The W.E.St. scripting language

Table of contents

W.E.St. Elektronik GmbH

Gewerbering 31
41372 Niederkrüchten

Tel.: +49 (0)2163 577355-0

Homepage: www.w-e-st.de
E-Mail: contact@w-e-st.de

Date: 21.04.2026

1 Scope, benefits and advantages

Application-oriented solutions for electrohydraulics: this motto characterises the recipe for success of our products. It means that the function of the devices is easy to understand and mostly intuitively usable.

"Parameterising instead of programming" is another guiding principle. The units should be very easy to use, as the structure of the program is sensibly designed for the typical applications of hydraulics.

However, a conflict arises here between a simple program structure and the flexibility to be able to realise special requirements.

The scripting language presented here solves this by supplementing the fixed core of the control application with a flexible shell that is just as easy to set up.

In this way, the devices equipped with the script language become even more versatile and can be easily adapted to all conceivable tasks.

Further advantages are:

- PLC - functionality can be replaced by script functions
- Extremely fast processing in a 1 ms cycle
- Simplest structure of the commands, quick training
- Possibility of an offline simulation of the scripts even without hardware
- Closed loop control functions can also be realised

Hydraulics imposes special requirements on automation, as the high dynamics of the components mean that extremely fast signal processing in the ms cycle is at least advantageous and often unavoidable. Even if the speed of the drives to be controlled appears to be rather low, this is an important aspect for reasons of stability (oscillation tendency exists with dead times) and accuracy. PLCs seldom fulfil these requirements. Due to the necessity of using modules with special speed and resolution, specialised devices are usually more economical and much easier to handle.

Now there is a possibility to keep a great flexibility at the same time, and this at an unbeatable price.

2 Basic concept

A script consists of a list in which predefined memory cells are listed. For each of these cells, one can specify a function with which the content of the memory cell is calculated.

During runtime, these functions are called cyclically and the cell content is recalculated.

There are two types of cells, namely freely usable (M1 ... Mxx) and cells permanently connected to outputs. The content of the latter is either passed directly to physical outputs of the unit or serves as an input signal into a fixed defined internal function, for example as the setpoint of a controller.

The naming and function of the cells depends on the device, M... cells always exist.

The called functions can have up to three parameters, which are themselves memory cells or physical input signals.

3 A first simple example

Consider the following script:

```
M1          = GT   PIN14  PAR1
M2          = LT   PIN14  PAR2
...
LED_YR      = RS   M1     M2
```

The first line (M1) checks whether the input signal at PIN14 is greater than a parameterisable fixed value (PAR1).

The second line (M1) checks whether the input signal at PIN14 is lower than a second parameter.

The output signal, with which the right yellow LED of the unit is controlled, is the switching state of an RS - flipflop, which is connected to the results of these comparisons.

As can be seen, the memory cells can have the meaning of an analogue value as well as a boolean variable. The script interpreter evaluates a content >= 1.0 as logically "TRUE" and functions that provide a logical output value set the corresponding memory cell to 0 or 1.0.

Analogue input and output signals are always scaled in the range 0 ... 100%.

So, in the above example, if you set the parameter PAR1 to the value 50.0 and PAR2 to 40.0, you get a comparator with hysteresis. A voltage > 5 V at PIN 14 will switch on the right yellow LED. The LED lights up until the voltage has dropped below 4 V again.

4 Command overview

What functions are available? An overview will be given here first. A detailed description can be found in chapter 7.

Command:	**Meaning:**	**Operand 1:**	**Operand 2:**	**Operand 3:**
Mathematics				
DIR	direct assignment	Source	-	-
ADD	Addition	Summand 1	Summand 2	Summand 3 (optional)
SUB	Subtraction	Minuend	Subtrahend	-
MUL	Multiplication	Factor 1	Factor 2	Factor 3 (optional)
DMUL	Multiplication + Division	Factor 1	Factor 2	Divisor
LIM	Limitation	Input value	Lower limit	Upper limit
SQRT	Square root function	Input value	-	-
SIN	Sine function	Input value	-	-
ABS	Absolute value	Input value	-	-
NORM(L)	Normalisation to a range	Input value	Supporting point X1	Supporting point X2
UNORM	Scaling	Normalised value (u)	Supp. point Y1 (u=0)	Supp. point Y2 (u=1)
INTEG	Integrator	Input value	Reset	Reset value (optional)
PT1	1st order low pass	Input value	Time constant	Reset
MIN	Minimum value selection	Value 1	Value 2	Value 3 (optional)
MAX	Maximum value selection	Value 1	Value 2	Value 3 (optional)
Logic				
SEL	Signal selector	Switching input (OP1)	Value at OP1 < 1	Value with OP1 >=1
GT	Comparison: OP1 > OP2	Value 1 (OP1)	Value 2 (OP2)	-
LT	Comparison: OP1 < OP2	Value 1	Value 2	-

GE	Comparison: OP1 >= OP2	Value 1	Value 2	-
LE	Comparison: OP1 <= OP2	Value 1	Value 2	-
AND	logical "and"	Value 1	Value 2	Value 3 (set "1" if necessary)
OR	logical "or"	Value 1	Value 2	Value 3 (optional)
NOT	logical negation	Input value	-	-
RS	RS - Flipflop	Set input	Reset input	-
Time functions				
RAMP	1 - Quadrant ramp	Input value	Ramp time	Reset
TE	Switch-on delay	Input value	Time	-
TA	Switch-off delay	Input value	Time	Reset
FP	Edge detection (rising)	Input value	-	-
FN	Edge detection (falling)	Input value	-	-
FUR	Square wave generator	Frequency	Amplitude	Reset
FUS	Sine wave generator	Frequency	Amplitude	Reset
FUT	Triangular wave generator	Frequency	Amplitude	Reset
Miscellaneous (not available with every device)				
FUN2	Secondary function value	-	-	-
SPAR	Read / write parameters	Trigger function	Index	Write value (- for read)

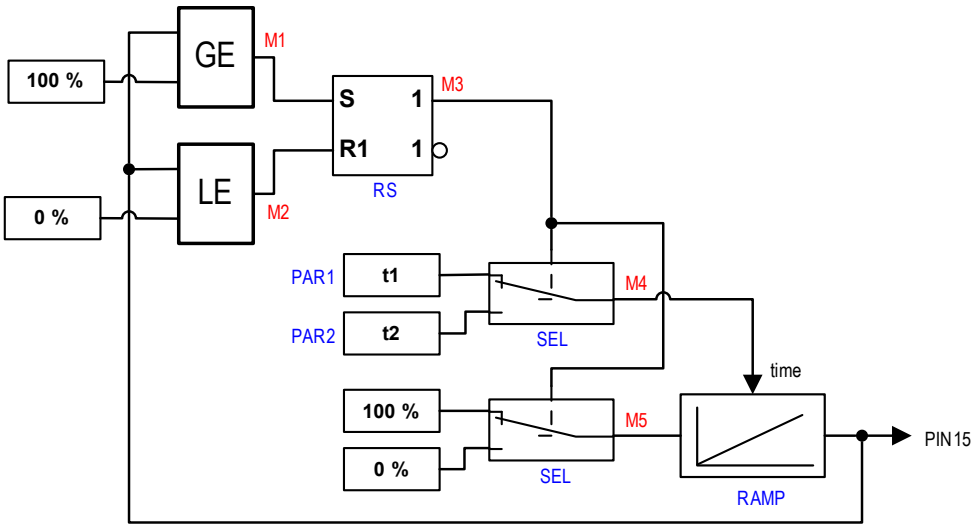
5 A practical exercise

You can follow this example directly. We provide the program "WEStScript" free of charge. You can try out your first scripts with it even without a connected device.

Task: Output of a sawtooth signal at PIN15, the voltage is to rise cyclically from 0V to 10V. The times of the rise and fall should be adjustable:

After looking at the available functions, it is clear that the ramp generator would be a good starting point. One switches both its input signal and its slope when the upper or lower reversal point is reached.

A possible program logic for this looks like this:



The blue font shows the designations of the elements in the script language, the red a possible allocation of the memory cells.

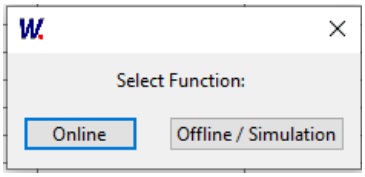
The function is as follows:

The RS flip-flop is set as soon as the output signal is >= 100%. This sets the target value of the ramp to "0" and switches the ramp time to t2 or PAR2. As soon as "0" has been reached, the LE comparison delivers the reset command, so that 100% is again the new target value and t1 or PAR1 the new time specification. This is repeated cyclically.

From top right to bottom left, we now want to translate this little plan into a script and test it.

The program "WestScript.exe" is used for this purpose.

After the call, you are asked whether you want to work online (with module) or in simulation mode:



After selecting the simulation mode, the program requires a file in which the properties of the module to be simulated are described. In particular, the information about the variable pool (sources, memory cells) is contained there.

We select the file "Demomodul.wsm" (*.wsm = West Script Module Definition).

Afterwards, the available memory cells are displayed in the table of the program mask:

Script

Online	Line	Function	Operand 1	Operand 2	Operand 3
	M1				
	M2				
	M3				
	M4				
	M5				
	M6				
	M7				
	M8				
	M9				
	M10				
	M11				

…

	M32				
	PIN15				
	PIN16				
	PIN1				
	PIN2				
	LED_GN				
	LED_YM				
	LED_YR				
	EN_SIG				
	SNAP				

To support the input, a right-click on one of the cells in the "Function" area provides a selection list of possible functions and a right-click in one of the operand fields provides a further subdivided selection of the memory cells, input signals and parameters that can be used as operators.

In the "Input" submenu, some constants such as 100 (%) and "0" are also available.

Exercise:

Now try to enter the function diagram of the last page as a script.

To do this, select the correct function from the table (p.3) for each of the cells to be used and make the connections by selecting the parameters.

Following you can see a sample solution.

Sample solution:

Script

Online	Line	Function	Operand 1	Operand 2	Operand 3
	M1	GE	PIN15	100.0	
	M2	LE	PIN15	0.0	
	M3	RS	M1	M2	
	M4	SEL	M3	PAR1	PAR2
	M5	SEL	M3	100.0	0.0
	M6				

... empty cells ...

	PIN15	RAMP	M5	M4	0.0

To be noted:

- The fixed numbers must be written exactly as shown here. Constants that are not offered in the "Input" menu must be created as parameters.
- The ramp function needs a third operand. Because a reset is not required here, the fixed value "0.0" = *False* was entered.
- You can enter the commands and parameters directly into the cells instead of using the context menu. A syntax check is then not carried out.
- The cell "PIN15" is both an output signal and an operand for other instructions.

After the input, it is first useful to save the work.

To do this, select the menu item File/Save Script. Scripts are saved in CSV format. This is a simple ASCII file that can also be read with a text editor or a spreadsheet, e.g. EXCEL.

