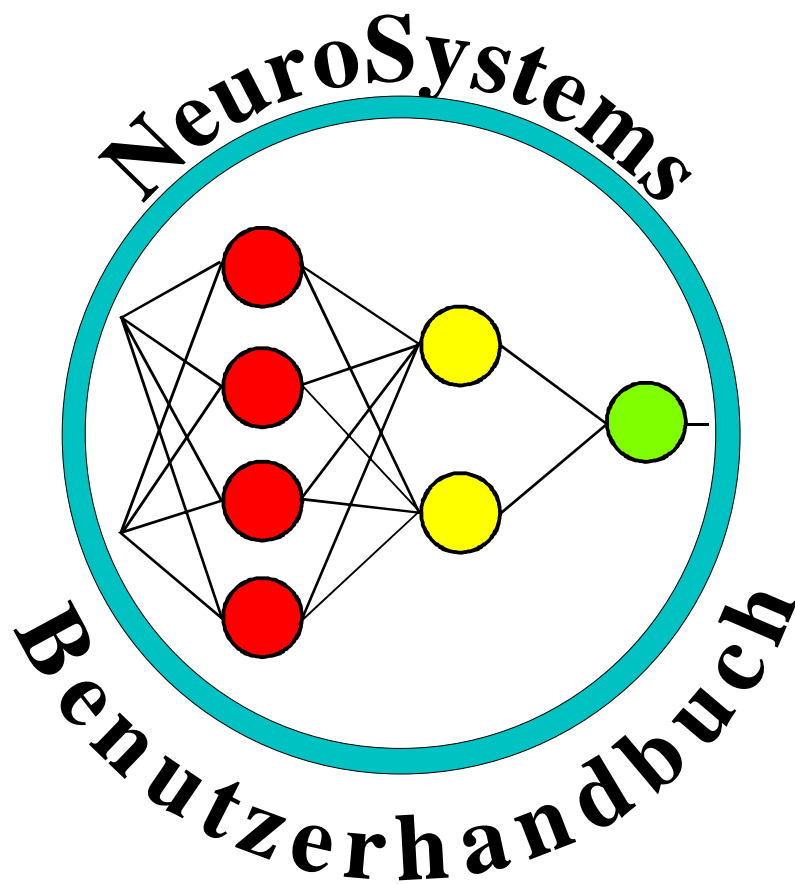
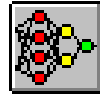


SIEMENS

NEUROSYSTEMS V5



- MANUAL -

NEUROSYSTEMS

NeuroFuzzy Projecting Tool, consisting of:

- Configuration Tool for PC
- Runtime-Modules for various Targetsystems

Version 5.0

Siemens AG, 2006

- User's Manual -

Siemens AG, I&S

Postfach 32 20

D-91052 Erlangen

Support you get from:



it4industry@siemens.com



+49 9131 7 46111



+49 9131 7 44757

Postfach 3240, 91050 Erlangen

IBM is a registered trademark of International Business Machines Corporation.

Windows is a protected name of the Microsoft Corporation.

The subject to change without a prior notice.

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Copyright © Siemens AG, 2006 All rights reserved.



1	Introduction.....	9
1.1	Neural Networks	9
1.1.1	History of Neural Networks.....	9
1.1.2	Fundamentals of Neural Networks	10
1.1.2.1	Applications and Properties.....	10
1.1.2.2	The Nature of Neural Networks	11
1.1.2.3	Multilayer Perceptron (MLP Network)	15
1.1.2.4	Radial Basis Function Network (RBF Network).....	17
1.1.3	How Neural Networks Work	19
1.2	Fuzzy Logic.....	21
1.2.1	History of Fuzzy Logic	21
1.2.2	Fundamentals of Fuzzy Logic	22
1.2.3	Fuzzy Control	24
1.3	NeuroFuzzy Systems.....	30
1.3.1	Principle	30
1.3.2	NeuroFuzzy Networks (NFN)	31
1.3.3	Applications	35



2	First Steps	40
2.1	Installation.....	40
2.1.1	System Requirements	40
2.1.2	Installation of <i>NEUROSYSTEMS</i>	41
2.1.3	Language Adjustment	44
2.2	What's new in V5?	45
2.3	Creating a Network	47
2.3.1	Configuration of the Network.....	48
2.3.1.1	Defining the Number of Inputs and Outputs and the Targetsyste.....	48
2.3.1.2	Editing the Inputs and Outputs	50
2.3.1.3	Determining the range with the learning data file	52
2.3.2	Type of Network.....	54
2.3.2.1	Selecting the type of network	54
2.3.2.2	Defining the network structure.....	55
2.4	Learning Process.....	58
2.4.1	Starting and Stopping the Learning Process	61
2.4.2	Editing the Learning Data File	65
2.4.2.1	Structure of the learning data file	65
2.4.2.2	Changing the learning data file.....	66
2.4.3	Evaluating the Learning Process	67



3 Runtime Modules..... 70

3.1	SIMATIC S7-Funktion Blocks (FBs).....	70
3.1.1	General.....	70
3.1.2	Library Contents	71
3.1.3	Block Structure	71
3.1.4	Block Diagram and Parameters of the FBs.....	72
3.1.5	Calling the Function Blocks	74
3.1.6	External Setting of the I/Os	75
3.1.7	Execution Control.....	75
3.1.8	Influence using the Configuration Tool.....	75
3.1.9	Evaluating the Parameter "ERROR"	76
3.1.10	Typical Execution Times.....	78
3.2	SIMATIC WinCC.....	79
3.2.1	Installing the WinCC-OLL	79
3.2.2	WinCC Applications with the WinCC-OLL	79
3.2.2.1	Editing Actions for a Neuro Object.....	79
3.2.2.2	Example C-Action for Neuro Object "Neuro1"	80
3.2.2.3	Simple Neuro Example.....	81
3.2.2.4	Limitations.....	81
3.3	ActiveXControl	82
3.3.1	Installing the ActiveX Control	82
3.3.2	The Properties.....	83
3.3.3	The Events	85
3.3.4	The Property Page.....	86
3.3.5	Graphical Interface	87
3.3.6	Using the Control in SIMATIC WinCC.....	89
3.3.6.1	Registering the ActiveX Control with SIMATIC WinCC.....	89



3.3.6.2	Inserting the ActiveX Control in SIMATIC WinCC.....	92
3.3.6.3	Setting the Properties with SIMATIC WinCC	97
3.3.6.4	Reacting to the Control's Events in SIMATIC WinCC	99
3.3.6.5	Example:.....	100
3.3.6.5.1	Using the Control with Tags and Direct Connections	101
3.3.6.5.2	Using the Control with Direct Connections only.....	109
3.3.7	Using the Control in SIMATIC WinCC Flexible.....	114
3.3.7.1	Inserting the ActiveX Control in SIMATIC WinCC Flexible.....	114
3.3.7.2	Example: Using the Control with SIMATIC WinCC Flexible and Scripts.....	117
3.3.8	Limitations	124
3.4	OPC.....	125
3.4.1	Basics OPC – OLE for Process Control	125
3.4.2	DCOM settings	126
3.4.3	Configuring DCOM on the OPC DA server.....	128



4 Operating Structure of NeuroSystems 129

4.1	Working Window.....	129
4.2	Menu: File.....	131
4.2.1	New.....	131
4.2.2	Open.....	133
4.2.3	Close	133
4.2.4	Save.....	134
4.2.5	Save As	134
4.2.6	Import / Export	135
4.2.7	1, 2, 3, 4	135
4.2.8	Exit.....	135
4.3	Menu: Edit.....	136
4.3.1	Inserting Inputs/Outputs	136
4.3.2	Deleting Inputs/Outputs.....	137
4.3.3	Editing Inputs/Outputs.....	137
4.3.4	Editing the Network Type	140
4.3.4.1	MLP Network.....	141
4.3.4.2	RBF network:	142
4.3.4.3	NFN (neurofuzzy network):	142
4.3.4.4	Network Properties	144
4.4	Menu: Targetsystem.....	145
4.4.1	Manager	146
4.4.1.1	Installed targetsystems.....	147
4.4.1.2	Deleting targetsystem components	148
4.4.1.3	Installing targetsystem components.....	148
4.4.2	Selection	148
4.4.3	Targetsystem S7.....	150



4.4.3.1	Selection	150
4.4.3.2	Targetsystem: Connect	153
4.4.3.3	Targetsystem: Disconnect.....	157
4.4.3.4	Targetsystem: Read	157
4.4.3.5	Targetsystem: Write	157
4.4.4	Targetsystem WinCC.....	158
4.4.4.1	Selection	159
4.4.4.2	Connect.....	160
4.4.4.3	Disconnect	160
4.4.4.4	Read.....	160
4.4.4.5	Write.....	160
4.4.5	Targetsystem ActiveX	161
4.4.5.1	Selection	161
4.4.5.2	Connect.....	162
4.4.5.3	Disconnect	162
4.4.5.4	Read.....	162
4.4.5.5	Write.....	162
4.4.6	Targetsystem OPC	163
4.4.6.1	Selection	163
4.4.6.2	Connect.....	164
4.4.6.3	Disconnect	168
4.4.6.4	Read.....	168
4.4.6.5	Write.....	168
4.5	Menu: Learn.....	169
4.5.1	Learn: File Selection.....	169
4.5.2	Range	170
4.5.3	Distribution	173
4.5.4	Combination	178
4.5.5	Input Relevancy	181
4.5.6	Start.....	183
4.5.7	Learn: Stop.....	186



4.5.8	Learn: Continue	187
4.5.9	Fine Tuning.....	188
4.5.10	Error Development	189
4.6	Menu: Test.....	190
4.6.1	3-D Graphics.....	190
4.6.1.1	Display limits.....	192
4.6.1.2	Animation	192
4.6.1.3	Setting the Parameters	194
4.6.2	Curve Plotter	195
4.6.2.1	Curve settings	196
4.6.2.1.1	Curve set	198
4.6.2.1.2	Value Fields	199
4.6.2.2	Start	200
4.6.2.3	Stop.....	201
4.6.2.4	Archiving the Recording Data in the Computer RAM.....	201
4.6.2.5	Curve Generator - Parameterization of the Curves.....	203
4.6.2.6	Online/Offline.....	205
4.6.3	File Selection	205
4.6.4	Error Diagram.....	206
4.6.5	Vector Representation	210
4.6.6	Signal Representation	212
4.6.7	Error Summary	215
4.7	Menu: View	217
4.7.1	Toolbar.....	217
4.7.2	Status bar	218
4.8	Menu: Window.....	219
4.9	Menu: Help ("?.").....	220



5	Miscellaneous	223
5.1	Trace	223
5.2	Spezifikation of the SIEMENS-Neuro-Language (SNL)	225
5.2.1	Introduction.....	225
5.2.2	Spezifikation	225
5.2.2.1	NET DEFINITION SECTION	226
5.2.2.2	CLUSTER DEFINITION SECTION	227
5.2.2.3	CONNECTOR DEFINITION SECTION	228
5.2.2.4	I/O NORM DEFINITION SECTION	229
5.2.2.5	UNIT DEFINITION SECTION	229
5.2.2.6	CONNECTION DEFINITION SECTION	230
5.2.2.7	Knotentypen und Gewichte	230
5.2.2.8	Comments	231
5.2.3	Examples.....	231
5.3	Example Coat Thickness Control.....	234
5.3.1	Creating a new project	236
5.3.2	Naming and normalizing the inputs and outputs	236
5.3.3	Selecting a suitable network type and structure	237
5.3.4	Selecting the learning file	238
5.3.5	The learning process	238
5.3.6	Visualizing the network behavior	239
5.3.7	Saving the project	242
5.4	References.....	244
5.5	Index.....	246



1 Introduction

This part is subdivided into three chapters and its aim is to provide an introduction to the neural networks, fuzzy logic and neuro fuzzy systems, which is as brief as possible and easy to understand.

Theory is kept to a minimum for clarity's sake and illustrated with simple, practical examples and diagrams.

1.1 Neural Networks

1.1.1 History of Neural Networks

The first neural networks were developed by biologists in the hope of being able to simulate natural neural networks. The term "neuron" is used in medicine and biology to denote the nerve cell. Mathematicians and engineers have adopted this model for use in mechanical information processing. These "artificial neural networks" are a greatly simplified attempt to imitate the function of nerve cells as found in lower-order organisms. When we talk about a "neural network" we mean an "artificial neural network". Like biological brain structures neural networks have the ability to learn.

The origins of artificial neural networks go right back to the early 1940's.

In 1943, W. S. McCulloch and W. Pitts developed a model ("MP neuron") that imitated the function of human nerve cells [5]. This greatly simplified abstract neuron model already shares the following important functional properties with a nerve cell:

- Structure: Many inputs (synapses) and one output (axon).
- States: Two possible states (inactive or activated).
- Connection: The neurons are interconnected (networked).
- Independence: The state of a neuron only depends on its input activation.
- Activation condition: A neuron is stimulated if enough inputs are activated.



D. O. Hebb formulated the first learning method in 1949 [9] ("Hebbian learning rule"). Neural networks capable of learning were created on this basis and finally in about 1955, the first computer simulations were implemented. P. J. Werbos [25] and D. E. Rumelhart [18] developed the backpropagation learning algorithm, which is the most frequently used learning method.

On the basis of this work the theory of artificial neural networks was elaborated still further. After 1970, neural networks with the associative memory function were created and T. Kohonen developed self-organizing feature maps [16]. After 1985 the further development and application of neural networks really took off.

Another - non-biologically inspired - approach was networks based on the radial basis functions (RBF). They originated from mathematical approximation theory and were first used in 1988 by D. S. Broomhead and D. Lowe [4].

1.1.2 Fundamentals of Neural Networks

The usefulness of neural networks stems from their ability to learn, i.e. their ability to simulate the behavior of a process from a collection of input and output data about a process (procedure, functionality). The range of applications, technical and non-technical, to which they can be applied, is therefore very wide.

1.1.2.1 Applications and Properties

Neural networks can be used for problems, whose structure and solution are not known or are only partially known. Using an example, the neural network learns the solution to a problem. This type of information processing is therefore fundamentally different from a conventional programmed system.

Applications of neural networks

Neural networks are typically used for the following tasks:

- Pattern recognition (e.g. for machine reading of handwriting)
- Identification of characteristics or processes
- Filtering of data (e.g. for intercontinental telephone systems)
- Data evaluation (e.g. for diagnostic systems)



- Data interpolation
- Closed-loop and open-loop control of processes
- Optimization tasks
- Prognoses
- Soft sensors
- Classification

Advantages and disadvantages of neural networks

- ☺ The great advantage of neural networks is solving a problem using example data. This data might, for example, be available in the form of measurement series.
- ☺ Another strength of neural networks is their adaptability, i.e. they can adapt to a new situation by changing their behavior.
- ☺ A neural network is especially suitable for simulating complex and nonlinear dependencies.
- ☹ One disadvantage of neural networks is that it is generally not possible to understand how they have solved a problem.
- ☹ The network is no "smarter" than the data you "trained" it with, i.e. the example must be an adequate representation of the problem.

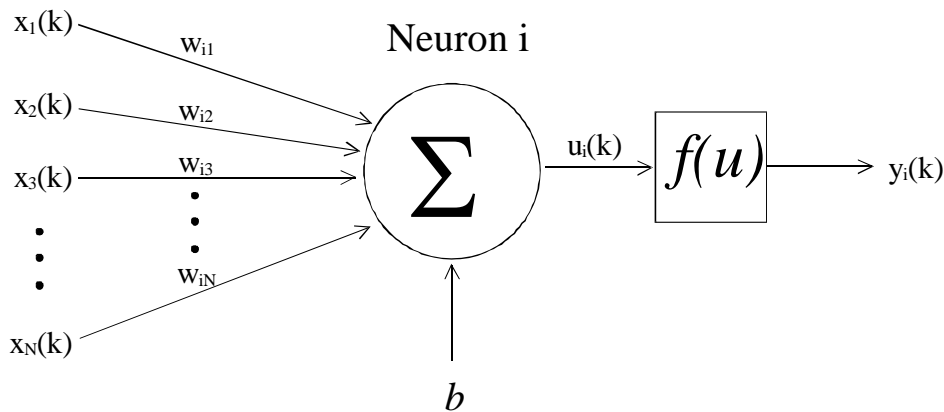
1.1.2.2 The Nature of Neural Networks

Relatively simply structured elements (neurons) receive data from numerous neighboring elements with which they are linked via weighted connections and associate this data according to simple rules. Although the complexity of each element is relatively low, interlinking them can considerably increase the power of the network as a whole.

The neurons are the individual elements of the neural network. The ordered links between them constitute the structure of the network. The fundamental principle of the neural network can be illustrated using the example of an individual neuron (see the diagram below).



The neuron



The diagram above is a generalization of the MP neuron developed by McCulloch & Pitts as early as 1943.

A neuron i has several inputs $x_j(k)$ (e.g. $j = 1 \dots N$) and 1 output $y_i(k)$, where k is the time at which the function of the neuron is considered. Information is available at the inputs and the output in the form of (initially) random real numbers. The sum of the numbers at the input, which might be derived from measurements, is also called the input pattern or input vector. The connection lines show that the neuron exhibits an input/output characteristic, in which the output value only depends on the input values and not vice versa.

(The output can also be "fed back" to produce an "additional input". This "neuron with internal feedback" is not discussed further here.)

The behavior of neurons is characterized by two properties: First the weighted sum $u_i(k)$ of all inputs is calculated from the weights (factors) on the connection lines (edges) w_{ij} . The value at the output is then derived from this sum using the activation function $f(u)$. The variable b (bias) only causes a shift in the y-axis of the activation function. The activation function and the weights have a major influence on the behavior of the neurons.

A neuron is considered *active*, if its output has a certain value, e.g. exceeds a threshold value. This activity is the result of *stimulation* of the neuron by its inputs and the weights of the connections. The weights can promote stimulation if they are greater than zero - or inhibit it if they are less than zero. If the weight is zero, the input in question has no affect on the activity of the neuron.

It is possible to express the behavior of the neuron mathematically:

For the weighted sum, the following applies initially:



$$u_i(k) = \sum_{j=1}^N x_j(k) \cdot w_{ij}(k) + b;$$

The overall input/output characteristic $y = g(\mathbf{x})$ of the neuron i can be represented by the following formula:

$$y_i(k) = f\left(\sum_{j=1}^N x_j(k) \cdot w_{ij}(k) + b_i\right)$$

The most common *activation functions* are:

(a) Switch function:
$$y = \begin{cases} 1, & \text{for } u > 0 \\ 0, & \text{else} \end{cases}$$

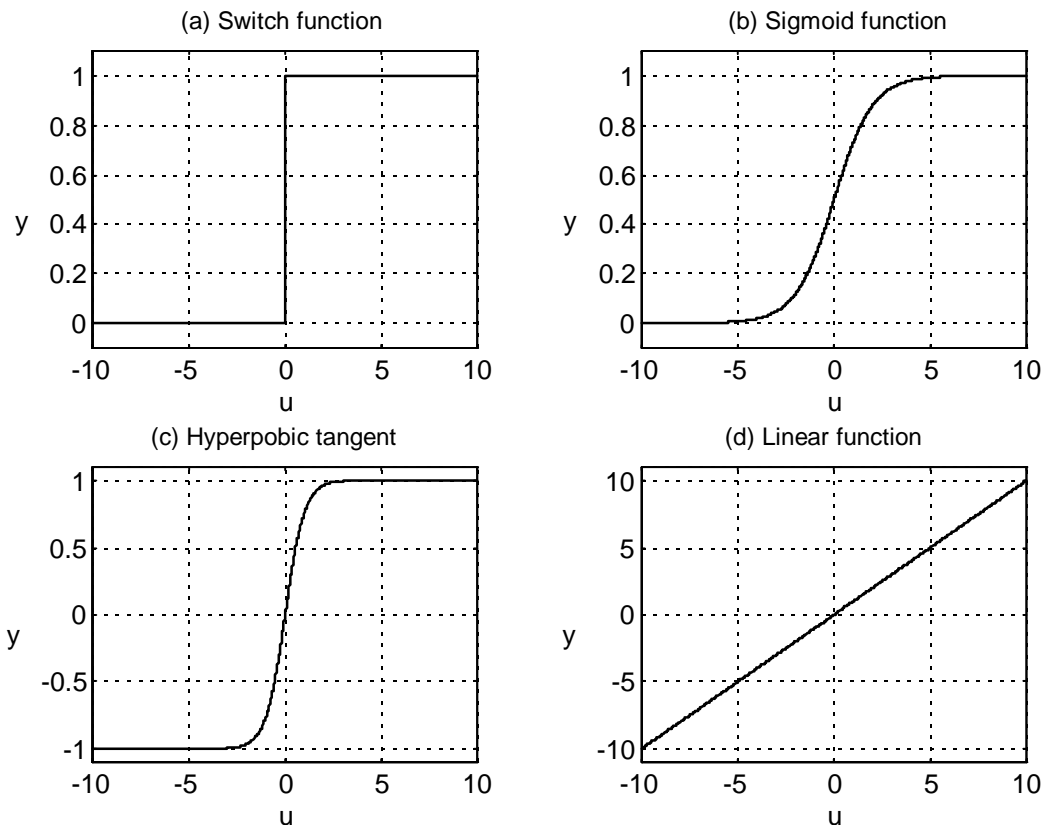
(b) Sigmoid function:
$$y = \frac{1}{1 + e^{-u}}$$

(c) Hyperbolic tangent:
$$y = \tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

(d) Linear function:
$$y = u$$



The following diagrams show the curve for each function.



For the function of the neuron it is therefore characteristic that both the input values and the weights have an influence on its output value. Moreover, for most applications, activation is nonlinearly dependent on the weighted input sum and can be either sudden (Fig. a) or gradual (Figs. b, c, d).

A neuron therefore has the ability to assign an output value to a certain input pattern that (for this pattern and a given activation function) depends only on the current weighting.

Networking of many neurons

If several neurons are interconnected, the structure formed is called a neural network. Each neuron contributes to the input/output characteristic of the overall neural network. We can therefore talk of the "knowledge" of a neural network being distributed throughout the entire network structure. It is the job of the learning method used to set and optimize the neuron weights for the problem in hand.



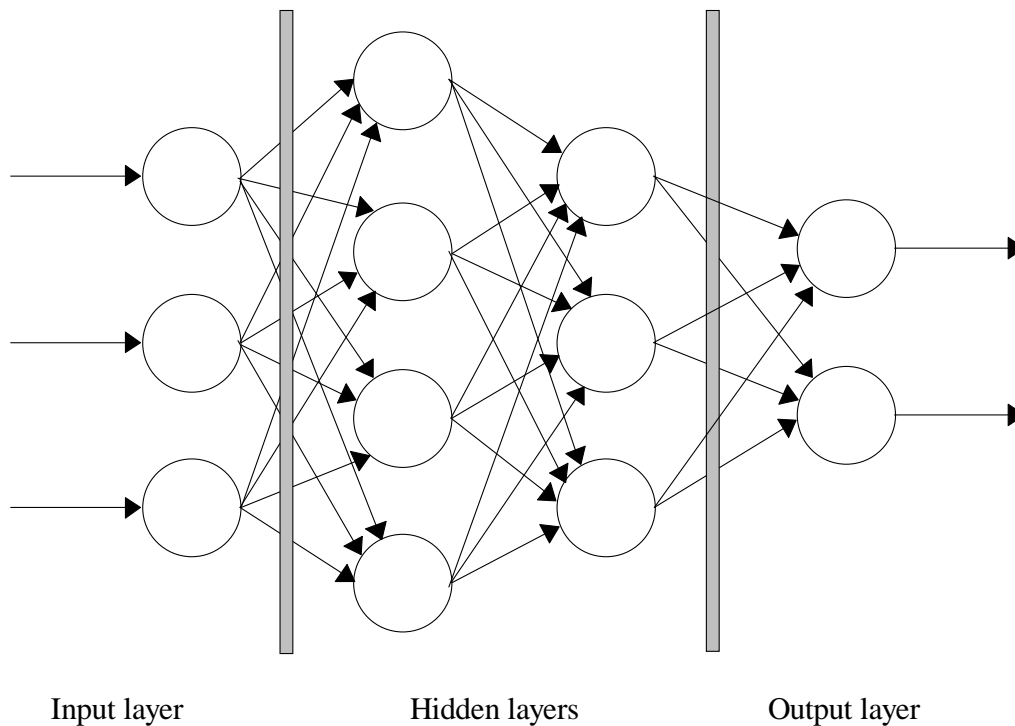
In principle, any links between neurons are possible and this has a strong influence on the behavior and properties of the network. In a neural network with only feed forward links, the input/output characteristic has the property that the output pattern at a specific point in time only depends on the input values and weights at that point in time. The time period that is required to transport and process the information in the network can usually be neglected. Networks of this type are called *static networks*. Static networks and their properties have now been well researched and are already being used for a wide range of applications.

Dynamic networks usually arise because internal feedback has been introduced so that the output of the neuron at time k depends on its own output at time $k-1$ (the output is an additional input). Dynamic networks are still at the research stage and are rarely used in practice because of the great difficulty in verifying their stability and the complex learning methods involved. However, there is another way of taking dynamic processes, such as changes in the input signals or the use of past values, into account: The dynamic input signals in question are simply made available to the static network as additional inputs (e.g. change in a value between time $k-1$ and time k).

1.1.2.3 Multilayer Perceptron (MLP Network)

Probably the best-known type of neural network is the *multilayer perceptron* (abbreviated to MLP). In this case, several neurons are interconnected by feedforward links without any internal feedback of their output signal. This is called a feedforward network and is an expansion of the classic perceptron (which only consists of a single neuron). The MLP is a static network.

In the MLP network, the neurons are arranged in several layers. The two layers that connect the structure with the outside world are called the input and output layers. The layers in between are called *hidden layers*). The following diagram illustrates this structure.



The neurons of the input layer are only used to distribute the inputs to the neurons of the first hidden layer. Each therefore only has 1 input that is assigned to the input value of the network at this position. The weight of the input connection can be used to scale the input signal. In this respect, input neurons are a special case of the neuron. Some authors do not therefore include the input layer in the number of layers of a network. In *NEUROSYSTEMS*, however, the input layer is counted.

The set of all input values at a particular point in time is the "input pattern" (input vector) of the network. The output values of all neurons of the output layer, at the same point in time, are the associated "output pattern" (output vector) of the network. The output pattern with which a network reacts to a specific input pattern is defined as the input/output characteristic of the network. It is also called the "response characteristic".

The connections between the neurons within the network are characterized by the fact that the output of each neuron of one layer usually branches to all the neurons of the next layer. Each output of one layer is therefore an input of all the neurons of the next layer. A following neuron therefore has as many inputs as it has preceding neurons. If a link has weight zero, it can be considered "non-existent" (in this particular case).



To summarize the above we can say:

A neural network consists of a number of interconnected neurons arranged in layers, resulting in a certain *architecture*. The *parameters* of the neural network are the characteristic values of the activation functions and the weights of the connections. The neural network can learn its *response characteristic*.

1.1.2.4 Radial Basis Function Network (RBF Network)

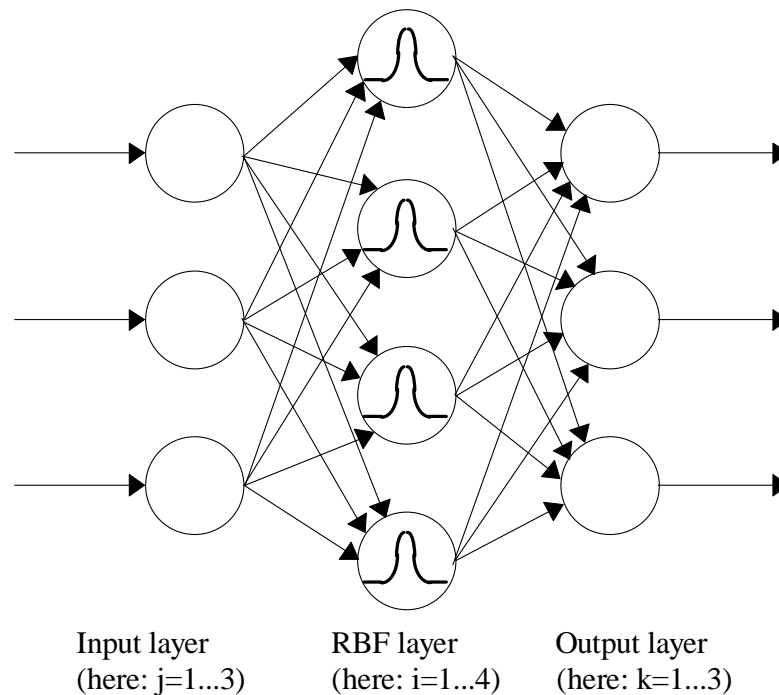
The method used in RBF networks is a little different from that used for the multilayer perceptron.

They always consist of three layers:

- The input layer,
- The second layer with RBF neurons whose activation function is the Gaussian function (as shown in the diagram below),
- And the output layer.

This is only a short outline of the principle. For more details, please consult the references [8].

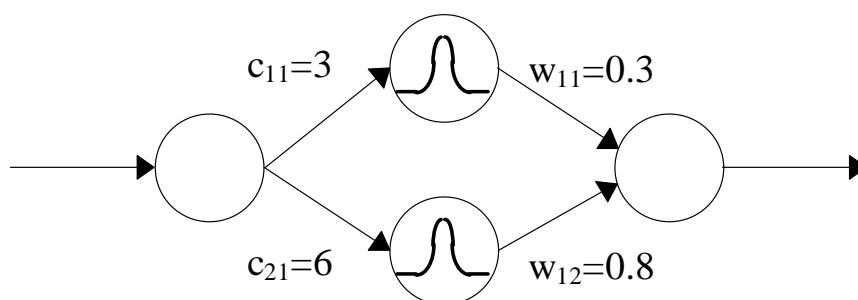
Here too, the connections between the input and the RBF layer are provided with weights c_{ij} . Unlike the MLP network, the weights are not multiplied with the inputs (that is the outputs of the neurons of the input layer). Instead, the RBF neurons process a measure of the similarity between the weight and the input value. The result is that each neuron of this second layer is stimulated more strongly, the more similar the input values and the corresponding weights are.



At the connections between the RBF layer and the output layer there are weights w_{ki} that are multiplied with the outputs of the RBF neurons, which results in additional weighting ("The neuron whose weights are most similar to the input values determines the output").

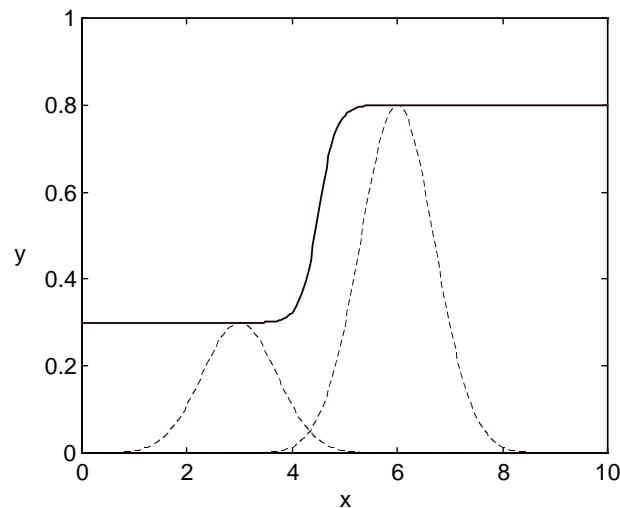
Each RBF neuron can therefore be seen as a sort of rule: If the inputs are similar to the weights c_{ij} , the output of the network is similar to weight w_{ki} between the i^{th} RBF neuron and the k^{th} output.

This can be illustrated with a simple example with one input ($j=1$), two RBF neurons ($i=1, 2$) and one output ($k=1$):





Two RBF neurons are placed at positions $x=3$ and $x=6$. The inference (weights) of the two neurons are $y=0.3$ and $y=0.8$. This results in the following output curve:



In this type of network, the position of the bell-shaped functions (weights \underline{c}), their width and their inference (weights \underline{w}) are learnt.

1.1.3 How Neural Networks Work

A neural network characterized by two stages: *learning* and *reproduction*. The latter is the actual working phase in which input data is processed to form the required output data (e.g. text recognition). The network can only do that if it has first learnt how. This is called "training" and is done with "suitable training data". Training data or learning data are the names given to the input pattern and corresponding output patterns that represent the input/output characteristic required. Whether this data is "suitable" or not depends on the type of network and on the required input/output characteristic of the network.

So, how does a neural network learn?

You will remember from the previous chapter that the input/output characteristic of a neuron can be changed by the connection weights and the parameters of the activation function.



This gives rise to different methods:

- Learning methods that only correct the connection weights of the neurons without changing the actual structure of the neural network (e.g. the backpropagation algorithm)
- Learning methods that not only change the weighting of connections but also change the structure of the neural network, i.e. add or remove neurons (e.g. cluster methods).

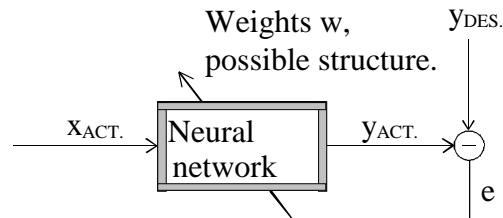
Most learning methods just alter the weighting depending on success or failure during training. These learning methods are, in principle, nothing other than a - usually - nonlinear optimization problem in which the network weights are the parameters to be optimized.

One of the possible training methods is "supervised learning". Supervised learning data can only be used with sets of learning that consist of a number of input patterns with corresponding output patterns (desired output patterns). Supervised learning is performed as follows:

- The network calculates the output values from the input values provided using the current input/output characteristic. The result is a current actual output pattern.
- The deviation between the actual pattern and the desired pattern is used to update the input/output characteristic of the network in such a way that the error is smaller after the next calculation of the output pattern.
- These two processes form one learning step. The learning steps are repeated cyclically until an error threshold, or a set number of steps or a set time is reached.



The following diagram illustrates the learning process.



The question remains as to how the neural network is corrected so that it really "learns", i.e. how the deviation between the desired and the actual value at the output of the neural network is minimized. In the simplest case (the backpropagation method), the contribution of each weight to the desired/actual value deviation is calculated and in a second stage, the weights are changed such that the error becomes smaller. The name of this method comes from the fact that the error component is "propagated back" to the weights causing the error.

There are numerous learning methods which we shall not explain in detail here. For more details, please consult the references, e.g. [7], [28].

1.2 Fuzzy Logic

1.2.1 History of Fuzzy Logic

Despite its name, fuzzy logic denotes a precise idea, as we shall see below. The notion of "fuzziness" in logic was pioneered by J. Lukasiewicz [17] and M. Black [2] as early as 1935.

The idea came about because it was recognized that in many cases a binary assignment true/false, yes/no, on/off is inadequate to describe real world situations. (Consider the question: "Is a car which travels at 60 mph fast or slow?" - yes or no?). In 1965, L. A. Zadeh [27] extended classic binary set theory (yes/no) with his "fuzzy set theory". He was the founder of "fuzzy logic", which can be seen as a sort of generalization of the binary logic known as Boolean algebra. On this basis, Mamdani in 1973 used fuzzy logic to control processes for the first time, which led to the development of fuzzy control as a special branch of fuzzy theory.



The importance of Zadeh's thinking initially went unrecognized. It was only many years later - in 1988 - that Japanese scientists and engineers first put fuzzy set theory into practical use [29]. After that, the fuzzy tool finally aroused worldwide interest. From 1991 onwards, fuzzy logic was used for various applications in the USA and Europe, too, and is now an established problem-solving method.

1.2.2 Fundamentals of Fuzzy Logic

Non-fuzzy and fuzzy sets

Classic set theory makes every element *a member of* or *not a member of* a set. Membership is therefore limited to just two values (yes/no, 1/0, true/false). Sets with such binary membership functions are called non-fuzzy sets. Fuzzy logic, on the other hand, expands the membership function to allow intermediate values and thus better simulates human thinking. Human beings use concepts, which have a gradual transition to the opposite meaning. For example, the terms "warm" and "cold" are temperature ranges with fuzzy boundaries, and they can even overlap. "Warm" and "cold" are therefore fuzzy sets.

As an example, let's consider the values of the temperature scale in °C as elements to be assigned to defined sets: One possible set of temperature values is called "frost". Frost denotes a non-fuzzy set of temperatures, i.e. all temperatures below 0 °C. Membership of the temperature "-5 °C" in this set is therefore "one", or "yes", and membership of temperature "+5 °C" is "zero", or "no".

The notion of "cold" denotes a fuzzy set because, while temperatures around 0 °C may be perceived as being 100% "cold", +5 °C is less "cold" and +10 °C is not "cold" at all, indeed it could be a member of a new fuzzy set called "lukewarm" ranging from +5 °C to +15 °C.

Applications of fuzzy logic

Information processing based on fuzzy logic is appropriate, often necessary, where processes are specified by verbally expressed algorithms in the form of "IF-THEN rules". For example, in complicated manufacturing processes it can be observed that operators take control actions on the basis of their experience using simple if-then criteria ("If the temperature is high and the pressure is medium, then I must open the valve halfway.") and using this way, they often achieve good operation with quality results.



The applications of fuzzy logic are not limited to closed-loop and open-loop process control; fuzzy logic is also used for information processing in measured value acquisition (pattern recognition, signal processing) and in operations management and planning (scheduling, forecasting). Fuzzy logic makes it possible to transfer the "experience" of an expert to a computer and obtain an initial and "automated" - though perhaps not yet optimal - solution quickly.

Advantages and disadvantages of fuzzy logic

The fundamental advantage of fuzzy logic is that no mathematically expressed description of the process to be automated, in the form of algebraic equations or differential equations etc, is required to design and use fuzzy systems.

However, not using a mathematical model of the process is also a source of uncertainty. For example, the many degrees of freedom in designing fuzzy systems can be a special disadvantage.

The following overview provides a comparison:

- ☺ Simple implementation of verbally expressed rules (if ..., then...) on a computer to solve a problem.
- ☺ The behavior of the fuzzy system is understandable to human beings.
- ☺ Avoids the costly development of a mathematical description when compared with conventional methods.
- ☺ Possible to use for processing complicated and involved processes.
- ☹ Task definitions with not enough knowledge of the system and little or very imprecise knowledge of the system behavior result in bad, possibly unusable, fuzzy solutions.
- ☹ Usually no adaptability and learning capability if the system behavior changes.
- ☹ Design of a system requires experience because of the many degrees of freedom.



1.2.3 Fuzzy Control

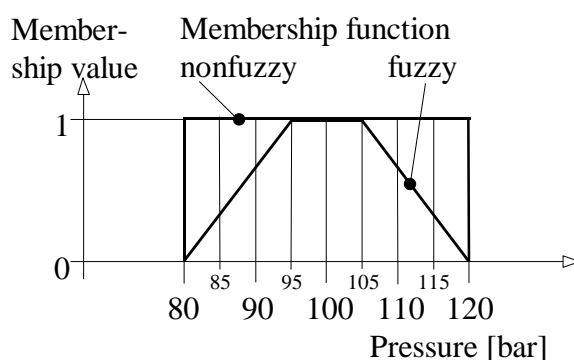
Through experience of using fuzzy logic for information processing in open- and closed-loop control (fuzzy control), especially in "fuzzy rule bases", tried and tested methods have emerged. The most important of these are explained below.

The membership function

An introductory example: Pressure monitoring in a processing plant

The operator of a processing plant determines a pressure of 100 bar as being optimal for the production process. However, he allows an operating range of 80 to 120 bar as acceptable ("Pressure OK"). The following diagram shows the difference between the set of pressure values "Pressure OK" defined both as a non-fuzzy set and a fuzzy set. The representation of the membership values (degrees of membership) via the elements of the set (here pressure values) is the *membership function*.

In general, membership can be defined by the most varied functions (such as normal distribution, sigmoid function, trapezoidal function). For practical purposes, *triangular* and *trapezoidal functions* have become the standard. They can be described with 3 or 4 points and therefore do not cause much computational overhead.



If you imagine how the operator works, you can see the advantage of modeling with fuzzy logic. The operator prepares to take action when the pressure display reaches 115 bar because the pressure can only be considered 30% OK. A binary control with a threshold value contact at 120 bar would not be able to perform such qualified operation and would perceive the situation all of a sudden when the pressure value left the non-fuzzy set. The

operator, on the other hand, can recognize critical values and tendencies early on and can therefore react in time taking any necessary counter-measures. To imitate this behavior it is necessary to use a fuzzy system instead of a binary control.

Function of a fuzzy system



A fuzzy system processes its information fuzzily in the form of rules, such as

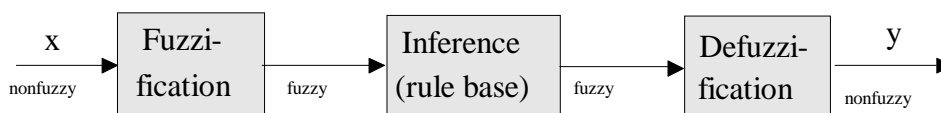
IF input value "pressure OK", THEN output value "do nothing", or

IF input value "medium", THEN make output value "small".

The input and output values are *linguistic variables* and the terms "medium" and "small" are not numeric but *linguistic values*. They are also called *fuzzy sets*. A *linguistic variable* usually possesses 3, 5 or 7 *linguistic values*. The complete collection of rules forms the *rule base*.

However, the fuzzy system is, necessarily, part of a technical system that works with numeric values (signals), i.e. with non-fuzzy values, at its inputs and outputs. So appropriate conversion must be performed at the input and output of the fuzzy system. This conversion is termed fuzzification or defuzzification.

The following diagram illustrates the principle.



- **Fuzzification**

The degrees of fulfillment for the linguistic values (degrees of membership of the fuzzy sets) of the linguistic variables are assigned to the non-fuzzy input values. So fuzzification determines, for example, to what degree the "pressure is OK" if it is, say, 115 bar. This is done using the membership function. This degree is called "Degree of fulfillment of the IF part".

- **Inference**

For each rule of the rule base, the degree of fulfillment of the THEN part is formed from the degree of fulfillment of the IF part by a certain method. This process is also called *implication*. The degree of fulfillment of the THEN part is equivalent to the degree of fulfillment of the rule, which is also called the rule intensity. All these individual rule evaluations put together result in one membership function for the output signal, which is also termed *composition*. The resulting membership function describes a "fuzzy control



command" that could, for example, be something like "Position valve just below the center".

When the IF part contains a combined statement "IF.... AND..." --- (THEN....), the fuzzy logic AND operation is executed first and the degree of fulfillment is used in the overall rule evaluation. All these statements together are often called an *aggregation*.

- **Defuzzification**

The most representative numeric (non-fuzzy) output value is calculated for the control variable from the fuzzy control command (in the form of the resulting membership function).

The following example illustrates these three typical function blocks. Here more emphasis is placed on clarity than creating realistic values.

Example: Acceleration control on a vehicle:

- Aim: Automatic acceleration and braking adapted to the situation of the vehicle
The following definitions are required before we can design the fuzzy rule base:
- Input signals of the rule base:
 1. Distance from the vehicle in front (linguistic variable "Distance")
linguistic values: "small", "medium", "large"
 2. Own current speed (linguistic variable "Speed")
linguistic values: "slow", "medium", "fast"
- Control Variables:
Acceleration a (positive - accelerate or negative - brake) of the vehicle (linguistic variable "Acceleration")
linguistic values: "neg_large", "neg_small", "zero", "pos_small", "pos_large"
- The rule base:

R1:	IF Distance = small	AND Speed = fast,	THEN a = neg_large;
R2:	IF Distance = small	AND Speed = medium,	THEN a = neg_small;
R3:	IF Distance = medium	AND Speed = fast,	THEN a = neg_small;
R4:	IF Distance = medium	AND Speed = medium,	THEN a = zero;



R5: IF Distance = medium AND Speed = slow, THEN a = pos_small;

R6: IF Distance = large AND Speed = medium, THEN a = pos_small;

R7: IF Distance = large AND Speed = slow, THEN a = pos_large;

The rules are also often represented in a matrix.

- Definition of the membership functions (fuzzy sets) for the linguistic values:

In this design step, the form of the fuzzy sets is selected along with their *range of influence*, i.e. the range of non-fuzzy values of the input and output signal is determined for which the fuzzy set has a degree of membership greater than zero.

- Definition of the inference and defuzzification procedure:

A number of methods exist. The common methods include MAX-MIN inference and MAX-PROD inference and defuzzification according to the centroid method. In practice, the choice depends on the design tools the computer program provides. In *FUZZYCONTROL++* a special method is used that only demands a low computational effort in the practical implementation of the fuzzy system. The principle is explained in the following section "how the fuzzy rule base works".

To explain how the fuzzy rule base works, let us look at an "operating case" in which the speed is 65mph and the distance is 51m. The membership functions are shown in the following two diagrams. In choosing the curves of the membership functions, the available degrees of freedom must be defined in a meaningful way (in the form of "intermediate points" of the triangles or trapezoids). For example, the fuzzy set "medium" of the input signal Distance refers to the range 30 to 120 m.

- Fuzzification:

The following can be seen from the curves of the membership functions chosen here for the two input signals:

The fuzzy statements about the distance, which is 51 m, are 30% true for "small" and 70% true for "medium". (The statement "large" is 0% true. Such cases are usually not pursued.) Similarly, the degrees of membership for "medium" (0.8) and for "fast" (0.2) result for the fuzzy sets of the "Speed" variable.

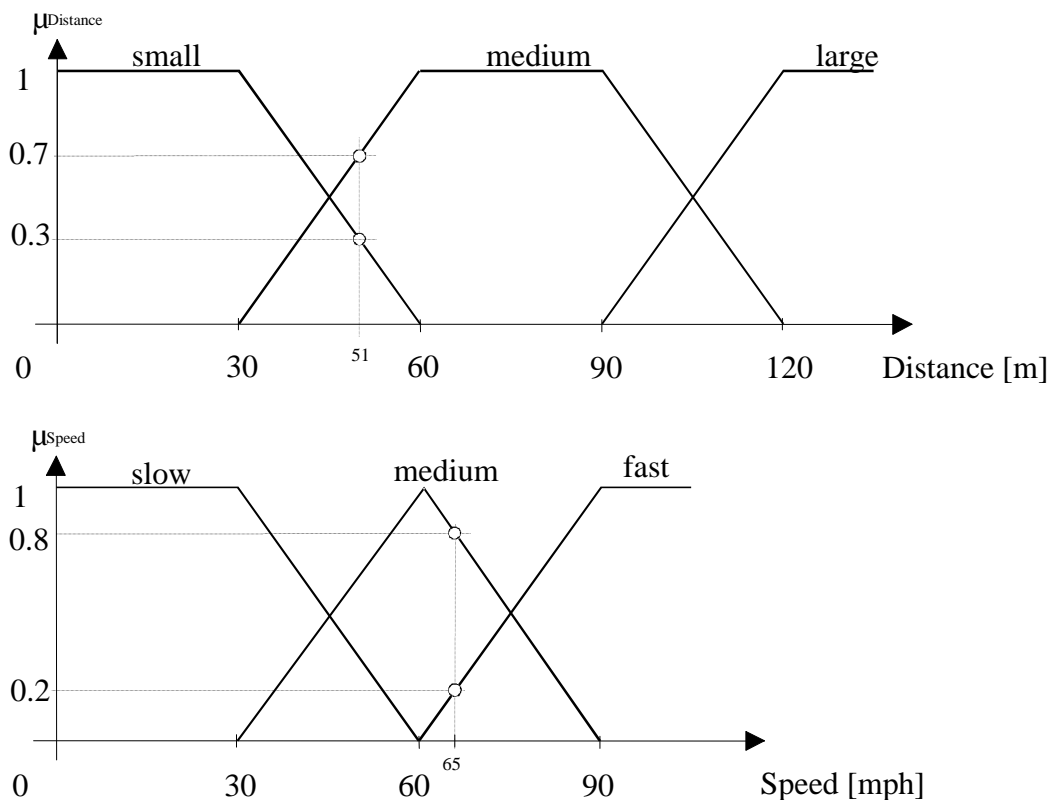


- Inference:

Let us consider rule 1 as an example:

In the IF part of the rule, the fuzzy statements about the two variables Distance and Speed are ANDed. Distance is "30% small" and Speed is "20% fast". The entire IF part of this rule is therefore considered to be 20% fulfilled after application of the MINIMUM operation for fuzzy logic AND combination (aggregation).

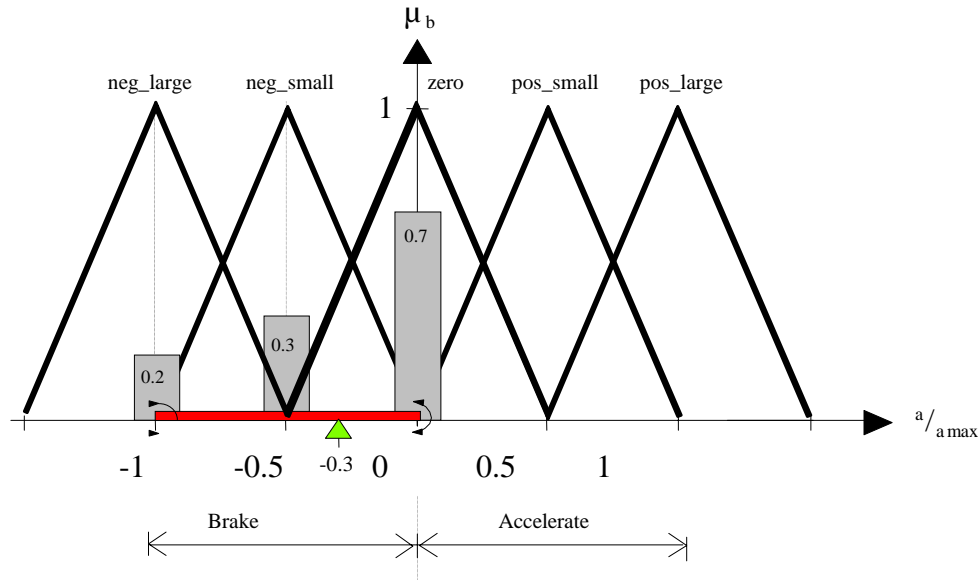
The THEN part of rule 1 refers to the fuzzy set "neg_large" of the output signal. The simplest implication method makes the (sensible) assumption that an implication can be fulfilled to no greater degree than the antecedent. Because the IF part is 20% fulfilled, it can be inferred that an acceleration is required which is 20% "neg_large" (implication). This degree of fulfillment must now be used to modify the membership functions of the output signal in the THEN part of rule 1.



The following diagram shows the output signal "Acceleration" with its 5 fuzzy sets. The fuzzy set "neg_large" is modified by the 20% degree of fulfillment from rule 1. One simple method is to consider only the height of the peak. This makes the membership functions vertical lines with height 1, i.e. 100% at the position on the output signal axis where the peak



is located. (This type of membership function is called a *singleton*). All that rule evaluation does is to shorten the line, in this example, to 20%, as shown in the diagram.



If you add the remaining rules to the evaluation for the same pair of non-fuzzy input values (51m and 65mph), the following degrees of fulfillment result:

- Rule 1: The command: Make acceleration "neg_large" is 0.2 fulfilled
- Rule 2: The command: Make acceleration "neg_small" is 0.3 fulfilled
- Rule 3: The command: Make acceleration "neg_small" is 0.2 fulfilled
- Rule 4: The command: Make acceleration "zero" is 0.7 fulfilled

In this case, rules 5 to 7 are not fulfilled, because at least one IF condition is not fulfilled, i.e. the degree of membership is zero for the current non-fuzzy input value (e.g. rule 5 with Speed slow).

The control recommendation of the fuzzy rule base consists of the above four partial statements in this operating case. (Please note that two different recommendations for the same fuzzy set "neg_small" result from rules 2 and 3.) Composition into a resulting membership function is best performed by superimposing all these "partial membership functions" (composition). One possible operation is the MAXIMUM operation. In this way, the result of rule 2 covers the result of rule 3 and the resulting control recommendation consists of the three columns shown in the diagram. With *FUZZYCONTROL++*, the SUM operation is used so that a rule intensity of 0.5 results from rules 2 and 3 at position $a/a_{\max} = -0.5$.

However, the control recommendation resulting from all rule activity is still fuzzy and might be expressed as "on no account full braking but not quite half the braking power".



- Defuzzification:

From this control recommendation, which is still fuzzy, we now have to calculate a non-fuzzy, numeric value for the control variable as a numeric value (from the manipulating range). It must be representative of the fuzzy control recommendation, i.e. the resulting membership function. For example, if the resulting membership function consists of the superimposition of partial areas of triangles and trapezoids, the position of the center of gravity of the total area on the output signal axis is the non-fuzzy value sought. In this case (see diagram), you can imagine a mechanical system in which the three columns for acceleration values, whose weight is a function of their height, are balanced on a pivot along the axis ("equilibrium"). This is the principle on which the centroid method is based.

In this case, it is possible to find the non-fuzzy value, i.e. the numeric control variable by the following simple calculation:

$$\frac{\sum a_v \cdot \mu_{a_v}}{\sum \mu_{a_v}} = \frac{-1 \cdot 0.2 - 0.5 \cdot 0.3 - 0 \cdot 0.7}{0.2 + 0.3 + 0.7} \approx -0.3$$

In this example, the vehicle's brake is applied with approx. 30% of the maximum possible range.

1.3 NeuroFuzzy Systems

1.3.1 Principle

The details of the previous have shown that the problems for which neural networks and fuzzy logic are used are fundamentally different. With some simplification, we can say:

Fuzzy logic imitates and automates human behavior and provides understandable solutions. With neural networks you generate a solution using a process of learning from sample data. The advantage is the ability of the network to learn, i.e. its ability to adapt to changed behavior and new situations. To make use of the advantages of both - the understandability of fuzzy systems and the learning capability of neural networks - the two techniques are combined. The result is a neurofuzzy system. The combination results from converting a fuzzy system into a neural network and vice versa. There are two typical procedures for applying such neurofuzzy systems:

1. There is a problem, for which you can only design a bad fuzzy solution if you do not have sufficient knowledge of the problem. The fuzzy system obtained in this way is transformed



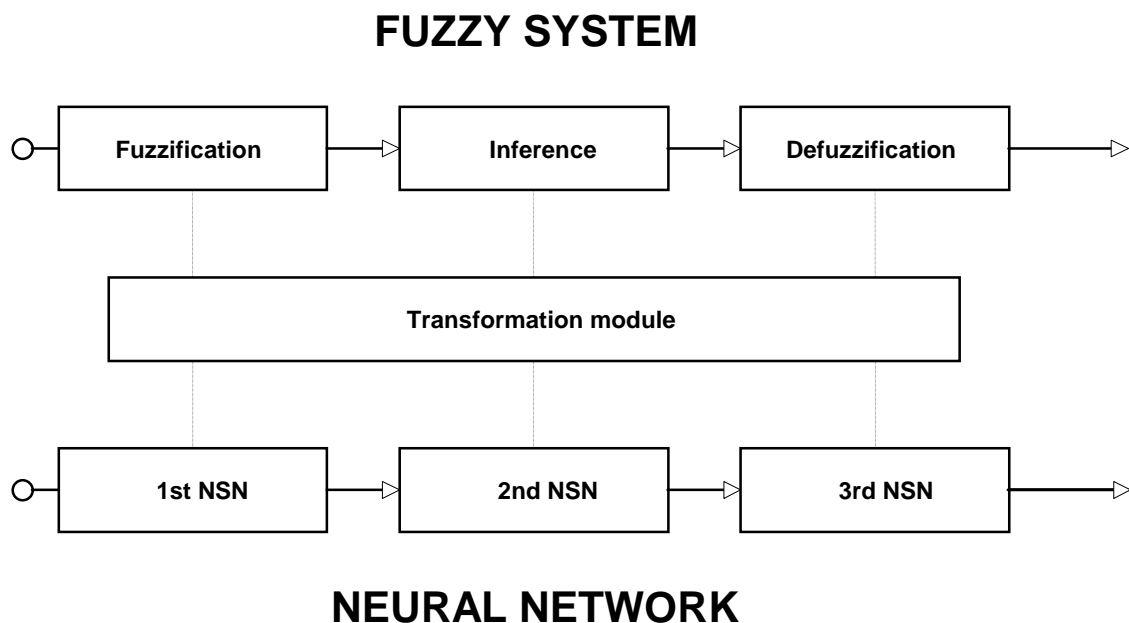
into a neural network that is optimized using example data. The optimized network can either be used directly or transformed back into a fuzzy system.

2. Using a neural network, you generate a solution from a collection of sample data that, for example, identifies an unknown process behavior. If you have used a suitable network structure, you can transform the neural network into a fuzzy system. The fuzzy system - or more precisely, its rule base - can be interpreted, and in this way you can obtain an understandable description of the problem, e.g. the process behavior.

1.3.2 NeuroFuzzy Networks (NFN)

To be able to make use of the advantages of neural networks and fuzzy systems in one project, you require a system that processes fuzzy membership functions and the rule base of the fuzzy model and can determine knowledge from sample data using neurons capable of learning.

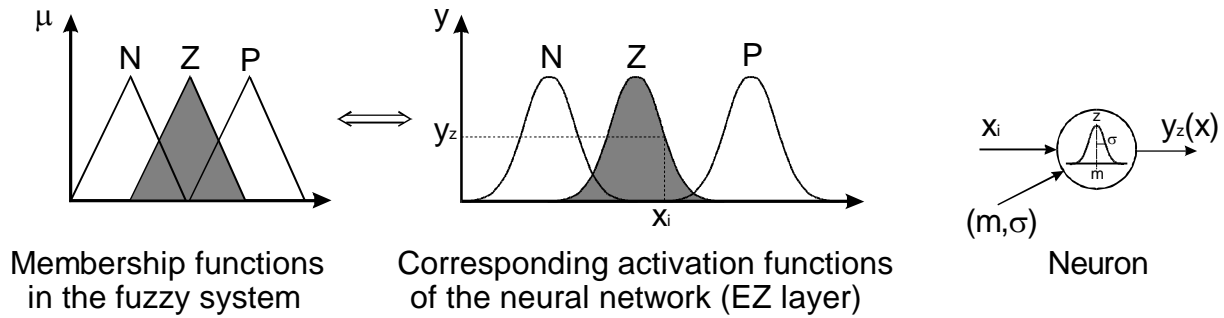
You can solve this problem by using a special neural network, called the neurofuzzy network (NFN). It consists of three neural subnetworks (NSN) that emulate the three subtasks - fuzzification, inference and defuzzification - of a fuzzy system.





1) Fuzzification in the 1st NSN

Fuzzification of the non-fuzzy input variables is implemented by a layer of neurons with activation functions similar to those of an RBF. One neuron is assigned to each membership function of the input variables.



The EZ layer of the 1st NSN has m neurons, where:

$$m = \sum_{i=1}^a n_i \quad a = \text{number of inputs,}$$

$$n_i = \text{number of membership functions of the } i^{\text{th}} \text{ input.}$$

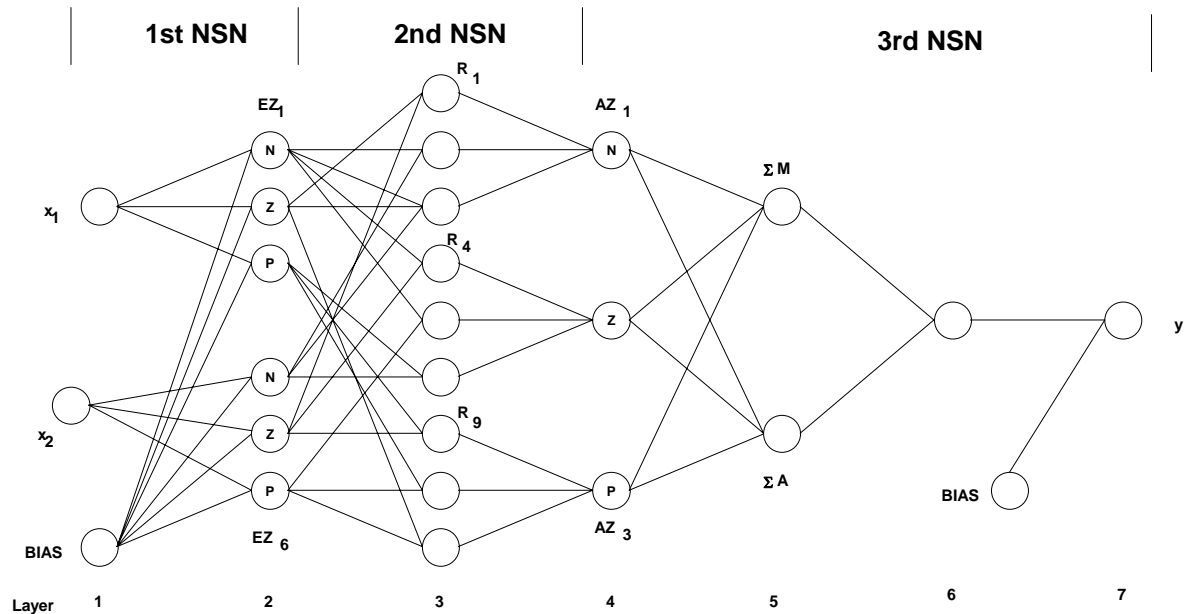
2) Inference in the 2nd NSN

The rule base of the fuzzy system is in the 2nd NSN and one neuron is assigned to each rule. The IF part of a fuzzy rules is implemented by the first neuron layer in the 2nd NSN and the THEN part of a fuzzy rule is implemented in a second neuron layer. The number of neurons in the second layer is equivalent to the number of membership functions of the output variables.

3) Defuzzification in the 3rd NSN

The output values of the system in the 3rd NSN are implemented by the standard defuzzification method with MAX-PROD inference followed by centroid calculation.

This results in a fuzzy-neuro topology like that shown in the following figure.



The system illustrated above has the input signals x_1 and x_2 . Three membership functions (N-negative, Z-zero, P-positive) are assigned to each input signal. Three membership functions (N-negative, Z-zero, P-positive) are also assumed for the output signal y . Because of its neural network structure the system is capable of learning (an advantage of neural systems) and because of its fuzzy-like topology it is possible to recreate the processing steps.

For example, a rule R_4

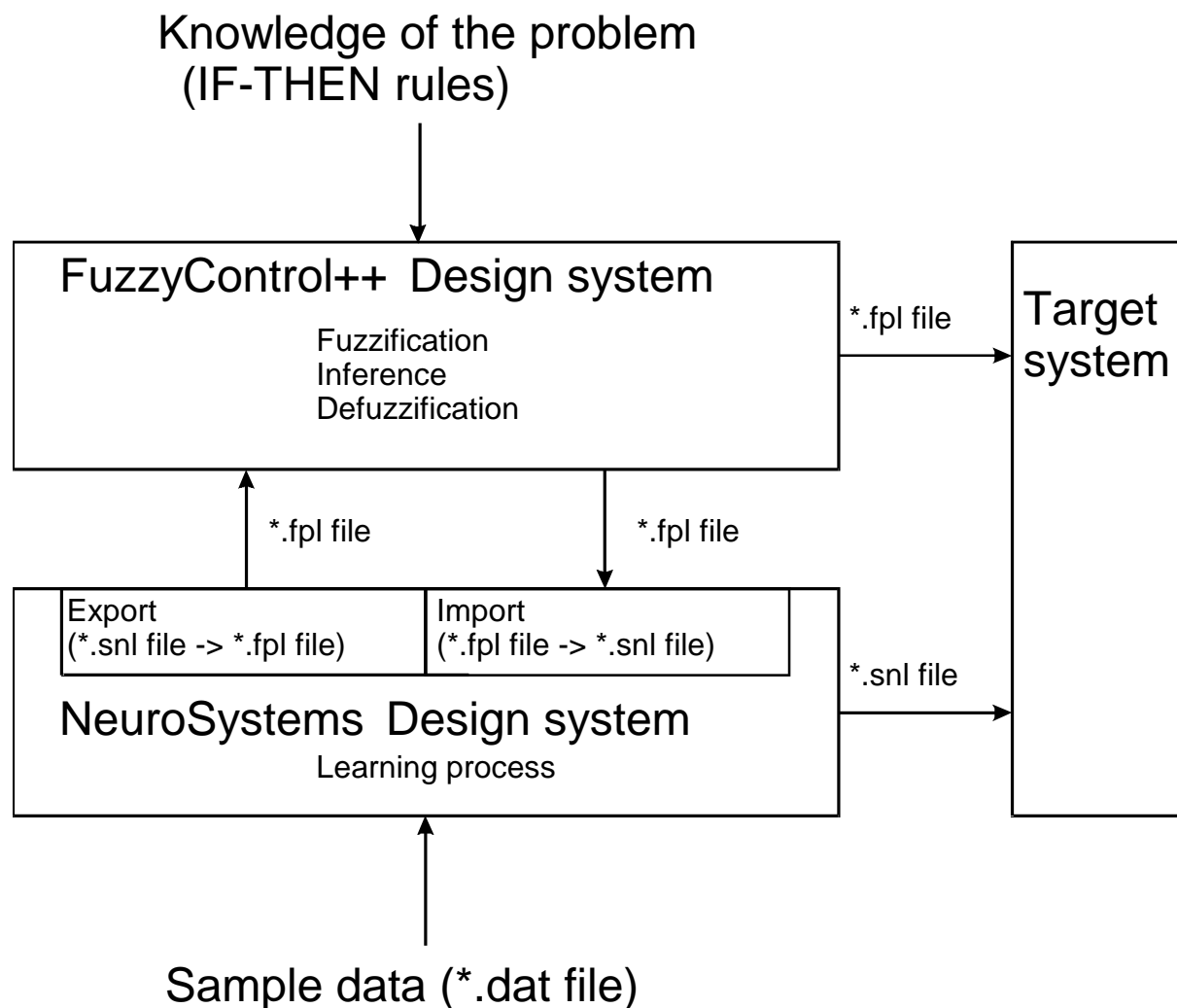
IF x_1 is equal to N AND x_2 is equal to Z, THEN y is equal to Z

causes activation of the neuron EZ_1 (x_1 ; N) and neuron EZ_5 (x_2 ; Z) in the first NSN, if the non-fuzzy values of x_1 are within membership function "N" and x_2 within membership function "Z". In the second NSN, the neuron R_4 can form the AND combination expressed in the rule because both the inputs are active and therefore activate neuron AZ_2 , which invokes the THEN condition $y = Z$. Defuzzification leading to a non-fuzzy output value y is performed by quotient calculation (6th layer) via the "moment" neuron (M) and the "area" neuron (A) in the 5th layer.

With this defined structure it is only possible to vary the numbers of membership functions with respect to the input and output signals. In *NEUROSYSTEMS*, therefore, you are only prompted to enter these quantities when defining the network structure *NFN*.



