OSCAT

Building:LIBRARY

Dokumentation

Version 1.00

Inhaltsverzeichnis

1. Rechtsgrundlagen

1.1. Haftungsausschluss

Die in der OSCAT Bibliothek enthaltenen Softwaremodule sind in der Absicht angeboten, als Entwicklungsvorlage und Leitfaden zur Softwareentwicklung für SPS nach IEC61131-3 zu dienen. Eine Funktionsgarantie wird von den Entwicklern nicht übernommen und wird explizit ausgeschlossen. Da die in der Bibliothek enthaltenen Softwaremodule ohne jegliche Kosten bereitgestellt werden, besteht keinerlei Gewährleistung, soweit dies gesetzlich zulässig ist. Sofern nicht explizit schriftlich vereinbart, stellen die Copyright-Inhaber und / oder Dritte die Softwaremodule so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich - aber nicht begrenzt - auf Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko und die volle Verantwortung bezüglich Qualität, Fehlerfreiheit und Leistungsfähigkeit der Softwaremodule liegen beim Anwender selbst. Sollte sich die Bibliothek, oder Teile der Bibliothek als fehlerhaft erweisen, liegen die Kosten für notwendigen Service, Reparatur und / oder Korrektur beim Anwender selbst. Sollten Teile oder die gesamte Bibliothek zur Erstellung von Anwendungssoftware verwendet werden, oder in Softwareprojekten eingesetzt werden, so haftet der Anwender für die Fehlerfreiheit, Funktion und Qualität der Anwendung. Eine Haftung durch OSCAT ist grundsätzlich ausgeschlossen.

Der Anwender der OSCAT Bibliothek hat durch geeignete Tests und Freigaben und Qualitätssicherungsmaßnahmen dafür zu sorgen, dass durch eventuelle Fehler in der Bibliothek von OSCAT keine Schäden entstehen können. Die vorstehenden Lizenzbedingungen und Haftungsausschlüsse gelten gleichermaßen für die Softwarebibliothek, sowie die in diesem Handbuch angebotenen Beschreibungen und Erläuterungen, auch wenn dies nicht explizit erwähnt wird.

1.2. Lizenzbedingungen

Die Verwendung der OSCAT Bibliothek ist kostenfrei und kann ohne Lizenzvereinbarung für private oder gewerbliche Zwecke eingesetzt werden. Eine Verbreitung der Bibliothek ist ausdrücklich erwünscht, hat aber kostenfrei und unter Hinweis auf unsere Webpage WWW.OSCAT.DE zu erfolgen. Wird die Bibliothek in elektronischer Form zum Download bereitgestellt oder auf Datenträgern verbreitet, so ist sicherzustellen, dass ein deutlich erkennbarer Hinweis auf OSCAT und ein Weblink zu WWW.OSCAT.DE entsprechend enthalten sind.

1.3. Eingetragene Markenzeichen

Alle in dieser Beschreibung benutzten Markennamen werden ohne Verweis auf deren Eintragung beziehungsweise Besitzer verwendet. Die Existenz solcher Rechte kann daher nicht ausgeschlossen werden. Die benutzten Markennamen sind Eigentum des jeweiligen Besitzers. Eine Verwendung der Beschreibung für kommerzielle Zwecke ist daher nicht gestattet, auch nicht auszugsweise.

1.4. Bestimmungsgemäßer Gebrauch

Die in der OSCAT Bibliothek enthaltenen und in dieser Dokumentation beschriebenen Softwaremodule sind ausschließlich für Fachpersonal mit einer Ausbildung in SPS Programmierung entwickelt worden. Die Anwender sind verantwortlich für die Einhaltung aller geltenden Normen und Vorschriften, die bei der Anwendung zum Tragen kommen. OSCAT weist weder im Handbuch noch in der Software auf diese Normen und Vorschriften hin.

1.5. Sonstiges

Alle rechtsverbindlichen Regelungen befinden sich ausschließlich im Kapitel 1 des Benutzerhandbuchs. Eine Ableitung oder Bezug von rechtlichen Ansprüchen aufgrund des Inhalts des Manuals, außer den Bestimmungen in Kapitel 1, ist gänzlich ausgeschlossen.

2. Einleitung

2.1. Ziele

OSCAT steht für "Open Source Community for Automation Technology".

OSCAT erstellt eine Open Source Bibliothek nach dem IEC61131-3 Standard, welche auf herstellerspezifische Funktionen verzichtet und deshalb auf alle IEC61131-3 kompatiblen Speicherprogrammierbaren Steuerungen portiert werden kann. Auch wenn Entwicklungen für SPS unter dem Einsatz von herstellerspezifischen Bibliotheken meist effizient zu lösen sind und diese Bibliotheken auch teilweise kostenfrei zur Verfügung gestellt werden, ergeben sich doch große Nachteile durch ihren Einsatz:

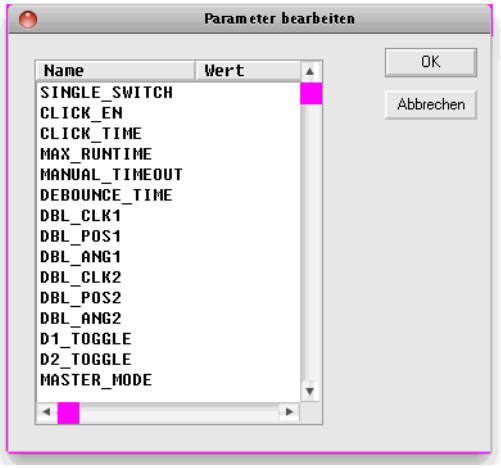
1. Die Bibliotheken der Hersteller sind praktisch alle geschützt und der Source Code ist nicht frei zugänglich, was im Fehlerfall eine Fehlersuche und auch die Behebung des Fehlers enorm schwierig, oft sogar unmöglich macht.
2. Die grafische Erstellung von Programmen kann mit herstellerspezifischen Bibliotheken schnell unübersichtlich, ineffizient und fehleranfällig werden, weil vorhandene Funktionen wegen des fehlenden Source Codes den eigentlichen Bedürfnissen nicht angepasst und erweitert werden können.
3. Ein Wechsel der Hardware, insbesondere der Wechsel zu einem anderen Hersteller, ist durch die geschützten Bibliotheken verhindert und die Vorteile, die ein Standard wie IEC61131 bieten würde, werden so eingeschränkt. An einen Austausch einer herstellerspezifischen Bibliothek mit der eines Wettbewerbers ist ausgeschlossen, denn die Bibliotheken der Hersteller unterscheiden sich enorm in Umfang und Inhalt.
4. Das Verständnis komplexer Module ist ohne einen Einblick in den Sourcecode oft sehr schwierig. Dadurch werden Programme ineffizient und fehleranfällig.

OSCAT will mit der quell offenen OSCAT Bibliothek einen mächtigen und umfassenden Standard für die Programmierung von SPS schaffen, der im Source Code zur Verfügung steht und durch vielfältige Anwendungen ausführlich verifiziert und getestet wurde. Weiterhin fließen durch die Vielzahl der Anwendungen umfangreiche Erkenntnisse und Anregungen in die Bibliothek ein. Dadurch kann die Bibliothek als sehr praxisnah bezeichnet werden. OSCAT versteht seine Bibliothek als Entwicklungsvorlage und nicht als ausgereiftes Produkt. Der Nutzer ist selbst verantwortlich dafür, eventuell in seiner Anwendung verwendete Module mit geeigneten Verfahren zu testen und die notwendige Fehlerfreiheit, Qualität und Funktionalität zu verifizieren. An dieser Stelle sei noch einmal auf die Lizenz-

bedingungen und den Haftungsausschluss in dieser Dokumentation hingewiesen.

2.2. Konventionen

1. Direkte Manipulationen im Speicher:
   Funktionen, die Eingangswerte über Pointer verändern, wie zum Beispiel _Array_Sort, beginnen alle mit einem Unterstrich „_“. _Array_Sort sortiert ein Array direkt im Speicher, was den gravierenden Vorteil hat, dass ein eventuell sehr großes Array erst gar nicht der Funktion übergeben werden muss und deshalb Speicher in der Größe des Arrays und die Zeit zum Kopieren eingespart wird. Es wird allerdings nur erfahrenen Anwendern empfohlen diese Funktionen einzusetzen, da eine Fehlanwendung zu gravierenden Fehlern und Abstürzen führen kann! Bei Anwendung von Funktionen die mit einem „_“ beginnen ist besondere Sorgfalt angebracht und insbesondere darauf zu achten, dass die Aufrufparameter niemals undefinierte Werte annehmen können.
2. Namensgebung bei Funktionen:
   Funktionsbausteine mit Zeitverhalten, wie etwa die Funktion PT1 werden durch die Namensgebung FT_Bausteinname (FT_PT1) beschrieben. Funktionen ohne Zeitbezug sind mit F_Bausteinname angegeben.
3. Logische Gleichungen:
   Innerhalb dieses Handbuchs werden die logischen Verknüpfungen & für UND bzw. AND, + für ODER bzw. OR, /A für ein negiertes A und # für XOR (exklusives ODER) verwendet.
4. Setup-Werte für Bausteine:
   Damit die Anwendung und Programmierung übersichtlich bleibt und komplexe Funktionen einfacher dargestellt werden können, haben viele der Bausteine der OSCAT Bibliothek einstellbare Parameter, die bei Anwendung durch einen Doppelklick auf das grafische Symbol des Bausteins bearbeitet werden können. Ein Doppelklick auf das Symbol öffnet eine Dialogbox, die das Bearbeiten der Setup-Werte erlaubt. Wird eine Funktion mehrfach verwendet, so können damit je Baustein die Setup-Werte einzeln festgelegt werden. Die Bearbeitung durch Doppelklick funktioniert unter CoDeSys ausschließlich in CFC. In ST müssen alle Parameter,



auch die Setup-Parameter, im Aufruf übergeben werden. Die Setup-Parameter werden einfach nach den normalen Eingängen angefügt. Die Parameter werden in der grafischen Oberfläche durch Doppelklick bearbeitet und dann wie Konstanten unter IEC61131 eingegeben. Es ist Dabei zu beachten dass Zeiten mit T#200ms und TRUE und FALSE in Großbuchstaben geschrieben sein müssen.

5. Error- und Status-Reporting (ESR):

   Komplexere Bausteine erhalten zum großen Teil einen Error- oder Status-Ausgang. Ein Error-Ausgang ist 0, falls kein Fehler bei der Ausführung auftritt. Tritt jedoch im Baustein ein Fehler auf, so nimmt dieser Ausgang einen Wert im Bereich 1 .. 99 an und meldet somit einen Fehler mit Fehlernummer. Ein Status- oder Error-Sammelmodul kann diese Meldungen sammeln und mit einem Zeitstempel versehen, in einer Datenbank bzw. Array speichern, oder per TCP/IP an übergeordnete Systeme weiterleiten. Ein Ausgang des Typs Status ist kompatibel zu einem Error-Ausgang mit identischer Funktion. Jedoch meldet ein Status-Ausgang nicht nur Fehler, sondern führt auch über Aktivitäten des Bausteins Protokoll. Werte zwischen 1 .. 99 sind weiterhin Fehlermeldungen. Zwischen 100 .. 199 befinden sind Meldungen über Zustandsveränderungen. Der Bereich 200 .. 255 ist reserviert für Debug-Meldungen. Mit dieser, innerhalb der OSCAT Bibliothek standardisierten Funktionalität, wird eine einfache und übergreifende Möglichkeit geboten, Betriebszustandsmeldungen und Fehlermeldungen auf einfache Weise zu integrieren, ohne die Funktion eines Systems zu beeinflussen. Bausteine, die dieses Verfahren unterstützen, werden ab der Revision 1.4 mit der Kennzeichnung „ESR-Fähig“ gekennzeichnet. Weitere Informationen zu den ESR-Bausteinen finden Sie im Abschnitt „Diverse Funktionen“.

2.3. Testumgebung

Die OSCAT Bibliothek wird unter CoDeSys entwickelt und auf verschiedenen Systemen getestet.

Die Testumgebung umfasst folgende Systeme:

1. Beckhoff BX 9000
   mit TwinCAT PLC Control Version 2.10.0
2. Beckhoff CX 9001-1001
   mit TwinCAT PLC Control Version 2.10.0
3. Wago 750-841
   mit CoDeSys Version 2.3.9.31
4. Möller EC4P222
   mit CoDeSys Version 2.3.9.31

5. CoDeSys Simulation auf I386 mit CoDeSys 2.3.9.31
6. CoDeSys Simulation auf i386 mit Codesys 3.4
7. S7 und STEP7: Die OSCAT Bibliothek wird seit Version 1.5 auf STEP7 übersetzt und auch verifiziert.
8. PCWORX / MULTIPROG: Die OSCAT Bibliothek wird seit Version 2.6 auf MULTIPROG übersetzt und verifiziert.
9. Bosch Rexroth IndraLogic XLC L25/L45/L65 mit Indraworks 12VRS
10. Bosch Rexroth IndraMotion MLC L25/L45/L65 mit Indraworks 12VRS
11. Bosch Rexroth IndraMotion MTX L45/L65/L85 mit Indraworks 12VRS

Wir sind stetig darum bemüht die OSCAT Bibliothek auch auf weiteren Testumgebungen zu testen.

2.4. Aktualität

OSCAT aktualisiert diese Beschreibung fortlaufend. Es wird empfohlen sich die jeweils aktuellste Version von der OSCAT Homepage unter www.OSCAT.DE zu laden. Hier wird das jeweils aktuellste Manual zum Download bereitgestellt. Neben dem Manual stellt OSCAT auch eine detaillierte Revision Historie bereit. Die "OSCAT Revision History" listet alle Revisionen der einzelnen Bausteine mit Änderungen und ab welcher Release der Bibliothek dieser Baustein enthalten ist.

2.5. Support

Support wird durch die Vielzahl der Anwender selbst im Forum unter WWW.OSCAT.DE zur Verfügung gestellt. Ein Anspruch auf Support besteht aber generell nicht, auch dann nicht, wenn sich herausstellen sollte, dass die Bibliothek oder Teile der Bibliothek fehlerhaft sind. Der im Rahmen des OSCAT Forums bereitgestellte Support wird von Anwendern freiwillig und untereinander bereitgestellt. Updates der Bibliothek und der Dokumentation werden in der Regel einmal im Monat auf der Homepage von OSCAT unter WWW.OSCAT.DE zur Verfügung gestellt. Ein Anspruch auf Wartung, Fehlerbehebung und Softwarepflege jedweder Art besteht generell nicht und ist als freiwillige Leistung von OSCAT anzusehen. Bitte senden Sie bei Support Anfragen keine email an OSCAT. Anfragen können schneller und effektiver bearbeitet werden wenn die Anfragen in unserem Forum gestellt werden.

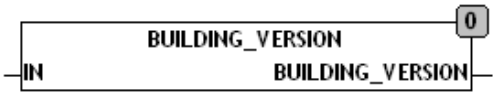
3. Diverse Funktionen

3.1. BUILDING_VERSION

Type Funktion : DWORD

Input IN : BOOL (wenn TRUE liefert der Baustein das Release Datum)

Output (Version der Bibliothek)



BUILDING_VERSION gibt wenn IN = FALSE die aktuelle Versionsnummer als DWORD zurück. Wird IN auf TRUE gesetzt so wird das Release Datum der aktuellen Version als DWORD zurückgegeben.

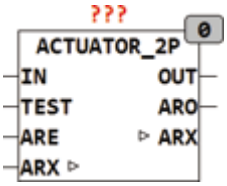
Beispiel: BUILDING_VERSION(FALSE) = 100 für Version 1.00

DWORD_TO_DATE(OSCAT_VERSION(TRUE)) = 2011-1-30

4. Stellglieder

4.1. ACTUATOR_2P

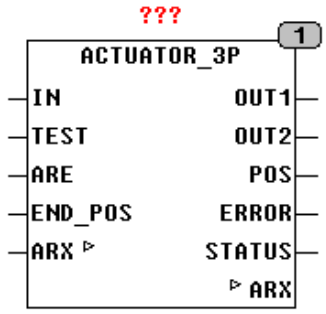
Type	Funktionsbaustein
Input	IN : BYTE (Steuereingang 0 - 255)
	TEST : BOOL (startet Autorun wenn TRUE)
	ARE : BOOL (Enable für Autorun)
I/O	ARX : BOOL (Autorun Signal Bus)
Output	OUT : BOOL (Schaltsignal für Ventil)
	ARO : BOOL (TRUE wenn Autorun aktiv ist)
Setup	CYCLE_TIME : TIME (Taktrate des Ventils)
	SENS : BYTE (Minimale und Maximale Eingangswerte)
	SELF_ACT_TIME : TIME (Selbstbetätigungszeit)
	SELF_ACT_PULSE : TIME (Schaltzeit bei Autorun)
	SELF_ACT_CYCLES : INT (Anzahl Zyklen bei Autorun)



ACTUATOR_2P ist ein Interface für 2-Punkt Aktuatoren wie z B. Magnetventile. Der 2-Punkt Aktuator kann nur Ein / Aus Schalten und deshalb wird der Eingangswert IN in ein Puls / Pause Signal am Ausgang OUT gewandelt. Die Zykluszeit (CYCLE_TIME) bestimmt die Schaltzeiten des Ausgangs. Damit ein Festkleben des Ventils durch langes ruhen verhindert wird, kann durch einstellen der Selbstbetätigungszeit (SELF_ACT_TIME) und der Anzahl der Selbstaktivierungszyklen (SELF_ACT_CYCLES) sowie der Impulsdauer (SELF_ACT_PULSE) bestimmt werden, nach welcher Zeit wie viele Schaltzyklen automatisch ausgeführt werden, um ein festkleben des Ventils zu verhindern. Nach Ablauf der Zeit SELF_ACT_TIME prüft der Baustein ob ARE = TRUE und ARX = FALSE sind und schaltet dann ARO für die Dauer der Selbstaktivierung auf TRUE. Gleichzeitig wird ARX auf TRUE gesetzt um zu verhindern das andere Bausteine die an ARX angeschlossen sind gleichzeitig in den Autorun gehen. Der Eingangswert IN kann von 0..255 variiert werden. Ist das Eingangssignal IN < SENS bleibt das Ventil dauernd geschlossen (OUT = FALSE) und IN > 255 - SENS bedeutet das Ventil ist dauernd offen (OUT = TRUE).

4.2. ACTUATOR_3P

Type	Funktionsbaustein
Input	IN : BYTE (Eingang Steuersignal 0 - 255)
	TEST : BOOL (Baustein führt Diagnose aus wenn TRUE)
	ARE : BOOL (Auto Diagnose ist erlaubt wenn TRUE)
	END_POS : BOOL (Eingang für Endschalter)
Output	OUT1 : BOOL (Steuersignal für Klappe in Richtung Auf)
	OUT2 : BOOL (Steuersignal für Klappe in Richtung Zu)
	POS : BYTE (Simulierte Klappenstellung)
	ERROR : BOOL (TRUE wenn Diagnosefehler)
	STATUS : BYTE (ESR kompatibler Status Ausgang)
I/O	ARX : BOOL (Autorun Kommunikation)
Setup	T_RUN : TIME (Laufzeit für volle Bewegung 0 - 255)
	T_EXT : TIME (Laufzeitverlängerung bei Diagnose)
	T_CAL : TIME (Klappenlaufzeit bis zur Kalibrierung)
	T_DIAG : TIME (Zeitspanne für Autodiagnose)
	SWITCH_AVAIL : BOOL (TRUE, wenn Endschalter angeschlossen ist)



ACTUATOR_3P ist ein 3-Punkt Aktuator Interface zum Ansteuern von Stellmotoren mit Auf / Ab Eingang. Das Signal am Eingang IN wird umgesetzt in Steuerimpulse an den Ausgängen OUT1 und OUT2 die den Motor entsprechend Steuern. Das Eingangssignal IN wird so verarbeitet und die beiden Steuerausgänge (OUT1 und OUT2) so gesteuert, dass ein Eingangswert von 0 Klappe geschlossen, 255 Klappe offen, 127 Klappe halb geöffnet usw. bewirkt. Der Baustein kann auch einen Endschalter verarbeiten. Die Endschalter müssen so angeschlossen werden, dass egal ob oberes oder unteres Ende erreicht wurden, der Eingang END_POS TRUE wird und damit anzeigt, dass die Klappe eine der beiden Endstellungen erreicht hat.

Um die Endschalterfunktion in Betrieb zu setzen muss die Setup-Variable SWITCH_AVAIL auf TRUE stehen, ansonsten wird der Endschalter ignoriert. Der Diagnose Eingang TEST kann zu jederzeit eine Klappen und Motor-Diagnose starten. Der Baustein durchläuft dann einen Diagnosezyklus und meldet eventuelle Fehler am Ausgang ERROR. Ein Diagnosezyklus fährt die Klappe zurück auf 0%, vermisst dann die Laufzeit von 0% - 100% und wieder zurück auf 0%. Er prüft auch, ob Endschalter funktionieren (falls diese durch die Setup-Variable SWITCH_AVAIL aktiviert wurden). Nach dem Diagnose-Zyklus fährt die Klappe wieder in die durch den Eingang IN definierte Stellung. Die während der Diagnose gemessenen Laufzeiten werden im Betrieb verwendet um die Klappe extrem genau auf die jeweils geforderte Position zu bewegen. Mit der Setup-Variable T_DIAG wird spezifiziert, nach welcher Zeit eine Diagnose selbständig ohne durch den Eingang TEST aktiviert zu werden, durchgeführt wird. Nach dem Einschalten wird automatisch immer ein Diagnose-Zyklus durchgeführt. Ist der Wert T_DIAG = T#0s, wird keine automatische Diagnose durchgeführt.

Eine Klappe wird Üblicherweise Auf und Ab bewegt um verschiedene Volumenströme einzustellen. Je mehr sich eine Klappe bewegt, desto mehr weicht sie von einer idealen absoluten Position ab, weil bei jeder Bewegung ein kleiner Positionsfehler auftritt und sich über viele Bewegungen addiert. Um diesem Fehler entgegen zu Wirken kann mit der Setup Variablen T_CAL nach einer definierten Laufzeit (aufaddierte Zeit aller Klappenbewegungen) der Klappe eine Kalibrierung automatisch durchgeführt werden. Bei dieser Kalibrierung fährt der Motor in Nullstellung und stellt die Klappe anschließend wieder auf den durch IN spezifizierten Wert. Ein Wert von T#0s für die CAL_RUNTIME bedeutet, dass keine automatische Kalibrierung durchgeführt wird.

Bei Kalibrierung und Diagnose ohne Endschalter wird für eine volle Bewegung die Zeit T_EXT zur Laufzeit T_RUN addiert um sicherzustellen das die Klappe seine Endposition auch ohne Endschalter sicher erreicht.

Am Ausgang POS simuliert der Baustein die aktuelle Klappenstellung mittels der eingestellten Zeit T_RUN. An diesem Ausgang kann auch festgestellt werden wann die Klapp die am Eingang angeforderte Stellung erreicht hat. Wird der Eingang TEST = TRUE gesetzt führt der Baustein einen Diagnosezyklus durch. Über die externe Variable ARX kommunizieren mehrere Bausteine miteinander und sorgen selbständig dafür das Diagnosezyklen nach dem Einschalten nacheinander und nicht Parallel ausgeführt werden. Der Anwender legt dabei fest wie viele und welche Bausteine an die gleiche Variable geschaltet werden und sich dadurch abstimmen können. Wird jeder Baustein an eine eigenen Variable ARX geschaltet erfolgt keine koordination der Diagnosezyklen. Weitere Informationen zu den Eingängen TEST, ARE und ARX ist beim Baustein Autorun nachzulesen.

Statusmeldungen des Bausteins:

STATUS		ARE	ARX
100	Normal operation	-	-

101	Calibration	-	-
103	Diagnostic UP	TRUE	TRUE
104	Diagnostic DN	TRUE	TRUE

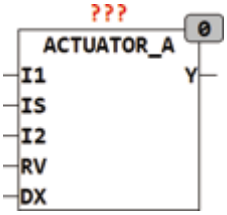
4.3. ACTUATOR_A

Type Funktionsbaustein

Input I1: BYTE (Steuersignal 1)
IS : BOOL (Eingangs Auswahl)
I2 : BYTE (Steuersignal 2)
RV : BOOL (Richtungsumkehr für Ausgang Y)
DX : BOOL (Selbstaktivierung)

Setup RUNTIME : TIME (Laufzeit des Stellmotors)
SELF_ACT_TIME : TIME (Zeit für automatische Bewegung)
OUT_MIN : DWORD (Ausgangswert bei I = 0)
OUT_MAX : DWORD (Ausgangswert bei I = 255)

Output Y : WORD (Steuersignal für den Stellmotor)



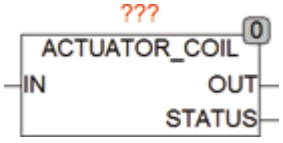
ACTUATOR_A dient zur Ansteuerung von Stellmotoren mit analog Eingang. Der Baustein hat zwei Eingänge (I1 und I2) die im Bereich 0..255 den gesamten Ausgangsbereich an Y abdecken. Der Ausgang Y ist vom Typ WORD, und sein Schaltbereich wird durch die Setup Werte OUT_MIN und OUT_MAX vorgegeben. Ein Eingangswert von 0 erzeugt den Ausgangswert OUT_MIN und ein Eingangswert von 255 erzeugt den Ausgangswert OUT_MAX, andere Eingangswerte erzeugen entsprechende Ausgangswerte zwischen OUT_MIN und OUT_MAX. Der Baustein kann direkt zur Ansteuerung von DA Wandlern mit 16Bit Eingang verwendet werden. Der Eingang IS selektiert zwischen zwei Eingängen I1 und I2, somit kann z.B. zwischen Hand und Automatikbetrieb umgeschaltet werden. Ein weiterer Eingang

DX schaltet bei steigender Flanke unmittelbar eine Selbstaktivierung ein. wenn SELF_ACT_TIME > t#0s dann wird die Selbstaktivierung nach Ablauf der Zeit SELF_ACT_TIME automatisch wiederholt, dabei schaltet der Ausgang Y für die Zeit RUNTIME auf OUT_MIN, anschließend für die gleiche Zeit auf OUT_MAX und kehrt danach wieder zum normalen Stellwert zurück. Der Eingang RV kann den Ausgang Invertieren, Y = OUT_MAX wenn I = 0 und Y = OUT_MIN wenn I = 255. Auf diese Weise kann ganz einfach die Laufrichtung des Stellmotors umgekehrt werden.

IS	RV	DX	Y
0	0	0	Y = (OUT_MAX-OUT_MIN) * I1 / 255 +OUT_MIN
1	0	0	Y = (OUT_MAX-OUT_MIN) * I2 / 255 +OUT_MIN
0	1	0	Y = OUT_MAX - (OUT_MAX-OUT_MIN) * I1 /255
1	1	0	Y = OUT_MAX - (OUT_MAX-OUT_MIN) * I2 /255
-	-	¿	startet einen Selbstaktivierungszyklus

4.4. ACTUATOR_COIL

Type	Funktionsbaustein
Input	IN: BOOL (Steuersignal)
Setup	SELF_ACT_CYCLE : TIME (Automatische Aktivierungszeit)
	SELF_ACT_TIME : TIME (Einschaltzeit bei Autoaktivierung)
Output	OUT : BOOL (Steuersignal für die Pumpe)
	STATUS : BYTE (ESR kompatibler Statusausgang)



ACTUATOR_COIL dient zur Ansteuerung von einfachen Ventilen. Der Ausgang OUT folgt dabei dem Eingangssignal IN. Wird die Setup Variable

SELF_ACT_CYCLE auf einen Wert größer 0 gesetzt, wird das Ventil Automatisch für die Dauer von SELF_ACT_TIME aktiviert falls es für die Zeit SELF_ACT_CYCLE ausgeschaltet war. Ein ESR kompatibler Statusausgang meldet Zustandsänderungen des Ventils zur Weiterverarbeitung oder zum Data Logging. Die Statusmeldungen sind wie folgt definiert:

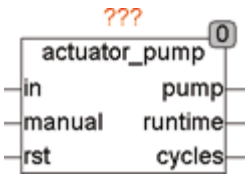
STATUS = 100, Standby.

