

Festo_Positioning_Basic library – Configuration H-/T-/XY-Gantry and coordinate systems

Commissioning of a selected system with the Festo_Positioning_Basic library is explained step by step, so that it's successfully integrated into the user program.

FPosB

Title Festo_Positioning_Basic library – Configuration H-/T-/XY-Gantry
Version 1.30
Document no. 100102
Originalen
AuthorFesto

Last saved 02.06.2021

Copyright Notice

This documentation is the intellectual property of Festo SE & Co. KG, which also has the exclusive copyright. Any modification of the content, duplication or reprinting of this documentation as well as distribution to third parties can only be made with the express consent of Festo SE & Co. KG.

Festo SE & Co KG reserves the right to make modifications to this document in whole or in part. All brand and product names are trademarks or registered trademarks of their respective owners.

Legal Notice

Hardware, software, operating systems and drivers may only be used for the applications described and only in conjunction with components recommended by Festo SE & Co. KG.

Festo SE & Co. KG does not accept any liability for damages arising from the use of any incorrect or incomplete information contained in this documentation or any information missing therefrom.

Defects resulting from the improper handling of devices and modules are excluded from the warranty.

The data and information specified in this document should not be used for the implementation of safety functions relating to the protection of personnel and machinery.

No liability is accepted for claims for damages arising from a failure or functional defect. In other respects, the regulations with regard to liability from the terms and conditions of delivery, payment and use of software of Festo SE & Co. KG, which can be found at www.festo.com and can be supplied on request, shall apply.

All data contained in this document do not represent guaranteed specifications, particularly with regard to functionality, condition or quality, in the legal sense.

The information in this document serves only as basic information for the implementation of a specific, hypothetical application and is in no way intended as a substitute for the operating instructions of the respective manufacturers and the design and testing of the respective application by the user.

The operating instructions for Festo products can be found at www.festo.com/sp.

Users of this document (application note) must verify that all functions described here also work correctly in the application. By reading this document and adhering to the specifications contained therein, users are also solely responsible for their own application.

Table of contents

1	System configuration, hardware example H-gantry	5
1.1	00_GVLs	5
1.1.1	GVL_CNC	5
1.1.2	GVL_Config	5
1.1.3	GVL_Motion_FB	11
1.1.4	GVL_Motion_INOUT	12
1.1.5	Adding CNC files for Internal CNC mode	14
2	Configuration of a linear gantry	16
2.1	Linear gantry	16
2.2	XY-gantry	17

1 System configuration, hardware example H-gantry

1.1 00_GVLs

1.1.1 GVL_CNC

Global parameters which can be used for CNC are defined in the list.

```
VAR_GLOBAL (*CNC*)

(*CNC_Parameter*)
fVel      : LREAL := 100;
fAcc      : LREAL := 1000;
fDec      : LREAL := -1000;
fZAxis    : LREAL := -1000;

(*Parameters for Pick and Place FB*)
fbPick_n_Place : FPosB.FB_PICK_AND_PLACE_CYCLIC_G10;
//fbPick_n_Place : FPosB.FB_PICK_AND_PLACE_CYCLIC_G51_52;
gstDynPnP      : FPosB.DYNAMIC := (lrVel:= 100, lrAcc:= 1000, lrDec:=-1000 );
gstDynPnP_AddAxes : FPosB.DYNAMIC := (lrVel:= 100, lrAcc:= 1000, lrDec:=-1000 );
gstPickPos     : FPosB.CART_POS := (lrX:= 30, lrY:= 30, lrZ:= 30,  lrA:= 90, lrU:= 90 );
gstParkPos     : FPosB.CART_POS := (lrX:= 10, lrY:= 10, lrZ:= 10,  lrA:= 10, lrU:= 10 );
gstPlacePos    : FPosB.CART_POS := (lrX:= 50, lrY:= 50, lrZ:= 30,  lrA:= 180, lrU:= 180 );
glrRadius      : LREAL:= 30;
glrUpperZ      : LREAL:= 10;
END VAR
```

Values are specified as examples for velocity, acceleration and deceleration, which can be invoked in the CNC code.

Example in the CNC code:

```
N000 F$fVel$ E$fAcc$ E$fDec$
```

1.1.2 GVL_Config

All parameters which effect the system's configuration are defined in GVL_CONFIG. As a rule these are values which are set only once.

And thus it's important to be familiar with the system along with its limits.

Note: KINEMATIK and DRIVE configurations may not be changed while the system is running. This results in malfunctioning of the system.

The **KINEMATIK_DATA_REF** structure is declared at the beginning of the GVL. This is elementary for all configuration and process data, as well as CNC path preparation modules.

```
gstGantryDataRef : FPosB.KINEMATIC_DATA_REF;
```

The rest of the variables are subdivided into 4 topics: DRIVE, DYNAMICS, HOMING and KINEMATICS.

A corresponding structure which contains the essential variables is assigned to each of these areas.

The descriptions of the function blocks can be found in Festo_Positioning_Basic_3 (onlinehelp Codesys).

The configuration is shown below at the decisive places for the sample system.

All other settings are application-specific and must be determined by the user. These values are left at their default settings for the example.

