UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN

INGEGNERIA DELLE TELECOMUNICAZIONI

(CLASSE DELLE LAUREE IN INGEGNERIA DELL'INFORMAZIONE N.9)

DIPARTIMENTO DI INGEGNERIA ELETTRICA

ELABORATO DI LAUREA

REALIZZAZIONE DI UNA SCHEDA INPUT/OUTPUT ANALOGICA/DIGITALE PER IL LABORATORIO "REMOTO" DI CIRCUITI

RELATORE

CH.MO PROF. ING.

MASSIMILIANO DE MAGISTRIS

CORRELATORE

ING. MASSIMO ATTANASIO

CANDIDATO

GIOVANNA MIRRA

MATR. 540/609

ANNO ACCADEMICO 2006/2007

Alla mia famiglia

e

ai miei nonni

Ringraziameti

Desidero ringraziare il Professore Massimiliano de Magistris, l'Ing. Massimo Attanasio e l'Ing. Marco Colandrea per i consigli e le informazioni necessarie per la buona riuscita di questo elaborato.
Infine ringrazio tutti coloro che mi sono stati vicino.

INDICE

Introduzione

Oggi la didattica a distanza è diventata un potente strumento di insegnamento dal momento che essa consente di raggiungere gli studenti in luoghi e tempi non più strettamente dettati dalle istituzioni scolastiche. Le nuove tecnologie hanno reso possibile superare molti degli ostacoli legati alla distanza. La posta elettronica, le chat e le video conferenze live interattive, per esempio, costituiscono utili strumenti per mettere in contatto persone coinvolte in un processo di insegnamento. Inoltre, hanno avuto grande diffusione applicazioni web per la simulazione didattica e prototipi di "laboratori virtuali" capaci di rendere disponibili senza vincoli di spazio e di tempo risorse che solitamente sono disponibili unicamente in un complesso scolastico.

La simulazione risponde alle esigenze della sperimentazione, ma non fornisce le stesse possibilità e le stesse opportunità che solitamente sono offerte dall'operare all'interno di un laboratorio sperimentale. In alcuni settori, come quello della fisica e dell'ingegneria, le risposte fornite da una sperimentazione reale sono fondamentali.

Molti dei tool per la didattica a distanza via Internet sono, invece, delle applicazioni software per la simulazione, ossia "laboratori virtuali". Questi ultimi, per definizione, pongono dei limiti alle possibilità degli studenti e dei docenti di effettuare sperimentazioni negli ambienti di un laboratorio reale. Inoltre la progettazione di un software di simulazione dipende largamente da come la percezione dello studente viene intesa dal progettista. Potenzialmente le varie procedure che lo studente deve eseguire potrebbero richiedere un livello di conoscenze superiore a quello in suo possesso. Un passo estraneo alla sequenza prevista di operazioni renderebbe l'intero esercizio un inutile tentativo. Le conoscenze acquisite attraverso esperimenti simulati dipendono in larga misura dal progetto, dai vincoli e dal costo del software. Le applicazioni software di simulazione producono approssimazioni che possono condurre a risultati errati. Sotto queste condizioni, l'apprendimento dello studente dipenderà dalla qualità del software più che dalle capacità intellettive dello studente. Inoltre esperimenti condotti tramite tool di simulazione privano gli studenti della libertà di poter operare in differenti condizioni sperimentali, che possono essere realizzate con differenti configurazioni

degli apparati del laboratorio reale. Per questo motivo pensiamo che la simulazione abbia un suo ruolo, ma non è un completo sostituto di un laboratorio reale.
Il laboratorio "remoto", al contrario, coinvolge i sensi individuali dello studente e ne agevola il processo di apprendimento. L'elemento di realtà da essa riprodotto rende lo studente vero protagonista degli esperimenti, perché gli consente di effettuare, via Internet, esperimenti reali grazie alla possibilità di controllare remotamente i dispositivi del laboratorio tramite delle interfacce software opportunamente progettate.
In questo elaborato viene illustrato il laboratorio "remoto" di Circuiti, realizzato presso il laboratorio di elettrotecnica, il quale rappresenta il risultato di un' attività di progettazione e sviluppo, che è stata realizzata da tesisti e tirocinanti, al fine di rendere possibile l' interazione in remoto con esperimenti automatizzati. Esso presentava inizialmente la seguente architettura: un PC collegato al circuito mediante sonde ed equipaggiato di software LabView 7.1 e di una scheda di acquisizione interna, della National Instrument, modello PCI-5102, utilizzata in modalità operativa da oscilloscopio digitale. Una scheda d'acquisizione interna, prodotta dalla National Instrument, modello PCI-6040E, collegata al circuito. Infine un circuito, quello caotico di Chua, con cui era possibile effettuare esperienze monoutente. Data l'esigenza di ampliare il laboratorio "remoto" con altri esperimenti e l'impossibilità della scheda interna di poterli pilotare a causa del limitato numero di terminali digitali si è deciso di realizzare una scheda di acquisizione ed un multiplexer, in modo tale da poter introdurre altri esperimenti ed effettuare la selezione in remoto dell'esperimento di interesse.
Con l'inserimento di questi due elementi, è stata modificata l' architettura del laboratorio, la quale attualmente si presenta come di seguito riportato: Il PC collegato alla scheda I/O A/D mediante interfaccia USB ed equipaggiato di software e scheda di acquisizione. La scheda I/O A/D, esterna, collegata al multiplexer. Il multiplexer, che serve a selezionare l'esperimento desiderato. Infine gli esperimenti, con cui è possibile effettuare esperienze, attualmente sono il circuito di Chua ed uno non autonomo con induttanza e diodo RLD.
Successivamente viene descritta la realizzazione hardware e software della scheda I/O A/D. In particolare vengono descritti i vari integrati che la compongono e la realizzazione dei loro collegamenti; vengono descritti i SubVI, che implementano le funzionalità di configurazione e controllo della scheda ed infine l'inserimento di alcuni di essi nel programma in LabView 7.1 che riproduce il pannello frontale dell' oscilloscopio, tramite il quale l'utente può interagire con l'esperimento selezionato.

Capitolo 1

Il laboratorio “remoto” di Circuiti

Il laboratorio “remoto” di Circuiti è un sistema hardware/software che consente agli utenti di interagire con processi fisici dislocati in altri luoghi attraverso la rete Internet (o altri tipi di rete), permettendo la fruizione di veri e propri esperimenti (pre-definiti) ad utenti in luoghi diversi dal laboratorio. Esso rappresenta il risultato di un’ attività di progettazione e sviluppo che è stata realizzata da alcuni studenti al fine di rendere possibile l’ esecuzione in remoto di esperimenti automatizzati presso il laboratorio di Circuiti. Vediamo più nel dettaglio come è strutturato. Possiamo considerare la seguente architettura:

- Interfaccia web
- Server del laboratorio
- Schede di acquisizione
- Esperimenti

I primi due componenti sono caratteristici di un’ architettura base di un laboratorio in remoto, in quanto il loro scopo principale è la definizione del modo in cui un utente si interfaccia con il laboratorio remoto. In particolare l’ interfaccia web mostra all’ utente il pannello frontale della strumentazione utilizzata: l’ oscilloscopio digitale, realizzato tramite l’ ambiente di sviluppo National Instruments LabView 7.1, in modo da rendere accessibile agli utenti tutte le funzioni dello stesso. Oltre al software per il controllo e la gestione degli esperimenti del laboratorio, è stato sviluppato un sistema ipertestuale per la didattica in relazione alle tematiche teoriche che affiancheranno le procedure operative del laboratorio; mentre il server del laboratorio è equipaggiato di software e scheda di acquisizione. In particolare, il software è il LabView, prodotto dalla NI, linguaggio grafico specifico per l’ acquisizione dei dati e misure; la scheda di acquisizione, invece, è un modello PCI-5102, sempre prodotta dalla NI.

Gli altri due componenti, invece, sono fondamentali per la realizzazione del laboratorio remoto. In particolare la scheda NI-SCOPE, interna al server, offre le funzionalità di un oscilloscopio digitale espanso, in quanto prevede una serie di funzioni di elaborazione e di memorizzazione delle forme d' onda acquisite; inoltre la scheda I/O A/D, esterna al PC con collegamento USB, per l' acquisizione ed elaborazione dei segnali dal PC e dagli esperimenti ad esso collegati ovvero, il circuito di Chua e il circuito RLD.

Possiamo osservare che l'utilizzo di Internet, consente una gestione più efficiente delle risorse del laboratorio che diventano fruibili durante tutto l'arco temporale della giornata e da qualunque computer collegato alla rete; crea un effetto "presenza" in laboratorio grazie alla visualizzazione dell' esperimento nella pagina web. Di conseguenza con l'utilizzo di questi strumenti gli studenti sono maggiormente motivati ad effettuare esperienze pratiche, acquisendo una più completa conoscenza della materia. Inoltre la piena accessibilità a differenza dei laboratori tradizionali, gli consente di effettuare esperienze ed esercizi pratici anche da casa a qualunque orario, anche durante le ore in cui i laboratori tradizionali restano chiusi; ciò comporta un incremento dell' efficienza ed una maggiore conoscenza dell' utilizzo degli strumenti. Infine tale laboratorio può essere utilizzato senza difficoltà anche da studenti con handicap motori.

1.1 Interfaccia Web

Al fine di rendere possibile la connessione al laboratorio tramite Internet, viene reso disponibile un link ,dal sito del Dipartimento di ingegneria elettrica, www.diel.unina.it -> laboratori-> circuiti e diagnostica elettrica e magnetica-> esperimenti on line. Il sito web, realizzato per fini didattici, attualmente consente di accedere all' esperimento con il circuito di Chua. Esso si presenta come mostrato in figura 1, in cui oltre al software per il controllo e la gestione dell' esperimento, che vedremo più avanti, è stato sviluppato un sistema ipertestuale per la didattica in relazione alle tematiche teoriche che affiancheranno le procedure operative del laboratorio.

Figura 1- Sito web

In particolare sono stati realizzati i seguenti collegamenti:

- Generalità: in cui è descritto il circuito di Chua
- Teoria: in cui sono descritti le dinamiche complesse, le biforcazioni e il caos nei circuiti caotici.

- Funzionamento: in cui sono riportati i modi di funzionamento del circuito di Chua, variando il valore della resistenza (trimmer).
- Implementazione: in cui sono descritti la realizzazione del Diodo di Chua e la scelta dell' induttore.
- Acquisizione: sono descritti il software LabView 7.1 e la scheda di acquisizione installata nel PC , la NI-PCI 5102, utilizzati per l' acquisizione dei dati e controllo dell' esperimento.
- Simulazione: viene illustrato il pannello frontale dello oscilloscopio

Per accedere alla pagina dell' esperimento, è necessario che l' utente abbia installati i plug-in di LabView sul proprio computer, in quanto consentono di utilizzare l'interfaccia, che garantisce il controllo remoto dell'esperimento ovvero, ci permettono di visualizzare sul PC il pannello frontale dell'oscilloscopio digitale (diagrammi nel tempo, nelle spazio delle fasi, e rappresentazione della caratteristica non lineare) in modo tale da poter effettuare esperienze sul circuito di Chua e ottenere sullo schermo la visualizzazione dei risultati. L' interfaccia è stata realizzata per un progetto precedente, utilizzando un PC ed una scheda d'acquisizione analogica della National Instruments, la "NI-5102-PCI", il cui software di controllo è stato sviluppato in LabView 7.1**,** linguaggio grafico di programmazione specifico nell'acquisizione dei dati e la loro presentazione. In particolare, è stato creato un applicativo del linguaggio (un VI) che riproducesse sullo schermo del PC i controlli usuali di un oscilloscopio su cui agire. Più nel dettaglio, sul **pannello frontale**, mostrato in figura 2, trovano posto i vari componenti grafici e serve ad organizzare l'interfaccia utente: il pulsante di stop, i pulsanti di attivazione dei 2+2 canali, la selezione dell'ampiezza verticale e della base dei tempi (che indirettamente fissa la frequenza di campionamento della scheda) la configurazione del sistema di trigger ed infine il display dell'output dell'oscilloscopio.

Questo è stato realizzato con due pannelli grafici, uno per la modalità standard 2/4 tracce ed un altro per la visualizzazione in modalità X-Y, posizionati sovrapposti e che vengono selettivamente visualizzati secondo necessità mediante apposito pulsante presente in alto sull'interfaccia.

Inoltre sono stati inseriti due grafici di tipo X-Y Graph, che consentono di descrivere rispettivamente la pendenza della retta della caratteristica non lineare del Diodo di Chua, e della caratteristica del Trimmer.

Durante il Run–Time, inoltre, mediante l'inserimento di un controllore digitale affianco ai due grafici, è possibile far variare il numero che identifica la pendenza all'interno di un range di valori positivi e negativi, e al tempo stesso vedere come cambia la stessa graficamente. L'interfaccia creata è stata poi remotizzata tramite un "tool di pubblicazione web" presente in LabView stesso [1].

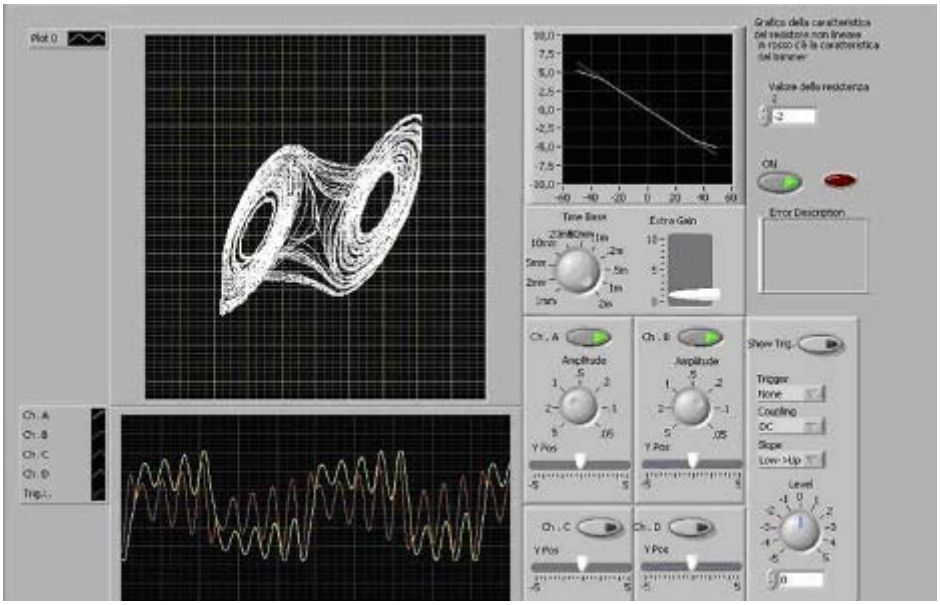


Figura 2- Interfaccia web (oscilloscopio digitale)

1.2 Esperimenti sui circuiti caotici

Molto interessanti sono i sistemi fisici che presentano comportamenti di tipo caotico, perchè mostrano che il caos non è né un'astrazione matematica né un fenomeno derivante dagli errori introdotti dagli algoritmi di calcolo numerico, ma un fenomeno reale.

Tutti i sistemi non lineari posseggono, per particolari scelte dei parametri di funzionamento, delle dinamiche caotiche; poiché è stato dimostrato che, è sufficiente un sistema autonomo del terzo ordine o un sistema non autonomo del secondo ordine affinché, potenzialmente, vi si possa sviluppare il caos. Tra i molti sistemi meccanici o circuiti elettronici fisicamente realizzabili che presentano dinamiche caotiche, sono stati realizzati per il laboratorio: il circuito di Chua (il quale fa parte dei sistemi autonomi, caratterizzati da generatori di tensione, ed eventualmente di corrente, variabili nel tempo) ed il circuito RLD (il quale fa parte dei sistemi non autonomi, in cui i generatori sono costanti, nel senso che non variano nel tempo).

1.2.1 Circuito RLD

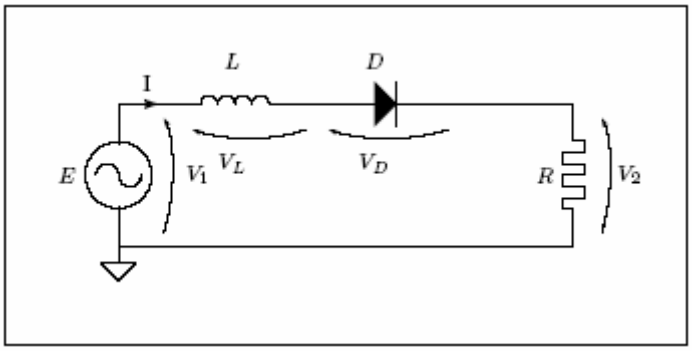


Figura 3– Circuito RLD

Il circuito RLD, mostrato in figura 3, è composto dalla serie di una induttanza, una resistenza ed un diodo ed è alimentato da un generatore sinusoidale ad ampiezza e frequenza variabile. Il solo componente non lineare è il diodo mentre resistenza ed induttanza sono da considerarsi funzionanti in regione lineare (si considera l' induttanza funzionante non in saturazione). Del diodo è certamente importante la capacità inversa

infatti i fenomeni più significativi si notano ad una frequenza del generatore prossima a quella della risonanza serie della induttanza e della capacità del diodo. Per questo motivo è necessario scegliere o un diodo di potenza lento o un varicap, non vanno invece bene i diodi di commutazione di piccolo segnale o i diodi veloci perchè hanno capacità inversa e tempo di recupero inverso troppo piccolo costringendo il circuito a lavorare a frequenze troppo elevate per avere fenomeni significativi.

La selezione del diodo, dei valori della induttanza e della resistenza e della frequenza di lavoro del generatore è stato oggetto di una tesina precedentemente svolta da altri studenti nell'ambito dell'esame di teoria dei circuiti. La linea guida che si è seguita principalmente è stata di trovare una frequenza di lavoro accettabile (non troppo elevata) per la strumentazione a disposizione.

Per far ciò si è selezionato un diodo, tra le tantissime scelte possibili, che presentasse una capacità non piccola, per questo motivo la scelta è caduta su un diodo di potenza invece che su un diodo di segnale.

La resistenza serie si è scelta relativamente piccola per non abbassare il fattore di merito del circuito risonante, si ricordi infatti che per piccoli valori della tensione il circuito può, in linea di principio, essere linearizzato ottenendo proprio un circuito R-L-C serie. La presenza di questa resistenza ha anche l'utile effetto collaterale di permettere la misura della corrente che passa nel circuito. La frequenza di funzionamento ottimale è stata cercata per tentativi, fino a trovarne una che presentasse un ampio spettro di dinamiche (soluzione fondamentale, sub-armoniche, caos, finestre periodiche nel caos).

E'comunque interessante studiare il circuito RLD facendo variare sia la frequenza che l'ampiezza del segnale di pilotaggio, si vede così che al variare della frequenza cambiano non solo le ampiezze a cui si sviluppano i vari fenomeni, ma che non sempre i fenomeni caotici sono presenti [2].

1.2.2 Circuito di Chua

Il circuito di Chua deriva dagli studi sul caos del prof. Leon O. Chua, docente dell'università della California, Berkeley, ed è l'unico circuito in cui la presenza del caos è stata provata in maniera analitica. Il vantaggio fondamentale del circuito di Chua è quello di essere un circuito autonomo, cioè di non aver bisogno di un segnale in ingresso.

Questo circuito, che fa parte della famiglia degli oscillatori caotici, presenta i tre requisiti minimi necessari per poter avere comportamenti caotici in circuiti autonomi:

• Dinamica almeno del terzo ordine, quindi almeno tre componenti dinamici indipendenti

• Almeno un componente non lineare

• Almeno un componente attivo

Queste sono condizioni necessarie ma non sufficienti perché un sistema possa generare un comportamento caotico inteso come comportamento aperiodico, duraturo nel tempo, delle traiettorie di un sistema deterministico.

In tal caso, a causa della dipendenza sensibile dalle condizioni iniziali, si possono avere traiettorie che non raggiungono punti di equilibrio e non si chiudono su cicli limite, ma continuano a muoversi nello spazio di stato presentando oscillazioni non periodiche non determinabili a priori .

In un circuito autonomo, come quello in esame, questo comportamento non è dovuto a fattori forzanti esterni ma è una proprietà intrinseca del sistema caotico.

Il circuito di Chua, in Figura 4, contiene tre elementi di accumulazione di energia che sono due condensatori passivi lineari e un induttore passivo lineare, poi un resistore passivo lineare e un resistore non lineare a due terminali detto anche "Diodo di Chua".

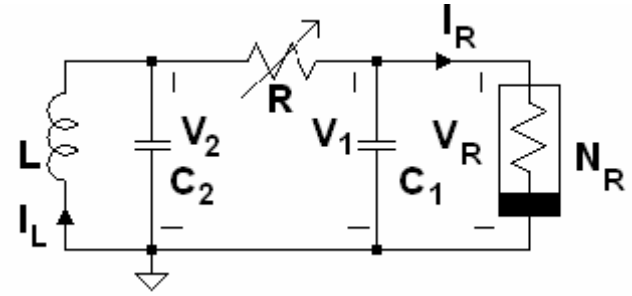


Figura 4-Circuito di Chua

Il sistema presenta diversi gradi di libertà, vale a dire che esiste la possibilità di agire sul valore delle due capacità, sul valore dell'induttore e sul valore del resistore variabile.
Si potrebbe pensare di variare il valore di una delle due capacità ma per esigenze costruttive queste variazioni non sono di grande praticità. Per quanto riguarda l'induttore è possibile utilizzare un induttore variabile (come si vedrà in seguito) ma, essendo vincolati nell'intorno di un determinato valore per ottenere le migliori condizioni di funzionamento del circuito, si utilizza come grado di libertà il valore del resistore variabile.
Dal momento che la resistenza R, l'induttanza L e le capacità C_1 e C_2 sono valori positivi, è chiaro che questo circuito per oscillare, e tanto più diventare caotico, dovrà presentare un resistore non lineare attivo, nel senso che la sua caratteristica tensione-corrente deve esibire regioni (secondo e quarto quadrante) in cui il prodotto $v \cdot i$ è negativo, quindi fornire energia agli elementi passivi.
Passiamo all'analisi circuitale facendo riferimento allo schema elettrico di Figura 5.

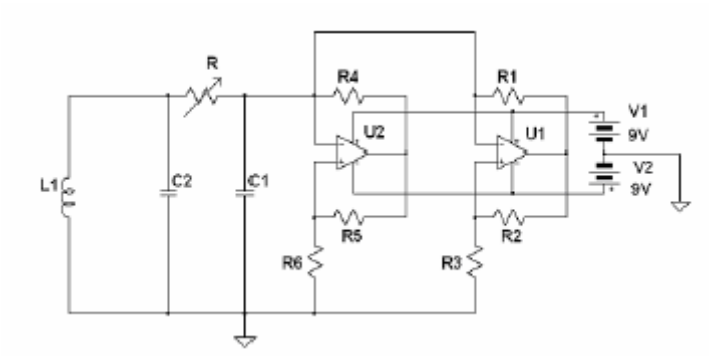


Figura 5– Schema elettrico del circuito di Chua

I condensatori utilizzati presentano due valori fissi pari a 10nF e 100nF rispettivamente identificati da C1 e C2. Il resistore lineare variabile è un trimmer dal valore 10kΩ. Gli altri componenti sono riportati in figura 6.

Elemento	Descrizione	Valore	Tolleranza
R1	Resistenza 1/4 W	220Ω	±5%
R2	Resistenza 1/4 W	220Ω	±5%
R3	Resistenza 1/4 W	2.2kΩ	±5%
R4	Resistenza 1/4 W	22kΩ	±5%
R5	Resistenza 1/4 W	22kΩ	±5%
R6	Resistenza 1/4 W	3.3kΩ	±5%
U1	OpAmp 1/2 TL082		
U2	OpAmp 1/2 TL082		
C1	Condensatore	10nF	±5%
C2	Condensatore	100nF	±5%

Figura 6- Valori componenti circuiti

La resistenza non lineare è un componente che presenta resistenza negativa e una non linearità.

Dato che si opera con valori di frequenza non superiori ai 24kHz, è stato inserito come elemento attivo un generico amplificatore operazionale (un TL08235), che connesso ad una opportuna rete di resistenze consente di giungere alla realizzazione del resistore non lineare, mostrato in figura 7.

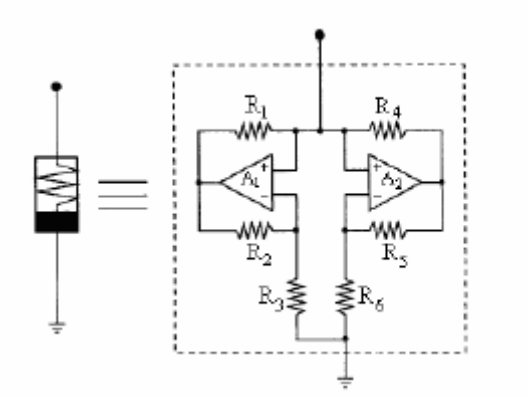


Figura 7- Resistore non lineare realizzato con TL082

Occupa un ruolo fondamentale nel circuito perché grazie ad esso si è in grado di mostrare la varietà di biforcazioni e di andamenti caotici.

L' induttore virtuale è realizzato con un amplificatore operazionale (TL082), una capacità e da un insieme di resistenze, opportunamente connesse, come si può osservare in Figura 8:

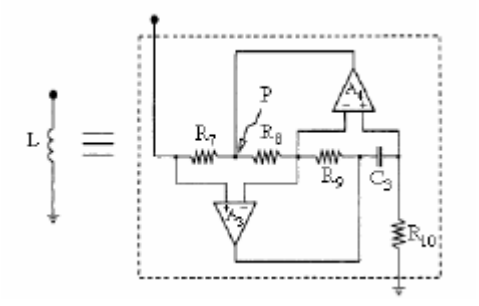


Figura 8-Il layout dell'induttore virtuale

Questo circuito simula il comportamento di un induttore ideale riferito rispetto alla massa; con l'induttore virtuale si riesce a migliorare il range di variabilità della resistenza lineare, infatti viene utilizzato per la regolazione digitale del circuito di Chua. La sua realizzazione è mostrata in figura 9.

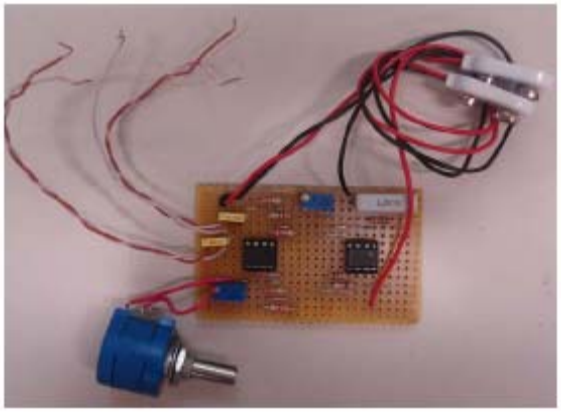


Figura 9- Circuito su piastra con induttore virtuale

Nella realizzazione pratica del potenziometro a controllo digitale, la configurazione adottata è la soluzione parallelo "reale", come mostrato in figura 10

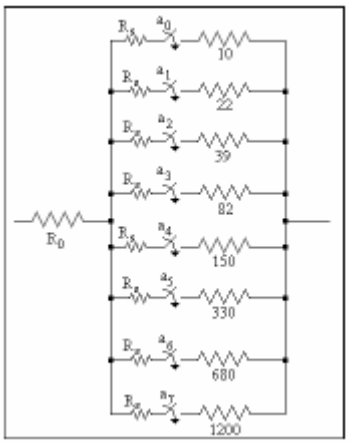


Figura 10- Rete resistiva realizzata

La rete consta di una sezione a valore costante (R0) seguita da una sezione variabile (soluzione parallelo "reale"). La prima sezione consente di avvicinarsi nella zona di interesse relativa al comportamento caotico mentre la seconda consente di visualizzare i vari modi di funzionamento al variare dei bit in ingresso e quindi derivare le migliori condizioni di funzionamento del circuito di Chua.

Il valore di Rs è 10Ω, 5Ω o 2.5Ω, in particolare con Rs = 2.5Ω si rilevano le migliori condizioni di funzionamento del circuito di Chua ed è quindi possibile evidenziare i vari modi di funzionamento dal ciclo stabile alle biforcazioni, dall'attrattore strano fino ad arrivare al caos completo. La realizzazione del circuito è mostrata in figura 11.

