ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SECONDA FACOLTA' DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA MECCANICA

TESI DI LAUREA SPECIALISTICA
in
Controllo dei motori a combustione interna LM

IMPLEMENTAZIONE DI UN SISTEMA HARDWARE IN THE LOOP SU PIATTAFORMA VERISTAND

CANDIDATO

Francesco Dall'Acqua

RELATORE:

Chiar.mo Prof. Enrico Corti

Anno Accademico 2011/12

Sessione I

Indice generale

Introduzione

Negli ultimi anni si è assistito ad un sempre più massiccio uso dell'elettronica per il controllo di componenti meccanici. Le centraline sono sempre più complesse e quindi più difficili da sviluppare. Una delle conseguenze più rilevanti durante la fase di sviluppo è l'incremento dei costi determinato dalla necessità di fare un numero rilevante di test: si pensi ad una centralina elettronica per auto, un ciclo di collaudo prevede montaggio e utilizzo in svariate condizioni con relativo monitoraggio di tutti i parametri. Attualmente questo modello di sviluppo non è più sostenibile , sia per i costi eccessivi, sia perché è necessario testare i vari sottosistemi quando non è ancora disponibile un prodotto finito. La potenza di calcolo disponibile nei computer moderni ha permesso di rivoluzionare il processo di sviluppo e prototipazione introducendo le tecniche HIL. La centralina non è più collegata ad un prodotto reale ma ad un computer che lo simula con un modello matematico. In questo modo si ottengono diversi vantaggi:

- si può testare anche sono una parte del sistema quindi si può lavorare su prodotti non finiti con riduzione dei tempi di sviluppo
- si lavora in sicurezza ed economia perché eventuali malfunzionamenti non compromettono la funzionalità o l'integrità dell'apparecchiatura
- si possono testare e correggere i modelli software oltre che quelli hardware
- si possono automatizzare lunghe sequenze di test

Per ottenere questi risultati è importante che la centralina esaminata non si accorga di essere in un ambiente virtuale, per fare questo il computer deve risolvere i modelli matematici in tempo reale e trasmettere degli stimoli coerenti con i comandi ricevuti. Al termine dello sviluppo non ci si può comunque sottrarre ad una fase di test sul campo per verificare il prodotto finale.

Introduzione

Il seguente studio, in collaborazione con CF3000, è volto alla realizzazione di un sistema hardware in the loop per testare una centralina di conversione a metano per veicoli a benzina. L'obiettivo primario è realizzare un software in grado di sottoporre la centralina a tutti i tipi di segnale che potrà riscontrare nella propria vita operativa e di verificare che la risposta della stessa sia corretta. Secondariamente il programma di test deve funzionare in modo manuale, simulando la guida di un'auto, oppure in modo automatico utilizzando segnali preregistrati. Una terza esigenza che occorre soddisfare è la semplicità di utilizzo: si è lavorato per realizzare un progetto che fosse completo, facilmente fruibile per l'utente e facilmente modificabile per lo sviluppatore. Per ottenere questo risultato si è utilizzato il software VeriStand 2011 della National Instruments. La scelta è stata orientata dal fatto che questo programma, sviluppato espressamente per i test HIL, coniuga una buona versatilità e un utilizzo intuitivo. Questo studio è anche un'occasione per accumulare esperienza sull'uso di VeriStand nell'ambito dei laboratori della facoltà di ingegneria di Forlì.

Elenco delle abbreviazioni

#	Rappresenta un numero generico
CAN	Controller area network
CD	Custom device
DIO	Digital input-output
ECU	Engine control unit
ECUG	Engine control unit gas
FPGA	Field Programmable Gate Array
PMI	Punto morto inferiore
PMS	Punto morto superiore
PXI	PCI eXtensions for Instrumentation

Ambiente di sviluppo

Lo sviluppo del progetto si è svolto nei laboratori hangar della facoltà di ingegneria presso l'aeroporto di Forlì. A parte i PC si sono utilizzati il PXI, l'FPGA e le morsettiere in dotazione al laboratorio assieme ai software Matlab 2010a, LabView 2011 SP1 e VeriStand 2011 SP1. Una voce a parte merita L'FPGA personality.

Hardware

Scheda CF3000

Si tratta di una ECU di conversione a gas (ECUG) attualmente in sviluppo presso l'azienda CF3000. La scheda viene inserita a valle della centralina di serie di un motore a benzina e filtra tutti i segnali che questa trasmette. Sulla base di questi segnali e di quelli provenienti dai sensori del veicolo la ECUG elabora i corretti anticipi e tempi di iniezione per il funzionamento a metano oppure trasmette i segnali direttamente per il funzionamento a benzina.

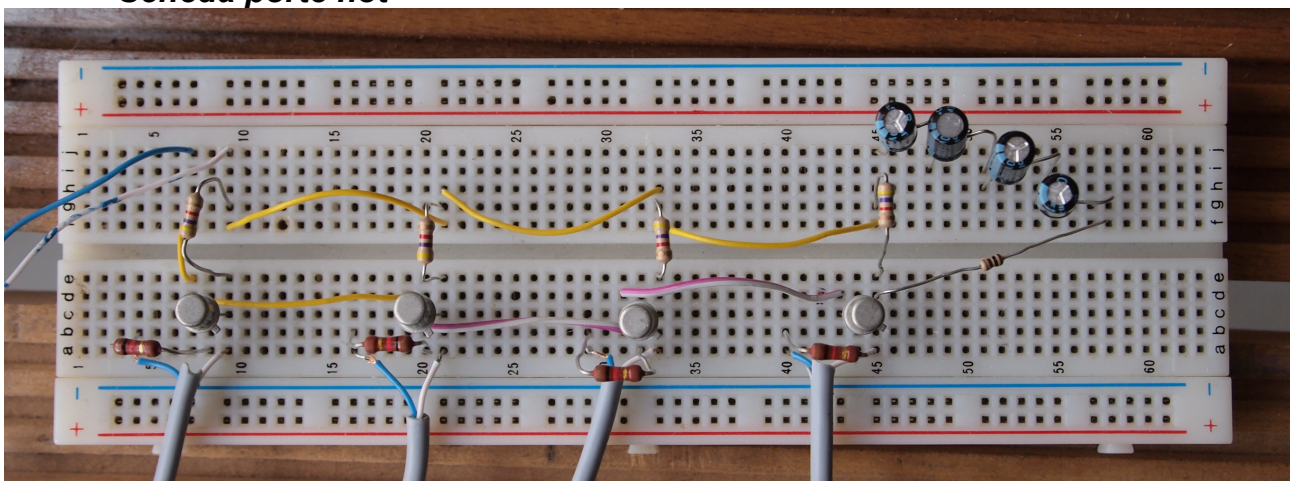
PXI

Il PCI eXtensions for Instrumentation (PXI) è una robusta piattaforma basata su PC in grado di offrire una soluzione di distribuzione ad elevate prestazioni e costi ridotti per i sistemi di misura e automazione. PXI combina le funzioni di bus elettrico PCI (Peripheral Component Interconnect) con il pacchetto modulare e robusto Eurocard di CompactPCI e aggiunge bus di sincronizzazione specializzati e funzioni software. Inoltre PXI aggiunge caratteristiche meccaniche, elettriche e software che definiscono sistemi completi di test e misura, acquisizione dati e applicazioni di produzione. Questi sistemi sono ideali per applicazioni come i test di produzione, applicazioni militari e aerospaziali, monitoraggio macchine, automotive e test industriali.

FPGA

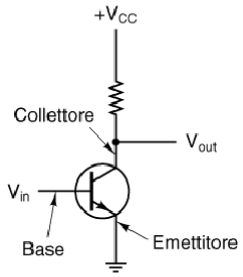
Un Field Programmable Gate Array, solitamente abbreviato in FPGA, è un circuito integrato digitale la cui funzionalità è programmabile via software. L'uso di componenti FPGA comporta alcuni vantaggi rispetto agli ASIC (Application Specific Integrated Circuit): si tratta infatti di dispositivi programmati direttamente dall'utente finale, consentendo la diminuzione dei tempi di progettazione, di verifica mediante simulazioni e di prova sul campo dell'applicazione. Il grande vantaggio rispetto agli ASIC è che permettono di apportare eventuali modifiche o correggere errori semplicemente riprogrammando il dispositivo in qualsiasi momento. Per estensione viene definita FPGA anche la scheda elettronica sulla quale il processore è installato. Questa apparecchiatura lavora a 40 MHz ed è in grado di eseguire calcoli semplici ad una velocità molto maggiore del PXI.

Scheda porte not



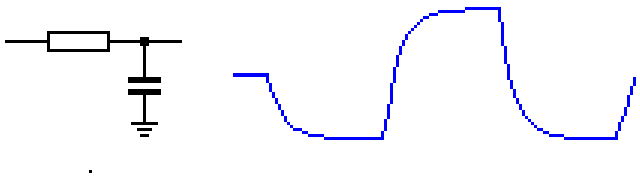
Una fase importante dello sviluppo ha riguardato il debugging del modello relativo al controllo degli errori di iniezione: è necessario assicurarsi che non si registrino falsi positivi causati dal modello stesso o dalla catena di comunicazione. Questa verifica è stata realizzata collegando i canali dell'FPGA che controllano l'attuazione degli iniettori con quelli che leggono gli stessi segnali prodotti dall'ECUG CF3000. I segnali sono onde quadre ma quelli FPGA

sono alti a iniettore chiuso e bassi a iniettore aperto, viceversa per quelli ECUG. Per rendere compatibili le comunicazioni è stato necessario realizzare delle porte NOT elettroniche di cui si riporta lo schema.



Le porte sono state realizzate con switch CDIL 2N2222A, del quale si riportano le specifiche in appendice, una resistenza da 4700 OHM tra Vinf e collettore, una resistenza da 1000 OHM tra Vin e base. Non è necessaria una alimentazione esterna perché la porta CONNECTOR 1 dell''FPGA, alla quale le porte not sono collegate, ha due canali in uscita a 5V utilizzabili a questo scopo.

E' stato effettuato un intervento su una delle quattro porte (a destra in figura) per distorcere il segnale e verificare il comportamento del sistema in condizioni sfavorevoli. La distorsione è stata realizzata con il filtro qui schematizzato



Si è utilizzata una resistenza da 100 OHM e un condensatore da 25 uF. Non essendo quest'ultimo disponibile sono stati installati 4 condensatori da 100 uF in serie. La differenza percentuale sul segnale pulito dipende dal tempo di iniezione perché la distorsione deforma l'onda sempre allo stesso modo.

LabView 2011 SP1

E' un ambiente di sviluppo avanzato particolarmente orientato

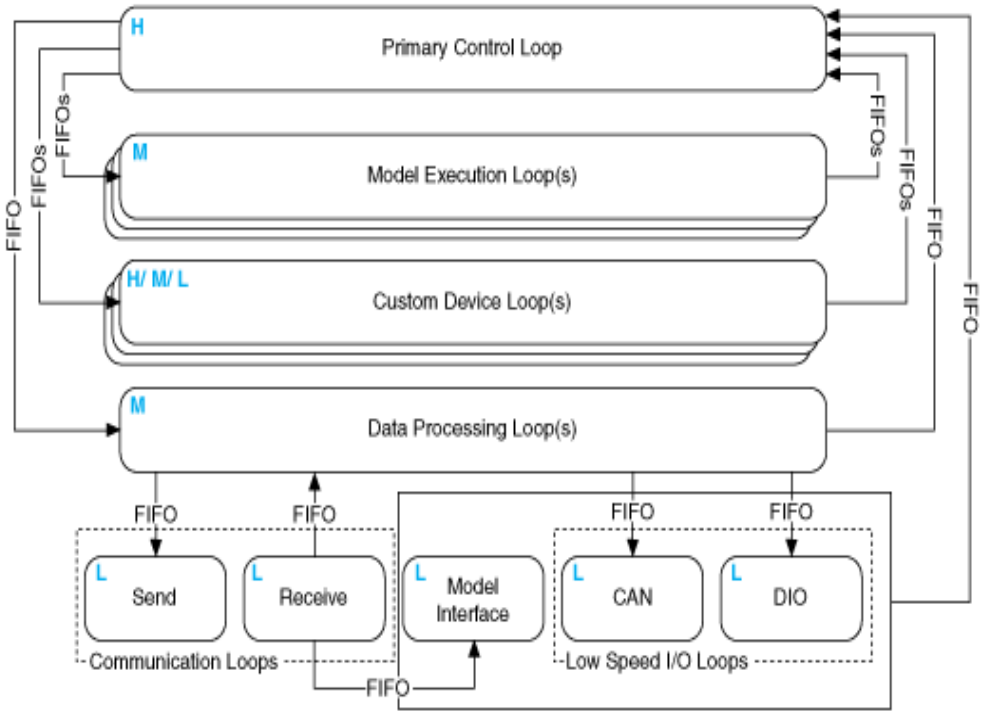
all'acquisizione di dati, alla gestione di strumentazione elettronica, all'analisi ed elaborazione di segnali. L'ambiente di programmazione è di tipo grafico, con diagrammi a blocchi. Nell'ambito del progetto qui descritto LabView è stato usato per creare componenti aggiuntivi che estendono le funzionalità del programma principale: VeriStand

Matlab 2010a

Ambiente di sviluppo particolarmente versatile basato sulla logica matriciale. Tra i moduli integrati in Matlab sicuramente il più noto è Simulink: un ambiente di programmazione a blocchi particolarmente utilizzato a livello industriale. Questo modulo è stato utilizzato per la realizzazione dei modelli matematici caricati in VeriStand.

VeriStand 2011 SP1

Si tratta di un software pensato specificamente per la realizzazione di sistemi HIL. I suoi punti di forza sono la semplicità di utilizzo, la possibilità di automatizzare totalmente i test, la possibilità di estendere le funzioni del programma con l'uso di LabView. Fondamen-



talmente VeriStand si occupa di mettere in comunicazione tra loro i modelli matematici e i componenti hardware curando, ove necessario, il sincronismo. Alla base del suo funzionamento sta un meccanismo che controlla i tempi di tutto il sistema così come la comunicazione tra il dispositivo di esecuzione e il computer host. Tale struttura è realizzata da più cicli temporizzati che usano le FIFO Real-Time per trasferire i dati tra i diversi Loop. Ogni ciclo esegue determinate attività con un'assegnata priorità. Sebbene non sia possibile cambiare la priorità o i compiti primari di ogni ciclo, è possibile personalizzare alcune operazioni, come ad esempio la frequenza di esecuzione.

Primary Control Loop

Il Primary Control Loop controlla il tempo di esecuzione dell'intera struttura e mantiene aggiornati i valori di ciascun canale. Per ogni iterazione, esegue i seguenti compiti:

- Legge e scrive ad alta velocità gli I/O dell'FPGA, gli I/O analogici e il counter delle schede di acquisizione, e i dati prodotti dai Custom Device;
- Applica lo scaling;
- Esegue un passo del profilo di stimolo, se un test è attualmente in esecuzione;
- Invia i dati Data Processing Loop, al fine di sincronizzare i valori di ciascun canale;
- Invia i dati Model Execution Loops;
- Richiede i dati al Data Processing Loop, Model Execution Loop, e Custom Device Loop;
- Esegue l'inserimento di Fault;
- Crea la mappatura delle connessioni;
- Esegue i Custom Device di tipo "Inline".

La frequenza di lavoro è impostabile dall'utente, sulla macchina Windows non può superare i 1000 Hz.

Model Execution Loop

Ogni Model Execution Loop esegue la DLL che corrisponde al modello (.mdl) realizzato. Il numero di Model Execution Loop è determinato dal numero di modelli specificato nel file di definizione del sistema. Per ogni iterazione, ogni modello di esecuzione Loop esegue i seguenti compiti:

- Legge i dati inviati dal Primary Control Loop e li assegna agli input del modello, sulla base di come è stata definita la mappatura (Mappings);
- Esegue un passo del modello;
- Legge i valori di output del modello e invia i dati al Primary Control Loop.

Custom Device Loop

È possibile determinare il numero, il comportamento e la priorità di un Custom Device Loop. VeriStand è responsabile solo per l'avvio dell'esecuzione e per la trasmissione dei dati in input e output, per ogni iterazione del Primary Control Loop.

Data Processing Loop

Come il Primary Control Loop, il Data Processing Loop mantiene una copia completa dei valori di ogni canale. Esegue le procedure, gli allarmi, i canali calcolati, così come distribuisce tra i Loop i comandi di esecuzione ricevuti dal Communication Receive Loop. Per ogni iterazione, il Data Processing Loop esegue i seguenti compiti:

- Riceve i valori di ogni canale dal Primary Control Loop.
- Esegue le procedure, gli allarmi, i canali calcolati.
- Trasmette i valori aggiornati di ogni canale al Primary Control Loop;
- Invia i dati ricevuti dal Communication Receive Loop;

Communication Loop

Il Communication Loop mantiene la comunicazione TCP / IP con il

Gateway. Ci sono due cicli di comunicazione:

- Communication Send Loop: trasmette i valori di ogni canale al Gateway;
- Communication Receive Loop: riceve dal Gateway i comandi di esecuzione.

CAN Loop (RT only)

Legge e scrive i dati a bassa velocità sulla linea CAN.

DIO Loop (RT only)

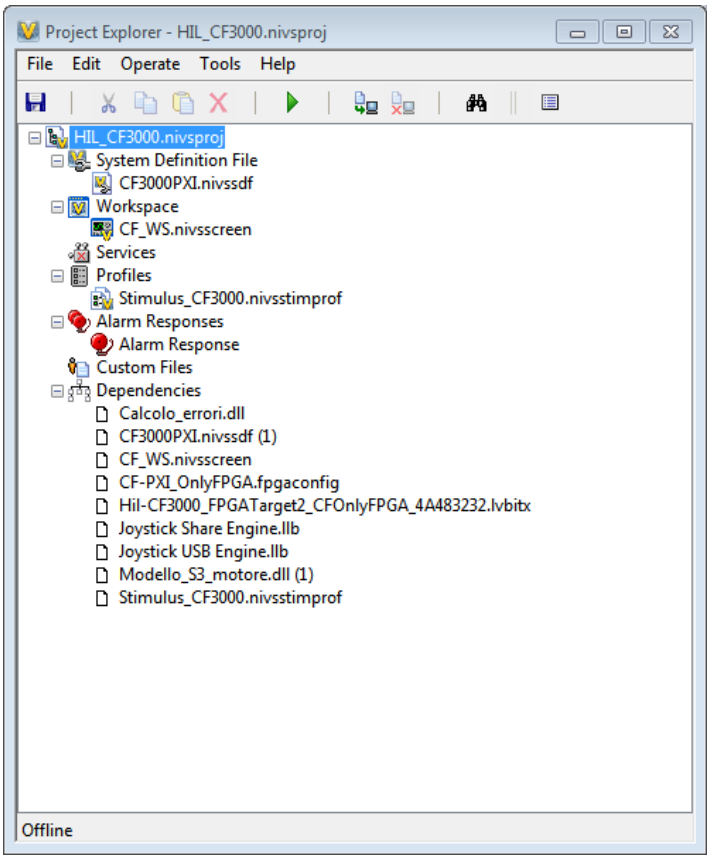
Legge e scrive i canali I/O digitali a bassa velocità sulle schede di acquisizione.

Model Interface Loop

Aggiorna i parametri del modello.

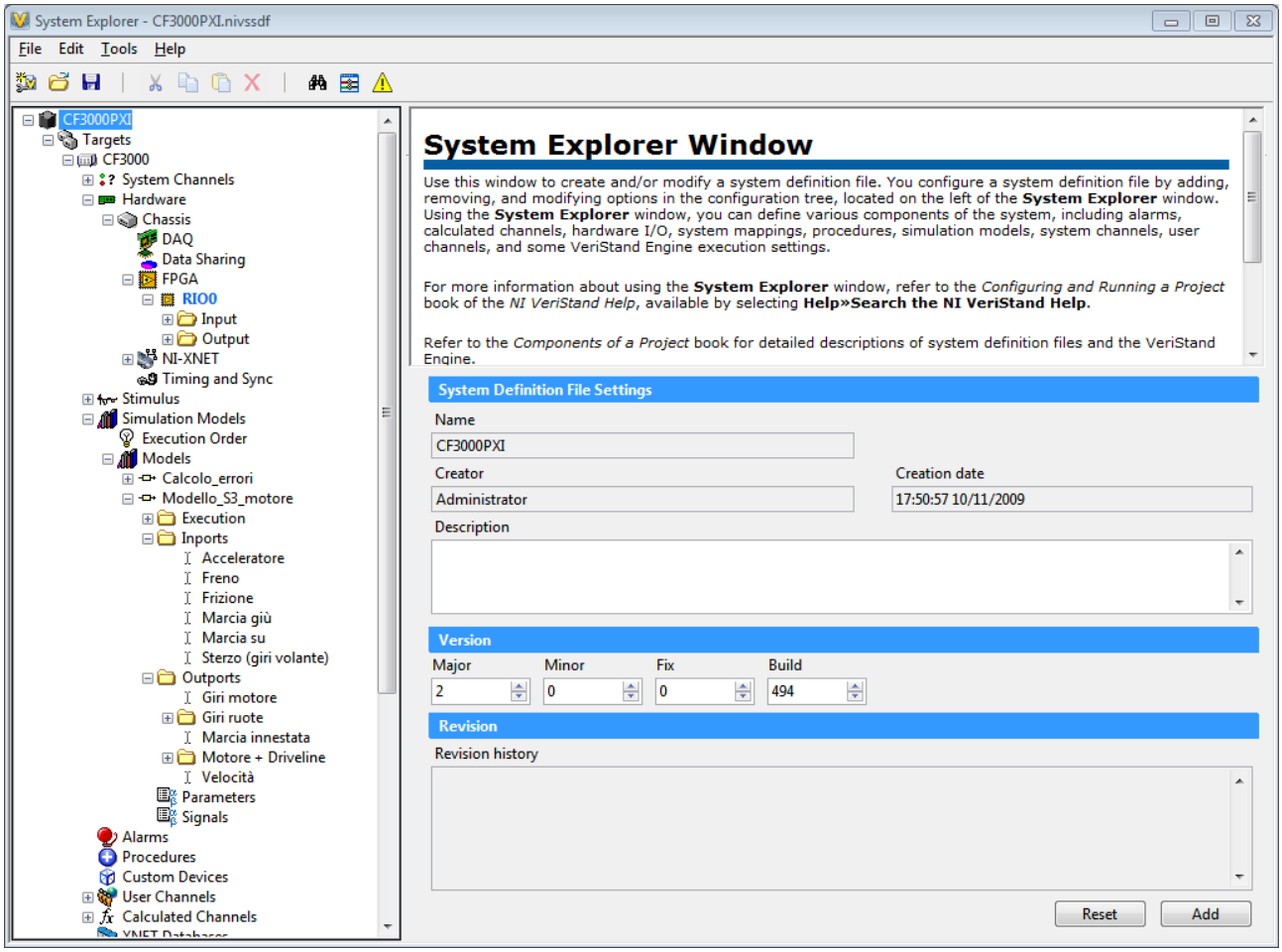
La realizzazione di un progetto in VeriStand richiede l'uso di diverse interfacce:

Project Explorer



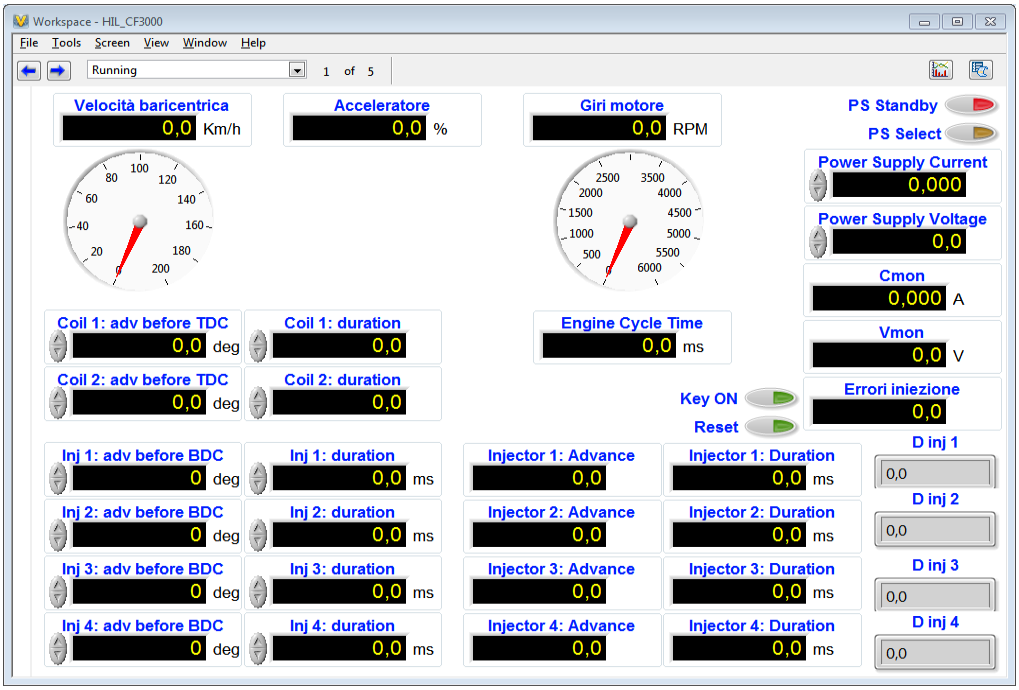
Si tratta dell'interfaccia base che si apre all'avvio di qualsiasi progetto. Da qui è possibile modificare le impostazioni del programma, accedere a tutte le altre sezioni, avviare e arrestare il progetto.

System Explorer



Quest'interfaccia consente di impostare e collegare tutto l'hardware e tutti i modelli matematici coinvolti nel progetto.

Workspace

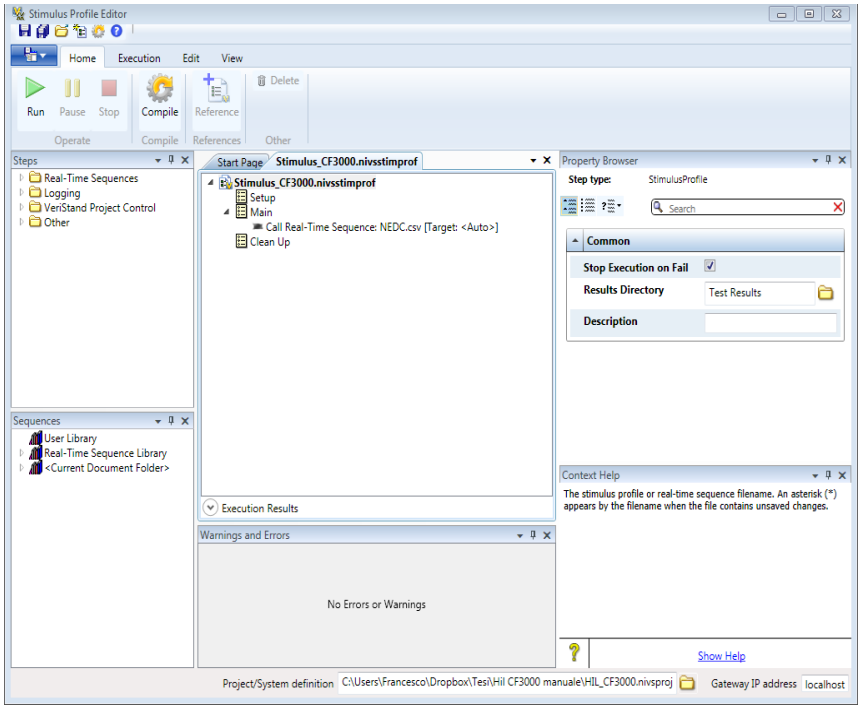


Questa è la schermata di lavoro dell'operatore, da qui è possibile

- comandare manualmente i segnali scambiati tra i vari componenti del progetto
- calibrare i canali (particolarmente importante quando si eseguono misure)
- monitorare i segnali di sistema
- controllare i dispositivi di allarme
- leggere i file di log generati dai test

Quest'interfaccia si compone di più pagine a scelta dell'operatore. E' totalmente riconfigurabile anche durante l'esecuzione del progetto.

Stimulus Profile Editor



Da quest'interfaccia è possibile realizzare dei profili di stimolo ovvero degli script che possono sostituire in parte o in toto l'operatore avendo i suoi stessi poteri. E' essenziale quando si vogliano realizzare test automatici.

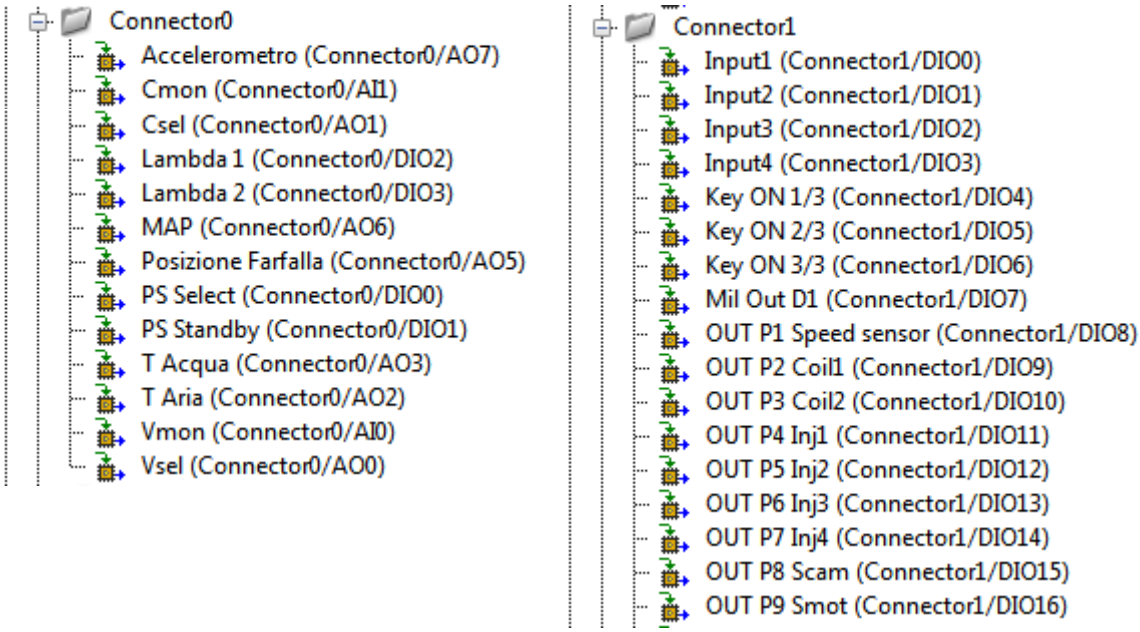
FPGA personality

Si tratta di un programma, realizzato il LabView, che gira sulla scheda FPGA. I suoi compiti sono di configurare i canali di comunicazione della scheda con il PXI e con l'esterno e di eseguire calcoli e strutture logiche. Nel caso in esame questo programma funge da tramite tra VeriStand e ECUG e si occupa di svolgere alcune funzioni, in genere semplici, che devono necessariamente essere eseguite a velocità molto maggiore di quella dell'RT engine. Un esempio molto evidente è la generazione del segnale di ruota fonica: si tratta di una ruota da 60-2 denti quindi il segnale generato è un'onda quadra con 60 periodi per giro motore. A 4500 RPM il segnale deve cambiare valore 9000 volte al secondo. Tutte le regolazioni relative alla fasatura fanno riferimento alla ruota fonica quindi

anch'esse devono essere calcolate con la stessa frequenza. Il PXI non è in grado di elaborare un progetto VeriStand a queste velocità nemmeno se quest'ultimo è vuoto, occorre necessariamente programmare in FPGA.

Canali di comunicazione FPGA-ECUG

Si utilizzano i connettori 0 e 1 della scheda FPGA.



In figura sono visibili tutti i canali di comunicazione con la scheda ECUG. La voce DIO# indica porte digitali, la voce AO# uscite analogiche (dall'FPGA) e la voce AI# ingressi analogici. I vari segnali sono analizzati in tabella:

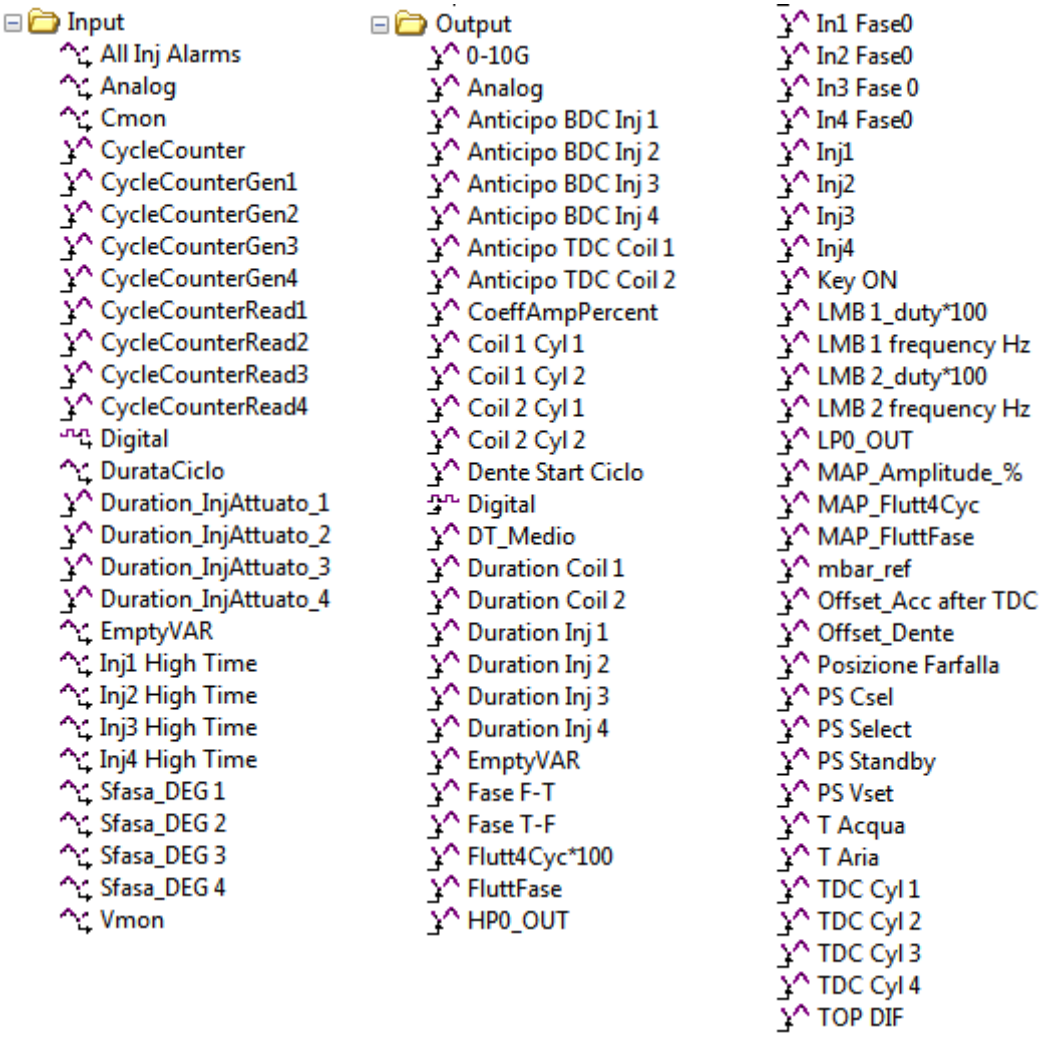
Nome	In/out	Volt	Porta	Note
Accelerometro	OUT	0-5	0/AO7	Segnale di detonazione
Cmon	IN	0-3	0/AI1	Lettura corrente alimentatore
Csel	OUT	0-3	0/AO1	Comando corrente alimentatore
Lambda1	OUT	0-0,9	0/DIO2	Onda quadra con duty cycle variabile, segnale sonda lambda1
Lambda2	OUT	0-0,9	0/DIO3	Onda quadra con duty cycle variabile, segnale sonda lambda2
MAP	OUT	0-5	0/AO6	Pressione relativa nel collettore aspirazione
Posizione Farfalla	OUT	0-5	0/AO5	
PS Select	OUT	0-Vbatt	0/DIO0	Controllo Alimentatore
PS Standby	OUT		0/DIO1	Controllo Alimentatore
T acqua	OUT	0-5	0/AO3	
T aria	OUT	0-5	0/AO2	
Vmon	IN	0-3	0/AI0	Lettura tensione di alimentazione
Vsel	OUT	0-3	0/AO0	Comando tensione di alimentazione
Input 1	IN	0-Vbatt	1/DIO0	Lettura della posizione iniettore (ECUG). Segnale alto = aperto, segnale basso =chiuso. Onda quadra con periodo e duty cycle variabile
Input 2	IN	0-Vbatt	1/DIO1	
Input 3	IN	0-Vbatt	1/DIO2	
Input 4	IN	0-Vbatt	1/DIO3	
Key on 1/3	OUT	0-Vbatt	1/DIO4	Controllo Alimentatore
Key on 2/3	OUT	0-Vbatt	1/DIO5	
Key on 3/3	OUT	0-Vbatt	1/DIO6	
Mil Out D1	OUT	0-Vbatt	1/DIO7	Attivo basso. Open collector
Speed Sensor	OUT	0-Vbatt	1/DIO8	Sensore di velocità, Onda quadra con duty cycle variabile
Coil1	OUT	0-Vbatt	1/DIO9	Segnale attivazione bobine, onda quadra con duty cycle variabile
Coil2	OUT	0-Vbatt	1/DIO10	Segnale attivazione bobine, onda quadra con duty cycle variabile
Inj1	OUT	0-Vbatt	1/DIO11	Segnale apertura iniettori (ECU). Onda quadra con duty cycle variabile. Alto=chiuso
Inj2	OUT	0-Vbatt	1/DIO12	

Nome	In/out	Volt	Porta	Note
Inj3	OUT	0-Vbatt	1/DIO13	basso=aperto.
Inj4	OUT	0-Vbatt	1/DIO14	
Scam	OUT	0-Vbatt	1/DIO15	Segnale di fase
Smot	OUT	0-Vbatt	1/DIO16	Segnale di ruota fonica

Vbatt dipende dalla batteria, nel caso dell'FPGA questi segnali valgono 0-5 V.

