

Legal information

Qualified Personnel

This function must only be handled by qualified personnel whose specialization matches the requirements. A qualified person is someone with relevant experience and the ability to identify hazards and risks associated with the operation. Improper handling by unqualified individuals can lead to operational failures and safety risks.

Disclaimer of Liability

We have thoroughly reviewed the contents of this publication to ensure that it aligns with the described hardware and software. However, as some variations may occur, we cannot guarantee complete consistency. The information in this publication is regularly reviewed, and any necessary updates or corrections will be included in future editions.

Proper use of Halow-Tech products

Products should only be utilized for the specific applications outlined in the accompanying catalog and related technical documents. If incorporating products or components from other manufacturers, they must be either recommended or authorized by the relevant company. To ensure safe and trouble-free operation, proper handling during transport, storage, installation, assembly, commissioning, operation, and maintenance is essential. It is important to adhere to the permissible environmental conditions and to follow the guidelines provided in the associated documentation.

Contents

HydroXact	2
Description	2
Block diagram	4
Inputs	4
Outputs	10
FAQ	11
Examples	11
General behavior	13
Setting control and feedforward Parameters	14
Stopping the current movement	17

Description

This function block has been developed to facilitate position control of different hydraulic axes, including:

- Single acting cylinder
- Plunger cylinder
- Double acting cylinder
- Double acting double rod cylinder

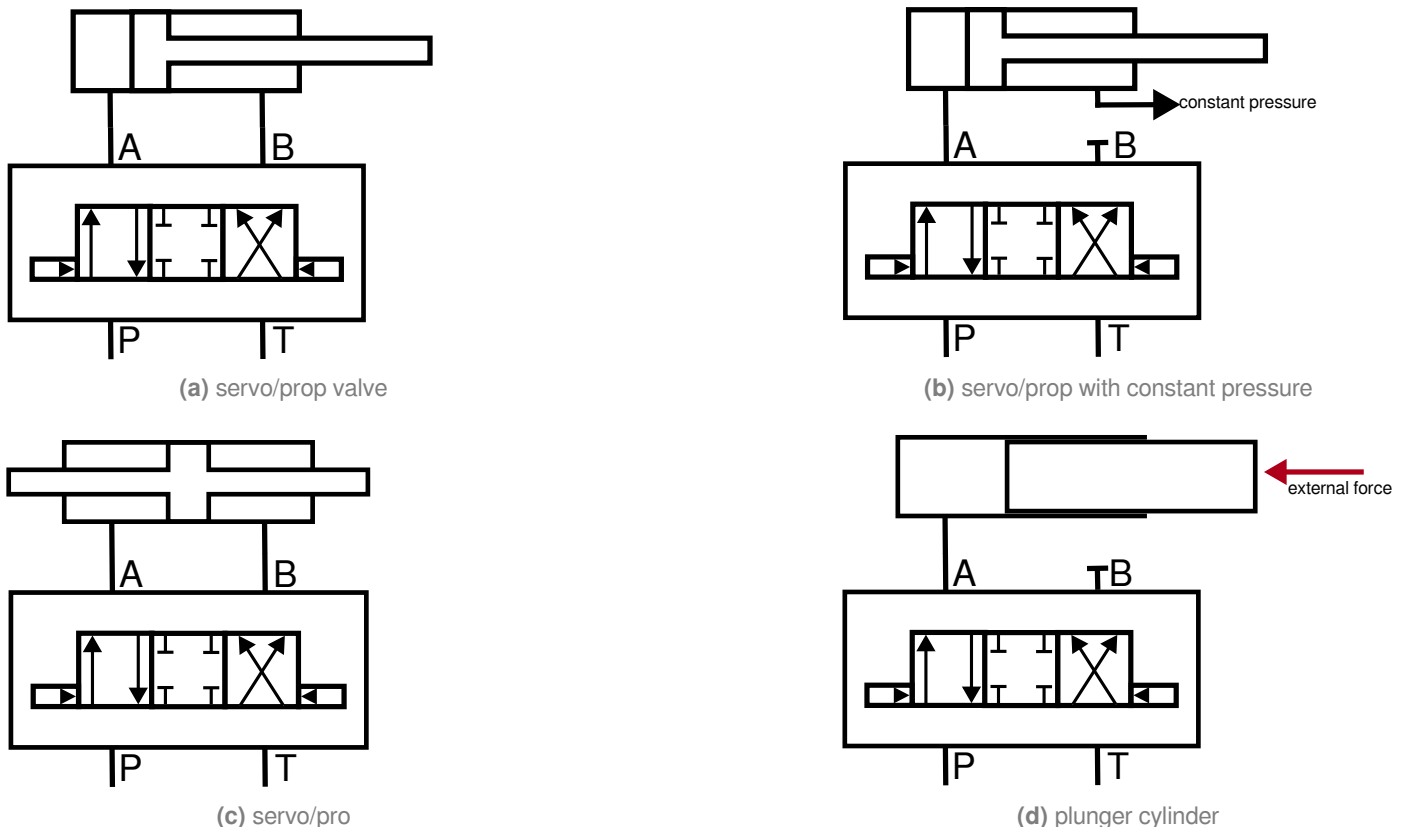


Figure 1: Position control for the hydraulic setup shown

The HydroXact PLC function block is an advanced and robust controller designed for the precise management of position within hydraulic axes. Engineered to support a wide range of hydraulic cylinders—including single-acting, plunger, double-acting, and double-acting double rod cylinders—it seamlessly integrates with linear servo or proportional valves. With versatile input parameters, the HydroXact block operates exclusively in position control mode.

At the heart of HydroXact is an internal trajectory generator that enables the controller to utilize both closed-loop and open-loop control algorithms simultaneously. This integration allows for rapid, precise, and stable adjustments in position control, enhancing the performance and reliability of your hydraulic system. Users can specify parameters such as target position, velocity, acceleration, and jerk for axis motion, tailoring the system’s response to meet specific application requirements.

One of the key advantages of HydroXact is its ease of parameterization and commissioning, making it straightforward for users to implement and operate. Additionally, the block provides valuable monitoring outputs, such as position, velocity, and acceleration trajectories, along with a busy signal to indicate active movements. These outputs are crucial for comprehensive system diagnostics and streamline troubleshooting processes during operation.

The accompanying diagram illustrates how the HydroXact function integrates into the overall control system, highlighting its role in achieving precise and reliable position control of the cylinder.

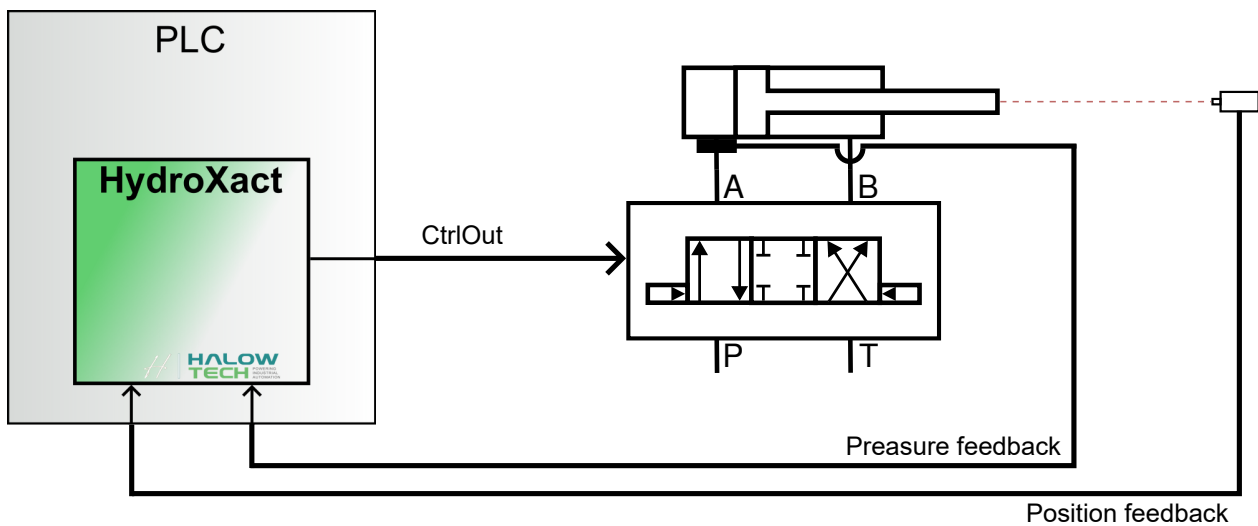


Figure 2: Integration of the function into the overall control system.

