TECHNICAL NOTE

Maple Model(s)	Title	TN7013
Maple PLCs HMC3000 HMC7000 HMC2000 HMC4000 MAPware-7000 2.35, 2.36	PID Loop Configuration	

P/N: 0907-7013
Rev. 02 Date: 11/03/2022

Summary

This tech note explains the PID (Proportional Integral Derivative) instructions available in MAPware-7000 for Maple HMI + PLC and PLC products. The PID function is used to control process variables such as temperature, pressure, liquid level, or flow rate. Because the controlled process characteristics vary widely, this guide explains basic/general methods to use the PID function. For precise PID tuning, knowledge of the process being controlled is required.

This tech note contains the following 3 sections:

Section 1: PID Basics

This section, starting on the next page, provides basic information on PID control and PID parameters. It will cover how each piece of the PID equation affects loop tuning.

Section 2: Native Ladder Implementation (MLC, HMC3000 and 7000 ONLY)

This section, starting on page 5, covers the specifics of the PID5 instruction when using Native Ladder Programming Language in MAPware-7000.

Section 3: IEC Mode Implementation (Includes HMC 2000 and 4000)

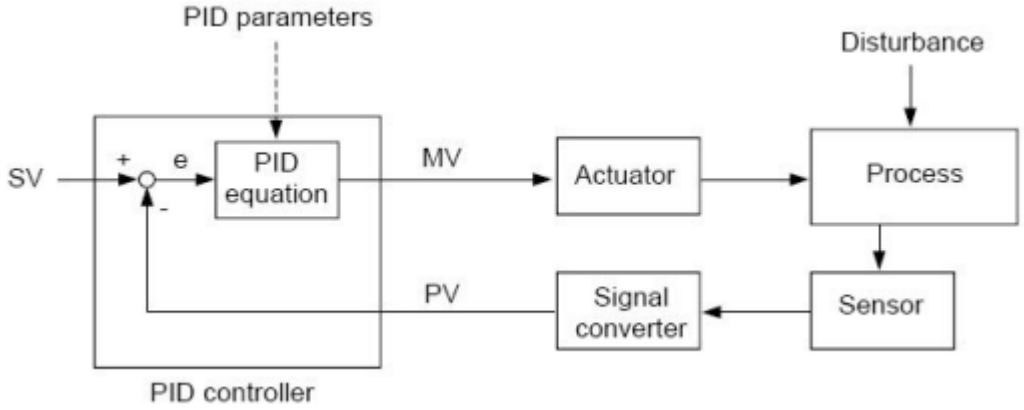
This section, starting on page 8, covers using the PID function block when using IEC 61131-3 Programming Language in MAPware-7000.

PID Basics

The basic idea behind a PID controller is to read a sensor, then compute the desired actuator output by calculating proportional, integral, and derivative responses and summing those three components to compute the output.

The PID controller receives the process variable (PV) and controls the manipulation variable (MV) in order to adjust the PV to match the set value (SV).

The figure below shows a typical configuration for a PID control system.



- **Process**: A physical system in which manipulating an input parameter has an effect on a measured value of the system such as temperature, pressure, flow rate, position, etc.
- **Sensor**: A detector that detects the controlled physical values. It can be a thermocouple, RTD, pressure gauge, flow meter, etc.
- **Signal Converter**: A device that transmits a weak sensor signal to the PID controller by converting it into the signal suited for the environment, such as 4 - 20 mA, 1 - 5 VDC, pulse train, etc.
- **Actuator**: A device that acts as an input to the process by regulating fluid or electric power according to the Signal generated by the PID controller. It can be a control valve, thyristor, and variable speed drive, etc.
- **Process Variable (PV)**: The measured observable value of the system which is to be controlled, normally 4 - 20 mA or 1 - 5 VDC analog signals.
- **Manipulation Variable (MV)**: The PID output which controls the actuator, normally 4 - 20 mA, 1 - 5 VDC, or a time-proportional pulse.
- **Set Point Value (SV)**: The desired target for the Process Variable.
- **Error (e)**: Difference between the PV and SV at a given measurement point.
- **PID Equation**: Algorithm relating the Error, time, and PID parameters to calculate a value for MV. Controls the MV based on the magnitude of E and the PID tuning Parameters. These parameters may be more or less aggressive based on system time responses and operator preferences.

In the PLC, a PID function block and analog input/output modules are used for PID control (one PID loop). To configure two or more PID loops, two or more PID function blocks are used. However, the number of the PID loops configurable by one PLC is normally limited by the available number of analog inputs/outputs.

Proportional (P) Control Action

Proportional (P) control generates the MV in proportion to the error (E).

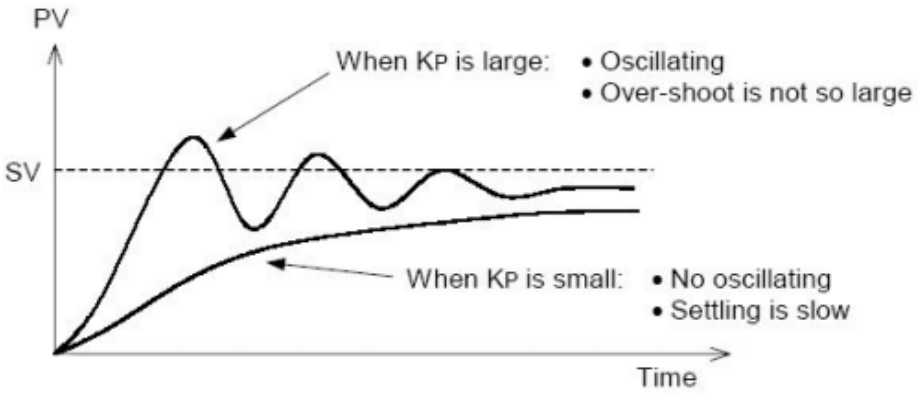
Here, the error (E) is the difference between the SV and the PV, and defined as follows:

$E = SV - PV$

In the P control action, the MV is calculated as follows:

$MV = Kp * E$

KP is called proportional gain. Generally, P control action works as follows:



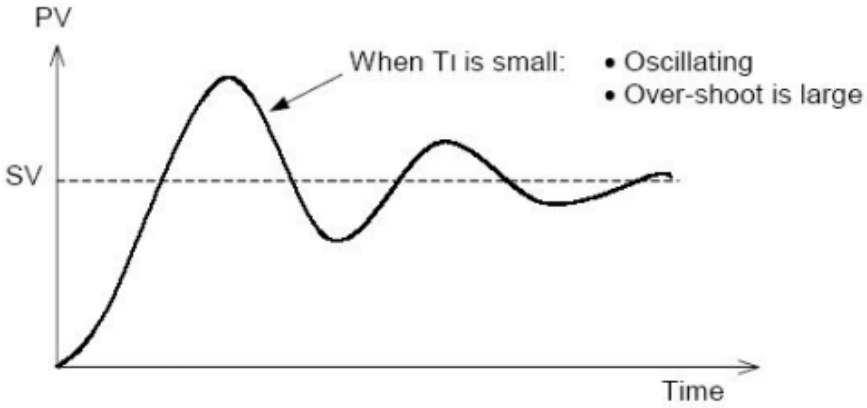
With just P control, an offset (residual error) will remain. Therefore, P control is used with I control (PI control) to eliminate the offset.

Integral (I) Control Action

Integral (I) control will generate the MV in proportion to the time-integral of the error (E). While the error (E) exists, the I control will modulate the MV to eliminate the error (E). Generally the MV with I control is calculated as follows:

$$MV = K_P \cdot \frac{1}{T_I} \int E dt$$

TI is called integral time, the unit of measure is 'seconds per repeat' When TI is large, the MV will change slowly. When TI is small, the MV will change rapidly. That is, the smaller the TI, the larger the integral gain. When TI is too small, oscillation of PV will appear as in the figure below:



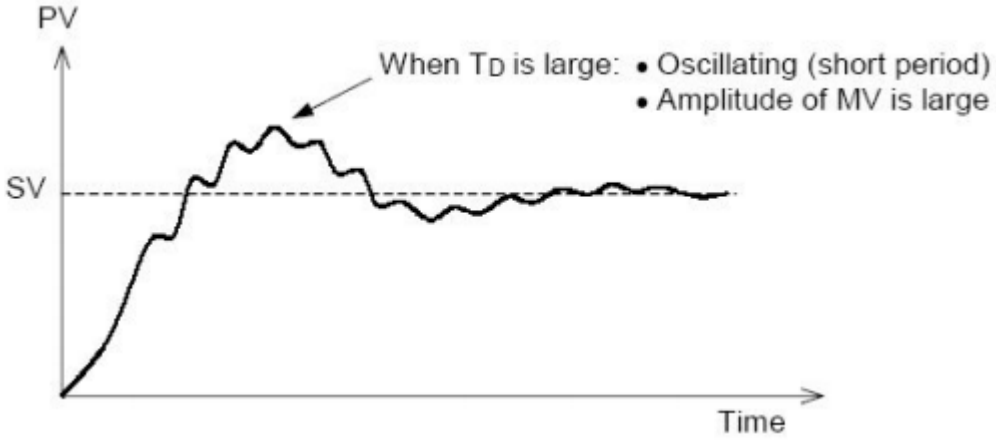
The I control is not used by itself. It is used with P (PI control) or P and D (PID control).

Derivative (D) Control Action

Derivative (D) control will generate the MV in proportion to the rate of change in the error (E). By adding D control, quick corrective action can be obtained at the beginning of an upset condition. Generally MV is related to the D parameter as follows:

$$MV = K_P \cdot T_D \frac{d}{dt} E$$

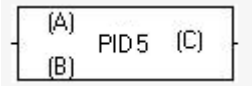
TD is called derivative time, the unit of measure is 'repeats per second'. When TD is large, MV is relatively large for any E. That is, the larger the TD, the larger the derivative gain. If TD is 0 (zero), the D control does not function. When TD is too large, short periodic oscillations of PV will appear:

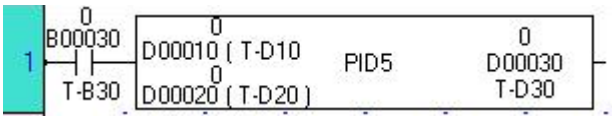


The D control is not used by itself. It is used with P and I (PID control).D control is poorly suited for processes which oscillate rapidly and is seldom used in processes which have a fast response time (most pressure or flow loops). For processes with long lag times (many temperature and some level loops), D control can reduce both the magnitude of E and the potential for large overshoots caused by a given upset condition.

Note that these same processes often act as mechanical integrator and the I tuning parameter can actually stimulate a worsening of the overshoot or oscillation problem.

Native Ladder Implementation





Input (B30)	**Operation**	**Output**
OFF	Initialize/No Operation	OFF
ON	Execute PID every setting interval	ON when Executing

Algorithm Used:

$$MVn = K_P\,[e(t) + \frac{1}{T_I}\int e(t)d(t) + K_D\,\frac{d}{dt}\,e(t)]$$

Here

$e = SVn - PVn$ if reverse action

$e = PVn - SVn$ if forward action

PID5 is the preferred PID instruction when using Native Ladder in MAPware-7000. Three assigned registers (Operands A, B, and C) point to the locations of 24 registers that are used by the PID5 instruction. Using the parameters stored in the 9 registers starting with the register specified by operand B, and the values stored in the 11 registers starting with the register specified by operand C, the PID calculation is executed as described above on the present value PV and the set value SV stored in the 4 registers starting with the register specified by operand A. The increments of manipulation value MV is calculated and stored in the register specified by the Operand C. Registers marked as Reserved below are used internally by the PID5 instruction and should not be otherwise used in your program.

