APPLICATION NOTE

AC500 V3

PID - FUNCTION BLOCK

Contents

1 Introduction

1.1 Scope of the document

This document gives an introduction into the HaModPidFixCycle function block who is internally a PID function block and use with fixed cycle time.

1.2 Compatibility

The application note explained in this document has been used with the below engineering system versions. They should also work with other versions, nevertheless some small adaptations may be necessary, for future versions.

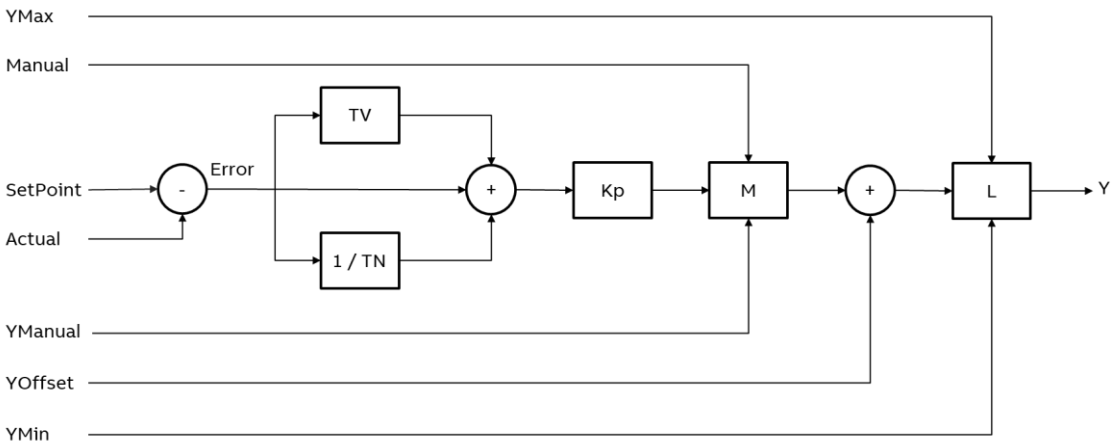
- AC500 V3 PLC

2 Explanation

	For fast and fix tasks it is recommended to use PID_FIXCYCLE instead of PID, because the cycle time is defined accurately, whereas PID only can measure the cycle time with a maximum accuracy of milliseconds. In case of very short cycles (1ms) this might lead to a rough run.

	It is not necessary to readjust the controller parameters (KP, TN, TV) if the cycle time changes.

A PID controller continuously calculates an error value e(t) as the difference between a desired set point and a measured process variable. The PID controller applies a correction based on proportional, integral, and derivative terms (sometimes denoted KP, TN and TV respectively) which give their name to the controller type.



- P accounts for present values of the error. For example, if the error is large and positive, the control output will also be large and positive.
- I account for past values of the error. For example, if the current output is not sufficiently strong, the integral of the error will accumulate over time, and the controller will respond by applying a stronger action.
- D accounts for possible future trends of the error, based on its current rate of change.[1]

As a PID controller relies only on the measured process variable, not on knowledge of the underlying process, it is broadly applicable. By tuning the three parameters of the model, a PID controller can deal with specific process requirements. The response of the controller can be described in terms of its responsiveness to an error, of the degree to which the system overshoots a setpoint, and of the degree of any system oscillation. The use of the PID algorithm does not guarantee optimal control of the system or even its stability.

YOffset, **YMin** and **YMax** serve for transformation of the manipulated variable within a prescribed range. **Manual** can be used to switch to manual operation; **Reset** can be used to reinitialize the controller. In normal operation (Manual=Reset=FALSE) the controller calculates the controller error e as difference from **SET_POINT–ACTUAL**, generates the derivation with respect to time and stores these values internally.

The output **Y** is the manipulated variable unlike the PD controller contains an additional integral part, and is calculated as follows:

Y:= YOffset + Kp * (rError + rIOutbyTn + D.Out * Tv)

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

As long as the limitation for the manipulated variable (YMin, Ymax) is active, the integral part will be adapted, like if the history of the input values had automatically affected the limited output value. If this behavior is not wanted, the following workaround is possible: Switch off the limitation at the PID controller (YMin >= Ymax) and instead apply the LIMIT operator (IEC standard) on output value Y (see an example in the figure below).

