MICROCOM
Simply More

# MICROPLC-II Manual

MICROPLC-II
language Programming manual

REV 25.01

MICRO PLC II

## Warning

This system has been designed to be installed by professionals, not by end users. Please contact our experts for any technical queries you may have.

We are continually committed to innovation both in terms of software and hardware. However, errors may result in discrepancies between the product and some of its specifications despite our best efforts to properly document our products. Therefore, please contact us at the following email address should you have any questions or comments: microcom@microcom.es.

GSM-based communications are highly reliable. However, we advise against using our device in critical systems unless some form of redundancy has been implemented for the communication network as the service may become unavailable in rare cases.

"Life Support": This unit is not designed for use in systems on which human life depends. In other words, in devices where a malfunction could pose a risk to human life.

Our liability in relation to the device shall be strictly limited to its repair or replacement in accordance with the terms set forth in the warranty.



Despite every precaution taken during the preparation of this documentation, neither the publisher nor the author assumes any liability for errors or omissions, or for any damages resulting from the use of the information contained in this document. The information contained in this document is subject to change without notice and does not represent any commitment by Microcom Sistemas Modulares, S.L.

The software described in this document is provided under a non-disclosure agreement. This software may be used or copied in accordance with the terms of these agreements.



Microcom Sistemas Modulares, S.L.
C/Gorostiaga, 53 • Irún
GIPUZKOA • 20305
Phone number: +34 943 639 724 • Fax +34 943 017 800
microcom@microcom.es
https://www.microcom360.com

# CONTENTS


DOCUMENT CONTENT ..... 5
VERSIONS AND COMPATIBILITY ..... 5
INFORMATION TABLES ..... 5
1 - GENERAL DESCRIPTION ..... 6
1.1 Access ..... 7
1.2 Interface presentation ..... 7
1.3 MICROPLC control panel ..... 9
2 - RUN AND STOP OPERATING STATUS ..... 10
2.1 Stop ..... 10
2.2 Run ..... 10
2.3 Associated commands. ..... 10
3 - VERIFICATION ..... 11
3.1 Error message format ..... 11
3.2 Description of error messages ..... 11
4 - VARIABLES MONITOR ..... 13
5 - CODE STRUCTURE AND SYNTAX ..... 13
5.1 Syntax ..... 14
6 - INSTRUCTIONS ..... 14
6.1 Instruction IF ..... 14
6.2 REM instruction ..... 15
6.3 ACT instruction ..... 16
7 - OPERATORS ..... 17
7.1 Assignment operators ..... 17
7.2 Arithmetical operators ..... 17
7.3 Mathematical constants ..... 17
7.4 Other operators and functions. ..... 17
7.5 Comparison functions ..... 18
7.6 Comparison operators ..... 18
7.7 Logical operators ..... 18
7.8 Status operators ..... 20
8 - INITIALISATION SECTION: #INIT AND #END_INIT ..... 20
8.1 Assigning aliases to identifiers (optional) ..... 21
9 - VARIABLES ..... 21
9.1 Variable declaration ..... 22
9.2 Assignment of values to variables ..... 23

9.3 Bits manipulation ... 23
9.4 Manipulation of half words ... 24
9.5 Saving and retrieving variables in persistent memory ... 26
10 - DATE AND TIME VARIABLES ... 27
11 - INPUT AND OUTPUT VARIABLES ... 27
11.1 List of identifiers ... 27
11.2 Check and configure GEOGRAPHICAL COORDINATES ... 31
11.3 Check and set alarms (ALM_CCX.PROP) ... 31
12 - TIMER LOCKS ... 34
12.1 Retentive on-delay timer (RTO) ... 36
12.2 Timer On delay (TON) ... 37
12.3 Timer off delay (TOF) ... 39
12.4 Pulse timer (TP) ... 41
12.5 Weekly timer (TW) ... 42
12.6 Cyclic timers (CT) ... 43
13 - SMS SENDING AND RECEIPT MODULE ... 44
13.1 Sending of SMS messages ... 44
13.2 Receipt of SMS ... 46
13.3 Authorised phone number declaration ... 46
14 - MODBUS COMMUNICATION MODULE ... 47
14.1 MODBUS-TCP SERVER IDENTIFIER IMPORT ... 48
15 - PUMP CONTROL MODULE (PUMP) ... 50
16 - PID MODULE ... 55
16.1 PID module settings ... 57
17 - PROGRAMMING EXAMPLES ... 60
17.1 Thermostat ... 60
17.2 Twilight or astronomical switch ... 62
17.3 Pump start-up automation ... 64
17.4 Pump control in primary/backup mode ... 69
17.5 PID pressure/flow control ... 77
17.6 MODBUS memory map configuration in slave mode ... 81

## DOCUMENT CONTENT

This manual describes the MicroPLC-II automation programming language, which is available for devices in the Hermes series.

Throughout this document, readers are assumed to be familiar with the MicroConf configuration tool and the general settings of these devices.

These manuals can be found in the downloads section of the Microcom website. http://www.microcom360.com/start

A set of video tutorials on device configuration and other videos of interest are available on our YouTube channel.

## VERSIONS AND COMPATIBILITY

The information found in this document corresponds to the version of the MicroConf configuration software and Hermes firmware as indicated below:

| Element | Version |
|---|---|
| Microconf Software | v.9.3.3 |
| Firmware | v.9.38 |

## INFORMATION TABLES

The following information tables are used throughout the manual:

| | |
|---|---|
| | Note: These are used to highlight information of special importance or interest. |
| | Attention: These are used to describe the conditions, practices or procedures that must be followed for the proper use of hardware or software. |
| | Example: These are used to show practical examples and therefore help to better understand the text described in the section. |

# 1 - GENERAL DESCRIPTION

MicroPLC-II is an automation programming language that stands out for its balance between ease of learning and power. Inspired by classic automation programming languages, MicroPLC-II simplifies the syntax and offers native logic modules that facilitate the implementation of solutions for common issues, such as pump control and communication between stations, to name a few.

One key feature of MicroPLC-II is its fixed 1-second scan cycle, regardless of the complexity or size of the program.

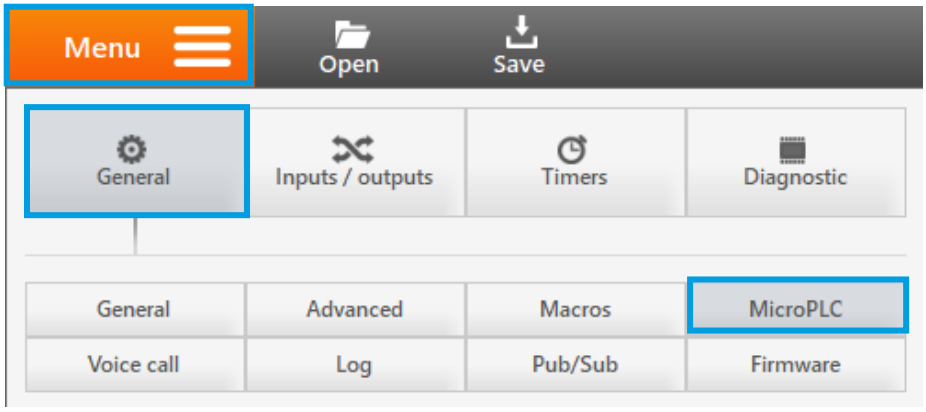
One example of SCRIPT for the automatic activation of a pump according to the tank level:

```
L1    #INIT
L2    REM EXAMPLE CONFIGURATION SECTION
L3    AI0 : TANK_LEVEL
L4    DO0 : PUMP
L5    #END_INIT

L6    REM EXAMPLE MAIN PROGRAM LOOP
L7    IF TANK_LEVEL >= 3.5 ; PUMP = 0
L8    IF TANK_LEVEL <= 1.5 ; PUMP = 1
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Comment to describe the program. This line does not run. |
| L3 | Assigns the alias “TANK_LEVEL” to the analogue input 0 “AI0”.<br>A sensor that measures the water level of a tank has been connected and configured at analogue input 0. |
| L4 | Assigns the alias “PUMP” to the digital output 0 “DO0”.<br>Digital output relay 0 controls the turning on and off of a water pumping motor that refills the water tank. |
| L5 | Keyword to finish the initialisation section. |
| L6 | Comment to describe the program. This line does not run. |
| L7 | “PUMP” is deactivated if the value in “TANK_LEVEL” is greater than or equal to 3.5 metres. |
| L8 | “PUMP” is activated if the value in “TANK_LEVEL” is less than 1.5 metres. |

## 1.1 ACCESS

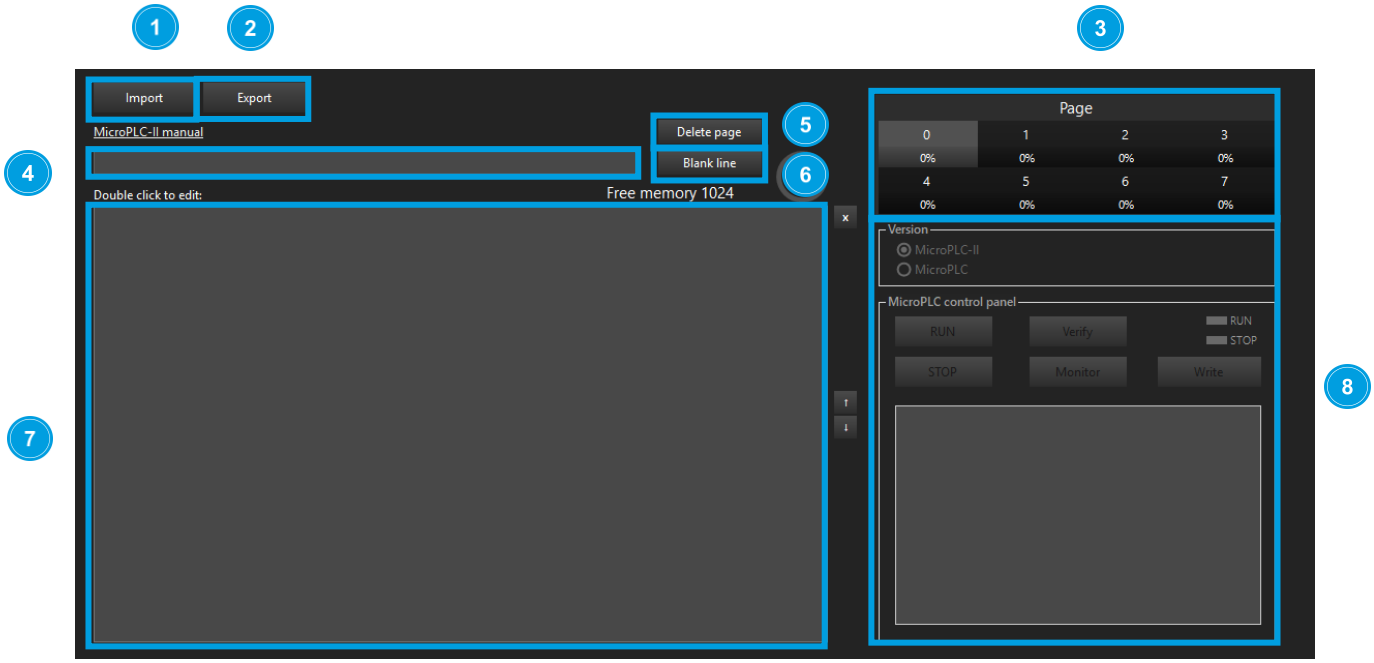


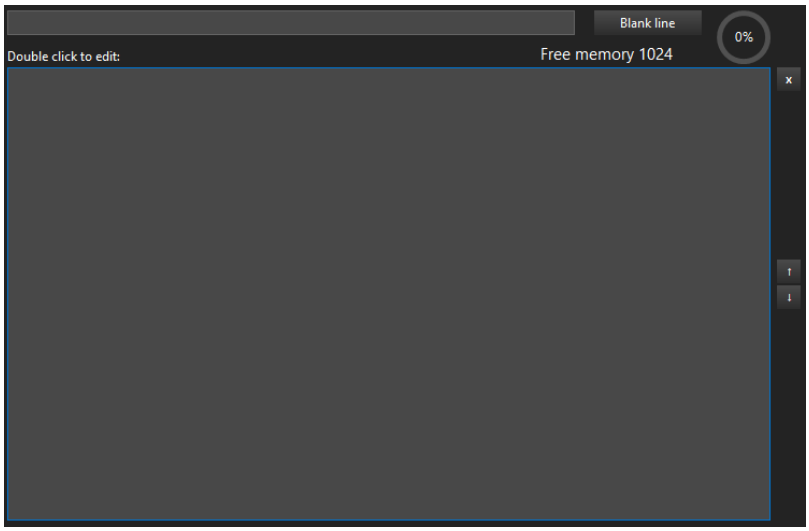

Menu → General → MicroPLC

Firmware version 9 (and later) and MICROCONF version 9 (and later) integrate MicroPLC-II.

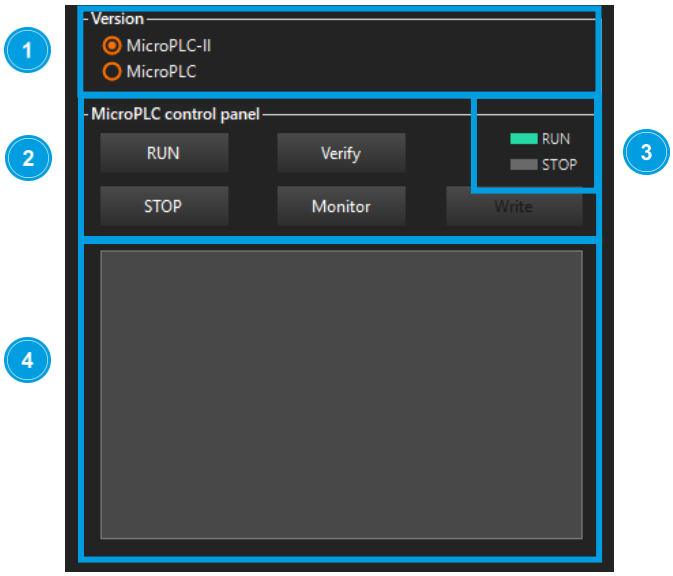
MicroPLC-II compatible devices are also backwards compatible with MicroPLC language.

## 1.2 INTERFACE PRESENTATION



| Item | Field | Description |
|---|---|---|
| 1 | Import | Allows you to import a plain text file (.TXT) with lines of code in MICROPLC-II language.<br>This file must include all pages. |
| 2 | Export | Allows you to export all lines of MICROPLC-II code to a plain text file (.TXT).<br>This file includes the information from all the pages. |
| 3 | Page selector | This allows you to navigate between pages and find out the percentage of use of each page. |
| 4 | Code input box | Program line.<br>Press the "ENTER" button to add the code to the page script. |
| 5 | Delete page | Allows you to delete the code from the selected page |
| 6 | Blank line | Allows you to add a blank line to the code. |
| 7 | Code panel | Lines of code or script.<br>  <br>Upper section: Amount of free memory.<br>Right side: Buttons to change the order of the lines of code and delete them. |
| 8 | MicroPLC control panel | See the following section. |

## 1.3 MICROPLC CONTROL PANEL




| Item | Field | Description |
|---|---|---|
| 1 | Language selector | Programming language to be used.<br>MicroPLC-II: Version available from firmware version 9. Described in this manual.<br>MicroPLC: Version available in previous firmware versions, which is maintained by backwards compatibility. |
| 2 | Buttons | Buttons enabled for MICROPLC-II language only:<br>RUN: Sets the run mode. See Section 2.1.<br>STOP: Sets safe mode. The code is not running. See Section 2.2.<br>Verify: Verification button. Reviews the program and looks for errors in the syntax. See Section 3.<br>The system needs to be in STOP in order to verify the Script.<br>Monitor: Opens the variable monitor window. See Section 4.<br>Write: Writes the programming code on the device. |
| 3 | Operational status | Indicates the status of the device. |
| 4 | Terminal | Message area. Shows error details and traces. Clicking on an error message highlights the line on which the error is located in the code panel. See Section 3. |

# 2 - RUN AND STOP OPERATING STATUS

Indicates whether you are running the programmed code in MICROPLC II.

## 2.1 STOP

Safe status. The device enters this state manually, by pressing the "STOP" button, or automatically, if it detects a code failure.

The device is not running the program, and all outputs remain disabled.

This status allows you to review the programming and installation safely and load new programs.

## 2.2 RUN

Run status. The device automatically enters this mode after switching on or when the "RUN" button is clicked.

The device:

- Will run the lines that are located within the initialisation section (# INIT ...#END_INIT) for all the pages.
- The code of every page will then be cyclically executed, from left to right and from top to bottom (once per second), reading the inputs, running all the program conditions, and activating the necessary outputs at all times.

## 2.3 ASSOCIATED COMMANDS.

The following commands are available for MicroPLC-II functionality management:

| Command | Description |
|---|---|
| PLCRUN | Sets RUN mode. |
| PLCSTOP | Sets STOP mode. |

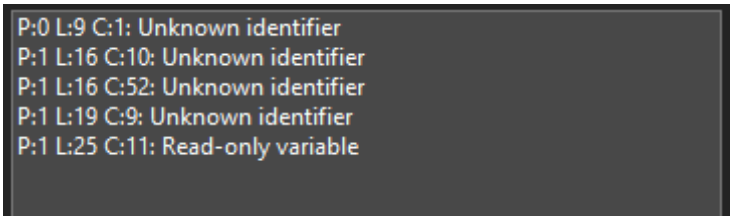
# 3 - VERIFICATION

The MICROCONF configuration software integrates a code reviewer that returns the following error messages.

## 3.1 ERROR MESSAGE FORMAT

P:*00* L:*00* C:*00*: *Error message*

Where:
- P:xx: xx indicates the programming page.
- L:xx: indicates the line.
- C:xx: xx indicates the column.
- Error message: See next section for more information.



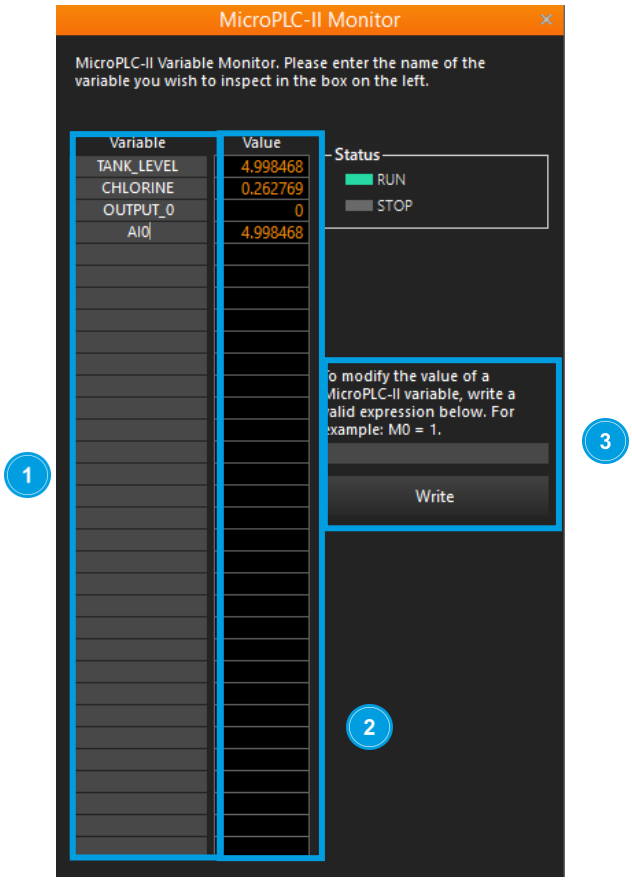

## 3.2 DESCRIPTION OF ERROR MESSAGES

| Error message | Description |
|---|---|
| Variable name or alias already exists | A previously declared variable name or alias declaration has been found. |
| Index out of range | The index is out of range; e.g., F32 is an out-of-range index since the flags cover F0 to F31. |
| Invalid variable name or alias | An invalid variable name or alias declaration has been found. The most common reasons are the use of a reserved name or the presence of invalid characters. |
| Invalid property | The object does not have any properties with the given name, e.g.: DI0.Q. 'Q' is not a valid DIx property. |
| No memory available for new variable | There is no memory space left for the declared variable. |
| Read-only variable | An assignment to a read-only variable has been found, e.g., assign 'DI0 = 1'. 'DI0' returns the value of the digital input 0, meaning it cannot be written. |
| Syntax error | A syntax error was found. |
| Unknown identifier | An operation with an undeclared variable or alias was found. |

| | |
|---|---|
| Variable redefinition | A variable name or alias has been declared more than once, e.g.:<br>BOOL : BOYA_LOW<br>INT : BOYA_LOW |
| Parameter not found | This error occurs when attempting to reference a variable or parameter that does not exist on the device. |
| Property not allowed in alias definition | An attempt was made to create an alias that includes a property modifier (e.g., AI0.MA). The valid syntax for declaring an alias is only the signal name, without including any property. |

# 4 - VARIABLES MONITOR

The variables monitor allows the inspection and modification of MicroPLC-II variables in real time for code debugging purposes.




| Item | Field | Description |
|---|---|---|
| 1 | Variable | Name of the variables to be inspected. Both the variables declared within MICROPLC-II and the internal variables of the system are valid. |
| 2 | Value | Value of the variable. The ? character indicates that the variable is not recognised. |
| 3 | Write | Modification of variables in real time. |

# 5 - CODE STRUCTURE AND SYNTAX

Each line of code can contain one or more instructions, separated by a semicolon (";"). Instructions are executed sequentially: the code runs from beginning to end on each line, unless it encounters an unmet "IF" condition. In this case, execution is interrupted and continues on the next line.

The code runs once per second. Page 0 is executed, followed by page 1, and so on until the last page is reached.

### 5.1 SYNTAX

A correct code in MICROPLC-II follows the following rules:

- A command ends at the end of the line or with a semicolon ";".
- The decimal separator is a full stop.
  Example:

```
LEVEL = 3.25
```

- Variable names can contain Latin letters, numbers, and underscores.
  Example:

```
AI0 : TANK_LEVEL_2
```

- Variable names cannot start with a digit.
- Blank spaces are allowed between statements and operators, and line breaks are allowed between lines of code.

```
L1    #INIT
L2    REM EXAMPLE CONFIGURATION SECTION
L3    AI0 : TANK_LEVEL
L4    DO0 : PUMP
L5    #END_INIT

L6    REM EXAMPLE MAIN PROGRAM LOOP
L7    IF TANK_LEVEL >= 3.5 ; PUMP = 0
L8    IF TANK_LEVEL <= 1.5 ; PUMP = 1
```

# 6 - INSTRUCTIONS

### 6.1 INSTRUCTION IF

The following instructions are executed if the specified condition is met. Otherwise, the program moves to the next line.

SYNTAX

```
IF condition; instructions…
```

Where:
- Condition: An expression the result of which will be evaluated to determine whether the code should be executed. Any result other than 0 is considered TRUE.
- Instructions: One or more instructions that follow the IF statement and are executed if the condition is evaluated as TRUE.

Example:

```
IF LEVEL < 3.25 ; LOW_WATER = 1
```

## 6.2 REM INSTRUCTION

Used to include a comment in the code.

SYNTAX

```
REM comment
```

Where:
- Comment: Free text that will not be evaluated.

Example:

```
REM DECLARE VARIABLES
```

## 6.3 ACT INSTRUCTION

Instructions for executing actions from MicroPLC-II.

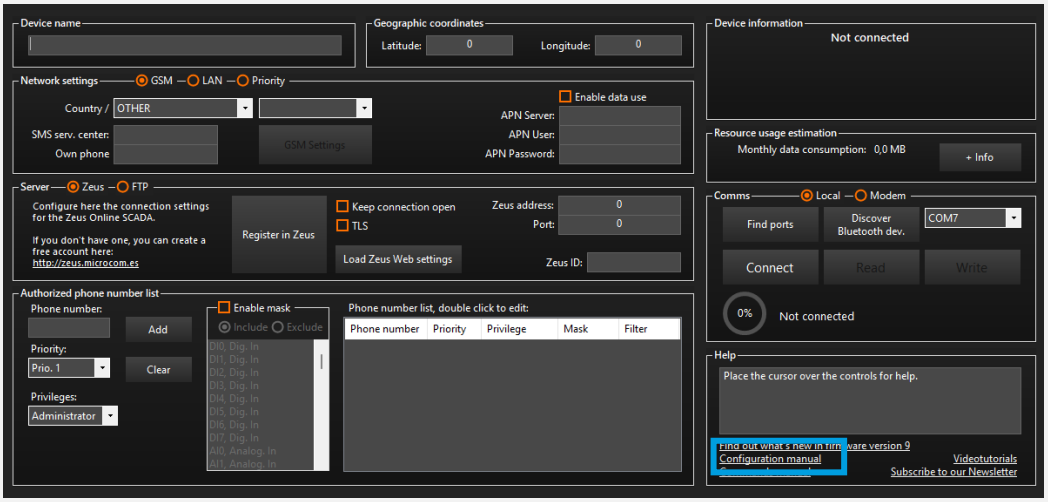
SYNTAX

```
ACT actions
```

Where:

- Actions: List up to 8 actions, separated by commas.

You can find the available actions in Appendix A of the configuration software manual, at www.microcom360.com.or in the general configuration software menu.




Example:

```
ACT 79
ACT 79,53,75,65,57,111,98,99
```

# 7 - OPERATORS

## 7.1 ASSIGNMENT OPERATORS

| | Description | Example |
|---|---|---|
| : | This is used when declaring variables, aliases, and other objects. | INT : MAX_TEMPERATURE |
| = | This assigns the value of the right-hand term to the left-hand operator. | INT : MAX_TEMPERATURE= 27 |

## 7.2 ARITHMETICAL OPERATORS

| | Description | | Description |
|---|---|---|---|
| + | Add | ^ | Exponentiation |
| - | Subtract | \ | Module. Integer of a division |
| * | Multiply | SQRT() | Square root |
| / | Divide | | |

## 7.3 MATHEMATICAL CONSTANTS

| | Description | | Description |
|---|---|---|---|
| TRUE | Equivalent to a logical value 1 | PI | PI Number |
| FALSE | Equivalent to a logical value 0 | E | The number “e”, Euler's number |

## 7.4 OTHER OPERATORS AND FUNCTIONS.

| | Description | | Description |
|---|---|---|---|
| SIN( ) | Sine | ASIN( ) | Arc sine |
| COS( ) | Cosine | ACOS( ) | Arc cosine |
| TAN( ) | Tangent | ATAN( ) | Arc Tangent |
| EXP( ) | Exponentiation | LN( ) | Napierian logarithm |
| ABS( ) | Absolute value | LOG( ) | Decimal logarithm |
| INT( ) | Returns the integer part of a value | FRAC( ) | Returns the decimal part of a value |
| RAND | Random number generator Range: 0… 1 | | |

## 7.5 COMPARISON FUNCTIONS

| | Description |
|---|---|
| MIN(n1,n2) | Compares two values (n1 and n2) and returns the lowest value. |
| MAX(n1,n2) | Compares two values (n1 and n2) and returns the highest value. |

## 7.6 COMPARISON OPERATORS

| Description | | Example (*) | Result example |
|---|---|---|---|
| > | More than | FLAG = LEVEL > 3 | If "LEVEL" is greater than 3, "FLAG" will obtain the value 1.<br>If "LEVEL" is less than 3 it will obtain the value 0. |
| < | Less than | FLAG = LEVEL < 4 | If "LEVEL" is less than 4, "FLAG" will obtain the value 1.<br>If "LEVEL" is greater than 4 " it will obtain the value 0. |
| = | Equal to | FLAG = LEVEL = 2 | If "LEVEL" is equal to 2, "FLAG" will obtain the value 1.<br>If "LEVEL" is other than 2 it will obtain the value 0. |
| >= | Greater than or equal to | FLAG = LEVEL >= 3 | If "LEVEL" is greater than or equal to 3, "FLAG" will obtain the value 1. If "LEVEL" is less than 3 it will obtain the value 0. |
| <= | Less than or equal to | FLAG = LEVEL <= 4 | If "LEVEL" is less than or equal to 4, "FLAG" will obtain the value 1. If "LEVEL" is greater than 4, it will obtain the value 0. |
| <> | Other than | FLAG = LEVEL <> 5 | If "LEVEL" is other than 5, "FLAG" will obtain the value 1.<br>If "LEVEL" is equal to 5 it will obtain the value 0. |

(*) In the examples, the FLAG variable is assumed to be a BOOL type, and the LEVEL variable a REAL type.

## 7.7 LOGICAL OPERATORS

| | Description | Example (*) | Result example |
|---|---|---|---|
| NOT | Reverses the result of the expression that follows immediately. | DO0 = NOT(LEVEL>3.25) | If "LEVEL" is greater than 3.25, DO0 will obtain the value 0. If "LEVEL" is less than or equal to 3.25, DO0 will obtain the value 1. |
| AND | Logical AND | FLAG = OVERFLOW AND PUMP_ON | FLAG will only be true when both OVERFLOW and PUMP_ON are true. |

| | Description | Example (*) | Result example |
|---|---|---|---|
| OR | Logical OR | FLAG = OVERFLOW OR PUMP_ON | FLAG will be true if either OVERFLOW or PUMP_ON is true. |
| & | Binary AND | A = B & C | If B=0101 and C=1100, A = 0100<br>0.101<br>& 1100<br>--------<br>0.100 |
| \| | Binary inclusive OR | A = B \| C | If B=0101 and C=1100, A = 0100<br>0.101<br>\| 1100<br>--------<br>1.101 |
| ~ | Supplements one. Each bit that is 1 in the the operand is 0 in the result and vice versa. | A = ~B | If B contains the value 39 (00100111 in binary), A will contain the value 216 (11011000 in binary). |
| >>n | Bit shifting 'n' positions to the right. | TEMP = LEVEL >> 2 | If LEVEL contains the value 8 (1000 in binary), TEMP will contain the value 2 (0010 in binary). |
| <<n | Bit shifting 'n' positions to the left. | TEMP = LEVEL << 1 | If LEVEL contains the value 2 (0010 in binary), TEMP will contain the value 4 (0100 in binary). |

(*) In the examples" the "FLAG", "OVERFLOW" and "PUMP_ON" variable is assumed to be BOOL type. The "LEVEL" and "TEMP" variable are defined as REAL.

## 7.8 STATUS OPERATORS

| | Description |
|---|---|
| PWR.VIN | Returns the external supply voltage. |
| PWR.VBAT | Returns the voltage value from the internal battery with the highest voltage. Compatible with all Nemos models. |
| PWR.VBAT1 | Returns the voltage value of the internal battery connected to port 1. Compatible with Nemos N200 and N200+. |
| PWR.VBAT2 | Returns the voltage value of the internal battery connected to port 2. Compatible with Nemos N200 and N200+. |
| PWR.FAIL | Returns 1 in the event of a power outrage and the device is running off its internal battery. Compatible with Hermes LC2+ and TCR210. |
| PWR.BL | (Battery Life) Returns a value between 0 and 2 (inclusive), which represents the status of the internal battery in a colour code, where: 0=Green (OK), 1=Orange (less than 3 months), 2=Red (Out of stock). In the case of two battery packs, it shows the one with the best status. Compatible with all Nemos models. |
| GSM | RSSI or GSM signal field strength. The device returns a numeric value between 1 and 32. The recommended minimum is 8.<br>Conversion equation to dBm: dBm = -113 + N * 2 (where N is the return value) |
| STAT.MBCE | Returns a 1 if there is a MODBUS communication error. |
| STAT.EXCE | Returns a 1 if there is a communication error with expansion modules. |
| STAT.PBCE | Returns a 1 if there is a communication error with the Microcom STDV01/STDV02 digital probes. |
| TEMP | The device's internal temperature. Compatible with all Nemos models. |

Note: While MicroPLC-II is only available on Herme devices, the status operators for Nemos devices have been included to be in line with the configuration manual.

# 8 - INITIALISATION SECTION: #INIT AND #END_INIT

#INIT and #END_INIT are keywords used to start and end the initialisation section in the code. There can be one initialisation section per page. This section is used to declare variables, timer locks, alias allocation, and other elements. Both keywords will be displayed in orange. Including the initialisation section in the code is optional.

The initialisation section is executed only once under the following circumstances: when power is applied to the device, and when moving from STOP mode to RUN mode (see Section 2.) or when loading a new configuration or script.

It runs as follows: the initialisation section of page 0 is firstly executed, followed by page 1, and so on until the final page. The regular code then beings running at a frequency of once per second.

### 8.1 ASSIGNING ALIASES TO IDENTIFIERS (OPTIONAL)

External interfaces, mathematical registers, and flags can be assigned an "alias" using the ":" symbol.

Aliases do not support accents or other special characters, such as the Spanish letter "Ñ".

Example of assigning an alias in the initialisation section.

```
L1 #INIT
L2 INT : TEMPERATURE
L3 DI0 : PUMP_ON
L4 TON : TEMPO, PT = 60
L5 #END_INIT
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Declares "TEMPERATURE" as a 32-bit signed integer variable. |
| L3 | Assigns the alias PUMP_ON to the digital input 0 "DI0". |
| L4 | Declares TEMPO as a timer on delay. Configures the timer to expire after 60 seconds. |
| L5 | Keyword to finish the initialisation section. |

## 9 - VARIABLES

MicroPLC-II allows you to define up to 64 BOOL type variables and up to 64 additional variables, combining REAL or INT ones. These variables are volatile, meaning their values are lost if the power supply is interrupted.

| Type | Description | Range |
|---|---|---|
| BOOL | Boolean type variable. | 0 … 1 |
| INT | 32-bit signed integer variable. | -2,147,483,648 … +2,147,483,647 |
| REAL | Single-precision floating-point type variable. | $1{,}210^{-38}$… $3{,}410^{38}$ |

## 9.1 VARIABLE DECLARATION

The ":" symbol is used to declare internal variables. Optionally, an initial value can be assigned to the variable on the same line of code using the "=" symbol. If no initialisation value is specified, the variable is initialised to 0. Variables must be declared within the initialisation section.

Variable names do not support accents or other special characters, such as the Spanish letter "Ñ".

Example of a variable declaration.

```
L1 #INIT
L2 BOOL : PUMPON
L3 BOOL : FAILURE = TRUE
L4 INT : COUNTER
L5 INT : AUTO = 2
L6 REAL : MAXLEVEL
L7 REAL : LEVEL = 5.5
L8 #END_INIT
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Declares "PUMPON" as a Boolean variable. The variable is initialised to 0 when nothing else is specified. |
| L3 | Declares "FAILURE" as a Boolean variable and initialise it to 1 (TRUE). |
| L4 | Declares "COUNTER" as a 32-bit signed integer variable. The variable is initialised to 0 when nothing else is specified. |
| L5 | Declares "AUTO" as a 32-bit signed integer variable and initialises it to 2. |

| Line | Description |
|---|---|
| L6 | Declares "MAXLEVEL" as a single-precision floating-point variable. The variable is initialised to 0 when nothing else is specified. |
| L7 | Declares "LEVEL" as a single-precision floating-point variable and initialises it to 5.5. |
| L8 | Keyword to finish the initialisation section. |

## 9.2 ASSIGNMENT OF VALUES TO VARIABLES

The "=" symbol is used to modify declared variables.

Example of assigning a value to an internal variable.

```
L1 #INIT
L2 INT : MAX_TEMPERATURE
L3 INT : TIMECOUNTER
L4 #END_INIT

L5 MAX_TEMPERATURE = 27
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Declares "MAX_TEMPERATURE" as a 32-bit signed integer variable. |
| L3 | Declares "TIMECOUNTER" as a 32-bit signed integer variable. |
| L4 | Keyword to finish the initialisation section. |
| L5 | Sets the value of the variable "MAX_TEMPERATURE" to 27. |

## 9.3 BITS MANIPULATION

INT type variables can be manipulated on a bit level using the ".Bx" property, where "x" represents the number of the bit to be modified. Bit 0 is the least significant within this context while bit 31 is the most significant bit.

The ".Bx" property enables both reading and writing.

Creates a variable that stores the status of digital input channels, MODBUS, and expansions.

```
L1  #INIT
L2  INT : INPUT_STATUS
L3  #END_INIT
L4  INPUT_STATUS.B0 = DI0
L5  INPUT_STATUS.B1 = DI1
L6  INPUT_STATUS.B2 = DI2
L7  INPUT_STATUS.B3 = DI3
L8  INPUT_STATUS.B4 = MB0
L9  INPUT_STATUS.B5 = MB1
L10 INPUT_STATUS.B6 = EXP0
L11 INPUT_STATUS.B7 = EXP1
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Declares "INPUT_STATUS" as a 32-bit signed integer variable. |
| L3 | Keyword to finish the initialisation section. |
| L4 | Sets "INPUT_STATUS" bit number 0 to the digital input 0 value. |
| L5 | Sets "INPUT_STATUS" bit number 1 to the digital input 1 value. |
| L6 | Sets "INPUT_STATUS" bit number 2 to the digital input 2 value. |
| L7 | Sets "INPUT_STATUS" bit number 3 to the digital input 3 value. |
| L8 | Sets "INPUT_STATUS" bit number 4 to the MODBUS 0 channel value. |
| L9 | Sets "INPUT_STATUS" bit number 5 to the MODBUS 1 channel value. |
| L10 | Sets "INPUT_STATUS" bit number 6 to the expansions 0 channel value. |
| L11 | Sets "INPUT_STATUS" bit number 7 to the expansions 1 channel value. |

## 9.4 MANIPULATION OF HALF WORDS

INT and REAL variables can be manipulated at a half-word level using the ".H" and ".L" properties. The ".H" property accesses the 16 most significant bits, while the ".L" property accesses the 16 least significant bits.

Both the ".H" and ".L" properties enable both reading and writing.

This functionality is particularly useful for defining the MODBUS map in slave mode.

INT variables, accessed through user registers, are read as a 32-bit integer, while REAL variables are read as a single-precision floating-point number (FLOAT).

Maps the value of analogue input 1 in the address 30001 as a single-precision floating-point number in "Big Endian" mode.