Input Data (Operand A)			**Control Parameters (B)**			**Output Data (C)**		
A	D10	Process Input Value (PV)	B	D20	Proportional Gain (KP)	C	D30	Manipulation Value (MV)
A+1	D11	Set Value (SV)	B+1	D21	Integral Time (TI)	C+1	D31	Previous error (e_{n-1})
A+2	D12	Reserved	B+2	D22	Derivative Gain (KD)	C+2	D32	Previous error (e_{n-2})
A+3	D13	Reserved	B+3	D23	Active Range (AR)	C+3	D33	MV in float
			B+4	D24	Reserved	C+4	D34	MV in float
			B+5	D25	Reserved	C+5	D35	Reserved
			B+6	D26	Reserved	C+6	D36	Reserved
			B+7	D27	Reserved	C+7	D37	Reserved
			B+8	D28	Action Type (ACT)	C+8	D38	Reserved
						C+9	D39	Iterm in float
						C+10	D40	Iterm in float

Parameter Notes:

A: Process Input Value (PV) Data Range: -32768 to +32767

A+1: Set Value (SV) Data Range: -32768 to +32767

B: Proportional Gain (KP) Data Range: -32768 to +32767

B+1: Integral Time (TI) (sec) Data Range: 0 to 32767

B+2: Derivative Gain (KD) Data Range: -32768 to +32767

B+3: Active Range (AR) (percentage) Data Range: 0 to 100

Active Region value = AR * SV / 100

Active Region value is calculated from the Active Range (AR) percentage of the set value (SV). The PID5 instruction is executed only if error (e_n) is less than the Dead band value. When the PID5 instruction is not executed, the MV is set automatically to either 0 or 4095 (MV_{MAX}) depending on comparison between SV and PV.

Note: Set the Active Range to 100 to execute the PID instruction over the full span. If the Active Range is set to 0, the PID5 output will toggle between 0 and 4095.

B+8: Action (ACT) Data Range: 0 to 1

0: Direct Action, MV increases when the error becomes more positive.

1: Reverse Action, MV decreases when the error becomes more positive.

There are two basic actions of PID with respect to the control direction of MV, direct action and reverse action. Direct action will lead the MV to increase when the PV is larger than the SV (for example, a cooling application). Reverse action will lead the MV to decrease when the PV is larger than the SV (for example, a heating application).

C: Manipulation Value (MV) Data Range: 0 to 4095

C+1: Previous error Value (e_{n-1}) Data Range: -32768 to +32767

C+2: Previous error Value (e_{n-2}) Data Range: -32768 to +32767

NOTE: You must ensure that the PID5 instruction is executed once every scan interval in the Ladder Logic. Multiple PID5 instructions can be used in a program.

NOTE: The PID4 instruction has been deprecated. It used the same PID equation as the PID5 instruction covered here, but variables are configured in slightly different locations. See the MAPware-7000 help files for PID4 configuration information if using an older version of MAPware-7000.

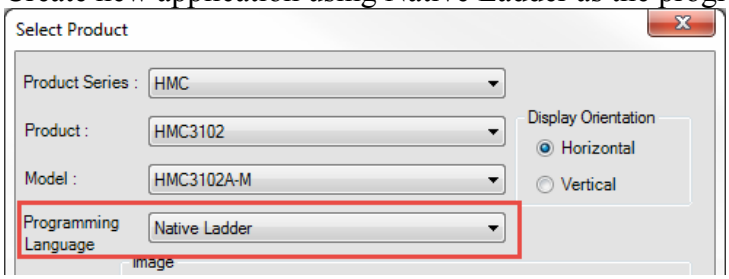
NOTE: The PID1 instruction is not covered in this tech note and is not recommended for PID control. It uses the following PID equation:

$$M = Kp \cdot \left[(e - e_{-1}) + INT\left(\frac{|K_{IL}| \cdot e + |r}{|K_{IH}|}\right) + INT\left\{\frac{|K_{DH}|}{|K_{DL}|} \cdot (2P_{-1} - P - P_{-2})\right\} \right]$$

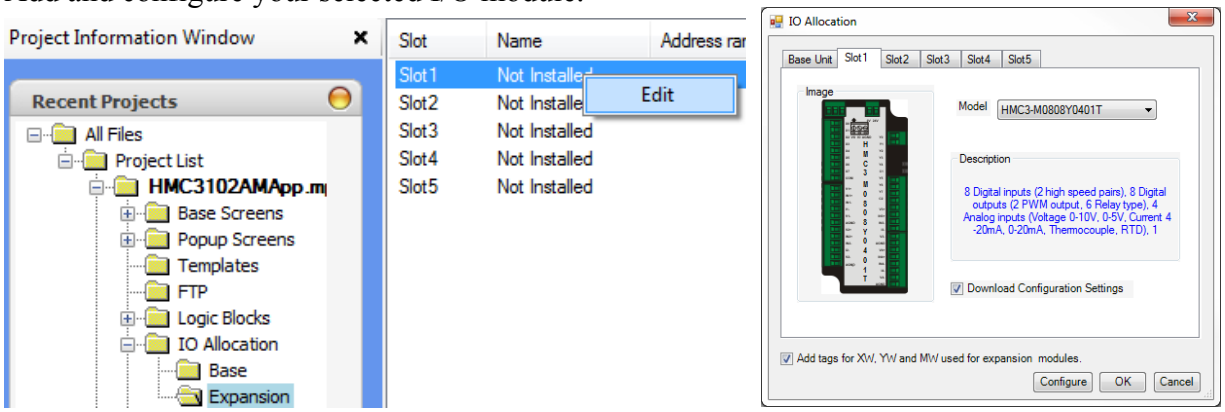
Only one PID1 instruction can be used in a program. See the MAPware-7000 help files for configuration instructions if this formula is desired.

Configure a PID5 Instruction

1. Create new application using Native Ladder as the programming language.



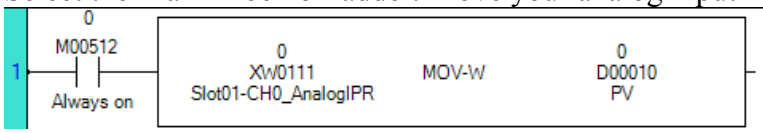
2. Add and configure your selected I/O module.