STATUS = 101, Ventil wurde durch TRUE am Eingang IN aktiviert.

STATUS = 102, Ventil wurde Automatisch aktiviert.

4.5. ACTUATOR_PUMP

Type	Funktionsbaustein
Input	IN: BOOL (Steuersignal für Pumpe)
	MANUAL : BOOL (Manuelles Steuersignal)
	RST : BOOL (Reset Signal)
Output	PUMP : BOOL (Steuersignal für die Pumpe)
	RUNTIME : REAL (Betriebsstunden des Motors in Stunden)
	CYCLES : REAL (Anzahl der Ein / Aus Zyklen der Pumpe)
Setup	MIN_ONTIME : TIME (Minimale Laufzeit für Motor)
	MIN_OFFTIME : TIME (Minimale Totzeit für Motor)
	RUN_EVERY : TIME (Zeit nach der die Pumpe selbsttätig läuft)



ACTUATOR_PUMP ist ein Pumpeninterface mit Betriebsstundenzähler. Die Pumpe kann sowohl mit IN oder Manual eingeschaltet werden. Die Setup-Variablen MIN_ONTIME und MIN_OFFTIME legen eine minimale Einschaltdauer und eine minimale Laufzeit fest. Wird der Eingang IN kürzer als MIN_ONTIME auf TRUE gesetzt, so läuft die Pumpe weiter bis die minimale Laufzeit erreicht ist. Wird der Eingang IN länger als MIN_ONTIME auf TRUE gesetzt, läuft die Pumpe bis IN wieder FALSE ist.

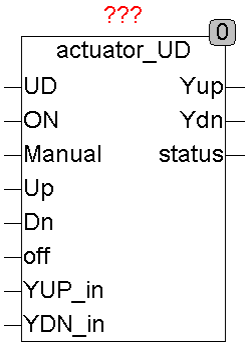
Wird die Pumpe kurz hintereinander eingeschaltet, so wartet die Pumpe bis die Zeit MIN_OFFTIME verstrichen ist, bis sie die Pumpe wieder einschaltet. Mit der Setup-Variablen RUN_EVERY wird die Zeit definiert, nach der die Pumpe selbsttätig läuft, wenn sie länger als RUN_EVERY stillsteht

um ein festsitzen der Pumpe zu vermeiden. Die Pumpe schaltet in diesem Fall selbsttätig ein und läuft für MIN_ONTIME. Durch RUN_EVERY = T#0s kann die automatische Aktivierung abgeschaltet werden.

Ein interner Betriebsstundenzähler zählt die Laufzeit der Pumpe in Stunden und auch die Anzahl der Schaltzyklen. Beide Werte können mit TRUE am Eingang RST auf Null zurückgesetzt werden. Der Betriebsstundenzähler ist permanent und geht weder bei Stromausfall oder Reset verloren. RUNTIME und CYCLES sind beides REAL-Werte, damit nicht der übliche Overflow wie bei TIME Werten nach 50 Tagen passiert.

4.6. ACTUATOR_UD

Type	Funktionsbaustein
Input	UD : BOOL (Richtungseingang in Auto Mode UP=TRUE)
	ON : BOOL (TRUE, wenn im Auto Mode)
	MANUAL : BOOL (TRUE, wenn Manual Mode)
	UP : BOOL (UP enable in Manual Mode)
	DN : BOOL (DN enable in Manual Mode)
	OFF : BOOL (Sicherheitsabschalter TRUE = Ausgänge FALSE)
	YUP_IN : BOOL (Rückführungseingang UP Relais)
	YDN_IN : BOOL (Rückführungseingang DN Relais)
Output	YUP : BOOL (Ausgang für Richtung UP)
	YDN : BOOL (Ausgang für Richtung DN)
	STATUS : Byte (ESR kompatibler Status und Fehler Ausgang)
Config	TON : TIME (minimale Einschaltzeit)
	TOFF : TIME (minimale Ausschaltzeit)
und	OUT_RETURN : BOOL (schaltet die Rückführeingänge YUP_In YDN_in ein)



ACTUATOR_UD ist eine Wendeschützinterface mit Verriegelung und konfigurierbarem Timing. Mit zusätzlichen Rückführeingängen wird eine Aktivierung verhindert solange ein Relais klemmt. Der Baustein kennt einen Automatik und einen Handbetrieb. Im Automatikmodus (ON = TRUE und Manual = FALSE) entscheidet der Eingang UD über die Richtung und ON über Ein / Aus. Sobald der Manual Eingang TRUE wird beginnt der Manual Modus und die Ausgänge folgen nur den Eingängen UP und DN. UP und DN dürfen nie gleichzeitig TRUE sein, falls trotzdem werden beide Ausgänge FALSE. Mit einem Sicherheitsausschalteingang OFF können sowohl im Manual als auch im Automatik Modus jederzeit die Ausgänge abgeschaltet werden.
Zwei Rückführeingänge YUP_IN und YDN_IN dienen dazu über separate Eingänge den Zustand der Schaltrelais auf den Baustein zurückzuführen und bei versagen eines Relais das aktivieren des anderen Ausgangs zu vermeiden. Dieser Fehler wird auch durch Fehlermeldungen am Ausgang STATUS gemeldet. Die Rückmeldefunktion ist jedoch nur verfügbar wenn die Config Variable OUT_RETURN auf TRUE gesetzt wird. Status meldet auch alle Aktivitäten des Bausteins um sie für eine Datenaufzeichnung zur Verfügung zu stellen. Der Status Ausgang ist ESR kompatibel und mit anderen ESR Modulen aus unserer Bibliothek kombinierbar.
Der Ausgang Status meldet 2 Fehler:

1 : YUP kann nicht gesetzt werden weil YDN_IN TRUE ist.

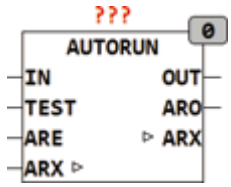
2 : YDN kann nicht gesetzt werden weil YUP_IN TRUE ist.

Mit den Config Variablen TON und TOFF kann eine Mindeste Einschaltzeit und eine Mindeste Totzeit zwischen 2 Ausgangsimpulsen definiert werden um das Schalten großer Motoren oder Getriebe die ein An und Auslaufen benötigen zu ermöglichen.

0	-	-	1	1	0	1	0	111
0	-	-	1	0	0	0	1	112
-	-	-	-	-	1	0	0	101
1	0	0	0	-	0	0	0	110
0	-	-	0	-	-	0	0	110

4.7. AUTORUN

Type	Funktionsbaustein
Input	IN : BOOL (Schalteingang)
	TEST : BOOL (aktiviert den Autorun Zyklus)
	ARE : BOOL (Enable Autorun)
Setup	TRUN : TIME (Mindestlaufzeit des Verbrauchers)
	TOFF : TIME (Maximale Standzeit des Verbrauchers)
I/O	ARX : BOOL (Autorun Enable Signal)
Output	OUT : BOOL (Ausgang für Verbraucher)
	ARO : BOOL (TRUE wenn Autorun aktiv)



AUTORUN überwacht die Laufzeit eines Verbrauchers und sorgt dafür, dass der Verbraucher an OUT nach Ablauf der Zeit TOFF mindestens für Die Zeit TRUN eingeschaltet wird. AUTORUN speichert die Laufzeit und schaltet den Ausgang erst dann ein wenn die Mindestlaufzeit TRUN innerhalb der Zeit TOFF unterschritten wird. Der Eingang IN ist der Schalteingang für den Ausgang OUT. Der Ausgang ARO signalisiert dass gerade Autorun aktiv ist. Der Eingang ARE muss TRUE sein um Autorun zu ermöglichen, an ARE kann ein Timer angeschlossen werden um Autorun zu bestimmten Zeiten zu starten. Der I/O ARX verhindert wenn TRUE einen Autorun, Autorun kann nur aktiv werden wenn ARI = FALSE. Wenn ARI = FLASE und die internen Timer abgelaufen sind schaltet der Baustein ARO und OUT auf TRUE und gleichzeitig setzt er ARI. Dieser Mechanismus kann auf verschiedene Weise genutzt werden:

a) Ein TRUE am I/O ARX kann verhindern das Autorun stattfindet, es kann z.B. von einem externen Timer gesteuert werden und so den Autorun nur während einer bestimmten Zeit erlauben.

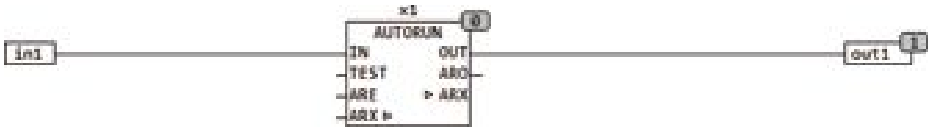
b) Die ARI Anschlüsse mehrerer Bausteine können zusammen geschaltet werden und somit wird verhindert das mehrere Bausteine gleichzeitig in den Autorun Modus schalten. Die Bausteine warten bis der erste Baustein mit Autorun fertig ist und dann wird der nächste Baustein beginnen. Dies

ist sehr sinnvoll um bei einer größeren Anzahl von Verbrauchern zu verhindern dass alle gleichzeitig den Autorun durchführen und somit unnötig hohe Strombelastung erzeugen.

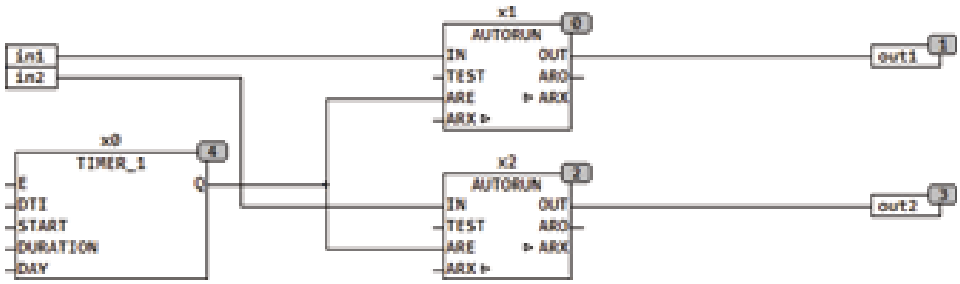
Die Betriebszustände von AUTORUN:

IN	TEST	ARE	ARX	ARO	OUT	
X	0	-	-	-	X	normaler Betrieb
-	1	-	1	1	1	TEST startet Autorun Zyklus
-	0	1	1	0 >> 1	1	Autorun Zyklus ist aktiv

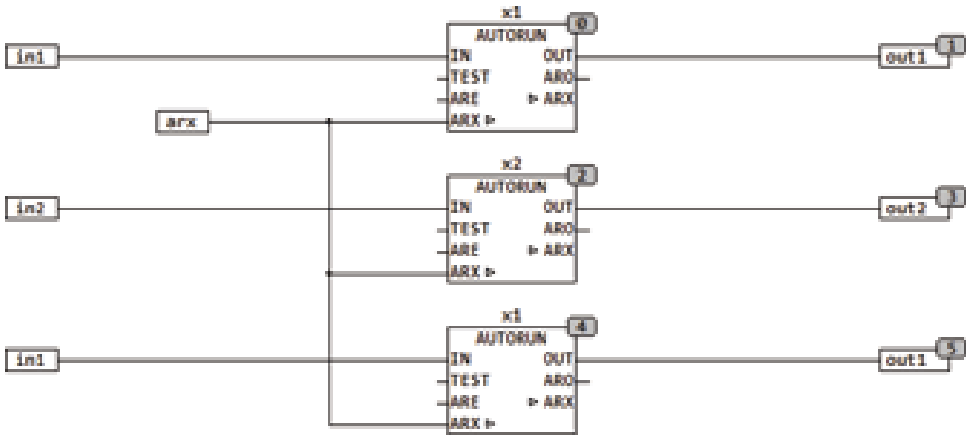
Eine simple Anwendung von Autorun mit Eingang und Ausgang:



Im nächsten Beispiel werden Die Eingänge ARE (Autorun Enable) durch einen Timer Freigegeben, so dass Autorun nur zu bestimmten Zeiten ausgeführt wird. Der Autorun der Bausteine X1 und X2 startet hierbei gleichzeitig.



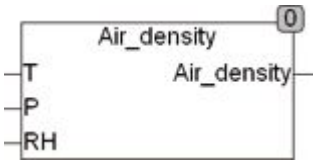
Das folgende Beispiel zeigt 3 Autorun Bausteine die über ARI gegenseitig verriegelt sind, so dass immer nur ein Baustein in den Autorun gehen kann und der andere entsprechend warten muss.


x1
AUTORUN
IN
TEST
ARE
ARX
OUT
ARO
ARX
0
in1
arx
out1
1
x2
AUTORUN
IN
TEST
ARE
ARX
OUT
ARO
ARX
2
in2
out2
3
x1
AUTORUN
IN
TEST
ARE
ARX
OUT
ARO
ARX
4
in1
out1
5

5. Heizung Lüftung und Klima

5.1. AIR_DENSITY

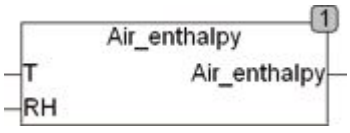
Type	Funktion : REAL
Input	T : REAL (Temperatur der Luft in °C)
	P : REAL (Luftdruck in Pascal)
	RH : REAL (Luftfeuchtigkeit in %)
Output	(Dichte der Luft in kg/m³)



AIR_DENSITY berechnet die Dichte der Luft in kg/m³ abhängig von Druck, Feuchte und Temperatur. Die Temperatur wird in °C, Druck in Pascal und die Feuchte in % (50 = 50%) angegeben.

5.2. AIR_ENTHALPY

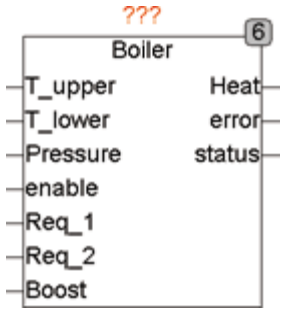
Type	Funktion : REAL
Input	T : REAL (Temperatur der Luft)
	RH : REAL (Relative Feuchte der Luft)
Output	(Enthalpie der Luft in J/g)



AIR_ENTHALPY berechnet die Enthalpie von Feuchter Luft aus den Angaben T für Temperatur in Grad Celsius und der relativen Feuchte RH in % (50 = 50%). Die Enthalpie wird in Joule / Gramm berechnet.

5.3. BOILER

Type	Funktionsbaustein
Input	T_UPPER : REAL (Eingang oberer Temperatursensor)
	T_LOWER : REAL (Eingang unterer Temperatursensor)
	PRESSURE : REAL (Eingang Drucksensor)
	ENABLE : BOOL (Warmwasseranforderung)
	REQ_1 : BOOL (Anforderungseingang für vordefinierte Temperatur 1)
	REQ_2 : BOOL (Anforderungseingang für vordefinierte Temperatur 2)
	BOOST : BOOL (Anforderungseingang für sofortige
	Bereitstellung)
Output	HEAT : BOOL (Ausgang für Ladekreis)
	ERROR : BOOL (Fehlersignal)
	STATUS : Byte (ESR kompatibler Status Ausgang)
Setup	T_UPPER_MIN : REAL (Mindesttemperatur für oben)
	Default = 50
	T_UPPER_MAX : REAL (Maximaltemperatur für oben)
	Default = 60
	T_LOWER_ENABLE : BOOL (FALSE, wenn unterer Temperatursensor nicht vorhanden ist)
	T_LOWER_MAX : REAL (Maximaltemperatur des unten)
	Default = 60
	T_REQUEST_1 : REAL (Temperatur bei Anforderung 1)
	Default = 70
	T_REQUEST_2 : REAL(Temperatur bei Anforderung 2)
	Default = 50
	T_REQUEST_HYS : REAL (Hysterese für Regelung) Default = 5
	T_PROTECT_HIGH : REAL (obere Grenztemperatur,
	Default = 80)
	T_PROTECT_LOW : REAL (untere Grenztemperatur,
	Default = 10)

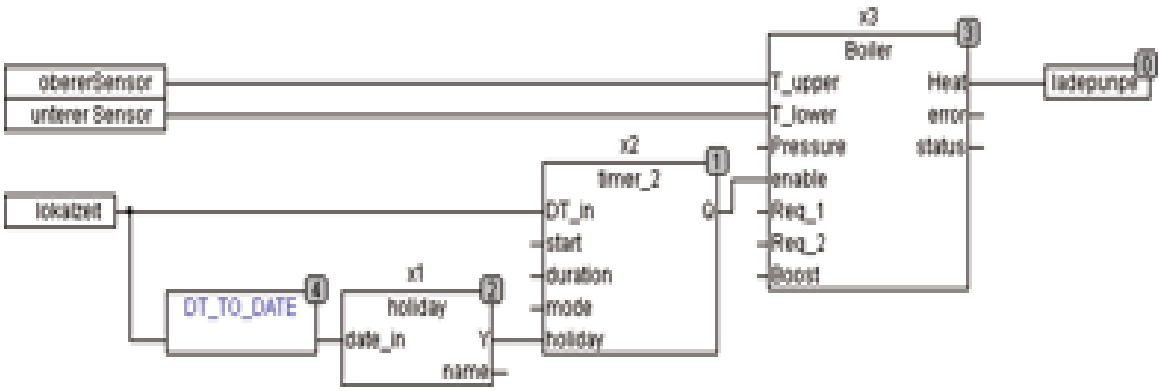


BOILER ist ein Controller für Pufferspeicher wie etwa Warmwasserspeicher. Durch 2 separate Temperatursensor-Eingänge können auch Schichtenspeicher geregelt werden. Mit der Setup-Variable T_LOWER_ENABLE kann der untere Temperatursensor aus- und eingeschaltet werden. Wenn der Eingang ENABLE auf TRUE ist, wird der Boiler aufgeheizt (HEAT = TRUE) bis die Vorgabetemperatur T_LOWER_MAX im unteren Bereich des Puffers erreicht ist und dann die Heizung ausgeschaltet, bis die untere Grenztemperatur des oberen Bereichs (T_UPPER_MIN) erreicht wird. Falls T_LOWER_ENABLE auf FALSE ist, wird der untere Sensor nicht ausgewertet und die Temperatur zwischen T_UPPER_MIN und T_UPPER_MAX im oberen Bereich geregelt. Ein PRESSURE-Eingang schützt den Boiler und verhindert die Ladung, wenn nicht genügend Wasserdruck im Boiler vorhanden ist. Falls ein Drucksensor nicht vorhanden ist, bleibt der Eingang unbeschaltet. Als weitere Schutzfunktion stehen die Vorgabewerte T_PROTECT_LOW (Frostschutz) und T_PROTECT_HIGH zur Verfügung und verhindern das die Temperatur im Puffer einen oberen Grenzwert nicht übersteigt und ein unterer Grenzwert nicht unterschritten wird. Bei Auftreten eines Fehlers wird der Ausgang ERROR auf TRUE gesetzt und gleichzeitig ein Statusbyte am Ausgang Status gemeldet, welches durch Bausteine wie ESR_COLLECT weiter ausgewertet werden kann. Durch eine steigende Flanke am Eingang BOOST wird die Puffertemperatur unmittelbar auf T_UPPER_MAX (T_LOWER_ENABLE = FALSE) beziehungsweise T_LOWER_MAX (T_LOWER_ENABLE = TRUE) aufgeheizt. BOOST kann zur außerplanmäßigen Aufheizung des Boilers, wenn ENABLE auf FALSE ist, benutzt werden. Die Aufheizung durch BOOST ist flankengetriggert und führt bei jeder steigenden Flanke an BOOST zu genau einem Aufheizvorgang. Durch eine steigende Flanke an BOOST während ENABLE TRUE ist wird die Heizung sofort gestartet bis die maximale Temperatur erreicht ist. Der Boiler wird also nachgeladen, um maximale Wärmekapazität bereitzustellen. Die Eingänge REQ_1 und REQ_2 dienen dazu, jederzeit eine vordefinierte Temperatur (T_REQUEST_1 oder T_REQUEST_2) bereitzustellen. REQ kann zum Beispiel zur Bereitstellung einer höheren Temperatur zur Legionellendesinfektion oder auch zu anderen Zwecken verwendet werden. Die Bereitstellung der Request-Temperaturen erfolgt durch Messung am oberen Temperatursensor und mit einer 2-Punkt Regelung deren Hysterese durch T_REQUEST_HYS voreingestellt wird.

Status	
1	oberer Temperatursensor hat die obere Grenztemperatur überschritten

2	oberer Temperatursensor hat die untere Grenztemperatur unterschritten
3	unterer Temperatursensor hat die obere Grenztemperatur überschritten
4	unterer Temperatursensor hat die untere Grenztemperatur unterschritten
5	Wasserdruck im Puffer ist zu gering
100	Standby
101	BOOST Nachladung
102	Standard Nachladung
103	Nachladung auf Request Temperatur 1
104	Nachladung auf Request Temperatur 2

Das folgende Beispiel zeigt die Anwendung von BOILER mit einem TIMER und einer Feiertagsschaltung:

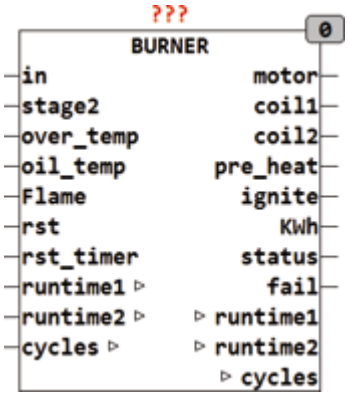


5.4. BURNER

Type Funktionsbaustein

Input IN : BOOL (Steuereingang)
STAGE2 : BOOL (Steuereingang Stufe 2)
OVER_TEMP : BOOL (Temperaturbegrenzung des Kessels)
OIL_TEMP : BOOL (Thermostat der Ölvorerwärmung)
FLAME : BOOL (Flammwächter)
RST : BOOL (Reset-Eingang Für Störrücksetzung)
RST_TIMER : BOOL (Reset für die Betriebszähler)

Output	MOTOR : BOOL (Steuersignal für den Motor)
	COIL1 : BOOL (Steuersignal für Ölventil Stufe 1)
	COIL2 : BOOL (Steuereingang für Ölventil Stufe 2)
	PRE_HEAT : BOOL (Ölvorerwärmung)
	IGNITE : BOOL (Zündung)
	KWH : REAL (Kilowattstundenzähler)
	STATUS : Byte (ESR Kompatibler Statusausgang)
	FAIL : BOOL (Störmeldung: TRUE, wenn Fehler Auftritt)
I/O	RUNTIME1 : UDINT (Betriebszeit Stufe 1)
	RUNTIME2 : UDINT (Betriebszeit Stufe 2)
	CYCLES : UDINT (Anzahl der Brenner Starts)
Setup	PRE_HEAT_TIME : TIME (maximale Zeit für die Ölvorerwärmung)
	PRE_VENT_TIME : TIME (Vorbelüftungszeit)
	PRE_IGNITE_TIME : TIME (Vorzündungszeit)
	POST_IGNITE_TIME : TIME (Nachzündungszeit)
	STAGE2_DELAY : TIME (Verzögerung Stufe 2)
	SAFETY_TIME : TIME ()
	LOCKOUT_TIME : TIME (Zeit die vergehen muss, bevor mit einem RST eine Störung gelöscht werden kann)
	MULTIPLE_IGNITION : BOOL ()
	KW1 : REAL (Leistung des Brenners auf Stufe 1 in KW)
	KW2 : REAL (Leistung des Brenners auf Stufe 2 in KW)

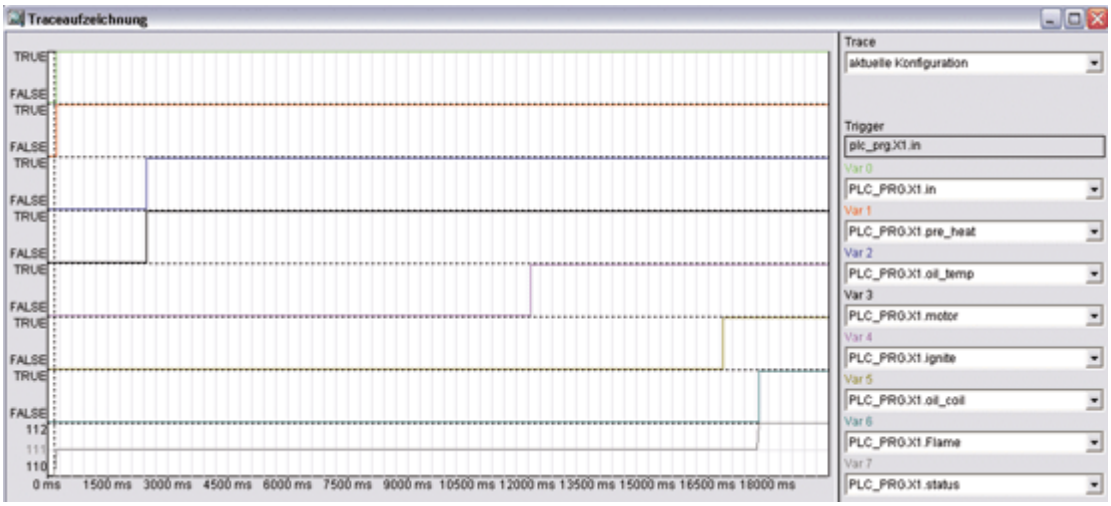


BURNER ist ein Steuerinterface für Öl- oder Gasbrenner mit Betriebszähler und Kilowattstundenzähler. Der Baustein steuert einen zweistufigen Bren-

ner mit optionaler Ölvorerwärmung. Der Eingang IN ist der Steuereingang, der den Brenner nur dann startet, wenn der Eingang OVER_TEMP FALSE ist. OVER_TEMP ist der Kesselschutzthermostat, der TRUE wird, wenn die Kesseltemperatur die maximal zulässige Temperatur erreicht hat. Ein Brennerstart beginnt mit der Ölvorerwärmung, indem PRE_HEAT TRUE wird. Dann wird auf ein Signal am Eingang OIL_TEMP gewartet. Falls innerhalb der PRE_HEAT_TIME das Signal OIL_TEMP nicht TRUE wird und die Öltemperatur nicht erreicht wird, wird die Startsequenz unterbrochen und der Ausgang Störung gesetzt. Gleichzeitig wird am Ausgang Status Der Fehler 1 ausgegeben. Nach der Ölvorerwärmung wird der Motor eingeschaltet und damit der Ventilator in Betrieb gesetzt. Anschließend wird nach definierter Zeit die Zündung eingeschaltet und danach das Ölventil geöffnet. Sollte dann nach spezifizierter Zeit (SAFETY_TIME) der Flammwächter nicht ansprechen, so geht der Baustein auf Störung. Eine Störung wird auch dann signalisiert, wenn der Flammwächter bereits vor der Zündung anspricht. Falls nach erfolgreicher Zündung die Flamme abreißt und die Setup-Variable MULTIPLE_IGNITION = TRUE steht, wird sofort wieder gezündet. Eine zweite Stufe wird nach Ablauf der STAGE2_DELAY Zeit automatisch zugeschaltet wenn der Eingang STAGE2 TRUE ist.

Tritt eine Störung auf, so wird der Baustein für eine feste Zeit LOCKOUT_TIME blockiert und erst danach kann ein RST den Betrieb wieder starten. Während der LOCKOUT_TIME muss der RST-Eingang FALSE sein. Ein TRUE am Eingang OVER_TEMP stoppt sofort jede Aktion und meldet den Fehler 9.