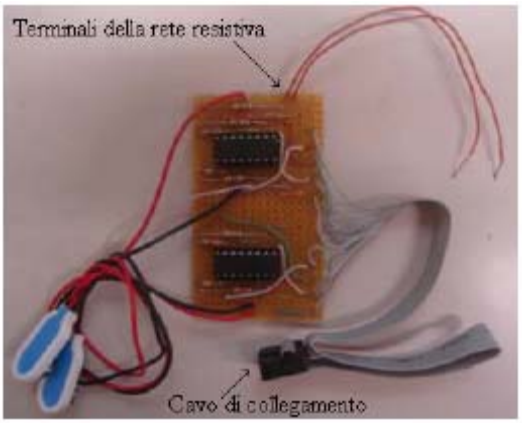


Figura 11-Potenziometro a controllo digitale

Per visualizzare i modi di funzionamento del circuito di Chua il potenziometro a controllo digitale è stato inserito nel circuito stesso al posto del resistore lineare variabile. Il potenziometro a controllo digitale presenta da un lato due terminali, che rappresentano le due estremità della rete resistiva, mentre dall'altro si trova un cavo che consente la connessione alla scheda di acquisizione.

L'interfaccia grafica, realizzata con LabView, permette di variare il valore resistivo in uscita dal potenziometro ed il relativo stato di funzionamento del circuito di Chua, collegando le due sonde ai capi dei condensatori C1 e C2 si è in grado di visualizzare con l'oscilloscopio i modi di funzionamento, dal ciclo stabile alle biforcazioni, dall'attrattore strano fino al caos [3].

1.3 Alcune interfacce per il collegamento

1.3.1 Interfaccia GPIB

Lo standard IEEE-488, conosciuto anche come GPIB (acronimo *Purpose Interface Bus*) o HI-IB (dal nome della ditta, l'Hewlett-Packard, che lo ha brevettato), consiste fondamentalmente di un bus ad alta velocità e di un certo numero di interfacce, a sé stanti o già integrate nei vari strumenti, che ne consentono l'allacciamento al bus. Lo standard prevede quindi un insieme di *segnalazioni* (sulle linee del bus di controllo), *messaggi* (byte di comando inviabili sul bus dati) e *processi* (da innescare all'interno delle interfacce) che permettono alle medesime d'intendersi nella fase di attivazione del collegamento e durante lo scambio di dati. Una volta che il collegamento è stabilito, lo scambio fisico di dati è gestito automaticamente dalle due interfacce coinvolte, in modo da costituire una struttura del tutto trasparente ai messaggi che gli strumenti così collegati si scambiano. Lo standard IEEE-488 ha posto le basi necessarie per superare lo scoglio dell'*incompatibilità tra prodotti di case diverse* semplificando l'allestimento di banchi di misura automatici. Inoltre esso ha rivelato doti di versatilità e completezza che ne hanno suggerito l'uso come generico bus di collegamento tra computer e periferiche, o anche per trasferimento di segnali tra personal computer. Da qui l'utilizzo dell' USB/GPIB nei laboratori, la quale consente di collegare PC portatili o fissi a strumentazione come Oscilloscopi, Generatori di segnale, ecc.[4].

1.3.1.1 USB/GPIB

Il modo più veloce e facile per collegare strumentazione elettrica ad un PC è di utilizzare un convertitore USB/GPIB, mostrato in figura 12 - un semplice cavo con un GPIB plug on ad una estremità ed un USB plug on all' altra - che consente una connessione diretta da una porta USB del PC ad uno strumento dotato di interfaccia GPIB, come in figura 13. A parità di prestazioni, costa meno delle corrispondenti schede interne [4].

Figura 12- USB/GPIB NI

Figura 13– Esempio di connessione con interfaccia USB e PC portatile

1.3.2 Interfaccia USB

L'USB (Universal Serial Bus) è uno standard di comunicazione seriale che consente di collegare al PC, per mezzo di un singolo cavo composto da una coppia di conduttori, una grande varietà di dispositivi.

Lo schema del connettore della porta di comunicazione USB è riportato in figura 14.

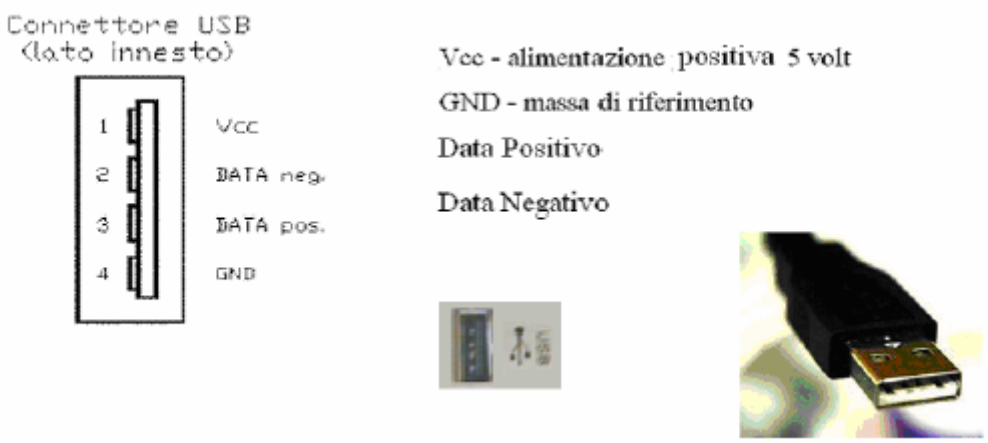


Figura 14- Schema del connettore della porta di comunicazione USB presente sul Pc.

Esistono tre versioni dello Standard USB: la versione 1.0, la quale supporta collegamenti a 1.5Mbit/sec, velocità adeguata per dispositivi lenti. La versione 1.1 aggiunge la modalità *full speed* che innalza la velocità a 12Mbit/sec. La versione 2.0, in cui la velocità di trasferimento arriva anche a 480Mbit/s. Oltre al trasferimento dei dati garantisce anche l' alimentazione per le periferiche: esso è, infatti, in grado di fornire una corrente di 500mA ad una tensione di 5V. Tale caratteristica può tranquillamente garantire l' alimentazione per piccole periferiche, che quindi possono fare a meno di batterie e di alimentatori esterni. Una periferica USB può essere collegata in qualsiasi momento, anche a computer acceso; in quanto il sistema operativo è in grado di riconoscere istantaneamente quando una nuova periferica viene collegata. E' comunque consigliato, prima di collegare il dispositivo per la prima volta al computer, installare i driver forniti con la periferica: dopo questa operazione la periferica è pronta per essere accesa e configurata. Presenta una flessibilità del protocollo per modalità sincrone nel trasferimento dei dati e asincrone per l' invio dei messaggi, supportandole sullo stesso cavo[4,5].

Sta sostituendo le uscite presenti “storicamente” sul PC, come porte parallele o seriali, e in particolare anche il Bus interno con un Bus esterno per periferiche di tipo digitale, mouse, tastiera, stampanti ed altre, perché come accennato consente la trasmissione dei dati a velocità elevata con prestazioni superiori rispetto alle porte seriali o parallele.

1.4 Scheda input/output analogica/digitale

Un altro componente molto importante all' interno di un laboratorio "remoto" è la scheda di acquisizione I/O A/D, in quanto consente l' interfacciamento tra PC e circuiti. Consideriamo quindi, più da vicino l' ambito specifico di questo elaborato di tesi: la realizzazione di una scheda di ingresso/uscita analogica/digitale.

1.4.1 Definizione di scheda input/output analogica/digitale

La scheda di acquisizione input/output analogica/digitale è un dispositivo hardware che consente la raccolta automatizzata di segnali analogici e digitali. Il suo compito è quello di trasferire nel modo più adatto e veloce i segnali esterni al PC, in modo tale da poterli elaborare e analizzare. In particolare, essa si adopera a "catturare" i segnali analogici e digitali uscenti da una qualsiasi sorgente e a salvarlo in un file riconoscibile e utilizzabile dal PC, preservandone naturalmente tutte le caratteristiche. Allo stesso modo questo dispositivo permette, nella maggior parte dei casi, il processo contrario. Inoltre,una scheda di acquisizione può presentare anche delle uscite controllate tramite degli appositi algoritmi [6]. Il suo impiego, nel laboratorio, è relativo alla possibilità di fornire al PC fisso o portatile i dati da esaminare, poiché si collega al personal computer tramite l'interfaccia USB, normalmente presente su tutti i personal computer.
Esempi di schede di acquisizione sono mostrate in figura 15:

Figura 15- Schede di acquisizione

1.4.2 Motivazioni della realizzazione di una scheda I/O A/D

Per realizzare l' interfacciamento tra il PC ed il potenziometro a controllo digitale del circuito di Chua, era stata impiegata la scheda di acquisizione NI 6040E della National Instruments, meglio conosciuta come la scheda NI PCI-MIO-16E-4 .

Tale scheda, tra le varie funzionalità, offre la possibilità di poter gestire 16 ingressi analogici, 2 uscite analogiche e 8 terminali digitali ingresso/uscita , questi ultimi adoperati per generare il segnale digitale in ingresso alla interfaccia digitale del circuito di Chua.

Data l' esigenza di ampliare il laboratorio "remoto" con altri esperimenti e l'impossibilità di questa scheda di poterli pilotare a causa del limitato numero di terminali digitali, si è deciso di realizzare una scheda di acquisizione ingresso/uscita analogica/digitale, esterna al PC e con interfaccia USB, in sostituzione ad essa, tale da poter gestire contemporaneamente, gli 8 terminali digitali ingresso/uscita, adoperati per generare il segnale digitale in ingresso alla interfaccia digitale del circuito di Chua e per controllare un multiplexer , tramite il quale sarà possibile selezionare il circuito RLD e il circuito di Chua, come mostrato in figura 16.

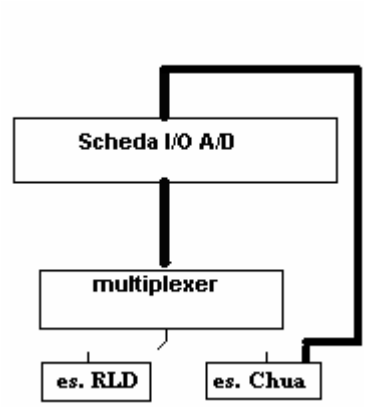


Figura 16- collegamento Scheda - Multiplexer

1.4.3 Descrizione del funzionamento della scheda I/O A/D

Le caratteristiche della **scheda I/O A/D** realizzata possono essere così riassunte:

- 16 bit digitali, configurabili come ingressi/uscite indipendentemente.
- 1 uscita analogica a 12 bit.
- 11 ingressi analogici a 12 bit.
- Possibilità di ulteriore espansione nel numero dei segnali aggiungendo ulteriori circuiti integrati a basso costo.
- Funzionamento "statico" della scheda a frequenze prossime allo zero, poiché non abbiamo l'esatto controllo delle tempificazioni della scheda e presenta un'interfaccia molto lenta, in quanto il protocollo SPI che simuliamo in software, richiede la trasmissione di un intero byte per ogni variazione di un suo qualsiasi segnale (per es. MISO, MOSI, ecc.).

1.4.4 Software di controllo della scheda

E' stato realizzato in LabView, si presenta come i driver che vengono forniti con le schede professionali, una serie di SubVi che implementano le funzionalità di configurazione e controllo della scheda. Sono riportate di seguito le funzioni (o primitive) che un utente hardware vedrebbe nell' utilizzare la scheda.

Il SubVI "**USBIO Init**" attiva e configura il dispositivo UM245R, inoltre effettua una configurazione generale del funzionamento interno del circuito integrato MCP23S17.

Il SubVI " **USBIO DigitalSetDir**" scrive nei registri IODIRA e IODIRB del Port Expander la configurazione dei pin rispettivamente come I/O.

Il SubVI "**USBIO DigitalWrite**" scrive nei registri OLATA e OLATB del Port Expander i dati in uscita dai rispettivi lati del dispositivo .

Il SubVI "**USBIO DigitalRead**" legge nei registri GPIOA e GPIOB del Port Expander i dati in ingresso sui rispettivi lati del dispositivo.

Il SubVI " **USBIO AnalogWrite**" scrive nel DAC.

Il SubVI "**USBIO AnalogRead**" legge dall' ADC e contemporaneamente lo configura per la successiva conversione.

Il SubVI " **USBIO Close**" chiude la comunicazione con un dispositivo precedentemente aperto.

L’ ambiente di sviluppo LabView presenta alcune peculiarità che lo differenziano notevolmente dai linguaggi procedurali più comunemente noti. La prima differenza sostanziale è l’ambiente di sviluppo grafico, sia dell’interfaccia utente sia dell’algoritmo di elaborazione. Altra differenza importante è che un programma LabView non segue il flusso delle istruzioni, ma il flusso di dati. Infatti i programmi scritti in LabView tendono ad essere di tipo data-driven, nel senso che si tende ad enfatizzare come i dati si muovono tra i diversi blocchi operativi più che la sequenza delle istruzioni da eseguire.

Capitolo 2

Realizzazione hardware di una scheda I/O A/D

2.1 Analisi dello stato dell' arte

Attualmente sul mercato sono disponibili diverse tipologie di schede di acquisizione aventi prestazioni, caratteristiche di funzionamento e costi differenti, secondo le proprie esigenze. Fra le varie schede di acquisizione esistenti in commercio, considereremo solo quelle attualmente in uso all' interno del laboratorio "remoto"di Circuiti per osservare vantaggi, svantaggi e prestazioni. In particolare la scheda **PCI-6040E**, alla quale è collegato il circuito di Chua, e la scheda **PCI-5102**, interna al computer, usata come front-end per realizzare l' oscilloscopio digitale.

La **PCI-6040E** [6,7], prodotta dalla National Instruments e mostrata in figura 17,

Family	Bus	Analog Inputs	Input Resolution	Max Sampling Rate	Input Range	Analog Outputs	Output Resolution	Output Rate	Output Range	Digital I/O	Counter/Timers	Triggers
NI 6040E	PCI, PXI	16 SE/8 DI	12 bits	500 kS/s	±0.05 to ±10 V	2	12 bits	1 MS/s	±10 V	8	2, 24-bit	Analog, digital

Figura 17- NI PCI-6040E

E' una scheda di acquisizione multifunzione, le cui prestazioni in termini di precisione e di velocità di acquisizione, risultano determinate dall' insieme calcolatore-scheda di acquisizione-periferica. In particolare, essa presenta 16 ingressi analogici Single Ended,

2 uscite analogiche, 8 ingressi/uscite digitali, una tensione nominale di ingresso da ±0.05V a ±10V. L' acquisizione può essere condotta su singolo canale per volta (single acquisition) o su più canali contemporaneamente (multipleacquisition). Nel caso di acquisizione multipla, è possibile stabilire quali sono i canali che interessano e l' ordine con cui devono essere acquisiti.

La massima frequenza di campionamento, nel caso di acquisizione singola, è di 500Ksample/sec (Sample: unità di misura che indica il numero di campioni), mentre nel caso di acquisizione multipla su N canali, è di (250/N) Ksample/sec per canale. Come si vede la frequenza massima del campionatore , scende a 250Ksample/sec nel caso di acquisizione multipla; la frequenza effettiva di campionamento per ogni canale è pari alla frequenza del campionatore diviso il numero di canali selezionati. Inoltre, è dotata di una memoria Buffer di tipo FiFo (First In First Out, cioè il primo elemento in ingresso è anche il primo elemento ad uscire) in cui vengono temporaneamente immagazzinati 1024 campioni. Il buffer allocato in memoria è di tipo circolare costituito da due puntatori, uno per la lettura ed uno per la scrittura. Per cui in operazioni di acquisizioni dati ad alta velocità, la scrittura avviene con un ritmo più veloce della lettura provocando, dopo un certo tempo, il riempimento del buffer . Inoltre secondo le specifiche lo Stream to disk, cioè quella modalità di acquisizione che dipendentemente dall' hardware del sistema che ospita la scheda, acquisisce e salva direttamente sulla memoria di massa, avviene senza saturazione per velocità di campionamento inferiori a 250Ksample/sec (system dependent).

Sebbene le prestazioni di questa scheda siano ottimali, presenta il problema del costo eccessivo e di un limitato numero di ingressi/uscite digitali. In questo momento i suoi 8 bit digitali sono completamente occupati per il controllo del circuito di Chua, di conseguenza il laboratorio resterebbe limitato ad un solo esperimento in remoto. Data l'esigenza di voler collegare anche il circuito RLD si è deciso di realizzare, in sostituzione una scheda d'acquisizione con un numero di bit i/o superiore, da qui la scelta di 16 bit facilmente configurabili come ingressi o uscite [7]. Passiamo ora a considerare l' altra scheda: la **PCI-5102** [6,8], prodotta dalla National Instruments e mostrata in figura 18,

Product	Bus	Channels	Sampling Rate	Bandwidth	Memory Total	Resolution
NI 5102	PCI, PXI, PCMCIA, USB, ISA	2	20 MS/s	15 MHz	663 kS to 16 MS	8 bits for PCI/PXI

Figura 18- NI PCI-5102

fa parte delle schede di acquisizione " High speed digitizers". In particolare, è in grado di acquisire ad una frequenza di campionamento massima pari a 20Mcampioni/sec con risoluzione a 8 bit da 2 canali ed una banda passante analogica di 15MHz, permettendo quindi di realizzare strumenti in grado di lavorare fino ad alcuni MHz senza grandi problemi. Inoltre, è dotata di una memoria Buffer di tipo FiFo, in cui vengono temporaneamente immagazzinati 663Kcampioni. Presenta una tensione nominale d' ingresso di $\pm$ 5V, è dotata di un front-end analogico di amplificazione a guadagno programmabile che permette di adattare la sensibilità per ogni canale tra un minimo di $\pm$ 50mV ad un massimo di $\pm$ 5 V. Con i driver della scheda NI vengono forniti una applicazione di test che realizza un semplice oscilloscopio ed una serie di programmi d'esempio sul funzionamento della scheda e sui passi di programma necessari per dialogarvi.

Sebbene il costo di questa scheda sia eccessivo, presenta il vantaggio di essere molto veloce e tramite i suoi driver è stato possibile realizzare il pannello frontale dell'oscilloscopio digitale[8].

2.2 Analisi degli integrati per la realizzazione della scheda I/O A/D e della scheda ausiliaria a Relé.

Per la realizzazione hardware della scheda I/O A/D e della scheda ausiliaria a Relé, mostrati in figura 19, sono stati utilizzati i seguenti componenti:

- Circuito di controllo e di interfaccia (UM245R).
- Port Expander (MCP23S17).
- DAC (TLV 5616)
- ADC (TLC 2543)
- Riferimento di tensione (TL 431).
- High-Voltage Darlington transistor array (ULN2003A).
- Relé (HM4101F)

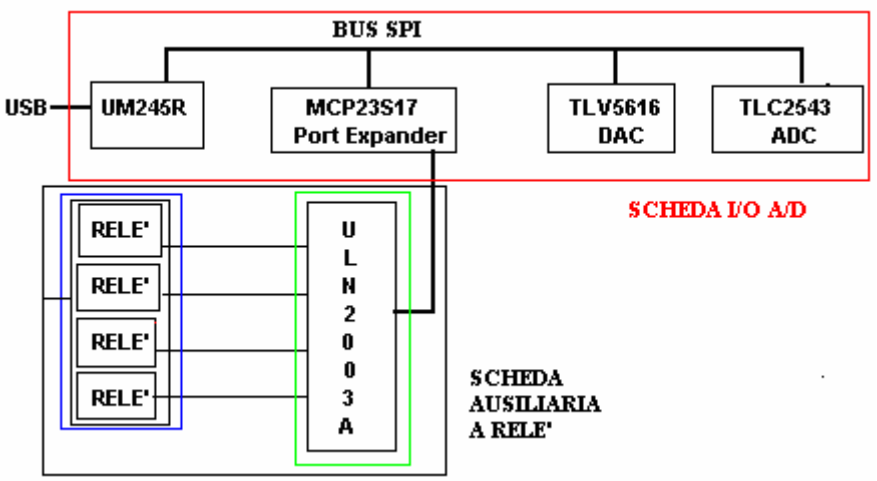


Figura 19- Scheda I/O A/D e Scheda ausiliaria di Relé

2.3 Il dispositivo UM245R

Il dispositivo UM245R, prodotto dalla FTDI e mostrato in figura 20, è l'evoluzione del circuito integrato FT245R, che rappresenta il più recente dei dispositivi a circuito integrato FTDI, con interfaccia USB UART [9].

Figura 20- Dispositivo UM245R

2.3.1 Descrizione generale

Il circuito integrato FT245R ha un' interfaccia parallela FIFO (First In First Out) dotata di connessione USB, con il nuovo sistema di protezione "FDTIChip-ID". Inoltre è disponibile un modo di interfacciamento che consente il "Bit Bang Mode", in modo sincrono e asincrono. Il "Bit Bang Mode" consiste in una particolare modalità di invio dei bit; in pratica i bit vengono trasmessi uno dietro l'altro, come se venissero sparati su un bus bidirezionale [9].

I dispositivi ad interfaccia parallela/USB usano l' FT245R con una versione per di più semplificata, che consta di una piena integrazione sul dispositivo della memoria esterna EEPROM, del circuito di clock e dei resistori USB. L'FT245R è dotato di una serie di nuove funzioni, rispetto al suo predecessore, ed è utilizzato per parecchie aree applicative. Durante la fabbricazione il dispositivo è coniato con un unico numero di identificazione, leggibile dalla USB, che può essere usato per proteggere l'applicazione software dell' utente.

L' UM245R è fornito su una PCB (Printed Circuit Board), cioè una basetta a circuito stampato, progettata per inserire il dispositivo nella presa standard USB, con un opportuno cavo, dotato di 24 pin, in modo tale da essere incastrato in uno zoccoletto.

2.3.2 Descrizione dei Pin

La descrizione dei Pin è indicata in figura 21, dove il dispositivo UM245R è visto dall' alto.

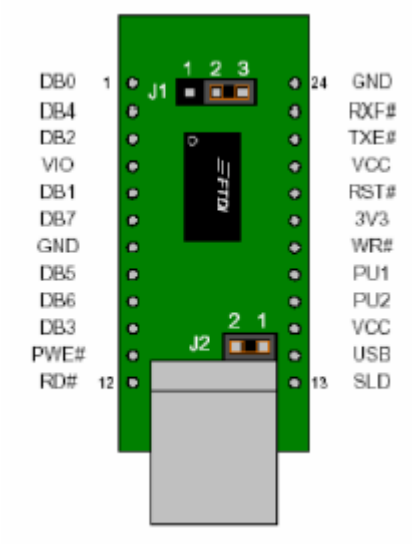


Figura 21- Descrizione dei Pin del dispositivo UM245R

Riguardo alla realizzazione della scheda I/O A/D, l' attenzione è rivolta ad un ristretto insieme di terminali:

Vcc- Tensione di alimentazione
Gnd- Massa
DBx- Generico bit di input/output

Il vantaggio principale di questo integrato consiste nella possibilità di configurarlo in modo tale da essere alimentato tramite la porta USB; in tale configurazione non è necessaria un' alimentazione esterna e quindi l' utente deve solo collegare il cavo USB, semplificando al massimo l' operazione di collegamento.
L' attenzione è rivolta essenzialmente agli otto bit DB0-DB7, che nella realizzazione della scheda I/O A/D , devono essere configurati opportunamente per il pilotaggio dei vari integrati.

2.3.3 Driver e Librerie

I produttori dei dispositivi dotati di interfaccia USB mettono a disposizione sia le necessarie librerie software, che consentono di adoperare agevolmente questi apparecchi, sia i driver per il sistema host, che nel nostro caso rappresenta il personal computer [11].

Visitando il sito del produttore UM245R[10], è possibile scaricare le librerie di gestione, scritte in LabView, ed il relativo driver per avere la possibilità di eseguirle.

Il driver scaricato è il file FTD2XX.dll, che va inserito nelle cartelle dove si trovano le librerie.

Nelle librerie sono presenti diversi tipi di funzioni che consentono di eseguire svariate operazioni sul dispositivo, che vanno dalle più semplici tipo azioni di apertura, lettura, scrittura a quelle più complesse di configurazione.

Per la costruzione delle SubVi, che vedremo più nel dettaglio nel capitolo 3, utilizzo le seguenti funzioni DLL:

La funzione **FT_Open** attiva il dispositivo e ritorna il valore di un puntatore denominato Handle, che è utilizzato per un accesso successivo al dispositivo.

La funzione **FT_SetBitMode** riceve in ingresso l' Handle generato dalla FT_Open e consente di settare i bit DB0-DB7 come bit di input o di output, attraverso il valore di una maschera "Bit Mode Mask" richiesto in ingresso. Se il valore del Bit Mask del corrispondente Pin vale 0 allora viene settato come input, se il valore del Bit Mask è 1 il corrispondente Pin è settato come output.

E' richiesto inoltre il valore di un altro parametro di ingresso detto "Mode", che determina il modo di interfacciamento del dispositivo che può risultare un interfacciamento parallelo, oppure del tipo "Bit Bang Mode", sincrono o asincrono.

Per la regolazione digitale l' UM245R è configurato con un interfacciamento parallelo.

La funzione **FT_Write** scrive i dati sul dispositivo. Analogamente alla funzione precedente, riceve in ingresso l' Handle, generato dalla FT_Open, un array di bit, analogo alla maschera "Bit Mode Mask", in cui è possibile selezionare i dati da scrivere, ed infine il parametro ottenuto dal modulo "size array", che indica il numero di byte che vengono scritti.

La funzione **FT_GetBitMode** consente la lettura degli "attuali" valori dei pin, in quanto restituisce un singolo byte contenente il valore corrente dei pin, sia per i pin di ingresso sia per quelli di uscita. Richiede in ingresso l'Handle, generato dalla FT_Open.

La funzione **FT_Close** serve a chiudere il dispositivo. Necessita in ingresso del parametro rappresentativo del Handle, generato dalla FT_Open.
In uscita da ogni funzione è presente una sorta di parametro di monitoraggio, che consente di valutare lo stato del modulo, detto FT_Status.
In base al valore della FT_Status è possibile capire se ci sono degli errori o delle anomalie di funzionamento. Di seguito vengono riportati alcuni valori di questo parametro:

FT_OK=0
FT_INVALID_HANDLE=1
FT_DEVICE_NOT_FOUND=2
FT_DEVICE_NOT_OPENED=3
FT_IO_ERROR=4

Lavorando in LabView è possibile generare, con queste funzioni, dei SubVi, che concatenati tra di loro, forniscono opportuni VI per configurare gli 8 bit, DB7-DB0, dell' UM245R come bit di i/o, in modo tale da poter pilotare adeguatamente i dispositivi che costituiscono la scheda.

2.4 L' integrato MCP23S17 (Port Expander I/O)

L' integrato MCP23S17, prodotto dalla Microchip, è un "Port Expander" bidirezionale a 16-bit I/O, con interfaccia seriale (SPI) [12]. Questi 16-bit sono divisi in due gruppi da 8-bit associati rispettivamente a due registri corrispondenti a due porte: PortA e PortB; in modo tale da poter configurare l' integrato in modalità 8-bit oppure 16-bit; dei quali, inoltre, è possibile configurare singolarmente i bit come ingressi oppure uscite. Il protocollo SPI consente al sistema Master, l' UM245R, la configurazione dei registri del Port Expander, semplicemente scrivendo in essi; per esempio scrivendo nel registro IOCON.BANK si può configurare il dispositivo in modo tale da poter operare in modalità 8-bit oppure 16-bit; tramite i registri IODIRA/B si effettua la configurazione dei bit I/O ecc. Inoltre il Master ha la possibilità di leggere in tutti i registri del "port expander", ottenendo, in tal modo, l'estensione da 8-bit a 16-bit. Tra i vari registri che lo caratterizzano, abbiamo utilizzato i seguenti:

Per la configurazione della direzione dei Pin

- IODIRA
- IODIRB

Per la configurazione generale del funzionamento interno dell' integrato MCP23S17

- IOCON(A)
- IOCON(B)

Per la lettura

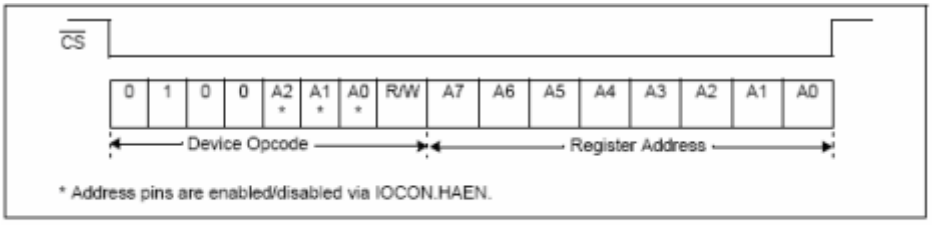
- GPIOA
- GPIOB

Per la scrittura

- OLATA
- OLATB

Inoltre, per quanto riguarda il trasferimento dei dati abbiamo scelto, nel nostro caso la modalità sequenziale (ottenuta ponendo il valore IOCON.SEQOP=0), nella quale viene incrementato automaticamente il "Puntatore di indirizzo" dopo ogni byte di dati trasmessi, fin quando non giunge all' ultimo registro, dopo il quale ritorna a puntare il registro iniziale. Mentre l' operazione di scrittura e lettura, che come già accennato avviene in base al protocollo SPI e la cui realizzazione software è mostrata nel capitolo

3, viene eseguita nel seguente modo: il comando di scrittura, inviato dal Master all' integrato, contiene lo "slave address" (costituito da 4 bit fissi e 3 bit da stabilire (A1-A2-A3) che definiscono l' indirizzo hardware del dispositivo) ed il bit che indica la tipologia di comando R/W=0 (0 indica la scrittura), come mostrato in figura 23 (in cui costituiscono il *byte di controllo* in quanto MCP23S17 è un SPI slave); questo comando costituisce l' opcode al quale devono essere aggiunti l' indirizzo del registro, sul quale si vuole operare ed almeno un byte di dati, come mostrato in figura 22. Analogamente si procede per il comando di lettura, in cui bisogna soltanto cambiare il bit che indica la tipologia di comando: R/W=1 (1 indica la lettura). Nel nostro caso, utilizziamo una modalità sequenziale, per cui parliamo di operazioni sequenziali di lettura e scrittura, nelle quali il Master trasmette il successivo byte puntato dal Puntatore di indirizzo fin quando la sequenza non termina, ovvero fin quando il Master non pone il CS ad un livello logico alto, nel qual caso il Puntatore, dopo aver puntato l' ultimo registro, ritorna all' indirizzo 0.



bit 7-0 **OL7:OL0:** These bits reflect the logic level on the output latch <7:0>
1 = Logic-high.
0 = Logic-low.