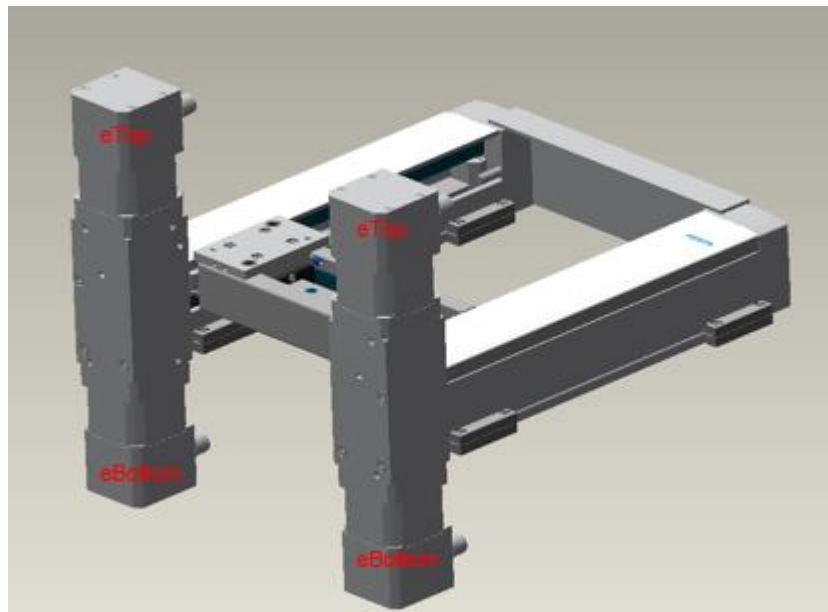
DRIVE_CONFIG

```
(*****DRIVE_CONFIG*****)
g_stDrive : FPosB.DRIVE_CONFIG := (
  arxAvailable      := [1,1,1,1,1,1], // Set available Axes of the System
  areControllerType := [FPosB.eFestoEmcxST_EC, FPosB.eFestoEmcxST_EC, FPosB.eFestoEmcxST_EC,
                        FPosB.eFestoEmcxST_EC, FPosB.eFestoEmcxST_EC, FPosB.eFestoEmcxST_EC], // Controller Type
  areMotorOrientation := [FPosB.eBottom, FPosB.eBottom, FPosB.eBottom,
                          FPosB.eBottom, FPosB.eBottom, FPosB.eBottom], // Motor Orientation for H/T-Portal
  arstRef           := [ADR(D1),ADR(D2),ADR(D3),ADR(D4),ADR(D5),ADR(D6)], // Axis reference
  arxLimitAxes      := [0,0,0,0,0,0], //Activate the Axis that need a Dynamic Limit
  arlrFeedConstance := [0,0,0,0,0,0]: //Feedconstance of the Axis
```

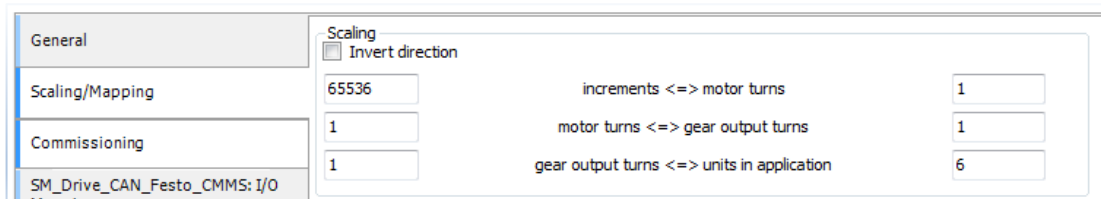
arxAvailable: The system has three controllers. The order is fixed and the first two controllers are reserved for the 2D gantry.

areControllerType: All three controllers are type CMMS devices. 6 different types of motor controller are available: CMMP, CMMP_ETC, CMMS, CMMT_ETC, EMCA and EMCX_ST.

areMotorOrientation: The motors are mounted at the bottom of the 2D gantry.



These settings don't have any effect on single axes. If the drive positions in the wrong direction, reversing must be changed in Codesys(to Use FposB the **Invert Direction** Setting of the Drive should **be always deactivated** for the H-/T-Gantry):



arstRef: The link between the physical soft motion axis and the Codesys project is established here.

arxLimitAxes In this example, only the dynamics of the Z axis are limited. These dynamics must be set in DYNAMIC_CONFIG.

arlrFeedConstance The feedconstant can be entered in the library itself for each drive. It will overwrite the settings done in chapter 2.4.1 of 02 Overview FPosB Basicproject_EVO2 in Scaling/Mapping. The basic scaling of each drive must be valid (increments of the encoder for instance) and it's recommended to have good basic settings.

If the values are set to "0", the library will use the default values, which are set in Scaling/Mapping.

NOTE: For **CMMT** the feedconstant is set in Festo Automation Suite.

Keep the `arlrFeedConstance` for CMMT set to zero, so the default values from the drives are used here.

DYNAMIC_CONFIG

```
(*****DYNAMIC_CONFIG*****)
g_stDynLimits      : FPosB.DYNAMIC_CONFIG := (
    //Max Dynamic of the Drives, If <> 0 Parameter at Drives of Devices will be overwritten with arstMaxAxisDyn
    arstMaxAxisDyn  := [(lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M1
                        (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M2
                        (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M3
                        (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M4
                        (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M5
                        (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0)], //M6

    arxLimitAxes    := [0,0,0,0,0,0], //Activate the Axis that need a Dynamic Limit
    // Limit Dynamic: Limit the Dynamic of Path to not exceed the Limit of then Axes
    arstLimitAxisDyn := [(lrVel:= 500, lrAcc:= 1000), //M1
                        (lrVel:= 500, lrAcc:= 1000), //M2
                        (lrVel:= 500, lrAcc:= 1000), //M3
                        (lrVel:= 500, lrAcc:= 1000), //M4
                        (lrVel:= 500, lrAcc:= 1000), //M5
                        (lrVel:= 500, lrAcc:= 1000)], //M6

    lrPathAngleTolerance := 0.1, // Max. Angle of 2 CNC Segment to not stop at this Corner with the Angle
    lrMaxJerk             := 100000, // Set the Max Jerk value
    lrQuickStopJerk       := 1000000, // Set the max Jerk value for Quickstop
    eIpoVelMode           := SMC_INT_VELMODE.QUADRATIC); // Velocity Mode SMC_INT_VELMODE
```

The settings must match to the application.

At this point just a few explanations to 3 settings.

