Nokeval

made to measure

PM10 system

Manual

Document ID 7805 V1 20.4.2016

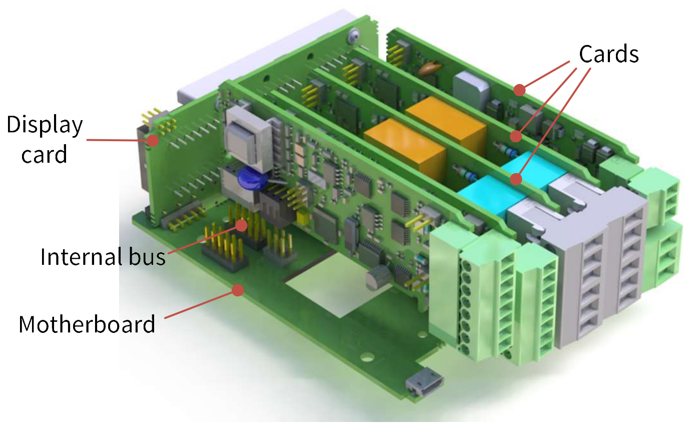
Table of contents

Introduction

PM10 is a modular panel meter system. It uses “cards” that can be used in several chassis types. Typical cards include a temperature sensor input card, an analog output card, and a relay card. The most basic chassis is a 96x48 mm panel meter “PM10A”. It is possible to construct various display and signal processing systems for industrial use.

The chassis consists of:

- Motherboard (or several motherboards) to hold the cards.
- Internal bus on the motherboard(s) to connect the cards. It delivers power and data.
- Display for viewing the readings and for configuration.
- Enclosure.



The cards are printed circuit boards without an enclosure of their own. The desired types of cards are inserted on the motherboard.

The manuals

This manual covers the PM10 principles and functions common to many chassis and card types.

Each chassis (like PM10A or PM20A) has a manual of its own, instructing assembling a device, mounting it, and using the display.

Each card has a manual of its own, e.g. the PM10UNI2A input card.

Naming of the products

The name of the final product begins with the chassis name e.g. “PM10A-“. Then each card is listed separated by dashes. PM10 is removed from the card names. An empty card slot is denoted with 0 (zero).

E.g. a product consisting of PM10A chassis filled with cards PM10UNI2A in slots A and B, PM10OUT2A in slot C, and PM10POW24SA in slot E is called PM10A-UNI2A-UNI2A-OUT2A-0-POW24SA.

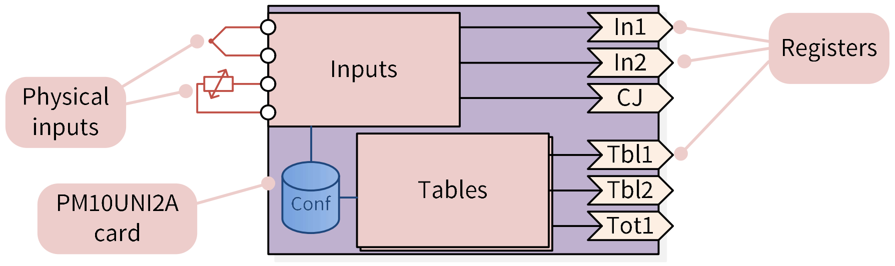
General principles

Blocks

Each PM10 card is an almost individual device. The card performs its tasks, not interacting with the other cards unless explicitly configured to do so. Each card is further divided into separate blocks, like analog input, mathematical function, table linearizer, etc. Even these blocks do not interact unless configured. Thanks to this, PM10 is configured step by step: configure one block, check it is working. Configure another block, link it to the first block, and check it is working.

