NavVision SoftpLC Manual

Product House

Publication type:	NavVision SoftpLC Manual
Publication number:	PH
Title:	PH-NAvVision-SoftPLC-Manual v2.1.1
Subject:	Product House
Issue:	2.1.1
Publication date:	11 May 2015
Total number of pages:	23
Author:	Vince Kerckhaert
Quality Control:	

Table of contents

Figures

References

[1] PH-NavVision-Operators-Manual v2.1.2

Introduction

A Programmable Logic Controller, PLC or Programmable Controller is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or light fixtures.

Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed-up or non-volatile memory. A PLC is an example of a hard real time system since output results must be produced in response to input conditions within a limited time, otherwise unintended operation will result.

FT NavVision integrated a PLC in the program itself to make it easier to implement programs for FT NavVision.

Abbreviations list

OR	Logic OR
XOR	Exclusive OR
SELECT	Select
TON	Timer ON
TOF	Timer OFF
TP	Pulse Timer
R-TRIG	Rising Trigger
F-TRIG	Falling Trigger
MUL	Multiply
DIV	Divide
AD	Add
SUB	Subtract
LT	Lesser Than
GT	Greater Than
LE	Lesser or equal
GE	Greater or equal
EQ	Equal
NE	Not Equal
RS	Reset before Set
SR	Set before Reset
MAX	Maximum
MIN	Minimum
LIMIT	Limit
CTU	Up Counter
CTD	Down Counter
CTUD	Up-Down Counter
PULSE	Pulse

About the Operating Manual

This manual contains the following sections:

- *Safety instructions* presents warning, caution and note information, which the user should pay attention to;
- *Installation and Settings* handles about all the necessary steps to install and setup the mobile NavVision version;
- *Program* handles about all the settings within the program itself;

For specific information on interfaces, but also in depth information on features, mentioned here or not, we refer you to their specific manuals which can be obtained through Imtech.

Safety instructions

This section provides only a summary of the safety requirements and notes in the following sections. To protect your health and prevent damage to the AM(C)S equipment or vessel, it is essential to read and carefully follow the safety instructions.

The indications NOTE, CAUTION and WARNING have the following significance:

NOTE:
An operating procedure, practice or condition etc., which it is important to emphasize.

CAUTION:
An operating procedure, practise or condition etc., which, if not strictly observed, may damage AM(C)S equipment or crash NavVision software.

WARNING:
An operating procedure, practise or condition etc., which, if not carefully observed may result in personal injury or damage to the vessel.

Revision history

Revisions issued since publication.

Issue	Date	Revision	Reason
1.0	January 31, 2013		initial release
2.1.1	February 6, 2015	Decimus update	Divers

1. Softplc

1.1 Introduction

As explained in the introduction the PLC is a programmable controller. It is used to make some logic controls that automate the operation and functionality of certain processes.

Normally this is programmed right into the PLC itself. The pro is that it can work stand alone. The con is that the programming of a PLC is time consuming and you need to know everything about the specific PLC. Especially when you need it to interact with FT NavVision© it becomes quit a hassle.

1.2 General

A PLC (programmable logic controller) is an electronic device with a microprocessor that, on the basis of its various inputs, controls its outputs. A good example is the Wago PLC that we use often with our system. To make it easier to use and also to extend the range to use it with, we developed a soft PLC for FT NavVision. It is way beyond the scope of this manual to teach you how to program a PLC. That is knowledge that you need to have or that you need to acquire. The functionality is the same as any other PLC programming device.

1.3 Basics

When you open Soft PLC for the first time you get an empty screen (see Figure 1-1)

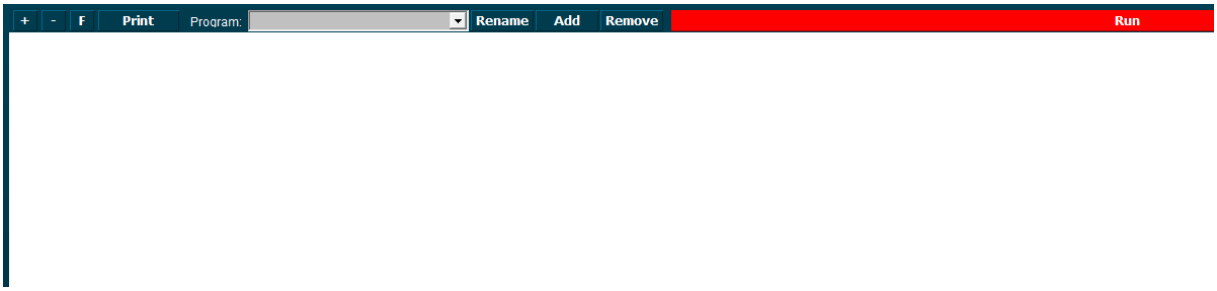

Figure 1-1: Soft PLC

The following figures apply to the buttons on the screen:

Soft PLC Switch	Function
+/-	Zoom in or out
F	Goto Fullscreen mode
Print	Print the Ladder Diagram
Program	Choose which PLC program you want to adjust by clicking the dropdown button
Rename	Rename the PLC program
Add	Add a new PLC program
Remove	Remove a PLC program
Run	Manually run or stop a specific PLC program

1.4 Simple example

Just to explain how it works, we will show a small example. This is merely to show how the diverse methods of implementing work in case you are already familiar with PLC programming.

1.4.1 Start

When you click “Add” you will start a new program. This program starts with an empty line and is called “SoftPLC1” if it is the first program you start. If you click “Rename” you can give it a distinctive name, which will pay off when you have a lot of PLC programs in your system (see Figure 1-2).

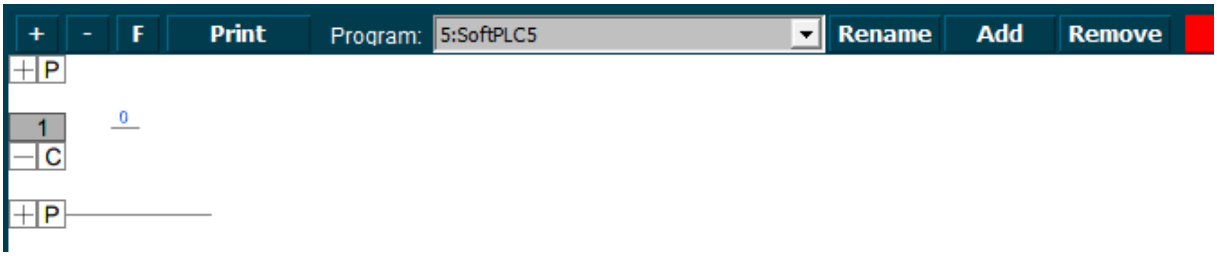

Figure 1-2: SoftPLC Rename

Once you renamed it, you can go on with the program. For those familiar with PLC programming, you will recognize this as a ladder diagram. With the “+” you can add lines before or after and with “–“ you can remove the line. With the “C” you can copy the whole line and with the “P” you can paste the whole copied line

We start this program with a bilge pump, which should run when a certain bilge alarm is high. When you click at the left side of the line (next to the 0) a new pop-up appear with choices (see Figure 1-3).

Figure 1-3: SoftPLC pop-up

You can choose between the following options:

Variable	This can be a free text to set your own variables (see Figure 1-4).
Field	This is one of the fields within NavVision (see Figure 1-5).
Value	This is a free chosen value for calculation purposes (see Figure 1-6).
Function	This is a logical function you can choose from (see Figure 1-7).

