FuzEvent®

User and Programming Manual

Publish date: March 2013

FuzEvent®
User and Programming Manual

Document History

Earlier versions of the FuzEvent® User Manual
26-08-2008

Impressum

FL-Soft©

Torvet 18, 1

DK-4600 Køge

Denmark

Tel: +45 20155262

Email: jjoe@fl-soft.dk

Editors:

Jens-Jørgen Østergaard

Table of Contents

Document History 2
Impressum 2
Table of Contents 3
Table of Illustrations 7
1 FuzEvent Introduction 10
1.1 FuzEvent system structure 10
2 FuzEvent control versus PID based control 11
2.1 Why FuzEvent or AWR? 11
2.2 How FuzEvent and AWR controls a process compared to PID 11
3 FuzEventMain 12
3.1 Introduction 12
3.2 Application Login 13
3.3 Auxiliary Operator Interface 15
4 Application windows 16
4.1 Introduction to Application windows 16
4.2 Header 17
4.3 Show 17
4.4 Windows 18
4.5 Excel interface 18
5 Browser 19
5.1 Browser introduction 19
5.2 The message list 20
5.3 The EventX property list 20
5.4 EventX properties 22
5.5 The Script 27
6 The EventX component 28
6.1 EventX introduction 28
6.2 The Priority Management System 28
6.2.1 The EventX Value and the EventX Weight 30
6.3 Start EventX / Stop EventX 31
6.4 On line / Off line 32
6.5 Click on EventX name 33
7 Edit Tags 34
7.1 Tag definition 34
7.2 Modify Tag 35
7.3 New Tag 35
7.4 Delete Tag 35
7.5 Find Tag 36

7.6 Tag used in EventX ... 36
7.7 Repair Tag table ... 36
8 Edit Globals ... 36
8.1 Globals definition ... 36
8.2 Modify Globals ... 37
8.3 New Globals ... 37
8.4 Delete Globals ... 37
8.5 Find Globals ... 38
8.6 Global used in EventX ... 38
8.7 Repair Globals ... 38
8.8 Array expand ... 38
8.9 Array hide ... 38
9 Edit Locals ... 39
9.1 Locals definition ... 39
9.2 The Locals menu items ... 39
10 Edit Properties ... 40
10.1 Basics ... 40
10.2 Control modes ... 40
10.3 Miscellaneous ... 41
10.4 Activation ... 41
10.5 Deactivation ... 42
11 Edit Script ... 43
11.1 Script introduction ... 43
11.2 The Script editor ... 44
11.2.1 Known words ... 45
11.2.2 Comments ... 45
11.2.3 Errors ... 45
12 The EventX type library ... 47
12.1 EventX type algorithms ... 47
12.2 EventX type 0 (No algorithm) ... 48
12.3 EventX type 1 (General control) ... 48
12.4 EventX type 10 FECA GT and type 11 FECA LT ... 49
12.4.1 Basics ... 51
12.4.2 Control modes ... 51
12.4.3 Miscellaneous ... 52
12.4.4 Activation ... 52
12.4.5 Deactivation ... 53
12.4.6 Stepwise reverse actions ... 53
12.4.7 Fuzzy activation ... 54
12.5 FECA scripts in practice ... 55
12.6 EventX type 13 (PID controller) ... 57
12.7 EventX type 19: Raw material proportioning ... 57

13 The FUEL script language 57
13.1 FUEL introduction 57
13.2 GOTO 58
13.3 IF THEN / END IF 59
13.4 Fuzzy IF-THEN/END IF 60
13.5 Standard membership functions 61
13.6 Fuzzy control rules 62
14 The FUEL Functions 64
14.1 ABS 64
14.2 CLOSE 64
14.3 DAY 64
14.4 EVAL 64
14.5 EVENTX 65
14.6 EXECUTE_TYPE 65
14.7 FUZZY 66
14.8 FUZZYFY 67
14.9 HOUR 67
14.10 LIMIT_CHECK 67
14.11 LOG_ACTIONS 68
14.12 LPOS, MPOS, SPOS etc. 68
14.13 MINUTE 68
14.14 MONTH 68
14.15 OPEN 69
14.16 PWR 69
14.17 RETRIEVE 69
14.18 RND 69
14.19 RUN 70
14.20 SCALE 70
14.21 SECOND 71
14.22 SHIFTUP 71
14.23 SHOW_VALUE 71
14.24 START 72
14.25 STOP 72
14.26 STORE 72
14.27 TIMER 73
14.28 TREND 74
14.29 YEAR 74
15 FuzEvent used in a WtE plant, example and breakdown 75
15.1 FuzEvent application overwiew 76
15.2 EventX scripts for auxiliary functions 76
15.2.1 EventX 1: Watch dog & ACC, TEST Override 77
15.2.2 EventX 2: Bumpless 78

15.2.3 EventX 3: Trend calculations Excel (10 sec base) ... 79
15.2.4 EventX 4: Check ... 80
15.2.5 EventX 5: Filter, I/O & Resistance ... 80
15.2.6 EventX 6: Grate resistance, Grate speed LL ... 81
15.2.7 EventX 8: Flame position parameters ... 82
15.2.8 EventX 9: Data to/from iFIX ... 83
15.2.9 EventX 10: Reporting ... 83
15.3 EventX scripts for Feeder speed control ... 85
15.3.1 EventX 11: High steam flow Grate speed ... 86
15.3.2 EventX 12: Low steam flow Grate speed ... 88
15.3.3 EventX 13: Very high steam flow Grate speed ... 89
15.3.4 EventX 14: Very low steam flow Grate speed ... 90
15.3.5 EventX 15: Change of grate speed from O_2 trend. ... 90
15.3.6 EventX 18: Feeder speed kick when low Steam Flow. ... 92
15.3.7 EventX 19: Waste quality ... 93
15.3.8 EventX 20: Grate speed control ... 93
15.3.9 EventX 40: Calc. Feeder Movement ... 94
15.4 EventX scripts for Primary and Secondary air control ... 95
15.4.1 EventX 21: High steam flow Primary air ... 97
15.4.2 EventX 22: Low steam flow Primary air ... 98
15.4.3 EventX 23: Very high steam flow Primary air ... 99
15.4.4 EventX 24: Very low steam flow Primary air ... 100
15.4.5 EventX 25: Change of Primary Air from O_2 Trend ... 101
15.4.6 EventX 26: Change of Primary Air from furnace pressure ... 102
15.4.7 EventX 27: Change of Primary Air from high/low O_2 ... 103
15.4.8 EventX 29: High/Low PA flow Primary Air ... 105
15.4.9 EventX 30: Primary Air control ... 106
15.4.10 EventX 39: Secondary Air Difference control ... 107

Table of Illustrations

Fig. 1: FuzEvent system structure 10
Fig. 1: Process with long Step response time 11
Fig. 2: Starting FuzEvent 12
Fig. 3: FuzEventMain window 13
Fig. 4: Application login 14
Fig. 5: Operator control and surveillance screen 15
Fig. 6: The Application Switch Panel 16
Fig. 7: The application menu items 17
Fig. 9: Header Text & Position 17
Fig. 10: Show new EventX 18
Fig. 8: The EventX module 17
Fig. 11: Select application windows 18
Fig. 12: The Excel interface 19
Fig. 13: The FuzEvent Browser window 19
Fig. 14: Using Find in the Browser window 20
Fig. 15: EventX properties in the Browser window 21
Fig. 16: Changing EventX properties in the Browser window 22
Fig. 17: EventX Script in the Browser window 27
Fig. 18: Context menu when right-clicking EventX No. 28
Fig. 19: The EventX component 28
Fig. 20: Three situations in a priority group 29
Fig. 21: Calculation of EventValue 30
Fig. 23: Changing EventX execution. 31
Fig. 22: EventX Weight and -Value 31
Fig. 25: Change parameters in the EventX Property window 33
Fig. 24: Click on EventX name to show properties 33
Fig. 26: Context menu when right-clicking EventX No. 34
Fig. 27: Tag definition table 34
Fig. 28: Modify Tag 35
Fig. 29: New Tag 35
Fig. 30: Tag used in EventX 36
Fig. 31: Globals definition table – Array collapsed 37
Fig. 32: Globals definitions table - Array expanded. 38
Fig. 33: Locals definitions table - Array collapsed. 39
Fig. 34: FECA Properties window 40
Fig. 35: Edit Script window 43
Fig. 36: List of Globals 44
Fig. 37: Using the pop-up menu to locate variables 45

Fig. 38: EventX type 0 properties 48
Fig. 39: EventX type 1 properties 48
Fig. 40: FECA type EventX (type 10 and 11) properties 49
Fig. 41: FECA basic control function 50
Fig. 42: FECA configuration window 51
Fig. 43: FECA Stepwise reverse actions 54
Fig. 44: FECA Fuzzy activation 55
Fig. 45: FECA scripts in quad configuration 56
Fig. 46: Standard membership functions 61
Fig. 47: Table of Standard membership functions 62
Fig. 48: Fuzzy singleton creation using FUZZY(VAR) 66
Fig. 49: Fuzzy membership creation using FUZZYFY 67
Fig. 50: SCALE function 70
Fig. 51: Real time display of values using SHOW_VALUE 72
Fig. 52: TREND function 74
Fig. 53: FuzEvent operator screen for one line 75
Fig. 54: Example of a FuzEvent application 76
Fig. 55: EventX 1 properties 78
Fig. 56: EventX 2 properties 78
Fig. 57: EventX 3 properties 79
Fig. 58: Fuzzyfication of 20' average feeder speed 80
Fig. 59: Operator input fields in iFIX 81
Fig. 60: EventX 6 values & parameters 82
Fig. 61: The Feeder control EventX components 86
Fig. 62: Parameters for EventX 11 - High steam flow Grate speed 87
Fig. 63: EventX 11 active state values 87
Fig. 64: Parameters for EventX 12 - Low steam flow Grate speed 88
Fig. 65: Parameters for EventX 13 - Very high steam flow Grate speed 89
Fig. 66: Parameters for EventX 14 - Very low steam flow Grate speed 90
Fig. 67: Values and parameters in EventX 15 91
Fig. 68: O_2 fuzzyfication using the SCALE function 91
Fig. 69: Values and parameters in EventX 18 92
Fig. 70: Values and parameters for EventX 19 93
Fig. 71: Grate speed operator limits and actual values 93
Fig. 72: Values and parameters in EventX 20 94
Fig. 73: Values and parameters for EventX 40 94
Fig. 74: The Air control EventX components 96
Fig. 75: Parameters for EventX 21 - High steam flow Primary air 97
Fig. 76: Parameters for EventX 22 - Low steam flow Primary air 98
Fig. 77: Parameters for EventX 23 – Very high steam flow Primary air 99
Fig. 78: Parameters for EventX 24 – Very low steam flow Primary air 100
Fig. 79: EventX 24 partly de-activated 101

Fig. 80: Values and parameters for EventX 25 101
Fig. 81: Values and parameters for EventX 26 102
Fig. 82: Primary air fan output offset as function of low O_2 103
Fig. 83: Primary air fan output offset as function of high O_2 103
Fig. 84: Values and parameters for EventX 27 104
Fig. 85: O2 set-point and delta values for high and low O2 104
Fig. 86: Values and parameters for EventX 29 105
Fig. 87: Primary air limits and actual Primary air flow 105
Fig. 88: Values and parameters for EventX 30 106
Fig. 89: Primary Air high limit in relation to Steam set-point 106
Fig. 90: Values and parameters for EventX 39 107

Introduction to FuzEvent.

1 FuzEvent Introduction

FuzEvent is a software tool for process optimization through high level control. A FuzEvent control strategy helps the process operator to produce better operation of the process in terms of higher throughput, less consumption of materials and energy, better product quality and less emission of hazardous waste product to the environment.

FuzEvent is a dedicated tool for making control strategies rather than for making controllers.

The basic control philosophy of FuzEvent is to use knowledge about manual control of the process as the starting point for design of an automatic control strategy.

FuzEvent is an open software system, which enables the end user to maintain and further develop the control applications. The system enables on-line modifications, and the configuration of control applications does not require extensive programming knowledge. Much more important, the programming background is the process knowledge and control experience. FuzEvent, in other words, is a tool for the process specialist rather than a tool for programmers.

1.1 FuzEvent system structure

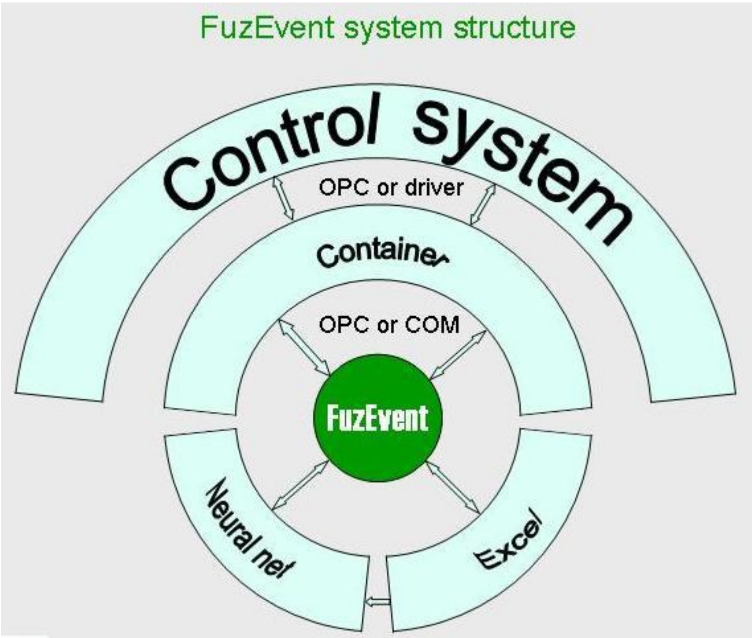


Fig. 1: FuzEvent system structure

The system structure of FuzEvent is shown in Fig. 1. FuzEvent runs in a so-called container, which normally is the existing control system.

The container communicates with the process, and data from the process is exchanged with FuzEvent through the process data base of the container. In most cases, the communication between FuzEvent and the container is done by an OPC link or a COM/DCOM type of communication. The variables in FuzEvent, which are used for communication with the container, are called Tag variables (refer to the “Edit Tags” help function).

2 FuzEvent control versus PID based control

2.1 Why FuzEvent or AWR?

Standard PID (Proportional-Integral-Derivative) controllers found in many control systems often give pure results when controlling processes with long step response times, in particular if the step response time has a large dead time component as in Fig. 2.

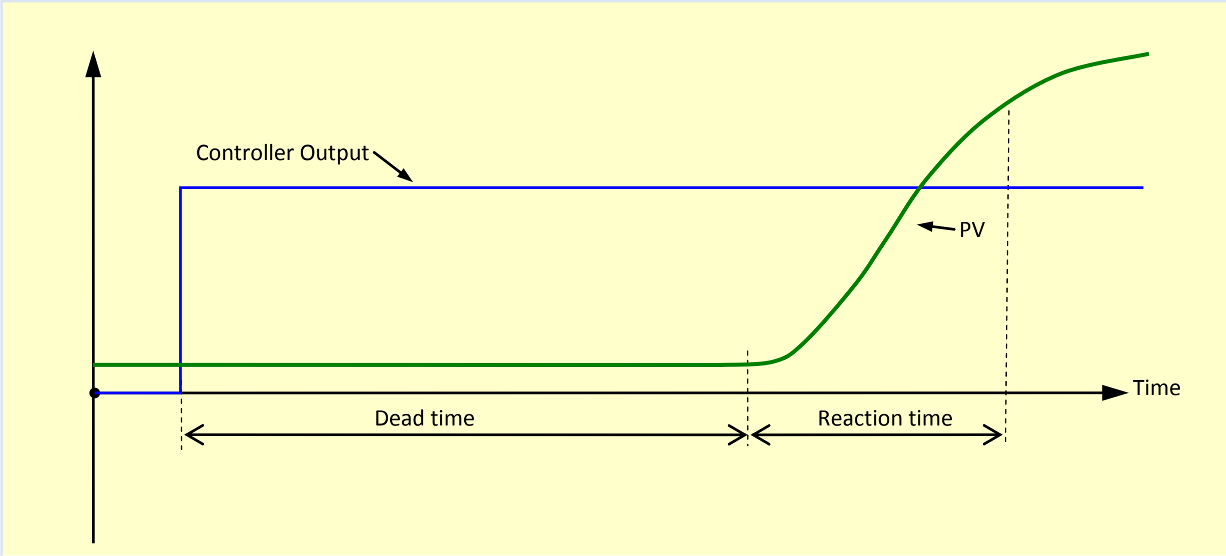


Fig. 2: Process with long Step response time

$$Step\ response\ time = Dead\ time + Reaction\ time$$

When the Output to the controlled variable is changed suddenly, it takes a very long time (Dead time) before the Process Variable (PV) reacts. Then the PV is a long time (Reaction time) to stabilize into a level corresponding to the new output level.

Such types of control loops are often found in processes involving incineration of solid fuels, like Waste to Energy plants, Biomass boilers, Coal fired rotary kilns etc. When the feeding rate is changed, the change in PV, e.g. the steam flow, lags far behind.

Using PID controllers to gain optimum control over such a process involves a whole cluster of PID controllers in a mix of cascade, feed-forward and other configurations. It also involves lots of special programming of logic and analogue processing. Due to the complexity, tuning and servicing becomes very labour intensive and time consuming and difficult to document.

Due to poor performance, it is often seen that these control loops are switched to open loop, operator controlled mode, resulting in increased workload for the plant operators and unpredictable plant performance.

2.2 How FuzEvent and AWR controls a process compared to PID

In the following the error e is defined as:

$$e = PV - SP\ or$$
$$e = SP - PV$$

Where

- PV = Process Value.

- SP = Set Point or Target or Desired value.

Output adjustment over time:

- PID controllers adjust the output continuously.
- FuzEvent / AWR adjust the output by well defined Control actions at well defined Control intervals, then waits to assess the situation.

Output adjustment when the process value deviation from the set point increases, i.e. increasing error:

- PID controllers adjust the controlled variable (output) continuously regardless of the PV – Setpoint difference (the error) size by amounts dependent on the error size.
- FuzEvent / AWR start adjusting the controlled variable (output) only when the activation limits i.e. the Acceptance band is exceeded.

Controlled variable (output) adjustment when the process value deviation from the set point decreases, decreasing error:

- PID controllers adjust the controlled variable (output) continuously regardless of the PV – Setpoint difference (the error) size.
- FuzEvent / AWR adjust the output only when the PV is beneath de-activation limit by an amount defined by the Accumulated actions and the Reverse factor.
  If Stepwise Reverse Actions is active, this amount is divided into three portions and the reduction is executed by a portion as the error decreases.

3 FuzEventMain

3.1 Introduction

FuzEvent is started by double clicking the FuzEventMain icon, shown in Fig. 3.

Fig. 3: Starting FuzEvent

After a short delay, the FuzEventMain window (Fig. 4) appears and the FuzEvent applications start loading.

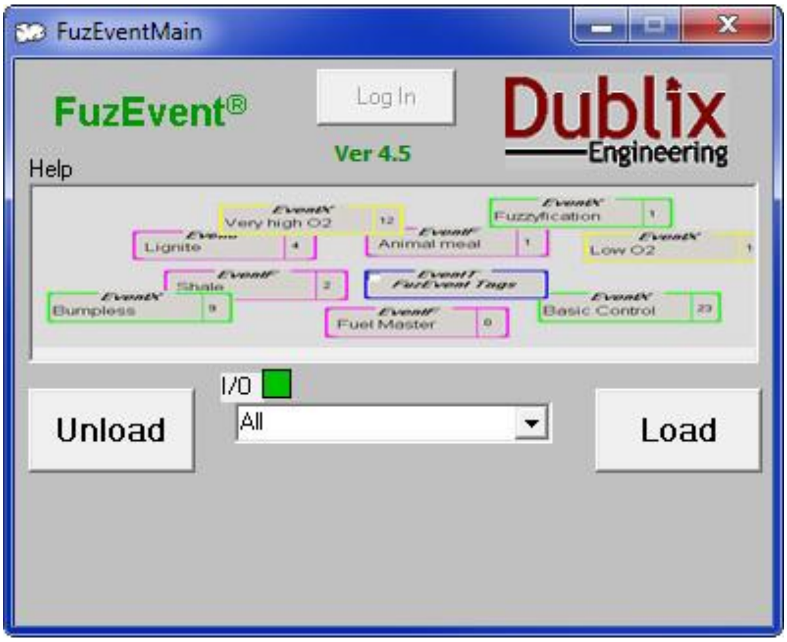


Fig. 4: FuzEventMain window

Click on the “Log In” button produces a window for specification of User name and Password. The default User name is FL-Soft, and the default Password is fuzevent

Having logged in on the FuzEventMain window actives the drop down list with the defined FuzEvent applications between the “Unload” and the “Load” buttons. One application may be loaded at a time, or all applications may be loaded at the same time. Click on one of the applications followed by click on the “Load” button to load a single application, or select All followed by click on the “Load” button to load all the defined FuzEvent applications.

The loaded applications are running when they have been loaded. It is, in other words, only necessary to load an application to activate its execution.

After loading one or all applications, FuzEvent shows the window of the last application. To work with the system, to make modifications for instance, it is necessary to log in also on individual applications, as described in the next section.

3.2 Application Login

The applications are running after they have been loaded from the FuzEventMain window. The default user for all the loaded applications is “Guest”. The “Guest” user does not have access to any functions or features in the system.

To gain access to the various functions of a FuzEvent application, it is necessary to log in at the application window.

Fig. 5: Application login

A click on the “Log In” button on the Application window shown in Fig. 5, produces a window for selection of User name and for specification of the corresponding Password. The User names, the Passwords and the corresponding privileges are agreed upon when the FuzEvent system is being installed.

3.3 Auxiliary Operator Interface.

Often FuzEvent is implemented on a separate computer and combined with a Scada / HMI / Operator Interface such as iFIX to control the most important parameters in FuzEvent, and to survey operation. An example on an operator screen picture covering a single combustion line is shown in Fig. 6.

Fig. 6: Operator control and surveillance screen

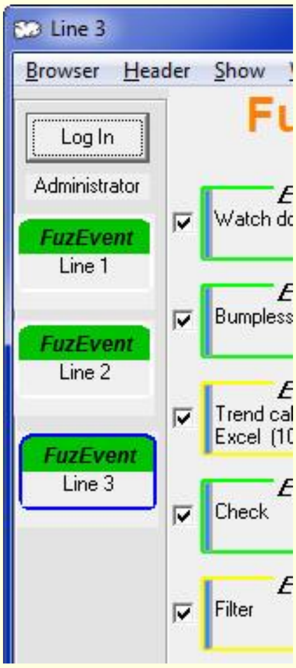
The fields with white background are operator input fields. These screen pictures are composed individually for each plant.

4 Application windows

4.1 Introduction to Application windows

The FuzEvent application window has three parts:

- The application switch panel, Fig. 7
- The application menu items, Fig. 8
- The application EventX modules



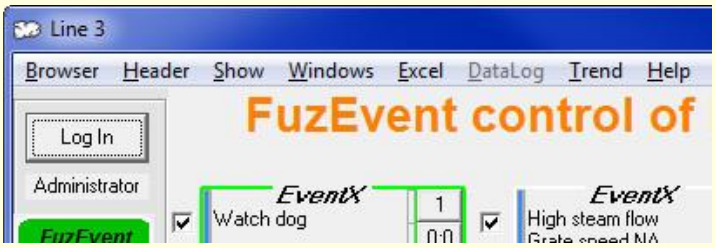
Click on ***FuzEvent*** to switch to another

Green background for ***FuzEvent*** and blue frame:

Green background for ***FuzEvent*** and white frame:

White background for ***FuzEvent*** and black frame:

Fig. 7: The Application Switch Panel



Browser: Activates the FuzEvent browser
Header: Used for modification of FuzEvent header text on the application window.
Show: Used to display and activate a new EventX.
Windows: Used to switch to another window with additional EventX components of the selected application.
Relevant when the application contains more than 49 EventX components.
Excel: Displays the Excel management window.
DataLog: Displays the data logger management window.
Trend: Displays the Trend View and the Trend Edit windows for the FuzEvent trending functions.
Help: Activates the FuzEvent Help functions

Fig. 8: The application menu items

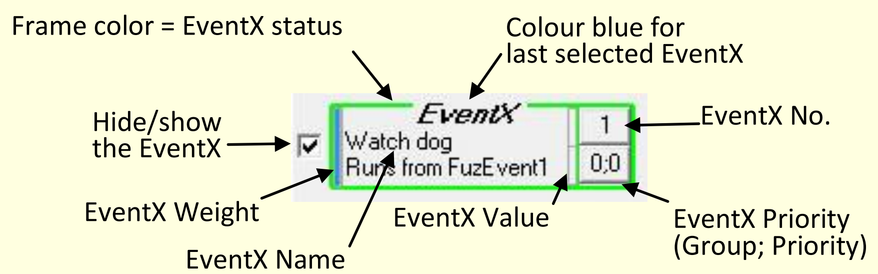


Fig. 9: The EventX module

4.2 Header

The “Header” menu item is used to specify the header text, which is shown on the application window above the EventX components.

The first step is to enter the header text, and the second step is to position the header text horizontally.

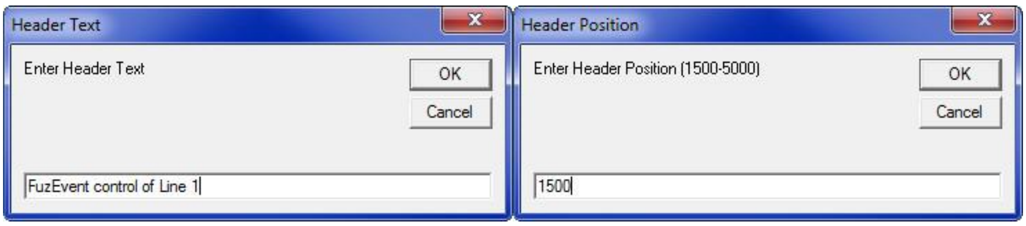


Fig. 10: Header Text & Position

4.3 Show

The “Show” menu item is used to display an EventX, which has been hidden. An application has a capacity for a certain amount of EventX components, and normally only the components, which are in use, are shown in the application window. To start working with a new EventX, the EventX must first be shown on the application window.

To show a new EventX component, click on show and enter the number of the new EventX in the input window shown below.

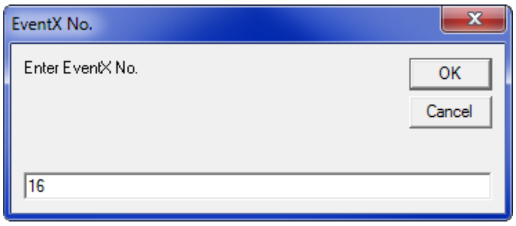


Fig. 11: Show new EventX

Note: To hide an EventX component, remove the tick mark to the left of the EventX symbol. It is, however, only possible to hide an EventX if it is not running, which means that the colour must be either white or red.

4.4 Windows

If the capacity of the application has been specified to include more than 50 EventX components, which is the capacity of one application window, then the “Windows” menu item is used to switch between the different windows of the application.

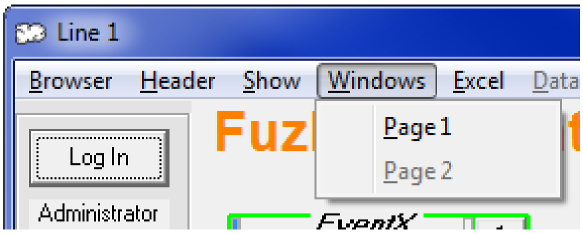


Fig. 12: Select application windows

If the application holds between 51 and 100 EventX components, then two windows are available, if it holds between 101 and 150 EventX components, then three windows may be accessed, etc.

4.5 Excel interface

The Excel menu item is used to enter the Excel management window. Click on Excel produces the following window:

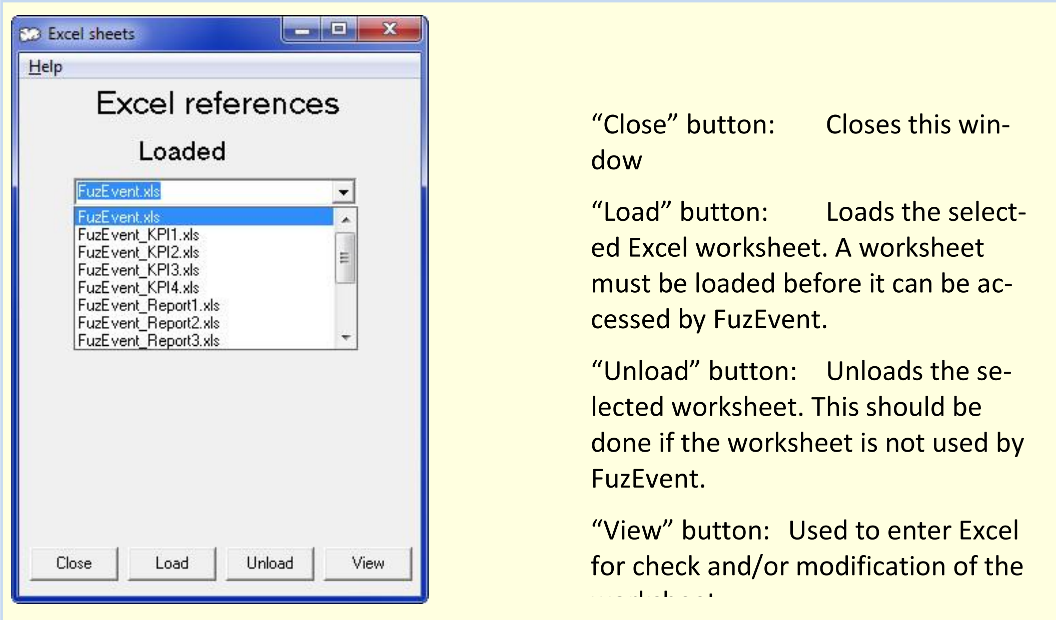


Fig. 13: The Excel interface

If a worksheet is selected from the drop down list, the current load status is displayed. A worksheet can only be accessed by FuzEvent if is loaded. On the other hand, all worksheets, which are not used in the FuzEvent, system should be unloaded.

Note: Excel **must** be closed by either the "Close" button or the "Hide" button on the Excel management window shown above. Excel must **not** be closed by the normal close functions of Excel. Closing the worksheet in Excel could lead to unpredicted results and crash of FuzEvent.

5 Browser

5.1 Browser introduction

The FuzEvent Browser is used to navigate through the whole FuzEvent system. The Browser is a useful tool for checking how the calculations and control functions are running, and it is the main module for maintenance of the control strategy through parameter adjustments. In addition, the Browser gives a good picture of the structure of the FuzEvent system.

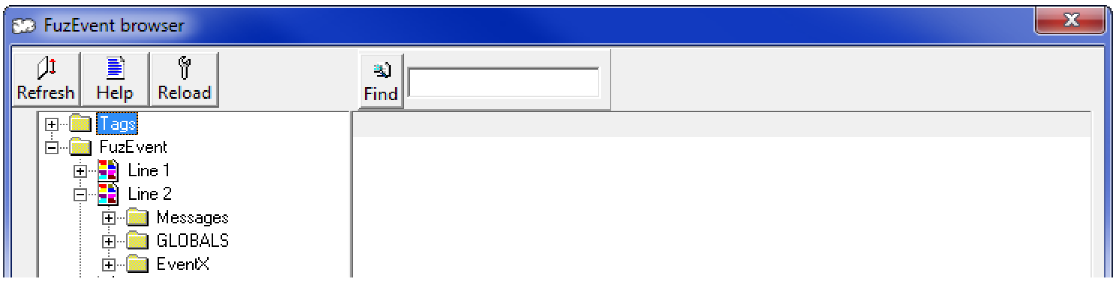


Fig. 14: The FuzEvent Browser window

The list view window to the left is used for display of:

- Tag variables

- Messages
- Global variables
- Local variables
- EventX status information
- EventX properties
- EventX script, including line values

The information in the list view window, such as values, is updated by click on the “Refresh” button. The “Reload” button closes the Browser and when it is re-opened, it starts in a fresh window where all the hierarchy is collapsed.

The “Find” button is used to locate variables or properties where the name includes the text, which is specified in the input field next to the “Find” button. If the “Find” button is clicked repeatedly, the Browser will step though the list and locate the next occurrence of the “Find”-text.

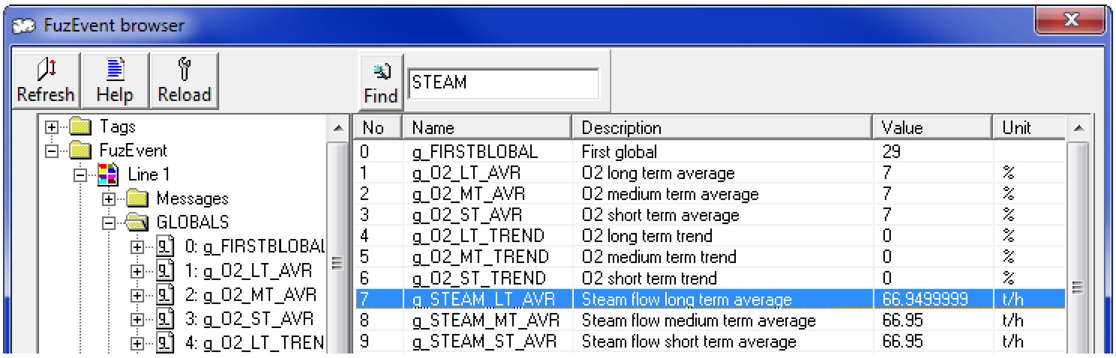


Fig. 15: Using Find in the Browser window

5.2 The message list

Click on “Messages” displays the message list in the list window. The newest message will be high lighted at the bottom of the list window. The message above the newest is the second newest, and the message below the newest is the oldest.

5.3 The EventX property list

Properties for an EventX component are defined in the Browser. Expansion of the "Property" folder for the selected EventX results in the property window shown below.