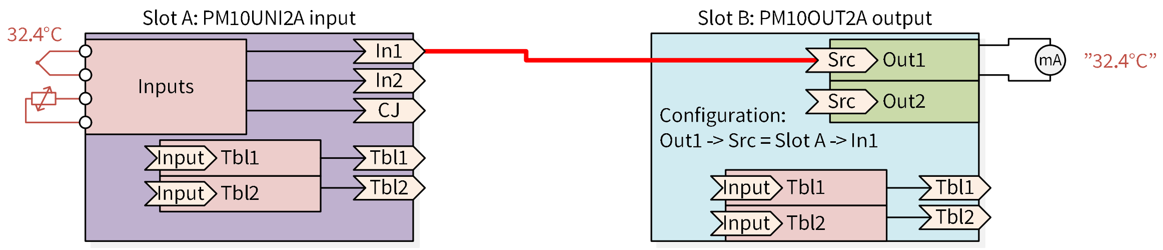
Registers and sources

The PM10UNI2A card measures two physical analog inputs and provides the measurement results for the other functions in so called registers. The registers are the logical outputs, or products, of a card.

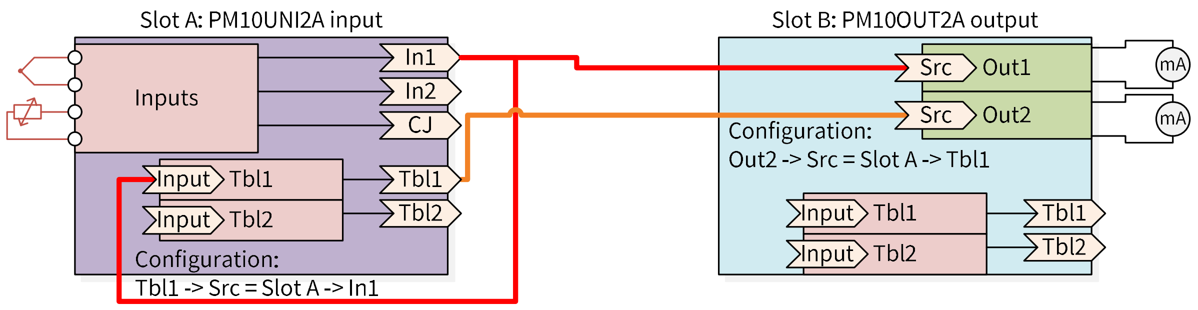


If you plug an analog output card in a PM10 device, the output will not work as it doesn't know what it should do. In addition to selecting the traditional signal type and the scaling, you must define, what the output is following or representing – the first analog input, the second, or something very different. The same applies for any kind of output, including the display – the source register must be explicitly configured.

The source settings consist of selecting the slot of the producing card, then the register within that card. For example, to make an analog output on slot B to follow the first input of the slot A card, go to the output settings, set the Src (source) setting by selecting first Slot A and then In1.



A register can be used by many blocks, and in the same or different card. As an example, we want to have another analog output that is linearized with a table:



When a block uses a register of the same card as its input (like slot A Tbl1 uses slot A In1 in the example above), there are two ways to refer to the register:

- Slot A -> In1
- This -> In1

The word This means the card itself. These ways differ, when the card is detached and moved to another slot. If it was configured “Slot A”, it will still refer to slot A. However if it was configured “This”, it will always use the registers of that physical card. Generally it is advisable to use This when possible.

Utilizing the functions

A function means an easy to use block that can take one or two inputs from any register and perform a calculation, comparison, delaying, or similar operation. The number of function blocks per card, including the display, varies card by card. The results are provided in registers named Func1, Func2, etc.

To use a function, navigate to the Funcs submenu in the configuration menu, and select an unused function in the submenu (named Func1, Func2, etc.).

Configure it:

Func	The function to perform. The functions are listed below. Every card type doesn't implement all the functions.
Input1	The first input. Select any register on any card, or None. If the first input should be constant, select Input1 = None, and use the Const setting.
Input2	The second input. If the second input should be constant, select Input2 = None, and use the Const setting.
Const	Constant input. If Input1 or Input2 is set to None, the value entered here will be used as that input. This is always a floating point number. This setting is hidden if neither Input1 nor Input2 is None.
Reset	Select the register that can reset this function block. The actual operation depends on the selected function. This setting is hidden if not available.
Set	Select the register that can activate this function block. The actual operation depends on the selected function. This setting is hidden if not available.
Value	A value related to the function, e.g. a tare value. Mainly for checking the operation, but may be set manually in some cases.
Output	The function result for checking the operation. Can't be set manually.