Der Status-Ausgang signalisiert den momentanen Zustand des Bausteins:



110 = Warten auf Startsignal (Standby)
111 = Startsequenz wird durchlaufen
112 = Brenner Läuft auf Stufe 1
113 = Brenner läuft auf Stufe 2

Eine Reihe von Fehlerzuständen werden am Ausgang STATUS bereitgestellt, wenn ein Fehler Auftritt:

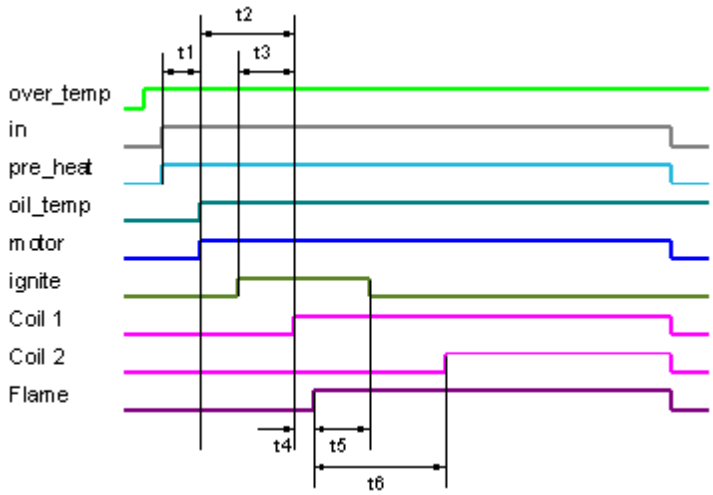
1 = Ölvorerwärmung hat innerhalb der PRE_HEAT_TIME nicht angesprochen
2 = Flammwächter ist aktiv während der Ölvorerwärmung (PRE_HEAT_TIME)
3 = Flammwächter ist aktiv während der Belüftungszeit (PRE_VENTILATION_TIME)
4 = Sicherheitszeit (Safety_TIME) wurde ohne Flamme überschritten
5 = Flamme ist im Betrieb Abgerissen
9 = Kessel Übertemperatur Kontakt hat Ausgelöst

Traceaufzeichnung einer Normalen Startsequenz:

Das Signal IN startet die Sequenz mit dem Ausgang PRE_HEAT. Nach erreichen der Öltemperatur (OIL_TEMP = TRUE) wird der Motor gestartet und die PRE_VENTILATION_TIME (Zeit von Motor Start bis Ölventil offen ist) abgewartet. Nach einer einstellbaren Zeit (PPR_IGNITION_TIME) vor dem Öffnen des Ölventils wird die Zündung eingeschaltet. Die Zündung bleibt dann solange ein, bis die POST_IGNITION_TIME abgelaufen ist. Die Betriebszeit wird je Stufe unabhängig in Sekunden gemessen.

In	over tem	Oil tem	Flame	Rst	motor	Oil coil	Pre hea	ignite	Status	fail	
0	0	-	-	0	0	0	0	0	110	0	Wartezustand
1	0	0	0	0	0	0	1	0	111	0	Ölvorwärmphase
1	0	1	0	0	1	0	1	0	111	0	Vorbelüftungsphase
1	0	1	0	0	1	0	1	1	111	0	Vorzündphase
1	0	1	0	0	1	1	1	1	111	0	Ventil Stufe 1 öffnen
1	0	1	1	0	1	1	1	1	112	0	Flamme brennt Nachzündphase
1	0	1	1	0	1	1	1	0	112	0	Brenner läuft
1	0	1	0	0	1	1	1	1	111	0	Nachzündung nach Flammabriß
-	1	-	-	-	-	-	-	-	9	1	Kessel Übertemperatur
1	0	1	1	0	1	0	1	0	3	1	Fremdlichtfehler

Das folgende Zeitdiagramm erläutert die verschiedenen Setup-Zeiten und den Ablauf:



Das Zeitdiagramm gibt den genauen Zeitverlauf wieder:

t1 = Vorheizzeit (PRE_HEAT_TIME)

t2 = Vorbelüftungszeit (PRE_VENT_TIME)

t3 = Vorzündungszeit (PRE_IGNITE_TIME)

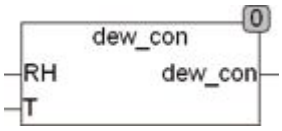
t4 = Sicherheitszeit (SAFETY_TIME)

t5 = Nachzündungszeit (POST_IGNITE_TIME)

t6 = Verzögerung für Stufe 2 (STAGE2_DELAY)

5.5. DEW_CON

Type	Funktion : REAL
Input	RH : REAL (Relative Feuchte)
	T : REAL (Temperatur in °C)
Output	REAL (Wasserdampf Konzentration in Gramm / m³)

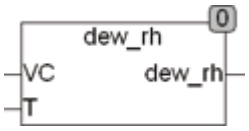


Der Baustein DEW_CON berechnet aus der Relativen Feuchte (RH) und der Temperatur (T in °C) die Wasserdampfkonzentration in der Luft. Das Ergebnis wird in Gramm / m³ ermittelt. RH ist in % (50 = 50%) anzugeben und die Temperatur in °C.

Die Baustein ist für Temperaturen von -40°C bis +90°C geeignet.

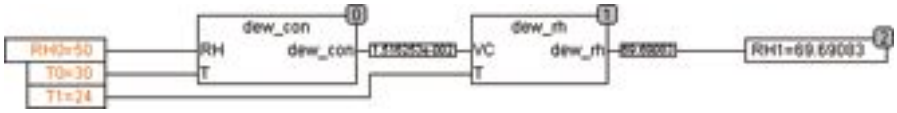
5.6. DEW_RH

Type	Funktion : REAL
Input	VC : REAL (Wasserdampfkonzentration in Luft in Gramm / m³)
	T : REAL (Temperatur in °C)
Output	REAL (Relative Luftfeuchtigkeit in %)



Der Baustein DEW_RH berechnet aus der Wasserdampfkonzentration (VC) und der Temperatur (T in °C) die relative Luftfeuchtigkeit in % (50 = 50%). Die Wasserdampfkonzentration wird in Gramm/m³ angegeben. DEW_CON kann für Berechnungen in beide Richtungen (aufheizen und abkühlen) verwendet werden. Wird zu stark abgekühlt, so ist die maximale relative Feuchte auf 100% begrenzt. Für Berechnungen des Taupunktes wird der Baustein DEW_TEMP empfohlen.

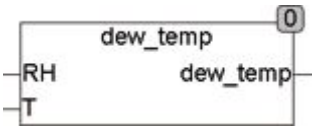
Im folgenden Beispiel wird der Fall berechnet, wenn Luft von 30°C und relativer Feuchte von 50% um 6 Grad abgekühlt wird. Der Baustein DEW_CON liefert die Feuchtigkeitskonzentration in der Ausgangsluft von 30° und DEW_RH berechnet die resultierende relative Luftfeuchtigkeit RH von 69,7%. Diese Berechnungen sind wichtig, wenn Luft abgekühlt oder aufgeheizt wird. In Klimaanlagen ist eine resultierende relative Feuchte von 100% wegen Taubildung und den daraus resultierenden Problemen zu vermeiden.



Siehe hierzu auch die Bausteine DEW_CON und DEW_TEMP.

5.7. DEW_TEMP

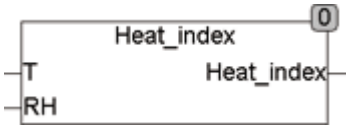
Type	Funktion : REAL
Input	RH : REAL (Relative Feuchte)
	T : REAL (Temperatur in °C)
Output	REAL (Taupunkttemperatur)



Der Baustein DEW_TEMP berechnet aus der Relativen Feuchte (RH) und der Temperatur (T in °C) die Taupunkttemperatur. Die Relative Feuchte wird in % angegeben (50 = 50%).

5.8. HEAT_INDEX

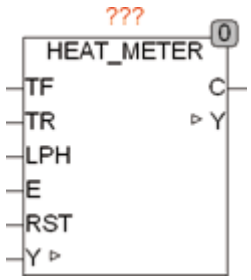
Type	Funktion : REAL
Input	T : REAL (Temperatur in °C)
	RH : REAL (Relative Feuchte)
Output	REAL (Heat Index Temperatur)



HEAT_INDEX berechnet die bei hohen Temperaturen und hoher Feuchte gefühlte Temperatur. Die Funktion ist definiert für Temperaturen größer 20 °C und relativer Feuchte > 10%. Für Werte außerhalb des Definitionsbereichs wird die Eingangstemperatur ausgegeben.

5.9. HEAT_METER

Type	Funktion : REAL
Input	TF : REAL (Vorlauftemperatur in °C)
	TR : REAL (Rücklauftemperatur in °C)
	LPH : REAL (Durchflussmenge in L/h bzw. L/Impuls)
	E : BOOL (Enable Signal)
	RST : BOOL (asynchroner Reset Eingang)
Setup	CP : REAL (Spezifische Wärmekapazität der 2ten Komponente)
	DENSITY : REAL (Dichte der 2ten Komponente)
	CONTENT : REAL (Anteil, 1 = 100%)
	PULSE_MODE : BOOL (Impulszähler wenn TRUE)
	RETURN_METER : BOOL (Durchflussmesser im Rücklauf wenn TRUE)
	AVG_TIME : TIME (Zeitintervall für Momentanen Verbrauch)
Output	C : REAL (aktueller Verbrauch in Joule / Stunde)
I/O	Y : REAL (Wärmemenge in Joule)



HEAT_METER ist ein Wärmemengenzähler. Die Wärmemenge Y wird in Joule gemessen. Die Eingänge TF und TR sind die Vorlauf und Rücklauftemperatur des Mediums. Am Eingang LPH wird die Durchflussmenge in Liter / Stunde beziehungsweise die Durchflussmenge je Impuls an E spezifiziert. Die Eigenschaft von E wird durch die Setup Variable PULSE_MODE bestimmt. PULSE_MODE = FALSE bedeutet das die Wärmemenge kontinuierlich aufaddiert wird solange E auf TRUE ist. PULSE_MODE = TRUE bedeutet das die Wärmemenge mit jeder steigenden Flanke von E aufaddiert wird. Der PULSE_MODE ist bei Verwendung von Wärmezählern einzuschalten, während am Eingang LPH die Flussmenge in Litern je Impuls anzugeben ist und am Eingang E wird der Wärmezähler angeschlossen. Wenn kein Flussmengenmesser Vorhanden ist, so wird am Eingang E das Pumpensignal angeschlossen und am Eingang LPH die Pumpenleistung in Litern / Stunde angegeben. Bei Verwendung eines Flussmengenmessers mit

Analogem Ausgang wird der Ausgang entsprechend auf Liter / Stunde umgerechnet und auf den Eingang LPH gelegt, Der Eingang E wird hierbei auf TRUE gesetzt. Mit den Setup- Variablen CP, DENSITY und CONTENT wird die 2te Komponente des Mediums spezifiziert. Für den Betrieb mit reinem Wasser sind keinerlei Angaben von CP, DENSITY und CONTENT nötig. Wenn ein Gemisch aus Wasser und einem 2ten Medium vorhanden ist werden mit CP die Spezifische Wärmekapazität in J/KgK, mit DENSITY die DICHTE in KG/l und mit CONTENT der Anteil der 2ten Komponente spezifiziert. Ein Anteil von 0.5 bedeutet 50% und 1 wäre entsprechend 100%. Mit der Setup Variablen RETURN_METER wird angegeben ob der Durchflussmesser im Vorlauf oder Rücklauf sitzt. RETRUN_METER = TRUE steht für Rücklaufmessung und FALSE für Vorlaufmessung. Am Ausgang C stellt der Baustein den momentanen Verbrauch zur Verfügung. Der momentane Verbrauch wird in Joule / Stunde angegeben, und in den Zeitabständen von AVG_TIME ermittelt.

Der Baustein hat folgende Vorgabewerte, die aktiv sind wenn die entsprechenden Werte vom Anwender nicht gesetzt werden:

PULSE_MODE = FALSE

RETURN_METER = FALSE

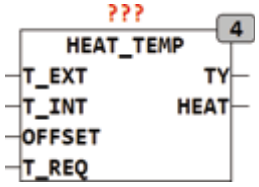
AVG_TIME = T#5s

5.10. HEAT_TEMP

Type	Funktionsbaustein
Input	T_EXT : REAL (Außentemperatur)
	T_INT : REAL (Soll Raumtemperatur)
	OFFSET : REAL (Absenkung oder Anhebung der Raumtemperatur)
	T_REQ : REAL (Temperaturanforderung)
Output	TY : REAL (Heizkreisvorlauftemperatur)
	HEAT : BOOL (Heizungsanforderung)
Setup	TY_MAX : REAL (maximale Heizkreistemperatur, 70°C)
	TY_MIN : REAL (Minimale Heizkreistemperatur, 25°C)
	TY_C : REAL (Auslegungstemperatur, 70°C)
	T_INT_C : REAL (Auslegungstemperatur Raum, 20°C)
	T_EXT_C : REAL (T_EXT bei Auslegungstemperatur -15°C)
	T_DIFF_C : REAL (Vor- Rücklaufdifferenz 10°C)

C : REAL (Konstante des Heizsystems, DEFAULT = 1,33)

H : REAL (Schwelle für Heizungsanforderung 3°C)



HEAT_TEMP berechnet die Vorlauftemperatur aus der Außentemperatur nach folgender Formel:

TY = TR + T_DIFF / 2 * TX + (TY_Setup - T_DIFF / 2 - TR) * TX ^ (1 / C)

mit: TR = T_INT + OFFSET

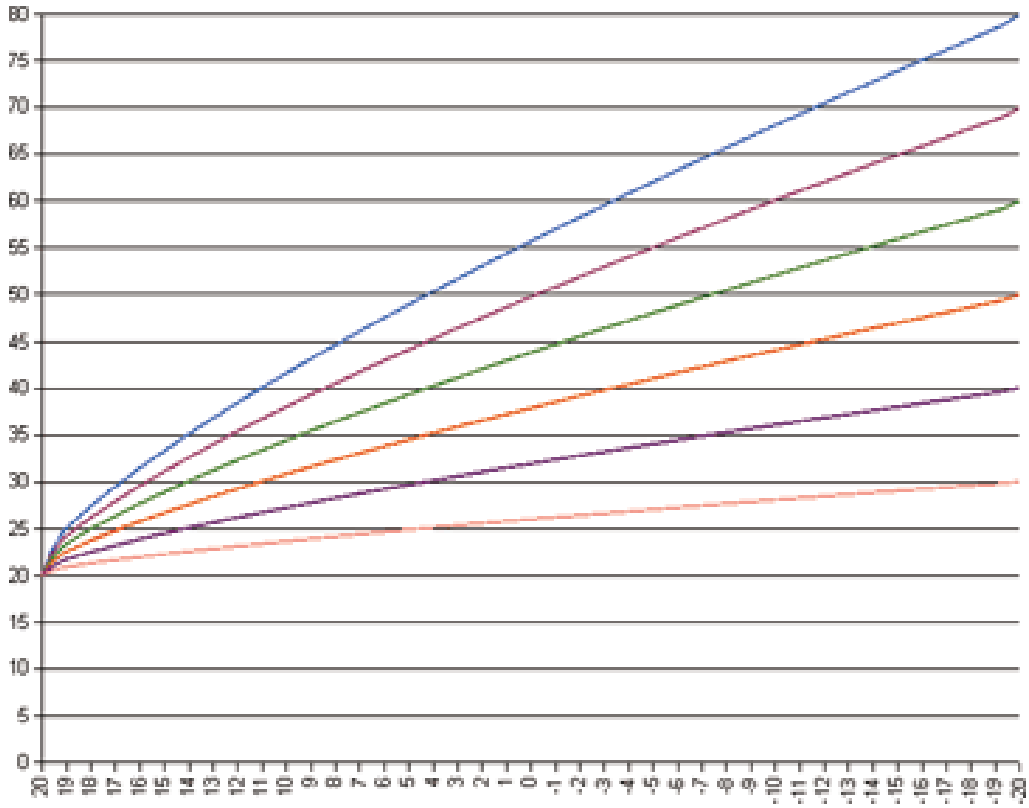
TX := (TR - T_EXT) / (T_INT_Setup - T_EXT_Setup);

Die Parameter der Heizkurve werden durch die Setup Variablen TY_C (Auslegungsvorlauftemperatur), T_INT_C (Raumtemperatur im Auslegungspunkt), T_EXT_C (Außentemperatur im Auslegungspunkt) und T_DIFF_C (Differenz Vor- Rücklauf im Auslegungspunkt) vorgegeben. Mit dem Eingang Offset kann die Heizkurve an Raumabsenkung (negativer Offset) oder Raumanhebung (positiver Offset) angepasst werden. Mit den Setup Variablen TY_MIN und TY_MAX kann die Vorlauftemperatur auf einen Minimal- und Maximalwert begrenzt werden. Der Eingang T_REQ dient dazu, externe Temperaturanforderungen wie z.B. vom Boiler zu unterstützen. Ist T_REQ größer als der aus der Heizkurve berechnete Wert für TY, so wird TY auf T_REQ gesetzt. Die Begrenzung auf TY_MAX gilt nicht für die Anforderung durch T_REQ. Durch die Setup Variable H wird festgelegt ab welcher Außentemperatur die Heizkurve berechnet wird, solange T_EXT + H >= als T_INT + OFFSET ist bleibt TY auf 0 und HEAT ist FALSE. Wird T_EXT + H < als T_INT + OFFSET wird HEAT TRUE und TY gibt die berechnete Vorlauftemperatur aus. Die Setup Variable C legt die Krümmung der Heizkurve fest. Die Krümmung ist abhängig vom verwendeten Heizsystem.

Konvektoren:	C = 1.25 – 1.45
Plattenheizkörper:	C = 1.20 – 1.30
Radiatoren:	C = 1.30
Rohre:	C = 1.25
Fußbodenheizung:	C = 1.1

Je größer der Wert von C, desto stärker ist die Heizkurve gekrümmt. Ein Wert von 1.0 ergibt eine Gerade als Heizkurve. Typische Heizsysteme liegen zwischen 1.0 und 1.5.

Die Grafik zeigt Heizkurven für Auslegungstemperaturen von 30 – 80 °C Vorlauftemperatur bei -20°C Außentemperatur und bei einem C von 1.33:



5.11. LEGIONELLA

Type	Funktionsbaustein
Input	MANUAL : BOOL (Manual Start Input)
	TEMP_BOILER : REAL (Boiler Temperatur)
	TEMP_RETURN : REAL (Temperatur der Zirkulationsleitung)
	DT_IN : DATE_TIME (Momentane Tageszeit und Datum)
	RST : BOOL (Asynchroner Reset)
Output	HEAT : BOOL (Steuersignal für Warmwasserheizung)
	PUMP : BOOL (Steuersignal für Zirkulationspumpe)
	STATUS : Byte (ESR kompatibler Statusausgang)
	VALVE0..7 : BOOL (Steuerausgänge für Ventile der Zirkulation)
	RUN : BOOL (TRUE wenn Sequenz läuft)
Setup	T_START : TOD (Tageszeit zu der die Desinfizierung startet)
	DAY : INT (Wochentag an dem die Desinfizierung startet)

TEMP_SET : REAL (Temperatur des Boilers)

TEMP_OFFSET : REAL ()

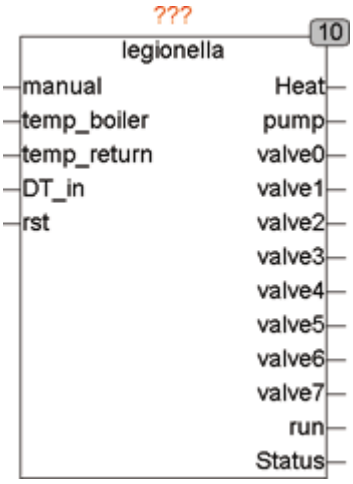
TEMP_HYS : REAL ()

T_MAX_HEAT : TIME (maximale Zeit zum Aufheizen des Boilers)

T_MAX_RETURN : TIME (maximale Zeit, bis der Eingang

TEMP_RETURN nach VALVE aktiv wird)

TP_0 .. 7 : TIME (Desinfektionszeit für Kreise 0..7)



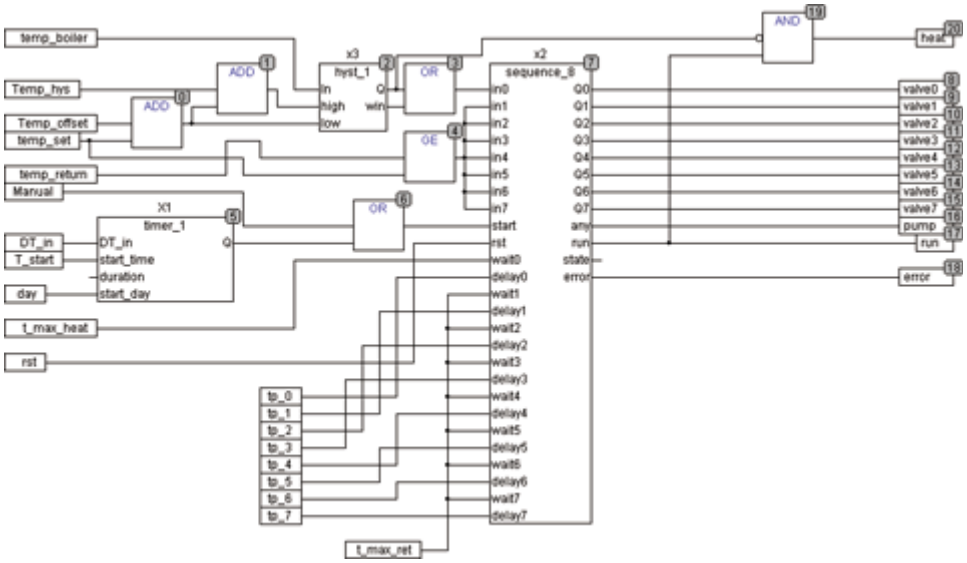
LEGIONELLA hat eine integrierte Schaltuhr, die an einem bestimmten Wochentag (DAY) zu einer bestimmten Tageszeit (T_START) die Desinfektion startet. Hierzu ist die externe Anschaltung der Lokalzeit nötig (DT_IN). Jederzeit kann mit einer steigenden Flanke an MANUAL die Desinfektion auch von Hand gestartet werden.

Der Ablauf eines Desinfektionszyklus wird mit einem internen Start aufgrund von DT_IN, DAY und T_START, oder durch eine steigende Flanke an MANUAL gestartet. Der Ausgang HEAT wird TRUE und steuert die Heizung des Boilers an. Innerhalb der Aufheizzeit T_MAX_HEAT muss dann das Eingangssignal TEMP_BOILER auf TRUE gehen. Wird die Temperatur nicht innerhalb von T_MAX_HEAT gemeldet, geht der Ausgang Status auf Störung. Die Desinfektion läuft aber trotzdem weiter. Nach der Aufheizphase wird die Boilertemperatur gemessen und falls nötig durch TRUE am Ausgang HEAT wieder nachgeheizt. Sobald die Boilertemperatur erreicht ist, wird PUMP TRUE und die Zirkulationspumpe eingeschaltet. Dann werden nacheinander die einzelnen Ventile geöffnet und gemessen, ob innerhalb der Zeit T_MAX_RETURN die Temperatur am Rücklauf der Zirkulationsleitung erreicht wurde. Falls ein Rückflussthermometer nicht vorhanden ist, kann der Eingang T_MAX_RETURN einfach offen bleiben.

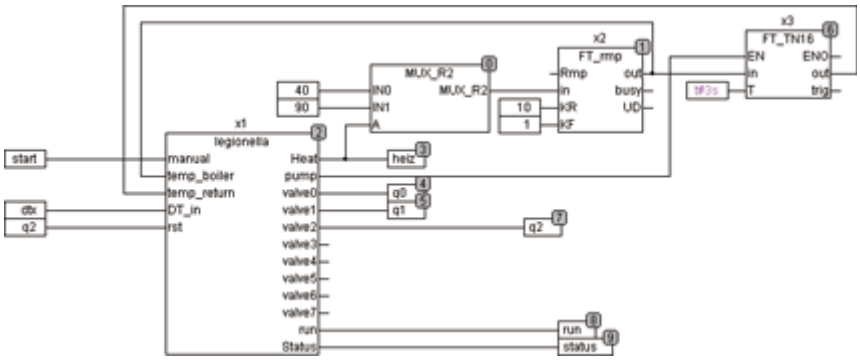
Der Ausgang Status ist ESR kompatibel und kann folgende Meldungen abgeben:

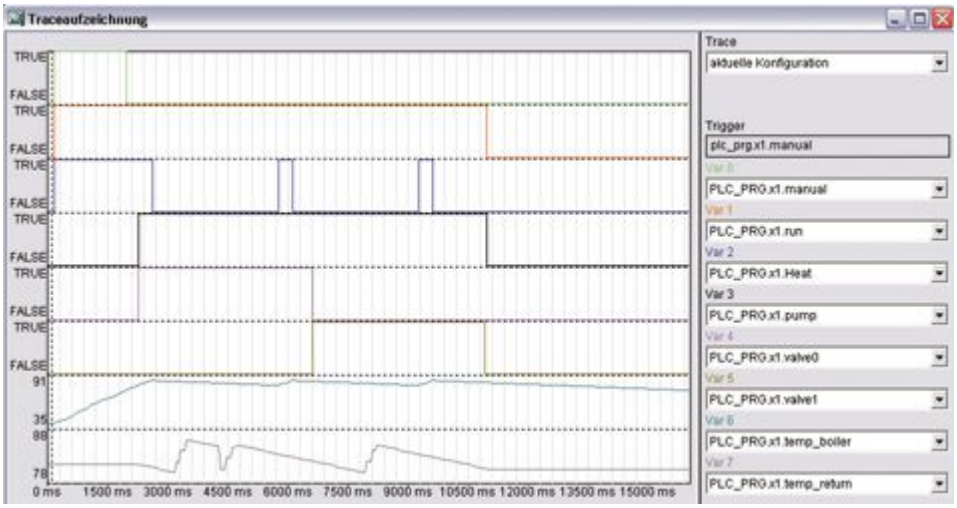
110	Wartestellung
111	Sequenz läuft
1	Boiler Temperatur wurde nicht erreicht
2	Rücklauftemperatur bei Ventil0 wurde nicht erreicht
3..8	Rücklauftemperatur bei Ventil1..7 wurde nicht erreicht

Schematischer interner Aufbau von LEGIONELLA:



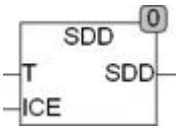
Das Folgende Beispiel zeigt eine Simulation für 2 Desinfektionskreise mit Traceaufzeichnung. In diesem Aufbau ist VALVE2 auf den Eingang RST geschaltet und unterbricht damit die Sequenz nach 2 Kreisen:





5.12. SDD

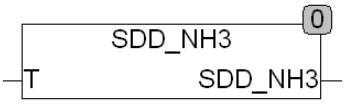
Type	Funktion : REAL
Input	T : REAL (Temperatur der Luft in °C)
	ICE : BOOL (TRUE für Luft über Eis und FALSE für Luft über Wasser)
Output	REAL (Sättigungsdampfdruck in Pa)



SDD berechnet den Sättigungsdampfdruck für Wasserdampf in Luft. Die Temperatur T wird in °C angegeben. Das Ergebnis kann sowohl für Luft über Eis (ICE = TRUE) und für Luft über Wasser (ICE = FALSE) berechnet werden. Der Gültigkeitsbereich der Funktion liegt bei -30°C bis 70°C über Wasser und bei -60°C bis 0°C über Eis. Die Berechnung wird nach der Magnusformel ausgeführt.

5.13. SDD_NH3

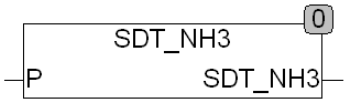
Type	Funktion : REAL
Input	T : REAL (Temperatur in °C)
Output	REAL (Sättigungsdampfdruck in Pa)



SDD_NH3 berechnet den Sättigungsdampfdruck für Ammoniak (NH3). Die Temperatur T wird in °C angegeben. Der Gültigkeitsbereich der Funktion liegt bei -109°C bis 98°C.

5.14. SDT_NH3

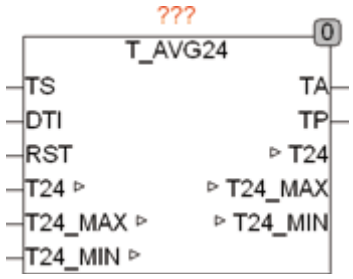
Type	Funktion : REAL
Input	T : REAL (Temperatur in °C)
Output	REAL (Sättigungsdampfdruck in Pa)



SDT_NH3 berechnet die Sättigungsdampftemperatur für Ammoniak (NH3). Der Druck P wird in °C angegeben. Der Gültigkeitsbereich der Funktion liegt bei 0.001 bar bis 60 bar.