Tutti i segnali del tipo onda quadra con duty cycle variabile vengono generati dall'FPGA su istruzione di VeriStand perché richiedono una risoluzione temporale di 0,1 ms.

Canali di comunicazione FPGA-VeriStand



Si tratta delle connessioni virtuali tra VeriStand e la scheda FPGA. I valori in questo caso sono adimensionali. Nella figura successiva compaiono le voci Analog e Digital. Questa distinzione riguarda la caratteristica del segnale in uscita dall'FPGA perché naturalmente la comunicazione tra quest'ultimo e VeriStand è solo digitale. Quando l'FPGA è impostato in modalità scansione queste cartelle vengono utilizzate per ordinare i segnali da trasmettere e ricevere. In questo caso sono vuote perché l'FPGA lavora in modalità programmazione quindi tutti i segnali elencati vengono elaborati all'interno della scheda e la distinzione tra analogico e digitale non ha più alcun Notare che Input e Output sono riferiti al punto di vi-

sta di VeriStand

In/Out	Nome	Funzione
OUT	0-10G	Accelerometro detonazione
	Analog	Cartella vuota
	Anticipo BDC Inj #	Anticipo dell'inizio iniezione rispetto al PMI in deg
	Anticipo TDC coil #	Anticipo della bobina rispetto al PMS in deg
	CoeffAmpPercent	Coefficiente di fluttuazione RPM percentuale
	Coil # Cyl #	Assegnazione dei cilindri alle bobine
	Dente Start Ciclo	Posizione del contatore cicli (0-120)
	Digital	Cartella vuota
	DT_Medio	Durata di un periodo dente della ruota fonica in ticks
	Duration Coil #	Durata scarica delle bobine in ms
	Duration inj #	Durata iniezioni in ms
	EmptyVAR	Vuota
	Fase F-T	Dente di innalzamento segnale fase
	Fase T-F	Dente abbassamento segnale fase
	Fluct4Cyc*100	Numero fluttuazioni di giri per ciclo
	FluttFase	Fase della fluttuazione in deg
	HP0_OUT	
	In # Fase #	Dente di riferimento per la fasatura dell'iniettore
	Inj#	Durata iniezione comandata dall'ECU in ms
	Key ON	Comando alimentatore esterno
	LMB #_duty*100	Duty cycle delle sonde lambda
	LMB# frequency Hz	Frequenza ciclo delle sonde lambda in Hz
	LP0_OUT	
	MAP_Amplitude %	Variazione percentuale pressione collettore.
	MAP_Flutt4Cyc	Numero variazioni di pressione collettore per ciclo
	MAP_FluttFase	Fase dell'onda di pressione in deg
	mbar_ref	Pressione relativa nel collettore di aspirazione in mbar

In/Out	Nome	Funzione
	Offset_Acc after TDC	Angolo della detonazione dopo il PMS in Deg
	Offset_Dente	Sfasatura dente di riferimento rispetto alla cava della ruota fonica espresso in denti
	Posizione farfalla	Posizione acceleratore in percentuale
	PS Csel	Corrente di alimentazione (unità di misura personalizzata)
	PS Select	Comando alimentatore esterno
	PS Standby	Comando alimentatore esterno
	PS Vset	Tensione di alimentazione (unità di misura personalizzata)
	T acqua	Temperatura acqua (unità di misura personalizzata)
	T aria	Temperatura aria (unità di misura personalizzata)
	TDC Cyl #	Dente di riferimento fase cilindro
	TOP DIF	Numero di campioni del filtro digitale integrale
IN	All Inj Alarms	Scollegato
	Analog	Vuoto
	Cmon	Corrente di alimentazione ECUG(unità misura pers.)
	CycleCounter	Contatore cicli generale
	CycleCounterGen#	Contatore cicli cilindro
	CycleCounterRead#	Contatore cicli cilindro
	Digital	Vuoto
	DurataCiclo	Durata ciclo motore in ms
	DurationInjAttuato #	Durata iniezione effettivamente inviata alla ECUG in ms
	EmptyVAR	Vuoto
	Inj# High Time	Durata iniezione ricevuta dall'ECUG in ms
	Sfasa_DEG #	Sfasamento dell'inizio iniezione dal PMI in deg
	Vmon	Tensione di alimentazione ECUG (unità misura pers.)

Unità di misura personalizzata significa che il segnale è stato scalato con un Calculated Channel quindi non ha più un'unità di misura standard.

Impostazione del progetto in VeriStand

Nonostante si sia lavorato per ridurre al minimo gli interventi da parte dell'utente finale sulla configurazione, questi rimangono necessari: si pensi al bisogno di sostituire la FPGA personality con una adeguata all'hardware in proprio possesso oppure alla necessità di invertire o meno gli assi del controller (volante) in proprio possesso. A queste necessità si oppone il fatto che la configurazione di VeriStand, man mano che si aggiungono funzioni, diventa sempre più complessa. Integrare tutti i modelli matematici in uno unico non è possibile per due ragioni: la prima e più importante è che lo sviluppo del progetto è ancora in corso e per l'operatore è molto più pratico lavorare su un numero ristretto di modelli semplici piuttosto che su un unico modello complesso. La seconda ragione è che, al crescere della complessità, il modello pesa sull'hardware in modo più che proporzionale, tanto che la NI consiglia di spezzare lo stesso se si hanno problemi di prestazione In conclusione si è deciso di realizzare due test differenti: uno pensato per un uso completamente automatico e uno pensato per un uso manuale.

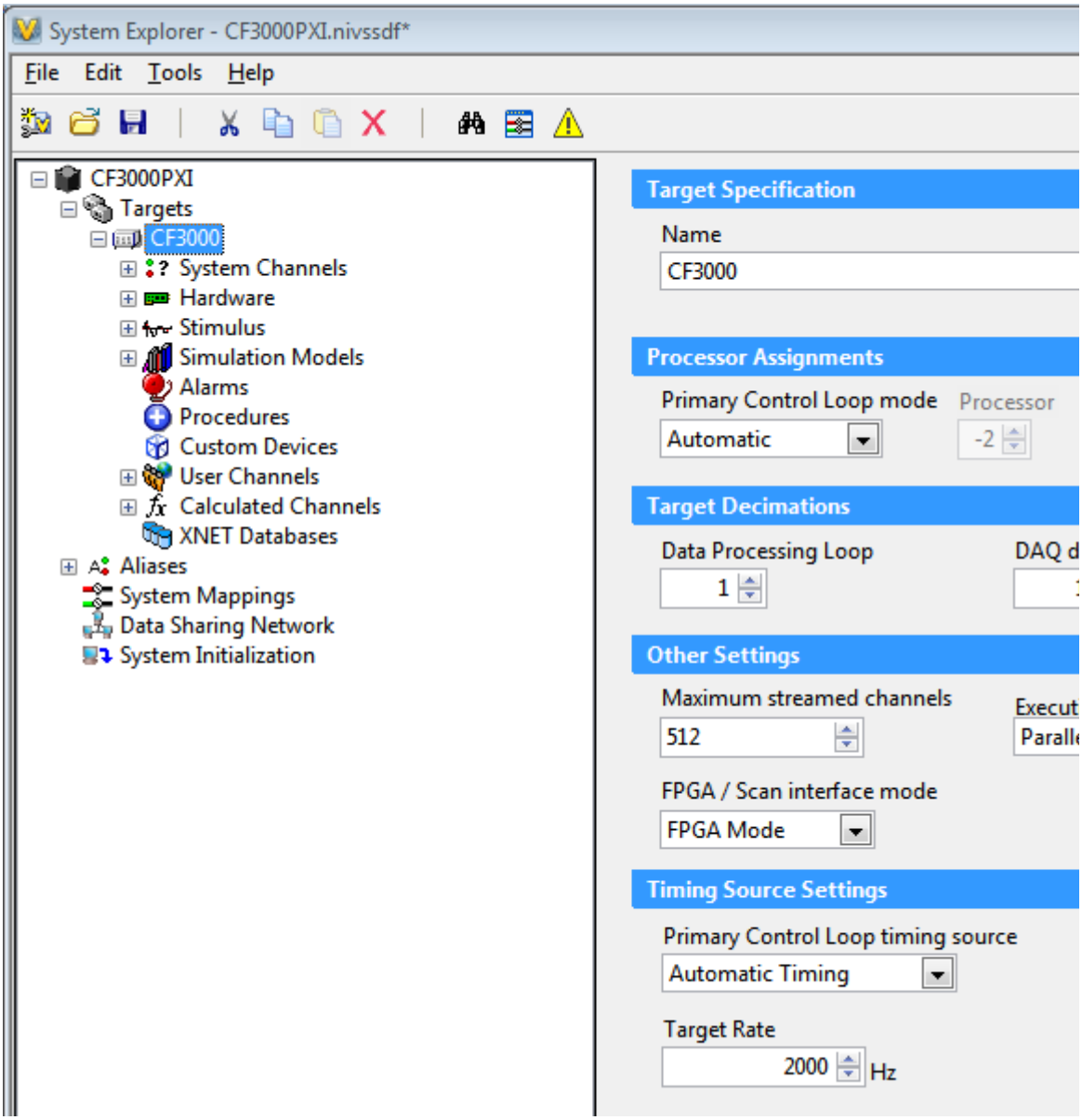
In questo capitolo ci si è concentrati sul file di configurazione del progetto. Per un'analisi dettagliata dell'FPGA, dei modelli matematici, dei loro ingressi e delle loro uscite si vedano i rispettivi capitoli. Le voci che non hanno subito alcuna modifica rispetto alla configurazione di default non sono state descritte, per una loro analisi dettagliata si rimanda alla documentazione tecnica (bibliografia).

Test per uso automatico

Si tratta della versione dalla struttura più semplice infatti non contiene modelli matematici se non quello di controllo degli errori e non consente l'uso di controller esterni.

Impostazione del progetto in VeriStand

Target CF3000

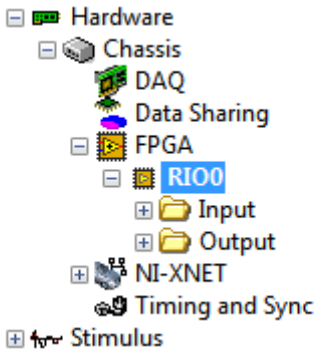


Sul lato sinistro dell'immagine è visibile l'albero di configurazione, sulla destra si può osservare parte delle impostazioni del target CF3000. La gran parte dei parametri di questa schermata è impostata ai valori di default, uniche eccezioni sono:

- Name: CF3000
- Operating System: PharLap (non visibile) perché si utilizza un PXI

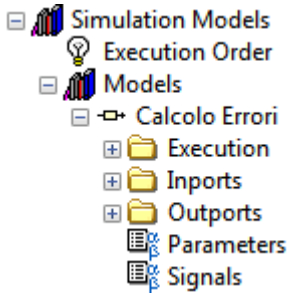
- IP Address: (non visibile) è impostato sul valore relativo allo strumento di laboratorio, un utente esterno deve adattarlo alle proprie esigenze
- Target Rate: è impostato su 2000 Hz, grazie alla leggerezza del progetto. Nella versione manuale è impostato a 1000 Hz

Hardware



L'unico hardware aggiuntivo per questo test è l'FPGA il quale non necessita di alcuna configurazione su VeriStand. Si ricordi che il flusso dei canali va interpretato dal punto di vista di VeriStand quindi la voce input contiene i segnali che l'FPGA invia a VeriStand e la voce Output tutti quelli che da VS vanno all'FPGA.

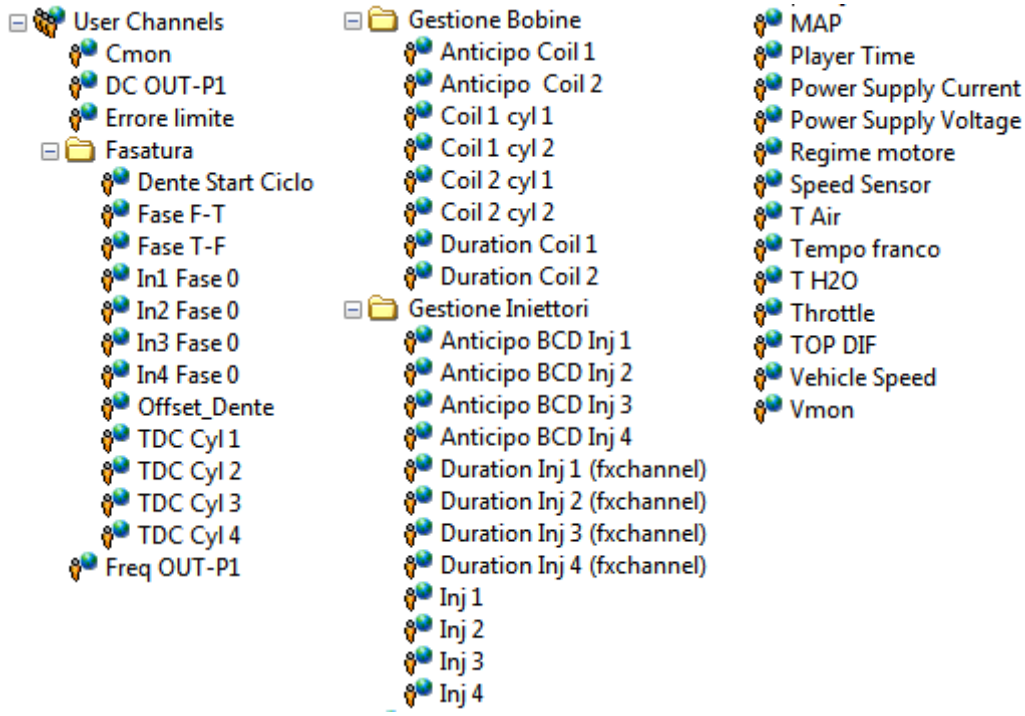
Simulation models



Solo il calcolo errori è stato inserito. E' possibile usare modelli Matlab in formato mdl e in formato dll ma solo questi ultimi possono funzionare su PXI. VeriStand preleva il file dalla cartella indicata quindi qualora venga aggiornato il modello viene aggiornato automaticamente anche il progetto. Occorre attenzione perché se si ag-

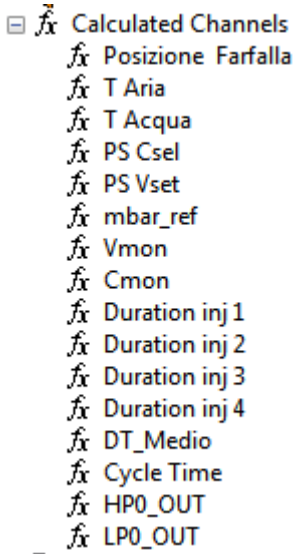
giungono, rimuovono o rinominano i terminali del modello occorrerà correggere manualmente gli allacciamenti su VS. Al contrario di input e output, le voci Inports e Outports sono relative al modello quindi si comportano all'opposto di quanto visto prima.

User Channels



Ogni User channel rappresenta una variabile modificabile dall'utente o dal programma, é adimensionale e non può svolgere calcoli ma solo essere scritto o letto. Tutti i canali di questa lista hanno omonimi nelle voci input, output, Inports e Outports dell'hardware e dei modelli e sono ad essi collegati. Non si tratta di oggetti strettamente necessari in questo test ma sono stati aggiunti per consentire una facile sostituzione di Hardware e modelli matematici: se si sostituisce un componente si possono utilizzare gli User Channels relativi come terminali che dall'altro lato rimangono collegati. Si aggiunga anche il fatto che gli User channel possono memorizzare un valore di default. Grazie a ciò tutte le variabili del workspace possono essere preimpostate e non reimmesse ad ogni avvio. L'aggiunta di sottocartelle consente di mantenere questa sezione ben leggibile.

Calculated Channels



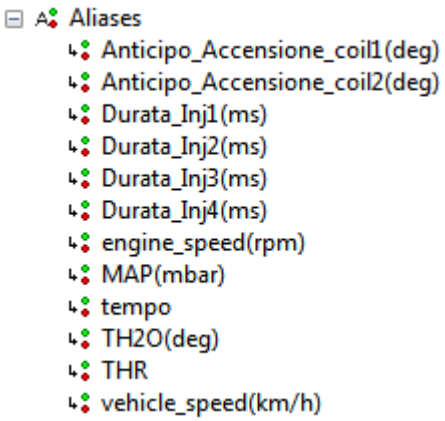
Questi canali possono svolgere semplici calcoli ma non sono organizzabili con sottocartelle. All'interno del progetto sono usati per convertire i valori immessi dall'utente nelle unità di misura congeniali alla macchina. Un'ulteriore conversione necessaria è quella relativa al tipo di numero: per esempio occorre convertire numeri interi che vanno da 0 a 32767 in numeri compresi tra 0 a 1. Come gli User Channel, anche la gran parte dei Calculated Channel non è strettamente necessaria perché tutti gli input e output verso i componenti aggiuntivi del progetto consentono di effettuare una scalatura. Il motivo per cui sono stati ugualmente inseriti è che la sostituzione dei componenti può avvenire senza dover riconfigurare ogni volta le scalature ma limitandosi all'opportuno allacciamento, un procedimento che può essere estremamente rapido.

- Posizione Farfalla: $\frac{\text{Throttle} * 32767}{200}$ converte la percentuale di pressione del pedale in volt (da 0 a 5). La moltiplicazione per 32767, come per i canali successivi, è necessaria a compensare la divisione per uno stesso valore che viene creata di default nei canali analogici dell'FPGA.
- T Aria: $\frac{\text{T Air} * 32767}{200}$ converte una temperatura che va da -20

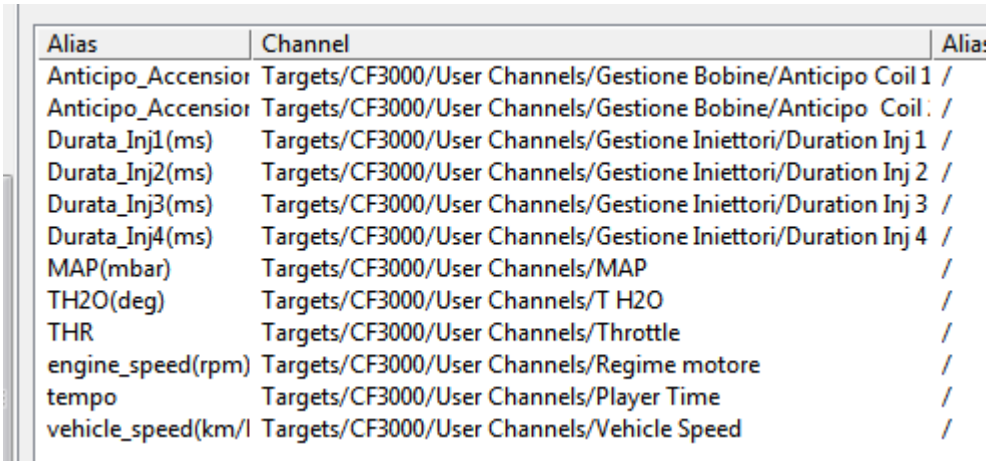
a 80°C (spostata con un offset a 0-100) in 0-5 V

- T Acqua: $\frac{\text{T H2O} * 32767}{300}$ fa lo stesso con la temperatura da 0 a 150°C del liquido di raffreddamento
- PS Csel: $\frac{\text{Power Supply Current} * 32767}{5}$ trasforma la richiesta di corrente all'alimentatore in un segnale di tensione
- PS Vset: $\frac{\text{Power Supply Voltage} * 32767}{32}$ trasforma la richiesta di tensione in un altro segnale sempre in tensione.
- mbar_ref: $\frac{\text{MAP} * 32767}{2000}$ da 0-1000 mBAR a 0-5 V
- Vmon: $\frac{\text{Vmon} * 32}{32767}$ solo monitoraggio
- Cmon: $\frac{\text{Cmon} * 5}{32767}$ solo monitoraggio
- Duration inj#: $\text{Duration Inj \# (fxchannel)} * 40000$ passa da ms a ticks
- DT_Medio: $\frac{40000000}{\text{Regime motore}}$ da RPM a ticks
- Cycle Time: $\frac{120000}{\text{Regime motore}}$ da RPM a ms
- HP0_OUT: $\frac{\text{DC OUT-P1} * 40000000 * 18}{5 * \text{Speed Sensor} * \text{Vehicle Speed}}$
- LP0_OUT: $\frac{(1 - \text{DC OUT-P1}) * 40000000 * 18}{5 * \text{Speed Sensor} * \text{Vehicle Speed}}$

Aliases



I canali possono essere controllati o inserendo l'indirizzo esatto o inserendo l'alias ovvero un nome univoco rappresentativo del canale. In questo caso sono stati inseriti per rendere più rapida la configurazione dei profili di stimolo. Le associazioni sono riportate in figura



Alias	Channel	Alia
Anticipo_Accensior	Targets/CF3000/User Channels/Gestione Bobine/Anticipo Coil 1	/
Anticipo_Accensior	Targets/CF3000/User Channels/Gestione Bobine/Anticipo Coil	/
Durata_Inj1(ms)	Targets/CF3000/User Channels/Gestione Iniettori/Duration Inj 1	/
Durata_Inj2(ms)	Targets/CF3000/User Channels/Gestione Iniettori/Duration Inj 2	/
Durata_Inj3(ms)	Targets/CF3000/User Channels/Gestione Iniettori/Duration Inj 3	/
Durata_Inj4(ms)	Targets/CF3000/User Channels/Gestione Iniettori/Duration Inj 4	/
MAP(mbar)	Targets/CF3000/User Channels/MAP	/
TH2O(deg)	Targets/CF3000/User Channels/T H2O	/
THR	Targets/CF3000/User Channels/Throttle	/
engine_speed(rpm)	Targets/CF3000/User Channels/Regime motore	/
tempo	Targets/CF3000/User Channels/Player Time	/
vehicle_speed(km/l	Targets/CF3000/User Channels/Vehicle Speed	/

System mappings

Source	Destination
LP0_OUT [CF3000/Calculated Channels/]	LP0_OUT [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
HP0_OUT [CF3000/Calculated Channels/]	HP0_OUT [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
TOP DIF [CF3000/User Channels/]	TOP DIF [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Dente Start Ciclo [CF3000/User Channels/Fasatura/]	Dente Start Ciclo [CF3000/Hardware/Chassis/FPGA/RIO0/Out
Offset_Dente [CF3000/User Channels/Fasatura/]	Offset_Dente [CF3000/Hardware/Chassis/FPGA/RIO0/Output
Fase T-F [CF3000/User Channels/Fasatura/]	Fase T-F [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Fase F-T [CF3000/User Channels/Fasatura/]	Fase F-T [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Inj 4 [CF3000/User Channels/Gestione Iniettori/]	Inj4 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Inj 3 [CF3000/User Channels/Gestione Iniettori/]	Inj3 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Inj 2 [CF3000/User Channels/Gestione Iniettori/]	Inj2 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Inj 1 [CF3000/User Channels/Gestione Iniettori/]	Inj1 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
DT_Medio [CF3000/Calculated Channels/]	DT_Medio [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Coil 2 cyl 2 [CF3000/User Channels/Gestione Bobine/]	Coil 2 Cyl 2 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Coil 2 cyl 1 [CF3000/User Channels/Gestione Bobine/]	Coil 2 Cyl 1 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Coil 1 cyl 2 [CF3000/User Channels/Gestione Bobine/]	Coil 1 Cyl 2 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Coil 1 cyl 1 [CF3000/User Channels/Gestione Bobine/]	Coil 1 Cyl 1 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
TDC Cyl 4 [CF3000/User Channels/Fasatura/]	TDC Cyl 4 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
TDC Cyl 3 [CF3000/User Channels/Fasatura/]	TDC Cyl 3 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
TDC Cyl 2 [CF3000/User Channels/Fasatura/]	TDC Cyl 2 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
TDC Cyl 1 [CF3000/User Channels/Fasatura/]	TDC Cyl 1 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
In4 Fase 0 [CF3000/User Channels/Fasatura/]	In4 Fase0 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
In3 Fase 0 [CF3000/User Channels/Fasatura/]	In3 Fase 0 [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]

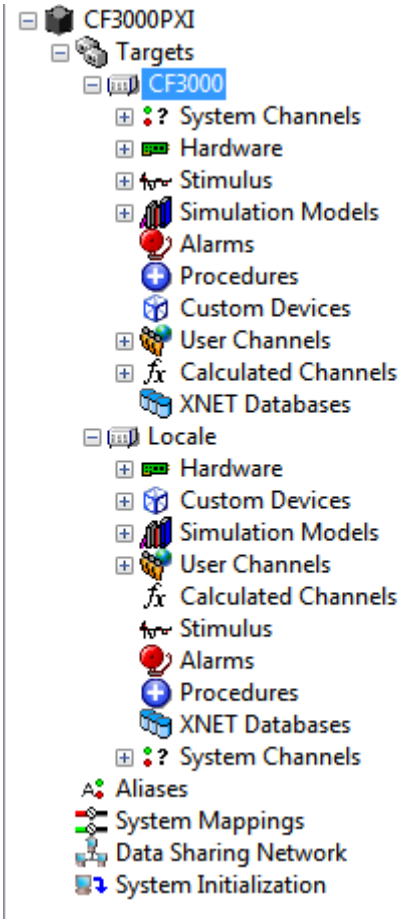
Al contrario degli altri casi la figura mostra solo una parte di tutte le voci. VeriStand non consente di ordinarle con sottocartelle quindi l'elenco è piuttosto lungo. I System Mappings sono come fili virtuali: leggono il valore di un canale qualsiasi e lo trasmettono ad un altro canale. Mentre un solo punto di partenza può giungere a diverse destinazioni un punto di arrivo può ricevere un solo segnale. Gran parte dei link di questo elenco collegano gli User Channel alle entrate e uscite dell'FPGA e del modello matematico. Quando questi vengono sostituiti, i rispettivi collegamenti vengono cancellati o vengono segnalati come problematici e vanno ricostruiti uno per uno. A questo punto è il caso di specificare perché si è scelto di creare tante variabili e tanti collegamenti che come detto non sono strettamente necessari. La ragione è che l'elenco dei collegamenti è esportabile e importabile. Se, per esempio, si vuole sostituire l'FPGA personality, non è necessario ricostruire la quarantina di link che la coinvolgono: basta esportare l'elenco, sostituire l'FPGA,

cancellare tutti i collegamenti del progetto (si può fare in blocco), importare l'elenco. In pochi secondi il cablaggio viene ripristinato, lo stesso non si può fare con le scalature dei canali. L'altra faccia della medaglia è che all'aumentare dei componenti diventa progressivamente più oneroso seguire una serie di link. Probabilmente questo aspetto di VeriStand verrà rivisto in una futura versione.

Test per uso manuale

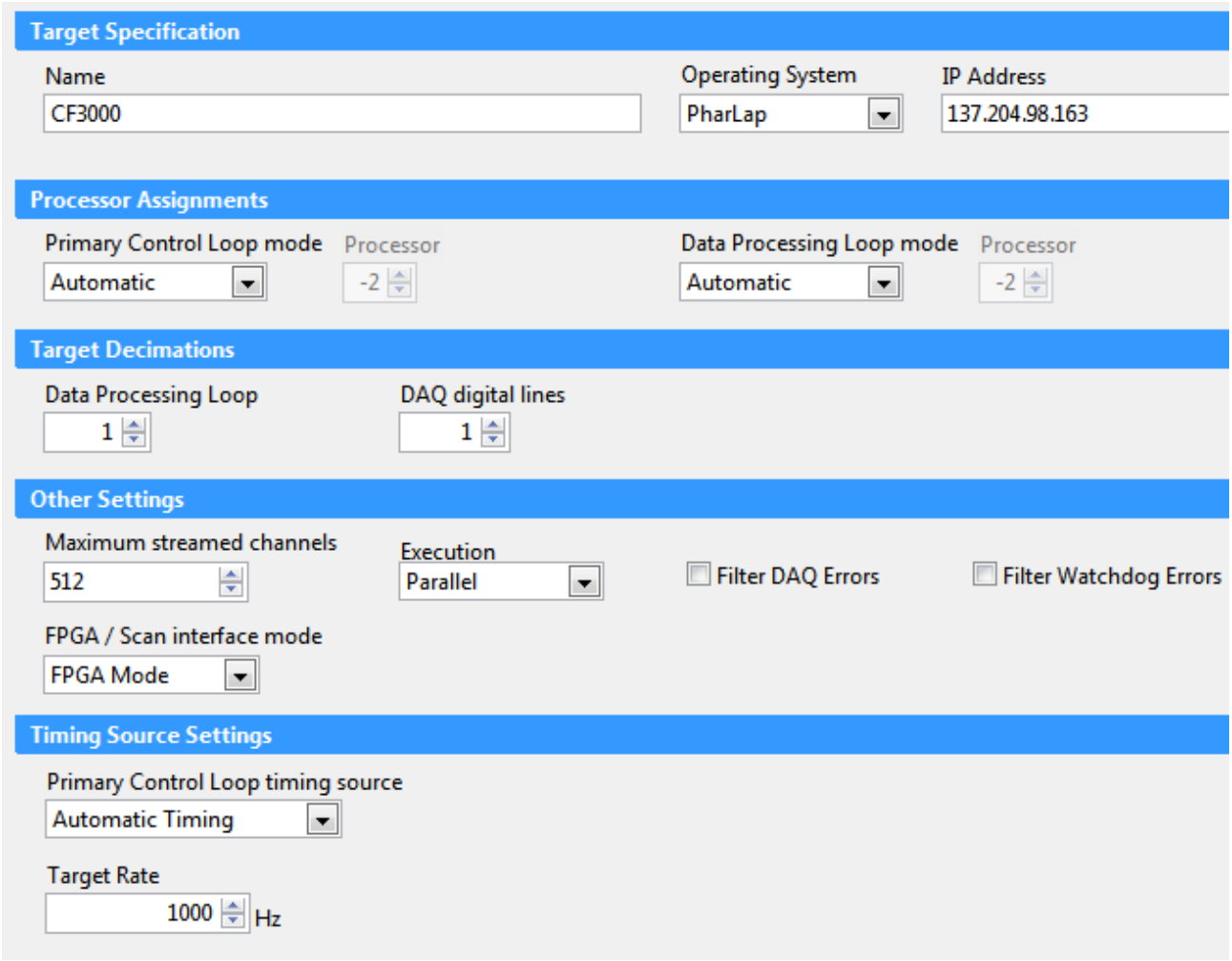
Questa versione del test comprende il modello motore ed ECU e il controllo via HID USB; la base di partenza per lo sviluppo è la precedente versione del test.

Target aggiuntivi



Dall'albero di configurazione si può notare che oltre a *CF3000* c'è il target *Locale*. Il secondo target è necessario per la gestione dei controller USB.

CF3000



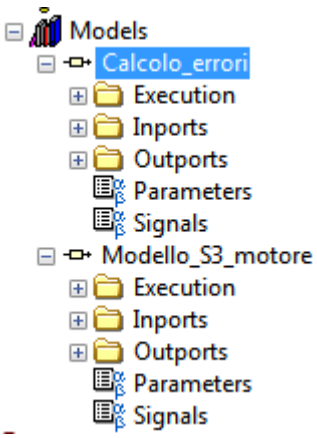
Il peso di questa versione fa sì che il processore del PXI vada in saturazione a 2000 Hz, inoltre il modello veicolo è tarato sui 1000 Hz quindi tutto il progetto è stato impostato a questa velocità.

Hardware (CF3000)

Questa sezione è esattamente identica al caso precedente

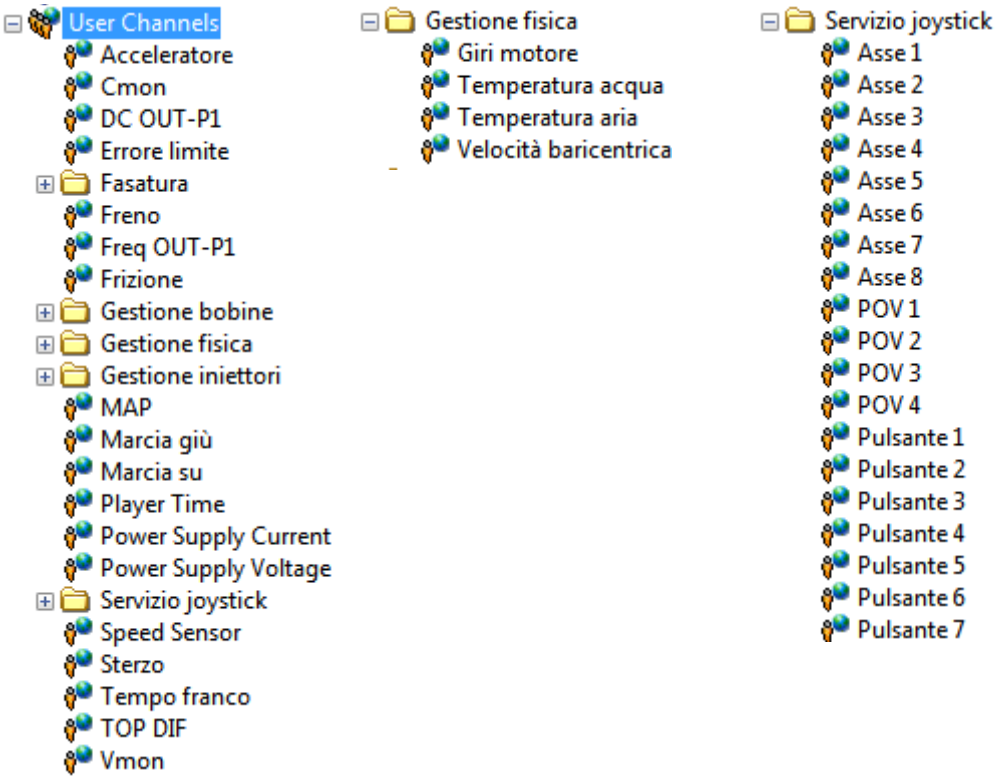
Simulation models (CF3000)

Impostazione del progetto in VeriStand



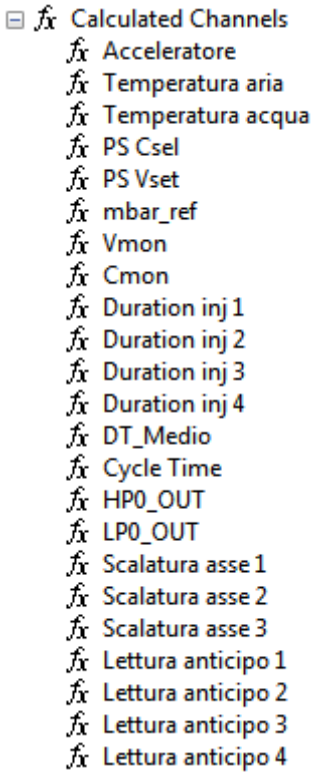
E' stato aggiunto il Modello_S3_motore, anche esso realizzato in Simulink. Ogni modello può essere eseguito alla velocità del PCL oppure ad una sua frazione con numeratore 1. Si è analizzata la possibilità di mantenere a 2000 Hz il controllo errori e a 1000 Hz il modello veicolo ma questa combinazione risulta comunque troppo gravosa e non offre reali vantaggi ai regimi di giri considerati.