You can record the results of your neural or fuzzy structure development in an *.snl or *.fpl file. You can only transform the parameters from one system to another in the program *NEUROSYSTEMS* using the menu commands *FILE / EXPORT* or *FILE / IMPORT*. The following diagram illustrates how the two programs *FUZZYCONTROL++* and *NEUROSYSTEMS* work together as a "neurofuzzy system".





1.3.3 Applications

For a closed neurofuzzy system, consisting of parts from *NEUROSYSTEMS* and *FUZZYCONTROL++*, two typical applications are presented that demonstrate the advantage of using the two complementary programs together:

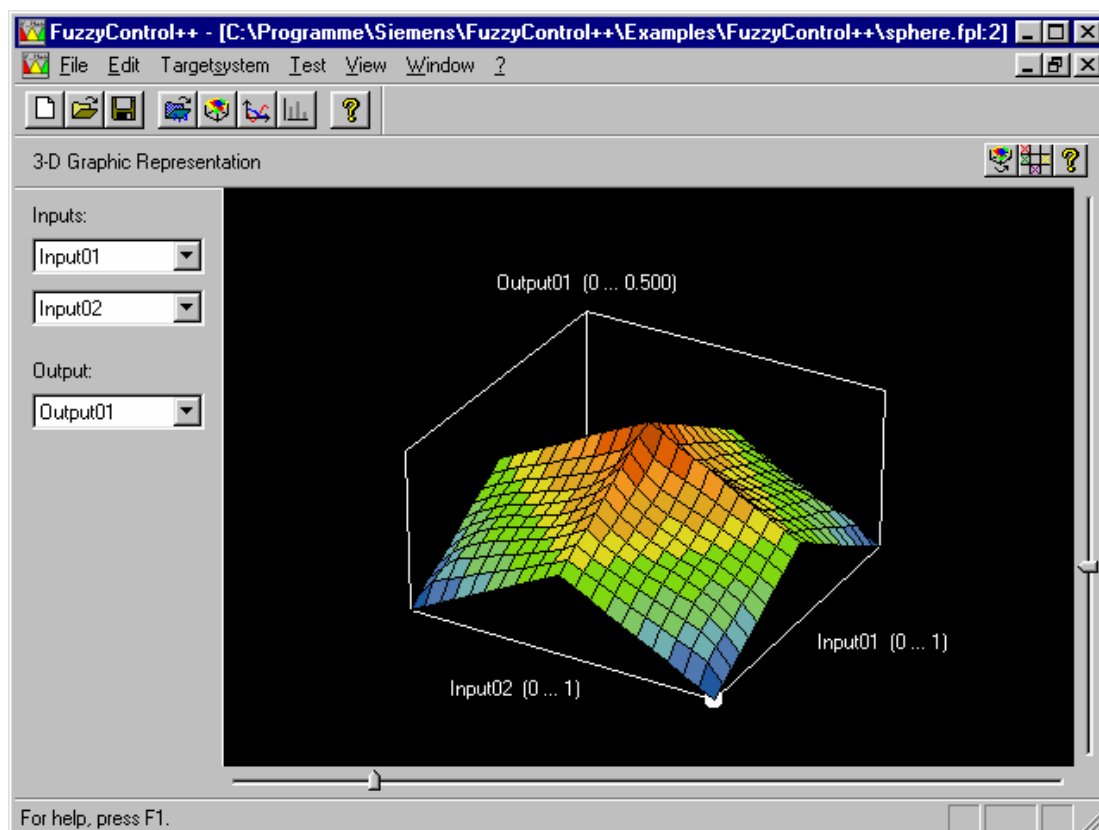
1st application example: A fuzzy model which is not yet sufficiently precise is optimized with a neural network.

The demonstration example is the visual display of a hemisphere in a Cartesian coordinate system (3-D display), where

Two input signals $x \rightarrow \text{Input 00} \quad (0 \text{ to } 1)$
 $y \rightarrow \text{Input 01} \quad (0 \text{ to } 1)$ and
One output signal $z \rightarrow \text{Output 00} \quad (0 \text{ to } 0.5)$

Form the sphere equation $x^2 + y^2 + z^2 = r^2$ (radius $r = 0.5$).

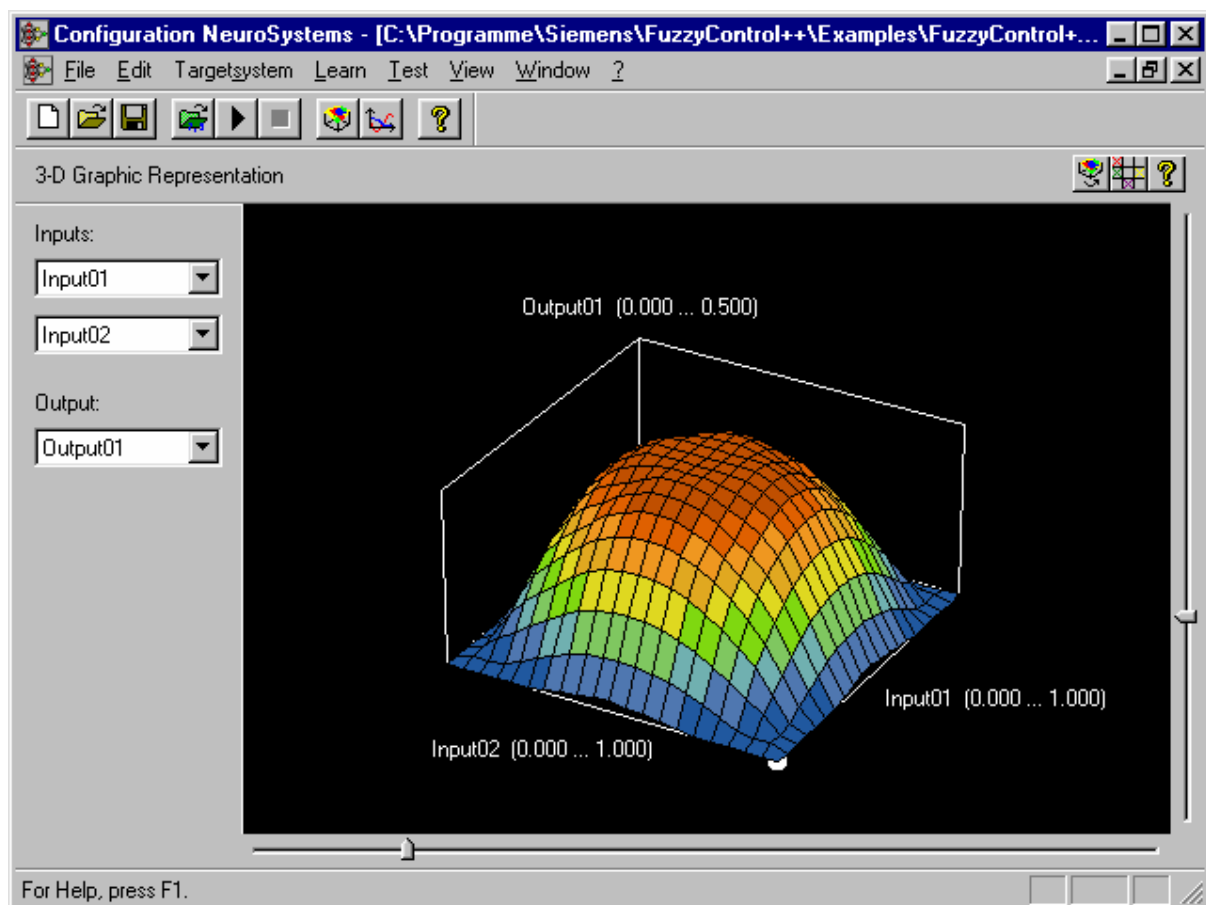
With the fuzzy rule base you can define a few points unambiguously, e.g. the center of the sphere, the corner points of the x-y surface. Overall descriptions as a fuzzy model by membership function and rule base is difficult, as the following diagram shows.



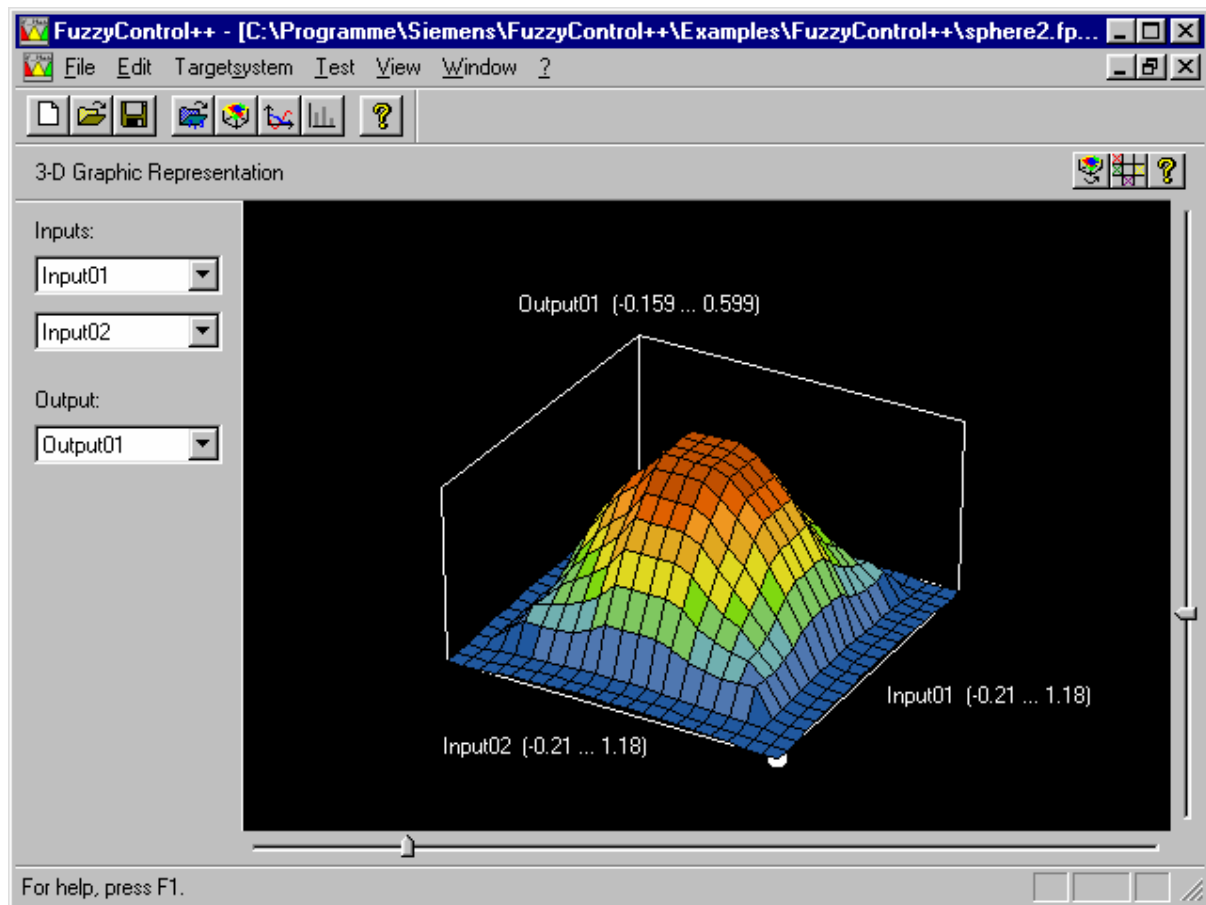


With symmetrical formation of the membership functions, as shown in this model (stored in the file "sphere.fpl"), you inevitably obtain an imprecise image of this hemisphere.

By conversion of the fuzzy parameters into those of a neural network and after a learning process using the learning data from the "sphere.dat" file (calculated by the sphere equation), the result is an improved model. (See next page for figure.)



Transformation back to the fuzzy system allows you to look at the rule base and the shape of the membership functions. This provides you with more information about the process for process identification.



Because the membership functions are linear, a visual representation of the fuzzy model appears to be somewhat more "straight-edged" than the neural network in this example.

If you vary the parameters (number of membership functions) systematically and run learning processes, you can achieve further optimization.



2nd application example: A process about which too little knowledge exists for a fuzzy model but for which series of measured data is available.

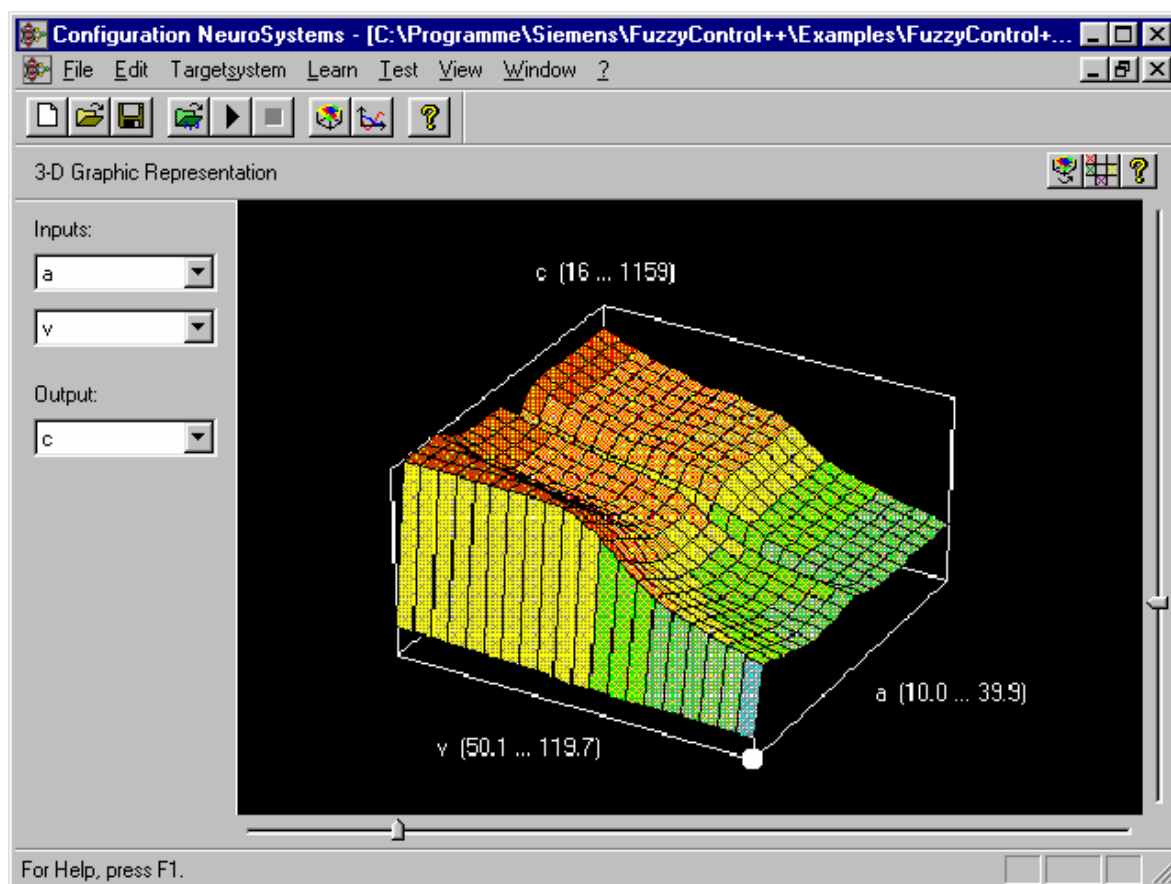
Let us take the example of coat thickness control to demonstrate the conversion process from a neural network to fuzzy model. The following measured data is available from the process

Three input signals v - Tape velocity
 p - Air pressure
 a - Distance between the nozzle and the tape

One output signal c - Coat layer thickness

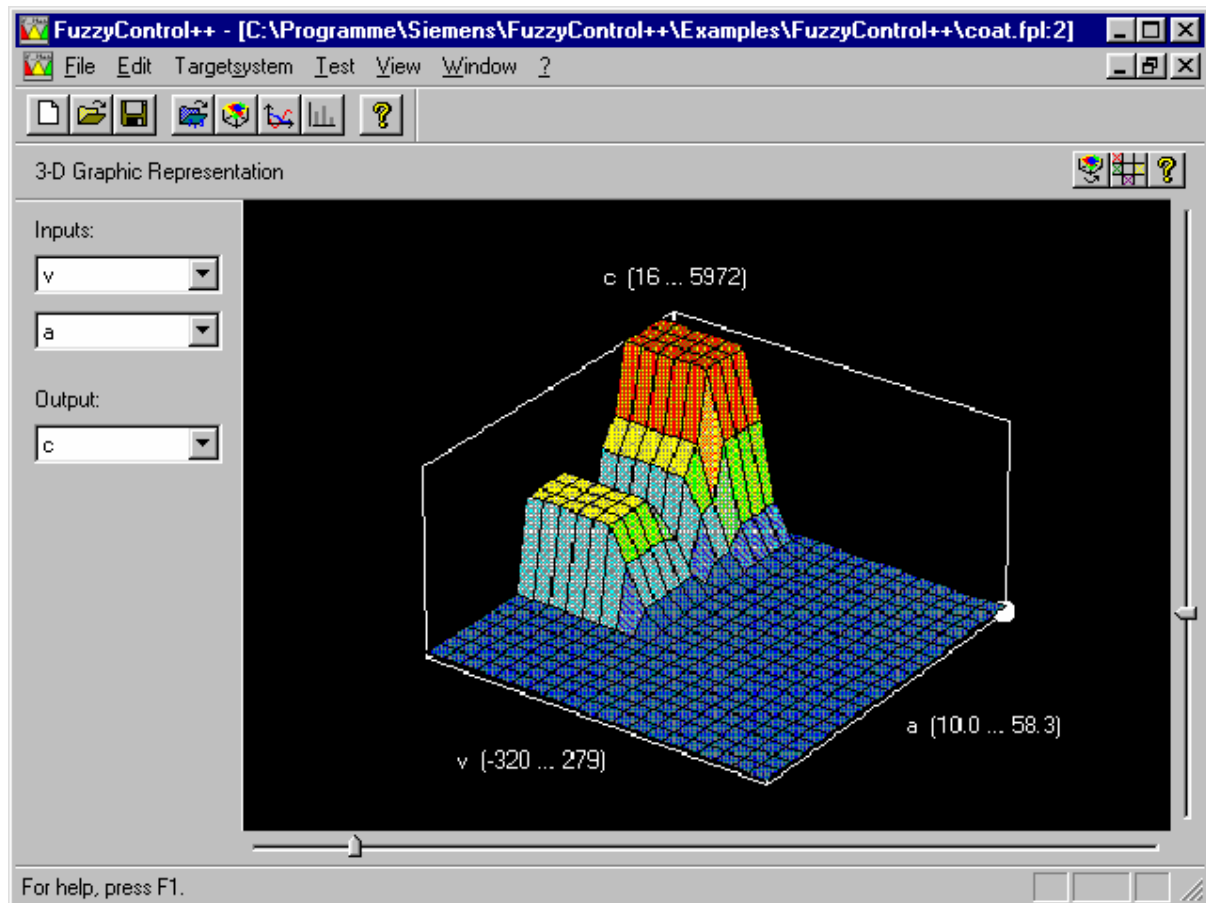
in the file "coat.dat". Because the technical and technological conditions are complicated, it is very difficult, if not completely impossible, to establish a fuzzy model.

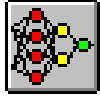
Training a neural network would seem the obvious solution here. The freely selectable parameters are the number of membership functions or the number of activation functions and therefore the number of neurons in the 2nd layer of the 1st NSN. The result of the learning process is shown in the following figure:





After conversion of the data file *.snl \rightarrow *.fpl and transfer to the *FUZZYCONTROL++*, you can see the visual representation (following figure) and look at the rule base and the membership functions. In this way, you can also obtain information about the process which you could not derive directly from the series of measurements.





2 First Steps

This part of the manual provides a short introduction in *NEUROSYSTEMS*.

2.1 Installation

The *NEUROSYSTEMS* projecting tool consists of a Windows program (configuration tool) and some Runtime-Modules (function- and data blocks for S7-300, S7-400, a runtime library for WinCC, ActiveX Control, ...). You generate and project the neuro systems using the configuration tool. At runtime, the runtime modules calculate these systems on the selected targetsystem.

There are runtime-modules for:

- SIMATIC S7
- SIMATIC CFC (optional)
- SIMATIC WinCC
- ActiveX
- OPC (optional)

2.1.1 System Requirements

- Hardware requirements:

Configuration tool:	PC or PG with 80486 processor (better higher), At least 256 Mbytes RAM At least 25 Mbytes free hard disk space
S7 function blocks:	S7-300 with CPU 314 (or better) or S7-400 MPI Interface on a PC The PB (PROFIBUS), TPC/IP and IE (Industrial Ethernet) bus systems are optionally supported.



- **Software requirements:**

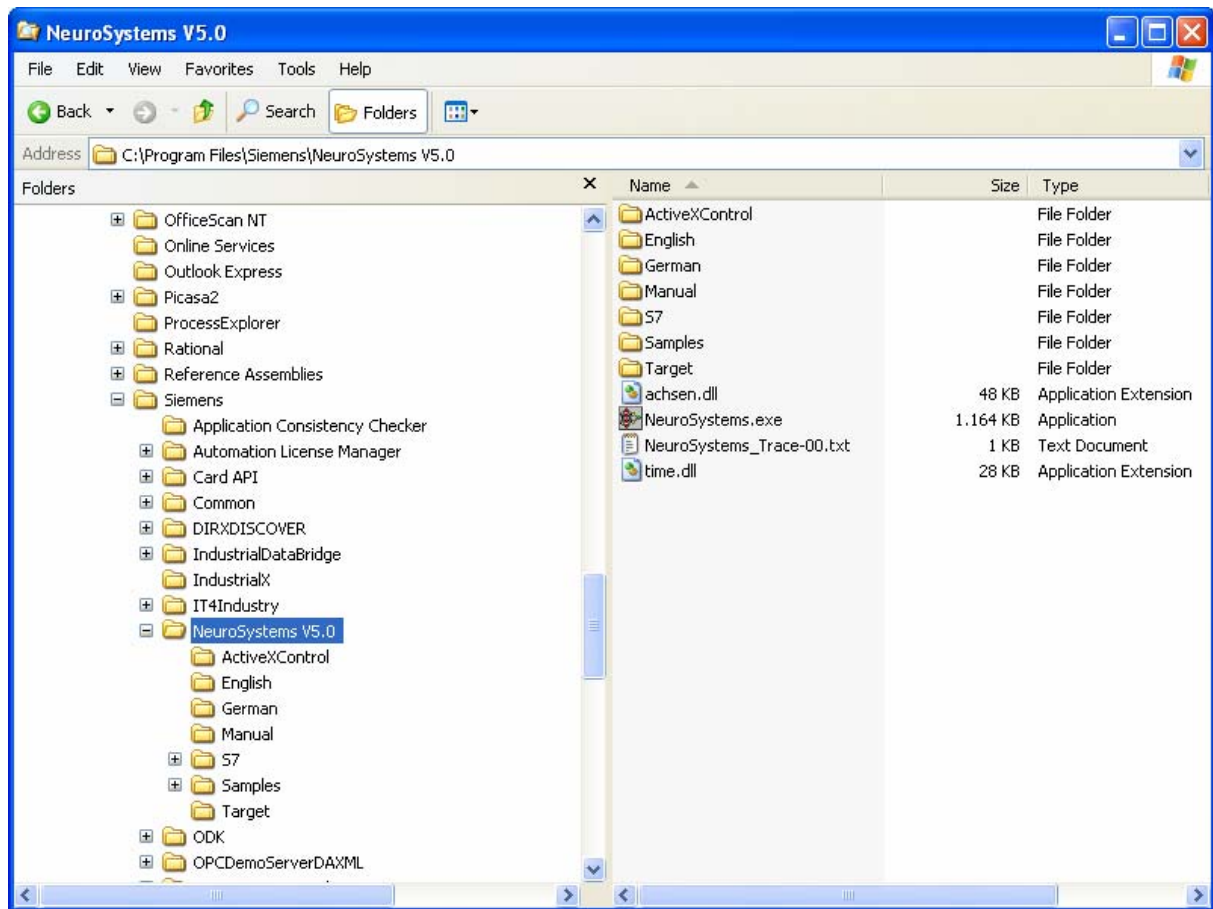
Configuration tool:	Windows 2000 or Windows XP
S7 function blocks:	SIMATIC SOFTNET S7 / PB (for linking via Softnet CPs) Optionally SIMATIC SOFTNET S7 / IE at TCP/IP or IE connection
WinCC object:	WinCC, Version 3.1 up to version 6.0
ActiveX object	SIMATIC WinCC or a different ActiveX environment

2.1.2 Installation of *NEUROSYSTEMS*

1. Insert the *NEUROSYSTEMS* Setup CD.
2. run *setup.exe*.
3. The setup program guides you through installation. Just follow the instructions on the screen.
4. To achieve optimum display quality on the screen, we recommend setting the Windows "Display Properties" to "small fonts"! The screen resolution should be at least 800x600 pixels.

After installation not only the configuration tool itself but also the following components are available for use in the relevant directories of *NEUROSYSTEMS*:

- SIMATIC S7 modules for S7 applications (**S7** Catalog)
- ActiveXControl (Catalog **ActiveX**)
- Targetsystem driver (**Target** Catalog)
- *NEUROSYSTEMS* example projects (Catalog **Samples**)
- Cookbook with applications (Catalog **Samples\Cookbook**)
- Manual (Catalog **Manual**)
- Help (**English** Catalog, **German** Catalog)



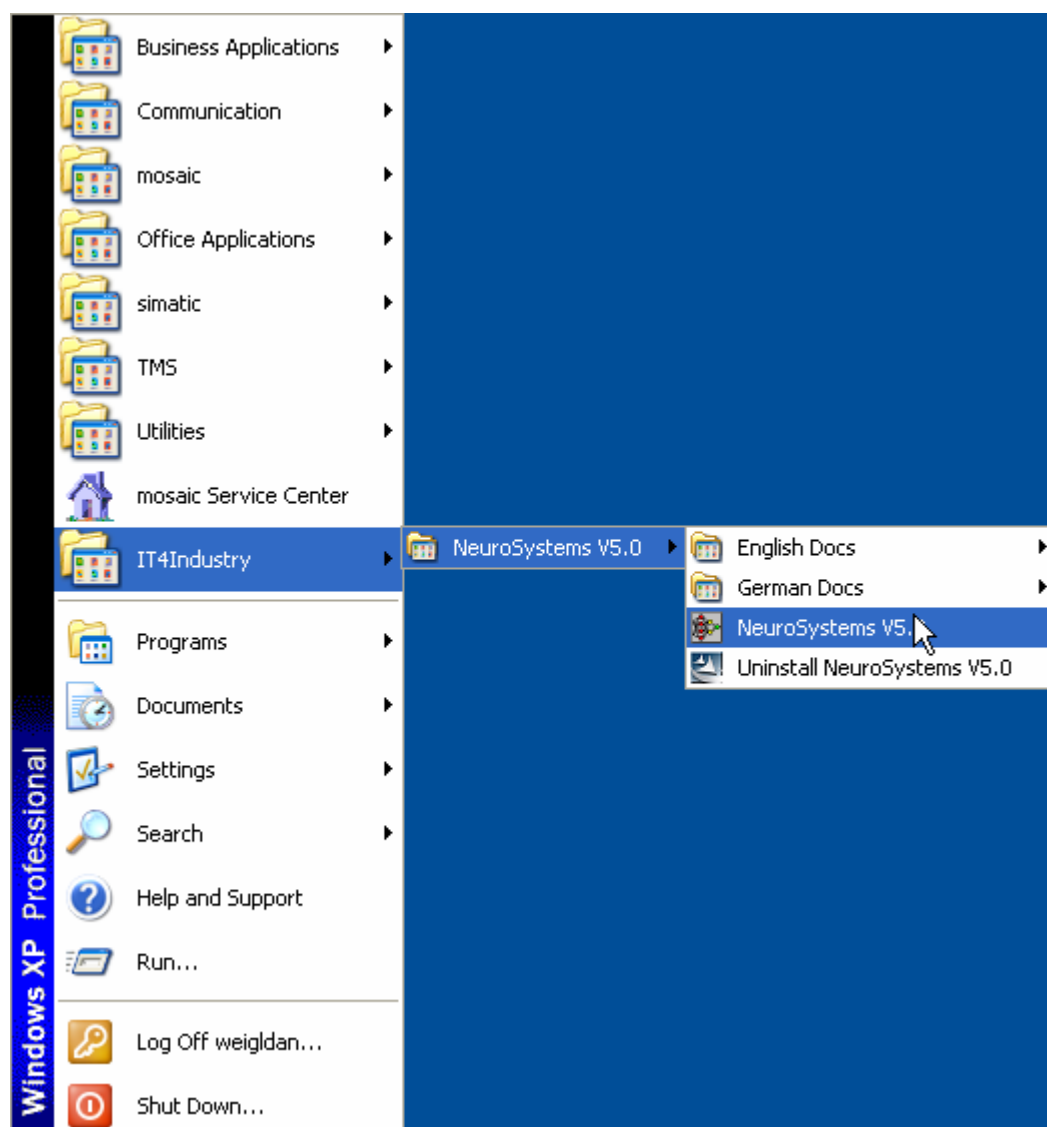
Needless to say you can uninstall the program and the installed files using the Windows utilities. Instructions can be found in the Windows help program and the Windows User Manual.



After installing *NEUROSYSTEMS* you can start via

Start->IT4Industry->NeuroSystems V5.0->NeuroSystems V5.0.

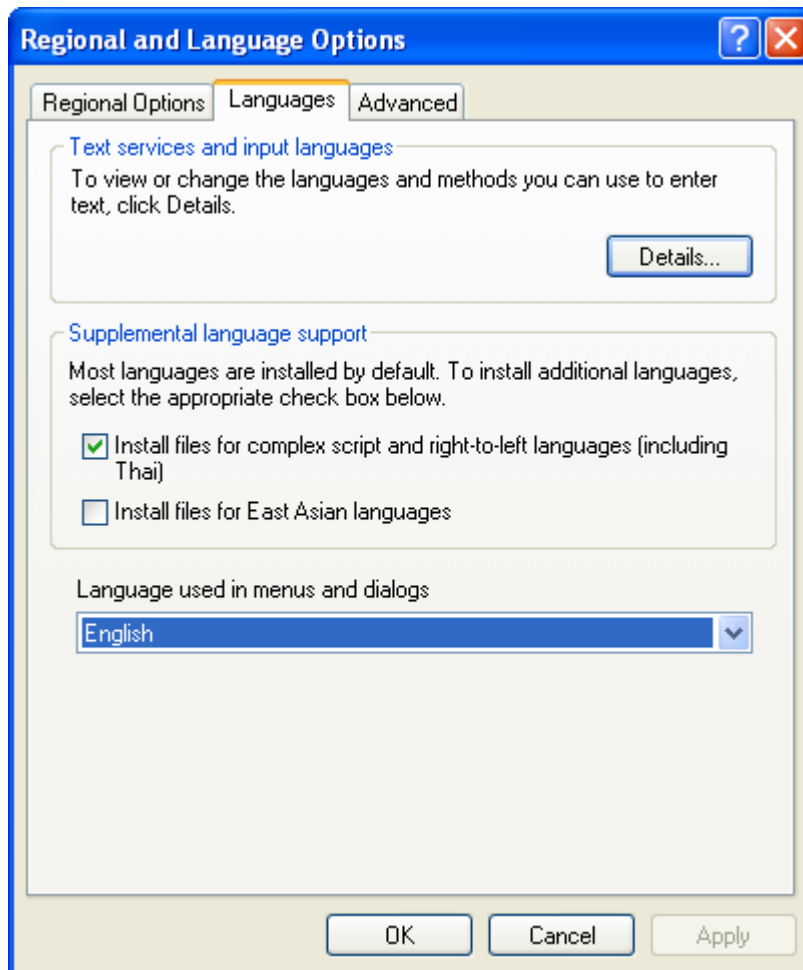
Further you can find the manuals in start as well. Further you can deinstall the program here as well.





2.1.3 Language Adjustment

The *NEUROSYSTEMS.exe* installed is bilingual. The language in the *NEUROSYSTEMS* menus and dialogs is automatically set depending on your Windows system settings (Start – Settings – Control Panel – Regional Options).





2.2 What's new in V5?

The version V5.0 of *NEUROSYSTEMS* has been completely revised, extended with many new functions and has been adapted to FuzzyControl++ V5.0. Multiple possibilities for data analysis, visualisation, testing and validation of neural networks have been integrated into the current version. Further you have various runtime modules as targetsystems.

Runtime Modules

- Targetsystem manager for the administration of the various targetsystems (runtime modules)
- Interface for the connection of arbitrary customer-specific targetsystems.
- Targetsystem-information during the connection.

SIMATIC S7

- The dialog “Connect Targetsystem“ to SIMATIC S7 has been completely revised.
- Connections via further networks PB (PROFIBUS), IE (Industrial Ethernet ISO) and TCP/IP (options)

ActiveX

- New runtime module as ActiveX control for the integration into containers
- Debonding of WinCC Neuro OLL

OPC

- Connection of in- and outputs via OPC (option)
- Possibility of browsing for server and tags and random selection and assignment of the tags to the in- and outputs of the neural network
- Reconnect in case of server breakdown

Data analysis

- Possibility of standardisation of the in- and outputs by means of the learning file
- Display of distribution of the learning files with indication of the average and variance
- Display of combination to visualize the relations between in- and output signals
- Input relevancy of the inputs
- Possibility of multiple zooming in most windows for exact data analysis



Test

- The curve plotter with dialogs have been revised. Arbitrary signal and color choice and visualisation of up to 110 signals.
- Zoomable error- and signal display.
- Display of error overview for a quick “to be – is” comparison.
- 3-D scaling dialog within the 3D-graphic

Miscellaneous

- Context menu in dialogs
- Project- and network specific text input for in-/outputs and network.
- Project specific saving of important setting.
- Manual completely revised
- Help revised
- Trace possibility



2.3 Creating a Network

To solve a problem with *NEUROSYSTEMS*, a *project* has to be drawn up, representing this problem. A *project* is a neural network, specified by the choice of a targetsystem, its inputs and outputs, its network type and its structure. To process a project, you can either use an existing project or create a new project. For an existing project, you can load its file with the extension **.snl* (*SIEMENS NEURO LANGUAGE*) with *Open...* in the menu *File*. A new project is automatically given the project name "New". You can give the project another name the first time you save the project (*File/Save as...*).

Let us take a very simple example: Suppose a controller has to execute the (binary) logic exclusive OR operation, i.e. the XOR function (exclusive OR function, non-equivalence function), for two input signals as its input/output characteristic. It is to be implemented in a neural network, and a program for this purpose is to be included in the controller. With the *NEUROSYSTEMS* tool, you can create the required neural network. The learning data for this problem can be taken from the truth table of the XOR function:

Tags:	Inputs		Outputs
	I0	I1	O0
Values:	0	0	0
	0	1	1
	1	0	1
	1	1	0

The following quick instructions gives an introductory explanation about processing a project. You can work through a typical project if you perform the indented sections marked **Example** with the *NEUROSYSTEMS* tool on the computer. The four learning data sets from above (XOR function) are used for the training of the neural network.

Attention: The emphasis here is not on solving a practical (complex) problem but on explaining the principle, using an example which is easy to understand!



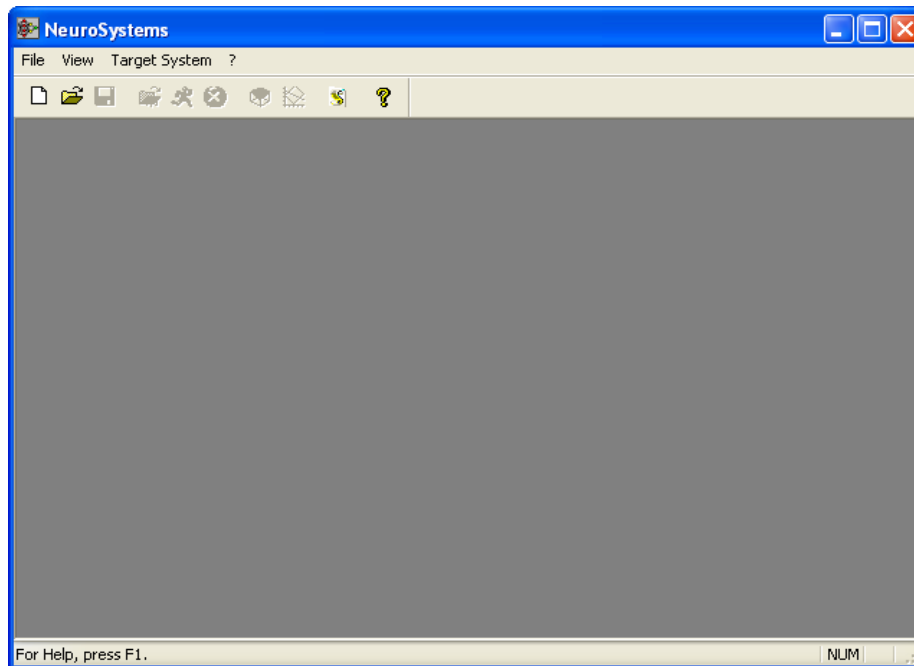
2.3.1 Configuration of the Network

Configuration of the network includes the selecting of the targetsystem and of the number of network inputs and outputs. Normalization must also be prepared by specifying minimum and maximum values for the inputs and outputs.

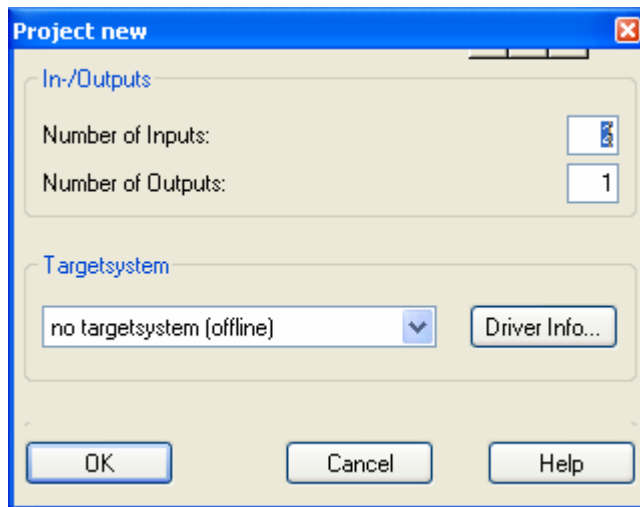
2.3.1.1 Defining the Number of Inputs and Outputs and the Targetsystem

Example:

After you have started *NEUROSYSTEMS*, the basic window *NeuroSystems* appears. This is the working window of *NEUROSYSTEMS*. The toolbar (above) and the status bar (below) are displayed throughout your work in all the windows and assist you with operation.



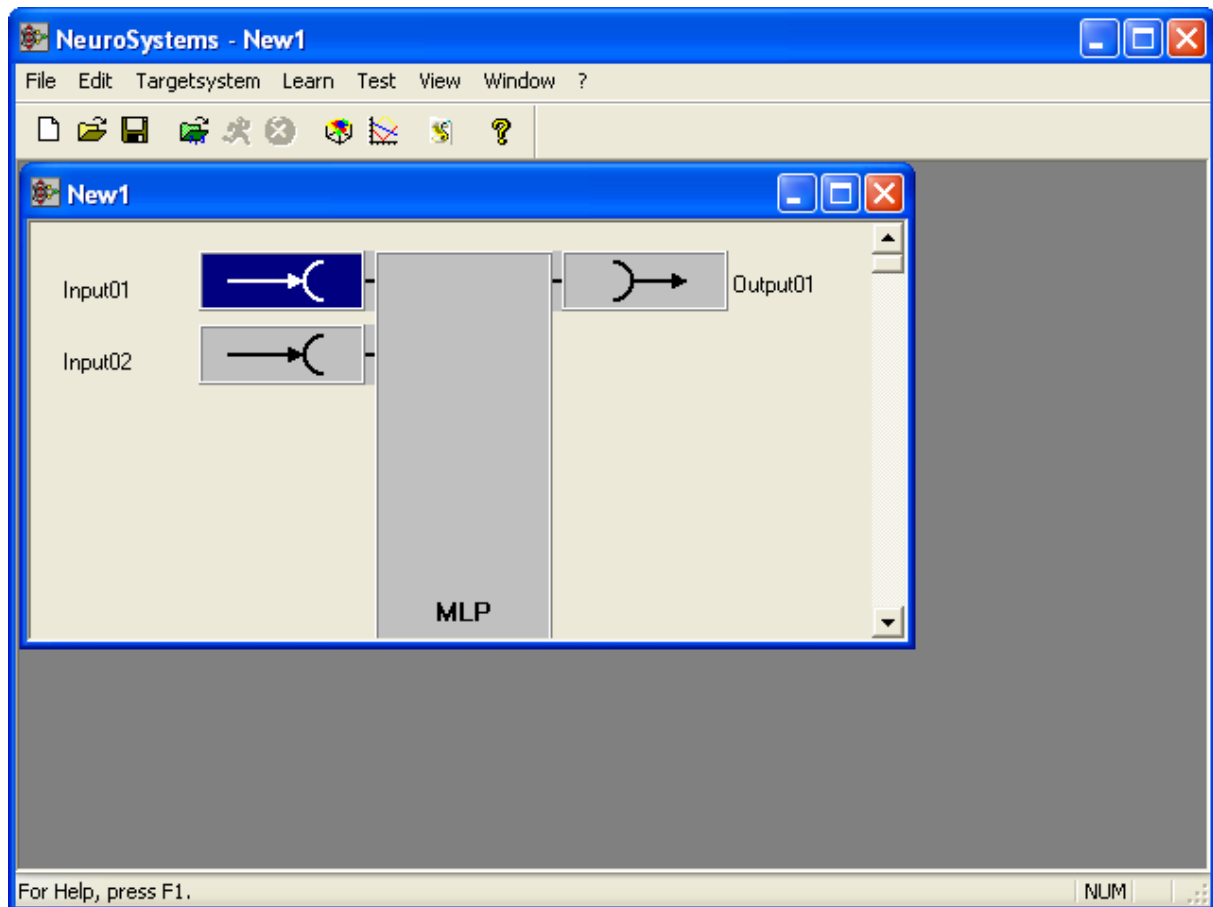
Execute the command *New* in the *File* menu displayed in the window above. First of all, a dialog box with the designation *Project New...* in the title bar appears, in which you can enter the external network structure.



The default number of inputs and outputs displayed are the same as those needed for the XOR example chosen. As the targetsystem, we could enter SIMATIC S7-4K, for example. If you now click OK, the setting of the external network structure is terminated. The project is now given the name *New1*. You do not have to set the number of inputs and outputs definitively, but can add or remove inputs and outputs later on with the *Edit* menu.

Note: You will see that once you have clicked *OK*, all the menus are available in the working window.

Example: After you have completed the external configuration, a window with a block diagram of the neural network is displayed. This window forms the basis for editing the project concerned (project window). It shows a symbolic representation of the network and its inputs and outputs (in this example the default number of inputs and outputs). In the network block you can see the default setting, which is an MLP network. The initial project name *New1* is used for the project window.

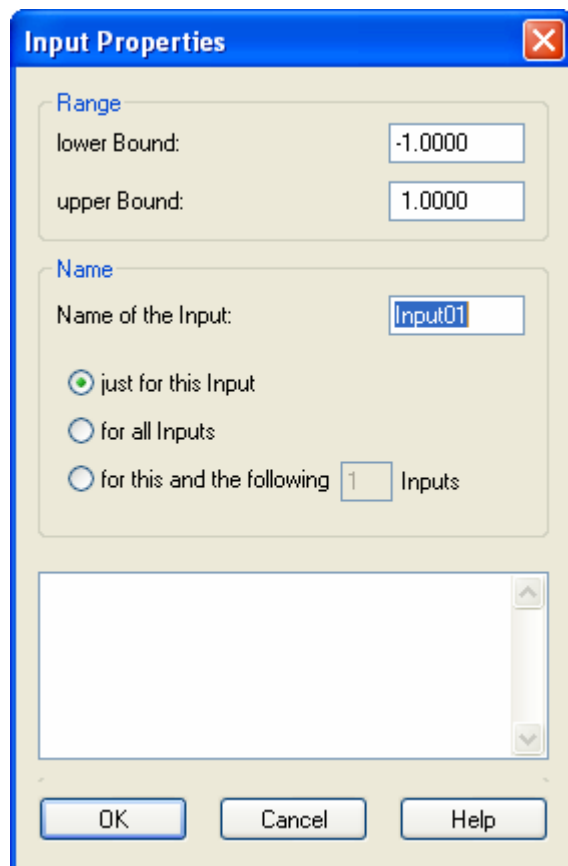


2.3.1.2 Editing the Inputs and Outputs

In further processing of the project, you can assign names to the inputs and outputs which are apt for your particular problem. However, in our example, we shall leave the default names unchanged. It is also necessary to normalize the inputs and outputs. To do this you must specify the minimum and maximum which are the lower and upper limits for the input and output signals. It is also possible to use the lowest and highest values for the corresponding input/output taken from of the learning data sets.

Example: Editing Input01:

We are in the project window *New1*. If you now double-click with the left mouse button on the upper input button with the default name *Input01*, the *Input Properties* dialog box for this input is displayed. The name appears in the enter box, where you can change it if you want to.



The dialog box is titled "Input Properties" and has a close button (X) in the top right corner. It is divided into two main sections: "Range" and "Name".

Range Section:

- lower Bound: -1.0000
- upper Bound: 1.0000

Name Section:

- Name of the Input: Input01
- Radio buttons for selection:
 - ☒ just for this Input
 - ☐ for all Inputs
 - ☐ for this and the following 1 Inputs

At the bottom of the dialog box are three buttons: OK, Cancel, and Help.

In the upper part of the dialog the current range is displayed. Here you can set the minimum and the maximum manually. The current name of the input appears in the input box in the lower part of the dialog. Here you can change the name and it can be adopted for the chosen or the following input. If you are done processing the input, confirm with OK.

- **Editing Input01:**

If you double-click the button of this input with the left mouse button, you obtain the *Input Properties* dialog box for this input, indicated by the input number 01. You can see that the minimum and maximum values have already been entered because we have had them applied to this input too.

- **Editing Output00:**

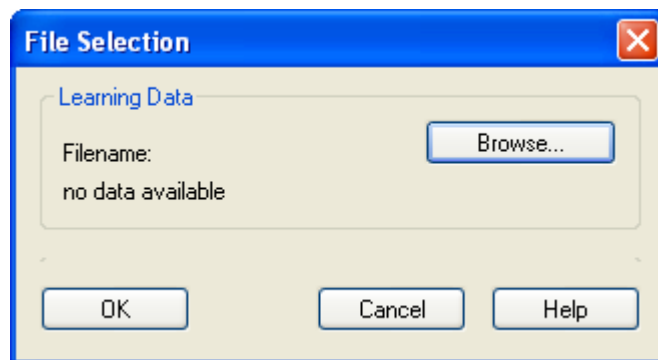
If you double-click on the output button with the left mouse button, you obtain the *Output Properties* dialog box for that output, indicated by the output number 01. Editing outputs is equivalent to editing inputs. By closing the window "output properties", you stop the editing process. The block diagram of the network (project window) is displayed again.



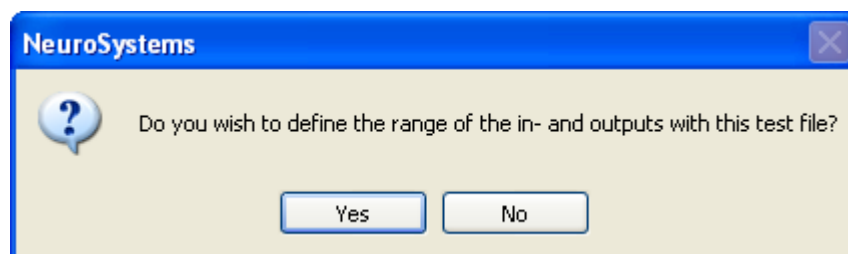
2.3.1.3 Determining the range with the learning data file

You should always carry out the determination of the range, because it has effects on the learning success when you train the network. It determines the "learning region" of the neural network. The chosen min-max interval indirectly allows a "weighting" of inputs and also affects graphic diagrams (scaling of the axes of 3D/4D representations, for example)

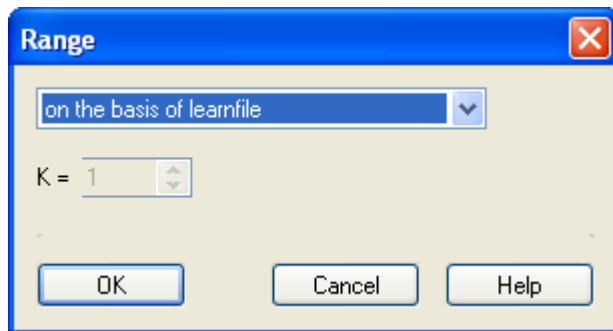
Example: To let the program determine the range of the in- and outputs with the available XOR example file, you must select the learning file first. For this purpose click on "Learn" and then "File Selection" in the menu and the accordant window will open.



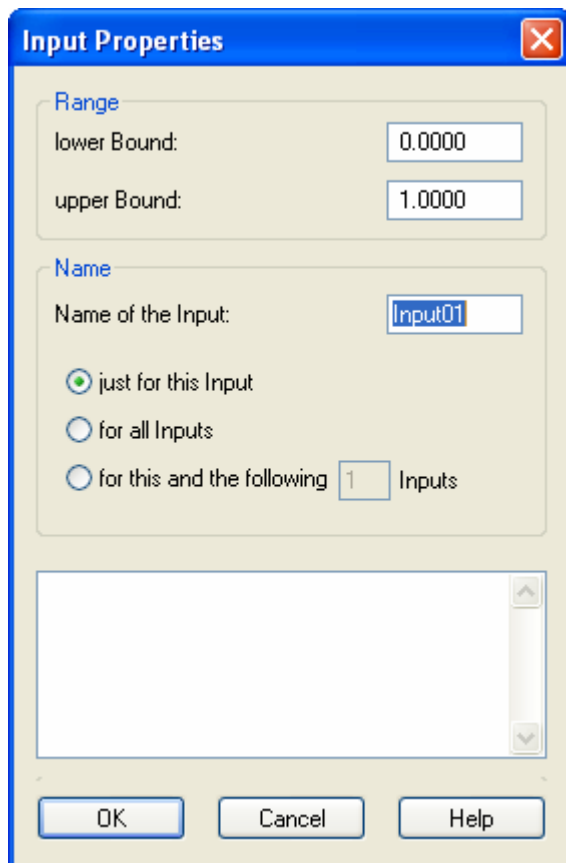
If you click *Browse...*, a dialog appears in which you can find and select the learning file *xor.dat* that corresponds with the example (supplied by the program in the subfolder *Samples/XOR*) which contains the four XOR learning data sets to be learned. You can open it by clicking the *Open* button and a window will appear with question, whether the range should be defined with this test file.



Confirm with "yes" and the "Range" window will open. Set the "Range" to "on the basis of learn file" and confirm with OK.



In the window *Input Properties*, you will now see that the value 0.0000 is entered as the minimum and 1.0000 as the maximum for Input01.





2.3.2 Type of Network

2.3.2.1 *Selecting the type of network*

When you define a new project, an MLP network (multilayer perceptron) with a default network structure is automatically created. However, you can change the type of network and the network structure as you process the project.

NEUROSYSTEMS provides the following types of network:

- MLP (multilayer perceptron network)

This type of network is suitable if only **relatively little learning data** is available.

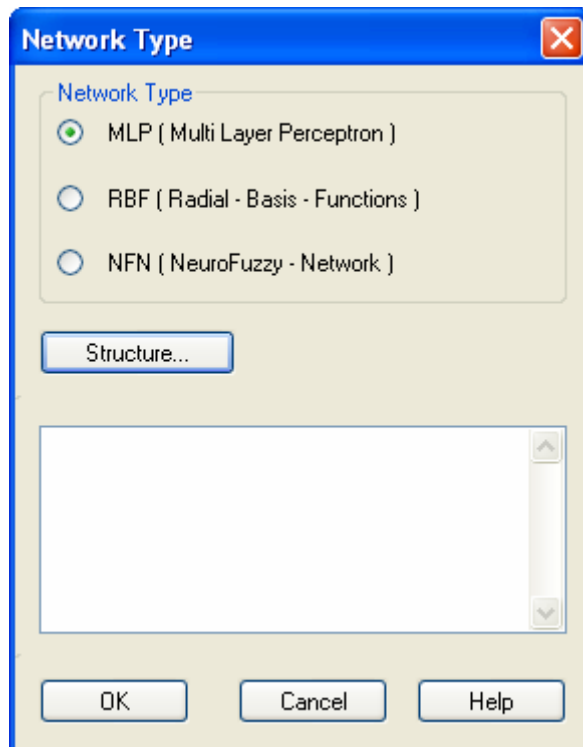
- RBF network (Radial basis function network)

This type of network should be used if **a lot of learning data** is available.

- NFN (neurofuzzy network)

Networks of this type process the membership functions and rule base of a fuzzy system.

If you double-click the network block in the project window, the dialog box for selecting the type of network is opened.



2.3.2.2 Defining the network structure

The network structure is defined by the number of layers and the number of neurons in the layers (for MLP and RBF), or the number of the membership functions of the inputs and outputs (for NFN). You can alter the structure of the selected network type yourself and adapt it to your requirements to a great extent.

Because the input and output layers connect the network to the outside world, the number of neurons in these layers is necessarily the same as the number of inputs and outputs defined in the project.

The number of hidden layers can be selected depending on the type of network. For this MLP network you can select one or two layers. To each hidden layer, you can assign 1 to 50 neurons.

For selection of the network structure the following information is generally useful:

- The less learning data available, the fewer neurons you should select.
- Start by using too few neurons rather than too many and increase the number until you obtain a satisfactory result.

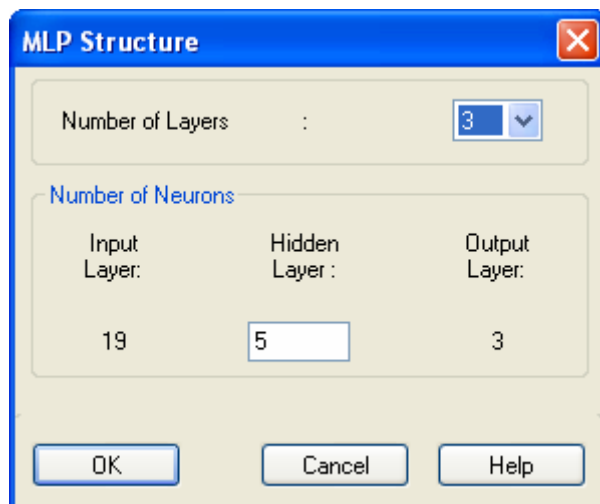


- As the complexity of a function to be modeled by the neural network increases, the number of neuron layers must also increase.

Please note that the input layer is also counted for the *Number of network layers* in the program *NEUROSYSTEMS*. In the literature, it is often not counted as a layer because its neurons mainly distribute the inputs signals. So they generally work in a different way than the neurons of the other layers.

Example:

To change the default structure created when the network type was selected (in this case MLP), activate the button *Structure...* in the window *Network Type*. The dialog box *MLP Structure* is displayed. Because the XOR example has very few inputs and outputs and the learning data file includes only a little number of data sets, just one hidden layer with two neurons will be used. Change the default setting as shown in the following figure.



After you have closed the structure and type window with *OK* the block diagram of the network is displayed again.

Note: If you click on the network block in the project window with the right mouse button, you can obtain information about the current type of network from the call-up window which appears, by selecting *Properties*. You can also make changes by selecting *Network Type...* . (If changes are made, the learning results obtained so far will be lost. So it is useful to save them first.)

**Summary:**

We have created a network with two inputs and one output for our XOR project. The names of the input/output signals were selected as in the default setting. For normalization, the minimum and maximum values of the input/output signals were determined from the learning data sets. The network type (MLP) and the network structure (3 layers, 2 neurons in the hidden layer) were selected to match the problem. At this point it is a good idea to save the work you have done on the project so far.

Example:

Execute the command *Save as...* in the *File* menu and enter the filename in the window that then appears, e.g. *xor_xmpl*. After you have closed the window with *Save*, the project is saved as the file *xor_xmpl.sn1* in the current directory.



2.4 Learning Process

Training the network means executing learning steps until the actual output patterns from the network matches the output patterns, specified by the learning data sets, as closely as possible.

Each learning step includes:

- Calculating the current actual output pattern from the input pattern;
- Determining the current difference between the actual and specified output patterns;
- Comparing with the difference from the previous step;
- Changing the connection weights of the neurons in such a way that the difference between the two differences becomes smaller.

A set of associated input and output patterns is called a *learning data set* (sometimes also *learning pair*)

Here, the network must learn the behavior of a logical XOR gate using the truth table provided in the learning data file *xor.dat*. The table below explains the values given in this example set of learning data. .

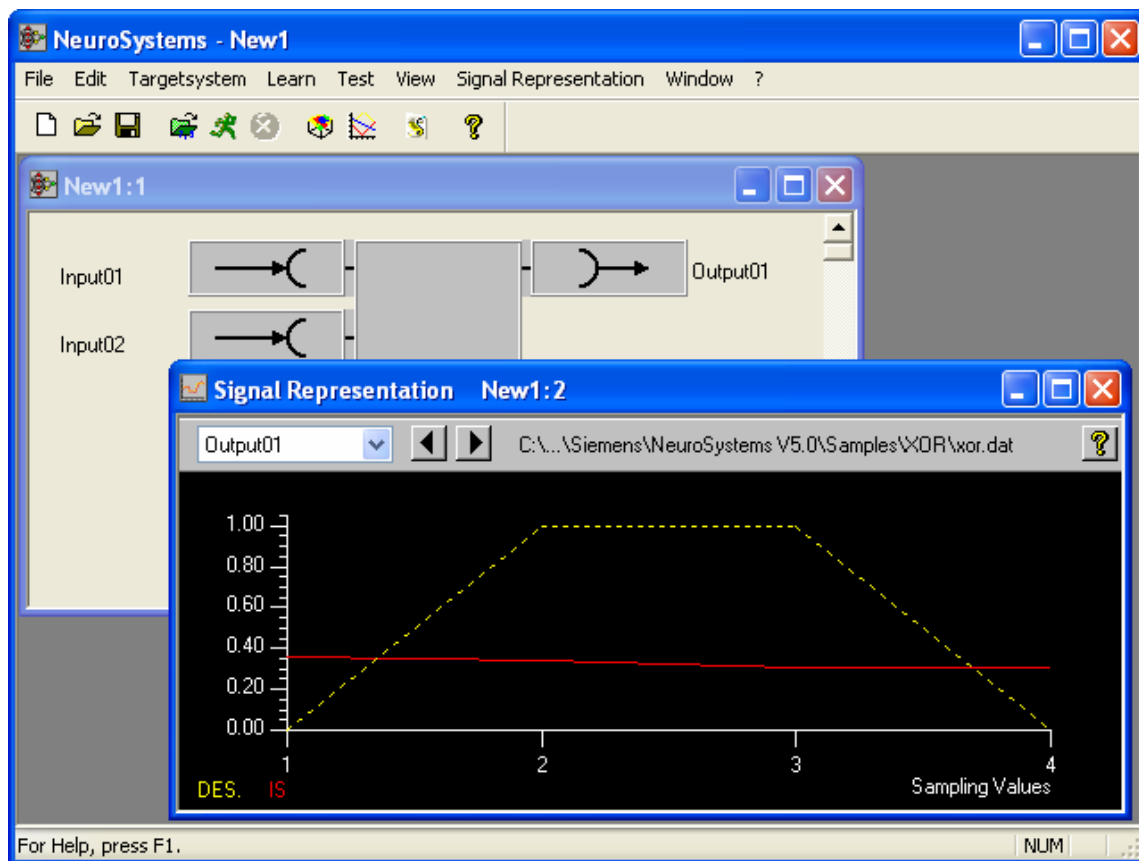
Time k ↓	Input pattern Input vector		⇒	Output pattern Output vector	
0	0	0		0	1st learning pair
1	1	0	⇒	1	2nd learning pair
2	0	1		1	3rd learning pair
3	1	1	⇒	0	4th learning pair
Designation	Input signal 1 (Input01)	Input signal 2 (Input02)	⇒	Output signal (Output01)	Learning pair, learning data set



At this point it is helpful to take a look at the learning data set graphically. Four values for each variable (input or output) can be connected by a straight line and displayed as a signal curve on the time axis. This interprets the learning data in such a way, that the four learning pairs (learning data sets) appear on the network one after the other at times k.

Example:

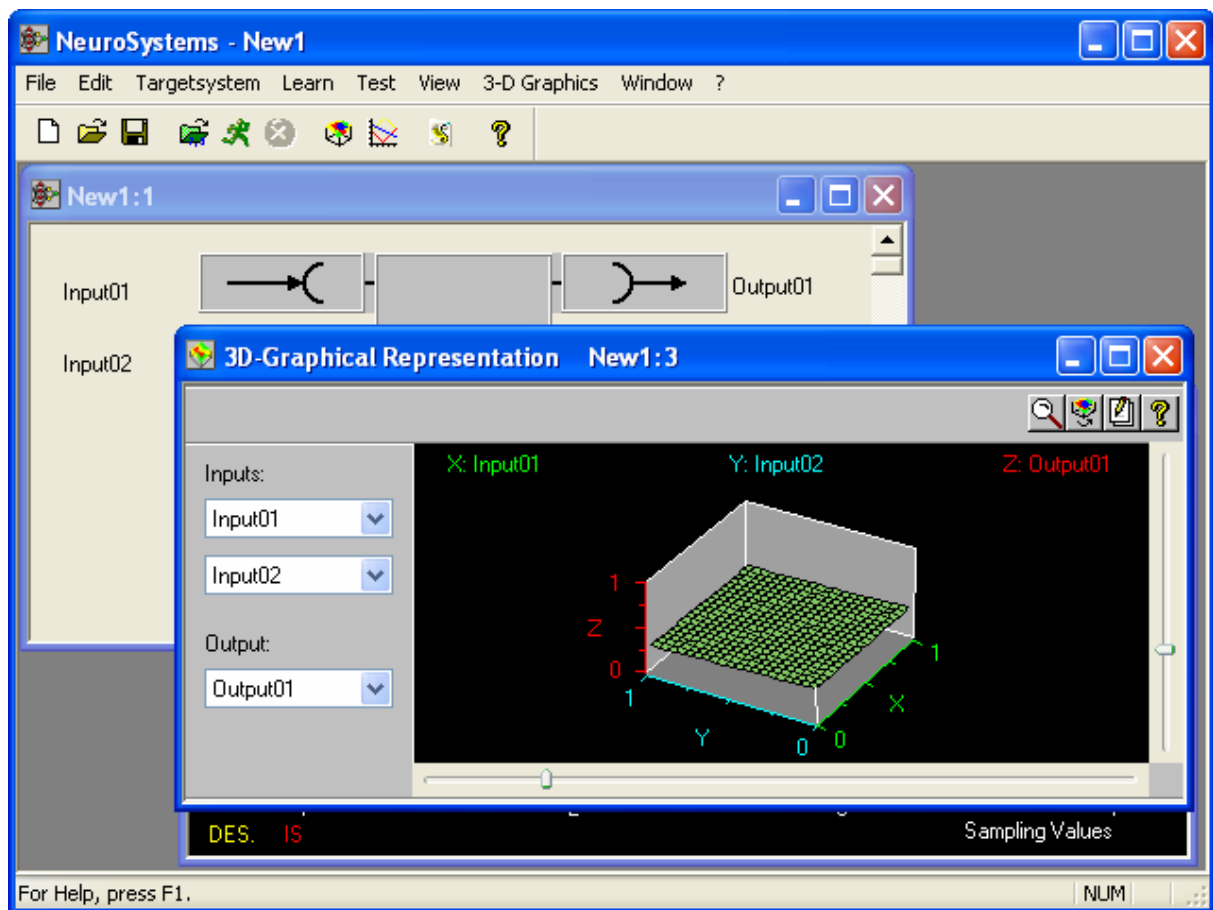
If applicable open the created project xor_bsp.sn1 and load the learning file xor.dat. Open the *Test* menu in the working window *NeuroSystems* and execute the command *Signal Representation*. The signal curve of Input01 is displayed first. You can have the other curves displayed by changing the selection field entry.





The figure above shows the output signal curve. On the screen, the **sketched yellow** curve (trapezoidal curve) is the specified output pattern and the **red** curve (almost horizontal line) is the actual output pattern of the (default) network in the current learning state. The axis *Sampling Values* shows the index k of the times at which the learning pairs appear on the network. (Because no learning process has been performed there is, of course, still a large difference between the specified and the actual output pattern.)

Now open the 3-D display, the characteristic surface, in another window by choosing the command *3-D Graphics* in the *Test* menu. In the characteristic surface, the output signal is displayed with respect to the two input signals in the horizontal plane.



The diagram shows a horizontal plane, representing the untrained state of the network, as the characteristic surface. In this example, learning data is available only for the four corners of this surface. During the learning process, the network will try to learn the output values of these four corners as good as possible. The network interpolates intermediate values and generates a complete surface.

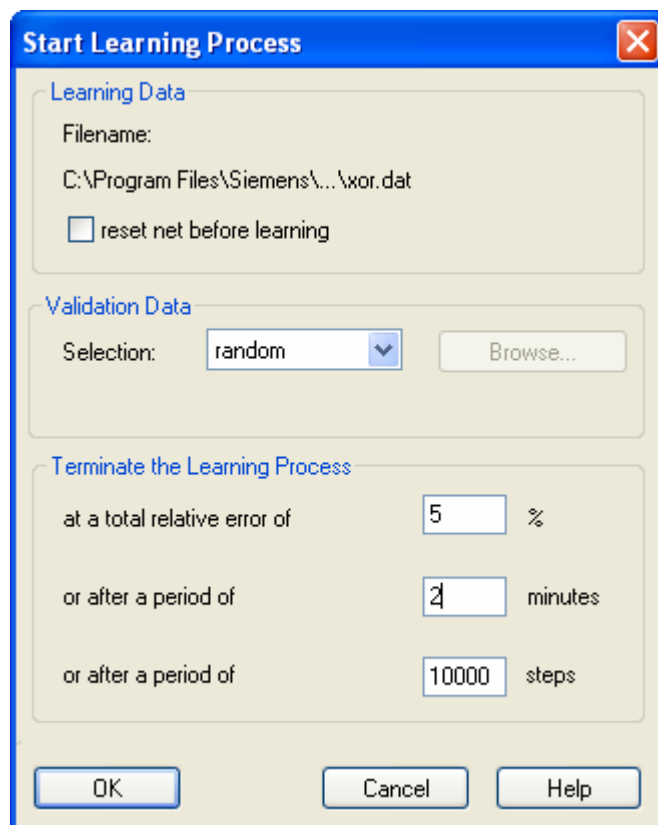


2.4.1 Starting and Stopping the Learning Process

If you have not yet loaded the learning data (for normalization purposes, for example), you have to read them with *Learn/File Selection...* (or by clicking the corresponding button from the toolbar) before you start the learning process.

