PROGRAMOVATELNÉ AUTOMATY

CONTROL LIBRARIES

FOR MOSAIC

TXV 003 23.02

CONTROL LIBRARIES FOR MOSAIC

2nd edition – december 2006

CONTENT

1. INTRODUCTION

Function´s and function block´s libraries are an inseparable part of a programmable Mosaic environment installation. In term of their conception, libraries can be divided into following types:

- Built-in libraries
- Standardly supplied external libraries
- User-defined libraries

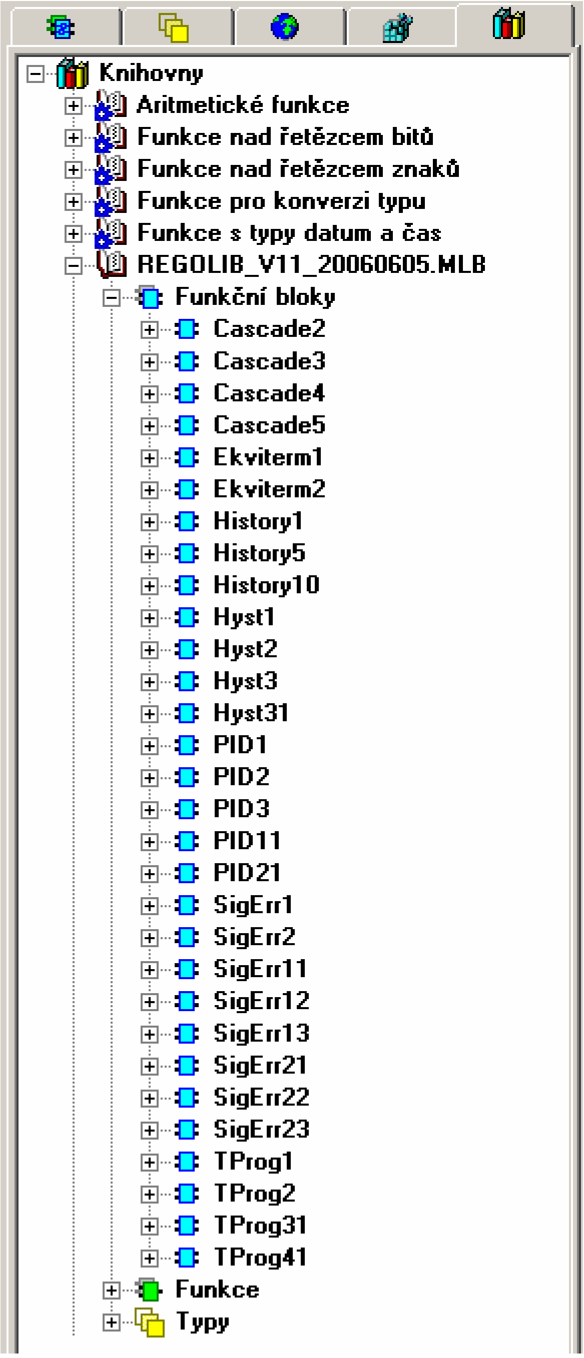
The library can contain function declaration, declaration of function blocks, data´s types and global variables.

2. REGOLIB LIBRARY

Regulation library RegoLib.mlb contains basic function blocks that are used especially for regulation task´s solutions within PLC Tecomat. However, number of function blocks of this library have also a wide range of use outside regulation tasks.

The basis of the RegoLib library is a set of selected components from a programmable environment Merkur. In particular, these are function blocks of PID,weather-compensated and hysteresis regulations, cascading, fault signallizations, fault history and schedule programs.

The following picture shows the structure of the RegoLib library in the Mosaic environment.



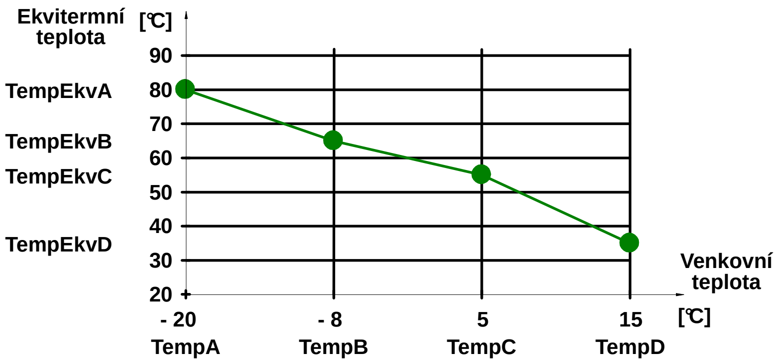
Obr. 2.1 RegoLib Library

2.1. WEATHER-COMPENSATED CURVES

2.1.1. Ekviterm1 – weather-compensated curve with fixed points of outdoor temperature

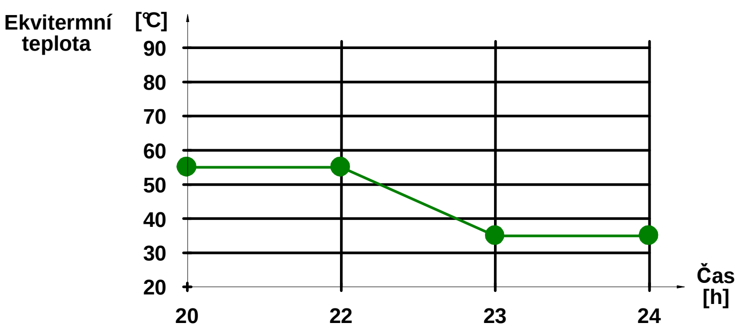
Ekviterm1 function block executes a calculation of a requested temperature *OUT* on the basis of measured outdoor temperature *IN*, set parameters of four-point weather-compensated curve and a requirement on attenuation program.

Parameters *TempEkvA*-*TempEkvD* enable to move the set weather-compensated curve in vertical direction. Temperatures of weather-compensated curve breakpoint are firmly set on *TempA* = -20℃, *TempB* = -8℃, *TempC* = +5℃, *TempD* = +15℃. Maximum calculated temperature according to the weather-compensated curve is equal to the *TempEkvA* point, the minimum is equal to *TempEkvD* point.



Obr. 2.2 Weather-compensated curve

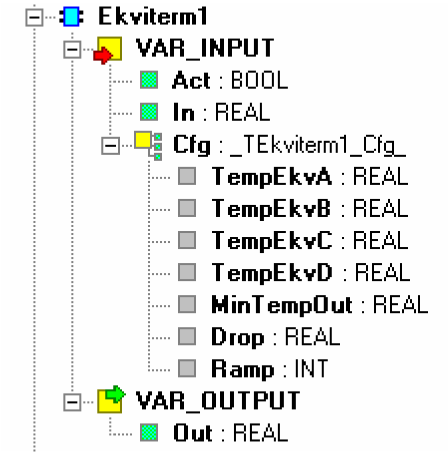
The calculated weather-compensated temperature can be lessened by a set attenuation, however, not more than it is determined by a parameter of a minimum output temperature *MinTempOut.* If the input variable *Act* is in log.0, then the calculated weather-compensated temperature is gradually decreased according to a ramp function to a level of a weather-compensated value lessened by a set attenuation *Drop.* If the variable *Act* changes from log.0 to log.1, then again the output temperature will according to the ramp function be increased to a level of the calculated weather-compensated temperature. The period during which is this value reached is due to a parameter *Ramp*. The function of attenuation algorithm is ilustrated by the following picture.

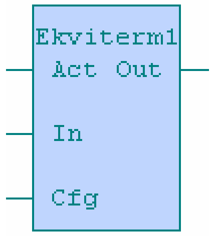


Obr. 2.3 Attenuation curve

The graph shows a course of output weather-compensated temperature *Out* during a transition to the attenuated operation for:

Calculated weather-compensated temperature *OUT* = 55℃,
Size of attenuation *DROP* = 20℃, *RAMP* = 60 min
Heating disconnection at 22.00 o´clock.





Obr. 2.4 The structure of FB Ekviterm1 *Obr. 2.5 The appearance of FB Ekviterm1*

Variables description :

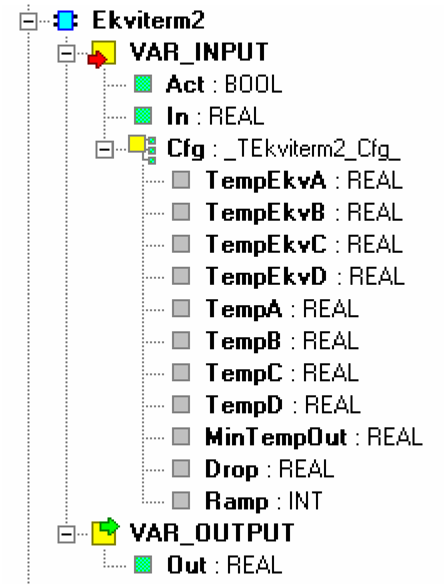
Term		**Signification**	**Type**	**Format**
Act		Heating operation (1/0 – topit/útlum)		bool
In		Outdoor temperature [℃]		real
Cfg		Configuration block parameters		_TEkviterm1_Cfg_
	.TempEkvA	Required weather-compensated temperature at outdoor temperature point A (-20) [℃]		real
	.TempEkvB	Required weather-compensated temperature at outdoor temperature point B (-8) [℃]		real
	.TempEkvC	Required weather-compensated temperature at outdoor temperature point C (+5) [℃]	output	real
	.TempEkvD	Required weather-compensated temperature at outdoor temperature point D (+15) [℃]		real
	.MinTempOut	Minimum weather-compensated output temperature [℃]		real
	.Drop	Output temperature attenuation [℃]		real
	.Ramp	Time-lag of initiation and termination of attenuated operation [min]		int
Out		Output weather-compensated temperature [℃]	output	real

2.1.2. Ekviterm2 – weather-compensated curve with adjustable points of outdoor temperature

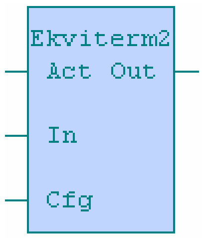
Ekviterm2 function block executes a calculation of a required temperature *OUT* on the basis of measured outdoor temperature *IN*, set parameters of four-point weather-compensated curve and a requirement on attenuation program.

With its functions the Ekviterm2 is similar to Ekviterm1, the only diference is that temperatures of weather-compensated curve breakpoint *TempA – TempD* are optional.

Parameters *TempEkvA-TempEkvD* enable to move the set weather-compensated curve in vertical direction and parameters TempA-TempD in horizontal direction.



Obr. 2.6 The structure of FB Ekviterm2



Obr. 2.7 The appearance of FB Ekviterm2

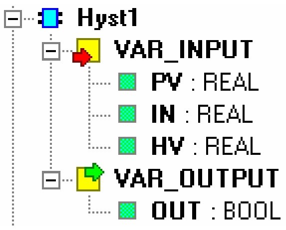
Variables description:

Term		**Signification**	**Type**	**Format**
Act		Heating operation (1/0 – heat/attenuation)	Output	bool
In		Outdoor temperature [℃]		real
Cfg		Configuration block parameters		_TEkviterm2_Cfg_
	.TempEkvA	Required weather-compensated temperature at outdoor temperature point A [℃]		real
	.TempEkvB	Required weather-compensated temperature at outdoor temperature point B [℃]		real
	.TempEkvC	Required weather-compensated temperature at outdoor temperature point C [℃]		real
	.TempEkvD	Required weather-compensated temperature at outdoor temperature point D [℃]		real
	.TempA	Required outdoor temperature point A [℃]		real
	.TempB	Required outdoor temperature point B [℃]		real
	.TempC	Required outdoor temperature point C [℃]		real
	.TempD	Required outdoor temperature point D [℃]		real
	.MinTempOut	Minimum weather-compensated output temperature [℃]		real
	.Drop	Output temperature attenuation [℃]		real
	.Ramp	Time-lag of initiation and termination of attenuated operation [min]		int
Out		Output weather-compensated temperature [℃]	Output	real

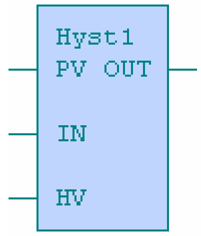
2.2. HYSTERESIS CONTROLLERS

2.2.1. Hyst1 - hysteresis

Hyst1 function block will realize a comparison of measured value *IN* with the set required value *PV* and hysteresis *HV* under consideration. If the measured value *IN* is higher than the required value PV+HV/2, then the output signal *OUT* is set to log.1. If the measured value *IN* is lower than the required value PV-HV/2, then the output signal *OUT* is set to log.0.



Obr. 2.8 The sturcture of FB Hyst1



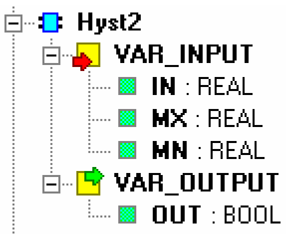
Obr. 2.9 The appearance of FB Hyst1

Variables description:

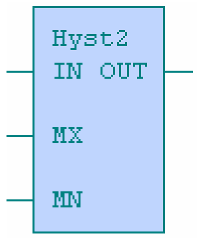
Term	**Signification**	**Type**	**Format**
PV	Required value		real
IN	Measured value	Input	real
HV	Hysteresis		real
OUT	Output value	Output	bool

2.2.2. Hyst2 - hysteresis MIN,MAX

If the measured value *IN* exeeds the required maximum value *MX*, then the output binary signal *OUT* is set to log.1. In case that the measured value *IN* falls below the required minimum value *MN*, then the output signal *OUT* is set to log.0.