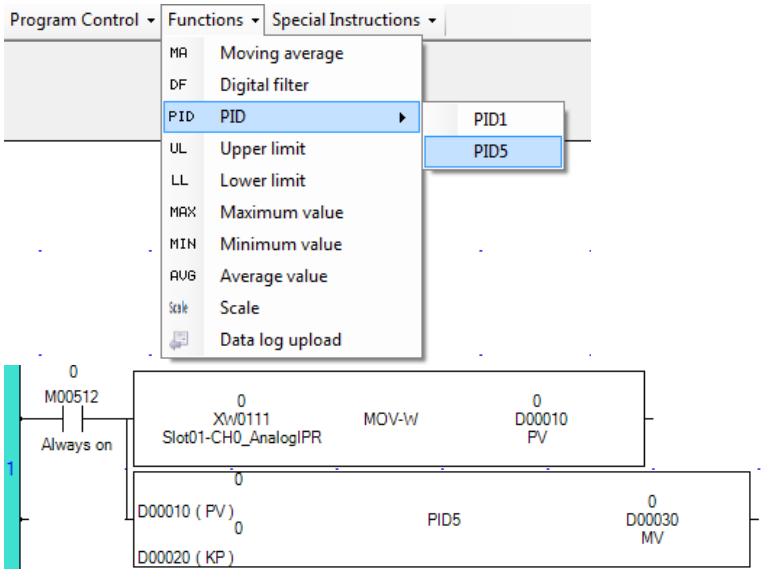
3. Create tags to reference in the PID5 instruction. Use the chart in the previous section to ensure parameters are created at the correct addresses.

Tag No	Tag Name	Tag	Attribute	Tag Address	Port	Node	Node Name	Tag Category
87	PV	2	Read Write	D00010	-	0	HMC3102A-M	User Defined Tag
88	SV	2	Read Write	D00011	-	0	HMC3102A-M	User Defined Tag
89	KP	2	Read Write	D00020	-	0	HMC3102A-M	User Defined Tag
90	TI	2	Read Write	D00021	-	0	HMC3102A-M	User Defined Tag
91	KD	2	Read Write	D00022	-	0	HMC3102A-M	User Defined Tag
92	AR	2	Read Write	D00023	-	0	HMC3102A-M	User Defined Tag
93	ACT	2	Read Write	D00028	-	0	HMC3102A-M	User Defined Tag
94	MV	2	Read Write	D00030	-	0	HMC3102A-M	User Defined Tag

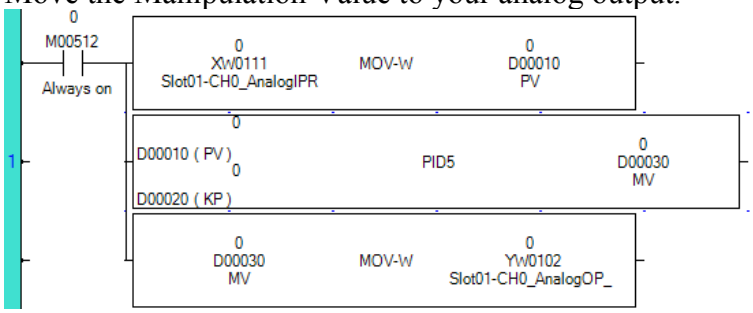
4. Select the Main Block of ladder. Move your analog input into the Process Input Value.



5. Add a PID5 instruction.



6. Move the Manipulation Value to your analog output.



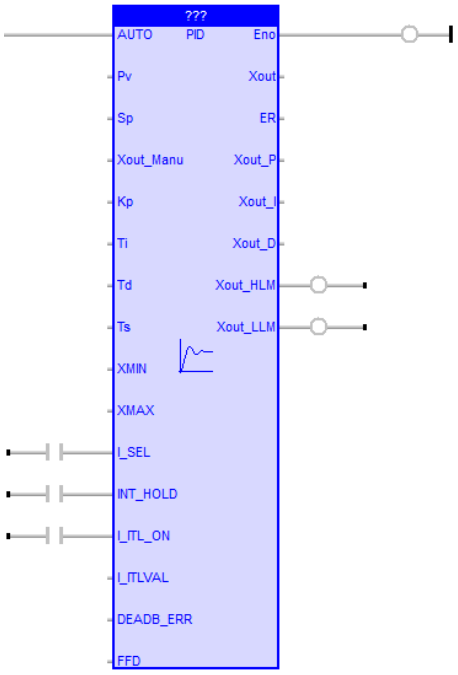
NOTE: In most cases, additional manipulation of the analog input and output signals will be required. This example shows a very simple loop.

7. Create a screen to allow the tuning parameters to be adjusted.

PID Settings

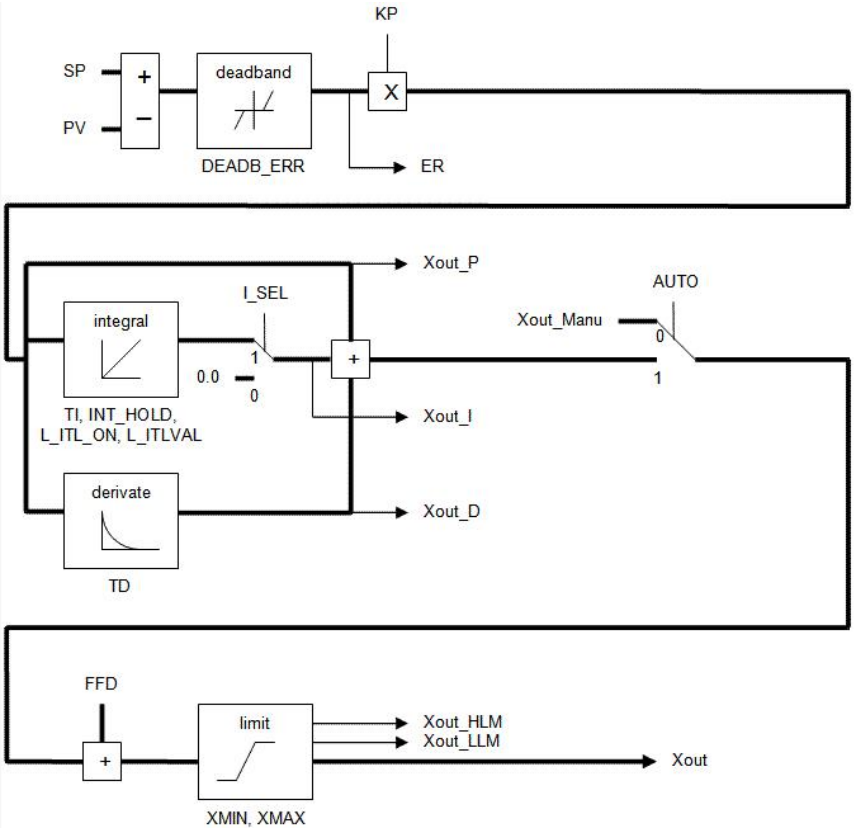
INPUTS		PARAMETERS		OUTPUT	
Process Value (PV)	99999	Proportional Gain (KP)	99999	Manipulation Value (MV)	99999
Set Value (SV)	99999	Integral Time (TI)	99999		
		Derivative Gain (KD)	99999		
		Active Range (AR)	99999		
		Action Type (ACT)	99999		

