1nCE7	TCTA3	GPIO22				
23	UARTRX [I]	M0nCE0	TCTB3	GPIO23				
24		M0nCE1	CLKOUT	GPIO24				
25		M0nCE2	TCTA0	GPIO25				
26		M0nCE3	TCTB0	GPIO26				
27		M1nCE4	TCTA1	GPIO27				
28		M1nCE5	TCTB1	GPIO28				
29	ADC4 [A]	M1nCE6	TCTA2	GPIO29				
30	ADC5 [A]	M1nCE7	TCTB2	GPIO30				
31	ADC6 [A]	M0nCE4	TCTA3	GPIO31				
32	ADC7 [A]	M0nCE5	TCTB3	GPIO32				
33	CMPRF1 [A]	M0nCE6		GPIO33				
34	CMPRF2 [A]	M0nCE7		GPIO34				
35		M1nCE0	UARTTX [O]	GPIO35				
36		M1nCE1	UARTRX [I]	GPIO36				
37	TRIG0 [I]	M1nCE2	UARTS [O]	GPIO37				
38	TRIG1 [I]	M1nCE3	UACTS [I]	GPIO38				
39	TRIG2 [I]	UARTTX [O]	CLKOUT	GPIO39				
40	TRIG3 [I]	UARTRX [I]		GPIO40				
41	TRIG4 [I]		SWO [O]	GPIO41				
42	TRIG5 [I]	M0nCE0	TCTA0	GPIO42				
43	TRIG6 [I]	M0nCE1	TCTB0	GPIO43				
44	TRIG7 [I]	M0nCE2	TCTA1	GPIO44				
45		M0nCE3	TCTB1	GPIO45				
46		M0nCE4	TCTA2	GPIO46				
47		M0nCE5	TCTB2	GPIO47				
48		M0nCE6	TCTA3	GPIO48				
49		M0nCE7	TCTB3	GPIO49				

Figura 2.9: Mappa delle funzionalità assegnabili ai pin *(Apollo MCU Datasheet Rev. 0.9)*

Color/ Symbol	Function	Pad Type
	Various	Supported on CSP package
	ADC Signals	Analog or Input, as indicated by [A] or [I] respectively
	I^2C/SPI Slave Signals	Input, Special or Push-pull output, as indicated by [I], [S] or [O] respectively,
	I^2C/SPI Master 0 Signals	Input, Special or Push-pull output, as indicated by [I], [S] or [O] respectively
	I^2C/SPI Master 1 Signals	Input, Special or Push-pull output, as indicated by [I], [S] or [O] respectively
	GPIO Signals	Controlled by GPIO Configuration
	Counter/Timer Signals	Controlled by CTIMER Configuration
	UART Signals	Input or Push-pull output, as indicated by [I] or [O] respectively
	CLKOUT Signals	Push-pull Output
	Loopback Connections	Tri-state
	Miscellaneous Signals	Input Special or Push-pull output, as indicated by [I], [S] or [O] respectively
*	High-side power switch	Pads with a (*) (pads 3 and 4) have selectable high side power switch transistors to provide ~1 Ω switches to VDDP.
**	Low-side power switch	Pads with a (**) (pad 11) have selectable low side power switch transistors to provide ~1 Ω switches to VSS.

Figura 2.10: Codice colori funzionalità *(Apollo MCU Datasheet Rev. 0.9)*

Attraverso i bit del campo PADnSTRNG all'interno del registro PADREGy viene controllata l'intensità di pilotaggio del pin. Per tutti i pin, eccetto il pin 20, il bit PADnPULL abilita un pull-up debole se settato, mentre per il pin 20 (PAD20PULL), nella stessa configurazione, abilita un pull-down debole.
Il bit PADnINPEN deve essere posto a uno per abilitare l'input sul rispettivo pin e deve essere lasciato a zero qualora il pin fosse inutilizzato, in modo da eliminare le correnti di leakage minimizzando i consumi.
I pin 3 e 4 hanno ognuno integrato uno switch a transistor che fornisce il collegamento all'alimentazione dei buck converter tramite una resistenza di circa 1 Ohm. Il pin 11 ha invece integrato uno switch a transistor che fornisce la connessione tramite una resistenza da 1 Ohm alla massa VSS. Sono poi presenti resistenze di pull-up opzionalmente attivabili sui pin 5, 6, 8 e 9, utilizzabili per implementare il protocollo di trasmissione I^2C senza dover inserire resistori esterni.

2.3.1 Configurazione ingressi e uscite per funzionalità GPIO

Ogni pin, impostando il campo PADnFNCSEL al valore 0x3, viene collegato al corrispondente segnale GPIO.
Attraverso il campo GPIOCFGy è possibile configurare le GPIO come ingresso o uscita e, nel secondo caso, può essere selezionata un'uscita di tipo push-pull, open drain, tri-state o disabilitata, attraverso il campo OUTCFG. Se l'output è configurato come open drain ed il bit corrispondente all'interno del registro GPIOWT è uno zero, il pin viene posto a livello logico basso. Se tale bit è invece a uno, il corrispondente pin viene lasciato flottante. Nel caso in cui l'uscita fosse invece configurata come tri-state al settaggio del bit corrispondente all'interno del registro, il pin viene posto a livello logico alto; se posto a zero, il pin viene lasciato flottante.
Tutti gli input sono leggibili in ogni istante, anche se il pin non è configurato come GPIO.
I valori assunti dai pin sono raggruppati e leggibili nei due registri GPIORDA (dal pin 0 al pin 31) e GPIORDB (dal pin 32 al pin 49). Impostando a uno il bit nINCFG, all'interno del registro CFGy, dal corrispondente GPIO verrà sempre letto uno zero.
Le uscite GPIO sono controllate dai registri GPIOWTA, GPIOWTB, GPIOENA e GPIOENB. È possibile scrivere e leggere direttamente da questi registri. Dal momento che ogni GPIO generalmente svolge una funzione indipendente dalle altre, esiste la possibilità di porre a uno o zero uno o più bit del registro senza dover effettuare un'operazione di lettura-modifica-scrittura. Scrivendo un dato sui registri di set, GPIOWTSA e GPIOWTSB, i bit posti a uno determineranno la scrittura di un uno nei corrispondenti bit dei registri GPIOWTA/B, i bit posti a zero invece determineranno il mantenimento del valore precedentemente assunto dal corrispondente bit. Viceversa, scrivendo un dato sui registri di clear, GPIOWTCA e GPIOWTCB, i bit posti a uno determineranno la scrittura di uno zero nei corrispondenti bit dei registri GPIOWTA/B, mentre gli altri bit saranno mantenuti al valore assunto precedentemente.
Per i pin configurati in modalità di uscita tri-state, i registri GPIOENA/B gestiscono l'abilitazione di ogni bit. È possibile scrivere direttamente su tali registri ed i relativi bit possono essere posti a uno o a zero scrivendo sui registri GPIOENSA/B e GPIOENCA/B nelle corrispondenti posizioni.
Ogni pin può inoltre essere configurato per generare un interrupt in corrispondenza di una transizione del livello logico. L'interrupt sarà generato anche se per il pin non è impostato come GPIO. Ogni interrupt è abilitabile, disabilitabile, cancellato o settato attraverso l'utilizzo dei rispettivi bit IER, ISR, WCR e WSR contenuti all'interno dei registri GPIOA (per i pin compresi tra 0 e 31) e GPIOB (per i pin compresi tra 32 e 49).

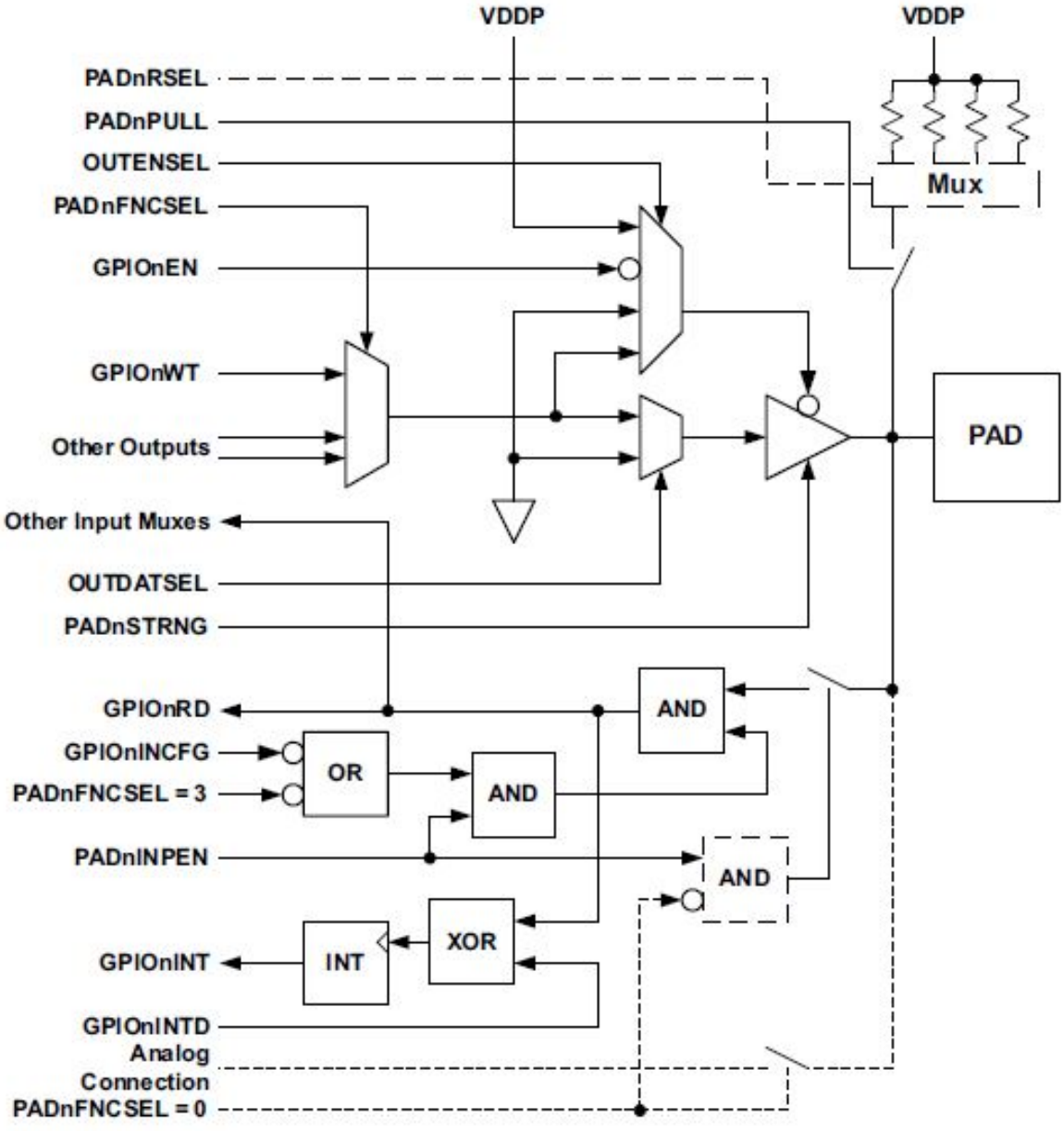

Figura 2.11: Schema di implementazione del collegamento ai singoli pin *(Apollo MCU Datasheet Rev. 0.9)*

In *figura 2.11* è riportato nel dettaglio lo schema di connessione implementato per ogni pin.
È possibile connettere ai pin diversi tipi di moduli interni per le relative funzioni, come ad esempio il modulo Timer/Contatore in modo da poter ad esempio contare impulsi ricevuti ad un determinato pin o i moduli I^2C, SPI e UART per implementare il relativo protocollo di comunicazione.

2.3.2 Configurazione ingressi e uscite per funzionalità specifiche dei moduli

Attraverso i registri PADnFNCSEL è possibile assegnare ai vari moduli specifici interni al microcontrollore Apollo determinate connessioni di ingresso e/o uscita. È quindi possibile configurare diversi tipi di comunicazione (SPI, I^2C, UART), indirizzare su pin esterni segnali di clock e forme d'onda generate dai contatori.
In *figura 2.12* sono riportate le possibili configurazioni relative al modulo UART necessarie per realizzare funzionalità implementate in questo progetto.

Pad	PADnFNCSEL	Name	Pad Type
1	0	SLSDA	Bidirectional Open Drain
2	0	SLWIR3	Bidirectional Tri-state
5	0	M0SCL	Open Drain
6	0	M0SDA	Bidirectional Open Drain
7	0	M0WIR3	Bidirectional Tri-state
8	0	M1SCL	Open Drain
9	0	M1SDA	Bidirectional Open Drain
10	0	M1WIR3	Bidirectional Tri-state
21	0	SWDIO	Bidirectional Tri-state

Figura 2.12: Configurazione ricezione UART *(Apollo MCU Datasheet Rev. 0.9)*

I segnali UART, TX e RX possono essere connessi, indipendentemente tra loro, a diversi pin. Per configurare la ricezione è necessario porre a uno il corrispondente bit PADnINPEN e a zero il relativo bit PADnPULL. In *figura 2.13* è riportato il mapping di memoria dei registri GPIO.

Address(s)	Register Name	Description
0x40010000	PADREGA	Pad Configuration Register A
0x40010004	PADREGB	Pad Configuration Register B
0x40010008	PADREGC	Pad Configuration Register C
0x4001000C	PADREGD	Pad Configuration Register D
0x40010010	PADREGE	Pad Configuration Register E
0x40010014	PADREGF	Pad Configuration Register F
0x40010018	PADREGG	Pad Configuration Register G
0x4001001C	PADREGH	Pad Configuration Register H
0x40010020	PADREGI	Pad Configuration Register I
0x40010024	PADREGJ	Pad Configuration Register J
0x40010028	PADREGK	Pad Configuration Register K
0x4001002C	PADREGL	Pad Configuration Register L
0x40010030	PADREGM	Pad Configuration Register M
0x40010040	CFGA	GPIO Configuration Register A
0x40010044	CFGB	GPIO Configuration Register B
0x40010048	CFGC	GPIO Configuration Register C
0x4001004C	CFGD	GPIO Configuration Register D
0x40010050	CFGE	GPIO Configuration Register E
0x40010054	CFGF	GPIO Configuration Register F
0x40010058	CFGG	GPIO Configuration Register G
0x40010060	PADKEY	Key Register for all pad configuration registers
0x40010080	RDA	GPIO Input Register A
0x40010084	RDB	GPIO Input Register B
0x40010088	WTA	GPIO Output Register A
0x4001008C	WTB	GPIO Output Register B
0x40010090	WTSA	GPIO Output Register A Set
0x40010094	WTSB	GPIO Output Register B Set
0x40010098	WTCA	GPIO Output Register A Clear
0x4001009C	WTCB	GPIO Output Register B Clear
0x400100A0	ENA	GPIO Enable Register A
0x400100A4	ENB	GPIO Enable Register B
0x400100A8	ENSA	GPIO Enable Register A Set
0x400100AC	ENSB	GPIO Enable Register B Set
0x400100B4	ENCA	GPIO Enable Register A Clear
0x400100B8	ENCB	GPIO Enable Register B Clear
0x40010200	INT0EN	GPIO Interrupt Registers 31-0: Enable
0x40010204	INT0STAT	GPIO Interrupt Registers 31-0: Status
0x40010208	INT0CLR	GPIO Interrupt Registers 31-0: Clear
0x4001020C	INT0SET	GPIO Interrupt Registers 31-0: Set
0x40010210	INT1EN	GPIO Interrupt Registers 49-32: Enable
0x40010214	INT1STAT	GPIO Interrupt Registers 49-32: Status
0x40010218	INT1CLR	GPIO Interrupt Registers 49-32: Clear
0x4001021C	INT1SET	GPIO Interrupt Registers 49-32: Set

Figura 2.13: Mapping registri GPIO *(Apollo MCU Datasheet Rev. 0.9)*

2.4 Modulo di comunicazione UART

Il modulo di comunicazione UART è in grado di operare indipendentemente, consentendo al microcontrollore di passare nelle modalità low-power durante la comunicazione.
Sono supportate comunicazioni di tipo full-duplex e half-duplex ed è presente una funzionalità di loopback per diagnostica e collaudo.
È possibile configurare la trama per quanto riguarda la dimensione del campo dati, il numero di bit di stop ed il tipo di parità.
Sono presenti due memorie FIFO separate in modo da ridurre il carico computazionale del microcontrollore, una organizzata in 32 word da 8 bit dedicata alla trasmissione ed una organizzata in 32 word da 12 bit dedicata alla ricezione (i 4 bit extra in ricezione sono utilizzati per gestire eventuali errori che il microcontrollore necessita di analizzare), ed un generatore di baud-rate programmabile in grado di fornire una frequenza massima di 921600 bit al secondo.