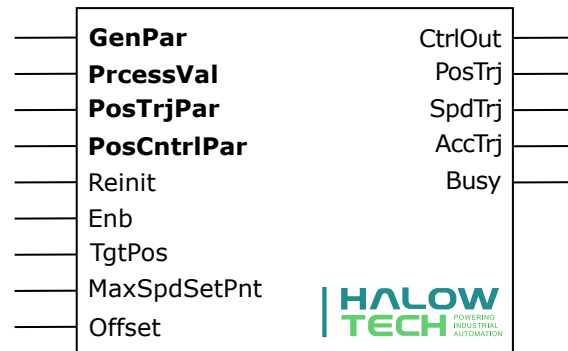
Compatible with various PLC platforms like Siemens S7, Siemens TIA Portal, Rockwell Studio AIO, Rockwell RSLogix 5000, Rexroth IndraWorks, B&R Automation Studio, and Beckhoff TwinCAT, FlexiMotion provides the same high performance on any platform.

Needed PLC Functions¹

Mainfunction:	HydPosControl	HaTe_HydPosControlV1a
Subfunction:	SubFunktion1	HaTe_SubFunktion1V3c

¹ In addition to the main function, various subfunctions may be required. These need to be imported into the PLC program as well. If you are already using other functions from Halow-Tech that utilize the same subfunction, it does not need to be imported again.

Block diagram



Inputs

GenPar		
	<i>SampleTime</i>	<REAL>
	<i>MaxOut</i>	<REAL>
	<i>MinOut</i>	<REAL>

GenPar → SampleTime - Calling frequency of the controler, <REAL>

The sample time, in seconds, of the cyclic interrupt task of the PLC at which the function is running. A higher sampling frequency allows for more precise control.

GenPar → MaxOut - Maximum Output, <REAL>

Is used to define the upper limit of the ControlSignal *CtrlOut* to a specific value. We recommend to set this limit at 100. This setting essentially represents the highest level of opening for the servo or proportional valve, expressed as a percentage in one direction.

GenPar → MinOut - Minimum Output, <REAL>

Is used to define the lower limit of the ControlSignal *CtrlOut* to a specific value. We recommend to set this limit at -100. This setting essentially represents the highest level of opening for the servo or proportional valve, expressed as a percentage in another direction.

Recommendation: When using a linear servo or proportional valve, it's advised to restrict the control signal to a range between -100 and 100. This range signifies the valve's opening percentage in either direction. Subsequently, these percentage values must be translated to the valve's physical input, which might be represented in current or voltage terms. In figure 3 and 4 are common mappings from output to valve input shown.

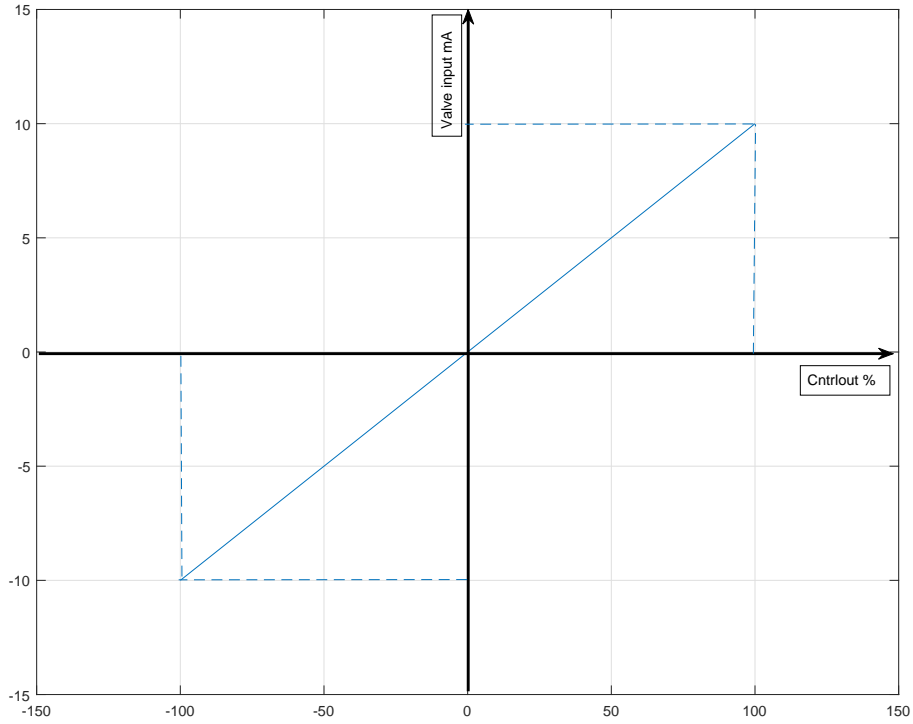


Figure 3: CtrlOut against valve input -10mA to 10mA

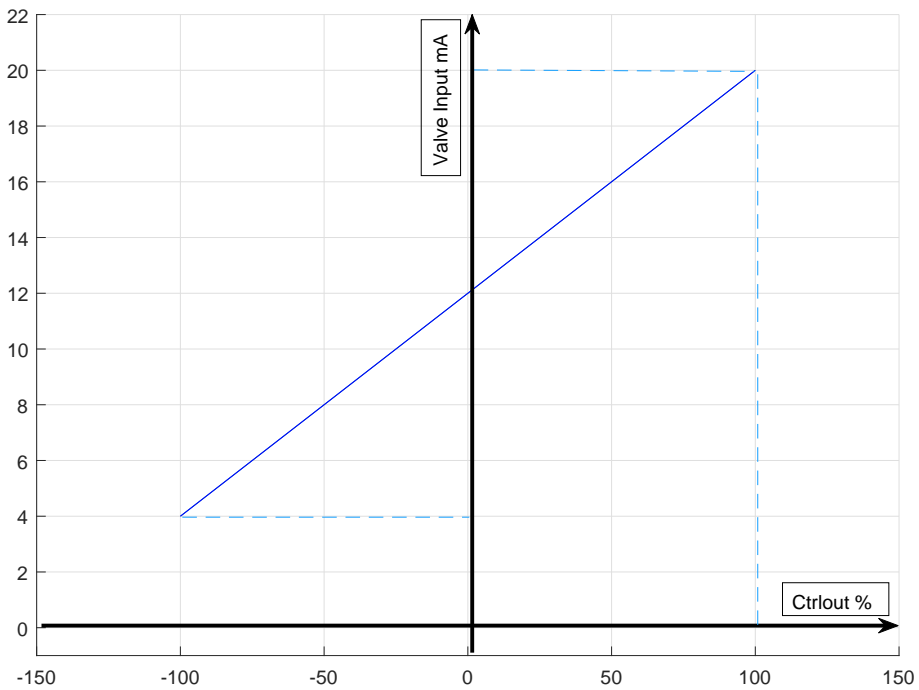


Figure 4: CtrlOut against valve input 4mA to 20mA

PrcssVal	
<i>ActPos</i>	<REAL>
<i>ActPr</i>	<REAL>
<i>HighPr</i>	<REAL>
<i>LowPr</i>	<REAL>

PrcssVal → ActPos - Cylinder Actual Position, <REAL>

The input represents the current measured position of the hydraulic cylinder. It is used in the error signal calculation of the internal controller. It should be provided in the same unit as *TgtPos*.

