Scuola universitaria professionale
della Svizzera italiana

Dipartimento
Tecnologie
Innovative

Introduzione a Stateflow

Prof. Roberto Bucher

19 giugno 2013

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Basi

Stateflow è un toolbox di MATLAB che permette la modellazione e la simulazione di macchine a stati e diagrammi di flusso.
Stateflow permette di modellare sistemi complessi che comprendono

- Simulazione di logiche complesse
- Simulazione di sistemi gerarchici e paralleli con operatori temporali ed eventi
- Diagrammi di stato, tavole di transizione di stati e matrici di transizione legati a macchine a stati
- Diagrammi di flusso e tavole di verità

L'ambiente stateflow permette inoltre le fasi si analisi e debugging, grazie a metodi di animazione delle attività e sistemi di controllo di integrità.
Attenzione: Stateflow necessita di un compilatore C anche per la fase di simulazione. Lo schema di Stateflow viene in effetti compilato in modo da generare una libreria dinamica per la simulazione.

1.2 L'ambiente di Stateflow

Stateflow viene integrato in uno schema di Simulink, con un blocco specifico, visibile nella figura 1.1

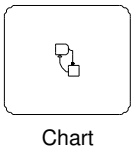

Figura 1.1: Blocco di una macchina a stati in SIMULINK

Un doppio click con il mouse sul blocco di Stateflow permette di aprire i vero e proprio ambiente di sviluppo (vedi figura 1.2).

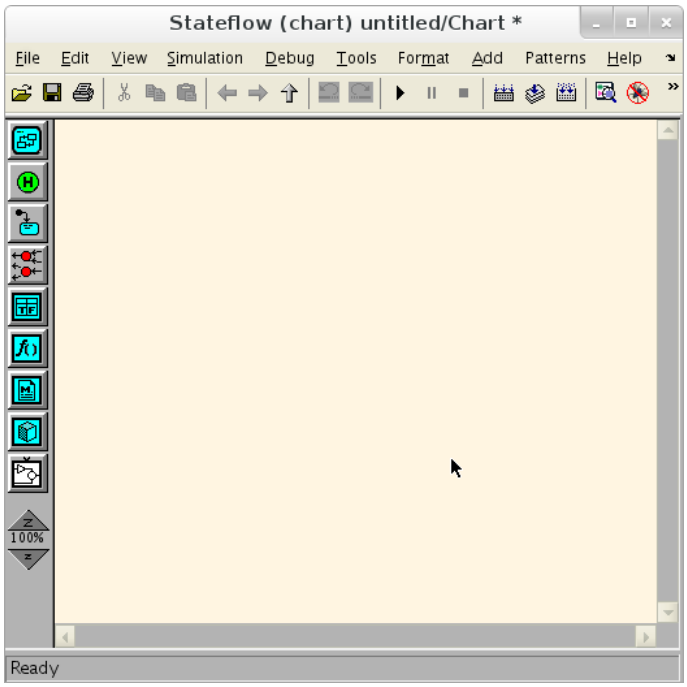

Figura 1.2: Gui di sviluppo per Stateflow

1.3 Gli elementi dello schema di Stateflow

1.3.1 Lo schema completo

Analizziamo ora la figura 1.3, che mostra un tipico schema di Stateflow. Lo schema rappresenta un sistema di due ventilatori che vengono accesi per raffreddare un ambiente. Il primo ventilatore (COOLER 1) viene acceso quando la temperatura T supera i 30^o, mntre il secondo viene acceso quando la temperatura T supera i 40^o.
Possiamo ora analizzare i singoli elementi di questo schema.

1.3.2 Stati esclusivi (OR)

Gli stati esclusivi del sistema sono rappresentati con una linea continua (figura 1.4).
Lo schema contiene 6 stati esclusivi:

- Power Off
- Power Off
- COOLER 1: On
- COOLER 1: Off
- COOLER 2: On
- COOLER 2: Off

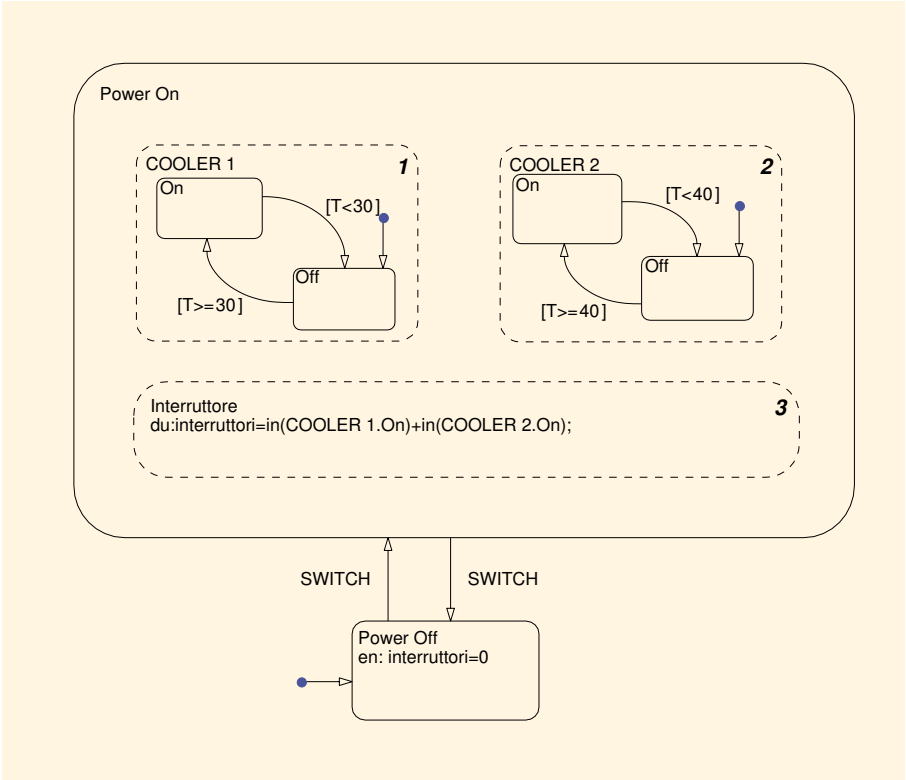

Figura 1.3: Schema di Stateflow

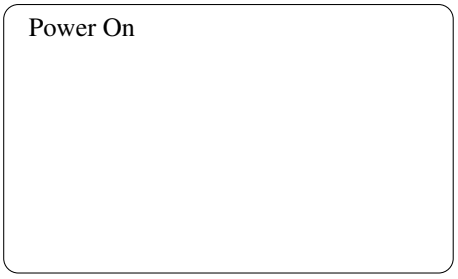

Figura 1.4: Stato esclusivo

A seconda della gerarchia dello schema questi stati possono esistere solo in modalità esclusiva:

- Power On o Power Off
- In COOLER 1: On oppure Off
- In COOLER 2: On oppure Off

1.3.3 Stati paralleli (AND)

Gli stati paralleli sono quelli contenuti nello stato "Power On", e sono caratterizzati dalla linea tratteggiata e dalla numerazione (figure 1.5). Per poter creare degli stati paralleli occorre modificare la proprietà "Decomposition" dopo aver cliccato con il mouse destro sullo stato "Power On", scegliendo quindi il sottomenu "Parallel".

Figura 1.5: Stato parallelo

La numerazione (numero in alto a destra nello stato) permette di gestire la sequenza con cui i vari stati interni vengono eseguiti (Sequenza: COOLER 1 $\rightarrow$ COOLER 2 $\rightarrow$ Interruttore).

1.3.4 Transizioni

Le transizioni sono rappresentate da frecce direzionate che vanno da uno stato ad un altro stato (figura 1.6).

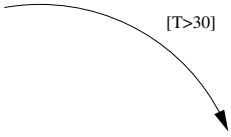

Figura 1.6: Transizione con condizione

Accanto alla freccia ci sono ad esempio le condizioni di transizione (es. $[T > 30]$).
Un caso particolare di transizione è quello della transizione di default in un sistema esclusivo (figura 1.7).

Figura 1.7: Transizione di default

1.3.5 Giunzioni

Le giunzioni sono rappresentate da cerchi cui possono giungere più transizioni, rispettivamente partire nuove transizioni. Nel caso di più transizioni in uscita da una giunzione, viene determinata la sequenza di esecuzione. In genere permettono di evitare di introdurre stati fittizzi nello schema stateflow.

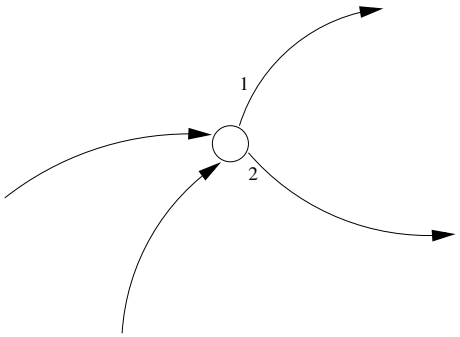

Figura 1.8: Giunzione

1.3.6 Giunzioni storiche

Sono dei cerchi contenenti la lettera "H". Se messi all'interno di uno stato (vedi figura 1.9 in alto a destra in *STATE1*) permettono di ricordare l'ultimo stato interno in cui ci si trovava prima di una transizione.
In questo esempio, se l'operazione di transizione da *STATE1* a *STATE2* fosse avvenuta mentre ci si trovava nello stato *US2*, al prossimo evento *SWITCH* per tornare in *STATE1*, ci si troverebbe nuovamente in *US2*, e non in *US1* (stato di default), cosa che invece avverrebbe se non ci fosse la giunzione storica all'interno di *STATE1*.
In caso di dubbio verrebbe attivato lo stato di default.

