Technische Dokumentation
PicoLogo®

Version 2.00

Funk-Electronic Piciorgros GmbH
Claudiastr. 5
51149 Köln

1. Allgemeines

PicoLogo® ist eine Software-Micro-SPS für Produkte der Firma Funk-Electronic Piciorgros GmbH. PicoLogo entspricht in Umfang und Art in etwa den Funktionen der Logo!®-SPS der Siemens AG.

PicoLogo ermöglicht den Anwendern, Steuerungsaufgaben vor Ort mit Hilfe der entsprechenden Funk- und GSM-Modems auszuführen. Auf eine zusätzliche SPS kann in diesen Fällen verzichtet werden.

Die Programmierung des PicoLogo-Moduls und die Übertragung des Programms in die Geräte erfolgt durch eine kostenlos verfügbare Windows-Compilersoftware in einer leicht zu erlernenden Programmiersprache. Die Installation und Bedienung der Software sowie eine Beschreibung der verfügbaren Befehle sind Gegenstand dieser Anleitung.

1.1 Haftungsausschluss

Der Inhalt dieser Dokumentation wurde von uns sorgfältig mit der darin beschriebenen Hard- und Software auf Übereinstimmung überprüft. Trotzdem können wir Abweichungen nicht ausschließen, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Eventuell notwendige Korrekturen sind in der jeweils nächsten Ausgabe dieser Dokumentation berücksichtigt.

Für Schäden, die durch den mittelbaren oder unmittelbaren Einsatz der PicoLogo Micro-SPS oder der Konfigurationssoftware entstehen, übernehmen wir keine Haftung.

PicoLogo® ist ein geschütztes Markenzeichen der Funk-Electronic Piciorgros GmbH.

Logo!® ist ein geschütztes Markenzeichen der Siemens AG.

1.1 Softwareversionen

Nachfolgend sind die Änderungen der einzelnen Firmwareversionen aufgeführt.

Version PicoLogo	Version Compiler	Version Doku	Bemerkungen / Änderungen
1.00	1.00	1.00	Erste ausgelieferte Version
1.00	1.10	1.10	• Bugfix: Für den Befehl DELSTO ließ sich der erste Parameter nicht eingeben, dies wurde korrigiert • Unterstützung von Parameterlabeln im Compiler
1.00	1.10	1.10B	• Befehlsreferenz in der Dokumentation in alphabetische Reihenfolge gebracht • Tabellarische Befehlsübersicht in die Dokumentation aufgenommen
1.01	1.20	1.20	• Unterstützung der RTU-700-Baureihe für PicoLogo implementiert • Das Löschen von Programmen im Gerät ist jetzt möglich

2. Installation der Software

PicoLogo wird über die mitgelieferte PicoLogo-Software programmiert. Mit der PicoLogo-Software können Programme erstellt, kompiliert und in das Gerät übertragen werden.

Um die Software auf Ihrem PC zu installieren, legen Sie die CD ein bzw. starten Sie die Installationsroutine "setup.exe". Folgen Sie dem Dialog, um PicoLogo zu installieren.

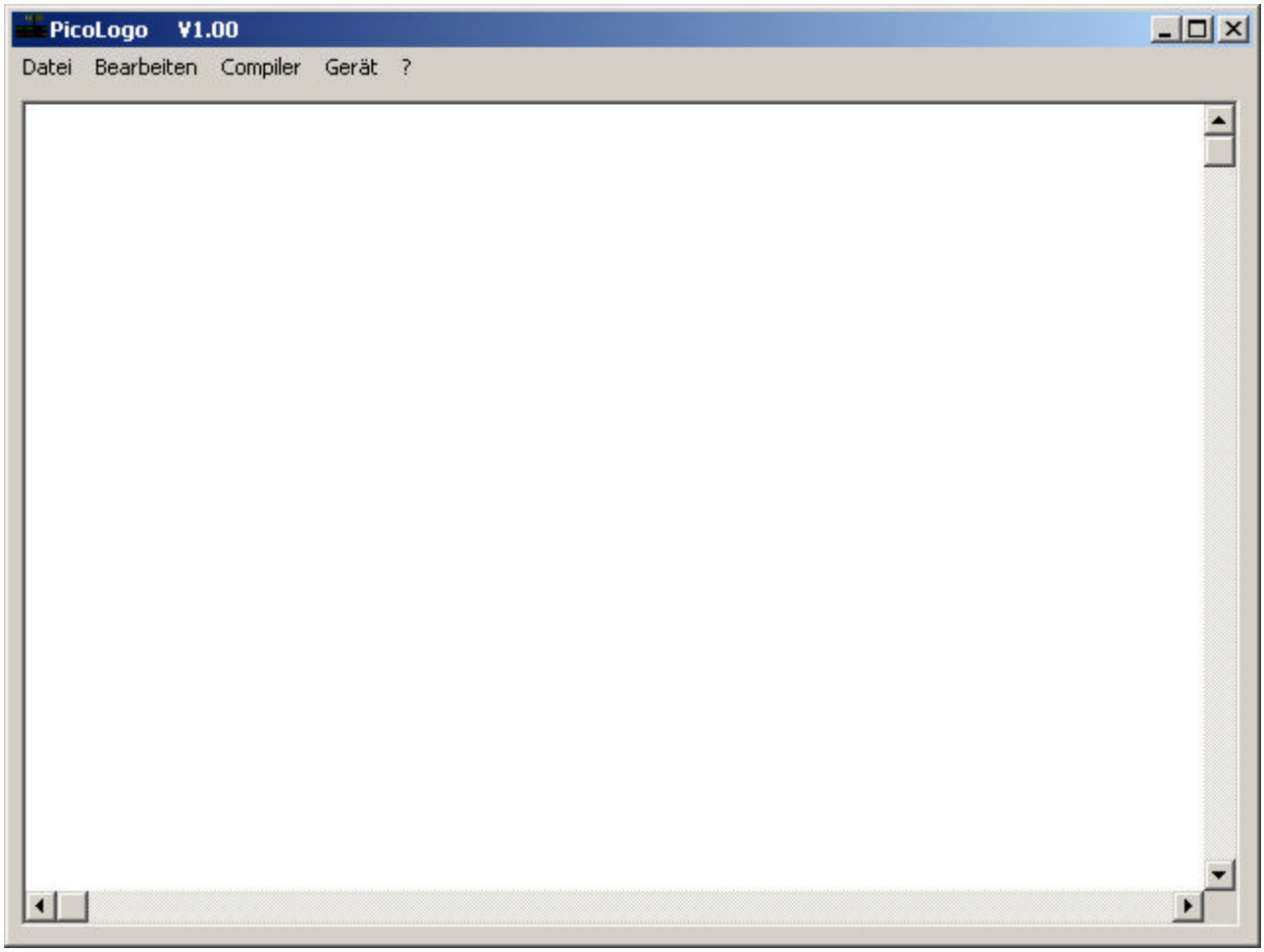
Die Installationsroutine legt einen Starteintrag unter "Start/Programme/PicoLogo" an und hinterlässt ein Icon auf dem Desktop. Über diese Elemente kann das Programm gestartet werden.

Wenn Sie die Software durch eine aktuellere Version ersetzen wollen, so installieren Sie die neue Version einfach über die vorhandene Installation.

3. Bedienhinweise

3.1 Starten des Programms

Die PicoLogo-Software wird über das Startmenü oder das Icon auf dem Desktop gestartet. Nach dem Start erscheint das Hauptfenster:



In dem Textfenster des Programms können nun PicoLogo-Programme erstellt werden. Ein kleines Demo-Programm liegt der Installation bei, und kann über Datei/Öffnen geöffnet werden.

PicoLogo-Programme weisen die Dateiendung ".plo" auf.

3.2 Erstellen eines PicoLogo-Programms

PicoLogo-Programme werden direkt im Textfenster des PicoLogo-Programms erstellt. Im Menüpunkt "Bearbeiten" stehen einige Editierhilfen wie Ausschneiden, Kopieren, Einfügen etc. zur Verfügung.

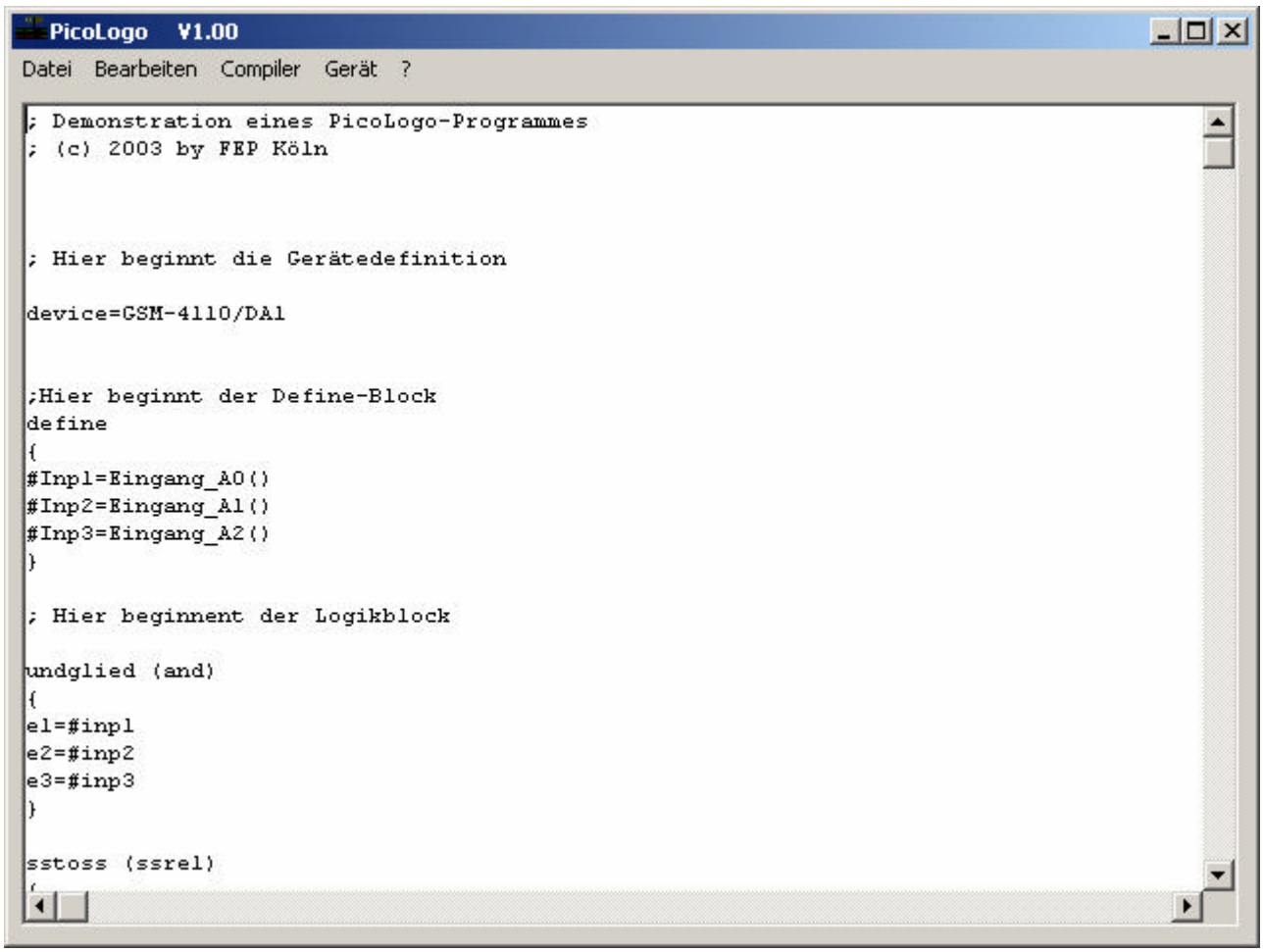
Unter "Bearbeiten/Eingabehilfe" wird das Eingabehilfefenster eingeblendet:

Hier sind 2 Pulldown-Menüs zu finden, über die alle verfügbaren Funktionen ausgewählt werden können. Wird eine Funktion ausgewählt, so wird ein komplett vorbereiteter und kommentierter Funktionsblock an der aktuellen Position des Cursor in das Textfenster eingefügt.

Das Fenster kann außerhalb des Eingabefensters auf den Desktop verschoben werden, so dass es stets erreichbar bleibt.

Über den Button "Nächster Fehler" kann nach einem Compilerlauf jeweils zum nächsten aufgetretenen Fehler gesprungen werden.