5.15. T_AVG24

Type	Funktionsbaustein
Input	TS : INT (Außentemperatur Sensor)
	DTI : DT (Datum und Tageszeit)
	RST : BOOL (Reset)
Setup	T_FILTER : TIME (T des Eingangsfilters)
	SCALE : REAL := 1.0 (Skalierungsfaktor)
	OFS : REAL (Nullpunktabgleich)
Output	TA : REAL (Momentane Außentemperatur)
	TP : BOOL (TRUE wenn T24 erneuert wird
I/O	T24 : REAL (Tagesmitteltemperatur)
	T24_MAX : REAL (Maximaltemp. in den letzten 24 Stunden)
	T24_MIN : REAL (Minimaltemperatur in den letzten 24 Stunden)



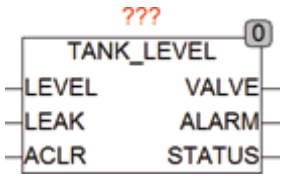
T_AVG24 ermittelt die Tagesmitteltemperatur T24. Der Sensor Eingang TS ist vom Typ INT und stellt die Temperatur * 10 dar (ein Wert von 234 bedeutet 23,4 °C). Die Daten von Filter laufen zur Unterdrückung von Störungen über ein Tiefpassfilter mit der Zeit T_FILTER. Mittels SCALE und OFS können Nullpunktfehler und Skalierung des Sensors angepasst werden. Der Ausgang TA gibt die aktuelle Außentemperatur, welche jede halbe und volle Stunde gemessen wird, an. Der Baustein schreibt alle 30 Minuten den über die letzten 48 Werte ermittelten Tagesmittelwert in die I/O Variable T24, die extern definiert werden muss und dadurch auch REMANENT oder PERSISTENT definiert werden kann. Wird beim ersten Start ein Wert von -1000 in T24 vorgefunden, so initialisiert sich der Baustein beim ersten Aufruf mit dem aktuellen Sensorwert, so dass danach alle 30 Minuten ein gültiger Mittelwert ausgegeben werden kann. Hat T24 einen beliebigen anderen Wert als -1000, so initialisiert sich der Baustein mit diesem Wert und Rechnet den Mittelwert basierend aus diesem Wert weiter. Dies ermöglicht bei Stromausfall und remanenter Speicherung von T24 ein sofortiges Weiterarbeiten nach dem wieder Einschalten. Ein Reset Eingang kann jederzeit einen Neustart des Bausteins erzwingen, wobei abhängig vom Wert in T24 der Baustein entweder mit TS oder dem alten Wert von T24 initialisiert wird. Möchte man den Baustein auf einen bestimmten Mittelwert setzen, so wird der gewünschte Wert in T24 geschrieben und dann ein Reset erzeugt.

T24_MAX und T24_MIN geben den Maximal- und Minimal-Wert der letzten 24 Stunden aus. Zur Ermittlung des Maximal und Minimal Wertes werden die Temperaturen zur jeweils halben Stunde berücksichtigt. Eine Temperaturwert der zwischen 2 Messungen auftritt wird hierbei nicht berücksichtigt.

5.16. TANK_LEVEL

Type	Funktionsbaustein
Input	LEVEL : BOOL (Eingang für Niveau Sensor)
	LEAK : BOOL (Eingang für Leck Sensoren)
	ACLR : BOOL (Eingang zum Rücksetzen des Alarms)
Setup	MAX_VALVE_TIME (Maximale Nachspeisezeit für Ventil)

	LEVEL_DELAY_TIME : TIME (Ansprechzeit für LEVEL Eingang)
Output	VALVE : BOOL (Ausgangssignal zum Ventil)
	ALARM : BOOL (Alarmausgang)
	STATUS : BYTE (ESR kompatibler Status Ausgang)



TANK_LEVEL dient dazu den Flüssigkeitsstand in einem Tank konstant zu halten. Am Eingang LEVEL wird der Niveausensor angeschlossen und am Ausgang VALVE das Nachspeiseventil. Um bei unruhigen Oberflächen den Niveausensor zu entprellen kann dessen Ansprechzeit mittels der Setup Variable LEVEL_DELAY_TIME entsprechend eingestellt werden. Am Eingang LEVEL wird mit TRUE angezeigt das der Flüssigkeitsstand zu niedrig ist, nachdem der Eingang durchgehend für Die Zeit LEVEL_DELAY_TIME auf TRUE war wird der Ausgang VALVE auf TRUE gesetzt um Flüssigkeit nachzuspeisen. Während des Nachspeisevorgangs wird MAX_VALVE_TIME überwacht und falls VALVE länger als diese Zeit auf TRUE bleibt wird ein ALARM generiert um bei Sensorfehlern oder Lecks ein Dauerndes Nachspeisen zu verhindern. Der Baustein überwacht zusätzlich den Eingang LEAK welcher für normalen Betrieb immer FALSE sein muss. Sobald LEAK auf TRUE geht wird sofort die Nachspeisung unterbrochen und ein Alarm generiert. LEAK dient zum Anschluß von Lecksensoren und oder zusätzlichen Niveausensoren oberhalb des normalen Niveaus. Tritt im Betrieb ein Alarm auf so stoppt de Baustein jegliche Nachspeisung bis der Fehler beseitigt wurde und der Eingang ACLR kurz auf TRUE getastet wird. Am ESR kompatiblen Statusausgang werden alle Betriebszustände mittels ESR Meldungen ausgegeben.

STATUS = 1, Lecksensor (LEAK) ist aktiviert.

STATUS = 2, Nachspeisezeit (MAX_VALVE_TIME) wurde überschritten.

STATUS = 100, Niveau ist erreicht, Nachspeisung abgeschaltet.

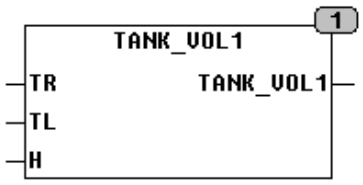
STATUS = 101, ACLR wurde betätigt.

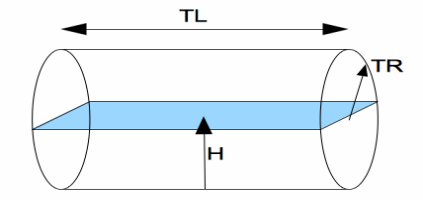
STATUS = 102, Niveau unterschritten, Nachspeisung läuft.

5.17. TANK_VOL1

Type	Funktion : REAL
Input	TR : REAL (Radius des Tanks)

	TL : REAL	(Länge des Tanks)
	H : REAL	(Füllhöhe des Tanks)
Output	Real	(Inhalt des Tanks bis zur Füllhöhe)

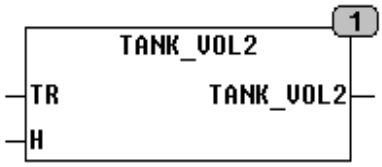


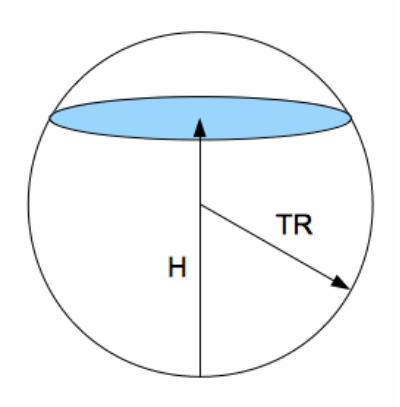


TANK_VOL1 berechnet den Inhalt eines Rohrförmigen Tanks der bis zur Höhe H gefüllt ist.

5.18. TANK_VOL2

Type	Funktion : REAL	
Input	TR : REAL	(Radius des Tanks)
	H : REAL	(Füllhöhe des Tanks)
Output	Real	(Inhalt des Tanks bis zur Füllhöhe)



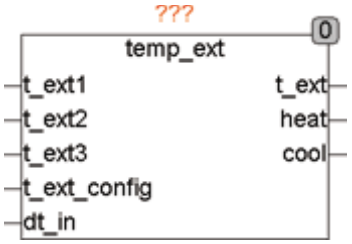


TANK_VOL2 berechnet den Inhalt eines Kugelförmigen Tanks der bis zur Höhe H gefüllt ist.

5.19. TEMP_EXT

Type	Funktionsbaustein
Input	T_EXT1 : REAL (Außentemperatur Sensor 1)

	T_EXT2 : REAL (Außentemperatur Sensor 2)
	T_EXT3 : REAL (Außentemperatur Sensor 3)
	T_EXT_Setup : BYTE (Abfragemodus)
	DT_IN : DATE_TIME (Tageszeit)
Output	T_EXT : REAL (Ausgang Außentemperatur)
	HEAT : BOOL (Heizsignal)
	COOL : BOOL (Kühlsignal)
Setup	T_EXT_MIN : REAL (Minimum Außentemperatur)
	T_EXT_MAX : REAL (Maximum Außentemperatur)
	T_EXT_DEFAULT : REAL (Default Außentemperatur)
	HEAT_PERIOD_START : DATE (Beginn der Heizperiode)
	HEAT_PERIOD_STOP : DATE (Ende der Heizperiode)
	COOL_PERIOD_START : DATE (Beginn der Kühlperiode)
	COOL_PERIOD_STOP : DATE (Ende der Kühlperiode)
	HEAT_START_TEAMP_DAY (Heiztriggertemperatur Tag)
	HEAT_START_TEAMP_NIGHT (Heiztriggertemperatur Nacht)
	HEAT_STOP_TEMP : REAL (Heizen Stopp Temperatur)
	COOL_START_TEAMP_DAY (Kühl Start Temperatur Tag)
	COOL_START_TEMP_NIGHT (Kühl Start Temperatur Nacht)
	COOL_STOP_TEMP : REAL (Kühl Stopp Temperatur)
	START_DAY : TOD (Anfang des Tages)
	START_NIGHT : TOD (Anfang der Nacht)
	CYCLE_TIME : TIME (Abfragezeit für Außentemperatur)



TEMP_EXT verarbeitet bis zu 3 Außentemperaturfühler und stellt eine durch Mode selektierte Außentemperatur der Heizungsregelung zur Verfügung. Es errechnet Signale für Heizung und Kühlung abhängig von Außentemperatur, Datum und Uhrzeit. Mit dem Eingang T_EXT_Setup wird festgelegt, wie der Ausgangswert T_EXT ermittelt wird. Wird T_EXT_Setup nicht beschaltet, so ist der Vorgabewert 0. Die Setup-Werte T_EXT_MIN und T_EXT_Max legen den Mindestwert und Maximalwert der Außentem-

peratureingänge fest. Werden diese Grenzen über- oder unterschritten, so wird von einem Fehler im Sensor oder Drahtbruch ausgegangen und an Stelle des Messwertes der Vorgabewert T_EXT_DEFAULT benutzt.

T_EXT_Setup	T_EXT
0	Durchschnittswert von T_EXT1, T_ext2 und T_ext3
1	T_EXT1
2	T_EXT2
3	T_EXT3
4	T_EXT_DEFAULT
5	Niedrigster Wert der 3 Eingänge
6	Höchster Wert der 3 Eingänge
7	Mittlerer Wert der 3 Eingänge

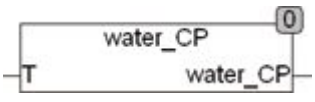
Mit den Setup-Variablen HEAT_PERIOD und COOL_PERIOD wird definiert, wann Heizen und wann Kühlen erlaubt ist. Die Entscheidung, ob der Ausgang HEAT oder COOL TRUE wird hängt weiterhin von den Setup-Werten HEAT_START- HEAT_STOP und COOL_START und COOL_STOP ab. Diese Werte können separat für Tag und Nacht definiert werden. Der Start für eine Tag- und Nachtperiode kann durch die Setup-Variablen START_DAY und START_NIGHT festgelegt werden. Ein Variable CYCLE_TIME legt fest, wie oft die Außentemperatur abgefragt werden soll.

5.20. WATER_CP

Type Funktion : REAL

Input T : REAL (Temperatur des Wassers in °C)

Output REAL (Spezifische Wärmekapazität bei der Temperatur T)



WATER_CP berechnet die spezifische Wärmekapazität von flüssigem Wassers in Abhängigkeit von der Temperatur bei Normaldruck. Die Berechnung ist gültig im Temperaturbereich von 0 bis 100 Grad Celsius und wird in

Joule / (Gramm * Kelvin) errechnet. Die Temperatur T wird in °C angegeben.

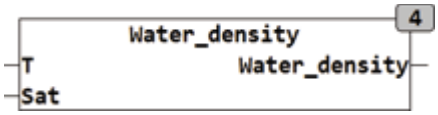
5.21. WATER_DENSITY

Type Funktion : REAL

Input T : REAL (Temperatur des Wassers)

SAT : BOOL (TRUE, wenn das Wasser mit Luft gesättigt ist)

Output REAL (Dichte des Wassers in Gramm / Liter)



WATER_DENSITY berechnet die Dichte von flüssigem Wassers in Abhängigkeit von der Temperatur bei Normaldruck. Die Temperatur T wird in °C angegeben. Die höchste Dichte erreicht Wasser bei 3,983 °C mit 999,974950 Gramm pro Liter. WATER_DENSITY berechnet die Dichte für flüssiges Wasser, nicht für gefrorenes oder verdampftes Wasser. WATER_DENSITY berechnet die Dichte von luftfreiem Wasser wenn SAT = FALSE und von luftgesättigtem Wasser wenn SAT = TRUE. Die Berechneten Werte werden nach einer Näherungsformel berechnet und liefert Werte mit einer Genauigkeit besser las 0,01% im Temperaturbereich von 0 - 100 °C bei einem konstanten Druck von 1013 mBar.

Die Abweichung der Dichte von mit Luft gesättigtem Wasser wird nach der Formel vom Bignell korrigiert.

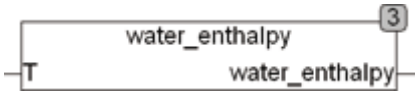
Die Abhängigkeit der Wasserdichte vom Druck ist verhältnismäßig gering mit ca. 0,046 kg/m³ je 1 Bar Druckerhöhung Im Bereich bis 50 Bar. Die geringe Druckabhängigkeit hat in praktischen Anwendungsfällen keinen nennenswerten Einfluss.

5.22. WATER_ENTHALPY

Type Funktion : REAL

Input T : REAL (Temperatur des Wassers)

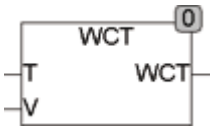
Output REAL (Enthalpie des Wassers in J/Gramm bei der Temperatur T)



WATER_ENTHALPY berechnet die Enthalpie (Wärmeinhalt) von flüssigem Wasser in Abhängigkeit von der Temperatur bei Normaldruck. Die Temperatur T wird in °C angegeben. Die Berechnung ist gültig für eine Temperatur von 0 bis 100 °C und das Ergebnis ist die Wärmemenge die benötigt wird um das Wasser von 0°C auf die Temperatur von T zu Erwärmen. Das Ergebnis wird in Joule/Gramm J/g Beziehungsweise KJ/Kg ausgegeben. Die Berechnung erfolgt durch lineare Interpolation in Schritten von 10° und erreicht damit eine für nicht wissenschaftliche Anwendungen hinreichende Genauigkeit. Eine mögliche Anwendung von WATER_ENTHALPY ist die Berechnung der Energiemenge die benötigt wird um zum Beispiel einen Pufferspeicher um X (T2 – T1) Grad zu erwärmen. Aus der benötigten Energie kann dann die Laufzeit eines Heizkessels berechnet werden und exakt die benötigte Energie bereitgestellt werden. Da Temperaturmesswerte in der Praxis stark zeitverzögert vorliegen ist mit dieser Methode eine bessere Aufheizung möglich.

5.23. WCT

Type	Funktion : REAL
Input	T : REAL (Außentemperatur in °C)
	V : REAL (Windgeschwindigkeit in km/h)
Output	REAL (Windchill Temperatur)

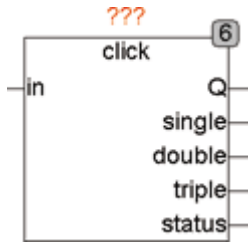


WCT berechnet die Windchill-Temperatur abhängig von der Windgeschwindigkeit in Km/h und der Außentemperatur in °C. Die Windchill-Temperatur ist nur definiert für Windgeschwindigkeiten größer als 5 Km/h und Temperaturen kleiner 10 °C. Für Werte außerhalb des definierten Bereichs wird die Eingangstemperatur ausgegeben.

6. Elektrotechnik

6.1. CLICK

Type	Funktionsbaustein
Input	IN : BOOL (Steuereingang für Taster)
Output	Q : BOOL (Schaltausgang)
	SINGLE : BOOL (Ausgang für einfachen Tastendruck)
	DOUBLE : BOOL (Ausgang für doppelten Tastendruck)
	TRIPLE : BOOL (Ausgang für dreifachen Tastendruck)
	STATUS : BYTE (ESR kompatibler Status Ausgang)
Setup	T_DEBOUNCE : TIME (Entprellzeit für Taster)
	T_SHORT : TIME (Maximale Zeit für kurzen Impuls)
	T_PAUSE : TIME (Maximale Pause zwischen 2 Impulsen)
	T_RESetup : TIME (Rekonfigurationszeit)

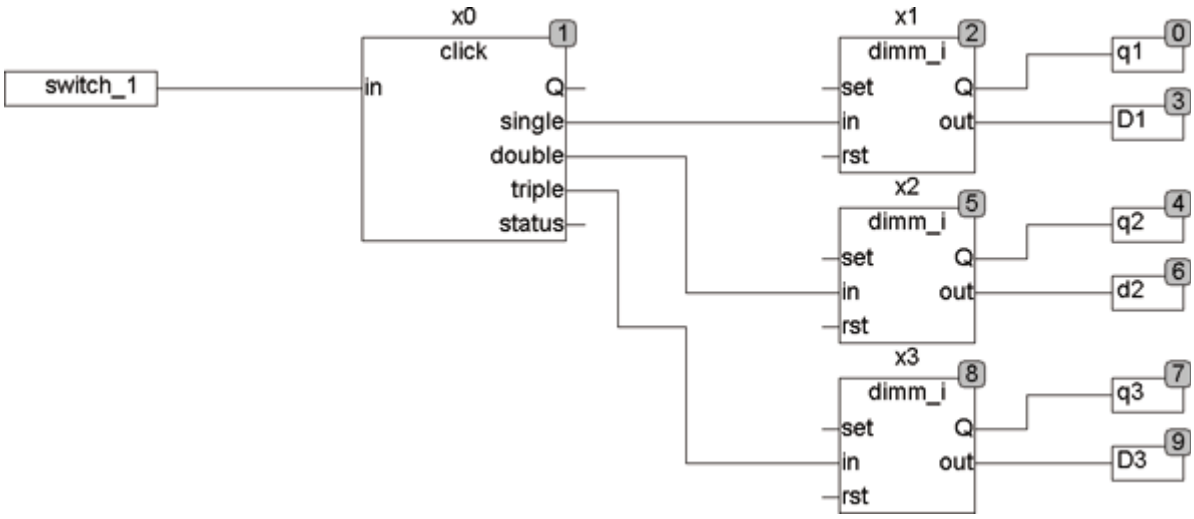


CLICK ist ein Tastinterface das sich selbsttätig auf den angeschlossenen Taster einstellt. Wird ein Taster angeschlossen, so erkennt CLICK selbst, ob es ein Öffner oder Schließer ist und wertet dann nur die jeweils erste Flanke aus. Mit der Setup-Variable T_DEBOUNCE wird die Entprellzeit des Tasters festgelegt. Sie ist standardmäßig auf 10ms voreingestellt. Die Zeit T_RESetup wird verwendet, um zu entscheiden ob ein Schließer oder Öffner am Eingang IN angeschlossen ist. Bleibt der Eingang länger als diese Zeit in einem Zustand, so wird dies als Ruhestellung angenommen. Der Vorgabewert für T_RESetup beträgt 1 Minute. Mit kurz aufeinander folgenden Impulsen wird ein einfacher, doppelter oder dreifacher Puls ausgewertet und schaltet entsprechend den Ausgang SINGLE, DOUBLE oder TRIPLE ein. Ist der Puls länger als die Setup-Zeit T_SHORT oder eine Pause zwischen 2 Impulsen länger als T_PAUSE, so wird die Puls-Sequenz unterbrochen und der entsprechende Ausgang gesetzt, bis der Eingangspuls wider inaktiv wird. Der Ausgang Q entspricht dem Eingangsimpuls. Jedoch ist er immer High-aktiv. Ein ESR kompatibler Status Ausgang meldet Zustandsänderungen an nachfolgende ESR kompatible Auswertemodule. Mit kurzen

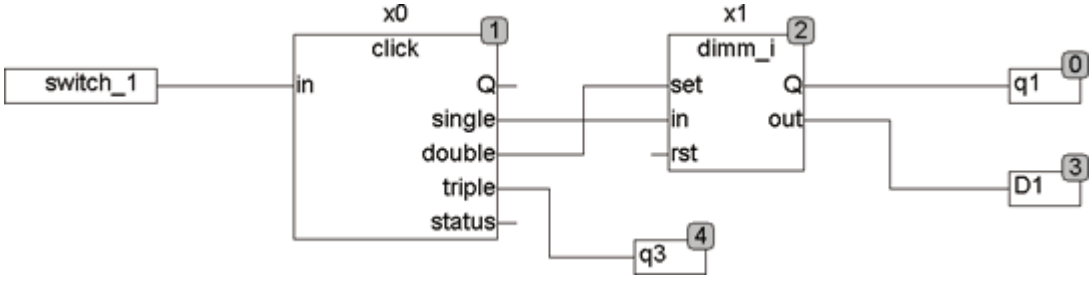
Impulsen wird der Ausgang SINGLE (ein Puls), DOUBLE (2 Impulse) oder TRIPLE (3 Impulse) ausgewählt. Der entsprechende Ausgang bleibt mindestens einen Zyklus aktiv und maximal solange wie der Eingang IN aktiv bleibt.

Status	
110	Eingang inaktiv
111	Ausgang SINGLE aktiviert
112	Ausgang DOUBLE aktiviert
113	Ausgang TRIPLE aktiviert

Beispiel 1 zeigt eine Anwendung von CLICK mit 3 nachfolgenden Dimm Bausteinen. Analog können auch bis zu 3 Schalter oder eine Mischung aus Dimmern oder Schaltern benutzt werden.

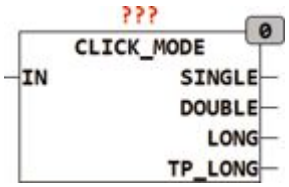


Beispiel 2 zeigt CLICK mit einem Dimmer, wobei sich der Dimmer wie ein Dimmer ohne CLICK verhält, jedoch ein kurzer Doppelklick den Ausgang des Dimmers sofort auf 100% setzt und ein Dreifachklick als weiterer Schaltausgang zur Verfügung steht.



6.2. CLICK_MODE

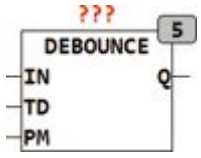
Type	Funktionsbaustein
Input	IN : BOOL (Steuereingang für Taster)
Output	SINGLE : BOOL (Ausgang für einfachen Tastendruck)
	DOUBLE : BOOL (Ausgang für doppelten Tastendruck)
	LONG : BOOL (Ausgang für einen langen Tastendruck)
	TP_LONG : BOOL (Impuls wenn Langer Tastendruck startet)
Setup	T_LONG : TIME (Dekodierzeit für Langen Tastendruck)



CLICK_MODE ist ein Taster Interface das sowohl einfachen Klick, Doppelklick oder Lange Tastendrücke dekodiert. Mit kurzen Impulsen wird ein einfacher oder Doppelklick dekodiert und schaltet entsprechend die Ausgänge SINGLE oder DOUBLE für jeweils einen Zyklus ein. Ist der Puls länger als die T_LONG, so wird der Ausgang TP_LONG für einen Zyklus auf TRUE gesetzt und der Ausgang LONG bleibt solange TRUE bis der Eingang IN wieder auf FALSE geht.

6.3. DEBOUNCE

Type	Funktionsbaustein
Input	IN : BOOL (Eingangssignal vom Schalter oder Taster)
	TD : TIME (Entprellzeit)
	PM : BOOL (Betriebsart TRUE = Impulsbetrieb)
Output	Q : BOOL (Ausgangssignal)

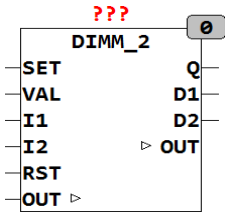


DEBOUNCE kann das Signal von einem Schalter oder Taster entprellen und am Ausgang Q entprellt bereitstellen. Wenn PM = FALSE folgt der Ausgang

Q dem entprellten Eingangsignal IN, ist PM = TRUE wird am Eingang In eine steigende Flanke detektiert und der Ausgang Q bleibt nur für einen Zyklus auf TRUE. Die Entprellzeit für den Eingang IN wird mit der Zeit TD eingestellt.

6.4. DIMM_2

Type	Funktionsbaustein
Input	SET : BOOL (Eingang zum Einschalten des Ausgangs auf VAL)
	VAL : BYTE (Wert für die SET Operation)
	I1 : BOOL (Steuereingang für Taster1, Auf)
	I2 : BOOL (Steuereingang für Taster2, Ab)
	RST : BOOL (Eingang zum Ausschalten des Ausgangs)
Output	Q : BOOL (Schaltausgang)
	D1 : BOOL (Ausgang für Doppelklick an I1)
	D2 : BOOL (Ausgang für Doppelklick an I2)
I/O	OUT : Byte (Dimmer Ausgang)
Setup	T_DEBOUNCE : TIME (Entprellzeit für Taster)
	T_ON_MAX : TIME (Einschaltbegrenzung)
	T_DIMM_START : TIME (Reaktionszeit zum Dimmen)
	T_DIMM : TIME (Zeit für eine Dimm Rampe)
	MIN_ON : BYTE (Minimalwert von OUT beim Einschalten)
	MAX_ON : BYTE (Maximalwert von OUT beim Einschalten)
	RST_OUT : BOOL (Reset setzt OUT auf 0 gesetzt wenn TRUE)
	SOFT_DIMM : BOOL (Soft Start beim Einschalten)
	DBL1_TOG : BOOL (Enable Toggle für D1)
	DBL2_TOG : BOOL (Enable Toggle für D2)
	DBL1_SET : BOOL (Enable Wert für Doppelklick I1)
	DBL2_SET : BOOL (Enable Wert für Doppelklick I2)
	DBL1_POS : BYTE (Stellwert bei Doppelklick an I1)
	DBL2_POS : BYTE (Stellwert bei Doppelklick an I2)



DIMM_2 ist ein intelligenter Dimmer für 2-Taster Bedienung. Der Dimmer kann über Setup-Variablen eingestellt werden. Die Zeit T_DEBOUNCE dient zur Entprellung des Tasters und ist standardmäßig auf 10ms eingestellt. Eine Einschaltbegrenzung T_ON_MAX schaltet den Ausgang automatisch ab, wenn sie überschritten wird. Die Zeiten T_DIMM_Start und T_DIMM legen das Zeitverhalten des Dimmers fest.

Mit den Eingängen SET und RST kann der Ausgang Q jederzeit Ein- beziehungsweise Aus-geschaltet werden. SET setzt dabei den Ausgang OUT auf den durch VAL Vorgegebenen Wert, RST setzt OUT auf 0 wenn die Setup Variable RST_OUT auf TRUE steht. RST schaltet zusätzlich D1 und D2 auf FALSE. SET und RST können unter anderem zum Anschluss von Brandmeldeanlagen oder Alarmanlagen benutzt werden. Mit SET können im Brandfall oder bei Einbruch alle Leuchten auf Ein oder mit RST beim verlassen des Gebäudes Zentral auf Aus geschaltet werden.