IEC Mode Implementation




```
// Built in FB with multiple inputs
// and multiple outputs
MyPIDInstance(auto,
              Pv,
              Sp,
              xout_manu,
              Kp,
              Ti,
              Td,
              Ts,
              xmin,
              xmax,
              i_sel,
              int_hold,
              i_itl_on,
              i_itlval,
              deadb_err,
              ffd
);
x_out := MyPIDInstance.Xout;
er := MyPIDInstance.ER;
x_out_p := MyPIDInstance.Xout_P;
x_out_i := MyPIDInstance.Xout_I;
x_out_d := MyPIDInstance.Xout_D;
x_out_hlm := MyPIDInstance.Xout_HLM;
x_out_llm := MyPIDInstance.Xout_LLM;
```

Ladder Diagram | Function Block | Structured Text



The PID instruction in IEC 61131-3 Mode is supported in Ladder Diagram (LD), Function Block Diagram (FBD), and Structured Text (ST) programming languages as pictured above. Additional inputs allow for more PID features.

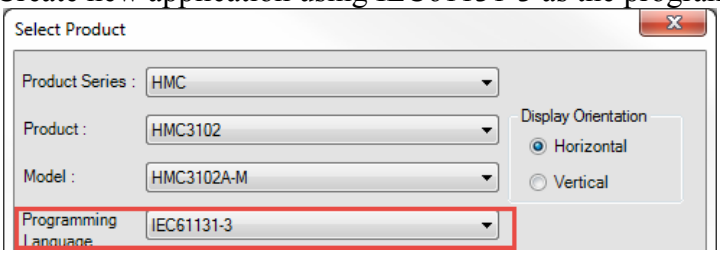
Input Operand	Data Type	Description
AUTO	BOOL	TRUE = normal mode - FALSE = manual mode.
PV	REAL	Process value
SP	REAL	Set point
Xout_Manu	REAL	Output value in manual mode
KP	REAL	Proportional gain.
TI	REAL	Integration factor
TD	REAL	Derivation factor
TS	TIME	Sampling period, must be larger than PLC cycle time
XMIN	REAL	Minimum allowed output value
XMAX	REAL	Maximum output value
I_SEL	BOOL	If FALSE, the integrated value is ignored. Must be set to TRUE for Integral Control to fuction.
INT_HOLD	BOOL	If TRUE, the integrated value is frozen
I_ITL_ON	BOOL	If TRUE, the integrated value is reset to I_ITLVAL
I_ITLVAL	REAL	Reset value for integration when I_ITL_ON is TRUE
DEADB_ERR	REAL	Hysteresis on PV. PV will be considered as unchanged if greater than (PVprev - DEADBAND_W) and less than (PRprev + DEADBAND_W)
FFD	REAL	Disturbance value on output

XMIN and XMAX are required inputs for the PID instruction to function and when used with the FFD disturbance value can scale your output. If Xout is saturated to either limit, the Integration output is frozen. If the PID loop is set to manual mode, Xout is set to Xout_Manu + FFD.

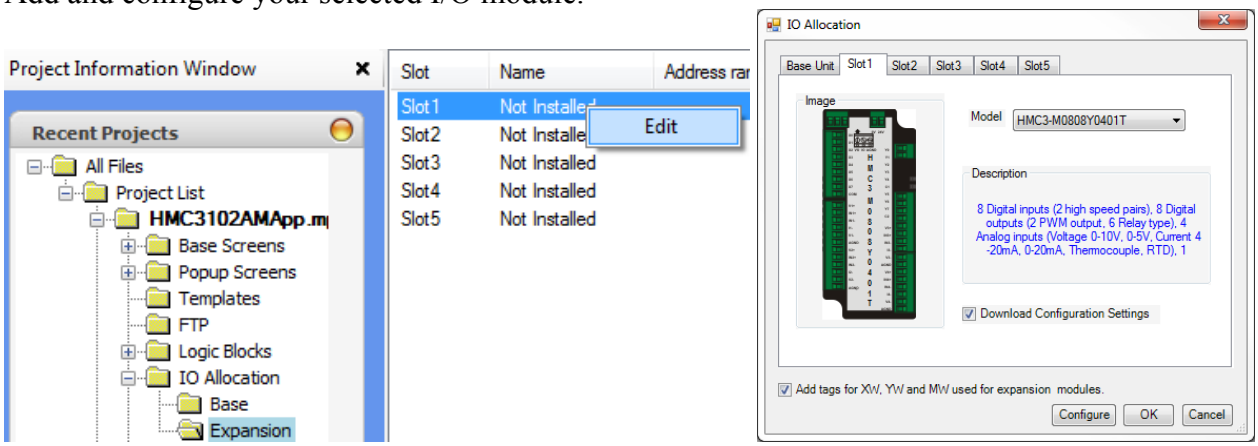
Output Operand	Data Type	Description
Xout	REAL	Output command value (Manipulation Value)
ER	REAL	Last calculated error
Xout_P	REAL	Last calculated proportional value
Xout_I	REAL	Last calculated integrated value
Xout_D	REAL	Last calculated derived value.
Xout_HLM	BOOL	TRUE if the output value is saturated to XMIN
Xout_LLM	BOOL	TRUE if the output value is saturated to XMAX

Configure an IEC PID Instruction

1. Create new application using IEC61131-3 as the programming language.



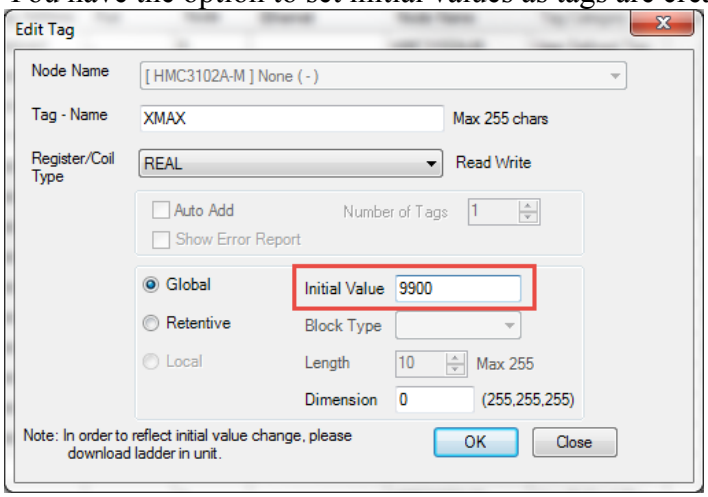
2. Add and configure your selected I/O module.



