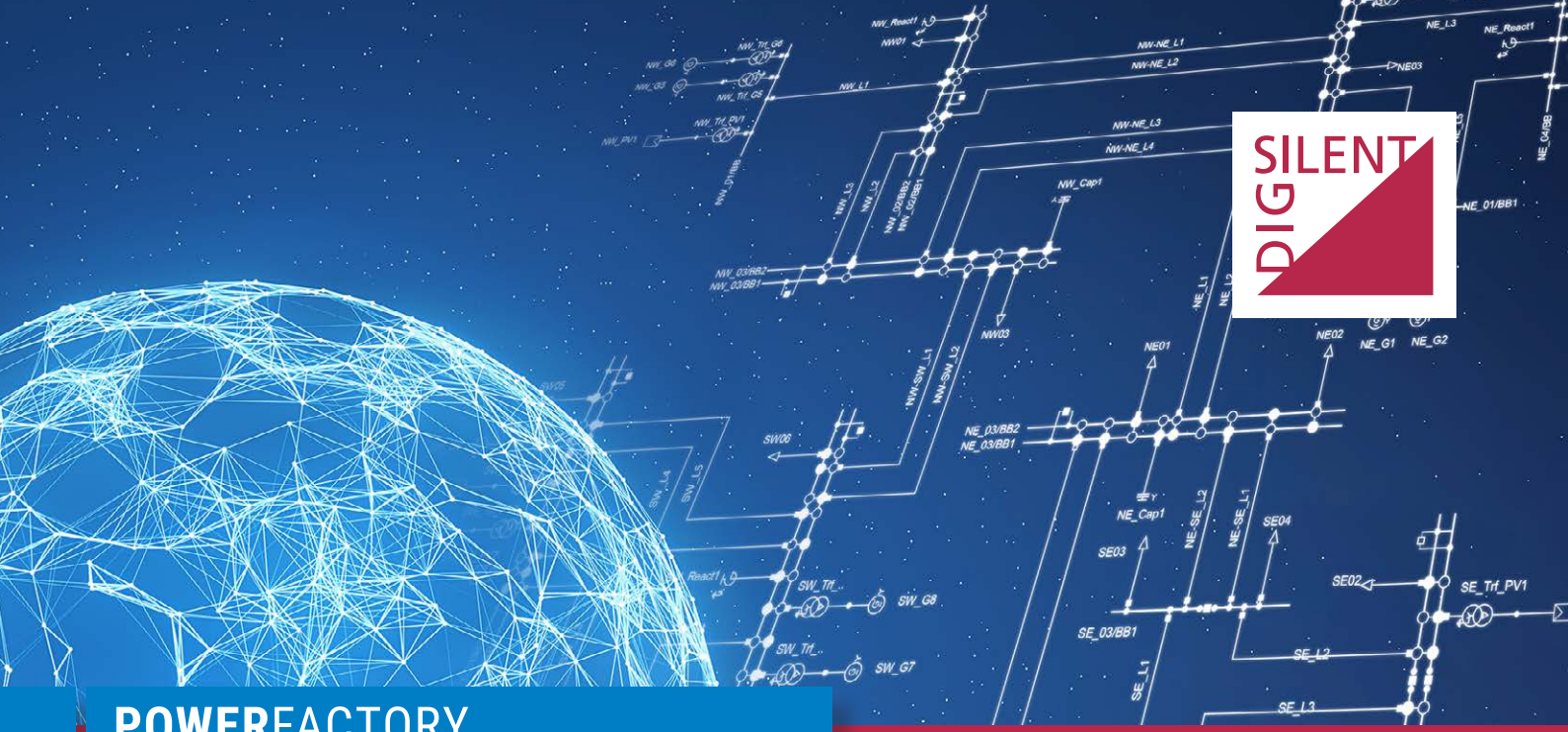

POWERFACTORY

PowerFactory 2021

Technical Reference

Logic/DIP

RelLogdip, TypLogdip

PF2021

POWER SYSTEM SOLUTIONS

MADE IN GERMANY

Publisher:
DIgSILENT GmbH
Heinrich-Hertz-Straße 9
72810 Gomaringen / Germany
Tel.: +49 (0) 7072-9168-0
Fax: +49 (0) 7072-9168-88
info@digsilent.de

Please visit our homepage at:
https://www.digsilent.de

December 1, 2020
PowerFactory 2021
Revision 1

Contents

1 General Description

The Logic/DIP object (*RelLogdip* class) is used to implement user defined trip logic, dip switch inputs and simplified calculation of analog signals. It supports a maximum of 64 input signals.

