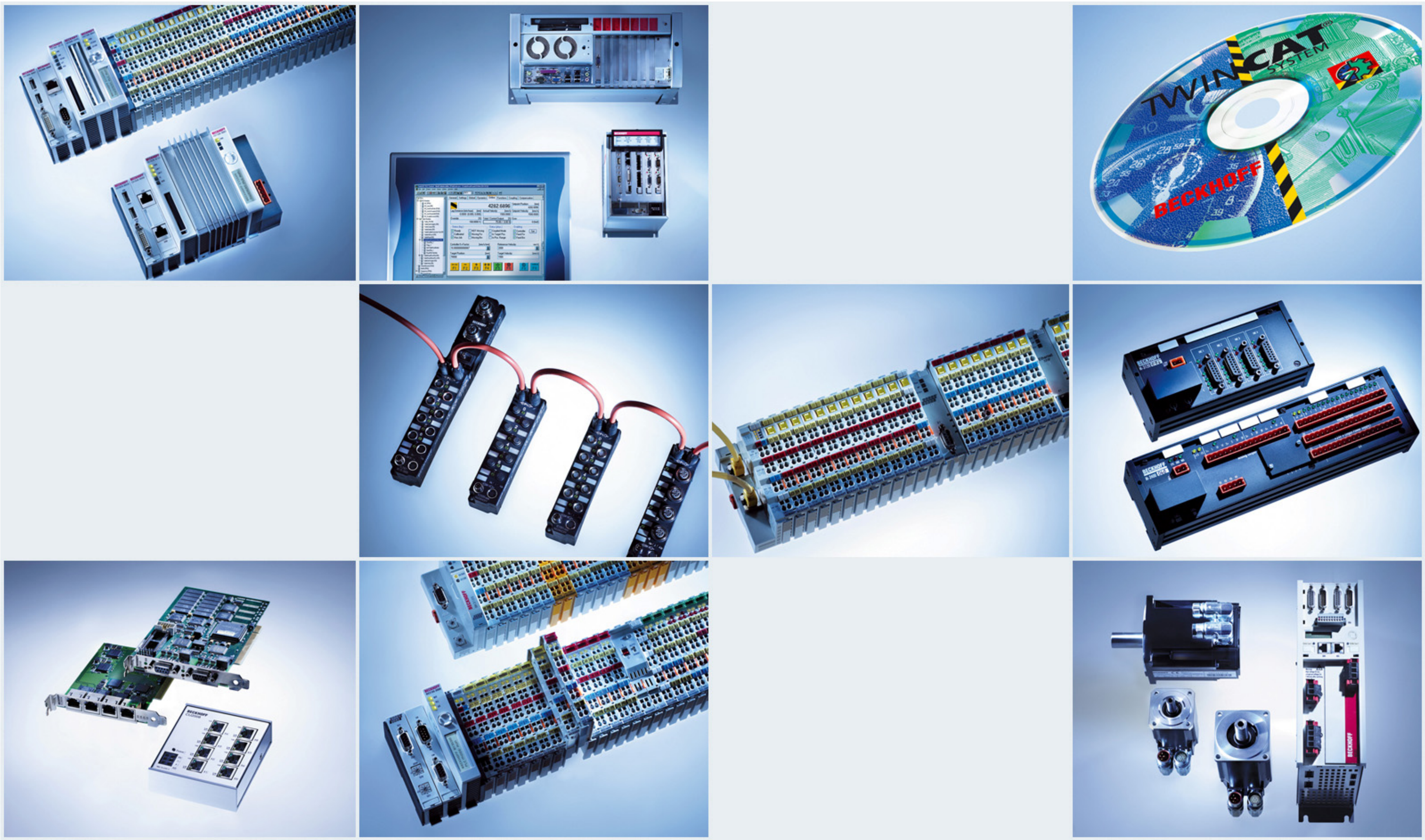
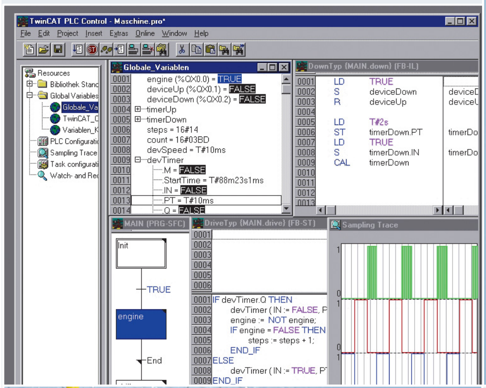
The Windows Control and Automation Technology

New Automation Technology



The Windows Control and Automation Technology

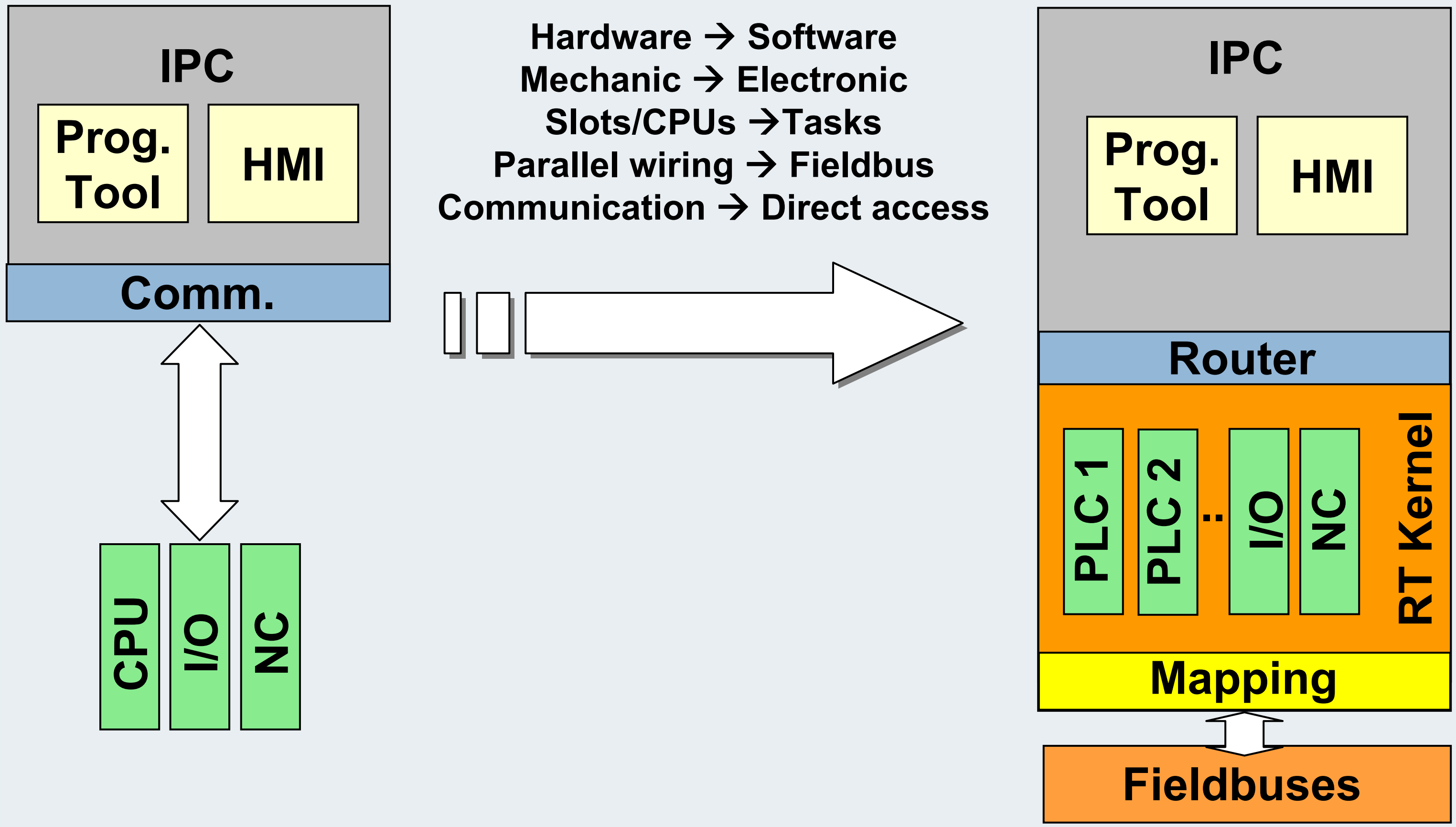


Content

- PC-based automation
- TwinCAT
  - Architecture
  - I/O
  - Control (PLC)
  - Motion (NC PTP)
  - Interpolated motion (NC I, CNC)
  - Connectivity

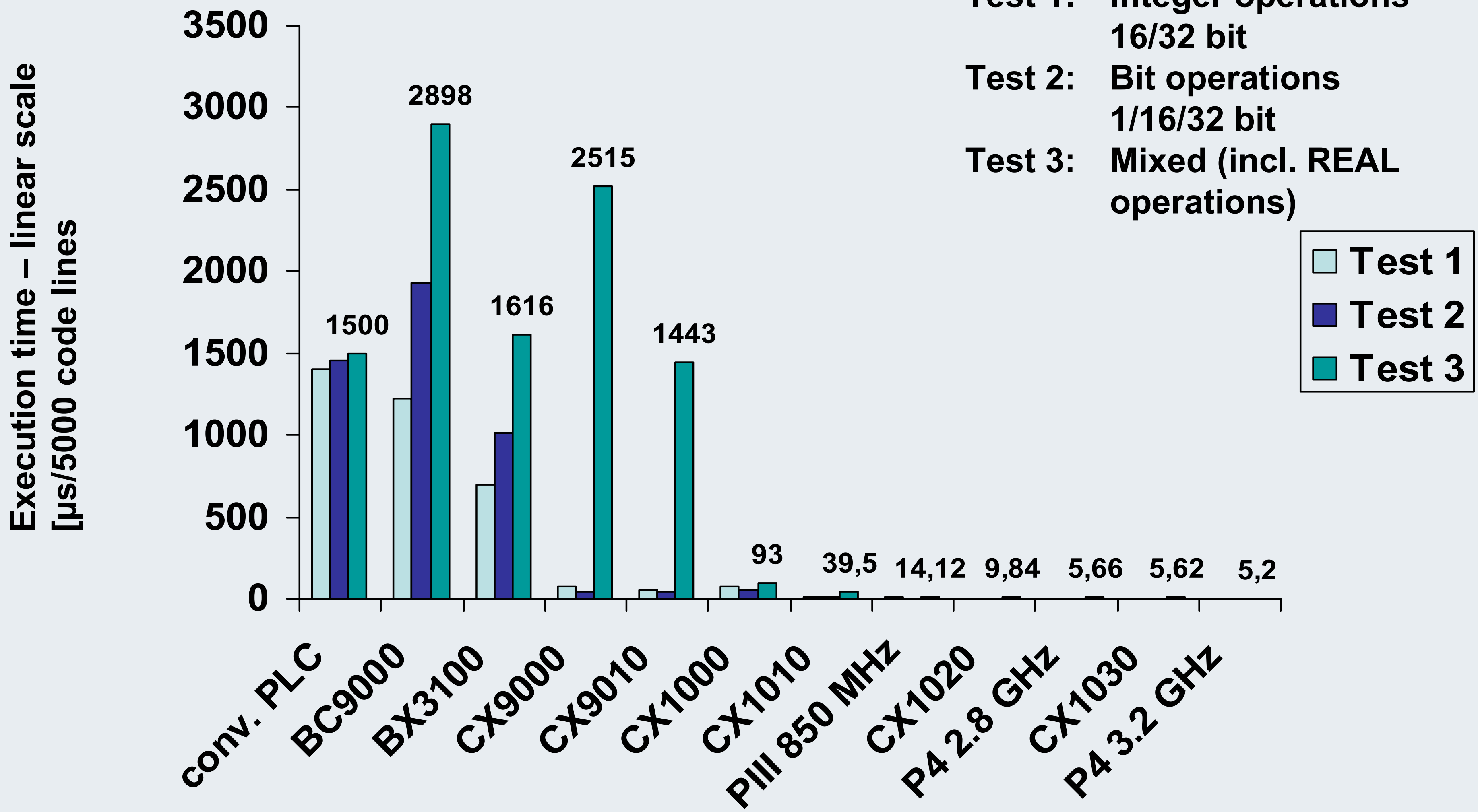
PC-based automation: development

Evolution in control



PLC processing times on different platforms

Test 1: Integer operations 16/32 bit
Test 2: Bit operations 1/16/32 bit
Test 3: Mixed (incl. REAL operations)

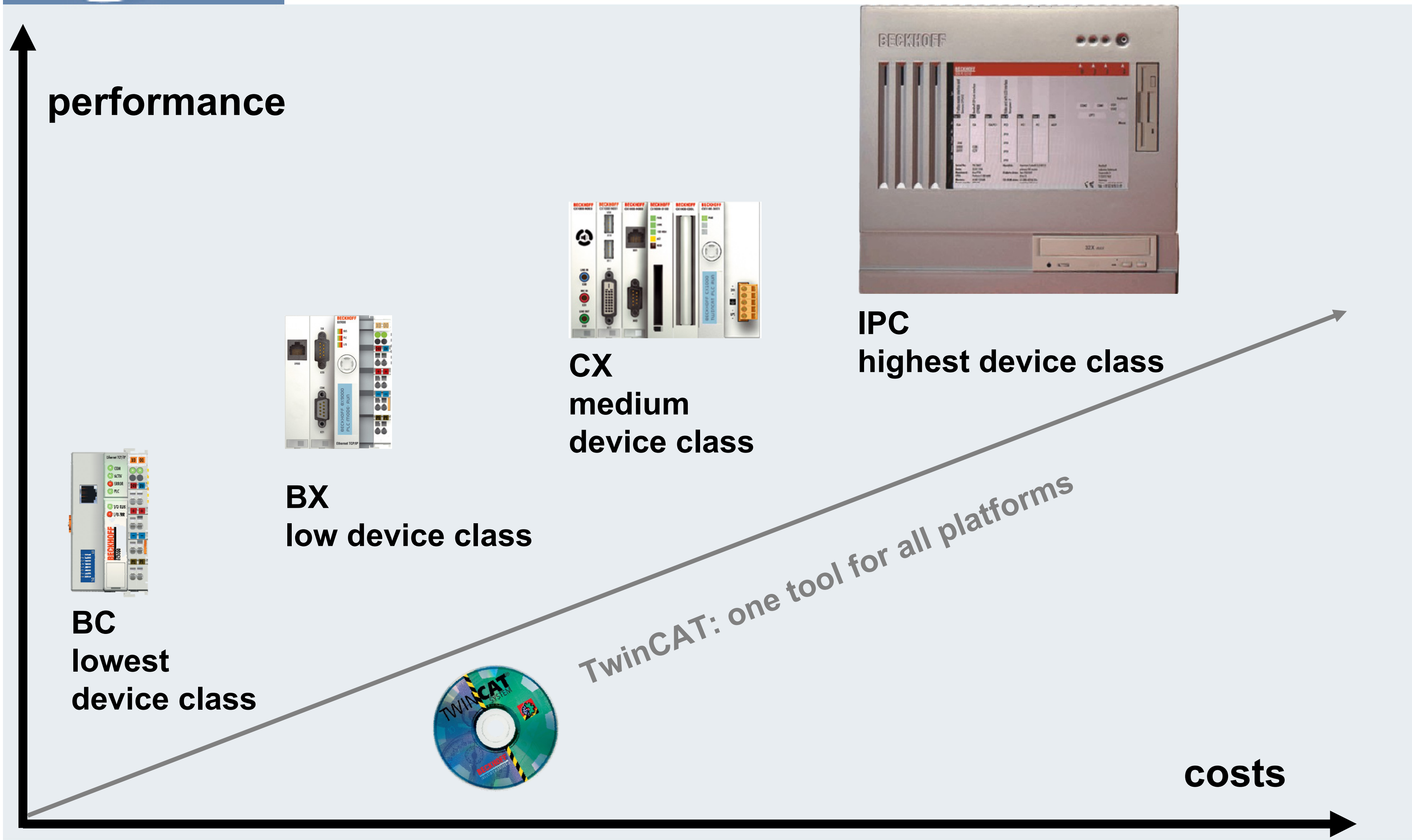


System overview

- TwinCAT – integrated engineering and runtime for
  - control
  - motion
  - technology

- TwinCAT – running on different hardware platforms
  - PC → high performance
  - CX → medium control level
  - BX → lower control level
  - BC → lowest control level/low price

Beckhoff control devices in 4 performances



TwinCAT architecture: platform PC

PC-based automation
TwinCAT
Architecture
I/O
Control
Motion
Interpolated Motion
Connectivity

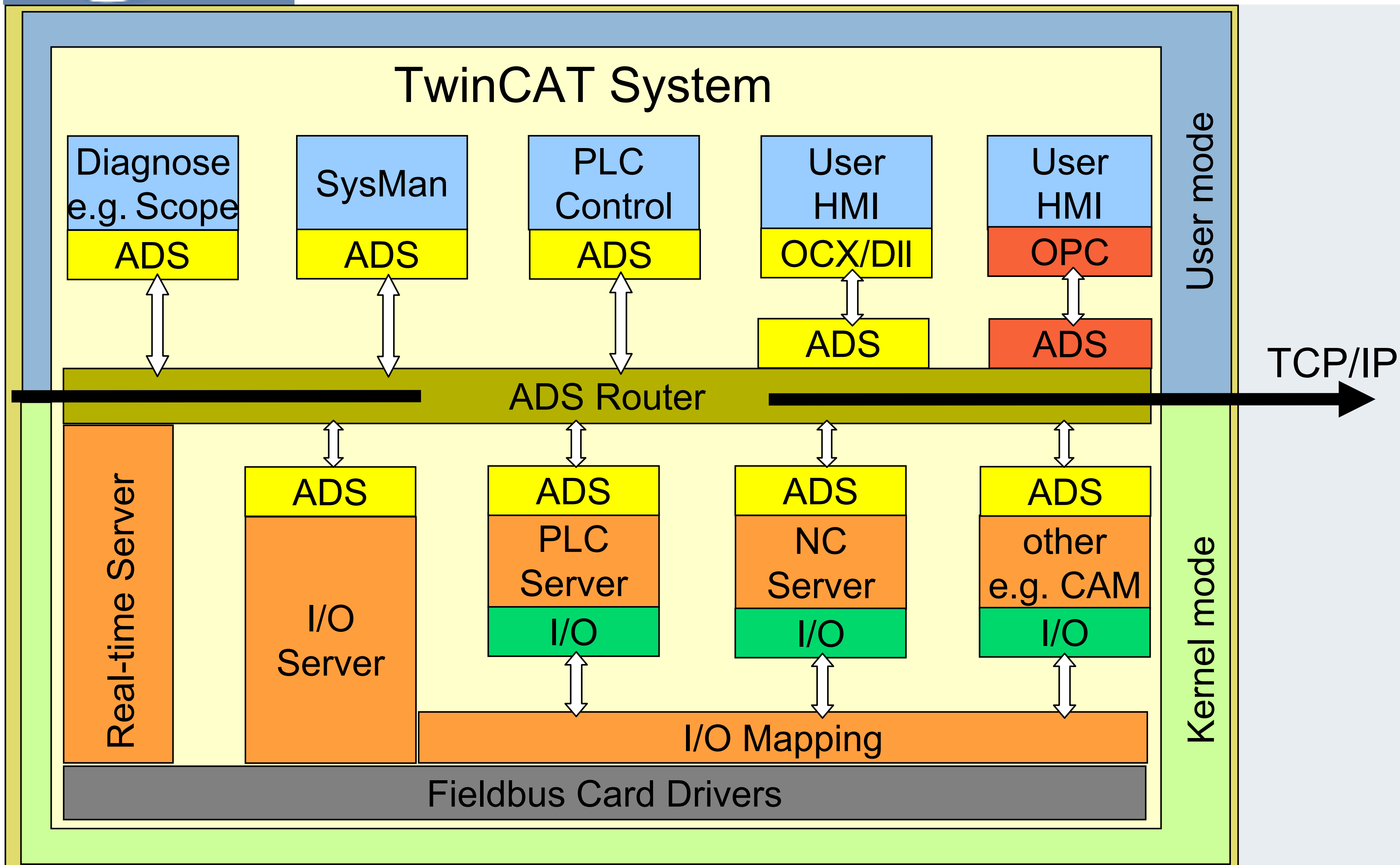
TwinCAT

- does not modify Windows
- needs no special hardware
- turns standard Windows to a real-time OS
- full access to Windows user interface via OCX, Dll
- remote access via TCP/IP

PC platform

- standard hardware, best performance
- use of PC resources
- use of mainstream operating system (Windows)
- easy integration into office networks
- open fieldbus communication

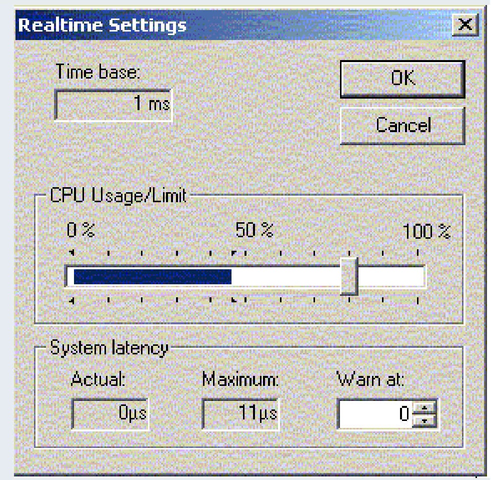
TwinCAT architecture: platform PC

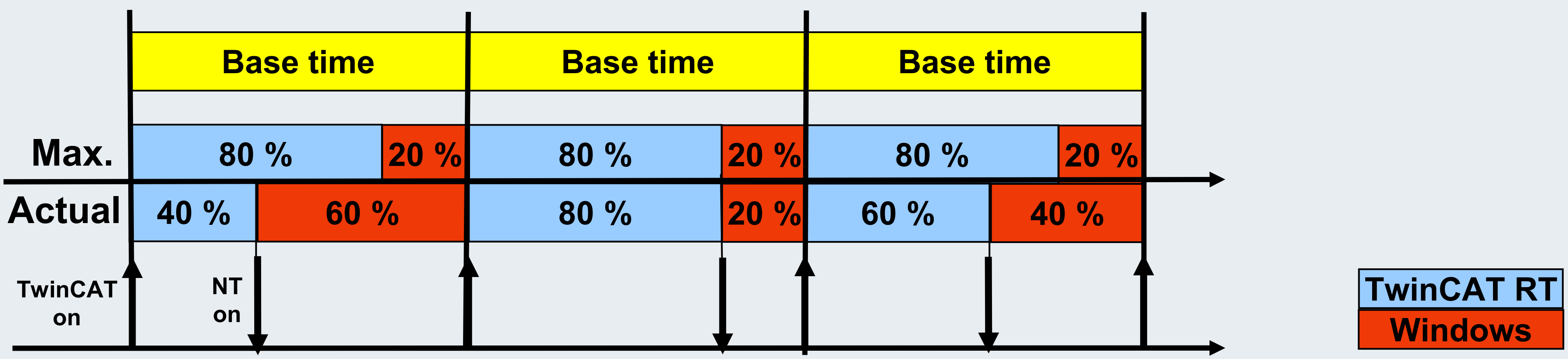


TwinCAT architecture: platform PC

TwinCAT real-time

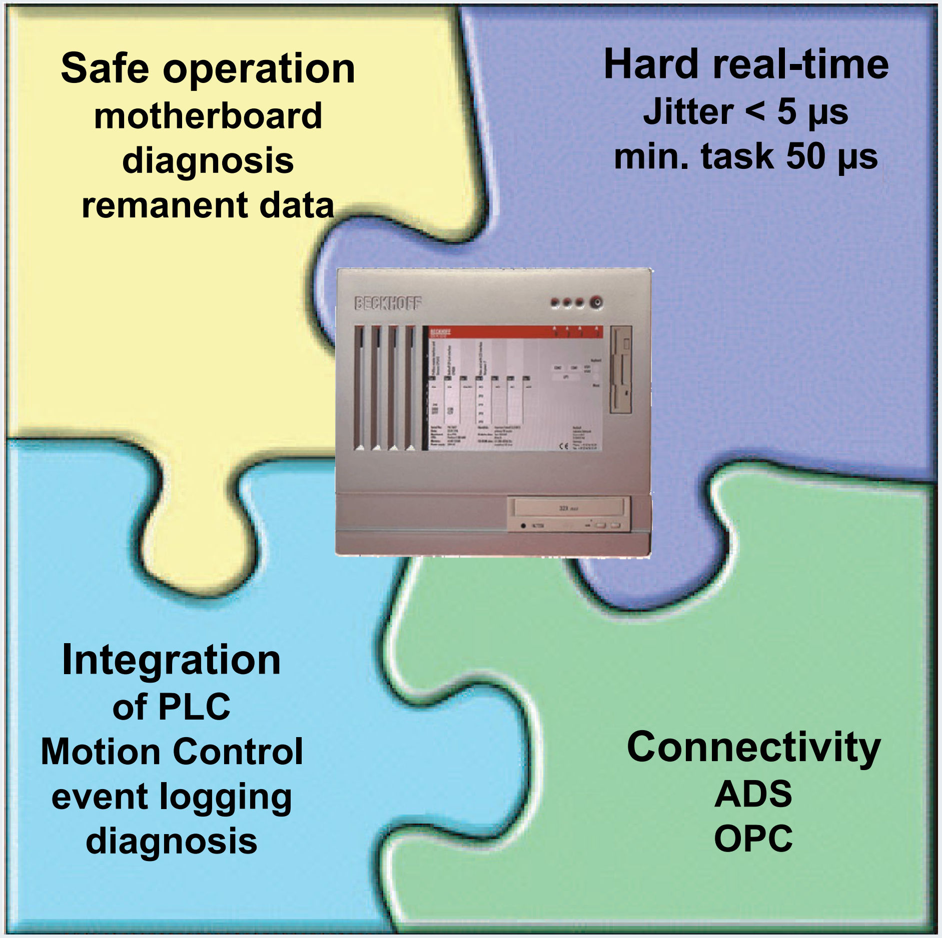
- cycle times down to 50 µs
- latency times < 5 µs (P4)
- adjustable real-time ratio to Windows
- message when latency time is too high





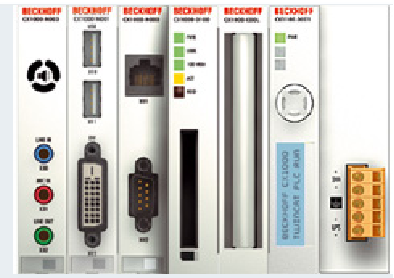
TwinCAT architecture: platform PC

Switch from office PC to a IPC with PLC and Motion Control



TwinCAT architecture: platform CX

PC-based automation
TwinCAT
- Architecture
- I/O
- Control
- Motion
- Interpolated Motion
- Connectivity



TwinCAT

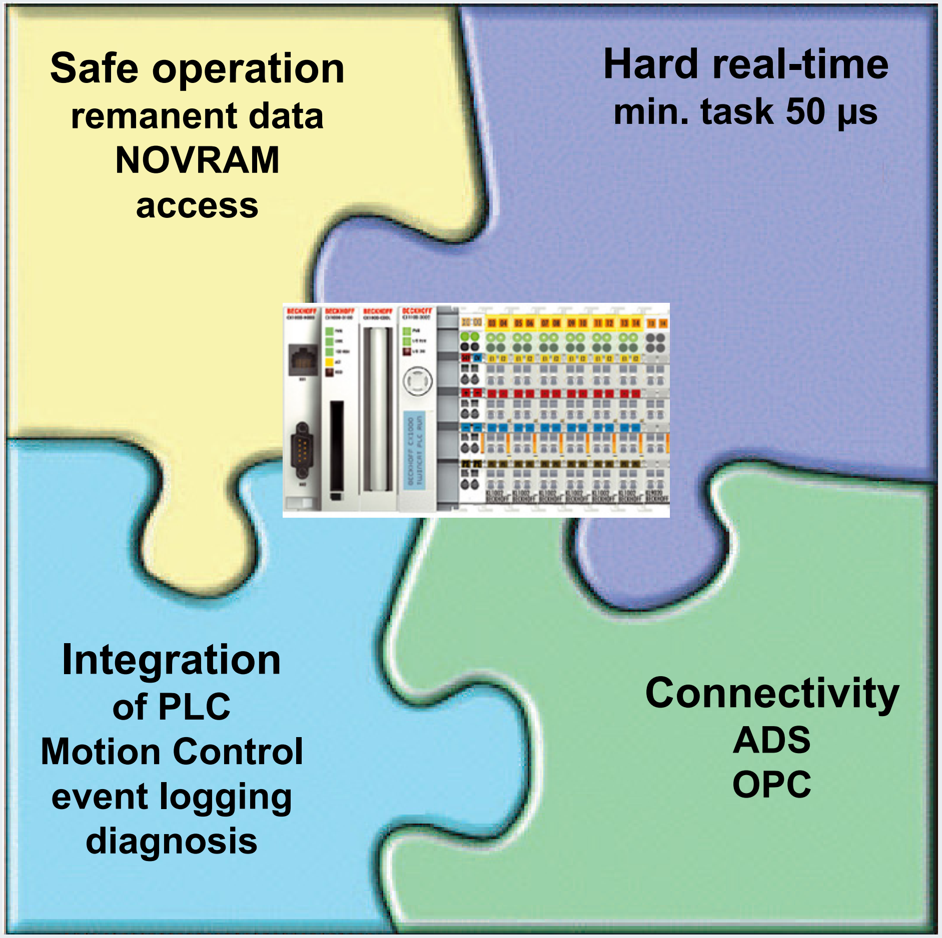
- pure software solution
- uses Windows CE real-time
- remote configuration, setup, programming

CX

- embedded hardware
- OS: Windows CE <u>or</u> Windows XP Embedded
- no rotating media, fanless
- direct access to terminal I/O
- more than one fieldbus
- master and/or slave

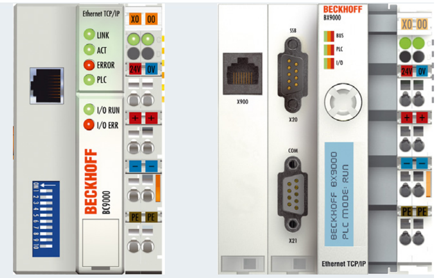
TwinCAT architecture: platform CX

Switch from Embedded PC to a IPC/PLC



TwinCAT architecture: platform BX/BC

PC-based automation
TwinCAT
Architecture
I/O
Control
Motion
Interpolated Motion
Connectivity



TwinCAT

- download program
- online debugging
- remote configuration, setup programming
- remote access via fieldbus

BX/BC

- embedded hardware
- fieldbus slave
- embedded operating system
- direct access to terminal I/O

TwinCAT and standards

- TwinCAT implements and uses established industrial standards for automation
- operating systems:
  - Windows NT/NT Embedded
  - Windows 2000
  - Windows XP/XP Embedded
  - Windows CE
- programming: IEC 61131-3
- Motion Control: PLCopen Motion Control function blocks
- vertical integration: OPC
- connectivity: fieldbuses

TwinCAT and standards

Benefit to customer

- quick orientation: same look and feel
- less training costs
- less maintenance costs
- reuse of software modules

Summary

- shorter delivery times, decreased costs

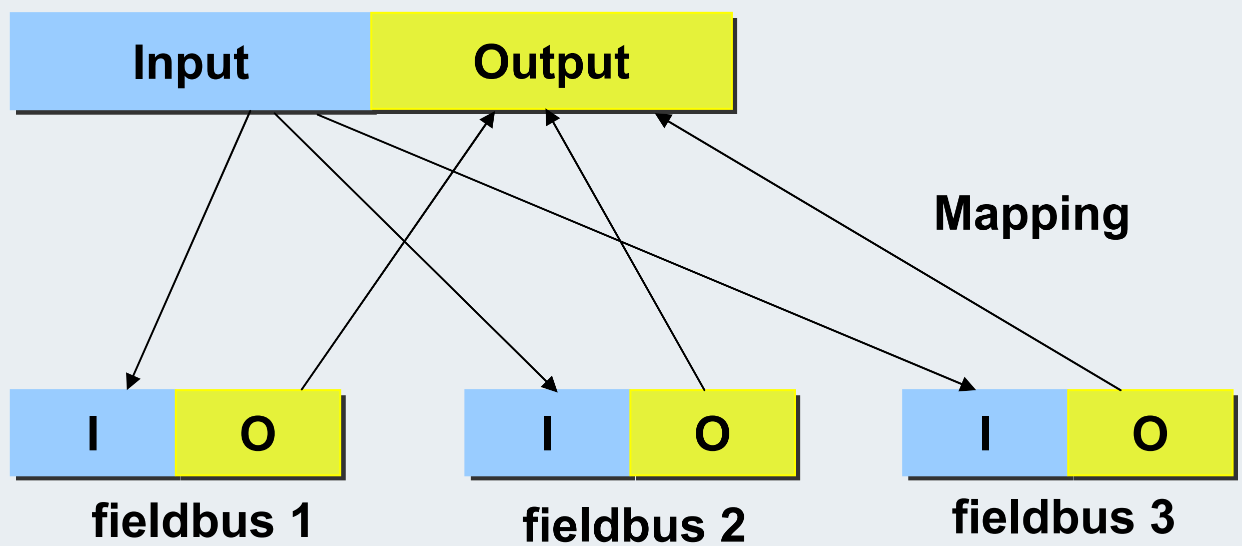
TwinCAT architecture

TwinCAT I/O system

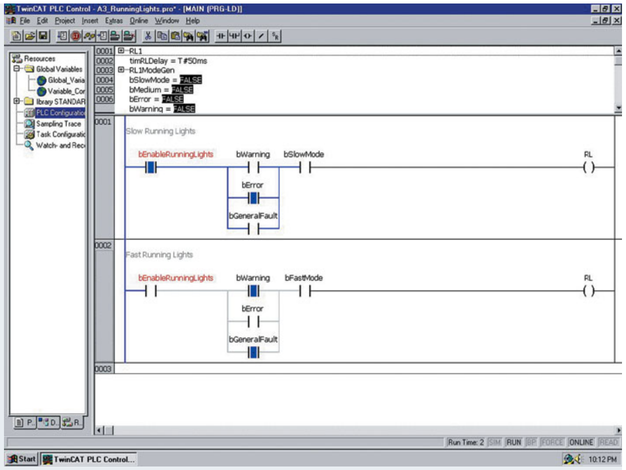
- open for all major fieldbuses
- support of PC hardware interfaces
- easy setup and diagnosis
- mapping from logical to physical I/O

DeviceNet
CANopen
Modbus
RS 232
RS 485
Ethernet TCP/IP
LIGHTBUS
USB
SERCOS interface
INTERBUS
PROFIBUS
ControlNet
EtherCAT

TwinCAT process image



TwinCAT control



Modular structure

- max. 4 PLCs on a PC

Timing

- max. 4 tasks in one PLC

Easy setup and maintenance

- online program change
- full debugging features:
  - breakpoint, monitoring, powerflow
  - scope

Choose the favourite programming language

- all IEC 61131-3 languages (IL, ST, FBD, LD, SFC)

TwinCAT motion

Shift from mechanical to electronical system

- mechanical cam → electronic cam
- mechanical gear → electronic gear
- mechanical clutch → electronic clutch
- mechanical cam shaft → electronic cam shaft
- “flying saw”

TwinCAT motion

Benefit

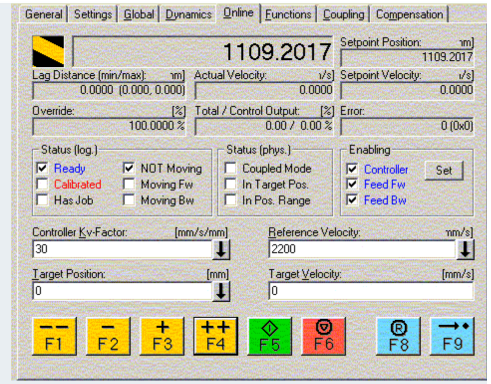
- greater flexibility
- increased machine output
- reduced setup time – no mechanical modification
- decreased stock – no different mechanical parts

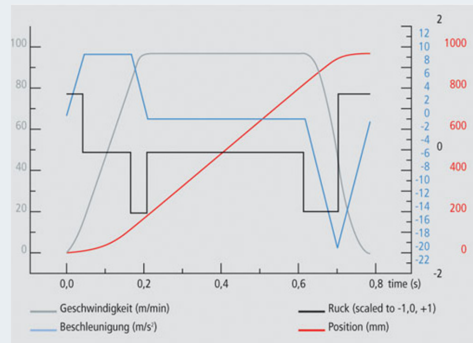
Summary

- shorten delivery/development time, decreased costs
- TwinCAT supplies all of this in one tool/run-time.

TwinCAT motion: NC PTP

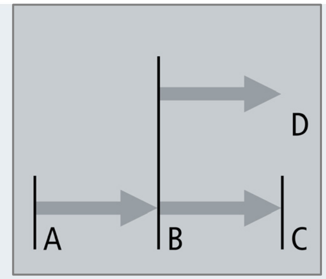
- easy setup and maintenance
- open for all axis types
  - servos
  - stepper
  - DC motors
  - switching axes
  - hydraulic axes
- several encoder
  - digital encoders: SERCOS, SSI
  - analog: ±10 V
- several controller:
  - P, PI, PID
- additional functions:
  - digital cam switch, flying saw, superposition

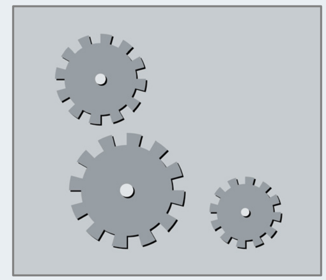


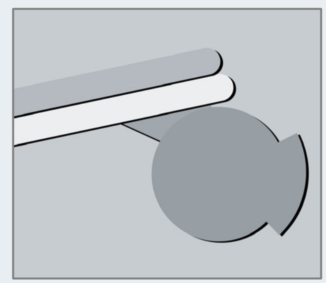


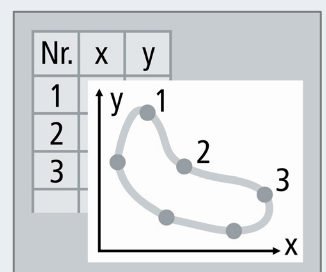
TwinCAT motion: NC PTP

- Point-to-point movement
- Gearing
- Digital cam switch
- Camming
- Superposition
- Flying saw







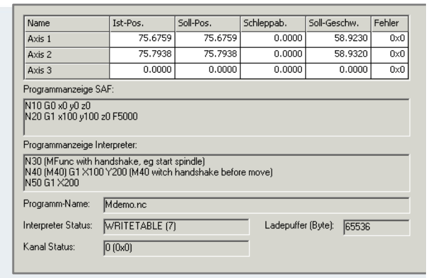


TwinCAT interpolated motion: NC I

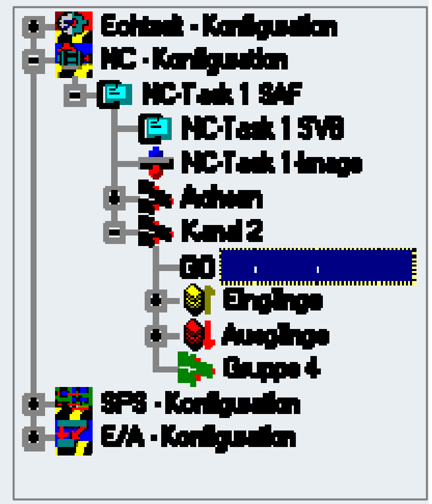
- PC-based automation
- TwinCAT
  - Architecture
  - I/O
  - Control
  - Motion
  - Interpolated Motion
  - Connectivity

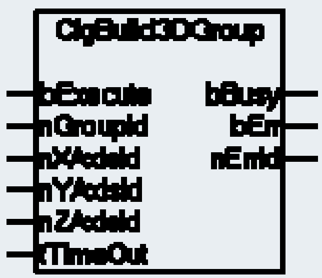
Interaction with PTP movements

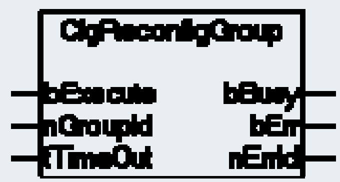
- interpolated movements for 3 axes plus 5 auxiliary axes
- programming in DIN 66025 code
- technological features
- easy access to PTP axes
- easy to use FB interface



Name	Ist-Pos.	Soll-Pos.	Schleppab.	Soll-Geschw.	Fehler
Axis 1	75.6759	75.6759	0.0000	58.9230	0x0
Axis 2	75.7938	75.7938	0.0000	58.9320	0x0
Axis 3	0.0000	0.0000	0.0000	0.0000	0x0





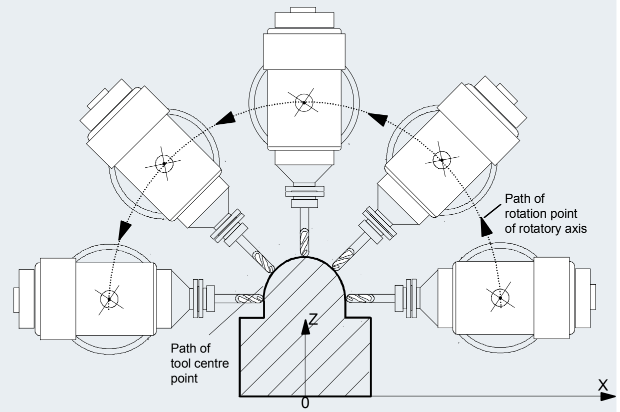


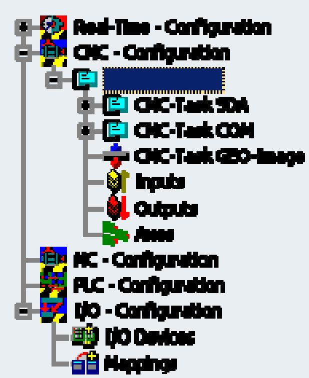
TwinCAT interpolated motion: CNC

Interaction with PTP movements

- interpolated movements for up to 32 axes in one channel
- programming in DIN66025 code
- different transformations

```
N00 #KIN ID [1]
N10 #RTCP ON
N20 G01 G18 X0 Y0 Z0 B90 F500
N30 X-4
N40 G02 X-20 I-40 B-90 F2000
N50 .................
.........................
```

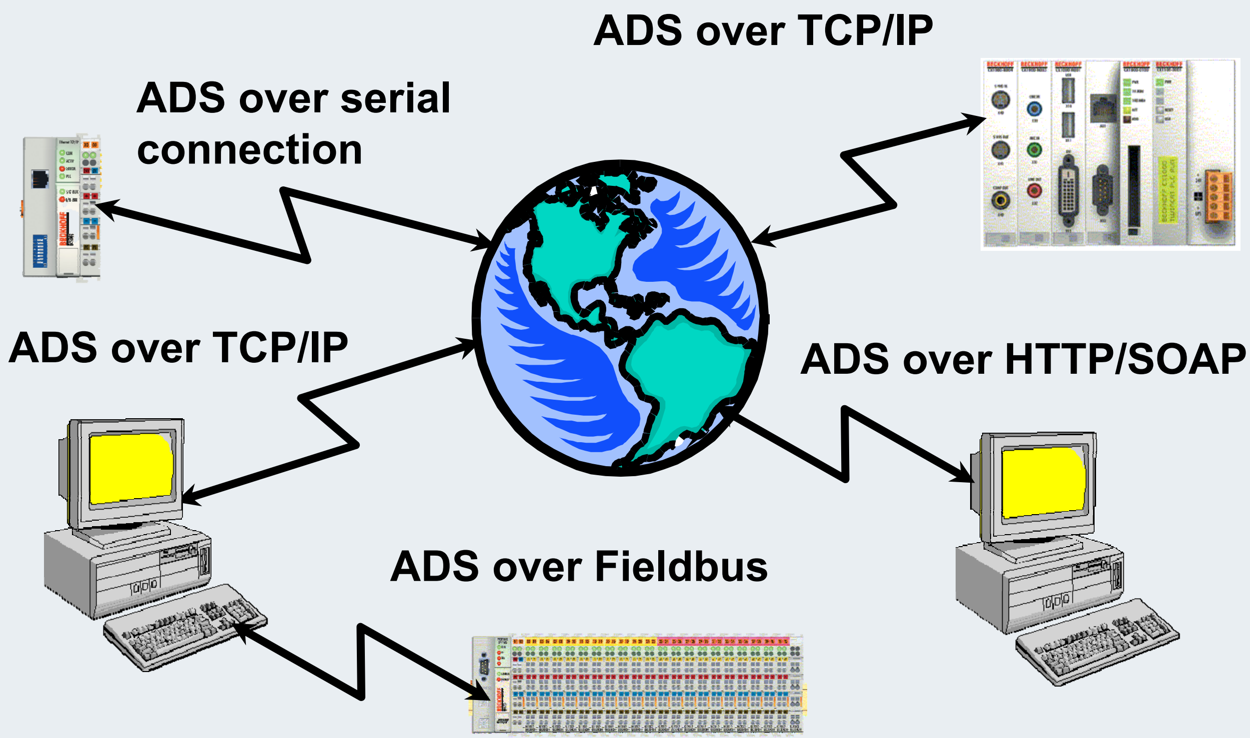




TwinCAT connectivity

Easy to use communication standard:
- **ADS (Automation Device Specification)**

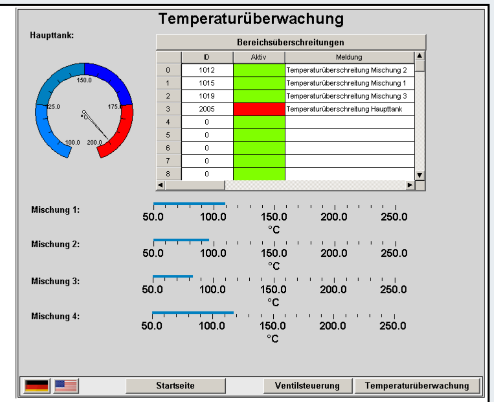
Access to ADS with standard Windows mechanism:
- **ActiveX control, Dll, .Net, ASP, OPC**

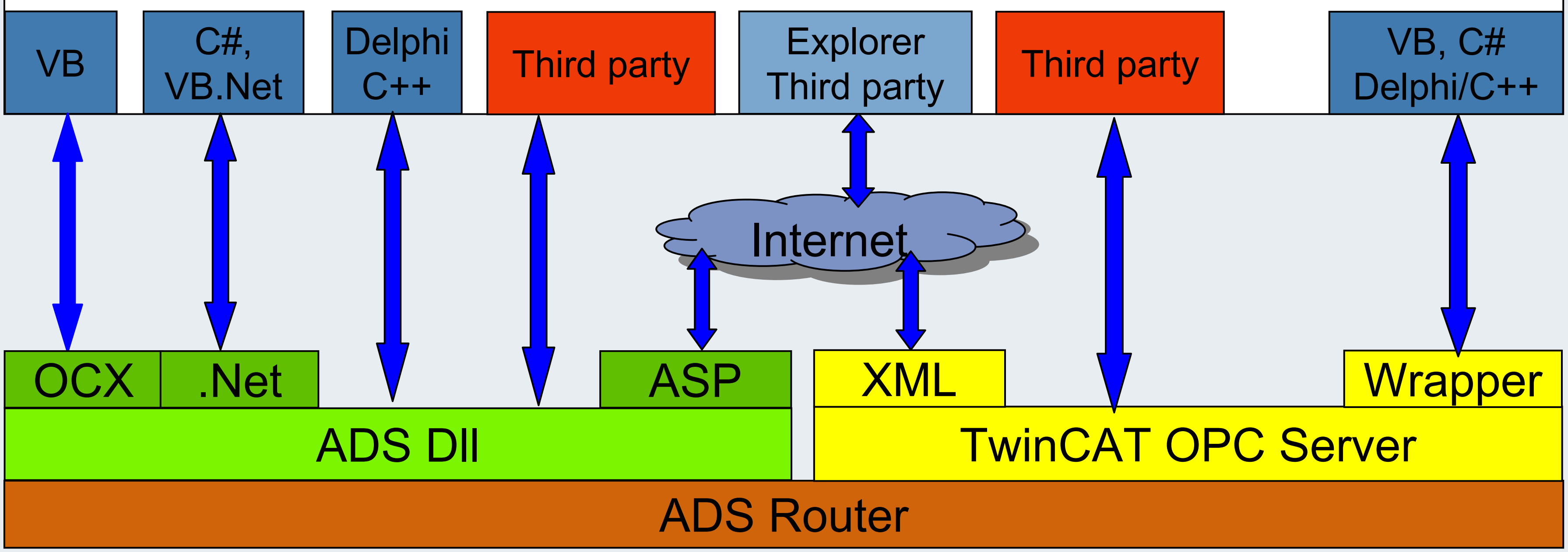


TwinCAT communication

Beckhoff ADS interface

- open, simple, for free, documented
- many Scada providers
- standard interface OPC
- DA/AE/XML





TwinCAT | The universal software platform for all control requirements

Beckhoff New Automation Technology

The IEC 61131-3

The IEC 61131-3

IEC 61131-3

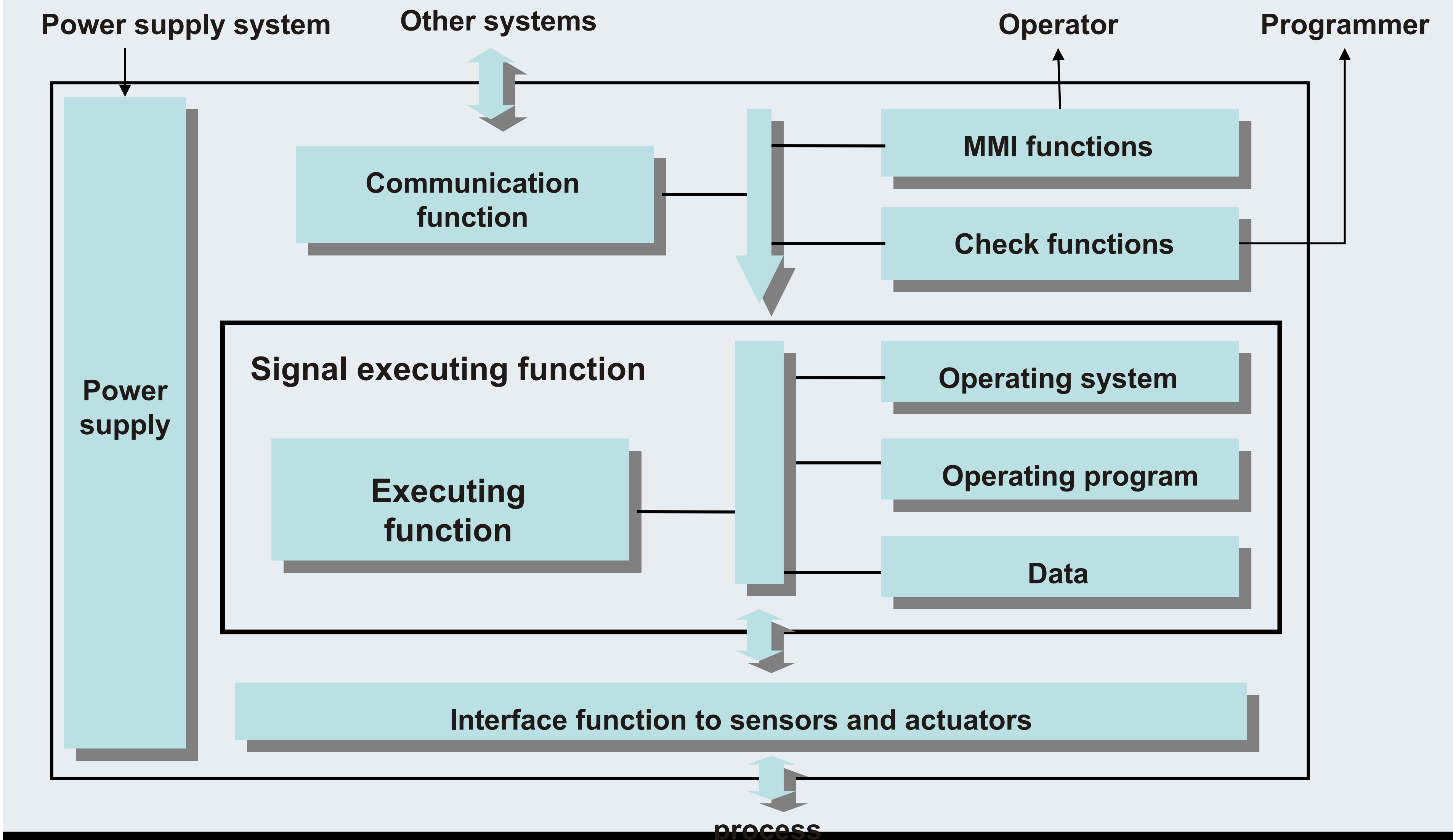
-1 General definitions and typical function (cyclic processing, process image input and output)	-2 Environmental conditions and conditioning classes of the control and the programming devices. (temperature, air humidity)	-3 Rules for using and implementation of PLC programming languages
-4 Guide line for the system analysis of the user, the system selection, the realisation of the application, as well as maintenance and servicing	-5 Definition of the communication via funcion blocks and communication via access paths (additionally to –3)	-6 Communication via fieldbus. -7 Fuzzy systems in the PLC

Standard guide

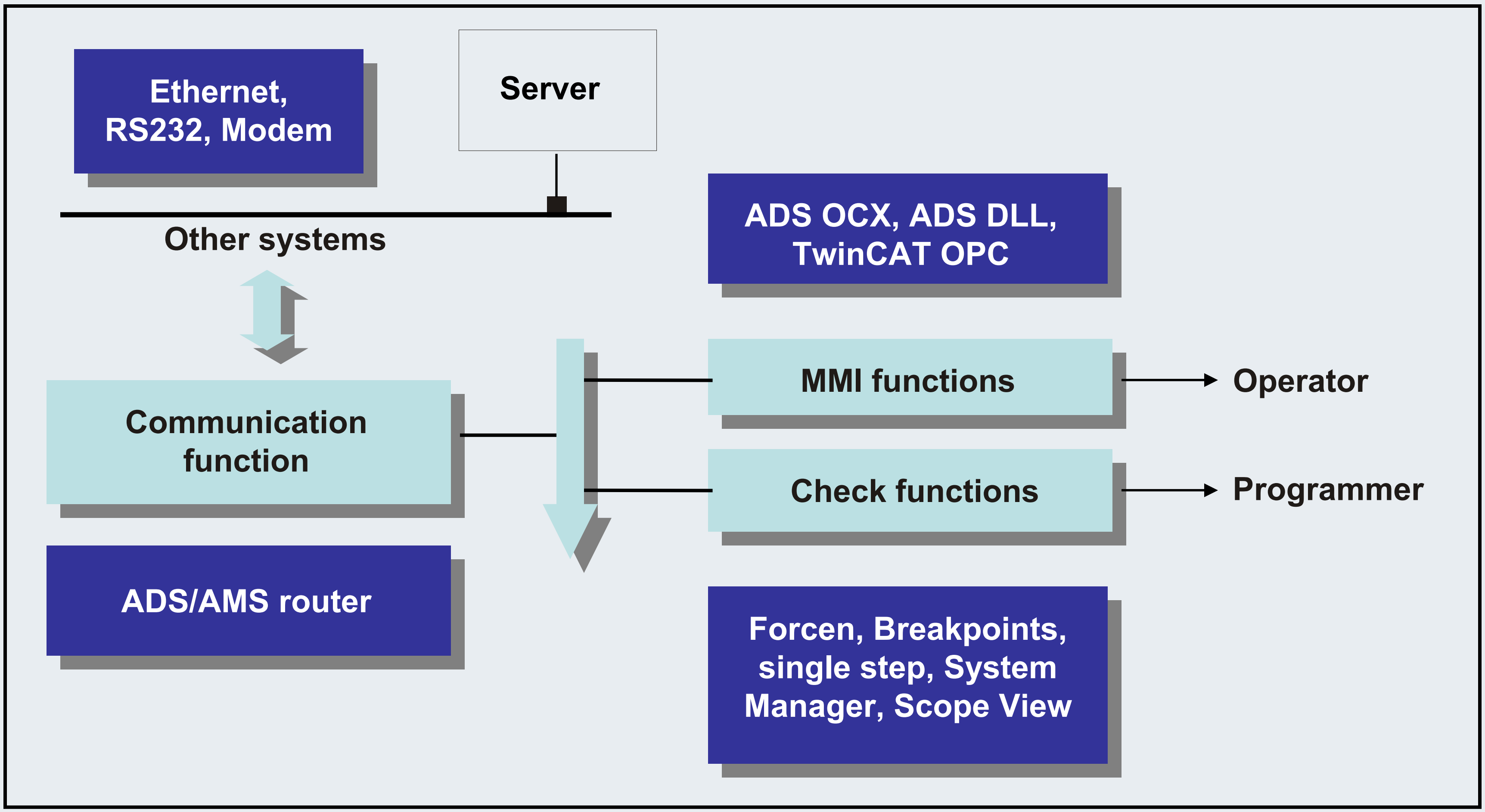
The PLCopen contains 3 devaluation compatible compliance classes:

Base level	Contains IL, ST, SFC, CFC (in preparation) a few data types, standard operators, functions, function blocks as well as local variables
Portability level	Data exchange format (8 bit ASCII). Data types with 32 bit strings, Arrays and all functions and operators based on this data type.
Full compliance level	Here the supreme compatibility degree must exist.

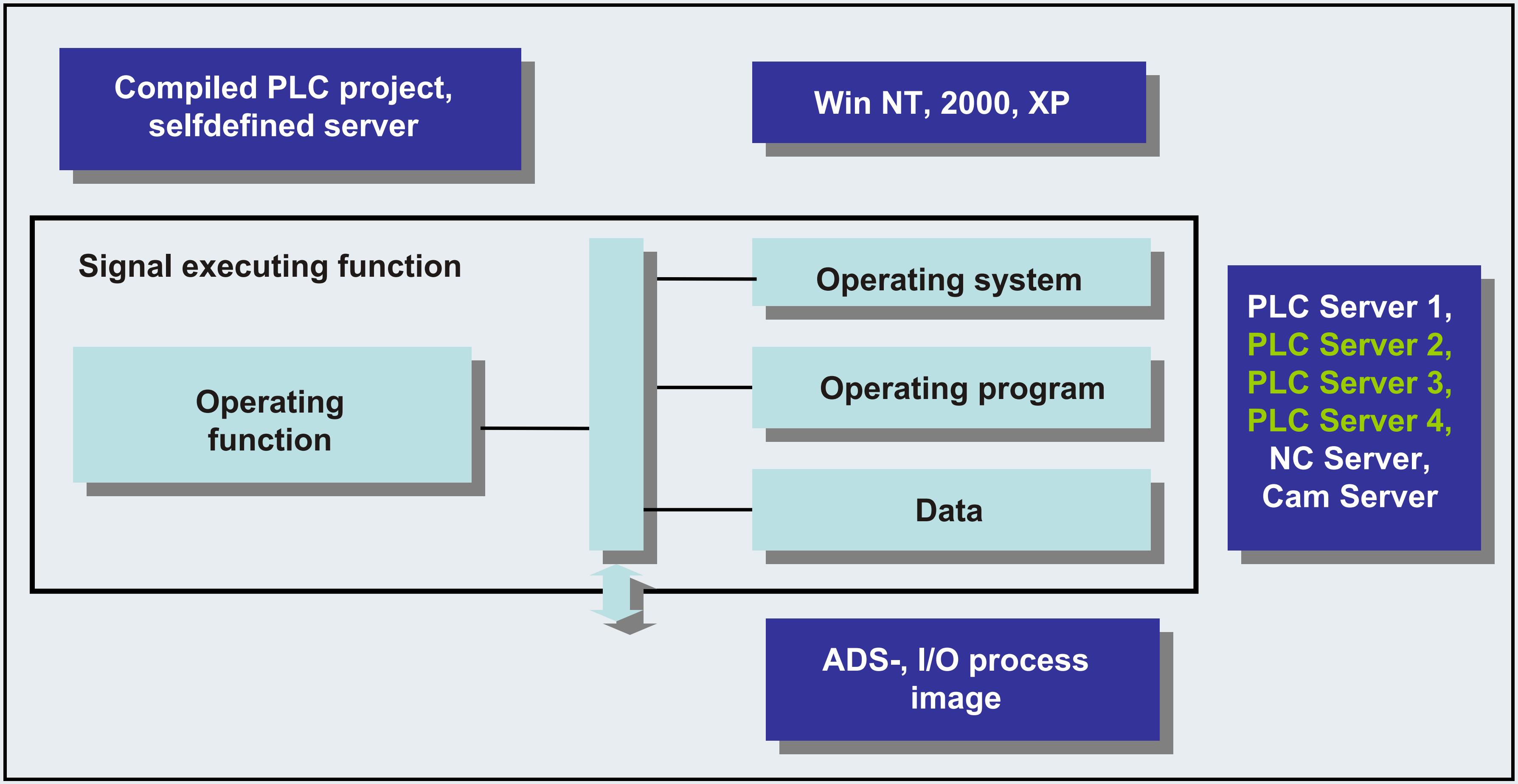
Functional structure of a PLC



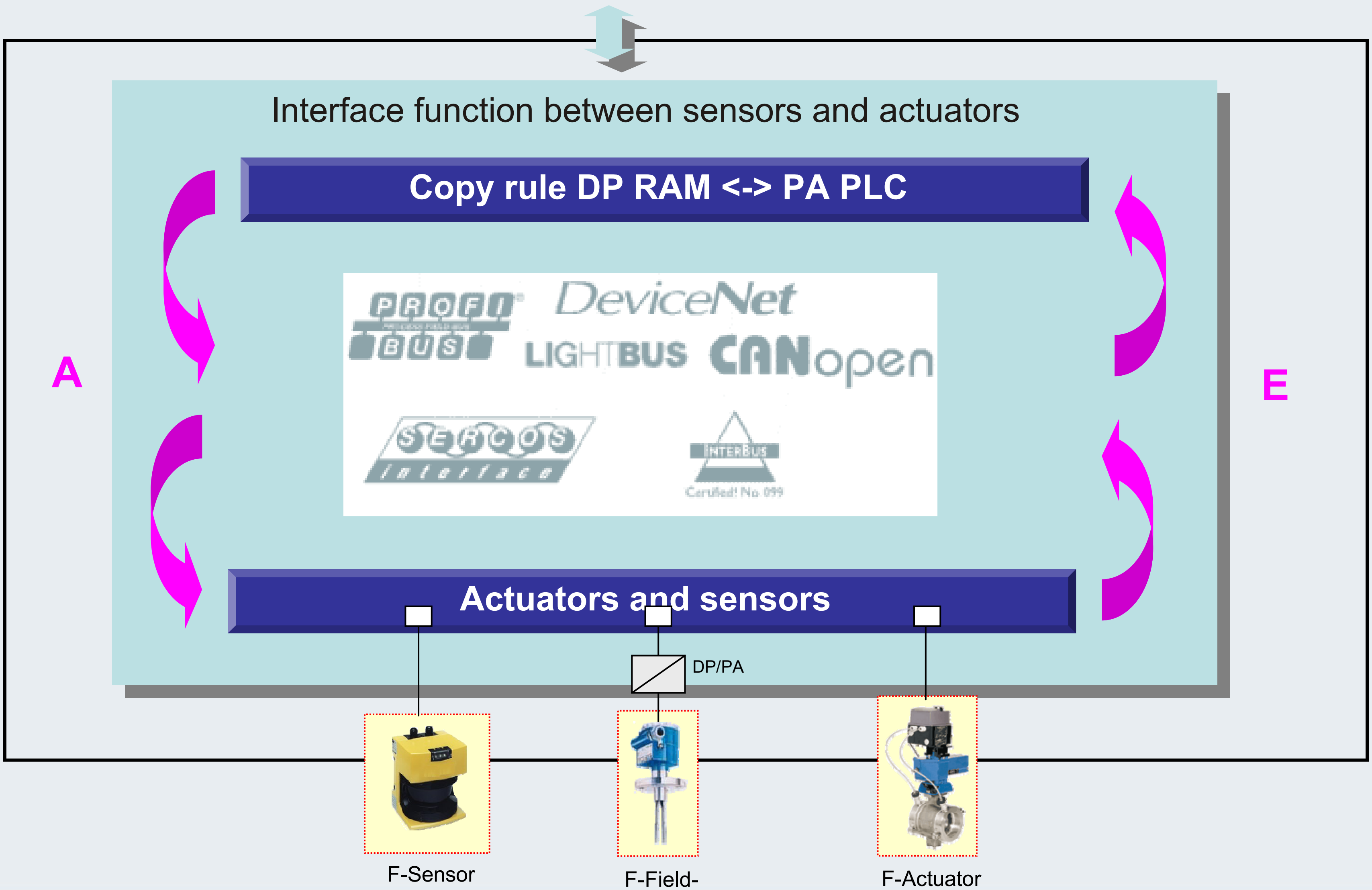
Communication functions



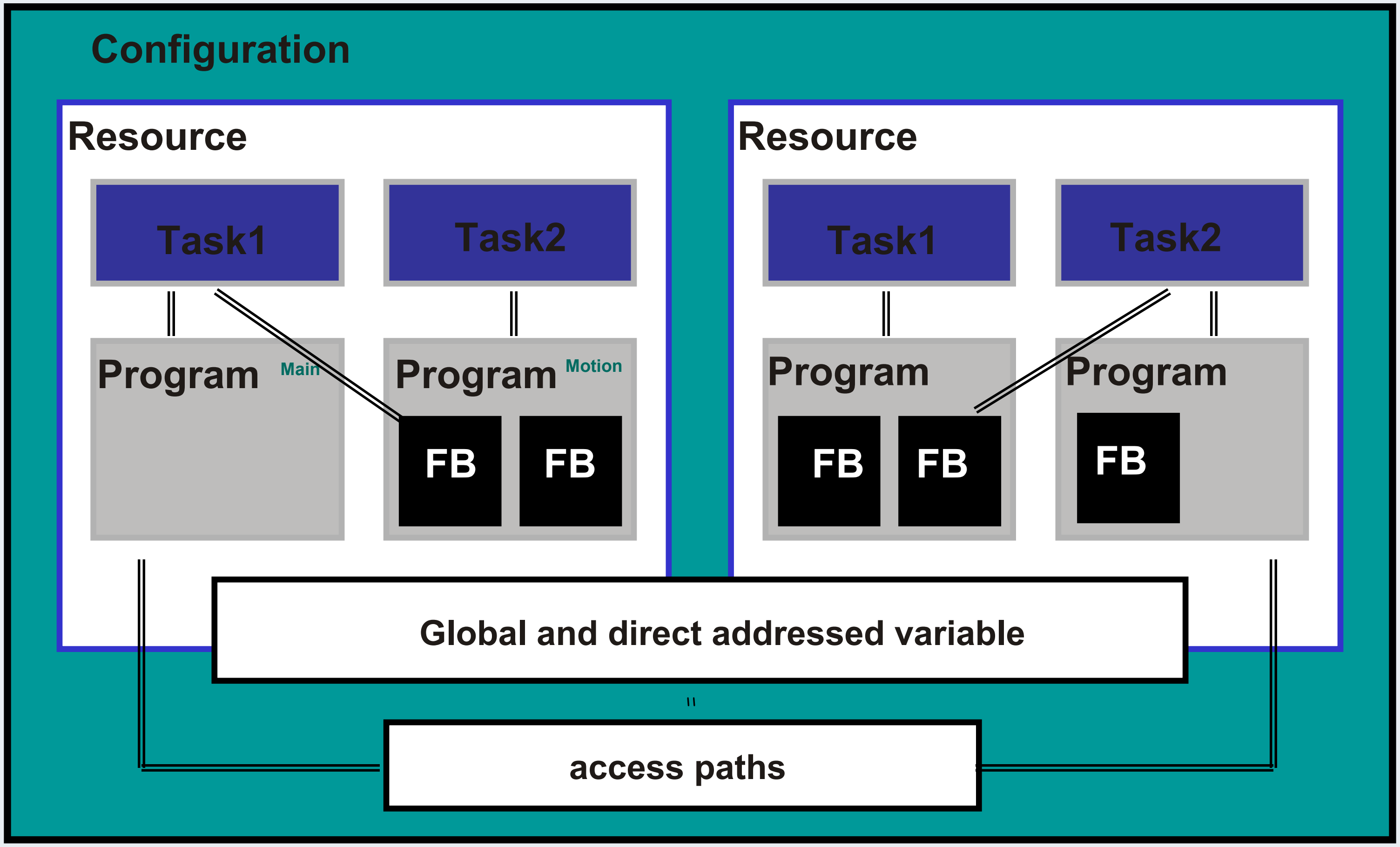
Signal executing function



Interface function between sensors and actuators



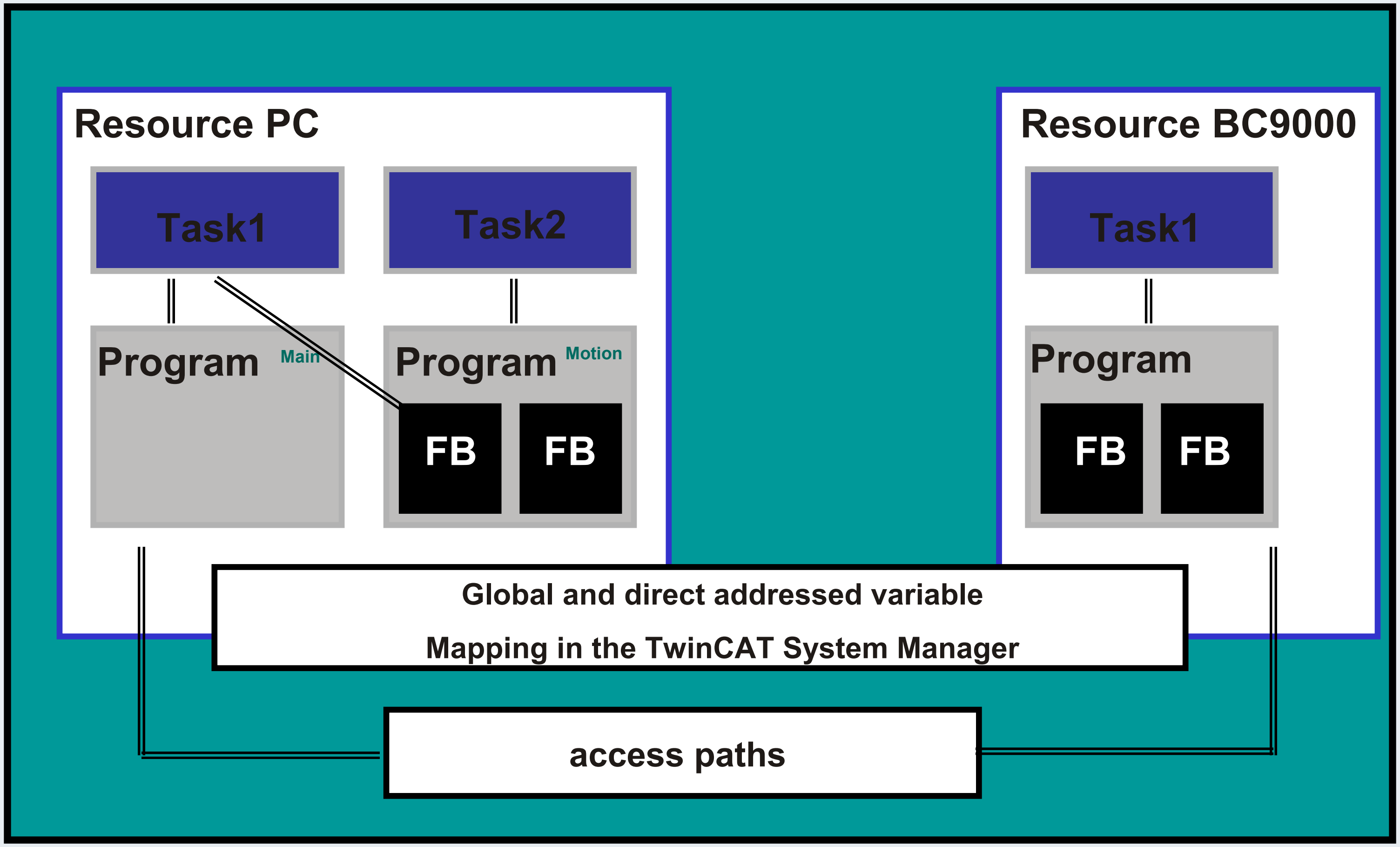
Software model



Software model Example

Example PC PLC with 1 run time und zwei Task 1 BC900 (Ethernet Controller)

Configuration



Identifier

Identifier serves to the individual name assignment for variables, data types, functions...

- The identifier begins with a letter or a underscore
- Followed by numbers, letters and underscore
- No difference between capital letters and small letters

Not allowed

- Special characters (!,“,§,$..)
- Blank character
- Sequential underscores
- mutated vowel

Prefix

Prefixes make the handling of the identifier easier. Here some suggestions:

Hungarian notation: Write part words together. The first letter of a part word must be a capital letter.

b – Boolean	bEndschalterLinks	
r – Real	rSollPosition	
s - String	sRxDaten	
ST_ - Declaration of structures		
st - Initialisation of structures	ST_MotorDaten	(declaration)
FB_ - Declaration of function blocks	stM1Parameter	(instance)
fb – Initialisation of function blocks		
M_ - Declaration of methods	FB_Ueberlast	(declaration)
	fbM1Ueberlast	(instance)

Key words and comments

Key words are preset indentifer by the IEC61131-3.
They are fixed components of the syntax and must not be used for other purposes.

TRUE, FALSE, AND, FUNCTION,...

Using the option Auto format, the keywords are written in capital letters.

The comments are limited with the characters (* at the beginning and *) at the end.
Comments can be placed there, where blank characters are allowed.
Exception: inside character string literals.

```
(*digital inputs*)
bStart AT%IX0.0:BOOL;(*Machine start*)

(*analog inputs*)
TemK1 AT%IW10(*Byte 10-11*):WORD;
```

Elementary data types

Type	ANY-Type	Key word	Data width (Bit)	Initial	Value range
Boolean	ANY_Bit	BOOL	1	FALSE	TRUE/FALSE
Bit string(8)		BYTE	8	0	0..16#FF
Bit string(16)		WORD	16	0	0..16#FFFF
Bit string(32)		DWORD	32	0	0..16#FFFF_FFFF
Short integer	ANY_Num	SINT	8	0	$-2^{7}...2^{7}-1$
Integer		INT	16	0	$-2^{15}...2^{15}-1$
Double integer		DINT	32	0	$-2^{31}...2^{31}-1$
Unsigned short integer		USINT	8	0	$0...2^{8}-1$
Unsigned integer		UINT	16	0	$0...2^{16}-1$
Unsigned double integer		UDINT	32	0	$0...2^{32}-1$

Elementary data types

Type	ANY-Type	Key word	Data width (Bit)	Initial	Value range
Slide point	ANY_Real	REAL	32	0.0	$-1.18*10^{-38}$.. $3.4*10^{38}$
Long slide point		LREAL	64	0.0	$-2.22*10^{-308}$.. $1.798*10^{308}$
Date	ANY_Date	DATE (D)	32	D#1970-01-01	
Time of day		TIME_OF_DAY (TOD)	32	TOD#00:00	TOD#00:00.. TOD#23:59
Date time of day		DATE_AND_TIME (DT)	32	DT#1970-01-01-00:00	
time	ANY_Time	TIME	32	T#0ms	
Sequential characters	ANY_String	STRING	(80+1)*8	,‘	

Constants

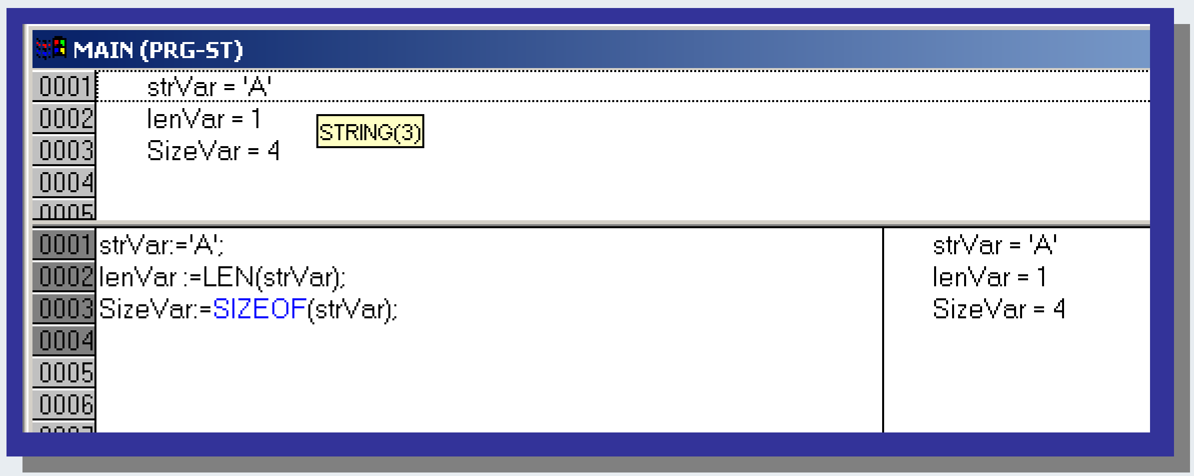
Variablentyp	Beispiele		
BOOL	0,1	16#0, 16#1	FALSE,TRUE
BYTE, WORD, DWORD	10	16#0A	2#1010
DWORD, UINT	32768	16#8000	2#1000_0000
INT	-10	---	---
TIME	T#1h2m4s11ms,	T#62m4s11ms,	T#3724011ms
REAL, LREAL	0.22, 2.2e-1 1000, 1000.0, 1e3 (		
STRING	“ empty string, ‘constant‘ ‘Text$0D$0A‘, ‘Text$R$N‘ Special characters		

String

A STRING type variable can contain any string of characters. The size entry in the declaration determines how much memory space should be reserved for the variable. It refers to the number of characters in the string and can be placed in parentheses or square brackets. If no size specification is given, the default size of 80 characters will be used.

```
VAR
    strVar :STRING(3);
    lenVar: INT;
    sizeVar: INT;
END_VAR
```

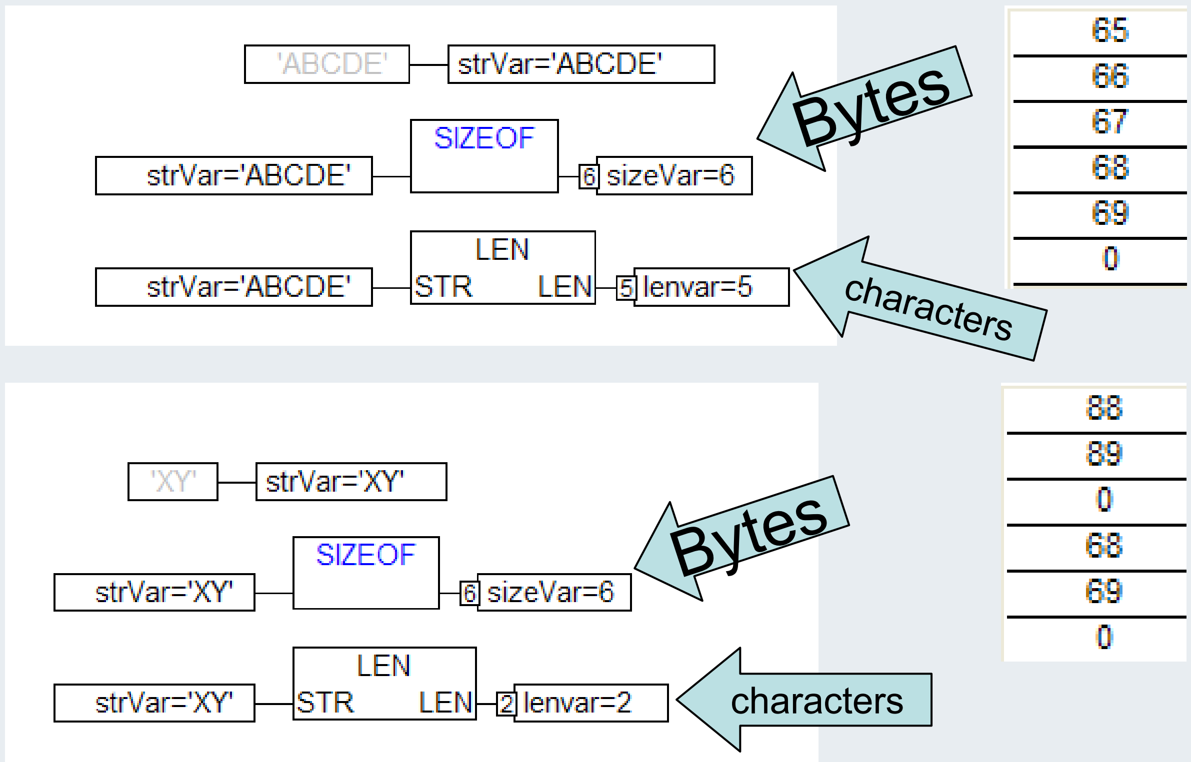
Strings are zero terminated, that means the last character of a string is always zero. Each character inside a string needs one byte.



String

Nullterminierung, LEN und SIZEOF

SPS memory




```
VAR
    strVar :STRING(5);
    lenVar: INT;
    sizeVar: INT;
END_VAR
```

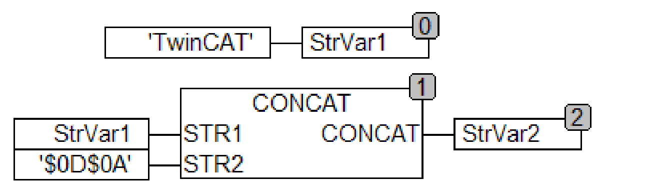
Special characters

If you want to add a special character into a string, you have to begin with a $-character.

Special Characters

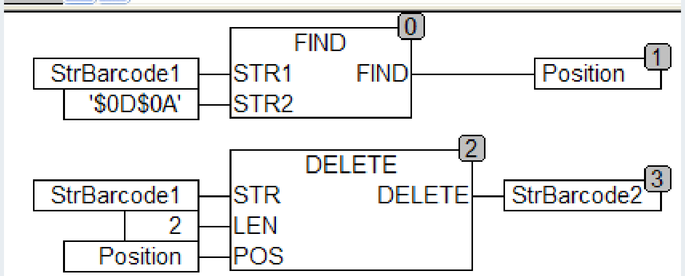
character	description
$$	dollar signs
$‘	Single quotation mark
$L or $l	Line feed
$N or $n	New line
$P or $p	Page feed
$R or $r	Line break
$T or $t	Tab

```
PROGRAM MAIN
VAR
    StrVar1 :STRING;
    StrVar2 :STRING;

END_VAR
```



StrVar1 = 'TwinCAT'
StrVar2 = 'TwinCATRN'

```
PROGRAM MAIN
VAR
    StrBarcode1 :STRING:='12368432$0D$0A';
    StrBarcode2 :STRING;
    Position: INT;
END_VAR
```

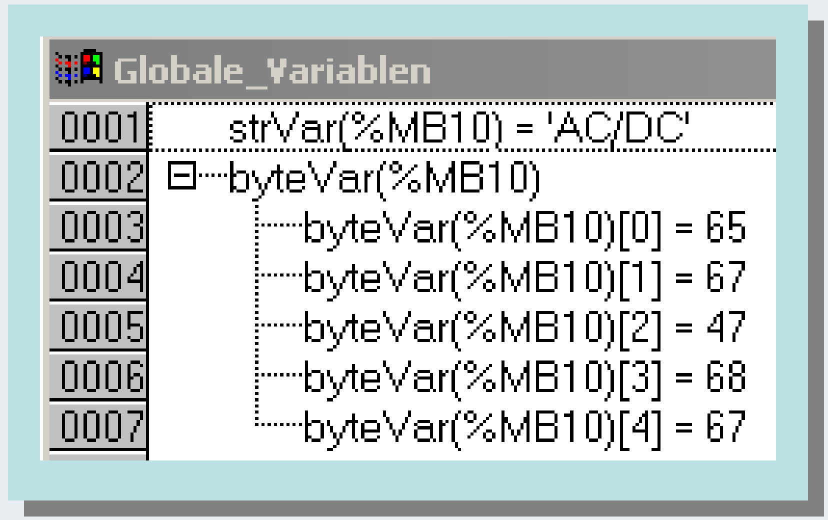


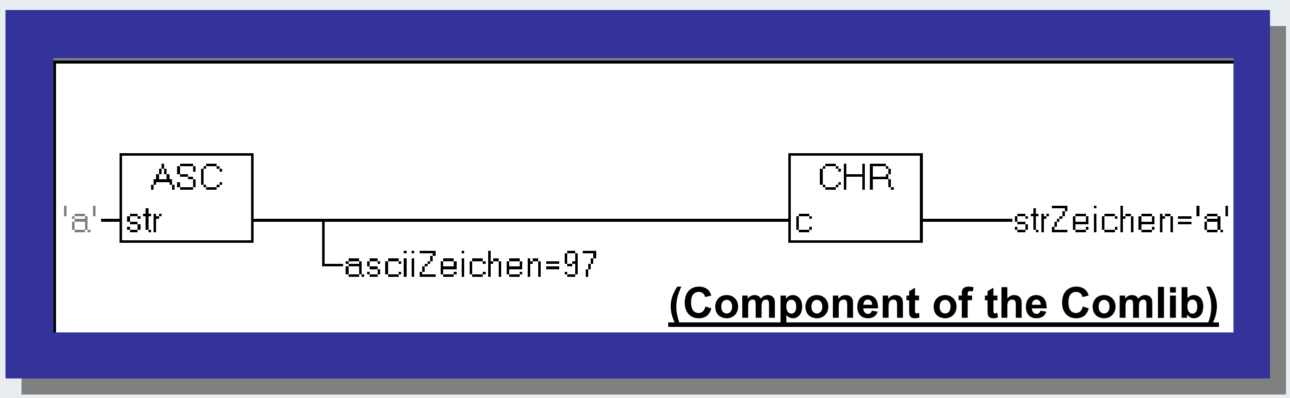
StrBarcode1 = '12368432RN'
StrBarcode2 = '12368432'
Position = 9

ASCII <-> CHR

If a character in a program ought to be converted to an ASCII character, two procedures are allowed:

1. Indirectly, by interpreting the data memory different.
2. Directly via the provided function block. ASC and CHR are both included in the library ChrAsc.lib.





(Component of the Comlib)

Variables declaration el. data types

A variable owns a name, behind which a value (number, string, date) hides. The name of the variable is a way description to the declared data. Variables distinguish themselves thereby, that their content can be changed to the run time.

Identifier Data type Initial value

```
bStellerUntenLinks:BOOL:=TRUE;
```

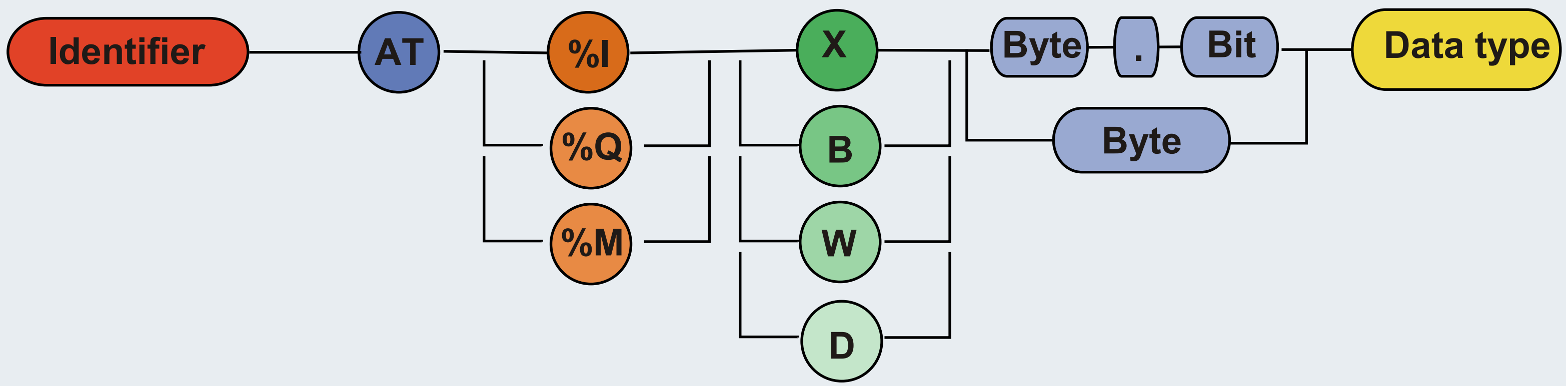
The physical logical storage location of the variable is not known by the operator (unlocated)

The degrees of freedom and the restrictions at the assignment of the identifiers can be seen on the slides identifier and prefixes.

Variables declaration el. data types

At the declaration of the variables it´s possible to link the name with an explicit specified address. For the mapping of the inputs and outputs to the symbolic variables, the locating of variables is essential.

Identifier **AT** **Address** **:** **Data type** **;**



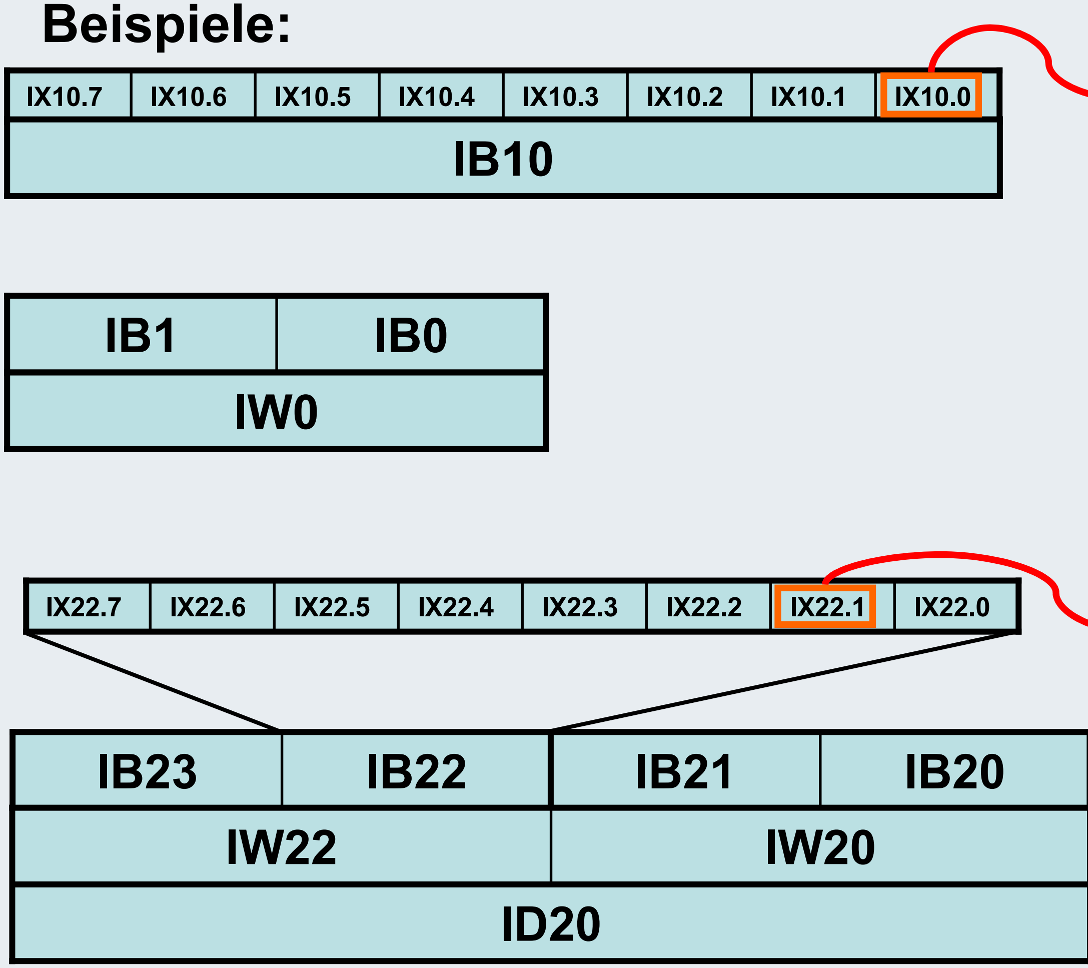
bStellerUntenLinks AT%IX0.0:BOOL:=TRUE;

From TwinCAT 2.8 the addressing can be done automatically. Then the program works with not completely located variables.

bStellerUntenLinks AT%I*:BOOL:=TRUE;

Adresses

Beispiele:



Din0 AT%IX10.0 : BOOL;

Ain AT%IB0 : INT;

Ain AT%IW0 : INT;

BitVar AT%IX22.1 : BOOL;

Posi AT%IB20 : UDINT;

Posi AT%ID20 : UDINT;

Validity range

Local variables are limited on the block, in which they were declared.

Key words

VAR ..

END_VAR
VAR_INPUT ..

END_VAR
VAR_IN_OUT ..

END_VAR
VAR_OUTPUT ..

END_VAR

Global variables are known in each block inside a project.

Key words

VAR_GLOBAL ..

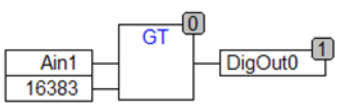
END_VAR
VAR_CONFIG ..

END_VAR

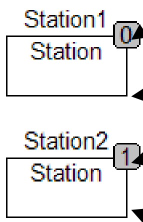
Examole VAR_CONFIG

Loacal variable directly as in and output

```
FUNCTION_BLOCK Station
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    DigIn0 AT%I* :BOOL;
    DigIn1 AT%I* :BOOL;
    DigIn2 AT%I* :BOOL;
    DigIn3 AT%I* :BOOL;

    DigOut0 AT%Q* :BOOL;
    DigOut1 AT%Q* :BOOL;
    DigOut2 AT%Q* :BOOL;
    DigOut3 AT%Q* :BOOL;

    Ain1 AT%I*:INT;
END_VAR
```




```
VAR
    Station1: Station;
    Station2: Station;
END_VAR
```




```
VAR_CONFIG
    MAIN.Station1.DigIn0 AT %IX0.0 : BOOL;
    MAIN.Station1.DigIn1 AT %IX0.1 : BOOL;
    MAIN.Station1.DigIn2 AT %IX0.2 : BOOL;
    MAIN.Station1.DigIn3 AT %IX0.3 : BOOL;
    MAIN.Station1.DigOut0 AT %QX0.0 : BOOL;
    MAIN.Station1.DigOut1 AT %QX0.1 : BOOL;
    MAIN.Station1.DigOut2 AT %QX0.2 : BOOL;
    MAIN.Station1.DigOut3 AT %QX0.3 : BOOL;
    MAIN.Station1.Ain1 AT %IB1 : INT;
    MAIN.Station2.DigIn0 AT %IX3.0 : BOOL;
    MAIN.Station2.DigIn1 AT %IX3.1 : BOOL;
    MAIN.Station2.DigIn2 AT %IX3.2 : BOOL;
    MAIN.Station2.DigIn3 AT %IX3.3 : BOOL;
    MAIN.Station2.DigOut0 AT %QX0.4 : BOOL;
    MAIN.Station2.DigOut1 AT %QX0.5 : BOOL;
    MAIN.Station2.DigOut2 AT %QX0.6 : BOOL;
    MAIN.Station2.DigOut3 AT %QX0.7 : BOOL;
    MAIN.Station2.Ain1 AT %IB4 : INT;
END_VAR
```

Access via the located variables

From program A is a direct access by address %MB2 to the local declared variable ‚locVar' in program B possible.

Project Machine

```
PROGRAM A
VAR
END_VAR

LD   %MB2
.
.
```

```
PROGRAM B
VAR
locVar AT%MB2:WORD;
END_VAR

.
.
.
.
```

Overlapping in the validity range

Project Machine

```
VAR_GLOBAL
    Var1:WORD;
END_VAR
```

```
PROGRAM A

VAR
    Var1 :WORD;
END_VAR
```

```
LD   Var1
.
.
```

As shown in the example on the left, there is an overlapping in the validity range.
In this case, the local declared variable Var1 is loaded into the accumulator.

The compiler generates no warning for this overlapping.

Attribute

Attributes to store variables remanent by shutdown the PLC

```
VAR RETAIN
        Zaehler:UINT;
END_VAR
```

```
VAR PERSISTENT
        Zaehler:UINT;
END_VAR
```

Action in PLC	RETAIN	PERSITENT
PLC RESET (Plc Control, TcSystem)	0,delete	unchanged
RESET ALL (PLC Control)	0, delete	delete
CLEAN ALL (PLC Control)	0, delete	unchanged

Attribute

Initial values:

```
VAR
        AccelerationTime  : TIME := T#3s200ms;
END_VAR
```

Attributes (constants)

Projekt Maschine

```
VAR_GLOBAL CONSTANT
.
END_VAR
```

```
PROGRAM A
VAR CONSTANT
.
END_VAR
.
.
.
```

If you want to use a mathematic, construction, or machine constant, you have to complete the regular key words VAR_GLOBAL .. END_VAR with the key word CONSTANT.
This completement can also be used with local keywords.
The state of these identifier is read.

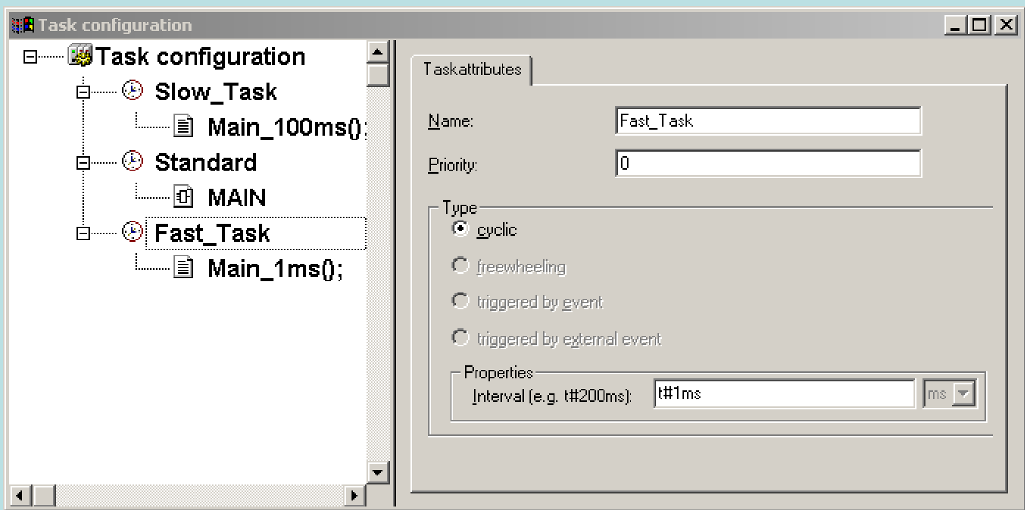
```
VAR_GLOBAL CONSTANT
    pi:REAL:=3.141592654;
END_VAR
```

POU program organisation units

In the IEC61131-3 exists under the main generic term three POUs:

- **Programs**
- **Function blocks**
- **Functions**

The *organisation POU* is replaced by the task configurator.