3. Create tags to reference in the PID5 instruction.

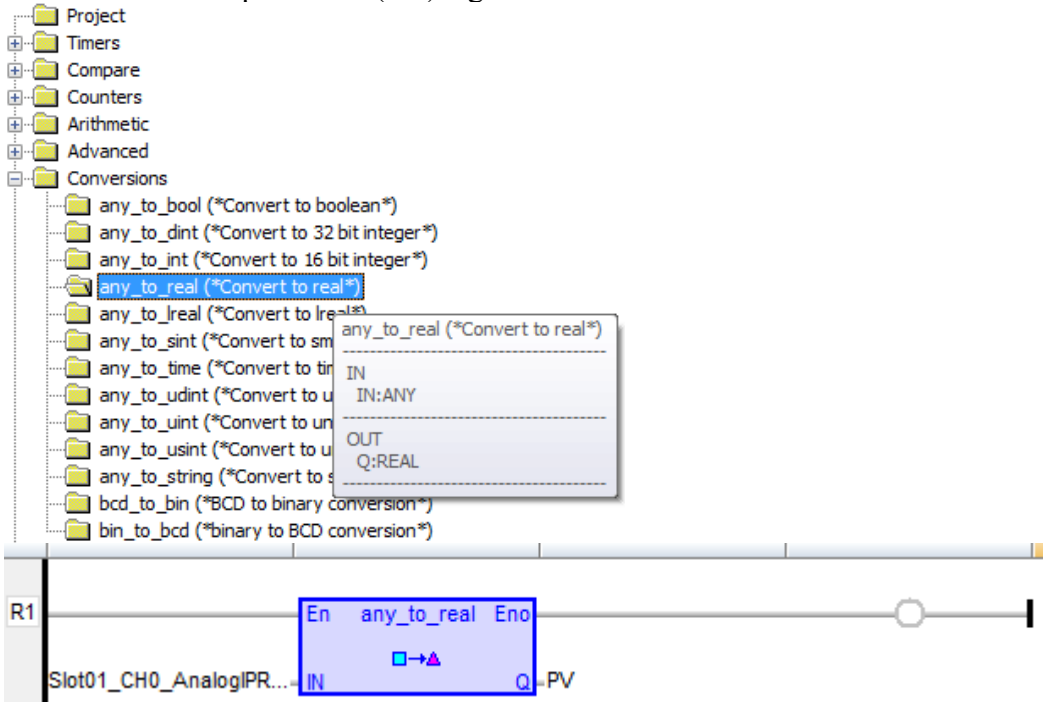
Tag No	Tag Name	Data Type	Attribute	Node	Ethernet	Node Name	Tag Category
298	Xout_LLM	BOOL	Read Write	0		HMC3102A-M	UserDefined Tag
297	Xout_HLM	BOOL	Read Write	0		HMC3102A-M	UserDefined Tag
296	Xout_D	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
295	Xout_I	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
294	Xout_P	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
293	ER	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
292	Xout	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
291	FFD	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
290	DEADBAND_ERR	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
289	I_ITLVAL	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
288	I_ITL_ON	BOOL	Read Write	0		HMC3102A-M	UserDefined Tag
287	INT_HOLD	BOOL	Read Write	0		HMC3102A-M	UserDefined Tag
286	I_SEL	BOOL	Read Write	0		HMC3102A-M	UserDefined Tag
285	XMAX	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
284	XMIN	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
283	TS	TIME	Read Write	0		HMC3102A-M	UserDefined Tag
282	TD	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
281	TI	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
280	KP	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
279	Xout_Manual	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
278	SP	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
277	PV	REAL	Read Write	0		HMC3102A-M	UserDefined Tag
276	AUTO	BOOL	Read Write	0		HMC3102A-M	UserDefined Tag

4. You have the option to set initial values as tags are created.

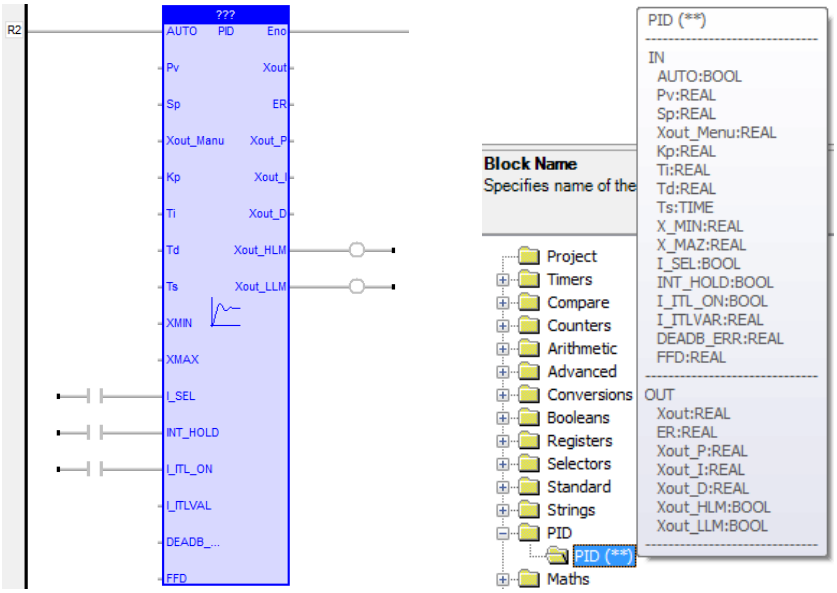


This can alternatively be done in a Power-up Task. The SP, KP, TI, TD, TS, XMIN, XMAX, FFD, and I_SEL values are all good candidates for preset values, as this will allow you to begin testing the PID loop quickly.