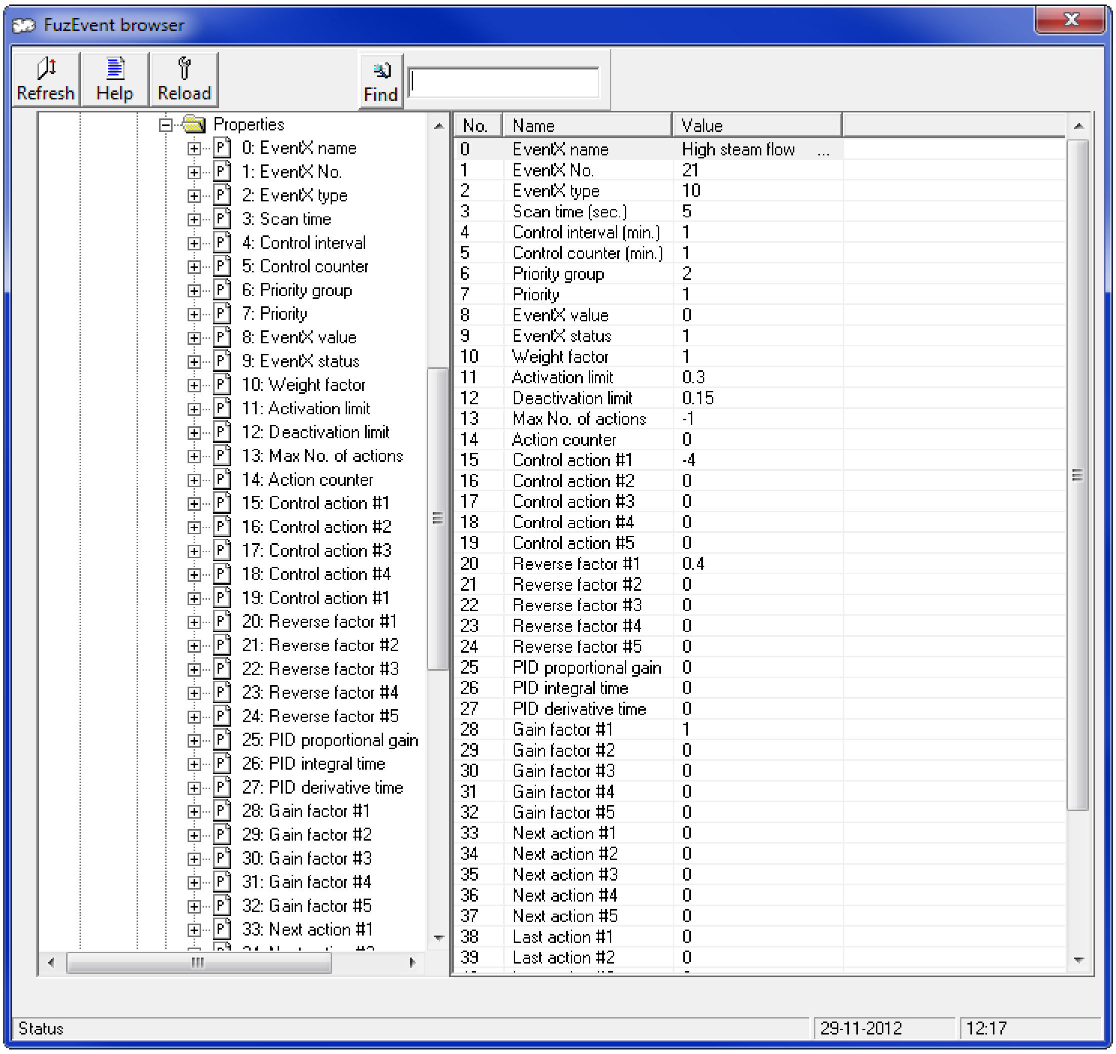


Fig. 16: EventX properties in the Browser window

The list of EventX properties is the same for all EventX components, but for a given EventX only some of the properties are used. The properties that are used depend on the EventX type. Some properties, however, are always used, and they are:

- EventX name — The name of the EventX component
- EventX type — Reference to a predefined control algorithm (default is 0, which means no predefined algorithm)
- Scan time — Time interval in seconds (default is 10 sec.)

The property value is changed by expansion of the property in the left part of the browser window, which produces an input field for key-in of the new value.

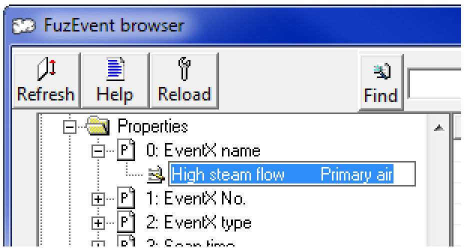


Fig. 17: Changing EventX properties in the Browser window

5.4 EventX properties

Properties for an EventX component are defined in the Browser. The following lists all EventX properties. It is the property "EventX type" that determines whether the property is used or not.

Each EventX holds properties for five control points (five set point values). It is, however, possible to work with more than five control points, but then No. 6, No. 7 etc. have to be configured without use of properties. Instead user defined variables (Locals or Globals) must be applied.

The list of properties is given in alphabetic order. The name below the property name is the variable name that can be accessed from FUEL scripts.

Acc actions #1
ACCACTIONS1
Accumulated control actions on control point No. 1.
This property is calculated by FuzEvent in connection with control algorithms wherethe accumulated actions are part of the algorithm, e.g. in connection with reverse actions.

Acc actions #2
ACCACTIONS2
Accumulated control actions on control point No. 2.
This property is calculated by FuzEvent in connection with control algorithms where the accumulated actions are part of the algorithm, e.g. in connection with reverse actions.

Acc actions #3
ACCACTIONS3
Accumulated control actions on control point No. 3.
This property is calculated by FuzEvent in connection with control algorithms where the accumulated actions are part of the algorithm, e.g. in connection with reverse actions.

Acc actions #4
ACCACTIONS4
Accumulated control actions on control point No. 4.
This property is calculated by FuzEvent in connection with control algorithms where the accumulated actions are part of the algorithm, e.g. in connection with reverse actions.

Acc actions #5
ACCACTIONS5
Accumulated control actions on control point No. 5.
This property is calculated by FuzEvent in connection with control algorithms where the accumulated actions are part of the algorithm, e.g. in connection with reverse actions.

Action counter
ACTIONCOUNT
Internal counter for the number of control actions.
This property is used in the control algorithms where "Max No. of actions" is a parameter.

Activation limit
Limit value for activation of the control algorithm. Typically this

ACTLIMIT	property is for control algorithms that activates when a certain process condition is detected.
Control action #1 CONTROLACTION1	Control action on control point #1. Typically, this property is used for control algorithms that activates when a certain process condition is detected.
Control action #2 CONTROLACTION2	Control action on control point #2. Typically, this property is used for control algorithms that activates when a certain process condition is detected.
Control action #3 CONTROLACTION3	Control action on control point #3. Typically, this property is used for control algorithms that activates when a certain process condition is detected.
Control action #4 CONTROLACTION4	Control action on control point #4. Typically, this property is used for control algorithms that activates when a certain process condition is detected.
Control action #5 CONTROLACTION5	Control action on control point #5. Typically, this property is used for control algorithms that activates when a certain process condition is detected.
Control counter CONTROLCOUNTER	Internal counter that counts the minutes between control actions. The counter is reset to 0 after a control action, and it s incremented every scan time.
Control interval CONTROLINTERVAL	Time interval in minutes between control actions. An EventX control component has a scan time and a control interval. Normally the scan time is shorter than the control interval time. The scan time may, for instance, be 10 seconds, whereas the control interval is 5 minutes.
Deactivation limit DEACTLIMIT	Limit value for deactivation of the control algorithm. Typically this property is for control algorithms that deactivates when a certain process condition is detected.
EventX active EVENTXACTIVE	User-calculated property to indicate that the EventX is active, which may be used by other EventX through the EVENTX(no,EVENTXACTIVE).
EventX passive EVENTXPASSIVE	User-calculated property to indicate that the EventX is not active, which may be used by the other EventX through the EVENTX(no,EVENTXACTIVE).
EventX status EVENTXSTATUS	Status value of the EventX component. The following status values are implemented, and their corresponding frame color of the EventX symbol:

Value:	Colour:	Description:
0	Green	Running and on-line
1	Yellow	Running
2	White	Stopped
9	Red	Error

Eventx type EVENTXTYPE	Type of pre-programmed control algorithm. The pre-programmed algorithms include various type of fuzzy algorithms, event type of algorithms as well as standard PID algorithms.

EventX value EVENTVALUE	User-calculated variable, which normally has a value between -1 and +1. For a control algorithm, the value normally indicates how active the EventX component is. At the EventX symbol at the application window, the EventX value is displayed graphically in the vertical bar graph to the right.
FUZZYON FUZZYON	This property is set by the user in the EventX script for switching ON/OFF the fuzzy mode for evaluation of rules. If FUZZYON is set to 1, then rules are treated as fuzzy rules, whereas FUZZYON equal to 0 means non-fuzzy treatment of IF-THEN rules.
Gain factor #1 GAINFACTOR1	Gain factor for control point #1. This property is normally used for control algorithms where the control action is first calculated as a normalized value, i.e. a value between -1 and +1. The gain factor is used to transform the normalized value into engineering units.
Gain factor #2 GAINFACTOR2	Gain factor for control point #2.
Gain factor #3 GAINFACTOR3	Gain factor for control point #3.
Gain factor #4 GAINFACTOR4	Gain factor for control point #4.
Gain factor #5 GAINFACTOR5	Gain factor for control point #5.
Last action #1 LASTACTION1	The last control action on control point #1, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
Last action #2 LASTACTION2	The last control action on control point #2, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
Last action #3 LASTACTION3	The last control action on control point #3, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
Last action #4 LASTACTION4	The last control action on control point #4, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
Last action #5 LASTACTION5	The last control action on control point #5, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
Max No. of actions MAXNOOFACTIONS	Maximum number of actions. Typically, this property is used for control algorithms that activates when a certain process condition is detected.
Next action #1 ACTIONVALUE1	The next control action on control point #1, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.

Next action #2 ACTIONVALUE2	The next control action on control point #2, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
Next action #3 ACTIONVALUE3	The next control action on control point #3, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
Next action #4 ACTIONVALUE4	The next control action on control point #4, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
Next action #5 ACTIONVALUE5	The next control action on control point #5, either calculated in the FUEL script, or as result of a pre-programmed control algorithm.
On Line EVENTXON	This property is not set in the Browser, but from the pop-up menu, which is displayed by right click on the EventX No. in the upper right corner of the EventX symbol. The EventX may also be set on-line from the script by setting EVENTXON equal to 1, and off-line by setting EVENTXON equal to 0
PID derivative time DERIVTIME	Derivative time of the PID controller. The PID algorithm of FuzEvent is: ACTIONVALUE1=Gain·(Δe + Itime·e + Dtime·(Δe-Δe_{t-1})), and this property specifies the value of Dtime.
PID integral time INTTIME	Integral time of the PID controller. The PID algorithm of FuzEvent is: ACTIONVALUE1=Gain·(Δe + Itime·e + Dtime·(Δe-Δe_{t-1})), and this property specifies the value of Itime.
PID proportional gain PROPGAIN	Proportional gain of the PID controller. The PID algorithm of FuzEvent is: ACTIONVALUE1=Gain·(Δe + Itime·e + Dtime·(Δe-Δe_{t-1})), and this property specifies the value of Gain.
Reverse factor #1 REVERSEFACTOR1	Factor that specifies the so-called reverse action on control point No. 1. This property is used for the control algorithms that has a reverse action, i.e. an action in the opposite direction of the adjustments, which were made to cope with the process situation. The reverse factor specifies the fraction of the accumulated action, which will be the reverse action.
Reverse factor #2 REVERSEFACTOR2	Factor that specifies the so-called reverse action on control point No. 2
Reverse factor #3 REVERSEFACTOR3	Factor that specifies the so-called reverse action on control point No. 3
Reverse factor #4 REVERSEFACTOR4	Factor that specifies the so-called reverse action on control point No. 4
Reverse factor #5 REVERSEFACTOR5	Factor that specifies the so-called reverse action on control point No. 5

Scan time SCANTIME	Scan time for execution of the EventX. The scan time is in seconds, and typically, the minimum scan time is 5 sec. If the scan time is set to 0, then the EventX may be executed from another EventX by using the RUN function. The EventX with scan time 0 is executed once every time the RUN function is executed.
Start EventX EVENTXSTART	This property is not set in the Browser, but from the pop-up menu, which is displayed by right click on the EventX No. in the upper right corner of the EventX symbol. The EventX may also be started or stopped from the script by setting EVENTXSTART equal to 1 or 0.
Weight factor EVENTXWEIGHT	For each EventX, FuzEvent calculates a so-called weight factor as part of the Priority Management System. The weight factor determines the weight that the EventX is allowed to execute its control adjustments.

5.5 The Script

The Browser display of the script code is a very efficient way to check how the calculations of the script are working. Next to the line numbers the Line Values are presented, which show the result of the script execution just before the “Refresh” button was clicked.

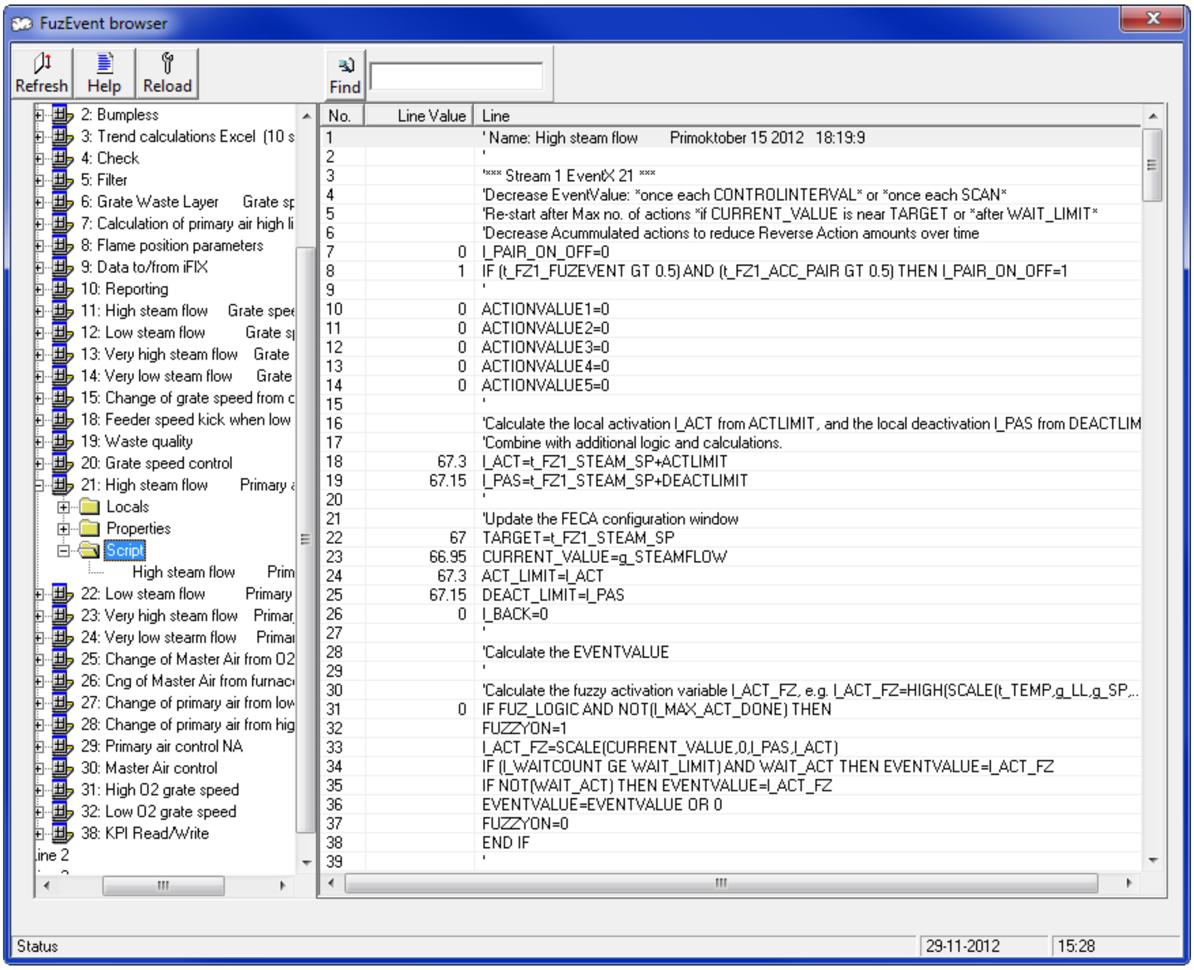


Fig. 18: EventX Script in the Browser window

The Line Values display the following information:

Assignment statement:	The values of the variable being calculated
Conditional assignment:	E.g.: IF X LT 3 THEN Y=Z displays the calculated value of Y if the condition is TRUE. If the condition is false, i.e. if X is greater than or equal to 3, then the line value is 0.
Label statement	The line values is equal to Label
Fuzzy rule	The line value is the grade of fulfilment, which is a Value between 0 and 1

The Line Values are updated when the “Refresh” button is clicked.

6 The EventX component

6.1 EventX introduction

A FuzEvent control strategy is composed by EventX components. Some of the components are for calculations, and others are control components, which calculate new set points for the controllers of the basic control system.

The EventX component is the entrance for definition of variables, for starting and stopping the execution of the EventX script, and for configuration of the EventX script.

Access to the definition of variables and to configuration of scripts is through a click with the right mouse button on the EventX No. at the upper right corner of the EventX component symbol. This will produce a pop-up window for selection of:

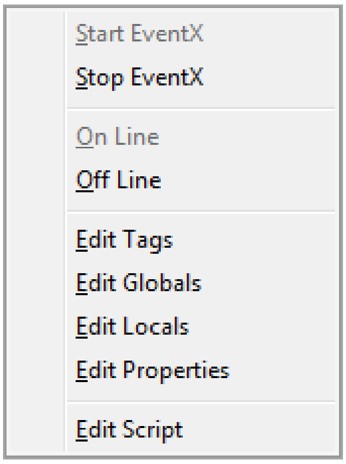


Fig. 19: Context menu when right-clicking EventX No.

By click on the EventX Name, a window will be displayed, which can be configured to show relevant information about the functionality of the EventX component. The main purpose of this feature is to enable display of information, which the operator may use for supervision of the FuzEvent control performance.

The EventX component symbol itself holds lots of dynamic information about the EventX status as shown below.

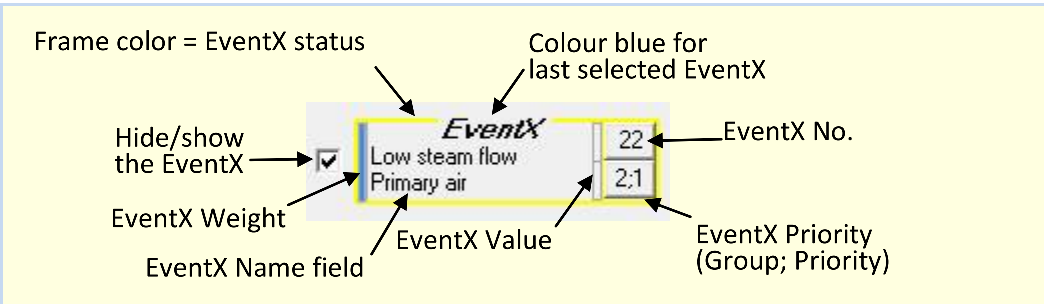


Fig. 20: The EventX component

6.2 The Priority Management System

The priority system of FuzEvent makes it possible to link EventX components into a priority network. The idea is to build a control strategy, which is composed of a number of individual control components that automatically are activated when belonging process situations are detected. If control components are in conflict, it is possible to specify which control components are more important that other components. In other words, it is possible to use the priority system to specify which process situations are the most important to control.

In a cement kiln, for instance, it is more important to control the CO content in the flue gasses than it is to control the oxygen content. Typically, two different EventX components are defined for CO control and for O2 control. To ensure that the CO control component takes over control of the kiln when CO is detected, and at the same time deactivates the O2 component, the two ExentX components are assigned to the same priority group, but with different priorities. The CO control component is for instance assigned priority 0 and O2 control is given the priority 1, i.e. the lower the priority number, the higher the priority.

The priority group and priority are EventX properties that are assigned values in the FuzEvent browser. The priority group and priority are shown in the lower right corner of the EventX symbol.

In waste incineration the main controls of Primary Air flow are typically using the Steam Flow as process value and the Primary Air Fan speed set point as the controlled value. The main controls are divided so that one set of EventX's are controlling when the situation is normal and another set takes over in exceptional process situations.

In the example shown below, the EventX named "Low steam flow – Primary air" belongs to group 2, and it has priority 1. The EventX component named "Very low steam flow – Primary air" also belongs to group 2, but it has priority 0, which shows that "Very low steam flow – Primary air" is more important to control than the control task of the PID controller.

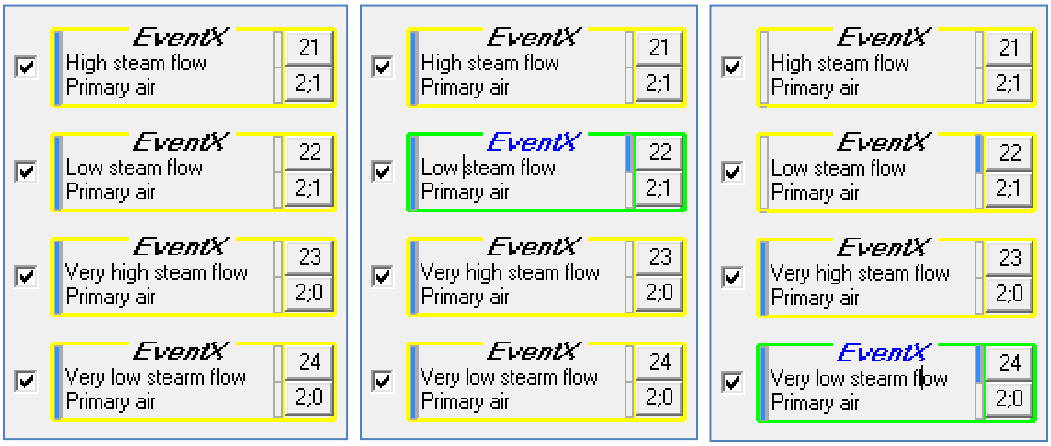


Fig. 21: Three situations in a priority group

The first picture in Fig. 21 illustrates a situation where the measured Steam Flow is within the Activation Limits of EventX 21 and 22.

In the second picture, the Steam Flow has dropped below the activation limit of EventX 22, but has not yet dropped below the activation limit of EventX 24. In this situation EventX 22 increases the output to the Primary Air fan inverter in small steps, small Actions. As seen, the EventX Value is high (+1) and the EventX Weight is high (1).

In the third picture, Steam Flow has dropped further, below the activation limit of EventX 24.
EventX 22's weight has dropped to zero (0) and EventX 22 has become passive.
EventX 24 now increases the output to the Primary Air fan inverter in large steps. As seen, the EventX Value is high (+1) and the EventX Weight is high (1). These "Very low...– Very high..." handlers are normally only allowed one or two actions, and when they have been executed, the control is gradually returned to the "High... – Low..." han-

dlers.

If the Steam Flow increases after EventX 24 have taken control, one or more Reverse Actions are executed, and the EventX Value of EventX 24 is decreased. This in turn increases the EventX Weight of EventX 22, allowing it to take part in the control.

6.2.1 The EventX Value and the EventX Weight

For an EventX component that enters a priority network, the so-called EVENTVALUE must be calculated. The EVENTVALUE must have a value between [-1,1], and the value should indicate how active the EventX component is. An example of a typical EVENTVALUE is an error signal that has been transformed into the interval from -1 to 1. See Fig. 22.

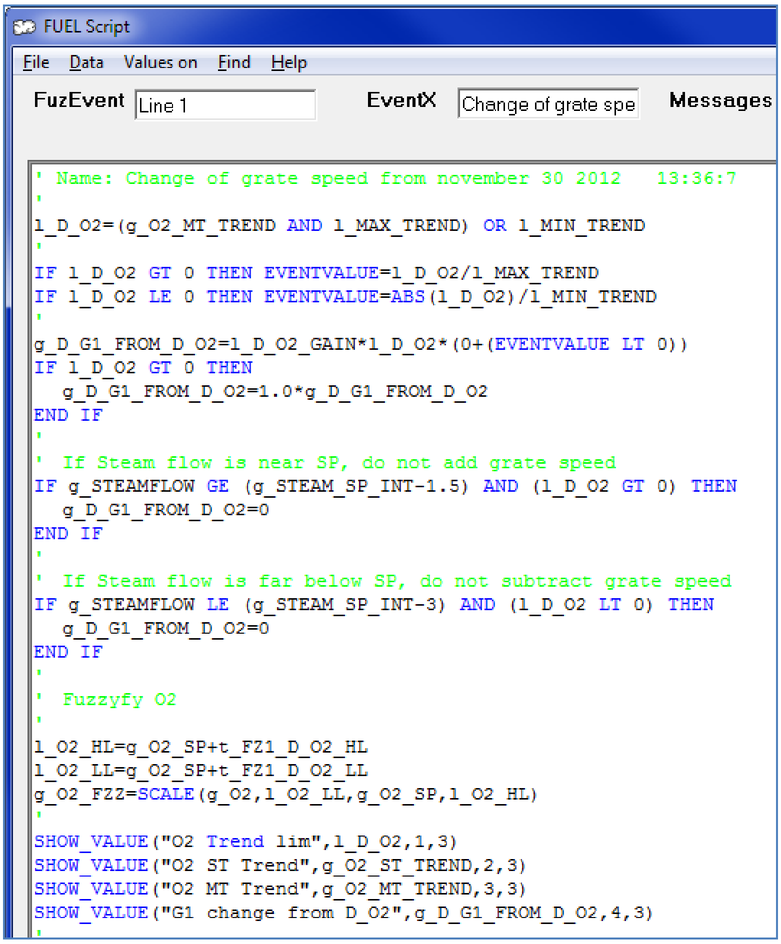


Fig. 22: Calculation of EventValue

For each EventX, FuzEvent calculates a weight factor W, the EventX Weight by:

W = (1 – ABS(EVENTVALUEi)) * (1 – ABS(EVENTVALUEj)) * etc.

Where EVENTVALUEi and EVENTVALUEj etc. are EVENTVALUEs of EventX components, which belong to the same priority net work, and which has a higher priority than the EventX, for which the weight factor is being calculated.

<u>Example:</u>

Assume we have three EventX components that belong to the same priority group, but with different priorities, e.g.

- EventX No. 1 with priority 1
- EventX No. 2 with priority 0

- EventX No. 3 with priority 0

If the EVENTVALUEs are 0.9, -0.5, and 0 for EventX No. 1, 2, and 3 respectively, then the weight factor for EventX No. 1 is:

W = (1 – ABS(-.5)) * (1 – ABS(0)) = 0.5

The weight factor is used to reduce the control actions from an EventX component, as FuzEvent automatically multiplies the control action, i.e. change of same set point value, by the weight factor. In other words, if the weight factor is less than 1, then the control adjustments are smaller than they would have been if no EventX with higher priority had been active.

The current weight factor and EVENTVALUE are shown graphically as a part of the EventX symbol as shown below. If the bar graph to the left does not fill from bottom to top, then the weight factor is correspondingly less than 1. The bar graph to the right shows the current EVENTVALUE. Above the centre line, the EVENTVALUE is positive, and below the centre line, the EVENTVALUE is negative.

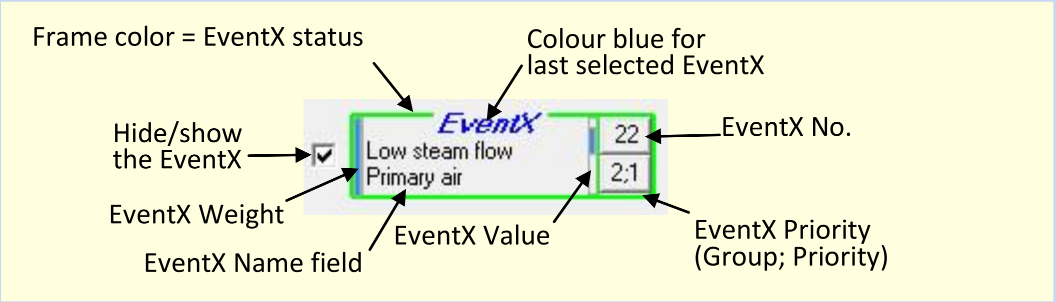


Fig. 23: EventX Weight and -Value

6.3 Start EventX / Stop EventX

Right click on the EventX No. in the upper right corner of the EventX component symbol produces the following pop-up window.

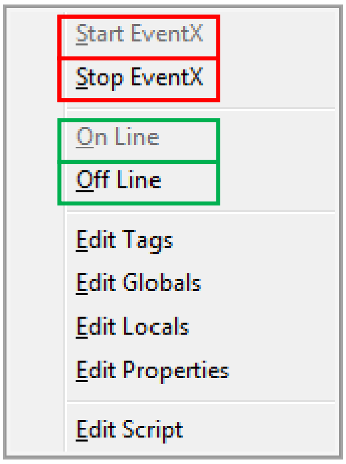


Fig. 24: Changing EventX execution.

If the script of the selected EventX is running, it is possible to select the Stop EventX item for stopping the execution. If the script execution is stopped, the EventX frame colour changes to white.

If the script of the selected EventX is stopped, it is possible to select the Start EventX item for starting execution of the script. If the script execution is started, the EventX frame colour changes to yellow.

6.4 On line / Off line

On-line means that Tag variables of type AO (analogue output) and DO (digital output) are transferred from FuzEvent to the container. By On/Off line it is thus possible to control when AO's and DO's are sent to the process.

If the selected EventX is On line, it is possible to select the Off Line item in Fig. 24 for stopping transfer of AO's and DO's to the container. If the EventX is switched Off-line, then the EventX frame colour changes from green to yellow

If the selected EventX is Off line and running (yellow EventX frame), it is possible to select the On Line item for starting transfer of AO's and DO's to the container. If the EventX is switched On-line, then the EventX frame colour changes from yellow to green.

If an EventX is On-line, and it is being stopped, then it is automatically switched Off-line.

6.5 Click on EventX name

Click on the EventX name, i.e. "Watch dog" in the example given below, produces the EventX property window.

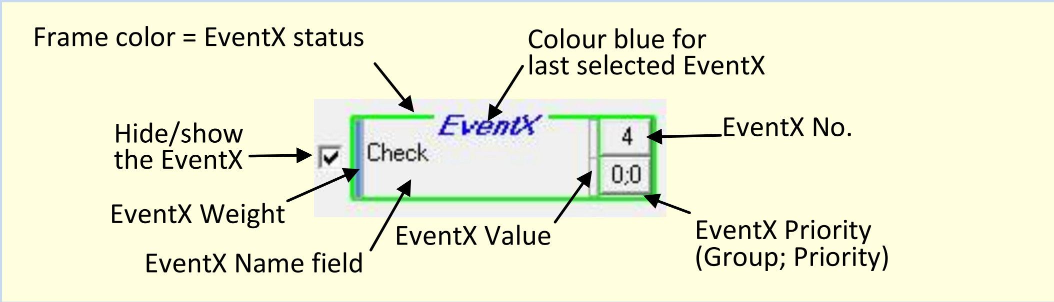


Fig. 25: Click on EventX name to show properties

The property window also displays the operator information, which has been configured in the EventX scripts by the SHOW_VALUE statement.

In addition, the operator may input values to variables, which has been configured to be shown on the property page. Right click on the name of the value for which a new value is to be keyed-in, produces an input window as shown below. Values and parameters changeable in this way are preceded by hatch #.

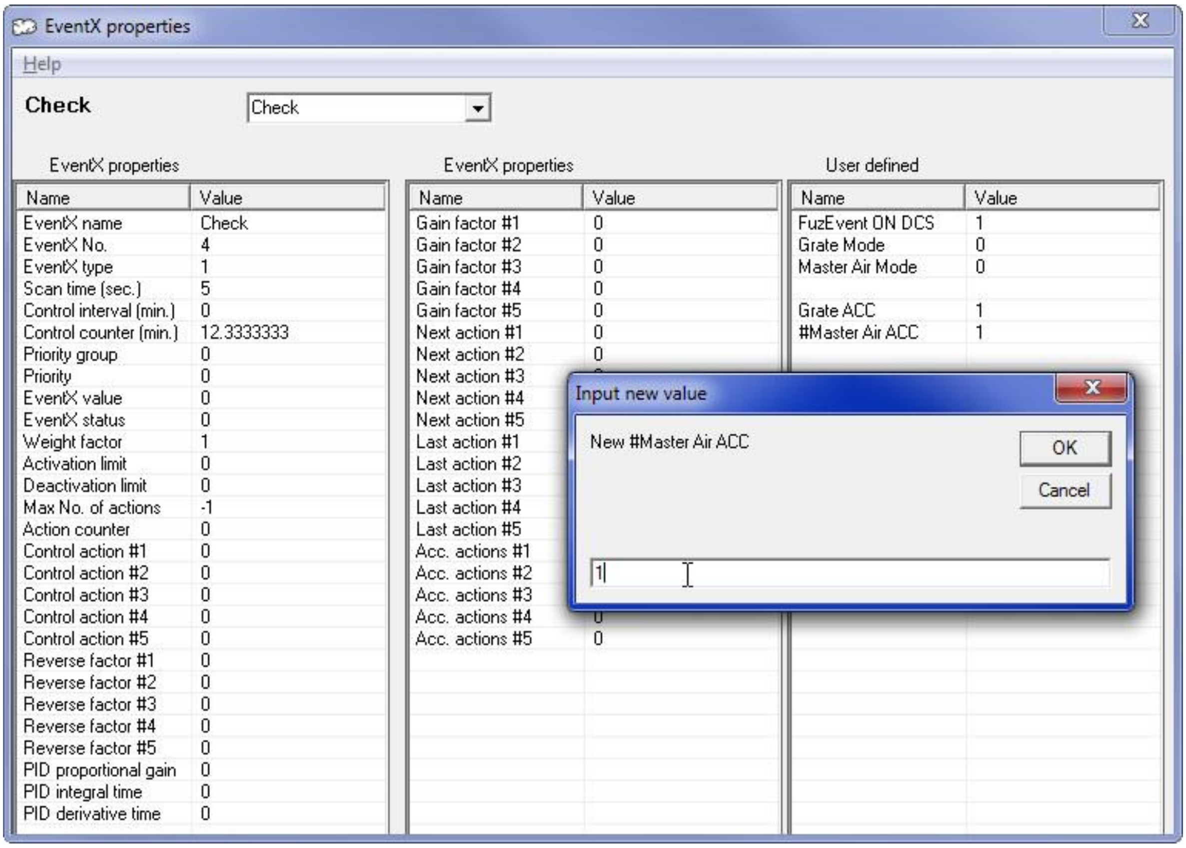


Fig. 26: Change parameters in the EventX Property window

7 Edit Tags

7.1 Tag definition

Right click on the EventX No. in the upper right corner of the EventX component symbol produces the following pop-up window.

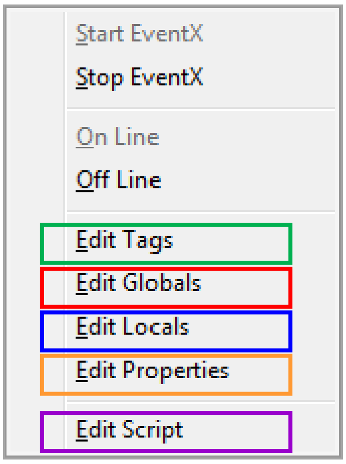


Fig. 27: Context menu when right-clicking EventX No.

Edit Tags is used for definition of Tag variables. Tag variables are used for exchange of data between FuzEvent and the container (refer to the FuzEvent “Introduction” help function).