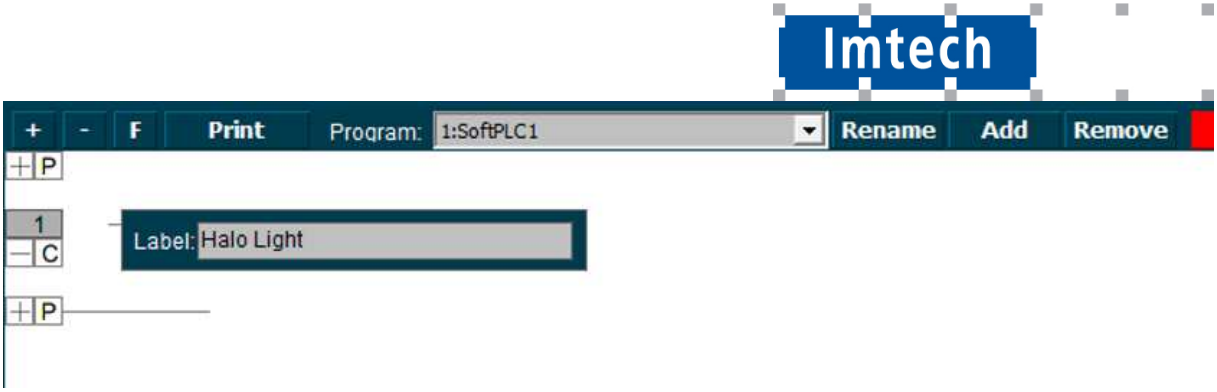

Figure 1-4: Variable

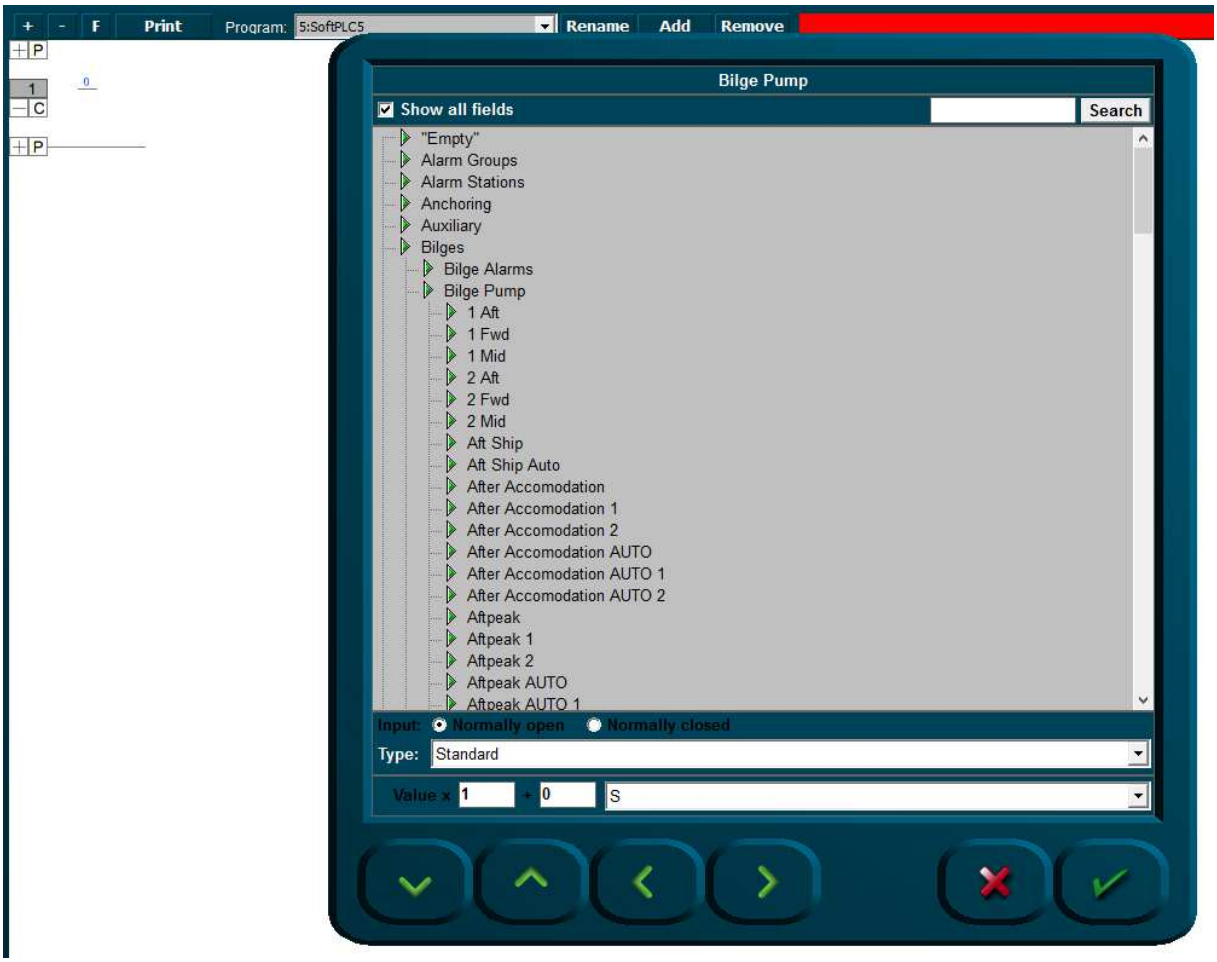

Figure 1-5: Field

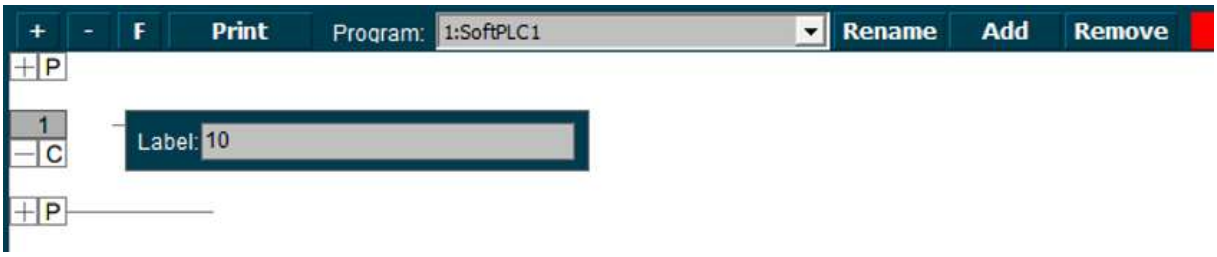

Figure 1-6: Value

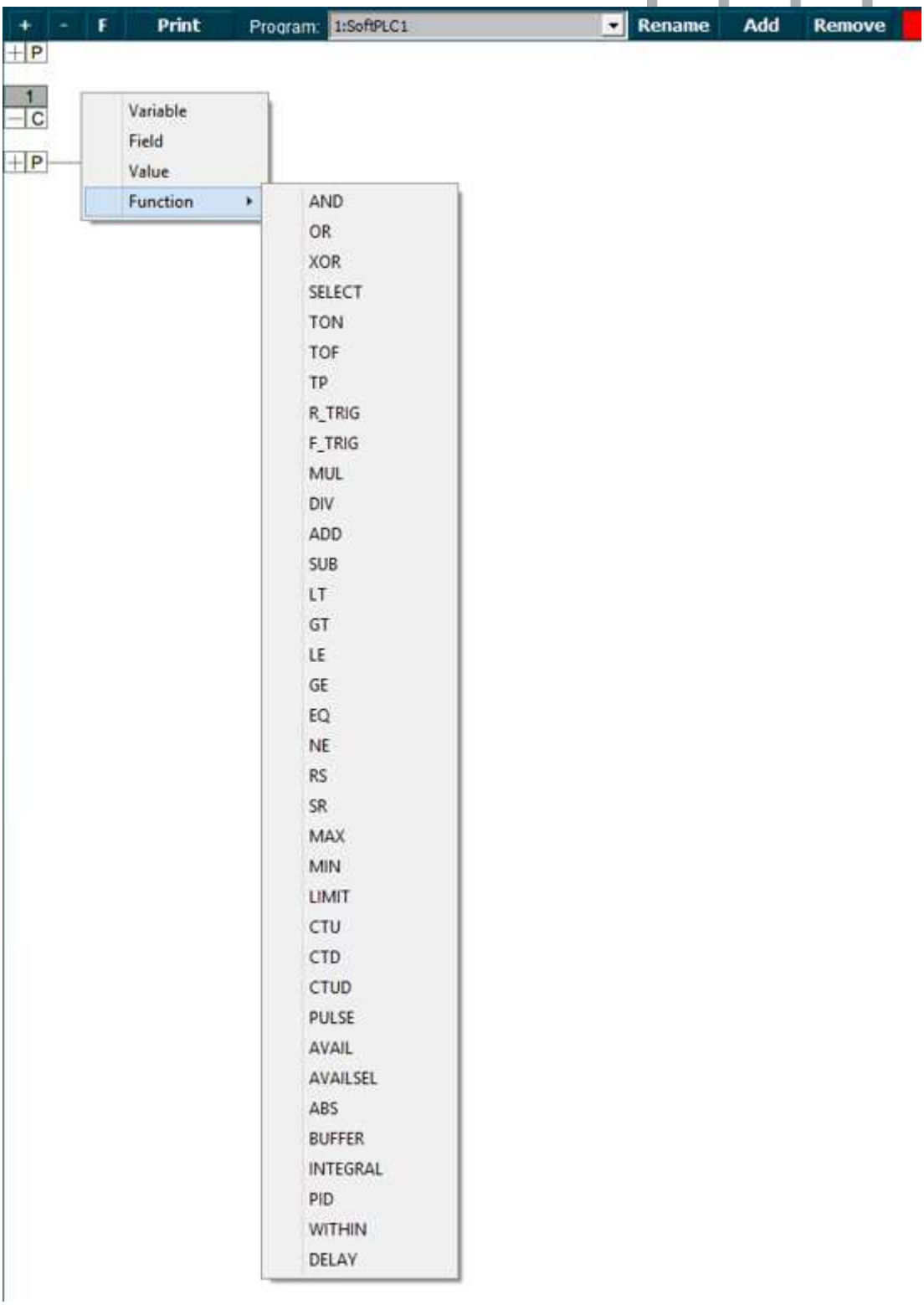

Figure 1-7: Function

Function	Explanation
Function AND	Logic AND
Function OR	Logic OR
Function XOR	Exclusive OR
Function SELECT	Select
Function TON	Timer ON
Function TOF	Timer OFF
Function TP	Pulse Timer
Function R-TRIG	Rising Trigger
Function F-TRIG	Falling Trigger
Function MUL	Multiply
Function DIV	Divide
Function AD	Add
Function SUB	Subtract
Function LT	Lesser Than
Function GT	Greater Than
Function LE	Lesser or equal
Function GE	Greater or equal
Function EQ	Equal
Function NE	Not Equal
Function RS	Reset before Set
Function SR	Set before Reset
Function MAX	Maximum
Function MIN	Minimum
Function LIMIT	Limit
Function CTU	Up Counter
Function CTD	Down Counter
Function CTUD	Up-Down Counter
Function PULSE	Pulse
Function AVAIL	Shows if a field is available
Function AVAILSEL	Select a field available
Function ABS	Sets absolute value
Function BUFFER	Creates a buffer
Function INTEGRAL	Measures the integral of a value
Function PID	Proportional Integral Derivative Controller
Function WITHIN	Measures if a value (IN) is within parameters
Function DELAY	Delay

Table 1-1: function table

We choose for “Field” to assign the Bilge Alarm as a trigger (see Figure 1-8). Now we get into the NavVision part of the SoftPLC. You’ll get the NavVision field dialog box where you can choose the appropriate field (see Figure 1-8). After choosing the field the PLC line will look as in Figure 1-9