User Channels (CF3000)



Sono stati aggiunti gli User Channel relativi alla guida del veicolo: Acceleratore (ha sostituito Throttle), Freno, Frizione, Marcia su, Marcia giù, Sterzo. Per rendere più fruibile l'elenco è stata inserita la voce Gestione Fisica nella quale sono stati trasferiti alcuni canali e la voce Servizio joystick che contiene tutti i canali necessari al custom device *Joystick Share* (i pulsanti, non tutti visibili, arrivano al 32). Al contrario di quanto detto per il caso precedente, gli User Channels relativi a Servizio joystick in questo caso sono fondamentali. E' certamente possibile migliorare l'organizzazione di questo settore ma, in questo caso, il procedimento risulta particolarmente lento e suscettibile di errori (vedi la voce System Mappings in fondo al capitolo)

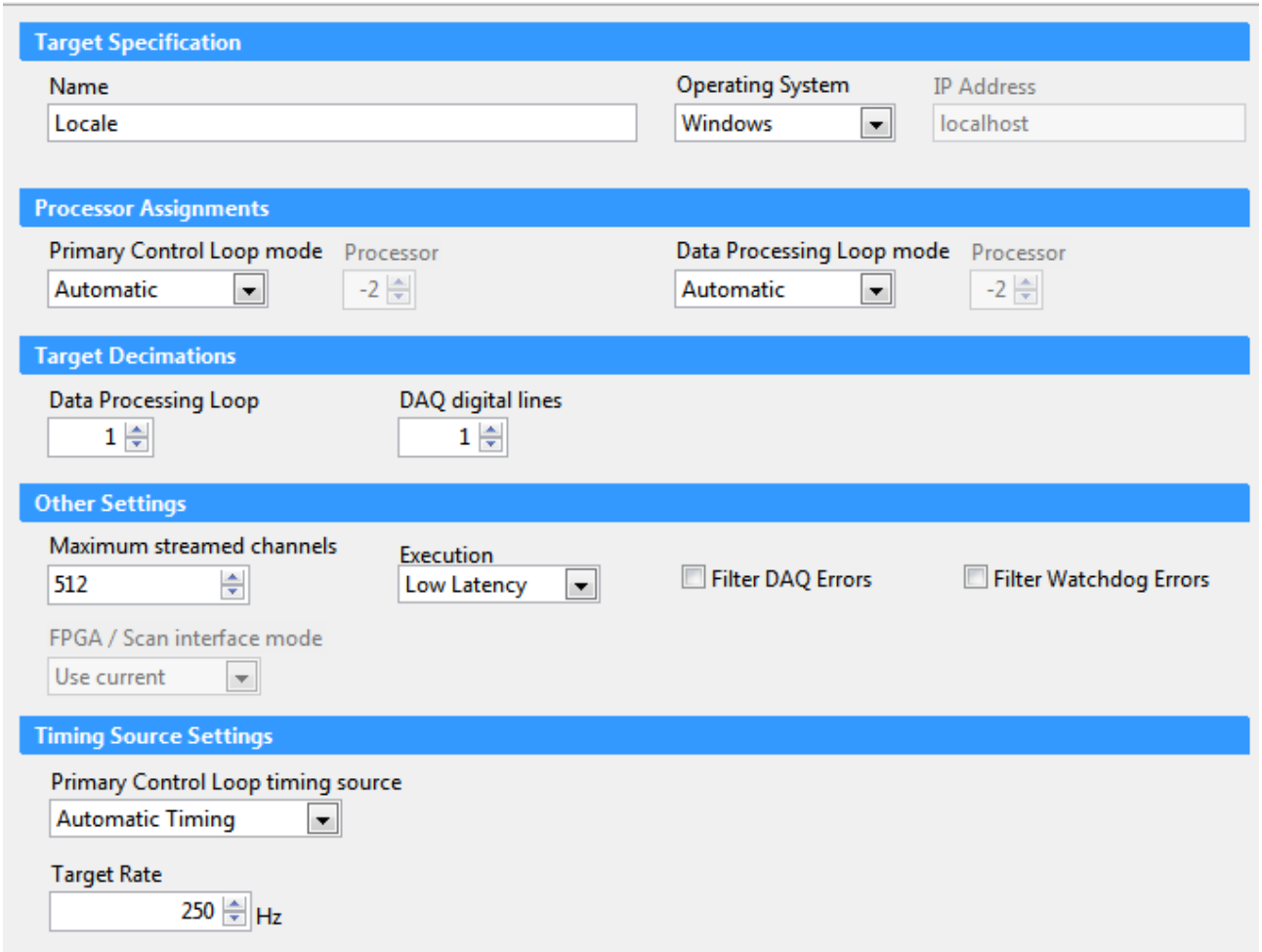
Calculated Channels (CF3000)



I canali sono gli stessi del caso precedente anche se alcuni sono stati rinominati (Acceleratore, Temperatura acque, Temperatura aria). Sono stati aggiunti i seguenti:

- Scalatura asse #: gli assi del joystick leggono un valore che va da -32767 a 32767, questi canali trasformano i valori in quelli necessari e consentono di invertire l'asse in caso di necessità. Purtroppo allo stato attuale l'utente deve intervenire direttamente su queste voci per tarare il joystick.
- Lettura anticipo #: Nel workspace è possibile leggere l'anticipo di iniezione rispetto al PMI. In questa versione del test la scalatura è stata spostata dal workspace ai Calculated Channels. E una moltiplicazione per 0,006

Locale

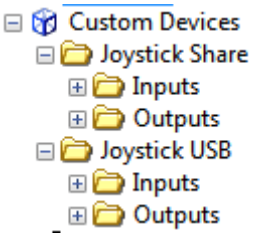


Questo target gestisce i custom device relativi al joystick. Si può notare il sistema operativo Windows e l'IP localhost. Si può anche notare il Target Rate impostato a 250 Hz. Il PC utilizzato per lo sviluppo, a causa della bassa potenza e dell'alto jitter (la fluttuazione degli Hz del PCL) non consente il funzionamento adeguato del test a frequenze superiori ai 500 Hz. Per stare sul sicuro si è impostato il valore a 250 ma si tenga presente che il solo scopo di questo target è la lettura dei valori joystick e che la rete TCP IP introduce comunque una sua latenza per cui questa velocità è adeguata.

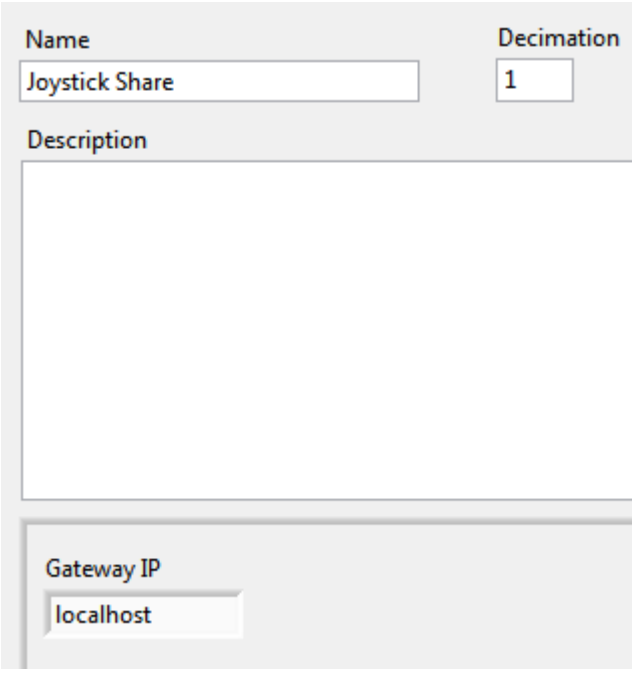
Hardware (Locale)

Questa sezione è vuota

Custom Devices (Locale)

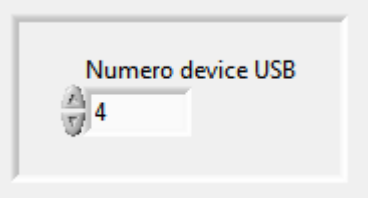


I CD realizzati con LabView vengono letti automaticamente da VeriStand all'avvio e resi disponibili in un apposito menù dal quale possono essere aggiunti ai target. Come si può notare la struttura appare identica a quella dell'FPGA personality e in effetti VS tratta i CD come componenti hardware esterni.



L'immagine sopra riportata é relativa al CD Joystick share come si può evincere dal nome. Notare la voce *Decimation:* come i modelli, anche i CD possono essere fatti funzionare ad una frequenza che è frazione di quella del PCL. Il numero indica quanti cicli PCL eseguire per ogni ciclo CD. Gateway IP rappresenta l'indirizzo dal quale viene avviato il Gateway ovvero il componente che si occupa di inviare ai giusti apparecchi tutte le parti del modello. Se non si hanno esigenze particolari il Gateway IP è localhost di default. La-

sciare la casella vuota in questa caso sarebbe corretto. Questo CD avrebbe potuto essere caricato sul target CF3000 ma si sarebbe ulteriormente caricato il PXI senza alcun beneficio. La schermata di configurazione di *Joystick USB* presenta la seguente casella:

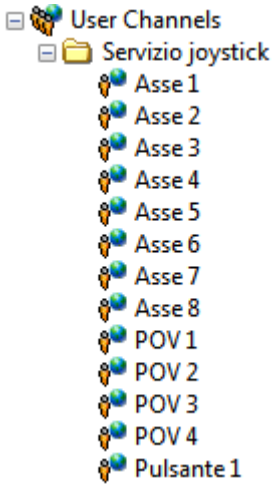


Il controller USB di Windows numera i vari device che vengono collegati. In teoria il valore di default se viene inserito un joystick è 0, in pratica dipende dal PC. Nel caso specifico il valore corretto è risultato 4 e purtroppo l'unica strada è andare per tentativi. Da qui è possibile impostare il valore. Per essere certi del funzionamento conviene collegare il joystick prima di avviare VS.

Simulation Models (Locale)

Questa voce risulta vuota

User Channels



Sono elencati solo i canali relativi a Servizio joystick, i quali risultano assolutamente identici a quelli di CF3000. Il CD Joystick USB trascrive qui le proprie letture, il CD Joystick share copia questi valori sugli stessi canali di CF3000 (lavora a senso unico). Questo è

l'unico modo di far comunicare due target diversi attraverso la rete TCP-IP, e inoltre si deve necessariamente rinunciare al sincronismo.

Calculated Channels (Locale)

Risulta vuoto

System Mappings

Source	Destination
LP0_OUT [CF3000/Calculated Channels/]	LP0_OUT [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
HP0_OUT [CF3000/Calculated Channels/]	HP0_OUT [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
TOP DIF [CF3000/User Channels/]	TOP DIF [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Dente Start Ciclo [CF3000/User Channels/Fasatura/]	Dente Start Ciclo [CF3000/Hardware/Chassis/FPGA/RIO0/Out
Offset_Dente [CF3000/User Channels/Fasatura/]	Offset_Dente [CF3000/Hardware/Chassis/FPGA/RIO0/Output
Fase T-F [CF3000/User Channels/Fasatura/]	Fase T-F [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]
Fase F-T [CF3000/User Channels/Fasatura/]	Fase F-T [CF3000/Hardware/Chassis/FPGA/RIO0/Output/]

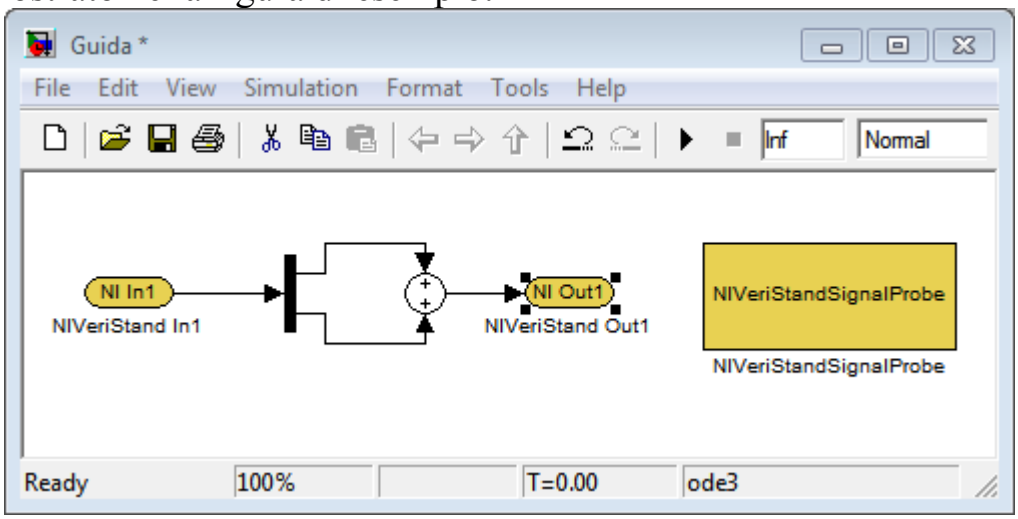
L'elenco è comune a tutti i target e questa volta comprende ben 124 voci. Qui si evidenzia uno degli attuali limiti di VeriStand, in quanto spostare un user channel in una sottocartella non è possibile, occorre cancellarlo e ricrearlo nella sottocartella. Sull'User Channel stesso non ci sono riferimenti ai collegamenti: questi vanno cercati nell'elenco, sia come sorgente che come destinazione, annotati a parte e ricreati a mano dopo lo spostamento. Questa è probabilmente la ragione principale per la quale conviene avere due test separati piuttosto che un modello aggiuntivo che filtra i canali per il sistema automatizzato e quello manuale.

Modelli matematici in SimuLink

I modelli SimuLink non sono direttamente compatibili con VeriStand ma richiedono alcuni accorgimenti a seconda del formato che si intende utilizzare.

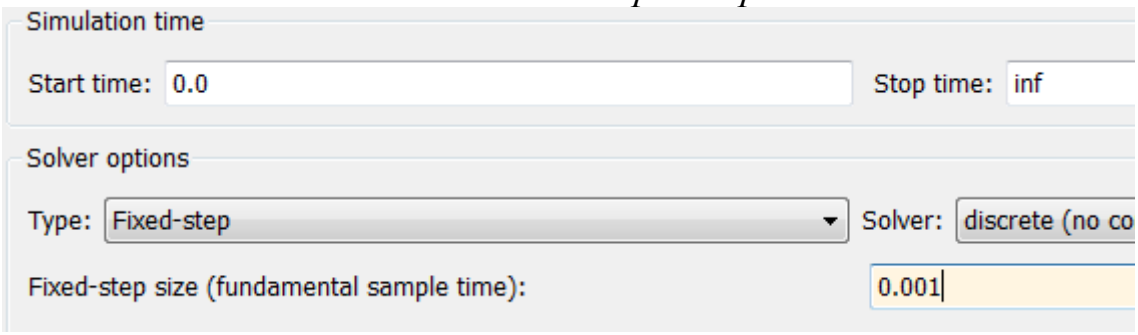
Modelli .mdl

Possono essere utilizzati solo su PC però permettono di correggere e testare il modello rapidamente. Prima di utilizzare questa modalità conviene fare due considerazioni. La prima è che i modelli non compilati sono poco performanti: uno dei modelli testati, di media complessità, è risultato oltre 20 volte più lento (da 1000 a 50 Hz) della sua copia compilata. La seconda è che la compilazione dello stesso richiede poche decine di secondi. Per utilizzare i modelli in formato MDL occorre che gli stessi contengano gli allacciamenti specifici di VeriStand e il blocco *NiVeriStandSignalProbe*. Come mostrato nella figura di esempio.



Il termine della simulazione deve essere settato su Inf perché sarà VeriStand ad occuparsi dell'avvio e dell'arresto. Se ci sono blocchi che integrano o derivano rispetto al passo temporale è importante

settare quest'ultimo su un valore compatibile con la frequenza settata su VeriStand. Queste impostazioni possono essere trovate cliccando su *Tools → Real-Time Workshop → Options*



Durante l'uso di questi modelli MatLab deve essere avviato, VeriStand gli comunica i dati e legge quelli da esso generati, per questo le performance diventano scadenti.

Modelli dll

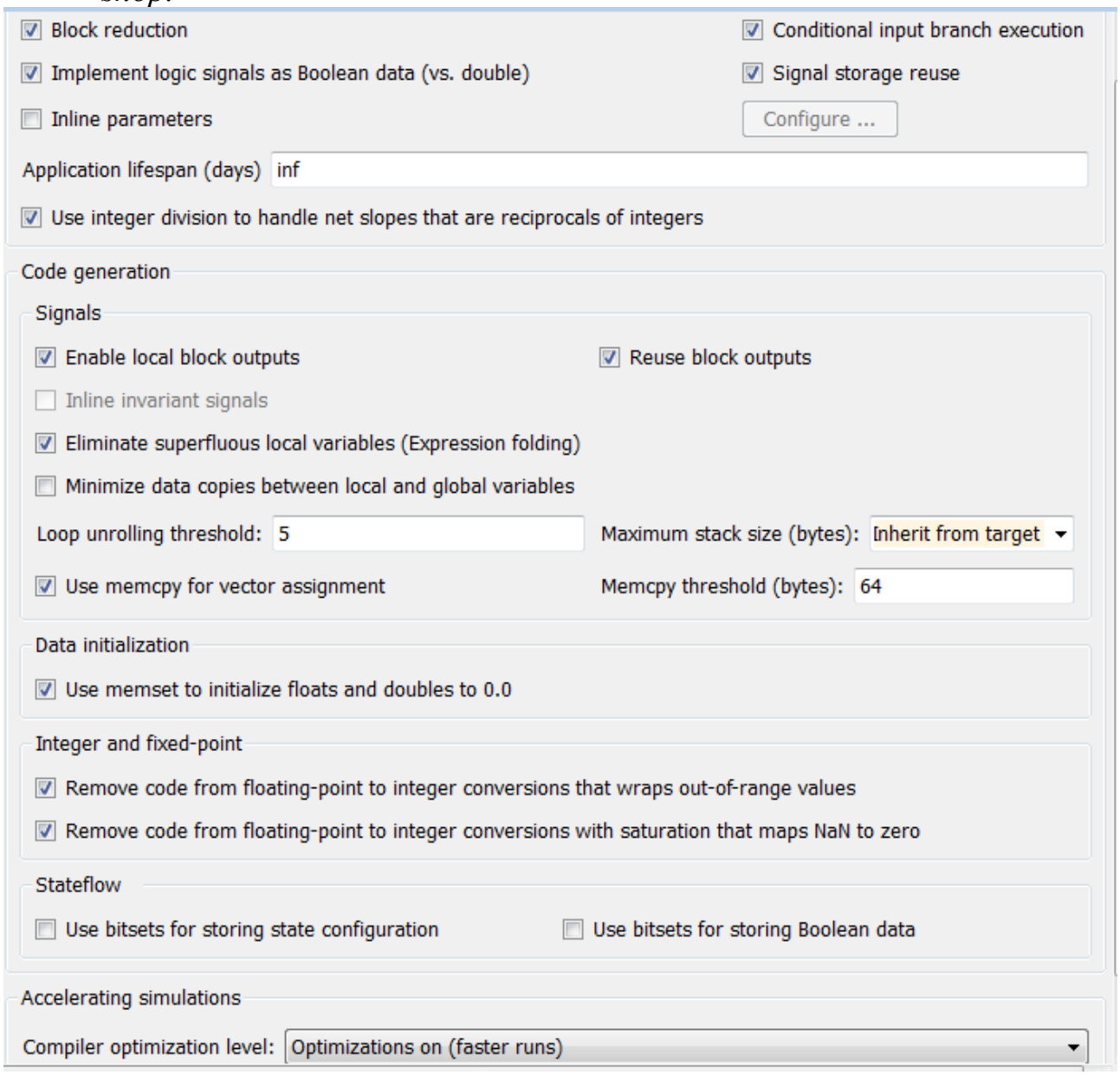
Questi modelli vengono compilati e possono essere inviati al PXI. Una volta prodotti non necessitano dell'appoggio di MatLab e si può omettere il blocco *NiVeriStandSignalProbe*. Le altre considerazioni rimangono le medesime della versione mdl. Un aspetto importante dell'utilizzo di dll è l'ottimizzazione: man mano che il modello si complica il suo peso saturerà sempre più la capacità di calcolo del processore. Visto che il PXI deve occuparsi anche di altri compiti quando si supera l'80% di saturazione cominciano a sorgere dei problemi: alcuni cicli non riescono ad essere completati in tempo e le informazioni trasmesse agli altri componenti del progetto risultano errate o addirittura assenti. Oltre alla semplificazione del modello o alla riduzione della frequenza di funzionamento si può agire sull'ottimizzazione. La prima ottimizzazione che si può fare è evitare al processore calcoli inutili, le seguenti strategie sono spesso facili da attuare e abbastanza incisive

- Se possibile raggruppare le moltiplicazioni
- Sostituire le divisioni per una costante con moltiplicazioni

per il reciproco (il processore impiega normalmente un ciclo di calcolo per una moltiplicazione e 20 per una divisione)

- Ove possibile ridurre il numero di divisioni raggruppando il denominatore
- Avviare *Model Advisor* ed eseguire il test *Identify questionable blocks within the specified system* e verificarne gli eventuali suggerimenti

La seconda strategia di ottimizzazione è specificare al compilatore per che tipo di processore va generato il codice. Per farlo si può accedere alla scheda *Optimization* nelle opzioni del *Real-Time Workshop*.

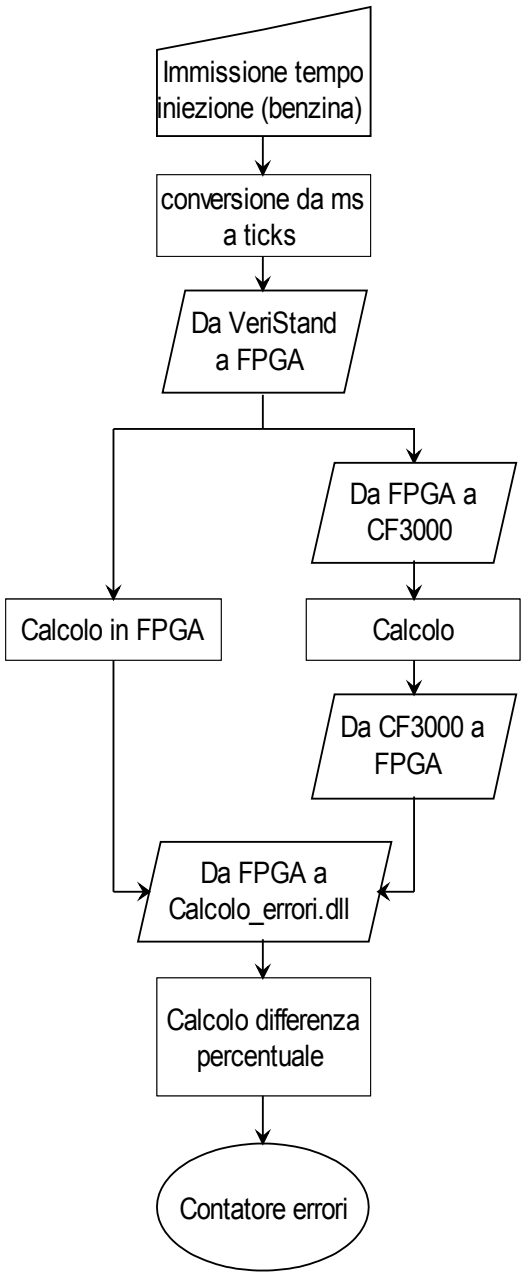


La configurazione riportata funziona correttamente si PXI. Si possono provare altre combinazioni ma occorre assicurarsi del buon funzionamento del codice compilato. Notare che nel primo blocco Inline Parameters è deselezionato: spuntandolo si produce codice non compatibile con PXI.

Calcolo degli errori

Struttura generale

Scopo primario del progetto è la verifica del corretto funzionamento della ECUG CF3000. Questa verifica è essenzialmente svolta confrontando i tempi iniezione prodotti dalla centralina stessa con quelli calcolati da un modello matematico. In VeriStand non è possibile imporre un tempo di calcolo dei modelli proporzionale al regime di giri del motore quindi è necessario imporre una velocità di esecuzione che garantisca la corretta analisi dei cilindri alla velocità di rotazione massima consentita. A 6000 giri/min, ogni cilindro compie 50 cicli al secondo (motore 4 tempi) quindi il modello motore richiede in teoria 200 elaborazioni al secondo. Per come è strutturato il calcolo errori, è necessario che vengano fatti come minimo 2 cicli PCL per ogni ciclo cilindro ma, considerata la necessità di lavorare con componenti non sincronizzate, è preferibile che siano in numero maggiore. E' stato impostato un regime di funzionamento di 1000 Hz che dovrebbe essere sufficiente in ogni condizione tuttavia, non essendo presenti derivazioni o integrazioni, l'operatore può incrementare liberamente questa frequenza in funzione del proprio Hardware. In figura è mostrata la procedura del calcolo errori seguita da una sua analisi dettagliata.



Immissione tempo iniezione

I valori, da immettere in millisecondi, sono i tempi iniezione, calcolati per benzina, che la ECU trasmetterebbe alla ECUG. Possono essere inseriti manualmente nelle apposite caselle del workspace oppure automaticamente, attraverso un profilo di stimolo o il modello matematico del motore incluso nel progetto. Una volta immessi, questi valori vengono scritti nei canali *Targets/CF3000/User*

Channels/Gestione iniettori/Duration Inj # (fxchannel) dove # indica un numero da 1 a 4

Conversione da ms a ticks

I ticks sono l'unità di misura del tempo in FPGA: ognuno di essi corrisponde ad uno scatto nell'orologio interno della scheda il quale lavora a 40 Mhz. La conversione viene effettuata in VeriStand leggendo il valore nei canali precedenti e moltiplicando per 40000 con dei Calculated Channels: *Targets/CF3000/Calculated Channels/Duration inj #*

Da VeriStand a FPGA

VeriStand è in grado di comunicare con le schede FPGA attraverso strutture chiamate FPGA personality. Queste ultime, create in LabView, possono essere semplicemente dei ponti per leggere segnali da una morsettiera esterna oppure possono contenere calcoli anche complessi. I canali *Duration inj #* sono collegati agli omonimi ingressi della FPGA personality caricata in VeriStand.

Calcolo in FPGA

All'interno della FPGA personality è stato caricato il modello matematico utilizzato dalla ECUG per effettuare la conversione. Ci si aspetta che il risultato di questo calcolo coincida con quello prodotto dalla ECUG. VeriStand legge questi valori dai canali *Duration_InjAttuato_#* dell'FPGA e li immette sugli omonimi canali in ingresso al modello *Calcolo_errori.dll*

Da FPGA a CF3000, calcolo, DaCF3000 a FPGA

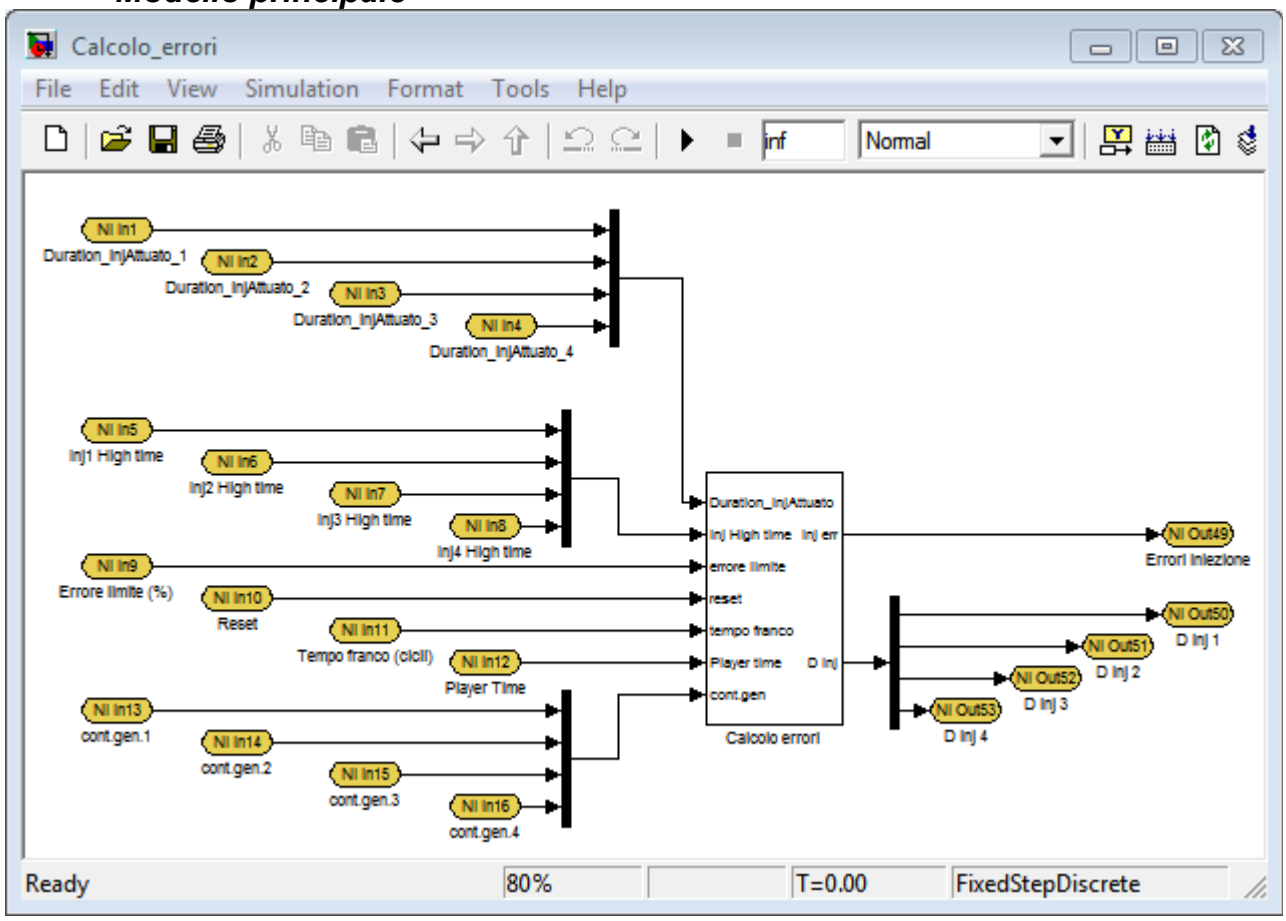
L'FPGA, oltre ad utilizzare i valori in ingresso per i propri calcoli, li trasmette alla ECUG. Questo è un passaggio delicato poiché la comunicazione non avviene attraverso un canale digitale bensì attraverso la generazione di un' onda quadra di periodo pari al tempo ciclo e con durata del segnale alto (5V) pari al tempo di iniezione. La centralina misura questo segnale, applica le correzioni necessa-

rie e lo ritrasmette all'FPGA. Durante questo passaggio possono essere introdotti errori limitati dovuti all'approssimazione dei due dispositivi in quanto questi non sono sincronizzati tra loro. I segnali elaborati vengono letti sui canali *Inj# High time* in uscita dall'FPGA e trasmessi ai canali omonimi in entrata al modello *Calcolo_errori.dll*

Calcolo differenza percentuale

Il modello *Calcolo_errori.dll* prende i segnali prodotti dall'FPGA e dalla ECUG e li confronta. Se la differenza percentuale supera un valore stabilito conta un errore. Gli errori vengono sommati e mostrati in un contatore resettabile. Il valore di soglia dell'errore è regolabile a piacere. Il risultato viene mostrato sul workspace.

Modello principale



In figura è mostrata la struttura di collegamento del modello. Per snellire i calcoli interni i canali relativi ai singoli cilindri sono stati fusi insieme. In entrata si ha:

- *Duration_InjAttuato_#* immette i tempi calcolati dall'FPGA
- *Inj# High time* immette i tempi calcolati da CF3000
- *Errore limite (%)* immette la differenza massima tollerabile prima di segnalare errori
- *Reset* se posto a 1 azzera il contatore
- *Tempo franco (cicli)* immette il numero di cicli per i quali il contatore resta comunque a 0 Serve per evitare segnalazioni anomale in fase di avvio del test
- *Player Time* legge il valore di un contacicli per poterlo con-

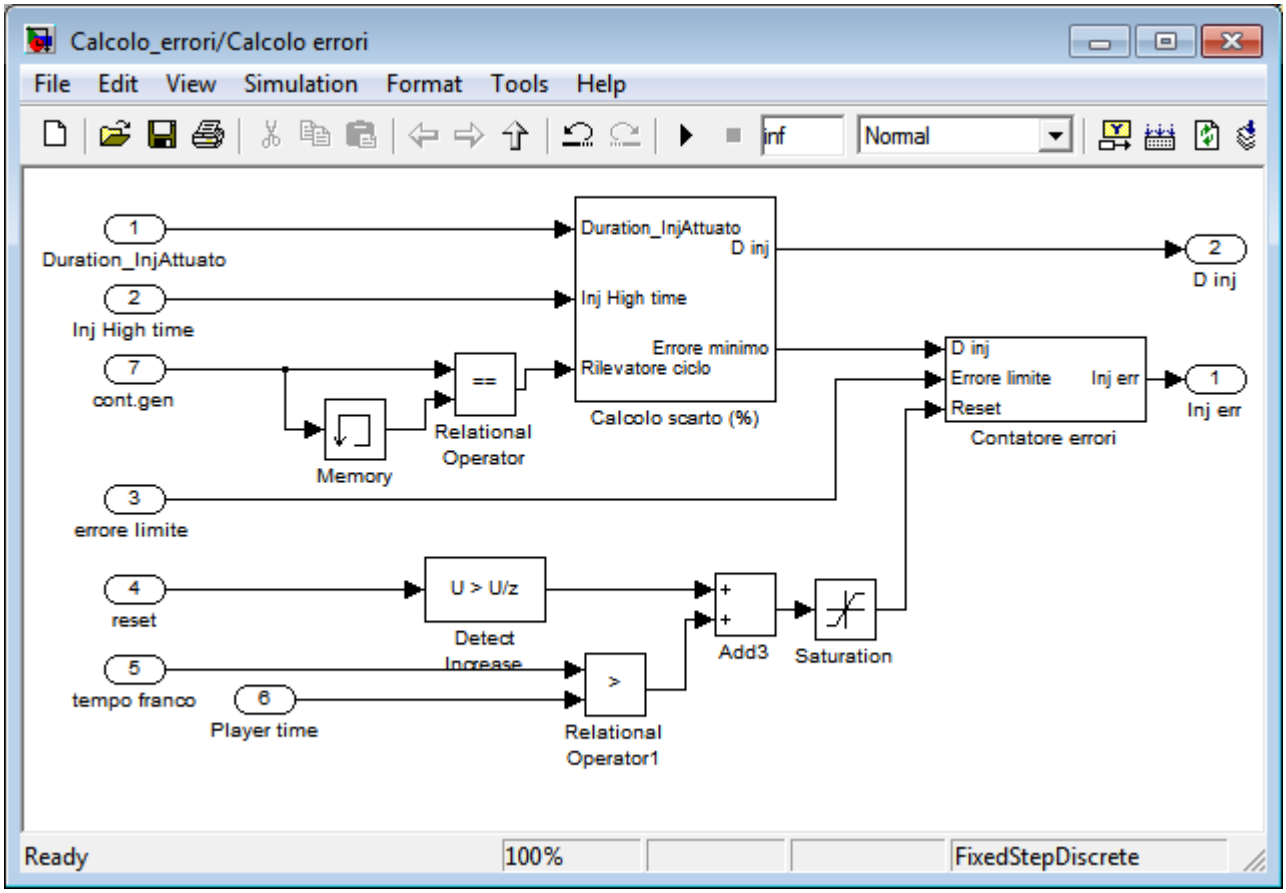
frontare a Tempo franco

- *cont.gen.#* sono gli ingressi dei contatori di ciclo per i singoli cilindri

In uscita si ha:

- *Errori iniezione* è il contatore totale degli errori
- *D inj #* segnala la differenza percentuale sui rispettivi canali calcolata dal modello

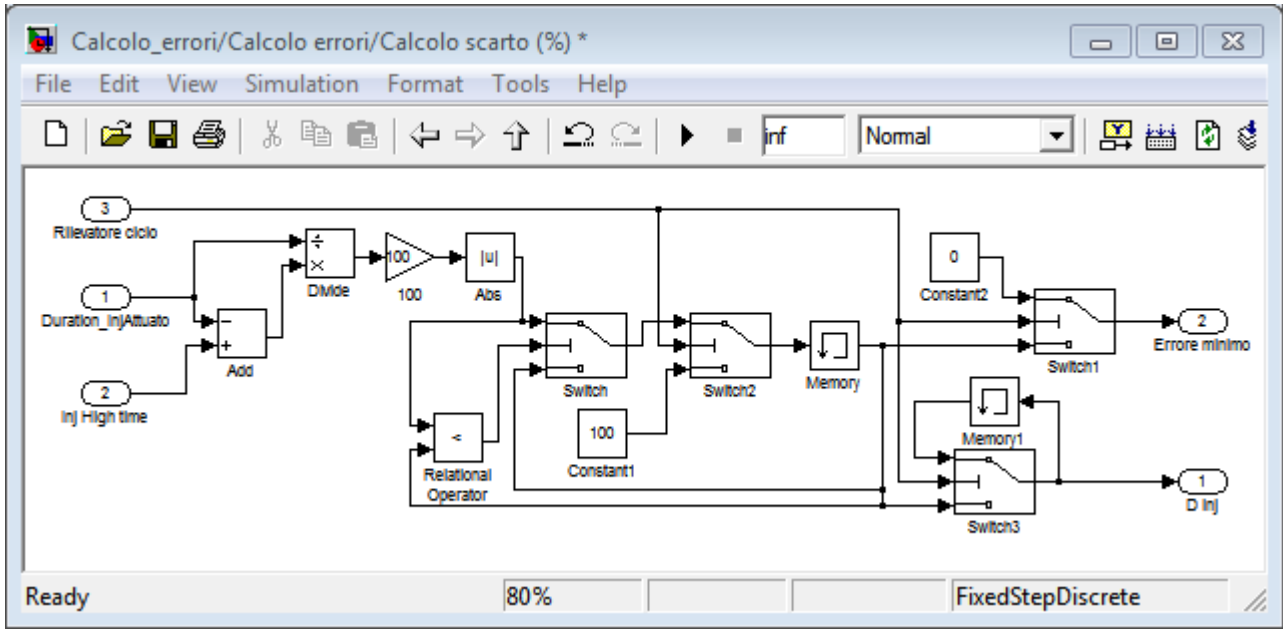
Sottosistema Calcolo errori



Il modello non lavora in sincrono con il motore quindi è necessario che venga immesso un segnale che rileva il cambio di ciclo nel cilindro. Questo segnale è *cont.gen.#* che però è un numero progressivo. Il blocchetto Memory registra il valore del contatore e al ciclo (del modello) successivo lo trasmette all operatore ==. Questo con-

fronta i due valori ed emette 1 se il ciclo cilindro non è cambiato, 0 se è cambiato. Nella parte inferiore dello schema è visibile il controllo del segnale di reset. Il blocco *Detect Increase* trasmette 1 se rileva un innalzamento del segnale, 0 altrimenti; in questo modo il pulsante Reset nel workspace funzionerà solo nel ciclo in cui è premuto invece che tutto il tempo. L'operatore > emette 1 per tutti i cicli nei quali *Tempo Franco* è > di *Player time*. Il saturatore evita che il casi particolari la somma dei segnali di reset superi il valore 1.