```
L1 #INIT
L2 REAL : LEVEL
L3 #END_INIT
L4 LEVEL = AI0
L5 MBIR0 = LEVEL.L
L6 MBIR1 = LEVEL.H
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Declares "LEVEL" as a 32-bit signed integer variable. |
| L3 | Keyword to finish the initialisation section. |
| L4 | Stores the value of analogue input 0 "AI0" in "LEVEL". |
| L5 | Stores the 16 least significant bits of the "LEVEL" variable in user register 0 to allow them to be accessible from address 30001. |
| L6 | Stores the 16 most significant bits of the "LEVEL" variable in user register 1 to allow them to be accessible from address 30002. |

See Section 16.4 for other examples of how this is used.

## 9.5 SAVING AND RETRIEVING VARIABLES IN PERSISTENT MEMORY

BOOL, INT and REAL variables are volatile, and their value is lost if the electrical current is interrupted. The value of these variables can be made persistent by saving and retrieving their value in FLAGS or MATH REGISTERS, which are non-volatile memory registers.

Configuration of a start-up counter, storing the counter value in non-volatile memory.

The counter value increases each time there is a new start-up, and the value is stored in non-volatile memory to prevent information loss in the event of a power outage.

```
L1 #INIT
L2 INT : COUNTER
L3 INT : PRE_PUMP
L4 COUNTER = M0
L5 EXP16 : PUMP
L6 #END_INIT
L7 IF PUMP = TRUE AND PUMP <> PRE_PUMP ; COUNTER = COUNTER + 1
L8 PRE_PUMP = PUMP
L9 M0 = COUNTER
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Declares "COUNTER" as a 32-bit signed integer variable. |
| L3 | Declares "PRE_PUMP" as a 32-bit signed integer variable. |
| L4 | Initialises the variable "COUNTER" with the "M0" value. The first time the program runs, "M0" will be equal to 0, but if the program resumes after a power outage, it will have the value assigned in L9. |
| L5 | Assigns the alias "PUMP" to the expansion channel 16. |
| L6 | Keyword to finish the initialisation section. |
| L7 | If "PUMP" is enabled and its value is different from the "PRE_PUMP" value, this means that a new start-up has occurred, and we add 1 to the "COUNTER" variable. |
| L8 | Stores the "PUMP" value in "PRE_PUMP" for the next check. |
| L9 | Stores the value of "COUNTER" in the math register 0 "M0". We store this value in non-volatile memory to prevent it from being lost in the event of a power outage. The value will be restored in L4 in the event of a power outage. |

See Section 16.3. for other examples of how this is used.

# 10 - DATE AND TIME VARIABLES

| Variable | Description |
|---|---|
| NOW.Y | Returns a four-digit number representing the current year. |
| NOW.MO | Returns a number between 1 and 12 (inclusive), representing the month of the current year. |
| NOW.D | Returns a number between 1 and 31 (inclusive), representing the day of the current month. |
| NOW.WD | Returns a number between 1 and 7 (inclusive), representing the day of the week:<br>Monday=1, Tuesday=2, Wednesday=3, Thursday=4, Friday=5, Saturday=6, Sunday=7 |
| NOW.YD | Returns a number between 1 and 366 (inclusive), representing the day of the current year. |
| NOW.HHMM | Returns the current time of day in 24hr format, where:<br>HH: returns a number between 0 and 23 (inclusive)<br>MM: returns a number between 0 and 59 (inclusive) |
| NOW.SOD | (Seconds Of Day). Returns a number between 0 and 86399 (inclusive), representing the elapsed seconds of the current day. |
| SUMMER | Returns a number between 0 and 1 (inclusive) indicating the time change: 0 for winter, 1 for summer. |
| SUNRISE | Returns a number between 0 and 86399 (inclusive), which represents the time sunrise takes place for the current day in seconds. The geographical coordinates must be provided to use this mathematical operator. |
| SUNSET | Returns a number between 0 and 86399 (inclusive), which represents the time sunset takes place for the current day in seconds. The geographical coordinates must be provided to use this mathematical operator. |

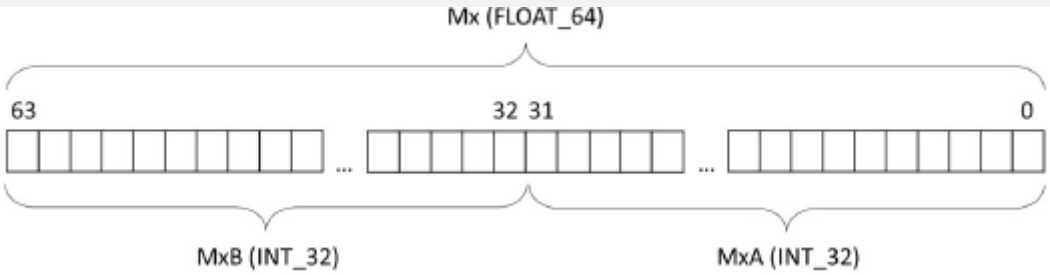
# 11 - INPUT AND OUTPUT VARIABLES

Each channel, as well as the mathematical registers and flags, has an associated identifier.

## 11.1 LIST OF IDENTIFIERS

The "x" should be replaced in all identifiers with the input, output, or channel number.

| Identifier | Description |
|---|---|
| Mx | Value of the math register x. A math register is a 64-bit double-precision floating-point variable. They are non-volatile variables, and their values are not lost if the electrical flow is interrupted. MICROPLC-II-compatible devices have 32 math registers. |
| MxA<br>MxB | Each 64-bit double-precision floating-point math register can be used as two 32-bit signed integer math register. Where MxA is the least significant 32 bits and MxB is the most significant 32 bits. This doubles the number of math registers available. |

<table>
<tr><th>Identifier</th><th>Description</th></tr>
<tr><td></td><td> 

Example:<br>M0A = M0A + 1<br>M0B = M0B + 1</td></tr>
<tr><td>Fx</td><td>Flag x status.</td></tr>
<tr><td>FLAG</td><td>Word containing the status of the device's 32 FLAGS. In binary representation the least significant bit corresponds to FLAG 0 and the most significant bit corresponds to FLAG 31.


<table>
<tr><th>Active flag</th><th>Decimal</th><th>Binary</th></tr>
<tr><td>None</td><td>0</td><td>0000 0000 0000 0000 0000 0000 0000 0000</td></tr>
<tr><td>0</td><td>1</td><td>0000 0000 0000 0000 0000 0000 0000 0001</td></tr>
<tr><td>1</td><td>2</td><td>0000 0000 0000 0000 0000 0000 0000 0010</td></tr>
<tr><td>2</td><td>4</td><td>0000 0000 0000 0000 0000 0000 0000 0100</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
<tr><td>29</td><td>536870912</td><td>0010 0000 0000 0000 0000 0000 0000 0000</td></tr>
<tr><td>30</td><td>1073741824</td><td>0100 0000 0000 0000 0000 0000 0000 0000</td></tr>
<tr><td>31</td><td>2147483648</td><td>1000 0000 0000 0000 0000 0000 0000 0000</td></tr>
</table>
Example:<br>If Flag 1 (second bit activated) and Flag 4 (fifth bit activated) are activated, the word FLAG returns an 18: FLAG = 18 (10010 binary)</td></tr>
<tr><td>DIx</td><td>Digital input status x. Electrical status that does not consider the NO/NC configuration of the input.</td></tr>
<tr><td>DIx.AC</td><td>Accumulated time in seconds that the digital input x has been activated (hour count).</td></tr>
<tr><td>DIx.T</td><td>Activation time in seconds since the last activation with digital input x.</td></tr>
</table>

<table>
<tr><th>Identifier</th><th>Description</th></tr>
<tr><td>DIx.H</td><td>True if the digital input x is in an active state, according to the NO/NC configuration of that input.</td></tr>
<tr><td>DIx.L</td><td>True if the digital input x is in an inactive state, according to the NO/NC configuration of that input.</td></tr>
<tr><td>CNTx</td><td>Raw value without conversion of counter x.</td></tr>
<tr><td>CNTx.H</td><td>16 bits heavier than the raw value without conversion of the counter x.</td></tr>
<tr><td>CNTx.L</td><td>16 bits lighter than the raw value without conversion of the counter x.</td></tr>
<tr><td>CNTx.M3</td><td>Value converted to cubic metres of the counter x.</td></tr>
<tr><td>DAYFLOW(x)</td><td>Value in cubic metres of the volume accumulated over 24hrs of the flowmeter x. Start time 12 a.m. and end time 11.59 p.m.</td></tr>
<tr><td>FRx</td><td>Flow value x</td></tr>
<tr><td>FRx.IM</td><td>Instantaneous value of the flow x.</td></tr>
<tr><td>DIP</td><td>Status of the digital input port. This allows you to read the status of several digital inputs at once by adding their decimal values. In binary representation, the least significant bit is digital input 0 and the most significant bit is digital input 7.
<table>
<tr><th>Digital input active</th><th>Decimal</th><th>Binary</th></tr>
<tr><td>None</td><td>0</td><td>0000 0000</td></tr>
<tr><td>0</td><td>1</td><td>0000 0001</td></tr>
<tr><td>1</td><td>2</td><td>0000 0010</td></tr>
<tr><td>2</td><td>4</td><td>0000 0100</td></tr>
<tr><td>3</td><td>8</td><td>0000 1000</td></tr>
<tr><td>4</td><td>16</td><td>0001 0000</td></tr>
<tr><td>5</td><td>32</td><td>0010 0000</td></tr>
<tr><td>6</td><td>64</td><td>0100 0000</td></tr>
<tr><td>7</td><td>128</td><td>1000 0000</td></tr>
</table>
Example: If digital inputs 1 (2 decimal) and 4 (16 decimal) are enabled, the DIP identifier returns an 18: DIP=18 (0001 0010 in binary).</td></tr>
<tr><td>AIx</td><td>Engineering unit value of analogue input x.</td></tr>
<tr><td>AIx.MA</td><td>Value in mA of analogue input x.</td></tr>
</table>

<table>
<tr><th>Identifier</th><th>Description</th></tr>
<tr><td>AIx.VLD</td><td>Validity flag of analogue input x. Will return 1 if the current in the loop is greater than 3.8mA.</td></tr>
<tr><td>DOx</td><td>Status of digital output x.</td></tr>
<tr><td>DOP</td><td>Digital output port status. Returns the status of several digital outputs at once by adding their decimal values. The least significant bit in binary is digital output 0 and the most significant bit is digital output 7.
<table>
<tr><th>Digital output active</th><th>Decimal</th><th>Binary</th></tr>
<tr><td>None</td><td>0</td><td>0000 0000</td></tr>
<tr><td>0</td><td>1</td><td>0000 0001</td></tr>
<tr><td>1</td><td>2</td><td>0000 0010</td></tr>
<tr><td>2</td><td>4</td><td>0000 0100</td></tr>
<tr><td>3</td><td>8</td><td>0000 1000</td></tr>
<tr><td>4</td><td>16</td><td>0001 0000</td></tr>
<tr><td>5</td><td>32</td><td>0010 0000</td></tr>
<tr><td>6</td><td>64</td><td>0100 0000</td></tr>
<tr><td>7</td><td>128</td><td>1000 0000</td></tr>
</table>
Example: If digital outputs 2 (4 decimal) and 3 (8 decimal) are enabled, the DOP operator returns a 12: DOP=12 (0000 1100 in binary).</td></tr>
<tr><td>AOx</td><td>Value of analogue output x.</td></tr>
<tr><td>AOx.MA</td><td>Value in milliamperes of the current in the loop of analogue output x.</td></tr>
<tr><td>MBx</td><td>Value of the MODBUS channel x.</td></tr>
<tr><td>MBx.VLD</td><td>Validity flag of the MODBUS channel x. It will return 1 if the reading is successful.</td></tr>
<tr><td>MBIRx</td><td>Value of the MODBUS INPUT REG x. See section 16.4</td></tr>
<tr><td>EXPx</td><td>Value of the expansion x.</td></tr>
<tr><td>EXPx.T</td><td>Time in seconds since the last activation of the expansion channel x in digital input mode.</td></tr>
<tr><td>EXPx.H</td><td>True if the expansion xin digital input mode is in an active state, according to the NO/NC configuration of that input. This is equivalent to EXPx and is provided to maintain consistency with the digital inputs of the main module (DIx.H).</td></tr>
<tr><td>EXPx.L</td><td>True if the expansion x in digital input mode is in an inactive state, according to the NO/NC configuration of that input.</td></tr>
<tr><td>EXPx.VLD</td><td>Validity flag of the expansion x. It will return 1 if the reading is successful.</td></tr>
</table>

| Identifier | Description |
|---|---|
| | The value in the loop for analogue expansions in current mode must also be greater than 3.8mA in order for the signal to be considered valid. |
| PBx | Probe value 1-wire x. |
| PBx.VLD | Probe validity flag 1-wire x. It will return 1 if the reading is successful. |
| ALM_CCx.PROP | Check and set alarms. Where CCx is the abbreviation of the corresponding channel and PROP is the property to be modified. |

## 11.2 CHECK AND CONFIGURE GEOGRAPHICAL COORDINATES

| Identifier | Description |
|---|---|
| POS.LAT | Latitude in decimal degrees. |
| POS.LON | Longitude in decimal degrees. |

## 11.3 CHECK AND SET ALARMS (ALM_CCX.PROP)

| Identifier | Description |
|---|---|
| ALM_DIx | Digital input x alarm. |
| ALM_AIx | Analogue input x alarm. |
| ALM_FRx | Flowmeter x alarm. |
| ALM_MBx | MODBUS channel x alarm. |
| ALM_PBx | MICROCOM probe alarm (STDV01 and STDV02) x. |
| ALM_EXPx | Expansion channel x alarm. |
| ALM_MCx | Math channel x alarm. |
| ALM_Fx | Flag x alarm. |

Check and modify channel alarms with digital values.

| Identifier | Property | Description | Type |
|---|---|---|---|
| ALM_DIx<br>ALM_MBx<br>ALM_EXPx<br>ALM_Fx | PH | Reading and writing. Persistence.<br>Time in seconds that the digital signal must be in the active state for the alarm to go off.<br>Maximum persistence: 65535 seconds. | INT |
| | PL | Reading and writing. Reset | INT |

| Identifier | Property | Description | Type |
|---|---|---|---|
| | | Time in seconds for the alarm to reset (become active again) after going off.<br>Maximum reset time: 65535 seconds. | |

Example of modifying an alarm configuration of a channel with digital values.

```
L1 ALM_DI2.PH = 10
L2 ALM_DI2.PL = 15
```

| Line | Description |
|---|---|
| L1 | Sets the persistence of the digital input alarm 2 configuration 0 to 10 seconds. |
| L2 | Sets the resetting of the digital input alarm 2 configuration 0 to 15 seconds. |

Checks and modifies channel alarms with analogue values.

| Identifier | Property | Description | Type |
|---|---|---|---|
| ALM_AIx<br>ALM_FRx<br>ALM_MBx<br>ALM_PBx<br>ALM_EXPx<br>ALM_MCx | MAX | Reading and writing. Maximum threshold.<br>The value above which the alarm goes off. | REAL |
| | MIN | Reading and writing. Minimum threshold.<br>The value below which the alarm goes off. | REAL |
| | HYS | Reading and writing. Hysteresis or "Dead Band".<br>Difference between alarm and reset threshold. | INT |
| | PER | Reading and writing. Persistence.<br>The amount of time the signal must be out of range before the alarm goes off.<br>Maximum persistence: 65535 reading cycles (Expressed in seconds, unless otherwise indicated). | INT |

Example of modifying an alarm of a channel with analogue values.

```
L1 ALM_AI3.MAX = 3.75
L2 ALM_AI3.MIN = 1.25
L3 ALM_AI3.HYS = 0.5
L4 ALM_AI3.PER = 60
L5 M0 = ALM_AI3.MAX
L6 M1 = ALM_AI3.MIN
```

| Line | Description |
|---|---|
| L1 | Sets the maximum threshold for the analogue input alarm 3 configuration 0 to 3.75. |
| L2 | Sets the minimum threshold for the analogue input alarm 3 configuration 0 to 1.25. |
| L3 | Sets the hysteresis in the analogue input alarm 3 configuration 0 to 0.5. |
| L4 | Sets the persistence for the analogue input alarm 3 configuration 0 to 60. |
| L5 | Stores in math register 0 the maximum threshold value in the analogue input alarm 3 configuration 0. |
| L6 | Stores in math register 1 the value of the minimum threshold in the analogue input alarm 3 configuration 0. |

Checks and modifies alarms in the different configurations (optional).

Each interface can have up to 4 independent alarm configurations identified as 0, 1, 2 and 3. Each of them is accessed by specifying the alarm number in the corresponding property, e.g.:"MAX1", which corresponds to configuration 1. If the alarm configuration alarm number is not specified, alarm number 0 is considered to be configured.

Example of modifying the alarm configurations 0 and 1 for analogue input 3.