There are two different limitations, which can be set:

LimitAxisDyn

These settings are valid for the **path** of a corresponding motion (`MaxAxisDyn = DRIVE`).

M1, M2 and M3 correspond here with the axes X, Y and Z.

These 3 axes/drives can be part of a transformation, like H- or T-gantry, so for instance for a Y movement 2 axes are active.

```
// Limit Dynamic: Limit the Dynamic of Path to not exceed the Limit of then Axes
arstLimitAxisDyn := [(lrVel:= 500, lrAcc:= 1000), //M1
                    (lrVel:= 500, lrAcc:= 1000), //M2
                    (lrVel:= 500, lrAcc:= 1000), //M3
                    (lrVel:= 500, lrAcc:= 1000), //M4
                    (lrVel:= 500, lrAcc:= 1000), //M5
                    (lrVel:= 500, lrAcc:= 1000)], //M6
```

To activate these limits, the limitation of the corresponding axis (or pathdirection) must be activated, too.

This is done in the `Drive_Config` (`arxLimitAxes`).

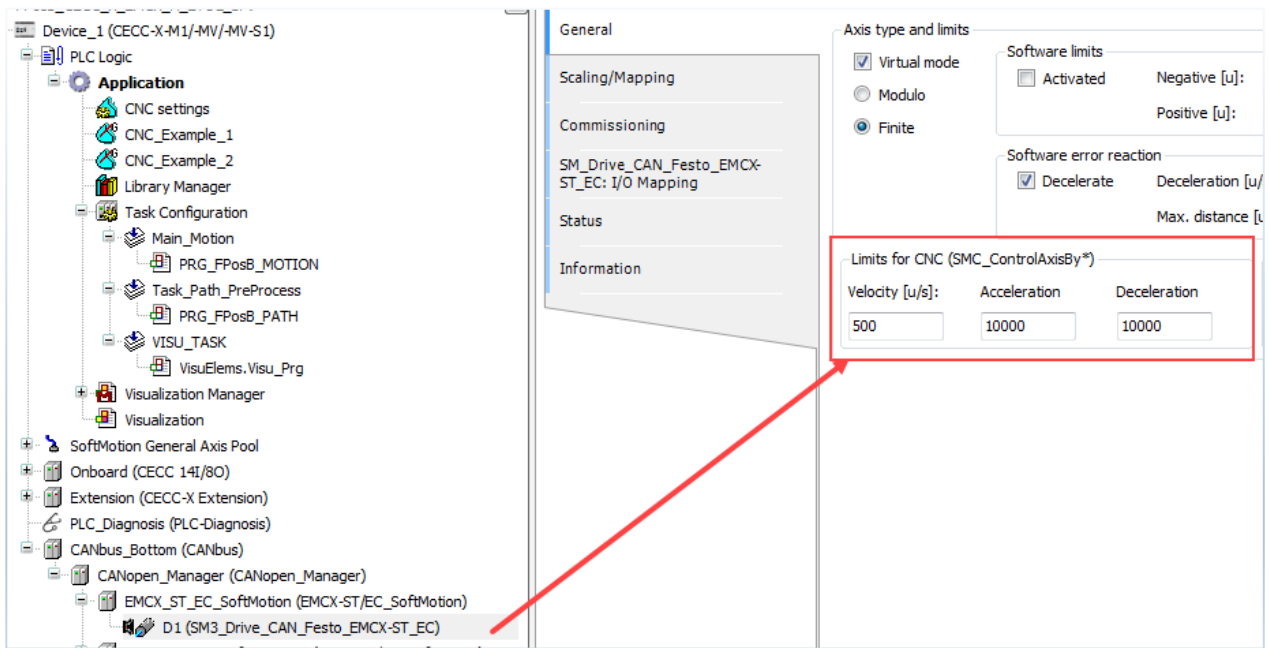
```
-----
arstRef      := [ADR(Dx), ADR(Dy), ADR(Dz)], // Axis reference
arxLimitAxes := [0,0,1,0,0,0], //Activate the Axis that need a Dynamic Limit
arlrFeedConstance := [38,38,1,0,0,0]); //Feedconstance of the Axis
```

MaxAxisDyn

These dynamic settings actually refer to the real drives (`LimitAxisDyn = PATH`).

```
//Max Dynamic of the Drives, If <> 0 Parameter at Drives of Devices will be overwritten with arstMaxAxisDyn
arstMaxAxisDyn := [(lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M1
                  (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M2
                  (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M3
                  (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M4
                  (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0), //M5
                  (lrVel:= 0, lrAcc:= 0, lrDec:= 0, lrJerk := 0)], //M6
```

These dynamics correspond to the settings done for each drive:



If one of the values for MaxAxisDyn is “0”, the library will use the default settings, which are set in “Limits for CNC (SMC_ControlAxisBy*)”.

The **difference between both** dynamic values is the following one:

While the LimitAxisDyn limits the axes (Path velocity) to the set velocity and acceleration (“cruise control”), the MaxAxisDyn will interrupt the drives with an error as soon as the velocity is too high (“handbrake”).

elpoVelMode:

It’s possible to change the velocity mode of the interpolater within the library.

There are 5 types: TRAPEZOID, SIGMOID, SOGMOID_LIMIT, QUADRATIC and QUADRATIC_SMOOTH. Please refer the Codesys online help for further information regarding these types (search for SMC_INT_VELMODE).

If the bit xLimitJerk is activated, this velocity mode will be automatically set to QUADRATIC internally.

```
lrPathAngleTolerance := 0.1, // Max. Angle of 2 CNC Segment to not stop at this Corner with the Angle
lrMaxJerk             := 100000, // Set the Max Jerk value
lrQuickStopJerk       := 1000000, // Set the max Jeerk value for Quickstop
eIpoVelMode           := SMC_INT_VELMODE.QUADRATIC; // Velocity Mode SMC_INT_VELMODE
```

lrMaxJerk:

Magnitude of the maximum allowed jerk: It’s only used for the quadratic velocity modes. (QUADRATIC and QUADRATIC_SMOOTH)

It must be positive and cannot be changed while the interpolator is running. The jerk are related to the CNC-Path and not to the Single Axis.