The measured position value provided to *ActPos* must be sampled at a rate at least as fast as the cycle time of the Cycle Interrupt Task in which the function block operates. This requirement is critical for real-time position control.

PrcssVal → ActPr - Actual pressure, <REAL>

Actual Pressure value measured in bar on the piston side of the hydraulic cylinder. It is used to adapt the dynamics of the position controller to the load borne by the hydraulic axis. This ensures that the hydraulic axis exhibits consistent control behavior regardless of the load it carries.

Important: If the hydraulic axis's pressure is not being measured, enter a default value of -1 in the *ActPr* input. This action will effectively deactivate the pressure-dependent dynamic calculation of controller.

PrcssVal → HighPr - High Pressure Line Value, <REAL>

Input the high-pressure line value of the valve in bar. This is not a measured value but a fixed constant.

PrcssVal → LowPr - Low Pressure Line Value, <REAL>

Input the Low-pressure line value of the valve in bar. This is not a measured value but a fixed constant.

PosTrjPar	
<i>MaxAccSetPnt</i>	<REAL>
<i>JerkSetPnt</i>	<REAL>

PosTrjPar → MaxAccSetPnt - Maximum Acceleration Setpoint, <REAL>

This input determines the maximum allowable rate of change in velocity for the position trajectory. It is measured in the same unit as *TgtPos* per second squared. This parameter needs to be customized based on the specific system requirements. *MaxAccSetPnt* plays a critical role in regulating the acceleration of the hydraulic cylinder as it follows the trajectory towards the target position. See figure 7.

PosTrjPar → JerkSetPnt - Maximum Jerk Setpoint, <REAL>

JerkSetPnt controls the abruptness or smoothness of motion by regulating the rate of change of acceleration. It is measured in the same unit as *TgtPos* per second cubed. Adjusting *JerkSetPnt* allows users to customize the motion profile according to their desired level of abruptness or smoothness. For example, if the maximum acceleration needs to be achieved within half a second, the *JerkSetPnt* value should be set to double the maximum acceleration value. See figure 7.

PosCntrlPar	
<i>KpPos</i>	<REAL>
<i>MaxKpAmp</i>	<REAL>
<i>KiPos</i>	<REAL>
<i>ModelIntCntrl</i>	<BOOL>
<i>GainFwdSpdCtrl</i>	<REAL>
<i>GainBwdSpdCtrl</i>	<REAL>

PosCntrlPar → KpPos - Controller P-Factor, <REAL>

KpPos represents the amplification factor of the P-controller within the position controller. A higher *KpPos* value can enhance control accuracy. However, caution is advised: setting the *KpPos* value excessively high may render the closed-loop system unstable. For optimal results and to maintain system stability, it's recommended to initiate with a modest *KpPos* value and incrementally adjust upwards to achieve the desired precision. See example 2 for optimal adjustment.

PosCntrlPar → MaxKpAmp - Maximum Kp Amplification, <REAL>

It relates to the P-controller (Position controller) within the function block, which is capable of dynamically altering the constant *KpPos* factor. This adjustment ensures that the hydraulic cylinder exhibits consistent control behavior regardless of the load it bears. The dynamic value of the *KpPos* factors is calculated based on the pressure on the piston side of the cylinder. If the cylinder lacks a pressure sensor, the controller operates with a constant *Kp* value. Through the *MaxKpAmp* input, users can set the upper limit of the amplification for this constant *Kp* value. We recommend setting it between 4 and 5.

Important: If the hydraulic cylinder's pressure is not being measured, enter a default value of -1 in the *ActPr* input. This action will effectively deactivate the pressure-dependent dynamic calculation of *Kp*.

PosCntrlPar → KiPos - Controller I-Factor, <REAL>

KiPos stands for the amplification factor of the I-controller within the position controller. A *KiPos* value set to zero effectively deactivates the I-controller. While a higher *KiPos* value results in swifter error integration, enhancing overall accuracy, it can also introduce increased oscillations in the closed-loop system. For best performance, it's advisable to commence with a low *KiPos* value and gradually increase it until the desired accuracy level is reached, balancing precision with system stability. See example 2 for optimal adjustment.

PosCntrlPar → ModelIntCntrl - Mode Selection for internal Integral Controller, <BOOL>

This input determines the behavior of the I-Controller. When set to false, the I-Controller remains continuously active. If set to true, the I-Controller operates only in the steady state, deactivating during the hydraulic axis movement. We advise setting this input to true, as allowing the I-Controller to function solely in the steady state often provides superior stability and accuracy for hydraulic axes. This approach typically minimizes, if not eliminates, significant overshoots and, by enabling an increase in the *Ki* factor, greatly enhances accuracy.

PosCntrlPar → GainFwdSpdCtrl - Positive Speed Feedforward, <REAL>

GainFwdSpdCtrl is a system parameter that adds a constant value for positive velocities to the control signal, improving trajectory tracking. It reduces the error signal and facilitates smoother following of the desired trajectory. The optimal positive feedforward value depends on the hardware and system dynamics, which can be determined through measurement and analysis, as illustrated in example 2.

PosCntrlPar → GainBwdSpdCtrl - Negative Speed Feedforward, <REAL>

GainBwdSpdCtrl is a system parameter that adds a constant value for negative velocities to the control signal, improving trajectory tracking. It reduces the error signal and facilitates smoother following of the desired trajectory. The optimal positive feedforward value depends on the hardware and system dynamics, which can be determined through measurement and analysis, as illustrated in example 2.

Enb - Turn controller On/Off, <BOOL>

This input controls the activation or deactivation of the function. When the input signal is turned off, the function no longer responds to changes in the setpoint and does not generate any trajectory. The monitoring signal *PosTrj* reflects the current position, while the other monitoring signals (*SpdTrj*, *AccTrj*, and *Busy*) are set to zero. In addition, when the function is deactivated, the output *CtrlOut* is set to zero, indicating that there is no valve movement caused by the controller.

TgtPos - Position setpoint, <REAL>

The input allows users to specify the desired target position for the hydraulic cylinder. It can be defined in any unit appropriate for the application (e.g., meters, centimeters, millimeters, micrometer). When *TgtPos* recognizes a change, the function generates a trajectory from the current position to the specified target. The internal controller ensures that the cylinder follows this trajectory accurately. When the function block is deactivated, changes to this input is without function.

Important: It's essential to understand that any changes to the *TgtPos*, *MaxSpdSetPnt*, *MaxAccSetPnt* and *JerkSetPnt* during axis movement are internally ignored by the block. The block responds to changes only when *PosTrj* matches *TgtPos*. The *Busy* output serves as an indicator of the block's readiness: When the *Busy* output of the block is logically true, it indicates that the cylinder is in motion, and the controller will not respond to new setpoints. Conversely, if it is logically false, the cylinder is in a steady state, allowing the controller to react to new setpoints. See figure 5.