Beim Ein- und Aus- schalten bleibt der letzte Ausgangswert des Dimmers am Ausgang OUT erhalten, lediglich ein FALSE am Ausgang Q schaltet das Leuchtmittel ab, und TRUE am Ausgang Q schaltet das Leuchtmittel wieder ein. Beim Einschalten durch einen kurzen Tastendruck limitiert der Baustein den Ausgang OUT auf mindestens MIN_ON und maximal MAX_ON. Steht z.B. der Dimmer auf 0 so setzt der Baustein den Ausgang OUT automatisch auf 50 und umgekehrt wird der Ausgang OUT falls er höher als MAX_ON steht auf MAX_ON begrenzt. Diese Parameter sollen verhindern das nach dem Einschalten ein sehr kleiner Wert am Ausgang OUT anliegt und trotz aktiven Q kein Licht angeht. Es wird durch den Parameter MIN_ON ein minimaler Leuchtwert beim Einschalten vorgegeben. Umgekehrt kann z.B: bei Schlafzimmern verhindert werden das beim Einschalten sofort volle Leuchtstärke anliegt. Wird der Parameter SOFT_DIMM auf TRUE gesetzt, so beginnt das DIMMEN beim Einschalten mit langem Tastendruck immer bei 0. zusätzlich zur Funktion des Dimmers wird an den Eingängen I1 und I2 ein Doppelklick dekodiert der die Ausgänge D1 beziehungsweise D2 für einen Zyklus auf TRUE setzt. Wird die Setup Variable D?_TOGGLE auf TRUE gesetzt so wird der Ausgang D? bei jedem Doppelklick invertiert. Die Ausgänge D1 und D2 können benutzt werden um zusätzliche Verbraucher oder Ereignisse mit einem Doppelklick zu schalten. Ein Ausgang D? kann auch auf den Eingang SET gelegt werden und der Dimmer mittels eines Doppelklicks auf einen durch VAL vordefinierten Wert gesetzt werden. Wird die Setup Variable DBL?_SET auf TRUE gesetzt, so wird bei einem entsprechenden Doppelklick nicht der zugehöriger Ausgang D? verändert, sondern es wird der Wert der Variable DBL?_POS auf den Ausgang OUT geschrieben und der Ausgang Q falls nötig eingeschaltet. OUT ist der Wert des Dimmers und wird als I/O Variable extern defi-

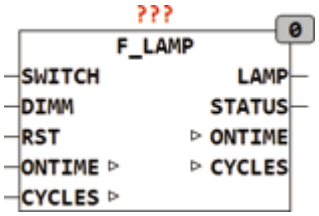
niert. dies hat den Vorteil das der Wert des Dimmers jederzeit extern beeinflusst werden kann und auch nach einem Stromausfall wieder rekonstruiert werden kann. OUT kann auf Wunsch Remanent und Persistent definiert werden.

Die folgende Tabelle zeigt die Betriebszustände des Dimmers:

I1	I2	SET	RST	Q	D1	D2	OUT
single	-	0	0	1	-	-	LIMIT(MIN_ON,OUT,MAX_ON)
-	single			0	-	-	
double	-	0	0		TOG PULSE		
-	double	0	0			TOG PULSE	
long	-	0	0	1	-		dimm up start from 1 if SOFT_DIMM = TRUE
-	long			1			dimm down and turn off at 0
-	-	1	0	ON	-		VAL
-	-	0	1	OFF	OFF		0 wenn RST_OUT = TRUE

6.5. F_LAMP

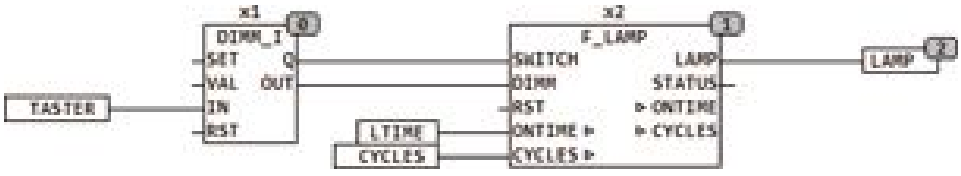
Type	Funktionsbaustein
Input	SWITCH : BOOL (Schalteingang vom Dimmer)
	DIMM : BYTE (Eingang vom Dimmer)
	RST : BOOL (Eingang zum Rücksetzen des Zählers)
Output	LAMP : BYTE (Dimmer Ausgang)
	STATUS : BYTE (ESR kompatibler Status Ausgang)
I/O	ONTIME : UDINT (Betriebszeit in Sekunden)
	CYCLES : UDINT (Anzahl der Schaltzyklen des Leuchtmittels)
Setup	T_NO_DIMM : UINT (Sperrzeit für Dimmer in Stunden)
	T_Maintenance : UINT (Meldezeit für Lampenwechsel in Stunden)



F_LAMP ist ein Lampeninterface für Leuchtstofflampen. Der Ausgang LAMP folgt dem Eingang DIMM und SWITCH. Wenn Dimm nicht beschaltet wird ist die Vorgabe 255 und der Ausgang LAMP schaltet zwischen 0 und 255 abhängig von SWITCH. Die Ausgänge ONTIME und CYCLES zählen die Betriebszeit des Leuchtmittels in Sekunden und die Schaltzyklen. Beide Werte werden extern gespeichert und können Remanent oder Persistent gespeichert werden, weitere Infos hierzu finden Sie beim Baustein ONTIME. Ein TRUE am Eingang RST setzt diese beiden Werte auf 0 zurück. Mit der Setup-Variablen T_NO_DIMM wird festgelegt, nach welcher Betriebsdauer eines neuen Leuchtmittels mit dem Dimmen begonnen werden darf. Dieser Wert ist, wenn er nicht vom Anwender anders eingestellt wird, auf 100 Stunden voreingestellt. Leuchtstofflampen dürfen während der ersten 100 Betriebsstunden nicht in Ihrer Leuchtkraft reduziert werden, sonst wird Ihre Lebensdauer drastisch verkürzt. Durch einen RST beim Lampenwechsel verhindert dieser Baustein das Dimmen in der Anfangsphase. Der Ausgang Status ist ESR kompatibel und kann Betriebszustände melden, aber auch eine Meldung zum Lampenwechsel absetzen. Die voreingestellte Zeit für T_MAINTENANCE beträgt, falls vom Anwender nicht anders eingestellt, 15000 Stunden. Wird T_Maintenance auf 0 gesetzt so wird keine Meldung zum Lampenwechsel generiert.

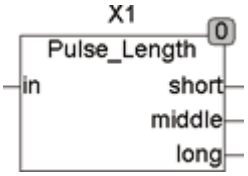
Status	
110	Lampe ausgeschaltet
111	Lampe eingeschaltet Dimmen nicht erlaubt
112	Lampe eingeschaltet Dimmen erlaubt
120	Aufforderung zum Lampenwechsel

Das folgende Beispiel zeigt die Anwendung des Bausteins F_LAMP in Verbindung mit DIMM_I:



6.6. PULSE_LENGTH

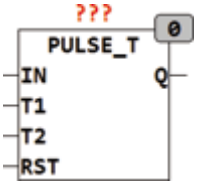
Type	Funktionsbaustein
Input	IN : BOOL (Eingangspuls)
Output	SHORT : BOOL (Puls wenn IN < T_SHORT)
	MIDDLE : BOOL (Puls wenn IN =< T_LONG und IN >= T_SHORT)
	LONG : BOOL (TRUE wenn IN > T_SHORT)
Setup	T_SHORT : TIME (Maximale Länge für kurzen Puls)
	T_LONG : TIME (Minimale Länge für Langen Puls)



PULSE_LENGTH setzt bei einem Eingangspuls an IN einen der 3 Ausgänge. Der Ausgang SHORT wird für einen Zyklus TRUE, wenn der Eingangspuls kleiner als T_SHORT ist. Der Ausgang MIDDLE wird für einen Zyklus TRUE, wenn der Eingangspuls zwischen T_SHORT und T_LONG lang ist. Der Ausgang LONG wird gesetzt, sobald der Eingangsimpuls T_LONG überschritten hat und bleibt solange auf TRUE, wie der Eingangsimpuls auf TRUE bleibt.

6.7. PULSE_T

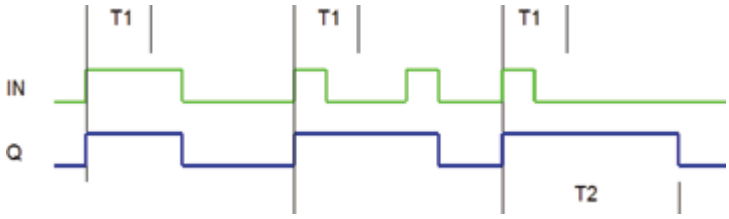
Type	Funktionsbaustein
Input	IN : BOOL (Eingangspuls)
	T1 : TIME (Minimalzeit)
	T2 : TIME (Maximalzeit)
Output	Q : BOOL (Ausgangspuls)



PULSE_T erzeugt einen Ausgangsimpuls der Länge T2 wenn der Eingang IN kürzer als T1 auf TRUE geht. Bleibt der Eingang IN länger als T1 auf TRUE so folgt der Ausgang Q dem Eingang IN und geht zeitgleich mit IN wieder auf FALSE. Bleibt IN länger als die Zeit T2 auf TRUE wird der Aus-

gang nach Ablauf der Zeit T2 automatisch auf FALSE zurückgesetzt. Eine weiterer Impuls an IN während der Ausgang TRUE ist setzt den Ausgang mit der fallenden Flanke von IN auf FALSE. liegt der Eingang IN länger als die Zeit T2 auf TRUE so wird der Ausgang Q nach Ablauf der Zeit T2 automatisch auf FALSE gesetzt.

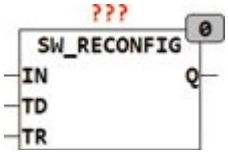
Das folgende Zeitdiagramm zeigt einen Eingangsimpuls der länger als T1



anliegt und den Ausgang Q der dem Eingang folgt. Anschließend wird am Eingang IN ein kurzer Puls (kürzer als T1) erzeugt und der Ausgang bleibt aktiv bis Ihn ein weiterer Impuls an IN wieder löscht. Ein weiterer kurzer Impuls am Eingang IN setzt den Ausgang auf TRUE bis dieser nach Ablauf der Zeit T2 selbsttätig gelöscht wird.

6.8. SW_RECONFIG

Type	Funktionsbaustein
Input	IN : BOOL (Taster Eingang)
	TD : TIME (Entprellzeit für den Eingang)
	TR : TIME (Rekonfigurationszeit)
Output	Q : BOOL (Schaltausgang)



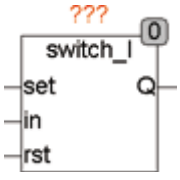
SW_RESetup ist ein intelligentes Taster Interface, es kann den Eingang entprellen und erkennt selbständig ob ein Öffner oder Schließer am Eingang IN angeschlossen ist. Wird am Eingang IN ein Öffner erkannt, so wird der Ausgang Q invertiert. Wird am Eingang IN ein Schalter angeschlossen, so erzeugt der Baustein bei jedem Zustandswechsel des Schalters einen Puls mit der Länge TR. TD ist die Entprellzeit und TR die Rekonfigurationszeit. immer Dann wenn der Eingang IN länger als die Rekonfigurationzeit in einem Zustand bleibt geht der Ausgang auf FALSE und wird somit beim nächsten Impuls an Eingang in einen High aktiven Impuls ausgeben. In der

praktischen Installationstechnik kann dies von großem Vorteil sein wenn Schalter manchmal als Öffner und manchmal als Schließer angeschlossen sind.

Die folgende Grafik verdeutlicht die Funktionsweise des Bausteins:

6.9. SWITCH_I

Type	Funktionsbaustein
Input	SET : BOOL (Eingang zum Einschalten des Ausgangs auf 100%)
	IN : BOOL (Steuereingang für Taster)
	RST : BOOL (Eingang zum Ausschalten des Ausgangs)
Output	Q : BOOL (Schaltausgang)
Setup	T_DEBOUNCE : TIME (Entprellzeit für Taster)
	T_RESetup : TIME (Rekonfigurationszeit)
	T_ON_MAX : TIME (Einschaltbegrenzung)

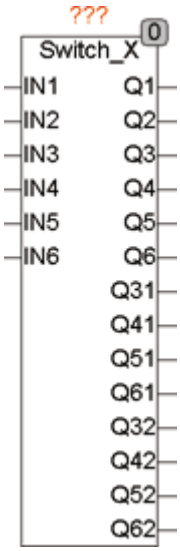


SWITCH_I ist ein intelligenter Schalter der sich selbsttätig auf den angeschlossenen Taster oder Schalter einstellt. Wird ein Schalter angeschlossen, so folgt der Ausgang jeder Schaltflanke des Schalters. Wird jedoch ein Taster angeschlossen, so erkennt SWITCH_I selbst, ob es ein Öffner oder Schließer ist und wertet dann nur die jeweils erste Flanke aus. Die Setup-Variable T_ON_MAX legt fest, nach welcher Zeit der Ausgang automatisch wieder ausgeschaltet werden soll. Mit den Eingängen SET und RST kann der Ausgang jederzeit auf 100% ein oder ausgeschaltet werden. Anwendungsbeispiele sind die Meldung von Rauchmeldern oder Alarmanla-

gen. Die Zeit T_DEBOUNCE dient zum Entprellen des Tasters und ist standardmäßig auf 10ms eingestellt. Die Zeit T_RESetup wird verwendet, um zu entscheiden, ob ein Schließer oder Öffner am Eingang IN angeschlossen ist. Bleibt der Eingang länger als diese Zeit in einem Zustand, so wird dies als Ruhestellung angenommen.

6.10. SWITCH_X

Type	Funktionsbaustein
Input	IN1..6 : BOOL (Taster Eingänge)
Output	Qx : BOOL (Schaltausgänge)
Setup	T_DEBOUNCE : TIME (Entprellzeit für Taster)

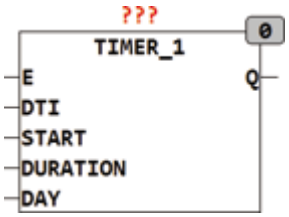


SWITCH_X ist ein Interface für bis zu 6 Taster. Die einzelnen Taster werden mit der Entprellzeit T_DEBOUNCE entprellt und schalten die jeweiligen Ausgänge Q1 bis Q6. IN3 bis IN6 werden direkt auf die Ausgänge geschaltet, wenn sie alleine betätigt werden. IN1 und IN2 erzeugen einen Puls für einen Zyklus nachdem sie betätigt wurden. Wird während IN1 oder IN2 betätigt ist, einer der Eingänge IN3 bis IN 6 betätigt, so wird kein Ausgangsimpuls an Q1 bis Q6 erzeugt, sondern es wird der entsprechende Ausgang Q31 bis Q62 aktiviert. Q42 wird zum Beispiel dann aktiviert, wenn IN4 betätigt wird, während IN2 betätigt ist. Q2 und Q4 werden dann nicht aktiv.

SWITCH_X erlaubt es also auf den Eingängen IN3 bis IN6 eine Dreifachbelegung zu Realisieren und diese durch Betätigen von IN1 oder IN2 und einem weiteren Eingang auszuwählen.

6.11. TIMER_1

Type	Funktionsbaustein
Input	E : BOOL (Enable Eingang)
	DTI : DATE_TIME (Datum Zeit Eingang)
	START : TOD (Startzeit)
	DURATION : TIME (Zeitdauer des Ausgangssignals)
	DAY : BYTE (Auswahl der Wochentage)
Output	Q : BOOL (Schaltausgang)



TIMER_1 erzeugt an selektierbaren Tagen der Woche ein Ausgangsereignis Q mit einer programmierbaren Dauer (DURATION) und festgelegter Anfangszeit START. DTI liefert dem Baustein die Lokalzeit. START und DURATION legt die Tageszeit und die Dauer des Ereignisses fest. Der Eingang DAY legt fest, an welchen Wochentagen das Ereignis erzeugt wird. Wird DAY auf 0 gesetzt, so kein Ereignis erzeugt. Eine DURATION = 0 legt fest, dass das Ausgangssignal nur für einen Zyklus gesetzt wird. Das erzeugte Ausgangssignal kann auch über Mitternacht verlaufen, oder es kann auch länger als einen Tag sein. Die maximale Impulsdauer liegt jedoch bei 49 tagen (T#49d). Der Eingang DAY ist vom Typ BYTE und die Bits 0..7 legen die Tage des Ereignisses Fest. Bit 0 entspricht Sonntag, Bit 1 Samstag .. Bit 6 entspricht Montag. Werden die Bits 0..6 in DAY gesetzt so wird jeden Tag ein Ereignis erzeugt, ansonsten nur für diejenigen Tage, für die das entsprechende Bit gesetzt ist. Der Eingang Default ist wenn er nicht beschaltet wird auf 2#0111_1111 gesetzt und somit ist der Baustein für jeden Tag aktiv. Ein zusätzlicher Enable Eingang E kann den Baustein Freischalten Dieser Eingang ist TRUE wenn er nicht beschaltet wird.

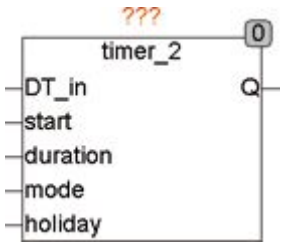
6.12. TIMER_2

Type	Funktionsbaustein
Input	DT_IN : DATE_TIME (Datum Zeit Eingang)
	START_TIME : TOD (Startzeit)
	DURATION : TIME (Zeitdauer des Ausgangssignals)

MODE : BYTE (Tagesauswahl)

HOLIDAY : BOOL (Feiertagssignal)

Output Q : BOOL (Schaltausgang)

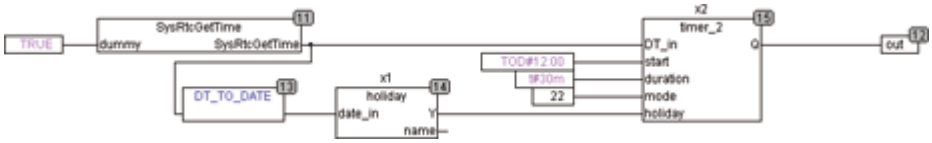


TIMER_2 erzeugt ein Ausgangsereignis mit einer programmierbaren Dauer. DT_IN liefert dem Baustein die Lokalzeit. START_TIME und DURATION legt die Tageszeit und die Dauer des Ereignisses fest. Der Eingang Mode legt fest, wie oft und an welchem Tagen das Ereignis erzeugt werden soll. HOLIDAY ist ein Eingangssignal, das anzeigt ob der aktuelle Tag ein Feiertag ist. Dieses Signal kann vom Baustein HOLIDAY erzeugt werden.

MODE	Q
0	Es wird kein Ausgangssignal erzeugt
1	nur am Montag
2	nur am Dienstag
3	nur am Mittwoch
4	nur am Donnerstag
5	nur am Freitag
6	nur am Samstag
7	nur am Sonntag
11	jeden Tag
12	alle 2 Tage
13	alle 3 Tage
14	alle 4 Tage
15	alle 5 Tage
16	alle 6 Tage
20	Wochentage (Montag bis Freitag)
21	Samstag und Sonntag
22	Arbeitstage (Wochentage ohne Feiertage)
23	Feiertage und Wochenende
24	Nur an Feiertagen
25	Erster Tag im Monat
26	Letzter Tag im Monat

27	Letzter Tag im Jahr (31. Dezember)
28	Erster Tag im Jahr (1. Januar)

Beispiel für die Anwendung von TIMER_2:



Das Beispiel zeigt die System-Routine (in diesem Fall für einen Wago Controller), die die interne Uhr ausliest und DATE_TIME für TIMER_2 und HOLIDAY bereitstellt. HOLIDAY liefert die Feiertagsinformation an TIMER_2. TIMER_2 liefert in diesem Beispiel an Wochenenden (Samstag und Sonntag), sowie an Feiertagen (Mode = 22) ein Ausgangssignal jeweils um 12:00 Mittags für eine Dauer von 30 Minuten. TIMER_2 erzeugt limitiert von der Zykluszeit immer die exakte DURATION am Ausgang. TIMER_2 merkt sich an welchen Tag er den letzten Ausgangsimpuls erzeugt hat, so dass sichergestellt ist das nur ein Impuls pro Tag erzeugt wird.

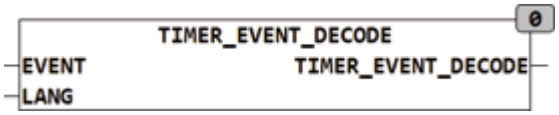
6.13. TIMER_EVENT_DECODE

Type Funktion

Input EVENT : STRING (Event Zeichenkette)

LANG : INT (Sprachauswahl)

OUTPUT TIMER_EVENT



TIMER_EVENT_DECODE erlaubt die Programmierung von Timer Ereignissen mittels Zeichenkette anstelle des Ladens der Struktur TIMER_EVENT.

Die Ereignisse werden wie folgt spezifiziert:

<Typ;Kanal;Day;Start;Dauer;Land;Lor>

Element	Beschreibung	**Formate**
< >	Start und Stopp Zeichen des Datensatzes.	
Typ	Typ des Ereignisses (siehe Beschreibung in TIMER_P4)	'123', 2#0101, 8#33, 16#FF

Kanal	zu programmierender Kanal	'123', 2#0101, 8#33, 16#FF
Day	Auswahlnummer z.B. Tag	'123', 2#0101, 8#33, 16#FF, 'Mo' 'MO,DI,DO'
Start	Startzeitpunkt (Tageszeit)	'TOD#12:00'
Dauer	Zeitdauer des Ereignisses	'T#1h3m22s'
Land	logische Und Verknüpfung	'123', 2#0101, 8#33, 16#FF
Lor	logische oder Verknüpfung	'123', 2#0101, 8#33, 16#FF

Das Feld DAY hat je nach Typ des Ereignisses verschiedenen Bedeutung und kann auch mit Wochentagen als Text oder einer Liste von Wochentagen spezifiziert werden. Der Eingang LANG spezifiziert die zu verwendende Sprache, 0 = die im Setup eingestellt Default Sprache, 1 = Englisch, ... nähere Infos zu Sprachen siehe im Kapitel Datentypen.

6.14. TIMER_EXT

Type	Funktionsbaustein
Input	ENA : BOOL (Baustein Enable, default TRUE!)
	ON : BOOL (zwingt den Ausgang Q auf TRUE)
	OFF : BOOL (zwingt den Ausgang Q auf FALSE)
	MAN : BOOL (Steuereingang wenn ON = OFF = TRUE)
	SWITCH : BOOL (Eingang für Taster)
	DT_IN : DATETIME (Eingang für Datum und Tageszeit)
	SUN_SET : TOD (Zeit des Sonnenuntergangs)
	SUN_RISE : TOD (Zeit des Sonnenaufgangs)
	HOLIDAY : BOOL (Eingang für Feiertagsmodul)
Setup	T_DEBOUNCE : TIME (Entprellzeit für den Eingang SWITCH)
	T_RISE_START : TIME (Einschaltzeit vor Sonnenaufgang)
	T_RISE_STOP : TIME (Ausschaltzeit nach Sonnenaufgang)
	T_SET_START : TIME (Einschaltzeit vor Sonnenuntergang)
	T_SET_STOP : TIME (Ausschaltzeit nach Sonnenuntergang)
	T_DAY_START : TOD (Einschaltzeit nach Tageszeit)

T_DAY_STOP : TOD (Ausschaltzeit nach Tageszeit)

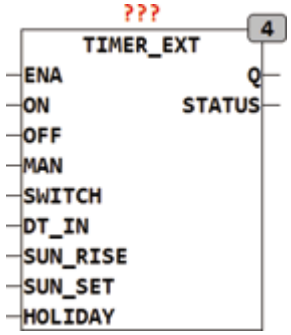
ENABLE_SATURDAY : BOOL (aktiv an Samstagen wenn TRUE)

ENABLE_SUNDAY : BOOL (aktiv an Sonntagen wenn TRUE)

ENABLE_HOLIDAY : BOOL (aktiv an Feiertagen wenn TRUE)

Output Q : BOOL (Schaltausgang)

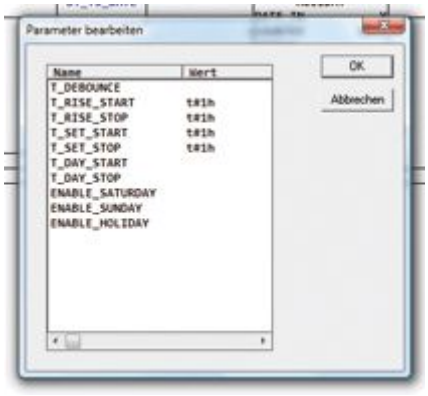
Status : BYTE (ESR kompatibler Status Ausgang)



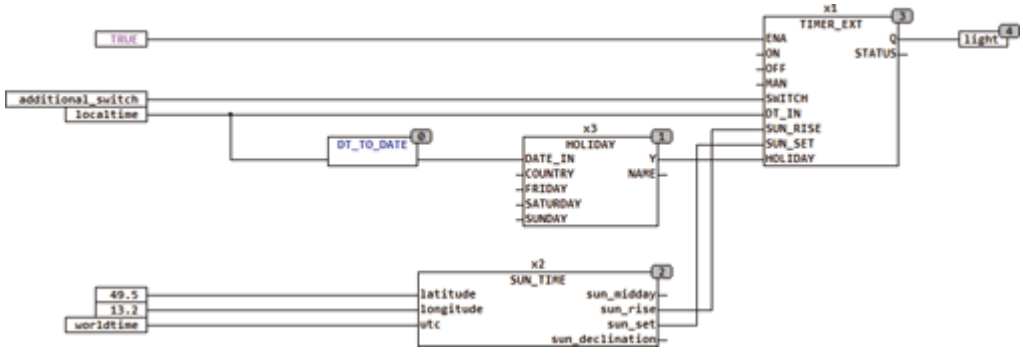
TIMER_EXT ist ein Timer speziell für Außenbeleuchtungen oder andere Verbraucher, die während der Dämmerung geschaltet werden sollen. Der Ausgang Q kann zu festen Tageszeiten ein-/ausgeschaltet werden, zusätzlich können Zeitspannen festgelegt werden, zu denen Q vor der Dämmerung ein- und nach der Dämmerung automatisch wieder ausgeschaltet wird. Ein zusätzlicher Eingang SWITCH schaltet den Ausgang unabhängig von der Tageszeit ein/aus. Die Eingänge ENA, ON, OFF und MAN erlauben eine ausführliche automatische und manuelle Steuerung des Ausgangs. Die folgende Tabelle gibt detaillierte Informationen über die Betriebszustände des Bausteins:

ENA	ON	OFF	MAN	SWITCH	Timer	Q	STATUS
L	-	-	-	-	-	L	104
H	H	L	-	-	-	H	101
H	L	H	-	-	-	L	102
H	H	H	X	-	-	MAN	103
H	L	L	-	¿	-	NOT Q	110
H	L	L	-	-	TOD = T_DAY_START	H	111
H	L	L	-	-	TOD = T_DAY_STOP	L	112
H	L	L	-	-	TOD = SUN_RISE - T_RISE_START	H	113
H	L	L	-	-	TOD = SUN_RISE + T_RISE_STOP	L	114
H	L	L	-	-	TOD = SUN_SET - T_SET_START	H	115

H	L	L	-	-	TOD = SUN_SET + T_SET_STOP	L	116

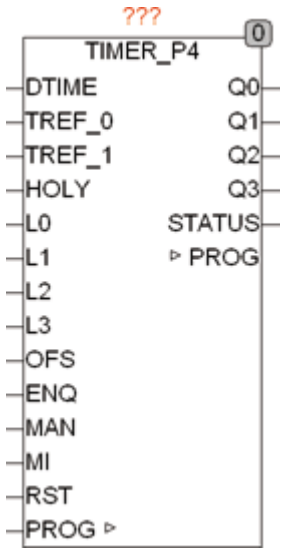


Die Setup Variablen ENABLE_SUNDAY, SATURDAY und HOLIDAY definieren die Aktivität des Bausteins an Samstagen, Sonntagen und Feiertagen. Sollen Feiertage durch diesen Baustein berücksichtigt werden, muss am Eingang HOLIDAY der Baustein HOLIDAY aus der Bibliothek angeschlossen werden. Dieser Baustein signalisiert mit einem TRUE, dass der aktuelle Tag ein Feiertag ist. Die Setup Variablen T_SET_START, T_SET_STOP, T_RISE_START, T_RISE_STOP, T_DAY_START und T_DAY_STOP legen die Schaltzeiten fest. Ein T#0s bzw. TOD#00:00 als übergebener Wert deaktiviert die jeweilige Schaltzeit. Das bedeutet, dass z.B. T_SET_START (Einschaltzeitspanne vor Sonnenuntergang) nur dann einschaltet, wenn sie auf mindestens 1 Sekunde (T#1s) eingestellt ist. Der Baustein schaltet zum Zeitpunkt T_DAY_START den Ausgang Q ein und zum Zeitpunkt T_DAY_STOP wieder aus. Steht eine der beiden Zeiten (T_DAY_START oder T_DAY_STOP) auf TOD#00:00, wird der entsprechende Schaltvorgang nicht ausgeführt. Der Baustein schaltet die Zeitspanne T_RISE_START vor Sonnenaufgang (SUN_RISE) ein und die Zeitspanne T_RISE_STOP nach Sonnenaufgang wieder aus. Gleiches gilt für die Zeiten zum Sonnenuntergang.