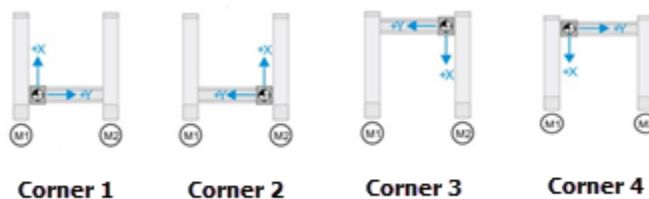
HOMING_CONFIG

```

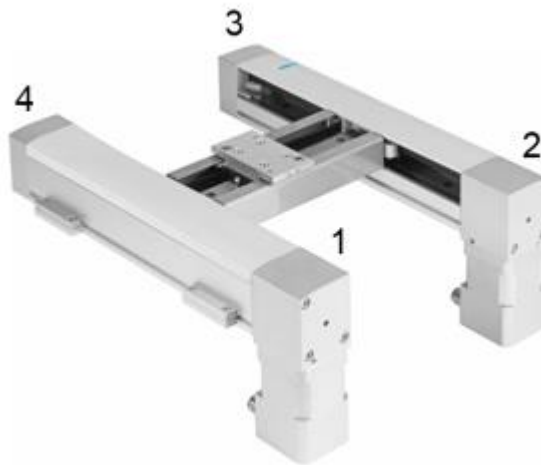
(*****HOMING_CONFIG*****)
g_stHoming : FPosB.HOMING_CONFIG := (
  xSetHomingCornerAsOrigin := TRUE,           // Set the Coordinat Origin to Active Homing Corner
  eHomingCorner             := FPosB.eCorner3, // Homing Corner for H/T-Portal
  stParkDynamic             := (lrVel:= 30, lrAcc:= 1000, lrDec:= -1000), // Danymic for moving to Parkposion after Homing
  areHomingPosSingleAxis   := [FPosB.eHWLimitsMin, FPosB.eHWLimitsMin, FPosB.eHWLimitsMin, FPosB.eUserHomingPos, FPosB.eUserHomingPos, FPosB.eUserHomingPos], // HomingPos For Homing
  arlrUserHomingPos        := [0,0,0,0,0,0], // Homing Pos define from User
  arxExcludeAxis           := [0,0,0,0,0,0], // Exclude Axis for the Homing procedure
  ariHomingOrder           := [4,3,2,1,1,1], // Set the Homing order for the system
  areHomingMethodeSingleAxis := [0,0,0,0,0,0], // Homingmethode for the drives
  arxSaveHomingOffsetToEncoder:= [0,0,0,0,0,0], // Only Trigger it in the Visu after Homing to save the Position into the MontorController
  xMoveToParkPos           := TRUE,           // Move to ParkPos after Homing
  xPortalAlreadyHomed      := FALSE,          // Preset the System Homed-bit
  stParkPos                := ( lrX:= 0, lrY:= 0, lrZ:= 0, // ParkPosition after Homing Done
                                lrA:= 0, lrB:= 0, lrC:= 0,
                                lrU:= 0, lrV:= 0, lrW:= 0),
  stEmcxHomingSensorConfig := (arxEnableHomingSensor := [0,0,0,0,0,0],
                                ariSensorTypeAtDI3    := [0,0,0,0,0,0],
                                ariSensorTypeAtDI4    := [0,0,0,0,0,0] ),
  tHomingTimeOut           := T#60S);          // Timeout for Homing duration

```

xSetHomingCornerAsOrigin: If this option is active, the Coordinate Origin of the gantry is set to the active Homing-Corner.



eHomingCorner: The definition of the corners in the library are as follows:



Tip: If possible, always home to the side with the motors (eCorner1 and eCorner2).

ariHomingOrder: The homing order is based on the cartesian system, not on the single drives. In that example the gantry will first home in Z-, then in Y- and finally in X-direction independently of the used kinematic.

areHomingPosSingleAxis: Homing Pos definition for individual Axes. Value for M1 and M2 will be ignored if a H-Gantry is doing homing, same for T-Gantry and M2 and M3

ENUM HOMING_POS_SINGLE_AXIS

Name	Type	Inherited from	Address	Initial	Comment
eUserHomingPos	INT			-1	Homing Position set to 0
eZeroPos	INT			0	Homing Position set to 0
eHWLimitsMin	INT			1	Homing Position set to Minimum Hardware Limit
eHWLimitsMax	INT			2	Homing Position set to Maximum Hardware Limit

arlrUserHomingPos:	individual Homing Position defined by user if the homing position is different to stKinematicHWLimit
areHomingMethodeSingleAxis:	The homing method of every single axis can be changed within the library itself. If the value is "0", it will use the default value. (e.g Homing on Block (-17, -18), Homing on ActPos (35,37)...))
arxSaveHomingOffsetToEncoder:	Activates the Save-Homing-Offset for Drives with multiturn encoder after homing.
stParkPos	MoveToParkPos is active. Travel to these positions takes place after homing.
stEmcxHomingSensorConfig:	Homing Configuration for EMCX Motor for Homing on Sensor, and Configuration of the Inputs of EMCX as Homing-Sensor Input For further Information please contact Festo Service.

