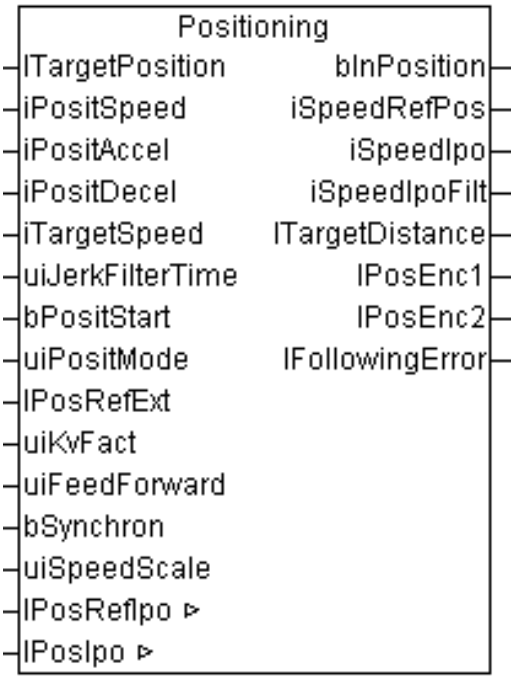
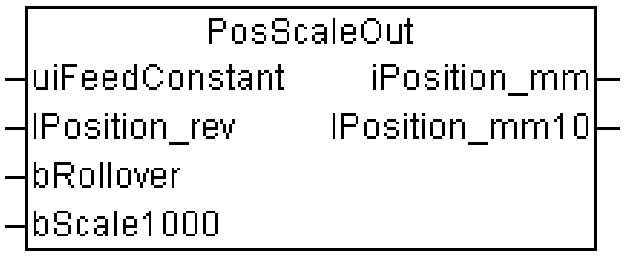
DCS800

Position Control Library

Description of DCS800 position control function blocks





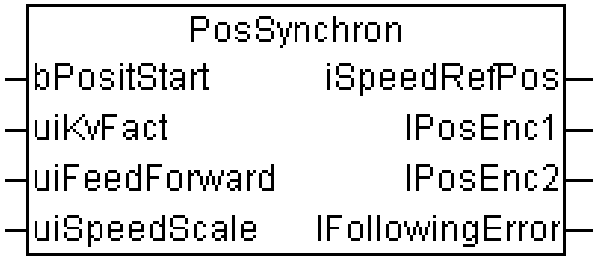


Table of contents

Table of contents **2**

Position Control with DCS800 **3**

Chapter overview ... 3
Compatibility ... 3
Safety instructions ... 3
Reader ... 3

What is positioning? **5**

Synchronous motion ... 5
Speed synchronous ... 5
Angular synchronous ... 5
Control structure ... 6
Homing ... 6

Limitations of the DCS800 **7**

General ... 7
Accuracy ... 7

Position Control Library **9**

DriveInterface ... 10
JerkFilter ... 12
MulDivLim ... 13
Positioning ... 14
PositParam ... 17
PosScaleIn ... 19
PosScaleOut ... 20
PosSynchron ... 21
VirtualMaster ... 22

Build a Position Control Application **23**

Function blocks for positioning ... 23
Create the project in CoDeSys ... 23

Typical Applications **25**

Electrical shaft ... 25
Explanation ... 25
CoDeSys program ... 26
Linear positioning ... 29
Explanation ... 29
CoDeSys program ... 31

Commissioning hints **33**

Basic drive system ... 33
Autotuning of Controllers ... 33
Parameterization of DCS800 firmware ... 33
Position control system ... 34
Check position control loop ... 34
Linear motion check ... 34

Position Control with DCS800

Chapter overview

The chapter gives an overview about opportunities for position control with DCS800.

Compatibility

ABB offers a library with special function blocks for positioning.
The “Position Control Library” works only with the DCS800 standard firmware version >=3.30.
It is recommended to use always the newest *DCS800 library* for positioning applications.
This will be explained later on!

Safety instructions

WARNING! The safety instructions for the appropriate DCS800 DC Drive model must be studied carefully before doing program modifications or starting any work with the unit.
Neglecting the safety instructions may cause damage to the equipment, physical injury or death.

Especially by doing software applications, read also the software function specific warnings and notes before changing the default settings of the function. For each function, the warnings and notes are given in the Firmware Manual in the subsection describing the related user-adjustable parameters.
Take care that the developed application will not constrain safety functions like E Stop, Off or like monitoring functions.

Reader

The reader of the manual is expected to:

- Know the standard electrical wiring practices, electronic components and electrical schematic symbols.

What is positioning?

Synchronous motion

In a couple of machines and plants there is the requirement that two or more shafts have to work with the same speed or angular synchronism.

Examples:

- Canting of crane chassis must be avoided
- Main drive and feeder drive in a machine tool have to work angular synchronism
- Flying sheers need a synchronism between the cross cutter and the material feeder

In old plants this jobs are solved by mechanical constructions.

By coupling of two or more converters it's possible to solve this synchronisation in an electronic way.

Examples:

- Gear box
- Electrical shaft (upright shaft)
- Synchronous applications (speed or angle)
- Flying shear

All this applications are possible by electrical coupling of drives.

For this applications in minimum one encoder is needed which gives position values to the software.

Speed synchronous

Speed synchronous means that two machines work with the same speed. A position loop isn't needed for this application.

Angular synchronous

Angular synchronous is a construction with a position loop because 1 revolution of motor 1 should also cause 1 revolution of motor 2 for example. The speed control loop in this configuration is subordinated to position control loop.

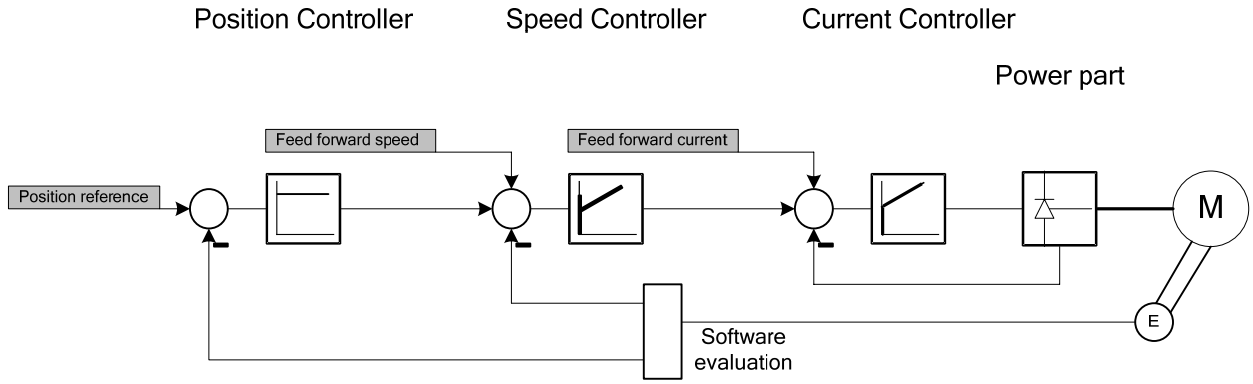
Control structure

A motion control system exists of the following controllers:

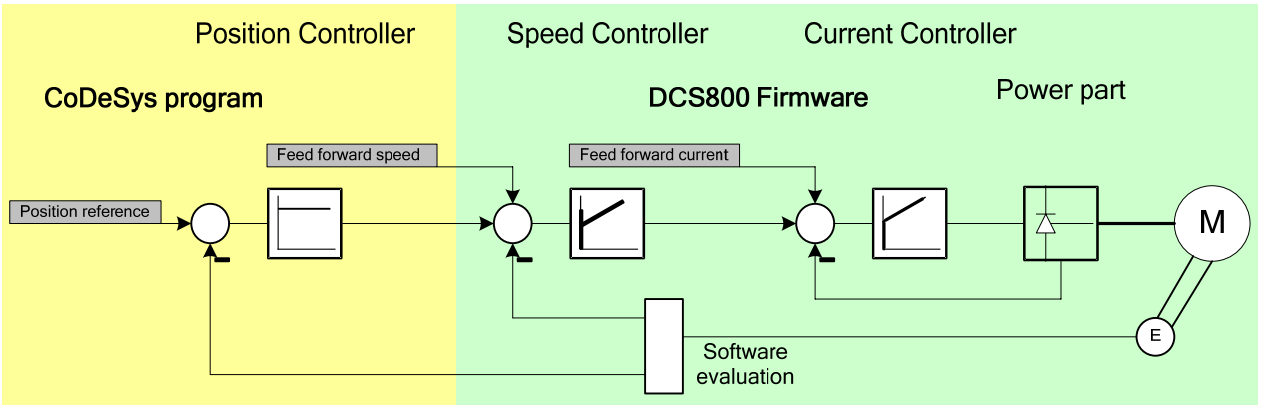
- Current controller (PI)
- Speed controller (PID)
- Position controller (P)

Control process is mostly integral containing.
Therefore the position controller should be only a P-Controller.

Control structure for position control:



Combination of DCS800 firmware and CoDeSys application program:



Homing

There are several opportunities to implement *Homing* functionality.

Please use function blocks *PosCorrect* and *PosLatch* of **DCS800 Drive Library** for cyclic referencing or to set a position reference e.g. from an overriding control system.

Limitations of the DCS800

General

Basically positioning with a line-commutated converter is possible if the response time isn't too fast. DCS800 executes current- and speed-controller in maximum 3.3 ms.
The position control loop can be calculated in fastest cycle time of DCS800 Control Builder (CoDeSys) with 5 ms.
This limits the response time of the position loop.

Don't expect a response time like with servo drives!

Accuracy

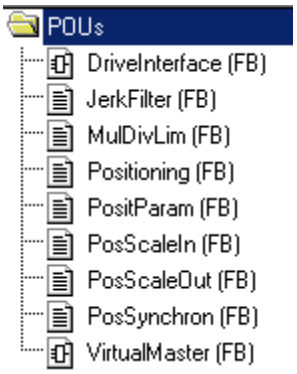
Accuracy in a position control loop depends on many indicators which have to be considered before doing a general statement.

- Encoder
  The number of pulses should be sufficient. Also the connection to motor shaft or mechanic must be fixed.
- Mechanics
  A very important argument for accuracy is the coupled mechanic. All components must be connected fixedly. Slackly connections (backlash gearbox) cause inaccuracy and oscillations in the control system.
- Control process
  Accuracy depends drastic on the control process itself. Each control process has to be tested by a frequency analysis.

All this arguments should be checked. Otherwise there is no chance to talk about accuracy and fast responses.

Position Control Library

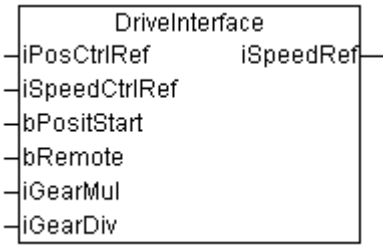
Name: **DCS800positioning.lib**
Version: **1.60**
Firmware: **3.3 or newer**



Note:
The DCS800 positioning library version 1.60 (DCS800positioning.lib) requires DCS800 drive library (DCS800lib.lib) version 1.80.

DriveInterface

Function block **DriveInterface** is used to handle the positioning reference and the emergency stop routine.



Name	LL	UL	Def	Scale	Unit	Type
iPosCtrlRef	-25000	25000	--	100% == 20000		INT
	Speed reference in position control mode					
iSpeedCtrlRef	-25000	25000	--	100% == 20000		INT
	Speed reference in speed control mode; can e.g. be taken from SpeedRef4 (2.18)					
bPositStart	FALSE	TRUE	FALSE	--	--	BOOL
	TRUE: position control mode FALSE: speed control mode					
bRemote	FALSE	TRUE	TRUE	--	--	BOOL
	FALSE: drive is in local control mode (panel or drive tool)					
iGearMul	-32767	32767	1			INT
	Nominator of the load gear function (see below)					
iGearDiv	-32767	32767	1			INT
	Denominator of the load gear function (see below)					
iSpeedRef	-32767	32767	0	100% == 20000		INT
	Resulting speed reference for monitoring purpose					

Interface block for **Positioning** and **PosSynchron** blocks.

Assumes speed reference selector P11.03 = SpeedRef2315 (18).

- Writes the position controller's speed reference output to speed reference value P23.15.
- Takes into account a load gear factor.
- Initialises the speed ramp to speed reference of position control on activation of E-Stop.
- Initialises the speed ramp to speed reference of position control on switchover to speed control mode.

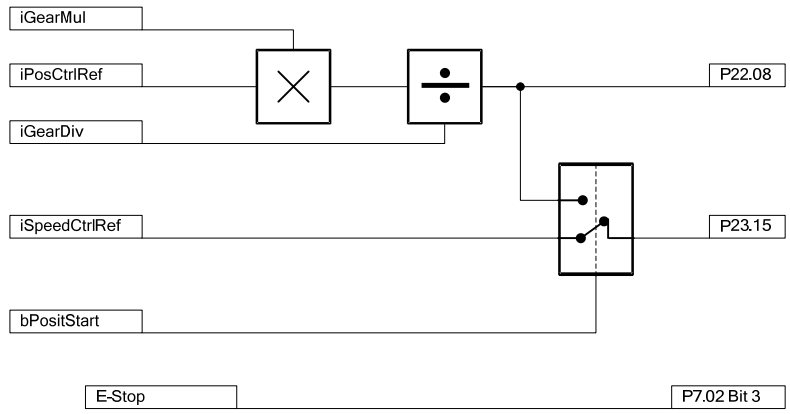
Note:
The function block writes to parameters *BalRampRef* (P22.08) and *AuxCtrlWord* (P7.02, only bit 3).

Load gear function:
If there is a gear ratio between the position control and the speed control (e.g. position control is load side related, speed control is motor shaft related), this ratio must be set (i.e. if the gear ratio is negative!):

$$\frac{\text{iGearMul}}{\text{iGearDiv}} = \frac{\text{Motor Nominal Speed}}{\text{iPosCtrlRef}}$$

Example:
speed feedback: encoder 1
position feedback: encoder 1
gear ratio from motor shaft to load = $\frac{-27}{123}$
Input *iLoadGearMul* of function block **PosSetGear**: -27
Input *uiLoadGearDiv* of function block **PosSetGear**: 123
iGearMul = 123
iGearDiv = -27
The resulting speed reference is *iPosCtrlRef* * $\frac{123}{-27}$

Circuit diagram:



If E-Stop is detected, firmware switches directly to E-Stop ramp

JerkFilter

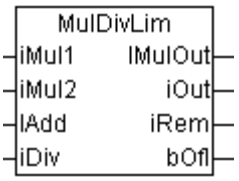
Function block **JerkFilter** is a sliding average filter i.e. for speed values. The filter length is variable and should be set as filter time. Maximum filter length is 2000 (10 seconds at 5 ms cycle time). The initialisation command starts a counter and copies the input value to an initialisation value. During the next [filter length] cycles this initialisation value is used instead of the filter array contents to calculate the output value. Changes of the filter length become active on the next initialisation command.