But now the exciting question is whether the program also works or produces the desired signal.

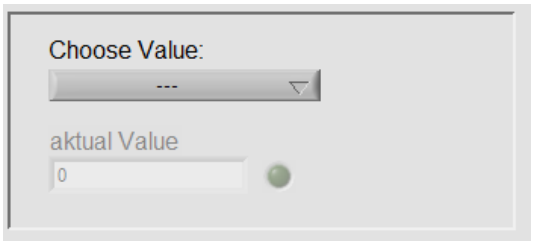
To test this, switch to simulation mode by pressing the button

Additional operating and display elements appear and you should see "living" numbers in the *Online* column. However, the function does not work properly yet, because the parameters PAR1 and PAR2 do not yet contain any usable times. The preset value "0" only results in a very fast signal change.

There are two ways to enter parameter values:

1.) You click with the mouse on one of the PAR. entries in the script table. A dialogue window appears displaying the current parameter value. You can then select the option "Change" and enter a new value. By the way, this method also works in "online" operation on the real module.

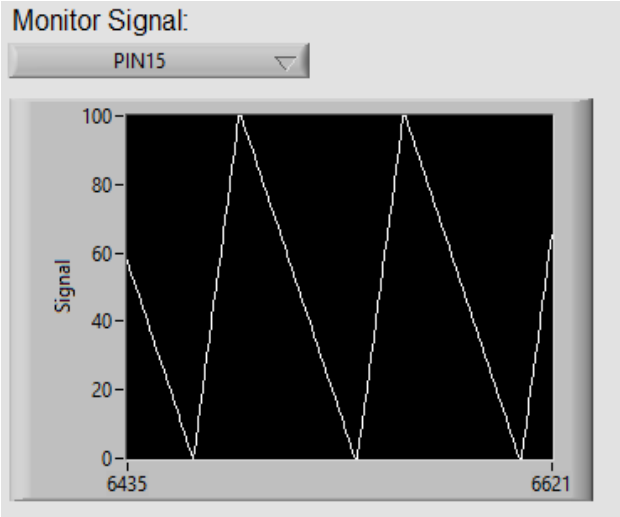
2.) These controls have appeared on the right-hand side:



Via a pull-down menu, one selects one of the input variables or parameters that can be simulated here and then has the option of entering a numerical value. To be able to simulate switching inputs more comfortably, there is an LED symbol to the right of the number field. This indicates whether the current value of a selected variable is >= 1, i.e. is interpreted as "True". By clicking on this field with the mouse, you can quickly change the state.

After you have entered appropriate values here, e.g. PAR1 = 2 (s) and PAR2 = 5 (s), you can see that the numerical value in the online column for PIN15 changes in smaller steps and the flip-flop at M3 switches in the correct cycle.

To get a better impression of the signal course over time there is a small oscilloscope window at the top right. For example, if we select PIN15 for "Monitor signal", we get this recording.



There are some setting options here via the context menu (right mouse button).

Try the options "Autoscaling X" or "Autoscaling Y". If you switch off the auto-scaling in Y - direction, the limits of the displayed range can also be entered directly there after clicking on the numerical values of the axis scale (minimum or maximum).

If you change the signal to be observed (e.g. from PIN15 to M3), you will notice that the scaling no longer fits. Here you can, for example, activate the auto-scaling again.

Pressing the Online/Simulation button again stops the simulation and the elements on the right side of the window are hidden again.

If you would now like to explore the possibilities of the script language a little yourself, these would be suggestions for exercises:

1.) Extend the sawtooth script so that the right LED indicates when the signal at PIN15 is greater than 80%.
2.) The sawtooth should be reset to "0" when a signal is applied to pin7 (switching input).
3.) Write a new script with the following function: PIN 5 is a switching input, PIN 6 is an analogue input. However, you can also use PIN6 as a switching input by querying whether the signal is > 50%. Now program a script with which one can let the output at PIN16 rise with a parameterised speed when PIN5 is actuated and let it fall with the same speed when there is a signal > 50% at PIN6. This corresponds to the "motor potentiometer" function.
4.) Imagine that the script from the previous example is used to set the setpoint in a position controller. When you activate the system, the axis is not necessarily in the "0 mm" position. You therefore want to take over the current position once as the setpoint. This should happen when the input at PIN8 (Enable) delivers a positive edge.

Another hint if you want to insert lines or move the whole block of instructions:

Open the CSV file of the script with EXCEL or similar and use the possibility to move entire blocks in the command and operand area there. Note, however, that the "Line" column remains unchanged. In particular, avoid deleting or inserting complete lines.

After working through these examples, you will already be quite familiar with script programming and you may already be able to think of the first practical applications from your environment.

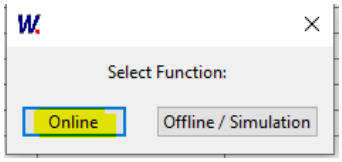
The next chapter will now deal with the step into practice with a real module.

6 Online at the module

6.1 Connect and read out data

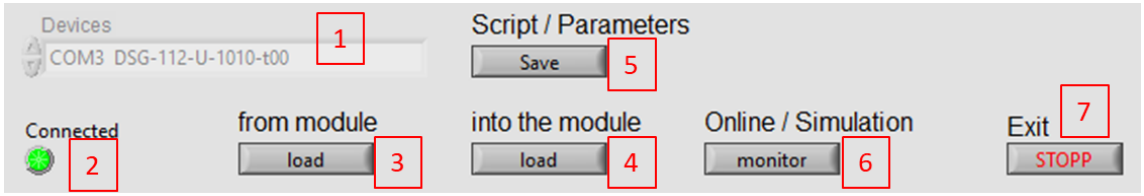
To connect to a module, there must be a USB connection, via which you could also connect via WPC, but WPC must not access the module in question.

The decision whether to work online or offline can only be made when the program is started:



Select "Online

New controls appear:

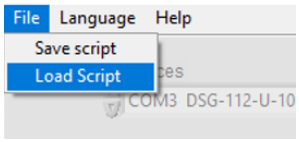


1.) Display of the active connection with the module identification. If several potential candidates (devices from W.E.St.) are found, one must first select the desired connection here. If there is only one possible partner, the connection is established immediately.
2.) The green indicator confirms the successfully established connection.
3.) This button uploads the current script program on the module to the editor.
4.) Transfer of the script from the editor to the module. Attention: The change is effective immediately.
5.) This button causes both the script and the currently set parameters to be permanently stored in the EEPROM of the unit. It corresponds to the homonymous button in the WPC - software.
6.) Activation of the observation mode (see below)
7.) The program should only be terminated via this button.

6.2 Load script created offline or enter script with connected module

If you want to transfer a script file from your computer to the unit, this is done in several steps:

1. Load the script from the file into the editor:



Save: Saves the displayed script table to a file

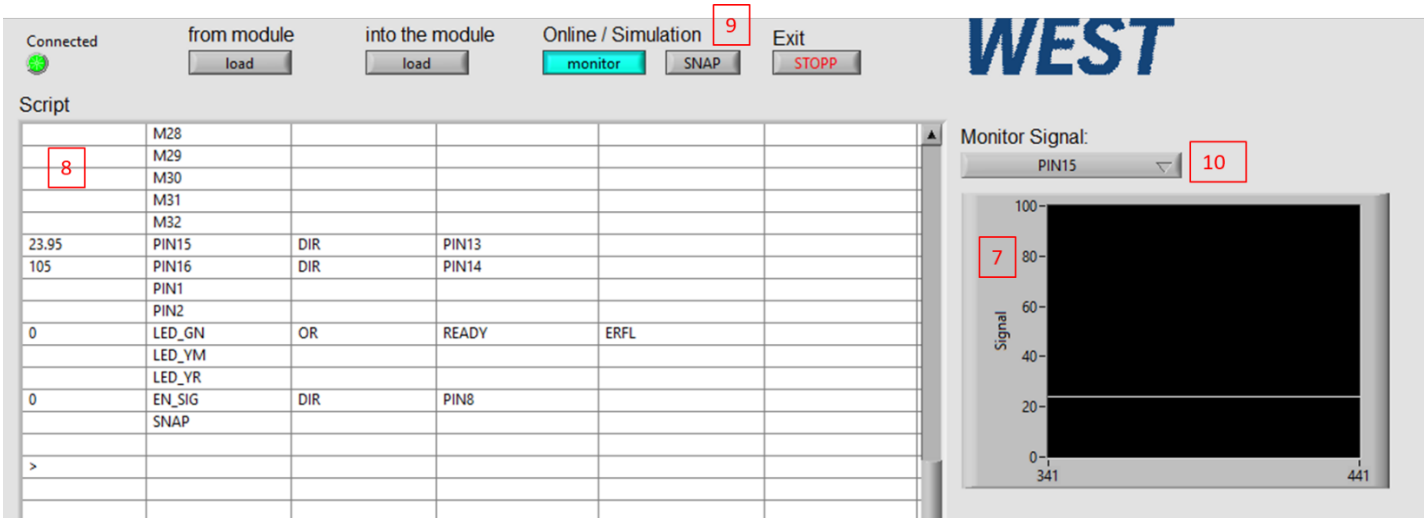
Load: Loads the script from a file into the table

2. Use button 4 (see above) to transfer the contents of the table to the unit.
   If there are faulty commands, the transmission stops at this point.
3. After successful transfer, the changed script is immediately active. You can now make further settings (e.g. parameters) and test the function. However, do not forget to permanently save the data in the non-volatile memory of the unit to complete the activities via the Save button in this software (5) or WPC.

Direct editing of a script in online mode:
After connecting a module, the data of the module definition is automatically read from the unit. As described in chapter 5, one can change the script directly in the displayed table. The context menus can also be activated accordingly via a right click. However, this is only enabled if no observation mode has been activated (see the following section). After changing, the script is loaded into the module by clicking on button 4.

6.3 Observation mode

The observation mode is used for commissioning and checking the script function. If one activates this mode via button 6, the current values for each line are displayed in the "Online" column of the script table (8):

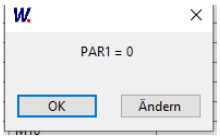


Pressing button 6 again deactivates the observation mode.

Special function possible in observation mode:

Parameter display and change

When (left)-clicking on a free parameter "PAR..." in the table, a dialogue window appears in which the current value is displayed and the possibility to change it is offered:



Please note that display and input are in floating point format, while WPC shows the same value with an artificially shifted decimal point. Example: A value of 1.23 would be displayed here as well, but in WPC as "123".

Signal recorder

In the observation mode, a strip chart (7) is visible in which the temporal course of one of the signals can be displayed. To do this, select a signal of interest via the pull-down menu 10. The scaling of the Y-axis can be changed by right-clicking on its scale: Deactivate autoscaling, then it is possible to change the lower and upper limit in the diagram by clicking directly on the value and entering a number there.

