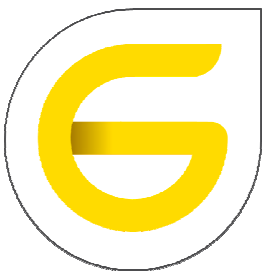


# Grafcet Designer v.2.1

## Reference Manual



May 2025



## Warranty Limitation

The media on which you receive Emmanuel Geveaux Software software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from the date of shipment, as evidenced by receipts or other documentation. Emmanuel Geveaux Software will, at this option, repair or replace software media that do not execute programming instructions if Emmanuel Geveaux Software receives notice of such defects during the warranty period. Emmanuel Geveaux Software does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. Emmanuel Geveaux Software will pay the shipping costs of returning to the owner parts which are covered by warranty.

Emmanuel Geveaux Software believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical error exists, Emmanuel Geveaux Software reserves the rights to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult Emmanuel Geveaux Software if errors are suspected. In no event shall Emmanuel Geveaux Software be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, EMMANUEL GEVEAUX SOFTWARE MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIM ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF EMMANUEL GEVEAUX SOFTWARE SHALL BE LIMITED TO THE AMOUNT THEREFORE PAID BY THE CUSTOMER. EMMANUEL GEVEAUX SOFTWARE WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCT, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVENT IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of Emmanuel Geveaux Software will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against Emmanuel Geveaux Software must be brought within one year after the cause of action accrues. Emmanuel Geveaux Software shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow Emmanuel Geveaux Software installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## Copyright

Grafcet Designer® software and its handbook reproduction are all rights reserved. Under the copyright laws, this publication and the software may not be reproduced or transmitted (except for a backup copy of the software), in any forms, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of Emmanuel Geveaux Software Corporation.

## Trademarks

LabVIEW® is a National Instruments Corporation trademark.

Macintosh® is an Apple Computer Inc trademark. Windows® is a Microsoft trademark.

Product and company names mentioned herein are trademarks or trade names of their respective companies.

## **Warning regarding use of Emmanuel Geveaux Software products**

- (1) EMMANUEL GEVEAUX SOFTWARE PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND ARE TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHICH FAILURE TO PERFORM CAN BE REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.
- (2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY. HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEM (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSE, OR ERROR ON PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENCE STEP TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM EMMANUEL GEVEAUX SOFTWARE'S TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE EMMANUEL GEVEAUX SOFTWARE PRODUCTS WITH OTHER PRODUCTS IN MANNER NOT EVALUATED OR CONTEMPLATED BY EMMANUEL GEVEAUX SOFTWARE, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF EMMANUEL GEVEAUX SOFTWARE PRODUCTS WHENEVER EMMANUEL GEVEAUX SOFTWARE PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

# Table of contents

---

<b>ABOUT THIS MANUAL.....</b>	<b>4</b>
CONVENTIONS.....	4
CONTACTS WITH CUSTOMERS.....	4
<b>CHAPTER 1. GRAFCET DESIGNER INSTALL .....</b>	<b>5</b>
MATERIAL REQUIREMENTS .....	5
SOFTWARE REQUIREMENTS .....	5
GRAFCET DESIGNER INSTALL.....	5
<b>CHAPTER 2. THE GRAFCET (OR SFC).....</b>	<b>6</b>
CONSTRUCTION RULES OF A GRAFCET .....	6
<i>Steps</i> .....	6
<i>Transitions</i> .....	6
<i>Directed arcs</i> .....	7
<i>Convergences and divergences</i> .....	7
RULES OF EVOLUTION OF A GRAFCET .....	7
<b>CHAPTER 3. USING GRAFCET DESIGNER.....</b>	<b>9</b>
PRINCIPLE OF GRAFCET DESIGNER.....	9
DEFINITION OF INPUTS/OUTPUTS .....	11
<i>Inputs</i> .....	11
<i>Outputs</i> .....	12
<i>Customize inputs/outputs</i> .....	13
EDITION OF A GRAFCET .....	14
<i>The menu « Functions Grafcet Designer »</i> .....	14
<i>Steps</i> .....	15
<i>The Conditional action editor wizard</i> .....	18
<i>The transition</i> .....	19
<i>The Conditional action editor wizard</i> .....	22
<i>Convergences and divergences</i> .....	23
<i>Arrows upwards</i> .....	23
<i>Directed arcs</i> .....	24
<i>Syntactic analysis of grafkets</i> .....	25
<i>Grafkets not related</i> .....	26
<i>Front panel of edited SFC VI</i> .....	26
CREATION OF THE APPLICATION .....	27
<i>Highlighting execution</i> .....	28
<i>LabVIEW Real Time</i> .....	28
GRAFCET DESIGNER MENUS AND WINDOWS.....	29
<i>Preferences windows</i> .....	29
<i>Examples</i> .....	30
<i>The wizards</i> .....	31
<b>APPENDIX .....</b>	<b>ERREUR ! SIGNET NON DEFINI.</b>
CONTACTS WITH CUSTOMERS.....	<b>ERREUR ! SIGNET NON DEFINI.</b>
TECHNICAL SUPPORT FORM .....	<b>ERREUR ! SIGNET NON DEFINI.</b>
DOCUMENTATION FORM.....	<b>ERREUR ! SIGNET NON DEFINI.</b>
BIBLIOGRAPHICAL REFERENCES .....	32
<i>Reference books</i> .....	32
<i>Standards</i> .....	32

# About this manual

---

The *Grafcet Designer reference manual* describes the user interface of the **Grafcet Designer** library.

To benefit fully from it, it is preferable that you are familiarised with *Windows* and with the programming language LabVIEW.

## Conventions

---

The following convention appears in this manual:

» The symbol » leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.



This icon denotes a note, which alerts you to important information.



This icon denotes a caution, which advice you of precaution to take to avoid injury, data loss, or a system crash.

**Bold**

Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names, controls and buttons on the front panel, dialog boxes, selection of dialog boxes, menu names, and palette names.

*Italic*

Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.

***Bold italic***

Bold and italic text denotes a note, an advice or a warning.

## Contacts with customers

---

You will find forms of technical support and informative at the end of this manual, in the appendix “Contact with customers”. Thanks to complete this forms and to return it indicating your comments and notes on this product and its handbook.

You can also description which you will develop with this product to obtain more information and to help you to solve possible problems which you can encounter.

# Chapter 1.

## Grafcet Designer Install

---

This part describes the requirements to use Grafcet Designer and its installation procedure.

### Material Requirements

---

Grafcet Designer, as LabVIEW, is a multi-platform product.



One version of Grafcet Designer is available on each platform that is supported by LabVIEW:

### Software requirements

---

To install Grafcet Designer, you must previously install LabVIEW® on your system. The version 2.0 of Grafcet Designer is available for each version of LabVIEW® since the version 2010.

### Grafcet Designer Install

---

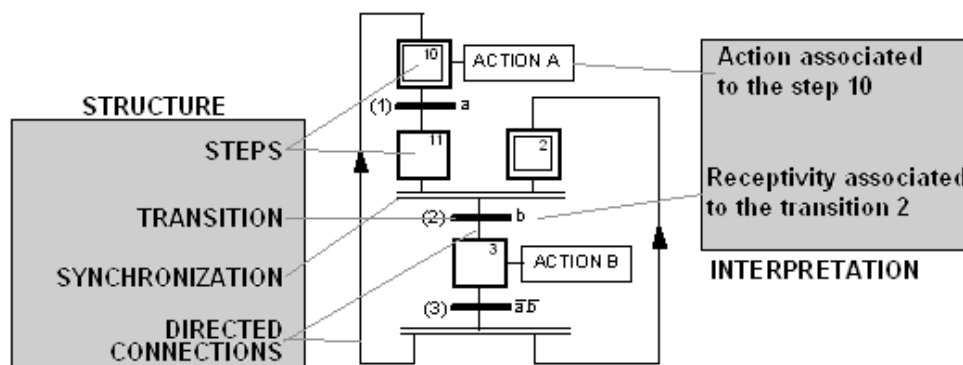
The install of Grafcet Designer is done automatically through its installer. You can download an autonomous version of this installer or a VIPM package (needs VI Package Manager).

# Chapter 2.

## The GRAFCET (or SFC)

The GRAFCET (Functional Graph of Control Step Transition) or SFC (Sequential Function Charts) is a formal model being used to specify and also to control reactive systems of the type “all or nothing” (Boolean inputs and outputs). An automated system of production (ASP) breaks up into two parts: the control part and the operative part. The operative part includes the process having to be controlled as well as the operator. The control part is intended to process the data coming from the operative part to control it.

The following presentation does not claim to be a definition of the grafcet but just an introduction. Moreover, one will reveal there only the concepts of GRAFCET which were implemented in Grafcet Designer.



Structure of a Grafcet and interpretation

## Construction rules of a grafcet

A grafcet is graph which is composed of *steps* and *transitions*, connected between them by *connections* or *directed arcs*.

### Steps



A *Step* is represented by a square to which a unique number is associated.



A *Step* can be *initial* (represented by a double square).

A *Step* is either *active* or *inactive*.

The whole of active *steps* (called *situation*) entirely defines the state of the system. We specify for each *step*, the actions to be executed. These *actions* are executed only when the corresponding *step* is active. We can associate a condition to those actions, the *action* is then executed only if the *step* is active and the *condition* is performed.

