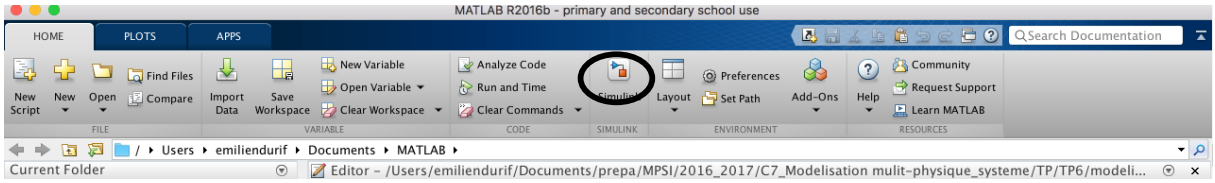
TUTORIEL MATLAB-SIMULINK-STATEFLOW

Stateflow est un outil graphique interactif intégré à **Simulink** pour modéliser et simuler des machines d'état fini, systèmes qui réagissent à des événements, dits systèmes réactifs. Ces systèmes passent d'un état à un autre en réponse à des événements et des conditions.

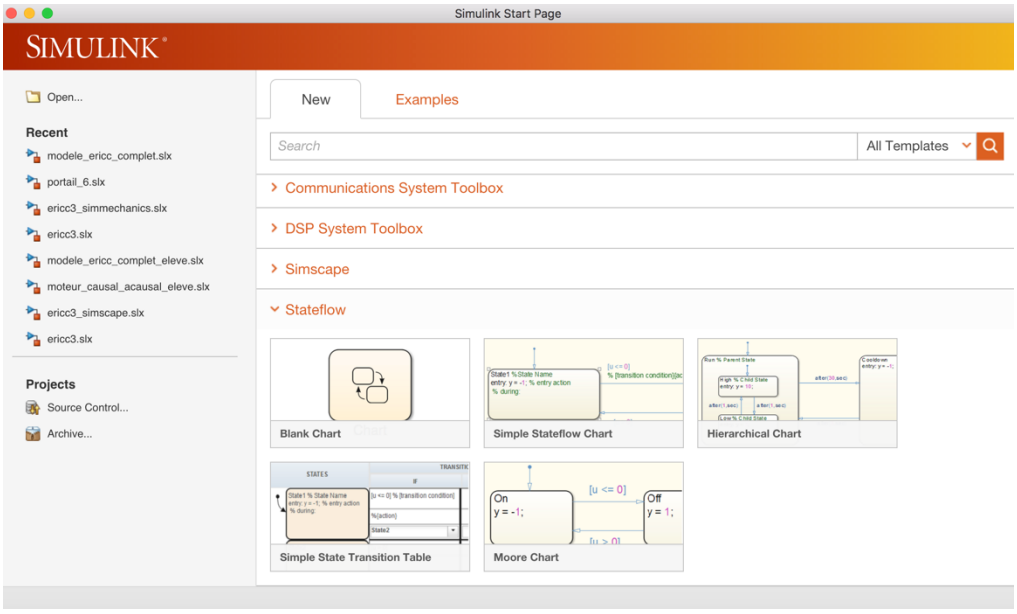
Une machine à états finis est une machine qui ne fonctionne que dans un nombre finis d'états, ou modes opératoires et permet de modéliser des processus dynamiques.

La description comportementale de tels systèmes est définie dans un diagramme d'état, faisant apparaître les différents états du système ainsi que les transitions permettant de passer d'un état à l'autre. Le logiciel **Stateflow** permet de dessiner ces diagrammes d'état.