6.15. TIMER_P4

Type	Funktionsbaustein
Input	DTIME : DATE_TIME (Datum Zeit Eingang)
	TREF_0 : TOD (Referenzzeit 0)
	TREF_1 : TOD (Referenzzeit 1)
	HOLY : BOOL (Feiertagseingang)
	L0..L3 : BOOL (Logische Eingänge)
	OFS : INT (Kanal Offset)
	ENQ : BOOL (Wenn ENQ = FALSE bleiben alle Ausgänge FALSE)
	MAN : BOOL (Umschalter für Handbetrieb)
	MI : BYTE (Kanalwahl bei Handbetrieb)
	RST : BOOL (Asynchroner Reset)
I/O	PROG : ARRAY[0..63] OF TIMER_EVENT (Programmdaten)
Output	Q0..Q3 : BOOL (Schaltausgänge)
	STATUS : BYTE (ESR kompatibler Status Ausgang)



TIMER_P4 ist ein universell programmierbarer Timer der ein Fülle von Möglichkeiten bietet. Neben Ereignissen zu festen Zeiten können auch Ereignisse in Abhängigkeit von externen Zeiten wie zum Beispiel Sonnen- Auf oder Sonnen- Untergang programmiert werden. Zusätzlich zur zeitlichen Programmierung können alle Ausgänge flexibel mit logischen Eingängen verknüpft werden. Mit maximal 63 unabhängig programmierbaren Ereignissen stehen dem Anwender praktisch unbegrenzte Möglichkeiten zur Verfügung.

Die Programmierung des Timers erfolgt über ein ARRAY[0..63] OF TIMER_EVENT. Es können dabei beliebig viele Ereignisse je Kanal und auch überlappende Ereignisse Erzeugt werden.

Die Datenstruktur TIMER_EVENT enthält folgende Felder:

Datenfeld	**Datentyp**	**Beschreibung**
CHANNEL	BYTE	Kanalnummer
TYP	BYTE	Ereignistyp
DAY	BYTE	Tag oder andere Nummer
START	TOD	Startzeitpunkt
DURATION	TIME	Zeitdauer des Events
LAND	BYTE	Maske für Logisches UND
LOR	BYTE	Maske für Logisches ODER
LAST	DWORD	Interne Benutzung

Im Datenfeld CHANNEL wird der für das Ereignis relevante Kanal spezifiziert, falls mehrere Kanäle gleichzeitig geschaltet werden sollen müssen je Kanal separate Ereignisse programmiert werden. Der TYP des Ereignisses legt fest welche Art von Ereignis programmiert werden soll, siehe hierzu die Übersicht in der folgenden Tabelle. DAY legt entweder als Bitmaske die Wochentage (Bit7 = MO, BIT0 = SO) fest, oder den Tag im Monat / Jahr oder eine je nach Ereignistyp definierte andere Nummer beziehungsweise Anzahl. START ist die jeweilige Anfangszeit (TIMEOFDAY) des Ereignisses, bei Ereignissen in Abhängigkeit einer externen Zeit kann START auch eine Zeitverschiebung definieren. Die Dauer legt unabhängig vom Typ des Ereignisses fest wie lange das Ereignis andauert. Wurde ein Ereignis gestartet merkt sich der Timer in der Datenstruktur den jeweiligen Tag so dass jedes Ereignis maximal einmal pro Tag gestartet wird. Sollen mehrere Ereignisse je Tag und Kanal definiert werden, können diese durch mehrere unabhängige Ereignisse programmiert werden. LAND und LOR definieren Logische Masken für zusätzliche Logische Verknüpfungen, eine detaillierte Beschreibung der möglichen Zustandsverknüpfungen findet sich weiter unten im Text.

Der Timer hat einen zusätzlichen manuellen Eingang der es erlaubt Ausgänge Manuell zu überschreiben. Wenn MAN = TRUE ist werden die 4 untersten Bits des Eingangs MI auf die Ausgänge Q geschaltet. Der Eingang ENQ ist ein Freigabeeingang und muss für normalen Betrieb auf TRUE stehen, wird ENQ auf FALSE gestellt, bleiben alle Ausgänge auf FALSE. Der Baustein kann jederzeit mittels des asynchronen Eingangs RST zurückgesetzt werden, dabei werden alle laufenden Ereignisse gelöscht. Der Eingang OFS wird nur dann benützt wenn mehrere TIMER Bausteine kaska-

diert werden, der WERT an OFS legt dann fest welche Kanalnummer der erste Ausgang des Bausteins hat. Wird OFS beispielsweise auf 4 gesetzt so reagiert der entsprechende Baustein nicht mehr auf die Kanalnummern 0..3 sondern auf die Kanäle 4..7. Somit lassen sich mehrere Bausteine einfach kaskadieren.

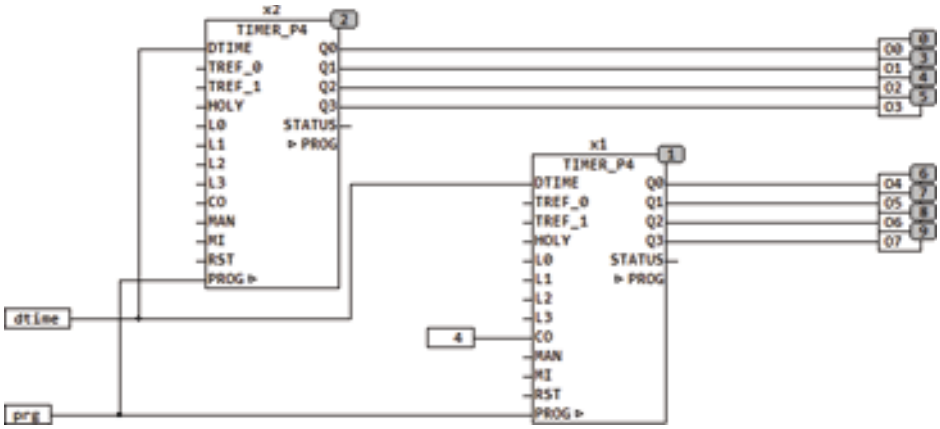
Der Ausgang STATUS ist ein ESR kompatibler Status Ausgang der die Betriebszustände des Bausteins meldet.

STATUS = 100 (Der Baustein ist disabled, ENQ = FALSE)

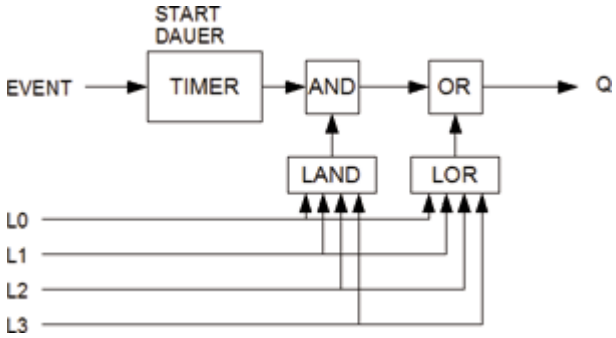
STATUS = 101 (Handbetrieb, MAN = TRUE)

STATUS = 102 (automatischer Betrieb)

Das folgende Beispiel zeigt 2 kaskadierte Timer:



Blockschaltbild des Timers:



Tritt ein programmiertes Ereignis ein so wird der entsprechende Timer des Angesprochenen Kanals mit der vordefinierten Zeitdauer gestartet. Der Kanalausgang kann durch logisches UND mit bis zu 4 Eingängen L0..L3 verknüpft werden, es werden dabei nur die Eingänge verknüpft die in der Ereignismaske LAND mit einer 1 Definiert sind. enthält die Maske LAND keine 1 (2#00000000) dann wird kein Eingang mit dem Ausgang verknüpft. Enthält die Maske LAND zum Beispiel 2#00001001) dann wird das Ausgangssignal des Timers mit den Logischen Eingängen L0 und L3 per

AND verknüpft. Der Ausgang wird in diesem Fall nur Dann TRUE wenn sowohl ein Ereignis den Timer gestartet hat und gleichzeitig auch L0 und L3 TRUE sind. Nach der UND Verknüpfung kann der Ausgang noch zusätzlich mit beliebigen logischen Eingängen in der selben Weise mittels der Maske LOR OR verknüpft werden.

Folgende Ereignisse können Programmiert werden:

TYP	**Beschreibung**	**DAY**	**Start**	**Duration**
1	tägliches Ereignis	-	Anfangszeit	Dauer
2	Ereignis an selektierten Wochentagen	B0..6	Anfangszeit	Dauer
3	Ereignis alle N Tage	N	Anfangszeit	Dauer
10	wöchentliches Ereignis	Tag der Woche	Anfangszeit	Dauer
20	monatliches Ereignis	Tag des Monats	Anfangszeit	Dauer
21	letzter Tag des Monats	-	Anfangszeit	Dauer
30	jährliches Ereignis	Tag des Jahres	Anfangszeit	Dauer
31	letzter Tag des Jahres	-	Anfangszeit	Dauer
40	Ereignis an Schalttagen	-	Anfangszeit	Dauer
41	Ereignis an Feiertagen	-	Anfangszeit	Dauer
42	an Feiertagen und Wochenenden	-	Anfangszeit	Dauer
43	Ereignis an Werktagen	-	Anfangszeit	Dauer
50	Ereignis nach externer Zeit	0,1	Offset	Dauer
51	Ereignis vor externer Zeit	0,1	-Offset	Dauer
52	Ausgang zu Zeit + Offset setzen	0,1,2	Offset	
53	Ausgang zu Zeit + Offset löschen	0,1,2	Offset	
54	Ausgang zu Zeit - Offset setzen	0,1,2	Offset	
55	Ausgang zu Zeit - Offset löschen	0,1,2	Offset	

Ereignistypen:

1. tägliches Ereignis

bei einem täglichen Ereignis wird lediglich Kanalnummer, Startzeit und Dauer des Ereignisses Programmiert. Das Feld DAY hat keine Bedeutung.

2. Ereignis an selektierten Wochentagen

bei diesem Ereignis wird der Timer an selektierbaren Wochentagen gestartet. Im Feld DAY wird dabei über eine Bitmaske festgelegt an welchen Wochentagen das Ereignis zu starten ist. Montag = Bit 6, ... Sonntag = Bit 0. Das Ereignis wird nur an den Wochentagen gestartet wenn das entsprechende Bit im Feld DAY TRUE ist.

3. Ereignis alle N Tage

hierbei wird nach Ablauf von N Tagen das definierte Ereignis gestartet. Im Feld DAY wird dabei angegeben nach wie vielen Tagen das Ereignis gestartet wird. N=3 bedeutet das das Ereignis jeden 3ten Tag gestartet wird. N kann hierbei Werte von 1..255 annehmen.

10. wöchentliches Ereignis

hierbei wird an einem bestimmten Tag in der Woche das Ereignis gestartet, der entsprechende Tag wird im Feld DAY festgelegt: 1 = Montag.....7 = Sonntag.

20. monatliches Ereignis

Bei monatlichen Ereignissen wird im Feld DAY der entsprechende Tag des Monats an dem das Ereignis stattfinden soll festgelegt. DAY = 24 bedeutet das das Ereignis jeweils am 24. eines Monats gestartet wird.

21. Ende des Monats

Da Monate keine feste Länge haben ist es Sinnvoll auch ein Ereignis am letzten Tag eines Monats generieren zu können. In diesen Mode hat DAY keine Bedeutung.

30. jährliches Ereignis

Bei jährlichen Ereignissen wird im Feld DAY der entsprechende Tag des Jahres an dem das Ereignis stattfinden soll festgelegt. DAY = 33 bedeutet das das Ereignis jeweils am 33. Tag eines Jahres gestartet wird, was dem dem 2. Februar entspricht.

31. Ende des Jahres

Da Jahre keine feste Länge haben ist es Sinnvoll auch ein Ereignis am letzten Tag eines Jahres generieren zu können. In diesen Mode hat DAY keine Bedeutung. Das Ereignis wird immer am 31. Dezember erzeugt.

40. Ereignis an Schalttagen

Dieses Ereignis wird nur am 29. Februar, also nur in einem Schaltjahr generiert. DAY hat hierbei keine Bedeutung.

41. Ereignis an Feiertagen

Dieses Ereignis wird nur dann erzeugt wenn der Eingang HOLY = TRUE ist. An diesem Eingang muss zu diesem Zweck der Baustein HOLDAY aus der

Bibliothek angeschlossen werden. Wird dieser Mode nicht genutzt kann der Eingang HOLY offen bleiben. Das Feld DAY hat hierbei keine Bedeutung.

42. Ereignis an Feiertagen und Wochenenden

Dieses Ereignis wird erzeugt wenn der Eingang HOLY = TRUE ist, oder ein Samstag oder Sonntag vorliegt. Am Eingang HOLY muss zu diesem Zweck der Baustein HOLDAY aus der Bibliothek angeschlossen werden. Wird dieser Mode nicht genutzt kann der Eingang HOLY offen bleiben. Das Feld DAY hat hierbei keine Bedeutung.

43. Ereignis an Werktagen

Dieses Ereignis wird nur an den Wochentagen Montag bis Freitag erzeugt. Das Feld DAY hat hierbei keine Bedeutung.

50. Ereignis nach externer Zeit

hierbei wird ein tägliches Ereignis erzeugt das von einer externen Zeit abhängig ist. IM Feld START wird hierbei nicht die Startzeit selbst, sondern der Offset von der externen Zeit festgelegt. Im Feld DAY wird angegeben welche externe Zeit als Referenz herangezogen wird. DAY = 0 bedeutet TREF_0 und DAY = 1 entspricht TREF_1. Ein Ereignis nach externer Zeit ist zum Beispiel ein Ereignis 1 Stunde nach Sonnenuntergang. In diesem Fall würde an TREF_1 (DAY muss hierzu auf 1 stehen) die Zeit des Sonnenuntergangs eingespeist werden, und im Feld START die Zeit 01:00 (eine Stunde Offset) angegeben. Die Zeiten für Sonnen- Auf und Sonnen- Untergang können aus dem Baustein SUN_TIME aus der Bibliothek eingespeist werden.

51. Ereignis vor externer Zeit

hierbei wird ein tägliches Ereignis erzeugt das von einer externen Zeit abhängig ist. IM Feld START wird hierbei nicht die Startzeit selbst, sondern der Offset vor der externen Zeit festgelegt. Im Feld DAY wird angegeben welche externe Zeit als Referenz herangezogen wird. DAY = 0 bedeutet TREF_0 und DAY = 1 entspricht TREF_1. Ein Ereignis vor externer Zeit ist zum Beispiel ein Ereignis 1 Stunde vor Sonnenuntergang. In diesem Fall würde an TREF_1 (DAY muss hierzu auf 1 stehen) die Zeit des Sonnenuntergangs eingespeist werden, und im Feld START die Zeit 01:00 (eine Stunde Offset vor TREF_1) angegeben. Die Zeiten für Sonnen- Auf und Sonnen-Untergang können aus dem Baustein SUN_TIME aus der Bibliothek eingespeist werden.

52 Ausgang setzen nach externer Zeit

Ein Ereignis vom Typ 52 schaltet den Ausgang bei erreichen der externen Zeit + START ein. Die externe Zeit ist TREF1 wenn DAY = 1 oder TREF_0 wenn DAY = 0, ist DAY > 1 ist die externe Zeit 0. Der Ausgang bleibt dann solange auf TRUE bis er durch ein neues Ereignis überschrieben oder durch ein separates Ereignis wieder gelöscht wird.

53 Ausgang löschen mit externem Offset

Ein Ereignis vom Typ 53 schaltet den Ausgang bei erreichen der externen Zeit + START aus. Die externe Zeit ist TREF1 wenn DAY = 1 oder TREF_0 wenn DAY = 0, ist DAY > 1 ist die externe Zeit 0.

54 Ausgang setzen mit negativen Offset

Ein Ereignis vom Typ 54 schaltet den Ausgang bei erreichen der externen Zeit - START ein. Die externe Zeit ist TREF1 wenn DAY = 1 oder TREF_0 wenn DAY = 0, ist DAY > 1 ist die externe Zeit 0. Der Ausgang bleibt dann solange auf TRUE bis er durch ein neues Ereignis überschrieben oder durch ein separates Ereignis wieder gelöscht wird.

55 Ausgang löschen mit negativen Offset

Ein Ereignis vom Typ 55 schaltet den Ausgang bei erreichen der externen Zeit - START aus. Die externe Zeit ist TREF1 wenn DAY = 1 oder TREF_0 wenn DAY = 0, ist DAY > 1 ist die externe Zeit 0.

7. Jalousiesteuerung

7.1. Einleitung

Die folgenden Bausteine sind so konzipiert und aufeinander abgestimmt dass Sie einen modularen Aufbau eines Jalousiecontrollers ermöglichen. Dieses modulare System erlaubt einen schnellen und einfachen Aufbau von einfachen bis komplexen Jalousiecontrollern die exakt auf die Anwendung abgestimmt sind. Das System lässt sich auch später jederzeit erweitern und erlaubt einen beliebigen Ausbau des Funktionsumfangs. Die Anwendungen umfassen Jalousien aller Art, Rollladen, und alle Arten von Beschattungseinrichtungen. Die Module sind so gestaltet das sie sich einfach hintereinander schalten lassen und die Reihenfolge der Verschaltung gleichzeitig die Priorität der Funktionen festlegt. Die Signale UP und DN für den Handbetrieb, sowie die Vorgaben für Winkel und Position im Automatikbetrieb (PI und AI) werden dabei von Modul zu Modul weitergereicht, was einen simplen Signalfluss und einen übersichtlichen Aufbau gewährleistet. Eine Besonderheit ist dabei das Die Signale UP und DN wenn beide gleichzeitig TRUE sind den Automatikmodus einschalten.

Alle Bausteine haben die Eingänge UP und DN mit denen der Manuelle Auf und Ab Befehl empfangen wird und durch QU und QD an den nächsten Baustein weitergereicht wird. Werden beide Eingänge UP und DN auf TRUE gesetzt so schalten die betreffenden Module auf Automatik Modus und werten die Eingänge PI und AI (Position- und Winkel- Eingang für die Jalousie aus.

Durch die Reihenschaltung einzelner Funktionen kann im Falle einer Fehlersuche auf einfache Weise der Signalfluss und die Funktionsweise überprüft werden. Durch die Reihenfolge der Module wird auch gleichzeitig die Priorität der einzelnen Funktionen festgelegt und kann ohne großen Aufwand vom Anwender jederzeit geändert werden.

Zukünftige oder Kundenspezifische Funktionen können auf diese Weise einfach und schnell in die Vorhandenen Blöcke eingeschaltet werden ohne dabei die vorhandene Programmierung ändern zu müssen.

Der Ausgang STATUS gibt ESR kompatible Meldungen über den Zustand des betreffenden Bausteins aus. Wenn keine eigenen Zustandsmeldungen anstehen reicht jeder Baustein die am Eingang S_IN anliegenden Statusmeldungen an den Ausgang STATUS weiter.

Zusammenfassung aller Blind Statusmeldungen:

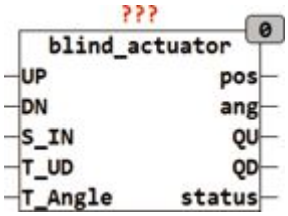
STATUS	Modul	Beschreibung
111	SECURITY	Sicherheitsstellung bei Feuer
112	SECURITY	Sicherheitsstellung bei Wind

113	SECURITY	Sicherheitsstellung bei ALARM
114	SECURITY	Sicherheitsstellung Türkontakt
115	SECURITY	Sicherheitsstellung bei Regen
1	ACTUATOR	Fehler UP und DOWN gleichzeitig
120	ACTUATOR	Auf Bewegung
121	ACTUATOR	Ab Bewegung
121	CONTROL	Auf Bewegung Position
122	CONTROL	Ab Bewegung Position
123	CONTROL	Auf Bewegung Winkel
124	CONTROL	Ab Bewegung Winkel
121	CONTROL_S	Auf Bewegung
122	CONTROL_S	Ab Bewegung
123	CONTROL_S	Auto Positionierung
127	CONTROL_S	Lockout Time
128	CONTROL_S	Kalibrierung
129	CONTROL_S	Extend Mode
130	INPUT	Standby
131	INPUT	Manual Timeout
132	INPUT	Manual Up
133	INPUT	Manual Down
134	INPUT	Single click up
135	INPUT	Single click down
136	INPUT	Forced position
137	INPUT	Double click 1
138	INPUT	Double click 2
141	NIGHT	Nachtstellung aktiv
151	SHADE	Beschattung aktiv
160-175	SCENE	Aktive Szene

178	SET	Set operation
179	SET	Restore operation

7.2. BLIND_ACTUATOR

Type	Funktionsbaustein
Input	UP : BOOL (Eingang AUF)
	DN : BOOL (Eingang AB)
	S_IN : BYTE (ESR kompatibler Status Eingang)
	T_UD : TIME (Laufzeit AUF / AB)
	T_ANGLE : TIME (Laufzeit der Lamellenverstellung)
Setup	T_LOCKOUT : TIME (Totzeit zwischen Richtungswechsel)
Output	POS : BYTE (Position der Jalousie, 0 = unten, 255 = oben)
	ANG : BYTE (Winkel der Lamelle, 0 = vertikal, 255 = horiz.)
	QU : BOOL (Motor Auf Signal)
	QD : BOOL (Motor Ab Signal)
	STATUS : BYTE (ESR kompatibler Status Ausgang)



BLIND_ACTUATOR ist ein Jalousie / Rollladen Aktor mit Simulation der Position und der Winkelstellung der Lamellen. Die Eingänge UP und DN sind gegenseitig verriegelt, so dass QU und QD nie gleichzeitig aktiv sein können. Mit der Zeit T_LOCKOUT wird die minimale Pause zwischen einem Richtungswechsel festgelegt. zusätzlich bietet BLIND_ACTUATOR noch 2 Ausgänge vom Typ Byte die die jeweilige Stellung und Position der Jalousie simulieren. Für eine exakte Simulation sind die Setup Zeiten T_UD und T_ANGLE entsprechend Einzustellen. T_UD legt die Zeit fest die zum Fahren von "geschlossen" auf "offen" (hochfahren) benötigt wird. T_ANGLE spezifiziert die Zeit die zur Verstellung der Lamellen von "senkrecht" nach

Horizontal benötigt wird. Der Aktor stellt sicher das beim Hochfahren zuerst die Lamellen auf Horizontal gestellt werden und anschließend erst mit dem Hochfahren begonnen wird. Umgekehrt werden beim Herunterfahren erst die Lamellen auf Senkrecht gestellt und dann mit dem Herunterfahren begonnen. POS = 0 bedeutet Jalousie heruntergefahren, und POS = 255 bedeutet Jalousie ist hochgefahren. Zwischenstellungen werden entsprechend mit Zwischenwerten 0 .. 255 ausgegeben. Der Winkel der Lamellen wird durch den Ausgang ANG ausgegeben, wobei ANG = 0 die vertikale Stellung und ANG = 255 die horizontale Stellung bedeuten, Werte zwischen 0 und 255 geben den entsprechenden Winkel an. Durch die Ausgänge POS und ANG wird die Information über die Jalousie Stellung der Steuerung zur Verfügung gestellt. ANG und POS können jedoch nur sinnvolle Werte liefern wenn die Zeiten T_UD und T_ANGLE exakt für die entsprechende Jalousie angepasst sind. Der Aktor kann, wenn T_ANGLE auf T#0s gesetzt wird auch für Rollladen aller Arten verwendet werden. Die Eingänge T_UD, T_ANGLE und T_LOCKOUT haben folgende Vorgabewerte:

T_UD = T#10S

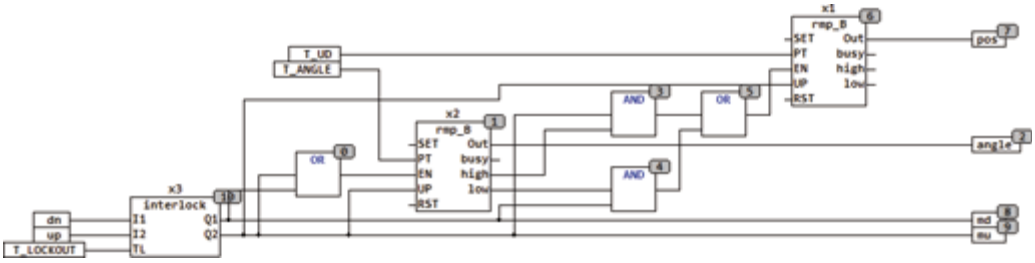
T_ANGLE = T#3S

T_LOCKOUT = T#100MS

Der Eingang S_IN und der Ausgang STATUS sind ESR kompatible Aus und Eingänge , über den Eingang S_IN melden vorgeschaltete Funktionen Ihren Status an das Modul, dieser Status wird an den Ausgang STATUS weitergeleitet, und eigene Statusmeldungen werden über STATUS Ausgegeben. Wenn am Eingang eine Statusmeldung vorliegt überschreibt diese die eigenen Statusmeldungen, ein Fehler wird mit höchster Priorität ausgegeben.