1.4 Tipi di azioni supportate per stati e transizioni

1.4.1 Tipi di azioni negli stati

All'interno degli stati sono possibili diverse azioni come descritto nella tabella 1.1.

1.4.2 Tipi di azioni sulle transizioni

Una transizione viene associata alla stringa seguente

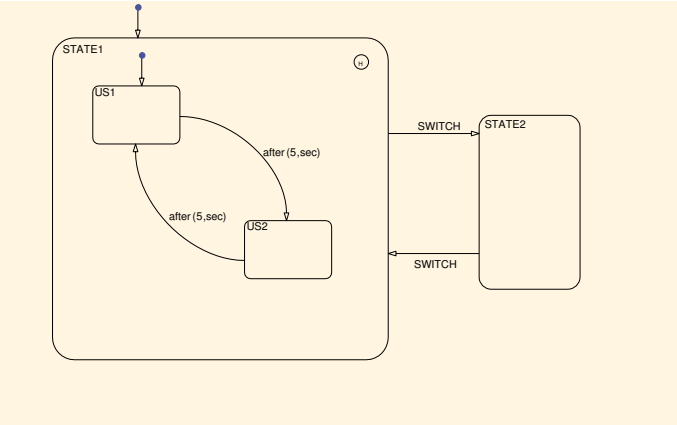

Figura 1.9: Giunzione storica

Azione	Abbr.	Significato
entry	en	esegue le operazioni quando lo stato diventa attivo.
exit	ex	Esegue le operazioni prime della transizione dallo stato.
during	du	Esegue l'azione quando lo stato è attivo e arriva un evento specifico.
bind		Collega un evento o un dato allo stato in modo che solo questo stato possa gestire l'evento o modificare il dato.
on *event_name*		esegue l'operazione solo se riceve l'evento specifico.

Tabella 1.1: Operazioni negli stati

event_trigger[condition]{condition_action}/transition_action

1.4.2.1 Event_trigger

Se definito, è il primo nome nella transizione, senza particolari formattazioni. Può anche non essere specificato, ed in questo caso ogni evento effettua la transizione. Più eventi possono essere definiti con un operatore logico OR (|).

1.4.2.2 Condition

La condizione è un'operazione booleana che determina se la transizione può essere effettuato oppure no. Condizioni possono essere:

- espressioni booleane
- funzioni che ritornano vero o falso
- L'operazione *in(nome_stato)* che definisce se lo stato definito è attivo

- condizioni temporali (vedi capitolo specifico).

1.4.2.3 Condition action

Sono operazioni messe tra due parentesi graffe () e vengono eseguite se la condizione è vera, prima che la destinazione della transizione sia raggiunta (stato o giunzione). A differenza della *transition action* l'operazione viene eseguita anche se poi in seguito non sarà possibile raggiungere un nuovo stato.

1.4.2.4 Transition action

Queste azioni sono precedute da un segno / e possono essere eseguite solo se è possibile raggiungere la fine della transizione, altrimenti vengono ignorate. Ciò significa ad esempio che se questa azione porta ad un giunto, deve essere possibile continuare e raggiungere lo stato finale.

1.4.3 Esempio di transizioni

Analizziamo l'esempio proposto da Mathworks per comprendere meglio le operazioni di transizione. L'esempio è mostrato nella figura 1.10.
I due sistemi lavorano in parallelo.
Il sistema descritto con "TransAction State" effettua le operazioni seguenti:

T=0.0 Lo stato TA diventa attivo. In entrata la variabile outVal diventa 0.

T=0.1 Avviene la transizione da TA a J1 (nessuna condizione.
Viene fatta la transizione da J1 a J2 (1. nella sequenza), la condizione è vera (1).
L'operazione di transizione non viene fatta (decremento di outVal) poiché la condizione tra J2 e TB è falsa. Si torna su J1.
Viene fatta la transizione da J1 a J3 (2. nella sequenza) non essendoci condizioni.
La condizione tra J3 e TB è falsa. L'esecuzione torna a TA.

T=0.2-1.0 Si ripete quanto successo con T=0.1

T=1.1 Avviene la transizione da TA a J1 (nessuna condizione.
Viene fatta la transizione da J1 a J2 (1. nella sequenza), la condizione è vera (1).
L'operazione di transizione non viene fatta (decremento di outVal) poiché la condizione tra J2 e TB è falsa. Si torna su J1.
Viene fatta la transizione da J1 a J3 (2. nella sequenza) non essendoci condizioni.
La condizione tra J3 e TB è vera. Lo stato TB diventa attivo. Essendo l'operazione di transizione completa, può essere eseguita l'operazione di transizione outVal=0.

Il sistema descritto con "CondAction State" effettua le operazioni seguenti:

T=0.0 Lo stato CA diventa attivo. In entrata la variabile outVal_2 diventa 0.

T=0.1 Avviene la transizione da CA a J1 (nessuna condizione.
Viene fatta la transizione da J1 a J2 (1. nella sequenza), la condizione è vera (1).
L'operazione di condizione viene fatta (decremento di outVal_2). Si torna su J1.
Viene fatta la transizione da J1 a J3 (2. nella sequenza) non essendoci condizioni.
La condizione tra J3 e CB è falsa. L'esecuzione torna a CA.

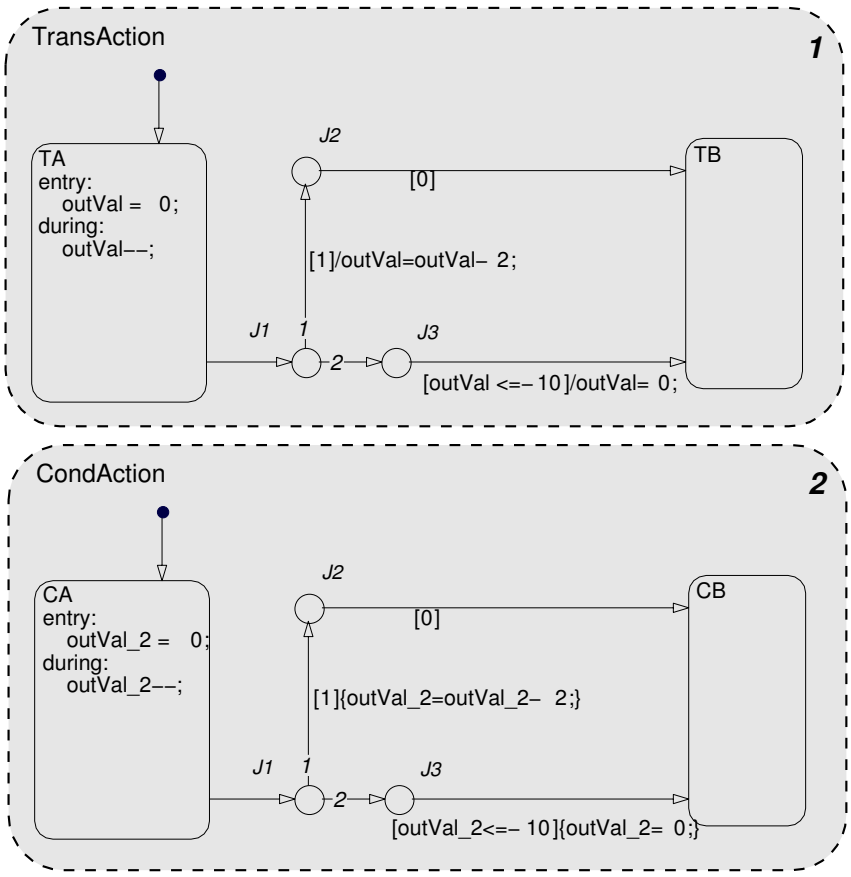

Figura 1.10: Esempio di transizioni

T=0.2-0.3 Si ripete quanto successo con T=0.1

T=0.4 Avviene la transizione da CA a J1 (nessuna condizione.
Viene fatta la transizione da J1 a J2 (1. nella sequenza), la condizione è vera (1). L'operazione di consizione viene fatta (decremento di outVal_2). Si torna su J1.
Viene fatta la transizione da J1 a J3 (2. nella sequenza) non essendoci condizioni. La condizione tra J3 e CB è vera. Lo stato CB diventa attivo. Essendo l'operazione di transizione completa, può essere eseguita l'operazione di transizione outVal_2=0.

1.5 Operatori di logica temporale

Si tratta di operatori booleani che permettono di introdurre condizioni legate al numero di eventi occorsi. Possono essere utilizzate in stati e transizioni. Nei vari casi "E" rappresenta un evento specifico e "n" un numero intero positivo.
La tabella 1.2 mostra la sintassi di questi comandi, mentre la tabella refT3 mostra alcuni esempi.

Operatore	Sintassi	Descrizione
after	after(n, E)	Ritorna "vero" se l'evento "E" è avvenuto almeno "n" volte dopo l'attivazione dello stato. Il contatore dell'evento "E" viene azzerato ogni volta che lo stato diventa attivo.
before	before(n, E)	Ritorna "vero" se l'evento "E" è avvenuto meno di "n" volte dopo l'attivazione dello stato. Il contatore dell'evento "E" viene azzerato ogni volta che lo stato diventa attivo.
at	at(n, E)	Ritorna "vero" quando l'evento "E" è avvenuto per la "n.ma" volta dopo l'attivazione dello stato. Il contatore dell'evento "E" viene azzerato ogni volta che lo stato diventa attivo.
every	every(n, E)	Ritorna "vero" ad ogni n.ma attivazione dell'evento "E" dopo l'attivazione dello stato. Il contatore dell'evento "E" viene azzerato ogni volta che lo stato diventa attivo.
temporalCount	temporalCount(E)	Incrementa di 1 e ritorna il valore dell'evento "E" ad ogni attivazione di "E" nello stato specifico. Il contatore dell'evento "E" viene azzerato ogni volta che lo stato diventa attivo.