Figura 22- SPI: tipologia di comando

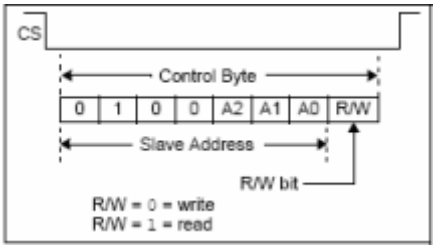


Figura 23- SPI formato: Byte di controllo

Passiamo ora alla descrizione dei registri del Port Expander, i quali sono raggruppati in due tabelle, le quali differiscono per la configurazione del bit BANK, appartenente al registro IOCON. In particolare,

- Se Bank=1 , come in tabella 1, avremo da un lato l'insieme dei registri associati alla porta A, e dall' altro l' insieme dei registri associati alla porta B.
- Se Bank=0, come mostrato in tabella 2, i registri A/B sono accoppiati.

Tabella 1- Registri di controllo (IOCON.BANK=1)

Register Name	Address (hex)	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	POR/RST value
IODIRA	00	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IOCON	05	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—	0000 0000
GPIOA	09	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
OLATA	0A	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000
IODIRB	10	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IOCON	15	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—	0000 0000
GPIOB	19	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
OLATB	1A	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000

Tabella 2- Registri di controllo (IOCON.BANK=0)

Register Name	Address (hex)	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	POR/RST value
IODIRA	00	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IODIRB	01	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IOCON	0A	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—	0000 0000
IOCON	0B	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—	0000 0000
GPIOA	12	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
GPIOB	13	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
OLATA	14	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000
OLATB	15	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000

Di queste due tipologie, la prima è vantaggiosa nel caso in cui si opera con uno solo dei registri (A oppure B); mentre la seconda, se si opera con entrambi i registri (A e B), in particolare per le applicazioni dove si deve ottimizzare al massimo la velocità di trasmissione. Nel nostro caso utilizzeremo il bit Bank=0.

Il *Registro di direzione I/O (I/O Direction Register)*: consente il controllo della direzione dei dati i/o, tramite la configurazione degli 8-bit come ingressi oppure uscite, come riportato in figura 24.

Figura 24- Registro IODIR-I/O Direction Register (ADDR 0x00)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0 **IO7:IO0:** These bits control the direction of data I/O <7:0>
1 = Pin is configured as an input.
0 = Pin is configured as an output.

In particolare, quando un bit è settato a 1, il pin corrispondente è configurato come ingresso, quando un bit è 0, il pin corrispondente è configurato come uscita.

Il *Registro di configurazione (IOCON)*: contiene i bit per la configurazione generale dell' integrato, riportati in figura 25.

Figura 25-Registro IOCON- I/O EXPANDER CONFIGURATION REGISTER (ADDR 0x05)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **BANK:** Controls how the registers are addressed
1 = The registers associated with each port are separated into different banks
0 = The registers are in the same bank (addresses are sequential)

bit 6 **MIRROR:** INT Pins Mirror bit
1 = The INT pins are internally connected
0 = The INT pins are not connected. INTA is associated with PortA and INTB is associated with PortB

bit 5 **SEQOP:** Sequential Operation mode bit.
1 = Sequential operation disabled, address pointer does not increment.
0 = Sequential operation enabled, address pointer increments.

bit 4 **DISSLW:** Slew Rate control bit for SDA output.
1 = Slew rate disabled.
0 = Slew rate enabled.

bit 3 **HAEN:** Hardware Address Enable bit (MCP23S17 only).
Address pins are always enabled on MCP23017.
1 = Enables the MCP23S17 address pins.
0 = Disables the MCP23S17 address pins.

bit 2 **ODR:** This bit configures the INT pin as an open-drain output.
1 = Open-drain output (overrides the INTPOL bit).
0 = Active driver output (INTPOL bit sets the polarity).

bit 1 **INTPOL:** This bit sets the polarity of the INT output pin.
1 = Active-high.
0 = Active-low.

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

In particolare, consente di controllare come i registri sono indirizzati (ovvero quale delle due tabelle, precedentemente mostrate, voglio considerare); si può decidere di connettere oppure no i pin INTA/B; consente di abilitare la modalità sequenziale oppure no; si può abilitare un rate maggiore; si può decidere di abilitare oppure no i pin relativi all' indirizzo hardware; è possibile configurare il pin INT come uscita di un open-drain ed infine si può settare la polarità del pin di uscita INT.

Il *registro GPIO*: riflette il valore sulla porta, nel senso che leggendo da questo registro leggiamo il valore sulla porta e scrivendo su questo registro viene modificato il registro di memoria in uscita (OLAT). La modalità di configurazione dei bit è riportata in figura 26.

Figura 26- GPIO – GENERAL PURPOSE I/O PORT REGISTER (ADDR 0x09)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0 **GP7:GP0:** These bits reflect the logic level on the pins <7:0>
1 = Logic-high.
0 = Logic-low.

In particolare, questi bit riportano il livello logico sulla pota, che può essere basso oppure alto.

Il *"Registro di memoria in uscita" (OLAT)*: fornisce l' accesso ai terminali di uscita, nel senso che una lettura da questo registro corrisponde ad una lettura dall' OLAT e non della Porta stessa. La modalità di configurazione dei bit è riportata in figura 27.

Figura 27- Registro OLAT-OUTPUT LATCHES REGISTER (0x0A)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-0 **OL7:OL0:** These bits reflect the logic level on the output latch <7:0>
1 = Logic-high.
0 = Logic-low.

In particolare, questi bit riportano il livello logico sui terminali di uscita, che può essere alto oppure basso.

Terminata la configurazione dei registri, per la realizzazione della lettura/scrittura nell' integrato, bisogna tener conto delle tempificazioni del protocollo seriale del Port Expander: la scrittura seriale dei dati nel dispositivo, come mostrato in figura 28, inizia portando il CS ad un livello logico basso, dopo di che i dati saranno trasferiti nell' integrato in corrispondenza dei fronti di salita del SCK per 8 impulsi di clock. In tal caso, il passaggio del singolo bit lo si ottiene in tre scritture: nella prima scrittura prepariamo il dato, nella seconda facciamo passare il clock da 0 a 1 (fronte di salita), nella terza facciamo passare il clock da 1 a 0 (fronte di discesa). Dopo gli 8 impulsi di clock il CS è riportato al livello logico alto. C' è da osservare, inoltre, che durante la scrittura dei dati nell' integrato il terminale SO (dei dati in uscita) è ad alta impedenza, cioè inattivo.

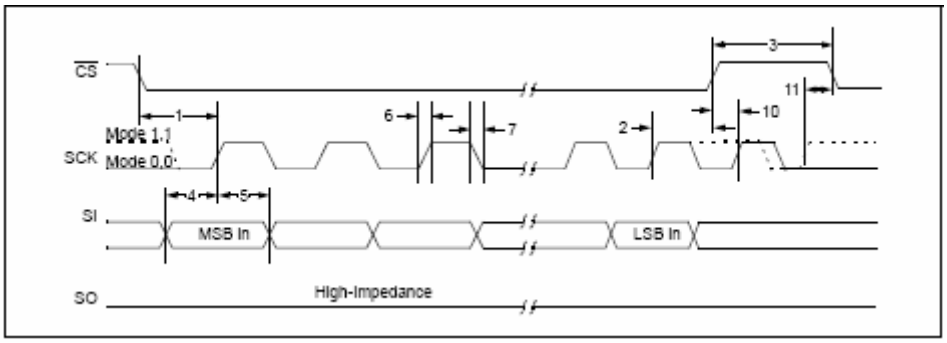


Figura 28- Tempificazoni del protocollo seriale del Port Expander per la scrittura

Nella fase di lettura, considerando che precedentemente c'è stata la scrittura di ciò che vogliamo leggere avremo,come mostrato in figura 29, il CS ad un livello logico basso, dopo di che i dati saranno letti in corrispondenza dei fronti di salita del SCK per 8 impulsi di clock. In tal caso, il passaggio del singolo bit lo si ottiene in tre fasi: nella prima fase ci prepariamo alla lettura, nella seconda fase portiamo il clock da 0 a 1 (fronte di salita), nella terza fase portiamo il clock da 1 a 0 (fronte di discesa). Durante la lettura può avvenire anche il trasferimento dei dati, il terminale SI (dei dati in ingresso) può trovarsi in un qualsiasi stato, non influenza la lettura. Terminata la lettura il CS va riportato ad un livello logico alto.

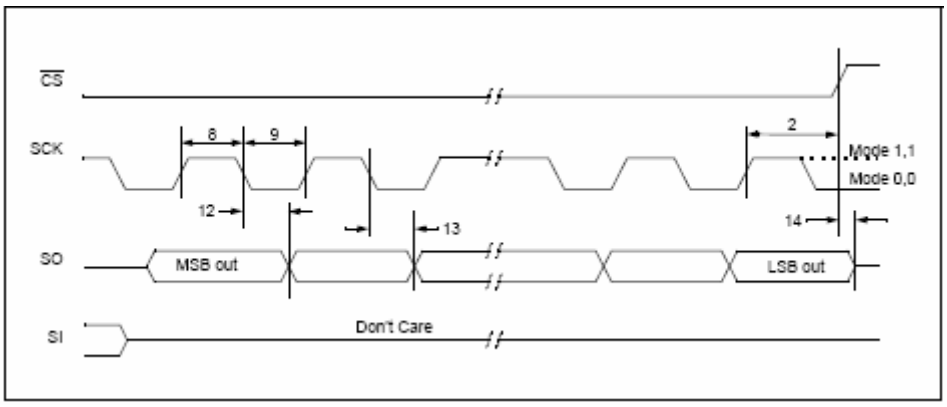


Figura 29- Tempificazioni del protocollo seriale del port expander per la lettura

Effettuiamo, ora, una breve descrizione esterna dell' integrato, in particolare dei terminali che lo caratterizzano, mostrati in figura 30.

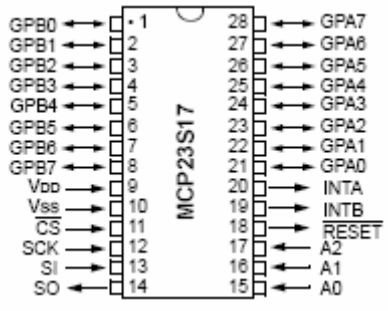


Figura 30- Descrizione dei Pin

GPBx- Pin I/O bidirezionali

Vdd- Tensione di alimentazione

Vss- Massa

CS- Chip Select

SCK- Serial Clock Input

SI- Dati in ingresso seriali

SO- Dati in uscita serali

Ax- Pin di indirizzo hardware

RESET- Reset hardware (è attivo quando è posto ad un livello logico basso)

GPAx- Pin I/O bidirezionali

Possiamo osservare che per il protocollo SPI, il master ha la possibilità di estendere i suoi 8-bit i/o ai 16-bit i/o del Port Expander, favorendo in tal modo il pilotaggio di un maggior numero di dispositivi.

2.5 L’ integrato TLV5616 (DAC)

IL DAC (Digital - to - Analog Converter), Convertitore Digitale-Analogico, è un componente elettronico in grado di produrre una determinata differenza di potenziale in funzione di un valore numerico che viene caricato [4].

Il DAC utilizzato nella realizzazione della scheda è il TLV5616, appartenente alla famiglia dei TLV5616X, dove X indica i diversi integrati. Il TLV5616 è alimentato nel range (2.7-5.5) V, assorbe poca potenza, presenta una bassa non linearità, è un DAC a 12-bit con una interfaccia SPI (Serial Peripheral Interface)[4]. Inoltre, è a tecnologia stringa seriale a 16-bit, contenente 4 bit di controllo e 12 bit per i dati, con uscita di tensione perché così progettati. Presenta, un amplificatore di uscita, in grado di generare tensioni alla sua uscita molto prossime alla tensione di alimentazione (nominalmente 5V); con guadagno pari a due. Un amplificatore ad alta impedenza, integrato sul terminale di REFIN, come mostrato in figura 31, per eliminare la necessità di avere una tensione di riferimento a bassa impedenza in uscita. La codifica di questo integrato è binaria diretta e la tensione in uscita è data dalla seguente relazione: 2 x Vref x (CODE/2^n) Volt; in cui Vref è la tensione di riferimento, Code è il valore in ingresso digitale, che può variare da 0 a 2^(n-1), in binario [13].

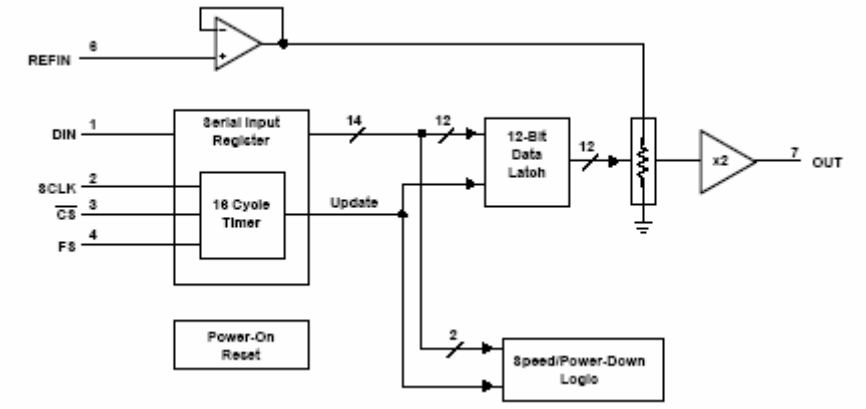


Figura 31- Diagramma a blocchi

Questo integrato è dotato di 5 pin, la cui descrizione è indicata in figura 32, dove il circuito integrato è visto dall’ alto.

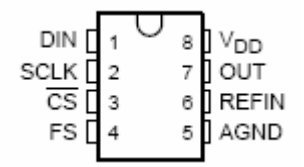


Figura 32- TLV5616: Descrizione dei pin del DAC

DIN- Dati in ingresso digitali e seriali

SCLK- Serial digital clock input

CS- Chip Select

FS- Frame sync.(per sincronizzare l' invio della frame)

Vdd- Tensione di alimentazione

OUT- Uscita analogica del DAC

REFIN- Tensione di riferimento analogica in ingresso

AGND- Massa

Per la realizzazione della scrittura nell' integrato, bisogna tener conto delle tempificazioni del protocollo seriale del DAC, mostrate in figura 33: per iniziare il trasferimento seriale dei dati, FS dovrebbe essere messo basso, dopo di che i dati saranno spostati nell' integrato in corrispondenza dei fronti di discesa del SCLK per 16 impulsi di clock. Il passaggio del singolo bit lo si ottiene in tre scritture: nella prima si effettua la scrittura nel dispositivo, nella seconda l' impulso di SCLK passa da 0 a 1, nella terza l' impulso di SCLK passa da 1 a 0. Dopo il sedicesimo fronte di discesa del SCLK, terminata la scrittura, FS è riportato al livello logico alto. Il CS non è stato commentato in quanto, nella nostra realizzazione, è stato collegato a massa, ovvero è stato posto direttamente ad un livello logico basso; assumendo, quindi, FS per l' attivazione della scrittura.

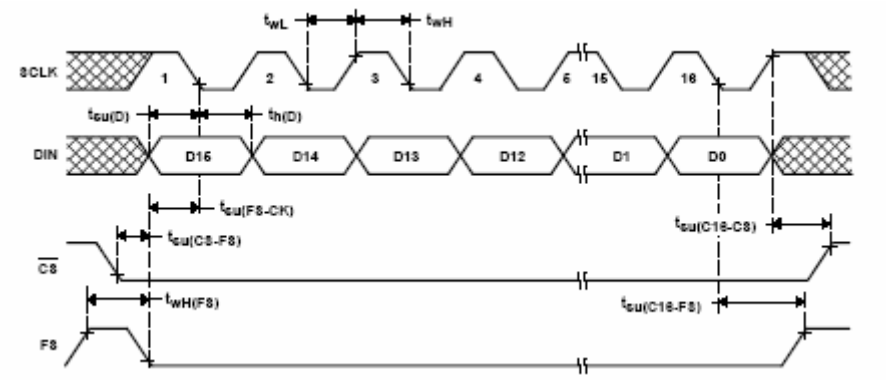


Figura 33- Tempificazioni del protocollo seriale del DAC

Il TLV5616 riceve in ingresso una word, 16-bit, mostrata in figura 34, costituita da:

- Bit di controllo (D15-D12)
- Valore nuovo del DAC (D11-D0)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	SPD	PWR	X	New DAC value (12 bits)											

X: don't care
SPD: Speed control bit. 1 → fast mode 0 → slow mode
PWR: Power control bit. 1 → power down 0 → normal operation

Figura 34- Formato dei dati

Nel nostro caso la word inviata è configurata con i bit di controllo posti a zero, in quanto il dispositivo funziona in condizioni normali.

2.6 Il dispositivo TLC2543 (ADC)

L' ADC (Analog to Digital Converter), convertitore analogico-digitale, è un circuito elettronico in grado di convertire una grandezza continua, ad es. una tensione, in una serie di valori discreti [4]. L'ADC utilizzato nella realizzazione della scheda è il TLC2543, appartenente alla famiglia dei TLC2543X, dove X indica i diversi integrati. Il TLC2543, prodotti dalla Texas Instruments, è caratterizzato da condensatori-commutatori con risoluzione a 12-bit e con interfaccia SPI, che effettuano conversioni da analogico a digitale per approssimazioni successive. Ogni strumento, con tre ingressi di controllo (Chip Select, I/O Clock, Data Input), come mostrato in figura 35, è progettato per comunicare con la porta seriale di un processore host o periferico attraverso un' uscita seriale a 3-stati. Il dispositivo permette trasferimenti dei dati dall' host ad alta velocità. In aggiunta al convertitore ad alta velocità e alla capacità di controllo versatile, il dispositivo ha nel chip un multiplexer a 14-canali, che gli consente di selezionare uno degli ingressi o una delle tre tensioni all' interno del test-self. La funzione di Sample-and-hold è automatica, per cui alla fine della conversione, l' uscita EOC va al livello logico alto, per indicare che la conversione è completa. Presenta il vantaggio di permette una conversione a basso errore [14].

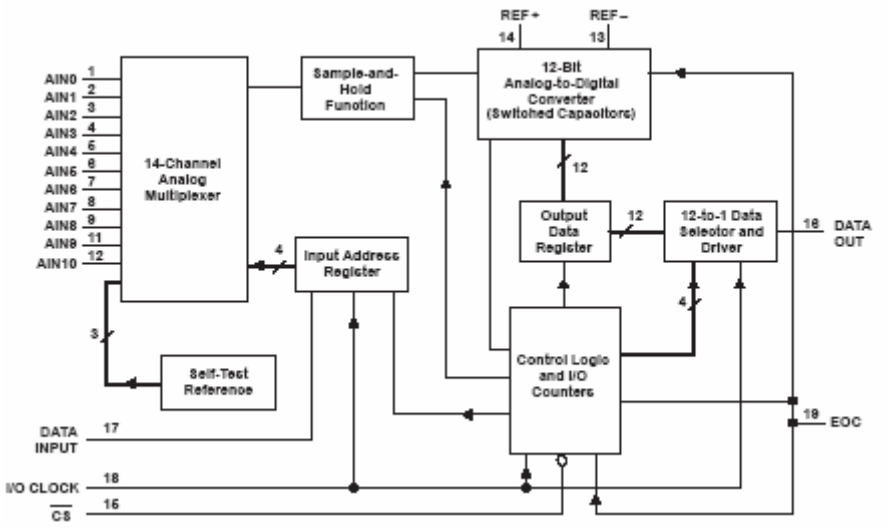


Figura 35-Diagramma a blocchi

Questo dispositivo è dotato di 20 pin, la cui descrizione è indicata in figura 36, dove il dispositivo è visto dall' alto.

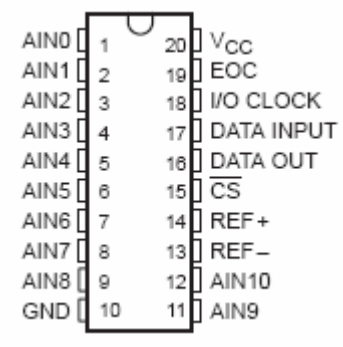


Figura 36- Descrizione dei Pin

AINx- Ingressi analogici, i quali vengono "multiplexati" all' interno.

GND- Massa

Vcc- Tensione di alimentazione

EOC- Fine conversione

I/O CLOCK- Input/output clock

DATA INPUT- Dati in ingresso seriali

DATA OUT- Dati in uscita seriali

CS – Chip Select

REF+- Tensione di riferimento positiva

REF- -Tensione di riferimento negativa

Per effettuare la lettura e contemporaneamente la configurazione del dispositivo per la successiva conversione, bisogna tener conto delle tempificazioni del protocollo seriale dell' ADC, mostrato in figura 37:

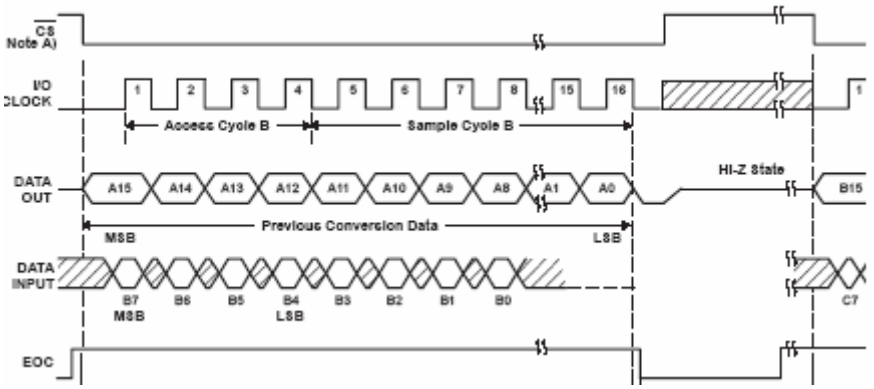


Figura 37-Tempificazioni del protocollo seriale dell' ADC

Inizialmente con il CS alto, I/O Clock e DATA INPUT sono disabilitati e DATA OUT è nello stato di alta-impedenza, cioè inattivo. Quando il CS viene portato al livello logico basso, ha inizio la sequenza di conversione. Vengono abilitati I/O CLOCK e DATA INPUT e viene rimosso il DATA OUT dallo stato di alta impedenza.
In particolare, il DATA INPUT riceve i dati in ingresso, come mostrato nella Tabella 1, ovvero un flusso di 8-bit: costituiti da 4-bit contenenti l' indirizzo del canale analogico (D7-D4), 2-bit per selezionare la lunghezza dei dati (D2-D3), un bit che indica se il primo bit in uscita è MSB (bit più significativo) oppure LSB (bit meno significativo) (D1) e un bit per selezionare la modalità unipolare o bipolare (D0) (nel nostro caso scegliamo la seguente sequenza xxxx0011, in cui le x possono variare mentre i bit 0011 restano fissi).

FUNCTION SELECT	INPUT DATA BYTE							
	ADDRESS BITS				L1	L0	LSBF	BIP
	D7 (MSB)	D6	D5	D4	D3	D2	D1	D0 (LSB)
Select input channel								
AIN0	0	0	0	0				
AIN1	0	0	0	1				
AIN2	0	0	1	0				
AIN3	0	0	1	1				
AIN4	0	1	0	0				
AIN5	0	1	0	1				
AIN6	0	1	1	0				
AIN7	0	1	1	1				
AIN8	1	0	0	0				
AIN9	1	0	0	1				
AIN10	1	0	1	0				
Select test voltage								
($V_{ref+} - V_{ref-}$)/2	1	0	1	1				
V_{ref-}	1	1	0	0				
V_{ref+}	1	1	0	1				
Software power down	1	1	1	0				
Output data length								
8 bits					0	1		
12 bits					X	0		
16 bits					1	1		
Output data format								
MSB first							0	
LSB first (LSBF)							1	
Unipolar (binary)								0
Bipolar (BIP) 2s complement								1

Tabella 1-Formato dei registri in ingresso

La sequenza I/O CLOCK applicata al terminale I/O CLOCK, trasferisce questi dati al registro dei dati in ingresso. Durante questo trasferimento, la sequenza I/O CLOCK deve, anche spostare il risultato della conversione precedente dal registro dei dati in uscita al terminale DATA OUT. Per cui l' I/O CLOCK riceve, per come abbiamo configurato il registro dei dati in ingresso, una sequenza di dati lunga 16 cicli di clock. Il prelievo dall' ingresso analogico ha inizio sul quarto fronte di discesa della sequenza di I/O CLOCK, ed avviene fino all' ultimo fronte di discesa della sequenza di I/O

CLOCK. L' ultimo fronte di discesa della sequenza di I/O CLOCK, porta anche EOC basso e comincia la conversione.
L' operazione di conversione avviene con una successione di due cicli distinti:

- *Il ciclo di I/O*
- *Il ciclo di conversione attuale*

Il *ciclo di I/O* è definito dall' I/O CLOCK fornito esternamente, dal periodo di clock selezionato e dalla lunghezza dei dati in uscita selezionati. Durante il ciclo di I/O, viene fornito al DATA INPUT il flusso di 16-bit di informazione sull' indirizzo e di controllo; questi dati, in pratica, vengono spostati nell' integrato sul primo fronte di salita dei 16 clock del I/O CLOCK. Durante il trasferimento di questi 16-bit, il DATA INPUT viene ignorato.
Per quanto riguarda i dati in uscita, 16-bit, sono forniti in maniera seriale sul DATA OUT.
Il primo bit dei dati in uscita si verifica sul fronte di salita del EOC e sul fronte di discesa del CS. Questi dati sono il risultato di un periodo di conversione precedente.
Dopo il primo bit dei dati in uscita, ogni bit successivo è trasmesso fuori sul fronte di discesa di ogni I/O CLOCK successivo.

Passiamo ora ad analizzare il successivo ciclo, che caratterizza la conversione: *il ciclo di conversione*. Esso è trasparente all' utilizzatore ed è controllato tramite un clock interno sincronizzato al I/O CLOCK. Durante il periodo di conversione, l' integrato compie conversioni per approssimazioni-successive sulla tensione in ingresso analogica.
L' uscita EOC è portato ad un livello logico basso, all' inizio del ciclo di conversione ed è portata ad un livello logico alto, quando la conversione è terminata e il registro dei dati in uscita ha memorizzato i dati.
Un ciclo di conversione ha inizio solo dopo che il CICLO I/O è completo, il quale limita l' influenza del rumore digitale esterno sulla accuratezza della conversione.

2.7 L’ integrato TL431

L’ integrato TL431 consente di avere ai suoi capi una tensione fissa e abbastanza precisa per il DAC e l’ADC. Esso è costituito da tre terminali “adjustable shunt regulators”, come rappresentati in figura 38 e 39 [15].