Example: Starting learning:

Start learning with “*Start..*”, in the *Learn* menu. The preparations for learning are made in the following dialog box. We first select *random* for the validation data . We will select 5% as the error limit and set the learning time to 2 minutes. After clicking on *OK* the learning process begins. You can observe it in the *Error Curve* window which then opens.

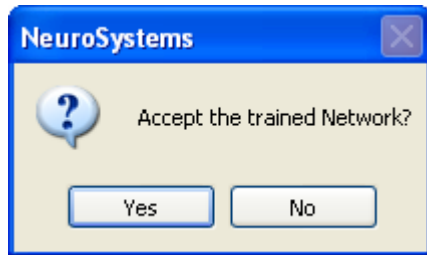




This initial learning process is unsuccessful and stops after 10000 learning steps.

Therefore interrupt the process with *Stop* in the *Learn* menu, if you do not want to wait for the set values.

Answer *No* to the query that appears after termination.

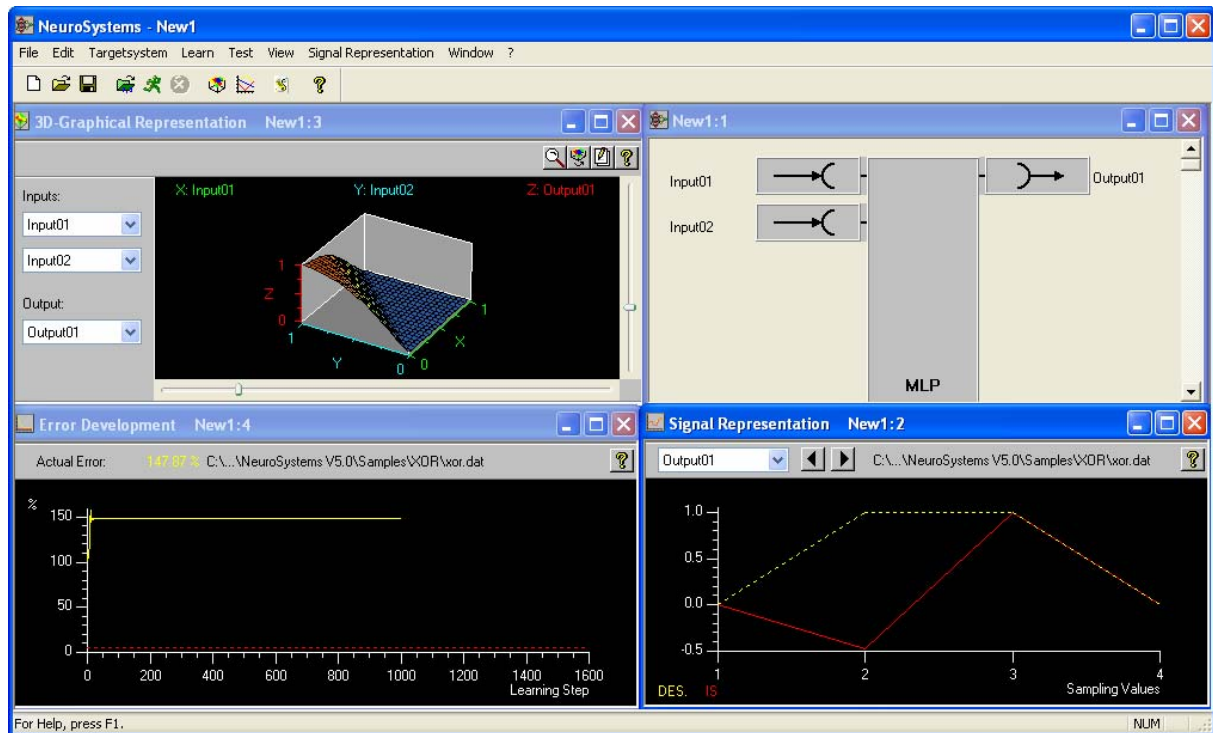


The learning process error curve does not reach the error limit. In the signal curve display, you can see that the pattern set has not been learnt at all. Learning was unsuccessful because the *random* selection of validation data (which we set) is an unsuitable training method for this network and this extremely small learning data file. In a second attempt, we shall use *no* validation data.

Note: To start and stop learning quickly you can use the buttons on the toolbar.



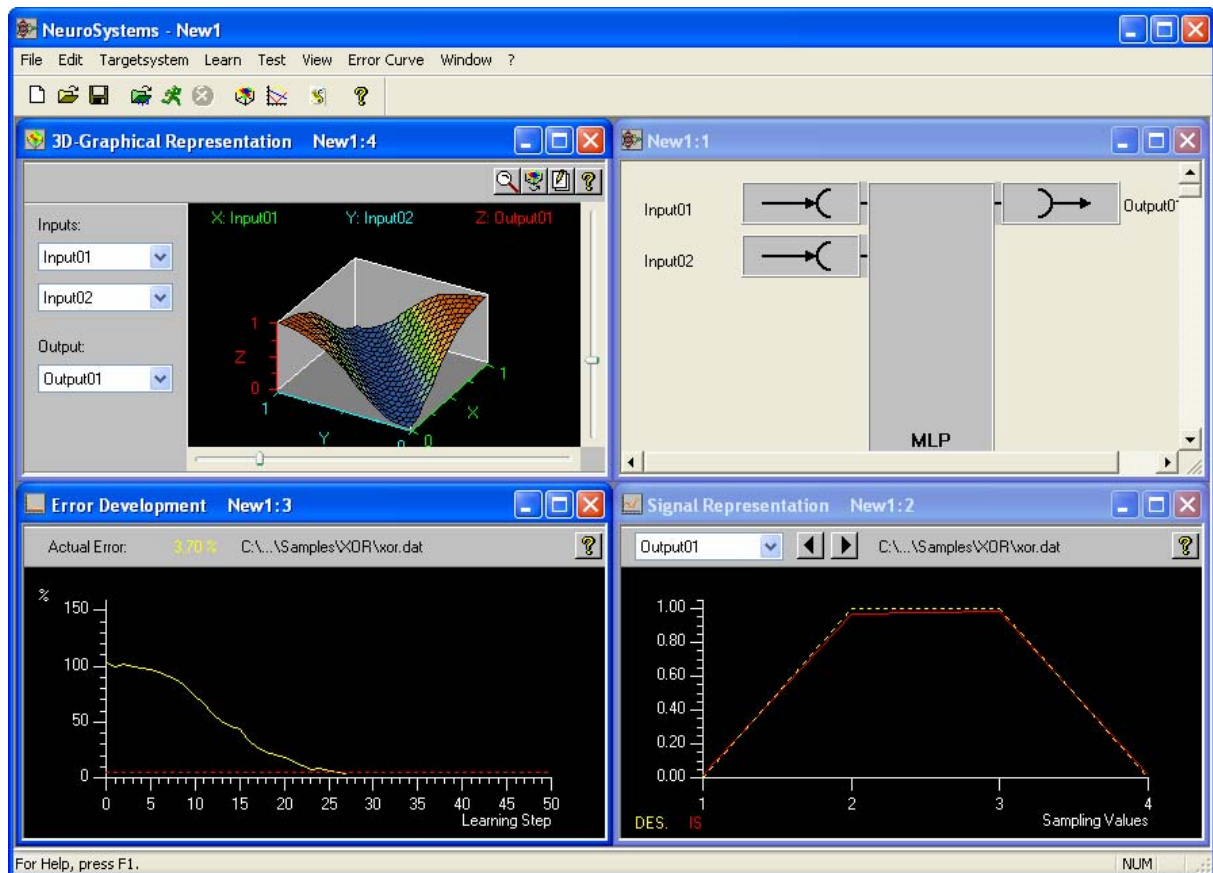
For further work on the project, it is advisable to set up simultaneous viewing of all the displays by moving the window borders with the mouse or with *Tile* in the *Window* menu, for example like this:



Note: During the learning process you can follow the changing characteristic surface. It alters with the learning progress. This enables a graphical rating of the learning success.

Example: Improving the learning process:

Start learning again and set *none* in the validation data selection box in the *Start Learning Process* window. The error limit and the learning time are set to 5% resp. 2 min., as supplied before. The learning process now reaches the error limit and is terminated successfully. The result of the learning process is satisfactory. You can answer *Yes* to the query about whether the learned network is to be saved under the name *xor_bsp.snl*, because the learning result is an improvement.



The signal representation shows a good match between the actual curve (**red** on the screen) and the specified curve (**yellow sketched** on the screen). You can rotate the 3-D diagram horizontally and vertically with the sliders on the edge of the display (by dragging them with the left mouse button) to obtain the view you require. You will notice that the corners of the input/output surface represent the XOR behavior of the system well. Looking at the diagram you are able to decide, whether you are satisfied with interpolated surface generated by the network, or whether you want to retrain the network using different parameters for "better" interpolation characteristics.

Note: Validation data are, of course, usually necessary but in this case, they reduced the learning data, of which there is little enough anyway, and they were not representative.

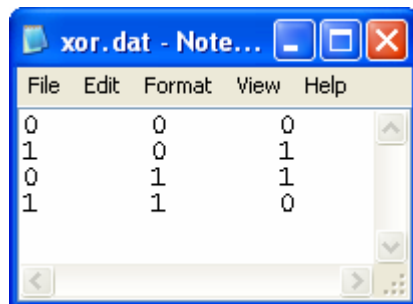


2.4.2 Editing the Learning Data File

2.4.2.1 Structure of the learning data file

A learning data file contains the numeric values of the input and output patterns in an ordered fashion. It is a text file in ASCII format and can be edited and created with any text editor. In practical use, the learning data file is usually the result of computer-assisted measured value acquisition. In the example below, the notepad of Windows is used as the editor.

Example: With the *Start/Programs/Accessories/Notepad* Windows menu command sequence you can open an empty editor window. You can then load the learning data file *xor.dat* with the command sequence *File/Open...* from the *Samples* directory of your *NEUROSYSTEMS* installation.



The first column contains the sequence number of Input01, the second Input02 and the third column the values of Output01.

A learning data file in *NEUROSYSTEMS* has the following structure in the general editor window:

- A row contains a set of input and output patterns (learning data set).
- In the row, the numbers are separated by *blanks* or the *tab* key.
- Each row is terminated with *Enter*.
- There are as many rows as there are learning data sets in the learning data file.
- The first column contains all numeric values of the first input.



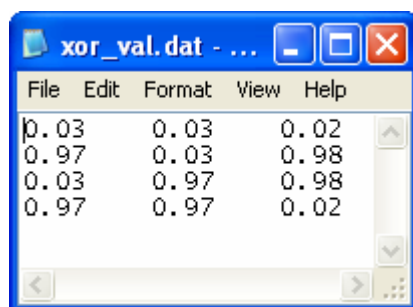
- The following columns are the remaining inputs and then the outputs in ascending order by their numbers.

Note: In a learning file, the number of columns must be equal to the sum of the number of inputs and outputs of the network in question. **Please pay attention that the learning data file does not contain "contradictory" data sets!** If you assign an input vector to different output vectors (in two or more data sets, for example), a certain rest error will remain after the learning process. In this case the network has to obey contradictory learning aims. The training result will be a compromise with an unavoidable (sometimes very large) error.

2.4.2.2 Changing the learning data file

You can modify the learning data by simply changing the numeric values. Before we deal with the problems of validation data let us create an example validation data file.

Example: Change the file *xor.dat* as follows in your editor. Please notice that you need to use a period instead of a decimal comma.



In the *File* menu of the editor we shall save the file with the name *xor_val.dat* with the command *Save as...* in the *NEUROSYSTEMS Samples* directory. You can exit the editor by closing the window.



2.4.3 Evaluating the Learning Process

If we look at the 3-D graphic representation window of the network generated (XOR example), we can summarize:

- The output values of the four learning data sets (represented by the four corners of the surface) were learned "pretty good".
- The interpolated surface between these corners possibly does not show the desired characteristic. So you probably want to have "softer" transitions or larger regions where the output stays near to "1" or "0".

In training a neural network there is generally a danger that the data to be learnt will be learnt "formally and parrot-fashion". In this case, a data set that deviates from the learnt data set only by a small amount is not recognized as being similar when the network will be executed on a targetsystem (runtime modules).

With the use of validation data you can check whether the required "global" input/output characteristic has been learnt or only a certain set of data. **The network learns using the available learning data and checks the learning aim using the validation data.**

NEUROSYSTEMS provides four different ways of selecting suitable validation data in the *Start Learning Process* window.

As the default setting NEUROSYSTEMS enters *none*. With the *random* setting 40% of the data sets to be learnt are selected at random. The network is trained with the remaining 60% of the data sets to be learnt, while validation is performed with randomly selected data sets. However, this is only suitable if you have a sufficient number of data sets available in the learning data file.

The way the validation data is selected, can influence the quality of the learning process of NEUROSYSTEMS. For example, remember how *random* selection of the validation data did not lead to success in learning the XOR behavior even after a large number of learning steps. In cases like that with too little learning data, it is advisable to pick *no* validation or a validation data *file*.

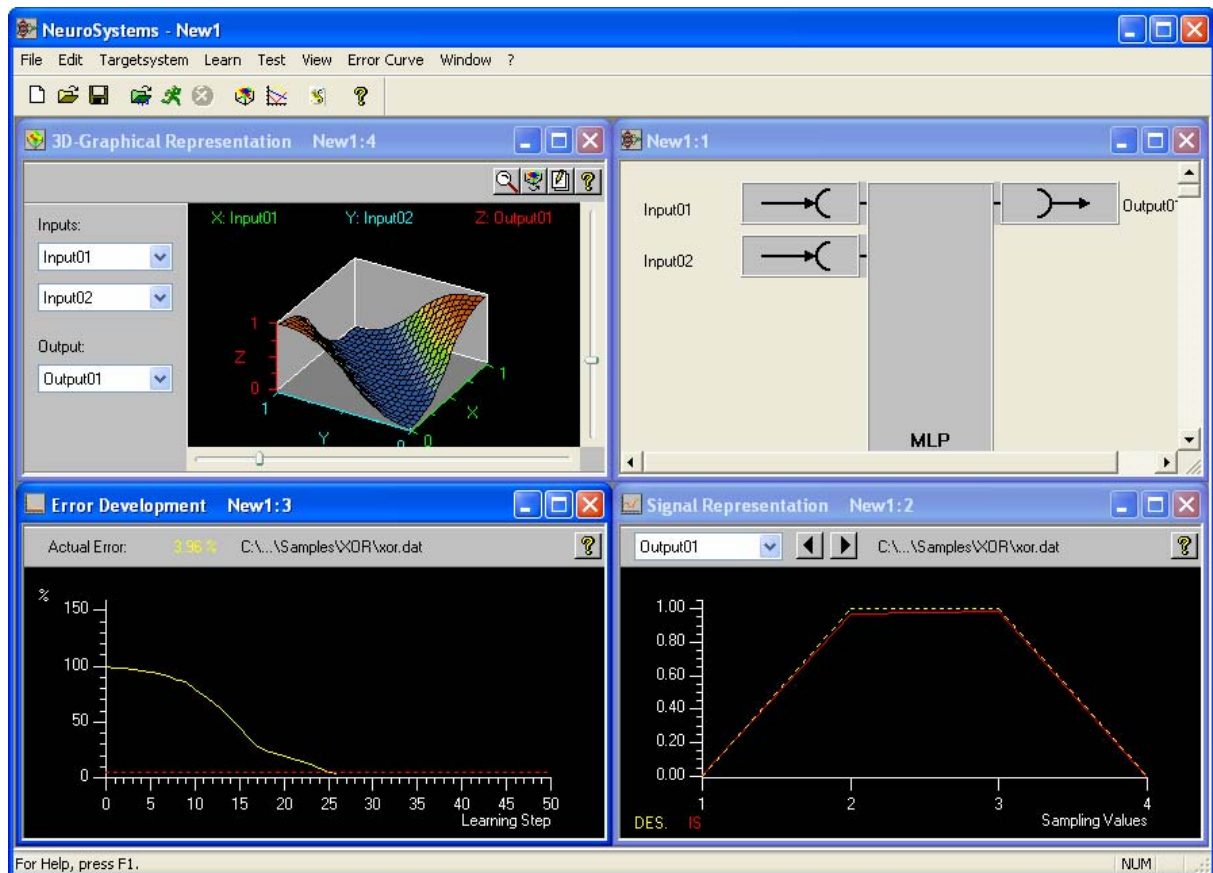
**Example:**

The current network (saved as *xor_xmpl.snl*) has been trained with the XOR data sets using no validation data. Now we want to restart from an unlearned network again and try to train the network following these requirements:

- The transitions of the characteristic surface (3-D graphic representation) should be very "soft".
- The network should be validated using the file *xor_val.dat* which we created above.
- The error of the trained network should be less than 3%.

Create a new network. As above, select an MLP network with one hidden layer. To achieve the soft transitions raise the number of neurons of the hidden layer from 2 to 5. The ranges are determined by the *xor.dat*. Open the 3D display and the *Start Learning Process* window with *Learn/Start*. Choose *file* for the validation data and select the file *xor_val.dat*. Set the error limit to 3%. Start the learning process by clicking *OK* and observe the learning process in the 3-D graphic representation window.

The network you have trained now shows the desired characteristics. Using the validation data file you forced the surface not to rise or fall steeply around the four "XOR training corners". Raising the number of neurons causes the surface to be smoother.



The design phase has now been completed. For complicated tasks in particular, it is advisable to analyze the network behavior in depth using the error diagram, the 4-D graphic representation, the vector and signal representation and the curve plotter. All testing tools are described in Part IV of this Manual in detail.



3 Runtime Modules

This part of the manual provides a systematic overview of Runtime modules of *NEUROSYSTEMS*.

3.1 SIMATIC S7-Funktion Blocks (FBs)

The neuro system is created with *NEUROSYSTEMS* and loaded into an SIMATIC S7-CPU for execution or process.

To communicate between the configuration tool and the SIMATIC S7, you require at least the *NEUROSYSTEMS* configuration tool, the SOFTNET product from SIMATIC NET, as well as an MPI card (Multi Point Interface) on the PC.

On the S7 side you require the corresponding CPU (e.g. CPU 314-1 or CPU 412-1) as well as the corresponding function and data blocks which will load the neuro system.

Attention: CP5511 will not be supported.

3.1.1 General

NEUROSYSTEMS S7 blocks can be used for process control in the same way as other SIMATIC S7 software components in the same programmable controllers – and also in conjunction with the functionality of other blocks.

The algorithms for a particular neuro application are calculated in the processor (CPU) of the SIMATIC S7 programmable controller and, more precisely, after an unconditional call by a user program or cyclically at time-controlled intervals. The results are stored in the associated instance DB and forwarded to the peripherals accordingly.

When being used by the SIMATIC S7 function blocks, each neuro application is represented by its DB (instance DB), which is described implicitly by the *NEUROSYSTEMS* configuration tool.

Knowledge of SIEMENS STEP 7 software is necessary for the interaction with the SIMATIC S7 blocks.



3.1.2 Library Contents

For the SIMATIC S7 series of controllers, the following function blocks for *NEUROSYSTEMS* exist:

FB 100:	Function block for S7-300/400	(2246 Bytes)
FB 101:	Function block for S7-400	(2210 Bytes)
DB 100:	Data block for FB 100	(4278 Bytes)
DB 101:	Data block for FB 101	(20626 Bytes)

For the SIMATIC S7-300/400 there is a 4Kbyte data block and a larger 20Kbyte data block available. For both DBs, an FB exists which is designed for the DB. It is possible to implement several neuro networks with one DB for each system and a common FB for all systems.

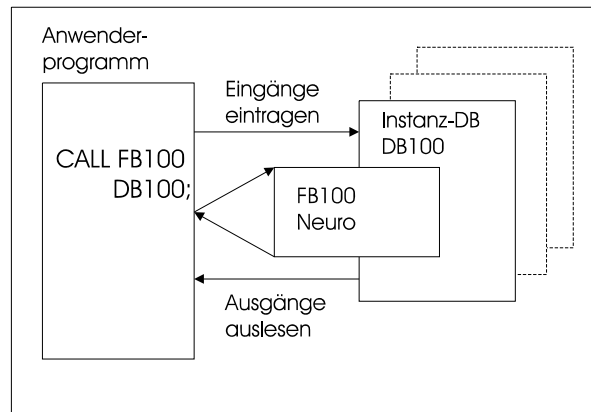
Information: FB and DB can be renamed.

3.1.3 Block Structure

In the neuro **function block** (FB), all algorithms and procedures are implemented with the all the functionality that goes with high performance neuro application:

The instance **data block** (DB) in the CPU of the programmable controller sets up the interface between the function block, the configuration tool and the user. When calling the FBs, the inputs (INPUT1,...,INPUT100) must be provided with the desired values, i.e. the addresses of where the inputs are stored should be stated. After being processed by the FB, the values written to the outputs can be read out from the instance DB.

The neuro input/output behavior is 'implicitly' entered by the configuration tool into the instance DB. You can transfer several neuro applications to- and operate them on one CPU. Each application is stored in a separate DB, which can be assigned any number. It must also be noted that in the neuro configuration tool, the DB number, in which the values are entered must match



3.1.4 Block Diagram and Parameters of the FBs

The neuro function block for S7 applications has the following block diagram:

NEURO		
INPUT1		ERROR
INPUT2		OUTPUT1
INPUT3		OUTPUT2
⋮		⋮
⋮		⋮

Input parameters

The following table shows the data types and the structure of the FB input parameters.

FB/DB 100

Byte	Parameter	Data type	Description	Default
62 (*4)	INPUT	ARRAY[1..4] REAL	4 Neuro-inputs	0.0 each

FB/DB 101

Byte	Parameter	Data type	Description	Default
62 (*4)	INPUT	ARRAY[1..100] REAL	100 Neuro-inputs	0.0 each

Output parameters



The following table shows the data types and the structure of the FB output parameters.

FB/DB 100

Byte	Parameter	Data type	Description	Default
0	ERROR	BYTE	Processing-Information	B#16#0
78 (*4)	OUTPUT	ARRAY[1..4] REAL	4 Neuro-Outputs	0.0 each

FB/DB 101

Byte	Parameter	Data type	Description	Default
0	ERROR	BYTE	Processing-Information	B#16#0
462 (*4)	OUTPUT	ARRAY[1..10] REAL	10 Neuro-Outputs	0.0 each

Additional parameters

In addition to the input and output parameters, the FB also has two further parameters:

FB/DB 100

Byte	Parameter	Data type	Description	Default
2	START_STOP	INT	>< 0: Execute neuro-application == 0: Do not execute neuro-application	0
94...	DATEN	BYTE	Internal field for the FB	B#16#0

FB/DB 101

Byte	Parameter	Data type	Description	Default
2	START_STOP	INT	>< 0: Execute neuro-application == 0: Do not execute neuro-application	0
502..	DATEN	BYTE	Internal field for the FB	B#16#0



3.1.5 Calling the Function Blocks

1. The neuro function block (FB) must be called by the user. The call can be made on a cyclically (in OB1) and/or on a time-controlled program execution level (e.g. every 100ms in OB35) and it the same applies for both FBs. The FB **must** be called unconditionally. It is controlled via the START_STOP variable in the instance DB.
2. During a call to the function block, you must specify the desired instance data block (neuro DB) that contains the neuro application, which was loaded and created using the configuration tool.
3. Unused inputs or outputs must not be connected.

4. Example:

A minimum call with the ERROR parameter would have the following listing:

STL
CALL FB100, DB100
(ERROR := MB100);

Note: In a new neuro application, the FB must be entered into the empty data block first. Only when it is entered, the data block is recognized as a neuro data block by the configuration tool and *NEUROSYSTEMS* can build-up a connection to the data block. Afterwards the data block can be written and read.

Note: You can freely change the numbers of both neuro function blocks - within the limits of the CPU. The presets are FB100 (for the “small” DB) and FB101 (for the “large” DB). They can be renamed using the STEP 7 software.



3.1.6 External Setting of the I/Os

External access to the neuro application in the program is possible if the DB is created as an instance and the name is defined in the symbols.

Example:

```
..
..
T  "Pendel".INPUT1
..
L  "PENDEL".OUTPUT2
..
USW
```

Symbolik des DBs

Variable im DB

3.1.7 Execution Control

Execution of the neuro application **must** be controlled via the START_STOP variable in the data block. This variable can be controlled and read by the operator. The configuration tool also changes the START_STOP variable during the data transfer.

You can directly influence execution of the neuro application via the START_STOP variable:

START_STOP	Meaning
=W#16#0000	The neuro application is not being executed
≠W#16#0000	The neuro application is being executed

Note: If you do not want a neuro application to be executed cyclically, you can control execution with the value of the START_STOP variable: e.g. by calling the function block in OB1 and by forming a time slice in the time-controlled OB.

3.1.8 Influence using the Configuration Tool

The execution of the neuro application is also controlled by the configuration tool. Before transmission of neuro data to the DB from the PG/PC is carried out, execution is stopped by setting the START_STOP variable to W#16#0000. After transmission, the tool enters the value not equal to W#16#0000 in the START_STOP variable, which allows execution to resume.

**Example:**

STL	Explanation
L 0	
T "Pendel".START_STOP	Do not execute neuro-system
l 123	
T "Pendel".START_STOP	Execute neuro-system
l "Pendel".START_STOP	
l W#16#FFFF	
==I	
= M 10.0	Change of the neuro application by the configuration tool

Note: The configuration tool always allows execution after a transmission even if the START_STOP variable was set to W#16#0000 by the user program before transmission.

3.1.9 Evaluating the Parameter "ERROR"

The block provides information about the state of the neuro application via the ERROR parameter. This information is subdivided into three categories: No error, Warning and Error.

- **No error**

If execution of the neuro application was completed without error, the variable ERROR = "B#16#00".

- **Warning**

If the START_STOP variable = W#16#0000 (neuro application not being executed), the ERROR parameter = B#16#01.

This shows you that the outputs have not been updated, but are actually the old values.

- **Error**



If an instance DB of the correct length is present in the CPU but no neuro application has been loaded from the configuration tool, the ERROR parameter = B#16#11.

If the length of the instance DB specified is inadequate, the ERROR parameter = B#16#21.

Content (B#16#...)	Interpretation
00	No error has occurred during execution
01	Execution of the neuro system is blocked by the user or the configuration tool
11	The instance DB contains no valid neuro system
21	Length of the data block is inadequate (no neuro-DB)

Note: If an error is found or a warning occurs, the outputs are not deleted. After evaluating the ERROR variable, you must decide whether the old output values are to be processed or whether a defined value should be outputted.

Caution: If the FB finds a warning or error, execution of the function block is terminated immediately.



3.1.10 Typical Execution Times

Execution of neuro functions are computationally intensive operations. The execution speed of specific neuro applications depends on the performance of the CPU used. The more often the CPU must calculate the output variables per unit time, the lower the number of neuro systems that can be installed. Depending on the number of inputs and outputs, the number of neurons, the weights, the network type and the automation device, there are differing processing periods. For your applications the following processing period measurements can be helpful (S7-300 with CPU 314 and S7-400 with CPU 413-1):

Inputs	2	4	4	4	4
Neurons	1	2*15	2*40	49	50
Outputs	1	2	2	2	2
Weights	5	347	1922	345	352
Network Type	MLP	MLP	MLP	MLP	MLP
S7-300	ca. 6,5 ms	ca. 215 ms	impossible	ca. 265 ms	impossible
S7-400	ca. 3,3 ms	ca. 86 ms	ca. 260 ms	ca. 137 ms	ca. 141 ms

Inputs	4	4	4	100	100
Neurons	25	10	4	16	2*14
Outputs	2	2	2	10	10
Weights	177	72	30	1786	1774
Network Type	MLP	MLP	MLP	MLP	MLP
S7-300	ca. 140 ms	ca. 58 ms	ca. 25 ms	impossible	impossible
S7-400	ca. 70 ms	ca. 27 ms	ca. 11 ms	ca. 86 ms	ca. 118 ms



3.2 SIMATIC WinCC

3.2.1 Installing the WinCC-OLL

For the various versions of SIMATIC WinCC there are available various OLLs, which you can free download.

Additional you can use ActiveXControl for WinCC (see next chapter).

After download the OLL, please copy the following files to the "bin" directory of WinCC (normally c:/Siemens/WinCC/bin):

- neuro.oll Object DLL that implements the neuro module in WinCC 3.1 or 4.0.
- NeuroDLL.dll Provides functions that neuro.oll requires.
- neuroio.dll Provides functions that neuro.oll requires.
- neuroenu.lng Is required for the English WinCC user interface.
- neuroFRA.lng Is required for the French WinCC user interface.

3.2.2 WinCC Applications with the WinCC-OLL

3.2.2.1 *Editing Actions for a Neuro Object*

The following actions must be performed.

- Put the Neuro.oll into the WinCC picture
- Assign the snl file with the file-path
- Create internal WinCC variables
- Link the variables with the I/O field

Next, you have to edit a C-action, which

- Gets the internal variables
- Sets the neuro inputs
- Queries the neuro outputs
- Sets the internal variables as the output value

It's also acceptable to put the C-action after input1. This C-action is then called cyclically.

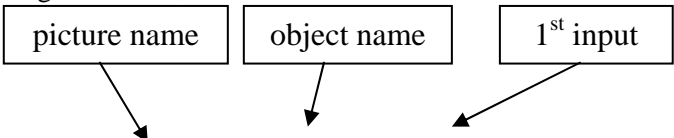



3.2.2.2 Example C-Action for Neuro Object "Neuro1"

```
#include "apdefap.h"

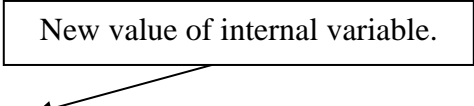
double _main(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
    double dTarget;           // local variable
    double dActual;           // local variable
    double dDiff;             // local variable
    double dNeuroOutput;      // local variable

    dtarget = GetTagDouble("target value"); // get value of the internal variable target value
    dactual = GetTagDouble("actual value"); // get value of the internal variable actual value
    dDiff=dTarget-dActual;           //difference creation

    
    SetPropDouble("neurodemo","Neuro1","NeuroIn1",dDiff);           //set Neuro-input

    
    dNeuroOutput = GetPropDouble("neurodemo","Neuro1","NeuroOut1"); //query Neuro-output
    dActual = GetTagDouble("actual value");           //query actual value

    dActual = dActual - dNeuroOutput;           //calculate new actual value

    
    SetTagDouble("actual value",dActual);           //set actual value

    return dDiff;
}
```



3.2.2.3 Simple Neuro Example

```
#include "apdefap.h"

double _main(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName)
{
    double Input-value1;
    double Input-value2;
    double Output-value1;

    Input-value1 = GetTagDouble("e1");           // get internal variable i1
    Input-value2 = GetTagDouble("e2");           // get internal variable i2

    SetPropDouble("NewPdl0","Neuro1","NeuroIn1",Input-value1);           //set Neuro-input 1
    SetPropDouble("NewPdl0","Neuro1","NeuroIn2",Input-value2);           //set Neuro-input 2

    Output-value1 = GetPropDouble("NewPdl0","Neuro1","NeuroOut1"); //query Neuro-output1
    SetTagDouble("o1",Output-value1);           //set internal variable o1
    return Output-value1;
}
```

3.2.2.4 Limitations

Please be aware, that the WinCC-OLL neural network is only active when the picture is active.

If you would like the neuro network to be always active, you must additionally put the WinCC-OLL into the overview picture.

Online-monitoring such as that with the SIMATIC S7 components is **not** possible with the WinCC-OLL.

Note: From the version WinCC V6 SP1 on there are no OLL-objects any more.



3.3 ActiveXControl

3.3.1 Installing the ActiveX Control

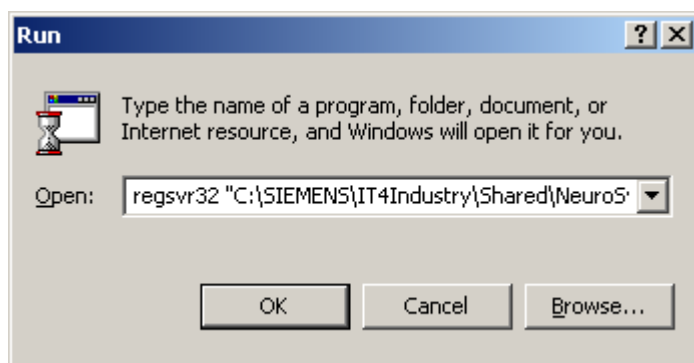
The component will be installed at the same time as *NEUROSYSTEMS* and using the same setup program.

The setup program will install the control's file (*NeuroSystems.ocx*) into the relevant folder.

Setup will register the component on your system so that it will be available to programs/containers that capable of inserting ActiveX controls.

If, however, the component needs to be registered manually follow these steps:

1. Press the 'Start' button
2. Select 'Run'
3. Enter "regsvr32 "<path to NeuroSystems.ocx>" in the text field
e.g. "regsvr32 "C:\...\ NeuroSystems V5.0\ActiveXControl\NeuroSystems.ocx""
4. Press the 'OK' button





3.3.2 The Properties

The control has 113 properties that can be divided in four groups:

- File path (1)
- Input values (100)
- Output values (10)
- Trigger (2)

The properties are listed in the list below:

Properties:	Description:
NeuroFilePath	The path to a valid <i>NEUROSYSTEMS</i> file (.snl)
NeuroIn1	First input (of type: float)
...	...
NeuroIn100	100th input (of type: float)
NeuroOut1	First output (of type: float)
...	...
NeuroOut10	10th output (of type: float)
Trigger	change from 0 to 1, control calculates new output (of type: boolean)
TriggerSetup	<i>=0, the control waits for Trigger property</i> <i>=1 the control calculates the output values every time an input value changes (of type: short).</i>

There are **100 input properties**, each with the name NeuroIn and the number of the input (1 to 100 inclusive) appended to the end of the name e.g. NeuroIn1, NeuroIn2, NeuroIn3 etc.

The property is of type 'float'.

There are **10 output properties**, each with the name NeuroOut and the number of the output (1 to 10 inclusive) appended to the end of the name e.g. NeuroOut1, NeuroOut2, NeuroOut3 etc. The property is of type 'float'.



The **NeuroFilePath** is normally an input – it specifies the file path to a valid *NEUROSYSTEMS* file (.snl).

As soon as the path to a valid *NEUROSYSTEMS* file (.snl), then the values of the inputs will be used to calculate the outputs and the outputs will be overwritten.

The **NeuroFilePath** property only allowed to be the path to a *NEUROSYSTEMS* file (.snl). An invalid file path will be replaced with an error message. The property will also change to an error message if no *NEUROSYSTEMS* ++ license is found or the authorization checking process fails.

The error message can be removed when the problem is solved, i.e. the license is put on the system, AuthorsW is installed etc. The error text will actually change when a new file path is loaded.

The value of the **TriggerSetup** property can be set to 0 or 1. If it is set to 0, the control waits for the **Trigger** property to change from 0 to 1 to calculate the new output. However, if it is set to 1 the control calculates the output values every time an input value changes. **TriggerSetup** property is an Integer, the **Trigger** property is a Boolean.

Each of these properties can be set manually.



3.3.3 The Events

The properties are listed in the table below:

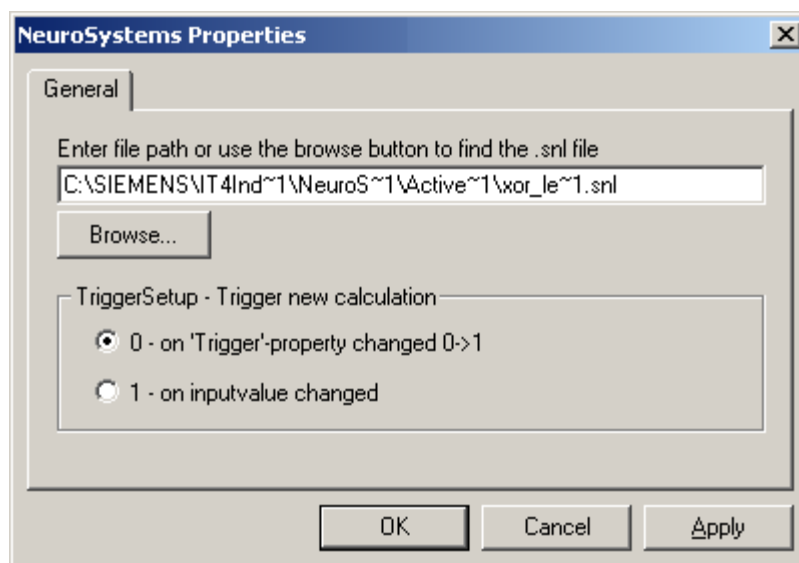
Event:	Description:
NeuroOut1Changed	The value of the first output has changed
NeuroOut2Changed	The value of the second output has changed
NeuroOut3Changed	The value of the third output has changed
NeuroOut4Changed	The value of the fourth output has changed
NeuroOut5Changed	The value of the fifth output has changed
NeuroOut6Changed	The value of the sixth output has changed
NeuroOut7Changed	The value of the seventh output has changed
NeuroOut8Changed	The value of the eighth output has changed
NeuroOut9Changed	The value of the ninth output has changed
NeuroOut10Changed	The value of the tenth output has changed

These events are fired (called) when the values of the outputs change. This happens when a valid *NEUROSYSTEMS* file (.snl), is loaded and when the inputs change while a valid *NEUROSYSTEMS* file (.snl), is loaded.



3.3.4 The Property Page

The property page allows you to enter the path to the *NEUROSYSTEMS* file (*.snl) and you can also browse your computer and local network for the file. The path will be entered into the text box when the 'Open' window is closed. You can also specify the control's behaviour as to when calculating the output values is desired via the "Trigger setup".



Note: The file path is not applied to the control yet – you must press the 'OK' or 'Apply' button.

If there is no license or a problem with the license, then there will be an error message in the 'Properties' window **the next time the window is opened**, i.e. when you use the apply button and the authorization process fails, the path will remain in the text field until the property page is closed. It will be replaced the next time the property page is opened.

The same applies when a license error message appears and a license is copied onto the system however the text will not change until a new file path is loaded.



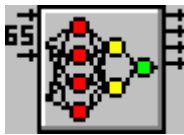
3.3.5 Graphical Interface

The control is drawn as a fixed-size control, i.e. it cannot be made larger or smaller. In some programs it may be possible to zoom in an out but the control will remain a fixed size.

It consists of 3 main parts:

1. The main icon to show the general status of the control
2. The input and output arrows to show how many inputs and outputs are being processed – defined by the *NEUROSYSTEMS* file (.snl) that is currently loaded
3. The background

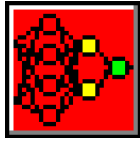
Normal Mode



This is when the license is found, a valid *NEUROSYSTEMS* file (.snl) is loaded and all the DLLs are available and functioning correctly. You can see the normal icon in the center and small icons indicating the number of inputs and outputs defined by the *NEUROSYSTEMS* file (.snl) currently loaded. The inputs are shown as arrows on the left of the main icon when they are less than or equal four. Are there more than four inputs, a number between two arrows shows the amount of the inputs. The same counts for the outputs on the right.



Error Mode



When this picture is drawn then no *NEUROSYSTEMS* file (.snl) is loaded, or no license was found or the license could not be authenticated because a DLL is missing or corrupted.

When this icon is shown,

1. Check the 'NeuroFilePath' property to ensure that a file path has been entered,
2. Then check that the file exists,
3. That the file is valid and not corrupted (open it in *NEUROSYSTEMS*),
4. That the *NEUROSYSTEMS* Base License is on the computer
5. And finally that AuthorsW is installed and AddOnAut.dll is located on the system in a folder listed in the 'Path' environment variable of your computer.

No NeuroDLL Mode



When this picture is drawn then the *NEUROSYSTEMS* DLL (NeuroDLL.dll) was not found or could not be loaded. Check that the DLL is on the computer



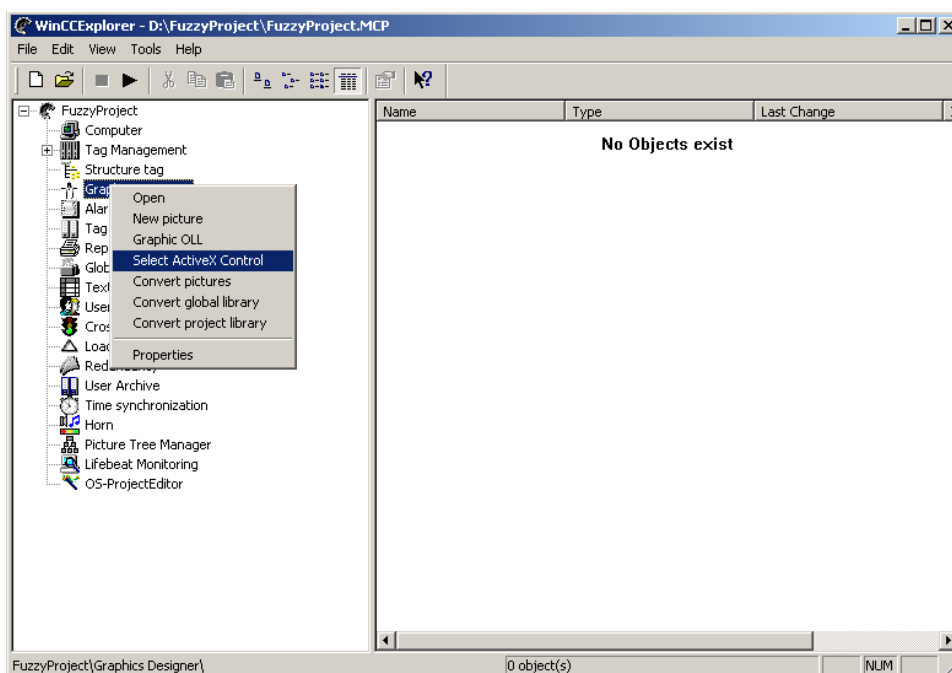
3.3.6 Using the Control in SIMATIC WinCC

Note: Please be aware, that the *NEUROSYSTEMS* ActiveX control is only active when the picture is active.

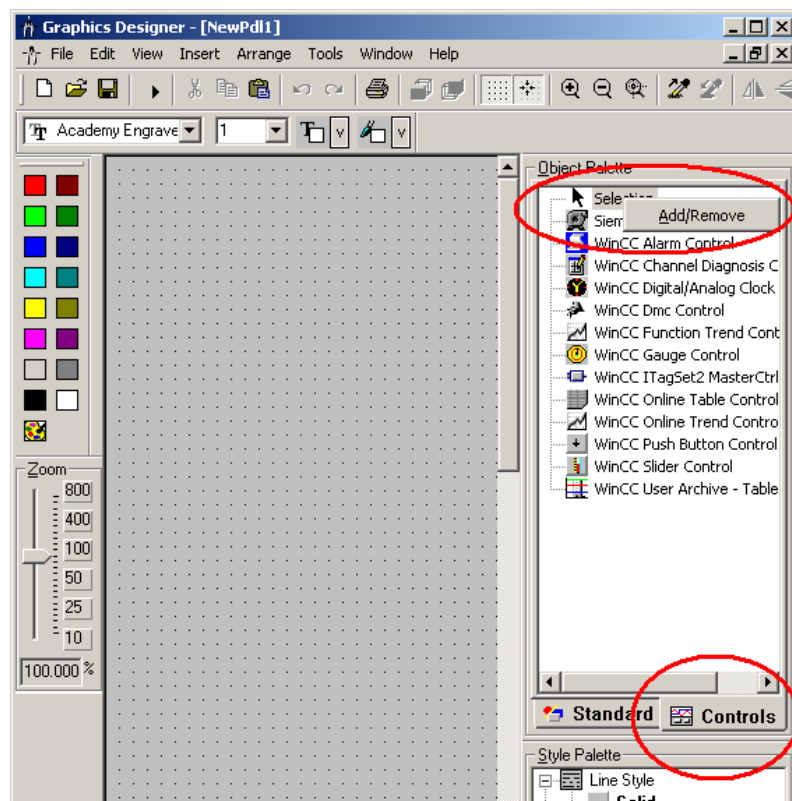
3.3.6.1 Registering the ActiveX Control with SIMATIC WinCC

SIMATIC WinCC can also register controls using the following steps

- Open the 'Select ActiveX Controls' dialog box by
 - Opening WinCC Explorer, right-clicking 'Graphics Designer' and selecting 'Select ActiveX Control' from the menu.

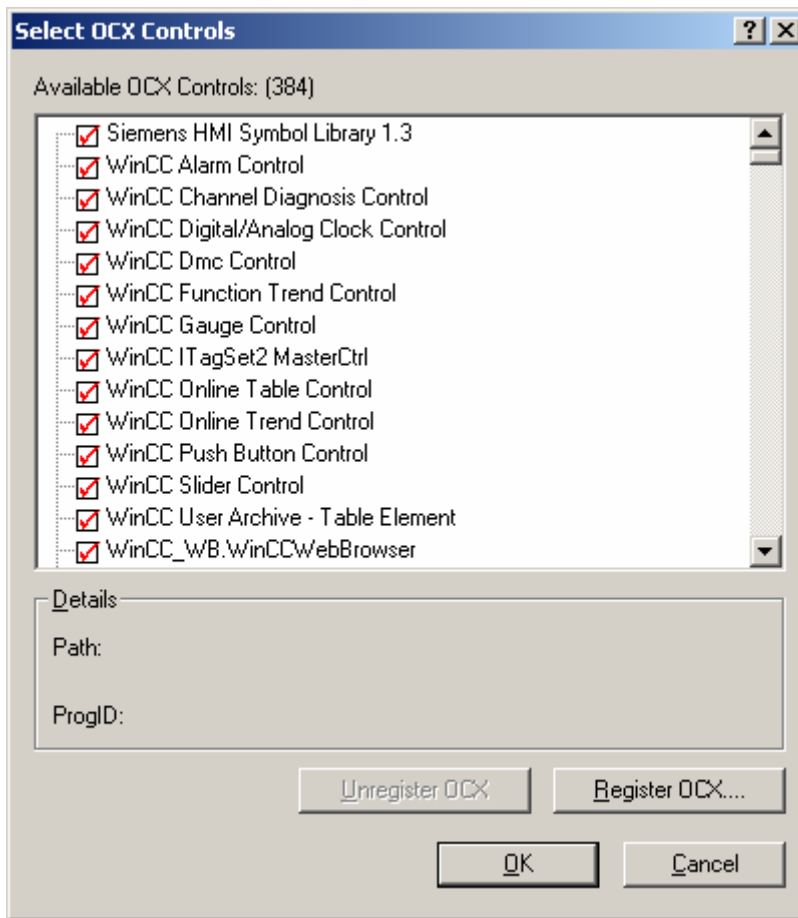


- Or by opening 'Graphics Designer', clicking the 'Control' tab in the 'Object Palette', right-clicking on any of the controls and choosing 'Add/Remove' from the menu that pops up.

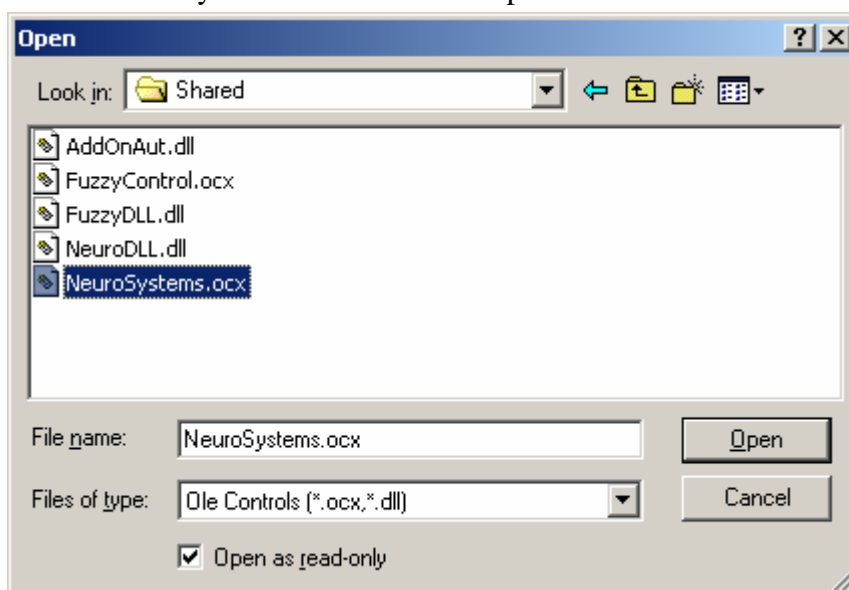




The following Dialog appears:



- Click 'Register OCX....'
- Locate NeuroSystems.ocx and click 'Open'

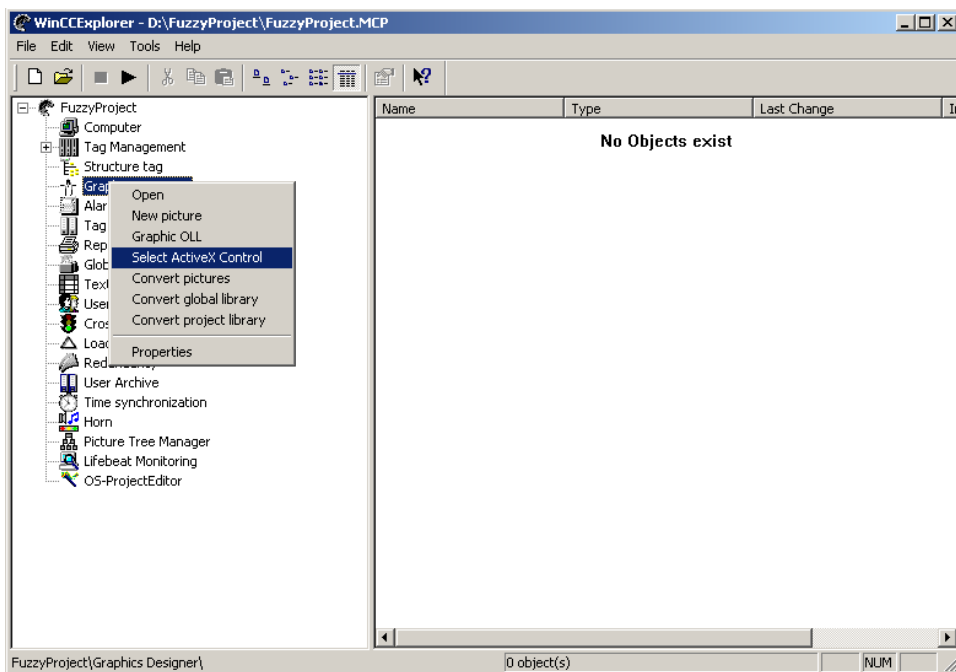




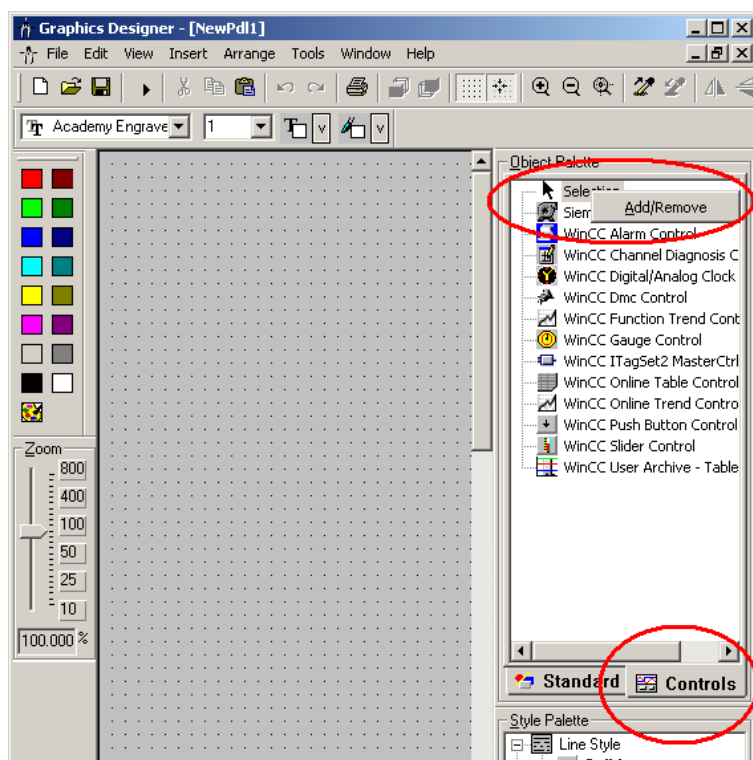
The ActiveX control is now registered on the system.

3.3.6.2 Inserting the ActiveX Control in SIMATIC WinCC

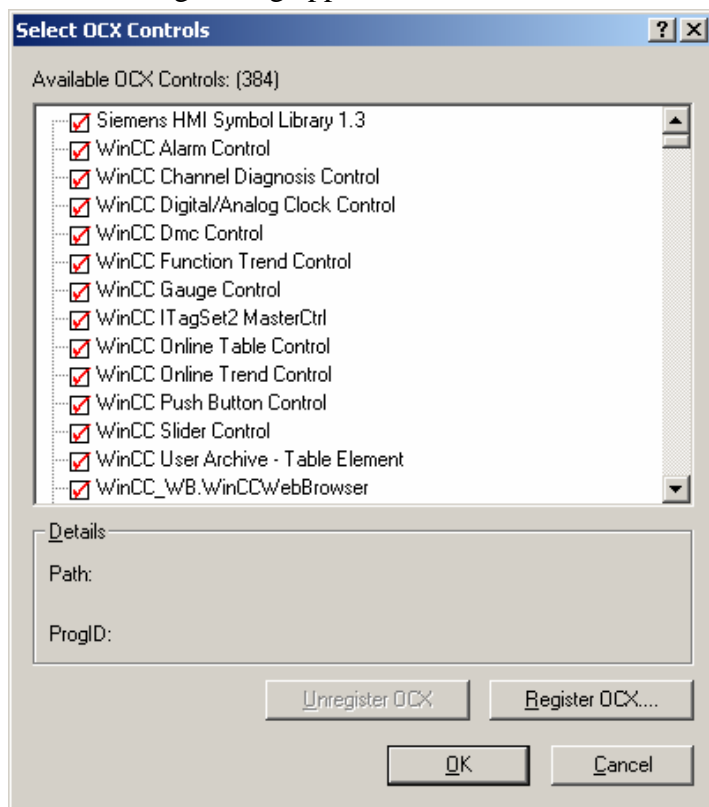
- Open the 'Select ActiveX Controls' dialog box by nicht wie im Deutschen
 - Opening WinCC Explorer, right-clicking 'Graphics Designer' and selecting 'Select ActiveX Control' from the menu.



- Or by opening 'Graphics Designer', clicking the 'Control' tab in the 'Object Palette', right-clicking on any of the controls and choosing 'Add/Remove' from the menu that pops up.

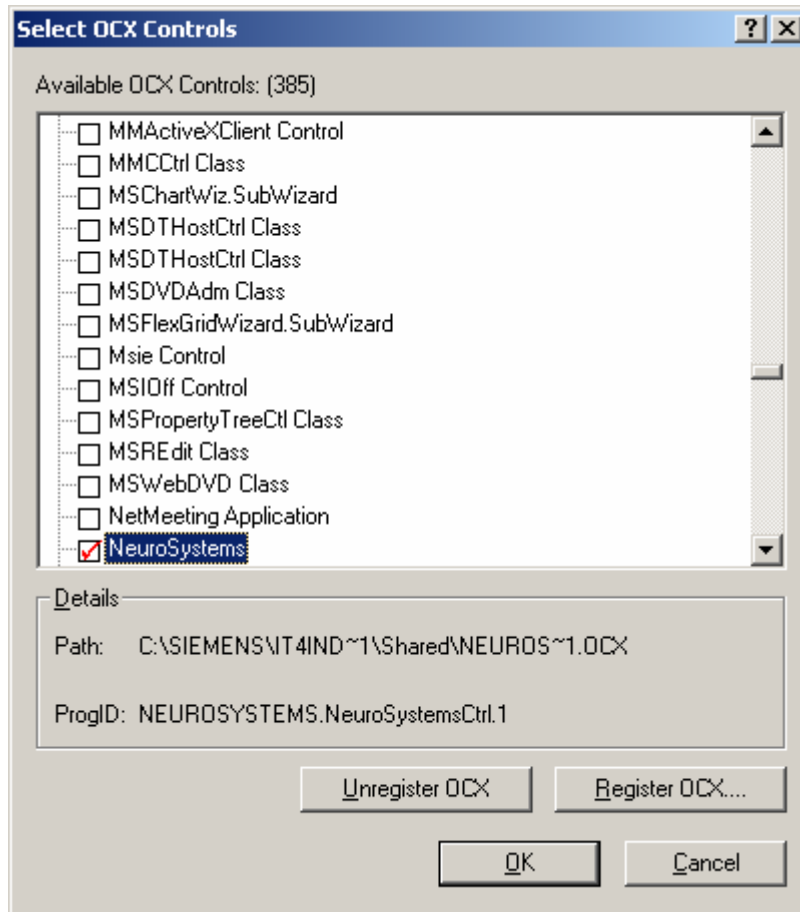


- The following Dialog appears:

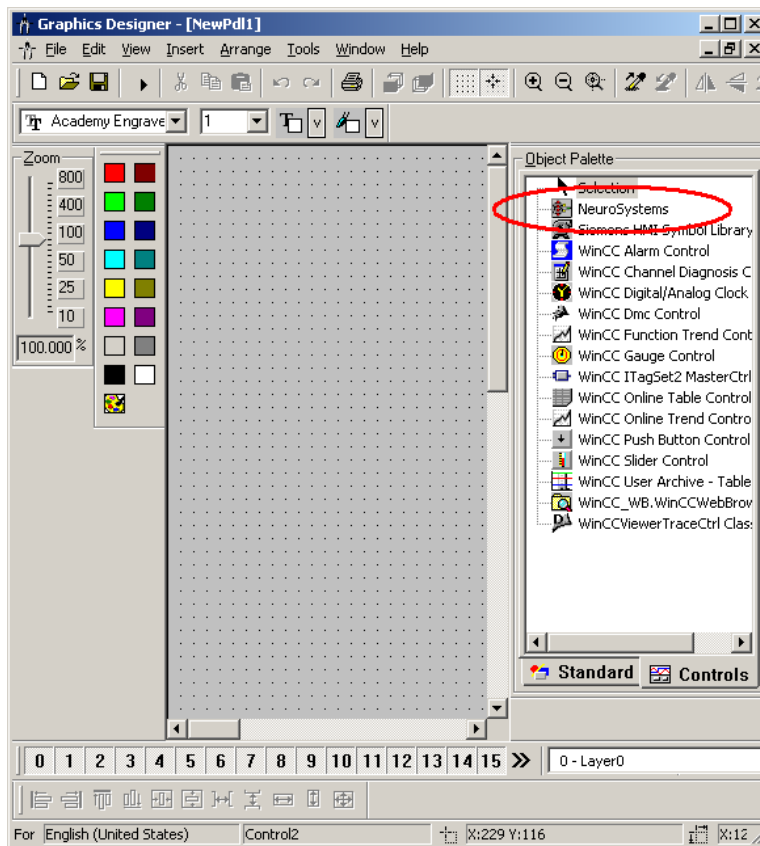




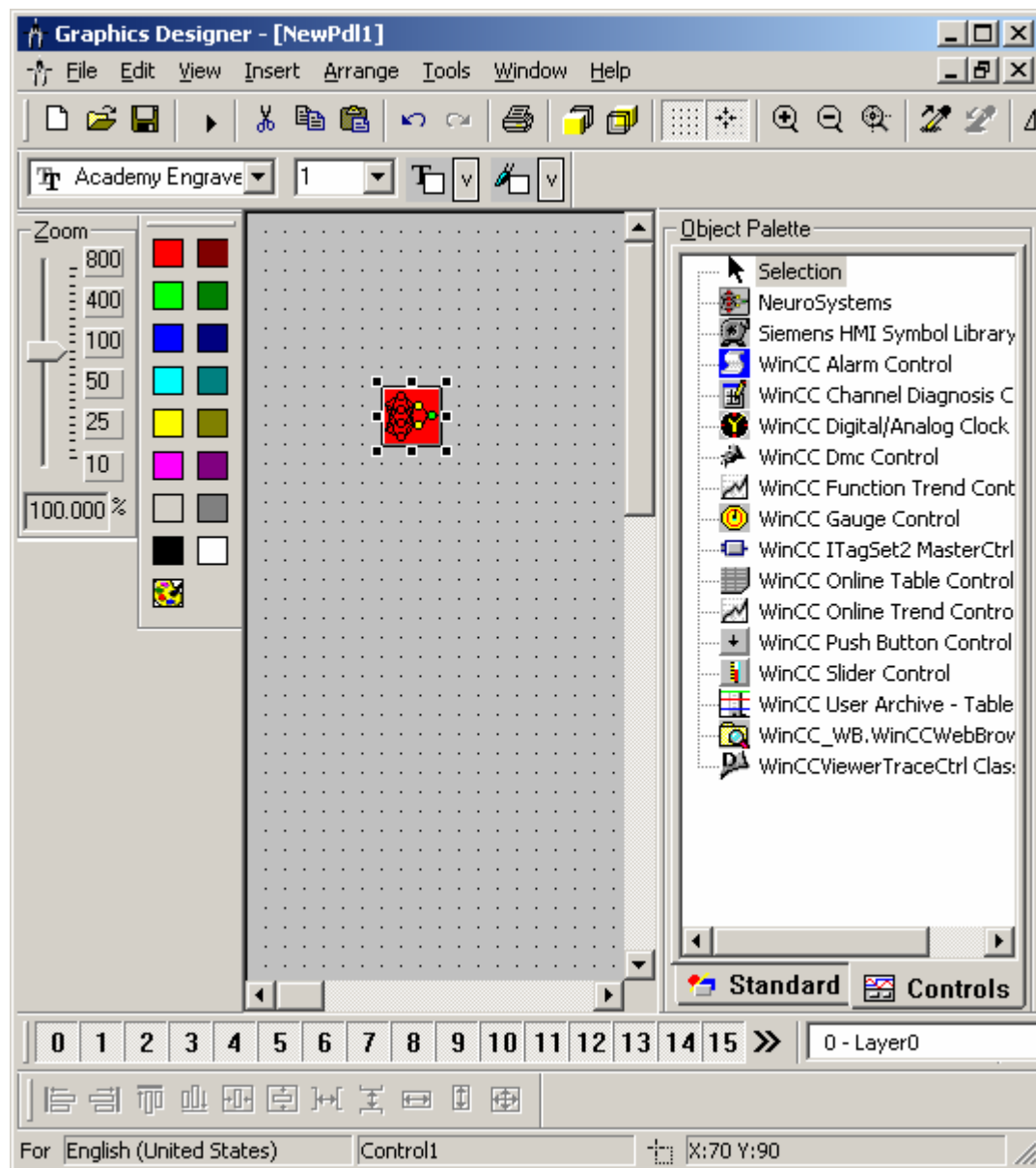
- Controls with a red tick in the checkbox are available in the Object Palette of 'Graphics Designer' and are listed before all other controls in this dialog box. Scroll down the list until you find *NEUROSYSTEMS*, place a tick in the checkbox and press 'OK'



- The window will close and *NEUROSYSTEMS* will be available in the object Palette. To remove it, reopen the 'Select OCX Controls' dialog and uncheck the checkbox.



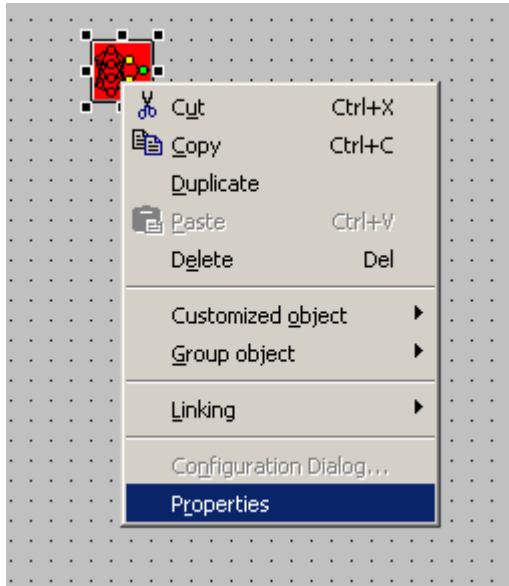
- To select the control, click on it once with the left mouse button
- To insert it into a picture (while it is selected), simply click on the picture once with the left mouse button.



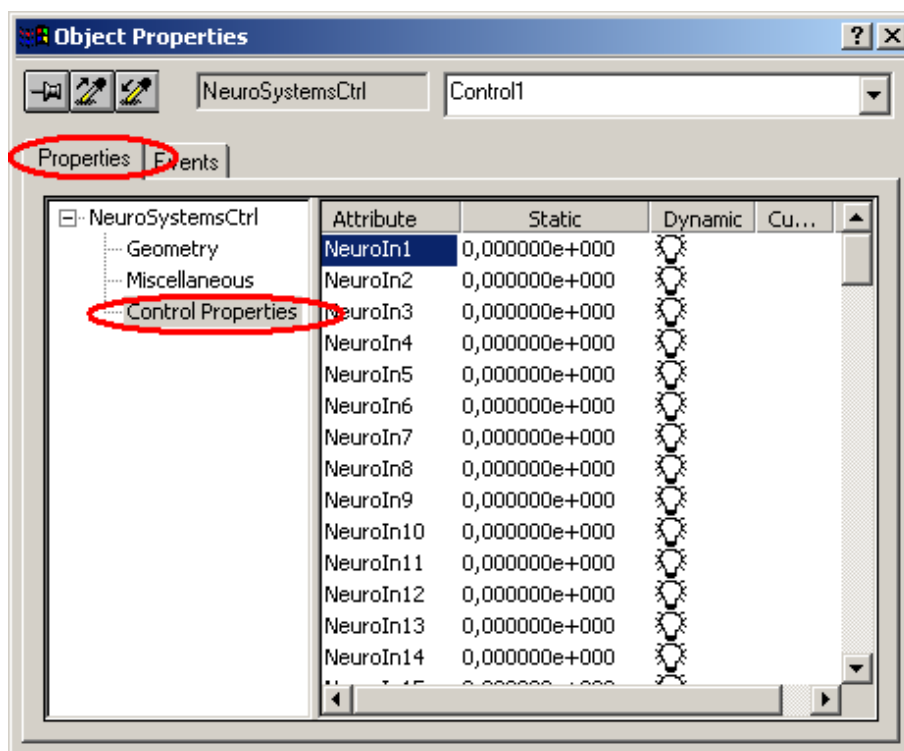


3.3.6.3 Setting the Properties with SIMATIC WinCC

To set the properties, right click the control and select 'Properties' from the menu'



Then select 'Control Properties' from the 'Properties' tab.