KINEMATIC_CONFIG

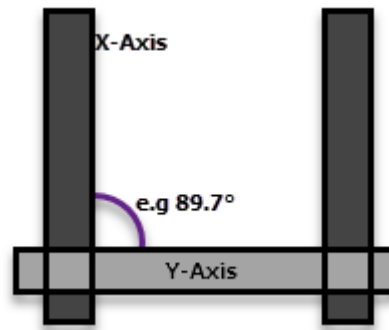
```

(*****KINEMATIK_CONFIG*****)
g_stKinematic      : FPosB.KINEMATIC_CONFIG := (

    eKinematicType      := FPosB.eH_Gantry,          // Kinematic Type
    stAddAxesConfig     := ( xAbcAxesAsOrientationAxes := FALSE,
                             unToolLength             := (ar0_2 := [0,0,0]), // TOOL Length IJK
                             eOriConv                  := SMC_ORI_CONVENTION.ZYZ),
    eH_GantryBeltLock    := FPosB.eH_MOTOR_SIDE,      // Beltlock of the H-Portal
    //Software Limits
    stKinematicSWLimit  := (stMin:= (lrX:= -5, lrY:= -5, lrZ:= -3, lrA:= -1080, lrB:= -3, lrC:= -3,
                                     lrU:= -1080, lrV:= -3, lrW:= -3 ),
                             stMax:= (lrX:= 380, lrY:= 330, lrZ:= 130, lrA:= 1080, lrB:= 100, lrC:= 100,
                                     lrU:= 1080, lrV:= 100, lrW:= 100)),
    // Hardware Limits
    stKinematicHWLimit  := (stMin:= (lrX:= -10, lrY:= -10, lrZ:= -5, lrA:= -1085, lrB:= -5, lrC:= -5,
                                     lrU:= -1085, lrV:= -5, lrW:= -5),
                             stMax:= (lrX:= 390, lrY:= 350, lrZ:= 140, lrA:= 1085, lrB:= 105, lrC:= 105,
                                     lrU:= 1085, lrV:= 105, lrW:= 105)),
    lrXY_Compensation    := 0.0,                      // dPhi of Portal X and Y in [°]
    xUseUvwAxes          := TRUE );

```

eKinematicType:	eH-Gantry is selected because a 2D gantry serves as the basis.
stKinematicSWLimit:	Three axes are described by the 2D gantry and the Z-axis. The software limits for min. and max. are set and should lie between the hardware limits.
stKinematicHWLimit:	These are the limits of the mechanical system. The difference between the minimum and maximum values is equal to the mechanical travel distance (e.g. Z-axis: 25 mm).
stAddAxesConfig:	This setting enables the possibility to use G43. G43 starts tool length compensation. Additional Information can be found here: https://help.codesys.com/webapp/_sm_cnc_din66025_preprocessor;product=codesys_softmotion;version=4.10.0.0
lrXY_Compensation:	this value is used to compensate the deviation of the perpendicularity between X and Y. Unit in [°]. Example: the angle between X and Y is 89.7° so the different to 90° is 0.3°. The compensation in this case is -0.3°. Deviation in CW is negative and CCW is positive.



1.1.3 GVL_Motion_FB

The 3 modules which are invoked in PRG_FPOSB_Motion are instantiated in this list.
If the instance names don't need to be changed, no adjustments are necessary.

```
VAR_GLOBAL

(* Config_Data *)
fbConfig_System : FPosB.FB_CONFIG_SYSTEM;

(* Input Data *)
fbInput_Value   : FPosB.FB_TARGET_PROCESS_VALUE;

(* Output Data *)
fbOutPut_Value  : FPosB.FB_ACTUAL_PROCESS_VALUE;

(* WebVisu Interface *)
fbWebVisuInterface : FPosB.FB_WebVisuInterface; (* instance of "FB_WebVisuInterface" to process webvisu *)
END_VAR
```

1.1.4 GVL_Motion_INOUT

This list is important for the modules from preceding section 7.1.3.

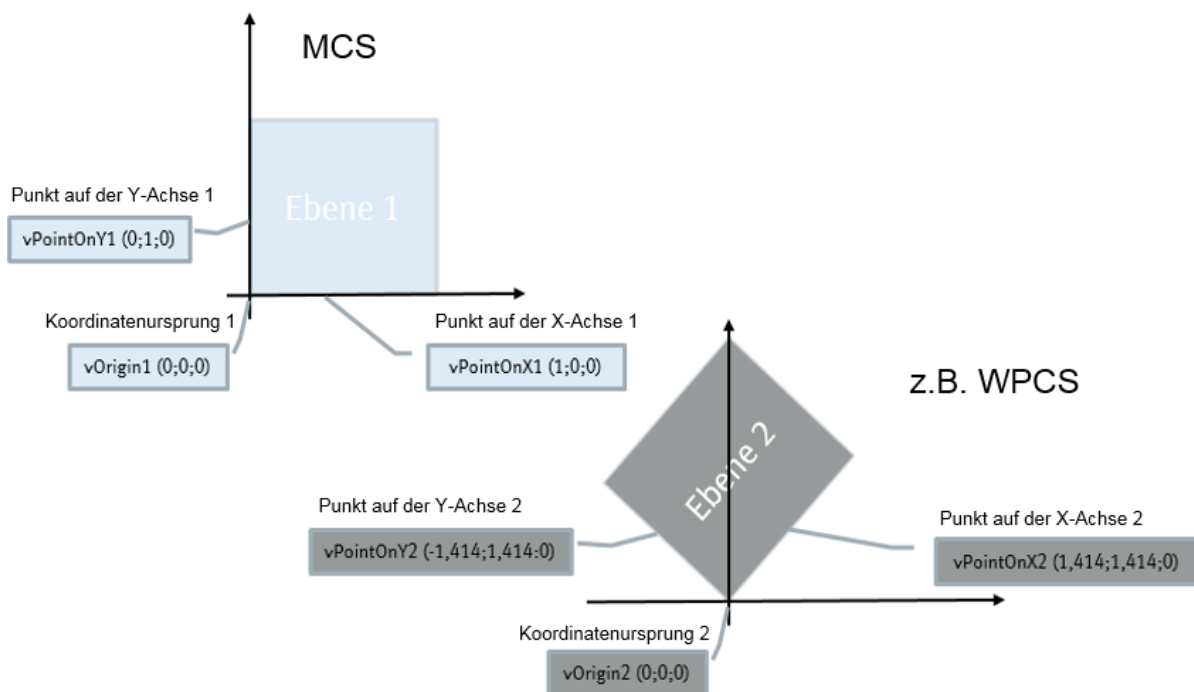
Description of the coordinate systems

```

vOrigin1           : SMC_VECTOR3D := (dX:= 0, dY:= 0, dZ:= 0);
vPointOnX1         : SMC_Vector3D := (dX:= 1, dY:= 0, dZ:= 0);
vPointOnY1         : SMC_Vector3D := (dX:= 0, dY:= 1, dZ:= 0);

vOrigin2           : SMC_VECTOR3D := (dX:= 0, dY:= 0, dZ:= 0);
vPointOnX2         : SMC_Vector3D := (dX:= 1.414, dY:= 1.414, dZ:= 0);
vPointOnY2         : SMC_Vector3D := (dX:= -1.414, dY:= 1.414, dZ:= 0);
    
```