Obr. 2.10 The structure of FB Hyst2



Obr. 2.11 The appearance of FB Hyst2

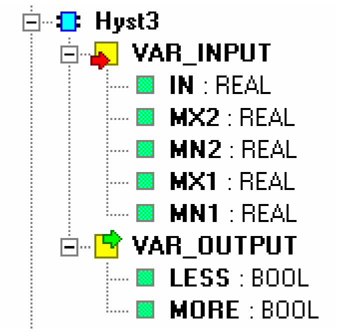
Variables description:

Term	**Signification**	**Type**	**Format**
IN	Measured value		real
MX	Maximum	Input	real
MN	Minimum		real
OUT	Output value	Output	bool

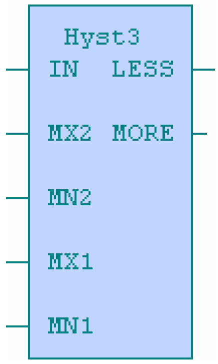
2.2.3. Hyst3 – double hysteresis MIN,MAX

If the measured value *IN* exeeds the required maximum value *MX2*, then the output binary signal *LESS* is set to log.1. In case that the measured value *IN* falls below the required minimum value *MN2*, then the output binary signal *MORE* is set to log.0.

If the measured value *IN* falls below the required minimum value *MN1*, then the output binary signal *MORE* is set to log.1. In case that the measured value *IN* exceeds the required maximum value *MX1*, then the output signal *MORE* is set to log.0.



Obr. 2.12 The structure of FB Hyst3



Obr. 2.13 The appearance of FB Hyst3

Variables description:

Term	**Signification**	**Type**	**Format**
IN	Measured value	Input	real
MX2	maximum 2		real
MN2	minimum 2		real
MX1	maximum1		real
MN1	minimum 1		real
LESS	Output „less“	Output	bool
MORE	Output „more“		bool

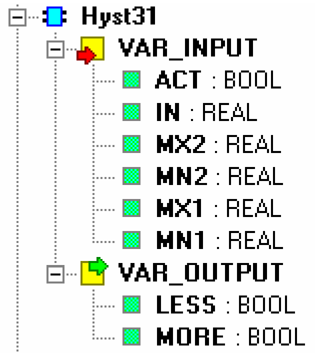
2.2.4. Hyst31 - double hysteresis MIN,MAX with control variable

If the input variable *ACT* is in log.1, then this function block is functioning in the same way as Hyst3, it means:

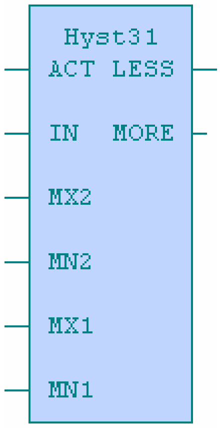
If the measured value *IN* exeeds the required maximum value *MX2*, then the output binary signal *LESS* is set to log.1. In case that the measured value *IN* falls below the required minimum value *MN2*, then the output binary signal *MORE* is set to log.0.

If the measured value *IN* falls below the required minimum value *MN1*, then the output binary signal *MORE* is set to log.1. In case that the measured value *IN* exceeds the required maximum value *MX1*, then the output signal *MORE* is set to log.0.

If the input variable *ACT* is in log.0, then also outputs *LESS* and *MORE* are set to log.0.



Obr. 2.14 The structure of FB Hyst31



Obr. 2.15 The appearance of FB Hyst31

Variables description:

Term	**Signification**	**Type**	**Format**
ACT	Control variable	Input	bool
IN	Measured value		real
MX2	maximum 2		real
MN2	minimum 2		real
MX1	maximum1		real
MN1	minimum 1		real
LESS	Output „less“	Output	bool
MORE	Output „more“		bool

2.3. PID CONTROLLERS

2.3.1. General description

PID controller in PLC Tecomat systems is implemented directly in a command file of individual PLCs. Individual function blocks of PID regulations work with this command in order to simplify the settings and operating of the controller for the programmer. They offer, from the whole structure of the controller, only selected variables and set implicit values to some variables.
The controller works according to a discrete version of this formula:

$$u(t) = K * \left[e(t) + \frac{1}{Ti} \int_0^t e(\tau) d\tau + Td * \frac{de(t)}{dt} \right]$$

General structure of variables:

MinY - minimum measured value, used for normalization of a deviation

MaxY -maximum measured value, used for normalization of a deviation

Input1 - measured (regulated) quantity

gW - required value, lying within measured value subrange <MinY,MaxY>

tiW - time response for filtr of the 1st order or linear interpolation of required value in multiples of automaton´s cycles

ConW - current required value

Dev (e) - the deviation of true value from required value [%]

Output - output set by an algorithm or manually. The action intervention can lie within maximum of -10000 up to +10000 (i.e.-100,00% up to +100,00%) range. Thus it is regulated so, that for gain 1 (zone of proportionality 100%) and deviation 100,00% is an interference 100,00%. The range is always limited by a subrange <MinU, MaxU>.

LastOut - last action interference, i.e. 1 step delayed [%] or valve possition

CurOut - output actually required in given step [%] or action interference gain

ConOut - output realized by the controller [%] or an actual value realized by output unit or by time proportional control on/off in an absolute value

DefOut - implicite output value at measurement error

MinU - minimum granted action interference 0-10000 [0-100,00 %]. The direct action interference can not be lower than this value.

MaxU -maximum granted action interference 0-10000 [0-100,00 %]. The direct action interference can not be higher than this value.

dMaxU - maximum granted action interference gain 0-10000 [0-100,00 %]. The new action interference can not in the absolute value differ from the last value by more than dMaxU

OutCycle - lenght of an output cycle, period of sampling [hundredths of sec] in range from 1 up to 60000 (i.e. 10 ms up to 10 min by 10 ms). It determines the period in which the action interference is not changing, or more precisely the period of frequency rate for time proportional control

Pbnd - range of proportionality, it is set in a range from 1 up to 30000 (0,1up to 3000,0%). It determines the gain coefficient by correlation:

$$K = \frac{1000}{PBnd}$$

RelCool - auxiliary range of proportionality for negative deviation, it is set in a range from 1 up to 30000 (0,1 up to 3000,0%). The gain coefficient is then determined by correlation:

$$K = \frac{1000}{PBnd} * \frac{1000}{\mathrm{Re}\,lCool}$$

Ti - constant of integration, it is set in a range from 1 up to 30000 (0,1 up to 3000,0 s). For zero value is the constant of integration switched off.

Td - derivative constant, it is set in a range from 1 up to 30000(0,1 up to 3000,0 s).

Egap - symmetric zone of insensibility, it is set in a range from 0 up to 10000 (0 up to 100,00%). If the deviation is lower than EGap, the action interference remains unchanged.

Dgap - symmetric deviation zone where the derivative constant is functioning, range is from 0 up to 10000 (0 up to 100,00%). It means that derivative constant functions constantly at DGap = 10000 .

Igap - symmetric deviation zone where the constant of integration is functioning, range is from 0 up to 10000 (0 up to 100,00%). It means that constant of integration functions constantly at IGap = 10000 .

Control - control word used for setting the controller´s activity. The controller can be in automatic, manual or emergency mode. It can function as a controller with direct or incremental algorithm. If a servo-operated valve is used as an action member, it is possible to use for action interference correction the measured value of its possition, i.e. cascade control. Under the condition of longer output cycle it is possible to realize time proportional output control on/off. The resolution is set by automaton cycle period. E.g. if the automaton cycle period is 100ms and output cycle 10s then the resolution is 1%.

.15	.14	.13	.12	.11	.10	.9	.8	.7	.6	.5	.4	.3	.2	.1	.0
FU2	FU1	FU0	-	-	P41	RIO	RF	HR	AM	IP	BU	KC	A12	AO	RC

RC	- 1 – required cold start of controller (component itself reset the bit)
AO	- 1 – zero movement of controller output in range 4-20mA
A12	- 1 – output on to twelve-bit converter D/A
KC	- 1 - cascade control
BU	- 0 – unified output 1 – binary output (time proportional control on/off)
IP	- 0 – direct control 1 – incremental control
AM	- 0 – manual mode 1 – automatic mode
HR	- 1 – more reliable measurement mode, two measured values used
RF	- 0 – adjustment of required value via filtr of 1st order 1 - adjustment of required value via linear interpolation
RIO	- 1 – ratio control
P41	- 1 - PID instruction is invoked in process P41, i.e. in a grid 10 ms

Note:In this case also it is possible to use the setting of period for variable OutCycle.This has a practical mening for type of control on / off only. (e.g. if OutCycle = 100, the period is 1s and width resolution of output pulse is 10ms ,i.e. 1%.While using e.g. output unit, it is possible to realize the output of cyclic control type this way, i.e. with resolution of one period of phase voltage.

FU2-FU0 - filtration of short action interferences. It is generally applied that if CurOut < 32 * FU, the interference is not proceeded and a content CurOut is set to zero.
0 – all action interferences allowed
1 - action interferences lower than 32 suppressed (i.e. 0,32%)
2 - action interferences lower than 64 suppressed (i.e. 0,64%)
3 - action interferences lower than 96 suppressed (i.e. 0,96%)
4 - action interferences lower than 128 suppressed (i.e. 1,28%)
5 - action interferences lower than160 suppressed (i.e. 1,6%)
6 - action interferences lower than 192 suppressed (i.e. 1,92%)
7 - action interferences lower than 224 suppressed (i.e. 2,24%)

Status - is used particularly for bit value transmision at on/off control as far as the action interference is dealt as time proportional control (pulse width). Further it contains error bit measuring.

.7	.6	.5	.4	.3	.2	.1	.0
-	EY3	EY2	EY1	DR	U-	UC	UH

.0 (UH)	- output for positive action interference i.e. heating
.1 (UC)	- output for negative action interference i.e.cooling
.2 (U-)	- action interference indication 0 - positive action interference 1 - negative action interference
.3 (DR)	- detection of linear interpolation course of required value 1 – interpolation is active
.4 (EY1)	- detection of measurement error y1(*Input1*) 1 - y1 outside subrange <MinY, MaxY>
.5 (EY2)	- detection of measurement error y1(*Input2*) 1 - y2 outside subrange <MinY, MaxY>
.6 (EY3)	- detection of measurement error y1(*Input3*) 1 - y3 outside subrange <MinY, MaxY>

AuxD - auxiliary controller variables. Entry and saving in this zone is prohibited!!!

2.3.2. PID1 - controller with incremental control

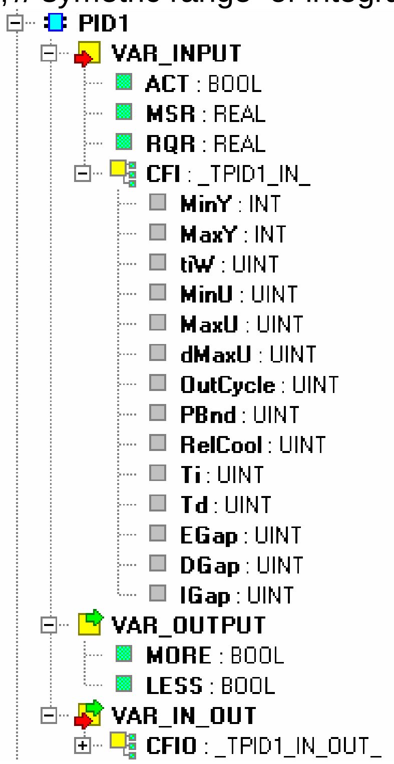
If the variable *ACT* is in log.1, other input variables of function block are accepted and PID algorithm of the controller is activated – regulation of measured value *MSR* to required value *RQR*. In variables *MORE* and *LESS* are transmitted statements for action member (regulation valve).

Function block sets these variables of general PID structure:

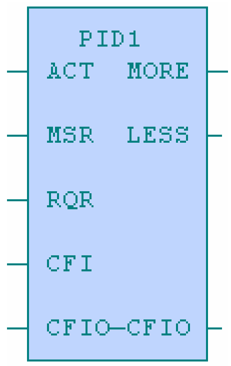
```
MSR      = Input1
RQR      = gW
MORE     = Status.0
LESS     = Status.1
CONTROL  = $0071 (at start of regulation only)
```

If there is no user values assigned to the input structure of function block *CFI,* the function block will set following default values to the structure during cold restart:

```
CFI :=
    (MinY      := 0,       // value of unified range corresponding to 0%
    MaxY       := 1000,    // value of unified range corresponding to 100%
    tiW        := 0,       // time constant of required value filter
    MinU       := 0,       // minimum granted action interference
    MaxU       := 10000,   // maximum granted action interference
    dMaxU      := 1000,    // maximum action quantity gain within one period
    OutCycle:= 820,        // period of sampling of regulation algorithm [10ms]
    PBnd       := 500,     // proportionality range
    RelCool    := 1000,    // relative proportionality range for negative regulation deviations
    Ti         := 740,     // integrative time constant
    Td         := 26,      // derivative time constant
    EGap       := 10,      // symetric range of insensibility
    DGap       := 10000,   // symetric range  of derivation unit action
    IGap       := 10000);  // symetric range  of integration unit action
```



Obr. 2.16 The structure of FB PID1



Obr. 2.17 The appearance of FB PID1

Variables description :

Term		**Signification**	**Type**	**Format**
ACT		Activation	input	bool
MSR		Measured value		real
RQR		Required value		real
CFI		Input control structure		_TPID1_IN_
	.MinY	value of unified range for 0%		int
	.MaxY	value of unified range for 100%		int
	.tiW	time constant of required value filter		uint
	.MinU	minimum granted action interference		uint
	.MaxU	maximum granted action interference		uint
	.dMAxU	maximum action quantity gain within one period		uint
	.OutCycle	Period of sampling of regulation algorithm [10ms]		uint
	.PBnd	proportionality range		uint
	.RelCool	relative proportionality range for negative regulation deviations		uint
	.Ti	integrative time constant		uint
	.Td	derivative time constant		uint
	.Egap	symetric range of insensibility		uint
	.DGap	symetric range of derivation unit action		uint
	.IGap	symetric range of integration unit action		uint
MORE		Positive action interference	output	bool
LESS		Negative action interference		bool
CFIO		Input/output control structure	Input/output	_TPID1_IN_OUT_
	.Control	Control word of controller (see c. 2.3.1.)		_TPID_Control_

2.3.3. PID11 - controller with incremental control and shortlist of variables

If the variable *ACT* is in log.1, other input variables of function block are accepted and PID algorithm of the controller is activated – regulation of measured value *MSR* to required value *RQR*. In variables *MORE* and *LESS* are transmitted statements for action member (regulation valve).