The signal recorder at this point is intended as a tool for quick assessment of individual signals. If you want to record several signals, save the result, etc., the oscilloscope function in WPC is a much more comprehensive and convenient tool.

Snapshot

If you want to reconstruct the situation in case of sporadic events, it is helpful if you can create a copy of the online values at the time in question. There is a special memory cell "SNAP" for this purpose. If the value of this variable rises >= 1.0, a snapshot of the online values is saved at that time. This snapshot can be viewed by pressing the "SNAP" button (9). The snapshot is overwritten with every rising edge of the variable "SNAP" in the table. If you want to save only one state, you can enter e.g. the function RS and connect only the set input. If you select the snapshot view and find only zeros in the online column (including the SNAP line), this means that no recording has been triggered after the unit was started.

7 Function Reference

7.1 Mathematical functions

7.1.1 DIR

Operand 1 is taken as the value of the memory cell.

7.1.2 ADD

Result = Operand 1 + Operand 2 + Operand 3

The specification of the 3rd operand is optional, i.e. if nothing is specified here, only the first two operands are added.

7.1.3 SUB

Result = Operand 1 - Operand 2

7.1.4 MUL

Result = Operand 1 * Operand 2 * Operand 3

The specification of the 3rd operand is optional, i.e. if nothing is specified here, only the first two operands are multiplied.

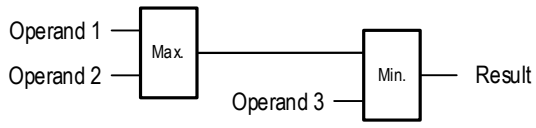
7.1.5 DMUL

Result = Operand 1 * Operand 2 / Operand 3

If you only want to perform a division, set operand 2 = 1.0.

If operand 3 has the value "0,0", the result of the division is undefined. To avoid numerical problems in subsequent functions, the output is also set to "0,0" in this case.

7.1.6 LIM



The value specified for operand 1 is limited to a range between a lower limit (operand 2) and an upper limit (operand 3). Should you specify operand 3 <= operand 2, the result corresponds to operand 3, due to the order of comparisons shown above.

7.1.7 SQRT

Square root function of the input signal. If the input signal is negative, it is inverted, the root is formed and the result is inverted again. In this way, the root function is continued point-symmetrically in the negative real range. This type of calculation makes it possible to use it, for example, to calculate a flow rate on the basis of a differential pressure.

7.1.8 SIN

Sine function, input is given by the angle in radians [rad], a value of 2 Pi corresponds to 360°.

7.1.9 ABS

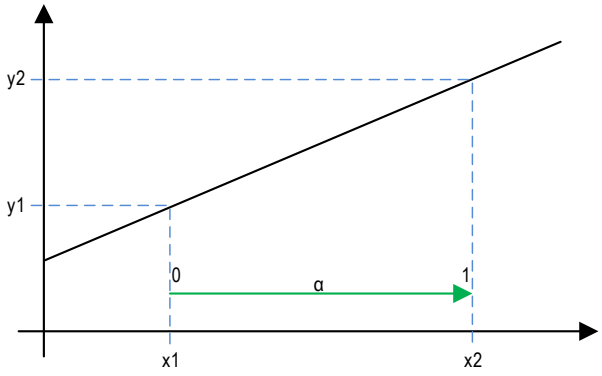
The absolute value of operand 1 is output.

7.1.10 NORM / NORML / UNORM

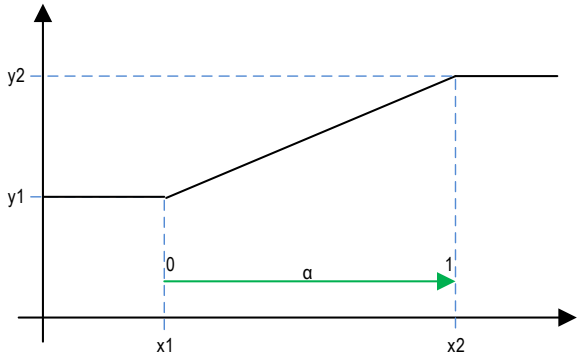
Functions for linearising and rescaling.

It often happens that one wants to convert values in a linear way. Measurement signals are transmitted as standard signals and must be converted to a physical unit, units must be converted, characteristic curves must be approximated by several linear segments, etc. These operations or parts of them can normally be described by a linear equation of the form y = m*x + b and are thus relatively easy to realise by the elementary functions addition and multiplication.

This becomes somewhat inconvenient if instead of the slope m and the y-axis section b, two pairs of values x1/y1 and y1/y2 are given. In this case, m and b must first be calculated from these.



Linearisation without limitation (with extrapolation)



Pure interpolation (limitation at the supporting points)

Another approach to such a task is to first convert the range between the support points to a normalised auxiliary variable and to convert this to the target variable in a second step.

The sketches above show how this is done:

With $\alpha = \frac{u-x1}{x2-x1}$ as an intermediate variable, the input signal u is first transferred to a normalised value α (0 - 1 by x1 .. x2), then the output variable Y can be obtained: $y = \alpha \cdot (y2 - y1) + y1$.

The functions NORM and NORML provide the first step, i.e. the conversion into the normalised intermediate value. The function NORML limits the result to the value range 0.0 - 1.0, so that in combination with the second step a pure interpolation takes place and the range defined by the interpolation points is not exceeded or undercut.

This second step is carried out by the function UNORM.

A practical example:

A pressure sensor with a measuring range of 100 bar at PIN6 of a module measures the pressure in a (plunger) cylinder with which a working platform is lifted. When the platform is unloaded, a pressure of 32 bar (= PAR1) is measured; when the platform is loaded with 8000 kg of weight (=PAR3), the pressure is 87 bar (=PAR2).

To determine the additional load during operation, write:

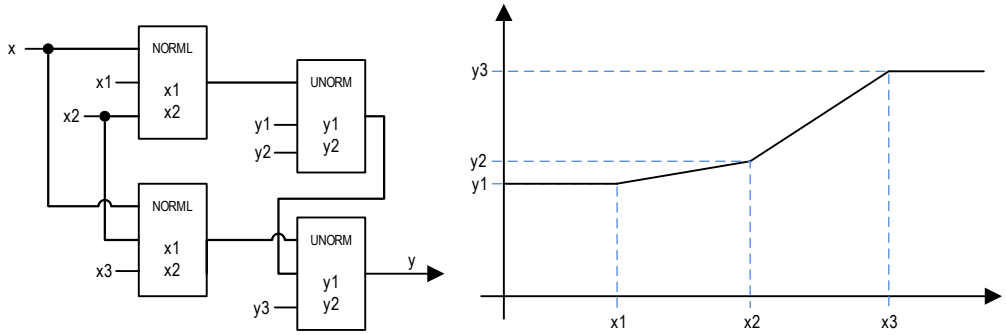
M1 = NORM PIN6 PAR1 PAR2

(With a measuring range of 100 bar, the signal PIN6 (%) corresponds to the pressure in bar)

M2 = UNORM M1 0.0 PAR3

Further application possibility:

Interpolation with several supporting points

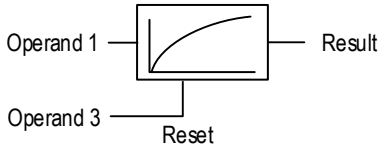


7.1.11 INTEG

Integrator: The signal at operand 1 is integrated (standard slope 1/s, i.e. the output increases by the input signal per second). Operand 2 is used for resetting, as long as this value is >= 1, the output is applied with the signal of operand 3 (0.0 if unconnected). After removing the reset, the integrator continues to run from this value.

7.1.12 PT1

Discrete-time implementation of a 1st order delay element. This function can be used to filter signals.



The operand 2 indicates the time constant in seconds.

The time constant is internally limited to 0.01 s, if one specifies smaller values, the input signal is passed on to the output without delay. The same happens if operand 3 (reset) >= 1.0.

7.1.13 MIN / MAX

Minimum or maximum value selection of the input signals. The smallest or largest of the three operands is passed on to the output.

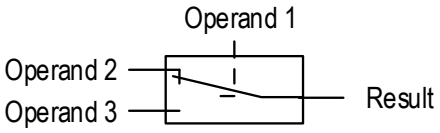
The third operand is optional, i.e. if nothing is provided here, only the first two operands are processed.

7.2 Logic

Here the specification applies that operands as input values are interpreted as TRUE if they are >= 1.0.

Either 1.0 or 0.0 is output as the result. This can be used as an operator for other logic commands or as a numerical value.

7.2.1 SEL



Only operand 1 is considered a Boolean variable. The value of operand 2 or 3 is taken over unchanged as the result.

7.2.2 GT / GE / LT / LE

GT: Result = Operand 1 > Operand 2

GE: Result = Operand 1 >= Operand 2

LT: Result = Operand 1 < Operand 2

LE: Result = Operand 1 <= Operand 2

7.2.3 AND / OR / NOT

Logical standard operations of Boolean variables (not bit patterns).

The AND always requires the specification of three operands. If only two are to be used, the third must be set to "1.0".

OR: The specification of the 3rd operand is optional, i.e. if nothing is specified here, only the first two operands are used.

7.2.4 RS

Truth table: (0 here stands for *false* / 1 for *true*)

Old output value	Operand 1 (S)	Operand 2 (R)	Result
0	0	0	0
1	0	0	1
0 or 1	1	0	1
0 or 1	0	1	0
0 or 1	1	1	0

7.3 Time functions

Overview of these functions and their operands:

Time functions				
Command:	**Meaning:**	**Operand 1:**	**Operand 2:**	**Operand 3:**
RAMP	1 - Quadrant ramp	Input value	Ramp time	Reset
TE	Switch-on delay	Input value	Time	-
TA	Switch-off delay	Input value	Time	Reset
FP	Edge detection (rising)	Input value	-	-
FN	Edge detection (falling)	Input value	-	-
FUR	Square wave generator	Frequency	Amplitude	Reset
FUS	Sine wave generator	Frequency	Amplitude	Reset
FUT	Triangular wave generator	Frequency	Amplitude	Reset

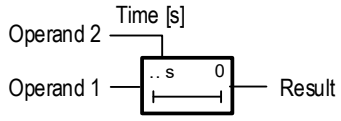
7.3.1 RAMP

This function traces the result to the value of operand 1, but with a maximum speed that is determined with operand 2. It is specified in 100/s, i.e. if the input value changes suddenly by 100 (%), the output will track the change within the time specified by operand 2 in seconds.

If operand 3 is connected and "true", the input value at operand 1 is directly taken over without delay. If you want to set the output to "0.0", a "0.0" must be specified by operand 1.

7.3.2 TE

Standard - switch-on delay:

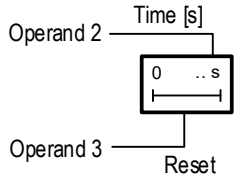


The output switches to "1" when the input (operand 1) has been "1" continuously for at least the time specified with operand 2. If operand 2 changes during this period, the time is extended or shortened accordingly. If the change takes place after the delay time has expired, this has no effect on the result, the output remains set.