1.1 User defined logic type (*TypLogdip*)

The dialogue consists of two tab pages: the *Basic data* page and the *Logic* page. It allows defining the block behaviour and usage. Here below you can find a picture displaying the Logic/DIP *Basic data* tab page with the parameters set to implement an user defined trip logic and to provide a *PUTT* output signal (it can be used for an under reach distance protection scheme)

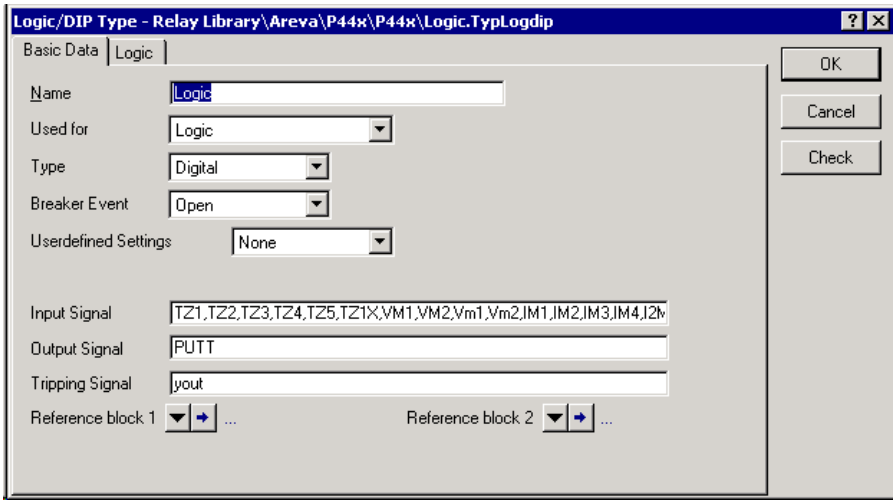

Figure 1.1: The Logic/DIP (*TypLogdip* class) *Basic data* tab page (the *default Dip* settings control is here not displayed)

1.1.1 Basic Data

Used for

The parameter (stored inside the *aUsage* variable) defines how the block is used. Six options are available: *Dip switch*, *Logic*, *Dip and logic*, *Switch logic*, *Dip + latched switch* and *Settings group logic*.

DIP Switch The block acts as a device dip switch. It can be used as input block. The dip switch status should be set in the *Logic* tab page in the Logic/DIP dialogue (*RelLogdip* class). The inputs are connected to the outputs through the dip switches.

Logic It allows defining any kind of user defined logic. This logic is connecting the input signals with the outputs. The dip switches inputs are disabled.

DIP and Logic When this option is activated the user can define a logic and can use the dip switch inputs at the same time.

Switch Logic It allows the user defining a *special* logic using the status of the associated breaker. At this purpose some additional pre-defined input signals are defined. _SWT1

represents the status of the breaker 1, _SWT2 the status of the breaker 2 etc (depending upon the number of the selected switches).

Dip + latched switch It is a special type of Dip switch (with the additional ability to use a reset signal): each dip switch can be set as Normally Closed or Normally Opened. Each output signal is changed from its *normal status* when the relevant input signal is on and the reset signal is off. The output signals are reset only when the reset signal is on.

Settings group logic It allows defining a logic having the purpose of changing the relay setting group.

Dip Commutator It allows selecting which input signal (or group of input signals) must be connected to an output signal (or to a group of output signals) .

Type

This parameter (stored inside the *aType* variable)defines how to process the user defined logic. Two different options are available: *Digital* and *Analog*.

- Select *Digital* to evaluate digital logical expression (AND, OR, NOT, NOR . . .)
- Select *Analog* to evaluate analog signals. Please note that when the *Analog* type has been selected the breaker can not be operated.