Sottosistema Calcolo scarto %



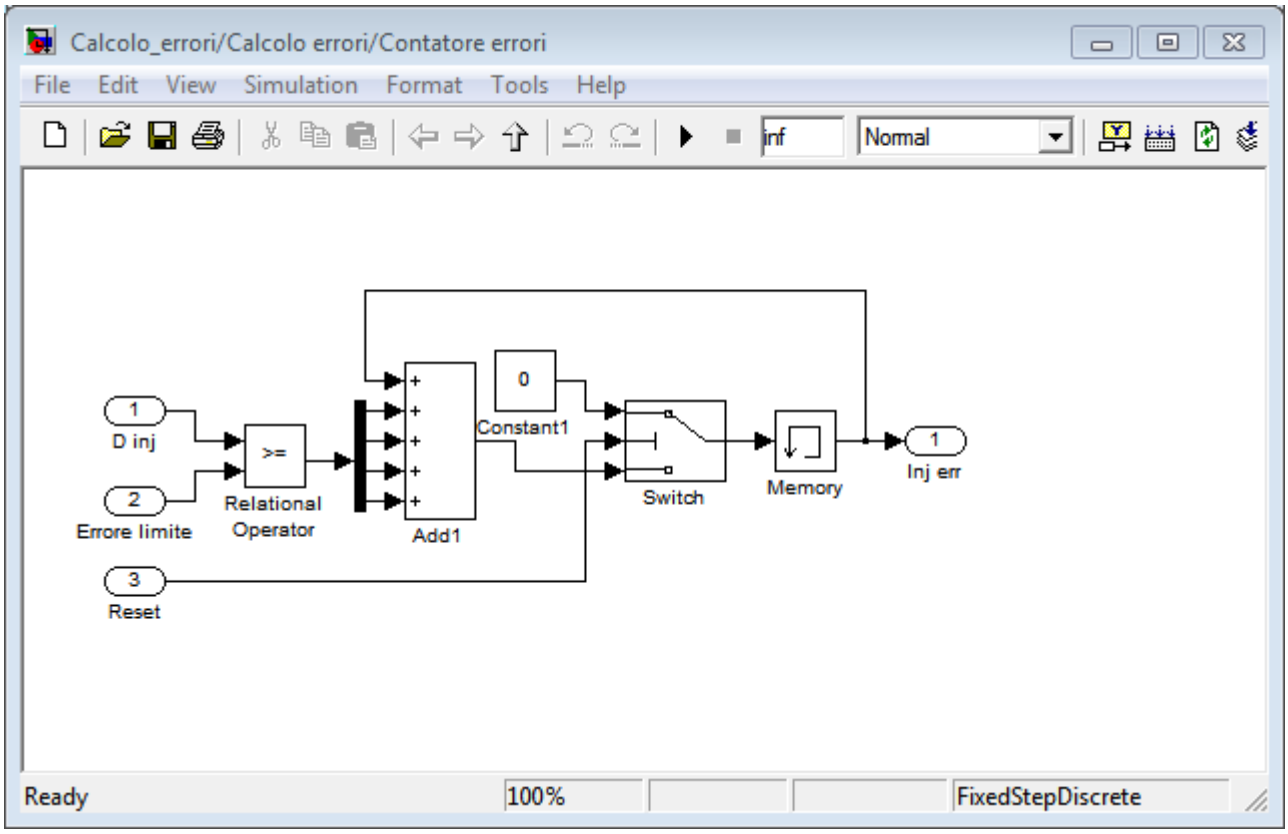
Il calcolatore dello scarto % non rileva gli errori, si limita a fornire la differenza percentuale tra la durata di iniezione calcolata dalla centralina *(Inj# High time,* segnale 2) e quella calcolata dall'FPGA (*Duration_InjAttuato,* segnale 1) calcolata in relazione a quest'ultima. Questo calcolo è devoluto ai blocchi in alto a sinistra nello schema: viene fatta la differenza tra i segnali 1 e 2 (blocco add), il risultato viene diviso per il segnale 1 (blocco Divide), moltiplicato per 100 (blocco gain) e viene assunto il valore assoluto del segnale (blocco Abs). A questo punto è bene ricordare che il segnale di iniezione dell'FPGA è un'onda quadra generata una volta per ciclo del

cilindro e inviato alla ECUG che lo elabora e lo restituisce. L'FPGA trasmette poi a VeriStand il valore digitale del segnale generato da lui stesso e di quello misurato in uscita della ECUG. I due segnali, a seconda del regime di giri, della durata dell'iniezione e dell'architettura della ECUG saranno sfasati di un numero di cicli PCL non prevedibile. Tutta la struttura a valle del blocco Abs serve a confrontare i segnali corretti sfruttando il seguente procedimento:

- Se nel precedente ciclo PCL il contatore cicli del cilindro è salito di valore, il segnale 3 risulta 0 e il calcolo scarto % si inizializza. I blocchi *Switch 1, Switch 2 e Switch 3* lasciano passare il segnale in basso. Il blocco *Memory* invia a *Errore minimo* e a *D inj* il valore precedentemente memorizzato quindi registra il valore 100. Il blocco *Memory 1* registra il valore *D inj*. Tutto quello che giunge al blocco *Switch* viene eliminato.
- Se nel precedente ciclo PCL il contatore cicli del cilindro è rimasto invariato, il segnale 3 risulta 1. Il blocco *Memory* emette 100, questo valore viene confrontato con il calcolo di differenza percentuale. Questa differenza può riguardare segnali emessi nello stesso ciclo cilindro o in quello precedente. Ad ogni ciclo PCL il blocco *Memory* emette il valore precedente e registra quello attuale. Il blocco *Relational operator,* assieme al blocco *Switch,* fa sì che solo il valore minore venga conservato. E' lecito supporre che in regime transitorio i valori con la differenza inferiore siano quelli legati al medesimo ciclo cilindro. I blocchi *Switch 1* e *Switch 3* fanno passare il segnale alto ed emettono il valore 0 per il canale di uscita 2 e il valore calcolato precedentemente per il canale *D inj*.

Questa struttura non può rilevare differenze maggiori del 100%, comunque erronee, e richiede un minimo di 2 cicli PCL per ciclo cilindro altrimenti emette sempre 100 come differenza percentuale.

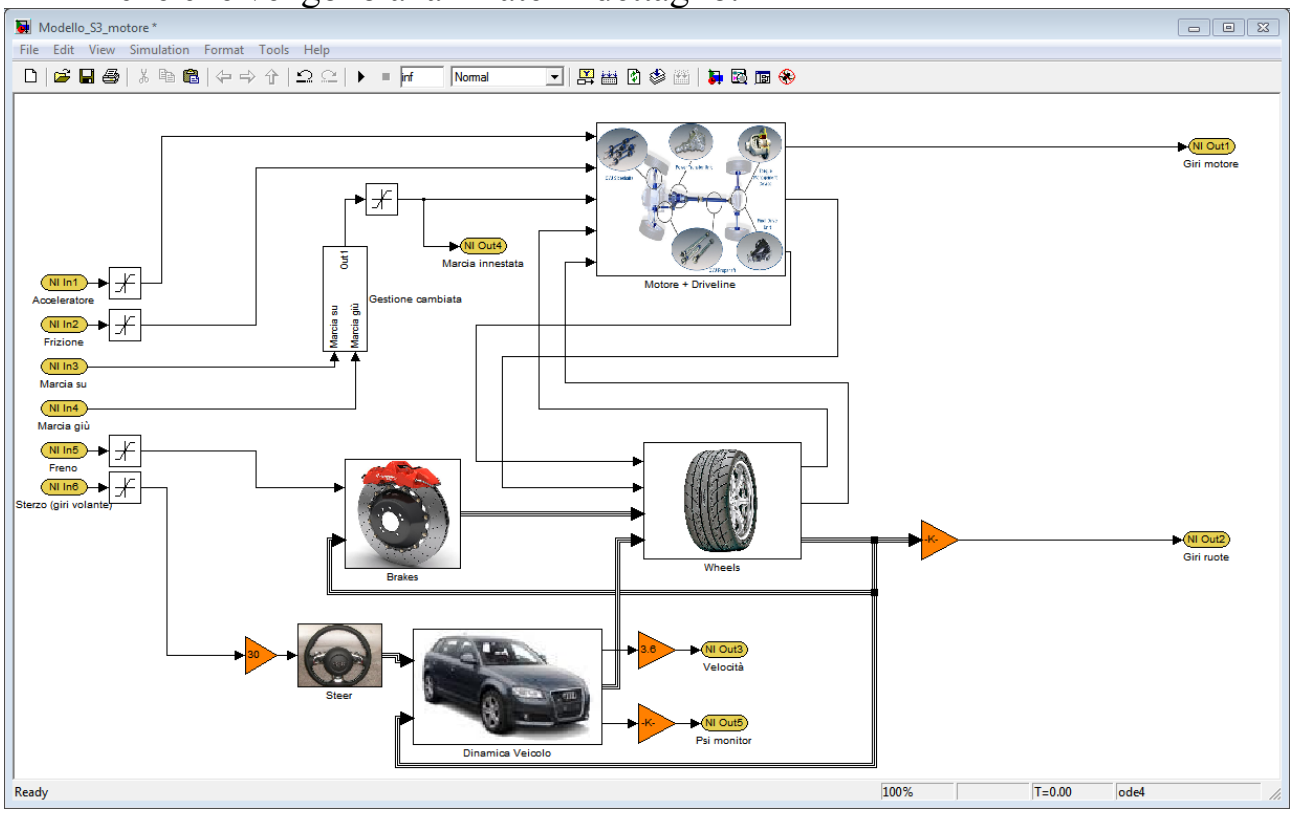
Sottosistema Contatore errori



Se *D inj* è maggiore del limite preimpostato, il blocco *Relational operator* emette il valore 1, altrimenti emette 0. Ad ogni ciclo viene fatta la somma di tutti i canali in ingresso e del valore del precedente ciclo. Il segnale di reset azzera il contatore tramite lo switch. *Inj err* è il segnale che viene inviato a VeriStand.

Modello_S3_motore.mdl

L'idea della modalità manuale è che l'utente controlli un veicolo virtuale e questo, simulando un'ECU, trasmetta all'ECUG dei parametri verosimili. Il test deve funzionare in real time quindi è necessario mantenere un modello veicolo relativamente semplice. Si è deciso di adattare un modello preesistente che funziona a valori medi utilizzando delle mappe per la coppia motore, effetti di pompaggio ecc. Una particolarità di questo modello è che è basato su un motore turbodiesel. L'unica controindicazione è che le mappe di coppia sono limitate a 4500 RPM massimi, oltre questa soglia il modello diventa del tutto irrealistico quindi è stato inserito un meccanismo di limitazione dei giri. Sono state apportate alcune modifiche che vengono analizzate in dettaglio.



Ingressi

- Acceleratore: va da 0 (pedale alzato) a 100 (pedale premuto). Il saturatore serve a limitare entro questi valori il dato in in-

gresso perché sul workspace di VeriStand è possibile inserire qualsiasi valore si preferisca.

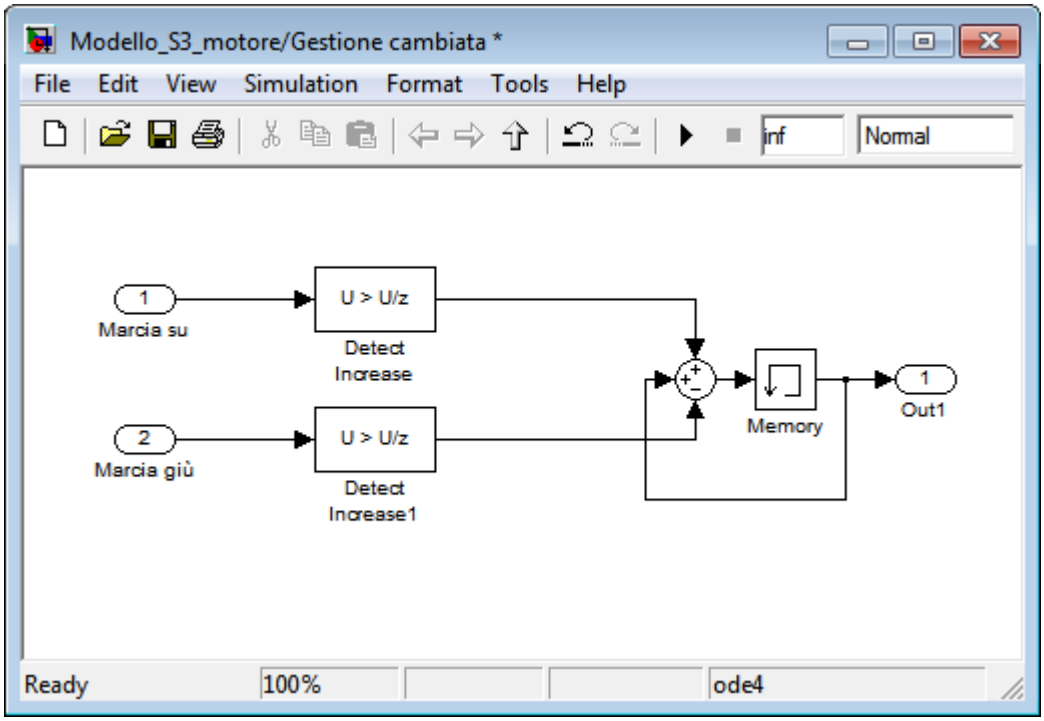
- Frizione: va da 0 (pedale alzato) a 1 (pedale premuto)
- Freno: va da 0 (pedale alzato) a 1 (pedale premuto)
- Marcia su: riceve il segnale booleano del pulsante per salire di marcia
- Marcia giù: riceve il segnale booleano del pulsante per scendere di marcia
- Sterzo: è la posizione del volante in giri, va da -1(tutto a destra) a 1 (tutto a sinistra)

Uscite

- Giri motore: è la velocità di rotazione in giri al minuto
- Giri ruota: sempre in giri al minuto, è un segnale vettoriale che contiene i 4 valori singoli
- Velocità: è il modulo della velocità baricentrica del veicolo (non necessariamente longitudinale)
- Psi monitor: indica la direzione del veicolo rispetto all'angolo (0) di partenza
- Marcia innestata: legge la posizione del cambio
- Acceleratore attuato: (sottosistema Motore + Driveline) lettura della posizione acceleratore
- Frizione attuata: (sottosistema Motore + Driveline) lettura della posizione frizione
- Anticipo Coil: (sottosistema Controllo iniettori) segnale di anticipo per le bobine
- Anticipo Inj: (sottosistema Controllo iniettori) segnale di anticipo per gli iniettori

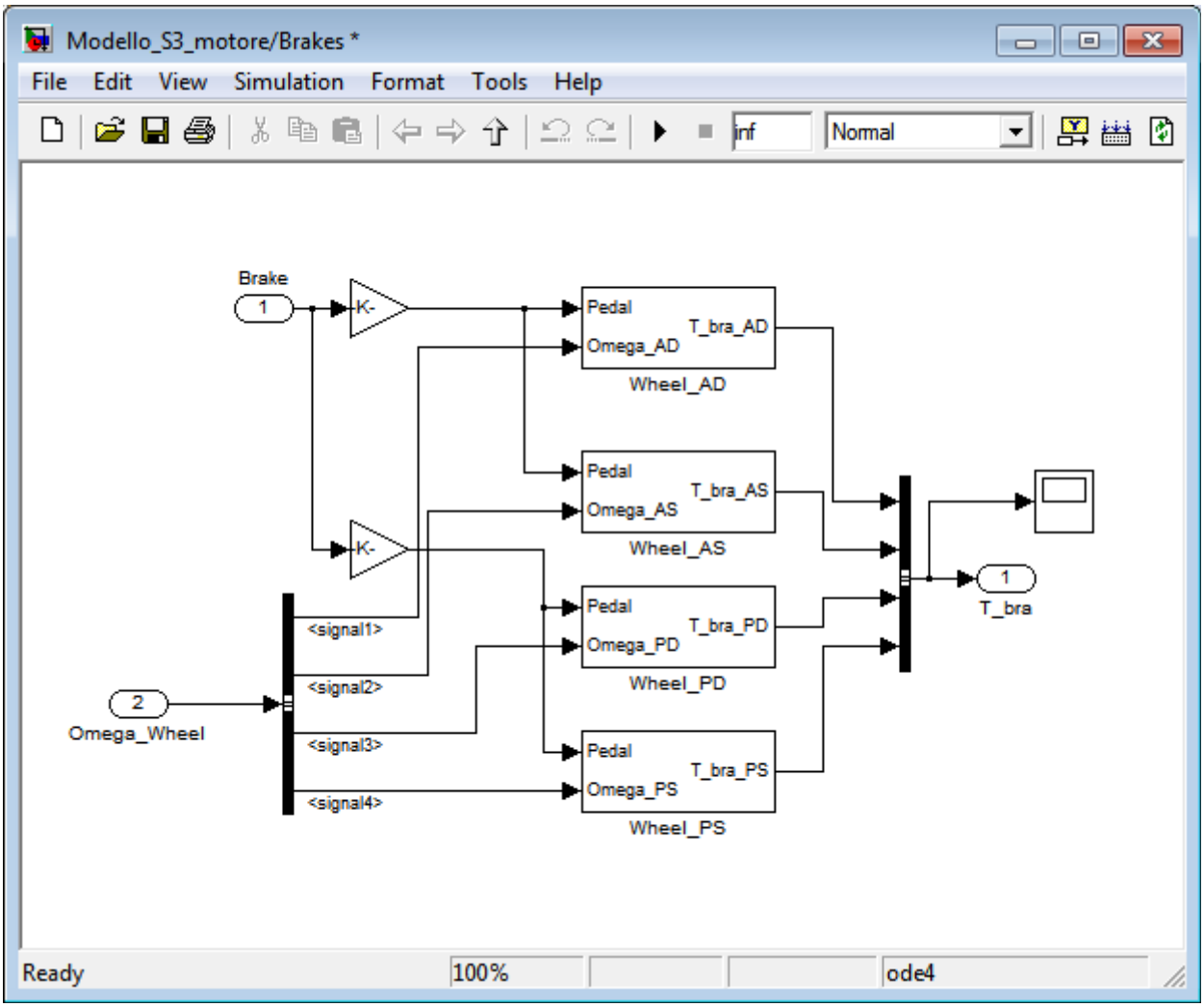
- Tempo iniezione: (sottosistema Controllo iniettori) segnale di durata dell'iniezione

Sottosistema Gestione cambiata



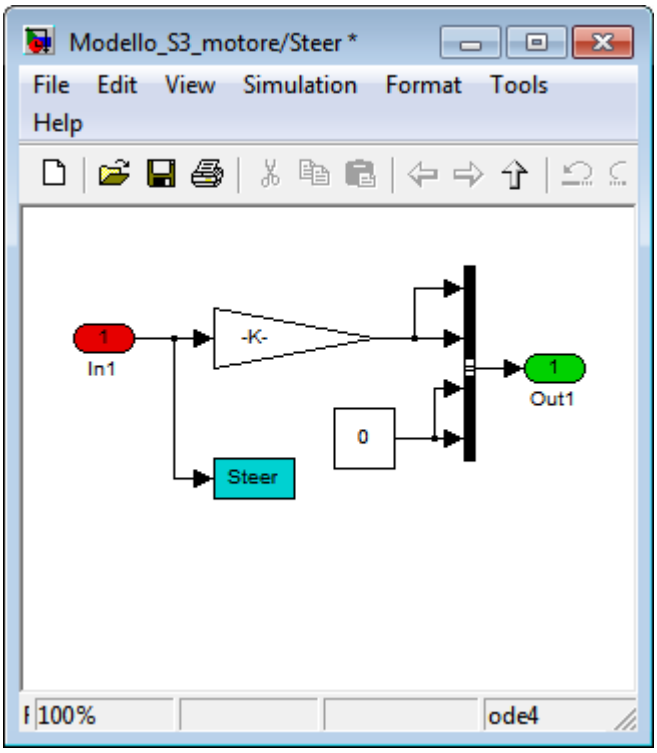
E' stato inserito per poter controllare il cambio con il joystick. I blocchi *Detect Increase* danno in uscita il valore 1 solo se rilevano un incremento del segnale dal ciclo precedente, in tal modo si evita che la pressione del pulsante faccia scorrere tutte le marce fino all'estremo. Il blocco *Memory* e quello *somma* tengono conto della marcia impostata precedentemente e della pressione del pulsante.

Sottosistema Brakes



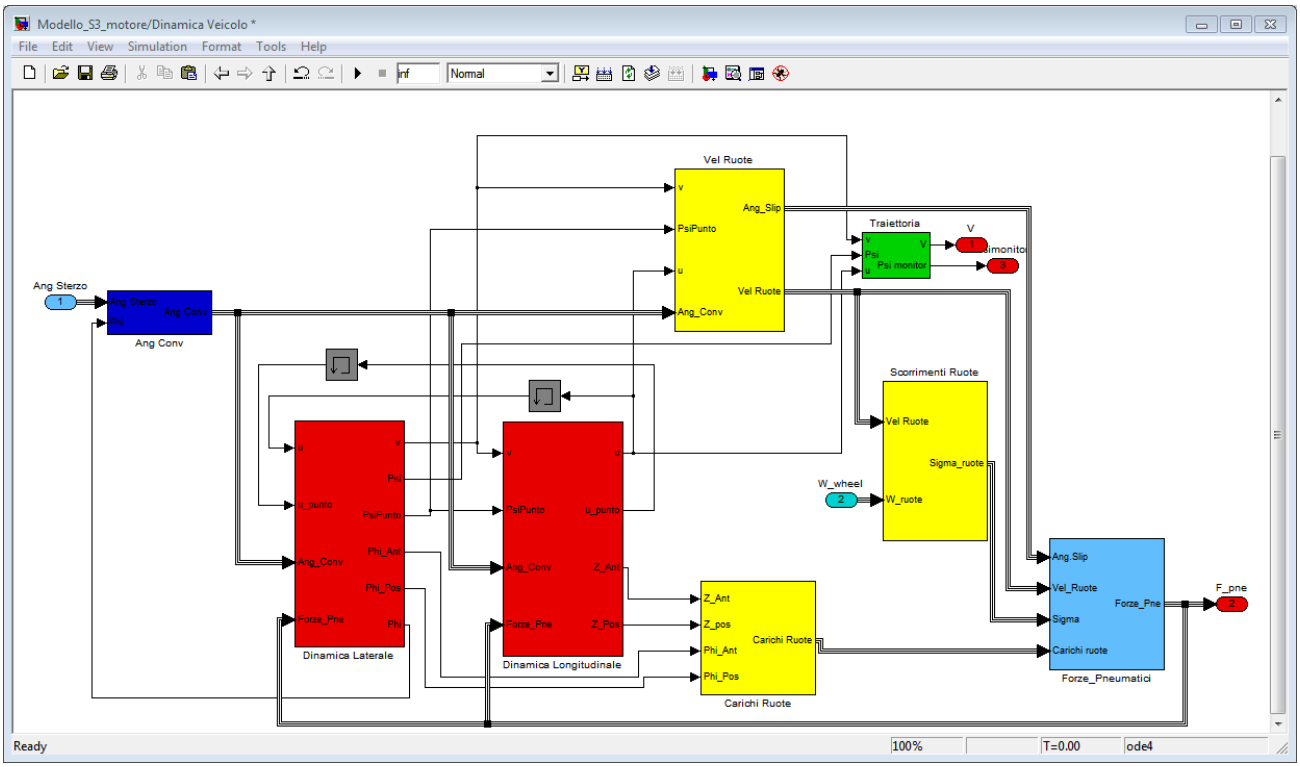
Genera una forza di frenata sulle ruote, proporzionale alla pressione del pedale e sempre opposta al moto. Questo sottosistema non è stato modificato.

Sottosistema Steer



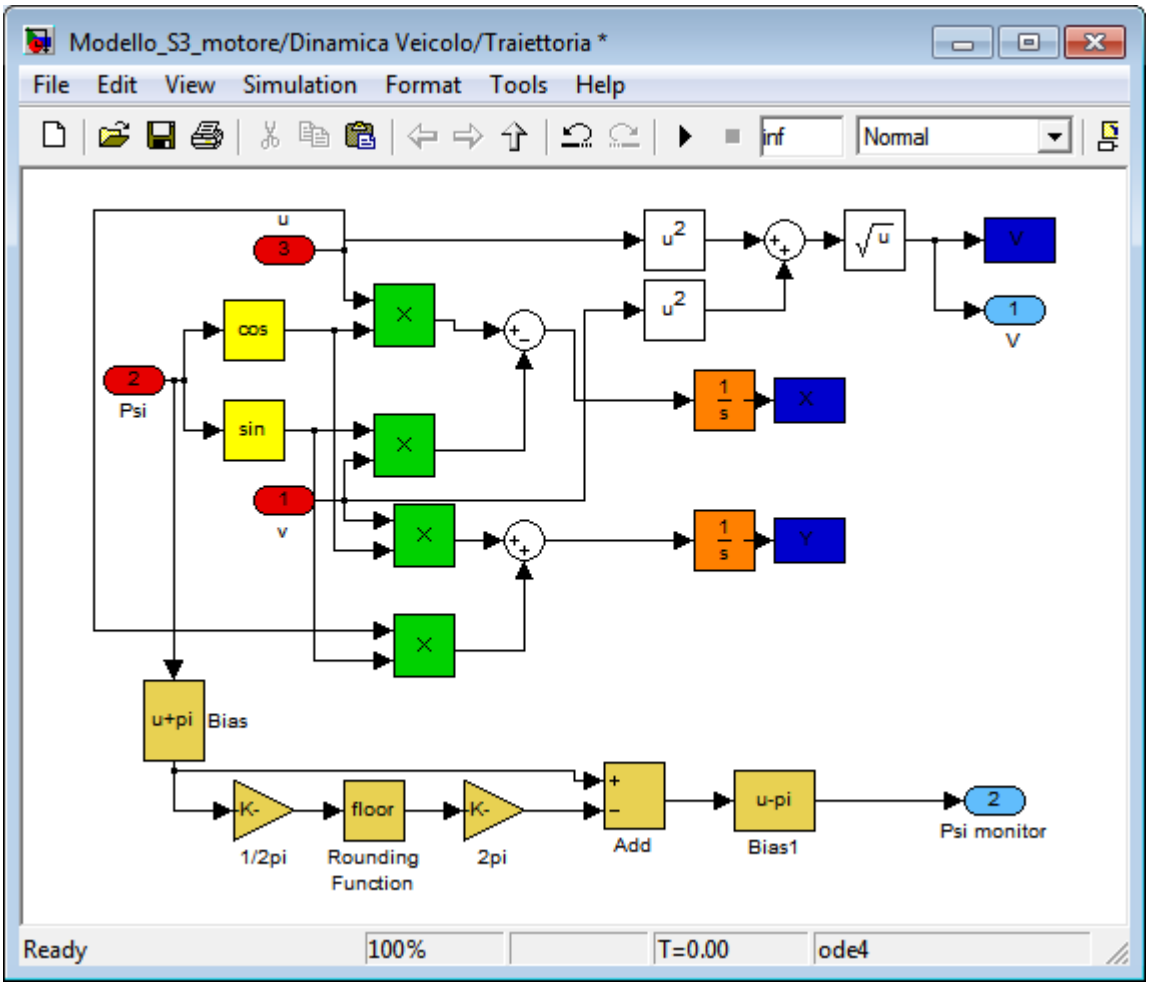
Si limita a trasformare i giri del volante in angolo di rotazione delle ruote rispetto all'asse longitudinale del veicolo. Non è stato modificato

Sottosistema Dinamica Veicolo



Legge i segnali di angolo di sterzo e velocità ruote e calcola la velocità del veicolo, la traiettoria e le forze che agiscono sugli pneumatici. E' stato modificato nel sottosistema *Traiettoria*.

Sottosistema Traiettoria



Questo sistema riceve in ingresso

- *u*: componente longitudinale della velocità baricentrica in m/s
- *v*: componente trasversale della velocità baricentrica in m/s
- *Psi*: angolo del veicolo rispetto ad un riferimento fisso

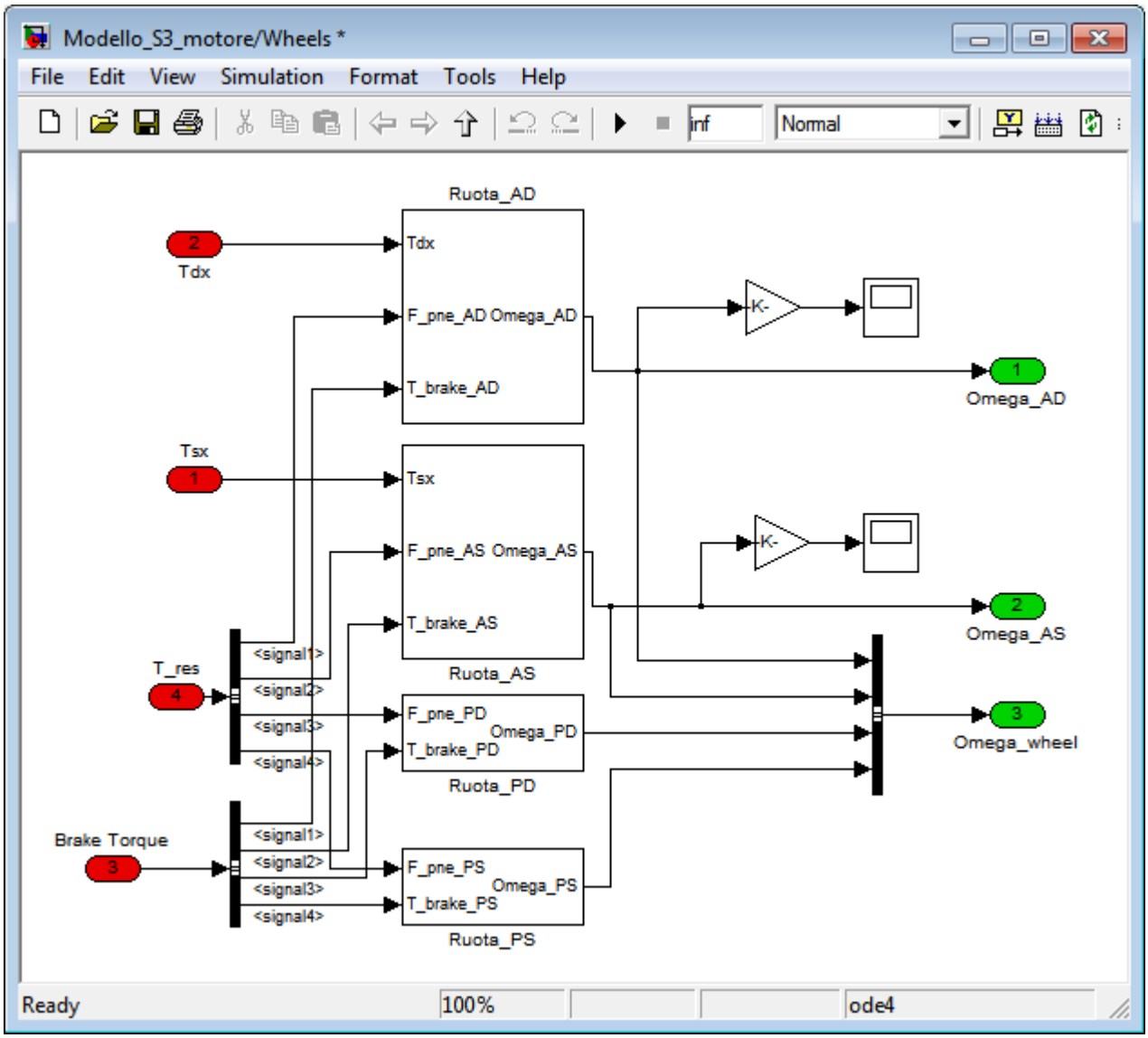
Con questi dati calcola

- *V*: modulo della velocità del veicolo in m/s. Calcolato secondo il teorema di Pitagora (blocchi u^2 , *somma* e $\sqrt{u}$)
- *X,Y*: coordinate della posizione del baricentro su un

piano cartesiano. Vengono ricavati sin e cos Psi, questi vengono moltiplicati per u e v ottenendo le componenti verticali e orizzontali delle due velocità che sommate e integrate danno la posizione.

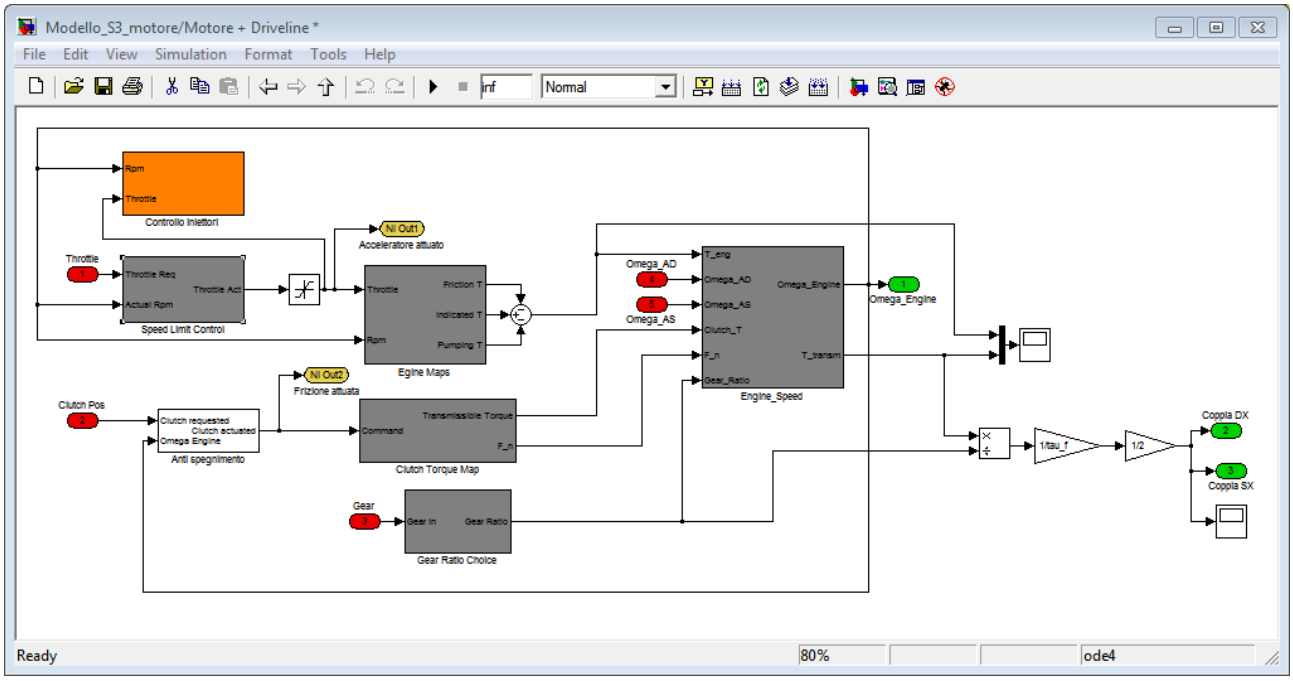
- *Psi monitor*: la struttura in giallo si limita a convertire l'angolo che va da -inf a inf in un angolo che va da -pi a pi. Questa operazione è necessaria per poter visualizzare la direzione del veicolo usando gli strumenti disponibili nel workspace di VeriStand. La catena *1/2pi floor 2pi* ricava il numero di giri interi effettuato dall'angolo dopodiché questo valore viene sottratto al valore di Psi. Si ottiene un angolo da 0 a 2π, il blocco *Bias1* sottrae π consentendo una lettura da -π a π. Il blocco *Bias* aggiunge π all'angolo riportando a 0 il punto di partenza.