If the signal changes to "0" at the input, the result at the output immediately becomes "0" as well.

7.3.3 TA

Standard - switch-off delay with extension (reset input):



The output immediately switches to "1" when the input (operand 1) is set.

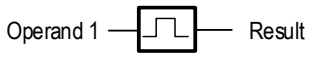
If operand 1 was then continuously "0" for at least the time specified with operand 2, the output is also reset. If operand 2 changes during the runtime, the time is extended or shortened accordingly.

An immediate reset of the delay can be achieved by operand 3. If operand 1 and operand 3 are "1" at the same time, operand 1 has priority and the result at the output remains "1".

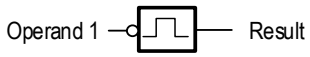
7.3.4 FP / FN

Edge detection of the input signal "Operand 1":

FP: A rising edge at the input sets the output to "1" for one processing cycle.



FN: A falling edge at the input sets the output to "1" for one processing cycle.



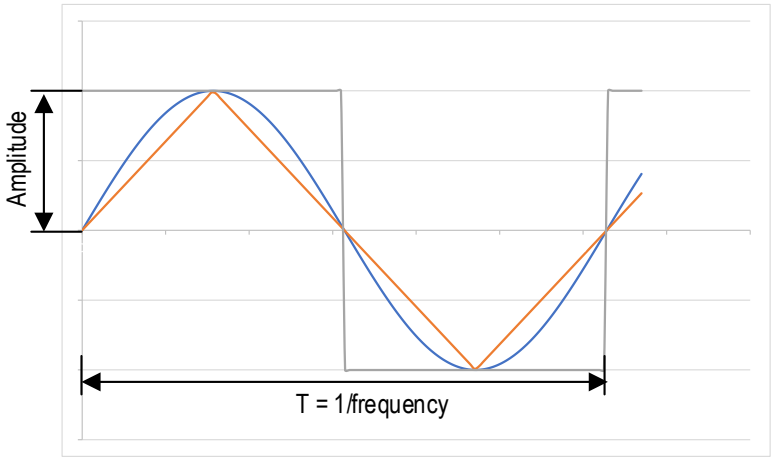
7.3.5 FUR / FUS / FUT

These three functions represent function generators.

FUR = Rectangular wave generator

FUS = Function generator sine

FUT = Function generator triangle



Command:	Meaning:	Operand 1:	Operand 2:	Operand 3:
FUR	Square wave generator	Frequency	Amplitude	Reset
FUS	Sine wave generator	Frequency	Amplitude	Reset
FUT	Triangular wave generator	Frequency	Amplitude	Reset

After resetting, the generators start running as shown above, i.e. sine and triangle start rising from "0", the square wave signal from the positive half.

7.4 Miscellaneous functions

7.4.1 FUN2

Some functions can return two values. The second value is accessed by this function, which always refers to the function in the preceding line.

TE	time already elapsed after activation [s], stops when the set delay time is reached.
TA	time remaining until switch-off in [s], is equal to the delay time when the input signal is active
FUR/FUS/FUT	sawtooth signal from -1.0 to 1.0
SPAR	Write / Read completed (asynchronous processing!)

Further use possible with module-specific complex functions, see associated documentation.

7.4.2 SPAR

Script-controlled reading or writing of parameters.

The presence of a connection to the third parameter determines whether reading or writing takes place.

With this function, it is possible to read or write the parameters of the module-internal complex functions or permanently programmed components of the firmware out of the script. In this way, it is possible, for example, to elegantly realise a parameter switchover or an adjustment of the zero position during operation.

A rising edge at operand 1 triggers the function. A continuous signal only causes the return value to be held, a new read or write requires a signal change.

If no third operand is available, the function reads the parameter that is assigned via an index (operand 2). The valid indices can be found in the respective module documentation.

The return value when reading corresponds to the set parameter in the format in which it is also entered in the parameter table (e.g. 0.0 - 10000.0 for 0 - 100%). If the index does not exist, the function returns the value -1.0.

For writing, the third parameter has to be connected, and the value has to be given in the same format. The return value for writing indicates whether the function was completed successfully (= 1.0). Values outside the limits for the parameter in question will not be accepted.

Please note that this command runs asynchronously and is not always completed in the same cycle in which it was triggered.

Application example: Switching a parameter depending on a switching input

M1 = SPAR PIN5 PAR11 PAR1

M2 = NOT PIN5

M3 = SPAR M2 PAR11 PAR2

PAR11 contains the parameter index, PAR1 and PAR2 the two values to be switched between.

Since the SPAR function already performs an internal edge detection, you can directly connect the input or the negated input as parameter 1.

7.5 Module-specific complex functions

In addition to the functions described here, there are, depending on the module, a varying number of functions that carry out a more extensive calculation with only one command and that have their own parameter scope. These can be controllers, ramps, characteristic curve modules, etc.

These functions are described in the respective module documentation.

An offline simulation of these functions is not possible.

8 Application examples

In the following, some typical and some more demanding applications are shown as examples, as a suggestion and template for your own projects. We will gladly provide you with the scripts belonging to the examples as files. We are also happy to help you with the realisation of your application, please do not hesitate to contact us.

8.1 Oscillating cylinder drive

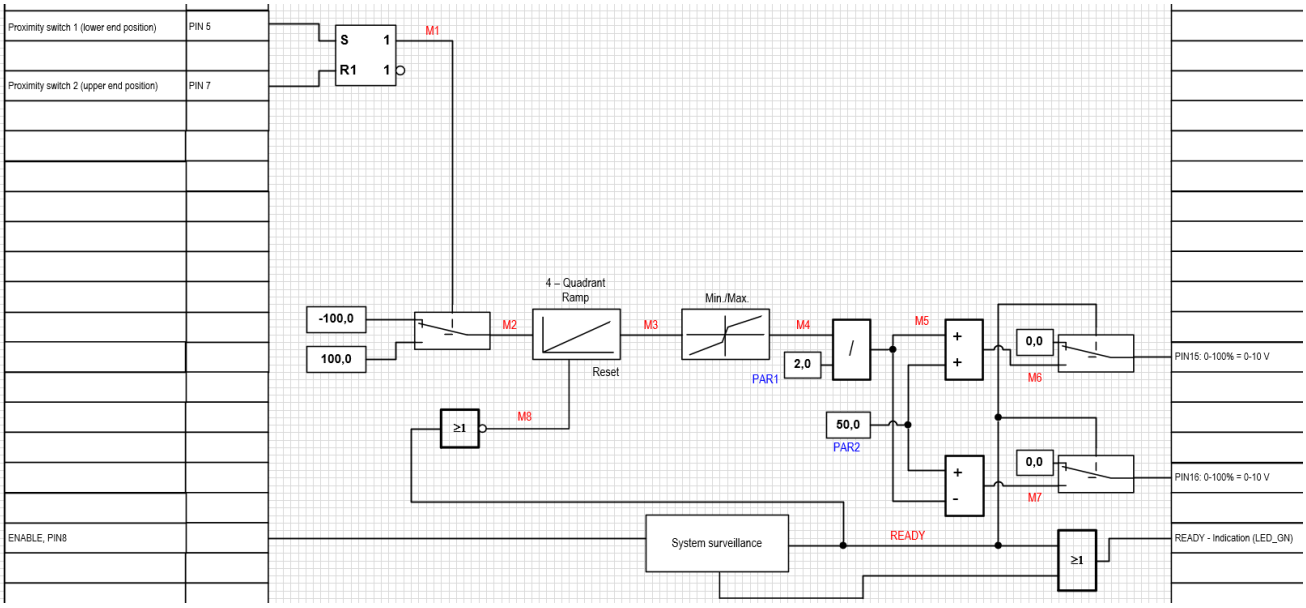
8.1.1 Task

A cylinder should move cyclically back and forth between two end positions. Exact positioning is not important. It should be possible to set the speed differently in the two directions of movement. Two variants are considered here, namely with the use of an analogue feedback signal (position sensor) or with the use of switching signals, e.g. from inductive proximity sensors.

8.1.2 Solution 1

Use of proximity sensors

8.1.2.1 Function plan and description



The proximity switches send a signal in good time before the end position is reached, by which the flip-flop M1 switches over. If M1 is set, a positive setpoint (+100 %) is generated, otherwise a negative one (M2). The following 4 - quadrant ramp is used to set acceleration and deceleration times so that smooth operation is possible. (M10)

The following Min./Max. function is used (if necessary) for overlap compensation and scaling. Ramp and Min./Max. are complex functions and do not require free parameters. The signal at M4 is already to be understood as a control signal in the range +/- 100%. Since PIN15 and PIN16 are unipolar voltage outputs, M6 and M7 are formed in such a way that they operate in the range of 0 ... 100% and PIN15 - PIN16 = M4. In the case of M4 = 0, M6 and M7 are each 50% corresponding to 5 V. If M4 increases, M6 increases and M7 decreases so that the voltage difference corresponds to the control signal. Via the following switches (SEL), both outputs are switched to 0V in the event of an error or if PIN8 is not enabled. If you want a different substitute value, you could also switch to free parameter values instead of "0".

Addition for bumpless switching on after an interruption: The negated "READY" signal (M8) is used to hold the ramp at "0,0" in the non-enabled state so that it starts from there when switching on.