Breaker Event

This parameter (stored inside the *itrip* variable) defines which kind of breaker event is generated by the signals defined inside the *Tripping signal* parameter:

None When this option has been activated no event is created. It must be used when the user is going to define an *internal* logic not operating the breaker(s).

Open This option allows creating an open breaker event when the *Tripping signal* becomes true.

Close In this case a close breaker event is created when the *Tripping signal* becomes true. For instance it is used by the reclosing block of the relay models implementing a reclosing feature.

Userdefined Settings

This parameter (stored inside the *iusedef* variable) defines which controls are active at the user level in the *RelLogdip* dialogue. It can be: *None*, *only DIP*, *only Logic*, *Dip & Logic*.

None In the Logic/DIP element (*RelLogdip*) dialogue both the *Logic* tab page and the *Dip settings* tab page are disabled.

only DIP In the Logic/DIP element (*RelLogdip*) dialogue the user can change only the status of the dip using the *Dip settings* tab page.

only Logic In the Logic/DIP element (*RelLogdip*) dialogue the user can insert only the user defined logic in the Logic tab page.

Dip & Logic Both the status of the dip switches and the user defined logic can be modified by the user inside the Logic/DIP element (*RelLogdip*) dialogue

Default dip Setting

This parameter (stored inside the *aDefDip* variable) is visible only when the *Used for* parameter is *Dip Switch* or *Dip Logic* or *Dip + latched switch*. Here below a picture showing the Logic/DIP *Basic data* tab page with the parameters set to implement an user defined trip logic and a Dip switches input interface.

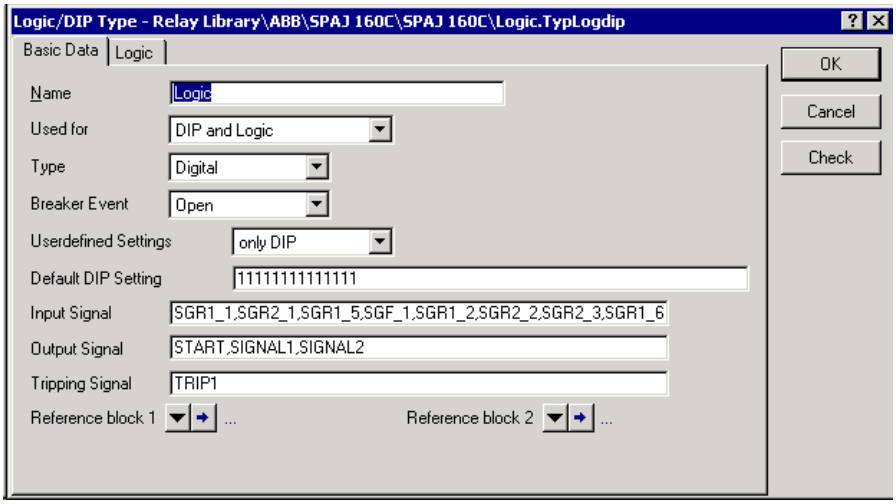

Figure 1.2: The *TypLogdip Basic data* tab page when the *Default Dip setting* control is visible

The *Default Dip setting* control contains the dip switches default values. To initialize each dip switch a number between 0 and 3 must be inserted. A row in the table in *Dip settings* tab page in the *RelLogdip* dialogue is added for each initialization number.

Input signals

This control should contain a comma separated list of input signal names (the list is stored inside the *sInput* variable); please note that semi-colons are not allowed to separate the input signal names. Moreover it is for the input and output signal name cannot start with a number (for instance 123I is not allowed).

Output signals

This control should contain a comma separated list of output signal names (the list is stored inside the sOutput variable); as for the *input signals* control the semi-colons are not allowed to separate the output signal names. The output signals are not used to open or close a break

Tripping signals

This control is displayed only when the *Breaker event* parameter is *Open* or *Close*. It should contain a comma separated list of possible tripping signal names (which are stored inside the sOutTrip variable). If the tripping signals set to true the logic automatically creates a breaker event and open or close the connected breakers.

textbfReference block 1 and Reference block 2

These controls can be used to define one or two references to other block belonging to the same model. If the name of a input signals starts with *_REF1* and a *C* is available, the block is trying to assign to the input the value of one of the settings located inside the *Reference block 1*. For instance the *_REF1K0* input is loaded with the value of the *k0* setting if a *TypZpol* block is the *Reference block 1*.