### Transitions



A *transition* is represented by a horizontal line.

A *transition* represents a possibility of change of the comportment of the system. This change of comportment (the passage from one



step to the following) corresponds to the crossing of a *transition*.

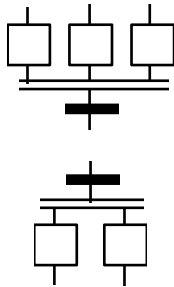
A *transition* is *validated* when all previous steps are active. The logical proposal which conditions the *transition* calls the *receptivity*.

## Directed arcs

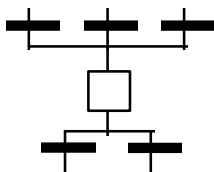


Each *directed arcs* links a step to a transition or a transition to a step: there is always strict alternation: step-transition. When this connection is directed upwards, we show the direction by an arrow upwards.

## Convergences and divergences



When the directed arcs start from several steps (known as *steps downstream*) and arrive on the same transition or when directed start from the same transition and arrive on several steps (known as *steps upstream*) then these regrouping are represented by two horizontal parallel lines respectively called convergence and divergence “in and”.



When separation is in the direction from several transitions to a common step (respectively from a step to several transitions), we name them convergences (respectively divergence) “in or”. Their representation is done by dividing the directed connections.

## Rules of evolution of a grafcet

---

The evolution of a grafcet is subject to five rules:

### Rule 1: Initial situation

The initial situation of a grafcet characterizes the initial behaviour of the control part with respect to the operative part, to the operator and/or to the external elements. It corresponds to the active steps at the beginning of operation: those steps are the *initial steps*.

### Rule 2: Crossing a transition

A transition is validated when all immediately preceding steps connected to this transition are active. The *crossing* of the transition occurs:

- when the transition is validated,
- **AND** when the receptivity associated with this transition is true.

### Rule 3: Evolution of activated steps

The *crossing* of a transition involves **simultaneously** the activation of all immediately following steps and the inactivation of all immediately preceding steps.

### Rule 4: Simultaneous evolution

Several simultaneously crossable transitions are simultaneously crossed.

### Rule 5: Simultaneous activation and inactivation

If during operation same step is simultaneously activated and inactivated, it remains active.



***Moreover, two evolution modes are generally accepted: evolution without searching for the stability or evolution with searching for the stability. Grafcet implements the last one.***

*Stability:*

For a value of the vector of inputs of the system leading to a given situation, this situation reached will be known as *stable* if after crossing of all the crossable transitions, a new situation can be obtained only on occurrence of an external event.

The outputs associated with the steps belonging to a non-stable situation are not emitted. For a given stable situation, the associated outputs which logical conditions are true are emitted with the value true, the others are emitted with the value false.



***During an evolution with search for stability, a new value of the vector of the inputs is considered only when a stable situation is reached. Consequently, a completely unstable situation (return to a same situation during the same evolution) involves a looping without end.***

# Chapter 3.

## Using Grafcet Designer

This part describes the how to create, how to execute and how to debug a grafcet with Grafcet Designer.



*The use of Grafcet Designer requires a preliminary knowledge of the LabVIEW programming language. It is pointed out here that the term VI is the abbreviation of Virtual Instrument: it indicates a LabVIEW program.*

### Principle of Grafcet Designer

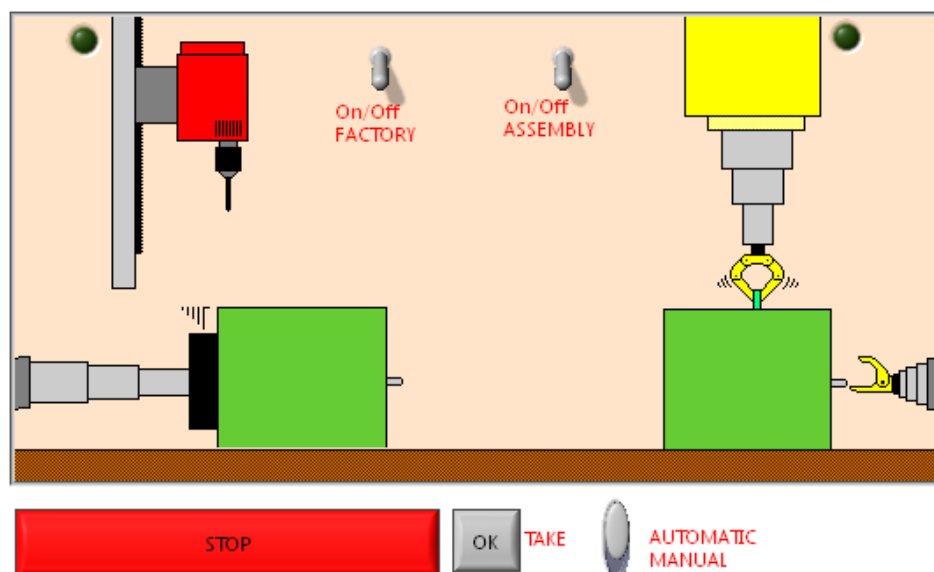
The creation of a process control application with the library Grafcet Designer is decomposed in 3 phases:

1. Definition of the inputs/outputs of the control part of the system. Those inputs/outputs can be of two types:
  - Software, by the means of the graphical interface (to or from the operator)
  - Material, by the means of physical devices (to or from the process), such as digital input/output devices, serial connections, network, ...
2. Edition of the grafcet specifying the comportment of the control part.
3. Integration of the two previous phases into an execution engine which allows executing the application.

We chose to illustrate each point on a test application, following the development of the final application step by step. The selected application is a workshop made up of a manufacture part and an assembly part. The manufacture part receives blanks which it machines. These parts are then deposited in a place of storage. Then, either on order of an operator, or in an automatic way (when the part is detected), an arm manipulator recovers the part to subject it to a machine of assembly where an assembly is carried out. The finished part is then evacuated.

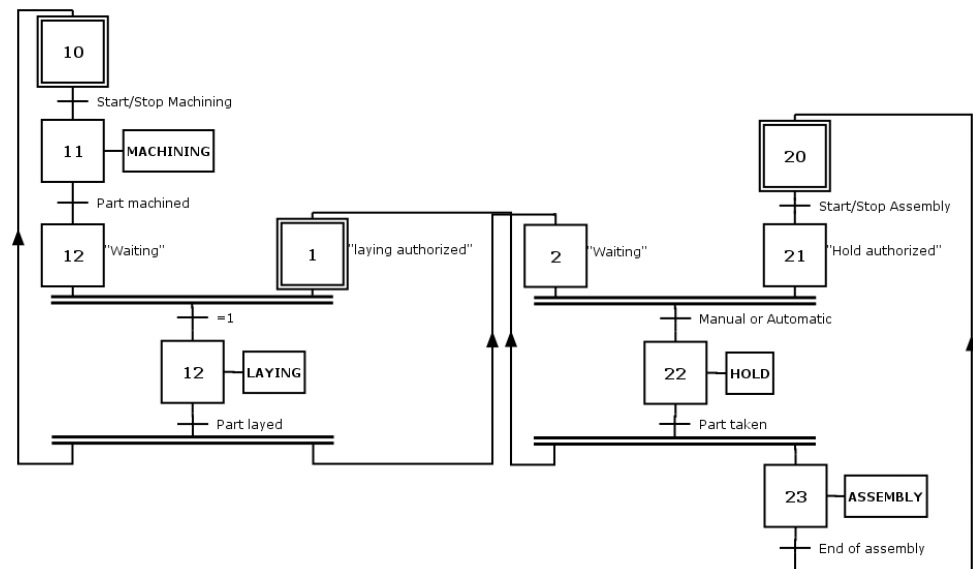


Other examples of use of Grafcet Designer are included with the toolkit. You'll find those examples under the directory .../[LabVIEW ]/examples/TecAtant/Grafcet Designer



Synoptic of the process

The grafcet specifying the behaviour of our order part is as follows :



### SFC or Grafcet of the control part

For this system, we defined the inputs of the control part with respect to the operative part, as follows :

Designation	Reference name	Source	
		Operator	Device
Start/Stop Machining	I0	✓	
Start/Stop Assembly	I1	✓	
Manual/Automatic	I2	✓	
Hold	I3	✓	
Machined part	I4		✓
Part laid	I5		✓
Part taken	I6		✓
End of assembly	I7		✓

In the same way, one defined the outputs of the control part with respect to the operative part, as follows :

Designation	Reference name	Source	
		Operator	Device
MACHINING	O0	✓	✓
WAITING 1	O1	✓	
WAITING 2	O2	✓	
LAYING	O3	✓	✓
HOLD	O4	✓	✓
ASSEMBLY	O5	✓	✓

## Definition of inputs/outputs

A grafcet specifies the behaviour of a control part, compared with an operative part. The interaction between the two parts is done thanks to the inputs/outputs.

The control part receives information from type “All or Nothing” (AoN) coming from the operative part. This information constitutes the inputs of the control part. The control part (which behaviour is specified by a grafcet) works out a whole of signals (also of AoN type) intended for the operative part, this whole of signals constitutes the outputs of the control part.

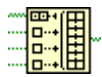
Thus, it is necessary to define these sets of boolean variables, also called respectively vector of inputs and vector of outputs. These definitions will be made in your main VI, which will contains your SFC (or grafcet).