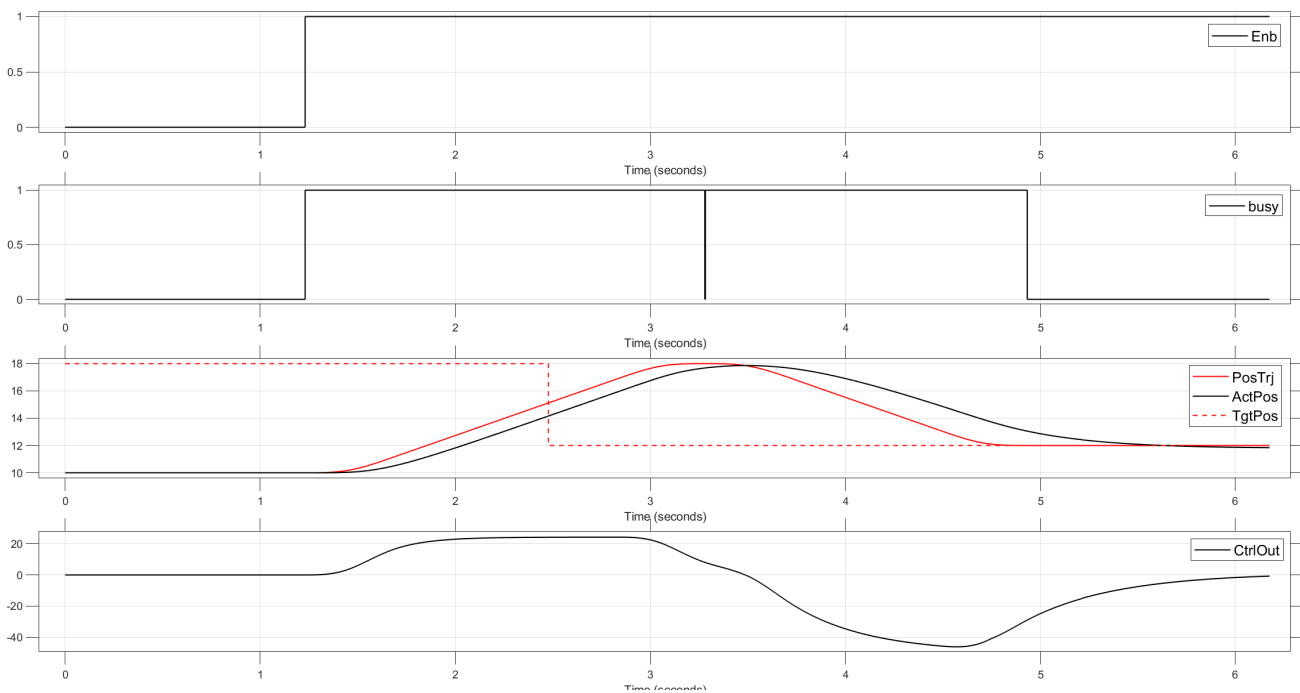


Figure 5: General behavior of the position controller

MaxSpdSetPnt - Maximum speed setpoint, <REAL>

This input sets the maximum speed allowed for the position trajectory generated by the internal generator. It is defined in the same unit as the *TgtPos* input, per second. This parameter ensures that the hydraulic cylinder moves at a controlled speed while reaching the target position defined by *TgtPos*. This sets the top speed for the hydraulic axis during movement. However, factors like distance, acceleration *MaxAccSetPnt*, and jerk *JerkSetPnt* might make it go slower than this speed. See Figure 6.