Tabella 1.2: Logica temporale

1.5.1 Logica con tempo assoluto

Le operazioni legate a stati e transizioni possono anche essere fatte dipendere da valori assoluti di tempo (tabella 1.4).

Op.	Dove	Esempio	Descrizione
after	stato	on after(5,CLK): status('on')	Chiama la funzione "status" ad ogni evento CLK, partendo dopo 5 eventi dall'attivazione dello stato.
after	Transition	ROTATE[after(10,CLK)]	La transizione avviene all'arrivo dell'evento "ROTATE" ma dopo almeno 10 volte l'arrivo dell'evento "CLK"

Tabella 1.3: Esempi di logica temporale

Operatore	Sintassi	Descrizione
after	after(n, sec)	Ritorna "vero" se sono passati meno di n secondi dall'attivazione dello stato. Resetta il tempo al momento dell'attivazione dello stato
before	before(n, sec)	Ritorna "vero" se sono passati almeno n secondi dall'attivazione dello stato. Resetta il tempo al momento dell'attivazione dello stato
temporalCount	temporalCount(sec)	Incrementa e ritorna il numero di secondi passati dall'attivazione dello stato. IL contatore dei secondi viene azzerato ogni volta che lo stato diventa attivo.

Tabella 1.4: Logica a tempi assoluti

Capitolo 2

Costruzione di un diagramma stateflow

2.1 Esempio

Quale primo esempio in Stateflow viene creato un contatore up/down.
Questo semplice sistema contiene due stati "counter_up" e "counter_down".
Il sistema comprende la possibilità di accendere e spegnere il contatore, e quindi, quando il contatore è acceso, effettuare le operazioni su e giù.
Il contatore si incrementa ad ogni clock fino a raggiungere il valore "10" e quindi poi inizia a decrementare ritornando a "0". Poi ricomincia il ciclo.
La prima operazione da fare in matlab è lanciare il comando "sf" nel workspace. Viene creato uno schema di simulink come da figura 2.1.

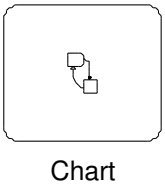

Figura 2.1: Diagramma di Simulink

2.2 Diagramma stateflow

Un doppio click con il mouse apre l'editore di stateflow come da figura 1.2.
A questo punto è possibile introdurre i due stati "acceso" e "spento" e rinominarli (figure refF22).
La transizione da "spento" ad "acceso" e da "acceso" a "spento" avviene tramite l'evento "SWITCH" (figura 2.3).
Nello stato "spento" possiamo mettere l'inizializzazione di una variabile "counter" a zero, e definire questo stato come "default" (figura 2.4).

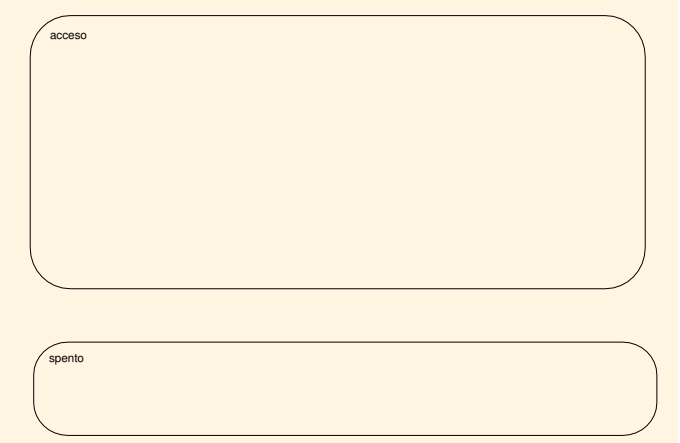

Figura 2.2: Stateflow - step 1

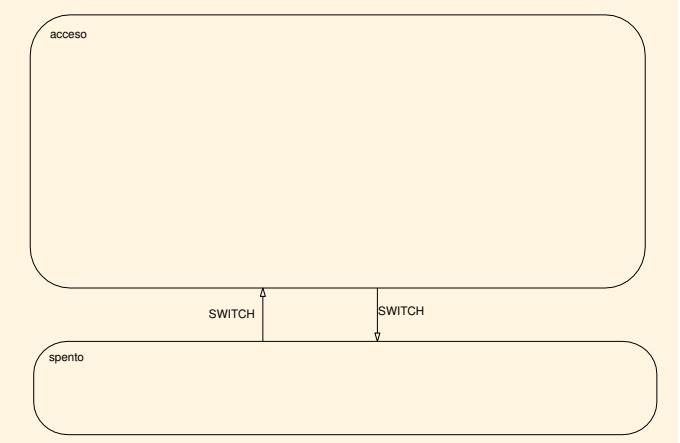

Figura 2.3: Stateflow - step 2

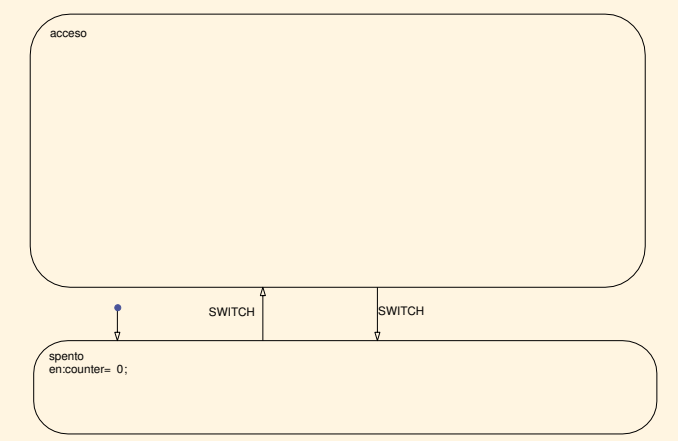

Figura 2.4: Stateflow - step 3

All'interno dello stato "acceso" costriuiamo un nuova macchina a stati con i due stati "counter_up" e "counter_down". I due stati sono di tipo esclusivo (figura 2.5).

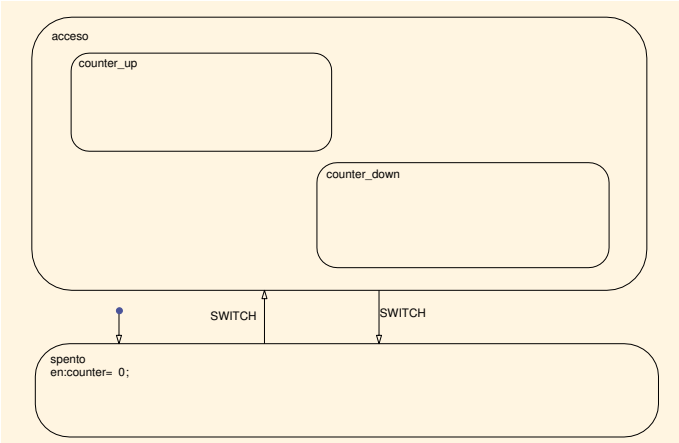

Figura 2.5: Stateflow - step 4

A questo punto si possono definire le operazioni da effettuare nei due stati ad ogni evento: le operazioni sono specificate nelle sezione "du" (during) all'interno dello stato (figura 2.6). Definiamo anche lo stato "counter_up" come stato di defult all'entrata.

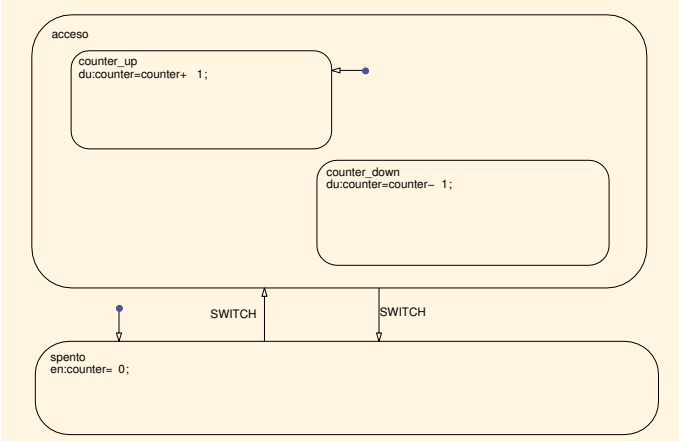

Figura 2.6: Stateflow - step 5

Definiamo ora le transizioni. La transizione dallo stato "counter_up" a "counter_down" avviene quando il contatore raggiunge il valore "10", mentre quella dallo stato "counter_down" a "counter_up" avviene quando il contatore è tornatoa "0" (figura 2.7).

2.3 Interfacciamento con Simulink

A questo punto è possibile interfacciare il nostro sistema stateflow con simulink, in modo da rappresentare graficamete il risultato in uno scope.
Inoltre è necessario fornire gli eventi esterni da simulink per gestire il diagramma.

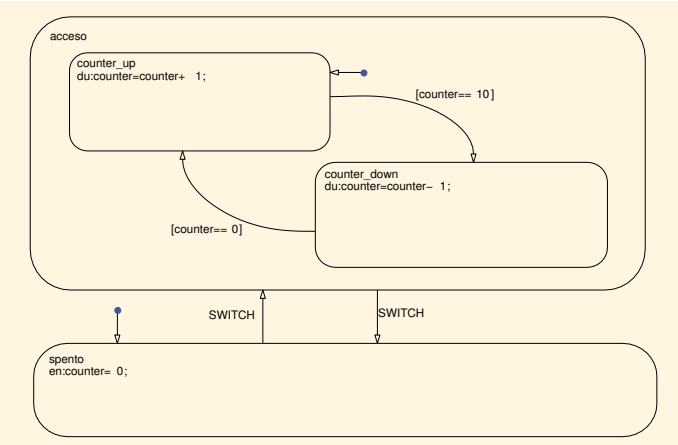

Figura 2.7: Stateflow - step 6

Queste operazioni si effettuano nella fnestra stateflow.