When you open the VI ‘**Running grafcet .vit**’, a copy of this VI is automatically created. You’ll find it in....\LabVIEW\Ntemplates\Emmanuel Geveaux Software\Grafcet Designer

## Inputs

An input must be a boolean variable, it can come from two sources: the operator interface (the front panel of one VI) or of outside (for example of an acquisition device, a serial connection, or of a network... it is the interface proceeded). In the case of an acquisition device, it is often necessary to convert a numerical or alphanumeric value into boolean values.



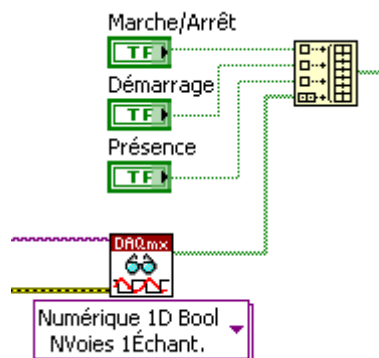
**Build Array**

In practice, to define the vector of inputs, it is just enough to assemble each one of its components using the function ‘**Build Array**’.

The order of the components is significant, because it makes it possible to index the inputs (the notation  $I_i$  will indicate the  $i^{\text{th}}$  component of the vector of inputs).

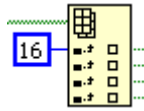
### Example

The inputs ‘**Start/Stop production line**’, ‘**Start/Stop assembly line**’, ‘**Manual/Automatic**’ and ‘**Hold**’ are inputs coming from the graphic interface (operator). The inputs can also come from an acquisition device.



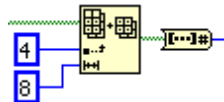
## Outputs

An output is a boolean variable, it can be redirected to two different destinations: to the operator interface (to a boolean indicator) or to the outside (acquisition card, serial connection, network,).



**Index Array**

We can read different components of a 1D vector of boolean by using the function '**Index array**' from the menu '**Array & Cluster**'.

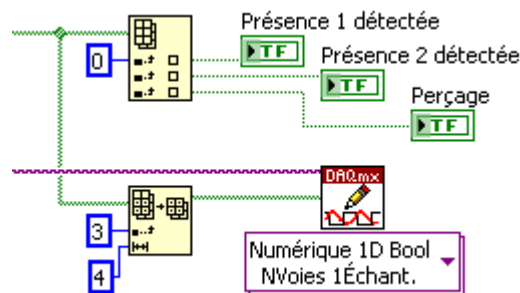


We can also use the function '**Array Subset**' from the menu '**Array & Cluster**' to read a part of a vector, for example, to be converted to an integer by using the function '**Boolean Array to number**' to be write on a port of an acquisition card.

The order in which we recover these components has an importance because it is in this order which we will make reference to the outputs in the grafcet (the notation  $O_i$  will indicate the  $i^{\text{th}}$  component of the vector of outputs).

### *Example:*

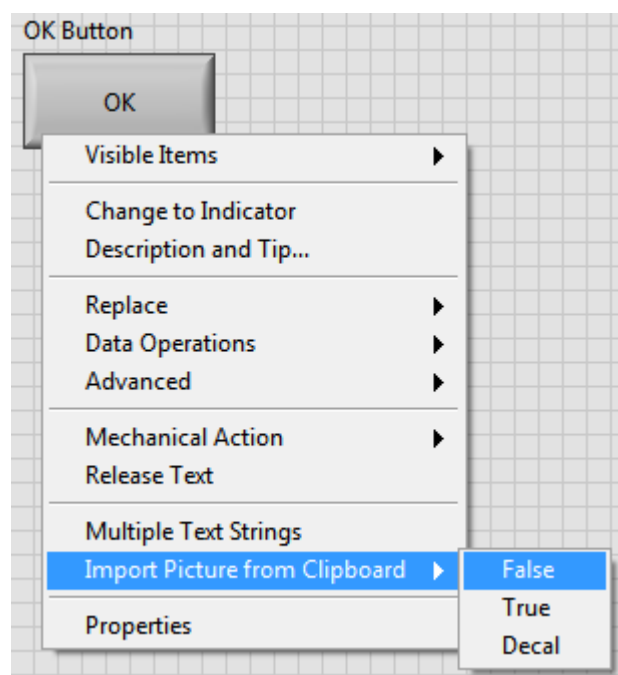
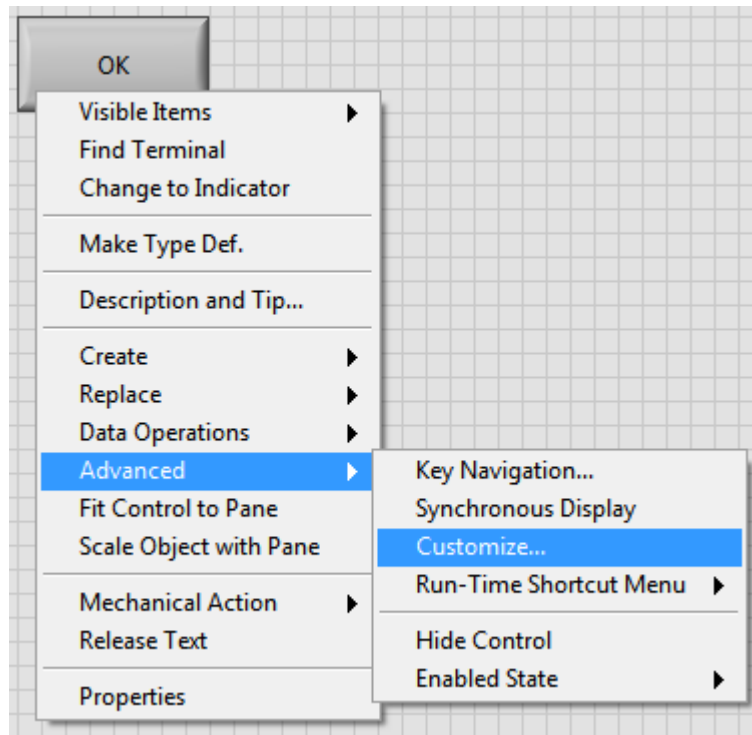
The first six outputs ('**Machining**', '**Waiting1**', '**Waiting2**', '**Laying**', '**Hold**', '**Assembly**') are software outputs (intended to inform the operator on the order given to the operative part). Only the outputs 0, 3, 4 and 5 are material outputs (they are the order given to the operative part). They are formatted ('**boolean array to number** ') and are directed to an acquisition device.



## Customize inputs/outputs

LabVIEW allows customizing controls and indicators, by using the controls editor (cf. the handbook of LabVIEW reference).

Thanks to this mechanism, it becomes possible to have a very realistic representation of the controlled process. It is enough to import the images representative of the two states to an exit.



**Example** The Boolean '**Hold**' has a representation an arm at rest in the false state and an arm taking a part in the true state. Thus the slackened and supported representation become respectively: arm at rest and arm taking a part.

## Edition of a grafcet

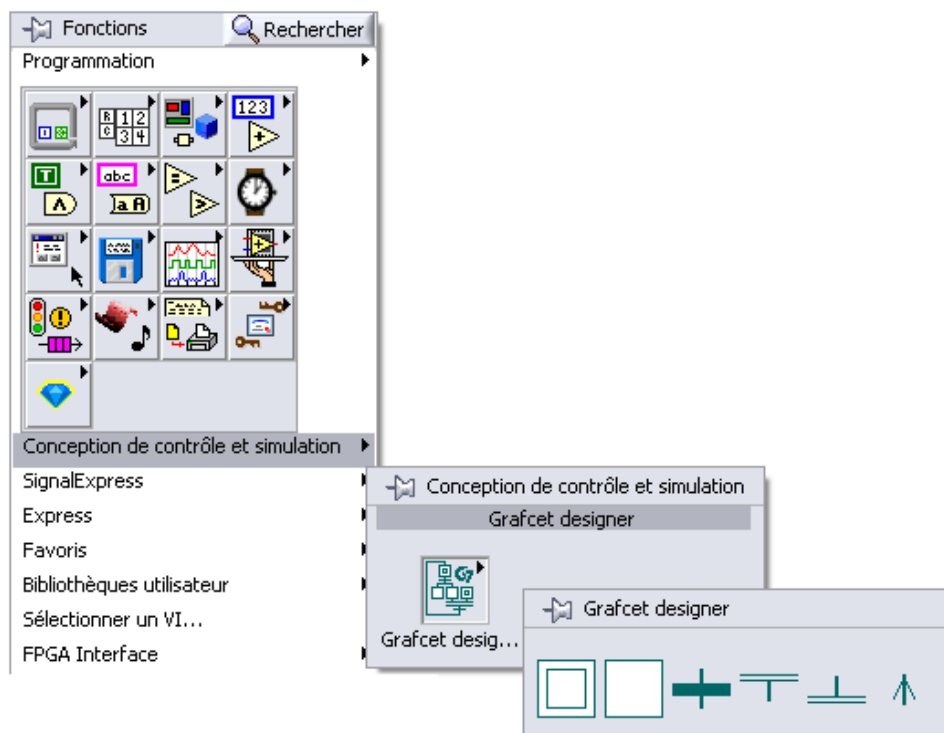
The edition of a new grafcet specifying the operation of the control part of the system which we wish to control is done in new VI (**File»New VI**).