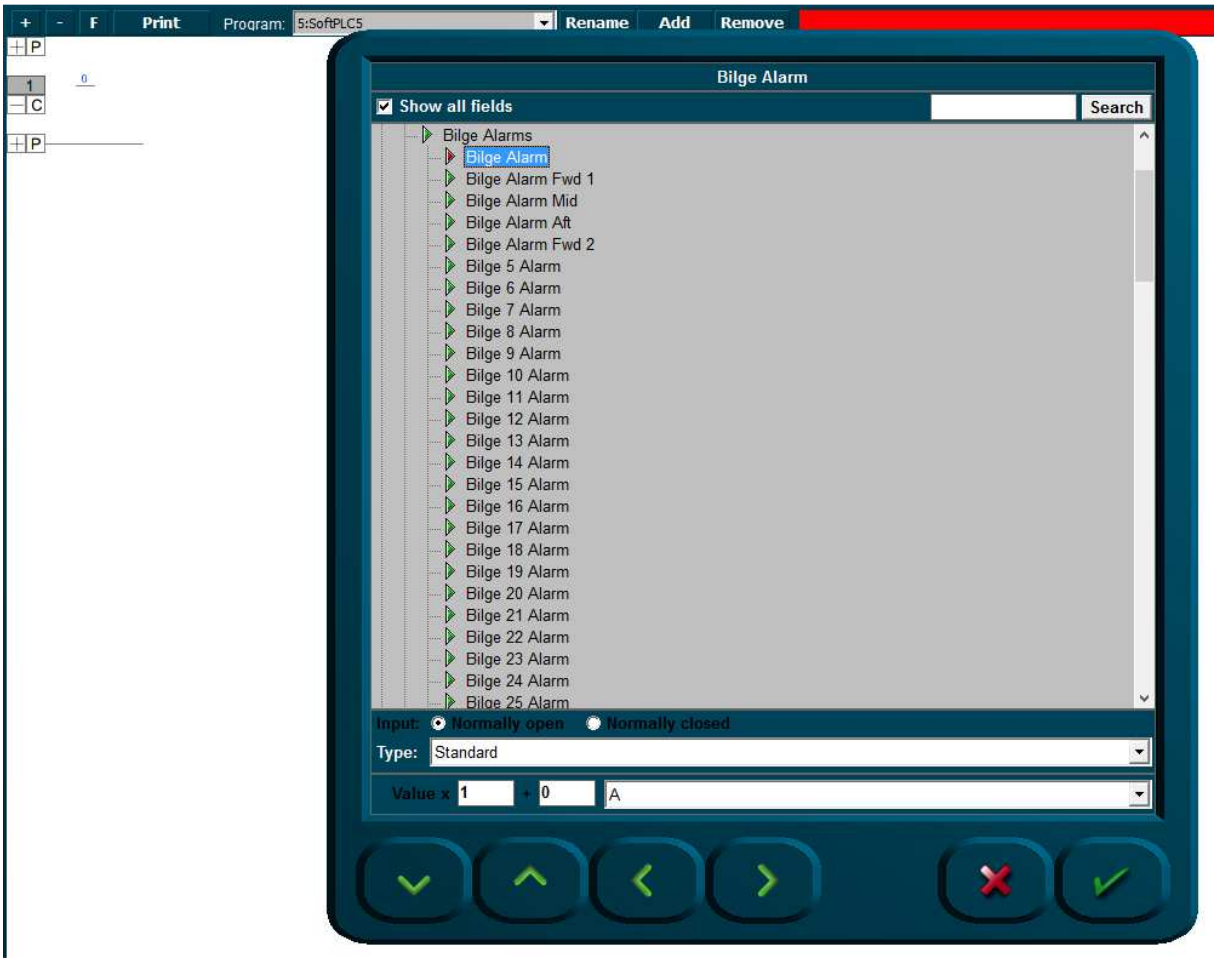

Figure 1-8: SoftPLC Assign Field

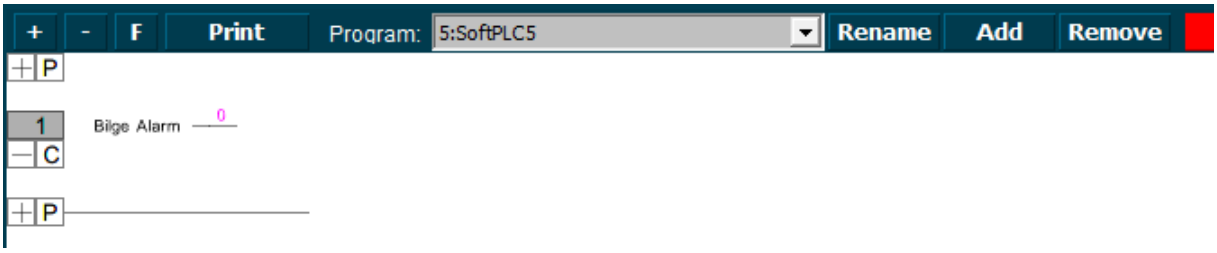

Figure 1-9: SoftPLC first Line

We do the same at the right side of the line but this time we choose the Bilge Pump. We end up with a line like in Figure 1-10.

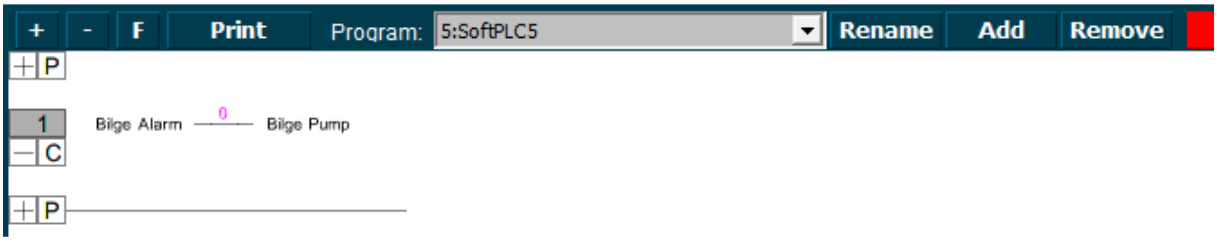

Figure 1-10: SoftPLC First Line_2

So now when you press “Run” the program will run and check the bilge alarm over and over. Once it gets high, the connection in the line gets high (1) and the Bilge Pump starts running until the alarm is not high anymore.

This is just a simple example of a PLC-program. It is not in the scope of this manual to extend on the programming itself. Merely an explanation of the handling of the program itself is in the scope. However we will give a few more examples.

1.5 More functions

If you are familiar with PLC programming, you’ll know that it is mostly about function blocks.

A generalized function block consists of input variables, output variables, through variables, internal variables, and an internal behavior description of the function block. Input variables can only be written from outside of an FB. From inside they can only be read. Output variables can be read and written from inside of an FB and only be read from outside. Through variables are special shared variables. If through variables of different FB instances are connected, they do all access the variable connected to the first input of the chain. Through variables are defined in [IEC 61131]. They are often called In-Out-variables. If their datatype matches, output variables can be connected to input variables by a connector. This is similar to a connection of ports with matching protocols.

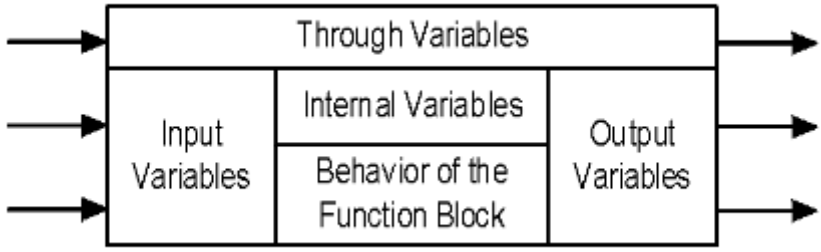

Figure 1-11: General function block model

In Table 1-1 you can see which function blocks are declared in NavVision these are the most commonly used.

If, for example, you want to make a program with a set and reset and different variables, it can be like the following.

Click on the left side of the line and choose”Function” and “AND”. A function block with the AND denominator will appear (see Figure 1-12). Now you can define the fields or variables that need to be true (all) to make the AND box work. In the example we use two denominators. So we need to assign two fields to the left side of the box.

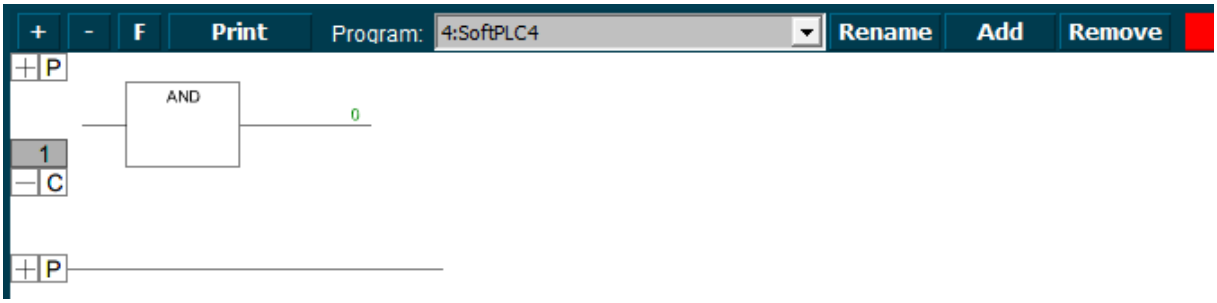

Figure 1-12: the AND box

First click at the left side of the line and choose “Field”. We choose an expansion tank overload (see Figure 1-13). Now we want the pump (that we implement later) to start only as both SB and PS expansion are at overload. We need to add the other tank in front of the ADD-box. Click at the bottom left and choose ”Function” and “AND” and add the other tank (see Figure 1-14).