```
L1 ALM_AI3.MAX = 3.75
L2 ALM_AI3.MIN = 1.25
L3 ALM_AI3.HYS = 0.5
L4 ALM_AI3.PER = 60

L5 ALM_AI3.MAX1 = 4.15
L6 ALM_AI3.MIN1 = 1.5
L7 ALM_AI3.HYS1 = 0.75
L8 ALM_AI3.PER1 = 300
```

| Line | Description |
|---|---|
| L1 | Sets the maximum threshold in the analogue input alarm 3 configuration 0 to 3.75. |
| L2 | Sets the minimum threshold in the analogue input alarm 3 configuration 0 to 1.25. |
| L3 | Sets the hysteresis in the analogue input alarm 3 configuration 0 to 0.5. |
| L4 | Sets the persistence in the analogue input alarm 3 configuration 0 to 60. |
| L5 | Sets the maximum threshold in analogue input alarm 3 configuration 1 to 4.15 |
| L6 | Sets the minimum threshold in the analogue input alarm 3 configuration 1 to 1.5. |
| L7 | Sets the hysteresis in the analogue input alarm 3 configuration 1 to 0.75. |
| L8 | Sets the persistence in the analogue input alarm 3 configuration 1 to 300. |

## 12 - TIMER LOCKS

MicroPLC-II allows up to 24 timer locks to be defined. The symbol ":" is used to assign an alias to the timer.

Aliases do not support accents or other special characters, such as the Spanish letter "Ñ".

Types of supported timer locks:

| | Description |
|---|---|
| RTO | Retentive on-delay timer |
| TON | Timer on delay |
| TOF | Timer off delay |
| TP | Pulse timer |
| TW | Weekly timer |
| CT | Cyclic timer |

You can also configure its properties in the same line where the timer lock is declared separated by commas “,”.

Example:

Declaration in different lines:

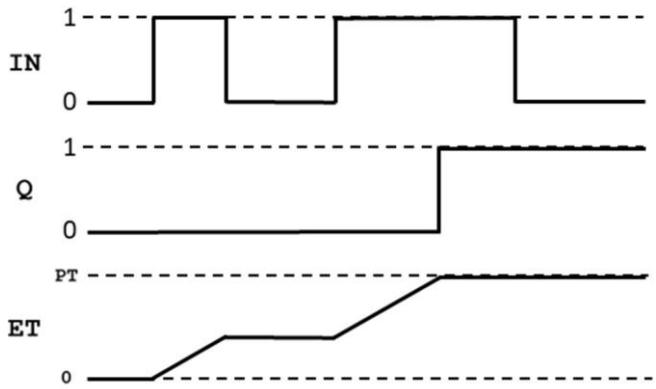
```
L1 #INIT
L2 TON : TIMER
L2 TIMER.PT = 10
L4 #END_INIT
```

Declaration in the same line:

```
L1 #INIT
L2 TON : TIMER, PT=10
L3 #END_INIT
```

## 12.1 RETENTIVE TIMER ON-DELAY (RTO)

While the input is active, an internal counter increments once per second. The internal counter does not reset when the input is deactivated and will continue counting during the next activation. The timer will activate the output when the configurable maximum time is reached.




| | Property | Description | Type |
|---|---|---|---|
| RTO | IN | Input. The ET counter increments once per second while this is 1. The ET counter stops incrementing but is not reset when a 0 is received. | BOOL |
| | PT | Input. Connection threshold. Configures the timer period. Value range: 0. 16777215. | INT |
| | R | Input. When this changes from 0 to 1, the ET value is set to 0 and the output Q is set to 0. | BOOL |
| | ET | Output. Returns the accumulated time, expressed in seconds. | INT |
| | Q | Output. It changes from 0 to 1 when ET equals PT. | BOOL |

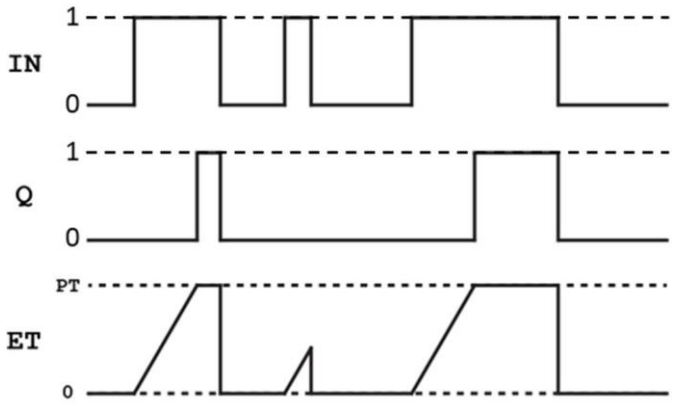
Registers the on-time of a group of lights and sends an alert when they reach 4500 hours of use, indicating that they need to be replaced.

```
L1  #INIT
L2  DI0 : LIGHTS_ON
L3  DI1 : RESET
L4  DO0 : ALARM
L5  M0 : TIME.
L6  RTO: TIMER, PT = 4500 * 3600
L7  TIMER.ET=TIME
L8  #END_INIT
L9  TIMER.IN = LIGHTS_ON
L10 TIMER.R = RESET
L11 TIME = TIMER.ET
L12 ALARM = TIMER.Q
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running |
| L2 | Assigns the alias "LIGHTS_ON" to the digital input 0. This input will be connected to the power switch for the lights. |
| L3 | Assigns the alias "RESET" to the digital input 1. |
| L4 | Assigns the alias "ALARM" to the digital output 0. |
| L5 | Assigns the alias "TIME"" to the math register 0 "M0". A math register is used instead of a variable to ensure that this value is not lost in the event of a power outage. |
| L6 | Declares "TIMER" as a retentive on-delay timer and sets the connection threshold (PT) to 16,200,000 seconds (4500 hours). |
| L7 | Initialises the accumulated time of the "TIMER.ET" timer with the value of the "TIME" variable. Related to L11 |
| L8 | Keyword to finish the initialisation section. |
| L9 | The "LIGHTS_ON" signal activates the "TIMER" IN input. |
| L10 | The "RESET" signal activates the "TIMER" R input. |
| L11 | The "TIME" variable is reset to the "TIMER" output ET. |
| L12 | The "ALARM" signa is reset to the "TIMER" output Q. |

## 12.2 TIMER ON DELAY (TON)

The output is activated following a configurable time period.




| | Property | Description | Type |
|---|---|---|---|
| TON | IN | Input. When it changes from 0 to 1, the timer starts to activate output Q after the configured PT time delay. | BOOL |

| | Property | Description | Type |
|---|---|---|---|
| | PT | Input. Connection threshold. Configures the timer period. Value range: 0. 16777215. | INT |
| | R | Input. When this changes from 0 to 1, the ET value is set to 0 and the output Q is set to 0. | BOOL |
| | ET | Output. Returns the duration in seconds that the IN input has been active. This time value is reset to 0 when the IN input changes from 1 to 0 and the R input changes from 0 to 1. | INT |
| | Q | Output. It changes from 0 to 1 when the elapsed activation time reaches the PT value, provided that the input IN is equal to 1. | BOOL |

Activation of a digital output after a 30-second persistence in relation to the digital input signal

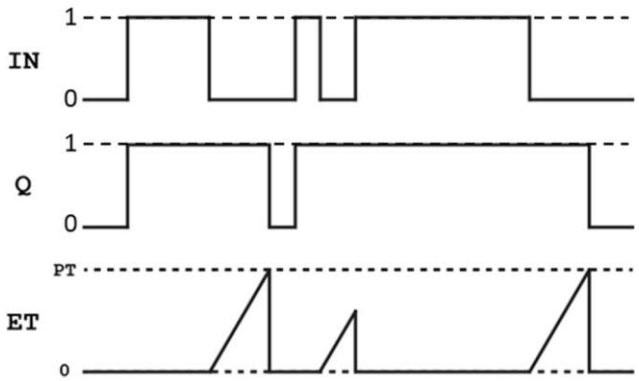
```
L1  #INIT
L2  DI0 : NETWORKFAILURE
L3  DI1 : RESET
L4  DO0 : ALARM
L5  INT : COUNTER
L6  TON : TIMER, PT = 30
L7  #END_INIT
L8  TIMER.IN = NETWORKFAILURE
L9  TIMER.R = RESET
L10 COUNTER = TIMER.ET
L11 ALARM = TIMER.Q
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running |
| L2 | Assigns the alias “NETWORKFAILURE” to the digital input 0. |
| L3 | Assigns the alias “RESET” to the digital input 1. |
| L4 | Assigns the alias “ALARM” to the digital output 0. |
| L5 | Declares “COUNTER” as an INT variable. |
| L6 | Declares “TIMER” as a on timer and sets the connection delay time (PT) to 30 seconds. |
| L7 | Keyword to finish the initialisation section. |
| L8 | The “NETWORKFAILURE” signal activates the “TIMER” input IN. |
| L9 | The “RESET” signal activates the “TIMER” R input. |

| Line | Description |
|---|---|
| L10 | The “COUNTER” variable is reset to the “TIMER” output ET. |
| L11 | The “ALARM” signal is reset to the “TIMER” output Q. |

## 12.3 TIMER OFF DELAY (TOF)

The output is deactivated following a configurable time period.




| | Property | Description | Type |
|---|---|---|---|
| TOFF | IN | Input. When this changes from 0 to 1, the output Q becomes equal to 1. When this changes from 1 to 0, the timer starts counting down to change the output Q from 1 to 0. The delay time is configured in the PT property. | BOOL |
| | PT | Input. Connection threshold. Configures the timer period. Value range: 0. 16777215. | INT |
| | R | Input. When this changes from 0 to 1, the ET value is set to 0 and the output Q is set to 0. | BOOL |
| | ET | Output. Returns the off-time duration. This time is expressed in seconds and indicates how long the input IN has been set to 0 after being set to 1. This counter resets to 0 when the PT condition is met or when the input R changes from 0 to 1. | INT |
| | Q | Output. It changes from 0 to 1 when a change from 0 to 1 is detected in the IN input. It changes from 1 to 0 when the off time reaches the PT value. | BOOL |

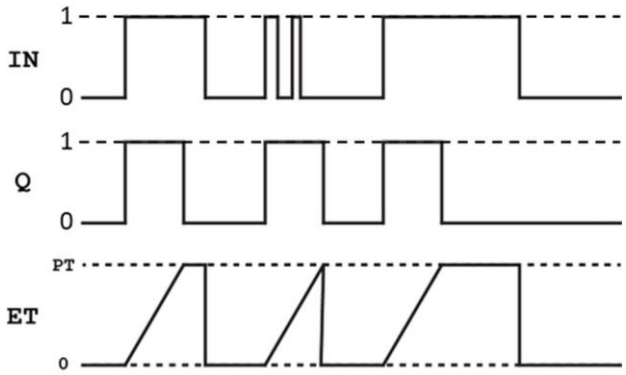
Deactivates a digital output 45 seconds after a digital input is deactivated.

```
L1  #INIT
L2  DI0 : START
L3  DI1 : RESET
L4  DO0 : MOTOR
L5  INT : COUNTER
L6  TOF : TIMER, PT = 45
L7  #END_INIT
L8  TIMER.IN = START
L9  TIMER.R = RESET
L10 COUNTER = TIMER.ET
L11 MOTOR = TIMER.Q
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running |
| L2 | Assigns the alias “START” to the digital input 0. |
| L3 | Assigns the alias “RESET” to the digital input 1. |
| L4 | Assigns the alias “MOTOR” to the digital output 0. |
| L5 | Declares “COUNTER” as an INT variable. |
| L6 | Declares “TIMER” as a off timer and sets the off-delay timer (PT) to 45 seconds. |
| L7 | Keyword to finish the initialisation section. |
| L8 | The “START” signal activates the “TIMER” input IN. |
| L9 | The “RESET” signal activates the “TIMER” R input. |
| L10 | The “COUNTER” variable is reset to the “TIMER” output ET. |
| L11 | The “MOTOR” signal is reset to the “TIMER” output Q. |

## 12.4 PULSE TIMER (TP)

The pulse timer generates a pulse at the output when the input changes from 0 to 1. The pulse duration is configurable.




| | Property | Description | Type |
|---|---|---|---|
| TP | IN | Input. Whenever it changes from 0 to 1, the output Q is set to 1 for the duration specified in the PT and the output Q is deactivated after this programmed time has elapsed.<br>The output Q will remain at 1 for the programmed time, even if the input IN parameter changes to 0. | BOOL |
| | PT | Input. Pulse width Value range: 0... 16777215. | INT |
| | R | Input. When this changes from 0 to 1, the ET value is set to 0 and the output Q is set to 0. | BOOL |
| | ET | Output. Returns the time in seconds that the output Q has been active. It stops counting once it reaches the PT value. This time is reset to 0 by setting the IN input to 0 after reaching the PT value, and by setting the R input to 1. | INT |
| | Q | Output. It changes from 0 to 1 when a 1 is received at the IN input. After the Q output has remained on for the duration of the PT, it will automatically switch to 0, regardless of the value at the IN input. | BOOL |

Pressing a button generates a 2-second output pulse.

```
L1 #INIT
L2 DI0 : BUTTON
L3 DO0 : PULSE_2S
L4 TP : TIMER, PT = 2
L5 #END_INIT
L6 TIMER.IN = BUTTON
L7 PULSE_2S = TIMER.Q
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running |
| L2 | Assigns the alias "BUTTON" to the digital input 0. |
| L3 | Assigns the alias "PULSE_2S" to the digital output 0. |
| L4 | Declares "TIMER" as a pulse timer and sets the pulse time (PT) to 2 seconds. |
| L5 | Keyword to finish the initialisation section. |
| L6 | The timer activation is controlled with the "BUTTON". |
| L7 | The "PULSE_2S" signal is reset to the "TIMER" output Q. |

## 12.5 WEEKLY TIMER (TW)

The output of this timer is controlled by programming the days of the week and the activation and deactivation times.

| | Property | Description | Type |
|---|---|---|---|
| TW | WEEK | Input. Days of the week selector. Please indicate each day of the week separately, using the following numbers: Monday=1, Tuesday=2, Wednesday=3, Thursday=4, Friday=5, Saturday=6, Sunday=7. | STRING |
| | ON | Input. Activation time in HHMM format, where HH represents the hour and MM the minutes. | STRING |
| | OFF | Input. Deactivation time in HHMM format, where HH represents the hour and MM the minutes. | STRING |
| | Q | Output. It changes from 0 to 1 whenever the activation conditions are met: day of the week + activation time. It changes from 1 to 0 whenever the deactivation conditions are met: day of the week + deactivation time. | BOOL |

Activation of a digital output according to the weekly programming.

```
L1 #INIT
L2 DO0 : PUMP_START
L3 TW : TIMER, WEEK = 12345, ON = 1230, OFF = 1845
L4 #END_INIT
L5 PUMP_START = TIMER.Q
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running |
| L2 | Assigns the alias “PUMP_START” to the digital output 0. This channel controls the start-up of pump 0. |
| L3 | Defines “TIMER” as a weekly timer and configures it to be enabled from Monday to Friday, from 12.30 p.m. to 6.45 p.m. |
| L4 | Keyword to finish the initialisation section. |
| L5 | The “PUMP_START” signal is reset to the “TIMER” output Q. |

## 12.6 CYCLIC TIMERS (CT)

The output is activated whenever the time associated with each timer has elapsed.

| | Description |
|---|---|
| CT.PPM | Activates every time a new minute starts. |
| CT.PPH | Activates every time a new hour starts. |
| CT.PPD | Activates every time a new day starts. |
| CT.PPW | Activates every time a new week starts. |
| CT.PPMO | Activates every time a new month starts. |

Calculate the daily volume of a counter

```
L1 #INIT
L2 CNT0.M3 : TOTALIZER
L3 M0 : PARTIAL_TOTALIZER
L4 REAL : DAILY_VOLUME
L5 #END_INIT
L6 DAILY_VOLUME = TOTALIZER – PARTIAL TOTALIZER
L7 IF CT.PPD ; PARTIAL_TOTALIZER = TOTALIZER ; CT.PPD=0
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Assigns the alias “TOTALIZER” to the digital input 0 counter in cubic meters. |
| L3 | Assigns the alias “PARTIAL_TOTALIZER” to the math register 0. |
| L4 | Declares “DAILY_VOLUME” as a REAL variable. |
| L5 | Keyword to finish the initialisation section. |
| L6 | Calculates the daily volume by subtracting the totalizer value at the beginning of the day from the current totalizer value. |
| L7 | Whenever it detects a change of day (CT.PPD=1), a new value is stored in the partial totalizer and CT.PPD is reset to zero to prepare it for the next day. |

# 13 - SMS SENDING AND RECEIPT MODULE

MicroPLC-II has natively implemented the functionality for exchanging information between stations via SMS, intended as a redundant communication method for the PUBLISH/SUBSCRIBE functionality.

This functionality allows the exchange of integer variables between stations, segmented into 4 logical channels.

## 13.1 SENDING OF SMS MESSAGES

The SMSTX command triggers the sending of an SMS message to another Microcom station, using the phone number to be called and a variable number of parameters containing the data to be sent as arguments.

### SYNTAX

```
SMSTX
phone_number,channel_0,channel_1,channel_2,channel_3
```

Where:
- Phone number: SMS for number to be called.
- Channel_0: 32-bit integer value that will be sent via logical channel 0.
- Channel_1: 32-bit integer value that will be sent via logical channel 1.
- Channel_2: 32-bit integer value that will be sent via logical channel 2.
- Channel_3: 32-bit integer value that will be sent via logical channel 3.

The inclusion of each channel is optional.

Examples:

Sending value 1 on channel 0 to the number +34637885326:

```
SMSTX +34637885326,1
```

Sending value 1 on channel 0 and value 100 to channel 1 to the number +34637885326:

```
SMSTX +34637885326,1,100
```

Sending value 200 on channel 3 to the number +34637885326:

```
SMSTX +34637885326,,,,200
```

Sending the MODE variable value on channel 2 to the number +34637885326:

```
SMSTX +34637885326,,MODE
```

MODE is an INT type variable that has been declared previously.

## 13.2 RECEIPT OF SMS

The variable SMSRX allows the information received by SMS to be read.

| | Property | Description | Type |
|---|---|---|---|
| SMSRX | CHx | Reading/writing. Value received in channel x. The initial value when resetting the system is 0. | INT |

Example:

Activate digital output 1 when value 3 is received on channel 0:

```
IF SMSRX.CH0 = 3 ; DO1 = 1; SMSRX.CH0 = 0
```

Channel 0 is reset to 0 after digital output 1 is activated. This method allows for a new value being received to be easily detected.

## 13.3 AUTHORISED PHONE NUMBER DECLARATION

The SMS reception module for inter-station communication only accepts SMS messages from a specific list of authorised phone numbers that is separate from the general list used for sending and receiving commands via SMS (main MicroConf screen). The SMSAUT command allows up to four authorised phone numbers to be declared.

SYNTAX:

```
SMSAUT list of phone numbers
```

Where:

- List of phone numbers: Between one and four phone numbers, separated by commas.

Phone numbers must be in international format, except for short internal office numbers.

Example:

```
SMSAUT +34637885326, 3005
```

# 14 - MODBUS COMMUNICATION MODULE

MicroPLC-II natively includes a module for data transmission via MODBUS-RTU or MODBUS-TCP. Up to 48 instances of this module, identified as MBOUT, can be defined.

<table>
<tr><th></th><th>Property</th><th>Description</th><th>Type</th></tr>
<tr><td rowspan="5">MBOUT</td><td>SL</td><td>Address of the MODBUS slave device you wish to write on.<br>This involves operations in MODBUS-RTU mode.</td><td>INT</td></tr>
<tr><td>SRV</td><td>Identifier of the MODBUS-TCP server you wish to write on.<br>  <br>This is obtained using the IMPORT MBSRV command.<br>This involves operations in MODBUS-TCP mode.</td><td>STRING</td></tr>
<tr><td>RA</td><td>Register address to be written, using standard MODBUS notation:<br>4xxxx: Entry in the holding register.<br>0xxxx: Write in coil.</td><td>INT</td></tr>
<tr><td>MODE</td><td>Type of variable to be written:<br><table><tr><th>Mode</th><th>Description</th></tr><tr><td>SET.MBI16</td><td>16-bit integer.</td></tr><tr><td>SET.MBI32</td><td>32-bit integer. Big endian.</td></tr><tr><td>SET.MBI32LE</td><td>32-bit integer. Little endian.</td></tr><tr><td>SET.MBFLT</td><td>32-bit float. Big endian.</td></tr><tr><td>SET.MBFLTLE</td><td>32-bit float. Little endian.</td></tr><tr><td>SET.MBCOIL</td><td>Boolean.</td></tr></table></td><td>STRING</td></tr>
<tr><td>VLD</td><td>Validity flag. True if the value was transmitted successfully.</td><td>BOOL</td></tr>
</table>

## 14.1 MODBUS-TCP SERVER IDENTIFIER IMPORT

The IMPORT MBSRV command is used to import MODBUS-TCP server identifiers.

SYNTAX:

```
IMPORT MBSRV : server
```

Where:

- Server: Name of the server to be imported, as specified in the MODBUS section of MicroConf.

e.g.:

```
IMPORT MBSRV : PLC1
```

Sending of the value 124 to the slave 1 address 40001. The value is sent as a 32-bit integer (big endian). Mode MODBUS-RTU.