The functions listed

Off
Off does nothing. Set any unused function blocks off in order to save the processing power.

Pass
Passes the Input1 value through. May be used for some special purposes, e.g. to feed a constant to some other block.

Hold
When the Set input is off, the function output will follow the register selected in Input1. When the Set input is on, the output is held constant.

Lopass
The output follows the Input1 filtered by a first-degree lowpass filter. The time constant in seconds is defined by Input2, or if that is set to None, then by the Const setting. If the Reset input is on, the filter is bypassed.

Tare
The output of the function is the Input1 subtracted by the tare value. If the Set input is on, the current input value (Input1) is stored as a new tare value, making the output to go to zero. If the Reset input is on, the tare value is reset to zero, making the output to follow the input. The reset input may be configured to None, if a reset is not needed.

The tare value can be viewed at the Value setting, and even manually edited. The tare value is stored in the EEPROM memory and thus retained at power-off, with some limitations.

Peak, Valley
The output remembers the highest (Peak, max hold) or lowest (Valley) Input1 value so far. If the Reset input is on, the output follows the input, resetting the function. The value is retained at power-off, with some limitations.

Sum
The sum of two inputs. The other input may be constant.

Diff
Difference Input1-Input2. The other input may be constant.

Avg
The average of Input1 and Input2. If any of the inputs is an error (e.g. sensor fault), the function output is too.

AvgPrio
The average of Input1 and Input2. If one of the inputs is an error, the function output will follow the other input. If both inputs are errors, then the output is an error too.

An example: Two temperature sensors as inputs. If one of them fails, the function output will still give a reading, although not an average anymore.

Prio
As far as the Input1 is not an error, the function output follows it. If the Input1 is an error, the output will follow Input2.

An example: One temperature sensor for the actual measurement, another sensor as a backup.

Mult
The function output is Input1 multiplied by Input2.

Div
The function output is Input1 divided by Input2.

Pow
The function output is Input1 raised to the power of Input2. If Input1 is negative, its sign is removed before the operation and restored afterwards.

An example: A square root of a pressure sensor: Input1 = Slot A / In1. Input2 = None. Const = 0.5.

Min
The lesser of the two inputs. Can be also used as a boolean “AND” function.

Max
The greater of the two inputs, or “OR” function.

Equals
The output is 1 (on) if the two inputs are equal, 0 otherwise.

Less
The output is 1 (on) if the Input1 is lesser than Input2, 0 otherwise.

An example: a simple "alarm" or other comparison. To remove too fast variations, connect another block in series, selecting Delay function there.

Greater
The output is 1 (on) if the Input1 is greater than Input2, 0 otherwise.

Mux
If the Set input is off, the output follows the Input1, otherwise the Input2.

Latch
If the Input1 or Set are on, the output will go on, and stay there. If Reset is on, the output will go off. The reset has higher priority. The value is retained at power-off, with some limitations.

An example: a latched alarm.

Delay
If the Input1 is unchanged for a time defined by Input2 (in seconds, typically a constant), the output follows the input. The maximum delay time is 6553 seconds.

An example: a delay in an alarm, to prevent short deflections from triggering an alarm.

An example: a delay in the front panel (or external) pushbutton to prevent accidental operations. Input1=Master / Left. Input2=None. Const = 2 (seconds).

PulseA
The output follows the Input1, except once it changes, it keeps the state at least the time defined by Input2 (in seconds, typically a constant). The maximum time is 6553 seconds.

An example: an alarm that must trigger quickly, but repeated switching operations must be prevented.

PulseB
When the Input1 changes from off to on, the output will be on for the time defined by Input2 (in seconds) and then off for the same time.

