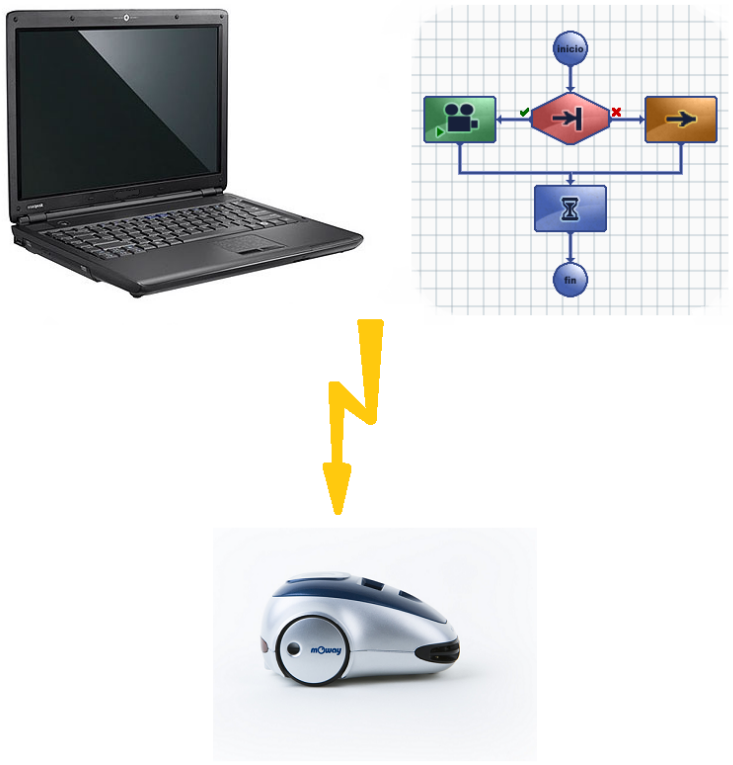
Introduction to programming

mOway



Contents

Introduction

In this tutorial we will be discussing some basic concepts for those new to programming the mOway robot. The easiest way to begin is to start using **flowcharts**.

We will be dealing with the basic aspects of programming, such as what a **condition** is, the concepts **AND** and **OR**, and how these can help us control our robot.

To gain a proper understanding, examples of programs carried out in **MowayWorld** (mOway's programming software) have been included. You can thus try to program the robot and get a better understanding of how it works.

Once you have run the demonstration programs used in this tutorial, you can take the courage to advance your knowledge with more complex programs.

Flowcharts

Robots are controlled by **programs** consisting of a series of tasks that the robot performs in a specific order. **Flowcharts** are used in programming the mOway robot.

Question:

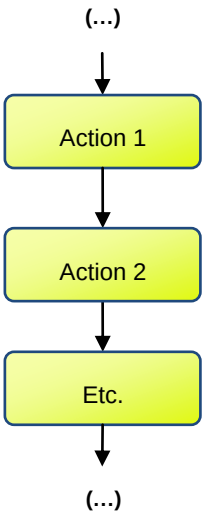
What is a flowchart?

Answer:

A **flowchart** represents a program graphically using blocks and arrows.

The **blocks** (or **modules**) are the actions to be performed by the robot when we program it, as illustrated below: move forward, switch a light on, check the sensors, etc.

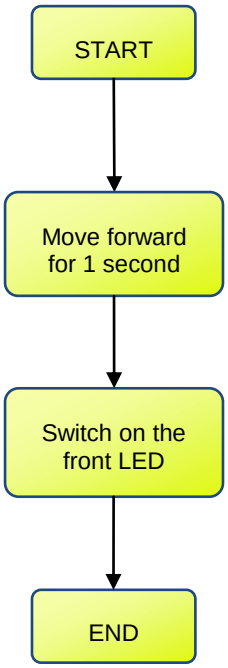
The **arrows** show the **program flow**, i.e. the order in which the robot will perform the actions indicated in the blocks.



A set of actions is displayed below to serve us as an example of a possible program for mOway. The actions are “move forward” and “switch the light on”.