Name	LL	UL	Def	Scale	Unit	Type
iFilterInput	-32767	32767	--	--	--	INT
	Filter input value					
uiFilterLength	0	2000	--	--	--	WORD
	Filter length					
bInitFilter	FALSE	TRUE*	FALSE	--	--	BOOL
	Initialisation command					
iFilterOutput	-32767	32767	--	--	--	INT
	Filter output value					

Note:
This function block is used inside function block **Positioning**.

MulDivLim

Function block **MulDivLim** is equal to function block **MulDiv** with the difference that output *iOut* is limited between -32767 and 32767 instead of -32768.



$$iOut = \frac{iMul1 \cdot iMul2 + lAdd}{iDiv}$$ iOut: 16 bit

$$lMulOut = iMul1 \cdot iMul2$$ lMulOut: 32 bit

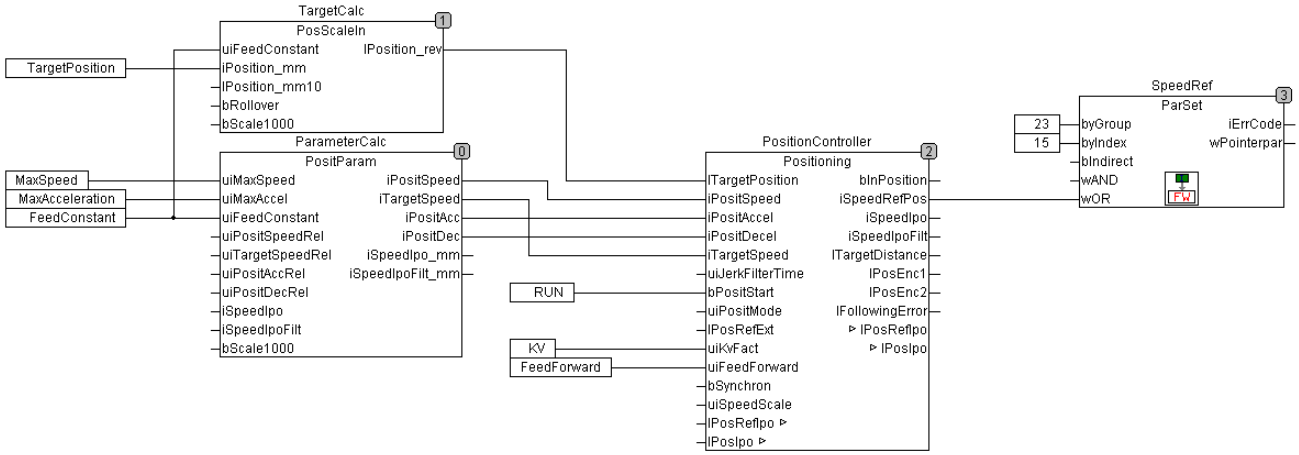
Name	LL	UL	Def	Scale	Unit	Type
iMul1	-32768	32767	0	--	--	INT
	Multiplication input 1 (16 bit)					
iMul2	-32768	32767	0	--	--	INT
	Multiplication value 2 (16 bit)					
lAdd	-2147483648	2147483647	0	--	--	DINT
	Add value as 32 bit number					
iDiv	-32768	32767	0	--	--	INT
	Divisor (16 bit)					
lMulOut	-2147483648	2147483647	0	--	--	DINT
	Result of multiplication as 32 bit number					
iOut	-32768	32767	0	--	--	INT
	Result of complete calculation as 16 bit number					
iRem	-32768	32767	0	--	--	INT
	Rest of division					
bOfl	FALSE	TRUE*	FALSE	--	--	BOOL
	Overflow flag. It is set if output iOut is in saturation / limitation.					

Note:
With firmware version 3.3 and later, function block **MulDiv** from DCS800 Drive Library includes also this function block.

Positioning

Function block **Positioning** is a positioning interpolator and position controller. It calculates with the input values speed, acceleration and jerk limitation the optimal distance to the new position. The output of this function block has to be connected to fast speed reference parameter from standard firmware (bypass speed ramp).

Example:



Description:
All physical values must be scaled for positioning. It is recommended to create a slow and a fast task cycle to save calculation time of the processor.

Fast task (5 ms cycle):

- Function block **Positioning**
- Write result to P23.15 *DirectSpeedRef*

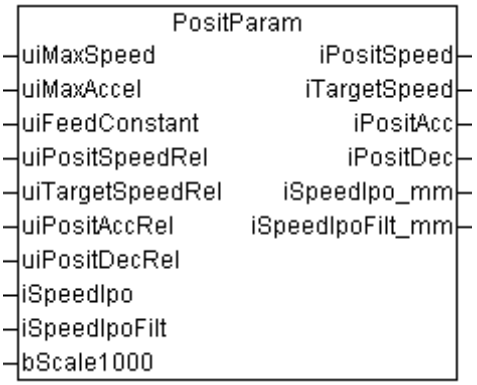
Slow task (20 ms cycle):

- Scaling between physical and internal / revolution values

Name	LL	UL	Def	Scale	Unit	Type
lTargetPosition	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Target position value					
iPositSpeed	0	32767	0	See below!	--	INT
	Positioning speed					
iPositAccel	0	32767	0	See below!	--	INT
	Positioning acceleration					
iPositDecel	0	32767	0	See below!	--	INT
	Positioning deceleration					
iTargetSpeed						
	Not implemented yet!					
uiJerkFilterTime	0	2000	0	See below!	--	WORD
	Jerk filter time. The nominal acceleration is generated within this time.					
bPositStart	FALSE	TRUE*	FALSE	--	--	BOOL
	Positioning start command. On the rising edge the interpolator is initialized with the drive's actual values.					
uiPositMode	0	3	0	--	--	ENUM
	Positioning mode: 0: Linear motion 1: Positive (rollover axis, positioning in positive direction) 2: Negative (rollover axis, positioning in negative direction) 3: Short (rollover axis, positioning on shortest path)					
lPosRefExt	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	External position reference, added to input of position controller					
uiKVFact	0	65535	0	1000 == 1000 / min		WORD
	Position gain					
uiFeedForward	0	10000	0	10000 == 100 %		WORD
	Factor for speed feed forward					
bSynchron	FALSE	TRUE*	FALSE	--	--	BOOL
	* Positioning is relative to the encoder 2 position					
uiSpeedScale	0	32767	0	1 == 1 rpm		WORD
	Actual speed scaling of DCS800 firmware. Connect it to P2.29, please!					
lPosRefIpo	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Jerk limited interpolator position					
lPosIpo	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Interpolator position					
bInPosition	FALSE	TRUE*	FALSE	--	--	BOOL
	* Position interpolator reached the target position					
iSpeedRefPos	-32767	32767	0	--	--	INT
	Speed reference from the position control loop. 20000 == [P2.29]. This output is forced to zero if input bPositStart == False.					
iSpeedIpo	-32767	32767	0	See below!	--	INT
	Actual interpolation speed					
iSpeedIpoFilt	-32767	32767	0	See below!	--	INT
	Actual interpolation speed, filtered by jerk filter					
lTargetDistance	-2147483648	2147483647	0	--	--	DINT
	Distance between target position and interpolator position					
lPosEnc1	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Encoder 1 position (assuming P50.07 scaled of rollover)					
lPosEnc2	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Encoder 2 position (assuming P50.07 scaled of rollover)					
lFollowingError	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Following error					

PositParam

Function block **PositParam** is used to calculate the positioning parameters for function block **Positioning**. Input values are physical values which are converted to internal values because of more accuracy during calculation.