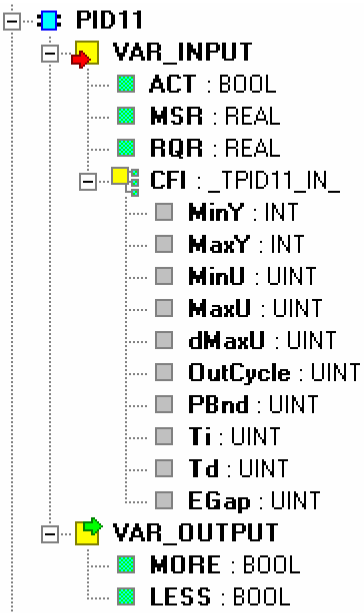
So the function block works similarly to PID1, the only diference is in shortlist of adjustable variables. Variables tiW, RelCool, DGap, IGap are set to fixed values.

The function block sets these variables of general PID structure:

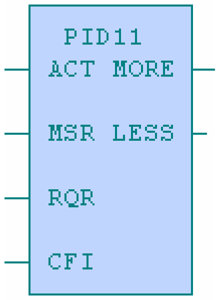
```
MSR       = Input1
RQR       = gW
MORE      = Status.0
LESS      = Status.1
CONTROL   = $0071 (at start of regulation only)
tiW       = 0
RelCool   = 1000
DGap      = 10000
IGap      = 10000
```

If there is no user values assigned to the input structure of function block *CFI,* the function block will set following default values to the structure during cold restart:

```
CFI :=
      (MinY     := 0,       // value of unified range corresponding 0%
      MaxY      := 1000     // value of unified range corresponding 100%
      MinU      := 0,       // minimum granted action interference
      MaxU      := 10000,   // maximum granted action interference
      dMaxU     := 1000,    // maximum action quantity gain within one period
      OutCycle:= 1000,      // period of sampling of regulation algorithm [10ms]
      PBnd      := 1000,     // proportionality range
      RelCool   := 1000,    // relative proportionality range for negative regulation deviations
      Ti        := 1000,    // integrative time constant
      Td        := 0,       // derivative time constant
      EGap      := 10)      // symetric range of insensibility
```



Obr. 2.18 The structure of FB PID11



Obr. 2.19 The appearance of FB PID11

Variables description :

Term		Signification	Type	Format
ACT		Activation	Input	bool
MSR		Measured value		real
RQR		Required value		real
CFI		Input control structure		_TPID11_IN_
	.MinY	value of unified range for 0%		int
	.MaxY	value of unified range for 100%		int
	.MinU	minimum granted action interference		uint
	.MaxU	maximum granted action interference		uint
	.dMAxU	maximum action quantity gain within one period		uint
	.OutCycle	Period of sampling of regulation algorithm [10ms]		uint
	.PBnd	proportionality range		uint
	.Ti	integrative time constant		uint
	.Td	derivative time constant		uint
	.Egap	symetric range of insensibility		uint
MORE		Positive action interference	Output	bool
LESS		Negative action interference		bool

2.3.4. PID2 – controller with direct control

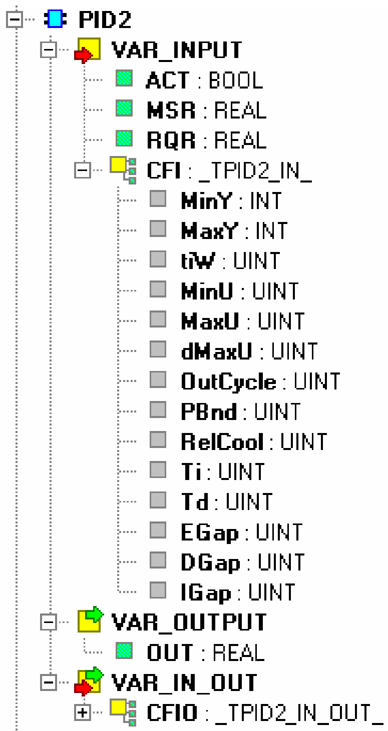
If the variable *ACT* is in log.1, other input variables of function block are accepted and PID algorithm of the controller is activated – regulation of measured value *MSR* to required value *RQR*. There is, in *OUT* variable, transmitted the required action interference for action member (regulation valve) in 0-100% range that corresponds to PCT format of analog output cards.

The function block sets these variables of general PID structure:

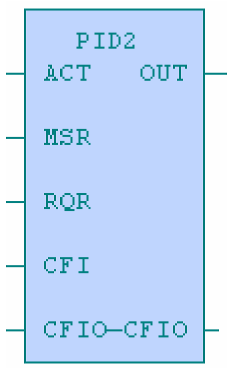
MSR = Input1
RQR = gW
OUT = Output
CONTROL = $0041 (at start of regulation only)

If there is no user values assigned to the input structure of function block *CFI,* the function block will set following default values to the structure during cold restart:

```
CFI :=
    (MinY     := 0,      // value of unified range corresponding 0%
    MaxY      := 1000,   // value of unified range corresponding 100%
    tiW       := 0,      // time constant of required value filter
    MinU      := 0,      // minimum granted action interference
    MaxU      := 10000,  // maximum granted action interference
    dMaxU     := 1000,   // maximum action quantity gain within one period
    OutCycle  := 820,    // period of sampling of regulation algorithm [10ms]
    PBnd      := 500,    // proportionality range
    RelCool   := 1000,   // relative proportionality range for negative regulation deviations
    Ti        := 740,    // integrative time constant
    Td        := 26,     // derivative time constant
    EGap      := 10,     // symetric range of insensibility
    DGap      := 10000,  // symetric range  of derivation unit action
    IGap      := 10000); // symetric range  of integration unit action
```



Obr. 2.20 The structure of FB PID2



Obr. 2.21 The appearance of FB PID2

Variables description :

Term		Signification	Type	Format
ACT		Activation	Input	bool
MSR		Measured value		real
RQR		Required value		real
CFI		Input control structure		_TPID2_IN_
	.MinY	value of unified range for 0%		int
	.MaxY	value of unified range for 100%		int
	.tiW	time constant of required value filter		uint
	.MinU	minimum granted action interference		uint
	.MaxU	maximum granted action interference		uint
	.dMAxU	maximum action quantity gain within one period		uint
	.OutCycle	Period of sampling of regulation algorithm [10ms]		uint
	.PBnd	proportionality range		uint
	.RelCool	relative proportionality range for negative regulation deviations		uint
	.Ti	integrative time constant		uint
	.Td	derivative time constant		uint
	.Egap	symetric range of insensibility		uint
	.DGap	symetric range of derivation unit action		uint
	.IGap	symetric range of integration unit action		uint
OUT		Action interference	output	real
CFIO		Input/output control structure	Input/ Output	_TPID2_IN_OUT_
	.Control	Control word of controller (see c. 2.3.1.)		_TPID_Control_

2.3.5. PID21 - controller with direct control and shortlist of variables

If the variable *ACT* is in log.1, other input variables of function block are accepted and PID algorithm of the controller is activated – regulation of measured value *MSR* to required value *RQR*. There is, in *OUT* variable, transmitted the required action interference for action member (regulation valve) in 0-100% range that corresponds to PCT format of analog output cards.

So the function block works similarly to PID2, the only diference is in shortlist of adjustable variables. Variables tiW, RelCool, DGap, IGap are set to fixed values.

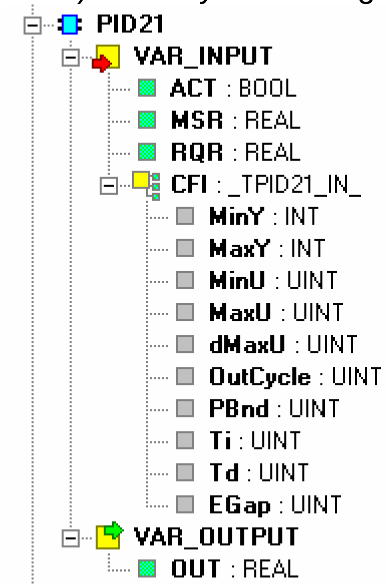
The function block sets these variables of general PID structure:

MSR	= Input1
RQR	= gW
OUT	= Output
CONTROL	= $0041 (at start of regulation only)
tiW	= 0
RelCool	= 1000
DGap	= 10000
IGap	= 10000

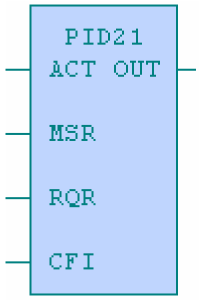
If there is no user values assigned to the input structure of function block *CFI,* the function block will set following default values to the structure during cold restart:

CFI :=

```
(MinY     := 0,      // value of unified range corresponding 0%
 MaxY     := 1000    // value of unified range corresponding 100%
 MinU     := 0,      // minimum granted action interference
 MaxU     := 10000,  // maximum granted action interference
 dMaxU    := 1000,   // maximum action quantity gain within one period
 OutCycle:= 1000,    // period of sampling of regulation algorithm [10ms]
 PBnd     := 1000,   // proportionality range
 RelCool  := 1000,   // relative proportionality range for negative regulation deviations
 Ti       := 1000,   // integrative time constant
 Td       := 0,      // derivative time constant
 EGap     := 10)     // symetric range of insensibility
```



Obr. 2.22 The structure of FB PID21



Obr. 2.23 The appearance of FB PID21

Variables description :

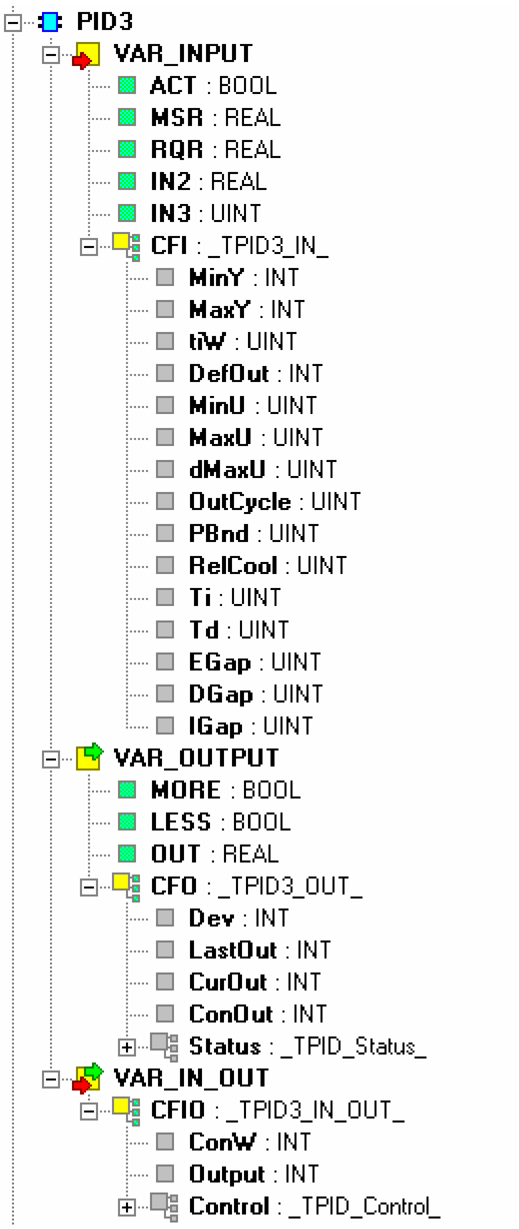
Term		Signification	Type	Format
ACT		activation		bool
MSR		Measured value		real
RQR		Required value		real
CFI		Input control structure		_TPID21_IN_
	.MinY	value of unified range for 0%		int
	.MaxY	value of unified range for 100%		int
	.MinU	minimum granted action interference	Input	uint
	.MaxU	maximum granted action interference		uint
	.dMAxU	maximum action quantity gain within one period		uint
	.OutCycle	Period of sampling of regulation algorithm [10ms]		uint
	.PBnd	proportionality range		uint
	.Ti	integrative time constant		uint
	.Td	derivative time constant		uint
	.Egap	symetric range of insensibility		uint
OUT		Action interference	Output	real

2.3.6. PID3 - freely adjustable controller

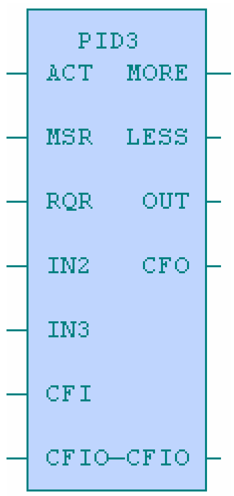
This function block allows the access to all variables of general data structure of PID regulation (see c. 2. 3. 1.) and so enables the user to use freely all PID regulation resources.

The function block sets these variables of general PID structure:

MSR	= Input1
RQR	= gW
MORE	= Status.0
LESS	= Status.1
OUT	= Output



Obr. 2.24 The structure of FB PID3



Obr. 2.25 The appearance of FB PID3

Variables description :

Term		**Signification**	**Type**	**Format**
ACT		Activation	Input	bool
MSR		Measured value		real
RQR		Required value		real
IN2		Ratio control		real
IN3		Servo-operated valve possition		uint
CFI		Input variables of PID structure		_TPID3_IN_
	.MinY	value of unified range for 0%		int
	.MaxY	value of unified range for 100%		int
	.tiW	time constant of required value filter		uint
	.DefOut	Pre-defined action interference at failure		int
	.MinU	minimum granted action interference		uint
	.MaxU	maximum granted action interference		uint
	.dMAxU	maximum action quantity gain within one period		uint
	.OutCycle	Period of sampling of regulation algorithm [10ms]		uint
	.PBnd	Proportionality range		uint
	.RelCool	relative proportionality range for negative regulation deviations		uint
	.Ti	Integrative time constant		uint
	.Td	Derivative time constant		uint
	.Egap	Symetric range of insensibility		uint
	.DGap	symetric range of derivation unit action		uint
	.IGap	symetric range of integration unit action		uint
MORE		Positive action interference	Output	bool
LESS		Negative action interference		bool
OUT		Action interference		real
CFO		Input variables of PID structure		_TPID3_OUT_
	.Dev	Control deviation		int
	.LastOut	Last action interference		int
	.CurOut	Actual required action interference		int
	.ConOut	Realized action interference		int
	.Status	Controller status word (see c. 2.3.1.)		_TPID_Status_
CFIO		Input/output variables of PID structure	input/ output	_TPID3_IN_OUT_
	.ConW	Actual required value		int
	.Output	Action interference		int
	.Control	Control word of controller (see c. 2.3.1.)		_TPID_Control_

2.4. CASCADING

Cascading function blocks work according to undermentioned characterization. Individual function blocks of cascading differ only by a number of switching cascade degrees (from 2 to 5).