It is not necessary to readjust the controller parameters (KP, TN, TV) if the cycle time changes.

InOut:

Scope	Name	Type	Initial	Comment
Input	Actual	REAL		Current value, process variable
	Set-Point	REAL		Desired value, set point
	KP	REAL		Proportionality const. P
	TN	REAL		Reset time I [sec]
	TV	REAL		Rate time, derivative time D [sec]. If set to 0, then it works as PI controller
	YManual	REAL		Y is set to this value as long as Manual = TRUE
	YOffset	REAL		Offset for manipulated variable
	YMin	REAL		Minimum value for manipulated variable
	YMax	REAL		Maximum value for manipulated variable
	Manual	BOOL		TRUE: Manual: Y is not influenced by controller FALSE: Controller determines Y

	Reset	BOOL		TRUE: Set Y output to YOffset and reset integral part
	Cycle	Real		Time in seconds between two calls
Output	Y	REAL		Manipulated variable, set value
	LimitsActive	BOOL	FALSE	TRUE: Y has exceeded the given limits YMin, YMax and is limited to these values
	OverFlow	BOOL	FALSE	Overflow in integral part: Because of the additional integral part, an overflow can come about by incorrect parameterization of the controller, if the integral of the error e becomes too great. Therefore, for the sake of safety a Boolean output called Overflow is present, which in this case would have the value TRUE. This only will happen if the control system is instable due to incorrect parameterization. At the same time, the controller will be suspended and will only be activated again by re-initialization. Note: When Ymax and Ymin are set to proper values, this will stop further integration once Ymax or Ymin are reached.

2.1 Tuning of PID

PID task time is defined as 50ms. Based on the process needs it has to be adapted.
To Tune PID controller Manually normally following steps are followed:

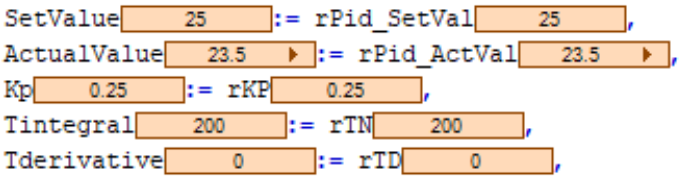
2.1.1 Set proportional Gain

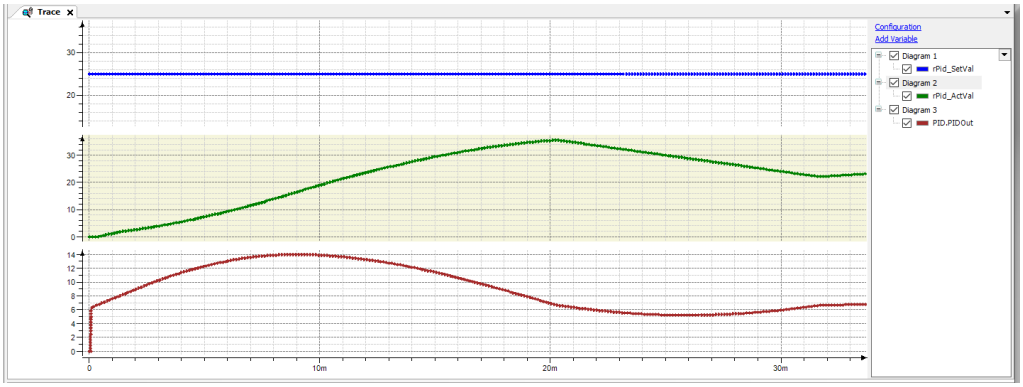
Adding proportional gain for the PID controller is the first step. To nullify Integral part, set a very high value to it e.g., 200 second or more. To nullify Derivative part set its value as zero.

- Set a very low P value like 0.25 and give a step set point change and observe behavior of PID output, actual value variation and how the output value is settled.
- Now increase the P gain in steps and check how fast would you like to make the system. However, output should not become oscillatory, otherwise proportional gain must be reduced.