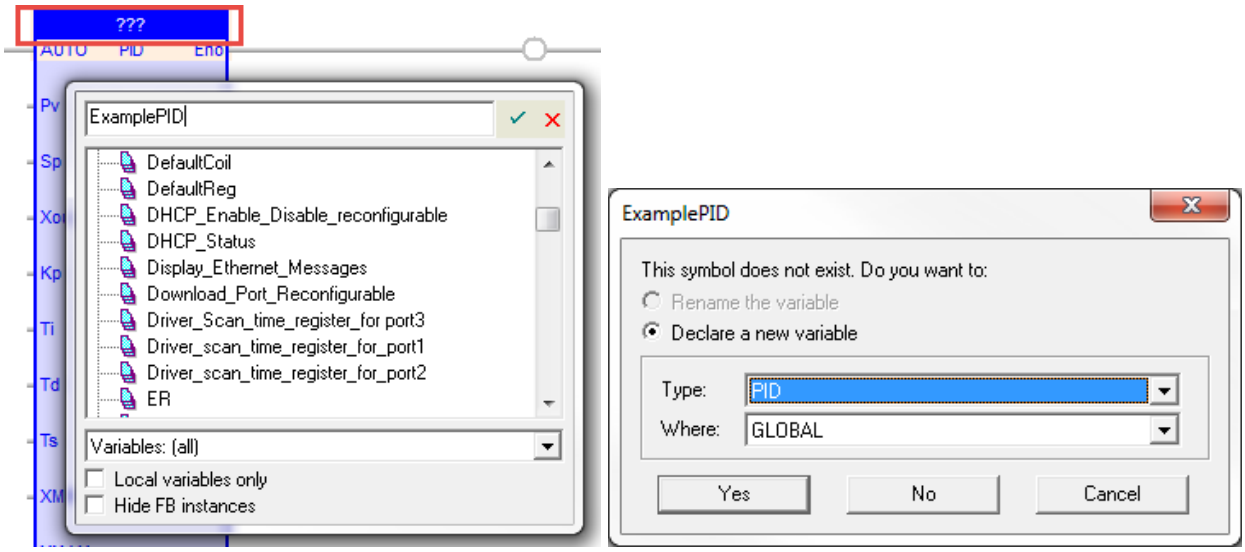
5. Open Main Block1. Select the **any_to_real** instruction from the **Conversions** folder of the Instruction List section in the bottom right corner of MAPware-7000 to move your analog input into the Process Input Value (PV) tag.