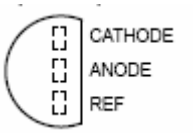


Figura 35– TL431

Figura 38- rappresentazione circuitale

Si tratta di un dispositivo la cui tensione di uscita può essere settata a qualsiasi valore, tra Vref (approssimativamente è 2.5 V) e 36 V, con due resistenze esterne. In cui, per avere il valore minimo della tensione, bisogna eliminare R2 e mettere R1 a zero, rappresentate in figura 39. Partendo da questa condizione, per ottenere 2.048 V, tensione di riferimento per il DAC, bisogna aggiungere un ulteriore partitore di tensione formato da un trimmer da 2KΩ e da una resistenza di 10KΩ. La presenza del trimmer consente di avere un valore preciso della tensione di riferimento (a 2.048V).

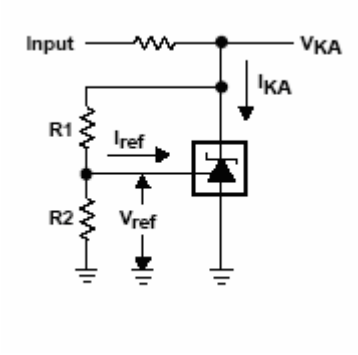


Figura 39– TL431 con R1 ed R2

Lo schema da considerare è riportato in figura 40.

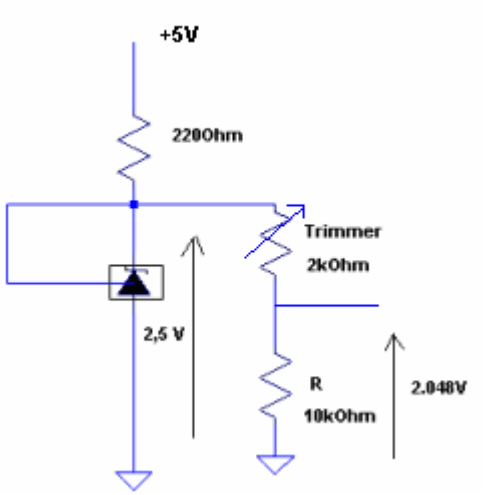


Figura 40- TL431 con il trimmer ed R

2.8 L' integrato ULN2003A

L' integrato ULN2003A consente di avere in uscita una tensione e una corrente elevata (rispetto agli standard delle porte logiche), grazie all' utilizzo di un array di transistori Darlington, come mostrato in figura 41; per questo motivo viene definito vettore di transistori Darlington ad alta-tensione e ad alta-corrente. Esso è costituito da sette coppie npn Darlington, che assicurano uscite ad alta-tensione tramite l'utilizzo di diodi di ricircolo con catodo-comune per commutare il carico induttivo. In cui la corrente in un collettore di una singola coppia Darlington è 500 mA. Inoltre, le coppie Darlington possono essere poste in parallelo nel caso di potenzialità di corrente elevata. Ha una serie di resistori di base a 2.7kΩ per ogni coppia Darlington per operare direttamente con gli strumenti TTL oppure 5-V CMOS [16]. Poiché include i driver dei relé e consente di trasformare le uscite digitali del PIO (associate a 0 e 5V), in uscite (ovvero tensioni e correnti) compatibili per i relè, è stato utilizzato nel ambito della scheda per il pilotaggio di alcuni relé, utilizzati per realizzare un multiplexer. Questo integrato, inoltre, consente di semplificare il circuito rispetto a componenti discreti, ovvero senza il suo utilizzo avremmo dovuto inserire per ogni singolo relé: 2 resistenze, 1 transistore e 1 diodo; per cui si è ottenuto un risparmio nello spazio e nei costi.

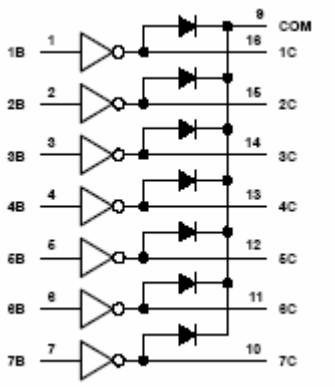


Figura 41–Diagramma logico

Questo integrato è dotato di 16 pin, la cui descrizione è indicata in figura 42, dove il circuito integrato è visto dall' alto.

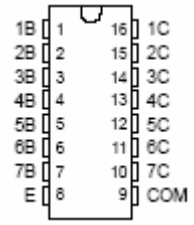


Figura 42-ULN2003A: Descrizione pin

1B a 7B-Ingressi

1C a 7C-Uscite

E- Massa

COM- Comune dei diodi (tipicamente collegato alla tensione di alimentazione dei carichi controllati)

2.9 Il Relé: HM4101F

Il Relé che utilizziamo, HM4101F, mostrato in figura 43, è costituito da un elettromagnete, che eccitato elettricamente, facendo passare un flusso di corrente in una bobina di filo, attrae una struttura di ferro, aprendo e chiudendo un contatto. In sostanza il relé è un interruttore che non viene azionato a mano ma da un elettromagnete, come mostrato in figura 44 [4].

Figura 43- HM4101F

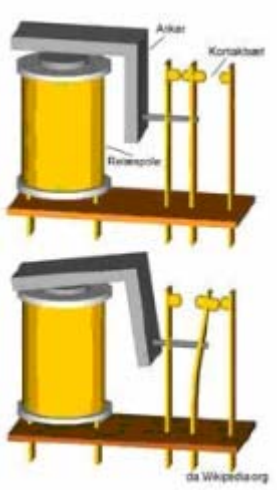


Figura 44– Relé

Con l'utilizzo di 4 relé, è stato realizzato un semplice multiplexer, mostrato in figura 45, in cui ogni Relé in base al segnale ricevuto in ingresso seleziona in uscita uno tra i due esperimenti caotici: circuito di Chua oppure il circuito non autonomo con induttanza e diodo RLD. In effetti abbiamo un pilotaggio con ingressi indipendenti, ma sincroni.

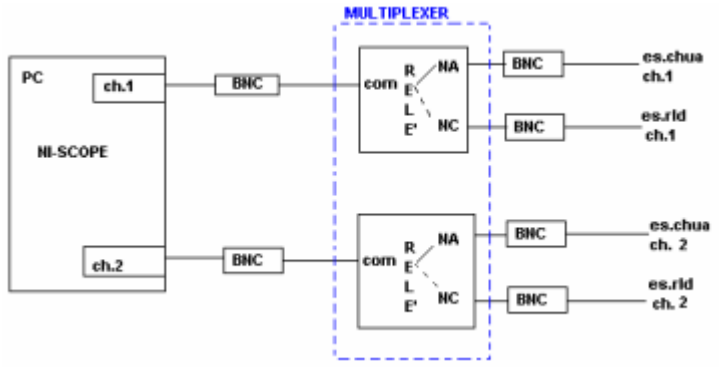


Figura 45- Multiplexer

Analizziamo più nel dettaglio il collegamento del singolo Relé, come mostrato in figura 46, ed il suo funzionamento [17]. Il primo piedino della bobina, in alto a destra, è collegato alla tensione di alimentazione esterna, pari a +12 V; il successivo piedino della bobina è collegato all' ULN2003A, che fornisce tensioni e correnti compatibili al Relé. Se la bobina non è alimentata il comune è collegato al piedino NC (normalmente chiuso), altrimenti il comune si collega al piedino NA (normalmente aperto). A questi due piedini NA ed NC sono stati collegati dei BNC, connettori standard nell' ambito del laboratorio, che consentono il collegamento con gli esperimenti.

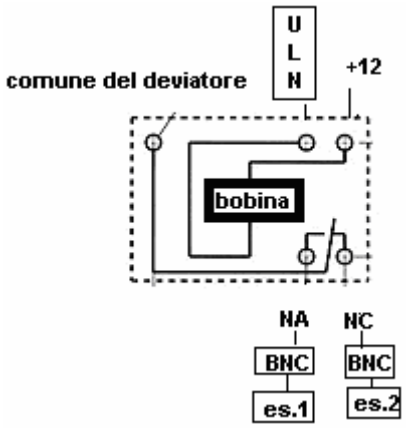


Figura 45– Rappresentazione del collegamento di un Relé

Con questo tipo di collegamento avremo che il multiplexer seleziona in uscita, il circuito di Chua nel caso in cui la bobina è alimentata, altrimenti viene selezionato il circuito RLD.

2.10 Interfaccia SPI

Una volta introdotti i vari integrati necessari per la realizzazione della scheda I/O A/D, prima di analizzare come li abbiamo collegati, è necessario presentare l'interfaccia utilizzata per consentire una comunicazione seriale tra i dispositivi periferici: la SPI (Serial Peripheral Interface). Essa è usata soprattutto per una comunicazione seriale sincrona di un "host processore" e delle unità periferiche; infatti è una semplice interfaccia che permette la comunicazione tra controllori e circuiti integrati periferici o l' interconnessione fra due o più controllori.

Il Bus SPI usa un protocollo sincrono, dove la trasmissione e la ricezione è guidata da un segnale di clock (SCKL) generato dal controllore. L' interfaccia SPI consente di collegare diversi dispositivi mentre il controllore seleziona uno di loro con il segnale CS (Chip Select, a volte denotato come Slave Select SS), l' underline significa che il chip selezionato è attivo se il segnale è a un livello logico basso, cioè uguale a zero.

SPI usa un modello Master-Slave, il dispositivo Master fornisce il segnale di clock e determina lo stato del chip select, cioè attiva lo Slave se vuole comunicare con esso, quindi CS e SCLK sono output. Il dispositivo Slave riceve il clock e il chip select dal Master, per cui CS e SCLK sono input. Ciò significa che c'è un Master, mentre il numero di Slave è limitato solamente al numero di chip select.

Il Bus SPI consiste di quattro linee di segnale, come mostrate in figura 47:

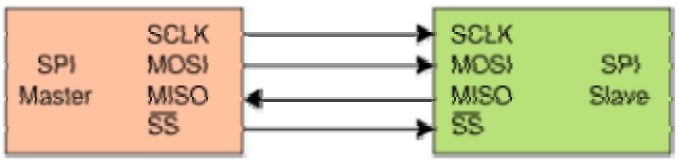


Figura 47- SPI Bus

SCKL- Serial Clock, è generato dal master per sincronizzare i dati di input e output trasmessi, quindi è un input in tutti i dispositivi Slave.

MOSI- Master out Slave in, il segnale di dati è generato dal Master e ricevuto dallo Slave,quindi il segnale MOSI è definito come output dal dispositivo Master e come input nei dispositivi Slave. Il trasferimento dei dati è solo in una direzione, dal Master allo Slave.

MISO- Master In Slave Out , i segnali di dati sono generati da Slave, quindi il segnale MISO è definito come input nel dispositivo Master e come output in un dispositivo Slave. Il trasferimento dei dati è solo in una direzione, da uno Slave al Master.
SS (o CS)- Slave Select (o Chip Select), è generato dal Master per attivare gli Slave[4].

Nel caso in cui il dispositivo Master comunica con più dispositivi Slave, come mostrato in figura 48:

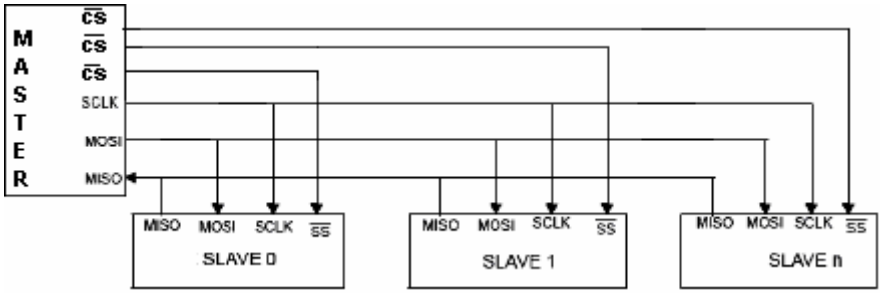


Figura 48- Spi Bus, con un Master e tre dispositivi Slave

il Master dovrà avere per ogni Slave un segnale di CS indipendente, inoltre dovrà generare una frequenza di clock più o meno uguale alla frequenza massima che i dispositivi Slave supportano.
Per quanto riguarda i dispositivi Slave, non attivati dal Master, dovranno ignorare il clock e le linee MOSI e MISO.

2.11 Analisi dei collegamenti tra gli integrati

Nella nostra realizzazione, utilizziamo tutte e quattro le linee di segnale che costituiscono il Bus SPI, in particolare, il segnale MOSI per il trasferimento dei dati in uscita, inviati dal PC, tramite il dispositivo UM245R che funge da Master, agli integrati periferici, ovvero al MCP23S17, al DAC, all' ADC che rappresentano gli Slave; il segnale MISO , per il trasferimento dei dati d' input per il PC, inviati da uno degli Slave al Master. I segnali di Chip Select per l' attivazione dei componenti Slave da parte del Master ed infine il segnale di Clock per le tempificazioni nelle operazioni di lettura e scrittura nei vari integrati, come mostrato in figura 49.

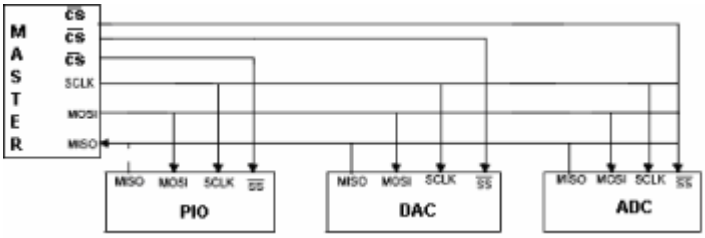


Figura 49- collegamento tra gli integrati

2.11.1 Realizzazione dei collegamenti

Vediamo più nel dettaglio come questi collegamenti sono stati realizzati dal punto di vista hardware.

Il primo integrato posto nella scheda realizzata, è l' **UM245R**, il quale svolge il ruolo del Master. Come mostrato nel capitolo 2, di questo dispositivo ci sono serviti solo alcuni terminali per il pilotaggio degli altri integrati. In particolare abbiamo configurato il pin DB0 per la trasmissione del segnale di dati (MOSI); il pin DB1 per il segnale di clock (SCLK); i pin DB2-DB6 per il segnale Chip Select (CS) ed infine il pin DB7 in cui riceviamo il segnale di dati (MISO). Tra i vari Slave ad esso collegati, cominciamo con il considerare il Port Expander, **MCP23S17**. Il collegamento, tra il master e questo slave, è stato effettuato nel seguente modo: il terminale SO (pin 14) al pin DB7; il terminale SI (pin 13) al pin DB0; il terminale SCK (pin 12) al pin DB1, il terminale CS (pin 11) al pin DB2. I restanti terminali sono stati collegati nel seguente modo: Vss (pin 10) è stato collegato a massa (0 V); Vdd (pin 9) è stato collegato alla tensione

di alimentazione (+5 V) e ad un condensatore, che effettua il filtraggio dei disturbi. In quanto i circuiti digitali quando sono in stato di riposo assorbono correnti trascurabili, mentre quando il loro stato interno varia assorbono impulsi di corrente; per cui per livellarli vengono utilizzati i condensatori. I terminali GPB7-GPB0 sono stati collegati ai terminali di ingresso,1B-7B, del ULN2003A; i terminali A0-A1-A2 (pin 15-16-17 rispettivamente) sono stati collegati a massa per dargli indirizzo zero; il terminale di RESET (pin 18) è stato collegato alla tensione di alimentazione, in quanto esso risulta attivo ad un livello logico basso; i terminali INTB-INTA (pin 19-20 rispettivamente) non sono stati collegati; i terminali GPA0-GPA7 sono stati collegati al circuito di Chua, per fornirgli gli otto bit in ingresso digitali.

Successivamente abbiamo il collegamento tra il **TL431** ed il Trimmer, il quale è stato effettuato nel seguente modo: il terminale REF è stato collegato ad una resistenza di 10kΩ; il terminale anodo è stato collegato a massa infine il terminale REF è stato collegato al terminale catodo e al terminale 1 del Trimmer; il terminale 2 del Trimmer è stato collegato al terminale REFIN del DAC; il terminale 3 del Trimmer è stato collegato ad un' altra resistenza di 2kΩ.

Il DAC, **TLV5616**, è stato collegato all' UM245R e al Port Expander (PIO) nel seguente modo: il terminale DIN (pin 1) è stato collegato al terminale SO del PIO; il terminale SCLK (pin 2) è stato collegato al terminale SCLK del PIO; il terminale CS (pin 3) è stato collegato a Massa ; gli altri terminali sono stati collegati nel seguente modo: il terminale FS (pin 4) è stato collegato al terminale DB3 del UM245R, lo usiamo come CS; il terminale AGND (pin 5) è stato collegato a massa; il terminale REFIN (pin 6) è stato collegato al terminale 2 del Trimmer, inoltre è stato collegato al terminale REF+ dell'ADC; il terminale OUT (pin 7) è stato collegato all' ingresso AIN0 dell' ADC per verificarne il funzionamento; il terminale V_{dd} (pin 8) alla tensione d'alimentazione (+ 5V) e ad un condensatore, che effettua il filtraggio dei disturbi. In quanto i circuiti digitali quando sono in stato di riposo assorbono correnti trascurabili, mentre quando il loro stato interno varia assorbono impulsi di corrente; per cui per livellarli vengono utilizzati i condensatori .

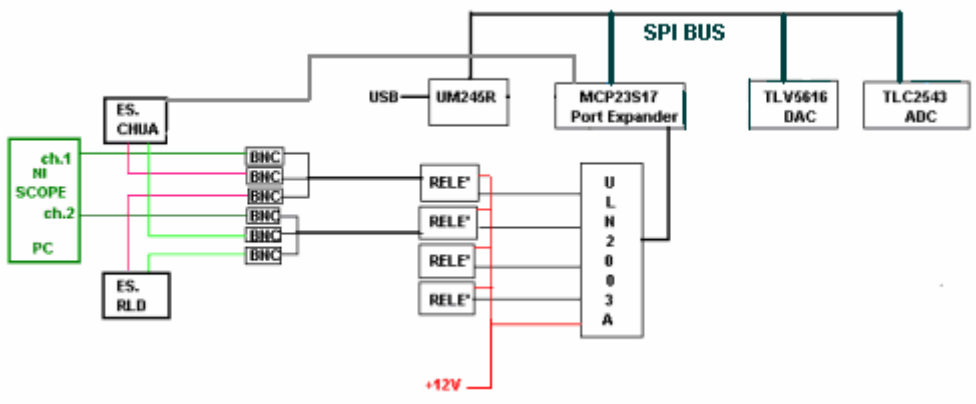
L' ADC, **TLC2543**, è stato collegato all' UM245R e al Port Expander (PIO) nel seguente modo: il terminale GND (pin 10) è stato collegato a massa; il terminale REF- (pin 13) è stato collegato a Massa; il terminale REF+ (pin 14) è stato collegato al terminale REFIN del DAC; il terminale CS (pin 15) è stato collegato al terminale DB5 del Master; il terminale DATA OUT (pin 16) al terminale SO del PIO; il terminale

DATA INPUT (pin 17) al terminale SI del PIO; il terminale I/O CLOCK (pin 18) è stato collegato al terminale SCLK del PIO; il terminale EOC (pin 19) non è stato collegato; il V_{cc} (pin 20) è stato collegato alla tensione di alimentazione e ad un condensatore, che effettua il filtraggio dei disturbi. In quanto i circuiti digitali quando sono in stato di riposo assorbono correnti trascurabili, mentre quando il loro stato interno varia assorbono impulsi di corrente; per cui per livellarli vengono utilizzati i condensatori .

All' esterno della scheda, possiamo considerare i seguenti due integrati: ULN2003A, che consente il pilotaggio dei relé da parte del Port Expander; ed alcuni RELE', con i quali è stato possibile realizzare il multiplexer.

L' **ULN2003A** è stato collegato nel seguente modo: gli ingressi 1B – 7B sono stati collegati ai terminali del lato B del Port Expander ; il terminale E è stato collegato a massa; le uscite 1C, 2C sono state collegate rispettivamente ad uno dei terminali della bobina, interni ai due Relé utilizzati per realizzare il multiplexer, ed infine il terminale COM è stato collegato alla tensione esterna +12V, che viene fornita ai relé.

I relé, **HM4101F**, sono stati collegati nel seguente modo: il terminale COMUNE è stato collegato ad un BNC, che verrà collegata ad uno dei canali della NI-SCOPE; il terminale NA è stato collegato ad un BNC, che verrà collegato ad uno dei canali degli esperimenti; il terminale NC è stato collegato ad un BNC che verrà collegato ad uno dei canali degli esperimenti; i due terminali della bobina, come mostrati nel capitolo 2, verranno collegati uno ad un alimentatore esterno per fornirgli una tensione di alimentazione di +12V e l' altro terminale verrà collegato all' ULN2003A, in modo tale da rendere compatibili le uscite del PIO agli ingressi del relé, consentendo il pilotaggio dei relé da parte del PIO. La realizzazione dei collegamenti è mostrata in figura 50.



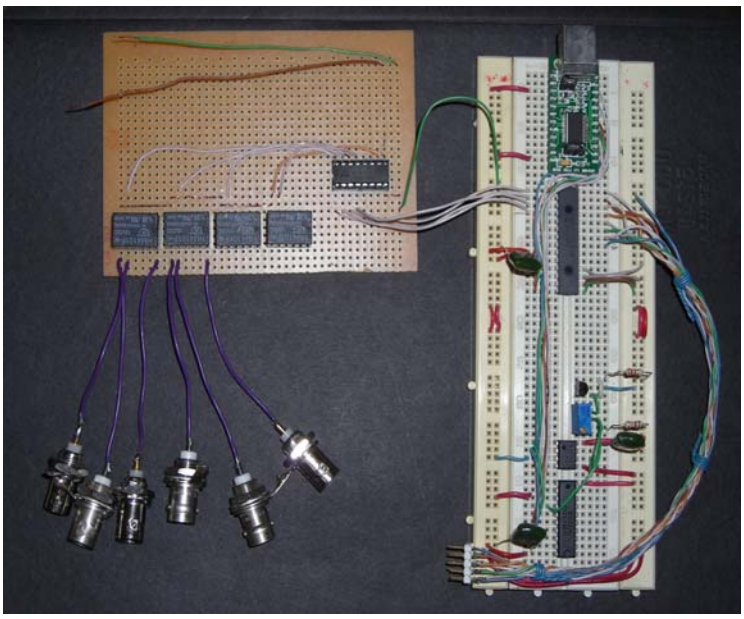


Figura 50- Realizzazione dei collegamenti

Capitolo 3

Controllo software della scheda

3.1 Breve premessa

Nel capitolo seguente, viene inizialmente presentato il linguaggio di programmazione grafica utilizzato, il LabView, ed in secondo momento la descrizione dei SubVI realizzati, che implementano le funzionalità di configurazione e controllo della scheda.

3.2 Introduzione al LabView

3.2.1 Cos'è il LabView

Il LabView (Laboratory Virtual Instrument Engineering Workbench) è un ambiente di sviluppo per applicazioni principalmente orientate: all' acquisizione di dati ed alla gestione di strumentazione elettronica, all' analisi ed elaborazione dei segnali. Il suo contributo specifico allo sviluppo lo si deve al vasto numero di schede di acquisizione e sistemi di misura, dalle quali è usato come principale software attraverso cui interagiscono e vengono programmate[18].

LabView fornisce un ambiente di programmazione di tipo grafico ad oggetti denominato "G-Language", il quale consente di realizzare programmi in forma di diagrammi a blocchi, in quanto utilizza icone invece di linee di testo per creare applicazioni.

LabView conserva comunque molte similitudini con gli ambienti di programmazione tradizionali: presenta tutti i tipi di dati e gli operatori predefiniti di uso comune, permette di generare nuovi tipi di dati combinando tra loro i tipi di dati elementari e di

controllare l' esecuzione dei programmi ricorrendo a strutture di controllo di flusso come ad esempio cicli e costrutti per l' esecuzione condizionale di codice.
Questo ambiente di sviluppo software presenta anche delle alcune peculiarità che lo differenziano notevolmente dai linguaggi procedurali più comunemente noti.
La prima differenza sostanziale è l' ambiente di sviluppo grafico, sia dell' interfaccia utente sia dell' algoritmo di elaborazione. Altra differenza importante è che un programma LabView non esegue il flusso delle istruzioni, ma bensì, segue il flusso di dati. Infatti i programmi scritti in LabView tendono ad essere di tipo data-driven, nel senso che si tende ad enfatizzare come i dati si muovono tra i diversi blocchi operativi più che la sequenza delle istruzioni da eseguire.
Mette inoltre a disposizione del programmatore una serie di librerie di funzioni che possono essere richiamate ed utilizzate all' interno dei programmi: le librerie comprendono funzioni di uso comune (funzioni aritmetiche e statistiche, funzioni per la manipolazione di stringhe, ecc.) ed inoltre funzioni specializzate per l' acquisizione e l'elaborazione dei segnali, il controllo di strumentazione numerica via interfaccia IEEE-488 o VXI, la trasmissione di dati mediante l' uso di porte speciali oppure mediante il protocollo di comunicazione TCP/IP.
E' possibile inoltre definire nuove funzioni ed arricchire le librerie in dotazione a LabView.
Infine il programma consente di effettuare il debug delle applicazioni create in linguaggio G attraverso opportune modalità di esecuzione dei programmi, come ad esempio il modo "highlight execution" o "single step" e per mezzo di oggetti che consentono in run-time la modifica di variabili di programma.
LabView presenta alcuni vantaggi rispetto ad un linguaggio di programmazione tradizionale: è di facile apprendimento, in quanto presenta una modalità di programmazione a blocchi, di tipo visuale ed intuitivo; permette di dare al codice una struttura modulare che consente di suddividere programmi complessi in sottoprogrammi più semplici che possono essere riutilizzati; Consente di raccogliere i VI in librerie, ovvero in un insieme di sub-vi utilizzabili da altri VI e velocemente inseribili nel codice sorgente dal programmatore; fornisce un considerevole insieme di librerie per lo sviluppo di applicativi, tra le quali si trovano funzioni di tipo matematico e statistico, controllo di dispositivi per mezzo di alcuni tipi di interfaccia, comunicazione tra calcolatori,ecc.

3.2.2 Utilizzo principale di LabView

LabView è stato pesato principalmente per il controllo di schede di espansione connesse direttamente al bus di un calcolatore o di strumentazione connessa al calcolatore stesso attraverso opportune interfacce come il bus IEEE 488, RS-232, strumenti VXI o ancora attraverso Internet mediante il protocollo TCP/IP.

L' ambiente di sviluppo consente di costruire programmi i quali prendono il nome di ***strumenti virtuali*** (Virtual Instrument, **VI**).

Un Virtual Instrument permette l' interazione tra calcolatore e strumentazione fornendo contemporaneamente all' utente un opportuno pannello frontale grafico per il dialogo con il VI stesso. In questo modo l' utente interagisce con un nuovo dispositivo (Instrument), costituito da calcolatore, interfacce, strumenti e programma il quale presenta una realtà (Virtual) diversa dai singoli oggetti fisici che compongono il sistema stesso. Tale fatto spiega il nome di Virtual Instrument dato ad un programma LabView[18].

3.2.3 Virtual Instrument (VI)

Si ricorda che i programmi che si possono realizzare utilizzando il linguaggio grafico LabView sono chiamati **Virtual Instrument (VI)**, dove il termine "strumenti" è dovuto al fatto che durante l' esecuzione i programmi sviluppati presentano agli utilizzatori una interfaccia analoga a quella di uno strumento di misura, mentre il termine "virtuali" è dovuto al fatto che l' interazione avviene con un programma in esecuzione e non con un dispositivo fisico dedicato.

L' utilizzatore può modificare il valore di alcune grandezze agendo su opportune manopole o interruttori visualizzati dal programma e può osservare il risultato delle elaborazioni condotte internamente al VI su display grafici molto simili a quelli che si trovano sulla strumentazione numerica.

Un VI è composto da tre parti fondamentali:Pannello frontale(Front Panel); Diagramma a blocchi funzionale (Block diagram);Icona/connettore (Icon/connector).

Il ***Front Panel*** (pannello frontale) è la finestra che rappresenta l' interfaccia tra il programma e l' utilizzatore: il nome deriva dal fatto che può essere strutturato in modo

tale da ricordare il pannello frontale dotato di display, indicatori, manopole, tasti, ecc… Nel pannello frontale vengono allocati i *controllori* gli *indicatori* dello strumento virtuale. Il *controllore* è una variabile di ingresso, che può essere modificata agendo sul pannello frontale, tramite la *tools palette*: è un pannello che mette a disposizione una serie di strumenti che consentono di modificare il valore di un controllore o di selezionare il testo interno; consente di selezionare, posizionare, ridimensionare oggetti; consente di editare il testo e di creare etichette;consente di collegare i terminali dei nodi (ossia controllori con VI, indicatori con VI, ecc.) presenti nel block diagramm; apre il menù di pop-up di un oggetto; ecc. Per *indicatore* si intende una variabile di uscita il cui valore può essere modificato dal programma e non dall'utente.