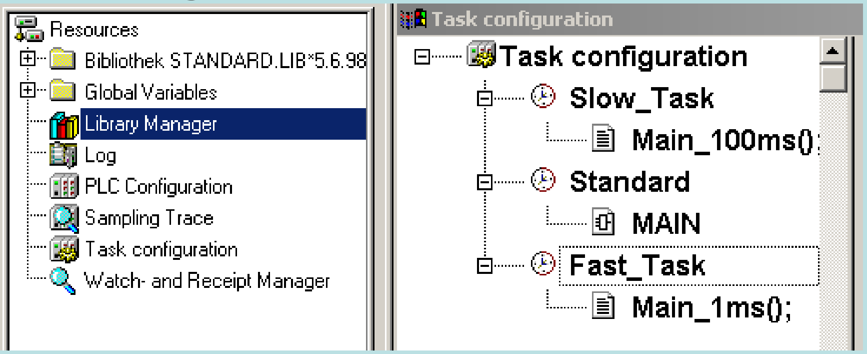


The data POUs are replaced by multi-dimensional fields (ARRAY's).

PROGRAM PRG

Program PRG

- Call by a task (One programm calls another)
- calls : FB's, Functions, (Programs)
- Local variable : static, i.e. the local data are available at the next cycle.
- Inputs: mostly 0, but VAR_INPUT possible
- Outputs: mostlys 0, but VAR_OUTPUT possible
- Transfer by reference: VAR_IN_OUT also possible
- Debug: The local data are directly visible in the online mode of the PLC Control
- Using: Main programms, main, hand, automatic....



Function block FB

Function block FB

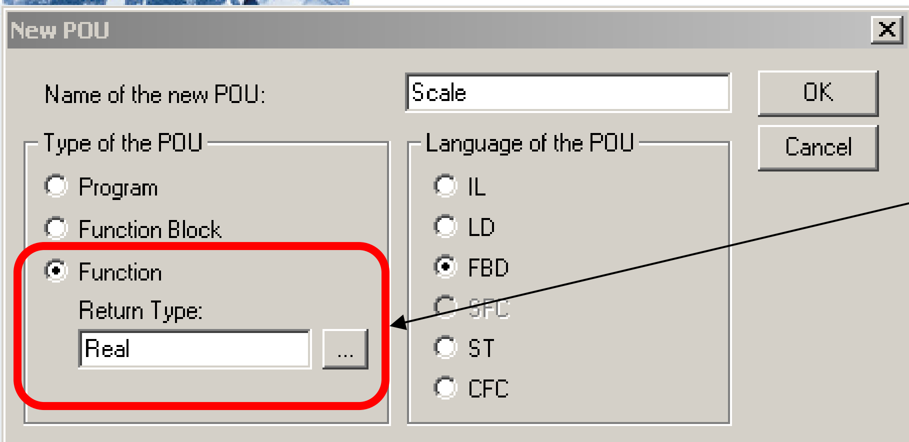
- Called by programs or other FB´s
- calls : FB‘s, functions,
- Locale variable : static, i.e. the local data are again available at the next cycle. At multiple call multiple instances (mulitply).
  Each FB call can have own local data.
- Inputs: 0,1,2,3 VAR_INPUT
- Outputs: 0,1,2,3 VAR_OUTPUT
- Transfer by reference 0,1,2,3 VAR_IN_OUT
- Debug: In the online mode of PLC Control, the instance of the according call has to declared. After this, the local data are visible for each call.
- Using: multiple used function blocks, which need an own data range each. Multiple sequences....

Function FC

Function FC

- called by: <u>programs, function blocks and other functions</u>
- calls: functions
- Local variable : <u>temporary</u>, i.e. the local data are only available for the operating time of the function. Afterwards this data range is used by other functions.
- Inputs: 1,2,3........ VAR_INPUT
- Outputs: exactly 1!, but structure varaibale possible. The output name is at the same time the name of the function.
- Except for TwinCAT: VAR_IN_OUT possible,
- Debug: The local variables are visible with „???“ in the online mode of PLC Control, because these variables are multiple used by all functions in the cycle, and the monitoring (debug) takes place at the cycle bounds. Hepl: program development with breakpoints Breakpoints
- Using: algorithms, at which the result is available after a pass. Scaling, compare......

FC Specials



From TwinCAT 2.8: The return value can be defined directly if a new function is created.

Function name

Inputs

Local variables are only valid for the operating time of the function

```
FUNCTION Scale : REAL
VAR_INPUT
    x:                   REAL;
    xug, xog, yug, yog: REAL;
END_VAR
VAR
    Scalierung, Offset :       REAL;
END_VAR

Scalierung:= (yog-yug)/(xog-xug);

Scale:= Scalierung * x + Offset;
```

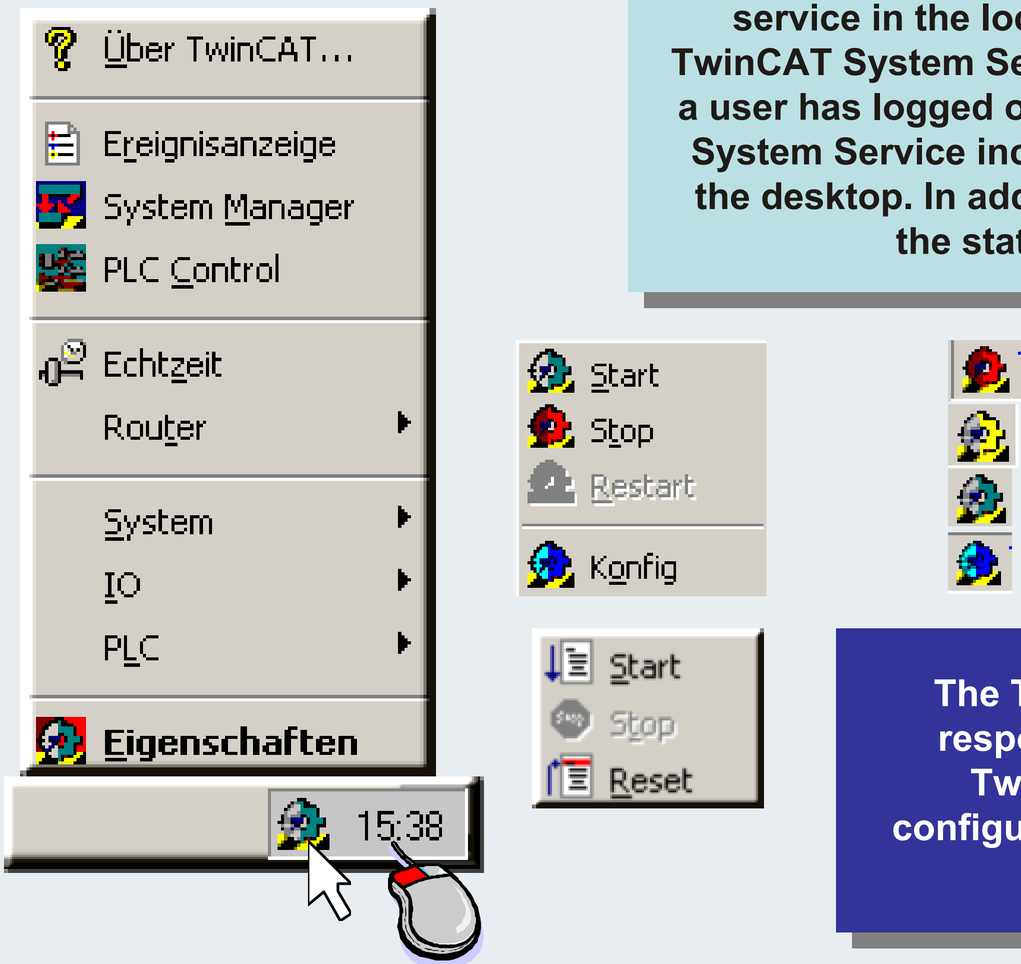
Return value

The name of the „output“ is scale.

Scale can be used as local variable inside the function(Write/Read)

TwinCAT System Service

The TwinCAT System Service operates as Windows NT service in the local system account. In this way, the TwinCAT System Service is started by Windows NT before a user has logged on. As an activity symbol, the TwinCAT System Service incorporates its icon into the task bar of the desktop. In addition, the colour of the icon indicates the state of the TwinCAT system.



TwinCAT stopped

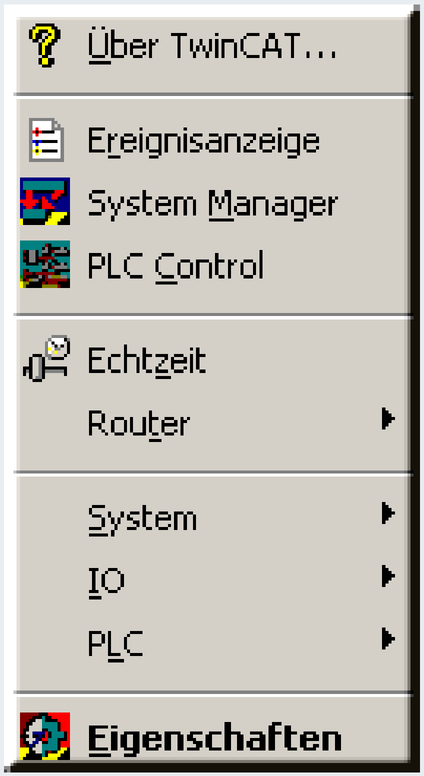
TwinCAT starting.

TwinCAT running.

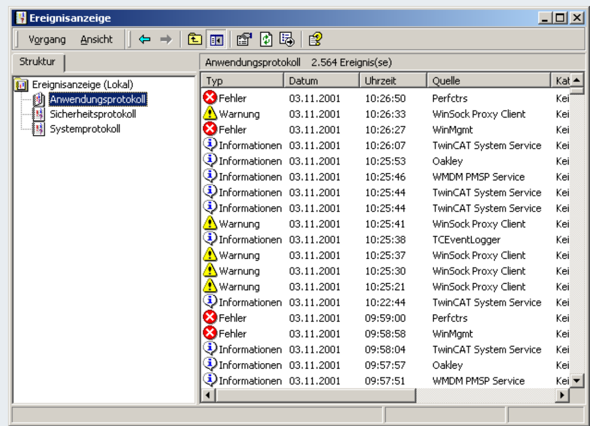
TwinCAT Config Mode

The TwinCAT System Service is primarily responsible for starting and stopping the TwinCAT run time system. It loads all configured servers and initialises them during the TwinCAT system start.

TwinCAT System Service



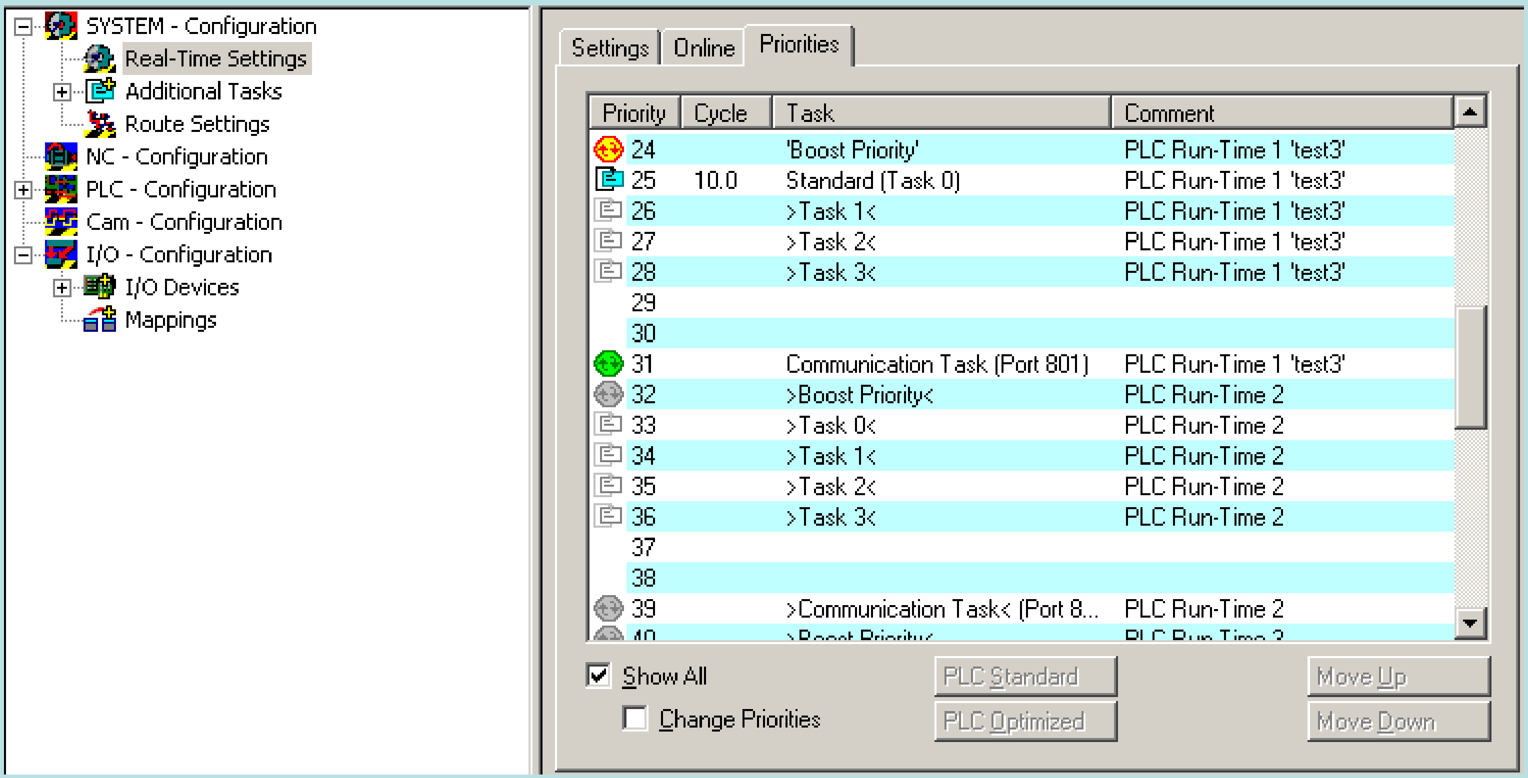
The event display is a programm to moniotor the events in the system. The event logging service starts automatically, if you execute Windows NT.



The TwinCAT I/O subsystem can be reset via the TwinCAT System Service. For this, the corresponding function must be selected in the context menu. The reset applies to all connected field bus systems.

Multitasking

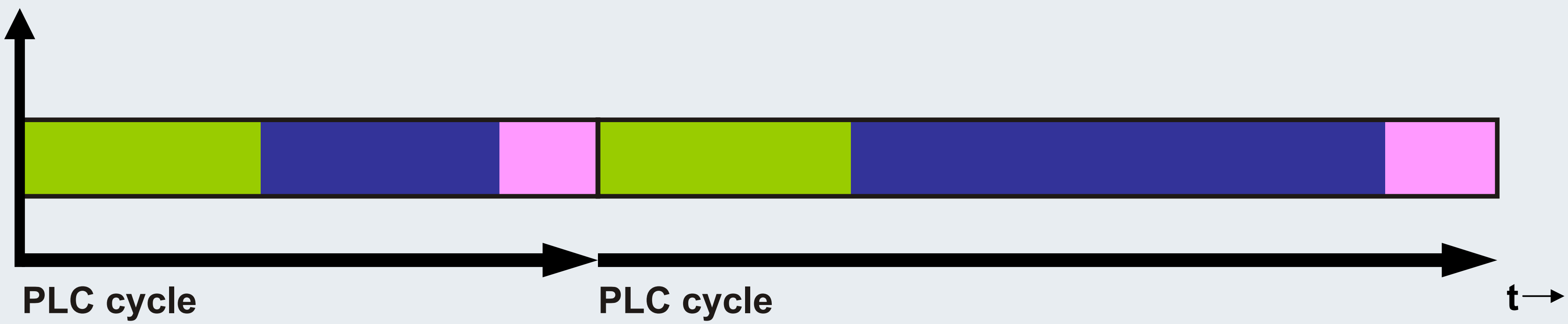
TwinCAT possesses more than 62 different tasks. The default settings can use preset profiles or change the priority individually.



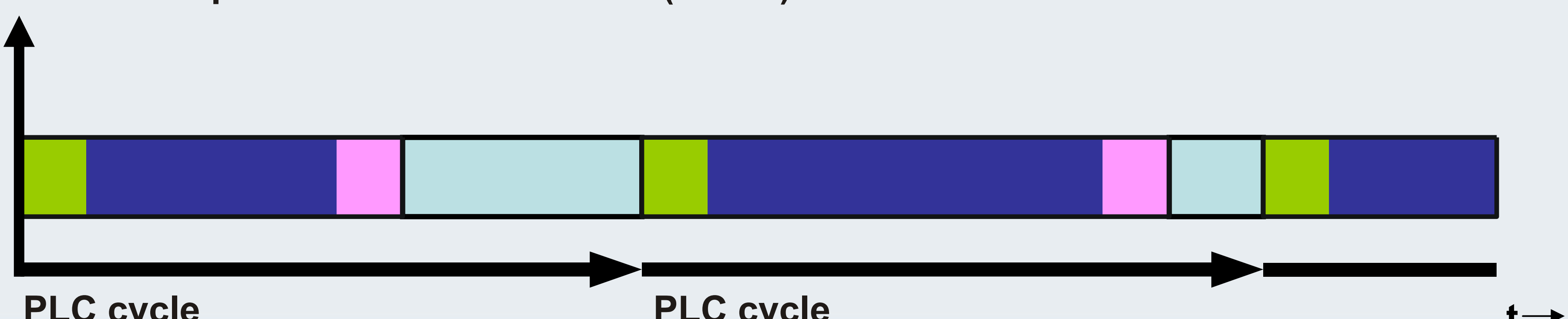
Assigning the computing power

Real time operation of PLC software in the classical PLC.

Read inputs (green)	Operate program (dark blue)
Write outputs (pink)	Win NT & HMI Interface (light blue)



Real time operation of PLC software (1 task) on a PC with windows NT.



Real time

Many industrial applications demand a guarantee, that, clearly predictable and reproduceable, the system load reacts sufficient fast to the process event in a defined time.

The real time is very important for the digital control. The sampling of an analog signal (actual position) with a PC should have absolute constant distances between two measurements.

Each part process requires different reaction times. Because of this, several part processes with different features and different reaction times can be created in one automation task.

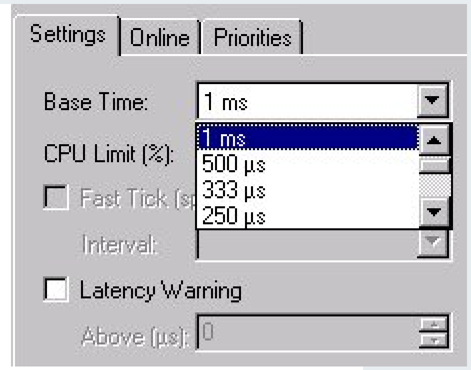
If several tasks want to access the CPU simultaneously, the IEC 61131-3 defines two procedures:

1. Preemptive (interruptible execution) multi tasking (TwinCAT)

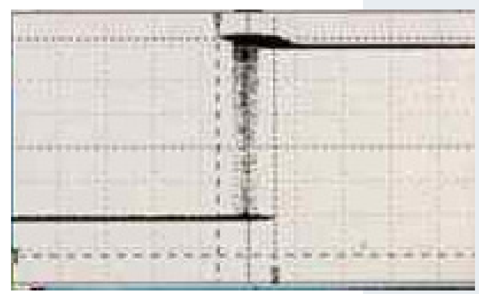
2. Non preemptive (not interruptible execution) multi tasking

Real time operation

▪ The real time operation will be achieved with deterministic time slices. The width of the time slices can be chosen in steps: (1000µs ... 50µs).
The default setting is 1ms.



▪ The time slices will be kept with an accuracy of ± 15µs (Jitter). Device with the lowest priority goes to the waiting loop and waits until the CPU is free.



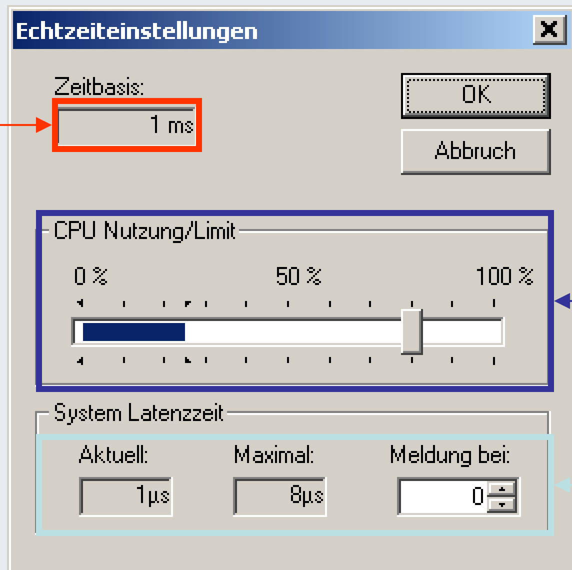
- 15µs +15µs

▪ With the begin of a new time slice, the software devices (PLC, NC) will be executed with priority control.

Real time

The TwinCAT real-time system can be configured via the context menu of the TwinCAT System Service.

Length of the time slice



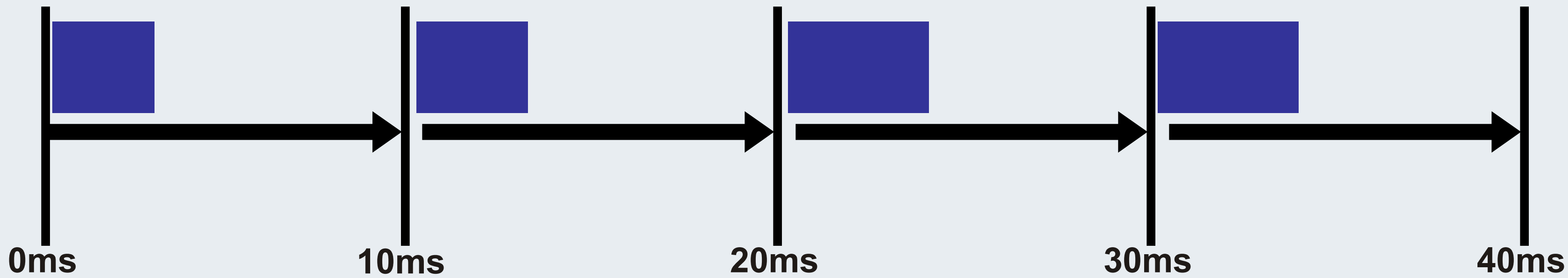
Processor time can be assigned to the TwinCAT real-time system via the linear regulator in the figure above. On a time basis of 1 ms, this means that TwinCAT has a maximum of 800µs available each milli-second.

When the TwinCAT real-time system switches to its idle task, the processor is returned to Windows NT. The bar in the linear regulator dis-plays the current utilisation level of the real-time system. The display is averaged over 256 cycles (ms).

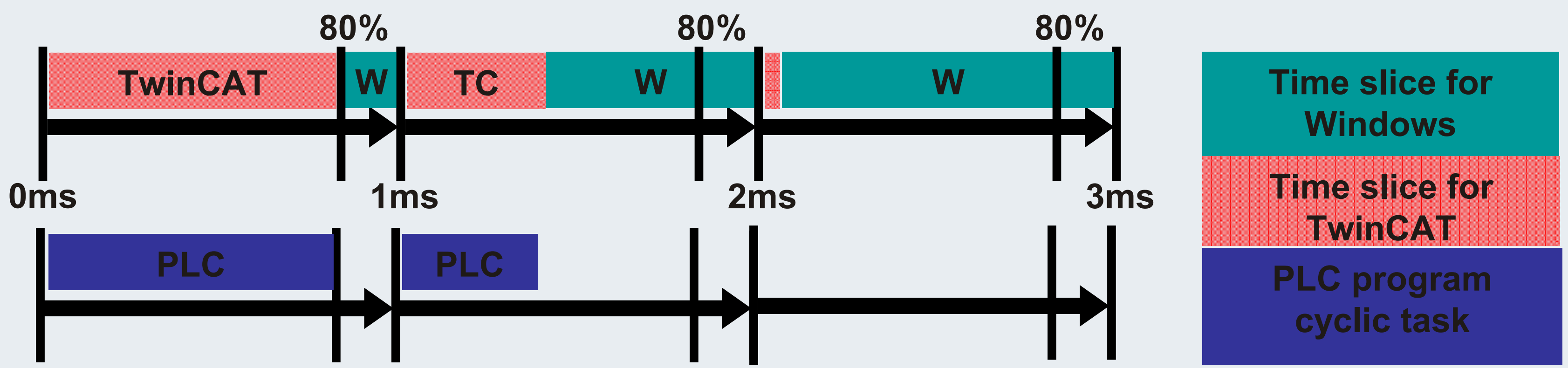
In this case, the current and maximum latency times in the real-time sys-tem are shown. The time by which the central system tick arrives too late is measured.

Real time operation

- Cyclic PLC task e.g. 10ms



- Refinement: Behavior under TwinCAT base cycle 1 ms



If TwinCAT does not need the (full) reserved time slice, the scheduler provides this computing power to windows.

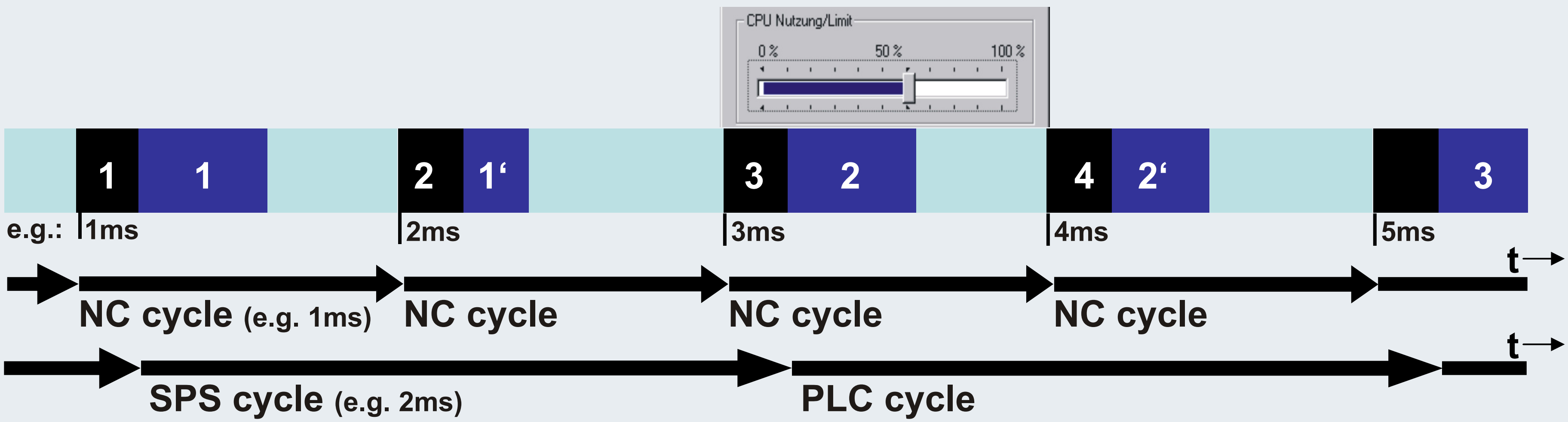
Real time operation

▪ PLC tasks and drive control will be executed <u>deterministically</u> with multiple tasking.

Real time operation of a PLC program and NC control with a PC

NC — PLC program

Win NT & HMI Interface

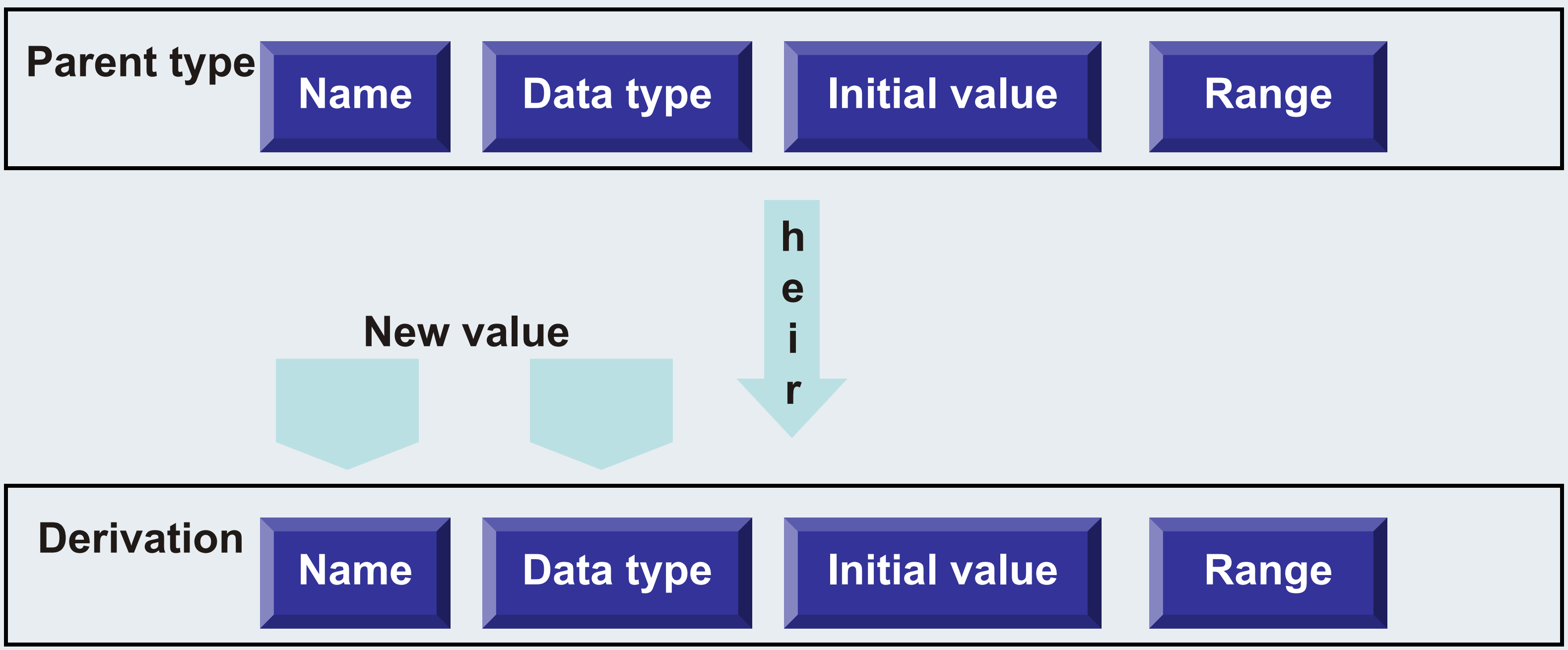


Real time operation

- The smaller the time slice, the shorter the reaction time of the highest priority task.
- This has the consequence that the software devices must be fairly often interrupted.
- If a device is interrupted, the program stack has to be safed. This has the consequence that the recopy expense rises.
- TwinCAT and the operating system are equal.
- For the operating system, calculating capacity is given regularly.
- The switch to the operating system takes place at the earliest, as soon as all TwinCAT devices complete the processing, and at the latest at the CPU limit.

Derivated data types

The user can create own data types on the base of elementary data types or already created data types. The new created data types are visible in the whole project.
They begin with the keyword TYPE and end with END_TYPE.



References (Alias Types)

You can use the user-defined reference data type to create an alternative name for a variable, constant or function block. Create your references as objects in the Object Organizer under the register card Data types.

They begin with the keyword TYPE and end with END_TYPE.

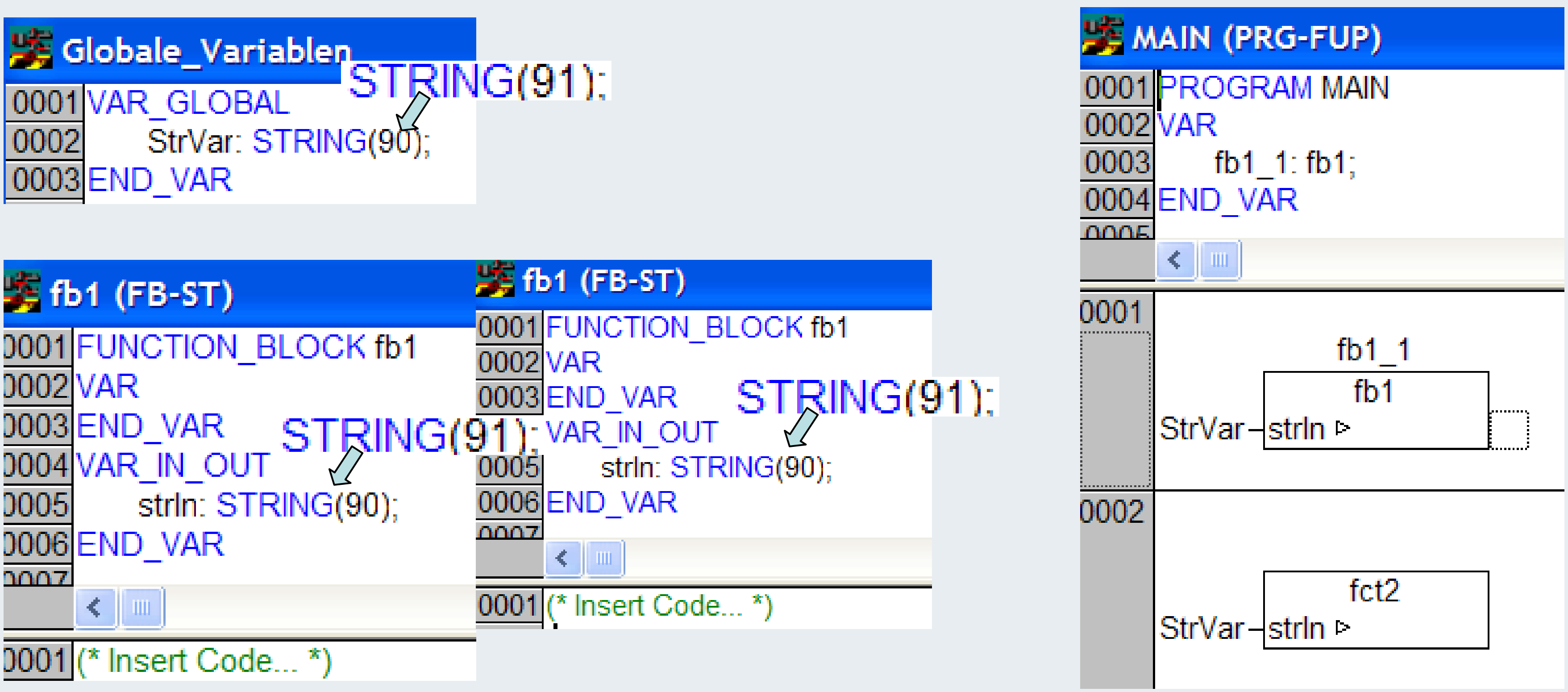
Syntax:

```
TYPE
 <Identifier>:<Assignment term>;
END_TYPE
```

Example: Ads_Net_ID

```
TYPE
    Net_ID:STRING(23);
END_TYPE
```

references (alias-types)



Enumeration

Enumeration is a user-defined data type that is made up of a number of string constants. These constants are referred to as enumeration values. Enumeration values are recognized in all areas of the project even if they were locally declared within aPOU. It is best to create your enumerations as objects in the Object Organizer under the register card Data types. They begin with the keyword TYPE and end with END_TYPE.

Syntax:

TYPE <Bezeichner>:(<Enum_0> ,<Enum_1>, ...,<Enum_n>);
END_TYPE

Beispiel:

TYPE Woche:(Mo, Di, Mi, Dn, Fr, Sa, So:=10);(*Mo = 0 Di = 1..
.. Sa = 6 So = 10*)

END_TYPE

TYPE Richtung:(Up, Dn);(*Up = 0 Dn = 1*)
END_TYPE

You may not use the same enumeration value more than once.

Enumeration

The <Identifier> can take on one of the enumeration values and will be initialized with the first one. These values are compatible with whole numbers which means that you can perform operations with them just as you would with INT. You can assign a number x to the <Identifier>. If the enumeration values are not initialized, counting will begin with 0. When initializing, make certain the initial values are increasing. The validity of the number will be reviewed at the time it is run.

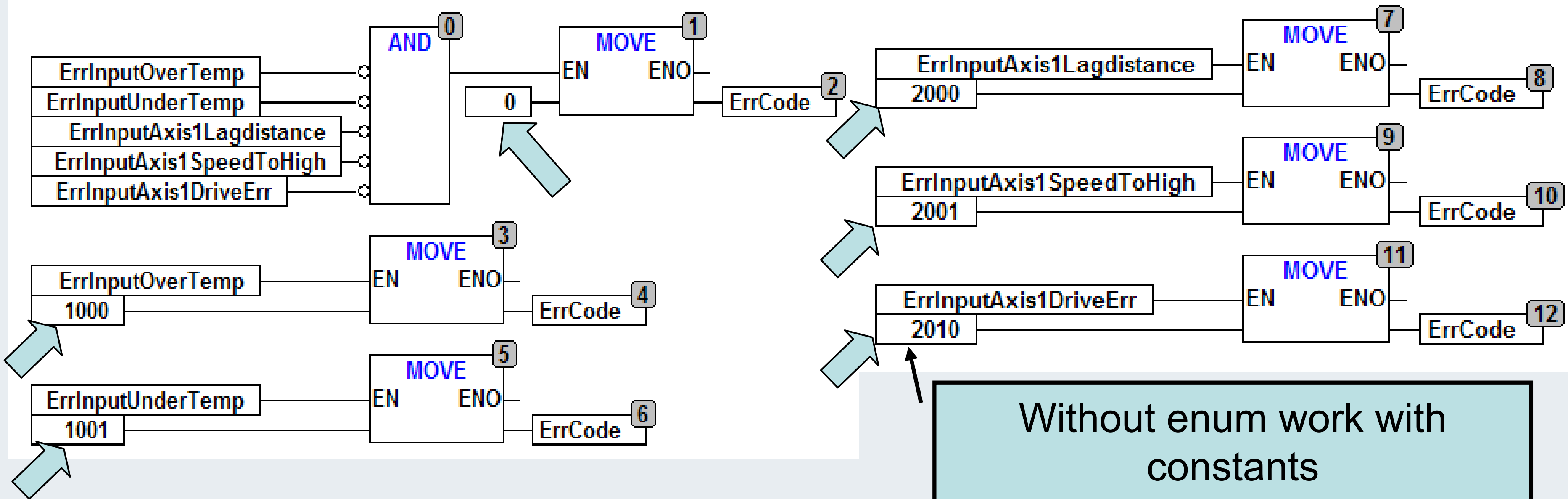
```
VAR
   WochenTag:Woche;
END_VAR
```

```
WochenTag:=3;
```

```
WochenTag = Dn
(Mo:=0,Di:=1,Mi:=2,Dn:=3,Fr:=4,Sa:=5,So:=10)
```

Enumeration type (Enum)

```
VAR
    ErrCode : UINT;(* 0: ERRCODE_NO_ERR *)
                   (* 1000: ERRCODE_OVERTEMP *)
                   (* 1001: ERRCODE_UNDERTEMP *)

                   (* 2000: ERRORCODE_AXIS1_LAGDISTANCE *)
                   (* 2001: ERRORCODE_AXIS1_SPEEDTOHIGH *)
                   (* 2010: ERRORCODE_AXIS1_DRIVEERR *)
    ErrInputOverTemp :BOOL;
    ErrInputUnderTemp :BOOL;
    ErrInputAxis1Lagdistance :BOOL;
    ErrInputAxis1SpeedToHigh :BOOL;
    ErrInputAxis1DriveErr :BOOL;
END_VAR
```



Without enum work with constants

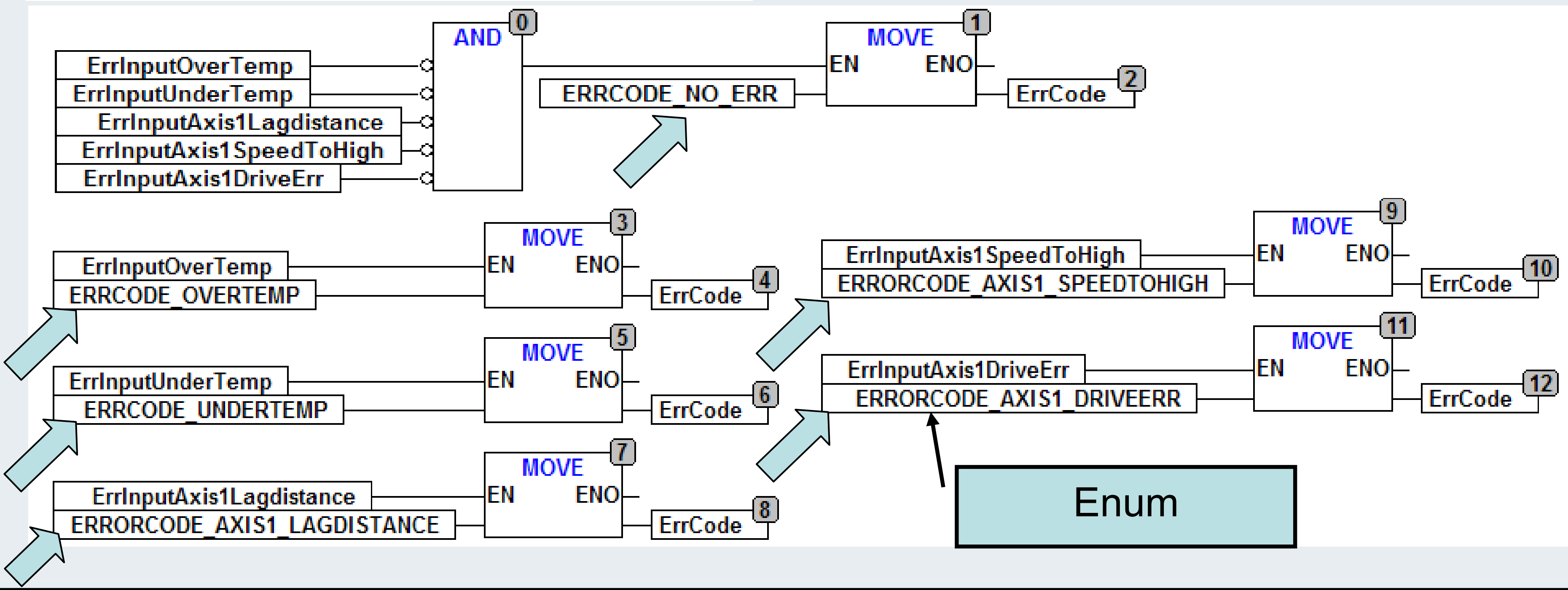
Enumertation type (Enum)

Beispiel mit ENUM

Datentypen
e_Errcode (ENUM)

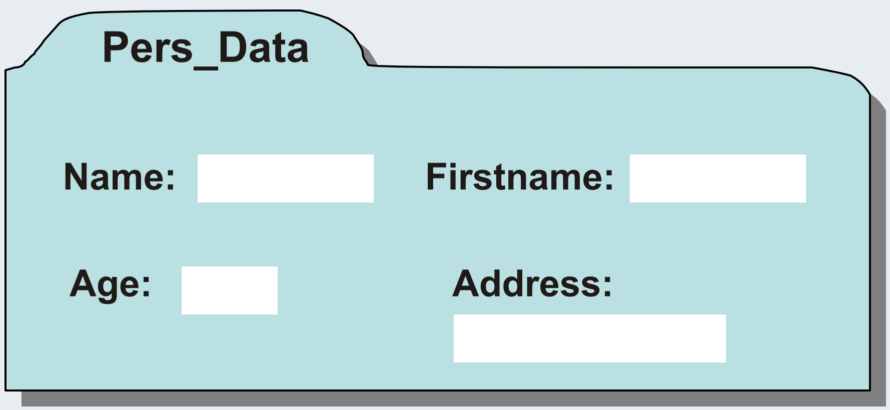
```
TYPE e_Errcode :
    (
    ERRCODE_NO_ERR,                        (* 0 *)
    ERRCODE_OVERTEMP:=1000,                (* 1000 *)
    ERRCODE_UNDERTEMP,                     (* 1001 *)
    ERRORCODE_AXIS1_LAGDISTANCE:=2000,     (* 2000 *)
    ERRORCODE_AXIS1_SPEEDTOHIGH,           (* 2001 *)
    ERRORCODE_AXIS1_DRIVEERR:=2010         (* 2010 *)
    );
END_TYPE
```

```
VAR
    ErrCode :  e_Errcode ;
    ErrInputOverTemp :BOOL;
    ErrInputUnderTemp :BOOL;
    ErrInputAxis1Lagdistance :BOOL;
    ErrInputAxis1SpeedToHigh :BOOL;
    ErrInputAxis1DriveErr :BOOL;
END_VAR
```



Structure declaration

form



Structures are self defined data types. They are important aids for managing the process data.

Furthermore the structures are suited for capsulated data transfer to function blocks.

Structures can be used like single element variables.

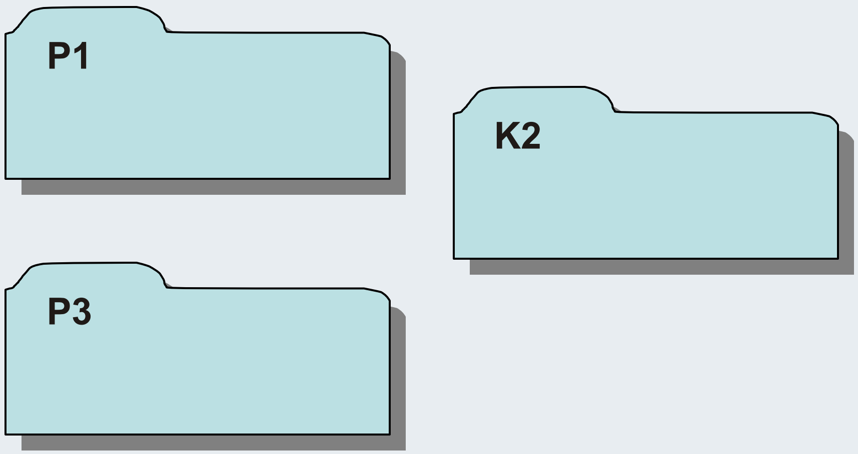
```
TYPE Pers_Data :
STRUCT
    Name: STRING(25);
    Firstname: STRING(25);
    Age:USINT;
    Address: STRING(55);
END_STRUCT
END_TYPE
```

Identifier for the new data type

Identifier : parents data type

-
-
-

Structures Instances



P1
Name:=‚Müller'
Firstname:=‚Peter'
Age:=32
Address:=‚Postweg 34'

P3
Name:=‚Koschnik'
Firstname:=‚Heinz'
Age:=37
Address:=‚Domplatz 10'

```
VAR
      P1, P3 : Pers_Data;
END_VAR
VAR_OUTPUT
      K2 : Pers_Data;
END_VAR
.
.
```

```
VAR_INPUT
      Employees: Pers_Data;
END_VAR

Name_total:=CONCAT(P3.Firstname, P3.Name)
                  (*Heinz Koschnik*)
```

Arrays

Arrays describe lists resp. data arrays. All elements in the arrays are from the same type. Arrays can also exist of own data types (structures).

One- , two-, and three-dimensional arrays are possible.

```
VAR
    Feld_1 :ARRAY[1..10] OF BYTE;           one-dimensional
    Feld_2 :ARRAY[1..10, 2..5] OF UINT;     two-dimensional
    Feld_3 :ARRAY[1..10] OF DINT;           three-dimensional
END_VAR
```

- It´s possible to put a data array to a direct addressed memory position

```
VAR
    Feld_1 AT%MB100:ARRAY[1..10] OF BYTE;
END_VAR
```

- Access to the sub-elements of a data array

```
Feld_1[2] := 120;  (* explicit access*)
Feld_2[i,j] := EXPT(i,j); (*indicated access*)
```

Array one-dimensional

One-dimensional

Identifier Field Data type Initial value

DBZeiten :ARRAY [0..6] OF TIME:= T#1s, T#2s, T#1s, 4(T#0s);

Faktor Wert

0	1	2	3	4	5	6
T#1s	T#2s	T#1s	T#0s	T#0s	T#0s	T#0s

The field length can be done explicit or with the aid of constants.

A dynamic change of the field size is not possible.

Access:

```
VAR
    WertAusArray : TIME;
END_VAR
```

WertAusArray := DBZeiten[1];

Array two-dimensional

To assign for example support points, an array is well qualified.

Identifier Field Data type Initial value

Supportpoint:ARRAY [0..1, 0..6] OF REAL:= 0, 1.7, 2, 4(3.33), 6, 6(1.2);

Factor Value

	0	1	2	3	4	5	6
0	0	1.7	2	3.33	3.33	3.33	3.33
1	6	1.2	1.2	1.2	1.2	1.2	1.2

Access:

```
VAR
    WertAusArray : REAL;
END_VAR
WertAusArray := Supportpoint[1 ,0];
```

Array initialisation more clearly with comments

Example: Drive jobs for an axis

```
Drivejob:ARRAY [0..3, 0..1] OF LREAL:=
                (* target position,      velocity   *)
(*Job 0*)       20.0,                    30.0,
(*Job 1*)       33.75,                   30.0,
(*Job 2*)       45.0,                    30.0,
(*Job 3*)       70.75,                   30.0;
```

Array three-dimensional

Stuetzpunkte:ARRAY [0..1, 0..2, 0..3] OF UINT:=
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23;

[1,0,0] =12	[1,0,1] =13	[1,0,2] =14	[1,0,3] =15
[1,1,0] =16	[1,1,1] =17	[1,1,2] =18	[1,1,3] =19
[1,2,0] =20	[1,2,1] =21	[1,2,2] =22	[1,2,3] =23

[0,0,0] =0	[0,0,1] =1	[0,0,2] =2	[0,0,3] =3
[0,1,0] =4	[0,1,1] =5	[0,1,2] =6	[0,1,3] =7
[0,2,0] =8	[0,2,1] =9	[0,2,2] =10	[0,2,3] =11

Zugriff:

```
VAR
    WertAusArray : UINT;

END_VAR
WertAusArray :=
Stuetzpunkte[ 1,1,3 ];
```

ARRAY [**0..1**, 0..2, 0..3]

ARRAY [0..1, **0..2**, 0..3]

ARRAY [0..1, 0..2, **0..3**]

Exceed bounds

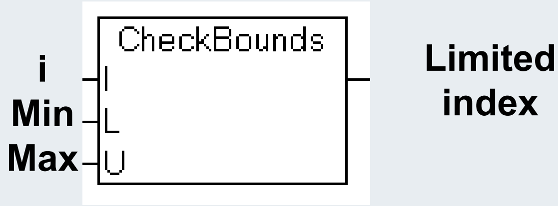
A dangerous state can arise in the PLC program, if an access to a range outside the data field takes place.

```
VAR
  Feld_1 :ARRAY[1..10] OF BYTE;
  Feld_2 :ARRAY[1..10, 2..5] OF UINT;
  Feld_3 :ARRAY[1..10] OF DINT;
END_VAR
```

i:= 9	9
Feld_1[i+2] := 120;	
.	
.	
Feld_1[9];	0
Feld_2[1,2];	120

CheckBounds (FUN)

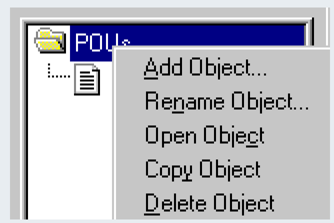
Checkbound works in the runtime (PLC)

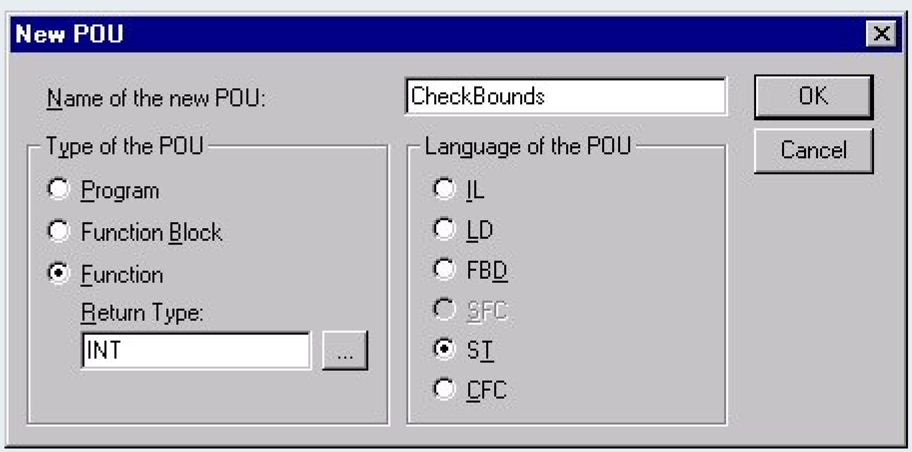



```
FUNCTION CheckBounds :DINT
VAR_INPUT
        I,L,U : DINT;
END_VAR

IF I< L THEN
Fehler-Fall     CheckBounds := L;
 ELSIF I > U THEN
Fehler-Fall     CheckBounds := U;
 ELSE
„OK"-Fall       CheckBounds := I;
END_IF
```

Inserting Check Bounds

CheckBounds can be copied with „Copy Project“ from another PLC project to the current project (e.g. training project). Checkbounds can also be created or written directley.





Inserting Check Bounds

So that CheckBounds is recognised by translating, the following may NOT be changed:

-Name and type of the inputs I,L and U

-Name (CheckBounds) and return value (DINT).

-In the function can be edited freely. At application of own local variables (e. g. error counter, instances of FBs) is to be considered that these are only temporary (at functions). Such a variable has to be declared (in this case) under the global variables.

CheckBounds (FUN-ST)

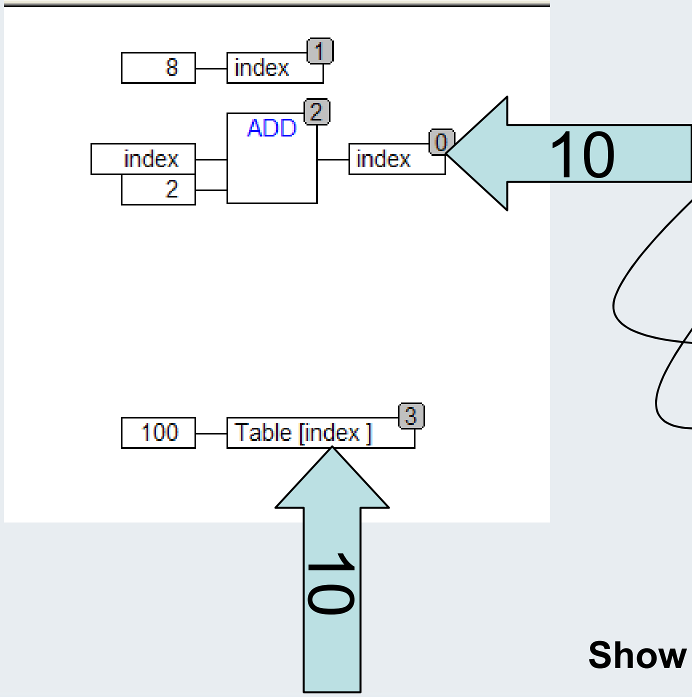
```
FUNCTION CheckBounds : DINT
(* check the array boundaries of all arrays in the project automatically *)
VAR_INPUT
	I,L,U : DINT;	(* dont change this interface ! *)
END_VAR
```

```
(* you can add/modify the code (i.e. write to logfile, set flag *)
IF I< L THEN
	CheckBounds := L; (* returns lower bound L, if index I is lower than lower bound L *)
ELSIF I > U THEN
	CheckBounds := U;	(* returns upper bound U, if index I is greater than upper bound U *)
ELSE
	CheckBounds := I;  (* returns index I, if index I is in the bounds *)
END_IF
```

CheckBounds

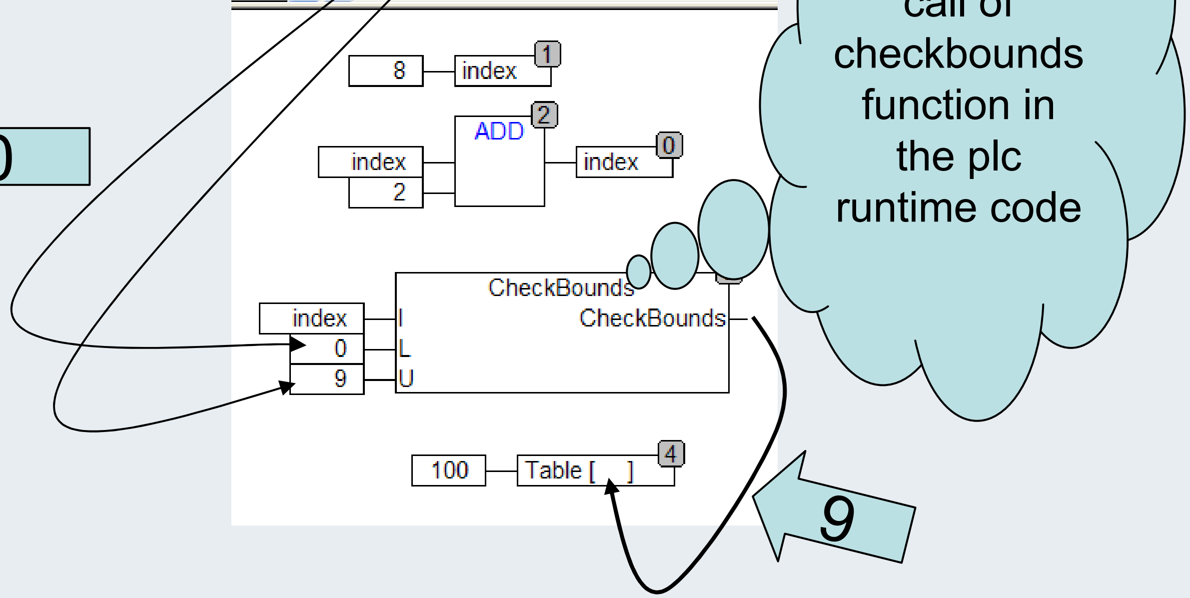
Sourcecode with programming error

```
PROGRAM MAIN
VAR
    Table:ARRAY[0..9] OF INT;
    index :INT;
END_VAR
```

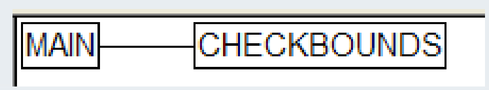


Rebuild „compiles“ checkbounds into the runtime code code

```
PROGRAM MAIN
VAR
    Table:ARRAY[0..9] OF INT;
    index :INT;
END_VAR
```



Show calltree:



Combination Structures and Arrays

An array can consist of structures:

Structure:

```
TYPE DrillPos :
STRUCT
          XPos:            LREAL;
          FeedrateX:       LREAL;
          AccelerationX:   LREAL;
          Decceleration X: LREAL;
          JerkX:           LREAL;
          YPos:            LREAL;
          FeedrateY:       LREAL;
          AcceleartionY:   LREAL;
          DeccelerationY:  LREAL;
          JerkY:           LREAL;
          FeedDrill:       LREAL;
          Kuehlen:         BOOL;   (*Pump ?*)
END_STRUCT
END_TYPE
```

Declaration of the arrays :

```
Positions :ARRAY[0..100] OF DrillPos;
```

Combination Structures and Arrays

- **Access to „Drillpos 55“:**
- **Access:**

```
MoveXAx (*FB Instance*)
        (
        Execute:=           TRUE,
        Position:=          Positions[55].XPos ,
        Velocity:=          Positions[55].FeedrateX
        Acceleration:=      Positions[55].AccelerationX,
        Deceleration:=      Positions[55].DecelerationX,
        Jerk:=              Positions[55].JerkX,
        Direction:=                 .........,
        Axis:=                      .............,
        );
```

ST Structured Text

Operation	Symbol	Binding strength
Put in parentheses	(expression)	Strongest binding
Function call	Function name (parameter list)	
Exponentiation	EXPT	
Negate	-	
Build. complements	NOT	
Multiply	*	
Divide	/	
Modulo	MOD	
Add	+	
Substract	-	
Compare	<,>,<=,>=	
Equal to	=	
Not Equal to	<>	
Bool AND	AND	
Bool XOR	XOR	Weakest binding
Bool OR	OR	

ST Structured text: Overview about Instructions

Instruction	Example
Assignment :=	PosWert := 10;
Callin a function block	Ton1(IN:=Start, PT:=T2s); Output:= Ton1.Q;
RETURN	RETURN;
IF	See the following pages
CASE	
FOR	
WHILE	
REPEAT	
EXIT	
Empty instruction	;

IF Instruction

Is needed to branch in a program depending on conditions.

With the IF instructions it´s not possible to jump back in the PLC cycle.

„GOTO“ is not available

Keywords:

IF THEN

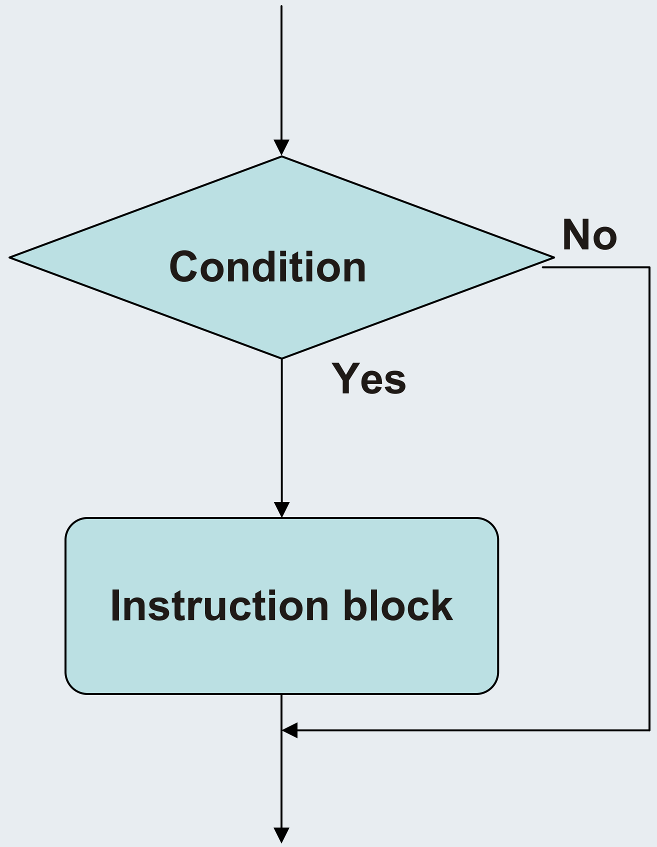
ELSIF

ELSE

END_IF

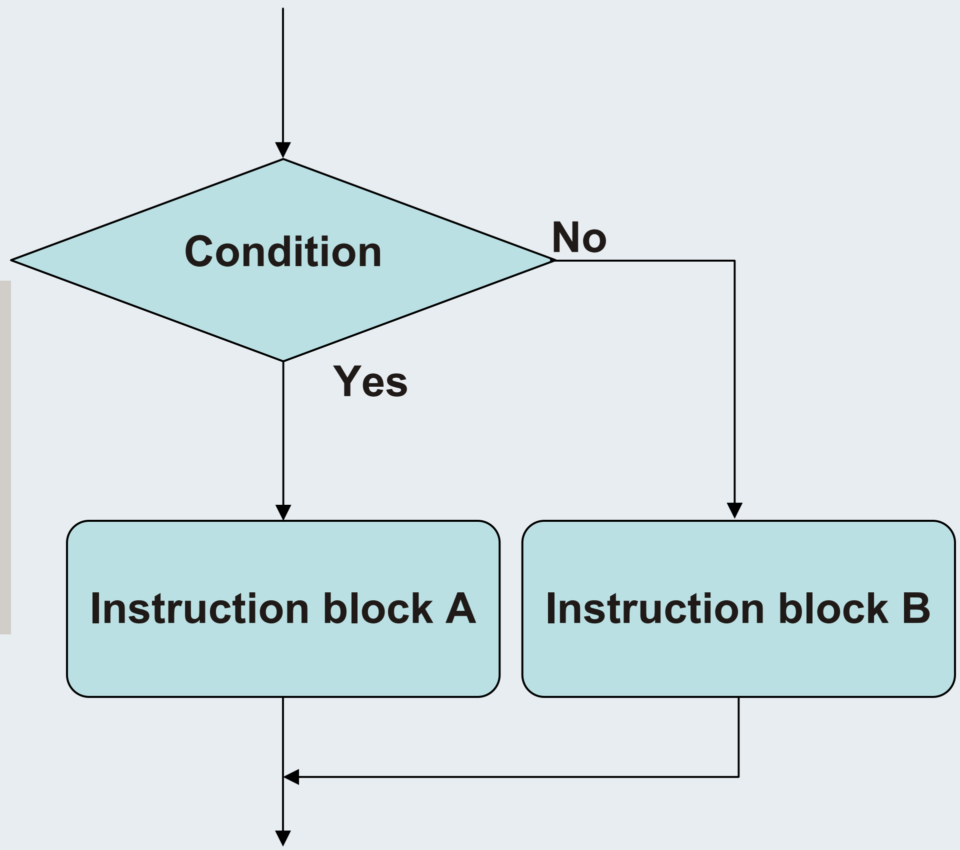
IF Instruction

```
IF Condition THEN
      Instruction block;
END_IF
```

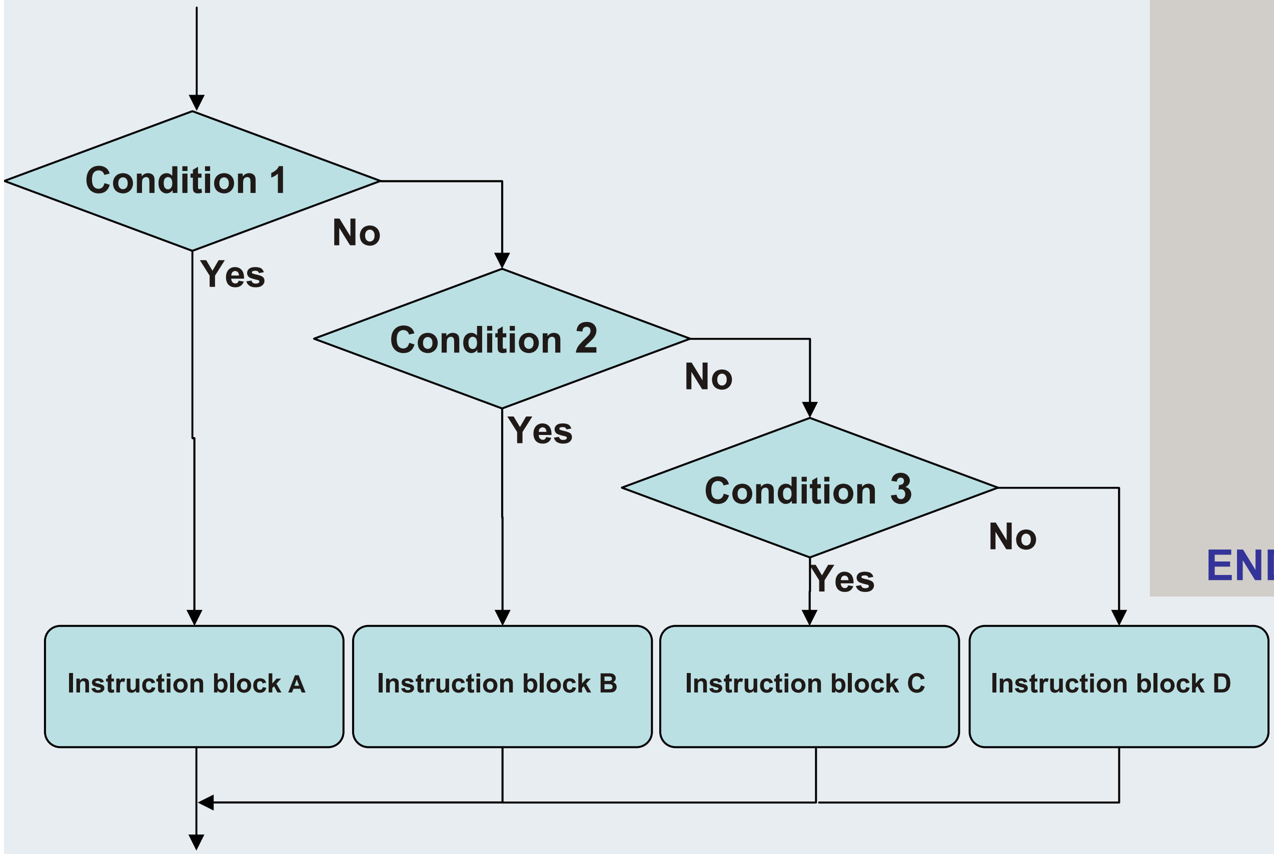


IF Instruction

```
IF a>b THEN
   Instruction block A;
ELSE
   Instruction block B;
END_IF
```

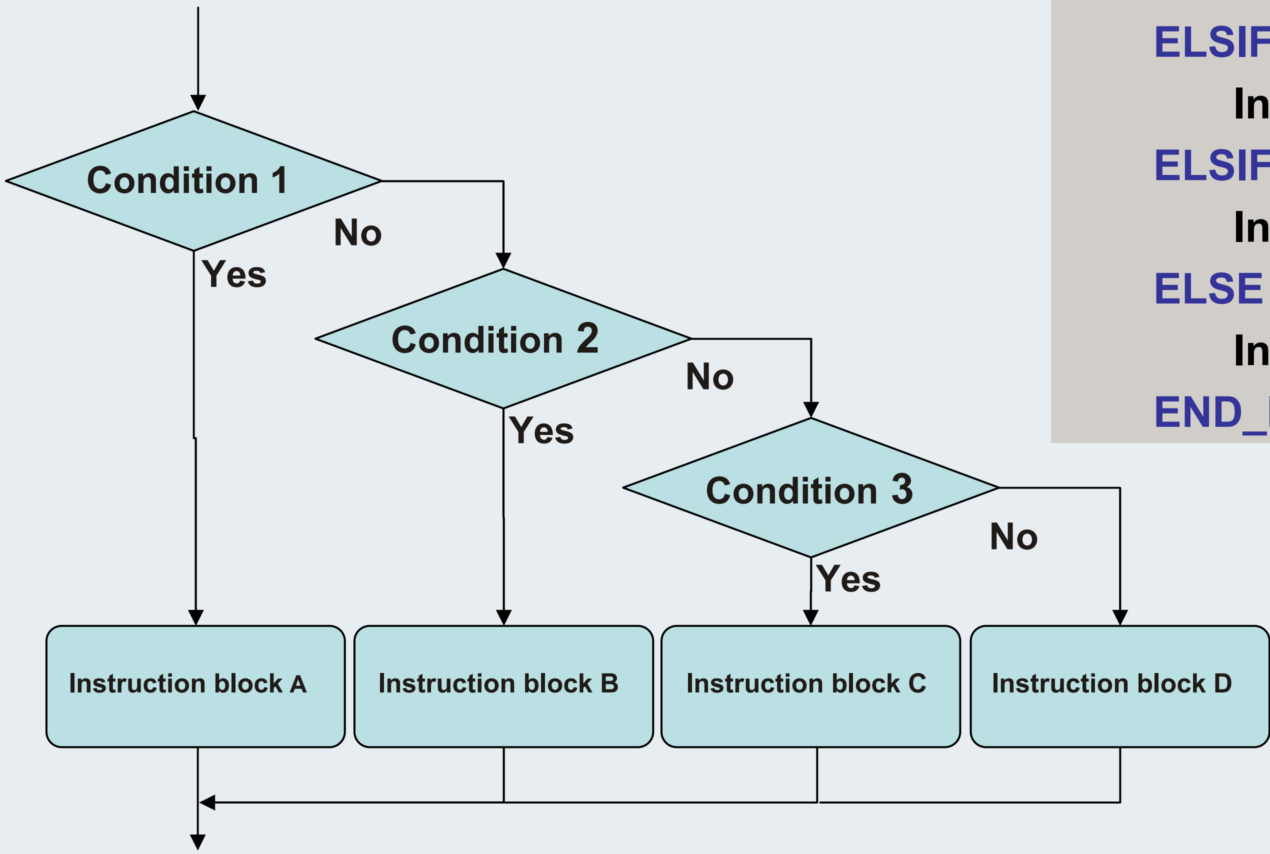


IF Instruction




```
IF Condition1 THEN
    Instruction block A;
ELSE
    IF Condition2 THEN
        Instruction block B;
    ELSE
        IF Condition3 THEN
            Instruction block C;
        ELSE
            Instruction block D;
        END_IF
    END_IF
END_IF
```

IF Instruction




```
IF Condition1 THEN
    Instruction block A;
ELSIF Condition2 THEN
    Instruction block B;
ELSIF Condition3 THEN
    Instruction block C;
ELSE
    Instruction block D;
END_IF
```

IF Instruction

What can the „BOOLEAN EXPRESSION“ be ?

Conditions :

- BOOLEAN Variable
- Comparison
- Function calls
- Call FB Instances
- NO FB call!

IF bVar THEN

.

IF a>b THEN

.

IF   LEFT(STR:= strVar, SIZE:=7) = 'TwinCAT'   THEN

.

IF Ton1.Q THEN

.

IF ~~Ton1(IN:=bVar, PT:=T#1s)~~ THEN

CASE Instruction

CASE Selection criterion **OF**

1: Instruction 1

2,4,6: Instruction 2

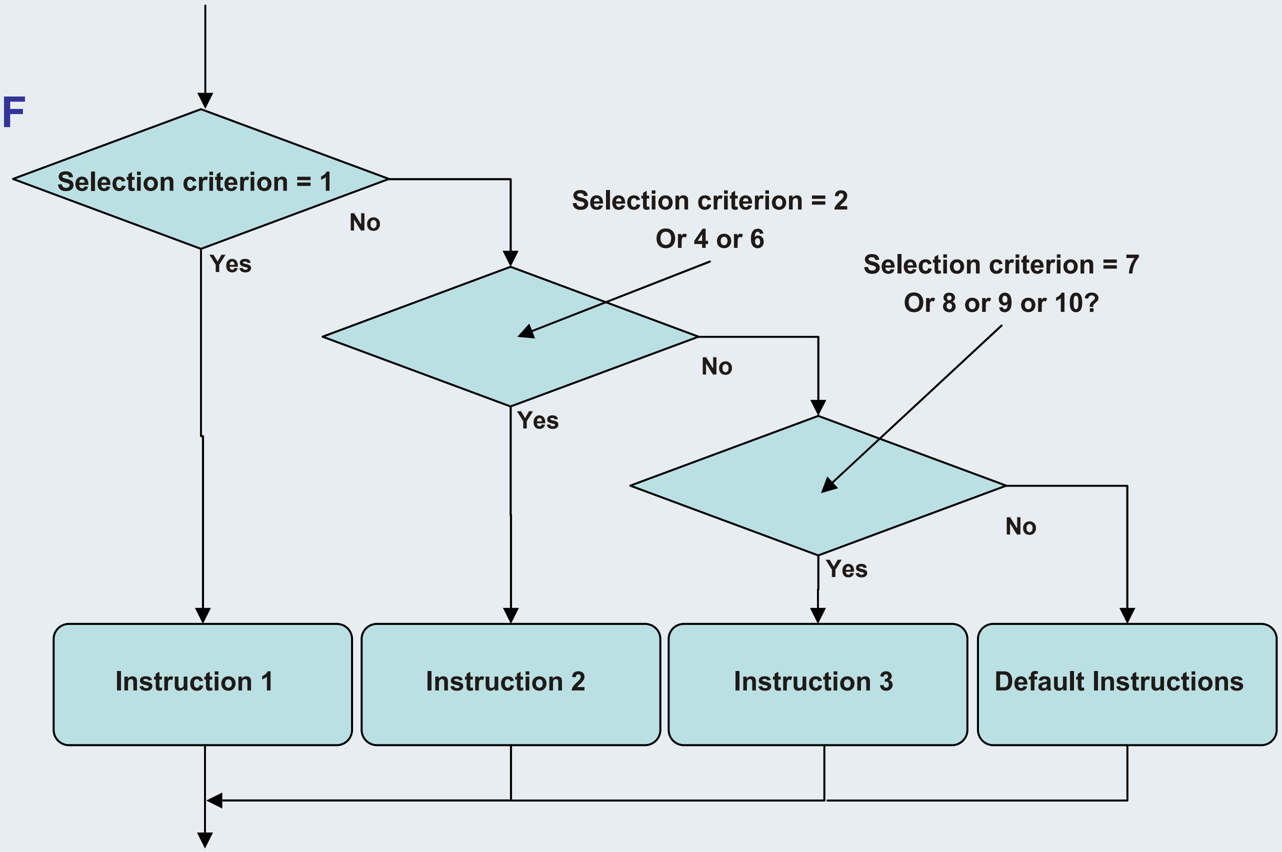
7..10 : Instruction 3

..

ELSE

Default Instructions

END_CASE;



Two same values mustn´t be available at the listing.

CASE-Instruction: Statemachine

Actions

Transitions

```
CASE State OF
   0:          Q0:=TRUE;
                    IF Transition THEN state := 1; END_IF
   1:          Q1:=TRUE;
                    IF Transition THEN state := 2; END_IF
   2:          Q2:=TRUE;
                    IF Transition THEN state := 3; END_IF
   3:          Q3:=TRUE;
                    IF Transition THEN state := 0; END_IF
END_CASE
```

CASE-Instruction: Statemachine

```
CASE State OF
  0:      Instructions;(*State=0*)
          IF ……THEN
  1:      Instructions;(*State=1*)

  2:      Instructions;(*State=2*)

  3:      Instructions;(*State=3*)

END_CASE
```

State = 0

State= 1

State = 2

State = 3

CASE Instruction Integer Selector Value with Enum types

Enum Typ:

```
TYPE Schritte :
	( INIT:=0, START, AUTOMATIK, ENDE);
END_TYPE
```

```
CASE State OF
	INIT:		instructions;(*State=0*)
	START:		instructions;(*State=1*)
	AUTOMATIK:	instructions;(*State=2*)
	ENDE:		instructions;(*State=3*)
END_CASE
```

```
CASE State OF				
	INIT:	;			State = START
	START: ;			State = START
	AUTOMATIK: ;			State = START
	ENDE:	;			State = START
END_CASE
```

If the integer selector variable state is declared as enum, the value of the variable is visible in the online mode.

```
VAR
	State:Schritte;
	(* State:INT also possible*)
END_VAR
```

CASE Instruction proposal for a Statemachine

```
TYPE Steps :
    ( INIT:=0, START, AUTOMATIC, END);
END_TYPE

CASE State OF
  INIT:       Q0:=TRUE;
              IF Transition THEN state := START; END_IF
  START:      Q1:=TRUE;
              IF Transition THEN state := AUTOMATIC; END_IF
  AUTOMATIC:  Q2:=TRUE;
              IF Transition THEN state := END; END_IF
  END:        Q3:=TRUE;
              IF Transition THEN state := INIT; END_IF
END_CASE
```

Instruction for the step

(Actions)

„step enabling condition“

(Transition)

Step

CASE Instruction Integer Selector Value with constants

```
VAR CONSTANT
    Step1 : INT:=   0;
    Step2 : INT:=   1;
    Step3 : INT:=   2;
    Step4 : INT:=   3;
END_VAR

VAR
    State:INT;
END_VAR
```

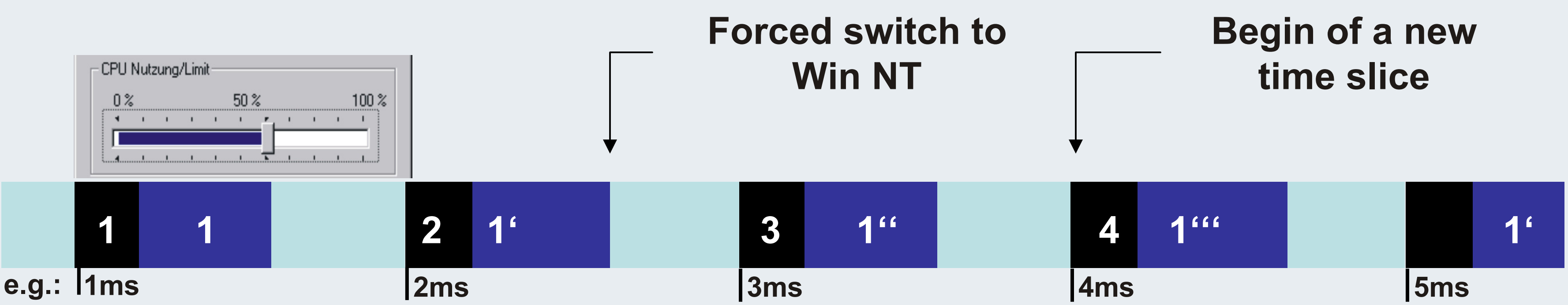
```
CASE State OF
  Step1:          instructions;(*State=0*)
  Step2:          instructions;(*State=1*)
  Step3..Step4:   instructions;(*State=2 oder 3*)
END_CASE
```

Repeat Instructions

The process flow requires the multiple handling of exactly the same program sequences, whose quantitiy is known at the run time.

Disadvantage of loops:
During faulty programming, many repetitions take place infinitely.

If a continuous loop is executed this does not impair the start of the time slice (real-time). Tasks that will have a higher priority are still executed on time. Tasks that will have a lower priority are not longer executed.



Loops (Overview)

All loops can be ended with the EXIT instruction, regardless of the break-off condition.

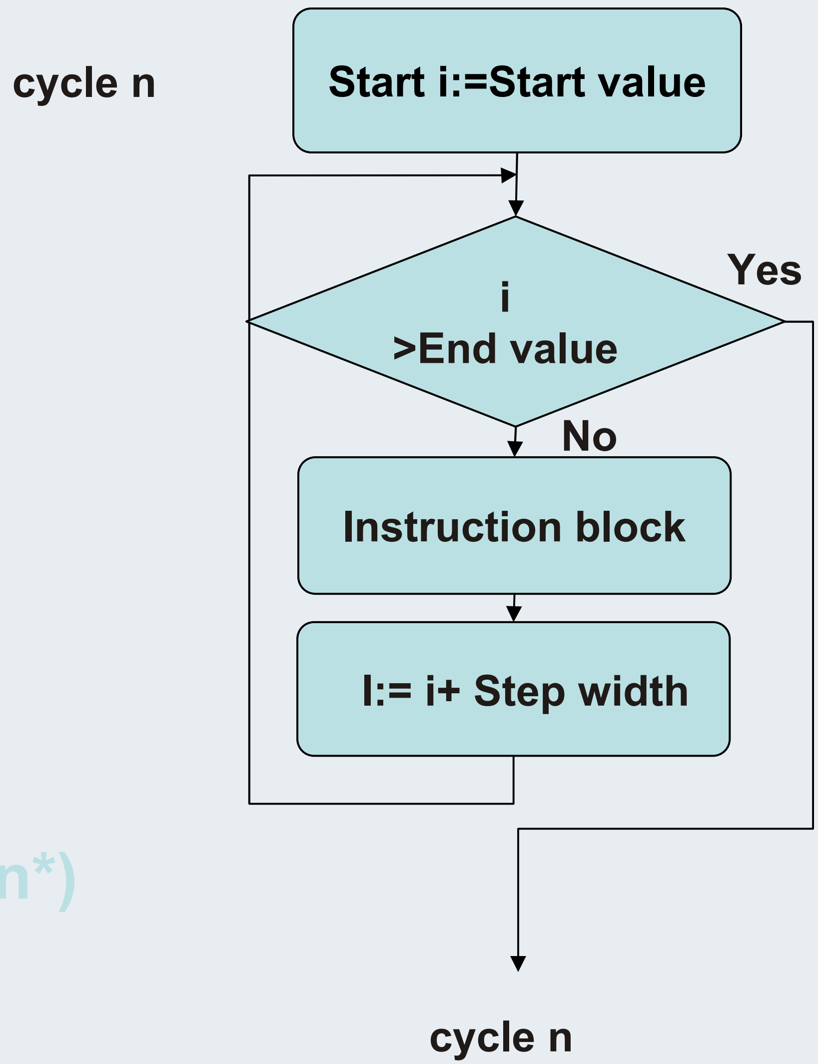
	Expression	Work flow	n cycle fix
FOR	SINT/ INT / DINT	Pre repel	Yes
WHILE	BOOL	Pre repel	No
REPEAT	BOOL	Post repel	No

FOR loop

At the beginning of the loop, the variable i is defined as start value (see example).
The variable in incremented or decremented in each cycle depending on the step width (value after the keyword BY)

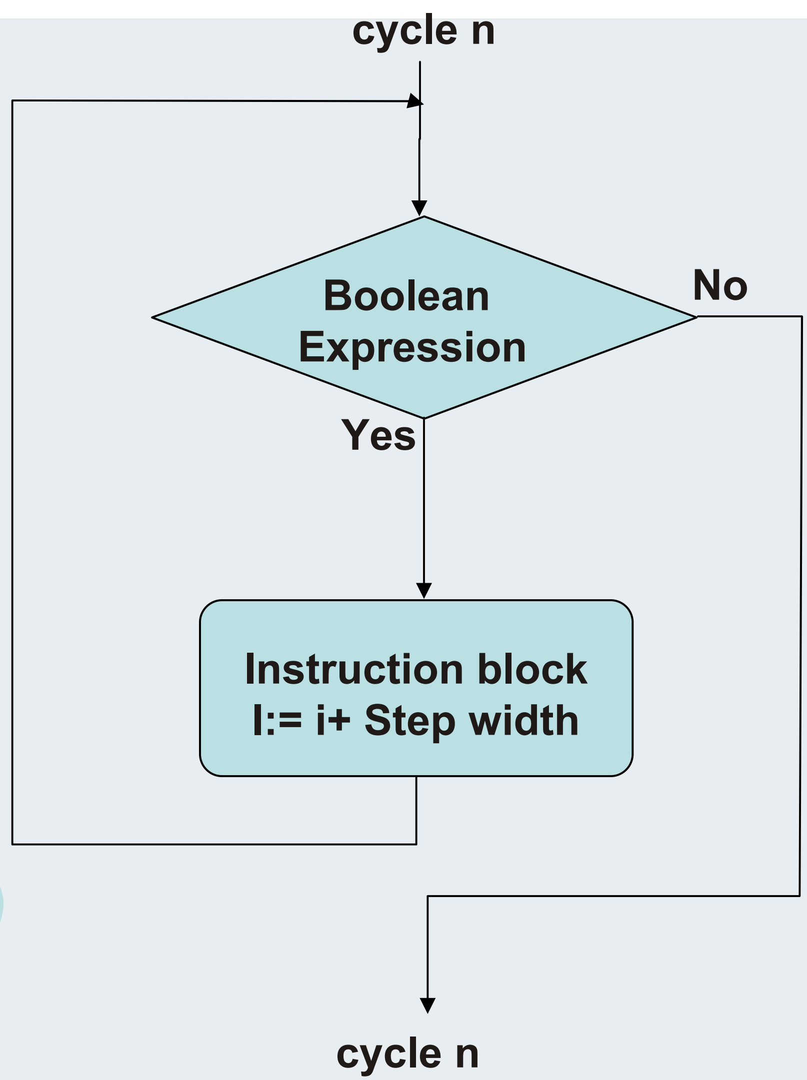
If i exceeds the end value (after TO), the loop is not longer processed.

```
FOR  i:=1 TO 12 BY 2 DO
    Field[i]:=i*2;(*instruction*)
END_FOR
```



WHILE loop

The instruction block of a WHILE loop is executed as long as the boolean expression supplies TRUE .
The exit condition contains variables which can be changed in the instruction block.
If the boolean expression is FALSE at the beginning, the instruction block of the WHILE loop is not processed.



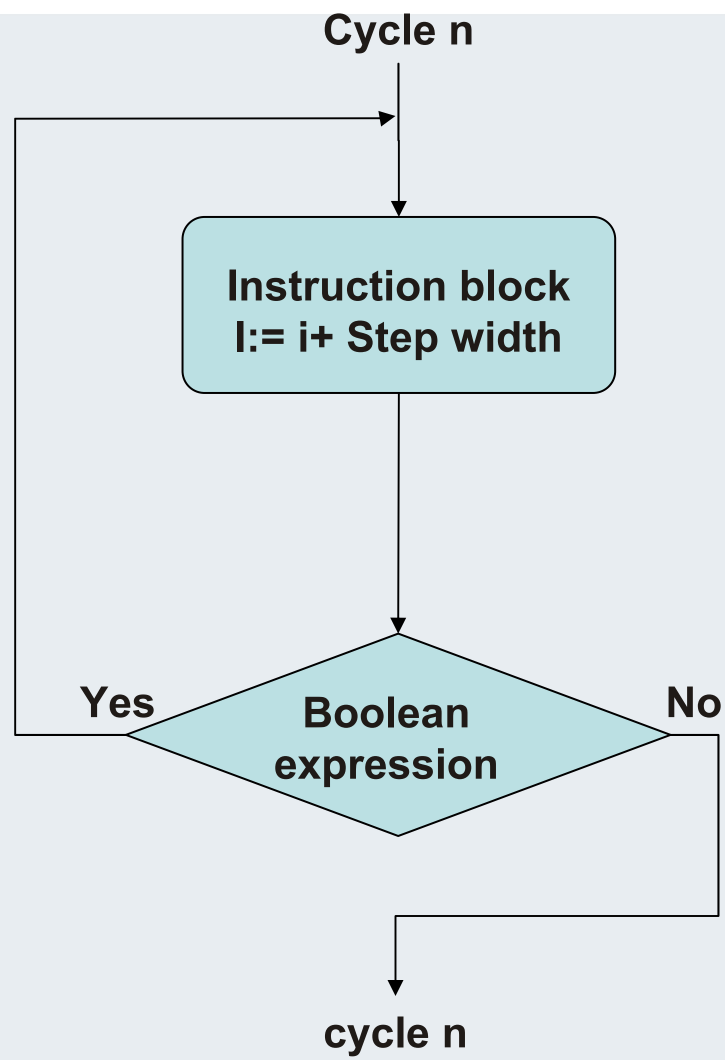

```
i:=0;
WHILE  i<100 DO
   Field[i]:=i*2;(*instruction*)
   i:=i+1;
END_WHILE
```

REPEAT loop

The instruction block of a REPEAT loop is processed as long as (UNTIL) the boolean expression is no longer fullfilled.

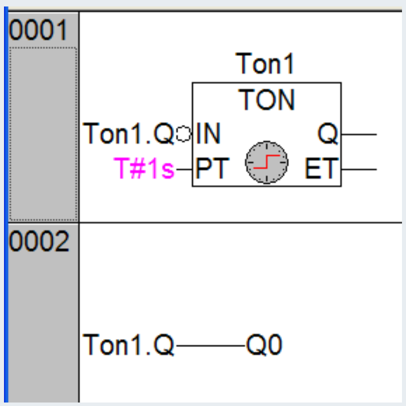
The instruction block is executed at least once.

```
i:=0;

REPEAT
    Field[i]:=i*2;(*Instruction*)
    i:=i+1;
UNTIL i>100

END_REPEAT
```

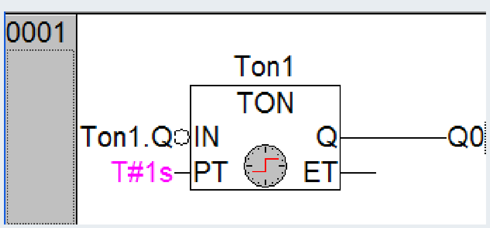


Fb calls in ST

```
VAR
  TON1:TON;
END_VAR
```



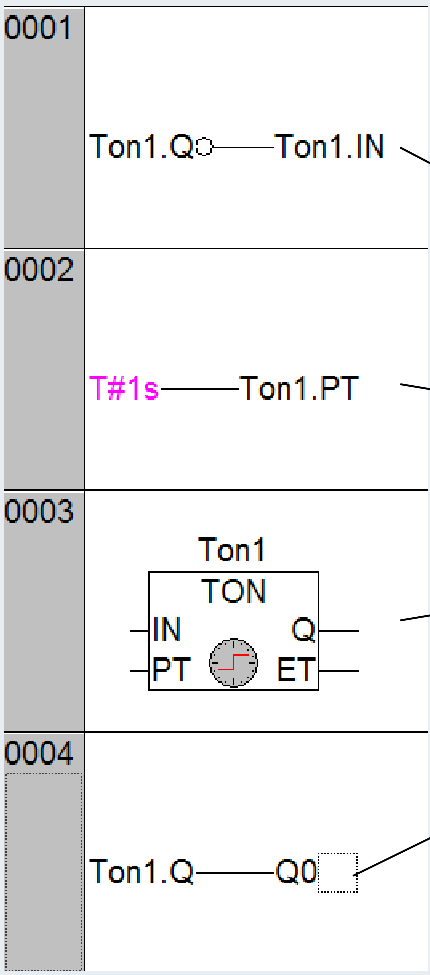

```
TON1 (IN:= NOT TON1.Q , PT:=T#1s );
Q0:= TON1.Q;
```




```
TON1(IN:= NOT TON1.Q, PT:=T#1s , Q=>Q0 );
```

FB calls in ST (alternative)

```
VAR
  TON1:TON;
END_VAR
```




```
TON1.IN:= NOT TON1.Q;
TON1. PT:=T#1s;
TON1();
Q0:= TON1.Q;
```

FC calls in ST

```
Result:=Scale    (x:=input, xug:=0.0, xog:=32767.0, yug:=0.0,yog:=100.0);
(* equal:*)
Result:=Scale    (input, 0.0, 32767.0, 0.0, 100.0);
(* equal :*)
Result:=Scale    (
                 x:=          input,
                 xug:=   0.0,
                 xog:=   32767.0,
                 yug:=   0.0,
                 yog:=   100.0
                 );
```

FC calls in ST explanation

```
Result := Scale (x:=input, xug:=0.0, xog:=32767.0, yug:=0.0,yog:=100.0);
```

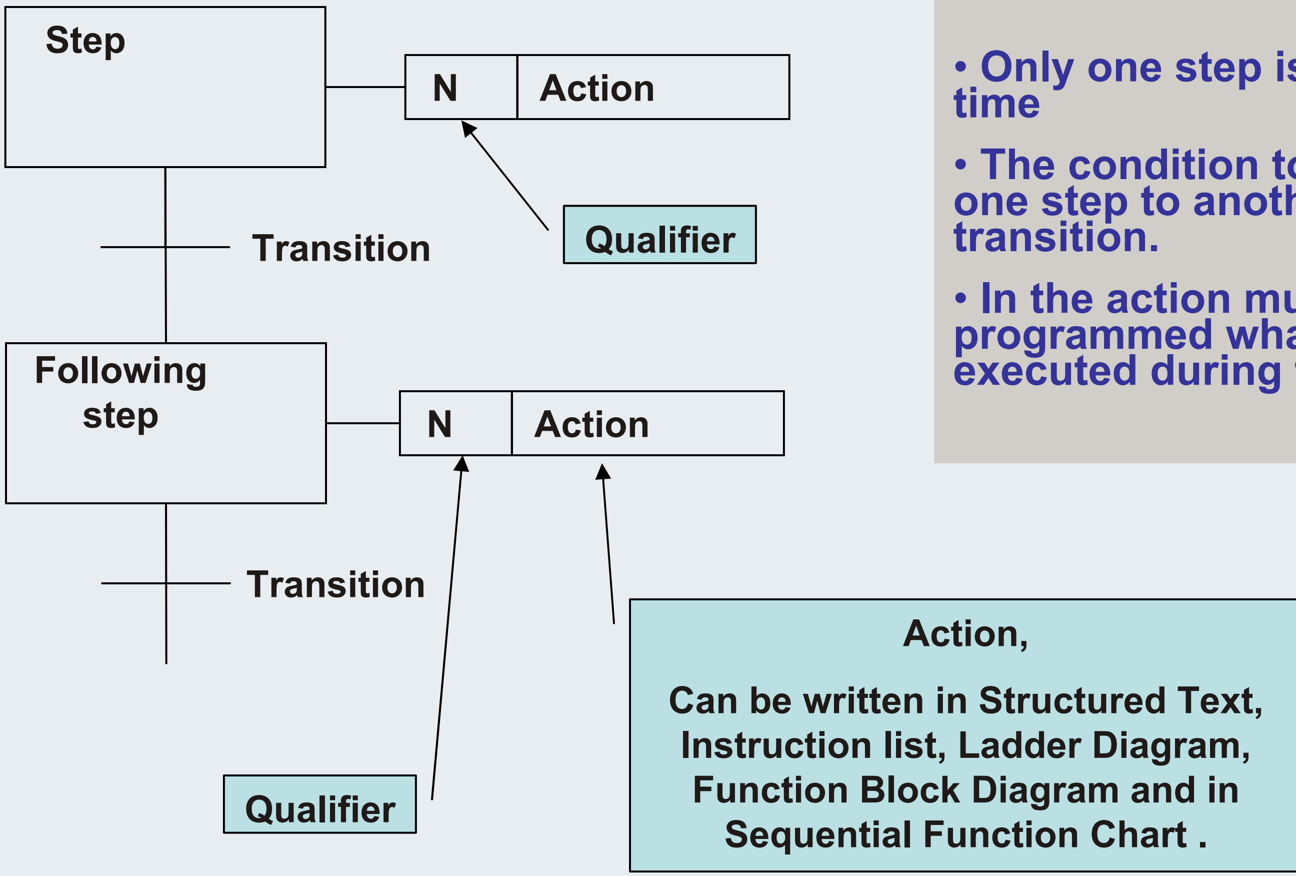
Result

CALL

Input parameters

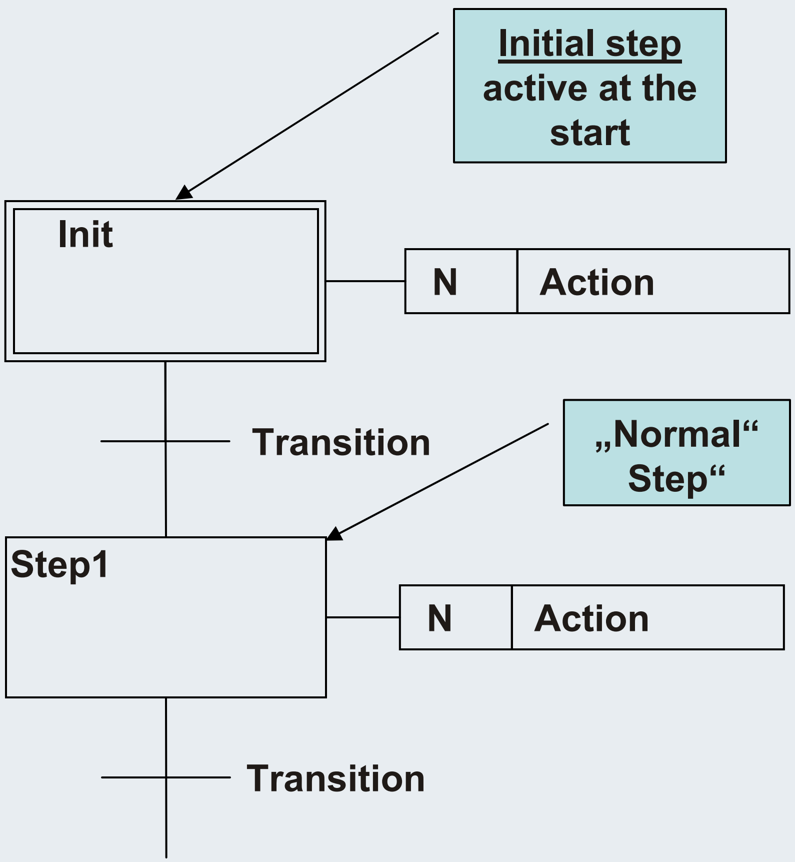
```
(* equal:*)
Result:=Scale (
        x:=     input,
        xug:=   0.0,
        xog:=   32767.0,
        yug:=   0.0,
        yog:=   100.0
        );
```

SFC Sequential Function Chart



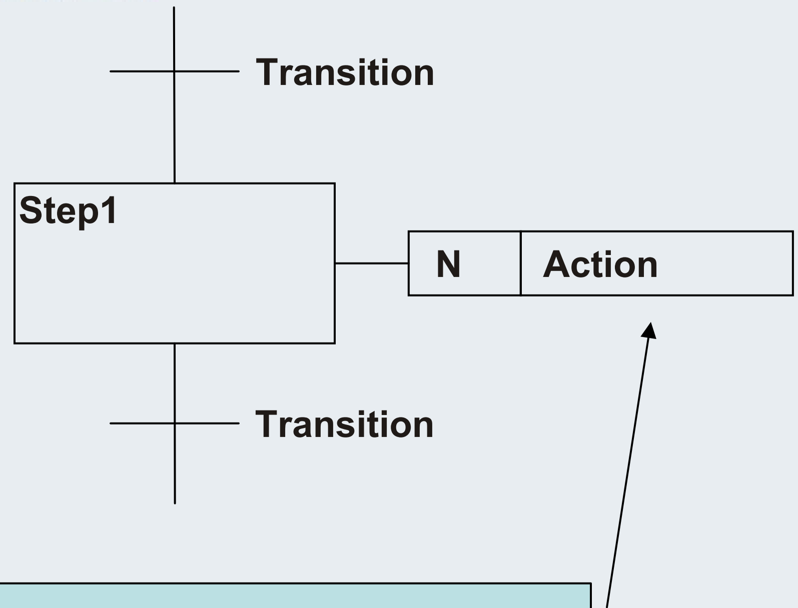
- Only one step is active at a time
- The condition to change from one step to another is the transition.
- In the action must be programmed what should be executed during the active step.