8.1.2.2 Script

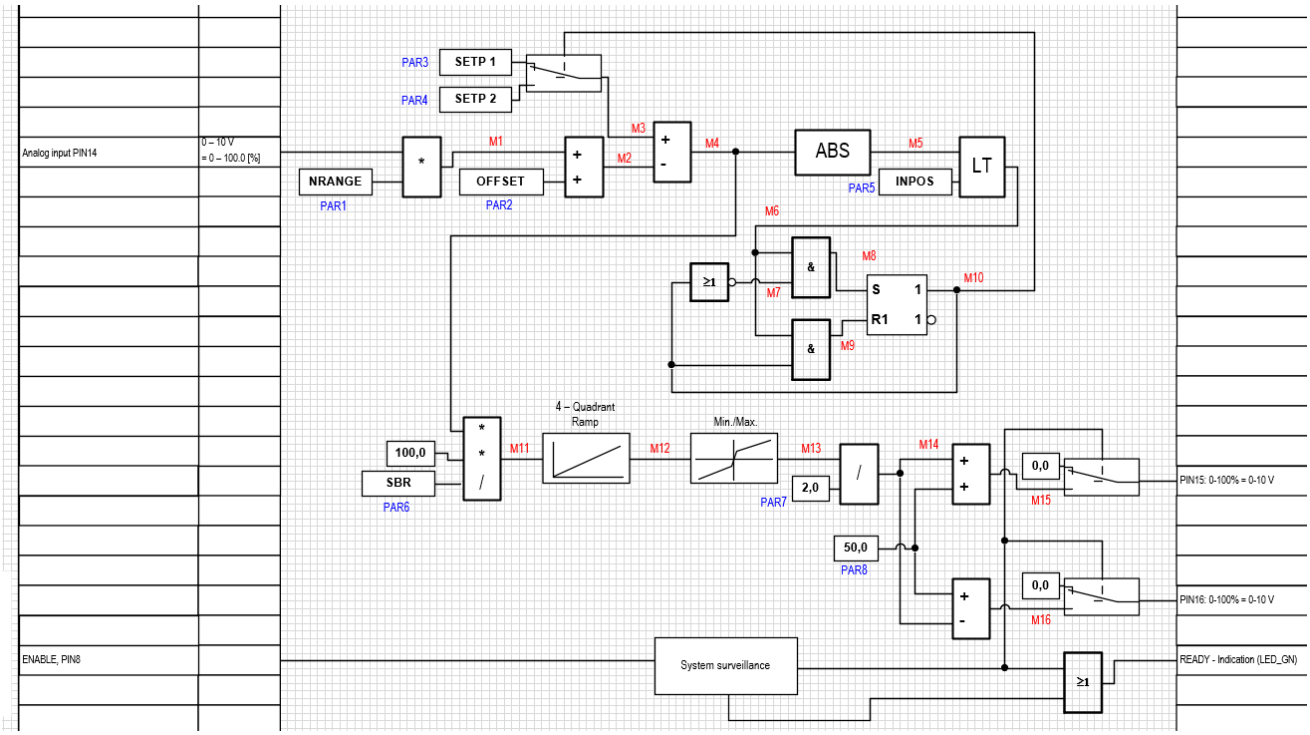
```
M1       RS       PIN5     PIN7     -
M2       SEL      M1       -100.0   100.0
M3       RAMP4Q   M2       M8       -
M4       MINMAX1  M3       -        -
M5       DMUL     M4       1.0      PAR1
M6       ADD      M5       PAR2     -
M7       SUB      PAR2     M5       -
M8       NOT      READY    -        -
PIN15    SEL      READY    0.0      M6
PIN16    SEL      READY    0.0      M7
LED_GN   OR       READY    ERFL     -
EN_SIG   DIR      PIN8     -        -
```

8.1.3 Solution 2

Use of an analog sensor.

Advantages: No overrun of the end position if set correctly. Defined deceleration process without bumps and clear stopping point.

8.1.3.1 Function plan and description



This task already requires a relatively extensive script.

The lower part from onwards to the ramp is identical to the previous example.

Starting from the input of the position sensor, the function can be described as follows: First, the signal is scaled to a physical value in mm by multiplication with the measuring length and addition of an offset value (M2). A difference is formed to a setpoint value (M3), which is selected via a switch. Let's first follow the generation of the switching signal (M10). The signal comes from a flip-flop that is switched via a pulse at M6. This pulse is generated when the absolute value of the control deviation (M5) becomes smaller than a set "INPOS" deviation, as the required approach to the switch-over point. Because the setpoint will have changed considerably in the next cycle, the deviation is only below the switching threshold for one processing step.

The control deviation is divided by an adjustable braking distance (PAR6), which determines the length of the braking process or the magnitude of the deceleration. If the control deviation is as large as the braking distance, the output (M10) will adopt the value 100 (%). If it is smaller, the valve is closed depending on the distance. If M11 is greater than 100%, a limitation to +/- 100% is carried out without further measures by passing it on to the complex function of the 4 - quadrant ramp. This is used to obtain a defined acceleration. It should be noted that the times in the 2nd and 4th quadrant should be set to the minimum value, because braking here is distance-dependent and not time-dependent.

8.1.3.2 Script

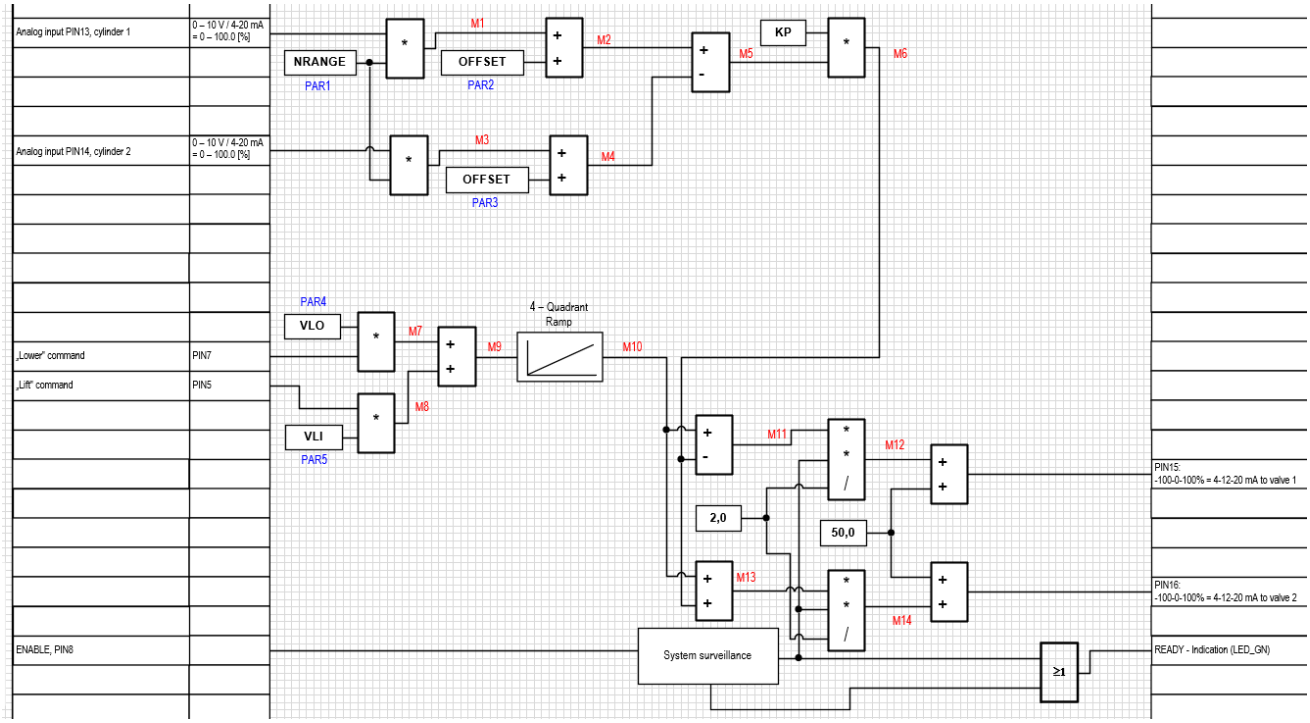
```
M1       MUL      PIN14    PAR1     -
M2       ADD      M1       PAR2     -
M3       SEL      M10      PAR3     PAR4
M4       SUB      M3       M2       -
M5       ABS      M4       -        -
M6       LT       M5       PAR5     -
M7       NOT      M10      -        -
M8       AND      M6       M7       -
M9       AND      M6       M10      -
M10      RS       M8       M9       -
M11      DMUL     M4       100.0    PAR6
M12      RAMP4Q   M11      -        -
M13      MINMAX1  M12      -        -
M14      DMUL     M13      1.0      PAR7
M15      ADD      M14      PAR8     -
M16      SUB      PAR8     M14      -
PIN15    SEL      READY    0.0      M14
PIN16    SEL      READY    0.0      M15
LED_GN   OR       READY    ERFL     -
EN_SIG   DIR      PIN8     -        -
```

8.2 Synchronisation of 2 cylinders (direction of movement controlled via switching inputs)

8.2.1 Task

In this example, a very simple synchronisation control of two cylinders is to be realised. Only switching signals (up/down) and the current position of both cylinders are processed as input signals. On the output side, the control of two proportional valves via outputs 4-12-20 mA is provided.

8.2.2 Solution



In the upper part of the plan, the input signals are first scaled, which appear in M2 and M4 in physical units, e.g. mm. A difference is formed for the control deviation M5. The proportional amplification KP determines the impact of the synchronisation component.

To form the actual control signal or its mean value (M10), the fact that digital signals can be used like numbers in a multiplication is exploited. The numerical value for VLO (PAR4) must be negative. A ramp ensures shock-free starting and braking. The synchronisation component is subtracted from the average control signal for cylinder 1 and added for cylinder 2. The bipolar control signals (M11 / 13) are then rescaled to the value range of the output signals. Again, a multiplication is used to introduce the READY signal of the error monitoring to switch off the movement.

8.2.3 Script

```
M1       MUL      PIN13    PAR1     -
M2       ADD      M1       PAR2     -
M3       MUL      PIN14    PAR1     -
M4       ADD      M3       PAR3     -
M5       SUB      M2       M4       -
M6       MUL      PAR6     M5       -
M7       MUL      PAR4     PIN7     -
M8       MUL      PIN5     PAR8     -
M9       ADD      M7       M8       -
M10      RAMP4Q   M9       -        -
M11      SUB      M10      M6       -
M12      DMUL     M11      READY    2.0
M13      ADD      M10      M6       -
M14      DMUL     M13      READY    2.0
PIN15    ADD      M12      50.0     -
PIN16    ADD      M14      50.0     -
LED_GN   OR       READY    ERFL     -
EN_SIG   DIR      PIN8     -        -
```

8.3 Splitting a setpoint signal between 2 variable displacement pumps

8.3.1 Task

Two pumps with different flow rates should operate in parallel. Externally, the decision is made whether both pumps or only one pump should run, and the module monitors whether both pumps should run based on the current setpoint signal. The two pumps have different efficiencies, so that pump 1 should take over the base load as far as possible and pump 2 should only deliver a larger flow rate from a certain demand.