E' possibile agire con un front panel mediante tastiera o mouse, introducendo valori numerici o stringhe di caratteri, modificare lo stato di elementi grafici, come ad esempio manopole, bottoni e così via.

Il **Block Diagram** (diagramma a blocchi funzionale) contiene il codice nella forma di diagramma a blocchi ed è costituito da: *Nodi*, sono degli elementi di elaborazione; *Collegamenti,* uniscono i nodi e permettono lo scambio di informazioni. Le informazioni passano da un nodo all'altro del pannello frontale per mezzo dei connettori, che li uniscono.

Pur presentandosi in forma grafica diversa, il diagramma a blocchi presenta possibilità di programmazione analoghe a quelle offerte da un comune linguaggio di programmazione del tipo text-based. La coppi ***Icon/connector*** (icona/connettore) è il terzo elemento fondamentale di un programma LabView e consente di trasformare un programma in un oggetto come verrà meglio chiarito nel paragrafo successivo. L' icona è un simbolo grafico di piccole dimensioni , che rappresenta sinteticamente il VI stesso:essa è normalmente visibile in alto a destra della finestra front panel e della finestra block diagram. Il connettore consente di associare, nell' ambito del pannello frontale alle aree dell' icona, i controllori e indicatori che caratterizzano il funzionamento del VI [18].

3.2.4 Sub-VI

Un **Sub-VI** è un VI, utilizzato all' interno di un altro VI come sottoprogramma.
Qualunque VI, una volta creato, può essere utilizzato come sub-VI all' interno di un altro VI di più alto livello. Di conseguenza un Sub-VI è analogo ad una subroutine in un linguaggio di programmazione di tipo text-based. Come avviene in altri linguaggi con le soubroutine, non c'è alcun limite al numero di Sub-VI inseribili all' interno di un programma in linguaggio G. E' inoltre possibile chiamare un Sub-VI dall' interno di un altro Sub-VI.
Quando un Sub-VI, viene inserito all' interno di un VI, presenta l' icona ad esso associata nel block diagram che lo contiene.
Quando impiegato come SubVI, un VI normalmente non mostra a video il proprio pannello frontale, ma si limita a ricevere i dati di ingresso per mezzo dei collegamenti tra il sub-VI stesso ed il resto del programma che lo contiene ad elaborarli e a fornire le eventuali uscite[18].

3.3 Creazione dei SubVI per il funzionamento della scheda

In questo ambiente di programmazione grafica, LabView, sono stati creati dei SubVI, che costituiscono il modello di programmazione della scheda. Nel caso particolare della realizzazione della scheda, sono stati utilizzati per pilotare da PC i vari integrati, interni alla scheda. Inoltre, alcuni di essi sono stati inseriti all' interno del VI, che riproduce sullo schermo del PC i controlli usuali dello oscilloscopio, in modo tale da poter collegare la scheda realizzata al PC e agli esperimenti per effettuarne la remotizzazione.

3.3.1 SubVI: USBIO Init

Questo SubVI è stato realizzato tramite un' opportuna concatenazione delle primitive, citate nel capitolo 2, ed alcuni SubVI, descritti più avanti, per ottenere un programma che consenta da PC di settare singolarmente i pin da utilizzare, dei due dispositivi UM245R ed MCP23S17, come input oppure output. In particolare, per la realizzazione della concatenazione è stata utilizzata la Flat Sequenze Structure, una struttura che permette la sequenzializzazione dei sub-diagrammi contenuti in essa, o per meglio dire delle funzioni presenti al suo interno. Consideriamo il primo sottoblocco.

-Attivazione UM245R

Questo primo sottoblocco utilizzata la funzione **FT_OPEN_DEVICE_BY_INDEX.vi** , la quale apre la comunicazione con il dispositivo (UM245R).

Essa riceve in ingresso il parametro Device Index , un controllore che indica il numero di dispositivi da aprire, connessi al PC, nel caso in cui ce ne sia uno solo occorre porlo uguale a zero, se invece sono due bisogna settarlo a uno e così via. Quindi, dato che nella nostra realizzazione usiamo un solo dispositivo questo parametro avrà come valore zero.

La funzione ritorna il valore del puntatore Handle, un indicatore che mostra una sorta di "indirizzo" identificativo del dispositivo collegato e che sarà usato per accessi successivi; ed il parametro FT_STATUS, un indicatore che fornisce lo stato della funzione. Il successivo sottoblocco è il seguente.

-Configurazione dei registri di direzione interni all' integrato UM245R

Quando si attiva l' integrato UM245R , per come è stato progettato, presenta al suo interno tutti i pin dei registri configurati come ingressi, nel senso che presenta come valore di configurazione dei pin (DB0-DB7) tutti zero; di conseguenza per evitare transazioni spurie sui CS conviene porre inizialmente tutti i pin pari a 1. Per la sua implementazione è stato utilizzato il SubVI FT_Write_All_Data.vi, che consente di scrivere dati all' interno del dispositivo UM245R; essa riceve in ingresso un Buffer di valori costanti, in particolare 11111111, ai quali corrispondono rispettivamente ai pin DB7-DB0; la funzione ritorna la FT_Status_A un indicatore che fornisce lo stato della funzione e i Bytes Written_A, un indicatore che indica il numero di Byte effettivamente scritti. Il successivo sottoblocco connesso è il seguente.

-Configurazione dell' integrato UM245R

Abbiamo utilizzato per la sua implementazione la funzione FT_SET_BIT_MODE.vi , la quale consente di configurare i bit del byte DB0-DB7 singolarmente come input e come output. In particolare, tale funzione riceve in ingresso oltre all'Handle della FT_OPEN_DEVICE_BY_INDEX.vi, che quindi diventa un controllore, il valore della maschera "Bit Mask", un controllore, nel quale i bit sono configurati singolarmente per l' input e per l' output; nel seguente modo, se il bit è 0, il corrispondente pin è settato come input, se il bit è 1, il corrispondente pin è settato come output. Esso è rappresentato nel pannello frontale del VI in binario per consentire all' utente di comprendere subito come sono stati utilizzati i vari pin. Nel nostro caso, è stato settato in tal modo: 00011111, ed è stato posto come valore di default nel pannello frontale; c'è da osservare che questi bit sono letti dal più significativo a quello meno significativo, ovvero da DB7 a DB0. In cui DB0 è configurato per l' invio dei dati (MOSI); il pin DB1 è configurato per la trasmissione del clock (SCLK), i pin DB2-DB6 sono configurati come CS , per l' attivazione dei dispositivi Slave ed infine il pin DB7 per la ricezione dei dati (MISO).

Un altro input della funzione è il "bit mode", un controllore che mostra il tipo di configurazione del dispositivo da usare , posto da noi, come costante, pari ad 1, in quanto indica la configurazione di tipo " bit bang mode" asincrona, quella da noi usata.

In uscita alla funzione abbiamo l' indicatore FT_STATUS_SET, che fornisce lo stato della stessa. L' ultimo blocco da analizzare è il seguente.

-Configurazione generale del funzionamento interno del circuito integrato MCP23S17

In questo sottoblocco viene effettuata la configurazione dei bit come ingressi/uscite, scrivendo nel registro IOCON (relativo al lato A e B). La sua implementazione è stata realizzata tramite la funzione FT_WRITE_BYTE_DATA.vi, che scrive i dati sull' UM245R ed è utilizzata per indicare sui bit selezionati per l' output il loro valore che può essere alto (1) oppure basso (0), rispettivamente per convenzione 5 V oppure 0 V. Successivamente i valori in uscita vengono scritti nel registro IOCON del Port Expander, tramite il quale è possibile configurare i byte del registro IOCONA e i byte del registro IOCONB come ingressi/uscite. In particolare, la funzione riceve in ingresso insieme all' Handle generato dalla FT_OPEN, l' Array in uscita dal SubVI

send23s17.vi, il quale riceve in ingresso tre array da 1 byte, in rappresentazione esadecimale:

- Device opcode: è un byte d' indirizzamento del dispositivo.
- Indirizzo del registro: in cui specifico il registro su cui voglio operare, per esempio A oppure B
- Valore: il dato che voglio scrivere.

Un altro ingresso da considerare per la FT_WRITE_BYTE_DATA.vi è l' uscita della funzione LabView "array size" (la quale fornisce in uscita la dimensione dell' array in ingresso) al cui ingresso è stata posta l'Array in uscita dal SubVI send23s17.vi.

Infine, in uscita alla Flat Sequenze Structure è stato posto l' indicato Handle, un indicatore che mostra una sorta di "indirizzo" identificativo del dispositivo collegato e che sarà usato per accessi successivi. Questo è il SubVI realizzato per configurare i due integrati l' UM245R e il Port Expander MCP23S17; il Front Panel e il Block Diagram sono mostrati in figura 51.

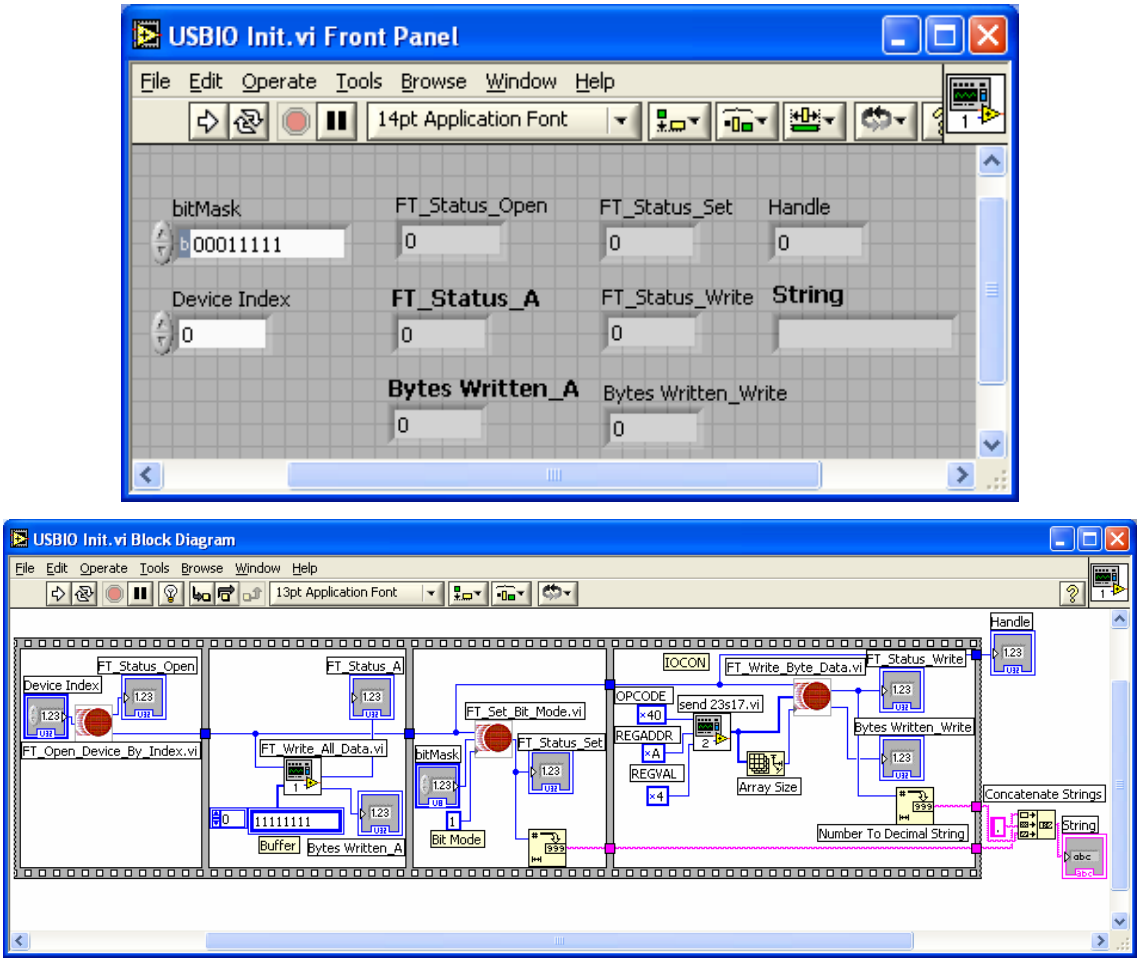


Figura 51- USBIO Init

3.3.2 SubVI: USBIO DigitalSetDir

Questo SubVi scrive nei registri IODIRA e IODIRB la configurazione dei rispettivi pin come i/o. La sua implementazione è stata realizzata utilizzando una struttura CASE, alla quale è stata posta come condizione di accesso la selezione di uno dei due registri, in particolare è stato associato 0 al registro A e 1 al registro B. All' interno della struttura troviamo inizialmente un collegamento diretto tra l'indicatore Handle in e il controllore Handle out, questa istruzione fa parte delle tecniche sequenziali, il cui vantaggio è quello di occupare poco spazio e di garantire la sequenzialità dell' esecuzione; inoltre è stata posta la funzione FT_WRITE_BYTE_DATA.vi, che riceve come ingressi insieme all' Handle_in l' array in uscita dal SubVi send23s17.vi e l'uscita della "array size", al cui ingresso è stato posto l'array in uscita dal send23s17.vi; la funzione fornisce in uscita gli indicatori Status, che fornisce lo stato della stessa e Bytes Written , che fornisce il numero di byte effettivamente scritti. Per quanto riguarda la send23s17.vi, è stato posto in ingresso insieme al bit mask, un controllo nel quale i bit sono configurati singolarmente per l' input e l' output, altri due ingressi: opcode (il cui valore, quando si effettua l' operazione di scrittura, è in rappresentazione esadecimale: x40 per entrambi i registri IODIRA/B) ed infine regaddr (contiene l' indirizzo del registro in rappresentazione esadecimale : x00 per IODIRA, mentre x01 per IODIRB), i cui valori sono stati descritti nel capitolo 2; il Front Panel ed il Block Diagram sono mostrati in figura 52.

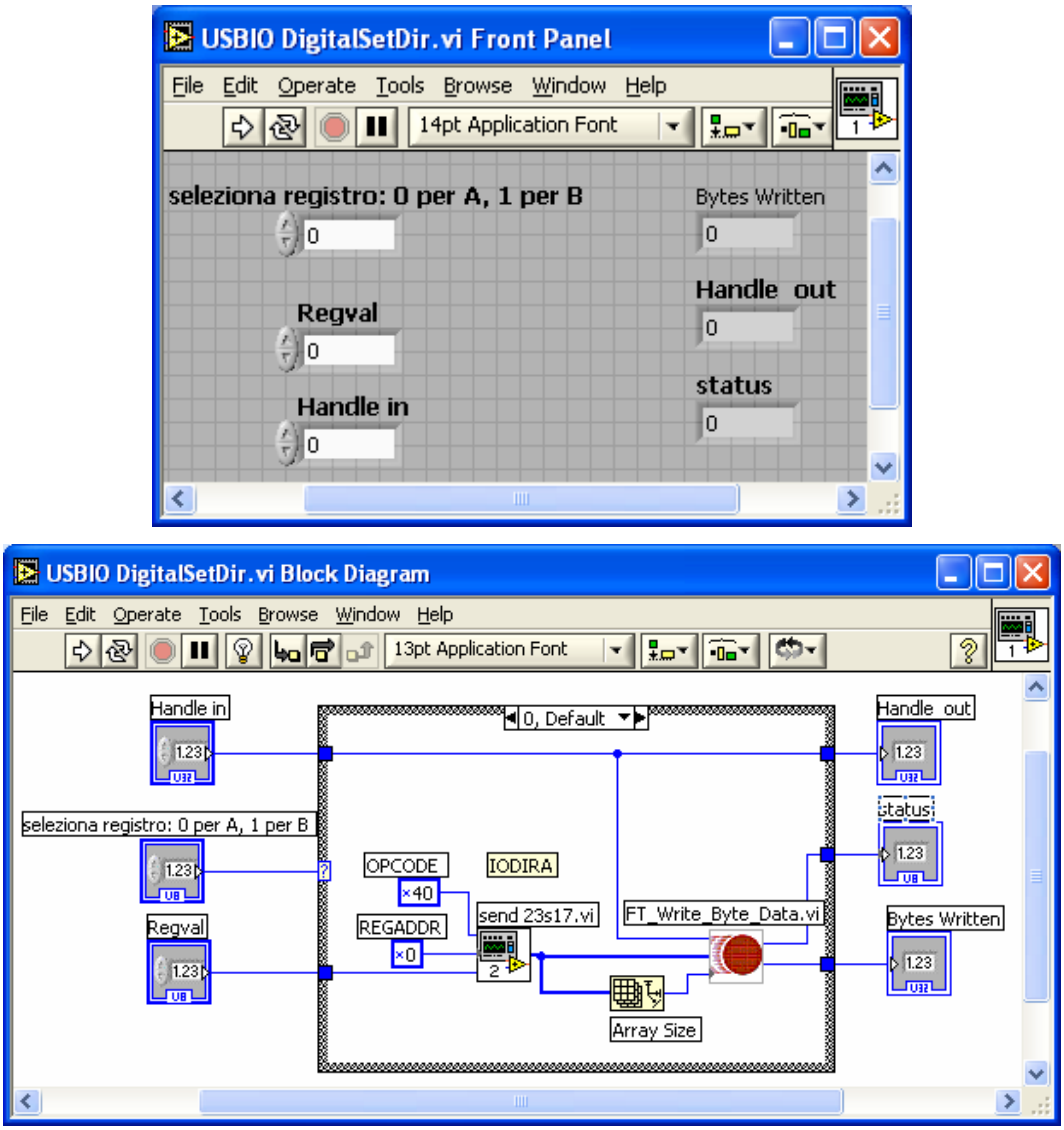


Figura 52- USBIO DigitalSetDir

3.3.3 SubVI: USBIO DigitalWrite

Quest SubVI effettua la scrittura nei registri OLATA e OLATB dei dati in uscita dai rispettivi lati del dispositivo. La sua implementazione è stata realizzata utilizzando una struttura CASE, alla quale è stata posta come condizione di accesso la selezione di uno dei due registri, in particolare è stato associato 0 al registro A e 1 al registro B. All' interno della struttura, troviamo inizialmente un collegamento diretto tra l'indicatore Handle in e il controllore Handle out, questa istruzione fa parte delle tecniche sequenziali, il cui vantaggio è quello di occupare poco spazio e di garantire la sequenzialità dell' esecuzione; inoltre è stata posta la funzione FT_WRITE_BYTE_DATA.vi, che riceve come ingressi insieme all' Handle_in, l'array in uscita dal SubVi send23s17.vi e l'uscita della funzione LabView "array size", al cui ingresso è stato posto l' array in uscita dal send23s17.vi; la funzione fornisce in uscita gli indicatori Status, che fornisce lo stato della stessa e Bytes Written, che fornisce il numero di byte effettivamente scritti. Per quanto riguarda la send23s17.vi, è stato posto in ingresso insieme all' indicatore Dato da trasmettere, il quale contiene i dati in uscita, altri due ingressi: opcode (il cui valore, quando si effettua l' operazione di scrittura, è in rappresentazione esadecimale: x40 per entrambi i registri OLATA/B) ed infine regaddr (contiene l' indirizzo del registro in rappresentazione esadecimale: x14 per OLATA, mentre x15 per OLATB) i cui valori sono stati descritti nel capitolo 2; il Front Panel ed il Block Diagram sono mostrati in figura 53.

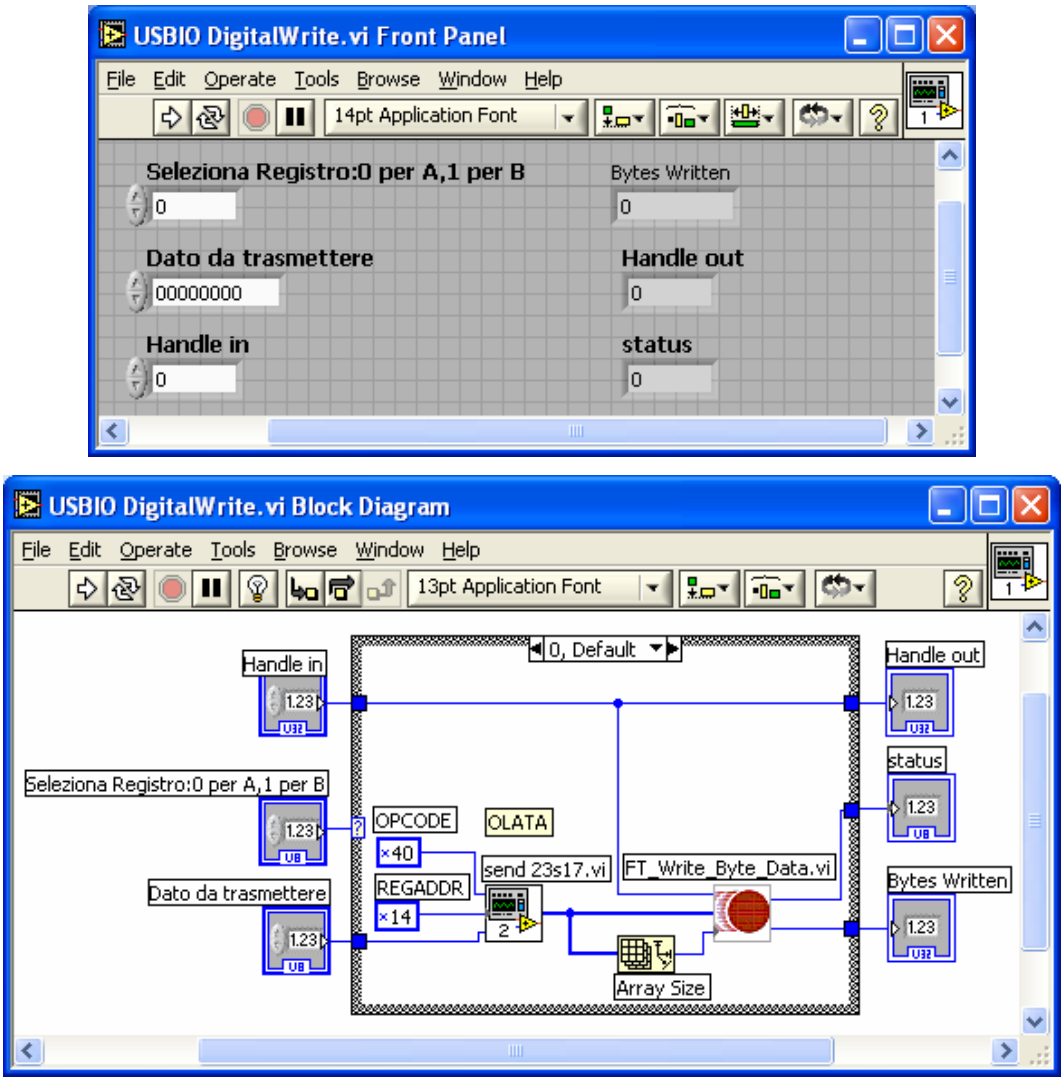


Figura 53-USBIO DigitalWrite

3.3.4 SubVI: USBIO GetDigitalDir

Questo SubVI effettua la lettura nei registri IODIRA e IODIRB.La sua implementazione è stata realizzata utilizzando una struttura CASE, alla quale è stata posta come condizione di accesso la selezione di uno dei due registri, in particolare è stato associato 0 al registro A e 1 al registro B. All' interno della struttura troviamo semplicemente la costante REGADDR, contiene l' indirizzo del registro in rappresentazione esadecimale: x0 per IODIRA (x1 per IODIRB); al suo esterno è stata posto il subvi READ_23S17.vi che riceve come ingressi insieme all' Handle_in, altri due ingressi: opcode (il cui valore, nel caso dell'operazione di lettura, è rappresentato in esadecimale: x41, per entrambi i registri IODIRA/B), regaddr (contiene l' indirizzo del registro in rappresentazione esadecimale: x00 per IODIRA, x01 per IODIRB) i cui valori sono stati descritti nel capitolo 2; la funzione fornisce in uscita gli indicatori Handle out, che mostra una sorta di "indirizzo" del dispositivo collegato e che sarà usato per accessi successivi e un Array; il Front Panel ed il Block Diagram sono mostrati in figura 54.

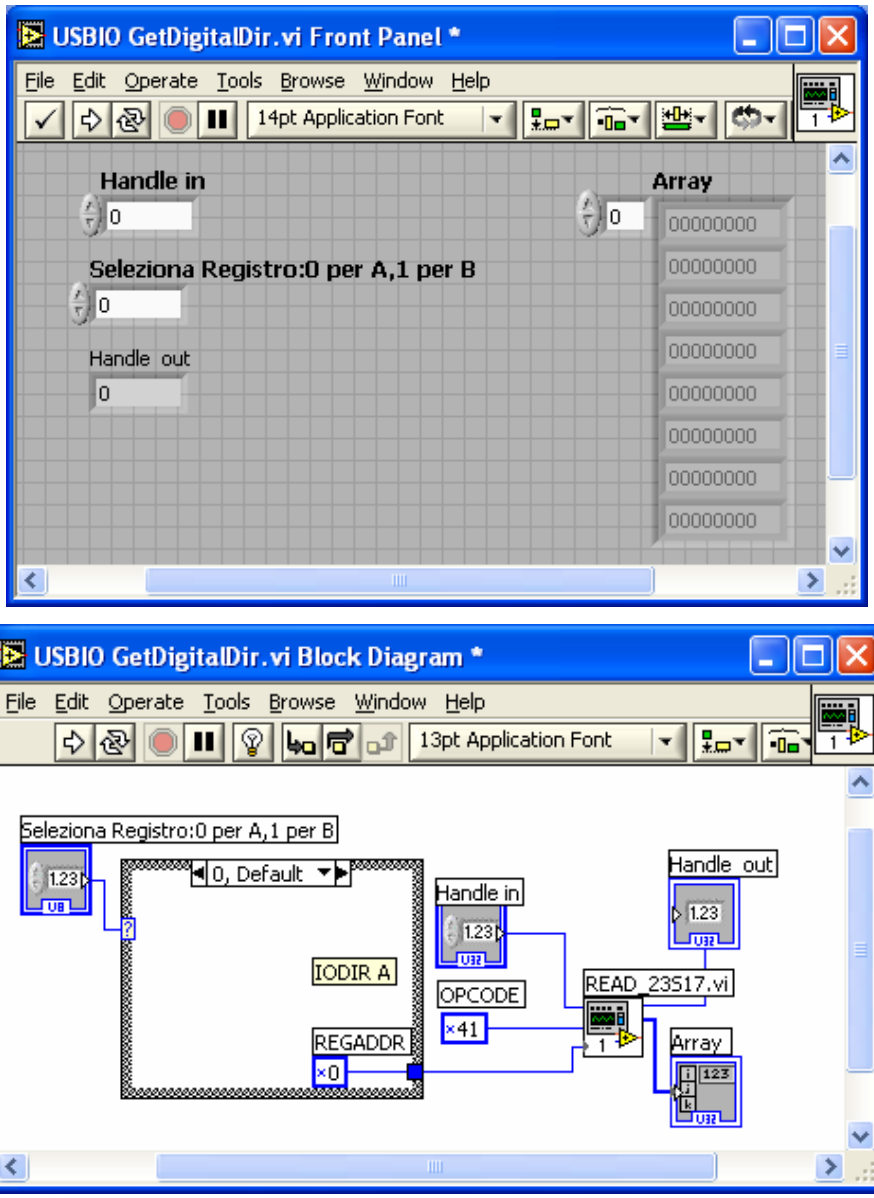
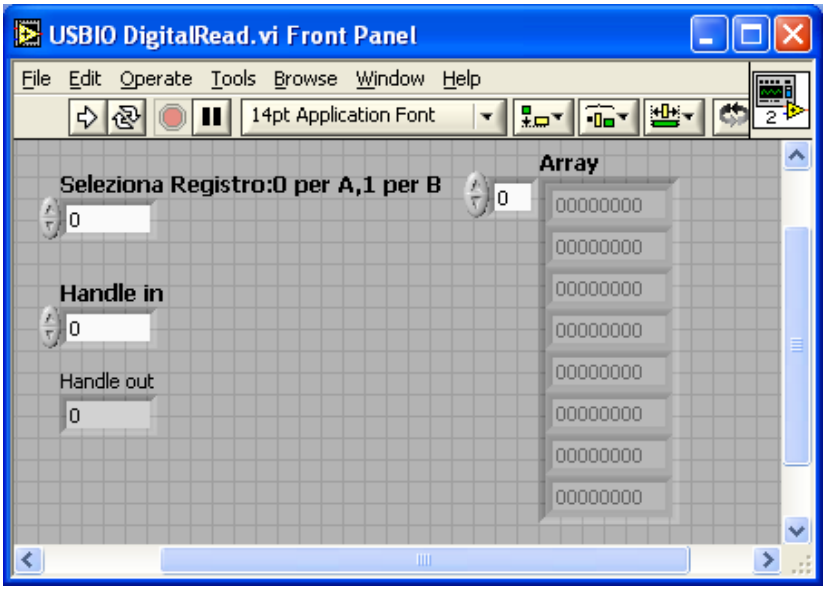


Figura 54- USBIO GetDigitalDir

3.3.5 SubVI: USBIO DigitalRead

Questo SubVI effettua la lettura dai registri GPIOA e GPIOB. La sua implementazione è stata realizzata utilizzando una struttura CASE, alla quale è stata posta come condizione di accesso la selezione di uno dei due registri, in particolare è stato associato 0 al registro A e 1 al registro B. All' interno della struttura troviamo semplicemente la costante REGADDR, che contiene l' indirizzo del registro in rappresentazione esadecimale: x12 per GPIOA (x13 per GPIOB); al suo esterno è stata posto il subvi READ_23S17.vi che riceve come ingressi insieme all' Handle_in, altri due ingressi costanti: opcode (il cui valore, nel caso dell'operazione di lettura, è rappresentato in esadecimale: x41, per entrambi i registri GPIOA/B), regaddr (contiene l' indirizzo del registro in rappresentazione esadecimale: x12 per GPIOA, x13 per GPIOB) i cui valori sono stati descritti nel capitolo 2; la funzione fornisce in uscita gli indicatori Handle out, che mostra una sorta di "indirizzo" del dispositivo collegato e che sarà usato per accessi successivi e Array; il Front Panel ed il Block Diagram sono mostrati in figura 55.