Sottosistema Wheels



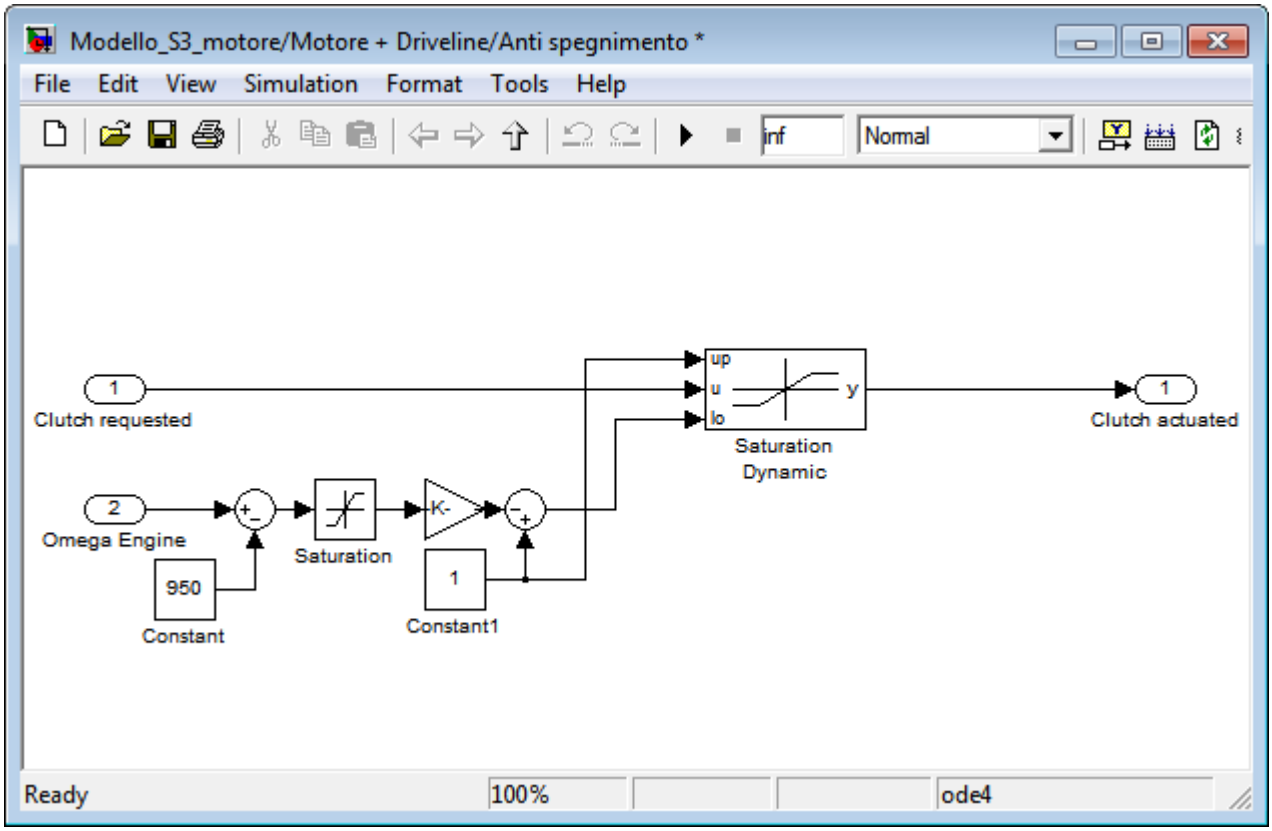
Legge le forze esterne che agiscono sugli pneumatici, la torsione trasmessa dai semiassi e le forze frenanti e calcola la velocità di rotazione delle ruote. Non è stato modificato

Sottosistema Motore + Driveline



Questo sistema legge la posizione della farfalla, della frizione, del cambio e la velocità di rotazione delle ruote anteriori. La versione originale calcola la velocità di rotazione del motore e la coppia sulle ruote anteriori. Sono stati aggiunti i canali in uscita *Acceleratore attuato* e *Frizione attuata* che hanno un puro scopo di monitoraggio, Sono stati aggiunti anche due sottosistemi: *Antispegnimento* e *Controllo iniettori.*

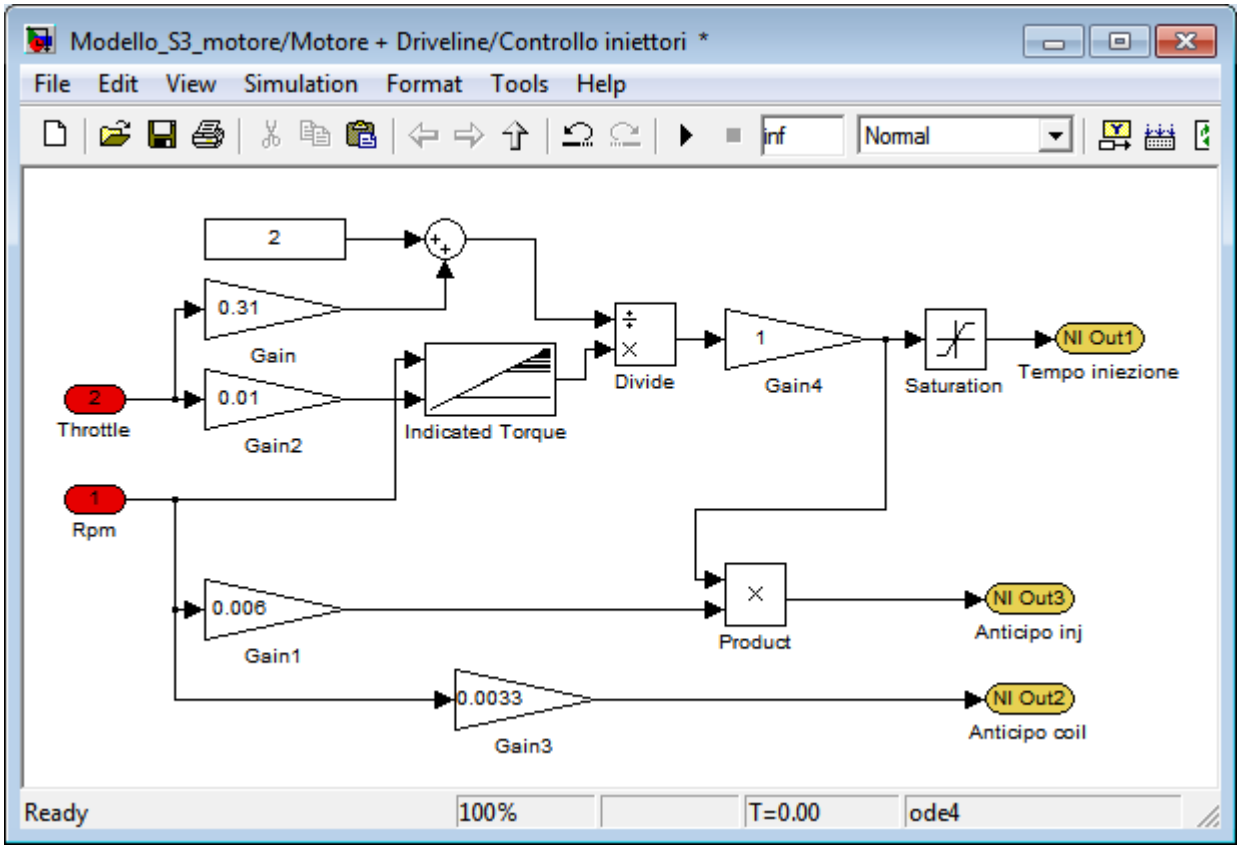
Sottosistema Anti spegnimento



In fase di utilizzo è possibile manovrare in modo da spegnere il motore però non è possibile riaccenderlo: è necessario riavviare il test. Questa procedura richiede oltre un minuto e riporta tutti i canali controllabili al valore di default quindi è assolutamente da evitare. Si è aggirato il problema creando un sottosistema che preme gradatamente la frizione quando il regime di giri è troppo basso. Grazie a questo stratagemma è anche possibile arrestare totalmente il veicolo e poi rimetterlo in movimento. Anzitutto al regime motore viene sottratto il valore 950 poi si passa in un saturatore con limiti 0 e 50. Quando gli RPM scendono a 950 o meno il valore in uscita è sempre 0, quando superano i 1000 il valore in uscita è sempre 50. Il segnale viene diviso per 50, viene sottratto 1 e il risultato è impostato come valore minimo in un saturatore dinamico il cui valore massimo è sempre 1. In questo saturatore entra il valore di pedale impostato dall'utente. Il valore 1 corrisponde a pedale pre-

muto quindi frizione disinnestata, il valore 0 corrisponde a pedale rilasciato, frizione innestata. Se il regime motore scende al disotto dei 1000 RPM il saturatore dinamico impone la graduale pressione del pedale, fino al totale disinnesto a 950 RPM. La coppia trasmissibile dalla frizione non è proporzionale alla pressione del pedale ma mappata in una look up table, durante le partenze a massima accelerazione il valore pedale si attesta sullo 0,44.

Sottosistema Controllo Iniettori



Occorre inviare alla ECUG del segnali di durata e anticipo iniezione che siano almeno verosimili. Nello stesso tempo occorre contenere la complessità del modello perché si lavora in tempo reale. Si è convenuto di assumere una durata dell'iniezione direttamente proporzionale alla coppia indicata e inversamente proporzionale al rendimento termodinamico. Quest'ultimo è stato ipotizzato direttamente proporzionale all'apertura della farfalla nella forma

$\eta TH(\%) = 0{,}31\,Throttle + 2$ che da un valore del 5% ad apertura minima e del 33% ad apertura massima. I tempi iniezione prodotti in questo modo vanno da circa 1 ms con farfalla chiusa fino a circa 6,5 ms con farfalla aperta e motore in coppia massima. In regime di minimo, con apertura farfalla al 13% e rendimento 5%, si ha un tempo di iniezione di circa 4 ms, sicuramente troppo alto. Queste semplificazioni sono indubbiamente pesanti ma la variazione dei tempi di iniezione rientra in un range verosimile. In futuro, si potrà apportare un facile miglioramento, sostituendo il calcolo del rendimento con una look up table che tenga conto di giri e pedale.

Il controllo dell'anticipo di iniezione, rispetto al punto morto inferiore, è stato fatto supponendo che la stessa termini esattamente al PMI. Il *Gain 1* converte i giri/minuto il deg/ms. Questi vengono moltiplicati per la durata dell'iniezione ottenendo l'anticipo in deg.

L'anticipo delle bobine varia linearmente con i giri da 3,3° a 1000 rpm fino a 14,85° a 4500 rpm.

Il blocco *Saturation* serve ad impedire che in rilascio il tempo di iniezione vada a 0. Questa protezione è necessaria in quanto la valutazione degli errori di iniezione è percentuale quindi in queste condizioni si registrerebbe invariabilmente un errore per ogni ciclo cilindro.

Controllo con hardware esterno

Il test per CF3000 può essere svolto automaticamente utilizzando dati registrati oppure in modo manuale. L'interfaccia standard di VeriStand consente di controllare i valori immessi cliccando sui singoli controlli e immettendo il numero con la tastiera; intuitivamente questo sistema rende quasi impossibile la simulazione della guida di un'auto quindi si è pensato di collegare il progetto ad un controller esterno (volante più pedaliera). Il primo passo è quello di collegare un joystick generico e leggerne i segnali in VeriStand. A quel punto basta collegare gli opportuni assi e pulsanti agli ingressi del modello veicolo per poterlo controllare. Prima di vedere come è stato realizzato il sistema di controllo occorre illustrare cosa sono i custom device e come si usano in VeriStand

Custom devices

VeriStand è in grado di gestire nativamente un buon numero di apparecchiature ma ha alcune lacune a volte sorprendenti: per esempio non è in grado di utilizzare hardware HID (human interface device) USB. Questo hardware comprende joystick, joypad, tavolette grafiche ecc. Quando si ha a che fare con componenti non supportati da VeriStand una possibile via d'uscita è la realizzazione di un custom device: un driver che mette in comunicazione VeriStand e l'hardware che si intende usare. I custom device sono realizzati in LabView quindi, a condizione di apprendere almeno in parte l'utilizzo di un software molto più complesso, è possibile comunicare pressoché con qualunque cosa emetta segnali elettrici. I custom devices,come detto, vengono progettati in LabView ma per poter essere utilizzati vanno compilati. Il programma li salva in una directory comune che viene scansionata da VeriStand all'avvio. Tutti i custom device individuati saranno disponibili nel System Explorer cliccando con il tasto destro sulla voce Custom Devices. Il procedi-

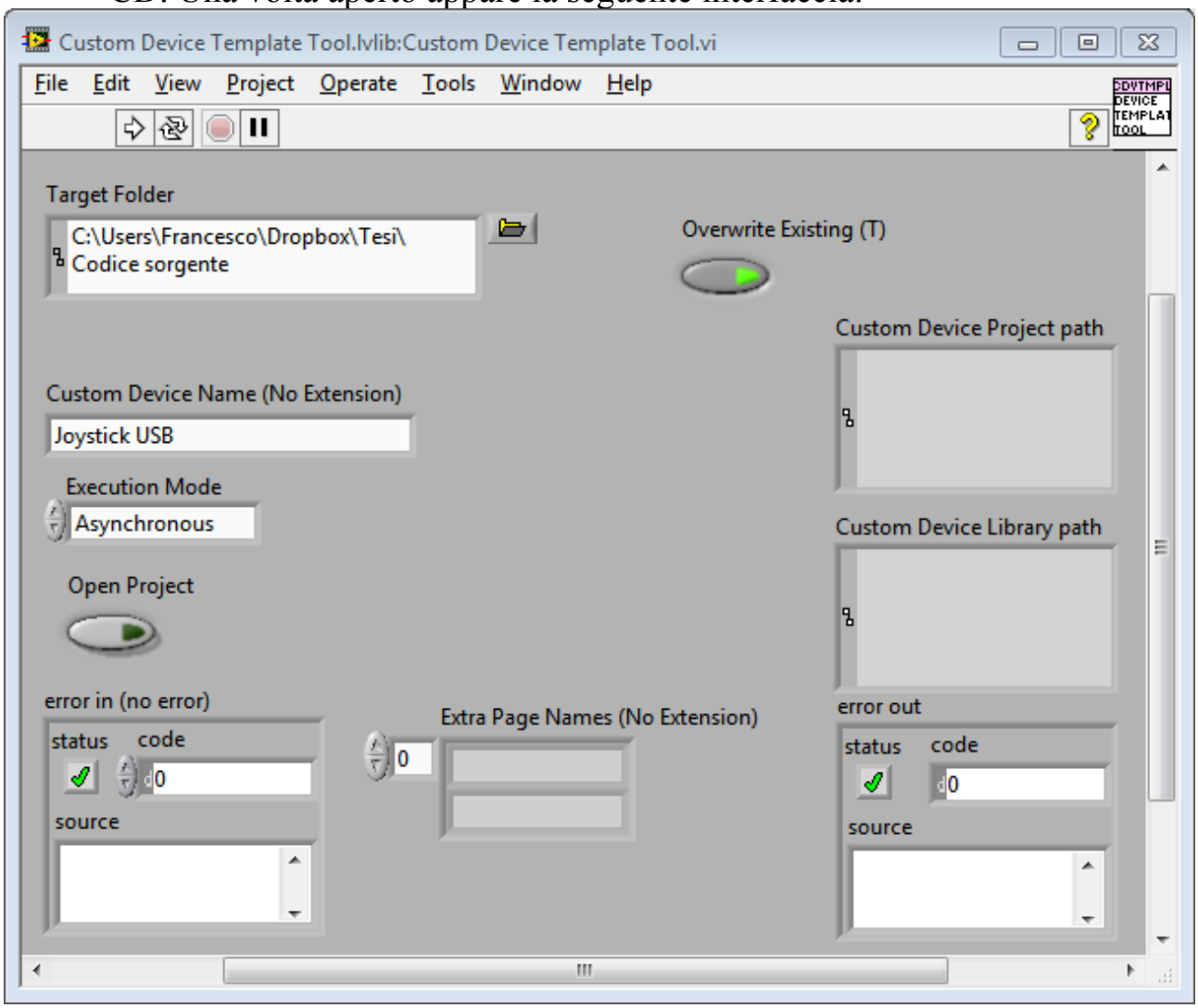
mento dettagliato per la creazione e l'uso di un custom device sono riportati alla voce *Joystick USB*.

Joystick USB

La diffusione ed economicità dei controller USB ha fatto propendere per questo tipo di periferiche. Lo sviluppo si è svolto utilizzando un joystick Saitek Cyborg Evo già disponibile. Anzitutto si sono cercate indicazioni nella documentazione e nei forum National Instruments per sapere come leggere i dati del joystick da VeriStand. Dal momento che non è stato trovato alcun riferimento al collegamento di hardware HID (human interface device) USB con VeriStand ci si è orientati per l'utilizzo di un custom device. Il sito della National Instruments mette a disposizione una sezione nella quale gli utenti distribuiscono i custom devices creati da loro in modo da risparmiare il lavoro ad altri. Benché non vi sia attualmente materiale relativo alle periferiche USB è presente un utilissimo programma che è l'Easy Custom Device Tool. Attraverso questo strumento è possibile realizzare CD concentrandosi sulla funzionalità e lasciando che lui si occupi della corretta configurazione del progetto.

Creazione del progetto LabView per un custom device

La creazione di CD richiede che l'utente lavori con uno schema e con impostazioni ben precise. Per semplificare questa fase NI rende disponibile un piccolo programma che si chiama *Custom Device Template Tool.vi*: questo è il punto di partenza per la creazione del CD. Una volta aperto appare la seguente interfaccia:



Le impostazioni mostrate sono quelle utilizzate per la creazione del joystick.

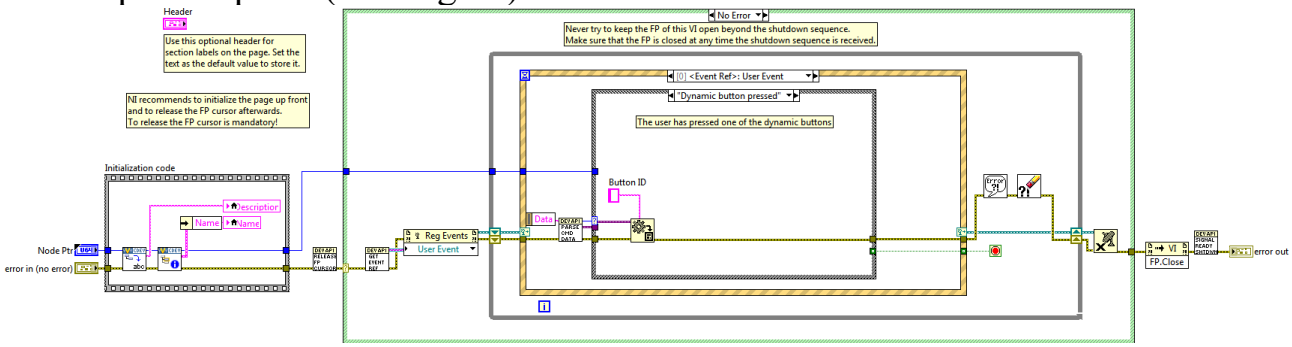
- Target Folder è la cartella da usare come base del codice sorgente

- Custom Device Name, oltre ad essere il nome del CD, crea una sottocartella con lo stesso nome sotto a Target Folder.
- Overwrite Existing serve se si vuole eliminare il contenuto precedente, è acceso di default
- Open project apre il progetto immediatamente dopo l'esecuzione del programma. E' acceso di default ma è stato spento perché ancora il progetto non va aperto.
- Extra Page Names serve ad aggiungere pagine nel progetto. Si rende necessario quando la complessità del CD suggerisce di spezzare in più parti il programma. E' una opzione avanzata che grazie all'uso dell' Easy Custom Device Tool può essere trascurata.

Una volta che tutto è pronto si può avviare il programma che crea i file necessari nella cartella indicata.

Modifica del progetto con Easy Custom Device Tool

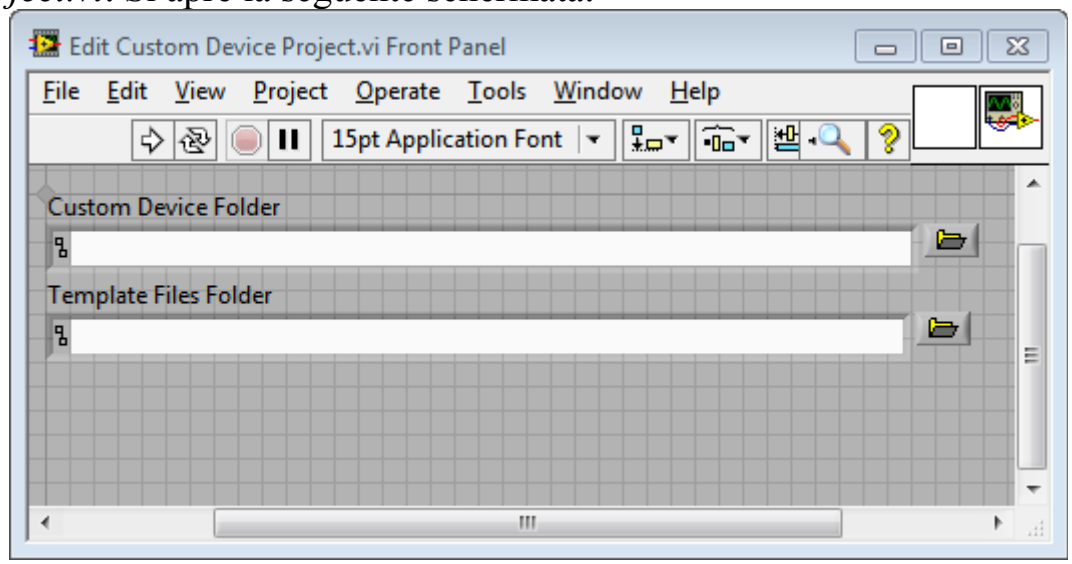
Il progetto realizzato dal passaggio precedente è uno scheletro quasi vuoto, utilizzabile solo da un utente avanzato: non ci sono riferimenti per i canali in ingresso e in uscita, non è semplice capire come connettere tra loro le varie parti e cosa mettere negli schemi precompilati (vedi figura).



Un ulteriore ostacolo all'utente inesperto è rappresentato dalla necessità di compilare manualmente il file xml che contiene le istruzioni necessarie a VeriStand per trovare i canali ecc.

L'Easy Custom Device Tool semplifica lo sviluppo mettendo a di-

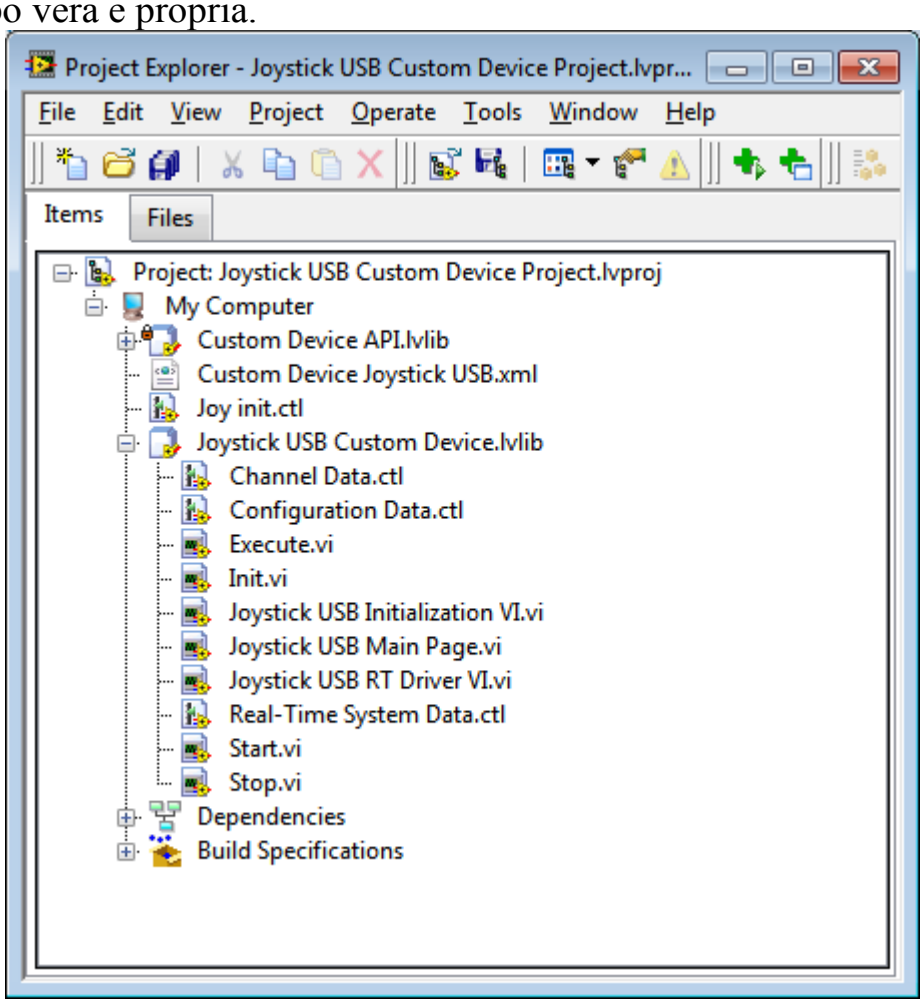
sposizione schemi estremamente semplici e occupandosi di inserirli correttamente nella struttura del CD. Va scaricato dal sito National Instruments e avviato cliccando il file *Edit Custom Device Project.vi*. Si apre la seguente schermata:



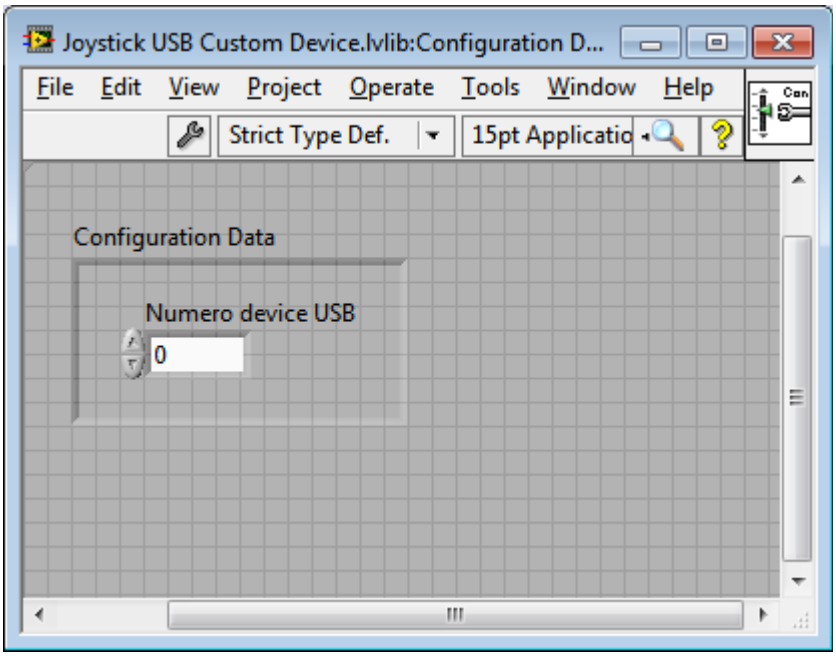
- Custom Device Folder è la cartella, creata precedentemente, che contiene il progetto
- Template Files Folder è una sottocartella dell'Easy Custom Device Tool che contiene i file standard da aggiungere al progetto

Una volta compilato e avviato, il programma può essere chiuso.

Nella figura seguente si può notare che sono state apportate alcune modifiche al progetto e sono stati aggiunti alcuni file; solo questi vanno modificati, mentre quelli creati nel passaggio precedente non vanno più toccati. A questo punto ci si può dedicare alla fase di sviluppo vera e propria.

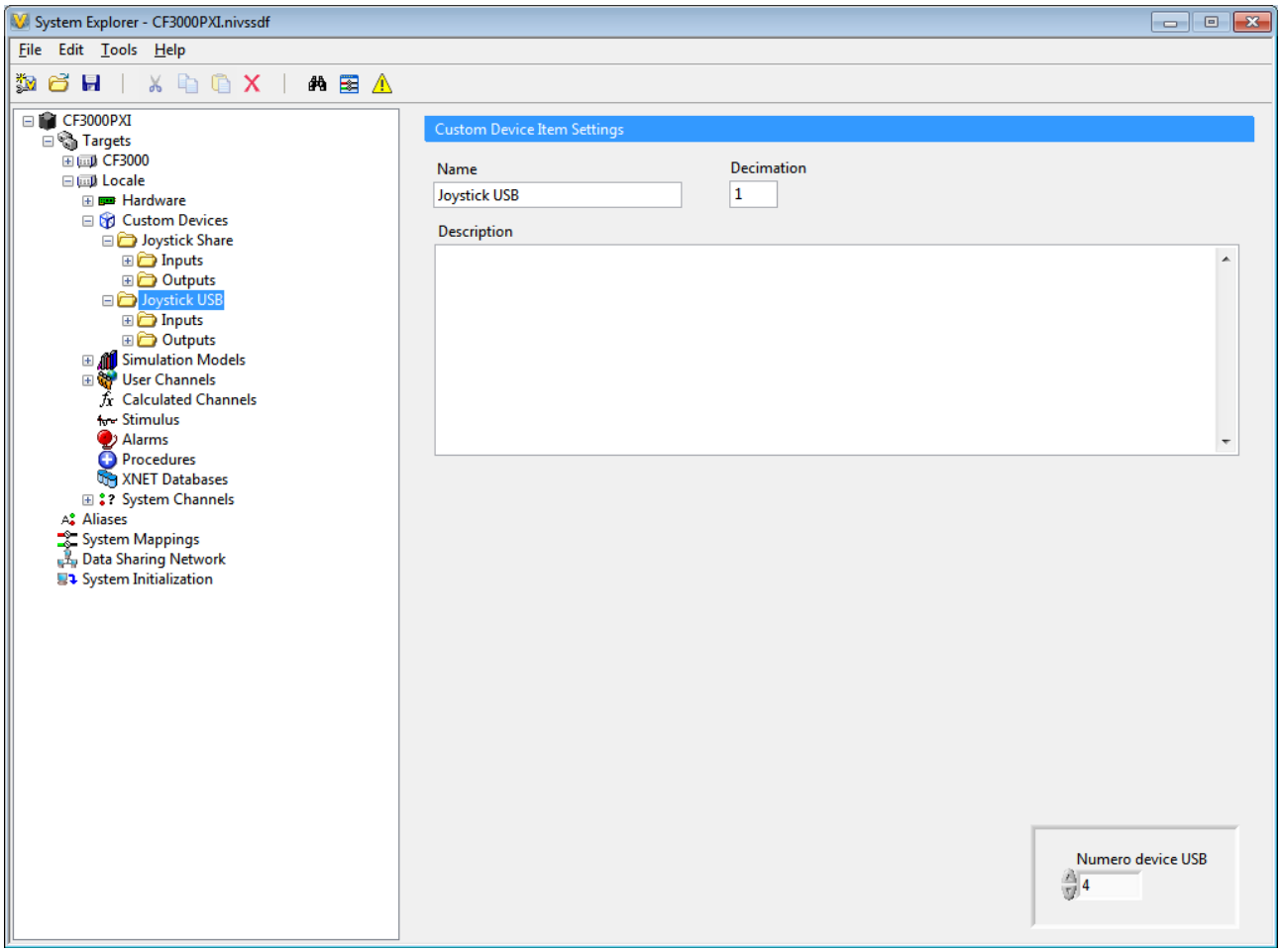


Configuration Data.ctl



Il primo file da modificare è questo. Si tratta di un cluster contenente tutti i controlli che dovranno comparire nella schermata principale del CD su VeriStand. Per il joystick è stato inserito un numeric control che indica il numero di device USB.

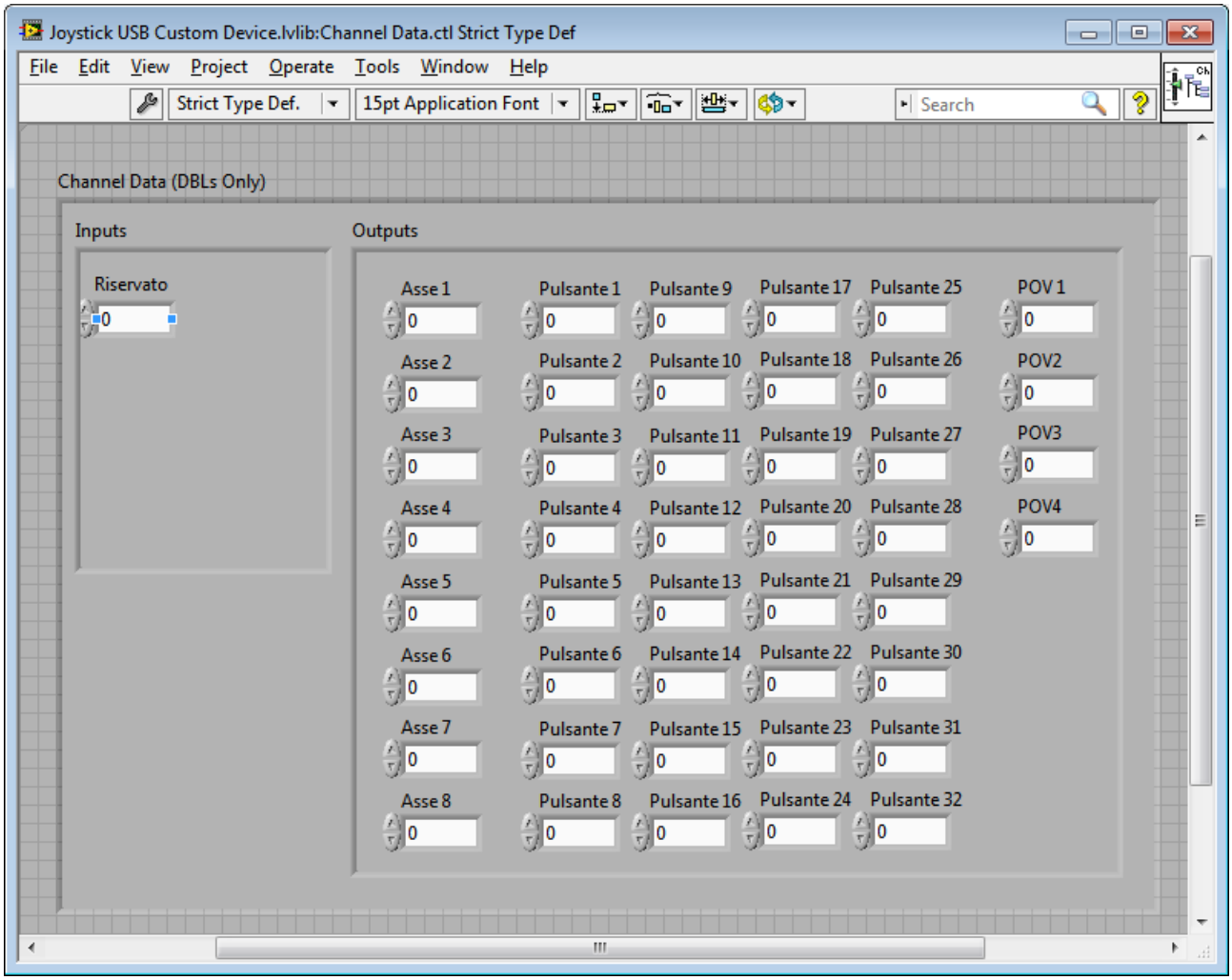
Il CD utilizza dei blocchi che si appoggiano a DirectX di Windows per leggere i dati del joystick. Windows numera i componenti USB collegati e di norma, se viene inserito un solo joystick, il valore corretto è 0. Nel caso del portatile sul quale è stato fatto lo sviluppo



il valore corretto è 4, in altri casi occorre effettuare prove. Per questa ragione è stato necessario inserire un controllo manuale modificabile da VeriStand. Ecco come appare la relativa schermata:

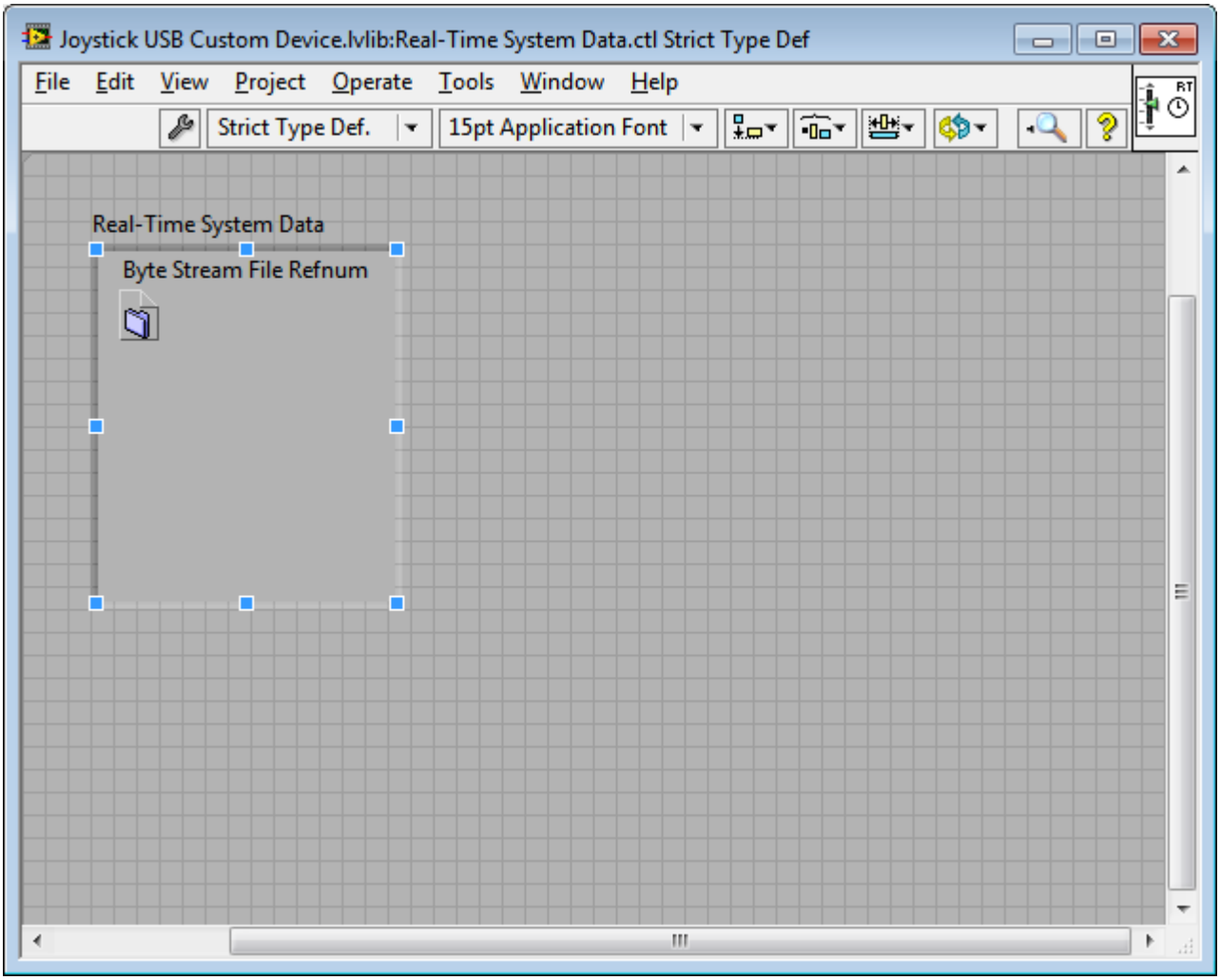
LabView richiede che non ci siano cluster vuoti quindi, anche se non è necessario inserire comandi in questa schermata, la cosa più semplice da fare è aggiungere un falso controllo, magari specificando nel titolo che è bene non utilizzarlo.