1.1.2 Logic

The Logic/DIP dialogue *Logic* tab page allows inserting an user defined logic. Here below in the Figure 1.3 an example of user logic definition: in this case 4 logic are defined for the *START*, *SIGNAL1* and *SIGNAL2* output signal and for the *TRIP1* tripping signal.

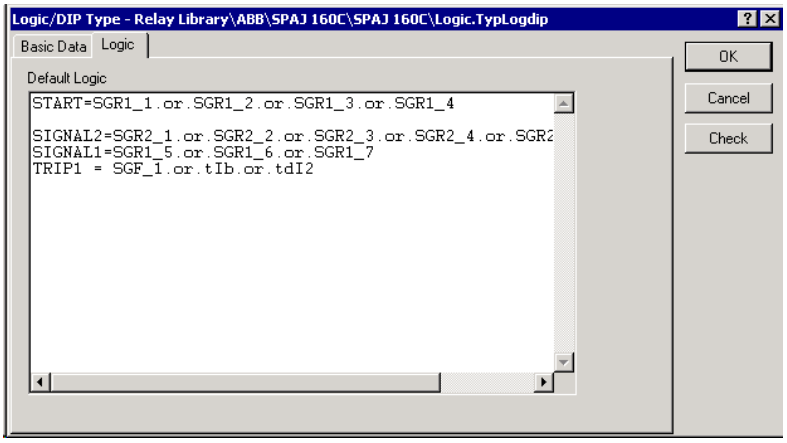

Figure 1.3: The *TypLogdip* Logic tab page

The *Check* button on the right side of the dialogue should be used to check the user defined expression and detect syntax errors.

1.2 User defined logic (*RelLogdip*)

The layout of the Logic/DIP block (*RelLogdip* class) depends upon the Logic/DIP type (*TypLogdip* class) parameters. In the following pages the available tab pages will be described.

1.2.1 Basic data

This page provide the normal features allowing to insert the block name (*Name edit box*), put the block out of service (*Out of Service* checkbox) and browse the relevant *TypDiplog* block (type control).

When in the Logic/DIP type dialogue (*TypLogdip* class) the *Breaker event* parameter is *Open* or *Close* the *Breaker* table is displayed (see Figure 1.4 here below). To associate a breaker to the user defined logic a reference to a breaker must be inserted inside the *Open* column. More than one line can be added; each line that has been inserted should store inside the *Open* column a link to a different breaker. The empty line showed in the Figure 4 represent the *default breaker*: the Logic/DIP block uses automatically the breaker located inside the cubicle where the relay is installed. To avoid a breaker operation is possible to use the *Out of Service* option. A breaker failure or a signal failure from the relay to the breaker can be simulated simply enabling the relevant the *Out of Service* option.

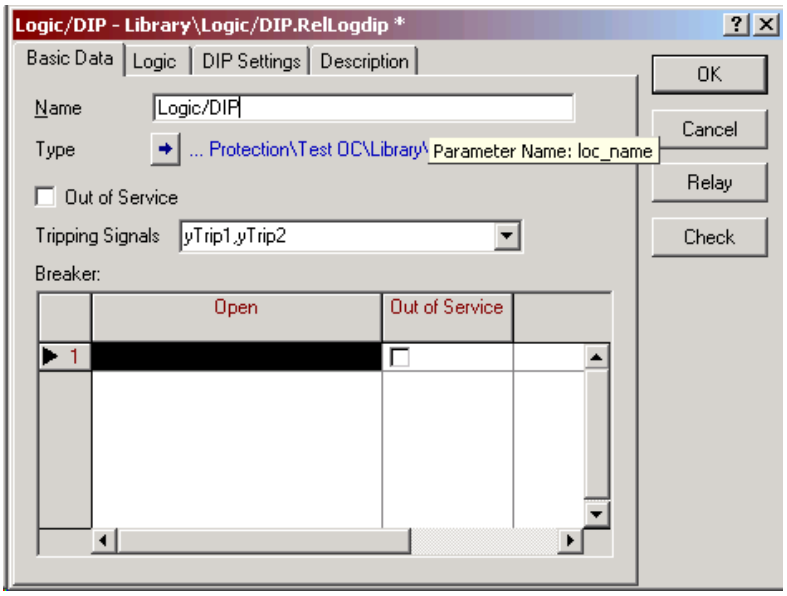

Figure 1.4: The Logic/DIP *Basic data* tab page when the *Breaker event* inside the Logic/Dip type dialogue is *Open* or *Close*