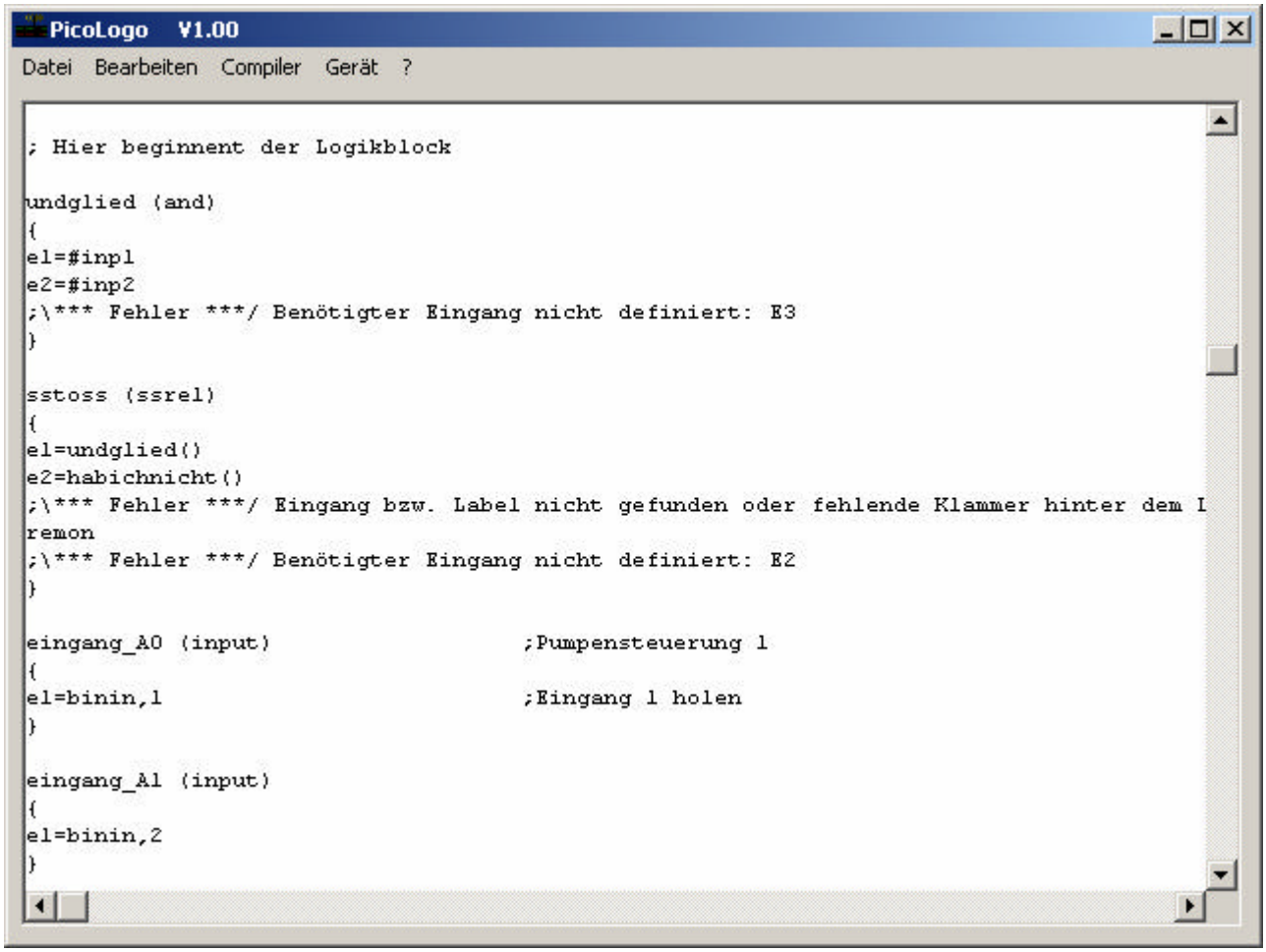
Nachfolgend ein Bild des Eingabefensters mit einem erstellten PicoLogo-Programm:



3.3 Kompilieren des Programms

Wenn das Programm fertig erstellt ist, so muss es kompiliert werden. Dies geschieht durch Aufruf des Menüs "Compiler/Programm kompilieren". Wurde das Programm neu erstellt und noch nicht gespeichert, so fordert das Programm zum Speichern des PicoLogo-Programms auf.

Werden während des Compilerlaufs Fehler entdeckt, so werden diese an den entsprechenden Stellen direkt in das PicoLogo-Programm eingeblendet:



Sie können über den Menüpunkt "Compiler/Nächster Fehler" alle Fehler im Programm direkt anspringen. Nach Aufruf des Befehls wird der nächste verfügbare Fehler nach der Cursorposition in das Eingabefenster in Zeile 6 gerückt – es werden also immer die 5 dem Fehler vorangehenden Zeilen mit dargestellt. Den gleichen Effekt hat der Button "Nächster Fehler" in dem Eingabehilfe-Fenster.

Der Befehl "Compiler/Fehlermeldungen entfernen" löscht alle Fehlermeldungen aus Ihrem Programm. Das Gleiche passiert automatisch, wenn Sie einen neuen Compilerlauf starten.

Wurde der Compilerlauf ohne Fehler abgeschlossen, so wird eine entsprechende Meldung eingeblendet:

Die Software hat nun eine Datei mit gleichem Namen wie das PicoLogo-Programm, jedoch mit der Endung ".plr" erzeugt. Diese Datei kann nun jederzeit in Geräte mit PicoLogo-Funktion übertragen werden.

3.3.1 Programmoptimierung

Der Compiler optimiert die erstellten PicoLogo-Programme, so dass Sie innerhalb des Gerätes mit höchstmöglicher Performance ausgeführt werden. Wenn ein PicoLogo-Programm ungünstig erstellt wurde, so kann es sein, dass es mehrere Zyklen benötigt, um alle Funktionen einer Kette durchlaufen zu können. Wenn z.B. ein Eingang, auf den mehrere Funktionsblöcke zugreifen müssen, erst am Ende des PicoLogo-Programms definiert ist, so würde die aktuelle Information dieses Funktionsblocks erst im nächsten Zyklus für die vorhergehenden Blöcke zur Verfügung stehen.

Der PicoLogo-Compiler optimiert die Programme jedoch so, dass sich eine optimale Reihenfolge ergibt. Die Optimierung kann über die erzeugte Protokolldatei mit der Endung ".lst" betrachtet werden.

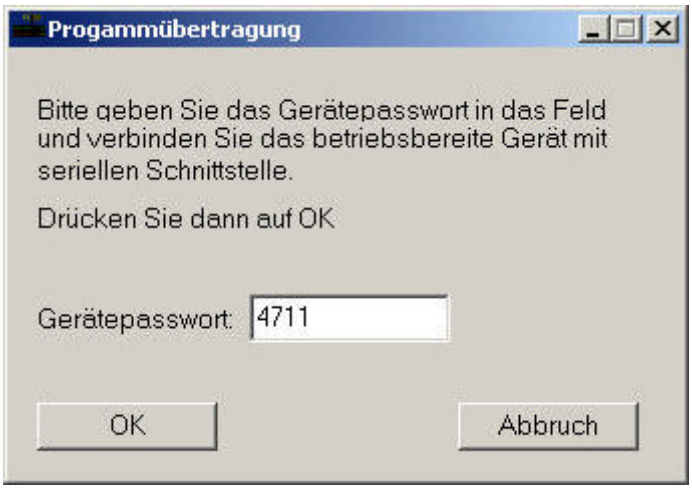
3.4 Übertragen des Programms in das Gerät

Ein kompiliertes PicoLogo-Programm muss nun noch in das entsprechende Gerät übertragen werden. Hierzu stehen unter dem Menüpunkt "Gerät" zwei Befehle zur Verfügung.

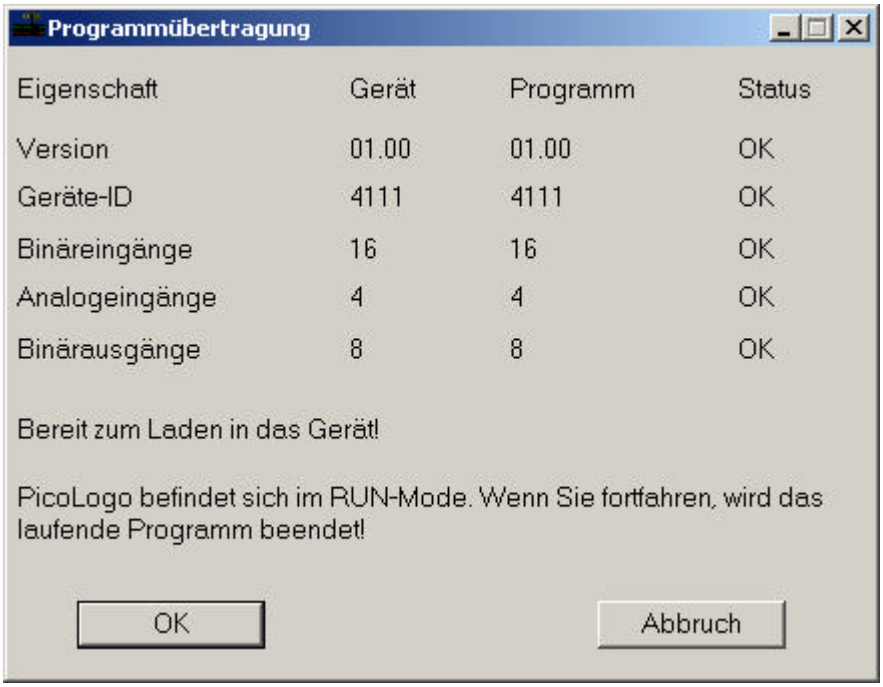
"Programm übertragen" überträgt das aktuell kompilierte Programm in das bzw. die Geräte. Dies funktioniert allerdings nur, wenn das im Textfenster befindliche Programm zuvor kompiliert wurde. Ist dies nicht der Fall oder wurde das Programm nach dem kompilieren geändert, so erscheint ein entsprechender Hinweis, dass das Programm nicht übertragen werden kann. In diesem Fall muss das Programm aus der entsprechenden .plr-Datei übertragen werden.

"Programm aus Datei übertragen" überträgt ein beliebiges bereits kompiliertes Programm in ein Gerät. Nach dem Aufruf des Befehls müssen Sie die .plr-Datei angeben, die übertragen werden soll.

Anschließend erscheint ein Hinweisfenster, dass das Gerät an die serielle Schnittstelle des PC anzuschließen ist. Bei GSM-4110-Geräten muss hier zusätzlich das Gerätepasswort eingegeben werden, das den Zugriff auf das Gerät ermöglicht:



Nach dem Bestätigen dieses Dialoges werden die Betriebsdaten aus dem Programm ausgelesen. Diese werden anschließend angezeigt:



Ein Übertragen der Daten in das Gerät ist nur möglich, wenn folgende Bedingungen erfüllt sind:

- Die PicoLogo-Version des Gerätes muss mindestens die Version aufweisen, die das Programm verlangt. Der Versionsbedarf des Programms richtet sich nach den verwendeten Funktionsblöcken: Kommen in späteren Versionen einmal Funktionsblöcke hinzu, so können diese nur in Geräten verwendet werden, die diese Blöcke in ihrer Firmware auch unterstützen.

- Geräte-ID: Das Zielgerät muss exakt dem Gerät entsprechen, für welches das Programm kompiliert wurde (wird im "device"-Befehl im PicoLogo-Programm festgelegt).

- Ein- und Ausgänge: Die Zahl der verfügbaren Ein- und Ausgänge muss ebenfalls exakt mit der Vorgabe des Programms übereinstimmen. Wurde ein Gerät (z.B. RTU-Baureihe) mit Erweiterungsmodulen erweitert, so muss dies im PicoLogo-Programm über die entsprechenden Befehle (z.B. "binin=32") angegeben werden.

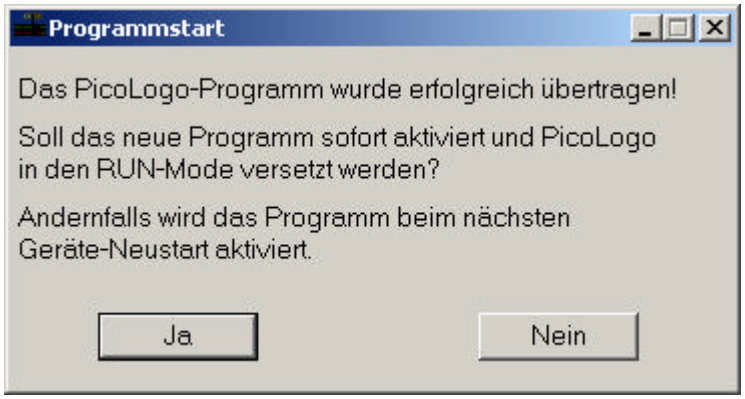
Sind alle Parameter übereinstimmend, so kann das Programm nun in das Gerät geladen werden. Hierzu ist der Button "OK" anzuklicken. Dieser Button steht nicht zur Verfügung, wenn im o.a. Dialog Fehler entdeckt wurden.

Wenn sich PicoLogo zum Zeitpunkt der Programmübertragung im RUN-Mode befindet und ein Programm ausgeführt wird, so erscheint ein Hinweis, dass das Programm beim Fortfahren mit der Programmierung beendet wird.

Bitte beachten Sie, dass ein Anklicken des OK-Buttons das PicoLogo zwangsläufig in den STOP-Mode bringt. Ein laufendes Programm wird jetzt beendet und aus dem Speicher des Gerätes gelöscht. Ein PicoLogo-Programm kann im Gerät nun erst wieder ausgeführt werden, wenn es erfolgreich in das Gerät übertragen wurde.

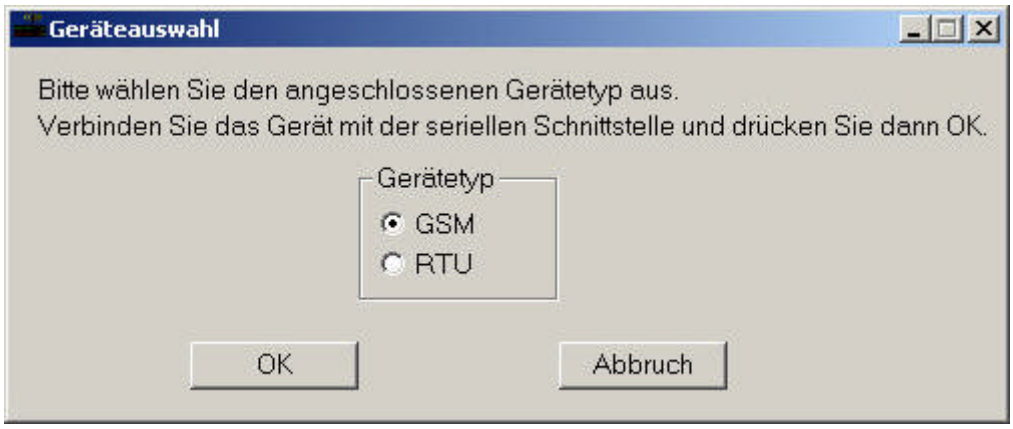
Der Fortschritt der Übertragung in das Gerät wird über eine Fortschrittsanzeige angezeigt:

Nach Beendigung der Übertragung fragt das Programm, ob das geladene Programm sofort initialisiert und gestartet werden soll. Wird dieser Dialog mit "Nein" bestätigt, so wird das geladene Programm erst nach dem nächsten Geräte-Neustart aktiviert!