Steps



- The activity of a step can be requested with Stepname.X.
- The duration of the activity of a step can be requested with Stepname.T .
- Both are components of a structure, which are created automatically from PLC Control. At the programming only the stepname has to be defined.
- Stepname.X and Stepname.T are local variable and can only be read.

Actions (92)



Action, can be programmed in

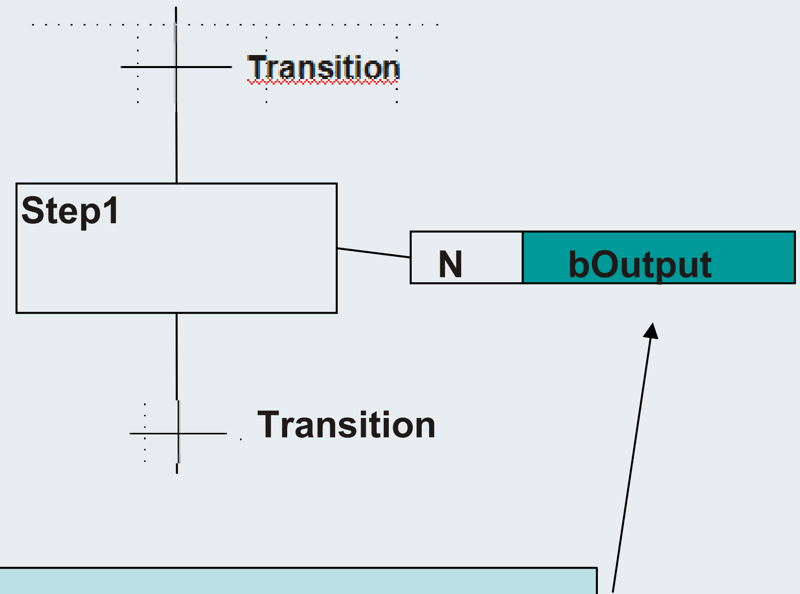
-> Structured Text,

-> Instruction List,

-> Ladder Diagram,

-> Function block diagram, CFC/FBD

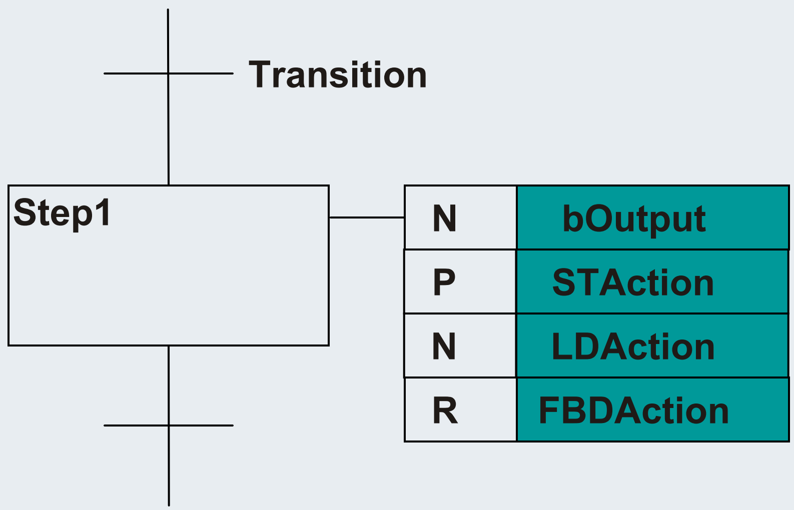
-> Sequential Function Chart



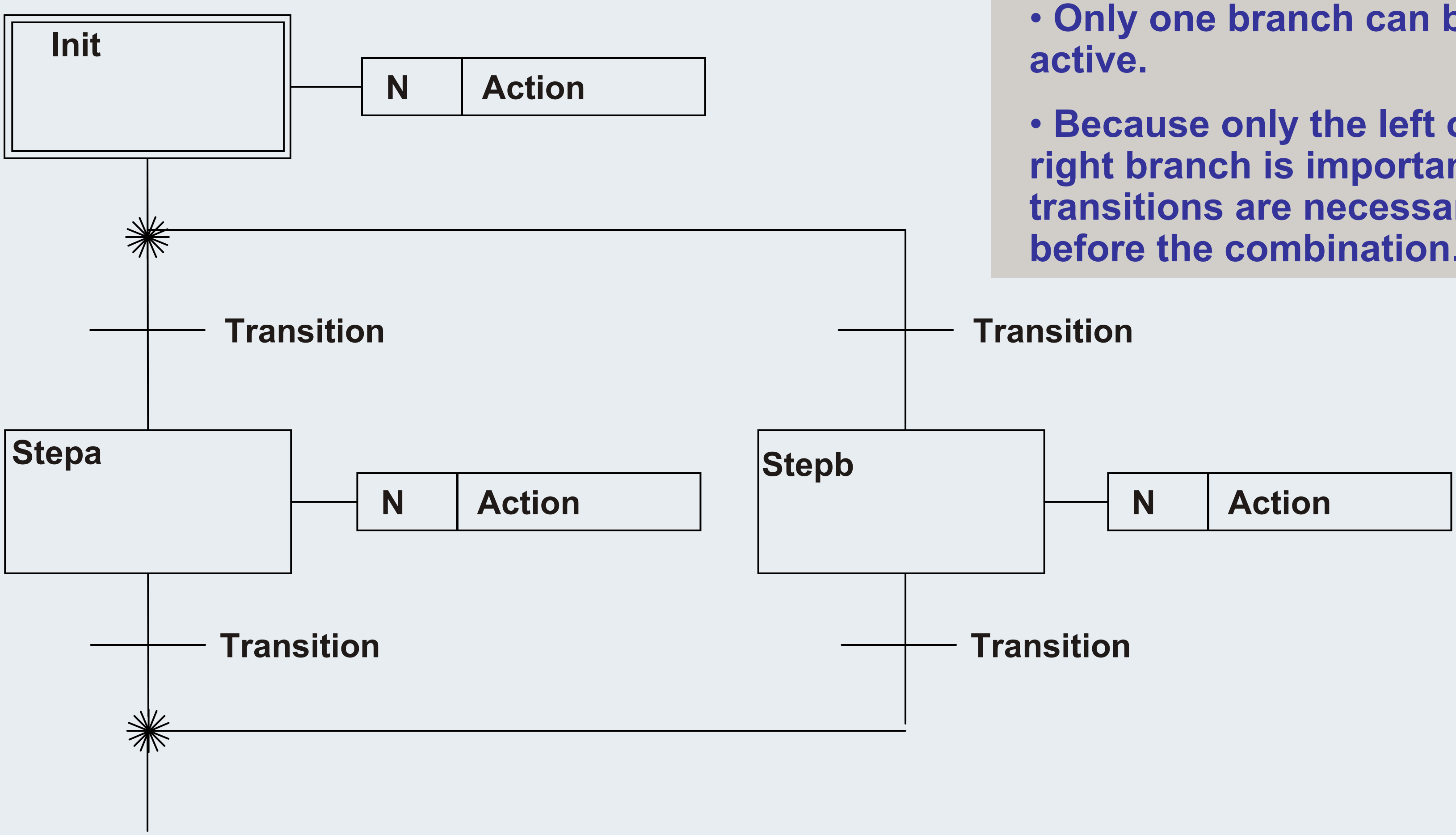
Action,
can be a variable of type BOOL.

The variable is TRUE by activating the the step and FALSE by leaving the step.

Actions, several allowed per step (93)



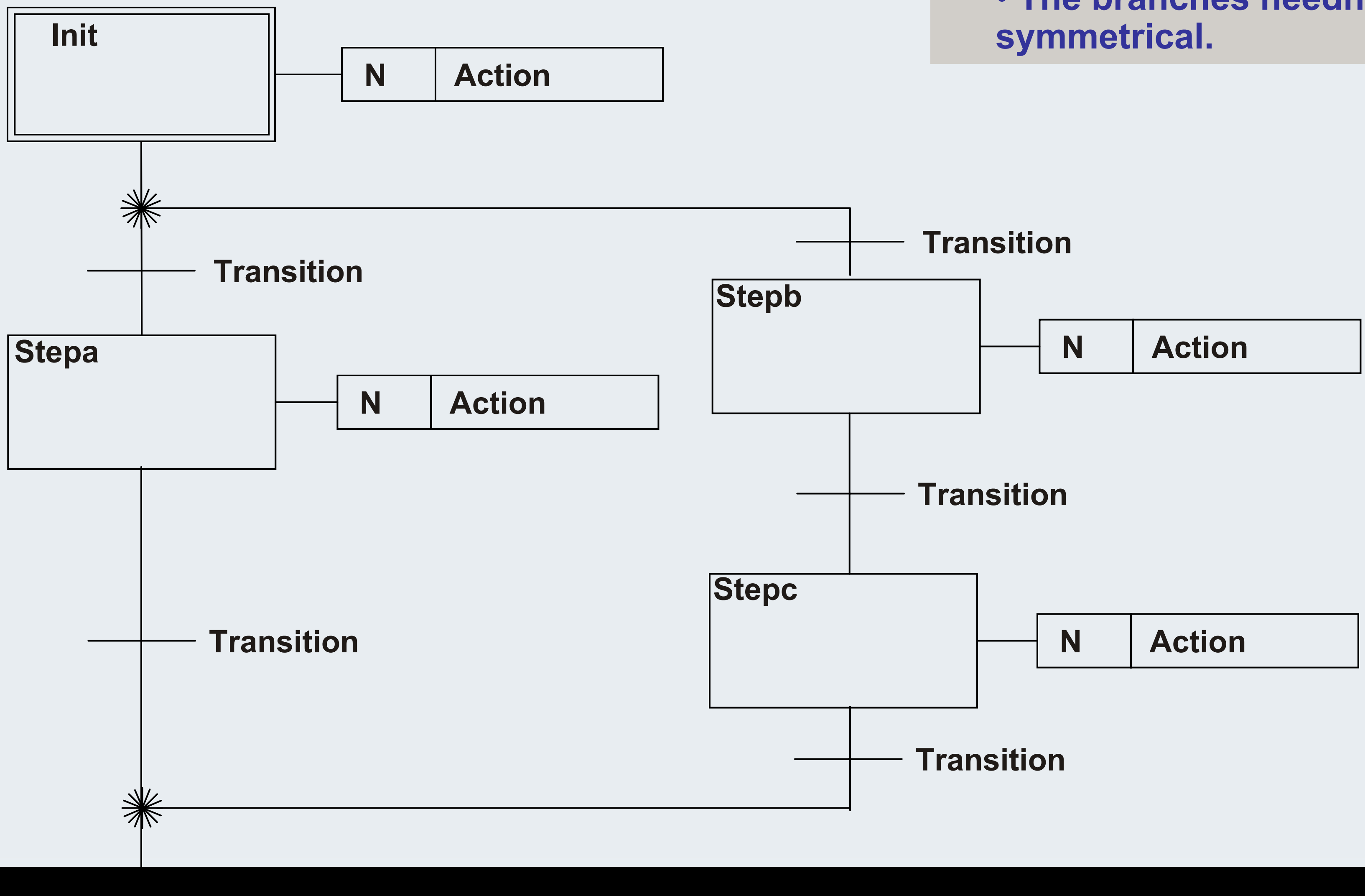
Steps /alternative branches



- Only one branch can be active.
- Because only the left or the right branch is important, two transitions are necessary before the combination.

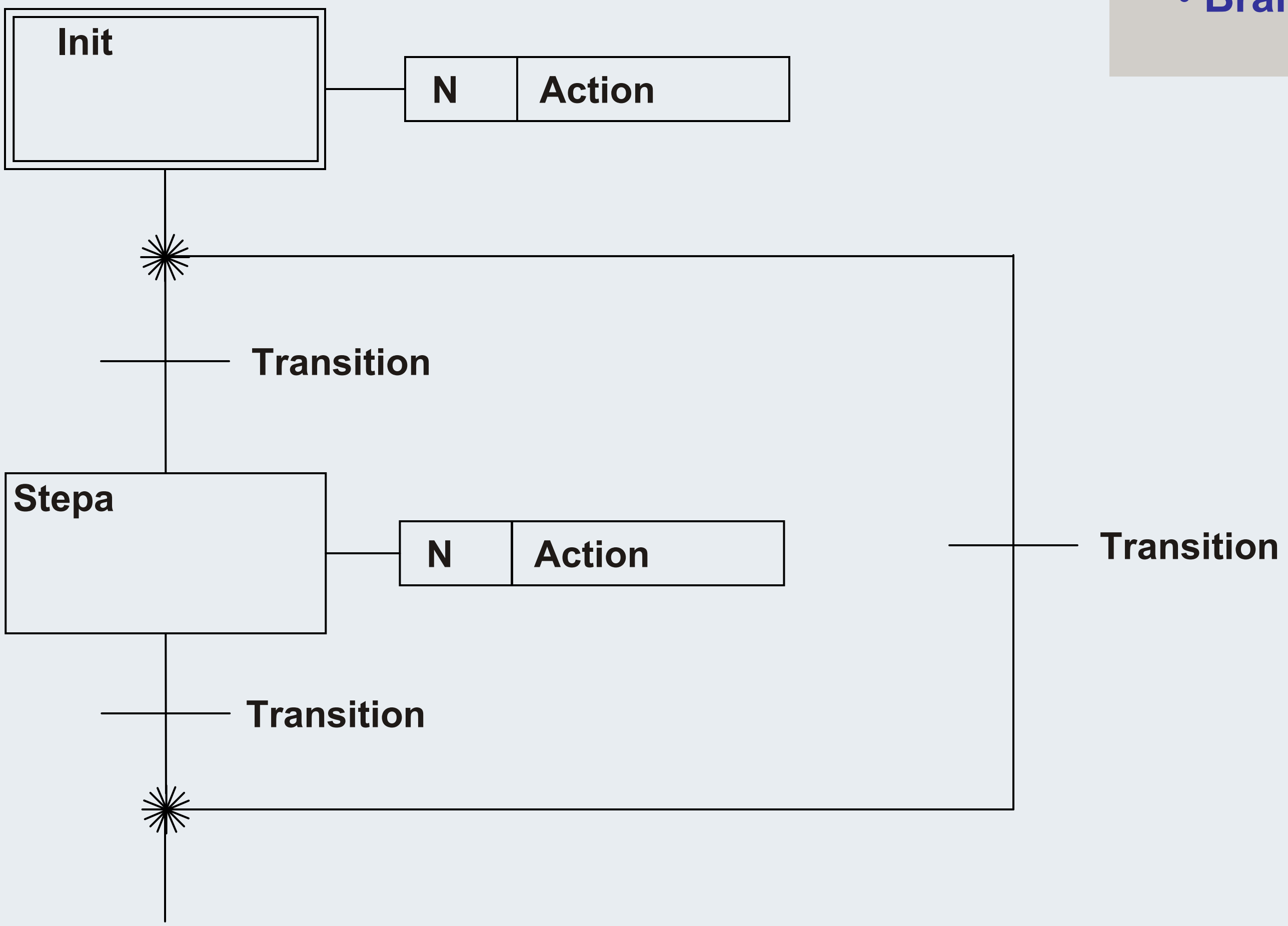
Steps / alternative branches

- The branches needn´t be symmetrical.



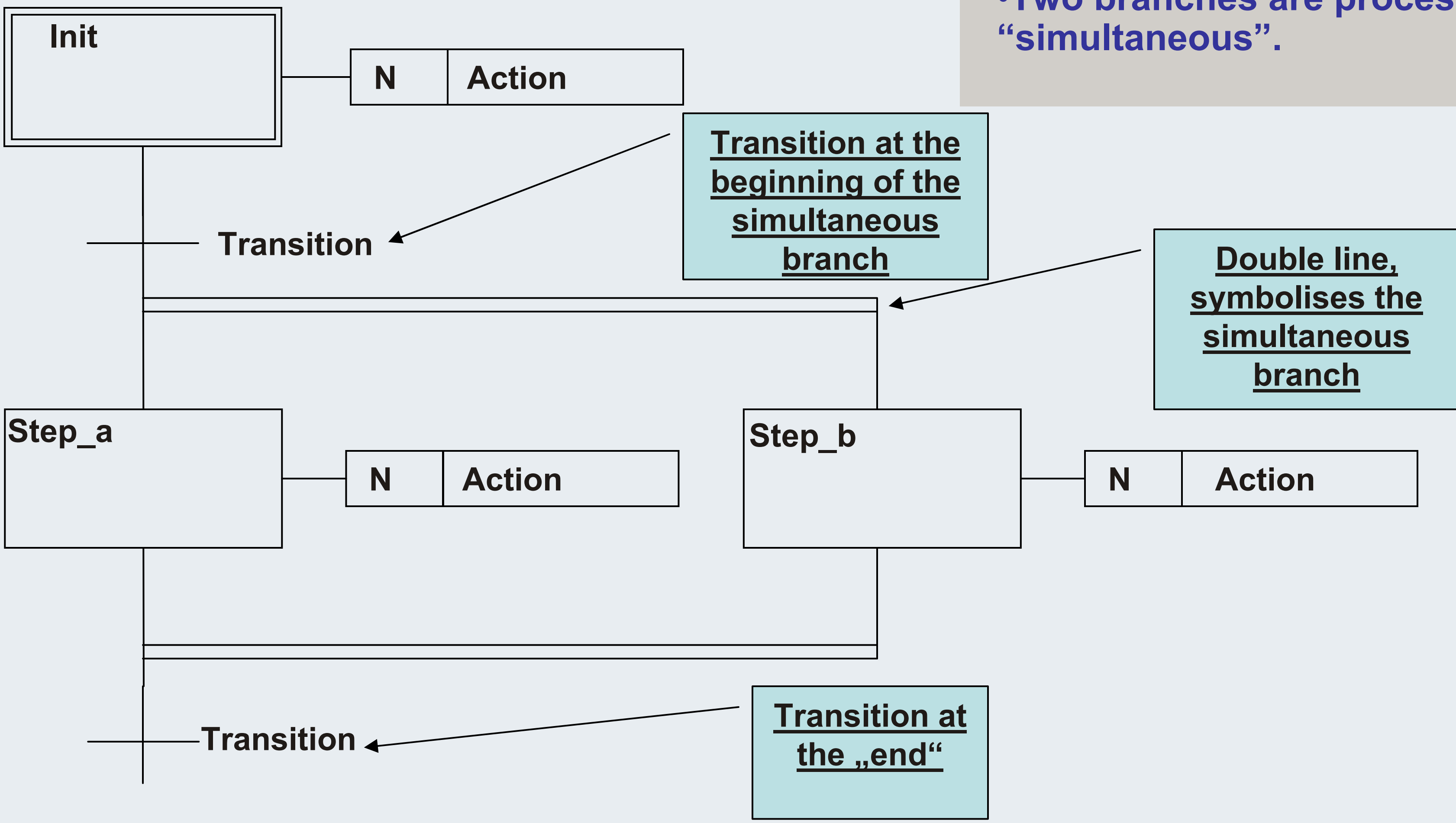
Steps /alternative branches

- Branches can be skipped.

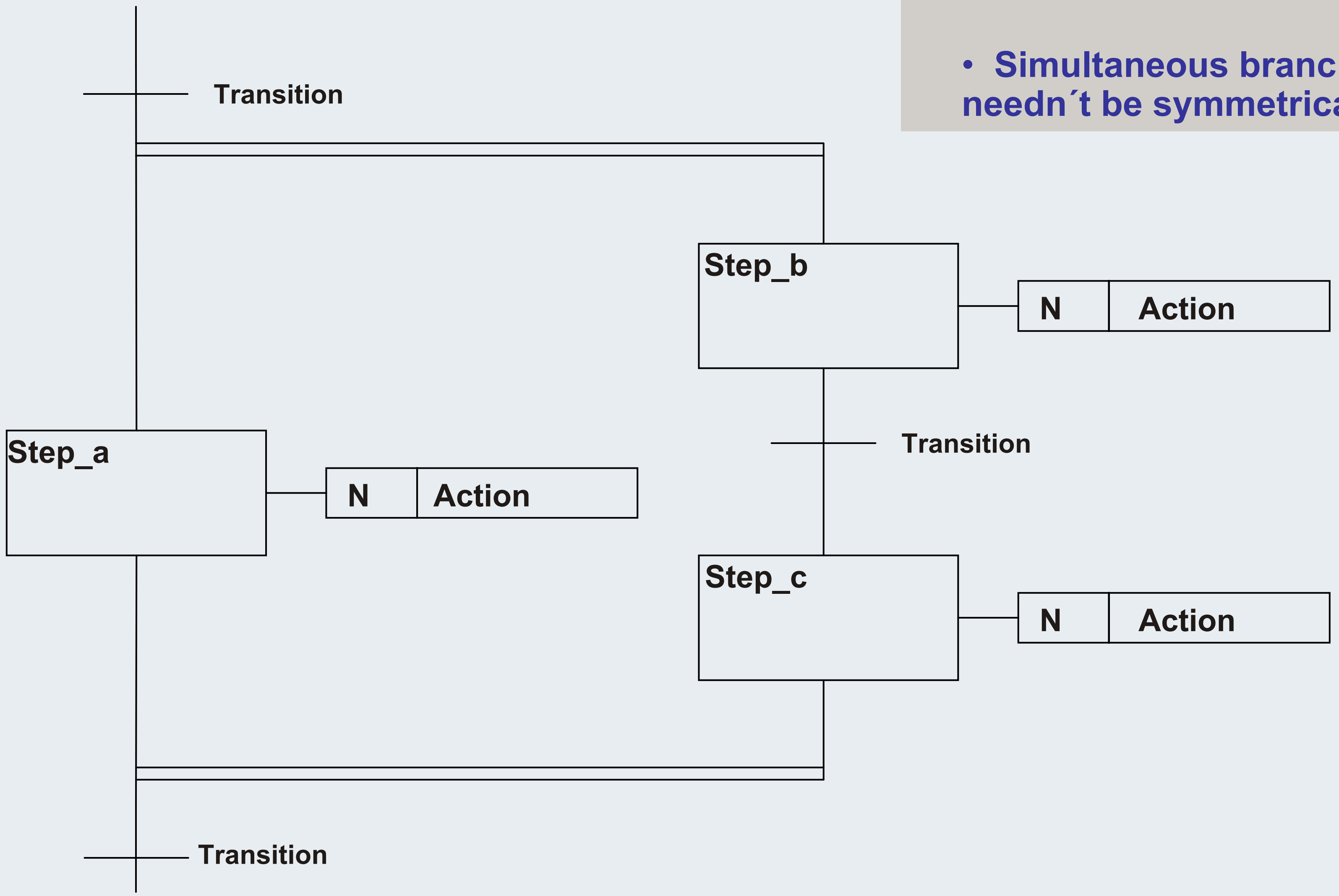


Steps /simultaneous branches

•Two branches are processed "simultaneous".

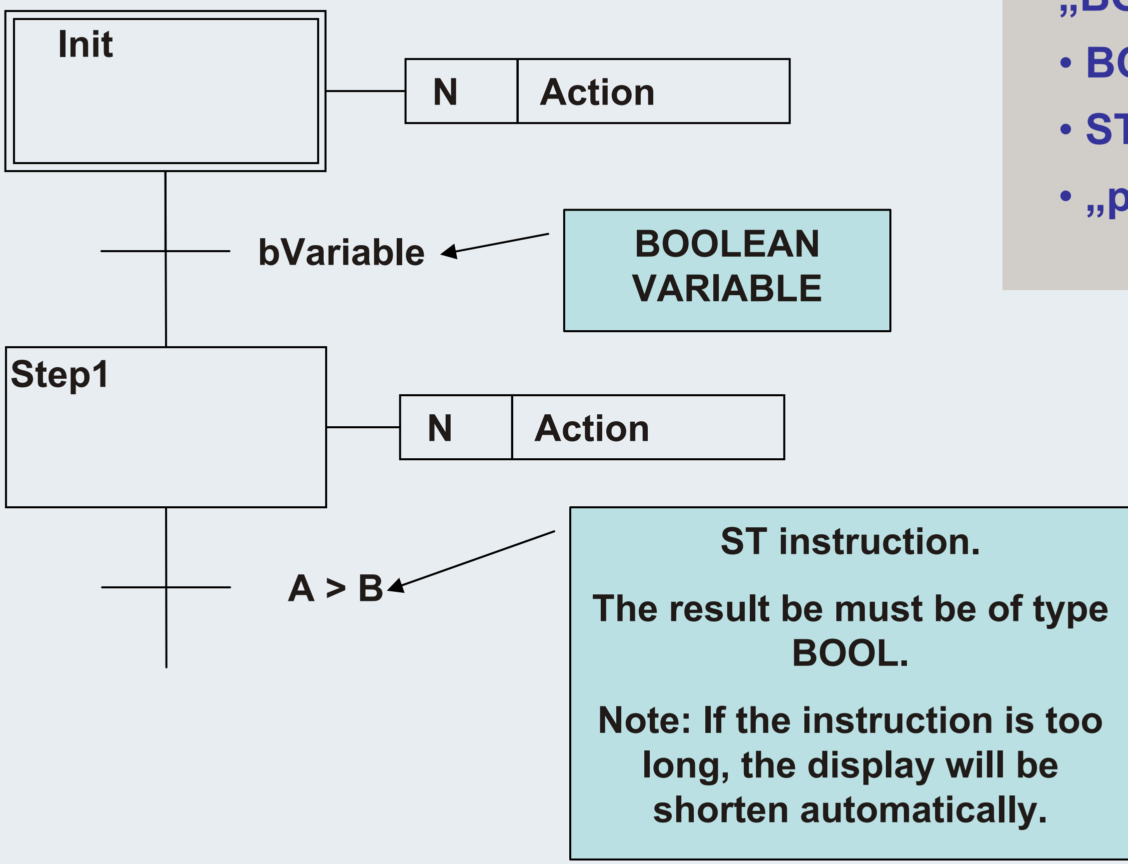


Steps /simultaneous branches



- Simultaneous branches needn´t be symmetrical.

Transitions

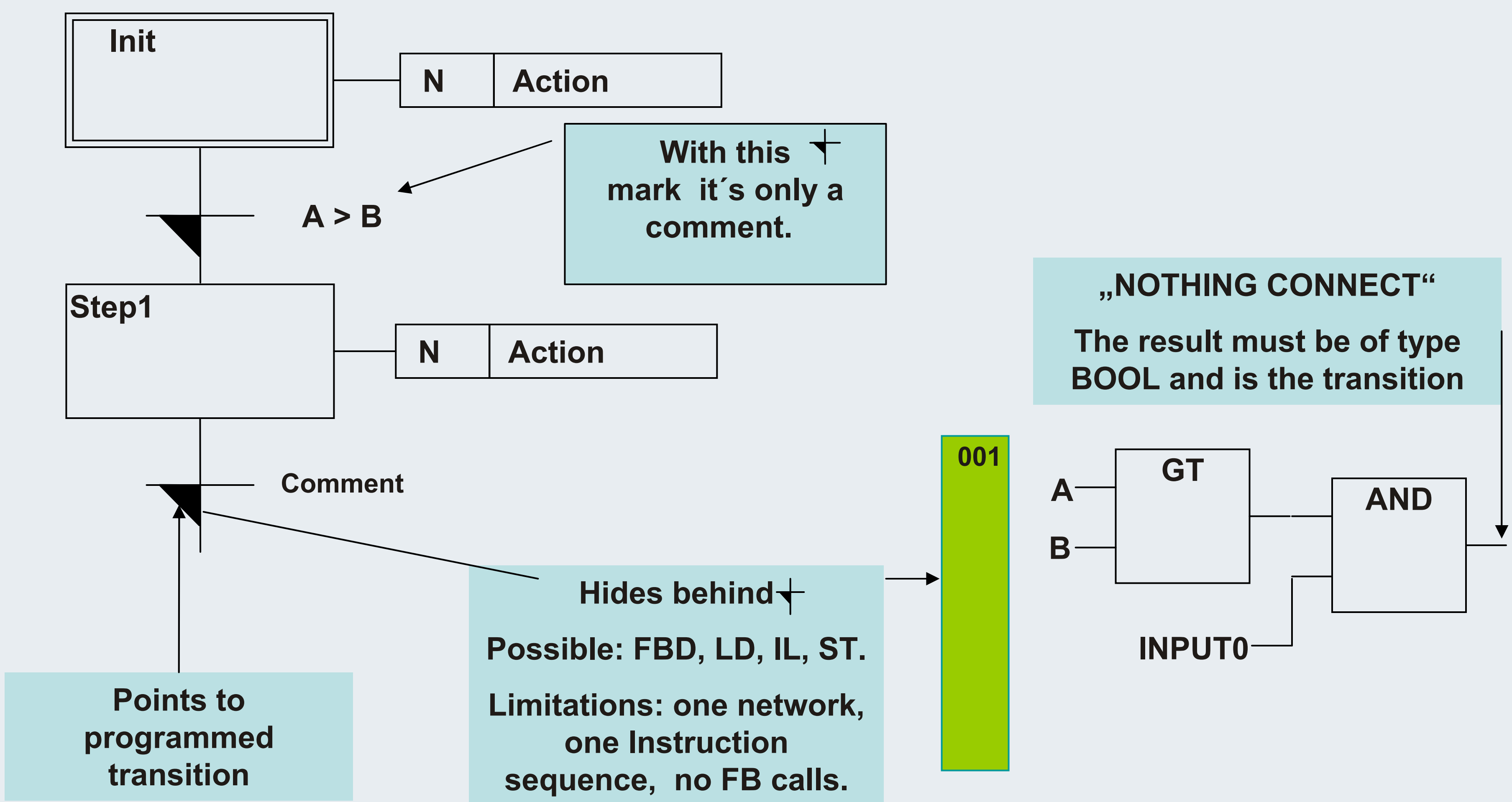


A Transition must be of type „BOOL“. Possibilities:

- BOOLEAN Variable
- ST Instruction
- „programmed“ Transition

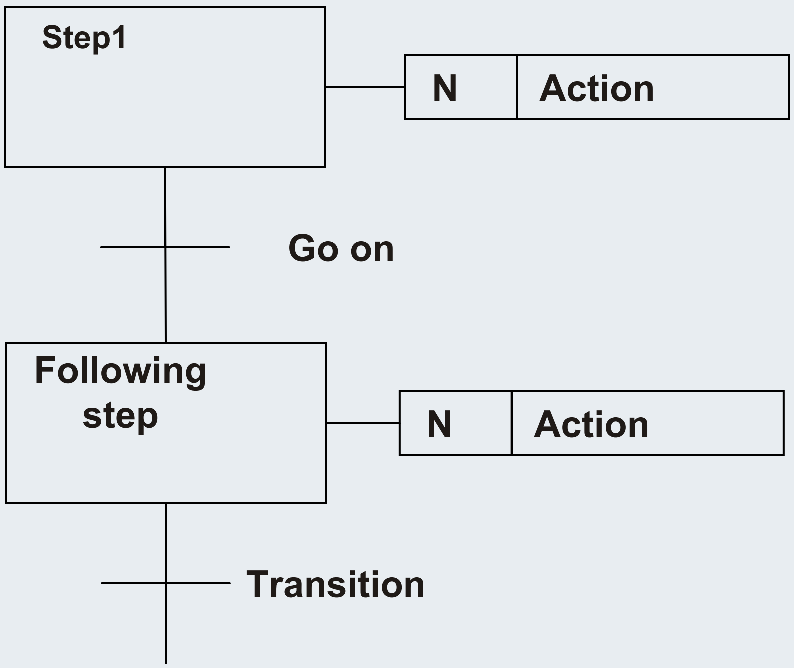
Transitions

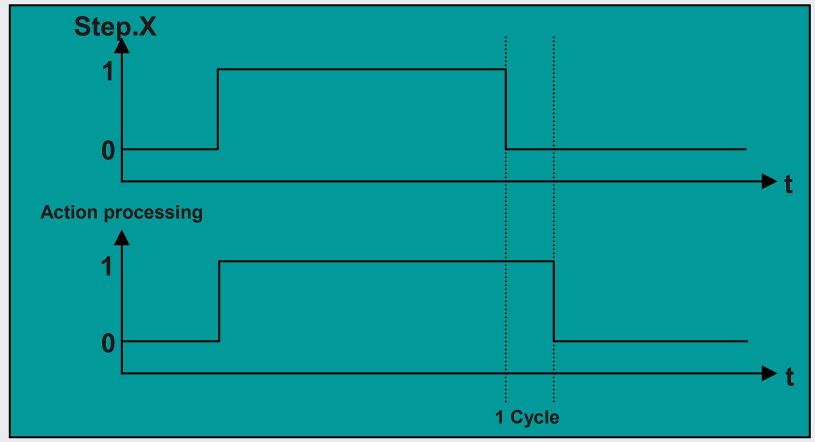
Programmed Transitions

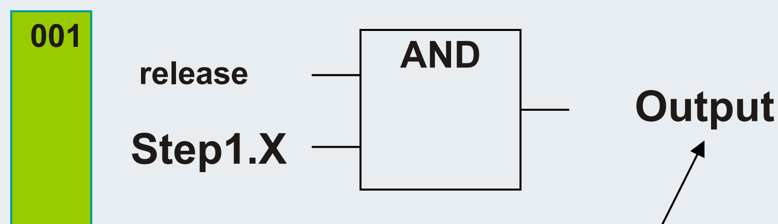


Final Scan

If a step is left, the processing takes exactly one more cycle. This behaviour can be used for “cleaning”in the action. Example: Reset outputs.



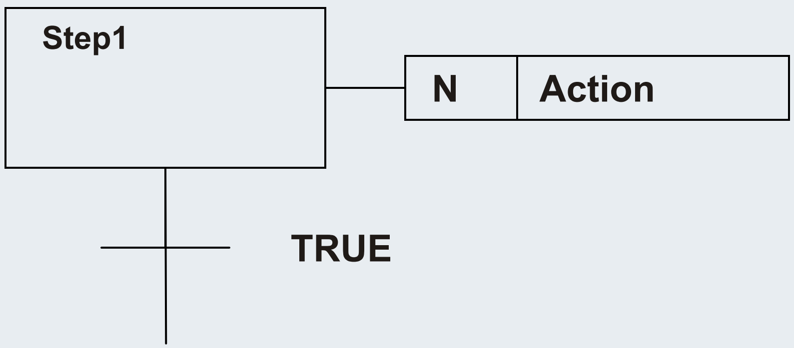


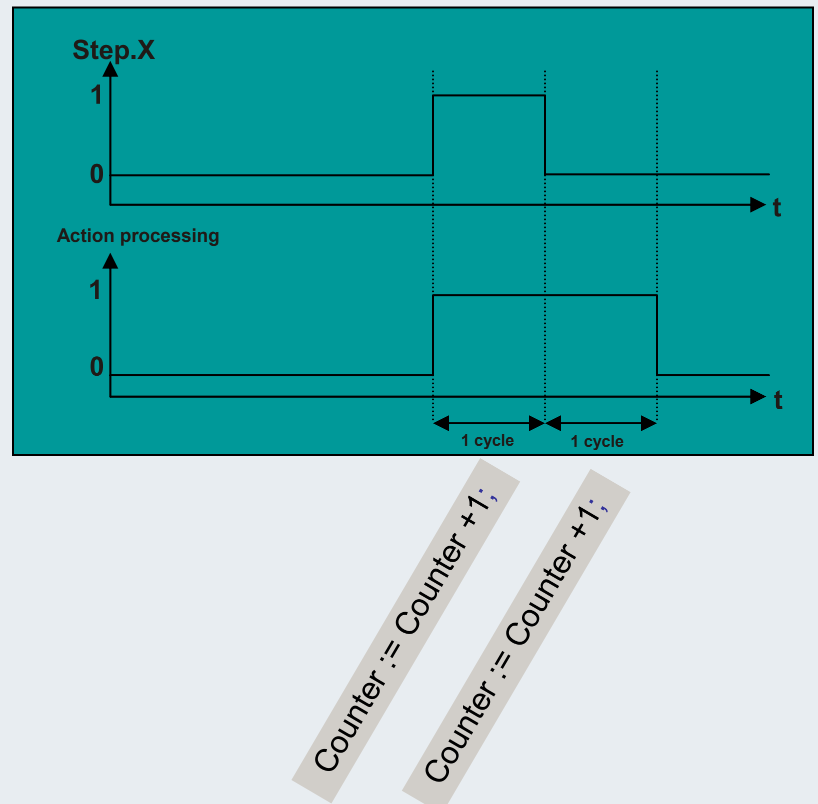


At the last pass the step.X = FALSE. Thus the variable „Output “ is FALSE .

Final Scan

At a certain action the final scan leads to an unwanted behaviour.





Behaviour:

Counter := Counter +1;

(*Counter increases at 2*)

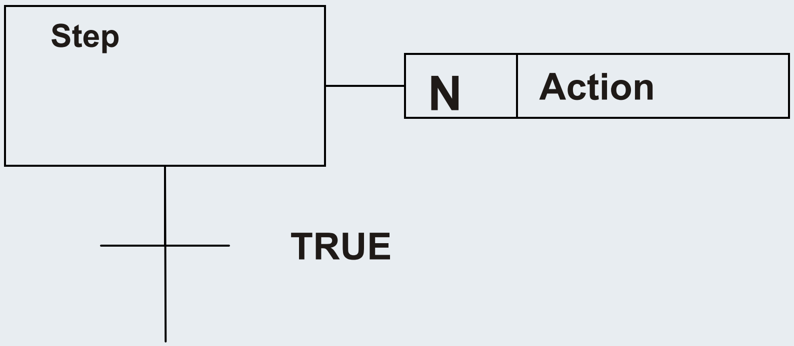
Remedy: The step flag is only for one cycle 1:

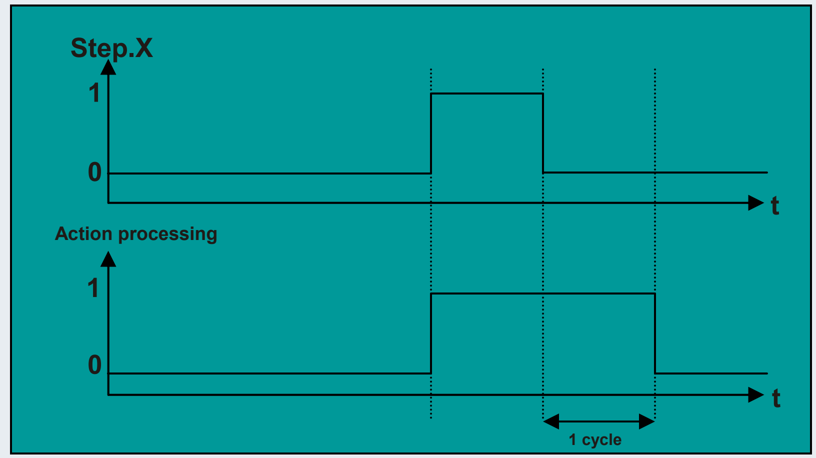
```
IF Schritt.X THEN
    Counter := Counter +1;
END_IF
(*Counter increases at 1*)
```

Qualifier

Controls the action processing after activating a step.

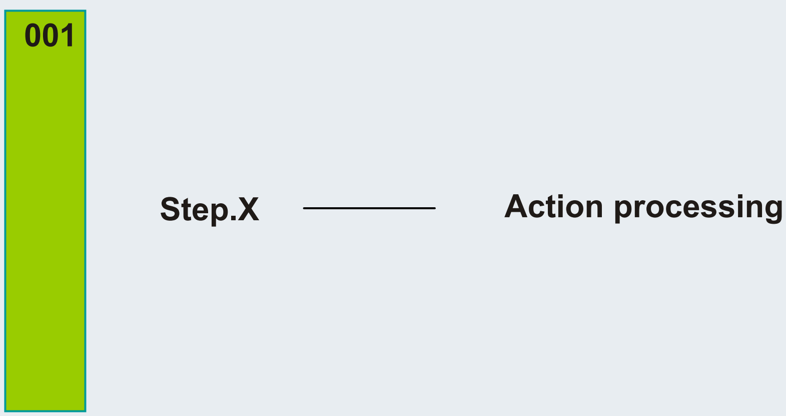
N: Non Stored





N: Non Stored

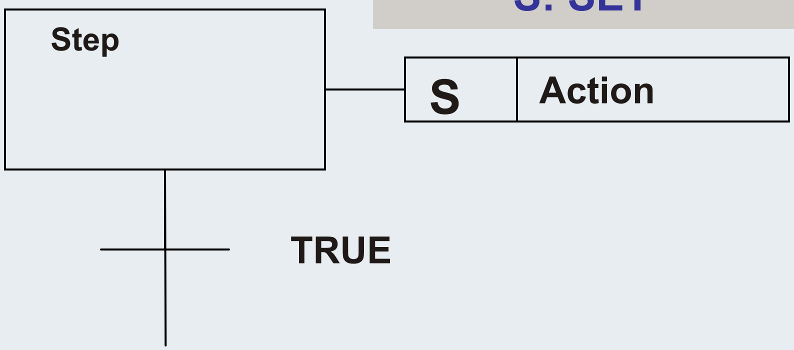
Combination in FBD

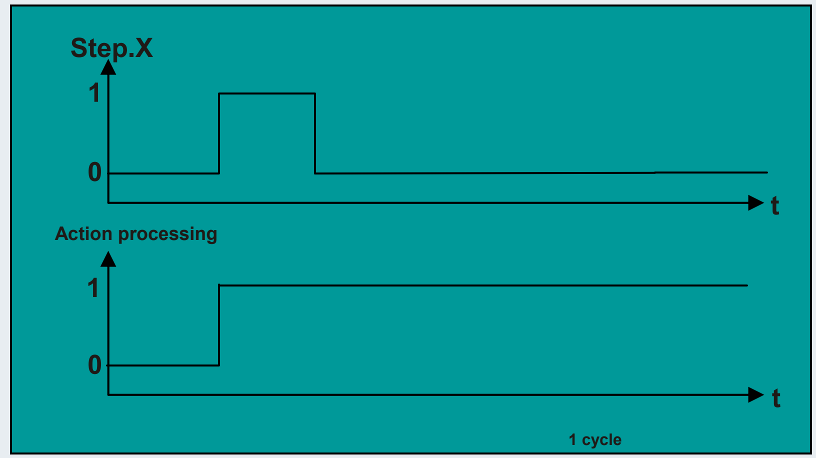


Qualifier

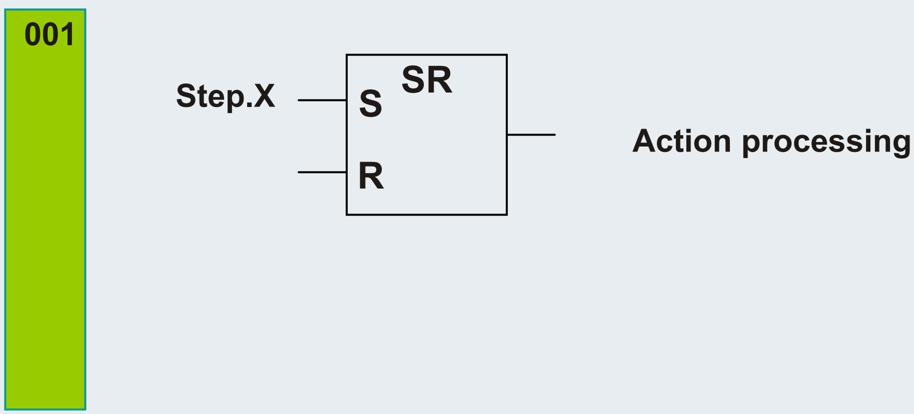
Controls the action processing after activating a step

S: SET





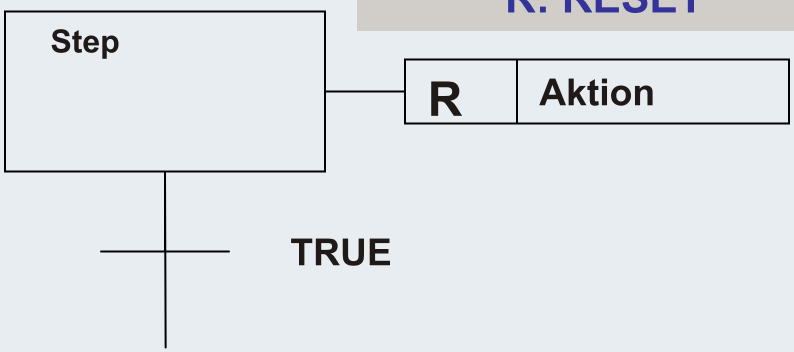
Combination in FBD

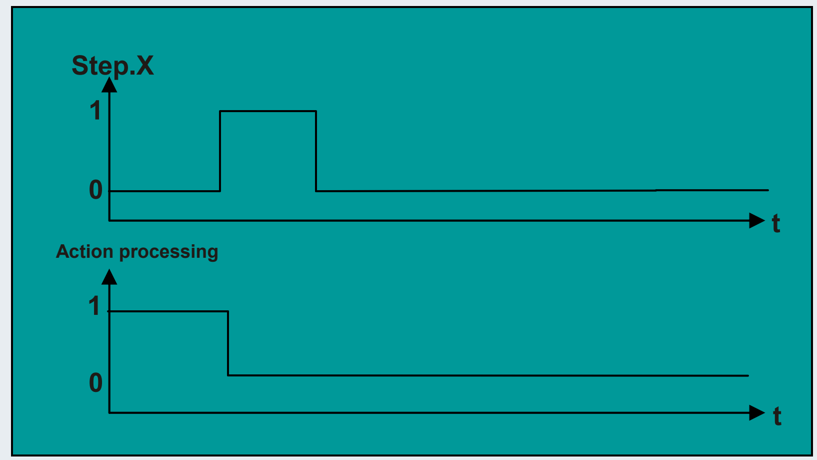


Qualifier

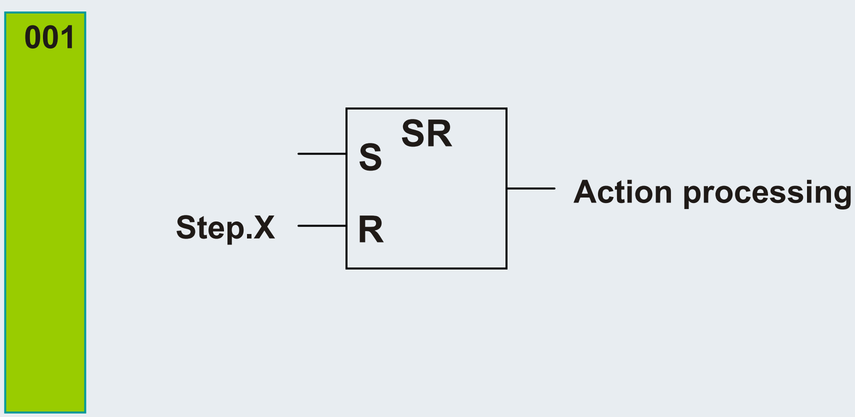
Controls the action processing after activating a step

R: RESET





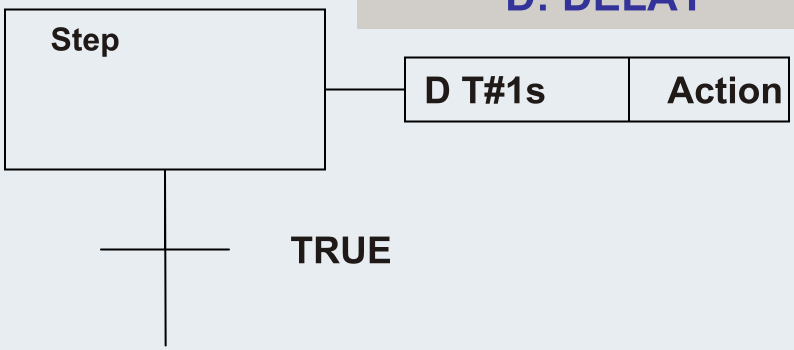
Combination in FBD

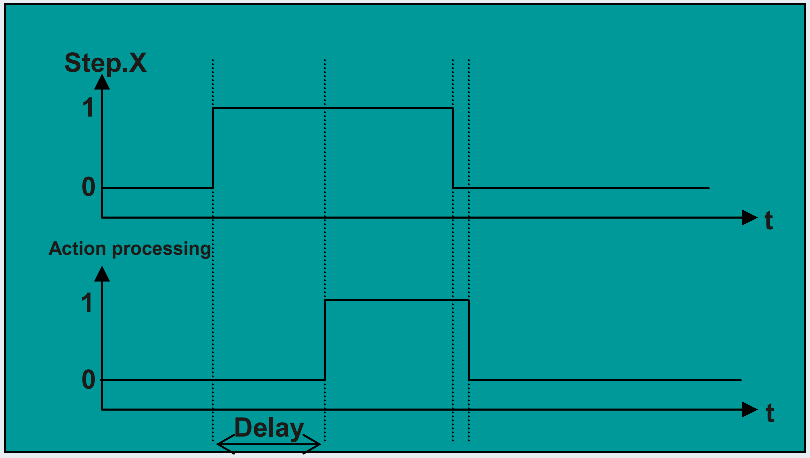


Qualifier

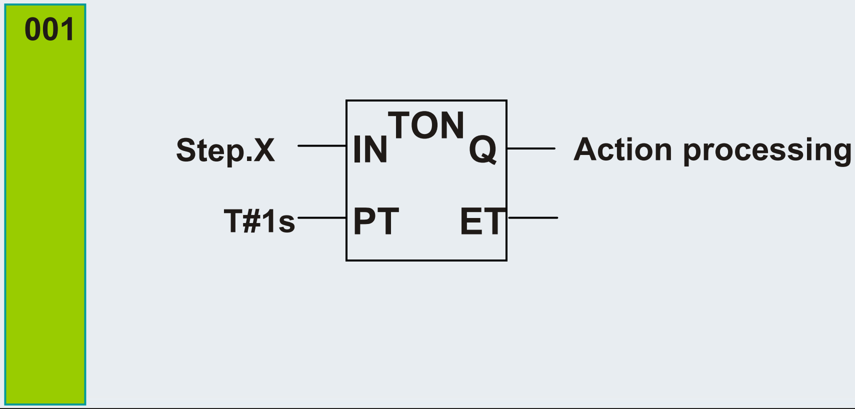
Controls the action processing after activating a step

D: DELAY





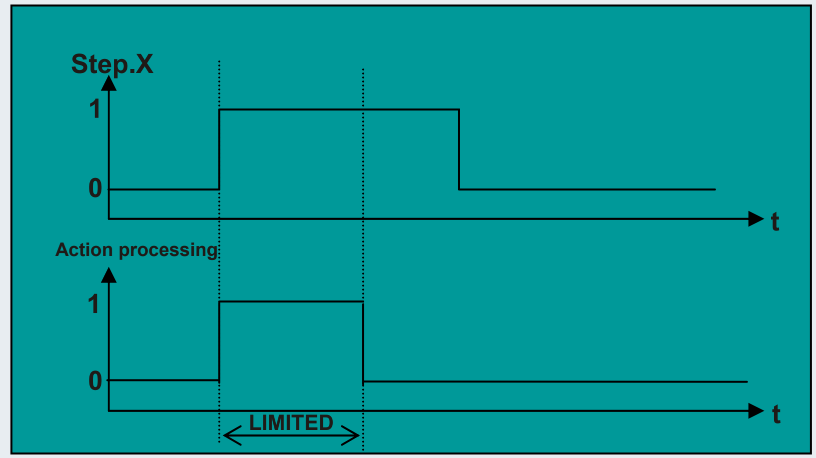
Combination in FBD



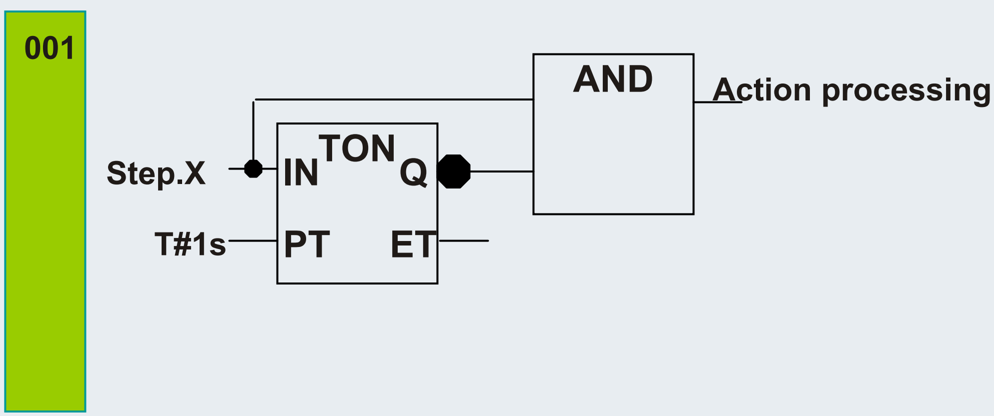
Qualifier

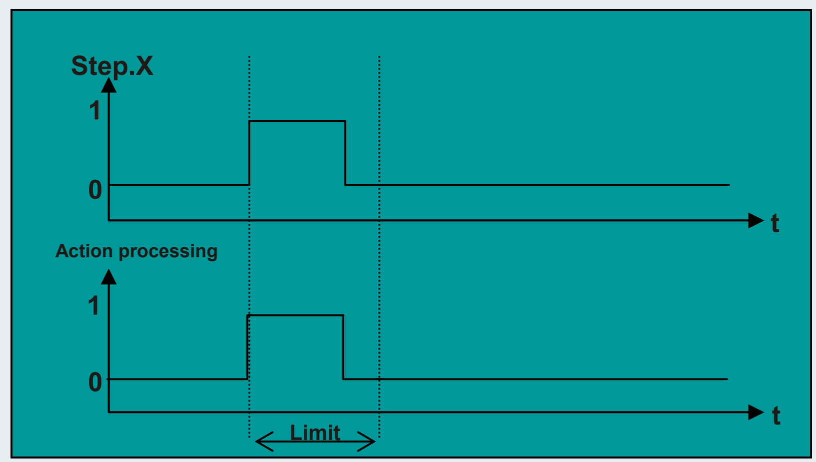
Controls the action processing after activating a step

L: LIMITED



Combination in FBD

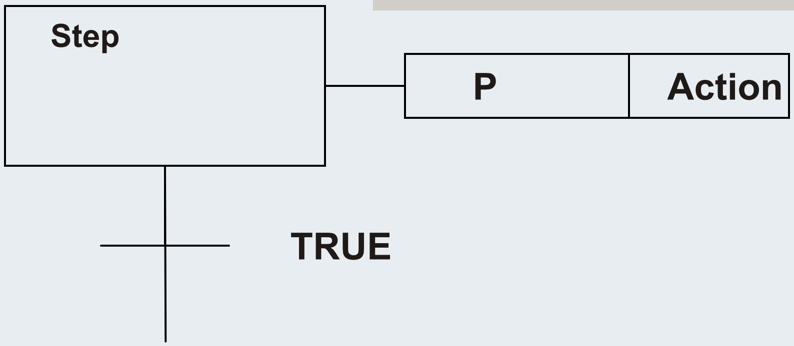


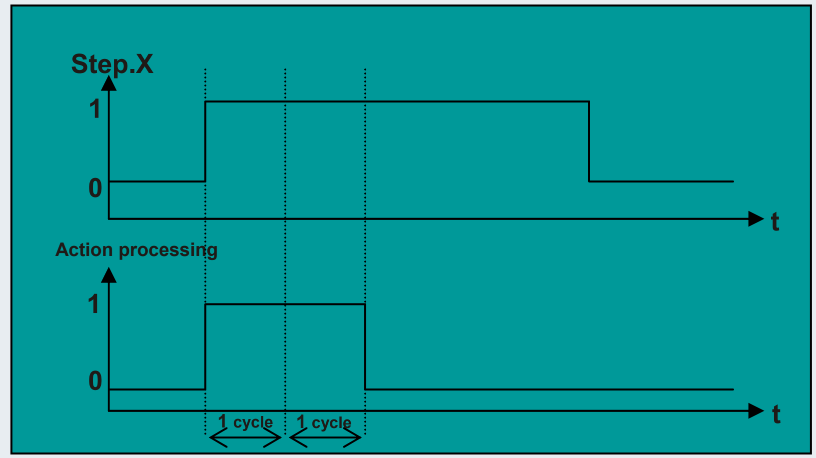


Qualifier

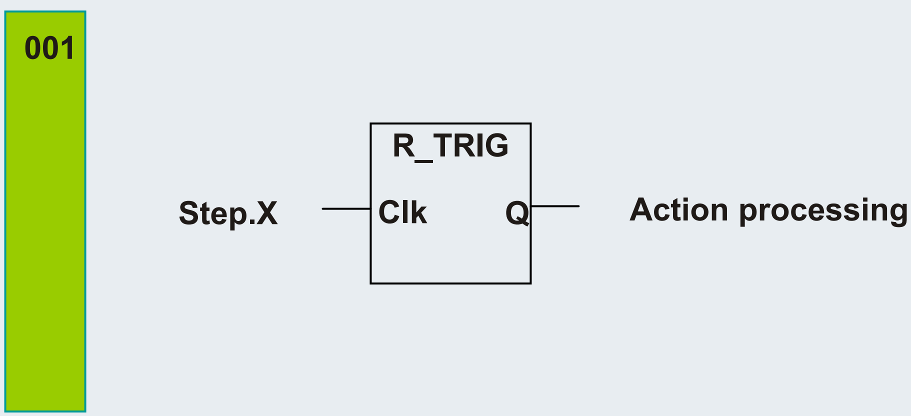
Controls the action processing after activating a step

P: PULSE





Combination in FBD



ATTENTION: A SECOND FLOW PROCESSES!

Qualifier, Combinations

- SD: Stored and delayed
- DS: Delayed and stored
- SL: Stored and time limeted

Ablaufsprache Stepdiagnostics

```
VAR
    SFCEnableLimit:                BOOL;
    (*enable monitoring timelimit *)

    SFCInit:                       BOOL;
    (*FORCE statemachine to init step
    IMPORTANT : DURING THIS VARIABLE IS TRUE THE
    INSTRUCTION IN THE STATEMACHINE ARE NOT
    EXECUTED*)
```

Sequential Function Chart step diagnosis

SFCReset: BOOL;

(*This variable, of type BOOL, behaves similarly to SFCInit. Unlike the latter, however, further processing takes place after the initialization of the Init step. Thus for example the SFCReset flag could be re-set to FALSE in the Init step.*)

Sequential Function Chart step diagnosis

SFCQuitError: BOOL;

(*<u>*Execution of the SFC diagram is stopped for as long as this boolean variable has the value TRUE*</u> whereby a possible timeout in the variable SFCError is reset.

All previous times in the active steps are reset when the variable again assumes the value FALSE.*)

SFCPause: BOOL;

(*<u>*Execution of the SFC diagram is stopped for as long as this boolean variable has the value TRUE.*)*</u>

SFCTrans: BOOL;

(*This boolean variable takes on the value TRUE when a transition is actuated. .*)

Sequential Function Chart step diagnosis

SFCError: BOOL;

(*This Boolean variable is TRUE when a timeout has occurred in a SFC diagram. If another timeout occurs in a program after the first one, it will not be registered unless the variable SFCError is reset first. *)

SFCErrorStep: STRING;

(*This variable is of the type STRING. If SFCError registers a timeout, in this variable is stored the name of the step which has caused the timeout. *)

SFCErrorPOU: STRING;

(*This variable of the type STRING contains the name of the block in which a timeout has occurred. *)

Sequential Function Chart step diagnosis

SFCCurrentStep: : STRING;

(*This variable is of the type STRING. The name of the step is stored in this variable which is active, independently of the time monitoring. In the case of simultaneous sequences the step is stored in the branch on the outer right.
No further timeout will be registered if a timeout occurs and the variable SFCError is not reset again.*)

Sequential Function Chart step diagnosis

SFCErrorAnalyzation: STRING;

(*This variable, of type STRING, provides the transition expression as well as every variable in an assembled expression which gives a FALSE result for the transition and thus produces a timeout in the preceding step. A requirement for this is declaration of the SFCError flag, which registers the timeout. SFCErrorAnalyzation refers back to a function called AppedErrorString in the TcSystem.Lib library. The output string separates multiple components with the symbol “|”. *)

SFCTip: BOOL;

SFCTipMode: BOOL;

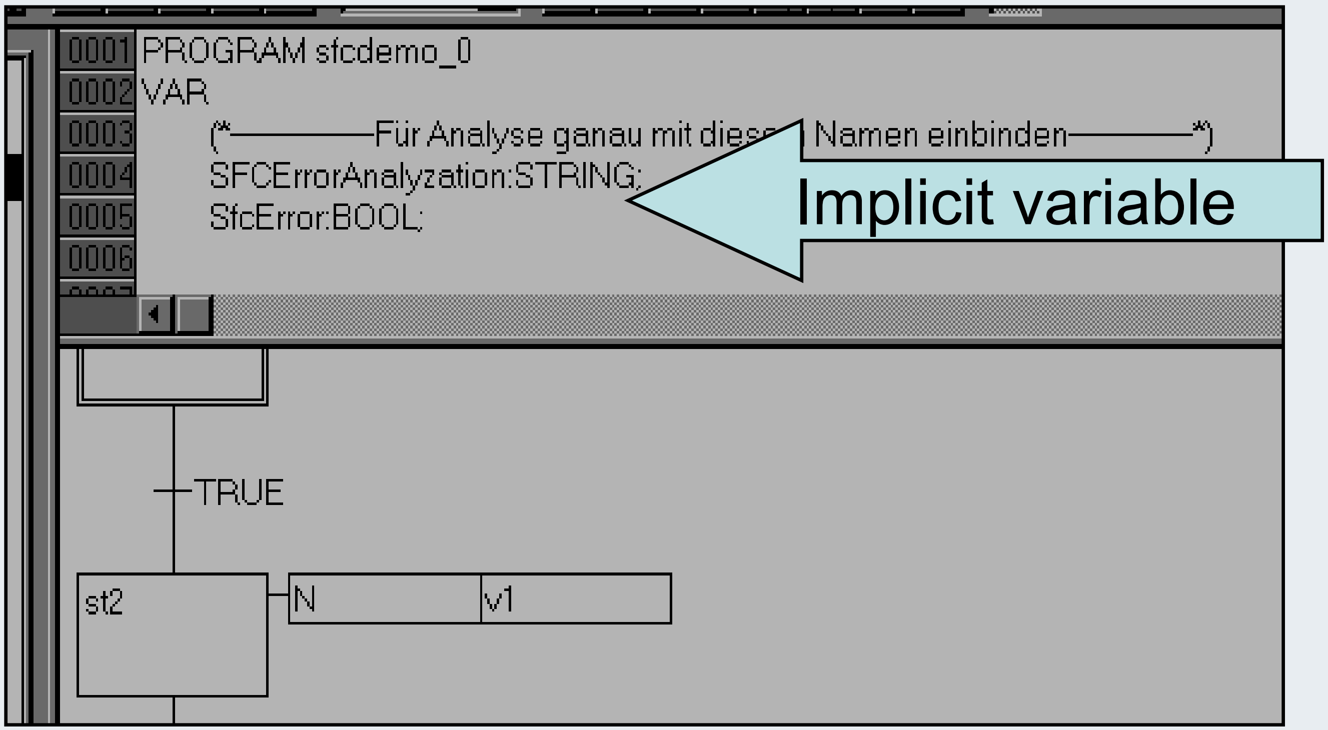
(*This variables of type BOOL allow inching mode of the SFC. When this is switched on by SFCTipMode=TRUE, it is only possible to skip to the next step if SFCTip is set to TRUE. As long as SFCTipMode is set to FALSE, it is possible to skip even over transitions.*)

END_VAR

Ablaufsprache

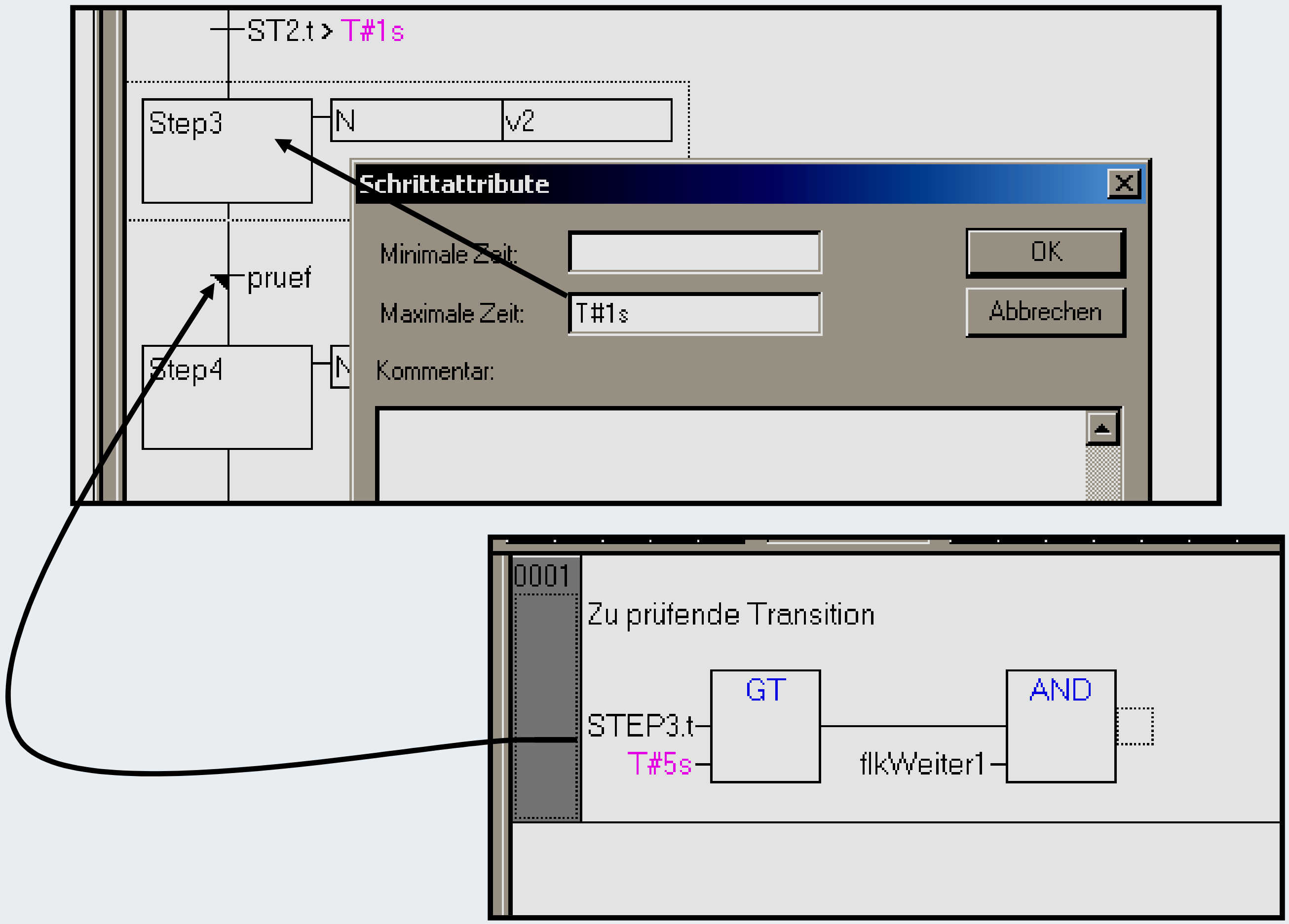
```
        SFCTip:              BOOL;
        SFCTipMode:          BOOL;
        (*run the statemachine in a manual mode*)
END_VAR
```

Sequential Function Chart process diagnosis

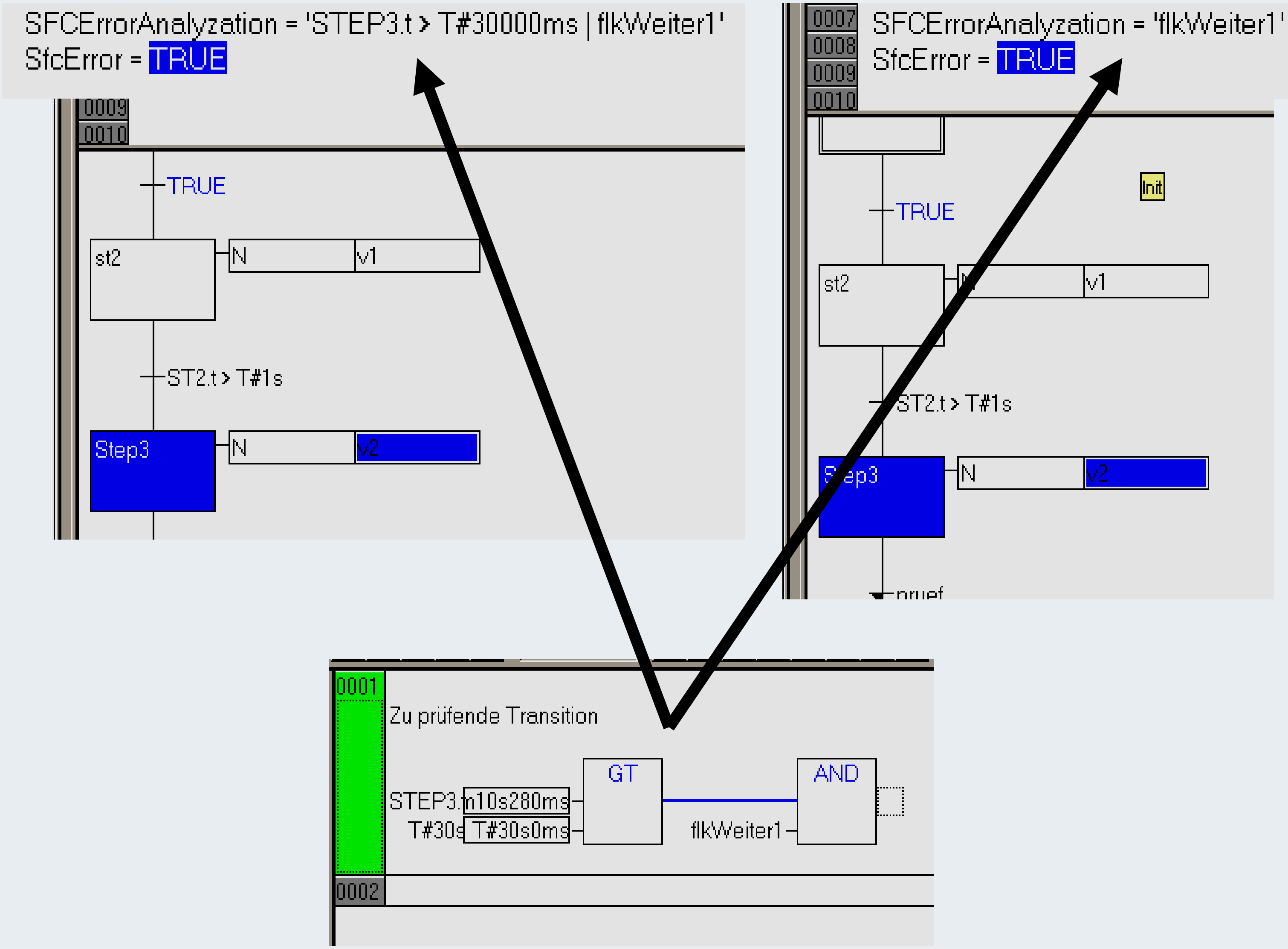


Sequential Function Chart process diagnosis

- set step attributes for the step to be observed

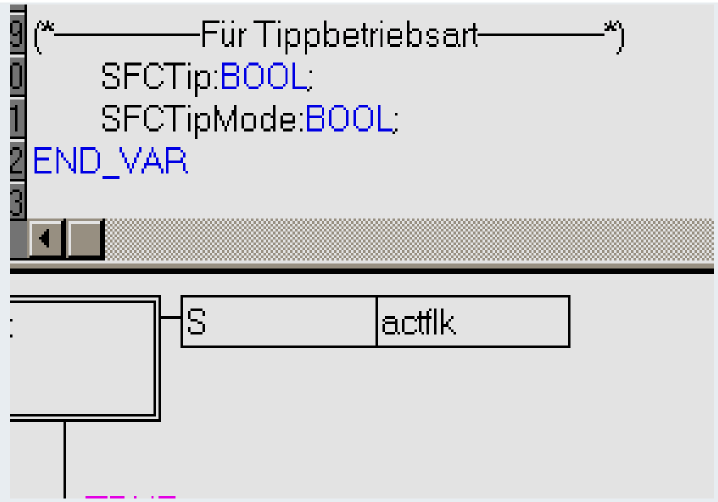


Online (and per ADS) can be requested



Sequential Function Chart Tipmode

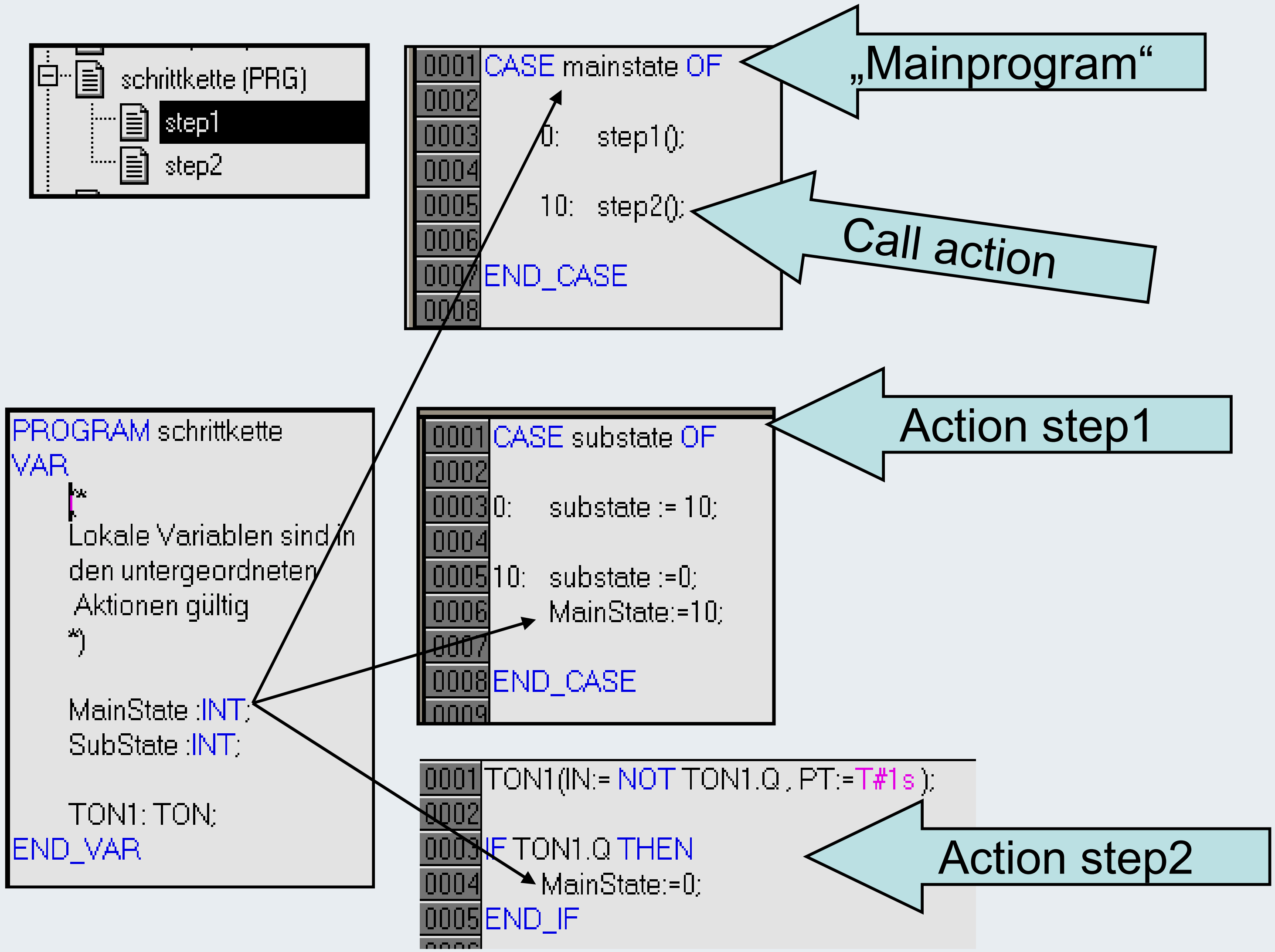
- insert implicit variable:



- effect:

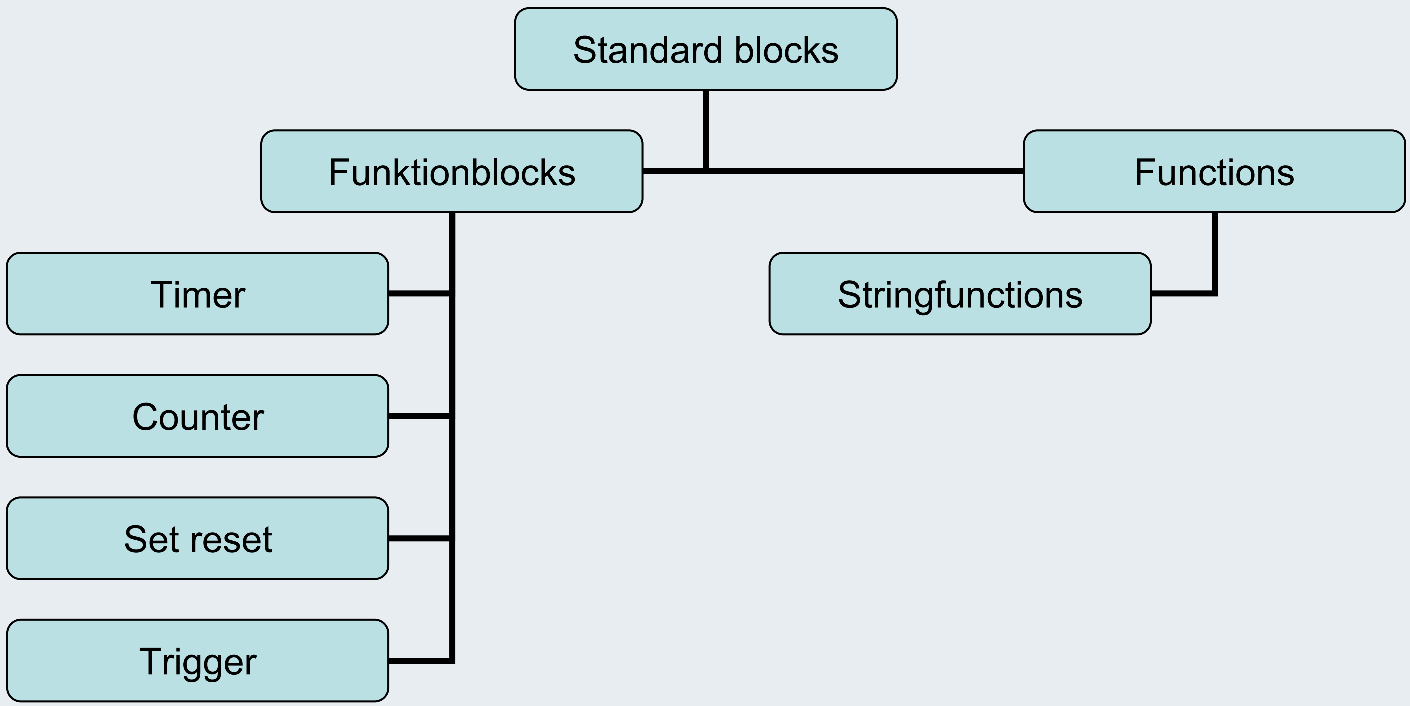
SFCTipMode	SFCTip	Transition	effect
TRUE	FALSE	TRUE	Process stays in the current step
TRUE	TRUE	TRUE	Change to next step
TRUE	TRUE	FALSE	Change to next step
FALSE	TRUE	FALSE	Process stays in the current step
FALSE	FALSE	TRUE	Change to next step

Actions also in other IEC languages possible! (POU type : PRG, FB)

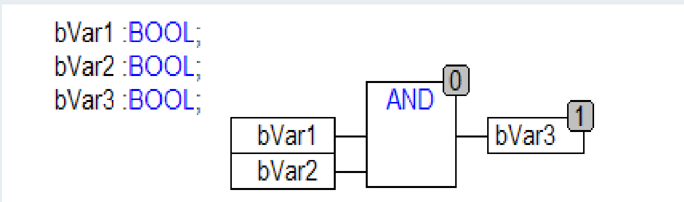
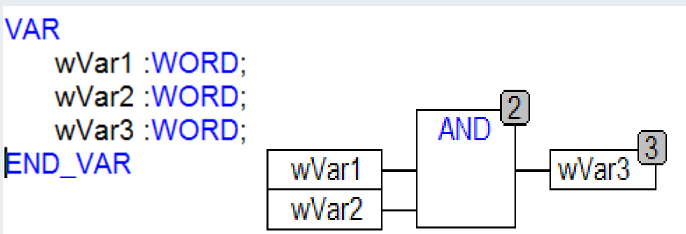
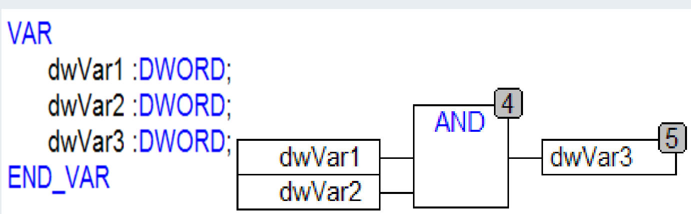


Standard IEC operators FB‘s

Standard.LIB



Logical operators

FUP / CFC	ST	Notes
	bVar3 := bVar1 AND bVar2 ;	**BOOL AND**
	wVar3 := wVar1 AND wVar2 ;	**WORD AND**
	dwVar3 := dwVar1 AND dwVar2 ;	**DWORD AND**

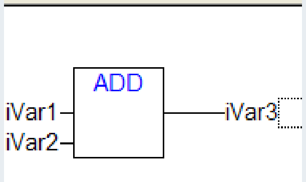
Numerical operators

FUP / CFC

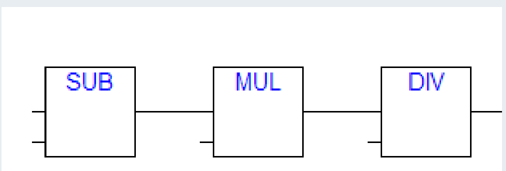
ST

notes

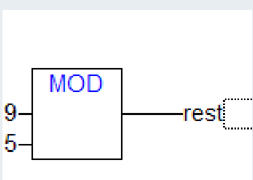
```
VAR
    iVar1 :INT;
    iVar2 :INT;
    iVar3 :INT;
END_VAR
```




```
iVar3 := iVar1 + iVar2 ;
```



- * /




```
rest := 9 MOD 5;
```

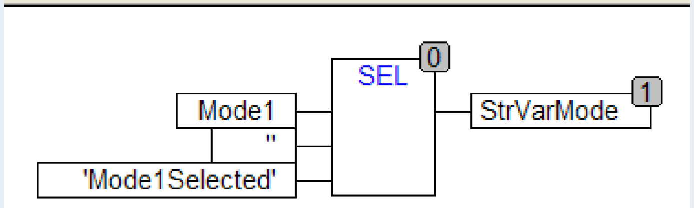
Rest of a division (4)

Selectors, SEL

Operator

FUP / CFC

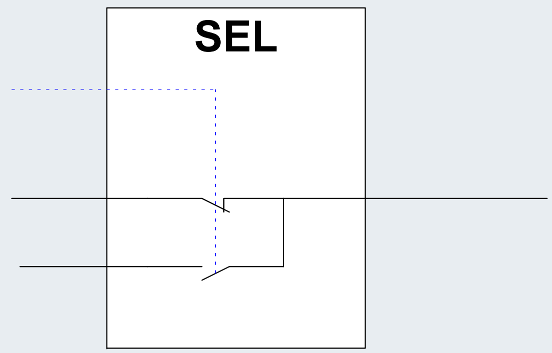
```
PROGRAM MAIN
VAR
	Mode1 :BOOL;
	StrVarMode :STRING;
END_VAR
```



ST

StrVarMode := SEL(Mode1, ‘‘, ,Mode1Selected);

notes



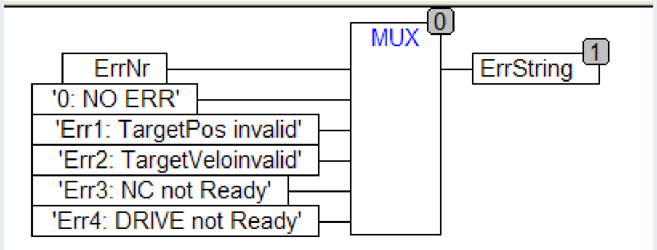
Beispiel:

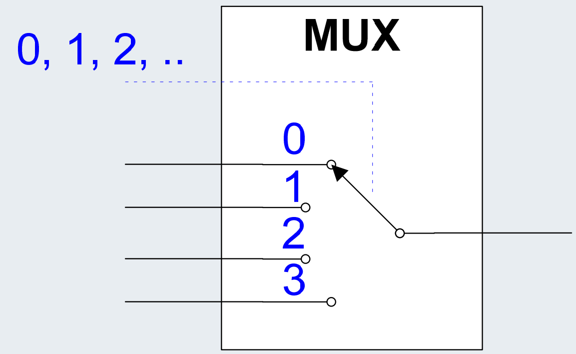
Mode1 = TRUE then StrVarmode is ‘Mode1Selected ‘ othewise empty

Multiplexer, MUX

FUP /CFC

```
0001 PROGRAM MAIN
0002 VAR
0003     ErrNr     :INT;
0004     ErrString :STRING;
0005
0006 END_VAR
```





ST:

```
ErrString := MUX(ErrNr ,
                 '0: NO ERR',
                 'Err1: TargetPos invalid',
                 'Err2: TargetVeloinvalid',
                 'Err3: NC not Ready',
                 'Err4: DRIVE not Ready');
```

Online:

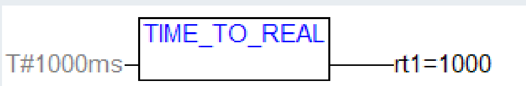
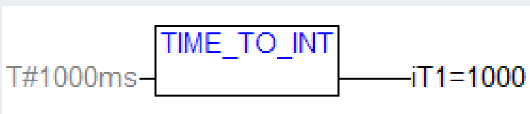
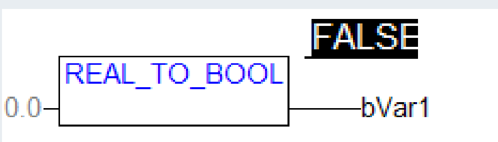
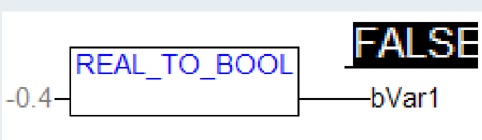
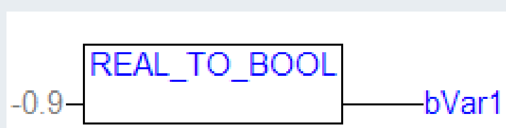
ErrNr = 32767
ErrString = 'Err4: DRIVE not Ready'

ErrNr = 0
ErrString = '0: NO ERR'

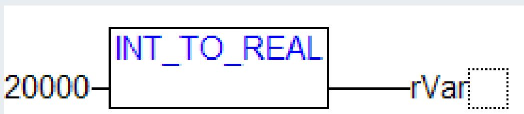
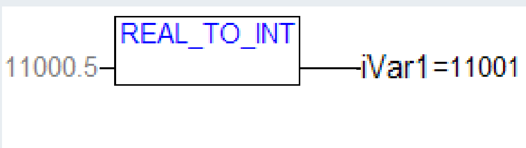
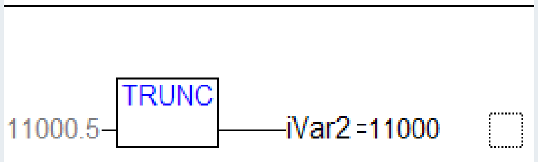
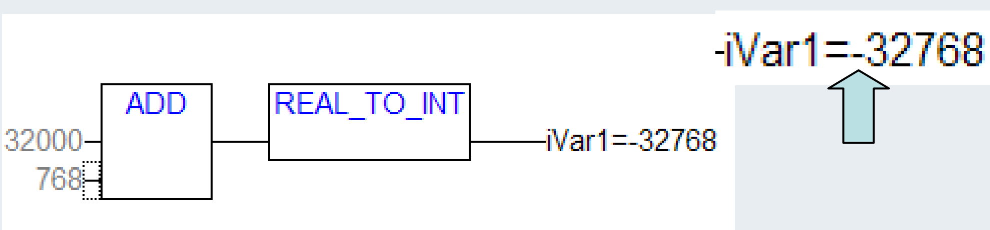
ErrNr = 1
ErrString = 'Err1: TargetPos invalid'

ErrNr = 2
ErrString = 'Err2: TargetVeloinvalid'

Conversions

Operator	ST	notes
	`rT1 := TIME_TO_REAL(T#1s);`	**Timevalues in ms**
	`iT1 := TIME_TO_INT(T#1s);`	**iT1:INT**
	`bVar1:= REAL_TO_BOOL(0.0);`	**bVar1:BOOL**
		**0.4 -> FALSE, >=0.5 ->TRUE**
		

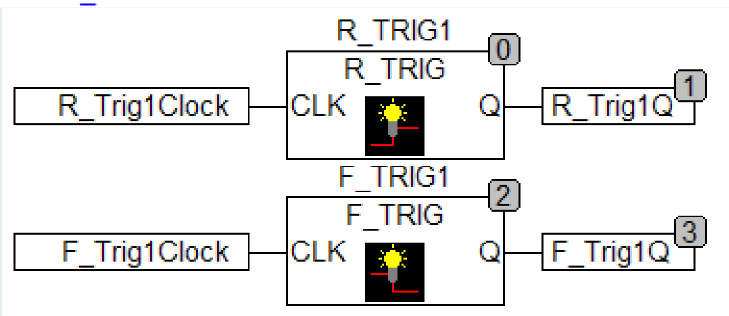
Conversions

FUP / CFC	ST	Bemerkungen
	rVar:= INT_TO_REAL(20000);	**Result 20000.0**
	iVar1:= REAL_TO_INT(11000.5);	
	iVar2 := TRUNC(11000.5);	**Conversion without rounding**
		**Overun!**

Trigger R_TRIG F_TRIG

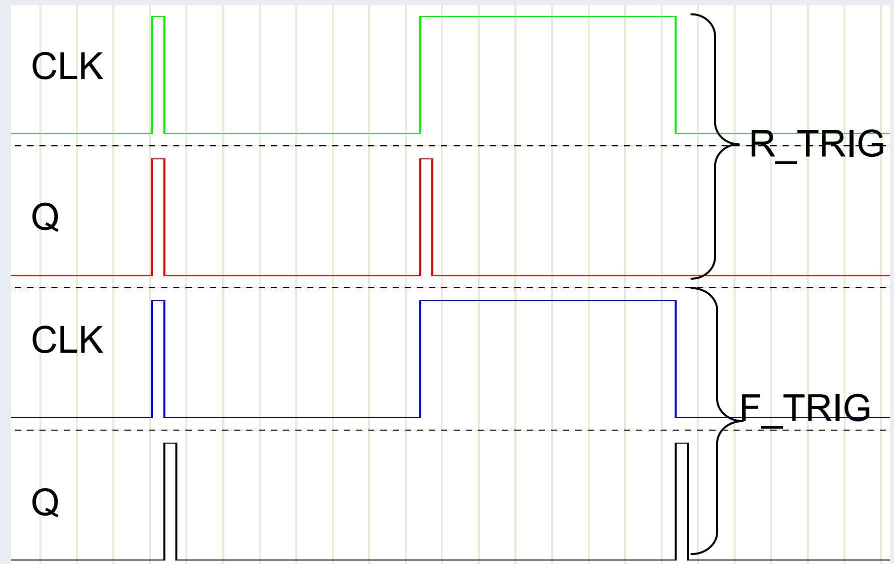
```
VAR
    R_TRIG1:          R_TRIG;     F_TRIG1: F_TRIG;
    R_Trig1Clock :    BOOL;       F_Trig1Clock: BOOL;
    R_Trig1Q :        BOOL;       F_Trig1Q: BOOL;
END_VAR
```

FUP / CFC



ST

```
R_TRIG1(CLK:=R_Trig1Clock , Q=>R_Trig1Q );
F_TRIG1(CLK:=F_Trig1Clock , Q=>F_Trig1Q );
```

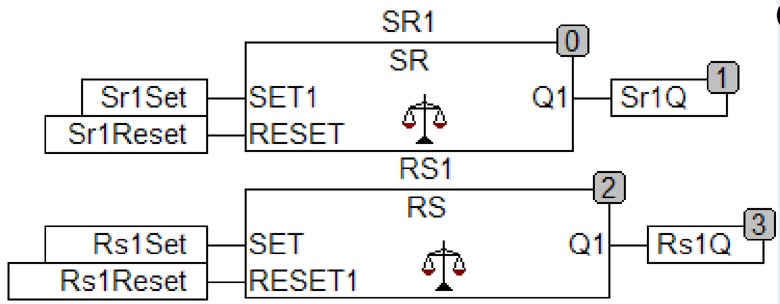


CLK	**BOOL**	**Triggerinput**	**R_TRIG: Rising Edge,**
Q	**BOOL**	**Output 1 PLC cy´cle**	**F_TRIG: Falling Edge**

Set/Reset SR RS

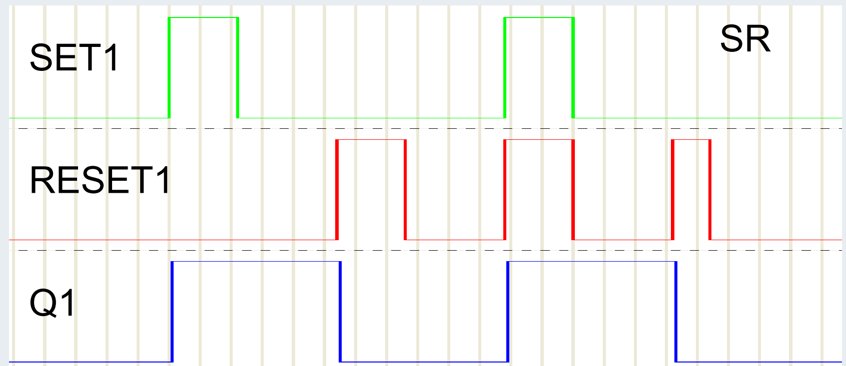
```
VAR
    SR1: SR;                RS1: RS;
    Sr1Set: BOOL;           Rs1Set: BOOL;
    Sr1Reset: BOOL;         Rs1Reset: BOOL;
    Sr1Q: BOOL;             Rs1Q: BOOL;
END_VAR
```

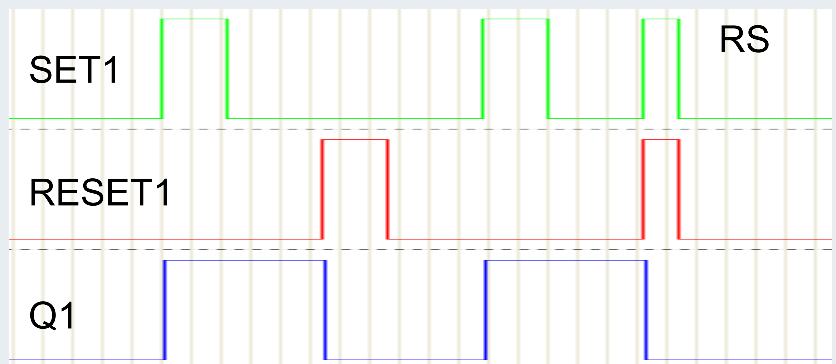
FUP / CFC



ST

```
SR1(SET1:=Sr1Set  , RESET:=Sr1Set  , Q1=>Sr1Q  );
RS1(SET:=Rs1Set  , RESET1:=Rs1Reset , Q1=>Rs1Q );
```