Please note here above in the Figure 1.4 the *Tripping signals* listbox. This list box is displayed only when more than one tripping signal is defined in the Logic/DIP type (*TypLogdip* class) in the *Tripping signals* control. Inside the list box an item for each tripping signal and for each possible combination of the tripping signals is available. In the Figure 1.4 for example the tripping signal *yTrip1* or the tripping signal *yTrip2* leads to a breaker event.

1.2.2 User defined logic

The *Logic* table in the Logic/DIP dialogue (RelDiplog class) is displayed only if the *Userdefined settings* parameter in the Logic/DIP type dialogue (*TypDiplog* class) is *only Logic* or *Dip & Logic*.

Inside this page the user can overwrite the logic located inside the *Logic* tab page of the Logic/DIP type dialogue (*TypDiplog* class). It should be used to support relay models where the user has the ability to define logical equation using internal variables (i.e. The Schweitzer SEL devices)

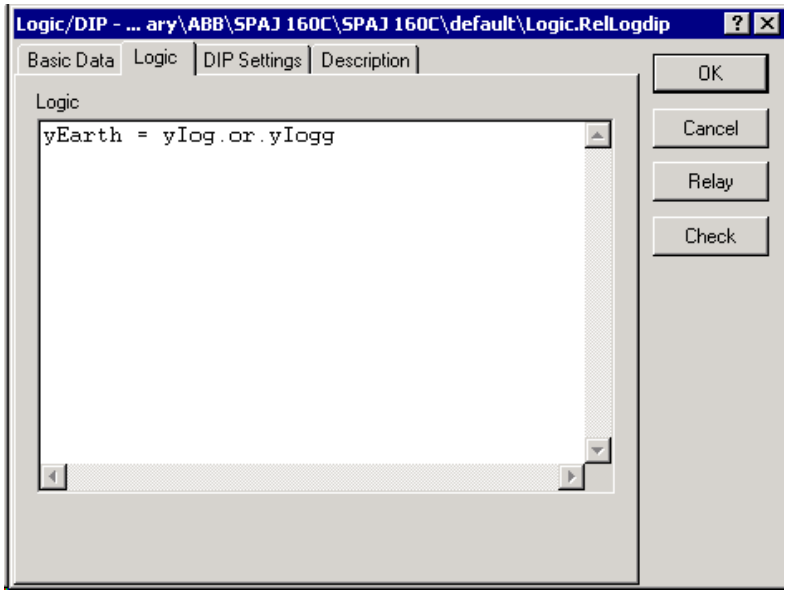

Figure 1.5: The Logic/DIP *Logic* tab page

The assignment of an output signal or of a tripping signal has higher priority and overwrites the line which are defined in the logic type.

Userdefined Logic in the Logic/DIP type dialogue (*TypLogdip* class)	**User defined Logic in the Logic/DIP element dialogue (*RelLogdip* class)**

`yPhase = yIg.or.yIgg.or.yIggg`
`yEarth = yIog.or.yIogg.or.yIoggg`
`yTrip = yPhase.or.yEarth`
	`yEarth = yIog.or.yIogg`

Active user defined logic	

`yPhase = yIg.or.yIgg.or.yIggg`
`yEarth = yIog.or.yIogg`
`yTrip = yPhase.or.yEarth`	

The *Check* button on the right side of the dialogue should be used to check the logical expression and detect syntax errors.

1.2.3 User defined DIP settings

The *Dip settings* tab page allows the user configuring the dip switches status. The table is displayed only if the *Userdefined settings* parameter in the Logic/DIP type dialogue (*TypDiplog* class) is *only DIP* or *Dip & Logic*.