```
L1 #INIT
L2 MBOUT : D_LEVEL, SL = 1, RA = 40001, MODE = SET.MBI32
L3 #END_INIT

L4  D_LEVEL=124
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Assigns the alias "D_LEVEL" to a MODBUS write variable configured to write a 32-bit big endian value in the slave 1, address 40001. |
| L3 | Keyword to finish the initialisation section. |
| L4 | Assigns the value 124 to the MODBUS write channel. |

Sends the value of analogue input 0 to the MODBUS-TCP server named PLC1. The value is sent as a floating-point number (big endian). Mode MODBUS-TCP.

```
L1 #INIT
L2 IMPORT MBSRV : PLC1
L3 MBOUT : D_LEVEL, SRV = PLC1, RA = 40001, MODE = SET.MBFLT
L4 #END_INIT

L5  D_LEVEL=AI0
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Imports the MODBUS-TCP server identifier named “PLC1”. |
| L3 | Assigns the alias “D_LEVEL” to a MODBUS write variable configured to write a floating-point value to the PLC1 server, address 40001. |
| L4 | Keyword to finish the initialisation section. |
| L5 | Assigns the value in the analogue input 0 to the MODBUS output channel. |

# 15 - PUMP CONTROL MODULE (PUMP)

MicroPLC-II includes a logic module for controlling up to 6 pumps. This module manages pump rotation, operating mode, running hours counter, start-up counter, pump disable inputs, running status confirmation inputs, and other features necessary for controlling a pumping system.

The PUMP module offers two operating modes to accommodate different installation configurations, the first of which is simple rotation mode that distributes the workload evenly among all pumps, while the second, in main/support mode, allows certain pumps to be designated as "main"· and others as "support", establishing service priorities between them. This flexibility allows the solution to be able to meet virtually any pumping automation requirement.

However, the module always starts in simple priority mode in order to maintain compatibility with previous firmware versions, meaning that all pumps operate as auxiliary pumps on a priority 3 scale, and automatically switches to the main/auxiliary mode as soon as at least one pump is configured as main.

The properties of the pump control module can be configured in the same line where the module itself is declared.

Example:

Declaration in different lines:

```
L1 #INIT
L2 PUMP.NUM = 2
L3 PUMP.REQ = 2
L4 PUMP.DON = 5
L5 #END_INIT
```

Declaration in the same line:

```
L1 #INIT
L2 PUMP.NUM = 2, REQ = 2, DON =5
L3 #END_INIT
```

| | Property. | Description | Type |
|---|---|---|---|
| PUMP | NUM | Input. Configuration of the total number of pumps in the system Value range: 2... 6. | INT |
| | REQ | Input. Indicates the number of pumps that you wish to be running currently. | INT |
| | AUTO | Input. Enables automatic mode. The pumps specified by the REQ requirement will be started, taking into account the rotation schedule. | BOOL |
| | HAND | Input. Enables manual mode. Allows the pumps to be manually activated using the MANx property. | BOOL |
| | OFF | Input. Activates the system's "stop" mode. All pumps remain shut down. | BOOL |
| | MANx | Input. The specified output is manually activated when PUMP.HAND is set to 1put | BOOL |

| Property. | Description | Type |
|---|---|---|
| LEADx | Input. Configures pump X as a priority or back-up pump.<br>Value range: 1 → Priority<br>0 → Support | BOOL |
| PRIOx | Input. Configures pump x priority.<br>Value range: 0 → Disabled<br>1 → Maximum priority<br>2 → Medium priority<br>3 → Low priority | INT |
| DON | Input. Startup delay time, expressed in seconds. No other operation (start-up or shut-down) will be performed during the specified time period following a pump start-up operation.<br>Value range: 0... 255. | INT |
| DOFF | Input. Shut-down delay time, expressed in seconds. No other operation (start-up or shut-down) will be performed during the specified time period following a pump shut-down operation.<br>Value range: 0... 255. | INT |
| PRTO | Input. Maximum time allowed to receive pump operation confirmation, expressed in seconds. Setting to 0 disables this function.<br>If the operation confirmation signal is not received within the configured time period, the pump is shut down, the corresponding PRFx signal is activated, and the next pump in the sequence is started up. | INT |
| PRCx | Input. Pump x start-up confirmation. | BOOL |
| PRFx | Output. Start-up failure in pump X.<br>Indicates a failure condition for pump X when the start-up confirmation signal is not received in time.<br>Once the start-up failure output is activated, the pump will remain disabled until the fault is reset using the Rx function. | BOOL |
| Rx | Input. Resets pump x start-up failure. | BOOL |
| DISx | Input. Disables pump x. E.g. Due to being out of service. | INT |
| Qx | Output. Pump x start-up command. | BOOL |
| ACx | Output. Returns the accumulated on-time of the pump connected to output Qx. Time expressed in seconds. | INT |
| STx | Output. Returns the number of start-ups of the pump connected to digital output x. | INT |

The pump control module memories are volatile. The time and start-up accumulators should be saved in math registers (non-volatile memories). See the following example.

Filling a tank using alternating pumps and with a running time counter.

```
L1  #INIT
L2  EXP20 : B0ON
L3  EXP21 : B1ON
L4  DI0 : P0DIS
L5  DI1 : P1DIS
L6  DI2 : START_B0
L7  DI3 : START_B1
L8  AI0 : LEVEL
L9  M8 : P0AC
L10 M9 : P1AC
L11 F0 : RF_B0
L12 F1 : RF_B1
L13 BOOL : BON
L14 REM TWO PUMPS. CYCLIC OPERATION
L15 PUMP.NUM = 2, DON = 5, DOFF = 5, PRTO=10
L16 PUMP.AC0 = P0AC, AC1 = P1AC
L17 #END_INIT
L18 IF LEVEL < 0.5 ; BON = TRUE
L19 IF LEVEL > 3.2 ; BON = FALSE
L20 PUMP.REQ=BON
L21 REM RESET START-UP FAILURE
L22 IF RF_B0; PUMP.R0=RF_B0; RF_B0=0
L23 IF RF_B1; PUMP.R1=RF_B1; RF_B1=0
L24 REM E/S MODULE PUMP
L25 B0ON = PUMP.Q0
L26 B1ON = PUMP.Q1
L27 PUMP.PRC0=START_B0
L28 PUMP.PRC1=START_B1
L29 PUMP.DIS0 = P0DIS
L30 PUMP.DIS1 = P1DIS
L31 P0AC = PUMP.AC0
L32 P1AC = PUMP.AC1
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Assigns the alias “B0ON” to expansion 20, “EXP20”. This channel controls the start-up of pump 0. |
| L3 | Assigns the alias “B1ON” to expansion 21, “EXP21”. This channel controls the start-up of pump 1. |
| L4 | Assigns the alias “P0DIS” to the digital input 0, “DI0”. This channel disables the use of pump 0. |
| L5 | Assigns the alias “P1DIS” to the digital input 1, “DI1”. This channel disables the use of pump 1. |
| L6 | Assigns the alias “START_B0” to the digital input 2, “DI2”. This channel indicates the start confirmation for pump 0. |
| L7 | Assigns the alias “START_B1” to the digital input 3, “DI3”. This channel indicates the start confirmation for pump 1. |
| L8 | Assigns the alias “LEVEL” to the analogue input 0 “AI0”. This channel indicates the tank water level in meters. |
| L9 | Assigns the alias “P0AC” to the math register 8,”M8”. This non-volatile memory will store the accumulated on-time of pump 0. |
| L10 | Assigns the alias “P1AC” to the math register 9,”M9”. This non-volatile memory will store the accumulated on-time of pump 1. |
| L11 | Assigns the alias “RF_B0” to flag 0, “F0”. This channel indicates the restart of the startup sequence for pump 0. |
| L12 | Assigns the alias “RF_B1” to flag 1, “F1”. This channel indicates the restart of the startup sequence for pump 1. |
| L13 | Declares “BON” as a BOOL variable. |
| L14 | Comment to describe the program. This line does not run. |
| L15 | Configures the pump control module. Number of pumps to be controlled = 2, maximum continuous operating time = 30 minutes, delay for switching off and on = 5 seconds. |
| L16 | Initialises the pump control module. Sets “PUMP.AC0”, the accumulated on-time for pump 0, to the “P0AC” value. Sets “PUMP.AC1”, the accumulated on-time for pump 1, to the “P1AC” value. The timer lock module's memory is volatile and will be reset if the device loses power. This line of code prevents data loss by loading the values stored in the non-volatile memory registers. Related to code lines L22 and L23. |
| L17 | Keyword to finish the initialisation section. |
| L18 | The variable “BON” is activated if “LEVEL” is less than 0.5 meters, indicating that water is required. |
| L19 | The variable “BON” is deactivated if “LEVEL” is greater than 3.2 meters, indicating that no water is needed. |
| L20 | “PUMP.REQ” indicates which pumps must be running at any given time. Setting it to “BON” will activate the pump when the water level is below 0.5 meters and turn it off when the level is above 3.2 meters. |

| Line | Description |
|---|---|
| L21 | Comment to describe the program. This line does not run. |
| L22 | If "RF_B0" is equal to 1, this value is written to the property that resets the pump 0 startup fault "PUMP.R0", and the value of "RF_B0" is set to zero.<br>This feature allows the user to reset the pump via the panel in Zeus. |
| L23 | If "RF_B1" is equal to 1, this value is written to the property that resets the pump 0 startup fault "PUMP.R1", and the value of "RF_B1" is set to zero.<br>This feature allows the user to reset the pump via the panel in Zeus. |
| L24 | Comment to describe the program. This line does not run. |
| L25 | Sets the status of the start-up command for pump 0 "PUMP.Q0" with the digital output that controls pump 0 "B0ON". |
| L26 | Sets the status of the start-up command for pump 1 "PUMP.Q1" with the digital output that controls pump 1 "B1ON". |
| L27 | Sets the start confirmation indicator of pump 0 "START_B0" with the corresponding control module input "PUMP.PRC0". |
| L28 | Sets the start confirmation indicator fo the pump 1 "START_B1" with the corresponding control module input "PUMP.PRC1". |
| L29 | Sets the disable indicator status of pump 0 "P0DIS" with the corresponding control module input "PUMP.DIS0". |
| L30 | Sets the disable indicator status of pump 1 "P1DIS" with the corresponding control module input "PUMP.DIS1". |
| L31 | Stores the on-time of pump 0, "PUMP.AC0", in "P0AC". This ensures that this data is stored permanently and not lost in the event of a power outage. Related to code line L12. |
| L32 | Stores the on-time of pump 1, "PUMP.AC1", in "P1AC". This ensures that this data is stored permanently and not lost in the event of a power outage. Related to code line L12. |

# 16 - PID MODULE

The MicroPLC-II PID module is designed to offer precise control in continuous process control applications. Its architecture supports the execution of multiple independent control loops, each with a refresh time of 1 second.

The proportional gain (Kp), integral time (Ti), and derivative time (Td) constants can be adjusted individually for each control loop, and the output limits and operating ranges can be defined, allowing the control system to be adapted to the specific features of pumps, valves, or any other type of actuator.

The system also incorporates advanced loop protection and optimisation features:

- Anti-windup in the integral part to prevent the control signal becoming saturated when the output reaches its limits.
- Setpoint smoothing, that allows setpoint filtering, preventing abrupt changes that could destabilise the process.
- Tracking, that allows the controller to follow a reference value or trajectory without causing abrupt transients when switching between control modes (e.g., from manual to automatic) or when switching between different PID loops.
- Automatic/manual mode selection, allowing for temporary manual interventions without compromising the control parameters.

Finally, the module continuously records process values, setpoint values, and control output, which facilitates start-up, performance diagnostics, and parameter optimisation. This versatility allows the MicroPLC-II to be integrated into a wide variety of pumping systems and industrial processes that require a robust and fast-responding PID loop.

Its properties can be configured in the same line where the PID module is declared.

Example:

Declaration in different lines:

```
L1 #INIT
L2 PID : REG
L3 REG.KP = 1
L4 REG.TI = 400
L5 REG.TD = 100
L6 RAMP = 0.1
L7 #END_INIT
```

Declaration in the same line:

```
L1 #INIT
L2 PID : REG, KP =1, TI =400, TD = 100, RAMP = 0.1
L3 #END_INIT
```

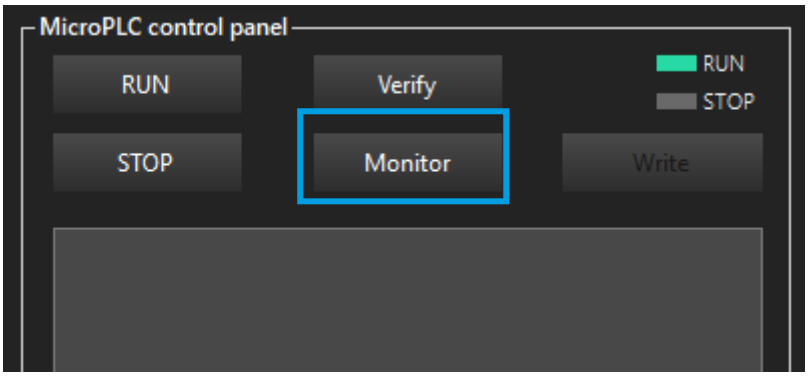
| | Property. | Description | Predefined values | Type |
|---|---|---|---|---|
| PID | KP | Input. Proportional gain. | 0 | REAL |
| | TI | Input. Integral time. 0 Disables. The comprehensive approach corrects errors that have accumulated over time. | 0 | REAL |
| | TD | Input. Derivative time. The derivative action anticipates the future evolution of the error. | 0 | REAL |
| | MIN | Input. Allows the minimum output of the PID to be configured. | 0 | REAL |
| | MAX | Input. Allows the maximum setting output of the PID to be configured. | 100 | REAL |
| | DA | Input.<br>Direct action=true → A larger action increases the value of the process variable.<br>Direct action= false → A larger action decreases the value of the process variable. | True | BOOL |
| | MAN | Input. Activates/Deactivates manual mode. Manual mode causes the PID output to be set to the value configured in the MO property. | False | BOOL |
| | MO | Input. This allows the PID output value to be configured when manual mode is activated. | 0 | REAL |
| | TRK | Input. Activates/Deactivates tracking mode. The tracking mode allows for external feedback to be used for the PID. | False | BOOL |
| | TV | Input. Track value. Feedback value for Tracking mode | 0 | REAL |
| | DFF | Input. Derivative filter factor.<br>This attenuates high-frequency disturbances in the error signal by applying an **infinite impulse response** (IIR) **filter** to the derivative term. | 0 | REAL |
| | RAMP | Input. This allows the change rate of the set point for each 1-second cycle to be configured. | 0 | REAL |
| | SP | Input. Set point. Regulator target value. | 0 | REAL |
| | PV | Input. Process variable. Actual input value. | 0 | REAL |
| | OUT | Output. Control variable. PID output value. | 0 | REAL |

## 16.1 PID MODULE SETTINGS

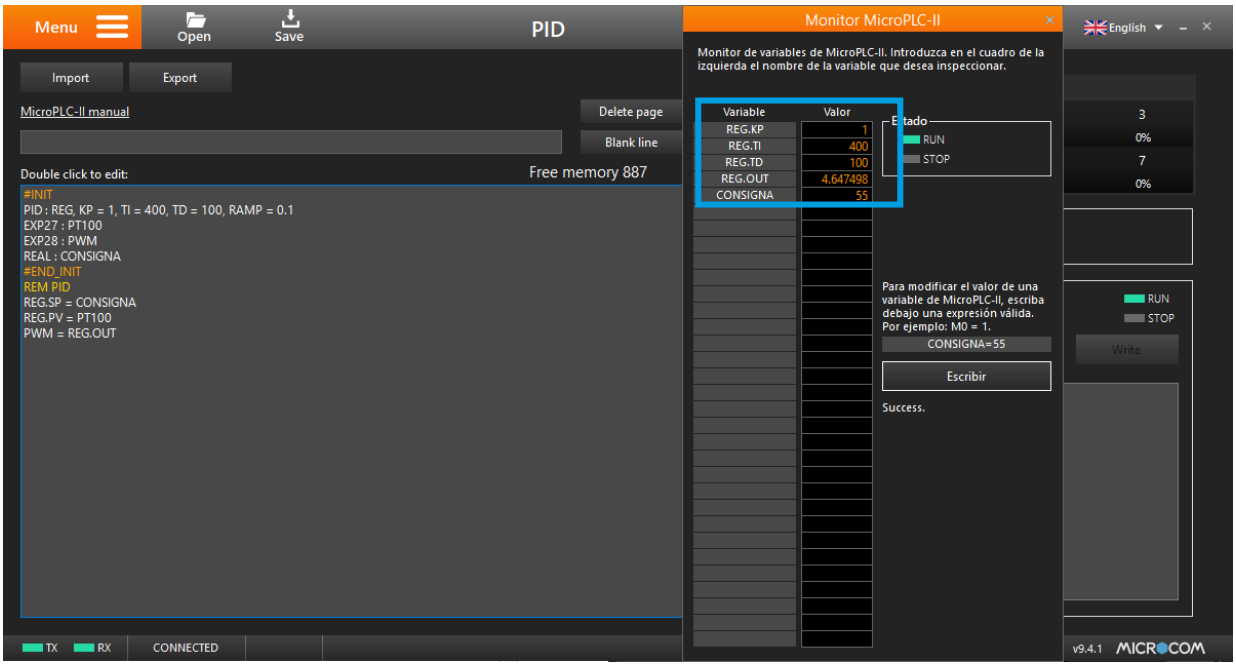
The MicroPLC-II monitor should be used to that the PID module is properly configured. This tool allows you to view the process values, the setpoint, and the control output generated by the PID module in real time It also allows the individual controller parameters to be viewed and accurately configured, including the proportional gain (Kp), integral time (Ti), and derivative time (Td). This monitoring and adjustment capability allows the user to easily optimise the PID loop performance according to the specific requirements of each application.