You can also go from template VI : 'grafcet.vit' from `..\LabVIEW\Templates\Emmanuel Geveaux Software\Grafcet Designer`

## The menu « Functions Grafcet Designer »

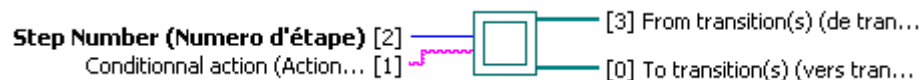
We lay out in the diagram, the VIs constituting the grafcet using the menu *Grafcet Designer* located under *Control and Simultaion* and also from *add-on* menu..



**Palette Grafcet Designer**

## Initial step

This VI allows representing the initial steps of the GRAFCET. It must be numbered (thanks to a numerical constant of the type *Unsigned Integer 32 bits*). You can also associate with it a conditional action (thanks to an alphanumeric constant of *string* type).



**Step number** specifies the number of the step. The number is essential and must be single.



**Conditional action** specifies the conditional action associated with the step. Its syntax and its semantics are clarified here after.





**From transitions upstream** is connected to the transitions upstream (which activate the step).



**To transitions downstream** is connected to the transitions downstream (which are validated by the step).

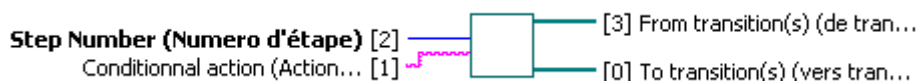


*This VI is implicitly directed from the top to the bottom.*

## Steps

### Step

This VI allows representing the steps of the GRAFCET, it must be numbered (thanks to a numerical constant of integer type 32 bits), and you can associate with it a conditional action (thanks to an alphanumeric constant of *string* type).



**Step number** specifies the number of the step. The number is essential and must be single.



**Conditional action** specifies the conditional action associated with the steps. Its syntax and its semantics are clarified here after.



**From transitions upstream** is connected to the transitions upstream (which activate the step).



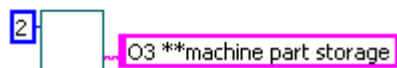
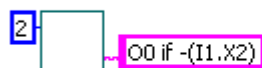
**To transitions downstream** is connected to the transitions downstream (which are validated by the step).



*This VI is implicitly directed from the top to the down.*

## Conditional actions

A conditional action breaks up into two parts, the action and the condition, separated by the reserved word "if". It is also possible to associate comments to each step of the GRAFCET.



The *action* is made up of one or several outputs, separate then by commas (,). If the action is empty then, no action is associated with the step.

The *condition* can exist only if the associated action is none empty. It is a logical expression being expressed using the operators "." (operator **and**), "+" (operator **or**) and "-" (operator **not**), of the operands "I<sub>i</sub>", "Xi" and "ti/Xj/tk", as well as brackets "(" and ")".

The *comments* are located at the end of the conditional action. They are defined by the operator "\*\*\*". When the step is active, the comment is returned on the output *comments* of VI containing your edited SFC.



The notation "**ROi**" indicates the rising of the output "**Oi**" to the true state, the notation "**FOi**" indicates the falling of the output "**Oi**" to the false state (Set and Reset).



The notation « **IOi** » indicates that the grafcet should generate a pulse on the output "**Oi**"

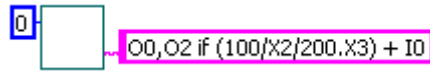
## Semantics of the conditional actions

An action associated with a step is revalued only in the stable situations of the grafcet considered (evolution with search for stability).

When the step carrying the conditional action is active then each associated output takes the true value as long as the condition is true and that the step remains active (not memorized action).

If several steps activate the same output then the value of this output is the value of disjunction between the outputs of the various steps.

**Example :**



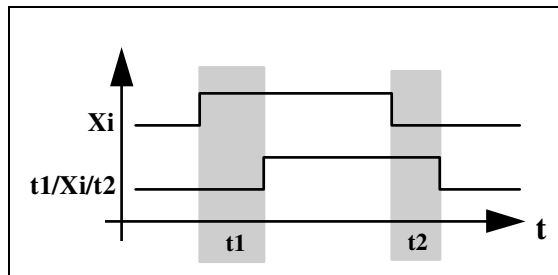
The outputs named **O0** and **O2** (index 0 and 2 in the vector of outputs) take the value true when:

- The situation reached is stable
- And the step 0 is active
- And if (the step 2 is active since more 100ms or inactive since less 200ms, and step 3 is active) or (the E0 input is true).

**Ii** has the value of the  $i^{\text{th}}$  component of the vector of inputs.

**Xi** is the variable of step associated with the step **i**, it is true if step **i** is active.

**t1/Xj/t2** is a temporization associated with the step variable **j**, where **t1** and **t2** are literal integer values. This temporization takes true value **t1** ms after the activation of the step **j** until **t2** ms after the desactivation of the step **j**.



## Complete syntax of conditional actions

Actions associated to a step can be or not associated to a condition, and a comment.

### Actions without condition

Associated action(s) emitted when steps is active.

Action expression : **Oi / ROi / FOi / IOi**

where **i** is the index of the digital output array value, that you want to set to True when the step is active.

Action can also be set, reset or impulse :

**ROi** : Rising of Output **i**. Output **i** remains to true until you reset **i**.

**FOi** : Falling of Output **i**. Ouput **i** is set to false.

**POi** : Pulse on Output **i**. Output **i** is set to true just one time. The next call to the grafcet will set Output **i** back to False.

**FXi** : freeze step **i** and it's associated tempos. Associated outputs are normally emitted if step is active.

**F0Xi** : freeze step **i** and it's associated tempos. Associated outputs are set to false.



You can also specify multiple Actions associated to a step : **Oi,Oj,(R/F/P)Ok**

**Examples of Actions :**

O0  
O1,O2  
RO1  
FO3  
PO7  
RO2,O7

**Action with condition**

Associated action(s) emitted when steps is active and condition is true.

Action expression : Oi / ROi / FOi if *condition*

where i is the index of the digital output array, that you want to be set to True when the step is active.

Action can also be set or reset.

You can't specify multiple Actions associated to a condition.

Expression of condition : any logical combination of digital inputs (Ei), Step Variable (Xi), temporisation using "and" and "or" operators. You also can use brackets. You can also use . and + for and and or operators.

Ii : Logical state of Digital input array at index i.

Xi : Step i Variable. Xi is true when Step i is active, false when inactive.

t1/Xj/t2: Temporisation. t1/Xj/t2 is true t1 ms after step j is activated and remain true t2 ms after step j is deactivated.

**Examples of conditions :**

I3 and I4	I3.I4
I5 or I7	I3+I7
I0.(I3+I5.X7)	I1+I5+1000/X7/0

**Examples of Action with conditions :**

O1 if I2.I5+I7  
RO2 if 1000/X4/0

**Comments :**

Comment expression : \*\*anything

Comments can be added to the end of an action or conditionnal action. Those comments are emitted when the associated step is active.