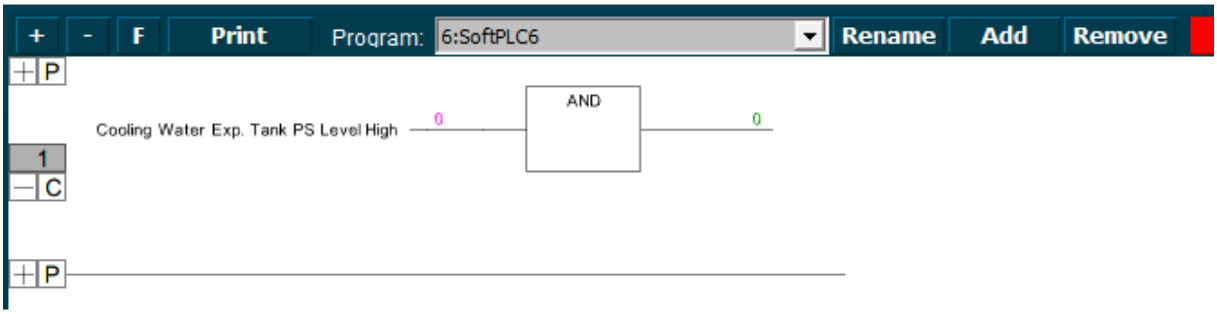

Figure 1-13: Example 1

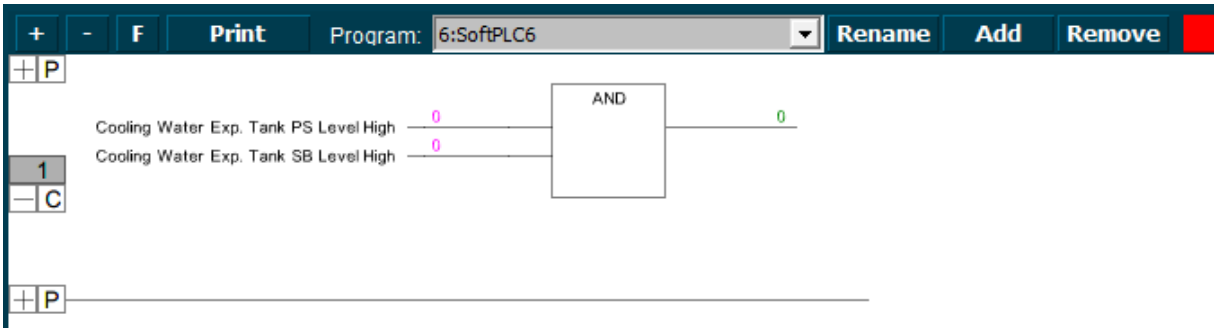

Figure 1-14: Example 2

Now we want the pump we are going to add to be started as both fields are true (1). But we cannot only set the pump, we also need to reset it. So at the right side after the “0” we will put a set/reset box. Click and choose “Function” and “SR” you’ll get a set/reset box (see Figure 1-15).

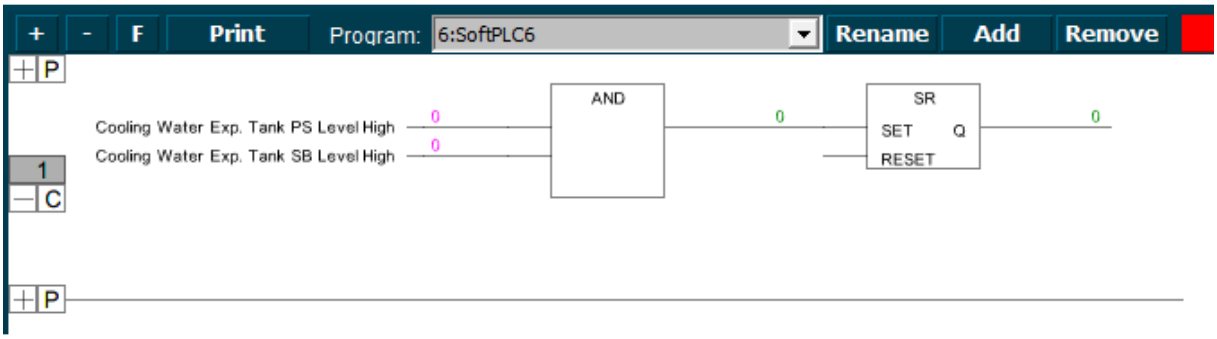

Figure 1-15: Example 3

Now we have the SET-side ready. If both tanks get a high alarm (1) it will get high on the set/reset box which will set the other side of the box. Here we can put the field for the desired pump so that pump will start as the box is set. Click all the way right and choose “Field” and choose the appropriate field. It will look like the following figure:

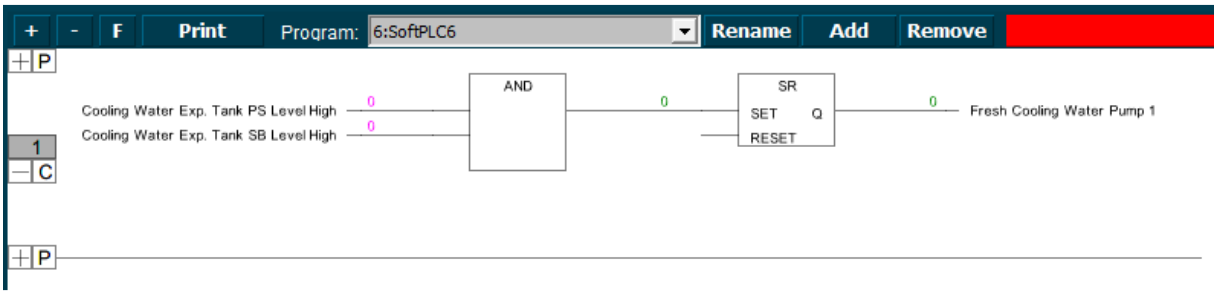

Figure 1-16: Example 4

So, the SET is made. The pump will run once the conditions are met. However we do want to reset the pump as well. In the Set/Reset-box you see a reset entrance. Click here and choose “Function” “AND” or “OR” whichever you want to use. We will use the AND-function for this example. You will get a new “AND” box which we will connect to the same fields as we did in the first box. Except this time we will negate them. This means that the conditions are met if the field is NOT high (1) or in this case they must be low (0) to meet the conditions. You can negate these fields by clicking your mouse at the line near the box. A circle will appear (see Figure 1-17).

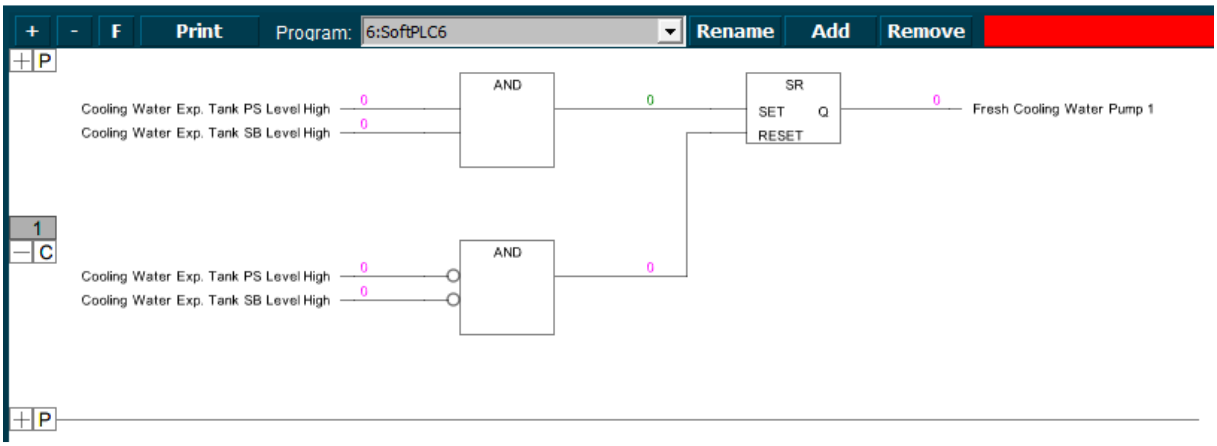

Figure 1-17: Example 5

Conclusion: this program makes the “Fresh cooling water pump” run as both tank level alarms gets high. This will remain so until both tank level alarms are out of alarm after which the switch will reset the pump.

1.6 Extended

Of course you can extend this program as far as you wish, but that Is beyond the scope of this manual. This manual is merely to show you how to work with the SoftPLC. Just as a reference we will show you some more examples to get familiarized with the program.