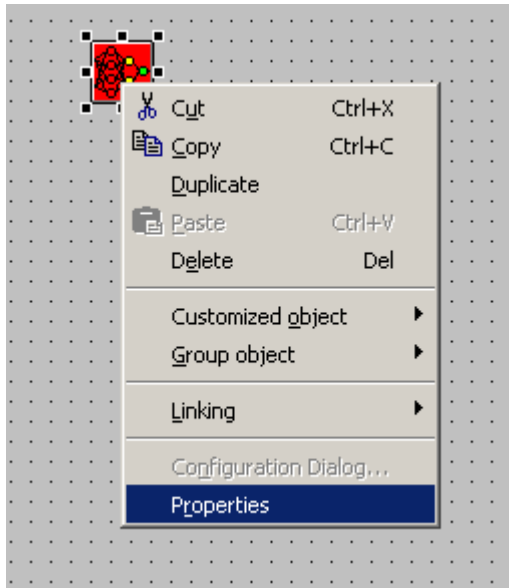


Now, each of the properties listed can be set manually and other WinCC functions can be defined to interact with these properties. The changes are executed immediately so the values of the outputs will be updated if a *NEUROSYSTEMS* file (.snl) is loaded and they will also be updated after each change to the inputs.

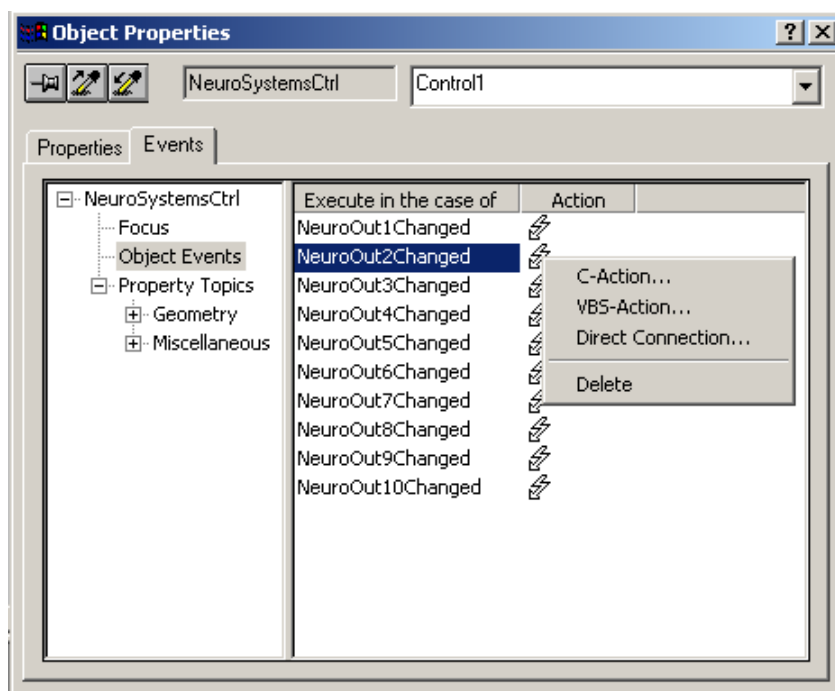


3.3.6.4 Reacting to the Control's Events in SIMATIC WinCC

To react to an event fired (caused) by the control, right click the control and select 'Properties' from the menu'



Then select 'Object Events' from the 'Events' tab.



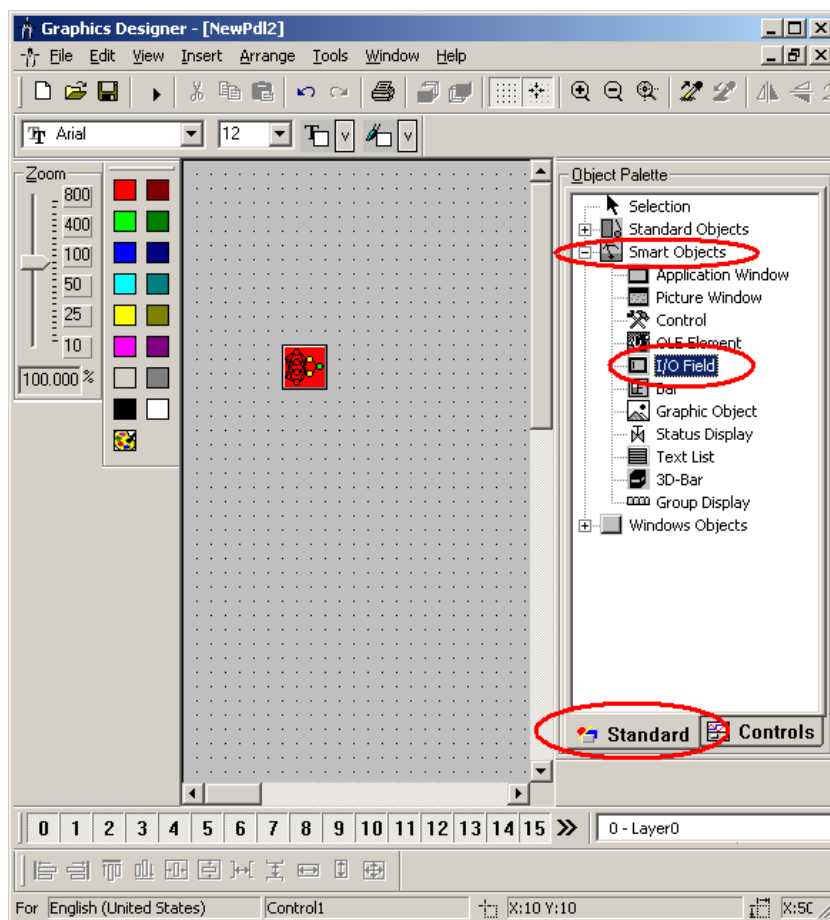


An 'Action' can be configured for each time an event is fired – in this case a 'C-Action', 'VBS-Action' or 'Direct Connection'. Double-clicking on the 'Action' symbol opens the 'Direct Connection' dialog and a right mouse click opens the context menu illustrated here.

3.3.6.5 Example:

We must first insert the objects that we will use in this example

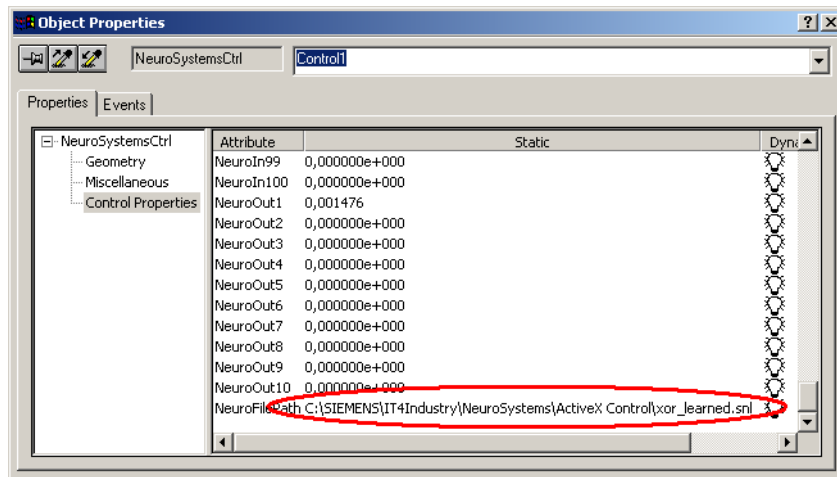
- Insert the *NEUROSYSTEMS* ActiveX control into the SIMATIC WinCC 'Object Palette', as described above.
- Select 'I/O Field' from the 'Smart Objects' section in the 'Standard' tab of the 'Object Palette'.



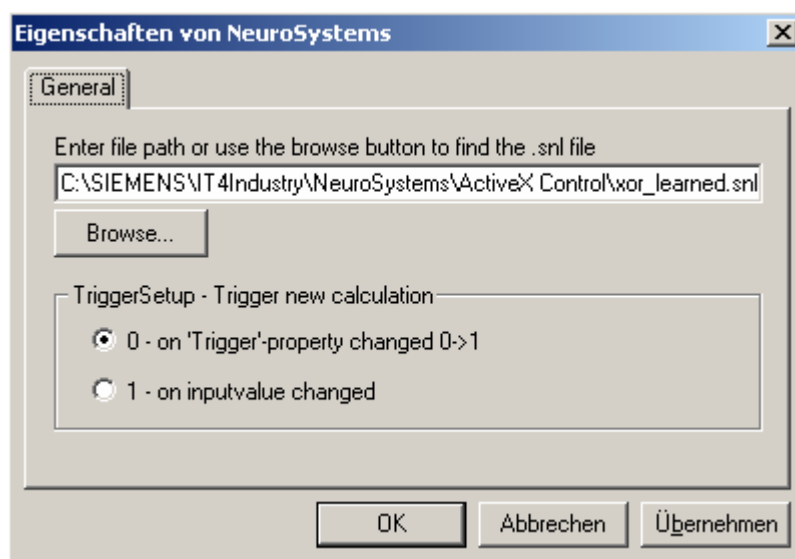
- Insert the I/O field into the picture by clicking on the area of the picture where you want to place the field and accept the default settings by pressing 'OK' when the 'I/O Field Configuration' dialog appears.
- Repeat this procedure depending on your needs – 2 times in this example for 2 inputs and 1 output.



Enter the path to the xor_learned.sn1 file in the properties dialog box for the control.



Or use the property page by double clicking on the object itself.

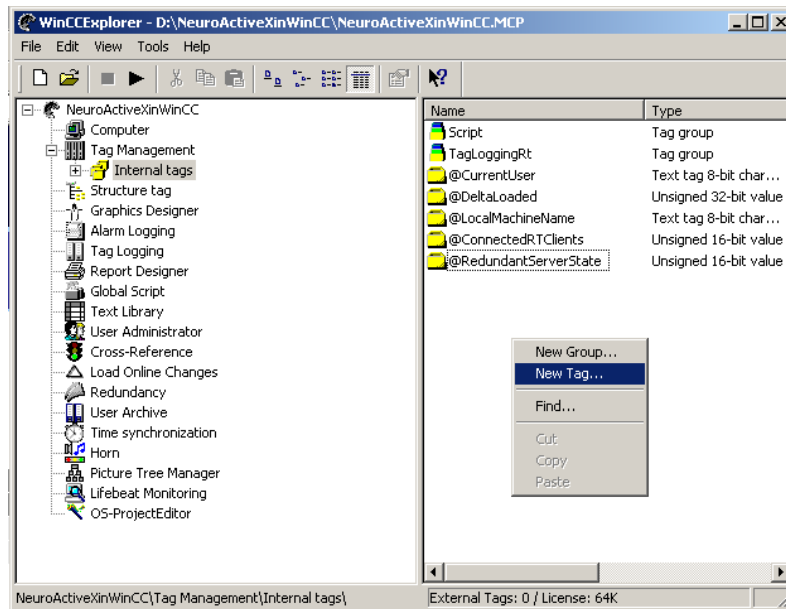


3.3.6.5.1 Using the Control with Tags and Direct Connections

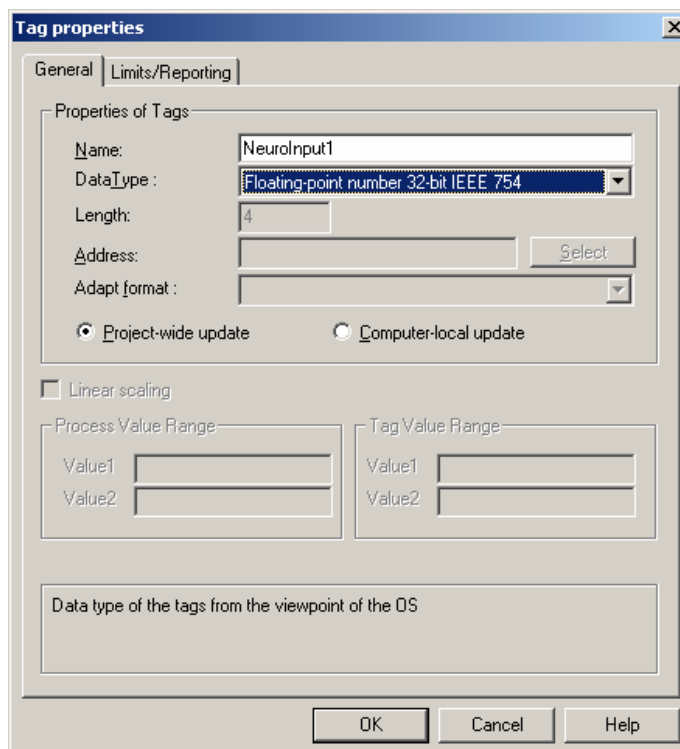
Tags are used widely within WinCC so our first example connects two input fields to the input properties of the ActiveX control via tags.

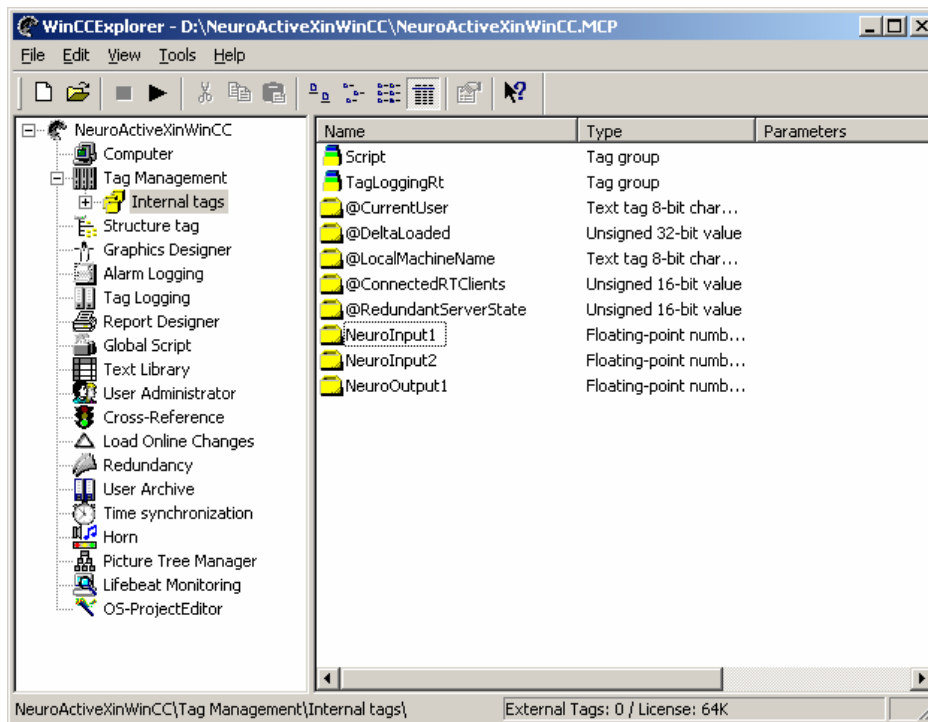


- We can do this in WinCC Explorer by opening 'Tag Management', followed by 'Internal Tags' and then right-clicking the main window and selecting 'New Tag...' from the menu

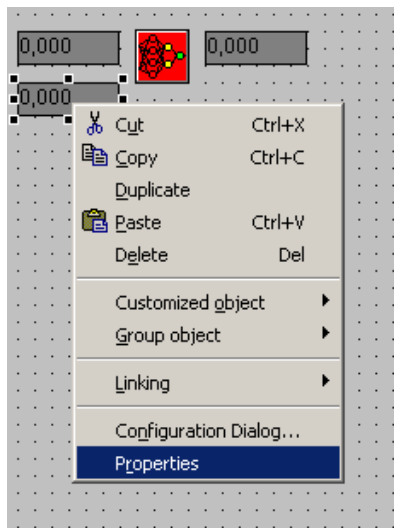


- Create 4 tags for the 4 I/O fields. Our variables are floating point values so that is the type we choose and we assign suitable names such as 'NeuroInput1', 'NeuroOutput1' etc.

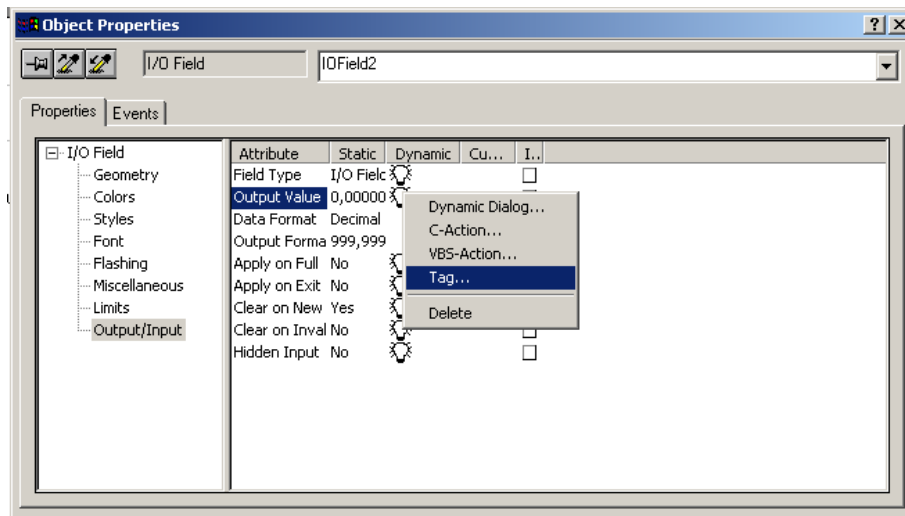




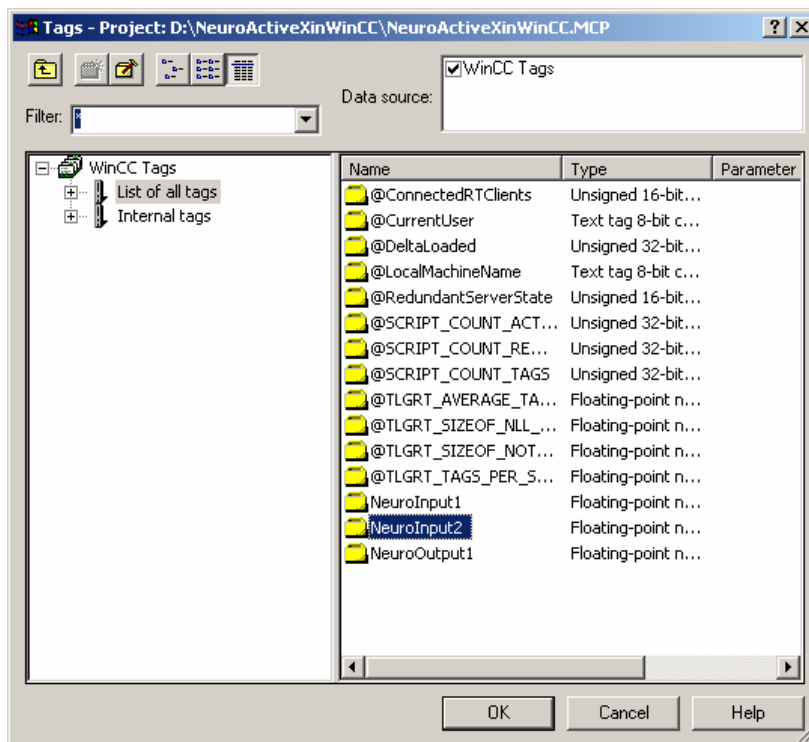
- Right-click the I/O fields that you wish to be the inputs and select the 'Properties' option from the menu.



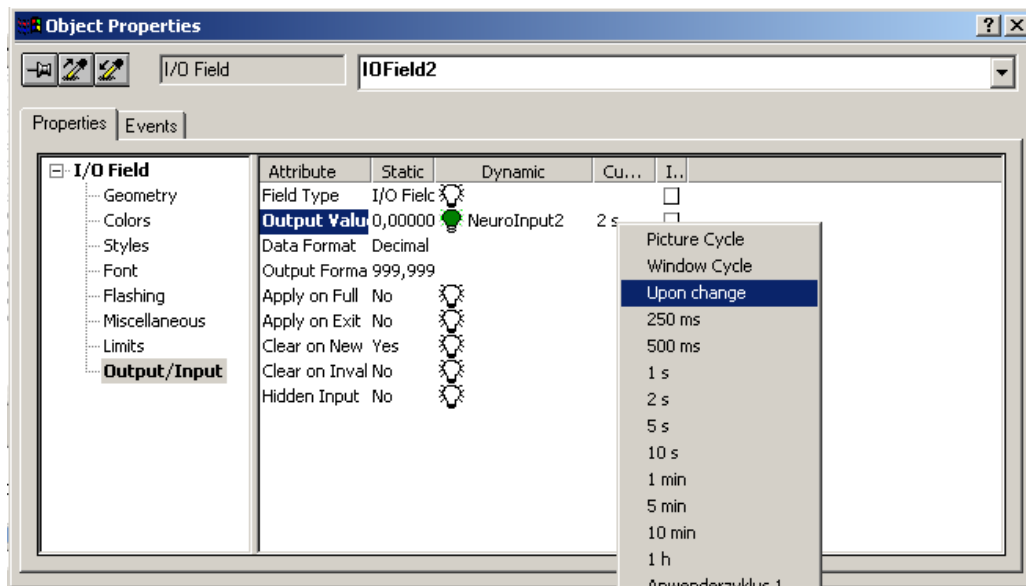
- We should connect the I/O field to the appropriate 'Tags'; otherwise the value will be the default. In the 'Properties' tab, select 'Input/Output' and right-click the 'Output Value' light bulb to open the menu required.



- Select the Tag option and select the appropriate tag, i.e. 'NeuroInput1' & 'NeuroInput1' for the two input fields and 'NeuroOutput1' for the output field.

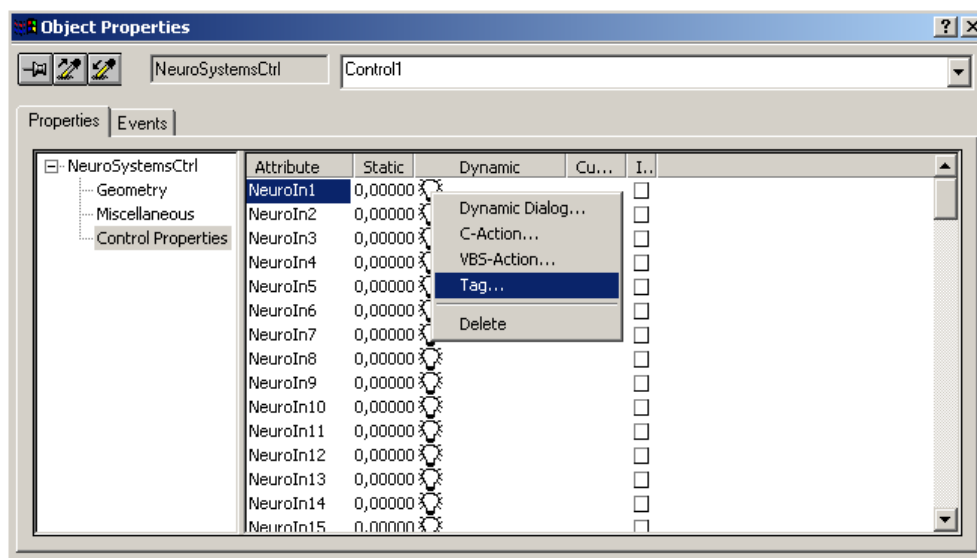


- Change the 'Update Cycle' to 'Upon Change' for each I/O field by right-clicking the 'Current' column.

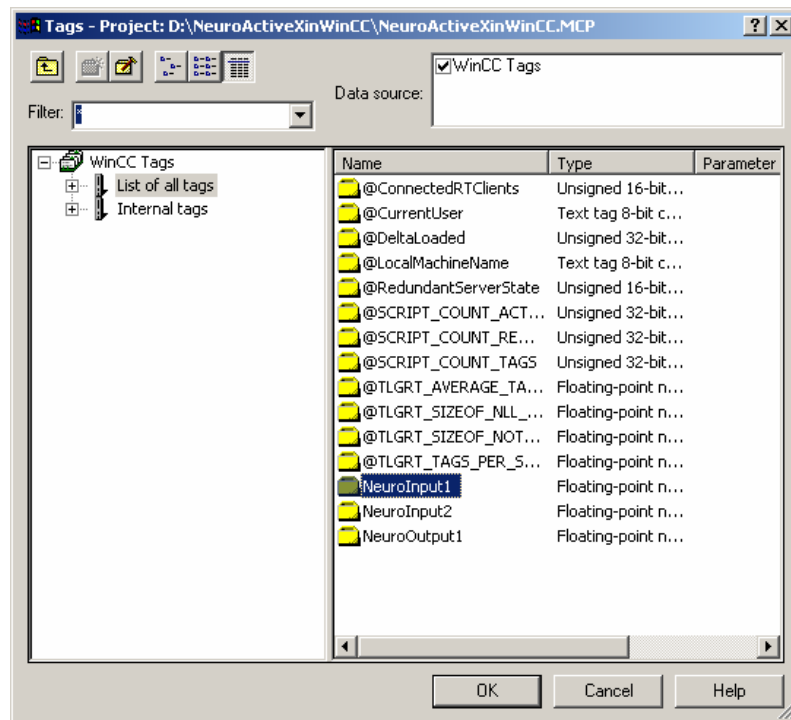


- Next we must connect the two input tags to the input properties. In the properties dialog box for the ActiveX control, right-click the light bulb for 'NeuroIn1' and select the Tag option.

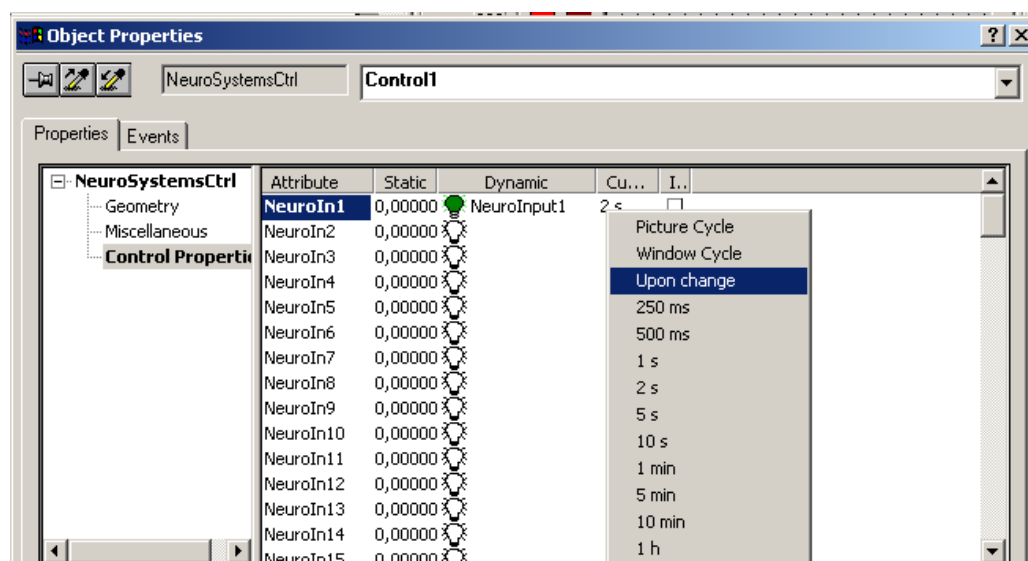
This option is found in the 'Properties' tab and the 'Control Properties' section.



- Now, select the 'NeuroInput1' tag and press 'OK'.

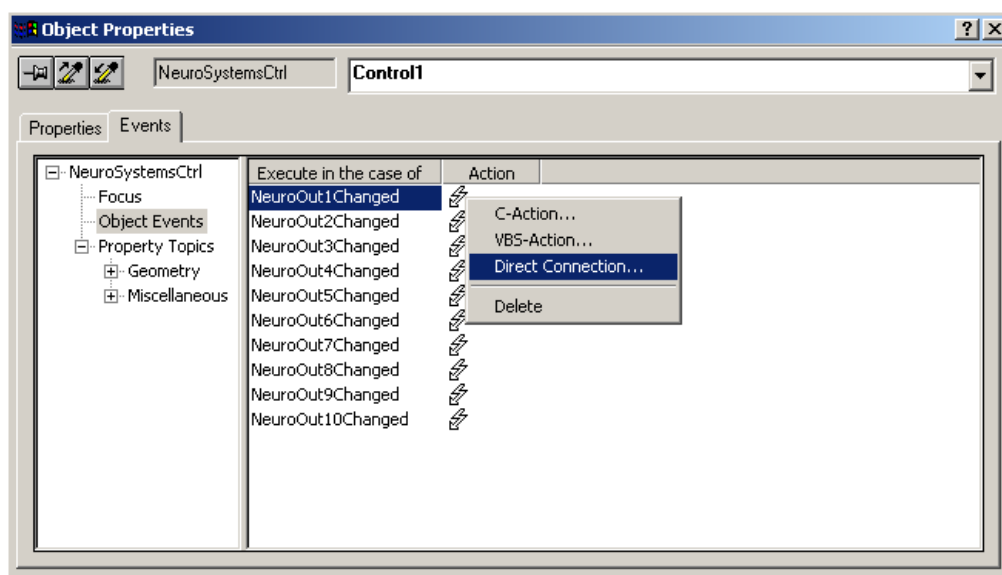


- Repeat this process for the control's second input property, 'NeuroIn2' and use the 'NeuroInput2' tag of course.
- Change the 'Update Cycle' to 'Upon Change' for each control property by right-clicking the 'Current' column.

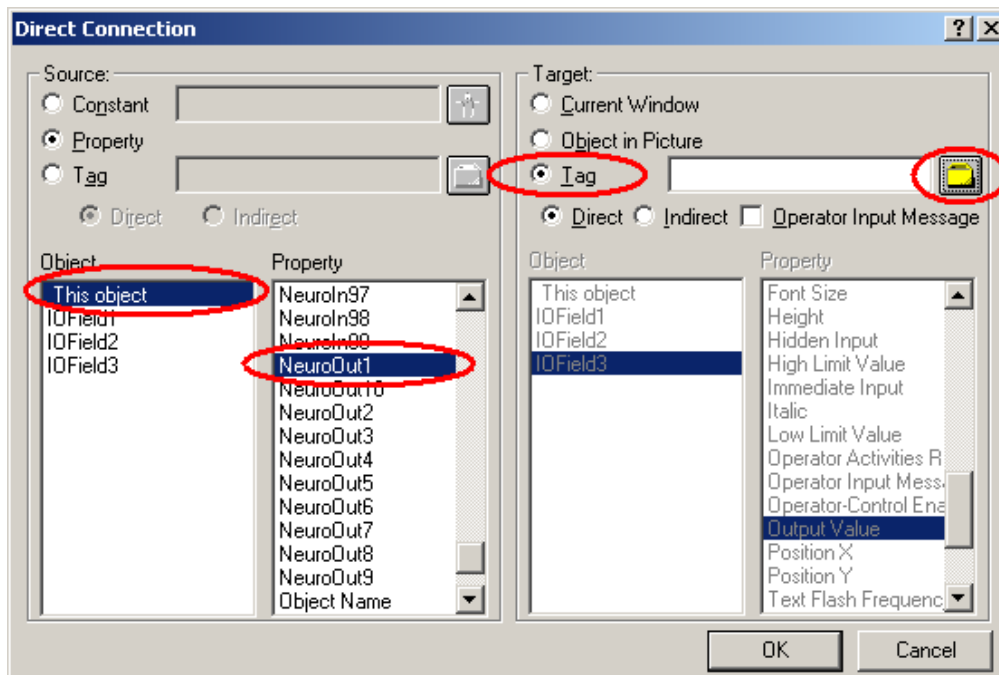




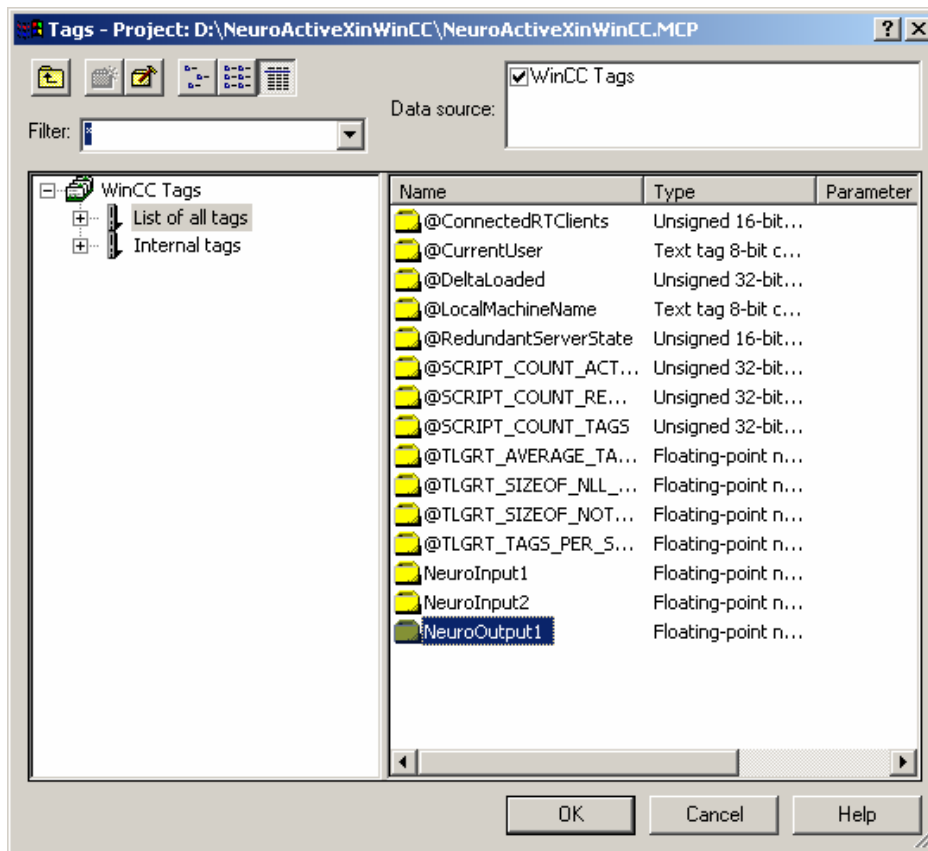
- The output properties will be handled slightly differently. We will use the 2 events 'NeuroOut1Changed' and 'NeuroOut2Changed' to update the value of the relevant tags – 'NeuroOutput1' and 'NeuroOutput2'. This can be done using a direct connection.
- Open the 'Events' tab of the control's property dialog box and then open 'Object Events'. There you will find 'NeuroOut1Changed' and when you right-click the corresponding lightning bolt, you will be able to select 'Direct Connection'.



- This time we will connect the first output property (Source), with a tag.



- To select the actual tag, click on the  button and choose 'NeuroOutput1' from the dialog that appears.

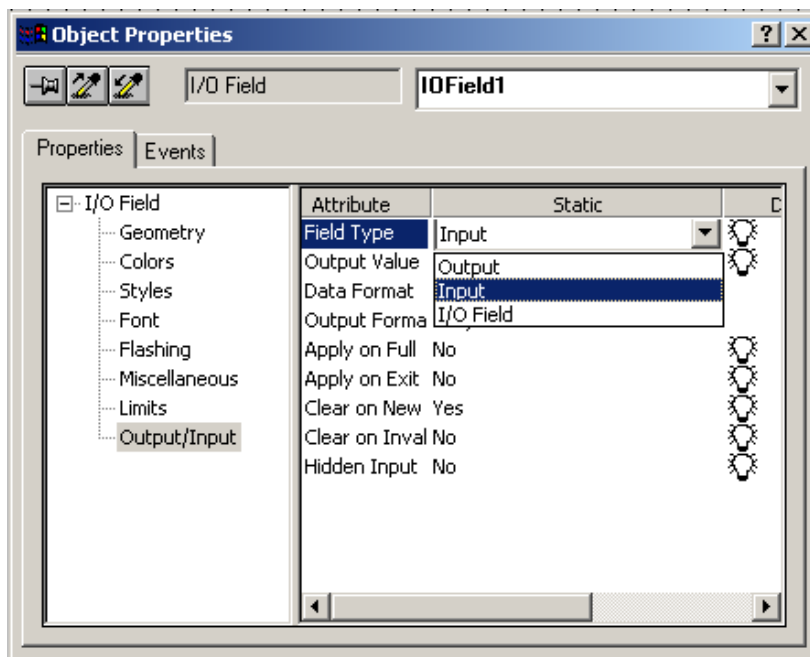


- The graphic can now be used in Runtime.

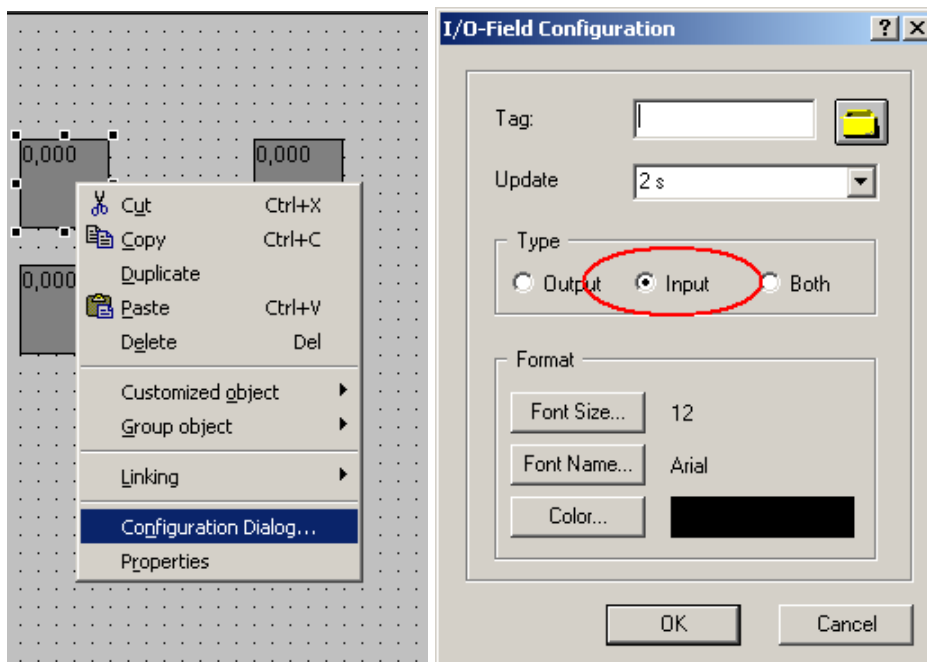
3.3.6.5.2 Using the Control with Direct Connections only

In this example we will use direct connections to connect the I/O fields to the control directly.

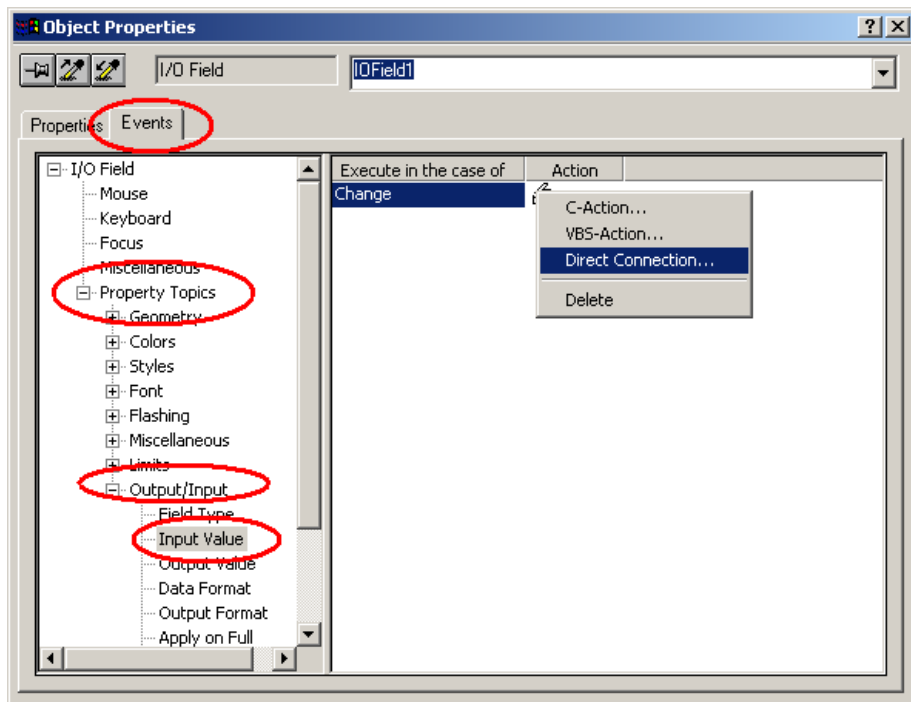
- Because we do not have any tags, the input fields must be configured to be just that, otherwise they will revert to their default value. You can do this from the I/O field property dialog box as shown.



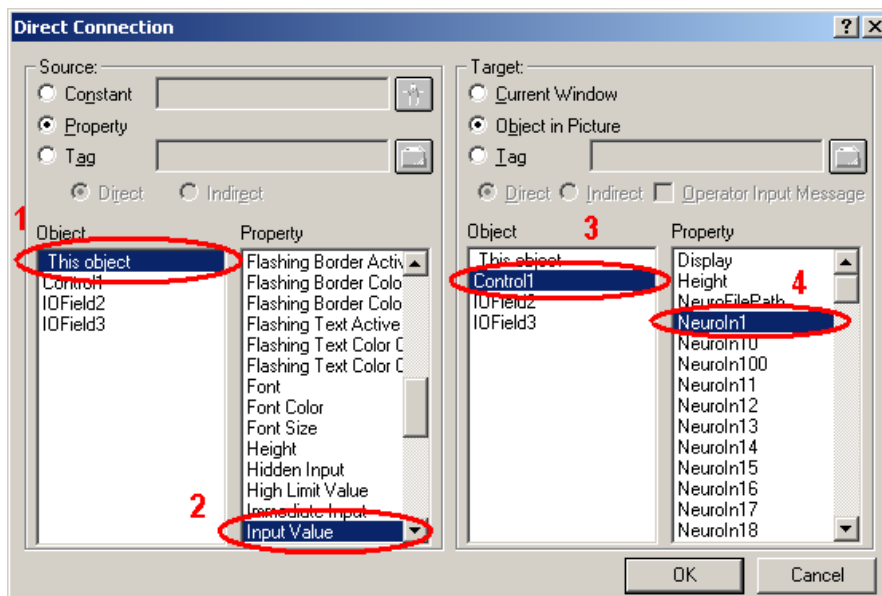
Or you can open the 'Configuration' dialog box by right clicking on the field and selection the option as shown.



- As you can see from the diagram, there is an 'Input/Output' section within 'Property Topics' section within the 'Events' tab and it is here that we find the 'Input Value' event.



- By right clicking on the lightning bolt for the 'Change' event we can select the 'Direct Connection' option and the following dialog appears.



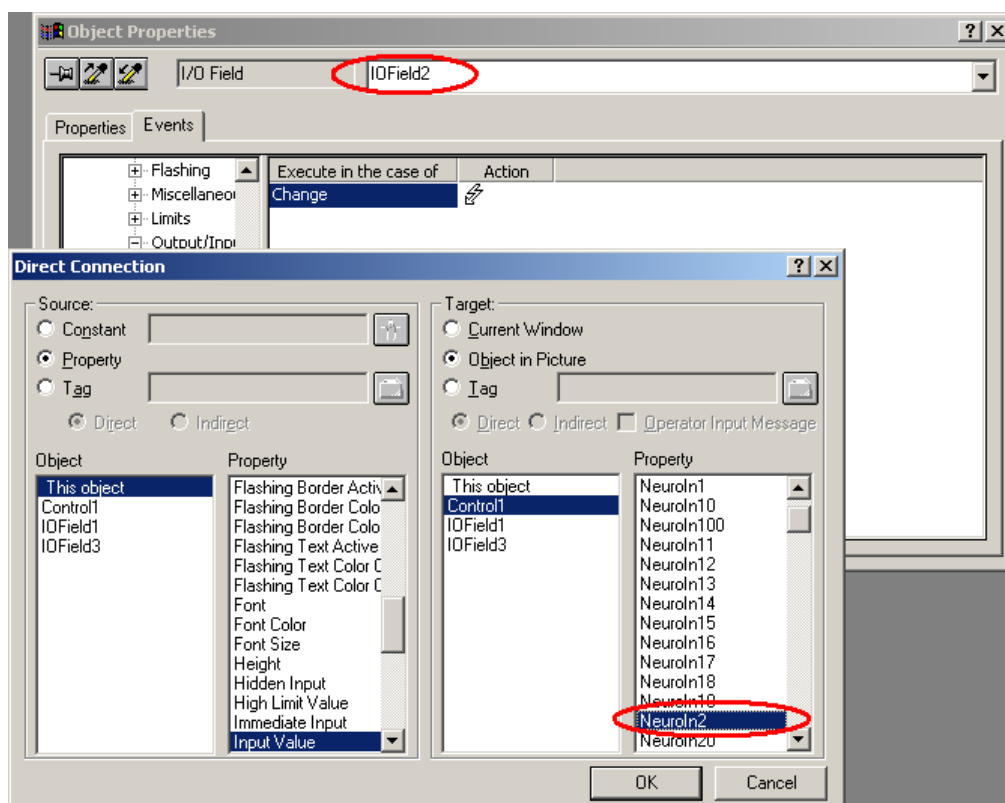
On the left hand side you can see that we have selected the 'Input Value' of 'This Object' ('IOField1' – the first I/O field) as the 'Source' and the 'Target' is defined as 'Control1' (the *NEUROSYSTEMS* ActiveX control) and more specifically, the 'NeuroIn1' property. The selection should be made in the order shown in the diagram to avoid confusion.



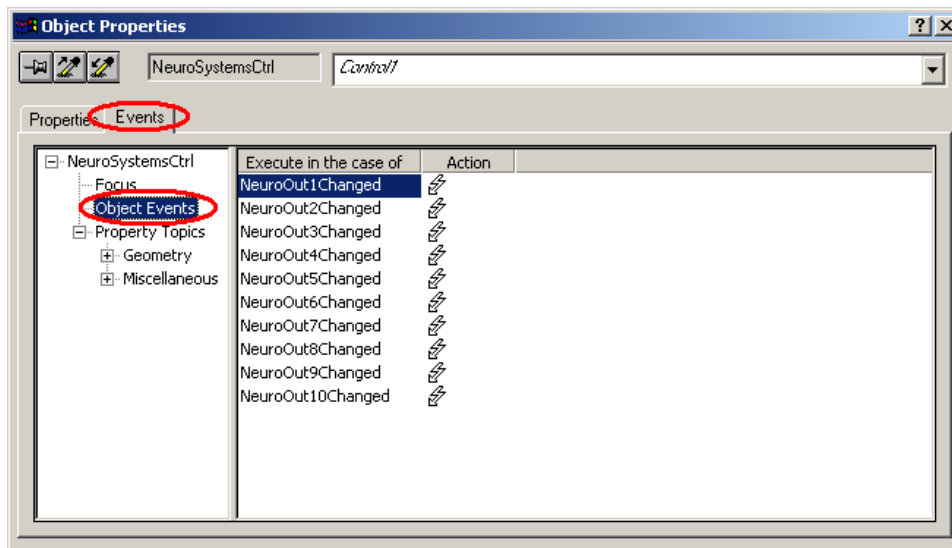
Click 'OK' when the connection is defined as shown. The lightning bolt will now be blue.

The name of the objects may be changed if you wish but the properties will always have the same names.

- Repeat the process of configuring a direct connection but use the second I/O field – 'IOField2' and 'NeuroIn2'.

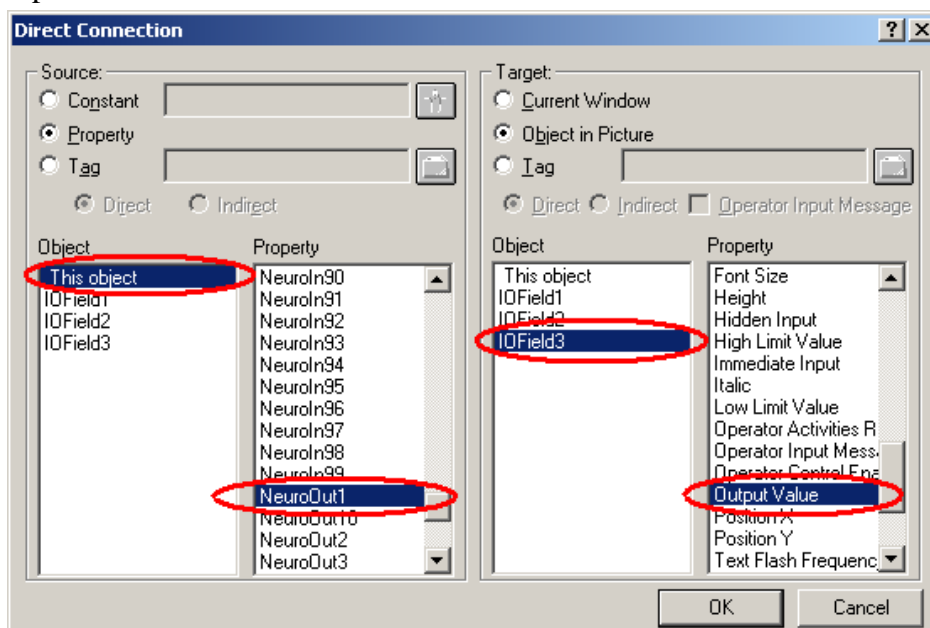


- For the Output fields we need to take a slightly different approach. This time we will configure an action when the outputs of the *NEUROSYSTEMS* ActiveX control changes. We begin by opening the object properties of the control in the same way as we opened the input properties of the I/O fields – right clicking on the object and selecting 'Properties'.



The relevant dialog box opens and you should open the 'Events' tab and then select the 'Object Events' category. You will then see the ten events for the *NEUROSYSTEMS* ActiveX control.

- In a similar manner to the configuration of the I/O fields, right click the lightning bolt for the first event – 'NeuroOut1Changed' and select 'Direct Connection' from the menu.
- This time we will connect the first output of the control to the output value of the third input field.



Confirm this connection using the 'OK' button



3.3.7 Using the Control in SIMATIC WinCC Flexible

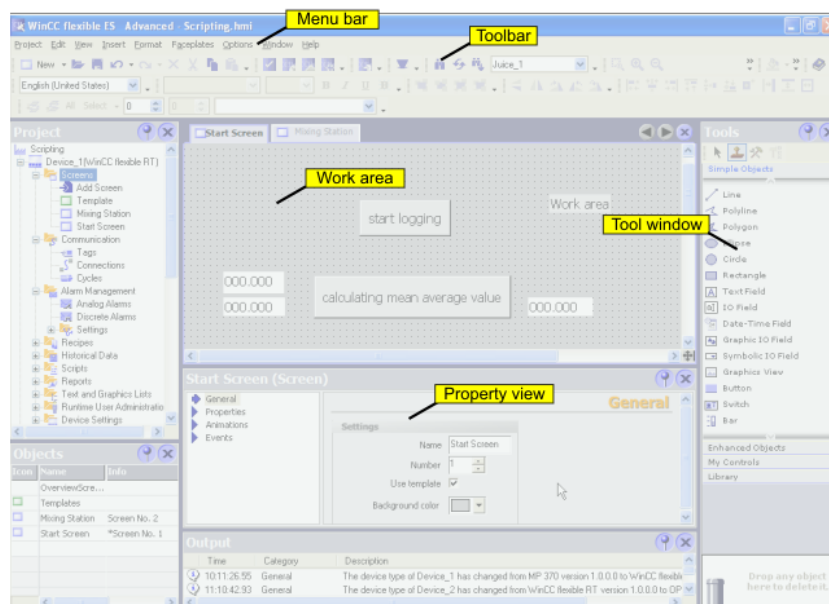
If you would like the control to be active all the time, you must also put the ActiveX control into the overview picture.

Note: Please be aware, that the *NEUROSYSTEMS* ActiveX control is only active when the picture is active.

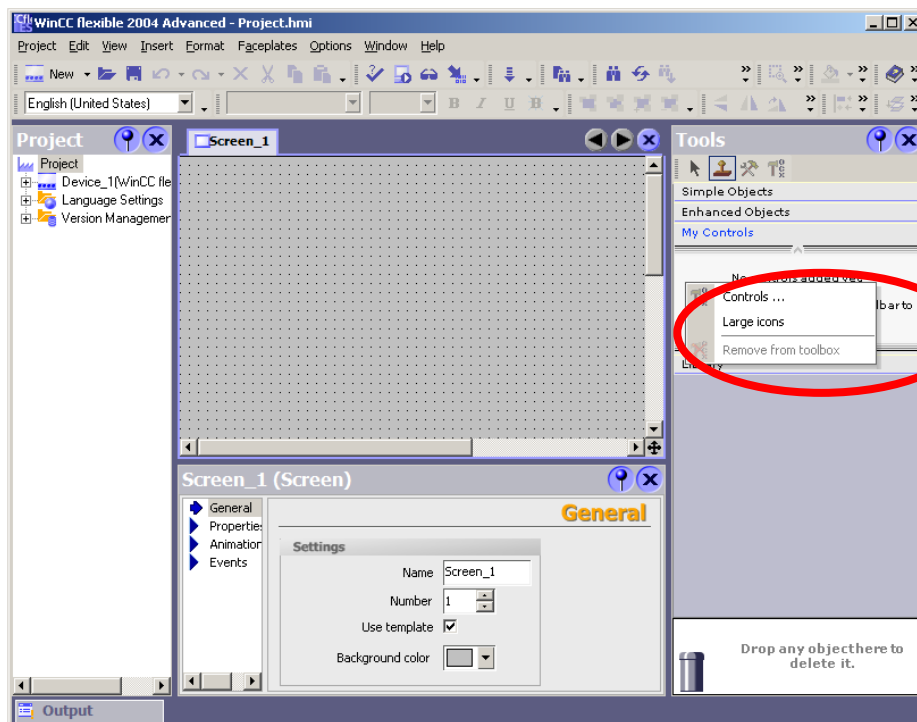
SIMATIC WinCC Flexible does not currently allow controls to be registered from within WinCC Flexible. Therefore the setup program for the *NEUROSYSTEMS* ActiveX Control should be used or the control can be registered manually.

3.3.7.1 Inserting the ActiveX Control in SIMATIC WinCC Flexible

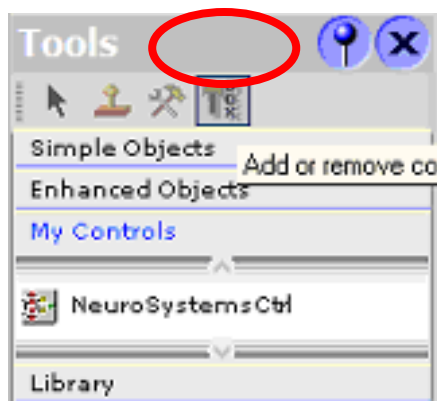
- Firstly, the “Tool Window” should be opened in your WinCC Flexible project. If it is not then select “View” and then “Tools” from the “Menu Bar” or press Ctrl+Shift+T.



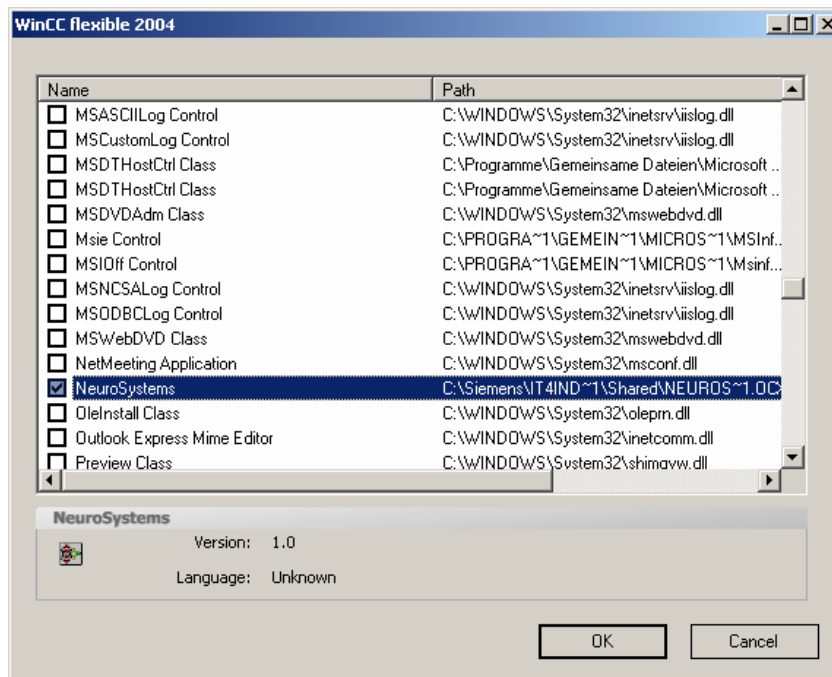
- There are two ways to open the “Add or Remove Controls” dialog box
 - In the “Tool Window”, select “My Controls”, right-click on the “My Controls” tab and select “Controls” .



- Or click on the “Add or Remove Controls” icon.



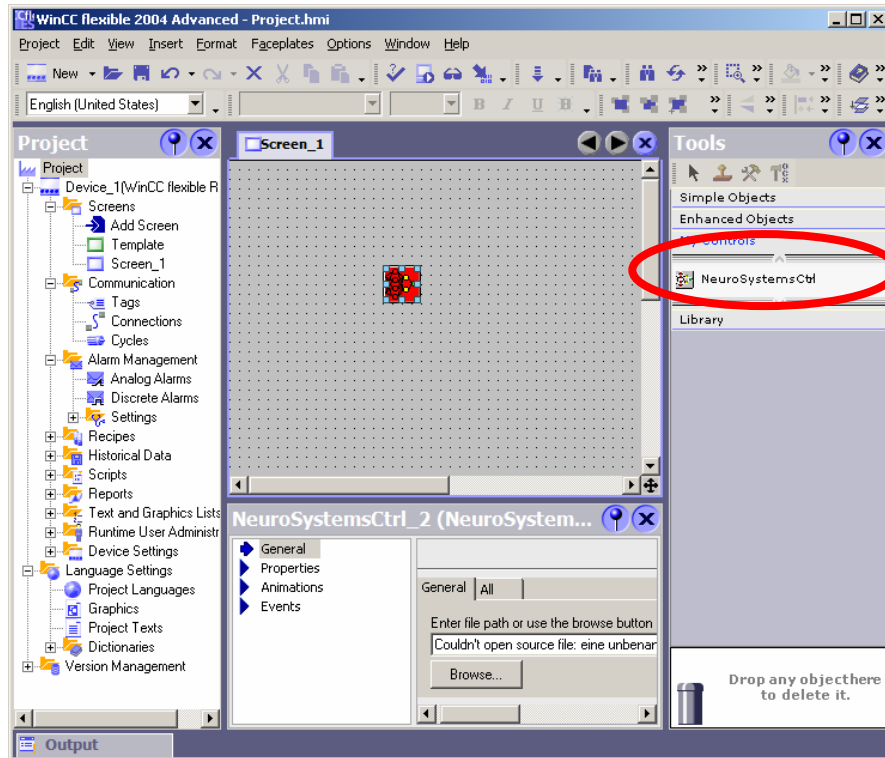
- Add the *NEUROSYSTEMS* Control to the list of controls by checking the checkbox beside the name of the control “*NEUROSYSTEMS*”.



- The window will close and *NEUROSYSTEMS* will be available in the “My Controls” section of the “Tool Window”. To remove it, reopen the “Add or Remove Controls” dialog and uncheck the checkbox.
- To select the control, click on it once with the left mouse button



- To insert it into a picture (while it is selected), simple click on the picture once with the left mouse button.



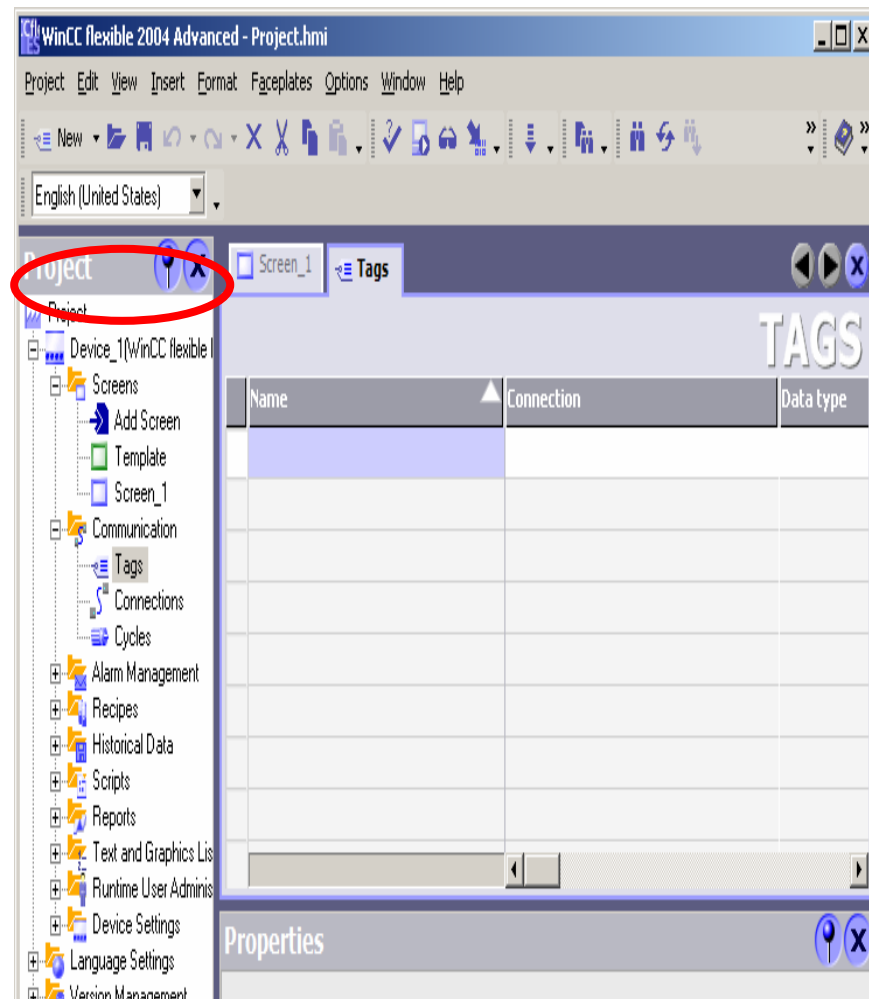
3.3.7.2 Example: Using the Control with SIMATIC WinCC Flexible and Scripts

In this example we will use two scripts to load the *NEUROSYSTEMS* file (.snl) and calculate results from two input fields. There are numerous other opportunities to these scripts – many involving no user interaction at all, e.g., loading the *NEUROSYSTEMS* file when the runtime screen is activated and calculating values when tags change value.

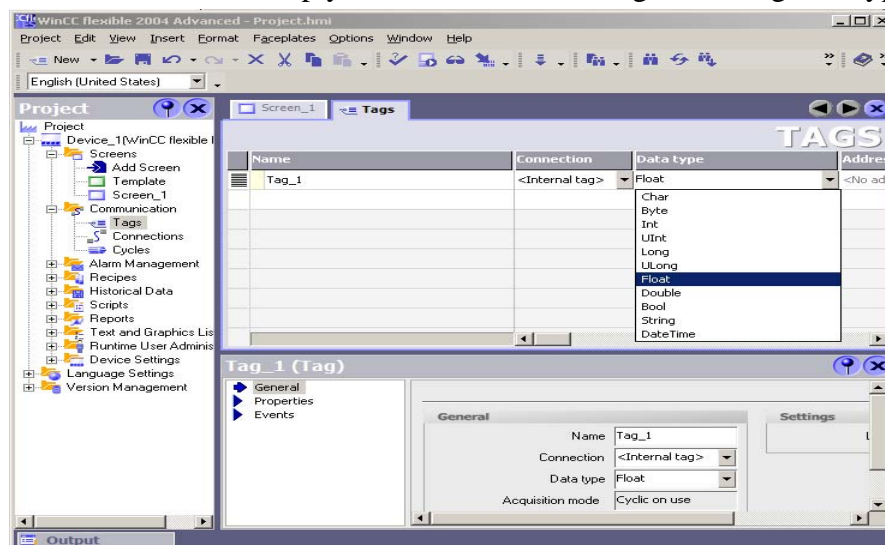
- **Create input and output tags**
 - In the project window, you should open the “Device” and then “Communication”



- Open the “Tags” window by double-clicking on “Tags”.

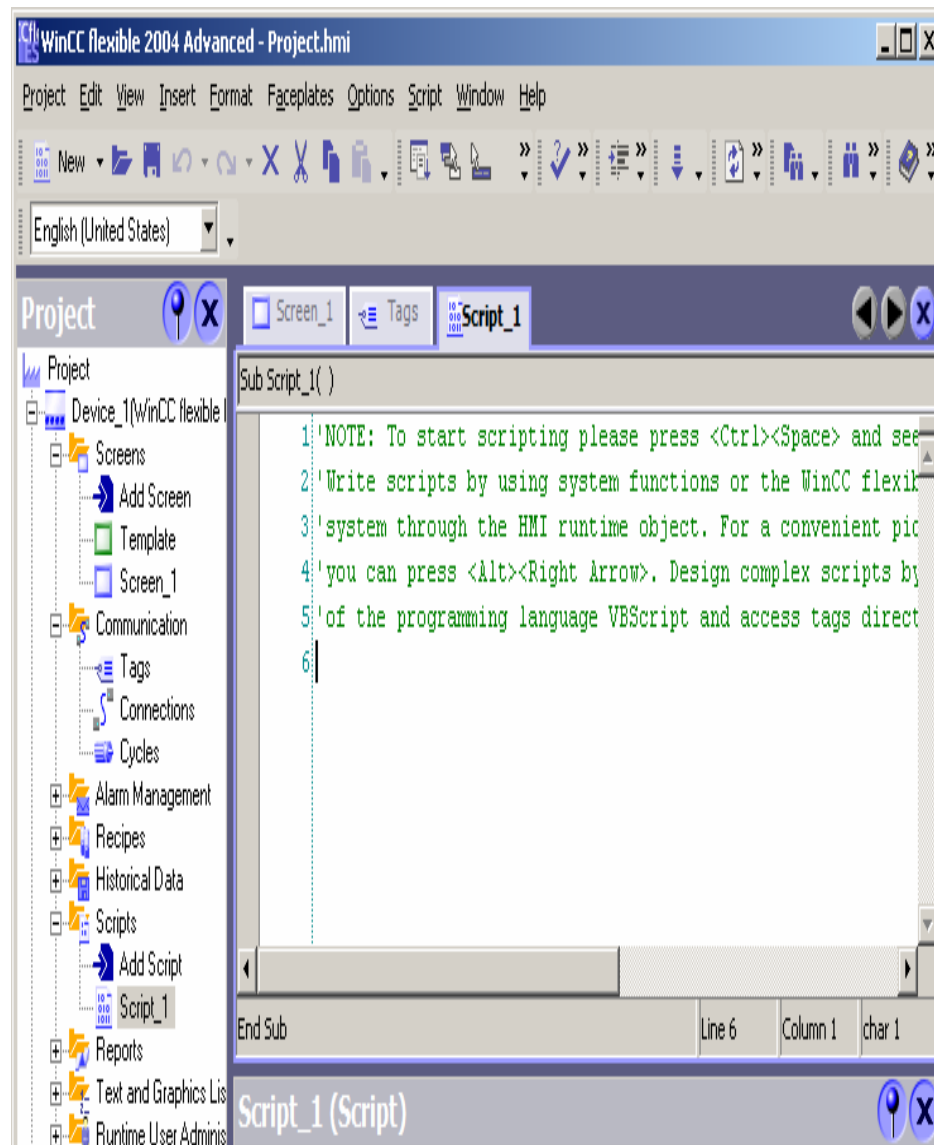


- Double-click on an empty row to create a new tag and change the type to float.