Example:
Motor speed maximum = 3000 rpm
Acceleration maximum = 0,1 m/sec²
Feed constant = 10 mm/revolution

Calculate parameters:

$$\text{uiMaxSpeed} = 10\frac{\text{mm}}{\text{rev}} \cdot 3000\frac{\text{rev}}{\text{min}} = 500\frac{\text{mm}}{\text{sec}}$$

$$\text{uiFeedConst} = \frac{10\,\text{mm}/\text{rev}}{0{,}1\,\text{mm}/\text{rev}} = 100$$

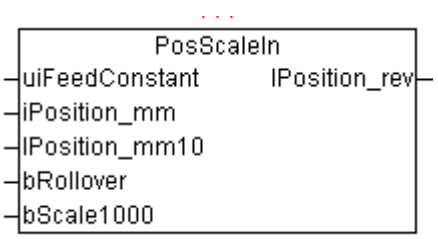
$$\text{uiMaxSpeed} = 0{,}1\frac{\text{m}}{\text{sec}^2} = 100\frac{\text{mm}}{\text{sec}^2}$$

Name	LL	UL	Def	Scale	Unit	Type
uiMaxSpeed	0	65535	0	1 == 1 mm / s		WORD
	Maximum positioning speed					
uiMaxAccel	0	65535	0	1 == 1 mm / s²		WORD
	Maximum positioning acceleration					
uiFeedConstant	0	32000	0	1 == 0.1mm / rev		WORD
	Feed constant (maximum 3200.0 mm)					
uiPositSpeedRel	0	32767	10000	10000 == 100 %		WORD
	Positioning speed in percent of *MaxSpeed*					
uiPositAccRel	0	32767	10000	10000 == 100 %		WORD
	Positioning acceleration in percent of *MaxAccel*					
uiPositDecRel	0	32767	10000	10000 == 100 %		WORD
	Positioning acceleration in percent of *MaxAccel*					
iSpeedIpo	-32767	32767	0	1 == 1000 rev / 5 * 65536 sec		INT
	Actual interpolation speed					
iSpeedIpoFilt	-32767	32767	0	1 == 1000 rev / 5 * 65536 sec		INT
	Actual filtered interpolation speed					
bScale1000	FALSE	TRUE*	FALSE			BOOL
	* scaling of feed constant = 0.001 mm					
iPositSpeed	-32767	32767	0	1 == 1000 rev / 5 * 65536 sec		INT
	Scaled positioning speed					
iTargetSpeed	-32767	32767	0	1 == 1000 rev / 5 * 65536 sec		INT
	Scaled target speed					
iPositAcc	-32767	32767	0	1 = 1000 * 1000 rev / 25 * 65536 * 32 sec²		INT
	Scaled positioning acceleration					
iPositDec	-32767	32767	0	1 = 1000 * 1000 rev / 25 * 65536 * 32 sec²		INT
	Scaled positioning deceleration					
iSpeedIpo_mm	-32767	32767	0	1 = 1mm/sec		INT
	Actual interpolation speed					
iSpeedIpoFilt_mm	-32767	32767	0	1 = 1mm/sec		INT
	Actual filtered interpolation speed					

PosScaleIn

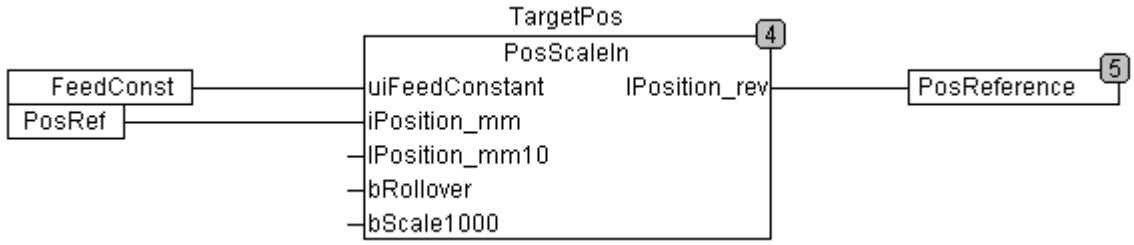
This function block scales position values from external scaling to internal scaling.

Internal scaling: 32-bit, 65536 = 1 revolution
External scaling: 16-bit, 1 = 1 mm
32-bit, 1 = 0.1 mm



Name	LL	UL	Def	Scale	Unit	Type
uiFeedConstant	0	32000	0	1 == 0.1 mm / revolution		WORD
	Feed constant, if bScale100 == FALSE					
iPosition_mm	0	32000	0	1 == 1 mm		WORD
	Position input value					
lPosition_mm10	0	32000	0	1 == 0.1 mm		WORD
	Position input value					
bRollover	FALSE	TRUE*	FALSE	--	--	BOOL
	* Rollover Axis					
bScale1000	FALSE	TRUE*	FALSE	--	--	BOOL
	* Scaling of feed constant = 0.001 mm					
lPosition_rev	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Position output value					

Example:



The position reference *PosRef* is the physical distance between point zero and the new point in millimetres. The block calculates by using the feed constant value like the following:

$$\text{PosReference} = \frac{\text{PosRef}}{\text{FeedConst}}$$

PosRef = 30 mm; FeedConst = 10 mm / rev

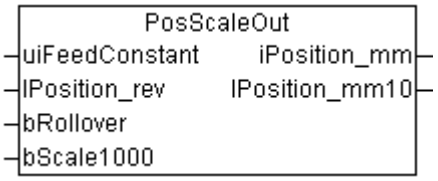
e.g. $\text{PosReference} = \frac{30\text{mm}}{100 \cdot 0{,}1\,\text{mm}/\text{rev}} = 3\text{rev}$

Note:
Position of this function block in the application program is mostly before position controller!

PosScaleOut

Function block **PosScaleOut** scales position values from internal scaling back to external scaling.

Internal scaling: 32-bit, 65536 = 1 revolution
External scaling: 16-bit, 1 = 1 mm
32-bit, 1 = 0.1 mm

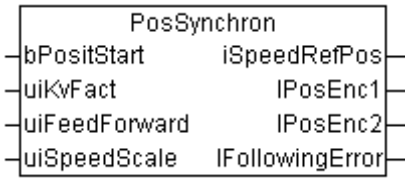


Name	LL	UL	Def	Scale	Unit	Type
uiFeedConstant	0	32000	0	1 == 0.1 mm / revolution		WORD
	Feed constant, if bScale1000 == FALSE					
lPosition_rev	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Position input value					
bRollover	FALSE	TRUE*	FALSE	--	--	BOOL
	* Rollover Axis					
bScale1000	FALSE	TRUE*	FALSE	--	--	BOOL
	* Scaling of feed constant = 0.001 mm					
iPosition_mm	-32767	32767		1 == 1 mm		INT
	Position output value					
lPosition_mm10	-2147483648	2147483647	0	1 == 0.001 mm		DINT
	Position output value					

Note:
This function block is exactly the opposite of **PosScaleIn**. It scales back from revolutions to physical millimetres. This can be used to calculate values which should be shown as actual signals.