Several coordinate systems can be used later. The systems are described with these vectors.



If CNC files are used which are read in, values of variables from the project can be assigned to the strings included there.

```

garSingleVar      : ARRAY[0..3]OF SMC_SingleVar := [ (strVarName := 'FVEL',    eVarType := SMC_TYPE_LREAL, pAdr := ADR(fVel)),
                                                       (strVarName := 'FACC',    eVarType := SMC_TYPE_LREAL, pAdr := ADR(fAcc)),
                                                       (strVarName := 'FDEC',    eVarType := SMC_TYPE_LREAL, pAdr := ADR(fDec)),
                                                       (strVarName := 'FZAXIS',  eVarType := SMC_TYPE_LREAL, pAdr := ADR(fZAxis))
                                                       ];
    
```

Input and output data are summarized in structures. These structures are declared so that they can be used in PRG_FH_MOTION.

The input structure is predefined with values.

Input:

```
(*FPB_INPUT_DATA*)
g_stIn      : FPosB.FPB_INPUT_DATA := (
    eMOP                := FPosB.eNoMode,
    //stCW               := gstCW,
    eSetCS              := FPosB.eMCS,
    stUCS               := (vOrigin:= vOrigin1, vPointOnX:= vPointOnX1, vPointOnY:= vPointOnY1),
    stWPCS              := (vOrigin:= vOrigin2, vPointOnX:= vPointOnX2, vPointOnY:= vPointOnY2),
    stDynPtp            := (lrVel:=300 , lrAcc:= 10000, lrDec:= -10000 ),
    stPosPtp            := (lrX:= 100, lrY:= 100, lrZ:= 20 , lrU:=0 , lrV:= 0, lrW:= 0),
    stDynJogStep        := (lrVel:=10 , lrAcc:= 1000, lrDec:= -1000 ),
    lrStepWidth         := 5,
    eSelectAxis         := FPosB.eX,
    sCNCFilePath        := ',','','ffx/codesys/_cnc/CNC1_TOOL_File.cnc',// '','','ffx/codesys/_cnc/^.cnc'
    uiUserCNCNo         := 1,
    //stCNC CorrParameters := ,
    uiOverride          := 100,
    xEnableRampIn       := TRUE,
    tRampInDelay        := T#8MS,
    stCNCVarList        := (wNumberVars := 16, psvVarList := ADR(garSingleVar[0])),
    stDefaultDynCNC     := (lrVel:=10 , lrAcc:= 1000, lrDec:= -1000 ),
    stDefaultDynFFCNCGO := (lrVel:=10 , lrAcc:= 1000, lrDec:= -1000 ),
    parCNCUser          := [(ADR(CNC_Example_1)) // Max.256 CNC_Example_1
    ]);
```

Output:

```
(*FPB_OUTUT_DATA*)
g_stOut      : FPosB.FPB_OUTPUT_DATA;
    (* stSW:= ,
    uiActualOverride:= ,
    lrActualPathVelocity:= ,
    arstSWLimitSwitch:= ,
    arstActualDynamicACS:= ,
    stActualPositionACS:= ,
    stActualPositionMCS:= ,
    stActualPositionUCS:= ,
    stActualPositionWPCS:= ,
    xIPOWorking:= ,
    iIPOStatus:= ,
    stIPOActSetPos:= ,
    stActualTangent:= ,
    sActSelectedPrgName:= ,
    eMoPDisplay:= ,
    eActiveKinematicType:= ,
    eActiveCS:= ,
    stActiveCSData:= ,
    lrQualityOfUCS:= ,
    lrQualityOfWPCS:= ,
    eSelectedAxis:= ,
    stCncFunctions:= ,
    stCNCList:= ,
    stMostSignificantErrorMessage:= ,
    arstMessageBuffer:= ); *)
```

1.1.5 Adding CNC files for Internal CNC mode

There are two ways, how the CNCs are loaded in this library:

There is also an additional Application Notes to handle this topic. **09 Internal And External CNC List.pdf**