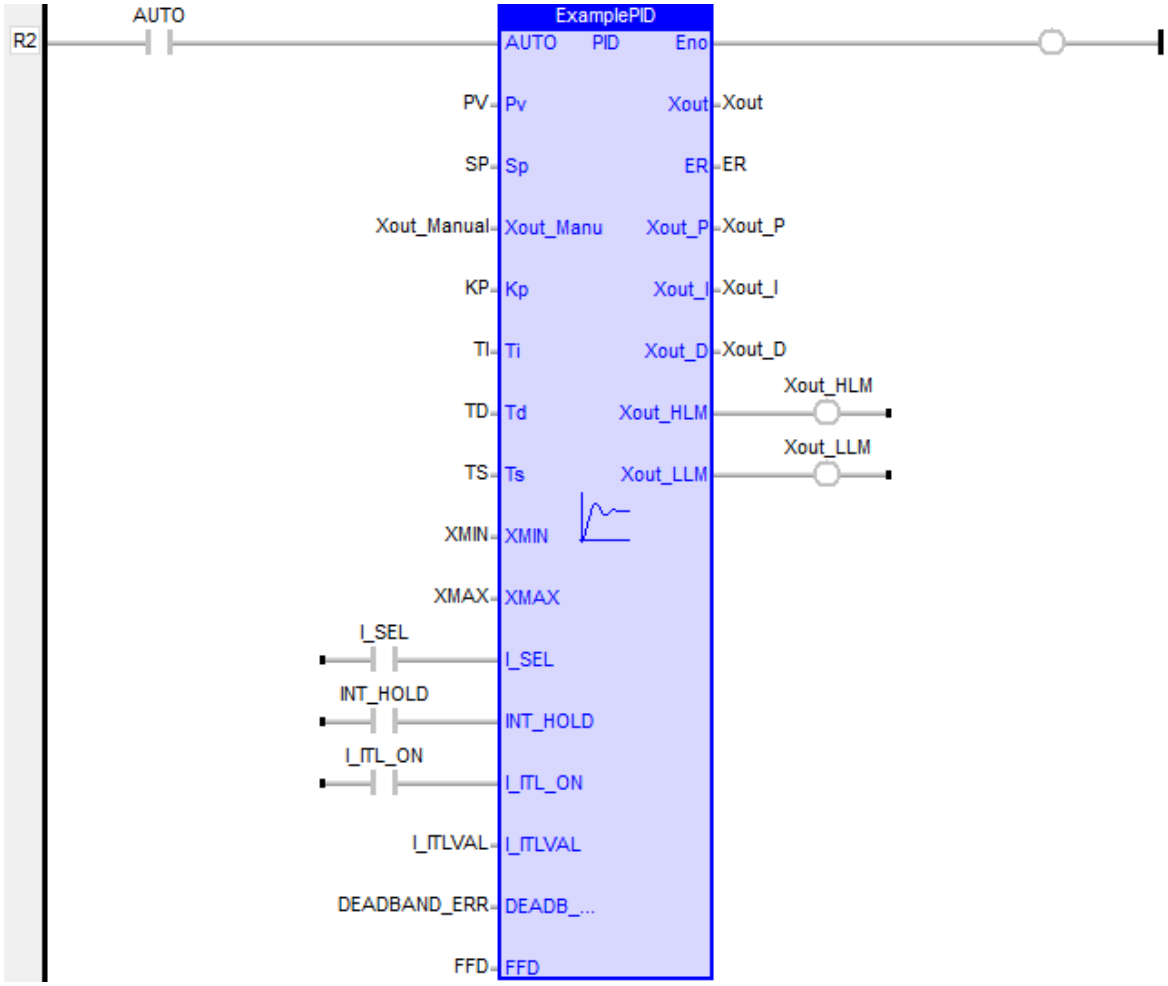


6. Add a **PID (**)** instruction.

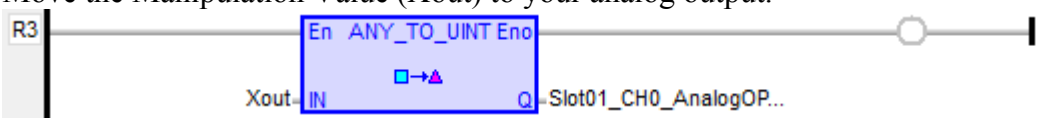


7. Double-click the **???** to create an instance of the PID instruction and then assign the tags created above.



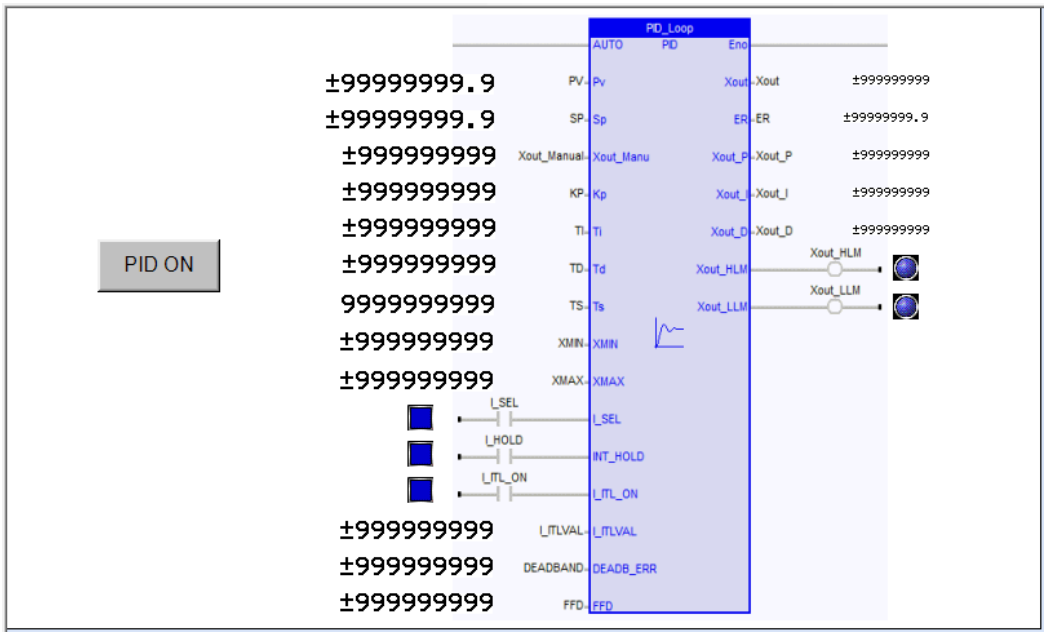


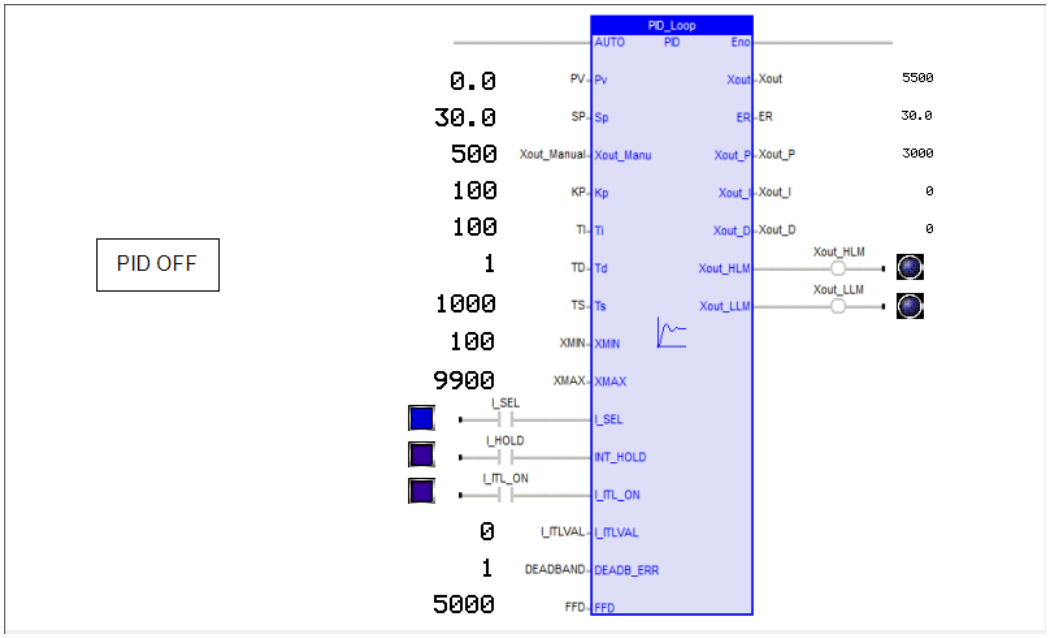
8. Move the Manipulation Value (Xout) to your analog output.



NOTE: In most cases, additional manipulation of the analog input and output signals will be required. This example shows a very simple loop.

9. Create a screen to allow the tuning parameters to be adjusted.





Short Sample Project Description

The IEC PID Sample Project takes a Thermocouple temperature input as the PV input, and outputs a PWM signal to adjust the external heating element. The Thermocouple and PWM are configured in the IO Allocation Expansion Configuration Settings.

XMIN (preset to 100), XMAX (preset to 9900), and FFD (preset to 5000) are used to scale Xout into a value that is easily processed to a duty cycle for the PWM output. The Eno (Output Enabled) signal from the PID instruction's ladder output is used to enable the PWM output when the instruction is enabled. TS is preset to 1000 (sample time of 1 second), and I_SEL is preset on to enable the Integral Control. Initial values of 100, 1000, and 1 are used for KP, TI, and TD respectively. These will be adjusted during runtime as the PID loop is tuned.