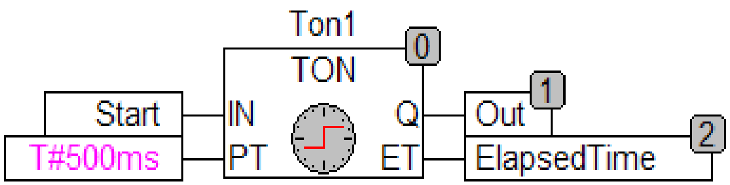
SET1	**BOOL**
RESET1	**BOOL**
Q1	**BOOL**

SR: prior set

RS: prior reset

On delay timer TON

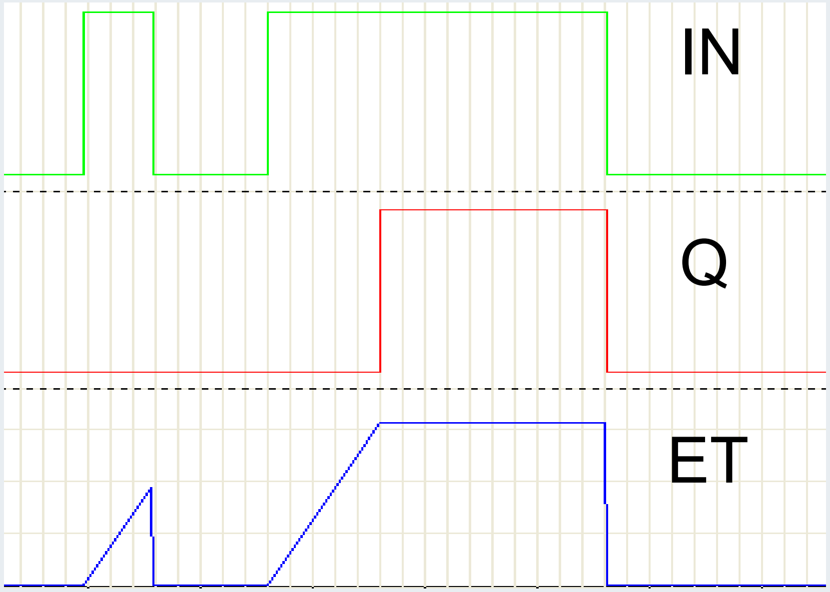
```
VAR
      Ton1: TON;
      Start: BOOL;
      Out: BOOL;
      ElapsedTime: TIME;
END_VAR
```



FUP / CFC

```
Ton1(    IN:=Start ,
         PT:=T#500ms ,
         Q=>Out ,
         ET=>ElapsedTime  );
```

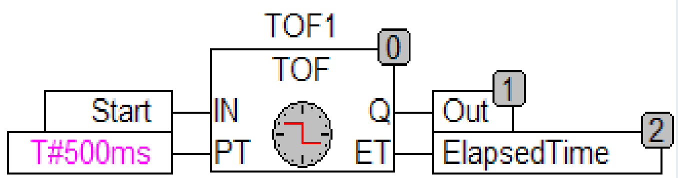
ST



IN	**BOOL**	**Start**
PT	**TIME**	**Preset Time**
Q	**BOOL**	**Output**
ET	**TIME**	**Elapsed Time,**

Off delay Timer TOF

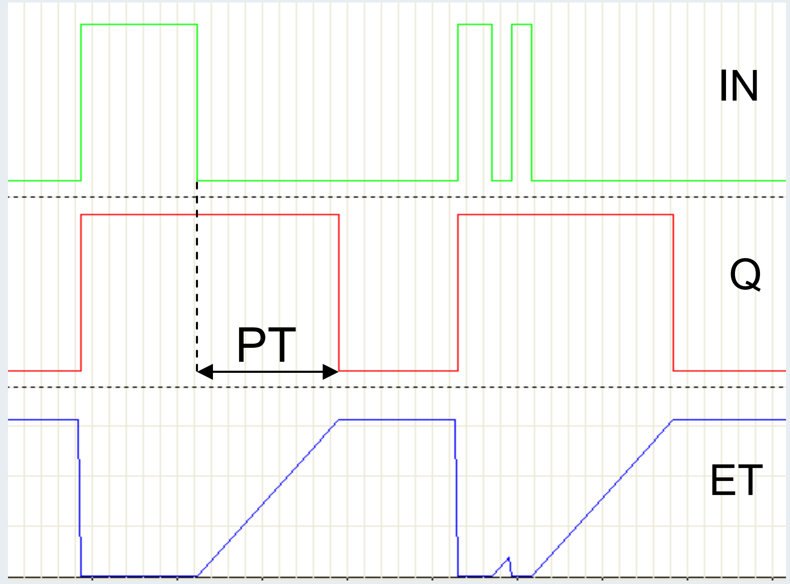
```
VAR
    TOF1: TOF;
    Start: BOOL;
    Out: BOOL;
    ElapsedTime: TIME;
END_VAR
```



FUP / CFC

```
TOF1(   IN:=Start ,
        PT:=T#500ms ,
        Q=>Out ,
        ET=>ElapsedTime );
```

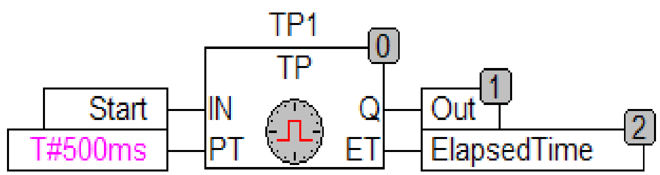
ST



IN	**BOOL**	**Start**
PT	**TIME**	**Preset Time**
Q	**BOOL**	**Output**
ET	**TIME**	**Elapsed Time**

Pulstimer TP

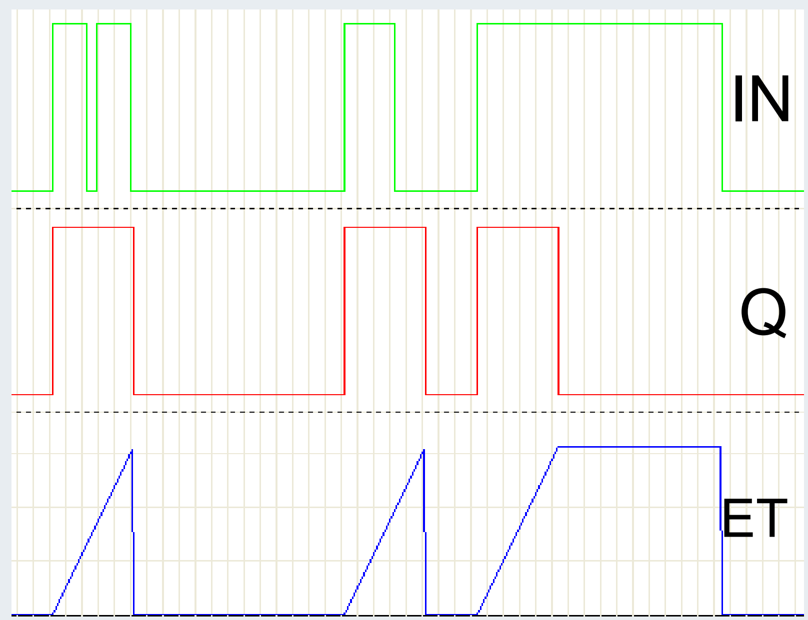
```
VAR
    TP1: TP;
    Start: BOOL;
    Out: BOOL;
    ElapsedTime: TIME;
END_VAR
```



FUP / CFC

```
Tp1(    IN:=Start ,
        PT:=T#500ms ,
        Q=>Out ,
        ET=>ElapsedTime );
```

ST

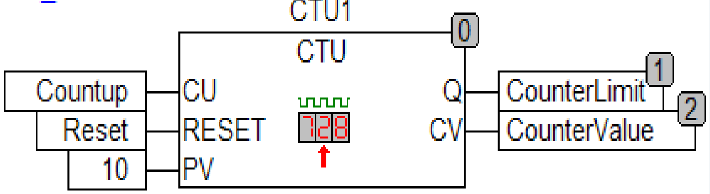


IN	**BOOL**	**Start**
PT	**TIME**	**Preset Time**
Q	**BOOL**	**Output**
ET	**TIME**	**Elasped Time**

Upcounter CTU

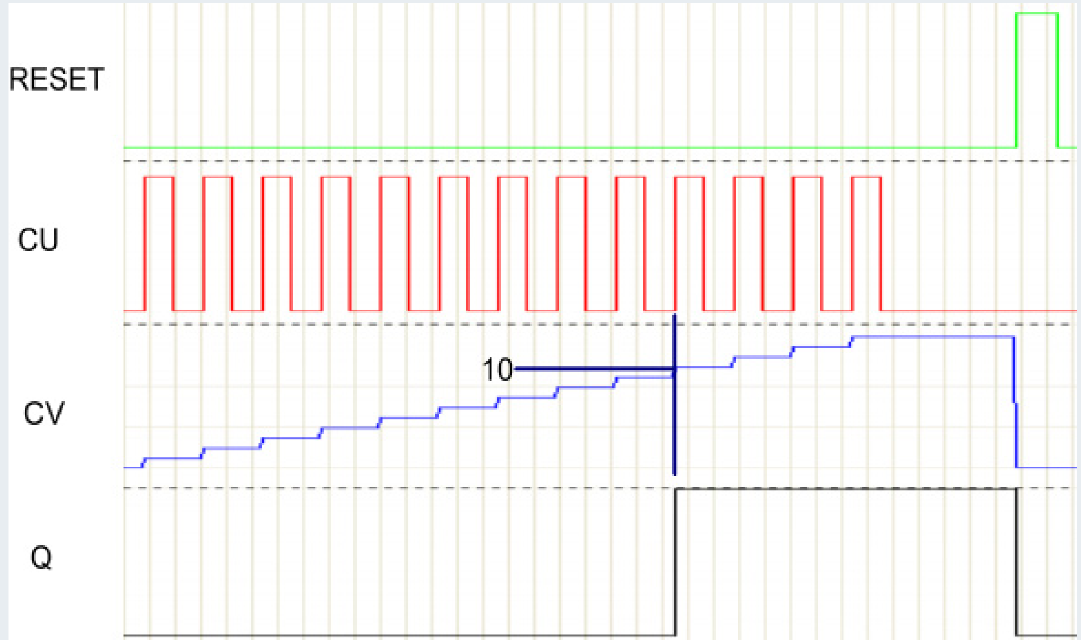
```
VAR
    CTU1: CTU;
    Reset :BOOL;
    Countup :BOOL;
    CounterValue :UINT;
    CounterLimit :BOOL;
END_VAR
```

FUP / CFC



ST

```
CTU1(
    CU:=       Countup ,
    RESET:=    Reset ,
    PV:=       10,
    Q=>        CounterLimit ,
    CV=>       CounterValue);
```

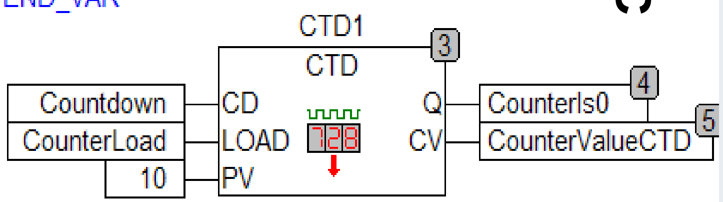


CU	**BOOL**	**Count Up**
RESET	**BOOL**	**reset**
PV	**UINT**	**Preset Value**
Q	**BOOL**	**Counter is „PV“**
CV	**WORD**	**Countervalue**

Downcounter CTD

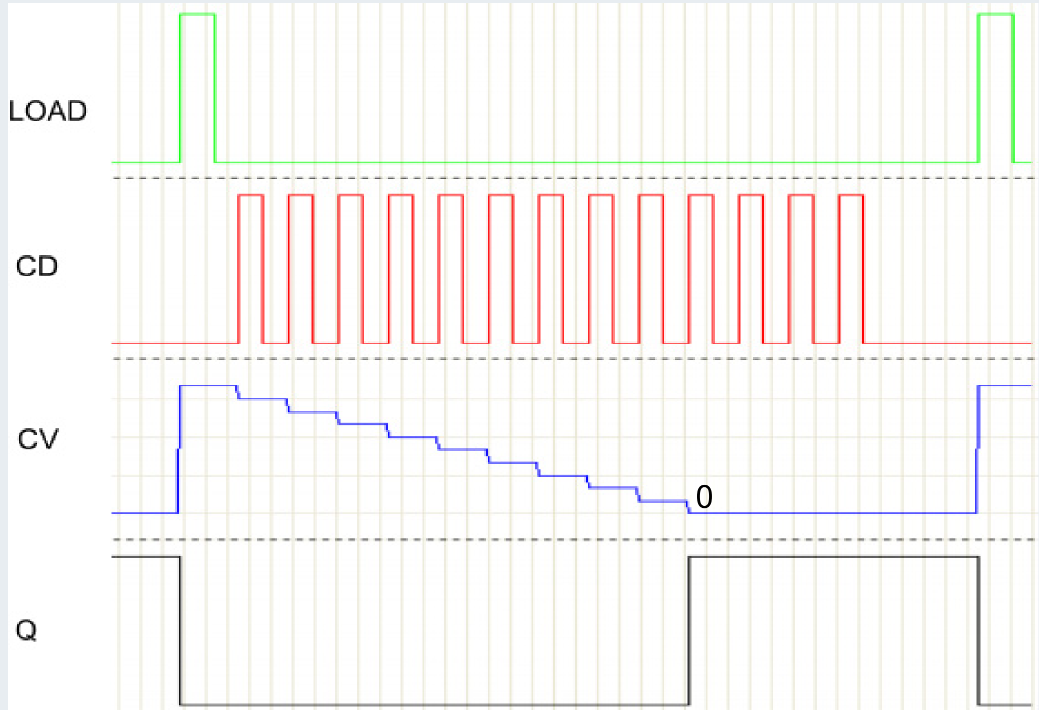
FUP / CFC

```
VAR
    CTD1 :CTD;
    CounterLoad :BOOL;
    Countdown :BOOL;
    CounterValueCTD :UINT;
    CounterIs0 :BOOL;
END_VAR
```



ST

```
CTD1(
    CD:=      Countdown ,
    LOAD:=    CounterLoad ,
    PV:=      10,
    Q=>       CounterIs0 ,
    CV=>      CounterValueCTD);
```

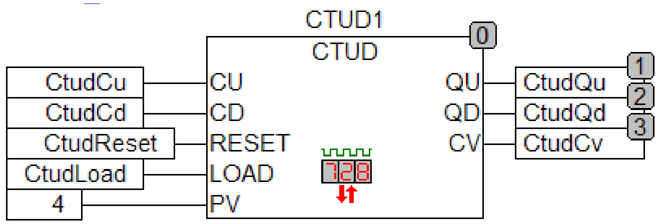


CD	**BOOL**	**Count Down**	**.**
LOAD	**BOOL**	**Set counter to PV**	
PV	**UINT**	**Preset Value**	
Q	**BOOL**	**Counter is 0**	
CV	**UINT**	**Countervalue**	

Up and Down counters CTUD

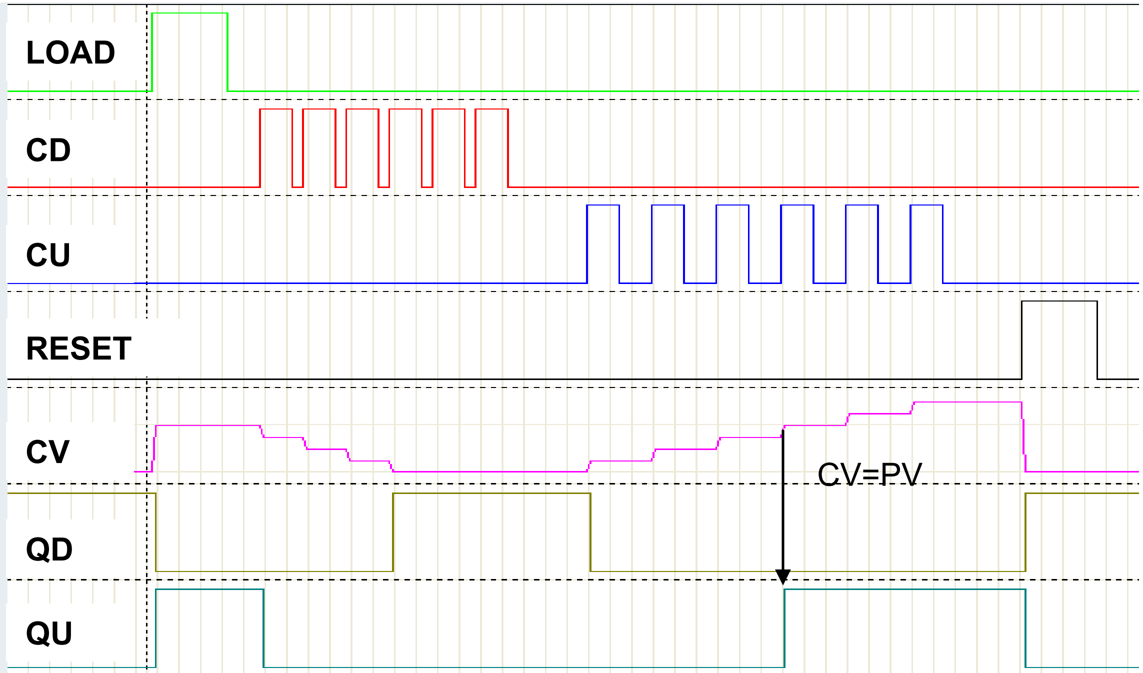
FUP / CFC

```
VAR
    CTUD1: CTUD;        CtudQu:BOOL;
    CtudCu :BOOL;       CtudQd:BOOL;
    CtudCd: BOOL;       CtudCv:UINT;
    CtudReset:BOOL;     CtudLoad:BOOL;
    CtuDPv:UINT;
END_VAR
```

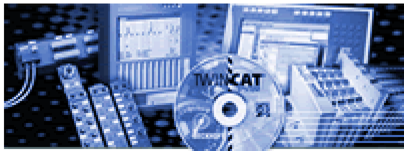


ST

```
CTUD1(
    CU:= CtudCu ,        CD:= CtudCd , RESET:=CtudReset
    LOAD:= CtudLoad ,    PV:= CtuDPv ,
    QU=>   CtudQu ,      QD=>CtudQd ,CV=>CtudCv );
```



CU	**BOOL**	**Count UP+**	**CV**	**UINT**	**Countervalue**
CD	**BOOL**	**Count DOWN-**			
RESET	**BOOL**	**Reset**			
LOAD	**BOOL**	**Set counter to PV**			
PV	**UINT**	**Preset Value**			
QU	**BOOL**	**Counter is „PV“**			
QD	**BOOL**	**Counter is 0**			



TwinCAT

The Windows Control and Automation Technology

TwinCAT ADS

Set values profiles

Definition

Excursion: TwinCAT device concept
- a.) Identification of TwinCAT ADS devices
- b.) TwinCAT message router

ADS
- a.) Introduction
- b.) Client-server relationship
- c.) Access types
- d.) Overview of methods

ADS (General)

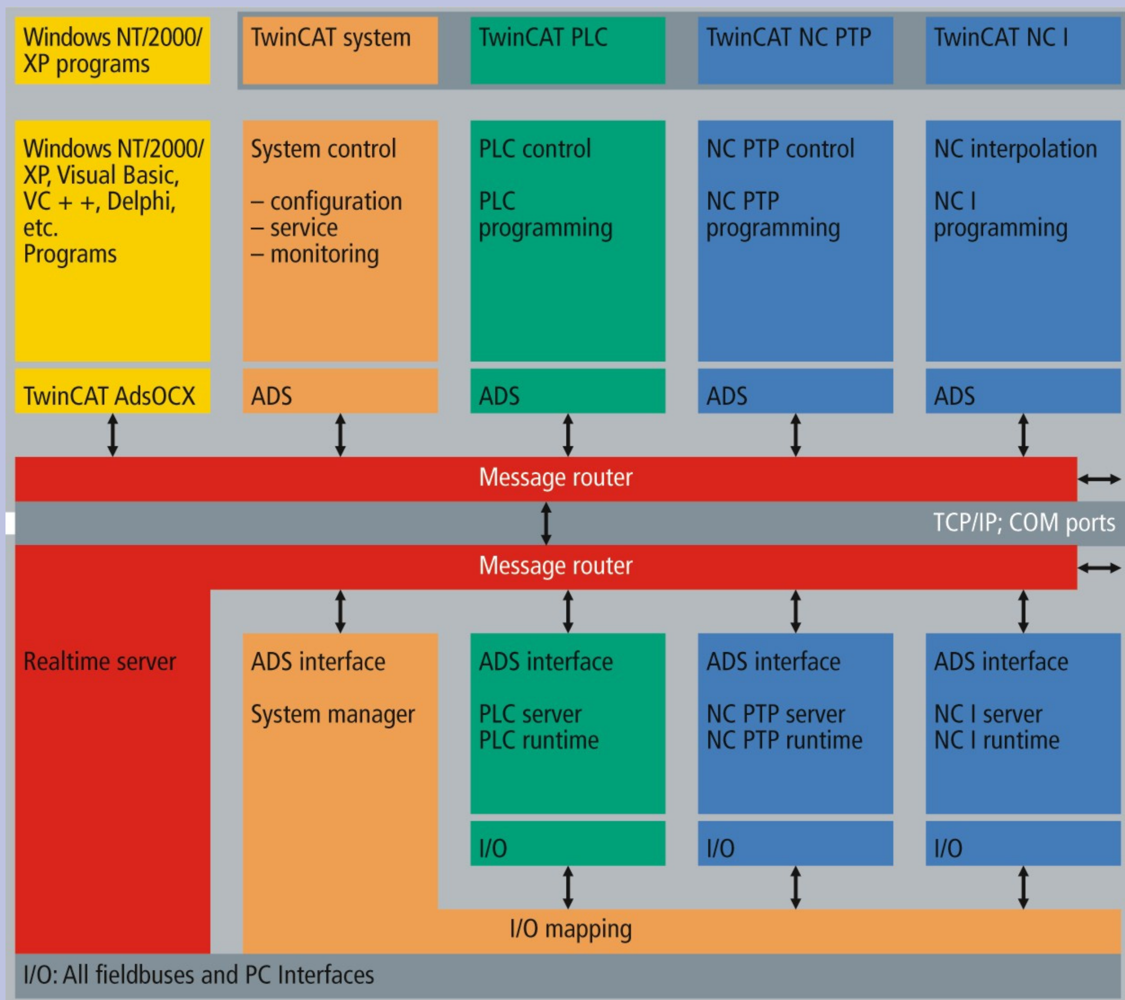
Definition

ADS = **A**utomation **D**evice **S**pecification

- modular ADS devices
  e.g. PLC (each run time system), NC,...

- message exchange by ADS via the message router

TwinCAT Device concept



ADS server unambiguously identifiable by: **AdsAmsServerPort**

Router unambiguously identifiable by: **AdsAmsServerNetID**

Identification of ADS devices

- Every (TwinCAT) PC in the network can be uniquely identified by means of a TCP/IP address such as "172.1.2.16".

- The AdsAmsServerNetID is an extension of the TCP/IP address, and identifies a message router, e.g. "172.1.2.16.1.1"

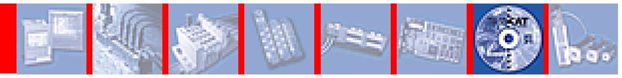
- The last two figures can be freely selected.

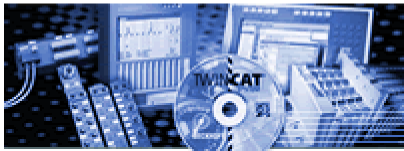
Identification of ADS devices

The ADS servers of a message router are clearly identified by a number (-> AdsAmsServerPort).

The following AdsAmsServerPort numbers are already assigned:

801,811,821,831	: TwinCAT PLC server, 1.-4. run-time system
500	: TwinCAT NC server
100	: TwinCAT logger





TwinCAT message router (I)

TwinCAT message router (I)

Example:
An ADS client sends an ADS message to an ADS server.

The TwinCAT message router (transport layer 4) carries out:

- acceptance of the request from the client

- forwarding the message to an router if appropriate

- provision of the messages to the ADS server

TwinCAT message router (II)

Existing message router:
- On every TwinCAT PC
- On every Embedded PC CX1000, CX1001
- On every bus controller BX1000
- On every bus controller (e.g. BC3100, BC8100, ..., BCxxxx)

Possible communication paths:
- Network (TCP/IP)
  A PLC run-time system sends data to another PLC on another TwinCAT PC in the network.
- Fieldbus (Lightbus / Profibus)
  A PLC run-time system sends data to another PLC on a bus controller in the fieldbus.

ADS - Interface

The ADS interface permits:

Standard:

- communication with other ADS devices
  Client requests data or services, server answers automatically

Special solutions:

- implementation of an ADS **server** (device)
  PLC : Extension of the existing ADS server functions of the PLC
  with the PLC Library function blocks “Indication Response”

Windows applications:TcSystem DLL
(detailed information about ADS communication necessary

Client-server relationship

1.) Confirmed services
2.) Unconfirmed services

ADS Client		**ADS Server**
Request	→	**Indication**
Confirmation	←	**Response**

Client – server relationship between 2 PLC devices

Example (PLC runtime 1) requests data from PLC (runtime2)

ADS-Client PLC I — **ADS-Server PLC II**

uses ADS FB — **Provides the data**

Request
(FB call)

```
ADSREAD
NETID : T_AmsNetId        BUSY : BOOL
PORT : T_AmsPort           ERR : BOOL
IDXGRP : UDINT           ERRID : UDINT
IDXOFFS : UDINT
LEN : UDINT
DESTADDR : DWORD
READ : BOOL
TMOUT : TIME
```

Confirmation
(Busy at FB)

Indication

```
obale_Variablen
VAR_GLOBAL
    AnalogWertPlc2ToPlc1 AT%MB1000 : INT;
END_VAR
```

Response

Access types (I)

1.) "By address"

Example:

Read / write a total of 100 bytes starting from byte offset 20 of the input/output process image in the PLC server or of the flags area. (Often used in process visualisation)

2.) "By name“

Example:

Read / write the PLC variable "temperature".

Access types, time flow (I)

Synchronous.

-> Client sends a request to server
-> Client waits until the result is existing
-> „Synchronous to the program line“

Access types, time flow (II)

Asynchronous

-> Client sends a query to the server
-> Client continues to work without waiting
-> Result of the server by callback

Notification

-> Client registers itself at the server
-> Server serves the client autonomously by callback
(until the client signs off from the server again)

ADS OCX Introduction

Definition OCX:

ActiveX-Control according to COM-(**C**omponent **O**bject **M**odel) Specification

The OCX contains:

- general ADS services
- extended ADS services, that simplify the process (e.g. synchronous communication)

ADS OCX Introduction methods (I)

Synchronous	AdsSyncReadBoolReq AdsSyncReadBoolVarReq AdsSyncReadIntegerReq AdsSyncReadIntegerVarReq AdsSyncReadLongReq AdsSyncReadLongVarReq	**AdsSyncxxyyReq -> per Adresse** **AdsSyncxxyyVarREQ ->per Name**
Asynchronous	AdsReadIntegerReq AdsReadLongReq AdsReadSingleReq AdsReadDoubleReq AdsReadStringReq	**Methods for request (= Req)**
	AdsReadIntegerConf AdsReadLongConf AdsReadSingleConf AdsReadDoubleConf AdsReadStringConf	**Events at data updating (= Conf)**

ADS OCX Introduction methods (II)

Notification	**ReadVarConnectExAdsReadConnectUpdateEx**	**Method for apply Notification**
	AdsReadConnectUpdateEx	**Event after data update**
	AdsDisconnectEx	**Method for sign off Notification**

Notification... Connect?.... EX?

Notification is a term from the communication.
Connect is a term of the ADS Ocx method.
EX: means Extended. It is the extended method of the old connect method. The difference is the easier handling.
(- > connectEx is recommended)

Notification Refresh

Server Cycle	(e.g. PLC send with every cycle)
Server on Change	(e.g.) PLC send only if variables changes
Client Cycle	Client (ADS Ocx) writes / reads data with own cycle.

ADS OCX methods overview

	Variable types	Access	Advantage	Disadvantage	Comment
Synchro-nous	All	By address By name	Easy handling in the Visual Basic Program	The user must de-termine the event himself. (Disadvantage when reading) If the answer pages are long, the VB program can slow down	Local communication, fast networks, communication initiated at need
Asynchro-nous	All except BOOL	Generally only by address. But (with more effort, i.e. using index group/index offset) also by name	The Visual Basic Program does not wait for an answer, but continues to operate	More administra-tive overhead in the Visual Basic program, since the answer requires reaction to an event.	Slow communication (networks) large data packets
Connect	All	By name By address	Updating, takes place on the server (PLC)	Unnecessarily high data traffic with many connect connections	Specify sensible refresh times, clear connections that are not needed

Specifying the ADS communication partner

Before communicating with an ADS device, the following properties of the communication partner must be specified <u>once</u>:

- **AdsAmsServerNetID** (e.g. "172.1.2.16.1.1") and
- **AdsAmsServerPort** (e.g. "801" PLC server, 1st run-time system)

A separate ADS-OCX should be used for each ADS communication partner:

e.g. "ADS_OCX1" for PLC server 1st run-time system
e.g. "ADS_OCX2" for NC server
e.g. "ADS_OCX3" for PLC server 2nd run-time system

For communication between one PC and several BC (9)000 Controllers one OCX and e.g. one additional I/O task can be used.
The IO task “collects”all data from the BCs and the ADS OCX access to this area.

ADS OCX Examples

ADS-OCX Examples:

ADS-OCX in Visual Basic project

I	Linking, simple data exchange
II	Examples: "Synchronous" and "Connect“
VBA I	VBA with Graphworks synchronous data exchange
VBA II	VBA with Graphworks data exchange with connect
VBA III	VAB with excel

TwinCAT.Ads.Dll Examples

TwinCAT.ADS.Dll Examples:

TwinCAT.ADS.Dll with **VisualStudio.**net and C#

Ads.Dll C# I	Linking, simple data exchange
Ads.Dll C# II	Examples: "Synchronous" and "Connect“

5.1.b) Example "Write a Variable Synchronously to the PLC"

The task:

PLC

In the PLC run-time 1
The local variable "iVbToPlc1" in the MAIN program is to be overwritten by the Visual Basic application.

Visual Basic

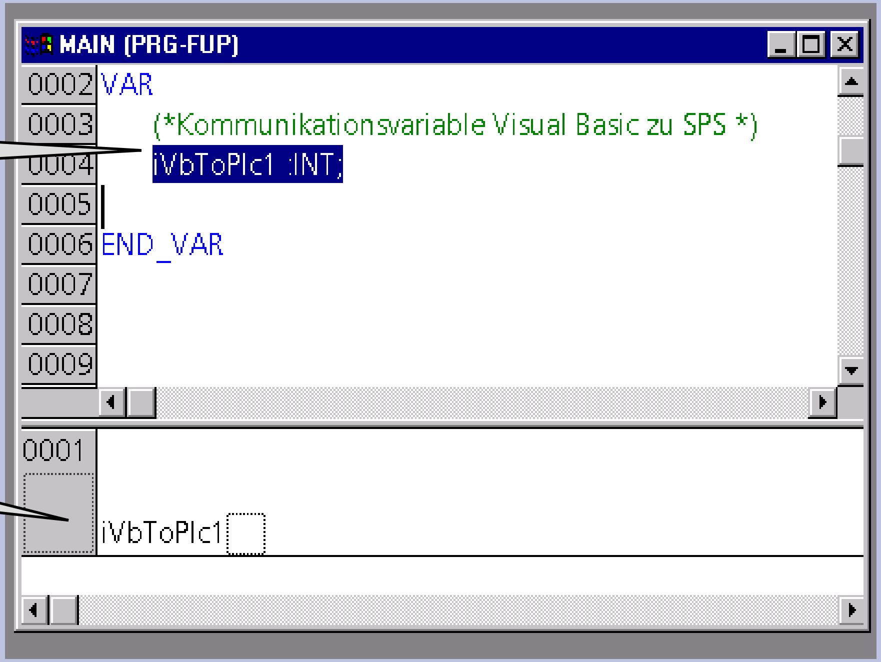
An (integer) value is to be read from a text field in the Visual Basic application. This value is converted into an integer and saved in the global variables of the PLC project. Name: iVbToPlc1
This global variable is written into the PLC with a command button.

Example 1

PLC:
Create a PLC project, create the main program and declare a local variable <u>iVbToPlc1</u>

Local var VbToPlc1

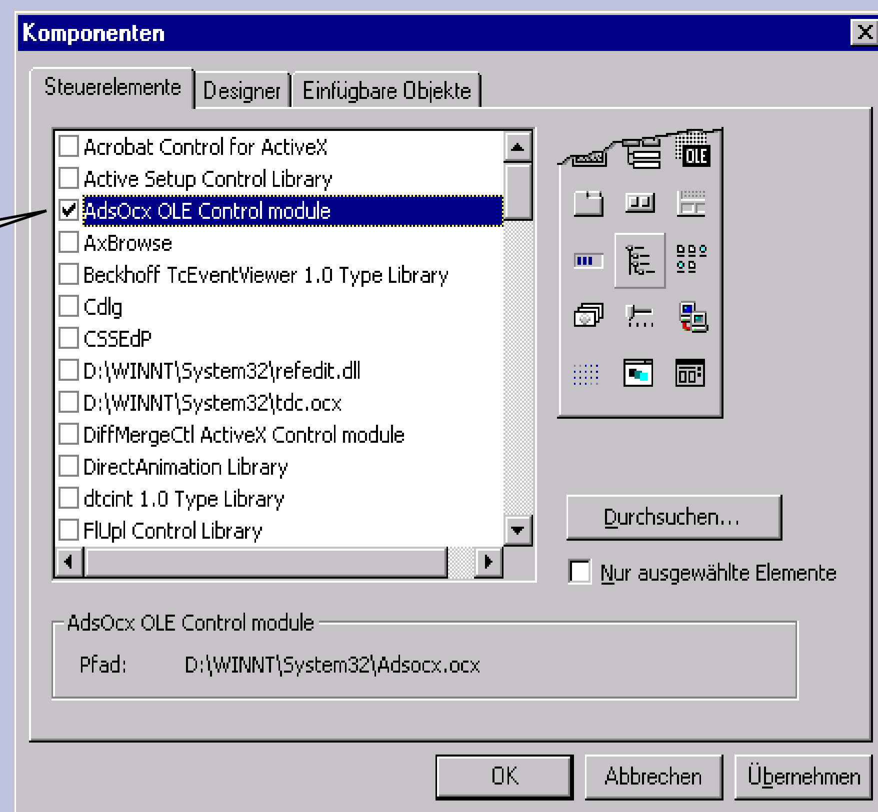
Only dummy instruction



Example 1

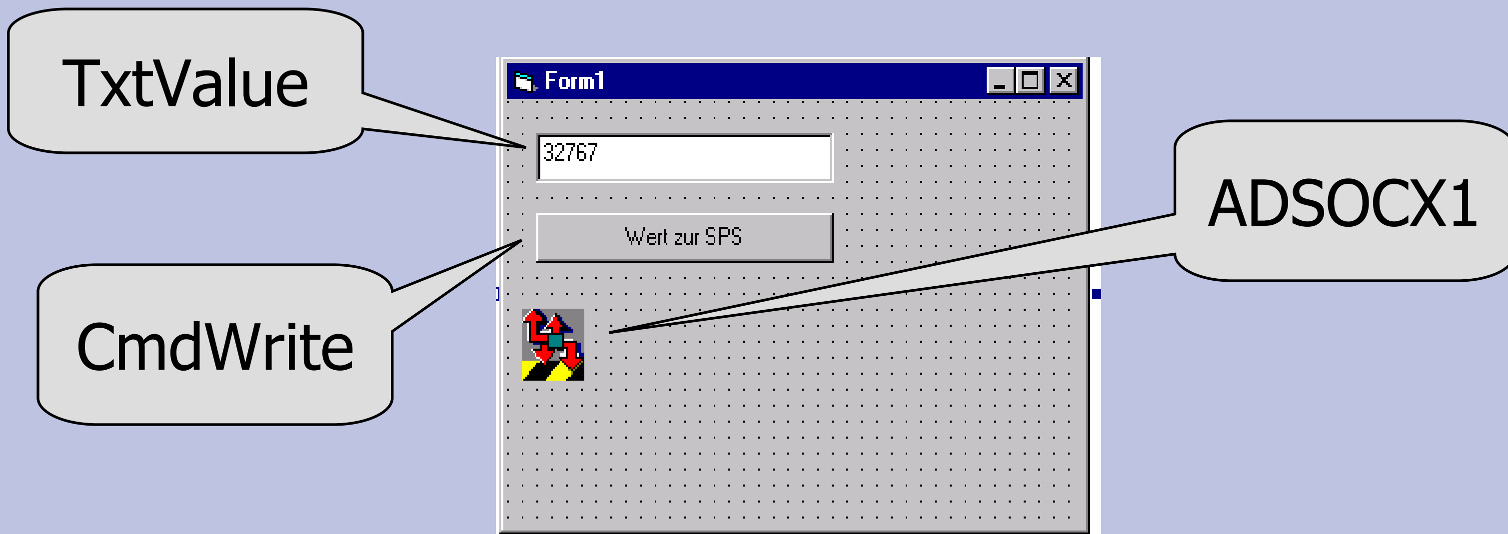
Visual Basic:
Create Visual Basic, (standard.exe), link ADSOCX

Project -> components



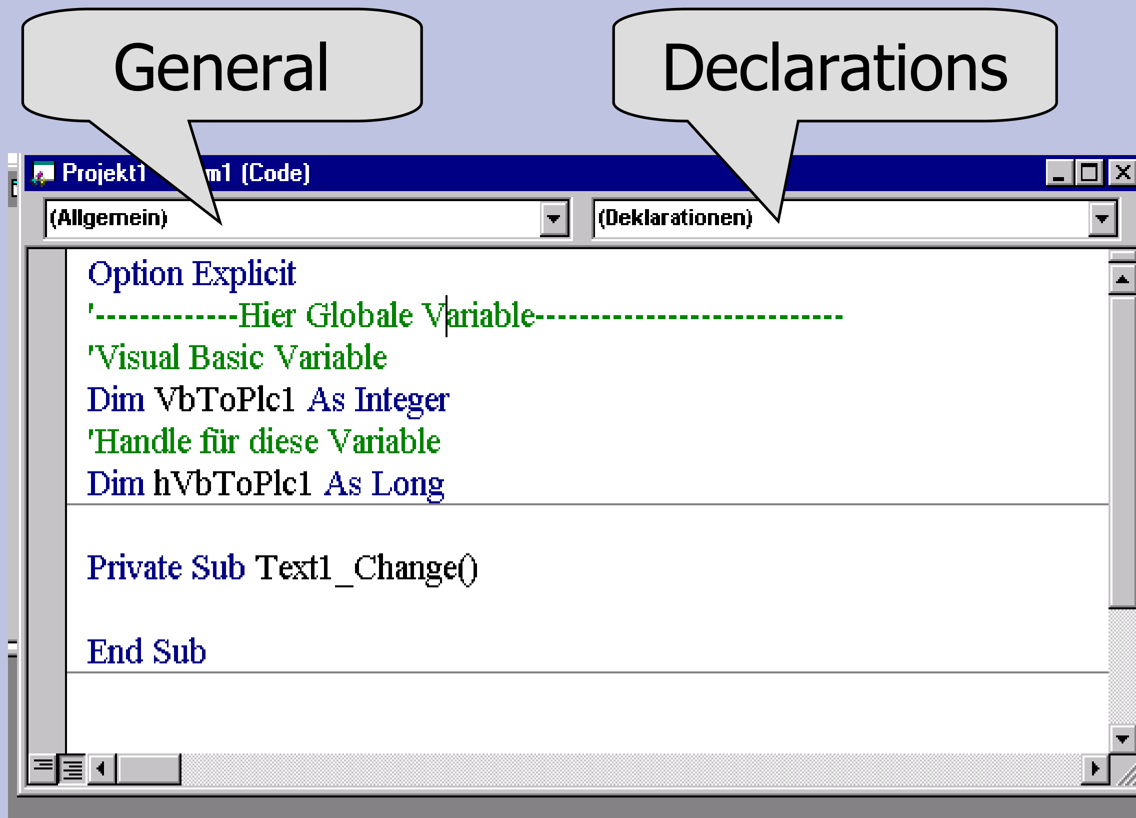
Example 1

Visual Basic:
Create form, place control elements



Example 1

Visual Basic:
Declare variable and handle for the variable



Example 1

Visual Basic:
In FORMLOAD event set PROPERTIES for ADSOCX1

Form Load Event
(double click on the form)

```
Form                                   Load
Private Sub Form_Load()

'---------Eigenschaften ADSOCX-----------------------------------------
'Festlegung lokal arbeiten
AdsOcx1.AdsAmsServerNetId = AdsOcx1.AdsAmsClientNetId
'Portnummer für Laufzeit 1 festlegen
AdsOcx1.AdsAmsServerPort = 801
'Fehlerhandlig aktivieren
AdsOcx1.EnableErrorHandling = True
```

Set properties
(work local)

Set properties
PLC Runtime should be 1
(=Port 801)

Set properties
Enable exception , if method call incorrect, VB stops.

Example I

```
'Fehlerhandlig aktivieren
AdsOcx1.EnableErrorHandling = True
```

At faulty completion of a message, a message dialog is generated automatically.
In the development environment of VB can be jumped to the program line along with the cause.
By using the VB instructions **On Error** Goto / Resume an own error handling can be programmed.
C#: **Try and Catch**

Example 1

Visual Basic:
Other possibility for error evaluation

Projekt1 - Form1 (Code)

Form | Load

```
'---------Aufruf mit Fehlerauswertung-------------------------
If AdsOcx1.AdsCreateVarHandle("MAIN.VbToPlc1", hVbToPlc1) <> 0 Then
    MsgBox ("Fehler Handle generieren")
End If
```

Method call with evaluation
(If an error occurs during execution
(of the method), the return value does not equal 0.
This can be used to trigger execution of an
individual error handling process
(here a message box).)

Example 1

Visual Basic:

The handle is provided by the PLC and is known to the Visual Basic program. This handle can now be used to demand the variable from the PLC. The "Click Event" of the command button is selected as the trigger (event) for this action.
First therefore read the text field, and assign this value to the variable.

Projekt1 - Form1 (Code)

cmdSchreiben | Click

```
End Sub

Private Sub cmdSchreiben_Click()

'Wert aus dem Textfeld lesen
 VbToPlc1 = Val(TxtWert.Text)
```

Example 1

Visual Basic:

The handle is provided by the PLC and is known to the Visual Basic program. This handle can now be used to demand the variable from the PLC. The "Click Event" of the command button is selected as the trigger (event) for this action.

As AdsOcx method is used

AdsOcx1.AdsSyncWriteIntegerVarReq, what is the meaning of this name?

Synchronous **writing**

(Synchronous is related to the processing of the Visual Basic program. Execution of the Visual Basic program only continues when execution of this method has been completed.)

A value of **type integer**

This value is to be written via the **name** variable.

Example 1

Visual Basic:

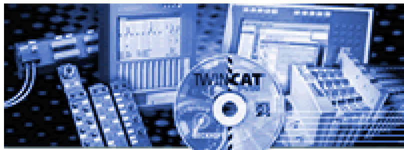
In the click event of the command button.

Projekt1 - Form1 (Code)

cmdSchreiben | Click

```
End Sub

Private Sub cmdSchreiben_Click()

'Wert aus dem Textfeld lesen
 VbToPlc1 = Val(TxtWert.Text)
```

This variable is to be written to the PLC.



Example 1

Visual Basic:

In the click event of the command button.

```
Projekt1 - Form1 (Code)
cmdSchreiben                Click

Private Sub cmdSchreiben_Click()

 Call AdsOcx1.AdsSyncWriteIntegerVarReq(
        AdsSyncWriteIntegerVarReq(hVar As Long, length As Long, pData As Integer) As Long
```

Pay attention to Intelli Sense from Visual Basic (Studio)

```
Projekt1 - Form1 (Code)
cmdSchreiben                Click

Private Sub cmdSchreiben_Click()

 Call AdsOcx1.AdsSyncWriteIntegerVarReq(hVbToPlc1, 2, VbToPlc1)

End Sub
```

The **VISUAL BASIC variable** from which the value to be written is to be taken

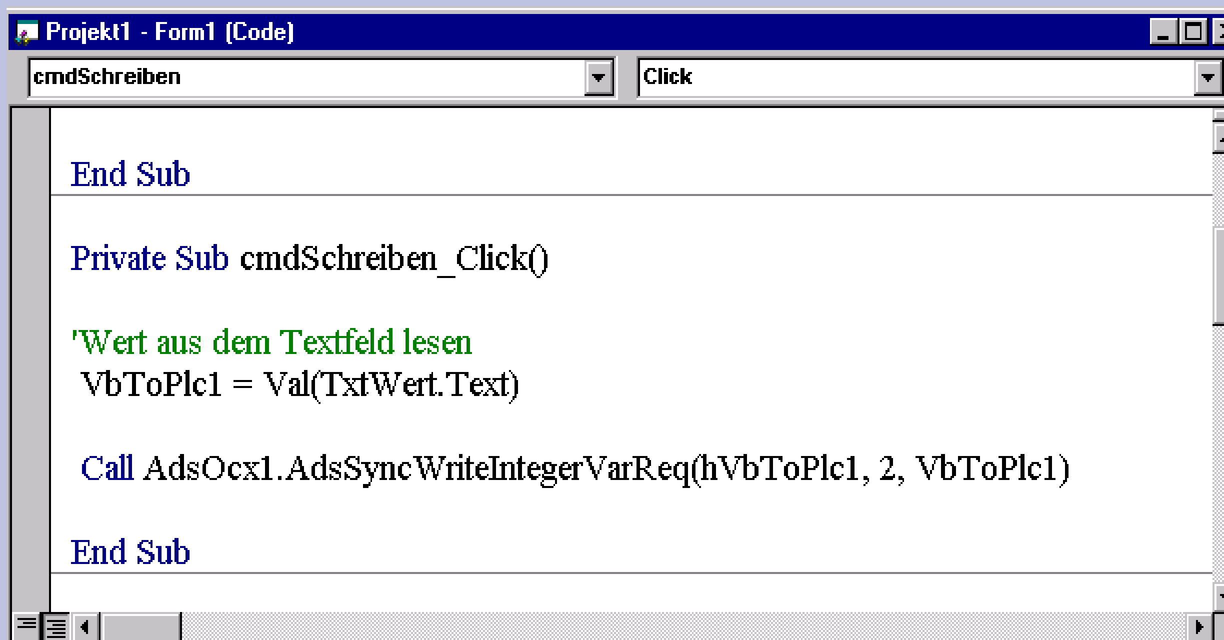
Handle which has previously been fetched

Length in bytes that will be required for transmission of the variable from the PLC to the Visual Basic application (integer ≅ 2 bytes)

Example 1

Visual Basic:

Summary: Click event



Test: Start PLC, save Visual Basic application and start. If everything is in order, the value in the text window of the Visual Basic program can be changed (-32768....+32767). After clicking the command button, the variable in the PLC program must have taken on this value.

Example 2

Visual Basic Example 2

Synchronous access to bitlocated address, synchronous by name, by name and connect to several variables and write arrays synchronously.

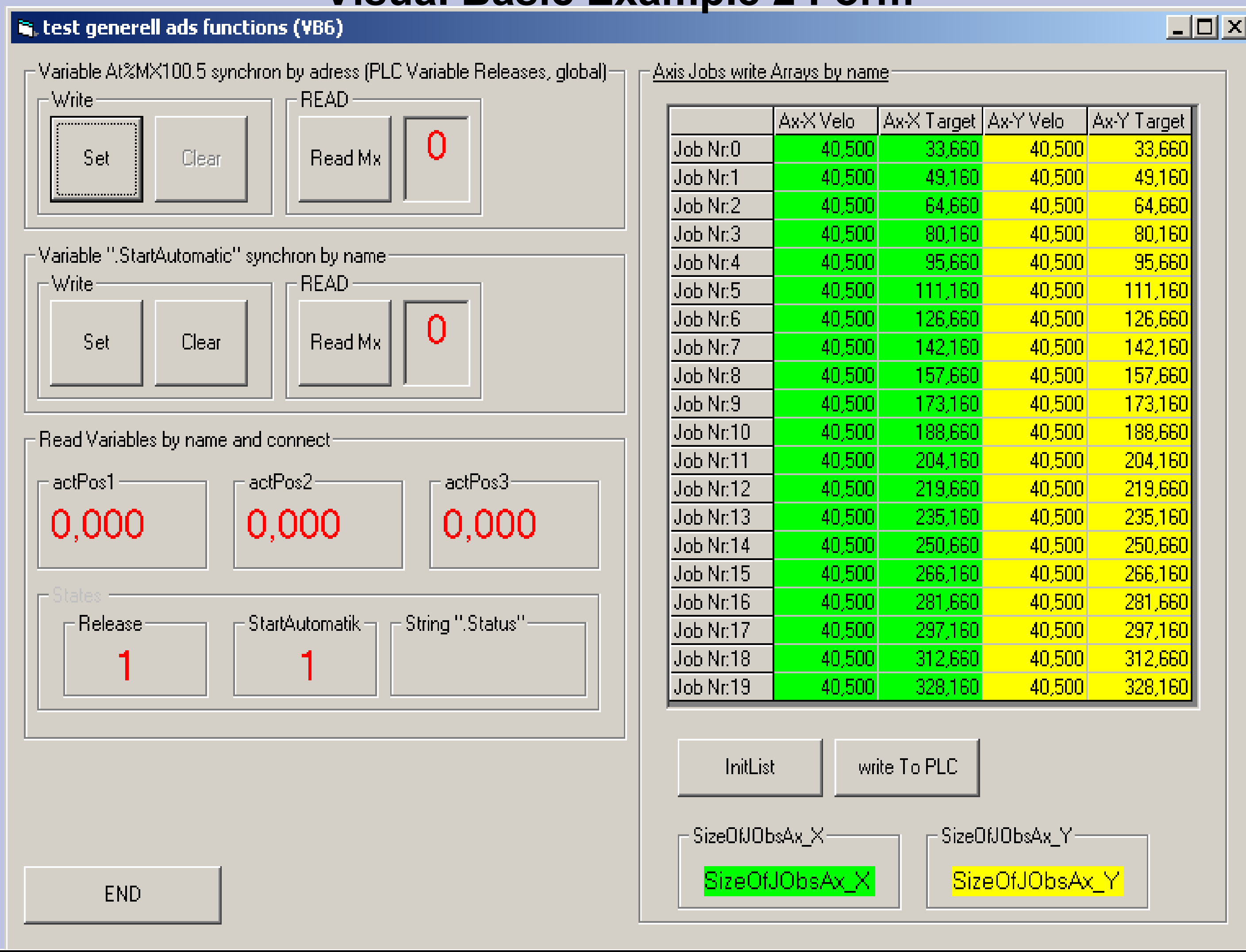
(Base project on training PC or disk)

The controls are already placed on the form. A small editing possibility has been programmed for the control „Msflexgrid“. This example shows the pure using of the methods of ADS Ocx to access to the variables.

The „Enable Errorhandling“ is used to handle with errors.

Example 2

Visual Basic Example 2 Form



	Ax-X Velo	Ax-X Target	Ax-Y Velo	Ax-Y Target
Job Nr:0	40,500	33,660	40,500	33,660
Job Nr:1	40,500	49,160	40,500	49,160
Job Nr:2	40,500	64,660	40,500	64,660
Job Nr:3	40,500	80,160	40,500	80,160
Job Nr:4	40,500	95,660	40,500	95,660
Job Nr:5	40,500	111,160	40,500	111,160
Job Nr:6	40,500	126,660	40,500	126,660
Job Nr:7	40,500	142,160	40,500	142,160
Job Nr:8	40,500	157,660	40,500	157,660
Job Nr:9	40,500	173,160	40,500	173,160
Job Nr:10	40,500	188,660	40,500	188,660
Job Nr:11	40,500	204,160	40,500	204,160
Job Nr:12	40,500	219,660	40,500	219,660
Job Nr:13	40,500	235,160	40,500	235,160
Job Nr:14	40,500	250,660	40,500	250,660
Job Nr:15	40,500	266,160	40,500	266,160
Job Nr:16	40,500	281,660	40,500	281,660
Job Nr:17	40,500	297,160	40,500	297,160
Job Nr:18	40,500	312,660	40,500	312,660
Job Nr:19	40,500	328,160	40,500	328,160

Example 2

Visual Basic Example 2 Form

Eigenschaften - AdsOcx1

AdsOcx1 AdsOcx

Alphabetisch | Nach Kategorien

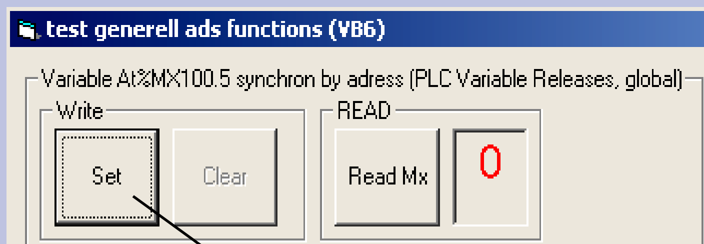
(Benutzerdefiniert)	
(Info)	
(Name)	AdsOcx1
AdsAmsClientNetId	172.16.5.27.1.1
AdsAmsClientPort	32817
AdsAmsCommTimeout	5000
AdsAmsConnected	True
AdsAmsSaveClientPort	False
AdsAmsServerNetId	172.16.5.27.1.1
AdsAmsServerPort	0
AdsClientAdsControl	
AdsClientAdsState	
AdsClientBuild	0
AdsClientDeviceControl	0
AdsClientDeviceState	0
AdsClientRevision	0
AdsClientType	
AdsClientVersion	0

Features of ADS OCX by assignment

```
'*****************************************************
'*
'* GENERELL SETTINGS FOR ADS OCX
'*
'*****************************************************
'
'-----------------------------------------------------

Sub setAdsOcxPropertys()
    With AdsOcx1
        .AdsAmsServerNetId = .AdsAmsClientNetId
        .AdsAmsServerPort = 801
        .EnableErrorHandling = True
    End With
End Sub
```

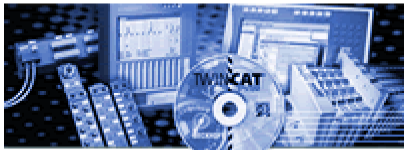
Example 2

Visual Basic Example 2 Form



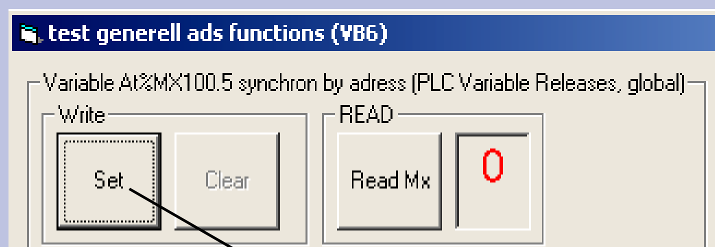
Access to bitlocated PLC variable

```
                                   *************************************
                                                                       *
'* EXAMPLE 1 : WRITE/READ  SYNCHRON BY ADRESS                          *
'*                                                                     *
'***********************************************************************
'
'-----------------------SET VALUE IN PLC--------------------------------------
Private Sub cmdSetMX_Click()
    'Write a located variable To PLC
    'Indexgroup for Bit 0x4021
    'Indexoffset  (DWOWRD)  ByteNr * 8 + Bit Nr
    Dim Idxgr As Long
    Dim idxOffs As Long
    Dim data As Boolean

    Idxgr = &H4021&
    idxOffs = 100 * 8 + 5 'Bytenr * 8 + BitNr
    data = True

    '--------write synchron by name------------------------------------
    Call AdsOcx1.AdsSyncWriteBoolReq(Idxgr, idxOffs, LenB(data), data)

    'NOT ads relevant
    cmdClearMx.Enabled = True
    cmdSetMX.Enabled = False
End Sub
```



Example 2

Visual Basic Example 2 Form

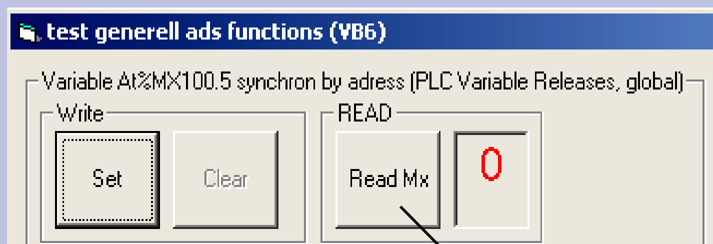


Access to bitlocated PLC variable

```
'-----------------------------CLEAR VALUE IN PLC--------------------------------------
Private Sub cmdClearMx_Click()
    'Write a located variable To PLC
    'Indexgroup for Bit 0x4021
    'Indexoffset  (DWOWRD)  ByteNr * 8 + Bit Nr
    Dim Idxgr As Long
    Dim idxOffs As Long
    Dim data As Boolean

    Idxgr = &H4021&
    idxOffs = 100 * 8 + 5 'Bytenr * 8 + BitNr
    data = False

    '--------write synchron by name-------------------------------------
    Call AdsOcx1.AdsSyncWriteBoolReq(Idxgr, idxOffs, LenB(data), data)

    'NOT ads relevant
    cmdClearMx.Enabled = False
    cmdSetMX.Enabled = True
End Sub
```

Example 2

Visual Basic Example 2 Form

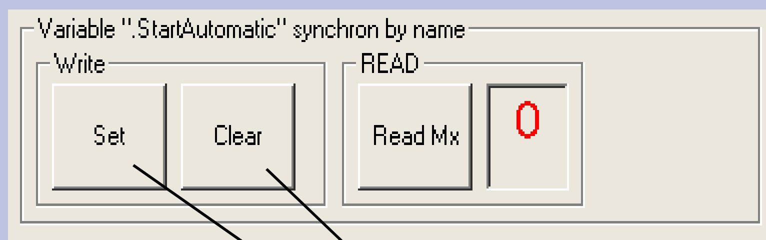


Access to bitlocated PLC variable

```
'------------------------READ VALUE BACK FROM PLC------
Private Sub cmdReadMx_Click()
    'Read a located variable To PLC
    'Indexgroup for Bit 0x4021
    'Indexoffset  (DWOWRD)  ByteNr * 8 + Bit Nr
    Dim Idxgr As Long
    Dim idxOffs As Long
    Dim data As Boolean

    Idxgr = &H4021&
    idxOffs = 100 * 8 + 5 'Bytenr * 8 + BitNr
    '--------write synchron by name-----------------------------------
    Call AdsOcx1.AdsSyncReadBoolReq(Idxgr, idxOffs, LenB(data), data)

    ' display value
    ' bool 1 -> INT -1 hex FFFF
    ' bool 0 -> INT 0  hex 0000

    lblValMx100.Caption = CStr(CInt(data) * -1)
End Sub
```

Example 2

Visual Basic Example 2 Form

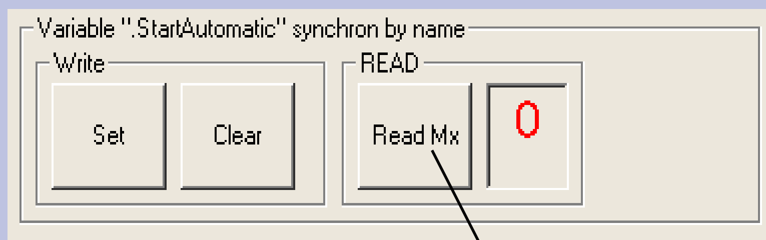


Access by name

```
'-----------------------WRITE THIS VALUE TO PLC---------------------------------------
Private Sub WriteStartAuto(StartAutomatic As Boolean)
' write bool variable by name
    'Dim StartAutomatic As Boolean
    Dim hStartAutomatic As Long

    'get handle
    Call AdsOcx1.AdsCreateVarHandle(".StartAutomatic", hStartAutomatic)
    'write synchron
    Call AdsOcx1.AdsSyncWriteBoolVarReq(hStartAutomatic, LenB(StartAutomatic), StartAutomatic)
    'release handle
    Call AdsOcx1.AdsDeleteVarHandle(hcStartAutomatic)
End Sub
```

Example 2

Visual Basic Example 2 Form

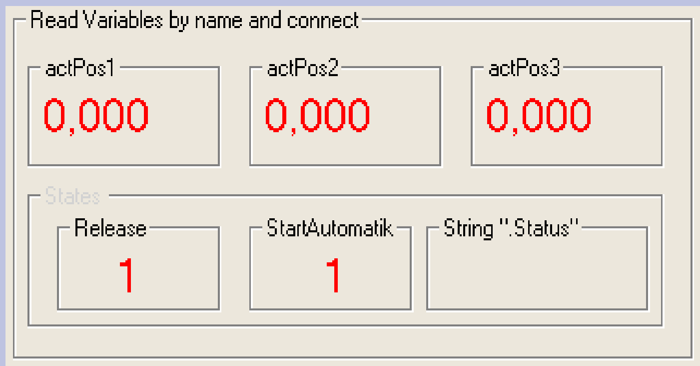


Access by name

```
'----------------------READ THIS VALUE FROM PLC--------------------------------------
Private Sub cmdReadStartAutomatic_Click()
    ' read bool variable synchron by name
    Dim StartAutomatic As Boolean
    Dim hStartAutomatic As Long

    'get handle
    Call AdsOcx1.AdsCreateVarHandle(".StartAutomatic", hStartAutomatic)
    'write synchron
    Call AdsOcx1.AdsSyncReadBoolVarReq(hStartAutomatic, LenB(StartAutomatic), StartAutomatic)
    'release handle

    'display variable
    lblValStartAutomatic.Caption = CStr(CInt(StartAutomatic) * -1)

    Call AdsOcx1.AdsDeleteVarHandle(hcStartAutomatic)
End Sub
```

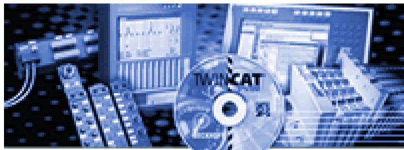
Example 2

Visual Basic Example 2 Form



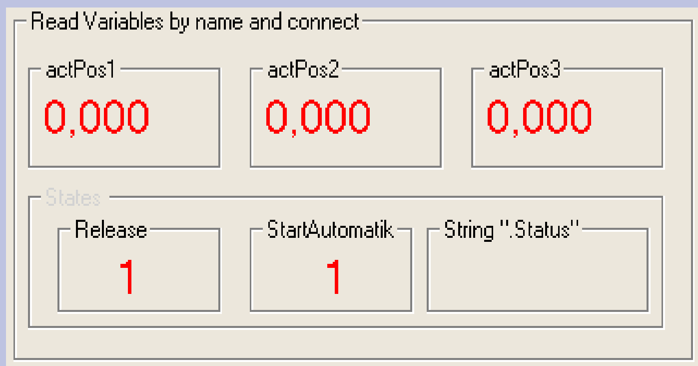
Access by name and connect
Step 1: connecting

```
'************************************************************************
'*                                                                      *
'* EXAMPLE 3 : READ CONNECT BY NAME                                     *
'*                                                                      *
'************************************************************************
'
'----------------------Connect all variables-----------------------------
Sub connectVars()
    ' connect variables and save handles
    Call AdsOcx1.AdsReadVarConnectEx(".release", ADSTRANS_SERVERONCHA, 100, hcrelease)
    Call AdsOcx1.AdsReadVarConnectEx(".StartAutomatic", ADSTRANS_SERVERONCHA, 100, hcStartAutomatic)
    Call AdsOcx1.AdsReadVarConnectEx(".actPos1", ADSTRANS_SERVERONCHA, 100, hcactPos1)
    Call AdsOcx1.AdsReadVarConnectEx(".actPos2", ADSTRANS_SERVERONCHA, 200, hcactPos2)
    Call AdsOcx1.AdsReadVarConnectEx(".actPos3", ADSTRANS_SERVERONCHA, 300, hcactPos3)
    Call AdsOcx1.AdsReadVarConnectEx(".status", ADSTRANS_SERVERONCHA, 100, hcstatus)
End Sub
```



Example 2

Visual Basic Example 2 Form

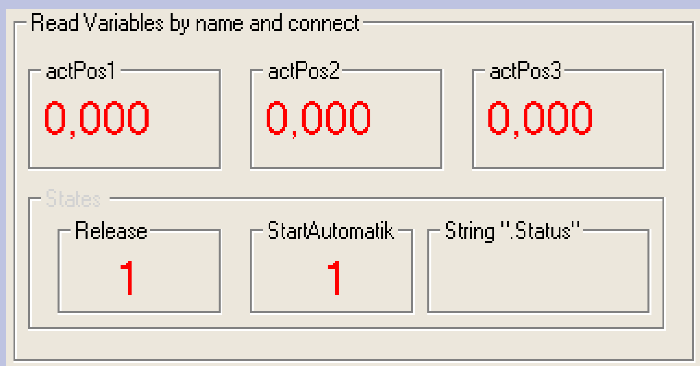


Access by name and connect
Step 2: analyse event from ADS OCX

```
'-----------------------Event from ADS  --------------------------------
Private Sub AdsOcx1_AdsReadConnectUpdateEx(ByVal dateTime As Date, ByVal nMs As Long, _
                                           ByVal hConnect As Long, ByVal data As Variant, _
                                           Optional ByVal hUser As Variant)
' called when new variables here
    Select Case hConnect
        Case hcrelease: lblReleaseConnect.Caption = CStr(CInt(data) * -1)
        Case hcStartAutomatic: lblStartAutomaticConnect.Caption = CStr(CInt(data) * -1)
        Case hcactPos1: lblactPos1.Caption = Format(data, "##0.000", standard)
        Case hcactPos2: lblactPos2.Caption = Format(data, "##0.000", fixed)
        Case hcactPos3: lblactPos3.Caption = Format(data, "##0.000", fixed)
        Case hcstatus: lblStatusString = data
    End Select

End Sub
```

Example 2

Visual Basic Example 2 Form

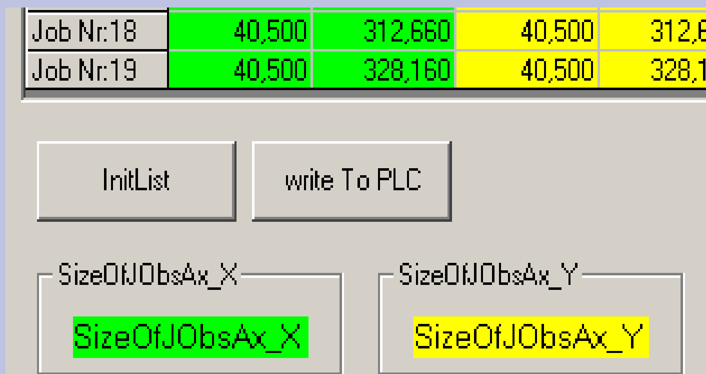


Access by name
and connect
Step 3: disconnect

```
'----------------------Disconnect Add variables  ------------
Sub disconnectVars()
    Call AdsOcx1.AdsDisconnectEx(hcrelease)
    Call AdsOcx1.AdsDisconnectEx(hcStartAutomatic)
    Call AdsOcx1.AdsDisconnectEx(hcactPos1)
    Call AdsOcx1.AdsDisconnectEx(hcactPos2)
    Call AdsOcx1.AdsDisconnectEx(hcstatus)
End Sub
```

Example 2

Visual Basic Example 2 Form

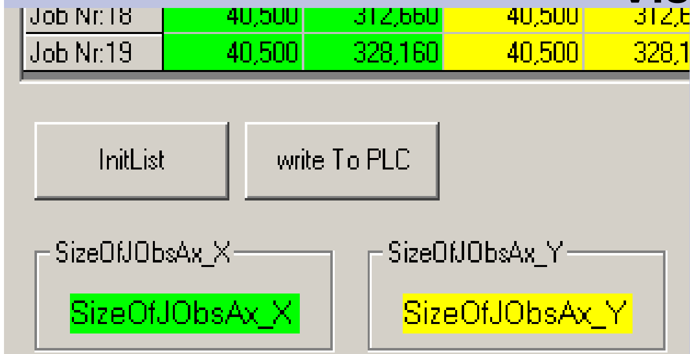


Access to an array synchronously by name
first array

```
'-----------------Write array for X-Ax-------------------------------------
Private Sub writeJobsAx_X()
    '---------Check size of array-------------
    'lenb(jobsax) -> not possible
    Dim NrOfField As Long
    ' Nr of Field in Array
    NrOfField = _
    (UBound(jobsax_X, 1) - LBound(jobsax_X, 1) + 1) * _
    (UBound(jobsax_X, 2) - LBound(jobsax_X, 2) + 1)
    'size in byte = lenof 1 element * Nr of fields
    SizeOfjobsax_X = LenB(jobsax_X(0, 0)) * NrOfField
    'Info to label
    lblSizeOfJObsAx_X.Caption = CStr(SizeOfjobsax_X)

    '--send via synchron WRITE---------------------------------------------------
    Call AdsOcx1.AdsCreateVarHandle(".jobsax_X", hjobsax_X)
    Call AdsOcx1.AdsSyncWriteDoubleVarReq(hjobsax_X, SizeOfjobsax_X, jobsax_X(0, 0))
    Call AdsOcx1.AdsDeleteVarHandle(hjobsax_X)
End Sub
```

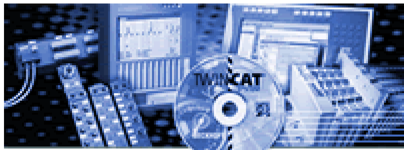
Example 2

Visual Basic Example 2 Form



Access to an array synchronously by name
second array

```
'-----------------Write array for Y-Ax-------------------------------------
Private Sub writeJobsAx_Y()
    '---------Check size of array-------------
    'lenb(jobsax) -> not possible
    Dim NrOfField As Long
    ' Nr of Field in Array
    NrOfField = _
    (UBound(jobsax_Y, 1) - LBound(jobsax_Y, 1) + 1) * _
    (UBound(jobsax_Y, 2) - LBound(jobsax_Y, 2) + 1)
    'size in byte = lenof 1 element * Nr of fields
    SizeOfjobsax_Y = LenB(jobsax_Y(0, 0)) * NrOfField
    'Info to label
    lblSizeOfJObsAx_Y.Caption = CStr(SizeOfjobsax_Y)

    '--send via synchron WRITE-------------------------------------------------
    Call AdsOcx1.AdsCreateVarHandle(".jobsax_Y", hjobsax_Y)
    Call AdsOcx1.AdsSyncWriteDoubleVarReq(hjobsax_Y, SizeOfjobsax_Y, jobsax_Y(0, 0))
    Call AdsOcx1.AdsDeleteVarHandle(hjobsax_Y)
End Sub
```



Example 2

Visual Basic Example 2 Form

Required PLC variable

[Global_Variables]

Extras Online Fenster Hilfe

```
VAR_GLOBAL

    jobsax_X : ARRAY[0..19, 0..1] OF LREAL;
    jobsax_Y : ARRAY[0..19, 0..1] OF LREAL;


    release   AT%MX100.5:BOOL;     (* Read by Adress and by Name Connect*)
    StartAutomatic :BOOL;          (* Read by Name connect*)

    actPos1 :LREAL;                (* Read by name and connect*)
    actPos2 :LREAL;
    actPos3 :LREAL;

    status:    STRING;             (* Read by name and connect*)
```

Example VBA I

Example AdsOcx with Genesis32 VBA

Synchronous data exchange

Example VBA I: Synchronized write/read of PLC variables

```
VAR_GLOBAL
	Poti1 :INT;
	Poti2 :INT;
	Produkt:STRING(10);
END_VAR
```

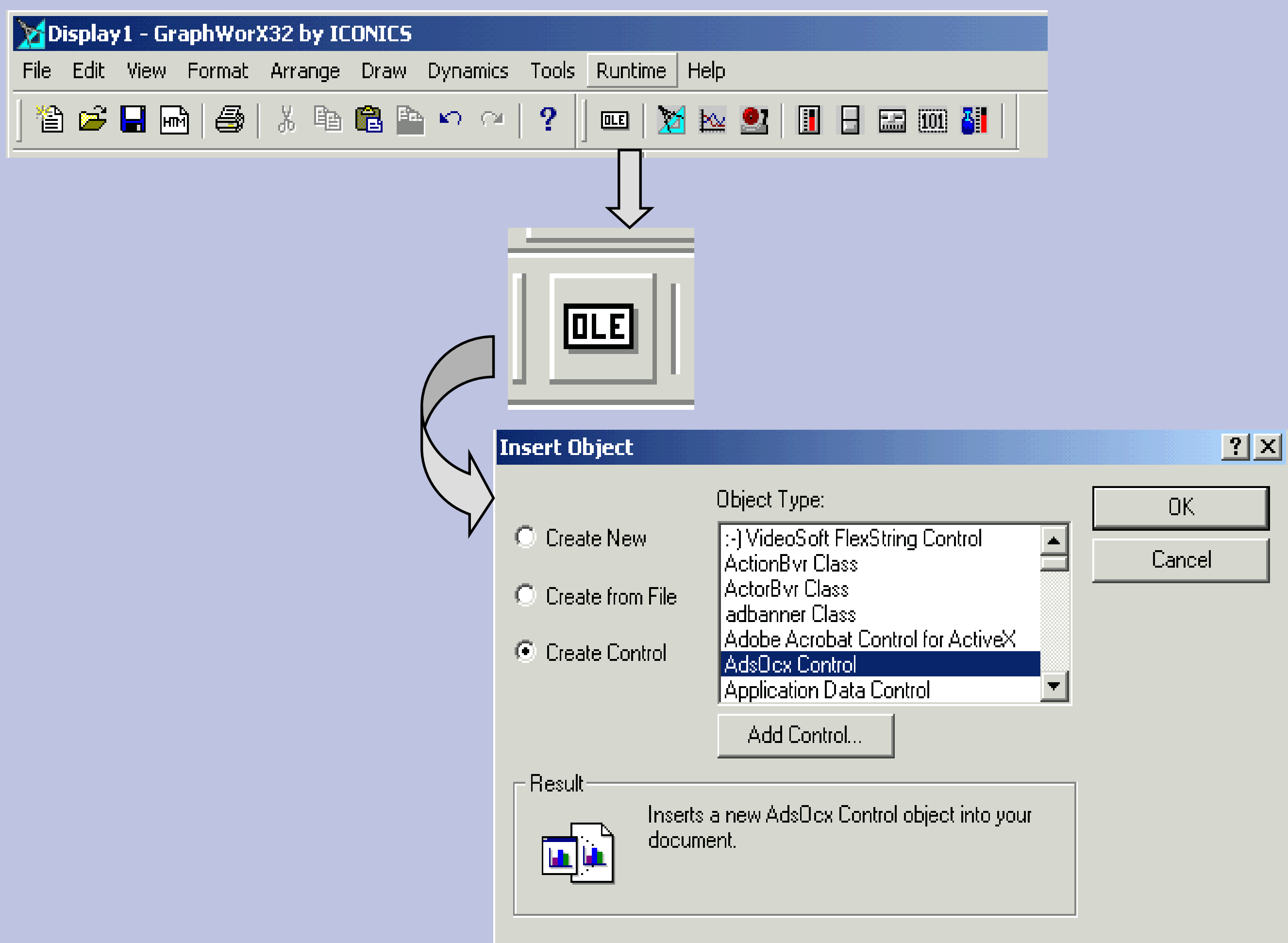
Synchronized read

Synchronized write

```
(*-Sim Poti1— *)
(* Takt*)
LD	t1.Q
STN		T1.IN
CAL	T1(PT:=T#100ms)
(* Zaehler*)
CAL	CTU1(CU:=T1.IN , RESET:=ctu1.Q , PV:=32767 )
LD	ctu1.CV
ST	Poti1
```

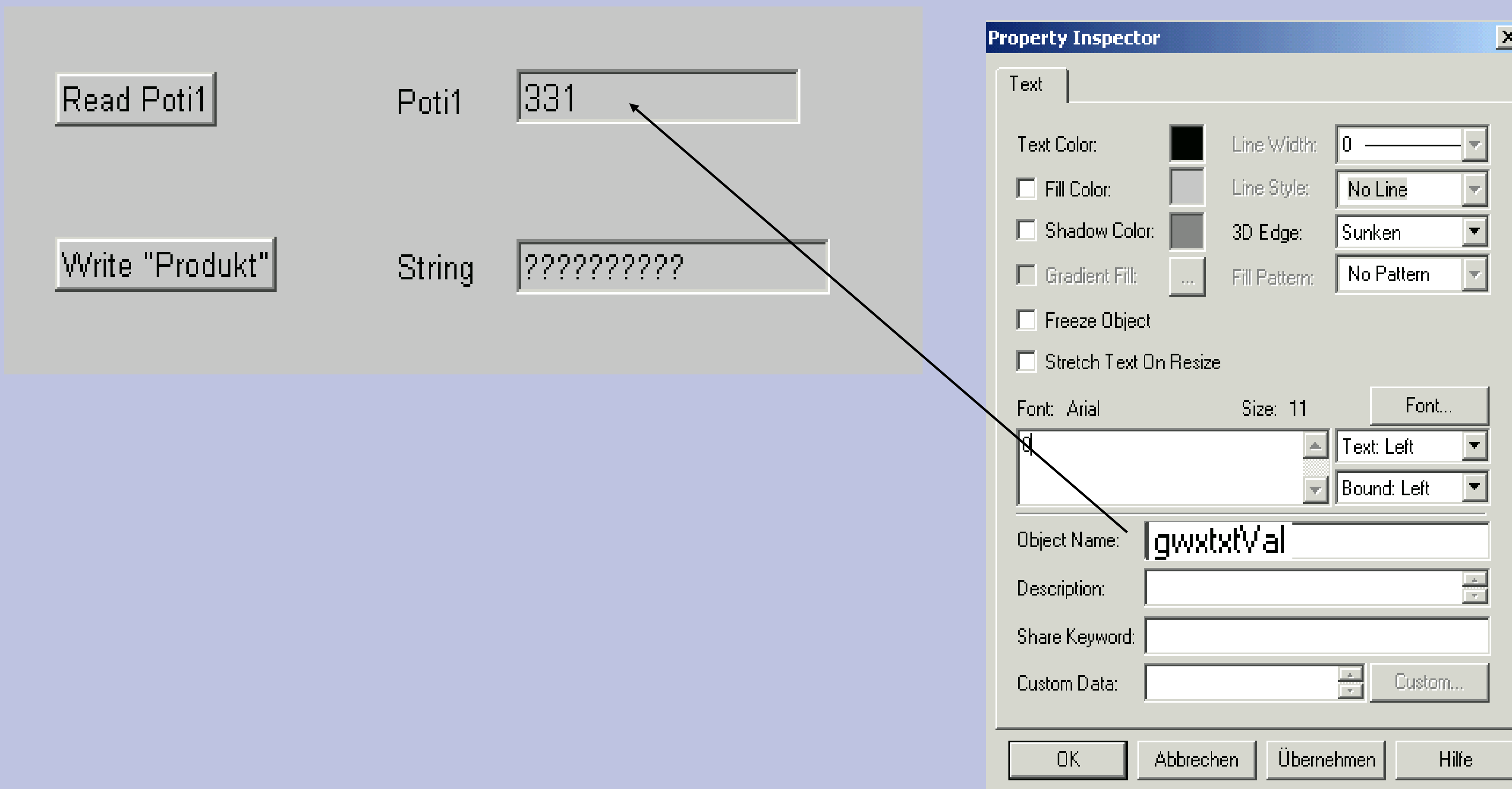
PLC program changes „Poti1“

Example VBA I: Graphworks32

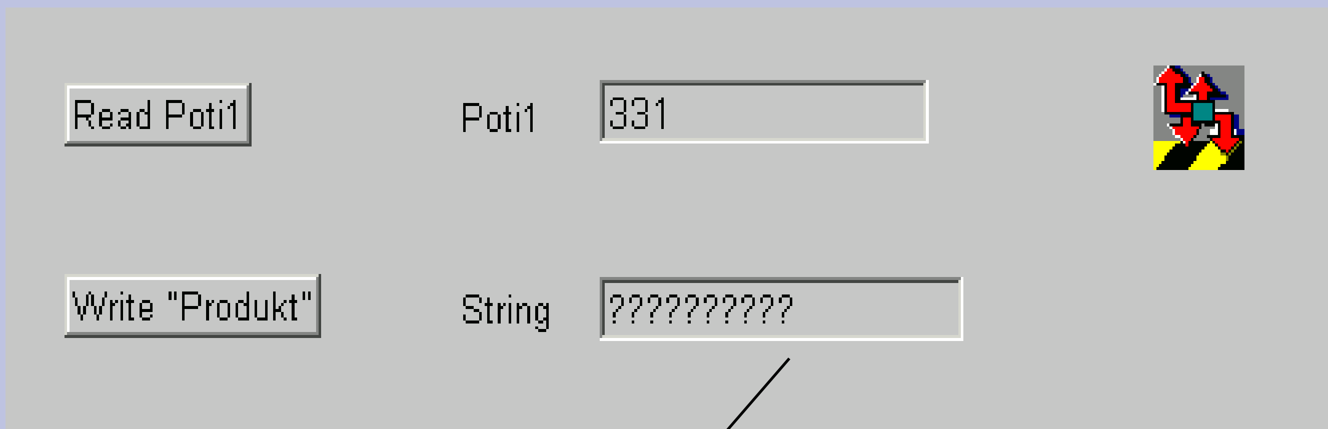


Example VBA I: Graphworks32

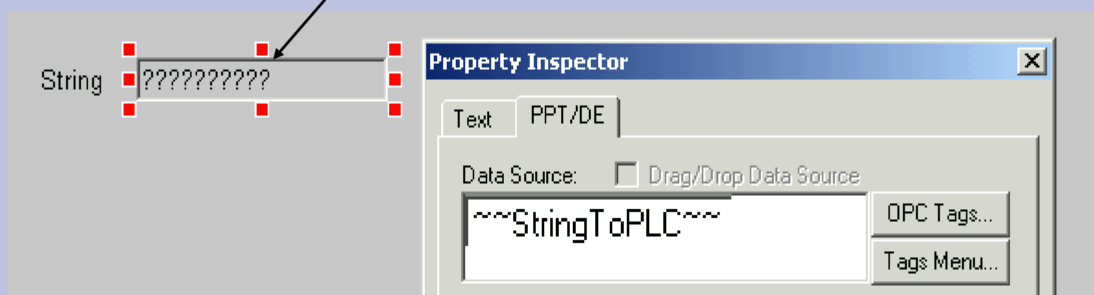
Text



Example VBA I: Graphworks32



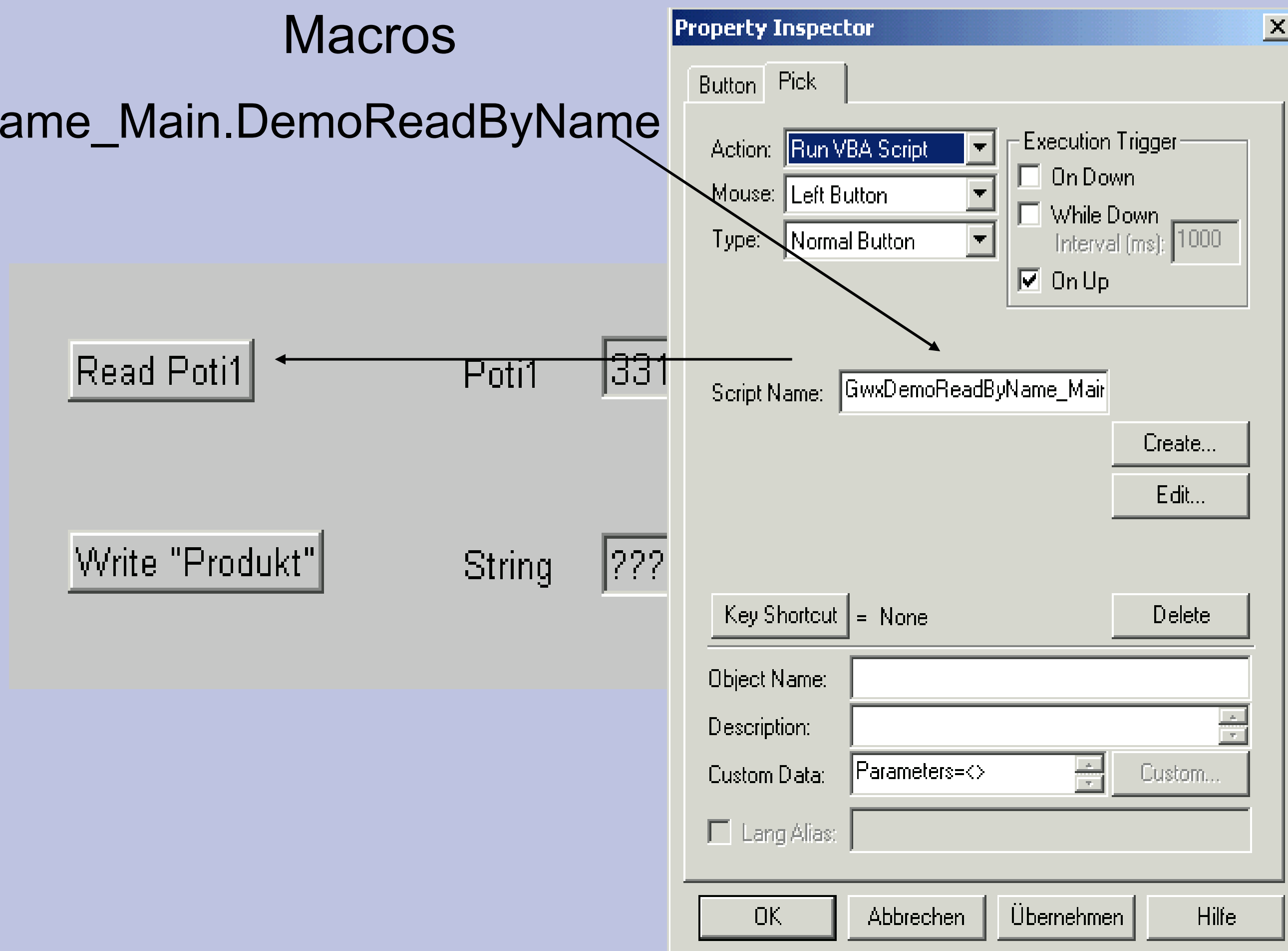
Process Point with located variable



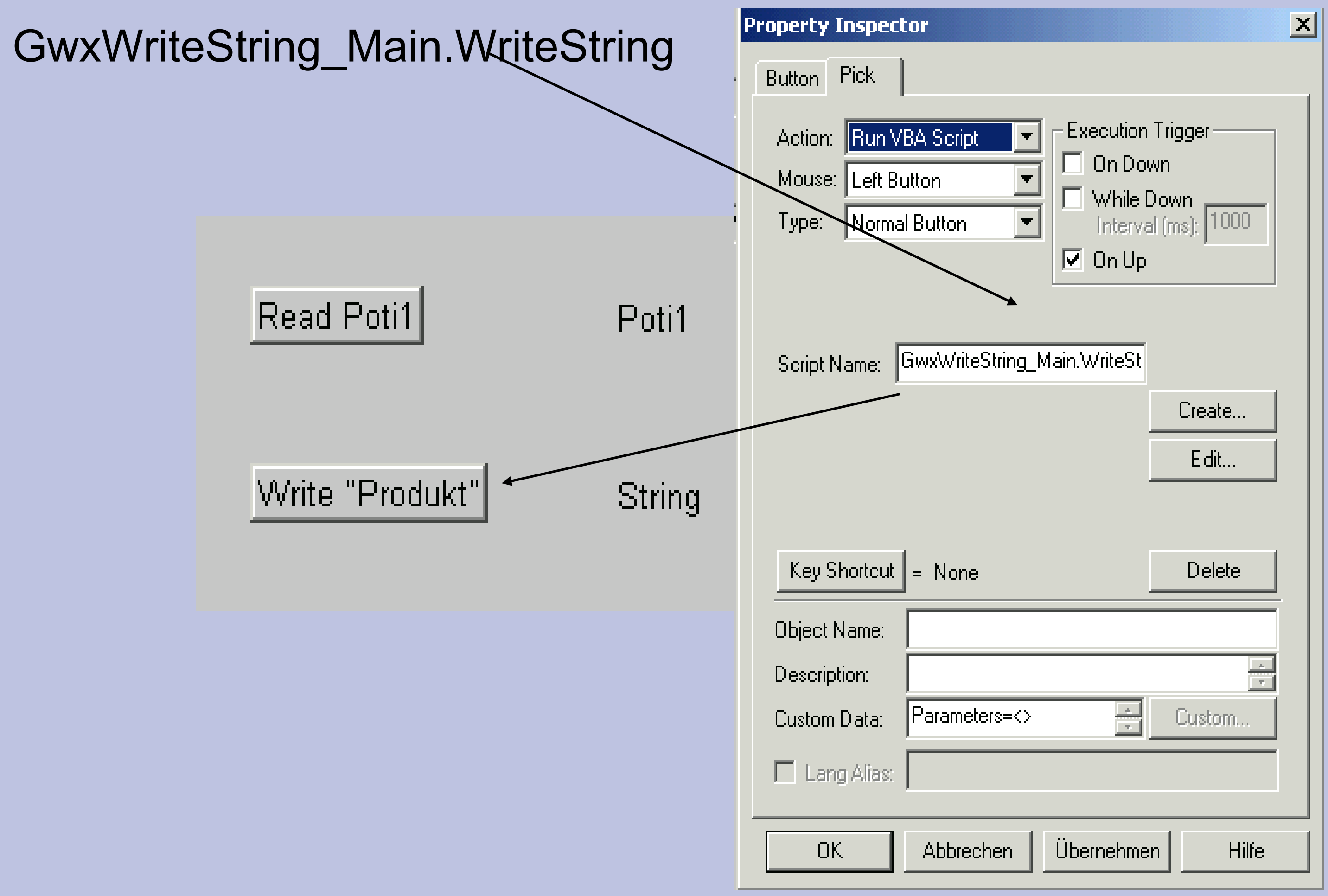
Example VBA I: Graphworks32

Macros

GwxDemoReadByName_Main.DemoReadByName



Example VBA I: Graphworks32



Example VBA I: Graphworks32

Set macro properties and get handle for „Poti“

PermanentLesen - ThisDisplay (Code)

GwxDisplay

```
'Handles
Public hConnect1 As Long
Public hConnect2 As Long
Public hConnectString As Long
```

```
Private Sub GwxDisplay_PostRuntimeStart()
    With ThisDisplay.AdsOcx1
        'Properties
        .AdsAmsServerNetId = "172.16.200.200.1.1"
        .AdsAmsServerPort = 801
        .EnableErrorHandling = True
        'Hanlde
        Call ThisDisplay.AdsOcx1.AdsCreateVarHandle(".Poti1", hpoti1)

    End With
End Sub
```

Example VBA I: Graphworks32

Macro “read poti”(„Poti lesen“)

```
Sub DemoReadByName(o As GwxPick)

    'read
    Dim vbpoti1 As Integer
    Call ThisDisplay.AdsOcx1.AdsSyncReadIntegerVarReq  (  ThisDisplay.hpoti1, _
                                                           2, _
                                                           vbpoti1)

    'Display value
    Dim vbtext As GwxText 'Objektvariable für das Anzeigefeld
    Set vbtext = ThisDisplay.GetVisibleObjectFromName("gwxtxtVal")
    vbtext.text = CStr(vbpoti1)
End Sub
```

Example VBA I: Graphworks32

Macro “Write product” („Produkt schreiben“)

```
Sub WriteString(o As GwxPick)

    Dim hProdukt As Long     ' Handle
    Dim Produkt As String    ' VB Variable

    Dim text As GwxPoint     'Process Point
    Set text = ThisDisplay.GetPointObjectFromName("~~StringToPLC~~")
    Produkt = text.Value

    'GetHandle / write / release handle
    With ThisDisplay.AdsOcx1
        Call .AdsCreateVarHandle(".Produkt", hProdukt)
        Call .AdsSyncWriteStringVarReq(hProdukt, 10, Produkt)
        Call .AdsDeleteVarHandle(hProdukt)
    End With
End Sub
```

If a variable is not written frequently, producing handle, writing value and dissolving value can be run through in a sequence.

Example VBA I: Graphworks32

Delete macro handle for „Poti“ by end

```
Private Sub GwxDisplay_PostRuntimeStop()
    Call ThisDisplay.AdsOcx1.AdsDeleteVarHandle(hpoti1)
End Sub
```

Example VBA II

Example AdsOcx

with Genesis32 VBA Connect

Start run-time-> create handle

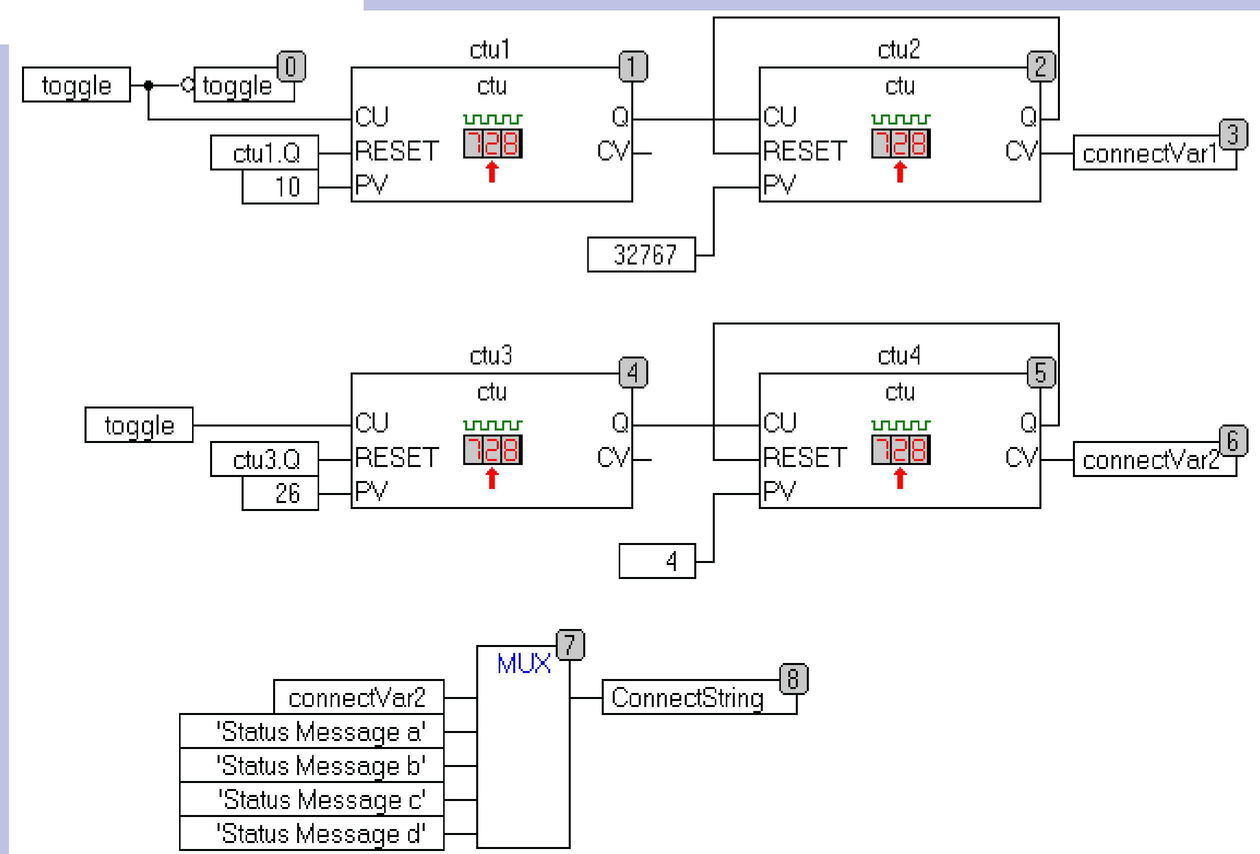
Ads Ocx -> analyse and display data

Stop run-time -> delete handle

Example VBA II: PLC

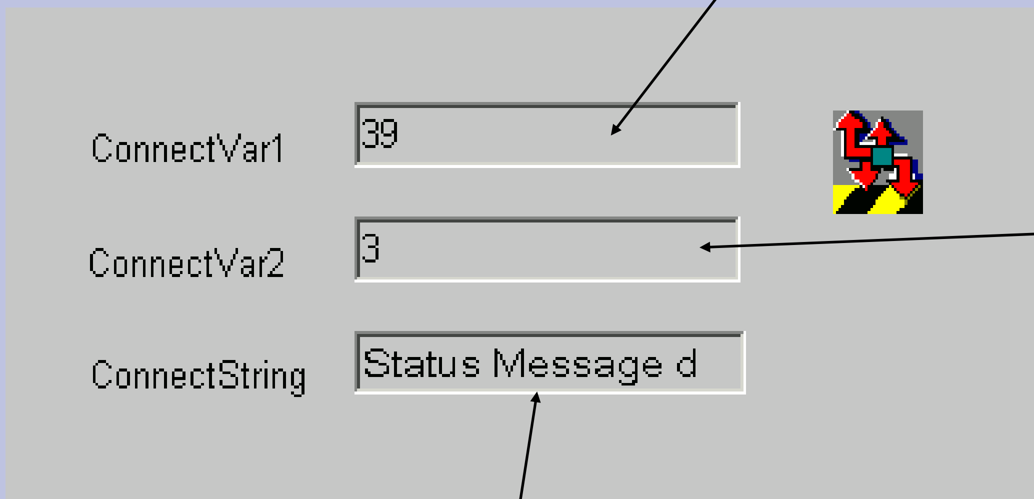
Globale_Variablen

```
0001 VAR_GLOBAL
0002     connectVar1:     UINT;
0003     connectVar2:     INT;
0004     ConnectString:   STRING(80);
0005 END_VAR
```



Example VBA II: VB macro handle to connection

„Text“ Object Name ConnectVar1



Object Name ConnectVar1

Object Name ConnectVar1

Example VBA II: Macro Connect

PermanentLesen - ThisDisplay (Code)

GwxDisplay

```
Public hConnect2 As Long
Public hConnectString As Long
```

PermanentLesen - ThisDisplay (Code)

GwxDisplay | PostRuntimeStart

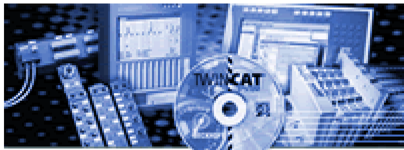
```
Private Sub GwxDisplay_PostRuntimeStart()
    With ThisDisplay.AdsOcx1
        .AdsAmsServerNetId = .AdsAmsClientNetId
        .AdsAmsServerPort = 801
        .EnableErrorHandling = True
        On Error GoTo er

        Call .AdsReadVarConnectEx(".connectVar1", _
                                  ADSTRANS_SERVERONCHA, _
                                  100, _
                                  hConnect1)
        Call .AdsReadVarConnectEx(".connectVar2", _
                                  ADSTRANS_SERVERONCHA, _
                                  100, _
                                  hConnect2)
        Call .AdsReadVarConnectEx(".ConnectString", _
                                  ADSTRANS_SERVERONCHA, _
                                  100, _
                                  hConnectString)
        Exit Sub

er:     MsgBox ("Fehler Connect!" + CStr(Err.Description))
        'Dim zaehler
        'zaehler = False
    End With
End Sub
```

Example VBA II: analyse event ADS OCX

PermanentLesen - ThisDisplay (Code)

AdsOcx1 | AdsReadConnectUpdateEx

```
Private Sub AdsOcx1_AdsReadConnectUpdateEx(ByVal dateTime As Date, _
                                            ByVal nMs As Long, _
                                            ByVal hConnect As Long, _
                                            ByVal data As Variant, _
                                            Optional ByVal hUser As Variant)

        Dim vbtext As GwxText 'Objektvariable für das Anzeigefeld
            If hConnect = hConnect1 Then
                Set vbtext = ThisDisplay.GetVisibleObjectFromName("ConnectVar1")
                vbtext.Text = CStr(data)
            ElseIf hConnect = hConnect2 Then
                Set vbtext = ThisDisplay.GetVisibleObjectFromName("ConnectVar2")
                vbtext.Text = CStr(data)
            ElseIf hConnect = hConnectString Then
                Set vbtext = ThisDisplay.GetVisibleObjectFromName("ConnectString")
                vbtext.Text = CStr(data)
            End If

End Sub
```



Example VBA II: Disconnect at run-time stop

PermanentLesen - ThisDisplay (Code)

GwxDisplay | PostRuntimeStop

```
Private Sub GwxDisplay_PostRuntimeStop()
    Call ThisDisplay.AdsOcx1.AdsDisconnectEx(hConnect1)
    Call ThisDisplay.AdsOcx1.AdsDisconnectEx(hConnect2)
    Call ThisDisplay.AdsOcx1.AdsDisconnectEx(hConnectString)
End Sub
```

Example VBA III

Example AdsOcx with Exel VBA

Data exchange with Connect and filling the list

(In preperation)

NC-PTP

TwinCAT

The Windows Control and Automation Technology

NC PTP

Numerical Control Point To Point

NC-PTP

Part I General
- Overview
- Axis types
- Functional principle
- Referencing
- Motion Control Function blocks

Teil II Practical Part:
- Setting up NC axes in the System Manager
- Starting NC axes from the PLC

Software NC PTP

TwinCAT NC Point-to-Point (PTP) is an axis positioning software with integrated PLC, NC interface, operating program for axes setup and I/O connection of the axes through the fieldbus.
Up to 255 axes can be moved at the same time.
TwinCAT NC PTP supports axis drive by switched motors, stepper motors, frequency controlled and servo controlled motors.