1. Related to the code.

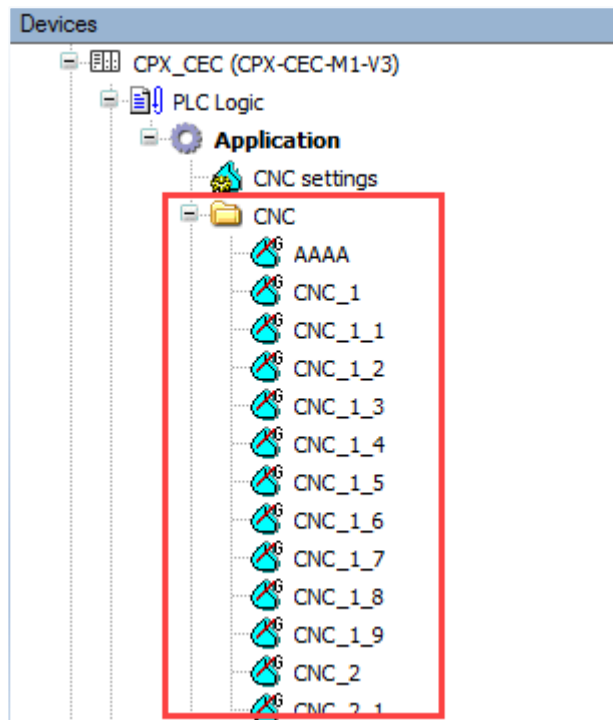
These CNCs are **prioritized**, so all coded CNCs are added to the internal CNC list (up to 255).

```
(*FPB_INPUT_DATA*)
g_stIn      : FPosB.FPB_INPUT_DATA := (
    eMOP                := FPosB.eNoMode,
    //stCW               := gstCW,
    eSetCS              := FPosB.eMCS,
    stUCS               := (vOrigin:= vOrigin1, vPointOnX:= vPointOnX1, vPointOnY:= vPointOnY1),
    stWPCS              := (vOrigin:= vOrigin2, vPointOnX:= vPointOnX2, vPointOnY:= vPointOnY2),
    stDynPtp            := (lrVel:=300, lrAcc:= 10000, lrDec:= -10000),
    stPosPtp            := (lrX:= 100, lrY:= 100, lrZ:= 20, lrU:=0, lrV:= 0, lrW:= 0),
    stDynJogStep        := (lrVel:=10, lrAcc:= 1000, lrDec:= -1000),
    lrStepWidth         := 5,
    eSelectAxis         := FPosB.eX,
    sCNCFilePath        := '//ffx/codesys/_cnc/CNC1_TOOL_File.cnc',// '//ffx/codesys/_cnc/^.cnc'
    uiUserCNCNo         := 1,
    //stCNC CorrParameters := ,
    uiOverride          := 100,
    xEnableRampIn       := TRUE,
    tRampInDelay        := T#8MS,
    stCNCVarList        := (wNumberVars := 16, psvVarList := ADR(garSingleVar[0])),
    stDefaultDynCNC     := (lrVel:=10, lrAcc:= 1000, lrDec:= -1000),
    stDefaultDynFFCNCGO := (lrVel:=10, lrAcc:= 1000, lrDec:= -1000),
    parCNCUser          := (ADR(CNC_Example_1)) // Max.256 CNC_Example_1
);
```

Or addressed to a specific slot:

```
fbPick_n_Place_EVO3(
    xGotoParkPos        := ,
    pstDynamic          := ADR(gstDynPnP),
    pstDynamicAddAxes   := ADR(gstDynPnP_AddAxes),
    xRepeat             := ,
    xAlreadyPicked      := ,
    pstPickPos          := ADR(gstPickPos),
    pstPlacePos         := ADR(gstPlacePos),
    pstParkPos          := ADR(gstParkPos),
    xTeachPickPos       := ,
    xTeachPlacePos      := ,
    xTeachParkPos       := ,
    xAckPickOrPlace     := ,
    lrUpPosZ            := ,
    lrRadius            := ,
    udiPickTimer        := ,
    udiPlaceTimer       := ,
    xPickPosReady       := ,
    xPlacePosReady      := ,
    pCNC                => g_stIn.parCNCUser[3],
    xPick               => ,
    xPlace              => ,
    iPickCounter        => ,
    iPlaceCounter       => ,
    udiCycle            => ,
    udiCycleTime        => ,
    tCycleTimePickPlace => ,
    stKineDataRef       := gstGantryDataRef);
```

2. Autoloaded from the available CNCs in the project itself (up to 100) even if the array (g_stIn.parCNCUser) has places for up to 256 CNCs.



These CNCs have a lower priority, than hardcoded CNCs (see possibility 1 above).

If you have already more than 100 CNCs defined, no more CNCs will be loaded automatically.

But if you have for instance 50 CNCs hardcoded and more available in your Codesys project itself, it will fill up the list up to 100 CNCs in total.

2 Configuration of a linear gantry

The actual configuration is described in the previous section. The main differences are included below.

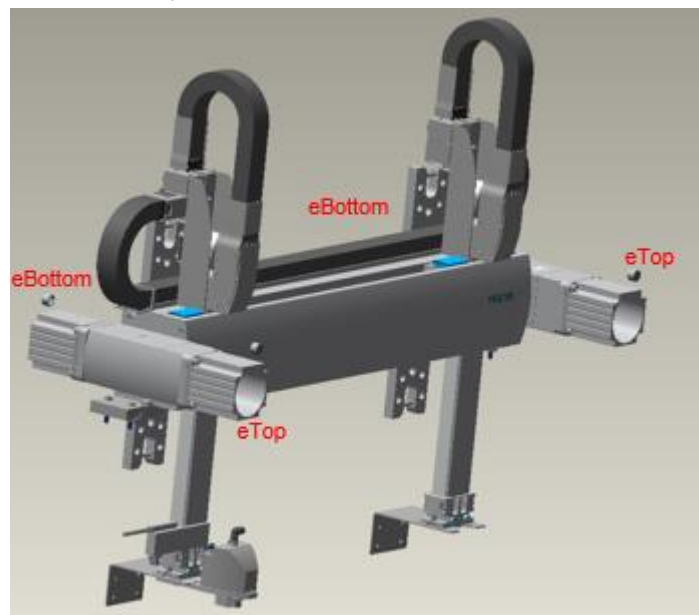
2.1 Linear gantry

DRIVE_CONFIG

The linear gantry is defined as a Y/Z gantry. If no X-axis is used, the corresponding place is replaced with a “0”.

```
(*****DRIVE_CONFIG*****)
g_stDrive      : FPosB.DRIVE_CONFIG := (
  arxAvailable      := [1,1,1,0,0,0], // Set available Axes of the System
  areControllerType := [FPosB.eFestoCMMF, FPosB.eFestoCMMF, FPosB.eFestoCMMF], // Controller Type
  areMotorOrientation := [FPosB.eTop, FPosB.eTop, FPosB.eTop], // Motor Orientation for H/T-Portal
  arstRef           := [0,ADR(D2),ADR(D3)], // Axis reference
  arxLimitAxes      := [0,0,0,0,0,0], //Activate the Axis that need a Dynamic Limit
  arlrFeedConstance := [0,0,0,0,0,0]; //Feedconstance of the Axis
```