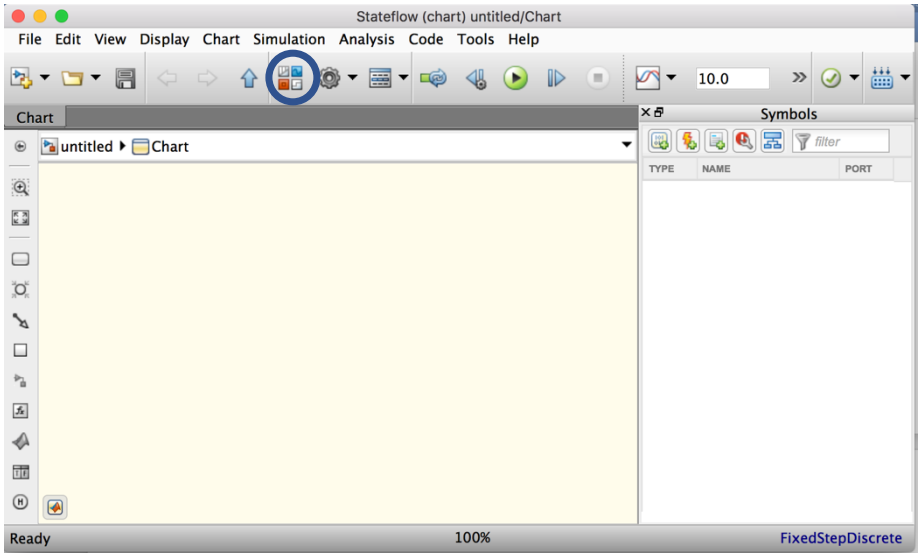
- Lancer le module Simulink



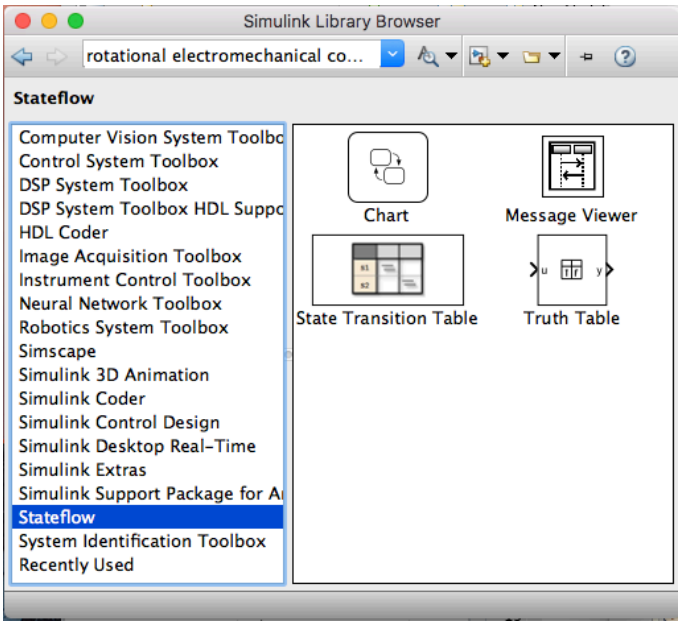
- Dans la fenêtre qui s'affiche, on accède au menu « stateflow » tout en bas. On peut alors sélectionner « Blank Chart »



- Une fenêtre apparaît alors et on peut faire apparaître la bibliothèque de fonctions



-
- On accède alors aux fonction présente dans stateflow



I. Le diagramme d'état

Dans l'arborescence des bibliothèques de **Simulink** le diagramme d'état est disponible à l'emplacement suivant :