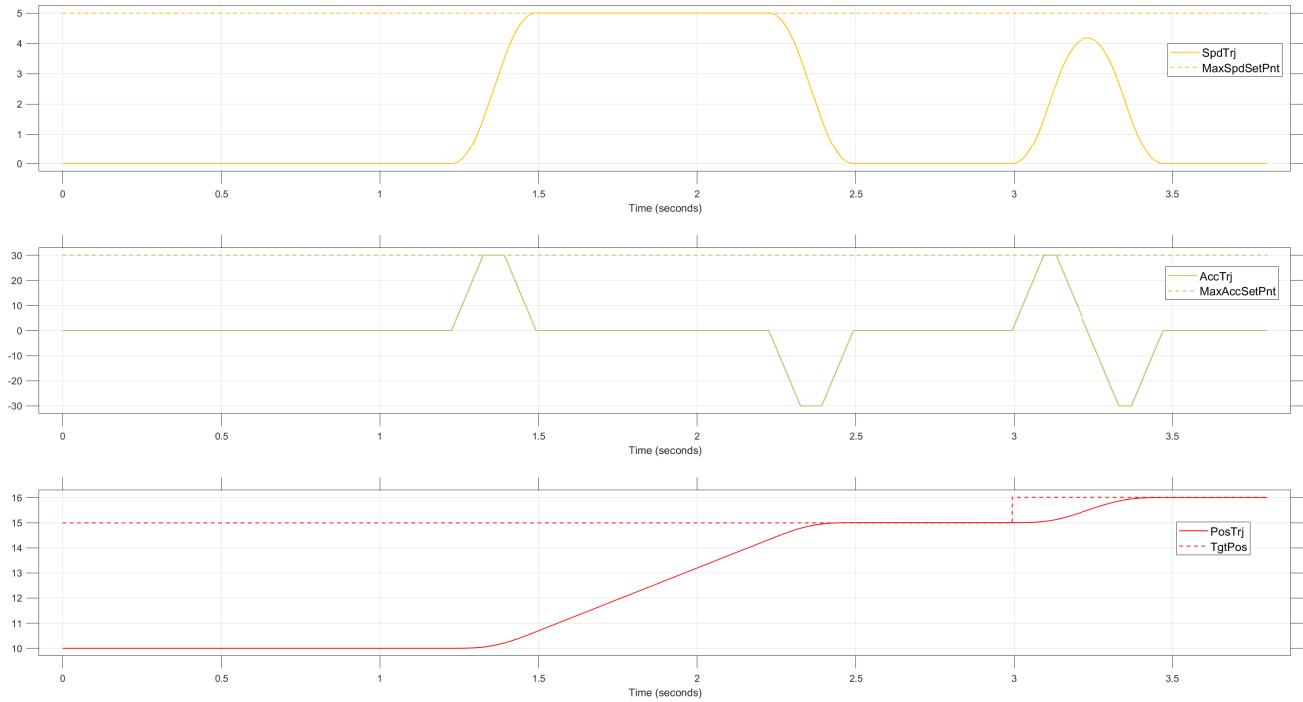


Figure 6: General behavior of the input *MaxSpdSetPnt*

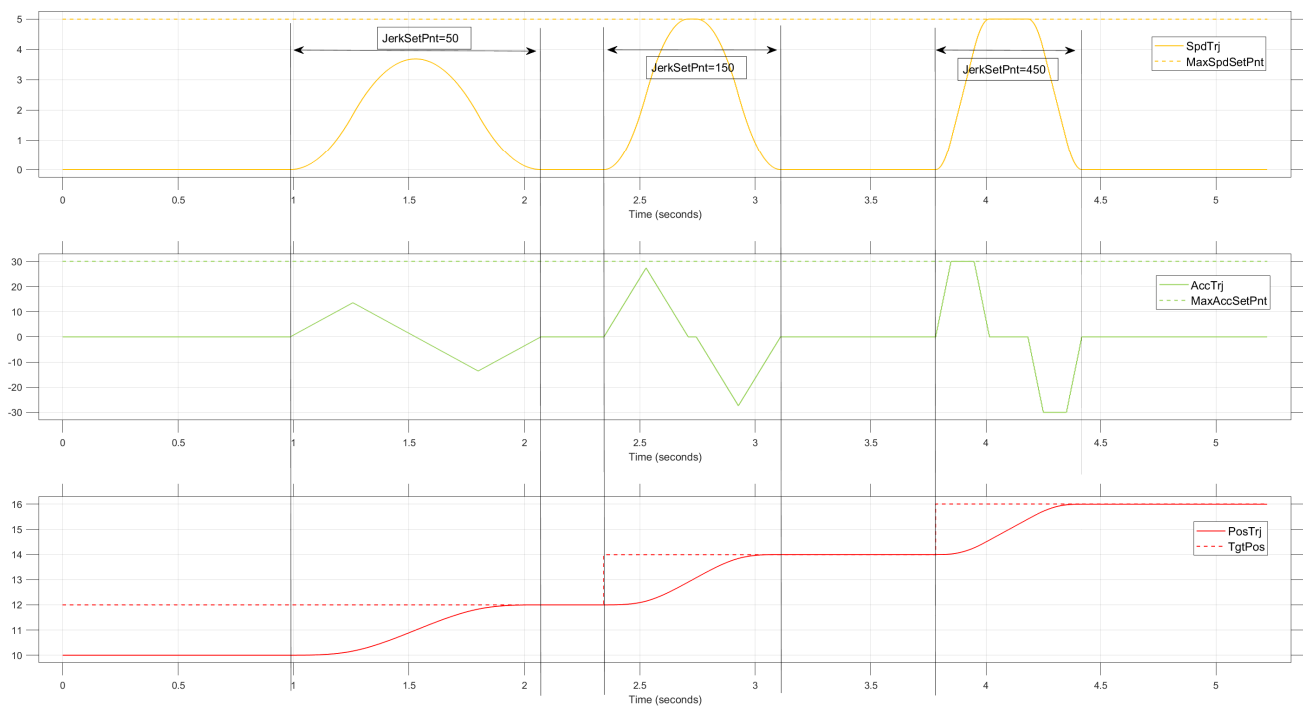


Figure 7: General behavior of the input *JerkSetPnt*

Offset - Displacement of ControlSignal, <REAL>

This input serves as a means of adjusting the control signal in both the activated and deactivated states of the controller. When the controller is enabled, the value is added to the control signal, allowing for an additional offset. This enables manual control of the cylinder's position in conjunction with the controller's output. In the deactivated state, where the controller output is fixed at zero, the input can be used as a direct means of manual control, independently influencing the cylinder's behavior. It provides a convenient way to switch between automatic and manual operation of the cylinder.

Outputs

CtrlOut - Control Signal, <REAL>

This is the primary output of the block, indicating the valve's opening value. When confined within the range of -100 and 100 , this value denotes the valve's opening percentage in one or another direction. It is essential to map this percentage to the valve's physical input, which may be quantified in terms of current or voltage.

PosTrj - Monitoring Signal: Position Trajectory, <REAL>

The *PosTrj* output provides information about the desired position trajectory that the hydraulic cylinder should follow. It represents the path from the current position (*ActPos*) to the target position (*TgtPos*). This output is mainly used for monitoring and visualization purposes, allowing users to observe the planned trajectory of the cylinder's movement.

Note: If the *Enb* input is set to false, the *PosTrj* output will match the current actual position of the hydraulic axis.

SpdTrj - Monitoring Signal: Trajectory velocity, <REAL>

The *SpdTrj* output represents the velocity trajectory which the hydraulic cylinder should follow. It starts and ends at zero and is limited by the *MaxSpdSetPnt* input. This output is primarily used for monitoring purposes.

Note: If the *Enb* input is set to false, the *SpdTrj* value will default to zero.

AccTrj - Monitoring Signal: Trajectory acceleration, <REAL>

The *AccTrj* output represents the acceleration which the hydraulic cylinder should follow. It starts and ends at zero and is limited by the *MaxAccSetPnt* input. This output is primarily used for monitoring purposes.

Note: If the *Enb* input is set to false, the *AccTrj* value will default to zero.

Busy - Monitoring Signal: trajectory is running, <BOOL>

This output reflects the block's responsiveness to new setpoints. If the hydraulic cylinder is moving, the *Busy* output becomes logically true, indicating that the block is not responding to changes in setpoints. The block resumes responsiveness when *PosTrj* aligns with *TgtPos*, at which point the *Busy* output switches to false, signaling readiness to accept setpoint modifications. See Figure 8

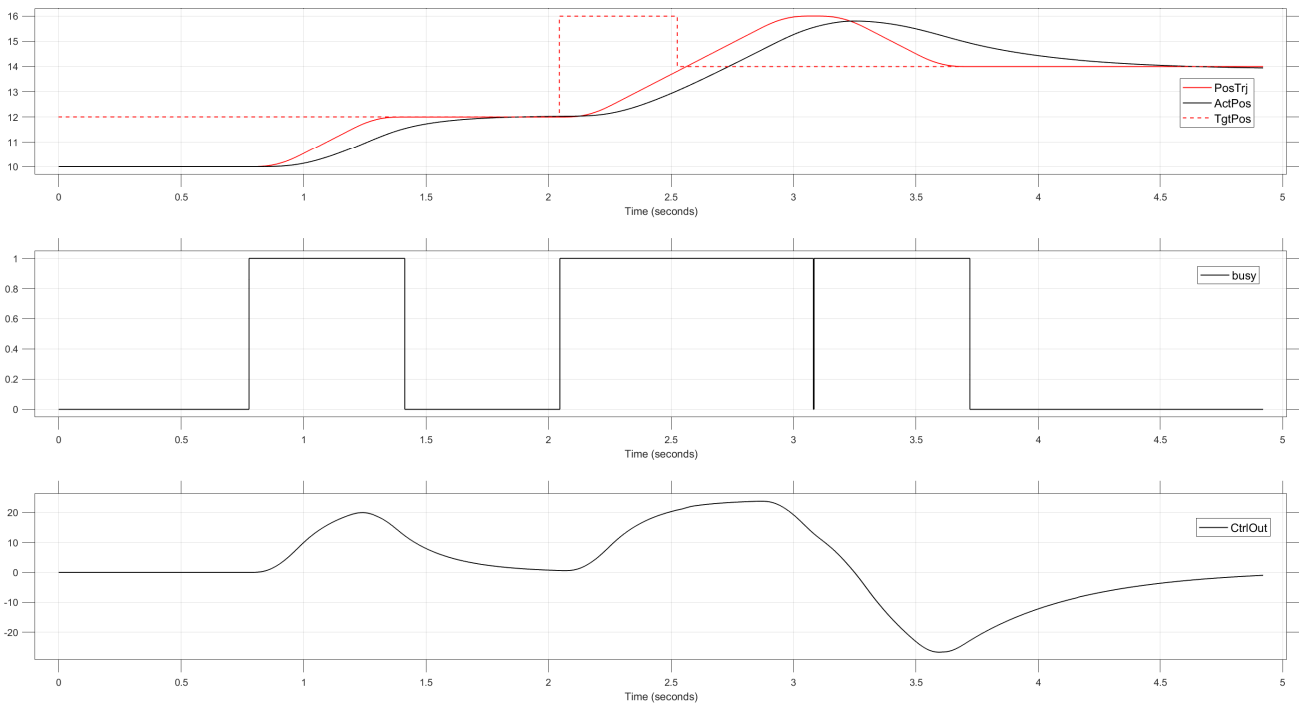


Figure 8: Output of the monitoring signal *Busy*

FAQ

How do I stop the hydraulic cylinder during its movement?

To halt the axis while it's moving, simply set the *Enb* Input to logical false. Doing this makes the *CtrlOut* immediately drop to zero, leading to the valve closing and consequently stopping the movement. For a more detailed description, see Example 3.

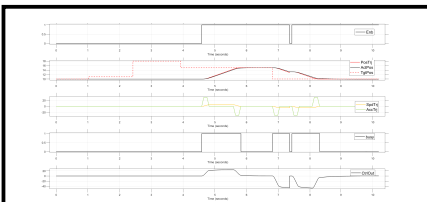
Can I input a new position target while the axis is still in motion?

If you wish to assign a new target position (*TgtPos*) during the axis's operation, you must first halt the movement using the *Enb* input. After stopping, wait for a specific delay (a few tens to hundreds of milliseconds depending on the specific axis) to ensure the axis has fully settled. Once stationary, you can then input the new *TgtPos*.