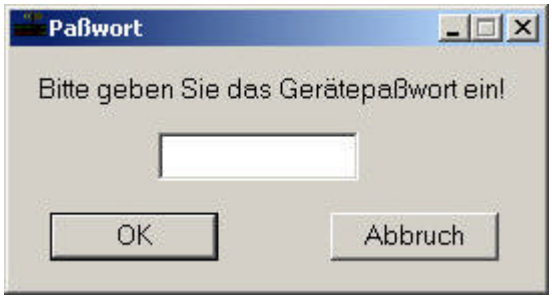


3.5 Löschen eines Programms in einem Gerät

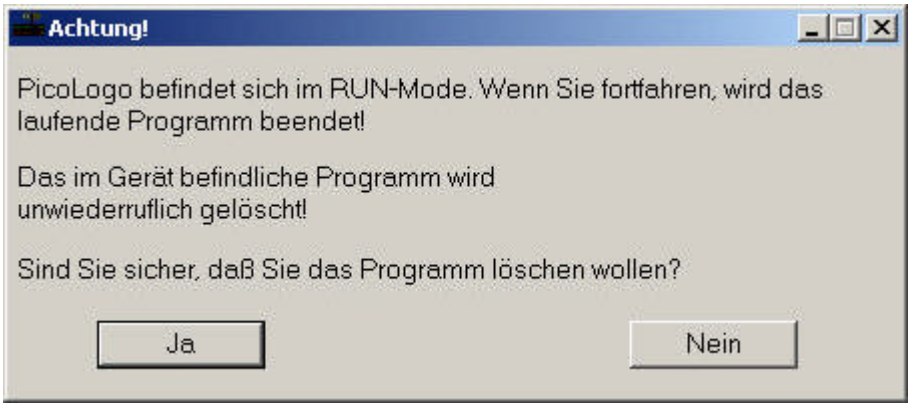
Soll ein PicoLogo-Programm aus einem Gerät gelöscht werden, so ist das mit der Compiler-Software ab Version 1.20 möglich. Hierzu ist das Gerät an den PC anzuschließen und der Menüpunkt "Gerät / Programm im Gerät löschen" aufzurufen. Anschließend erwartet das Programm die Angabe der Gerätefamilie:



Handelt es sich um ein GSM-Gerät, so wird nun das Gerätepasswort abgefragt:



Die nachfolgende Sicherheitsabfrage muß zum Löschen des Programms mit "Ja" bestätigt werden:

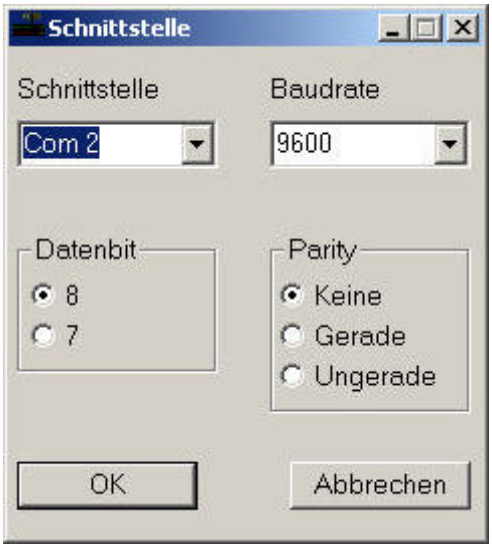


Nach der Bestätigung der Sicherheitsabfrage wird das Programm nun im Gerät gelöscht:

Wenn das Programm erfolgreich gelöscht wurde, so wird dies am Ende des Vorgangs bestätigt:

3.6 Schnittstelleneinstellung

Unter "Gerät/Schnittstelle" können Sie die Schnittstellenparameter konfigurieren, die das PicoLogo-Programm zur Übertragung in das Gerät nutzt.



Unter "Schnittstelle" ist die serielle Schnittstelle auszuwählen, an der das Gerät an den PC angeschlossen ist. Baudrate, Datenbit und Parität sind so auszuwählen, dass diese den Einstellungen des Zielgerätes entsprechen.

Das PicoLogo-Programm merkt sich die vorgenommenen Einstellungen.

3.7 Sicherheit

PicoLogo weist mehrere Mechanismen zur Überwachung des Programms im Gerät auf. Während der Übertragung in das Gerät werden die übertragenen Daten vollständig wieder aus dem Gerät ausgelesen und verglichen. Treten hierbei Fehler auf (Übertragungsfehler), so bricht die Übertragung mit einer entsprechenden Fehlermeldung ab.

Die vom Compiler erzeugten PicoLogo-Programme sind prüfsummengesichert. Entspricht das Programm im Gerät nicht der ursprünglich erzeugten Prüfsumme (ROM-Fehler), so wird PicoLogo in den STOP-Mode versetzt und kann nicht mehr gestartet werden, ohne das Programm neu in das Gerät zu übertragen.

Treten zur Laufzeit Veränderungen am PicoLogo-Programm auf (RAM-Fehler), so werden alle Vorgänge im Zielgerät sofort beendet und das Gerät durchläuft einen Hard-Reset.

Geht PicoLogo in den STOP-Mode, so werden alle von PicoLogo benutzten Binärausgänge in den sicheren Zustand gebracht (deaktiviert). Da bei einem RAM-Fehler die Konsistenz des PicoLogo-Programms nicht mehr gewährleistet sein kann, geschieht dies in diesem Fall über einen Reset des Gerätes.

4. Aufbau eines PicoLogo-Programms

4.1 Allgemeines

Ein PicoLogo-Programm ist besteht grundsätzlich aus 3 Sektionen: Der Gerätedefinition, dem Definitionsblock für Platzhalter und dem eigentlichen Programm, welches sich aus einzelnen Funktionsblöcken zusammensetzt.

In dem Abschnitt der Gerätedefinition wird das Gerät spezifiziert, auf dem das PicoLogo-Programm laufen soll.

Im Definitionsblock für Platzhalter können globale Definitionen vorgenommen werden. Auf diese Art und Weise können numerische Konstanten, Namen von Funktionsblöcken etc. am Programmanfang einem Namen (Platzhalter) zugewiesen werden . Dieser Platzhalter kann nun beliebig im Programm verwendet werden. Wann immer ein Platzhalter im Programm entdeckt wird, so wird der Platzhalter durch den definierten Text ersetzt. So können Werte und Referenzen, die mehrfach identisch im Programm vorkommen, durch einen Platzhalter ersetzt werden. Bei einer Änderung braucht dann nur die Platzhalterdefinition geändert zu werden.

Im eigentlichen Programmabschnitt werden die einzelnen Funktionsblöcke, aus denen das PicoLogo-Programm besteht, angegeben.

Es können an beliebiger Stelle im Programm Kommentarzeilen gesetzt werden. Kommentarzeilen dienen der Programmdokumentation und werden vom Compiler nicht beachtet. Sie beginnen immer mit einem Semikolon (";").

Ist ein Semikolon innerhalb einer Programmzeile zu finden, so wird der Text ab dem Semikolon nicht mehr beachtet.

Groß- und Kleinschreibung wird grundsätzlich nicht unterschieden. So ist "Fuellstand" das Gleiche wie "fuellstand" oder "FUELLSTAND". Dies ist insbesondere bei der Vergabe von Labels zu beachten!

4.2 Definition des Endgerätes

PicoLogo ist so konzipiert, dass es seine Funktionen auf unterschiedlichen Gerätetypen und auch auf eventuell angeschlossenen I/O-Erweiterungsmodulen ausführen kann. Hierzu ist es allerdings nötig, dass der PicoLogo-Compiler Informationen über das System erhält, auf dem das Programm laufen soll.

Hierzu muss am Beginn eines Programms der verwendete Gerätetyp angegeben werden. Hierzu dient der "DEVICE"-Befehl:

```
DEVICE=GSM-4110/DA1
```

Folgende Geräte werden z.Zt. unterstützt und können nach dem Gleichheitszeichen angegeben werden:

- GSM-4110/DA1
- GSM-4110/DA2
- GSM-4110/DA3
- RTU-700/PL1
- RTU-700/PL2
- RTU-700/PL3

PicoLogo geht nach der Definition des Gerätes davon aus, dass es sich um Grundgerät mit den fest implementierten I/O handelt. Die Ein- und Ausgänge einiger Gerätefamilien können jedoch per Erweiterungsmodule ausgebaut werden. Daher kann die Anzahl der jeweiligen Ein- und Ausgänge durch folgende Befehle angegeben werden, wenn sie von dem Grundgerät abweicht:

```
BININ=x
ANAIN=x
BINOUT=x
ANAOUT=x
```

Der Befehl "BININ" weist die tatsächlich vorhandenen Binäreingänge zu, der Befehl "ANAIN" die Anzahl der tatsächlich vorhandenen Analogeingänge. "BINOUT" gibt die Anzahl der vorhandenen Binärausgänge an, und "ANAOUT" die Anzahl der vorhandenen Analogausgänge. Für x ist die jeweilige Zahl einzusetzen.

Es ist zu beachten, dass diese Sonderzuweisungen **nach** dem Device-Befehl stehen müssen. Es müssen nur die Sonderzuweisungen der Ein-/Ausgänge angegeben werden, die von dem definierten Grundgerät abweichen.

4.3 Definition von Platzhaltern

Platzhalter sind Textbausteine, die einen Text an beliebigen Stellen im Programm durch einen anderen Text ersetzen können. Dies ist besonders dann sinnvoll, wenn ein bestimmter Wert oder Verweis auf einen Block mehrfach im Programm vorkommt. Man kann hierfür dann einen Platzhalter definieren und an den entsprechenden Stellen im Programm einsetzen. Änderungen müssen dann nur noch einmal in dem Platzhalterabschnitt erfolgen, und nicht mehrfach im Programm.

Der Platzhalterabschnitt beginnt mit dem Befehl "DEFINE". Dieser wird von einem Parameterblock gefolgt, der durch eine einzelne offene geschweifte Klammer in einer Zeile eingeleitet und durch eine einzelne geschlossenen geschweifte Klammer in einer Zeile beendet wird. Zwischen den beiden Klammern werden dann die einzelnen Platzhalter definiert. Ein Platzhalter besteht immer aus einem Namen, der durch ein Rautezeichen ("#") eingeleitet wird. Der Name darf ausschließlich Buchstaben, Ziffern sowie den Unterstrich (_) enthalten. Im folgenden Beispiel wird die Definition eines Platzhalters dargestellt:

```
DEFINE
{
#Pumpe1=Eing_Pumpe1()
#Einschaltverzoegerung=17
}
```

Es werden hier zwei Platzhalter angelegt: Ein Platzhalter mit dem Namen "#Pumpe1" und ein Platzhalter mit dem Namen "#Einschaltverzoegerung".

Die Platzhalter können nun weiter unten im Programm verwendet werden. Gehen wir davon aus, dass es im Programm einen INPUT-Block gibt, der den Namen "Eing_Pumpe1" hat:

```
Eing_Pumpe1 (INPUT)
{
E1=BININ(1)
}
```

Wenn nun ein Funktionsblock auf diesen Eingangsblock referenzieren soll, so kann dafür der Platzhalter verwendet werden. An Stelle von...

```
Inverter (INV)
{
E1=Eing_Pumpe1()
}
```

... kann nun folgendes eingesetzt werden:

```
Inverter (INV)
{
E1=#Pumpe1
}
```

Sinnvoll ist dies, wenn mehrere Blöcke die gleiche Eingangsreferenz aufweisen. Sollen alle diese Blöcke durch eine Änderung auf einen anderen Funktionsblock verweisen, so ist dies lediglich in der Platzhalterzuweisung zu ändern.

Auch bei Konstanten kann dieses Verfahren angewendet werden: Wenn im Programm mehrere Einschaltverzögerungen verwendet werden, die eine Verzögerungszeit von 17 Sekunden besitzen, so kann an Stelle von...

```
EV_Pumpe7 (DELON)
{
E1=Eingangsreferenz()
PAR1=17
}
```

folgendes geschrieben werden:

```
EV_Pumpe7 (DELON)
{
E1=Eingangsreferenz()
PAR1=#Einschaltverzoegerung
}
```

wenn diese Zeit einmal für alle Blöcke geändert werden soll, so muss lediglich an einer Stelle die Zuweisung des Platzhalters geändert werden.

Wichtige Hinweise:
Es können maximal 256 Platzhalter definiert werden. Platzhalternamen dürfen nicht doppelt vergeben werden. Platzhalter können im Programm ausschließlich im Parameterbereich eines Funktionsblocks (der durch die geschweiften Klammern eingeschlossen ist) eingesetzt werden.

4.4 Befehlssyntax

PicoLogo besteht aus eigenen sogenannten Funktionsblöcken. Jeder Funktionsblock führt völlig unabhängig von den anderen Funktionsblöcken eine bestimmte Funktion aus, die durch das Befehlswort des Funktionsblocks vorgegeben wird. Ein Funktionsblock weist immer einen oder mehrere Eingänge und genau einen Ausgang auf. Einige Funktionsblöcke verlangen zusätzlich die Eingabe von Parametern.

Jeder Eingang eines Funktionsblocks verweist in der Regel auf den Ausgang eines anderen Funktionsblocks. Eine Ausnahme hiervon ist der INPUT-Block, dessen Eingang auf einen beliebigen Binäreingang des Gerätes verweist und dessen Wert einliest.

Generell weisen alle Funktionsblöcke, die der PicoLogo-Compiler verarbeitet, die gleiche Syntax auf. Sie beginnen alle mit dem sogenannten Label, das ist ein vom Benutzer frei zu vergebender Name für den Funktionsblock. Label müssen eindeutig sein und dürfen daher nur einmal vergeben werden. Ebenso dürfen Label keine Leerzeichen beinhalten und müssen stets mit einem Buchstaben beginnen. Das Label wird von dem Typ des Funktionsblocks gefolgt, welcher stets in Klammern gesetzt wird:

```
Fuellstand (INPUT)
```

definiert z.B. einen Funktionsblock mit dem Namen "Fuellstand" als INPUT-Block, welcher die Aufgabe hat, einen Binäreingang einzulesen.