Stateflow > Chart

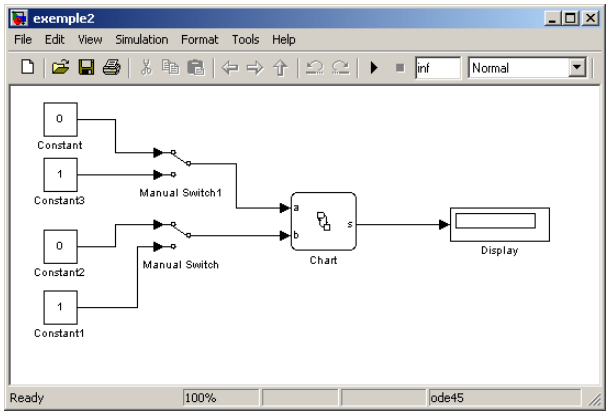
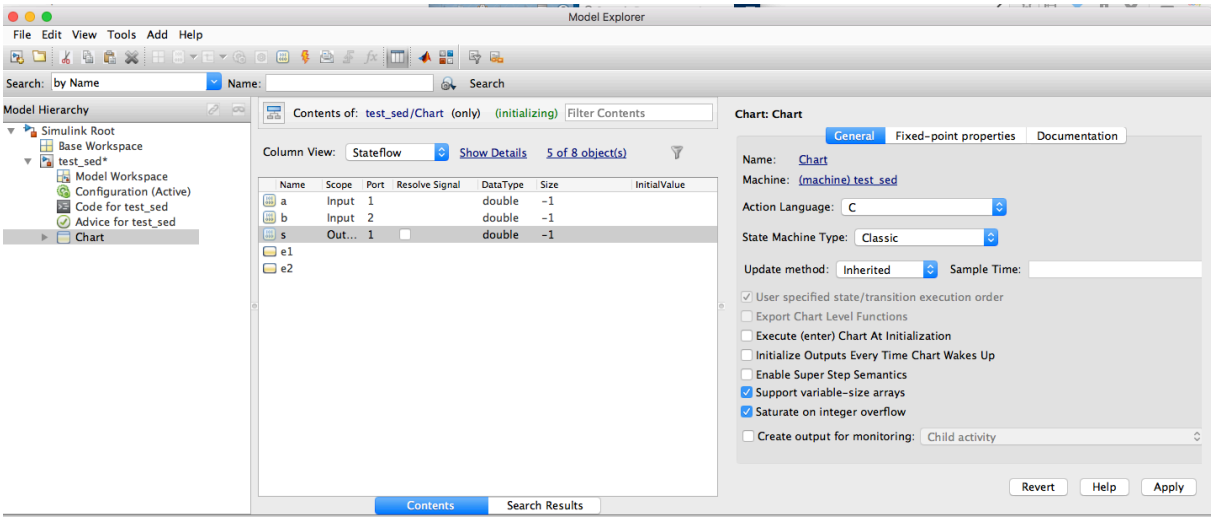


Schéma bloc dans Simulink

Configuration du diagramme d'état :