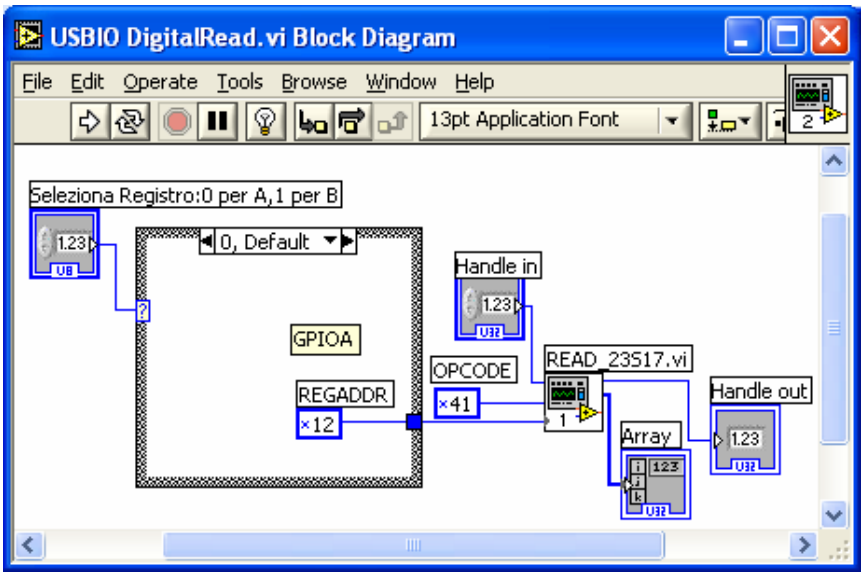


Figura 55- USBIO DigitalRead

3.3.6 SubVI: USBIO AnalogWrite

Questo SubVI effettua la scrittura nel DAC. La sua implementazione è stata realizzata utilizzando la funzione FT_WRITE_BYTE_DATA.vi, che riceve come ingressi insieme all' Handle_in, l'array in uscita dal SubVi send_DAC.vi e l' uscita della funzione LabView "array size" (che fornisce in uscita la dimensione dell' array in ingresso), al cui ingresso è stato posto l'array in uscita dal send_DAC.vi; la funzione fornisce in uscita gli indicatori Status, che fornisce lo stato della funzione e Bytes Written, che fornisce il numero di byte effettivamente scritti. Inoltre al controllore Handle in è stato collegato direttamente l' indicatore Handle out, in modo tale da averlo in uscita al SubVI. Per quanto riguarda il SubVI send_DAC.vi, richiede in ingresso l'uscita della funzione booleana AND. In particolare un controllore word, e una costante, rappresentata da 16 bit 0000111111111111, sono posti in ingresso all' operatore AND; perchè come illustrato nel capitolo 2, il DAC deve riceve in ingresso 16 bit, di cui i primi 4 sono di controllo e i restanti sono relativi al nuovo valore da scrivere. Nel nostro caso abbiamo scelto i primi 4-bit pari a zero; affinché questa scelta non possa essere modificata, viene effettuata questa operazione di mascheratura; il Front Panel ed il Block Diagram sono mostrati in figura 56.

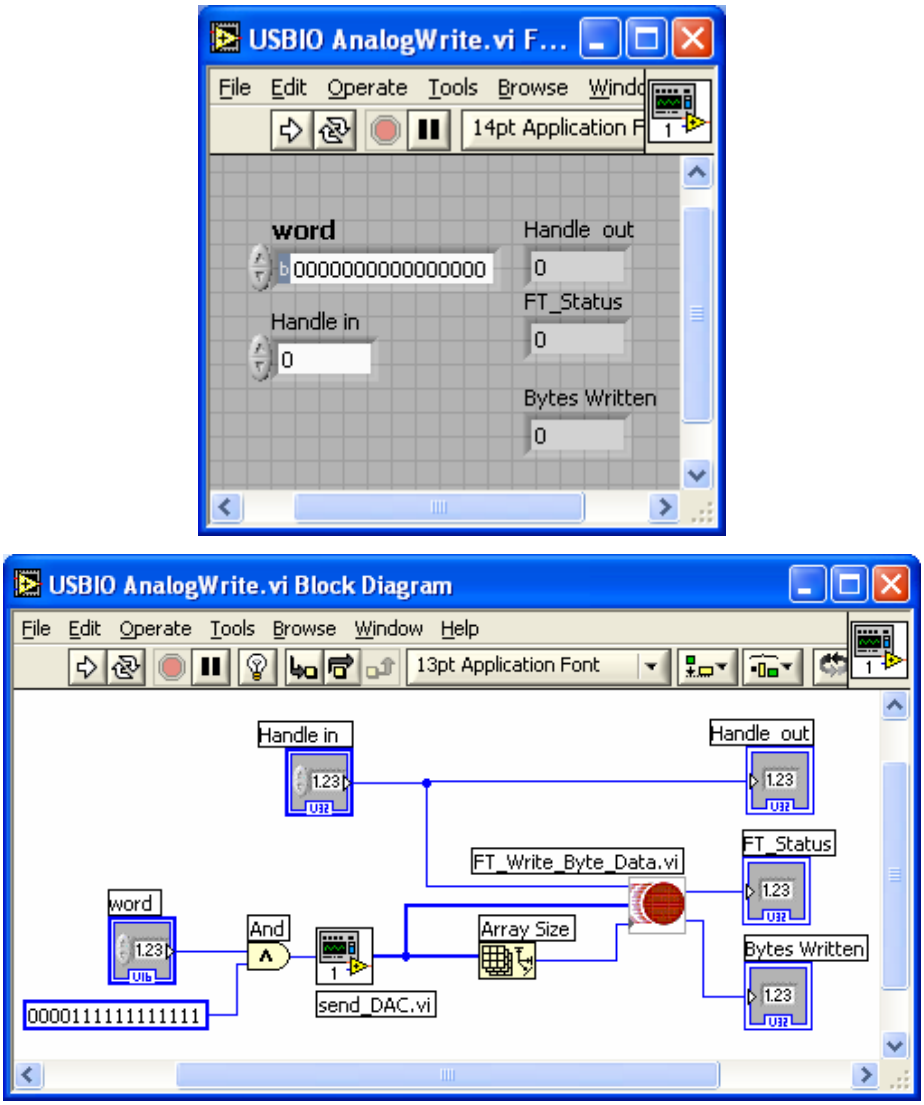
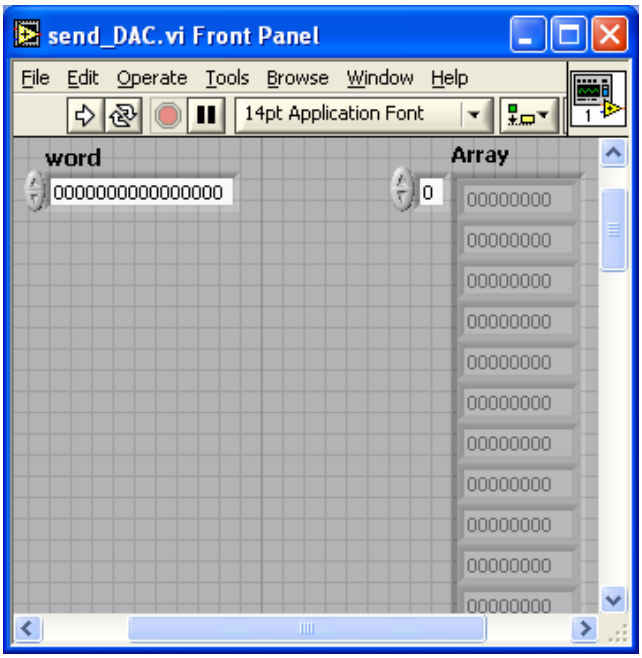


Figura 56- AnalogWrite

3.3.6.1 SubVI: send_DAC.vi

Questo SubVI consente la realizzazione del segnale effettivamente inviato al DAC a partire dai dati che si vogliono effettivamente trasmettere. La sua implementazione è stata realizzata utilizzando il SubVI word_in_bit_3clk_DAC.vi , che riceve in ingresso una word, 16 bit; il SubVI fornisce in uscita un Array di 48 elementi, ai quali viene aggiunto un elemento in testa, in posizione zero, rappresentato da una costante di 8-bit, 01110100, letti dal più significativo a quello meno significativo, con la quale si effettua l'attivazione del CS (=DB3) del DAC, ovvero viene portato il CS al livello logico basso ; ed un elemento in coda, in posizione 49, rappresentato da una costante di 8-bit, 01111100, con la quale viene disattivato il CS del DAC, ovvero viene portato il CS al livello logico alto; in uscita alla funzione è stato posto l' indicatore Array; il Front Panel ed il Block Diagram sono mostrati in figura 57.



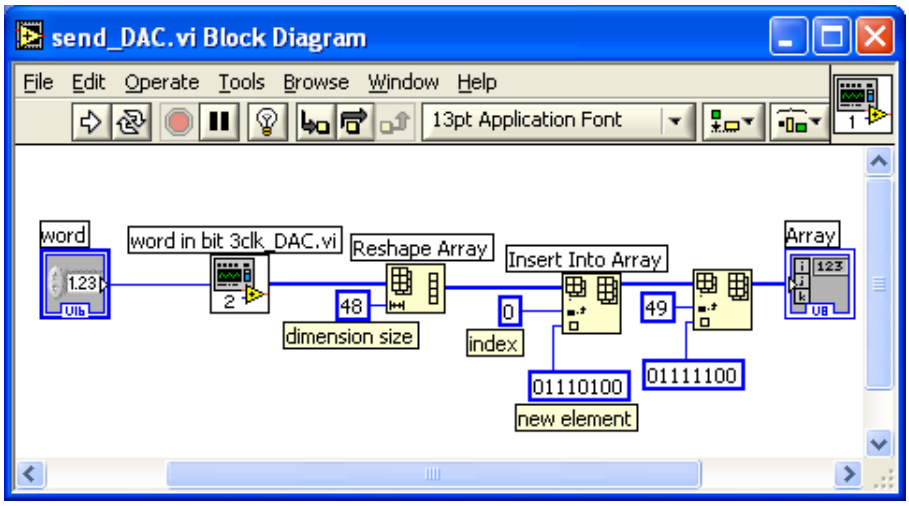


Figura 57-send_DAC

3.3.6.2 SubVI: word in bit 3clk DAC

Questo SubVi è di fondamentale importanza nella realizzazione della scheda poiché regola l'operazione di scrittura nel dispositivo in base alle sue tempificazioni. I dati sono caricati come una parola di 16 bit sotto il controllo del Serial Clock Input, generato dal Master (UM245R). I dati possono, però, essere trasferiti nel dispositivo solamente quando il segnale di Chip Select è basso, in quanto ciò rende attivo l'integrato. Considerando, quindi, le tempificazioni di questo integrato, analizzate nel capitolo 2, è stato realizzato il seguente subvi. La sua implementazione richiede un controllore, indicato col nome di Numeric, in cui inserire i dati da inviare e che viene rappresentato come unsigned word e con formato binario, pur essendo un numero intero, unicamente per permettere all'utente la visualizzazione dei 16 bit da trasferire. Ricordando che i dati vengono inviati serialmente come parole di 16 bit, che un bit è inviato in corrispondenza del bordo di discesa del SCK (nel caso del DAC), che il tutto avviene in corrispondenza del CS basso e che i dati sono trasmessi settando opportunamente i pin DB0-DB7 dell'UM245R; abbiamo collegato il controllore con la funzione LabView "Number To Boolean Array", che permette di convertire un numero intero in un array booleano di 8, 16, o 32 elementi, in base al numero di bit nel numero intero, dove l'elemento più a destra corrisponde al bit meno significativo della rappresentazione binaria del numero intero stesso.

L'array così ottenuto è posto in ingresso alla funzione LabView "Boolean To (0,1)", che converte un valore booleano vero o falso in un intero a 16 bit con un valore di 1 o di 0, rispettivamente. Si ottiene così la trasformazione dalla word da trasferire in un array di 16 bit per poter isolare i singoli bit da trasmettere serialmente. L'array risultante va in ingresso alla funzione LabView "Reverse 1D Array", che inverte l'ordine dei suoi elementi, poiché dobbiamo trasmettere a partire dal bit più significativo.

L'array ottenuto è posto in ingresso a due cicli For. Col primo ciclo For ricavo i singoli bit della word da inviare, ecco perché N è posta pari a 16. Col secondo ciclo For triplico il valore del singolo bit per mantenerlo costante nelle tre scritture, ottenendo un array di 48 elementi rappresentante la word da trasmettere. Inoltre nel segnale da trasmettere, per ogni bit, bisogna settare il SCLK che cambia ad ogni scrittura passando da 0 a 1 nella prima e da 1 a 0 nella seconda. Quindi nel secondo ciclo For sono state necessarie tre operazioni matematiche: una moltiplicazione tra il risultato della divisione tra l'indice "i" del For, che varia tra 0, 1 e 2 essendo N pari a 3, ed una costante pari a 2, ed

una operazione di OR tra il risultato della moltiplicazione, che può essere 0 o 2, ed il singolo bit da trasmettere.

Otteniamo, quindi che se sommo 0 (così rappresentato in binario su 8 bit 00000000) il bit SCLK non cambia, se sommo 2 (così rappresentato in binario su 8 bit 00000010) il bit SCLK cambia riproducendo le sue variazione nelle due scritture.

I valori risultanti sono posti in ingresso alla funzione LabView "Reshape Array", che cambia le dimensioni di un array conformemente al valore di dimensione inserito, nel nostro caso 48. In uscita otteniamo l' indicatore array_out; il Block Diagram e il Front Panel di tale SubVI sono mostrati in figura 58.

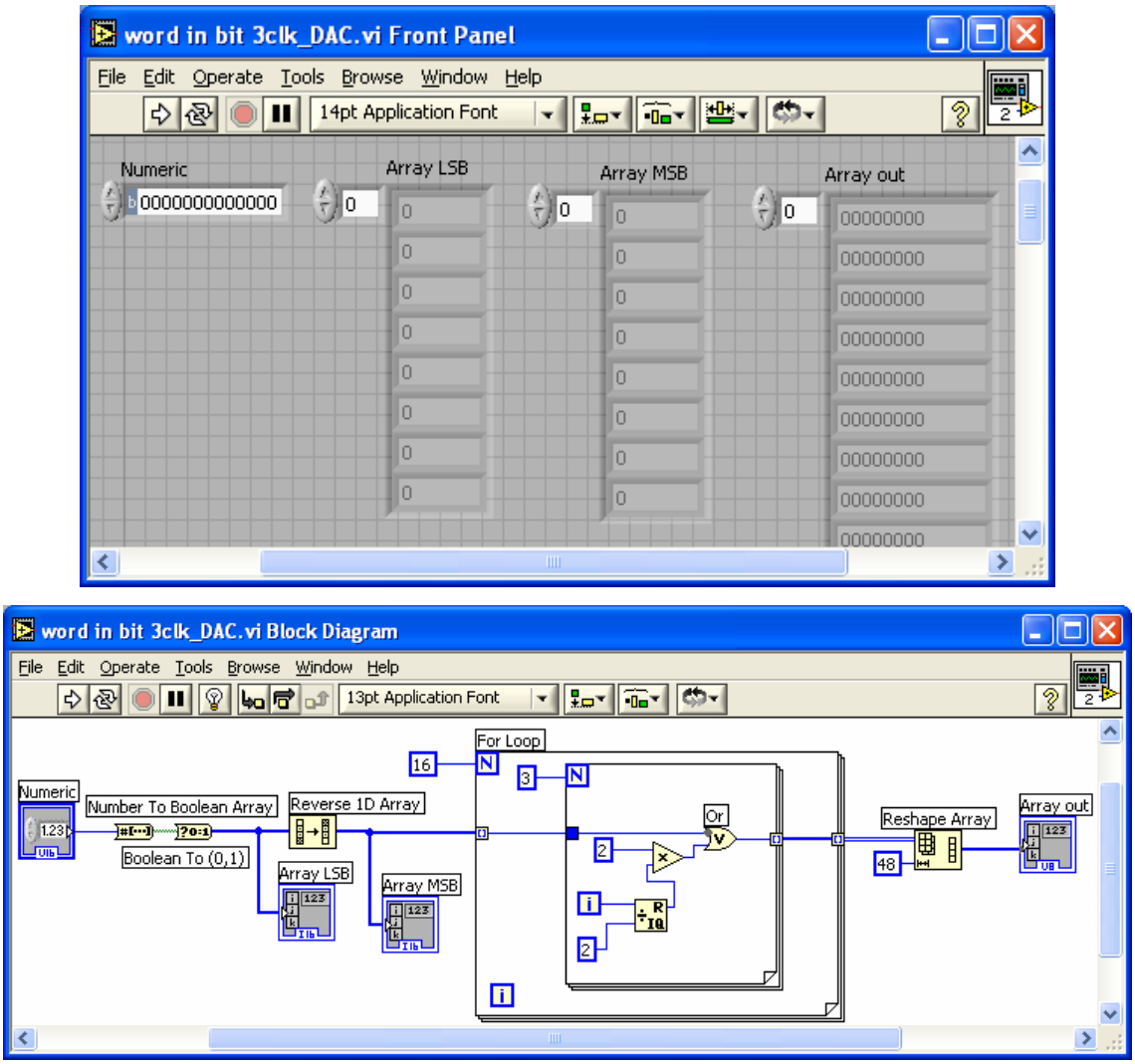


Figura 58- word in bit 3clk_DAC

3.3.7 SubVI: USBIO AnalogRead

Questo subvi effettua la lettura e contemporaneamente la configurazione dell' ADC per la successiva conversione. Esso è stato realizzato tramite un'opportuna concatenazione delle subvi, descritti in questo capitolo e di una delle primitive, citate nel capitolo 2. In particolare, per la realizzazione della concatenazione è stata utilizzata la Flat Sequence Structure, una struttura che permette la sequenzializzazione dei subdiagrammi contenuti in essa, o per meglio dire delle funzioni presenti al suo interno. Consideriamo il primo blocco.

-Preparazione dei dati (il CS viene portato al livello logico basso)

Questo primo sottoblocco utilizza la funzione FT_Write_All_Data.vi, per effettua la scrittura nello slave ADC tramite il master. In particolare, riceve in ingresso l'Handle_in (che mostra una sorta di "indirizzo" identificativo del dispositivo collegato e che sarà usato per accessi successivi), posto all' esterno della Flat Sequence Structure, e un array di costanti costituito da un solo elemento con 8-bit, 01011100 (corrispondono a DB7-DB0). Con questa istruzione viene portato il CS dell' ADC, che corrisponde al bit DB5, al livello logico basso. Passiamo al successivo sottoblocco.

-Lettura, scrittura, scrittura

Questo sottoblocco utilizza un ciclo for, con il terminale di conteggio N=16, in cui entra un bit per volta per un totale di 16 bit. All' interno di questo ciclo troviamo un' altra Flat Sequence Structure, costituita da due sottoblocchi. Nel primo sottoblocco viene effettuata la lettura dei dati dall' ADC, attraverso la funzione FT_GET_BIT_MODE.vi. In particolare questa riceve in ingresso l'Handle_in e fornisce in uscita due indicatori FT_STATUS_GET_BIT_MODE, che fornisce lo stato della funzione e un array Bit Mode. Nel secondo sottoblocco viene settato il clock in due scritture,nel senso che nella prima scrittura viene portato il clock da 0 a 1, nella seconda scrittura viene portato da 1 a 0, e vengono inviate alcune informazioni relative al formato dei registri. In quanto l'ADC, come citato nel capitolo 2, deve ricevere in ingresso 16-bit di cui i primi 8 bit contengono le informazioni generali per la configurazione (sono bit di controllo), in particolare i primi 4-bit (D7-D4), più significativi, contengono l' indirizzo del canale analogico (o ingresso analogico) da cui viene prelevato il segnale; i successivi 2-bit (D3-D2) consentono di selezionare la lunghezza dei dati in uscita, un altro bit successivo (D1) consente di selezionare il formato dei dati in uscita, ovvero stabilire se la sequenza dei bit in uscita ha inizio dal bit più significativo o da quello meno

significativo; l'ultimo bit (D0) consente di scegliere il tipo di modalità unipolare oppure bipolare. Per cui la sequenza da trasmettere sarà xxxx1100xxxxxxxx. In cui 1100 sono i valori dei bit da noi scelti, che non possono essere cambiati, mentre le x sono i bit che possono variare.

La sua implementazione è stata effettuata utilizzando il subvi USBIO Formato_registri_ADC.vi in ingresso ad una funzione di LabView, "Boolean Array to Number" che mi converte i valori booleani in valori numerici, posti all' esterno del ciclo FOR . L'uscita è posta in ingresso all' operatore Add insieme ad un array di due vettori di 8 bit rispettivamente, 01011110 (per portare il clock da 0 a 1) e 01011100 (per portare il clock da 1 a 0), per far sì che la scrittura avvenga contemporaneamente alla lettura.

L' uscita dall' Add è quindi posta in ingresso alla FT_Write_All_Data, che consente la scrittura nell' ADC. In uscita abbiamo FT_Status_A e Bytes Written_A.

-**Fine letture e scritture** (il CS viene portato al livello logico alto)

Questo sotto blocco utilizza la funzione FT_Write_All_Data.vi, la quale effettua la scrittura nel dispositivo. In particolare, riceve in ingresso l'Handle_in, rappresentato da un intero di 12-bit, posto all' esterno della flat sequenze structure, e un array di costanti costituito da un solo elemento con 8-bit, 01111110 (che corrispondono a DB7-DB0); in effetti con questa scrittura il CS dell' ADC , che nella sequenza corrisponde a DB5, viene portato al livello logico alto; di conseguenza vengono terminate le operazioni di lettura e scrittura.

In uscita dalla Flat Sequenze Structure esterna abbiamo un indicatore Handle out, che mostra una sorta di "indirizzo" identificativo del dispositivo collegato e che sarà usato per accessi successivi; il Front Panel ed il Block Diagram sono mostrati in figura 59.

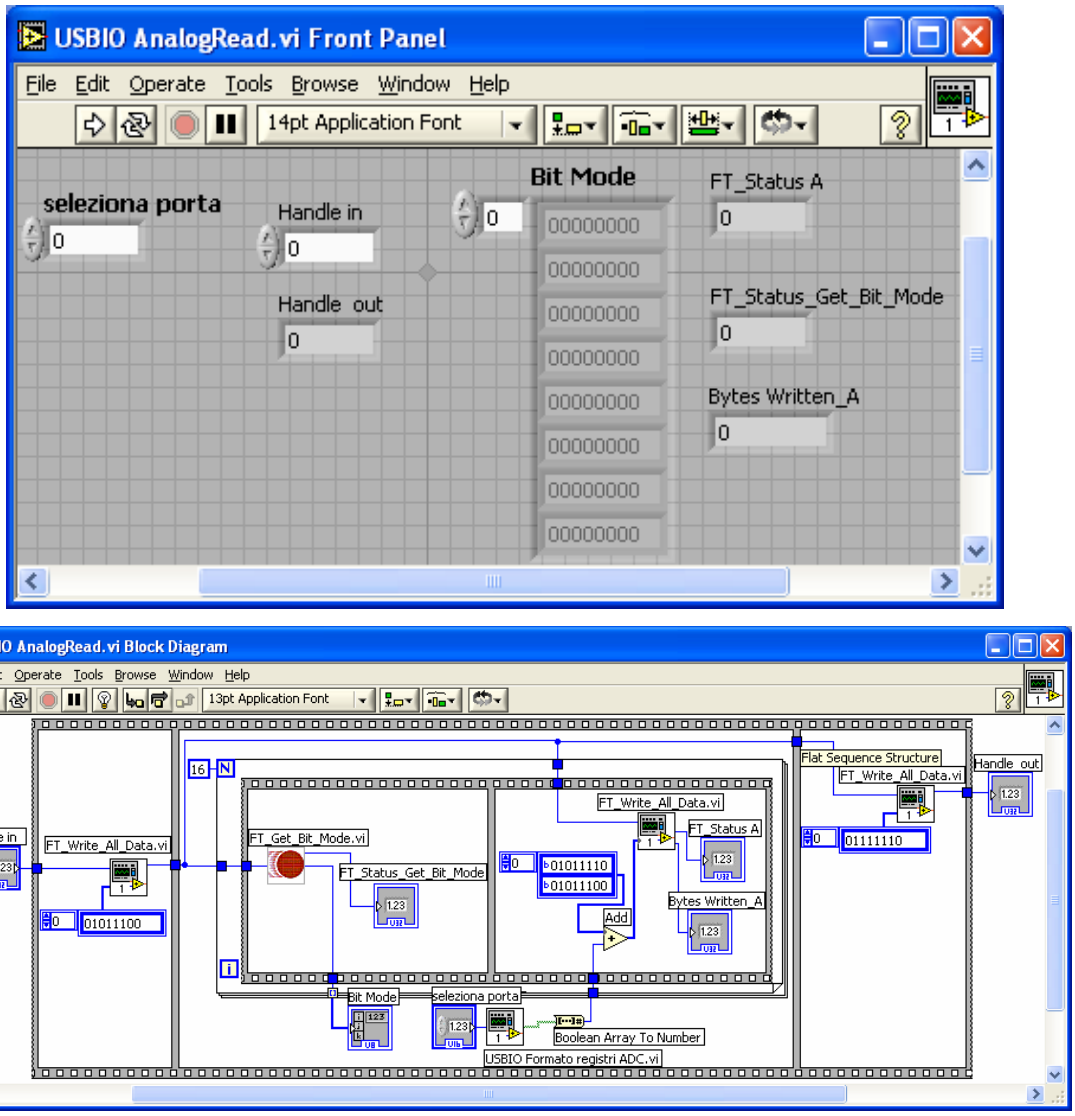


Figura 59- USBIO AnalogRead

3.3.7.1 USBIO Formato registri ADC

Questo SubVI configura il "registro dei dati in ingresso" dell'ADC.

Come illustrato nel capitolo 2, l'ADC deve ricevere in ingresso 16-bit di cui i primi 8 bit contengono le informazioni generali per la configurazione (sono bit di controllo), in particolare i primi 4-bit (D7-D4), più significativi, contengono l' indirizzo del canale analogico (o ingresso analogico) da cui viene prelevato il segnale; i successivi 2-bit (D3-D2) consentono di selezionare la lunghezza dei dati in uscita, un altro bit successivo (D1) consente di selezionare il formato dei dati in uscita, ovvero stabilire se la sequenza dei bit in uscita ha inizio dal bit più significativo o da quello meno significativo; l'ultimo bit (D0) consente di scegliere il tipo di modalità unipolare oppure bipolare. Per cui la sequenza da trasmettere sarà xxxx1100xxxxxxxx. In cui 1100 sono i valori dei bit da noi scelti, che non possono essere cambiati, mentre le x sono i bit che possono variare.