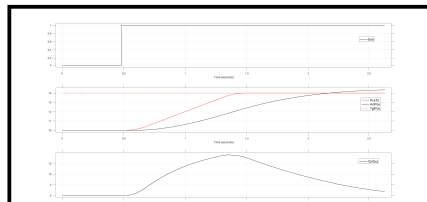
How can I effectively adjust the parameter *KpPos*, *KiPos*, *GainFwdSpdCtrl* and *GainBwdSpdCtrl*

For a detailed description, see example 2.

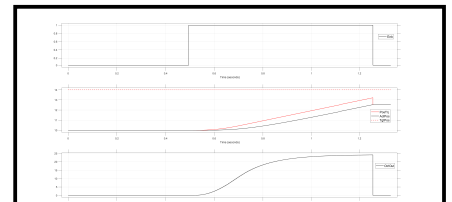
Examples



Example 1: General behavior of the controller
This example illustrates commonly occurring input variables.



Example 2: Setting control and feedforward Parameters
This example shows how you can adjust the control and feed-forward parameters experimentally.



Example 3: Stopping current movement
What inputs are needed to stop the current movement.

Important for all examples: These values are provided solely as examples to illustrate the approach for setting control parameters. Under no circumstances should these values be directly applied to your machine, not even as initial starting points. The control parameters must be explicitly adjusted for each machine individually.

General behavior

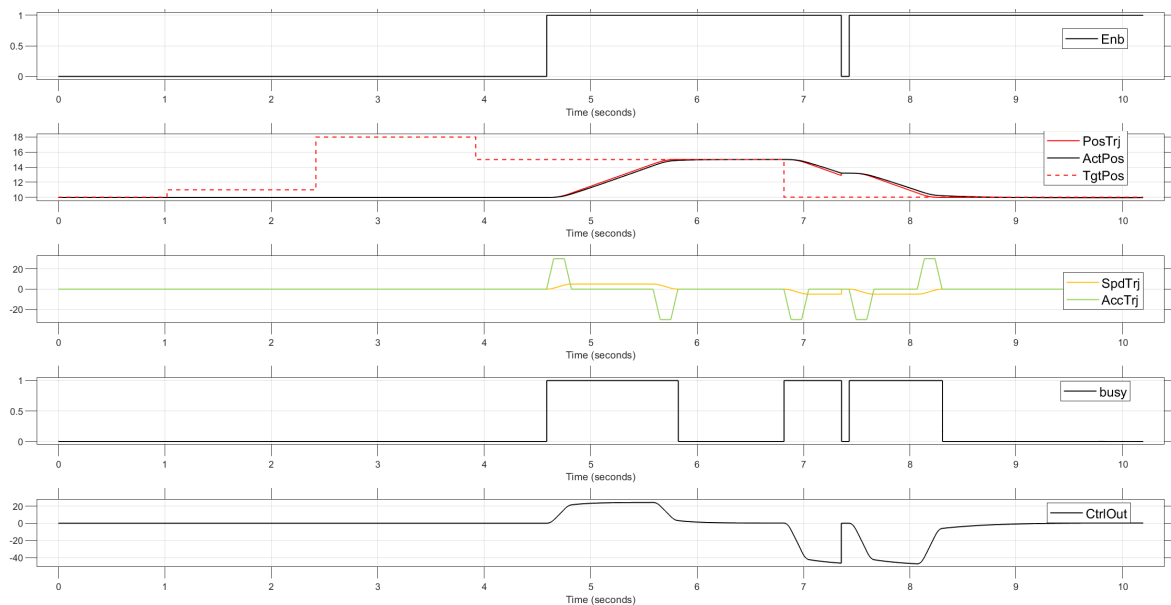


Figure 9: Reaction to several input combinations of enable and position setpoint

Figure 9 shows how the function reacts to different inputs of enable and the position setpoint. The highest plot shows the enable input. The second plot the position setpoint and the generated position trajectory which the cylinder has to follow. After that the corresponding speed and acceleration and the busy monitoring signal. The last plot shows the control input provided to the valve.

You can separate the plots into four time intervals.

1. from 0 to 4.5s: During this time interval, the *Enb* input is set to a state of logical false. This means that regardless of any setpoint provided for the position, the function remains unresponsive. There is no action or movement registered in this phase. Concurrently, the *Busy* signal, indicative of ongoing processing or movement, remains at a state of false as well. Essentially, this period represents an inactive state of the system.
2. from 4.5s to 5.8s: In the second time interval, the *Enb* input transitions to a logical true state, activating the function. Upon activation, the system generates a position trajectory, ensuring that both the speed and acceleration do not surpass specific predefined limits. A controller is engaged to ensure the cylinder accurately follows this trajectory. Throughout this motion, the *Busy* signal is set to true, indicating an active operation. Once the cylinder reaches its target position, the *Busy* signal reverts back to false, signaling the completion of the movement.
3. from 5.8s to 7.4s: In the third time interval, an interruption is observed when the *Enb* input is set to false in the midst of a movement. As a consequence, the position of the cylinder becomes static, holding its value achieved up to that moment. Both the speed and acceleration trajectories are promptly set to zero, halting any ongoing motion dynamics. The immediate response to this change is also reflected in the *Busy* signal, which transitions from true back to false, denoting a paused or inactive state of the operation.
4. from 7.4s to 10s: In the fourth time interval, the *Enb* input is re-activated by setting it to true. In response, the system generates a new position trajectory starting from the previously held constant value, established by the deactivation of *Enb*, moving towards the originally intended target. Corresponding speed and acceleration trajectories are also formulated to align with this new position trajectory. During this movement process, the *Busy* signal is activated, taking a value of true. It remains in this state until the cylinder successfully reaches its destination, after which the *Busy* signal returns to false, indicating the end of the operation.

Setting control and feedforward Parameters

Begin by initializing all parameters $KpPos$, $KiPos$, $GainFwdSpdCtrl$ and $GainBwdSpdCtrl$ to zero. Start the tuning process with the $KpPos$ value by setting it to a minimal level. Then, specify a position target using the $TgtPos$ and enable the controller. Ideally, adjust $KpPos$ so that, during the movement the actual position runs parallel with the position trajectory output ($PosTrj$) at certain moments. If $KpPos$ is too low, the actual position might never gets paralleled with the position trajectory.

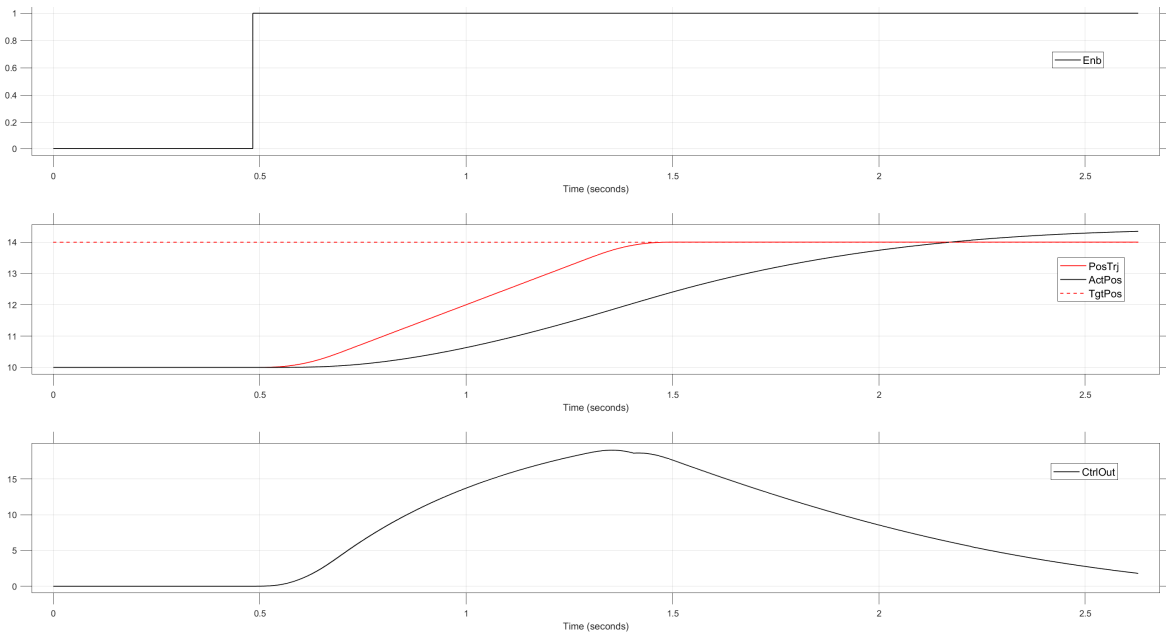


Figure 10: Reaction of the controller with the parameters $KpPos = 10$; $KiPos = 0$; $GainFwdSpdCtrl = 0$; $GainBwdSpdCtrl = 0$ and $GainAccCtrl = 0$.