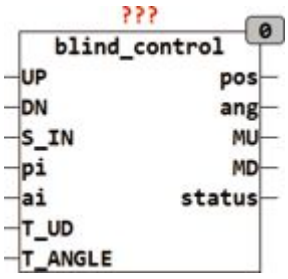
STATUS	Bedeutung
0	keine Meldung
1	Fehler, UP und DN gleichzeitig aktiv
101	Manual UP
102	Manual DN
NNN	weitergereichte Meldung

Die folgende Grafik zeigt den inneren Aufbau und die Funktionsweise des Moduls:



7.3. BLIND_CONTROL

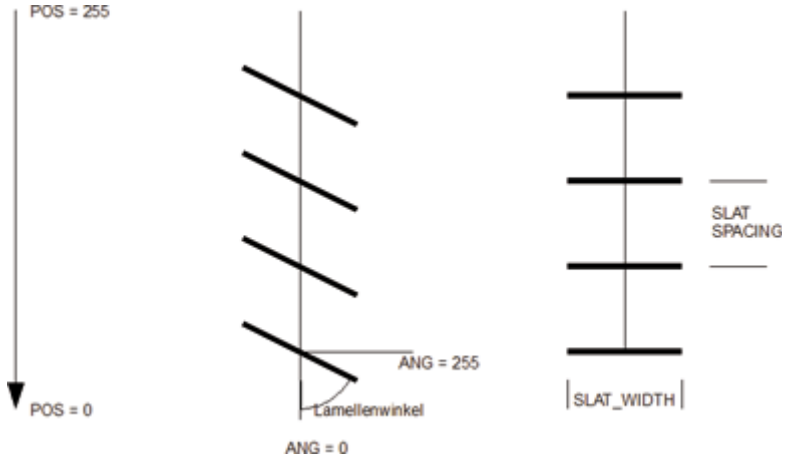
Type	Funktionsbaustein
Input	UP : BOOL (Eingang AUF)
	DN : BOOL (Eingang AB)
	S_IN : BYTE (ESR kompatibler Status Eingang)
	PI : BYTE (Vorgabe der Position)
	AI : BYTE (Vorgabe des Lamellenwinkels)
	T_UD : TIME (Zeit zum Hochfahren von 0 .. 255)
	T_ANGLE : TIME (Zeit um den Lamellenwinkel von 0 .. 255 zu verstellen)
Setup	SENS : BYTE (Auflösung der Regelung)
	T_LOCKOUT : TIME (Totzeit bei Richtungswechsel der Motoren)
Output	POS : BYTE (Simulierte Jalousiestellung)
	ANG : BYTE (Simulierter Lamellenwinkel)
	MU : BOOL (Motor Auf Signal)
	MD : BOOL (Motor Ab Signal)
	STATUS : BYTE (ESR kompatibler Status Ausgang)



BLIND_CONTROL regelt die Jalousiestellung und den Lamellenwinkel ge-

mäß den Vorgaben von PI und AI wenn UP und DN gleichzeitig TRUE sind (Automatik Modus). POS und ANG sind dabei die Aktualwerte der Jalousie, An diesen Ausgängen simuliert der Baustein den aktuellen Winkel und die Position der Jalousie. BLIND_CONTROL schaltet die Ausgänge MU oder MD solange in der geeigneten Reihenfolge auf TRUE bis die Werte an POS und ANG den Sollwerten an PI und AI entsprechen. Ein interner Sequenzer berücksichtigt dabei dass beim Auf- und Ab-fahren der Jalousie sich der Lamellenwinkel zuerst verstellt und anschließend erst das Hoch- oder Runter-fahren Der Jalousie beginnt. Wird also die Jalousie Hoch- oder Runter-gefahren so verstellt sich zwangsläufig der Lamellenwinkel welcher dann anschließend wieder durch entsprechende Gegenfahrt eingestellt werden muss. Der Eingang SENS legt fest ab welche Regeldifferenz die Regelung aktiv wird und den Ausgang so einstellt das er den Eingängen PI und AI entspricht. Ist SENS = 0 so wird jede Abweichung ausgeregelt, Ist SENS = 5 (Vorgabewert) so wird erst ab einer Differenz von 5 zwischen den Vorgabewerten und den tatsächlichen werten ausgeregelt. Wenn UP und DN nicht beide TRUE sind verlässt BLIND_CONTROL den Automatikmodus und die Ausgänge QU und QD werden im Handbetrieb direkt von UP und DN gesteuert. BLIND_CONTROL benötigt nicht den Baustein BLIND_ACTUATOR um eine JALOUSIE zu steuern, BLIND_ACTUATOR ist bereits in BLIND_CONTROL integriert. Wenn keine Automatische Regelung der Jalousie benötigt wird so kann auf BLIND_CONTROL verzichtet werden und BLIND_ACTUATOR zum Einsatz kommen. Beim Einsatz von BLIND_CONTROL ist darauf zu achten dass die Zykluszeit für den Baustein kleiner als T_ANGLE / 512 * SENS beträgt. SENS kann durch einen Doppelklick auf das grafische Symbol von BLIND_CONTROL verändert werden, der Vorgabewert ist 5. Wird die Zykluszeit zu groß so beginnt die Jalousie unregelmäßig hin und herzufahren. Wenn eine kleiner Zykluszeit nicht möglich ist kann der Wert von SENS vergrößert werden.

Die folgende Grafik verdeutlicht die Geometrie der Jalousie:



Die folgende Tabelle stellt die Betriebszustände des Bausteins dar:

UP	DN	PI	AI	MU	MD	
L	L	-	-	L	L	keine Aktion
H	L	-	-	H	L	Jalousie fährt aufwärts
L	H	-	-	L	H	Jalousie fährt abwärts
H	H	P	A	X	X	Position P und Winkel A werden Automatisch angefahren

Der Eingang S_IN und der Ausgang STATUS sind ESR kompatible Aus und Eingänge , über den Eingang S_IN melden vorgeschaltete Funktionen Ihren Status an das Modul, dieser Status wird an den Ausgang STATUS weitergeleitet, und eigene Statusmeldungen werden über STATUS Ausgegeben.

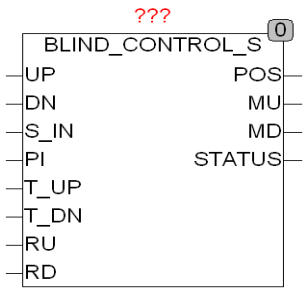
STATUS	Bedeutung
0	keine Meldung
101	Manual Auf
102	Manual Ab
121	Position Auf
122	Position Ab
123	Lamellenstellung Horizontal
124	Lamellenstellung Vertikal
NNN	weitergereichte Meldungen

7.4. BLIND_CONTROL_S

Type Funktionsbaustein

Input UP : BOOL (Eingang AUF)
DN : BOOL (Eingang AB)
S_IN : BYTE (ESR kompatibler Status Eingang)
PI : BYTE (Vorgabe der Position)
T_UP : TIME (Laufzeit des Rollos nach oben)
T_DN : TIME (Laufzeit des Rollos nach unten)
RU : BOOL (Freigabe für Öffnung unten)

	RD : BOOL (Freigabe für Öffnung oben)
Setup	T_LOCKOUT : TIME (Totzeit bei Richtungswechsel der Motoren)
	T_EXT : TIME (Verlängerungszeit bei Endanschlag)
	EXT_TRIG : BYTE (Trigger für Verlängerungszeit)
	R_POS_TOP : BYTE (Maximale Position wenn RD = TRUE)
	R_POS_BOT : BYTE (Minimale Position wenn RU = TRUE)
Output	POS : BYTE (Simulierte Position)
	MU : BOOL (Motor Auf Signal)
	MD : BOOL (Motor Ab Signal)
	STATUS : BYTE (ESR kompatibler Status Ausgang)

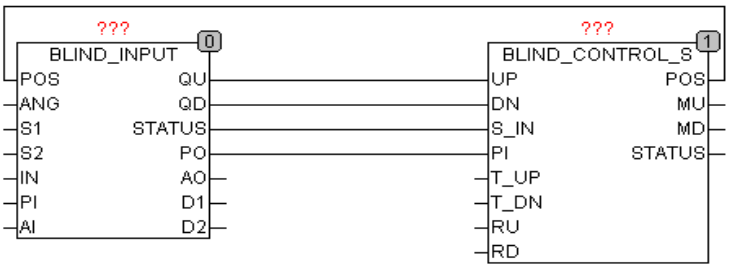


BLIND_CONTROL_S steuert und Regelt die Stellung von Rollos. Die Ausgänge MU und MD steuern die Auf und Ab Richtung der Motoren. Die Zeit T_LOCKOUT ist die Wartezeit für eine Richtungsumkehr zwischen MU und MD und die Zeiten T_UP und T_DN legen fest wie lange der Motor für eine volle Bewegung nach unten beziehungsweise nach oben benötigt. Da die Laufzeit der Motoren variieren kann wird bei erreichen einer Endposition (oben oder unten) der entsprechende Motor zusätzlich um die Zeit T_EXT angesteuert um sicherzustellen das die Endposition sicher erreicht wird, was für eine fortlaufende Kalibrierung der Anlage sorgt. Bei der ersten Inbetriebnahme und nach einem Stromausfall wird automatisch eine Kalibrierungsfahrt nach oben durchgeführt. Die Variable EXT_TRIG gibt an ab welcher Distanz vom Endwert die Fahrzeit verlängert wird. Im Automatik Modus limitiert die Einstellung R_POS_TOP wenn RD = TRUE die maximale Stellung des Rollos. So bleibt zum Beispiel der Rollo bei 240 stehen wenn RD = TRUE und R_POS_TOP = 240 sind, was im Winter ein Einfrieren in der oberen Stellung verhindern kann. Analog ist R_POS_BOT und RU = TRUE für die unterste mögliche Stellung zuständig, was im Sommer für eine Zwangslüftung sorgen kann. Der Ausgang POS ist die simulierte Stellung des Rollos, 0 = unten und 255 = oben. S_IN und STATUS sind die ESR kompatiblem Status Ein beziehungsweise Ausgänge.

UP	DN	STATUS		MU	MD
H	H	103	POS wird auf PI geregelt	Auto	Aut

					o
L	H	102	Handbetrieb Ab	L	H
H	L	101	Handbetrieb Auf	H	L
L	L	-	Manual Timeout	L	L
-	-	107	Lockout Time	L	L
-	-	108	Auto Kalibrierung	H	L
-	-	109	Zeit Verlängerung	X	X

Der Baustein wird mit anderen Bausteinen der Jalousiesteuerung verschaltet:



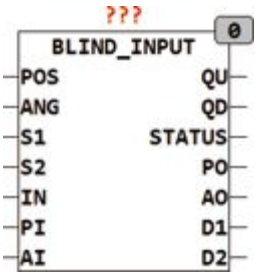
BLIND_CONTROL_S ist speziell für die Ansteuerung von Rollos und hat im Gegensatz zu Jalousien keine Winkelstellung, weshalb der Baustein auch keinen Eingang AI und keinen Ausgang ANG besitzt. BLIND_CONTROL_S kann selbstverständlich auch mit den anderen BLIND Bausteinen der Bibliothek verschaltet werden.

Der Baustein unterstützt eine automatische Kalibrierung, welche dazu führen kann das nach einem Stromausfall alle Rollos nach oben Fahren, was unter Umständen bei Abwesenheit unerwünscht ist. Deshalb ist bei Abwesenheit die gewünscht Stellung der Rollos auf den Eingang PI zu legen. Die Rollos Fahren dann zum Kalibrieren nach oben, und anschließend automatisch wieder in die gewünschte Stellung. Die automatische Kalibrierung kann jedoch wenn beide Eingänge UP und DN auf FALSE liegen verhindert werden.

7.5. BLIND_INPUT

Type Funktionsbaustein

Input	POS : BYTE (Rückführung der Jalousiestellung)
	ANG : BYTE (Rückführung des Lamellenwinkels)
	S1 : BOOL (Eingang AUF)
	S2 : BOOL (Eingang AB)
	IN : BOOL (Gesteuerter Betrieb wenn TRUE)
	PI : BYTE (Position wenn IN = TRUE)
	AI : BYTE (Winkel wenn IN = TRUE)
Setup	SINGLE_SWITCH : BOOL (TRUE für Einzeltasterbetrieb)
	CLICK_EN : BOOL (TRUE für Single Klick Mode)
	CLICK_TIME : TIME (Timeout für Klick Erkennung)
	MAX_RUNTIME : TIME (Timeout für eine Bewegung)
	MANUAL_TIMEOUT : TIME (Timeout des Handbetriebs)
	DEBOUNCE_TIME : TIME (Entprellzeit für die Eingänge S)
	DBL_CLK1 : BOOL (bei Doppelklick Position anfahren wenn TRUE)
	DBL_POS1 : BYTE (Position bei S1 Doppelklick)
	DBL_ANG1 : BYTE (Winkel bei S1 Doppelklick)
	DBL_CLK2 : BOOL (bei Doppelklick Position anfahren wenn TRUE)
	DBL_POS2 : BYTE := 255 (Position bei S2 Doppelklick)
	DBL_ANG2 : BYTE := 255 (Winkel bei S2 Doppelklick)
	D1_TOGGLE : BOOL := TRUE (Toggle Modus für D1)
	D2_TOGGLE : BOOL := TRUE (Toggle Modus für D2)
	MASTER_MODE : BOOL (aktiviert den Master Mode wenn TRUE)
Output	QU : BOOL (Motor Auf Signal)
	QD : BOOL (Motor Ab Signal)
	STATUS : BYTE (ESR kompatibler Status Ausgang)
	PO : BYTE (Ausgang Position)
	AO : BYTE (Ausgang Winkelstellung)
	D1 : BOOL (Kommandoausgang für Doppelklick Funktion 1)
	D2 : BOOL (Kommandoausgang für Doppelklick Funktion 2)



BLIND_INPUT dient als Taster Interface zur Bedienung von Jalousien. Der Baustein unterstützt 3 Modi, Handbetrieb, Automatikbetrieb und gesteuerter Betrieb. wenn IN = FALSE (Handbetrieb) werden die Eingänge S1 und S2 benutzt um die Ausgänge QU und QD zu steuern. Wenn die Setup Variable SINGLE_SWITCH = TRUE ist, dann wird der Eingang S2 ignoriert, und die gesamte Steuerung erfolgt über den Taster S1. S1 schaltet dann abwechselnd QU und QD so dass durch aufeinander folgendes Drücken des Tasters S1 zwischen Auf und Ab Bewegung gewechselt wird. Der interne Vorgabewert ist FALSE (2 Taster Konfiguration). Die Setup Variable MANUAL_TIMEOUT definiert nach welcher Ruhezeit (Zeit ohne Signal auf S1 oder S2 der Baustein selbständig in den Automatikbetrieb wechselt. Wird dieser Wert nicht spezifiziert so wird der Interne Vorgabewert von 1 Stunde verwendet. Wenn der Eingang IN = TRUE ist, werden die Ausgänge QU und QD auf Automatik (beide TRUE) gesetzt und die Eingänge PI und AI auf die Ausgänge PO und AO geschaltet. IN kann zur Übernahme der Werte kurz gepulst werden, der Baustein steuert diese Werte für die Zeit MAX_RUNTIME an und schaltet dann wieder in den Automatikmodus. Solange IN = TRUE bleibt wird der Automatikmodus mit den Werten von AI und PI forciert. Die Eingänge POS und ANG sind die Rückführungseingänge für die aktuelle Position der Jalousie. Diese Werte werden von dem Modul BLIND_CONTROL bereitgestellt. Mit der SETUP Variable CLICK_MODE wird ein Klick Betrieb festgelegt, ein kurzer Tastendruck startet die Richtung Auf für S1 und Ab für S2 und ein zweiter kurzer Tastendruck beendet die entsprechende Richtung oder kehrt die Richtung um. Diese Einstellung ist für Rollladen mit langer Laufzeit Sinnvoll, oder um mit einem kurzen Tastendruck in eine Endstellung zu fahren. wird der Tastendruck länger als die Setup Zeit CLICK_TIME so wird für diesen Tastendruck der CLICK Modus verlassen und die Jalousie fährt solange wie die Taste gedrückt bleibt im Handbetrieb. Ist ein Tastendruck kürzer als CLICK_TIME so Fährt die Jalousie weiter bis ein weiterer Klick die Fahrt beendet oder eine Endstellung erreicht wird. Der Vorgabewert für CLICK_TIME ist 500 Millisekunden und die Vorgabe für CLICK_MODE ist TRUE. Wenn beide Setup Variablen CLICK_MODE und SINGLE_SWITCH gleichzeitig TRUE sind wird ein Tastbetrieb mit nur einem Taster an S1 ermöglicht. Mit der über MAX_RUNTIME eingestellten Zeit wird die Laufzeit begrenzt die durch einen einfachen Click gestartet wird aber nicht mit einem weiteren Click beendet wird. Der Wert von MAX_RUNTIME ist mit T#60s vorbelegt und sollte solange sein das die Jalousie sicher aus jeder beliebigen Stellung die Endstellung erreichen kann. Zwei Ausgänge D1 und D2 können benutzt werden um einen Doppelklick auf S1 oder S2 auszuwerten, wenn D?_TOGGLE = TRUE schal-

tet ein Doppelklick den entsprechenden Ausgang ein und ein weitere Doppelklick wieder aus, ist D?_TOGGLE = FALSE so wird mit jedem Doppelklick ein Impuls am entsprechenden Ausgang erzeugt.

Nach einem manuellen Fahrbefehl bleibt der Baustein für die Zeit MANUAL_TIMEOUT im Modus „Manual Standby“ (STATUS = 131), die manuell angefahrene Position wird also für diese Zeit beibehalten und auch die Automatikfunktionen aller nachgeschalteten Bausteine werden unterdrückt. Durch einen langen (länger als CLICK_TIME) Druck auf beide Taster, kann der „Manual Standby“-Modus vorzeitig beendet und in den Automatikmodus zurückgekehrt werden.

Die folgende Tabelle zeigt die Betriebszustände des Bausteins:

POS ANG	**S1**	**S2**	**IN**	**PI AI**	**QU**	**QD**	**PO AO**	**D1**	**D2**	
X	L	L	L	-	H	H	X*5	-	-	Standy / Automatik Betrieb
-	-	-	H	Y	H	H	Y	-	-	gesteuerter Betrieb, PI und AI werden angefahren
X	H	L	L	-	H	L	X	-	-	Handbetrieb Auf
X	L	H	L	-	L	H	X	-	-	Handbetrieb Ab
X	H	H	L	-	H	H	X	-	-	Manual Mode vorzeitig Beenden
X	L	L	L	-	L	L	X	-	-	Handbetrieb Standby bis Timeout abläuft
X	*4	L	L	-	H	L	X	-	-	CLICK_EN = TRUE
X	L	*4	L	-	L	H	X	-	-	CLICK_EN = TRUE
-	*2	L	L	-	H	H	-	/D1	-	D1_TOGGLE = TRUE
-	*2	L	L	-	H	H	-	*3	-	D1_TOGGLE = FALSE
-	L	*2	L	-	H	H	-	-	/D2	D2_TOGGLE = TRUE
-	L	*2	L	-	H	H	-	-	*3	D2_TOGGLE = FALSE

*1 bei Übergang in den Automatikbetrieb werden die Ausgänge PO und AO auf den letzten Stand von POS und ANG gesetzt

*2 Doppelklick

*3 Ausgangsimpuls für einen Zyklus

*4 Single Klick, Jalousie läuft für MAX_RUNTIME in eine Richtung

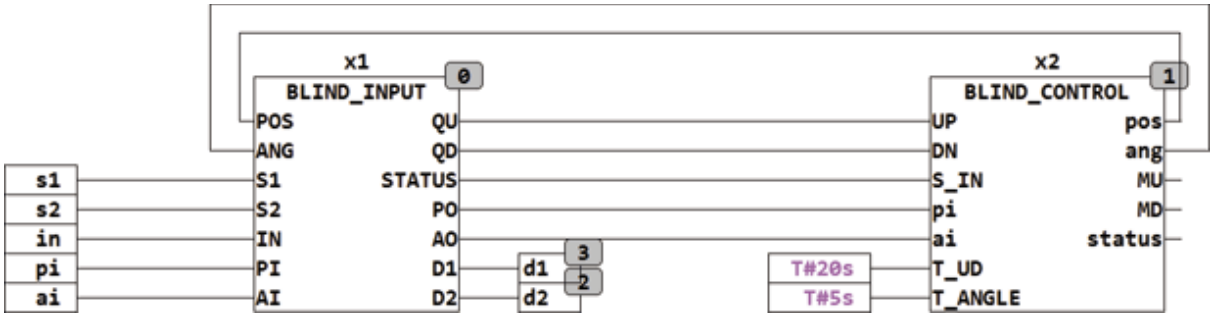
*5 Winkel und Position werden nicht übertragen wenn die Variable MASTER_MODE = TRUE ist

Der Ausgang STATUS ist ESR kompatibel und gibt Statusmeldungen über Zustandsänderungen aus.

STATUS	**Bedeutung**

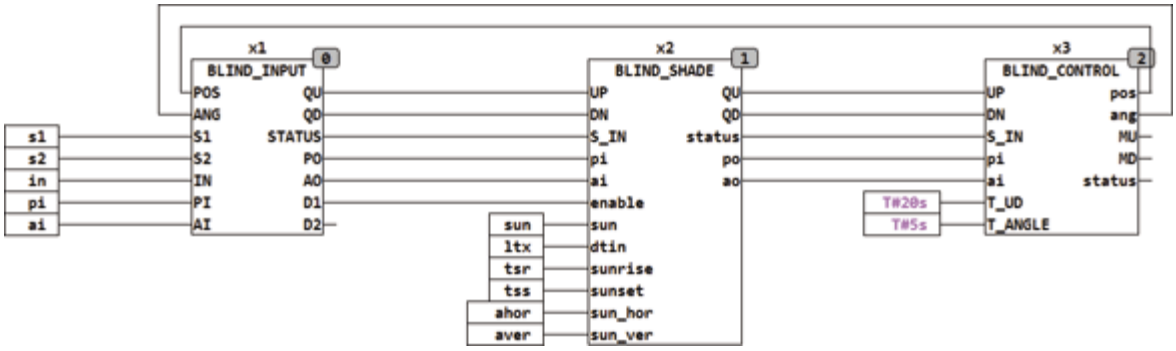
130	Standby Modus
131	Manual Standby
132	Manual Auf
133	Manual Ab
134	Single Klick Auf
135	Single Klick Ab
136	IN = TRUE forcierte Werte
137	Doppelklick Position 1 wird angefahren
138	Doppelklick Position 2 wird angefahren
139	Force Automatik Mode

Das folgende Beispiel zeigt den Aufbau eines Jalousiecontrollers mit dem Baustein BLIND_INPUT und BLIND_CONTROL:



Die Verwendung weiterer BLIND Module ist optional und dient dazu den Funktionsumfang zu erweitern. BLIND_INPUT und BLIND_CONTROL ergeben bereits eine vollwertige Jalousiesteuerung.

BLIND_INPUT kann an den beiden Eingängen S1 und S2 jeweils einen Doppelklick dekodieren und schaltet damit die beiden Ausgänge D1 und D2. Diese Ausgänge können dazu benutzt werden nachgeschaltete Funktionsblöcke oder sonstige Ereignisse zu steuern.



Master Mode:

Mit der Variable MASTER_MODE = TRUE kann der Master Mode eingeschaltet werden. Im Master Modus wird verhindert das Winkel ANG und Position POS an die Ausgänge AO und PO im Standby Mode 130 übertragen werden. Blind Bausteine die zwischen die Input- und Control- Module geschaltet sind können die Stellung der Jalousie verändern und nach Beenden der Veränderung verharrt die Jalousie in der neuen Stellung (wenn MASTER_MODE = FALSE). Wird jedoch die Variable MASTER_MODE = TRUE gesetzt so wird sichergestellt das nach beenden eines Automatischen Eingriffs durch nachgeschaltete Module der Blind Input wieder selbständig die alte Position anfährt. Wenn MASTER_MODE = FALSE wird im Status 130 POS und ANG auf die Ausgänge PO und AO übertragen. Ist MASTER_MODE = TRUE beleibt im STATUS 130 an den Ausgängen PO und AO der letzte gültige Wert erhalten und die Eingänge POS und ANG werden nicht übertragen. Das Modul BLIND_INPUT behält also die letzte gültige BLIND_INPUT Position.

7.6. BLIND_NIGHT

Type Funktionsbaustein

Input UP : BOOL (Eingang AUF)

DN : BOOL (Eingang AB)

S_IN : BYTE (ESR kompatibler Status Eingang)

PI : BYTE (Wert der Jalousiestellung im Automatikbetrieb)

AI : BYTE (Lamellenwinkel im Automatikbetrieb)

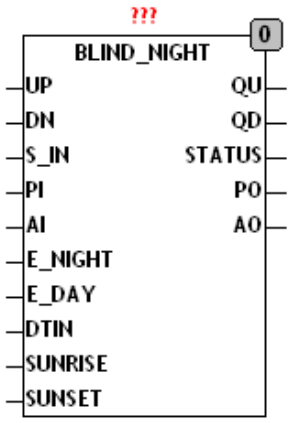
E_NIGHT : BOOL (Automatische Nachtschaltung ein)

E_DAY : BOOL (Automatische Tagschaltung ein)

DTIN : DT (Aktuelle Zeit / Datum)

SUNRISE : TOD (Sonnenaufgangszeit)

	SUNSET : TOD (Sonnenuntergangszeit)
Setup	SUNRISE_OFFSET : INT (Offset vom Sonnenaufgang in Minuten)
	SUNSET_OFFSET : INT (Offset vom Sonnenuntergang in Min.)
	NIGHT_POSITION : BYTE (Position für Nachtschaltung)
	NIGHT_ANGLE : BYTE (Winkel für Nachtschaltung)
Output	QU : BOOL (Motor Auf Signal)
	QD : BOOL (Motor Ab Signal)
	STATUS : BYTE (ESR kompatibler Status Ausgang)
	PO : BYTE (Aktuelle Jalousiestellung)
	AO : BYTE (Aktueller Lamellenwinkel)



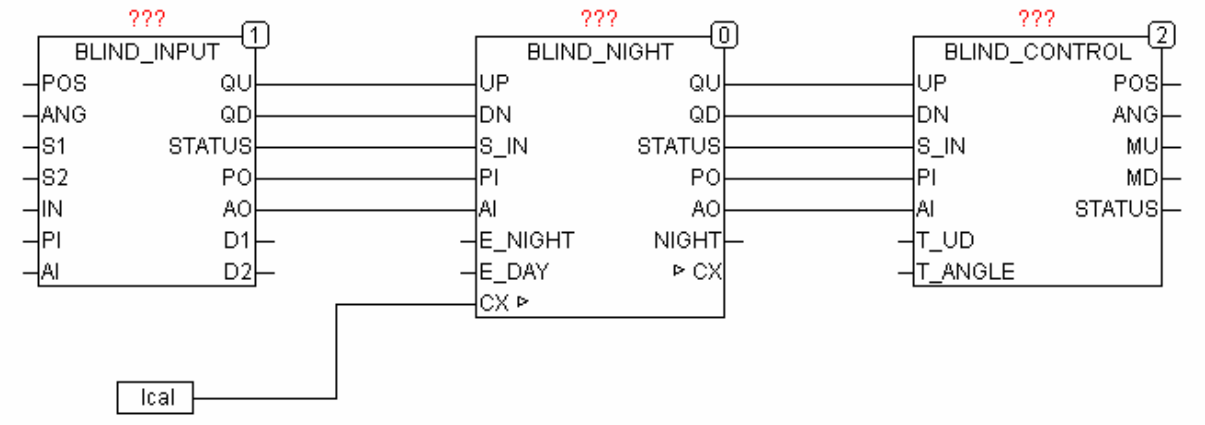
BLIND_NIGHT dient dazu die Rollladen oder Jalousie bei Nacht zu schließen. Der Baustein schließt automatisch nach Sonnenuntergang mit einer Verzögerung von SUNSET_OFFSET die Jalousie und fährt die Jalousie nach Sonnenaufgang mit einer Verzögerung von SUNRISE_OFFSET wieder hoch. Das Schließen und Öffnen kann separat mit den Eingängen E_NIGHT für schließen und E_DAY für öffnen Freigeschaltet werden. Wird zum Beispiel E_NIGHT auf TRUE gestellt und E_DAY nicht so fährt am Abend bei Dämmerung die Jalousie herunter, jedoch muss sie am nächsten Morgen manuell hochgefahren werden. Werden E_NIGHT und E_DAY nicht beschaltet so werden beide intern auf TRUE gesetzt. Damit die entsprechenden Zeiten ermittelt werden können benötigt der Baustein eine externe Datenstruktur vom Typ CALENDAR. UP, DN und S_IN sind die Eingänge von anderen BLIND Modulen und werden im Tagesbetrieb an die Ausgänge QU, QD und STATUS weitergegeben. Die Signale PI, AI und PO, AO reichen die Werte für die Position und den Lamellenwinkel der Jalousie an die folgenden Bausteine weiter. Im Nachtbetrieb werden an den Ausgängen PO und AO die Werte für den Nachtbetrieb ausgegeben, jegliche manuelle Betätigung löscht den Automatischen Nachtbetrieb. Wenn E_DAY = TRUE ist wird zum Ende der Nacht die durch DAY_POSITION und DAY_ANGLE definierte Tagesstel-

lung hergestellt. Die Zeit RESTORE_TIME ist die maximale Zeit zum anfahren der Tagesstellung.

Der Eingang S_IN und der Ausgang STATUS sind ESR kompatible Aus und Eingänge , über den Eingang S_IN melden vorgeschaltete Funktionen Ihren Status an das Modul, dieser Status wird an den Ausgang STATUS weitergeleitet, und eigene Statusmeldungen werden über STATUS Ausgegeben.