Click on “Edit Tags” produces the Tag definition table as shown below:

Tag definitions

File Modify New Delete Find EventX Repair Help

No.	Name	Description	Value	Tag Unit	Tag Type	Tag address	
0	t_FZ_WATCHDOG	Watch dog	758		OUT	FuzTags;Stream1;DCS.FZ_WATCHI	4096
1	t_OPC_TEST	Description	0	%	AI		C
2	t_FZ1_FUZEVENT	FuzEvent ON/OFF switch (f_SCADA)	1		DI	FuzTags;Stream1;DCS.FZ1_FUZEV	C
3	t_FZ1_FUZEVENT_FZ	FuzEvent ON/OFF (t_iFIX)	1		OUT	FuzTags;Stream1;DCS.FZ1_FUZEV	C
4	t_FZ1_ACC_SAIR_FZ	Secondary air control ACC (f_iFIX)	0		AI		C
5	t_FZ1_ACC_GRATE	Grate speed control ACC	1		OUT	FuzTags;Stream1;DCS.FZ1_ACC_G	C
6	t_FZ1_ACC_PAIR	Primary air control ACC	1		OUT	FuzTags;Stream1;DCS.FZ1_ACC_P	C
7	t_FZ1_ACC_SAIR	Secondary air control ACC	0		OUT	FuzTags;Stream1;DCS.FZ1_ACC_S	C
8	t_AI_1401	O2 (f_SCADA)	7	%	AI	FuzTags;Stream1;DCS.AI-1401.F_0	C
9	t_FIC_1203	Steam flow (f_SCADA)	66.95	t/h	AI	FuzTags;Stream1;DCS.FIC-1203.F_C	C
10	t_FZ1_STEAM_SP	Steam flow SP (f_SCADA)	67	t/h	AI	FuzTags;Stream1;DCS.FIC-1203.F_1	C
11	t_AIC_1402	Grate speed output (f_SCADA)	7	%	AI	FuzTags;Stream1;DCS.AIC-1402.F_C	C
12	t_FZ1_GRATE_MODE	Grate speed controller mode	0		AI	FuzTags;Stream1;DCS.AIC-1402.F_3	C
13	t_FZ1_GRATE_MODE_	Grate speed controller mode from FuzEvent	0		OUT	FuzTags;Stream1;DCS.FZ1_GRATE	C
14	t_FZ1_GRATE_SPEEC	Grate speed from FuzEvent	32.4	%	OUT	FuzTags;Stream1;DCS.FZ1_GRATE	C
15	t_FZ1_GRATE_HL_FZ	Grate speed high limit f. iFIX	80	%	AI	FuzTagsOpr;Stream1Opr;FIX.FZ1_G	C
16	t_FZ1_GRATE_LL_FZ	Grate speed low limit f. iFIX	5.000381	%	AI	FuzTagsOpr;Stream1Opr;FIX.FZ1_G	C
17	t_FCI_1208	Primary air	80000	kg/h	AI	FuzTags;Stream1;DCS.FCI-1208.F_C	C
18	t_FCI_1209	Secondary air	19000	kg/h	AI	FuzTags;Stream1;DCS.FCI-1209.F_C	C
19	t_FZ1_PAIR_HL_FZ	Primary air high limit	111000.2	kg/h	AI	FuzTagsOpr;Stream1Opr;FIX.FZ1_P.	C
20	t_FZ1_PAIR_LL_FZ	Primary air low limit	20000.92	kg/h	AI	FuzTagsOpr;Stream1Opr;FIX.FZ1_P.	C

Fig. 28: Tag definition table

The columns in the Tag table are:

Tag No.: Tag variable No.

Tag Name: Name of Tag variable.
Note that all Tag variables start with t_.

	The t_ is automatically added to the Tag name when the Tag variable is defined.
Description:	Description of Tag variable
Tag Value:	Current value of the Tag variable
Tag Unit:	Engineering unit of the Tag variable
Tag Type:	Type of Tag variable. The following variable types exist:

AI	Analogue input from container to FuzEvent
DI	Digital (1/0) input from container to FuzEvent
AO	Analogue output from FuzEvent to container
DO	Digital output (1/0) from FuzEvent to container
INT	Variable, which can be used for both input and output.

In addition to the columns mentioned above, the Tag table may contain columns, which are used for specific container systems only.

When a Tag variable has been modified, or a new Tag has been defined, FuzEvent will automatically reload all scripts to update the references to the new Tag variable list.

7.2 Modify Tag

The “Modify” menu item is used for modification of an existing Tag variable. Selection of a Tag variable in the Tag table, followed by click on “Modify” opens Tag variable fields below the Tag table, i.e.:

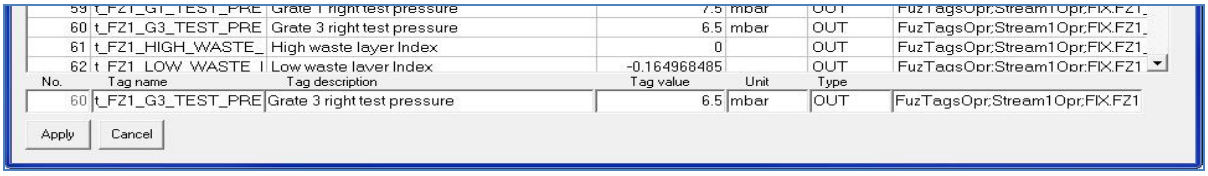


Fig. 29: Modify Tag

Fill-in the input fields for modification of the existing Tag, followed by click on “Apply”. Double click on an existing Tag variable also produces the Tag variable fields below the Tag able.

7.3 New Tag

The “New” menu item is used for definition of a new Tag variable. Select the Tag in the Tag table after which the new Tag variable should be positioned. Then click on “New”, which will produce input fields below the Tag table, and an empty line in the Tag table, i.e.:

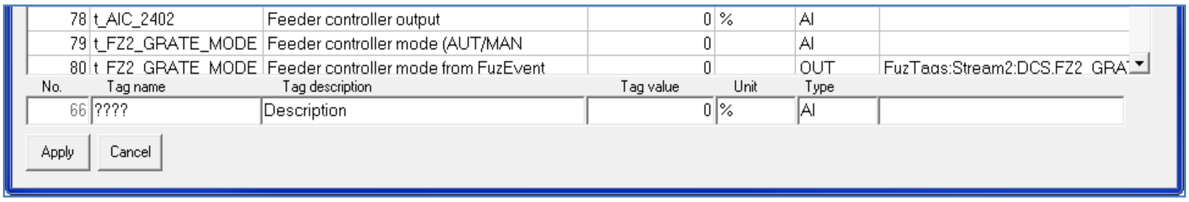


Fig. 30: New Tag

Fill-in the input fields for definition of the new Tag variable, followed by click on “Apply”

7.4 Delete Tag

The “Delete” menu item is used to delete an existing Tag variable. Select the Tag in the Tag table, which should be deleted, followed by click on “Delete”. FuzEvent asks for a confirmation that you really want to delete the selected Tag variable.

7.5 Find Tag

The “Find” menu item is used to locate the next occurrence in the Tag name of a text, which is specified in the input box that is displayed after a click on “Find”. The next occurrence is the Tag name below the currently selected Tag in the Tag table.

7.6 Tag used in EventX

The “EventX” menu item is used to find the EventX components where a selected Tag variable is being used. Click on a Tag variable in the Tag table, followed by click on “EventX”, produces a window a shown below showing the names of the EventX where the Tag is used, the EventX No. and the name of the FuzEvent application.

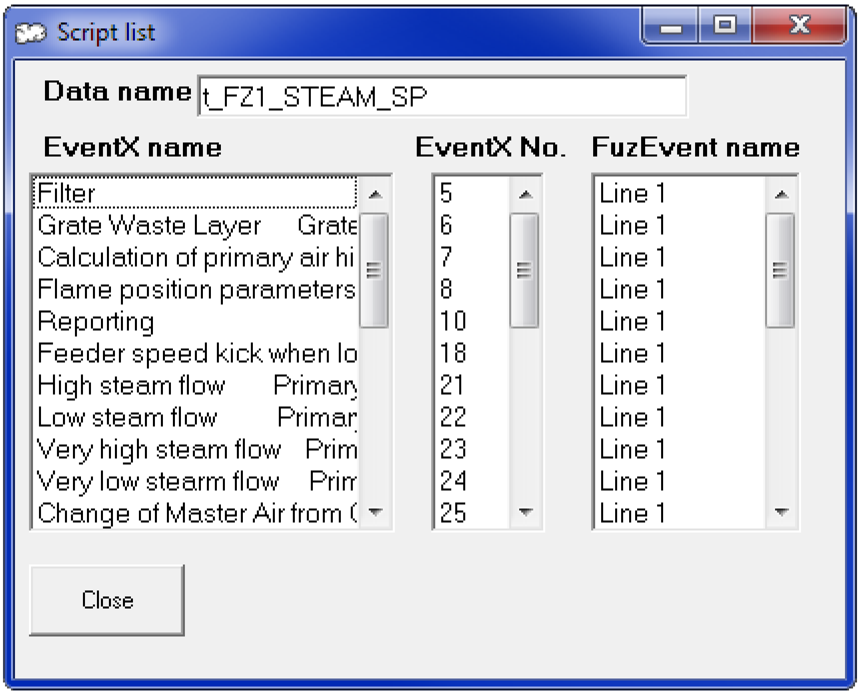


Fig. 31: Tag used in EventX

7.7 Repair Tag table

If unexpectedly the Tag data base is corrupted, then it is possible to repair the data base by click on “Repair”.

8 Edit Globals

8.1 Globals definition

Edit Globals in Fig. 27 is used for definition the so-called Global variables. Global variables can be accessed from all EventX of the belonging FuzEvent application. Global variables may thus be used to transfer information from one EventX to another EventX in the same application

Click on “Edit Globals” produces the definition table for Global variables as shown below:

No	Name	Description	Value	Unit	Elements	Height
0	g_FIRSTBLOBAL	First global	29		1	0
1	g_O2_LT_AVR	O2 long term average	7	%	1	0
2	g_O2_MT_AVR	O2 medium term average	7	%	1	0
3	g_O2_ST_AVR	O2 short term average	7	%	1	0
4	g_O2_LT_TREND	O2 long term trend	0	%	1	0
5	g_O2_MT_TREND	O2 medium term trend	0	%	1	0
6	g_O2_ST_TREND	O2 short term trend	0	%	1	0
7	g_IR_MAXTEMP$1	Max. temperature in the four rows: Row1	952.08	DegC	4	0
11	g_STEAM_LT_AVR	Steam flow long term average	66.949999999	t/h	1	0
12	g_STEAM_MT_AVR	Steam flow medium term average	66.95	t/h	1	0
13	g_STEAM_ST_AVR	Steam flow short term average	66.95	t/h	1	0
14	g_STEAM_LT_TREND	Steam flow long term trend	0	/min	1	0
15	g_STEAM_MT_TREND	Steam flow medium term trend	0	/min	1	0
16	g_STEAM_ST_TREND	Steam flow short term trend	0	/min	1	0
17	g_O2	Filtered O2	7.0000000000	%	1	0
18	g_STEAMFLOW	Filtered steam flow	66.95	t/h	1	0
19	g_GRATE_FZ	Grate speed from FuzEvent	32.4	%	1	0
20	g_PAIR_FZ	Primary air from FuzEvent	29	%	1	0
21	g_SAIR_FZ	Secondary air from FuzEvent	0	%	1	0
22	g_GRATE_HL	Grate speed high limit f. iFIX	80	%	1	0
23	g_GRATE_LL	Grate speed low limit f. iFIX	5.000381	%	1	0
24	g_D_GRATE_HL	Delta grate speed high limit	-0.000279193	%	1	0
25	g_D_GRATE_LL	Delta grate speed low limit	0	%	1	0
26	g_D_G1_FROM_D_O2	Change of G1 from change of O2	0	%	1	0
27	g_FLAMEPOS_SP	Flame position SP	17000		1	0
28	g_FLAMEPOS_BACK_TI	Time for the flame to go back to SP (min)	60	min	1	0
29	g_FURNACE_TEMP	Average furnace temperature	900	DegC	1	0

Fig. 32: Globals definition table – Array collapsed

The columns in the Globals table are:

No.:	Global variable No.
Name:	Name of Global variable. Note that all Global variables start with g_. The g_ is automatically added to the Global name when the Global variable is defined.
Description:	Description of the Global variable
Value:	Current value of the Global variable
Unit:	Engineering unit for the Global variable
Elements:	Number of elements in the Global variable. In the Globals table, Global array elements are shown as the name of the Global variable followed by $ and the element No. In the table shown above, g_IR_MAXTEMP$1 is the first element of the Global array g_IR_MAXTEMP, which has four elements. Use the “ArrayExpand” menu item for display of all the array elements.
Height:	Internal variable used when the Global variable is used in fuzzy rule calculations.

When a Global variable has been modified, or a new Global variable has been defined, FuzEvent will automatically reload all the scripts of the FuzEvent application to which the Global variable belongs.

8.2 Modify Globals

The “Modify” menu item is used for modification of an existing Global variable in the same way as described under “Modify Tag”.

8.3 New Globals

The “New” menu item is used for definition of a new Global variable in the same way as described under “New Tag”.

8.4 Delete Globals

The “Delete” menu item is used to delete an existing Global variable in the same way as described under “Delete Tag”

8.5 Find Globals

The "Find" menu item is used to locate the next occurrence of a Global variable in the same way as described under “Find Tag”.

8.6 Global used in EventX

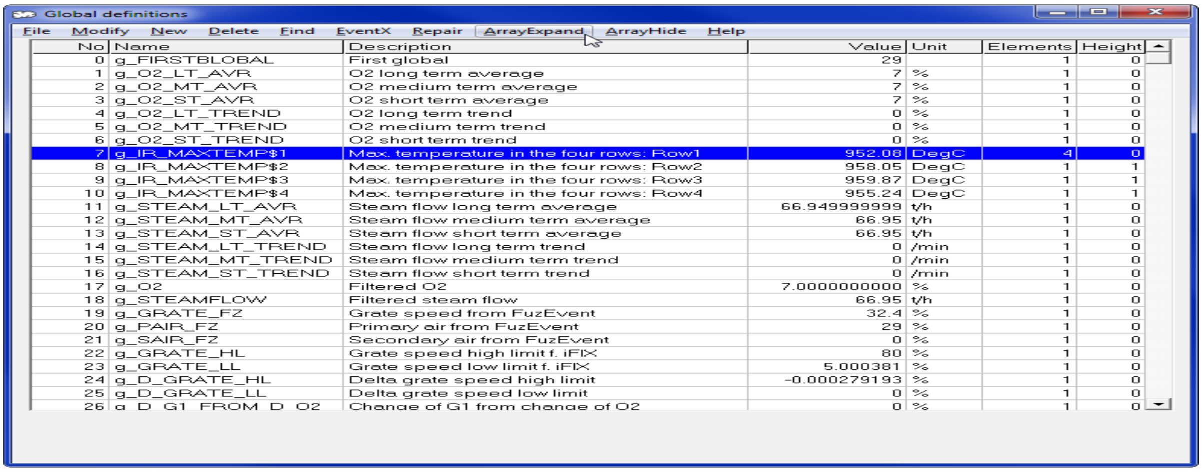
The "EventX" menu item is used to find the EventX components where a selected Global variable is being used in the same way as described in “Tag used in EventX”.

8.7 Repair Globals

If unexpectedly the Globals data base is corrupted, then it is possible to repair the data base by click on "Repair".

8.8 Array expand

The “ArrayExpand” menu item is used for display all elements in the selected Global variable. This item is only relevant to use if the variable has more than one element. After selection of a Global variable with more than one element, a click on “ArrayExpand” produces the following variable list, where the four elements of the array g_IR_MAXTEMP$1..$4 are displayed.



No	Name	Description	Value	Unit	Elements	Height
0	g_FIRSTBLOBAL	First global	29		1	0
1	g_O2_LT_AVR	O2 long term average	7	%	1	0
2	g_O2_MT_AVR	O2 medium term average	7	%	1	0
3	g_O2_ST_AVR	O2 short term average	7	%	1	0
4	g_O2_LT_TREND	O2 long term trend	0	%	1	0
5	g_O2_MT_TREND	O2 medium term trend	0	%	1	0
6	g_O2_ST_TREND	O2 short term trend	0	%	1	0
7	g_IR_MAXTEMP$1	Max. temperature in the four rows: Row1	952.08	DegC	4	0
8	g_IR_MAXTEMP$2	Max. temperature in the four rows: Row2	958.05	DegC	1	1
9	g_IR_MAXTEMP$3	Max. temperature in the four rows: Row3	959.87	DegC	1	1
10	g_IR_MAXTEMP$4	Max. temperature in the four rows: Row4	955.24	DegC	1	1
11	g_STEAM_LT_AVR	Steam flow long term average	66.949999999	t/h	1	0
12	g_STEAM_MT_AVR	Steam flow medium term average	66.95	t/h	1	0
13	g_STEAM_ST_AVR	Steam flow short term average	66.95	t/h	1	0
14	g_STEAM_LT_TREND	Steam flow long term trend	0	/min	1	0
15	g_STEAM_MT_TREND	Steam flow medium term trend	0	/min	1	0
16	g_STEAM_ST_TREND	Steam flow short term trend	0	/min	1	0
17	g_O2	Filtered O2	7.0000000000	%	1	0
18	g_STEAMFLOW	Filtered steam flow	66.95	t/h	1	0
19	g_GRATE_FZ	Grate speed from FuzEvent	32.4	%	1	0
20	g_PAIR_FZ	Primary air from FuzEvent	29	%	1	0
21	g_SAIR_FZ	Secondary air from FuzEvent	0	%	1	0
22	g_GRATE_HL	Grate speed high limit f. iFIX	80	%	1	0
23	g_GRATE_LL	Grate speed low limit f. iFIX	5.000381	%	1	0
24	g_D_GRATE_HL	Delta grate speed high limit	-0.000279193	%	1	0
25	g_D_GRATE_LL	Delta grate speed low limit	0	%	1	0
26	g_D_G1_FROM_D_O2	Change of G1 from change of O2	0	%	1	0

Fig. 33: Globals definitions table - Array expanded.

8.9 Array hide

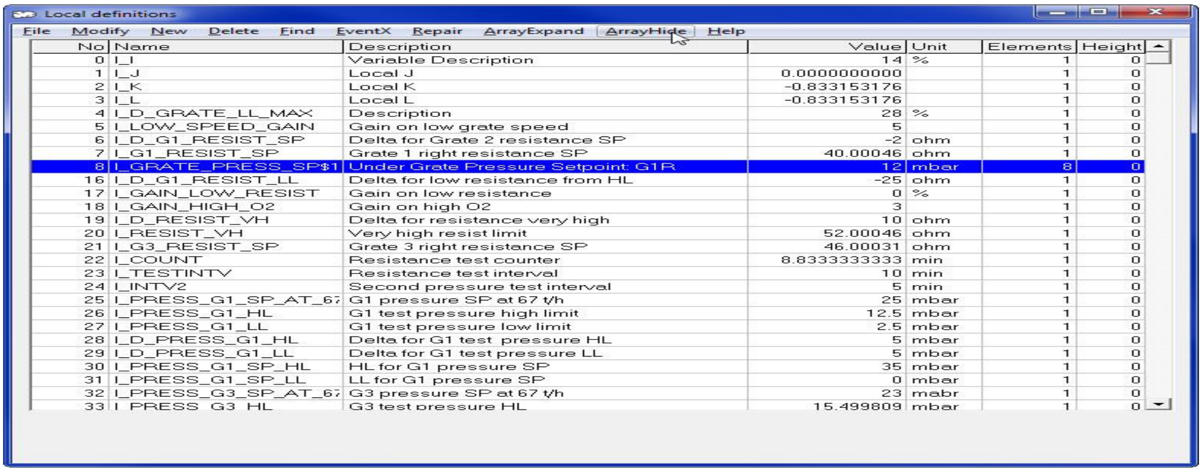
The “ArrayHide” menu item is used to turn off the display of all elements in the selected Global variable, see “Array expand”.

9 Edit Locals

9.1 Locals definition

Edit Locals in Fig. 27 is used for definition the so-called Local variables. A Local variable can only be accessed from the EventX where it is defined.

Click on “Edit Locals” produces the definition table for Local variables as shown below:


Local definitions

File Modify New Delete Find EventX Repair ArrayExpand ArrayHide Help

No	Name	Description	Value	Unit	Elements	Height
0	l_I	Variable Description	14	%	1	0
1	l_J	Local J	0.0000000000		1	0
2	l_K	Local K	-0.833153176		1	0
3	l_L	Local L	-0.833153176		1	0
4	l_D_GRATE_LL_MAX	Description	28	%	1	0
5	l_LOW_SPEED_GAIN	Gain on low grate speed	5		1	0
6	l_D_G1_RESIST_SP	Delta for Grate 2 resistance SP	-2	ohm	1	0
7	l_G1_RESIST_SP	Grate 1 right resistance SP	40.00046	ohm	1	0
8	l_GRATE_PRESS_SP$1	Under Grate Pressure Setpoint: G1R	12	mbar	8	0
16	l_D_G1_RESIST_LL	Delta for low resistance from HL	-25	ohm	1	0
17	l_GAIN_LOW_RESIST	Gain on low resistance	0	%	1	0
18	l_GAIN_HIGH_O2	Gain on high O2	3		1	0
19	l_D_RESIST_VH	Delta for resistance very high	10	ohm	1	0
20	l_RESIST_VH	Very high resist limit	52.00046	ohm	1	0
21	l_G3_RESIST_SP	Grate 3 right resistance SP	46.00031	ohm	1	0
22	l_COUNT	Resistance test counter	8.8333333333	min	1	0
23	l_TESTINTV	Resistance test interval	10	min	1	0
24	l_INTV2	Second pressure test interval	5	min	1	0
25	l_PRESS_G1_SP_AT_67	G1 pressure SP at 67 t/h	25	mbar	1	0
26	l_PRESS_G1_HL	G1 test pressure high limit	12.5	mbar	1	0
27	l_PRESS_G1_LL	G1 test pressure low limit	2.5	mbar	1	0
28	l_D_PRESS_G1_HL	Delta for G1 test pressure HL	5	mbar	1	0
29	l_D_PRESS_G1_LL	Delta for G1 test pressure LL	5	mbar	1	0
30	l_PRESS_G1_SP_HL	HL for G1 pressure SP	35	mbar	1	0
31	l_PRESS_G1_SP_LL	LL for G1 pressure SP	0	mbar	1	0
32	l_PRESS_G3_SP_AT_67	G3 pressure SP at 67 t/h	23	mabr	1	0
33	l_PRESS_G3_HL	G3 test pressure HL	15.499809	mbar	1	0

Fig. 34: Locals definitions table - Array collapsed.

The columns in the Locals table are:

No.:	Global variable No.
Name:	Name of Local variable. Note that all Local variables start with l_. The l_ is automatically added to the Local name when the Local variable is defined.
Description:	Description of the Local variable
Value:	Current value of the Local variable
Unit:	Engineering unit for the Local variable
Elements:	Number of elements in the Local variable. In the Locals table, Local array elements are shown as the name of the Local variable followed by $ and the element No. In the table shown above, l_GRATE_PRESS_SP$1 is the first element of the Local array , l_GRATE_PRESS_SP, which has eight elements. Use the “ArrayExpand” menu item for display of all the array elements.
Height:	Internal variable used when the Global variable is used in fuzzy rule calculations.

When a Local variable has been modified, or a new Local variable has been defined, FuzEvent will automatically reload the script to which the Local variable belongs.

9.2 The Locals menu items

All the menu items of the Locals definition window have the same functions as described in connection with definition of Global variables in 8.1.

10 Edit Properties

Edit Properties in Fig. 27 is used for adjusting the properties of the FECA routines.

FECA is the name of the FuzEvent Control Algorithm. The FUEL library script of the FECA algorithm can be fetched by selecting Open from the File-menu in the FUEL editor.

An existing FECA script can be opened and edited as described in 11 Edit Script.

A dedicated configuration window has been defined for FECA. The configuration picture is displayed by first click on the EventX No. in the upper right corner of the EventX symbol, and then by selecting Edit Properties, which produces the following picture:

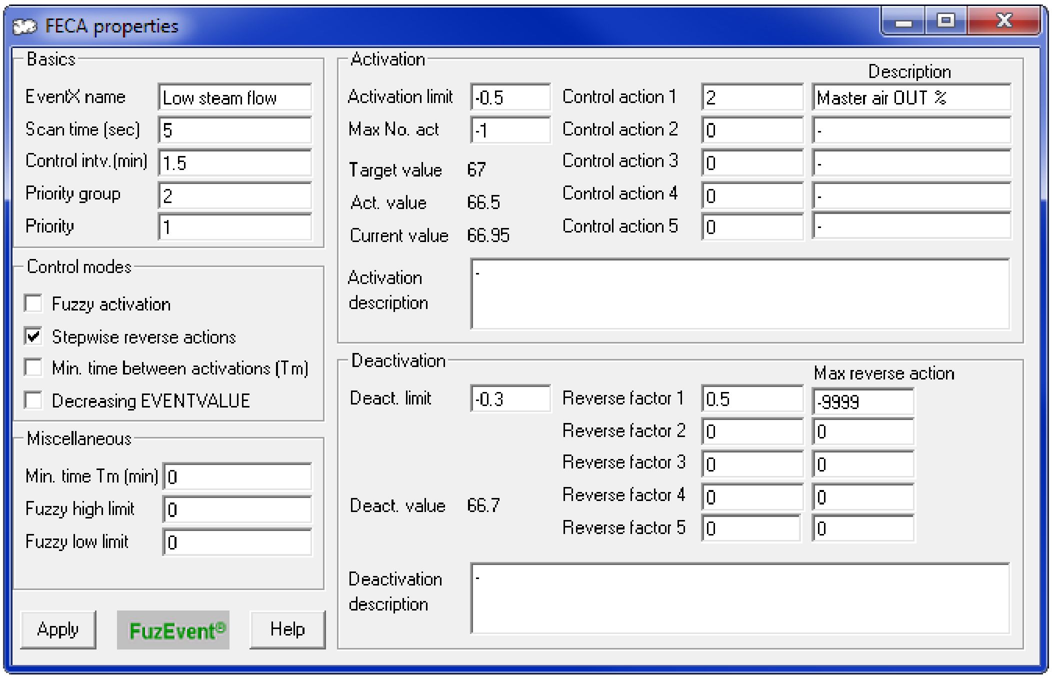


Fig. 35: FECA Properties window

Note: The configuration picture is only displayed if the EventX has EventX Type No. 10 or No. 11, which is specified in the Browser under the EventX properties.

10.1 Basics

The Basics box is used to specify:

- The name of the EventX
- The scan in seconds, which is the time interval between execution of the EventX script
- The control interval in minutes, which is the time interval between change of the control parameter
- The EventX priority group and priority. Please refer to 6.2 The Priority Management System earlier in this manual.

10.2 Control modes

The Control modes box is used to specify:

- Selection of "Fuzzy activation" activates the so-called fuzzy activation
- Selection of "Stepwise reverse actions" activates the so-called stepwise reverse actions

- Selection of "Min time between activations Tm" enables the feature by which it is possible to specify a minimum time in minutes between two activations. This function is used if there is a risk that EventX activates again immediately or shortly after it has deactivated.
- Selection of decreasing EVENTVALUE. This feature is used if a maximum number of actions have been specified, and if lower priority EventX gradually should become active when this EventX has reached the maximum number of actions. If this feature is selected, then the action counter will continue to run even after the maximum number of actions has been reached. No actions, of course, will be executed after the maximum number of actions has been reached. The EVENTVALUE, however, will be calculated by the following, after the maximum number of actions has been reached:

  EVENTVALUE = EVANTVALUE * MAXNOOFACTIONS / ACTIONCOUNT

  By this it can be seen that the EVENTVALUE will decrease as the ACTIONCOUNT increases, by which a lower priority EventX gradually will regain its weight factor (Please refer to section 6.2 The Priority Management System earlier in this manual.)

10.3 Miscellaneous

The Miscellaneous box is used to specify:

- The minimum time in minutes between activation of the EventX
- The two spare parameters named Fuzzy high limit and Fuzzy low limit are the system variables, i.e. FUZHL and FUZLL, which may be used e.g. for calculation of the fuzzy activation logic.

10.4 Activation

The Activation box is used to specify:

- The Activation limit, which is the system variable ACTLIMIT. Normally ACTLIMIT is a delta-value, which is used to calculate the activation value from a target or set point value. The Activation box shows the set point (TARGET) and the calculated activation value (ACT_LIMIT), which are updated from the EventX script e.g. by:

  ```
  l_ACT=t_FZ1_STEAM_SP+ACTLIMIT
  TARGET=t_FZ1_STEAM_SP
  ACT_LIMIT=l_ACT
  ```

  The Activation box also shows the current value of the process measurement, which is updated from the EventX script e.g. by:

  ```
  CURRENT_VALUE=g_STEAMFLOW
  ```

- Max No. act, which is the parameter used for specification of the maximum number of actions. During one activation of the EventX it is thus possible to define the maximum number of actions that the EventX is allowed to execute. The value -1 means no limit to the number of actions.
- Control action 1 to 5, which defines the adjustment of up to five control parameters. Next to the control adjustments it is possible to specify a description of the actual control parameter.
- Finally, the Activation box holds a field for description of the calculations and/or logic, which is used for activation of the EventX.

10.5 Deactivation

The Deactivation box is used to specify:

- The Deactivation limit, which is the system variable DEACTLIMIT. Normally DEACT-LIMIT is a delta-value, which is used to calculate the deactivation value from a target or set point value. The Deactivation box shows the deactivation value, which is updated from the EventX script e.g. by:
  l_PAS=t_FZ1_STEAM_SP+DEACTLIMIT
  DEACT_LIMIT=l_PAS
- Reverse factor 1 to 5, which defines the reverse factor for up to 5 control points. Next to the Reverse factor a maximum reverse action must be specified by which it is possible to limit the size of the reverse action.
- Finally, the Deactivation box holds a field for description of the calculations and/or logic, which is used for deactivation of the EventX.

11 Edit Script

11.1 Script introduction

Edit Script in Fig. 27 is used for configuration of the calculations and the control functions of FuzEvent. The script language is named FUEL, i.e. FUzzy Event Language. FUEL is especially designed for configuration of the control algorithms, which represents the high level control philosophy of FuzEvent.

Click on “Edit Script” produces a programming window for the selected EventX as shown below.

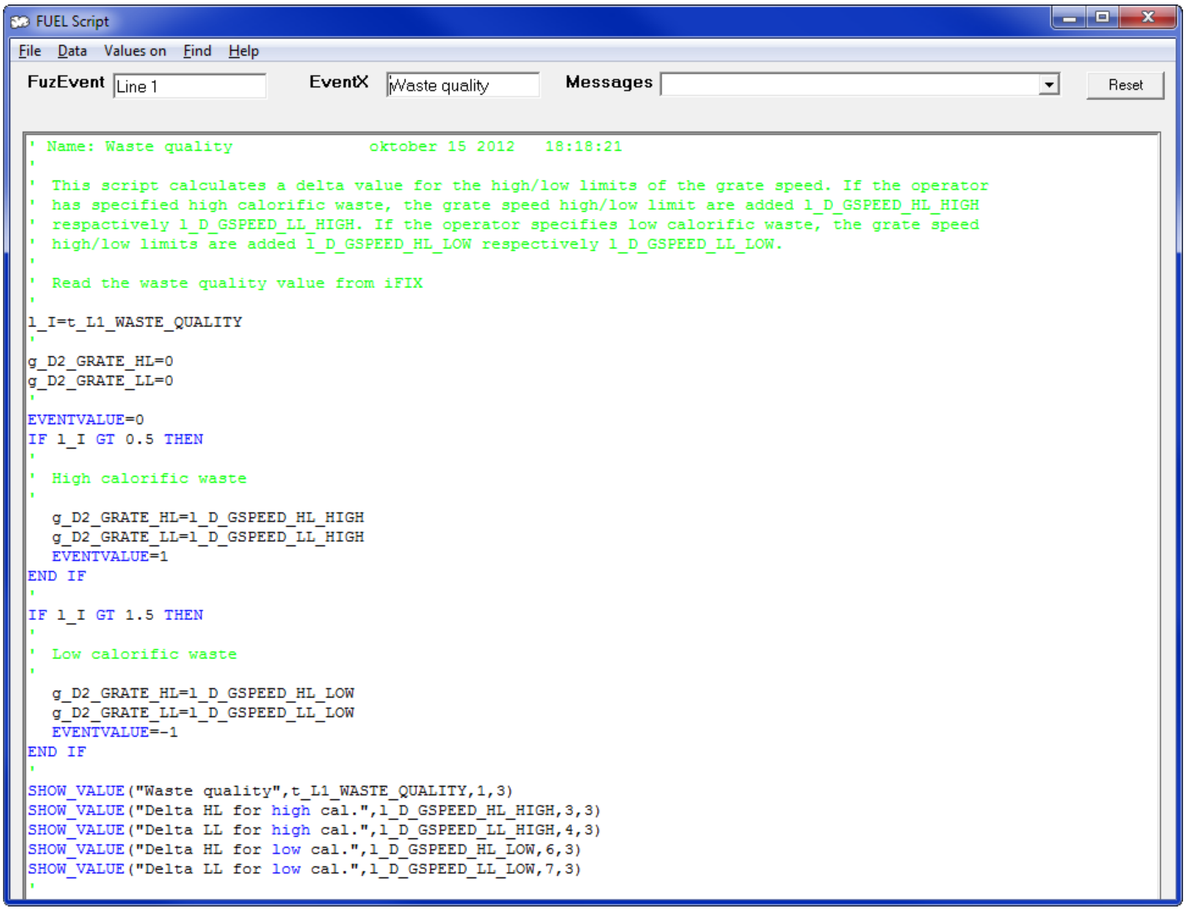


Fig. 36: Edit Script window

The menu item “File” holds the following sub-menu items:

- “Save” Used for compilation of the script into a new executable program
- “Save As” Used to save the script in a user named file, which then can be used as starting point for a new EventX script.
- “Open” Used to insert script code that previously has been saved by using the “Save As” menu item
- “Close” Used to close and exit the script definition window

The menu item “Data” holds the following sub-menu items:

- “Tags” Used for display of the Tag variables in a window, which includes Tag No., Tag name, actual Tag value, and the Tag description.

- “Globals”Used for display of the Globals variables in a window, which includes Global No., Global name, actual Global value, and the Global variable description.
- “Locals” Used for display of the Local variables in a window, which includes Local No., Local name, actual Local value, and the Local variable description.

The window below shows an example with Global variables.