2.3.1 Output a Simulink

Scegliendo il menu "Add" → "Data" → "Output to Simulink" è possibile definire la variabile "counter" come uscita (figura 2.8).
Sul blocco stateflow compare una uscita chiamata "counter".

2.3.2 Trigger degli eventi

Il trigger degli eventi è formato da due segnali:

- Il primo segnale viene da un generatore di impulsi con periodo di 1 sec.
- Il secondo segnale è costruito cme la differenza tra due gradini: il primo passa a 1 dopo 10 sec., mentre il secondo va a 1 dopo 50 sec.. Si crea quindi un segnale che va ad 1 dopo 10 sec. e ritorna a 0 dopo 50.

Occorre ora creare una entrata "evento" al diagramma stateflow: da menu si sceglie "Add" → "Event" → "Input from Simulink" e si riempie il dialogo di figura 2.9.
Si tratta di creare due segnali: il primo viene chiamato clock, associato al generatore di impulsi, e reagisce quando il segnale sale (trigger: "rising"), mentre il secondo viene associato alla variabile "SWITCH" con trigger su "Either".
Sul blocco stateflow compare un simbolo di evento.

2.3.3 Schema finale

La figura 2.10 mostra lo schema di simulink pronto per la simulazione.
Prima della simulazione occorre mettere il tempo finale ad esempio a 100 sec.
Quando viene lanciata la simulazione viene generata una libreria dinamica dallo schema stateflow e quindi poi viene fatta la simulazione.
La figura 2.11 mostra l'output della simulazione.

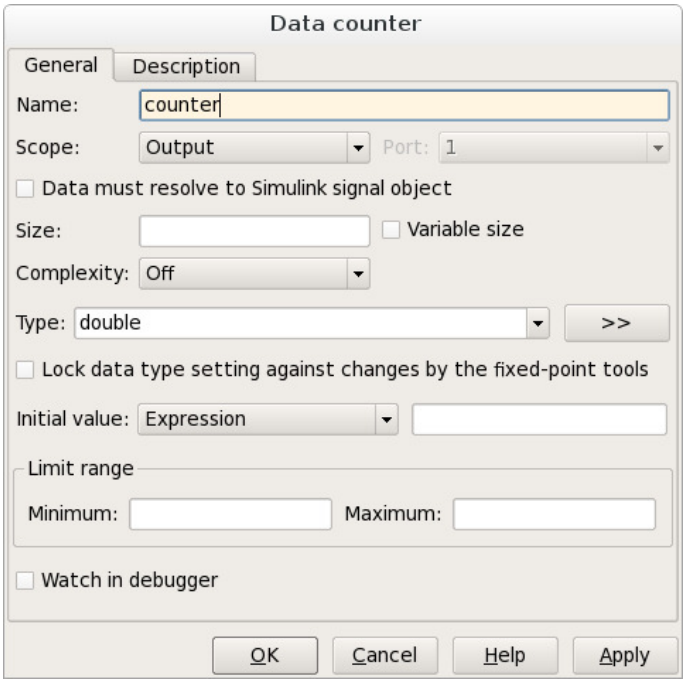

Figura 2.8: Stateflow - step 7

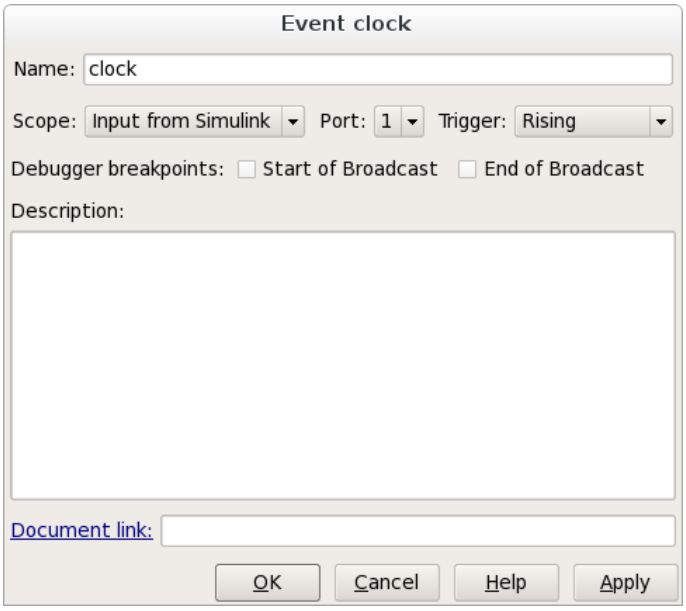

Figura 2.9: Stateflow - step 8

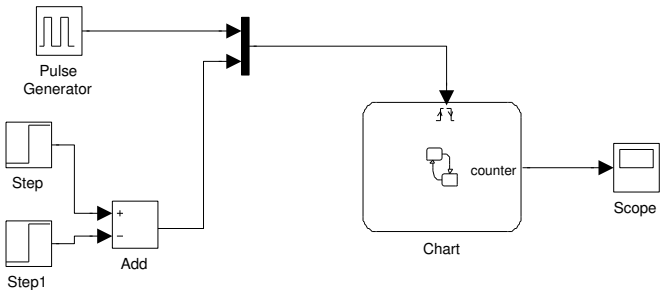

Figura 2.10: Simulink - schema finale

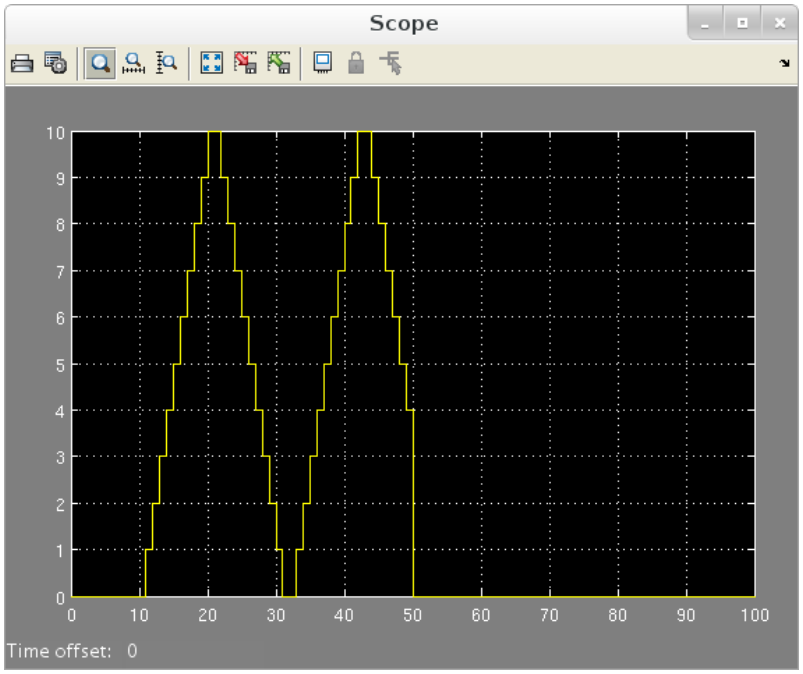

Figura 2.11: Risultato della simulazione

Capitolo 3

Trigger e Sampling Time

Analizziamo lo schema di figura 3.1, con una macchina a stati che implementa un relay.

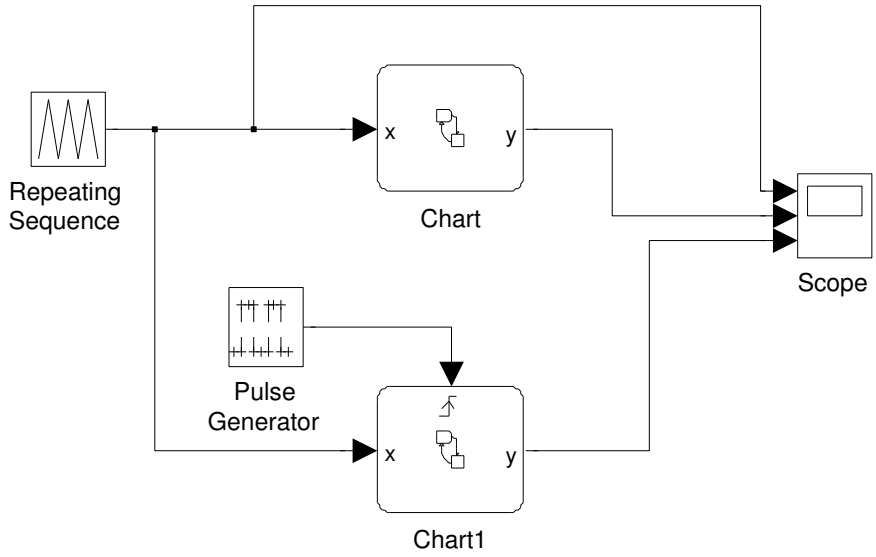

Figura 3.1: Schema Simulink

In entrambi blocchi stateflow la macchina a stati è la stessa (figura 3.2), ma nella macchina a stati inferiore c'è un trigger.
Nella macchina a stati superiore impostiamo tramite il menu "Tools" → "Explore" il tempo di campionamento come discreto a 0.01sec., mentre nella secona macchina a stati il tempo di campionamento è ereditato (impostazione obbligatoria).
Le impostazioni per il trigger sono visibili nella figura 3.3. Da notare che il campo "Phase delay" ìmpostato a "0".
La figura 3.4 mostra il risultato della simulazione.
Come si può vedere dalla figura 3.4 le due risposte allo stesso segnale d'entrata sono diverse, perchè nel grafico inferiore prima di attualizzare il valor in uscita della macchina a stati occorre attendere il rossimo evento di trigger, che avviene solo dopo un secondo.
Con un minimo cambiamento nelle impostazioni del segnale di trigger ("Phase Delay" messo a "1") (figura 3.5) la visualizzazione grafica cambia completamente.