1.6.1 Example 1

The example we used in the last paragraph is a little bit straight forward. It is just on and off switching of a pump depending on the state of a couple of level switches. Now we can change that to make it more accurate by, for example reset the pump when either of the alarms goes out of alarm (see Figure 1-18). This is easily changed by clicking on the box. A drop-down menu appears where you can choose all the functions (see Figure 1-19).

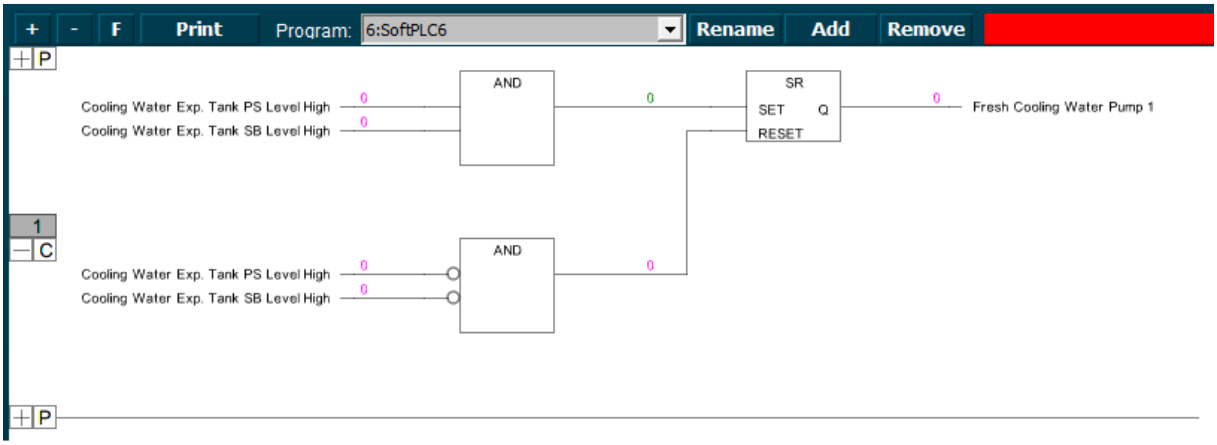

Figure 1-18: Extended example 1

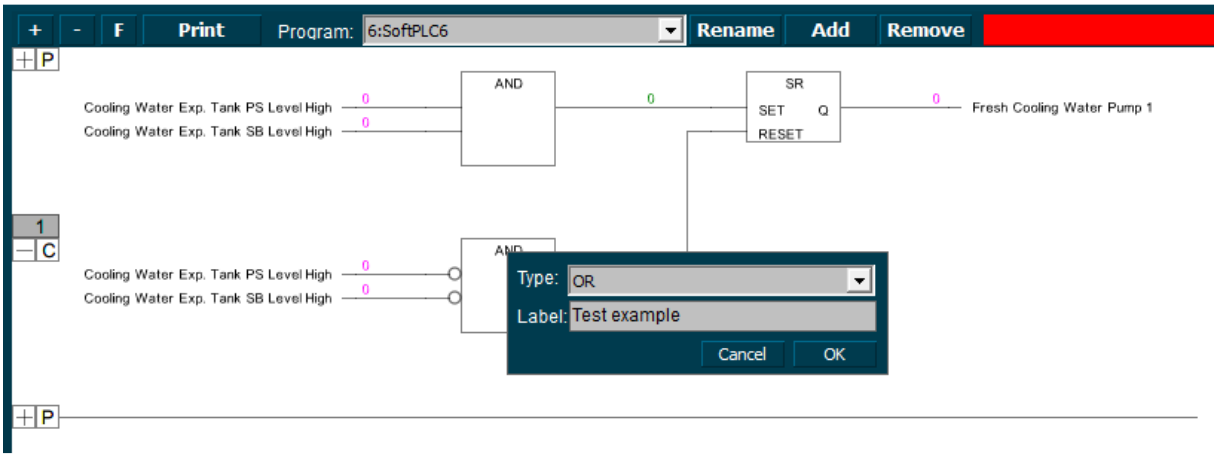

Figure 1-19: Drop down box

You can even give the function blocks a label. Just to keep it synoptic in bigger programs (see and).

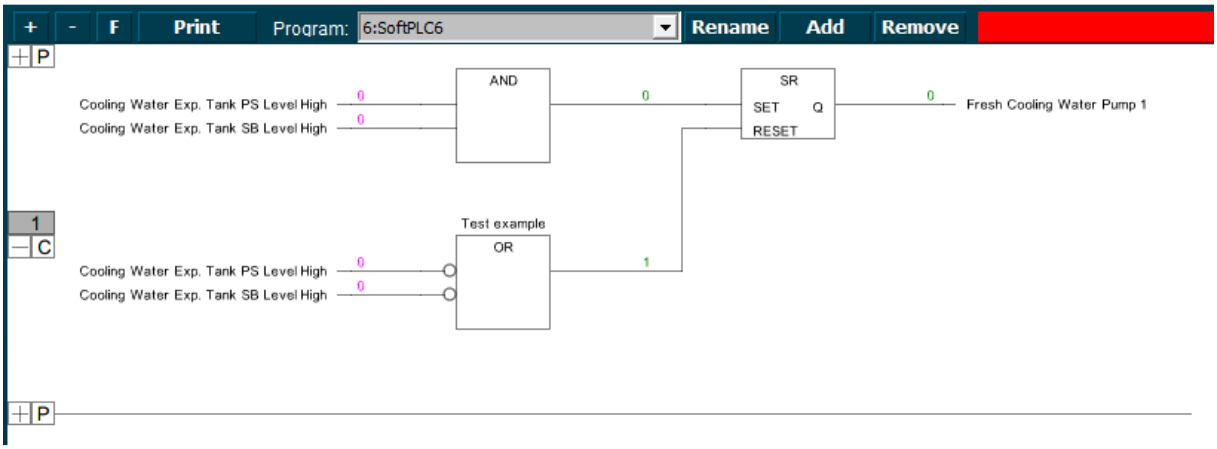

Figure 1-20: Labelling a Function

1.6.2 Example 2

In example 1 we changed AND to OR. So now the pump will be rest if one of the level alarms will get low (0). You can imagine that you want the pump to run a little longer, just to empty a little bit more out of the tanks. This can be done by adding a timer between the OR-box and the reset.

Click at the height of the “number” at the right side of the OR-box. Choose “Function” “TON” A timer box will appear (see Figure 1-21). Now you can set a time (in milliseconds) that the OR-function will hold until it will give the function to the reset. If you like the function to wait for 10 seconds then you click the line before PT and choose “Value”. A box will appear. Type 10000 (for milliseconds which is 10 seconds) and choose OK (see Figure 1-22). Now the timer will wait 10 seconds before it will trigger the reset.