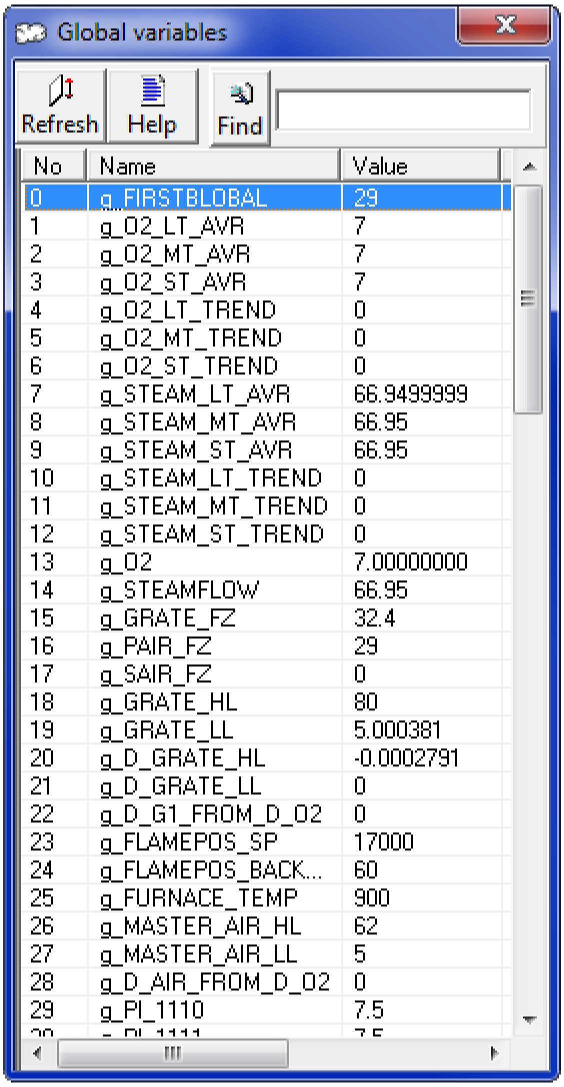


Fig. 37: List of Globals

Click on “Refresh” displays the actual values of the Global variables.

The “Find” button is used to find the next occurrence in the variable name of the search text, which has been entered in the input field next to the “Find” button.

The “Help” menu item activates the help functions on the FUEL script language.

11.2 The Script editor

The FuzEvent script editor includes features, which help the user to key-in the variable names.

In FUEL, it is easy to find out whether a variable is a Tag variable, a Global variable or a Local variable. Tag variables starts with t_ , Global variables start with g_ , and Local variable start with l_ . In the example in Fig. 36, t_L1_WASTE_QUALITY is a Tag variable, g_D2_GRATE_HL is a Global variable, and l_D_GSPEED_HL_HIGH is a Local variable.

If, for instance, the user wants to refer to a Tag variable, then just write t_ , after which the list of Tag variables pops-up as shown below. The “Find” button can be used to locate the Tag variable, or the user may use the slider to manually find the Tag variable, which should be inserted into the script.

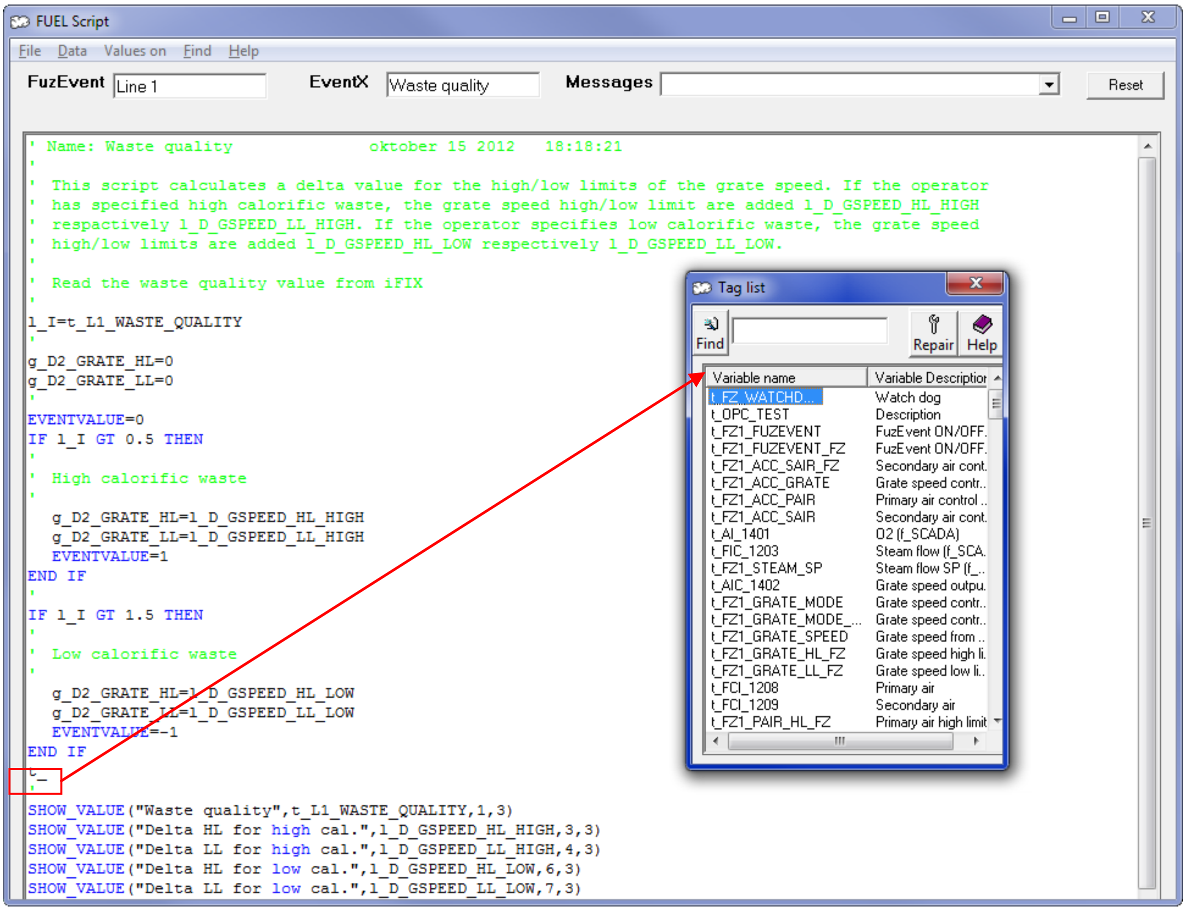


Fig. 38: Using the pop-up menu to locate variables

When the Tag variable has been located in the list, the user double click on the Tag name after which the complete Tag name is inserted in the script.

The same works for Global variables and Local variables, where the variable list pops-up when g_ or l_ has been keyed-in.

11.2.1 Known words

The FUEL editor automatically changes the colour to blue for known words. In the example above END, IF, SHOW_VALUE, EVENTVALUE are all predefined terms, which automatically are written in blue to make the script easier to read.

11.2.2 Comments

The ‘ as the first character is used to put comments into the script. The colour of a comment line is automatically changed to green.

11.2.3 Errors

If an error is detected during compilation of the script, the line with the error is changed into red colour.

```
'  Low calorific waste
'
   g_D2_GRATE_HL=1_D_GSPEED_HL_LOW
   g_D2_GRATE_LL=1_D_GSPEED_LL_LOW
   EVENTVALUE=-1
END IF
t_L3_GRATE_SPEED=3
'
```

Simultaneously, an error message is shown in the “Messages” field in the upper right corner of the script editor window. As shown below, the message line includes the following information:

- Date and time
- FuzEvent application No. and EventX component No.
- Message text.

In the message shown above, -1-19 means FuzEvent application No. 1, and EventX No. 19

The “Reset” button next to the messages field is used to acknowledge messages so that it is easy to identify new messages. Messages which have been acknowledged have a double asterisk in front of the message line, i.e.:

12 The EventX type library

12.1 EventX type algorithms

FuzEvent includes a library of predefined control algorithms, which are referred to by an EventX type number.

The predefined library functions are:

Type Name

- No algorithm
- General control
-
-
-
-
-
-
-
- FECA GT
- FECA LT
-
- PID Controller
- Neural net
-
-
-
-
- Proportioner
- Fuel master
- Fuel type

12.2 EventX type 0 (No algorithm)

The EventX type 0 is for user defined internal control algorithms using fuzzy rules or other types of calculations. Only the most important EventX properties are shown in the EventX property window, which is displayed by click on the EventX name on the EventX symbol. Further EventX type 0 cannot write TAG values to the container. That is, it is always considered to be off-line as described in 6.4 On line / Off line. EventX scripts of type 0 are mostly used for internal service like Trending, Filter and Reporting.

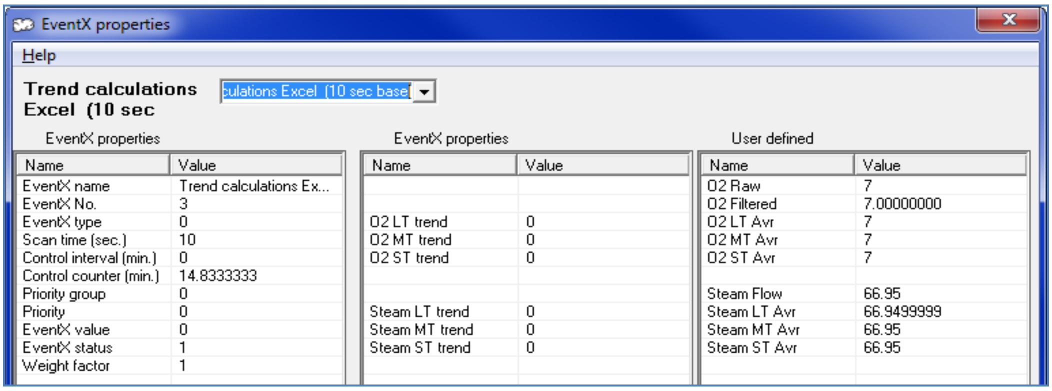


Fig. 39: EventX type 0 properties

12.3 EventX type 1 (General control)

The EventX type 1 is for user defined control algorithms using fuzzy rules or other types of calculations. All EventX properties are shown in the EventX property window, which is displayed by click on the EventX name on the EventX symbol. EventX type 1 can write TAG values to the container.

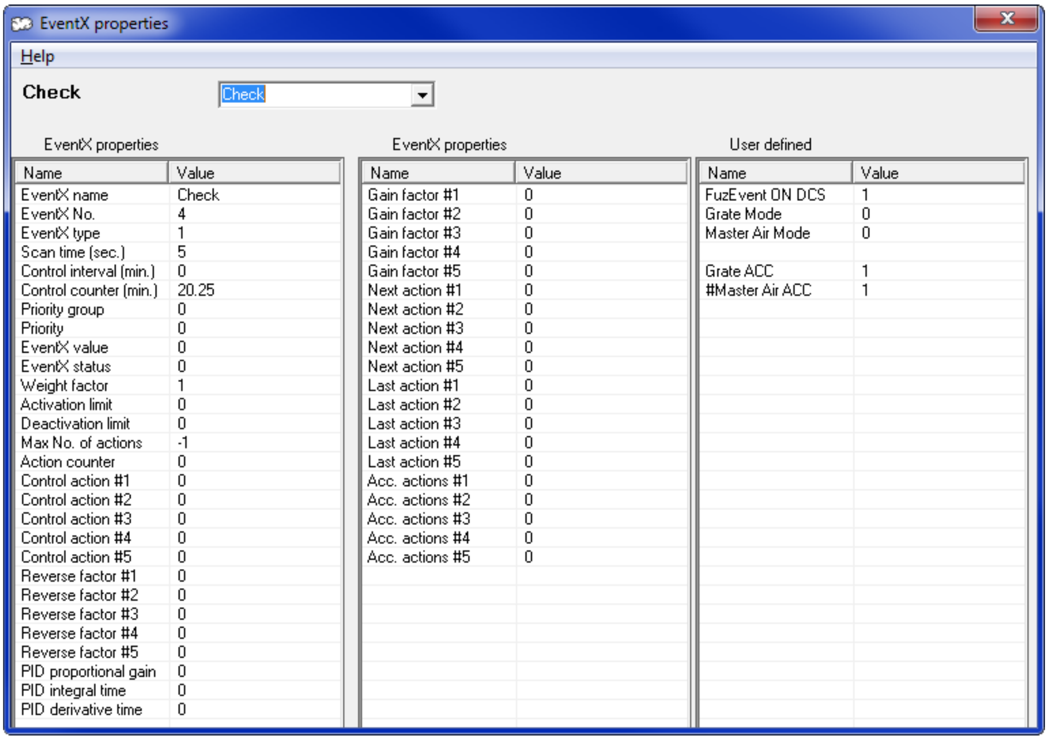


Fig. 40: EventX type 1 properties

12.4 EventX type 10 FECA GT and type 11 FECA LT

The script code of the FECA algorithm is accessed from the FUEL script editor. In the script editor, click on the "File" menu item in the upper left corner, and then click on the "Open", which produces the pop-up window with the FuzEvent library scripts, i.e.:

Entry No. 10 and No. 11 refer to two different versions of FECA. To fetch the script and the set of Local variables, double click on either No. 10 or No. 11 by which the script is transferred to the EventX for which the FUEL editor is open.

Library script No. 10, i.e. FECA_GT is used for control of a process state where a process measurement, or a combination of measurements, is greater than a target- or a set point value, hence the _GT, which refers to Greater Than.

Library function No. 11, i.e. FECA_LT, is similar to FECA_GT, except that FECA_LT is used for control of a process state, where the process value is Less Than the target.

Having double clicked on either FECA_GT or FECA_LT, the script is sent to the EventX, and the necessary Local variables are defined automatically. The next step is to Save the script after which it is ready to be adapted to the actual control task. The library script holds various references to a Local variable named l_TO_BE_CHANGED, which indicate which variables normally have to be changed.

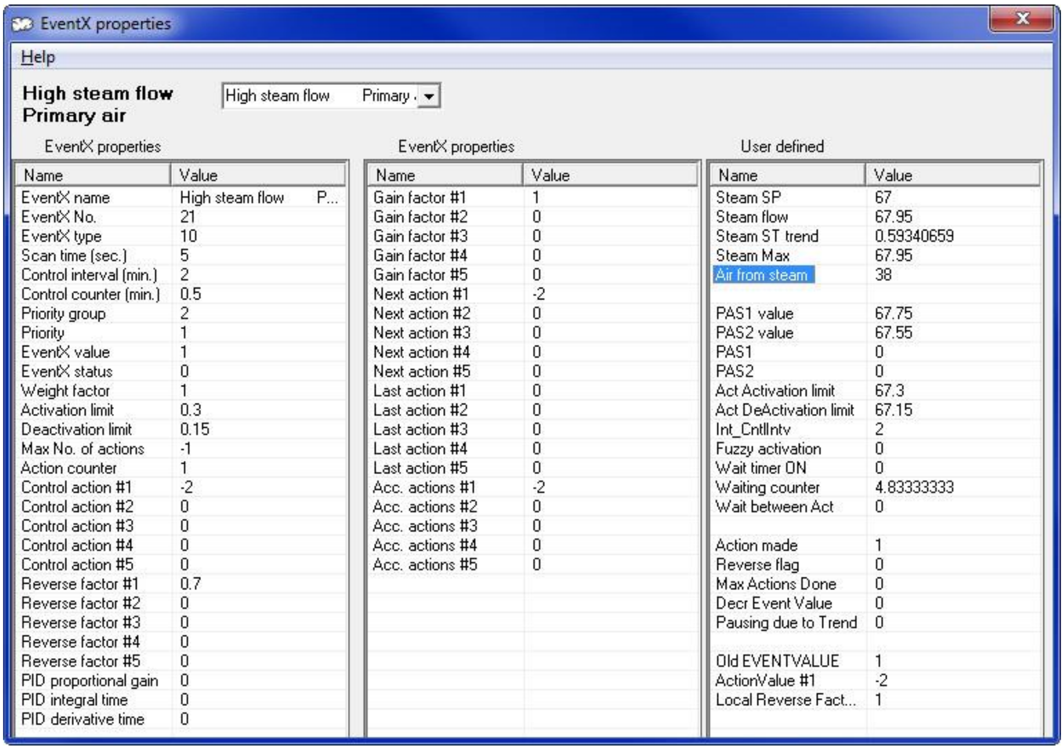


Fig. 41: FECA type EventX (type 10 and 11) properties

The basic FECA control function is illustrated below:

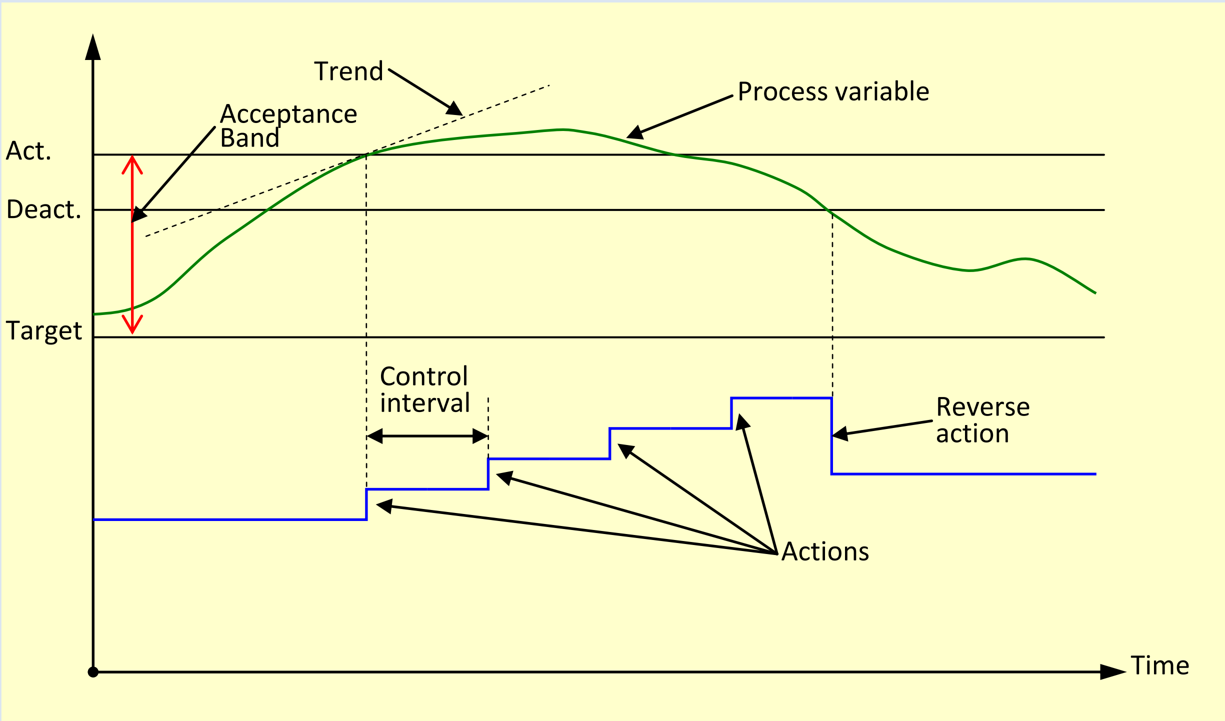


Fig. 42: FECA basic control function

In Fig. 42, the green curve shows a process measurement for which a high activation limit (Act. limit) has been defined. Typically the high activation limit is defined in relation to a set point or target value, i.e. the activation limit is the set point plus some delta value, marked as the Acceptance band. In addition to an activation limit, a deactivation value is defined. The activation values defined when the control algorithm should start making control actions, and the deactivation value is the value where the algorithm should become passive again, i.e. where it will stop making control adjustments.The blue curve illustrates the control parameter (output) that is used for keeping the process measurement close to the target.

A steam flow is an example of a process measurement for which a target or set point has been defined, and the speed of a motor driving a fan is an example of a control parameter by which the steam flow is kept close to the steam flow set point.

For the example shown above, an EventX activation value has been defined. Normally the activation value is defined as a target value plus a delta value, i.e. ACT = SP + DELTA. A deactivation value is also defined which is a little lower than the activation value to avoid the EventX from switching between activation and deactivation if the process measurement is oscillating around the activation value.

A control action has been defined together with an interval between actions, the control interval. Finally, a reverse factor has been defined, which determines the size of the control action in the opposite direction of the changes, which were made when the EventX was active.

The reverse action is the accumulated value of the changes, which were made when the EventX was active, multiplied with the specified Reverse factor.

If, for instance, the sum of the four actions shown above is 8, i.e. 4*2, and if the Reverse factor is 0.7, then the reverse action will be -8*0.7 = -5.6. The control parameter, in other words, is decreased by 5.6.

A dedicated configuration window has been defined for FECA. The configuration picture is displayed by first click on the EventX No. in the upper right corner of the EventX symbol, and then by selecting Edit Properties, the configuration window for the basic FECA algorithm is shown as in Fig. 43.

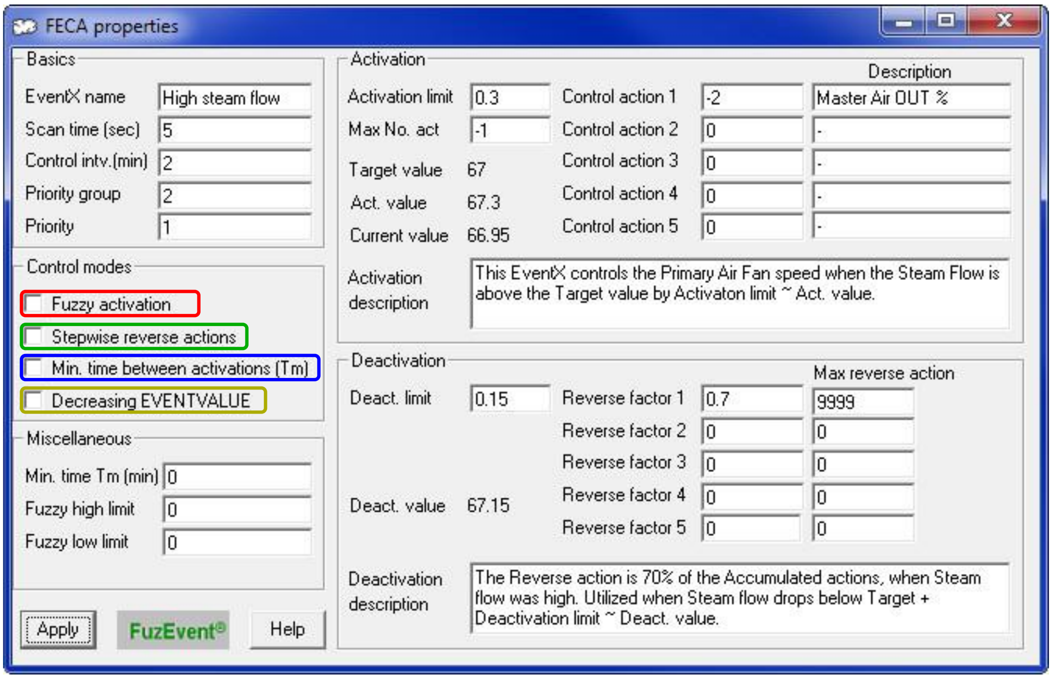


Fig. 43: FECA configuration window

Note: The configuration picture is only displayed if the EventX has EventX Type No. 10 or No. 11, which is specified in the Browser under the EventX properties.

Activation Desc.: This EventX controls the Primary Air Fan speed when the Steam Flow is above the Target value by Activation limit ~ Act. value.

Deactivation Desc.: The Reverse action is 70% of the Accumulated actions, when Steam flow was high. Utilized when Steam flow drops below Target value + Deactivation limit ~ Deact. Value.

12.4.1 Basics

The Basics box is used to specify:

- The EventX name.
- The Scan time in seconds, which is the time interval between execution of the EventX script.
- The Control interval in minutes, which is the time interval between change of the control parameter.
- The EventX Priority group and Priority. Refer to The Priority Management System on page 28 for further explanation.

12.4.2 Control modes

The Control modes box is used to specify:

- Selection of Stepwise reverse actions activates the stepwise reverse actions, where Reverse actions are divided into three steps and are executed at three different levels between Max. Value and Deact. value as seen in Fig. 44. Stepwise reverse actions is further explained in 12.4.6 under
- Selection of Fuzzy activation activates the fuzzy activation where the first control actions are executed when the measured process value are above the Deact. value. These actions are not counted in Action counter. Fuzzy activation is further explained in paragraph 12.4.7 under
- Selection of Min time between activations Tm enables the feature by which it is possible to specify a minimum time in minutes between two activations. This function is used if there is a risk that EventX activates again immediately or shortly after it has deactivated.
- Selection of Decreasing EVENTVALUE . This feature is used if a maximum number of actions have been specified, and if lower priority EventX gradually should become active when this EventX has reached the maximum number of actions.

If the “Decreasing EVENTVALUE” feature is selected, then the action counter will continue to run even after the maximum number of actions has been reached. No actions will of course be executed after the maximum number of actions has been reached. The EVENTVALUE, however, will be calculated by the following, after the maximum number of actions has been reached:

EVENTVALUE = EVENTVALUE * MAXNOOFACTIONS / ACTIONCOUNT

By this it can be seen that the EVENTVALUE will decrease as the ACTIONCOUNT increases, by with lower priority EventX gradually will regain their weight factor (refer to the "Priority Management System" of the FuzEvent Help)

12.4.3 Miscellaneous

The Miscellaneous box is used to specify:

- The minimum time in minutes between activation of the EventX
- The two spare parameters named Fuzzy high limit and Fuzzy low limit are the system variables, i.e. FUZHL and FUZLL, which may be used e.g. for calculation of the fuzzy activation logic, user programming is needed for this.

12.4.4 Activation

The Activation box is used to specify:

- The Activation limit, which is the system variable ACTLIMIT. Normally ACTLIMIT is a delta-value, which is used to calculate the activation value from a target or set point value. The Activation box shows the set point (TARGET) and the calculated activation value (ACT_LIMIT), which are updated from the EventX script e.g. by:

TARGET = t_FZ1_STEAM_SP

I_ACT = t_FZ1_STEAM_SP + ACTLIMIT

ACT_LIMIT=I_ACT

- The Activation box also shows the current value of the process measurement, which is updated from the EventX script e.g. by

CURRENT_VALUE = g_STEAMFLOW

- Max No. act, which is the parameter used for specification of the maximum number of actions. During one activation of the EventX it is thus possible to define the maximum number of actions that the EventX is allowed to execute. The value -1 means no limit to the number of actions.
- Control action 1 to 5, which defines the adjustment of up to five control parameters. Next to the control adjustments it is possible to specify a description of the actual control parameter.
- Finally, the Activation box holds a field for description of the calculations and/or logic, which is used for activation of the EventX.

12.4.5 Deactivation

The Deactivation box is used to specify:

- The Deactivation limit, which is the system variable DEACTLIMIT. Normally DEACTLIMIT is a delta-value, which is used to calculate the deactivation value from a target or set point value. The Deactivation box shows the deactivation value, which is updated from the EventX script e.g. by:

l_PAS = t_FZ1_STEAM_SP + DEACTLIMIT

DEACT_LIMIT = l_PAS

- Reverse factor 1 to 5, which defines the reverse factor for up to 5 control points. Next to the Reverse factor a maximum reverse action must be specified by which it is possible to limit the size of the reverse action.
- Finally, the Deactivation box holds a field for description of the calculations and/or logic, which is used for deactivation of the EventX.

12.4.6 Stepwise reverse actions

The FECA control algorithm features a stepwise reverse actions-function. Stepwise reverse actions means that instead of making the reverse action in one step, it is divided into three steps, which are executed when the process measurement starts to approach the deactivation value.

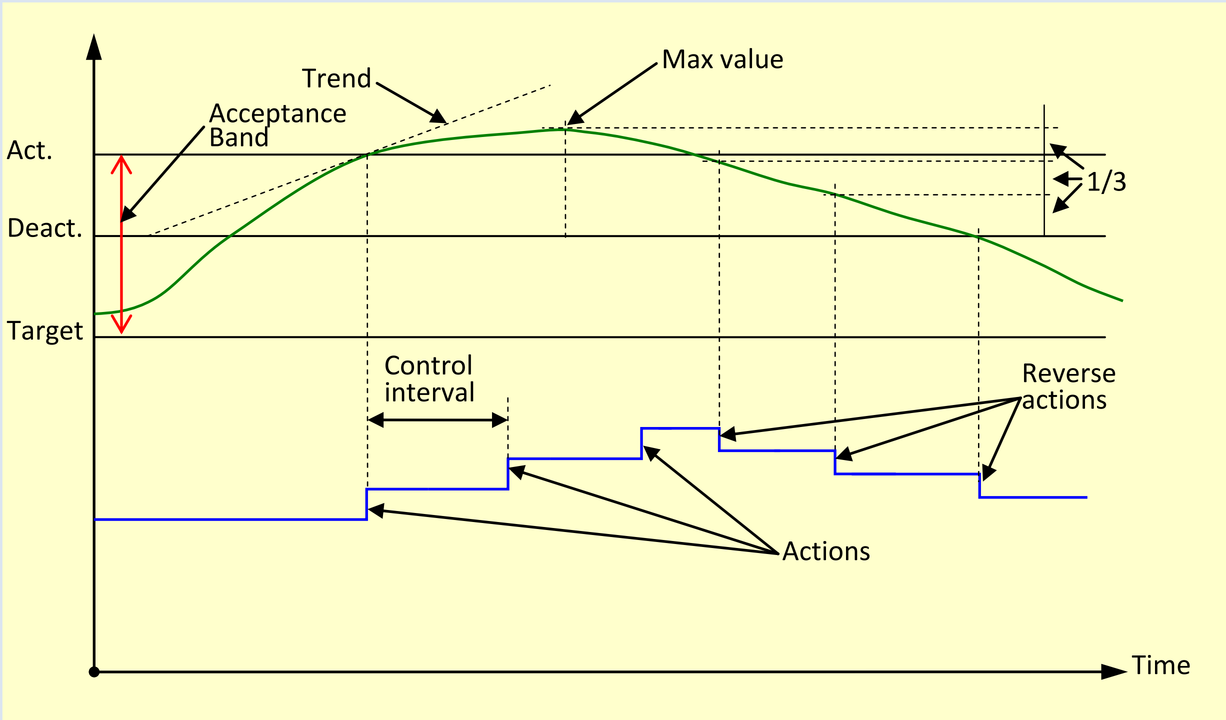


Fig. 44: FECA Stepwise reverse actions

The stepwise deactivation logic works in the following way:

- The maximum value during active state of the EventX is registered automatically.
- The interval between the deactivation logic and the maximum value is divided into three subintervals.
- When the process measurement goes below the first subinterval limit, then one third of the total reverse action is executed.
- When the process measurement goes below the second subinterval, then another one third of the total reverse action is executed.
- Finally, when the process measurement goes below the deactivation value, then the last third of the total reverse action is executed. The advantage of the stepwise reverse actions is that normally it results in more smooth control compared to making one large reverse action.

The advantage of Stepwise reverse actions is that process value over-swings in the opposite directions might be prevented.

Stepwise reverse actions is selected by setting the tick mark named Stepwise reverse actions in the FECA configuration window under Control modes.

12.4.7 Fuzzy activation

The FECA algorithm allows what is called fuzzy activation. Fuzzy activation means that the first control actions start when the process measurement is above the deactivation value as shown in Fig. 45.

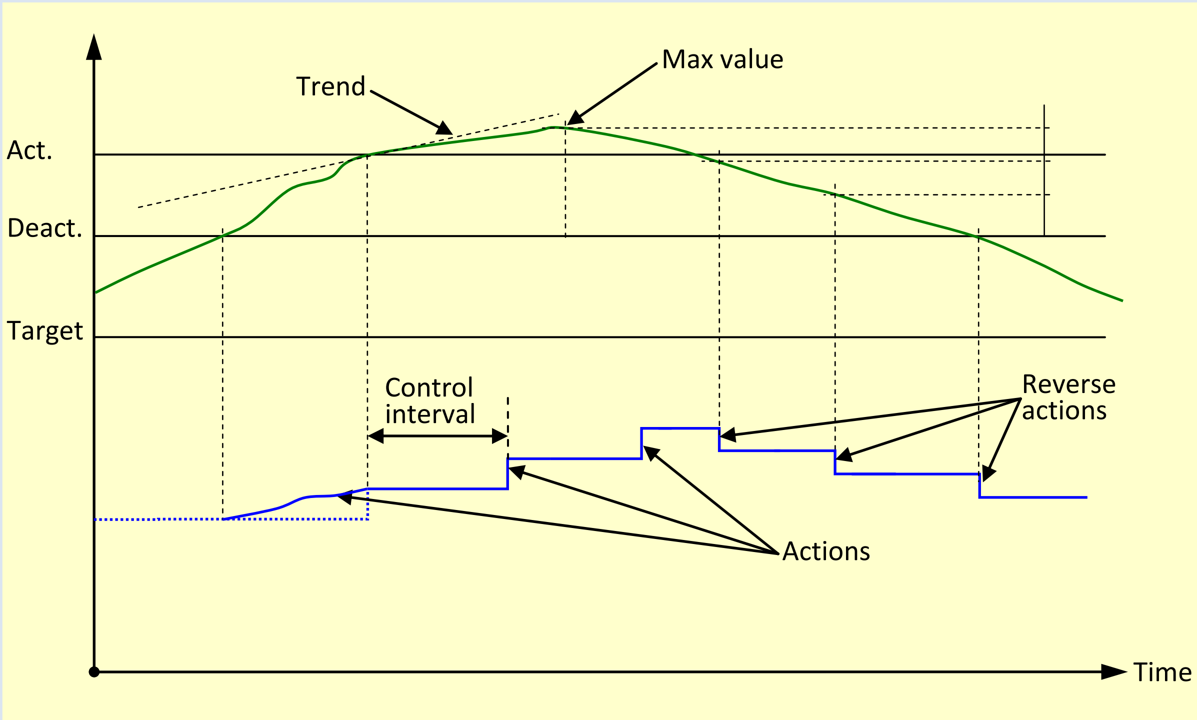


Fig. 45: FECA Fuzzy activation

The first action, in other words will has already been executed when the process measurement reaches the activation value. After the first action, the FECA algorithm works in the same way as the basic algorithm, and the following actions are executed in steps with the specified control interval between the adjustments.The advantage of fuzzy activation is that early small adjustments may prevent the process value from reaching the activation value.Fuzzy activation is selected by setting the tick mark named Fuzzy activation in the FECA configuration window under Control modes.

12.5 FECA scripts in practice

Often the FECA (FuzEvent Control Algorithm) scripts are configured to operate in pairs or in quads for handling the same controlled process output value (e.g. Feeder speed, Grate speed or PA Fan speed, etc.).

If configured for operation in pairs, one FECA script is Type 10: FECA GT, handling process states where the measured process value(s) are larger than the target value or setpoint. The other is Type 11: FECA LT, handling process states where the measured process value(s) are smaller than the target value or setpoint. They both control the same process output variable.

If configured for operation in quads, two FECA scripts are Type 10: FECA GT, handling process states where the measured process value(s) are larger than the target value or setpoint. The others are Type 11: FECA LT, handling process states where the measured process value(s) are smaller than the target value or setpoint. They all control the same process output variable.