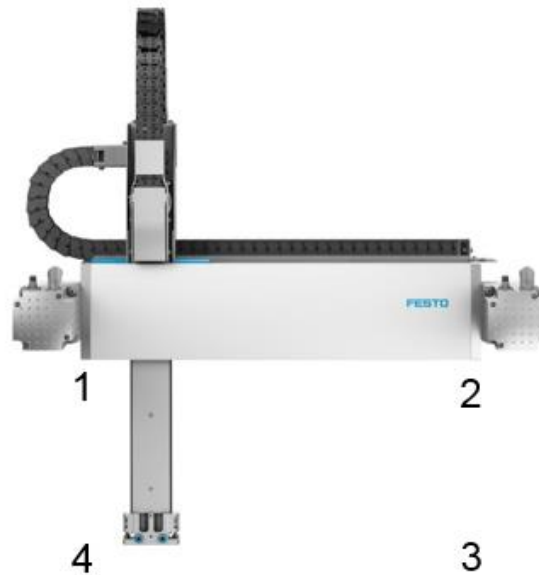
areMotorOrientation: The mounting position of the motors is selected in accordance with the kinematics.



HOMING_CONFIG

```
(*****HOMING_CONFIG*****)
g_stHoming      : FPosB.HOMING_CONFIG := (
  eHomingCorner      := FPosB.eCorner2, // Homing Corner for H/T-Portal
  stParkDynamic      := (lrVel:= 50, lrAcc:= 1000, lrDec:= -1000), // Danymic for moving to Parkposion after Homing
  areHomingPosSingleAxis := [FPosB.eHNLimitsMin, FPosB.eHNLimitsMin, FPosB.eHNLimitsMin], // HomingPos For Homing
  arxExcludeAxis     := [0,0,0,0,0,0], // Exclude Axis for the Homing precedure
  ariHomingOrder      := [0,2,1,0,0,0], // Set the Homing order for the system
  areHomingMethodeSingleAxis := [0,0,0,0,0,0], // Homingmethode for the drives
  arxSaveHomingOffsetToEncoder:= [0,0,0,0,0,0], // Only Triger it in the Visu after Homing to save the Position into the MontorController
  xMoveToParkPos      := TRUE, // Move to ParkPos after Homing
  xPortalAlreadyHomed := FALSE, // Preset the System Homed-bit
  stParkPos           := ( lrX:= 0, lrY:= 0, lrZ:= 0, // ParkPosition after Homing Done
                           lrA:= 0, lrB:= 0, lrC:= 0,
                           lrU:= 0, lrV:= 0, lrW:= 0),
  tHomingTimeOut      := T#60S); // TimeOut for Homing duration
```

eHomingCorner: In the case of the linear gantry there are four corners to which homing can take place:



ariHomingOrder: In this example first the Z-drive will move, afterwards the Y-drive.

It's recommended to use the homing procedure from the commissioning description of the EXCT, which can be found in our support portal. Usually the EXCT is used with multi-turn encoders, so the homing has to be done only once.

In the library the gantry can be set to already homed.

```

xMoveToParkPos := TRUE, // Move to ParkPos after Homing
xPortalAlreadyHomed := FALSE, // Preset the System Homed-bit
stParkPos := ( lrX:= 0, lrY:= 0, lrZ:= 0, // ParkPosition

```

KINEMATIC_CONFIG

```

(*****KINEMATIK_CONFIG*****)
g_stKinematic : FPosB.KINEMATIK_CONFIG := (
    eKinematicType := FPosB.eT_Gantry, // Kinematic Type

```

The kinematics type is correspondingly set to eT_Gantry.

2.2 XY-gantry

DRIVE_CONFIG

```

(*****DRIVE_CONFIG*****)
g_stDrive : FPosB.DRIVE_CONFIG := (
    arxAvailable := [1,1,1,0,0,0], // Set available Axes of the System
    areControllerType := [FPosB.eFestoCMMP, FPosB.eFestoCMMP, FPosB.eFestoCMMP], // Controller Type
    areMotorOrientation := [FPosB.eTop, FPosB.eTop, FPosB.eTop], // Motor Orientation for H/T-Portal
    arstRef := [ADR(D1),ADR(D2),ADR(D3)], // Axis reference
    arxLimitAxes := [0,0,0,0,0,0], //Activate the Axis that need a Dynamic Limit
    arlrFeedConstance := [0,0,0,0,0,0]; //Feedconstance of the Axis

```

At least the X and Y-axes must be available for the XY-gantry.

Motor orientation has no influence. The axes must be seen as single axes. The settings in FCT are used – use the same procedure as for the Z-axis in the sample application.

KINEMATICS

```
(*****KINEMATIK_CONFIG*****)
g_stKinematic      :   FPosB.KINEMATIC_CONFIG := (
                        eKinematicType         := FPosB.eXY_Gantry,           // Kinematic Type
```

The kinematics type is correspondingly set to eXY_Gantry.

All other settings have the same effect as is also the case with the single axis (Z-axis) in the sample application (H-Portal)