If the variable *ACT* is in log.1, then on the basis of input variable *IN* single cascade degrees *STx* are controlled. Degrees are switched and isolated according to individual limits *LIMITx* (relating to input variable IN) and according to a set hysteresis *HYSTER.* In case of failure of one of the degrees *ERRx* is this degree switched off and replaced by another (the other one in the cascade)

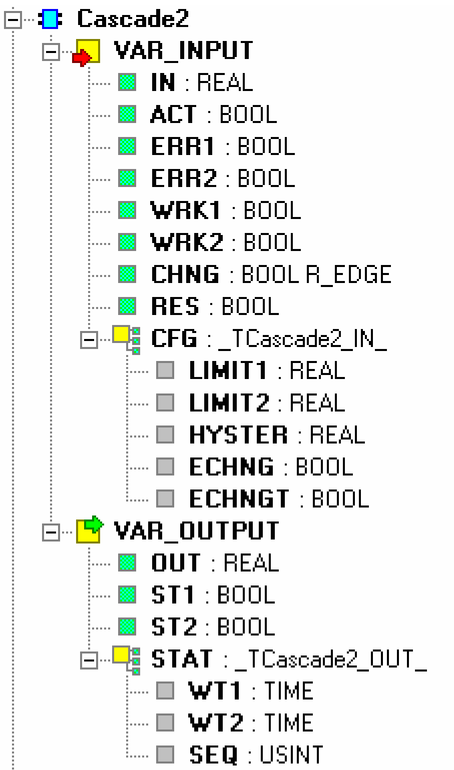
The component also contains a function for variation in switching the degrees. If the variable *ECHNG* is in log.1, the sequence of degrees is changed at entering edge of binary signal *CHNG.*

Individual degrees are lined so, that their use is even (see the sequence table). In case that the variable *ECHNGT* is also in log.1, then degrees of the cascade are lined according to running hours *WTx* in such a way that the one with the lowest number of hours would be on the first place in the cascade. Counting of cascade degree´s running hours is activated by the setting of particular variable *WRKx* into log.1 in a period where the run degree is given.

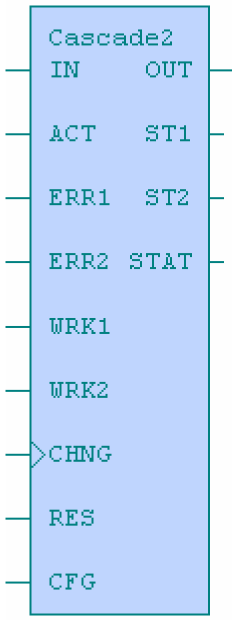
By the *RES* signal are all running hours of all cascade degrees resetted.

The input *IN* is connected to joint input of regulation PID function blocks and can take the value in range from 0 to 100 [%]. The value of input *IN* is coppied to output *OUT* for the purpose of further cascading.

2.4.1. Cascade2 - cascade of 2 degrees with variation



Obr. 2.26 The structure of FB Cascade2



Obr. 2.27 The appearance of FB Cascade2

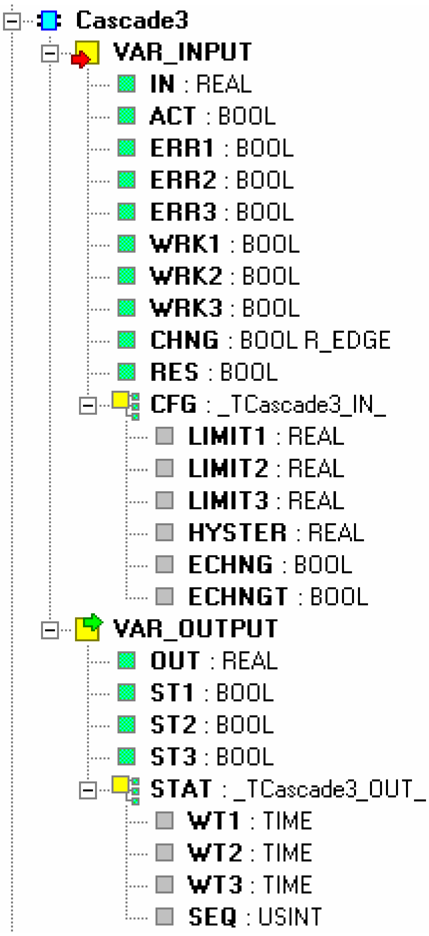
Variables description :

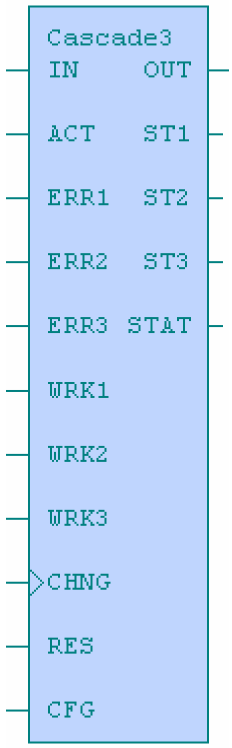
Term		**Signification**	**Type**	**Format**
IN		Input power required	Input	real
ACT		Action activation		bool
ERRx		Failure x. cascade degrees		bool
WRKx		run x. cascade degrees		bool
CHNG		Variation in the sequence of cascade degrees		bool r_edge
RES		Running hours reset		bool
CFG		Control structure		_Tcascade2_IN_
	.LIMITx	On/off limit x. cascade degrees		real
	.HYSTER	On/off hysteresis		real
	.ECHNG	Permission for variation of cascade degrees		bool
	.EGNGT	Permission for variation of cascade degrees according to running hours		bool
OUT		Output power required	Output	real
STx		x. cascade output		bool
STAT		Status structure		_Tcascade2_OUT_
	.WTx	Running hours x. cascade degrees		time
	.SEQ	Sequence of cascade degrees		usint

Sequence table :

SEQ	**Order**
0	12
1	21

2.4.2. Cascade3 - cascade of 3 degrees with variation





Obr. 2.28 The structure of FB Cascade3

Obr. 2.29 The appearance of FB Cascade3

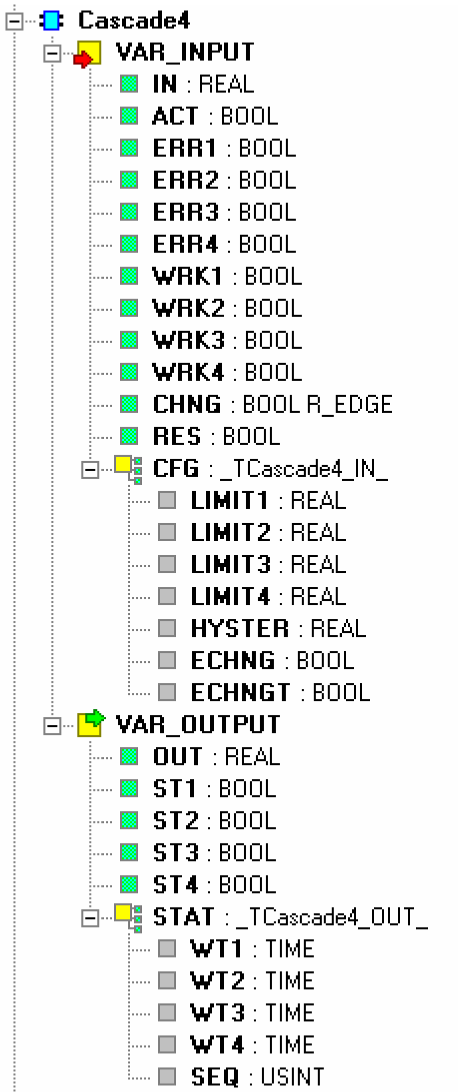
Variables description :

Term	Signification	Type	Format
IN	Input power required		real
ACT	Action activation		bool
ERRx	Failure x. cascade degrees		bool
WRKx	run x. cascade degrees		bool
CHNG	Variation in the sequence of cascade degrees		bool r_edge
RES	Running hours reset	Input	bool
CFG	Control structure		_Tcascade3_IN_
.LIMITx	On/off limit x. cascade degrees		real
.HYSTER	On/off hysteresis		real
.ECHNG	Permission for variation of cascade degrees		bool
.EGNGT	Permission for variation of cascade degrees according to running hours		bool
OUT	Output power required		real
STx	x. cascade output		bool
STAT	Status structure	Output	_Tcascade3_OUT_
.WTx	Running hours x. cascade degrees		time
.SEQ	Sequence of cascade degrees		usint

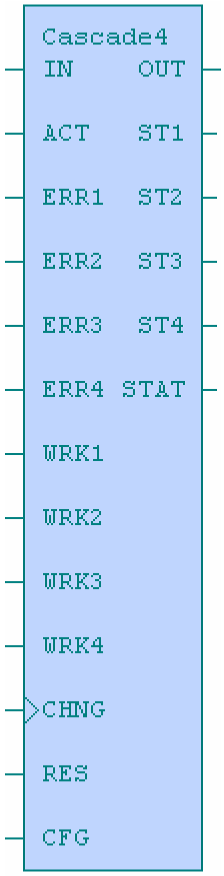
Sequence table :

SEQ	Order
0	123
1	231
2	312

2.4.3. Cascade4 - cascade of 4 degrees with variation



Obr. 2.30 The structure of FB Cascade4



Obr. 2.31 The appearance of FB Cascade4

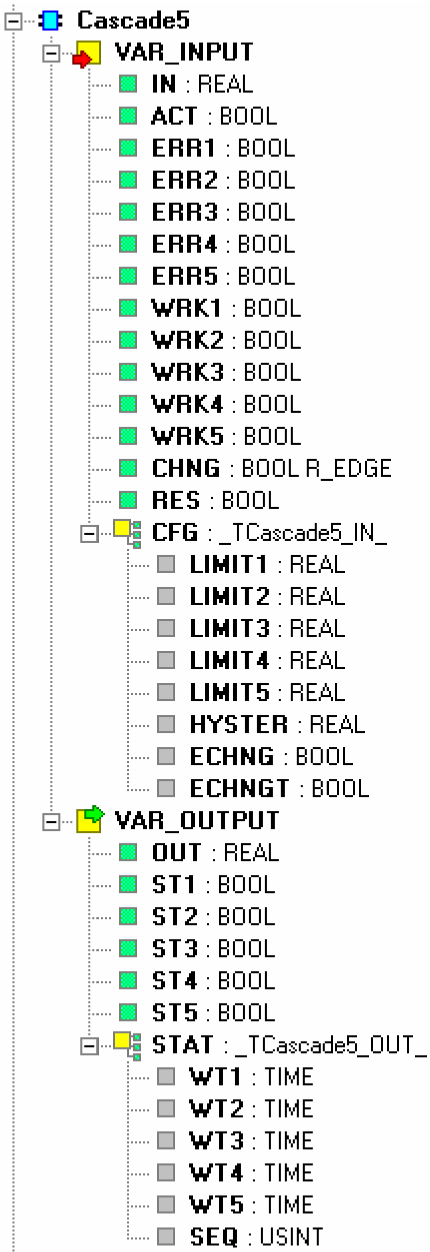
Variables description :

Term		Signification	Type	Format
IN		Input power required	Input	real
ACT		Action activation		bool
ERRx		Failure x. cascade degrees		bool
WRKx		run x. cascade degrees		bool
CHNG		Variation in the sequence of cascade degrees		bool r_edge
RES		Running hours reset		bool
CFG		Control structure		_Tcascade4_IN_
	.LIMITx	On/off limit x. cascade degrees		real
	.HYSTER	On/off hysteresis		real
	.ECHNG	Permission for variation of cascade degrees		bool
	.EGNGT	Permission for variation of cascade degrees according to running hours		bool
OUT		Output power required	Output	real
STx		x. cascade output		bool
STAT		Status structure		_Tcascade4_OUT_
	.WTx	Running hours x. cascade degrees		time
	.SEQ	Sequence of cascade degrees		usint

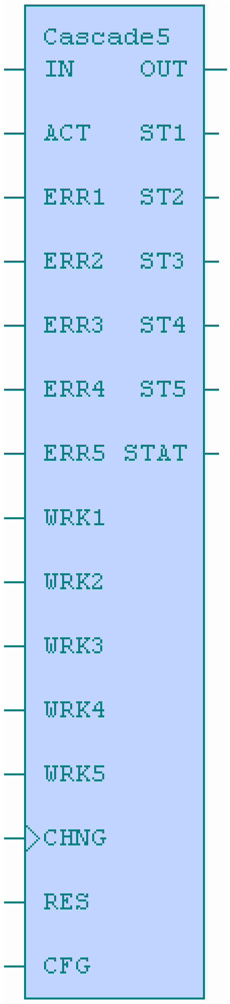
Sequence table :

SEQ	Order
0	1234
1	2341
2	3412
3	4123

2.4.4. Cascade5 - cascade of 5 degrees with variation



Obr. 2.32 *The structure of FB Cascade5*



Obr. 2.33 The appearance of FB Cascade5

Variables description :

Term		Signification	Type	Format
IN		Input power required	Input	real
ACT		Action activation		bool
ERRx		Failure x. cascade degrees		bool
WRKx		run x. cascade degrees		bool
CHNG		Variation in the sequence of cascade degrees		bool r_edge
RES		Running hours reset		bool
CFG		Control structure		_Tcascade5_IN_
	.LIMITx	On/off limit x. cascade degrees		real
	.HYSTER	On/off hysteresis		real
	.ECHNG	Permission for variation of cascade degrees		bool
	.EGNGT	Permission for variation of cascade degrees according to running hours		bool
OUT		Output power required	Output	real
STx		x. cascade output		bool
STAT		Status structure		_Tcascade5_OUT_
	.WTx	Running hours x. cascade degrees		time
	.SEQ	Sequence of cascade degrees		usint