Jeder Funktionsblock benötigt nun eine Reihe von Angaben (Parameterbereich) die definieren, welche Signale er verarbeiten soll. Welche Angaben jeder Block benötigt, ist detailliert in der Beschreibung des jeweiligen Befehls zu finden. Allen gemeinsam ist, dass jeder Parameterbereich mit einer einzelnen geschweiften Klammer in der nächsten Zeile beginnt und nach Angabe aller erforderlichen Parameter mit einer einzelnen geschlossenen, geschweiften Klammer in einer Zeile endet:

```
Fuellstand (INPUT)
{
.....
}
```

Wir vervollständigen den Funktionsblock "INPUT" in unserem Beispiel nun, in dem wir dem Funktionsblock den benötigten Parameter mitgeben, der definiert, welcher Binäreingang eingelesen werden soll:

```
Fuellstand (INPUT)
{
E1=BININ,2
}
```

Der Funktionsblock ist nun komplett. Dieser Funktionsblock mit dem Namen "Fuellstand" liest den 2. Binäreingang des Gerätes ein (Port A1). Weitere Parameter benötigt der Funktionsblock nicht.

Für die internen Eingänge des jeweiligen Gerätes (nicht die Eingänge eventuell angeschlossener Erweiterungsmodule) kann optional auch der Name des jeweiligen Eingangs direkt in eckigen Klammern angegeben werden, für obiges Beispiel würde das bedeuten:

```
Fuellstand (INPUT)
{
E1=BININ[A2]
}
```

Der Funktionsblock "INPUT" ist der Einzige Block, der die Binäreingänge des Gerätes einlesen kann. Wenn in einem PicoLogo-Programm Binäreingänge verwendet werden sollen, so müssen diese erst über entsprechende INPUT-Blöcke eingelesen werden. Diese INPUT-Blöcke können dann als Eingangsreferenz für beliebige weitere Blöcke verwendet werden.

Generell gilt: Der Ausgang eines jeden Funktionsblocks kann in beliebig vielen Funktionsblöcken als Eingang verwendet werden!

In nachfolgendem Beispiel wollen wir ein kleines Programm schreiben, welches unseren Eingang A1 einliest, invertiert, und auf Port C0 (erster Ausgang am Gerät) wieder ausgibt. Nachdem der Eingang ja bereits eingelesen wird, muss er jetzt invertiert werden. Hierzu muss folgender Befehl hinzugefügt werden:

```
Inv_Fuellstand (INV)
{
E1=Fuellstand()
}
```

Dieser Funktionsblock stellt an seinem Ausgang den invertierten Wert des Funktionsblockes "Fuellstand" dar. Wenn ein Eingang auf einen anderen Funktionsblock verweist, so muss – wie in diesem Beispiel verdeutlicht – den Name des Funktionsblocks gefolgt von einer offenen und geschlossenen Klammer "()" verwendet werden.

Zum Schluss soll das Signal jetzt an Ausgang C0 (erster Ausgang des Gerätes) ausgegeben werden. Analog zum INPUT-Block müssen wir hierfür einen OUTPUT-Block verwenden. Ein Ausgangsblock benötigt die Angabe eines Eingangs (welcher Block das auszugebende Signal bereitstellt) und eines Ausgangs (auf dem das Signal ausgegeben wird). Ein Ausgang wird immer mit dem Parameter "Q" angegeben:

```
Fuell_Out (OUTPUT)
{
E1=Inv_Fuellstand
Q=BINOUT,1
}
```

Auch für die Ausgänge gilt: Sofern ein interner Ausgang des Gerätes angesprochen werden soll, kann direkt der Name des jeweiligen Ausgangs in eckigen Klammern angegeben werden, die Definitionszeile des Ausgangs kann also ebenso wie folgt dargestellt werden:

```
Q=BINOUT,[C0]
```

Auf der nachfolgenden Seite ist ein komplettes Beispielprogramm dargestellt, welches die Eingänge A1 und A2 einliest, den Eingang A1 invertiert, den invertierten Eingang A1 mit dem Eingang A2 UND-Verknüpft und das Ergebnis dann mit einer Einschaltverzögerung von 5 Sekunden auf dem Ausgang C0 ausgibt.

```
;PicoLogo Demoprogramm 1

DEVICE=GSM-4110/DA1          ;Definition des Gerätes

DEFINE                       ;Platzhalterdefinition
{
#EMV-Zeit=5
{

Eing_A1 (INPUT)              ;Binäreingang A1 holen
{
E1=BININ,2
}

Eing_A2 (INPUT)              ;Binäreingang A2 holen
{
E1=[A2]
}

INV_A1 (INV)                 ;Eingang A1 invertieren
{
E1=Eing_A1()
}

UND_A1_A2 (AND)              ;Eingänge UND-Verknüpfen
{
E1=INV_A1()                  ;A1 wird invertiert verwendet
E2=Eing_A2()                 ;A2 direkt...
E3=1                         ;Leerer Eingang mit 1 belegen
}

EV_A1A2 (DELON)              ;Einschaltverzögerung
{
E1=UND_A1_A2()               ;... aus dem Ausgang des UND-Glieds
PAR1=#EMV-Zeit               ;mit der als Platzhalter definierten Zeit
}

OUT_EV (OUTPUT)              ;Ergebnisausgabe
{
E1=EV_A1A2()                 ;Ausgang der Einschaltverzögerung
Q=BINOUT,1                   ;Auf den Ausgang C0 legen
}
```

4.5 Remanenzen

Einige Funktionsblöcke können remanent sein. Dies bedeutet, dass ihr aktueller Zustand stets stromausfallsicher zwischengespeichert wird. Nach einem Spannungsausfall haben diese Funktionsblöcke den gleichen Betriebszustand wie vor dem Spannungsausfall.

PicoLogo kann bis zu 255 Remanenzeinträge verwalten. Remanenzen sind für folgende Funktionen verfügbar und müssen im jeweiligen Funktionsblock aktiviert werden:

- Selbsthalterelais: Zustand des Ausgangs (1 Remanenzeintrag)
- Stromstoßrelais: Zustand des Ausgangs (1 Remanenzeintrag)
- Vor-/Rückwärtszähler: Zählerstand (Ausgang wird ständig aus dem Zählerstand gebildet) (1 Remanenzeintrag)
- Betriebsstundenzähler: Zählerstand (gesamte Betriebsstunden), Sekundenzähler (zur Stundenbildung), Komparator-Stundenzähler (3 Remanenzeinträge)
- Wochenschaltuhr: Zustand des Ausgangs (1 Remanenzeintrag)
- Jahresschaltuhr: Zustand des Ausgangs (1 Remanenzeintrag)

In den jeweiligen Blöcken wird die Remanenz durch den Befehl "REMON" aktiviert.

4.6 Virtuelle Ein- und Ausgänge

Um von außen auf ein PicoLogo-Programm einwirken zu können, ohne Binärein- und Ausgänge zu verwendet, besitzt PicoLogo 16 virtuelle Eingänge sowie 16 virtuelle Ausgänge. Diese Ein- und Ausgänge liegen jeweils in einem Register, die per Zugriff auf das Gerät (bei GSM per GMop oder MMI-Befehl, bei RTU über MoP) erreicht werden können.

Die virtuellen Eingänge besitzen die Bezeichnung I0...I15. Sie liegen beim GSM-4110 im Register 59, bei der RTU-700 im Register 99. Der virtuelle Eingang I0 ist hier Bit 0 zugeordnet, der Eingang I15 Bit 15.

Diese Eingänge können nun innerhalb des Programms wie normale Binäreingänge angesprochen werden:

```
Eing_A2 (INPUT)              ;Binäreingang A2 holen
{
E1=BININ[I0]                ;Virtuellen Binäreingang einlesen
}
```

Sobald über die o.a. Zugriffsmöglichkeiten das Bit 0 im virtuellen Eingangsregister gesetzt ist, ist der Ausgang des Funktionsblocks aktiv.

Die virtuellen Ausgänge besitzen die Bezeichnung O0...O15. Sie liegen beim GSM-4110 im Register 324, bei der RTU-700 im Register 399. Der virtuelle Ausgang O0 ist hier Bit 0 zugeordnet, der Ausgang O15 Bit 15.

Die virtuellen Ausgänge werden innerhalb eines PicoLogo-Programms wie reale Ausgänge angesprochen.

```
Eing_A2 (INPUT)              ;Binäreingang A2 holen
{
E1=Eingang1()
Q=BINOUT[O3]
}
```

Sobald der Eingang des o.a. Funktionsblocks logisch "1" wird, wird das Bit 3 im virtuellen Ausgangsregister ebenfalls "1".

4.7 Parameter-Label

Für zukünftige PicoLogo-Versionen ist geplant, bis zu 64 Parameter im Programm ändern zu können, ohne das Programm neu zu kompilieren.

Der Compiler bietet jetzt schon Unterstützung für dieses Feature. Hierzu müssen die Parameter, die im laufenden Betrieb geändert werden sollen, mit einem eindeutigen Namen versehen werden. Dies geschieht durch Einfügen dieses Label in Klammern hinter der Parameterbezeichnung:

```
Par1(Labelname)=4711
```

Parameterlabel sind generell für alle Parameter par1...par9 und in allen Funktionsblocks zulässig. Parameterlabel dürfen nicht doppelt verwendet werden und nicht mehr als 16 Zeichen lang sein. Es können programmweit maximal 64 Parameterlabel vergeben werden..

4.8 Systemgrenzen

Die Grenzen der PicoLogo-Programme werden im wesentlichen durch 2 Werte bestimmt: Die Anzahl Funktionsblöcke, die PicoLogo für das jeweilige Gerät verwalten kann, und die Anzahl der Datenworte, die das Programm belegt.

Jeder Funktionsblock belegt immer eine Funktionsblockressource, kann aber je nach Funktion eine unterschiedliche Anzahl Worte im Datenspeicher belegen. Wie viele Worte jeder Funktionsblock benötigt, ist in der nachfolgenden Befehlsreferenz für jeden Funktionsblock aufgezeigt.

Beim GSM-4110 gelten folgende Grenzen:

- Maximal 192 Funktionsblöcke
- Maximal 800 Datenworte Speicherplatz

5. Befehlsreferenz

Auf den nachfolgenden Seiten sind alle PicoLogo-Befehle ausführlich dargestellt. Neben einer ausführlichen Beschreibung des jeweiligen Funktionsblocks finden sich auch Angaben wie Speicherplatzbedarf, Anzahl der Eingänge und Parameter sowie die PicoLogo-Version, ab der dieser Befehl verfügbar ist.

(AAD) Addierender Schwellwertschalter Analog

Speicherbedarf:	3 Worte	Ab Version:	1.10
Eingänge:	3	Parameter:	1

Funktionsweise:
Der Schwellwertschalter addiert den Wert des Analogeingangs E1 mit dem Wert des Analogeingangs E2 und subtrahiert davon den Wert des Analogeingangs E3. Dieser Wert wird mit fest vorgegebenen Werten verglichen. Ist das Ergebnis der Analogeingänge größer als die über PAR1 festgelegte Einschaltschwelle, so wird der Ausgang des Funktionsblocks aktiv. Fällt der Wert unter die über PAR2 festgelegte Ausschaltschwelle, so wird der Ausgang wieder deaktiviert. Ein Unterschreiten der Ausschaltschwelle hat Vorrang.

Die Analogparameter können skaliert angegeben werden – hierbei ist eine Umrechnung des echten Messwertes in die 12-Bit-Auflösung der Analogeingänge nicht nötig. Hinter dem Parameter können – durch Kommata getrennt – der Minimalwert (welcher 0mA entspricht) und der Maximalwert (welcher 20mA entspricht) angegeben werden. Optional ist auch noch die Angabe der Maßeinheit möglich. Das Format ist wie folgt:

```
PARx=Schwellwert,Minwert,Maxwert,Einheit
```

Es ist zu beachten, dass die 3 Analogeingänge nicht unterschiedlich skaliert werden können.

Minwert und Maxwert können jeweils im Bereich von -32767 bis +32768 liegen. Es werden 4 Nachkommastellen berücksichtigt. Der Name der Einheit darf maximal 7 Zeichen lang sein. Werden Minwert und Maxwert nicht angegeben, so muss der Schwellwert zwischen 0 und 4095 liegen.

Syntax:
```
Label (AAD)
{
E1=ANAIN,1
E2=ANAIN,2
E3=ANAIN,3
PAR1=6.35,0,11.5,Meter
PAR2=3.75,0,11.5,Meter
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(ASS)	Befehlscode für den Block
E1	Spezifiziert den ersten Analogeingang. Die Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
E2	Spezifiziert den zweiten Analogeingang, welcher zu E1 hinzuaddiert wird. Die Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
E3	Spezifiziert den dritten Analogeingang, welcher von der Summe von E1 und E2 subtrahiert wird. Die Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
PAR1	Vorgabewert der Einschaltschwelle. Ist der Analogwert größer als Einschaltschwelle, so wird der Ausgang des Funktionsblocks aktiv.
PAR2	Vorgabewert der Ausschaltschwelle. Ist der Analogwert kleiner oder gleich der Ausschaltschwelle, so wird der Ausgang des Funktionsblocks deaktiviert.

(ABLNK) Asymmetrischer Taktgeber

Speicherbedarf:	5 Worte	Ab Version:	1.0
Eingänge:	2	Parameter:	2

Funktionsweise:
Der asymmetrische Taktgeber erzeugt ein im 100ms-Bereich parametrierbares, asymmetrisches Taktsignal. Hierbei sind die Zeit, in der der Ausgang des Funktionsblocks "1" ist, so wie die Zeit, in der der Ausgang "0" ist, jeweils unabhängig voneinander parametrierbar. Zusätzlich kann über einen zweiten Eingang (E2) der Ausgang des Funktionsblocks invertiert werden.

Der Taktgeber läuft, so lange der Eingang E1 logisch "1" ist. Andernfalls (E1="0") bleibt der Ausgang des Funktionsblocks "0". Ist der Eingang E2 logisch "1", so wird der Ausgang des Funktionsblocks invertiert (das äußert sich im praktischen Betrieb in einer Vertauschung der Ein/Aus-Zeiten). Die Invertierung wird nur ausgeführt, wenn E1 = "1" ist.

Über den Parameter PAR1 wird die Zeit in 100ms-Schritten festgelegt, in der der Ausgang jeweils "1" ist. Der Parameter PAR2 legt die Zeit in 100ms-Schritten fest, in der der Ausgang logisch "0" ist. PAR1 und PAR2 dürfen Werte im Bereich von 1-65535 (0,1-6553,5 Sekunden) annehmen.

Syntax:
```
Label (ABLNK)
{
E1=Pumpe1()
E2=Inverter()
PAR1=50
PAR2=30
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(ECNT)	Befehlscode für den Block
E1	So lange der Eingang E1 "1" ist, erzeugt der Funktionsblock Takte mit den definierten Impulsbreiten. Ist der Eingang "0", so werden keine Taktimpulse erzeugt. Der Ausgang des Funktionsblocks ist dann immer "0".
E2	Wenn E2 logisch "1" ist, so wird der Ausgang des Funktionsblocks invertiert.
PAR1	Impulsbreite in 100ms für den Zustand "1" des Ausgangs. Der Wert 50 erzeugt z.B. Impulse, bei denen der Ausgang des Funktionsblocks 5 Sekunden lang "1" ist.
PAR2	Impulsbreite in 100ms für den Zustand "0" des Ausgangs. Der Wert 30 erzeugt z.B. Impulse, bei denen der Ausgang des Funktionsblocks 3 Sekunden lang "0" ist.