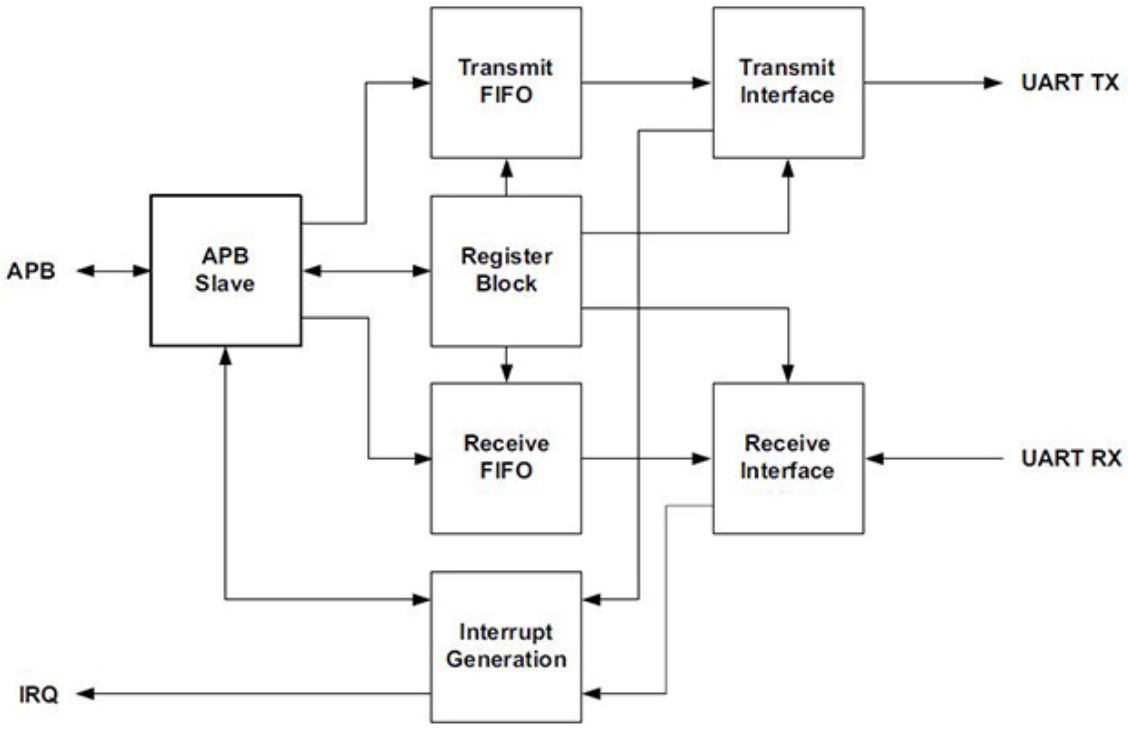

Figura 2.14: Schema a blocchi del modulo UART *(Apollo MCU Datasheet Rev. 0.9)*

Il modulo, il cui schema a blocchi è riportato in *figura 2.16* , si occupa di convertire i dati paralleli scritti attraverso la porta APB Slave in dati seriali successivamente trasmessi ad un dispositivo esterno e/o convertire i dati ricevuti serialmente da un dispositivo esterno in dati paralleli memorizzati su un buffer in attesa di essere letti dalla CPU.

Un generatore di interrupt può opzionalmente inviare interrupt al processore per trasmettere, ricevere o comunicare eventi legati a errori di comunicazione e funzionamento.
Il clock utilizzato dalla logica seriale è prodotto dal generatore dedicato UARTCLK, contenuto all'interno del modulo di generazione dei segnali di clock. La frequenza di questo clock è determinata dalla massima baud-rate desiderata e il valore massimo è di 24MHz.
Per la configurazione si opera sui registri dedicati, mappati in memoria agli indirizzi riportati in *figura 2.15* . Attraverso il registro LCRH sono impostati il numero di bit del dato trasmesso in ogni trama, il numero di stop bit e il tipo di parità. Attraverso i registri IBRD e FBRD è configurata la baud-rate attraverso la seguente formula:

$$\frac{F_{UART}}{16 \cdot BR} = IBRD + \frac{FBRD}{64}$$

Dove BR rappresenta la baud-rate desiderata, F_{UART} è la frequenza di funzionamento del modulo, IBRD la parte intera del divisore ottenuto e FBRD la parte frazionale dello stesso moltiplicata per un fattore 64.
Il modulo supporta controllo del flusso dati di tipo Clear-to-Send e Request-to-Send in maniera indipendente ed è possibile configurare queste funzionalità attraverso il registro CR.
Prima di poter comunicare è necessario selezionare una frequenza di funzionamento per il modulo e abilitare il clock. La frequenza di funzionamento F_R è configurabile all'interno del registro CR mentre per l'abilitazione o disabilitazione del clock è necessario operare sul registro UARTEN del modulo del generatore di clock.
Per garantire il minimo dispendio di energia, il clock fornito al modulo UART deve essere abilitato per il minor tempo possibile e disabilitato non appena la comunicazione giunge a termine.

Address(s)	Register Name	Description
0x4001C000	DR	UART Data Register
0x4001C004	RSR	UART Status Register
0x4001C018	FR	Flag Register
0x4001C020	ILPR	IrDA Counter
0x4001C024	IBRD	Integer Baud Rate Divisor
0x4001C028	FBRD	Fractional Baud Rate Divisor
0x4001C02C	LCRH	Line Control High
0x4001C030	CR	Control Register
0x4001C034	IFLS	FIFO Interrupt Level Select
0x4001C038	IER	Interrupt Enable
0x4001C03C	IES	Interrupt Status
0x4001C040	MIS	Masked Interrupt Status
0x4001C044	IEC	Interrupt Clear

Figura 2.15: Mapping registri UART *(Apollo MCU Datasheet Rev. 0.9)*

2.5 Modulo timer/contatori

Il modulo relativo ai timer e contatori include quattro coppie timer/contatore, la cui struttura è illustrata in *figura 2.16* .

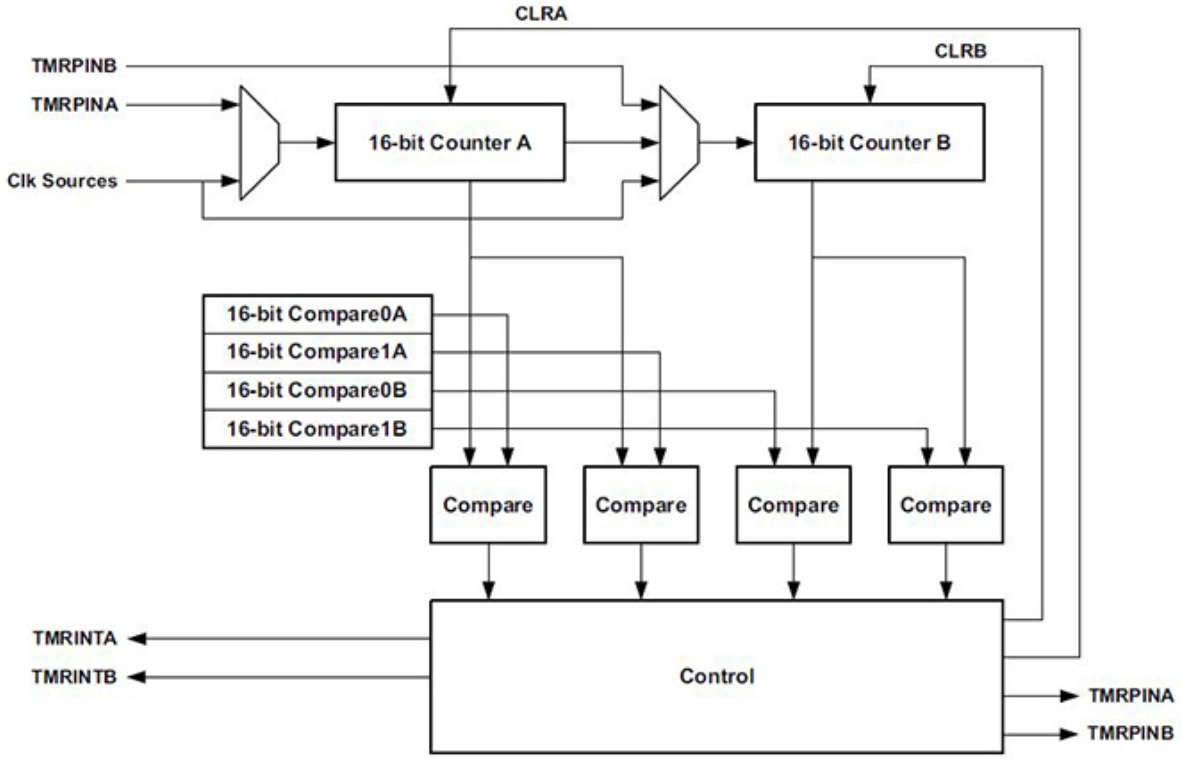

Figura 2.16: Schema a blocchi singola coppia timer/contatore *(Apollo MCU Datasheet Rev. 0.9)*

All'interno di ogni coppia sono presenti due contatori very-low power asincroni a 16 bit che possono essere combinati per fornire un contatore a 32 bit. Quattro registri contengono i valori di reset per i contatori e/o i valori di riferimento dei comparatori che permettono la generazione di segnali complessi in uscita. Ogni contatore ha una connessione esterna per un pin e può fornire diversi tipi di funzionalità.
È possibile attivare interrupt dopo uno specifico ritardo oppure periodicamente ad una determinata frequenza, ricavare i tempi che intercorrono tra determinati eventi, generare impulsi in uscita di durata prefissata con temporizzazione configurabile, oppure segnali PWM a determinate frequenze e duty-cycle, contare fronti in salita e/o discesa ricevuti su input esterni e generare interrupt dopo un determinato numero di questi.
Ogni contatore può operare in cinque diverse modalità, in relazione alla configurazione dei bit TMRyxFN del registro CTRLx (con x compreso tra 0 e 3 e y=A oppure B).

Ogni modalità determina una diversa generazione degli interrupt e un diverso tipo di controllo per il pin associato in uscita. Di seguito la descrizione dettagliata.

2.5.1 Modalità Single Count (FN=0)

In questa modalità il timer viene incrementato ad ogni impulso di clock, contando da 0 al valore assegnato al registro CMPR0 e poi si interrompe, generando opzionalmente un interupt. Il pin in uscita è al valore iniziale selezionato tramite il bit POL e quando viene raggiunto il valore di riferimento inverte il livello logico (questa funzionalità è abilitata attraverso il bit TMRyxPE del registro CTRLx) generando un interrupt se il bit PE è stato precedentemente settato. A questo punto il timer viene resettato e il valore in uscita è mantenuto fino al reset del bit CLR. L'interrupt eventualmente generato può essere cancellato ponendo a uno il corrispondente bit WC all'interno del registro TMRWCR.

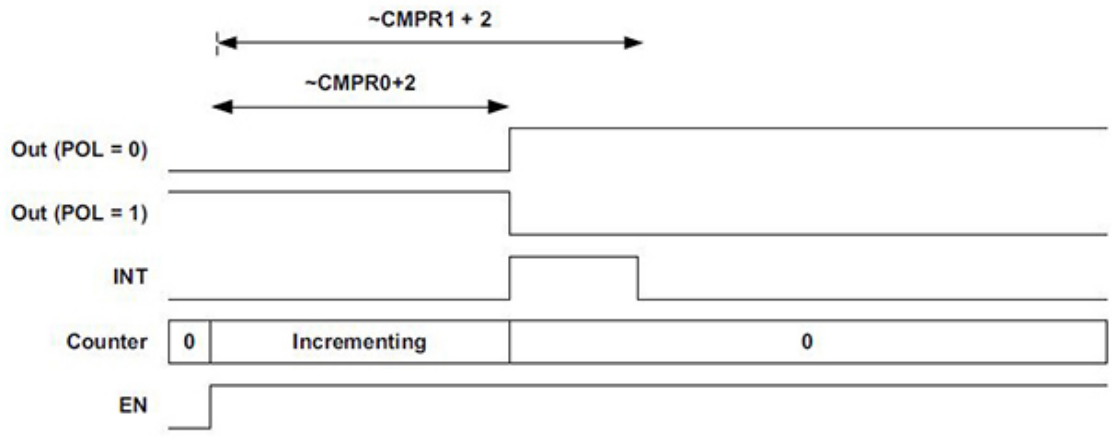

Figura 2.17: Modalità Single Count, FN=0 *(Apollo MCU Datasheet Rev. 0.9)*

2.5.2 Modalità Repeated Count (FN=1)

In modalità Repeated Count il timer conta gli impulsi di clock e, al raggiungimento del valore assegnato al registro CMPR0, inverte il valore assegnato all'uscita precedentemente impostato tramite il bit POL (se l'inversione del valore è stata abilitata attraverso il bit TMRyxPE del registro CTRLx) generando un interrupt se il bit PE è stato precedentemente settato.
A questo punto il timer resetta il conteggio, mantenendo il valore in uscita per un ciclo di clock, per poi riportarlo al valore iniziale. Il conteggio viene poi riavviato ed il processo ripetuto. In questa maniera è possibile creare un flusso di impulsi, di durata pari a ad un ciclo di clock, e/o interrupt ad intervalli fissati. L'interrupt

eventualmente generato può essere cancellato ponendo a uno il corrispondente bit WC all'interno del registro TMRWCR in ogni istante antecedente l'impulso seguente.