An example: a pulse to drive a motor.

TotDiv
When the Input1 value has increased by Input2, the output gives a short pulse. The Input1 is typically a totalizer output, and Input2 is a constant. The output generates a pulse when the specified amount of substance has flown.

To shape the short output pulse for a relay or a digital output, use PulseB function in series to define the final output pulse width. For maximal pulse width accuracy, it is best to place the PulseB function on the relay card.

Suppress
Initially the output follows the Input1 value. However if the Set input is on even momentarily, the output will be forced off, i.e. suppressed. When the Input1 is off (or Reset is on), the output will again follow Input1. Typical use is manually suppressing an alarm siren.

IsFault
If Input1 has a NaN (not-a-number) or other fault indicating value, the output will be on.

Using the totalizers

A totalizer increases its output on the speed proportional to the input. It can be used to calculate the total amount of substance flown based on a flow measurement.

The PM10 cards may implement any number of independent totalizers. E.g. the PM10ADISP display card has two totalizers.

The totalizer uses a 64-bit floating point number for the summing to avoid rounding errors, but the public output value is rounded to a 32-bit floating point number.

If the totalizer input is an error (e.g. sensor fault), the totalizer is halted for 15 seconds. If the problem persists, the totalizer value is considered lost and the totalizer output will indicate an error until the totalizer is reset.

The totalizer value is retained at power-off, but with some limitations. The value will be saved to a non-volatile memory (EEPROM) every five minutes. In addition, the value is stored when a loss of power supply is detected, but the device must be powered on for at least five minutes before that.

To set up a totalizer

1. Navigate to one of the Tot menus under the Totalizers menu in the configuration menu.
2. Configure the settings:

Input	Select the flow or other input source. Typically an analog input that is scaled appropriately, e.g. flow m^3/min.
Timebase	Select the time base according to the output unit: • x/s: select 1 • x/min: select 60 • x/h: select 3600 This is actually just a divider for the input value.
Dead	Set the smallest significant flow (in the same units as the input). The lesser values are rounded to zero, stopping the totalization. Set negative, e.g. -1, if a dead zone is not desired.
Rollover	Set the value at which the totalizer output is rolled over to zero. Typical value is 1000000 (one million). To prevent any rollover, set to 0 or a negative value.
Hold	When this input is active, the totalizer is halted. The output remains unchanged.
Reset	When this input is active, the totalizer is reset to 0.
Output	The totalizer output for checking the operation. It is also possible to alter the value, e.g. to fix the value after it has been corrupted. If the time base is correctly selected, the unit is the same as the input unit with the time part removed, e.g. m^3.

The output registers

The totalizer outputs two registers, Tot1 and TotTime1 (or any other number). Tot1 is the totalizer output.

TotTime1 is the running time of the totalizer since the last reset in seconds (as a 32-bit integer). The Hold input will pause the time, but Dead will not. The time is not affected by the Rollover setting.

Configure one of the display pages to display the totalizer output: Display ▶ Page1 ▶ Src = Reg ▶ This ▶ Tot1. You may configure another page to display the flow input.

Resetting the totalizer

The totalizer should have means to reset it. The reset input may be:

- An external switch.
- One of the front panel keys (registers Left, Down, Up, Right, Fire, Left1…).
- An event setpoint in some display page.

Alternatively the totalizer may be reset by manually setting a value 0 in the Output item in the configuration menu.

Non-zero initial value / downwards

If you need a totalizer that will reset to a non-zero value, e.g. to “load” a batch that then decreases towards zero, use one of the function blocks:

1. Select Func = Diff.
2. Set Input1 = None to make it constant. (Or it might come from a setpoint or other variable source).
3. Set Const = the initial value (batch), e.g. 200.
4. Set Input2 = Tot1 (or whichever totalizer you use).

This function will calculate Func1 = 200 – Tot1, being 200 when the totalizer is reset and then decreasing.