Software NC PTP

TwinCAT NC PTP

Programming	Performed using function blocks for TwinCAT PLC according to IEC61131-3, convenient axis commissioning menus
Debugging	Online monitoring of all axis state variables such as actual/set value, enable, controller values, online axis tuning, forcing axis variables
Runtime system	NC Point-to-Point (NC PTP) including TwinCAT PLC
Number of axes	Up to 255 in up to 255 channels
Axis types	Electrical and hydraulic servo drives, frequency converter drives, stepper motor drives, switched drives (fast/crawl axes)
Cycle time	Min. 50 µs, typ. 1 ms (freely adjustable)
Axis functions	Standard axis functions: start/stop/reset/reference Velocity override, target override Special functions: master-slave cascading, electronic gearboxes, online distance compensation of segments

Camshafts, Flying saw, FIFO

Camshafts

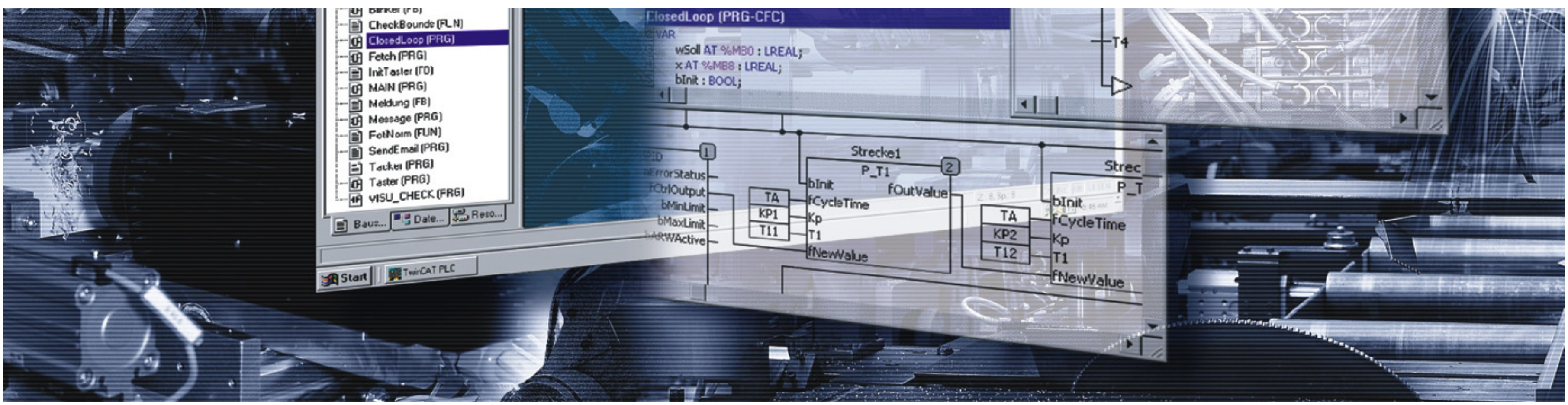
Software solution for electronic camshafts, obviating the need to use mechanical camshafts and special hardware assemblies. A table relates the position of the master axis (mainshaft) to the associated position to which the slave axis is driven.

Flying saw

The "flying saw" (diagonal slave) is a special kind of slave coupling. The slave axis is brought from standstill to a speed synchronous with the master.

FIFO

Instead of using internal generation of standard set values, an axis can also obey an externally calculated sequence of set values that can be supplemented as the movement of the axis proceeds (FIFO buffer).



Software NC I

TwinCAT NC Interpolation (NC I) is the NC system for linear or circular interpolated path movements of axis groups each involving two or three drives. TwinCAT NC I offers 2D and 3D interpolation (interpreter, set point generation, position controller), an integrated PLC with an NC-I interface and an I/O connection for axes via the field bus.

TwinCAT NC I

Programming	DIN 66025 programs for NC interpolation, access via function blocks for TwinCAT PLC according to IEC61131-3
Debugging	Online monitoring in the TwinCAT System Manager with the following displays: present set/actual positions, following errors of all axes, NC program line presently being executed/interpreted, channel status
Runtime System	NC PTP + NC interpolation, including TwinCAT PLC
Number of axes	3 axes per group, 1 group per channel, max. 255 channels
Axis types	Electrical servo-axes
Interpreter-functions	Subroutines and jumps, programmed loops, zero shifts, tool compensations, M and H functions,
Geometries	Straight lines and circular paths in 3D space, circular paths in all main planes, helixes with base circles in all main planes
Axis functions	Online reconfiguration of axes in groups, path override, slave coupling to path axes

Axis types

Continuous axes

The axis responds to a continuously changeable set value
The set value is generated by TwinCAT NC,

e.g. servo with +/- 10 V, Sercos drive, frequency converter, linearised hydraulic axis,
stepper motor drive with amplifier

Axis types

High/low speed axes

The axis responds to a two-stage set speed value including direction of rotation:

FAST/SLOW and FORWARDS/REVERSE

The set value is generated by TwinCAT NC,

e.g. frequency converter with fast/slow inputs, combination interlock. Warning: Acquisition of actual value (Encoder is necessary)

Axis types

Low cost stepper motor

The axis consists of a stepper motor which is connected to digital outputs and reacts to pulses (A/B from the terminals)
Fast pulse sequence -> motor turns quicklylSlow pulse sequence -> motor turns slowly

The set value (= pulse pattern) is generated by TwinCAT NC.

Axis types

Low cost stepper motor, Hardware

e.g. 24 Volt stepper motor with 2A output terminals

An encoder is NOT required
for acquisition of the actual value, since the pulses that are output are counted.

! The mechanical design and/or maximum rotary speed/torque should be examined to ensure that the motor will be able to "keep up", since an output terminal cannot provide an increased voltage at higher frequency

Axis types

Virtual encoder axis,

An axis that only consists of an encoder.

"Normal” (continuous) axes can be coupled to this axis as slaves, and follow the set encoder value of the virtual encoder axis.
(Gear ration possible)

HAND WHEEL FUNCTION

Axis types

Output is a speed value
The actual position is monitored.

<u>Output:</u>
Speed pre-control
+ controller output

(acceleration pre-control also is optional)

<u>Feedback:</u>
Actual position value
At specific axis types e.g. SERCOS is also a direct output of the **Setposition** in NC time possible.

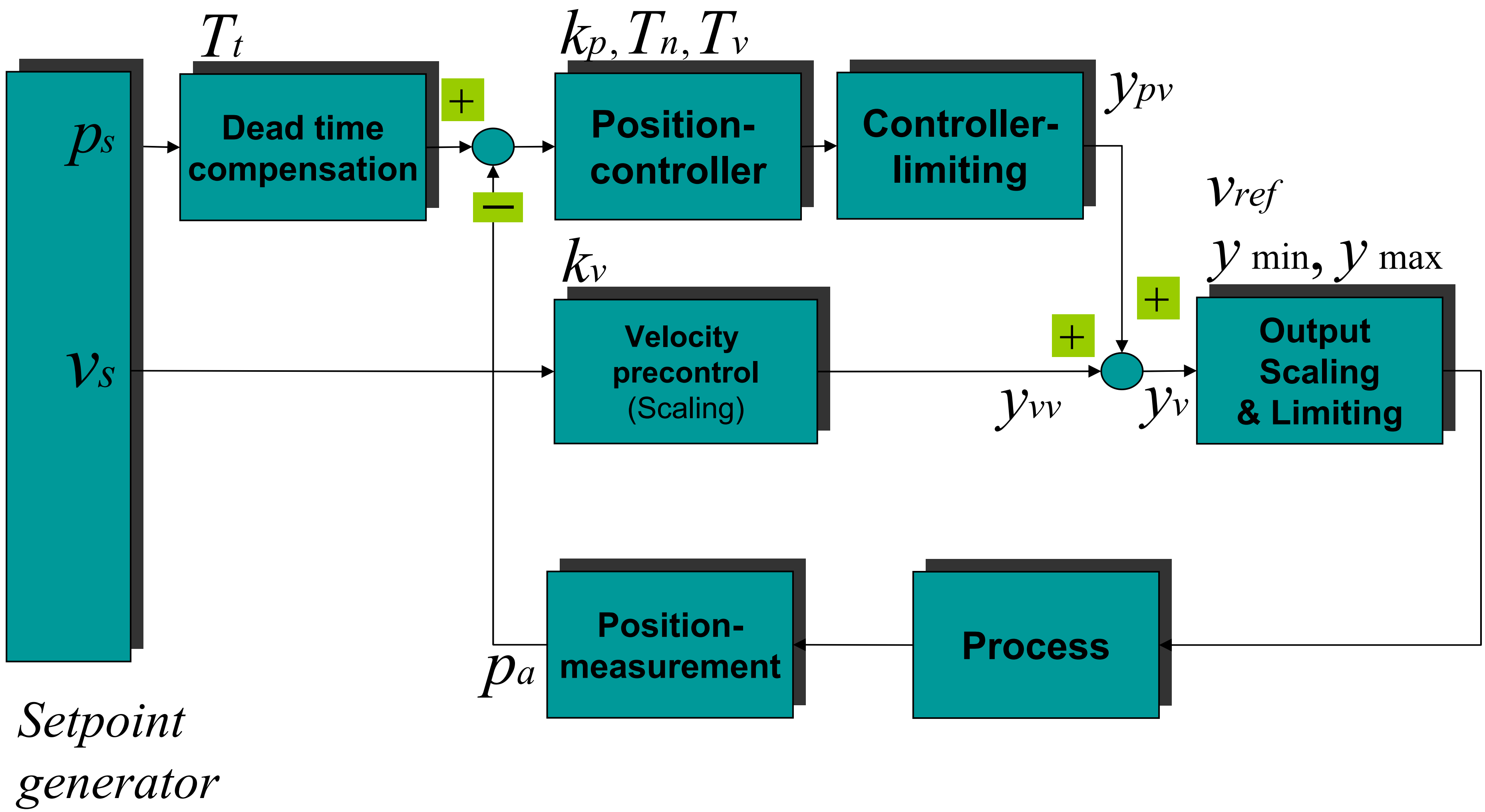
Functional principle of the TwinCAT NC

TwinCAT NC works with a velocity pre control.
The Position controller controls the observance of the set position („Motion“ and position control).

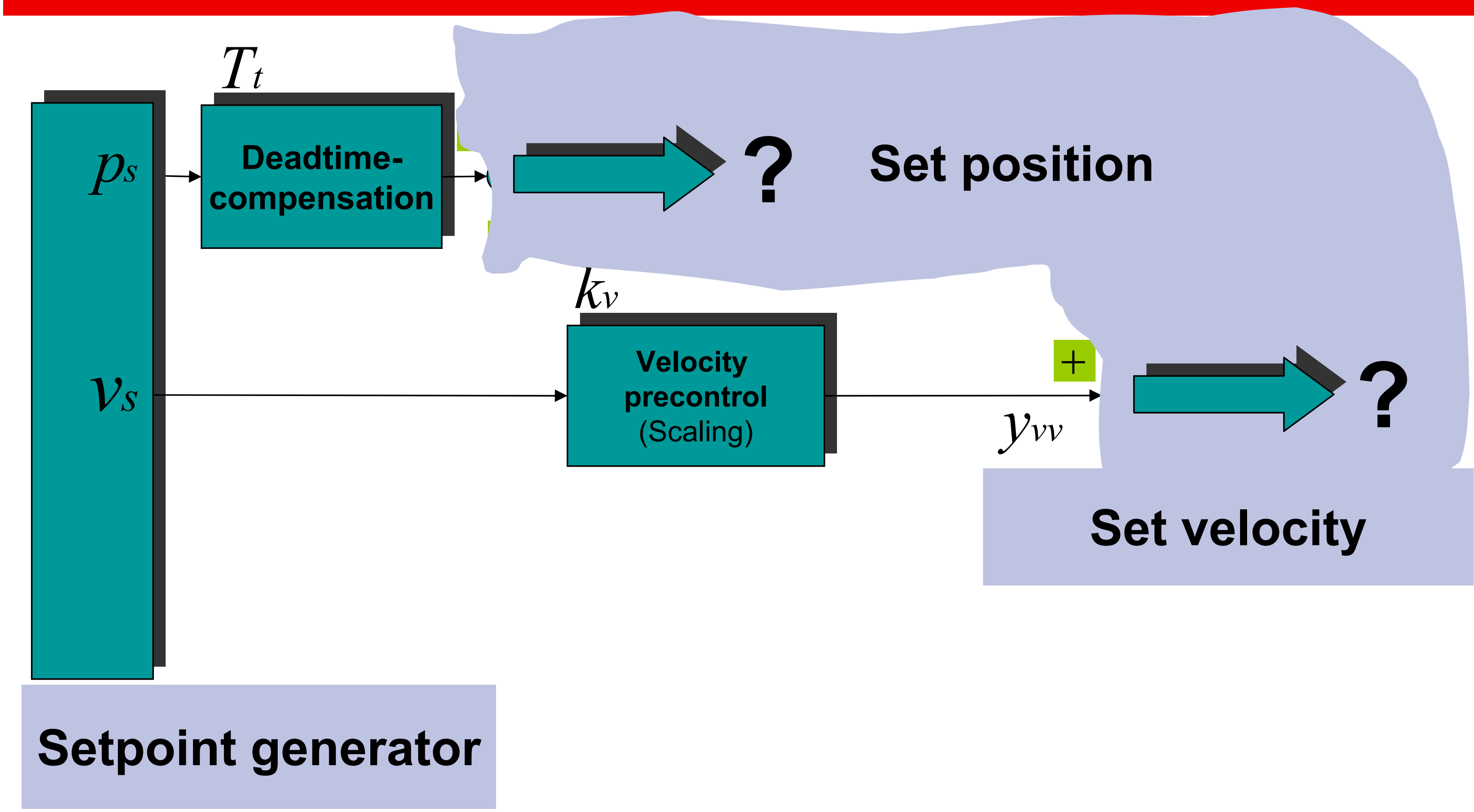
Further available options:
-Acceleration pre control
-Position control with two P constants
-direct output of the position. (Sercos Axes)

-High / low speed controller
-Stepper motor controller
-External Setpoint generation (ab TwinCAT 2.9)
-Linearisation of pre control for non linear axes (Hydraulic axes).

Functional principle of the TwinCAT NC



Set value generation

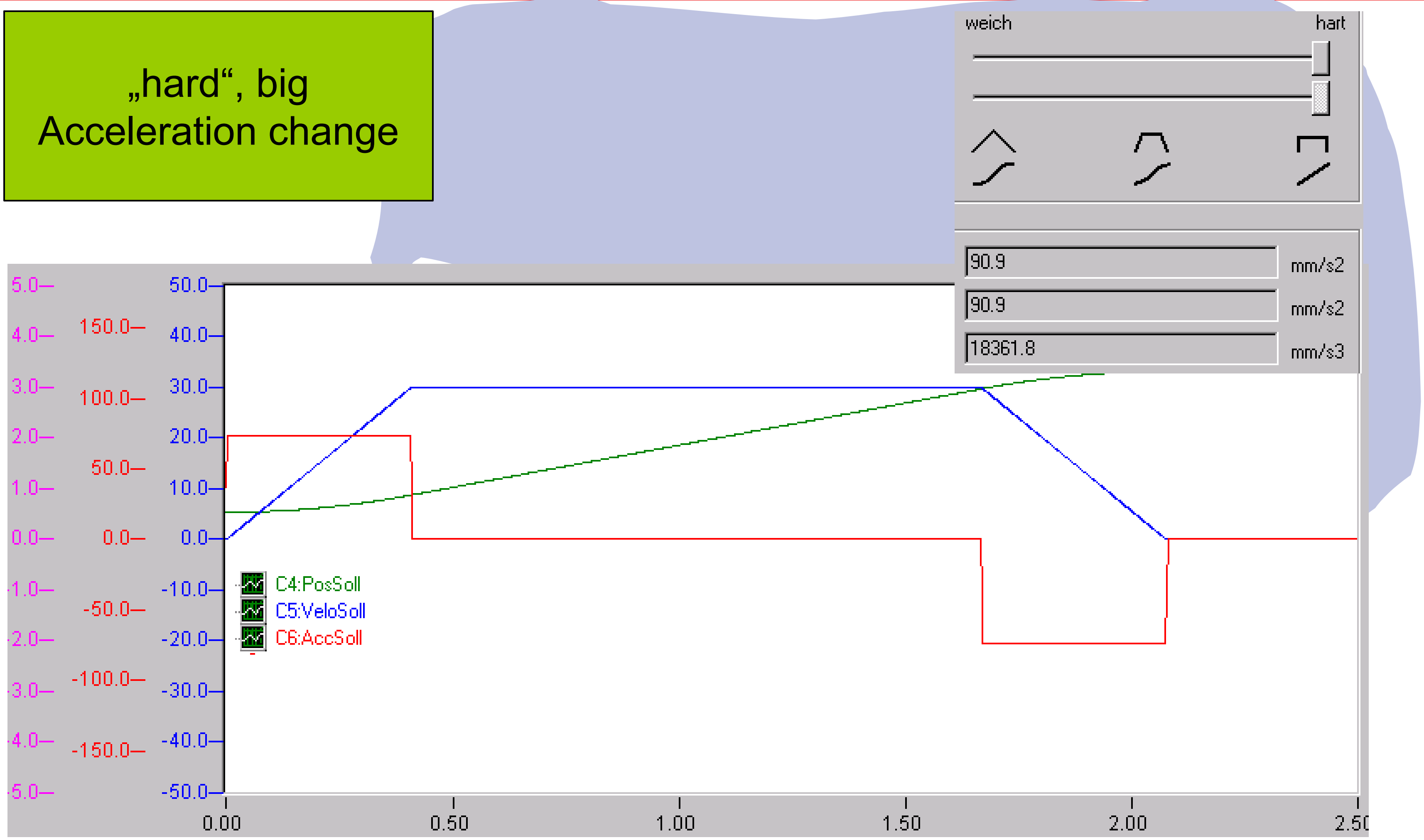


Set value profiles

The profile of the velocity output can be variied during an defined brake time
Thereby the acceleration change (jerk) can be reduced considerably.
This works out on in the short run mechanical burdens and commensurate with as well on the electric burden of the drive.

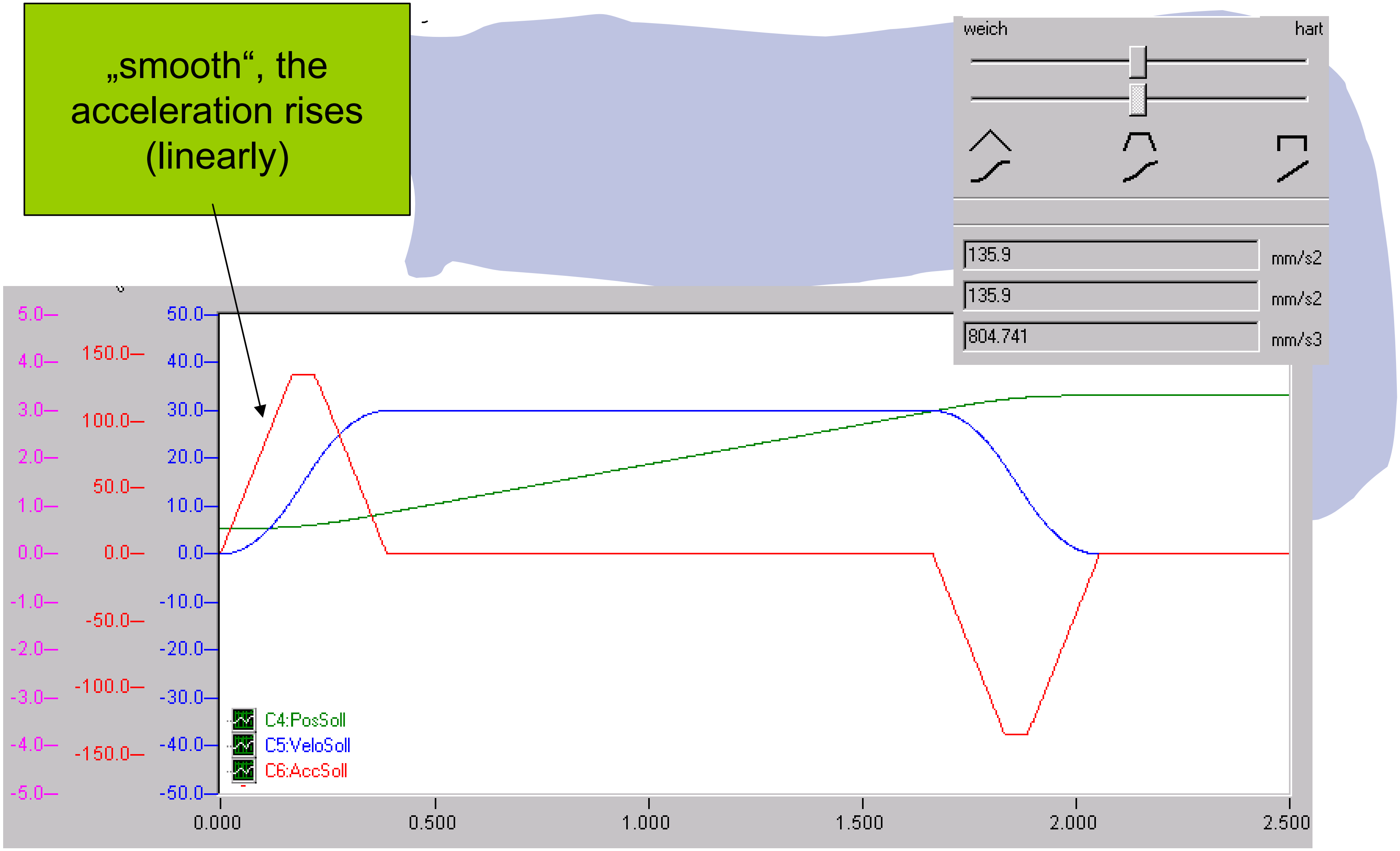
Set value profiles

„hard“, big
Acceleration change



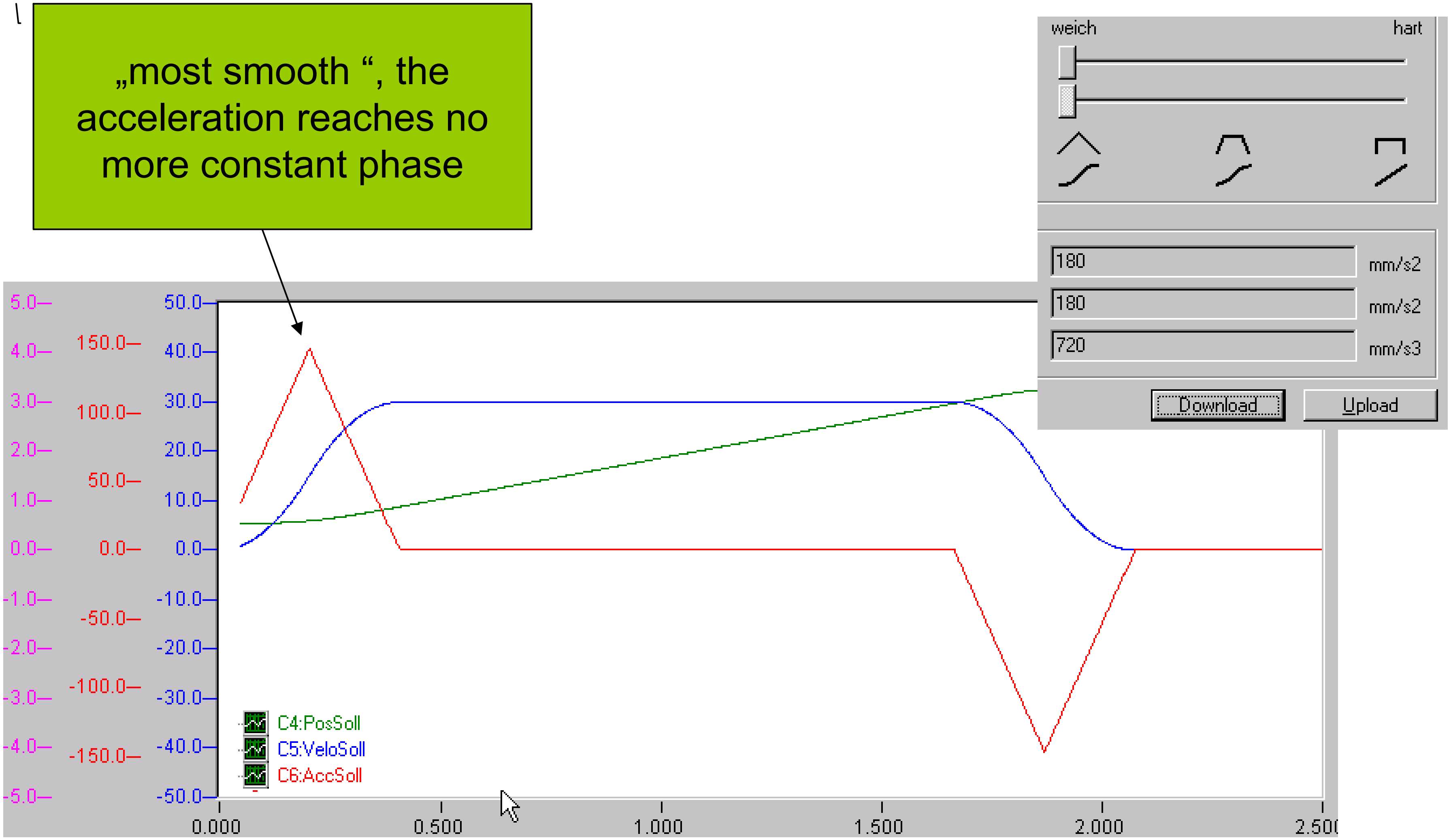
Set value profiles

„smooth“, the acceleration rises (linearly)



Set value profiles

„most smooth “, the acceleration reaches no more constant phase



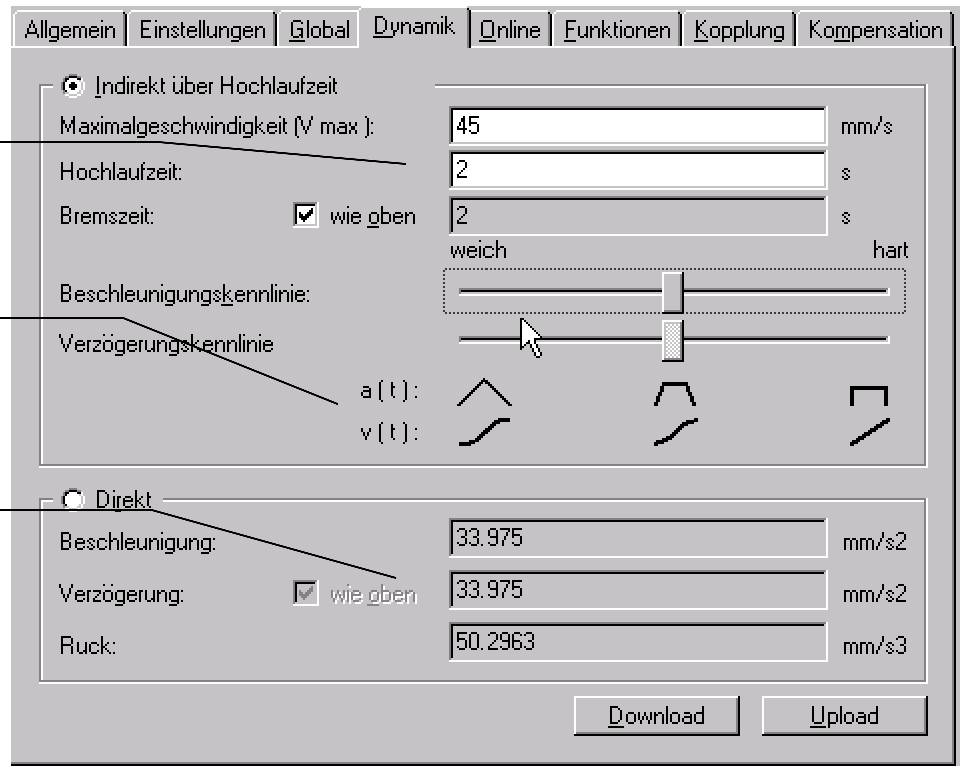
Set value profiles

Die Vorgabe kann sehr einfach über die Vorgabe der Hochlaufzeit und der Auswahl des Profils im System Manager erfolgen!

Input via run-up time

Preselect profile

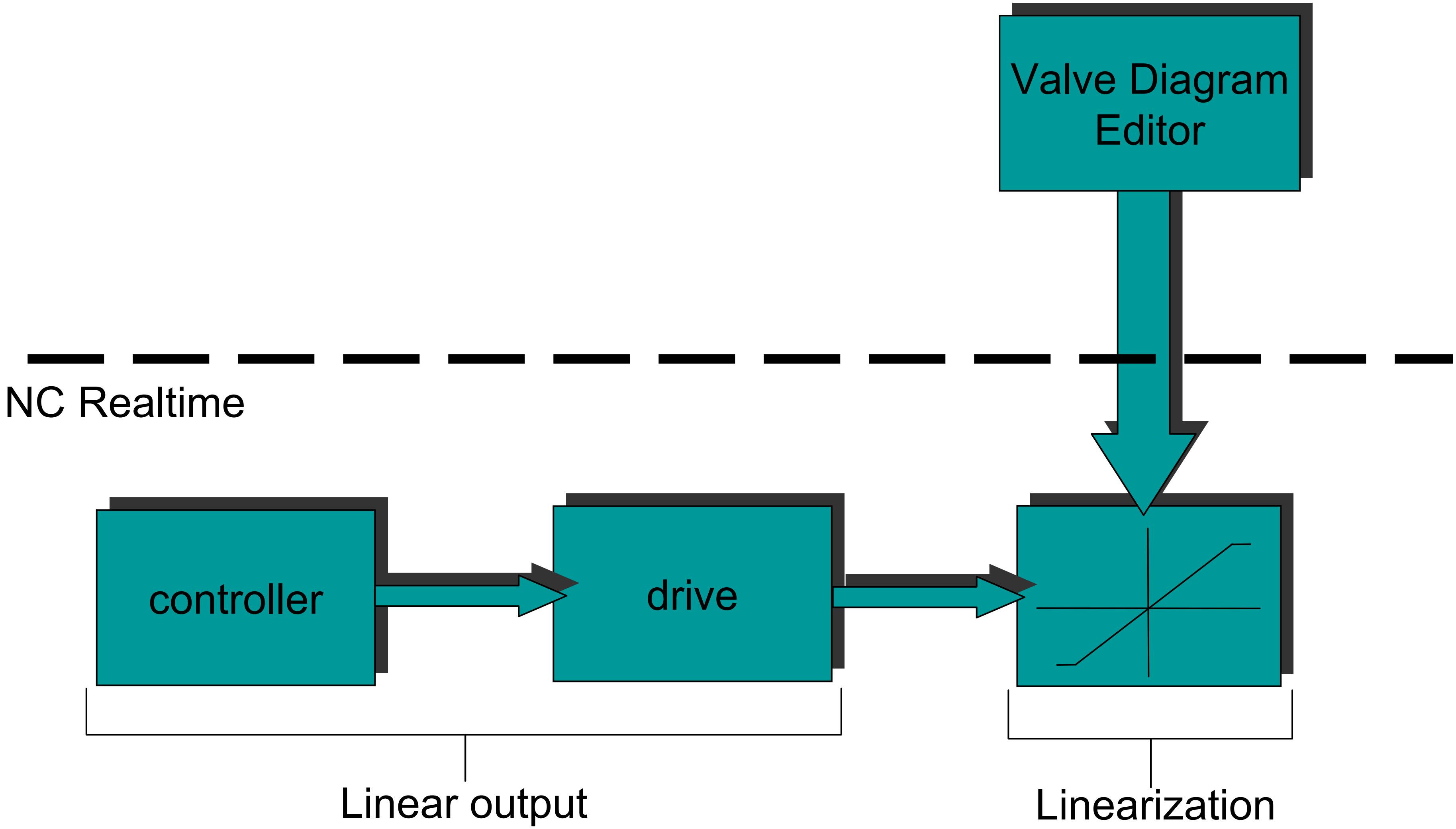
Calculation by the TwinCAT System Manager



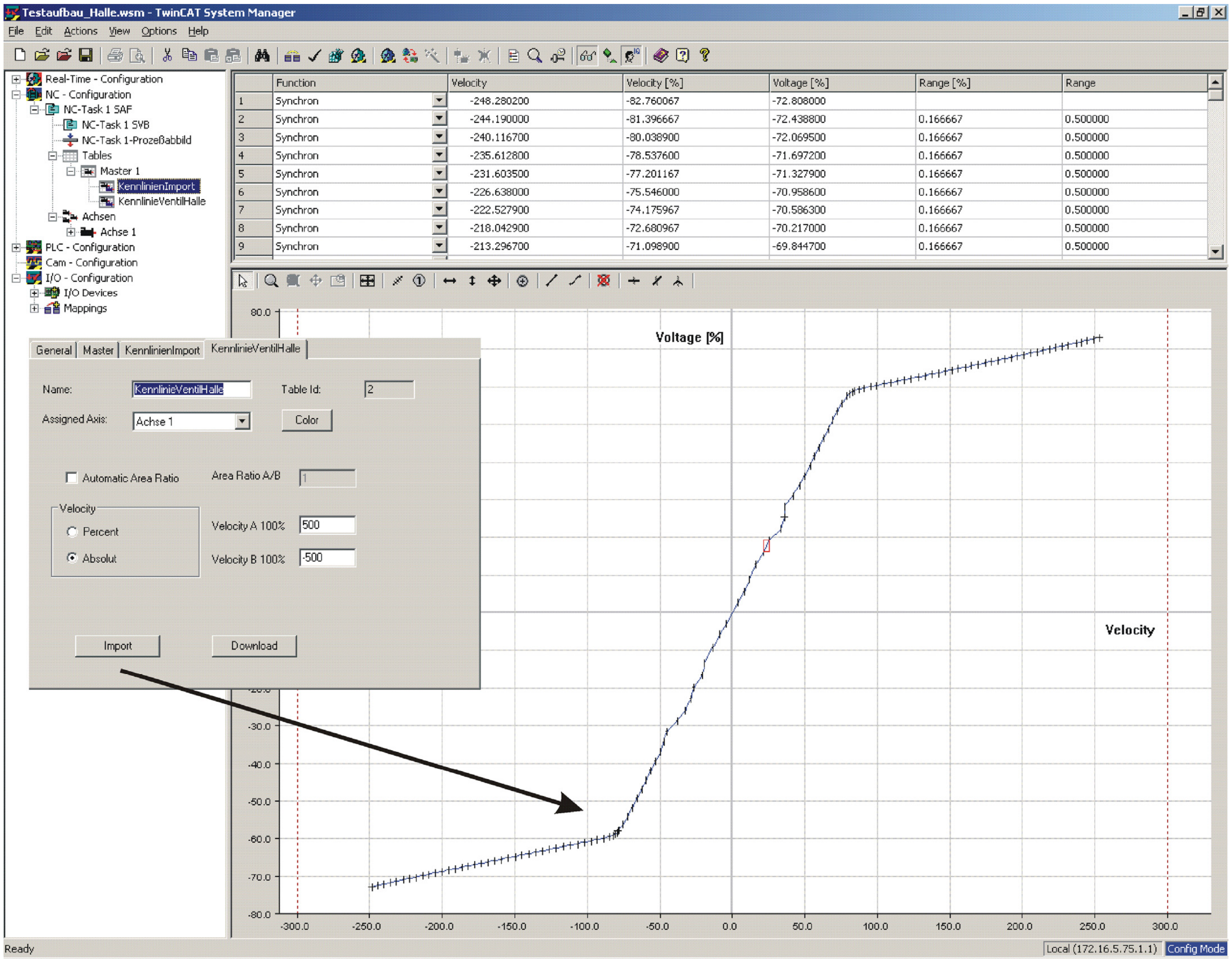
Output Linearisation: TwinCAT Valve Diagram Editor

- Problem: non linear charcateristic curves of valves
- Solution
- Measurement of the curves with the PLC Program
- import the values to the System Manager
- graphical Linearisation
- Interpolation (Polynomial of 5th degree)
- Load to NC Outputs are linearized

TwinCAT Valve Diagram Editor



TwinCAT Valve Diagram Editor

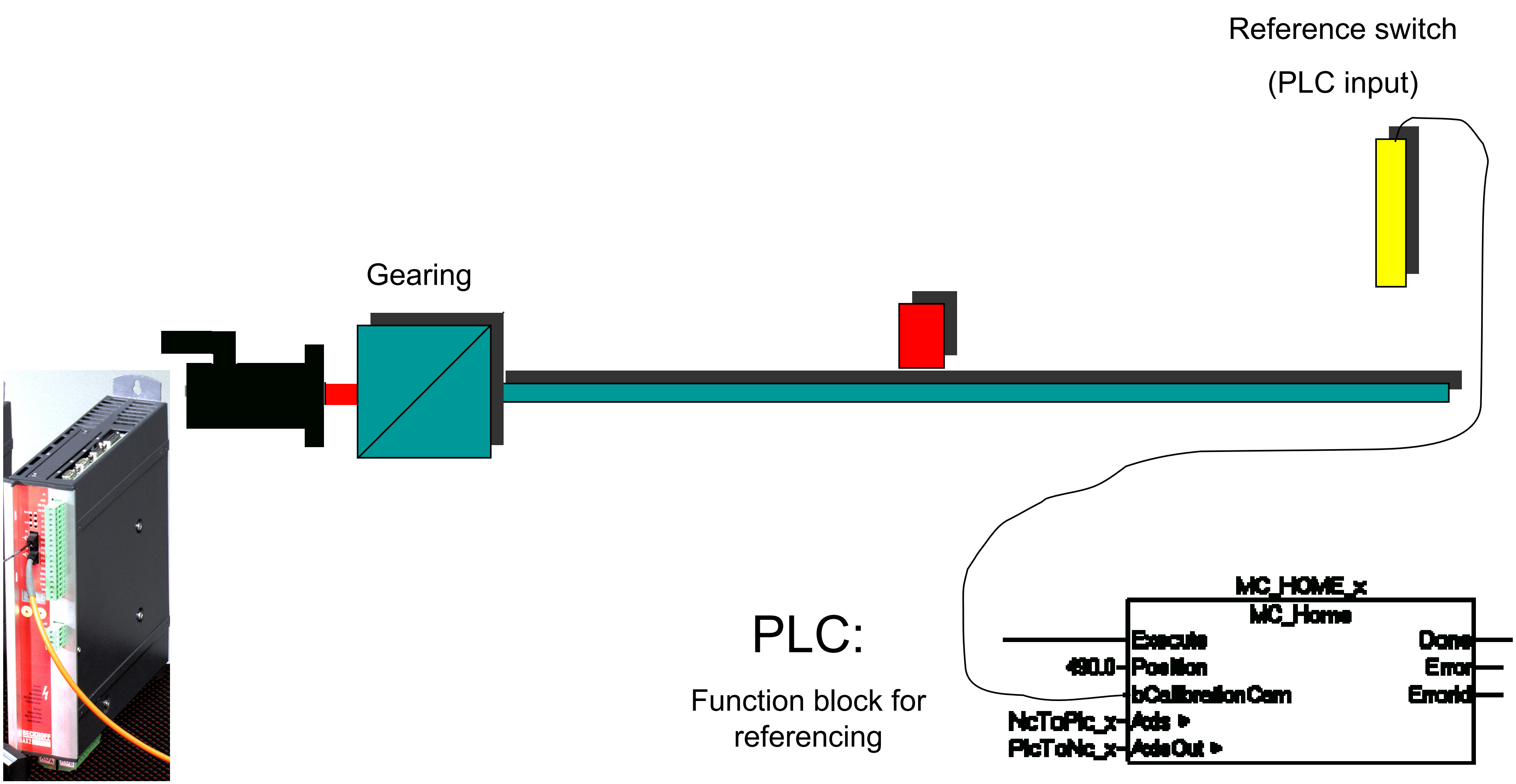


Referencing

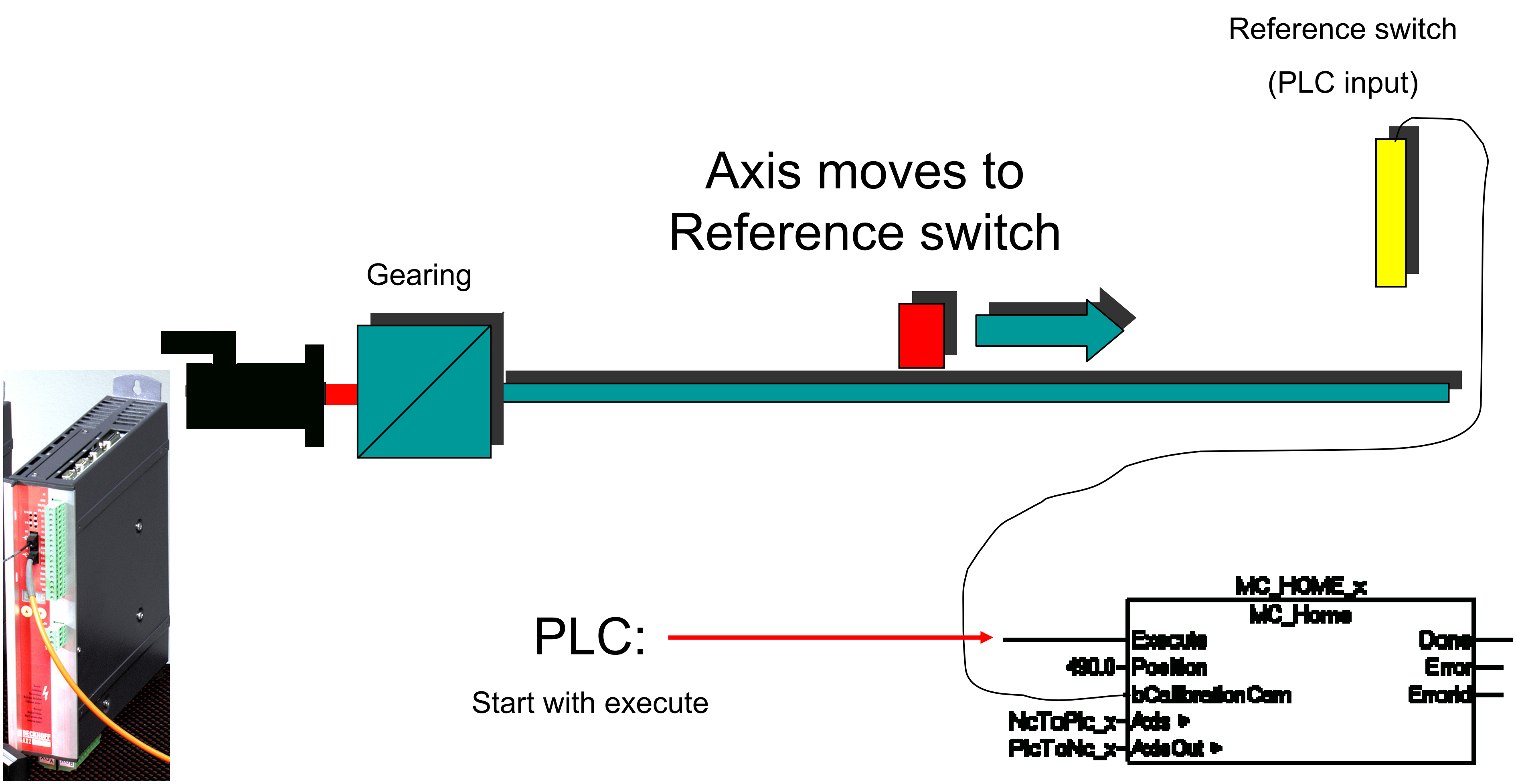
Referencing (calibrate) is necessary for axis with not absolute encoder systems. Incremental Encoder, Single Turn Absolute Encoder, or not absolute encoder systems direct from the drive, (e.g. actual position value of AX2000).

At referencing the axis is lead to a fix reference position and the encoder is set to the current actual position.

Referencing initial state

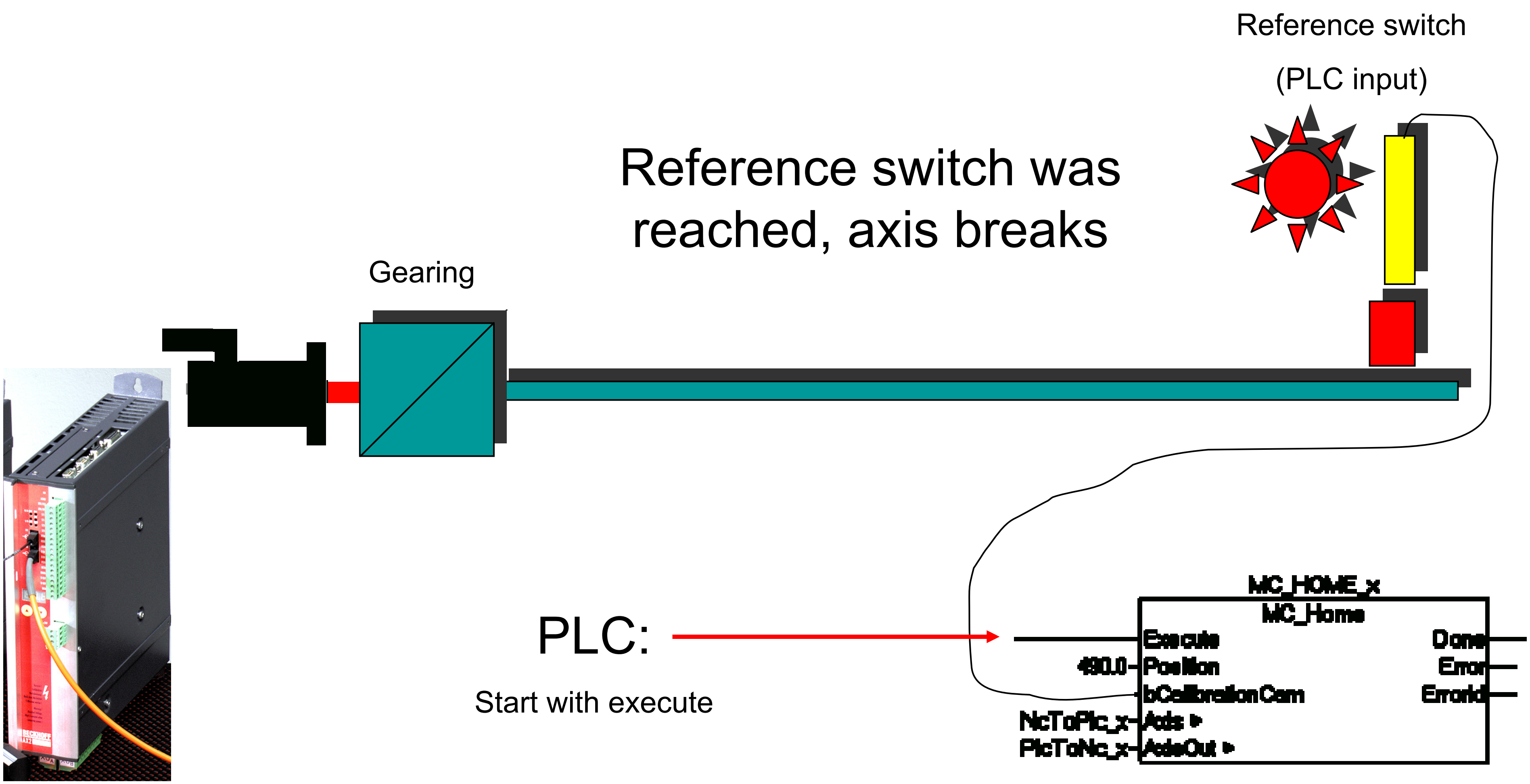


Referencing



Referencing

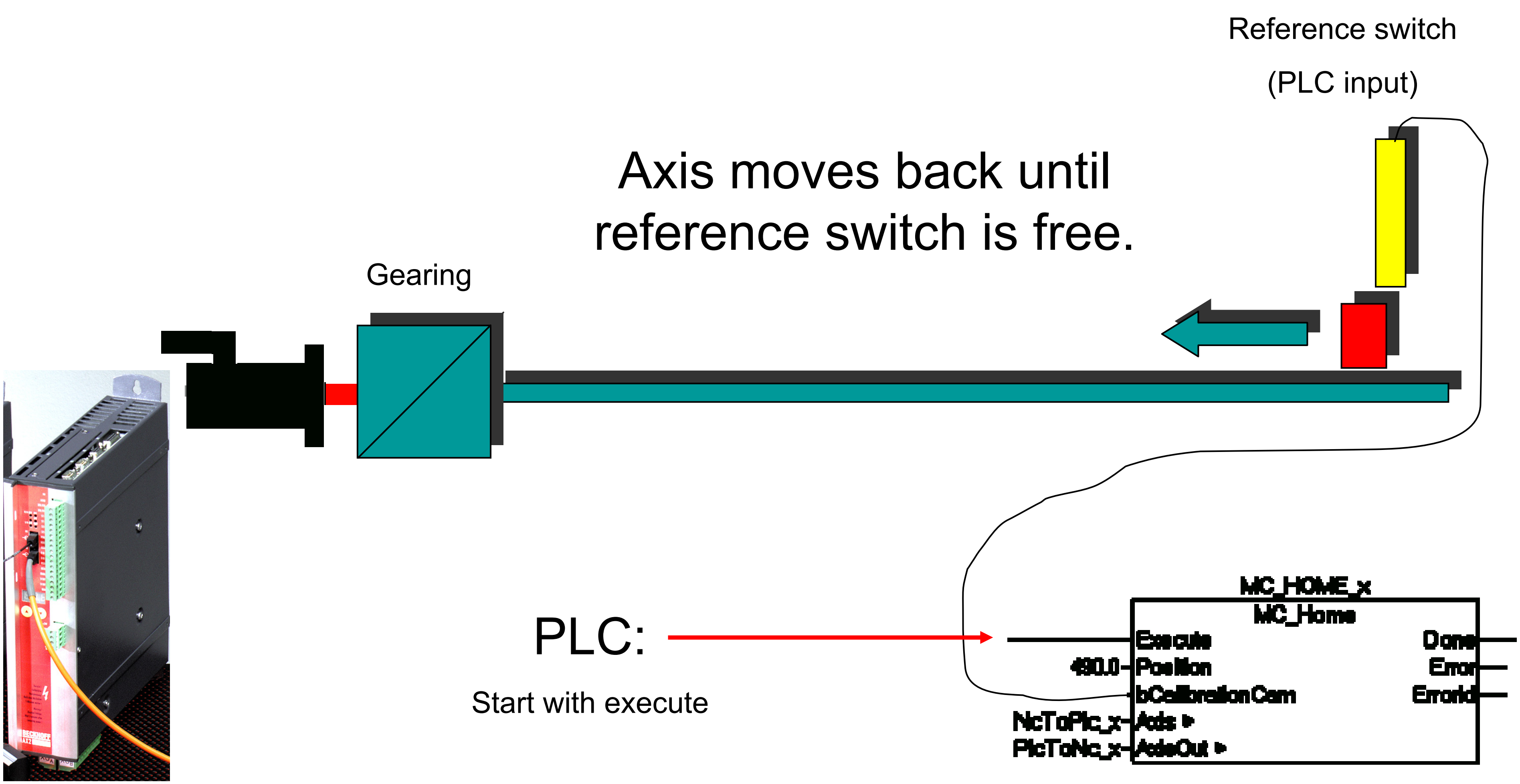
Reference switch was reached, axis breaks



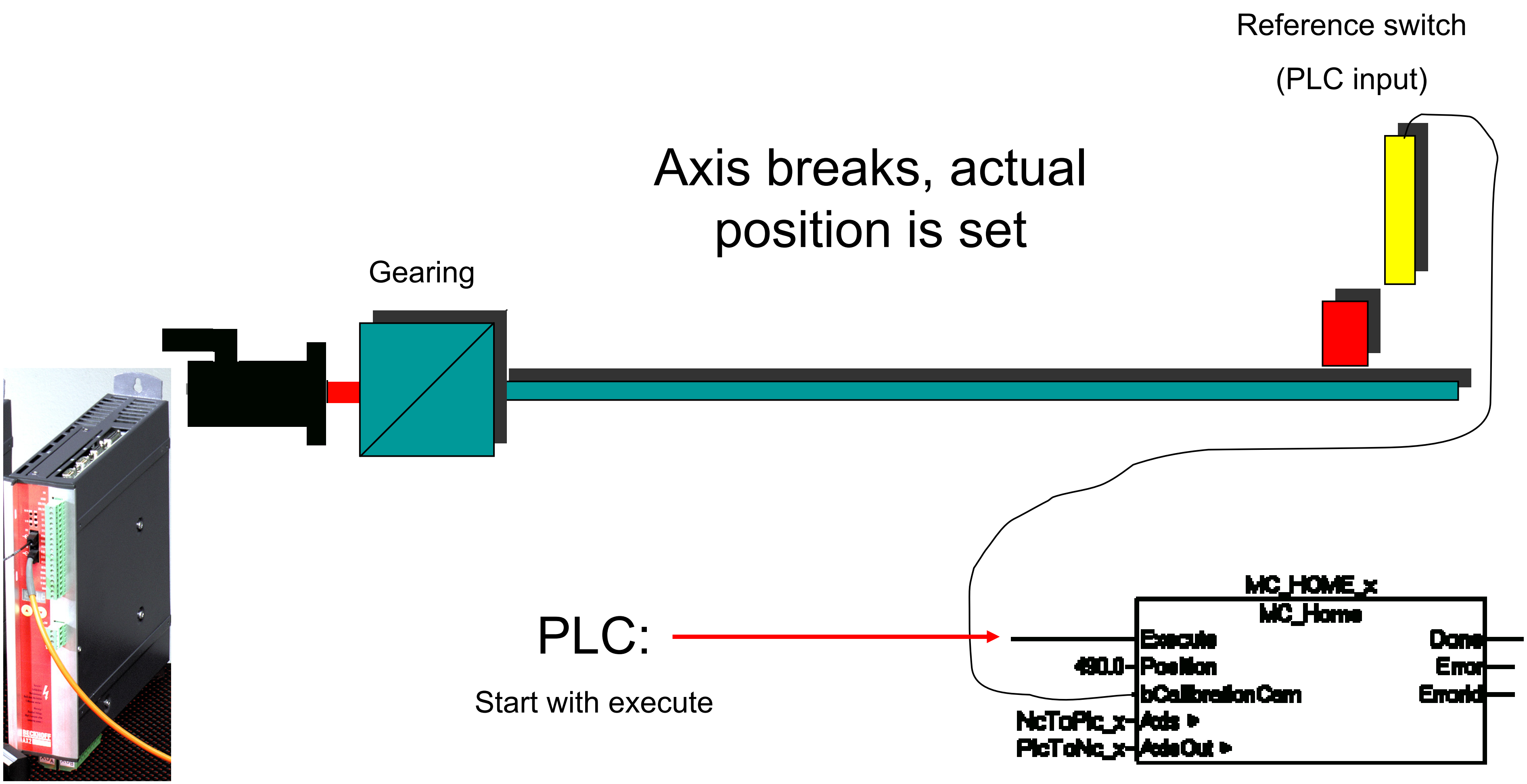
PLC:

Start with execute

Referencing



Referencing completed (a)



Referencing completed (b)

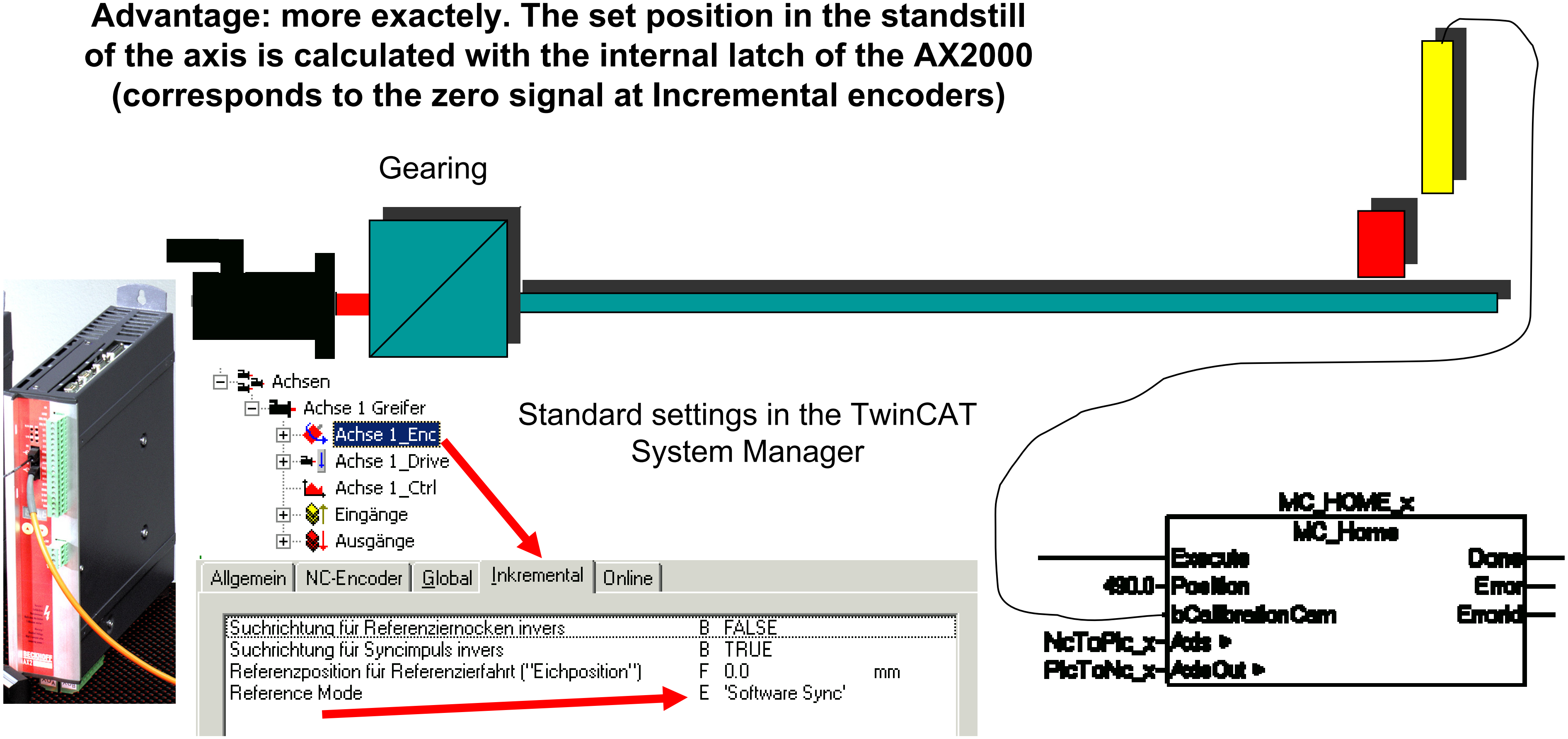
AX2000:
After leaving the reference switch, TwinCAT NC waits for the "Syncsignal" of AX2000 and then stops.

Advantage: more exactely. The set position in the standstill of the axis is calculated with the internal latch of the AX2000 (corresponds to the zero signal at Incremental encoders)

Reference switch

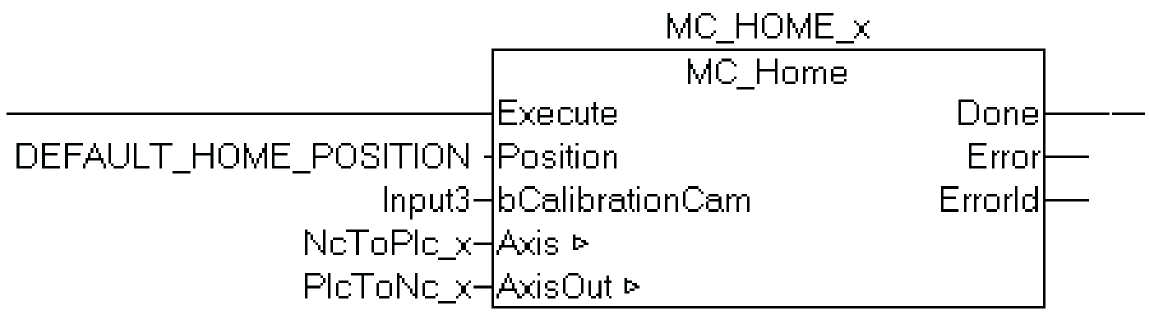
(PLC input)

Gearing



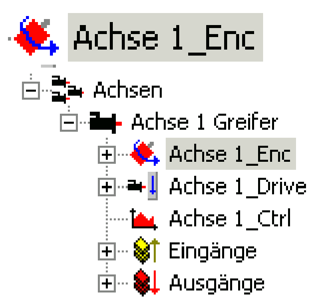
Standard settings in the TwinCAT System Manager

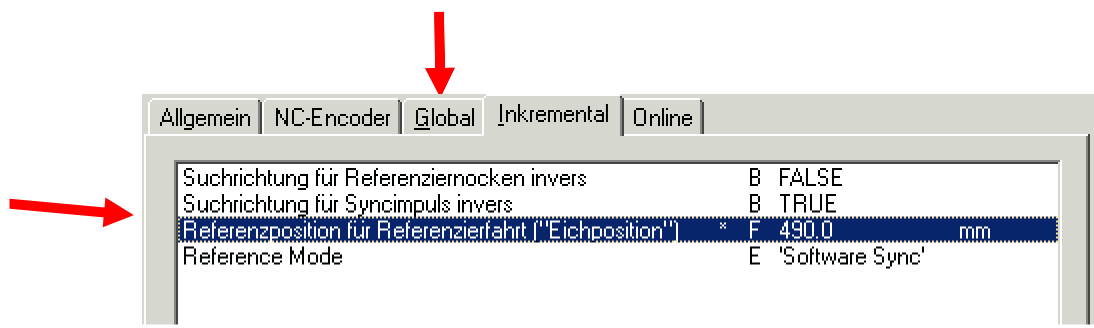
Referencing completed. Which position is set?



If „Position“DEFAULT_HOME_POSITION (global variable from TCMC.LIB) is submitted at the Fb input, the value is taken out of the System Manager.

Otherwise the value ist taken at the input „Position“





Motion Control Function blocks

- •Part I General
- •Overview
- •Axis types
- •Functional principle
- •Referencing
- •Motion Control Function Blocks

- •Teil II Practical Part:
- •Setting up NC axes in the System Manager
- •Starting NC axes from the PLC

Target: IEC61131-3 compatible programming interface for motion tasks

Beckhoff:

Motion Control Function blocks

Why a standard?

Beckhoff:

Beispiel : es müssen nicht unbedingt alle FB‘s aus der spec vorhanden sein

-Hardware independent Programming

-the same look and feel, identical Syntax

-IEC 61131-3 as Base

-Expansions for new application areas possible

-TwinCAT: Combination of MC blocks and TwinCAT specific Axis blocks possible.

⇒ Existing applications can be expanded with Motion Control blocks, without a new writing of the existing flows.

Beckhoff:

Motion Control Function blocks

Defined in:

The PLCopen Task Force Motion Control
by Manufacturer and end user

- **Atlas Copco Control**
- **Baumueller**
- **Beckhoff**
- **Control Techniques**
- **Elau**
- **Giddings & Lewis**
- **Indramat**
- **Infoteam Software**
- **KW Software**
- **Lenze**
- **Siemens**
- **Softing**

TetraPak

Rovema Packaging Machines

Ford

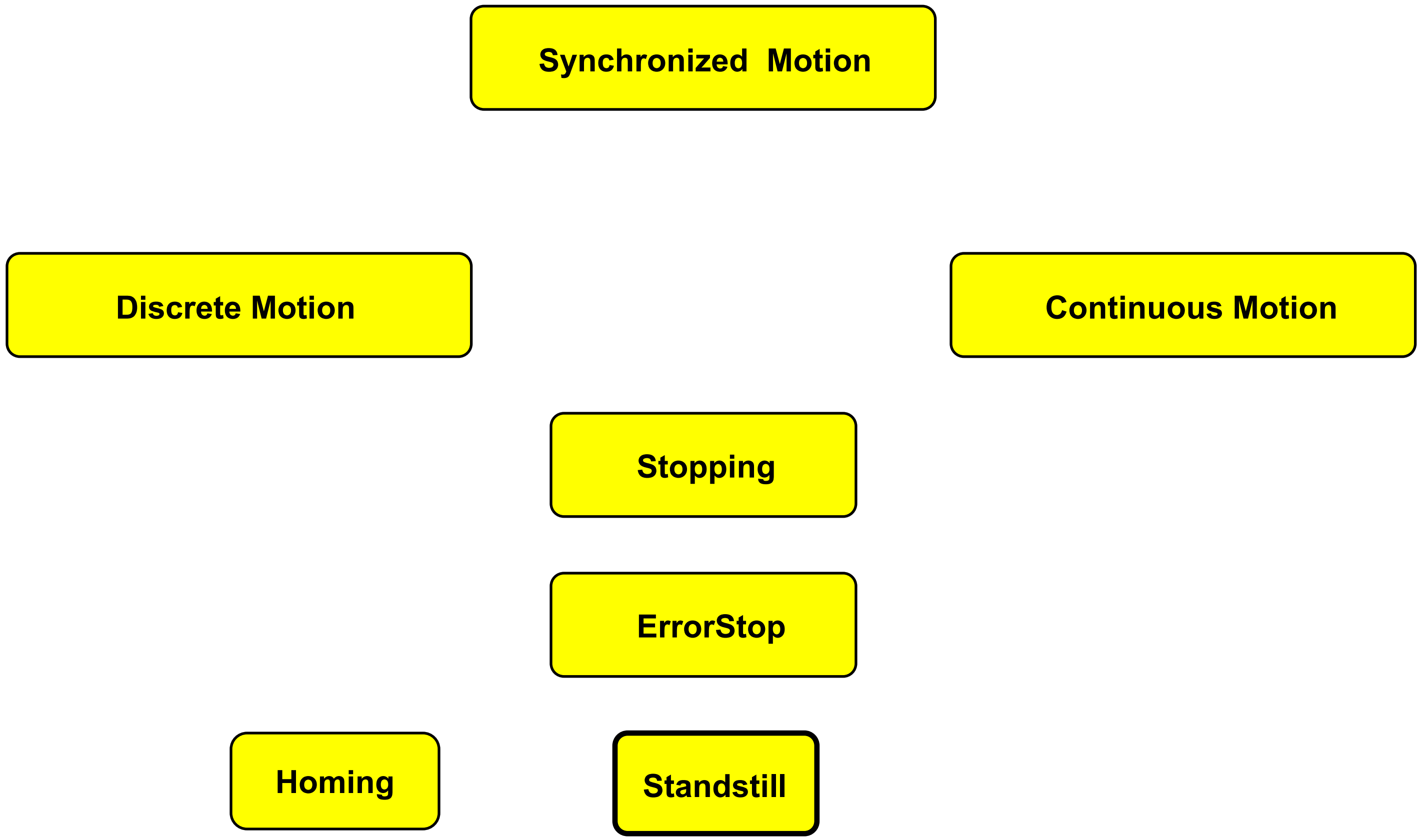
General Motors

Root: Task Force Motion Control presentation Version Febr2002. (www.plcopen.org)

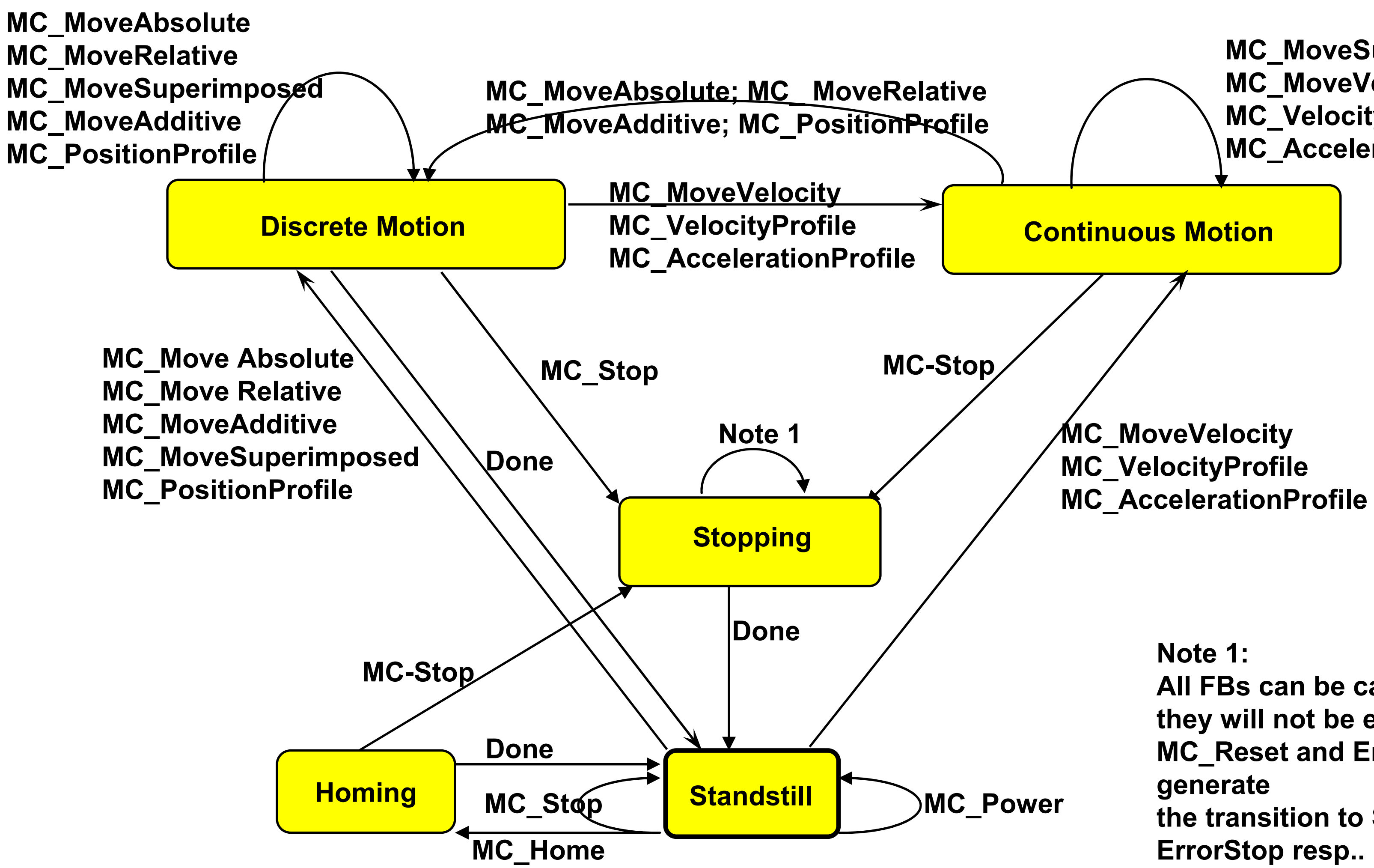
Beckhoff:

Beispiel : es müssen nicht unbedingt alle FB's aus der spec vorhanden sein

Statemachine:



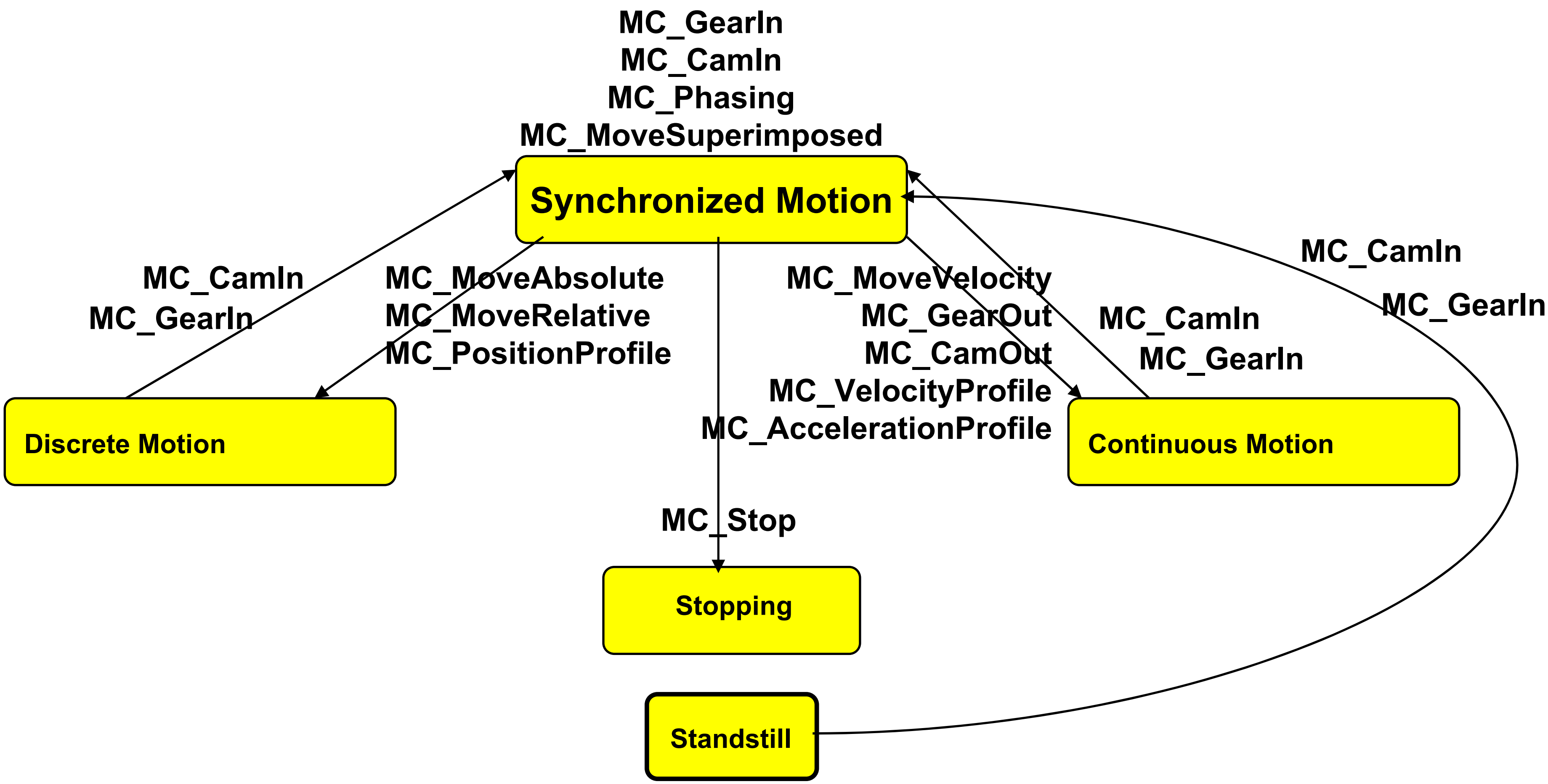
Statemachine:



Note 1:
All FBs can be called, although they will not be executed, except MC_Reset and Error– will generate the transition to StandStill or ErrorStop resp..

- Root: Task Force Motion Control presentation Version Febr2002. (www.plcopen.org)

Statemachine Synchronized Motion



Overview Function Block Class:

Administrative

Single Axes

- *Power*
- *Reset*
- *ReadStatus*
- *ReadAxisError*
- *ReadParameter*
- *WriteParameter*
- *ReadActualPosition*

Multiple Axes

- *CamTableSelect*

Motion

Single Axes

- *MoveAbsolute*
- *MoveRelative*
- *MoveAdditive*
- *MoveSuperImposed*
- *MoveContinuous*
- *MoveVelocity*
- *Home*
- *Stop*
- *PositionProfile*
- *VelocityProfile*
- *AccelerationProfile*

Multiple Axes

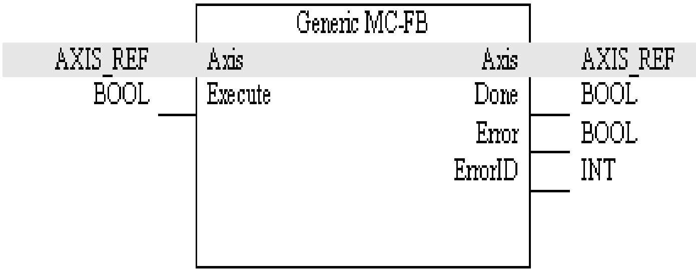
Non-Interpolated

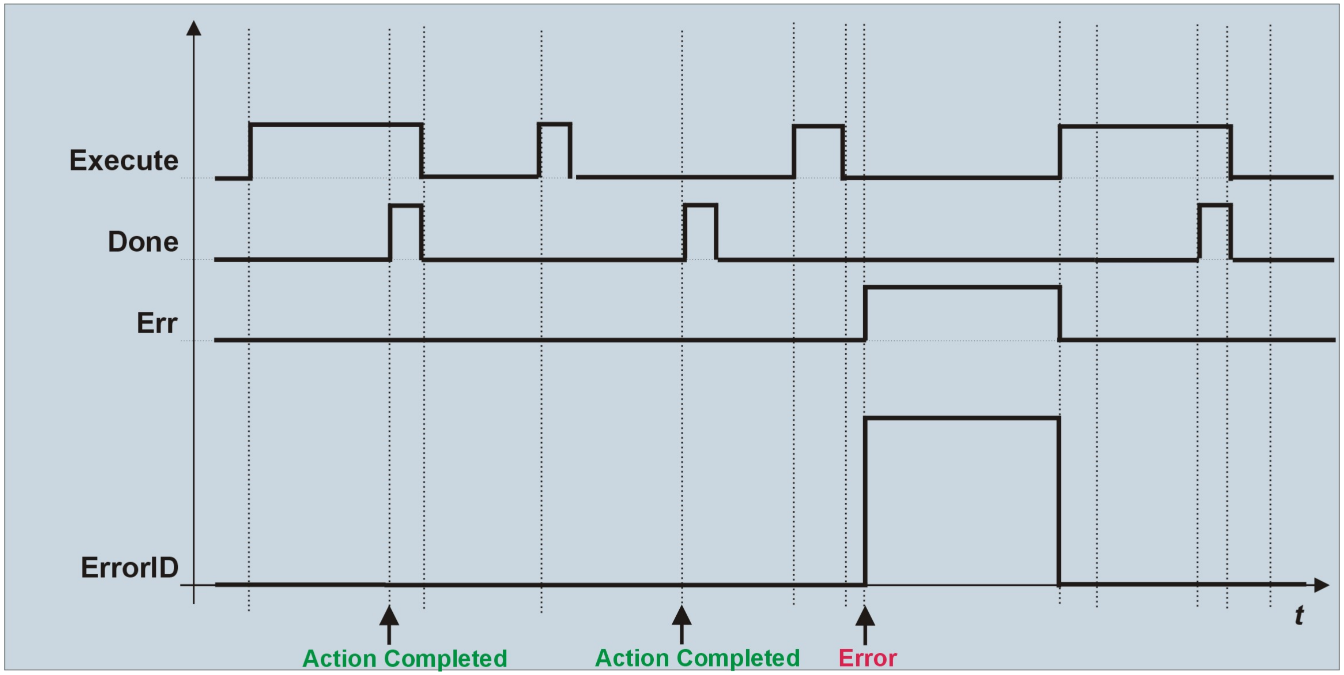
- *CamIn*
- *CamOut*
- *GearIn*
- *GearOut*
- *Phasing*

Interpolated

- …

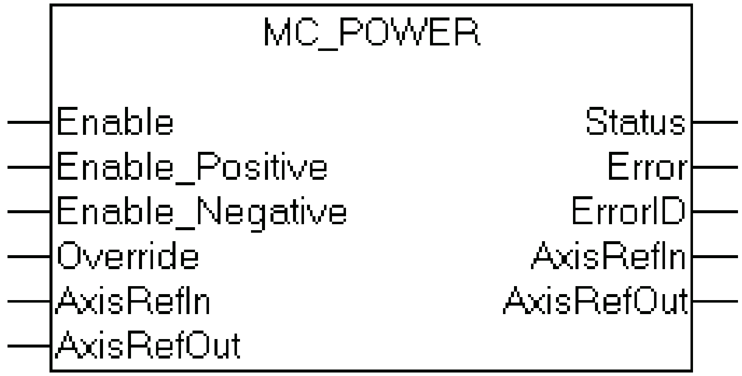
Standardized Handshake





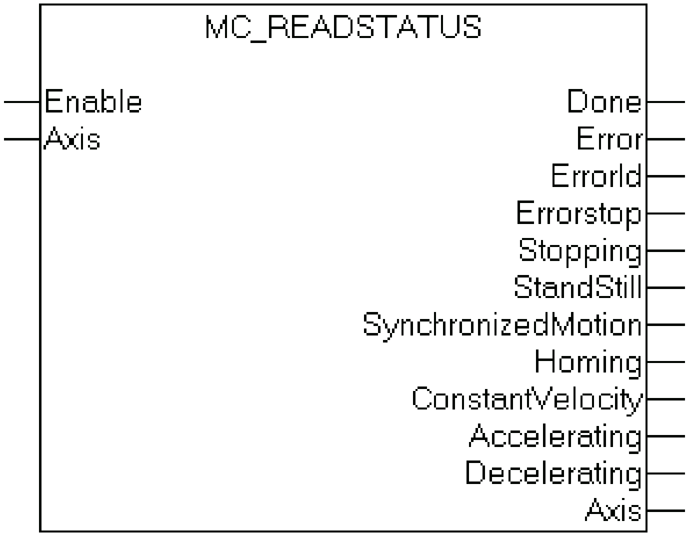
Administrative Function Blocks

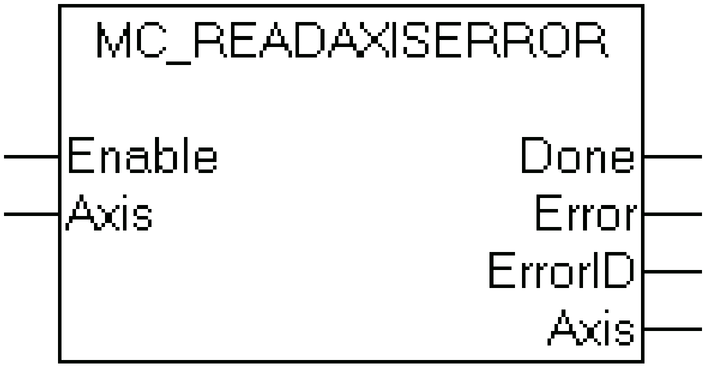
MC Power



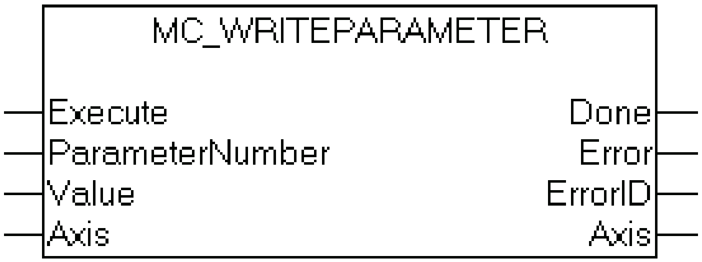
Enable	Enable_Positive	Enable_Negative	NC Controller allows:
1	0	0	Position control
1	1	0	Position control + Start in positive direction
1	0	1	Position control + Start in negative direction
1	1	1	Position control + Start in positive or negative direction

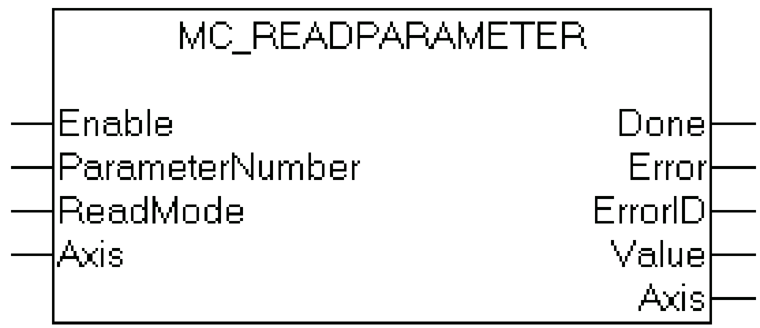
MC Read_...


MC_READSTATUS
Enable
Axis
Done
Error
ErrorId
Errorstop
Stopping
StandStill
SynchronizedMotion
Homing
ConstantVelocity
Accelerating
Decelerating
Axis

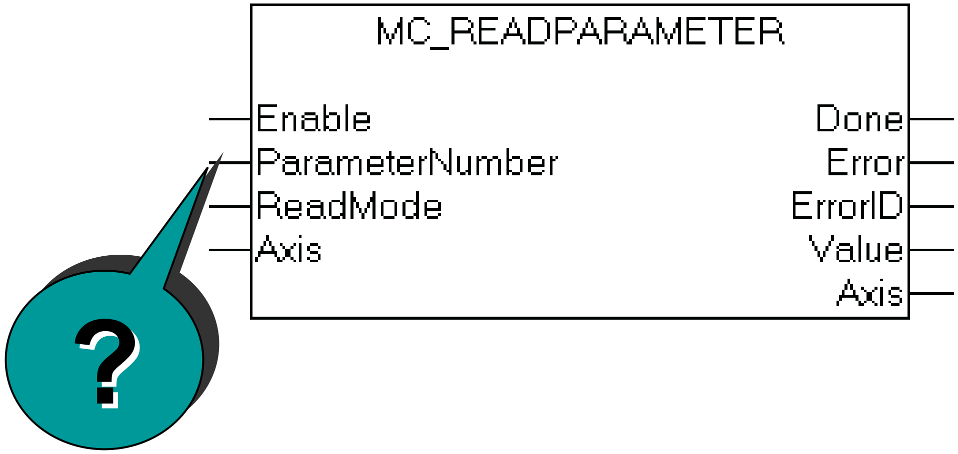

MC_READAXISERROR
Enable
Axis
Done
Error
ErrorID
Axis

MC Read_...

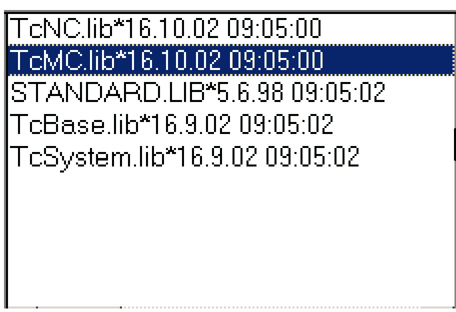

MC_WRITEPARAMETER
Execute
ParameterNumber
Value
Axis
Done
Error
ErrorID
Axis

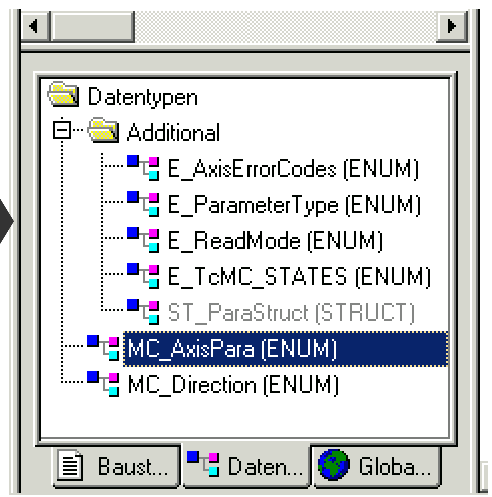

MC_READPARAMETER
Enable
ParameterNumber
ReadMode
Axis
Done
Error
ErrorID
Value
Axis

MC Read /Write Parameter Number in TCMC.LIB



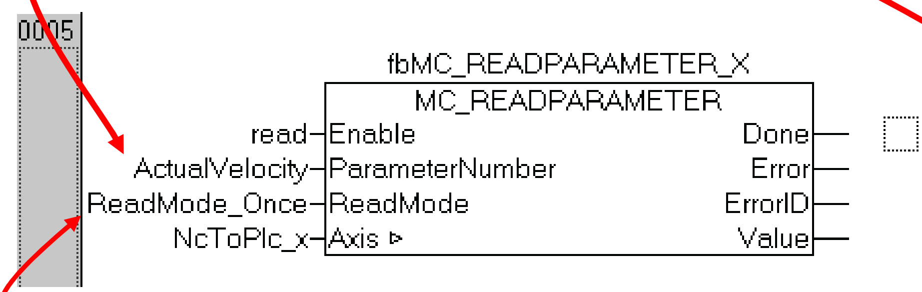
PLC Control Library Manager






```
TYPE MC_AxisPara : (
(*                 PLCopen specific parameters *)
                   CommandedPosition := 1,    (* lreal *)
                   SWLimitPos,                (* lreal *)
                   SWLimitNeg,                (* lreal *)
                   EnableLimitPos,            (* bool *)
                   EnableLimitNeg,            (* bool *)
                   EnablePosLagMonitoring,    (* bool *)
                   MaxPositionLag,            (* lreal *)
                   MaxVelocitySystem,         (* lreal *)
                   MaxVelocityAppl,           (* lreal *)
                   ActualVelocity,            (* lreal *)
                   CommandedVelocity,         (* lreal *)
                   MaxAccelerationSystem,     (* lreal *)
                   MaxAccelerationAppl,       (* lreal *)
                   MaxDecelerationSystem,     (* lreal *)
                   MaxDecelerationAppl,       (* lreal *)
                   MaxJerk,                   (* lreal *)
(*                 Beckhoff specific parameters *)
                   AxisId     :=     1000,    (* lreal *)
                   AxisVeloManSlow,           (* lreal *)
                   AxisVeloManFast            (* lreal *)
```

Example Read ActualVelocity




```
TYPE MC_AxisPara : (
(*                    PLCopen specific parameters *)
                      CommandedPosition := 1,    (* lreal *)
                      SWLimitPos,                (* lreal *)
                      SWLimitNeg,                (* lreal *)
                      EnableLimitPos,            (* bool *)
                      EnableLimitNeg,            (* bool *)
                      EnablePosLagMonitoring,    (* bool *)
                      MaxPositionLag,            (* lreal *)
                      MaxVelocitySystem,         (* lreal *)
                      MaxVelocityAppl,           (* lreal *)
                      ActualVelocity,            (* lreal *)
                      CommandedVelocity,         (* lreal *)
                      MaxAccelerationSystem,     (* lreal *)
                      MaxAccelerationAppl,       (* lreal *)
                      MaxDecelerationSystem,     (* lreal *)
                      MaxDecelerationAppl,       (* lreal *)
                      MaxJerk,                   (* lreal *)
(*                    Beckhoff specific parameters *)
                      AxisId    :=    1000,      (* lreal *)
                      AxisVeloManSlow,           (* lreal *)
                      AxisVeloManFast            (* lreal *)
```

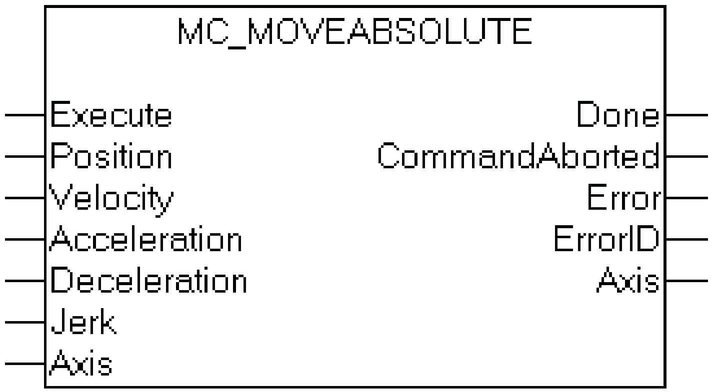
With ReadMode the single resp. permanent reading can be determined.