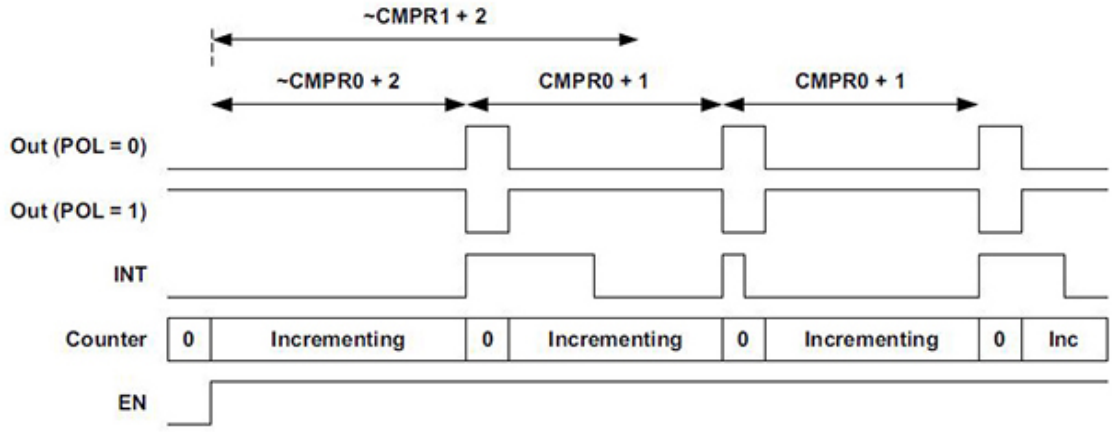

Figura 2.18: Modalità Repeated Count, FN=1*(Apollo MCU Datasheet Rev. 0.9)*

2.5.3 Modalità Single Pulse (FN=2)

Tramite questa modalità è possibile generare impulsi controllandone la durata.
Il timer, una volta abilitato, incrementa il conteggio ad ogni ciclo di clock fino a raggiungere il valore contenuto nel registro CMPR0, generando opzionalmente un interrupt se il bit PE è stato precedentemente settato.
Il pin in uscita è inizialmente al valore selezionato tramite il bit POL che viene invertito (se l'inversione del valore è stata abilitata attraverso il bit TMRyxPE del registro CTRLx) al raggiungimento del valore di CMPR0. A questo punto il timer continua il conteggio e il valore in uscita è mantenuto fino al raggiungimento del valore contenuto nel registro CMPR1 ed in questo istante il valore in uscita viene riportato al valore iniziale. L'interrupt eventualmente generato può essere cancellato ponendo a uno il corrispondente bit WC all'interno del registro TMRWCR.

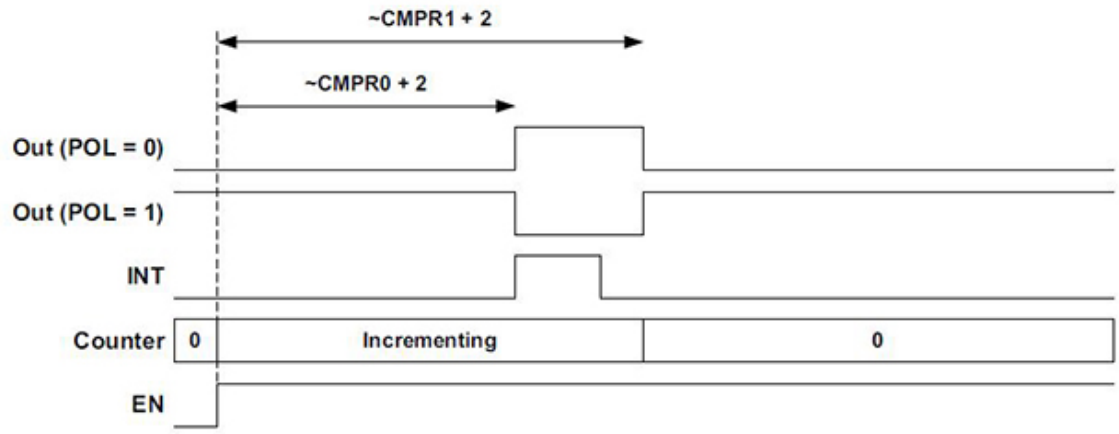

Figura 2.19: Modalità Single Pulse, FN=2 *(Apollo MCU Datasheet Rev. 0.9)*

2.5.4 Modalità Repeated Pulse (FN=3)

Tramite la modalità Repeated Pulse è possibile generare sequenze di impulsi di durata e periodo regolabili.
Il timer, una volta abilitato, incrementa il conteggio ad ogni ciclo di clock fino a raggiungere il valore contenuto nel registro CMPR0, generando opzionalmente un interrupt se il bit PE è stato precedentemente settato.
Il pin in uscita è inizialmente al valore selezionato tramite il bit POL che viene invertito (se l'inversione del valore è stata abilitata attraverso il bit TMRyxPE del registro CTRLx) al raggiungimento del valore di CMPR0, generando un interrupt se il bit PE è stato precedentemente settato. A questo punto il timer continua il conteggio e il valore in uscita è mantenuto fino al raggiungimento del valore contenuto nel registro CMPR1 ed in questo istante il valore in uscita viene riportato al valore iniziale. Il conteggio viene poi riavviato e il processo è ripetuto. L'interrupt eventualmente generato può essere cancellato ponendo a uno il corrispondente bit WC all'interno del registro TMRWCR.

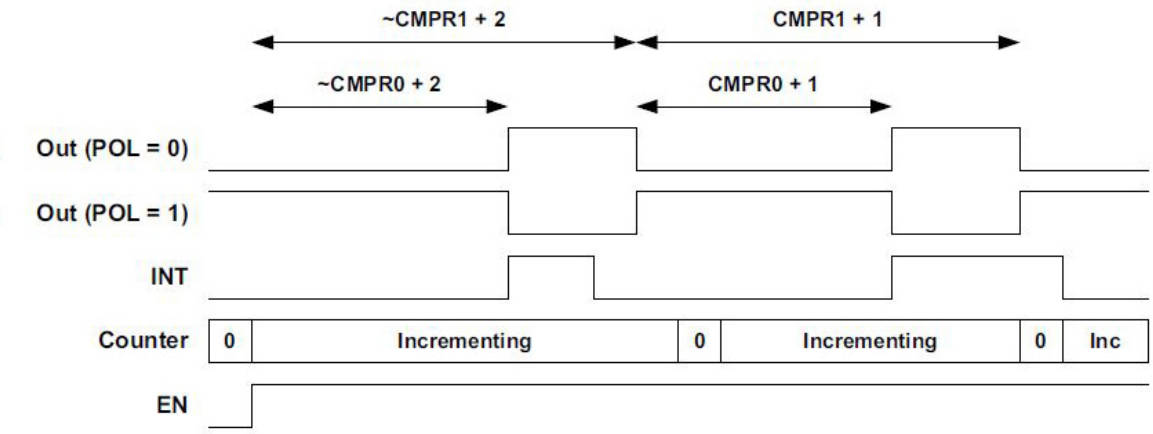

Figura 2.20: Modalità Repeated Pulse, FN=3 *(Apollo MCU Datasheet Rev. 0.9)*

2.5.5 Modalità Continuous (FN=4)

Questa modalità è principalmente utilizzata per due funzioni. La prima è il conteggio di impulsi ricevuti su un determinato pin di ingresso, generando un interrupt ad uno specifico numero di impulsi. La seconda è la funzionalità di timer per la misura di periodi di tempo.
Incrementando il conteggio ad ogni impulso di clock, invertendo il valore di uscita (se l'inversione del valore è stata abilitata attraverso il bit TMRyxPE del registro CTRLx) e generando un interrupt, se il bit PE è stato precedentemente settato, al raggiungimento del valore assegnato al registro CMPR0. Il timer, in questa modalità, prosegue il conteggio e non è mai resettato automaticamente. Qualora il conteggio arrivasse a saturazione e ricominciasse da zero, al nuovo raggiungimento del valore di riferimento non è né generato un interrupt, né invertito il valore in uscita.

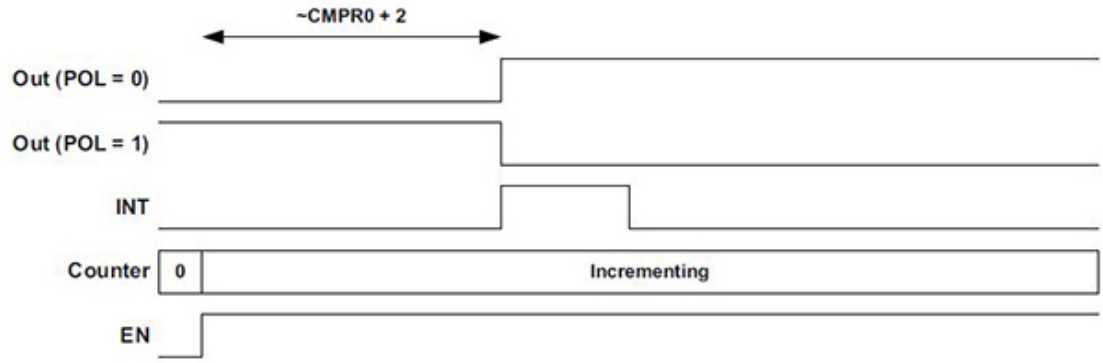

Figura 2.21: Modalità Continuous, FN=4 *(Apollo MCU Datasheet Rev. 0.9)*

Address(s)	Register Name	Description
0x40008000	TMR0	Counter/Timer Register
0x40008004	CMPRA0	Counter/Timer A0 Compare Registers
0x40008008	CMPRB0	Counter/Timer B0 Compare Registers
0x4000800C	CTRL0	Counter/Timer Control
0x40008010	TMR1	Counter/Timer Register
0x40008014	CMPRA1	Counter/Timer A1 Compare Registers
0x40008018	CMPRB1	Counter/Timer B1 Compare Registers
0x4000801C	CTRL1	Counter/Timer Control
0x40008020	TMR2	Counter/Timer Register
0x40008024	CMPRA2	Counter/Timer A2 Compare Registers
0x40008028	CMPRB2	Counter/Timer B2 Compare Registers
0x4000802C	CTRL2	Counter/Timer Control
0x40008030	TMR3	Counter/Timer Register
0x40008034	CMPRA3	Counter/Timer A3 Compare Registers
0x40008038	CMPRB3	Counter/Timer B3 Compare Registers
0x4000803C	CTRL3	Counter/Timer Control
0x40008200	INTEN	Counter/Timer Interrupts: Enable
0x40008204	INTSTAT	Counter/Timer Interrupts: Status
0x40008208	INTCLR	Counter/Timer Interrupts: Clear
0x4000820C	INTSET	Counter/Timer Interrupts: Set

Figura 2.22: Mapping registri modulo timer/contatori *(Apollo MCU Datasheet Rev. 0.9)*

Capitolo 3

Il Firmware

Il firmware realizzato, come abbiamo già discusso in precedenza, si occupa di gestire la comunicazione tra tag e lettore attraverso modulatore backscattering.
Il microcontrollore riceve in ingresso tramite linea seriale il segnale UART_RX e comanda i tre segnali in uscita START_LTC, START_CLK e START_SW con cui pilota gli switch SW1, SW2 ed SW3.

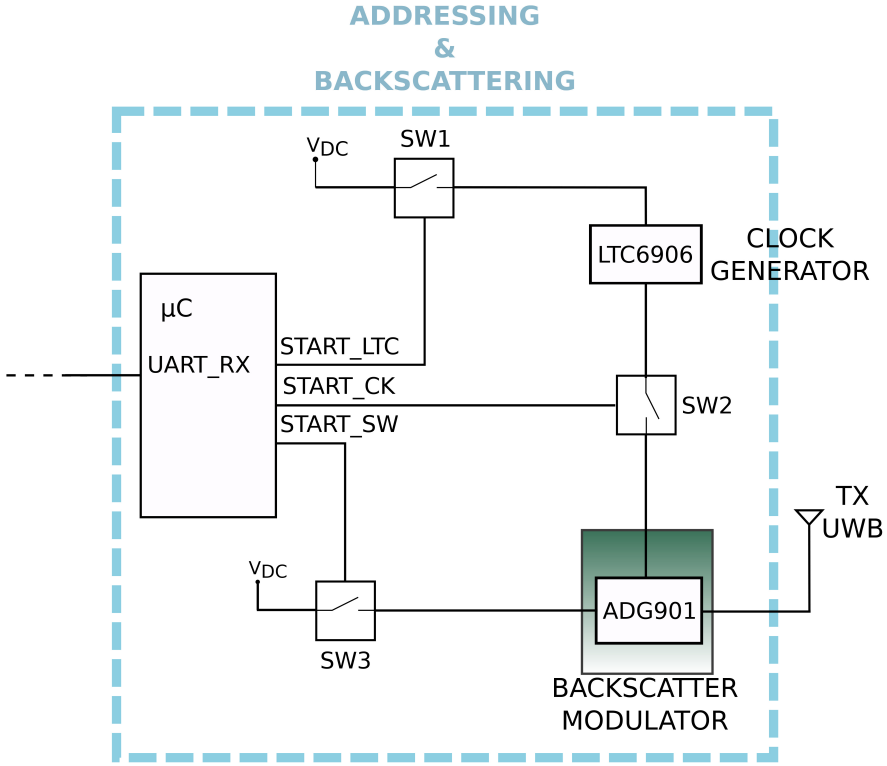

Figura 3.1: Sezione addressing e backscattering

L'obiettivo del software è svolgere le operazioni richieste abbattendo il più possibile i consumi, disponendo per l'alimentazione esclusivamente della carica immagazzinata

sulla capacità di storage durante la fase di ricezione del segnale. É quindi importante adottare una strategia volta a minimizzare il numero di operazioni ed il tempo di esecuzione per ogni task. Data questa premessa, nella scrittura del codice si è deciso di adottare una gestione dei servizi richiesti tramite interrupt.
Inoltre, il microcontrollore Apollo offre la possibilità di operare nelle modalità low-power nelle condizioni descritte nel capitolo 2. Si è quindi cercato di progettare una soluzione che consenta di posizionarsi nelle modalità a consumo più ridotto per il maggior tempo di esecuzione del programma.
Per la programmazione si è fatto uso dell'IDE Keil μVision V5.23.0.0 di ARM.
Il codice realizzato è strutturato nella seguente maniera: sono presenti una procedura di main e tre routine di servizio degli interrupt, ISR_GPIO, ISR_UART e ISR_TIMER, utilizzate per gestire la comunicazione.

3.1 Funzionamento del sistema

La logica di funzionamento del firmware progettato è riportata in *figura 3.2* .