**Example of action with comments :**

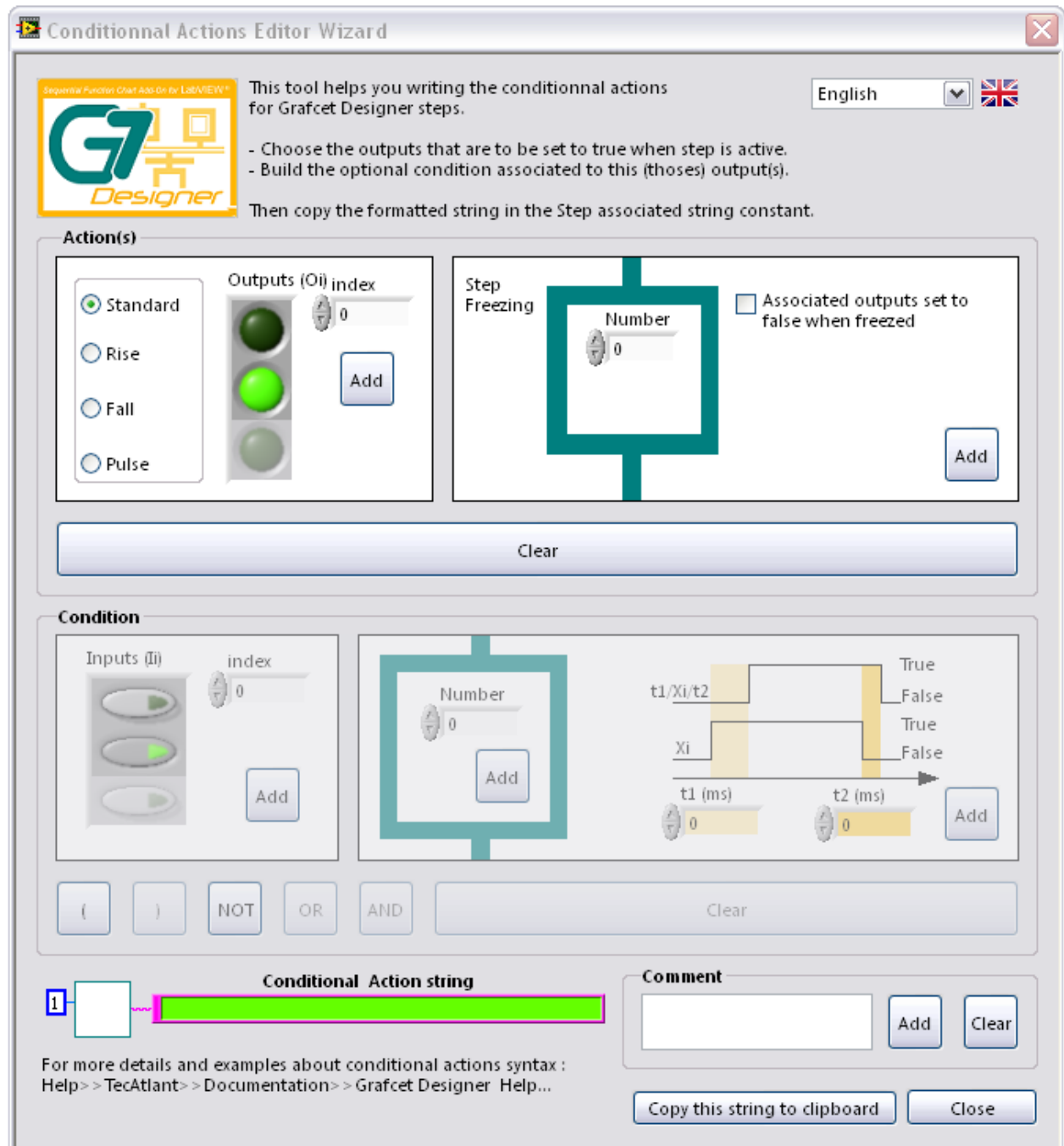
O4 \*\*Hello

## The Conditional action editor wizard

Though the menu :

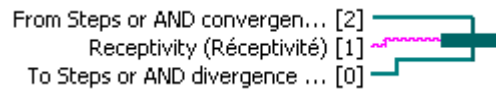
*Tools>>Control and Simulation>>Grafcet Designer>>Conditionnal Action Editor..*

You access to a wizard that helps you building conditional action. You then just have to paste the string built to step the associated string.



## The transition

This VI makes it possible to represent a *transition* of the grafcet. It is possible to associate *receptivity* to it (thanks to an alphanumeric constant of *string* type).



**From step upstream or convergence 'in and'** is connected to the step upstream which validates the transition or to a convergence 'in and ', or of nothing in the case of a transition source.



**To step downstream or divergence 'and'** is connected to the step downstream (which are activated by the transition) or to a divergence 'in and ', or of nothing in the case of a transition well.



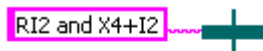
**Receptivity** specifies the receptivity associated with the transition. Its syntax and its semantics are clarified here after.



*This VI is implicitly directed from the top to the down.*

## Syntax of a receptivity

In Grafcet Designer, a **receptivity** breaks up into two parts, the event and the condition, separated by the reserved word *and*.



**The event** is either a front going up (noted "**R**"), or a front going down (noted "**F**") of the variables "**I**", "**X**", "**t1/Xj/t2**". The event can be always occurrent, in this case it is noted **e** (or not noted). Thus an event is written "**RI**", "**FI**", "**RXi**", "**FXi**", "**Rt1/Xj/t2**", "**Ft1/Xj/t2**", "**e**" or anything (equivalent to **e**).

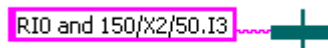
**The event** can also be an order (noted "%order"), sent to the grafcet by the *Order* input of the VI that contains your edited SFC.



*in the case of an order, the event can not be followed by a condition.*

## Semantics of the receptivities

A transition will be crossed if it is validated (All its *steps upstream* are active), if the associated event arrived (always in the case of **e**) and if the associated condition is true.



This transition will be crossed when:

- All its *steps upstream* are active
- *and* the **I0** input passes from the false state to the true state (front going up of **I0**)
- *and* (step 2 is active since at least 150ms or inactive since less 50ms) *and* (the **I3** input is true).

## Complete syntax of Transitions Receptivity

### Empty Receptivity

Transition is immediatly fired when up-link steps are actives.

### Receptivity without condition :

Transition is fired when up-link steps are actives and when receptivity is true.

Expression of receptivity :  $I_i / RI_i / FI_i / X_i / t1/Xj/t2$

$I_i$  : Logical state of Digital input array at index i.

$RI_i$  : Rising edge of Digital input array at index i.

$FI_i$  : Falling edge of Digital input array at index i.

$X_i$  : Step i Variable.  $X_i$  is true when Step i is active, false when inactive.

$t1/Xj/t2$ : Temporisation.  $t1/Xj/t2$  is true 1000 ms after step j is activated and remain true 500 ms after step j is deactivated.

#### *Examples of Receptivity :*

$I0$

$RI3$

$FI4$

$1000/X2/500$

### Receptivity with condition

Transition is fired when up-link steps are actives and when receptivity is true and condition is true.

Expression of receptivity :  $RI_i / FI_i$  and condition

$RI_i$  : Rising edge of Digital input array at index i.

$FI_i$  : Falling edge of Digital input array at index i.

Expression of condition : any logical combination of digital inputs ( $E_i$ ), Step Variable ( $X_i$ ), temporisation using "and" and "or" operators. You also can use brackets. You can also use . and + for and and or operators.

$I_i$  : Logical state of Digital input array at index i.

$RI_i$  : Rising edge of Digital input array at index i.

$FI_i$  : Falling edge of Digital input array at index i.

$X_i$  : Step i Variable.  $X_i$  is true when Step i is active, false when inactive.

$t1/Xj/t2$ : Temporisation.  $t1/Xj/t2$  is true 1000 ms after step j is activated and remain true 500 ms after step j is deactivated.

#### *Examples of conditions :*

$I3$  and  $I4$

$I3.I4$

I5 or I7

I3.I7

I0.(I3+I5.X7)

I1+I5+1000/X7/0

***Examples of Receptivity with conditions :***

RI11 and I2.I5+I7

FI2 and 1000/X4/0

**Receptivity is an Event**

Transition is fired when up-link steps are actives and when the string associated to event is sent to the grafcet VI

Expression of receptivity : %anything

***Examples of events***

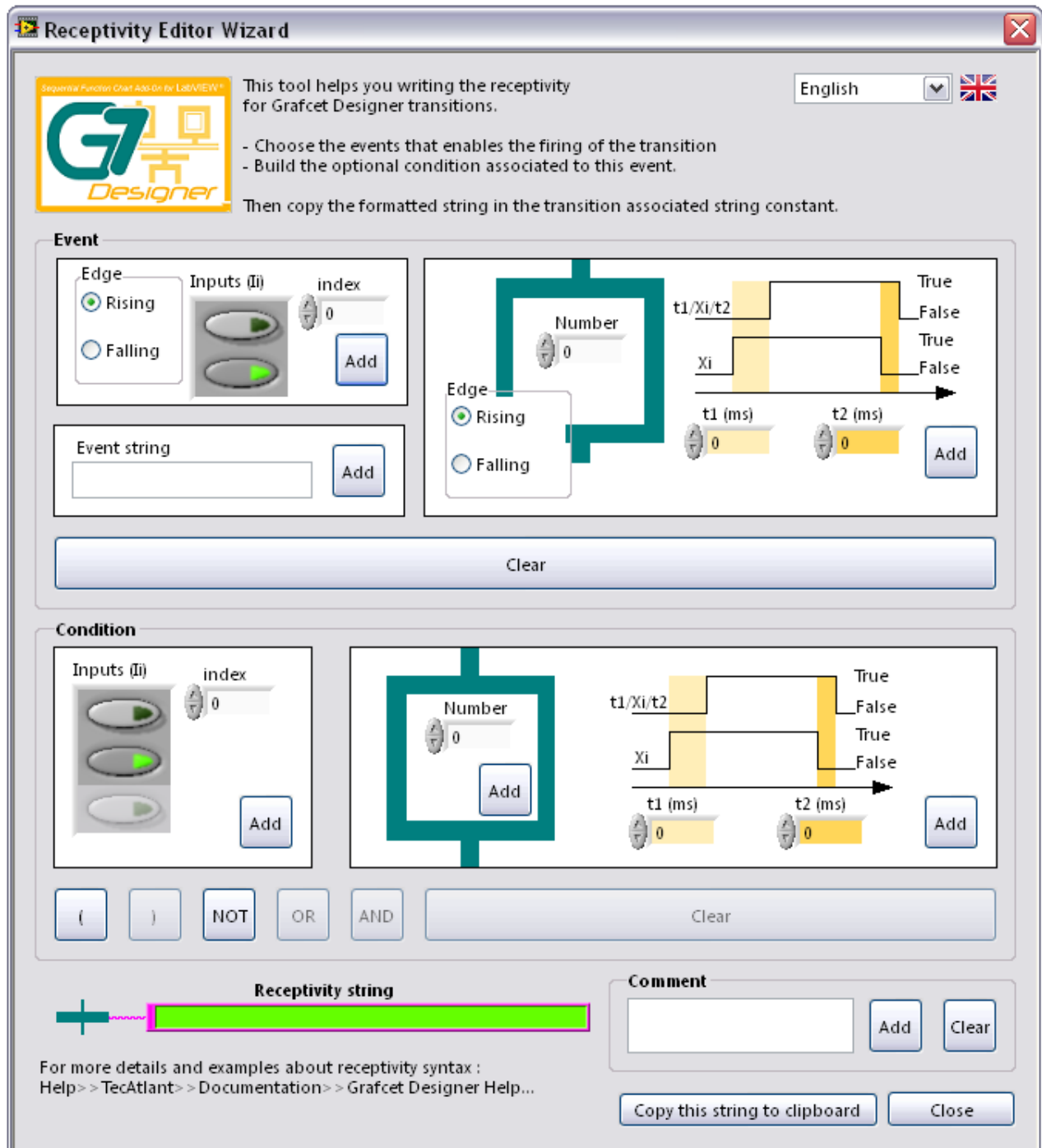
%Go !