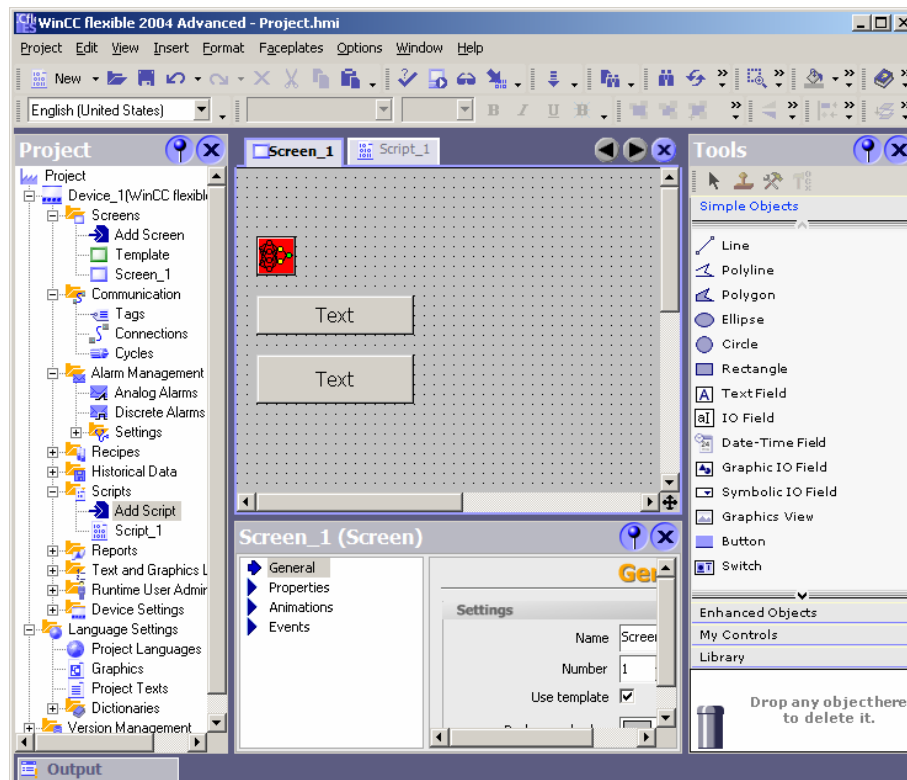




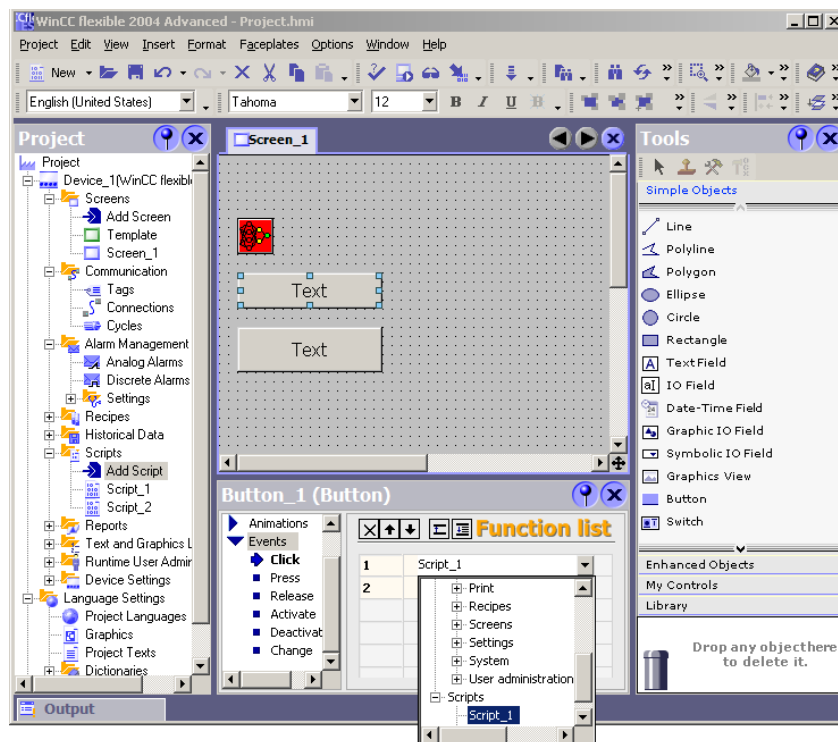
- Repeat this 2 more times so that you have 3 tags in total.
- **Create the scripts**
 - In the project window, you should open the “Device” and then “Scripts”.
 - Double click the “Add Script” option and an new script is created and opened.



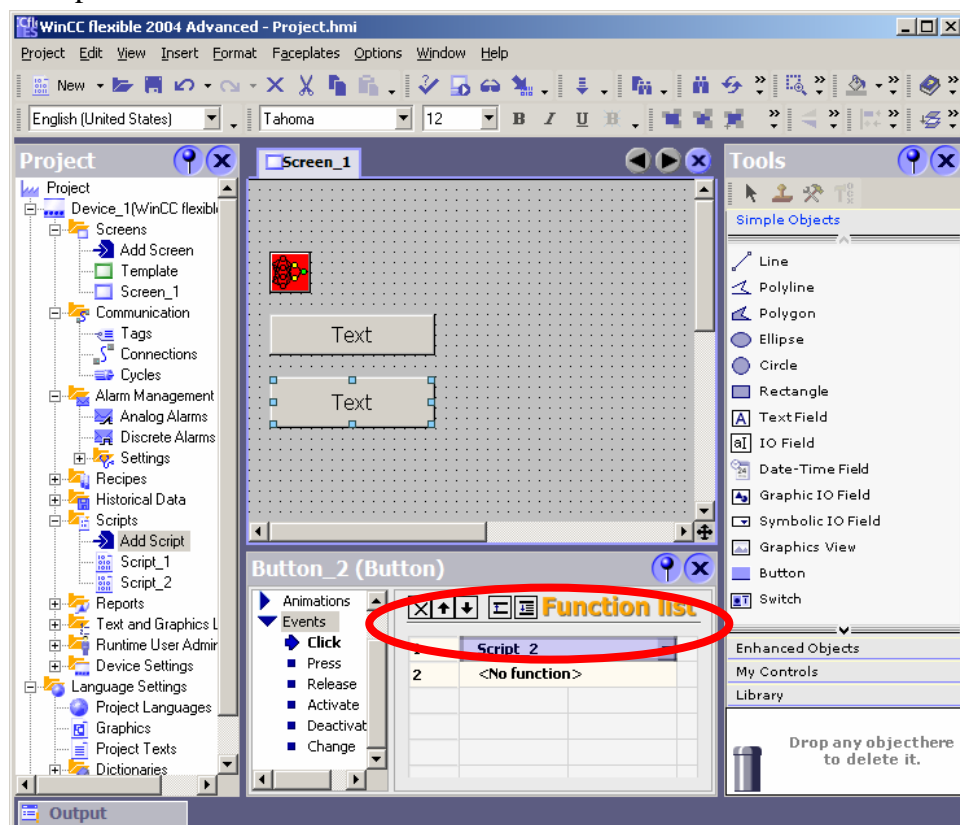
- Repeat this to create a second script.
- **Create buttons to activate the scripts**
 - Reopen “Screen_1” and use the “Simple Objects” tab in the “Tools Window” to insert two buttons.



- For the first button, open the “Properties Window” if it is not yet open by double-clicking the button.
- Change the text in the “General” properties section to “Load *NEUROSYSTEMS* file”.
- Open the “Events” section and open the drop-down list, which reads “No function”. Select “Script_1” from the “Scripts” section, which is found under the “System Functions” section. This means that “Script_1” will be executed for the default event of the first button – in this case, when it is clicked.

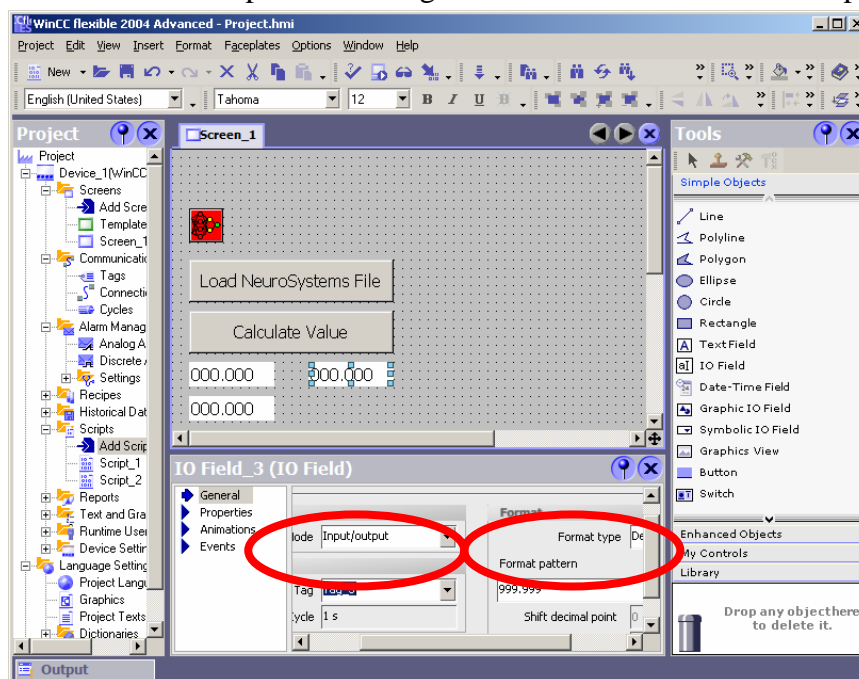


- Now change the text of the second button to “Calculate Values” and insert “Script_2” into the function list of the “Click” event of the second button.





- Resize the buttons to suit the size of the text entered.
- **Create the 3 IO fields**
 - Like you created the buttons, insert 3 IO fields into “Screen_1”
 - In the “General” section of the “Properties Window” Change the “Format Pattern” to “99.999” and assign a different tag to each IO field in the “Tag” property field. In this case we use “Tag_1”, “Tag_2”, “IO Field_1” and “IO Field_2” as the ‘inputs’ and “Tag_3” and “IO Field_3” as the outputs.



- **Write the appropriate scripts**
 - This example uses the first script to load the appropriate *NEUROSYSTEMS* (.snl) file. The code is given here:

```
Dim MyControl
```

```
Set MyControl = HmiRuntime.Screens("Screen_1").ScreenItems("NeuroSystemsCtrl_1")
```

```
MyControl.NeuroFilePath =
```

```
"C:\Simens\IT4\Insustry\NeuroSystems\ActiveX\Control\Ixor_learned.snl"
```

- The second script takes the values from the ‘input’ variables, processes them using the *NEUROSYSTEMS* ActiveX Control and writes the calculated values to the ‘output’ variables. The code is given here.

```
Dim MyControl
```

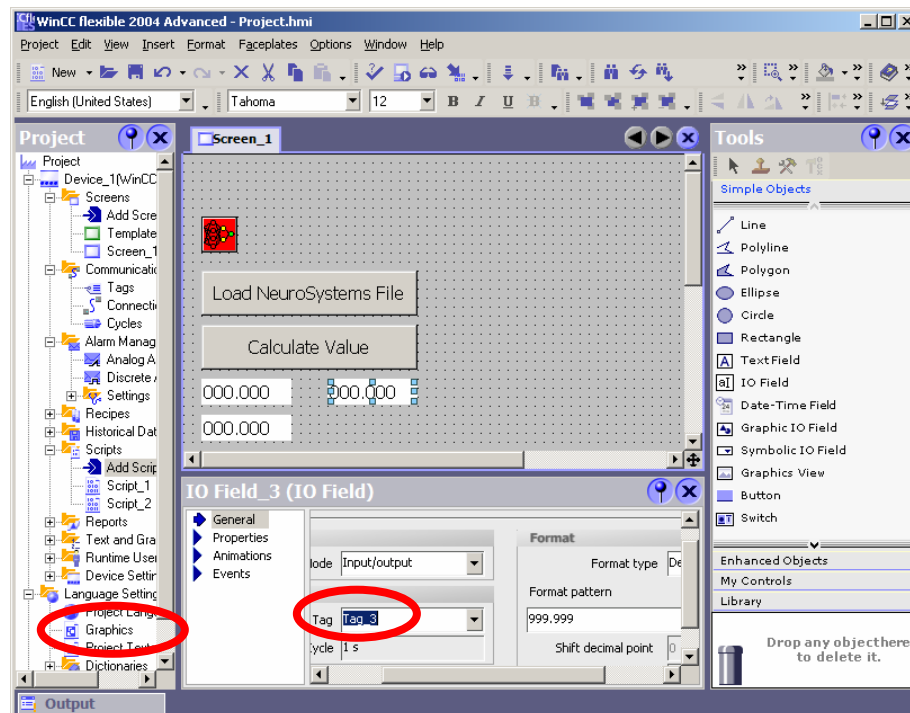
```
Set MyControl = HmiRuntime.Screens("Screen_1").ScreenItems("NeuroSystemsCtrl_1")
```



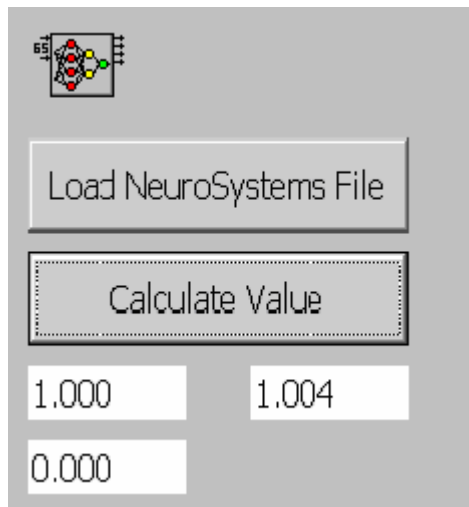

```
MyControl.NeuroIn1 = SmartTags("Tag_1")  
MyControl.NeuroIn2 = SmartTags("Tag_2")  
SmartTags("Tag_3") = MyControl.NeuroOut1
```

- **Testing the example**

- Save “Screen_1” and the scripts by clicking on the “Save Current Project” button.



- Then click the “Start Runtime System” button.



- Clicking on the first button, “Load *NEUROSYSTEMS* file” results in the status of the control changing from “Error Mode” to “Normal Operation”
- When the *NEUROSYSTEMS* file (.snl) has been successfully loaded, the second button, “Calculate Values” will calculate the values from the ‘input’ fields and return the results in the ‘output’ fields.

Note: When entering values into the input fields, use the Enter/Return buttons, otherwise the values entered will be deleted immediately.

3.3.8 Limitations

Please be aware, that the *NEUROSYSTEMS* ActiveX control is only active when the picture is active.

If you would like the control to be active all the time, you must also put the ActiveX control into the overview picture.

Note: Online-monitoring of input and output values such as that with the SIMATIC S7 component is **not** possible with the ActiveX Control.



3.4 OPC

OPC has no extra Runtime Module. Functionality is integrated in *Targetsystem OPC Connect*.

This chapter shall help explain general terms in reference to OPC, DCOM and their properties.

3.4.1 Basics OPC – OLE for Process Control

Introduction

OPC (OLE for Process Control) is a uniform, multi-vendor software interface. OPC Data Access (OPC DA) is based on the Windows technology COM (Component Object Model) and DCOM (Distributed Component Object Model). OPC XML, on the other hand, is based on the Internet standards XML, SOAP, and HTTP.

DCOM

DCOM expanded COM by adding the ability to access objects beyond the limits of one computer.

This basis allows a standardized data exchange between applications from industry, office, and manufacturing.

Previously, applications that access process data were restricted to the access mechanisms of the communications network. In OPC, devices and applications of different manufacturers can be uniformly combined.

The OPC client is an application that accesses process data of an OPC server. The OPC server is a program that offers a standardized software interface to applications of different vendors. The OPC server is an intermediate layer between these applications to process the process data, the various network protocols, and the interfaces for access to the data.

When exchanging data with OPC, you can use only devices with operating systems based on the Windows technology of COM and DCOM. Windows 2000 and Windows XP currently have this software interface.



Communications concept

An OPC configuration must consist of at least one OPC server and one OPC client. The OPC server is a DCOM application that transfers data to an OPC client for further processing. In the opposite direction, a client can also supply a server with data. The data is exchanged in the form of OPC items. These are addressed using their symbolic names that you assign during configuration of the OPC server.

Access mechanism

The open interface standard OPC uses the RPC (Remote Procedure Call) access mechanism. RPC is used to forward messages with which a distributed application can call up services on several computers in the network.

The OPC client is an application that requests process data from the OPC server over the OPC software interface.

The OPC server is a program that offers a standardized software interface to applications of different vendors. The OPC server is an intermediate layer between these applications to process the process data, the various network protocols, and the interfaces for access to the data.

Protocol profile

OPC can use all standard protocols available to DCOM (Distributed Component Object Model) on a computer to access data of an automation system via an OPC server. OPC is not restricted to any particular standard protocol. The preferred standard protocol for communication is the datagram TCP/IP.

3.4.2 DCOM settings

Data is exchanged between an OPC DA server and OPC client over the DCOM interface. Before communication is possible, the launch and access permissions of DCOM must be set correctly. The DCOM settings depend, for example, on the network configuration and security aspects.

Notice

The description below deals with the full enabling of the OPC DA server without considering safety aspects. With these settings, it is possible to communicate over OPC. There is, however, no guarantee that the functionality of other modules will not be impaired. We recommend that you ask your network administrator to make these settings. To configure



DCOM, you require fundamental knowledge of the network technology of Windows 2000 and Windows XP.

For more detailed information on DCOM, refer to the documentation of Windows 2000 and Windows XP. You configure DCOM using the program "dcomcnfg.exe."

Starting dcomcnfg in Windows 2000

1. Click on "Run" in the start menu of the operating system. Type in "dcomcnfg.exe." The "Properties of DCOM configuration" dialog opens.
2. Click on the "Applications" tab. Select desired OPC server as "Applications."
3. Click "Properties". The desired OPC server properties" dialog opens. Set the properties of the application.

Starting dcomcnfg in Windows XP

Windows XP basically provides the same configuration options for DCOM as the other versions of Windows. The difference is that the dialog layouts have changed.

1. Go to the Start menu of the operating system and select the command "Settings Control Panel".
2. Double-click on "Administrative tools Component services". The "Component services" dialog opens.
3. In the structure tree, expand
"Console Root\Component\Services\Computers\My Computer\DCOM Configuration"
4. Select desired OPC server as "Applications." Open the context-sensitive menu of desired OPC server and select "Properties." The dialog opens. Set the properties of the application.



3.4.3 Configuring DCOM on the OPC DA server

Introduction

Before the OPC client can launch the OPC DA server and establish the process communications connection successfully, the launch and access permissions of the OPC server must be set correctly.

Requirements

- You have started the "dcomnconf.exe" program.
- The desired OPC server properties dialog is open.

Procedure

1. Click on the "General" tab.
2. As the "Authentication level", select "None."
3. Click on the "Security" tab.
4. Click "Use custom access permissions."
5. Click the "Edit" button. The "Registry value permissions" dialog opens.
6. Add the users "Administrators", "Interactive", "Everyone", "Network", and "System" and select "Allow Access" as the "Type of Access." Click "OK" to close the dialog box.
7. Click "Use custom launch permissions."
8. Click the "Edit" button. The "Registry value permissions" dialog opens.
9. Add the users "Everyone" and "Network" and select "Allow Launch" as the "Type of Access." Click "OK" to close the dialog box.
10. Click on the "Location" tab. Click "Run application on this computer."
11. Click "OK" to close all open dialogs.



4 Operating Structure of NeuroSystems

This part of the manual provides a systematic overview of the options provided in *NEUROSYSTEMS*. It is ordered according to the menu items and operating structures available in the working window.

With the program *NEUROSYSTEMS* you can create a project. A *project* is a neural network specified by the choice of targetsystem, its inputs and outputs (number and properties), its network type and its structure. Editing is performed in a window with the name *NeuroSystems*. It is the working window (basic window) of *NEUROSYSTEMS*.

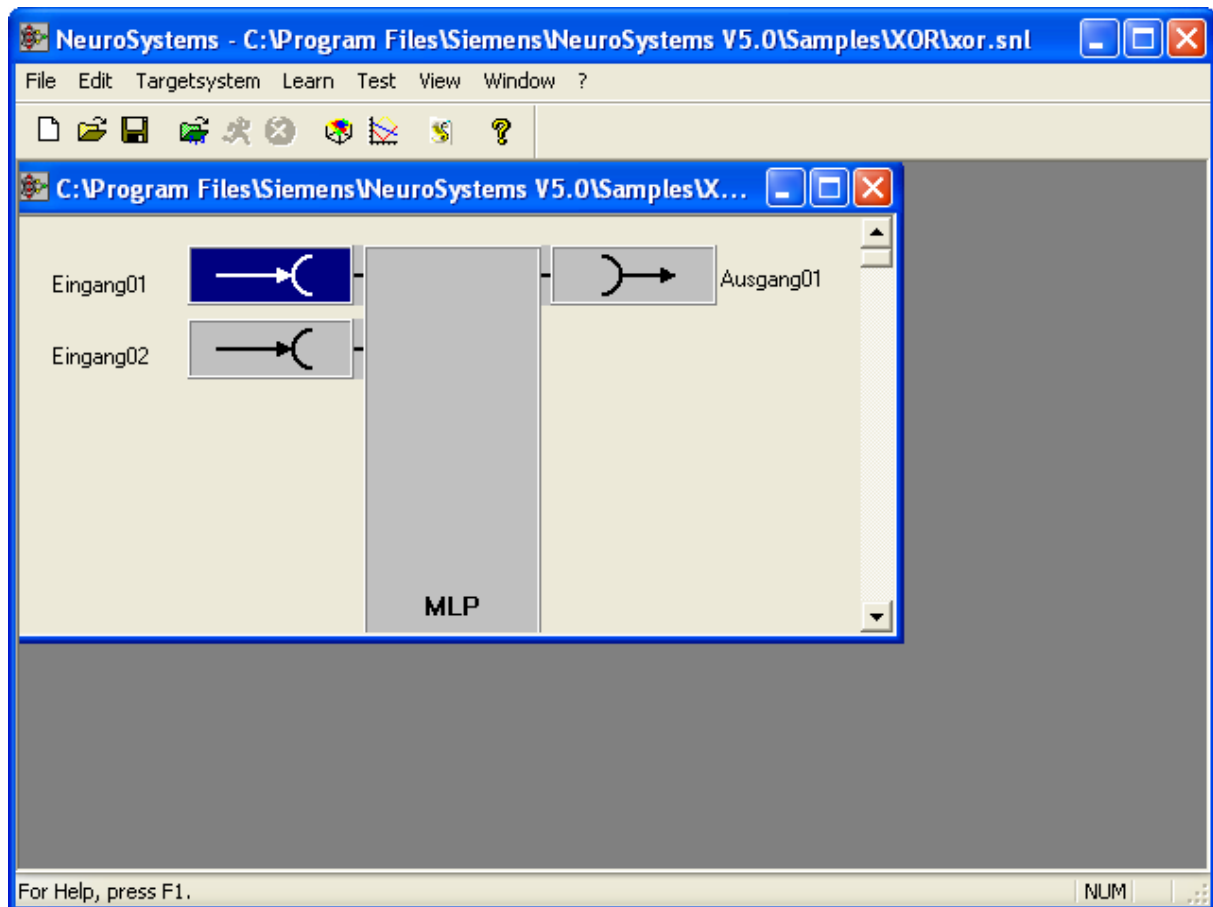
4.1 Working Window

When you first start *NEUROSYSTEMS* an incomplete working window is opened which does not yet contain a project. Only the menu items *File*, *View* and *Help* are available. The working window is only displayed completely when there is a open project. You can open and edit a project in one of two ways.

- Loading an existing project with *File/Open*
- Creating a new project with *File/New* and entering the external network structure (number of inputs and outputs and the targetsystem).

The complete working window (which is what we shall refer to as the "working window") shows the complete menu bar and a window with the block diagram of the neural network. This block diagram window plays a central role in processing the project and is therefore called the project window current project window contains all activities for editing the neural network. You can access them via the block symbols of the inputs and outputs and of the network which function as buttons. The activities are subdivided into two sections:

- Properties of the inputs and outputs and
- Properties of the network.



Operation of the program when you are working on a project is mainly based on the menu bar and the project window. Its blocks function as buttons. It must remain open during the entire time you are working on the project, but it can be minimized to icon size of course. If other windows are opened as you work on a project, they are given the project name and are numbered. The actual project window is then always labeled as number 1, e.g.

"C:\Programs\SIEMENS\NeuroSystems\Samples\NeuroSystems\xor.sn1:1".

If you close the project window you also close the windows associated with it and terminate the project.

In one session with *NEUROSYSTEMS* you can work on more than one project in parallel. In this case, several project windows are open in the working window (distinguished by the project name).

The menus of the working window and its sub items are usually activated as follows

- with a click of the left mouse button on the menu name in the menu bar or table or
- with the key combination <Alt + ... (letter underlined in the name in the menu)>.



Other options for operation are stated with the menu items below under "Shortcut:".

Example: Key combination:

To close a file activate the sub item Close in the File menu. You can also use the key combinations <Alt+F, C>.

Note: From now on, the abbreviation *LMB* is used for "left mouse button" and *RMB* is used for "right mouse button".


Menu items buttons are activated by clicking them with the *LMB*.

4.2 Menu: File

With this menu you can:

- begin and exit project work,
- save a project,
- exchange *.fpl project files with the Siemens program tools *FUZZYCONTROL++*, *FUZZYCONTROL*, and *PROFUZZY*.

4.2.1 New

Shortcut: In the toolbar click with the LMB on the icon 
Key combination <CTRL+N> or <ALT+F, N>

With this command you create a new project. It opens the dialog box *New...*

Dialog box *New...*

In the dialog box "*New...*" you can set the number of inputs and outputs and the targetsystem for the project.

The maximum number of inputs and outputs depend on the targetsystem you plan your neural network for:



Targetsystem	inputs	outputs	neurons	weights
S7-4K:	4	4	99	199
S7-20K:	100	10	160	1660
WinCC:	10	5	no limit	no limit
ActiveX:	100	10	no limit	no limit
OPC:	100	10	no limit	no limit
CFC-4K:	4	4	?	?
CFC-20K:	100	10	?	?

S7-4K as well as S7-20K is running under SIMATIC S7-300 and SIMATIC S7-400.

If you use a **neurofuzzy network (NFN)**, there is general restriction to a maximum of **8 inputs and 4 outputs**.

The definition made here can still be modified later on in the *Edit* menu, where you can add or delete inputs and outputs (within the limits mentioned above).


The S7-4K, S7-20K, WinCC, ActiveX and OPC can be selected as the targetsystem you want to design the network for. The default setting is S7-4K. Correct selection of the targetsystem avoids any possible memory and capacity problems when loading the trained networks to the targetsystems later on. The setting made here can still be modified later on using the menu item *Targetsystem/Selection*. However, you should avoid this, because some operations in *NEUROSYSTEMS* work dependent on the targetsystem setting. If you alter the setting later on, the performance and the exactness of the neural network eventually can not reach the maximum possible level. **So please select the correct targetsystem when you create your project and do not change it later on, if possible!**

Once you have closed the window by clicking *OK*, the project is defined and is given the initial name *New1*. The corresponding project window is displayed with a symbolic representation of the neural network. If you create several new projects one after the other, they appear with the initial names *New1*, *New2*, etc. in the working window *NEUROSYSTEMS*.



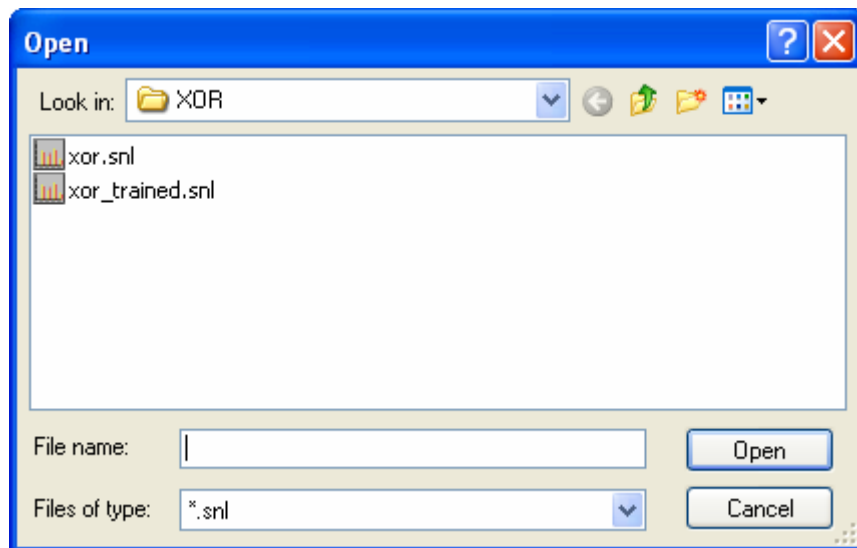
4.2.2 Open

Shortcut:

In the toolbar click with the LMB on the icon 

Key combination <CTRL+O> or <ALT+F, O>


Use this command if you want to resume work on an previously saved *NEUROSYSTEMS* project.
The Windows dialog box *Open* appears



If you mark the filename in the list with the LMB and click on *Open*, the project, here *xor.sn1*, appears in the working window in the form of the associated project window. With *NEUROSYSTEMS* you can edit more than one project in parallel. You can therefore open more than one project, in which case, each appears in a project window, with the project name as its title, in the working window *NeuroSystems*.

4.2.3 Close

Shortcut:

In the project window with a single LMB click on the icon 

with a double LMB click on the icon 

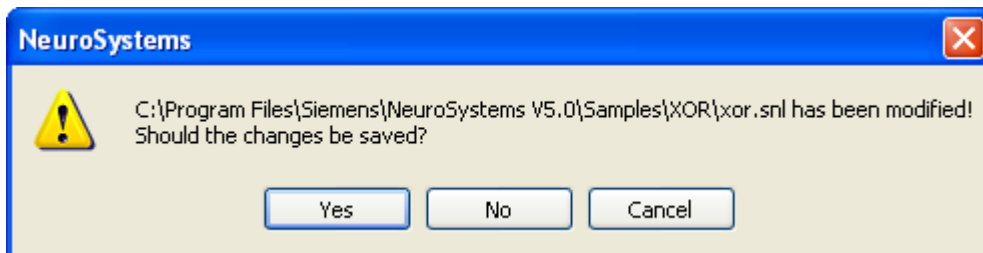
Hotkeys

<ALT+F, C>



This command terminates the current project and closes the project window and all associated windows. The working window of *NEUROSYSTEMS* remains.

If changes were made to the project (which is usually the case), *NEUROSYSTEMS* asks whether you want the project to be saved under the same name.



If you answer “Yes”, the previous version is overwritten with the current version. If you answer *No* any changes that were made are lost and the project remains in its previous state.

If you close a newly created project, *NEUROSYSTEMS* opens the dialog box *Save as*, where you can define the name of the project before it is saved.

4.2.4 Save

Shortcut:

In the toolbar click with the LMB on the icon



Key combination <CTRL+S> or <ALT+F, S>

Use this command to save the current project under its current name and directory. The project is saved with the extension **.sn1* (*SIEMENS NEURO LANGUAGE*).

4.2.5 Save As ...

Shortcut:

Key combination <ALT+F, A>

If you want to change the name and/or the directory of an existing project, you must pick the command *Save as*. When a project is first saved, *NEUROSYSTEMS* displays the dialog box *Save as*, so that you can give your project a name. A project is saved with the extension **.sn1* (*SIEMENS NEURO LANGUAGE*).



4.2.6 Import / Export

Shortcut: Key combination <ALT+F, I/E>



These commands are used to exchange *.fpl files between *NEUROSYSTEMS* and the *FUZZYCONTROL++* tool from Siemens AG. With *Import* you can import a fuzzy *.fpl file created with the *FUZZYCONTROL++* tool and edit it as an NFN-Structure with *NEUROSYSTEMS*. With *Export* you can export an NFN (NeuroFuzzy Network) configured with *NEUROSYSTEMS* as a fuzzy *.fpl file, and edit it with the *FUZZYCONTROL++* tool for example.

You can also exchange *.fpl files with the Siemens programs *FUZZYCONTROL*, and *PROFUZZY*.

4.2.7 1, 2, 3, 4

With this menu item you can re-open directly one of the last four projects you have been closed. The project at the top of the list was the last closed.

4.2.8 Exit

Shortcut: In the working window *NeuroSystems* with a single LMB click on the icon  or a double click on 

Key combination <Alt+F4> or <Alt+F, X>

Use this command to exit your *NEUROSYSTEMS* session. If changes were made to the project, *NEUROSYSTEMS* asks whether you want to save the project under its previous name.



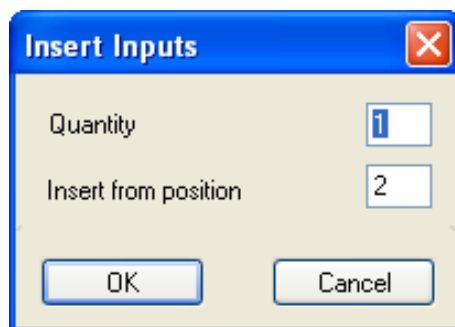
4.3 Menu: Edit

The *Edit* menu is used to add or remove inputs and outputs in the current project. Moreover, you can edit the network type.

4.3.1 Inserting Inputs/Outputs

Shortcut: Click on the input/output concerned with the RMB in the project window and select *Insert* or *Delete*. To edit the network just double-click it with the LMB.

Hotkeys: <ALT+E, I, I> (Input)
<ALT+E, O, I> (Output)



This dialog box appears for inserting inputs. You must state the number of inputs to be inserted and the position at which they are to be inserted. The input which is uppermost in the block diagram is at position 1. After confirmation with *OK* the inputs are inserted at the required location and the existing inputs are shifted downward.

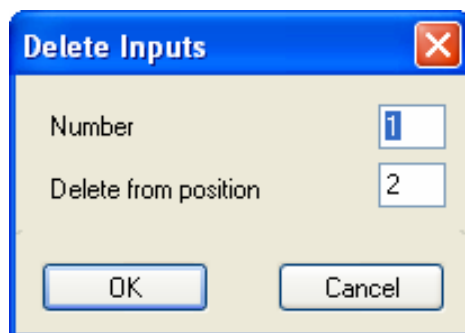
Insertion of outputs is performed analogously.

Note: Inserting in- or outputs is not possible, if learn or test windows are open.



4.3.2 Deleting Inputs/Outputs

- Shortcut:** Click on the input/output concerned with the RMB in the project window and select *Delete*.
- Hotkeys:** <ALT+E, I, D> (Input)
<ALT+E, O, D> (Output)



You can remove inputs with this dialog box. You must state the number inputs to be deleted and the position from which they are to be removed. The input at the specified position is deleted. The input which is uppermost in the block diagram is at position 1. After you have clicked *OK* a query is displayed, which you must confirm to delete the inputs. There must always be at least one input and one output.

Deletion of outputs is performed analogously.

Note: Deleting in- or outputs is not possible, if learn or test windows are open.

4.3.3 Editing Inputs/Outputs

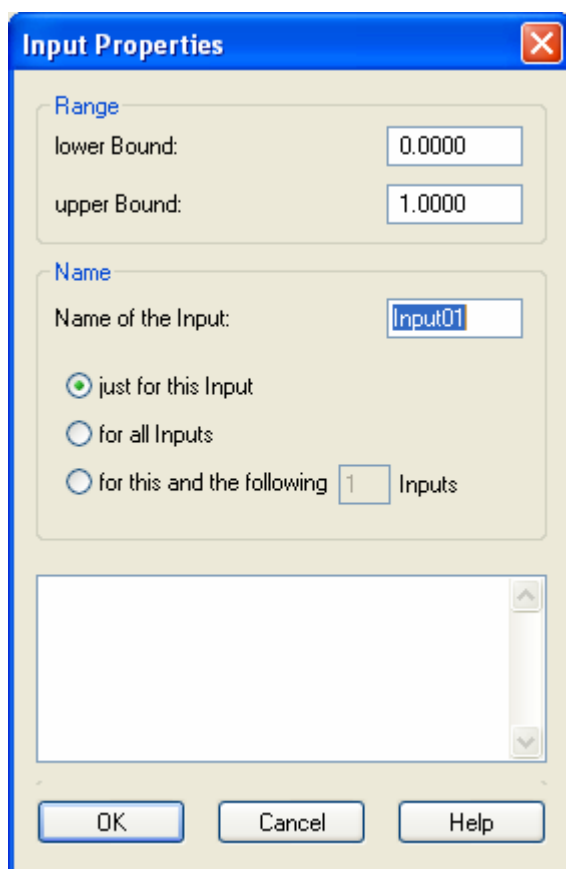
- Shortcut:** Double-click with the LMB on the button of the affected input.
Click with the RMB on the input concerned and select *Properties...*
- Hotkeys** <ALT+E, I, P> (Input)
<ALT+E, O, P> (Output)



The properties of inputs and outputs are the same. They are:

- *Range*, which can be changed or displayed
- *Name*, which can be changed
- *Text input*

We shall explain the above properties using the example of the *Input Properties* dialog box shown in the following figure.



Range

The smallest and largest numerical values that occur for the selected input are displayed as the entries for *minimum* and *maximum*. These indications refer to the currently selected input and can be changed manually only for this input. Further please keep in mind that the minimum must be smaller than or equal to the maximum. Appropriate selection of the limits is important for the success of the neural network's learning and for the graphic displays (e.g. 3-D graphic representation).



Note: Once you have normalized a network, you should not change the values. If this is necessary (e.g. because of new data), you must retrain the network to avoid erroneous results.

Name

In the enter box, you can enter a name relevant to the problem being solved for this input. The name may contain up to ten characters. The first character must not be a number. Special characters and accented characters are also not permitted. If you would like to use the name for the other inputs, you must provide it with a continuous numeration and it will be applied to the inputs.

Text input

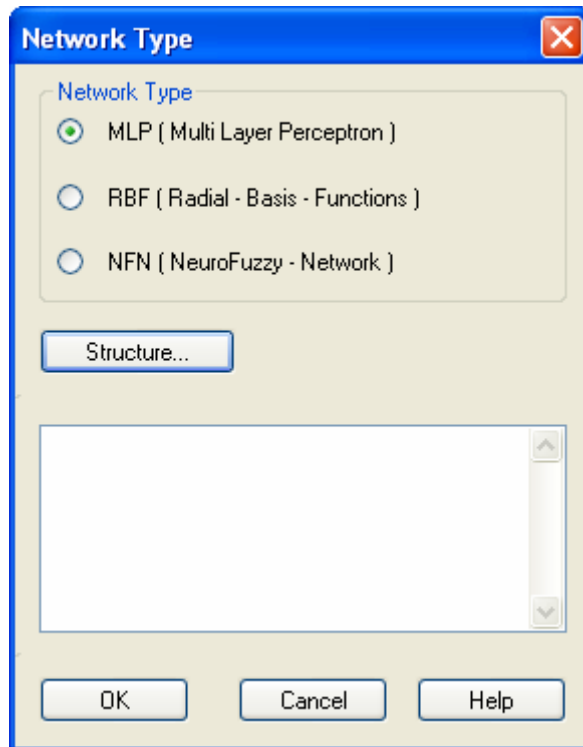
In the lower textbox you can enter an according text for the in- or output.



4.3.4 Editing the Network Type

Shortcut: Double-click on the network block with the LMB
Click with the RMB on the network and select *Network Type...*

Hotkeys: <ALT+E, N



By clicking on a radio button with the LMB you can select one of the three possible network types. Each type of network has its own options. The settings you can make for the network type you select are made in the ... *Structure* dialog box, which you can open with the *Structure* button.

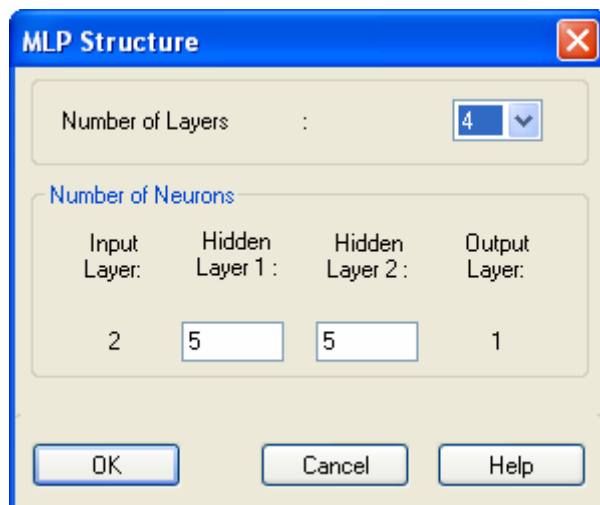
Text input

In the lower textbox you can enter an according text for the in- or output.



4.3.4.1 MLP Network

You can select one or two hidden layers. The network can therefore have 3 to 4 layers. In each hidden layer you can choose 1 to 50 neurons. Wrong values are rejected by the program (message box). The number of neurons in the input and output layer is the same as the number of inputs and outputs defined in the *Edit* menu.



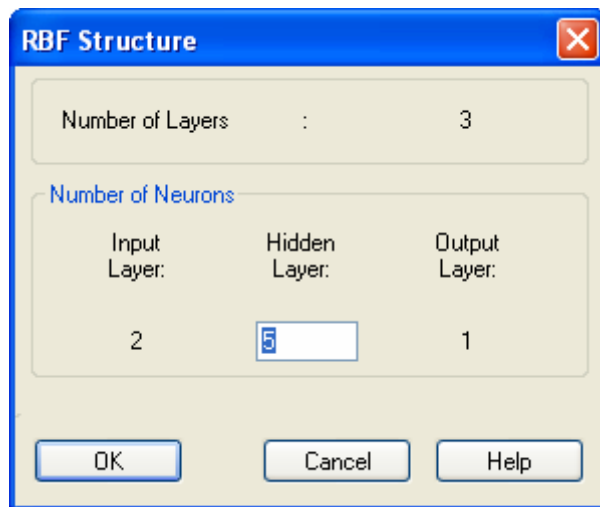
The dialog box titled "MLP Structure" contains the following fields and buttons:

- Number of Layers:** A dropdown menu showing the value 4.
- Number of Neurons:** A section containing four input fields:
 - Input Layer:** 2
 - Hidden Layer 1:** 5
 - Hidden Layer 2:** 5
 - Output Layer:** 1
- Buttons:** OK, Cancel, and Help.



4.3.4.2 RBF network:

This type of network always has 3 layers. It therefore always has one hidden layer, for which you can select 1 to 50 neurons. Impossible numeric values are rejected by the program (message box). The number of neurons in the input and output layer is the same as the number of inputs and outputs defined in the *Edit* menu.



The RBF Structure dialog box is shown with a blue title bar and a close button. It contains the following fields:

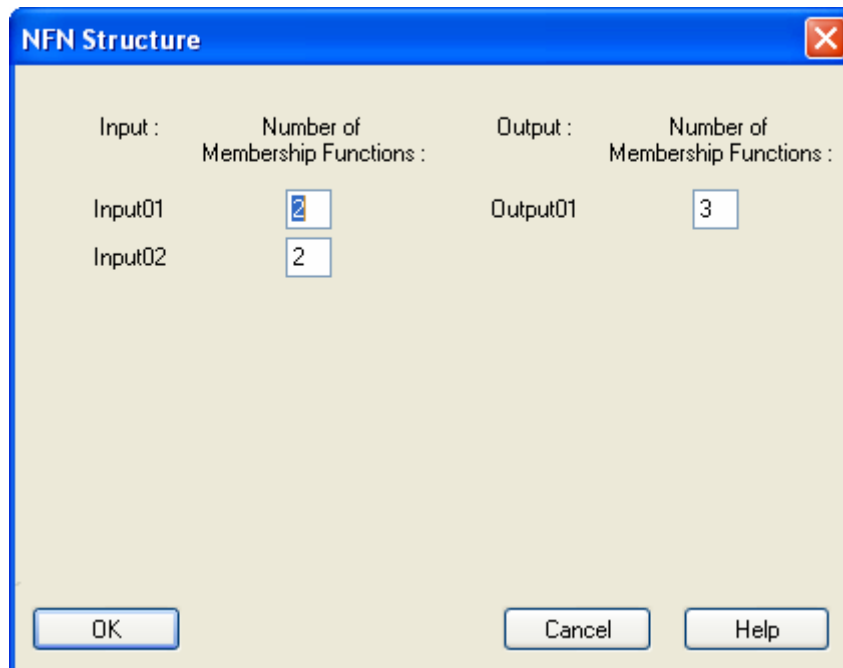
Number of Layers		
Number of Layers	:	3

Number of Neurons		
Input Layer:	Hidden Layer:	Output Layer:
2	5	1

At the bottom are three buttons: OK, Cancel, and Help.

4.3.4.3 NFN (neurofuzzy network):

When using an NFN there are a maximum number of **8 inputs** and **4 outputs** at your disposal. You can enter the number of membership functions (linguistic values, fuzzy sets) for each of your inputs and outputs (linguistic variables). Trapezoidal membership functions (MSF) are used for the inputs, singletons for the outputs. There is a maximum of **7 membership functions per input** and **9 membership functions per output**. In addition, the product of the number of all input membership functions has to be less or equal to 2000.



The NFN Structure dialog box is shown with a blue title bar and a close button. It contains two columns of input/output labels and their corresponding membership function counts. The 'Input' column has 'Input01' and 'Input02' with values 2 and 2 respectively. The 'Output' column has 'Output01' with a value of 3. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

Input :	Number of Membership Functions :	Output :	Number of Membership Functions :
Input01	2	Output01	3
Input02	2		

Example:

3 inputs with 5 MSF each and 2 inputs with 4 MSF each $5*5*5*4*4=2000$: possible!

4 inputs with 5 MSF each and 1 input with 4 MSF each $5*5*5*5*4=2500$: **not** possible!

If you choose this network type the fuzzy membership functions of the inputs and outputs will be adapted automatically during the learning process and a suitable rule base will be generated. *NEUROSYSTEMS* will only generate rules with AND operations in the IF part. The Sugeno inference method (sum-min-inference with singletons) will be used.

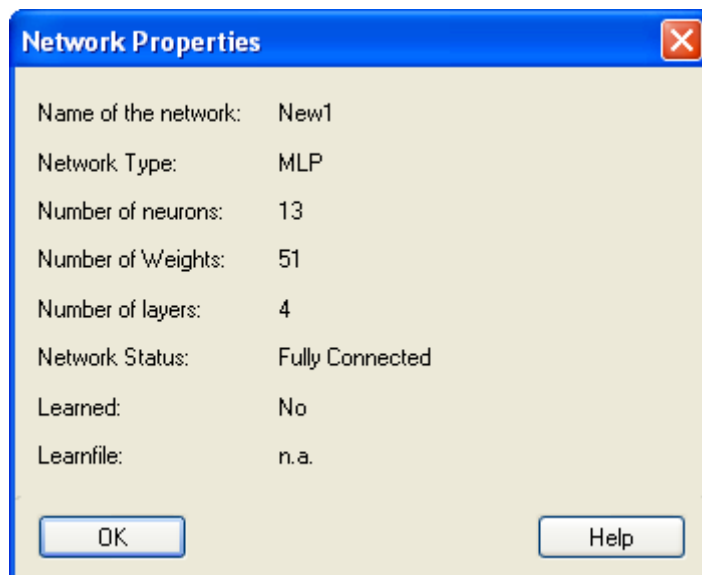
After a successful learning run, you can export the result to *FUZZYCONTROL++* (see *File/Export* menu).

Note: You can already take a look at the fuzzy rules generated in the *.fpl file, which contains the necessary information as an ASCII text.



4.3.4.4 Network Properties

The network is characterized by its type and its structure. You can access the settings for the network via the button representing the network in the block diagram. You can select the set options with two dialog boxes *Network Type* and *Structure*. Please refer Part I of this Manual for information about the structures of the neural networks used here and how they work. You can obtain information about the properties of the current network in the *Network Properties* window, which you can open by clicking with the RMB on the network block and selecting *Properties*.



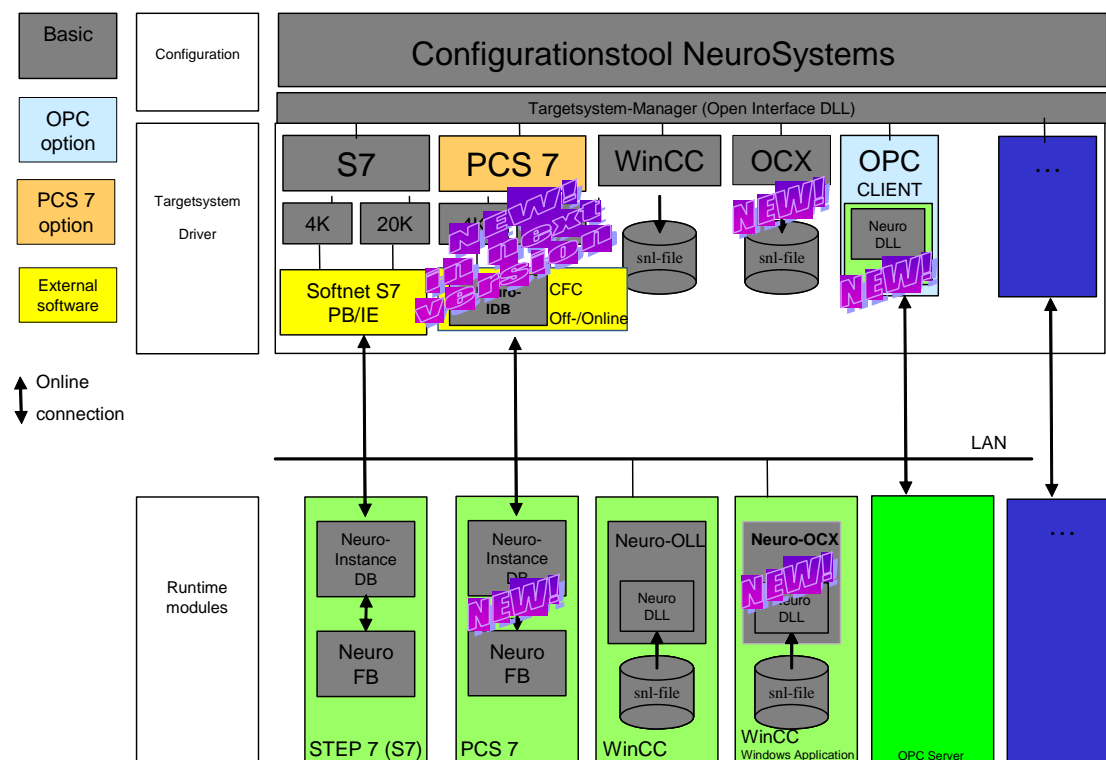


4.4 Menu: Targetsystem

The runtime modules for the targetsystems S7, ActiveX, OPC and SIMATIC WinCC compute the output values to given input values online, using the algorithm of the parameterized neural network. If you use S7 the network information will be transmitted to the targetsystem by writing an S7 data block (DB). If you use SIMATIC WinCC and ActiveXControl the network information will be handed to the runtime module (Neuro.OLL) via an *.snl-file.

The menu items "Targetsystem/Manager" and "Targetsystem/Selection" are made available for all targetsystems. The menu items "Targetsystem/Connect", "Targetsystem/Disconnect", "Targetsystem/Read" and "Targetsystem/Write" are depending on selected targetsystems.

This picture shows an overview of the interfaces, drivers and targetsystems (Runtime Modules):

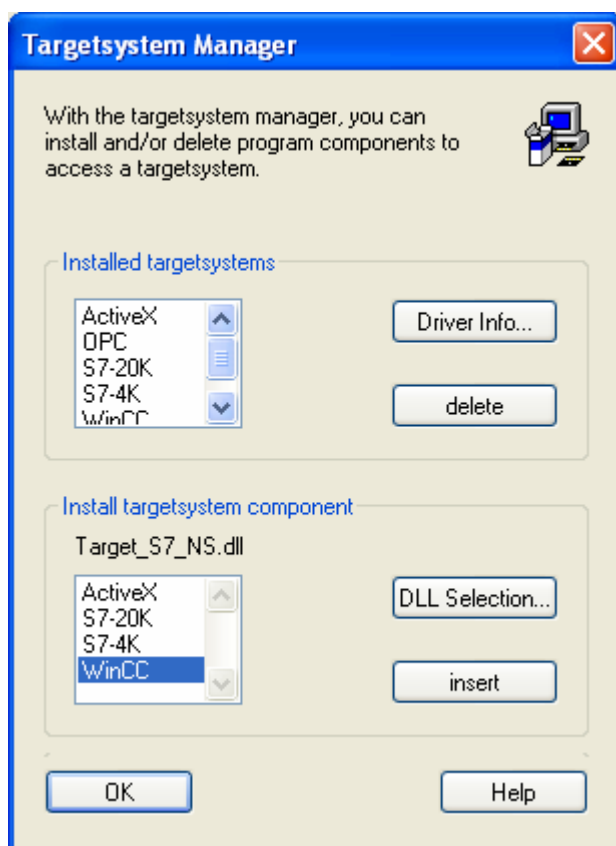




4.4.1 Manager

Shortcut: Hotkeys <ALT+S, M>

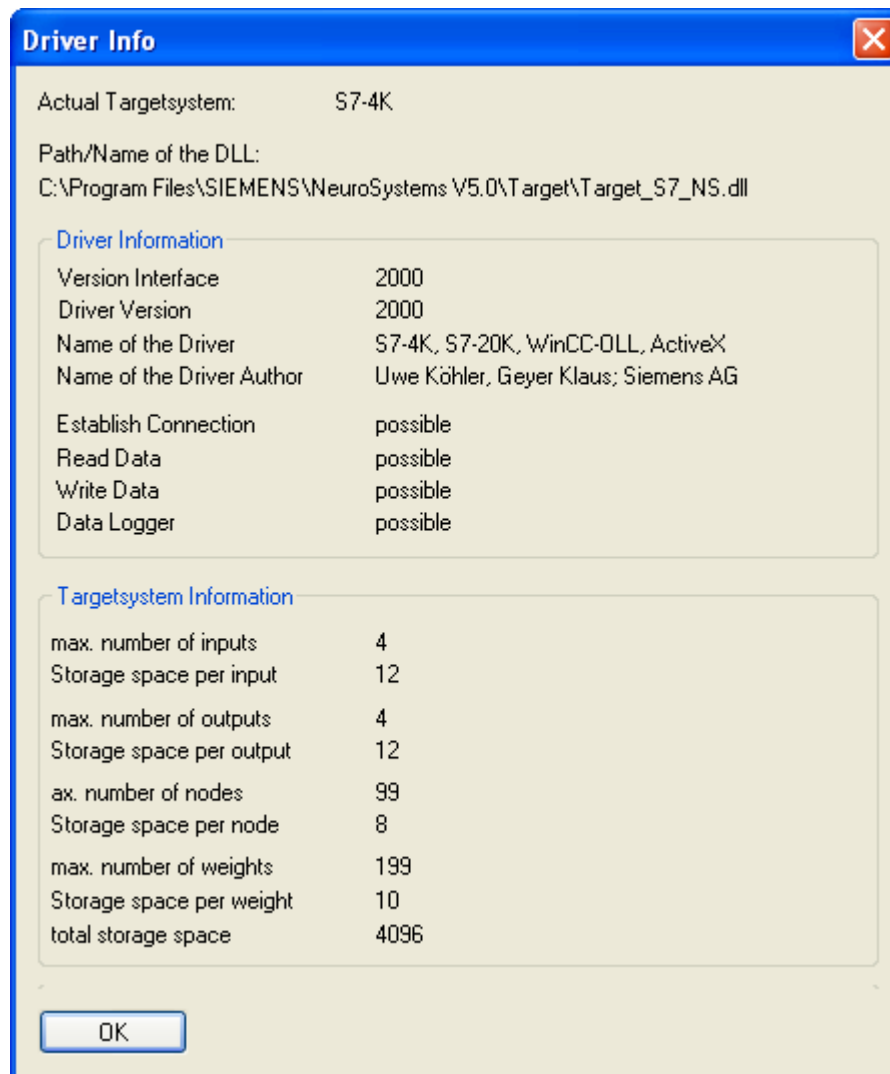
With help of the targetsystem manager you receive information on the installed targetsystems, special information on a selected targetsystem or you can install or delete a further targetsystem.





4.4.1.1 Installed targetsystems

In the dialog “targetsysteem manager“ all installed targetsystems are displayed in a list. By selecting one of the targetsystems and by pressing the button “driver info” you get to the “Driver Info” dialog that contains information on the selected targetsystem component.



Further you can receive the dialog “Driver Info” while creating a new project, by clicking the button “driver info” in the “project new” dialog.

In the “Driver Info” dialog you can read up general driver data and information on the quantity structure, e.g. the number of in- and outputs, that the targetsystem can support at most. These limits are controlled by *NEUROSYSTEMS* and *FUZZYCONTROL++* during the processing of the project. If a violation occurs the user receives an error message with the note that the quantity structure has been violated and will not be supported by the targetsystem.



4.4.1.2 Deleting targetsystem components

To delete a no longer required targetsystem select the targetsystem from the list in the “targetsystem manger” and press the “delete” button. The targetsystem will be removed from the list as well as out of the registry.

4.4.1.3 Installing targetsystem components

To install a targetsystem you must select a targetsystem-DLL with the “Select DLL” button. A dialog will appear in which you must select the analogical targetsystem-DLL. After you have selected a DLL, targetsystems will be offered in the list below which can be adopted into the upper list and be made available for further dialogs (“targetsystem selection” or “new file”), by clicking the “add” button.

The path/name of the DLL may appear more than once (when for example both targetsystems S7-4K and S7-20K are present in one DLL).

The standard is that targetsystems **WinCC**, **S7-4K**, **S7-20K** and **ActiveX** are supported. These four targetsystems are included in the driver-DLL **Target_S7_NS.dll**.

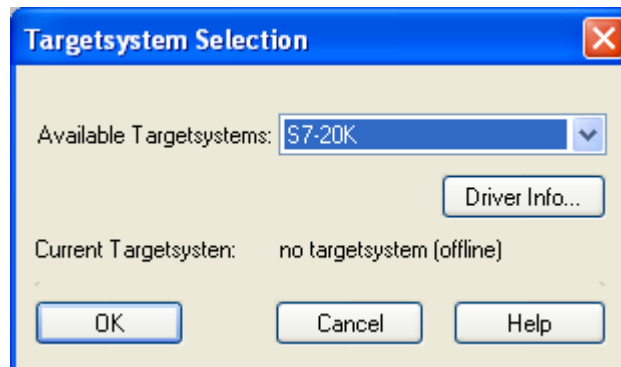
Targetsysteem OPC is included in the driver-DLL **Target_OPC_NS.dll**.

Attention: Since registry entries are made once you install, delete or select a targetsystem, the user must have administration rights to be able to do targetsystem settings. A normal user will get an error message.

4.4.2 Selection

Shortcut:	Hotkeys	<ALT+S, S>
------------------	---------	------------

With this command you can select the targetsystem on which you intend to run your neuro application. You can choose between the systems ActiveX, S7-4K, S7-20K, OPC and WinCC.



The maximum number of inputs and outputs, neurons and weights which can be defined is dependent on the chosen targetsystem:

Targetsystem	inputs	outputs	neurons	weights
S7-4K:	4	4	99	199
S7-20K:	100	10	160	1660
WinCC:	10	5	no limit	no limit
ActiveX:	100	10	no limit	no limit
OPC:	100	10	no limit	no limit
CFC-4K:	4	4	?	?
CFC-20K:	100	10	?	?

Caution: When using a NeuroFuzzy-Network (NFN) there is a maximum of 8 inputs and 4 outputs for all targetsystems!

Both S7-4K and S7-20K are run capable in SIMATIC S7-300 and SIMATIC S7-400.

Attention: Please select the targetsystem when you create your project with File/New and do not change it later on as possible, because some operations in *NEUROSYSTEMS* work dependant on the targetsystem setting. If you alter the setting the performance and the exactness of the neural network eventually can not reach the maximum possible level.

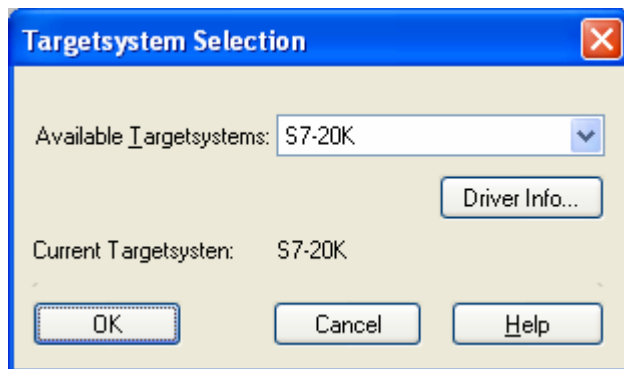


4.4.3 Targetsystem S7

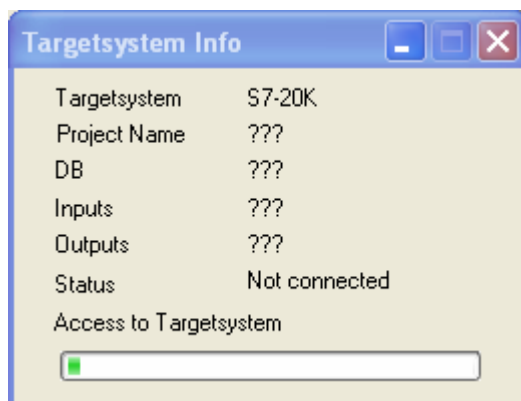
4.4.3.1 Selection

Shortcut: Hotkeys <ALT+S, S>

In this dialog you can choose between different targetsystems. Pick S7-4K or S720K from the list and confirm with *OK*.



The window targetsystem info is used as an information window and opens once the targetsystem S7-4K or S7-20K have been selected. It can be closed at any time and has no bearing on the going concern. Every file access from the projection tool *NEUROSYSTEMS* to the targetsystem instance is displayed by the progress bar. The number of accesses rises with connected targetsystems, once the curve plotter is active.

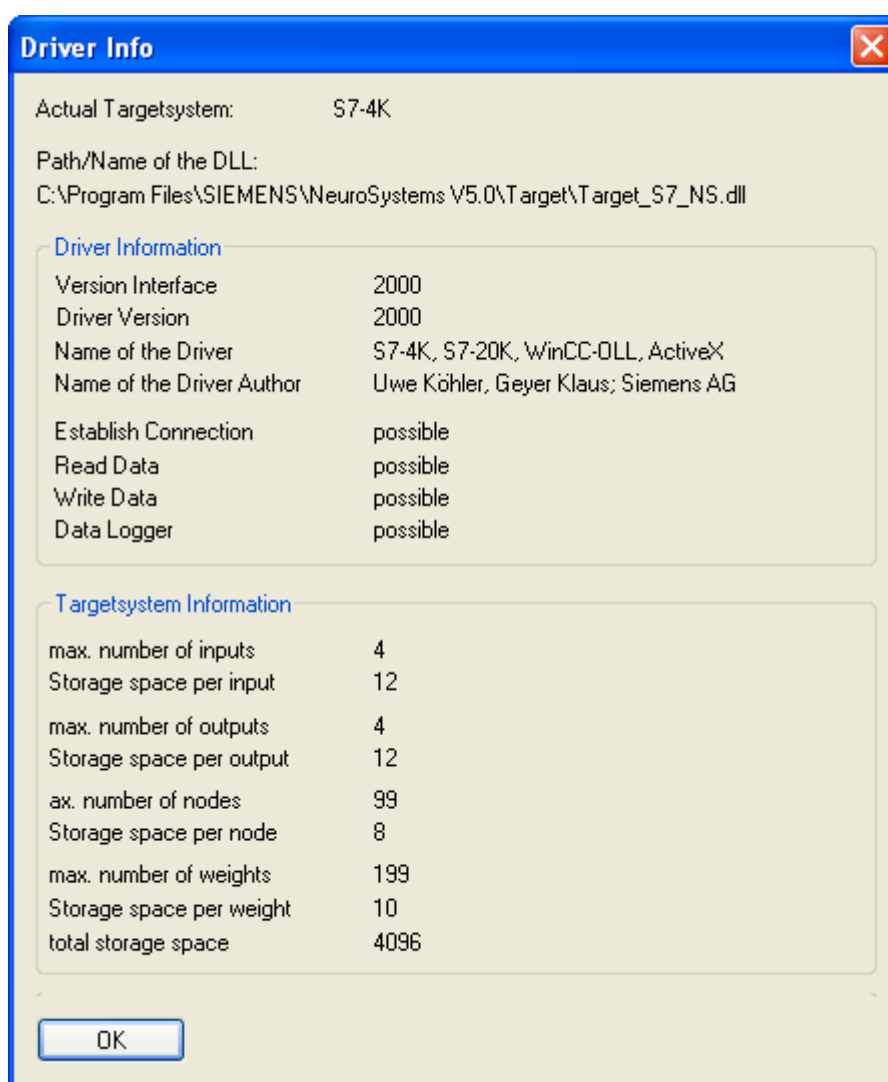




After a connection the parameters will be set.