PosSynchron

Function block **PosSynchron** is used as position controller for synchronous applications (via encoder 2).

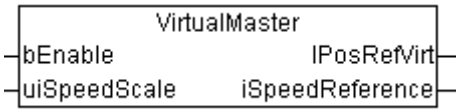


Name	LL	UL	Def	Scale	Unit	Type
bPositStart	FALSE	TRUE*	FALSE	--	--	BOOL
	Position controller start command. If FALSE, the speed reference is forced to zero.					
uiKvFact	0	65535	0	1000 == 1000 / min		WORD
	Position gain					
uiFeedForward	0	10000	0	10000 == 100 %		WORD
	Feed forward					
uiSpeedScale	0	32767	0	1 == 1 rpm		WORD
	Speed scaling: given speed is represented by speed value 20000					
iSpeedRefPos				20000 == [P2.29]		INT
	Speed reference from position control loop. Forced to zero, if input *bPositStart* = FALSE					
lPosEnc1	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Encoder 1 position					
lPosEnc2	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Encoder 2 position					
lFollowingError	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Following error					

Find an example in the next chapter!

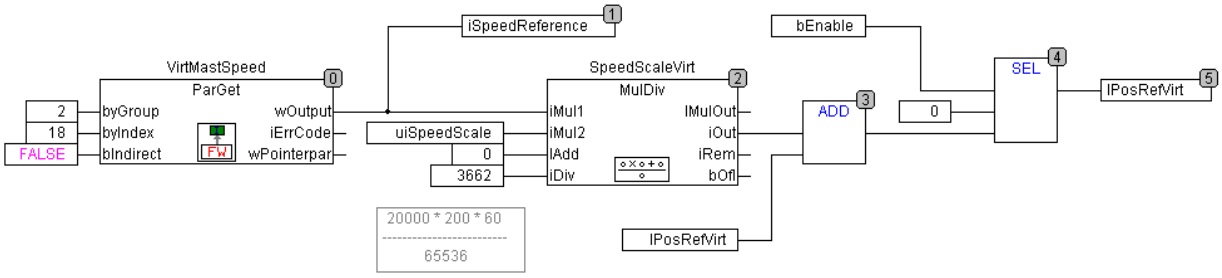
VirtualMaster

Function block VirtualMaster generates a position reference value from drive signal P2.18.



Name	LL	UL	Def	Scale	Unit	Type
bPositStart	FALSE	TRUE*	FALSE	--	--	BOOL
	TRUE: Enable position reference generation FALSE: Force position reference to zero					
uiSpeedScale	0	32000	1	1 == 1 rpm		INT
	Speed scaling: given speed is represented by speed value 2000.					
lPosRefVirt	-2147483648	2147483647	0	65536 == 1 revolution		DINT
	Position reference.					
iSpeedRef	-32767	32767	0	== P2.18		INT
	Speed reference value P2.18					

Function:



Build a Position Control Application

Positioning with DCS800 is only possible by using DCS800 Control Builder (CoDeSys). ABB DC Drives offers a library with special function blocks which are needed for positioning applications. Responsible for the functionality of the application program is always the commissioning engineer.

Function blocks for positioning

Some function blocks are included in the DCS800 positioning library and some can be found in the DCS800 drive library. Anyway the following libraries have to be embedded to CoDeSys:

- DCS800lib.lib version 1.80 (firmware 3.30) or newer
- DCS800positioning.lib version 1.50 or newer

Create the project in CoDeSys

Position control loops requires fast calculations to get an adequate accuracy. Therefore it is necessary to calculate the position controller and the write block to *P23.15 DirectSpeedRef* in the fastest task cycle by 5 ms.

Calculation of reference values rescaling or something else can be calculated much slower because they are not involved in a closed loop. These calculations can be done within 20 ms to save processor load for the important calculations.

It is recommended to check the actual load of each task cycle in CoDeSys *Task Manager* and the total application load in P4.22!

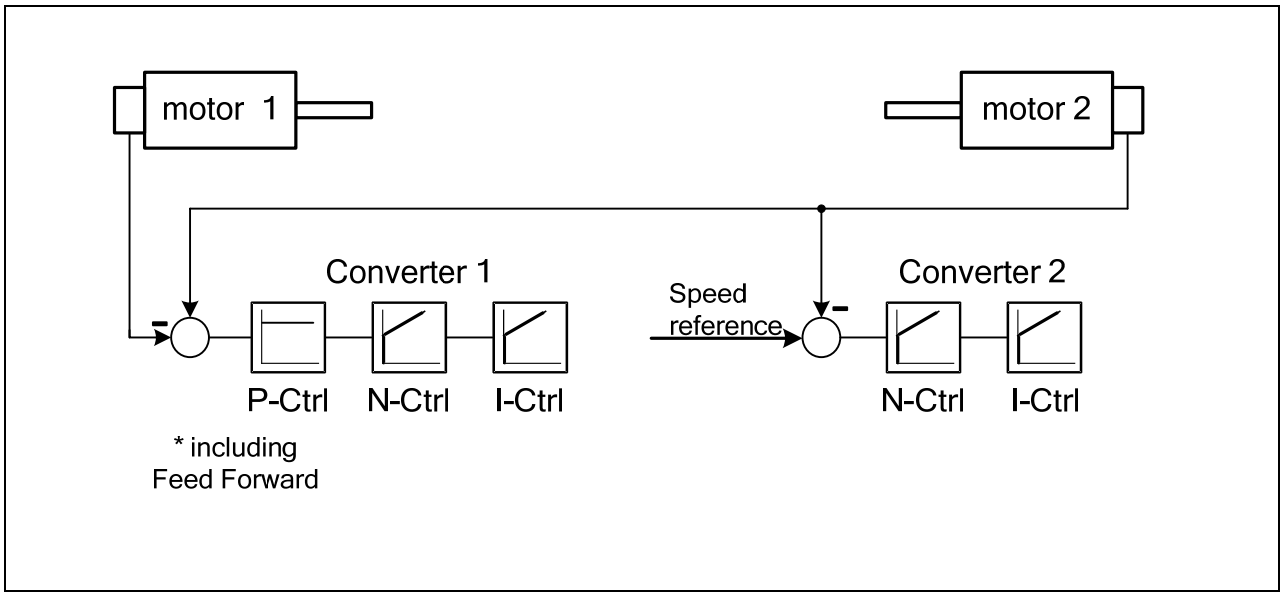
Typical Applications

Electrical shaft

Explanation

Idea is that two motors which are uncoupled have to run in the same angle (parallel feed). This configuration is often called *electrical shaft*.

The principal configuration is shown in the picture below:



Basically there are several opportunities to realize an *electrical shaft* application. One is shown in the picture. In this configuration converter 2 operates in speed control mode and converter 1 operates in position control mode. That means a position controller is added to the cascade control.