Getting a divider pulse output

Configure one of the functions (under Functions submenu of the configuration menu) on the same card as the totalizer itself:

Func	TotDiv (not available on all the cards)
Input1	Reg ▶ This ▶ Tot1 (or Tot2)
Input2	None (for constant)
Const	Divider value
Reset	None

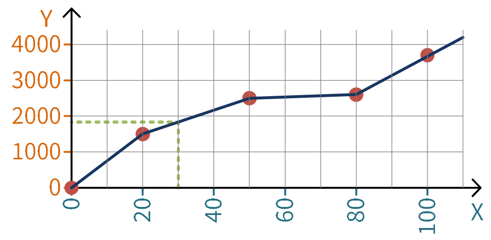
Then configure another function to shape the pulse width. This function should be placed on the card where the physical output is if possible, for maximal pulse width accuracy.

Func	PulseB (not working in firmware version V0.3)
Input1	Select the TotDiv function output
Input2	None (for constant)
Const	Pulse width in seconds
Reset	None

Finally configure a relay or a digital output to follow this function.

Using the tables

A table consists of a set of configurable X and Y value pairs, or points. When the table input corresponds to one of the X values, the table output will correspond to the associated Y value. Between the points, linear interpolation is used. Outside the points, linear extrapolation is used using the two nearest points. The table output is always a floating-point number.



An example where the points (X,Y) are (0,0), (20,1500), (50,2500), (80,2600), and (100,3700) as in the graph above. When the table input is 30, the output will be 1833. When the table input is 110, the output is 4250.

The number of tables varies according to the card type; some cards may contain none. The same applies to the maximum number of table points.

To use a table, navigate to the Tables menu under the configuration menu and select one of the Tbl menus. Configure:

Input	Select the input source for the table; any register on any card.
Pts	Select the number of points, or X/Y pairs.
X1	The first point X (input) value.
Y1	The output value when the input corresponds to X1.
X2…	The next points. When there are more than two points, the X values must be in ascending order (X1<X2, X2<X3, etc).
Y2…	The Y values do not have to be ascending.
Output	Check the output value.

After configuring, the table output is available to all cards in Tbl1, Tbl2 etc. registers.

It is possible to "teach" the points:

1. Apply a real input to the device.
2. Give a Lock command for the X1 setting. In Mekuwin, click the L button. Using the PM10ADISP/PM20ADISP front panel for configuring, navigate to the X1 setting but do not edit the value (do have X1 in the display), long-press ✱ and select Lock.
3. Enter the corresponding desired output in Y1.
4. Repeat the procedure for the next points, keeping in mind that the X values must be ascending.

Elo programming

Elo is a simple scripting language developed by Nokeval. Small programs can be used to extend the capabilities offered by the other function blocks. Elo can do basic calculation, conditional execution and timing. The program can handle both floating point values and binary values. The Elo program is interpreted, not compiled.

The program can be entered with Mekuwin software. It is entered in the *Elo* item in the program submenu of the configuration menu. The maximum length is typically 320 characters, depending on the card, each line feed counted as one character.

Each card runs an independent Elo program – if the card implements Elo.

Input, variables, and output

Elo program can read all the registers of the card it is running on. The registers are referred simply by their names. The case of the register name is important: "f1" does not do the job if the correct name is F1.

To access the registers (outputs) of the other cards, use an agent on the same card. Set the agent Src to the desired register on any card, and leave Dest = None. Then, in the Elo program, use A1 to read the agent 1 value, etc.

There are typically twelve floating point registers on each card that are intended for the Elo program: F1 to F12. The program may write and read the F registers freely. These registers are readable by every card, functioning both as temporary variables and final outputs. E.g. an analog output can be configured to follow any F register on any card.

The F registers are initialized to 0 at power-up.

A special register Intv contains the time interval between the beginning of the previous and the current execution of the Elo program in seconds. It can be used to count time. Writing F1+=Intv will make F1 to count seconds (with a resolution of dozens of milliseconds).