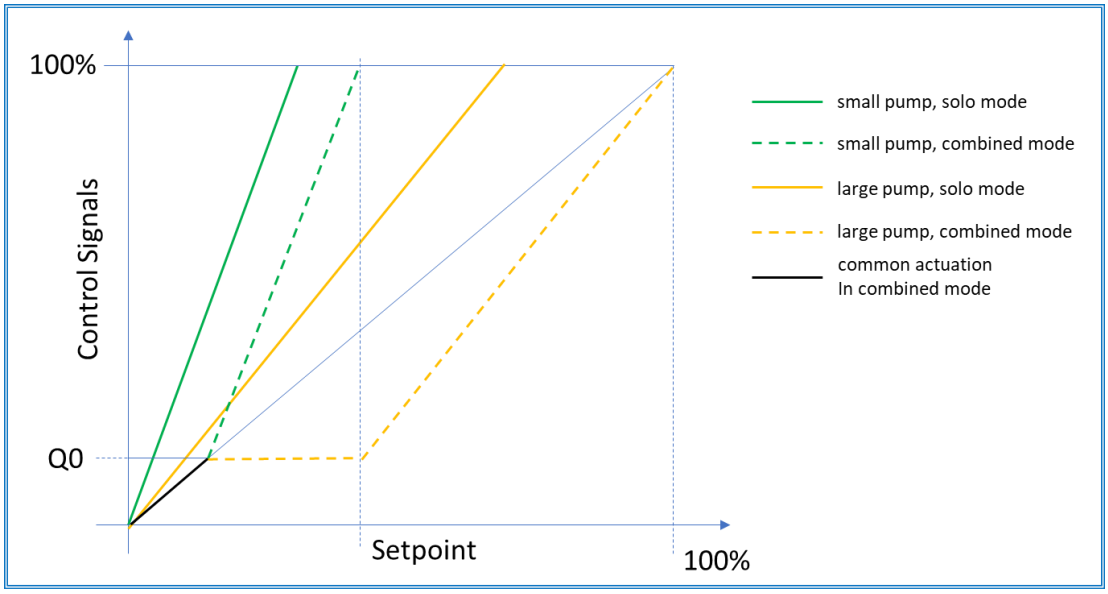
8.3.2 Concept

Regardless of whether only one pump is working or both pumps in combination, the setpoint input signal to the module should always result in the same total volume flow. In other words: The increase of the volume flow per % signal increase at the input should always be the same. If only one pump is running, the limit at which the flow rate cannot increase any further and the control signal to this pump reaches 100% will already be reached before the 100% setpoint is reached. To enable the user to react in time, a DO should signal this at 90% control signal so that a second pump can be switched on manually or automatically.

The diagram on the next page shows the control law to be implemented.

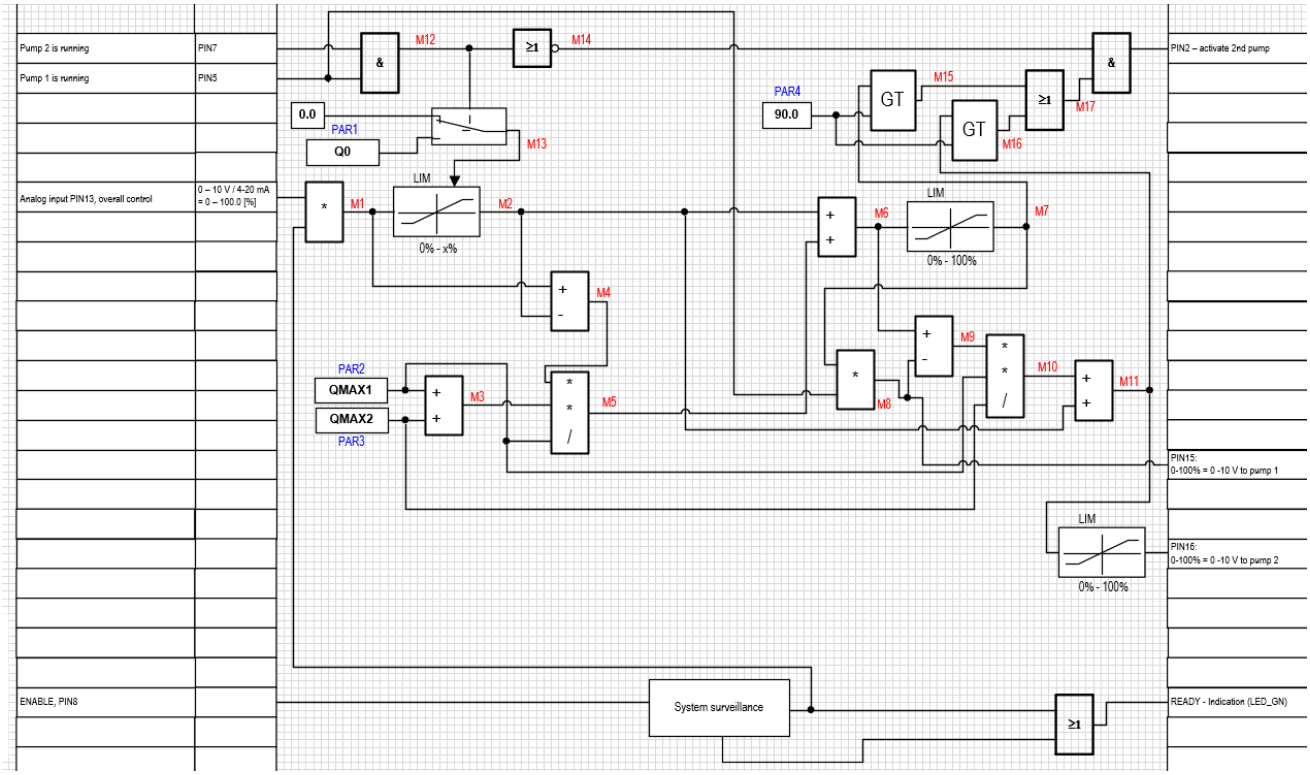
The solid lines show the output signals for single operation. Due to the different flow rates, the gradients are different and the delivery limit (100% control signal) is reached at different setpoints.

If both pumps are running, they should first be swung open in combination up to a minimum quantity (Q1), e.g. 10 %, and then, due to the better efficiency, the smaller pump 1 should first take over the further increase of the delivery quantity. Only when this is fully utilised does the larger pump then take over the further increase until the maximum point (100/100) is reached.



8.3.3 Solution

The minimum control Q0, the maximum flow rate of pump 1 and that of pump 2 are provided as parameters. Only a quotient is formed from the latter. The physical unit is therefore unimportant, it only has to be the same.



Function when both pumps are running:

The setpoint control should be the same as the input signal for both pumps up to the value "Q0" = PAR1. For this, M2 is formed as the input value limited to Q0.

For the first pump, a deviation M4 is calculated between the input signal and this component, so that M4 contains the part of the control that exceeds Q0.

This value is multiplied by the quotient of the total delivery quantity and the capacity of the first pump. In this way it is achieved that the increase in delivery rate continues to take place in this range with the same gradient. M6 thus contains the control of the first pump, which, however, must still be limited to 100%. As soon as this value is exceeded, the level at M9 rises above 0. This difference is multiplied by a second factor, which ensures a further continuous increase of the delivery rate via pump 2 with the same gradient. With 100% input signal, both outputs at PIN15 and 16 are also controlled with 100%. The limiting function according to M11 is only intended for the case that the input signal goes into the overdrive range and thus the output is to be controlled above 100%. Since the signal is limited in the module before it is passed on to the physical output, this function is not absolutely necessary.

If both pumps are not running, the value M13 is also kept at "0.0" via M12 (= 0). Thus, the minimum quantity at M2 is suppressed and M4 contains the full control, which takes place with the gradient according to the quantity ratio (as shown by the solid green line in the diagram).

If only pump 2 is running, the signal at M8 remains "0.0" and the control is diverted to this pump, again with the correct amplification.

In solo mode, a message is given via PIN2 if one of the two control signals exceeds 90% or the value in PAR4.

8.3.4 Script

```
M1      MUL     PIN13   READY   -
M2      LIM     M1      0.0     M13
M3      ADD     PAR2    PAR3    -
M4      SUB     M1      M2      -
M5      DMUL    M4      M3      PAR2
M6      ADD     M2      M5      -
M7      LIM     M6      0.0     100.0
M8      MUL     M7      PIN5    -
M9      SUB     M6      M8      -
M10     DMUL    M9      PAR2    PAR3
M11     ADD     M10     M2      -
M12     AND     PIN7    PIN5    -
M13     SEL     M12     0.0     PAR1
M14     NOT     M12     -       -
M15     GT      M8      PAR4    -
M16     GT      M11     PAR4    -
M17     OR      M15     M16     -
PIN15   DIR     M8      -       -
PIN16   LIM     M11     0.0     100.0
PIN2    AND     M14     M17     -
LED_GN  OR      READY   ERFL    -
EN_SIG  DIR     PIN8    -       -
```

8.4 Power limit control

8.4.1 Task

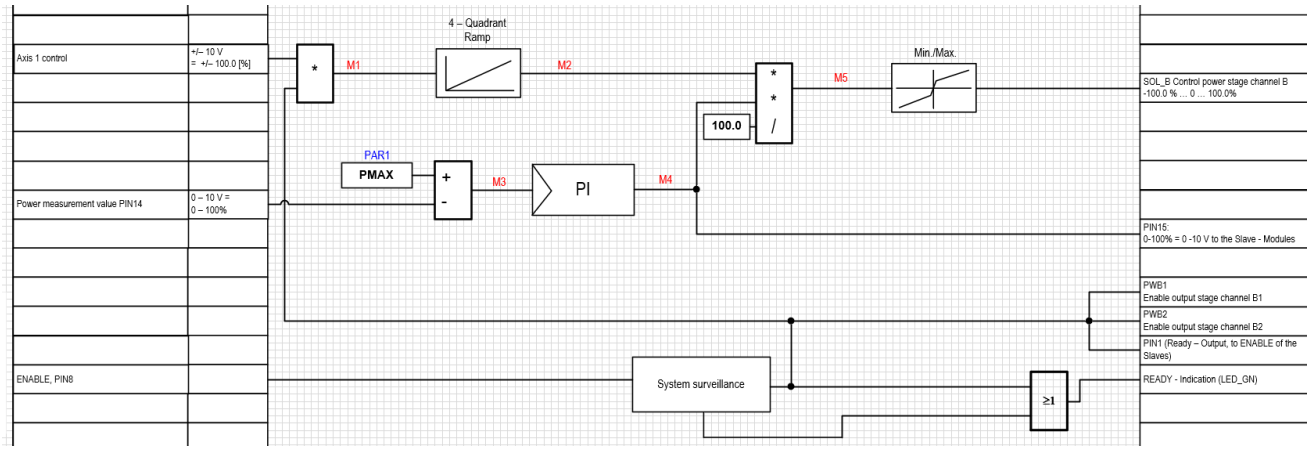
A pump with limited power (motor power smaller than the system peak load) feeds parallel consumers that are controlled via proportional valves. In this case, a PAM-195-P-S3 can be used for a single consumer. For several parallel consumers, a different solution is required, in which a limiting controller acts on all axes involved.

For this purpose, one unit is configured as the "master", i.e. it contains the controller and at the same time the actuation of one of the involved axes, the other units contain a reduced functionality, i.e. they receive the output signal of the controller from the master and limit the actuation of the connected consumer accordingly.