(ACMP) Analogkomparator

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	2	Parameter:	---

Funktionsweise:
Der Schwellwertschalter vergleicht den Wert eines Analogeingangs (E1) mit dem eines zweiten Analogeingangs (E2). Überschreitet die Differenz der beiden Analogwerte zueinander den Wert der Schaltschwelle (PAR1), so ist der Ausgang des Funktionsblocks aktiv..

Der Analogparameter PAR2 kann skaliert angegeben werden – hierbei ist eine Umrechnung des echten Messwertes in die 12-Bit-Auflösung des Analogeingangs nicht nötig. Hinter dem Parameter können – durch Kommata getrennt – der Minimalwert (welcher 0mA entspricht) und der Maximalwert (welcher 20mA entspricht) angegeben werden. Optional ist auch noch die Angabe der Maßeinheit möglich. Das Format ist wie folgt:

```
PARx=Schwellwert,Minwert,Maxwert,Einheit
```

Minwert und Maxwert können jeweils im Bereich von -32767 bis +32768 liegen. Es werden 4 Nachkommastellen berücksichtigt. Der Name der Einheit darf maximal 7 Zeichen lang sein. Werden Minwert und Maxwert nicht angegeben, so muss der Schwellwert zwischen 0 und 4095 liegen.

Syntax:
```
Label (ACMP)
{
E1=ANAIN,1
E2=ANAIN,3
PAR=0.55,0,11.5,Meter
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(ACMP)	Befehlscode für den Block
E2	Spezifiziert den ersten Analogeingang, der verglichen werden soll. Sie Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
E2	Spezifiziert den zweiten Analogeingang, der verglichen werden soll. Sie Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 3 – so wird der dritte Analogeingang zum Vergleich herangezogen.
PAR1	Legt die Schaltschwelle. Wenn die Differenz zwischen E1 und E2 größer als der hier definierte Wert ist, so ist der Ausgang des Funktionsblocks aktiv.

(AND) Logische UND-Verknüpfung

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	---

Funktionsweise:
Dieser Funktionsblock stellt eine logische UND-Verknüpfung für 3 Eingänge zur Verfügung. Die Bedingung des Blockes ist dann erfüllt (Ausgang logisch "1"), wenn alle drei Eingänge logisch "1" sind.

Syntax:
```
Label (AND)
{
E1=Pumpe1()
E2=#Fuellstand_hoch
E3=1
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(AND)	Befehlscode für den Block
E1 E2 E3	Eingänge, von denen jeder auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist. Unbelegte Eingänge müssen auf "1" gesetzt werden! Im obigen Beispiel wird für E2 die Verwendung eines Platzhalters für die Definition des Eingangs aufgezeigt. Dieser muss im Define-Block des PicoLogo-Programms definiert sein!

(ASS) Schwellwertschalter Analog

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	1

Funktionsweise:
Der Schwellwertschalter vergleicht den Wert eines Analogeingangs mit fest vorgegebenen Werten. Ist der Wert des Analogeingangs größer als die über PAR1 festgelegte Einschaltschwelle, so wird der Ausgang des Funktionsblocks aktiv. Fällt der Wert unter die über PAR2 festgelegte Ausschaltschwelle, so wird der Ausgang wieder deaktiviert. Ein Unterschreiten der Ausschaltschwelle hat Vorrang.

Die Analogparameter können skaliert angegeben werden – hierbei ist eine Umrechnung des echten Messwertes in die 12-Bit-Auflösung des Analogeingangs nicht nötig. Hinter dem Parameter können – durch Kommata getrennt – der Minimalwert (welcher 0mA entspricht) und der Maximalwert (welcher 20mA entspricht) angegeben werden. Optional ist auch noch die Angabe der Maßeinheit möglich. Das Format ist wie folgt:

```
PARx=Schwellwert,Minwert,Maxwert,Einheit
```

Minwert und Maxwert können jeweils im Bereich von -32767 bis +32768 liegen. Es werden 4 Nachkommastellen berücksichtigt. Der Name der Einheit darf maximal 7 Zeichen lang sein. Werden Minwert und Maxwert nicht angegeben, so muss der Schwellwert zwischen 0 und 4095 liegen.

Syntax:
```
Label (ASS)
{
E1=ANAIN,1
PAR1=6.35,0,11.5,Meter
PAR2=3.75,0,11.5,Meter
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(ASS)	Befehlscode für den Block
E1	Spezifiziert der Analogeingang, der verglichen werden soll. Sie Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
PAR1	Vorgabewert der Einschaltschwelle. Ist der Analogwert größer als Einschaltschwelle, so wird der Ausgang des Funktionsblocks aktiv.
PAR2	Vorgabewert der Ausschaltschwelle. Ist der Analogwert kleiner oder gleich der Ausschaltschwelle, so wird der Ausgang des Funktionsblocks deaktiviert.

(DELOFF) Ausschaltverzögerung

Speicherbedarf:	3 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	1

Funktionsweise:
Mit Hilfe dieses Funktionsblocks kann eine Ausschaltverzögerung realisiert werden. Die Zeit der Ausschaltverzögerung wird als Parameter übergeben. Der Ausgang dieses Blocks wird sofort aktiviert (logisch "1"), sobald der Eingang logisch "1" ist. Der Ausgang wird deaktiviert, wenn für wenigstens x * 100ms der Ausgang logisch "0" ist.

Der Zeitparameter wird in 100ms-Einheiten angegeben. Ein Zeitparameter von 450 realisiert so z.B. eine Einschaltverzögerung von 45 Sekunden. Maximal ist ein Zeitparameter von 65535 (entspricht ca. 1,8 Stunden) möglich.

Syntax:
```
Label (DELOFF)
{
E1=Pumpe1()
PAR1=450
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(DELOFF)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter, gibt die Zeit in 100ms-Schritten an. Eine Zeit von 450 entspricht so z.B. 45 Sekunden.

(DELON) Einschaltverzögerung

Speicherbedarf:	3 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	1

Funktionsweise:
Mit Hilfe dieses Funktionsblocks kann eine Einschaltverzögerung realisiert werden. Die Zeit der Einschaltverzögerung wird als Parameter übergeben. Der Ausgang dieses Blocks wird aktiviert (logisch "1"), wenn für wenigstens x * 100ms der Eingang konstant logisch "1" ist. Der Ausgang wird sofort deaktiviert, wenn der Eingang logisch "0" ist.

Der Zeitparameter wird in 100ms-Einheiten angegeben. Ein Zeitparameter von 450 realisiert so z.B. eine Einschaltverzögerung von 45 Sekunden. Maximal ist ein Zeitparameter von 65535 (entspricht ca. 1,8 Stunden) möglich.

Syntax:
```
Label (DELON)
{
E1=Pumpe1()
PAR1=450
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(DELON)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter, gibt die Zeit in 100ms-Schritten an. Eine Zeit von 450 entspricht so z.B. 45 Sekunden.

(DELONOFF) Ein- und Ausschaltverzögerung

Speicherbedarf:	4 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	2

Funktionsweise:
Mit Hilfe dieses Funktionsblocks kann eine Ein-/Ausschaltverzögerung realisiert werden. Die Zeit der Einschaltverzögerung und Ausschaltverzögerung werden jeweils als Parameter übergeben. Der Ausgang dieses Blocks wird aktiviert (logisch "1"), wenn für wenigstens x * 100ms der Eingang konstant logisch "1" ist. Der Ausgang wird deaktiviert, wenn für wenigstens y * 100ms der Ausgang logisch "0" ist.

Der Zeitparameter wird in 100ms-Einheiten angegeben. Ein Zeitparameter von 450 realisiert so z.B. eine Einschaltverzögerung von 45 Sekunden. Maximal ist ein Zeitparameter von 65535 (entspricht ca. 1,8 Stunden) möglich.

Syntax:
```
Label (DELONOFF)
{
E1=Pumpe1()
PAR1=450
PAR2=200
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(DELON)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter der **Einschaltverzögerung**, gibt die Zeit in 100ms-Schritten an. Eine Zeit von 450 entspricht so z.B. 45 Sekunden.
PAR2	Zeitparameter der **Ausschaltverzögerung**, gibt die Zeit in 100ms-Schritten an. Eine Zeit von 200 entspricht so z.B. 20 Sekunden.

(DELSTO) Speichernde Einschaltverzögerung

Speicherbedarf:	4 Worte	Ab Version:	1.0
Eingänge:	2	Parameter:	1

Funktionsweise:
Bei der speichernden Einschaltverzögerung wird der Ausgang aktiviert, wenn der Eingang E1 für eine parametrierbare Zeit (PAR1) ununterbrochen aktiv ist. Ist der Ausgang aktiviert, so verbleibt er in diesem Zustand, auch wenn der Eingang E1 danach auf "0" zurückfällt.

Der Ausgang kann erst wieder deaktiviert werden, wenn am Rücksetzeingang E2 logisch "1" anliegt. Nach dem Rücksetzen kann die Einschaltverzögerung erneut gestartet werden, wenn der Rücksetzeingang E2 wieder logisch "0" ist.

Der Zeitparameter wird in 100ms-Einheiten angegeben. Ein Zeitparameter von 450 realisiert so z.B. eine Einschaltverzögerung von 45 Sekunden. Maximal ist ein Zeitparameter von 65535 (entspricht ca. 1,8 Stunden) möglich.

Syntax:
```
Label (DELSTO)
{
E1=Pumpe1()
E2=Ruecksetz1()
PAR1=450
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(DELSTO)	Befehlscode für den Block
E1	Eingang zur Aktivierung der Einschaltverzögerung. Der Eingang muss auf einen anderen Block verweisen. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
E2	Eingang zum Rücksetzen des Ausgangs. Der Eingang muss auf einen anderen Block verweisen. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter, gibt die Zeit in 100ms-Schritten an. Eine Zeit von 450 entspricht so z.B. 45 Sekunden.

(ECNT) Vor-Rückwärtszähler

Speicherbedarf:	4 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	1

Funktionsweise:
Der Vor- / Rückwärtszähler zählt an seinem Eingang E1 auftretende Ereignisse (Ein Ereignis ist der Wechsel von logisch "0" nach logisch "1"). Die Zählrichtung kann über den Eingang E2 festgelegt werden ("0"=Aufwärtszählen, "1"=Abwärtszählen). Der Zähler kann über den Eingang E3 zurückgesetzt werden (Zähler wird auf 0 gesetzt, solange E3="1" ist). So lange der Rücksetzeingang aktiv ist, bleibt der Ausgang auf "0" und eingehende Impulse werden nicht gezählt.

Über einen Parameter kann ein Zählwert im Bereich von 0-65535 definiert werden. Wenn der Zählerwert diesen Parameterwert erreicht oder überschreitet, wird der Ausgang des Funktionsblocks gesetzt.

Findet ein Überlauf (Aufwärtszählimpuls bei Zählerstand 65535) oder Unterlauf (Abwärtszählimpuls bei Zählerstand 0) statt, so bleibt der Zähler unverändert beim jeweiligen Zählerstand stehen.

Dieser Funktionsblock kann mit dem Befehl "REMON" remanent konfiguriert werden.

Syntax:
```
Label (ECNT)
{
E1=Pumpe1()
E2=Zaehlrichtung()
E3=Ruecksetzblock()
PAR1=2430
REMON
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(ECNT)	Befehlscode für den Block
E1	Zähleingang. Jeder Wechsel von "0" nach "1" erhöht oder erniedrigt (je nach Richtung) den Zähler um den Wert 1
E2	Zählrichtung. Wenn der Eingang "0" ist, zählt der Zähler aufwärts. Ist der Eingang "1", zählt der Zähler abwärts.
E3	Rücksetzeingang: Wenn E3 = "1" ist, so wird der Zähler auf "0" gesetzt und so lange auf 0 gehalten, bis E3 wieder "0" ist.
PAR1	Zählwert, bei dessen Erreichen der Ausgang des Funktionsblocks aktiviert wird. Der Ausgang bleibt so lange aktiv, bis der Zähler zurückgesetzt wird.

(HBCON1) Hochbehältersteuerung

Speicherbedarf:	4 Worte	Ab Version:	1.0
Eingänge:	2	Parameter:	6

Funktionsweise:
Dieser Funktionsblock stellt eine Hochbehältersteuerung getrennt nach Haupttarifzeit und Nebentarifzeit zur Verfügung.