1. Program start
2. To move forward for a period of 1 second.
3. To switch the front light on
4. Program end

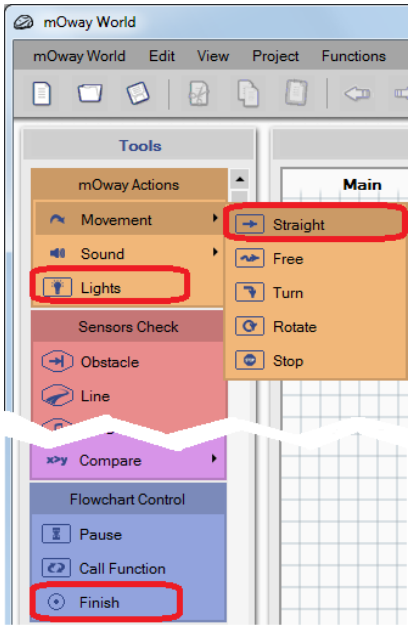
Let’s now see how this program can be represented in a flowchart. As has been said before, actions are viewed inside **blocks**. The order of the actions is indicated by connecting these with **arrows**. An example of a possible set of tasks performed by the robot is shown on the right hand side:

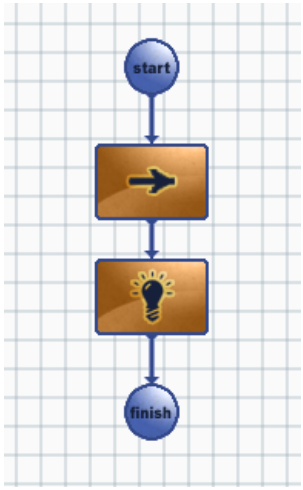


This way to represent programs makes it much easier to understand, especially in the case of complex programs, with a number of actions. The actions to be performed (and their order) can be seen and understood at a brief glance.

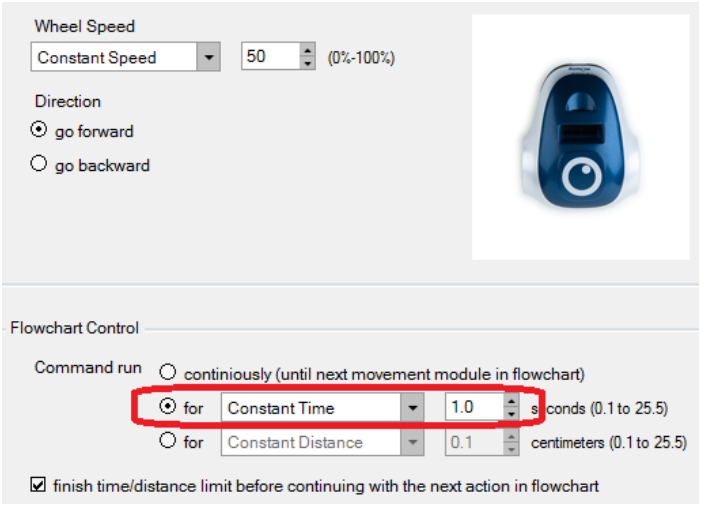
NOTE: The order of the tasks is of key importance. In the previous example, the robot moves forward with the light off, stops and then switches the light on. If we changed the order of the tasks (i.e. commanding the robot to switch on the light first and then move forward), the robot would move forward with the light on and then it would stop.

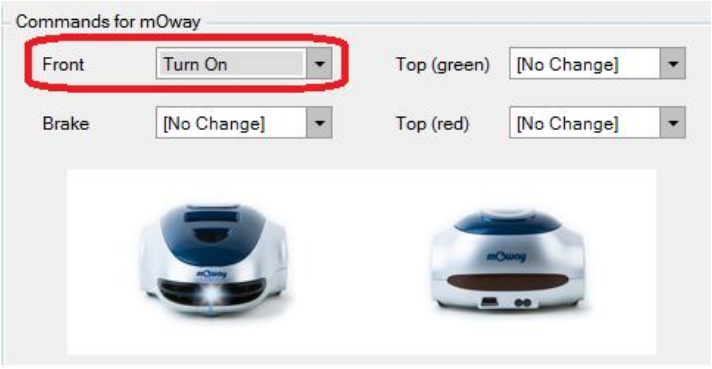
The MowayWorld software allows users to program the mOway using flowcharts, which makes it very easy to start working with mOway. Our previous example with MowayWorld is shown below: Use the “Movement -> Straight” block and the “Lights” block. Complete the program using the “Finish” block.