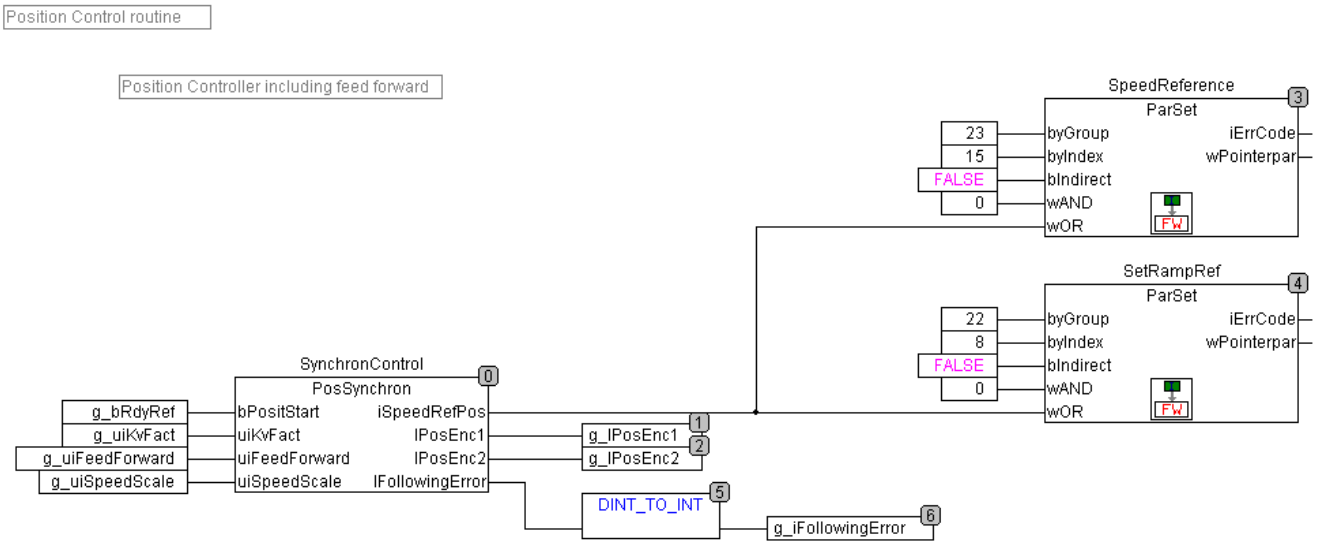
In this configuration converter 2 is the lead system which sends the actual speed and the position to the follower system. The follower system balances differences between motor 1 and motor 2.

Another possibility is to have one position controller and divide the position difference in half. Then the result will be send to converter 1 e.g. as positive value and to converter 2 e.g. as negative value.

There are lots of other ways to do an electrical shaft application. Which solution should be used depends on requirements and circumstances. This has to be decided by commissioning engineer!

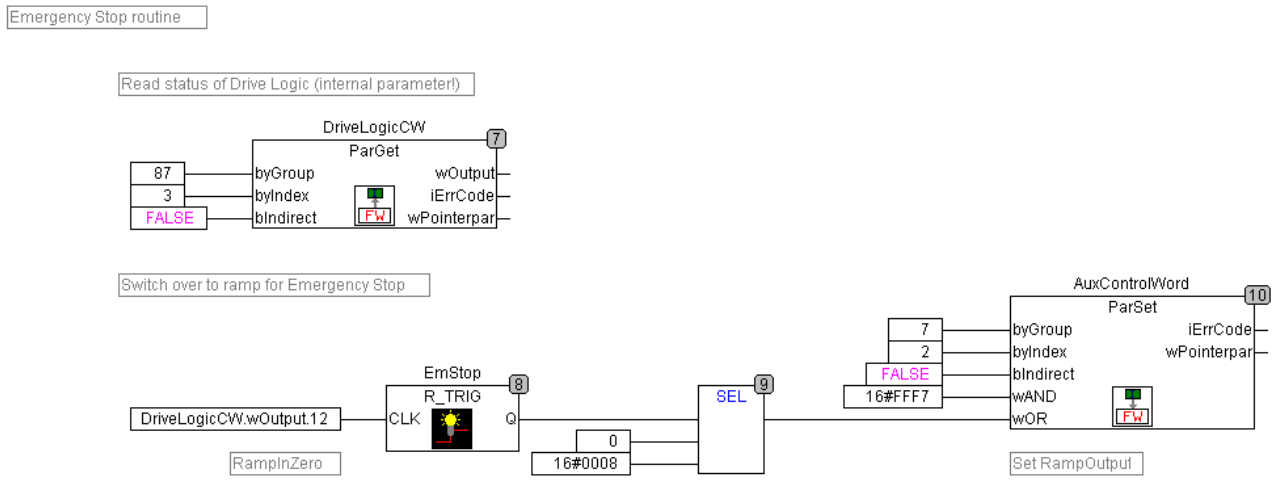
CoDeSys program

Position control and speed reference (5 ms cycle):



- Encoder 1 is connected with machine 1 and the speed control loop.
    - *P50.02* Measuring mode of encoder
    - *P50.03* Encoder
    - *P50.04* Number of pulse edges per revolution (encoder 1)
    - *P50.07* Rollover, because of a synchronous application
- Encoder 2 is connected with the second machine
    - *P98.01* Slot of RTAC-01.option module
    - *P50.18* Measuring mode of encoder 2
    - *P50.19* Number of pulse edges per revolution (encoder 2)
- Speed Reference Selection
    - *P11.03 SpeedRef 23.15*: This parameter can be used for a fast speed reference without ramp time delay!

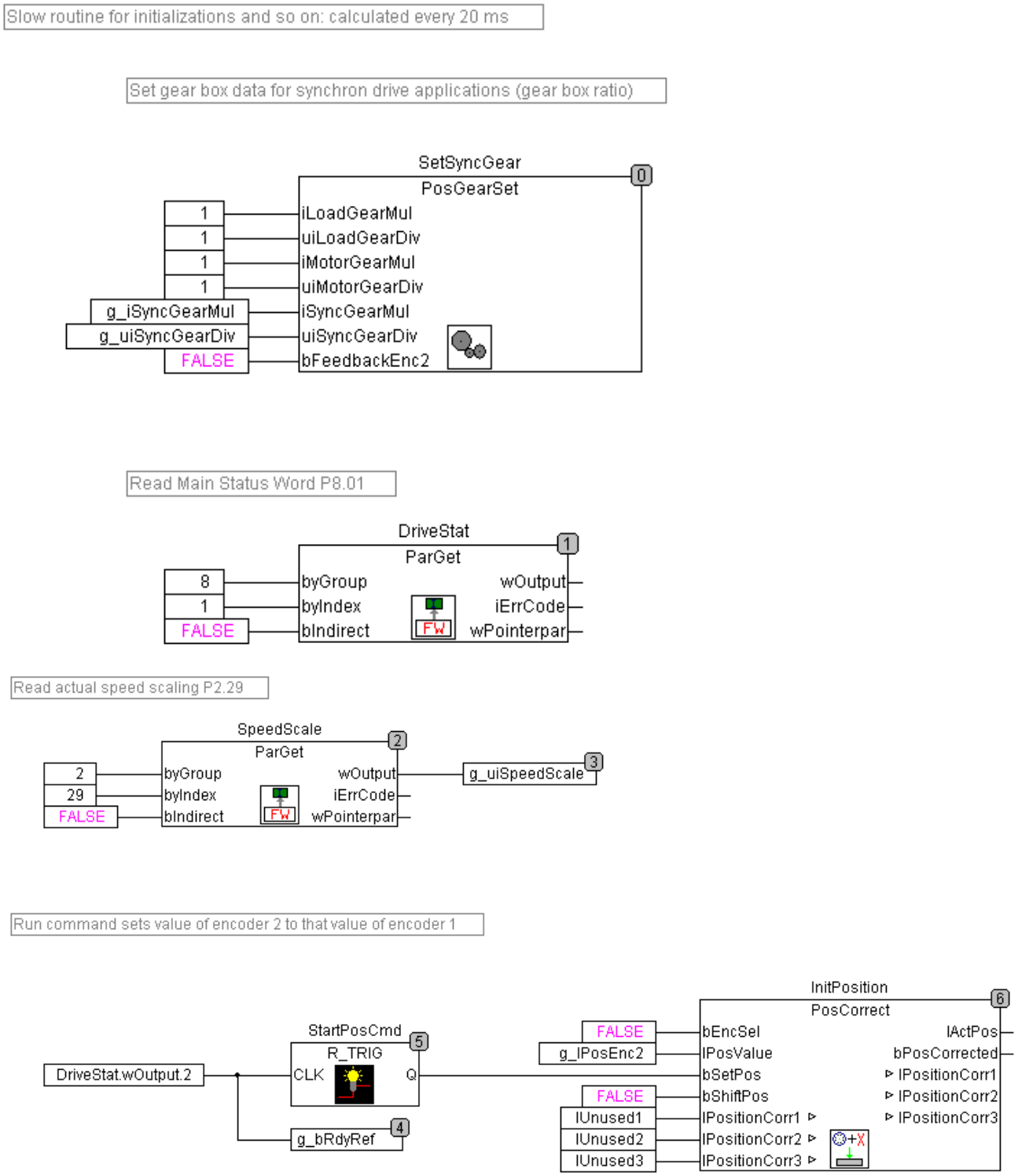
Emergency Stop routine (5 ms cycle):