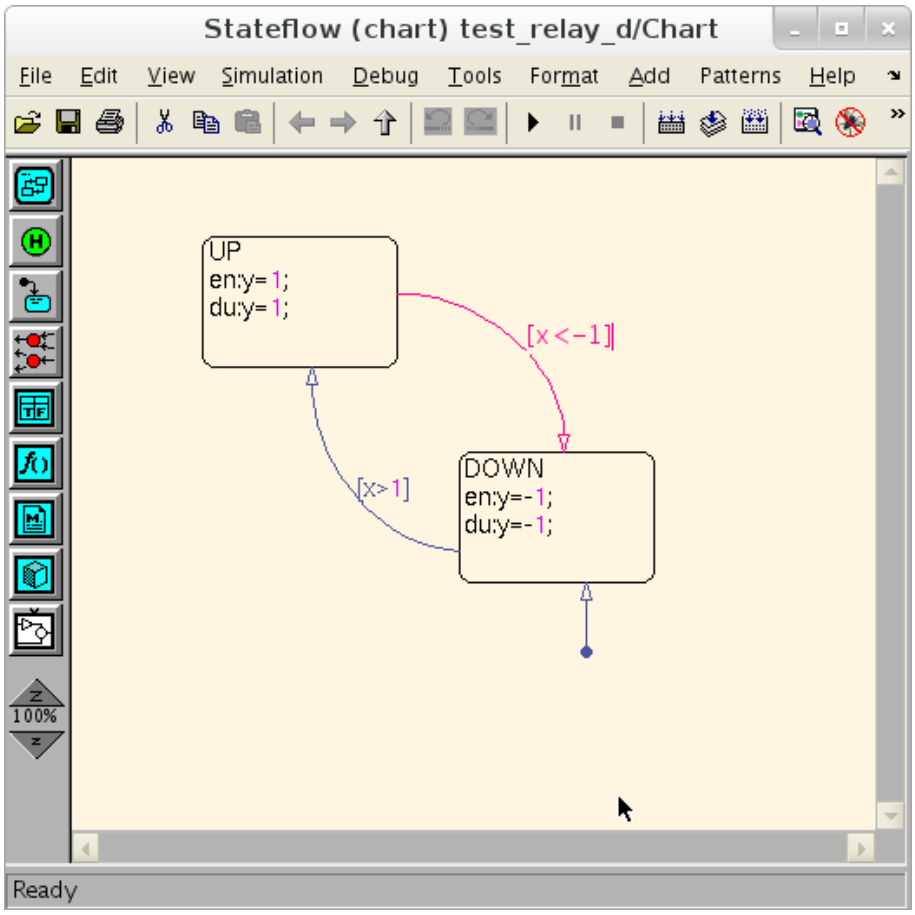

Figura 3.2: Macchina a stati

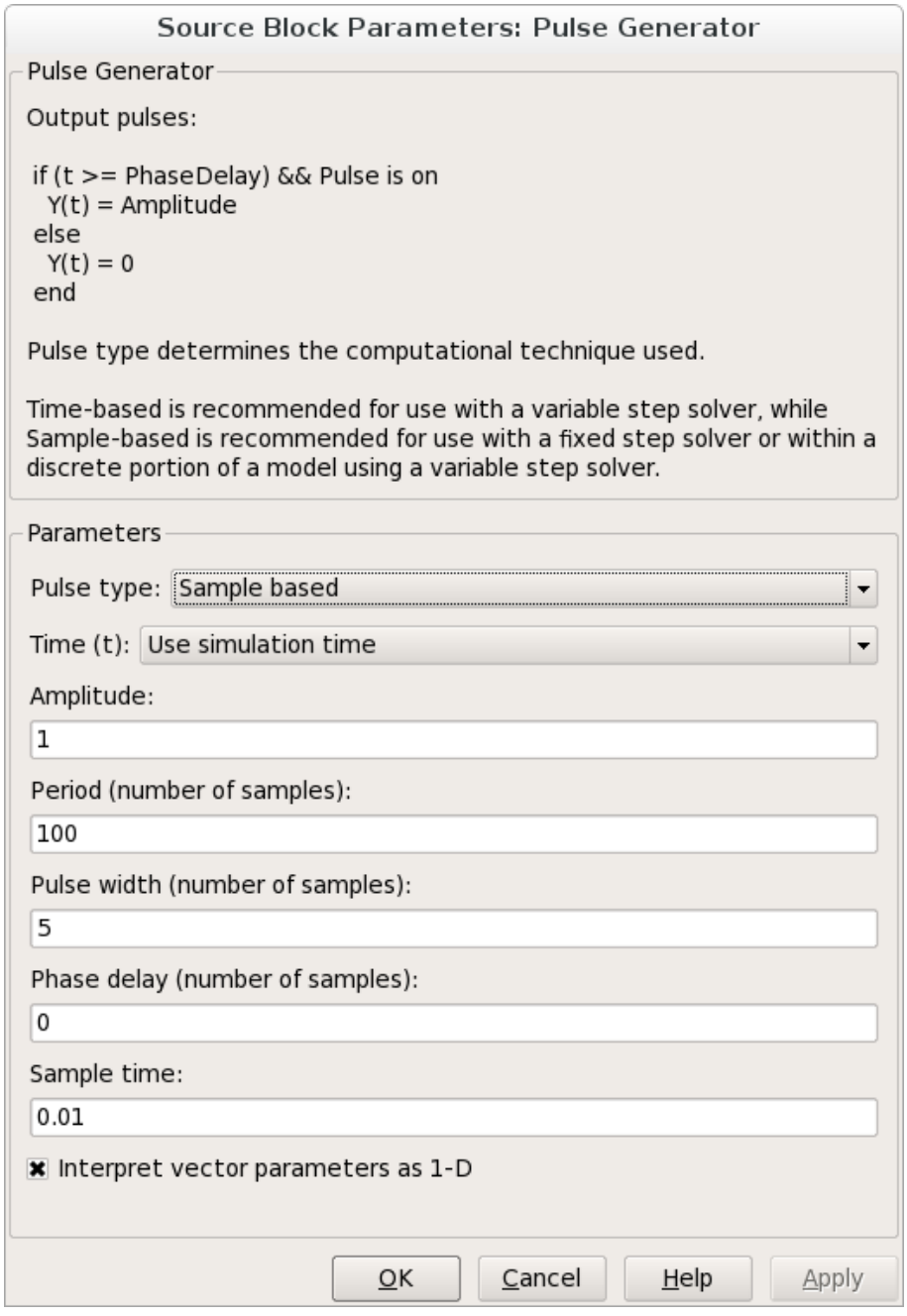

Figura 3.3: Impostazioni del segnale di trigger

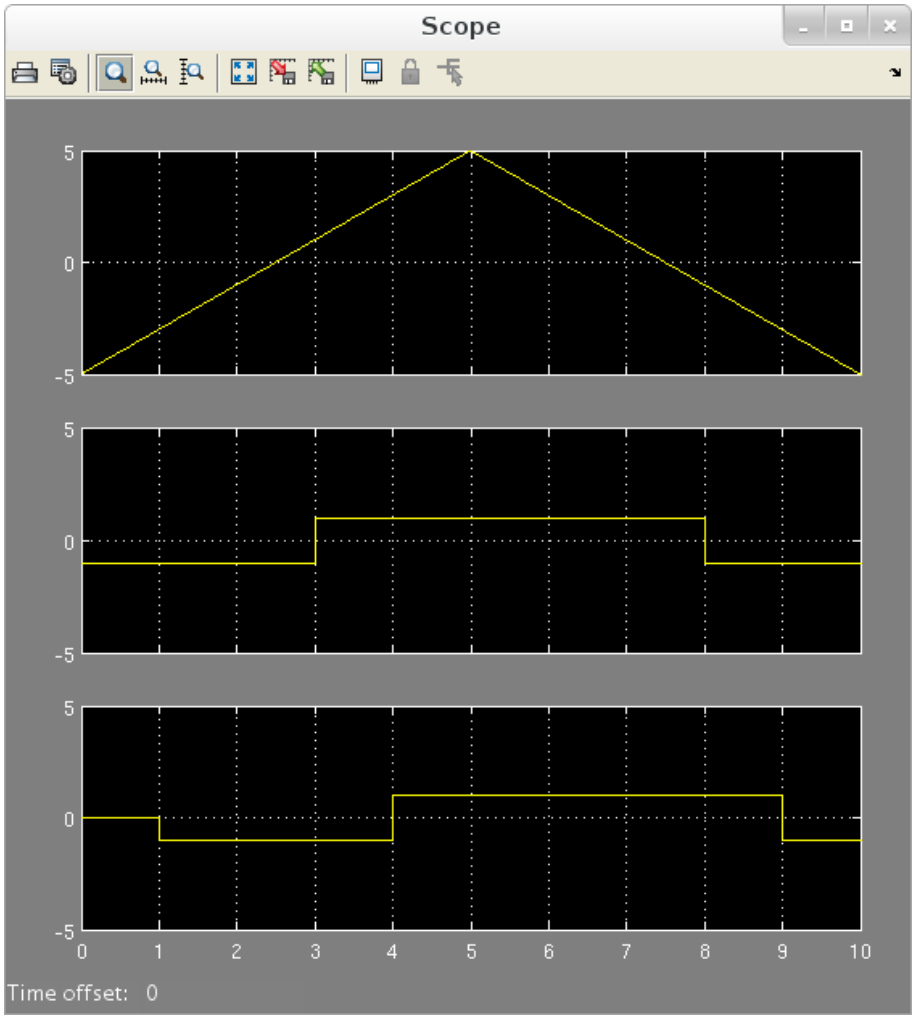

Figura 3.4: Grafico della simulazione

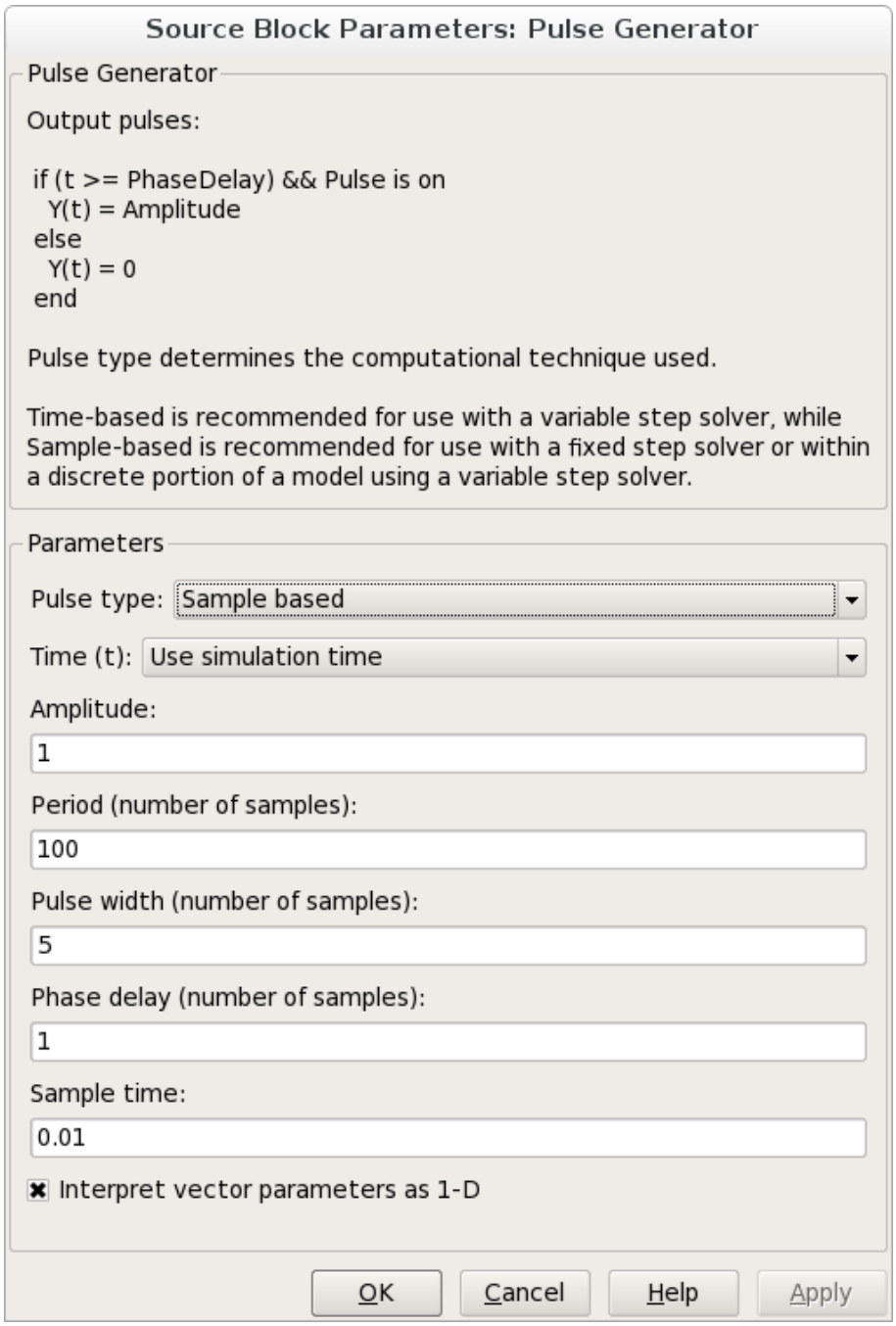

Figura 3.5: Nuova impostazione del trigger

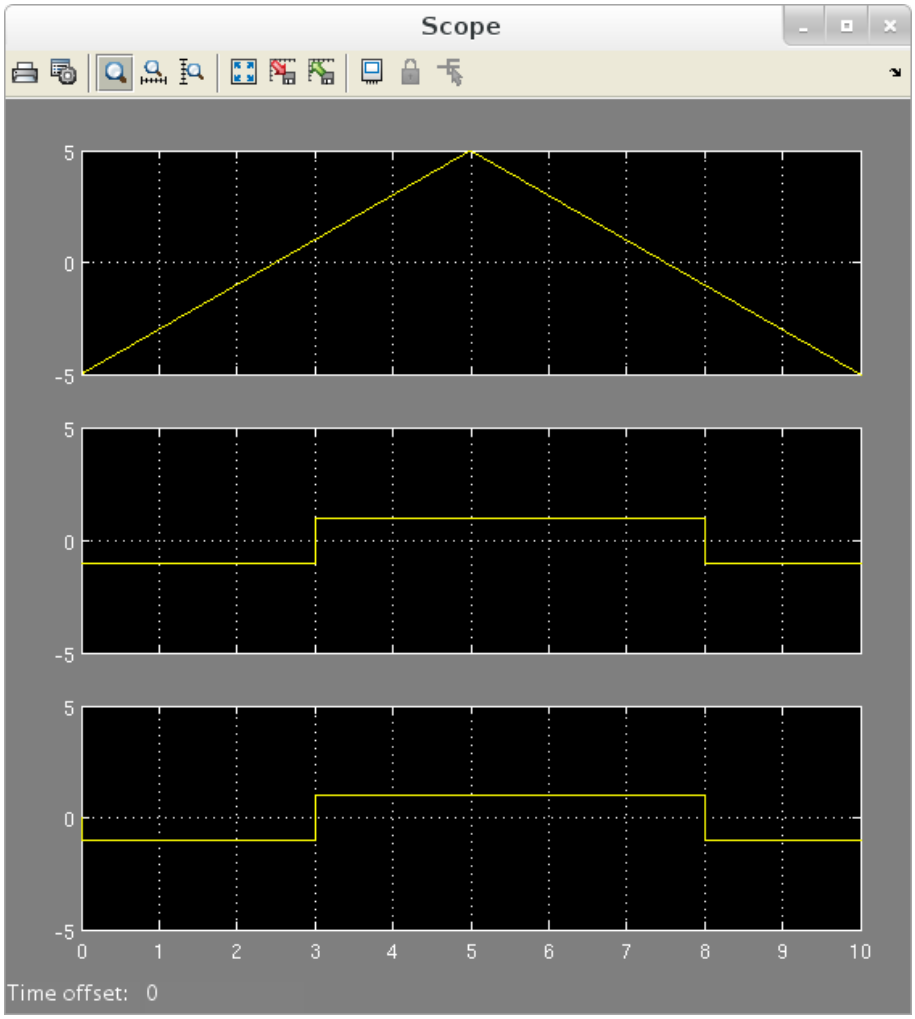

Figura 3.6: Grafico della simulazione

Capitolo 4

Funzioni e tabelle

4.1 Basi

All'interno di uno schema stateflow è possibile introdurre elementi quali funzioni grafiche, funzioni Matlab e tabelle di verità.

I tre elementi sono accessibili e configurabili all'interno dell'editore di stateflow (figura 4.1).

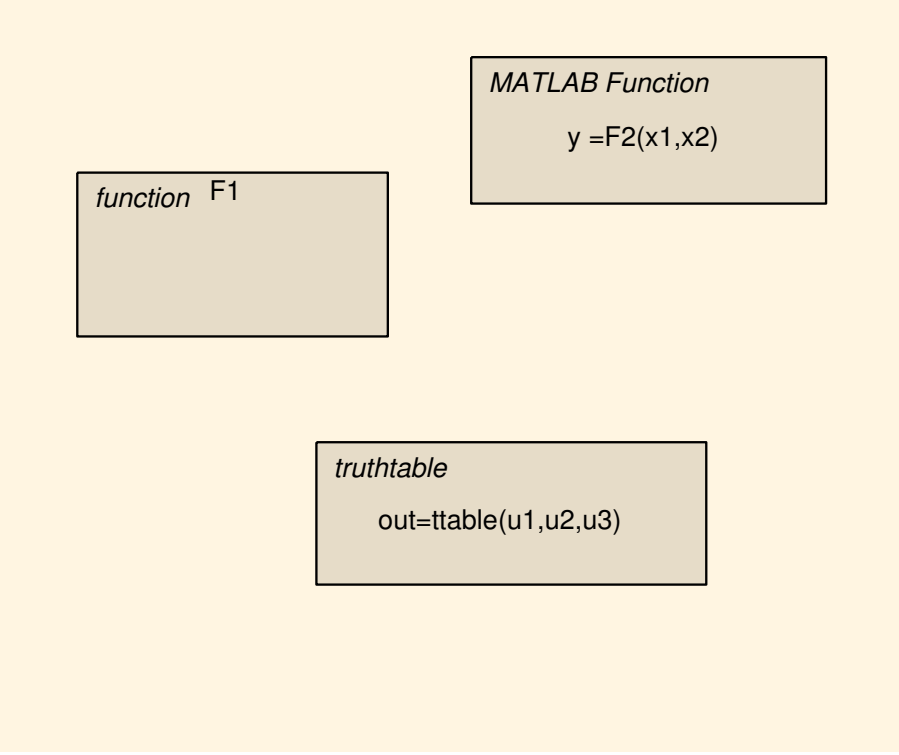

Figura 4.1: Funzioni e tabelle di verità

4.2 Matlab function

Prendiamo come esempio in stateflow una sveglia in cui possiamo impostare i secondi per un conteggio da decrementare fino a 0.
Quando impostiamo la sveglia abbiamo3 possibilità

- incremento o decremento di 100 secondi
- incremento o decremento di 10 secondi
- incremento o decrmento di 1 secondo

Quando però non ci sono sufficienti secondi da decrementare, si vuole ch questa operazione non abbia validità.
utilizziamo quindi una Matlab function che conterrà il codice seguente:

```
function retv=decval(val,fact)
if val>=fact
    retv = val-fact;
else
    retv=val;
end
```

4.3 Graphical function

In questo caso il codice viene rappresentato utilizzando il codice stateflow (vedi figura 4.2).