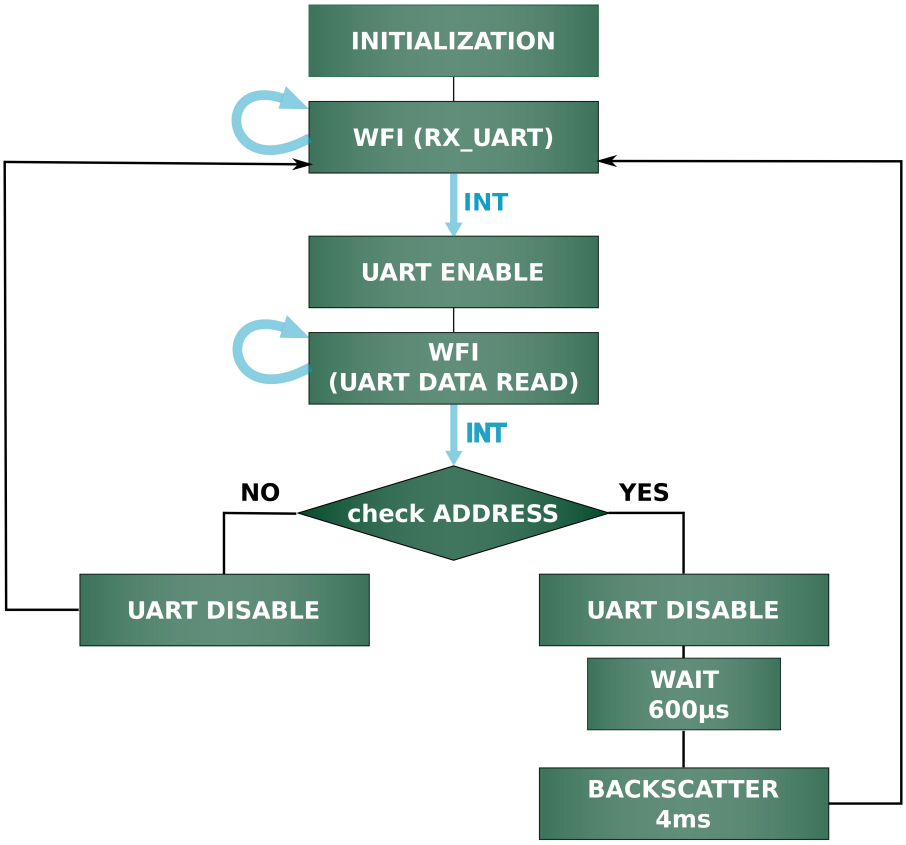

Figura 3.2: Schema a blocchi del funzionamento

Dopo una prima parte di inizializzazione di ingressi, uscite e periferiche, il programma si pone in uno stato di attesa (Wait for Interrupt - WFI) e vi rimane fintanto che un interrupt, generato da un fronte in discesa rilevato sulla porta dedicata alla ricezione tramite UART, segnala la presenza di un segnale in arrivo.
Alla ricezione del suddetto interrupt, la periferica UART viene attivata ed il programma si posiziona in un secondo stato di attesa, in cui rimane fino all'attivazione dell'interrupt segnalante la fine del processo di ricezione.
A questo punto ha inizio la fase di addressing in cui il microcontrollore confronta il dato ricevuto tramite la periferica seriale con il proprio identificativo.
In caso di mancata coincidenza la periferica UART viene disabilitata ed il programma si riporta nel primo stato di attesa. Se la condizione di uguaglianza tra dato ricevuto e identificativo è invece soddisfatta, il microcontrollore provvede a disabilitare la periferica, abilitare l'alimentazione del generatore di clock esterno LTC6906, attendere per un periodo di 600μs necessario a quest'ultimo per produrre un segnale di clock stabile, ed infine attivare la fase di backscattering vera e propria, di durata pari a 4ms, durante la quale il segnale è trasmesso dal modulatore ADG901 attraverso l'antenna UWB. Anche in questo secondo caso, alla fine delle operazioni necessarie per la trasmissione, il programma si riporta nel primo stato di attesa, in modo da poter ricevere una nuova trasmissione.

3.2 Procedura di Main

Nel main, il cui diagramma di flusso è riportato in *figura 3.3*, sono configurati i registri I/O e i registri delle periferiche.

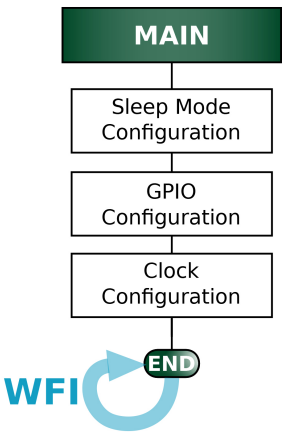

Figura 3.3: Diagramma di flusso procedura di Main

Innanzitutto, subito dopo aver disattivato il watchdog, è stato configurato il modulo di Power Management, attraverso le istruzioni riportate in *figura 3.4*.

Con queste istruzioni sono impostati i bit SEVONPEND, SLEEPDEEP e SLEEPONEXIT del System Control Register (SCR). Il bit SEVONPEND, una volta attivato, fa sì che il presentarsi di una richiesta di interrupt costituisca un evento di wake-up (Wake Up Event - WUE), in seguito al quale il microcontrollore, trovandosi eventualmente in uno stato low-power, avvierà la procedura di wake-up per passare in modalità Active. Il bit SLEEPDEEP, se settato, abilita la modalità Deep-Sleep, nella quale, oltre al generatore di clock ad alta frequenza, saranno disattivati anche la CPU e i Buck Converter del generatore di tensione.
Il bit SLEEPONEXIT è settato per far sì che all'uscita delle routine di servizio degli interrupt il microcontrollore rientri in modalità low-power.

```
62 // STOP WatchDog
63   WDT_REGA(AM_REG_WDT_CFG_O) &= ~AM_REG_WDT_CFG_WDTEN(1);// Disable Watchdog Timer
64
65 //-----------------------------------------------
66
67 //Low Power configuration:
68 SYSCTRL_REGA(AM_REG_SYSCTRL_SCR_O) |= AM_REG_SYSCTRL_SCR_SEVONPEND(1)\
69                     | AM_REG_SYSCTRL_SCR_SLEEPONEXIT(1);
70                     //pending event=WUE ;
71                     //return from ISR = go to sleep;
72 am_hal_sysctrl_sleep(AM_HAL_SYSCTRL_SLEEP_DEEP);
73                     //the sleep mode is deep sleep
74
75 //-----------------------------------------------
76
```

Figura 3.4: Firmware: Configurazione Sleep Mode

Il codice relativo alla configurazione del clock di sistema è riportato in *figura 3.5*. La configurazione avviene tramite la scrittura del registro CCTRL. Per poter scrivere in questo registro è precedentemente necessario inserire nel registro CLKKEY il valore KEYVAL pari a 0x47. É stata impostata la massima frequenza di funzionamento per il processore, pari a 24MHz, dal momento che il modulo UART permette di scegliere come clock di funzionamento solamente sottomultipli del clock generato dall'oscillatore ad alta frequenza HFRC. Dato questo vincolo, si è scelto di utilizzare il massimo valore di frequenza per abbattere i tempi di esecuzione. Sempre in questa fase, è stato disattivato l'oscillatore a cristallo interno XT tramite scrittura del registro OCTRL, essendo questo tipo di sorgente generalmente molto dispendiosa dal punto di vista energetico.

```
77  // Clock Registers Configuration:
78
79    CLKGEN_REGA(AM_REG_CLKGEN_CLKKEY_O) = AM_REG_CLKGEN_CLKKEY_KEYVAL; //Insert KEYVAL
80    CLKGEN_REGA(AM_REG_CLKGEN_OCTRL_O) |=  AM_REG_CLKGEN_OCTRL_STOPXT_STOP;
81                                          //Stop the XT Oscillator from driving the RTC
82    am_hal_clkgen_sysclk_select(AM_HAL_CLKGEN_SYSCLK_MAX);
83                                          //System Clock = 24MHz
84
85
```

Figura 3.5: Firmware: Configurazione dei registri del generatore di clock

Per la configurazione delle porte GPIO è stato necessario configurare le funzioni assegnate ai pin del microcontrollore. É possibile visualizzare in *figura 3.6* la configurazione realizzata. Al pin 15 è stata assegnata la funzionalità UARTRX e ne è stato abilitato l'input. I pin 8, 10 e 12 sono le uscite del sistema e sono state configurate come uscite GPIO, in modo da fornire un valore logico tale da poter pilotare rispettivamente gli switch SW2, SW1 ed SW3. La scrittura di questi registri è stata abilitata scrivendo sul registro PADKEY il relativo valore di abilitazione KEYVAL, pari a 0x73. Per l'abilitazione delle porte GPIO è stata poi impostata all'interno del registro ENA la maschera di bit corrispondente ai GPIO utilizzati.

```
86  //Pad function Configuration:
87
88    GPIO_REGA(AM_REG_GPIO_PADKEY_O)  =  AM_REG_GPIO_PADKEY_KEYVAL; //Insert KEYVAL
89
90    GPIO_REGA(AM_REG_GPIO_PADREGD_O) |= AM_REG_GPIO_PADREGD_PAD15FNCSEL_GPIO15 \
91                                     | AM_REG_GPIO_PADREGD_PAD15INPEN_M;
92
93    GPIO_REGA(AM_REG_GPIO_PADREGC_O) |= AM_REG_GPIO_PADREGC_PAD8FNCSEL_GPIO8 \
94                                     |  AM_REG_GPIO_PADREGC_PAD8STRNG_HIGH; // Pad 8 = START_CLK
95    GPIO_REGA(AM_REG_GPIO_PADREGC_O) |= AM_REG_GPIO_PADREGD_PAD12FNCSEL_GPIO12 \
96                                     |  AM_REG_GPIO_PADREGD_PAD12STRNG_HIGH; // Pad 12 = START_SW
97    GPIO_REGA(AM_REG_GPIO_PADREGC_O) |= AM_REG_GPIO_PADREGC_PAD10FNCSEL_GPIO10\
98                                     |  AM_REG_GPIO_PADREGC_PAD10STRNG_HIGH; // Pad 10 = START_LTC
99
100
101
102   AM_REG(GPIO, ENA) |= 0x9500; //ENABLE GPIO (15,8,10,12)
```

Figura 3.6: Firmware: Configurazione dei registri dei pin

In *figura 3.7* è riportato il codice attraverso il quale sono state configurati i tipi di input e output per le GPIO precedentemente configurate. Attraverso il registro CFGB è abilitata la lettura del GPIO15 (UARTRX) e definita la direzione del fronte sul quale è possibile generare un interrupt, scegliendo il fronte in discesa.
Sempre sul registro CFGB sono state definite anche le configurazioni delle porte di uscita, abilitandone l'output. Sono poi stati assegnati i valori di default delle porte, assegnando a tutte le porte di default un valore logico basso. Infine è stato abilitato l'interrupt sulla GPIO15.

```
111 //GPIO Configuration:
112   GPIO_REGA(AM_REG_GPIO_CFGB_O)|= AM_REG_GPIO_CFGB_GPIO15INCFG_READ\
113                                |AM_REG_GPIO_CFGB_GPIO15INTD_INTHL;
114                                //(Pad 15 input enable
115                                //interrupt on High-to-Low transistion
116                                //for UART Module enable
117   GPIO_REGA(AM_REG_GPIO_CFGB_O)|= AM_REG_GPIO_CFGB_GPIO8OUTCFG_PUSHPULL\
118                                |AM_REG_GPIO_CFGB_GPIO12OUTCFG_PUSHPULL \
119                                |AM_REG_GPIO_CFGB_GPIO10OUTCFG_PUSHPULL;
120                                //(PUSHPULL Pad 8; Pad 12; Pad 10)
121
122   GPIO_REGA(AM_REG_GPIO_WTA_O )&= AM_REG_GPIO_WTA_WTA(0x00000000); //Default value "0"
123   GPIO_REGA(AM_REG_GPIO_WTB_O )&= AM_REG_GPIO_WTB_WTB(0x0000); //Default value "0"
124
125   //Interrupt GPIO1 configuration--------------
126   GPIO_REGA(AM_REG_GPIO_INT0CLR_O)|= AM_REG_GPIO_INT0CLR_GPIO15(1);//Clear GPIO15 interrupt
127   GPIO_REGA(AM_REG_GPIO_INT0EN_O) |= AM_REG_GPIO_INT0EN_GPIO15(1); //Enable GPIO15 Interrupt
128
129   AM_REG(NVIC, ISER0) = 0x1 << ((25 - 16) & 0x1F); //Enable GPIO Interrupts for the NVIC
130
131   AM_REGn(GPIO, 0, PADKEY) = 0; //Lock GPIO register
```

Figura 3.7: Firmware: Configurazione dei registri GPIO

La funzione di questo interrupt è rilevare l'impulso negativo relativo al bit di start dell'UART. Questa opzione si è resa necessaria dal momento che la periferica UART necessita dell'abilitazione del clock ad alta frequenza HFRC per il funzionamento. L'attivazione di un modulo che utilizza tale clock rende impossibile entrare in modalità low-power.
Nell'ottica di minimizzare i consumi, si è quindi reso necessario individuare uno stratagemma attraverso il quale fosse possibile disabilitare questa periferica senza perdere le trame ricevute.
In questo modo è possibile disattivare il modulo UART, ponendosi in modalità Deep-Sleep, e riattivarlo solo in seguito alla ricezione dell'interrupt generato dal primo fronte in discesa sulla GPIO. A questo punto l'imminente arrivo della trama sarà gestito dal modulo appena riattivato.