- In normal operation the speed reference will be given to DCS800 firmware by parameter *P23.15 DirectSpeedRef*. In case of an emergency stop, the ramp will be used. Therefore also parameter *P22.08 BalRampRef* gets the speed reference.

- If emergency stop is activated, the application switches to the ramp generator and stops the drive by E-Stop ramp.

Slow calculation routine:

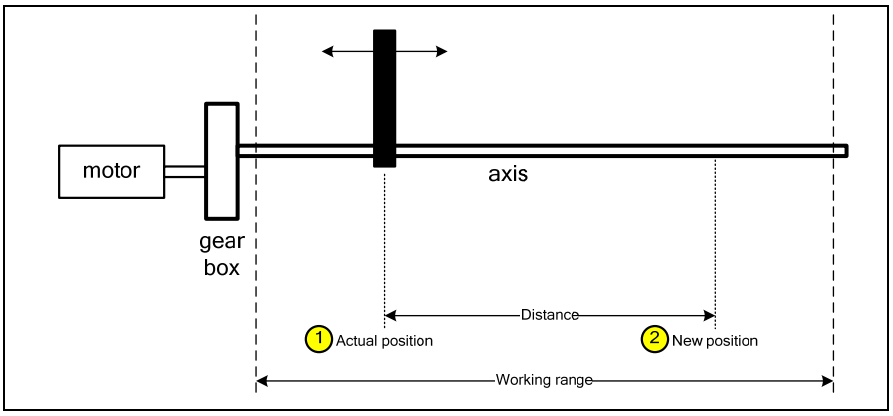


- Function block **PosGearSet** is used to connect a gear ratio to the application.
- Function block **PosCorrect** is used to set the value of encoder 2 to encoder 1.

Linear positioning

Explanation

Linear positioning means a motion between a start and an end point. The position value is always an absolute value.
The principle configuration is shown in the picture below:



The example shows a component which has to be moved from an absolute position to a new one. The spindle length is fixed what means that the motion is limited.

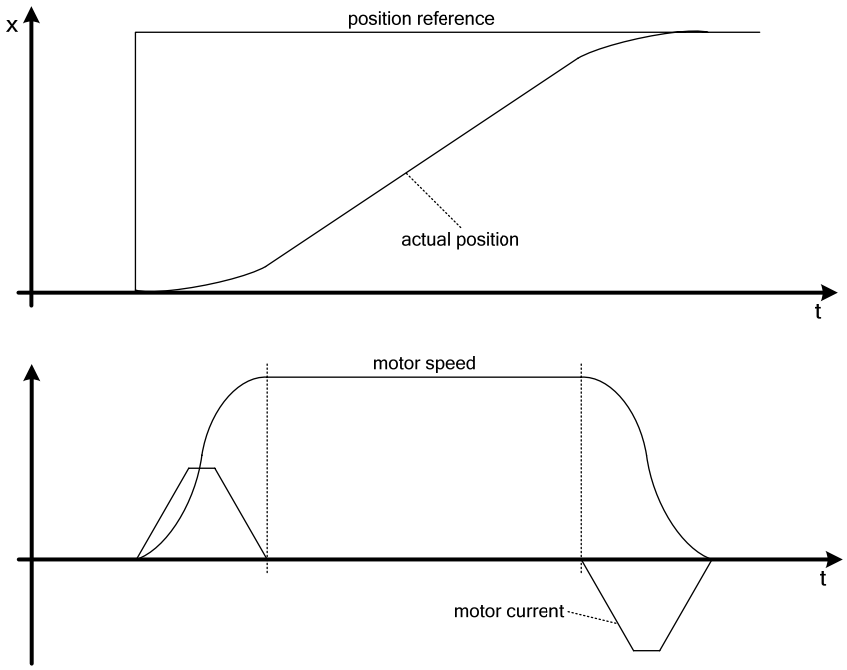
If a spindle should be used also the gradient of the axis is an important parameter for the application program which is called feed constant. The feed constant defines the motion which is moved during one revolution of the axis.

Example:

1. The component is in position 1 which is the actual position
2. Software gets a new position which is marked in the picture with position 2
3. The position interpolator calculates the optimal way to this new position considering the maximum speed and acceleration values
4. With start command of the software the component is moving to the new position

What is the following error?
Sometimes there is a difference between the position reference and the actual position of the axis. This difference is called following error or contouring error.
A following error occurs if the motor is in current limitation. This following error have to be monitored and must generate a fault in the drive if the difference become too much.

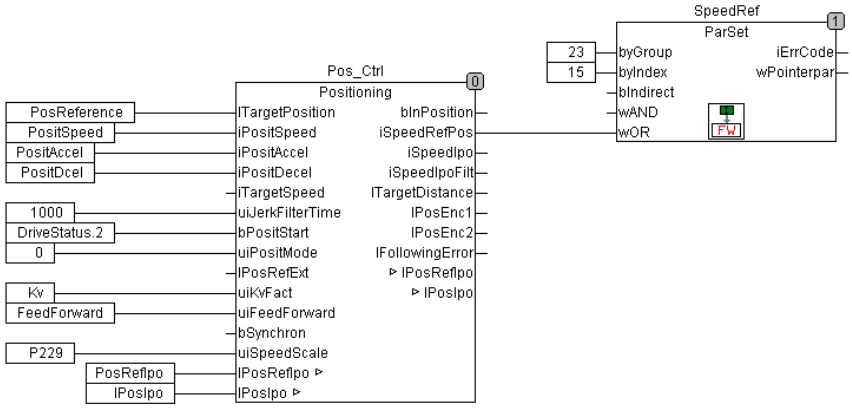
The picture shows an example for a positioning process:



If the software gets a new position the position interpolator starts to calculate the new interpolator reference and the motor drives the component to the new position.

CoDeSys program

Position control and fast speed reference (calculated in 5 ms):



PosReference is the main input of function block **Positioning**. The target position has to be connected scaled for 65536 as 1 revolution.
Variables *PositSpeed*, *PositAccel* and *PositDcel* coming from function block **PositParam**.