La sua implementazione è stata realizzata utilizzando un controllore Seleziona Porta posto in AND con una costante di 16 bit, 0000000000001111, realizzando una mascheratura. La cui uscita è posta in ingresso all' operatore di moltiplicazione con un valore costante 4096, corrispondente a 2^12, questa operazione mi consente di effettuare uno shift a sinistra di 12 posizioni. Successivamente abbiamo l'operatore OR, che riceve in ingresso l' uscita della moltiplicazione insieme alla costante 0000110000000000; l'uscita è posta in ingresso ad una funzione di LabView "Number to boolean array" (che effettua la conversione di un numero in un array di valori booleani); la cui uscita è posta sia in ingresso ad un'altra funzione di LabView "Array size"(che fornisce in uscita le dimensioni dell' array in ingresso), sia in ingresso alla funzione "Reverse 1D Array"(che inverte l'array ovvero l'ultimo elemento è posto al primo posto); essa mi consente di avere al primo posto dell' array il bit più significativo. In uscita ottengo un indicatore array e un indicatore number; il Front Panel ed il Block Diagram sono mostrati in figura 60.

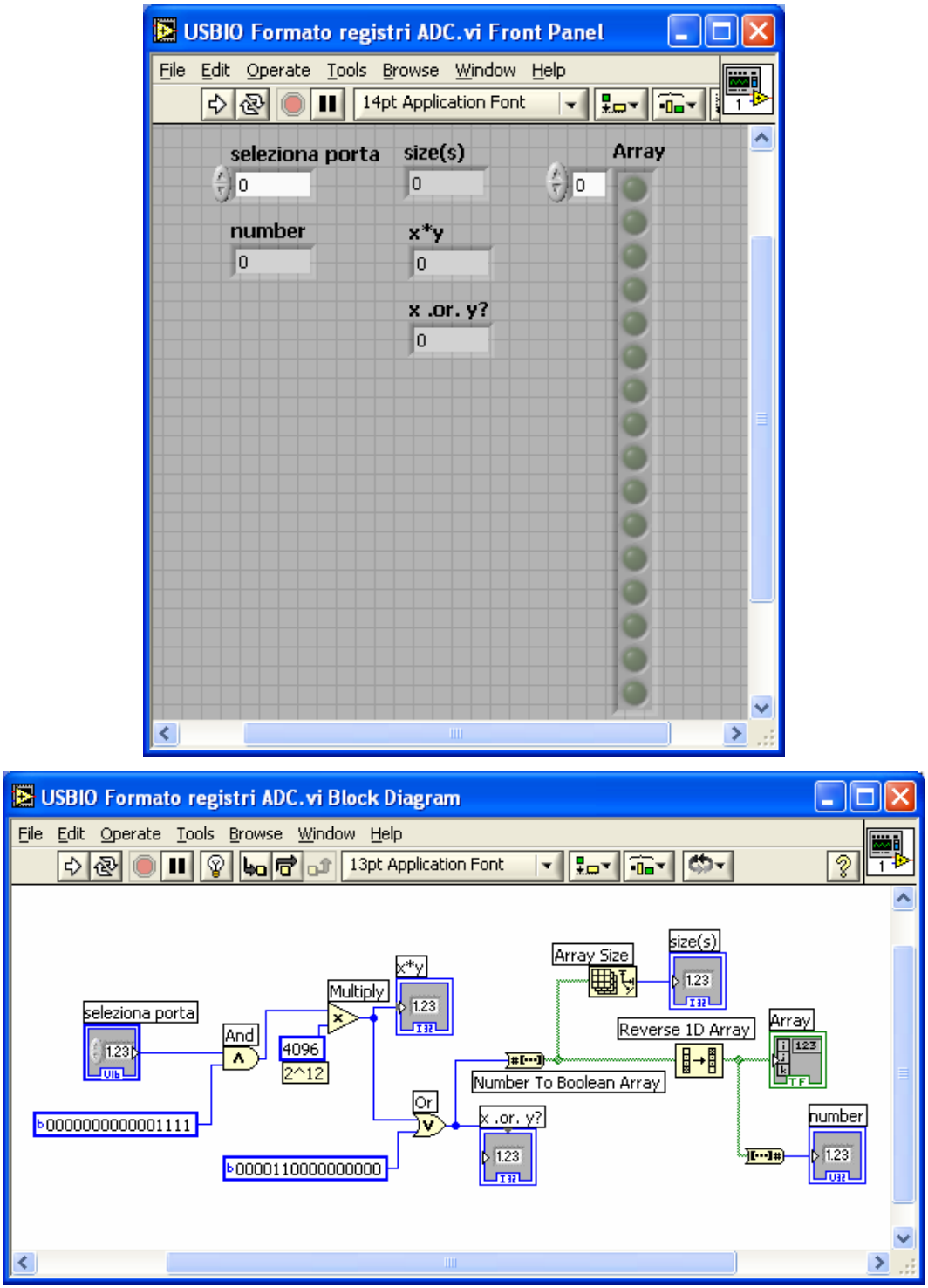


Figura 60- USBIO Formato registri ADC

3.3.8 SubVI: USBIO Close

Questo subvi chiude la comunicazione con un dispositivo precedentemente aperto.

La sua implementazione è stata realizzata utilizzando la funzione FT_CLOSE_DEVICE.vi, che chiude la comunicazione con un dispositivo precedentemente aperto. In particolare, questa funzione riceve in ingresso il controllore Handle e fornisce in uscita l' indicatore FT_Status_Close, che fornisce lo stato della funzione; il Front Panel ed il Block Diagram sono mostrati in figura 61.

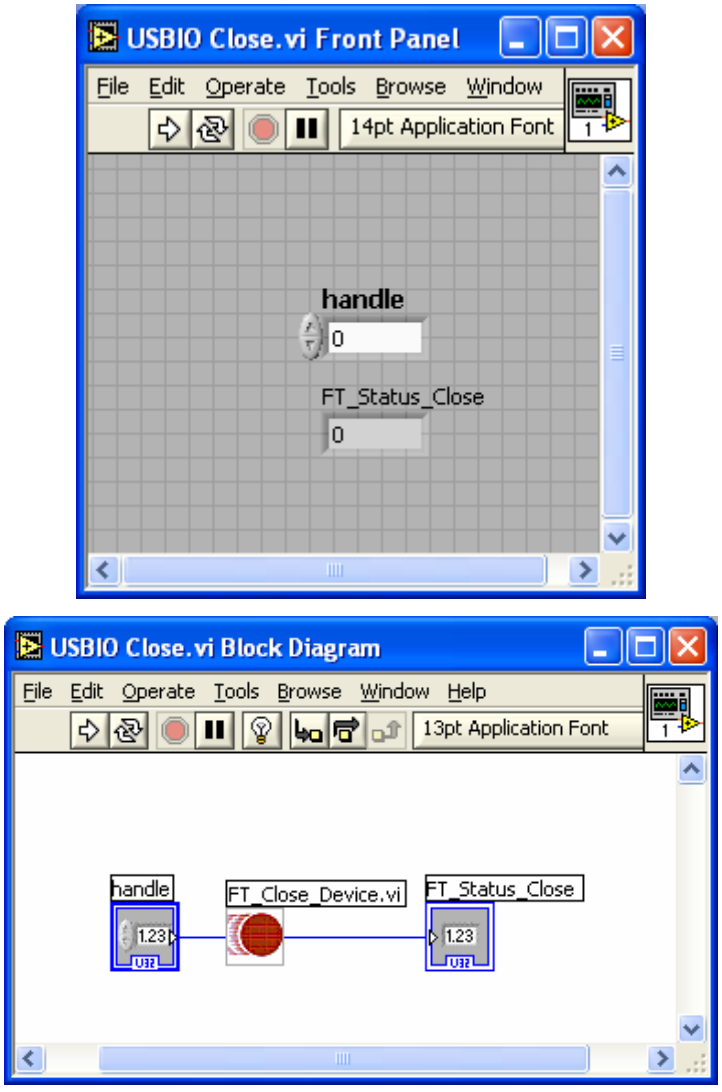


Figura 61- USBIO Close

3.3.9 SubVI: Byte in bit 3clk

Questo SubVi è di fondamentale importanza nella realizzazione della scheda poiché regola l'operazione di scrittura e lettura nel Port Expander in base alle sue tempificazioni. I dati sono caricati come un array di 8 bit sotto il controllo del Serial Clock Input, generato dal Master (UM245R). Essi possono,però, essere trasferiti nel dispositivo solamente quando il segnale di Chip Select è basso, in quanto ciò rende attivo l'integrato. Considerando , quindi le tempificazioni di questo integrato, analizzate nel capitolo 2, è stato realizzato questo SubVI. La sua implementazione richiede un controllore, indicato col nome di Numeric, in cui inserire i dati da inviare, posto in ingresso alla funzione LabView " Number To Boolean Array", che permette di convertire un numero intero in un array booleano di 8, 16, o 32 elementi, in base al numero di bit nel numero intero, dove l'elemento più a destra corrisponde al bit meno significativo della rappresentazione binaria del numero intero stesso.

L'array così ottenuto è posto in ingresso alla funzione LabView "Boolean To (0,1)", che converte un valore booleano vero o falso in un intero a 8 bit con un valore di 1 o di 0, rispettivamente. L'array risultante, di 8 bit, va in ingresso alla funzione LabView "Reverse 1D Array", che inverte l'ordine dei suoi elementi, poiché dobbiamo trasmettere a partire dal bit più significativo.

L'array ottenuto è posto in ingresso a due cicli For dato che il passaggio del singolo bit lo si ottiene in tre scritture in cui il suo valore si deve mantenere costante. Col primo ciclo For ricavo i singoli bit dell'array da inviare, ecco perché il terminale di conteggio N è posta pari a 8. Col secondo ciclo For triplico il valore del singolo bit per mantenerlo costante nelle tre scritture, ottenendo un array di 24 elementi rappresentante l'array da trasmettere. Inoltre nel segnale da trasmettere, per ogni bit, bisogna settare anche il clock SCK che cambia ad ogni scrittura passando da 0 a 1 nella prima e da 1 a 0 nella seconda. Quindi nel secondo ciclo For sono state necessarie alcune operazioni: una divisione tra una costante pari a 2 e l'indice "i" (terminale di iterazione) del For, (che varia tra 0, 1 e 2 essendo N pari a 3; un' operazione di moltiplicazione tra il risultato della divisione ed una costante pari a 2; infine l' operazione OR tra il risultato della moltiplicazione, che può essere 0 o 2, ed il singolo bit da trasmettere.

Otteniamo, quindi che se sommo 0 (così rappresentato in binario su 8 bit 00000000) il bit SCK non cambia, se sommo 2 (così rappresentato in binario su 8 bit 00000010) il bit SCK cambia riproducendo le sue variazione nelle due scritture.

I valori risultanti sono posti in ingresso alla funzione LabView “Reshape Array”, che cambia le dimensioni di un array conformemente al valore di dimensione inserito, nel nostro caso 24; in uscita abbiamo un indicatore array 3; il Front Pannel ed il Block Diagram di tale SubVI sono mostrati in figura 62.

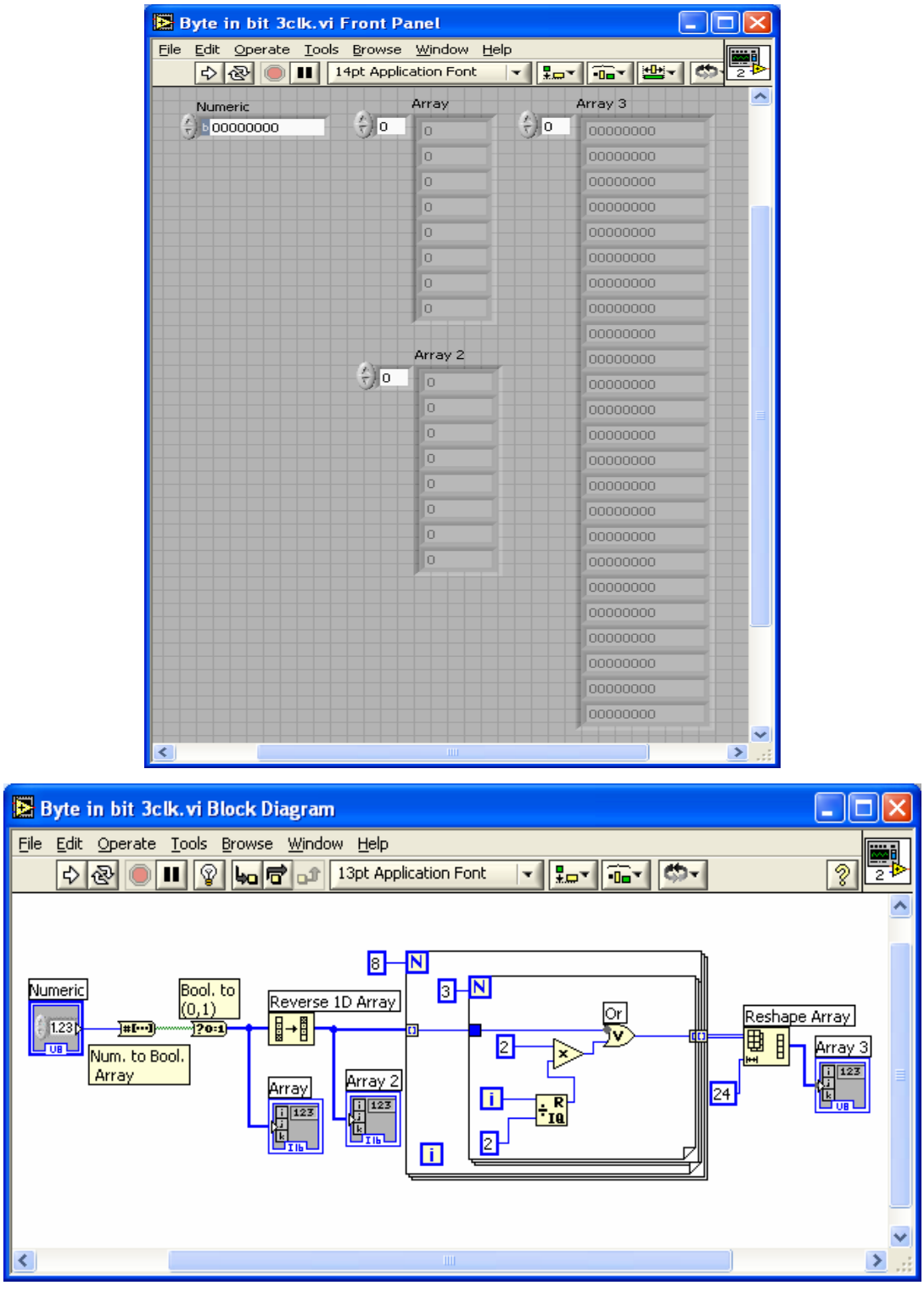
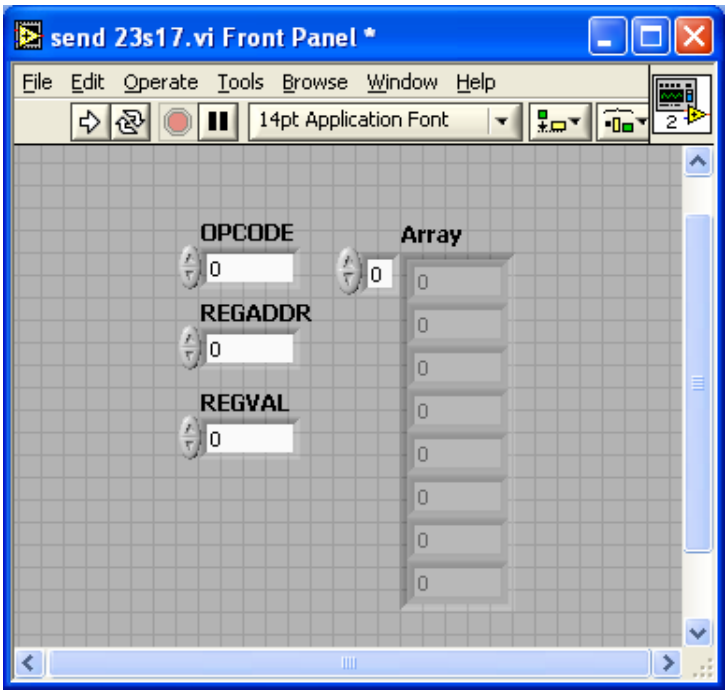


Figura 62- Byte in bit 3 clk

3.3.10 SubVI: Send23s17

Vediamo come è stato realizzato il SubVI send23s17.vi, il quale consente la realizzazione dei dati effettivamente inviati al Port Expander. La sua implementazione richiede tre controllori, rispettivamente OPCODE, REGADDR, REGVAL, in cui verranno inseriti i dati dei relativi registri del Port Expander per la loro attivazione. Ognuno di essi va in ingresso al SubVI Byte in bit 3clk.vi, che mi fornisce in uscita un array di 24 elementi. Questi tre array di 24 bit, sono modificati tramite la funzione di LabView "Buld Array", che permette di concatenare più array in un unico array. L' array, così ottenuto, è posto in ingresso alla funzione di LabView "Reshape Array", che cambia le dimensioni di un array conformemente al valore di dimensione inserito, nel nostro caso 72. L'array ottenuto è ulteriormente modificato tramite la funzione LabView "Insert Into Array", che permette d'inserire un nuovo elemento nell'array nella posizione indicata. Difatti inseriamo due nuovi elementi, uno in testa e l'altro in coda, che consentono rispettivamente di impostare il CS ad un livello logico basso per dare inizio alla trasmissione e ad un livello logico alto per concludere l'operazione. Questo è il SubVI realizzato per costruire l'effettivo segnale inviato; il Front Pannel ed il Block Diagram di tale SubVI sono mostrati in figura 63:



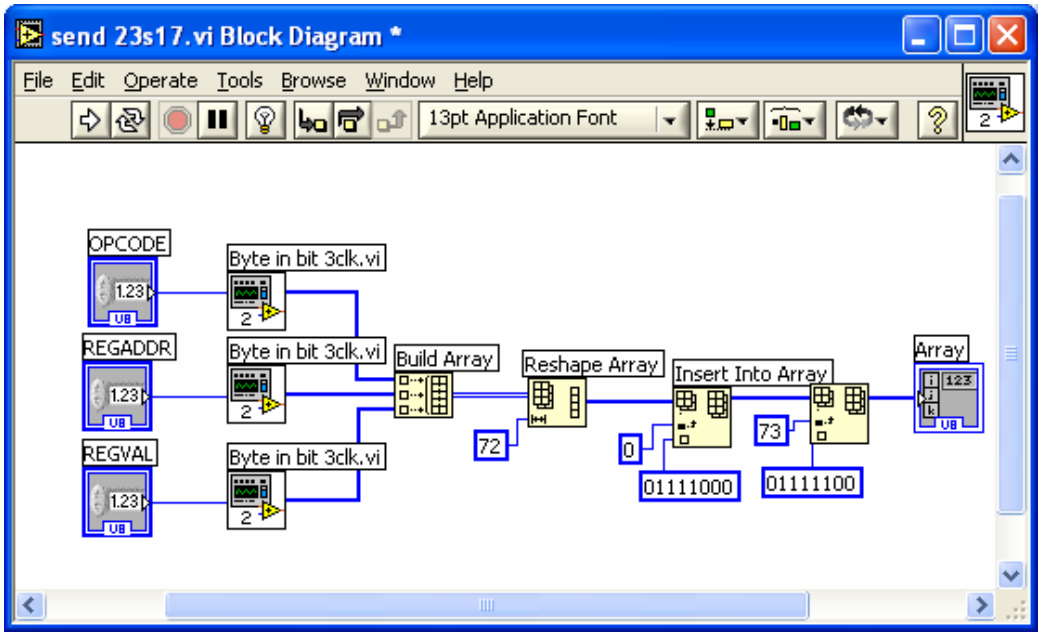
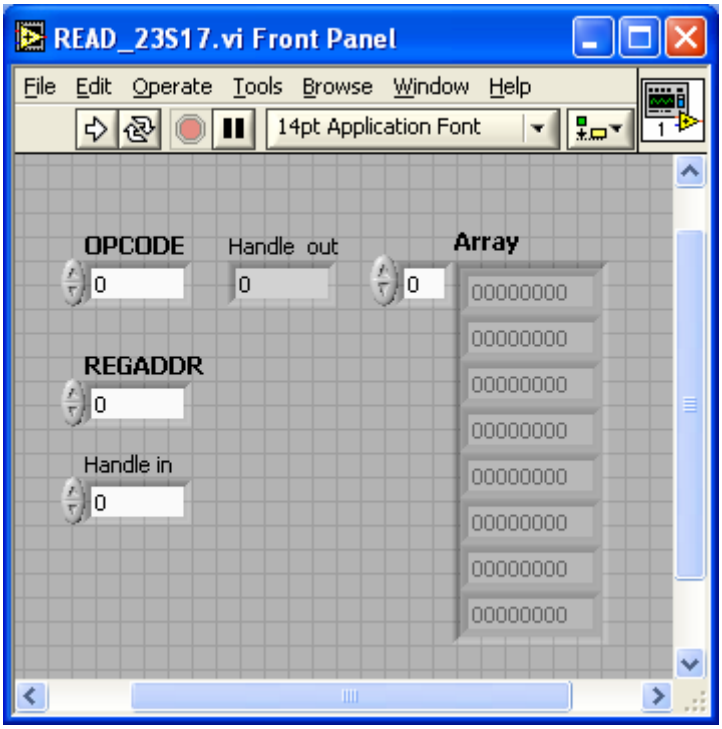


Figura 63- send 23s17

3.3.11 SubVI: READ_23s17

Vediamo come è stato implementato il SubVI READ_23s17.vi, il quale consente la lettura del MCP23S17 da parte del UM245R. La sua implementazione richiede tre controllori, rispettivamente abbiamo l' Handle, l' OPCODE e il REGADDR, in questi ultimi due verranno inseriti i dati dei relativi registri del Port Expander per la loro attivazione. Essi vanno in ingresso al SubVI 1_fase_read_23s17 OPCODE-REGADDR.vi, che mi consente di attivare i registri del Port Expander coinvolti nella lettura; la funzione fornisce in uscita un Handle posto in ingresso alla SubVI CLOCK_fase_read_23s17.vi, che regola l'operazione di lettura nel dispositivo in base alle sue tempificazioni; la funzione mi fornisce in uscita due indicatori un Handle e un Array; il Front Pannel ed il Block diagramm sono mostrati in figura 64.



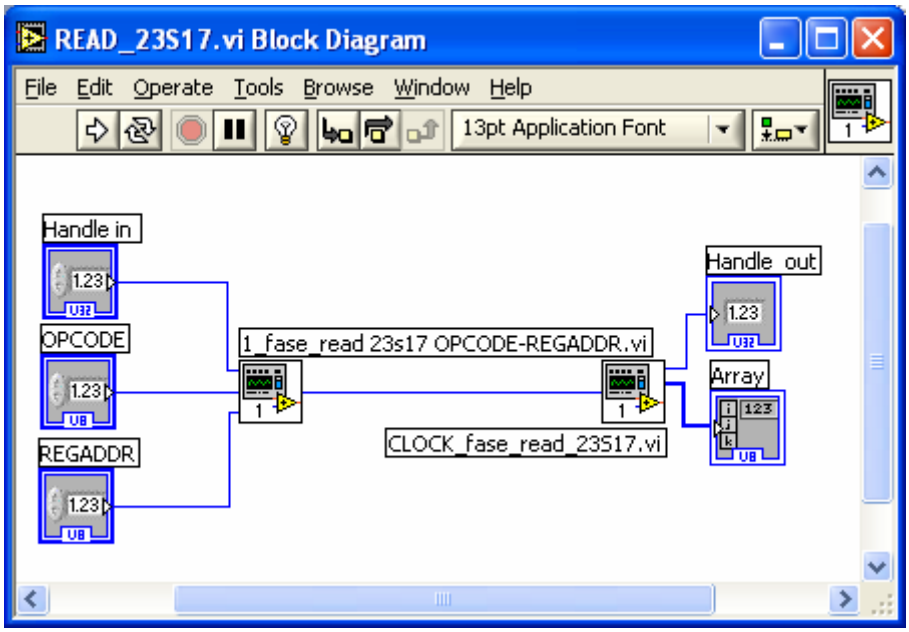


Figura 64- READ_23s17

3.3.12 SubVI: 1_fase_read 23s17

Vediamo come è stato implementato il SubVI 1_fase_read_23s17 OPCODE-REGADDR.vi, il quale consente di attivare i registri del Port Expander coinvolti nell' operazione di lettura. La sua implementazione richiede due controllori OPCODE e REGADDR, in cui verranno inseriti i dati dei relativi registri del Port Expander per la loro attivazione. Ognuno di essi va in ingresso al SubVI Byte in bit 3clk.vi, che regola l'operazione di lettura e scrittura nel MCP23S17 in base alle sue tempificazioni; la funzione mi fornisce in uscita un array di 24 elementi. Questi due array di 24 bit sono modificati tramite la funzione di LabView " Built Array", che consente di concatenare più array in un unico array. L'array così ottenuto è posto in ingresso alla funzione LabView "Reshape Array", che cambia le dimensioni dell' array conformemente al valore della dimensione inserita, nel nostro caso 48. L' array ottenuto è ulteriormente modificato tramite la funzione LabView "Insert Into Array", che permette di inserire un nuovo elemento nell' array nella posizione indicata. Difatti inseriamo un nuovo elemento in testa che ci consente di portare il CS ad un livello logico basso per dare inizio alla trasmissione. L'uscita va sia in ingresso alla funzione LabView "Array Size", sia alla FT_Write_Byte_Data.vi, insieme all' Handle; consentendomi la scrittura sul UM245R. Inoltre il controllore Handle in è collegato all'indicatore "Handle out", in

modo tale da ritrovarlo in uscita al SubVI; il Front Pannel ed il Block Diagram sono mostrati in figura 65.

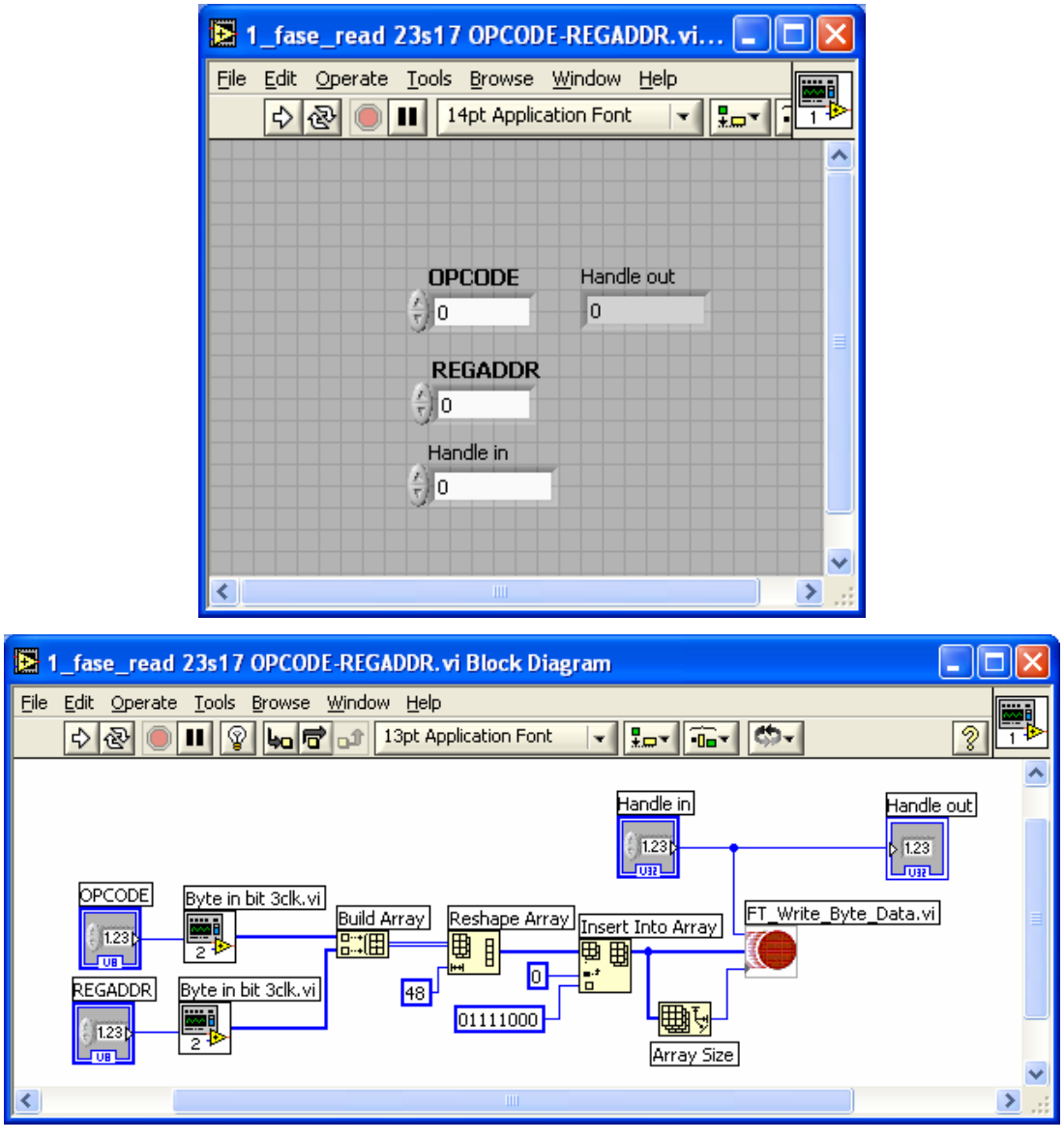


Figura 65- 1_fase_read 23s17 OPCODE-REGADDR

3.3.13 SubVI: CLOCK_fase_read_23S17

Questo SubVi è di fondamentale importanza nella realizzazione della scheda poiché regola l'operazione di lettura dei dati dal Port Expander in base alle sue tempificazioni. I dati vengono caricati come un array di 8 bit sotto il controllo del Serial Clock Input, generato dal Master (UM245R); inoltre essi possono essere letti dal master solamente quando il segnale di Chip Select è basso, in quanto ciò rende attivo l'integrato. La sua implementazione richiede una Flat Sequenze Structure per rendere sequenziali i seguenti sottoblocchi:

- *Lettura,scrittura,scrittura*

  Tenendo presente la tempificazione del Port Expander, commentata nel Capitolo 2, l'implementazione richiede un controllore Handle in , in ingresso alla flat sequenze esterna e al ciclo FOR, con il terminale di conteggio N=8, nel quale entra un bit alla volta. All' interno del ciclo FOR abbiamo un' altra flat sequenze structure costitutita da due blocchi. Nel primo sottoblocco viene effettuata la lettura dei dati nel Port Epander attraverso la funzione FT_Get_Bit_Mode.vi. Essa riceve in ingresso l' Handle in, prodotto dalla FT_Open_Device.vi e restituisce in uscita due indicatori FT_Status e Array.