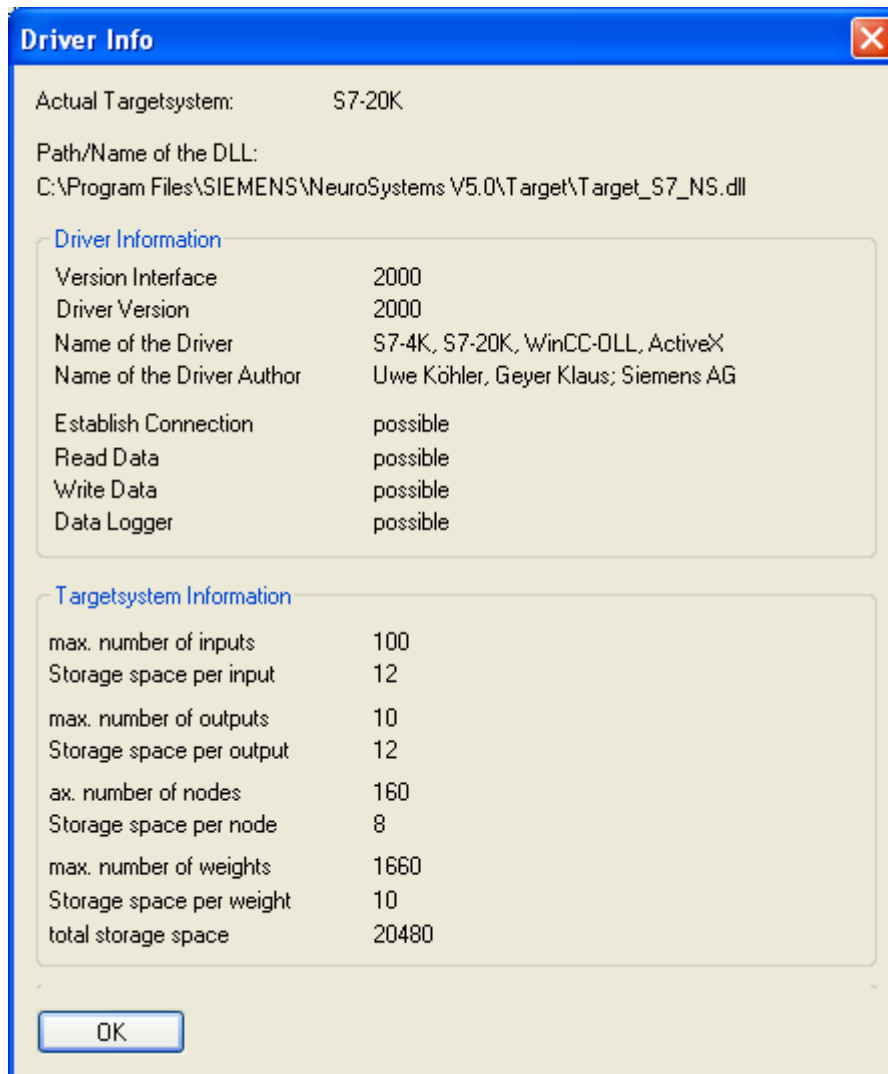
The targetsystem specific settings of S7-4K are shown by the dialog *diver info*, which you can receive by pressing the button *driver info*.

The targetsystem specific settings for S7-4K are the following.



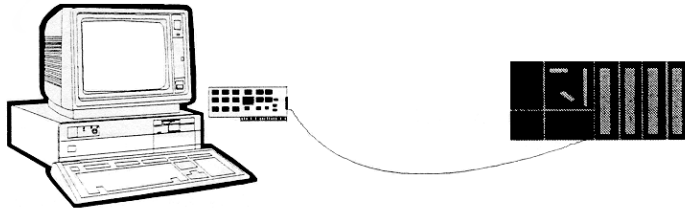


The targetsystem specific settings for S7-20K are the following.



Here you can use the S7-4K and S7-20K function blocks. For the interconnection to the SIMATIC S7 you need the configuring tool SIMATIC NET SOFTNET. In addition you need the S732.dll file which is installed together with the SIMATIC NET SOFTNET tool.

All cards, except CP5511 will be supported.



4.4.3.2 Targetsyst^{em}: Connect

Shortcut: Hotkeys <ALT+S, C>

This command sets up a connection to the data block of the selected targetsyst^{em} (S7-4K, S7-20K). This enables you to read or write a data block (DB), belonging to a neuro function block (FB), to or from an automation device. For the SIMATIC S7-300/400 there is a 4 KByte and a 20 KByte data block available. For both DBs there are matching FBs. It is possible to realize multiple neural networks with DBs for each network and one shared FB for all the networks.

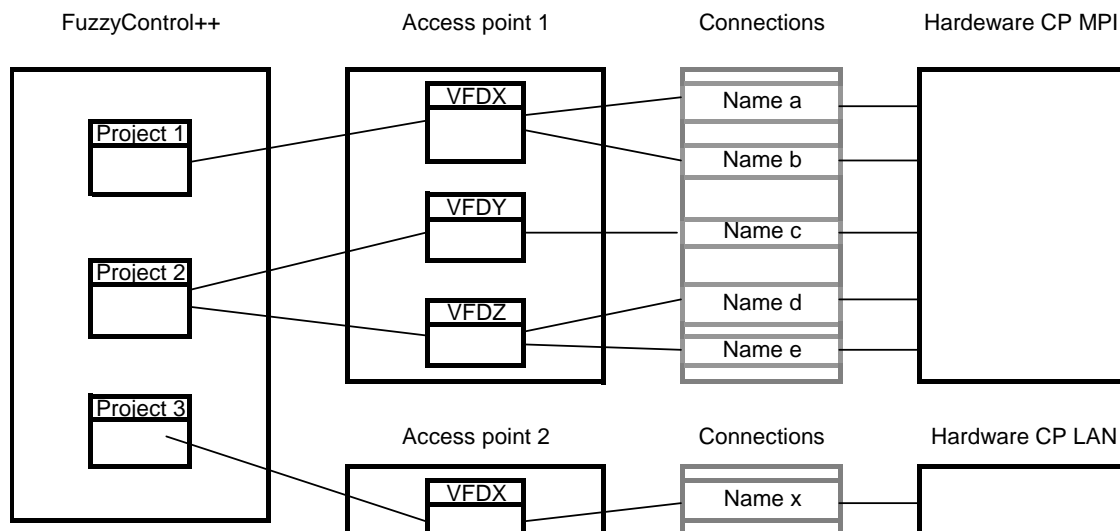
Note: The small 4 KByte DB and the matching FB can be applied to the SIMATIC S7-400, too.

To establish a connection between the configuring tool and the SIMATIC S7, proceed as the following:

1. There is a neuro function block (FB100 or FB101) as well as at least one neuro data block located (e.g. DB100) on the targetsyst^{em}.
2. The neuro function block has called and initialised the instance DB at least once.
3. The configuring tool *NEUROSYSTEMS* is open and a project will be processed.
4. The menu "Targetsyst^{em}/Connection" will be activated.
5. In the dialog *Connection* you will be asked to specify your connection closer.
 - At first you have to enter the path to the SIMATIC communication software "S732.DLL" on you computer. The default setting is your system directory. If this file is located in a different directory, you can select it by clicking on "Browse..." or directly entering the complete path name.
 - Choose the right access point. The access points will be assigned to the CPs in "PG-PC-interface configuring (SIMATIC NET)". Over an access point the analogical CP will be addressed.



- The VFD (Virtual Field Device) in the field *VFD-name* will be generated automatically. The S7 connections are assigned to this VFD-name. If you want to connect more than one project to the targetsystem at the same time, all of your opened projects need different VFD-names.



Assingment project-Access point-VFD-connections-CP

- For the choice of your destination adress you must make the following differentiation:

MPI

The connection to the SIMATIC S7 must be established with a MPI- card (Multi Point Interface). For communication you require the SIMATIC NET - Software SOFTNET S7 PB. Enter the MPI- adress of the CPU into the field *destination adress*. This coherent number (between 0 and 126) may only have three digits at maximum.

PB (PROFIBUS)

The connection to the SIMATIC S7 must be established with a PROFIBUS-card For communication you require SIMATIC NET - Software SOFTNET S7 PB. Enter the MPI/Profibus- adress of the CPU into the field *destination adress* This coherent number (between 0 and 126) may only have three digits at maximum.



IE (Industrial Ethernet)

The connection to the SIMATIC S7 must be established with a Ethernet-card For communication you require SIMATIC NET - Software SOFTNET S7 IE. Enter the Industrial Ethernet- address of the CPU into the field *destination address* These six double-digit hex-numbers between 0x0 and 0xFF are separated by dots (e.g.: 00.FF.0A.F0.1.EE).)

TCP/IP

The connection to the SIMATIC S7 must be established with a Ethernet-card For communication you require SIMATIC NET - Software SOFTNET S7 IE. Enter the TCP/IP-address of the CPU into the field *destination address* These four triple-digit decimal-numbers between 0 and 255 are separated by dots (e.g.: 141.8.10.237).

- Enter the S7- rack (between 0 and 7) into the edit box *rack*.
- Enter the number of the slot used in the rack (between 0 and 18) into the edit box *slot*.
- Enter the number of the neural data block for the S7-CPU (between 0 and 999999) into the box *number of data block*.
- In the lower text box you can enter in according text for this connection.

The image shows a Windows-style dialog box titled "Connect". The title bar is blue with standard minimize, maximize, and close buttons. The main area has a light beige background. At the top, it says "Communication Software SIMATICNET (S732.DLL)". Below this is a text field containing "C:\WINNT\SYSTEM32\S732.DLL" and a "Browse..." button. The dialog is divided into two main sections: "Connection Information" on the left and "Connection Parameter" on the right. In "Connection Information", there is a "Targetsystem name:" field with "S7-4K", an "Access point:" dropdown menu showing "S7ONLINE", a "VFD-Name:" text field with "VFD 10/20/06 13:38:36", and an "edit" checkbox. In "Connection Parameter", there are four radio buttons: "MPI" (selected), "PB", "TCP/IP", and "IE". Below these are four input fields: "Target address:" with a spinner set to 3, "Rack:" with a text box containing 0, "Slot:" with a text box containing 3, and "Number of the DB:" with a text box containing 101. At the bottom of the dialog is a large empty text area. The bottom of the dialog has three buttons: "OK", "Cancel", and "Help".



Note: For existing connections, you can not only read data from neuro data blocks, but also archive and display process data online in the curve plotter of the configuration tool.

For more information please refer to the corresponding SIMATIC documentation SIMATIC NET.

If the connection is established successfully, you can assign the neural system to the data block in the CPU.



4.4.3.3 *Targetsysteem: Disconnect*

Shortcut: Hotkeys <ALT+S, D>

This command breaks off an existing connection to the targetsystem (S7-4K, S7-20K).

4.4.3.4 *Targetsysteem: Read*

Shortcut: Hotkeys <ALT+S, R>

This command reads a data block (DB) from the targetsystem (S7-4K, S7-20K). **When reading the data, the active network in NEUROSYSTEMS will be overwritten with the data of the DB.** Because of this, the reading of the DB has to be confirmed in a dialog box. If you do not want to overwrite your current network you have to create a new network and overwrite it with the DB network data. The number of inputs and outputs, the network type etc. of the new network does not need to be the same as those of the read network!

4.4.3.5 *Targetsysteem: Write*

Shortcut: Hotkeys <ALT+S, W>

With this command you can write a data block (DB) to the targetsystem (S7-4K, S7-20K). It is possible to transmit the DB during the operation of the S7. However, please note that the execution of the neuro module has to be stopped during transmission!



4.4.4 Targetsysten WinCC

Note: This targetsystem will be supported up to the version WinCC V6.0 SPO. From the version WinCC V6.0 SP1 on, there are no *NEUROSYSTEMS* OLL- objects any more. You should use the ActiveX component.

The neural runtime module, which is realised by the „Neuro.oll“ file, represents an expansion of the Siemens software SIMATIC-WinCC. It makes the use of modern methods and applications possible, in which the abilities of neural systems come to use. The PC-configuring tool *NEUROSYSTEMS* is used for the projection of neural systems.

To make the OLL- object available in the graphical WinCC-Editor *Graphics Designer* you must proceed as the following:

- Open the context menu, which belongs to the *Graphics Designer* by clicking on the *Graphics Designer* with the right mouse button. Next please select the *Graphic-OLL*.
- Now select „neuro.oll“ from the left window (available *Graphic-OLL*) and adopt it to the right window (selected *Graphic-OLL*) with help of the arrow push button.

The *Graphics Designer* object palette now includes the new block in the group *Smart-objects*. The new object can be used in the same way that the already existing WinCC-Smart-objects have been used.

After creating a neural object on the drawing page of the *Graphics Designers*, you can determine its behaviour with the appending *object properties* window. The window is opened by clicking on the object with the right mouse button and subsequent selection of *properties*, or by double clicking with the left mouse button. The assignment of a certain neural in-/output behaviour to the built WinCC- object, happens by indicating a *NEUROSYSTEMS *.snl*-file. The analogical file has to be indicated for the attribute *snl-file* (in *properties*). The entry however must be carried out with the entire path name. After the file has been imported successfully, a display with the number of in- and outputs (in *properties*) of the projected neural system will appear. In addition the in- and outputs will be illustrated graphically by small lines on the object. In *properties/in-/outputs* you can see all in- and outputs and their current values. **Neural objects are limited to a maximum of 10 in- and 5 outputs.**

A simple function test of the block can be done by changing the input values (window *object properties* in *properties/in-/outputs*) manually and watching the resultant changes of the output values.



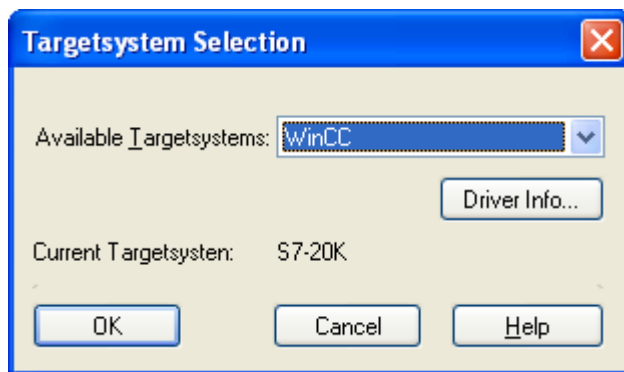
The single in- and outputs can be activated by a direct connection, tags or c- actions or linked with other objects (dynamic sampling). The new object is now fully integrated into the WinCC- surrounding.

For further information please use the accordant WinCC- manuals.

4.4.4.1 Selection

Shortcut: Hotkeys <ALT+S S>

In this dialog you can choose between different targetsystems. Pick WinCC from the list and confirm with *OK*.

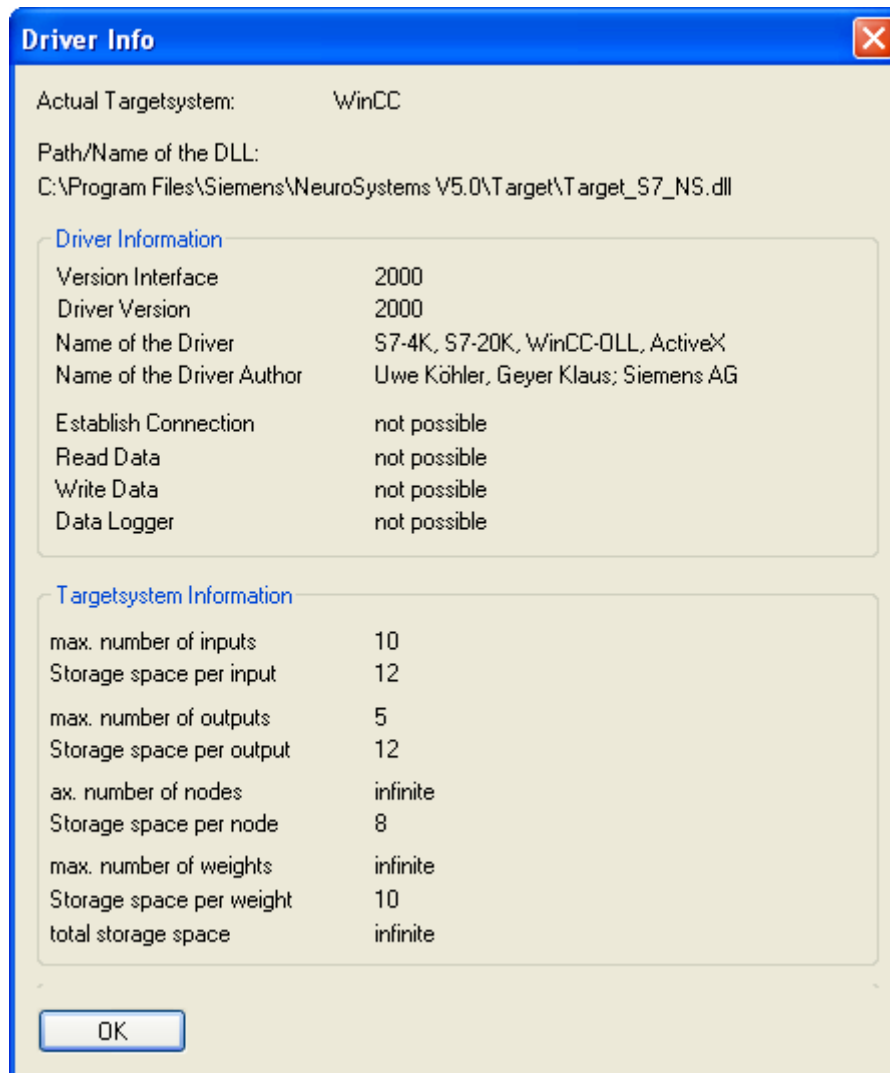


By selecting a targetsystem you can possibly exclude memory and capacity problems while loading the neural network to the targetsystem WinCC.

During projection of the neural network the maximum possible number of in- and outputs will be checked automatically and reported if violated.

The targetsystem specific settings of WinCC are shown by the dialog *driver info*, which you can receive by pressing the button *driver info*.

The targetsystem specific settings in WinCC are shown by the following dialog.



4.4.4.2 Connect

Connect targetsystem is not realised in WinCC

4.4.4.3 Disconnect

Disconnect targetsystem is not realised in WinCC

4.4.4.4 Read

Read targetsystem is not realised in WinCC

4.4.4.5 Write

Write targetsystem is not realised in WinCC.



4.4.5 Targetsysteem ActiveX

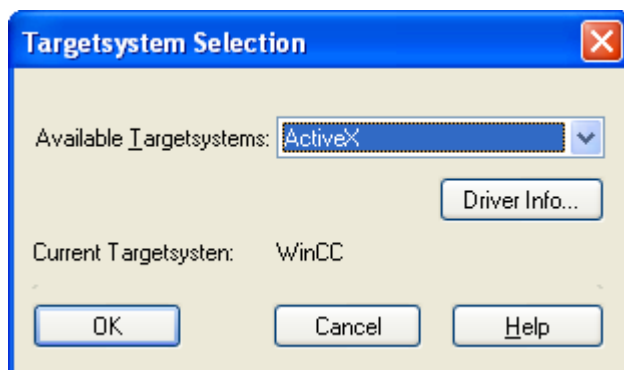
Note: Because there are no *NEUROSYSTEMS* OLL- objects from the version SIMATIC WinCC V6.0 SP1 on, that are required by the targetsysteem WinCC; you should use the ActiveX component for SIMATIC WinCC.

The realised neural runtime module ActiveX can also be used for the Siemens software SIMATIC-WinCC. In contrast to the OLL- version it has not been designed especially for SIMATIC WinCC. The ActiveX component can be integrated within any ActiveX container. There is no limitation to the number of in- and outputs. 100 in- and 10 outputs are possible.

4.4.5.1 Selection

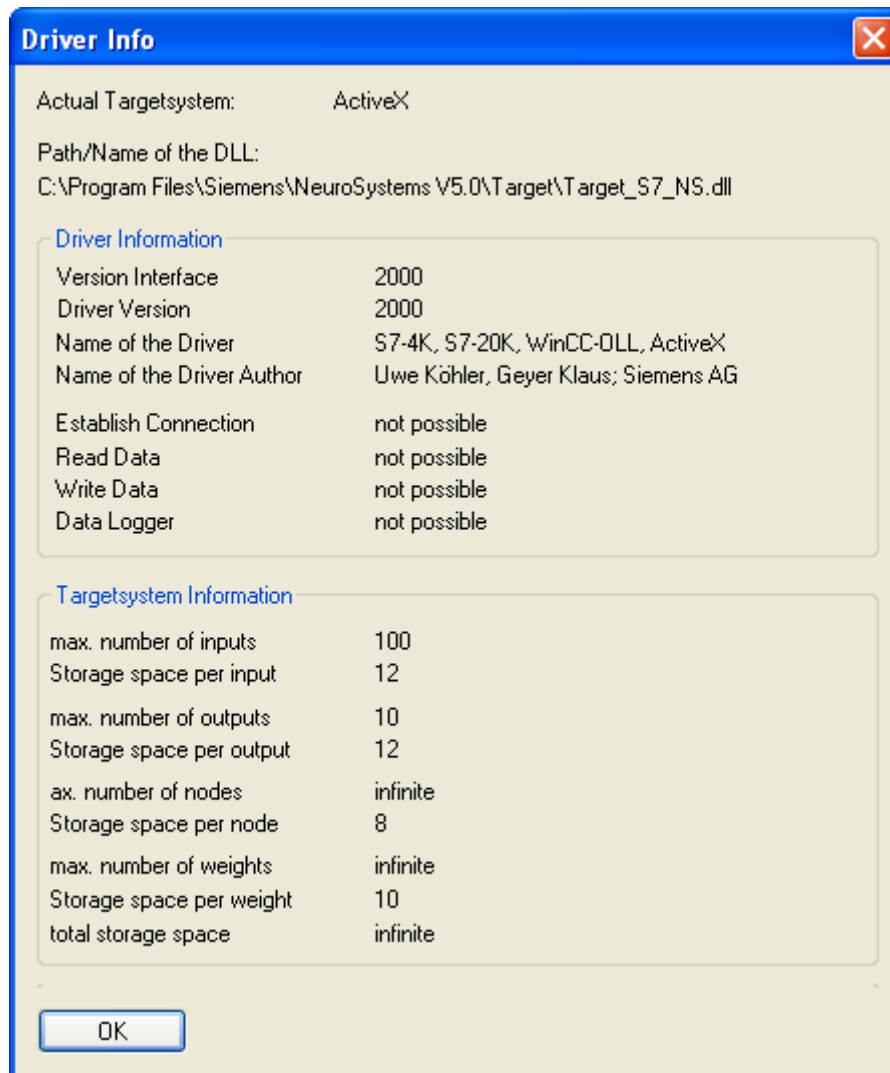
Shortcut: Hotkeys <ALT+S, S>

In this dialog you can choose between different targetsystems. Pick ActiveX from the list and confirm with *OK*.



By selecting a targetsysteem you can possibly exclude memory and capacity problems while loading the neural network to the targetsysteem ActiveX

The targetsysteem specific settings in ActiveX are shown by the following dialog.



4.4.5.2 Connect

Connect targetsystem is not realised in ActiveX.

4.4.5.3 Disconnect

Disconnect targetsystem is not realised in ActiveX

4.4.5.4 Read

Read targetsystem is not realised in ActiveX

4.4.5.5 Write

Write targetsystem is not realised in ActiveX

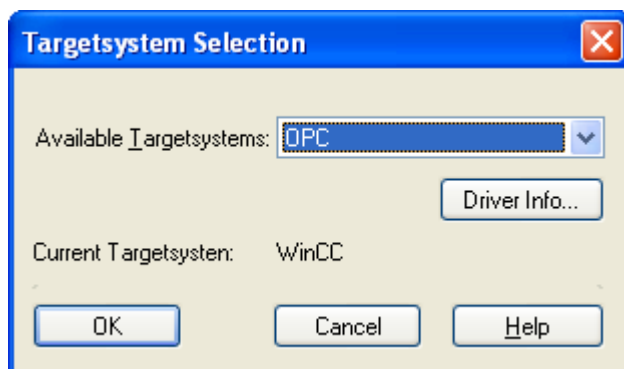


4.4.6 Targetsystem OPC

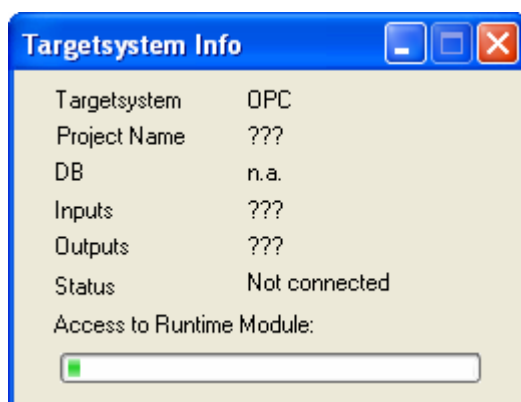
4.4.6.1 Selection

Shortcut: Hotkeys <ALT+S, S>

In this dialog you can choose between different targetsystems. Pick OPC from the list and confirm with *OK*.



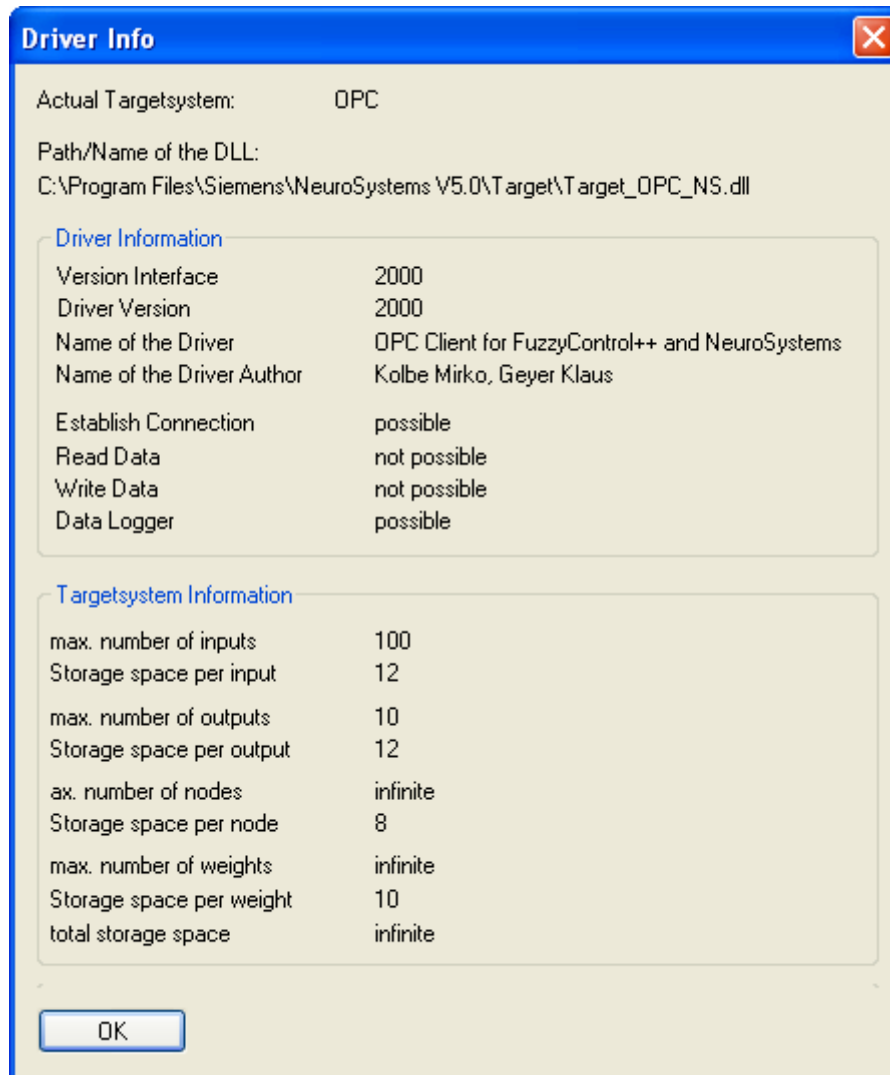
The window Targetsysteem Info is used as an information window and opens once the targetsystem OPC has been selected. It can be closed at any time and has no bearing on the going concern. Every file access from the projection tool *NEUROSYSTEMS* to the targetsystem instance is displayed by the progress bar. The number of accesses rises with connected targetsystems, once the curve plotter is active.



After successful connection the parameters will be set.

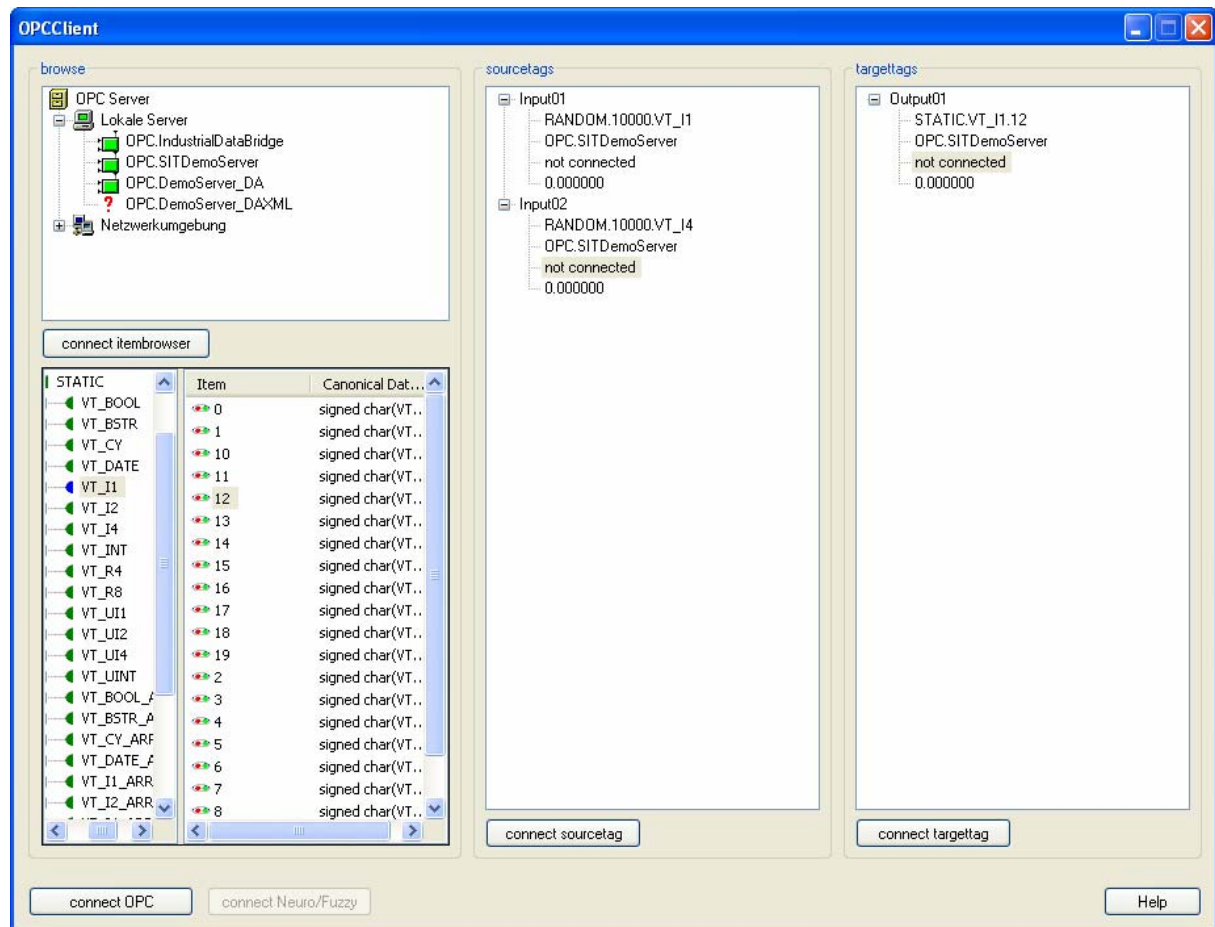


The targetsystem specific settings of OPC are shown by the dialog *DiverInfo*, which you can receive by pressing the button *DiverInfo*.



4.4.6.2 Connect

The targetsystem OPC represents a PC solution. The network is currently running on a PC and receives its input values online via OPC connections. The network calculated output values are written by OPC connections likewise. Each in- and output can receive its own OPC tag on an OPC server. Up to 110 different OPC servers are possible for the 100 in- and 10 outputs.



The names of the inputs of the neural network are listed in the **sourcetags** list. Further you can find the assigned tag name below the name, the according OPC server, the condition of the connection and the tags' value.

In the list **targettags** the names of the outputs are listed. Further you can see the assigned tag name below, the according OPC server, the condition of the connection and the value of the tag.

In the upper left list all the accessible OPC servers are listed. The red OPC servers are AE (alarm event) servers. They may not be selected. The green OPC servers are the DA (data access V2.0) servers, with whom you can arrange a connection. It is possible to browse between servers within the list. By selecting a DA OPC server and confirming with the button **“connect itembrowser”** the tags of the selected OPC server will be displayed in the lower left list. It is possible to browse between tags within the list.

The assignment of a tag to an input happens by selecting a tag, selecting an input and subsequent pressing the button **“connect sourcetag”**. The assignment of a tag to an output happens by selecting a tag, selecting an output and subsequent pressing the button **“connect**



targettag”.

During projection the tags should be assigned only, if their type fits to the real value of the network. If possible, the value adaptation will be done automatically. If the type does not fit, an error report will appear.

Permitted data types

When OPC servers are connected, all data types are supported, which can be changed and are in range.

The neural network needs for inputs and outputs FLOAT.

To avoid converting problems only tags from type FLOAT should be used.

OPC data type	data type	change to FLOAT
VT_BOOL	BOOL	yes
VT_I1	CHAR	yes
VT_UI1	BYTE	yes
VT_I2	SHORT	yes
VT_UI2	WORD	yes
VT_UI4	DWORD	yes
VT_I4	LONG	yes
VT_R4	FLOAT	yes
VT_R8	DOUBLE	yes
VT_DATE	DATE	no
VT_BSTR	STRING	no



The data types have the following value ranges:

OPC data type	Range of values
VT_BOOL	0 or 1
VT_I1 -	128 to 127
VT_UI1	0 to 255
VT_I2 -	32768 to 32767
VT_UI2	0 to 65535
VT_UI4 -	2147483648 to 2147483647
VT_I4	0 to 4294967295
VT_R4	3.402823466 e-38 to 3.402823466 e+38
VT_R8	1.7976931486231e-308 to 1.7976931486231e+308
VT_Date	1 Januar 100 to 31 Dezember 9999

Beneath the name of the in- or output the assigned tag name, the according OPC server and the condition of the connection will be displayed. The current value of the tag will be displayed, once the **button “connect OPC”** has been pressed. To not strain the system to hard the values of the tags will only be read if changed. This applies for source as well as target tags.

After pressing the button **“connect Neuro/Fuzzy”** the network will be fed with the upcoming input values and calculates the output values that have been written to the target tags. As soon as an input value has changed, a new calculation of the output values will occur.

The values of the target tags also will be read by the OPC servers if changed. After changing the values will be displayed in the list of the target tags. Consequently it can be guaranteed that the values have also been listed on the OPC servers.

The curve plotter can record and archive the values (see chapter curve plotter).

If a connection is disconnected during transmission, it will be recognized and reconnected (automatic reconnection). Consequently a continuous operation without a handling intervention is guaranteed.



By clicking the button **“Disconnect Neural/Fuzzy”** the network will be deactivated, the OPC connections however will stay active.

The button **“Disconnect OPC”** will disconnect the OPC connection.

Attention:

Closing the window will also stop transmission. Therefore the window should only be minimized, if it is in the way.

The entire projection information, so the assignment of the OPC tags to the in- and outputs will be read back and saved within the projection file in case of disconnect targetsystem. Thus the next time you connect the targetsystem all assignments will already be existant and you can activate the data transmission once more via the “connect OPC” and “connect Neuro/Fuzzy” buttons.

4.4.6.3 Disconnect

Shortcut: Hotkeys <ALT+S, D>

This command will disconnect an existing connection to the target system OPC.

Via “disconnect targetsystem” the OPC connections will be shut down, if it hasn’t been done via the **“disconnect OPC”** button. The entire projection information will be read back and can be saved within the project file via”save project”.

4.4.6.4 Read

It is not necessary to read the targetsystem in OPC.

4.4.6.5 Write

It is not necessary to write the targetsystem in OPC. It will happen automatically via “connect targetsystem” .



4.5 Menu: Learn

The *Learn* menu is used to prepare and execute the learning process.


Preparation involves selecting the learning file and setting the learning environment (validation data, error limit, learning time).

You can control learning with *Start*, *Stop* and *Continue*.

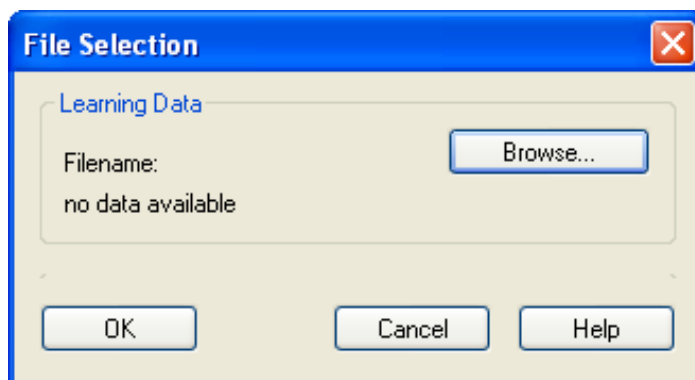
Often you can improve a learning result with *Fine Tuning*.

NOTE: Once you have created a network and have trained it with learning data you should not "retrain" it using a different learning data file. In this case it is recommended to create a new neuro project and to train the new network with this learning data!

4.5.1 Learn: File Selection

Shortcut: In the toolbar with the LMB on the icon 

Hotkeys <ALT+L, F>



In the *Select Learning File* dialog box you can click on *Browse* to load the learning data file for the problem to be solved. The Windows dialog box *Open* will appear. After that, in the *Select Learning File* dialog box the message *No data available* is replaced by the name of the learning data file loaded. You can terminate the process with *OK*.

If the network has already been trained with a learning data file, the corresponding *.dat file is already displayed in the *Learn File Selection* dialog box when it is called up. If the number



of inputs and outputs of the configured network does not match the number of columns in the learning data file, an error message appears and the learning file cannot be loaded.

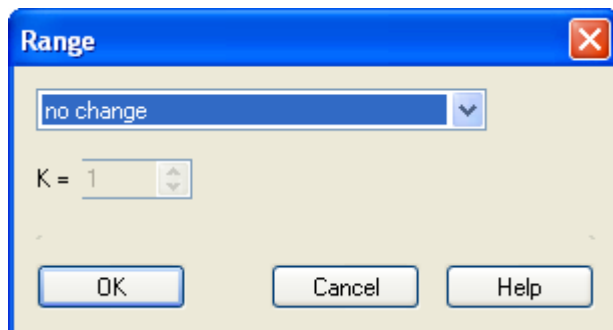
After choosing a learning file and confirming with OK, a window opens that gives you the possibility to access the dialog *range*, in which you can determine the range of the in- and outputs with the selected learning file automatically.

4.5.2 Range

Shortcut: Hotkeys <ALT+L, R>

You have the following possibilities to open the dialog *range* after selecting a learning file:

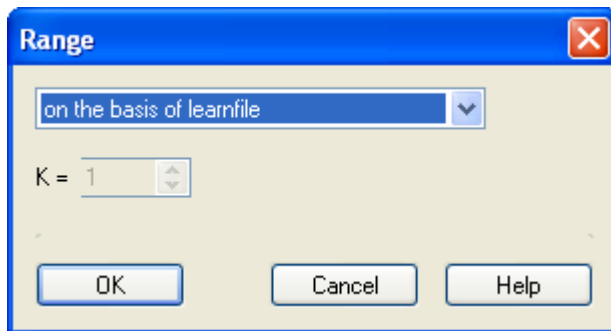
- After selecting a learning file the message appears whether the range of the in- and outputs should be determined by this learning file. If the message is confirmed with *yes*, the range dialog will open.
- Further the dialog *range* can be opened via calling up *learning, range*, after choosing a file.



Within the dialog you have the following possibilities to determine the range with the selected learning file:

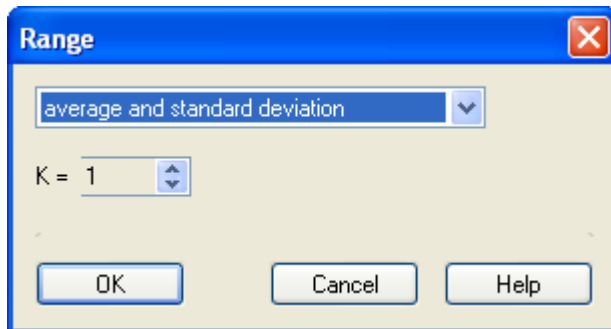
- No change

The ranges stay unaltered, which means that this setting is analogical to leaving the dialog via *abort*.



- On basis of learnfile

By selecting this, the lower/ upper limit of the single in- or output will be allocated with the respectively smallest/ biggest value from the learning file.



- By means of the average and standard deviation

After selecting this option, the average and standard deviation will separately be determined for each in- and output by means of the learning file. The lower and upper limit of the single in- and outputs can be set as the following shows.

Lower limit: $\text{average} - k * \text{standard deviation}$

Upper limit: $\text{average} + k * \text{standard deviation}$

k = 1, 2, 3, 4

The smaller the chosen value for K is, the higher the number of values of the in- or outputs will be that are treated as outliers and the smaller the ranges will be.



Note: The learning file itself will not be altered.

The chosen procedure to determine the ranges concerns all in- and outputs. The procedure will be applied to each in- and output separately. Determining for the lower and upper limit of the in- or output are exclusively those values of the learning file that are assigned to that in- or output. The determined ranges can be seen in *in-/output properties* and can be changed manually if required.

Note: If the range of the in- and output is determined, you can generally reach better learning results. If you are using a NFN- network a determination has to occur, so the creation of the rule basis provides you with useful results.

Note: Once a determination of the range has been done you should not change it anymore. If it should be necessary (e.g. because of new data), please arrange a new learning process to avoid erroneous results

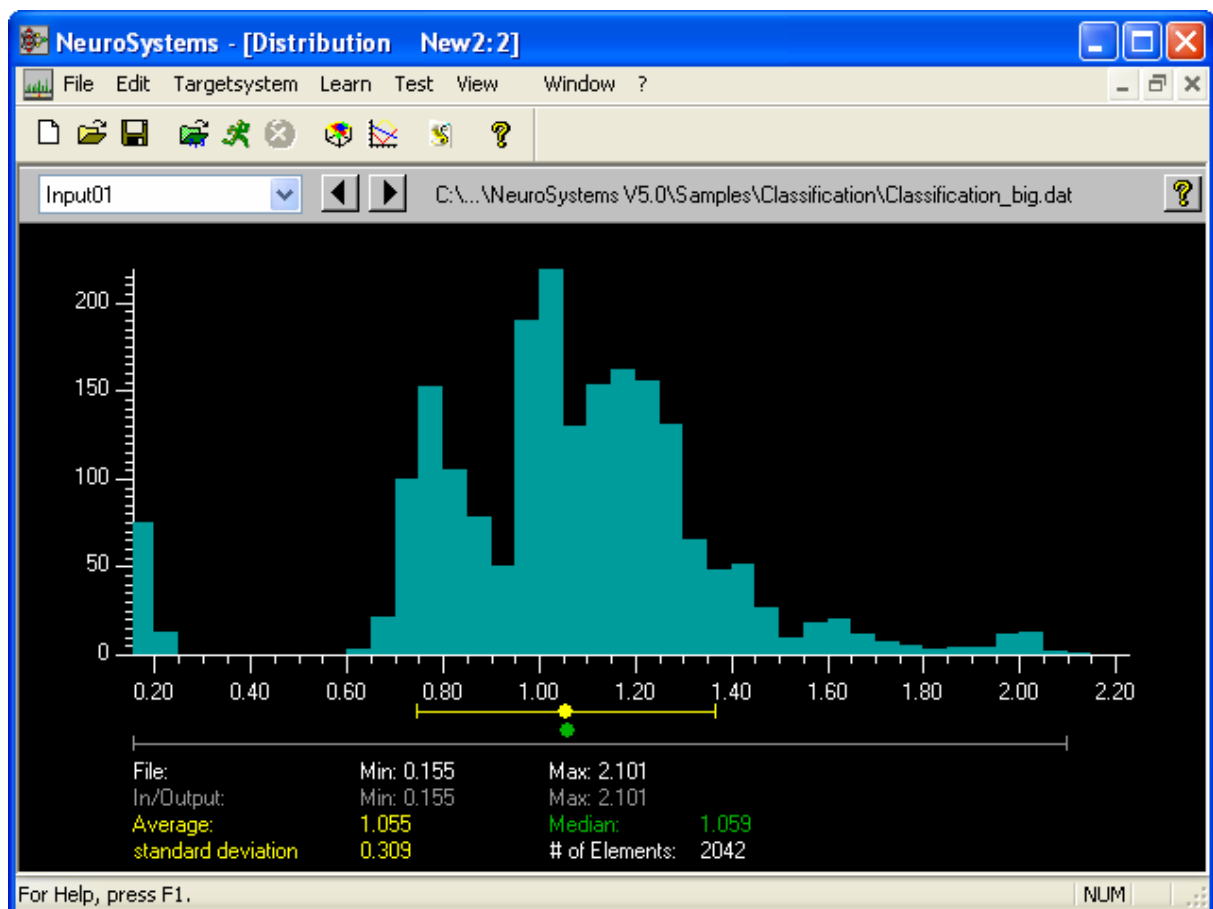


4.5.3 Distribution

Shortcut: Hotkeys <ALT+L, D>

In the window distribution the value distribution of an in- or output is displayed graphically.

The window can only be opened, if a learning file has been selected.



The name of the learning file, which has been selected from the *submenu file selection* in the menu *learn*, will be shown in the upper part of the window.

The according in- or output can be selected with the combo box or with the backwards-forwards button.

The range of the selected in- or output is plotted onto the **x-axis**.



In case that the standardisation of the range has been carried out by the menu *learn/range* and the range has not been change manually via *edit/input/output/properties*, then the denoted values are identical under file and under In-/Output. If selecting a file with who has not been learned, the ranges are mostly different. The x-axis includes both ranges.

The x-axis is divided in narrow ranges that adapt to the axis scaling autonomously, according to the size of the window and the selected zoom range. The ranges are geared to the scaling of the x-axis. A range includes the distance between 2 small lines of the x-axis. The height of the bar normally minimises by zooming within the diagram. Through this the single ranges will be downsized and consequently fewer values will lie in a range.

In each range, where values are existent, a bar with according height is applied.

The **y-axis** represents the number of values in the particular bar.

File

Min. and max. indicate the smallest and biggest value within a learning file.

In-/Output

Min. and max. indicate the, in the menu *in-/output/properties*, set values. The gray crossways lying bar underneath the x-axis shows the set range of the in- or ouput.

Average

The average is listed as a value, as well as a yellow dot drawn in underneath the x-axis.

Median

The median is that value that is seated in an odd- numbered numerical series, so that the same amount of values appears on both sides. If the numerical series is even it is the median between the both middle values. It is drawn in underneath the x-axis as a green dot.

Standard deviation

Here you can find the value as well as the range between the median minus the standard deviation and the median plus the standard deviation displayed as a yellow crossways laying bar underneath the x-axis.



#Elements

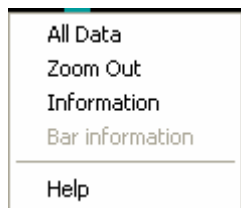
The number of values in the learning file.

Zoom In:

By clicking on the display with the left mouse button, holding the button and moving the cursor and then letting go of the button you can select a cut-out and display it. Repeated zooming in is possible and allows precise examinations within a certain range of the x-axis.

Context sensitive menu:

You get to the context sensitive menu for the diagram distribution by clicking and letting go of the display field with the right mouse button. The following select box opens.



All Data:

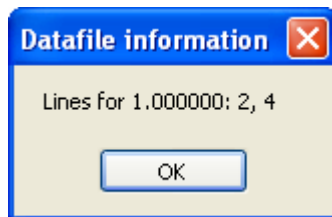
By choosing this option you have the possibility to get back to the general view after zooming.

Zoom Out:

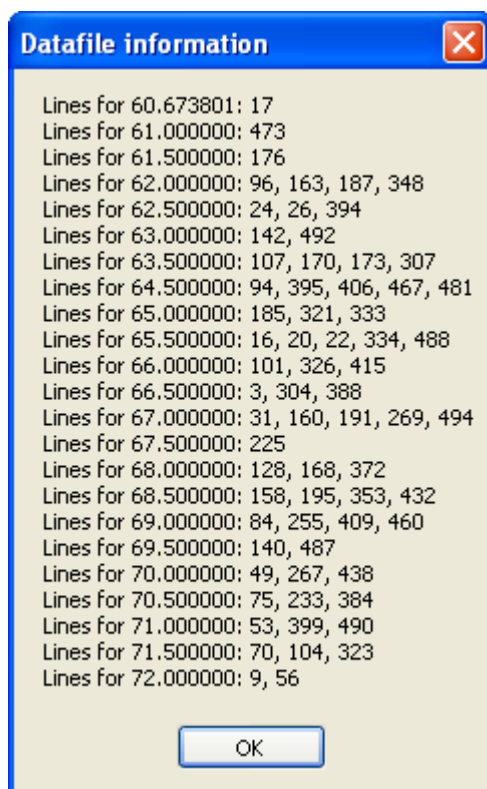
By choosing an entry you have the possibility to go back one step and to look at the next to last selection.

Bar information:

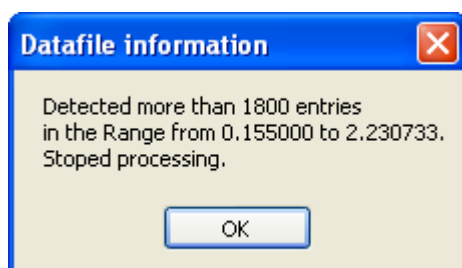
By choosing this option you receive information on a selected bar. The lines of the lernan file are listed that add up to this bar. Thus it is easy to detect and if necessary to delete outliers in the file.

**Information:**

By choosing this option you receive information on all visible bars. The lines of the learn file are listed that add up to the single bars. An accordingly small cutout, created by a zoom in, is requirement.



Otherwise you receive an error box.

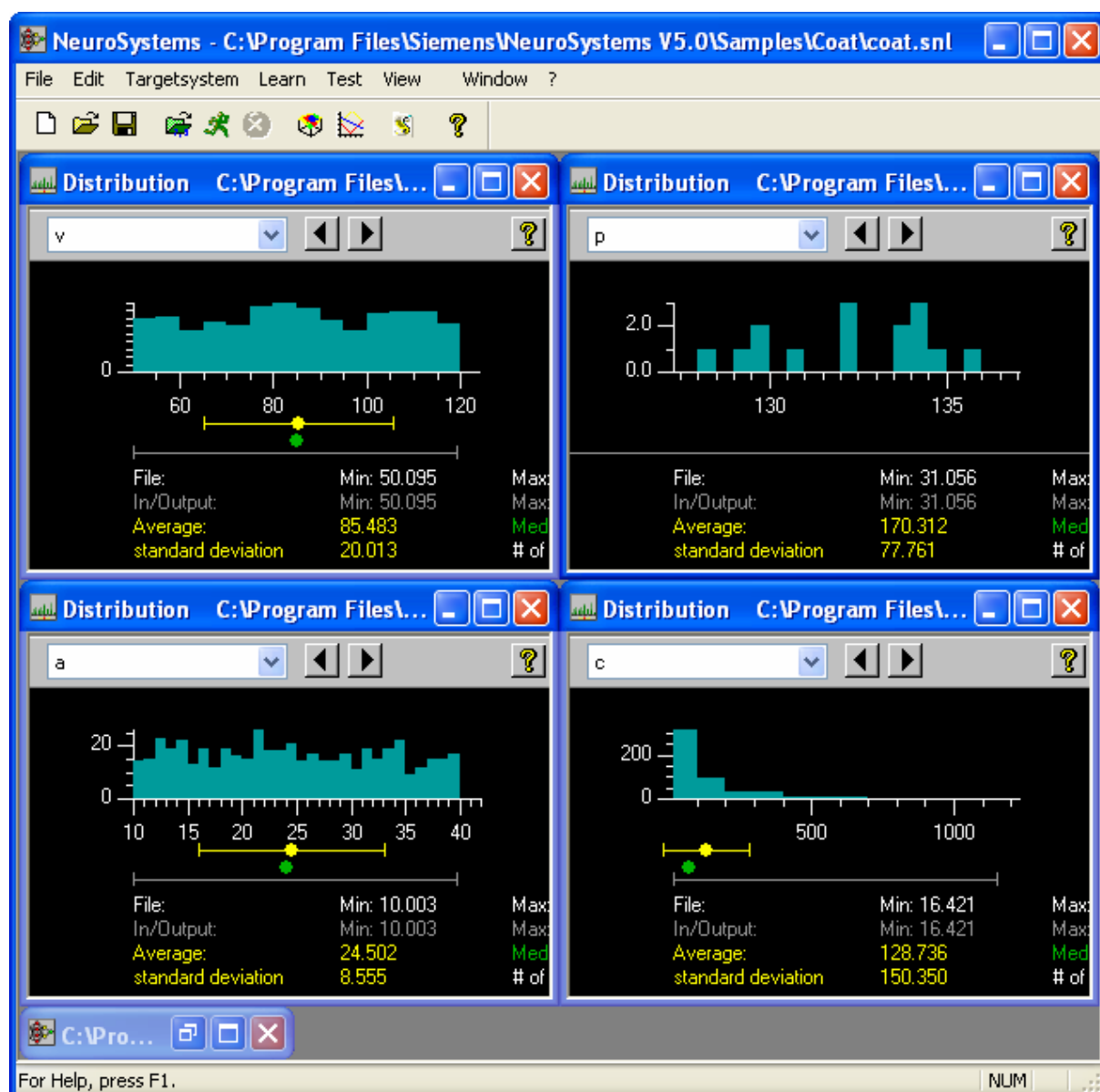




Help:

By choosing this option you get to the help for this window.

The window distribution can be opened more than once and therefore offers an overview of the distribution of all in- and output signals.



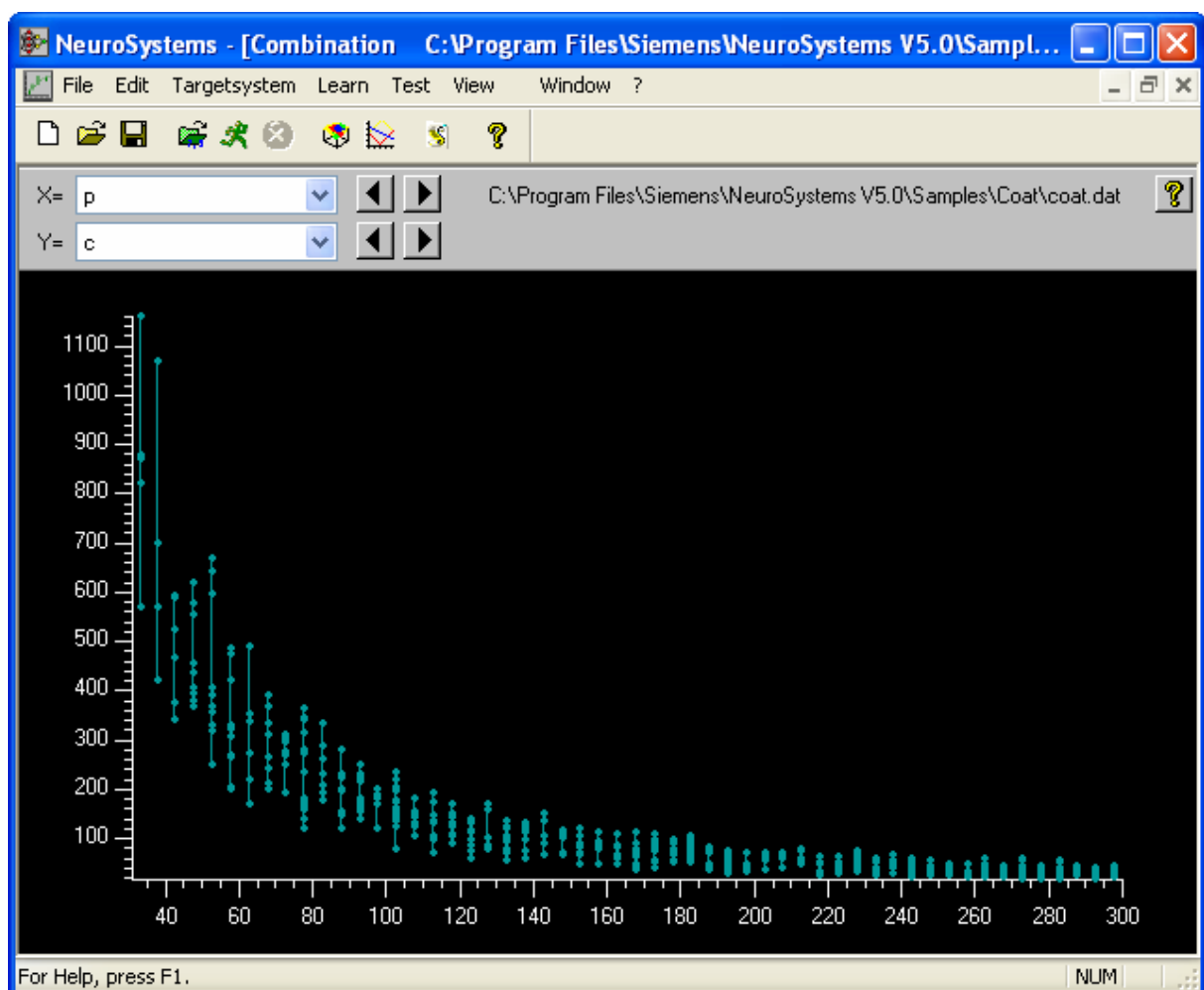


4.5.4 Combination

Shortcut: Hotkeys <ALT+L, O>

The window can only be opened. If a learning file has been selected.

In the window combination the values of an output signal are assigned to the values of an input signal graphically. Here you can see, which output values appear in which input values. The combination of all in- and outputs among themselves is possible. For this you can use either the 2 **combo boxes** or the backwards- forwards buttons.



The name of the learning file, which has been selected from the *submenu file selection* in the menu *learn*, will be shown in the upper part of the window.

If there are more output values for a single input value, than they will be visualised by an vertical connection. More precisely it doesn't concern the exact values, but the ranges, that's



quantity is dependant on the zoom factor. A range is geared to the scaling of the x-axis. A range includes the distance between 2 lines of the x-axis. Through zooming the vertical connections may disappear, because thereby the ranges are downsized and consequently fewer points will lie in the range.

In contrast to the window consistency it is possible that several output values are assigned to an input value (not input vector). They can digress from each other strongly.

By means of the displayed graphic it is possible to conclude that there is a dependency (none, linear, non linear) between an in- and an output.

The according input can be selected from the **upper combo box** or with the backwards-forwards button. It is corresponding to the x-axis.

The according output can be selected from the **lower combo box** or with the backwards-forwards button. It is corresponding to the y-axis.

The range of the input is applied to the **x-axis**.

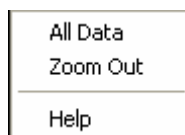
The range of the output is applied to the **y-axis**.

Zoom In:

By clicking on the display with the left mouse button, holding the button and moving the cursor and than letting go of the button you can select a cut-out and display it. Repeated zooming in is possible and allows precise examinations within a certain range of the x-axis.

Context sensitive menu:

You get to the context sensitive menu for the diagram distribution by clicking and letting go of the display field with the right mouse button. The following select box opens.



All Data:

By chosing this option you have the possiblity to get back to the general view after zooming.

Zoom Out:

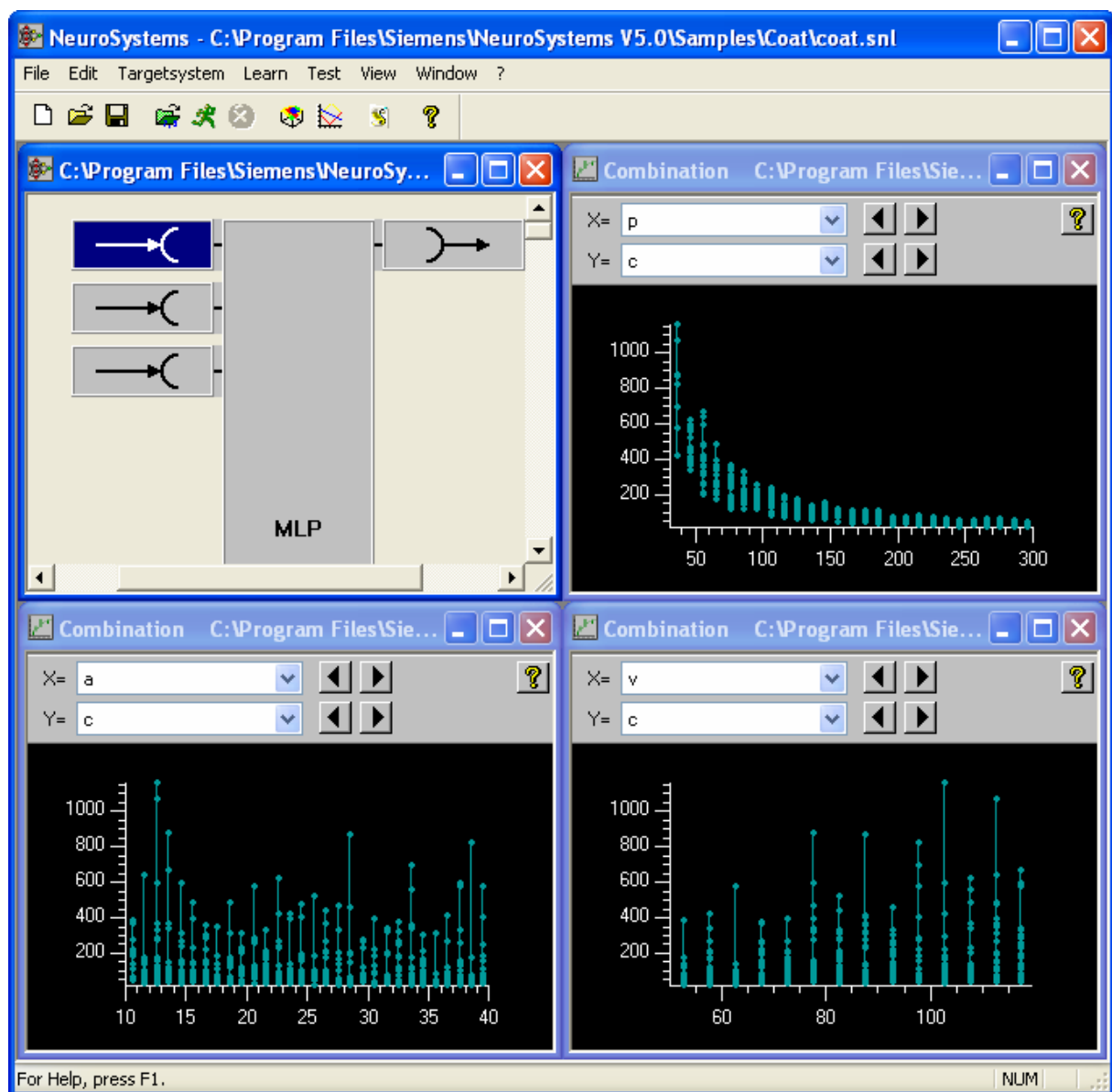


By choosing an entry you have the possibility to go back one step and to look at the next to last selection.

Help:

By choosing this option you get to the help for this window

The window combination can be opened more than once and therefore offers an overview of the relations of all input signals to one output signal.





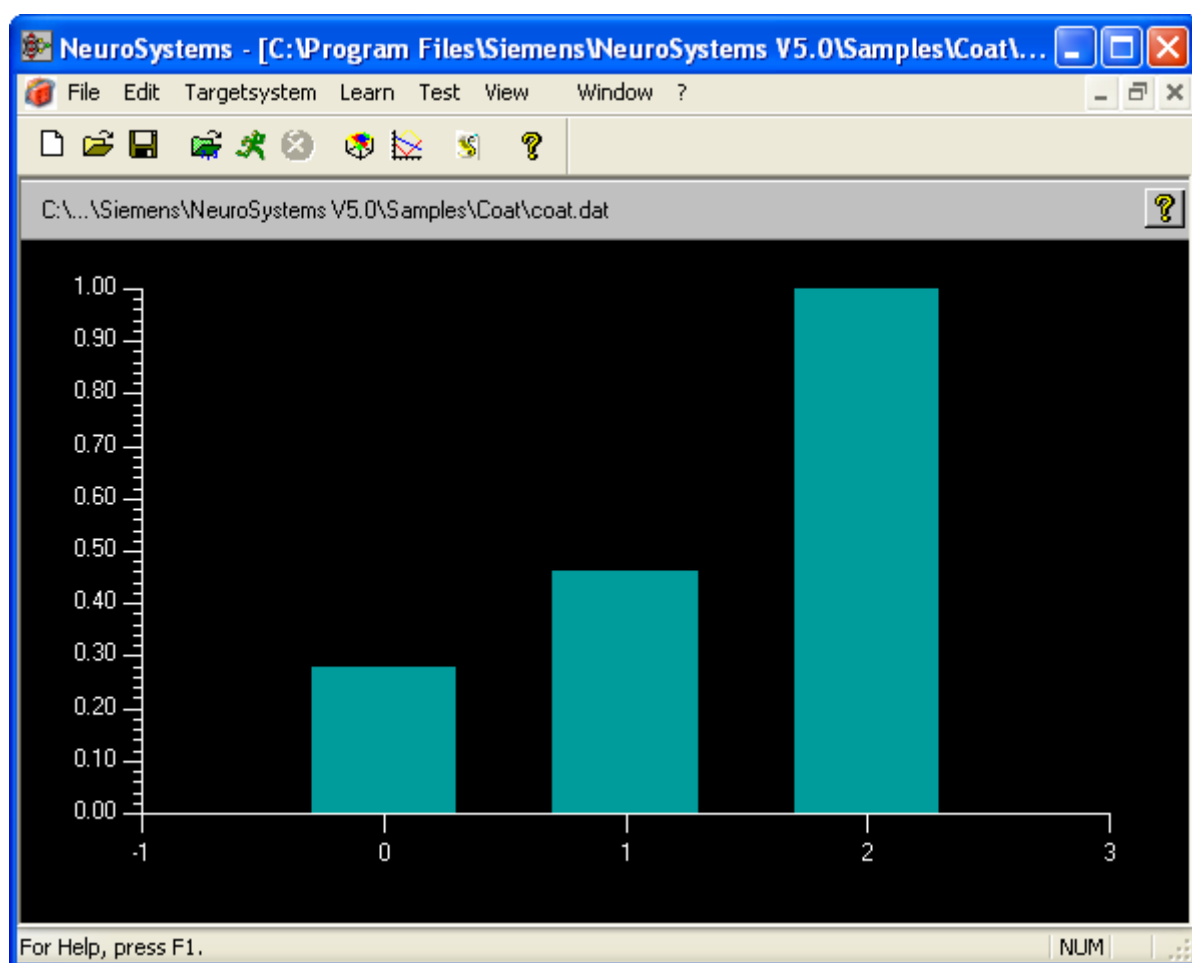
4.5.5 Input Relevancy

Shortcut: Hotkeys <ALT+L, I>

The window can not be opened until a learning file has been selected.