## The Conditional action editor wizard

Though the menu :

*Tools>>Control and Simulation>>Grafcet Designer>>Receptivity Editor..*

You access to a wizard that helps you building receptivity. You then just have to paste the string built to transition the associated string.

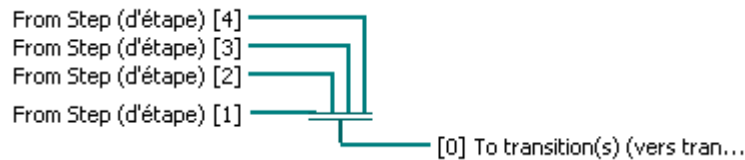




## Convergences and divergences

### Convergence 'in and'

This VI makes it possible to represent convergence '**in and**' of the GRAFCET. We can connect to it up to four steps upstream. It is possible to put several specimens of these VI in cascade if we wish to connect more than four steps.



**From step upstream** is connected to a step upstream (which validates the transition).



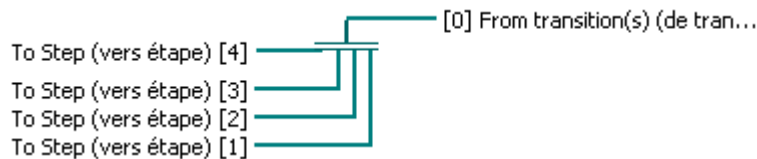
**To Transition Downstream** is connected to the transition downstream (which is validated by the stages).



*This VI is implicitly directed from the top to the bottom.*

### Divergence 'in and'

This VI allows representing the divergence '**in and**' of the GRAFCET. We can connect to it up to four steps downstream. It is possible to put several specimens of these VI in cascade if we wish to connect more than four steps.



**To step downstream** is connected to a step downstream (which is activated by the transition).



**From transition upstream** is connected to the transition upstream (which activates the steps).



*This VI is implicitly directed from the top to the down.*



Convergences and divergences '**in or**' do not have a representation by VI (they are done directly thanks to wiring by the LabVIEW wiring tool).

## Arrows upwards

This VI allows displaying rising arrows on the arcs. The VIs *step*, *initial step*, *transition*, *convergence* and *divergence 'in and'* being implicitly directed from the top to the bottom. The use of this VI is not obligatory, but makes it possible to make appear in an explicit way the implicit orientation of the rising arcs.



**To step upstream** is connected to a step upstream (activated by the transition) or to a **divergence 'in and'**.



**From transition downstream** is connected to the transitions downstream (which activate the step).



*The low connection of this VI must go to the low connection of a transition and high connection must come from the high connection of a step or a divergence 'in and'.*

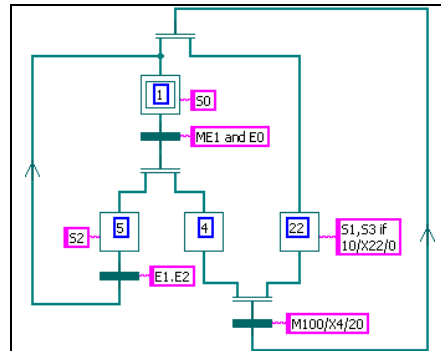
## Directed arcs



Once the objects (*steps, transitions, etc...*) placed in the diagram, it does not remain any more that to connect them between them using the tool winds: it is this link which constitutes the directed arcs between the various entities. The Grafcet Designer entities being implicitly directed from the top to the bottom, the arcs are thus directed from the entity upstream to the entity downstream. In order to return clarifies the rising arcs, we can insert one VI '**arrow upwards**' (this VI will receive obligatorily its input from a step or a divergence '**in and**' and will emit its outputs to one or more transitions).

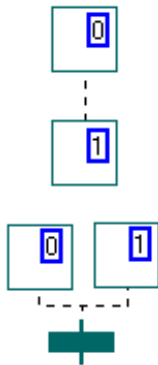


*Grafcet Designer does not authorize to place entities other than the VI 'arrow upwards' on a link going up (under penalty of having a broken bond).*



## Syntactic analysis of grafcets

### Graphic syntactic analysis



Grafcet Designer benefits from the properties of the graphic editor of LabVIEW, thus thanks to the typing of the arcs between entities, the correction of the graphic syntax of a grafcet is immediate: if an arc is into dotted then one of the rules of constructions of the GRAFCET is not respected (not respect of alternation step-transition, several steps are connected directly to the same transition, etc...)

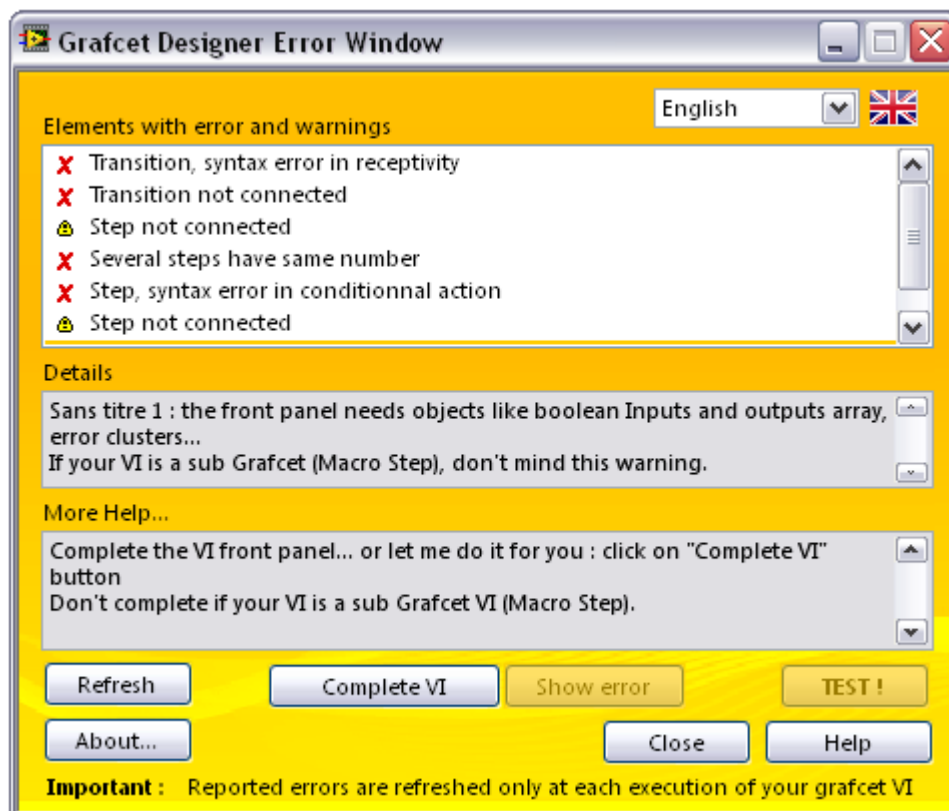
The appearance of the broken arrow (VI no achievable) indicates a syntactic error of the grafcet in the course of edition.

### Analyze conditional actions and receptivities

If the graphic syntax of the grafcet is correct, it also should be checked that several steps do not have the same number, that there is no transition not connected to a step, that the conditional actions and the receptivities are correct.

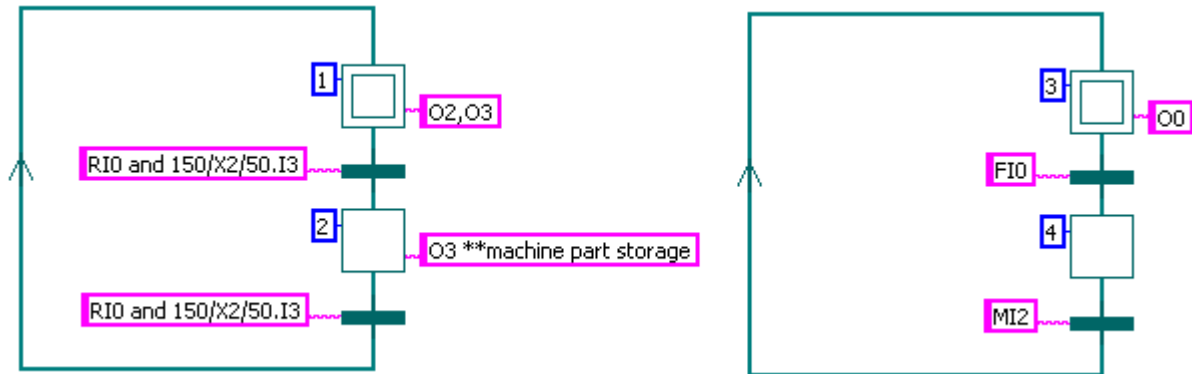


This analysis is realized by executing the VI which contains the published grafcet. If an error occurs in the grafcet, a VI 'window of error report' displays the list of errors.