Channel Data.ctl



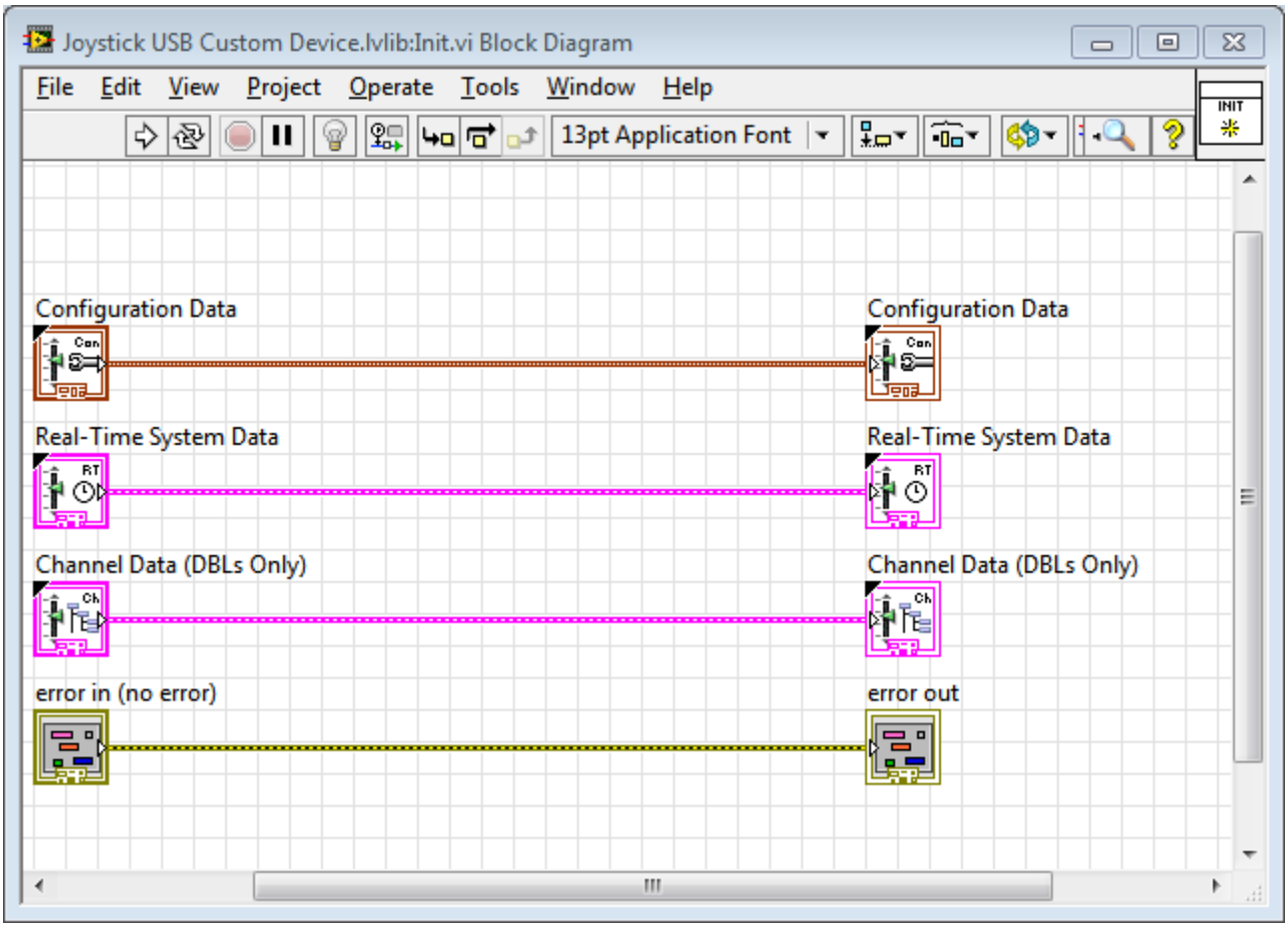
Si tratta di due cluster: uno contiene tutti gli input e uno tutti gli output che compariranno in VeriStand. Il joystick a disposizione è dotato di 4 assi, 12 pulsanti e 1 POV ma si è deciso di estendere i controlli non sapendo cosa sarà disponibile in futuro: infatti in questa fase aumentare il numero di uscite non comporta alcun problema. Notare il canale Riservato in ingresso: il joystick non ha bisogno di ingressi ma il cluster non può restare vuoto.

Real Time System Data.ctl



Questo cluster contiene tutti i riferimenti necessari all'esecuzione da parte dell'RT engine.

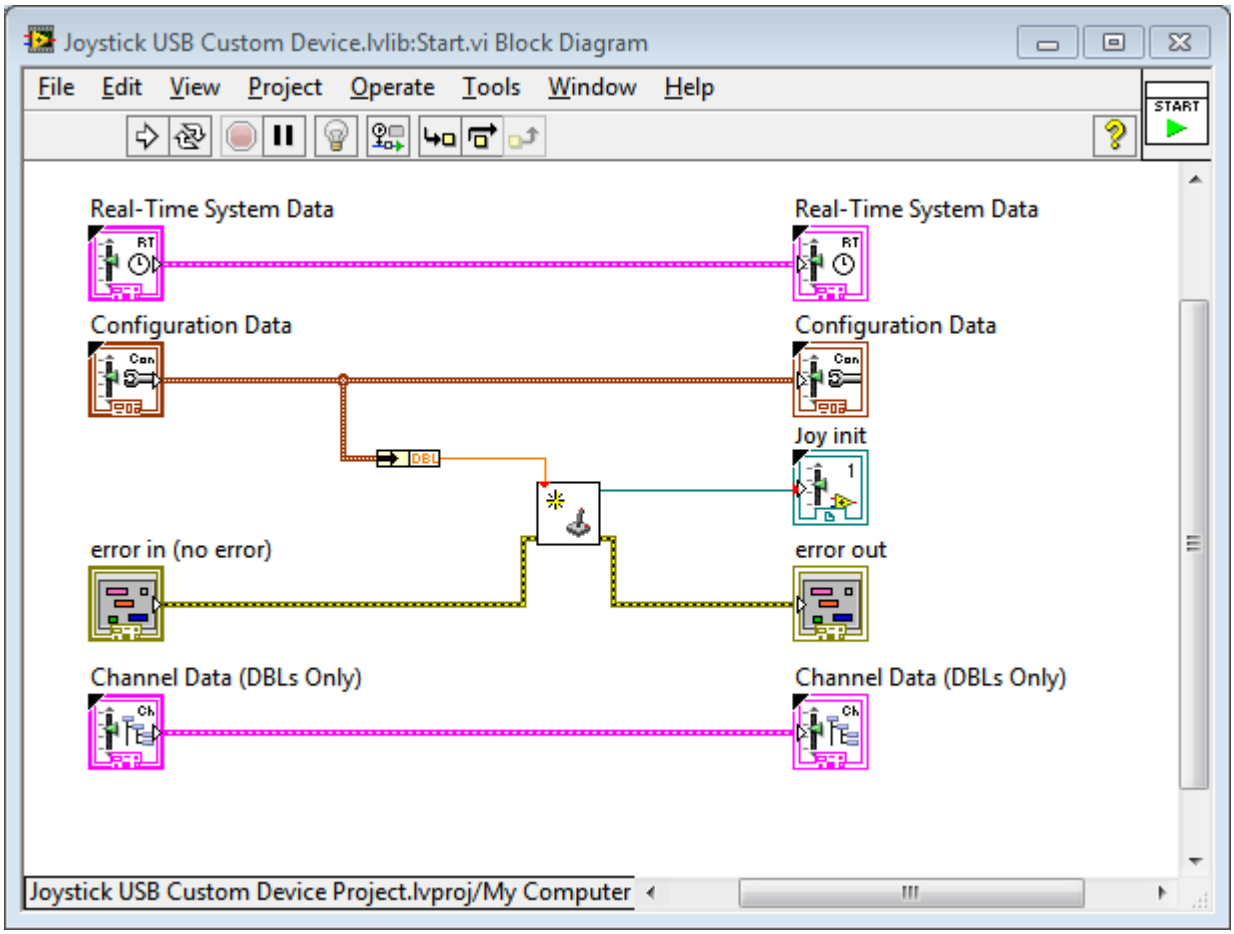
Init.vi



Questo .vi è il primo ad essere eseguito dall'RTE una volta avviato il progetto quindi dovrebbe essere usato per impostare i valori di partenza e la configurazione delle successive funzioni. In questo caso non è stato utilizzato ma gli ingressi e le uscite della struttura interna sono state collegate tra loro

Tutti i .vi successivi hanno questa struttura di base. Si noti quanto è più semplice di quella standard creata da LabView e mostrata in precedenza.

Start.vi

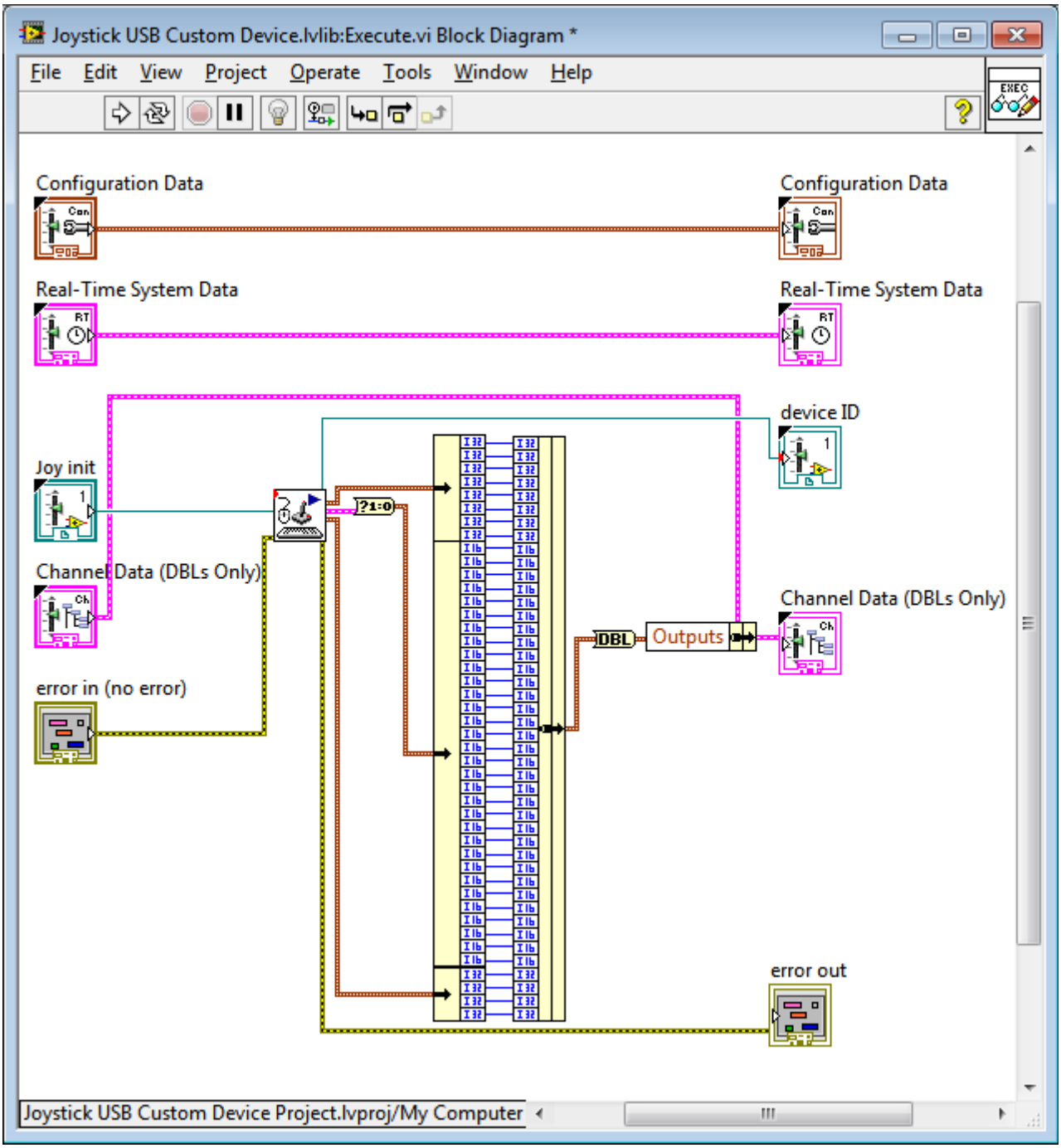


Dovrebbe essere usato per avviare tutte le funzioni da eseguire prima del loop principale.

Il blocchetto al centro si chiama *Initialize joystick.vi* ed è un blocco standard di LabView che informa i blocchi successivi delle caratteristiche del joystick collegato. In ingresso ha bisogno del numero di riferimento della periferica USB contenuto nel cluster Configuration data.ctl. La struttura contiene un blocco in uscita aggiuntivo chiamato joy init: vista la particolarità dei dati trasmessi da *Initialize joystick* non è stato possibile creare dei connettori funzionali all'interno dei cluster precedenti. Alla fine si è optato per la creazione di un ulteriore connettore usando il comando *create control*, una procedura semplice ma che, come si vedrà, richiede un piccolo intervento aggiuntivo. I blocchi error in e error out collegano tutte le

componenti del progetto e si occupano di trasmettere all'utente i messaggi relativi agli errori che possono verificarsi. Non sono mandatori ma, se ci sono problemi in parti non collegate, l'utente non può sapere cosa è successo.

Execute.vi

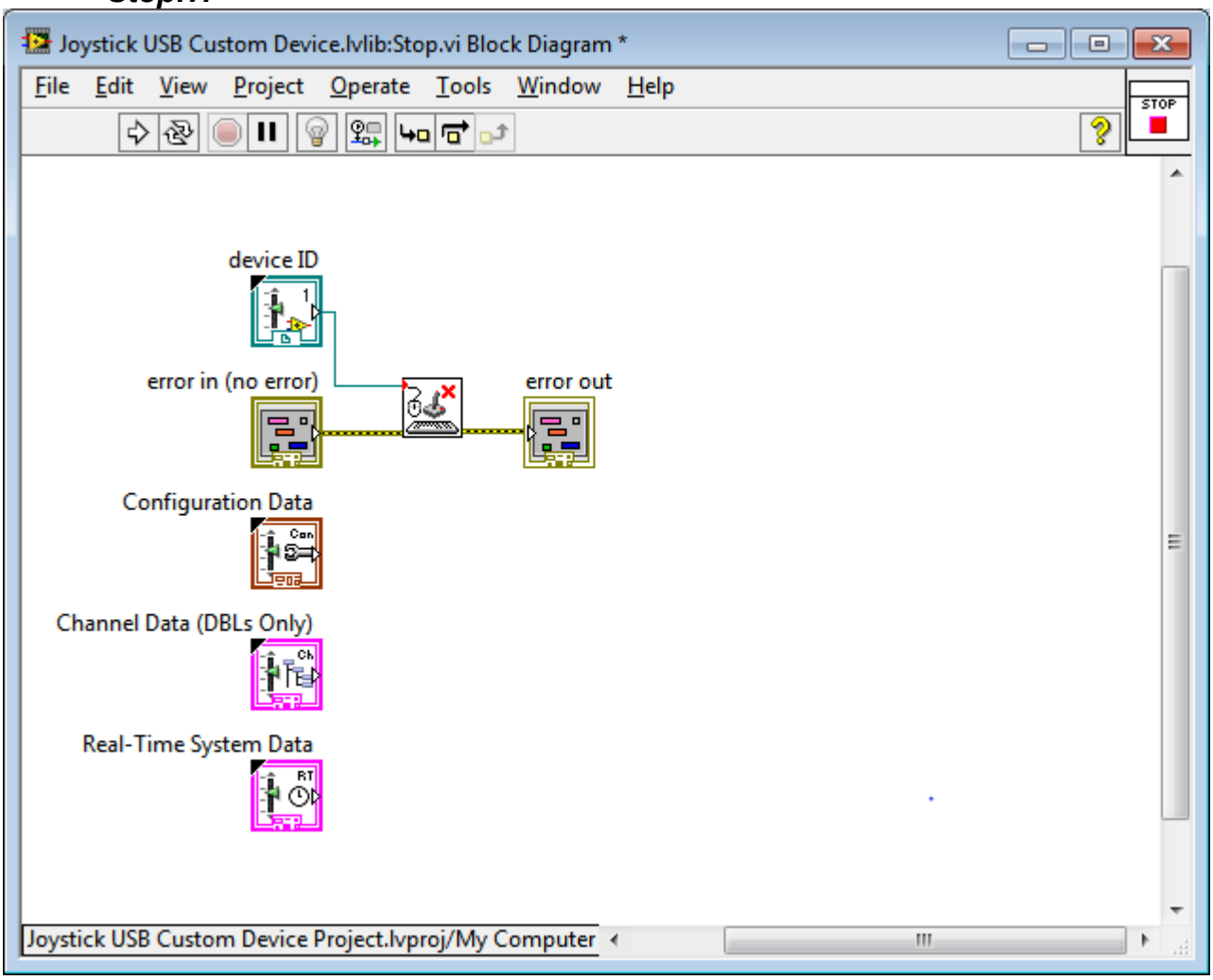


Questo .vi è contenuto nel loop principale del CT: in altre parole, mentre i .vi precedenti vengono eseguiti una volta sola all'avvio del progetto, questo viene eseguito a ripetizione fino a che non giunge il comando di stop. In questo .vi è stata inserita la struttura di ac-

quisizione dei segnali del joystick.

In ingresso c'è il blocco aggiuntivo Joy init e in uscita device ID che al di là del nome è esattamente lo stesso controllo. Il blocchetto con disegnate le periferiche si chiama Acquire Input Data.vi e fa parte dei blocchi standard di LabView. Il suo ruolo è di acquisire i segnali di pulsanti, assi e POV relativi a tastiere, mouse e joystick USB. Riceve le informazioni necessarie dal blocco Initialize joystick inserito nella voce precedente e trasmette i segnali separati in 3 array di cui due di numeri interi e uno di booleani. VeriStand, come segnali in ingresso e uscita, tratta solo numeri in virgola mobile e doppia precisione, inoltre occorre che tutti gli ingressi e le uscite siano preimpostati al momento dell'inserimento del CD nel progetto. Il blocco Acquire Input Data varia la dimensione degli array in base alle caratteristiche della periferica creando dei problemi in fase di utilizzo del CD quindi è stato necessario prevedere una periferica molto più estesa di quella disponibile, per consentire in futuro la massima flessibilità. Il pettine visibile al centro del .vi , assieme ai blocchetti che lo collegano, ha lo scopo di trasformare i booleani in numeri (1= true 0=false), scomporre i tre array, creare un unico array con 8 assi, 32 bottoni e 4 POV, convertire ogni numero in doppia precisione e sostituire tutti i segnali presenti nel cluster Channel data con quelli acquisiti. Naturalmente se la periferica non supporta più di un certo numero di segnali, tutti quelli successivi risultano 0.

Stop.vi



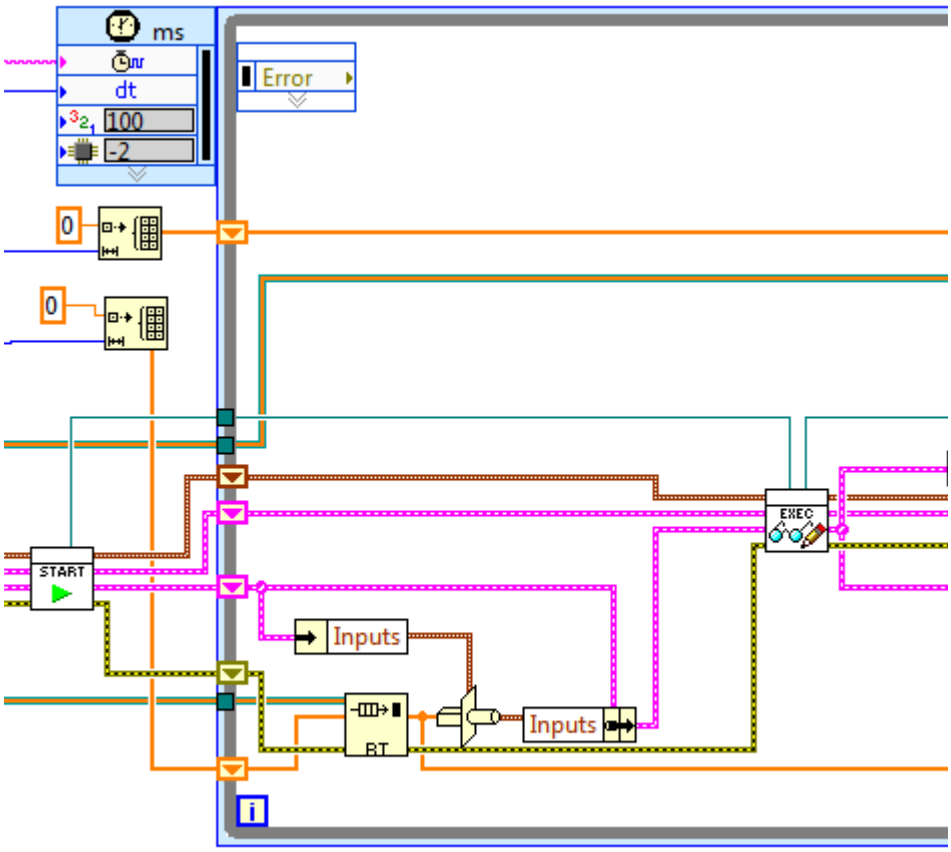
Quando si chiude il progetto VeriStand il loop principale viene interrotto e viene attivato questo .vi che ha lo scopo di terminare la varie funzioni e liberare le risorse.

Il blocchetto centrale si chiama *Close Input Device* e si occupa di liberare la risorsa joystick. Ci si aspetta che tutte le funzioni abbiano termine qui perciò non ci sono blocchi in uscita a parte error out.

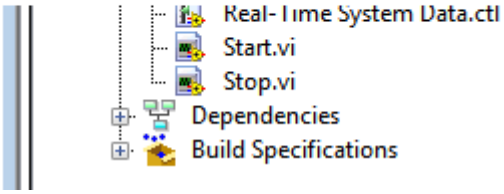
Ulteriori modifiche e compilazione

L'aggiunta di terminali extra nei .vi precedenti impone che questi siano collegati a mano, si tratta comunque di un'operazione molto semplice. Si apre *Joystick USB RT Driver VI.vi,* in figura è mostra-

ta solo una parte.

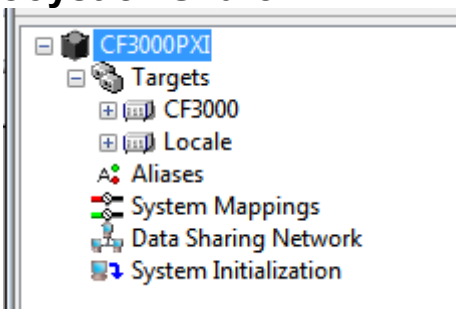


Il blocco *Start* e il blocco *Exec* sono i vi che abbiamo modificato in precedenza. Le linee rosa e gialle tratteggiate sono state create automaticamente, la linea verde acqua che collega i blocchi da sopra è stata aggiunta a mano. Dal blocco *Exec* parte una ulteriore linea verde acqua che si collega al blocco *Stop*, fuori dal loop principale (il rettangolo grigio). Usando l'Easy Custom Device Tool non è necessario analizzare a fondo la struttura di questo vi perché a parte il piccolo intervento descritto tutta la configurazione è automatica. L'ultimo passaggio è la compilazione: nell'albero del CD si clicca col destro sulla voce Build Specifications e si seleziona *Build all*.



Il computer compila il progetto e inserisce il custom device nell'elenco di quelli a disposizione di VeriStand.

Joystick Share



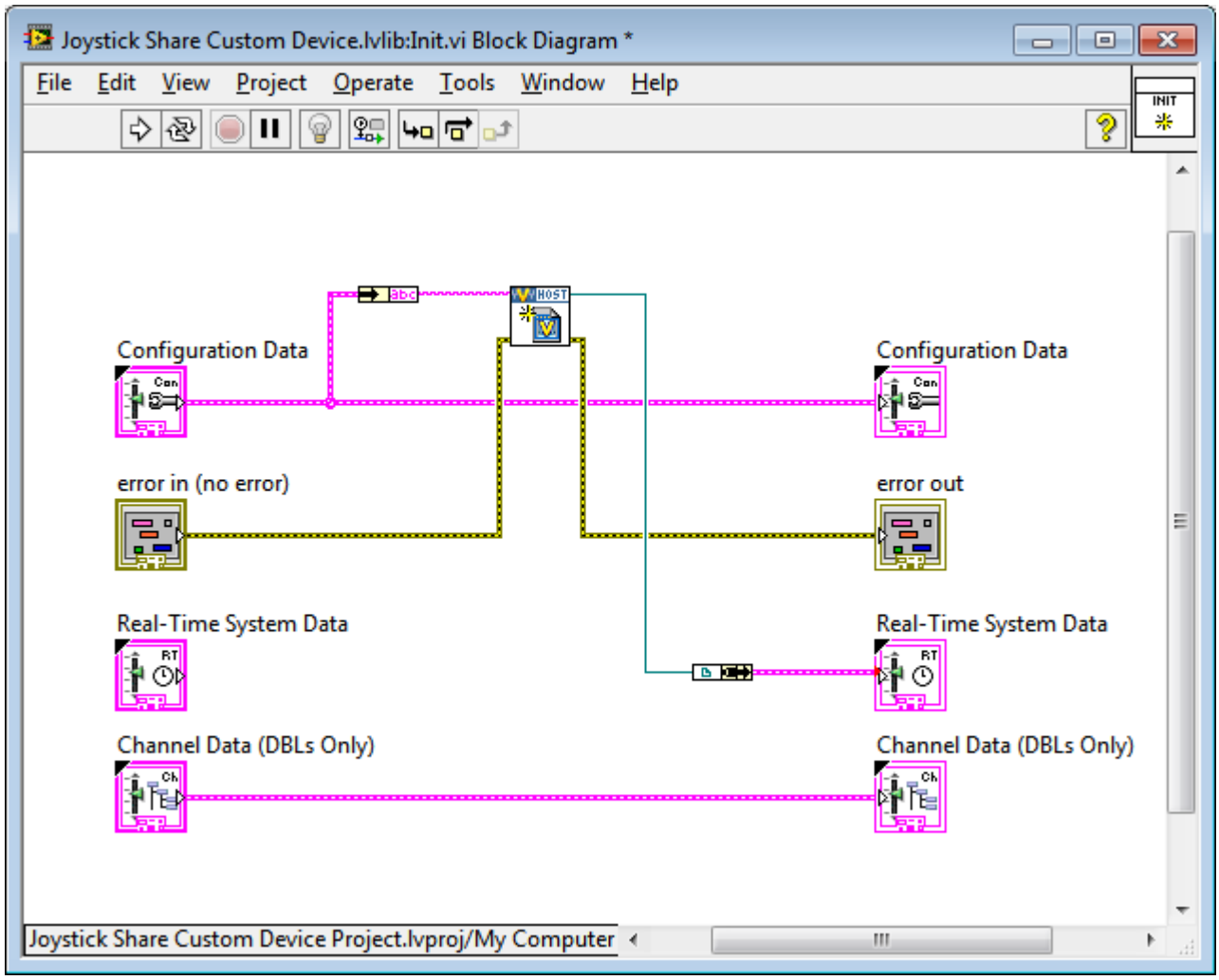
Guardando l'albero del System Explorer in VeriStand, sotto la voce Targets abbiamo le voci CF3000 e Locale. In VeriStand i Target sono i calcolatori sui quali sono eseguiti i compiti che vengono affidati al progetto e ognuno di loro può essere un PXI, un CompactRio o il PC locale (solo uno). Quando si aggiunge un custom device questo non è comune a tutti bensì viene inserito all'interno di un target. Poichè i modelli matematici devono girare su PXI e il joystick sul PC dell'operatore, è stato necessario creare due target: CF3000 per i modelli e Locale per il joystick. VeriStand tratta i target come compartimenti stagni: i canali di CF3000 e quelli di Locale non possono essere collegati tra loro in modo diretto. Per creare una struttura di calcolo distribuito occorre cliccare su *Data Sharing Network* quindi su *Add Reflective Memory Network.* La RMN è una rete creata collegando alle singole macchine delle schede Reflective Memory. Queste comunicano tra loro con delle connessioni a bassa latenza e si assicurano che tutti gli apparati abbiano gli stessi dati e siano perfettamente sincronizzati. Il progetto attualmente è asincrono quindi una RMN sarebbe una spesa inutile, senza contare il fatto che lo sviluppo è stato fatto su PC portatile, quindi incompatibile con le schede Reflective Memory. Su suggerimento ricevuto nel forum NI, è stato sviluppato un ulteriore CD che si occupa di trasmettere i segnali del joystick attraverso la normale rete TCP/IP. Questo CD è molto specializzato perché le operazioni che deve svolgere richiedono l'indirizzo esatto dei canali e quindi è compati-

bile solo con questo specifico progetto di VeriStand. Sarebbe possibile integrare questa funzione nel joystick ma si è preferito evitarlo per mantenere il suddetto più generico possibile. Si è utilizzato ancora una volta l'Easy Custom Device Tool quindi verranno evidenziati solo i passaggi salienti.

Configuration Data.ctl, Channel data.ctl, Real-Time System data.ctl

Il primo contiene una stringa con l'indirizzo IP del gateway di VeriStand ovvero del componente che si occupa di caricare tutto il progetto delle rispettive macchine all'avvio. In condizioni standard, come in questo caso, basta lasciarlo vuoto o scrivere localhost. Il secondo ha solo un canale di servizio allo scopo di non lasciare vuoto il cluster infatti questo CD non ha ingressi ed uscite, gestisce tutto internamente. Probabilmente in futuro sarà possibile integrare queste funzioni nel custom device del joystick. Il terzo cluster contiene il riferimento del workspace manager, necessario per le funzioni successive.

Init.vi

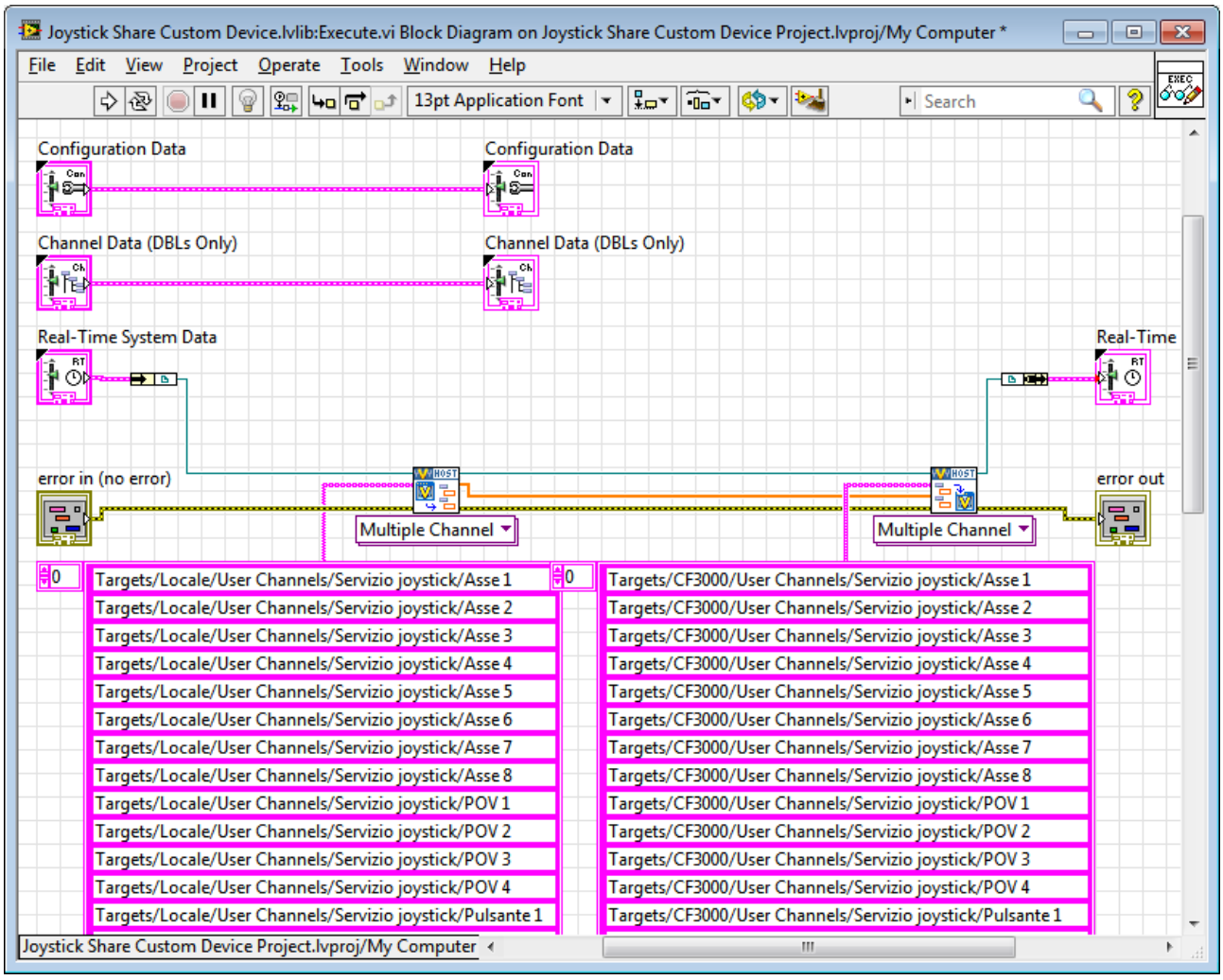


Il blocco centrale si chiama *Open Workspace Manager Reference VI* e consente ai successivi blocchi di intervenire sui canali di VeriStand mentre questo è in funzione: sostanzialmente assicura al programma successivo gli stessi poteri dell'utente. Il valore del blocco *Real-Time System Data* viene sostituito dal riferimento generato dall'*Open Workspace Manager Reference VI.*

Start.vi

È vuoto.

Execute.vi



Prima di analizzarne la struttura occorre dire che in entrambi i target di VeriStand sono stati creati degli user channel identici da collegare alle uscite del joystick, questi si trovano in una sottocartella di servizio per facilitare la lettura. Questo VI mette in comunicazione tra loro gli user channel per cui, se necessario, i canali non utilizzati dal joystick possono essere sfruttati per trasmettere altri segnali. E' importante ribadire che, anche se i due sistemi lavorano alla stessa frequenza, con questa struttura il sincronismo è pressoché impossibile.

Il primo blocco si occupa della lettura dei canali indicati (l'elenco visibile non è completo: in totale i canali sono 44). Il secondo scri-

ve nei nuovi indirizzi il valore. Questa configurazione lavora a senso unico, trasmette dal Locale al PXI, quindi è poco versatile ma adeguata per il joystick.

Stop.vi

E' vuoto

Ulteriori considerazioni

Questo CD, al contrario del precedente, può essere caricato sia su CF3000 che su Locale; conviene però metterlo sul secondo perché lavori in sincronia con il joystick e per alleggerire il più possibile il carico di lavoro del PXI che con l'attuale progetto risulta piuttosto gravoso.

Esecuzione dei test CF3000

Gli strumenti dedicati all'esecuzione dei progetti VeriStand sono il workspace e lo Stimulus Profile Editor. Il primo consente l'intervento manuale e il monitoraggio, il secondo l'esecuzione automatica. A seconda delle esigenze è anche possibile un utilizzo ibrido infatti il workspace resta a completa disposizione dell'operatore anche durante l'esecuzione dei profili di stimolo.