For proper setup, please follow these steps:

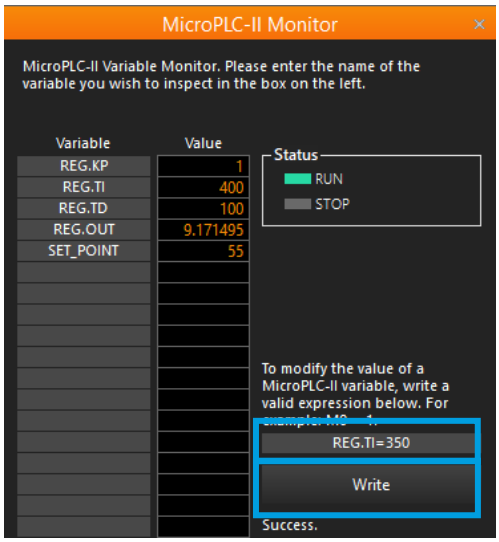
1. Open the monitor from the MicroPLC-II control panel.



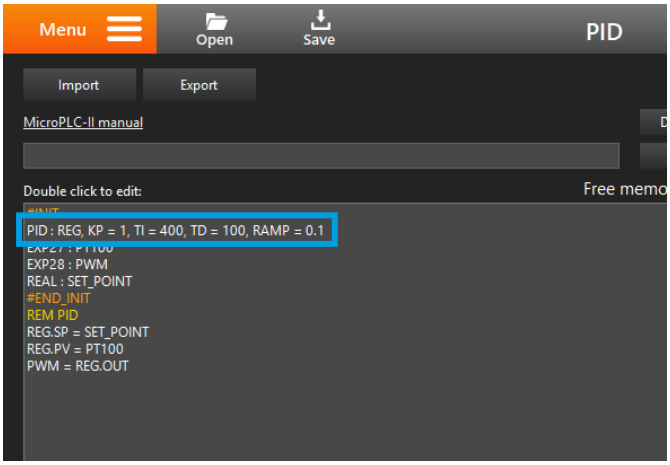

2. **Configure the PID parameters of interest for the proper configuration of the system.** These typically include the proportional gain, integration time, derivative time, and output.



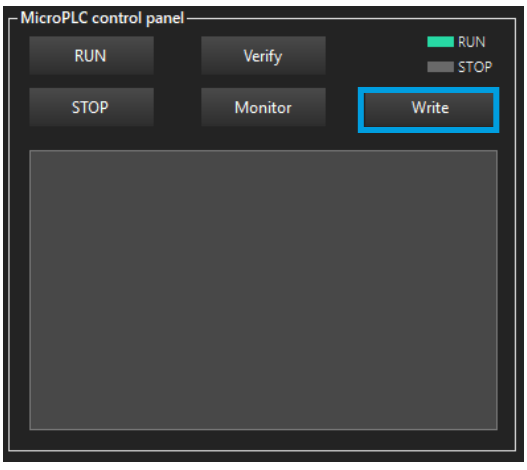
3 Adjust the PID settings until you achieve the desired output response.




4 Once the PID parameters have been correctly adjusted, enter the corresponding values in the lines where the PID is declared in order for the system to start up with the new predefined values.




5 Write the new Micro PLC-II script from the control panel to load the final values for the PID module.



Temperature control in an oven

```
L1  #INIT
L2  PID : REG, KP =1, TI = 400, TD = 100, RAMP = 0.1
L3  EXP27 : PT100
L4  EXP28 : PWM
L5  REAL : SETPOINT
L6  #END_INIT

L7  REM PID
L8  REG.SP = SETPOINT
L9  REG.PV = PT100
L10 PWM = REG.OUT
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Declares a PID with the alias "REG", a proportional gain of 1, an integral time of 400, a derivative time of 100, and a ramp rate of 0.1. |
| L3 | Assigns the alias "PT100" to expansion 27, "EXP27". This channel receives the temperature reading. |
| L4 | Assigns the alias "PWM" to expansion 28, "EXP28". The output signal is sent to the PWM from this analogue output. |
| L5 | Declares "SETPOINT as a REAL variable. |
| L6 | Keyword to finish the initialisation section. |
| L7 | Comment to describe the program. This line does not run. |
| L8 | Enters the value from the "SETPOINT" variable in the PID SP. This property represents the set point for the PID controller. It is the target temperature value |
| L9 | Enters the value from the "PT100" input into the PID PV. This property is the process variable. The actual temperature reading we have from the system |
| L10 | Maps the PID output to the analogue output "PWM" |

# 17 - PROGRAMMING EXAMPLES

## 17.1 THERMOSTAT

Example of automatic activation of a heating system based on ambient temperature, the setpoint or activation temperature, and a hysteresis or deadband.

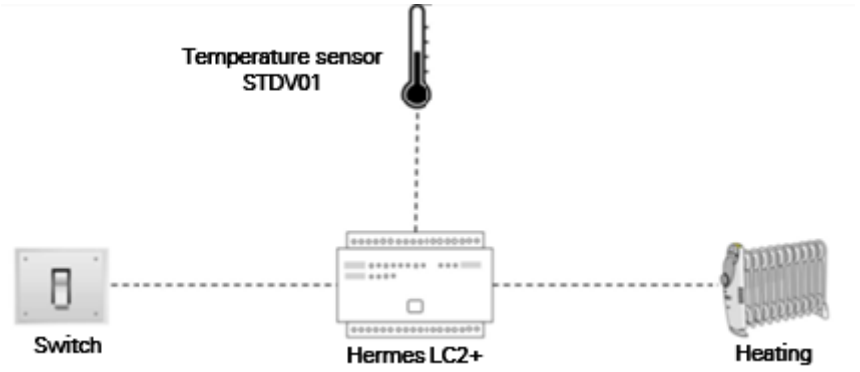
The system in this example consists of the following components:

- Microcom Hermes LC2+. This is the remote control and automation unit to which all other components are connected: the STDV01 temperature probe, the automatic power on enable switch, and the heating system. This unit runs the control logic program.

- STDV01 sensor. Ambient temperature probe compatible with the Hermes LC2+.

- Switch to manually enable or disable automatic boiler power-on. Even if the remote-control system determines that the boiler should be turned on, it will only start if this switch is activated.

- Heating system. It starts up after receiving the power supply voltage. In this example, the remote-control unit, through one of its relay outputs, will activate a contactor, which will then allow the power supply voltage to flow to the heating system.




Example of system programming using the MICROPLC-II language:

```
L1 #INIT
L2 DI0 : THERMO_ON ; REM INPUT IGNITION HEATING
L3 DO2 : HEAT_ON ; REM OUTPUT IGNITION HEATING
L4 PB0 : ROOMT ; REM ROOM TEMPERATURE
L5 REAL : SETPOINT = 21 ; REM SETPOINT TEMPERATURE
L6 REAL : HYST = 0.5 ; REM HYSTERESIS
L7 BOOL : TURN-ON; REM FLAG LOWER TEMPERATURE TO SETPOINT
L8 #END_INIT
L9  REM THERMOSTAT
L10 IF ROOMT >SETPOINT ; TURN-ON= FALSE
L11 IF ROOMT - HYST< SETPOINT ; TURN-ON= TRUE
L12 HEAT_ON = THERMO_ON AND TURN-ON
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Assigns the alias “THERMO_ON” to the digital input 0 “DI0”. The switch for manually enabling and disabling the boiler's automatic ignition is connected in this digital input. Comment: |
| L3 | Assigns the alias “HEAT_ON” to the digital output 2 “DO2”. This digital output is connected to the relay that activates the heating system. Comment: |
| L4 | Assigns the alias “ROOMT” to the probe 0 “PB0” Comment: |
| L5 | Declares “SETPOINT as a REAL variable. Initialises the variable with the value 21 (degrees Celsius). This variable sets the maximum temperature to be maintained in the room. Comment: |
| L6 | Declares “HYST” as a REAL variable. Initialises the variable with the value 0.5 (degrees Celsius). This variable is used to configure the HYSTERESIS, and in this case, determines the temperature at which the heating system activates together with the setpoint. Comment: |
| L7 | Declares “ON” as a BOOL variable. This will be used as an indicator of the heating system's start-up command. This variable is activated and deactivated according to the programming logic, and uses the room temperature value “ROOMT”, setpoint temperature “SETPOINT”, and hysteresis value “HYSTERESIS”. Comment: |
| L8 | Keyword to finish the initialisation section. |
| L9 | Comment to describe the program. This line does not run. |
| L10 | If the ambient temperature “ROOMT” is higher than the target or setpoint temperature “SETPOINT”, the command to activate the heating system is deactivated, and the “ON” status is set to 0. In this example, it will be deactivated when the ambient temperature is higher than 21 degrees Celsius. |
| L11 | If the result of the ambient temperature operation “ROOMT” is lower than the hysteresis “HYST”, is less than the target or “SETPOINT” temperature, the command to turn on the heating system is activated, “ON” is set to 1. In this example, the activation temperature condition is met when “ROOMT” is less than 20.5 degrees Celsius. |
| L12 | Two conditions must be met for the digital output that controls the boiler ignition to activate (“HEAT_ON” equals 1), the switch must be on (“ON” equals 1) and the temperature activation command outside of the setpoint must be active (“THERMO_ON” equals 1). The boiler will not activate if on or both conditions are not met. |

## 17.2 TWILIGHT OR ASTRONOMICAL SWITCH

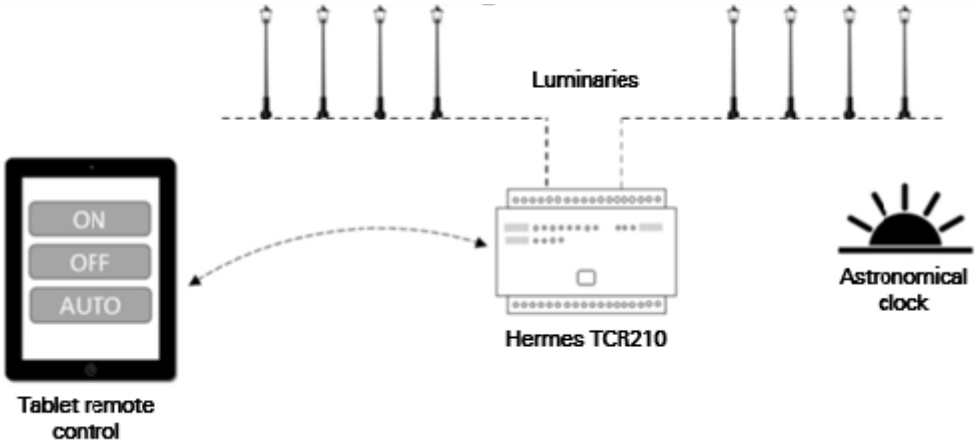
Example of the automatic and manual lighting system activation in businesses and for street lighting.

Twilight or astronomical timers are time-based controllers that operate by automatically calculating the sunrise and sunset times based on the geographical coordinates of the location where they are installed. Its typical application involves automatically turning on the lights at dusk and turning them off at dawn. The transition from winter to summer time is automatic. The system in this example consists of the following components:

- Microcom Hermes TCR210. This remote control and automation system controls the switching on and off of the lighting system. This unit runs the control logic program.

- Lighting system. This system starts operating after receiving the power supply voltage. In this example, the remote control will activate a contactor via one of its outputs, which in turn will switch on the lights.

- ZEUS platform: Web and smartphone application that allows the remote monitoring and control of MICROCOM devices.




Example of astronomical switch programming in MICROPLC-II language:

```
L1  #INIT
L2  DO3 : LIGHT_ON ; REM OUTPUT LIGHTS ON
L3  M0 : MODE ; REM 0→OFF 1→ON 2→AUTO
L4  INT : OFF = 0
L5  INT : ON = 1
L6  INT : AUTO = 2
L7  #END_INIT
L8  REM INT. TWILIGHT
L9  IF MODE = OFF ; LIGHT_ON = FALSE
L10 IF MODE = ON ; LIGHT_ON = TRUE
L11 IF MODE = AUTO ; LIGHT_ON = NOW.SOD > SUNSET OR NOW.SOD < SUNRISE
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Assigns the alias “LIGHT_ON” to the digital output 3 “DO3”. This output will be connected to the system for turning on the lights. Comment: |
| L3 | Assigns the alias “MODE” to the math register 0 “M0”. Selects the lighting operation mode. There are three modes: “OFF”, “ON” and “AUTO” |
| L4 | Declares “OFF” as an INT variable and initialises it to a value 0. This variable is used as a constant. |
| L5 | Declares “ON” as an INT variable and initialises it to a value 1. This variable is used as a constant. |
| L6 | Declares “AUTO” as an INT variable and initialises it to a value 2. This variable is used as a constant. |
| L7 | Keyword to finish the initialisation section. |
| L8 | Comment to describe the program. This line does not run. |
| L9 | If “MODE” is equal to “OFF”, then “LIGHT_ON” is set to 0. |
| L10 | If “MODE” is equal to “ON”, then “LIGHT_ON” is set to 1. |
| L11 | If “MODE” is equal to 2, then “LIGHT_ON” will be set to 1 if the current time (NOW.SOD) is greater than the sunset time (SUNSET) or less than the sunrise time (SUNRISE). This will be 0 for the remaining time. If automatic mode is enabled, the command to turn on the lights will activate at dusk and deactivate at dawn. |

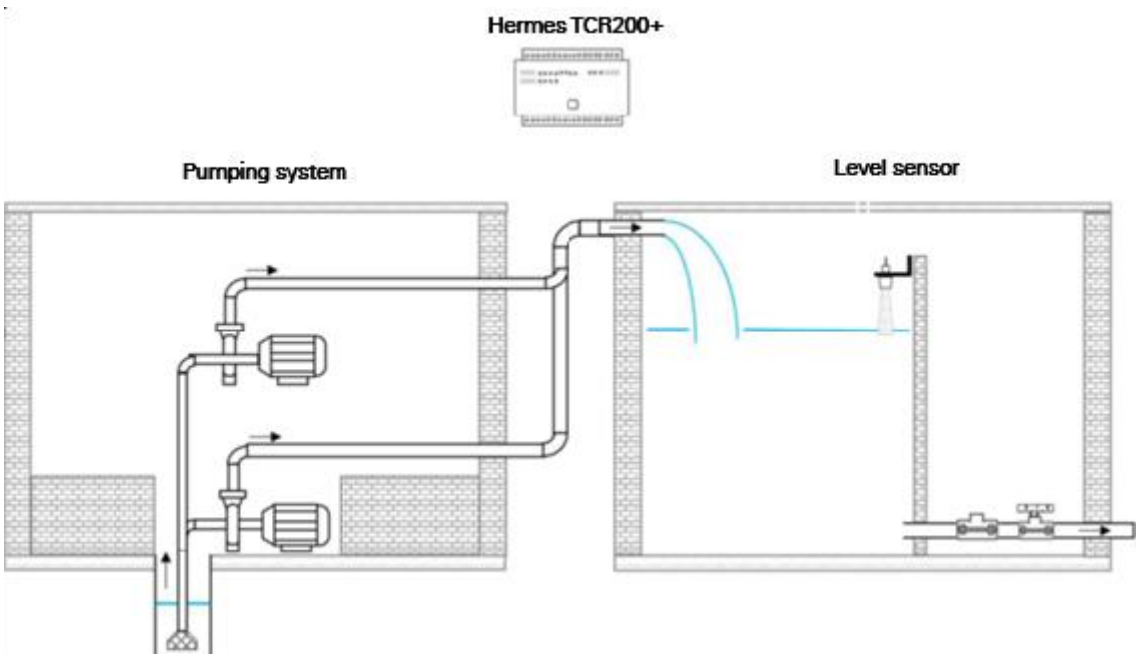
## 17.3 PUMP START-UP AUTOMATION

Example of automatic pump station operation.

This example illustrates an advanced pump management strategy that combines multiple operating conditions simultaneously. The system enables the switching between manual and automatic control, incorporates pump rotation to balance wear and tear, and dynamically adjusts the level setpoint based on electricity tariffs. During off-peak hours when energy is cheaper, the tank is filled to a minimum level of 4 meters (80% of its capacity). During mid-peak hours when an intermediate rate applies, the target is reduced to 2 meters (40%), optimising energy consumption by utilising the water that has already accumulated. During peak hours when electricity is most expensive, the level is allowed to drop to 0.5 meters (10%), maximising energy savings by avoiding pumping operations during the most expensive periods.

The system in this example consists of the following components:

- Microcom Hermes TCR200+: The remote control and automation system to which all other elements are connected: water level sensor, pump protection systems, and pump start-up commands.
- Level probe: Analogue probe for measuring the water level in the tank to be maintained with water.
- Pump system: This system consists of 2 pumps and has a separate access to its start-up control and status of each pump.




This configuration uses 2 of the 8 available pages on the Hermes devices.

Page 0:

```
L1  #INIT
L2  DO0 : START_B0
L3  DO1 : START_B1
L4  DI0 : FAILURE_B0
L5  DI1 : FAILURE_B1
L6  DI2 : CONF_B0
L7  DI3 : CONF_B1
L8  M8A : ACU_B0
L9  M8B : ACU_B1
L10 M10 : MODE; REM MODE=0-> STOP, 1-> MANUAL, 2 -> AUTO
L11 AI0 : LEVEL
L12 REAL : LOWL
L13 REAL : HIGHL
L14 BOOL : LOW_WATER
L15 INT : STOP=0
L16 INT : MANU=1
L17 INT : AUTO=2
L18 REM TWO PUMPS CYCLIC OPERATION
L19 PUMP.NUM=2, DON=5, DOFF=5, PRTO=10, AC0=ACU_B0, AC1=ACU_B1
L20 REM PEAK RATE MORNING (MON-FRI 10 a.m.- 2 p.m.)
L21 TW : PEAKM, WEEK = 12345, ON = 1000, OFF = 1400
 L22 REM PEAK RATE AFTERNOON (MON-FRI 6 p.m -10 p.m.)
L23 TW : PEAKA,WEEK = 12345, ON = 1800, OFF = 2200
 L24 REM MID.PEAK RATE MORNING (MON-FRI 8 a.m. -10 a.m.)
L25 TW : MIDPEAKM,WEEK = 12345, ON = 0800, OFF = 1000
 L26 REM MID-PEAK RATE AFTERNOON (MON-FRI 2 p.m.-6 p.m.)
L27 TW : MIDPEAKA,WEEK = 12345, ON = 1400, OFF = 1800
L28 #END_INIT
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Assigns the alias “START_B0” to the digital output 0 “DO0”. Output connected to the pump start system 0 |
| L3 | Assigns the alias “START_B1” to the digital output 1 “DO1”. Output connected to the pump start system 1 |
| L4 | Assigns the alias”FAILURE_B0” the digital input 0 “DI0”. Input connected to the pump failure Indicator 0 |
| L5 | Assigns the alias “FAILURE_B1” to the digital input 1 “DI1”. Input connected to the pump failure Indicator 1 |
| L6 | Assigns the alias “CONF_B0” to the digital input 2 “DI2”. Input connected to the start confirmation for pump 0. |

| Line | Description |
|---|---|
| L7 | Assigns the alias "CONF_B1" to the digital input 3 "DI3". Input connected to the start confirmation for pump 1. |
| L8 | Assigns the alias "ACU_B0" to the math register 8 (least significant 32 bits) "M8A". Stores the pump 0 power-on. |
| L9 | Assigns the alias "ACU_B1" to the math register 8 (most significant 32 bits) "M8B". Stores the pump 1 power-on. |
| L10 | Assigns the alias "MODE" to the math register 10 "M10". This variable indicates the system's operating mode. |
| L11 | Assigns the alias "LEVEL" to the analogue input 0 "AI0". Input connected to the tank level sensor. |
| L12 | Declares "LOWL" as a REAL variable. Activation or low level setpoint. |
| L13 | Declares "HIGHL" as a REAL variable. High level or deactivation setpoint. |
| L14 | Declares "LOW_WATER" as a BOOL variable. Indicates a request for water from the pump. |
| L15 | Declares "STOP" as an INT variable and initialises it to a value 0. This is a constant value. |
| L16 | Declares "MANU" as an INT variable and initialises it to a value 1. This is a constant value. |
| L17 | Declares "AUTO" as an INT variable and initialises it to a value 2. This is a constant value. |
| L18 | Comment to describe the program. This line does not run. |
| L19 | Configures the pump control module as follows: pumps to control 2 (PUMP.NUM=2), ignition delay 5 seconds (DON=5), stop delay 5 seconds (DOFF=5), start confirmation time 10 seconds (PRTO=10), restores the pump 0 ignition value from "ACU_B0" (AC0=ACU_B0), and restores the pump 1 ignition value from "ACU_B1" (AC1=ACU_B1). |
| L20 | Comment to describe the program. This line does not run. |
| L21 | Declares "PEAKM" as a weekly timer. Configure it so that it is enabled from Monday to Friday (WEEK=12345), activates at 10 a.m. (ON=1000), and deactivates at 2 p.m. (OFF=1400). |
| L22 | Comment to describe the program. This line does not run |
| L23 | Declares "PEAKA" as a weekly timer. Configure it so that it is enabled from Monday to Friday (WEEK=12345), activates at 6 p.m. (ON=1800), and deactivates at 10 p.m. (OFF=2200). |
| L24 | Comment to describe the program. This line does not run. |
| L25 | Declares "MIDPEAKM" as a weekly timer. Configure it so that it is enabled from Monday to Friday (WEEK=12345), activates at 8 a.m. (ON=0800), and deactivates at 10 a.m. (OFF=1000). |
| L26 | Comment to describe the program. This line does not run |
| L27 | Declares "MIDPEAKA" as a weekly timer. Configure it so that it is enabled from Monday to Friday (WEEK=12345), activates at 2 p.m. (ON=1400), and deactivates at 6 p.m. (OFF=1800). |
| L28 | Keyword to finish the initialisation section. |

Page 1

```
L1  REM PARAMETERS FILLING ACCORDING TO RATE
L2  REM OFF-PEAK
L3  LOWL=4 ; HIGHL=5
L4  REM MID-PEAK
L5  IF MIDPEAKM.Q OR MIDPEAKA.Q ; LOWL=2 ; HIGHL=3
L6  REM PEAK
L7  IF PEAKM.Q OR PEAKA.Q ; LOWL=0.5 ; HIGHL=1
L8  REM AUTOMATIC FILLING
L9  IF MODE<>AUTO AND MODE<>MANU AND MODE<>STOP ; MODE=STOP
L10 IF LEVEL<LOWL ; LOW_WATER=TRUE
L11 IF LEVEL<HIGHL ; LOW_WATER= FALSE
L12 IF MODE=AUTO ; PUMP.REQ=LOW_WATER
L13 IF MODE=MANU ; PUMP.REQ=1
L14 IF MODE=STOP ; PUMP.REQ=0
L15 REM E/S MODULE PUMP
L16 START_B0=PUMP.Q0
L17 START_B1=PUMP.Q1
L18 PUMP.DIS0=FAILURE_B0
L19 PUMP.DIS1=FAILURE_B1
L20 PUMP.PRC0=CONF_B0
L21 PUMP.PRC1=CONF_B1
L22 ACU_B0=PUMP.AC0
L23 ACU_B1=PUMP.AC1
```

| Line | Description |
|---|---|
| L1 | Comment to describe the program. This line does not run. |
| L2 | Comment to describe the program. This line does not run. |
| L3 | Sets “LOWL” to 4 and “HIGHL” to 5. The start-up and stop settings for the off-peak time period, which is when the cheapest rate applies, are configured so that the water level in the tank remains above 4 meters and below 5 meters. |
| L4 | Comment to describe the program. This line does not run. |
| L5 | If the “MIDPEAKM” or “MIDPEAKA” timers are activated, sets “LOWL” to 2 and “HIGHL” to 3. The start-up and stop settings for the mid-peak time period, which is when the intermediate rate applies, are configured so that the water level in the tank remains above 2 meters and below 3 meters. |
| L6 | Comment to describe the program. This line does not run. |
| L7 | If the “PEAKM” or “PEAKA” timers are activated, sets “LOWL” to 0.5 and “HIGHL” to 1. The start-up and stop settings for the peak time period, which is when the most expensive rate applies, are configured so that the water level in the tank remains above 0.5 meters and below 1 meter. |
| L8 | Comment to describe the program. This line does not run. |

| Line | Description |
|---|---|
| L9 | If "MODE" is different from "AUTO", "MANU" or "STOP", then "MODE" is set to "STOP·. The operating modes indicate whether the system operates automatically based on setpoint levels or manually via remote commands. This line serves as a safety control; if the MODE variable has a value other than 0, 1, or 2, it is considered a fault, and the system is forced into STOP mode. |
| L10 | If the tank level ("LEVEL") is below the minimum or start-up level setpoint ("LOW_L"), the low water level indicator is activated ("LOW_WATER" is set to 1 (TRUE)). The status of this indicator is later used as a condition for the command to start the pumps in automatic mode (see line L12). |
| L11 | If the tank level ("LEVEL") is below the maximum or start-up level setpoint ("HIGHL"), the low water level indicator is deactivated ("LOW_WATER" is set to 0 (FALSE)). The status of this indicator is later used as a condition for the command to start the pumps in automatic mode (see line L12). |
| L12 | If the operating mode ("MODE") is set to automatic ("AUTO"), the pump control module's property that determines the number of active pumps ("PUMP.REQ") will be set to the value of the "LOW_WATER" indicator, meaning the command to activate a pump will be issued if the "LOW_WATER" indicator is active, If the "LOW_WATER" indicator is inactive, the command to not activate any pumps will be issued. |
| L13 | If "MODE" is set to manual ("MANU"), "PUMP.REQ" will be set to 1. Configure the pump control module in order for 1 pump to operate. One of the pumps will remain on while in this mode. |
| L14 | If "MODE" is equal to stop ("STOP"), "PUMP.REQ" will obtain the value 0. Configures the pump control module to prevent any pumps from operating, meaning, the pumping system will remain off. |
| L15 | Comment to describe the program. This line does not run. |
| L16 | "START_B0" is reset to the value "PUMP.Q0". The start-up command for pump 0 in the pump control block controls the digital output that activates pump 0. |
| L17 | "START_B1" is reset to the value "PUMP.Q1". The start-up command for pump 1 in the pump control block controls the digital output that activates pump 1. |
| L18 | "PUMP.DIS0" is reset to the value "FAILURE_B0". This controls the disabling of pump 0. While digital input 0 is active, pump 0 will remain disabled, meaning the automation system will not attempt to start it up. |
| L19 | "PUMP.DIS1" is reset to the value "FAILURE_B1". This controls the disabling of pump 1. While digital input 1 is active, pump 1 will remain disabled, meaning the automation system will not attempt to start it up. |
| L20 | "PUMP.PRC0" is reset to the value "CONF_B0". This informs the PUMP module of the successful start-up of pump 0. |
| L21 | "PUMP.PRC1" is reset to the value "CONF_B1". This informs the PUMP module of the successful start-up of pump 1. |
| L22 | "ACU_B0" is reset to the value "PUMP.AC0". This stores the operating time or running hours in non-volatile memory. This ensures that this value is not lost even if there is a power outage. |
| L23 | "ACU_B1" is reset to the value "PUMP.AC1". This stores the operating time or running hours in non-volatile memory. This ensures that this value is not lost even if there is a power outage. |

## 17.4 PUMP CONTROL IN PRIMARY/BACKUP MODE

Example of pump control for a wastewater pumping station.

This example describes a control system for wastewater pumping stations based on a lead/lag pump operation structure. The system consists of four pumps, each with a separate access to its status and start/stop control. The configuration allows us to define which pumps will operate as primary and which as standby ones, as well as setting a priority level for the use of each pump. The water level in the tank is measured using an analogue probe, allowing the system operation to be dynamically adjusted according to demand. Unlike the previous example, this scenario does not provide for time-of-day rates, meaning that the activation criterion is based solely on the water level and the load-sharing logic between the pumps.

The system in this example consists of the following components:

- Microcom Hermes M103 The remote control and automation system to which all other elements are connected: water level sensor, pump protection systems, and pump start-up commands.
- Level probe: Analogue probe for measuring the water level in the tank to be maintained with water.
- Pump system: This system consists of 4 pumps and has a separate access to its start-up control and status of each pump.

Page 0:

```
L1  #INIT
L2  DI0 : FAILURE_B0
L3  DI1 : FAILURE_B1
L4  DI2 : FAILURE_B2
L5  DI3 : FAILURE_B3
L6  DI4 : START_B1
L7  DI5 : START_B2
L8  DI6 : START_B3
L9  DI7 : START_B4
L10 EXP16 : RELAY_B0
L11 EXP17 : RELAY_B1
L12 EXP18 : RELAY_B2
L13 EXP19 : RELAY_B3
L14 AI0 : LEVEL
L15 M0 : HIGHL
L16 M1 : LOWL
L17 M2 : SUPPORTL
L18 M3 : PRIO_B0
L19 M4 : PRIO_B1
L20 M5 : PRIO_B2
L21 M6 : PRIO_B3
L22 M10 : MODE; REM 0–>STOP, 1–>START, 2–>AUTO
L23 F0 : MAIN_B0
L24 F1 : MAIN_B1
L25 F2 : MAIN_B2
```

```
L26 F3 : MAIN_B3
L27 F4 : B0MAN
L28 F5 : B1MAN
L29 F6 : B2MAN
L30 F7 : B3MAN
L31 F8 : DIS_MAN_B0
L32 F9 : DIS_MAN_B1
L33 F10 : DIS_MAN_B2
L34 F11 : DIS_MAN_B3
L35 F12: RESET_FB0
L36 F13: RESET_FB1
L37 F14: RESET_FB2
L38 F15: RESET_FB3
L39 BOOL : START
L40 BOOL : SUPPORT
L41 INT : STOP=0
L42 INT : MANU=1
L43 INT : AUTO=2
L44 PUMP.NUM=6,DON=5,DOFF=5,PRTO=10
L45 #END_INIT
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Assigns the alias"FAILURE_B0" the digital input 0 "DI0". Input connected to the pump failure Indicator 0 |
| L3 | Assigns the alias "FAILURE_B1" to the digital input 1 "DI1". Input connected to the pump failure Indicator 1 |
| L4 | Assigns the alias "FAILURE_B2" to the digital input 2 "DI2". Input connected to the pump failure Indicator 2 |
| L5 | Assigns the alias "FAILURE_B3" to the digital input 3 "DI3". Input connected to the pump failure Indicator 3 |
| L6 | Assigns the alias "START_B0" to the digital input 4 "DI4". Input connected to the start confirmation for pump 0. |
| L7 | Assigns the alias "START_B1" to the digital input 5 "DI5". Input connected to the start confirmation for pump 1. |
| L8 | Assigns the alias "START_B2" to the digital input 6 "DI6". Input connected to the start confirmation signal for pump 2 |
| L9 | Assigns the alias "START_B3" to the digital input 7 "DI7". Input connected to the start confirmation signal for pump 3 |
| L10 | Assigns the alias "RELAY_B0" to expansion 16, "EXP16". Output connected to the pump start system 0 |

| Line | Description |
|---|---|
| L11 | Assigns the alias "RELAY_B1" to expansion 17 "EXP17". Output connected to the pump start system 1 |
| L12 | Assigns the alias "RELAY_B2" to expansion 18, "EXP18". Output connected to the pump start system 2 |
| L13 | Assigns the alias "RELAY_B3" to expansion 19, "EXP19". Output connected to the pump start system 3 |
| L14 | Assigns the alias "LEVEL" to the analogue input 0 "AI0". Input connected to the tank level sensor. |
| L15 | Assigns the alias "LOWL" to the math register 0 "M0". Activation or low level setpoint. |
| L16 | Assigns the alias "HIGHL" to the math register 1 "M1". High level or deactivation setpoint. |
| L17 | Assigns the alias "SUPPORTL" to the math register 2 "M2". Setpoint for activating/deactivating the support pump. |
| L18 | Assigns the alias "PRIO_B0" to the math register 3 "M3". Pump 0 priority. |
| L19 | Assigns the alias "PRIO_B1" to the math register 4 "M4". Pump 1 priority. |
| L20 | Assigns the alias "PRIO_B2" to the math register 5 "M5". Pump 2 priority. |
| L21 | Assigns the alias "PRIO_B3" to the math register 6 "M6". Pump 3 priority. |
| L22 | Assigns the alias "MODE" to the math register 10 "M10". This variable indicates the system's operating mode. |
| L23 | Assigns the alias "MAIN_B0" to flag 0 "F0". Configures pump 0 as a priority or support. |
| L24 | Assigns the alias "MAIN_B1" to flag 1 "F1". Configures pump 1 as a priority or support. |
| L25 | Assigns the alias "MAIN_B2" to flag 2 "F2". Configures pump 2 as a priority or support. |
| L26 | Assigns the alias "MAIN_B3" to flag 3 "F3". Configures pump 3 as a priority or support. |
| L27 | Assigns the alias "B0MAN" to flag 4 "F4". Necessary to activate pump 0 in manual mode. |
| L28 | Assigns the alias "B1MAN" to flag 5 "F5". Necessary to activate pump 1 in manual mode. |
| L29 | Assigns the alias "B2MAN" to flag 6 "F6". Necessary to activate pump 2 in manual mode. |
| L30 | Assigns the alias "B3MAN" to flag 7 "F7". Necessary to activate pump 3 in manual mode. |
| L31 | Assigns the alias "DIS_MAN_B0" to flag 8 "F8". Manually disables pump 0. |
| L32 | Assigns the alias "DIS_MAN_B1" to flag 9 "F9". Manually disables pump 1. |
| L33 | Assigns the alias "DIS_MAN_B2" to flag 10 "F10". Manually disables pump 2. |
| L34 | Assigns the alias "DIS_MAN_B3" to flag 11 "F11". Manually disables pump 3. |
| L35 | Assigns the alias "RESET_FB0" to flag 12 "F12". Resets the start-up failure for pump 0. |
| L36 | Assigns the alias "RESET_FB1" to flag 13 "F13". Resets the start-up failure for pump 1. |

| Line | Description |
|---|---|
| L37 | Assigns the alias "RESET_FB2" to flag 14 "F14". Resets the start-up failure for pump 2. |
| L38 | Assigns the alias "RESET_FB3" to flag 15 "F15". Resets the start-up failure for pump 3. |
| L39 | Declares "REQUESTWATER" as a BOOL variable. Activates whenever a request for water is received. |
| L40 | Declares "REQUESTSUPPORT" as a BOOL variable. Activates whenever a support pump request is received. |
| L41 | Declares "STOP" as an INT variable and initialises it to a value 0. This is a constant value. |
| L42 | Declares "MANU" as an INT variable and initialises it to a value 1. This is a constant value. |
| L43 | Declares "AUTO" as an INT variable and initialises it to a value 2. This is a constant value. |
| L44 | Configures the pump control module: number of pumps to control 4 (PUMP.NUM=4), ignition delay 5 seconds (DON=5), stop delay 5 seconds (DOFF=5), running confirmation time 10 seconds (PRTO=10) |
| L45 | Keyword to finish the initialisation section. |

Page 1:

```
L1  REM DEFINE MAIN/SUPPORT
L2  PUMP.LEAD0=MAIN_B0
L3  PUMP.LEAD1=MAIN_B1
L4  PUMP.LEAD2=MAIN_B2
L5  PUMP.LEAD3=MAIN_B3
L6  REM DEFINE PRIORITIES
L7  PUMP.PRIO0=PRIO_B0
L8  PUMP.PRIO1=PRIO_B1
L9  PUMP.PRIO2=PRIO_B2
L10 PUMP.PRIO3=PRIO_B3
L11 REM RESET START-UP FAILURE
L12 IF RESET_FB0; PRF0=0; RESET_FB0=0
L13 IF RESET_FB1; PRF1=0; RESET_FB1=0
L14 IF RESET_FB2; PRF2=0; RESET_FB2=0
L15 IF RESET_FB3; PRF3=0; RESET_FB3=0
L16 REM WATER REQUEST
L17 IF LEVEL>HIGHL ; START=TRUE
L18 IF LEVEL<LOWL ; START=FALSE
L19 IF LEVEL>SUPPORTL; SUPPORT=TRUE
L20 IF LEVEL<HIGHL; SUPPORT=FALSE
L21 REM OPERATION MODE
L22 IF MODE<>STOP AND MODE<>MANU AND MODE<>AUTO ; MODE=STOP
L23 PUMP.HAND=(MODE=MANU) ; PUMP.AUTO=(MODE=AUTO) ; PUMP.OFF=(MODE=STOP)
L24 REM PUMP START-UP/STOP
L25 PUMP.REQ=START + SUPPORT
L26 REM E/S PUMP MODULE
L27 RELAY_B0=PUMP.Q0
L28 RELAY_B1=PUMP.Q1
L29 RELAY_B2=PUMP.Q2
L30 RELAY_B3=PUMP.Q3
L31 PUMP.MAN0=B0MAN
L32 PUMP.MAN1=B1MAN
L33 PUMP.MAN2=B2MAN
L34 PUMP.MAN3=B3MAN
L35 PUMP.PRC0=START_B0
L36 PUMP.PRC1=START_B1
L37 PUMP.PRC2=START_B2
L38 PUMP.PRC3=START_B3
L39 PUMP.DIS0=FAILURE_B0 OR DIS_MAN_B0
L40 PUMP.DIS1=FAILURE_B1 OR DIS_MAN_B1
L41 PUMP.DIS2=FAILURE_B2 OR DIS_MAN_B2
L42 PUMP.DIS3=FAILURE_B3 OR DIS_MAN_B3
```

| Line | Description |
|---|---|
| L1 | Comment to describe the program. This line does not run. |
| L2 | Assigns the value of “MAIN_B0” to the pump control module PUMP.LEAD0 property. This property determines whether pump 0 is the primary or the support pump. |

| Line | Description |
|---|---|
| L3 | Assigns the value of "MAIN_B1" to the pump control module PUMP.LEAD1 property. This property determines whether pump 1 is the primary or the support pump. |
| L4 | Assigns the value of "MAIN_B2" to the pump control module PUMP.LEAD2 property. This property determines whether pump 2 is the primary or the support pump. |
| L5 | Assigns the value of "MAIN_B3" to the pump control module PUMP.LEAD3 property. This property determines whether pump 3 is the primary or the support pump. |
| L6 | Comment to describe the program. This line does not run. |
| L7 | Assigns the value of "PRIO_B0" to the pump control module PUMP.PRIO0 property. This property determines the priority of pump 0. |
| L8 | Assigns the value of "PRIO_B1" to the pump control module PUMP.PRIO1 property. This property determines the priority of pump 1. |
| L9 | Assigns the value of "PRIO_B2" to the pump control module PUMP.PRIO2 property. This property determines the priority of pump 2. |
| L10 | Assigns the value of "PRIO_B3" to the pump control module PUMP.PRIO3 property. This property determines the priority of pump 3. |
| L11 | Comment to describe the program. This line does not run. |
| L12 | If the "RESET_FB0" command to reset the pump 0 startup failure is activated, the failure is deactivated, and the command is deactivated again to allow the failure to be activated again at another time. |
| L13 | If the "RESET_FB1" command to reset the pump 1 startup failure is activated, the failure is deactivated, and the command is deactivated again to allow the failure to be activated again at another time. |
| L14 | If the "RESET_FB2" command to reset the pump 2 startup failure is activated, the failure is deactivated, and the command is deactivated again to allow the failure to be activated again at another time. |
| L15 | If the "RESET_FB3" command to reset the pump 3 startup failure is activated, the failure is deactivated, and the command is deactivated again to allow the failure to be activated again at another time. |
| L16 | Comment to describe the program. This line does not run. |
| L17 | If the tank level ("LEVEL") is above the maximum or start-up level setpoint ("HIGHL"), the start indicator is activated ("START" is set to 1 (TRUE)). The status of this indicator is later used as a condition for the command to start the pumps in automatic mode (see line L20). |
| L18 | If the tank level ("LEVEL") is below the minimum or stop level setpoint ("LOWL"), the start indicator is deactivated ("START" is set to 0 (FALSE)). The status of this indicator is later used as a condition for the command to start the pumps in automatic mode (see line L20). |
| L19 | If the tank level ("LEVEL") is above the support level setpoint ("SUPPORTL"), the support indicator is activated ("SUPPORT" is set to 1 (TRUE)). The status of this indicator is later used as a condition for the command to start the pumps in automatic mode (see line L20). |

| Line | Description |
|---|---|
| L20 | If the tank level ("LEVEL") is below the high level setpoint ("HIGHL"), the support indicator is deactivated ("SUPPORT" is set to 0 (FALSE)). The status of this indicator is later used as a condition for the command to start the pumps in automatic mode (see line L20). |
| L21 | Comment to describe the program. This line does not run. |
| L22 | If "MODE" is different from "AUTO", "MANU" or "STOP", then "MODE" is set to "STOP"·. The operating modes indicate whether the system operates automatically based on setpoint levels or manually via remote commands. This line serves as a safety control; if the MODE variable has a value other than 0, 1, or 2, it is considered a fault, and the system is forced into STOP mode. |
| L23 | The property PUMP.HAND, PUMP.AUTO, or PUMP.OFF is activated depending on the value of the "MODE" register. |
| L24 | Comment to describe the program. This line does not run. |
| L25 | The property of the pump control module that determines the number of active pumps ("PUMP.REQ") is calculated as the sum of "REQUESTWATER" and "REQUESTSUPPORT", indicating how many pumps need to be running. |
| L26 | Comment to describe the program. This line does not run. |
| L27 | "RELAY_B0" is reset to the value "PUMP.Q0". The start-up command for pump 0 in the pump control block controls the digital output that activates pump 0. |
| L28 | "RELAY_B1" is reset to the value "PUMP.Q1". The start-up command for pump 1 in the pump control block controls the digital output that activates pump 1. |
| L29 | "RELAY_B2" is reset to the value "PUMP.Q2". The start-up command for pump 2 in the pump control block controls the digital output that activates pump 2. |
| L30 | "RELAY_B3" is reset to the value "PUMP.Q3". The start-up command for pump 3 in the pump control block controls the digital output that activates pump 3. |
| L31 | "PUMP.MAN0" is reset to the value "B0MAN". The command to start pump 0 in the pump control block, whenever manual mode is activated. |
| L32 | "PUMP.MAN1" is reset to the value "B1MAN"". The command to start pump 1 in the pump control block, whenever manual mode is activated. |
| L33 | "PUMP.MAN2" is reset to the value "B2MAN". The command to start pump 2 in the pump control block, whenever manual mode is activated. |
| L34 | "PUMP.MAN3" is reset to the value "B3MAN". The command to start pump 3 in the pump control block, whenever manual mode is activated. |
| L35 | "PUMP.PRC0" is reset to the value "START_B0". Start-up confirmation for pump 0 in the pump control unit. |
| L36 | "PUMP.PRC1" is reset to the value "START_B1". Start-up confirmation for pump 1 in the pump control unit. |
| L37 | "PUMP.PRC2" is reset to the value "START_B2". Start-up confirmation for pump 2 in the pump control unit. |

| Line | Description |
|---|---|
| L38 | "PUMP.PRC3" is reset to the value "START_B3". Start-up confirmation for pump 3 in the pump control unit. |
| L39 | If the automatic pump 0 disable indicator "FAILURE_B0· or the manual disable indicator "FAILUREM_B0" is activated, the corresponding control module input "PUMP.DIS0" is activated to disable the pump. |
| L40 | If the automatic pump 1 disable indicator "FAILURE_B1 or the manual disable indicator "FAILUREM_B1" is activated, the corresponding control module input "PUMP.DIS1" is activated to disable the pump. |
| L41 | If the automatic pump 2 disable indicator "FAILURE_B2 or the manual disable indic<br>ator "FAILUREM_B2" is activated, the corresponding control module input "PUMP.DIS2" is activated to disable the pump. |
| L42 | If the automatic pump 3 disable indicator "FAILURE_B3 or the manual disable indicator "FAILUREM_B3" is activated, the corresponding control module input "PUMP.DIS3" is activated to disable the pump. |

## 17.5 PID PRESSURE/FLOW CONTROL

Drinking water pumping station flow/pressure control.

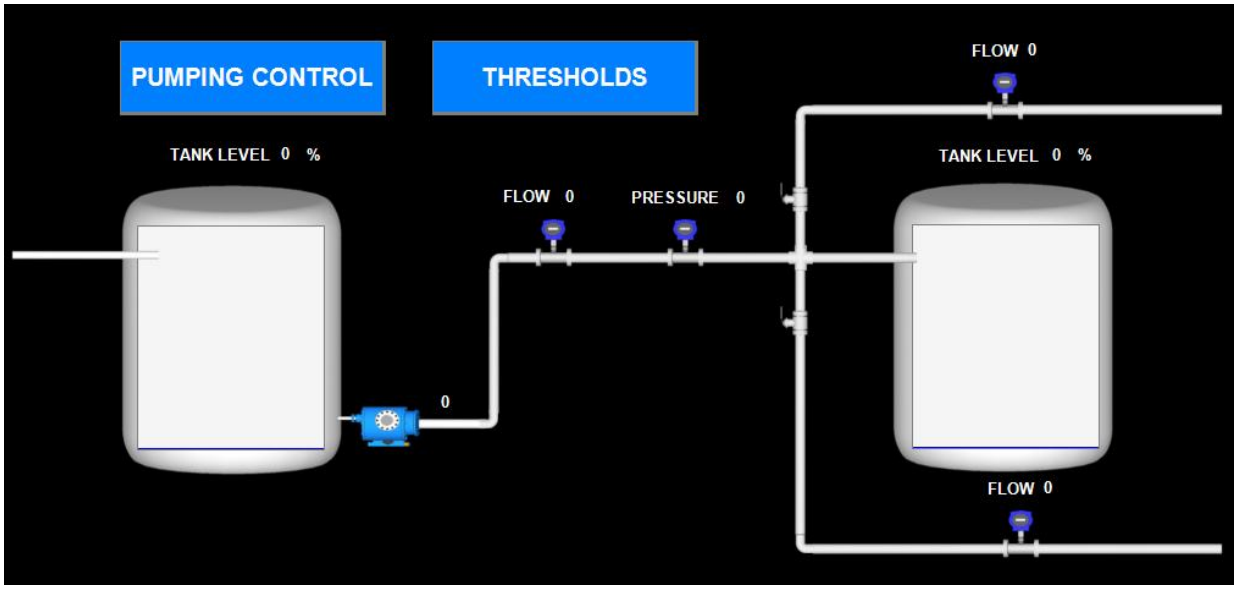
This example demonstrates the application of a PID module in a pressure boosting system where it is necessary to maintain a constant output pressure without exceeding a predefined maximum flow rate.

This involves two PID controllers operating in tandem, one of which is dedicated to pressure regulation and the other to flow rate limitation. The system dynamically evaluates which of the two loops should take control, depending on the operating conditions, where:

- The pressure PID controller operates normally whenever the flow rate stays within the allowed threshold.
- Control is transferred to the flow rate PID if the flow rate reaches the set limit, which indirectly reduces pressure to restrict the flow.

The controller that is not currently in control remains in tracking mode, following the output of the active PID controller to avoid any transients when it resumes control.

This strategy allows for precise pressure regulation without compromising the system's hydraulic limits.



Page 0:

```
L1  #INIT
L2  AI0 : PRESSURE
L3  AI1 : FLOW RATE
L4  M0 : PRESSURE_TARGET
L5  M1 : FLOW_TARGET
L6  M2 : MIN_START-UP
L7  M3 : MODE; REM 0 –> STOP 2 –> AUTO
L8  BOOL : PRES_FLOW; REM 0 –> DOMINATE PRESSURE 1 –> DOMINATE FLOW
L9  INT : STOP=0
L10 INT : AUTO=2
L11 REAL : SETPOINT
L12 EXP16 : PUMP
L13 PID : CONTROL_P,KP=2,TI=10,TD=0,RAMP=0.1
L14 PID : CONTROL_C,KP=2,TI=10,TD=0,RAMP=0.1
L15 #END_INIT
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Assigns the alias “PRESSURE” to the analogue input 0 “AI0”. Input connected to the pressure transducer. |
| L3 | Assigns the alias “FLOW to the analogue input 1 “AI1”. Input connected to the flow meter analogue output. |
| L4 | Assigns the alias “PRESSURE_TARGET” to the math register 0 “M0”. Memory register where the target pressure value is stored. |
| L5 | Assigns the alias FLOW_TARGET” to the math register 1 “M1”. Memory register where the flow target is stored. |
| L6 | Assigns the alias “MIN_START-UP” to the math register 2 “M2”. Memory register that stores the minimum setpoint value for starting up the pump. |
| L7 | Assigns the alias “MODE” to the math register 3 “M3”. This variable indicates the system’s operating mode. |
| L8 | Declares “FLOW_PRES” as a BOOL variable. Indicates whether flow rate or pressure regulation is predominant. |
| L9 | Declares “STOP” as an INT variable and initialises it to a value 0. This is a constant value. |
| L10 | Declares “AUTO” as an INT variable and initialises it to a value 2. This is a constant value. |
| L11 | Declares “SETPOINT as a REAL variable. |
| L12 | Assigns the alias “PUMP” to the analogue output configured in expansion 16 “EXP16”. |
| L13 | Assigns the alias “CONTROL_P” to a PID module with a proportional gain of 2, an integral time of 10, a derivative time of 0, a ramp rate of 0.1, and a maximum output of 200. |

| Line | Description |
|---|---|
| L14 | Assigns the alias “CONTROL_P” to a PID module with a proportional gain of 2, an integral time of 10, a derivative time of 0, a ramp rate of 0.1, and a maximum output of 200. |
| L15 | Keyword to finish the initialisation section. |

Page 1:

```
L1  REM PID PRESURE
L2  CONTROL_P.SP=PRESSURE_TARGET
L3  CONTROL_P.PV=PRESSURE
L4  REM PID FLOW
L5  CONTROL_C.SP=FLOW_TARGET
L6  CONTROL_C.PV=FLOW
L7  REM PID STOP/START
L8  IF MODE<>STOP AND MODE<>AUTO ; MODE=STOP
L9  IF MODE=STOP; PUMP=0
L10 IF MODE=AUTO AND SETPOINT>MIN_START-UP; PUMP=SETPOINT
L11 REM PID DOMINANT
L12 IF FLOW>=TARGET_FLOW;PRES_FLOW=1
L13 IF FLOW<(TARGET_FLOW-HYSTERESIS);PRES_FLOW=0
L14  IF  PRES_FLOW  ;  CONTROL_P.TRK=1  ;  CONTROL_P.TV=CONTROL_C.OUT  ;  CONTROL_C.TRK=0  ;
       SETPOINT=CONTROL_C.OUT
L15 IF  PRES_FLOW=0  ;  CONTROL_C.TRK=1  ;  CONTROL_C.TV=CONTROL_P.OUT  ;  CONTROL_P.TRK=0  ;
       SETPOINT=CONTROL_P.OUT
```

| Line | Description |
|---|---|
| L1 | Comment to describe the program. This line does not run. |
| L2 | Assigns the value “PRESSURE_TARGET” to the PID setpoint, ”CONTROL_P”. |
| L3 | Assigns the value “PRESSURE” to the PID process variable, “CONTROL_P”. |
| L4 | Comment to describe the program. This line does not run. |
| L5 | Assigns the value “FLOW_TARGET” to the PID setpoint “CONTROL_C”. |
| L6 | Assigns the value “FLOW” to the PID process value “CONTROL_C”. |
| L7 | Comment to describe the program. This line does not run. |
| L8 | If “MODE” is different from “AUTO” or “STOP”, then “MODE” is set to “STOP”. This line serves as a safety control; if the MODE variable has a value other than 0 or 2, it is considered a fault, and the system is forced into STOP mode. |
| L9 | If “MODE” is equal to “STOP”, the pump is turned off. |
| L10 | If “MODE” is equal to “AUTO” and the “SETPOINT” is greater than “MIN_STARTUP”, the setpoint value is taken from the analogue output expansion “PUMP”. |

| Line | Description |
|---|---|
| L11 | Comment to describe the program. This line does not run. |
| L12 | If the current flow rate exceeds the target flow rate, the flow rate PID controller takes control ("PRES_FLOW=1"). |
| L13 | If the current flow rate is lower than the target flow rate less the configured hysteresis, the pressure PID takes control ("PRES_FLOW=0"). |
| L14 | If flow rate control is the primary mode "PRES_FLOW=1", the tracking mode of PID "CONTROL_P" is activated, the PID output "CONTROL_C" is fed into the track value input of PID "CONTROL_P", the tracking mode of PID "CONTROL_C" is deactivated, and the output of PID "CONTROL_C" is set as the setpoint. |
| L15 | If flow rate control is the primary mode ("PRES_FLOW=0"), the tracking mode of PID "CONTROL_C" is activated, the PID output "CONTROL_P" is fed into the track value input of PID "CONTROL_C", the tracking mode of PID "CONTROL_P" is deactivated, and the output of PID "CONTROL_P" is set as the setpoint. |

## 17.6 MODBUS MEMORY MAP CONFIGURATION IN SLAVE MODE

Memory map configuration example.

- At address 30001, the value of analogue input 0 is mapped as a 32-bit single-precision floating-point number in "Big Endian" format. The H and L properties of the variable R are used for this purpose.
- At address 30003, the value of analogue input 1 is mapped as a 32-bit single-precision floating-point number in "Big Endian" format. The H and L properties of the variable R are used for this purpose.
- At address 30005, the value of the totalizer counter 0 is mapped as a 32-bit integer in "Big Endian" format. The H and L properties of the variable I are used for this purpose.
- At address 30007, A status word is mapped, which contains the validity flag status of analogue inputs 0 and 1 in bits 0 and 1. The Bx properties of the variable I are used for this purpose.
- At address 30008, the digital input status is mapped as a 16-bit integer.

```
L1  #INIT
L2  REAL : R
L3  INT : U
L4  #END_INIT
L5  R = AI0
L6  MBIR0 = R.L
L7  MBIR1 = R.H
L8  R = AI1
L9  MBIR2 = R.L
L10 MBIR3 = R.H
L11 I = CNT0
L12 MBIR4 = I.L
L13 MBIR5 = I.H
L14 I = 0
L15 I.B0 = AI0.VLD
L16 I.B1 = AI1.VLD
L17 MBIR6 = I
L18 MBIR7 = DIP
```

| Line | Description |
|---|---|
| L1 | Keyword to start the initialisation section. The initialisation section is executed only once when the script starts running. |
| L2 | Declares "R" as a REAL variable. |
| L3 | Declares "I" as an INT variable. |
| L4 | Keyword to finish the initialisation section. |
| L5 | Loads the value of analogue input 0 in the variable "R". |

| Line | Description |
|---|---|
| L6 | Loads the 16 least significant bits of the variable “R” into the MBIR0 (MODBUS Input register 0). |
| L7 | Loads the 16 most significant bits of the variable “R” into the MBIR1 (MODBUS Input register 1). |
| L8 | Loads the value of analogue input 1 in the variable “R”. |
| L9 | Loads the 16 least significant bits of the variable “R” into the MBIR2 (MODBUS Input register 2). |
| L10 | Loads the 16 most significant bits of the variable “R” into the MBIR3 (MODBUS Input register 3). |
| L11 | Loads the value of the totalizer counter 0 in the variable “I”. |
| L12 | Loads the 16 least significant bits of the variable “I” into the MBIR4 (MODBUS Input register 4). |
| L13 | Loads the 16 most significant bits of the variable “I” into the MBIR5 (MODBUS Input register 5). |
| L14 | Initialises the variable “I” to 0. |
| L15 | Loads the value of the analogue input 0 validity bit into bit 0 of variable “I”. |
| L16 | Loads the value of the analogue input 1 validity bit into bit 1 of variable “I”. |
| L17 | Loads the value of the variable “I” into the MBIR6 (MODBUS Input register 6). |
| L18 | Loads the digital input port status into the MBIR7 (MODBUS Input register 7). |