Working with eval tags

Architecture

Calculation of values in D2000 is handled by the calculation process D2000 Calc. It has eval tags as descendants.

Most applications have a single calculation process (SELF.CLC) which calculates all eval tags.

Configuration

When configuring an eval tag, it is necessary to configure:

- **Value type:**
  - De - Boolean (True/False)
  - Ce - Integer
  - Ae - Analog
  - TmA - Absolute time
  - TmR - Time interval
  - Txt - Text
- **Calculation expression** - mathematical data processing that represents an expression into which the values of other D2000 objects enter (I/O tags, other eval tags, user variables, items of structured variables, historical values, and others). There are also buttons to insert various mathematical functions, constants, and attributes of object values into the expression.
  The Extended syntax for the expressions is also supported, where you can define auxiliary local variables and use labels and the GOTO statement.
- **Evaluation method:**
  - Periodic: with a defined period and offset
  - On change: the calculation is performed when the value of any of the objects in the expression is changed
  - Trigger: the calculation is performed when the value or value attribute of the selected object is changed
- **Maximum density of evaluation** - it is possible to limit the calculation density to 1 calculation per defined interval (number of seconds or milliseconds). If a reason (or reasons) for the calculation occurs again within the interval, the calculation is postponed until after the interval has expired and then performed.
- **Replace invalid values with 0** - Invalid values that would invalidate the entire calculation will be replaced by 0. If a different replacement value is needed, the %IsNull function or a conditional expression with the VLD attribute test can be used in the expression.

Structured eval tags

An eval tag is structured if a Destination column is defined (e.g. *SV.Something[0]^SomeColumn*). In such a case, the eval tag represents a "template" that is instantiated for each row of the destination structured variable, Wherever indices [0] of structured variables are used in the expression, they are replaced by indices 1, 2, etc. according to the number of rows of the destination structured variable).

An example of an expression for the sum of the columns of two different structures (the target column can be, for example, *SV.StructC[0]^PowerSum*:

```
SV.StructA[0]^PowerA + SV.StructB[0]^PowerB
```

A more complicated example of an expression using a nested conditional expression:

```
{ SV.Channel[0]^Input\VLD } ?           ; if the value is valid
[
  {SV.Channel[0]^Input =-32768} ?       ; -32768 is defined as invalid
  [%SetInvalid(0)]                      ; invalid
  :
  [SV.Channel[0]^Input * %Power(10, SV.Channel[0]^Exponent)]  ; normal value
]
:
[%SetInvalid(0)]                        ; invalid value (communication is not ok)
```

Tip

Debugging the eval tags:

A listing of detailed tuning information for a specific eval tag(s) can be shown with the SHOW_DYN_INFO <mask> command.

For structured eval tags, it is possible to compare the values of destination columns (structured variables) configured in eval tags with calculated values using the CHECK_DESTID_VALUES tell command.