The configuration of the “Movement -> Straight” and “Lights” blocks is as follows:

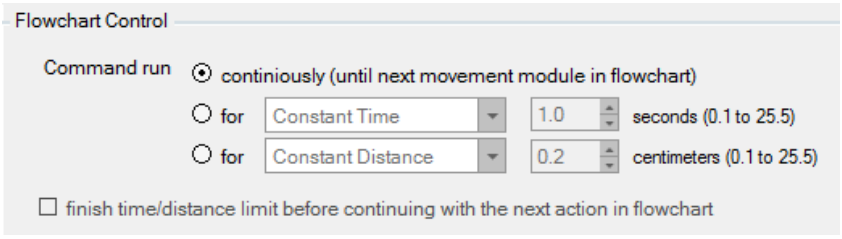


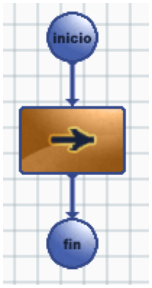


Movement blocks

Movement blocks are used for the robot to move on its two wheels. You can command the robot to move forward, move backwards, turn round or even control each of the wheels independently.

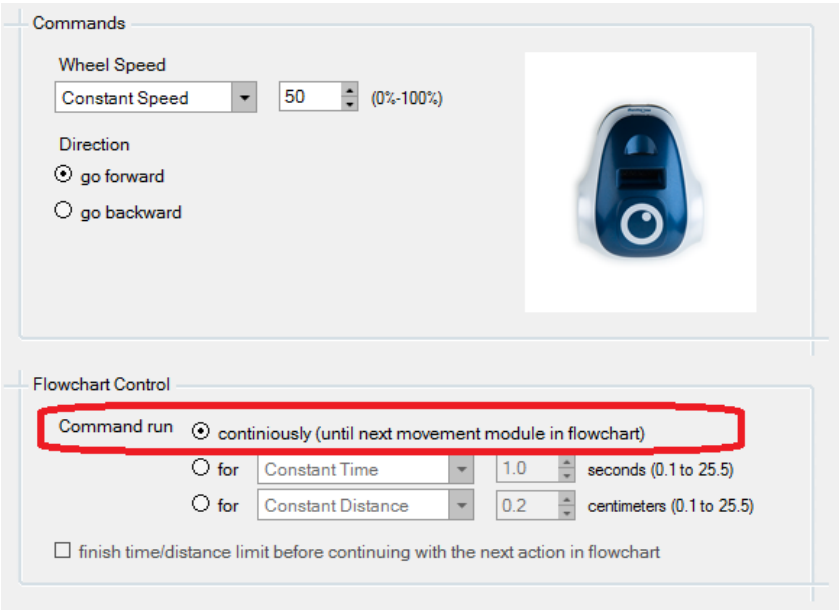
These blocks can run for both a **specified** and **unspecified** period of **time** and/or **distance**. This means you can select the robot (a) to move for some time and stop afterwards; (b) to move until travelling a distance you have fixed, or (c) to move nonstop.





✓ **PUT IT TO THE TEST:** You can start with a program to make mOway move forward (“Movement -> Straight” block). Change the parameters of the **Flow control** section (continuously, constant time, constant distance…) and check up on what happens.

If you use a block to make mOway go forward in a straight line for an indefinite period of time, and then you use the "Movement -> Rotation" block, **the robot will only rotate**.



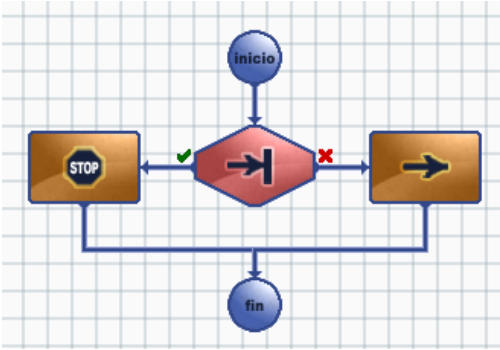
✓ **PUT IT TO THE TEST:** Test this set of blocks on your robot. You will see that the robot only rotates without moving forward. If you wish to know why, read on.

When selecting the option “continuously”, the robot will move forward until the next movement block is reached. However, as the next block (rotation) is right afterwards, the robot hardly has time to move forward and immediately goes on to rotate. That is why we cannot see the robot’s forward movement.

In order to solve it, set in the "Movement -> Straight" block **either how far or how long** we want the robot to move forward before rotating.