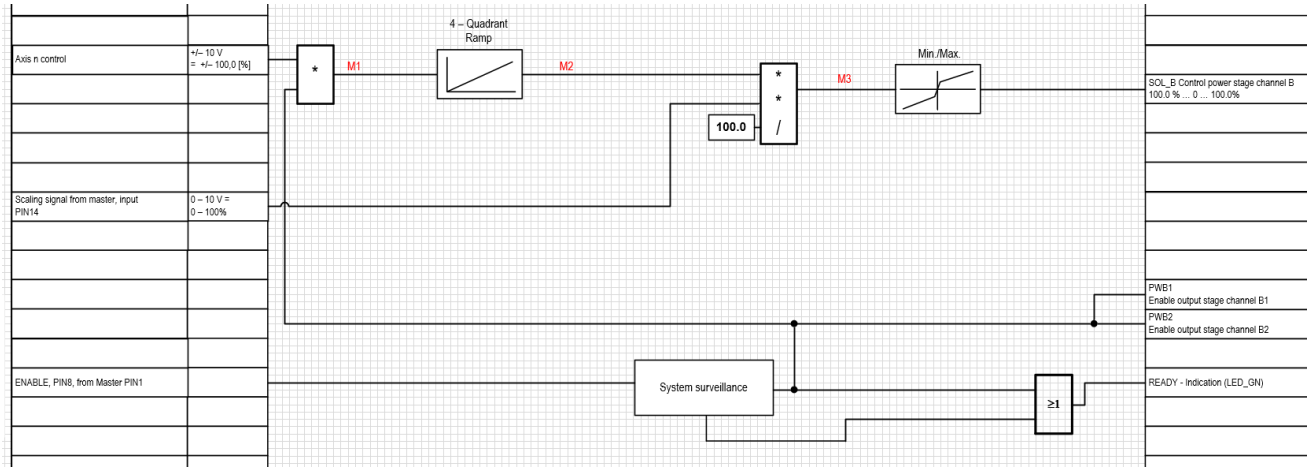
8.4.2 Solution

The DSG-112-P is used here, which has an integrated output stage. One valve with two solenoid coils is controlled per unit.

Part 1: Master



Part 2: Slave



The scripts for master and slave are structured similarly. The control setpoint of PIN9/10 (bipolar!) is first multiplied by the READY so that the ramp returns to "0" in the event of a switch-off. This is followed by the complex function "4 - quadrant ramp" specified in the module. The output of this function is scaled with the output of the power limit controller in both units; the connection is realised internally in the master, the slave receives this factor via an analogue input. The power controller normally outputs a signal of 100% and is limited upwards to this value. If its setpoint is exceeded, the output signal decreases and limits the control of all valves in the same proportion. It is important that the intervention takes place after the ramp, so that it shifts without delay and before the linearising min./max. function. The output signal of this function can be applied to the module-internal output stage; a positive signal acts on channel B1, a negative signal on B2.

8.4.3 Script

Part 1: Master

```
M1       MUL      PIN910   READY    -
M2       RAMP4Q   M1       -        -
M3       SUB      PAR1     PIN14    -
M4       PI_1     M3       50.0     -
M5       DMUL     M2       M4       100.0
PIN15    DIR      M4       -        -
SOL_B    MINMAX1  M5       -        -
PIN1     DIR      READY    -        -
LED_GN   OR       READY    ERFL     -
PWB1     DIR      READY    -        -
PWB2     DIR      READY    -        -
EN_SIG   DIR      PIN8     -        -
```

Additional note: The controller receives a fixed feedback value of 50% here. Together with the parameter setting PID1:YR = 5000 (50%), this means: Its output signal is limited to 50% - 50% = 0% to 50% + 50% = 100%. This setting is necessary for correct operation.

Part 2: Slave

```
M1       MUL      PIN910   READY    -
M2       RAMP4Q   M1       -        -
M3       DMUL     M2       PIN14    100.0
SOL_B    MINMAX1  M3       -        -
LED_GN   OR       READY    ERFL     -
PWB1     DIR      READY    -        -
PWB2     DIR      READY    -        -
EN_SIG   DIR      PIN8     -        -
```

8.5 Control of a bearing test stand

8.5.1 Task

Roller bearings are to be loaded with a pulsating force in a endurance test. The force profile is to be sinusoidal, with adjustable frequency and amplitude. The load is applied by a hydraulic cylinder.

A fast control valve with a zero-overlapped spool is used for control.

A load cell is used, which provides a sufficiently fast measuring signal.

The user specifies the static and the dynamic force (as amplitude).

8.5.2 Solution

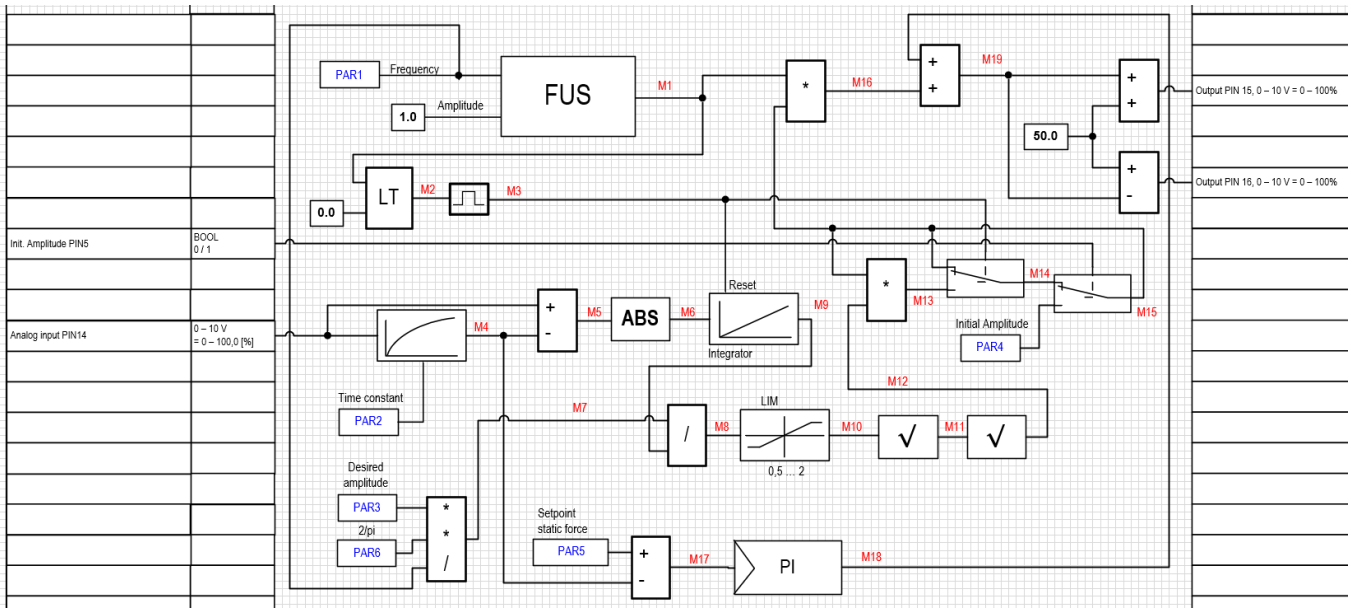
In this case, two variables are controlled, the static force and the dynamic force. The valve is controlled with a sinusoidal signal whose amplitude is adjusted so that the desired force amplitude results and which is provided with an offset by which the static force component is set.

To determine this component, the input signal of the measurement is first freed from the alternating component by a low-pass filter with a low cut-off frequency (M4). This is followed by a PI controller, which is adjusted relatively slow and which, with the signal M18, determines the average value of the control.

The alternating component (M5) is the difference between the total force minus the static component. The amount of this component is then integrated over a period. For this purpose, a pulse is generated via M2 and M3 as soon as the sine signal of the generator (M1) falls below zero. This takes place once per full wave.

The integral $\int_0^T |\sin(\omega \cdot t)| dt = \frac{4}{\omega} = \frac{2}{f \cdot \pi}$ would result at an amplitude of "1" at M9 when the pulse occurs.

Thus, the setpoint for this signal is equal to the desired amplitude times $\frac{2}{f \cdot \pi}$. In M8, the quotient of the setpoint and actual value is formed. When the desired amplitude is reached, this value becomes "1". M8 can also be seen as a correction factor with which the current control would have to be multiplied in order to obtain the desired amplitude. The controller can apply a correction once per pass (full wave), so that this is not too strong, a signal limitation is first applied (M10) and then a further reduction or approximation to the factor 1.0 is effected by taking the square root twice. The circuit consisting of M13/M14/M15 introduces the variable amplification into the signal flow. M15 can be used to switch to an initial amplitude value from outside, e.g. immediately after starting the system or after changing the operating point. This type of control is similar to the concept of the MR controller patented by W.E.St. and produces a fast and stable adjustment to the target value. It should be noted that, due to the circuitry, a correction only takes place at the moment when a complete sine wave has been integrated. Due to the exchanged processing sequence M8 / M9, the result of the integration from the previous step is used for the further calculation at this moment. The output of the integrator is reset to "0" in the cycle of the pulse.



8.5.3 Script

```
M1       FUS      PAR1     1.0      -
M2       LT       M1       0.0      -
M3       FP       M2       -        -
M4       PT1      PIN14    PAR2     -
M5       SUB      PIN14    M4       -
M6       ABS      M5       -        -
M7       DMUL     PAR3     PAR6     PAR1
M8       DMUL     M7       1.0      M9
M9       INTEG    M6       M3       -
M10      LIM      M8       PAR7     2.0
M11      SQRT     M10      -        -
M12      SQRT     M11      -        -
M13      MUL      M15      M12      -
M14      SEL      M15      M13      -
M15      SEL      PIN5     M14      PAR4
M16      MUL      M1       M15      -
M17      SUB      PAR5     M4       -
M18      PI_1     M17      -        -
M19      ADD      M16      M18      -
PIN15    ADD      M19      50.0     -
PIN16    SUB      M19      50.0     -
LED_GN   OR       READY    ERFL     -
EN_SIG   DIR      PIN8     -        -
```

Since the values 2/π and 0.5 are not available as fixed constants, the parameters PAR6 and PAR7 are used and must be set accordingly.

8.6 Frequency response measurement

8.6.1 Task

The frequency response of a controlled system with a proportional pressure valve should be determined. The valve has an integrated power stage and is controlled with a 0-10 V signal. The pressure is measured with a sensor, the signal of which is available to the device as an input signal for evaluation. The aim is to generate a Bode diagram for different signal amplitudes.

This example can be adapted so that frequency responses of any control loop can be measured. It is not limited to hydraulic systems. However, the limiting factor is the own sample time of 1 ms. As will be seen in section 8.6.3, meaningful measurements are possible up to a maximum frequency of about 100 Hz. However, this requires that above 10 Hz a compensation with the frequency response of the module is applied.

8.6.2 Solution

The script programme employed here is relatively elaborate and demonstrates the capabilities of script programming. The so-called correlation method is used. A sine test signal is applied to the circuit, the frequency of which is slowly increased (sweep). The "FUS" module is used for this purpose.