Another special register First will have a value 1 on the first execution of the program, and 0 afterwards. It can be used to do some initializations.

It is important to remember that a register controlled by another block (e.g. the In1 register on an analog input card) may be updated any time during the Elo execution. Reading the register value may yield a different value than at the previous time. This applies to the register obtained via agents too. If this is an issue, take a working copy of the register to one of the F registers.

On the other hand, it is safe to write intermediate results to the F registers. They are double-buffered: The other blocks will see a buffered copy of the F registers, that is updated only when the Elo program has been executed.

Not-a-number

Not-a-number or NaN is a special value for a floating point register to indicate an erroneous value. The Elo program can set any F register to NaN, and test any register against NaN.

The NaN value is used by the other blocks too: the analog input In1 register will contain NaN in case of a sensor fault. If an analog output is configured to follow a register that has a NaN value, the output will indicate fault as defined by its settings.

Any calculation involving a NaN will yield NaN, e.g. NaN/2 will result in NaN. A comparison against NaN yields an undetermined result.

Trigger

The *Trigger* setting defines when the program execution is launched. One register (on the same card) can be selected as a trigger. Whenever the value of that register is updated, the program will execute. If the program is mainly processing an input reading, a natural choice as a trigger is the In1 register.

If the trigger register is not updated for a long time, the program will be executed approximately 1 second after the previous execution finished.

It cannot be guaranteed that the program execution is finished before the trigger is tripped again. In this case the program will re-execute almost immediately when the previous execution is finished.

If the Trigger setting is set to None, the program is re-executed periodically, e.g. 50 milliseconds after the end of the previous execution.

Program structure

The program consists of lines. Every line has one simple command. The command can change a register or cause a conditional or unconditional jump inside the program.

The program may contain short loops, but eventually it must exit. The program exits when reaching the end of it, or by jumping out of it, e.g. with ?99. The program will soon be re-executed from the beginning.

The program execution is limited to 200 operations (one line counted as one operation) in order to prevent total lock-up in a case of e.g. an eternal loop.

Math commands

Each row can contain only one command, i.e. it is not allowed to write dest=src1+src2+src3.

dest=src	Copies a value from src to dest. Src can be a register reference, an agent, or a constant. E.g. F1=3.14 will place a value 3.14 in register F1.
dest=NaN	Puts a not-a-number value to dest.
dest=src1+src2	Sums src1 and src2 value and places the result in dest register. E.g. F1=In+10 will add register In contents to a value of 10 and place the result in register F1.
dest=src1-src2	Subtracts.
dest=src1*src2	Multiplies.
dest=src1/src2	Divides.
dest=src1src2**	Src1 raised to power of src2.
dest=src10.5**	Square root.
dest=src1&src2	Bitwise AND. If applied to a floating-point value, it is first converted to an 8-bit integer.
dest=src1|src2	Bitwise OR.

dest=src1^src2	Exclusive OR.
dest+=src	Sums src and dest and places the result in dest. Exactly same as writing dest=dest+src.
destX=src	The same for other operators, e.g. F1*=10.

Jumps and conditional jumps

?lines	Jumps given number of lines forward (+) or backward (-). E.g. ?-2 will execute the command that is two lines above this line. ?2 will skip the next line. ?0 creates an eternal loop. Jumping outside of the program is allowed: ?99 will exit the program.
x==y?lines	Jumps given number of lines if x equals y. E.g. F2==0?3
x!=y?lines	Jumps if x is not equal to y.
x<y?lines	Jumps if x is less than y.
x<=y?lines	Jumps if x is less or equal to y.
x>=y?lines	Jumps if x is greater or equal to y.
x>y?lines	Jumps if x is greater than y.
x==NaN?lines	Jumps if x is not-a-number
x!=NaN?lines	Jumps if x is not not-a-number

References

1.23	Decimal constant. Allowed characters plus, minus, point, digits 0…9.
F1	Register F1 of this card.
A1	Agent 1 of this card.
@0	The first register of this card.
@F1	Register defined by register F1 contents – indirect reference.