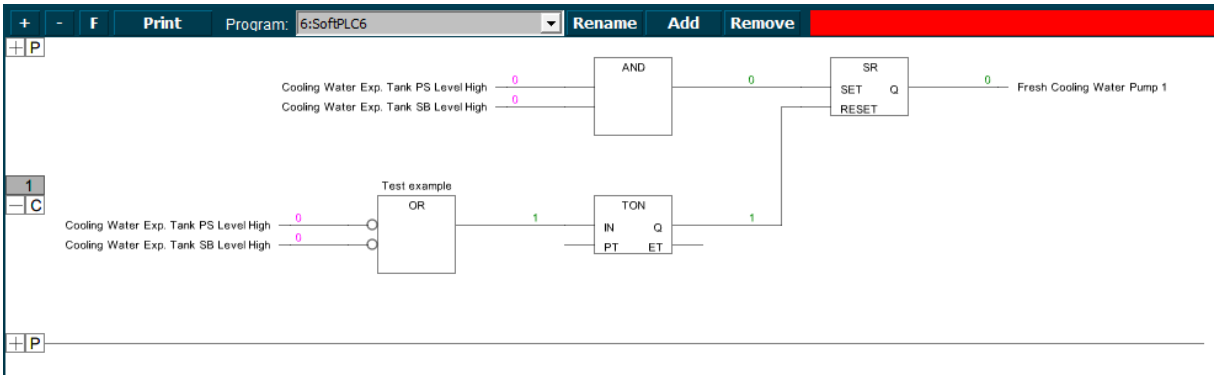

Figure 1-21: Add TON

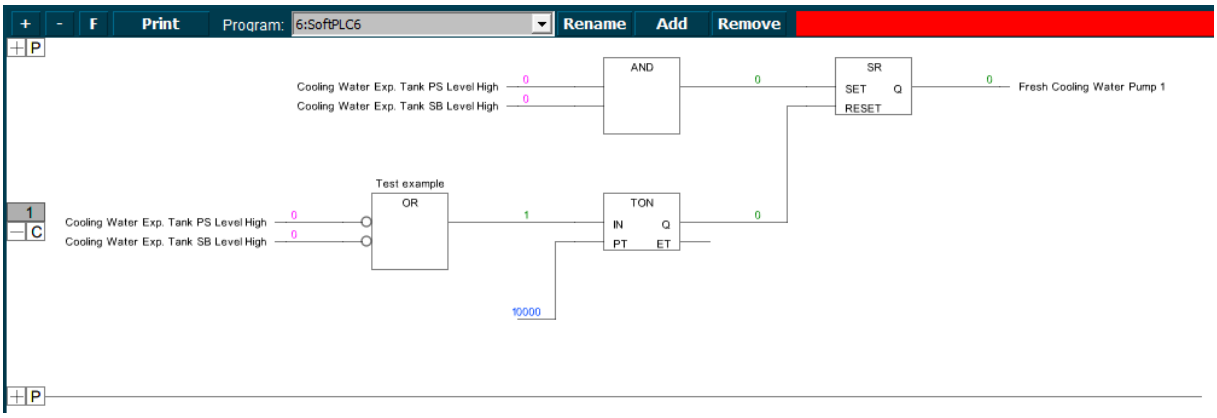

Figure 1-22: Add milliseconds

To remove a particular box, value or field, just hover your mouse over it and press "Delete".

1.7 Specific explanation

Most of the functions will speak for themselves. However a few will be a little bit harder to understand. Especially the input variables and output variables can be a bit confusing. For the less understandable function blocks we will give a clear explanation here.

1.7.1 R_TRIG

The function block R_TRIG detects a rising edge. It looks like the following figure:

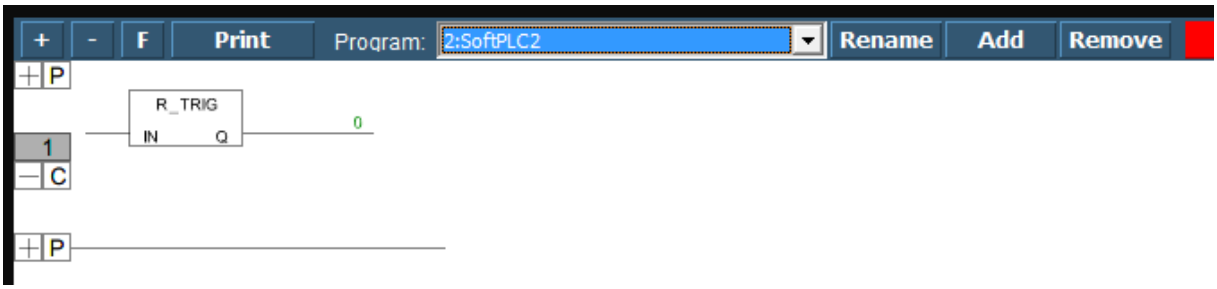

Figure 1-23: R_TRIG

```
FUNCTION BLOCK R_TRIG
VAR_INPUT
 CLK : BOOL;
END_VAR
VAR_OUTPUT
 Q : BOOL;
END_VAR
VAR
 M : BOOL := FALSE;
END_VAR
 Q := CLK AND NOT M;
 M := CLK;
```

The output Q and the help variable M will remain FALSE as long as the input variable CLK is FALSE. As soon as CLK returns TRUE, Q will first return TRUE, then M will be set to TRUE. This means each time the function is called up, Q will return FALSE until CLK has falling edge followed by a rising edge.

1.7.2 LIMIT

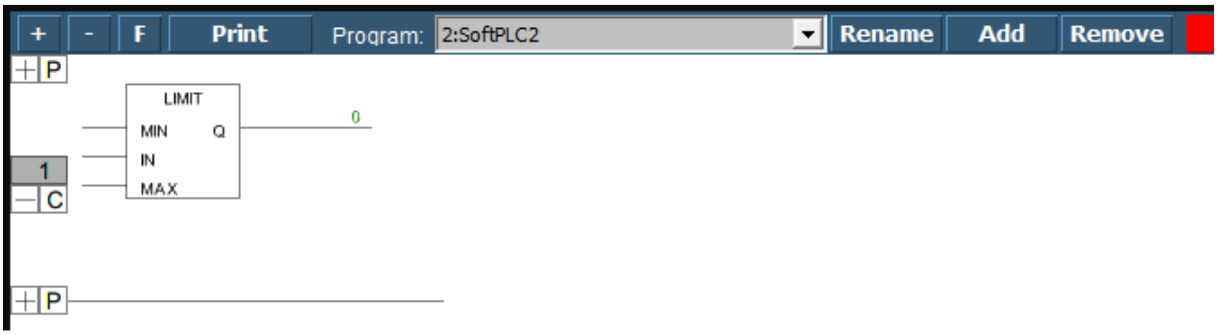

Figure 1-24: LIMIT

OUT := LIMIT(Min, IN, Max) means:
OUT := MIN (MAX (IN, Min), Max)

Max is the upper and Min the lower limit for the result. Should the value IN exceed the upper limit Max, LIMIT will return Max. Should IN fall below Min, the result will be Min.
IN and OUT can be any type of variable.

1.7.3 CTU

Function block Incrementer:

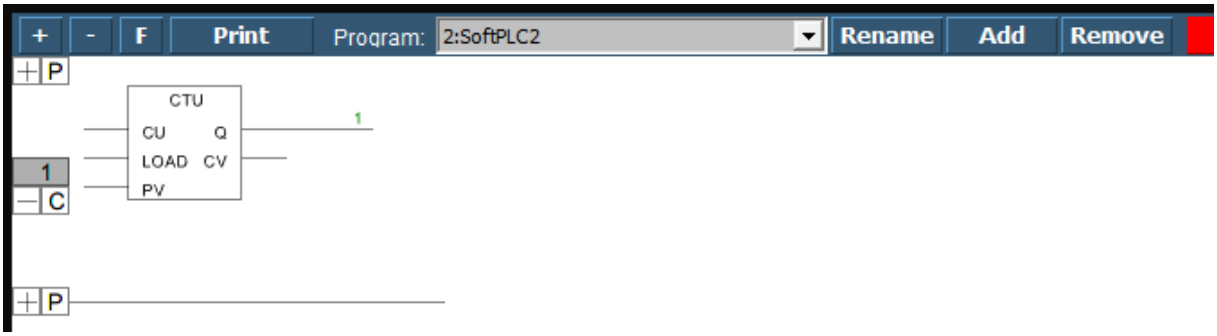

Figure 1-25: CTU

The input variables CU and LOAD as well as the output variable Q are type BOOL, the input variable PV and the output variable CV are type WORD.
The counter variable CV will be initialized with 0 if LOAD is TRUE. If CU has a rising edge from FALSE to TRUE, CV will be raised by 1.Q will return TRUE when CV is greater than or equal to the upper limit PV.