With sinusoidal control of a system, the real and imaginary parts of its frequency response can be determined by:

$$Re = \frac{4 \cdot \int_0^T y(t) \cdot \sin(\omega \cdot t) dt}{\pi \cdot \hat{u} \int_0^T \sin(\omega \cdot t) dt} = \frac{2 \cdot \int_0^T y(t) \cdot \sin(\omega \cdot t) dt}{T} \quad \text{and}$$

$$Im = -\frac{2 \cdot \int_0^T y(t) \cdot \cos(\omega \cdot t) dt}{T}$$

Where y(t) is the output signal of the circuit, i.e. the input signal of the module, and û is the amplitude of the control signal.

From this follows for the amplitude response:

$$|G(j\omega)| = \sqrt{Re^2 + Im^2}$$

And for the phase curve:

$$\varphi = \tan^{-1}\left(\frac{Im}{Re}\right)$$

These final calculations can also be carried out offline afterwards. In the module, the control of the test object by a sinusoidal signal of increasing frequency and the integration of the measured value or the continuous determination of the quantities "Re" and "Im" have to be implemented. If you want to carry out a measurement, you record these values and the frequency with WPC and then save them in a *.csv file for further processing. This procedure has the advantage that you do not have to record higher frequency information, but already pre-processed data, so that the recording speed is not a limiting factor.

The function plan of a suitable script is shown below.

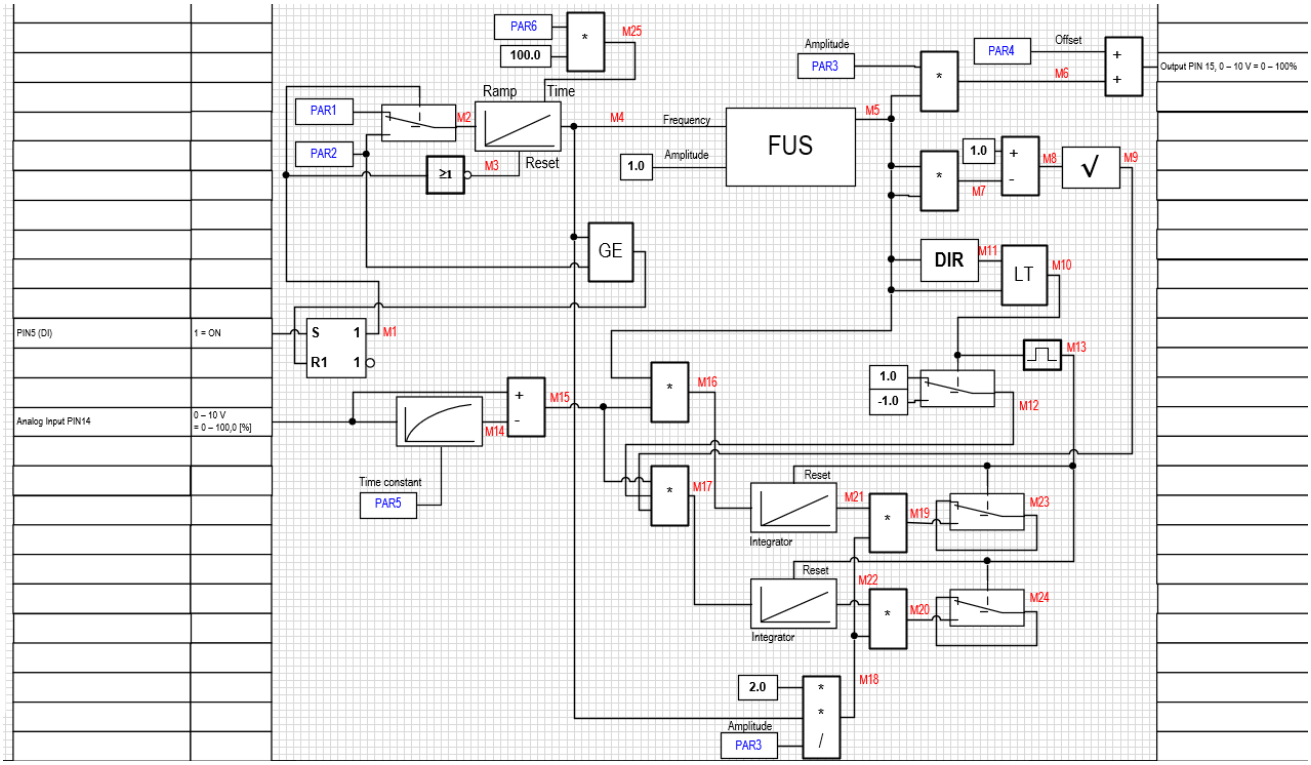
The upper part shows the generation of the sweep signal, realised with ramp and sine generator. A sweep is started by setting the input at PIN5. At the end, the frequency jumps back to the start value. The frequency limits are set by PAR1 (start) and PAR2 (end). The speed to be set with PAR6 should be very slow so that stationary conditions can be assumed.

At M5, a sinusoidal signal with amplitude "1" is present during the measurement. With PAR3 the amplitude at the test object is determined and with PAR4 the offset or operating point.

To form the integrand in the formula of the imaginary part, it is necessary to form the cosine in addition to the sinusoidal stimulation. This is realised here by using the Pythagorean theorem ($\cos^2(\alpha) + \sin^2(\alpha) = 1$) in M9, where M9 contains the magnitude of the cosine and the switch after M12 brings the sign. The cosine is then negative when the sine falls, because it is the derivative of the sine function. Once per full wave, the pulse M13 is triggered, which initiates the updating of the result.

M15 contains the response of the circuit, freed from the stationary component. For this purpose, the time constant of PT1 for averaging the input value must be set extremely high, so that M14 can be considered constant (> factor 100 higher than the period at the smallest measuring frequency). In the two integrators the two integrals given above are formed, after reaching the time T the result is multiplied by the remaining factors M18 and then stored in the selectors used as signal memory.

The user defines the signals M4 (frequency), M23 (real part) and M24 (imaginary part) in the WPC as script variables SC:A /:B /:C to be observed and records these values during the measurement.



8.6.3 Script

```
M1      RS      PIN5    M26     -
M2      SEL     M1      PAR1    PAR2
M3      NOT     M1      -       -
M4      RAMP    M2      M25     M3
M5      FUS     M4      1.0     -
M6      MUL     M5      PAR3    -
M7      MUL     M5      M5      -
M8      SUB     1.0     M7      -
M9      SQRT    M8      -       -
M10     LT      M11     M5      -
M11     DIR     M5      -       -
M12     SEL     M10     1.0     -1.0
M13     FP      M10     -       -
M14     PT1     PIN14   PAR5    -
M15     SUB     PIN14   M14     -
M16     MUL     M5      M15     -
M17     MUL     M5      M12     M9
M18     DMUL    2.0     M4      PAR3
M19     MUL     M21     M18     -
M20     MUL     M22     M18     -
M21     INTEG   M16     M13     -
M22     INTEG   M17     M13     -
M23     SEL     M13     M23     M19
M24     SEL     M13     M24     M20
M25     MUL     PAR6    100.0   -
M26     GE      M4      PAR2    -
PIN15   ADD     PAR4    M6      -
```

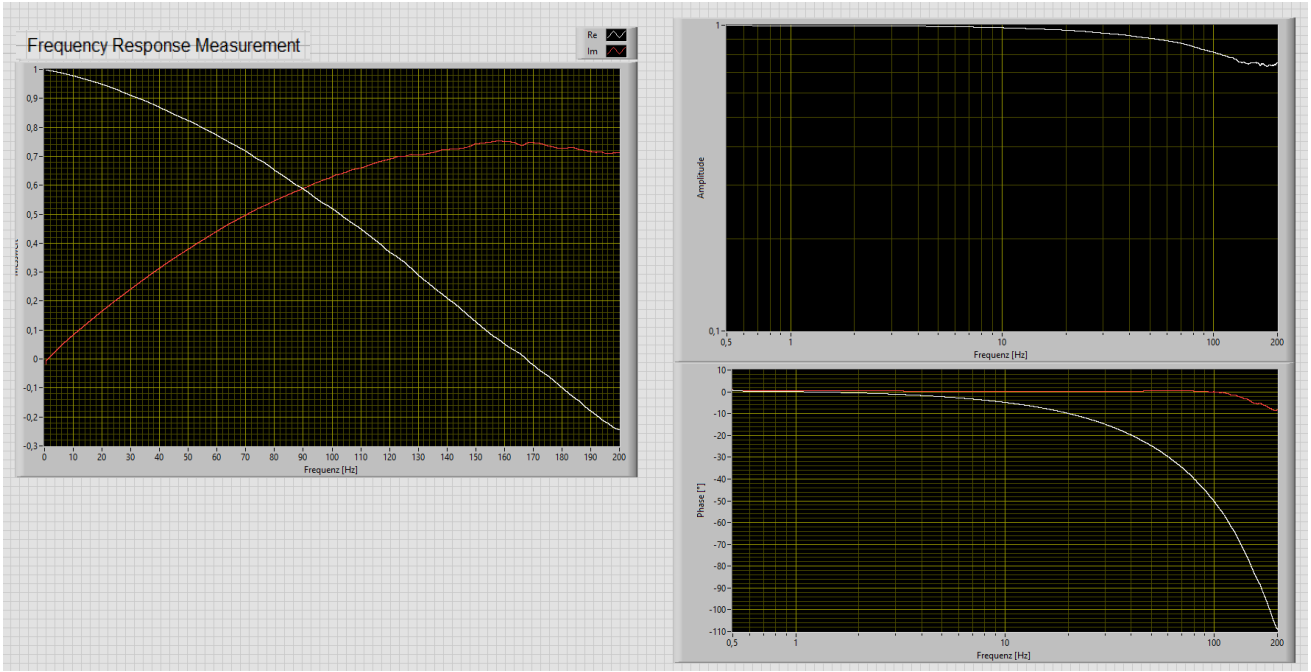
8.6.4 Results

For test purposes, this example was run on a module with directly connected output and input. Thus, the script measures the frequency response of the module periphery.

For evaluation, the above-mentioned signals were recorded with the oscilloscope of the WPC and saved in a *.csv file. As expected, the integral values fluctuated somewhat, so their curves were smoothed by filtering, then amplitude and phase response were calculated.

In this illustration you can see the filtered raw values on the left and the generated Bode - diagram on the right.

The raw value of the imaginary part was mirrored on the X-axis for better display.



As can be seen, a range of 0.5 - 200 Hz was measured.

The phase curve (white curve) shows a clear drop above 10 Hz.

If you change the X - axis to a linear scale, you will notice that it is a linear decay.

This effect is produced by the discrete-time sampling and the associated dead time. It was empirically determined that the behaviour can be described by a dead time element with a delay of 1.4 ms. If the phase response is compensated accordingly, the red curve is obtained, which shows no significant drop up to 100 Hz. The amplitude has dropped to approx. 0.8 up to that point.

Up to a frequency of 100 Hz it is therefore possible to carry out a meaningful measurement of circuits with this simple set-up.

On request, we will gladly provide the simple LabView VI used here for evaluation as a source or in compiled form.