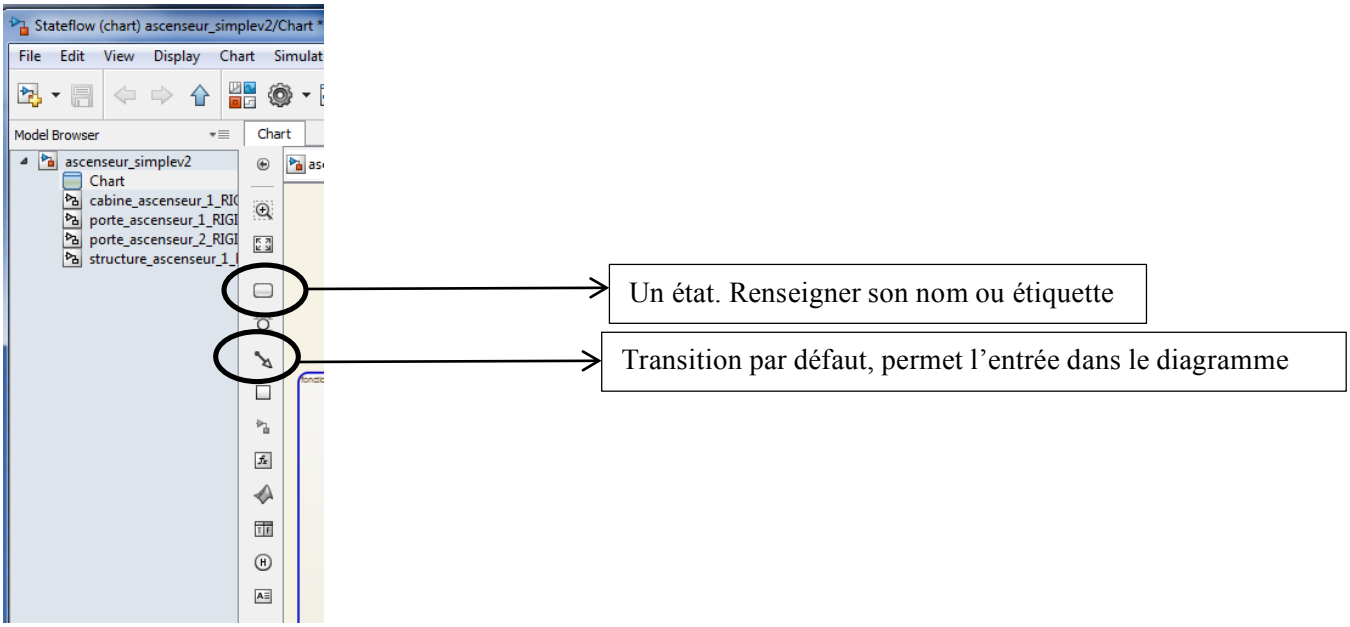
- cliquer droit sur le bloc **Chart** puis cliquer sur **Explore**
- ajouter une donnée (**Ctrl-D** ou menu **Add**+**Data**) pour chacune des entrées, les renommer, et les configurer en entrée (**Input** dans la colonne **Scope**)
- ajouter une donnée (**Ctrl-D** ou menu **Add**+**Data**) pour la sortie, la renommer et la configurer en sortie (**Output** dans la colonne **Scope**)
-  permet d'ajouter des données (icone de gauche) ou des événements (icone de droite)



Configuration du diagramme d'état dans l'explorateur de modèle

II. Les états STATE et transitions

Pour ajouter un état dans **Stateflow**, il faut ouvrir le chart en cliquant droit dessus et sélectionner « open in a new window ».



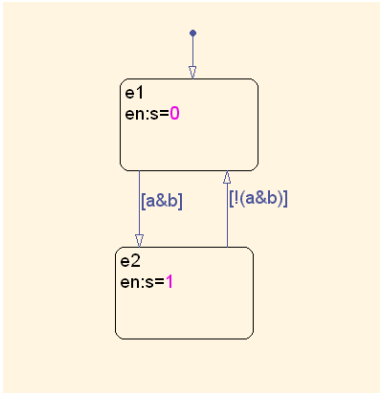


Diagramme d'état d'un porte logique ET dans Stateflow

L'état peut être OR exclusive (trait plein) ou AND parallèle trait pointillé.

L'état porte un nom ou une étiquette « state label », en haut à gauche. Les principales commandes d'actions associées aux états peuvent avoir lieu :

- A l'activation de l'état : « entry : *action* ; » ou en :
- A la désactivation de l'état : « exit : *action* ; »
- Pendant l'activité de l'état : « during : *action* ; » ou du :
- Pendant l'activité de l'état et au déclenchement de l'évènement event « on *event* : *action* ; »

Si plusieurs actions, les séparer par un point-virgule.
Les commentaires suivent le signe %.

Pour ajouter une ou plusieurs **transitions** entre deux états dans **Stateflow,** il suffit de cliquer d'un bord d'un rectangle d'état vers l'autre bord de l'état. Si on ajoute plusieurs transitions entre deux états, elles sont numérotées et elles sont traitées dans l'ordre.

Les transitions sont caractérisées par une étiquette qui décrit les circonstances ou les conditions de passage d'un état à un autre.

La transition réflexive « self loop transition » part d'un état ou pseudo état et y revient.

Les transitions sont constituées entre autre de conditions entre [], de temporisations after(10,sec)

Une transition peut contenir l'état d'un état (très utile pour synchroniser ou hiérarchiser des graphes) : [in(etat1.etat11)] signifie que la condition est vraie quand l'état11 appartenant à l'état1 est actif (état1 actif aussi).