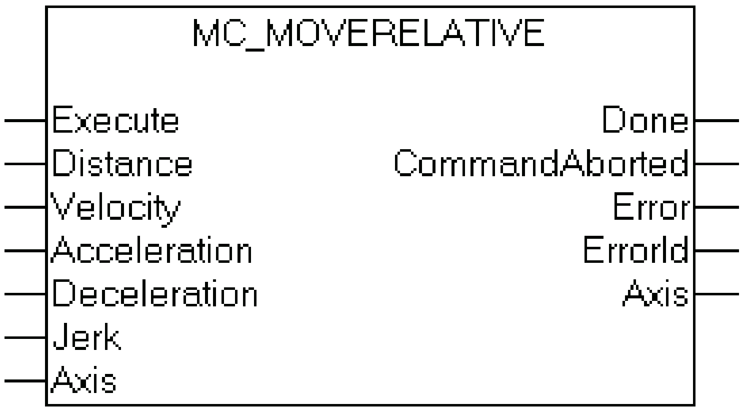
```
TYPE
    E_ReadMode :
        (   ReadMode_Once := 1,
            ReadMode_Cyclic
        );
END_TYPE
```

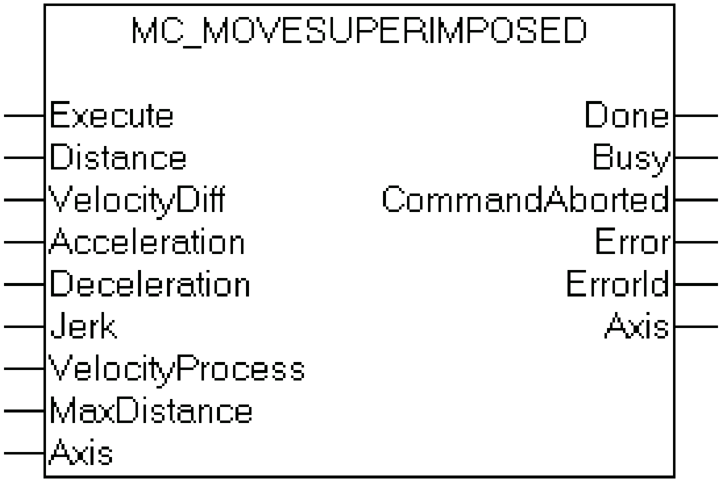
Single Axis

Motion Function Blocks

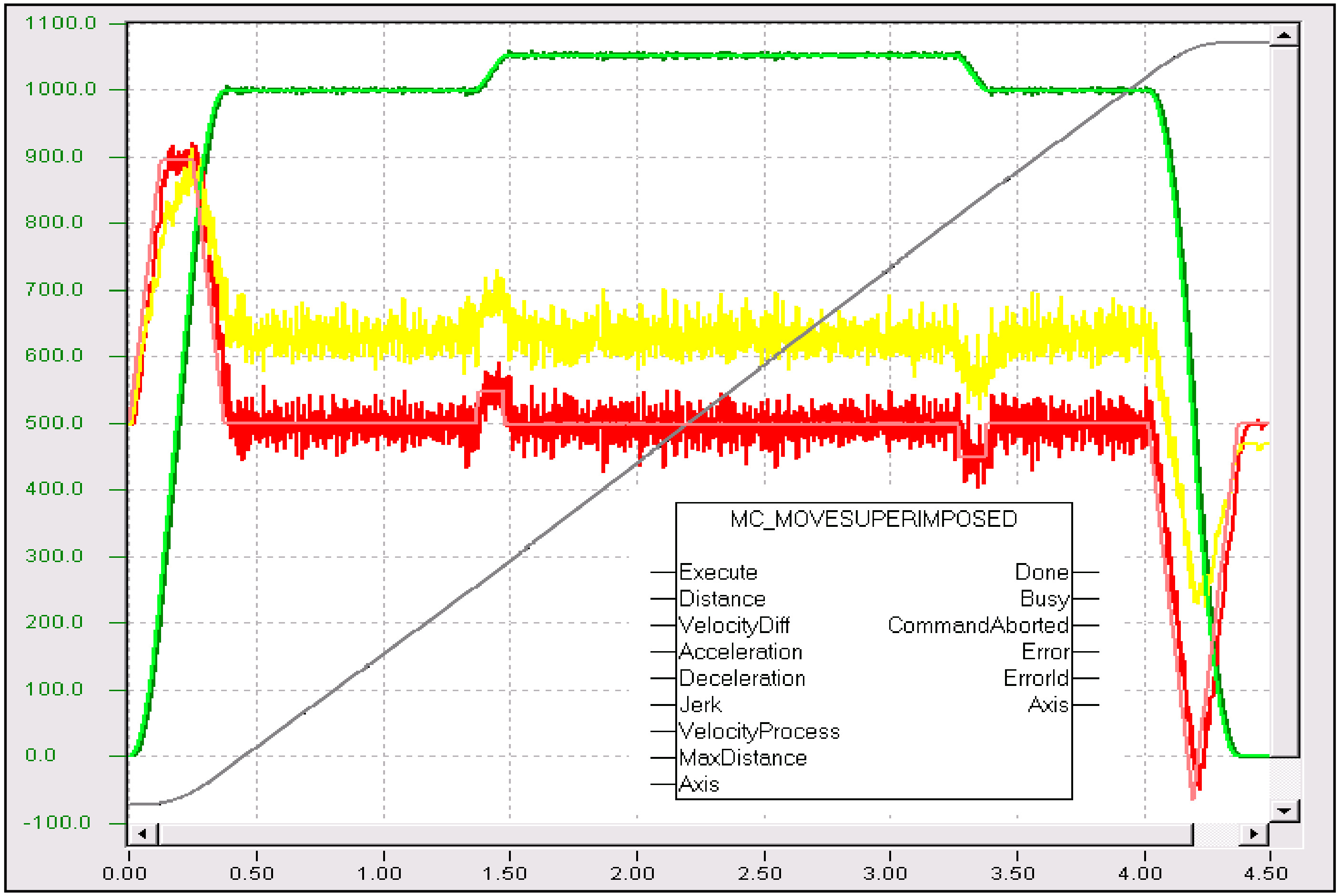
Motion Function Blocks





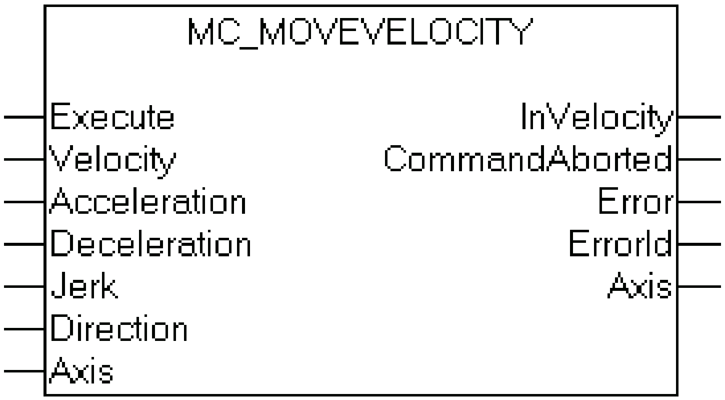


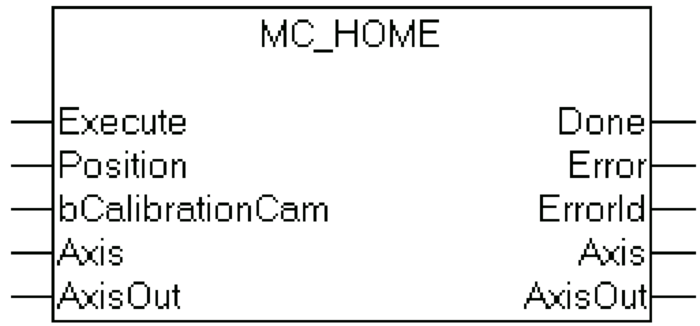
Mode of Operation Move Superimposed



Actual Pos
Actual Velo
Actual Acc
Setpoint Pos
Setpoint Velo
Setpoint Acc

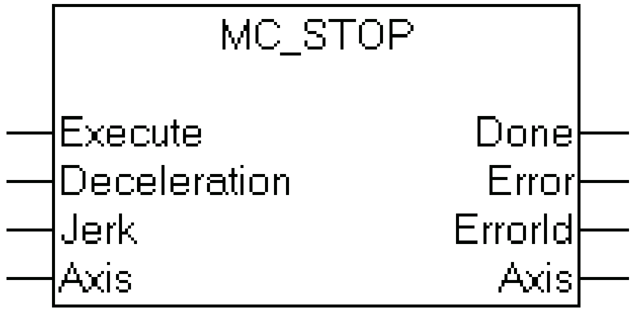
Motion Function Blocks

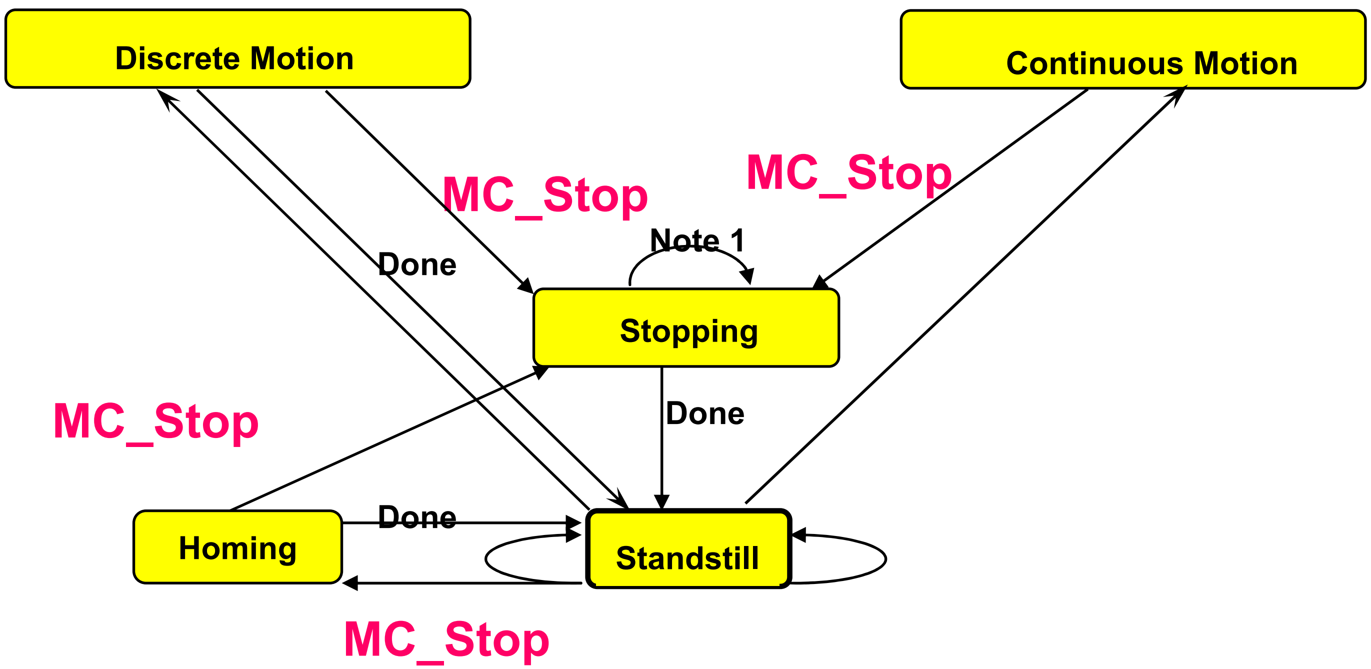




Mode of operation see „Referencing“

Motion Function Blocks





Motion Function Blocks Multiple Axis

Multiple Axis

Motion Function Blocks

(non-interpolated)

Motion Function Blocks Multiple Axis

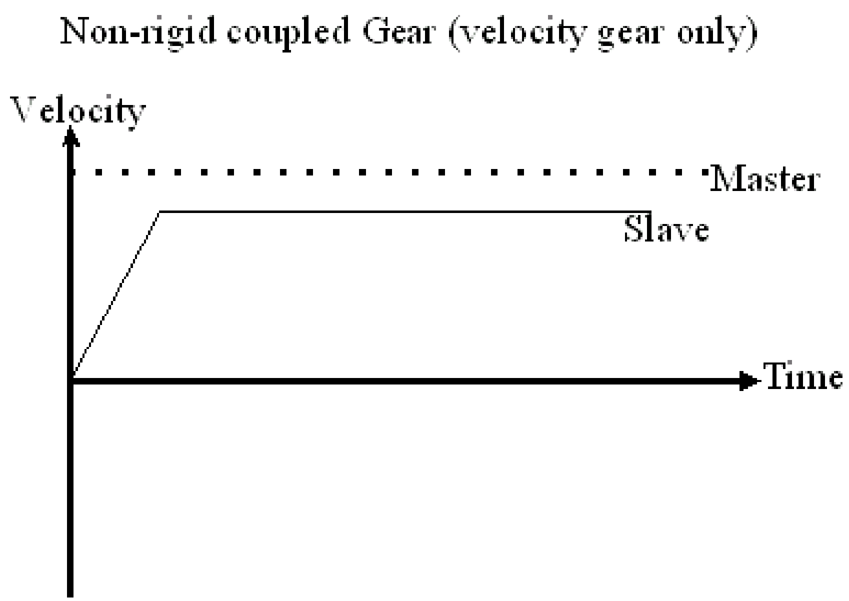
GEARING is the activation of a numeric ratio between master and slave axis.

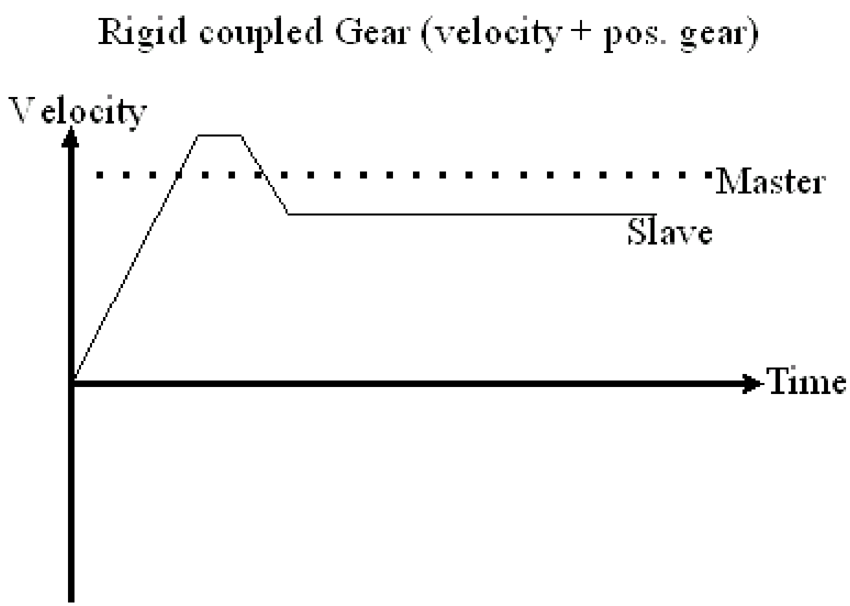
(comparable with a mechanical gearbox).

Motion Function Blocks Multiple Axis

Linear "gearbox"
fixed ratio of transmission : Vm/Vs

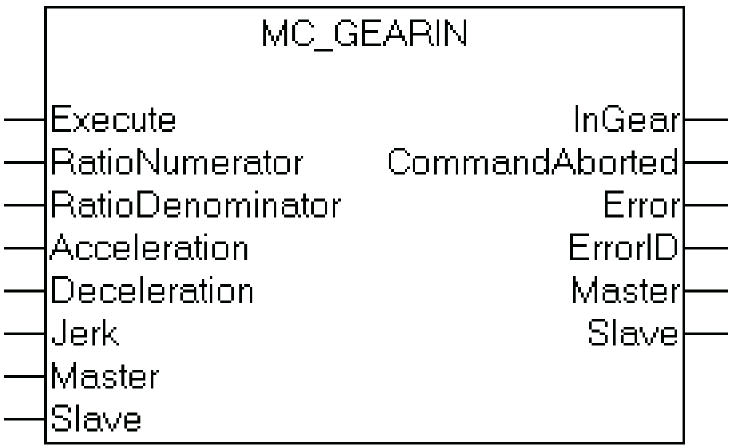
"Flying Saw"

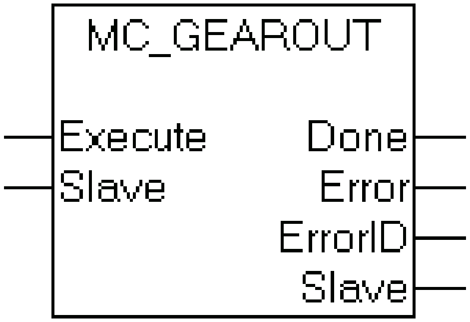




Motion Function Blocks Multiple Axis

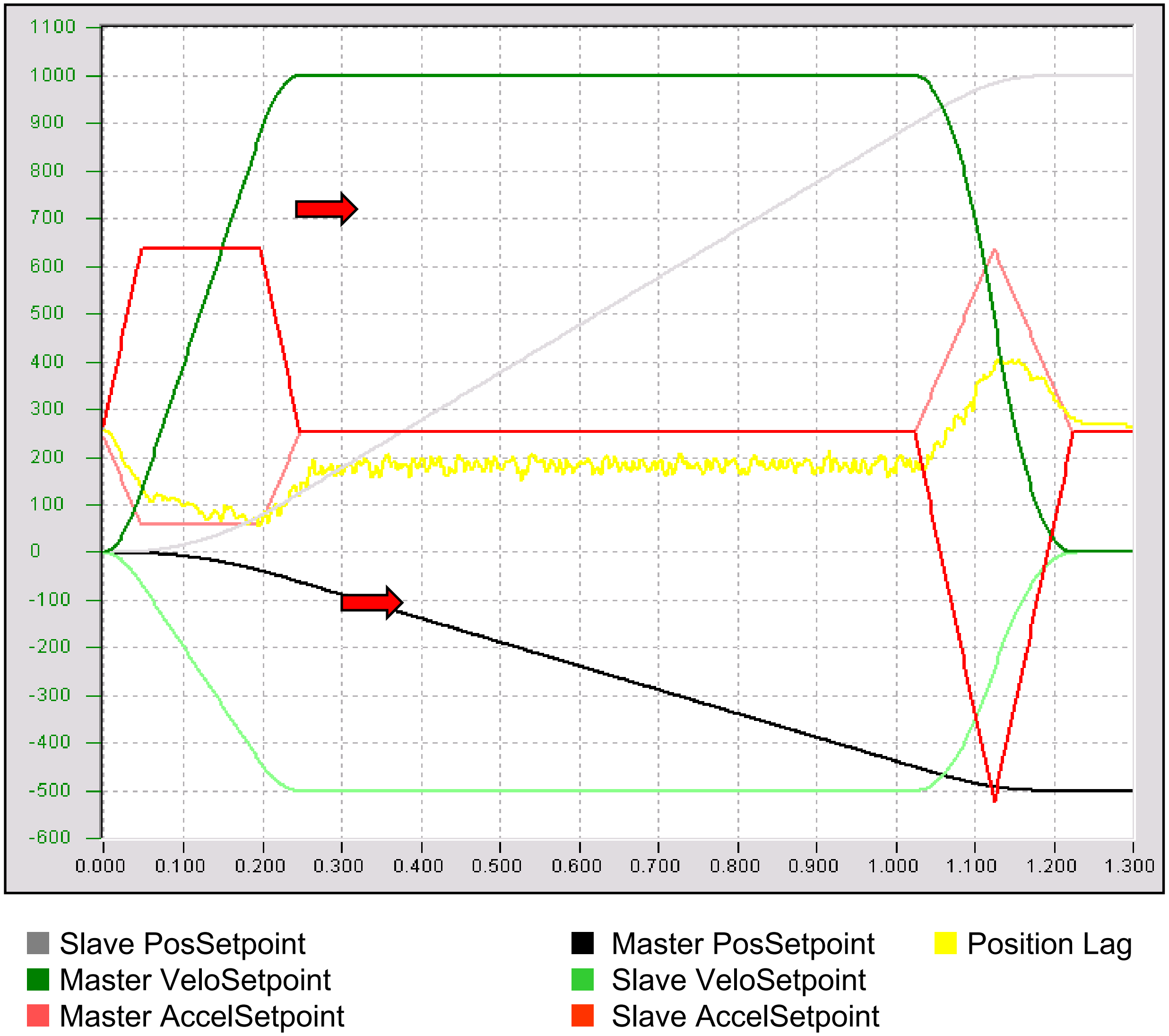
MOTION DIAGRAM FOR GEARING





Motion Function Blocks Multiple Axis

Movement diagram



Setting up NC Axes in System Manager

Note: These bitmaps show all basic steps in the System Manager in for AX2000. Not all possible combinations are shown.

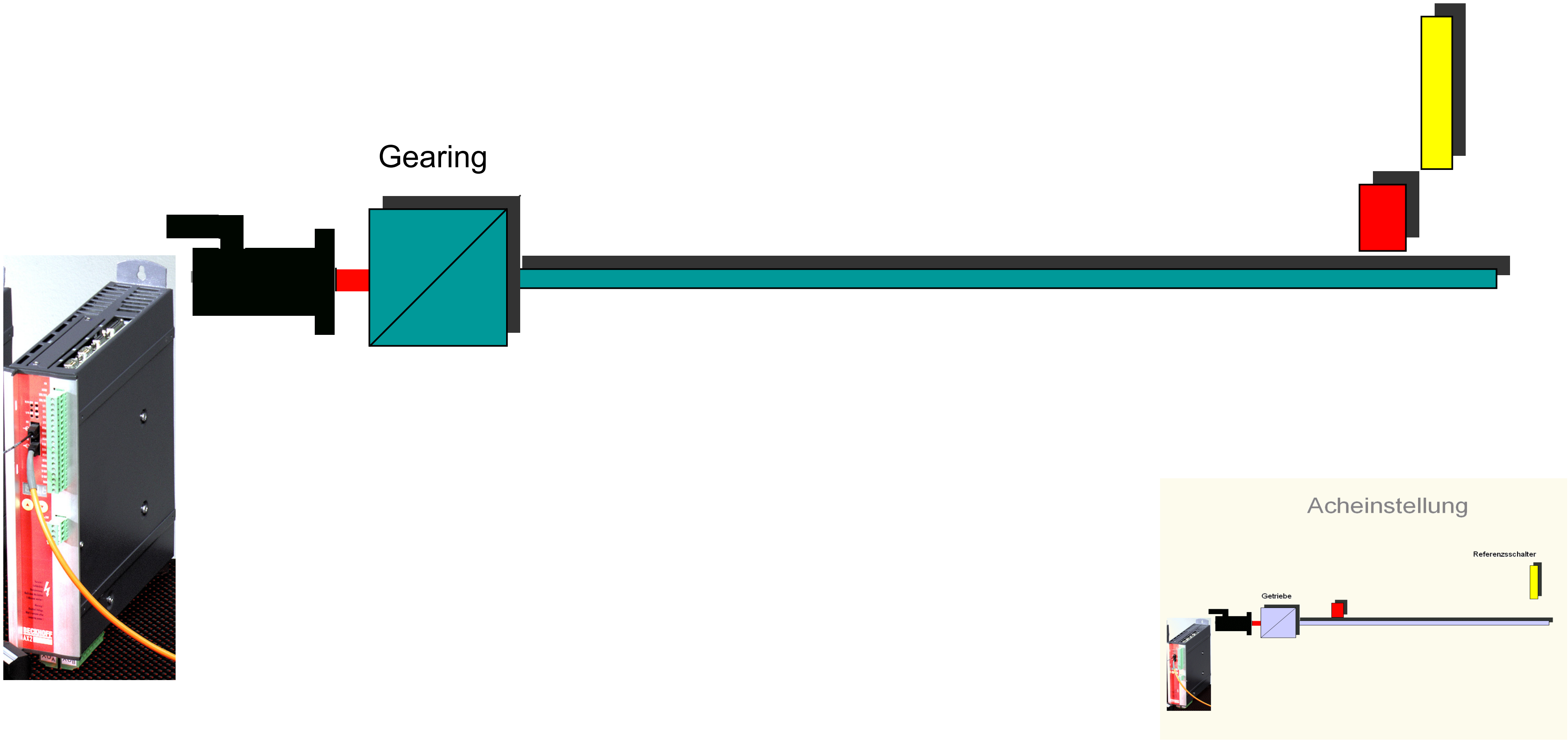
Furthermore the Safety instructions are to be considered absolutely.

TwinCAT Information System NC -> Safety functionalities.

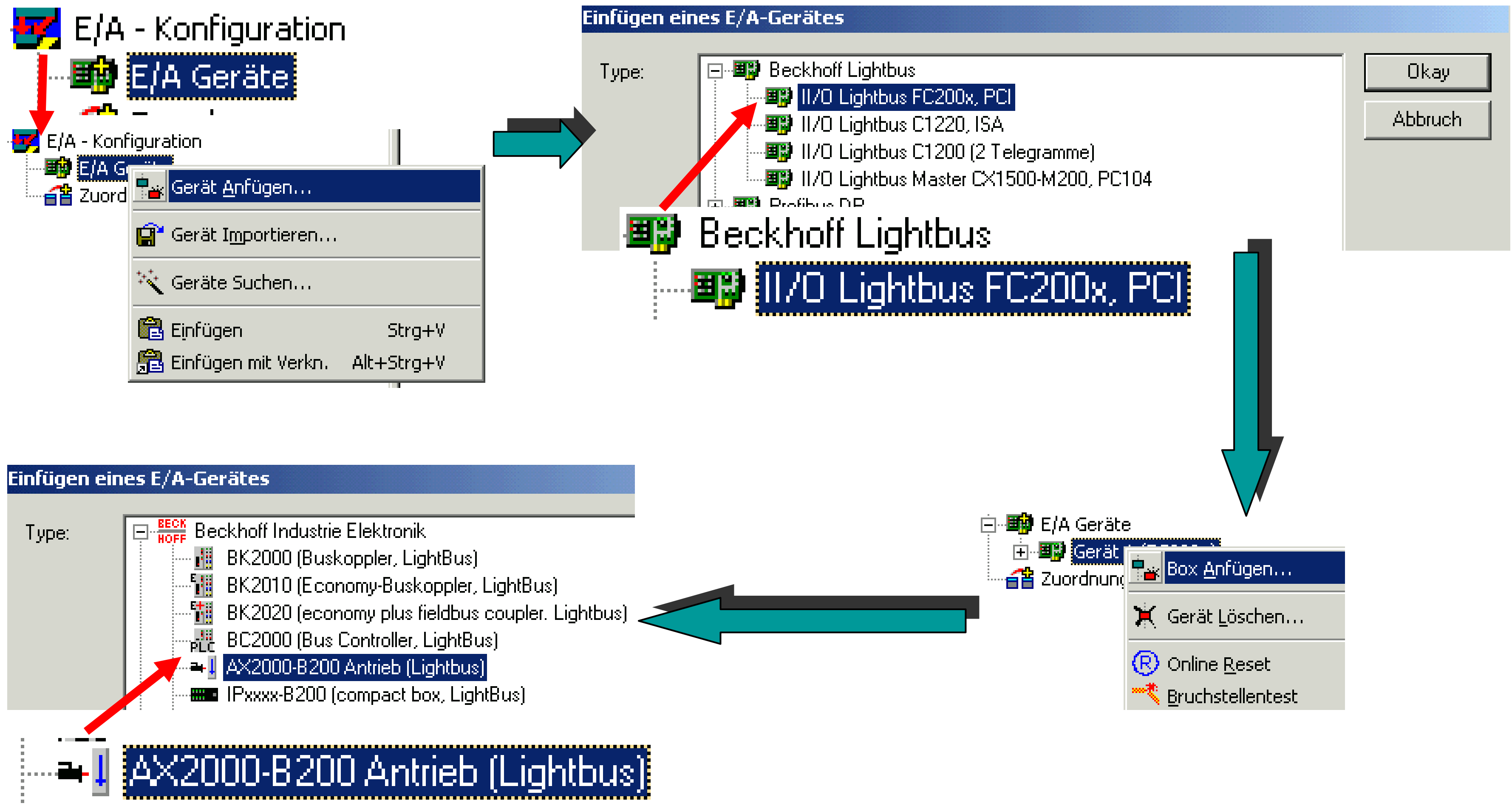
Axes Settings

Example configuration	
Data AX2000: Settings at the training devices	
Max. r.p.m.	3000
Increments per Motor revolution (Encoderemulation of AX2000)	65535
Adopted mechanical ration	1 motor revolution is equivalent to 1mm mechanical way

Axes Settings (adopted Application modell)

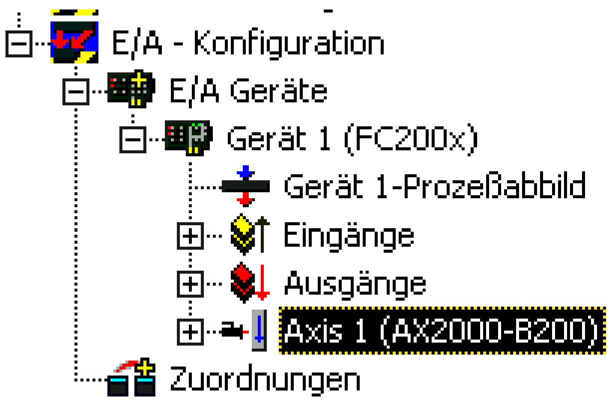


Enter Hardware

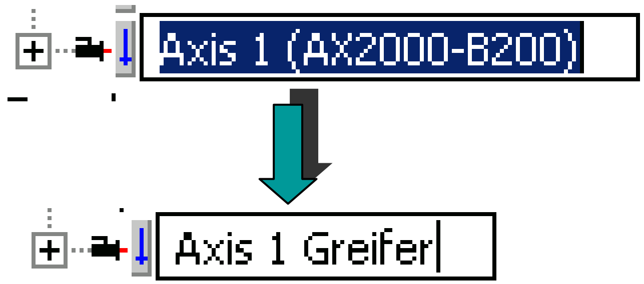


Enter Hardware

Result

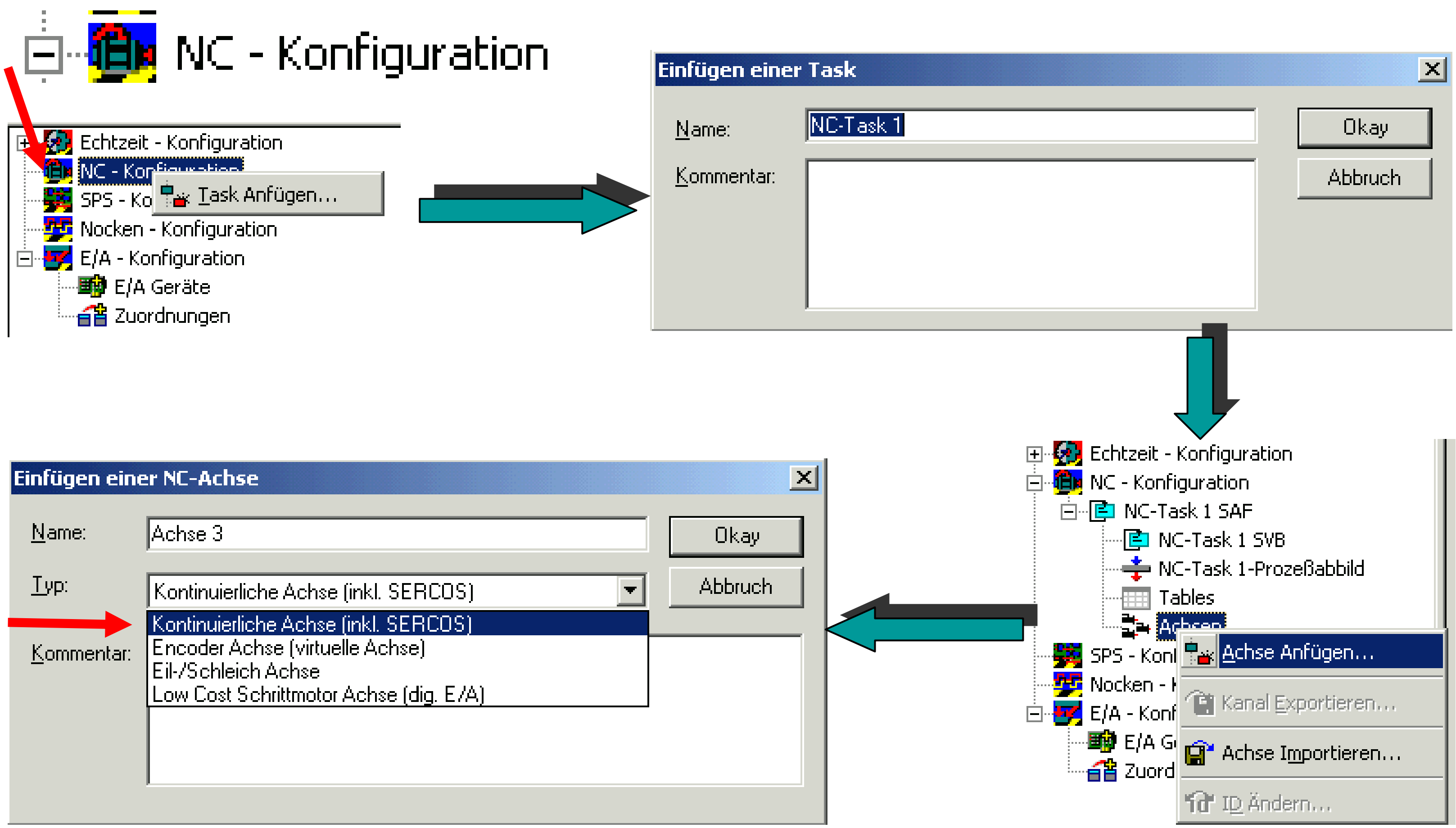


Rename?

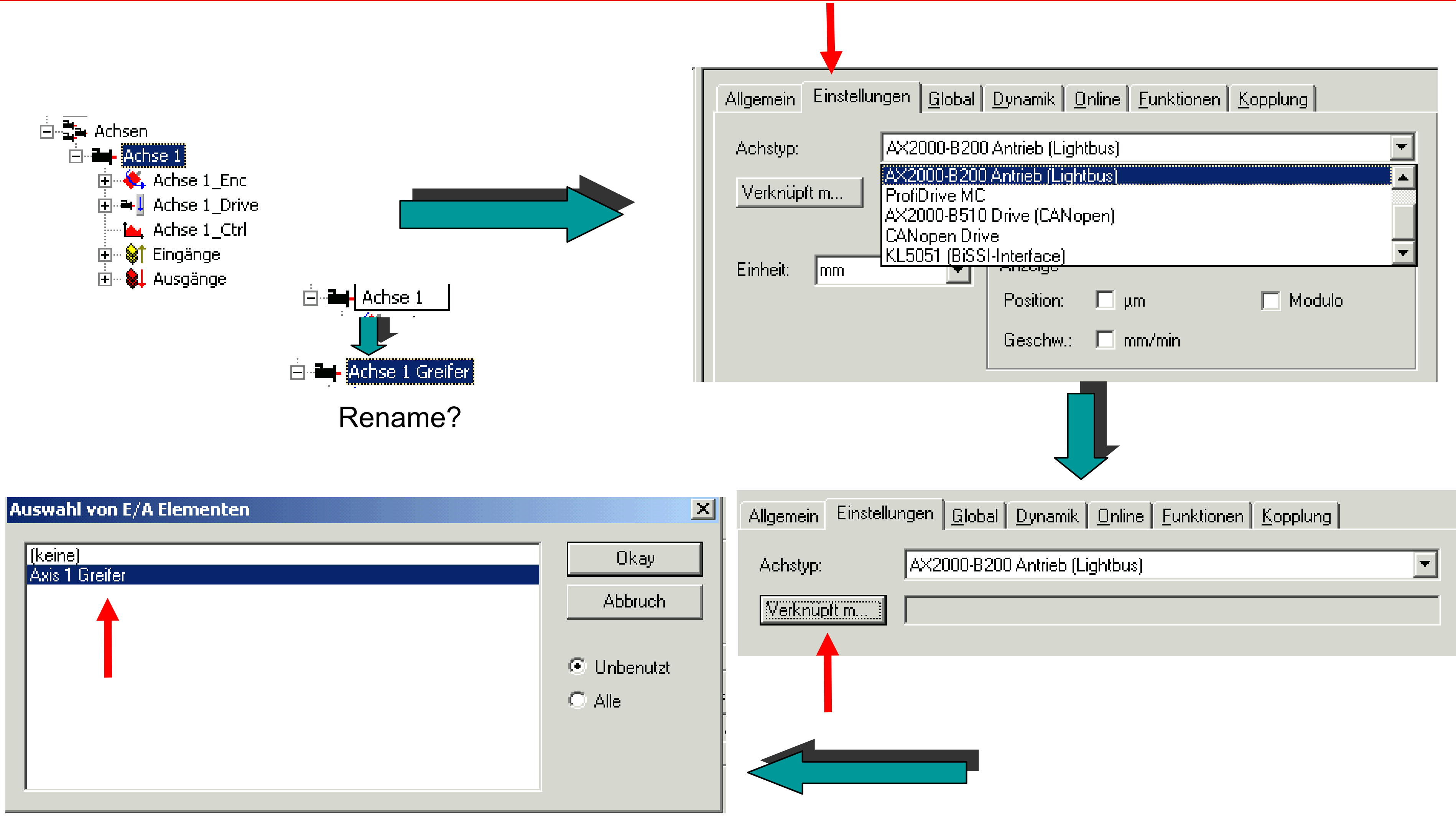


Repeat steps for all further drives

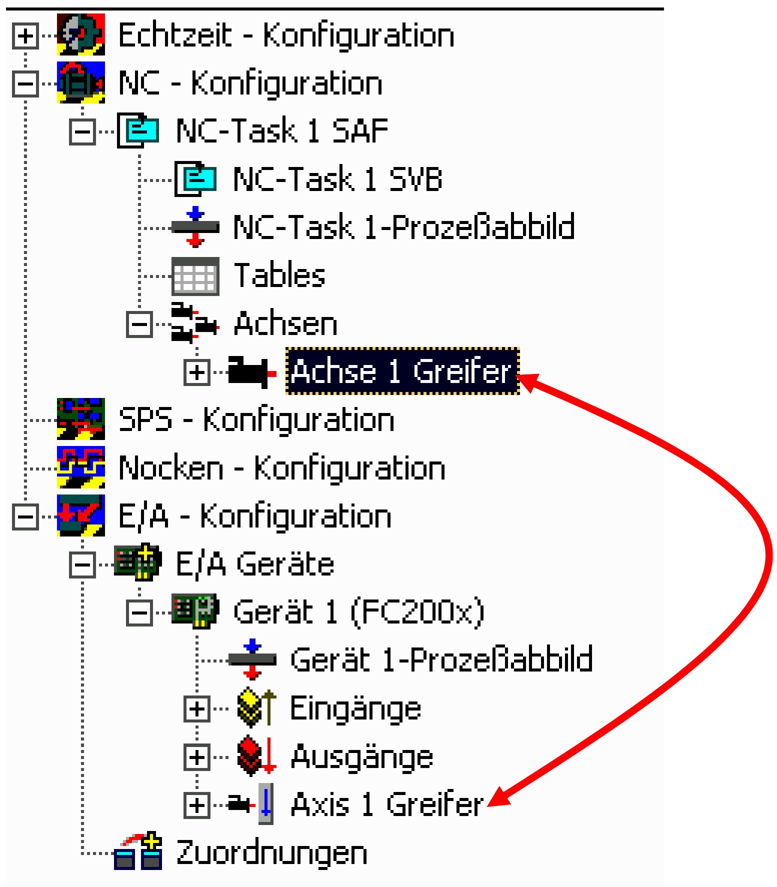
Setting up NC Axes in the System Manager



Select and link drive type

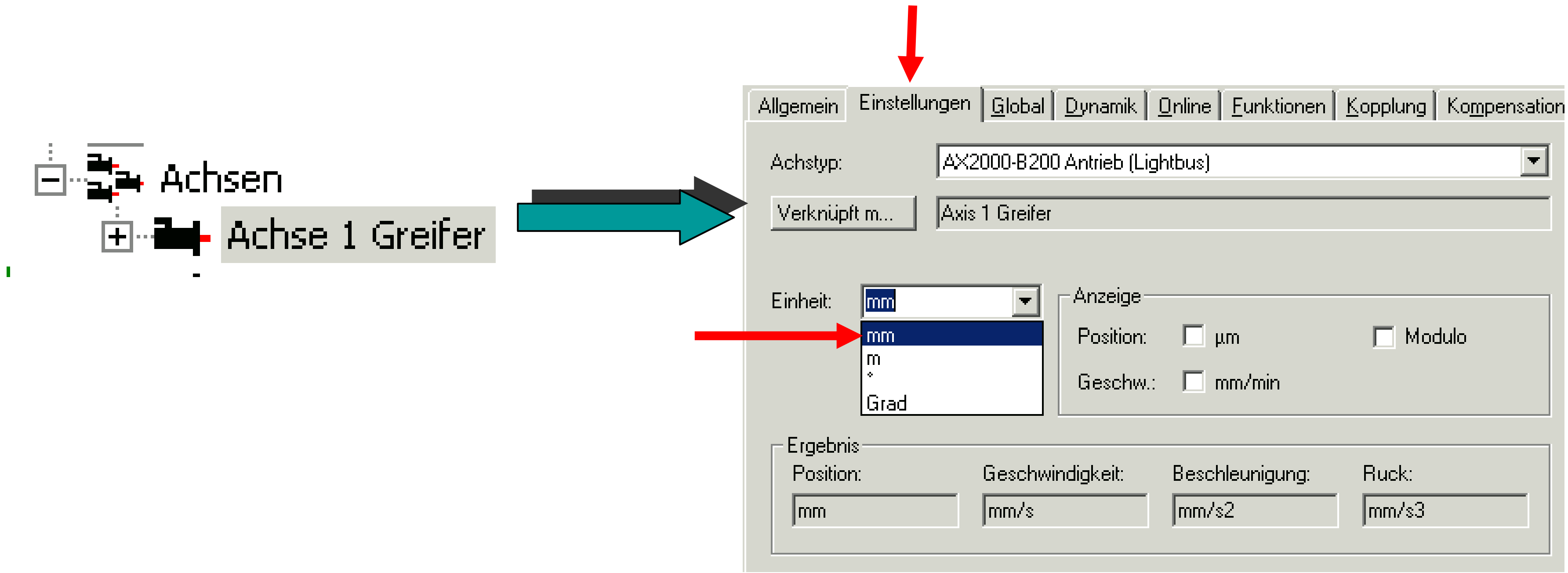


Select and link drive type



Inputs and outputs between TwinCAT NC Controller and AX 2000 are linked

Define Axis parameter units



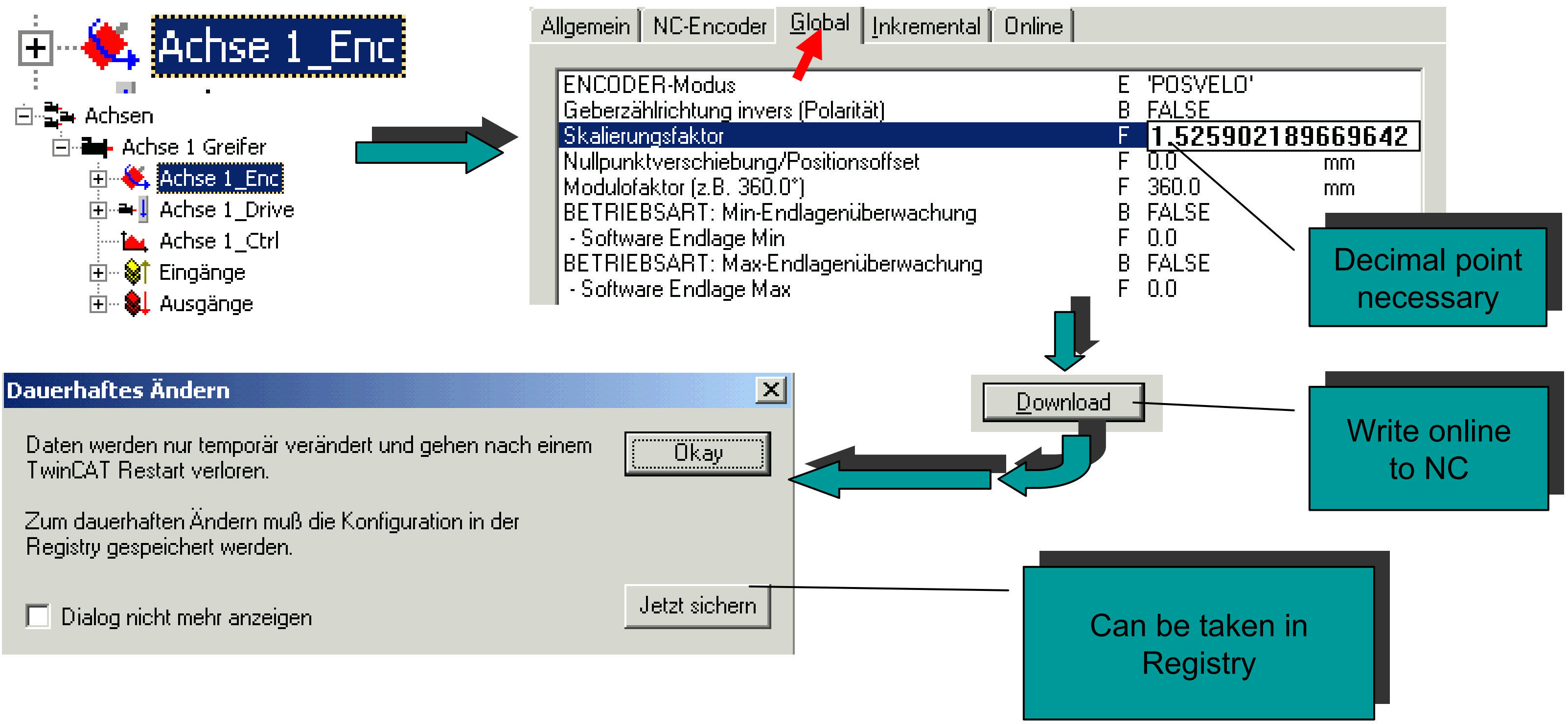
Encoder parameter Scaling factor

Translation of the collected actual position value in the way unit

$$ScalingFactor = \frac{milageWay}{NumberIncrements}$$

Example :

$$ScalingFactor = \frac{1mm}{65535inc} = 1{,}52590\text{e} - 5\,\frac{\text{mm}}{\text{inc}}$$



Further Encoder parameter (Notice)

Allgemein | NC-Encoder | Global | Inkremental | Online

ENCODER-Modus	E	'POSVELO'	
Geberzählrichtung invers (Polarität)	B	FALSE	
Skalierungsfaktor	F	0.01	mm/INC
Nullpunktverschiebung/Positionsoffset	F	0.0	mm
Modulofaktor (z.B. 360.0°)	F	360.0	mm
BETRIEBSART: Min-Endlagenüberwachung	B	FALSE	
- Software Endlage Min	F	0.0	mm
BETRIEBSART: Max-Endlagenüberwachung	B	FALSE	
- Software Endlage Max	F	0.0	mm

Drive parameter reference velocity

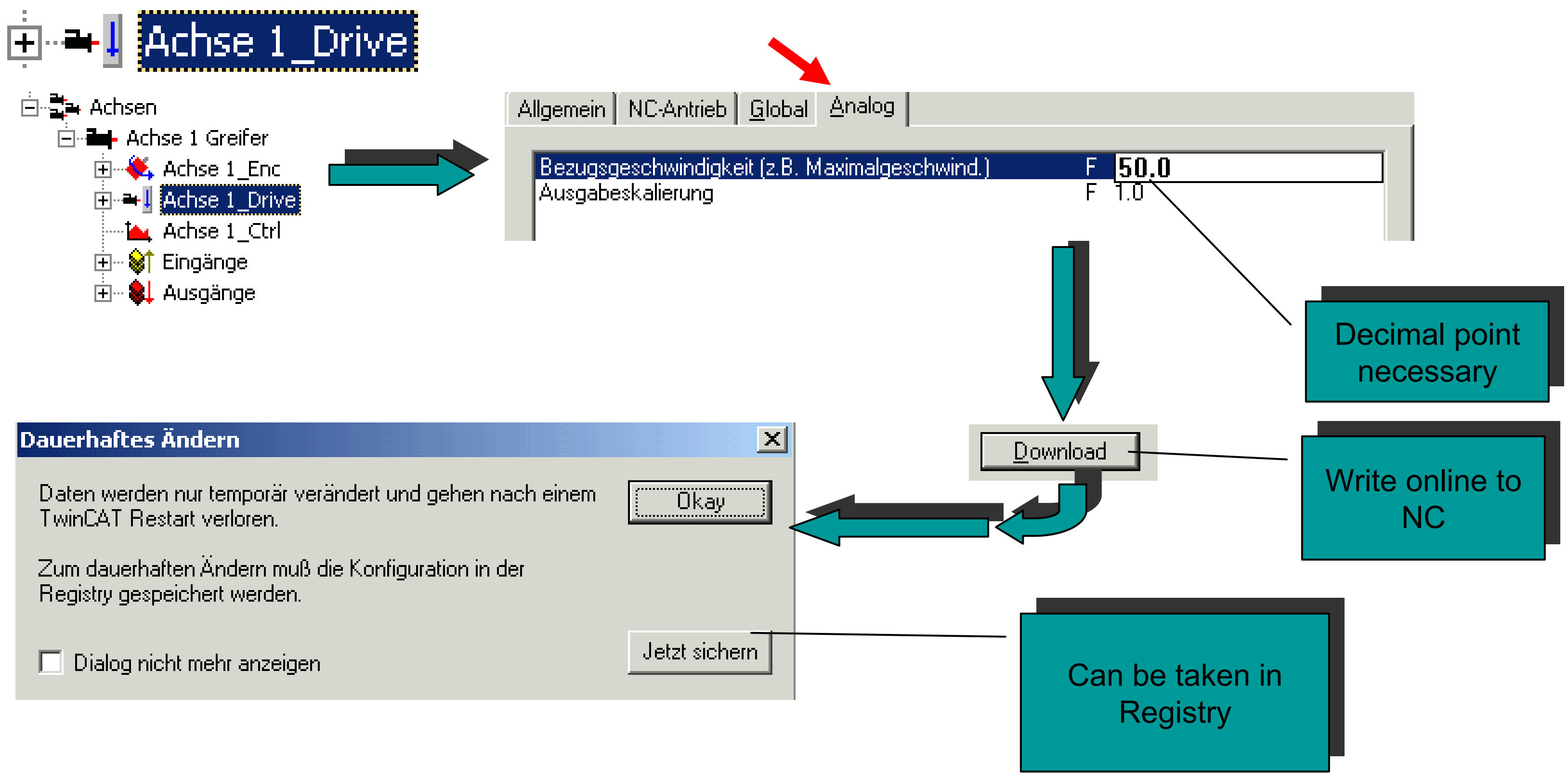
Translation of the max. rotation speed of the drive to a velocity

RPM rotation
per minute

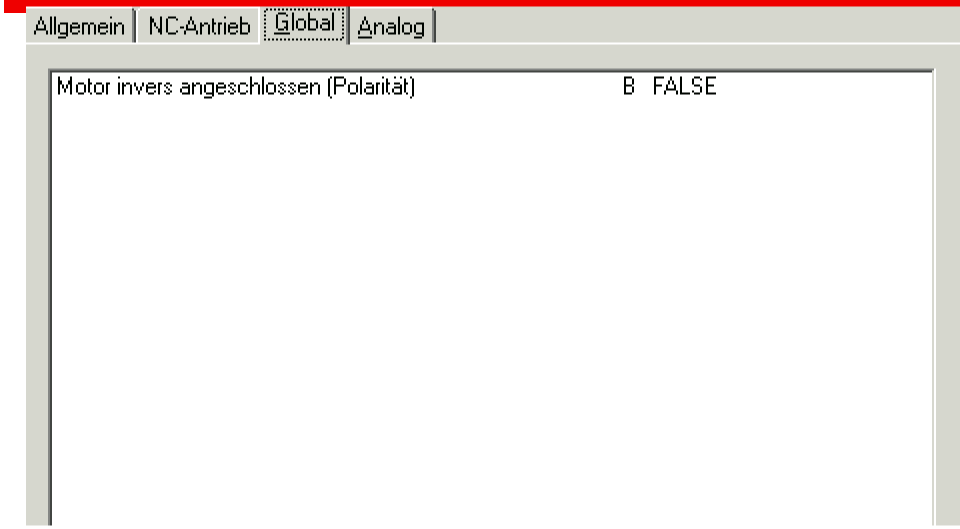
$$\mathrm{Re}\ frencevelocity = \max Rotationspeed * \frac{Way}{Rotation}$$

Example:

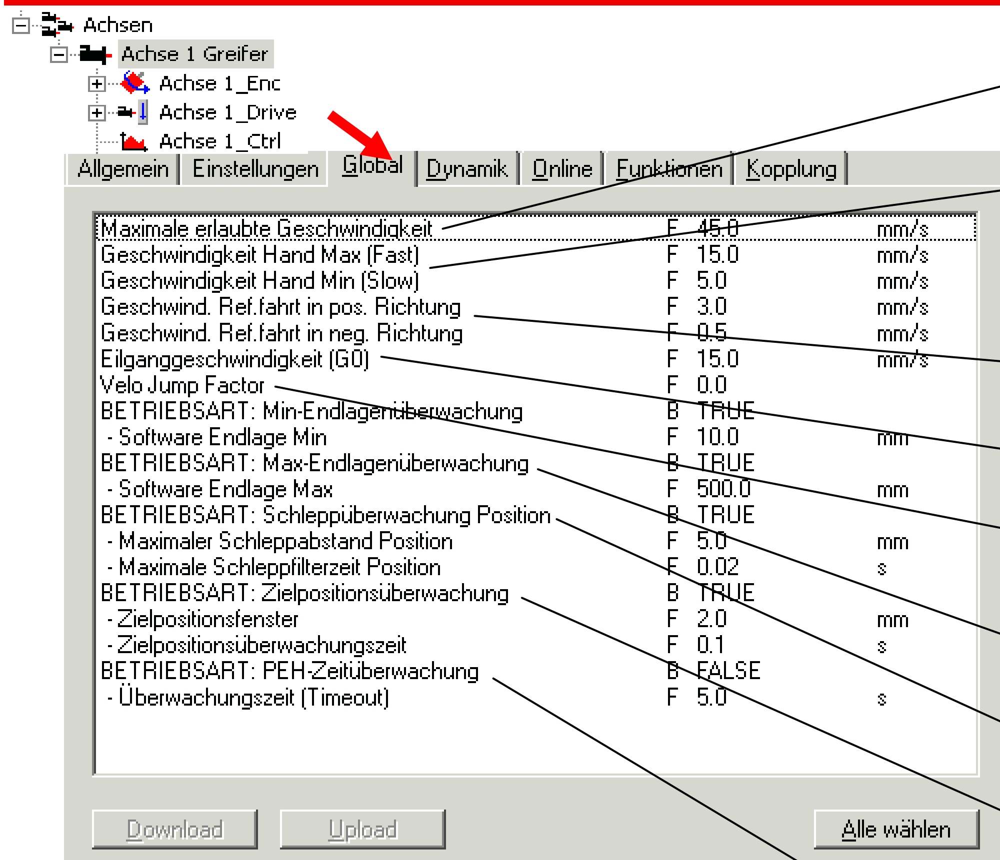
$$\mathrm{Re}\ ferenceVelocity = \frac{3000U}{60s} * \frac{1mm}{U} = 50\frac{mm}{s}$$



Further Drive parameter (Notice)

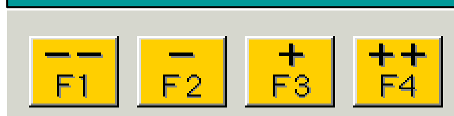


Global Axis parameter



Limitation of the max. allowed velocity

Hand velocities in the Online Menue



Referencing velocities

Infeed velocity (NCI only)

Velocity reduction at segment transitions (NCI only)

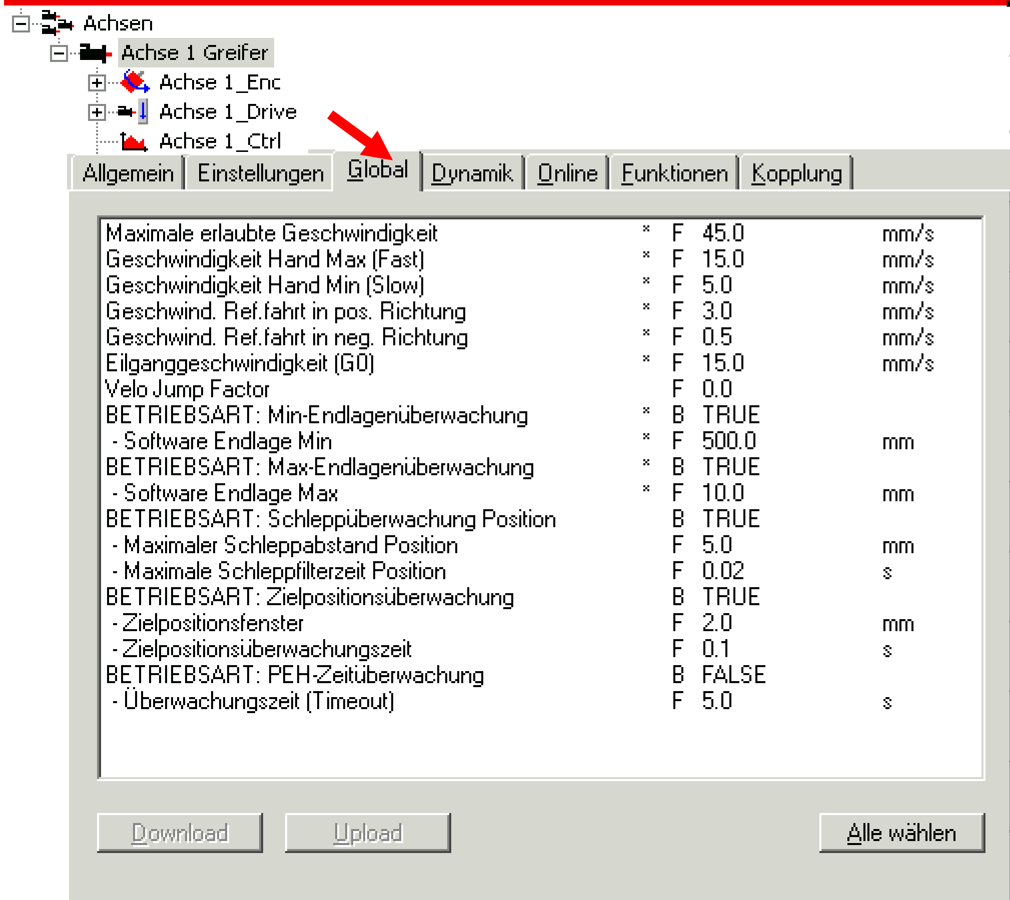
Software limit switch

Following Error monitoring

Target position monitoring
Precision and filter for message to PLC „Axis in target position“

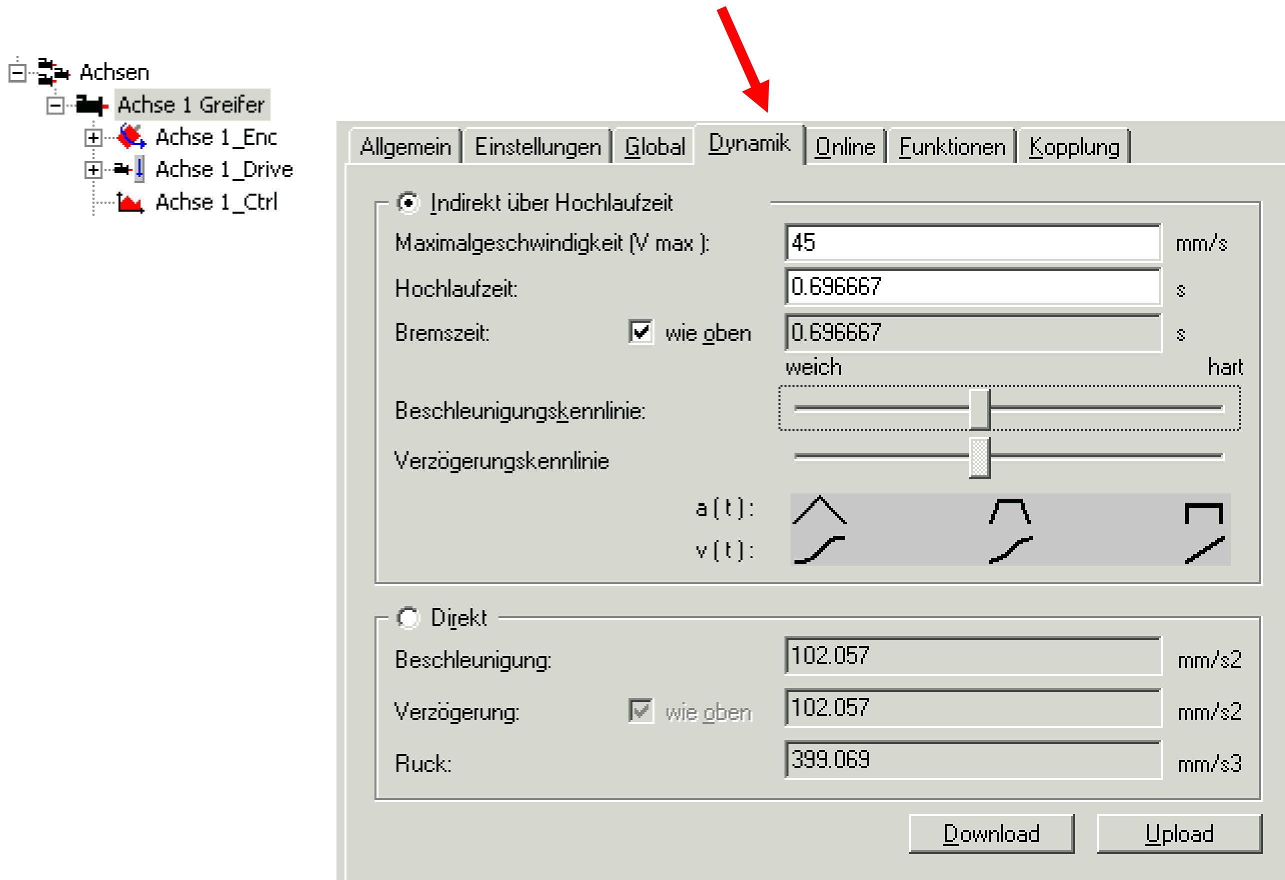
Timeout monitoring if axis is in target position

Global Axis parameter (Notice)



Maximale erlaubte Geschwindigkeit	*	F	45.0	mm/s
Geschwindigkeit Hand Max (Fast)	*	F	15.0	mm/s
Geschwindigkeit Hand Min (Slow)	*	F	5.0	mm/s
Geschwind. Ref.fahrt in pos. Richtung	*	F	3.0	mm/s
Geschwind. Ref.fahrt in neg. Richtung	*	F	0.5	mm/s
Eilganggeschwindigkeit (G0)	*	F	15.0	mm/s
Velo Jump Factor		F	0.0	
BETRIEBSART: Min-Endlagenüberwachung	*	B	TRUE	
- Software Endlage Min	*	F	500.0	mm
BETRIEBSART: Max-Endlagenüberwachung	*	B	TRUE	
- Software Endlage Max	*	F	10.0	mm
BETRIEBSART: Schleppüberwachung Position		B	TRUE	
- Maximaler Schleppabstand Position		F	5.0	mm
- Maximale Schleppfilterzeit Position		F	0.02	s
BETRIEBSART: Zielpositionsüberwachung		B	TRUE	
- Zielpositionsfenster		F	2.0	mm
- Zielpositionsüberwachungszeit		F	0.1	s
BETRIEBSART: PEH-Zeitüberwachung		B	FALSE	
- Überwachungszeit (Timeout)		F	5.0	s

Dynamic



Reference switch

Effects see Set value profiles

Starting NC Axes from the PLC

Example: A small project for starting an single axis should be created with the help of the MC Library

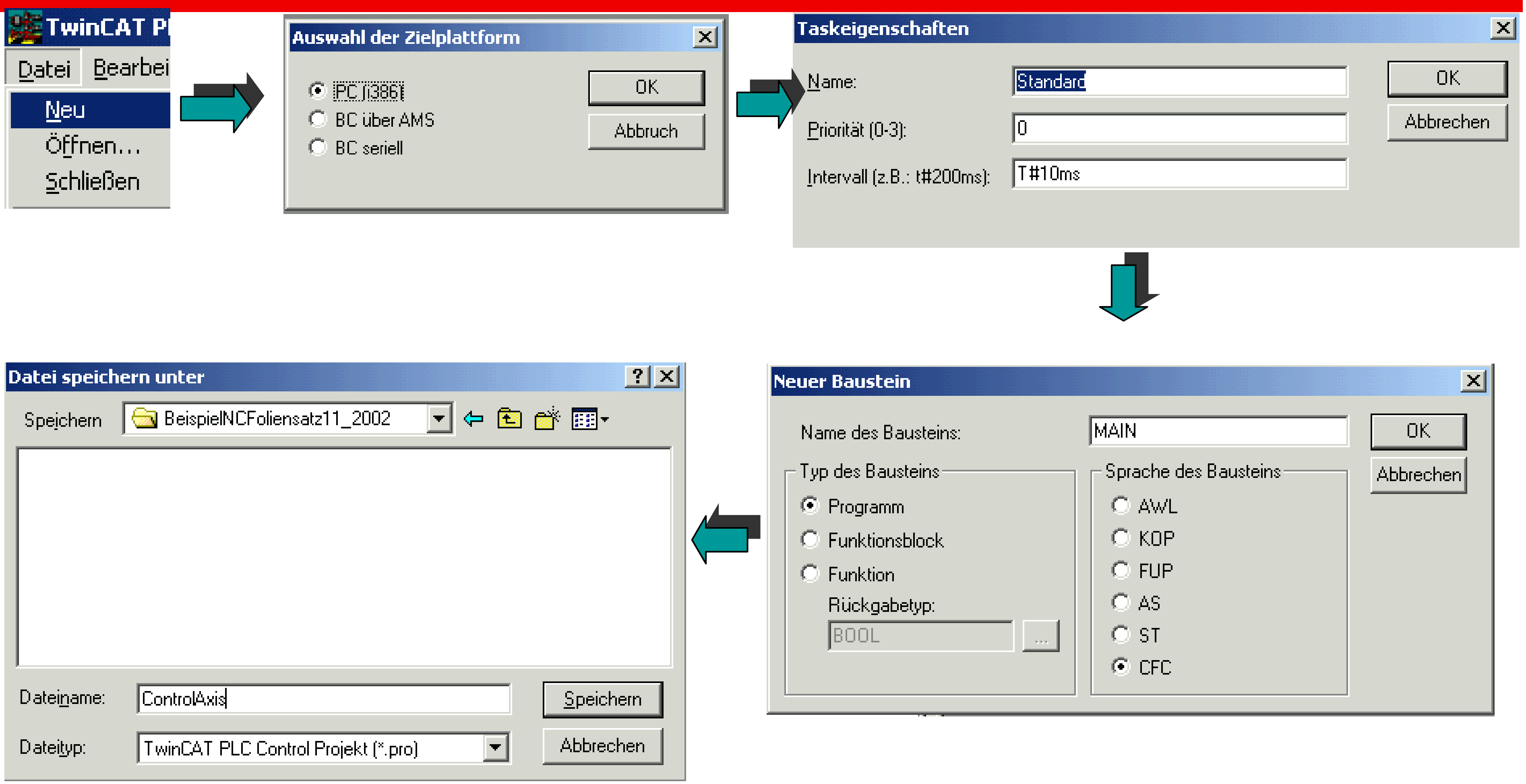
Notes:

- The control of the enable signals (hardware) is not treated in this example.

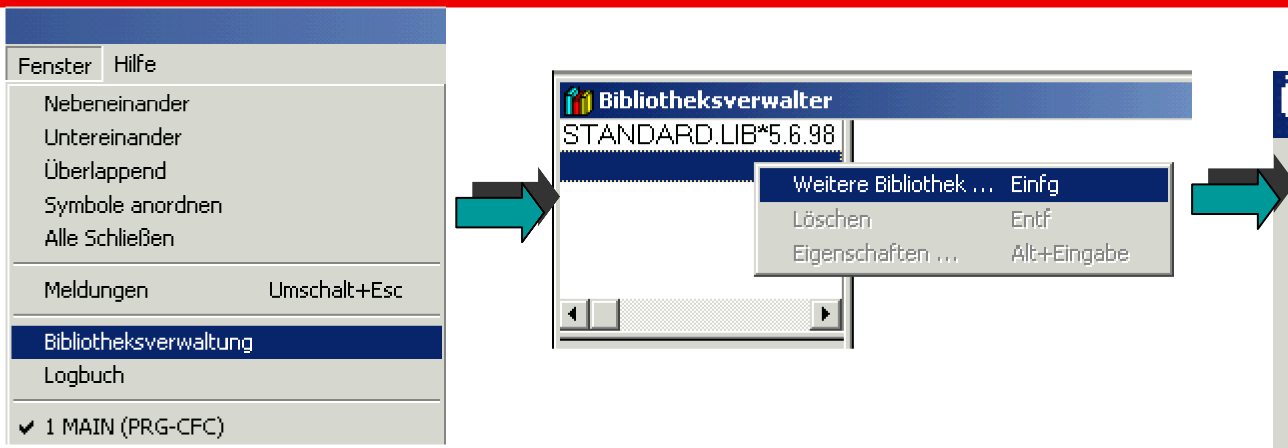
- The programming mode corresponds in this example to the classical PLC programming, that means global variables for the inputs and outputs and referencing in the POUs.

- An alternative is the creation of FBs, which work internally with not total located variables for the axis interface.(VAR_CONFIG)

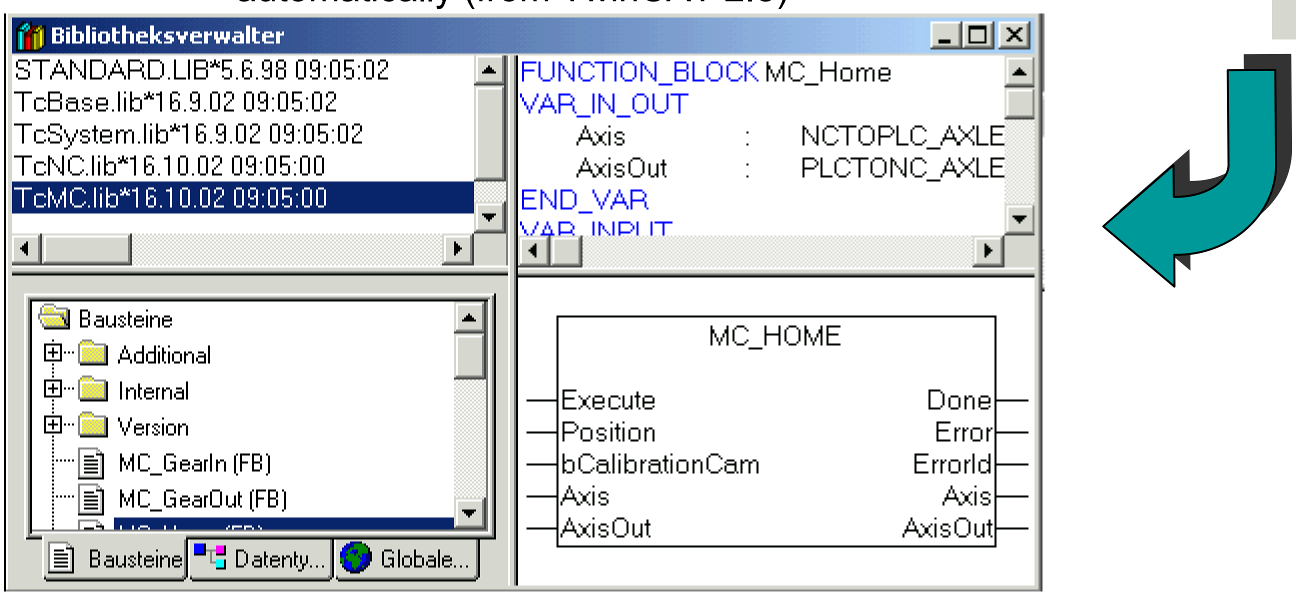
Creating a PLC Project



Inserting TcMC Library



Further required libraries will be inserted automatically (from TwinCAT 2.8)



Creating Input / Output variable between NC and PLC

Globale_Variablen

```
VAR_GLOBAL
    (*Axisinterface*)
    Axis1GreiferPlcToNc    AT%QB1000 :      PLCTONC_AXLESTRUCT;
    Axis1GreiferNcToPlc    AT%IB1000 :      NCTOPLC_AXLESTRUCT;
```

To consider:

1 Variable occupies 128 Byte. Thus the next free address to start is IB/QB 1128.

In addition the possibility of auto addressing can be used.

Axis1GreiferPlcToNc AT%QB* : PLCTONC_AXLESTRUCT;

Globale_Variablen

```
VAR_GLOBAL
    (*Axisinterface*)
    Axis1GreiferPlcToNc    AT%QB1000 :      PLCTONC
    Axis1GreiferNcToPlc    AT%IB1000 :      NCTOPLC


    (*I/O for control *)
    genRelease             AT%IX0.0:        BOOL;
    RequestHoming          AT%IX0.1:        BOOL;
    RequestSequence        AT%IX0.2:        BOOL;
    SwitchReferenceCamAxis1 AT%IX0.3:       BOOL;
```

Control inputs.

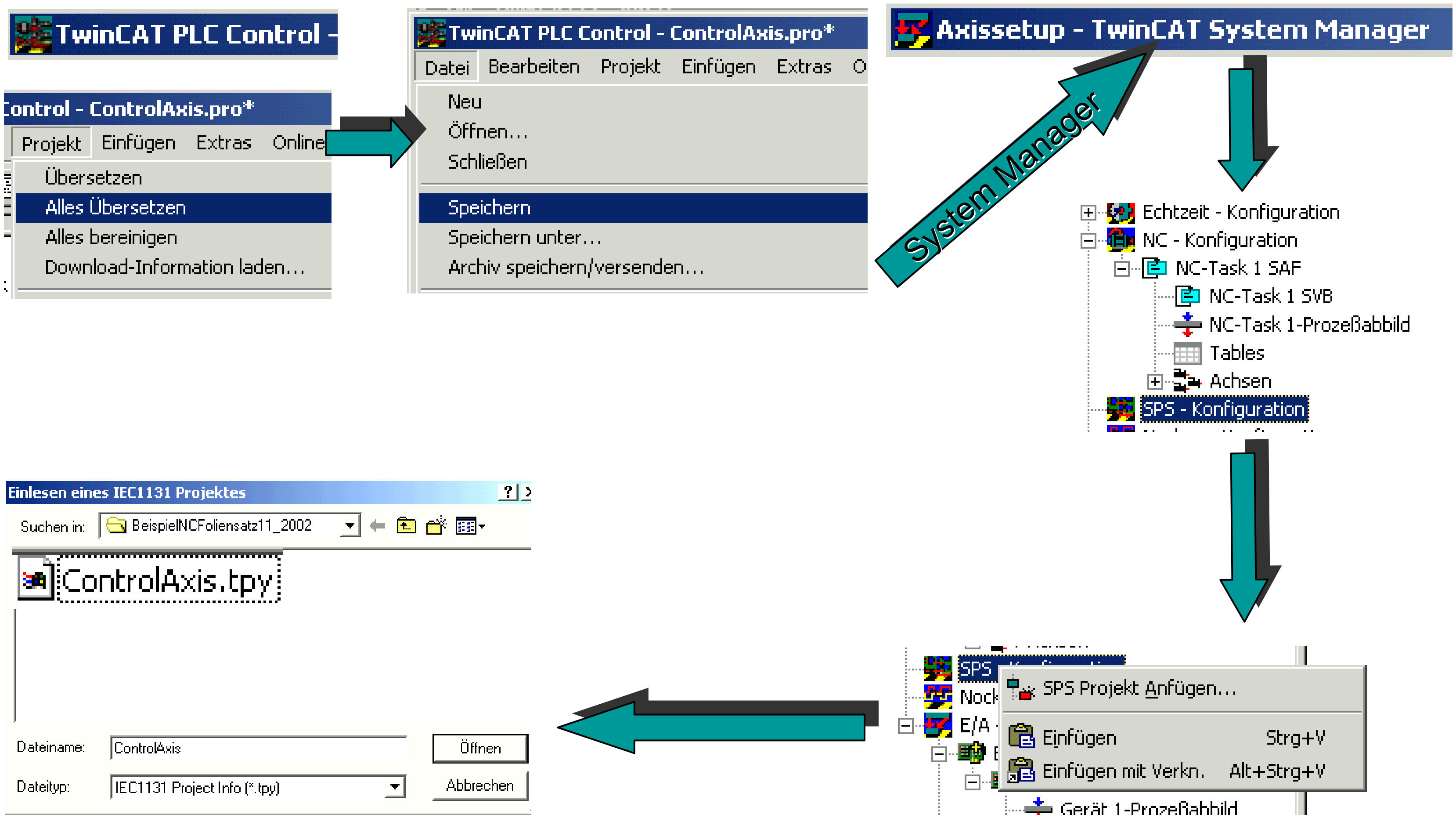
Linking with hardware,

„ Writing values“ in PLC Control, or

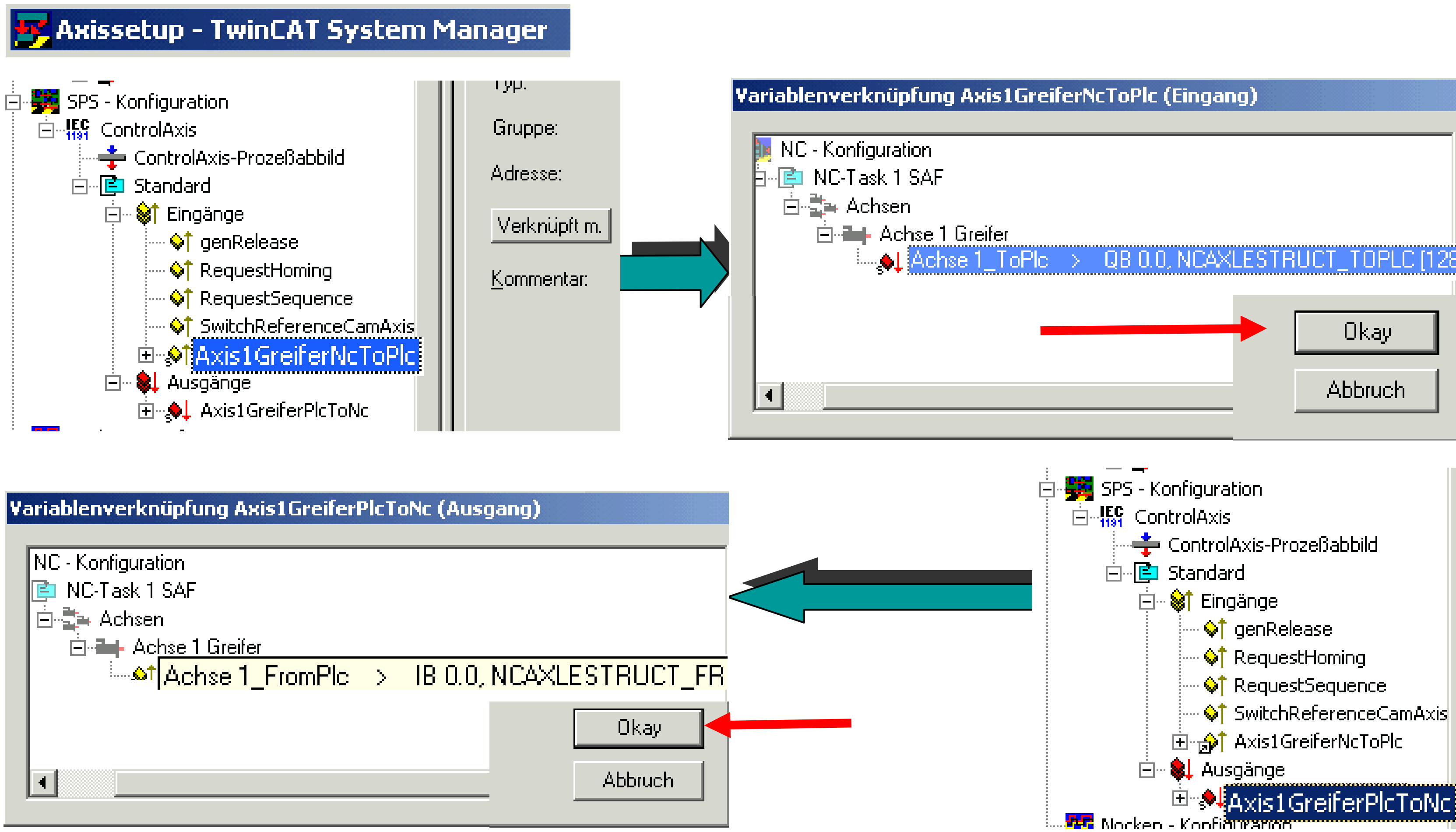
Control with a small VB / VC++

It´s understood, that at direct commissioning at a movement, the safety precautions are to be considered

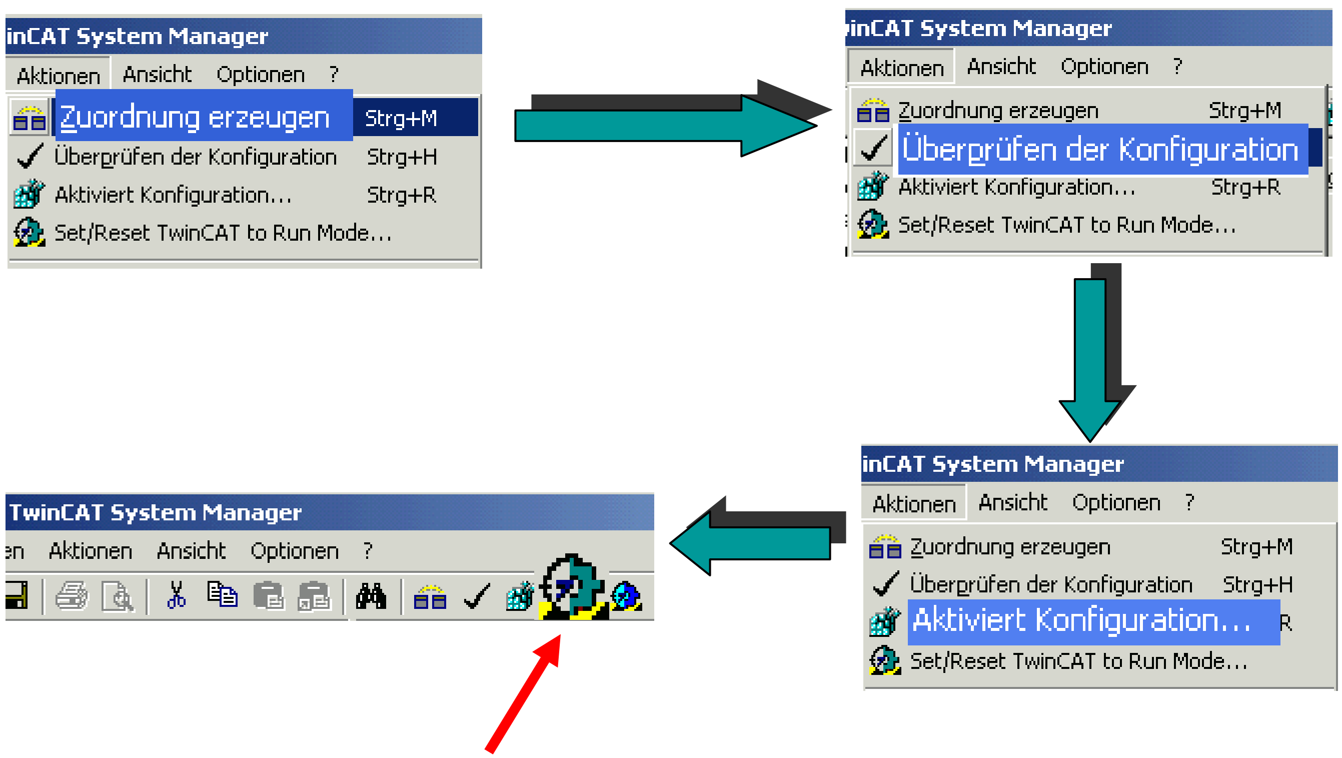
Linking Input / Output variable between NC and PLC



Linking Input / Output variable between NC and PLC

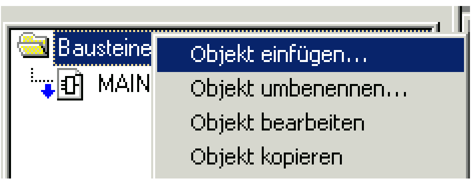


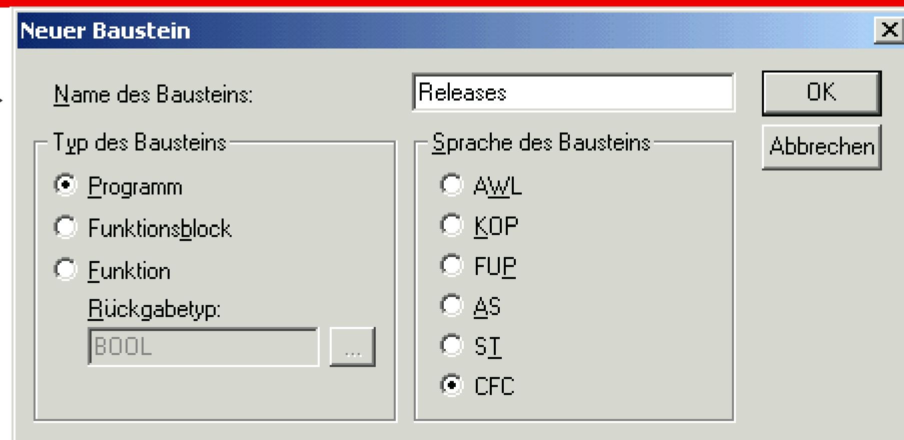
Linking further control inputs and writing configuration in registry

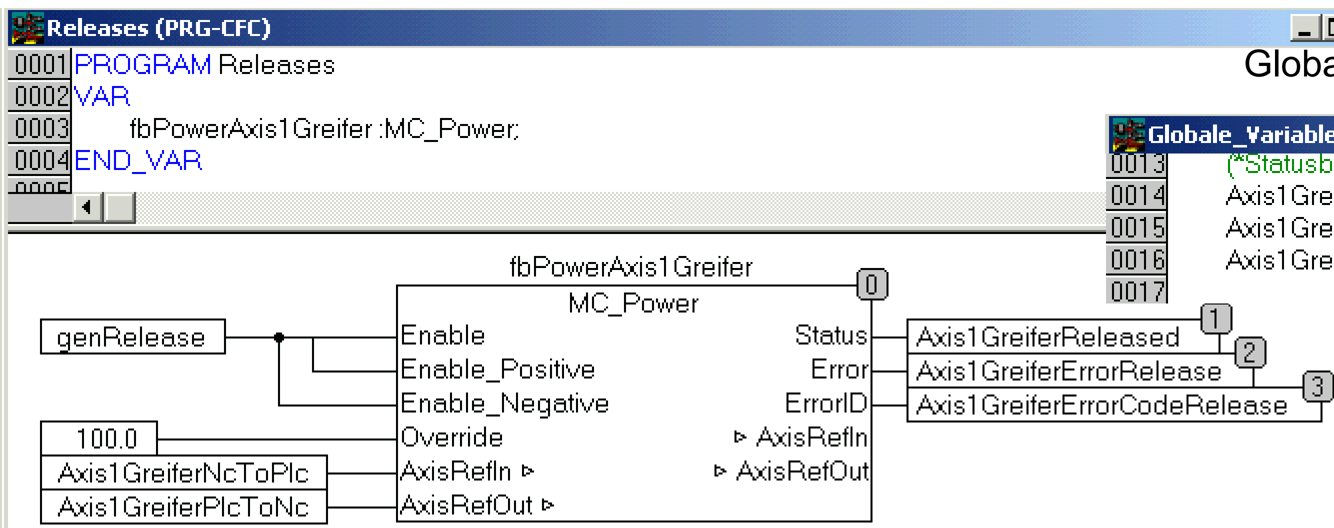


System can be started here.

Programming axis enables MC_POWER



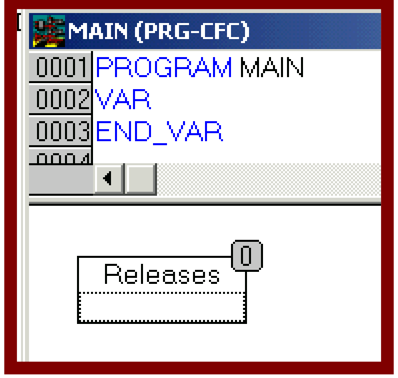




Global Status Variable for MC_Power

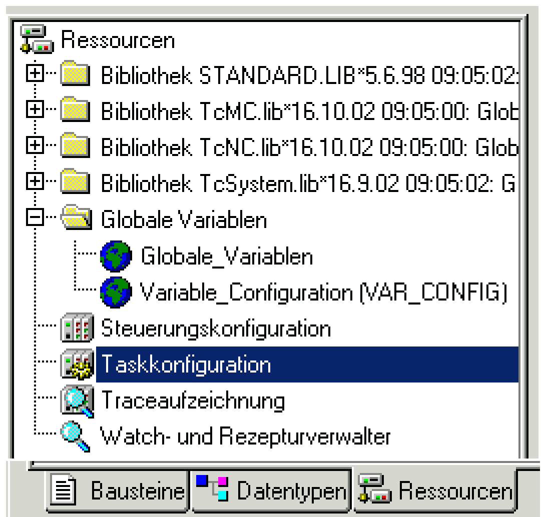
Calling axis enables

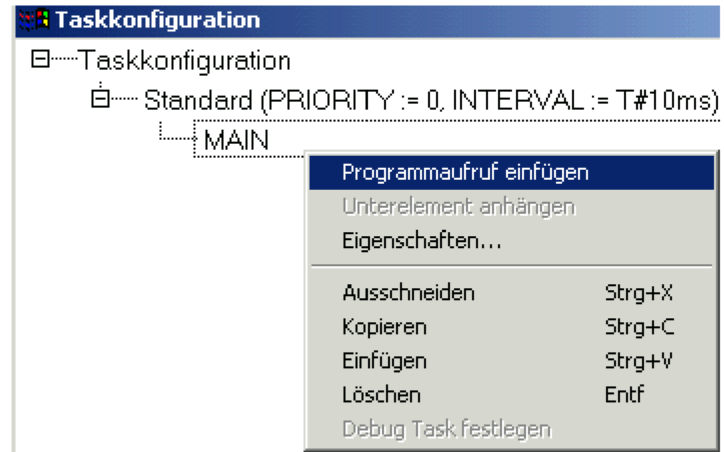
Direct in Main

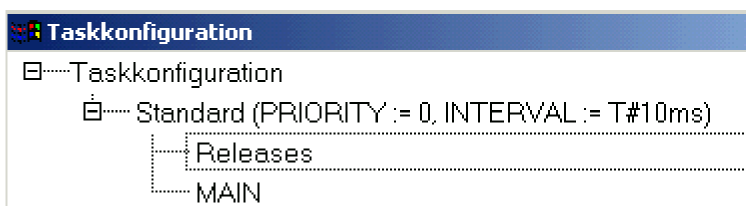


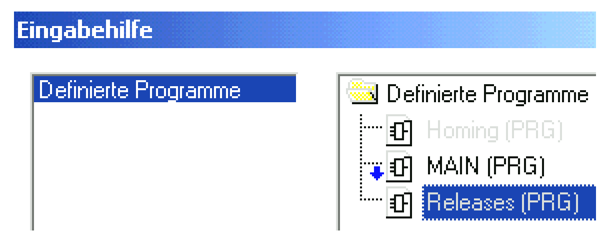
USE ONLY ONE POSSIBILITY!

Or via Taskconfiguration

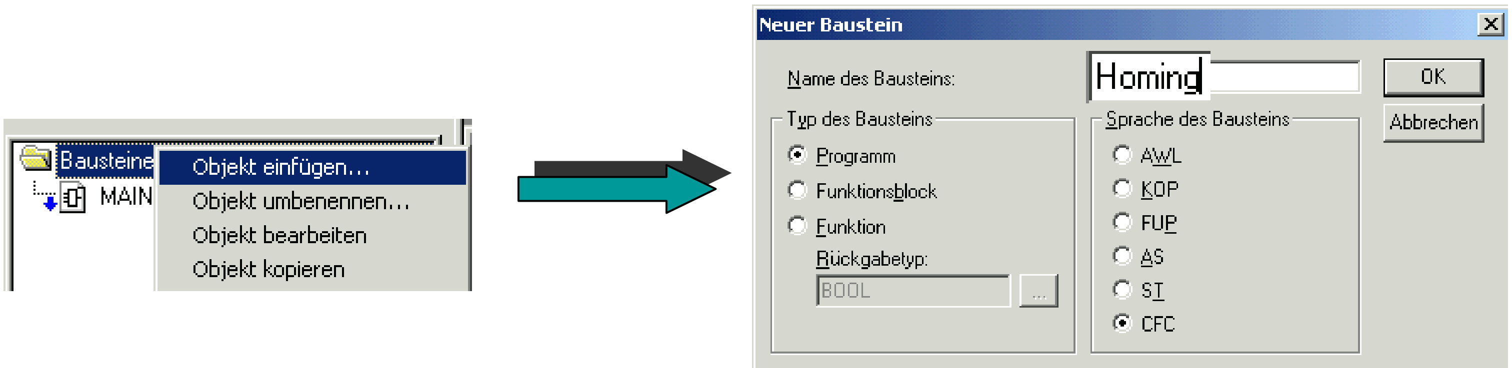




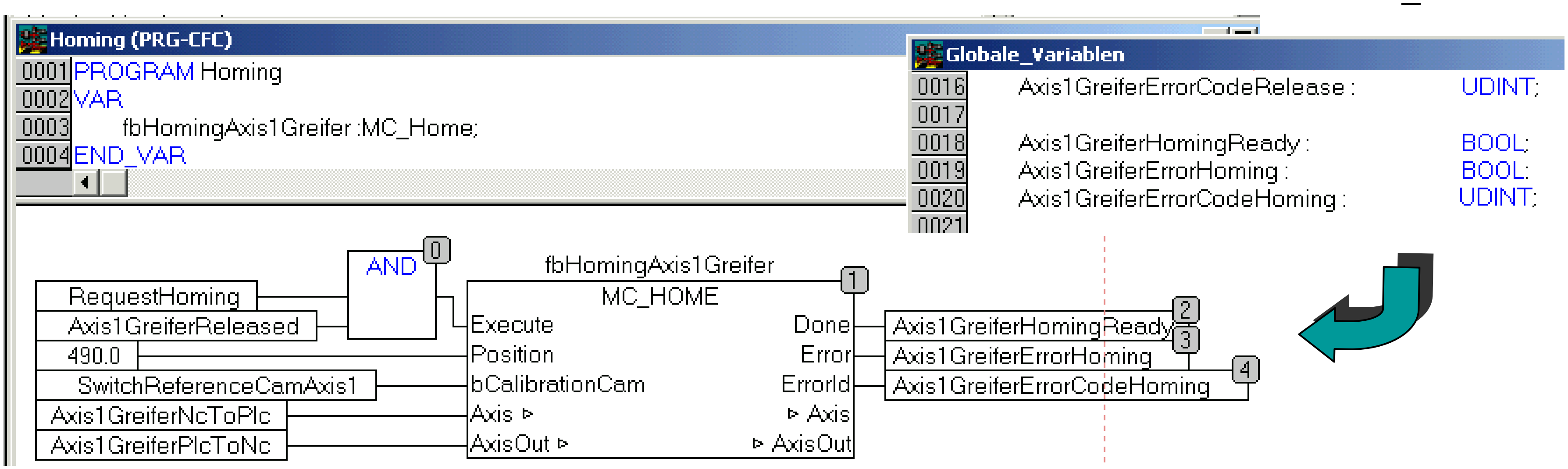




Instantiate and call MC Home block

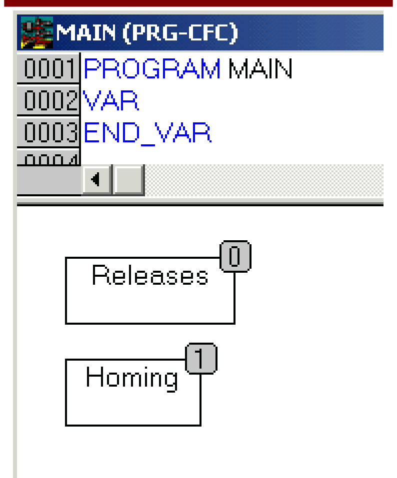


Global Status Variable for MC_Home



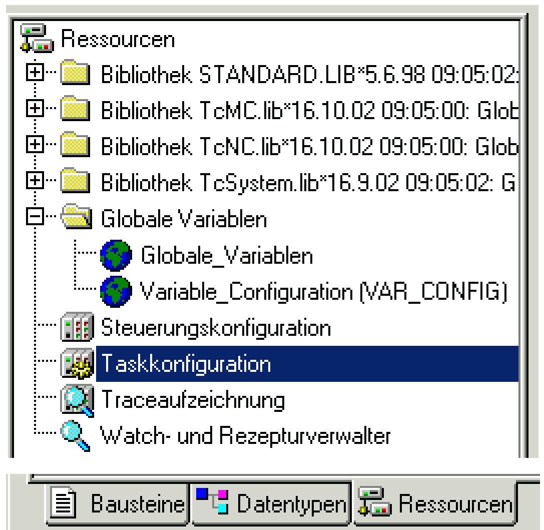
Calling Homing

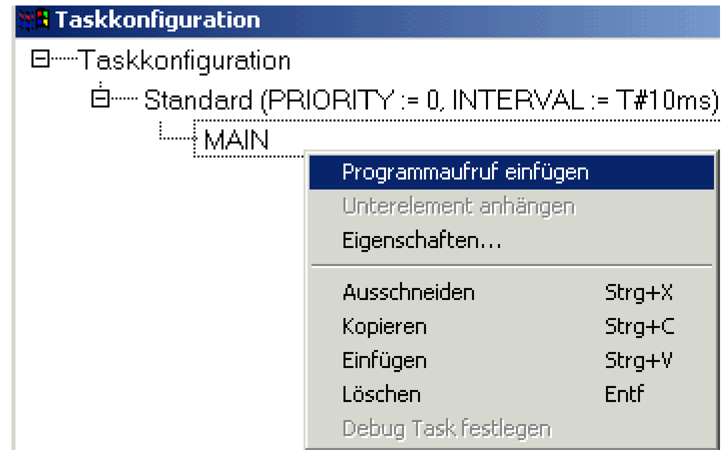
Direct in Main

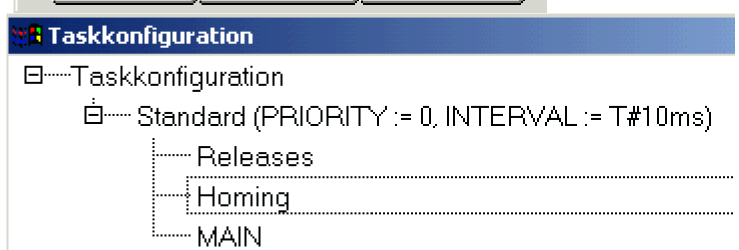


USE ONLY ONE POSSIBILITY!