Explanation of the above process is as follows
We are controlling pressure inside water tank, which has a fixed outlet flow. Inlet flow is controlled using PID

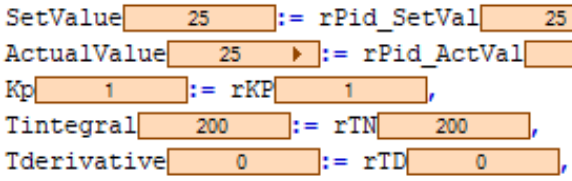
A. KP = 0.25, TN = 200 seconds, TV = 0

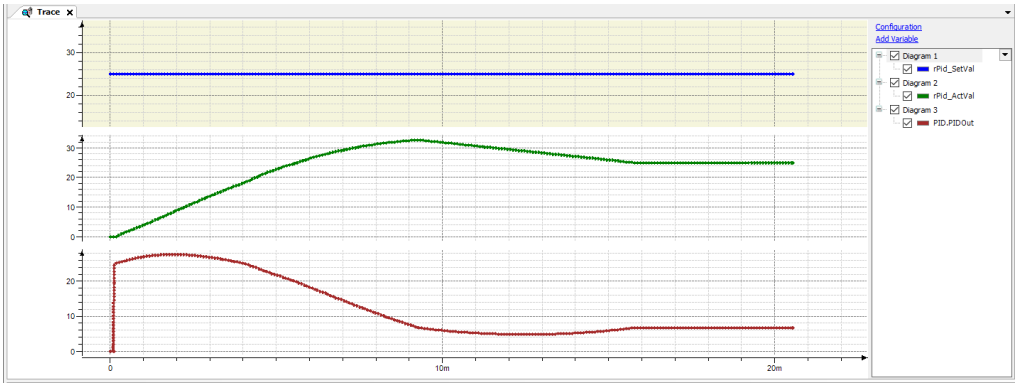




After giving a step setpoint 25, we can observe that output value (PIDOut) is changing based on the PID Act value and PID Set value. As only proportional gain is active, PID needs more time to settle down.

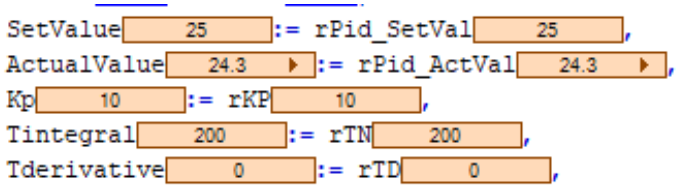
B. Kp = 1, TN = 200 seconds, Tv = 0

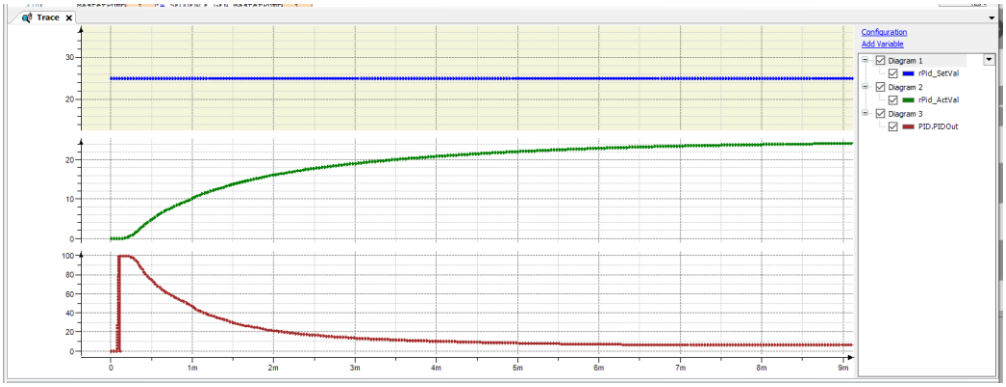




Proportional gain is increased to 1 to check its effects on the process.
As proportional gain is higher, PID will change PID output quicker than Case A, it means PID is faster.