Sequence table :

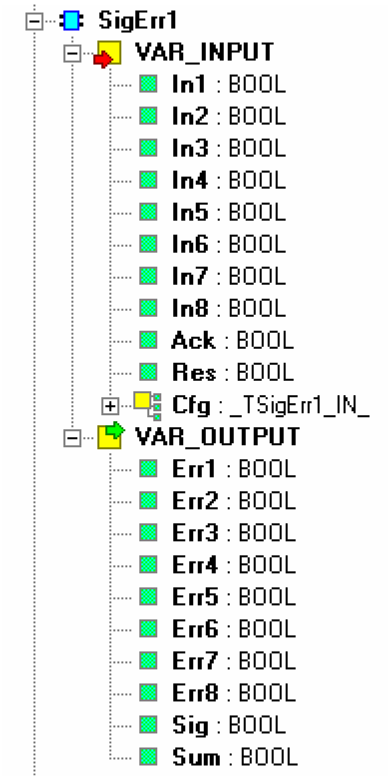
SEQ	Order
0	12345
1	23451
2	34512
3	45123
4	51234

2.5. ERROR INDICATION

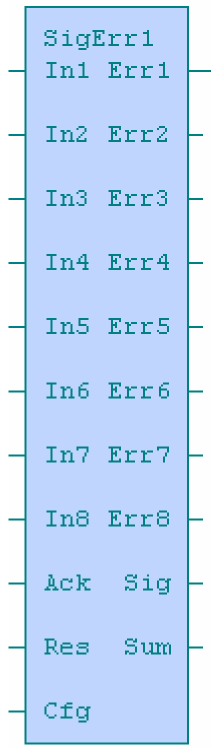
2.5.1. SigErr1 – binary error indication

The function block executes an evaluation of failure occurance with set time-lag *PRESETTIMEx* for 8 binary inputs. If the input signal *INx* is active for more than the set preselection, the output signal *ERRx* is set to log.1. Furthermore, the function block executes a logical sum of all evaluated failures into a variable *SUM* and accomplish an indication of new evaluated failure *SIG*.

The failure occurance can be confirmed by *ACK* signal and non-active failures can be resetted by *RES* signal. Every new evaluated failure will invoke blinking of optical indication output *SIG* at intervals of 1sec. If there is after confirmation (at input ACK log.1) the variable *SUM* in log.1, the optical indication *SIG* is in log.1. Conversely it is in log.0.



Obr. 2.34 The structure of FB SigErr1



Obr. 2.35 The appearance of FB SigErr1

Variables description :

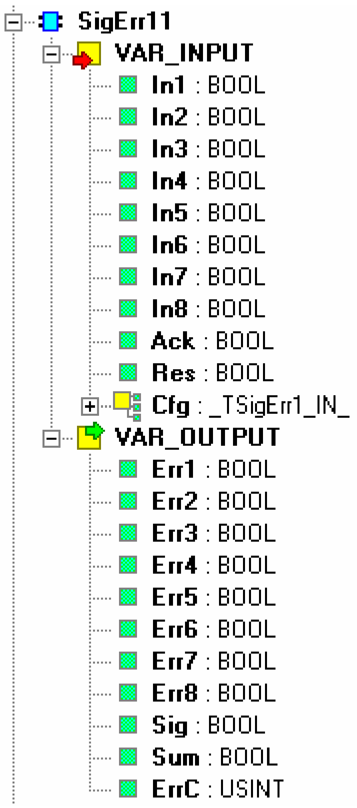
Term	Signification	Type	Format
InX	Input failure indication X		bool
Ack	Failure confirmation		bool
Res	Failure reset	Input	bool
Cfg	Configuration block structure		_TSigErr1_IN_
.PresetTimeX	Failure evaluation delay X		time
ErrX	Output failure indication X		bool
Sig	Indicator	Output	bool
Sum	Joint failure		bool

2.5.2. SigErr11 - binary error indification with fault No. indication

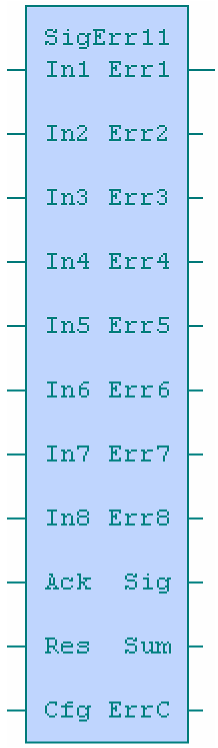
The function block executes an evaluation of failure occurance with set time-lag *PRESETTIMEx* for 8 binary inputs. If the input signal *INx* is active for more than the set preselection, the output signal *ERRx* is set to log.1. Furthermore, the function block executes a logical sum of all evaluated failures into a variable *SUM* and accomplish an indication of new evaluated failure *SIG*.

The failure occurance can be confirmed by *ACK* signal and non-active failures can be resetted by *RES* signal. Every new evaluated failure will invoke blinking of optical indication output *SIG* at intervals of 1sec. If there is after confirmation (at input ACK log.1) the variable *SUM* in log.1, the optical indication *SIG* is in log.1. Conversely it is in log.0.

The function block further contains an output variable with the number of last active failure *ERRC* that is intended for connection onto the function block of failure history (History1,History5,History10).



Obr. 2.36 The structure of FB SigErr11



Obr. 2.37 The appearance of FB SigErr11

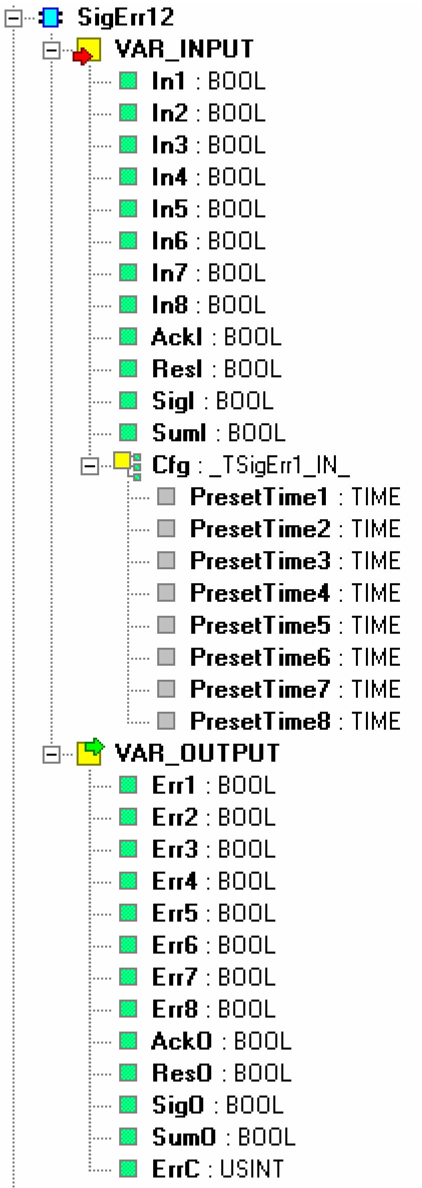
Variables description :

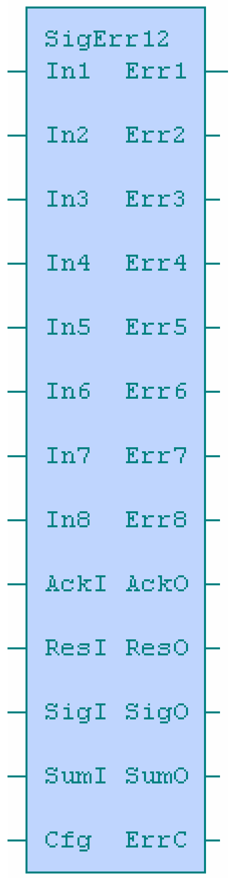
Term	Signification	Type	Format
InX	Input failure indication X		bool
Ack	Failure confirmation		bool
Res	Failure reset	Input	bool
Cfg	Configuration block structure		_TSigErr1_IN_
.PresetTimeX	Failure evaluation delay X		time
ErrX	Output failure indication X		bool
Sig	Indicator	Output	bool
Sum	Joint failure		bool
ErrC	Failure number		usint

2.5.3. SigErr12 - binary error indication with connection attendance

The function block executes an evaluation of failure occurance with set time-lag *PRESETTIMEx* for 8 binary inputs. If the input signal *INx* is active for more than the set preselection, the output signal *ERRx* is set to log.1. Furthermore, the function block executes a logical sum of all evaluated failures into a variable *SUMO* and accomplish an indication of new evaluated failure *SIGO.*

The failure occurance can be confirmed by *ACKI* signal and non-active failures can be resetted by *RESI* signal. Variables *ACKO, RESO, SIGO, SUMO* are used for cascading of more components. The function block further contains an output variable with the number of last active failure *ERRC* that is intended for connection onto the function block of failure history (History1,History5,History10). Every new evaluated failure will invoke blinking of optical indication output *SIGO* at intervals of 1sec. If there is after confirmation (at input ACKI log.1) the variable *SUMO* in log.1, the optical indication *SIGO* is in log.1. Conversely it is in log.0.





Obr. 2.38 The structure of FB SigErr12

Obr. 2.39 The appearance of FB SigErr12

Variables description :

Term	Signification	Type	Format
InX	Input failure indication X	Input	bool
AckI	Failure confirmation		bool
ResI	Failure reset		bool
SigI	Indicator		bool
SumI	Joint failure		bool
Cfg	Configuration block structure		_TSigErr1_IN_
.PresetTimeX	Failure evaluation delay X		time
ErrX	Output failure indicator X	Output	bool
AckO	Failure confirmation		bool
ResO	Failure reset		bool
SigO	indicator		bool
SumO	Joint failure		bool
ErrC	Failure number		usint

2.5.4. SigErr13 - binary error indication with resetting option

The function block executes an evaluation of failure occurance with set time-lag *PRESETTIMEx* for 8 binary inputs. If the input signal *INx* is active for more than the set preselection, the output signal *ERRx* is set to log.1. Furthermore, the function block executes a logical sum of all evaluated failures into a variable *SUMO,* accomplish an indication of new evaluated failure *SIGO* and acoustic indication *AKUO*.

The failure occurance can be confirmed by *ACKI* signal and resetted by *RESI* signal. Variables *CASI, CASO* are used for cascading of more components. The function block further contains an output variable with the number of last active failure *ERRC* that is intended for connection onto the function block of failure history (History1,History5,History10). The *ERRC* code is compared to other components SigErr set for one cycle only at new failure occurance. Alongside every failure it is possible to set the form of resetting of particular failure within variable CONTROLx. The variable CONTROLx determine the form of optical and acoustic indication.

CONTROLx = 0 - if the input signal *INx is not active, the output variable ERRx* is resetted (self-reverse failure)

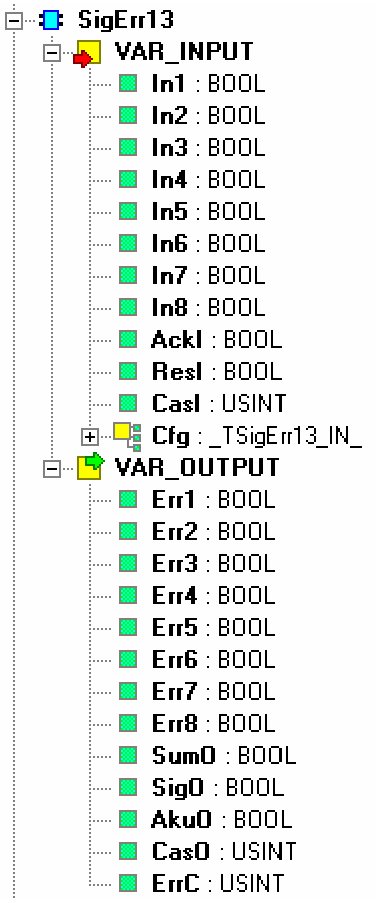
CONTROLx = 1 - the output variable *ERRx* is resetted by *RESI* signal regardless of a value of input signal

CONTROLx = 2 - if the input signal *INx is not active, the output variable ERRx* is resetted by *RESI* signal

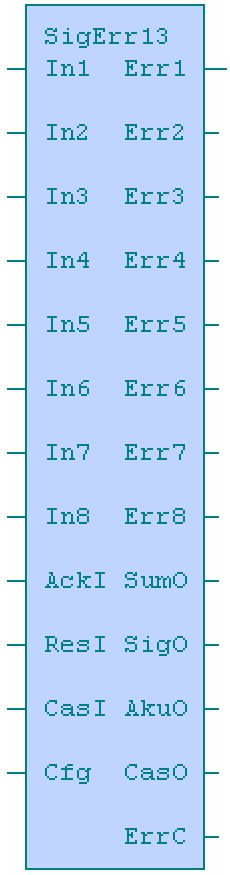
Every new evaluated failure will invoke blinking of optical indication output *SIGO* at intervals of 1sec and set to log.1 the output for acoustic indication *AKUO*. After confirmation (at input ACKI log.1) is the acoustic indication resetted. If the variable *SUMO* is in log.1, the optical indication *SIGO* is also after confirmation in log.1. Conversely it is in log.0.

The variable *CONTROLS* determines the form of indication if the evaluated failures are resetted without a previous confirmation. (for self-reverse failures).

CONTROLS = 0 - optical and acoustic indication is in log.0
CONTROLS = 1 - acoustic indication is in log.0, optical indication blinks at intervals of 1sec.
CONTROLS = 2 - acoustic indication is in log.1, optical indication blinks at intervals of 1sec.