STATUS	Bedeutung
0	keine Meldung
141	Nachtbetrieb
142	Tagstellung wird angefahren
NNN	weitergereichte Meldungen

Die folgende Grafik zeigt die Verschaltung von BLIND_NIGHT mit anderen Modulen zur Jalousiesteuerung:



7.7. BLIND_SCENE

Type Funktionsbaustein

Input UP : BOOL (Eingang AUF)

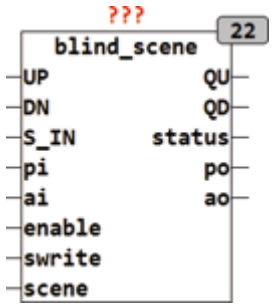
DN : BOOL (Eingang AB)

S_IN : BYTE (ESR kompatibler Status Eingang)

PI : BYTE (Eingangswert der Jalousiestellung im Automatikbetrieb)

AI : BYTE (Eingangswert des Lamellenwinkels im Automatikbetrieb)
ENABLE : BOOL (Freigabeeingang für Szenen)
SWRITE : BOOL (Schreibeingang für Szenen)
SCENE : BYTE (Nummer der Szene)

Output
QU : BOOL (Motor Auf Signal)
QD : BOOL (Motor Ab Signal)
STATUS : BYTE (ESR kompatibler Status Ausgang)
PO : BYTE (Ausgangswert der Jalousiestellung im Automatikbetrieb)
AO : BYTE (Ausgangswert des Lamellenwinkels im b Automatikbetrieb)



BLIND_SCENE speichert bis zu 16 Szenen bestehend aus jeweils aktueller Jalousiestellung und Winkel ab und kann diese Szenen bei Abruf wieder herstellen. jede einzelne Szene kann aktiv oder inaktiv sein, abhängig davon ob beim Speichern der Szene der Eingang ENABLE TRUE war oder nicht (ENABLE = TRUE bedeutet aktiv). Eine Szene wird abgerufen indem die Nummer der Szene (0 .. 15) am Eingang SCENE angelegt wird und gleichzeitig ENABLE auf TRUE gesetzt wird. Eine Szene kann nur dann abgerufen werden wenn die beiden Eingänge UP und DN gleichzeitig TRUE sind (Automatik Modus). Dadurch ist sichergestellt das eine aktive Szene immer von der Manuellen Betriebsart überschrieben wird.

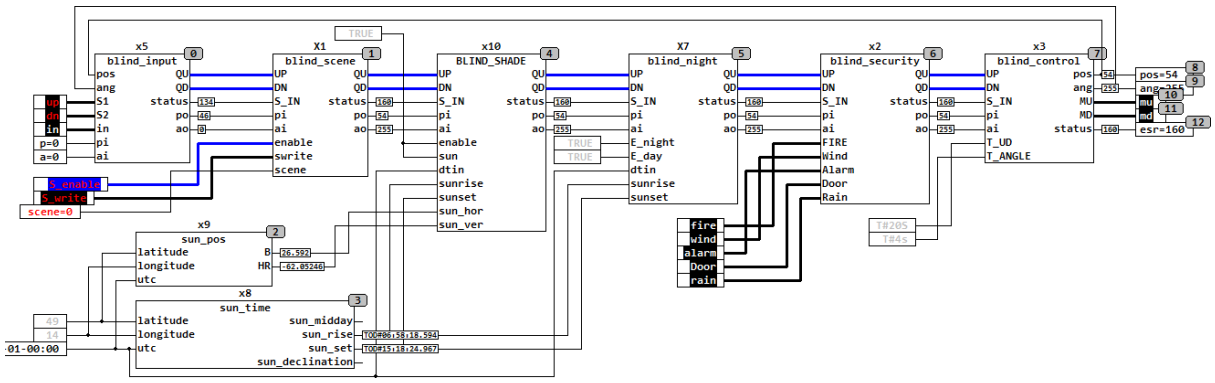
Die folgende Tabelle verdeutlicht die Funktionsweise von BLIND_SCENE:

UP	DN	ENABLE	SWRITE	SCENE	QU	QD	PO	AO	
1	1	0	0	-	1	1	PI	AI	no scene
-	-	1	1	y	-	-	-	-	write scene number y
-	-	0	1	y	-	-	-	-	disable scene number y
1	1	1	0	y	1	1	-	-	recall scene number y

Der Eingang S_IN und der Ausgang STATUS sind ESR kompatible Aus und Eingänge , über den Eingang S_IN melden vorgeschaltete Bausteine Ihren Status an das Modul, dieser Status wird an den Ausgang STATUS weitergeleitet, und eigene Statusmeldungen werden über STATUS Ausgegeben.

STATUS	Bedeutung
160 .. 175	Szenen 0..15 aktiv
176	Szene geschrieben
NNN	weitergereichte Meldungen

Die folgende Grafik zeigt die Anwendung von BLIND_SCENE mit anderen Modulen zur Ansteuerung einer Jalousie:



7.8. BLIND_SECURITY

Type Funktionsbaustein

Input UP : BOOL (Eingang AUF)

DN : BOOL (Eingang AB)

S_IN : BYTE (ESR kompatibler Status Eingang)

PI : BYTE (Jalousiestellung im Automatikbetrieb)

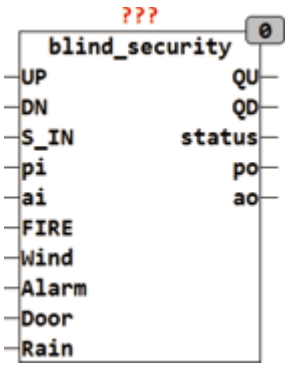
AI : BYTE (Lamellenwinkel im Automatikbetrieb)

FIRE : BOOL (Eingang für Brandalarm)

WIND : BOOL (Eingang für Windalarm)

ALARM : BOOL (Eingang für Einbruchsmeldung)

	DOOR : BOOL (Eingang für Türkontakt)
	RAIN : BOOL (Eingang für Regenmelder)
Setup	ALARM_UP : BOOL (Vorgaberichtung bei ALARM, Default = Up)
	WIND_UP : BOOL (Vorgaberichtung bei Wind, Default = Up)
	RAIN_UP : BOOL (Vorgaberichtung bei Regen, Default = Down)
Output	QU : BOOL (Motor Auf Signal)
	QD : BOOL (Motor Ab Signal)
	STATUS : BYTE (ESR kompatibler Status Ausgang)
	PO : BYTE (Ausgangswert der Jalousiestellung im Automatikbetrieb)
	AO : BYTE (Ausgangswert des Lamellenwinkels im Automatikbetrieb)

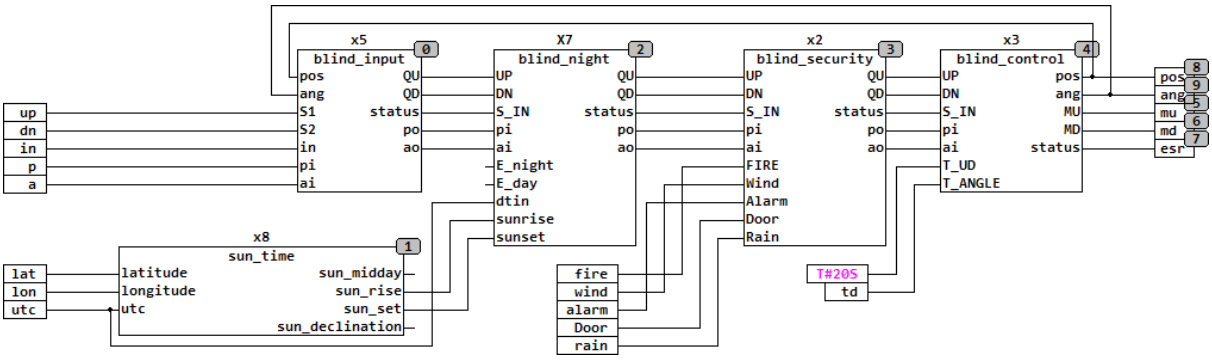


BLIND_SECURITY stellt sicher das Jalousien bei bestimmten Ereignissen entweder nach oben oder nach unten gefahren werden. Die Eingänge UP und DN steuern über die Ausgänge QU und QD ein nachgeschaltetes Modul BLIND_ACTUATOR. Mit den Eingängen FIRE, WIND, ALARM und RAIN werden die Eingänge UP und DN überschrieben und die Jalousie entweder ganz nach oben oder ganz nach unten gefahren. Hierbei hat FIRE die höchste Priorität, gefolgt von WIND, Alarm und mit der niedrigsten Priorität RAIN. Rain kann als einziger auch von den manuellen Eingängen UP und DN überschrieben werden. Sollte also der Benutzer entscheiden dass trotz Regen die Jalousie offen bleiben soll, so muss er lediglich den Regenschutz durch einen kurzen Tastendruck auf UP oder DN unterbrechen. FIRE fährt die Jalousie nach oben, während RAIN, Wind und Alarm für Auf oder Ab konfigurierbar sind. ALARM ist mit der Setup Variablen ALARM_UP sowohl für Hoch- als auch Runter-Fahrt konfigurierbar, Die Setup Variable WIND_UP legt fest ob bei Wind nach oben oder runter gefahren wird. Mit der Variable RAIN_UP wird festgelegt welche Stellung bei Regen angefahren wird. Die Vorgabewerte sind UP für Alarm, UP für Wind und DN für Regen. Die Setup Variablen können durch einen Doppelklick auf das Symbol jederzeit verändert werden.

Der Eingang S_IN und der Ausgang STATUS sind ESR kompatible Aus und Eingänge , über den Eingang S_IN melden vorgeschaltete Funktionen Ihren Status an das Modul, dieser Status wird an den Ausgang STATUS weitergeleitet, und eigene Statusmeldungen werden über STATUS Ausgegeben.

STATUS	Bedeutung
0	keine Meldung
111	Feuer
112	Wind
113	Einbruch Alarm
114	Türalarm
115	Regen
NNN	weitergereichte Meldungen

Die folgende Grafik zeigt die Anwendung von BLIND_SECURITY mit BLIND_ACTUATOR zur Steuerung einer Jalousie:



BLIND_SECURITY muss unbedingt direkt vor BLIND_CONTROL eingesetzt werden. Sollten andere Module zwischen BLIND_SECURITY und BLIND_CONTROL eingebaut werden so sind die Sicherheitsfunktionen nicht mehr gewährleistet.

7.9. BLIND_SET

Type Funktionsbaustein

Input UP : BOOL (Eingang AUF)

DN : BOOL (Eingang AB)

S_IN : BYTE (ESR kompatibler Status Eingang)

PI : BYTE (Jalousiestellung im Automatikbetrieb)

AI : BYTE (Lamellenwinkel im Automatikbetrieb)

IN : BOOL (Eingang für Brandalarm)

PX : BYTE (Eingang für Windalarm)

AX : BYTE (Eingang für Einbruchsmeldung)

Setup OVERRIDE_MANUAL : BOOL (erlaubt Manual Override wenn TRUE)

RESTORE_POSITION : BOOL (WENN TRUE wird alte Position wiederhergestellt)

RESTORE_TIME : TIME (Laufzeit zum Herstellen der Letzen Position Default = T#60s)

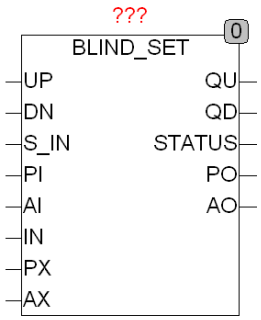
Output QU : BOOL (Motor Auf Signal)

QD : BOOL (Motor Ab Signal)

STATUS : BYTE (ESR kompatibler Status Ausgang)

PO : BYTE (Ausgangswert der Jalousiestellung)

AO : BYTE (Ausgangswert des Lamellenwinkels)



BLIND_SET kann an jeder beliebigen Stelle einer BLIND Anwendung eingesetzt werden um eine definierte Position (PX, AX) zu forcieren. Mittels der Setup Variable OVERRIDE_MANUAL wird festgelegt ob der Baustein auch einen Manual Betrieb überschreiben darf. Wird die Variable RESTORE_POSITION auf TRUE gesetzt merkt sich der Baustein die letzte Position und steuert diese nach dem forcierten Betrieb wieder automatisch an. Die Variable RESTORE_TIME legt fest wie lange der Baustein aktiv bleibt um die letzte Position wieder anzufahren. Wird RESTORE_POSITION nicht gesetzt so bleibt der forcierte Zustand beim Rückschalten in den Automatik Modus besethen.

Zustandstabelle von BLIND_SET:

UP	DN	PI AI	IN	PX AX	QU	QD	STATUS	PO AO	MANUAL_ OVERRIDE	

1	1	X	0	-	1	1	S_IN	X	-	Standby
1	1	-	1	Y	1	1	178	Y	-	Forcierte Position
-	-	-	1	Y	1	1	178	Y	1	Forcierte Position
-	-	-	-	-	1	1	179	Z	-	Restore old position
0	1	X	-	-	0	1	S_IN	X	-	Manual operation
1	0	X	-	-	1	0	S_IN	X	-	Manual operation
0	0	X	-	-	0	0	S_IN	X	-	Manual operation

7.10. BLIND_SHADE

Type Funktionsbaustein

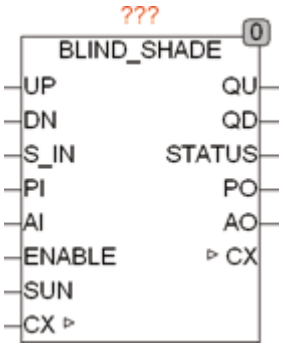
Input UP : BOOL (Eingang AUF)
DN : BOOL (Eingang AB)
S_IN : BYTE (ESR kompatibler Status Eingang)
PI : BYTE (Jalousiestellung im Automatikbetrieb)
AI : BYTE (Lamellenwinkel im Automatikbetrieb)
SUN : BOOL (Eingangssignal vom Sonnensensor)

I/O CX : CALENDAR (aktuelle Zeit und Kalenderdaten)

Setup SUNRISE_OFFSET : TIME (Verzögerung bei Sonnenaufgang)
SUNSET_PRESET : TIME (Verzögerung bei Sonnenuntergang)
DIRECTION : REAL (Fassadenausrichtung, 180° = Südfassade)
ANGLE_OFFSET : REAL (Horizontaler Öffnungswinkel für Beschattung)
SLAT_WIDTH : REAL (Breite der Lamellen in mm)
SLAT_SPACING : REAL (Abstand der Lamellen in mm)
SHADE_DELAY : TIME (Verzögerungszeit der Beschattung)
SHADE_POS : BYTE (Position bei Beschattung)

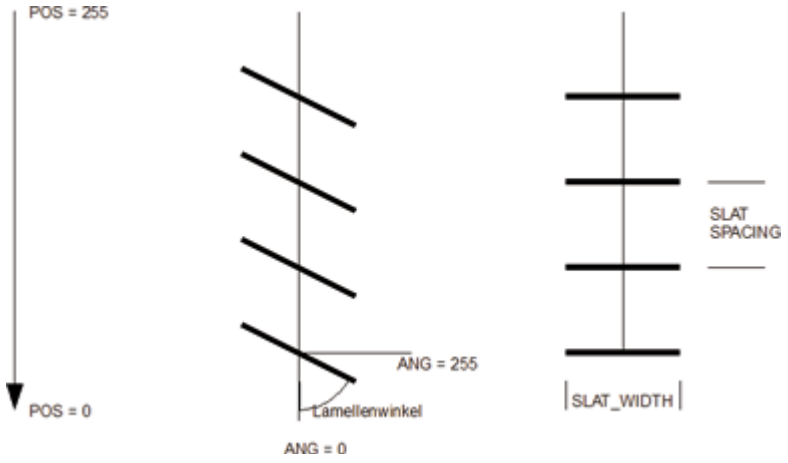
Output QU : BOOL (Motor Auf Signal)
QD : BOOL (Motor Ab Signal)
STATUS : BYTE (ESR kompatibler Status Ausgang)
PO : BYTE (Jalousiestellung im Automatikbetrieb)

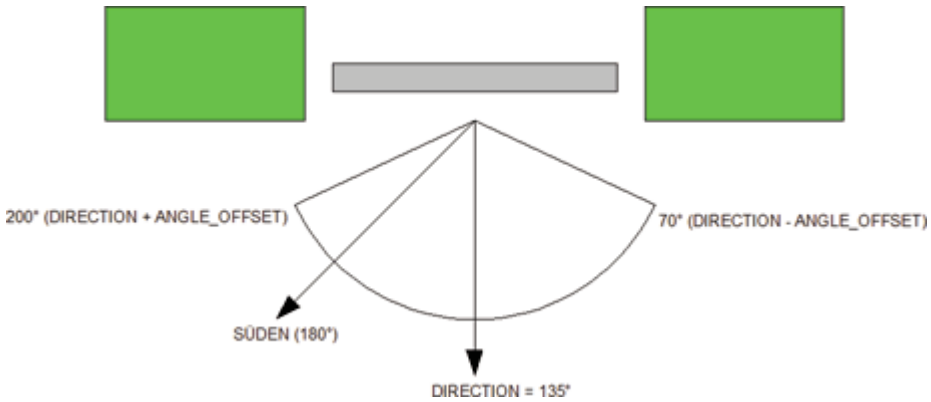
AO : BYTE (Lamellenwinkel im Automatikbetrieb)



BLIND_SHADE berechnet aus dem momentanen Sonnenstand den geeigneten Winkel der Lamellen um eine optimale Beschattung zu gewährleisten. Die Lamellen werden dem Sonnenstand nachgeführt, so dass über den Tagesverlauf immer Beschattung Sichergestellt ist. Mit dem Eingang ENABLE wird die Funktion freigeschaltet wenn gleichzeitig UP und DN (Automatik Modus) aktiv sind. Der Baustein wertet weiterhin den EINGANG SUN aus welcher durch TRUE Sonnenschein anzeigt. Wird SUN oder ENABLE FALSE so schaltet sich der Baustein automatisch ab. SUNRISE_OFFSET definiert mit welchem Zeitversatz nach Sonnenaufgang die Beschattung aktiv wird. SUNSET_PRESET legt fest mit welcher Zeitspanne vor Sonnenuntergang die Beschattung ausgesetzt wird. Die Beschattung ist dann Aktiv wenn SUN = TRUE, ENABLE = TRUE, UP = TRUE, DN = TRUE, der horizontale Sonnenwinkel sich innerhalb des Bereichs DIRECTION - ANGLE_OFFSET und DIRECTION + ANGLE_OFFSET befindet und die Tageszeit sich innerhalb des durch SUNRISE, SUNRISE_OFFSET, SUNSET und SUNSET_PRESET definierten Zeitbereichs befindet. DIRECTION legt die Ausrichtung der Fassade fest, 180° bedeutet Fassade zeigt genau nach Süden, 90° liegt im Osten und 270° im Westen. Mit der Setup Variable SHADE_DELAY wird festgelegt wie lange nachdem SUN auf FALSE geht die Beschattung aktiv bleibt. Der Vorgabewert liegt bei 60 Sekunden. SHADE_DELAY Verhindert das bei Teilbewölkung die Jalousie dauernd auf und ab Fährt. Beim Einsatz von BLIND_SHADE ist darauf zu achten dass die Zykluszeit für den Baustein kleiner als T_ANGLE / 512 * SENS beträgt. SENS ist hierbei der SENS Wert des BLIND_CONTROLLERS. Wird die Zykluszeit zu groß so beginnt die Jalousie unregelmäßig hin und herzufahren. Die Setup Variable BLIND_POS legt fest wie weit die Jalousie bei Beschattung nach unten fährt.

Die folgende Grafik beschreibt die Geometrie der Jalousie:



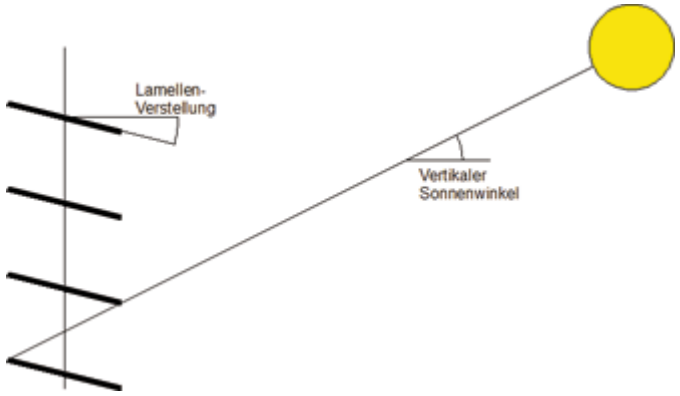


Die folgende Grafik zeigt eine nach Süd-Ost gerichtete Fassade mit DIRECTION = 135°und ANGLE_OFFSET = 65°:

Die Beschattungsfunktion berechnet den Lamellenwinkel so dass die Lamellen immer nur soweit geschlossen werden dass die Sonne abgeschattet wird, aber dennoch soviel Licht wie möglich in den Raum gelangt. Aus den Angaben DIRECTION und ANGLE_OFFSET wird berechnet wann der horizontale Einstrahlwinkel der Sonne eine Beschattung erfordert. Je nach Stärke der Mauer und Breite des Fensters kann der ANGLE_OFFSET so eingestellt werden das unnötige Beschattung vermieden wird. Mit DIRECTION wird die Himmelsrichtung der Fassade angegeben. Mithilfe der Abmessungen der Lamellen, Breite und Abstand in Millimeter (SLAT_WIDTH und SLAT_SPACING) wird berechnet wie weit die Lamellen geneigt werden müssen um die Sonneneinstrahlung zu verhindern. Ziel ist es dabei die Lamellen nur soweit wie unbedingt nötig zu neigen damit optimale Lichtverhältnisse garantiert sind. Um bei Sonnenaufgang und Sonnenuntergang die Stimmung und Lichtverhältnisse nicht zu beeinflussen kann ein OFFSET vom Sonnenaufgang und ein PRESET vor dem Sonnenuntergang eingestellt werden. Mit einen OFFSET von 30 Minuten und einem PRESET von 60 Minuten wird zum Beispiel erst 30 Minuten nach Sonnenaufgang mit der Beschattung begonnen und bereits 60 Minuten vor Sonnenuntergang die Beschattung beendet. Der Eingang SUN des Moduls dient Dazu einen Son-

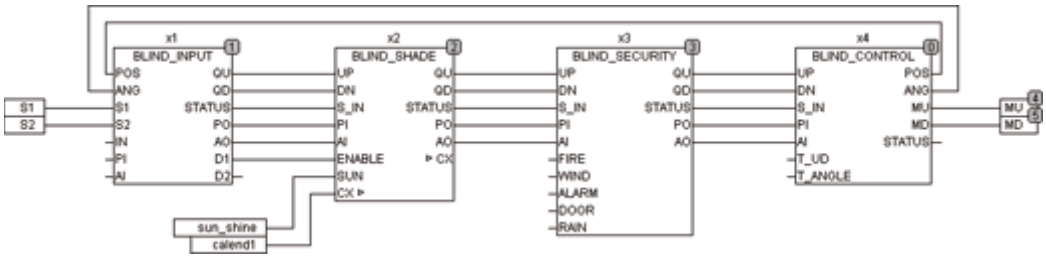
nenintensitätssensor oder einen beliebigen geeigneten Sensor anzuschließen der die Funktion unterbricht wenn keine Sonnenstrahlung vorliegt.

In der Folgenden Grafik wird die Abschattung verdeutlicht:



Der Eingang S_IN und der Ausgang STATUS sind ESR kompatible Aus und Eingänge , über den Eingang S_IN melden vorgeschaltete Funktionen Ihren Status an das Modul, dieser Status wird an den Ausgang STATUS weitergeleitet, und eigene Statusmeldungen werden über STATUS Ausgegeben. BLIND_SHADE meldet am STATUS Ausgang den STATUS 151 wenn die Beschattungsfunktion aktiv ist.

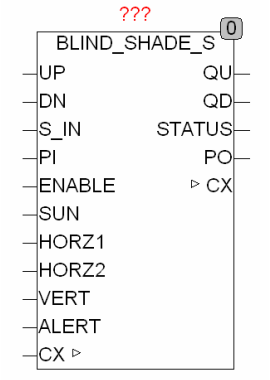
Das folgende Beispiel zeigt die Anwendung von BLIND_SHADE innerhalb einer Jalousiesteuerung:



7.11. BLIND_SHADE_S

Type	Funktionsbaustein
Input	UP : BOOL (Eingang AUF)
	DN : BOOL (Eingang AB)
	S_IN : BYTE (ESR kompatibler Status Eingang)
	PI : BYTE (Vorgabe der Position)
	ENABLE : BOOL (Beschattung aktiviert)

	SUN : BOOL (Eingangssignal vom Sonnensensor)
	HORZ1 : REAL (Horizontaler Sonnenwinkel Beschattungsbeginn) [100.0]
	HORZ2 : REAL (Horizontaler Sonnenwinkel Beschattungsende) [260.0]
	VERT : REAL (Vertikaler Beschattungswinkel) [90.0]
	ALERT : BOOL (Zwangsöffnung des Rollos) [FALSE]
I/O	CX : CALENDAR (aktuelle Zeit und Kalenderdaten)
Setup	SUNRISE_OFFSET : TIME (Delay bei Sonnenaufgang) [T#1h]
	SUNSET_PRESET : TIME (Delay bei Sonnenuntergang) [T#1h]
	SHADE_DELAY : TIME (Delay der Beschattung) [T#60s]
	SHADE_POS : BYTE (Maximalposition bei Beschattung)
Output	QU : BOOL (Motor Auf Signal)
	QD : BOOL (Motor Ab Signal)
	STATUS : BYTE (ESR kompatibler Status Ausgang)
	PO : BYTE (Jalousiestellung im Automatikbetrieb)



BLIND_SHADE_S ist eine wesentlich vereinfachte Funktion von BLIND_SHADE speziell für die Verwendung mit Rollos. Bei diesen muss kein Lamellenwinkel für die Beschattung berechnet werden, sondern lediglich sichergestellt, dass bei Sonnenschein das Rollo weit genug geschlossen ist.

Im inaktiven Zustand reicht der Baustein die Eingänge UP, DN, PI und S_IN unverändert an die Ausgänge QU, QD, PO und STATUS durch.

Der Baustein wird scharfgeschaltet, wenn UP = TRUE, DN = TRUE, ENABLE = TRUE und SUN (für mindestens SHADE_DELAY) = TRUE. Sind diese Bedingungen erfüllt, prüft der Baustein, ob aktuell der horizontale Sonnenwinkel im Bereich zwischen HORZ1 und HORZ2 liegt und der vertikale Sonnenwinkel niedriger als VERT ist. Liegt nun auch noch die aktuelle Uhrzeit zwischen Sonnenaufgang + SUNRISE_OFFSET und Sonnenuntergang –

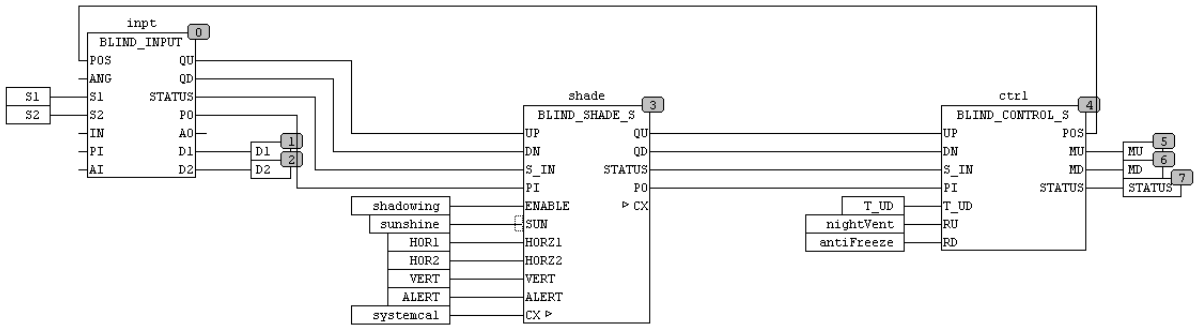
SUNSET_PRESET, dann wechselt der Baustein in den STATUS 151 (Beschattung) und stellt sicher, dass am Ausgang PO kein größerer Wert als SHADE_POS ausgegeben wird (PO ist dann das Minimum aus PI und SHADE_POS).

Für die Winkel HORZ1 und HORZ2 gilt: 90° = Osten, 180° = Süden, 270° = Westen.

SHADE_DELAY verhindert, dass bei Teilbewölkung die Jalousie dauernd auf und ab fährt.

Über den Eingang ALERT, kann auf einfache Weise erreicht werden, dass das Rollo z.B. bei geöffneter Tür ganz nach oben fährt. Der ALERT Eingang hat im Baustein höchste Priorität, erzwingt STATUS = 152 unabhängig von den Eingängen und setzt QU = TRUE, QD = FALSE, führt also einen manual UP durch.

Innerhalb einer Rollosteuerung kann BLIND_SHADE z.B. wie folgt eingesetzt werden:



Verzeichnis der Funktionsbausteine