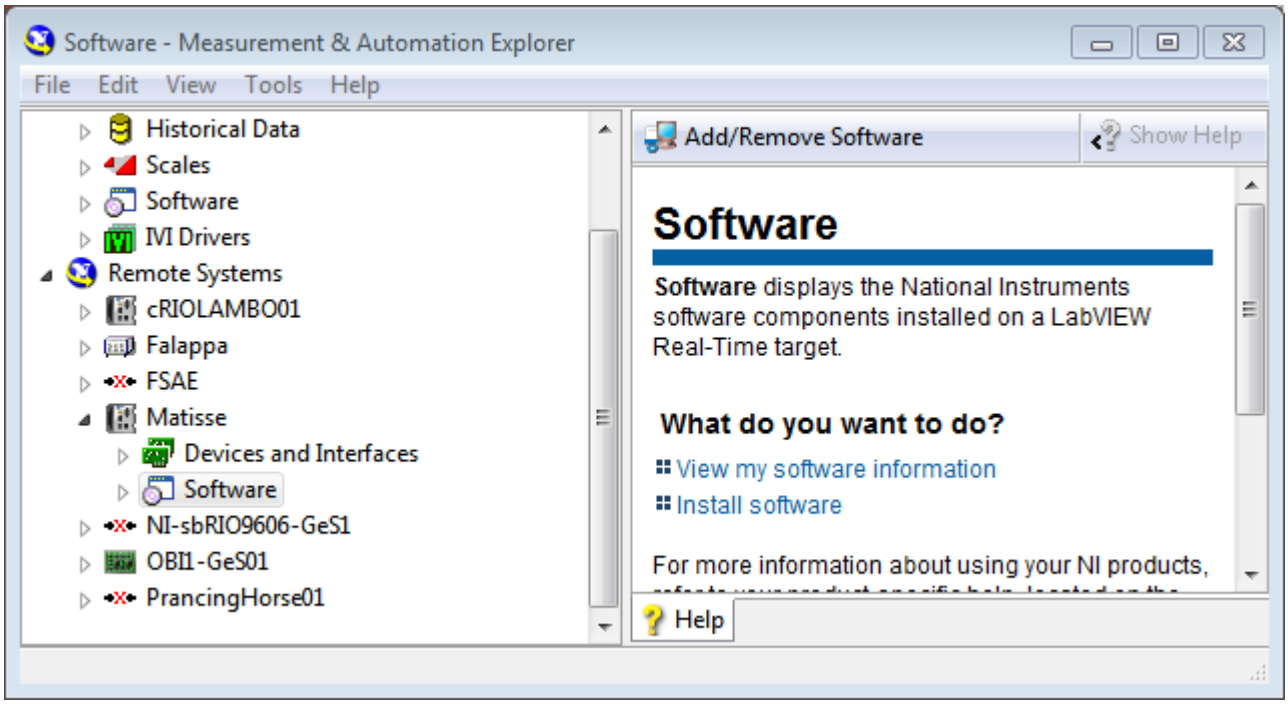
Lo Stimulus Profile Editor è, dalla versione 2011 di VeriStand, uno strumento particolarmente versatile ed esteso; per non appesantire eccessivamente questa sezione se ne suppone la conoscenza. Un'analisi approfondita di questo strumento è riportata in appendice.

Preparazione del PXI

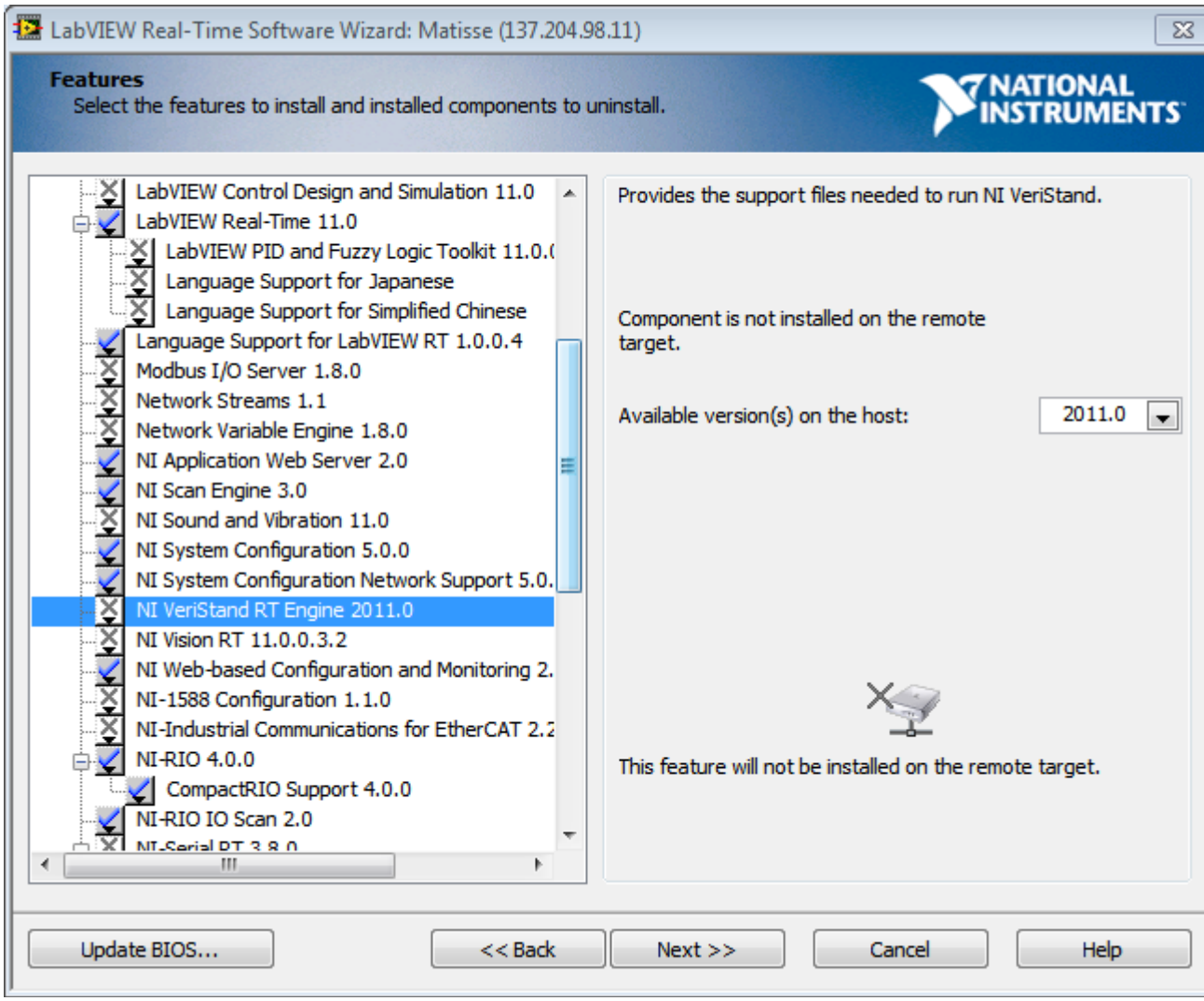
Prima di poter eseguire i test occorre configurare il PXI per il funzionamento con VeriStand. Questa fase è necessaria se non si è mai utilizzato VS sup PXI in questione e, purtroppo, nel caso che sia stato utilizzato LabView. L'utilizzo di LabView su PXI arresta il *VeriStand RT Engine* e, sebbene non lo disinstalli, impone l'avvio del *LabView RT Engine* alla partenza del sistema. L'unica soluzione finora trovata è la reinstallazione di VS. Per installare il Vs engine si segua la scaletta riportata

- Accendere il PXI e collegarlo alla rete.
- Avviare *Measurement & Automation Explorer:* nel diagramma ad albero cliccare la voce *Remote Systems* → *Nome del PXI* → *Software.* E' possibile che il sistema desiderato non sia visibile nell'elenco. In questo caso provare ad aggiornare la pagina premendo F5, se neanche questa procedura funziona si può aggiungere il PXI manualmente: click destro su *Remote System*, click su

Create New, click su *Remote Device,* click su *Next,* inserire l'indirizzo IP, click su *finish.*



- Cliccare su *Add/Remove Software*. Si aprirà una schermata simile alla seguente (dipende dal software disponibile sul PC host)

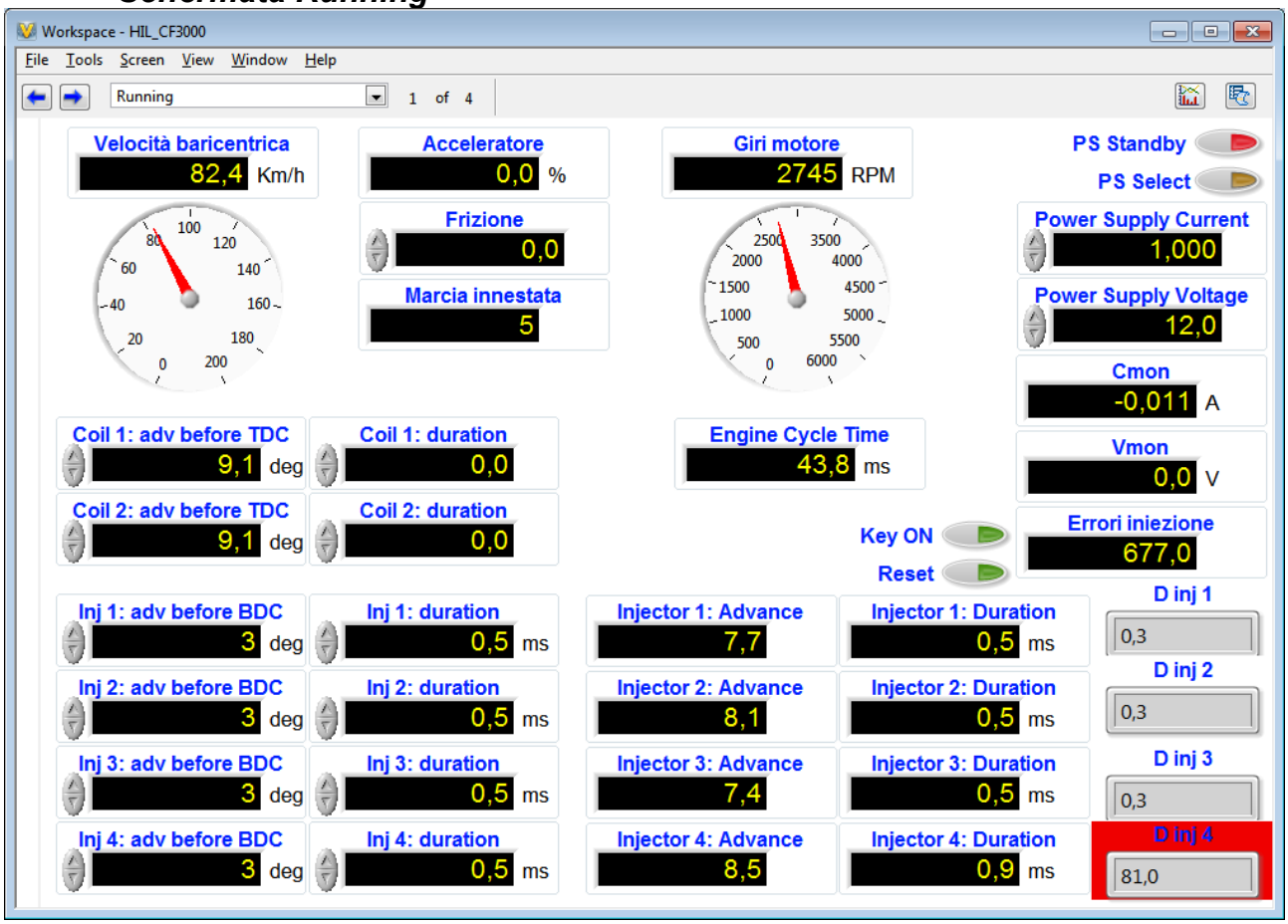


- Doppio click su *NI VeriStand RT Engine 2011.0* poi su *Install the feature,* infine *Next.* A questo punto verranno installati sia l'engine che le sue dipendenze. Se si verificano problemi con le dipendenze è sufficiente disinstallare tutto e riavviare. ATTENZIONE: potrebbero essere installati contemporaneamente sia *VeriStand RT Engine* che *LabVIEW Real-Time.* Non è un problema ma funzionerà soltanto l'ultimo engine installato.
- Riavviare il PXI, il VeriStand engine impiega circa 2 minuti ad avviarsi.

Test versione automatica

Il workspace è preconfigurato in funzione delle necessità di sviluppo del test. Poichè tutti i controlli sono collegati a user channels, se si vuole cambiare la configurazione di partenza è consigliabile intervenire sui valori di default di questi ultimi piuttosto che sui comandi del workspace. In questo modo la nuova configurazione verrà conservata ad ogni riavvio del test. E' possibile aprire il workspace sia a progetto avviato che fermo, nel secondo caso tutti i valori risultano azzerati. Ciò non significa che non siano configurati ma solo che il Workspace non può accedere ai valori corretti, si può comunque configurare lo spazio di lavoro e collegare controlli e sensori ai rispettivi canali.

Schermata Running



Questa è la schermata che compare appena aperto il workspace.

Cliccando sulle frecce subito sotto al menù è possibile passare alle schermate *Setup* e *Sensors*. Tutti i controlli sono riconoscibili dalla doppia freccia alla loro sinistra, i sensori ne sono invece privi.

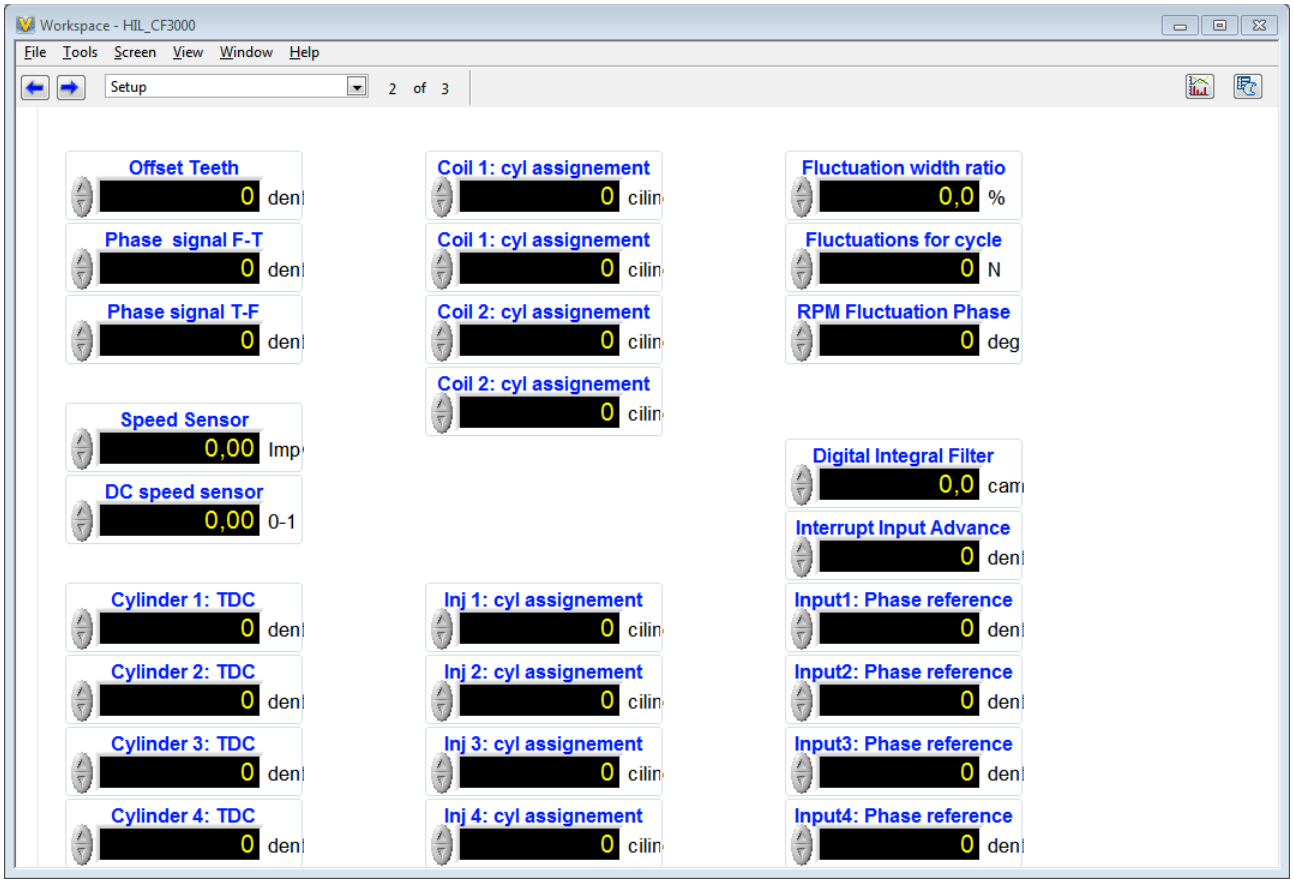
La parte superiore e sinistra della schermata Running contiene i segnali di attuazione che sono output della ECU di serie e input per la ECUG di CF3000:

- *Vehicle* Speed: è la velocità, espressa in km/h, sulla base della quale viene generato il segnale di velocità
- *Engine Speed*: è la velocità media di rotazione del motore espressa in giri/minuto
- *Coil #: adv* Before *TDC*: rappresenta l'anticipo (espresso in gradi) del comando di bobina rispetto al punto morto superiore di ognuno dei quattro cilindri (due per ogni bobina, impostati da "Setup")
- *Coil #:* duration: è la durata, espressa in ms, del comando di accensione relativo a ciascuna delle due bobine. Nella figura il valore non è stato impostato.
- Throttle: è l'apertura farfalla espressa in percentuale
- *Injector #: adv* Before *BDC*: rappresenta l'anticipo (espresso in gradi) del comando di iniezione rispetto al punto morto inferiore di ognuno dei quattro cilindri (impostati da "Setup");
- *Inj #: duration*: è la durata, espressa in ms, del comando di iniezione relativo a ciascuno dei quattro iniettori.

I comandi di iniezione hanno alla loro destra dei sensori che leggono i parametri corretti dalla ECUG. Si noti in particolare la voce *D inj 4* rappresentata con sfondo rosso: fa parte di un gruppo di sensori che indica la differenza percentuale tra la durata iniezione corretta a quella reale generata dalla centralina. In questo caso il segnale è stato distorto con un filtro passa-basso provocando un errore di lettura eccessivo e il relativo conteggio di iniezioni difettose.
Sopra questi sensori sono visibili i comandi dell'alimentatore remo-

to (attualmente non utilizzato) e il comando *Reset* che si occupa di azzerare il contatore errori.

Schermata Setup



Questa schermata consente di inserire tutte le variabili relative alla configurazione iniziale del sistema. Le variabili necessarie sono:

- *Offset* Teeth: il sistema genera un segnale di fonica 60-2 denti e assegna al dente successivo ai due denti mancanti il nome di dente zero (60). Questa variabile consente di attribuire un valore diverso al dente zero. (Per esempio assegnando a questa variabile il valore 3 il primo dente dopo i due denti mancanti sarà il dente 3 (63)). Si precisa che il dente viene sempre riconosciuto in corrispondenza del fronte di discesa del segnale digitale di ruota fonica
- *Phase signal F-T*: indica in corrispondenza di quale dente il

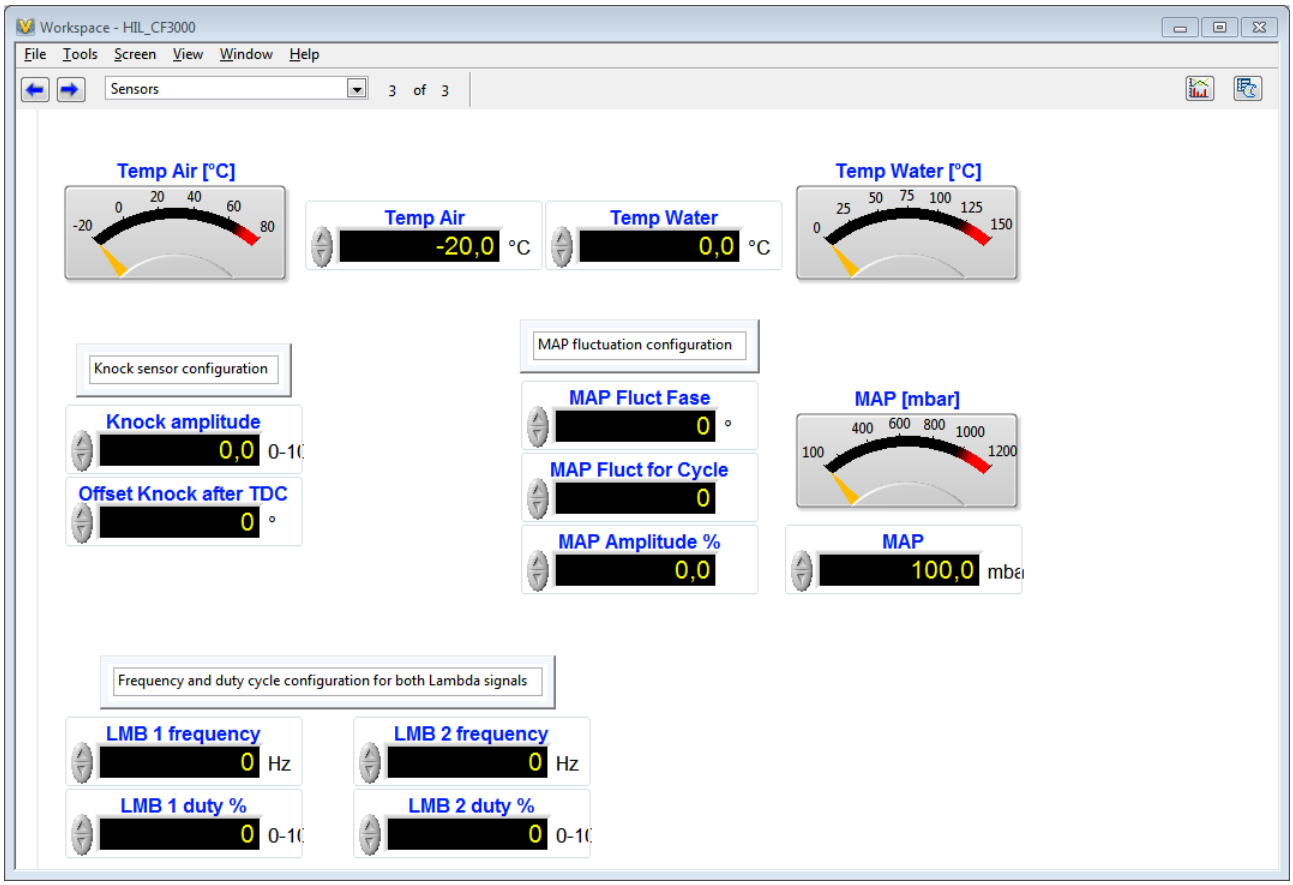
segnale digitale di fase (segnale di camma) dovrà essere riportato da basso (falso) a alto (vero);

- *Phase signal T-F*: indica in corrispondenza di quale dente il segnale digitale di fase (segnale di camma) dovrà essere riportato da alto (vero) a basso (falso);
- *Speed Sensor*: rappresenta la caratteristica del sensore di velocità veicolo riportata in impulsi per metro;
- *DC Speed Sensor*: è il duty-cycle con il quale verrà generato il segnale di velocità;
- *Cylinder #: TDC*: rappresenta la posizione angolare del punto morto superiore di fine compressione dei rispettivi cilindri espressa in denti;
- *Coil #: Cyl assignment*: in questi quattro controlli (due per ogni bobina) è necessario assegnare due cilindri ad ogni bobina. Il sistema calcolerà la posizione angolare del punto morto inferiore di ogni cilindro e, con i rispettivi anticipi, comanderà le bobine in funzione di queste assegnazioni;
- *Injector #: Cyl assignment*: rappresenta il collegamento tra le quattro attuazioni di iniezione ed i rispettivi cilindri. Attraverso questa maschera e l'assegnazione dei punti morti superiori, il sistema attuerà le iniezioni con i relativi anticipi;
- *Digital Integral Filter*: è il numero di campioni che devono essere raggiunti per il passaggio di stato dei segnali digitali in input, calcolato con la logica del filtro digitale integrale;
- *Fluctuation width ratio*: rappresenta l'ampiezza delle fluttuazioni di RPM rispetto al regime medio. (Per esempio inserendo 10 il sistema inserirà fluttuazioni di RPM pari al 10 % della velocità media di rotazione);
- *Fluctuations for cycle*: rappresenta il numero intero di oscillazioni degli RPM per ogni ciclo motore (in termini di perio-

di di una funzione seno);

- *RPM Fluctuation Phase*: è il ritardo, espresso in gradi, della fluttuazione richiesta rispetto alla funzione seno. Deve essere positiva;
- *Input #: Phase reference*: è il dente rispetto al quale verrà calcolato l'anticipo del segnale di iniezione relativo. Se in questo campo viene impostato il valore 60, l'anticipo letto nel pannello *Running* per questa attuazione sarà zero se il segnale passa basso in corrispondenza del dente 60;
- *Interrupt Input Advance:* è il dente in corrispondenza del quale vengono calcolati gli anticipi dei comandi di iniezione letti in input. Si consiglia di impostare questo valore sufficientemente lontano dalle attuazioni, in particolare di non impostarlo compreso tra l'attuazione e il suo riferimento di fase ("Input Phase Reference").

Schermata Sensors



Questo pannello agisce sulla sensoristica del motore:

- *Temp Air*: permette di impostare la temperatura ambiente da -20 a +80°C;
- *Temp water*: permette di regolare la temperatura del liquido refrigerante del motore da 0 a 150°C;
- *Knock amplitude*: è l'ampiezza del segnale di detonazione e va da 0 a 10 G;
- *Offset Knock after TDC*: è il ritardo, espresso in gradi, dell'inizio del fenomeno della detonazione;
- *MAP Fluct for Cycle*: è il numero di fluttuazioni di pressione per ciclo;
- *MAP Fluct Phase*: è la fase espressa in gradi della fluttuazio-

ne di pressione;

- *MAP Amplitude %*: è l'ampiezza della fluttuazione di pressione;
- *MAP*: Manifold Air Pressure espressa in millibar;
- *LMB # frequency*: sono le frequenze in Hz di generazione del segnale delle sonde lambda;
- *LMB # duty %*: rappresenta il duty cycle del segnale delle sonde lambda e può assumere il valore da 0 a 100, cioè da magro a grasso, con 50 di stechiometrico.

Uso in manuale

Aprire il "Project Explorer"; selezionare "Run" e attendere l'avvio del progetto. Si aprirà automaticamente il Workspace, a questo punto procedere come segue:

1. accendere l'alimentatore ed impostare i valori di tensione e corrente necessari;
2. nella pagina di *Setup* verificare ed eventualmente correggere la configurazione della prova, ricordandosi di mantenere il valore di "Interrupt Input Advance" lontano dalle attuazioni;
3. nella schermata "Sensors" impostare i valori necessari per la sensoristica;
4. nella schermata "Running" impostare i valori di anticipo e durata per bobine ed iniettori;
5. impostare velocità, regime di motore e farfalla desiderati.

Attenzione!

Questo test non è pensato per un uso manuale: sul sistema di sviluppo, con alcune configurazioni, è stato possibile generare dei falsi positivi in corrispondenza di consistenti variazioni nel regime di giri o nei tempi di iniezione. Questi fenomeni non si sono verificati in modalità automatica.

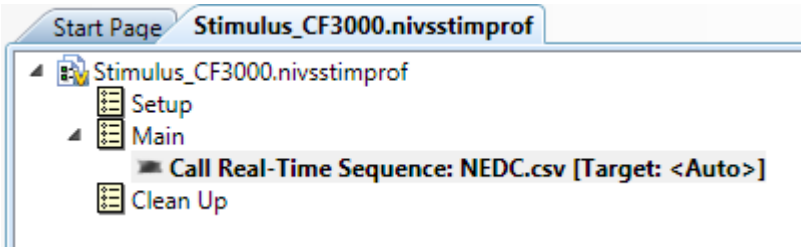
Uso in automatico

Per l'esecuzione della prova in automatico si intende la riproduzione di un file CSV, opportunamente configurato, per sovrascrivere i valori di alcuni controllori presenti nel workspace. Questi, in ordine di apparizione nel file csv, sono:

1. timestamp (ms);
2. Durata iniettore 1 (ms);
3. Durata iniettore 2 (ms);
4. Durata iniettore 3 (ms);
5. Durata iniettore 4 (ms);
6. Throttle (%);
7. Anticipo accensione Bobina 1 (gradi);
8. Anticipo accensione Bobina 2 (gradi);
9. MAP (mbar);
10. Engine speed (rpm);
11. Temperatura acqua (°C);
12. Vehicle speed (km/h);
13. Tempo.

Dal *Project Explorer* di VeriStand, sotto la voce *Profiles*, selezionare ed aprire *Stimulus_CF3000.nivsstimprof*.

Per caricare un file selezionare in Main *Call Real-Time sequence: file_name.csv.*

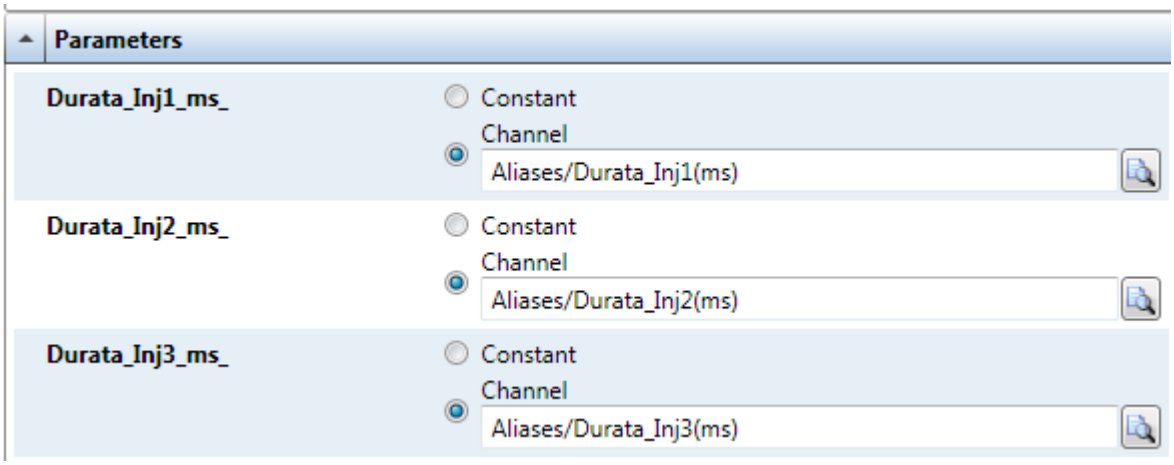


Nella categoria Common, alla voce "file path", selezionare la prova

da riprodurre. Questa dovrà essere:

- in formato *.csv con la virgola come separatore di elenco;
- con la prima colonna contenente la sequenza temporale scandita in millisecondi;
- con la prima riga contenente i nomi delle colonne sottostanti; questi diverranno le label dei parametri;
- la colonna del tempo deve essere chiamata “timestamp”, mentre le altre possono essere scelte dall’utente.

Nel file csv si crea una seconda colonna con il valore del tempo per avere un riferimento temporale sulla schermata “Running” relativo all’avanzamento dell’esecuzione. Questa non viene inviata al PXI, serve solo all’utente. Si consiglia di non mettere mai il valore di RPM troppo vicino a zero, altrimenti si crea un errore di lettura nelle attuazioni che comprometterebbe la riproduzione.



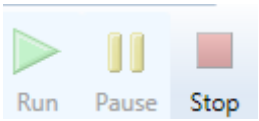
Premere “Update Parameters” che aggiorna i parametri esistenti. Alla voce “Parameters” collegare tutti i segnali nella lista ai rispettivi “User channels”. Raggiunta la configurazione desiderata salvare e compilare lo stimolo così creato, premendo su 

.

Prima di lanciare l’esecuzione della prova, aprire e mettere in esecuzione il workspace del progetto per procedere alla configurazione dei parametri non gestiti dalla riproduzione del csv (procedere

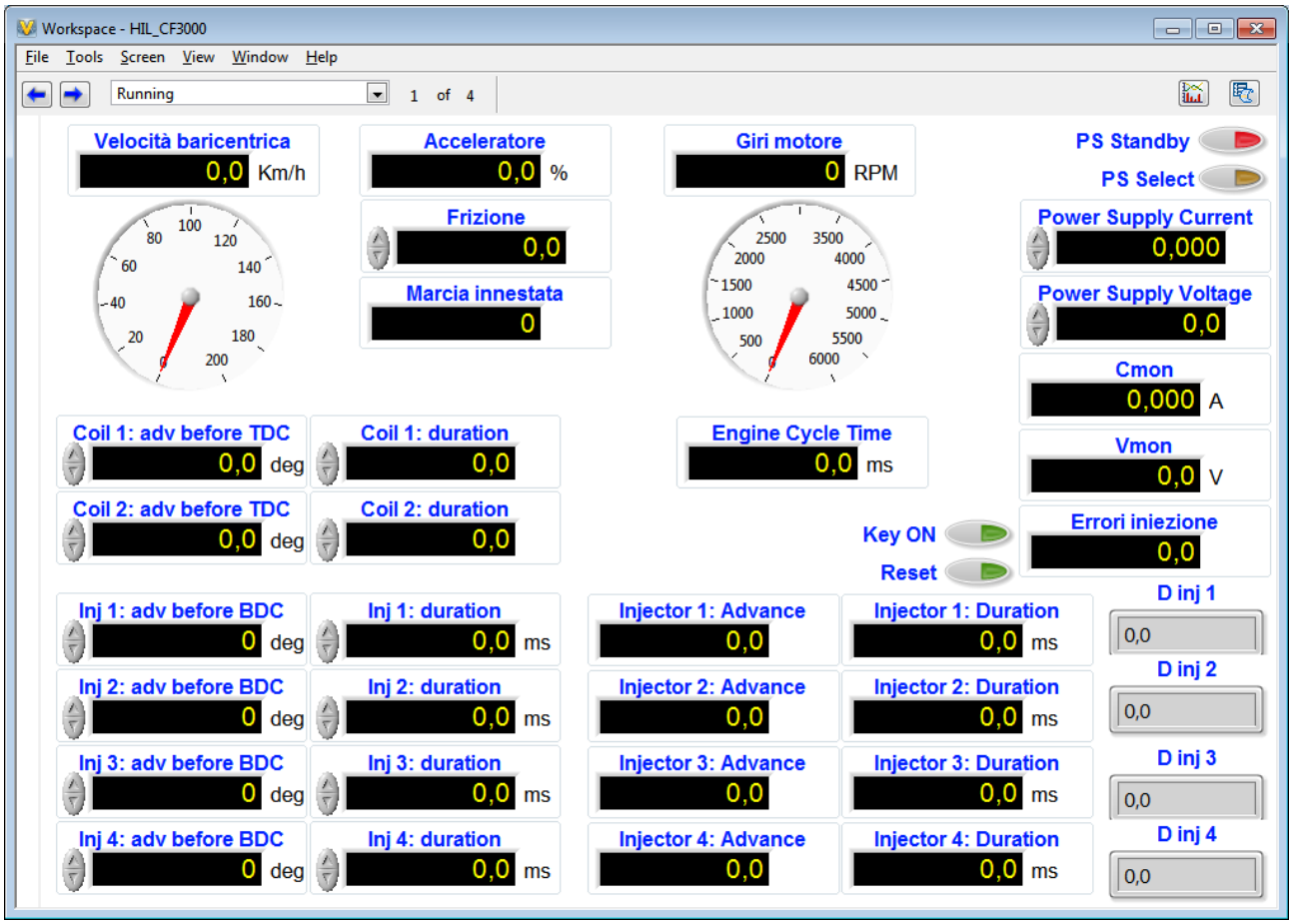
come per l'uso in manuale precedentemente descritto).

Tornando sullo "Stimulus Profile Editor", premere quindi " Run" per lanciare la prova. In ogni momento è possibile mettere in pausa la riproduzione, agire sul front panel manualmente e ripartire con la riproduzione dal punto in cui si era. Ovviamente premendo stop si interrompe la riproduzione e si dovrà ripartire con la riproduzione dall'inizio del ciclo.