1.7.4 INTEGRAL

This function block approximately determines the integral of the function.

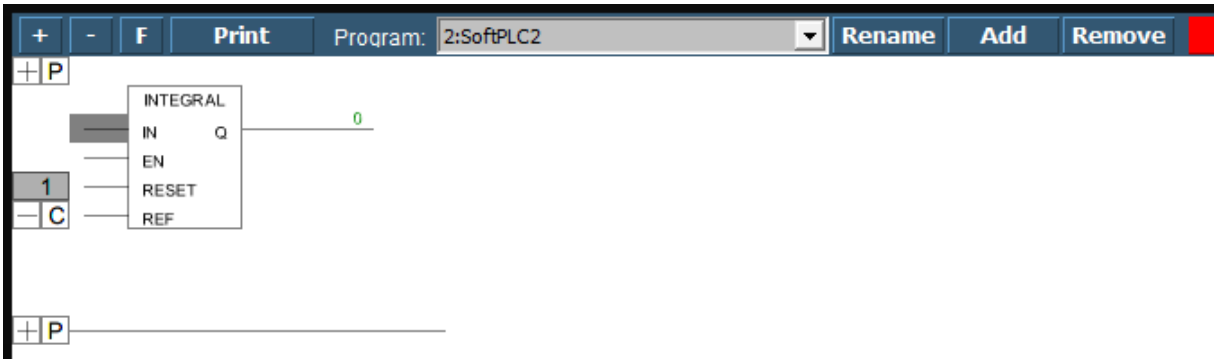

Figure 1-26: INTEGRAL

In an analogue fashion to DERIVATIVE, the function value is delivered as a REAL variable by using IN. EN contains the time which has passed in msec in a DWORD and the input of RESET of the type BOOL allows the function block to start anew with the value TRUE.
The output OUT is of the type REAL.
The integral is approximated by two step functions. The average of these is delivered as the approximated integral.

1.7.5 PID

A proportional-integral-derivative controller (PID controller) is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller calculates an error value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable.

The PID controller algorithm involves three separate constant parameters, and is accordingly sometimes called three-term control: the proportional, the integral and derivative values, denoted P, I, and D. Simply put, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change.The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve, a damper, or the power supplied to a heating element.

A PID controller relies only on the measured process variable, not on knowledge of the underlying process, making it a broadly useful controller. By tuning the three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the setpoint, and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

Some applications may require using only one or two actions to provide the appropriate system control. This is achieved by setting the other parameters to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are fairly common, since derivative action is sensitive to measurement noise, whereas the absence of an integral term may prevent the system from reaching its target value due to the control action.
Unlike the PD controller, this function block contains a further REAL input EN for the readjusting time in sec (e.g. "0.5" for 500 msec).

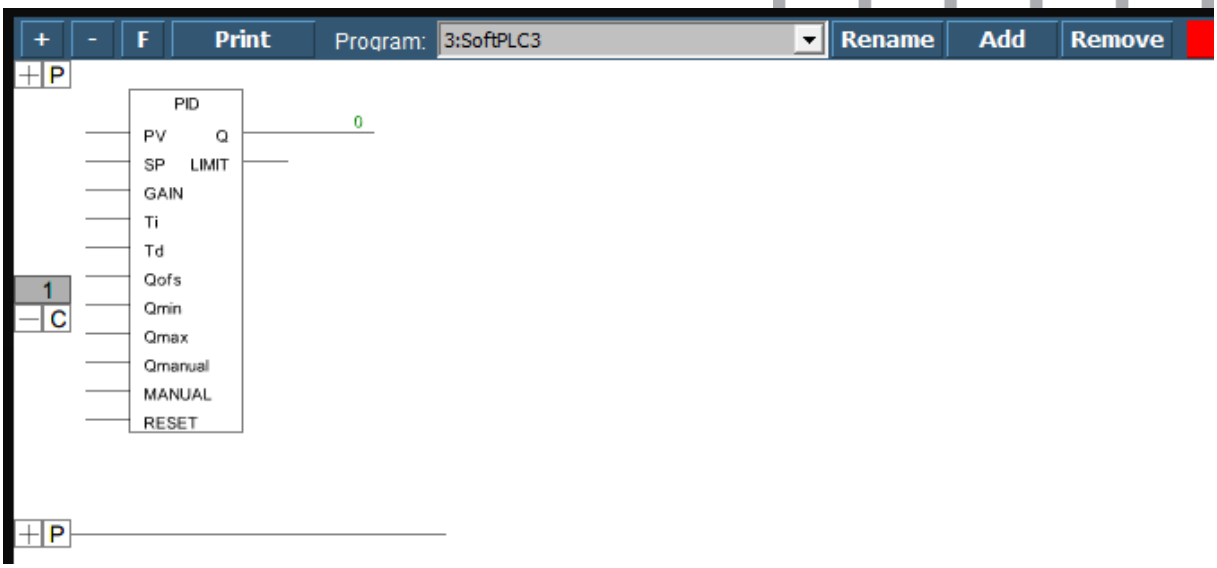

Figure 1-27: PID

Inputs of the function block:

Variable	Data type	Description
ACTUAL	REAL	Current value of the controlled variable
SET_POINT	REAL	Desired value, command variable
KP	REAL	Proportionality coefficient, unity gain of the P-part
TN	REAL	Reset time, reciprocal unity gain of the I-part; given in seconds, e.g. "0.5" for 500 msec
TV	REAL	Derivative action time, unity gain of the D-part
Q_MANUAL	REAL	Defines output value Y in case of MANUAL = TRUE
Q_OFFSET	REAL	Offset for the manipulated variable Y
Q_MIN, Q_MAX	REAL	Lower resp. upper limit for the manipulated variable Y. If

Q exceeds these limits, output LIMITS_ACTIVE will be set to TRUE and Y will be kept within the prescribed range. This control will only work if Q_MIN<Q_MAX.
MANUAL BOOL If TRUE, manual operation will be active, i.e. the manipulated value will be defined by Y_MANUAL.
RESET BOOLTRUE resets the controller; during re-initialization Y = Y_OFFSET.

So besides the P-part also the current change of the controller error (D-part) and the history of the controller error (I-part) influence the manipulated variable.
The PID controller can be easily converted to a PI-controller by setting TV=0.
Because of the additional integral part, an overflow can come about by incorrect parameterization of the controller, if the integral of the error D becomes to great. Therefore for the sake of safety a BOOLean output called OVERFLOW is present, which in this case would have the value TRUE. This only will happen if the control system is instable due to incorrect parameterization. At the same time, the controller will be suspended and will only be activated again by re-initialization.

1.8 SoftPLC.ini attention point

Normally in the network folder there was a file called “SoftPLC.uc.ini” with all the parameters of the SoftPLC inside. This has been changed to make it a bit more readable. For every single SoftPLC program that you write there will be created a new SoftPLC.ini file. So it will be SoftPLC1.uc.ini, SoftPLC2.uc.ini etc. Make sure that this is set correctly in the network folder. If the SoftPLC.uc.ini is already created, you can delete it.