In the window input relevance the meaning of the single inputs is displayed graphically. The higher the bar is, the more important the assigned is.

Less important inputs can be neglected.

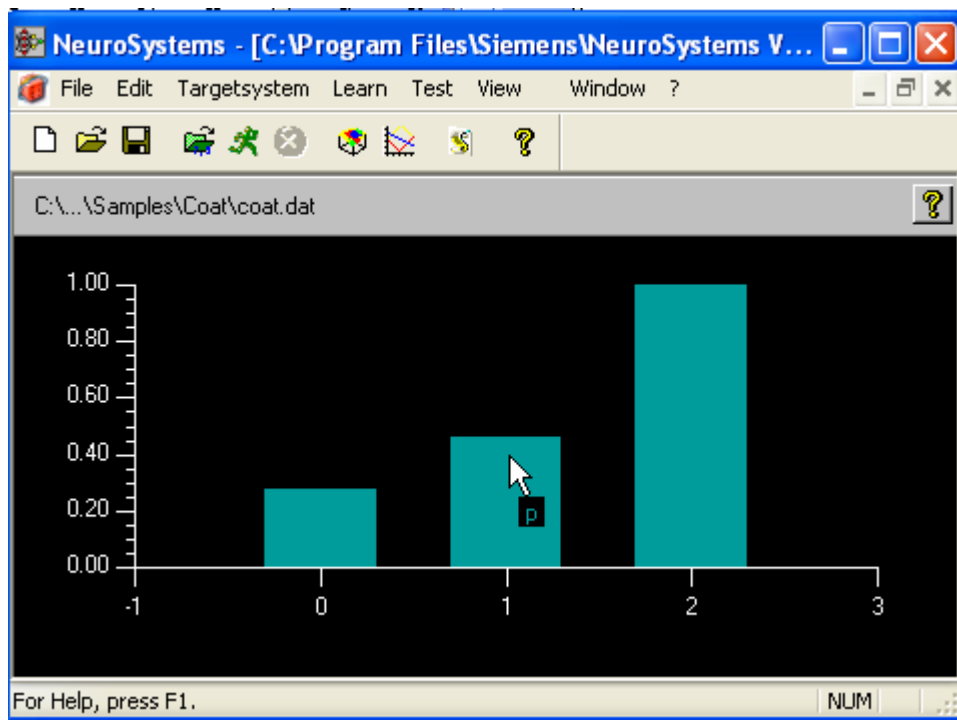


The name of the learning file, which has been selected from the *submenu file selection* in the menu *learn*, will be shown in the upper part of the window.

The calculation of the graphic can take up some time, especially if the learning file is big. The calculation is being done by a modified MLP network, which needs the time for the training phase.



The single inputs are applied to the **x-axis**. By clicking a bar with the left mouse button the name of the input will appear.



The meaning of the input is applied to the **y-axis**. The higher the bar the more important the input is.

Attention:


Because of the fact, that the weights are preallocated with random numbers, the graphics may vary from each other after repeated calling up.

Attention:

The Input relevance does not provide reliable information concerning the inputs at any rate.



4.5.6 Start

Shortcut: In the toolbar click with the LMB on the icon 

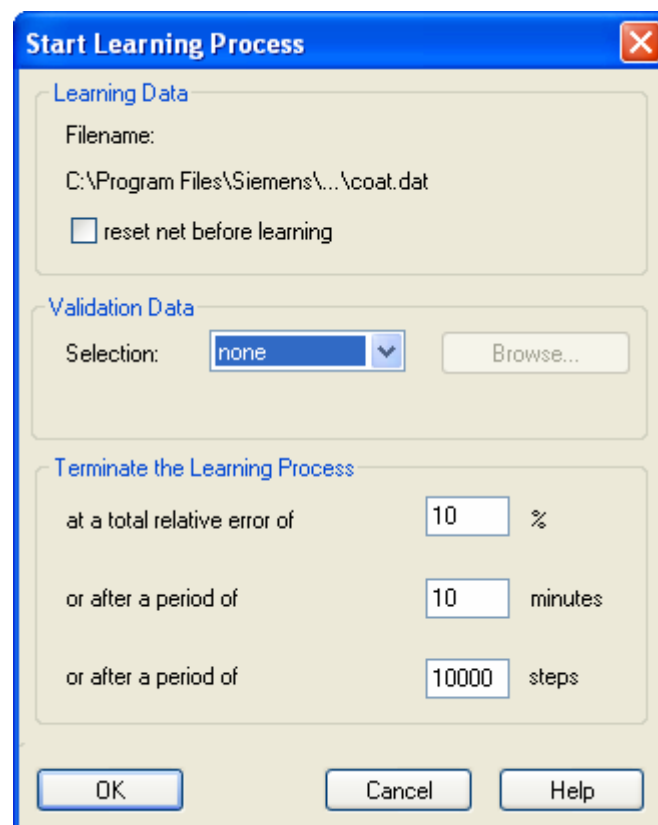
Hotkeys <ALT+L, S>

It is only possible to start learning once you have selected a valid learning data file. After you have activated *Start*, the *Start Learning Process* dialog box appears.

It contains:

- The Name and directory of the associated learning data file,
- Selection option for the validation data,
- Options for terminating the learning.

After you have confirmed the necessary entries, the learning process begins.





Selecting validation data

Using validation data you can check whether the network really has the required input/output characteristic or if it has only learnt the data sets "formally and in parrot-fashion". The network learns using the available learning data and can check the learning aim using a validation data file or validation data selected from the learning data file.

For the selection of suitable validation data you have the following options:

- Use no validation data (*none*)

The entire learning data file is used for learning.

- Select *random* validation data from the learning data file

40% of the data sets to be learnt are selected randomly as the validation data. The network is trained with the remaining 60% of the data sets.

- Perform *drift recognition*

The last 40% of the data sets to be learnt are selected from the learning data file as validation data. The network is trained with the first 60% of the data sets while validation is performed using the last data sets. In this way it is possible to ascertain a drift in the learning data file.

- Use a *file* containing validation data

It is necessary to select a further *.dat file, in which the validation data are located. If you click *Browse* you can load the validation file (via the *Open* dialog box). The complete learning data file is used for learning in this case.

The default setting in *NEUROSYSTEMS* is the entry *none*. If no suitable validation data file is available, it is advisable to retain this setting. With learning data files with a lot of (perhaps also redundant) learning data, the setting *random* should be used.

If the error can not reach the error limit when using validation data this might be caused by too small a quantity of learning data.



Terminating the learning process

In this part of the *Start Learning Process* window you can set "planned" termination of learning depending on two conditions:

- *At a total relative error of ... %*

You can enter a number for the percentage error limit in the enter box. The error is calculated as follows: The sum of the current squared single errors between the desired and the actual output values (number = number of outputs times number of learning and validation data sets) is divided by the number of outputs and the number of learning and validation data sets. The root of this expression is the current total relative error.

- *Or after a period of ... minutes*

You can set a limit on the learning time by entering a positive integer for the maximum learning time in minutes.

- *Or after...learning steps*


By entering a positive, whole number between 1 and 1.000.000 you can determine the number of learning steps.

After you have confirmed the entries in the *Start Learning Process* dialog box with *OK*, learning starts. During the learning process, the *Error Development* window is displayed in which the total relative error is shown as a percentage with respect to the number of learning steps.



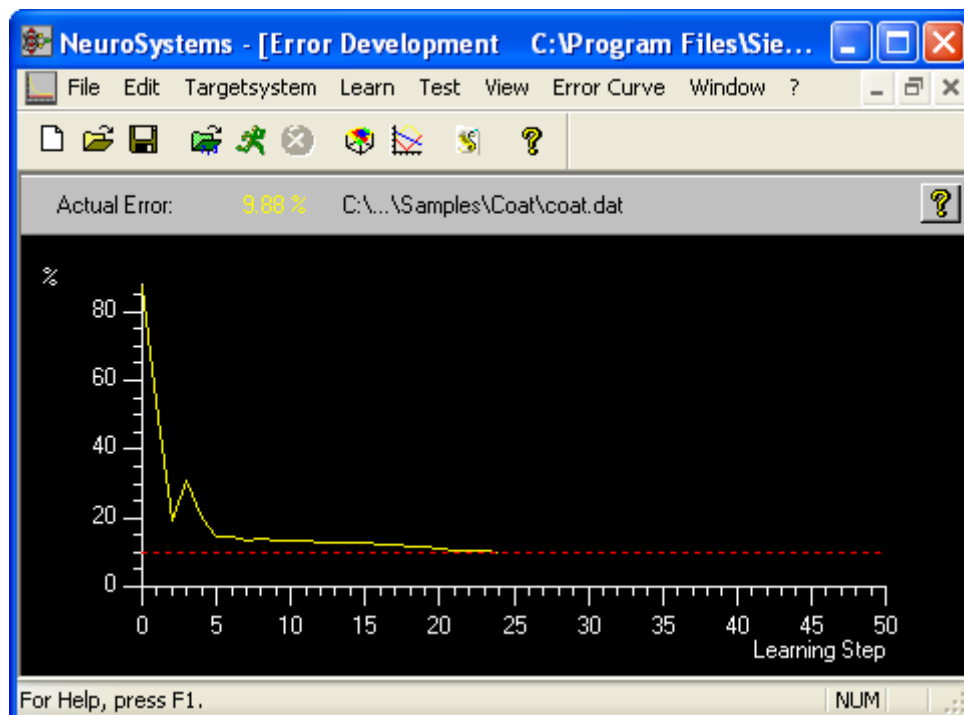
4.5.7 Learn: Stop

Shortcut:

In the toolbar click with the LMB on the icon 

Hotkeys <ALT+L, T>

You can stop the learning process manually at any time with *Stop*. Learning also stops as soon as one of the conditions set before the learning is started (error limit, time limit, learning steps) is fulfilled. After that, a box with the query "Accept the trained network?" appears in the *Error Curve* dialog box. The display below shows an example of an error curve during a learning process that has stopped automatically on reaching the 10% error line.

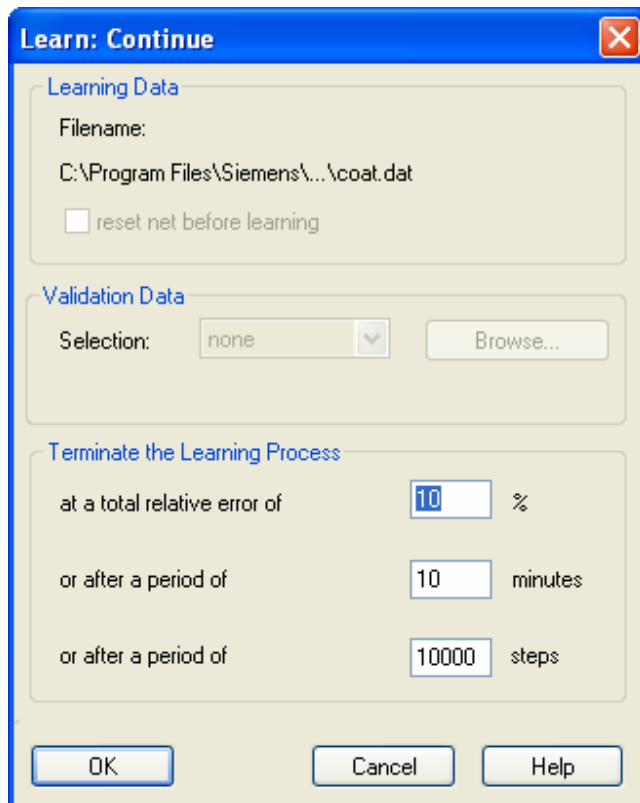


If you answer "No" at this point and do not accept the trained network, the error curve is deleted and the learning result "forgotten". You can trigger the next learning process with "Start". Otherwise, the error curve is retained and the learning result is accepted for the current network.



4.5.8 Learn: Continue

Shortcut: Hotkeys <ALT+L, C>



After you have stopped a learning process, it is possible to continue the learning process with a changed error limit or learning time.

The learning state achieved before stopping must be accepted first (confirmation box: "Accept the trained network?" ... "Yes").

You must make the entries required to continue in the *Learn: Continue* window.

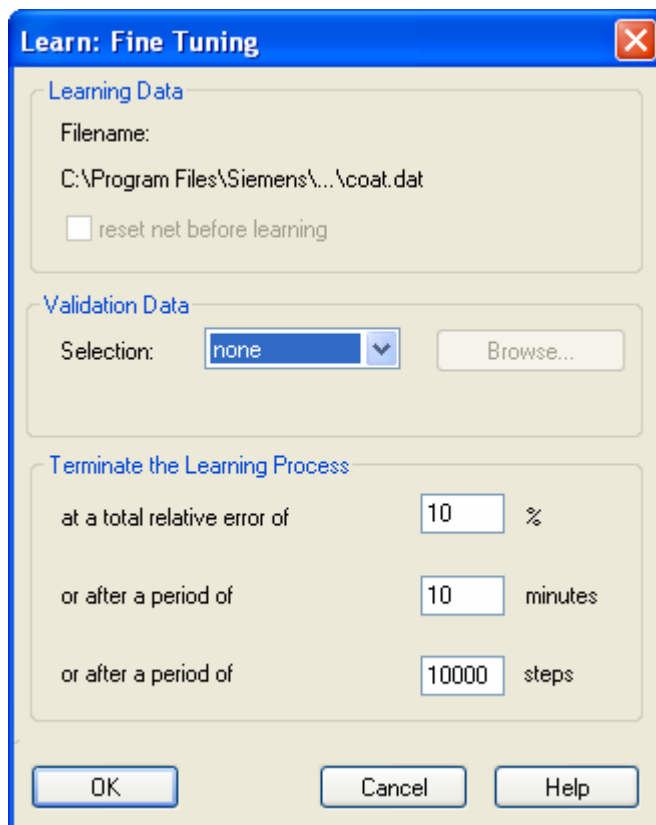


4.5.9 Fine Tuning

Shortcut: Hotkeys <ALT+L, N>

This menu item permits fine tuning of a pretrained network using slightly changed learning data. This can be used, for example, for optimizing a configured network to adapt it to slightly changed conditions later on in practical operation.

Here the connection weights of the neural network are only slightly varied in the learning environment set for the preceding learning process, which affects the fine tuning of the network. The backpropagation algorithm is used here. The result of the fine tuning can be more or less successful depending on the network structure and learning state.

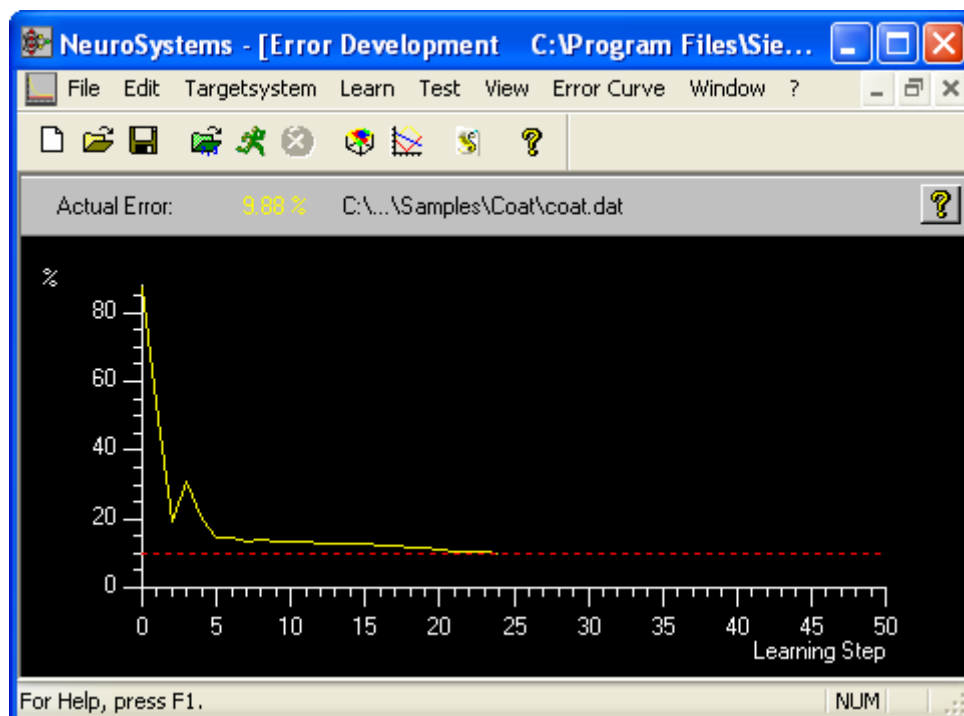




4.5.10 Error Development

Shortcut: Hotkeys <ALT+L, E>

If a learned network has been adopted you can call up the accordant error development once more in the form of an error curve by selecting *Learn, Error Development*.





4.6 Menu: Test

In the *Test* menu you can:

- Check the learning results
- Display the results graphically in various ways.

4.6.1 3-D Graphics

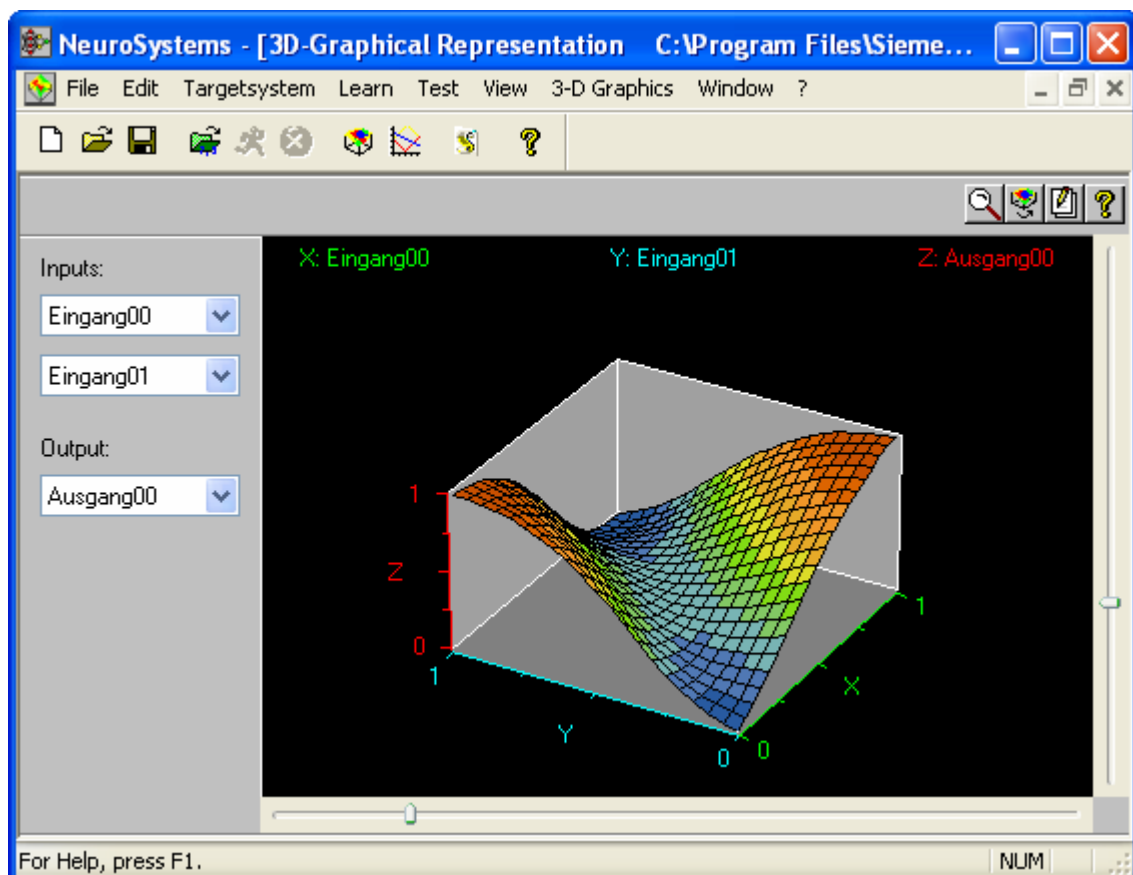
Shortcut:

In the toolbar click with the LMB on the icon 

Hotkeys <ALT+T, D>

3-D Graphic Representation

With the *3-D Graphic Representation* it is possible to view the input/output characteristic of two inputs and one output of the neural network in spatial representation. The inputs are assigned to the X or Y axis, the output to the Z axis.





The figure shows the characteristic surface that you can obtain by training the XOR response. The ranges of the three axes have been chosen (as they were defined during project editing in the *Input/Output Properties* and *Normalize* dialog boxes) as the minimum and maximum for the inputs and outputs in question. Normalization of the inputs and outputs affects the axis scaling in the 3-D display. The dot (front, right in the figure) marks the minimum value of the signals represented by the axes.

You can view the characteristic surface from various angles, if you rotate the axis system horizontally and vertically using the sliders on the lower and right-hand edges of the display. You must drag the sliders with the mouse pointer holding the LMB down.

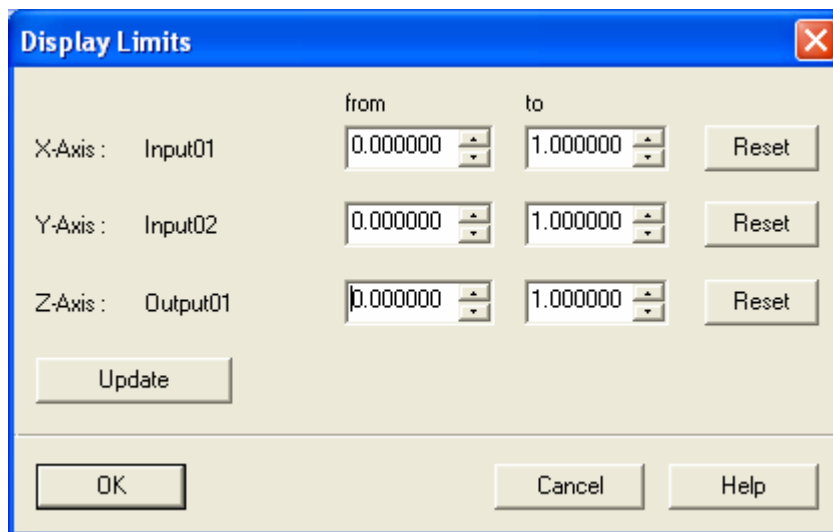
If the project has more than two inputs and/or more than one output, you can make a selection and assignment in the selection boxes for the axes under *Inputs* or *Outputs*. In this way, it is possible to view the entire input/output characteristic of a project with different characteristic surfaces.



4.6.1.1 Display limits

Shortcut: toolbar LMB on the icon

The display areas of the three axis are initially set as they were determined as lower and upper outlier for the according in- and outputs. The determination of the range affects the axis arrangement in the 3-D graphic. According to your requirement you can alter the display limits in this dialog.



4.6.1.2 Animation

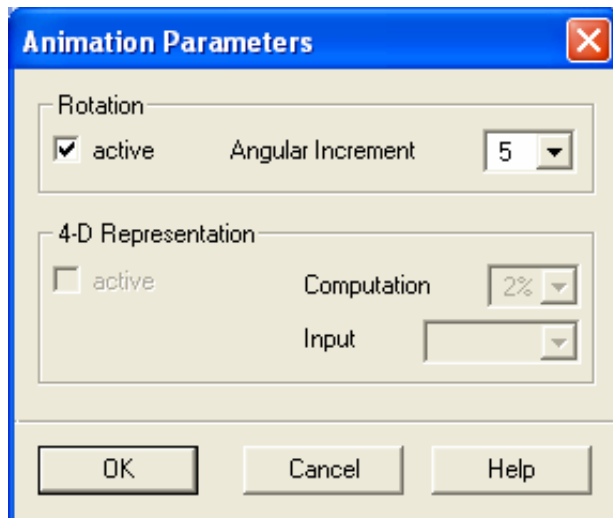
Shortcut: In the *3-D Graphic Representation* window
Switch on: Activate the button
Click with the RMB and select *Animation*
Switch off: Activate the button

The animation function

of *NEUROSYSTEMS* permits automatic changes in the display parameters of the 3-D graphic:

- Rotation of the characteristic surface about the vertical axis
- Change in the characteristic surface in accordance with a fourth parameter that runs through its defined range ("4-D representation")

After you have activated the animation button, the *Animation Parameters* dialog box appears with which you can organize the *Rotation* and the *4-D Representation*:



Rotation

You can select this function by clicking on the associated *active* button. You can set the increments for step-by-step rotation by entering the *Angular Increment*. To do this select one of the angles specified in the enter box (5° to 45°). The larger the angle selected, the faster rotation will be. Rotation begins once you have closed the *Animation Parameters* window with *OK*.

4-D Graphic Representation

This function of *NEUROSYSTEMS* allows you to view the influence of a third input signal on the input/output characteristic of the neural network. The characteristic surface of the 3-D display is repeatedly calculated and updated for successive values of this input signal, considered to be a parameter. In this way, you can obtain an indirect impression of the "fourth dimension" from the sequence of characteristic surfaces. The sequence is repeated cyclically.

You select the function *4-D Representation* by clicking the corresponding button *active*. (Of course, this is only possible if the project has more than two input signals). Under *Input* you have a selection table from which you can pick the "third" input (parameter variable) from the "remaining" inputs. Since you can select any two inputs in the *3-D Graphic Representation* window for the 3-D display, you can also select any input signal as the "third" input signal for the 4-D display function.

You can set the increments for the change of parameter value in *Computation Step*. To do that, select a percentage value in the corresponding enter box. It represents the part of the value range rated at 100% of the parameter value to distinguish successive characteristic surfaces. For example, if you select 5%, an animation cycle contains 20 steps and 21 characteristic surfaces. If you select a greater percentage, the animation runs faster and with



greater increments. The animation begins once you have closed the *Animation Parameter* window with *OK*.

Note: If you set both *Rotation* and *4-D Representation active*, you can have the animation of the 4-D Representation rotate as it runs.

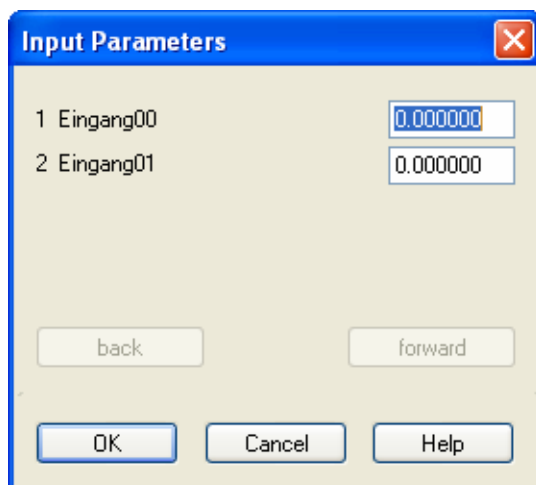
4.6.1.3 Setting the Parameters

Shortcut: In the *3-D Graphic Representation* window:

Activate the  button

In many projects, more than three inputs are used. If you want to include their influence on a certain output signal in the display, you can do that by assigning certain numeric values to those inputs.

After activation the following dialog box called *Parameters* appears in which you can enter the parameter values.




The parameter designations correspond to the input names of the current project. You can assign each input any value (with seven-digit accuracy).


Apart from the two parameters of the input values on the X/Y axis, the values assigned to the other inputs are taken into account directly in the graphic display. If you change an input assigned to the X or Y axis, the new graphic output is based on the parameter value of the removed input.

After confirmation with *OK*, the output characteristic surface which is valid for the parameters set is calculated and displayed.



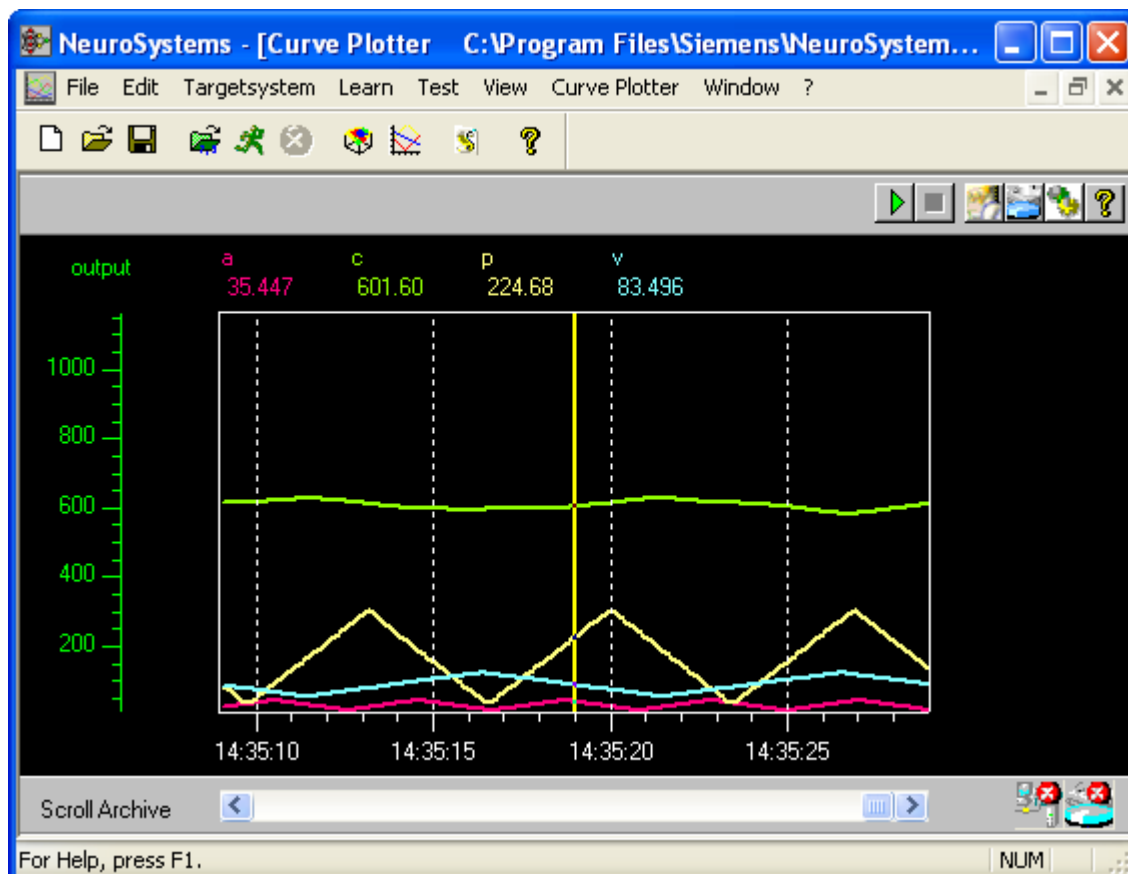
Note: After you have stopped a 4-D Representation, you can read the associated value of the parameter variable ("third" input signal) for the "frozen" surface by activating the Parameter button  and selecting this input. This makes selective evaluation of the influence of the third input signal possible.

4.6.2 Curve Plotter

Shortcut: In the toolbar click with the LMB on the icon 

Hotkeys <ALT+T, C>

The curve plotter permits graphic display of the progression of arbitrary in- and output quantities. The representation is real time. It is possible to archive the curve plotter data on hard-disk and to read it in again when required. The yellow reading line can be moved with the mouse. The values that are displayed above the diagram correspond to the values that have been taken from the reading line.





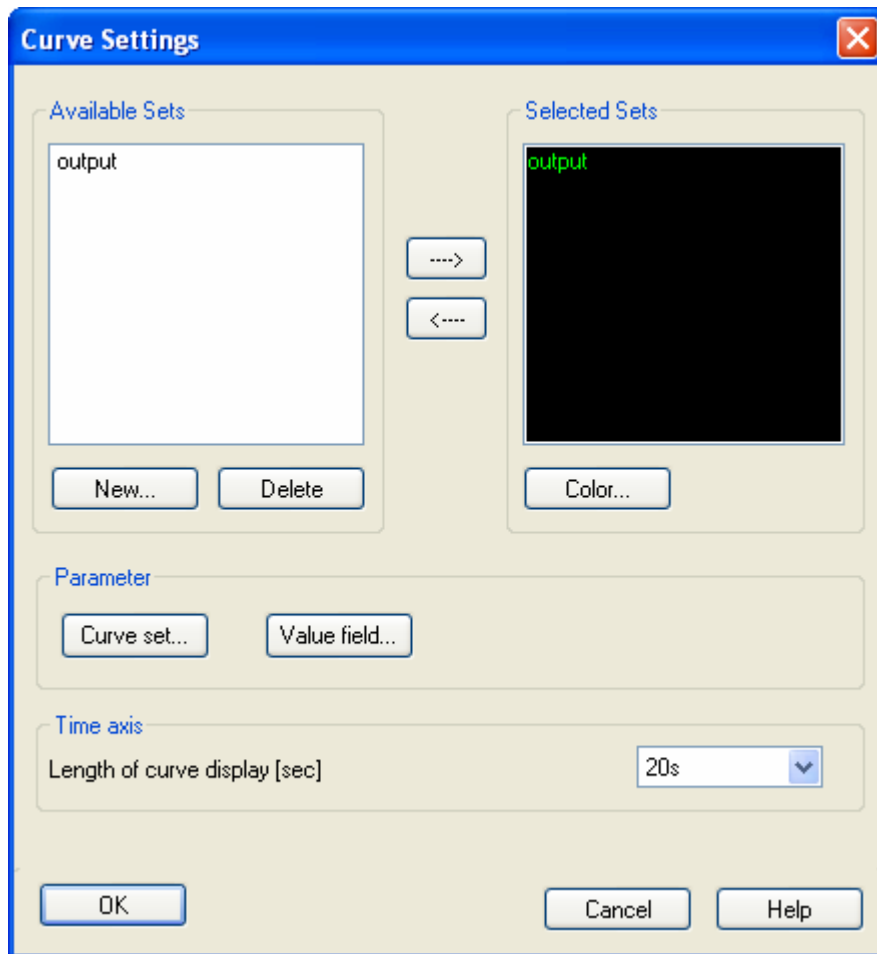
In “stop” mode and with existing curves you can drive onto a curve with the mouse. By clicking on the curve with the LMB the name and value will appear. For generating input signals a curve plotter is available for simulation, which makes vivid graphical test of the Fuzzy system possible.

Note: If the dialog is being opened for the first time, you must apply at least one set in the curve settings. The applied sets will be saved in the project.

4.6.2.1 Curve settings

Shortcut: In the window *curve plotter* LMB on the icon 

The curve plotter can be configured with the help of the sets. A set can sum up arbitrary in- and outputs. Sets can be created arbitrarily. For display in the curve plotter you can select 4 sets at the same time at maximum. Every set has its own y-axis. With the button “new...” you can arrange a new set, with “delete” you can delete a selected set.



A selected set can be enlisted into the right list with the button with the right arrow or by double clicking. The set will appear in the color of the axis. You can enlist four sets at maximum. Consequently you can display four different axis at maximum. By pressing the button “color” you can change the color of the set in the right list. You can create as many sets as you want in the left list.

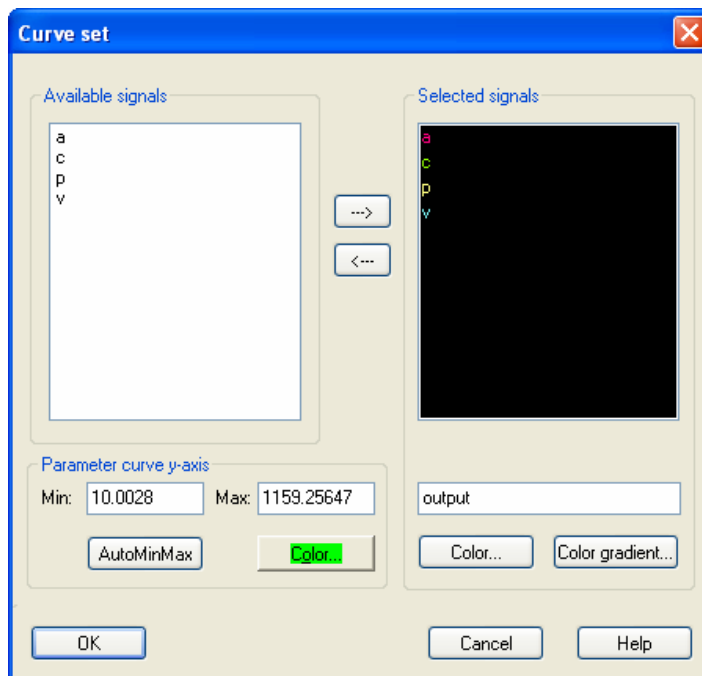
If you select a set from the right list and click the button “curve set....”, you will open the set and will be able to change its settings and add or remove in-/outputs.

A time scale runs along the lower edge of the diagram window, whichs scaling matches the length of the time axis that you set (default: 20 sec.) You can determine the length of the curve display by choosing a value in the window *curve settings*. The displayed time axis has a length between 10 sec. and 5 h. 150 points per curve are displayed and the curve is interpolated linearly between these points.. The resulting sampling time displayed in the *Curve Plotter Settings* window is calculated by dividing the selected length of the curve by 150 (number of points displayed). The sampling time is at least 0,1 sec.



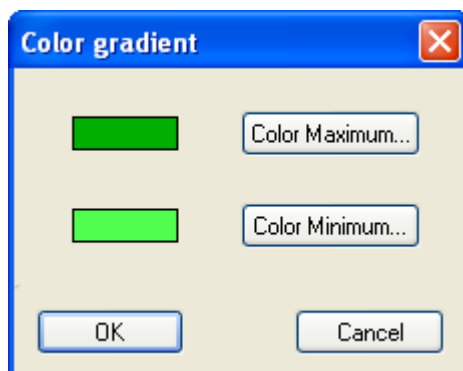
4.6.2.1.1 Curve set

In the *Curve set* window, you will find a list of all the inputs and outputs in the project on the left, and on the right a list of the variables currently displayed in the curve plotter window, along with the curve colors assigned to them. You can define the right-hand list as a selection of signals from the left-hand list. You can do this by selecting the variables in the lists and then clicking the "arrow buttons". It is also possible to double-click on a signal in the left-hand list to place it in the right-hand list and to double-click on a signal in the right-hand list to remove it from the list of signals to be displayed in the curve diagram.



If you want to assign a certain color to a variable in the right-hand list, select this input or output and activate the *Color* button. You can select the basic colors or user-defined colors.

(The basic color is always used for display of the axes and the curves!)





The value range displayed is always selected automatically as it was defined in the *Properties* window of the corresponding input and output (*Minimum* and *Maximum* in *Normalize*).

If you want to change the curve scaling, you click on the *AutoMinMax* button, or you can enter the minimum and maximum of the value range manually.

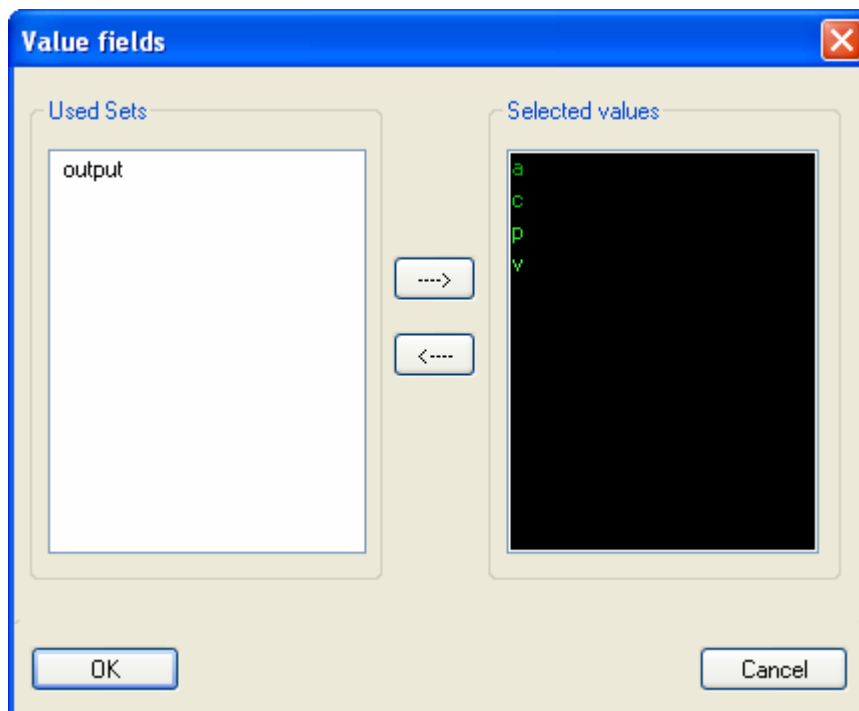
At “parameter Y curve axis” you can arrange settings for the axis of the set. The min- max value describes the range of the Y-axis of this set. With the button *autominmax* you can determine the min- max value of the selected signals. With the button *color* you can determine the color of the axis. The button shows which color is presently set.

After closing the dialog with OK the recently created set will appear in the list of *curve settings*.

4.6.2.1.2 Value Fields

Via the button „value field...” you open a dialog in which you can display single signals of selected sets as a value above the curve diagram. Value fields

The left list in this dialog shows the selected sets. By double clicking on one of the sets you can make the signals visible in the left list as well. Single marked signals can be transferred into the right list by clicking the button with the arrow on it.







Signals that have been selected in the value display, will be drawn a bit thicker in the curve plotter.

4.6.2.2 Start


Shortcut: In the *Curve Plotter* window:

Start: Activate the  button, Starting recording“

You can start the recording by activating the  button. Recording progresses from left to right and is updated in the cycles of the sampling time. When the end of the diagram window is reached, the entire curve display is shifted left by a quarter of the diagram length. During recording, the current values of the signals entered are displayed (in the curve color) in the upper part of the diagram.



4.6.2.3 Stop

Shortcut: In the *Curve Plotter* window:
Activate the  button

If you press the *STOP* button, recording is stopped.

4.6.2.4 Archiving the Recording Data in the Computer RAM

At the end of the recording, not only the currently displayed section of the curve (in accordance with the set *Length of the curve display*), but also a range to the left of the current section is available. Archiving the values permits retrospective analysis of the behavior of the neural application. The curves are archived in the computer RAM up to a length of about 500 sampling values; all "previous" values are cut off. If the length of the recording is not sufficient, or if you want to archive the recording, there is a disk archive made available for you.

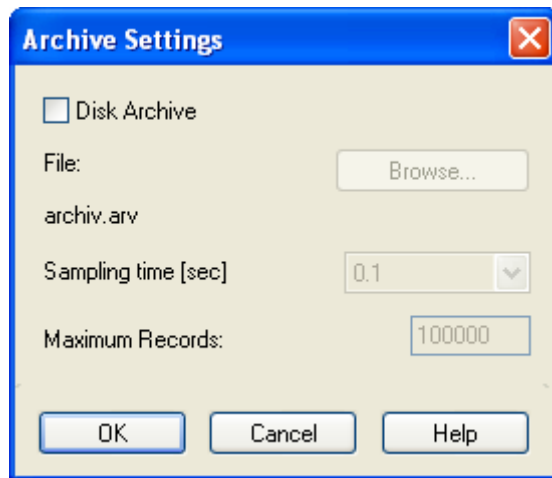
The Disk Archive

Shortcut: In the *Curve Plotter* window: Activate the  button

The option of archiving curve plotter data on (hard-) disk has the following advantages:

1. You can store more data than is possible in the computer RAM.
2. Permanent storage of data is possible.
3. You can analyze the data again later on.
4. You can use the archived data directly as learning data later on.

If you want to archive the data on the disk, you must activate the *Archive Settings* button before starting the recording. Check the option *Disk Archive* in the *Curve Plotter Settings* window and state the maximum number of entries. The number of entries must be at least 10 and no more than 100,000. The data is stored in an *.arv file (archive file). With *Browse* you can either open an existing *.arv files or creates a new archive file.




When you open an existing archive, the stored data are displayed in the curve plotter diagram. The limits of the value ranges are automatically adapted to the archive data.

If you start recording again, a query is displayed, asking whether you want to overwrite the existing archive data, which would delete it.

If an archive is selected, it will be displayed in the curve plotter by the following symbol



If no archive is selected it will be displayed by the following symbol. 

Archive Analysis

This section applies both to archiving in the RAM and on the disk.

The scroll bar under the diagram is used to view the archived parts of the curves ("Scroll Archive"). If you click the right and left arrows, the diagram is scrolled in fine steps. If you click between the arrows and the slider, it is scrolled in coarse steps. Fast positioning by "dragging" the slider is also possible. Archive Analysis

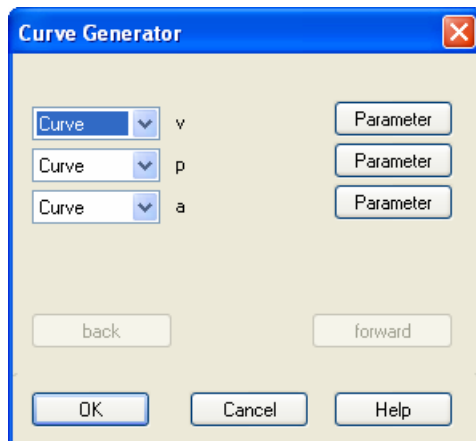
Use of the vertical positioning line, which appears after you stop recording, permits a precise analysis. You can place it in any position in the displayed section of the curve with the mouse. In the upper part of the diagram, the values of the recorded signals that are valid at the position of the positioning line are displayed.

During analysis of the archive, it is also possible to change the inputs and outputs displayed. If you select another input or output signal, the curves and the numeric values are updated.



4.6.2.5 Curve Generator - Parameterization of the Curves

Shortcut: In the *Curve Plotter* window: Activate the  button Curve Generator



To test the input/output characteristic of the configured neural network you can stimulate the inputs with different signals using the **curve generator** and observe and analyze the response of the network with the curve plotter. You can set the input curves by activating the *Curve Generator* button. Stimulation with constant or triangular signals is possible. The default setting for all inputs is *constant* stimulation with the value 0.0. If there are more than five inputs, you can switch between them with the *Forward* and *Backward* buttons.

If you activate the arrow corresponding to an input, you can set the following values:

- **Constant:**

After you have entered a numeric value and confirmed with *OK*, the numeric value is accepted and displayed.

- **Maximum:**

To assess the effect of the input signals at their value range limits, you can set the input in question to the maximum value of the value range using the *Maximum* menu item. This value is the *Maximum* set in the *Input Properties* window under *Normalize*.

- **Minimum:**

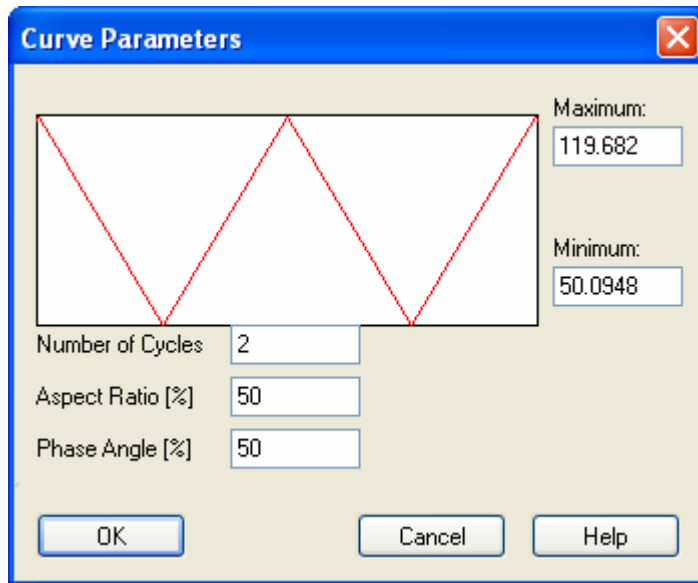
Similarly, you can apply the minimum value of its value range to the input by selecting *Minimum*. This value is the *Minimum* set in the *Input Properties* window under *Normalize*.



- **Zero:**

If you select the menu item *Zero* you set the input concerned to zero.

- **Curve:**



If you select the menu item *Curve* the input is stimulated with a triangular curve.

In the *Curve* window you can adapt the input signal individually:

1. Define the amplitude of the test signal by entering the values for *Maximum* and *Minimum*. If these values correspond to the range limits of the input signal, the ramp function just touches the top and bottom horizontal borders of the window, but smaller and larger values are also possible.
2. Enter a value for the number of cycles. This specifies how often the triangular test function is repeated during the length of the curve display. The larger the value, the steeper the ramps.
3. The aspect ratio is the percentage ratio of the time to the peak to the cycle time. The default setting is 50%.
4. The phase angle defines the starting point of the test signal with respect to the curve display in the diagram. For example, a phase angle of 25% shifts the test signal right by 25% of the length of the curve display. The default setting is 0%.



After each change in the parameters, the schematic diagram of the test signal is recalculated and updated in the graphic display of the window. If you press the *OK* button, the parameters set are stored and the *Curve* window is closed. In the *Curve Parameters* window, *Curve* is displayed instead of the numeric value for the input concerned.


Further the simultaneously resulting values for the outputs will be displayed. Only if the values of The display will only happen, if constant values have been adopted for the inputs. If curve is selected for an input, than the prinout „???“will appear for all outputs.

4.6.2.6 Online/Offline

If you are using an online connection, the following symbol will appear in the curve plotter



. This means., after connect targetsystem, the online data from the controls . (SIMATIC S7-300, SIMATIC S7-400) will be shown and can be recorded. Online/Offline

If you are using an offline connection, the following symbol will appear .

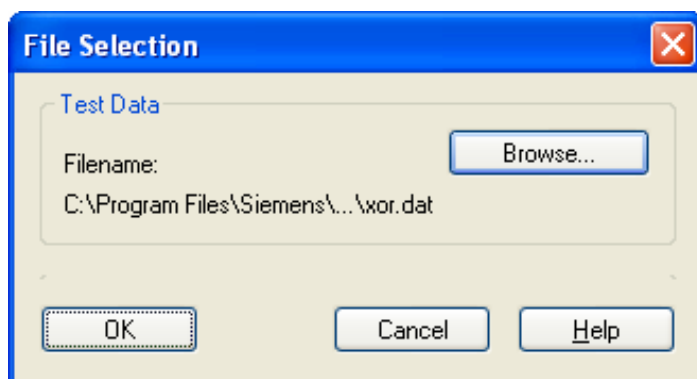
4.6.3 File Selection

Shortcut: Hotkeys <ALT+T, F>

This option allows you to load a test data file (test file) . This file has the same structure as a learning data file. It can be used to test the properties of a trained network.

A test file differs from a validation data file in that the error values resulting from the test file do not affect the error curve and the error diagram. (The results of testing the learning process with the validation data file are used for the error calculation. So they are displayed in the *Error Curve* and *Error Diagram*.)

If you click on *File Selection* in the *Test* menu, the *Test File Selection* dialog box appears.





If you activate *Browse* you can load an ASCII file **.dat* from the *Open* dialog box which then appears. After it has been loaded, its name is displayed in the dialog box *Test File Selection*. After you close the window with *OK*, the file is used as the test file for all further work with *NEUROSYSTEMS*.

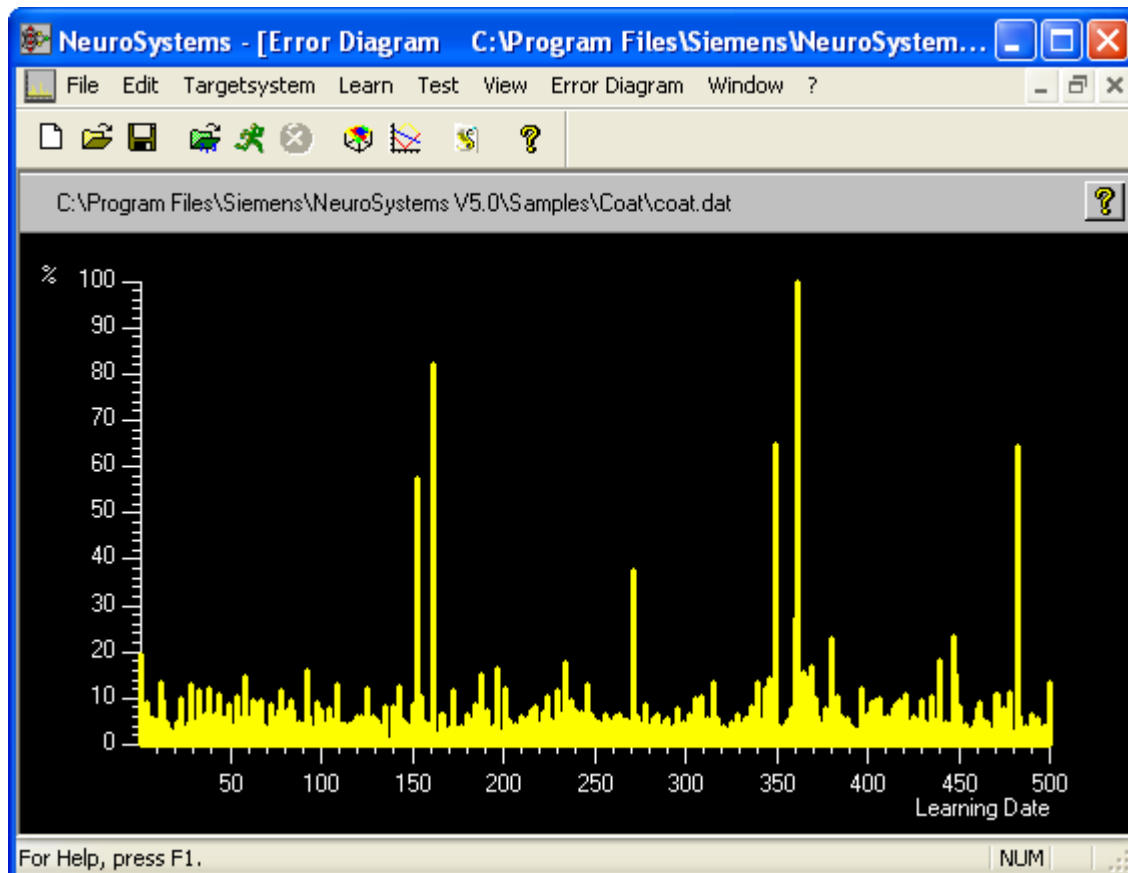
Note: Until you load a test file in the current project, *NEUROSYSTEMS* uses the selected learning data file.

4.6.4 Error Diagram

Shortcut: Hotkeys <ALT+T, E>

The *Error Diagram* display is used to provide a visual examination of the learning success. For each learning and validation data set, the relative error is entered as a column. The error calculation is based on the Euclidean norm of deviation of the normalized output vectors that are to be compared. The smaller the relative error, the better the neural network has learned the learning data.

Note: The window can only be opened, after a learning file or test file has been selected



During the most errors many learning data sets are existent, whereby the error diagram moves together to a tight column sequence. Thus the error auf the single learning date can hardly be localised, the visual impression of the entire learning succes however will be confirmed.

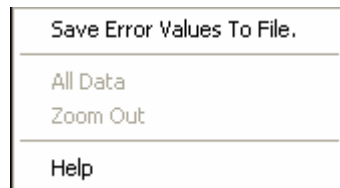
If you want to look at the errors of the learning data in detail, you have the possibility to zoom in single cut-outs within the diagram.

Zoom In:

By clicking on the display with the left mouse button, holding the button and moving the cursor and than letting go of the button you can select a cut-out and display it. Repeated zooming in is possible and allows precise examinations within a certain range of the x-axis.

Context sensitive menu:

You get to the context sensitive menu for the diagram distribution by clicking and letting go of the display field with the right mouse button. The following select box opens.

**Save Error Values To File:**

By choosing this option you can save the error values in a *.dat-file (ASCII-format).

All Data:

By choosing this option you have the possibility to get back to the general view after zooming.

Zoom Out:

By choosing an entry you have the possibility to go back one step and to look at the next to last selection.

Help:

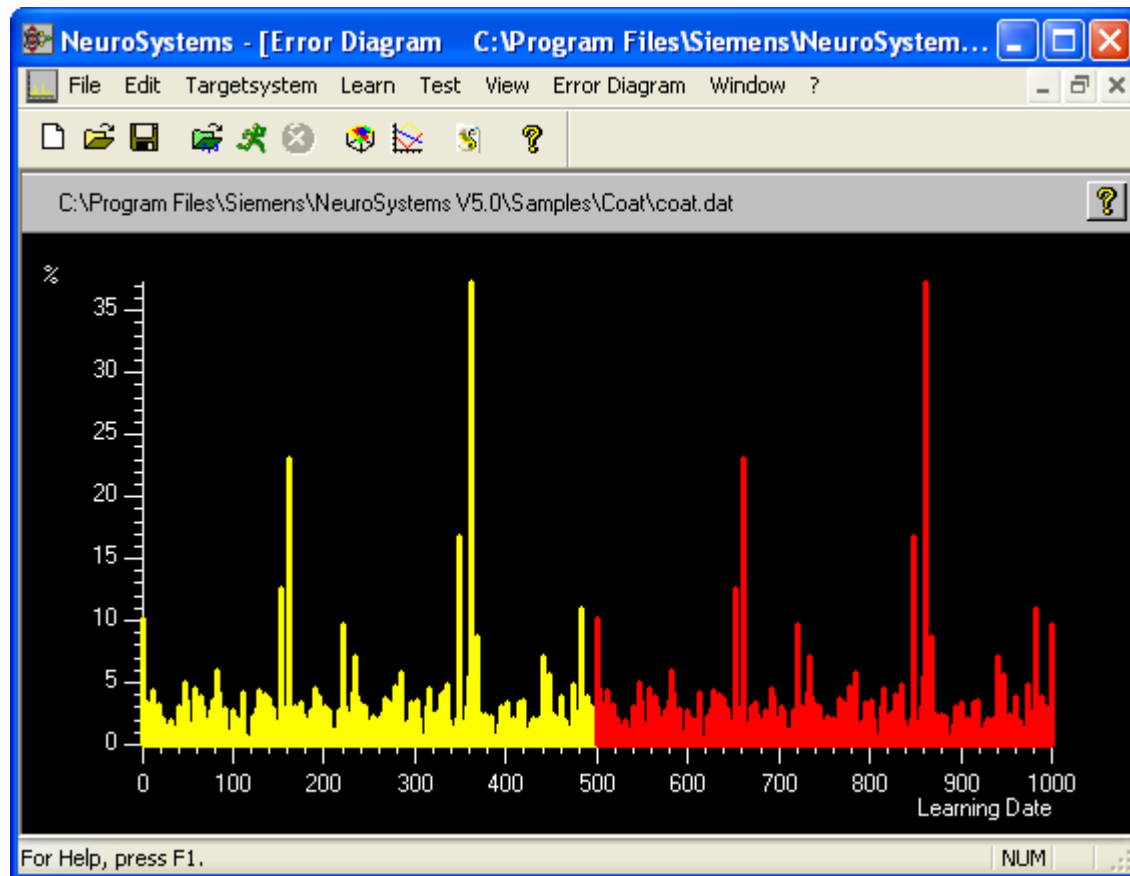
By choosing this option you get to the help for this window

The error diagram refers to the learning data sets and chosen validation data sets, **NOT** to the test data file.

The error for each learning data and validation data set is calculated as follows:

The sum of the quadratic single errors between the “to be” and “is” output values (number = number of outputs) is divided by the number of outputs. The root of this expression is the error for one particular data set shown in the error diagram.

If **validation data** is used for the learning process, the diagram lines of the validation data are displayed in **red**. The according diagram lines of the learning data are displayed in **yellow** by contrast.



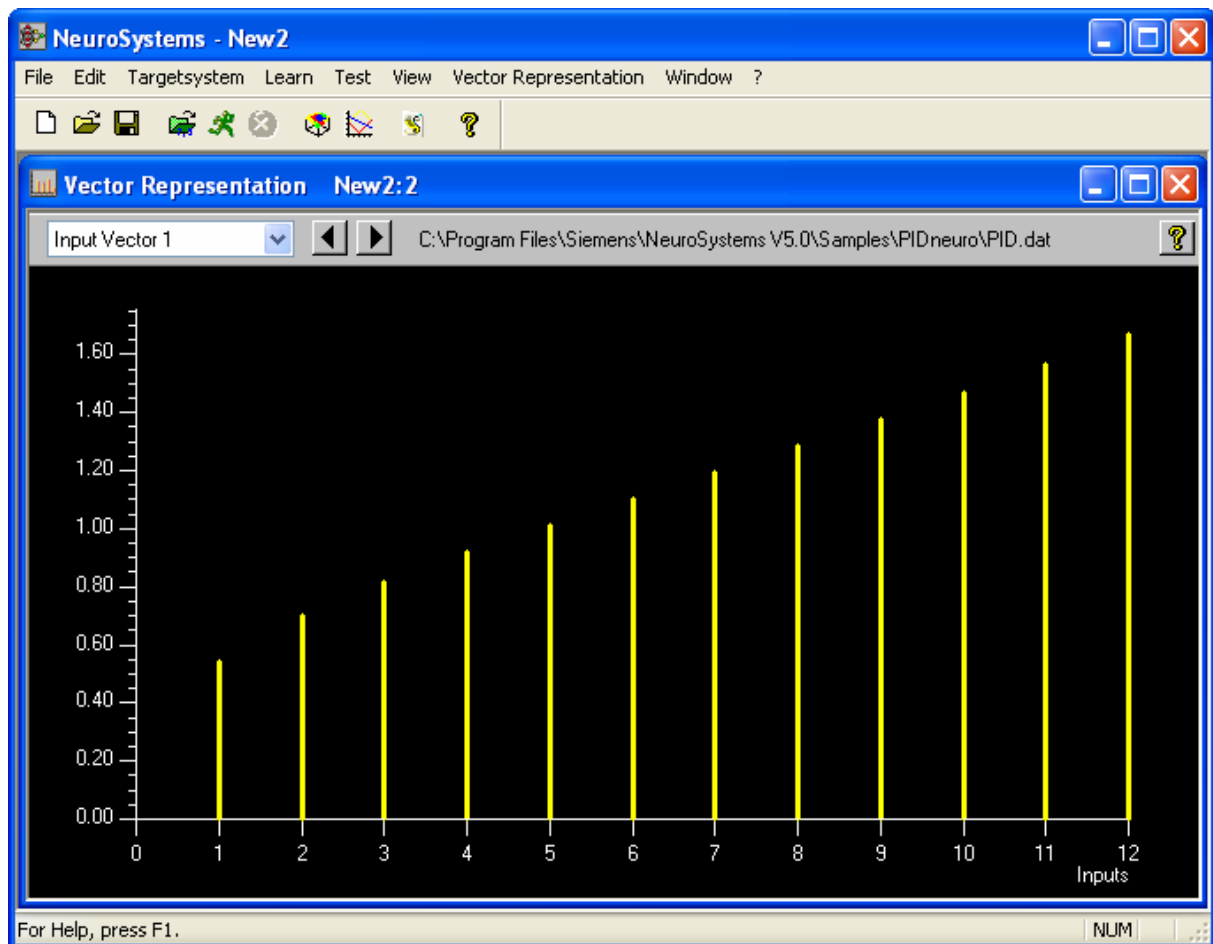


4.6.5 Vector Representation

Shortcut: Hotkeys <ALT+T, V>

In the *vector representation* window, you can either display an input vector or an output vector of the test file. If you display the output vectors of the test file, they will be compared to the output vectors calculated by the network. The test output vector is **yellow** and the output vector generated by the neural network as a response to the test input vector is displayed in **red**. This shows you to what extent the network has the required input/output characteristic if using data other than those from the learning data file.

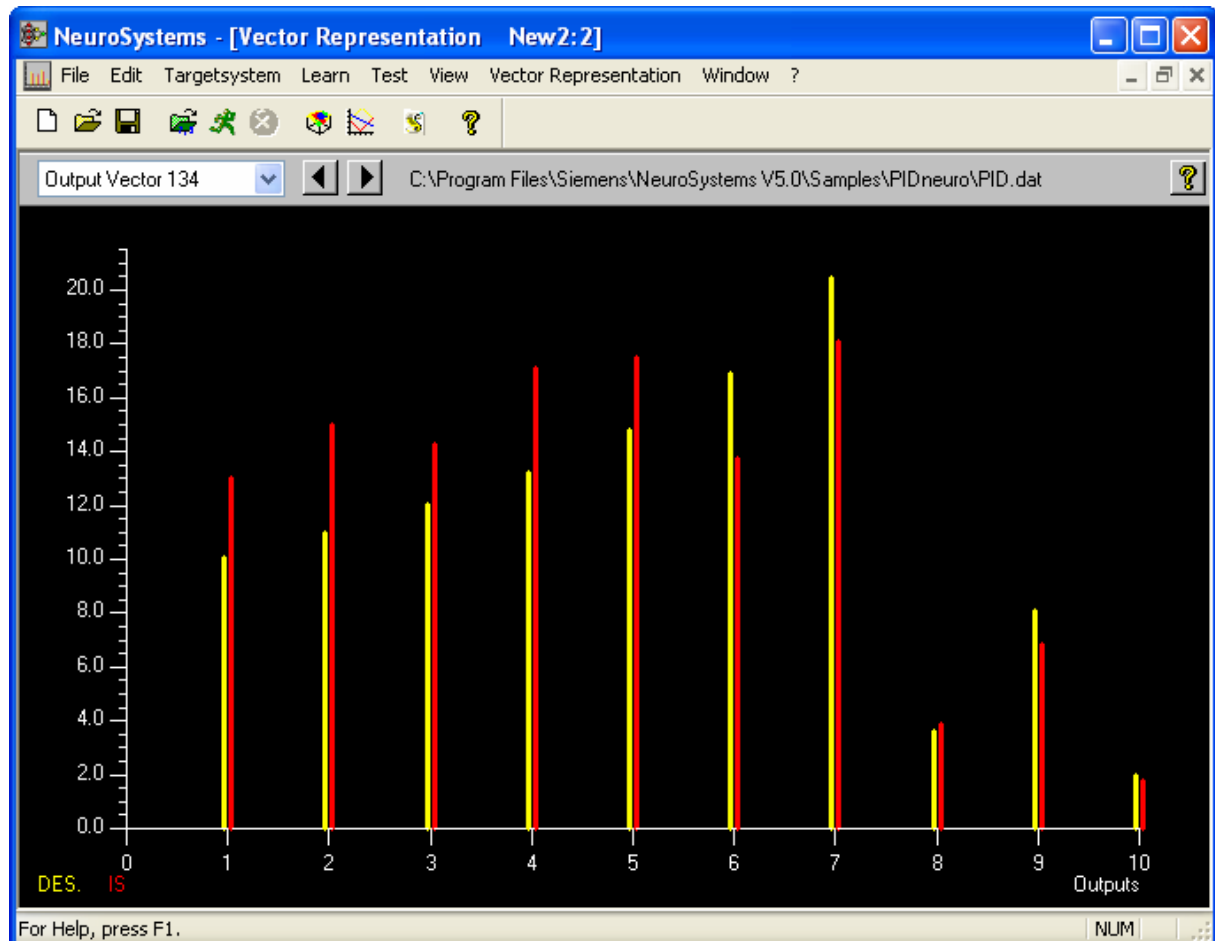
Note: The window can only be opened, after a learning file or test file has been selected



The display, for example, shows the second input vector of a XOR test file with its two components $E1 = 0.9$ and $E2 = 0.1$. (Please note that the number of the vector components begin at 1, whatever names were selected for the inputs, e.g. Input00, Input01 etc.)