The dipswitch names located in the first column of the table are the first *Input signals* names listed in the Logic/DIP type dialogue (*TypDiplog* class). The number of characters present inside the *Default DIP settings* control in the Logic/DIP type dialogue (*TypDiplog* class) defines the number of available dip switches listed in the *Dip Switch* table.

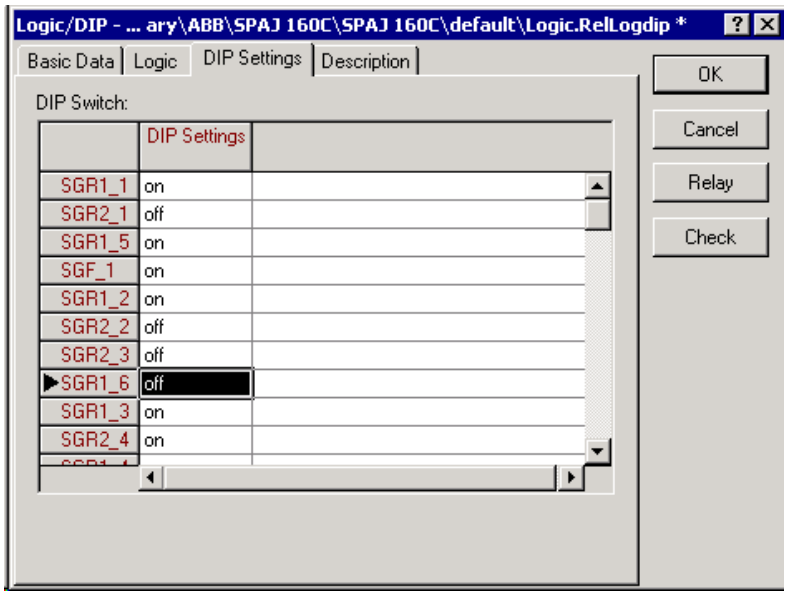

Figure 1.6: The Logic/DIP *Dip Settings* tab page

Double clicking on a cell of the *DIP Settings* column inside the *Dip Switch* table the window displayed in Figure 1.7 is opened.

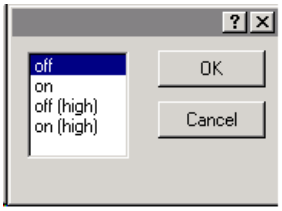

Figure 1.7: The *DIP switch* table selection window

Each dip switch can be set *off* or *on* or *off (high)* or *on (high)*. The *off (high)* and the *on (high)* option can be used when the relevant output signal must be set equal to 1 also when the input signal is not connected.

When inside the Logic/Dip type dialogue the *Used for* parameter is set equal to *DIP + Latched Switch* two additional options are available: *on (NC+latched)* and *on (NO+latched)*. Their meaning is the follow:

on (NC+latched) the switch is opened when the input signal is on and there is no reset signal (or in steady state the reset signal is greater than the relevant input signal)

on (NO+latched) the switch is closed when the input signal is on and there is no reset signal (or in steady state the reset signal is greater than the relevant input signal)

2 Using the block

In this section more details are provided regarding the operation mode corresponding to each value of the *Used for* parameter (*TyLogdip* class)

2.1 DIP Switch

When *Used for* is equal to *DIP Switch* the user can switch on and off the signals using the logic represented in the following figure:

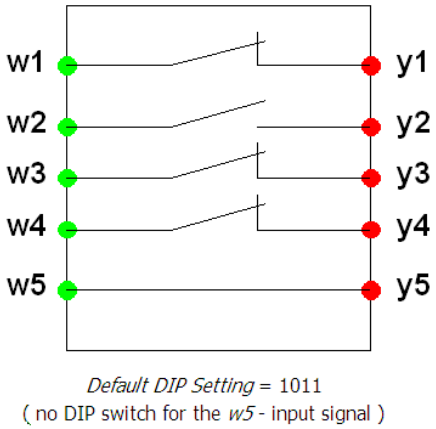

Figure 2.1: DIP Switch Example

The state of the DIP switches can be entered inside the *Default DIP Setting* parameter (stored inside the *aDefDip* variable) in the Logic/DIP type dialogue (*TypLogdip* class). For each input signal a number between 0 and 5 should be entered. If the number of input signals is greater than the number of the characters in the *Default DIP Setting* parameter, then no DIP switch is operating between the input and the output signal. This is the case for instance for *w5* input signal in Figure 2.1.