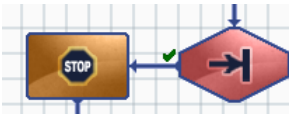
Conditionals

A **conditional block** is an action whose result depends on a **condition**. For example, let’s say we want mOway to move forward until it encounters an obstacle. When the obstacle is detected, mOway will have to stop until we remove the former. Next, the robot will start moving again.

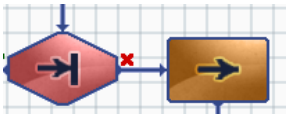


That is, the **condition** needed for the robot to stop is that its obstacle sensors detect an obstacle in front of the robot.

- If there is an obstacle, the sensors will detect it and the condition will be true (green tick). At this point, the robot is supposed to stop.



- If there is no obstacle, the sensors will not detect anything and the condition will be false (red cross). The robot will then move forward.



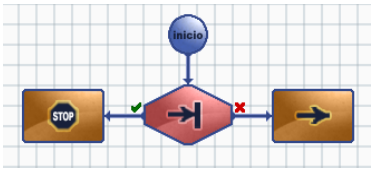
✓ **PUT IT TO THE TEST:** Test this set of blocks on your robot. As you will see, it does **not work properly**. If you wish to know why, read the following passage.

Loops

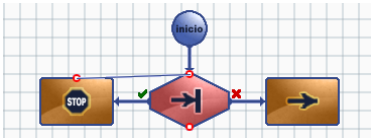
So far, we have seen that programs have a start and an end. This is not always the case. The problem about the previous chapter was that mOway would check the obstacle sensors only once, it would perform one action (moving forward if an obstacle is not detected or stopping if it is detected) and then the program would end.

In order to make the previous program work correctly, it is necessary to continually check the sensors. For that purpose, we will write a looping program, that is, a “closed circle”:

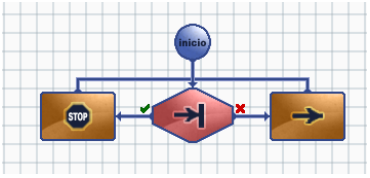
1. Remove the “Finish” block.