## Grafcets not related

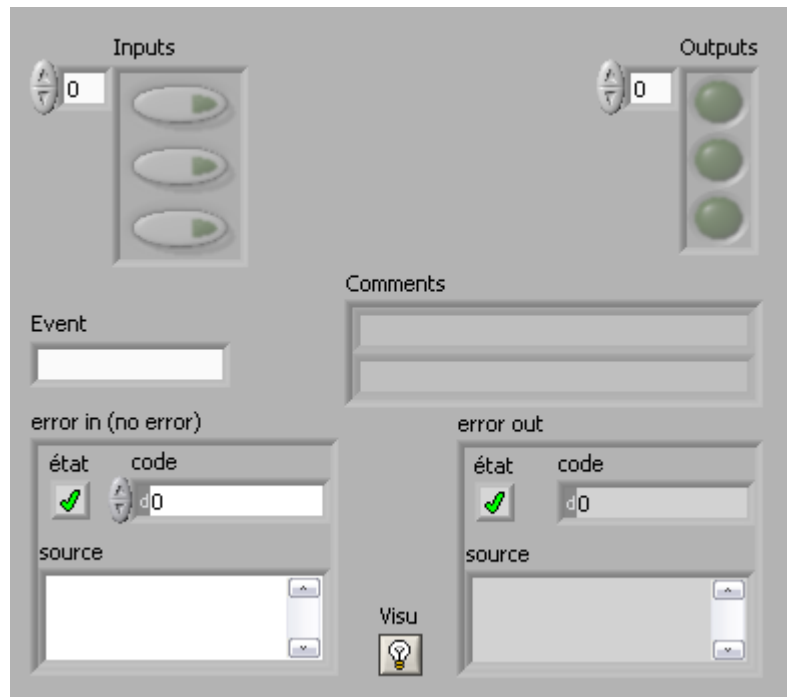
Nothing prohibits the creation of not related grafcets (without directed arcs to connect them). In particular small separated grafcets can possibly be used as memorizing... They can quite naturally be referred by step variable.



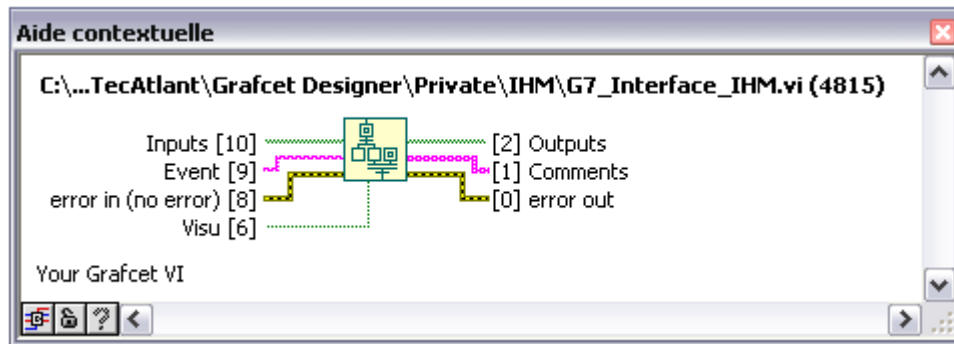
Example of not related SFCs

## Front panel of edited SFC VI

In order to be able to call your edited SFC VI from another VI, you need to put the following controls and indicators :



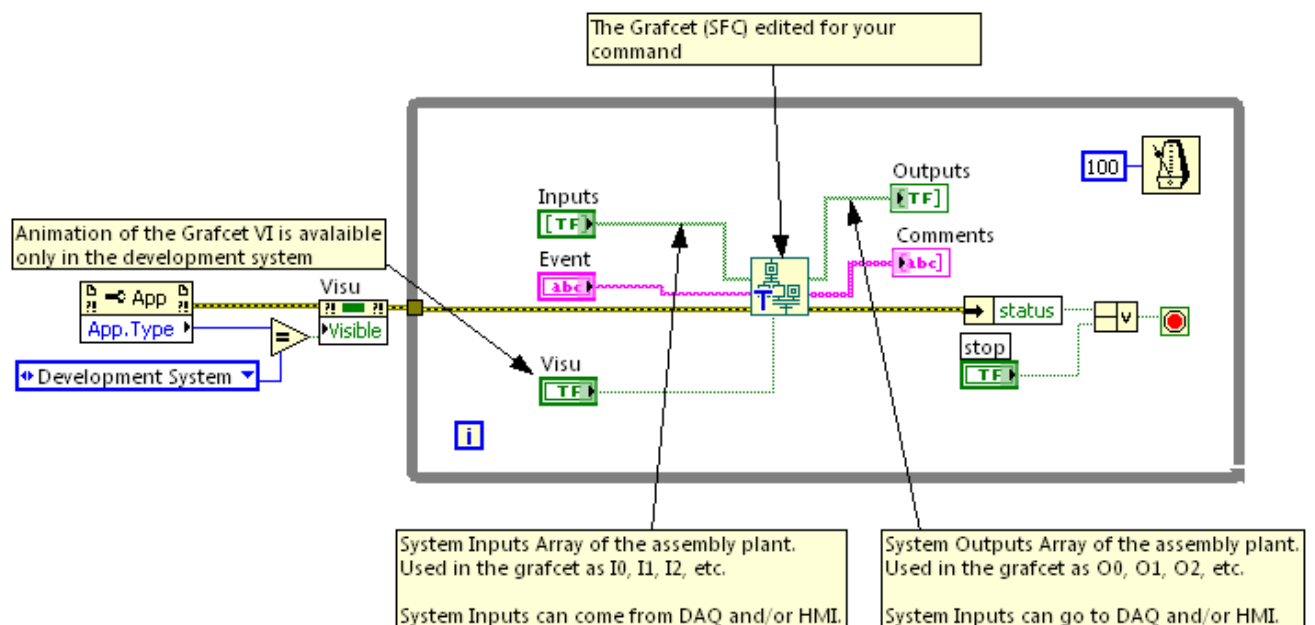
In order to be able to pass values to your edited SFC VI from another VI, you need to create associated connector to those objects (cf. LabVIEW reference manual).



If you forget to do this, then at run time, the error window will show up and signals you this forget. With the button **complete**, LabVIEW will automatically create needed front panel objects, and also the connector pane and an icon to your VI.

## Creation of the application

When the inputs/outputs of our control part were defined and that the grafcet specifying its behaviour was published, it remains to integrate these various elements in the final VI, one which will make live to our application.

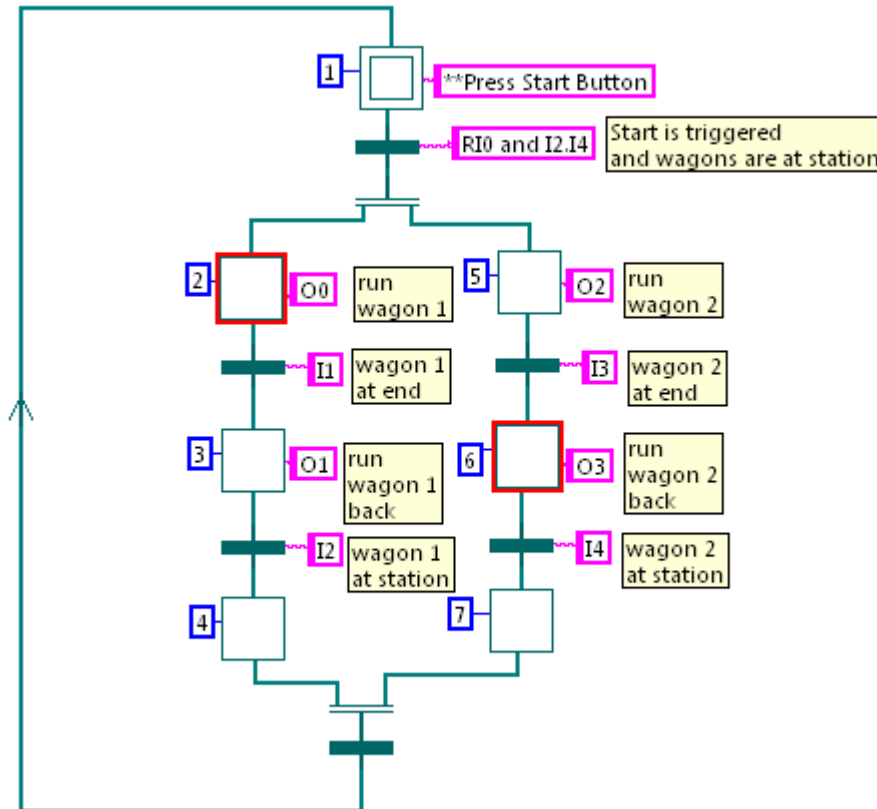


## Highlighting execution

Passing True value to input **Visu** of your SFC VI, let you visualize active steps at run time.



Execution highlighting works only under development environment (IDE).



Highlighting execution mode example

## LabVIEW Real Time

Grafset designer is LabVIEW Real Time compatible (**NI LabVIEW Real-Time Module**) and work on device like Compact RIO (**NI CompactRIO embedded control systems**).



You must validate your edited SFC under your LabVIEW for Windows IDE. Once your SFC is syntactically correct and verified then you can run it under labVIEW Real-Time.



You can't visualize active steps under Real Time environment.