The *Default DIP setting* parameter defines the default values of the DIP switch status. The meaning of the values is listed here below:

0 = off	the switch is open	if the input signal is not connected, the output is set to low (false or 0)
1 = on	the switch is closed	if the input signal is not connected, the output is set to low (false or 0)
2 = off(high)	the switch is open	if the input signal is not connected, the output is set to high (true or 1)
3 = on(high)	the switch is closed	if the input signal is not connected, the output is set to high (true or 1)

The number of the input signals must be equal to the number of the output signals. Inside the *Input signals* control in the Logic/DIP dialogue (*TypDiplog* class) the signals are separated with a comma, semi-colon are not allowed as separators.

2.2 Logic

When *Used for* is equal to *Logic* the block can be used to calculate user defined digital logic (with *Type* equal to *Digital*) or simple operations using analog signals (with *Type* equal to *Digital*).

2.2.1 Type: Analog

The main purpose for the *Analog* type is performing simple operations like summating two currents of two different current transformers or modifying the magnitude of a signal. Only the basic mathematical expressions are available.

Example:

sum1 = wIn1 + wIn2	build a sum of input signal wIn1 and wIn2
negesum2 = wIn1 - wIn2	subtract wIn1 and wIn2

2.2.2 Type: Digital

The *Digital* type is used to handle logical expressions, including for instance AND, OR. For steady state calculations (load flow, short-circuit) digital signals are normally tripping times. The parser interprets then the OR operator as min() and the AND operator as max(). The mathematical operator $+$ is mapped as *OR* and the multiplier $*$ as *AND*.

Example:

Trip = y1 * y2	is equal to	Trip = y1.and.y2
Trip = y1 + y2	is equal to	Trip = y1.or.y2

Moreover two pre-defined values are available:

TRIP	: for steady state calculation = 0 seconds
	: for time domain simulations = 1 (high)
NOTRIP	: for steady state calculation = 9999.999 seconds
	: for time domain simulations = 0 (low)

2.3 DIP & Logic

When *Used for* is equal to *DIP & Logic* the block is performing a combination of *DIP Switch* and *Logic*. Each input signal is *reaching* the *internal* signal which can be used by the user defined logic only if the corresponding DIP switch is closed. The *internal* signals can be evaluated by the user defined logical expressions to figure out the output and the tripping signal values.

2.4 Switch Logic

When *Used for* is equal to *Switch Logic* the user defined logic can use the breaker status. Accordingly with the number of selected breakers in the Logic/DIP element (RelLogdip class) for each breaker an input signal is created internally. The name of the created input signals is *_SWT1*, *_SWT2*, . . . *_SWTn*. If the breaker is closed the relevant input signal is set to 1 (for time domain simulations) or set to 0 seconds for steady state calculations. The user defined logic

can use these signals to set the output and the tripping signals. A typical example is to use the *Switch Logic* for the reclosers, where the reclosing logic must know if the breaker is open or closed.

2.5 DIP + Latched switch

When *Used for* is equal to *DIP + Latched switch* the block is working as when *Used for* is equal to *Dip switch* but the behaviour is slightly different: a reset signal can be used (it is automatically defined as *_RESET*, so the user doesn't have to add it manually in *Input signals* control) and two additional options (*on (NC+latched)* and *on (NO+latched)*) are available in the *Dip switch* table in the *Switch Settings* tab page of the Logic/DIP dialogue (RelLogdip class).

The *Default DIP setting* parameter defines the default values of the DIP switch status. The meaning of the values is listed here below:

0 = off	the switch is open if the input signal is not connected, the output is set to low (false or 0)
1 = on	the switch is closed if the input signal is not connected, the output is set to low (false or 0)
4 = on (NC+latched)	the switch is opened when the input signal is on and there is no reset signal (or in steady state the reset signal is greater than the relevant input signal)
5 = on (NO+latched)	the switch is closed when the input signal is on and there is no reset signal (or in steady state the reset signal is greater than the relevant input signal)

The switches are reset only when the reset signal is on (The *NC+latched* switches are closed and the *NO+latched* switches are opened).