Obr. 2.40 The structure of FB SigErr13



Obr. 2.41 The appearance of FB SigErr13

Variables description :

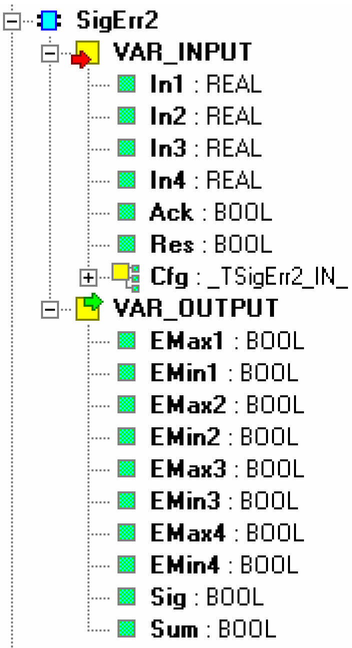
Term		Signification	Type	Format
InX		Input failure indication X		bool
AckI		Failure confirmation		bool
ResI		Failure reset		bool
CasI		Cascade input		usint
Cfg		Configuration block structure	Input	_TSigErr1_IN_
	.PresetTimeX	Failure evaluation delay X		time
	.controlX	Control word for failure resetting form X		usint
	.controls	Control word for non-confirmed failures indication form		usint
ErrX		Output failure indicator X		bool
SumO		Joint failure		bool
SigO		Indicator	Output	bool
AkuO		horn		bool
CasO		Cascade output		usint
ErrC		Failure number		usint

2.5.5. SigErr2 - analog error indication

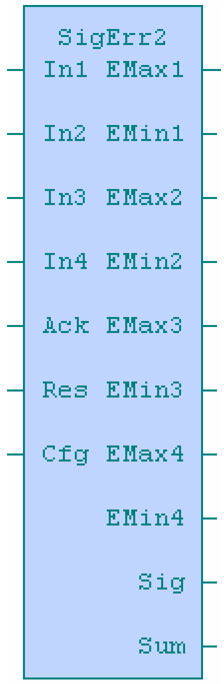
The function block executes the control of four input analog values. If a measured value *INx* is higher than the set maximum *PRESETMAXx* for a period longer than the set preselection *PRESETTIMEx*, the output signal *EMAXx* is set to log.1. In case that the measured value *INx* is lower than the set minimum *PRESETMINx* for a period longer than the set preselection *PRESETTIMEx*, then into log.1 is set the signal *EMINx.*

Furthermore, the function block executes a logical sum of all evaluated failures into the variable *SUM* and an indication of newly evaluated failure *SIG.* The occurance of failures is possible to confirm by *ACK* signal and non-active failures can be resetted by *RES* signal.

Every new evaluated failure will invoke blinking of optical output indication *SIG* at intervals of 1sec. If the variable *SUM* is after confirmation (at input ACK log.1) in log.1, the optical indication *SIG* is in log.1. Conversely it is in log.0.



Obr. 2.42 The structure of FB SigErr2



Obr. 2.43 The appearance of FB SigErr2

Variables description :

Term		Signification	Type	Format
InX		Input analog value X		real
Ack		Failure confirmation		bool
Res		Failure resetting		bool
Cfg		Configuration block structure	Input	_TSigErr2_IN_
	.PresetTimeX	Failure evaluation delay X		time
	.PresetMaxX	Limit for failure evaluation of maximum X		real
	.PresetMinX	Limit for failure evaluation of minimum X		real
EMaxX		Excess of input maximum X		bool
EMinX		Input minimum not reached X	Output	bool
Sig		indicator		bool
Sum		Joint failure		bool

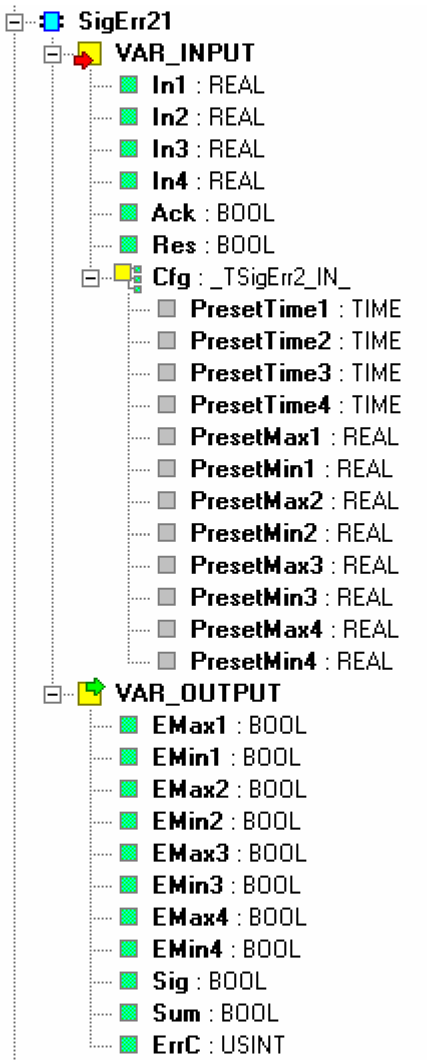
2.5.6. SigErr21 - analog error indication with error No. indication

The function block executes the control of four input analog values. If a measured value *INx* is higher than the set maximum *PRESETMAXx* for a period longer than the set preselection *PRESETTIMEx*, the output signal *EMAXx* is set to log.1. In case that the measured value *INx* is lower than the set minimum *PRESETMINx* for a period longer than the set preselection *PRESETTIMEx*, then into log.1 is set the signal *EMINx.*

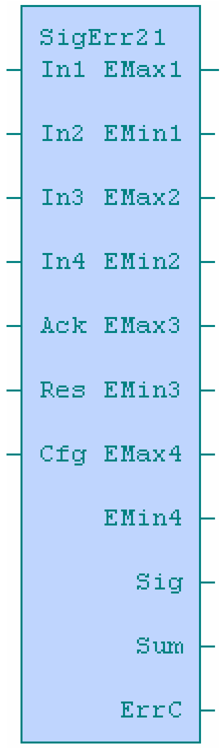
Furthermore, the function block executes a logical sum of all evaluated failures into the variable *SUM* and an indication of newly evaluated failure *SIG.* The occurance of failures is possible to confirm by *ACK* signal and non-active failures can be resetted by *RES* signal.

Every new evaluated failure will invoke blinking of optical output indication *SIG* at intervals of 1sec. If the variable *SUM* is after confirmation (at input ACK log.1) in log.1, the optical indication *SIG* is in log.1. Conversely it is in log.0.

The function block further contains an output variable with the number of last active failure *ERRC* that is intended for connection onto the function block of failure history (History1,History5,History10)



Obr. 2.44 The structure of FB SigErr21



Obr. 2.45 The appearance of FB SigErr21

Variables description :

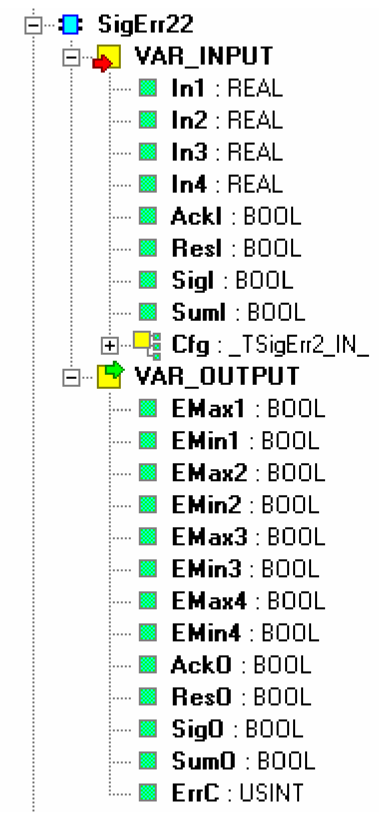
Term		Signification	Type	Format
InX		Input analog value X	Input	real
Ack		Failure confirmation		bool
Res		Failure resetting		bool
Cfg		Configuration block structure		_TSigErr2_IN_
	.PresetTimeX	Failure evaluation delay X		time
	.PresetMaxX	Limit for failure evaluation of maximum X		real
	.PresetMinX	Limit for failure evaluation of minimum X		real
EMaxX		Excess of input maximum X	Output	bool
EMinX		Input minimum not reached X		bool
Sig		indicator		bool
Sum		Joint failure		bool
ErrC		Error No.		usint

2.5.7. SigErr22 - analog error indication with connection attendance

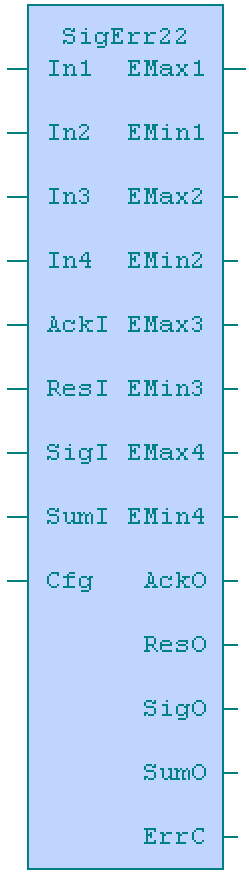
The function block executes the control of four input analog values. If a measured value *INx* is higher than the set maximum *PRESETMAXx* for a period longer than the set preselection *PRESETTIMEx*, the output signal *EMAXx* is set in log.1. In case that the measured value *INx* is lower than the set minimum *PRESETMINx* for a period longer than the set preselection *PRESETTIMEx*, then into log.1 is set the signal *EMINx.*

Furthermore, the function block executes a logical sum of all evaluated failures into the variable *SUMO* and an indication of newly evaluated failure *SIGO.* The occurance of failures is possible to confirm by *ACKI* signal and non-active failures can be resetted by *RESI* signal. Tha variables *ACKO, RESO, SIGO, SUMO* are used for cascading of more components.

The function block further contains an output variable with the number of last active failure *ERRC* that is intended for connection onto the function block of failure history (History1,History5,History10). Every newly evaluated failure will invoke blinking of optical output indication *SIGO* at intervals of 1sec. If the variable *SUMO* is after confirmation (at input ACKI log.1) in log.1, the optical indication *SIGO* is in log.1. Conversely it is in log.0.



Obr. 2.46 The structure of FB SigErr22



Obr. 2.47 The appearance of FB SigErr22

Variables desctiption :

Term	**Signification**	**Type**	**Format**
InX	Input analog value X	Input	real
AckI	Failure confirmation		bool
ResI	Failure resetting		bool
SigI	Indicator		bool
SumI	Joint failure		bool
Cfg	Configuration block structure		_TSigErr2_IN_
.PresetTimeX	Failure evaluation delay X		time
.PresetMaxX	Limit for failure evaluation of maximum X		real
.PresetMinX	Limit for failure evaluation of minimum X		real
EMaxX	Excess of input maximum X	Output	bool
EMinX	Input minimum not reached X		bool
AckO	Failure confirmation		bool
ResO	Failure resetting		bool
SigO	Indication		bool
SumO	Joint failure		bool
ErrC	Error No.		usint

2.5.8. SigErr23 - analog indication with resetting option

The function block executes the control of four input analog values. If a measured value *INx* is higher than the set maximum *PRESETMAXx* for a period longer than the set preselection *PRESETTIMEx*, the output signal *EMAXx* is set in log.1. In case that the measured value *INx* is lower than the set minimum *PRESETMINx* for a period longer than the set preselection *PRESETTIMEx*, then into log.1 is set the signal *EMINx.*

Furthermore, the function block executes a logical sum of all evaluated failures into the variable *SUMO,* an indication of newly evaluated failure by *SIGO* optical indication and an acoustic indication *AKUO.* The occurance of failures is possible to confirm by *ACKI* signal and non-active can be resetted by *RESI* signal. Tha variables *CASI, CASO* are used for cascading of more components.

The function block further contains an output variable with the number of last active failure *ERRC* that is intended for connection onto the function block of failure history (History1,History5,History10). The *ERRC* code is, on the contrary of other components SigErr, set for one cycle only within new failure occurance.

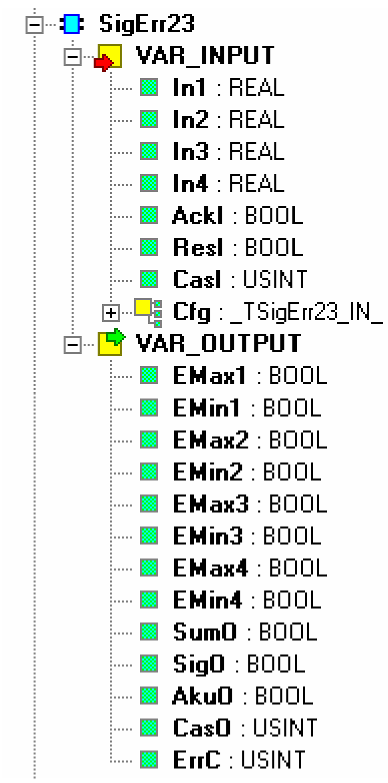
Alongside every input it is possible to set the form of resetting of particular failure within variable CONTROLx. The variable CONTROLx determine the form of optical and acoustic indication.

CONTROLx = 0 - if the input signal *INx is not active, the output variable ERRORx* is resetted (self-reverse failure)
CONTROLx = 1 - the output variable *ERRORx* is resetted by *RESI* signal regardless of a value of input signal
CONTROLx = 2 - if the input signal *INx is not active, the output variable ERRORx* is resetted by *RESIN* signal

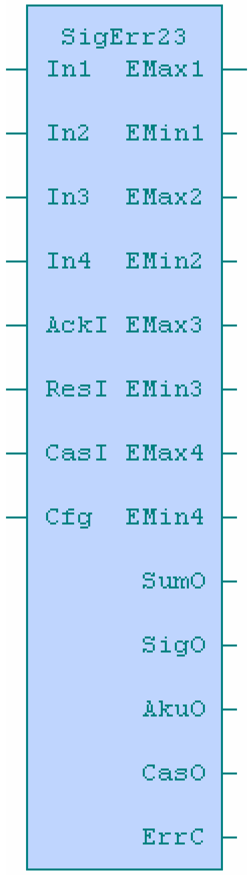
Every newly evaluated failure will invoke blinking of optical output indication *SIGO* at intervals of 1sec. and set into the log.1 the output for acoustic signalization *AKUO*. After confirmation (at input ACKI log.1) is the acoustic indication resetted. If the variable *SUMO* in log.1, the optical indication *SIGO* is after confirmation also in log.1 . Conversely it is in log.0.