In Figure 10, it is observable that during the movement, the $ActPos$ and $PosTrj$ are never parallel. This indicates that the $KpPos$ value is too small.

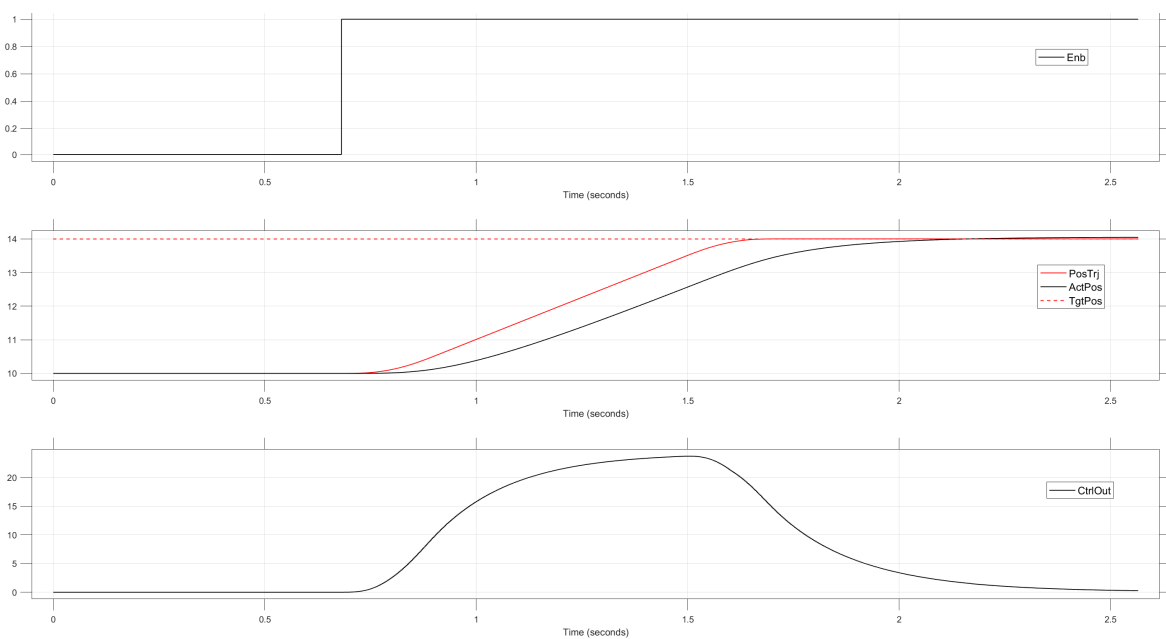


Figure 11: Reaction of the controller with the parameters $KpPos = 25$; $KiPos = 0$; $GainFwdSpdCtrl = 0$; $GainBwdSpdCtrl = 0$ and $GainAccCtrl = 0$.

According to Figure 11 when the $KpPos$ is increased to 25, the $PosTrj$ and $ActPos$ now run parallel during movement, indicating that the $KpPos$ has been optimally adjusted for the current conditions.

Setting $GainFwdSpdCtrl$ and $GainBwdSpdCtrl$: After setting the $KpPos$ value, read the value of $CtrlOut$ at any point where $ActPos$ and $PosTrj$ run parallel during movement and divide it by $MaxSpdSetPnt$. Enter the resulting value into the $GainFwdSpdCtrl$ input. (See Figure 12)

$$GainFwdSpdCtrl = \frac{CtrlOut_{t=2.28s}}{MaxSpdSetPnt_{t=2.28s}} = \frac{21.85}{5} = 4.37$$

Repeat the same process when retracting the cylinder, without changing the $KpPos$ value. Divide $CtrlOut$ by $MaxSpdSetPnt$ and enter this result into the $GainBwdSpdCtrl$ input. (See Figure 12)

$$GainBwdSpdCtrl = \frac{CtrlOut_{t=5.04s}}{MaxSpdSetPnt_{t=5.04s}} = \frac{34.15}{5} = 6.83$$

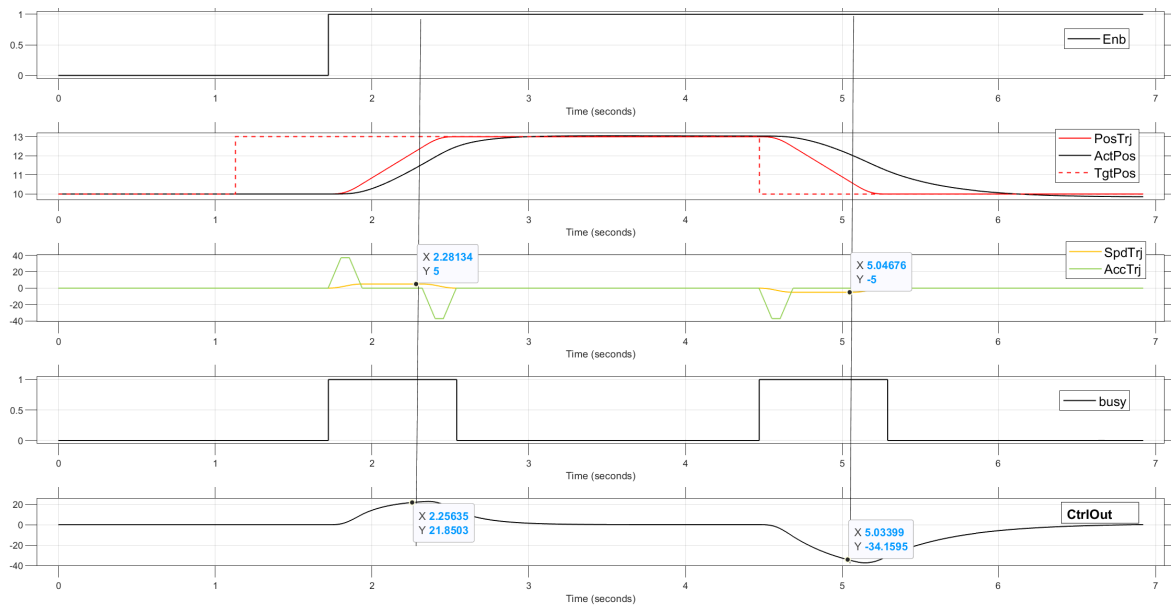


Figure 12: Adjusting the feedforward parameters based on the system's response to a control input.

Setting $KiPos$: For setting $KiPos$, we suggest setting the $ModelIntCntrl$ input to one, ensuring that the I-controller is active only in steady states. Then, observe the persistent control deviation of the position and incrementally increase $KiPos$ until you are satisfied with the controller's accuracy.