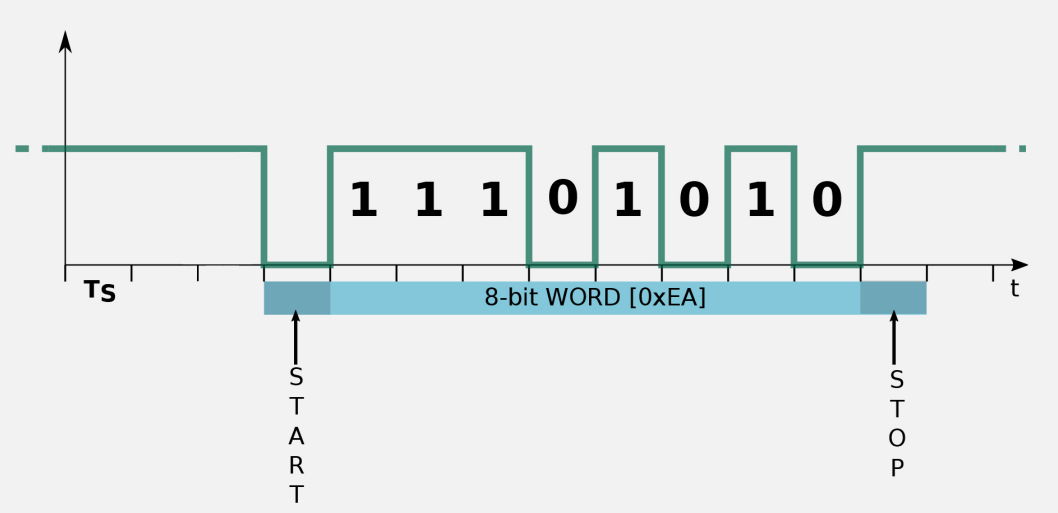

Figura 3.8: Struttura della trama UART utilizzata

Sono stati istanziati due timer annidati A0 e B0 per gestire i ritardi necessari per le attivazioni degli switch. Il timer B0 si occupa di introdurre un ritardo di 600μs tra l'attivazione del primo switch digitale SW1 e l'attivazione dei seguenti SW2 e SW3, attivati in contemporanea una volta che il segnale di clock fornito dal generatore esterno sarà stabilizzato. Il timer B0 genera il ritardo di 4ms all'interno del quale SW1, SW2 ed SW3 devono essere mantenuti accesi in modo da portare a termine la trasmissione. In *figura 3.9* è possibile notare l'impostazione dei limiti di conteggio e l'abilitazione degli interrupt dei timer. Le modalità di conteggio e le sorgenti di clock saranno poi impostate direttamente all'interno delle routine di servizio dei relativi interrupt.

```
135 //Timers Configuration
136
137 CTIMER_REGA(AM_REG_CTIMER_CMPRA0_0) = 3;  //count 4ms
138 CTIMER_REGA(AM_REG_CTIMER_CMPRB0_0) = 1800; //count 600us : 3*600=1800
139
140 CTIMER_REGA(AM_REG_CTIMER_INTEN_0)|=  AM_REG_CTIMER_INTEN_CTMRA0INT(1);
141                                    //Enable Timer A0 Interrupt
142 CTIMER_REGA(AM_REG_CTIMER_INTEN_0)|=  AM_REG_CTIMER_INTEN_CTMRB0INT(1);
143                                     //Enable Timer B0 Interrupt
144
145 AM_REG(NVIC, ISER0) = 0x1 << ((26 - 16) & 0x1F);////Enable CTIMER Interrupts for the NVIC
146
```

Figura 3.9: Firmware: Configurazione dei Timer

La periferica UART è stata in questa prima fase disattivata, così come il clock ad essa dedicato, in modo da poter entrare, subito dopo questa, in modalità low-power Deep-Sleep.
Prima che il modulo possa comunicare è necessario abilitare il clock attraverso il registro UARTEN del modulo di generazione del clock. Per garantire bassi consumi. Tale clock deve essere attivato per il minor tempo possibile e disattivato non appena la trasmissione è giunta a termine. In questo caso l'abilitazione del clock, così come l'attivazione e la configurazione della periferica, è affidata alla routine di gestione dell'interrupt generato dal primo fronte in discesa della trama ricevuto sulla GPIO15.

3.3 Routine di gestione degli interrupt

Come anticipato, sono presenti tre routine di interrupt dedicate alla gestione della ricezione della trama UART e della risposta attraverso modulazione backscattering. In corrispondenza dell'interrogazione da parte del lettore, sulla linea dedicata alla comunicazione seriale, posta in condizione di riposo ad un livello logico alto, è imposta dall'impulso di pre-trasmissione una transizione in discesa. Il primo interrupt generato è quindi quello attivato sull'ingresso GPIO15.

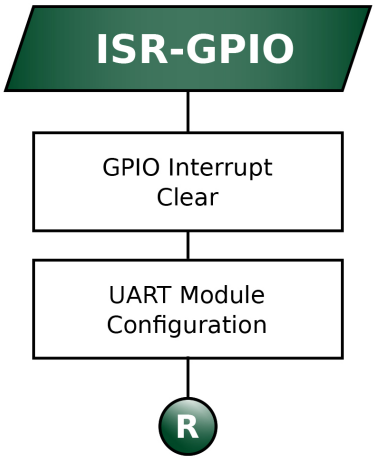

Figura 3.10: Diagramma di flusso della routine di servizio dell'interrupt GPIO

All'interno della routine di servizio di questo interrupt sono inserite, oltre alla disabilitazione e la cancellazione dell'interrupt,l'attivazione della periferica di trasmissione UART, l'abilitazione del clock ad essa dedicato, e la configurazione dei parametri necessari per la ricezione della trama in arrivo.
Il modulo UART è stato configurato impostando una baud-rate di 7kHz, come è possibile vedere in *figura 3.11*.
Sono inoltre impostati un clock di frequenza pari a 24MHz, la lunghezza del campo dati ad un valore di 8 bit e la generazione di un interrupt a fine ricezione. Le configurazioni riguardati parità e numero di stop bit sono state lasciate invariate, in modo da avere un solo bit di stop e assenza di bit per il controllo di parità.
Per la configurazione dettagliata si rimanda al firmware riportato integralmente in allegato.

```
165  //ISR GPIO15 UARTRX for transmission detection
166  void
167 am_gpio_isr(void){
168
169  GPIO_REGA(AM_REG_GPIO_INT0CLR_0)|= AM_REG_GPIO_INT0CLR_GPIO15(1);
170                                                    //Clear interrupt
171  GPIO_REGA(AM_REG_GPIO_INT0EN_0) &= ~AM_REG_GPIO_INT0EN_GPIO15(1);
172                                                  //Disable GPIO Interrupt
173  setup_serial(0, 7000);  //UART Configuration - BR =7kHz
174
175  }
```

Figura 3.11: Firmware: routine di servizio dell'interrupt GPIO

A questo punto il modulo UART è posto in ascolto e alla fine della ricezione della trama è attivato il secondo interrupt, servito dalla routine uart_isr. Il diagramma di flusso e il codice relativi alla routine sono riportati in *figura 3.12*e *figura 3.13*.

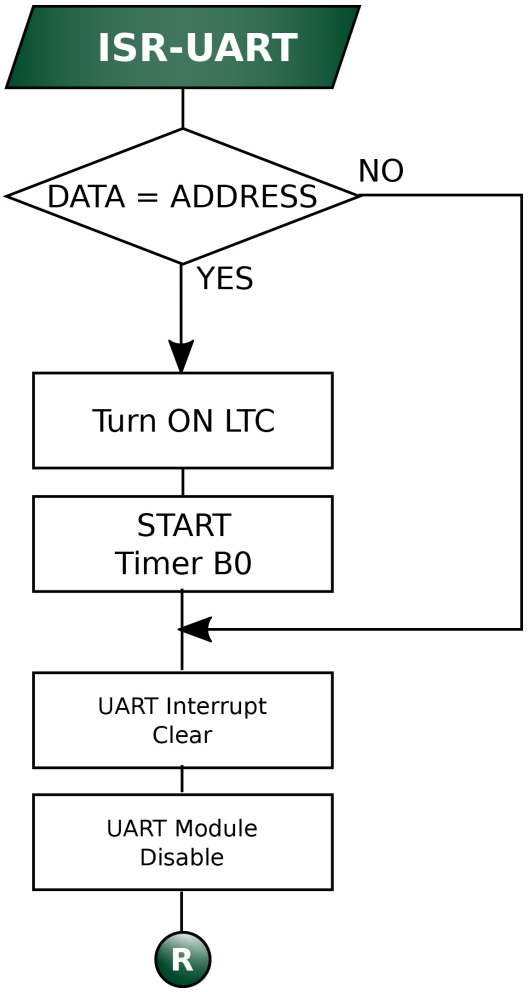

Figura 3.12: Diagramma di flusso della routine di servizio dell'interrupt generato dall'UART

La routine si occupa di resettare l'interrupt, disattivare il modulo UART, il clock ad essa dedicato e confrontare il dato ricevuto con l'identificativo del tag, memorizzato in fase di dichiarazione all'interno della variabile ADDRESS.
Se il dato corrisponde all'identificativo viene posta a livello logico alto l'uscita GPIO10 (START_LTC) e configurato e abilitato il timer B0. L'uscita è pilotata in modo da attivare il generatore di clock esterno LTC6906, mentre il timer è utilizzato per fornire il ritardo necessario al generatore per stabilizzare il segnale prodotto.
La configurazione comprende la scelta della modalità di conteggio (Continuous), l'abilitazione dell'interrupt al raggiungimento del valore limite e l'assegnazione della sorgente di clock. Per quest'ultima è stato assegnato l'oscillatore ad alta frequenza HFRC con divisore di frequenza pari a 8, ottenendo una frequenza di funzionamento pari a 3MHz. Avendo impostato nel main un valore limite per il conteggio pari a 1800, l'interrupt del contatore sarà generato dopo un periodo pari a 600μs.

```
179  //ISR UART for TAG addressing
180  void
181 am_uart_isr(void){
182
183     am_hal_uart_int_clear(0,0x7FF); // reset UART interrupts
184
185     UARTD = UART_DATA;
186     UARTDbe = 0; //Big Endian received DATA
187     for (int i=0;i<8;i++){
188       UARTDbe |= (((UARTD>>(7-i))&0x1) << (i));
189     }
190
191     if (UARTDbe == ADDRESS){
192       CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)&=AM_REG_CTIMER_CTRL0_TMRB0CLR_RUN;
193                                         //Allow Timer B0 to RUN
194       CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)|=  AM_REG_CTIMER_CTRL0_TMRB0IE_EN \
195                       |AM_REG_CTIMER_CTRL0_TMRB0FN_CONTINUOUS\
196                       |AM_REG_CTIMER_CTRL0_TMRB0CLK_HFRC_DIV8 \
197                       |AM_REG_CTIMER_CTRL0_TMRB0EN_EN;;
198                                             //TIMER B0 interrupt enable;
199                                             //Continuous Mode;
200                                             //CONF TIMER B0 clock 3MHz;
201                                             //Start A0
202
203       GPIO_REGA(AM_REG_GPIO_WTA_O )|= AM_REG_GPIO_WTA_WTA(START_LTC);
204                                     //turn on LTC6906 (CLOCK GENERATOR)
205         }
206
207     CLKGEN_REGA(AM_REG_CLKGEN_UARTEN_O) &= ~AM_REG_CLKGEN_UARTEN_UARTEN_EN;
208                                         //disable UART system clock
209
210
211   am_hal_uart_disable(0); //Disable UART
212 }
```

Figura 3.13: Firmware: routine di servizio dell'interrupt generato dall'UART

Il modulo dei timer è in grado di generare un unico interrupt ed è possibile identificare il singolo timer che l'ha generato tramite la consultazione del registro di stato dedicato INTSTAT. Per questo motivo, pur utilizzando due timer, è presente una sola routine di interrupt relativa alle temporizzazioni. All'interno di questa sono quindi presenti due test che permettono di identificare la sorgente dell'interrupt.

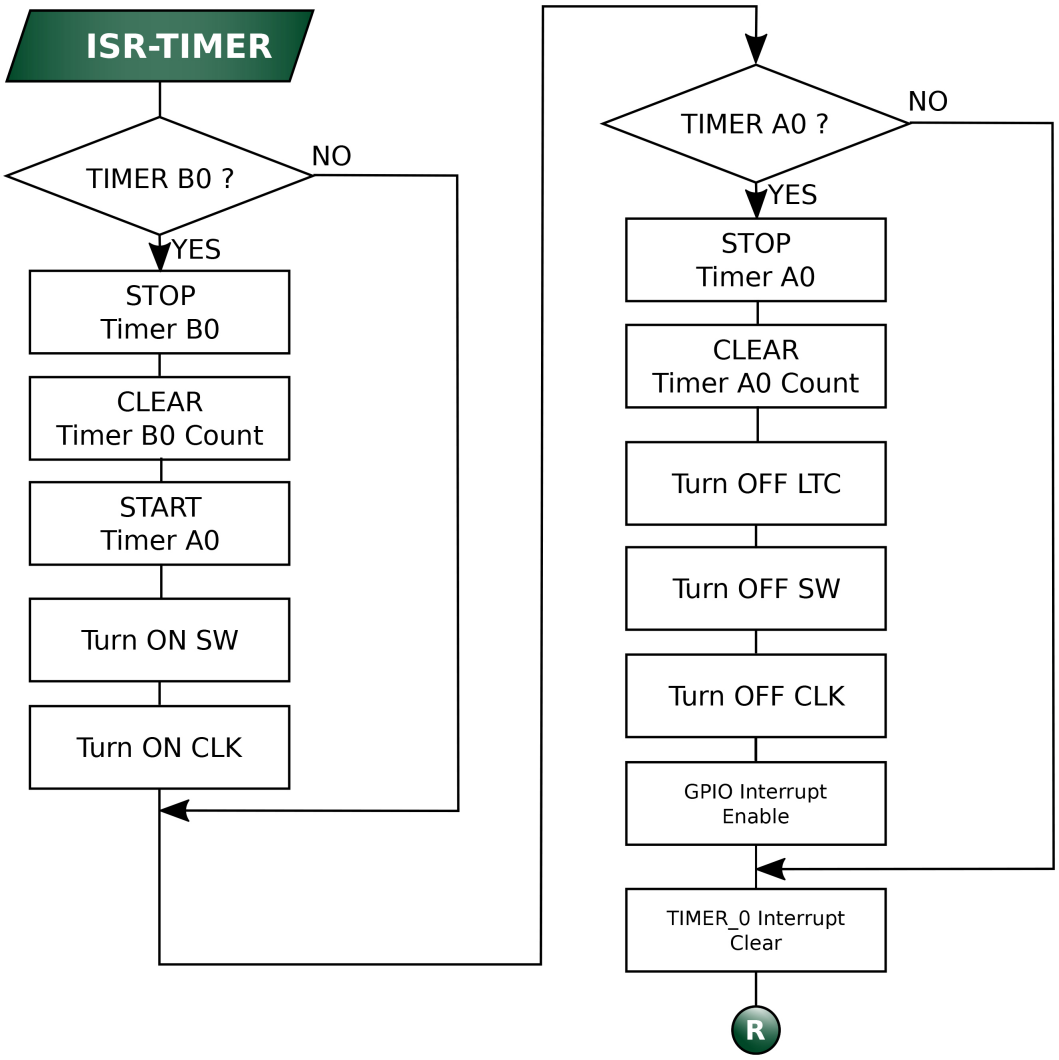

Figura 3.14: Diagramma di flusso della routine di servizio dell'interrupt generato dai timer

Se la condizione del primo test è verificata significa che l'interrupt è stato generato dal timer B0. Sono quindi passati i 600μs e il clock fornito dal dispositivo esterno si è stabilizzato. Sotto questa condizione viene quindi fermato e resettato il conteggio di B0, poste a livello logico alto le uscite GPIO8 e GPIO12 (START_CLK e START_SW, rispettivamente), in modo da chiudere gli switch SW2 ed SW3, ed infine è attivato il conteggio del timer A0, per il quale sono impostati i registri di configurazione. Le impostazioni del timer, analogamente a quelle relative a B0, comprendono l'abili-

tazione dell'interrupt a fine conteggio, la modalità di funzionamento (Continuous) e la sorgente di clock dedicata al timer. Questa volta si è impostato un clock da 1024Hz, ricavato dall'oscillatore a bassa frequenza LFRC, in modo da implementare un ritardo di circa 4ms necessario per la trasmissione tramite la rete di modulazione esterna del segnale di backscattering.
Per la sorgente di clock dedicata a questo timer è stato possibile scegliere l'oscillatore a basso consumo LFRC sia perchè si deve contare un intervallo temporale lungo, sia perchè da specifica non sono posti vincoli sulla precisione.