Les opérateurs logiques utilisables dans les transitions du diagramme d'état tracé dans **Stateflow** sont les suivants :

opérateur logique	symbole dans Stateflow
ET	**&**
OU	**|**
OU-Exclusif	**^**
NON	**!**

Pour le OU logique, la commande clavier est « Alt0124 » ou Alt Gr 6

Pour le front montant « Alt24 » et le front descendant « Alt25 » (non reconnu par matlab). Il faut utiliser **rising**, **falling** ou **either**

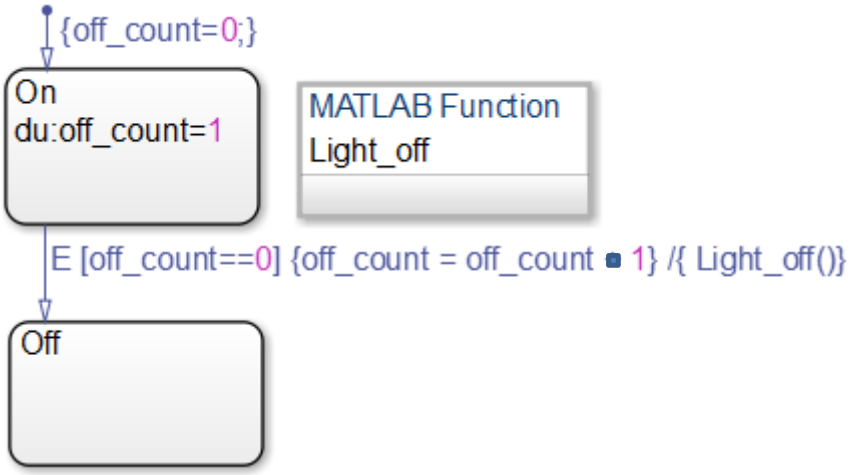
Syntaxe des transitions

event[*condition*]{*condition_action*}/*transition_action*

Il suffit de remplacer les noms de l'étiquette pour *event*, *condition*, *condition_action*, et *transition_action* avec un contenu approprié comme sur l'exemple de transition ci-dessous. Chaque partie de l'étiquette est facultative

Exemple de Transition : sur un minuteur

.



Event. Indique un événement qui produit la transition, à condition que l'état On, est vrai. Spécifier un événement est facultatif .Vous pouvez spécifier plusieurs événements en utilisant les opérateurs logiques.

Dans l'exemple précédent, le déclenchement de l'événement E produit la transition de ON à OFF aussi longtemps que la condition [off_count == 0] est vraie .

Condition. Indique une expression booléenne qui, lorsqu'elle est vraie, valide une transition à prendre pour le déclenchement d'événement spécifié. Mettre la condition entre crochets ([]).

Dans l'exemple précédent, la condition [off_count == 0] doit évaluer vraie pour le passage de l'état On à l'état Off.

Condition Action. La condition sur une transition est enfermé dans des accolades {}. Elle est exécutée dès que la condition est évaluée comme vraie et avant que la destination de la transition a été déterminée valide.

Dans l'exemple précédent, si la condition [off_count == 0] est vraie, l'état du compteur off_count est immédiatement exécuté .

Action de transition. S'exécute après que la destination de la transition a été déterminée comme valide pour autant que la **Condition**, si elle est spécifié, est vraie . Si le passage est constitué de plusieurs segments, l'action de transition n'est exécutée que lorsque l'ensemble du trajet de transition vers la destination finale est déterminée comme étant valide. Précéder l'action de transition avec un /.

Dans l'exemple précédent, si la Condition [off_count == 0] est vraie , et l'État de destination Off est valide, la fonction Light_off de transition est exécutée

.