C. Kp = 10, TN = 200 second, Tv = 0





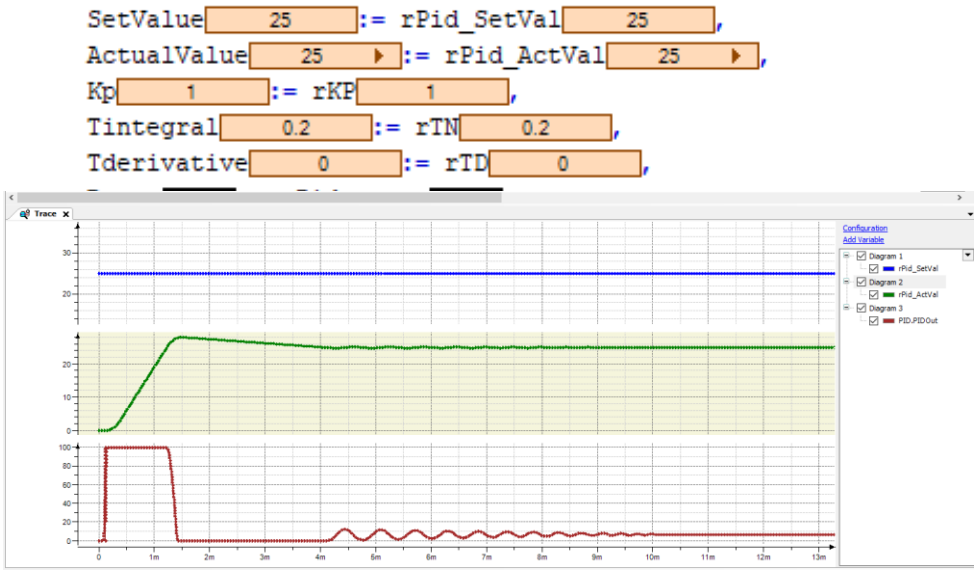
Proportional gain is increased to 10 to check its effects.
As proportional gain is higher, PID will change PID output quicker than Case B, it means PID is faster.

Depending on the process, user needs to decide how fast response the process can withstand and what is the safety limit and only according to that proportional gain must be set.
For this example, we are setting proportional gain to 1.

2.1.2 Set Integral part

Once proportional gain is set, then we need to decrease integral time in such a way that output should settle down with minimum oscillation. This is one of the critical procedures in PID tuning and needs expertise in the process and its behavior.