All'arrivo del secondo interrupt è quindi verificata la condizione del secondo test, essendo trascorso l'intervallo temporale necessario per la trasmissione. A questo punto è quindi possibile porre a livello logico basso tutte le uscite, fermare il conteggio del timer A0, resettarlo e riabilitare l'interrupt sul fronte in discesa sulla GPIO15. Fuori dalle condizioni, come di consueto, è stato cancellato l'interrupt responsabile dell'attivazione della routine.

```
216  // ISR CTIMER
217  void
218 am_ctimer_isr(void){
219    INTSTATUS = am_hal_ctimer_int_status_get(true);
220                      //get ctimer interrupt status
221    INTSTATUSA = INTSTATUS & 0x00000001;
222    INTSTATUSB = INTSTATUS & 0x00000002;
223    if((INTSTATUSB)==0x00000002) {
224      //INT TIMER B0 [ 600uS ] for 600us timing after that the clock generator is ready
225      CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)|= AM_REG_CTIMER_CTRL0_TMRB0CLR_CLEAR \
226                                        |AM_REG_CTIMER_CTRL0_TMRB0EN_DIS;
227                                        // CLEAR Counter ; Stop Counter B0
228      am_hal_ctimer_int_clear(AM_HAL_CTIMER_INT_TIMERB0);
229                                        // CLEAR TIMERB0 Interrupt
230      CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)&=AM_REG_CTIMER_CTRL0_TMRA0CLR_RUN;
231                                        //Allow Timer A0 to RUN
232      CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)|=AM_REG_CTIMER_CTRL0_TMRA0IE_EN\
233                                        |AM_REG_CTIMER_CTRL0_TMRA0FN_CONTINUOUS\
234                                        |AM_REG_CTIMER_CTRL0_TMRA0CLK_LFRC  \
235                                        |AM_REG_CTIMER_CTRL0_TMRA0EN_EN;
236                                                  //TIMER A0 interrupt enable;
237                                                  //Continuous Mode;
238                                                  //CONF TIMER A0 clock 1024Hz;
239                                                  //Start A0
240      GPIO_REGA(AM_REG_GPIO_WTA_O )|= AM_REG_GPIO_WTA_WTA(START_SW + START_CLK);
241                                                    // turn ON SWITCH and CLK
242    }
243    if((INTSTATUSA)==0x00000001){
244      //INT TIMER A0 [ 4ms ] for the backscattering signal generation
245      CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)|= AM_REG_CTIMER_CTRL0_TMRA0CLR_CLEAR \
246                                        |AM_REG_CTIMER_CTRL0_TMRA0EN_DIS;
247                                       // CLEAR Counter ; Stop Counter A0
248      am_hal_ctimer_int_clear(AM_HAL_CTIMER_INT_TIMERA0);
249                                       // CLEAR TIMERB0 Interrupt
250
251      GPIO_REGA(AM_REG_GPIO_WTCA_O ) |= \
252      AM_REG_GPIO_WTCA_WTCA(START_SW + START_CLK + START_LTC);
253                               //turn OFF SWITCH , LTC and  CLK
254
255      GPIO_REGA(AM_REG_GPIO_INT0CLR_O)|= AM_REG_GPIO_INT0CLR_GPIO15(1);//Clear GPIO15 interrupt
256      GPIO_REGA(AM_REG_GPIO_INT0EN_O) |= AM_REG_GPIO_INT0EN_GPIO15(1); //Enable GPIO15 Interrupt
257      //----------------------------------------
258
259    }
260  }
```

Figura 3.15: Firmware: routine di servizio dell'interrupt generato dai timer

In *figura 3.16* è riportato un riepilogo contenente i diagrammi di flusso del firmware completo.

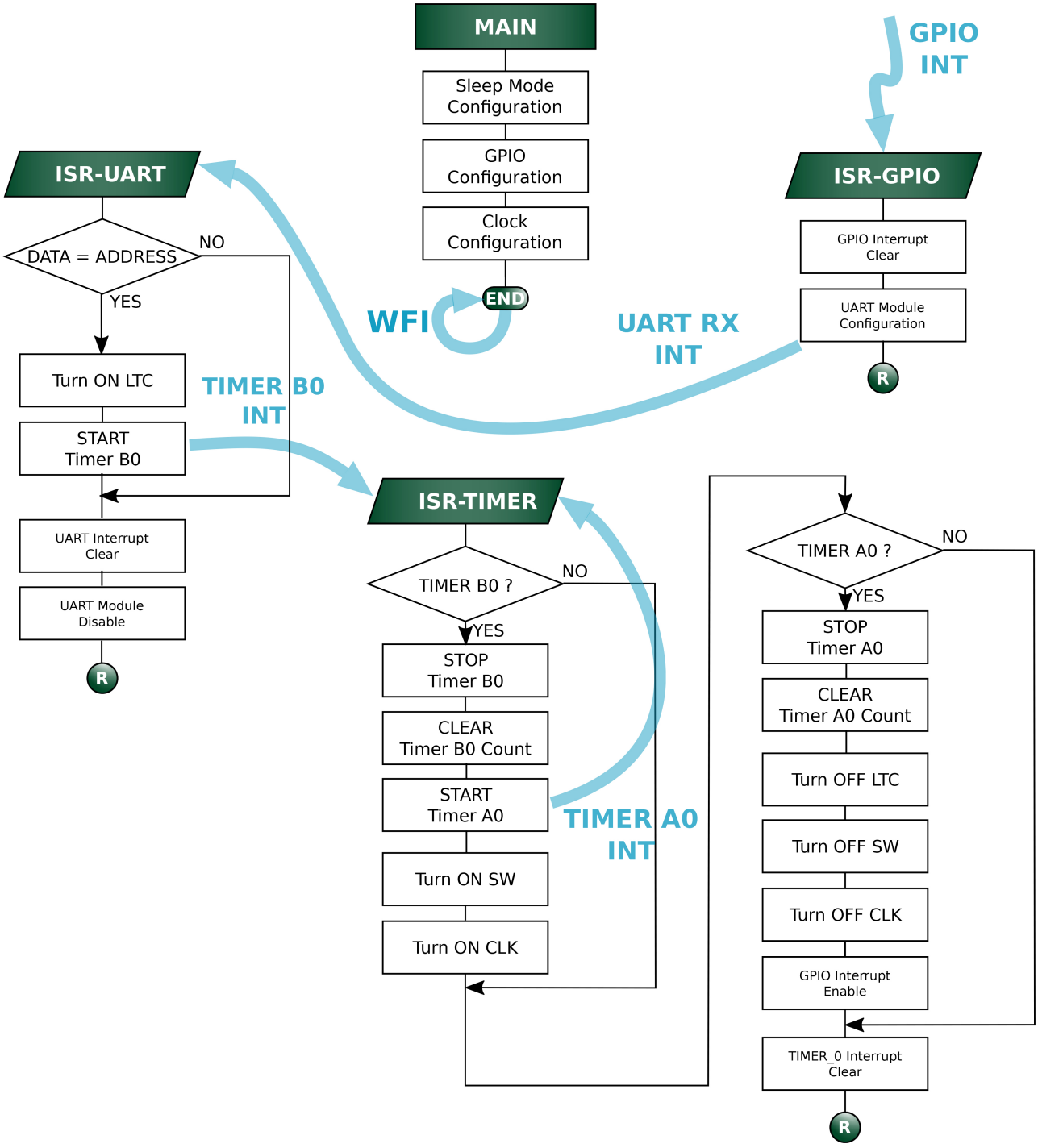

Figura 3.16: Diagramma di flusso del firmware completo

3.4 Risultati

Il software realizzato è stato infine collaudato attraverso l'utilizzo della scheda di valutazione Ambiq Apollo EVB.

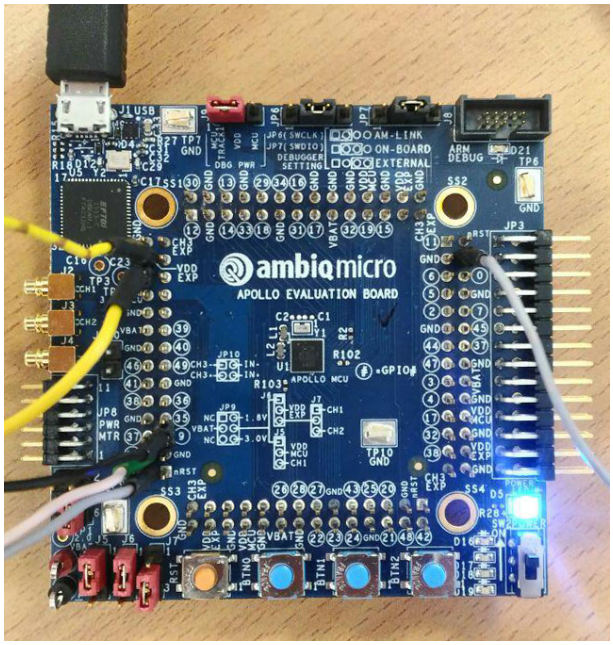

Figura 3.17: Setup dell'evaluation board Ambiq Apollo EVB

Durante questa fase, oltre all'analisi svolta tramite oscilloscopio, volta a verificare il corretto funzionamento delle procedure di attivazione, addressing e trasmissione, è stata svolta una valutazione del consumo di potenza attraverso l'utilizzo di una scheda di power monitoring.
Per l'analisi funzionale la scheda è stata collaudata in modalità debug attraverso l'ambiente di sviluppo Keil μVision, alimentando e collegando la scheda tramite USB al PC.

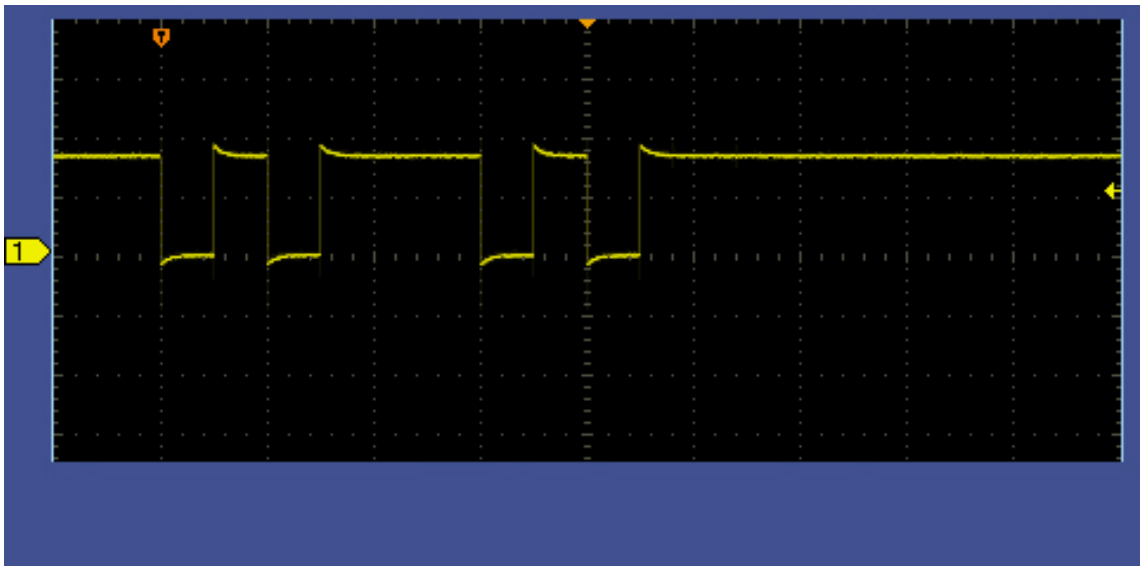

Figura 3.18: Forme d'onda: La trama ricevuta attraverso l'interfaccia UART (Nell'esempio è possibile notare l'invio del dato 0xBA)

In *figura 4.2* è presentata la forma d'onda della trama utilizzata in fase di collaudo del software, generata tramite l'utilizzo di FPGA Altera Cyclone II.
La trama in questione corrisponde al dato 0xBA (10111010 in binario) preceduto dal bit di start ("0") e seguito dal bit di stop ("1").

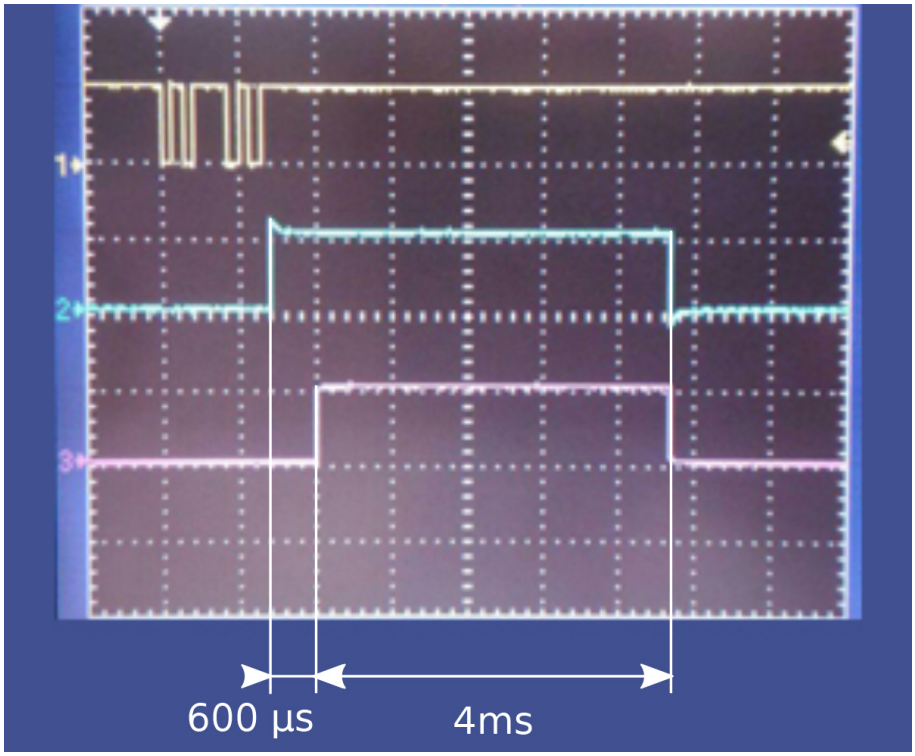

Figura 3.19: Forme d'onda: Trama UART ricevuta e segnali generati alle porte GPIO

La *figura 4.3* riporta le forme d'onda della trama in ricezione e dei segnali relativi all'architettura del tag illustrata in*figura 3.1.* La trama è sul canale 1 dell'oscilloscopio mentre i segnali di pilotaggio degli switch SW1 e SW2(e/o SW3) sono riportati rispettivamente sui canali 2 e 3. Sono sottolineati gli intervalli temporali caratteristici dei gradini così ottenuti, il primo dei quali, relativo al ritardo da 600 μs che intercorre tra l'attivazione del primo switch SW1 e l'attivazione degli switch SW2 e SW3 in contemporanea, è ripresentato nel dettaglio in *figura 4.4* .