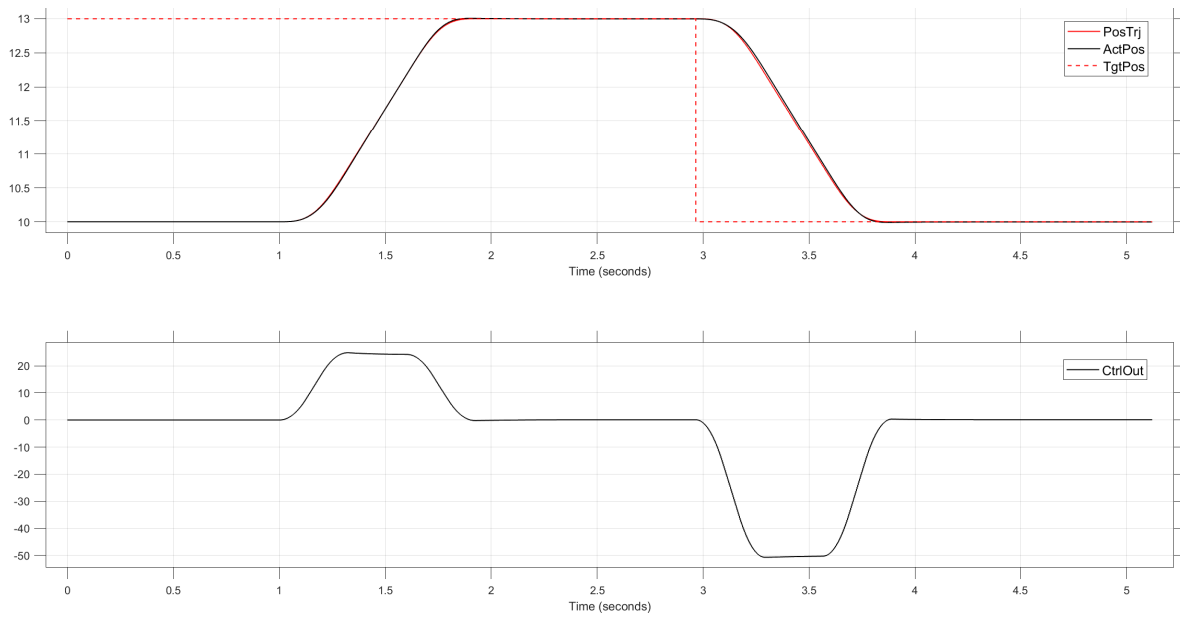


Figure 13: System behavior with well-adjusted parameters.

When all parameters are adjusted as described, the system should be able to closely follow the desired trajectory, as illustrated in Figure 13.

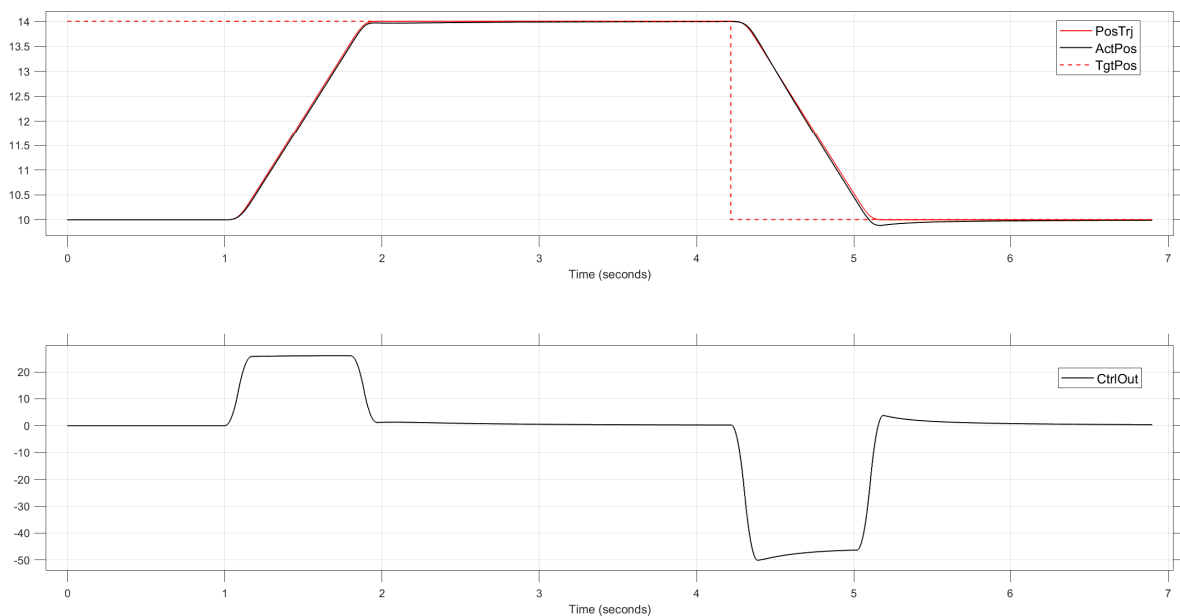


Figure 14: Overshoot despite proper parameter adjustment.

Note: If, despite proper parameter adjustment, overshooting is observed in the control behavior (as shown in Figure 14), it is an indication that the dynamics of the position trajectory are faster than your machine's maximum capability. Therefore, you should reduce the *MaxAccSetPnt* and *JerkSetPnt* parameters.

Stopping the current movement

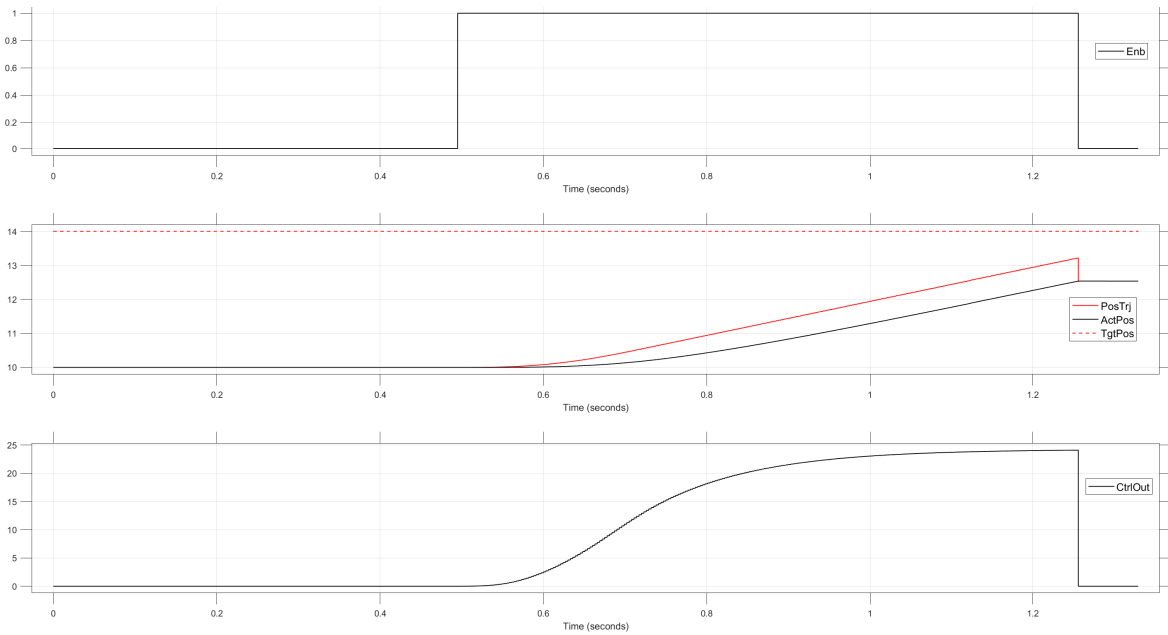


Figure 15: Controller response when the *Enb* input is set to logical 0.

As seen in Figure 15, the controller follows the target position *TgtPos* upon activation of the *Enb* input. The slope angle is dependent on the maximum allowable speed, set with the *MaxSpdSetPnt* input. To terminate the motion, the *Enb* must be deactivated again.