Wenn der Eingang E1 des Funktionsblocks aktiviert ist (logisch "1"), so wird der Analogeingang E2 überwacht. Fällt der Wert des Analogeingangs unter eine parametrierbare Min-Schwelle, so wird der Ausgang des Funktionsblocks aktiv. Übersteigt der Wert des Analogeingangs eine parametrierbare Max-Schwelle, so wird der Ausgangs des Funktionsblocks inaktiv.

Es existieren je eine eigene Min- und Max-Schwelle für die Haupttarifzeit und die Nebentarifzeit. Die Zeitspanne der Haupttarifzeit kann frei definiert werden.

Der Parameter PAR1 gibt den Zeitpunkt des Beginns der Haupttarifzeit an. Die Zeit ist immer fünfstellig im Format hh:mm einzugeben. Für 7 Uhr 30 ist z.B. der Wert "07:30" zu setzen. Im gleichen Format gibt der Parameter PAR2 den Zeitpunkt der Nebentarifzeit an. Sind die Parameter PAR1 und PAR2 nicht definiert, so werden ständig die Schaltschwellen der Haupttarifzeit zur Steuerung herangezogen.

Der Parameter PAR3 gibt die Min-Schwelle der Haupttarifzeit an, bei deren Unterschreitung der Ausgang des Funktionsblocks aktiv wird. Der Parameter PAR4 gibt die Max-Schwelle der Haupttarifzeit an, bei deren Erreichen der Ausgang des Funktionsblocks wieder deaktiviert wird.

Analog hierzu definieren PAR5 und PAR6 die Min-Schwelle bzw. Max-Schwelle der Nebentarifzeit.

Alle Analogparameter PAR3-PAR6 können skaliert angegeben werden – hierbei ist eine Umrechnung des echten Messwertes in die 12-Bit-Auflösung des Analogeingangs nicht nötig. Hinter dem jeweiligen Parameter können – durch Kommata getrennt – der Minimalwert (welcher 0mA entspricht) und der Maximalwert (welcher 20mA entspricht) angegeben werden. Optional ist auch noch die Angabe der Maßeinheit möglich. Das Format ist wie folgt:

```
PARx=Schwellwert,Minwert,Maxwert,Einheit
```

Minwert und Maxwert können jeweils im Bereich von -32767 bis +32768 liegen. Es werden 4 Nachkommastellen berücksichtigt. Der Name der Einheit darf maximal 7 Zeichen lang sein. Werden Minwert und Maxwert nicht angegeben, so muss der Schwellwert zwischen 0 und 4095 liegen.

Unabhängig von Zeiten und Schwellen ist der Ausgang des Funktionsblocks immer deaktiviert, wenn der Eingang E1 logisch "0" ist.

Syntax:

```
Label (HBCON1)
{
E1=Pumpe1()
E2=ANAIN,1
PAR1=07:30
PAR2=19:00
PAR3=1.5,0,12,Meter
PAR4=2.25,0,12,Meter
PAR5=4,0,12,Meter
PAR6=7,0,12,Meter
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(HBCON1)	Befehlscode für den Block
E1	Eingang zur Freigabe des Ausgangs. Der Eingang muss auf einen anderen Block verweisen. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
E2	Spezifiziert der Analogeingang, der z.B. den Füllstand liefert. Die Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
PAR1	Definiert den Beginn der Haupttarifzeit. Die Zeit ist fünfstellig einzugeben. "07:30" bedeutet z.B. 07 Uhr 30.
PAR2	Definiert das Ende der Haupttarifzeit. Die Zeit ist fünfstellig einzugeben. "19:00" bedeutet z.B. 19 Uhr.
PAR3	Definiert die Min-Schwelle zur Haupttarifzeit, bei deren Unterschreitung der Ausgang des Funktionsblocks aktiv wird.
PAR4	Definiert die Max-Schwelle zur Haupttarifzeit, bei deren Erreichen der Ausgang des Funktionsblocks deaktiviert wird.
PAR5	Definiert die Min-Schwelle zur Nebentarifzeit, bei deren Unterschreitung der Ausgang des Funktionsblocks aktiv wird.
PAR6	Definiert die Max-Schwelle zur Nebentarifzeit, bei deren Erreichen der Ausgang des Funktionsblocks deaktiviert wird.

(HBCON2) Hochbehältersteuerung 2 (Zweikammersysteme)

Speicherbedarf:	4 Worte	Ab Version:	1.10
Eingänge:	2	Parameter:	6

Funktionsweise:
Dieser Funktionsblock stellt eine Hochbehältersteuerung getrennt nach Haupttarifzeit und Nebentarifzeit zur Verfügung.

Wenn der Eingang E1 des Funktionsblocks aktiviert ist (logisch "1"), so wird die Summe der Analogeingänge E2 und E3 überwacht. Fällt die Summe der Analogeingänge unter eine parametrierbare Min-Schwelle, so wird der Ausgang des Funktionsblocks aktiv. Übersteigt die Summe der Analogeingänge eine parametrierbare Max-Schwelle, so wird der Ausgangs des Funktionsblocks inaktiv.

Es existieren je eine eigene Min- und Max-Schwelle für die Haupttarifzeit und die Nebentarifzeit. Die Zeitspanne der Haupttarifzeit kann frei definiert werden.

Der Parameter PAR1 gibt den Zeitpunkt des Beginns der Haupttarifzeit an. Die Zeit ist immer fünfstellig im Format hh:mm einzugeben. Für 7 Uhr 30 ist z.B. der Wert "07:30" zu setzen. Im gleichen Format gibt der Parameter PAR2 den Zeitpunkt der Nebentarifzeit an. Sind die Parameter PAR1 und PAR2 nicht definiert, so werden ständig die Schaltschwellen der Haupttarifzeit zur Steuerung herangezogen.

Der Parameter PAR3 gibt die Min-Schwelle der Haupttarifzeit an, bei deren Unterschreitung der Ausgang des Funktionsblocks aktiv wird. Der Parameter PAR4 gibt die Max-Schwelle der Haupttarifzeit an, bei deren Erreichen der Ausgang des Funktionsblocks wieder deaktiviert wird.

Analog hierzu definieren PAR5 und PAR6 die Min-Schwelle bzw. Max-Schwelle der Nebentarifzeit.

Alle Analogparameter PAR3-PAR6 können skaliert angegeben werden – hierbei ist eine Umrechnung des echten Messwertes in die 12-Bit-Auflösung des Analogeingangs nicht nötig. Hinter dem jeweiligen Parameter können – durch Kommata getrennt – der Minimalwert (welcher 0mA entspricht) und der Maximalwert (welcher 20mA entspricht) angegeben werden. Optional ist auch noch die Angabe der Maßeinheit möglich. Das Format ist wie folgt:

```
PARx=Schwellwert,Minwert,Maxwert,Einheit
```

Es ist zu beachten, dass die Skalierung für beide Analogeingänge identisch sein muss.

Minwert und Maxwert können jeweils im Bereich von -32767 bis +32768 liegen. Es werden 4 Nachkommastellen berücksichtigt. Der Name der Einheit darf maximal 7 Zeichen lang sein. Werden Minwert und Maxwert nicht angegeben, so muss der Schwellwert zwischen 0 und 4095 liegen.

Unabhängig von Zeiten und Schwellen ist der Ausgang des Funktionsblocks immer deaktiviert, wenn der Eingang E1 logisch "0" ist.

Syntax:

```
Label (HBCON1)
{
E1=Pumpe1()
E2=ANAIN,1
E3=ANAIN,2
PAR1=07:30
PAR2=19:00
PAR3=1.5,0,12,Meter
PAR4=2.25,0,12,Meter
PAR5=4,0,12,Meter
PAR6=7,0,12,Meter
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(HBCON1)	Befehlscode für den Block
E1	Eingang zur Freigabe des Ausgangs. Der Eingang muss auf einen anderen Block verweisen. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
E2	Spezifiziert den ersten Analogeingang, welcher zu E3 hinzuaddiert wird. Die Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
E3	Spezifiziert den zweiten Analogeingang, welcher zu E2 hinzuaddiert wird. Die Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
PAR1	Definiert den Beginn der Haupttarifzeit. Die Zeit ist fünfstellig einzugeben. "07:30" bedeutet z.B. 07 Uhr 30.
PAR2	Definiert das Ende der Haupttarifzeit. Die Zeit ist fünfstellig einzugeben. "19:00" bedeutet z.B. 19 Uhr.
PAR3	Definiert die Min-Schwelle zur Haupttarifzeit, bei deren Unterschreitung der Ausgang des Funktionsblocks aktiv wird.
PAR4	Definiert die Max-Schwelle zur Haupttarifzeit, bei deren Erreichen der Ausgang des Funktionsblocks deaktiviert wird.
PAR5	Definiert die Min-Schwelle zur Nebentarifzeit, bei deren Unterschreitung der Ausgang des Funktionsblocks aktiv wird.
PAR6	Definiert die Max-Schwelle zur Nebentarifzeit, bei deren Erreichen der Ausgang des Funktionsblocks deaktiviert wird.

(INPUT) Binäreingang einlesen

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	---

Funktionsweise:
Dieser Funktionsblock liest einen binären Eingang des Gerätes bzw. dessen angeschlossenen Erweiterungsmodulen aus. Wenn der zugewiesene Eingang aktiv ist, so ist die Blockbedingung erfüllt (logisch "1").

Syntax:
```
Label (INPUT)
{
E1=Binin,12
}
```

Alternativ kann für die internen Eingänge des Grundgerätes auch direkt der Name des Eingangs in eckigen Klammern gesetzt werden. Der oben aufgeführte Eingang 12 (Port B3) kann auch wie folgt angegeben werden:

```
E1=Binin[B3]
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(INPUT)	Befehlscode für den Block
E1	Definiert, welcher Binäreingang eingelesen wird. Die Binäreingänge sind von Port A0 des Grundgerätes an laufend durchnumeriert. Im o.a. Beispiel wird der 12. Binäreingang (Eingang B3 des Grundgerätes) eingelesen. Ist die Nummer des Binäreinganges höher als die im Gerät verfügbaren (und im Kopf des PicoLogo-Programms definierten) Eingänge, so wird eine Fehlermeldung ausgegeben.

(INV) Logische INVERTER-Verknüpfung

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	---

Funktionsweise:
Dieser Funktionsblock stellt an seinem Ausgang das invertierte Signal des Eingangs zur Verfügung. Ist der Eingang "0", so ist der Ausgang des Blocks "1" und umgekehrt.

Syntax:
```
Label (INV)
{
E1=Pumpe1()
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(INV)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.

(MELDE) Alarmierung über Meldeblock (nur GSM-4110)

Speicherbedarf:	3 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	1

Funktionsweise:
Dieser Funktionsblock ist nur bei dem Gerät GSM-4110 verfügbar. Durch den Block ist es möglich, einen der 16 Binär-Meldeblöcke auszulösen.

Wenn der Eingang des Funktionsblockes "1" ist, so wird der über PAR1 parametrierte Meldeblock (1...16) ausgelöst und das Gerät nimmt die entsprechende Alarmierung vor.

Syntax:
```
Label (MELDE)
{
E1=Pumpe1()
PAR1=1
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(MELDE)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Gibt den auszulösenden Meldeblock im Bereich von 1-16 an.

(NAND) Logische NAND-Verknüpfung

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	---

Funktionsweise:
Dieser Funktionsblock stellt eine logische NAND-Verknüpfung für 3 Eingänge zur Verfügung. Die Bedingung des Blockes ist dann erfüllt (Ausgang logisch "1"), wenn nicht alle drei Eingänge logisch "1" sind.

Syntax:
```
Label (NAND)
{
E1=Pumpe1()
E2=#Fuellstand_hoch
E3=1
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(NAND)	Befehlscode für den Block
E1 E2 E3	Eingänge, von denen jeder auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist. Unbelegte Eingänge müssen auf "1" gesetzt werden! Im obigen Beispiel wird für E2 die Verwendung eines Platzhalters für die Definition des Eingangs aufgezeigt. Dieser muss im Define-Block des PicoLogo-Programms definiert sein!

(NOR) Logische NOR-Verknüpfung

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	---

Funktionsweise:
Dieser Funktionsblock stellt eine logische NOR-Verknüpfung für 3 Eingänge zur Verfügung. Die Bedingung des Blockes ist dann erfüllt (Ausgang logisch "1"), wenn alle drei Eingänge logisch "0" ist.

Syntax:
```
Label (NOR)
{
E1=Pumpe1()
E2=#Fuellstand_hoch
E3=0
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(NOR)	Befehlscode für den Block
E1 E2 E3	Eingänge, von denen jeder auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist. Unbelegte Eingänge müssen auf "0" gesetzt werden! Im obigen Beispiel wird für E2 die Verwendung eines Platzhalters für die Definition des Eingangs aufgezeigt. Dieser muss im Define-Block des PicoLogo-Programms definiert sein!

(OR) Logische ODER-Verknüpfung

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	---

Funktionsweise:
Dieser Funktionsblock stellt eine logische UND-Verknüpfung für 3 Eingänge zur Verfügung. Die Bedingung des Blockes ist dann erfüllt (Ausgang logisch "1"), wenn wenigstens einer der drei Eingänge logisch "1" ist.

Syntax:
```
Label (OR)
{
E1=Pumpe1()
E2=#Fuellstand_hoch
E3=0
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(OR)	Befehlscode für den Block
E1 E2 E3	Eingänge, von denen jeder auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist. Unbelegte Eingänge müssen auf "0" gesetzt werden! Im obigen Beispiel wird für E2 die Verwendung eines Platzhalters für die Definition des Eingangs aufgezeigt. Dieser muss im Define-Block des PicoLogo-Programms definiert sein!

(OUTPUT) Binärwert auf Ausgang ausgeben

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	---

Funktionsweise:
Dieser Funktionsblock gibt den Wert, der an seinem Eingang anliegt, auf einem anzugebenden Binärausgang des Gerätes aus.