Typically, the two type 10 FECA GT scripts will be named “High <Process Value> <Controlled Value>” and “Very high <Process Value> <Controlled Value>” whereas the two type 11 FECA LT scripts will be named “Low <Process Value> <Controlled Value>” and “Very low <Process Value> <Controlled Value>”.

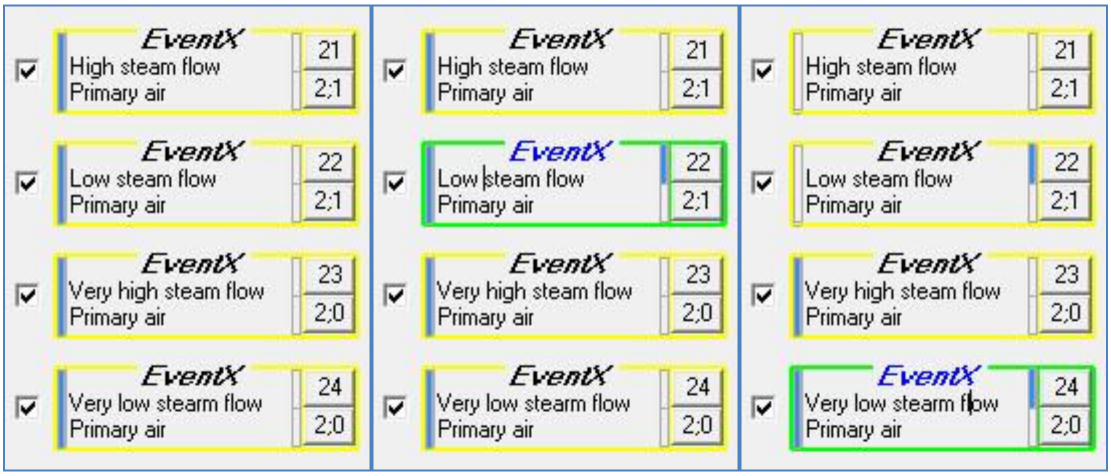


Fig. 46: FECA scripts in quad configuration

The naming of the four FECA scripts in Fig. 46 follows this convention. They are shown in three different situations:

- Left: The measured Steam Flow is within the Activation Limits of EventX 21 and 22.
- Middle: The measured Steam Flow is below the Activation Limit of EventX 22 and this has become active.
- Right: The measured Steam Flow is below the Activation Limit of EventX 24 and this has become active. Due to the priorities defined, that is EventX 21 and 22 have lower priority than EventX 23 and 24, EventX 21 and 22 are forced passive by reducing their EVENTXWEIGHT value to zero.

12.6 EventX type 13 (PID controller)

The PID control algorithm of FuzEvent is:

$$\text{ACTIONVALUE1} = \; Gain \; \times \left(\Delta e + Itime \times e + Dtime(\Delta e - \Delta e_{t-1}) \right)$$

The Gain is the EventX property named “PID proportional gain”, the Itime is the property named “PID integral time”, and Dtime is the property named “PID derivative time”.

12.7 EventX type 19: Raw material proportioning

The raw material proportioning algorithm of FuzEvent, which works in combination with the Excel workbook named FuzProp.

13 The FUEL script language

13.1 FUEL introduction

The FUzzy Event Language (FUEL) is a specialised programming or script language for implementation of high level control strategies based on the theory of fuzzy sets, which was introduced by Professor Lotfi A. Zadeh in the mid nineteen sixties. A FuzEvent control strategy is composed of a mixture of calculation EventX components and control EventX components.

Types of variables

FUEL works with three types of variables, i.e.:

Tag variables

Global variables

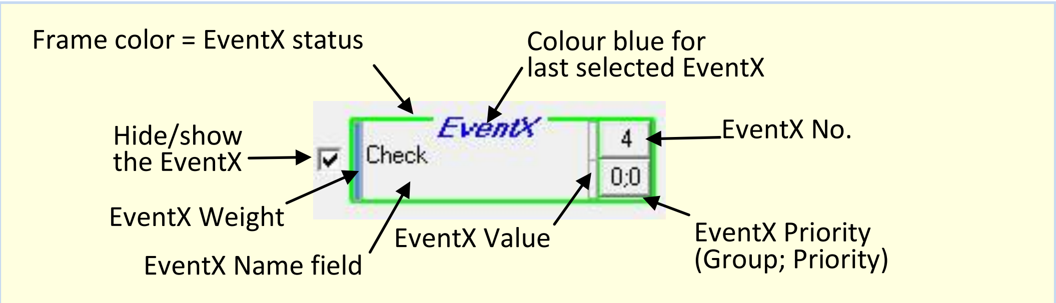
Local variables

Tag variables are used for communication with the container system, i.e. for exchange of process measurements.

Global variables are variables, which may be used for exchange of data between different EventX components within the same FuzEvent application.

Finally, Local variables can only be accessed from the EventX where they are defined.

Variables are defined by right click on the EventX number in the upper right corner of the EventX symbol.



Variable values and constants in FUEL are all real number. The number 1 and 1.0 are thus treated as the same number, and logical values are also represented by real numbers. A condition is true if the value of the condition is 1. If the value is different from 1, then the condition is treated as false.

Arithmetic operators

FUEL holds the following arithmetic operators:

*	Multiplication	(priority 1)
/	Division	(priority 1)
+	Addition	(priority 2)
-	Subtraction	(priority 2)

The priorities are used to determine the order of execution as priority 1 operators are executed before the priority 2 operators, unless parentheses change the order of execution.

```
X1 = SET + FR*DEL
```

First `FR*DEL` is calculated, after which `SET` is added to the result.

Parentheses are used to control the order of execution, e.g.:

```
X1 = (SET + FR)*DEL
```

First `SET` is added to `FR`, after which the sum is multiplied with `DEL`.
Parentheses are also used to increase the script readability.

Logical operators.

FUEL holds the following logical operators:

AND	X1 AND X2	The result is the smallest of X1 and X2
OR	X1 OR X2	The result is the largest of X1 and X2
GT	X1 GT X2	Is equal to 1 if X1>X2, else the result is 0
GE	X1 GE X2	Is equal to 1 if X1>=X2, else the result is 0
LT	X1 LT X2	Is equal to 1 if X1<X2, else the result is 0
LE	X1 LE X2	Is equal to 1 is X1<=X2, else the result is 0
EQ	X1 EQ X2	Is equal to 1 if X1=X2, else the result is 0
NE	X1 NE X2	Is equal to 1 if X1<>X2, else the result is 0

All the logical operators have the same priority.

Examples:

```
DEFT = O2 LT O2_LL
```

`DEFT` is equal to 1 if `O2` is less than `O2_LL`, else `DEFT` is equal to 0.

```
DEFT = (DEFT + 1)*(O2 GE O2_HL)
```

`DEFT` is equal to `DEFT+1` if `O2` is greater than or equal to `O2_HL`,
else `DEFT` is equal to 0

These examples show that in FUEL it is possible to combine arithmetic calculations with logical expressions.

13.2 GOTO

Unconditional GOTO

The syntax is:

```
GOTO(LABNAME)
```

or

```
GOTO LABNAME
```

where LABNAME is the name of a Label, which is defined by:

```
LABEL: LABNAME
```

The unconditional GOTO statement is used to jump to the line, which follows the LABEL statement.

Example:

```
GOTO LAB10      When the GOTO line is executed, the
 ******************     program jumps to the line after the
 **********   LABEL: LAB10 statement.
 ******
LABEL: LAB10
X1=***
```

Conditional GOTO

The syntax is:

```
IF <condition> THEN GOTO(LABNAME)
```

or

```
IF <condition> THEN GOTO LABNAME
```

where LABNAME is the name of a Label, which is defined by:

```
LABEL: LABNAME
```

If the <condition> results in the value 1, then the next line to execute will be the line following the line with the Label. If the <condition> results in any other value than 1, then the next that will be executed is the line immediately after the conditional GOTO. The <condition> may be a logical expression, an arithmetic calculation or even a combination of logical and arithmetic expressions.

Example:

```
IF O2 LT O2L THEN GOTO LO2 If O2 is less than O2L, then
 *********** execution will jump to the line
 **** following the Label LO2:
LABEL: LO2
****
```

13.3 IF THEN / END IF

Conditional assignment

The syntax is:

```
IF <condition> THEN Var1=<Expression>
```

If <condition> is equal to 1 then the variable Var1 is set equal to <Expression>, which may be an arithmetic or a logic expression. If <condition> is different from 1, then the value of Var1 is not changed by this statement.

IF-THEN/END IF

The syntax is:

```
IF <condition> THEN
    ******
    ******
    ******
END IF
```

If <condition> is equal to 1 then, then the statements between IF and END IF are executed. If, however, the value of <condition> is different from 1 then statements between IF and END IF are not executed.

It is possible to have conditional assignment statements nested in an IF-THEN/END IF block, and it is possible to have other IF-THEN/END IF blocks nested inside each other, i.e.:

```
IF <condition1> THEN
    IF <condition2> THEN
        Line1
        Line2
    END IF
    Line3
    Line4
END IF
```

If <condition1> is equal to 1 and <condition2> is equal to 1 then statements Line1 and Line 2 are executed. If <condition1> is equal to 1 and <condition2> is different from 1 then only Line3 and Line4 are executed. If both <condition1> and <condition2> are different from 1 then Line1, Line2, Line3 and Line4 are not executed.

When programming the EventX scripts, it is easy to mark the nested statements in an IF-THEN/END IF block by using the Tab-character, as in the example above.

13.4 Fuzzy IF-THEN/END IF

Fuzzy IF-THEN

The syntax is:

```
IF <condition> THEN Var1=FUZZY(Singleton)
```

Where the argument of the function FUZZY is a so-called singleton.

The calculations involved in the execution of fuzzy rules are explained in details in section “Fuzzy rules”. At this point, the following is emphasized:

A fuzzy output variable is not automatically reset to 0. A fuzzy output variable is the variable following the THEN in a fuzzy control rule. This means the FUEL code must reset the variable to 0, if previous results and other EventX should not influence the result.

Example:

```
DAIR = 0
    IF HIGH(O2) THEN DAIR = FUZZY(1.5)
    IF   OK(O2) THEN D ********
```

If DAIR was not reset to 0, then DAIR from the previous calculations would have been combined with all the future evaluations of this set of rules. This feature may be used to build rule bases in different EventX instances, which work together.

In case of nested fuzzy rules it is necessary to specify that the condition should be treated as a fuzzy condition. This is done by setting the system variable FUZZYON equal to 1.

Example:

```
FUZZYON = 1
    IF HIGH(FuzO2) THEN
        IF NOT(HIGH(FuzTemp)) THEN
        DAIR = FUZZY(1)
        END IF
    END IF
```

In this example, the statement FUZZYON = 1 ensures the nested rules will be evaluated even if FuzO2 is not high in degree 1. If FUZZYON = 1 was not included, then the nested rule

IF NOT(HIGH(FuzTemp)) THEN DAIR = FUZZY(1)

would only have been executed if HIGH(FuzO2) had the value 1. For details about evaluation of fuzzy rules, please refer to section "Fuzzy rules".

13.5 Standard membership functions

FUEL includes standard membership functions of the shoulder type, which all are defined on the interval from –1 to +1. The figure below shows the membership functions, and Table below the functions gives the parameter values, as each function is defined by four parameters P1, P2, P3, and P4, where P1 is the leftmost parameter.

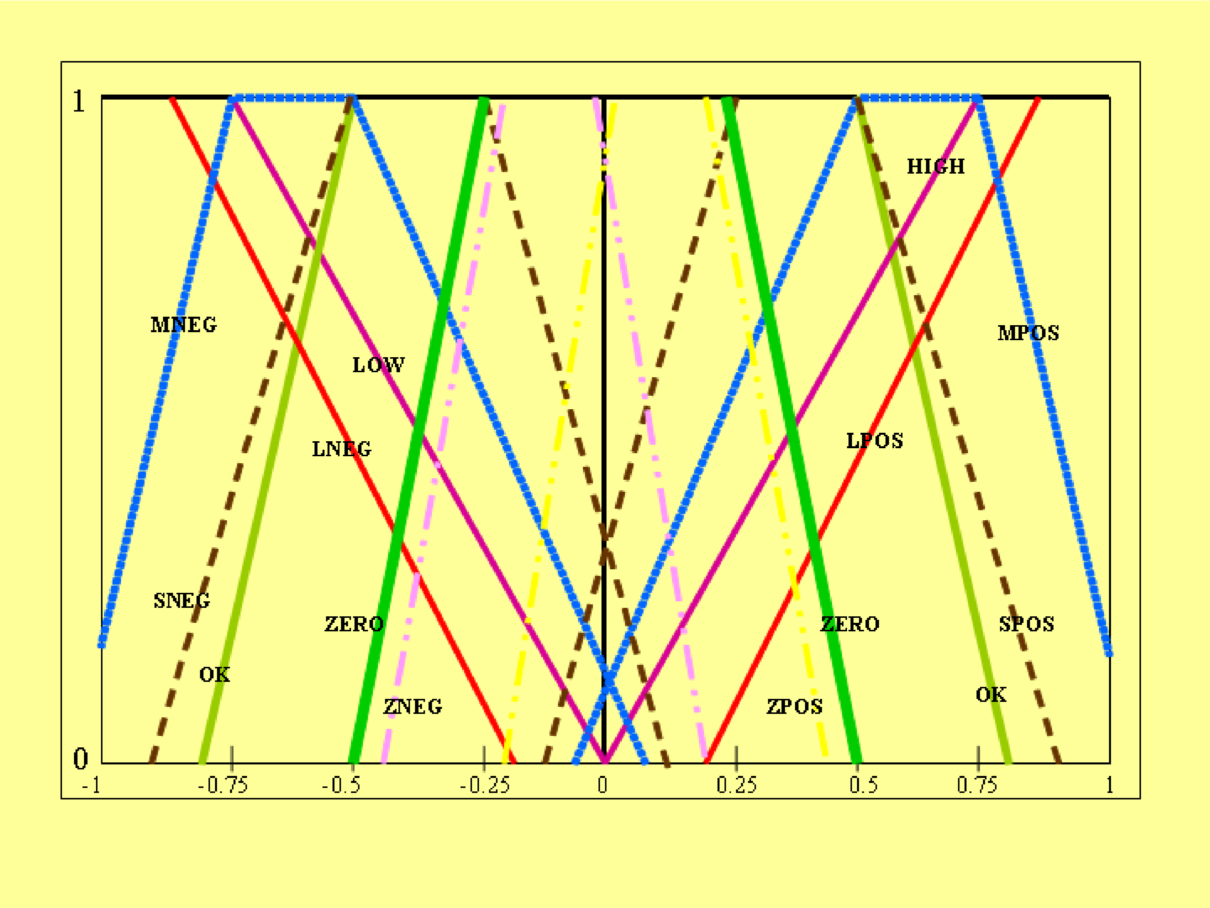


Fig. 47: Standard membership functions

The exact values of the four parameters P1, P2, P3, and P4 are shown the table Standard membership functions in Fig. 48

Standard membership functions					
Name	Meaning	P1	P2	P3	P4
LOW	Low	-99999	-999	-0.75	0.0
OK	Ok	-0.8	-0.5	0.5	0.8
HIGH	High	0.0	0.75	999	99999
LNEG	Large negative	-99999	-999	-0.85	-0.2
MNEG	Medium negative	-1.05	-0.75	-0.5	0.05
SNEG	Small negative	-0.9	-0.5	-0.25	0.15
ZNEG	Zero negative	-0.45	-0.2	-0.05	0.2
ZERO	Zero	-0.5	-0.25	0.25	0.5
ZPOS	Zero positive	-0.2	0.05	0.2	0.45
SPOS	Small positive	-0.15	0.25	0.5	0.9
MPOS	Medium positive	-0.05	0.5	0.75	1.05
LPOS	Large positive	0.2	0.85	999	99999

Fig. 48: Table of Standard membership functions

13.6 Fuzzy control rules

The following examples illustrate how fuzzy control rules are evaluated in FuzEvent. A fuzzy control rules has the syntax:

```
IF <condition> THEN OUTVAR=FUZZY(VAR)
```

where <condition> is an expression, which is composed of membership functions and fuzzy logic operators. OUTVAR is the fuzzy output variable and FUZZY(VAR) is a so-called singleton at the position VAR. An example of a fuzzy control rule is:

```
IF HIGH(INCR) AND LOW(TEMP) THEN DAIR = FUZZY(2.3)
```

HIGH(INCR) AND LOW(TEMP) results in a degree of fulfilment DFUL, which is a number between 0 and 1 that expresses to what degree INCR is high and TEMP is low. The contribution to DAIR from the above mentioned rule is a singleton at position 2.3 with the height equal to DFUL.

Simple set of fuzzy rules

The following is a simple set of fuzzy rules:

```
0.88IF HIGH(INCR) THEN DAIR = FUZZY(2.3)
0.32IF   OK(INCR) THEN DAIR = FUZZY(0)
0.0 IF  LOW(INCR) THEN DAIR = FUZZY(-1.8)
```

The numbers to the left of the rules show the degree of fulfilment. The first rule, in other words, is fulfilled in the degree 0.88, which means that INCR is high in the degree 0.88. INCR is ok in the degree 0.32, and it is low in the degree 0.0. The result of the first rule is a singleton at position 2.3, and with a height on 0.88. The second rule is a singleton at position 0.0 with the height 0.32, and the last rule results in a singleton at –1.8, but with the height 0.

The combination of the three rules is calculated as a sort of weighted average of the individual results, i.e.

```
DAIR = (0.88*2.3 + 0.32*0)/(0.88 + 0.32)  = 1.69
DAIR = (MaxDegree*DAIR + 0.0*(-1.8))/(MaxDegree + 0.0)  = 1.69
```

Where MaxDegree is the largest degree of fulfilment of the previous rules.

DAIR, in other words, is a compromise between the first and the second rule, where rule No. 1 will have a larger influence then rule No. 2, because rule No. 1 has a higher degree of fulfilment than rule No. 2.

Nested set of fuzzy rules

The following is an example of so-called nested fuzzy rules:

```
     FUZZYON = 1
0.4  IF HIGH(X1) THEN
1.0      IF NOT(LOW(X2)) THEN
0.4      DAIR = FUZZY(1)
         END IF
     END IF
1.0  IF OK(X1) THEN
0.7      IF LOW(X3) THEN DAIR = FUZZY(0.7)
     END IF
     IF LOW(X4) THEN DAIR = FUZZY(-0.1)
0.667    W = DAIR
```

The line values show:

X1 is high in the degree 0.4
X2 is not low in the degree 1.0

FUZZY(1) will have a weight of 0.4, which is the degree of fulfilment of the first set of nested rules.

X1 is OK in the degree 1.0
X3 is low in the degree 0.7
X4 is low in the degree 0.13.

Note! `FUZZYON = 1` is inserted to mark that the following IF-THEN rules are to be treated as fuzzy rules. This means that even if the condition is different from 1, then the statements after the IF will be executed.

The rules are combined in the following way:

```
     DAIR = (0.4*1 + 0.7*0.7)/(0.4 + 0.7) = 0.809
     DAIR = (MaxDegree*0.809 + 0.13*(-0.1))/(MaxDegree + 0.13)
          = (0.7*0.809 + 0.13*(-0.1))/(0.7 + 0.13) = 0.667
```

14 The FUEL Functions

14.1 ABS

ABS(var)

This function returns the absolute value of the variable var.

Example:

```
l_AVR = -23
l_ABVAR = ABS(l_VAR)
```

The value of l_ABVAR will be 23.

14.2 CLOSE

CLOSE

The syntax is:

```
CLOSE(”WorkbookName”,”WorksheetName”,mode)
```

This function is used close the referenced Excel Workbook and Worksheet. The mode parameter can have one of the following three values:

0 The workbook is closed

1 The workbook is saved with the name “WorkbookName”_ ddmmyy_hhmmss.xls before it is closed

2 The workbook is saved with the name “WorkbookName”_ ddmmyy_hhmmss before.xls it is closed, and a new empty workbook is opened with the name “WorkbookName”.xls

Example:

CLOSE(“FuzPropReport”,”SHEET1”,2)

This CLOSE statement saves the worksheet ”SHEET1”, of the workbook “FuzPropReport” in a file with the name FuzPropReport_ddmmyy_hhmmss.xls, after which it opens an empty workbook/worksheet with the name “FuzPropReport”.xls/”SHEET1”

14.3 DAY

DAY

The syntax is:

```
Var = DAY(TIMER)
```

This function returns the current day, e.g. 4 for Wednesday.

14.4 EVAL

EVAL(Var)

This function returns the EVENTVALUE of the EventX with the number var.

Example:

```
g_EVVAL = EVAL(4)
```

The value of the Global variable g_AVVAL is set equal to the EVENTVALUE of EventX No. 4.

14.5 EVENTX

EVENTX

The syntax is:

```
Var = EVENTX(XNo,PropName)
```

This function returns the current value of PropName for the EventX component with the number XNo.

Example:

```
l_EVAL = EVENTX(6,EVENTVALUE)
```

The value of l_EVAL will be equal to the EVENTVALUE of EventX No. 6

```
l_CINTRV = EVENTX(21,CONTROLINTERVAL)
```

The value of l_ CINTRV will be equal to the CONTROLINTERVAL of EventX No. 21

14.6 EXECUTE_TYPE

EXECUTE_TYPE

The syntax is:

```
EXECUTE_TYPE
```

The EventX property named “EventX type” refers to a pre-programmed control algorithm, which is executed from the EventX script by the statement `EXECUTE_TYPE`

Normally the structure for application of one of the EventX type algorithms is:

- Prepare input variables for the control algorithm
- Execute the `EXECUTE_TYPE` statement
- Perhaps evaluate the results of the control algorithm. The results are returned from the control algorithm in the properties `ACTIONVALUE1`, `ACTIONVALUE2`, etc.

The EventX type property is defined in the browser. The default EventX type is 0, which does not refer to any pre-programmed algorithm.

The pre-programmed control algorithms are described in section 12.1 EventX type algorithms.

14.7 FUZZY

FUZZY(Var)

Fuzzy singletons are used as control actions, i.e. following the `THEN` statement of a fuzzy control rule. A Singleton is a special type of membership functions with a value different from 0 in only one point.

The syntax is:

`FUZZY(VAR)` which creates a singleton at the position `VAR`.

`FUZZY(0.7)` will create a Singleton at position 0.7 as shown in Fig. 49.

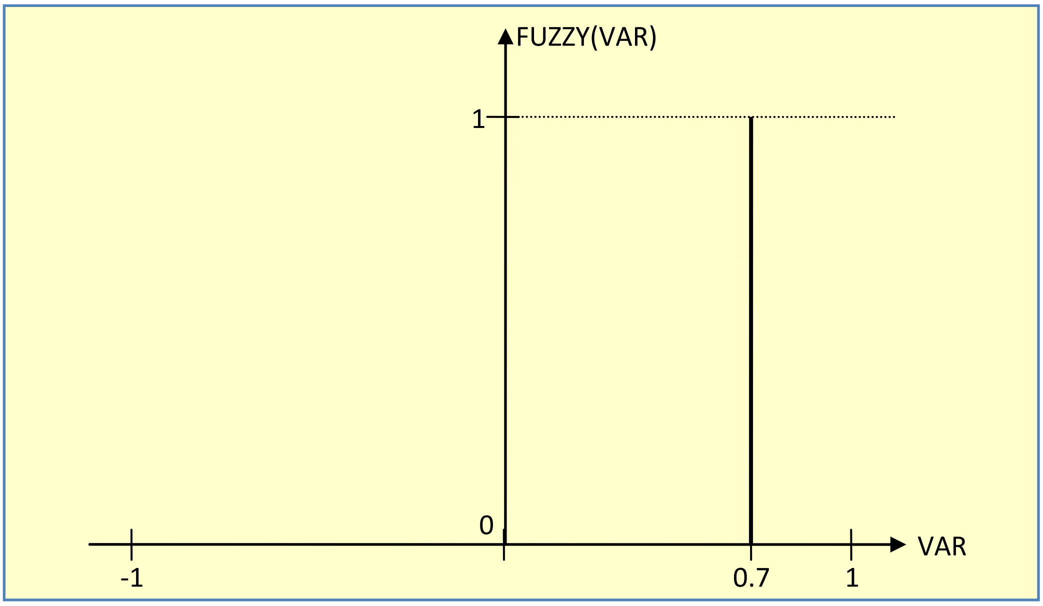


Fig. 49: Fuzzy singleton creation using FUZZY(VAR)

Example:

```
IF  LOW(l_FUZO2SP) THEN l_DO2SP=FUZZY(0)
IF MNEG(l_FUZO2SP) THEN l_DO2SP=FUZZY(0)
IF SNEG(l_FUZO2SP) THEN l_DO2SP=FUZZY(0)
IF ZERO(l_FUZO2SP) THEN l_DO2SP=FUZZY(0)
IF SPOS(l_FUZO2SP) THEN l_DO2SP=FUZZY(-0.3)
IF MPOS(l_FUZO2SP) THEN l_DO2SP=FUZZY(-0.6)
IF HIGH(l_FUZO2SP) THEN l_DO2SP=FUZZY(-1)
```

14.8 FUZZYFY

FUZZYFY (Definition of membership function and fuzzyfication)

The syntax is:

```
varFZ = FUZZYFY(var,P1,P2,P3,P4)
```

This function transforms a variable into the interval from –1 to +1, using the membership function of the shoulder type, which is defined by the four function parameters as shown in Fig. 50.

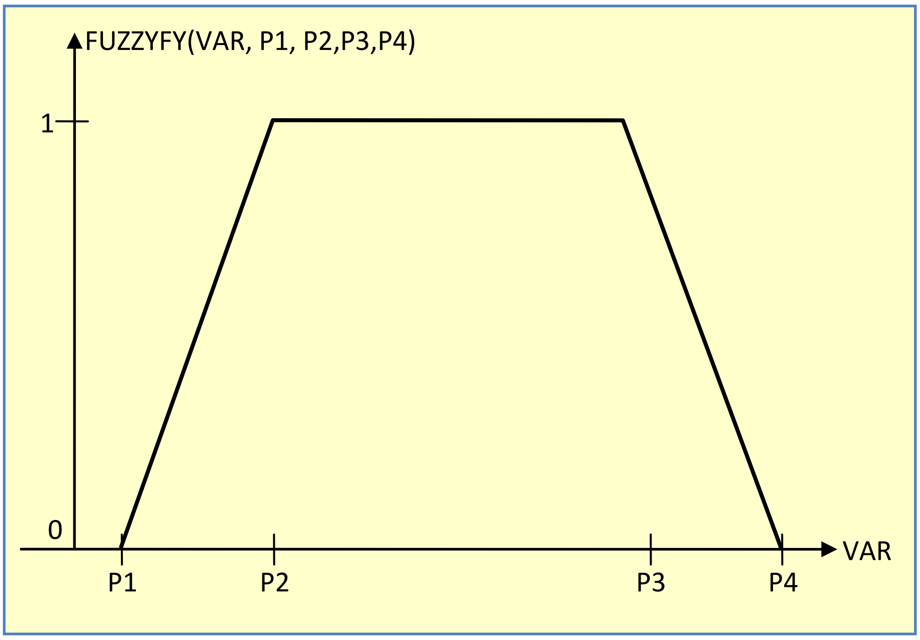


Fig. 50: Fuzzy membership creation using FUZZYFY

Example:

```
g_FZTQETREND = FUZZYFY(l_TQETREND,-10,-8,8,10)
```

14.9 HOUR

HOUR

The syntax is

```
Var = HOUR(TIMER)
```

This function returns the current hour, e.g. 10 for 10:33:10

14.10 LIMIT_CHECK

LIMIT_CHECK

The syntax is:

```
VAR = LIMIT_CHECK(TAG_SP,ACT,LL,HL)
```

Where TAG_SP is the Tag (Global or Local), which holds the set point for which the limits apply. ACT is the change that will be added to TAG_SP, and LL and HL are the low limit and the high limit respectively. The value of VAR is either equal to ACT or it is equals the value by which the sum of TAG_SP and VAR is equal to the low limit or the high limit. In other words TAG_SP + VAR is not above the high limit or below the low limit.

Example:

```
ACTIONVALUE1=LIMIT_CHECK(t_GRATESPEED, ACTIONVALUE1,g_LL,g_HL)
t_GRATESPEED = t_GRATESPEED + ACTIONVALUE1
```

This ensures that t_GRATESPEED is in the interval [g_LL,g_HL].

14.11 LOG_ACTIONS

LOG_ACTIONS

The syntax is:

```
LOG_ACTIONS("Text",var1,var2)
```

This function is used to insert a message in the “Message list” of the Browser. Typically this function is used to insert a message about a control action, where “Text” explains about the action and var1 is the old value of the control point, and var2 is the new value

Example:

```
LOG_ACTIONS("Change of kiln feed",g_FEED_OLD,g_FEED_NEW)
```

This message is display in the “Message list” of the Browser in the following way:

showing date, time, FuzEvent application No., EventX No., and the LOG_ACTIONS parameters.

14.12 LPOS, MPOS, SPOS etc.

Membership functions

The standard membership functions in FUEL are described in section 13.5 Standard membership functions.

The syntax is

```
LPOS(var), MPOS(var), SPOS(var), ZPOS(var), ZERO(var), ZNEG(var),
SNEG(var), MNEG(var), LNEG(var), HIGH(var), OK(var), LOW(var)
```

The argument var has a value between -1 and +1, which is the definition interval for the standard membership functions. Normally, a variable is first fuzzyfied, i.e. transformed into the interval from [-1,1] by using the SCALE function

Example:

```
IF  LOW(l_FUZO2SP) THEN l_DO2SP=FUZZY(0)
IF MNEG(l_FUZO2SP) THEN l_DO2SP=FUZZY(0)
IF SNEG(l_FUZO2SP) THEN l_DO2SP=FUZZY(0)
IF ZERO(l_FUZO2SP) THEN l_DO2SP=FUZZY(0)
IF SPOS(l_FUZO2SP) THEN l_DO2SP=FUZZY(-0.3)
IF MPOS(l_FUZO2SP) THEN l_DO2SP=FUZZY(-0.6)
IF HIGH(l_FUZO2SP) THEN l_DO2SP=FUZZY(-1)
```

14.13 MINUTE

MINUTE

The syntax is

```
Var = MINUTE(TIMER)
```

This function returns the current minute, e.g. 33 for 10:33:15

14.14 MONTH

MONTH

The syntax is

```
Var = MONTH(TIMER)
```

This function returns the current month, e.g. 3 for March.

14.15 OPEN

OPEN

The syntax is:

```
OPEN(”WorkbookName”,”WorksheetName”)
```

This function is used open the referenced Excel Workbook and Worksheet.

Example:

```
OPEN(l_FUZPROPREPORT,l_SHEET1)
```

This OPEN statement opens the worksheet ”SHEET1”,of the workbook “FuzPropReport”.

14.16 PWR

PWR

The syntax of the power function is

```
X1 PWR N
```

which calculates X1 in the power of N.

Example:

```
X1 PWR 0.5
```

Calculates the square root of X1, i.e. √X1

14.17 RETRIEVE

RETRIEVE

The syntax is:

```
RETRIEVE(”WorkbookName”,”WorksheetName”,row,column,var)
```

This function is used to retrieve the value(s) from the row,column(s) of the refrenced Excel workbook and worksheet, and assign the retrieved value to the variable var. The referenced workbook must be loaded, which is done from the Excel menu item on the application window. If the variable is an array, then all the array elements are retrieved from column, column+1, column+2 etc.

Example:

```
RETRIEVE(“FUZPROP”,”DATA”,3,11,g_LSF_M)
```

This RETRIEVE statement retrieves the value from row 3, column 11 of the worksheet named “DATA” in the workbook named “FUZPROP”, and assigns the retrieved value to the Global variable g_LSF_M .

14.18 RND

RND

The syntax is:

```
var = RND(i)
```

This function generates a random number in the interval from 0 to 1. The parameter i is an initial value in the random number sequence. A typical value for i is 1.

14.19 RUN

RUN

The syntax is:

```
RUN(EVENTXNO)
```

where EVENTXNO is the number of the EventX that should execute. The RUN statement is typically used in connection with an EventX that has the ScanTime property equal to 0. Another EventX may schedule one execution of this EventX.

Example:

```
RUN(5)
```

If this statement is included in e.g. EventX No. 11, and EventX No. 5 has ScanTime equal to 0, then EventX No. 11 makes EventX No. 5 run once every time the RUN(5) statement is executed in EventX No. 11

14.20 SCALE

SCALE(var,varLL,varSP,varHL)

The syntax is:

```
varFZ = SCALE(var,varLL,varSP,varHL)
```

This function transforms the variable var into the interval from -1 to +1. The operation is also named fuzzyfication of the variable var. The transformation interval is defined by the arguments varLL, varSP, and varHL, and the transformation is shown in the figure below. It is paramount that varLL ≤ varSP ≤ varHL.

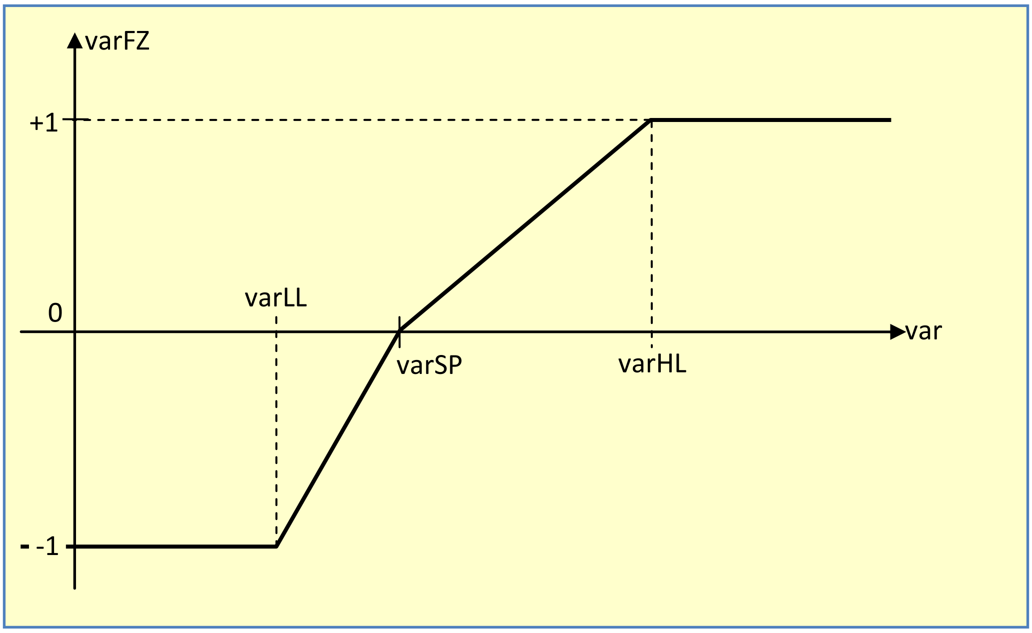


Fig. 51: SCALE function

Example:

```
g_FZTQETREND = SCALE(l_TQETREND,g_TQETREND_LL,0,g_TQETREND_HL)
```

14.21 SECOND

SECOND

The syntax is:

```
Var = SECOND(TIMER)
```

This function returns the current second, e.g. 15 for 10:33:15

14.22 SHIFTUP

SHIFTUP(Avar(e),var)

This function shifts array elements. The argument Avar(e) refers to the array element that will be lost, and the value of var is put into the array element with the highest element No.