Exemples de diagrammes d'état

Commande de feux tricolores

Le diagramme d'état contient 3 états (nommés ici **e1 e2** et **e3**) allumant chacun 1 feu et éteignant les 2 autres. Pour passer automatiquement d'un état à l'autre au bout d'un certain temps on utilise la fonction **after** :

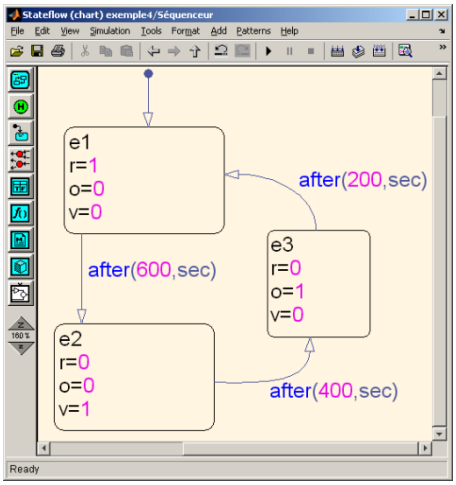


Diagramme d'état séquentiel des feux tricolores dans Stateflow

Réalisation d'un compteur

Grâce au diagramme d'état il est possible de programmer n'importe quel système séquentiel. Voyons un nouvel exemple : un compteur/décompteur.

Notre compteur doit avoir 3 entrées :

- une entrée d'horloge **H** active sur front montant
- une entrée de remise à zéro **R** active au niveau haut
- et une entrée **C** permettant de choisir le sens de comptage (si **C=1** le compteur **compte**, et si **C=0** le compteur **décompte**)

Le schéma bloc contient le diagramme d'état avec ces 3 entrées et 3 interrupteurs permettant d'agir sur ces entrées :

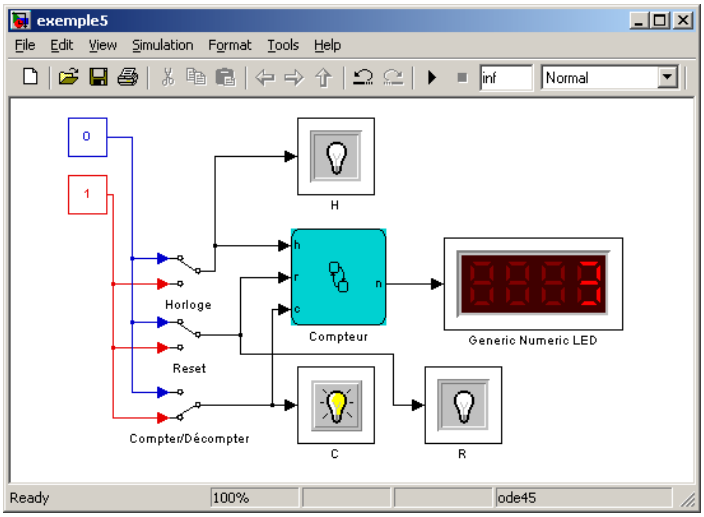
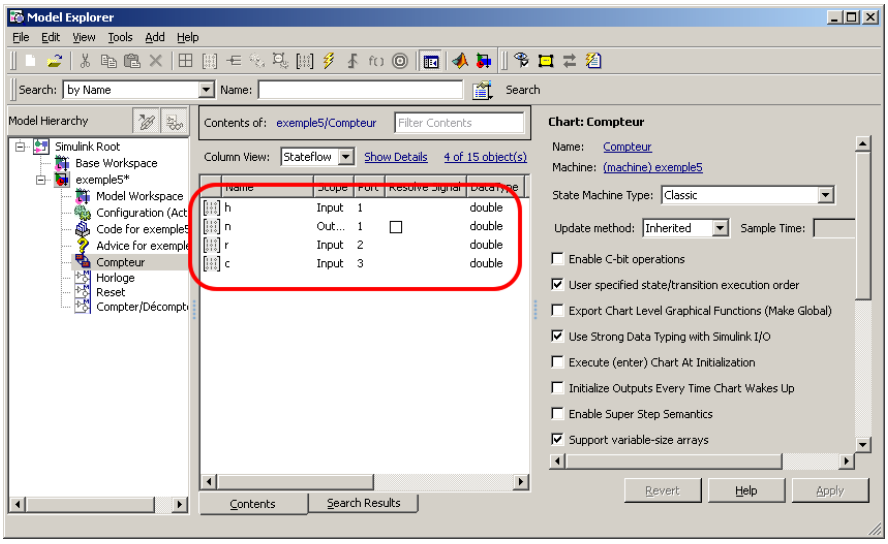