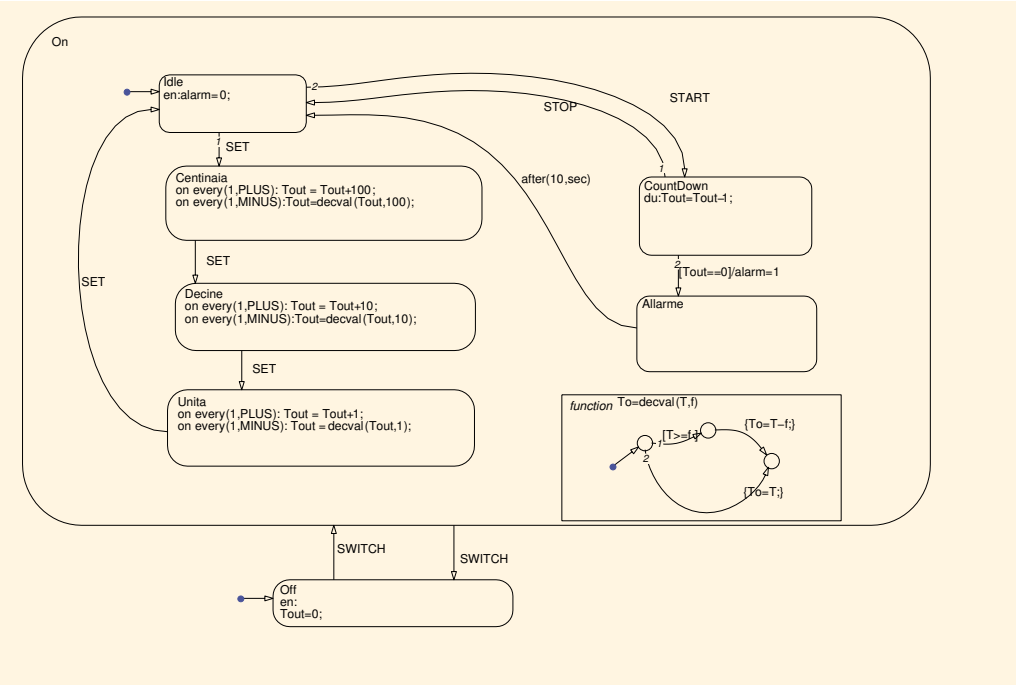

Figura 4.2: Funzione grafica

4.4 Tabelle di verità

All'interno di uno schema stateflow possono anche essere inserite tabelle di verità che possono determinare i valori in uscita in funzione di paramentri di ingresso.
Nel prossimo esempio possiamo far generare un bit di parità a seconda delle entrate.
Lo schema stateflow è rappresentato nella figura 4.3.

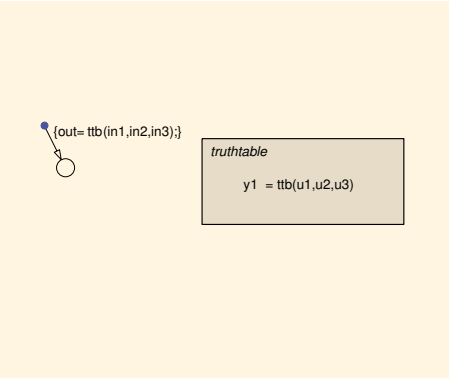

Figura 4.3: Schema stateflow con tabella di verità

Il contenuto della tabella viene fatto come da figura 4.4.

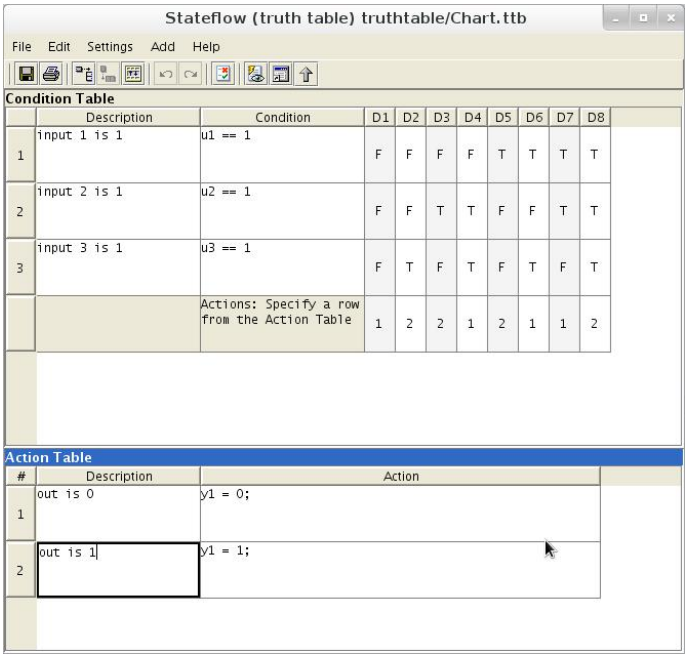

Figura 4.4: Tabella di verità

Capitolo 5

Interfacciamento con Matlab e Simulink

5.1 Interfacciamento con Simulink

L'interfacciamento con Simulink può avvenire in diversi modi:

- Utilizzo di porte in entrata e uscita alla chart come già visto negli esempi precedenti
- Utilizzare i blocchi di Data Store" di Simulink e aggiungerli alla chart nello stesso modo utilizzato per le porte di input e output.

La figura 5.1 mostra uno schema simulink che utilizza lil blocco "Data Store" associato ad una variabile "a" che viene gestita in Simulink ma utilizzata nel diagramma stateflow (vedi figura 5.2).

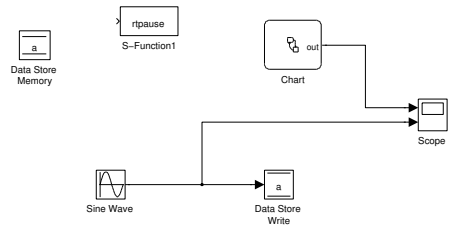

Figura 5.1: Schema simulink con "Data Store"

Il blocco "Data Store" permette anche di scambiare dati tra differenti blocchi stateflow.

5.2 Interfacciamento con Matlab

Variabili o funzioni definite nel workspace di Matlab possono essere accessibili utilizzando il formato

```
ml.x = ....
ml.y = ....
```

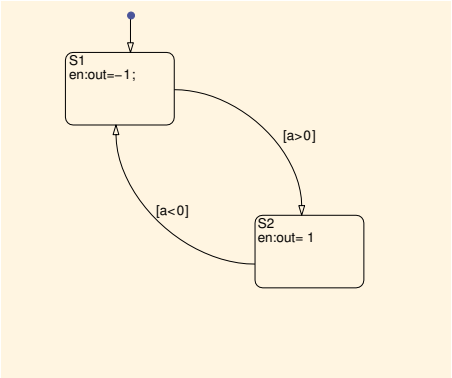

Figura 5.2: Schema stateflow con “Data Store”

```
a = ml.x + ml.sin(c);
...
```

Si può utilizzare anche la capacità di “ml” di eseguire stringhe. Ecco alcuni esempi:

```
a = ml('sin(x)')
```

Viene eseguita l'operazione “sin(x)” sulla variabile di Matlab “x” e il risultato associato alla variabile di stateflow “a”.

```
a = ml('sin(%f)', d1)
```

L'operazione “sin” di Matlab viene in quasto caso applicata alla variabile “d1” di stateflow.

Capitolo 6

Tecniche di debugging

6.1 Animazione della macchina a stati

Per visualizzare una animazione della macchina a stati occorre aprire la finestra di debug dall'editore di stateflow (figura 6.1). Si può notare il valore del campo "Delay(sec)" (1) vicin ad "Animation".

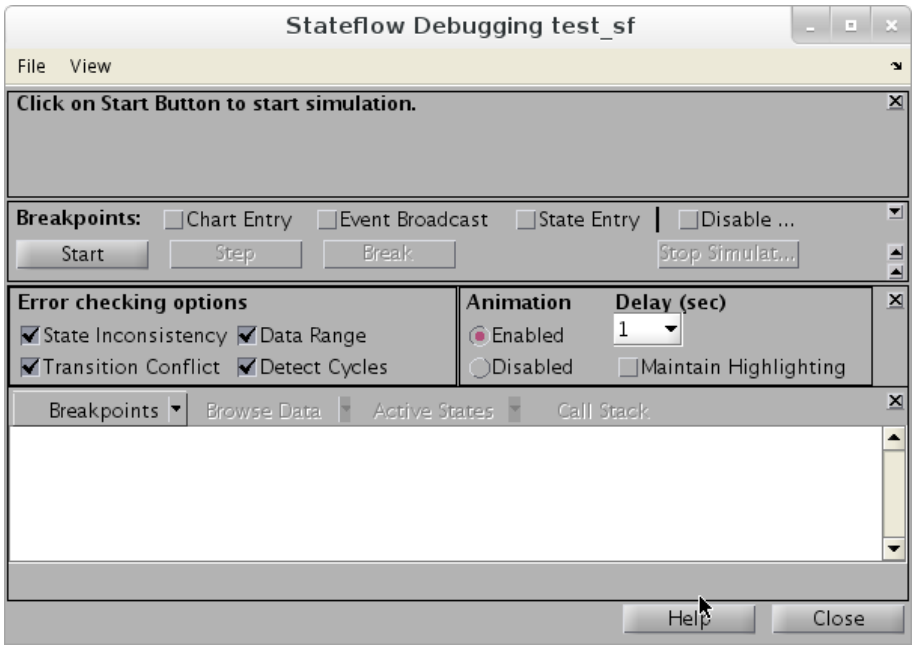

Figura 6.1: Finestra di debugging

Premendo sul bottone "Start" la simulazione permette di vedere tutti i vari cambiamenti di stato, ma viene eseguita da inizio a fine.