Test versione manuale

Schermata Running



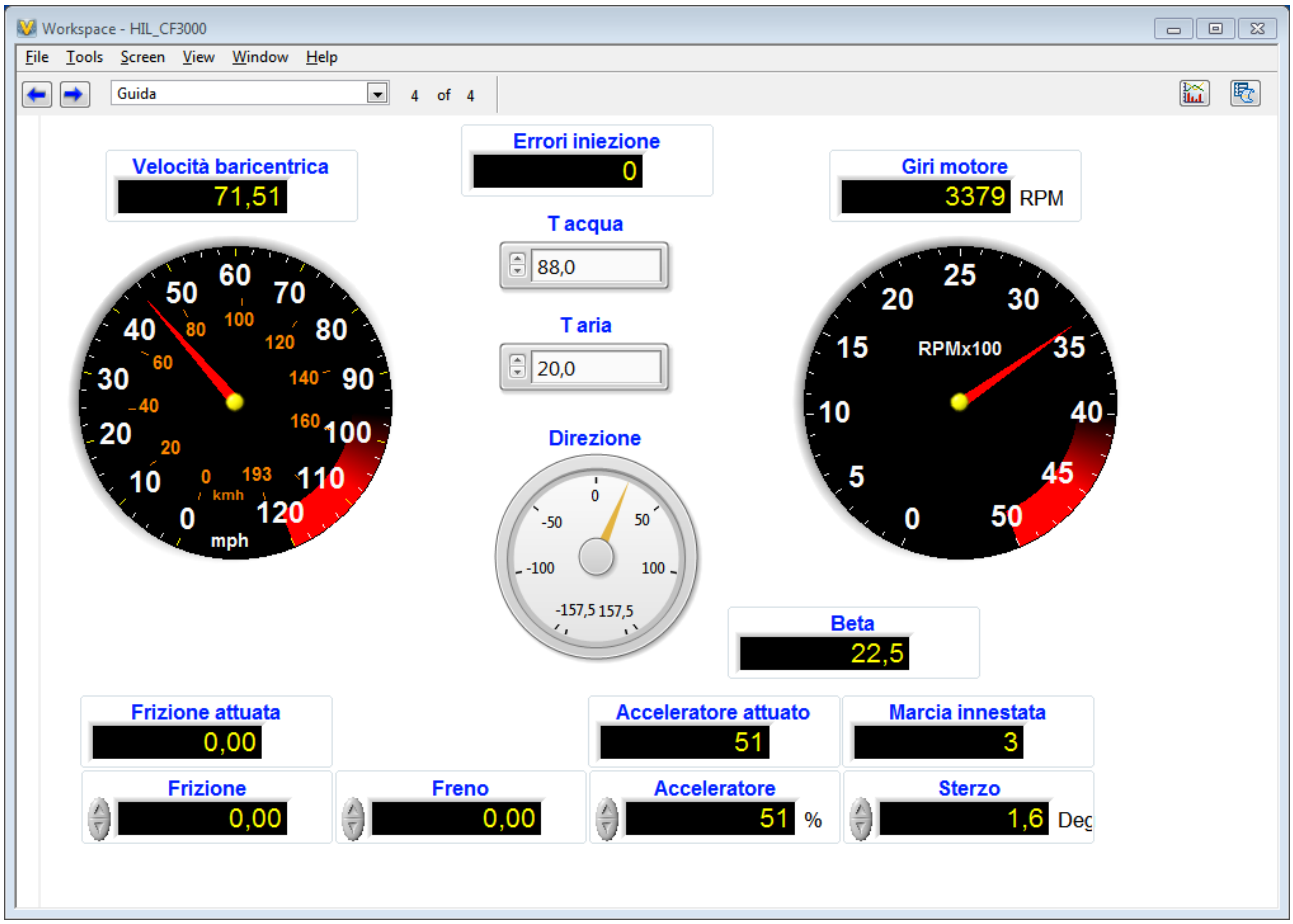
E' stato aggiunto il controllo *frizione* e l'indicatore M*arcia Innestata*. I controlli degli iniettori, delle bobine, della velocità e dei giri motore sono gestiti dal modello *Modello_S3_Motore.mdl* quindi il

loro comportamento non è controllabile dal Workspace.

Schermata Setup e schermata Sensors

Sono identiche al caso precedente al quale si rimanda per la descrizione e l'utilizzo

Schermata Guida



Questa schermata è dedicata alla guida del veicolo, è possibile gestire manualmente i pedali, il volante e le temperature oppure si può usare un controller USB. L'indicatore *Direzione* serve ad evidenziare gli effetti dello sterzo.

Uso in manuale

Aprire il “Project Explorer”, aprire il *System Definition file* e verificare che la scalatura degli assi del joystick sia tarata sull'hardware disponibile. selezionare “Run” e attendere l'avvio del progetto. Si aprirà automaticamente il Workspace, a questo punto procedere come segue:

1. accendere l’alimentatore ed impostare i valori di tensione e corrente necessari;
2. dalla pagina di “Setup” configurare la prova, ricordandosi di mantenere il valore di “Interrupt Input Advance” lontano dalle attuazioni e di lasciare a 0 la fluttuazione degli RPM.
3. nella schermata “Sensors” impostare i valori necessari per la sensoristica;
4. andare nella schermata *Running* o *Guida* e pilotare il veicolo.

Attenzione: all'avvio del progetto, dopo che si è aperto il workspace, con questo test possono passare molti secondi prima che la gestione FPGA sia funzionale (nel sistema di sviluppo oltre 50), durante questo tempo i sensori *Inj #: Duration* mostrano il valore 0. Chiaramente gli errori eventualmente conteggiati durante questa fase non sono da considerare.

Uso in automatico

Anche questo test può essere guidato da un profilo di stimolo. Occorre in questo caso agire solo sui comandi dei pedali e del volante perché l'intervento su altri canali provoca conflitti.

APPENDICE

Specifiche CDIL 2N2222A

Foglio 1

Watertown, MA
Microsemi
Progress Powered by Technology

580 Pleasant St.
Watertown, MA 02172
PH: (617) 926-0404
FAX: (617) 924-1235

2N2222A

75 Volts
0.8 Amps

NPN
BIPOLAR
TRANSISTOR

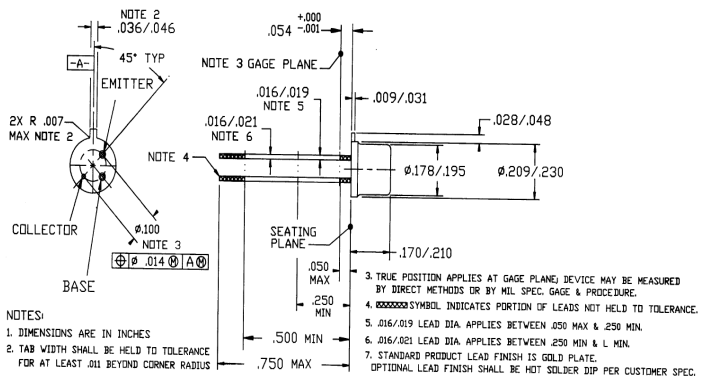
Features

- Meets MIL-S-19500/255
- Collector-Base Voltage 75
- Collector Current: 800mA
- Fast Switching 335 nS

Maximum Ratings

RATING	SYMBOL	MAX.	UNIT
Collector-Emitter Voltage	V_{CEO}		Vdc
Collector-Base Voltage	V_{CBO}	75	Vdc
Emitter-Base Voltage	V_{EBO}	6.0	Vdc
Collector Current	I_C	800	mAdc
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	0.5 2.85	Watt mW/°C
Total Device Dissipation @T_C = 25°C Derate above 25°C	P_D	1.8 10.3	Watt mW/°C
Thermal Resistance, Junction to Ambient	$R_{\theta JA}$	350	°C/W
Thermal Resistance, Junction to Case	$R_{\theta JC}$	97	°C/W
Operating Temperature Range	T_J	-65 to + 200	°C
Storage Temperature Range	T_{STG}	-65 to + 200	°C

Mechanical Outline



Datasheet# MSC0275A 5/19/1997

Foglio 2

2N2222A

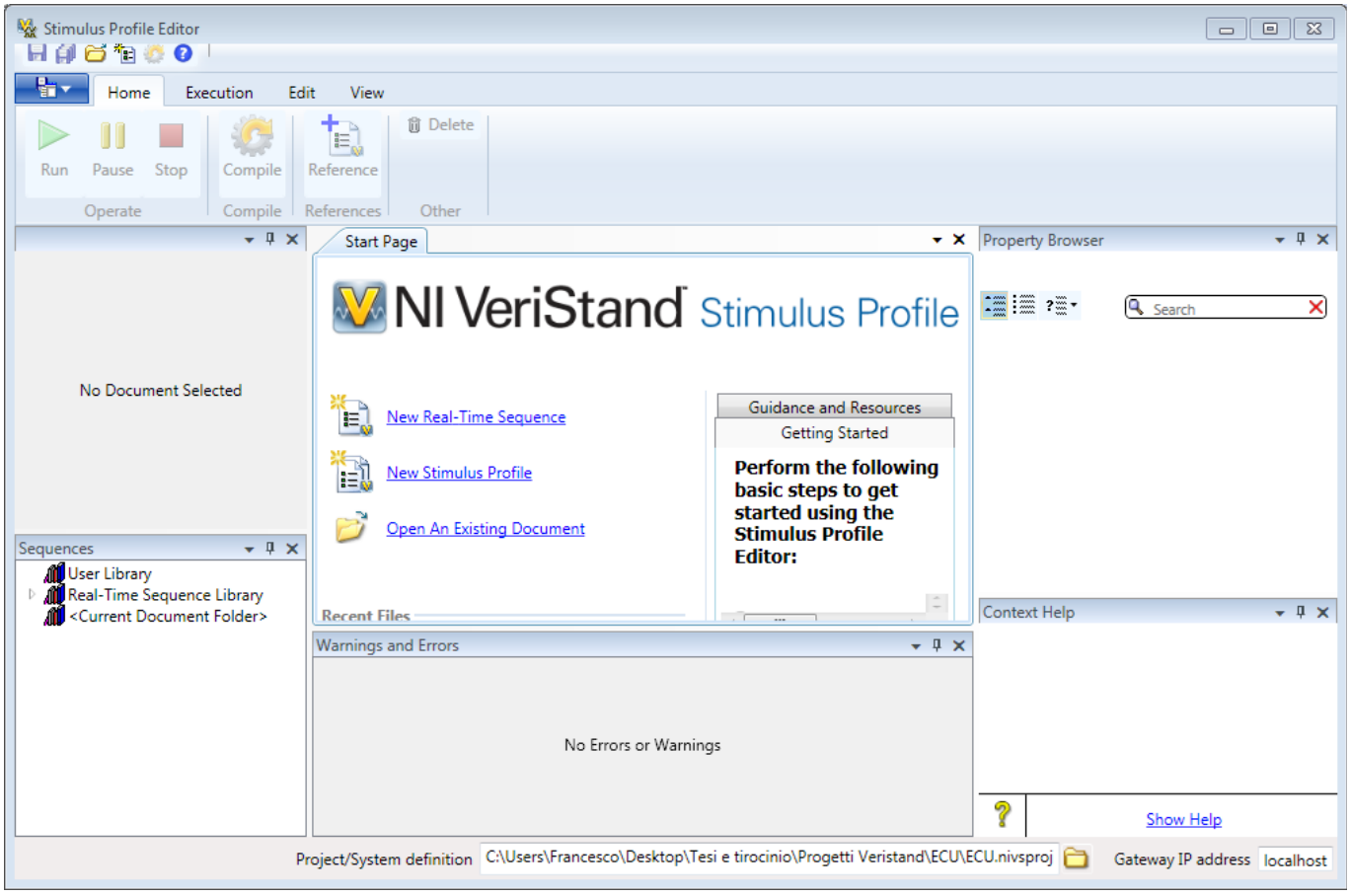
Microsemi
Watertown, MA
Progress Powered by Technology

Electrical Parameters (T_A @ 25°C unless otherwise specified)

CHARACTERISTICS	SYMBOL	MIN.	TYP.	MAX.	UNIT
Off Characteristics					
Collector-Emitter Breakdown Voltage (I_C = 10 mAdc, I_B = 0)	BV_{CE0}	50		--	Vdc
Collector-Emitter Breakdown Voltage (I_C = 10 μAdc, I_E = 0)	BV_{CBO}	75		--	Vdc
Emitter-Base Breakdown Voltage (I_E = 10 μAdc, I_C = 0)	BV_{EBO}	6.0		--	Vdc
Collector to emitter Cutoff Current (V_{CE} = 30 Vdc)	I_{CES}	--		50	nAdc
Collector to base Cutoff Current (V_{CE} = 60 Vdc)	I	--		10	nAdc
D.C. Current Gain (I_C = 0.1 mAdc, V_{CE} = 10 Vdc) (I_C = 1.0 mAdc, V_{CE} = 10 Vdc) (I_C = 10 mAdc, V_{CE} = 10 Vdc)(1) (I_C = 10 mAdc, V_{CE} = 10 Vdc, T_A = -55°C)(1) (I_C = 150 mAdc, V_{CE} = 10 Vdc)(1) (I_C = 500 mAdc, V_{CE} = 10 Vdc)(1)	h_{FE}	 50 75 100 35 100 30		 -- 325 -- -- 300 --	
Collector-Emitter Saturation Voltage(1) (I_C = 150 mAdc, I_B = 15 mAdc) (I_C = 500 mAdc, I_B = 50 mAdc)	$V_{CE(Sat)}$	 -- --		 0.3 1.0	Vdc
Base-Emitter Saturation Voltage(1) (I_C = 150 mAdc, I_B = 15 mAdc) (I_C = 500 mAdc, I_B = 50 mAdc)	$V_{BE(Sat)}$	 0.6 --		 1.2 2.0	Vdc
Current Gain-Bandwidth Product(2) (I_C = 20 mAdc, V_{CE} = 20 Vdc, f = 100MHz)	f_T	 250		 --	Mhz
Output Capacitance(3) (V_{CB} = 10 Vdc, I_E = 0, 100kHz ≤ f ≤ 1MHz	C_{OBO}	 --		 8.0	pf
Input Capacitance (V_{EB} = 0.5 Vdc, I_C = 0, 100kHz ≤ f ≤ 1MHz)	C_{IBO}	 --		 25	pf
Switching Characteristics Delay Time: (V_{CC} = 30 Vdc, $V_{BE(off)}$ = -0.5 Vdc, Rise Time: I_C = 150 mAdc, I_{B1} = 15 mAdc)(Figure 12) Storage Time: (V_{CC} = 30 Vdc, I_C = 150 mAdc, Fall Time: I_{B1} = I_{B2} = 15 mAdc)	t_{ON} t_d t_r t_{off} t_S t_f	 -- -- -- --		 10 25 225 60	ns

Utilizzo dello Stimulus Profile Editor

Una funzione preziosa di VeriStand è la possibilità di variare automaticamente i valori degli ingressi nel modello secondo sequenze prefissate. Questo consente di creare test anche molto lunghi e complessi e di lasciarli lavorare in modo del tutto automatizzato. Per impostare un test aprire la versione automatizzata del test CF3000, lo Stimulus Profile Editor si trova nella voce Tools di Project Explorer e di Workspace. Questo strumento è disponibile anche se il progetto è avviato. Vista la vastità del programma ci si occuperà solo di ciò che è necessario per creare un profilo di stimolo legato al progetto che si sta realizzando, per un tutorial molto più approfondito si può consultare il seguente documento: http://zone.ni.com/devzone/cda/tut/p/id/13604



Nella parte centrale della schermata ci sono 3 voci selezionabili.

New Real-Time Sequence permette di programmare dei comandi di base, per esempio si possono creare onde con una forma particolare. *New Stimulus Profile* consente invece di creare sequenze, con i comandi preimpostati o creati precedentemente, che modificano nel modo e nel tempo desiderato il valore dei canali o delle variabili indicate. In basso a destra il *context help* dà informazioni su tutti gli oggetti presenti nell'editor.

Creazione di un profilo di stimolo semplice

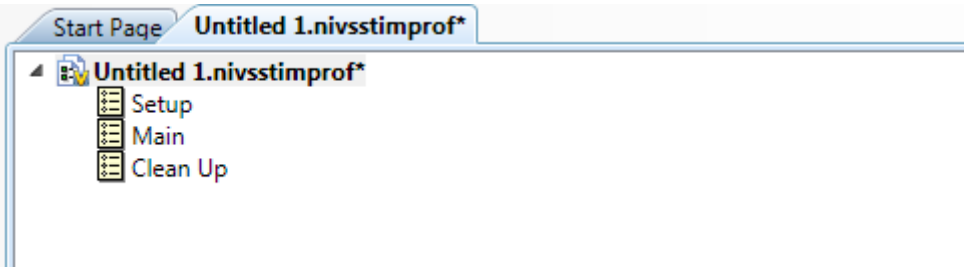
Per creare un profilo di stimolo è possibile utilizzare una serie di sequenze real-time preconfigurate, creare le proprie usando gli strumenti disponibili, oppure utilizzare delle sequenze esterne in formato .csv.

Sequenze preimpostate

Per quel che riguarda le sequenze in catalogo bisogna tenere presenti alcune differenze dalle convenzioni:

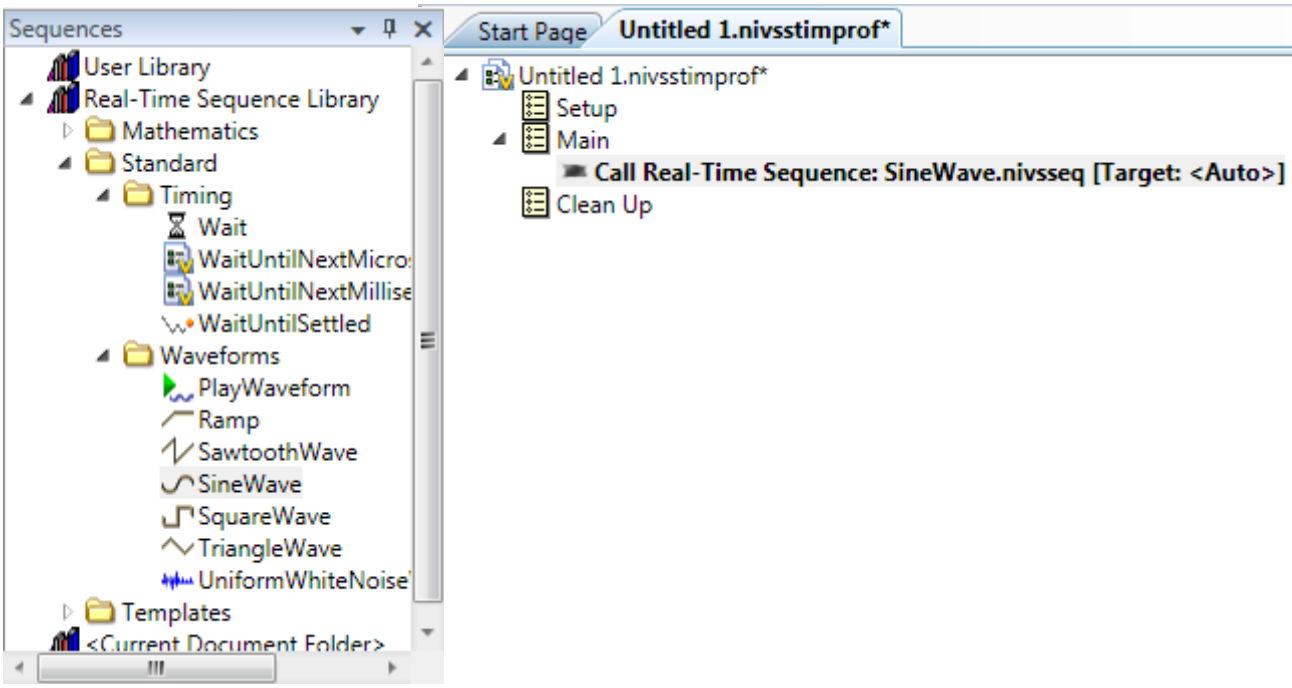
- tutti i parametri sono adimensionali quindi *frequency* non è legata al tempo ma al passo di simulazione. Per esempio una sinusoide con *frequency* = 2, applicata ad un modello con frequenza di ciclo di 200Hz, avrà una frequenza reale di 400 Hz
- Amplitude indica la distanza del picco dal valor medio. Se si sceglie amplitude 5 si avrà un'ampiezza picco-picco di 10.

Cliccando su *New Stimulus Profile* si aprirà una scheda con questo aspetto:



dal catalogo in basso a sinistra scegliere *SineWave* e trascinarla su

main.



Sul lato destro della schermata si trova *Property Browser* che merita un'analisi più dettagliata.

Property Browser

Questa scheda consente di modificare le proprietà di ognuno dei passi inseriti nella sequenza.

File Path indica dove si trova il file che, in questo caso, genera la sinusoide.

Target name è importante se il progetto è distribuito su più target perché permette di vedere solo quello scelto per il canale che si vuole manipolare. Nel presente caso lo si può lasciare in bianco o si può scegliere *CF3000.*

Timeout indica quanto tempo attendere, se lo stimolo non parte, prima di segnalare un errore. Si può lasciare alle impostazioni di default

Pass/Fail Evaluation permette di verificare il corretto funzionamento della sequenza. Si può lasciare alle impostazioni di default

SinewaveOut, come tutte le voci successive, può essere impostato con una costante o un canale. In questo caso una costante non ha senso quindi scegliere *Channel* e, nel diagramma ad albero, selezionare *Targets/CF3000/User Channels/Regime motore.*

Amplitude è l'ampiezza di una semi-onda. Impostarlo a 1000, l'ampiezza reale dell'onda sarà di 2000.

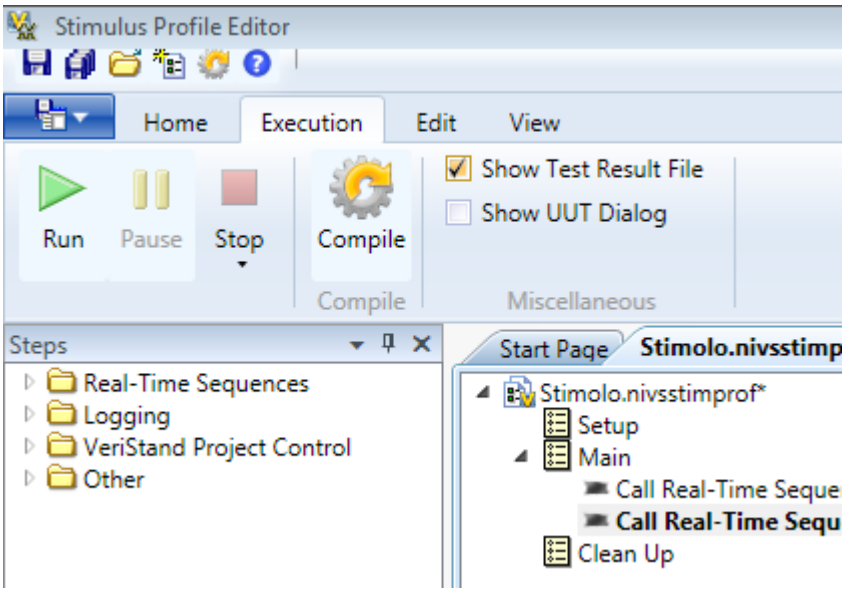
Frequency è il numero di cicli dell'onda per ogni ciclo del *Primary Control* Loop (PCL). In altre parole lasciandolo ad 1 non si vede la sinusoide perché per ogni ciclo di funzionamento del progetto la sinusoide fa un ciclo completo. Impostarlo a 0,001: un periodo ogni 1000 cicli del PCL.

Phase si può lasciare a 0

Bias è il valor medio attorno al quale la sinusoide fluttua. Si imposti a 2500.

Duration è la durata in secondi dello stimolo. Si imposti a 120.

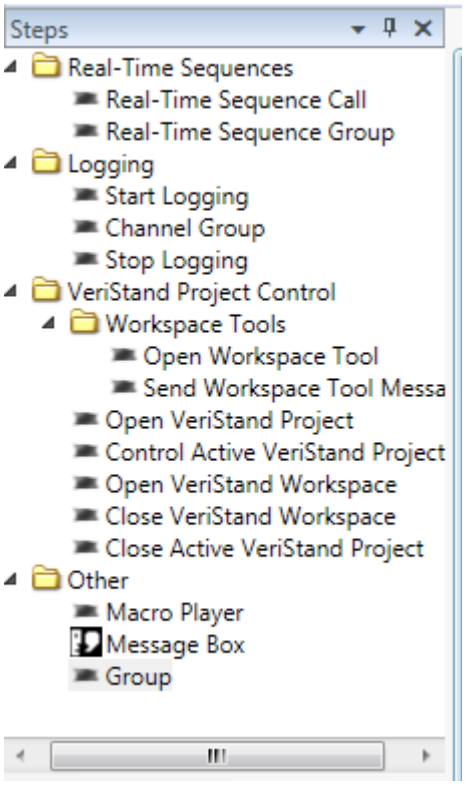
Ora si trascini *SawtoothWave* subito sotto a *SineWave* e si associ a *Targets/CF3000/User Channels/Throttle*. Si Imposti *Frequency* a 0,0005 in modo che sia la metà del periodo sinusoide, *Duration* a 120, *Bias* a 50 e si lasci il resto invariato. Si salvi il file dove si desidera e si verifichi di avere eseguito il deploy, poi si prema il tasto *Run* che si trova sotto alla scheda *Execution* e ci si sposti sul workspace.



Il dente di sega sul comando T*hrottle* pare non funzionare: in effetti si attiva solo dopo aver terminato la sequenza della sinusoide. Non importa se si è intervenuti su due canali diversi, VeriStand rispetta la sequenza. Al termine della sequenza viene mostrata una schermata di log in Internet Explorer.

Creazione di un profilo di stimolo avanzato

In VeriStand 2011 lo *Stimulus Profile* non è pensato per l'esecuzione diretta di comandi ma come piattaforma per il lancio del progetto, l'esecuzione di sequenze real-time e il logging. Per eseguire questi compiti offre una serie di comandi nel pannello in alto a sinistra.



Real Time Sequence Call si occupa di lanciare sequenze in real time: nell'esempio precedente, quando abbiamo trascinato le sequenze sul profilo, il programma ha usato questo comando.

Real-Time Sequence Group ordina a VeriStand di eseguire contemporaneamente tutti i blocchi al suo interno e non passa al gruppo successivo finché questi non sono terminati.

Group consente di organizzare visivamente il profilo ma non ha alcun impatto sulla sequenza di esecuzione. Per esempio non può essere inserito all'interno di *RT Sequence Group* quindi non è utilizzabile per costruire sequenze complesse.

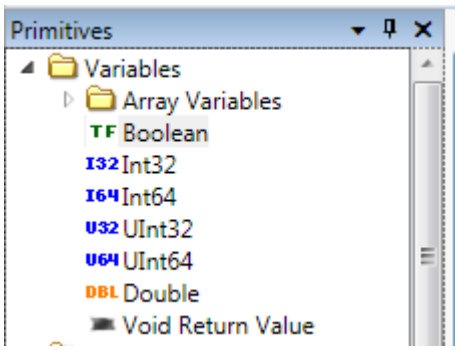
Gli altri comandi sono piuttosto semplici.

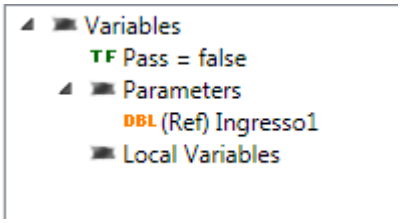
Per poter creare profili più complessi si devono creare apposite sequenze real time.

Real-Time Sequence

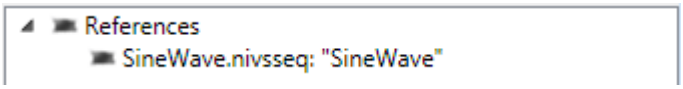
Si tratta di una lista di comandi che inviamo a uno o più canali. Per

creare un profilo per il canale *Regime motore* andare sulla scheda *Start Page* e premere *New Real-Time Sequence*. Poiché le sequenze non sono altro che calcoli matematici, prima di tutto occorre dire al programma quali variabili usare. *Regime motore* è un canale monodimensionale e in virgola mobile quindi dal pannello *Primitives* → *Variables* scegliere *Double* e trascinarlo in *Variables* → *Parameters* sul pannello a destra



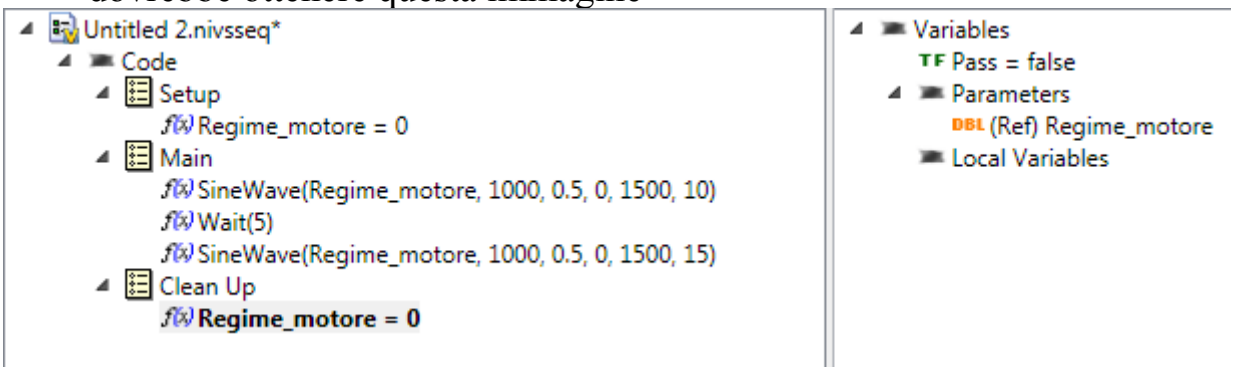


cliccare sulla variabile appena creata e rinominarla *Regime_motore*, inoltre su *Default Assignments* selezionare il canale *Targets/CF3000/User Channels/Regime motore.* Trascinare la variabile appena creata sul blocco Setup. Si crea l'espressione *Regime_motore = 0* che va bene, perché imposta la nostra variabile a 0 come punto di partenza. Ora dal pannello *Sequences* trascinare *SineWave* su *Main.* La lunga stringa tra parentesi descrive le attuali impostazioni della funzione. Per sapere esattamente cosa significano spostarsi sul pannello *References* e selezionare la funzione.

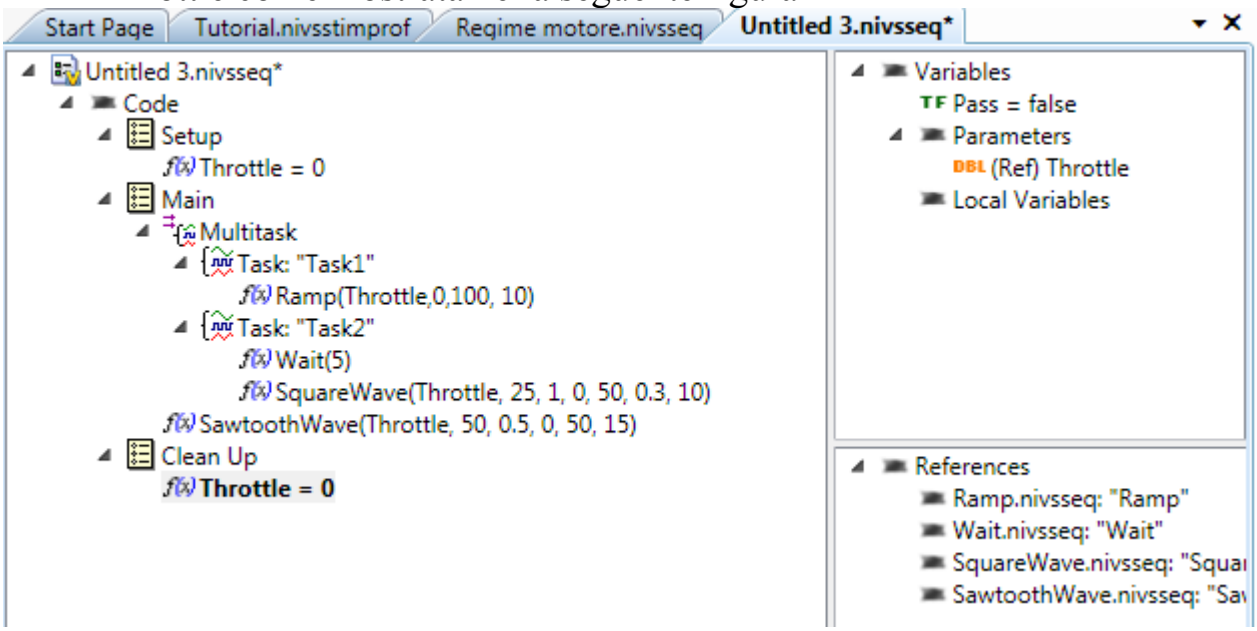


Il context help da una descrizione dettagliata di tutte le voci. Per impostare la funzione si devono sostituire tutte le voci separate da virgola con i valori e i riferimenti che interessano: inserire questa stringa **(Regime_motore, 1000, 0.5, 0, 1500, 10)**. In questo caso la frequenza è in Hz come specificato anche dalla guida. Inserire un'attesa di 5 secondi trascinando su *Main* il blocco *Wait* e sostituendo 5 alla parola *Duration*. Infine inserire un'altra sinusoide con i seguenti parametri **(Regime_motore, 1000, 0.5, 0, 1500, 15)**. Nel

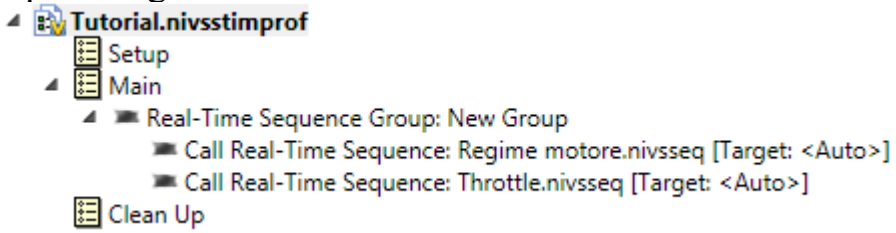
blocco *Clean Up* inserire ancora una volta *Regime_motore = 0.* Si dovrebbe ottenere questa immagine



Ora salvare la sequenza come Regime motore e creare la sequenza Throttle come mostrata nella seguente figura



Multitask è una funzione che fa girare contemporaneamente i sotto-gruppi detti task. Come si vede, sarebbe stato possibile creare un'unica sequenza che gestisce entrambi i canali ma, in caso di necessità ogni intervento correttivo sarebbe stato più complesso. A questo punto salvare e tornare a *Stimulus Profile.* Ora si deve riprodurre questa figura:



Salvare e premere *Run*. Se tutto è impostato correttamente si possono vedere i controlli *Throttle* e *Regime motore* che variano contemporaneamente.

Bibliografia e fonti

- Corti Enrico, ATTIVITà HIL CF3000 del 13/07/2011
- documento interno: MANUALETTO DI CONFIGURAZIONE ED UTILIZZO DELL'HIL PER CF3000
- CF3000 Engineering & Electronics Group: Lista canali Simulatore
- documento interno: Guida Rapida Veristand del 10/5/2012
- National Instruments: Sito ufficiale www.ni.com
- Nationa Instruments: Furum http://forums.ni.com/
- Corti Enrico, Valbonetti Manuel, Galli Davide, Vandi gabriele, Rinaldi Matteo, Bertacin Roberto, Ravaglioli Vittorio: supporto diretto, modelli matematici, documentazione varia.

Ringraziamenti

Con questo lavoro si chiude un capitolo importante della mia vita. Ci sono state grandi soddisfazioni e momenti di crisi, senso di orgoglio e di colpa, soprattutto, molte lezioni di vita. Ho conosciuto i miei limiti e imparato ad accettare le sconfitte: in ogni cosa ci sarà sempre qualcuno migliore di me, l'importante è dare sempre il meglio che si può. Se sono arrivato fino in fondo lo devo alle tante persone che mi circondano, che mi hanno sostenuto e a volte calciato nel sedere per farmi andare avanti. Non posso elencare tutti ma un ringraziamento speciale va alle seguenti persone:

- I miei genitori Lazzaro e Annamaria che mi hanno spinto e motivato nei momenti di difficoltà, nonostante le minacce mi hanno lasciato entrare in casa fino a veneranda età, si sono occupati dei miei bisogni primari e hanno curato il decoro del mio abbigliamento
- I miei fratelli Simone, Emanuele e Beatrice (in ordine di dimensione) che mi hanno sempre sostenuto dicendomi che ero bravissimo. Sospetto che lo dicessero solo quando ero presente.
- L'Istituto Agrario che mi ha insegnato moltissime cose belle ma non matematica e fisica
- I parenti che mi hanno pasciuto durante le feste e quelli che con me si sono pasciuti
- Il nevone del '12 che rimandando l'ultimo esame mi ha dato l'opportunità di contribuire ancora una volta al sostentamento dell'università Italiana
- Gli scout che con la vita all'aria aperta, le uscite in tenda, il fuoco serale mi hanno preparato fin dall'infanzia ad affrontare le conseguenze della crisi economica
- Marika che mi ha passato gli appunti di una buona metà degli esami di specialistica e mi ha schiavizzato per scrivere una relazione che valeva ben 1 punto sul voto d'esame

Ringraziamenti

- La mia ragazza che mi ha sempre lasciato massima libertà per lo studio e per coltivare i miei interessi, un giorno saprò chi è!
- Il professor Corti che ha accettato un tesista che non aveva idea di cosa fosse un sistema HIL e che non aveva mai seguito il corso relativo alla tesi
- I tecnici di laboratorio che mi hanno aiutato a capire cosa il professore volesse da me
- Il software VeriStand che mi ha insegnato la pazienza
- Il software LabView che mi ha insegnato l'impazienza
- Il coro polifonico che, per distrarmi dalla tensione degli esami, organizzava concerti durante le sessioni d'esame