Syntax:
```
Label (OUTPUT)
{
E1=Pumpe1()
Q=BINOUT,3
}
```

Alternativ kann für die internen Ausgänge des Grundgerätes auch direkt der Name des Ausgangs in eckigen Klammern gesetzt werden. Der oben aufgeführte Ausgang 3 (Port C2) kann auch wie folgt angegeben werden:

```
Q=BINOUT[C2]
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(OUTPUT)	Befehlscode für den Block
E1	Eingänge, von denen jeder auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
Q	Definiert, auf welchem Binärausgang der am Eingang des Blocks liegende Wert ausgegeben wird. Die Binärausgänge sind von Port C0 des Grundgerätes an laufend durchnumeriert. Im o.a. Beispiel wird der 3. Binärausgang (Ausgang C2 des Grundgerätes) zur Ausgabe verwendet. Ist die Nummer des Binärausganges höher als die im Gerät verfügbaren (und im Kopf des PicoLogo-Programms definierten) Ausgänge, so wird eine Fehlermeldung ausgegeben.

(PUMPCO1) Pumpenüberwachung

Speicherbedarf:	4 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	2

Funktionsweise:
Dieser Funktionsblock überwacht eine Pumpe nach den Kriterien Durchfluss und Motorschutzschalter.

Wenn der Eingang E1 des Funktionsblocks aktiviert wird (Wechsel von "0" nach "1"), wird der Ausgang des Funktionsblocks ebenfalls aktiv. Gleichzeitig wird eine über PAR1 in 100ms-Schritten parametrierbare Zeit gestartet. Nach Ablauf dieser Zeit muss der Analogwert E2 (Durchflussmesser mit analogem Ausgang) mindestens einen über PAR2 vorgegebenen Wert erreicht haben., andernfalls wird der Ausgang nach Ablauf dieser Zeit wieder deaktiviert.

Fällt der Analogeingang zu einem beliebigen Zeitpunkt nach Ablauf der Verzögerungszeit unter den parametrierten Wert, so wird der Ausgang des Funktionsblocks abgeschaltet.

Wird der Eingang E1 wieder "0", so wird der Ausgang des Funktionsblocks wieder zurückgesetzt.

Wenn der Eingang E3 (Motorschutzschalter) "1" ist, wird der Ausgang des Funktionsblocks sofort auf "0" gesetzt.

Ein durch den Motorschutzschalter oder Unterschreiten des Analogwertes abgeschalteter Ausgang wird erst wieder aktiviert, nachdem der Steuereingang E1 wieder auf "0" und dann wieder auf "1" gesetzt wurde.

Der Wertebereich von PAR1 liegt bei 0-65535, der Analogparameter PAR2 kann skaliert angegeben werden – hierbei ist eine Umrechnung des echten Messwertes in die 12-Bit-Auflösung des Analogeingangs nicht nötig. Hinter dem Parameter können – durch Kommata getrennt – der Minimalwert (welcher 0mA entspricht) und der Maximalwert (welcher 20mA entspricht) angegeben werden. Optional ist auch noch die Angabe der Maßeinheit möglich. Das Format ist wie folgt:

```
PARx=Schwellwert,Minwert,Maxwert,Einheit
```

Minwert und Maxwert können jeweils im Bereich von -32767 bis +32768 liegen. Es werden 4 Nachkommastellen berücksichtigt. Der Name der Einheit darf maximal 7 Zeichen lang sein. Werden Minwert und Maxwert nicht angegeben, so muss der Schwellwert zwischen 0 und 4095 liegen.

Syntax:

```
Label (PUMPCO1)
{
E1=Pumpe1()
E2=ANAIN,1
E3=Motorschutz()
PAR1=100
PAR2=22,0,100,cm3/s
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(PUMPCO1)	Befehlscode für den Block
E1	Eingang zum Einschalten der Pumpe, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
E2	Spezifiziert der Analogeingang, an dem der Durchflussmesser angeschlossen ist. Sie Syntax ist ANAIN,x – wobei x den Analogeingang spezifiziert. Ist x, wie im obigen Beispiel, 1 – so wird der erste Analogeingang zum Vergleich herangezogen.
E3	Eingang des Motorschutzschalters, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter, gibt die Zeit in 100ms-Schritten an, nach der am Analogeingang E2 der unter PAR2 definierte Mindestwert anliegen muss. Eine Zeit von 100 entspricht so z.B. 10 Sekunden.
PAR2	Gibt den Wert an, den der Analogeingang nach Ablauf der unter PAR1 definierten Zeit mindestens aufweisen muss.

(RND) Zufallsgenerator

Speicherbedarf:	4 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	2

Funktionsweise:
Wenn der Eingang E1 des Funktionsblocks von 0 nach 1 wechselt, so wird eine zufällige Einschaltverzögerungszeit ermittelt, die zwischen 0 und dem Parameterwert PAR1 liegt. Bleibt der Eingang E1 für die Dauer der Einschaltverzögerung auf "1", so wird nach Ablauf der Zeit der Ausgang des Funktionsblocks aktiviert. Wechselt der Eingang E1 vor Ablauf der Zeit wieder nach 0, so wird die Zeit zurückgesetzt und der Eingang nicht aktiviert.

Wechselt der Eingang E1 von 1 nach 0, so wird eine zufällige Ausschaltverzögerungszeit ermittelt, die zwischen 0 und dem Parameterwert PAR2 liegt. Bleibt der Eingang E1 für die Dauer der Ausschaltverzögerung auf "0", so wird nach Ablauf der Zeit der Ausgang des Funktionsblocks deaktiviert. Wechselt der Eingang E1 vor Ablauf der Zeit wieder nach 0, so wird die Zeit zurückgesetzt und der Eingang nicht aktiviert.

Hinweis:
Ein selbstlaufender Zufallsgenerator kann gebildet werden, in dem der Ausgang des Funktionsblocks über einen Inverter an den eigenen Eingang gelegt wird.

Syntax:
```
Label (RND)
{
E1=Eingang()
PAR1=1000
PAR2=500
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(RND)	Befehlscode für den Block
E1	Aktivierungseingang, der auf einen anderen Block verweisen muss. Ein Wechsel von "0" nach "1" aktiviert eine Einschaltverzögerung mit einer über PAR1 parametrierbaren zufälligen Zeit, ein Wechsel von "1" nach "0" aktiviert eine Ausschaltverzögerung mit einer über PAR2 parametrierbaren, zufälligen Zeit.
PAR1	Maximaler Zufallswert in Sekunden für die Zeit der Einschaltverzögerung
PAR2	Maximaler Zufallswert in Sekunden für die Zeit der Ausschaltverzögerung

(SBLNK) Symmetrischer Taktgeber

Speicherbedarf:	3 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	1

Funktionsweise:
Der symmetrische Taktgeber erzeugt ein im Sekundenbereich parametrierbares, symmetrisches Taktsignal. Hierbei ist die Zeit, in der der Ausgang des Funktionsblocks "1" ist, genauso groß wie die Zeit, in der der Ausgang "0" ist.

Der Taktgeber läuft, so lange der Eingang E1 logisch "1" ist. Ist E1 logisch "0", so wird auch der Ausgang auf "0" gesetzt.

Über den Parameter PAR1 wird die Zeit in 100ms-Schritten festgelegt, in der der Ausgang jeweils "1" oder "0" ist. Eine Zeitvorgabe von 50 bedeutet, dass der Ausgang des Funktionsblocks bei aktivem Eingang E1 jeweils 5 Sekunden "0" ist, dann 5 Sekunden "1" und so weiter. PAR1 darf im Bereich von 1-65535 Sekunden liegen.

Syntax:
```
Label (SBLNK)
{
E1=Pumpe1()
PAR1=50
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(ECNT)	Befehlscode für den Block
E1	So lange der Eingang E1 "1" ist, erzeugt der Funktionsblock Takte mit der definierten Impulsbreite. Ist der Eingang "0", so verweilt der Ausgang des Funktionsblocks in der momentanen Stellung.
PAR1	Impulsbreite in 100ms-Einheiten. Der Wert 50 erzeugt z.B. Impulse, bei denen der Ausgang des Funktionsblocks 5 Sekunden lang "1" und 5 Sekunden lang "0" ist.

(SCOMF) Komfortschalter

Speicherbedarf:	4 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	2

Funktionsweise:
Der Ausgang des Funktionsblocks "Komfortschalter" wird durch ein kurzes "1"-Signal am Eingang aktiviert und bleibt anschließend für die eingestellte Zeitspanne (PAR1) aktiv. Wird während der Ausgang aktiv ist bereits ein neuer "1"-Impuls am Eingang entdeckt, so beginnt die parametrierte Einschaltzeit von vorne.

Zusätzlich kann der Ausgang des Komfortschalters daueraktiviert werden. Hierzu ist der Eingang für mindestens die mit PAR2 parametrierte Zeitspanne zur Aktivierung der Dauerlichtfunktion aktiv zu halten. Ist der Ausgang daueraktiv geschaltet, so wird er bei der nächsten Aktivierung des Eingangs wieder auf "0" zurückgesetzt.

Die Zeit zur Auslösung der Dauerlichtfunktion (PAR2) wird in 100ms-Einheiten angegeben. Ein Wert von 15 bedeutet hier also z.B. 1,5 Sekunden. Die Zeit der Einschaltdauer (PAR1) wird in Sekunden angegeben.

Syntax:
```
Label (SCOMF)
{
E1=Pumpe1()
PAR1=500
PAR2=25
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(SCOMF)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter, gibt die Zeit der Einschaltdauer des Ausgangs in Sekunden an. Eine Zeit von 50 entspricht so z.B. 50 Sekunden.
PAR2	Zeitparameter, gibt die Zeit in 100ms-Schritten zur Aktivierung der Dauerlichtfunktion an. Eine Zeit von 25 entspricht so z.B. 2,5 Sekunden.

(SHREL) Selbsthalterelais

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	2	Parameter:	---

Funktionsweise:
Ein Selbsthalterelais verfügt über 2 Eingänge: Einen Setz-Eingang (E1) und einen Rücksetz-Eingang (E2). Ist der Setz-Eingang logisch "1", so wird der Ausgang des Funktionsblocks ebenfalls logisch "1". Der Ausgang wird gehalten, wenn der Eingang wieder auf "0" zurückfällt. Ist der Rücksetzeingang des Funktionsblocks "1", so wird der Ausgang auf "0" zurückgesetzt. Er verweilt auf "0", auch wenn der Rücksetzausgang logisch "0" wird.

Der Rücksetzeingang hat Vorrang vor dem Setzeingang – d.h. wenn sowohl der Setzeingang und der Rücksetzeingang logisch "1" sind, ist der Ausgang des Funktionsblocks logisch "0".

Dieser Funktionsblock kann mit dem Befehl "REMON" remanent konfiguriert werden.

Syntax:
```
Label (SHREL)
{
E1=Pumpe1()
E2=#Fuellstand_hoch
REMON
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(SHREL)	Befehlscode für den Block
E1	Setz-Eingang des Selbsthalterelais, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
E2	Rücksetz-Eingang des Selbsthalterelais, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist. Im obigen Beispiel wird die Verwendung eines Platzhalters für die Definition des Eingangs aufgezeigt. Dieser muss im Define-Block des PicoLogo-Programms definiert sein!

(SSREL) Stromstoßrelais

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	2	Parameter:	---

Funktionsweise:
Ein Stromstoßrelais verfügt über 2 Eingänge: Einen Setz-/Rücksetz-Eingang (E1) und einen reinen Rücksetz-Eingang. Wenn der Setz/Rücksetzeingang seinen Zustand von "0" nach "1" ändert, wird der Wert des Ausgangs des Funktionsblocks invertiert. D.h: Ist der Ausgang logisch "0", und der Setz-/Rücksetzeingang ändert sich von "0" nach "1", so wechselt der Ausgang auf "1". Ist der Ausgang "1", und der Setz-/Rücksetzeingang ändert sich von "0" nach "1", so wechselt der Ausgang auf "0".

Der Rücksetzeingang setzt den Ausgang der Funktionsblocks immer auf "0" zurück, solange dieser Eingang "1" ist. Wird der Rücksetzeingang nicht verwendet, so ist er daher (wie im Beispiel) auf "0" zu setzen.

Dieser Funktionsblock kann mit dem Befehl "REMON" remanent konfiguriert werden.

Syntax:
```
Label (SSREL)
{
E1=Pumpe1()
E2=0
REMON
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(SSREL)	Befehlscode für den Block
E1	Setz-/Rücksetz-Eingang des Stromstoßrelais, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
E2	Rücksetz-Eingang des Stromstoßrelais, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.

(STAIR) Treppenlichtschalter

Speicherbedarf:	4 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	2

Funktionsweise:
Der Ausgang des Funktionsblocks "Treppenlichtschalter" wird durch ein kurzes "1"-Signal am Eingang aktiviert und bleibt anschließend für die eingestellte Zeitspanne (PAR1) aktiv. Wird während der Ausgang aktiv ist bereits ein neuer "1"-Impuls am Eingang entdeckt, so beginnt die parametrierte Einschaltzeit von vorne.

Über den Parameter PAR2 kann eine Ausschaltwarnung aktiviert werden. Bei Erreichen des Sekundenwertes von PAR2 wird der Ausgang für 1 Sekunde kurz aus- und danach wieder eingeschaltet. Wenn PAR2 z.B. mit "15" angegeben wurde, so wird der Ausgang 15 Sekunden vor Ablauf der Zeit für die Dauer von 1 Sekunde deaktiviert. Ist PAR2 = 0, so erfolgt keine Ausschaltwarnung.

Die Zeitparameter wird in Sekunden angegeben. Ein Zeitparameter PAR1 von 50 realisiert so z.B. eine Aktivierungszeit von 50 Sekunden. Maximal ist ein Zeitparameter von 65535 (entspricht ca. 18 Stunden) möglich.

Syntax:
```
Label (SSTAIR)
{
E1=Pumpe1()
PAR1=50
PAR2=15
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(SSTAIR)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter, gibt die Einschaltzeit in Sekunden an. Eine Zeit von 50 entspricht so z.B. 50 Sekunden.
PAR2	Gibt den Zeitpunkt in Sekunden vor Ablauf der Einschaltzeit an, zu der eine Ausschalt-Vorwarnung erfolgt. Ist PAR2=0, so erfolgt keine Ausschaltvorwarnung.