6.2 Breakpoint sugli stati

Possiamo inserire dei breakpoint ad ogni cambio di stato seplicemente mettendo un check nei campi "Chart Entry" e "State Entry" (figure 6.2). Evitiamo la scelta "Event broad-

cast" altrimenti ad ogni clock il sistema si ferma...

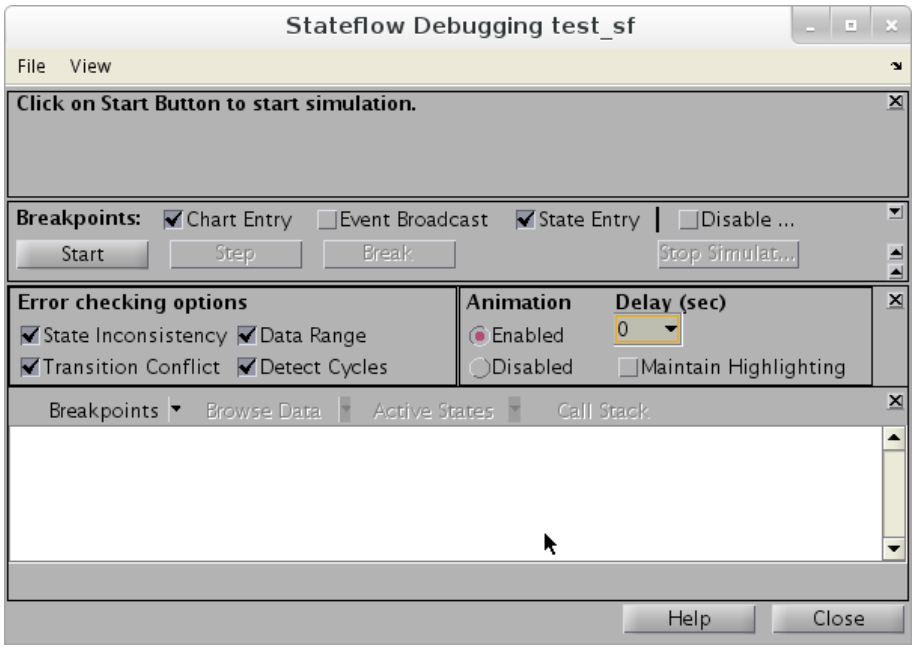

Figura 6.2: Finestra di debugging

Al primo breakpoint è possibile agire sul bottone "Browse Data" e vedere informazioni sulla simulazione (figura 6.3).

6.3 Breakpoint su inconsistenze

Per analizzare delle inconsistenze tramite un breakpoint, scegliamo un esempio portato da Mathworks. Possiamo caricarlo tramite i comandi matlab

```
>> addpath(fullfile(docroot,'toolbox','stateflow','gs','examples'))
>> Stage6Simulate
```

In questo esempio la variabile in uscita "airflow" è stata scelta limitando il valore tra "0" e "2".
Nella macchina a stati si può ad esempio modificare l'operazione associata a "during" nello stato "SpeedValue" cambiandola in

```
during: airflow = in(FAN1.On) + in(FAN2.On)+1;
```

Si può ora aprire la finestra di debugging e settare i valori come da figura 6.4.
In particolare si è scelto di mettere tra i breakpoint di "Error checking options" "Data Range".
Ora può partire la simulazione.
La simulazione si blocca al tempo 350-
Nella finestra di Matlab è possibile analizzare i vari dati, guardando ad esempio la variabile "airflow":

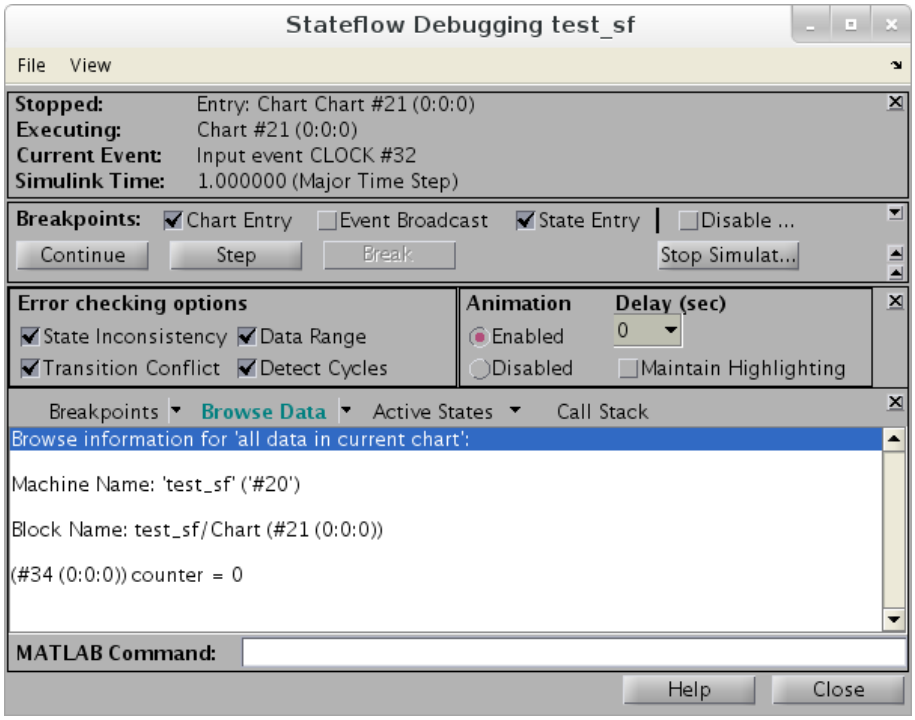

Figura 6.3: Finestra di debugging

```
Parsing failed for machine: "Stage6Simulate"(#20)


Runtime error: Data out of range
Model Name: Stage6Simulate
Block Name: Stage6Simulate/Air Controller
Data '#43 (0:0:0)': 'airflow'
Runtime error: Data out of range
Model Name: Stage6Simulate
Block Name: Stage6Simulate/Air Controller
Data '#43 (0:0:0)': 'airflow'
debug>> airflow

airflow =

    3


debug>>
```

Con un doppio click del mouse sull'ultima riga nella finestra dei breakpoint si apre la finestra delle propriteà della variabile "airflow" che è quella che ha causato l'errore.

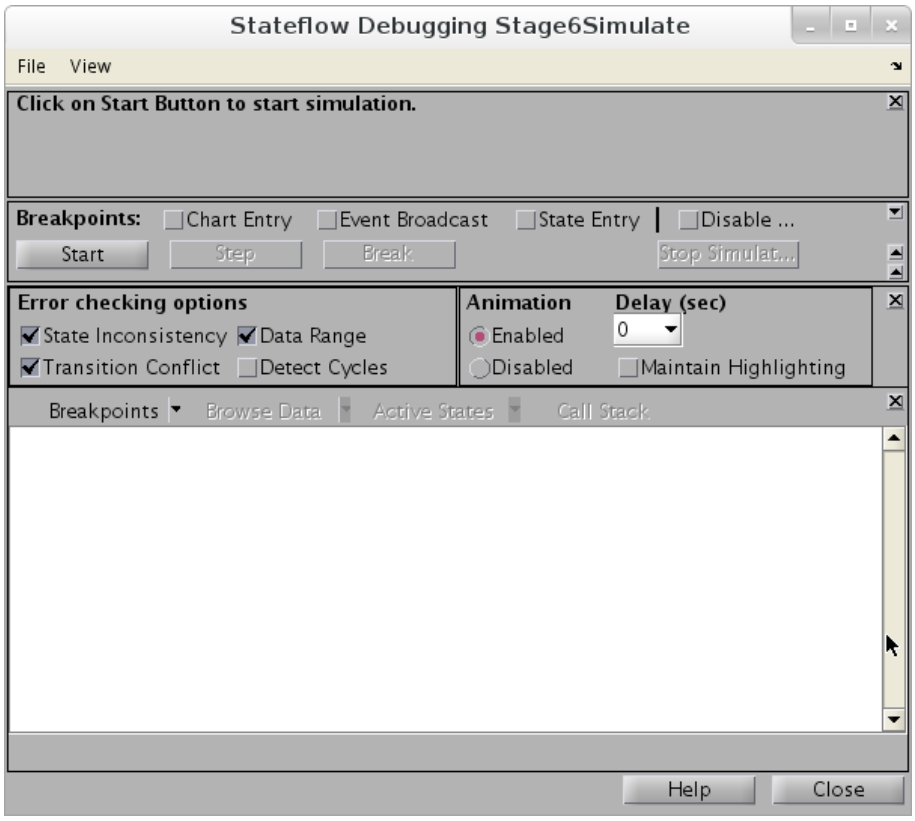

Figura 6.4: Finestra di debugging

È possibile anche scegliere nella finestra di debugging il menu “Browse Data” → “Watched Data (Current Chart)“ e vengono visualizzati i valori attuali delle variabili.