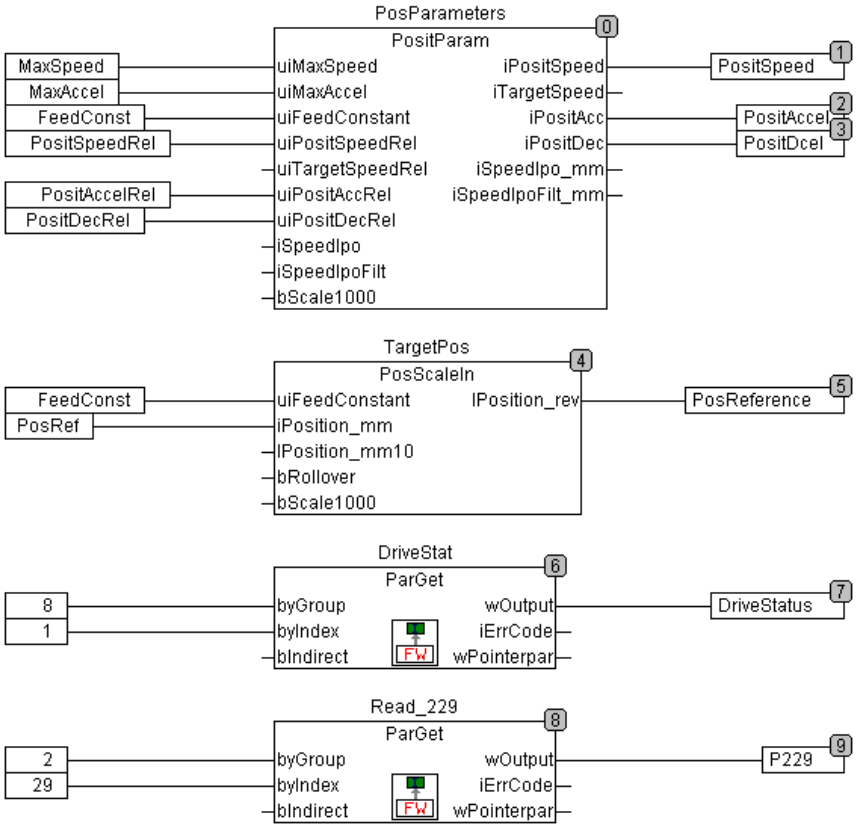
The gain of the position controller is the *Kv* factor.

FeedForward is the factor for speed which is connected to DCS800 firmware.

Variable *PosRefIpo* and *IPosIpo* are input and output variables which include the actual position reference calculated from the interpolator and the actual position of the motor.

This example shows the minimum number of needed parameters for this function block.

Slow calculation routine (e.g. 20 ms):



Function block **PositParam** scales the physical values of the mechanical construction and the motor to internal values which are used for function block **Positioning**.

Result of this function block is in this example:

- *PositSpeed*
- *PositAccel*
- *PositDcel*

These variables have to be connected to function block **Positioning**.

Function block **PosScaleIn** scales the position reference *PosRef* which is in millimeters into an internal value (32 bit) *PosReference*. This should be connected to function block **Positioning**.

Main Status Word (P8.01) is used to connect the *RUN*-command to function block **Positioning** to start the position interpolator.
Parameter P2.29 includes information about the scaling of the motor speed.

Commissioning hints

Basic drive system

Autotuning of Controllers

First step for commissioning of position control applications should be a standard commissioning of the armature current and field current controller. This can be done by start-up-assistant and / or autotuning functionality. It is recommended to switch-off the DCS800 Control Builder application (CoDeSys) during this process because it isn't needed during tuning phase.
Speed controller autotuning can be done if the field and armature circuit are working properly. If it is a linear motion which is limited by the process, the load should be uncoupled. Often the results of an autotuning are not usable and a manual tuning has to be done. The gain of the speed controller which is measured with uncoupled load is always too small because parameters for the speed controller depending only from load characteristics.
Please note all results from autotuning have to be checked by commissioning engineer. If necessary the values must be adapted to the actual control process.

Parameterization of DCS800 firmware

Further parameters settings are required to commission the drive system. This concerns all fault functions and limitations as well as the reference calculation.
If the brake logic should be used also this parameters have to be handled by application program. Please note that the main contactor must be controlled by DCS800 firmware because in regenerative mode it is dangerous to open it. In this case the thyristors and the motor could be damaged. For DC motors the standstill-time with motor nominal current is limited by time. Please note that the brake should close immediately when the drive is at the correct position and then the motor current must be switched-off.

Position control system

Check position control loop

First connect the DCS800 Control Builder application and enable it. Then check that the output of the position controller is properly connected to speed reference input. It is recommended to select the fast speed reference input P23.15 *DirectSpeedRef*. This reference can be selected by parameter P11.03 *SpeedRefSel*. Advantage is that the ramp generator is bypassed because it is only in 5 ms calculated. This reduces the delay time in speed reference circuit.

Only the position controller gain must be set. But gains of position controller and speed controller have to work together. This causes a manual adaption of speed controller gain. Please test the response of the position control loop by getting reference steps to speed reference input (Kv factor default setting 500…1000).

Linear motion check

Last step is to set the parameters for the positioning interpolator. These parameters depend only on mechanics and must be taken from existing mechanical construction.

By setting a new target position and give the start signal, the position controller starts working and the motor runs to the new position. Please be careful with distance limited linear applications. The emergency stop functionality must work all the time!

DCS800 family

DCS800-S modules

The versatile drive for any application

20 … 5,200 A_{DC}
0 … 1,160 V_{DC}
230 … 1,000 V_{AC}
IP00

- Compact
- Highest power ability
- Simple operation
- Comfortable assistants, e.g. for commissioning or fault tracing
- Scalable to all applications
- Free programmable by means of integrated IEC61131-PLC

DCS800-A enclosed converters

Complete drive solutions

20 … 20,000 A_{DC}
0 … 1,500 V_{DC}
230 … 1,200 V_{AC}
IP21 – IP54

- Individually adaptable to customer requirements
- User-defined accessories like external PLC or automation systems can be included
- High power solutions in 6- and 12-pulse up to 20,000 A, 1,500 V
- In accordance to usual standards
- Individually factory load tested
- Detailed documentation

DCS800-E series

Pre-assembled drive-kits

20 … 2,000 A_{DC}
0 … 700 V_{DC}
230 … 600 V_{AC}
IP00

- DCS800 module with all necessary accessories mounted and fully cabled on a panel
- Very fast installation and commissioning
- Squeezes shut-down-times in revamp projects to a minimum
- Fits into Rittal cabinets
- Compact version up to 450 A and Vario version up to 2,000 A

DCS800-R Rebuild Kit

Digital control-kit for existing powerstacks

20 … 20,000 A_{DC}
0 … 1,160 V_{DC}
230 … 1,200 V_{AC}
IP00

- Proven long life components are re-used, such as power stacks, (main) contactors, cabinets and cabling / busbars, cooling systems
- Use of up-to-date communication facilities
- Increase of production and quality
- Very cost-effective solution
- Open Rebuild Kits for nearly all existing DC drives
- tailor-made solutions for…
  - BBC PxD
  - BBC SZxD
  - ASEA TYRAK
  - other manufacturers

ABB Automation Products
Wallstadter-Straße 59
68526 Ladenburg • Germany
Tel: +49 (0) 6203-71-0
Fax:+49 (0) 6203-71-76 09
www.abb.com/motors&drives

Ident. No.: 3ADW000348R0101 Rev A
12_2008

348R101A85000