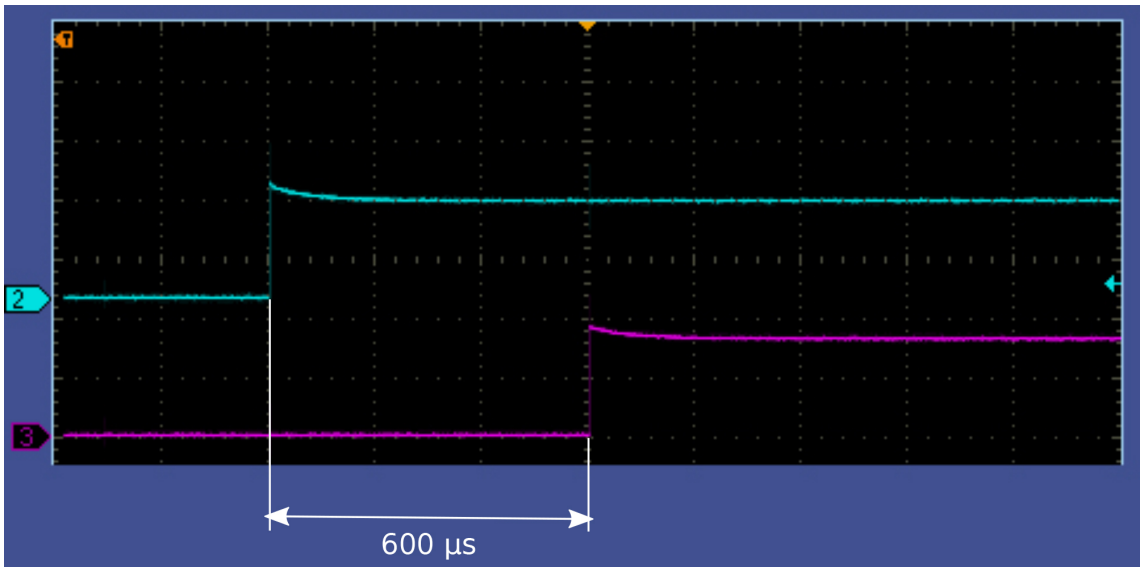

Figura 3.20: Forme d'onda: dettaglio del ritardo di μs tra l'attivazione dello switch SW1 e lo switch SW2/SW3.

Per l'analisi energetica la scheda è stata configurata in modalità stand-alone applicando un alimentazione di 2V.

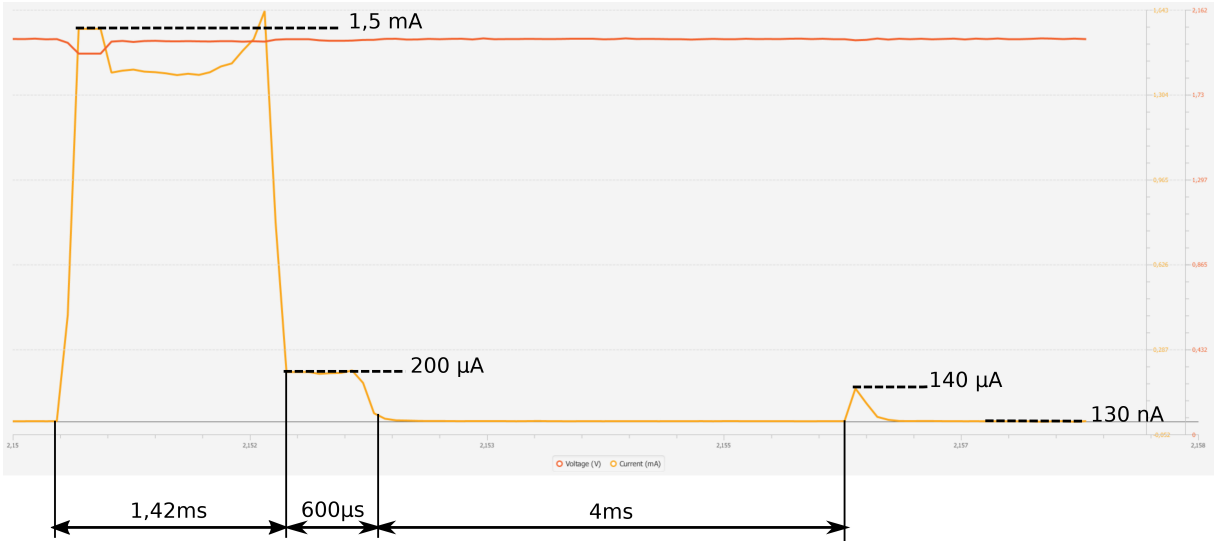

Figura 3.21: Analisi qualitativa del profilo energetico dell'applicazione attraverso Power Monitoring.

In *figura 4.5* è possibile visionare il profilo energetico dell'applicazione durante la fase di ricezione, addressing - nella condizione verificata per l'attivazione della rete di backscattering, quella cioè di caso peggiore per quanto riguarda i consumi - e trasmissione del segnale di risposta.
Si può apprezzare la presenza di diversi intervalli temporali che presentano varie fasce di consumo di corrente. Il primo intervallo è quello in cui il microcontrollore è posto in modalità deep-sleep nell'attesa di rilevare un segnale in ricezione. I consumi relativi a questa prima fase sono quelli più bassi del profilo, con una corrente di circa 130nA. Alla ricezione della trama si passa poi al secondo intervallo temporale, questa volta ad alto consumo, di durata 1,42ms, in cui il modulo UART è attivato per la ricezione. In questa fase il valore di corrente assorbita dal circuito è di circa 1,5mA. Dopo la fase di addressing si passa alla fase di generazione dei segnali di comando della rete di backscattering, divisa nei due intervalli precedentemente descritti. Il primo intervallo, di durata 600μs, è ottenuto tramite l'utilizzo del clock HFRC per il modulo del contatore ed è caratterizzato da un consumo di corrente pari a 200μA. Il secondo intervallo relativo al pilotaggio delle uscite è quello di durata pari a 4ms, ottenuto tramite l'utilizzo dell'oscillatore a bassa frequenza e bassi consumi LFRC. In questo intervallo, dato che non è utilizzato il clock ad alta frequenza, il microcontrollore si riporta in modalità low-power Deep-Sleep e consuma una corrente pari a 130nA, come nella fase iniziale.
Alla fine di questo intervallo, il programma riceve l'ultimo interrupt di fine conteggio

del timer, in seguito al quale effettua l'ultimo test tramite l'utilizzo della CPU.
É quindi presente un ultimo picco di consumo di corrente del valore di circa 140 μA, dopo il quale si ritorna allo stato iniziale, in attesa di una trama in ricezione.

Relativamente a questo profilo energetico sono stati calcolati infine le seguenti grandezze caratteristiche dell'applicazione:

Parametro	Valore	
E_{PEAK}	4,5	μJ
P_{LEAK}	260	nW
P_{TOT}	75,26	μW

Il software utilizzato per il monitoraggio di potenza fornisce il valore di energia consumata dal profilo illustrato pari a $E_{PEAK} = 4,5\mu J$;
Dal valore di corrente consumato durante le fasi in modalità Deep Sleep, pari a 130 nA, è possibile quantificare il contributo minimo di potenza consumato, moltiplicando questo valore per la tensione di alimentazione, ottenendo il parametro $P_{LEAK} = 130nA \cdot 2V = 260nW$;
La potenza totale dissipata è calcolata in relazione alla periodicità con la quale il microcontrollore esegue l'operazione di wake-up, adressing e controllo uscite. Considerando che da specifica viene richiesto di identificare almeno 15 tag al secondo, il periodo di risveglio per ogni tag è pari a T_{WAKE}=0,06s. Si ottiene quindi una potenza $P_{TOT} = P_{LEAK} + \frac{E_{PEAK}}{T_{WAKE}} = 75,26\mu W$.

Capitolo 4

Conclusioni

Con questo elaborato ci si proponeva di implementare ed analizzare il firmware dedicato alla sezione logica di controllo della Wake-up Radio illustrata. L'applicativo è stato realizzato attraverso l'utilizzo del microcontrollore Ultra-Low Power Apollo, prodotto da Ambiq Micro e basato sulla tecnologia all'avanguardia SPOT.
Il software realizzato è stato progettato in una logica di ottimizzazione del dispendio energetico, sfruttando le modalità low-power offerte dal dispositivo.
Per la gestione della comunicazione seriale si è deciso di utilizzare la periferica UART, mettendo in atto una gestione efficiente dei moduli hardware del microcontrollore, abbattendo di conseguenza i consumi.
Per l'utilizzo del microcontrollore preso in esame si è reso necessario implementare un ambiente di sviluppo per dispositivi di ultima generazione e si ha avuto modo di familiarizzare con l'utilizzo di hardware e software dedicati al monitoraggio dinamico dei consumi.

Il funzionamento del sistema è stato infine collaudato con successo ed è stata eseguita un'analisi dettagliata del profilo energetico dell'applicazione.

Tra gli sviluppi futuri si prevede la possibilità di utilizzare direttamente il chip del microcontrollore senza scheda di valutazione. Implementando una scheda ad hoc dedicata all'applicazione, costituita dai soli componenti necessari al funzionamento delle periferiche utilizzate, è infatti possibile ridurre ulteriormente i consumi.