The number of the input signals must be equal to the number of the output signals. Inside the *Input signals* control in the Logic/DIP dialogue (*TypDiplog* class) the signals are separated with a comma, semi-colon are not allowed as separators.

2.6 Settings Group Logic

When *Used for* is equal to *Setting Group Logic* the block is evaluating an user defined logic but instead of tripping the breaker the result is used to change the relay active setting group. For this reason the *Breaker event* parameter in the Logi/DIP dialogue (*TypDiplog* class) must be set equal to *None*. In the *Output signals* control each setting group must be represented by an output signal. In the *Logic* tab page the logical equations to activate each setting group must be associated to the output signals.

Here below an example where the active group is controlled by 2 input signals:

Input Signals:	Input1, Input2
Output Signals:	Group1, Group2, Group 3
Logic:	Group 1 = Input1.and.Input2
	Group 2 = Input1.and.{.not.Input2}
	Group 3 = {.not.Input1}.and.Input2

2.7 DIP Commutator

When *Used for* is equal to *DIP Commutator* the user using the dip switches can select which input signal (or group of input signals) must be connected to the output signal (or to a group of output signals).

The number of DIP switches is defined inside the *Default DIP Setting* parameter (stored inside the *aDefDip* variable) in the Logic/DIP type dialogue (*TypLogdip* class). Each number defines a dip switch. The number of dip switches must be equal to number of inputs divided by the number of outputs. If it is not equal a error message is displayed when the user tries to close the Logic/DIP type dialogue (*TypLogdip* class).

3 Logic/DIP input parameters definition

3.1 Logic/DIP type (*TypLogdip*)

Parameter	Description	Unit
loc_name	Name assigned by the user to the block type	Text
aUsage	Parameter defining how the block is used. It can be *Dip switch* or *Logic* or *Dip and logic* or *Switch logic* or *Dip + latched switch* or *Settings group logic*.	Text
aType	The type of calculation used to process the user logic. It can be *Digital* or *Analog*	Text
Itrip	The type of operation performed on the associated breaker. It can be *None* or *Open* or *Close*	Integer
iusedef	The active control available in the RelDiplog dialogue. It can be *None* or *only DIP* or *only Logic* or *Dip & Logic*	Integer
aDefDip	A string containing the initialization values of the dip switches	Text
sInput	The names of the input signals (the names must be separated by a colon)	Text
sOutput	The names of the output signals (the names must be separated by a colon)	Text
sOutTrip	The names of the signals used to operate the breaker (the names must be separated by a colon)	Text
prefblock1	Reference to a block which is used to assign a value to input signals having name starting with *_REF1*	Reference
Prefblock2	Reference to a block which is used to assign a value to input signals having name starting with *_REF2*	Reference
sDefLogic	The user defined logic definition	Text

3.2 Logic/DIP Element (*RelLogdip*)

Parameter	Description	Unit
loc_name	Name assigned to the user to the block element	Text
Typ_id	Pointer to the relevant *TypLogdip* object	Pointer
Outserv	Flag to put out of service the block	Y/N
sTripsig	It allows defining which trip signals are operating the breaker. The relevant combobox is displayed only if more than one trip signal is available	Text
pSwitch	An array of reference to the breakers which will be operated by the Rellogdipblock	Array of object reference
iout	An array of integer which are enabling(1) or disabling(0) the ability of the RelLogdip block to operate the relevant breaker	Array of integer
sLogic	The user defined logic (it overwrite the logic defined inside *TypLogdip*)	Text
aDipset	A table containing the input dip switch status	Array of integer

4 Signals definition

Input Signal	Example
All the variable defined by the user inside the *Input signals* control (sInput variable)	TZ1, TZ2, TZ3, TZ4, TZ5, TZ1X, VM1, VM2

Output Signal	Example
All the variable defined by the user inside the *Output signals* control (sOutput variable)	Putt
All the variable defined by the user inside the *Tripping signals* control (sOutTrip variable)	yout