The syntax is:

```
SHIFTUP(Avar(e),var)
```

Example:

```
SHIFTUP(l_TEMPARRAY(1),t_COOLERTEMP)
```

Suppose the array l_TEMPARRAY has 6 elements, i.e.:

Element	Value
1.	123.2
2.	133.2
3.	112.0
4.	110.3
5.	115.1
6.	120.1

If the value of t_COOLERTEMP is 113.7, then the array values after the SHIFTUP operation are:

Element	Value
1.	133.2
2.	112.0
3.	110.3
4.	115.1
5.	120.1
6.	113.7

Note! It is possible to shift a part of an array by specifying a different element No. than 1.

14.23 SHOW_VALUE

SHOW_VALUE

The syntax is:

```
SHOW_VALUE(”Text”,var,Line,Page)
```

This function is used to display the value of a variable on the EventX property page, which is display by click on the EventX name of the EventX symbol on the application window, i.e.

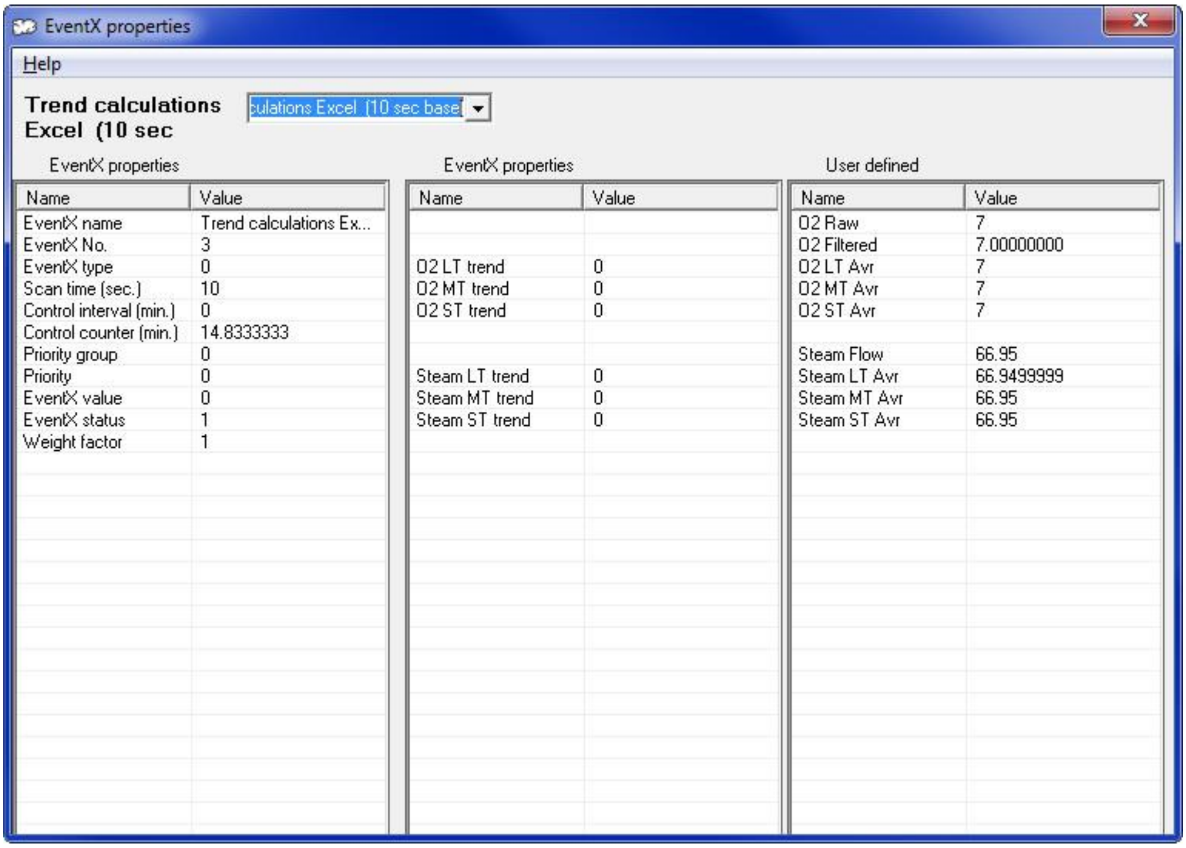


Fig. 52: Real time display of values using SHOW_VALUE

The display window is divided into three pages. Page No. 1 is the leftmost page. The Page parameter may have the value 2 or 3, whereas the Line parameter may have a value between 1 and 28, corresponding to one of the 28 lines on each page.

Example:

```
SHOW_VALUE("Steam ST Avr",g_STEAM_ST_AVR,10,3)
```

This SHOW_VALUE displays the text “Steam ST Avr”, and the value of g_STEAM_ST_AVR in the 10th line of page No. 3, as shown in Fig. 52.

14.24 START

START

The syntax is:

```
START(EVENTXNO)
```

where EVENTXNO is the number of the EventX that will be started.

14.25 STOP

STOP

The syntax is:

```
STOP(EVENTXNO)
```

where EVENTXNO is the number of the EventX that will be stopped.

14.26 STORE

STORE

The syntax is:

```
STORE("WorkbookName","WorksheetName",row,column,var,mode)
```

This function is used to store the value(s) of the variable var in the row,column(s) of the referenced Excel workbook and worksheet. The referenced workbook must be loaded, which is done from the Excel menu item on the application window. If the variable is an array, then all the array elements are stored in column, column+1, column+2 etc.

The mode parameter can have the value 0 or 1. If mode is 0, the value(s) is stored in row,column. If mode is 1, then data and time are automatically added in front of the value(s) in column and column+1, and the value(s) is stored in column+2.

The STORE statement may be used to trigger a calculation, which has been programmed in VBA of Excel. The calculation is triggered from FuzEvent by storing a value in a certain cell, which activates e.g. a sub-program of the type:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
```

Example:

```
STORE("FuzProp","DATA",6,4,g_SAND_LL,0)
```

This STORE statement stores the value of the global variable g_SAND_LL in row 6, column 4 of the worksheet named "DATA" in the workbook named FuzProp.

14.27 TIMER

TIMER

TIMER is a system variable that holds the current date and time.

TIMER is used as argument in the functions YEAR, MONTH, DAY, HOUR, MINUTE and SECOND.

14.28 TREND

TREND(avar(ElNo),NOVAL)

This function is used to calculate the trend of a variable from data, which are stored in an array. The syntax is:

```
TREND(AVAR(ElNo),NOVAL)
```

where AVAR(ElNo) is the first array element, which enters the trend calculation, and NOVAL is the number of elements from the array that will be used in the trend calculation. The array AVAR must be organized so that the newest value is the last element of the array. The function SHIFTUP will maintain an array so that it can be used directly in the trend calculation.

The trend calculation is done by approximating the real signal, which is represented by the data in the array, by a regression line. The time interval is thus determined by the time interval between storage of data in the array, and the number of elements in the array. The trend calculation is illustrated in Fig. 53

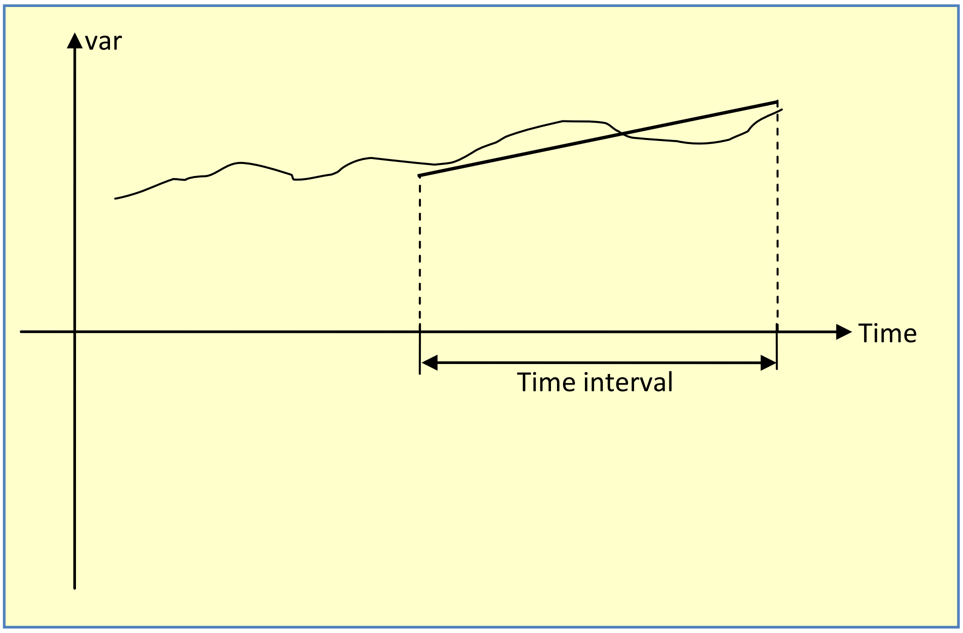


Fig. 53: TREND function

Often long term trends are calculated using an external Excel spreadsheet to limit array sizes.

14.29 YEAR

YEAR

The syntax is:

```
Var = YEAR(TIMER)
```

This function returns the current year, e.g. 2012

15 FuzEvent used in a WtE plant, example and breakdown

The following example is derived from a Waste to Energy plant with two combustion lines burning household waste. Each boiler has a nominal steam capacity of 13 tonnes per hour. On this plant, FuzEvent is installed on a separate desktop computer with its own screen, keyboard and mouse.

FuzEvent controls:

- The feeding of waste into the furnace by controlling the frequency of the feeder pusher strokes.
- The primary- and secondary air flows by controlling the speed of the primary- and secondary air fans.

FuzEvent controls both combustion lines and therefore consists of two applications, named Oven 1 and Oven 2.

The implementation is done so that the existing Scada/DCS system can take over these controls from FuzEvent at any given time, e.g. during start-up or shut down.

The FuzEvent operator screen for one combustion lines is shown below:

Fig. 54: FuzEvent operator screen for one line

15.1 FuzEvent application overwiew

The EventX scripts in an application are divided into groups: Auxiliary functions, Feeder speed control tasks and Primary air / Secondary air control tasks.

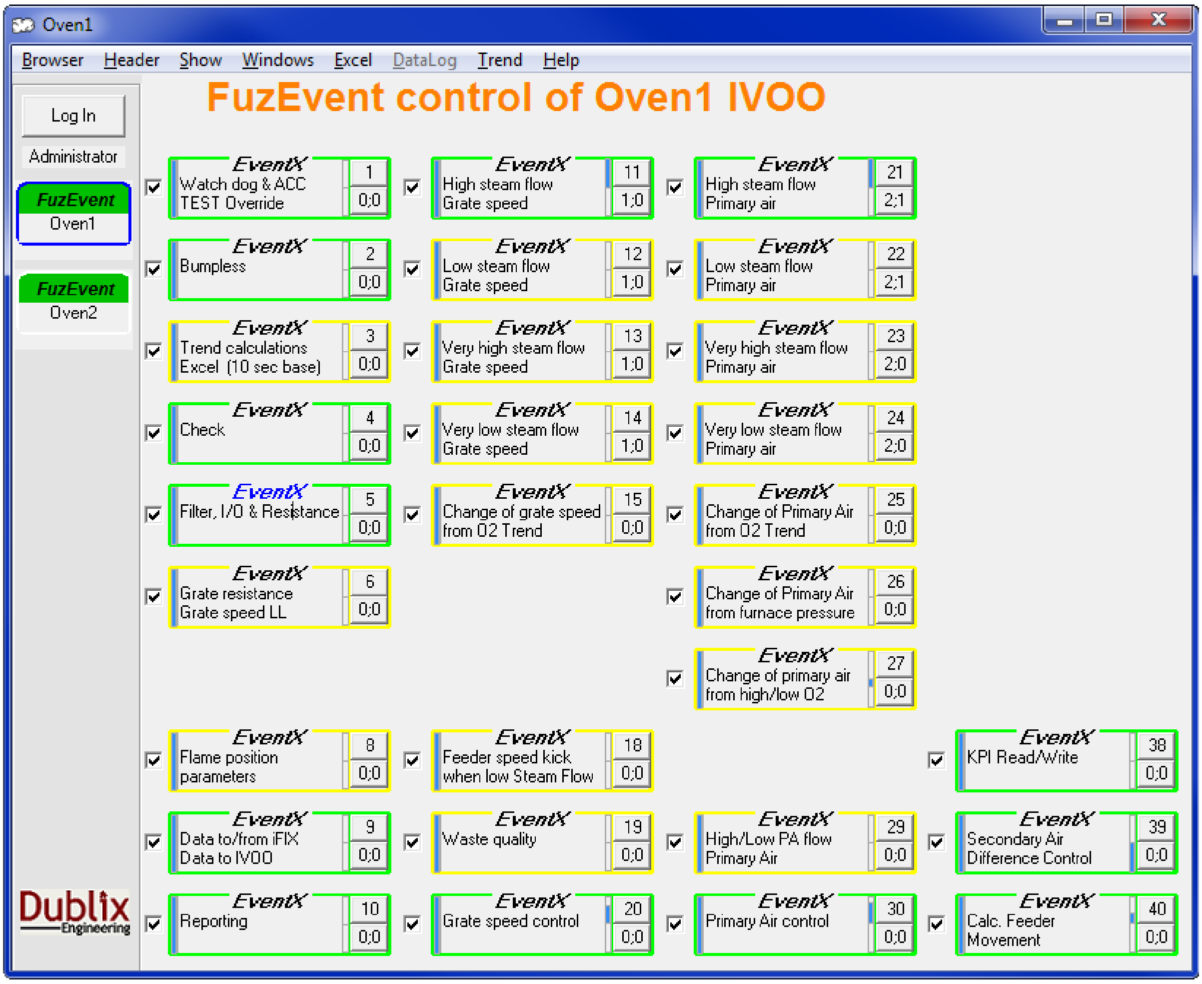


Fig. 55: Example of a FuzEvent application

15.2 EventX scripts for auxiliary functions

In the example in Fig. 55 the EventX scripts numbered 1 to 10 and 38 are auxiliary functions performing the following tasks:

EventX	Naming	Task
1 Type 1	Watchdog & ACC TEST Override	Communication supervision and Accepted Computer Control.
2 Type 1	Bumpless	Handling transfer of control between FuzEvent and the plant DCS
3 Type 0	Trend Calculations Excel (10 sec Base)	Calculating trends and average values of measured (input) or calculated values.
4 Type 1	Check	Switching controllers in DCS between Auto and Manual modes
5 Type 0	Filter, I/O & Re- sistance	Reading measured values from DCS TAGS, filtering values and writing them to Global variables. Calculating

EventX	Naming	Task
		the grates resistances.
6 Type 0	Grate resistance Grate speed LL	Calculating offsets to feeder speed and PA fan from Grate resistance. The purposes are to protect the furnace from waste over feeding, and to protect the grate from exposure to high temperature when the waste layers are thin.
8 Type 0	Flame position parameters	Calculating offsets to feeder speed and PA fan from operator indicated flame position. The purpose is to slow down the grate or to redirect air when the flame front is too far forward, i.e. there is risk of un-burnt waste falling into the slag chute.
9 Type 1	Data to/from iFIX Data to IVOO	Reading data from and writing data to the iFIX HMI. Writing auxiliary data to the plant DCS
10 Type 0	Reporting	Sampling of data, calculating averages and collect them to day reports.
38 Type 1	KPI	Calculation of long term Key Performance Indicators for presentation in the iFIX HMI.

15.2.1 EventX 1: Watch dog & ACC, TEST Override

Every 5 seconds the watch dog counter is increased by 1. See Fig. 56. When the counter reaches 1000, the value is reset to 0. The watch dog counter value is sent to the Tag named KOMMUNIKATIE_FUZEVENT in the SRV_IVOO-PDB of SRVSUB01.
Every 30 seconds, a VBA script in SRVSUB01 checks that the value of FZ_WATCHDOG has changed. If the value has not changed, an alarm is generated, and FuzEvent is switched OFF automatically.

The signals `t_FZ_MASTER`, `t_FZ_FEEDER` and `t_FZ_AIR` coming from SRVSUB01 are mirrored back as `t_FZ_MASTER_ACC`, `t_FZ_FEEDER_ACC` and `t_FZ_AIR_ACC`, where ACC stands for Accepted Computer Control. Conditions for switching to FuzEvent operation can be programmed here.

```
'   *** ACC signals to IVOO Scada ***
'   If conditions for ACC's exist, insert them here
'
t_OV1_FZ_MASTER_ACC=t_OV1_FZ_MASTER
t_OV1_FZ_FEEDER_ACC=t_OV1_FZ_FEEDER
t_OV1_FZ_AIR_ACC=t_OV1_FZ_AIR
't_OV1_FZ_WATER_ACC=t_OV1_FZ_WATER
```

Note! Only the Oven 1 application sends a watch dog signal to SRVSUB01.

To test the open loop performance of FuzEvent, EventX 1 gives access to three global tags: `g_TEST_GRATE` (**#TEST Fz Feeder 0/1**), `g_TEST_AIR` (**#TEST Fz Air 0/1**) and `g_TEST_WATER` (**#TEST Fz Water 0/1**). These ***MUST*** be set to zero during normal operation.

User defined

Name	Value
Watchdog	611
FZ Master f_Scada	1
Master ACC >Scada	1
FZ Feeder f_Scada	1
Feeder ACC >Scada	1
FZ Air f_Scada	1
Air ACC >Scada	1
FZ Water f_Scada	0
Water ACC >Scada	0
#TEST Fz Feeder 0/1	0
#TEST Fz Air 0/1	0
#TEST Fz Water 0/1	0

Fig. 56: EventX 1 properties

15.2.2 EventX 2: Bumpless

This script ensures bumpless transfer from non-FuzEvent control to FuzEvent control, which means that FuzEvent starts with the current values for waste feeder speed, primary air and secondary air controller outputs, which were present just before the operator switched FuzEvent ON.

User defined

Name	Value
Fz Master f_Scada	1
Fz Feeder f_Scada	1
Fz Air f_Scada	1
Fz Water f_Scada	0
#PA Fan Min by OFF	15
#SA Fan Min by OFF	25

Fig. 57: EventX 2 properties

The two parameters, **#PA Fan Min by OFF** and **#SA Fan Min by OFF** are the minimum speeds assigned to the PA- and SA fan speed outputs when switching from DCS to FuzEvent control.

15.2.3 EventX 3: Trend calculations Excel (10 sec base)

Excel workbooks are running in the back ground, which are used for average and trend calculations. The name of the work books are Trend_Calcul1.xls for Oven 1 and Trend_Calcul2.xls for Oven 2.

The work books are stored in the folder C:\Program Files\data\FuzEvent_V4\FuzEvent0.

Average and trend calculations are made for the following signals:

- O2
- Steam flow
- Grate speed
- Grate 1, 2, 3 & 4 resistances.

Trends are expressed as change per minute in the unit of the signal. E.g. Steam flow is measured in t/h, a trend of 0.2 means that the Steam flow changes by 0.2 t/h per minute.

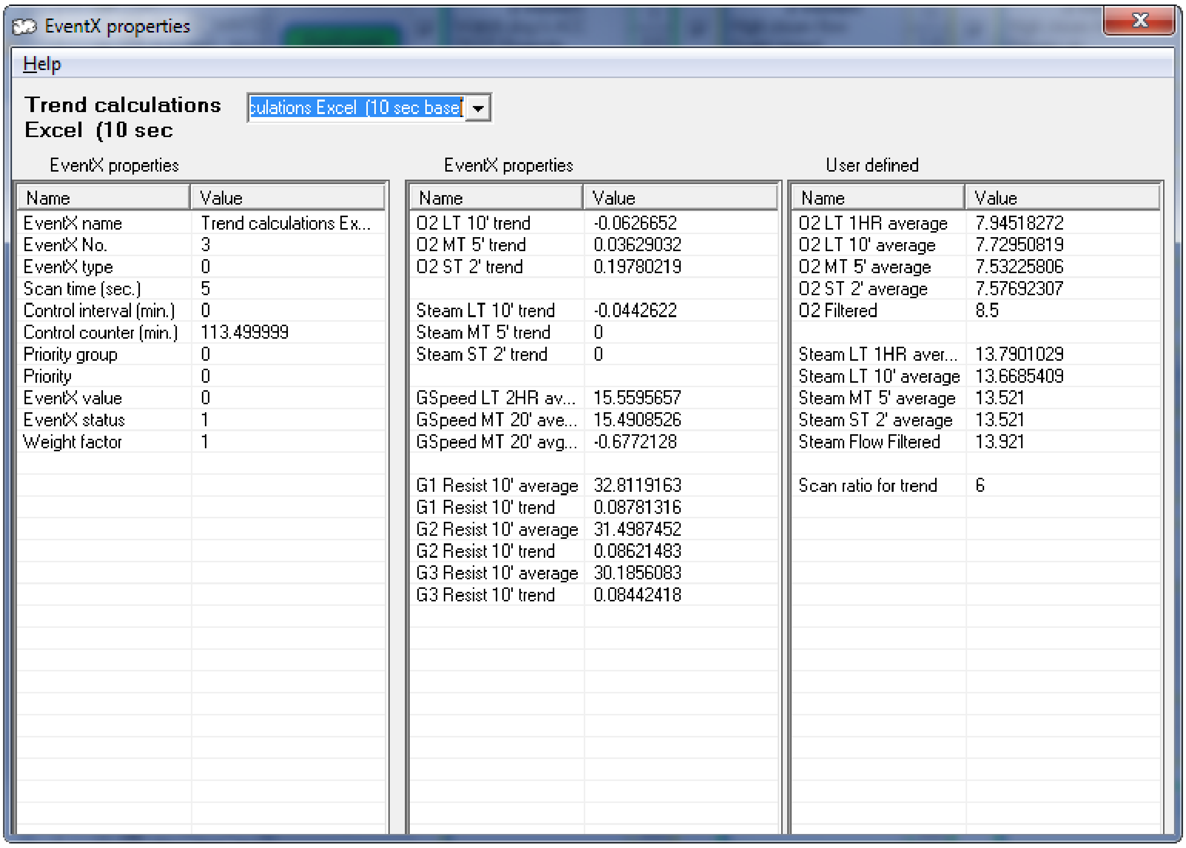


Fig. 58: EventX 3 properties

To facilitate commissioning and supervision of the FuzEvent control system, the results of the EventX 3 calculations are shown in Fig. 58 by programming the SHOW VALUE function. A few lines are shown below:

```
SHOW_VALUE("O2 LT 10' trend",g_O2_LT_TREND,1,2)
SHOW_VALUE("O2 MT 5' trend",g_O2_MT_TREND,2,2)
SHOW_VALUE("O2 ST 2' trend",g_O2_ST_TREND,3,2)
'
SHOW_VALUE("Steam LT 10' trend",g_STEAM_LT_TREND,5,2)
SHOW_VALUE("Steam MT 5' trend",g_STEAM_MT_TREND,6,2)
SHOW_VALUE("Steam ST 2' trend",g_STEAM_ST_TREND,7,2)
```

The fuzzyfied 20 minutes average feeder speed is calculated in stored in the Global variable named g_GRATE_SPEED_FZZ. The fyzzyfied average feeder speed has a value between -1 and +1, and it is calculated as shown in Fig. 59, using the SCALE instruction described in section 14.20.

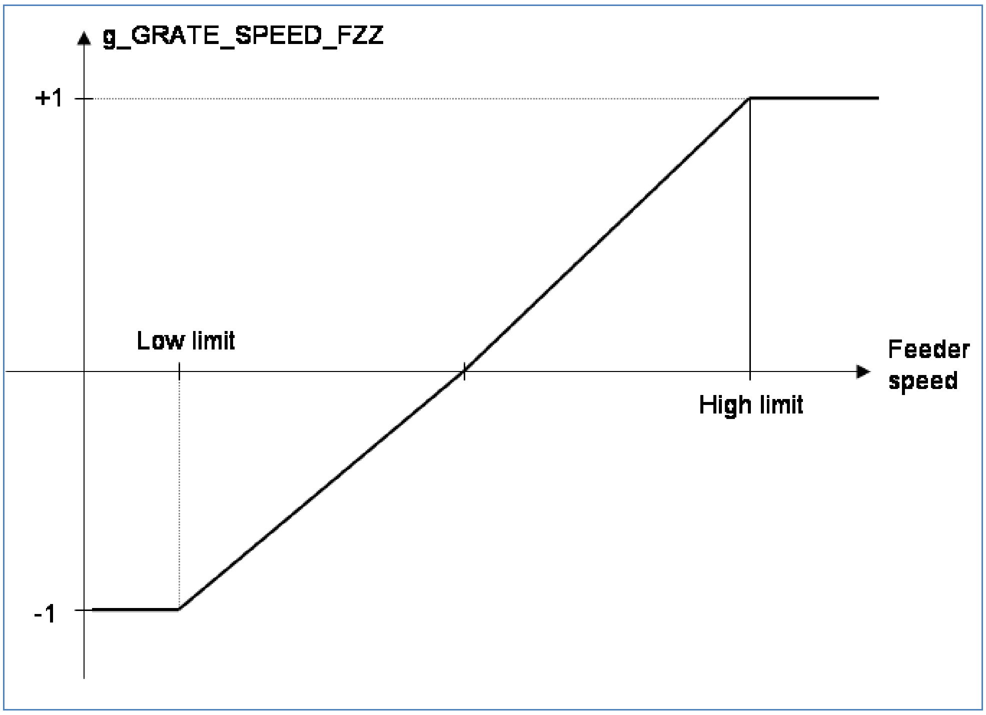


Fig. 59: Fuzzyfication of 20' average feeder speed

The script code is shown below:

```
l_I=0.5*(g_GRATE_SPEED_HL-g_GRATE_LL_PROD)+g_GRATE_LL_PROD
g_GRATE_SPEED_FZZ=SCALE(g_GRATE_SPEED_MT_AVR,g_GRATE_LL_PROD,l_I,
g_GRATE_SPEED_HL)
```

15.2.4 EventX 4: Check

Not used in this application example.

Normally used to switch the plant DCS main PID controllers to:

- Manual mode when FuzEvent is switched on.
- Automatic mode when FuzEvent is switched off.

15.2.5 EventX 5: Filter, I/O & Resistance

This script filters the following signals:

- O2: t_OV1_O2_001 → g_O2
- Steam flow: t_OV1_FIA_021 → g_STEAMFLOW
- Air flows:t_OV1_FIC_001 → g_PAIR_FLOW
  t_OV1_FIC_003 → g_SAIR_FLOW
- Furnace temperature t_OV1_TICA_003 → g_FURNACE_TEMP

In addition, the script calculates the filtered furnace pressure and the filtered under grate pressures.

Finally, the grate resistances are calculated for grates 1 to 4 from the filtered values, using the formula:

$$\mathrm{RESISTANCE} = \frac{\Delta P}{\mathrm{PairFlow}^2 * T} K$$

Where:

- ΔP is the pressure drop over the grate, i.e.
  ΔP = (under grate pressure) – (furnace pressure).
- PAirFlow is the primary air flow
- T is the absolute primary air temperature
- K is a constant to bring the resistance in the range 0 to 100 ‘ohm’ during normal operation.

The grate resistance gives an accurate indication of the waste layer thickness on each grate, largely independent of the air flow.

15.2.6 EventX 6: Grate resistance, Grate speed LL

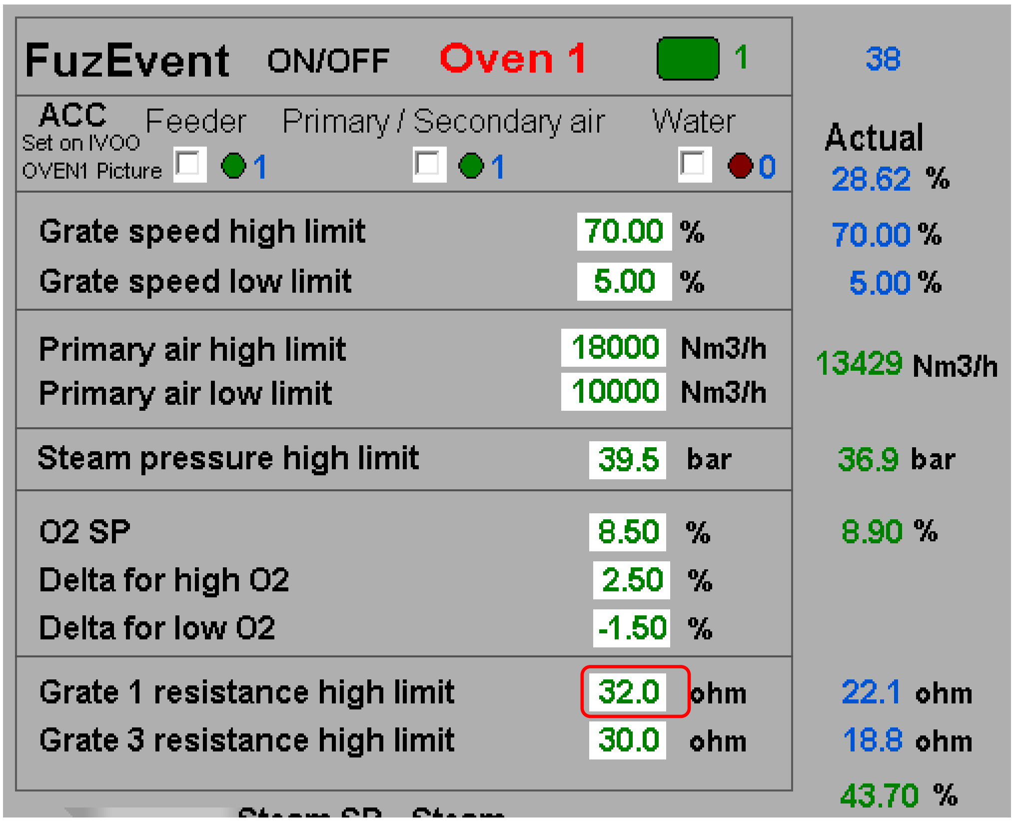


Fig. 60: Operator input fields in iFIX

From the tag, `t_G1_RESIST_HL` → `g_G1_RESIST_HL`, marked in Fig. 60, a number of other global resistance limit variables are calculated from local parameters:

- `g_G1_RESIST_SP` from `g_G1_RESIST_HL` and **#Delta for res SP f_HL**
- `g_G1_RESIST_LL` from `g_G1_RESIST_HL` and **#Delta for low res f_HL**
- `g_G1_RESIST_VH` from `g_G1_RESIST_HL` and **#Delta for VHi res f_HL**

Similar calculations are made for the grate 3 resistance, but are presently not used.

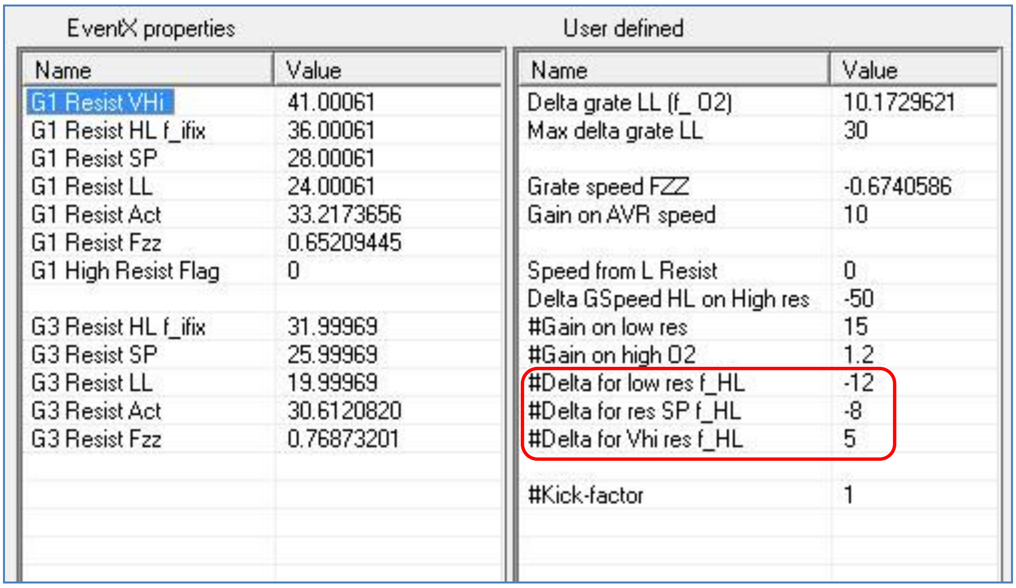


Fig. 61: EventX 6 values & parameters

This script also calculates global values, which are used in the control of the feeder speed. The values are:

- `g_G1_RESIST_FZZ`. Fuzzyfied value of Grate 1 resistance.
- `g_D_GRATE_LOW_RESIST`. This Global variable holds an increase of the feeder speed, calculated from the fuzzyfied grate 1 resistance, if:

  [Grate 1 resistance is below the calculated `g_G1_RESIST_SP`]

  The parameter **#Gain on low res** determines how large the increase in feeder speed will be if the abovementioned rule is fulfilled.

  A further contribution to `g_D_GRATE_LOW_RESIST` is calculated from `g_O2_FZZ` if:

  [Grate 1 resistance is below the calculated `g_G1_RESIST_SP`] AND [O_2 is above the O2 set-point].

  The magnitude of this O_2 High contribution is controlled by the parameter **#Gain on high O2**
- `g_D_O2_LL`. This Global variable hold a positive offset for the feeder speed low limit. The rule for increase of the feeder speed low limit is:

  [Average feeder speed is low] OR [[O2 is above the O2 set point] AND [Steam flow is below the steam flow set point]]
- `g_G1_HIGH_RESIST`. Is calculated as a logical flag for too high grate 1 resistance. If the grate 1 resistance is above the Grate 1 resistance high limit, the flag is 1, else it is 0.

The operator may change the Grate 1 resistance high limit, by changing the number on the white back ground as marked in Fig. 60. Increasing this value will increase the layer thickness on grate 1, and subsequently the layer on grates 2 to 4. The value should be in the range 25 to 40 ohm.

15.2.7 EventX 8: Flame position parameters

The idea is to adjust the Flame Position number in the lower right corner of the operators screen to indicate the position of the fire. The nearer the flame front is to the ash hopper the higher the number should be.

This feature in the FuzEvent system is mainly used when the flame front is too near to the front, resulting in un-burned waste in the hopper.
This result in the internal steam set-point is reduced proportionally by to a maximum given by the parameter Max Delta (normal value: -2.0), when the Flame Position number is varied between 60% and 100%.

This results in a lower feeder speed and a higher primary air flow.

When the situation has normalized, the operator can then write the number back to its normal 50%

15.2.8 EventX 9: Data to/from iFIX

This script is used to read from and write data to iFIX in the FuzEvent PC and to write auxiliary data to the plant DCS.

15.2.9 EventX 10: Reporting

This script writes report data to an Excel spread sheet. A report has been defined for each combustion line, and the reporting period is from midnight to midnight. The reports are saved in the folder:

C:\Program Files\FuzEvent_V4\data\Reports

For Oven 1, the report name has the following format:

FuzEvent_Report1_20110514_235915

For Oven 2, the report name format is

FuzEvent_Report2_20110515_235945

Where 20110515 is the date, i.e. May 15, 2010, and 235945 is the time for saving the report, i.e. 11:59:45 pm.

The following data are reported:

- Grate ON/OFF
- Air ON/OFF
- Water ON/OFF
- Steam Flow SP
- Steam Flow
- O2 content
- Grate Cycle time
- Primary Air flow
- Secondary Air flow
- G1 pressure
- G2 pressure
- G3 pressure
- G4 pressure
- G1 resistance
- G3 resistance
- Feeder Movements per cycle time

- Grate 1 Actual High Limit
- Grate 1 Actual Low Limit
- O2 Set-point
- Flame Position
- Steam Flow internal set-point

The report hold one line per minute, and the values reported are the minute average values based on an instant value sampled every 10 seconds.

15.3 EventX scripts for Feeder speed control

In the example in Fig. 55 the EventX scripts numbered 11 to 20 and 40 are Feeder speed control tasks performing the following tasks:

EventX	Naming	Task
11 Type 10	High steam flow Grate speed	Reducing the Feeder speed in small steps over time when the actual Steam flow is above the Steam flow set point.
12 Type 11	Low steam flow Grate speed	Increasing the Feeder speed in small steps over time when the actual Steam flow is below the Steam flow set point.
13 Type 10	Very high steam flow Grate speed	Reducing the Feeder speed in one large step when the actual Steam flow is far above the Steam flow set point. Only one action is allowed in each activation.
14 Type 11	Very low steam flow Grate speed	Increasing the Feeder speed in one large step when the actual Steam flow is far below the Steam flow set point. Only one action is allowed in each activa-tion.
15 Type 1	Change of grate speed from O2 Trend	Calculate an offset value to the Feeder speed. When the flue gas oxygen content rises and the trend is positive, the Feeder speed is increased. When it falls and the trend is negative, the Feeder speed is reduced.
18 Type 1	Feeder speed kick when low Steam Flow	Generates a momentary increase in Feeder speed when the steam flow is an adjustable amount below the Steam set-point
19 Type 0	Waste quality	Adjusts the Feeder speed High and Low limits ac-cording to the operator indicated calorific value: High, Normal or Low.
20 Type 1	Grate speed control	Summarizing the contributions and offsets to the Feeder speed and the Feeder speed limits. Limit the Feeder speed to within the actual Feeder speed High and Low limits.
40 Type 1	Calc. Feeder Move-ment	From the Feeder speed, the Feeder pusher frequen-cy is calculated.

The task performed in EventX 40 is special for this application. In other applications, the feeder speed output from EventX 20 is transmitted directly to the plant DCS.

The most important connections between the EventX scripts for Feeder speed control are shown in Fig. 62

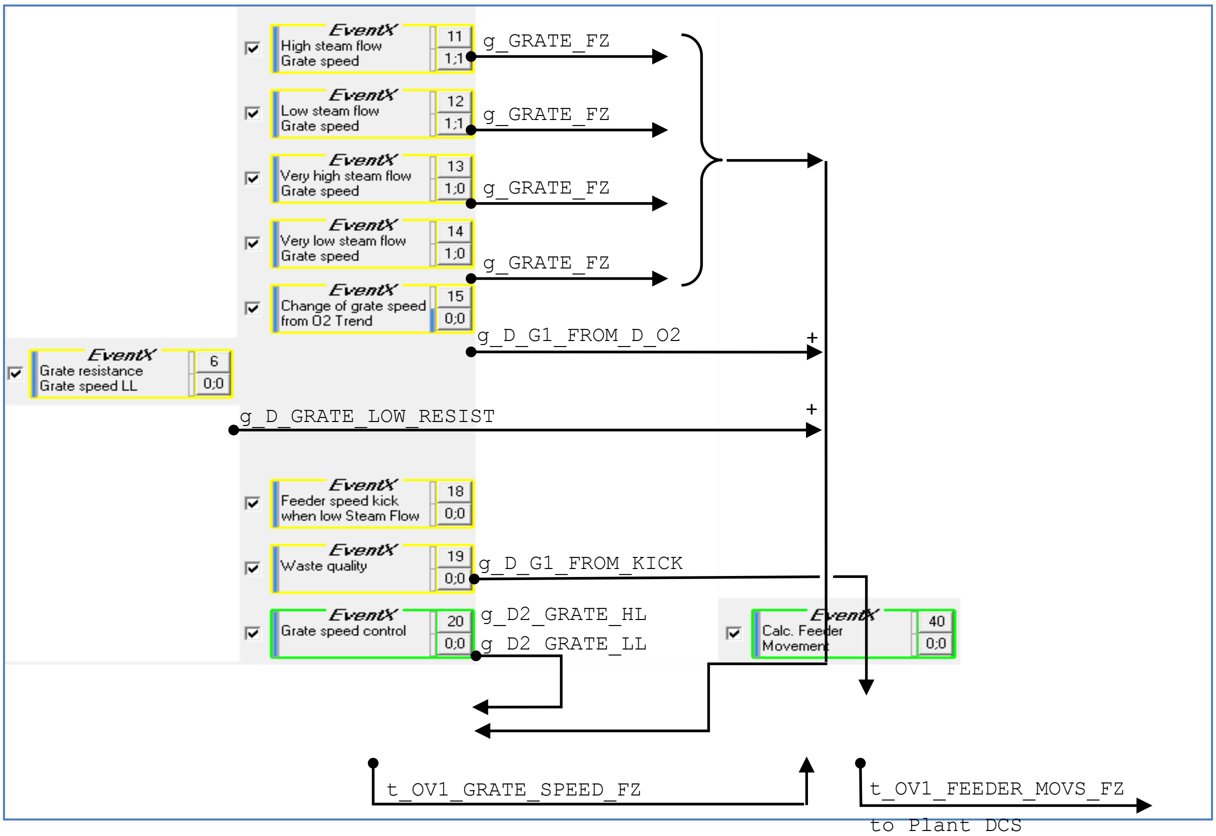


Fig. 62: The Feeder control EventX components

Note: The frames of EventX 20 and 40 turns green when the operator switches FuzEvent ON for the feeder. If FuzEvent is OFF, then frame color is yellow.

The frames of EventX 11, 12, 13 and 14 turns green when they become active.

15.3.1 EventX 11: High steam flow Grate speed

Standard FECA Type 10 script for control of feeder speed, when steam flow is above the steam flow set-point. The parameters are shown in Fig. 63.

The meanings of the parameters are:

Activation limit	=	**0.5**	Means that this EventX becomes active when the stream flow is 0.5 t/h above the steam flow set-point.
Control action 1	=	**-7.5**	Means that the feeder speed will be decreased by 7.5% on each control action.
Max No. act	=	**-1**	Means that the number of control actions during each activation is unlimited.
Deact. Limit	=	**0.2**	Means that this EventX becomes inactive when the stream flow is 0.2 t/h above the steam flow set-point.
Reverse factor 1	=	**0.2**	Means that the accumulated actions will be reversed by a factor 0.2 when the EventX becomes inactive. This will be done in three steps if Stepwise reverse actions is selected

Stepwise reverse actions	**V**		When this is selected, reverse actions will be done in three steps as the steam flow decreases towards the set-point.
Control intv. (min)	**=**	**1.5**	Means that the interval between control actions is 90 seconds.

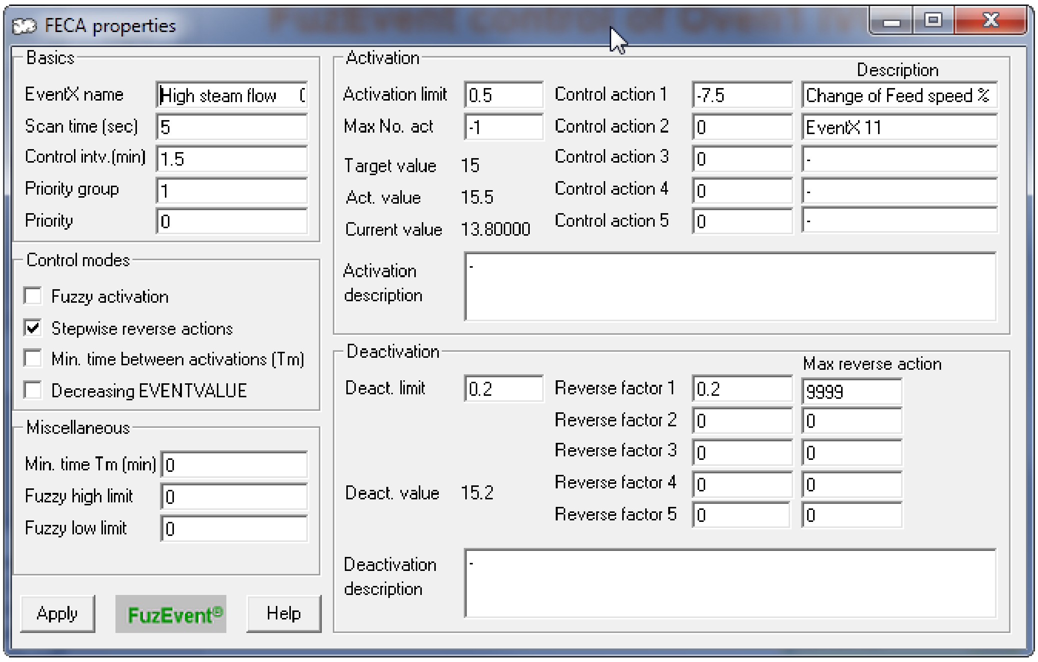


Fig. 63: Parameters for EventX 11 - High steam flow Grate speed

Below, in Fig. 64, a snapshot of the values in EventX 11 in its active state is shown.

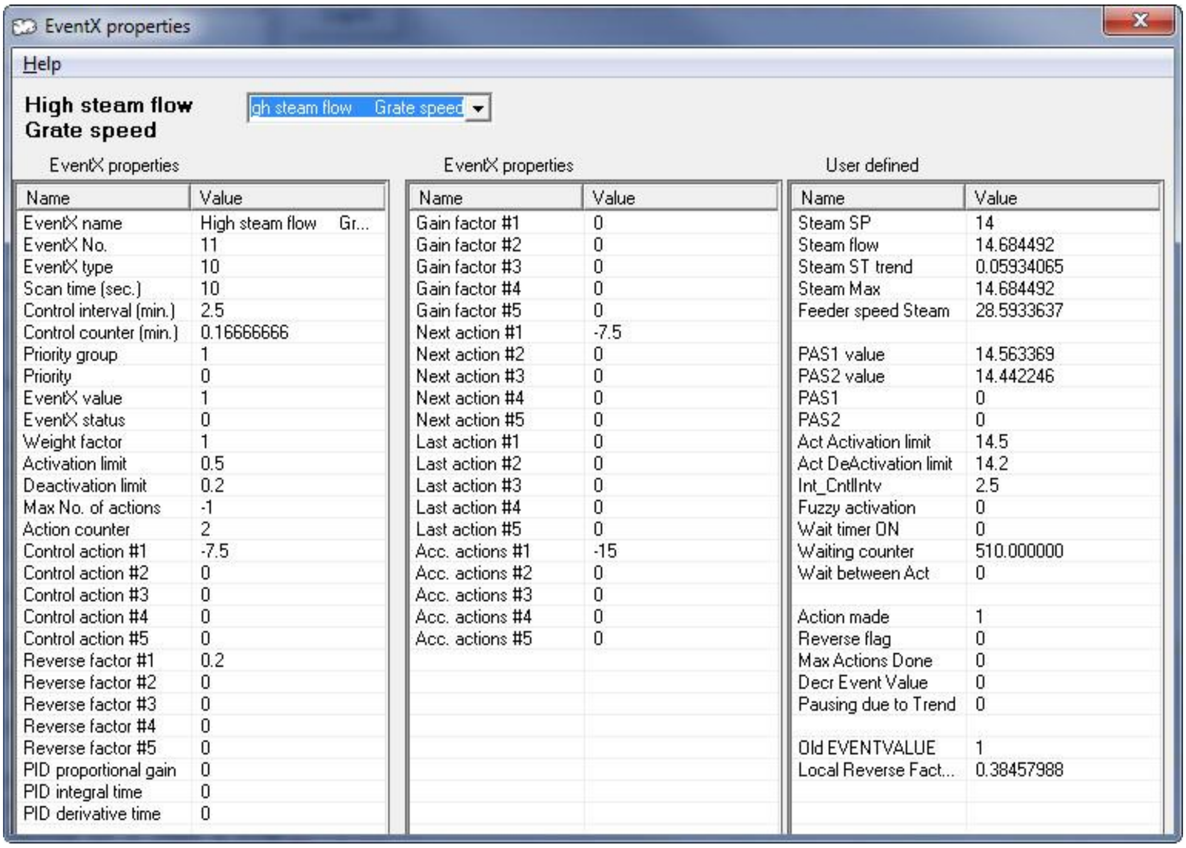


Fig. 64: EventX 11 active state values

15.3.2 EventX 12: Low steam flow Grate speed

Standard FECA Type 11 script for control of feeder speed, when the steam flow is lower than the steam flow set-point. The parameters are shown in Fig. 65

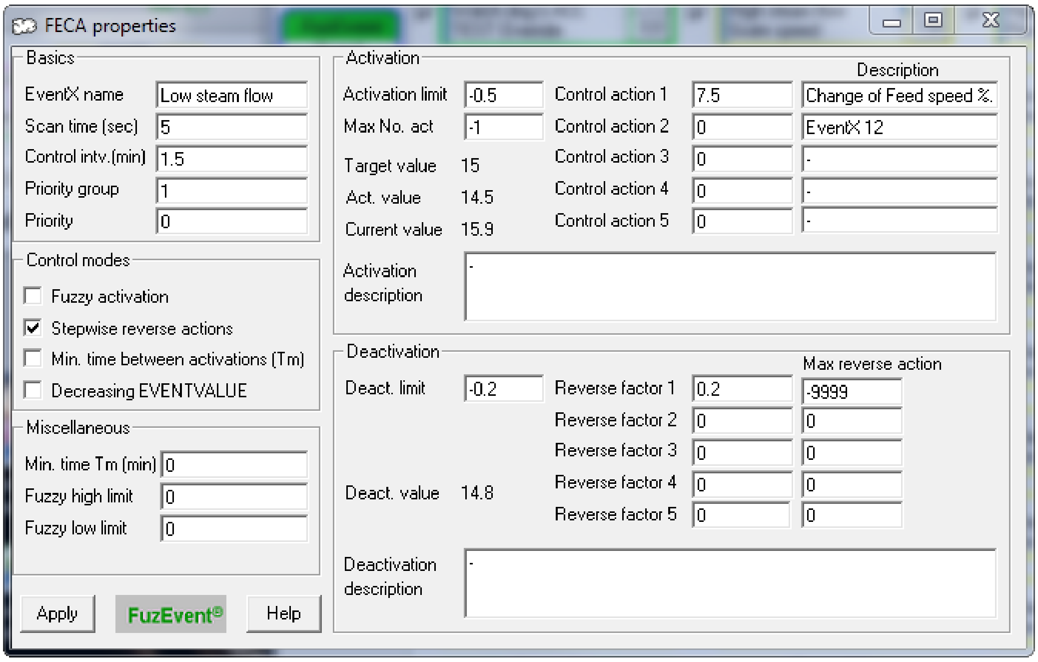


Fig. 65: Parameters for EventX 12 - Low steam flow Grate speed

15.3.3 EventX 13: Very high steam flow Grate speed

Standard FECA Type 10 script for control of feeder speed, when the steam flow is much higher than the steam flow set-point. The parameters are shown in Fig. 66 This EventX is an assist function to EventX 11 and is only allowed one control action during each activation.

Max No. act	**=**	**1**	Means that the number of control actions during each activation is one.
Decreasing EVENTVALUE		**V**	Means that this script will deactivate gradually over time reducing its EVENTVALUE. This re-activates EventX 11 by increasing its EVENTWEIGHT gradually. This is true even if the Process Value, here the Steam flow, has not decreased below the **Deact. Value**.

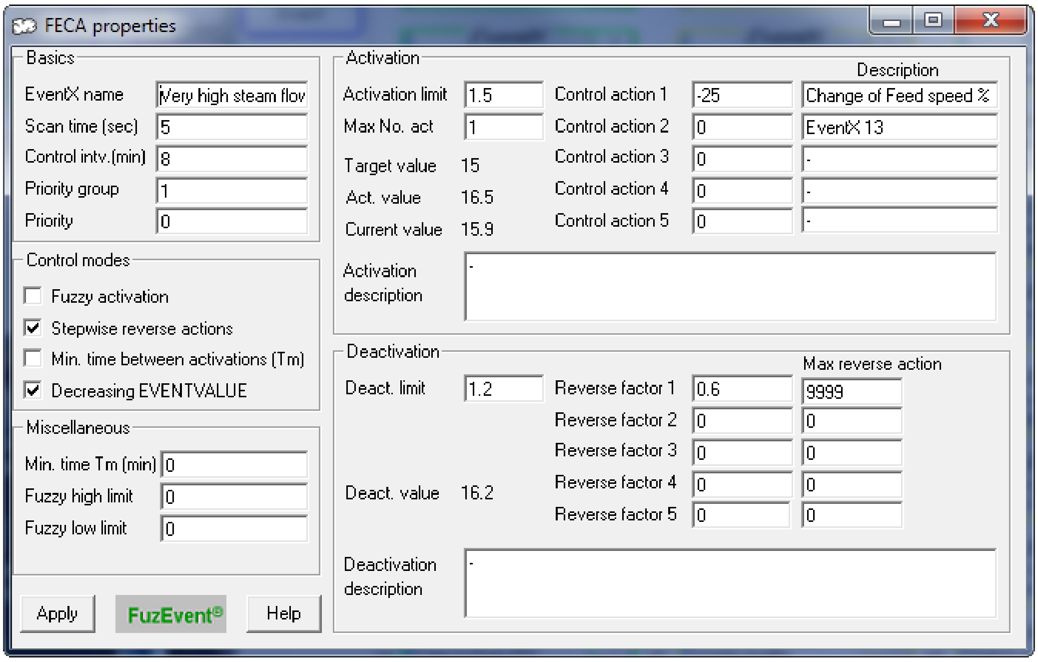


Fig. 66: Parameters for EventX 13 - Very high steam flow Grate speed

15.3.4 EventX 14: Very low steam flow Grate speed

Standard script for control of feeder speed, when the steam flow is much lower than the steam flow set-point. The parameters are shown in Fig. 67. This EventX is an assist function to EventX 12 and are only allowed one action during each activation.

Max No. act	=	**1**	Means that the number of actions during each activation is one.
Decreasing EVENTVALUE		**V**	Means that this script will deactivate gradually over time by reducing its EVENTVALUE. This re-activates EventX 12 by increasing its EVENTWEIGHT gradually. This is true even if the Process Value, here the Steam flow, has not increased above the **Deact. Value**.

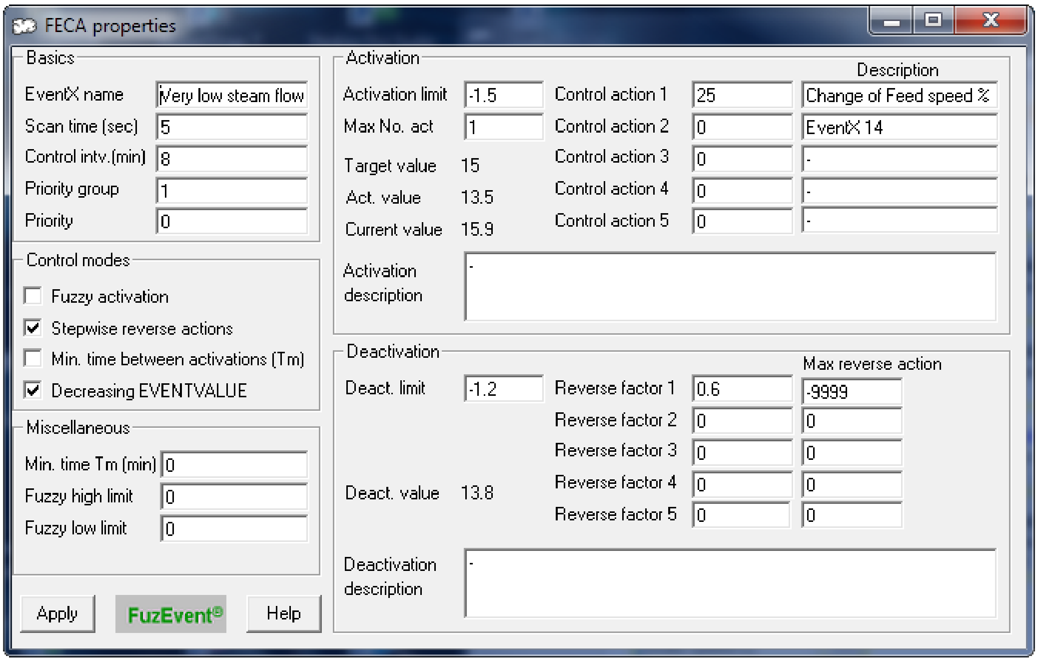


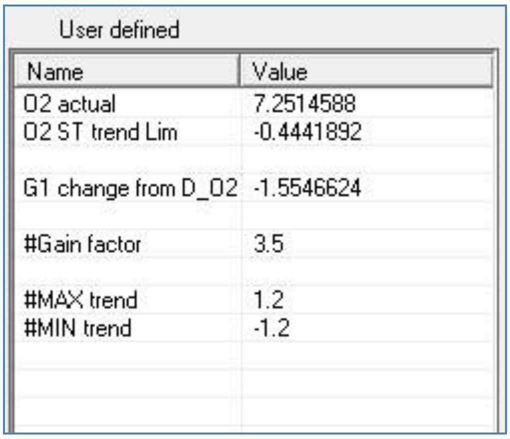
Fig. 67: Parameters for EventX 14 - Very low steam flow Grate speed

15.3.5 EventX 15: Change of grate speed from O_2 trend.

This script calculates an offset value (delta value) `g_D_G1_FROM_D_O2`, shown as `G1 change from D_O2` in Fig. 68, for the feeder speed as a function of the change of oxygen. If O2 increases, then the offset value is positive, and if O2 decreases, then the offset value for feeder speed will be negative.

In EventX No. 20, the offset value is added to the feeder speed set point, `g_GRATE_FZ`, calculated in EventX No. 11 to No. 14.

The parameter named **#Gain factor** determines how much the O2 trend should influence the speed of the feeder. Further **#MAX trend** and **#MIN trend** controls the trend limits. These can be changed on the Property window of EventX No. 15, shown in Fig. 68.


User defined

Name	Value
O2 actual	7.2514588
O2 ST trend Lim	-0.4441892
G1 change from D_O2	-1.5546624
#Gain factor	3.5
#MAX trend	1.2
#MIN trend	-1.2

Fig. 68: Values and parameters in EventX 15

A fuzzyfied value of O_2, `g_O2_FZZ`, is calculated from `g_O2`, `t_OV1_D_O2_LL`, `g_O2_SP` and `t_OV1_D_O2_HL` for use in other EventX scripts. See Fig. 69.

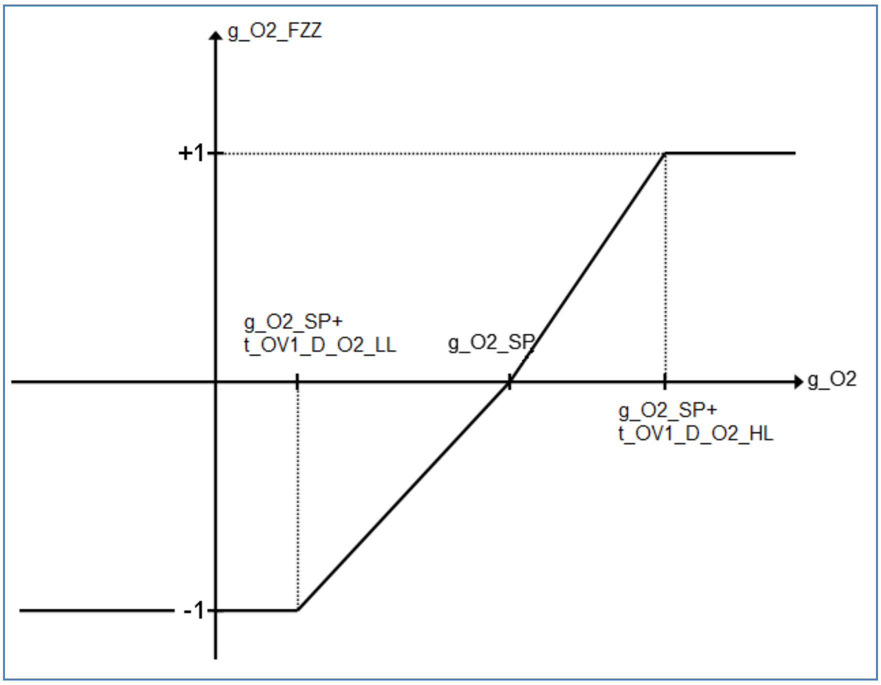


Fig. 69: O_2 fuzzyfication using the SCALE function

15.3.6 EventX 18: Feeder speed kick when low Steam Flow.

This script generates a momentary increase in feeder speed when the steam flow is **#Delta for Steam LL** below the steam set-point. This function increases the feeder speed by **#SP Speed Kick** for **#Kick duration** and then pauses for **#Pause duration**, as long as the above condition exists and the **Steam ST Trend** is below the fixed value 0.2.

User defined

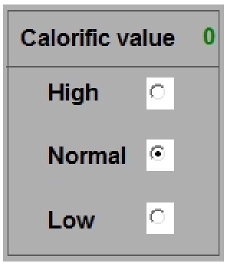
Name	Value
#Delta for Steam LL	-1.8
Steam LL for kick	12.2
Act Steam flow	11.684492
Steam ST trend	0
02 ST Trend	0.72527472
#Gain on 02 ST	10
Delta Speed Kick	0
#SP Speed Kick	35
#Kick duration	5
#Pause duration	15
Counter1	0
Counter2	15

Fig. 70: Values and parameters in EventX 18

The output, `g_D_G1_FROM_KICK` (**Delta Speed Kick**), is used in EventX 40, Calc. Feeder Movement

15.3.7 EventX 19: Waste quality

On the operators screen picture it is possible to select between three different qualities of the waste, i.e.:



The High/Low calorific value changes the high/low limits for the speed of the feeder. These parameter values can be specified on the property window of EventX No. 19, as shown in Fig. 71.

The meanings of the parameters are:

#Delta HL for high cal: Change in feeder speed high limit for high calorific waste

#Delta LL for high cal: Change in feeder speed low limit for high calorific waste

#Delta HL for low cal: Change in feeder speed high limit for low calorific waste.

#Delta LL for low cal: Change in feeder speed low limit for low calorific waste.

User defined

Name	Value
Waste quality	0
#Delta HL for high cal_	10
#Delta LL for high cal_	0
#Delta HL for low cal_	-20
#Delta LL for low cal_	0
Delta GSpeed HL	0
Delta GSpeed LL	0

Fig. 71: Values and parameters for EventX 19

15.3.8 EventX 20: Grate speed control

This EventX summarizes the contributions to the set point for feeder speed that come from EventX No. 11 to No. 15, as shown in Fig. 62 on page 86. These contributions are marked on Fig. 73 The output, `t_OV1_GRATE_SPEED_FZ` in % (**Feed Speed OUT**), is sent to EventX 40 for calculation of the feeder movements in cycle time.

In addition, the actual high/low limits are calculated from parameters `g_GRATE_HL` and `g_GRATE_LL`, which have been specified on the iFIX user's interface picture.

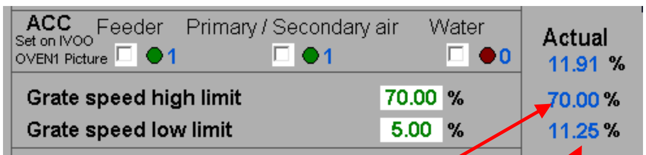
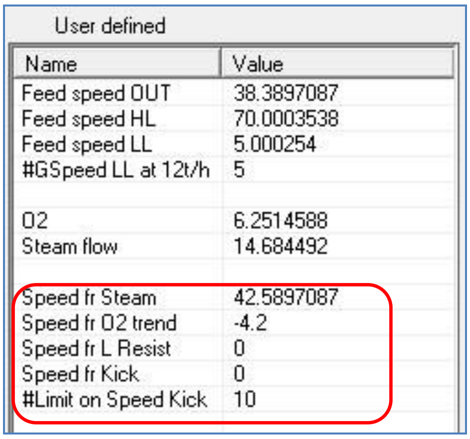


Fig. 72: Grate speed operator limits and actual values

These actual limits **Feed Speed HL** and **Feed Speed HL** are adjusted as a function of Grate 1 resistance, Waste calorific value and from Flame position.

If the `g_G1_HIGH_RESIST` flag is set from EventX 6, indicating a too thick waste layer situation on grate 1, then `g_GRATE_HL` is reduced by the value of the parameter `g_D_GRATE_HL_RESIST`.

`g_GRATE_HL` is the common high limit for the FECA scripts EventX 11 to 14, so the value Speed from Steam `g_GRATE_FZ` will be limited to `g_GRATE_HL`. When the thick layer condition is over, the FECA scripts will control `g_GRATE_FZ` again, preventing large sudden jumps in the feeder speed.


User defined

Name	Value
Feed speed OUT	38.3897087
Feed speed HL	70.0003538
Feed speed LL	5.000254
#GSpeed LL at 12t/h	5
O2	6.2514588
Steam flow	14.684492
Speed fr Steam	42.5897087
Speed fr O2 trend	-4.2
Speed fr L Resist	0
Speed fr Kick	0
#Limit on Speed Kick	10

Fig. 73: Values and parameters in EventX 20

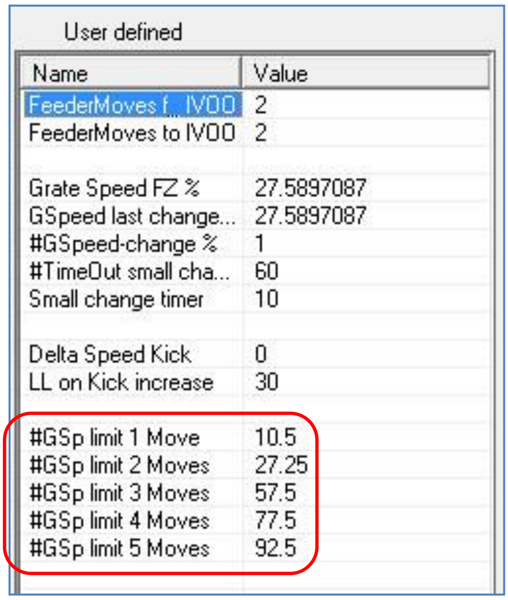
15.3.9 EventX 40: Calc. Feeder Movement

This script calculates the feeder movements in cycle time, `t_OV1_FEEDER_MOVS_FZ`, from `t_OV1_GRATE_SPEED_FZ` originating from EventX 20. The calculation is based on a set of parameters defining the limits for change as marked in Fig. 74.

I.e. the feeder speed limit for changing from zero (stop) to one feeder moves per cycle time is here 10.5 %.

This relates to 'Grate speed high limit' on the iFIX operator picture, which are shown as set to 70% on **Fejl! Henvisningskilde ikke fundet.**on page **Fejl! Bogmærke er ikke defineret.**. Therefore the maximum speed is three moves per cycle time, as the limit **GSP limit 4 Moves** is set to 77.5 % in Fig. 74.

The effect of `g_D_G1_FROM_KICK` from EventX 18 will simply increase `t_OV1_FEEDER_MOVS_FZ` by one, but is limited to a maximum of three moves to prevent overfilling the furnace.


User defined

Name	Value
FeederMoves f. IVOO	2
FeederMoves to IVOO	2
Grate Speed FZ %	27.5897087
GSpeed last change...	27.5897087
#GSpeed-change %	1
#TimeOut small cha...	60
Small change timer	10
Delta Speed Kick	0
LL on Kick increase	30
#GSp limit 1 Move	10.5
#GSp limit 2 Moves	27.25
#GSp limit 3 Moves	57.5
#GSp limit 4 Moves	77.5
#GSp limit 5 Moves	92.5

Fig. 74: Values and parameters for EventX 40

Small changes less than **GSpeed-change %** are considered each **TimeOut small change** seconds to prevent oscilating between two movement levels. **Small change timer** shows the running time.

The output, `t_OV1_FEEDER_MOVS_FZ`, **(Feeder moves to IVOO)** is transmitted to the IVOO Scada and controls the number of feeder movements in cycle time directly.

15.4 EventX scripts for Primary and Secondary air control

In the example in Fig. 55 the EventX scripts numbered 21 to 30 and 39 are Primary and Secondary fan speed control tasks performing the following tasks:

EventX	Naming	Task