The variable *CONTROLS* determines the form of indication if the evaluated failures are resetted without a previous confirmation. (for self-reverse failures).

CONTROLS = 0 - optical and acoustic indication is in log.0
CONTROLS = 1 - acoustic indication is in log.0, optical indication blinks at intervals of 1sec.
CONTROLS = 2 - acoustic indication is in log.1, optical indication blinks at intervals of 1sec.



Obr. 2.48 The structure of FB SigErr23



Obr. 2.49 The appearance of FB SigErr23

Variables determination:

Term	Signification	Type	Format
InX	Input analog value X		real
AckI	Failure confirmation		bool
ResI	Failure resetting		bool
CasI	Cascade input		usint
Cfg	Joint failure		_TSigErr23_IN_
.PresetTimeX	Failure evaluation delay X	Input	time
.PresetMaxX	Limit for failure evaluation of maximum X		real
.PresetMinX	Limit for failure evaluation of minimum X		real
.ControlX	Form of failure resetting control word X		usint
.Controls	Form of identification of unconfirmed failures control word		usint
EMaxX	Excess of input maximum X		bool
EMinX	Input minimum not reached X		bool
SumO	Joint failure		bool
SigO	Indicator	Output	bool
AkuO	Horn		bool
CasO	Cascade output		usint
ErrC	Error No.		usint

2.6. ERROR´S HISTORY

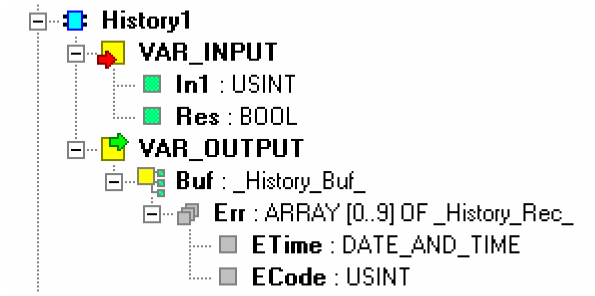
Failure history function blocks works according to undermentioned description. Individual failure history function blocks differs only by number of failure indications that are possible to be connected to the history block (1, 5 or 10).

The function block executes saving of failure occurance into a packet of FIFO type. There is, onto an input INx of history function block, inducted an output *ERRC* from failure indications (SigErr11, SigErr12, SigErr13 or SigErr21, SigErr22, SigErr23). Into and internal diabase of 10 failures BUF is always saved a newly arisen failure with a date and time of origination. The last – tenth failure, disappears owing to shift of the packet. According to the time period the newest failure´s No. is [0] and the oldest has No. [9]. Using a RES signal it is possible to erase the diabase of saved failures.

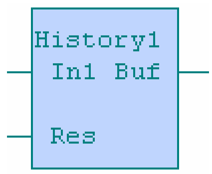
Alongside failures from components inducted onto a second input and higher, there is to the failure No. added an offset 8 and further multiples of 8 (applies to History 5 and History 10 only).

(e.g.: the number of the fifth failure from a SigPor12 component inducted onto an input IN3 of a History5 component will be 21).

2.6.1. History1 - error ´s history on one error indication



Obr. 2.50 The structure of FB History1

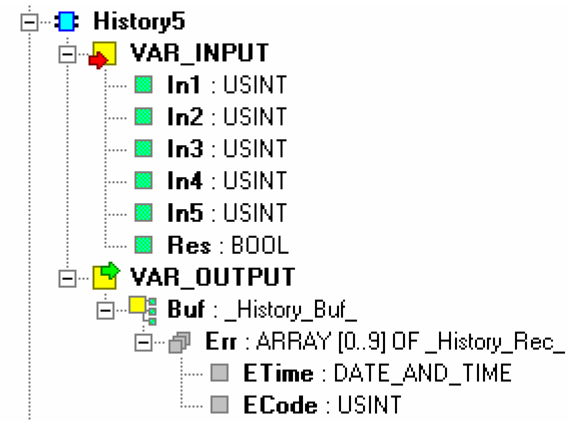


Obr. 2.51 The appearance of FB History1

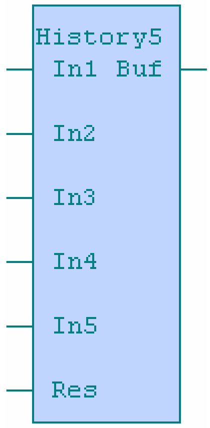
Variables description :

Term	Signification	Type	Format
In1	The actual failure No. from a failure block indication	Input	usint
Res	saved failures packet resetting		bool
Buf	Output structure	Output	_History_Buf_
.Err[X]	error buffer		array [0..9] of _History_Rec_
.ETime	Date and time of failure origination		date_and_time
.ECode	Failure code		usint

2.6.2. History5 - error ´s history on five error indications



Obr. 2.52 The structure of FB History5

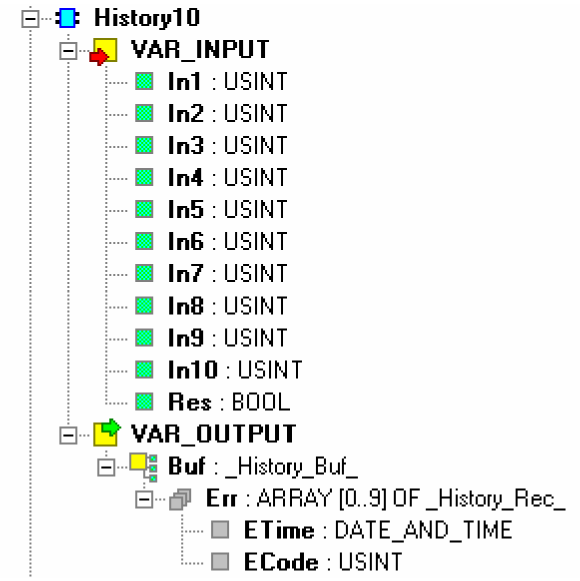


Obr. 2.53 The appearance of FB History5

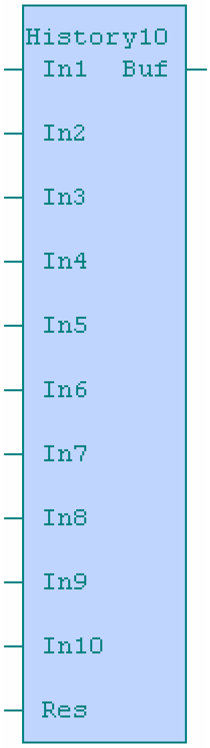
Variables description :

Term	**Signification**	**Type**	**Format**
In1	The actual failure No. from failure indication 1		usint
In2	The actual failure No. from failure indication 2		usint
In3	The actual failure No. from failure indication 3	Input	usint
In4	The actual failure No. from failure indication 4		usint
In5	The actual failure No. from failure indication 5		usint
Res	saved failures packet resetting		bool
Buf	Output structure		_History_Buf_
.Err[X]	error buffer	Output	array [0..9] of _History_Rec_
.ETime	Date and time of failure origination		date_and_time
.ECode	Failure code		usint

2.6.3. History10 - error ´s history on ten error indications



Obr. 2.54 The structure of FB History10



Obr. 2.55The appearance of FB History10

Variables description :

Term	Signification	Type	Format
In1	The actual failure No. from failure indication 1	Input	usint
In2	The actual failure No. from failure indication 2		usint
In3	The actual failure No. from failure indication 3		usint
In4	The actual failure No. from failure indication 4		usint
In5	The actual failure No. from failure indication 5		usint
In6	The actual failure No. from failure indication 6		usint
In7	The actual failure No. from failure indication 7		usint
In8	The actual failure No. from failure indication 8		usint
In9	The actual failure No. from failure indication 9		usint
In10	The actual failure No. from failure indication 10		usint
Res	saved failures packet resetting		bool
Buf	Output structure	Output	_History_Buf_
.Err[X]	error buffer		array [0..9] of _History_Rec_
.ETime	Date and time of failure origination		date_and_time
.ECode	Failure code		usint

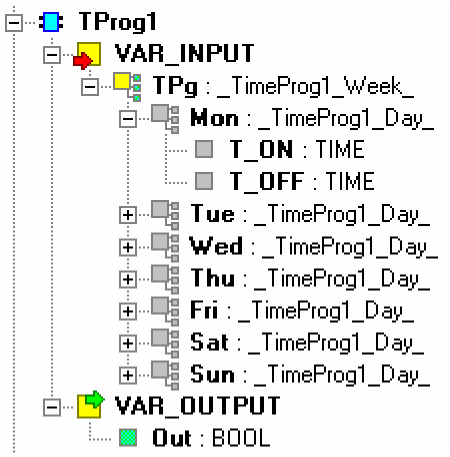
2.7. SCHEDULE PROGRAMS

2.7.1. TProg1 - weekly schedule with one ON/OFF interval a day

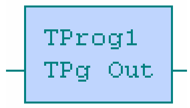
The function block sets on the basis of set weekly schedule and PLC system date the output operation signal *OUT*.

The variable *OUT* is in log.1 if the actual system date is between parameters T_ON and T_OFF for a set day in a week, otherwise, it is in log.0. Variables can be set in a range from 00:00:00 to 24:00:00 where value 00:00:00 is thought as a start and value 24:00:00 as the end of the set day.

For each day in a week there can be set one time period of operation.



Obr. 2.56 The structure of FB TProg1



Obr. 2.57 The appearance of FB TProg1

Variables description :

Term	Signification	Type	Format
TPg	Weekly schedule	Input	_TimeProg1_Week_
.Mon	Schedule for Monday		_TimeProg1_Day_
.T_ON	Start of operation		time
.T_OFF	End of operation		time
. . .	. . .		. . .
.Sun	Schedule for Sunday		_TimeProg1_Day_
.T_ON	Start of operation		time
.T_OFF	End of operation		time
Out	operation	Output	bool

Examples:

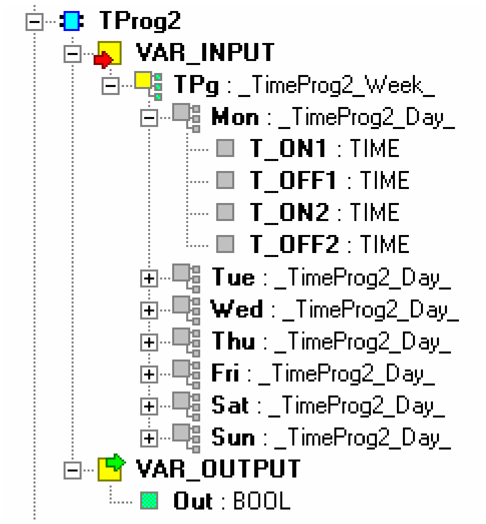
Mon.T_ON = 00:00:00, Mon.T_OFF = 24:00:00 ... continuous operating on Monday
Mon.T_ON = 06:30:00, Mon.T_OFF = 20:15:00 ... operating from 06:30 to 20:15 on Monday
Mon.T_ON = 00:00:00, Mon.T_OFF = 00:00:00 ... operating is switched off on Monday

2.7.2. TProg2 - weekly schedule with two ON/OFF intervals a day

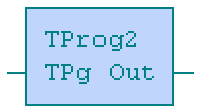
The function block sets on the basis of set weekly schedule and PLC system date the output operation signal *OUT*.

The variable *OUT* is in log.1 if the actual system date is between parameters T_ON1 and T_OFF1, or T_ON2 and T_OFF2 for a set day in a week, otherwise, it is in log.0. Variables can be set in a range from 00:00:00 to 24:00:00 where value 00:00:00 is thought as a start and value 24:00:00 as the end of the set day.

For each day in a week there can be set two time periods of operation.



Obr. 2.58 The structure of FB TProg2



Obr. 2.59 The appearance of FB TProg2

Variables description:

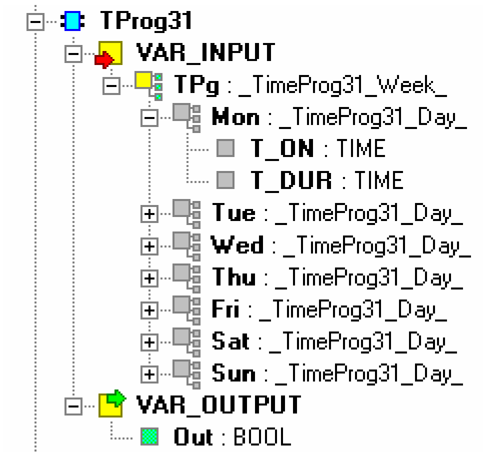
Term	**Signification**	**Type**	**Format**
TPg	Weekly schedule	Input	_TimeProg2_Week_
.Mon	Schedule for Monday		_TimeProg2_Day_
.T_ON1	Start of operation 1		time
.T_OFF1	End of operation 1		time
.T_ON2	Start of operation 2		time
.T_OFF2	End of operation 2		time
. . .	. . .		. . .
.Sun	Schedule for Sunday		_TimeProg2_Day_
.T_ON1	Start of operation 1		time
.T_OFF1	End of operation 1		time
.T_ON2	Start of operation 2		time
.T_OFF2	End of operation 2		time
Out	Operation	Output	bool

2.7.3. TProg31 - weekly schedule with one operating time

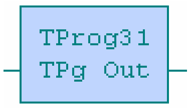
The function block sets on the basis of set weekly schedule and PLC system date the output operation signal *OUT*.

The variable *OUT* is in log.1 if the actual system date is greater than or equal to the parameter *T_ON* and less than *T_OFF* + parameter *T_DUR* for a set day in a week. Otherwise, the output OUT is in log.0. Variables *T-ON* and *T_DUR* can be set in a range from 00:00:00 to 24:00:00 where value 00:00:00 is thought as a start and value 24:00:00 as the end of the set day. During the set day it is possible in the sum of values *T_ON* and *T_DUR* to reach maximum value of 24:00:00.

For each day in a week there can be set one time period of operation.