(TCNT) Betriebsstundenzähler

Speicherbedarf:	6 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	1

Funktionsweise:
Der Betriebsstundenzähler zählt im Stundentakt aufwärts, so lange der Eingang "E1" logisch "1" ist. Ist dieser Eingang "0", so stoppt der Zähler an seinem momentanen Zählwert. Die interne Auflösung des Zählers ist 1 Sekunde. Der Zähler kann auf 0 zurückgesetzt werden, wenn der Reset-All-Eingang E3 logisch "1" ist.

Der Betriebsstundenzähler verfügt über einen Komparator, welcher nach Verstreichen einer parametrierbaren Betriebszeit den Ausgang des Funktionsblocks setzt. Dieser Wert wird ebenfalls in Stunden angegeben. Der Ausgang wird zurückgesetzt, wenn der Rücksetzeingang E2 logisch "1" ist. In diesem Fall wird die Zeit bis zum Setzen des Ausgangs wieder neu gestartet, der Stundenzähler behält jedoch seinen Wert.

Im Unterschied hierzu setzt der Reset-All-Eingang (E3), wenn er logisch "1" ist, ebenfalls den Ausgang zurück und startet die Zeit bis zum Setzen des Ausgangs neu. Hierbei wird außerdem jedoch der Betriebsstundenzähler auf 0 zurückgesetzt.

Maximalwerte für den Betriebsstundenzähler und für den Parameter PAR1 sind 65535 Stunden (ca. 7,5 Jahre). Erreicht der Betriebsstundenzähler diesen Wert, so wird nicht mehr weitergezählt.

Dieser Funktionsblock kann mit dem Befehl "REMON" remanent konfiguriert werden.

Syntax:

```
Label (TCNT)
{
E1=Pumpe1()
E2=Ruecksetzblock()
E3=Resetall()
PAR1=270
REMON
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(ECNT)	Befehlscode für den Block
E1	Zähleingang. Der Betriebsstundenzähler zählt in Stundenschritten aufwärts, so lange dieser Eingang "1" ist. Die Auflösung der Erfassung ist 1 Sekunde.
E2	Rücksetzeingang, setzt den Ausgang zurück und startet die Zeit bis zum Setzen des Ausgangs neu. Lässt den eigentlichen Betriebsstundenzähler jedoch unverändert
E3	Reset-All-Eingang, setzt den Ausgang zurück und startet die Zeit bis zum Setzen des Ausgangs neu. Setzt auch den Betriebsstundenzähler auf 0 zurück
PAR1	Zählwert, bei dessen Erreichen der Ausgang des Funktionsblocks aktiviert wird. Der Ausgang bleibt so lange aktiv, bis der die Zeit zum Aktivieren des Ausgangs über E2 oder E3 zurückgesetzt wird. In diesem Beispiel ist die Zeit auf 270 Stunden gesetzt.

(WCLK) Wochenschaltuhr

Speicherbedarf:	9 Worte	Ab Version:	1.0
Eingänge:	---	Parameter:	9

Funktionsweise:
Eine Wochenschaltuhr verfügt über 3 getrennte "Nocken", d.h. für den Ausgang des Funktionsblocks können 3 verschiedene Ein- und Ausschaltzeiten angegeben werden. Die Parameter 1-3 stehen für die erste Nocke, die Parameter 4-6 für die zweite Nocke, und die Parameter 7-9 für die dritte Nocke. Wird eine Nocke nicht verwendet, so ist die Wochentagsangabe (Parameter 1,4,7) der jeweiligen Nocke nicht zu definieren.

Der erste Parameter (PAR1, PAR4, PAR7) jeder Nocke definiert die Wochentage, an denen der Schaltvorgang stattfindet. Der Parameter startet mit einem "W", gefolgt von den Ziffern der Wochentage (Montag = 1 bis Sonntag = 7), an denen der Schaltvorgang stattfinden soll. "W123" bedeutet also, dass diese Nocke an den Tagen Montag, Dienstag und Mittwoch schalten soll.

Der zweite Parameter (PAR2, PAR5, PAR8) jeder Nocke definiert einen Einschaltvorgang. Die Zeit ist immer fünfstellig im Format hh:mm einzugeben. Für 7 Uhr 30 ist z.B. der Wert "07:30" zu setzen. Ist für die entsprechende Nocke kein Einschaltvorgang gewünscht, so ist der entsprechende Parameter nicht zu definieren.

Analog hierzu definiert der dritte Parameter (PAR3, PAR6, PAR9) jeder Nocke die entsprechende Ausschaltzeit.

Dieser Funktionsblock kann mit dem Befehl "REMON" remanent konfiguriert werden.

Syntax:

```
Label (WCLK)
{
PAR1=W123
PAR2=07:30
PAR3=13:45

PAR4=W1234567
PAR5=15:00

PAR7=W167
PAR8=20:00
PAR9=23:00

REMON
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(WCLK)	Befehlscode für den Block
PAR1 PAR4 PAR7	Definiert die Wochentage, an denen der Schaltvorgang stattfinden soll. Der Parameter muss mit einem "W" starten, gefolgt von den Ziffern der Wochentage, an denen geschaltet werden soll (1=Montag bis 7=Sonntag). Für die oben aufgeführten Beispiele gilt: W123 = Montag, Dienstag, Mittwoch W1234567 = täglich W167 = Montag, Samstag, Sonntag
PAR2 PAR5 PAR8	Definiert die Zeit, an der die Nocke **eingschaltet** werden soll. Die Zeit ist fünfstellig einzugeben. "17:30" bedeutet z.B. 17 Uhr 30.
PAR3 PAR6 PAR9	Definiert die Zeit, an der die Nocke **ausgschaltet** werden soll. Die Zeit ist fünfstellig einzugeben. "17:30" bedeutet z.B. 17 Uhr 30.

(WREL) Wischrelais (Impulsausgabe)

Speicherbedarf:	3 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	1

Funktionsweise:
Ein Wischrelais erzeugt an dessen Ausgang einen "1"-Impuls mit einer parametrierbaren Länge, wenn sich sein Eingang von "0" nach "1" ändert und auf "1" verbleibt. Wenn z.B. eine Wischzeit von 3 Sekunden eingestellt ist, so würde nach einer Aktivierung des Eingangs (logisch "1") der Ausgang des Funktionsblocks für 3 Sekunden ebenfalls logisch "1" sein und dann auf logisch "0" zurückfallen .

Wird der Eingang während der parametrierten Wischzeit auf 0 zurückgesetzt, so wird der Ausgang ebenfalls sofort auf 0 zurückgesetzt.

Der Zeitparameter wird in 100ms-Einheiten angegeben. Ein Zeitparameter von 500 realisiert so z.B. einer Impulsdauer von 50 Sekunden. Maximal ist ein Zeitparameter von 65535 (entspricht ca. 1,8 Stunden) möglich.

Syntax:
```
Label (WREL)
{
E1=Pumpe1()
PAR1=500
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(WREL)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter, gibt die Zeit in 100ms-Schritten an. Eine Zeit von 500 entspricht so z.B. 50 Sekunden.

(WRELF) Flankengetriggertes Wischrelais (Impulsausgabe)

Speicherbedarf:	3 Worte	Ab Version:	1.0
Eingänge:	1	Parameter:	1

Funktionsweise:
Ein flankengetriggertes Wischrelais erzeugt an dessen Ausgang einen "1"-Impuls mit einer parametrierbaren Länge, wenn sich sein Eingang von "0" nach "1" ändert. Wenn z.B. eine Wischzeit von 3 Sekunden eingestellt ist, so würde nach einer Aktivierung des Eingangs (logisch "1") der Ausgang des Funktionsblocks für 3 Sekunden ebenfalls logisch "1" sein und dann auf logisch "0" zurückfallen – auch wenn der Eingang des Funktionsblocks weiterhin logisch "1" ist.

Beim flankengetriggerten Wischrelais ist es unerheblich, ob der Eingang während eines aktivierten Ausgangs wieder auf "0" zurückfällt, es wird in jedem Fall der Impuls mit mindestens der definierten Länge erzeugt, d.h.: Ist der Eingang nur für die Dauer von z.B. 1 Sekunde "1" und fällt dann wieder auf "0" zurück, so bleibt der Ausgang trotzdem für 3 Sekunden auf "1".

Wird während eines gesetzten Ausgangs eine weitere Flanke von "0" nach "1" am Eingang festgestellt, so bleibt der Ausgang gesetzt, und die Impulszeit beginnt erneut (Retriggerung).

Der Zeitparameter wird in 100ms-Einheiten angegeben. Ein Zeitparameter von 500 realisiert so z.B. einer Impulsdauer von 50 Sekunden. Maximal ist ein Zeitparameter von 65535 (entspricht ca. 1,8 Stunden) möglich.

Syntax:
```
Label (WREL)
{
E1=Pumpe1()
PAR1=500
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(WREL)	Befehlscode für den Block
E1	Eingang, der auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist.
PAR1	Zeitparameter, gibt die Zeit in 100ms-Schritten an. Eine Zeit von 500 entspricht so z.B. 50 Sekunden.

(XOR) Logische EXCLUSIV-ODER-Verknüpfung

Speicherbedarf:	2 Worte	Ab Version:	1.0
Eingänge:	3	Parameter:	---

Funktionsweise:
Dieser Funktionsblock stellt eine logische EXCLUSIV-ODER-Verknüpfung für 3 Eingänge zur Verfügung. Die Bedingung des Blockes ist dann erfüllt (Ausgang logisch "1"), wenn exakt **einer** der 3 Eingänge logisch "1" ist

Syntax:
```
Label (XOR)
{
E1=Pumpe1()
E2=#Fuellstand_hoch
E3=0
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(XOR)	Befehlscode für den Block
E1 E2 E3	Eingänge, von denen jeder auf einen anderen Block verweisen muss. Ein Eingang ist logisch "1", wenn die Bedingung des Blockes, auf den der Eingang verweist, erfüllt ist. Unbelegte Eingänge müssen auf "0" gesetzt werden! Im obigen Beispiel wird für E2 die Verwendung eines Platzhalters für die Definition des Eingangs aufgezeigt. Dieser muss im Define-Block des PicoLogo-Programms definiert sein!

(YCLK) Jahresschaltuhr

Speicherbedarf:	9 Worte	Ab Version:	1.0
Eingänge:	---	Parameter:	3

Funktionsweise:
Die Jahreschaltuhr besitzt nur 1 "Nocke". Der Ausgang der Jahresschaltuhr kann jeweils an einem bestimmten Tag zu einer bestimmten Uhrzeit ein- bzw. ausgeschaltet werden. Die Jahresschaltuhr besitzt 4 Parameter:

Der erste Parameter (PAR1) definiert das Datum, an dem der Einschaltvorgang stattfinden solle. Es ist immer fünfstellig im Format TT.MM anzugeben: "12.03" steht z.B. für den 12. März. Ist für die Jahresschaltuhr kein Einschaltvorgang gewünscht, so ist der entsprechende Parameter nicht zu definieren.

Der zweite Parameter (PAR2) definiert die Einschaltzeit. Die Zeit ist immer fünfstellig im Format hh:mm einzugeben. Für 7 Uhr 30 ist z.B. der Wert "07:30" zu setzen. Wird der Zeitparameter nicht definiert, jedoch das Einschaltdatum, so wird der Einschaltvorgang um 00:00 Uhr ausgeführt.

Analog hierzu definieren der dritte Parameter (PAR3) das Ausschaltdatum und der vierte Parameter (PAR4) die Ausschaltzeit.

Dieser Funktionsblock kann mit dem Befehl "REMON" remanent konfiguriert werden.

Syntax:
```
Label (YCLK)
{
PAR1=12.03
PAR2=07:30
PAR3=01.04
PAR4=23:45
REMON
}
```

Label	Eindeutiger Name für den Funktionsblock. Namen dürfen jeweils nur einmal vergeben werden, müssen mit einem Buchstaben beginnen und dürfen keine Leerzeichen enthalten
(YCLK)	Befehlscode für den Block
PAR1	Definiert das Datum des Einschaltzeitpunkts
PAR2	Definiert die Uhrzeit des Einschaltzeitpunkts
PAR3	Definiert das Datum des Ausschaltzeitpunkts
PAR4	Definiert die Uhrzeit des Ausschaltzeitpunkts

5.1 Befehlsübersicht

Befehl	Beschreibung
ABLNK	Asymmetrischer Taktgeber
ACMP	Analogkomparator
AND	Logische UND-Verknüpfung
ASS	Schwellwertschalter analog
DELOFF	Ausschaltverzögerung
DELON	Einschaltverzögerung
DELONOFF	Ein- und Ausschaltverzögerung
DELSTO	Speichernde Einschaltverzögerung
ECNT	Vor-/Rückwärtszähler
HBCON1	Hochbehältersteuerung (nur Geräte mit Echtzeituhr)
INPUT	Binäreingang einlesen
INV	Inverter
MELDE	Alarmierung über Meldeblock (nur GSM-Geräte)
NAND	Logische Nicht-UND-Verknüpfung
NOR	Logische Nicht-ODER-Verknüpfung
OR	Logische ODER-Verknüpfung
OUTPUT	Binäreingang ausgeben
PUMPCO1	Pumpenüberwachung
RND	Zufallsgenerator
SBLNK	Symmetrischer Taktgeber
SCOMF	Komfortschalter
SHREL	Selbsthalterelais
SSREL	Stromstoßrelais
STAIR	Treppenlichtschalter
TCNT	Betriebsstundenzähler
WCLK	Wochenschaltuhr (nur Geräte mit Echtzeituhr)
WREL	Wischrelais (Impulsausgabe)
WRELF	Flankengetriggertes Wischrelais
XOR	Logische Exclusiv-Oder-Verknüpfung
YCLK	Jahresschaltuhr (nur Geräte mit Echtzeituhr)