## Grafcet Designer menus and windows

### Preferences windows

Through the menu *Help>>Emmanuel Geveaux Software>>Preferences>>Grafcet Designer preferences...* You can :

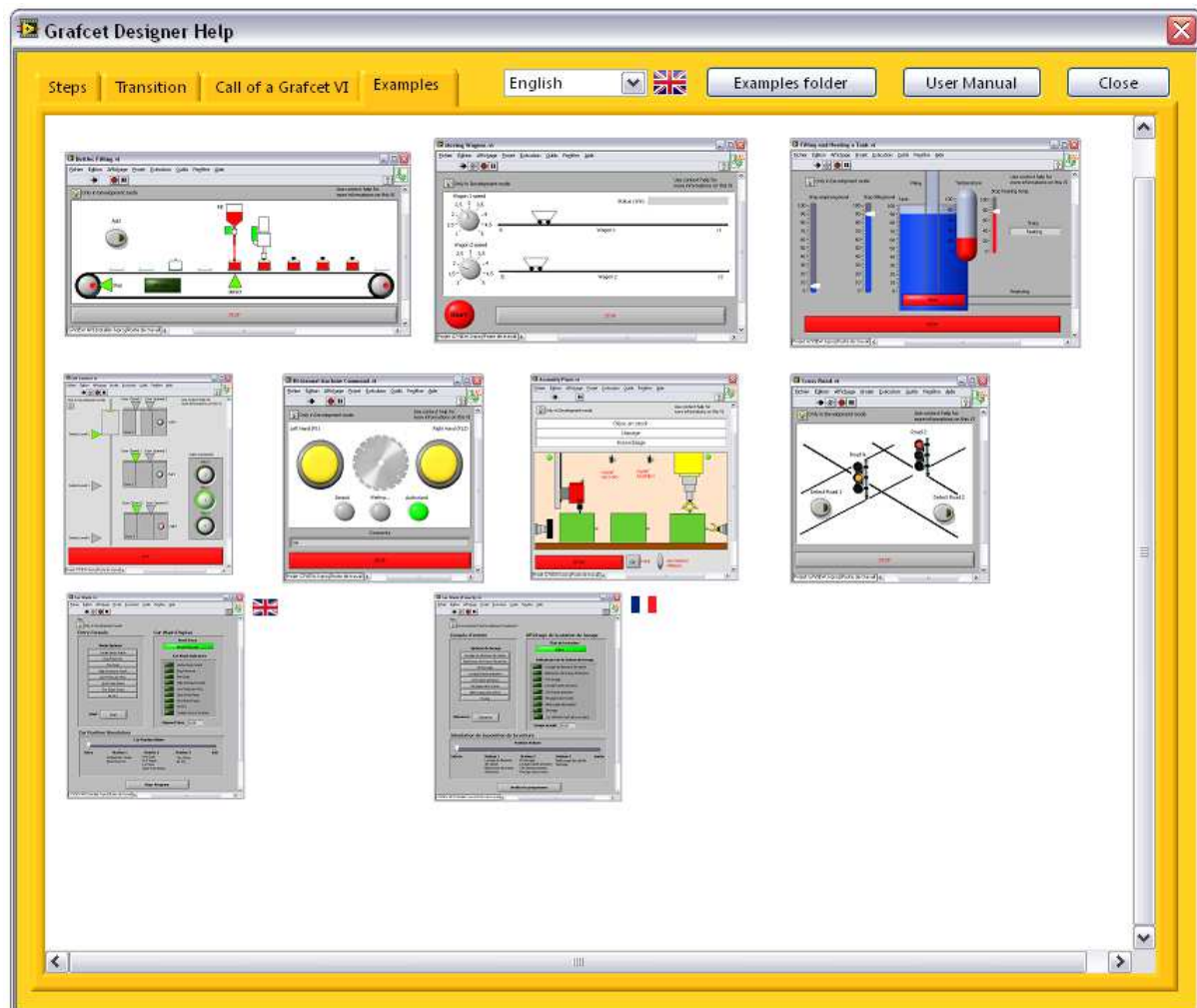
- Change Grafcet Designer Language : English or French,
- Desactivate automatic routing of wires (recommanded for a better experiment of Grafcet Designer)
- Activate Block Diagram Grid (recommanded for a better experiment of Grafcet Designer)
- Choose the palette type of Grafcet Designer : Classic or pre-wired (step number, conditional action string, receptivity string)



## Examples

You can access to examples Vis of Grafcet Designer through the NI Examples Finder tool (Menu *help*>>*find examples...*) or through the help window of Grafcet designer ( menu *help*>>*Emmanuel Geveaux Software*>>*Documentation*>>*Grafcet Designer Help...* )

Exemples are an efficient way to understand and use Grafcet designer.



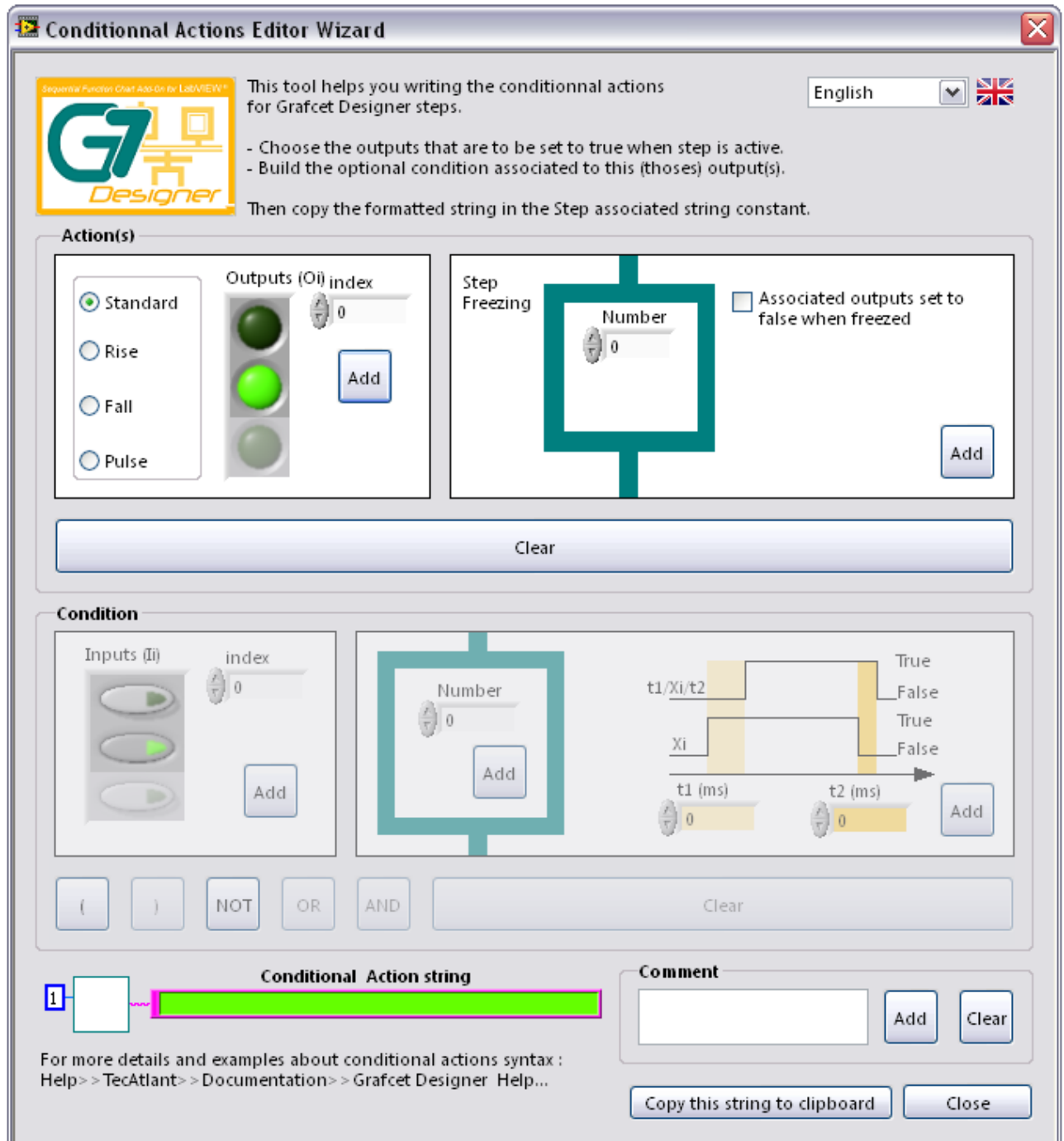


## The wizards

Two wizards help you in expression of steps conditional actions and transitions receptivities. You'll find them under menus :

*Tools >> Control and Simulation >> Grafcet Designer >> Conditional Action Editor*

*Tools >> Control and Simulation >> Grafcet Designer >> Receptivity Editor*



# Bibliographical References

---

## Reference books

*Comprendre, maîtriser et appliquer le GRAFCET*, M. Blanchard, CEPADUES-EDITIONS, Collection NABLA.

*LE GRAFCET*, N. Bouteille, P. Brard, G. Colombari, N. Cotaina, D. Richet, CEPADUES-EDITIONS.

*Du GRAFCET aux réseaux de Petri*, R. David, H. Alla, HERMES, Traité des Nouvelles Technologies, Série Automatique.

## Standards

Preparation of function charts for control systems. International Standard, CEI/IEC 848, December 1988, CEI - 3 rue Varembe Genève - Suisse.