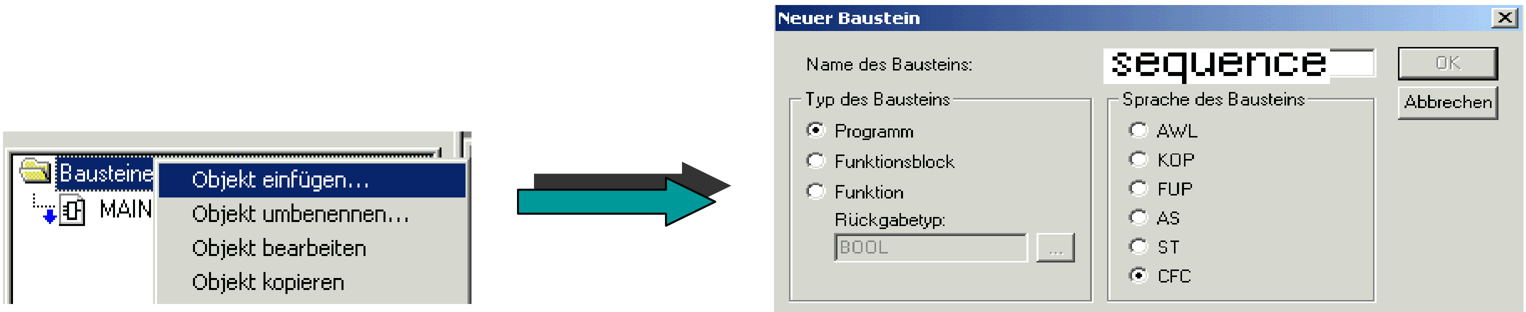
Or via Taskconfiguration



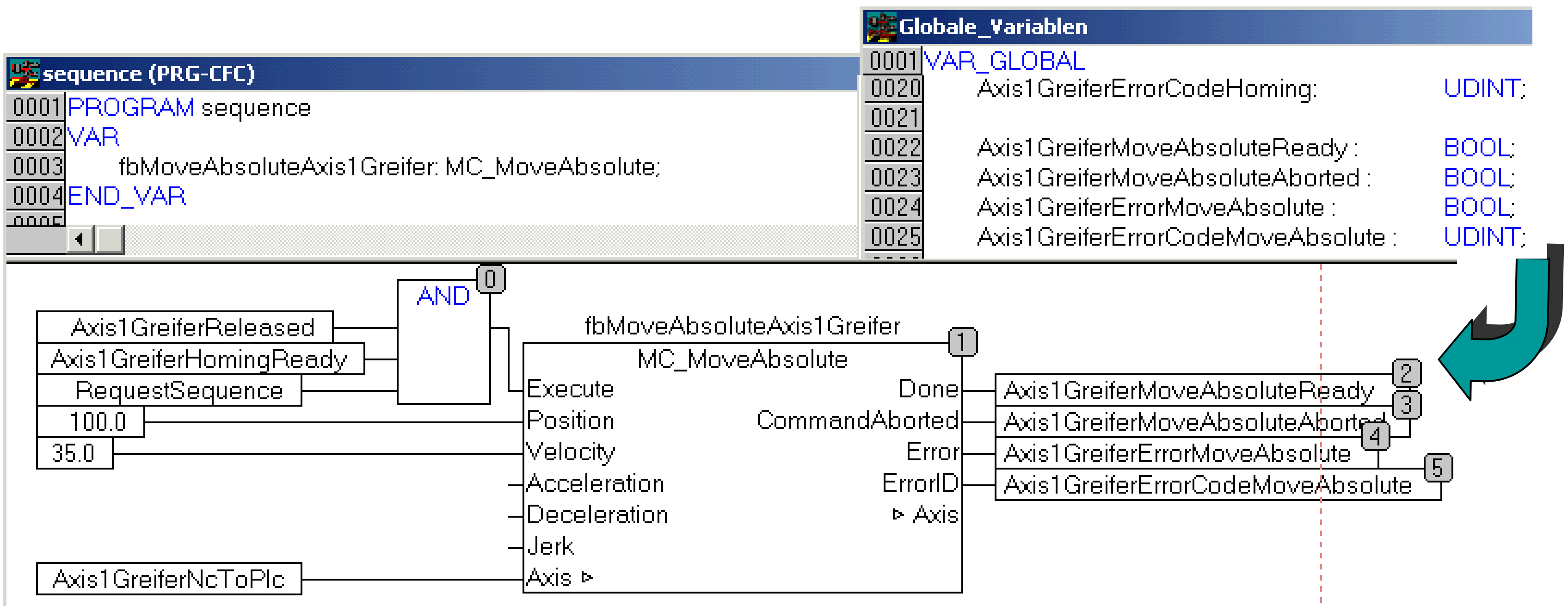




Instantiate and call MC MoveAbsolute block

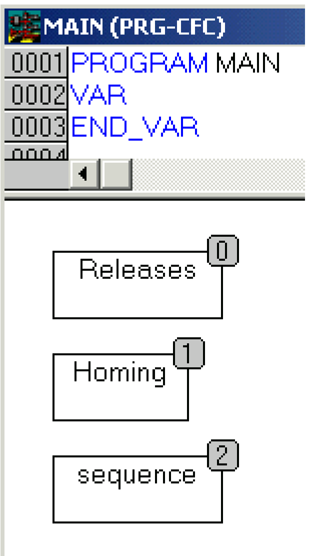


Global Status Variable for MC_MoveAbsolute



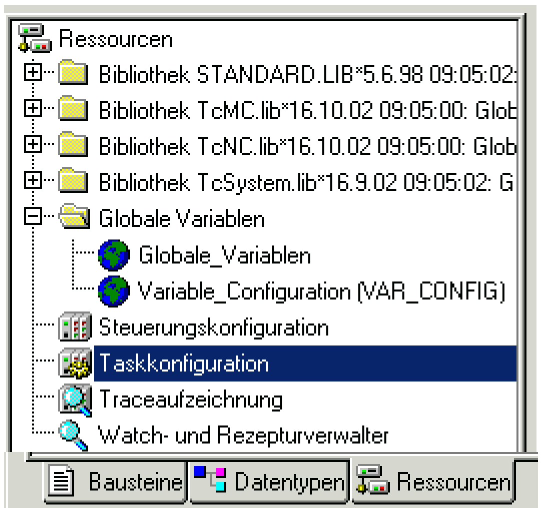
Calling Sequence

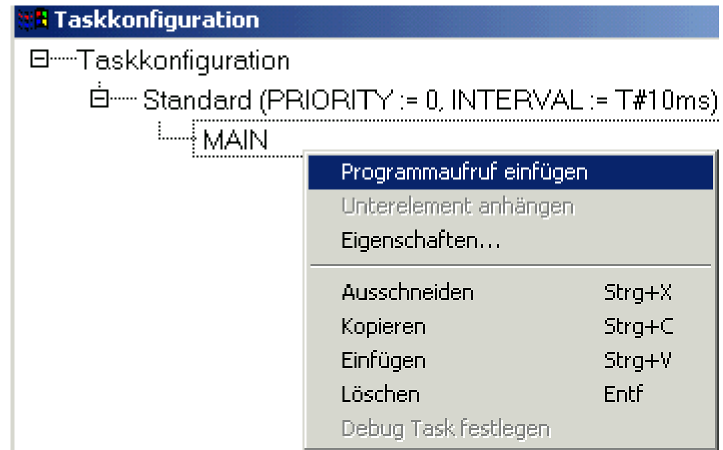
Direct in Main

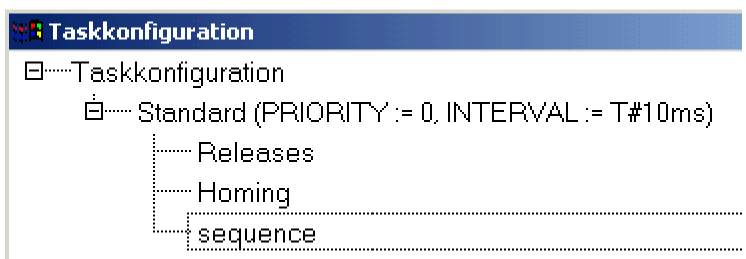


USE ONLY ONE POSSIBILITY!

Or via Taskconfiguration





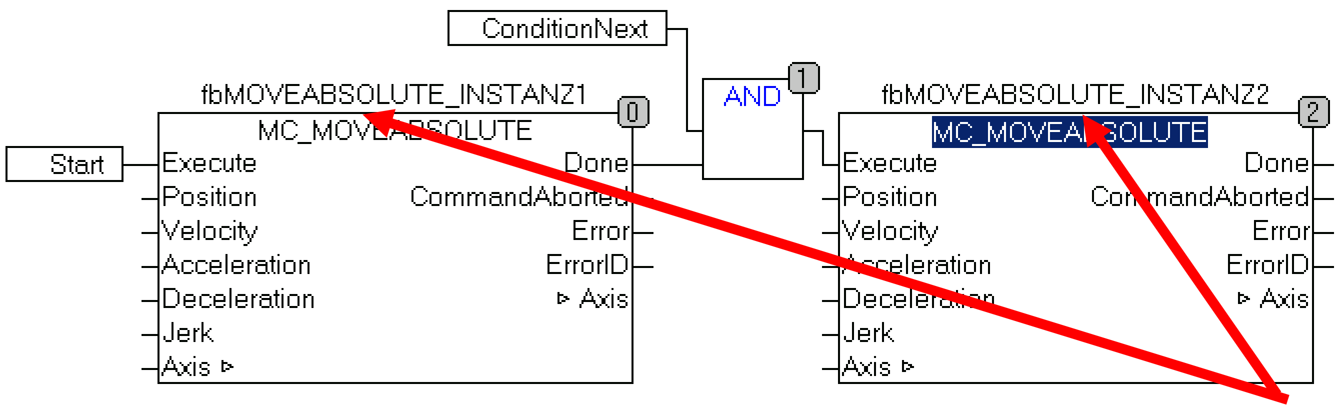


How can a flow be realised??

If the application requires flows, the MC blocks are normally used in sequence cascades.

The MC_XXXXX blocks are suited well for the use in Sequential Function Chart or in a case instruction in ST.

At graphic languages like CFC is in the first attempt a so-called cascading possible:



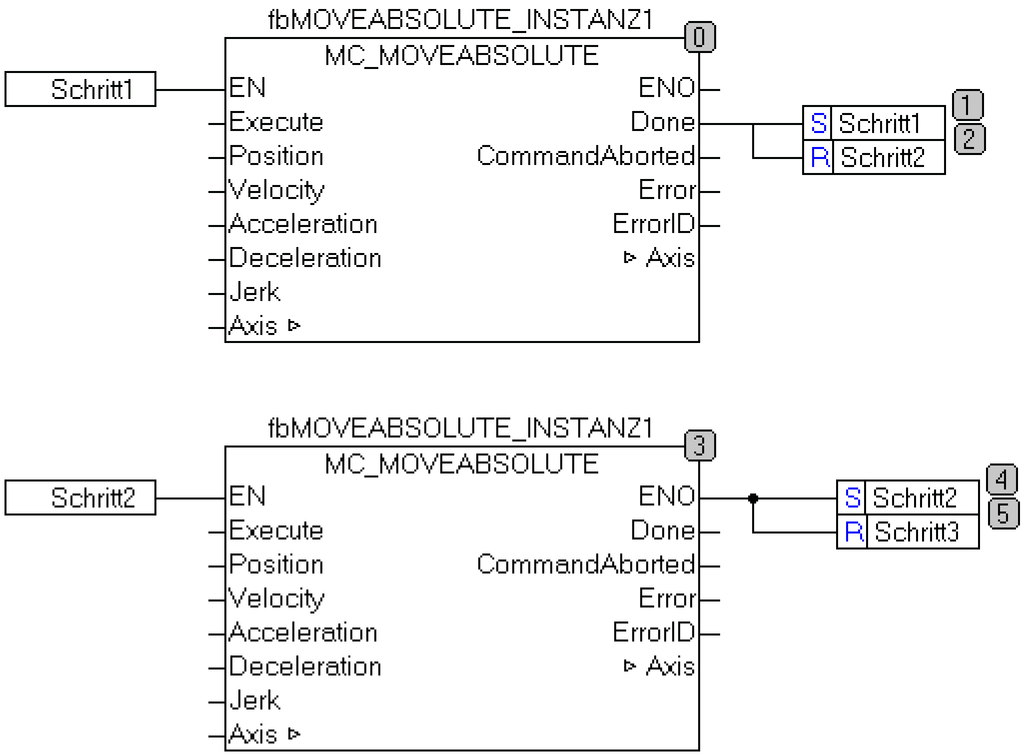
This acts reasonable, if for each command a new instance of the MC block is created

How can a flow be realised?

A further possibility is the using of the same instance with „EN” inputs which are controlled by step reminder.

**To consider:
The Fb accepts the next „execute” only, if there´s a flank at the input.**

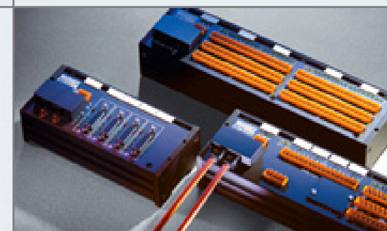
In addition „Disturbances in the flow“ like Command aborted and Error have to be considered.



Beckhoff-Training

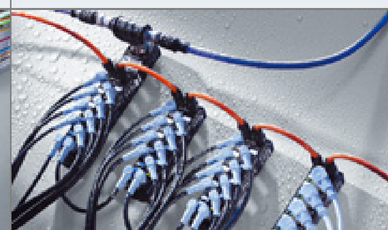
Beckhoff
Industrie-PC

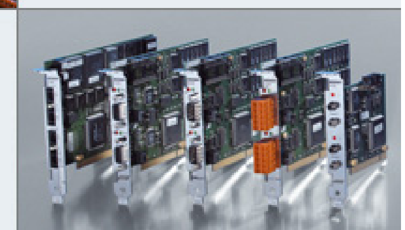
Beckhoff
Lightbus



Beckhoff
TwinCAT

Beckhoff
Embedded-PC





Beckhoff
Bus Terminals

Beckhoff
Fieldbus Boxes

Beckhoff
PC-Fieldbus
Cards, Switche

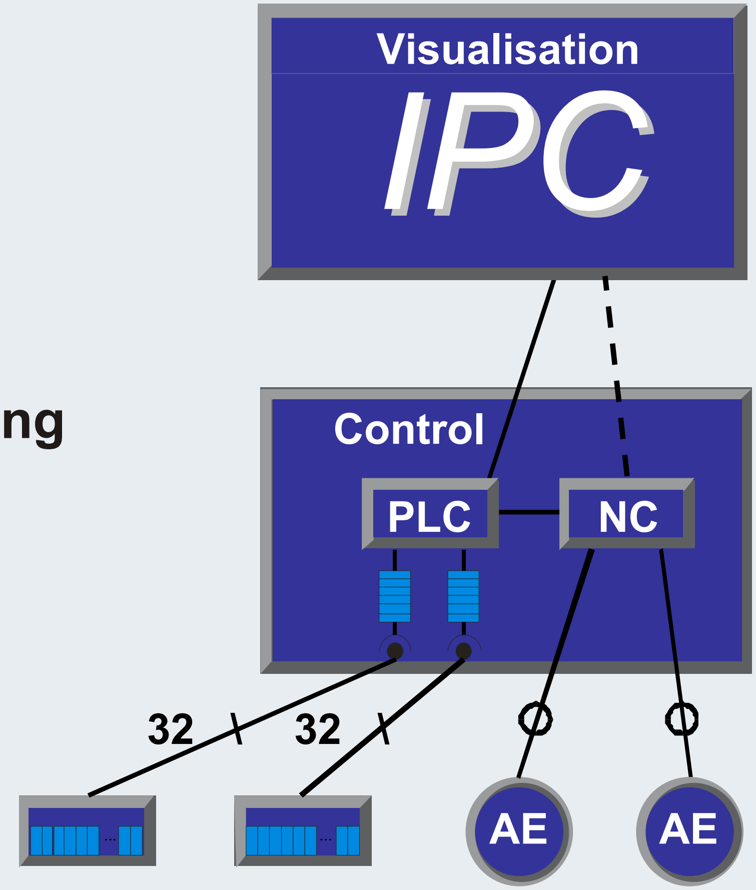
Beckhoff
EtherCAT

Beckhoff
Drive Control

Maintenance / Commissioning

Comparison of the structure: traditional and PC control technology - traditional PLC and NC

- Traditional PLC
  - Standard PLC with plug-in card
  - I/O via fieldbus or parallel
- NC drive control for PLC
  - Drive control on coprocessor
  - Position recording (s) and
  - velocity control (v) with parallel wiring
- PC in the automation
  - PC is used as master computer
  - runs the HMI program
  - is used for system networking
  - has the <u>most powerful</u> CPU of the 3 systems

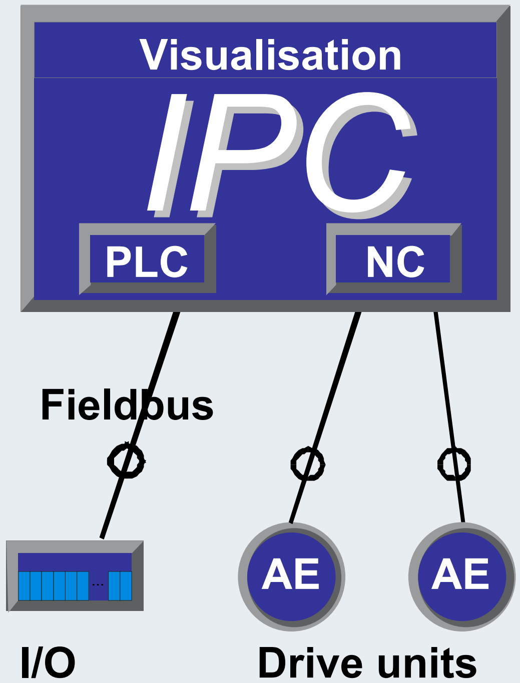


Comparison of the structure: traditional and PC control technology - PLC and NC as PC coprocessor

- Coprocessor PLC
  - Standard PLC as plug-in card
  - I/O via fieldbus

- NC drive control via coprocessor
  - Drive control on coprocessor
  - Position recording (s) and
  - velocity control (v) with parallel wiring

- PC in the automation
  - PC continues to be used as master computer
  - runs the HMI program
  - is used for system networking
  - has the <u>most powerful</u> CPU of the 3 systems

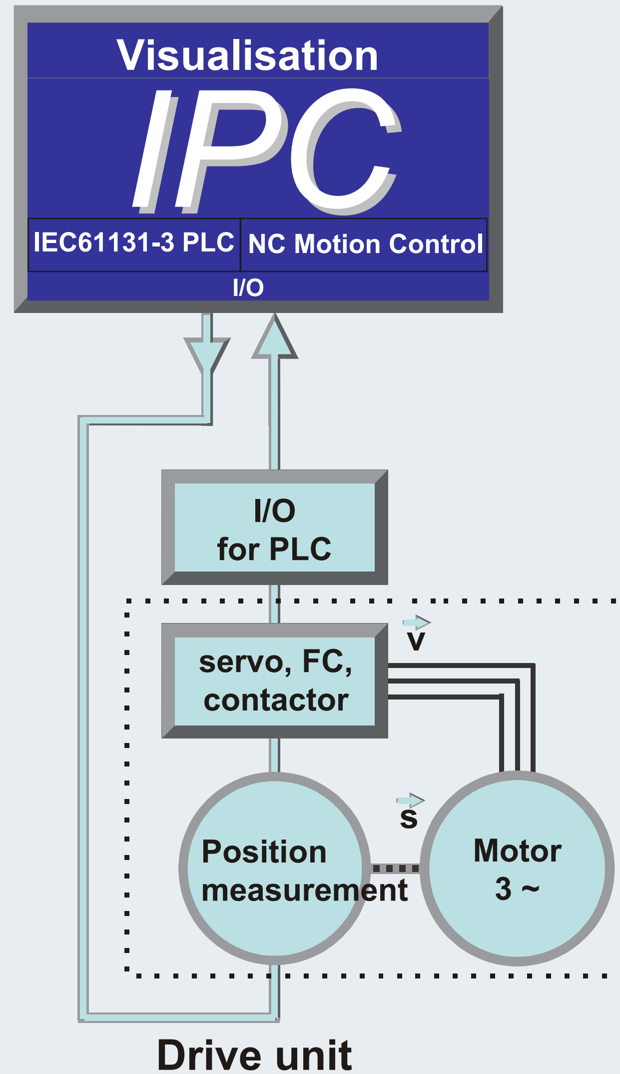


Comparison of the structure: traditional and PC control technology - PLC and drive control on the PC

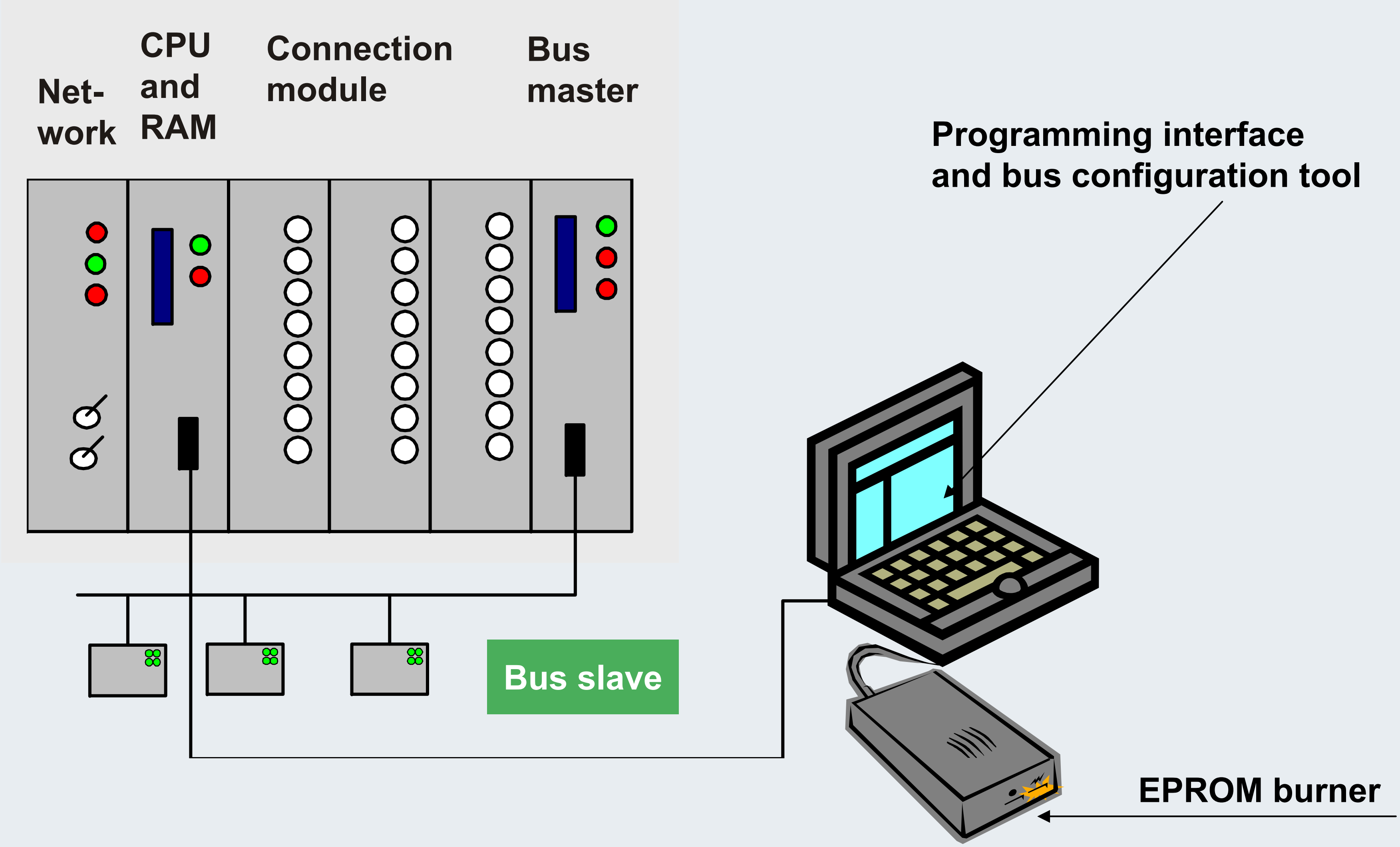
- PLC on the PC
  - Software PLC with <u>hard</u> real-time behaviour
  - I/O via fieldbus, all standards

- NC drive control on the PC
  - Drive control on PC processor
  - Position recording (s) and velocity control (v) are handled in the position control cycle via the fieldbus

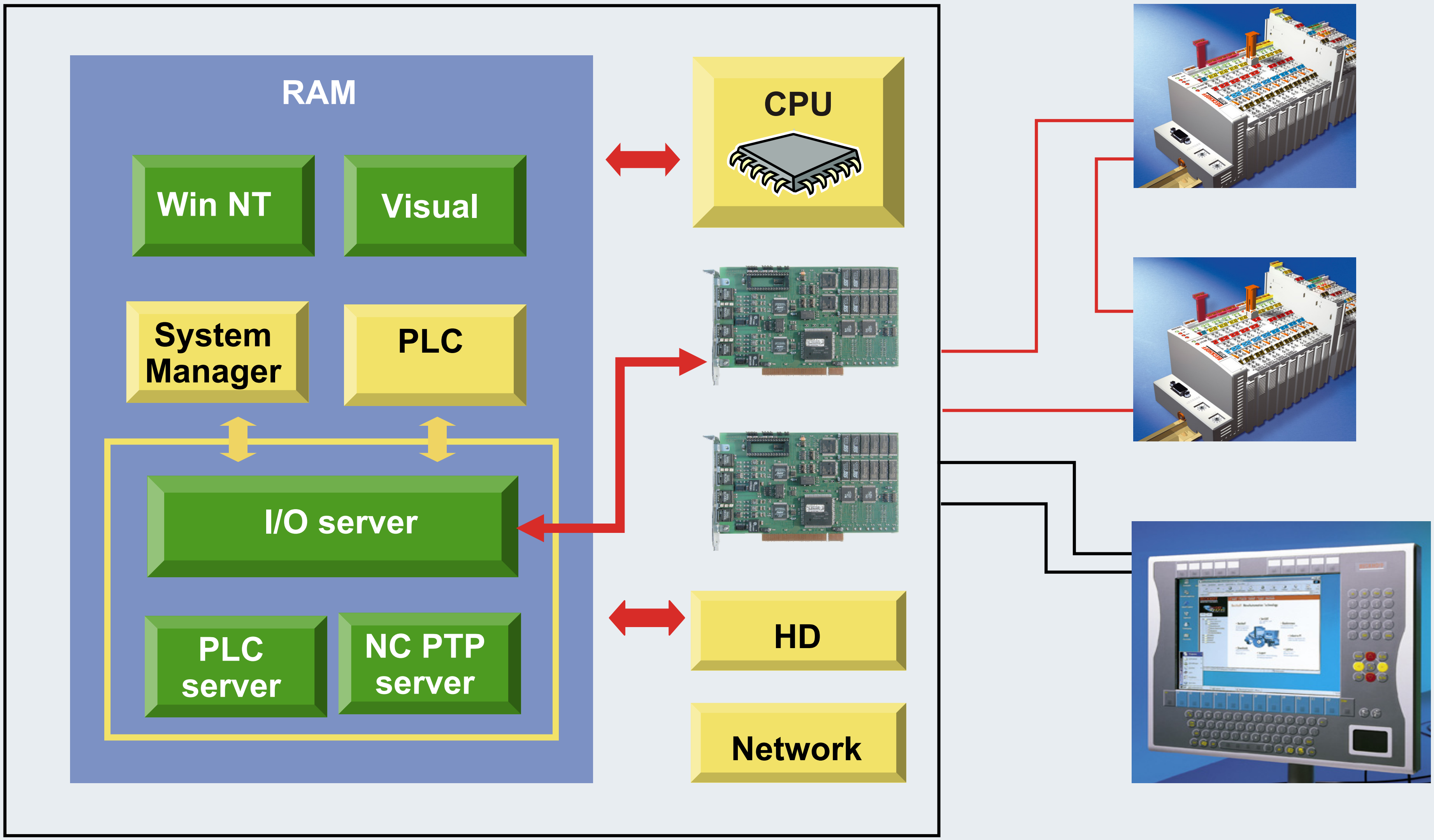
- Benefits
  - Central execution enables flexible configurations
  - Effective solution, no additional interfaces are required
  - Almost unlimited memory space for programs and data



Hardware PLC



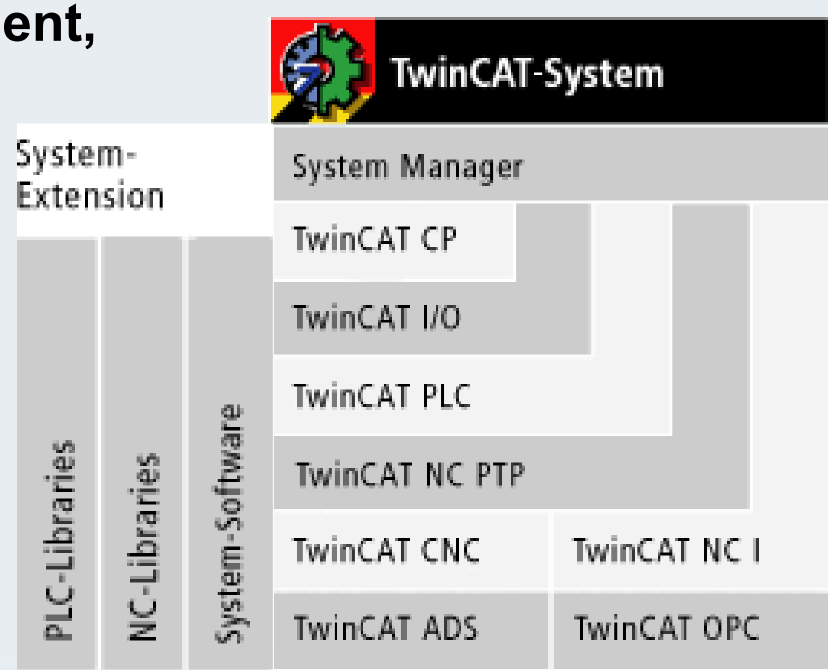
TwinCAT: Soft-PLC



System software product overview

TwinCAT - "The Windows Control and Automation Technology"

- **The TwinCAT automation software is a complete automation solution for PC-compatible computers.**
- **TwinCAT extended any compatible PC with: real-time control, multiple IEC 61131-3 PLC, NC positioning, programming environment, user interface.**
- **TwinCAT combines real-time control capability with the open and worldwide largest software platform, i.e. Windows operating systems.**



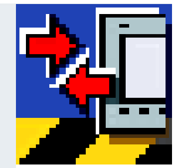
Synopsis: TwinCAT System Manager

The TwinCAT System Manager is the configuration centre for the system:
It establishes relationships between the number and programs of PLC systems, the axis control configuration, and the connected I/O channels.

- **Variable-oriented connection of I/O devices and tasks**
- **Variable-oriented connection between tasks**
- **The smallest unit is one bit**
- **Synchronous or asynchronous relationships are supported**
- **Data regions and process images are exchanged consistently**

SERCOS interface
LIGHTBUS
DeviceNet
INTERBUS Certified! No. 099
PROFIBUS® PROCESS FIELD BUS
CANopen
ControlNet™

Synopsis: TwinCAT CP



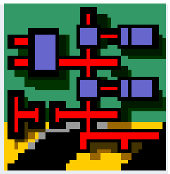
- TwinCAT CP is a driver for Beckhoff Control Panels, the industrial operating and display devices for the PC world.
- Control Panels are optimised for use as a man-machine interface.
- Operating and display elements create an independent unit, separated from the PC by a simple cable link.
- TwinCAT CP establishes the driver connection between general Windows programs and Beckhoff operating and display elements:
  - direct switches for fast machine functions
  - switch feedback by LEDs
  - UPS support

Synopsis: TwinCAT I/O

Direct access from Windows programs to fieldbusses and PC.

- **Online diagnosis with force option simplifies commissioning.**
- **All signals are processed variable-oriented for real-time or Windows programs.**
- **DLL/OCX offers fast real-time access from Windows programs.**
- **Supports Beckhoff Lightbus, PROFIBUS DP, Interbus, CANopen, DeviceNet, SERCOS, and PC hardware.**

Synopsis: TwinCAT PLC



- **Software PLC for Windows NT/2000/XP**
  The TwinCAT PLC is programmed in accordance with IEC 61131-3 independently of the manufacturer. Online connections with PLC runtime systems can be implemented worldwide via TCP/IP or fieldbus on the IPC.

- **TwinCAT PLC programming system**
  TwinCAT PLC offers all languages defined in IEC 61131-3 standard. TwinCAT PLC has a powerful 32-bit development environment for programs whose code size and data regions far exceed the capacities of conventional PLC systems.

TwinCAT PLC: practical features

- all defined programming languages: IL, FBD, LD, SFC, ST and CFC
- certified in accordance with base level (IL/ST)
- structured programming with modular program management
- recompilation while PLC running with maximum data retention (online change)
- Source code is stored in the target system
- criterion analysis
- convenient library management
- conversion between languages
- incremental compilation
- all common data types, structures, arrays, including multi-dimensional arrays
- programming support:auto-format, auto-declare, cross reference, search/replace
- convenient project comparison

TwinCAT PLC: Debugging - facilities

- online connection with PLC runtime system worldwide via TCP/IP or fieldbus
- online change of new variables, instances or programs at run-time with maximum data retention
- online monitoring of variables in variable lists, watch windows, editors
- online status and powerflow (accumulator contents) of programs and instances
- triggering, forcing and setting variables
- single step, breakpoints
- display of the current call stack
- watch list shows a selection of variables
- trace function precisely records variable cycle
- online management of all variable names and structures across the whole system
- TwinCAT ScopeView as a graphical diagnostic and analysis tool for the display of values

Synopsis: TwinCAT NC PTP

- **Position control with the PC**
  TwinCAT NC PTP includes axis positioning software (set value generation, position control), an integrated software PLC with NC interface, operating program for commissioning and an I/O connection to the axes through various fieldbusses. TwinCAT NC PTP replaces conventional positioning modules and NC controllers.

- **NC PTP software on the PC**
  The position controller is calculated on the PC processor and cyclically exchanges data via the fieldbus with drives and measurement systems. The capacity of a PC allows axes to be moved in parallel with the PLC functionality. PC performance means that some tens of axes can easily be positioned simultaneously.

Synopsis: TwinCAT NCI

Follow the path with the PC

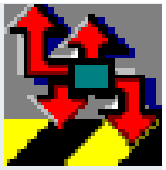
- **The TwinCAT NC Interpolation (NC I) is the NC system for interpolated path movements.**
- **TwinCAT NC I offers 3D interpolation (interpreter, set point generation, position controller), an integrated PLC with an NC interface and an I/O connection for axes via the fieldbus.**
- **All well known fieldbus systems and programming standards in the CNC world, such as DIN 66025, are supported.**
- **PLC blocks are additionally available with which axes can be controlled through an interpolation procedure using a table type description.**
- **TwinCAT NC I substitutes open PC solutions for standard axial components and CNC controls.**
- **TwinCAT NC I utilises PC performance and enables axis control under Windows NT/2000/XP. Hardware components are recreated in software and thus replaced.**

Synopsis: TwinCAT CNC

Complex tasks - new solutions

- TwinCAT CNC offers complete CNC functionality as a pure PC-based software solution.
- TwinCAT CNC covers the complete range of classic CNC path control, including high-end systems for complex motion and kinematics requirements.
- The powerful, continuously evolving PC platform and the hard real-time base of the TwinCAT real-time kernel offer ideal preconditions for software CNC.

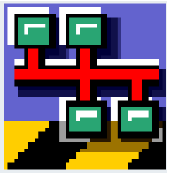
Synopsis: TwinCAT ADS OCX / DLL



Access to TwinCAT functions and data:

- **The TwinCAT system can be integrated via TCP/IP connections,**
- **as ActiveX Control (OCX) or DLL,**
- **for visualisation, SCADA and Office applications such as Excel**
- **Programming languages:Visual Basic, VBA, Visual C++, Delphi,**
- **suitable for all TwinCAT levels and also TwinCAT BC.**

Synopsis: TwinCAT OPC



Standard interface for automation, data exchange via OPC server:

- **Connection to Windows programs, e.g. visualisation, SCADA and Office applications,**
- **simple configuration via file import,**
- **monitoring of variables in the OPC server**
- **Data transfer via local or remote servers**

IBK - T1

Which system software running TwinCAT offers drivers for the Beckhoff Control Panel?

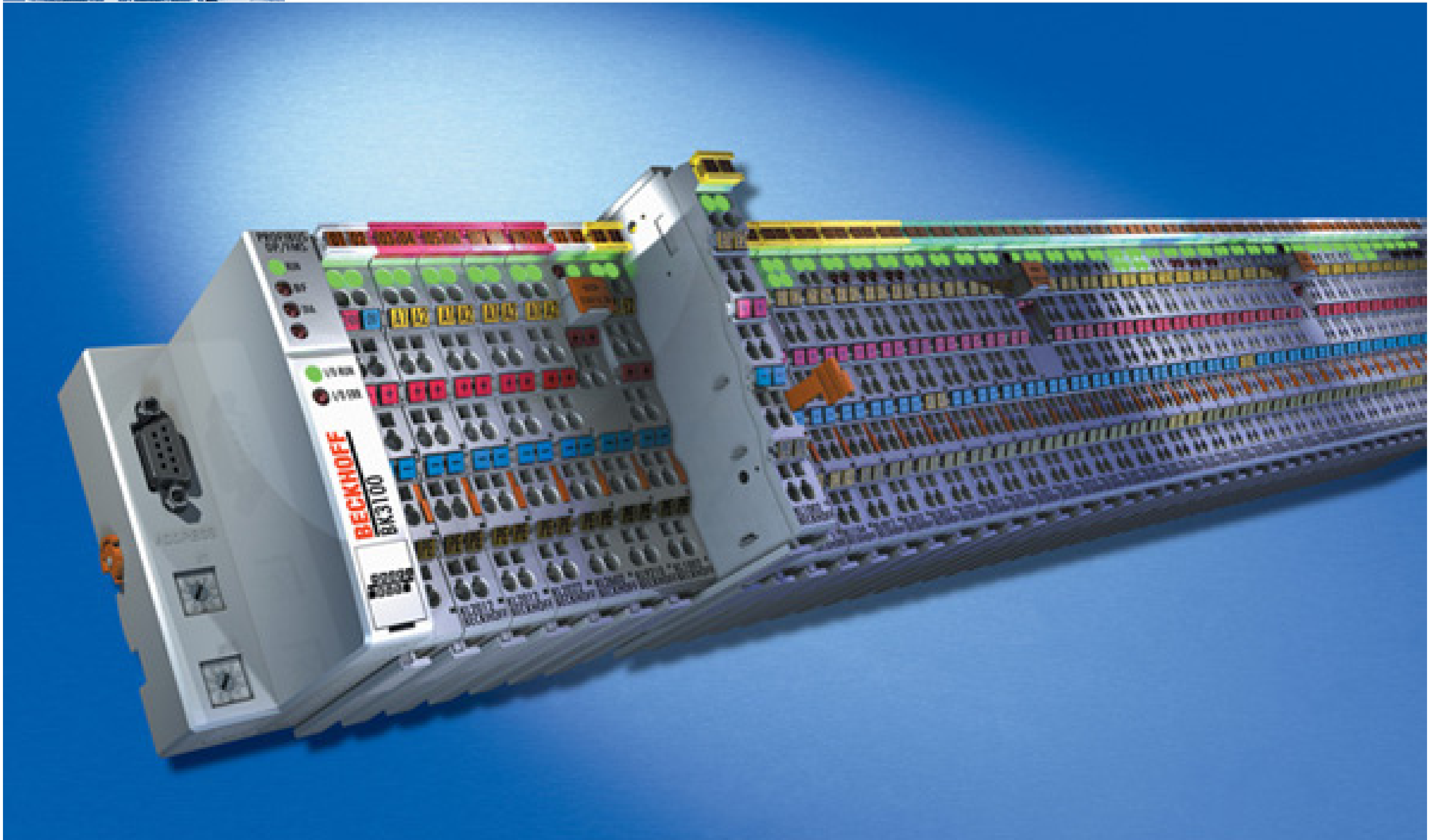
Which operating systems are compatible with TwinCAT?

Which programming languages does TwinCAT offer?

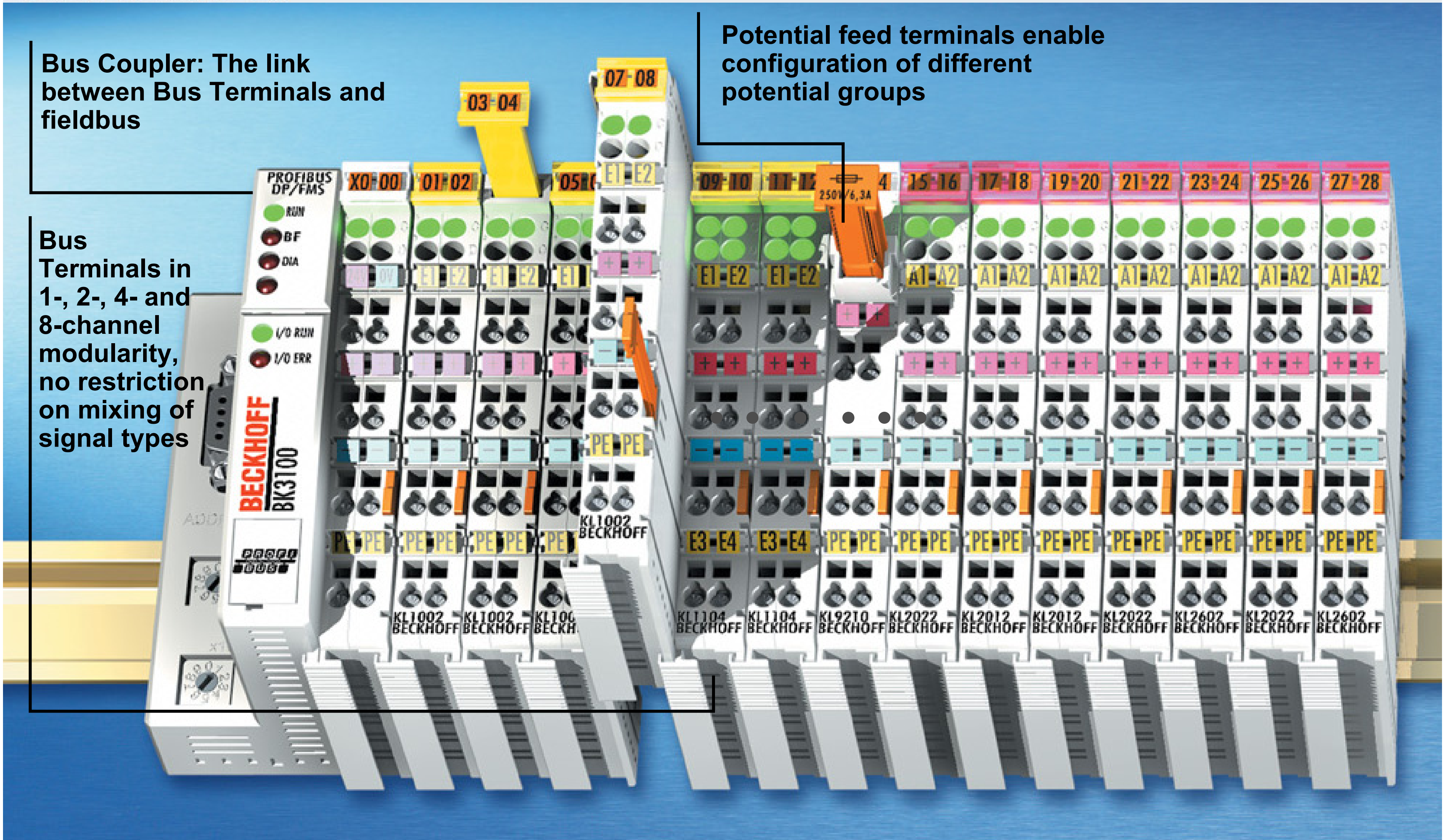
Which system software is axis positioning based on?

Which system software can be used for programming a Bus Terminal controller?

The Beckhoff Bus Terminal system



The Beckhoff Bus Terminal system



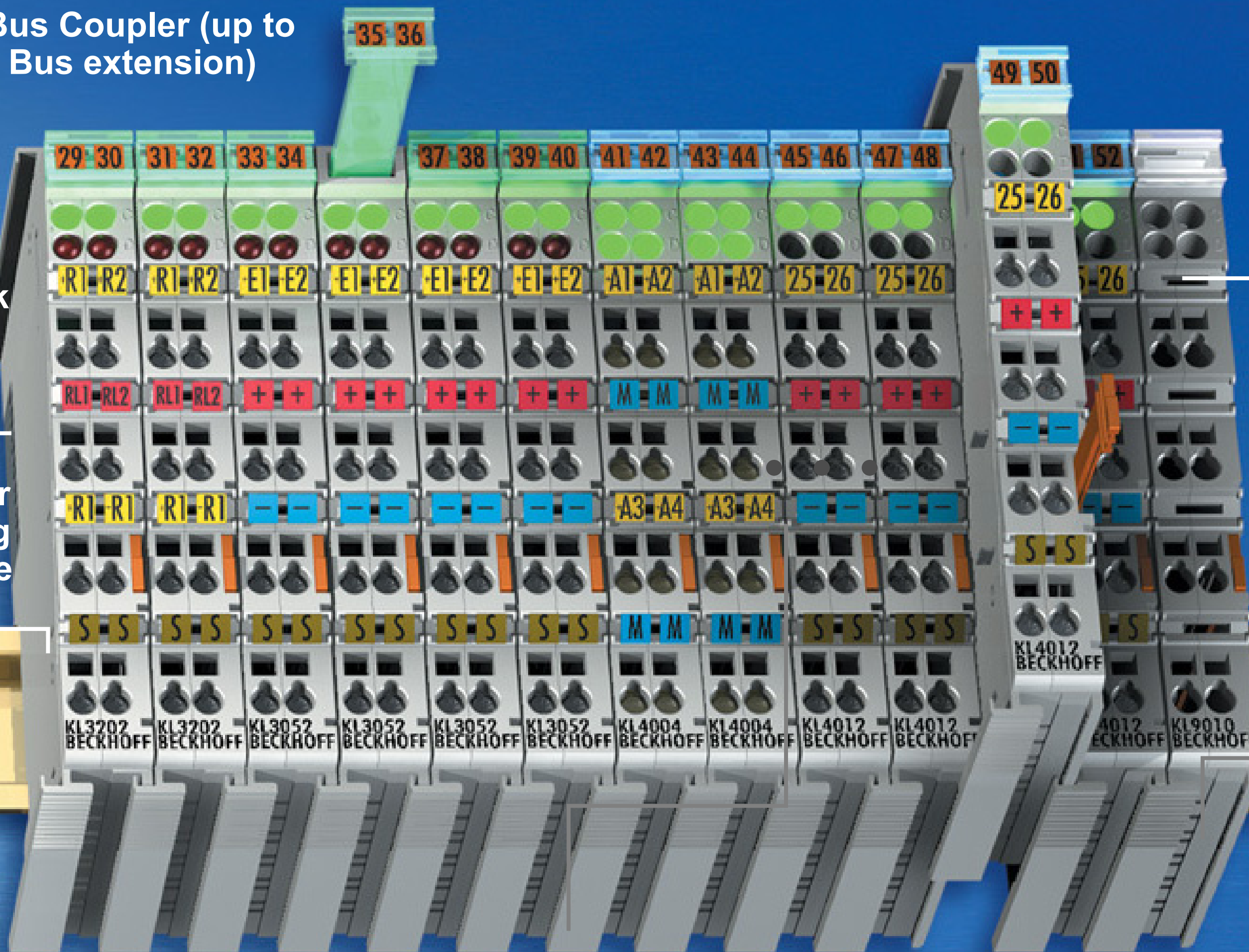
The Beckhoff Bus Terminal system

Between 1 and 64 Bus Terminals can be operated at a Bus Coupler (up to 255 with Terminal Bus extension)

Fast, safe data link via serial Terminal Bus

Power contacts for automatic bridging of the performance potential

Bus end terminal

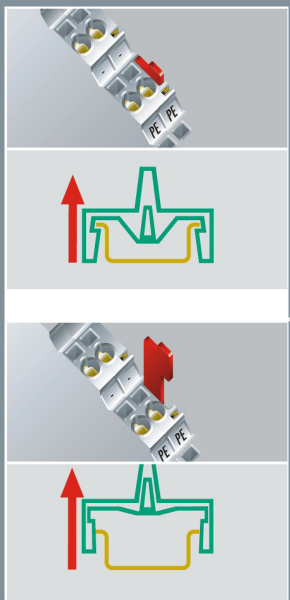


Bus Terminal system characteristics

The electronic terminal blocks are clipped onto the Bus Coupler. The contacts are made as the terminal clicks into place, without any other manipulation. This means that each electronic terminal block can be individually exchanged. It can be placed on a standard mounting rail. As well as horizontal fitting, all other fitting methods are permitted.

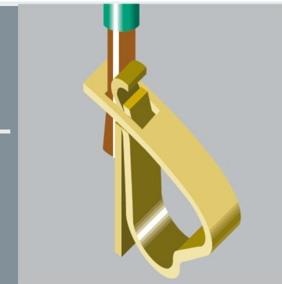
The outside contour of the Beckhoff Bus Terminals fits the dimensions of terminal boxes with technical perfection. An informative connection panel having LEDs for status display, push-in contact labelling and removable labelling areas ensures clarity in practice. The three-wire system, with an additional connection for a protective conductor, make it possible to wire sensors and actuators directly.

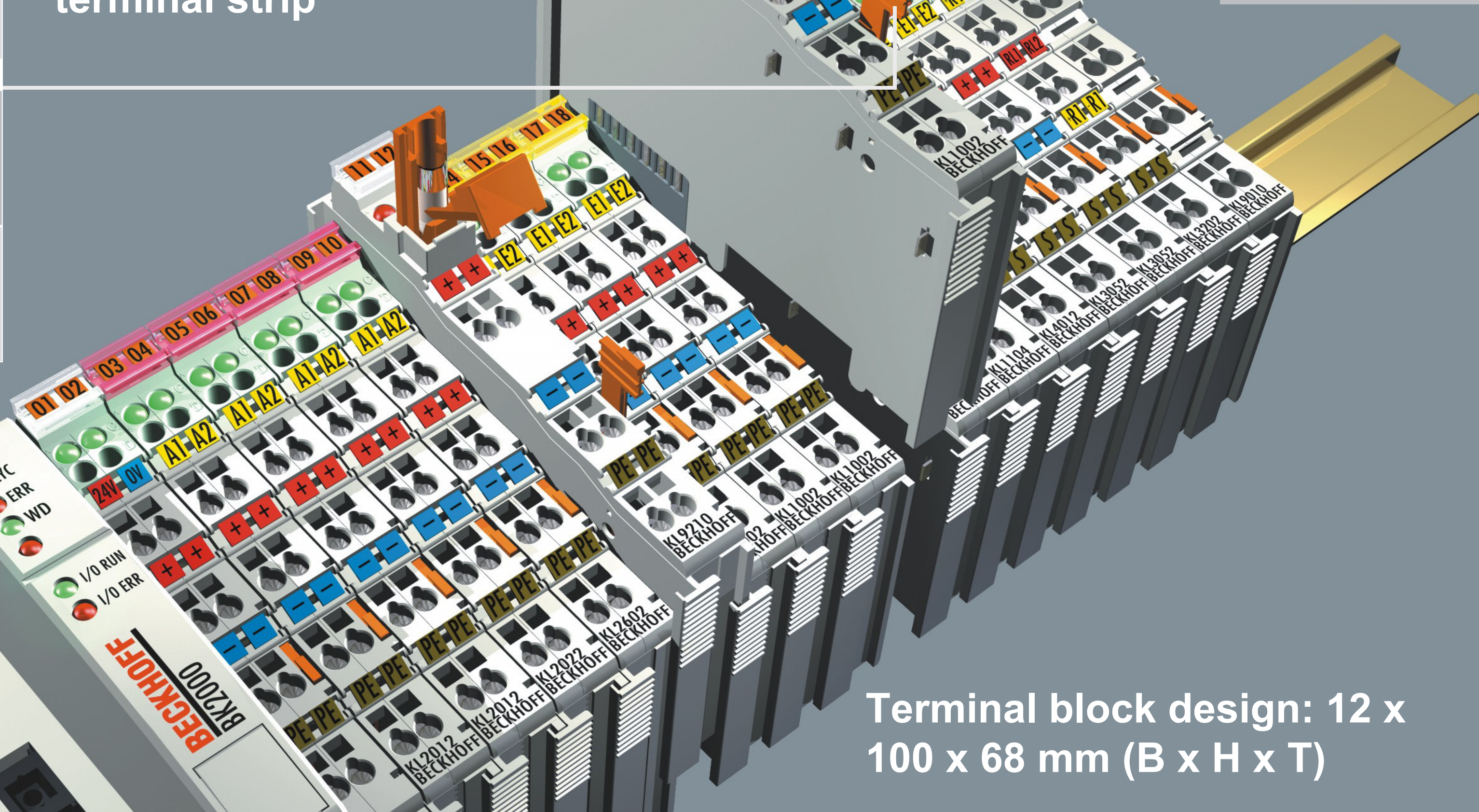
Bus Terminal Features



Symmetrical release permits vertical removal from the terminal strip

Screwless connection technique

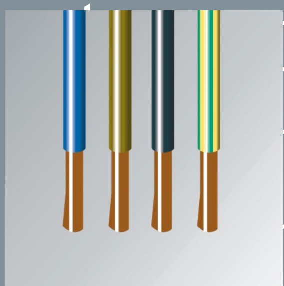


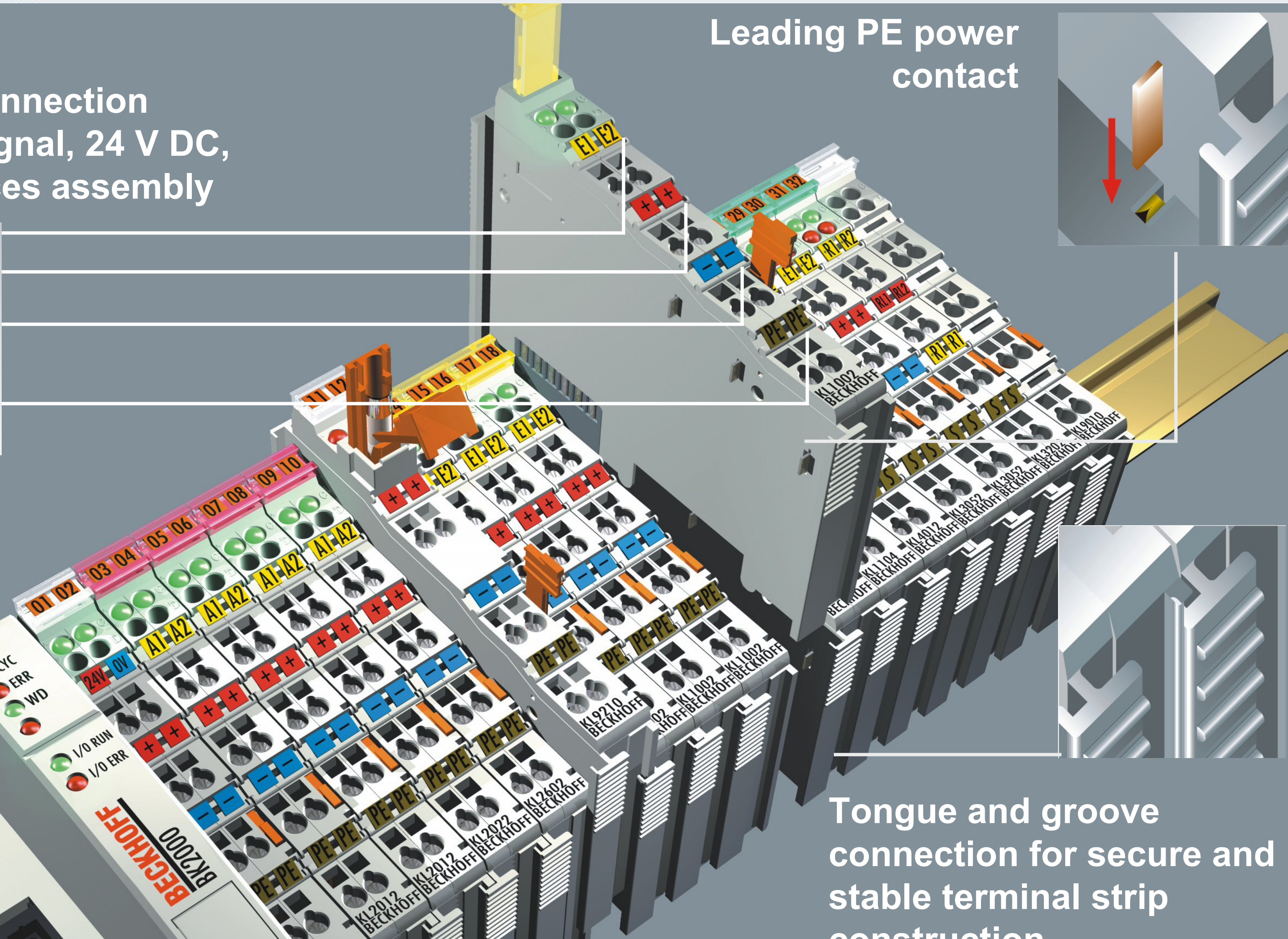


Terminal block design: 12 x 100 x 68 mm (B x H x T)

Bus Terminal Features

The 4-wire connection technique (signal, 24 V DC, 0V, PE) reduces assembly

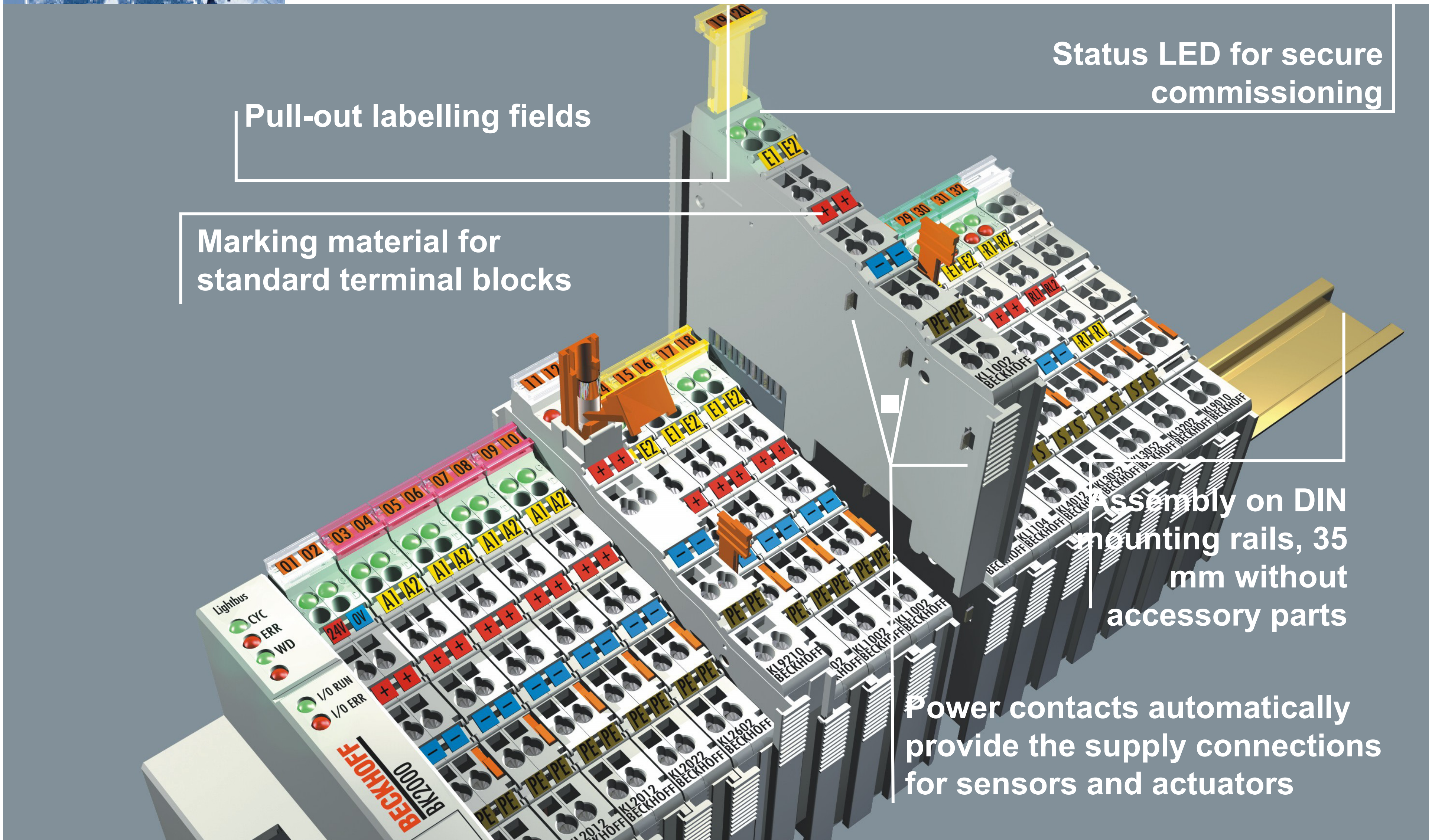




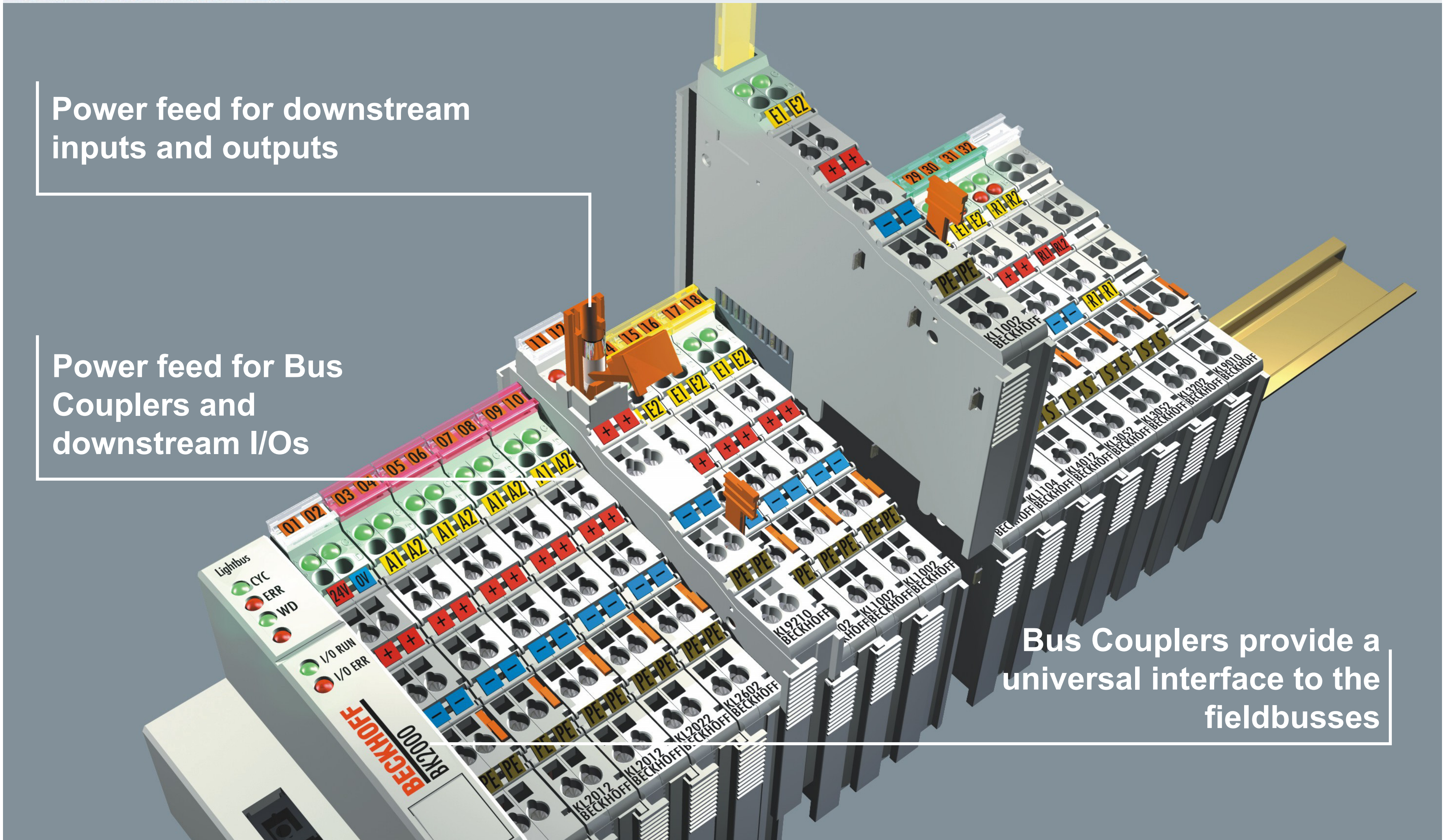
Leading PE power contact

Tongue and groove connection for secure and stable terminal strip construction

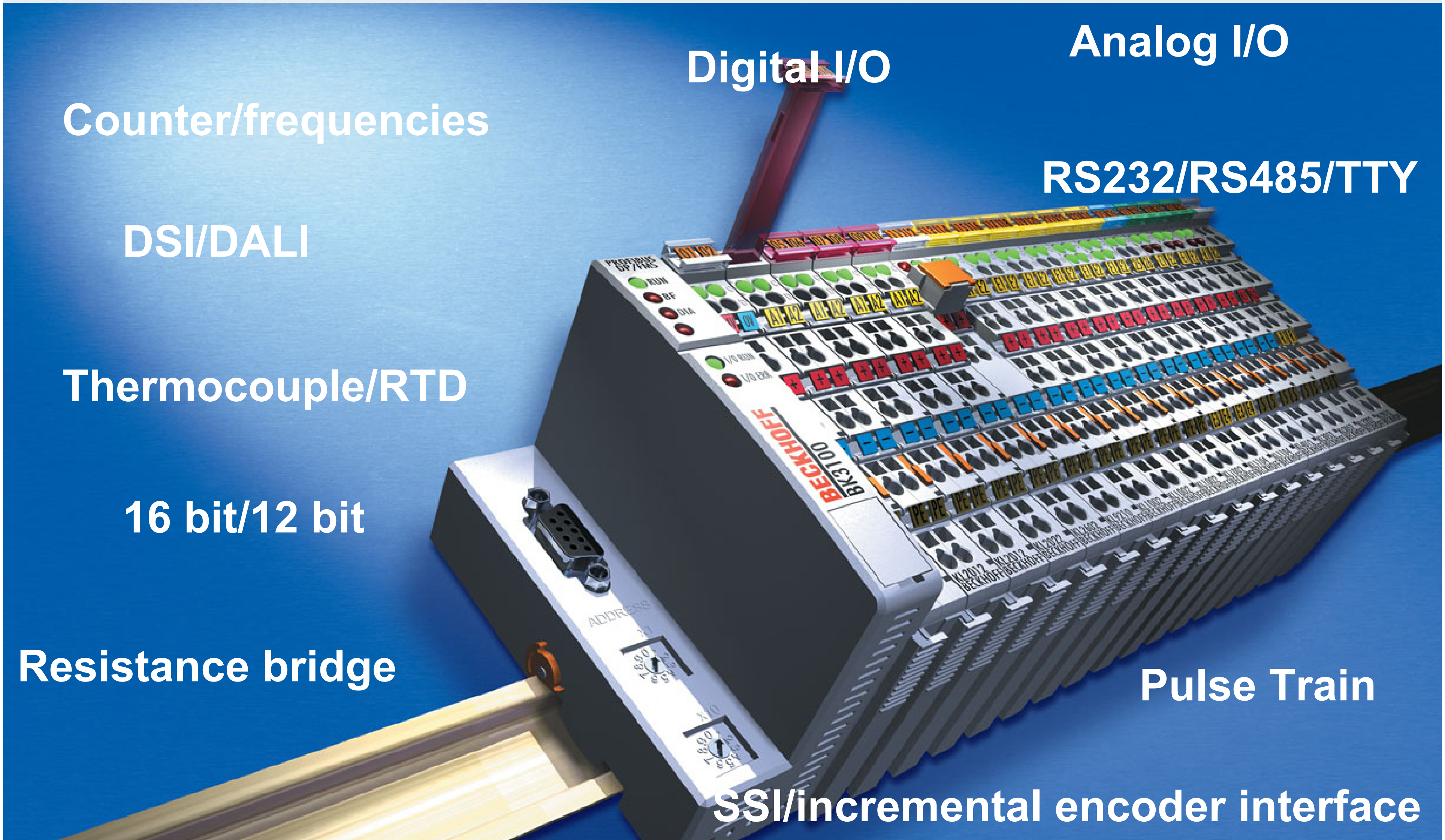
Bus Terminal Features



Bus Terminal Features



Free mix of signals



Free mix of signals

The Beckhoff Bus Terminal components allow the user to operate an unrestricted assortment of signals at each station.

In addition to the digital input/output terminals with two channels, in which the 24 V DC outputs of a terminal can be loaded right up to 2.0 A, there are terminals for analog signal forms.

Terminals are available for current and voltage with standardised signal levels, and also for PT100 and thermocouple signals.

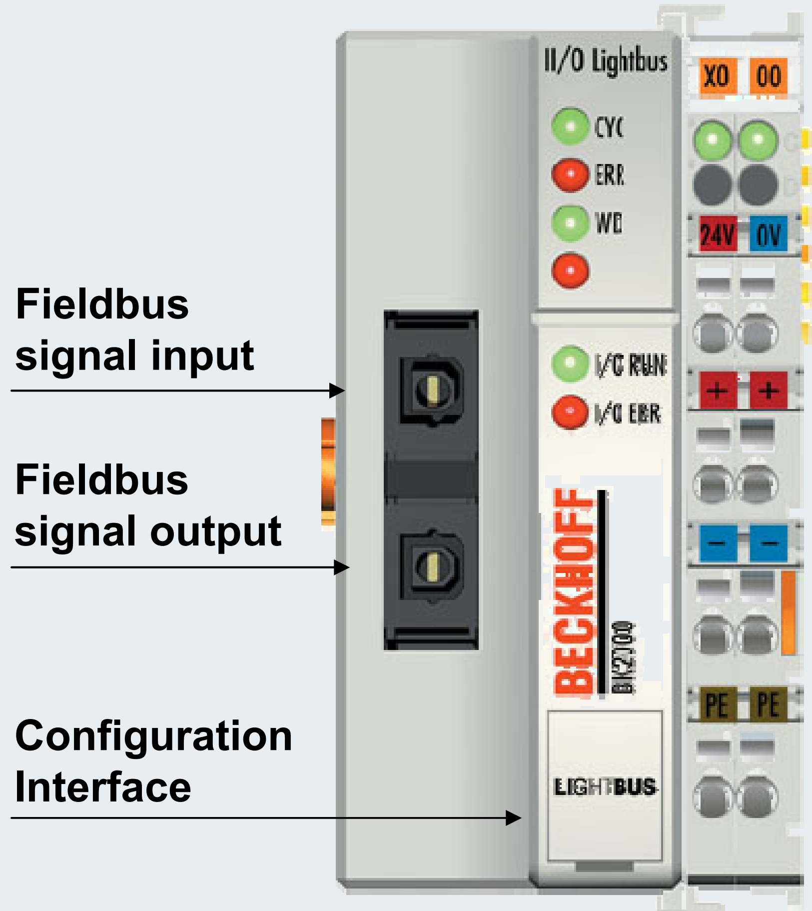
Intelligent devices can be connected via Terminals with serial interfaces in accordance with RC 232C, RS 485 or 20 mA TTY.

Lightbus - Bus Couplers BK2000, BK2010 and BK2020

The Lightbus Bus Couplers link the Lightbus system with the modular electronic terminal blocks.

One unit consists of one Bus Coupler, any number from 1 to 64 terminals and one end terminal.

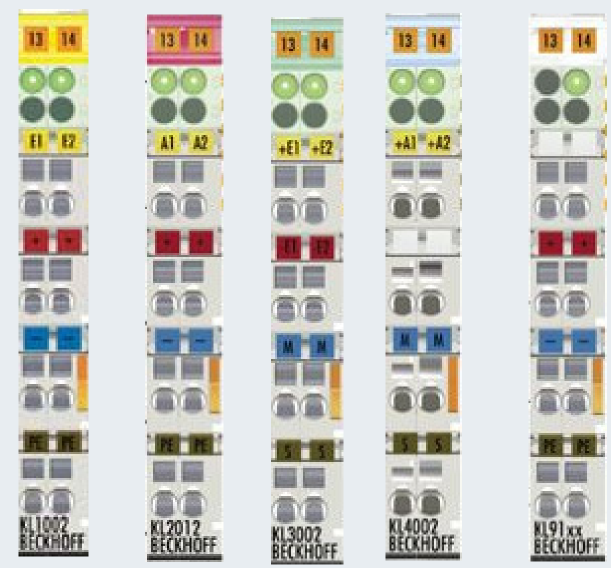
The BK2010 economy variant permits particularly economical creation of peripheral interfacing connections. Up to 64 digital input/output terminals can be connected.



Selection of local terminals

Terminals can be selected from the product catalogue based on the technical description.
The colour pull-out label field and the index number at the bottom of the terminal front provide information about the terminal type

- **Yellow: digital input terminal (KL1xxx)**
- **Red: digital output terminal (KL2xxx)**
- **Green: analog input terminals (KL3xxx)**
- **Blue: analog output terminal (KL4xxx)**
- **White: system terminal (KL9xxx)**



IBK – T2

What is the maximum number of terminals that can be connected to a bus station?

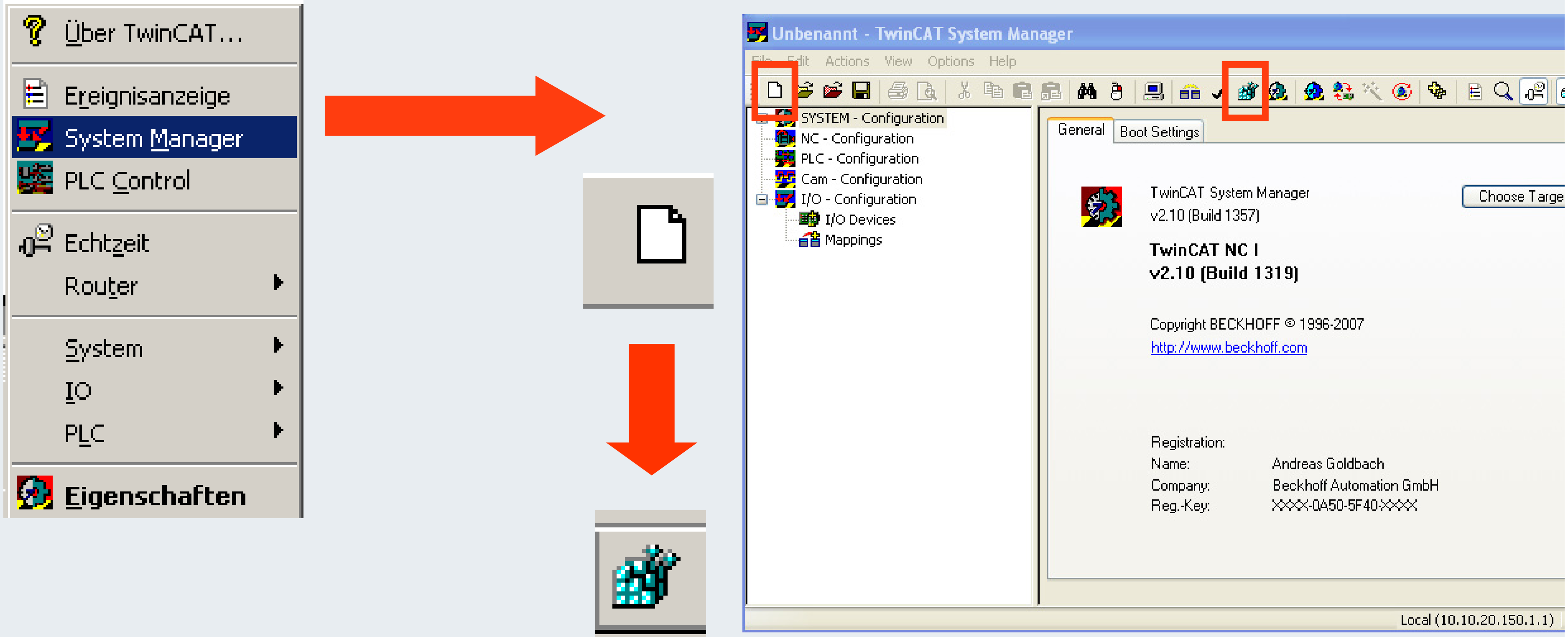
What is the maximum number of terminals that can be connected via a system expansion terminal?

In what position can the terminals be installed?

What is the name of the economy version of the Lightbus system for connecting up to 64 digital input/output terminals?

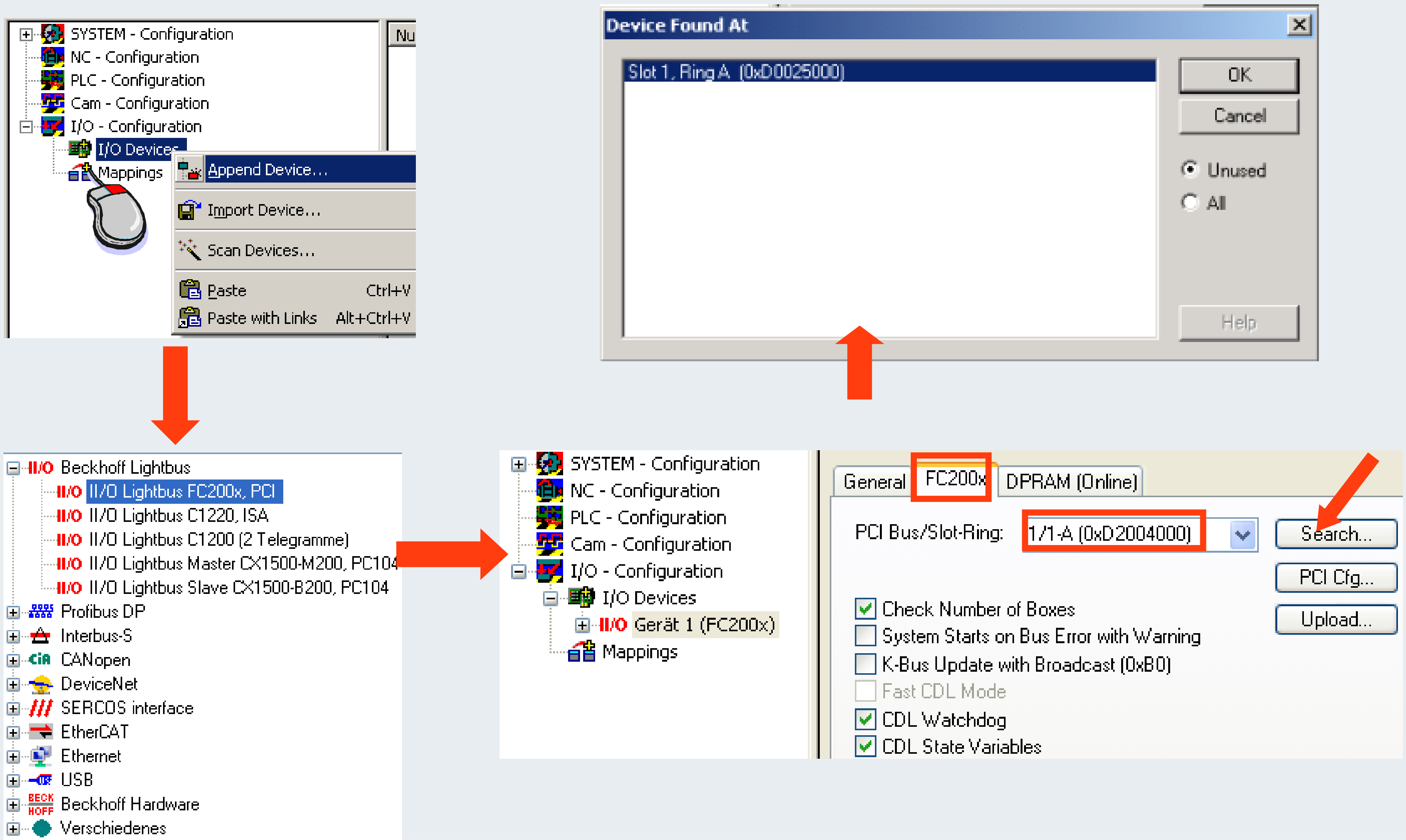
What is the index number identifying the group of digital output terminals?

Setting up an empty configuration in the System Manager

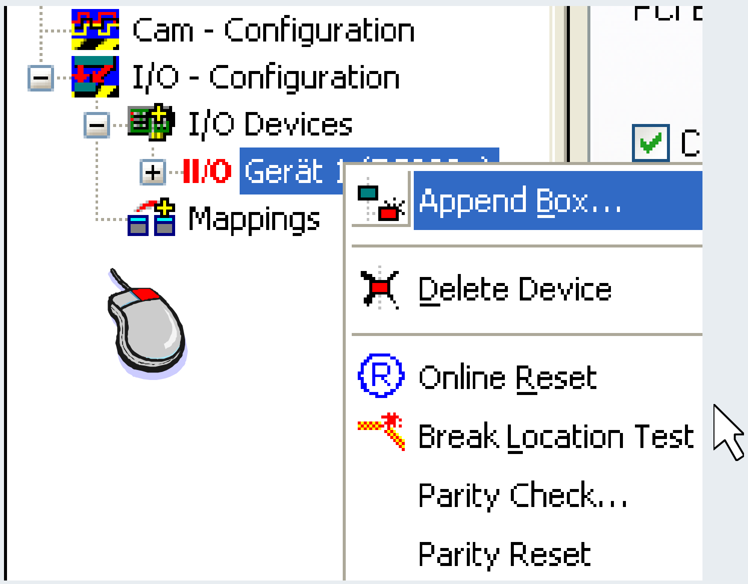


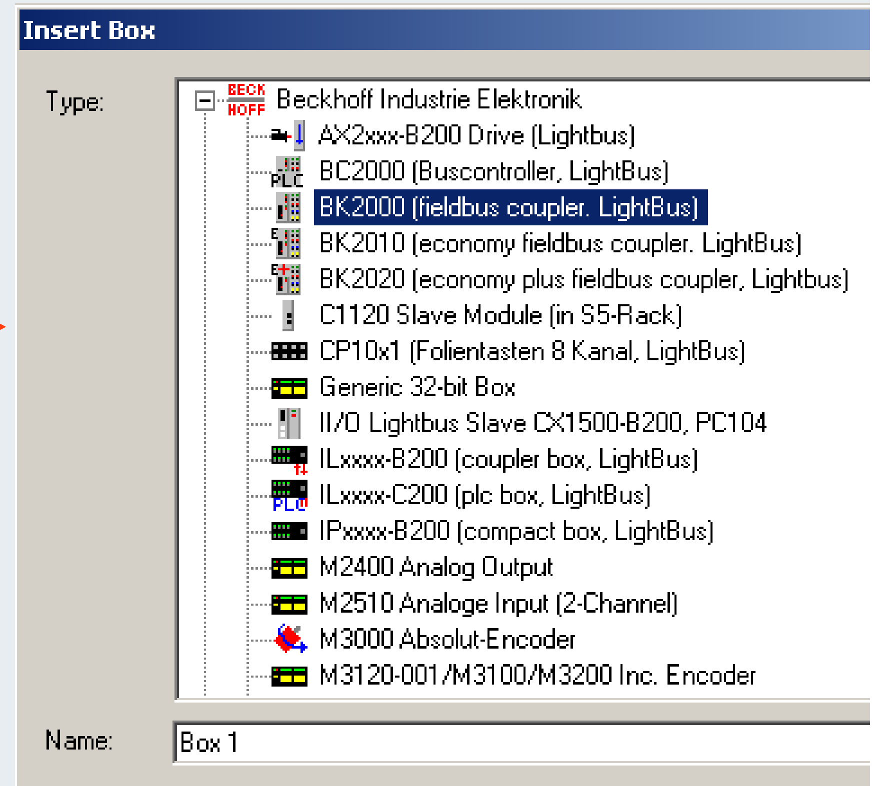
This step creates a defined start-up configuration in the System Manager, which prevents an existing mapping interfering with PLC program execution.

Example: first steps



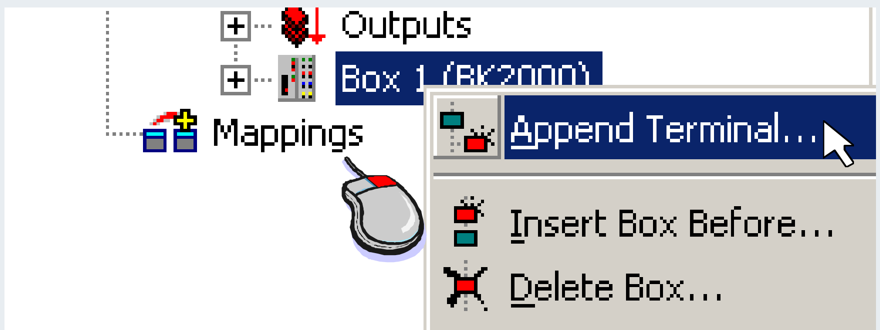
Integration of the slave in the System Manager

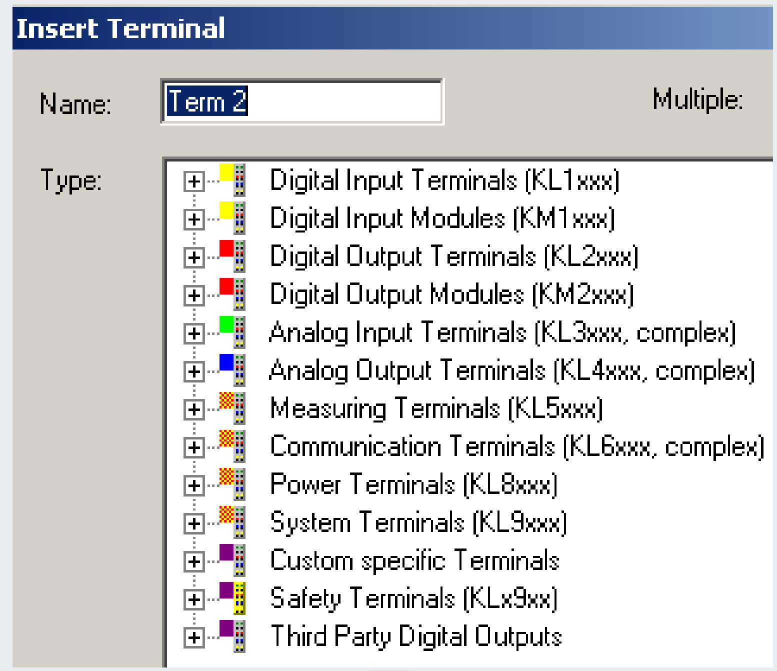


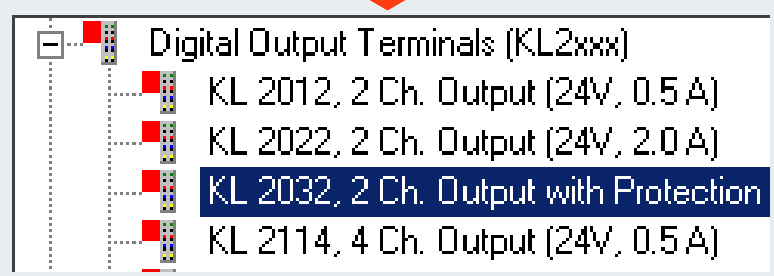


If the exact description of a module cannot be found in the list (e.g. M1400), a general 32-bit box is inserted.