2. Link the “Stop” block with the “Obstacles” block.

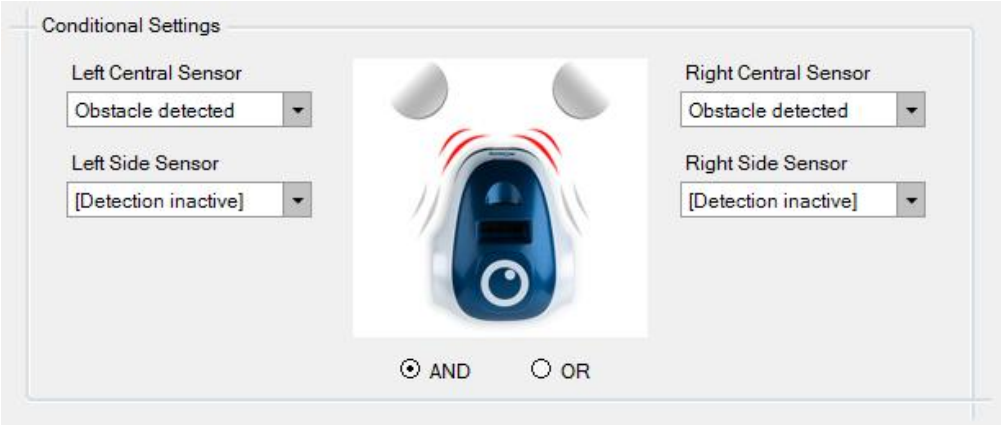


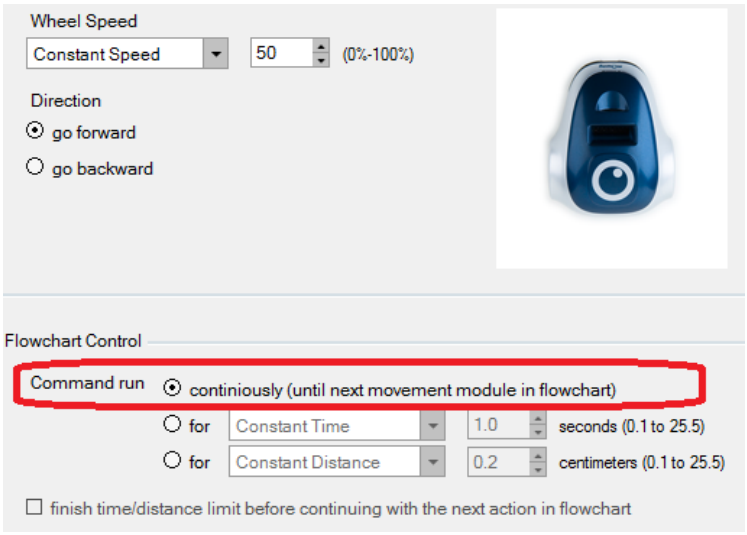
3. Finally, link the “Movement -> Straight” block with the “Obstacles” block.



We have just created a **loop**. The robot will perform the same program over and over and check for any obstacle.

✓ **PUT IT TO THE TEST:** Test this program on your robot. The "Obstacle" block can be configured to check the central sensors. The “Straight” block is configured to perform continuously.





When you switch the robot on, it will move forward. If you place your hand in front of the robot, it will stop until the hand is turned away.

AND/OR Operators

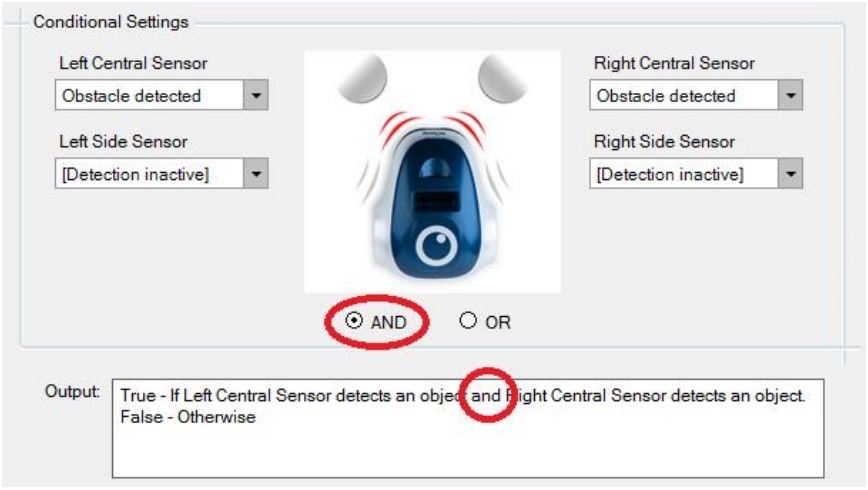
In the “Obstacle” and “Line” conditional blocks, you can select either **AND** or **OR**, together with what sensors you want mOway to use. Let’s see the difference between these two options.

? Question:

What is the meaning of the AND/OR options?

Answer:

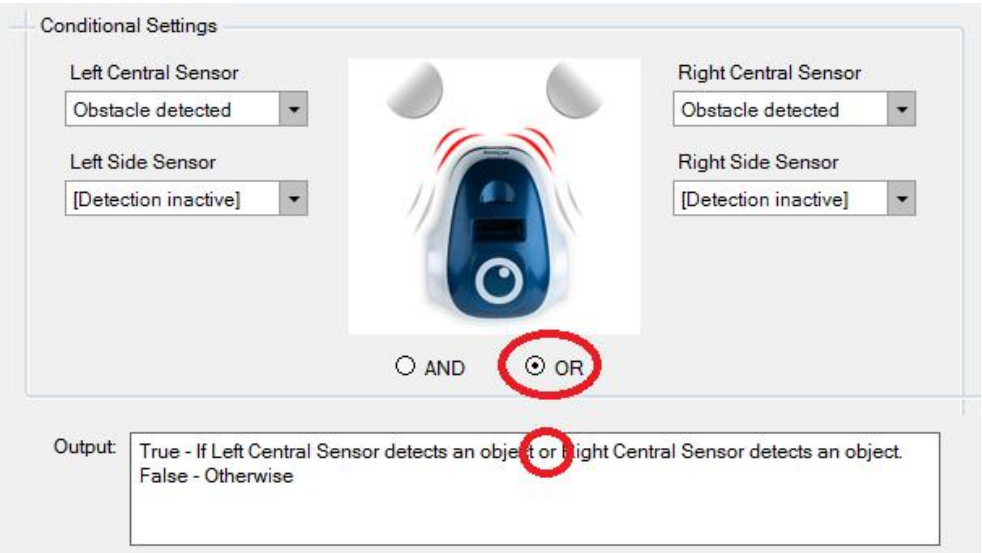
If you choose the **AND** option, the output of the block will be true when **ALL** the conditions are met. For example, if the two obstacle central sensors are activated and the “AND” box is ticked, the output will be true when the left central sensor detects the obstacle **and** the right central sensor detects another obstacle.