21 Type 10	High steam flow Primary air	Reducing the Primary air fan speed in small steps over time when the actual Steam flow is above the Steam flow set point.
22 Type 11	Low steam flow Primary air	Increasing the Primary air fan speed in small steps over time when the actual Steam flow is below the Steam flow set point.
23 Type 10	Very high steam flow Primary air	Reducing the Primary air fan speed in two large steps when the actual Steam flow is far above the Steam flow set point. Only two actions are allowed in each activation.
24 Type 11	Very low steam flow Primary air	Increasing the Primary air fan speed in two large steps when the actual Steam flow is far below the Steam flow set point. Only two actions are allowed in each activation.
25 Type 0	Change of Primary Air from O2 Trend	Calculate an offset value to the Primary air fan speed. When the flue gas oxygen content rises and the trend is positive, the PA fan speed is increased. When it falls and the trend is negative, the PA fan speed is reduced.
26 Type 0	Change of Primary Air from furnace pressure	Calculate an offset value to the Primary air fan speed. As the absolute furnace pressure increase, the PA fan speed is reduced to maintain under pres-sure in the furnace and boiler. When the absolute furnace pressure is back to, or below normal, the offset is removed.
27 Type 0	Change of primary air from high/low O2	Calculate two offset values to the Primary air fan speed. The PA fan speed is increased gradually as the flue gas oxygen content rises, up to an operator defined limit “Delta for high O2” above “O2 SP” The PA fan speed is reduced gradually as the flue gas oxygen content falls, up to an operator defined limit “Delta for low O2” below “O2 SP”
29 Type 0	High/Low PA flow Primary Air	Calculate an offset value to the Primary air fan speed. Gradually reduce the PA fan speed over time when the steam flow is above the operator indicat-ed “Primary air high limit”. Gradually increase the PA fan speed over time when the steam flow is be-low the operator indicated “Primary air low limit”.
30 Type 1	Primary Air control	Summarizing the contributions and offsets to the Primary air fan speed. Limit the PA fan speed to within the actual PA fan speed High and Low limits. Limit the PA fan speed output slew rate to prevent PA fan inverter trips.

EventX	Naming	Task
39 Type 13	Secondary Air Difference Control	The demand for Secondary air is controlled by the difference between the Total Air set-point and the actual Primary air flow The SA fan speed output is controlled by a PID type controller.

The EventX components for Primary and Secondary and their most important connections air are shown in Fig. 75

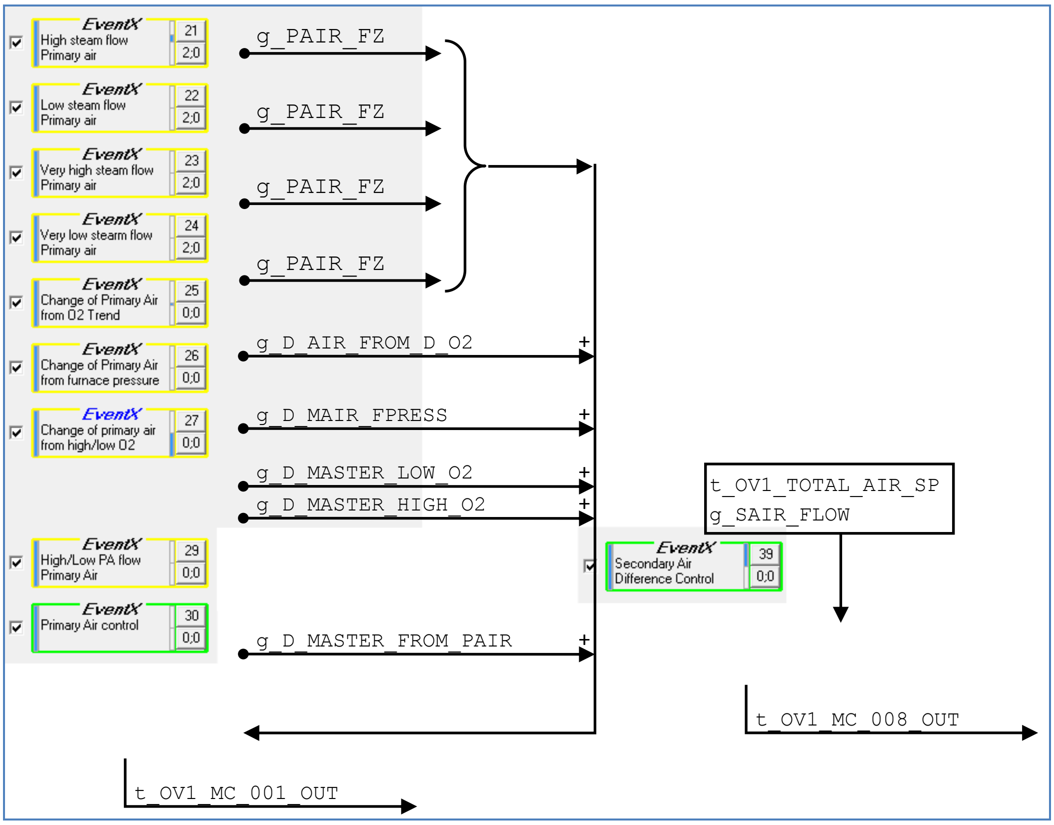


Fig. 75: The Air control EventX components

Note: The frame of EventX 30 and 39 turns green when the operator switches FuzEvent ON for the air. If FuzEvent is OFF, then frame color is yellow.

The frames of EventX 21, 22, 23 and 24 turns green when they become active.

15.4.1 EventX 21: High steam flow Primary air

Standard FECA Type 10script for control of primary air fan output, when the steam flow is high. The parameters are shown in Fig. 76.

The meanings of the parameters are:

Activation limit	=	**0.3**	Means that this EventX becomes active when the stream flow is 0.3 t/h above the steam flow set-point.
Control action 1	=	**-3**	Means that the primary air fan output will be de-creased by 3 % on each control action.
Max No. act	=	**-1**	Means that the number of control actions during each activation is unlimited.
Deact. Limit	=	**0.1**	Means that this EventX becomes inactive when the stream flow is 0.1 t/h above the steam flow set-point.
Reverse factor 1	=	**0.5**	Means that the accumulated actions will be reversed by a factor 0.2 when the EventX becomes inactive. This will be done in three steps if Stepwise reverse actions is selected
Stepwise reverse actions		**V**	When this is selected, reverse actions will be done in three steps as the steam flow decreases towards the set-point.
Control intv. (min)	=	**1**	Means that the interval between control actions is 1 minute.

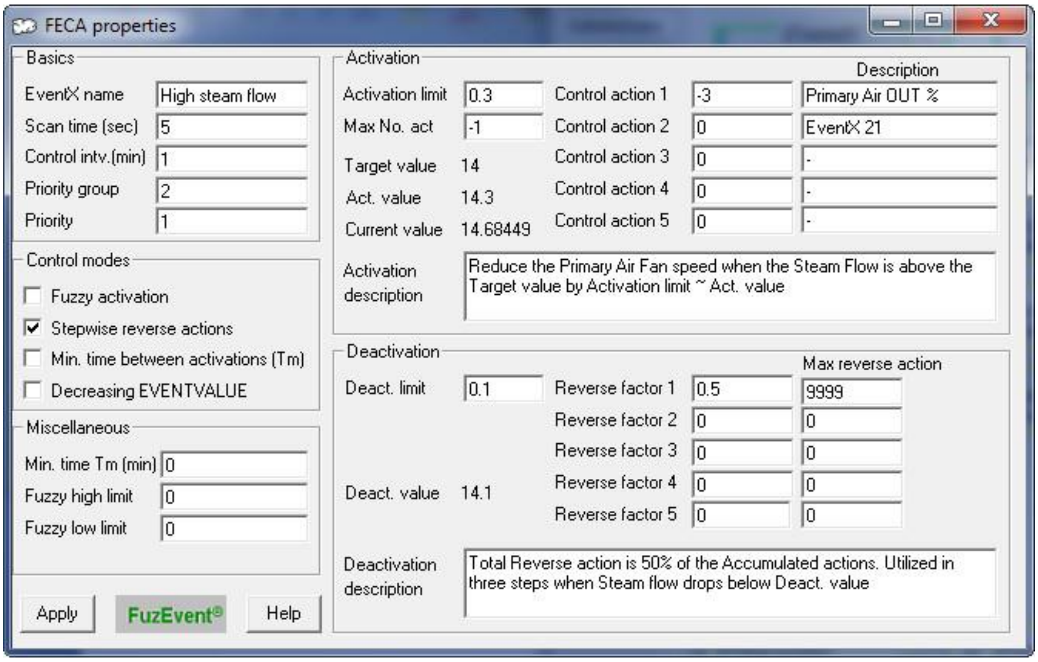


Fig. 76: Parameters for EventX 21 - High steam flow Primary air

15.4.2 EventX 22: Low steam flow Primary air

Standard FECA Type 11 script for control of primary air fan output, when the steam flow is low. The parameters are shown in Fig. 77.

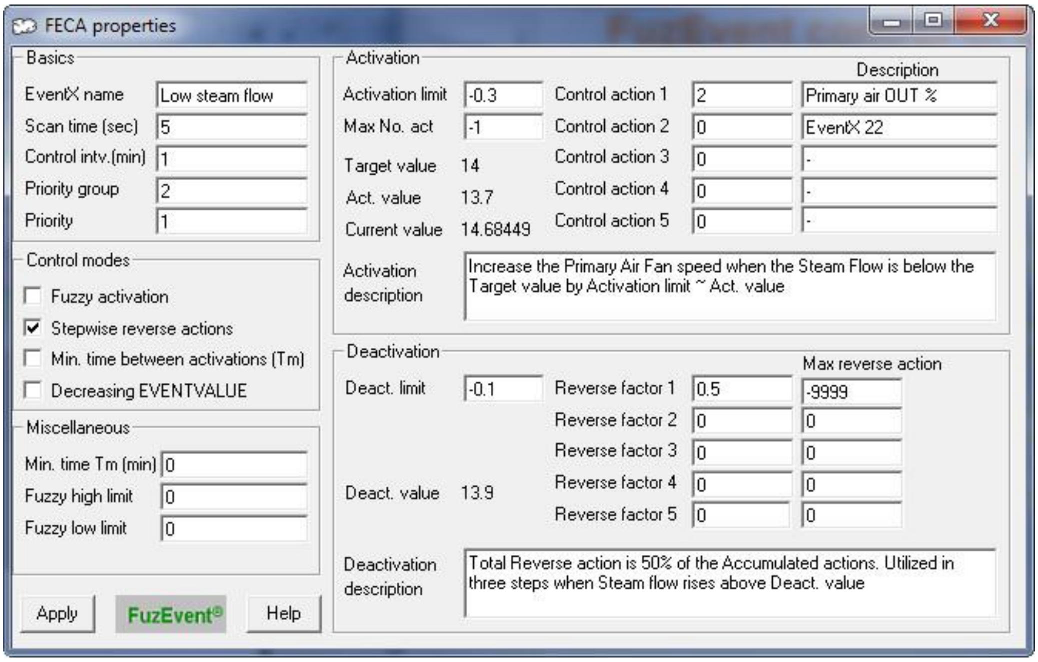


Fig. 77: Parameters for EventX 22 - Low steam flow Primary air

15.4.3 EventX 23: Very high steam flow Primary air

Standard FECA Type 10 script for control of primary air fan output, when the steam flow is very high. The parameters are shown in Fig. 78. This EventX is an assist function to EventX 21 and is only allowed one control action during each activation.

Max No. act	=	1	Means that the number of control actions during each activation is one.
Decreasing EVENTVALUE		√	Means that this script will deactivate gradually over time by reducing its EVENTVALUE. This re-activates EventX 21 by increasing its EVENTWEIGHT gradually. This is true even if the **Current** (Process) **Value**, here the Steam flow, has not decreased below the **Deact. Value**.

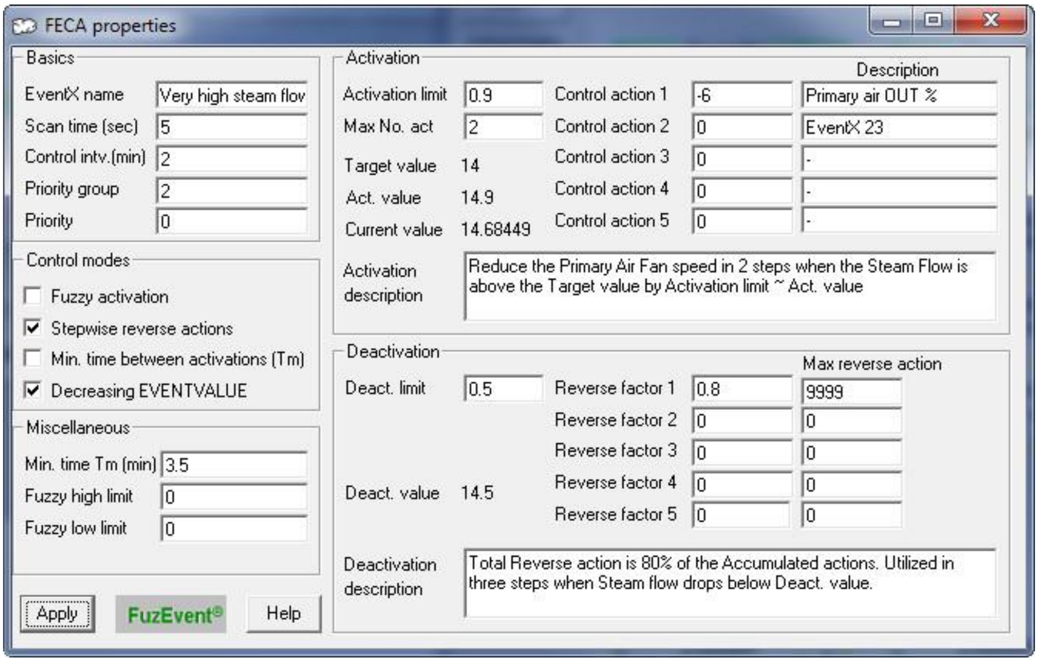


Fig. 78: Parameters for EventX 23 – Very high steam flow Primary air

The way the EVENTVALUE is reduced can be programmed individually for each case.

In this example the EVENTVALUE is reduced by subtracting 0.2 every CONTROLINTERVAL after MAXNOOFACTIONS have been performed. That is, this script will be passive in 5 CONTROLINTERVALs. Further the ACCACTIONS1 is reduced by 20% each time the EVENTVALUE is reduced. This gradually transfers full control to EventX 21.

The purpose is to avoid sudden rises in the Controlled Value, here the PA fan speed output, by the activation of Reverse Actions after the control system have been outside its operational window for a long time. This could be caused by sudden feed of light dry fast-burning waste.

The **Current** (Process) **Value**, here the Steam flow, has to decrease below the **Deact. Value** before this script can be re-activated

15.4.4 EventX 24: Very low steam flow Primary air

Standard script for control of primary air fan output, when the steam flow is very low. The parameters are shown in Fig. 79. This EventX is an assist function to EventX 22 and is only allowed one control action during each activation.

Max No. act	=	1	Means that the number of control actions during each activation is one.
Decreasing EVENTVALUE		V	Means that this script will deactivate gradually over time by reducing its EVENTVALUE. This re-activates EventX 22 by increasing its EVENTWEIGHT gradually. This is true even if the **Current** (Process) **Value**, here the Steam flow, has not decreased below the **Deact. Value**.

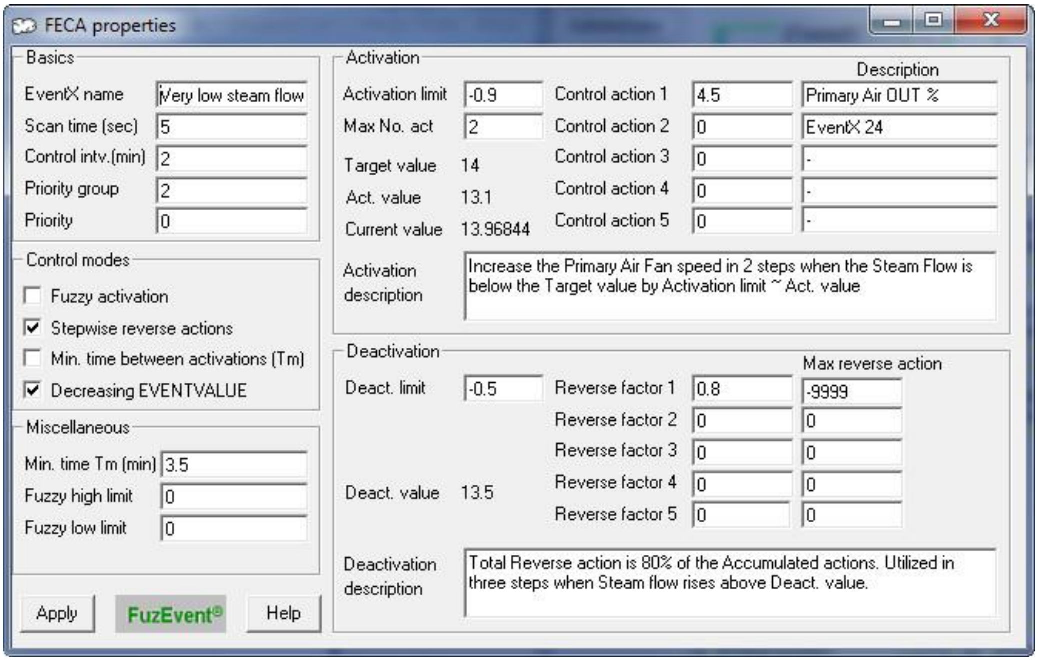


Fig. 79: Parameters for EventX 24 – Very low steam flow Primary air

The way the EVENTVALUE is reduced can be programmed individually for each case.

In this example the EVENTVALUE is reduced by subtracting 0.2 every CONTROLINTERVAL after MAXNOOFACTIONS have been performed. That is, this script will be passive in 5 CONTROLINTERVALs. Further the ACCACTIONS1 is reduced by 20% each time the EVENTVALUE is reduced. This gradually transfers full control to EventX 22.

The purpose is to avoid sudden falls in the Controlled Value, here the PA fan speed output, by the activation of Reverse Actions after the control system have been outside its operational window for a long time. This could be caused by heavy wet slow-burning waste.

The **Current** (Process) **Value**, here the Steam flow, has to rise above the **Deact. Value** before this script can be re-activated

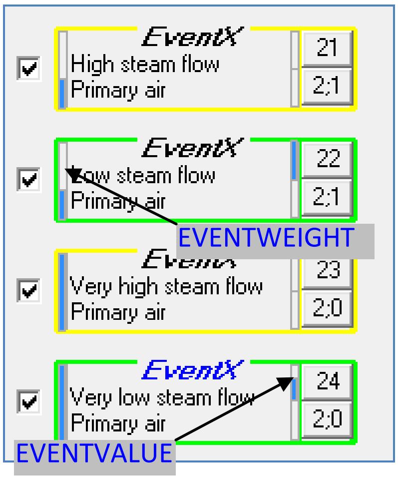


Fig. 80: EventX 24 partly de-activated

In Fig. 80 a situation is shown, where EventX 24 is partly deactivated and its EVENTVALUE was reduced to 0.6, this has partly re-activated EventX 22 by increasing its EVENTWEIGHT to 0.4. The actions of EventX 22 will be 0.4 * CONTROLACTION1

15.4.5 EventX 25: Change of Primary Air from O_2 Trend

This script calculates an offset value (delta value) `g_D_AIR_FROM_D_O2` (**Air from O2 trend**), for the primary air fan output as a function of the change of oxygen.

If O_2 increases, then the offset value is positive, and if O_2 decreases, then the offset value for primary air fan output will be negative.

In EventX No. 30, the offset value is added to the primary air fan output set-point, `g_PAIR_FZ`, calculated in EventX No. 21 to No. 24.

The parameter named **#Gain factor** determines how much the O2 trend should influence the speed of the feeder. Further **#MAX trend** and **#MIN trend** controls the trend limits.

They can be changed on the Property window of EventX No. 25, Fig. 81

User defined

Name	Value
O2 ST Trend Lim	0
Air from O2 trend	0
Act O2	6.2514588
Act Steam flow	13.9684492
#Gain factor	1.2
#MAX trend	0.5
#MIN trend	-1.2
PA Fan OUT	49.7543764

Fig. 81: Values and parameters for EventX 25

15.4.6 EventX 26: Change of Primary Air from furnace pressure

This script calculates a negative offset value (delta value), `g_D_MAIR_FPRESS` (**Air from Furnace Press**), for the primary air fan output as function of the furnace pressure.

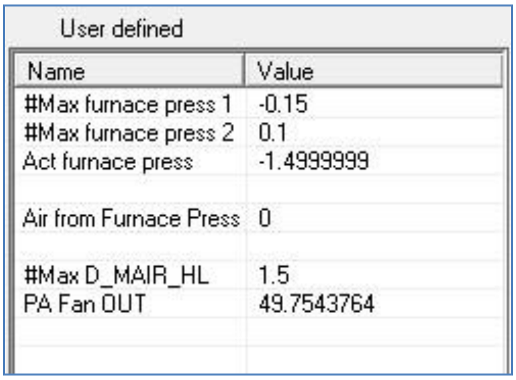
If the furnace pressure is less (more negative) than `l_MAX_FPRESS1` (**#Max furnace press 1**), then the offset value is 0.

When the furnace pressure increases to between `l_MAX_FPRESS1` and `l_MAX_FPRESS2` (**#Max furnace press 2**), the offset is the scaled value of the furnace pressure between `l_MAX_FPRESS1` and `l_MAX_FPRESS2`

If the furnace pressure is greater (less negative or positive) than `l_MAX_FPRESS2`, then the offset value is equal to `–l_D_MAIR_HL` (**#Max D_MAIR_HL**), =-1.5%.

In other words, if the furnace pressure is too high, then the primary air fan output is decreased by up to 1.5%.

The parameters can be changed via the Property page of EventX No. 26, Fig. 82



User defined

Name	Value
#Max furnace press 1	-0.15
#Max furnace press 2	0.1
Act furnace press	-1.4999999
Air from Furnace Press	0
#Max D_MAIR_HL	1.5
PA Fan OUT	49.7543764

Fig. 82: Values and parameters for EventX 26

15.4.7 EventX 27: Change of Primary Air from high/low O_2

This script calculates two offset values (delta values): `g_D_MASTER_LOW_O2` (**Air from low O2**) and `g_D_MASTER_HIGH_O2` (**Air from high O2**) for the primary air fan output as function of O2 content.

The relationship between O_2 `g_D_MASTER_LOW_O2` (**Air from low O2**) is shown in Fig. 83.

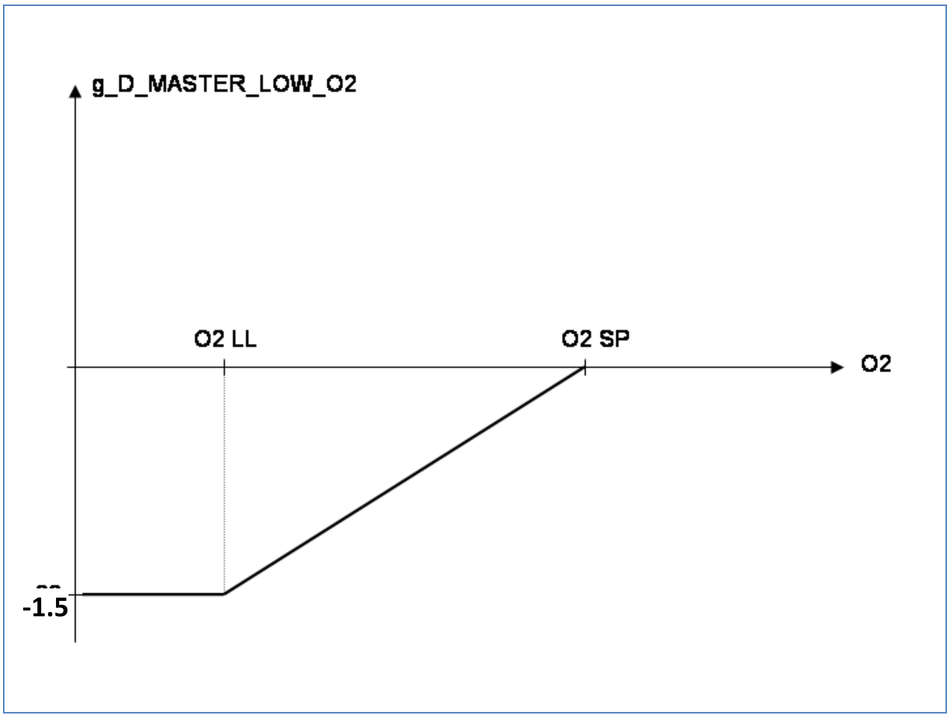


Fig. 83: Primary air fan output offset as function of low O_2

The relationship between O_2 and the `g_D_MASTER_HIGH_O2` (**Air from high O2**) is shown in Fig. 84.

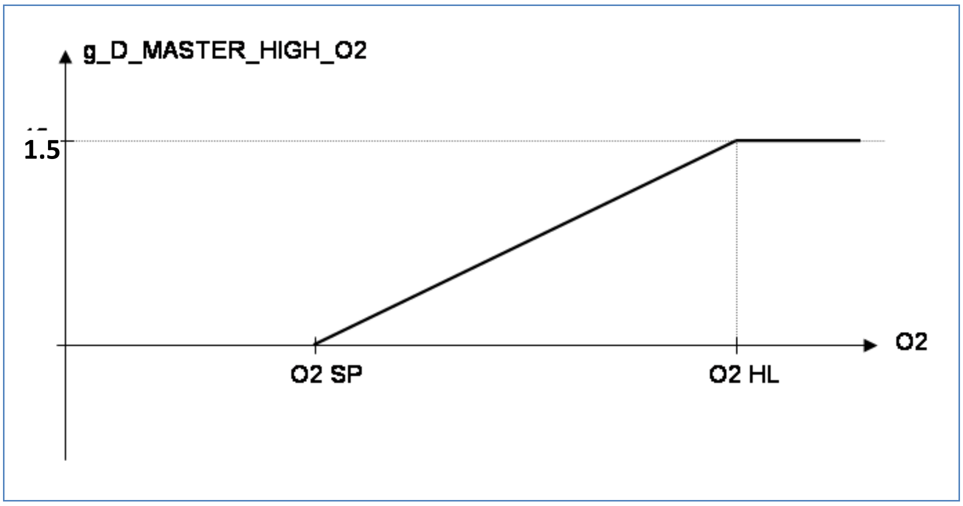


Fig. 84: Primary air fan output offset as function of high O_2

The parameter **#Master air MAX change** can be changed via the Property window for EventX No. 27, as shown in Fig. 85.

User defined

Name	Value
Air from low O2	-2.2456236
Air from high O2	0
#Master air MAX change	3
Act O2	6.2514588
Act Steam flow	13.9684492
O2 SP	7
PA Fan OUT	49.7543764

Fig. 85: Values and parameters for EventX 27

The O2 set point is specified by the operator, and the delta values, which defines the high and low limits for O_2 is specified on the iFIX picture, as shown in Fig. 86

O2 SP	9.50	%	9.00 %
Delta for high O2	2.50	%	
Delta for low O2	-1.10	%	

Fig. 86: O2 set-point and delta values for high and low O2

15.4.8 EventX 29: High/Low PA flow Primary Air

This script gradually reduces or increases an offset value (delta value), `g_D_MASTER_FROM_PAIR` (**Delta**), when the actual **PAir flow** is outside the limits 'Primary air high limit' (**Pair flow HL**) and 'Primary air low limit' (**Pair flow LL**) as specified by the operator on the iFIX picture, as shown in Fig. 88

When the actual PA flow is outside the limits, `g_D_MASTER_FROM_PAIR` (**Delta**) is reduced or increased by **#Delta each scan** until **#MAX delta** is reached. Increased if the actual **PAir flow** is below **Pair flow LL** and decreased if the actual **PAir flow** is above **Pair flow HL.**

When the actual PA flow is inside the limits, `g_D_MASTER_FROM_PAIR` (**Delta**) is reduced or increased by 2 * **#Delta each scan** until **#MAX delta** equals 0.00. The scan time is 10 sec.

User defined

Name	Value
Delta	0
PAir flow	20999.9999
PAir flow HL	22999.92
PAir flow LL	10000
PA Fan OUT	49.7543764
#Delta each scan	0.05
#MAX Delta	2.5

Fig. 87: Values and parameters for EventX 29

The purpose is to prevent the PA Fan running at the minimum or maximum output limits for long periods.

The PA Fan running at the minimum limit could lead to a too low Primary air flow resulting in air deficit, again resulting in high furnace temperature and excessive flue gas CO levels.

The PA Fan running at the maximum output limit could lead to a too high Primary air flow with excess air, not contributing to the primary combustion, resulting in falling furnace temperature and increased O_2 content.

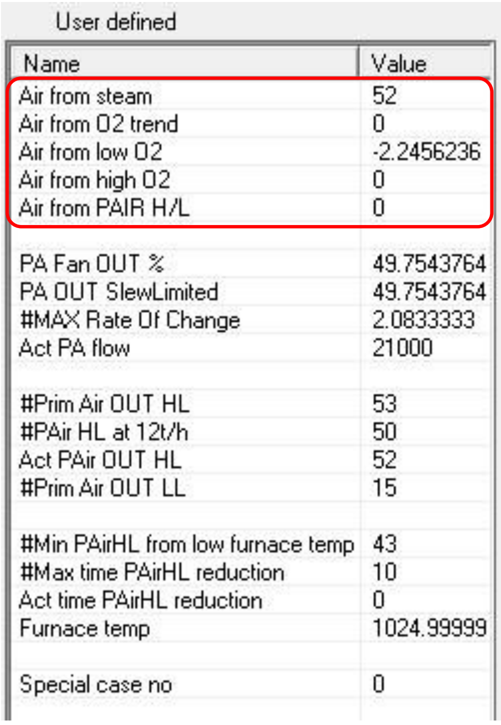
Primary air high limit	18000	Nm3/h	10720 Nm3/h
Primary air low limit	10000	Nm3/h	

Fig. 88: Primary air limits and actual Primary air flow

15.4.9 EventX 30: Primary Air control

This EventX summarizes the contributions to the set-point for primary air fan output that come from EventX 12 to 29, as shown in Fig. 75 on page 96. These contributions are marked on Fig. 89. The global variable `g_PA_FAN_OUT` (**PA Fan OUT %**) holds the summarized contributions which is limited between the actual high limit and the low limit.

To prevent too fast rises and falls in the fan speed set-point, `g_PA_FAN_OUT` is limited in slew rate to **#MAX rate of Change** in % per 10 seconds and stored in the output, `t_OV1_MC_001_OUT` in % (**PA OUT SlewLimited**).This is transmitted to the IVOO Scada and controls the Primary Air Fan VFD.



User defined

Name	Value
Air from steam	52
Air from O2 trend	0
Air from low O2	-2.2456236
Air from high O2	0
Air from PAIR H/L	0
PA Fan OUT %	49.7543764
PA OUT SlewLimited	49.7543764
#MAX Rate Of Change	2.0833333
Act PA flow	21000
#Prim Air OUT HL	53
#PAir HL at 12t/h	50
Act PAir OUT HL	52
#Prim Air OUT LL	15
#Min PAirHL from low furnace temp	43
#Max time PAirHL reduction	10
Act time PAirHL reduction	0
Furnace temp	1024.99999
Special case no	0

Fig. 89: Values and parameters for EventX 30

The actual high limit, `g_MASTER_OUT_HL` (**Act PAirOUT HL**) is calculated from parameters `g_MASTER_AIR_HL`, (**#Prim Air OUT HL**) and `g_MASTER_HL_AT_12`, (**#PAir HL at 12t/h**), depending on the Steam set-point. See Fig. 90.

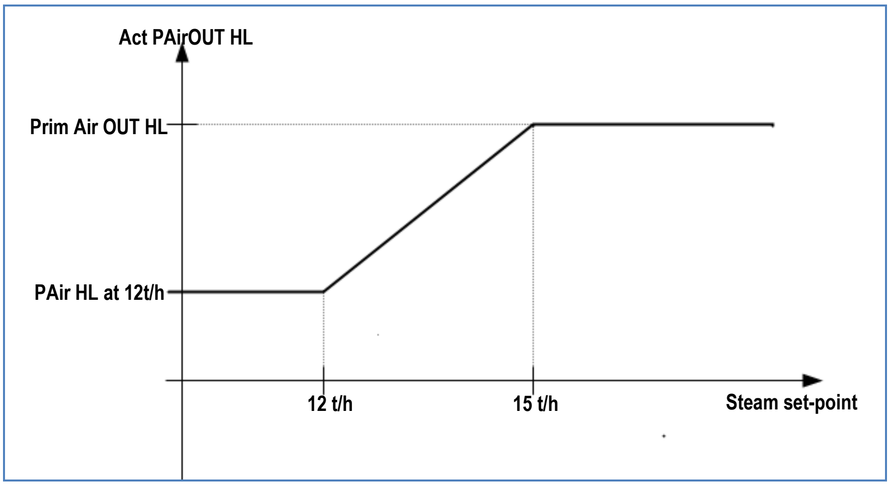


Fig. 90: Primary Air high limit in relation to Steam set-point

This script also handles four special cases indicated in **Special case no**:

1. If Steam flow is greater than (Steam set-point + 3.0 t/h), `g_PA_FAN_OUT` (**PA Fan OUT %**) is immediately reduced to 17 % plus the contributions from `g_D_MASTER_HIGH_O2` and `g_D_MASTER_LOW_O2`.
2. If Steam flow is greater than (Steam set-point + 5.0 t/h), `g_PA_FAN_OUT` (**PA Fan OUT %**) is immediately reduced to 13 % plus the contributions from `g_D_MASTER_HIGH_O2` and `g_D_MASTER_LOW_O2`.
3. If the Steam drum pressure, `t_OV1_PIA_022` is higher than Steam pressure high limit, `t_OV1_PIA_022_HL`, the for primary air fan output set-point, `t_OV1_MC_001_OUT`, is limited to 2 %, in reality stopping the Primary air fan

Steam pressure high limit	39.5 bar	37.3 bar

4. If Furnace Temperature falls below 800 °C, **Act PAirOUT HL**, `g_PA_FAN_OUT` (**PA Fan OUT %**) is gradually reduced, reaching `l_MIN_OUT_HL` (**#Min PAirHL from low furnace temp**) when Furnace Temperature falls to 750 °C. This reduction is restricted `l_MAX_MINUTES` (**#Max time PAirHL reduction**) minutes. The purpose is to prevent excess primary air not contributing to the primary combustion but actually cooling the furnace.
5. If the O2 content is below **O2 SP** + **Delta for low O2** from Fig. 86, `g_PA_FAN_OUT` (**PA Fan OUT %**) is gradually reduced, reaching `g_MASTER_AIR_LL`, (**#Prim Air OUT LL**) The purpose is to react fast to sudden increase in waste CV.
6. If the O2 content is above **O2 SP** + **Delta for high O2** from Fig. 86, `g_PA_FAN_OUT` (**PA Fan OUT %**) is gradually increased, reaching `g_MASTER_AIR_HL`, (**#Prim Air OUT HL**). The purpose is to react fast to sudden drops in waste CV.

15.4.10 EventX 39: Secondary Air Difference control

This script controls the amount of secondary air by the difference between the Total Air set-point and the actual primary air flow.

The Total Air set-point is given by the operator on the Plant Scada Combustion Line picture.

User defined

Name	Value
SA missing in TOTA...	7000.00000
Offset in TOTAL AIR	0
Actual SA flow	6487.34482
Error signal	512.655172
Scaled Error	0.17088505
#Scale HL	3000
#Scale LL	-3000
SA fan speed SP	62.6888452
Actual SA fan speed	0
#MC008 OUT HL	70.5
#MC008 OUT LL	35
OUT change since l...	0.03417701
Furnace Temp	1024.99999

Fig. 91: Values and parameters for EventX 39

EventX 39 is implemented as a normal PID-controller using the difference between the Total Air set-point and the actual primary air flow, **SA missing in TOTAL AIR** as set-point and **Actual SA flow** as process value. The **Error signal** is fuzzyfied in **Scaled Error** using **#Scale LL**, zero and **#Scale HL**. The output, **SA fan speed SP**, is limited between **#MC008 OUT HL** and **#MC008 OUT LL**, and transmitted to to the IVOO Scada and controls the Secondary Air Fan VFD.

If Furnace Temperature falls below 840 °C, the actual limiting values of **MC008 OUT HL** and **MC008 OUT LL** are gradually reduced, reaching 25 % when the Furnace Temperature falls to 800 °C. The purpose is to prevent excess secondary air, which is not contributing to the combustion, cooling the furnace, e.g. when burning heavy, wet waste.