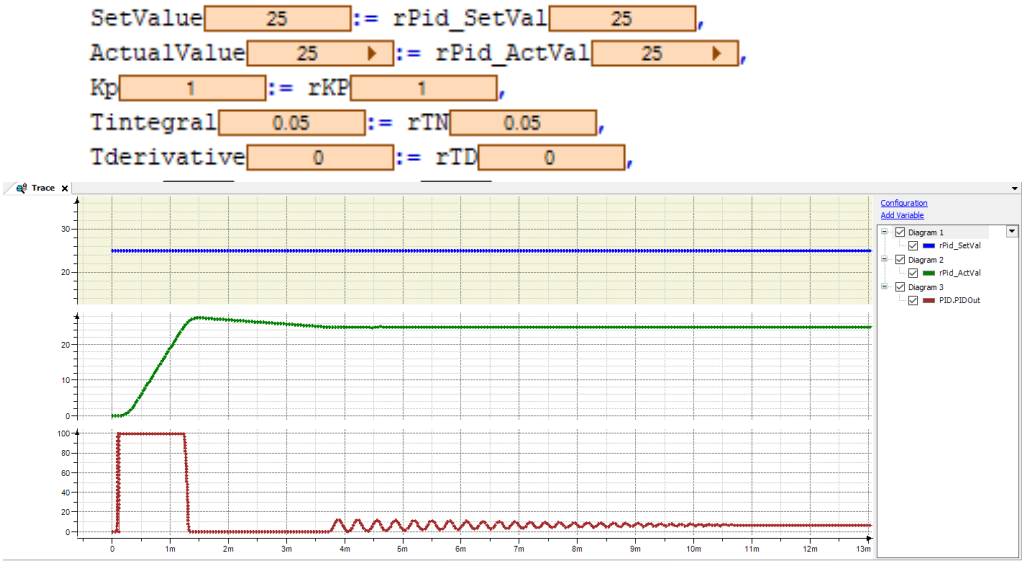
A. KP = 1, TN = 0.2 seconds, TV = 0

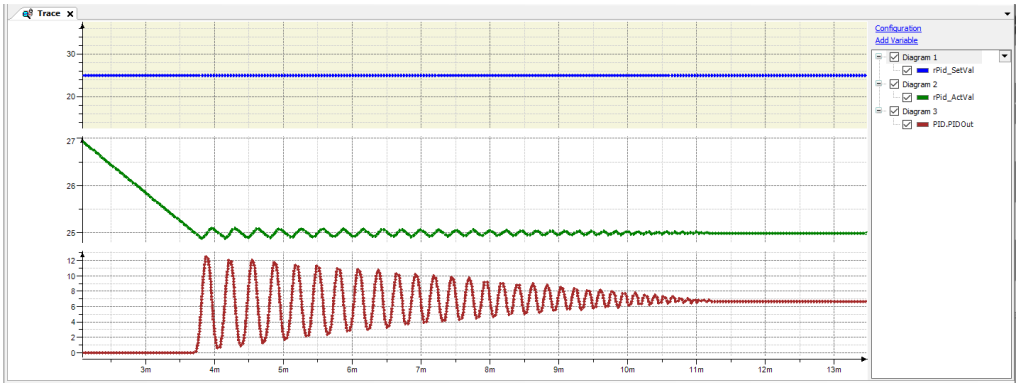


Integral time decreased to 0.2 seconds from 200 to check its effects on the process.

Due to continuous integration of the error at 0.2 seconds, PID gives output much faster than previous case. This leads to faster response and settling of the process.

B. KP = 1, TN = 0.05, seconds, TV = 0





Integral time decreased to 0.05 seconds from 200 to check its effects on the process.
Due to continuous integration of the error at 0.2 seconds, PID gives output much faster than previous case. This leads to faster response and settling of the process. PID output oscillations are faster with smaller peak compared to last case.

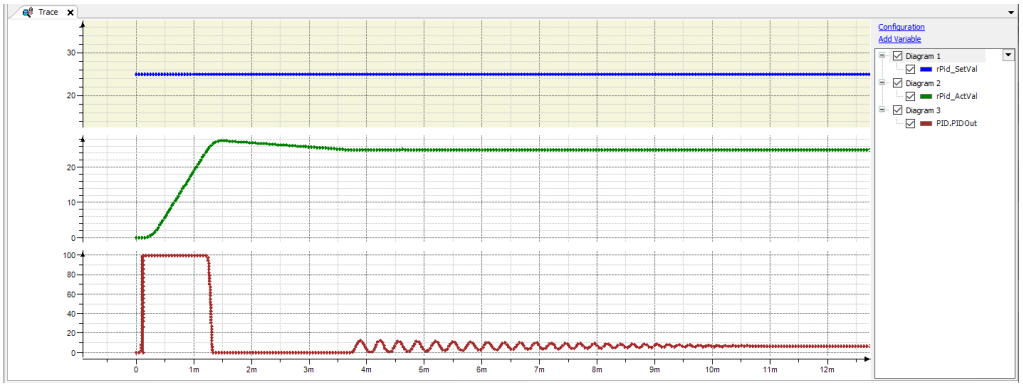
However, as we have observed that PID output is changing much faster compared to earlier setting i.e., when only proportional gain is active. User must check if the process is allowing this behavior. If smoother response is needed, then user needs to increase Integration time. If faster response is allowed, then user can further decrease integration time.
For this example, we are setting proportional gain to 1 and Integration time to 0.05 seconds.

2.1.3 Set differential part

This is last step of the PID tuning, D accounts for possible future trends of the error, based
on its current rate of change. D part comes into the picture when there is change in the setpoint
and it helps actual value to settle down quicker.
D part has to be kept very minimum or it might damage the system as it tends to increase output
very quickly.

A. KP = 1, TN = 0.05 seconds, TV = 0.2

```
SetValue       25    := rPid_SetVal      25    ,
ActualValue    25  ▶ := rPid_ActVal      25  ▶ ,
Kp              1    := rKP              1    ,
Tintegral      0.05  := rTN              0.05 ,
Tderivative    0.2   := rTD              0.2  ,
```



Derivative time 0.2 added to check its effects.
When the PID set value is changed, then only Derivation part comes into the picture. Traditionally many processes work only with PI controller and Derivative part is kept zero.
Extra care must be taken by user for setting TV, based on the process and allowed change in parameters.

ABB AG
Eppelheimer Straße 82
69123 Heidelberg, Germany
Phone: +49 62 21 701 1444
Fax: +49 62 21 701 1382
E-Mail: plc.support@de.abb.com
www.abb.com/plc