✓ **PUT IT TO THE TEST** Make a program that switches the frontal LED on when the output of the "Obstacles" block is true. You will see that the LED only switches on when both of the sensors detect obstacles.



However, if you choose the **OR** option, the output of the block will be true when **ANYONE** of the conditions is met. For example, if you activate the two obstacle central sensors and tick the OR checkbox, the output will be true when either the left central sensor detects an obstacle **or** the right central sensor detects an obstacle.



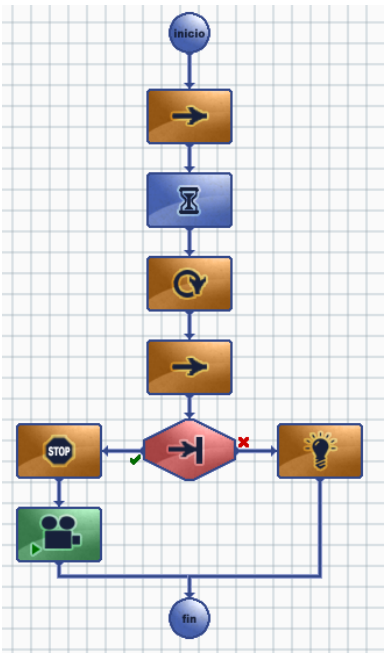
✓ **PUT IT TO THE TEST** Make a program that switches the frontal LED on when the output of the "Obstacles" block is true. You will see that in this case the LED switches on when any of the sensors detects an obstacle.

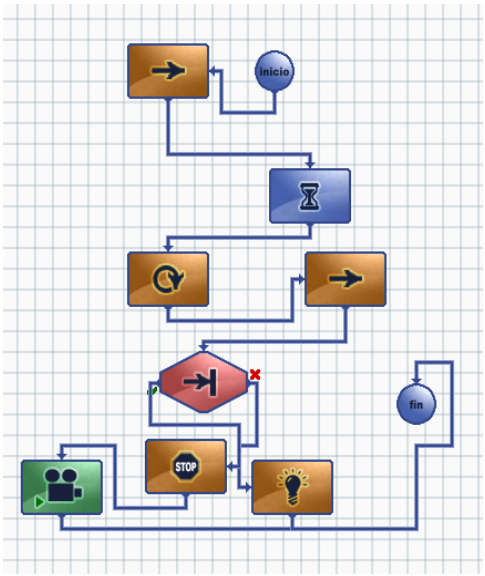


Setting flowcharts in order

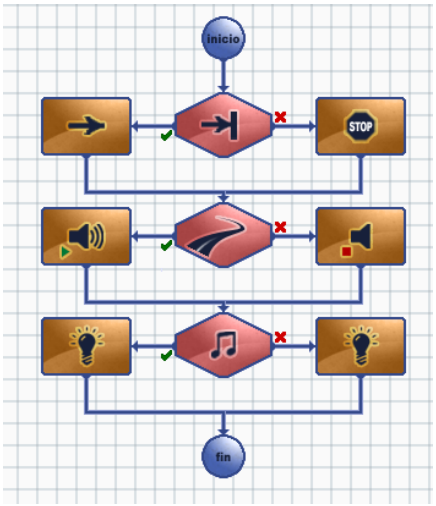
Finally, here are some tips for building your flowcharts:

- Set the blocks in a neat order to gain a clearer understanding of the **program flow**. The following two images show the same program. As can be seen, the left-hand image is much easier to understand.





- Set the true outputs of the conditionals on the same side. For example, in the case of a flowchart containing several conditional blocks, set the true output actions on the left and the true ones on the right. This is not compulsory, but it leads to a better understanding of the program.



- Try to avoid arrow crossing. If that happens, it is possible to modify the design through the marks that will appear whenever you place the cursor on them.