Schéma bloc du compteur/décompteur dans Simulink

l'afficheur 7 segments permet de visualiser la valeur numérique **n** en sortie du compteur.

Les ampoules permettent de visualiser facilement l'état logique des entrées.

Dans la configuration du modèle du diagramme d'état, 4 données ont été ajoutées, toutes de type double. Trois d'entre elles sont des entrées (**h**, **r** et **c**), et la quatrième est une sortie (**n**, la valeur numérique en sortie du compteur) :



Et voici le diagramme d'état décrivant le comportement du compteur en fonction de l'état logique appliqué sur les entrées :

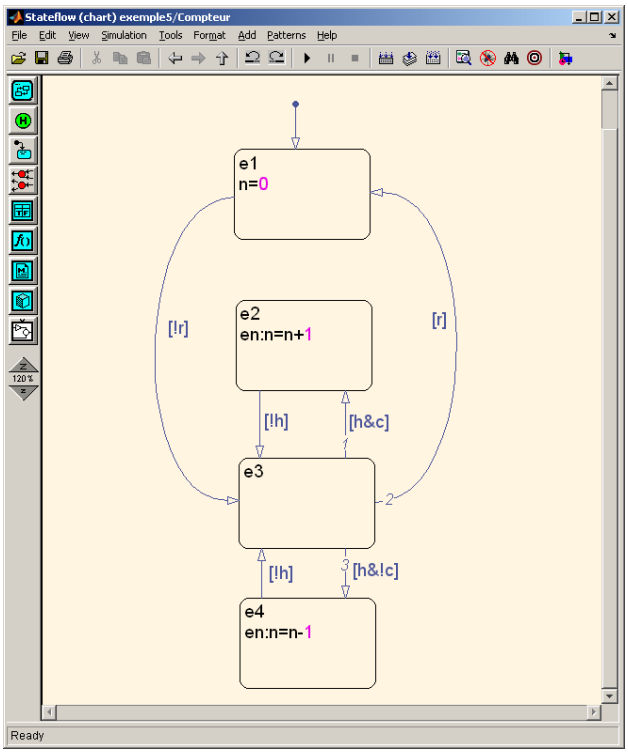


Diagramme d'état du compteur/décompteur dans Stateflow

Explications et commentaires de ce diagramme d'état du compteur :

- le diagramme contient 4 états nommés **e1**, **e2**, **e3** et **e4**
- au démarrage le programme rentre dans le diagramme par l'état **e1** qui remet le compteur à zéro

- dans l'état **e1** : si **r=0** (transition **!r**) on passe dans l'état **e3** qui est l'état d'attente ou rien ne se passe
- dans l'état **e3** le compteur ne fait qu'attendre une action sur ses entrées :
    - si **h=1** et **c=1** (transition **h&c**) on passe dans l'état **e2** pour incrémenter le compteur (action **n=n+1**)
    - si **h=1** et **c=0** (transition **h&!c**) on passe dans l'état **e4** pour décrémenter le compteur (action **n=n-1**)
    - si **r=1** (transition **r**) on passe dans l'état **e1** pour remettre le compteur à zéro
- dans les états **e2** et **e4** :
    - on ne fait évoluer le compteur qu'une seule fois à l'entrée dans l'état (grâce à **en:** qui précède l'action) et non en continu
    - dès que l'horloge **h** repasse à zéro (transition **!h**) on retourne dans l'état d'attente **e3**