  Nel secondo sottoblocco viene settato il clock in due scritture; nella prima viene portato da 0 a 1 , nella seconda viene portato da 1 a 0. La sua implementazione richiede un controllore Handle in e due valori costanti , rappresentati in binario con 8 bit 01111010 (per portare il clock da 0 a 1) e 01111100, (per portare il clock da 1 a 0) in ingresso alla FT_Write_All_Data.vi; la funzione fornisce in uscita due indicatori FT_Status e Bytes Written.

- *Scrittura dei dati*

  Nel secondo blocco viene effettuata la scrittura dei dati nel Port Expander. La sua implementazione richiede un valore costante di 8 bit 00111111, in quanto come illustrato nel capitolo 2, nel Port Expander, al bit 1 è associato 5V, mentre al bit 0 è associato 0V.

  La FT_Write_All_Data.vi riceve in ingresso questi 8-bit e fornisce in uscita la FT_Status, che fornisce lo stato della funzione e i Bytes Written, indica il numero di byte effettivamente scritti.

In uscita dalla Flat Sequenze Structure esterna abbiamo un indicatore Handle out, che mostra una sorta di “indirizzo” identificativo del dispositivo collegato e che sarà usato per accessi successivi; il Front Pannel ed il Block diagramm sono mostrati in figura 66.

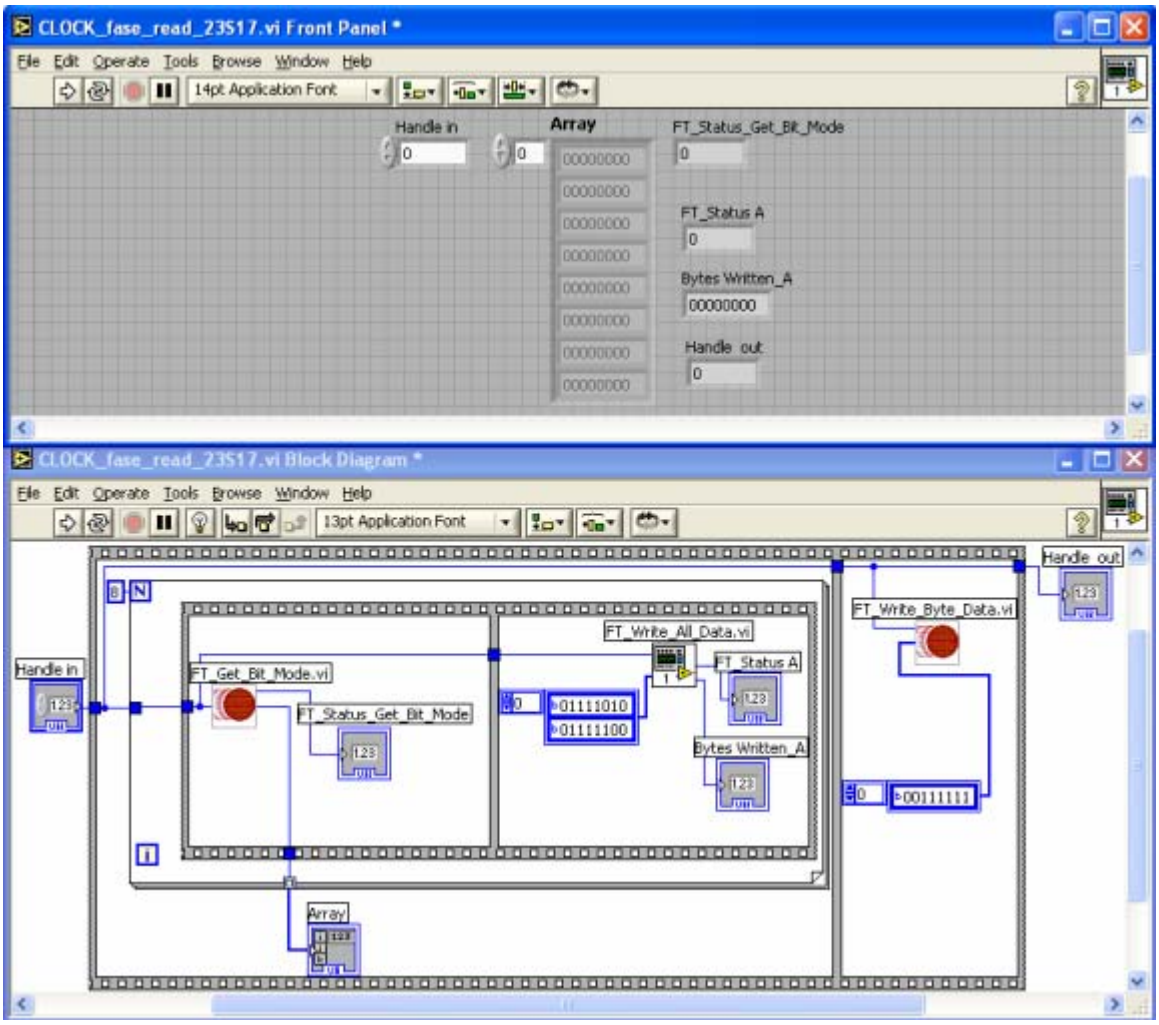


Figura 66- CLOCK_fase_read_23s17

3.3.14 SubVI: FT_Write_All_Data

Questo SubVi consente di scrivere all' interno del dispositivo UM245R. Esso è stato realizzato per semplificare la rappresentazione dei SubVI, nei quali è richiesta questa operazione di scrittura. La sua implementazione richiede in ingresso due controllori, un Handle in e un Array, quest'ultimo insieme alle sue dimensioni (ottenute in uscita dall' array size), sono posti in ingresso alla FT_Write_Byte_Data.vi, la quale fornisce in uscita due indicatori FT_Status e Bytes Written. Inoltre il controllore Handle in è collegato all' indicatore Handle out, in modo tale da ottenerlo anche in uscita; il Front Pannel ed il Block diagramm sono mostrati in figura 67.

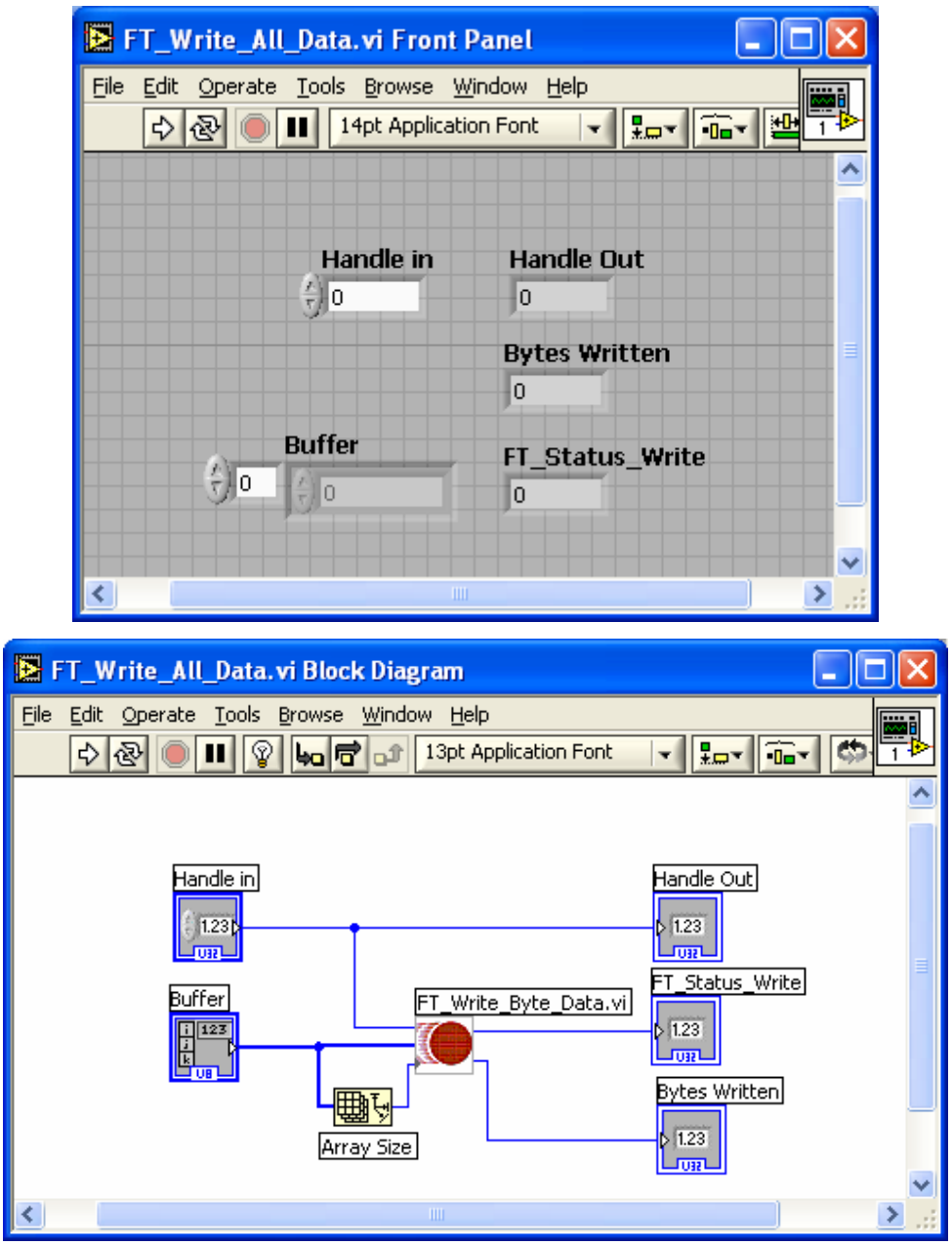


Figura 67- FT_Write_All_Data

3.3.15 SubVI: Test_23S17 (Port Expander)

Questo SubVI è stato realizzato per verificare il corretto funzionamento dell' integrato MCP23S17 in fase di lettura e scrittura. E' stato ottenuto collegando alcuni dei SubVI, analizzati in questo capitolo, in particolare quelli relativi all' integrato considerato; il Front Pannel ed il Block diagramm sono mostrati in figura 68.

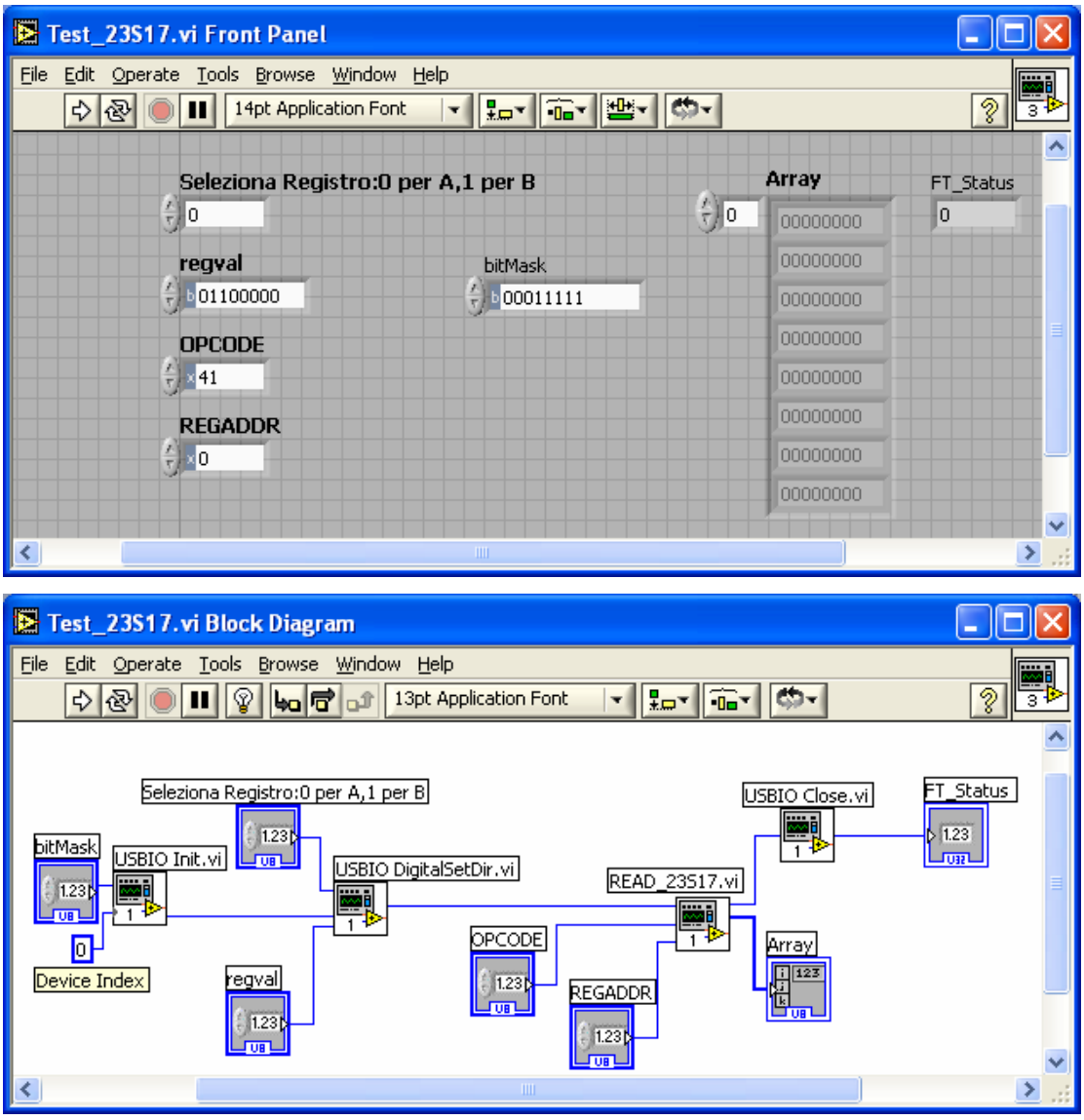


Figura 68- Test_23s17

3.3.16 SubVI: Test DAC

Questo SubVI è stato realizzato per verificare il corretto funzionamento dell' integrato DAC in fase di scrittura. E' stato realizzato collegando alcuni dei SubVI , descritti in questo capitolo, in particolare quelli relativi all' integrato considerato; il Front Pannel ed il Block diagramm sono mostrati in figura 69.

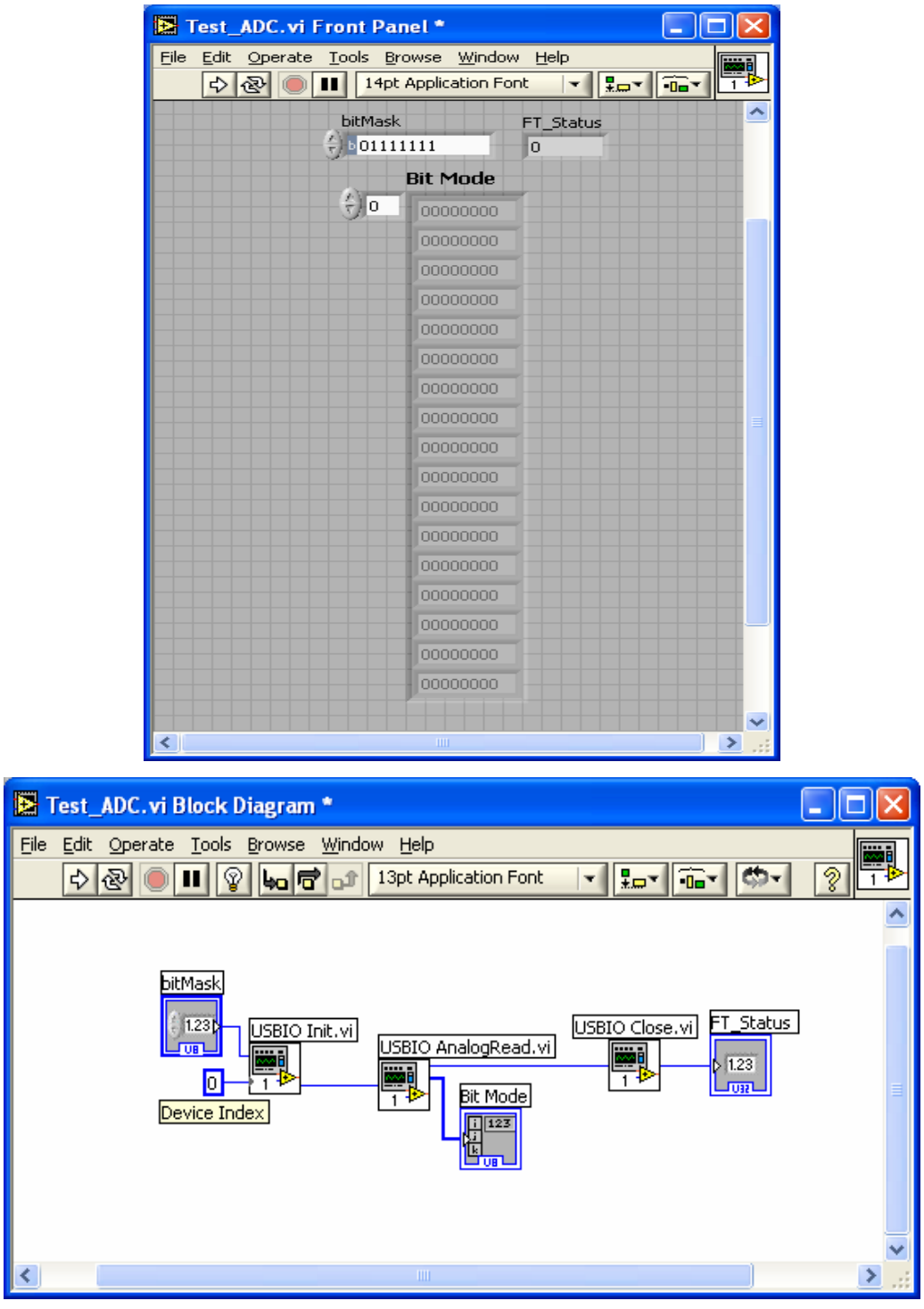


Figura 69- Test_ADC

3.3.17 Vecchie e nuove funzionalità

Nella versione precedente del laboratorio "remoto", per effettuare esperienze monoutente, era stato realizzato un programma in Labview (VI), che riproduceva il pannello frontale di un oscilloscopio. Al cui interno erano state inserite le primitive fornite dalla scheda commerciale (NI_PCI_6040E). Esse sono state sostituite, nella versione attuale del laboratorio, con alcuni dei SubVI realizzati per la scheda I/O A/D, per consentire l'interazione con il circuito caotico di Chua ed RLD.

I SubVI utilizzati sono riportati di seguito:

- USBIO Init.vi
- USBIO DigitalSetDir.vi
- USBIO DigitalWrite.vi
- USBIO Close.vi

Questi sono concatenati tra di loro nel seguente modo: il SubVi USBIO_Init.vi, che consente da PC di settare i pin da utilizzare dei due dispositivi UM245R ed MCP23S17, riceve in ingresso il controllore bit mask, nel quale i bit sono configurati singolarmente per l' input e per l' output; la funzione fornisce in uscita l' handle out, il quale va in ingresso insieme a due costanti, rispettivamente una con valore 1 (per indicare che stiamo operando con la PortB del Port Expander), l' altra di 8-bit, 11111111 (che ci consente di configurare tutti i bit come uscite) al subvi USBIO DigitalSetDir.vi, che consente la scrittura della configurazione dei singoli pin come i/o nei registri IODIRA e IODIRB; la funzione USBIO DigitalSetDir.vi ha in uscita un indicatore Handle. Il quale, come controllore, è posto in ingresso insieme a due costanti, rispettivamente una con valore 1 (per indicare che lavoro con la PortB), l' altra di 8 bit, 00000000 (che ci consente di configurare tutti i bit come ingressi in modo tale da poter operare con i relé) al USBIO DigitalWrite.vi, che effettua la scrittura nei registri OLATA e OLATB dei bit in uscita dai rispettivi lati del dispositivo; la funzione USBIO DigitalWrite.vi ha in uscita un indicatore Handle. Esso va in ingresso insieme a due costanti, rispettivamente una con valore 0 (per indicare che stiamo operando con la PortA del Port Expander), l' altra di 8-bit, 11111111 (che ci consente di configurare tutti i bit come uscite) al subvi USBIO DigitalSetDir.vi; la funzione ha in uscita un controllore Handle posto in ingresso insieme ad una costante pari a 0 (indica che lavoro con la PortA), ed insieme

al valore, di 8 bit ottenuto dal VI, al USBIO DigitalWrite.vi; la funzione ha in uscita un indicatore Handle posto in ingresso alla USBIO_CLOSE, che consente di chiudere la comunicazione con un dispositivo precedentemente aperto; la quale fornisce in uscita l' indicatore FT_Status_CLOSE, come mostrato in figura 70.

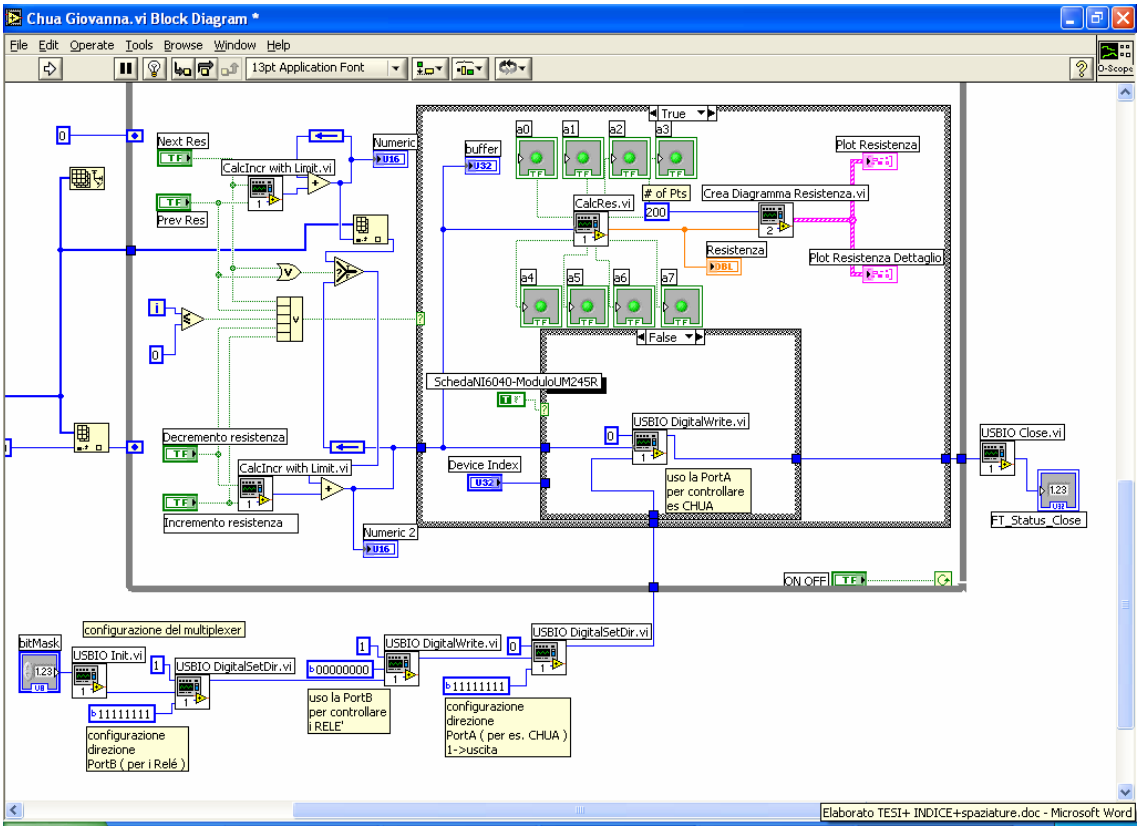


Figura 70- VI: Oscilloscopio digitale

Conclusioni

La scheda I/O A/D presentata in questo elaborato è stata realizzata per consentire, nell' ambito del laboratorio "remoto", l'estensione ad altri esperimenti ed effettuare la selezione in remoto dell'esperimento di interesse, tramite l'utilizzo della scheda ausiliaria a Relé. Oltre alla realizzazione hardware, ovvero alla scelta dei componenti e del loro collegamento, sono state realizzate dal punto di vista software, dei SubVI, che costituiscono l'interfaccia software della scheda I/O A/D. Si presentano come i driver che vengono forniti con le schede professionali e implementano le funzionalità di configurazione e controllo della scheda. Alcuni di essi sono stati inseriti nel VI che riproduce il pannello frontale dell' oscilloscopio digitale in modo tale da consentire l'interazione in remoto del circuito di Chua ed RLD. Il risultato finale è stato il raggiungimento di una versione, funzionante in tempo reale, di un sistema elettronico per l' acquisizione ed elaborazione dei dati degli esperimenti per il PC. Dall' analisi dei dati raccolti è risultato che la scheda è capace di lavorare rispettando le specifiche. Con il collegamento al PC mediante interfaccia USB, si è ottenuto una semplificazione nel cablaggio. Infine c'è la possibilità di espandere ulteriormente la scheda.

E' importante osservare che la scheda è stata realizzata utilizzando dei componenti forniti gratuitamente da società produttrici, ma che presentano, comunque, un costo molto basto. L' unico componente acquistato è l'UM245R. Nel complesso la scheda ha un costo molto ridotto rispetto a quello di una scheda commerciale (possiamo valutarla intorno ai 30, 40 euro). Per quanto riguarda il laboratorio "remoto", il risultato finale è stato il raggiungimento di una versione estesa a più esperimenti e la possibilità di selezionare l'esperimento di interesse.

Bibliografia

[1] "Remotizzazione su web di esperimenti su circuiti caotici";
Dr. Dario Acanfora (Elaborato di laurea).

[2] "Realizzazione di un dimostratore didattici per circuiti caotici";
Ing. Marco Colandrea (Tesi di laurea).

[3] Realizzazione di un circuito caotico di Chua;
Ing.Giovanni Amato (Tesi di laurea).

[4] www.wikipedia.org

[5] www.usb.org

[6] www.ni.com;

[7] PCI6040 NI Data Sheet

[8] PCI5102 NI Data Sheet

[9] FTDI-FT245R Data Sheet

[10] FTDI-UM245R Data Sheet

[11] www.ftdichip.com

[12] MCP23S17 Microchip Data Sheet

[13] TLV5616 Texas Instruments Data Sheet

[14] TLC2543 Texas Instruments Data Sheet

[15] TL431 Texas Instruments Data Sheet

[16] ULN2003A Texas Instruments Data Sheet

[17] HM4101F Hong Mei Relay Data Sheet

[18] M. Bertocco "Introduzione al LabView" Università di Padova Facoltà di Ingegneria