Errors

Whenever an error occurs in the execution, the card will report an error. The error number and the error line can be checked in the configuration menu items Error and ErrLine correspondingly. The error numbers are:

0	No errors.
1	Too long operand or operator.
2	Unknown operator.
3	Execution exceeds 200 operations.
4	Failed to write a register.
5	Illegal register reference. Please remember that the references are case-sensitive.

Examples

Prevent negative values

Read an input and copy it to a F register. If the value is negative, replace it by 0. The result is available in the F1 register.

F1=In1	Copy the input
F1>=0?2	If the value is acceptable, skip the next line
F1=0	Otherwise reset the value.

One minute average

Collect an average for one minute of the ambient light sensor, then publish it in F1, and start collecting another average.

F2+=Amblight	Sum up the input on every execution
F3+=1	Count the number of samples
F4+=Intv	Measure the cumulated time
F4<60?99	If a minute has not elapsed, jump out. The following lines are executed once a minute.
F1=F2/F3	The average is the sum divided by count. The other blocks may read it from F1.
F4=0	Reset the timer for the next shot.

Polynomial

Calculates y = 30 x^3 – 20 x^2 + 10 x – 5. Input: In1. Output: F1.

F2=In1	Take a working copy, because In1 register can change anytime
F1=30	Highest coefficient
F1*=F2	Multiplied by x
F1+=-20	Second coefficient
F1*=F2	Etc
F1+=10	
F1*=F2	
F1+=-5	

Dynamic settings with agents

The configuration menus of the cards contain dozens of settings that define the behavior of the device. They are usually changed very seldom, mainly when setting up a new device. On the other hand, the registers contain continuously varying values, e.g. the input reading, the Elo program variables, and the table output.

An agent is a bridge between these. Once configured, it reads one register and copies its value to a certain configuration setting. Each agent has two configuration settings in its menu under the Agents branch of the configuration menu. ***Src*** defines the register to read. It can read from any card. ***Dest*** selects the configuration setting to write to. It can access the same card only. To disable an agent, set either or both of the settings to None.

It is not possible to write any configuration setting. The agent can do some type conversions, but not all, e.g. it cannot process strings (text). As the configuration settings are not intended to be continuously varied, there may be some side effects.

The agents serve another purpose: they obtain register values from the other cards to be used in the Elo program. In these cases, set Dest=None.

An example: User-controllable display color on PM10ADISP

1. Define a setpoint on the Master card: Conf ▶ Setpoints ▶ Setp1: Type=ListIndex, List=Red<LF>Yelw<LF>Grn (replace <LF> with a real linefeed). The setpoint register Setp1 will have a value of 0 for red, 1 for yellow, and 2 for green selection.
2. Make the setpoint accessible in one of the display pages: Conf ▶ Display ▶ Page1: Tag=C (or whatever), Src=Reg ▶ This ▶ Setp1.
3. Use a table to convert the list index (0=red, 1=yellow etc) to real color setting values. Conf ▶ Tables ▶ Table1: Input=Reg ▶ This ▶ Setp1, Pts=3, X1=0, Y1=0, X2=1, Y2=60 (yellow), X3=2, Y3=100.
4. Hook an agent to copy the table output to the color setting: Conf ▶ Agents ▶ Count=1, Conf ▶ Agents ▶ 1 ▶ Src = Reg ▶ This ▶ Table1, Dest = This ▶ Conf ▶ Display ▶ Colors ▶ Normal ▶ Value ▶ Hue
5. Exit the menus. Navigate to the configured page. Push ✱. The color selection should blink. Try different values.

Manufacturer

Nokeval Oy
Rounionkatu 107
FI-37150 Nokia
Finland

Tel +358 3 342 4800 (Mo-Fr 8:30-16:00 EET)
WWW http://www.nokeval.com/
Email sales@nokeval.com,
support@nokeval.com