If you have not selected a file with the *File Selection...* option before you activate the *vector representation*, no test file is loaded, and the learning data file is displayed in the *vector representation* instead.



Note: If you click the display with the *RMB* a selection box opens. If you select *Save data to file*, you can save the error values to a *.dat file (ASCII format).

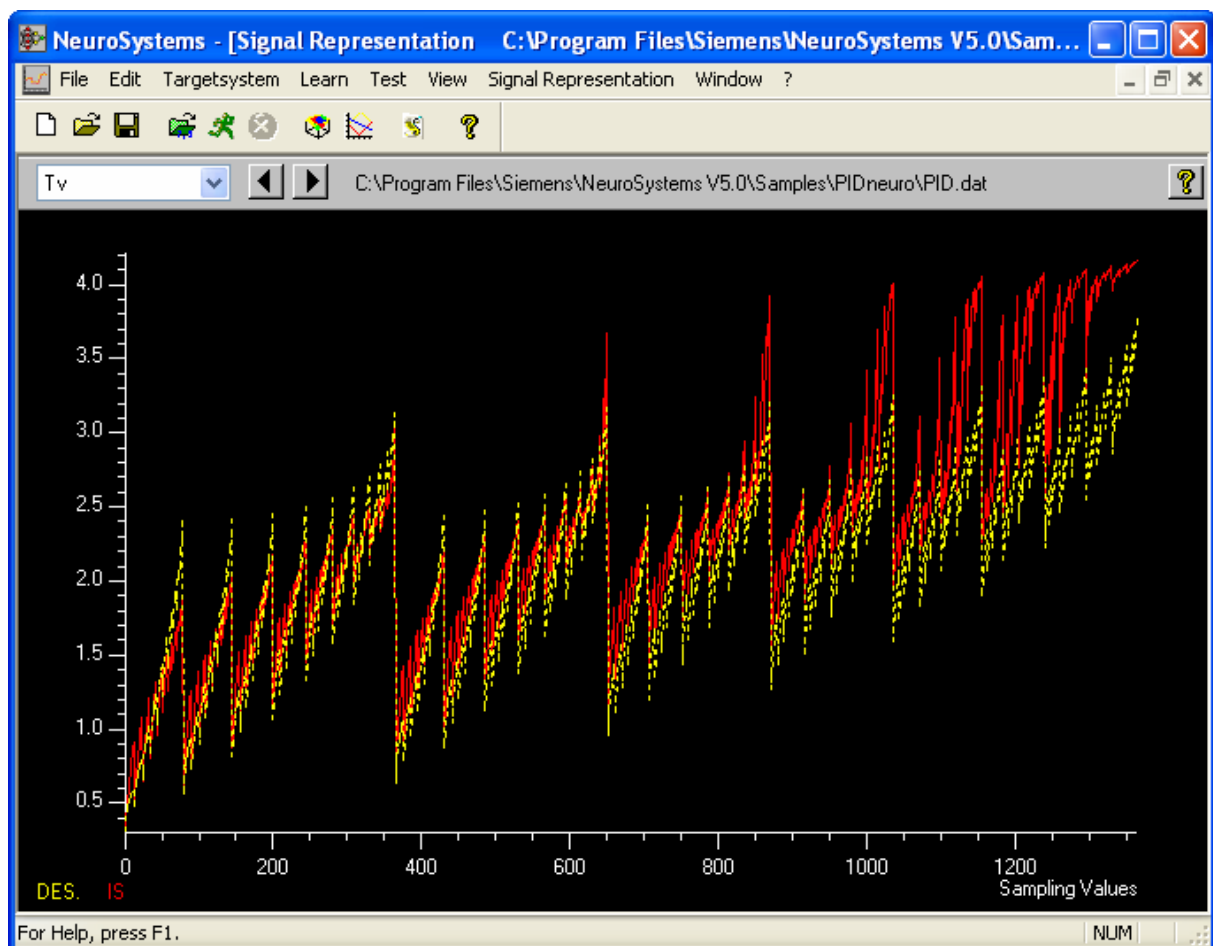


4.6.6 Signal Representation

Shortcut: Hotkeys <ALT+T, S>

In the *signal representation* window, you can graphically display the chronological sequence of either one input or output signal, according to the test data file. The signal curve is interpolated linearly between the values for the points in time (sampling values). The output signals of the test file are displayed together with the output signals calculated by the network. The test output signal is displayed **yellow sketched** and the output signal generated by the neural network as a response to the test input signal is displayed in **red**. This shows you to what extent the network has the required input/output characteristic if using data other than those from the learning data file.

Note: The window can only be opened, after a learning file or test file has been selected.





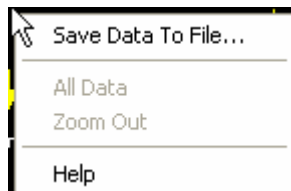
If you have not selected a file with the *File Selection...* option before you activate the *signal representation*, no test file is loaded, and the learning data is displayed in the *signal representation* instead.

Zoom In:

By clicking on the display with the left mouse button, holding the button and moving the cursor and then letting go of the button you can select a cut-out and display it. Repeated zooming in is possible and allows precise examinations within a certain range of the x-axis.

Context sensitive menu:

You get to the context sensitive menu for the diagram distribution by clicking and letting go of the display field with the right mouse button. The following select box opens.

**Save Data To File:**

By choosing this option you can save the values in a *.dat-file (ASCII-format).

All Data:

By choosing this option you have the possibility to get back to the general view after zooming.

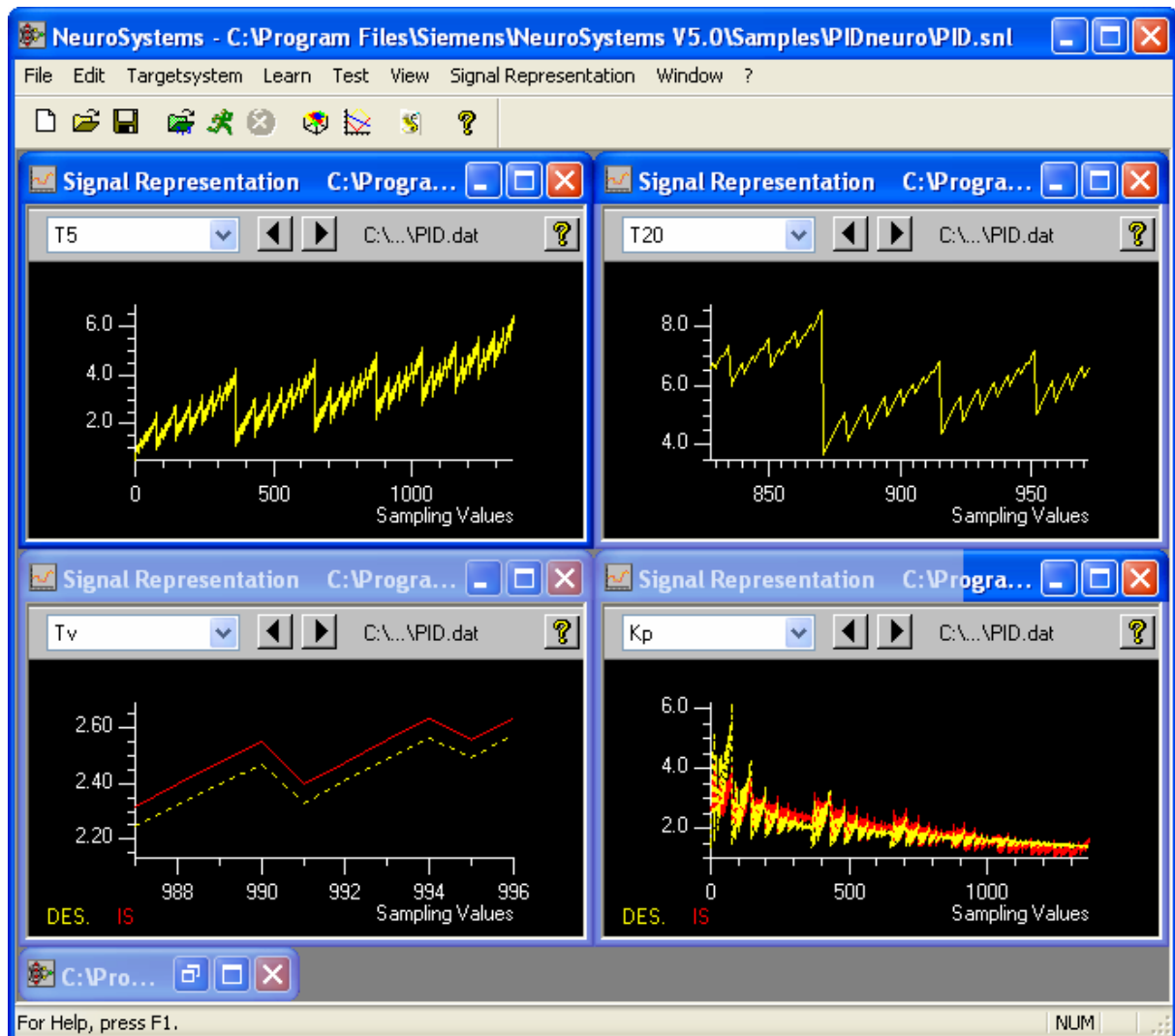
Zoom Out:

By choosing an entry you have the possibility to go back one step and to look at the next to last selection.

Help:

By choosing this option you get to the help for this window.

The window signal representation can be opened several times and and therefore offers an overview of the relations of all input signals to one output signal.



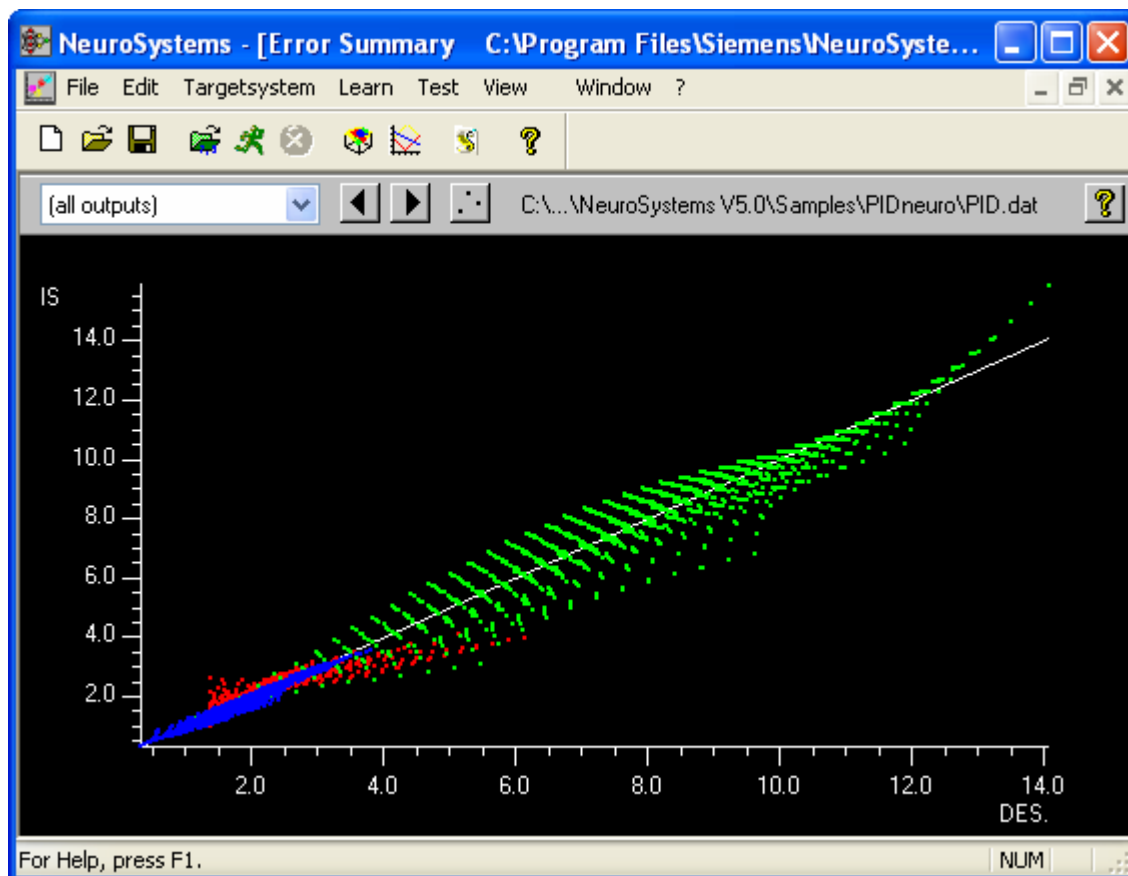


4.6.7 Error Summary

Shortcut: Hotkeys <ALT+T, R>

In the window error summary the “to be – is – aberration” of the output signal is displayed graphically. In this connection the “to be” values of the test file are compared with those that have been calculated by the network (“is values”). The aberration is displayed graphically.

Note: The window can only be opened, after a learning file or test file has been selected.



You must arrange the option *file selection* from the menu test prior to activating the *error overview*. If you don't the test file will not be loaded and a learning file will appear in the error overview. The name of the file will be shown in the upper part of the window.

The **button** with the three dots is used to maximize/minimize the displayed dots.



In the combo box you have the possibility to select each output individually or all outputs in total. You can switch between the outputs with the backwards- forwards button.

The diagonal that is plotted in the diagram is corresponding to the ideal network behaviour. All dots should be on, or next to this line. The output signals that have been created by the neural network as an answer to the test input signals are displayed as a dot cloud.

On the **axes** the set range of the output, which has been set in *output properties*, is plotted.

The x-axis the “to be value”, the y-axis is corresponding to the “is value”. Consequently the “to be – is – aberration” is easy to recognize through the distance to the diagonal.

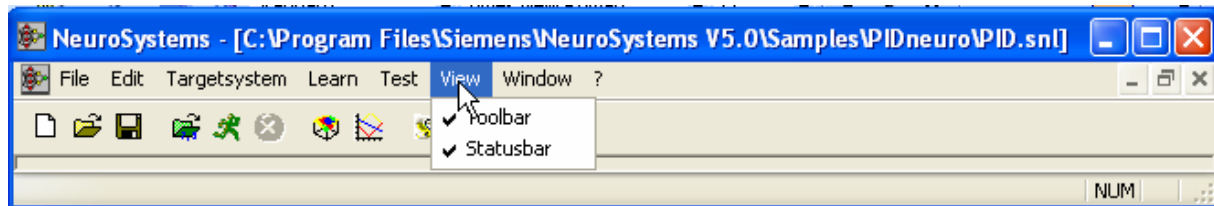
This diagram allows a quick estimation of how good the network is trained and in which areas it is deflecting from the ideal behavior to strongly. It also can be used to do a quick comparison to other trained networks. For this you must open the *error overview* and project data of the networks that are to be compared.

Further you can easily visualize how good the network reacts to other test data.



4.7 Menu: View

In the *View* menu you can show or hide the *Toolbar* and the *Status Bar* by clicking the menu items with the LMB.



4.7.1 Toolbar

Shortcut: Hotkeys <ALT+V, T>

The toolbar is displayed if a checkmark appears in front of the menu item in the *View* menu. It is positioned horizontally at the top of the working window underneath the menu bar, and contains the most frequently required command buttons in *NEUROSYSTEMS*.



The toolbar provides quick access to the program options of *NEUROSYSTEMS* simply by clicking with the LMB.

Icons and functions of the toolbar



Opens the dialog box *New...* to create a new *NEUROSYSTEMS* project.



Opens an existing *NEUROSYSTEMS* project and displays the *Open* dialog box, in which you can find and load the required file.



This saves the current project under the current name. If the project has not yet been given a name, *NEUROSYSTEMS* opens the *Save as...* dialog box.



This opens the “*Select Learning File*” dialog box to load a learning data file.



Start learning process.



Stop the learning process.



Activate *3-D Graphics*.



Start the *curve plotter*.



Opens the *NEUROSYSTEMS* help window.

4.7.2 Status bar

Shortcut: Hotkeys <ALT+V, S>

The status bar provides you with information on the current operating situation. It is displayed if there is a checkmark in front of this item in the *View* menu. The status bar is displayed on the lower edge of the *NEUROSYSTEMS* working window.



Left-hand side of the status bar

As you move through the menus with the mouse or the cursor keys, this part displays a short description of the menu command selected or the button on the toolbar selected with the mouse pointer.

Right-hand side of the status bar

This displays which of the following keys are "locked":

- *CL* The *CAPSLOCK* is activated.
- *NUM* The *NUM* key (keypad for numeric input on the right of the keyboard) is locked.
- *SL* The *SCROLL* key is locked.

Note: A yellow information box, that follows the mouse pointer, appears after a short time if you hold the mouse pointer at the same position ("Tooltips"), to provide you with running assistance when working with *NEUROSYSTEMS*.



4.8 Menu: Window

With the *Window* menu you can set how the windows of *NEUROSYSTEMS* are displayed. It contains the following commands:

- ***Cascade***

Shortcut:	Hotkeys	<ALT+W, C>
------------------	---------	------------

All windows, except those that have been minimized to icon size, are positioned overlapping diagonally and have the same size. The active window is in the foreground.

- ***Tile***

Shortcut:	Hotkeys	<ALT+W, T>
------------------	---------	------------

All windows, except those that have been minimized to icon size, are arranged next to each other without overlapping (if there is more than one).

- ***Arrange Icons***

Shortcut:	Hotkeys	<ALT+W, A>
------------------	---------	------------

The windows that have been minimized to icon size are arranged along the lower edge of the working window of *NEUROSYSTEMS*. If there is an opened project window in this space, some or all of the icons may be covered.

- ***Window1, 2, ...***

Shortcut:	Hotkeys	<ALT+W, 1> etc.
------------------	---------	-----------------

At the end of the *Window* menu there is a list of opened windows. A checkmark appears in front of the active window name. If you select an entry in this list, you can switch directly to that window, which becomes the active window.

Note: Use the part *Window1*, *Window2*, ... in the *Window* menu to find out which windows are open because some of them might be minimized or covered by other windows.



4.9 Menu: Help ("?")

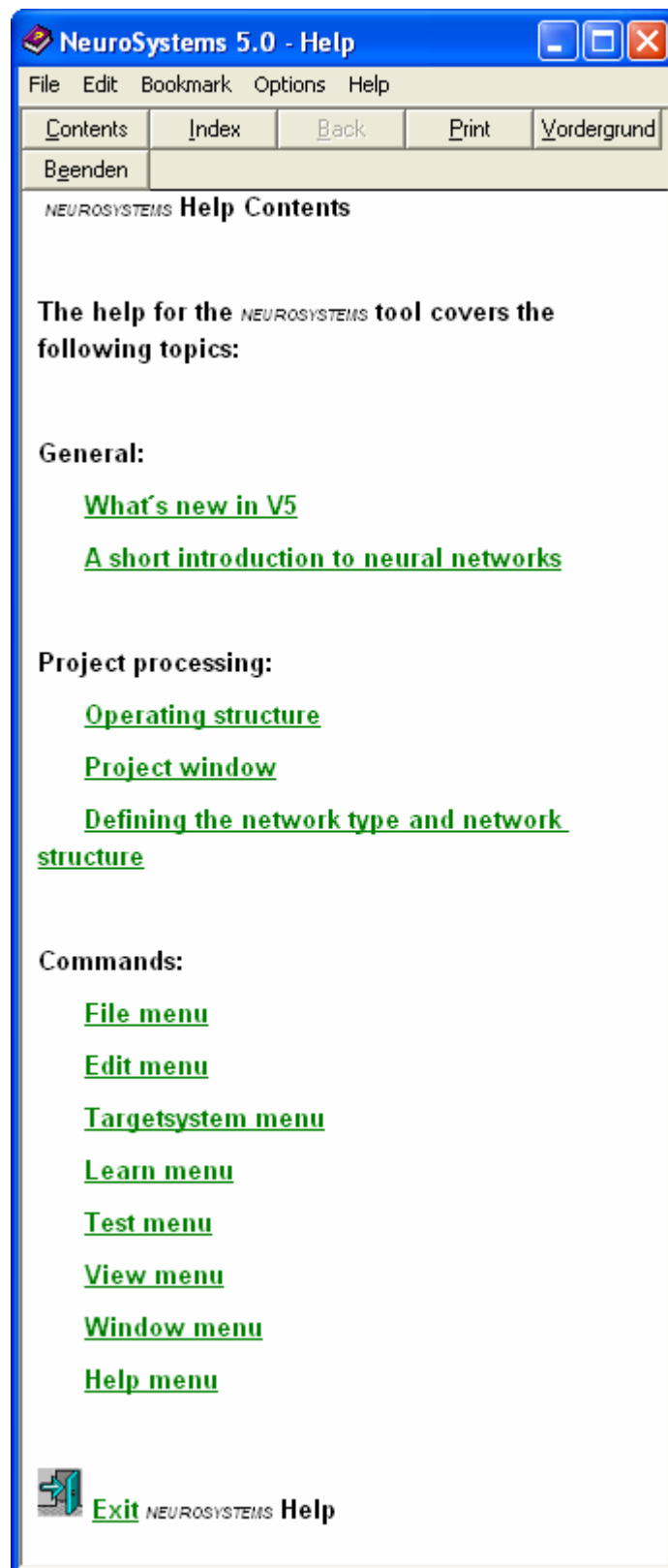
The *Help* ("?") menu gives you assistance in using *NEUROSYSTEMS*. It is subdivided into the following sections:

- Help Topics

Shortcut:

Hotkeys<ALT+?, H>

This part opens the online help window . From this window, you can obtain information by going straight to a specific topic or by branching down step by step until you reach the topic.



With a mouse click on the underlined words (highlighted in color) in the help text, you can jump to a further help topic. To return to the previous information click on *Back* in the header



bar of each help window. The *Contents* calls up the first help window directly. A mouse click on the green words with a broken underlining opens an explanation box.

With the *Foreground* button you can keep the help window in the foreground of the screen as you work with *NEUROSYSTEMS*.

- **Using Help**

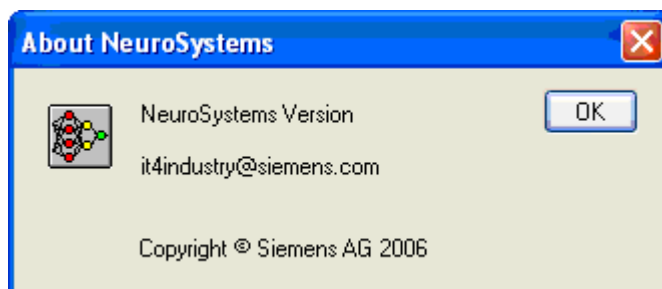
Shortcut: Hotkeys <ALT+?, U>

This command opens the *Windows Help* window. Here you can inform yourself about the use of the help system.

- **About NeuroSystems**

Shortcut: Hotkeys <ALT+?, A>

Here you can obtain information on the copy of the *NEUROSYSTEMS* program. Use this command to display the copyright message and the version number of *NEUROSYSTEMS*.





5 Miscellaneous

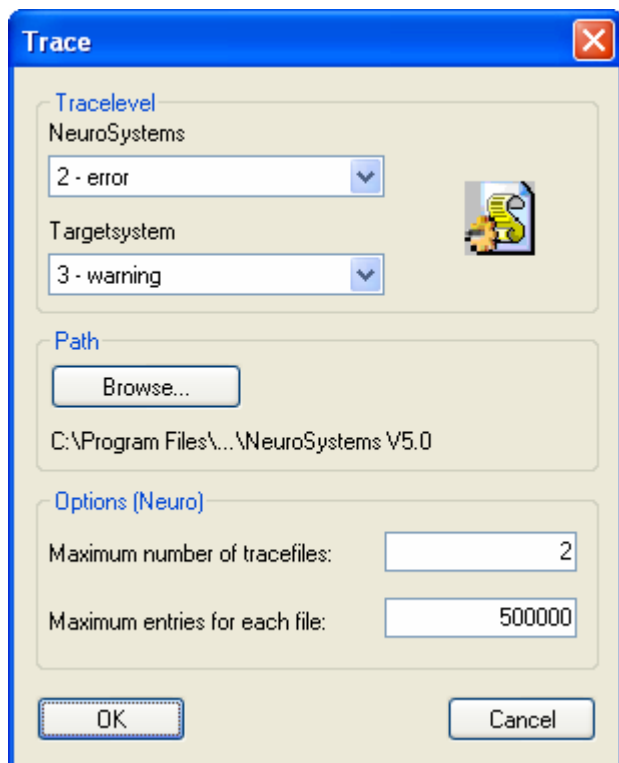
5.1 Trace

Shortcut: Hotkeys <STRG+SHIFT, T>

After setting the trace level to unequal 0 you can also select the trace dialog via the tool bar.

Shortcut: toolbar LMB on the icon  Trace.ico

If both trace levels are set to 0, than you can not use the toolbar for selection.

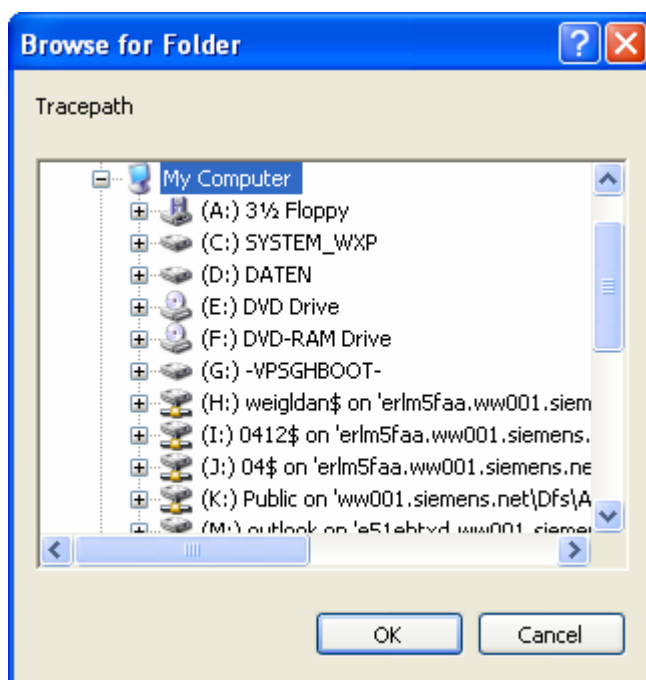


For *NEUROSYSTEMS* and the targetsystems you have the possibility to set different trace levels individually. You can choose from the following trace levels:



- 0 off
- 1 exception
- 2 error
- 3 warning
- 4 informationen
- 5 debug
- 6 operation

You can select the place where the trace files should be saved, by clicking on *Browse*.



Further you can determine the number of trace files and the maximum number of entries within a trace file.



5.2 Spezifikation of the SIEMENS-Neuro-Language (SNL)

5.2.1 Introduction

The **SIEMENS-Neuro-Language** was defined to have a unitary syntax for the description of various types of neural networks. Further it is used to keep the project files that have been generated by *NEUROSYSTEMS*, readable.

5.2.2 Spezifikation

The SNL-file is divided in sections that are introduced with according headers. The following sections are applied:

- NET DEFINITION SECTION
- CLUSTER DEFINITION SECTION
- CONNECTOR DEFINITION SECTION
- I/O NORM DEFINITION SECTION
- UNIT DEFINITION SECTION
- CONNECTION DEFINITION SECTION



5.2.2.1 NET DEFINITION SECTION

Here the name, type and targetsystem of the network are defined and information on the networks size is given.

net name	name of the network <i>NEUROSYSTEMS</i> appoints the network with the file name without the extension *.snl.
net type	network type 0 Multilayer-Perzeptron (MLP) 1 Radial-Basis-Function (RBF) 2 Neuro-Fuzzy-Network
target_system	targetsysteem 0 S7-4K 1 S7-20K 2 WinCC-OLL 3 ActiveX 4 OPC
number_of_clusters	The number of clusters of the neural network. In <i>NEUROSYSTEMS</i> every layer consists of exactly one cluster. Further the BIAS used in the MLP and NF networks, is a cluster.
number_of_units	In <i>NEUROSYSTEMS</i> the meaning of the term unit is to be comprehended as either a neuron or the BIAS. That's why, the number of entries in the section "UNIT DEFINITION SECTION" is indicated, that have to be increased by 1 in MLP- and NF networks, due to the BIAS.
number_of_connectors	The number of connectors is corresponding to the number of entries in the section "CONNECTOR DEFINITION SECTION"
nu_of_connector_types	The number of different connector types. All of the connectors used in the network are defined with the designation of their type, in the section "CONNECTOR DEFINITION SECTION".
number_of_connections	The number of connections is according to the number of entries in "CONNECTION DEFINITION SECTION"



5.2.2.2 CLUSTER DEFINITION SECTION

Here is where clusters are defined that consist of either one layer or the BIAS in *NEUROSYSTEMS*.

Syntax extraction:

lay	type	no of units	trans_func	norm	cluster name
i 1	STD	3	None	None	no Name

lay	The description of the layer that represents the cluster.	
	i1	input layer
	o1	output layer
	hx	hidden layer x
type	Type of the cluster	
	STD	Standard cluster
	RBF	Cluster with RBF
no of units	Number of knots within the mentioned cluster. A knotAis either a neuron or the BIAS.	
trans_func	Transfer function of the neurons in the respectively cluster.	
	none	no tranfer function or identity
	id	sum function (linear neuron)
	minimum	minimum function
	product	product function
	logistic	sigmoidial function
	tanh	tangens hyperbolicus
norm	standardisation function of the neurons in the respectively cluster.	
	None	no standardisation
	L1	= $\text{sum}(w_i \ x_i) / \text{sum}(x_i)$ with
		w _i : input value of the synapse I x _i : emphasis of the synapse i
cluster name	cluster name	
	NEUROSYSTEMS calls every cluster „no name“ and doesn't analyze the name.	



5.2.2.3 CONNECTOR DEFINITION SECTION

The connectors connect the knots that are setted in various clusters.

Syntax extraction:

target	source	type	learn	connector name
h1	b	STD	Y	no name

target The description of the cluster that the connector is showing at.
In addition to all layers (o1, i1, hx) it is allowed to indicate the BIAS.

o1 output layer
hx hidden layer x
B BIAS

source The description of the cluster from which the connector is showing away from.

i1 input layer
hx hidden layer x
B BIAS

type Type of connector
STD Standard connector

Learn Learning ability between two clusters.
Y Weights may be modified
N Weights may not be modified

connector name connector name
NEUROSYSTEMS calls every cluster „no name“ and doesn't analyze the name.



5.2.2.4 I/O NORM DEFINITION SECTION

All input and output values are scaled with a individual adjustable linear function $y = scale * x + shift$.

Syntax extraction:

I/O	no.	shift	scale	name
i 1	0	84.887001037598	34.792999267578	v

I/O	Cluster that includes the in- and/or output
no.	Number of the in- and/or output
shift	Coefficient of the scaling function
scale	Coefficient of the scaling function
name	Name of input and/or output

5.2.2.5 UNIT DEFINITION SECTION

Each neuron is understood as a unit.

Syntax extraction:

lay	no.	no of inputs	weight
i 1	0	0	

lay	Description of the layer that includes the neuron
i1	input layer
o1	output layer
hx	hidden layer x
no.	number of the neuron within the layer
no of inputs	number of inputs of the neuron
weight	emphasis of the neuron
	In <i>NEUROSYSTEMS</i> no neuron emphasis is used. That's why no value is indicated.



5.2.2.6 CONNECTION DEFINITION SECTION

A emphasized connection can exist between the neurons.

Syntax extraction:

l a y e r		u n i t		w e i g h t
target	source	target	source	
h1	b	0	0	-1.446825265884

target layer	layer of the target neuron
target unit	number of the target neuron within the layer
source layer	layer of the source neuron
source unit	number of the source neuron within the layer
weight	weight of the connection as a floating-point number with a dot as a decimal divider

5.2.2.7 Knotentypen und Gewichte

The following knot types and weights appear within *NEUROSYSTEMS*:

Knot types:

KE	helping knot for the input
KN	sigmoid-neuron
KT	tangens hyperbolicus neuron
KL	linear neuron
KP	product-neuron
KR	gauss-neuron
KW	standardised sum-neuron

Weight type:

WN	normal, learnable weight
WF	set, non learnable value
W1	learnable offset
WO	set offset; always '1'



5.2.2.8 Comments

Comments can be entered into the syntax of the SNL file in the programming language C. Comments start with the string `/*` and end with the string `*/`. They may not be boxed and must stand in the beginning of or at the end of the SNL file.

5.2.3 Examples

```
/* SNL-File NEUROSYSTEMS */
```

NET DEFINITION SECTION :

```
net name           : Example
net type           : 0 (MLP)
target_system      : 3 (S7-20K)
number_of_clusters : 5 (including bias)
number_of_units    : 15
number_of_connectors : 6
nu_of_connector_types : 1
number_of_connections : 56
```

CLUSTER DEFINITION SECTION :

lay	type	no of units	trans_func	norm	cluster name
i 1	STD	3	None	None	no Name
h1	STD	5	tanh	None	no Name
h2	STD	5	tanh	None	no Name
o1	STD	1	i d	None	no Name

CONNECTOR DEFINITION SECTION :

target	source	type	learn	connector name
h1	b	STD	Y	no name
h1	i 1	STD	Y	no name
h2	b	STD	Y	no name
h2	h1	STD	Y	no name
o1	b	STD	Y	no name
o1	h2	STD	Y	no name

I/O NORM DEFINITION SECTION :

I/O	no.	shi ft	scal e	name
i 1	0	84.888450622559	34.793647766113	Ei ngang01



i 1	1	165. 516969096680	134. 463394165039	Ei ngang02
i 1	2	24. 970249176025	14. 967449188232	Ei ngang03
o1	0	587. 838745117188	571. 417724609375	Ausgang01

UNIT DEFINITION SECTION :

lay	no.	no of inputs	wei ght

i 1	0	0	
i 1	1	0	
i 1	2	0	
h1	0	4	
h1	1	4	
h1	2	4	
h1	3	4	
h1	4	4	
h2	0	6	
h2	1	6	
h2	2	6	
h2	3	6	
h2	4	6	
o1	0	6	

CONNECTION DEFINITION SECTION :

layer		unit		
target	source	target	source	wei ght

h1	b	0	0	0. 185244306922
h1	i 1	0	2	-0. 184017449617
h1	i 1	0	1	0. 038151189685
h1	i 1	0	0	-0. 299249231815
h1	b	1	0	0. 237577438354
h1	i 1	1	2	-0. 089825130999
h1	i 1	1	1	-0. 012076173909
h1	i 1	1	0	0. 051005583256
h1	b	2	0	0. 215366065502
h1	i 1	2	2	-0. 195535138249
h1	i 1	2	1	0. 147962883115
h1	i 1	2	0	0. 193704038858
h1	b	3	0	-0. 291009247303
h1	i 1	3	2	-0. 117603078485
h1	i 1	3	1	0. 008120975457
h1	i 1	3	0	0. 126300856471
h1	b	4	0	-0. 200460836291
h1	i 1	4	2	-0. 211612299085
h1	i 1	4	1	-0. 081328779459
h1	i 1	4	0	-0. 245158240199



h2	b	0	0	-0.073271892965
h2	h1	0	4	-0.294653147459
h2	h1	0	3	-0.297198414803
h2	h1	0	2	-0.228550061584
h2	h1	0	1	-0.032584611326
h2	h1	0	0	0.293115019798
h2	b	1	0	0.097827084363
h2	h1	1	4	-0.200259402394
h2	h1	1	3	0.064299449325
h2	h1	1	2	0.061058383435
h2	h1	1	1	0.042710654438
h2	h1	1	0	0.018997771665
h2	b	2	0	0.181563764811
h2	h1	2	4	0.169991150498
h2	h1	2	3	0.064610734582
h2	h1	2	2	-0.265776544809
h2	h1	2	1	-0.088726460934
h2	h1	2	0	-0.029526658356
h2	b	3	0	0.255430757999
h2	h1	3	4	0.273540467024
h2	h1	3	3	0.136005744338
h2	h1	3	2	0.225583672523
h2	h1	3	1	-0.118829920888
h2	h1	3	0	0.011929685250
h2	b	4	0	-0.174239322543
h2	h1	4	4	0.217343673110
h2	h1	4	3	-0.158803060651
h2	h1	4	2	-0.022751549259
h2	h1	4	1	-0.214597001672
h2	h1	4	0	0.023612171412
o1	b	0	0	-0.064537495375
o1	h2	0	4	0.066899627447
o1	h2	0	3	0.299816876650
o1	h2	0	2	0.298077344894
o1	h2	0	1	0.206192210317
o1	h2	0	0	0.167793810368



5.3 Example Coat Thickness Control

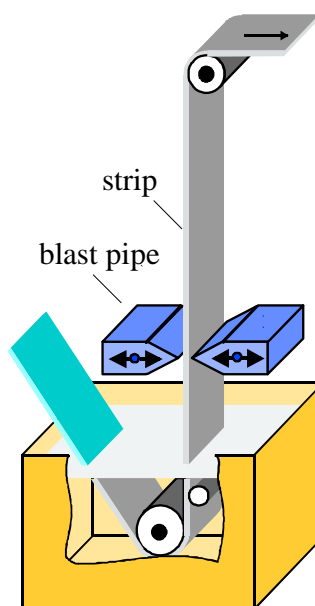
Example for modeling nonlinear functions: Control of the coat thickness in fire coating plants

It is the intention of this distinct example to give you a quick and easy introduction to your work with *NEUROSYSTEMS*. The basic problem of this example can be solved with the help of neural networks and has already been successfully applied to practice.

Example is *NEUROSYSTEMS* V1.

The problem:

The aim of the desired control system is the exact matching of the coating thickness, for example for a zinc coating of steel strips as shown in figure 1. If the zinc thickness falls below a strict tolerance limit, the produced material will be devaluated. On the other hand, an unnecessarily thick coating is not economical. The problems involved with a conventional control design are mainly operating point-dependent effects and influences of the controlled system, which can not be completely described mathematically.





The **coating thickness c** is fundamentally influenced by three factors:

- **strip velocity v**
- **air pressure p**
- **gap a** between the blast pipe and the strip

With the help of a neural learning method it is possible to run a process identification of the controlled system, where the behavior of the controlled system can be imaged to the neural network, even at varying working points. The structure of the neural network can be derived from the principal character of the relationship $c=f(v,p,a)$ and considers the physical process of the coating action. By having a sufficiently large number of measuring value sets v , p , a , and c available you can use them as training data for the neural network. From the trained network you are able to generate inverse models of the controlled system. These models can be used as "neuro controllers".

Designing the neural network using *NEUROSYSTEMS*:

This section is a step by step explanation of how to use NeuroSystems for creating a neural network which will show a desired input/output behavior.

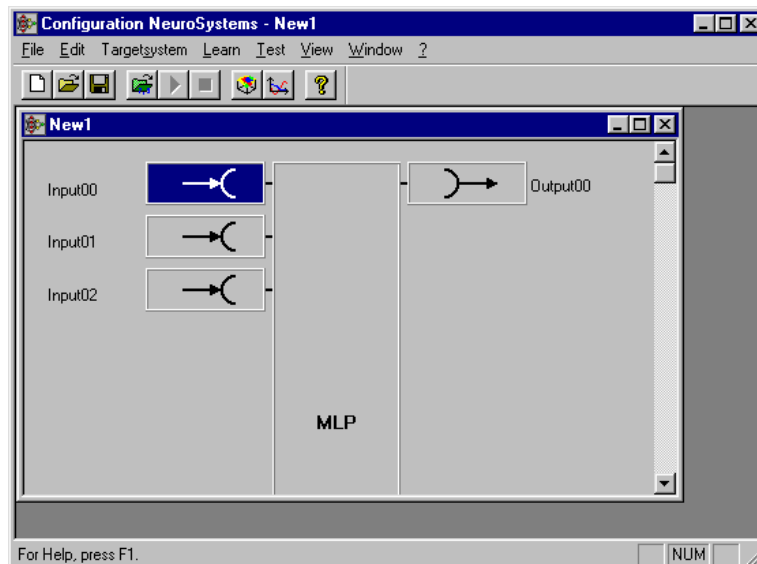
The file "coat.dat" has to be used for the training of the network. This file contains the input/output behavior to be learned. This information is represented by data sets showing the relationship $c=f(v,p,a)$. It is an ASCII-file containing four columns which are separated by spaces and/or Tabs. The first three columns represent the input variables v , p , and a . The fourth column lists the output values c . This file contains 500 data sets.

118.3981	52.5576	13.7877	6.6953590e+002
76.8565	74.9188	10.3024	2.7371804e+002
66.8556	59.8673	22.3652	2.6702663e+002
51.3384	79.1624	21.0885	1.3775896e+002
75.3120	102.3377	14.6197	1.5429955e+002
79.3079	269.7912	13.3048	3.8929054e+001
107.4508	180.6029	23.0264	8.1029074e+001
115.8322	232.6550	34.1137	5.0593370e+001
72.0223	272.9676	18.6191	3.1493492e+001
88.0730	106.6111	30.6006	1.3033576e+002
73.3648	273.2548	15.0566	3.4169096e+001
112.7599	74.1176	36.2257	2.6612936e+002
95.1959	94.6565	25.7509	1.8114071e+002



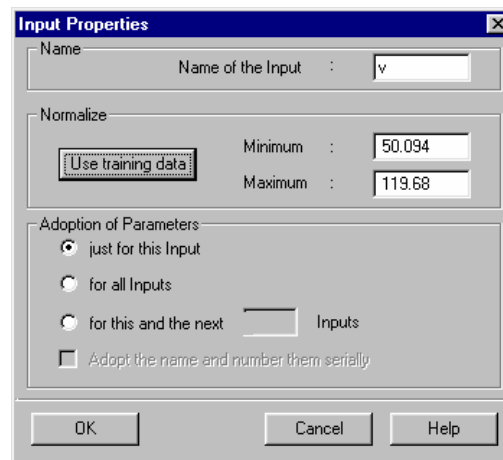
5.3.1 Creating a new project

You can create a new project by selecting the *NEUROSYSTEMS* menu item **"File/New"**. Alternatively, you may use the "New" symbol from the toolbar. This opens the "New..." window where you have to enter the number of inputs and outputs and the target system you intend to use. For this example, the function to be modeled has to represent the coating thickness c as a function of the three variables v , p , and a . For that, you have to enter **three inputs** and **one output**. You can accept the default configuration for the target system, because this chapter will only describe the training process using *NEUROSYSTEMS*. After clicking the **"OK"** button the network block and the blocks for the inputs and the output will be displayed in a new window.



5.3.2 Naming and normalizing the inputs and outputs

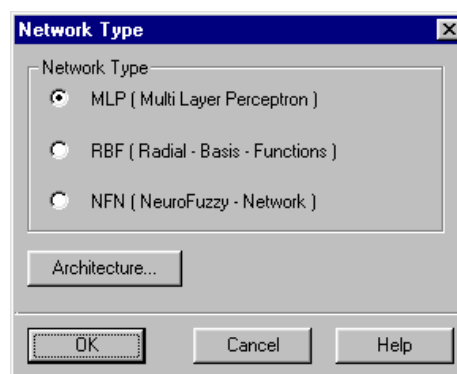
Click on the block **"Input00"** with your right mouse button and select the item "Properties..." or double-click it with your left mouse button. This will open the window "Input Properties". Enter **"v"** for the name of this input and click on the button **"Use training data"** located in the "Normalize" section. Since you have not assigned suitable training data to your network, the window "Select learning file" will be opened. Click the **"Browse..."** button and select the ASCII-file **"coat.dat"** which has been described above. Confirm your choice by clicking the **"OK"** button. The normalization will now be executed. The automatic entry at "Minimum" is the smallest value of the numerical data contained in the first column of "coat.dat", the entry at "Maximum" is the biggest. Adopt these parameters just for this input.



Name the other two inputs **"p"** and **"a"** and the output **"c"** in the same way as described above. Press the **"Use training data"** button in the "Normalize" section of all three variables, in order to carry out the normalization using the minimum and maximum values of the last three columns of "coat.dat". Since the learning file was already chosen when editing the first input, the normalization is now carried out without the need to select a *.dat-file.

5.3.3 Selecting a suitable network type and structure

Click on the network block **"MLP"** with your right mouse button or double-click it with your left mouse button. By selecting the item "Network Type..." the corresponding window will be opened. Now you can choose one of the three implemented network types and adapt the network architecture for your needs. Here, the default setting is a MLP- network with two hidden layers, each of them containing five neurons. We keep this default setting for this example.



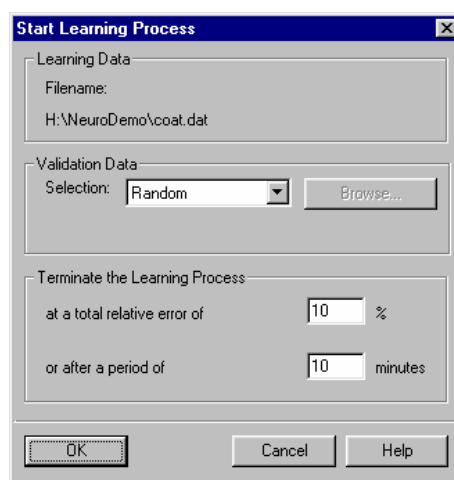


5.3.4 Selecting the learning file

Via the menu item "learning/file selection" a window opens. After clicking on "browse" choose a described ASCII file "coat.dat" and confirm your choice with "OK". Afterwards a window appears with the question whether you would like to determine the range of the in- and outputs by means of the chosen learning file. By clicking on "yes" the window "range" opens, in which you can assign the method of determination. In this example please choose "lower and upper limit with outlier" and confirm with "OK". For the lower and upper limit the smallest or biggest value, that has been noted in the learning file for the respectively in- or output, will be assigned. You can see this in the properties of the in- and output and if necessary can change it manually.

5.3.5 The learning process

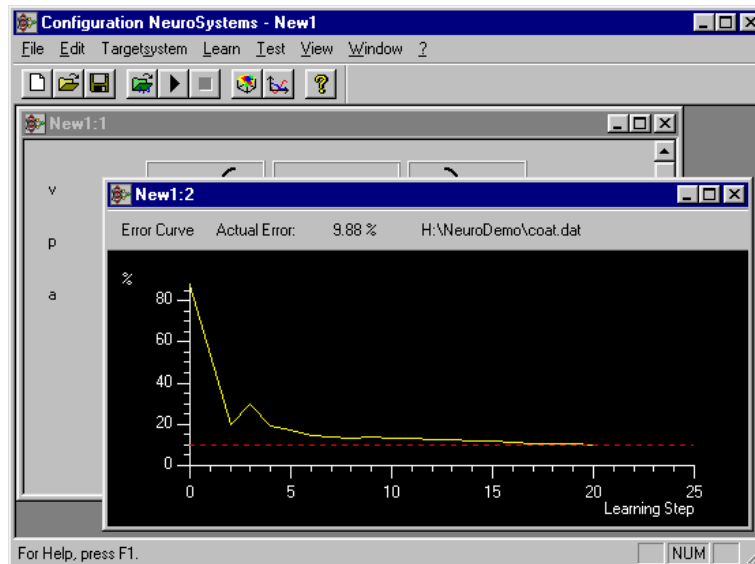
The "Start Learning Process" window is opened by selecting the *NEUROSYSTEMS* menu item "**Learn/Start**". Alternatively, you may use the "Start" symbol from the toolbar. The "**random**" selection of validation data can be kept, as the learning data file "coat.dat" contains a lot of (partly redundant) data sets, from which a random selection of 40% can be used for validation without losing too much information. The criteria for terminating the learning process can be kept at the 10% error limit and the 10 minute time limit, respectively. If there is the need to improve the training results to reach a lower error limit, the already trained network can later be optimized with "Learn/Continue" at any time.



After clicking the "OK" button a window will be opened which displays an error curve over the number of learning steps performed by *NEUROSYSTEMS*. If the current error falls below the



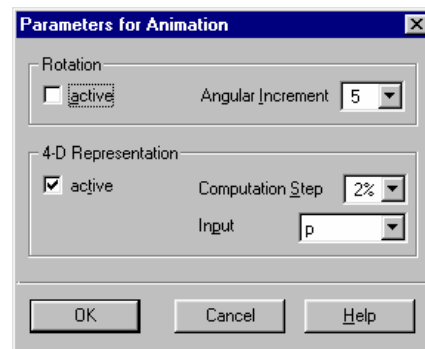
dotted red 10% limit (which will be after just a few learning steps), the learning process will be stopped und a message will be displayed, asking you whether to accept the trained network. Click the **"Yes"** button to accept the network in its current state.



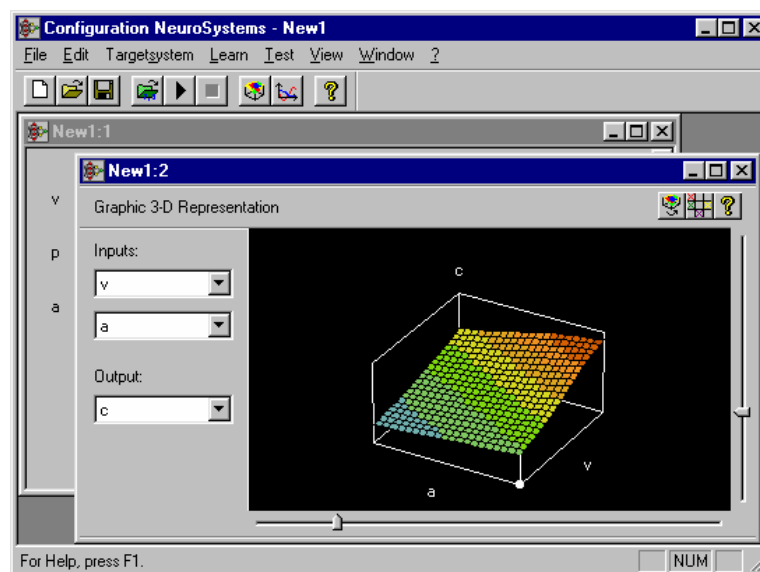
For further improvement of the network, choose the menu item **"Learn/Continue"** and reduce the error limit to 1%, for example. After your confirmation by pressing the **"OK"** button, the learning process will be continued, re-starting at the current 10% error level. Now it may take some time until the error falls to the new 1%-value. If you do not want to wait for this goal to be reached, you can interrupt this process at any time by choosing the menu item **"Learn/Stop"** or clicking the "stop" button on the toolbar, so you can accept the network with the current error.

5.3.6 Visualizing the network behavior

The **"Test"** menu offers some tools for the visualization of the networks' input/output behavior and for testing the grade of the learning process. The option **"3-D Graphics"** allows you to illustrate the input/output behavior very easily. To get an overview of the characteristic curves of the function you just modeled above, you can choose the variables **v** and **a** for the input axes to be displayed and click the "animation" button for activating the 4-D representation, selecting the pressure **p** as a parameter, for example.

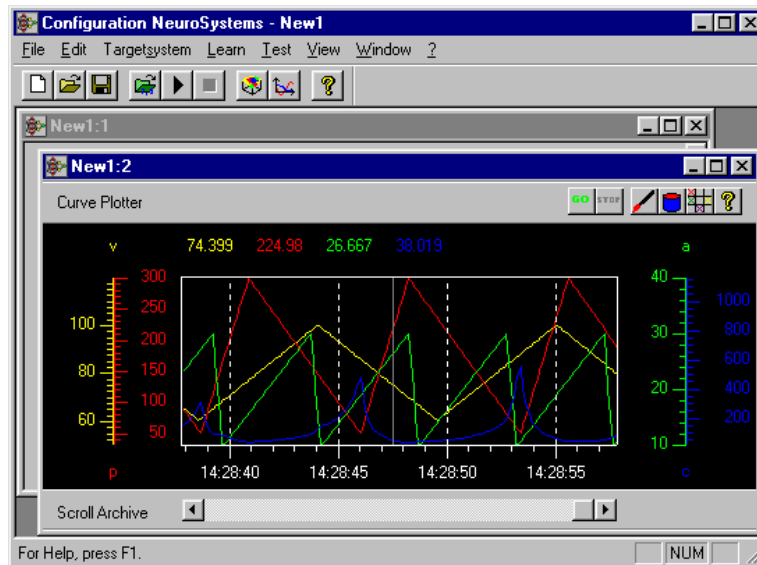


The animation shows very vividly how the characteristic of the coat thickness c varies with the pressure p . By pressing the "animation" button another time you can stop the animated graphic display. The current value of p will then be kept as input parameter of the 3-D representation.



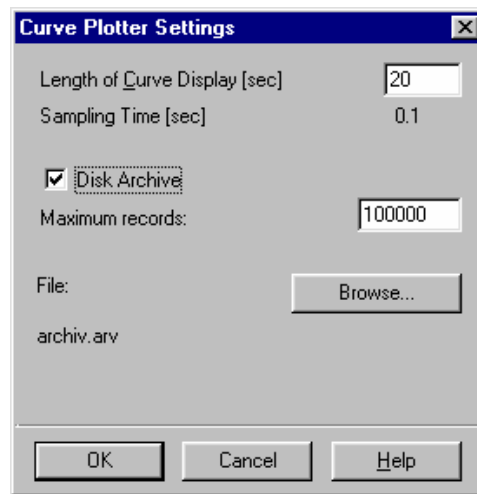
There is even the possibility of opening the 3-D graphics window before you start the learning process so that you can visualize the learning progress by directly viewing the changes of the characteristic curves.

The curve plotter offers another way of visualization. Select "**Test/Curve Plotter**" for starting it. In the offline-mode you can assign an individually configurable triangle signal to all of the inputs just by clicking the "curve generator" button. Pressing the "**GO**" button will start the recording session.



The current values of the four recorded variables are permanently displayed in the corresponding curve color. The horizontal axis will be automatically labeled with the current time. After you stop the curve plotter by clicking on the "**STOP**" button a vertical "reading line" will be shown in the diagram. You can click on the line and drag it over the whole visible diagram region. By doing so, the curve values which belong to the reading line position will be shown.

If you want to store this data in a hard disk archive, you have to activate the button "**Archive Settings**" before starting the curve plotter and select the option "**Disk Archive**". You can enter the maximum number of records and the name of the archive file. The default name setting is "**archiv.arv**".



5.3.7 Saving the project

After designing, adapting, training, and testing the neural network, you can now save your project in an *.snl project file (e.g. "coat.snl") using **"File/Save"** or **"File/Save As"**.



5.4 References

- [1] C. von Altrock: Fuzzy Logic, Band 1: Technologie; R. Oldenbourg Verlag; München, Wien; 1993
- [2] M. Black: Vaguess: An Exercise in Logical Analysis; Philosophy of Science 50; S.427-455; 1937
- [3] R. Brause: Neuronale Netze: Eine Einführung in die Neuroinformatik; B. G. Teubner Verlag; Stuttgart; 1995
- [4] A. Cichocki, R. Unbehauen: Neural Networks for Optimization and Signal Processing; B. G. Teubner Verlag; Stuttgart; 1993
- [5] W. S. McCulloch, W. Pitts: A logical calculus of the ideas immanent in nervous activity; Bulletin of Mathematical Biophysics 5; S.115-133; 1943
- [6] D. Driankov, H. Hellendorn, M. Reinfrank: An Introduction to Fuzzy Control; Springer-Verlag; Berlin, Heidelberg; 1993
- [7] K. A. Flaton: Neuronale Netze: Grundlagen und Anwendungen; Reihe: Neuro+Fuzzy; Zimmermann (Hrsg.); VDI-Verlag GmbH; Düsseldorf; 1995
- [8] S. Hafner: Neuronale Netze in der Automatisierungstechnik; R. Oldenbourg Verlag, München, Wien; 1994
- [9] D. O. Hebb: The organization of behavior; John Wiley; New York; 1949
- [10] N. Hoffmann: Kleines Handbuch Neuronale Netze; Vieweg & Sohn Verlag; Braunschweig; 1993
- [11] J. Kahlert: Fuzzy Control für Ingenieure: Analyse, Synthese und Optimierung von Fuzzy-Regelungssystemen; Vieweg & Sohn Verlag GmbH; Braunschweig; 1995
- [12] H. Kiendl: Fuzzy Control methodenorientiert; R. Oldenbourg Verlag; München, Wien; 1997
- [13] G. J. Klir, B. Yuan: Fuzzy Sets and Fuzzy Logic: Theory and applications; Prentice Hall Inc.; New Jersey; 1995
- [14] M. Koch, Th. Kuhn, J. Wernstedt: Fuzzy Control: Optimale Nachbildung und Entwurf optimaler Entscheidungen; R. Oldenbourg Verlag; München, Wien; 1996
- [15] T. Kohonen: Self-organized formation of topologically correct feature maps; Biological Cybernetics; S. 59-69; 1972



- [16] B. Kosko: Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence; Prentice Hall, Inc.; New Jersey; 1992
- [17] J. Lukasiewicz, A. Tarski: Untersuchungen über den Aussagenkalkül; Comptes Rendus Soc. Sci. et Lettr. Varsovie III, 23 ; S. 30-50; 1932
- [18] D. E. Rumelhart, G. E. Hinton, R. J. Williams: Learning internal representations by error propagation; Rumelhart/McClelland (Hrsg.), Parallel distributed, Processing: Explorations in the Microstructure of Cognition 1; MIT Press; S. 318-362; 1986
- [19] Siemens AG: Grundlagen der Fuzzy-Logik: Handbuch *FuzzyControl* für SIMATIC S7; S. 1-2; 1995
- [20] Siemens AG Automatisierungstechnik: Fuzzy-Systeme in Theorie und Anwendungen Fuzzy aus den Erfahrungen der Siemens AG; Nürnberg; 1997
- [21] M. Sugeno: Industrial Applications of Fuzzy Control; North-Holland-Amsterdam; London; 1985
- [22] M. Sugeno: An Introductory Survey of Fuzzy Control; Information Sciences 36; S. 59-83; 1985
- [23] T. Terano, K. Asai, M. Sugeno: Applied Fuzzy Systems; AP Prof.; Boston; 1994
- [24] Vdi; VDE-Gesellschaft Meß- und Automatisierungstechnik: Neuronale Netze: Anwendungen in der Automatisierungstechnik ; VDI Berichte 1184; VDI-Verlag GmbH; Düsseldorf; 1995
- [25] P. J. Werbos: Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences; Diss.; Harvard University; Cambridge; 1974
- [26] T. Wolf, Siemens AG: Optimization of Fuzzy Systems using Neural Networks and Genetic Algorithms; Proceedings of the 2nd European Congress of Intelligent Techniques and Soft Computing EUFIT; S.544-551, Aachen; 1994
- [27] L. Zadeh: Fuzzy Sets; Information and Control 8; S.338-353; 1965
- [28] A. Zell: Simulation neuronaler Netze; Addison-Wesley Publishing Company; Bonn; 1994
- [29] H.-J. Zimmermann: Fuzzy Technologien: Prinzipien, Werkzeuge, Potentiale; VDI-Verlag GmbH; Düsseldorf; S. 203-218; 1993



5.5 Index

3

3-D Graphics	
Animation	193
Display limits	193
Rotation	194
Setting the parameters	195

A

About NeuroSystems	223
Activation function	13, 17
ActiveX	146
ActiveXControl	83
Error Mode	89
Events	86
Example	101
Graphical Interface	88
Limitations	125
No NeuroDLL Mode	89
Normal Mode	88
path file (.snl)	85
Properties	84, 87
Trigger	85
WinCC	93
WinCC flex	115
ActiveXControl Installation	83
Actual curve	65
Adaptation of input signal	205
Additional parameters	74
Advantages and disadvantages of fuzzy logic	23
Advantages and disadvantages of neural networks	11
Animation of the 4-D graphic	70
Applications of fuzzy logic	22
Applications of neural networks	10
Archive Analysis	203
ASCII file	207
Aspect ratio	205

B

Backpropagation algorithm	10, 20, 189
Binary assignment	21
Block Diagram	130
Block Diagram and Parameters of the FBs	73
Block Structure	72



Block Symbols	130
---------------	-----

C

Calling the Function Blocks	75
Changing the learning data file	67
Characteristic surface	64, 192
Component structure	72
Configuration of the network	49
Configuration tool	41
Creating a Network	48
Curve Generator	204, 205
Curve plotter	
Archive Analysis	203
Archiving data	202
Curve colour	199
Curve Generator	203, 204
Curve scaling	200
Curve set	199
Curve settings	197
Disk archive	202
Online/Offline	206
Parameterization of curves	203
Sampling time	198, 201
Scaling	198
Starting recording	201
Value fields	200

D

Data block (DB)	72, 154
DB 100	72
DB 101	72
DCOM	126
Settings	127
Settings OPC Server	129
Settings Windows 2K	128
Settings Windows XP	128
Defuzzification	25, 26, 27, 30
Deleting inputs/outputs	138
Determining the range	53
Dialog box <i>New...</i>	132
Disk archive	202
Distribution	
Average	175
Bar information	176
Median	175
Standard deviation	175
Drift recognition	185
Driver info	148
Dynamic network	15
Dynamic sampling	160



E

Editing Actions for a Neuro Object	80
Editing inputs/outputs	138
Editing the inputs and outputs	51
Editing the learning data file	66
Editing the Network Type	141
Error calculation	186, 209
Error Curve	62
Error Development	186, 190
Error diagram	207
Error Summary	216
Evaluating the ERROR Parameter	77
Evaluating the learning process	68
Example C-Action for Neuro Object	81
Example of coat thickness	235
Execution control	76
Execution times	79
Export fpl file	144
External setting of the I/O	76

F

FB 100	72
FB 101	72
File	
1, 2, 3, 4	136
Close	134, 135
Exit	136
Import / Export	136
New	132
Open	134
Save	135
Save as ...	135
File with validation data	185
Following neuron	16
fpl file	136, 144
Function block (FB)	72, 154
Function of a fuzzy system	25
Fundamentals of fuzzy logic	22
Fundamentals of neural networks	10
Fuzzification	25, 27
Fuzzy Control	24
Fuzzy logic	21
Fuzzy sets	27
Fuzzy.OLL	159, 162

G

Gaussian function	17
Graphic representation	61
Graphic representation (3-D)	191



Graphic representation (4-D)	194
Graphics Designer	159

H

Hardware-Requirements	41
Hidden layer	15
History	
Fuzzy logic	21
Neural networks	9
How neural networks work	19

I

Icons and functions of the toolbar	218
Inference	25, 28
Influence using the Configuration Tool	76
Input parameters	73
Input/Output characteristic	12, 13, 14, 15, 16, 19, 20, 48, 68, 185, 191, 194, 204, 211, 213
Inserting inputs/outputs	137
Installation	41, 42
Installation Language	45
Installing the WinCC-OLL	80
Instance DB	75
Interpolation	61
Interpolation surface	65

L

Learn	
Continue	188
File Selection	170
Fine Tuning	189
Start	184
Stop	187
Learning data file	66
Learning data set	59, 66
Learning process	59
Learning supervised	20
Library Contents	72
Linguistic variable	25
LMB	132

M

MAXIMUM operation	29
MAX-MIN inference	27
MAX-PROD inference	27
Membership function	24, 25, 26, 27, 28, 29, 30, 55, 56, 143
Menu	
Combinaton	179
Curve plotter	196
Deleting targetsystem components	149



Display limits	193
Distribution	174
Edit	137
File	132
Help	221
Info version	223
Input relevancy	182
Installed targetsystems	148
Installing targetsystem components	149
Learn	170
Manager	147
Range	171
Targetsystem	146
Test	191
Value fields	200
View	218
Window	220
Miscellaneous	224
MLP	55
MLP network	15, 17, 50, 55, 56, 142
Multilayer perceptron	15

N

Network	
Defining the structure	56
Properties	145
structure	49
Structure	145
Type	145
Neural networks	9
Neurofuzzy systems	30
Neuron	9, 11, 12, 14, 15, 16, 17, 18, 19, 20, 56, 57, 58, 59, 142, 143
New in V5.0	46
NFN (neurofuzzy network)	55, 143
Non-equivalence function	48
Non-fuzzy sets and fuzzy sets	22
Non-fuzzy value	30
Number of inputs and outputs	132
Number of layers	16
Number of network layers	57

O

Online help	221
OPC	126, 146
Basics	126
Operating structure of NeuroSystems	130
Output parameters	74

P

Parallel Project Working	131
--------------------------	-----



Parameter designations	195
Parameter values	195
Preceding neuron	16
Project	48, 130
Project window	50

Q

Quick instructions	48
--------------------	----

R

Radial basis function	10
Random validation data	185
RBF network	17, 55, 143
RMB	132
Rule base	25, 26
Runtime module	41, 71, 146

S

Sampling Values	61
Selecting the type of network	55
Selection of the network structure	56
Sensitive analysis input	182
SIEMENS Neuro-Language	226
Sigmoid function	13
Signal representation	213
SIMATIC Function Blocks	71
SIMATIC S7	146
SIMATIC WinCC	80
Simple Neuro Example	82
Singleton	29
Smart-Objekt	159
snl (SIEMENS NEURO LANGUAGE)	48
Software Requirements	41
Spatial representation	191
Specified curve	65
Start learning	62
Starting recording	201
Static network	15
Status bar	49, 219
Status Bar	218
Stop learning	186
Structure	57
Structure of the learning data file	66
System Requirements	41

T

Targetsystem	
ActiveX	162
Connect	154, 165



Connect Industrial Ethernet	156
Connect MPI/Profibus	155
Connect TCP/IP	156
Disconnect	158, 169
Limits	133, 150
Menue	146
OPC	164
Read	158, 169
Selection	149
SIMATIC S7	151
WinCC	159
Write	158, 161, 169
Test	
Error diagram	207
Error Summary	216
file selection	206
Signal representation	213
Vector representation	211
Test file	206
The learning process, starting and stopping	62
The nature of neural networks	11
to be – is – aberration	216
Toolbar	49, 218
Trace	224
Training	59
Trapezoidal membership functions	143
Type of network	55

U

Using Help	223
------------	-----

V

Validation data	62, 63, 64, 65, 67, 68, 170, 184, 185, 209
Vector representation	211
Virtual field device (VFD)	156

W

Weighted sum	12
WinCC	
Limitations	82
OLL	82
WinCC applications	80
Working window	49, 130

X

XOR function	48
xor.dat	59





NOTES