Obr. 2.60 The structure of FB TProg31



Obr. 2.61 The appearance of FB TProg31

Variables description :

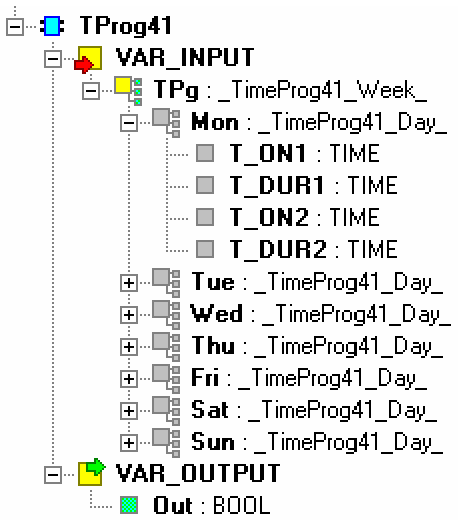
Term	Signification	Type	Format
TPg	Weekly schedule	Input	_TimeProg31_Week_
.Mon	Schedule for Monday		_TimeProg31_Day_
.T_ON	Start of operation		time
.T_DUR	Period of operation		time
. . .	. . .		. . .
.Sun	Schedule for Sunday		_TimeProg31_Day_
.T_ON	Start of operation		time
.T_DUR	Period of operation		time
Out	Operation	Output	bool

2.7.4. TProg41 - weekly schedule with two operating times

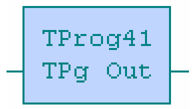
The function block sets on the basis of set weekly schedule and PLC system date the output operation signal *OUT*.

The variable *OUT* is in log.1 if the actual system date is greater than or equal to the parameter *T_ON* and less than *T_OFF* + parameter *T_DUR* for a set day in a week. Otherwise, the output OUT is in log.0. Variables *T-ON* and *T_DUR* can be set in a range from 00:00:00 to 24:00:00 where value 00:00:00 is thought as a start and value 24:00:00 as the end of the set day. During the set day it is possible in the sum of values *T_ON* and *T_DUR* to reach maximum value of 24:00:00.

For each day in a week there can be set two time periods of operation.



Obr. 2.62 The structure of FB TProg41



Obr. 2.63 The appearance of FB TProg41

Variables description :

Term	Signification	Type	Format
TPg	Weekly schedule	Input	_TimeProg41_Week_
.Mon	Schedule for Monday		_TimeProg41_Day_
.T_ON1	Start of operation 1		time
.T_DUR1	Period of operation 1		time
.T_ON2	Start of operation 2		time
.T_DUR2	Period of operation 2		time
. . .	. . .		. . .
.Sun	Schedule for Sunday		_TimeProg41_Day_
.T_ON1	Start of operation 1		time
.T_DUR1	Period of operation 1		time
.T_ON2	Start of operation 2		time
.T_DUR2	Period of operation 2		time
Out	Operation	Output	bool

3. IRCLIB LIBRARY

The IRCLib.mlb library contains only an IRC function block (Inteligent Room Control).

3.1. IRC – PALATINE MODULE

The function block is used for control of up to 16 room modules of Tecoreg IRC line (Tecoreg TR100) connected to a serial PLC channel. The communication with room modules is proceed via communications registry Tnet which is realized in the function block. Datas gained from room IRC modules are saved in PLC notebook in a global public data structure *_IRC_PS (this structure is created automaticly by adding a library* IRCLib.mlb into the project).

There is, for this structure, in the corporate visualization tool Reliance implemented a direct support (IRC component) and so there is not any concrete knowledge of the whole IRC structure needed.

Simultaneously with the data structure*_IRC_PS*, a control structure*_IRC_Flags* is created which contains tags for address implementation and time setting to IRC room modules.

Functions of address implementation and time setting are also accessible directly via inputs of the function block *SetAdr* a *SetTime*.

The function block assignment to the serial channel is realized via an input variable *Chnum* that contains the number of the serial channel configured for Tnet registry.

Communications zones of this serial channel are assigned to the block via input/output variables UNI_CH_IN and UNI_CH_OUT. It is necessery to interconnect these variables onto system variables UNI_CHx_IN and UNI_CHx_OUT (where x represents the number of selected serial channel).

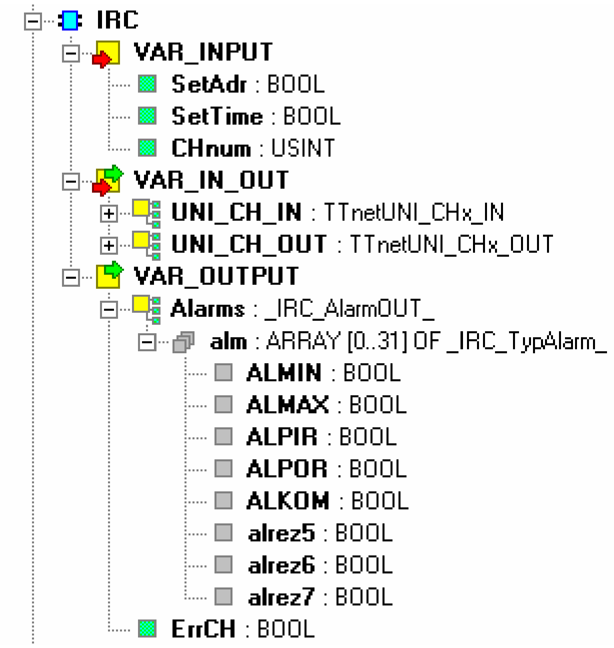
It is necessary to have one dedicated serial *PLC* communication channel for the use of IRC function block. This channel must be mounted by RS-485 interface and set to „uni“ mode. Communications parameters of this channel must be set in accordance with the picture 2.35.

The communication via Tnet registry will not be functional if parameters of the channel are set differently!!!

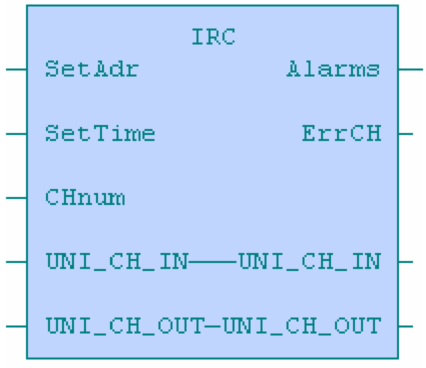
Simultaneously the output signal of *ErrCH* block is set. There are also alarm reports from individual room IRC modules directly accesible on the output of the block (alarms are included also in the public data structure *_IRC_PS).*

Warning!

The IRC function block can be used in every project at the most just once!!! Multiple usage of this block in one project leads to a data conflict within the public data structure*_IRC_PS.*



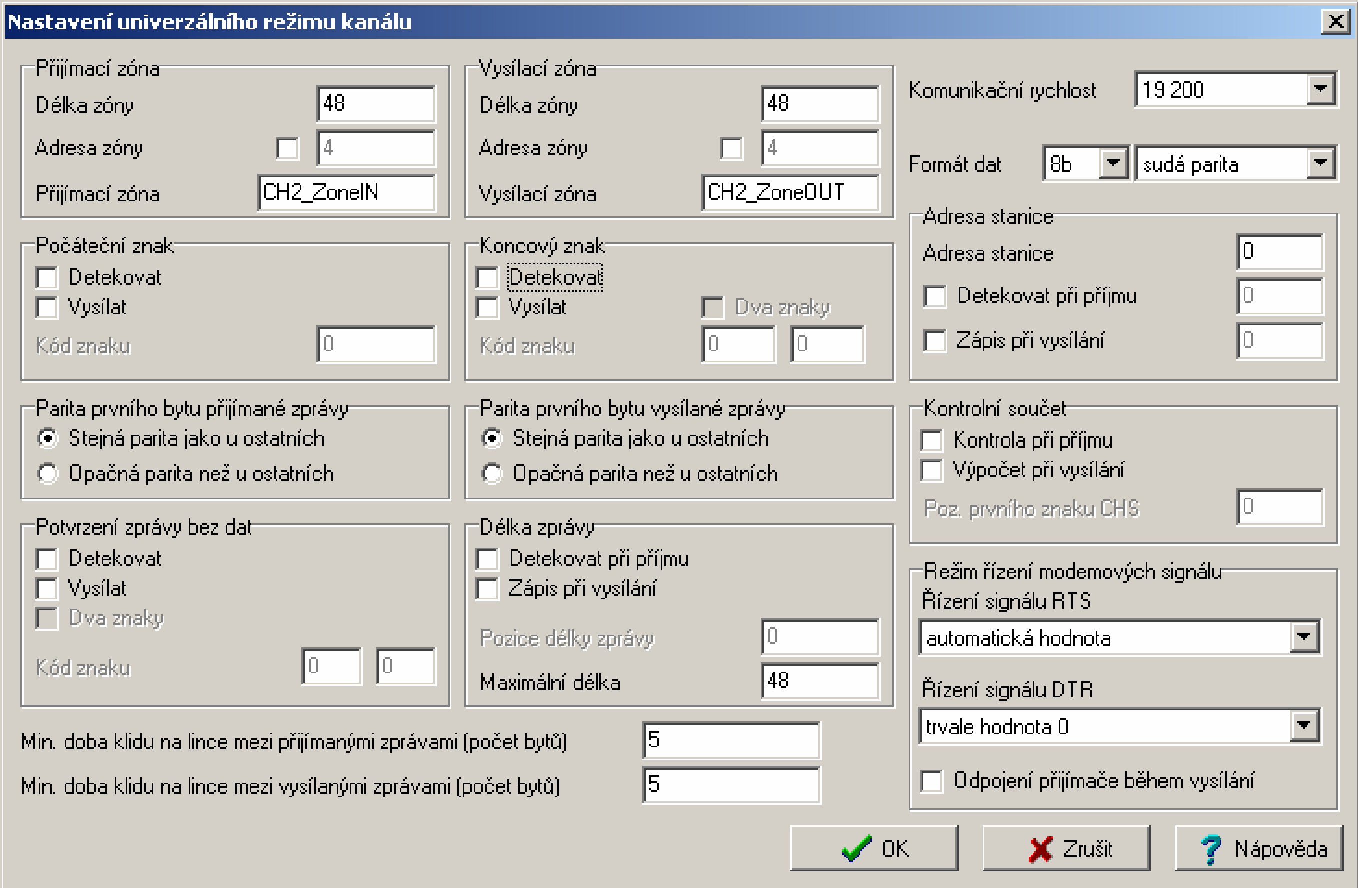
Obr. 3.64 The structure of FB IRC



Obr. 3.65 The appearance of FB IRC

Variables description :

Term	**Signification**	**Type**	**Format**
SetAdr	Tnet network address implementation		bool
SetTime	Tnet network time setting	Input	bool
CHnum	Serial channel No. (configurated for Tnet)		usint
UNI_CH_IN UNI_CH_OUT	Tnet input communication zone Tnet output communication zone	input/ output	TTnetUNI_CHx_IN TTnetUNI_CHx_OUT
Alarms	Output structure		_IRC_AlarmOUT_
.alm[X]	Room modules alarm reports		array [0..31] of _IRC_TypeAlarm_
.ALMIN	Area minimum temperature (<+8 ℃)		bool
.ALMAX	Area maximum temperature (>+39,5 ℃)	Output	bool
.ALPIR	breaking doors (if it is activated)		bool
.ALPOR	Room module failure		bool
.ALKOM	Failure of communication with the room module		bool
ErrCH	Failure of parameters of the serial channel		bool



Obr. 3.66 setting of the serial channel in the „uni" mode for Tnet registry (for channel CH2 here)

Structures *_IRC_Flags* and *_IRC_PS are surveyed in the PLC notepad and are intended for data transfer from/to a superior system(visualization SW Reliance)*.

The structure _IRC_Flags

The structure *_IRC_Flags* occupies in the PLC notepad 1 byte and contains only a flag register with these variables of bool type:

```
STRUCT
  SetAdr1   : bool;  (* address implementation of Tnet network *)
  dummy1    : bool;
  SetTime1  : bool;  (* time setting of Tnet network *)
  dummy3    : bool;
  dummy4    : bool;
  dummy5    : bool;
  dummy6    : bool;
  dummy7    : bool;
END_STRUCT;
```

The variable *SetAdr1* is used for activation of address implementation mode of room modules. The variable *SetTime1* is used for setting of system time of room modules (according to PLC system time).

The structure _IRC_PS

The structure *_IRC_PS* occupies in the PLC notepad 7200 bytes and is divided in to four data zones. These are operational datas, control datas, configuration datas and room modules alarms (alarms are accesible also as output variables of IRC function block).

```
STRUCT
 ProvTR : ARRAY [0..31] OF _IRC_ProvData_;(* room module oper. datas  *)
 ContTR : ARRAY [0..31] OF _IRC_ContData_;(* room module control datas *)
 KfgTR  : ARRAY [0..31] OF _IRC_KfgData_; (* room module config. datas*)
 AlmTR  : ARRAY [0..31] OF _IRC_TypAlarm_;(* room modules alarms *)
END_STRUCT;
```

The whole structure*_IRC_PS* is designed for 32 IRC room modules, however, the IRC function block contains only 16 modules within addresses 0-15. The concrete meaning of individual items of the whole structure is described in TXV 138 04 documentation, Technical equipment of communication module TR101, chapter Public data structure. Compared to TR101 is this structure not fixly surveyed (it is R100 register in TR101).

The example of use at ST

If the IRC function block is used while programming at ST language, than it is necessary while calling the block to suppres a type control of input/output variables UNI_CH_IN and UNI_CH_OUT via a directive void(). The source text of calling FB IRC will then look e.g. like this:

```
iIRC(CHnum:=2, UNI_CH_IN:=void(UNI_CH2_IN), UNI_CH_OUT:=void(UNI_CH2_OUT));
```

IRC link on to Reliance

There is in the corporate visualization system Reliance prepared a direct support for IRC system in the form of IRC component. This component enables a user friendly parametrization as well as IRC system operation itself. It operates above the public data structure *_IRC_PS* and for correct operation it requires information on location of this data structure in the PLC notepad. In Mosaic environment is this information accesible in menu *Tools->Map of user registers,* variable*_IRC_PS.*

Notes :

Notes :

Objednávky a informace:
Teco a. s. Havlíčkova 260, 280 58 Kolín 4, tel. 321 737 611, fax 321 737 633

TXV 003 23.02