Capitolo 5

Allegati

```
1
2   //*****************************************************************************
3   //
4   //Firmware Tag WUR
5   //
6   //Monti Michele
7   //
8   //Tesi di Laurea in Ingegneria Elettronica, Informatica e Telecomunicazioni
9   //
10  //Ottimizzazione e confronto di sistemi Wake-up Radio per applicazioni RFID
11  //attraverso l'utilizzo di microcontrollori Ultra-Low Power
12  //
13  //*****************************************************************************
14
15  #include <stdint.h>
16  #include <stdbool.h>
17  #include "am_mcu_apollo.h"
18
19
20  // Register ACCESS:
21
22  #define REGA(n)  *((uint32_t *)n)
23
24  #define ADC_REGA(n)          *((uint32_t *)(REG_ADC_BASEADDR + (0x00000000 | n)))
25  #define CLKGEN_REGA(n)       *((uint32_t *)(REG_CLKGEN_BASEADDR + (0x00000000 | n)))
26  #define MCUCTRL_REGA(n)      *((uint32_t *)(REG_MCUCTRL_BASEADDR + (0x00000000 | n)))
27  #define CTIMER_REGA(n)       *((uint32_t *)(REG_CTIMER_BASEADDR + (0x00000000 | n)))
28  #define GPIO_REGA(n)         *((uint32_t *)(REG_GPIO_BASEADDR + (0x00000000 | n)))
29  #define IOMSTR0_REGA(n)      *((uint32_t *)(REG_IOMSTR0_BASEADDR + (0x00000000 | n)))
30  #define IOMSTR1_REGA(n)      *((uint32_t *)(REG_IOMSTR1_BASEADDR + (0x00000000 | n)))
31  #define IOMSTR_REGA(n)       *((uint32_t *)(REG_IOMSTR_BASEADDR + (0x00000000 | n)))
32  #define IOSLAVE_REGA(n)      *((uint32_t *)(REG_IOSLAVE_BASEADDR + (0x00000000 | n)))
33  #define RSTGEN_REGA(n)       *((uint32_t *)(REG_RSTGEN_BASEADDR + (0x00000000 | n)))
34  #define RTC_REGA(n)          *((uint32_t *)(REG_RTC_BASEADDR + (0x00000000 | n)))
35  #define UART_REGA(n)         *((uint32_t *)(REG_UART_BASEADDR + (0x00000000 | n)))
36  #define VCOMP_REGA(n)        *((uint32_t *)(REG_VCOMP_BASEADDR + (0x00000000 | n)))
37  #define WDT_REGA(n)          *((uint32_t *)(REG_WDT_BASEADDR + (0x00000000 | n)))
38  #define SYSCTRL_REGA(n)      *((uint32_t *)(REG_SYSCTRL_BASEADDR + (0x00000000 | n)))
39  //
40  #define UART_DATA  UART_REGA(AM_REG_UART_DR_DATA(AM_REG_UART_DR_O))
41  //-----------------------------------------------------------------------------------
42
43
44  short unsigned int ADDRESS = 0xBA;
45  unsigned int temp;
46  unsigned int INTSTATUS, INTSTATUSA, INTSTATUSB;
47  short unsigned int UARTD ;
48  short unsigned int UARTDbe; //Big Endian received DATA
49  void setup_serial(int32_t i32Module, uint32_t ui32BaudRate);
50
51  //GPIO Define
52
53  #define RESET          0x00000001
54  #define TXD_OUT        0x00000010
55  #define START_CLK      0x00000100
56  #define START_SW       0x00001000
57  #define START_LTC      0x00000400
58
59
60  int main(void){
61
62  // STOP WatchDog
63    WDT_REGA(AM_REG_WDT_CFG_O) &= ~AM_REG_WDT_CFG_WDTEN(1);// Disable Watchdog Timer
64
65  //-----------------------------------------------
66
67  //Low Power configuration:
68  SYSCTRL_REGA(AM_REG_SYSCTRL_SCR_O) |= AM_REG_SYSCTRL_SCR_SEVONPEND(1)\
69                      | AM_REG_SYSCTRL_SCR_SLEEPONEXIT(1);
70                      //pending event=WUE ;
71                      //return from ISR = go to sleep;
72  am_hal_sysctrl_sleep(AM_HAL_SYSCTRL_SLEEP_DEEP);
73                      //the sleep mode is deep sleep
74
75  //-----------------------------------------------
76
77  // Clock Registers Configuration:
78
```

```
  79      CLKGEN_REGA(AM_REG_CLKGEN_CLKKEY_O) = AM_REG_CLKGEN_CLKKEY_KEYVAL; //Insert KEYVAL
  80      CLKGEN_REGA(AM_REG_CLKGEN_OCTRL_O) |=  AM_REG_CLKGEN_OCTRL_STOPXT_STOP;
  81                                             //Stop the XT Oscillator from driving the RTC
  82      am_hal_clkgen_sysclk_select(AM_HAL_CLKGEN_SYSCLK_MAX);
  83                                             //System Clock = 24MHz
  84
  85
  86    //Pad function Configuration:
  87
  88      GPIO_REGA(AM_REG_GPIO_PADKEY_O)  =  AM_REG_GPIO_PADKEY_KEYVAL; //Insert KEYVAL
  89
  90      GPIO_REGA(AM_REG_GPIO_PADREGD_O) |= AM_REG_GPIO_PADREGD_PAD15FNCSEL_GPIO15 \
  91                                        | AM_REG_GPIO_PADREGD_PAD15INPEN_M;
  92
  93      GPIO_REGA(AM_REG_GPIO_PADREGC_O) |= AM_REG_GPIO_PADREGC_PAD8FNCSEL_GPIO8 \
  94                                        |  AM_REG_GPIO_PADREGC_PAD8STRNG_HIGH; // Pad 8 = START_CLK
  95      GPIO_REGA(AM_REG_GPIO_PADREGC_O) |= AM_REG_GPIO_PADREGD_PAD12FNCSEL_GPIO12 \
  96                                        |  AM_REG_GPIO_PADREGD_PAD12STRNG_HIGH; // Pad 12 = START_SW
  97      GPIO_REGA(AM_REG_GPIO_PADREGC_O) |= AM_REG_GPIO_PADREGC_PAD10FNCSEL_GPIO10\
  98                                        |  AM_REG_GPIO_PADREGC_PAD10STRNG_HIGH; // Pad 10 = START_LTC
  99
 100
 101
 102      AM_REG(GPIO, ENA) |= 0x9500; //ENABLE GPIO (15,8,10,12)
 103    /*
 104      UNUSED GPIO   -> " The REG_GPIO_PADREGy_PADnINPEN bit must be set to enable the pad input, and
      should be left
 105                        clear whenever the pad is not used in order to eliminate any leakage current in
      the pad."
 106                       " If the pad is not configured as an output, the
 107                       pad enable is forced high to turn the driver off"
 108
 109    */
 110
 111    //GPIO Configuration:
 112      GPIO_REGA(AM_REG_GPIO_CFGB_O)|= AM_REG_GPIO_CFGB_GPIO15INCFG_READ\
 113                                    |AM_REG_GPIO_CFGB_GPIO15INTD_INTHL;
 114                                    //(Pad 15 input enable
 115                                    //interrupt on High-to-Low transistion
 116                                    //for UART Module enable
 117      GPIO_REGA(AM_REG_GPIO_CFGB_O)|= AM_REG_GPIO_CFGB_GPIO8OUTCFG_PUSHPULL\
 118                                    |AM_REG_GPIO_CFGB_GPIO12OUTCFG_PUSHPULL \
 119                                    |AM_REG_GPIO_CFGB_GPIO10OUTCFG_PUSHPULL;
 120                                    //(PUSHPULL Pad 8; Pad 12; Pad 10)
 121
 122      GPIO_REGA(AM_REG_GPIO_WTA_O )&= AM_REG_GPIO_WTA_WTA(0x00000000); //Default value "0"
 123      GPIO_REGA(AM_REG_GPIO_WTB_O )&= AM_REG_GPIO_WTB_WTB(0x0000); //Default value "0"
 124
 125
 126      //am_hal_gpio_pin_config(15, AM_HAL_PIN_15_UARTRX); //GPIO15<- UARTRX
 127
 128
 129
 130      //Interrupt GPIO1 configuration--------------
 131      GPIO_REGA(AM_REG_GPIO_INT0CLR_O)|= AM_REG_GPIO_INT0CLR_GPIO15(1);//Clear GPIO15 interrupt
 132      GPIO_REGA(AM_REG_GPIO_INT0EN_O) |= AM_REG_GPIO_INT0EN_GPIO15(1); //Enable GPIO15 Interrupt
 133
 134      AM_REG(NVIC, ISER0) = 0x1 << ((25 - 16) & 0x1F); //Enable GPIO Interrupts for the NVIC
 135
 136      AM_REGn(GPIO, 0, PADKEY) = 0; //Lock GPIO register
 137
 138
 139
 140    //Timers Configuration
 141
 142    CTIMER_REGA(AM_REG_CTIMER_CMPRA0_O) = 3;  //count 4ms
 143    CTIMER_REGA(AM_REG_CTIMER_CMPRB0_O) = 1800; //count 600us : 3*600=1800
 144
 145    CTIMER_REGA(AM_REG_CTIMER_INTEN_O)|=  AM_REG_CTIMER_INTEN_CTMRA0INT(1);
 146                                       //Enable Timer A0 Interrupt
 147    CTIMER_REGA(AM_REG_CTIMER_INTEN_O)|=  AM_REG_CTIMER_INTEN_CTMRB0INT(1);
 148                                        //Enable Timer B0 Interrupt
 149
 150    AM_REG(NVIC, ISER0) = 0x1 << ((26 - 16) & 0x1F);////Enable CTIMER Interrupts for the
      NVIC
 151
 152    //-------------------------------------------------------------------------------------
 153
```

```
154
155
156   //Before the UART Module can communicate, a clock frequency must be selected for the UART clock, and
157   //the clock must then be enabled. Use the UART_CR register to select the desired clock frequency,
      FUART.
158   //After selecting the desired frequency, enable the UART clock using the CLK_GEN_UARTEN register.
159   //Unlike other modules that automatically turn off their clock sources automatically, the UART clock
      must be
160   //manually disabled using the aforementioned CLK_GEN_UARTEN register. To ensure minimum energy
161   //operation, the UART clock should be enabled for the minimum time possible and should be disabled as
162   //soon as UART communication is complete.
163   //----------------------
164   }
165
166
167   //-------------------ISR-------------------------------
168
169
170   //ISR GPIO15 UARTRX for transmission detection
171   void
172   am_gpio_isr(void){
173
174   GPIO_REGA(AM_REG_GPIO_INT0CLR_O)|= AM_REG_GPIO_INT0CLR_GPIO15(1);
175                                                   //Clear interrupt
176   GPIO_REGA(AM_REG_GPIO_INT0EN_O) &= ~AM_REG_GPIO_INT0EN_GPIO15(1);
177                                                 //Disable GPIO Interrupt
178   setup_serial(0, 7000);  //UART Configuration - BR =7kHz
179
180   }
181
182
183
184   //ISR UART for TAG addressing
185   void
186   am_uart_isr(void){
187
188       am_hal_uart_int_clear(0,0x7FF); // reset UART interrupts
189
190       UARTD = UART_DATA;
191       UARTDbe = 0; //Big Endian received DATA
192       for (int i=0;i<8;i++){
193         UARTDbe |= (((UARTD>>(7-i))&0x1) << (i));
194       }
195
196       if (UARTDbe == ADDRESS){
197         CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)&=AM_REG_CTIMER_CTRL0_TMRB0CLR_RUN;
198                                           //Allow Timer B0 to RUN
199         CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)|=  AM_REG_CTIMER_CTRL0_TMRB0IE_EN \
200                         |AM_REG_CTIMER_CTRL0_TMRB0FN_CONTINUOUS\
201                         |AM_REG_CTIMER_CTRL0_TMRB0CLK_HFRC_DIV8 \
202                         |AM_REG_CTIMER_CTRL0_TMRB0EN_EN;;
203                                           //TIMER B0 interrupt enable;
204                                           //Continuous Mode;
205                                           //CONF TIMER B0 clock 3MHz;
206                                           //Start A0
207
208         GPIO_REGA(AM_REG_GPIO_WTA_O )|= AM_REG_GPIO_WTA_WTA(START_LTC);
209                                       //turn on LTC6906 (CLOCK GENERATOR)
210           }
211
212       CLKGEN_REGA(AM_REG_CLKGEN_UARTEN_O) &= ~AM_REG_CLKGEN_UARTEN_UARTEN_EN;
213                                           //disable UART system clock
214
215
216     am_hal_uart_disable(0); //Disable UART
217   }
218
219
220
221   // ISR CTIMER
222   void
223   am_ctimer_isr(void){
224     INTSTATUS = am_hal_ctimer_int_status_get(true);
225                       //get ctimer interrupt status
226     INTSTATUSA = INTSTATUS & 0x00000001;
227     INTSTATUSB = INTSTATUS & 0x00000002;
228     if((INTSTATUSB)==0x00000002) {
229       //INT TIMER B0 [ 600uS ] for 600us timing after that the clock generator is ready
```

```
230       CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)|= AM_REG_CTIMER_CTRL0_TMRB0CLR_CLEAR \
231                                           |AM_REG_CTIMER_CTRL0_TMRB0EN_DIS;
232                                           // CLEAR Counter ; Stop Counter B0
233       am_hal_ctimer_int_clear(AM_HAL_CTIMER_INT_TIMERB0);
234                                           // CLEAR TIMERB0 Interrupt
235       CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)&=AM_REG_CTIMER_CTRL0_TMRA0CLR_RUN;
236                                           //Allow Timer A0 to RUN
237       CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)|=AM_REG_CTIMER_CTRL0_TMRA0IE_EN\
238                                           |AM_REG_CTIMER_CTRL0_TMRA0FN_CONTINUOUS\
239                                           |AM_REG_CTIMER_CTRL0_TMRA0CLK_LFRC  \
240                                           |AM_REG_CTIMER_CTRL0_TMRA0EN_EN;
241                                                     //TIMER A0 interrupt enable;
242                                                     //Continuous Mode;
243                                                     //CONF TIMER A0 clock 1024Hz;
244                                                     //Start A0
245       GPIO_REGA(AM_REG_GPIO_WTA_O )|= AM_REG_GPIO_WTA_WTA(START_SW + START_CLK);
246                                                           // turn ON SWITCH and CLK
247     }
248     if((INTSTATUSA)==0x00000001){
249       //INT TIMER A0 [ 4ms ] for the backscattering signal generation
250       CTIMER_REGA(AM_REG_CTIMER_CTRL0_O)|= AM_REG_CTIMER_CTRL0_TMRA0CLR_CLEAR \
251                                           |AM_REG_CTIMER_CTRL0_TMRA0EN_DIS;
252                                         // CLEAR Counter ; Stop Counter A0
253       am_hal_ctimer_int_clear(AM_HAL_CTIMER_INT_TIMERA0);
254                                         // CLEAR TIMERB0 Interrupt
255
256       GPIO_REGA(AM_REG_GPIO_WTCA_O ) |= \
257       AM_REG_GPIO_WTCA_WTCA(START_SW + START_CLK + START_LTC);
258                               //turn OFF SWITCH , LTC and  CLK
259
260       GPIO_REGA(AM_REG_GPIO_INT0CLR_O)|= AM_REG_GPIO_INT0CLR_GPIO15(1);//Clear GPIO15 interrupt
261       GPIO_REGA(AM_REG_GPIO_INT0EN_O) |= AM_REG_GPIO_INT0EN_GPIO15(1); //Enable GPIO15 Interrupt
262       //-----------------------------------------
263
264     }
265   }
266
267
268   am_hal_uart_config_t g_sUartConfig =
269   {
270       .ui32BaudRate = 115200,
271       .ui32DataBits = AM_HAL_UART_DATA_BITS_8,
272       .bTwoStopBits = false,
273       .ui32Parity   = AM_HAL_UART_PARITY_NONE,
274       .ui32FlowCtrl = AM_HAL_UART_FLOW_CTRL_NONE,
275   };
276
277
278   void
279   setup_serial(int32_t i32Module, uint32_t ui32BaudRate)
280   {
281       //
282       // Make sure the UART RX and TX pins are enabled.
283       //
284
285
286       am_hal_gpio_pin_config(0, AM_HAL_PIN_0_UARTTX);
287       am_hal_gpio_pin_config(15, AM_HAL_PIN_15_UARTRX);
288
289
290       //
291       // Start the UART interface, apply the desired configuration settings, and
292       // enable the FIFOs.
293       //
294       am_hal_uart_clock_enable(i32Module);
295
296       //
297       // Disable the UART before configuring it.
298       //
299       am_hal_uart_disable(i32Module);
300
301       //
302       // Configure the UART.
303       //
304       g_sUartConfig.ui32BaudRate = ui32BaudRate;
305       am_hal_uart_config(i32Module, &g_sUartConfig,AM_REG_UART_CR_CLKSEL_3MHZ);
306
307
```

Bibliografia

[1] Bill Glover, Himanshu Bhatt *RFID Essentials*. O'Reilly Media, Inc, Sebastopol, CA 2006. ISBN 0596009445

[2] A. Romani *Dispensa del corso di Elettronica dei Sistemi Digitali L-A: Protocolli di Comunicazione*. Università di Bologna, Cesena, A.A. 2005-2006 http://www-micro.deis.unibo.it/ romani/Dida01/lezioni/protocolli_comunicazione_v3.pdf

[3] Ambiq Micro Inc. *Apollo Datasheet Ultra-Low Power MCU Family*. ©2016 Ambiq Micro, Inc., Austin, TX 78730-1156 February 2016.

[4] Ambiq Micro Inc. *White Paper: Sub-Threshold Design - A Revolutionary Approach to Eliminating Power*. ©2016 Ambiq Micro, Inc., Austin, TX 78730-1156 May 2014.

[5] Ambiq Micro Inc. *Application Note: Apollo EVK User's Guide, Rev. 0.2*. ©2016 Ambiq Micro, Inc., Austin, TX 78730-1156 February 2016.

[6] *Cortex-M4F Devices Generic User Guide, document number DUI0553A*. Copyright © 2010 ARM. All rights reserved. ARM DUI 0553A (ID121610).

[7] *ARM®v7-M Architecture Reference Manual*. Copyright © 2006-2008, 2010, 2014 ARM. All rights reserved. ARM DDI 0403E.b (ID120114).

[8] AMS *AS3930 Single Channel Low Frequency Wakeup Receiver*. AMS AS3930 Datasheet [v1-62] 2014-Nov-27.

[9] M. Magno, L. Benini *Article: An Ultra Low Power High Sensitivity Wake-Up Radio Receiver with Addressing Capability*. The Second International Workshop on GReen Optimized Wireless Networks, 2014.

[10] S. Bdiri, F. Derbel *Article: An Ultra-Low Power Wake-Up Receiver for Real-time constrained Wireless Sensor Networks*. Leipzig University of Applied Sciences, Wachter Str. 13, 04107, Leipzig, Germany - DOI 10.5162/sensor2015/D6.2.

[11] M. Magno, V. Jelicic, B. Srbinovski, V. Bilas, E. Popovici, L. Benini *Article: Design, Implementation, and Performance Evaluation of a Flexible Low-Latency Nanowatt Wake-Up Radio Receiver*. IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 12, NO. 2, APRIL 2016.

[12] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, L. Reindl *Article: Performance Evaluation and Comparative Analysis of SubCarrier Modulation Wake-up Radio Systems for Energy-Efficient Wireless Sensor Networks*. Sensors 2014, 14, 22-51; doi:10.3390/s140100022 - ISSN 1424-8220 www.mdpi.com/journal/sensors.

[13] S. J. Marinkovic, E. M. Popovici *Journal: Nano-Power Wireless Wake-Up Receiver With Serial Peripheral Interface*. IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 29, NO. 8, SEPTEMBER 2011.

[14] S. Marinkovic, E Popovici *Article:Nano-Power Wake-Up Radio Circuit for Wireless Body Area Networks*. Dept. of Microelectronic Engineering, University College Cork, Ireland.

[15] J. Donovan *Article: New Applications for Energy Harvesting*. http://www.mouser.it/applications/energy-harvesting-new-applications/ - Copyright ©2017 Mouser Electronics, Inc

[16] Institute of Physics *Article: Energy harvesting*. https://www.iop.org/resources/energy/ - © Institute of Physics (the "Institute") and IOP Publishing 2014.

[17] MAXIM *Article:Energy Harvesting Systems Power the Powerless*. https://www.maximintegrated.com/en/app-notes/index.mvp/id/5259 - APPLICATION NOTE 5259 - © Mar 29, 2012, Maxim Integrated Products, Inc.

Elenco delle figure