Inserting terminals at the coupler



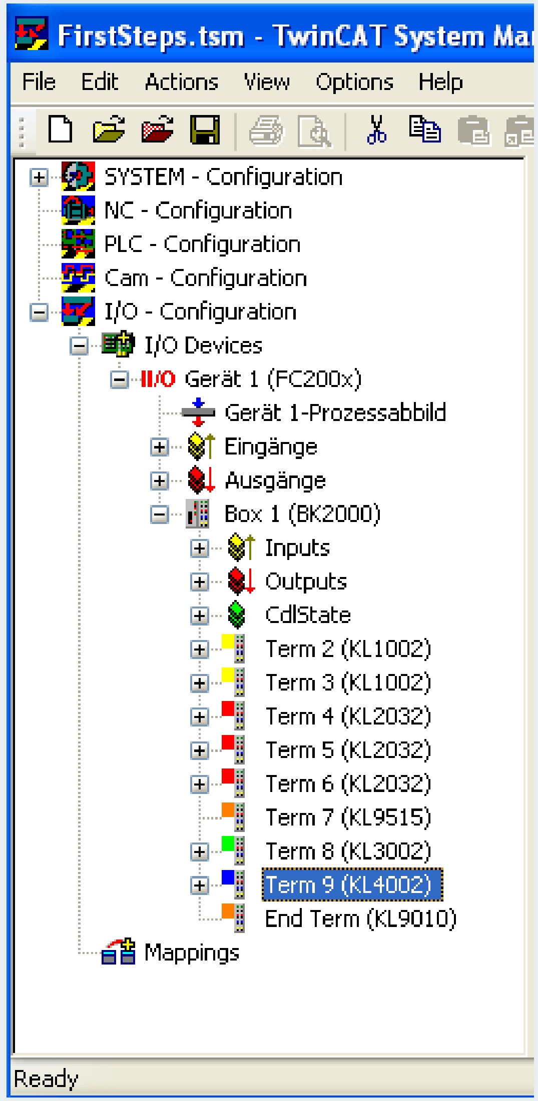


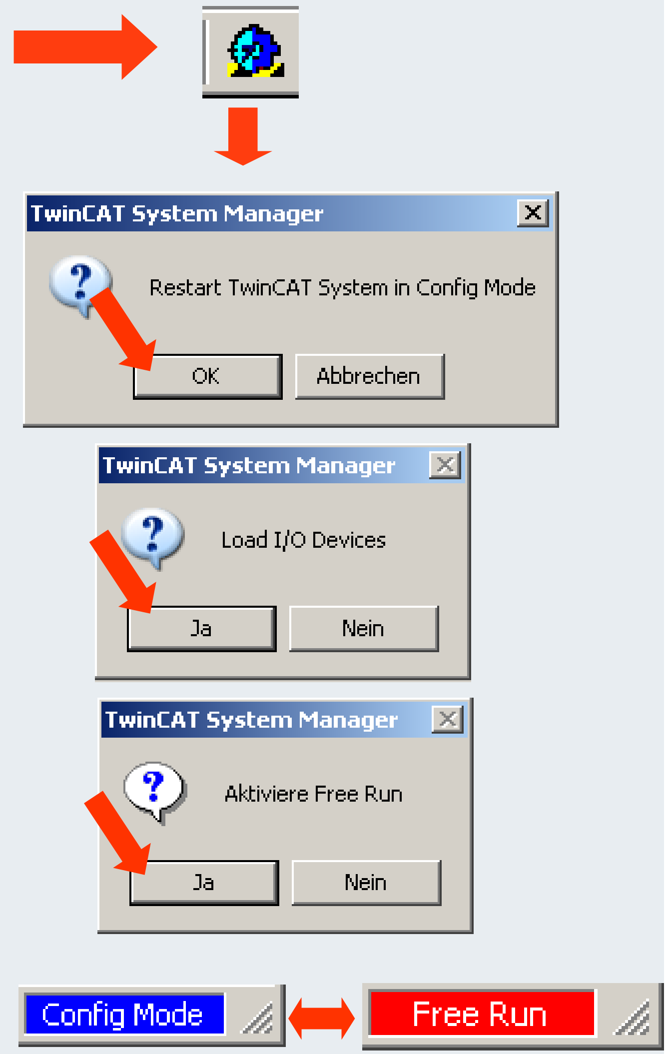


1. **Insert terminal**
2. **Select group**
3. **Locate and select terminal**
4. **Repeat steps 1 and 3 until all terminals have been entered**

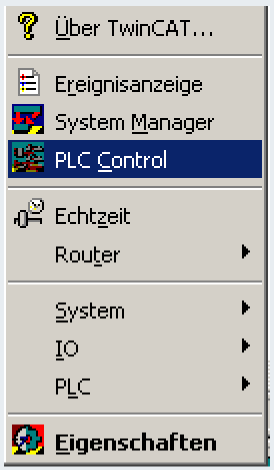
Activate free run

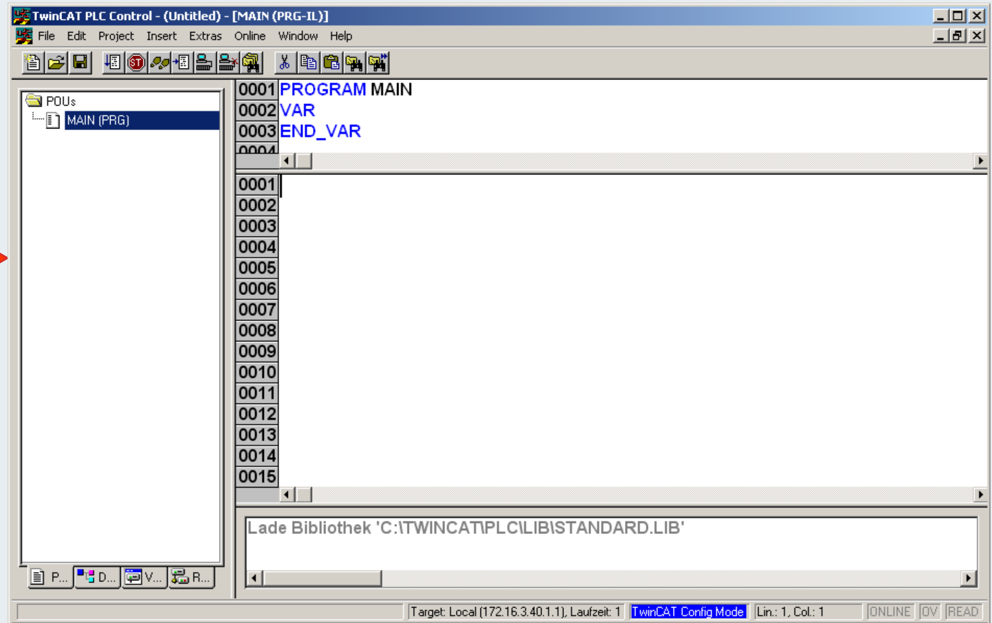
- First save the project via File/Save As
- Then switch to Config mode
- Then reload the I/O devices and activate free run
- Free run and Config mode flash alternately





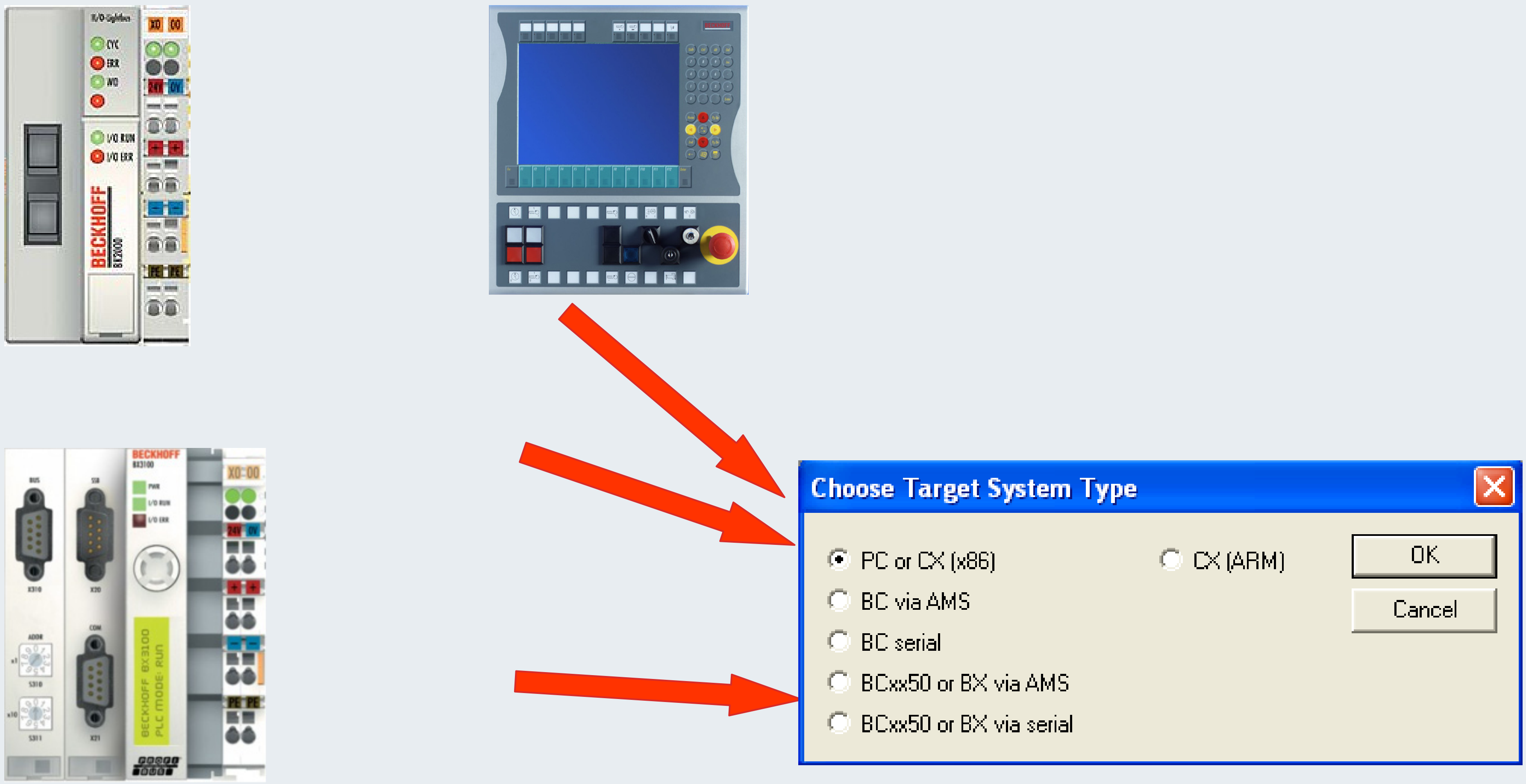
Creating a new project in PLC Control, part 1



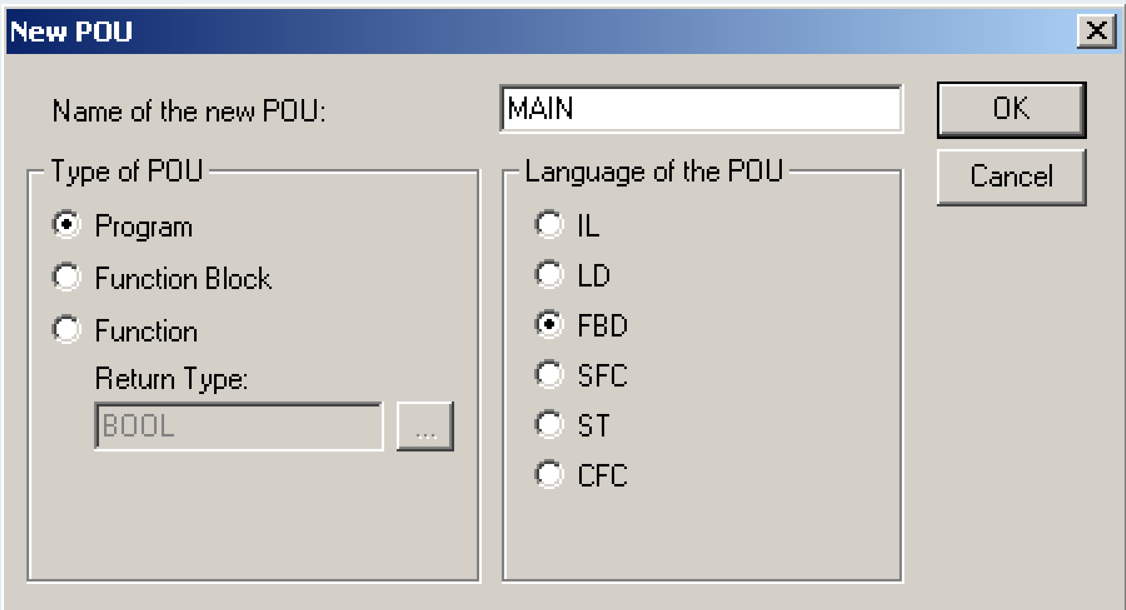


File New

Creating a new project in PLC Control, part 2

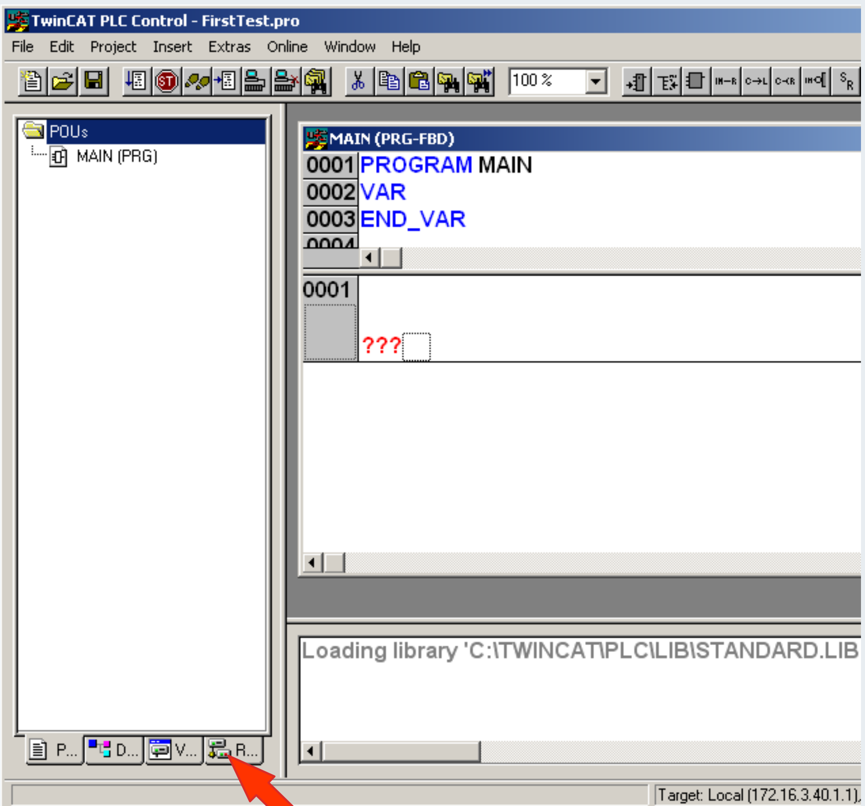


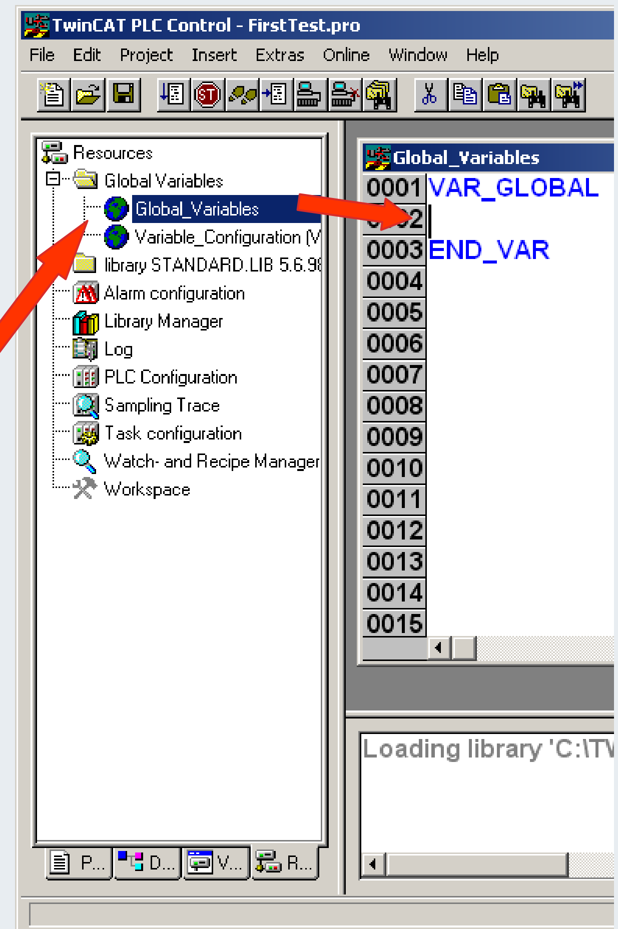
Creating a new project in PLC Control, part 3



IL: **Instruction List**
LD: **Ladder Diagram**
FBD: **Function Block Diagram**
SFC: **Sequential Function Chart**
ST: **Structured Text**
CFC: **Continuous function chart**

Creating a new project in PLC Control, part 4





Global Variables

Resources

Variable Declaration

Variable name

AT

%

I
Q
M

X
B
W
D

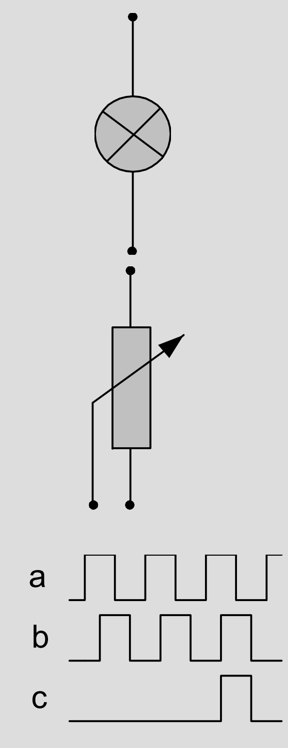
1.0
20

:

BOOL
BYTE
INT
REAL
TIME
STRING

;

Declaration example



Lamp AT %QX0.0 : BOOL;

Switch AT %IX0.0 : BOOL;

AnalogValue AT %IW2:INT;

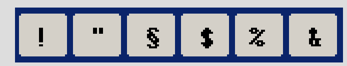
Temperature AT %IW100 :INT;

Counter AT %IB3 : UINT;

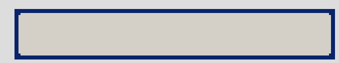
PWM_Output AT %QB10 :INT;

Naming restrictions

1. No special characters

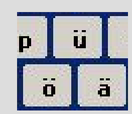


2. No spaces



3. Permitted for separation

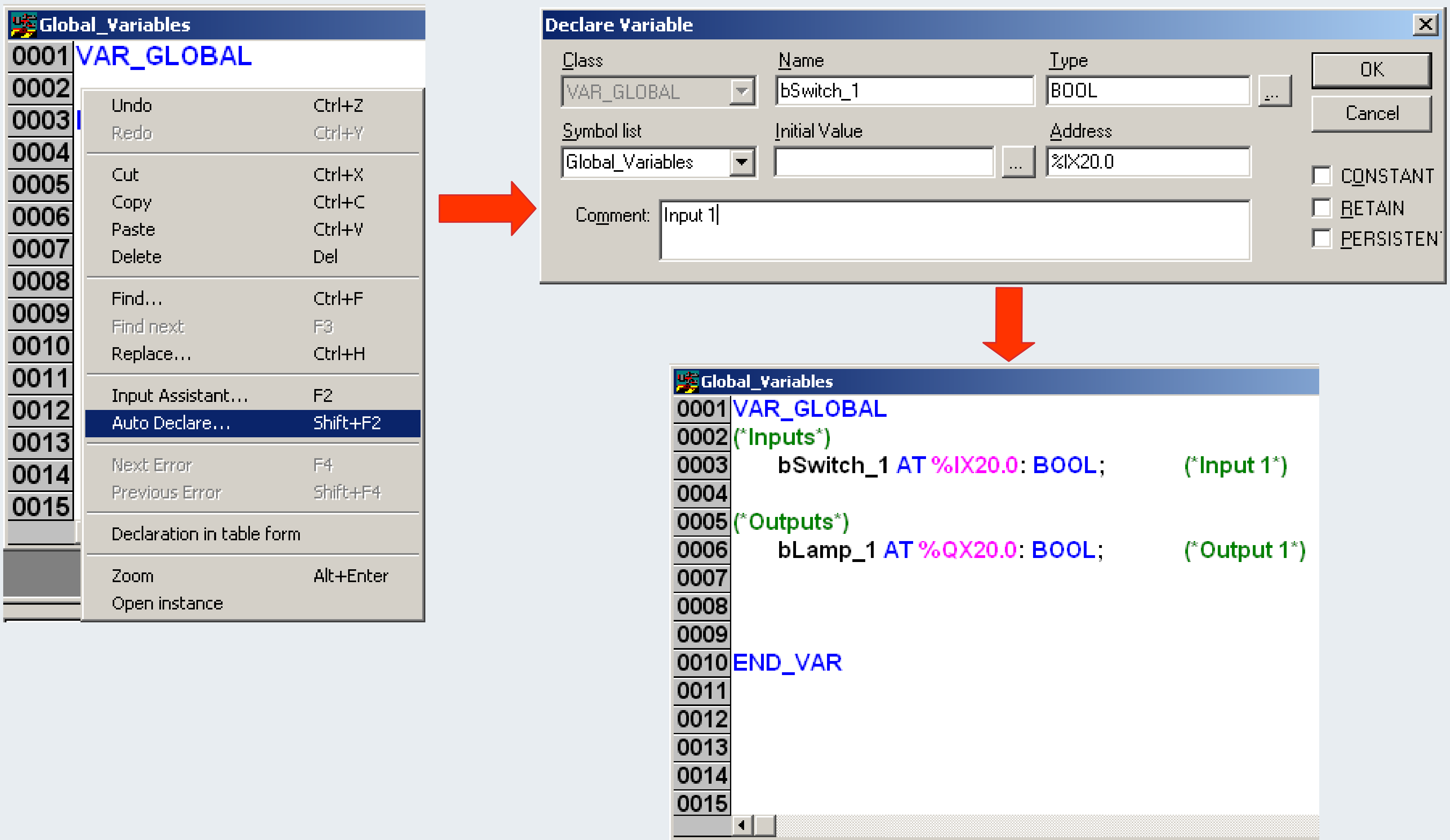
4. No umlauts



Integral Data Types

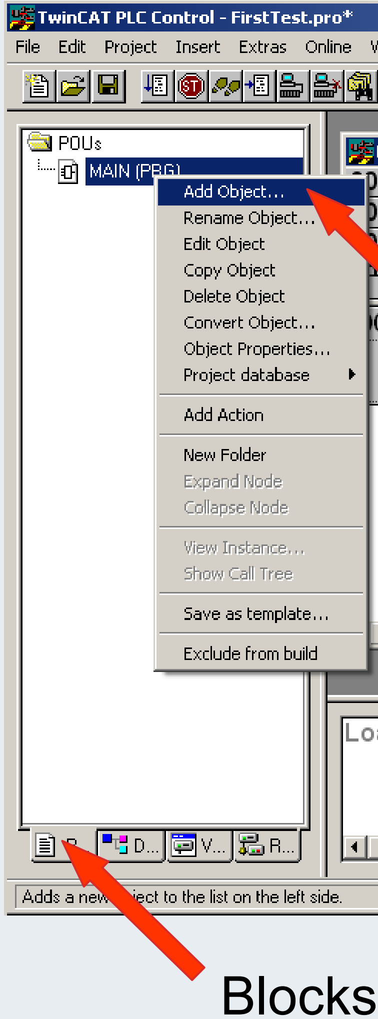
Type	Lower limit	Upper limit	Memory space
BYTE	0	255	8 bit
WORD	0	65535	16 bit
DWORD	0	4294967295	32 bit
SINT	- 128	127	8 bit
USINT	0	255	8 bit
INT	- 32768	32767	16 bit
UINT	0	65535	16 bit
DINT	- 2147483648	2147483647	32 bit
UDINT	0	4294967295	32 bit

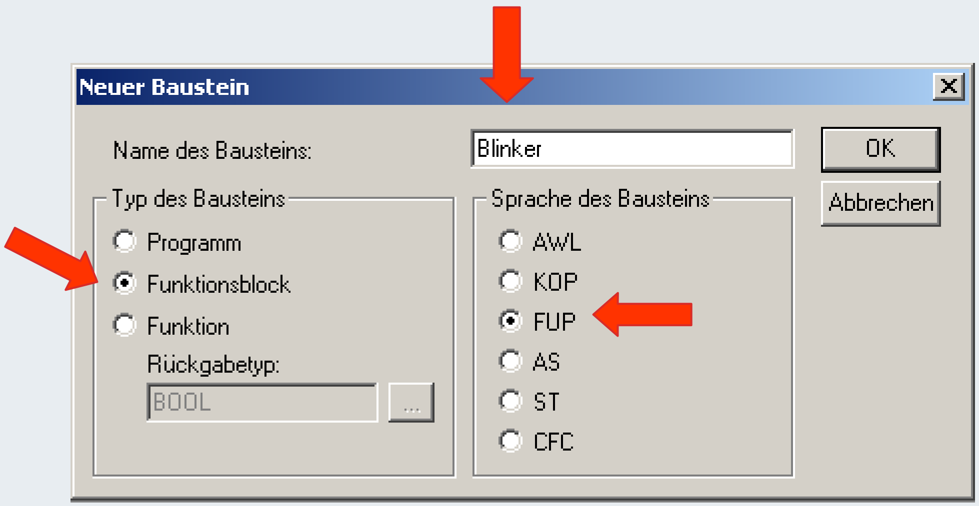
Variable declaration in PLC Control



Example: Indicator under FBD (1)

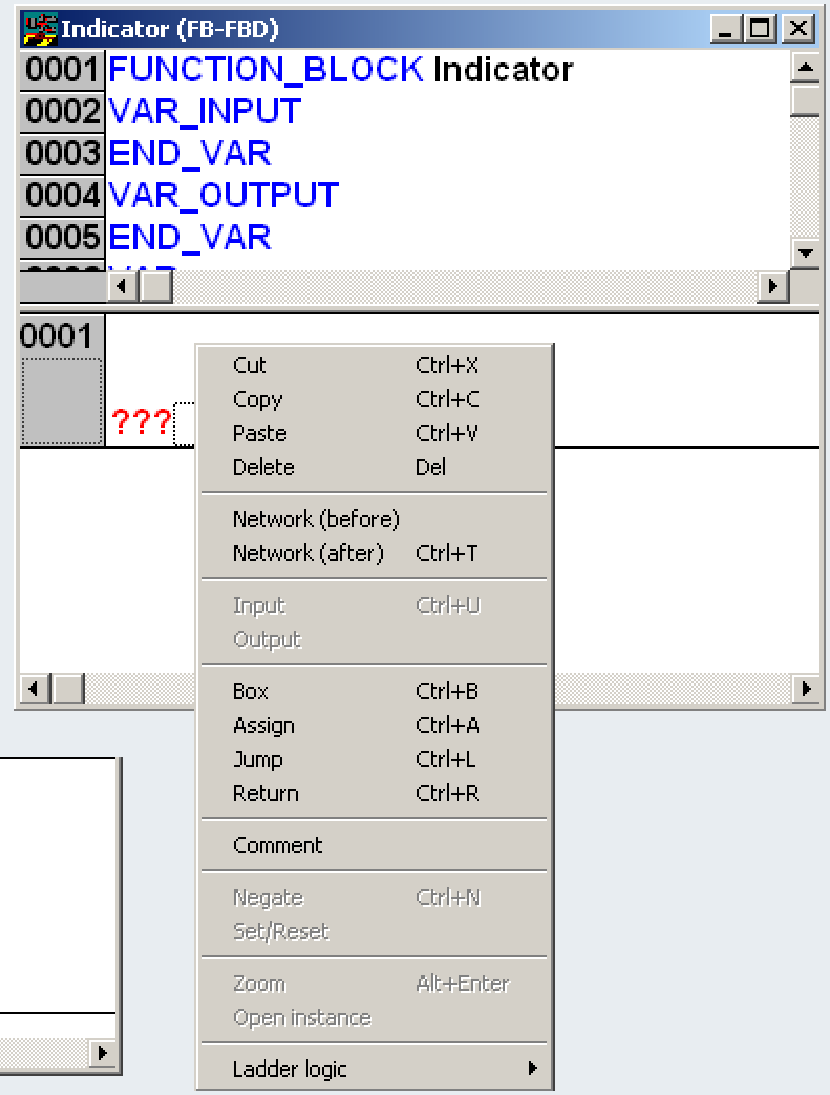
- First save the project via File/Save As.
- Click on the Blocks tab.
- Right-click the blocks folder at the top and insert an object.
- Select an FBD function block with the title ”Blinker” (Indicator) and confirm with OK.

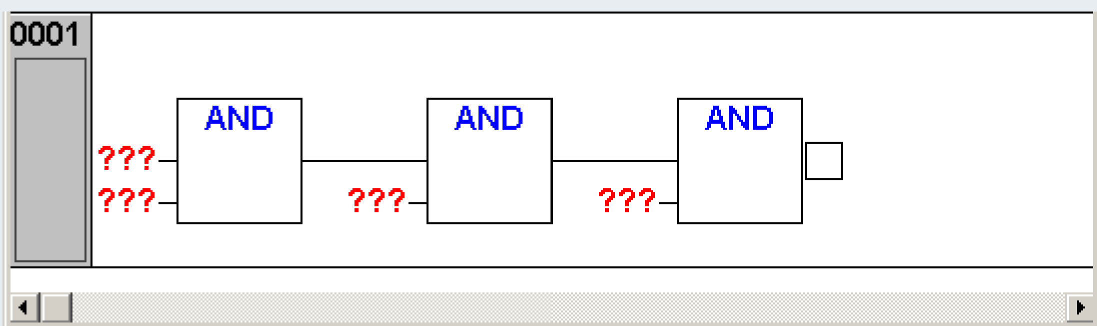




Example: Indicator under FBD (2)

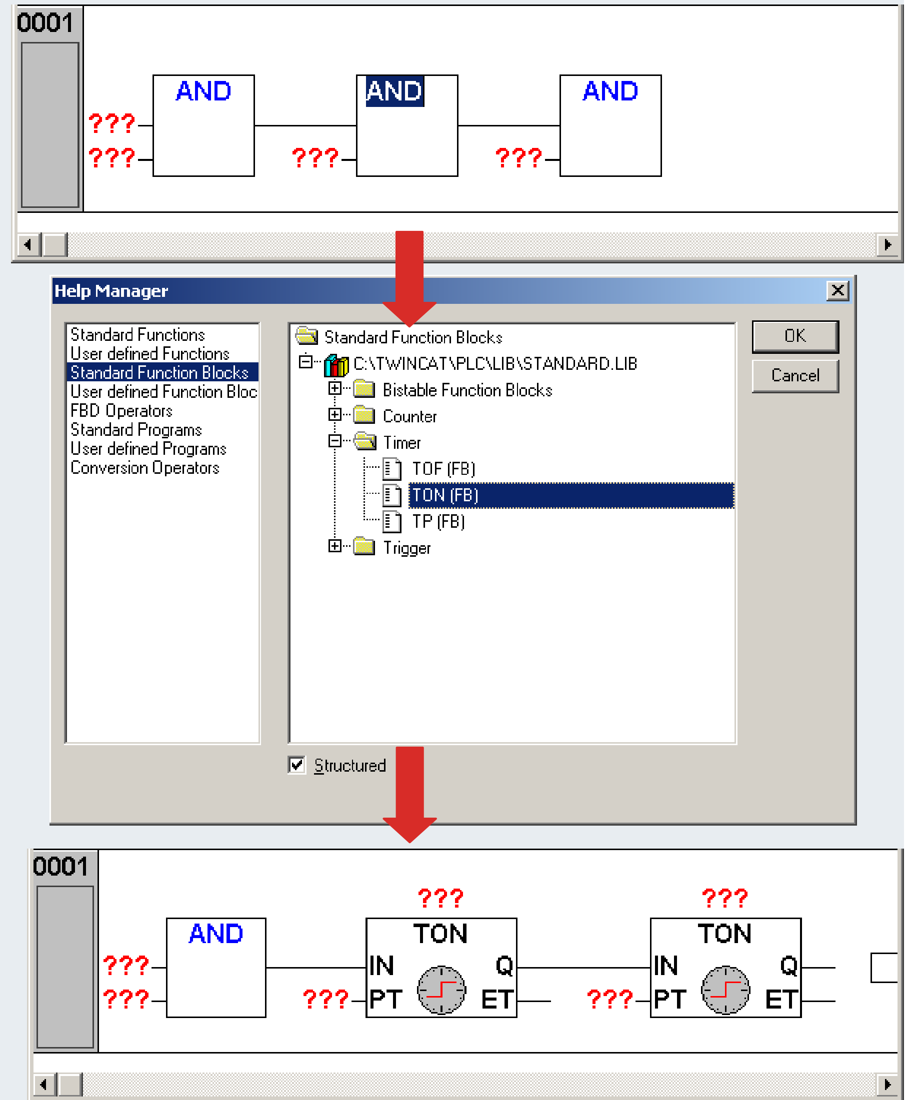
- In network 0001, right-click in the square behind the ???,
- Select Box.
- An AND Box is inserted in the editor.
- Repeat the process at the end of the AND Box.
- A total of 3 Boxes are required





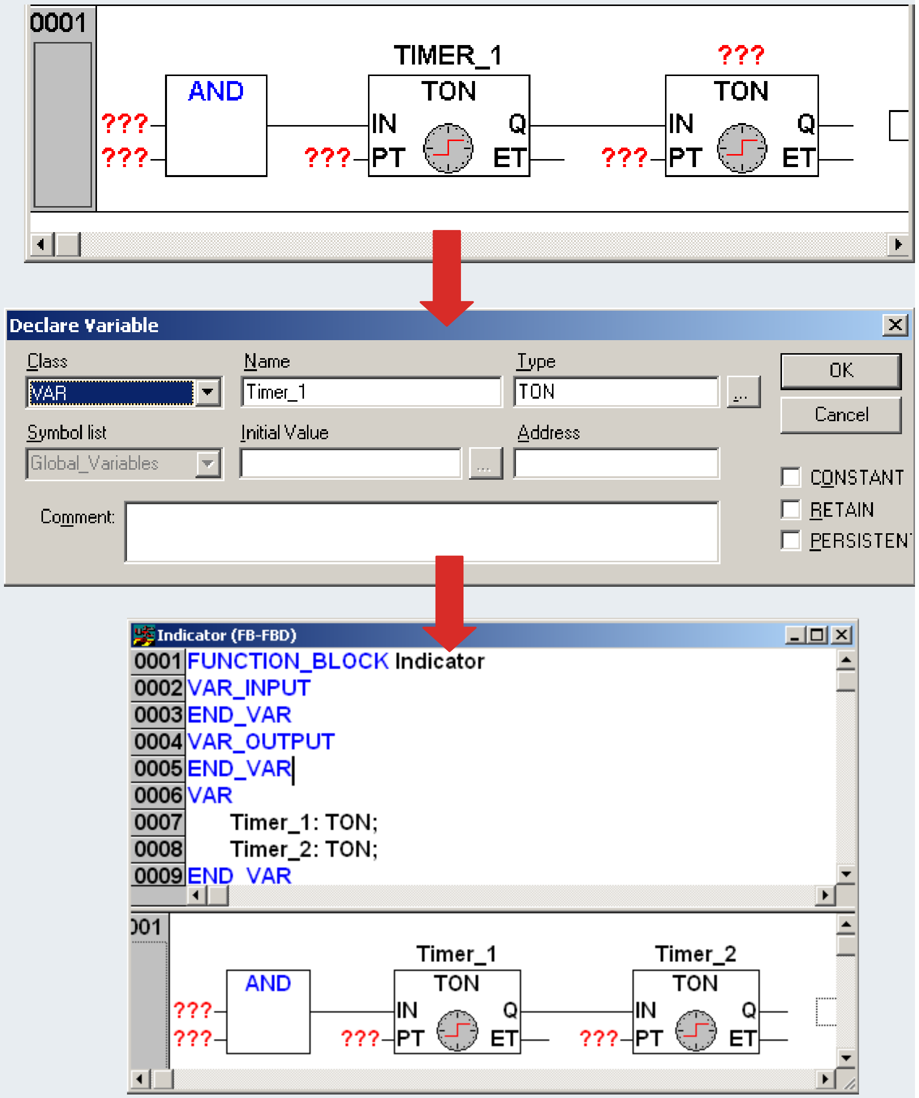
Example: Indicator under FBD (3)

- Click on the name of the centre AND Box.
- Call up a new window with Input Assistant via function key F2.
- The Box can be overwritten.
- Under standard function blocks select the Timer folder an search for TON.
- Confirm with OK and click anywhere in the network in order to update the diagram.
- Repeat the process for the last AND Box.



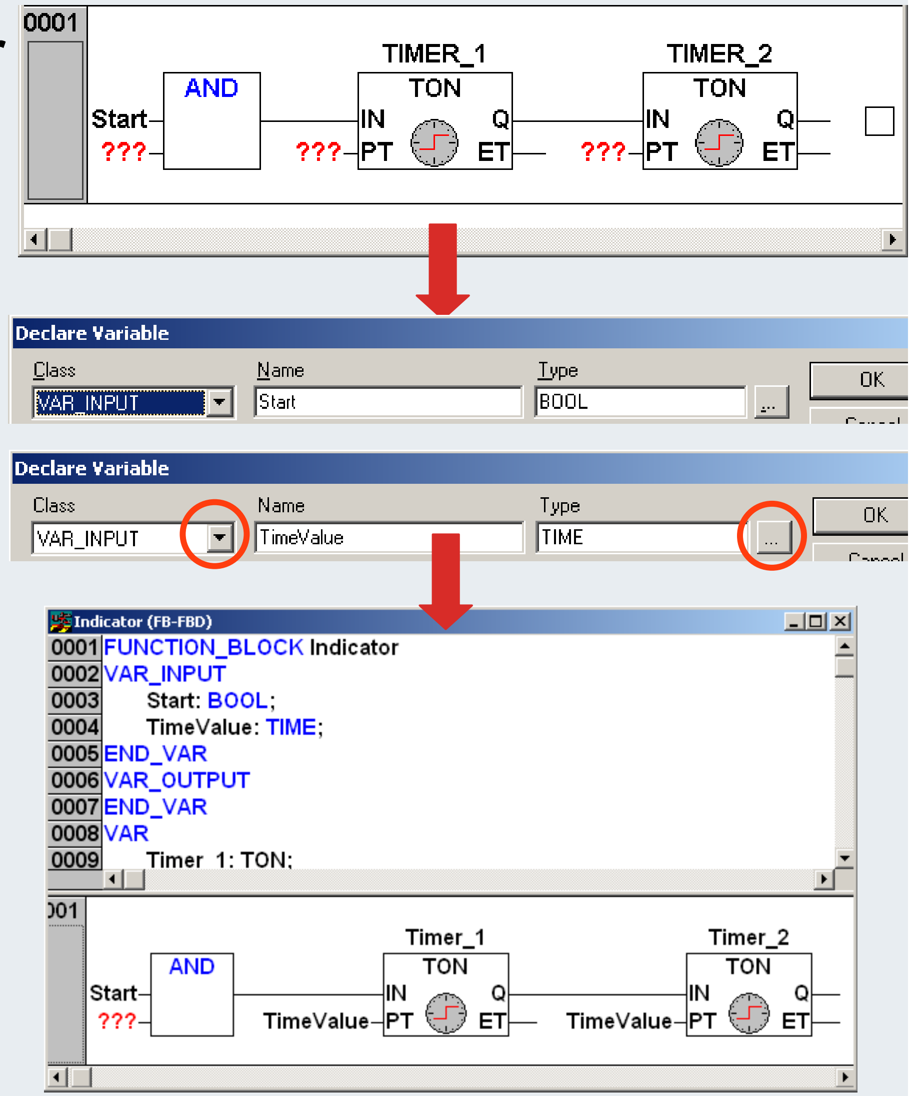
Example: Indicator under FBD (4)

- Enter a name at the ??? above the first TON Box and click anywhere in the network.
- The variable declaration window appears.
- The entries for the Box are correct. Confirm window with OK and click anywhere in the network in order to update the diagram.
- Repeat the process for the second TON (use a different name).



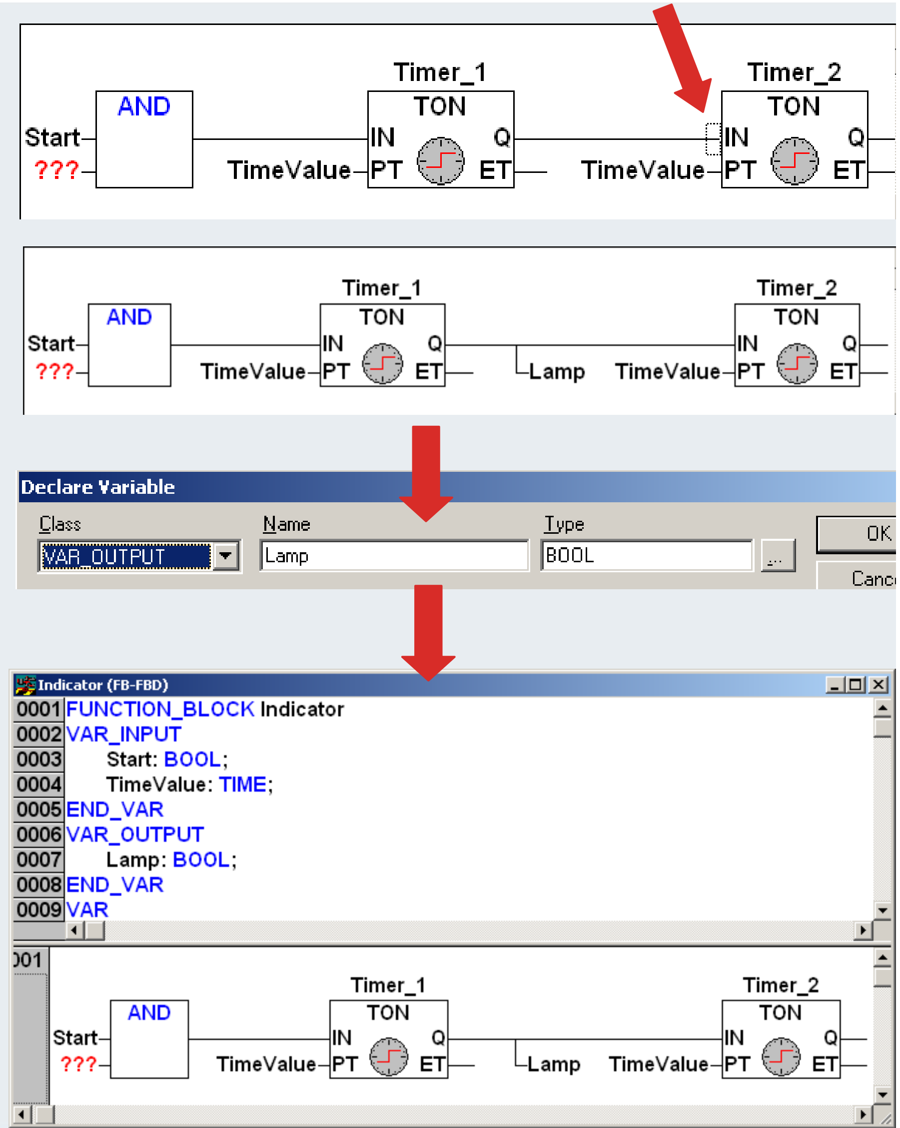
Example: Indicator under FBD (5)

- Enter the name „Start“ at the ??? for first AND, then click anywhere in the network.
- The variable declaration window appears.
- The entry under class has to be changed to VAR_INPUT for the variable. Confirm window with OK and click anywhere in the network in order to update the diagram.
- Repeat the process for the variable TimeValue and change the type to TIME.



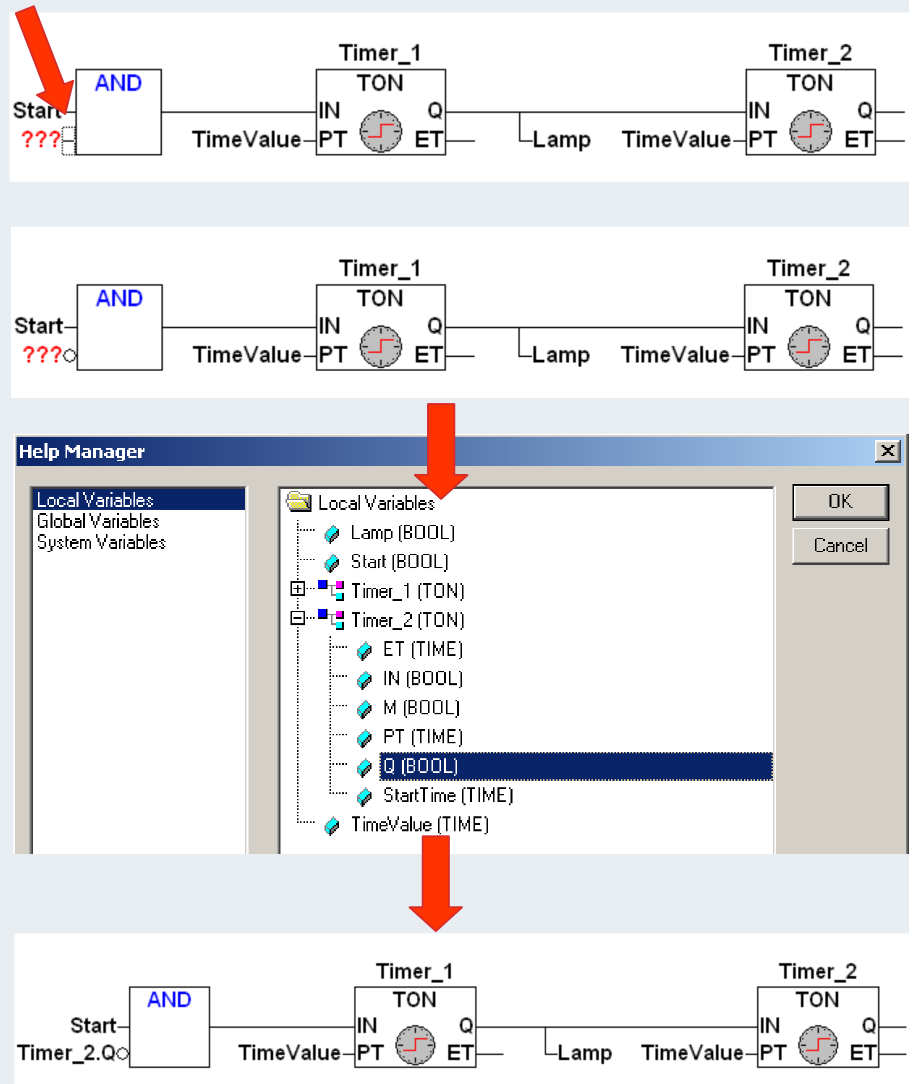
Example: Indicator under FBD (6)

- Right-click next to IN at Timer_2 and select an Assign.
- Enter lamp as the variable at the new ???, then click anywhere in the network.
- The variable declaration window appears.
- The entry under class has to be changed to VAR_OUTPUT for the variable. Confirm window with OK and click anywhere in the network in order to update the diagram.



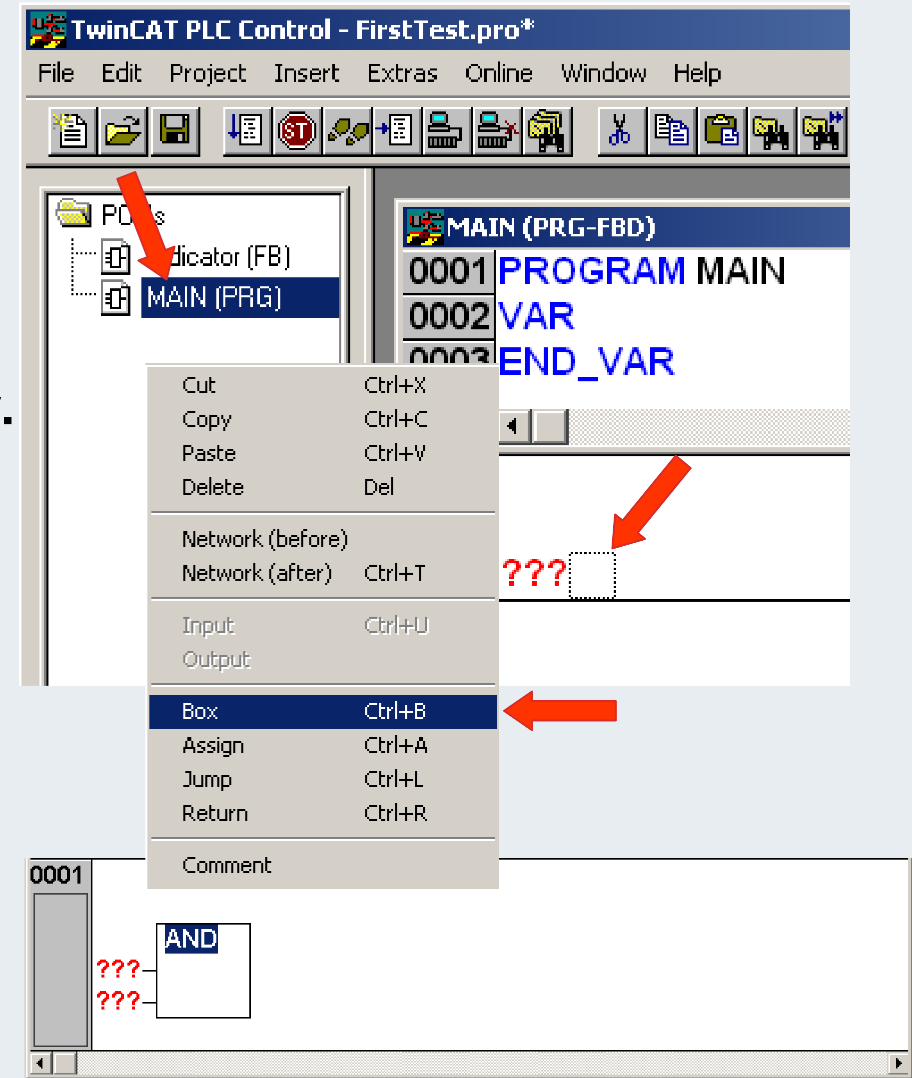
Example: Indicator under FBD (7)

- Right-click next to the ??? of AND and select a negation.
- Then click on the ??? for marking and press F2. The Input Assistant opens.
- Select local variables/Timer_2.Q. Confirm window with OK and click anywhere in the network in order to update the diagram.
- The Indicator FB has been created.



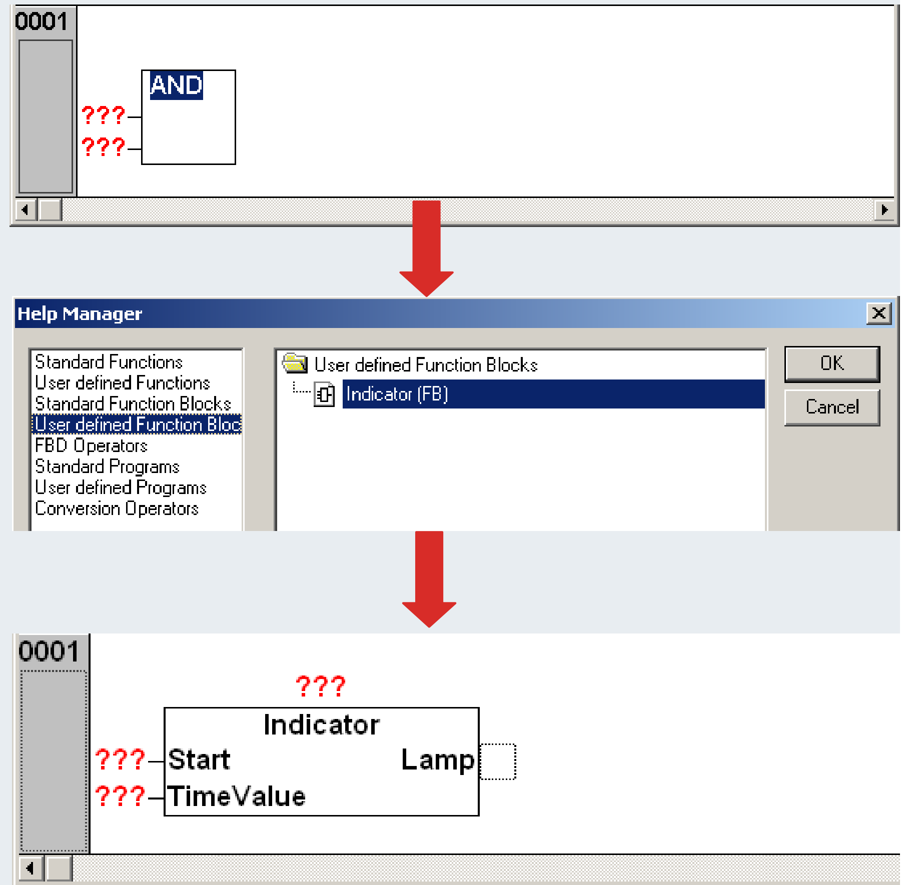
Example: Indicator under FBD (8)

- Double-click on Main to open the main program.
- In network 0001, right-click in the square behind the ???.
- Select Box.
- An AND Box is inserted in the editor.



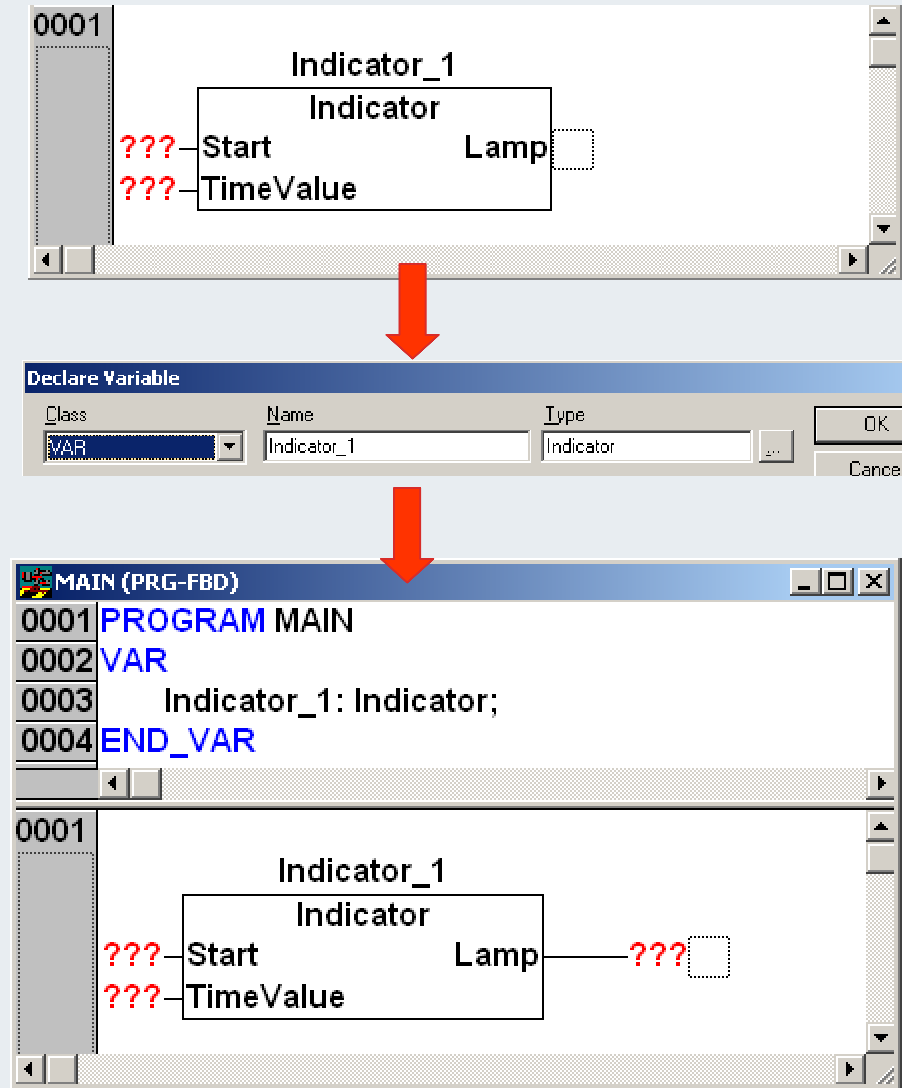
Example: Indicator under FBD (8)

- Click on the name of the AND Box.
- Call up a new window with Input Assistant via function key F2.
- The block can be overwritten.
- From the defined function blocks select the Indicator block.
- Confirm with OK and click anywhere in the network in order to update the diagram.



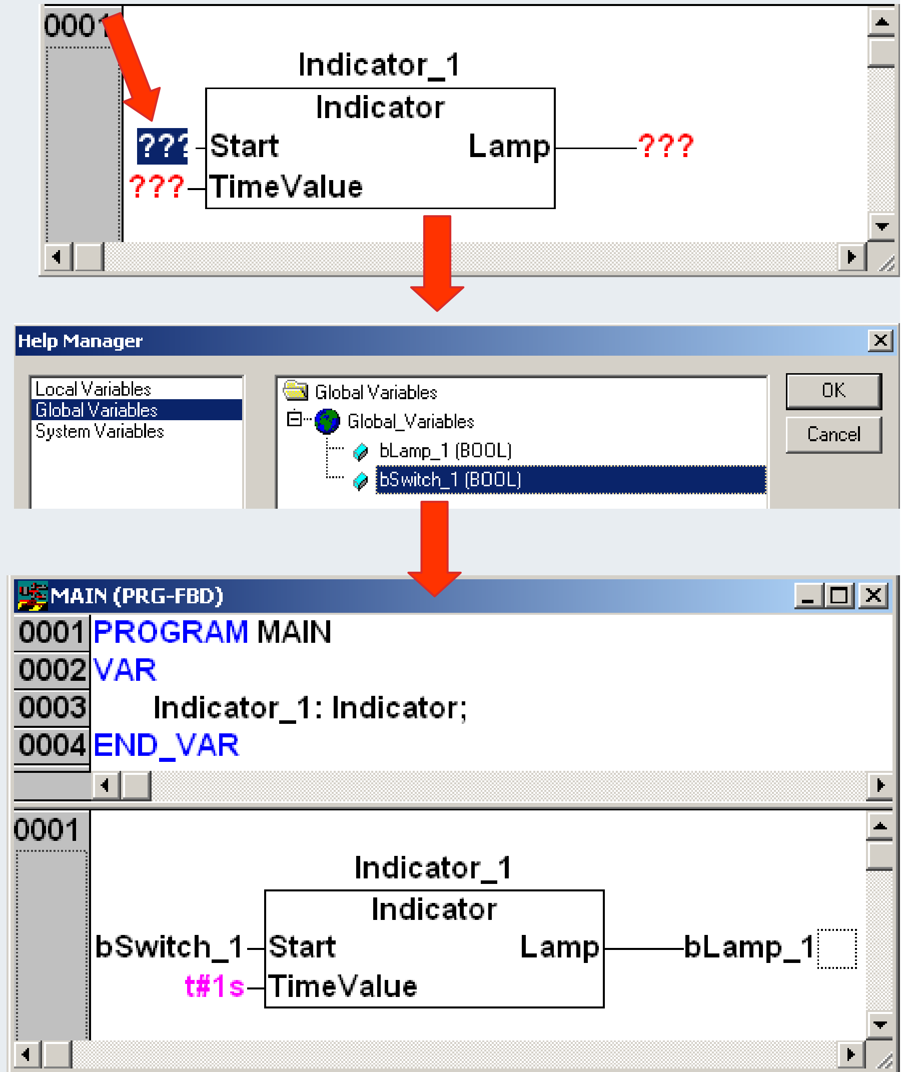
Example: Indicator under FBD (9)

- Enter a name at the ??? above the indicator Box and click anywhere in the network.
- The variable declaration window appears.
- The entries for the Box are correct. Confirm window with OK and click anywhere in the network in order to update the diagram.
- Right-click at the end of the block and select an Assign.



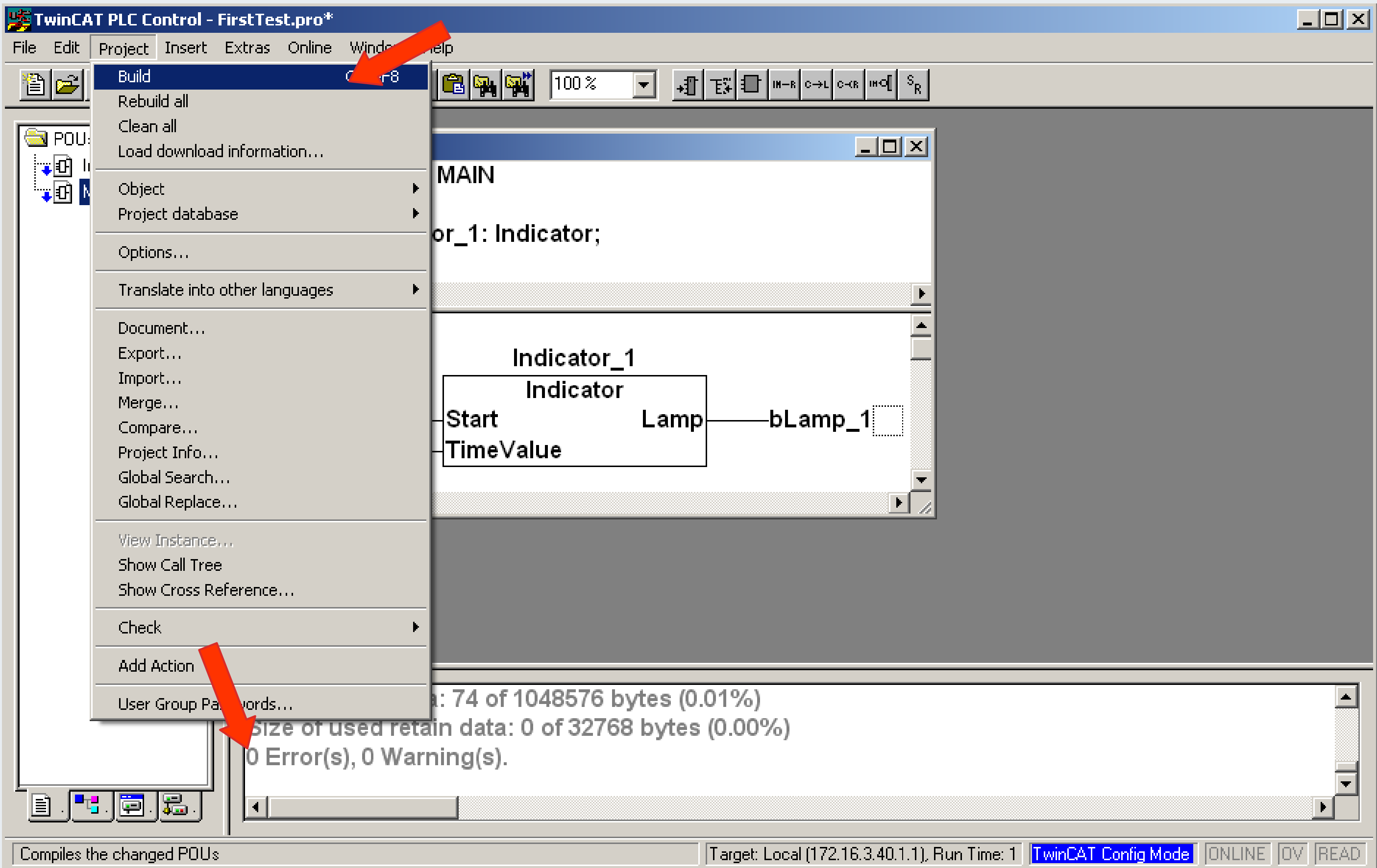
Example: Indicator under FBD (10)

- Then mark the ??? at the start input by clicking and press F2. The Input Assistant appears.
- Select variable bSwitch_1 under global variables. Confirm window with OK and click anywhere in the network in order to update the diagram.
- Repeat the process at the ??? at lamp output and select variable bLamp_1.
- Finally, enter a time at the ??? at the time value input. T#1s (s=sec.)



Example: Indicator under FBD (11)

- Finally select Project/Compile.
- Check for error messages in the message window.



IBK - T3

What target platform can be used for programming the PC?

The priorities for the PLC task are specified under task features. What is the maximum number of tasks that can be created?

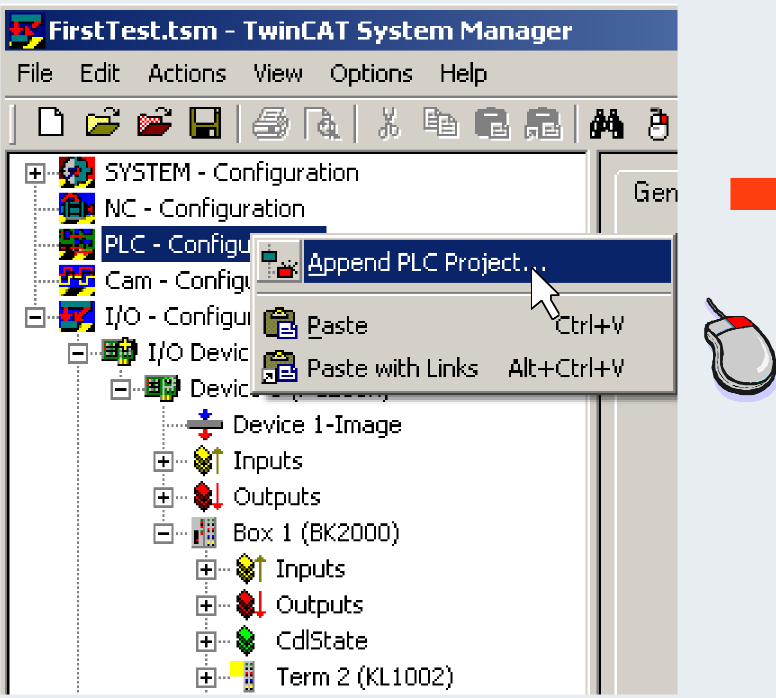
What data type has to be entered for a variable in bit format?

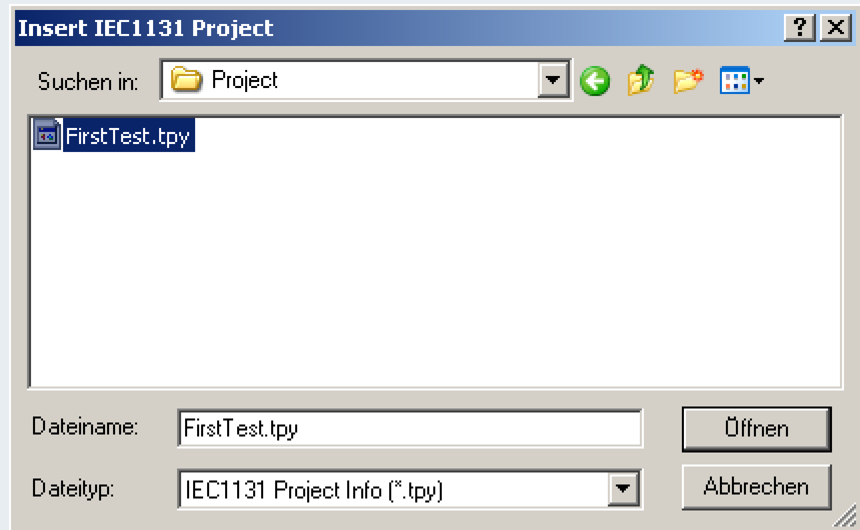
What is wrong with the following variable declaration?

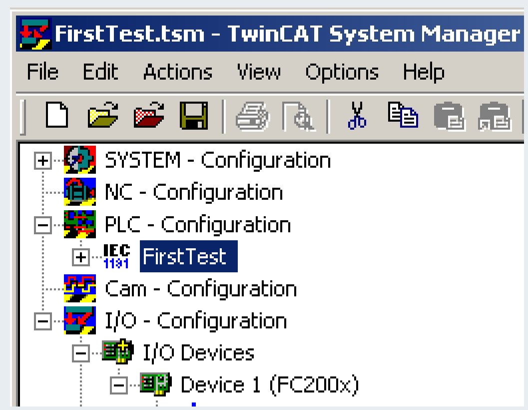
```
Drive_contactor_On      AT%IX0.0    :    BOOL;
```

```
    Potentiometer1      AT%IW2      :    INT;
    Potentiometer2      AT%IW3      :    INT;
```

Integrating new PLC variables in the System Manager

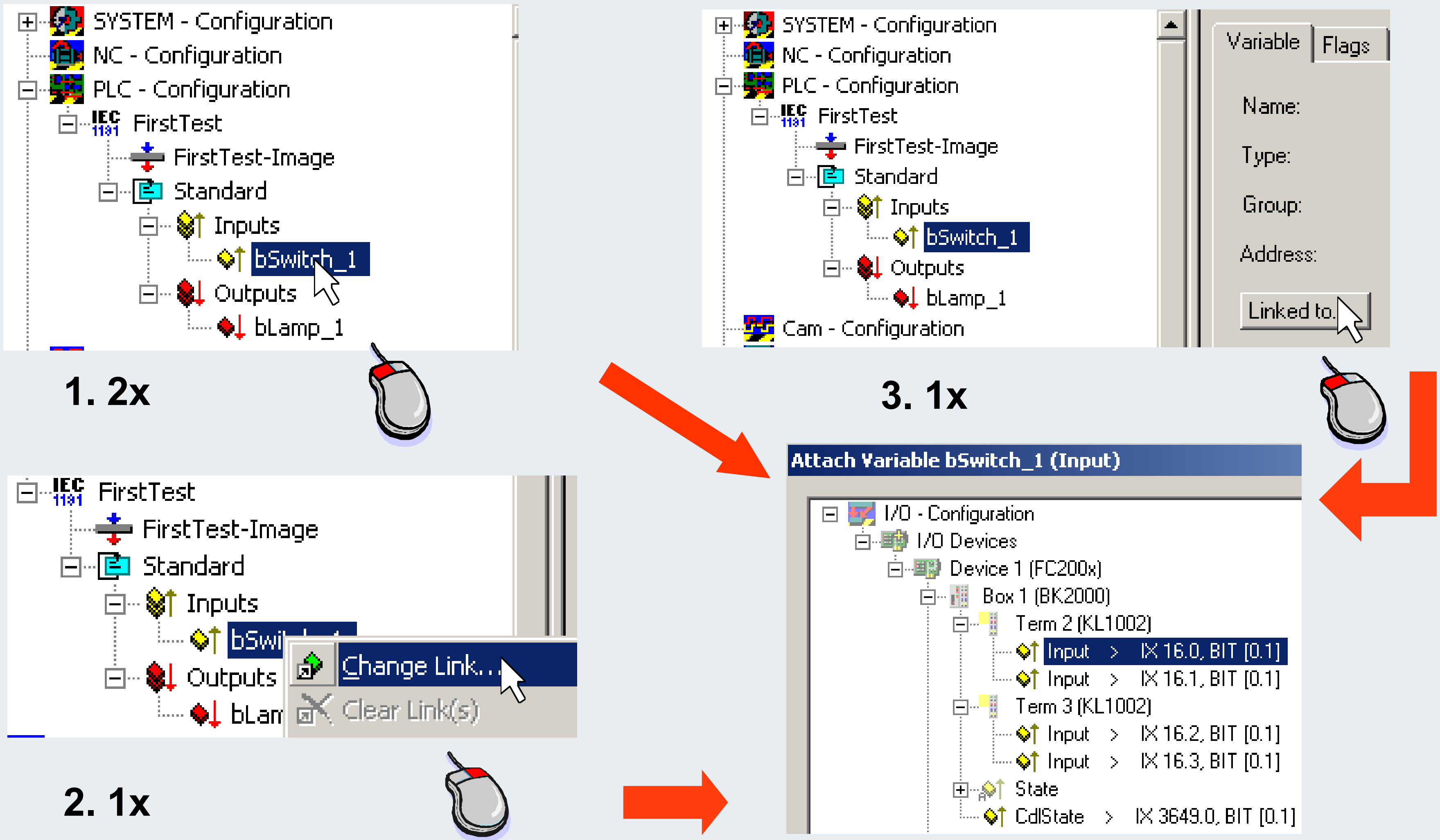






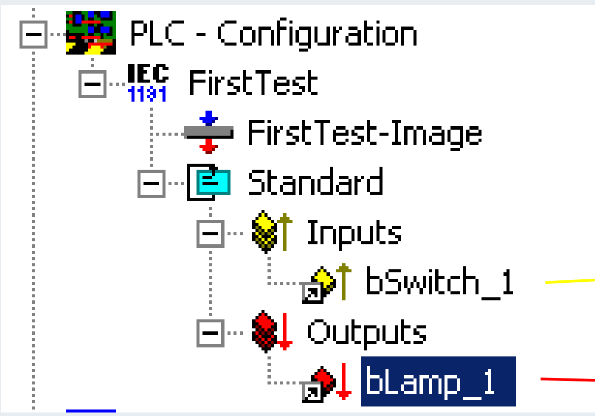
The file with project name and the extension *.tpy generated by PLC Control is entered in the System Manager

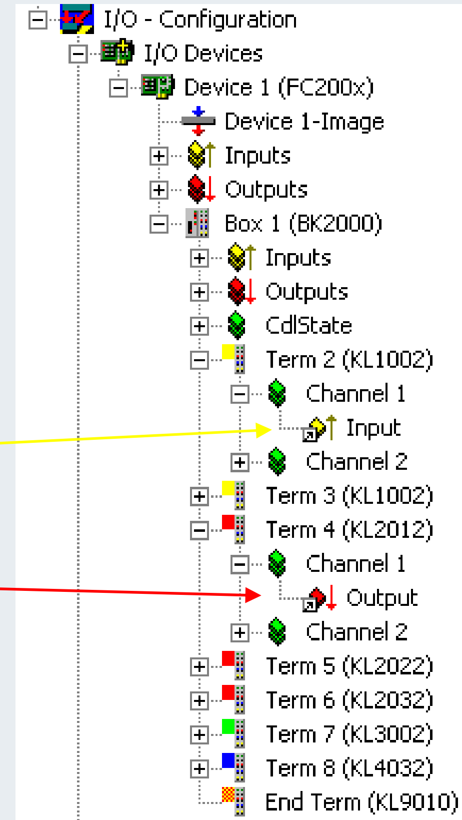
Linkage


SYSTEM - Configuration
NC - Configuration
PLC - Configuration
FirstTest
FirstTest-Image
Standard
Inputs
bSwitch_1
Outputs
bLamp_1
1. 2x
FirstTest
FirstTest-Image
Standard
Inputs
Outputs
Change Link...
Clear Link(s)
2. 1x
SYSTEM - Configuration
NC - Configuration
PLC - Configuration
FirstTest
FirstTest-Image
Standard
Inputs
bSwitch_1
Outputs
bLamp_1
Cam - Configuration
Variable
Flags
Name:
Type:
Group:
Address:
Linked to...
3. 1x
Attach Variable bSwitch_1 (Input)
I/O - Configuration
I/O Devices
Device 1 (FC200x)
Box 1 (BK2000)
Term 2 (KL1002)
Input > IX 16.0, BIT [0.1]
Input > IX 16.1, BIT [0.1]
Term 3 (KL1002)
Input > IX 16.2, BIT [0.1]
Input > IX 16.3, BIT [0.1]
State
CdlState > IX 3649.0, BIT [0.1]

Allocation the PLC variables to hardware

System Manager



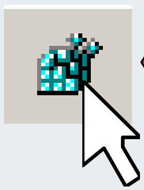


Process Images

- Software side
- Hardware side

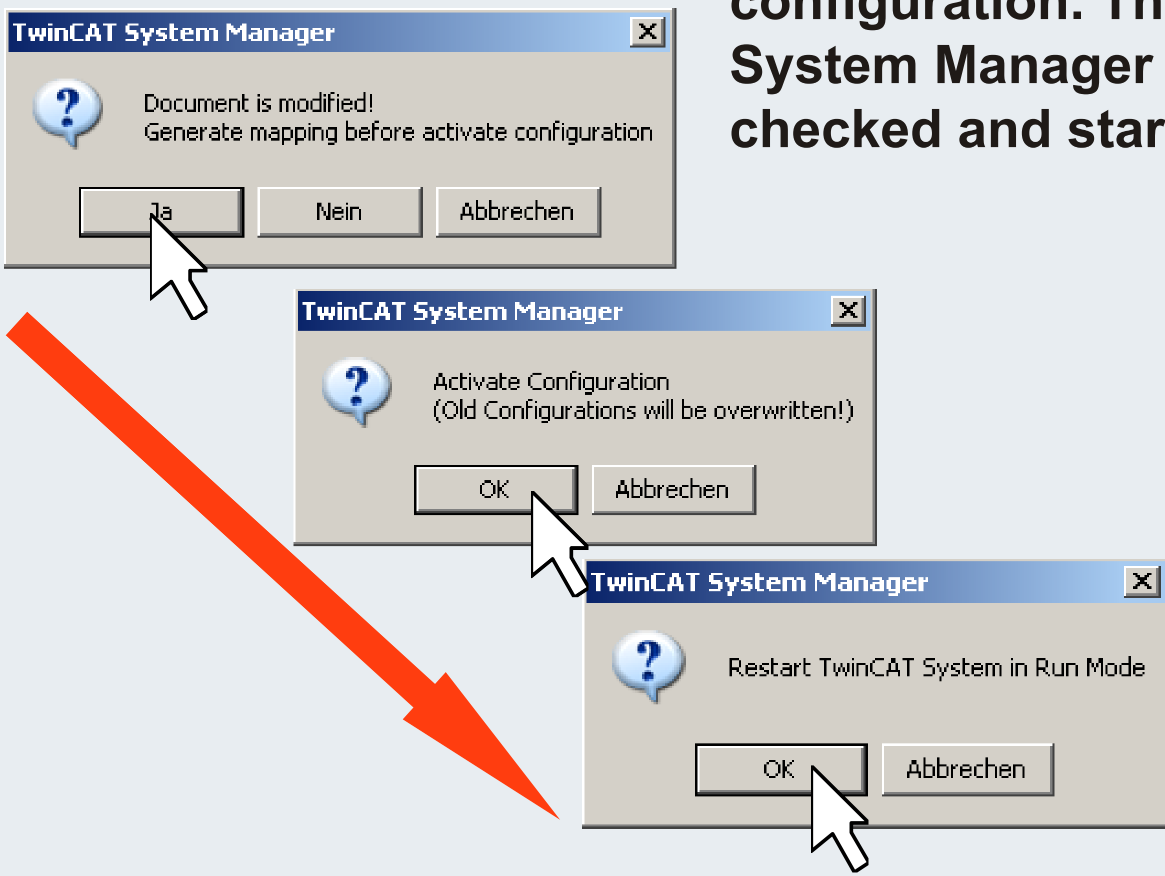
bSwitch_1	X	BOOL	0.1	20.0	Input	0	Input . Channel 1 . Term 2 (KL1002) . Box 1 (BK2000) . Device 1 (FC200x) . I/O Devices
bLamp_1	X	BOOL	0.1	20.0	Output	0	Output . Channel 1 . Term 4 (KL2012) . Box 1 (BK2000) . Device 1 (FC200x) . I/O Devices
Input	X	BOOL	0.1	16.0	Input	0	bSwitch_1 . Inputs . Standard . FirstTest
Output	X	BOOL	0.1	16.0	Output	0	bLamp_1 . Outputs . Standard . FirstTest

Saving the configuration

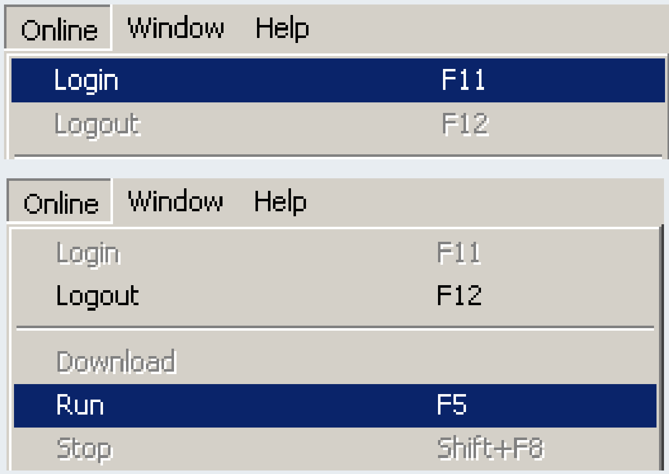


Activate configuration

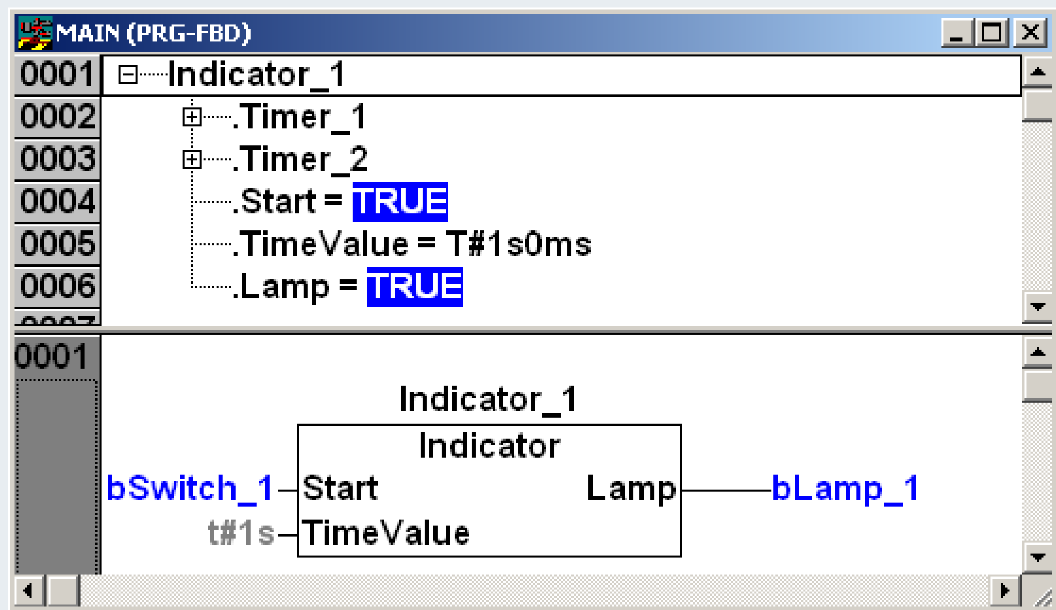
Saves and activates the current configuration. The current System Manager setting is checked and started.



Start PLC Control



In PLC Control under menu item Online, PLC login, load and start program. The message “Running” appears at the bottom right, and online representation is activated.



5 reasons for changing the configuration entry

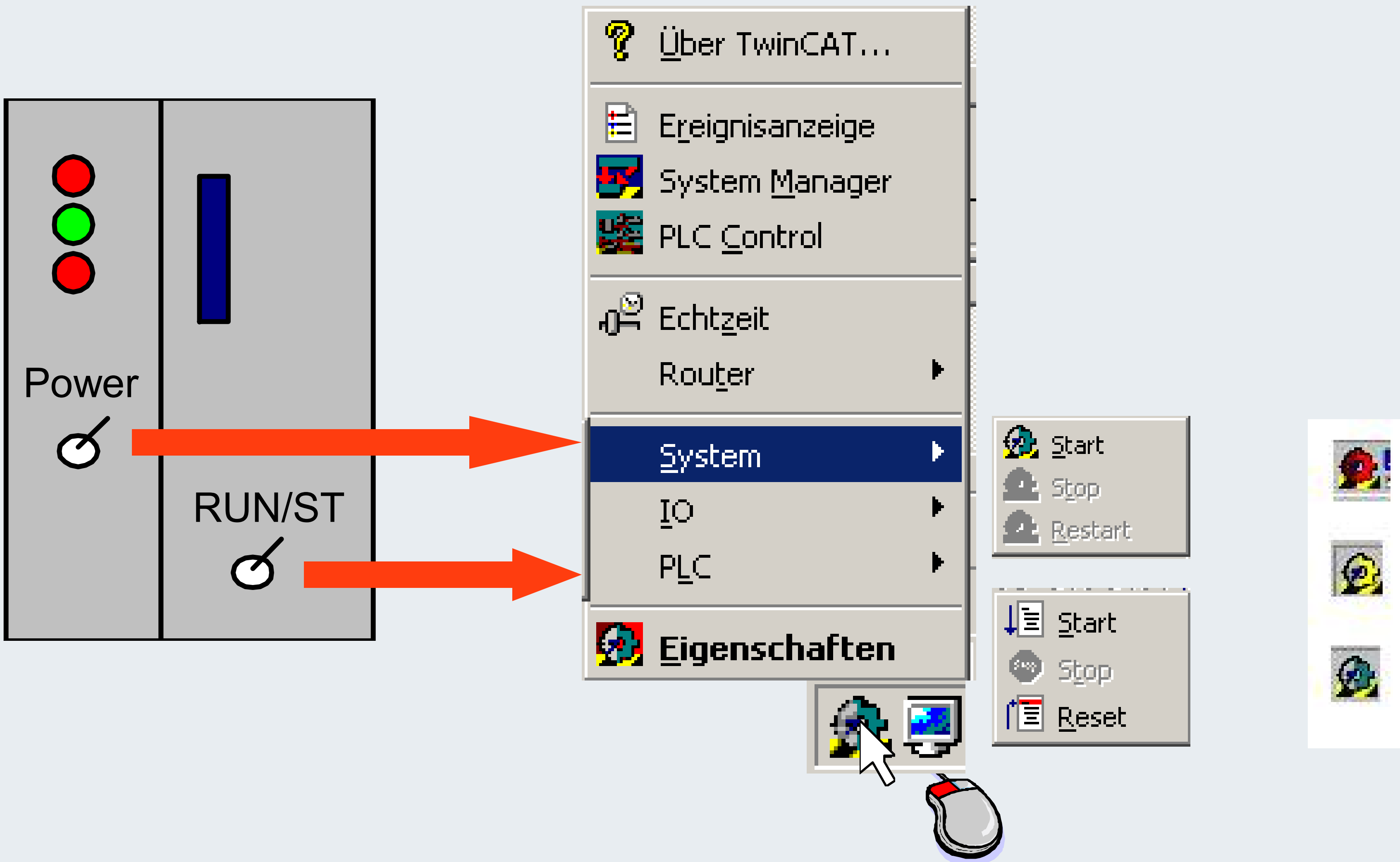
required

1. The list of allocated variables was modified in the PLC (e.g. name or address)
2. An allocated variable was added or removed
3. The hardware was modified (e.g. new module)
4. The allocation should be changed.
5. Changes in the task (cycle, priority or titles)

not required

1. The list of non-allocated variables (flags) was modified in the PLC
2. The program code was modified in the PLC(e.g. AND to OR)

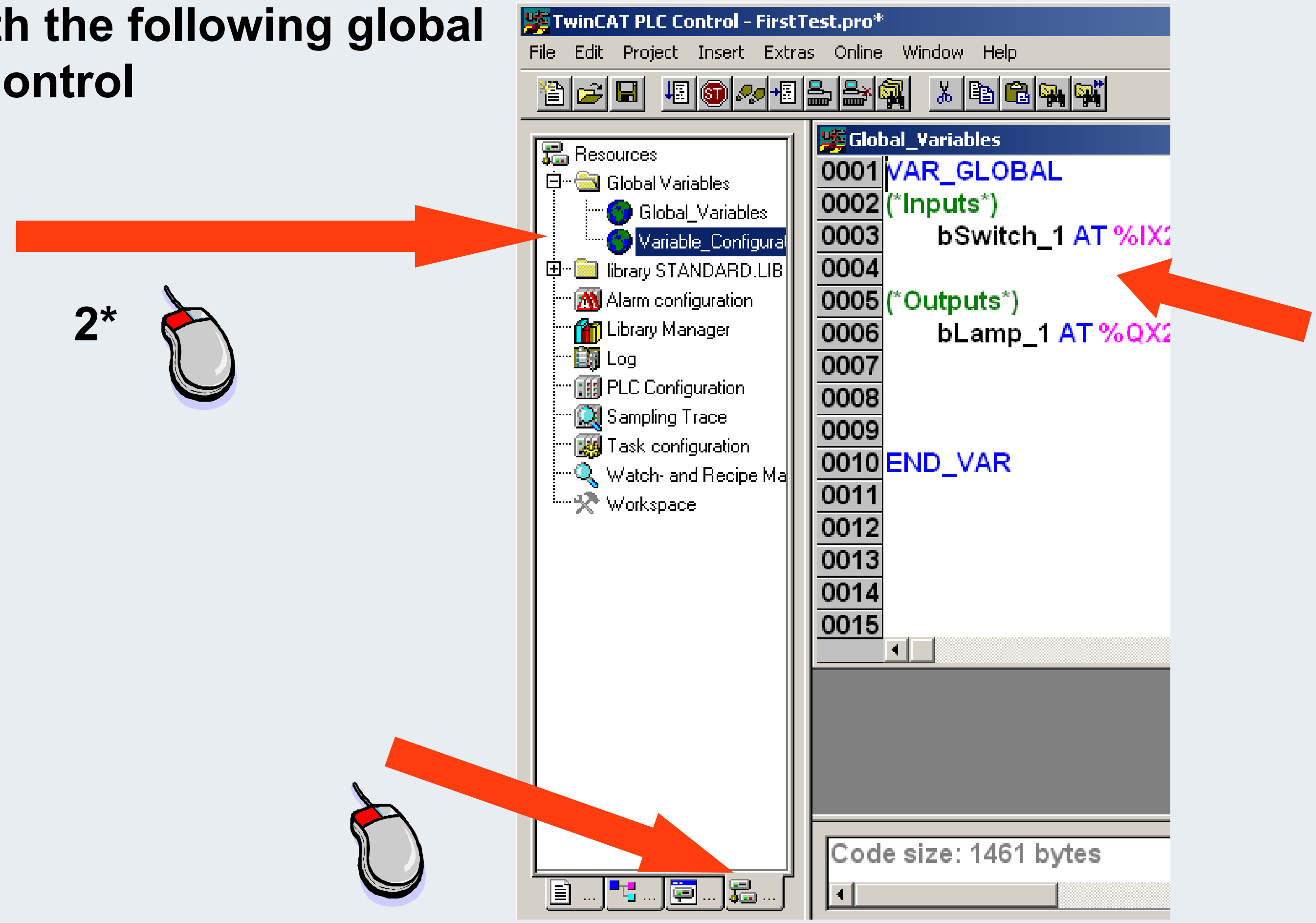
Windows environment



Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

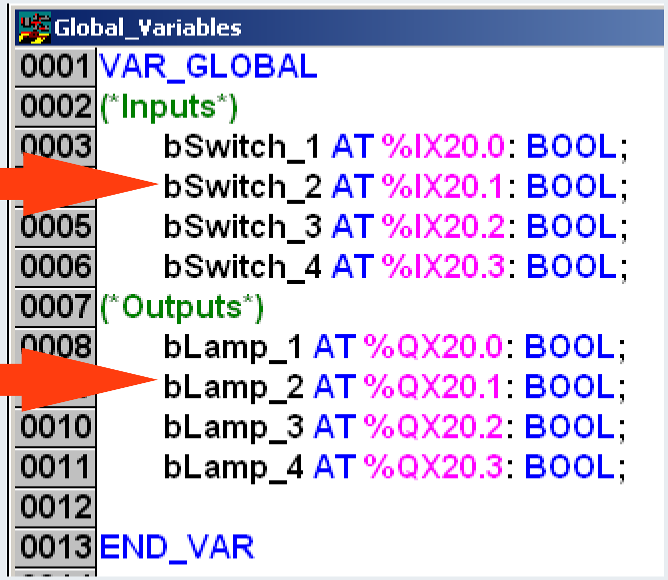
Initial state: previous example program Erster_test.pro with the following global variables in PLC Control

2*

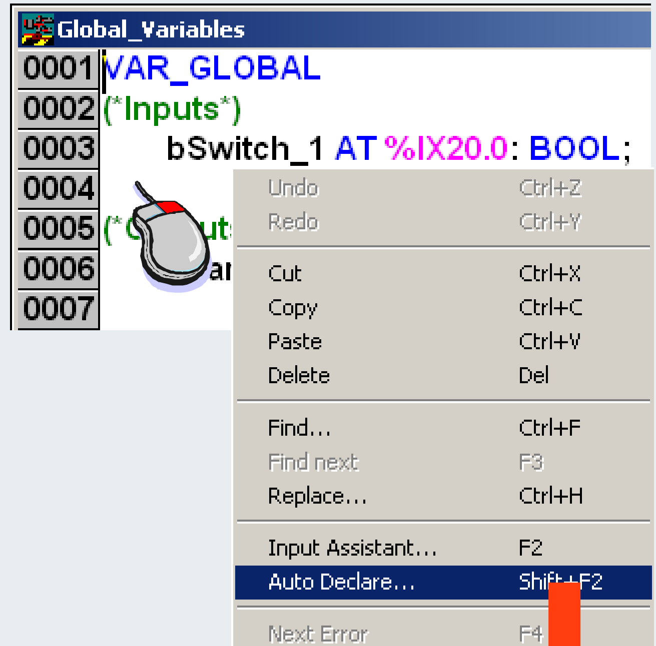


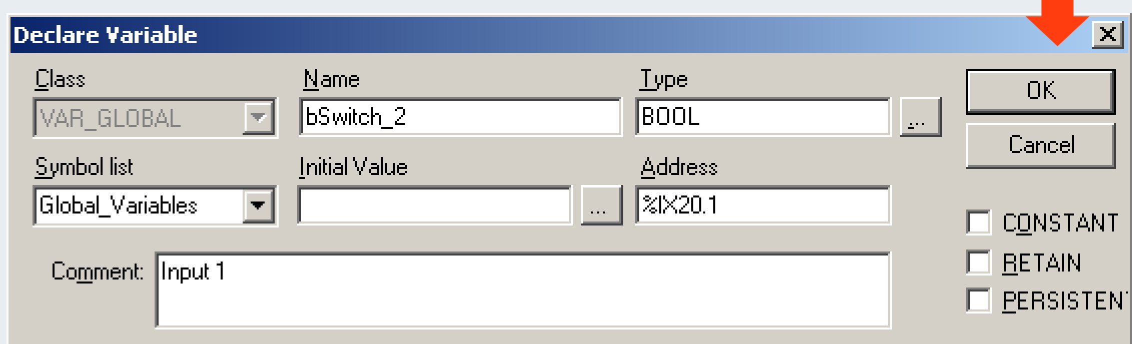
Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

Enter new variable. Use direct input or ”Input Assistant” function



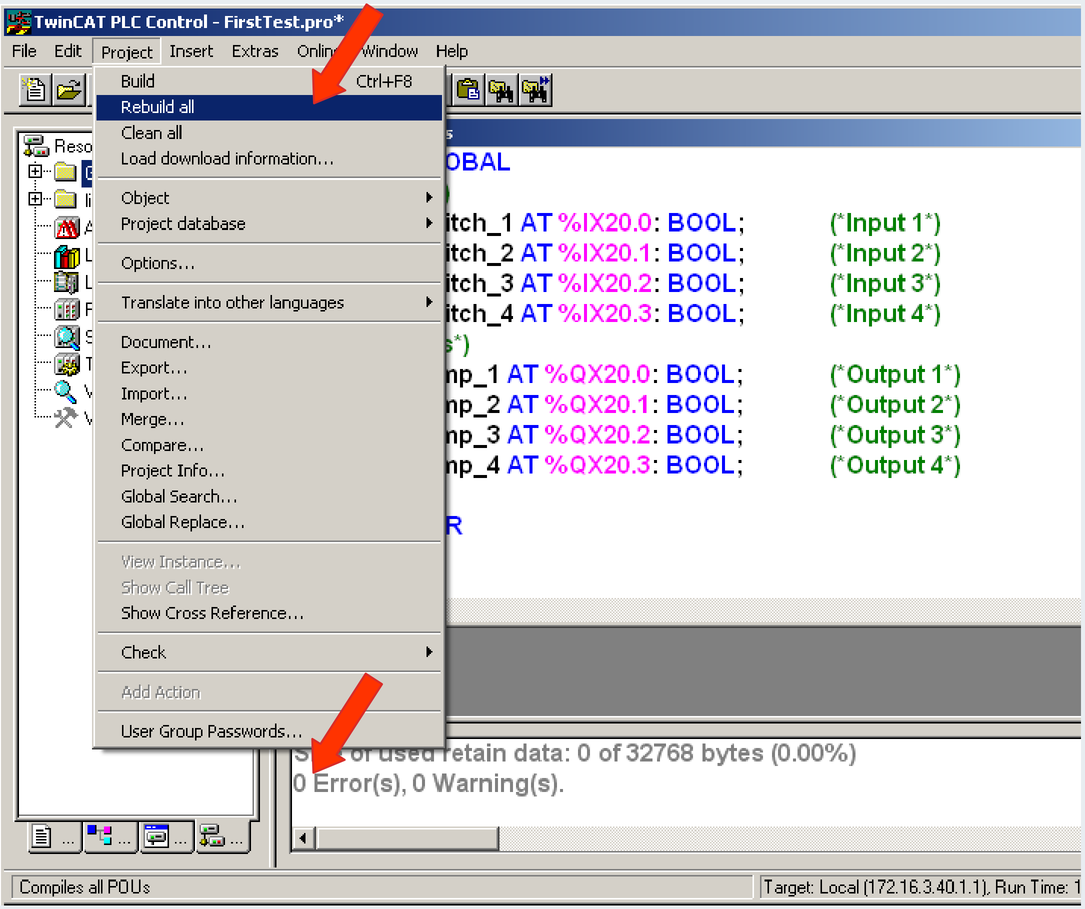
With Input Assistant:





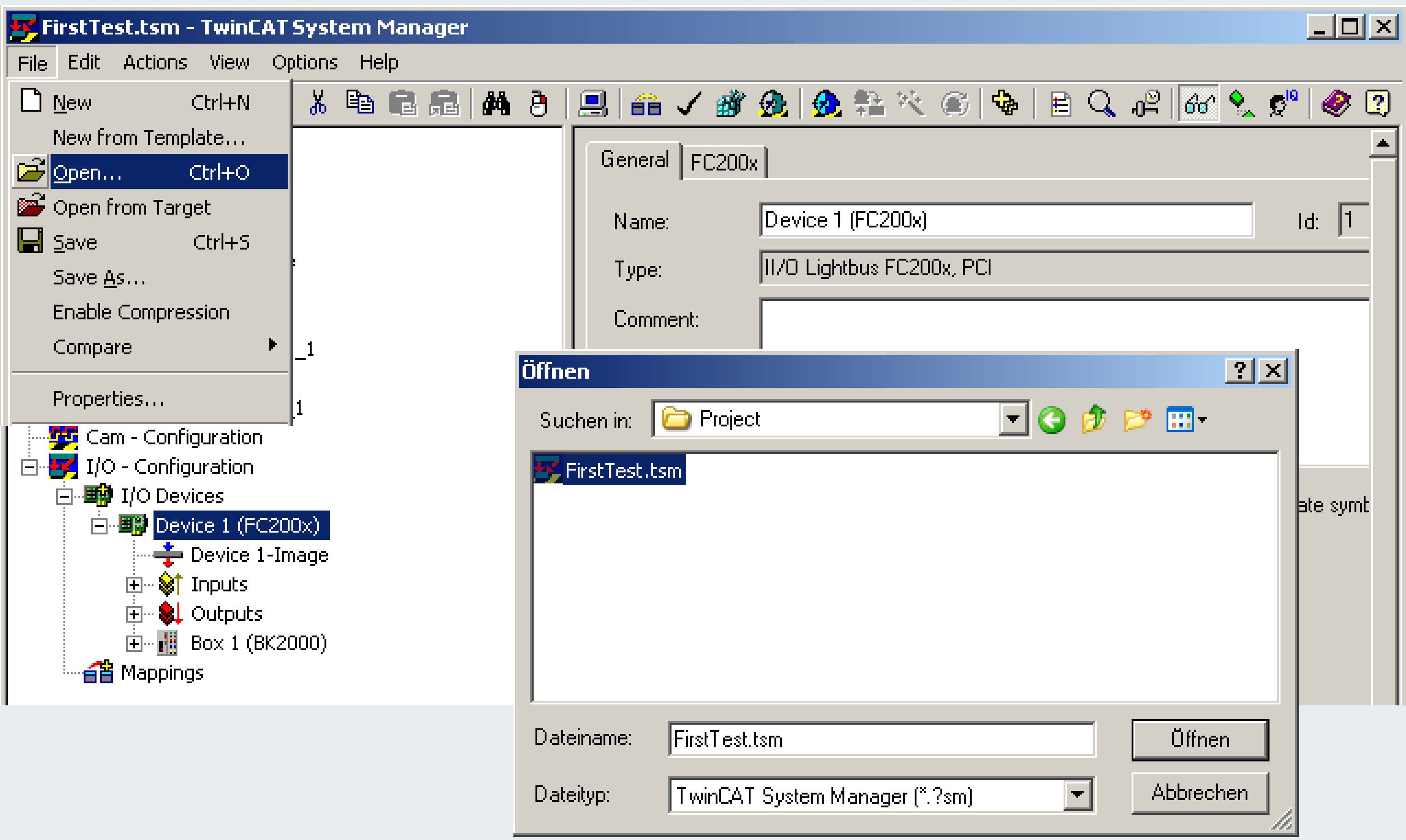
Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

- Finish with Project/Rebuild all.
- Check for error messages in the message window.



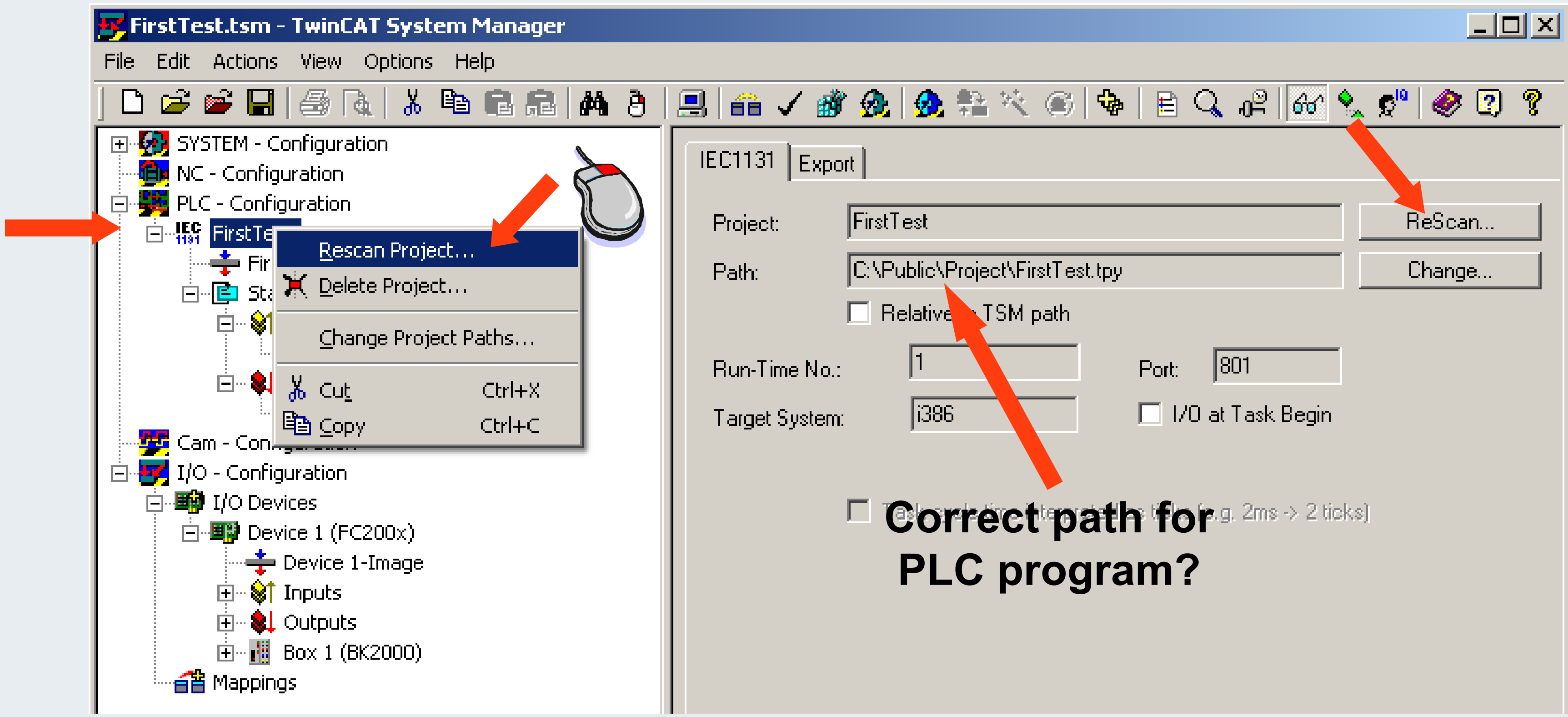
Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

- Open System Manager with FirstTest.tsm



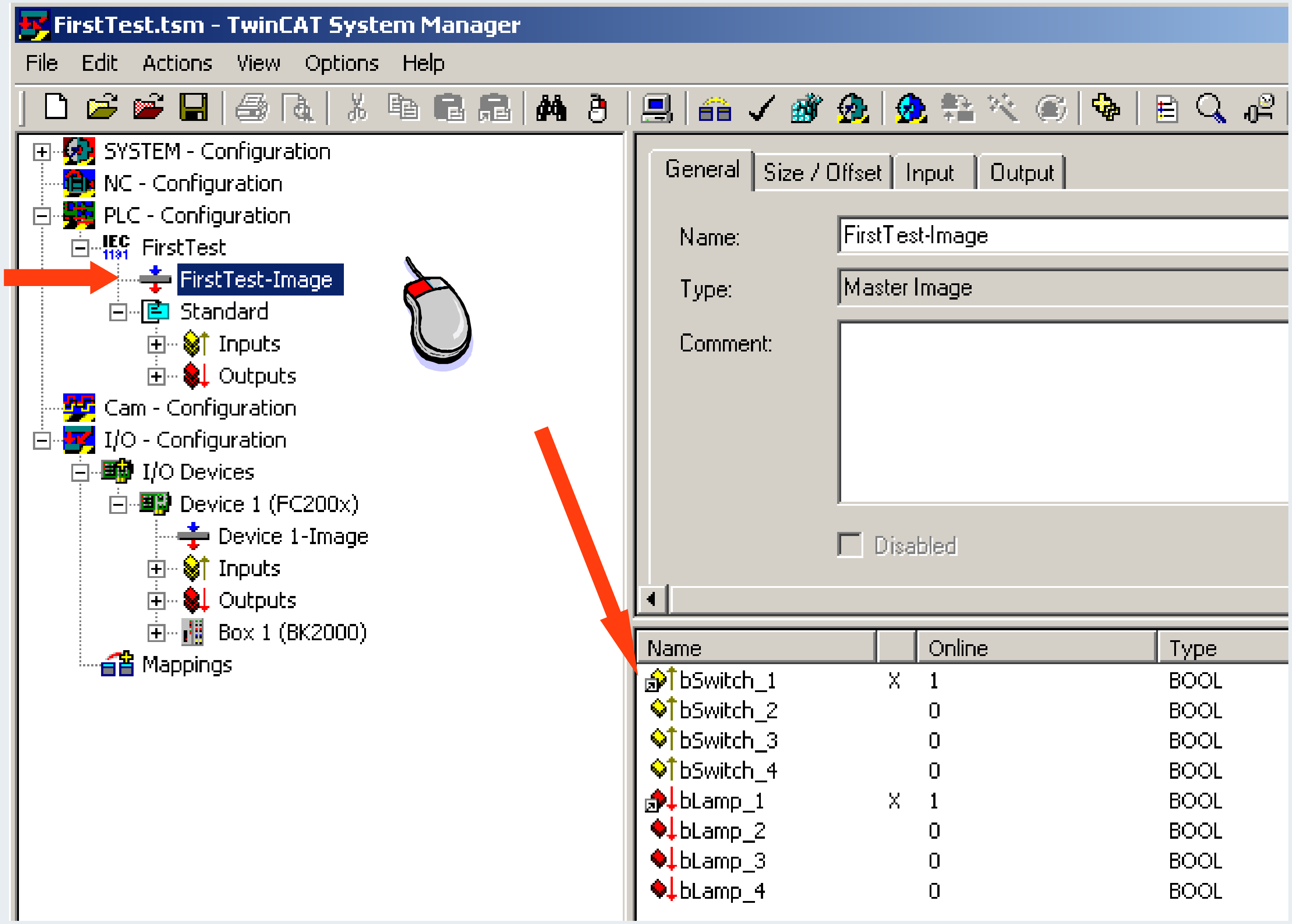
Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

- Read new PLC project



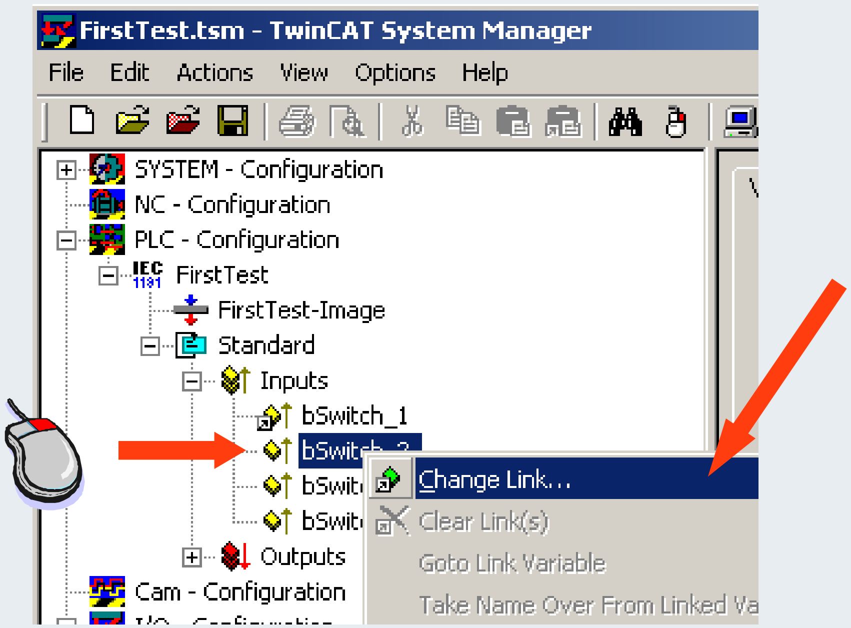
Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

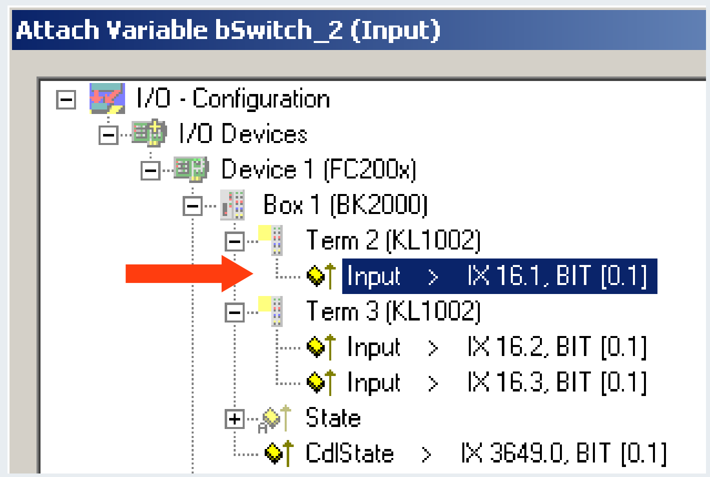
- Check whether new inputs and outputs are known



Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

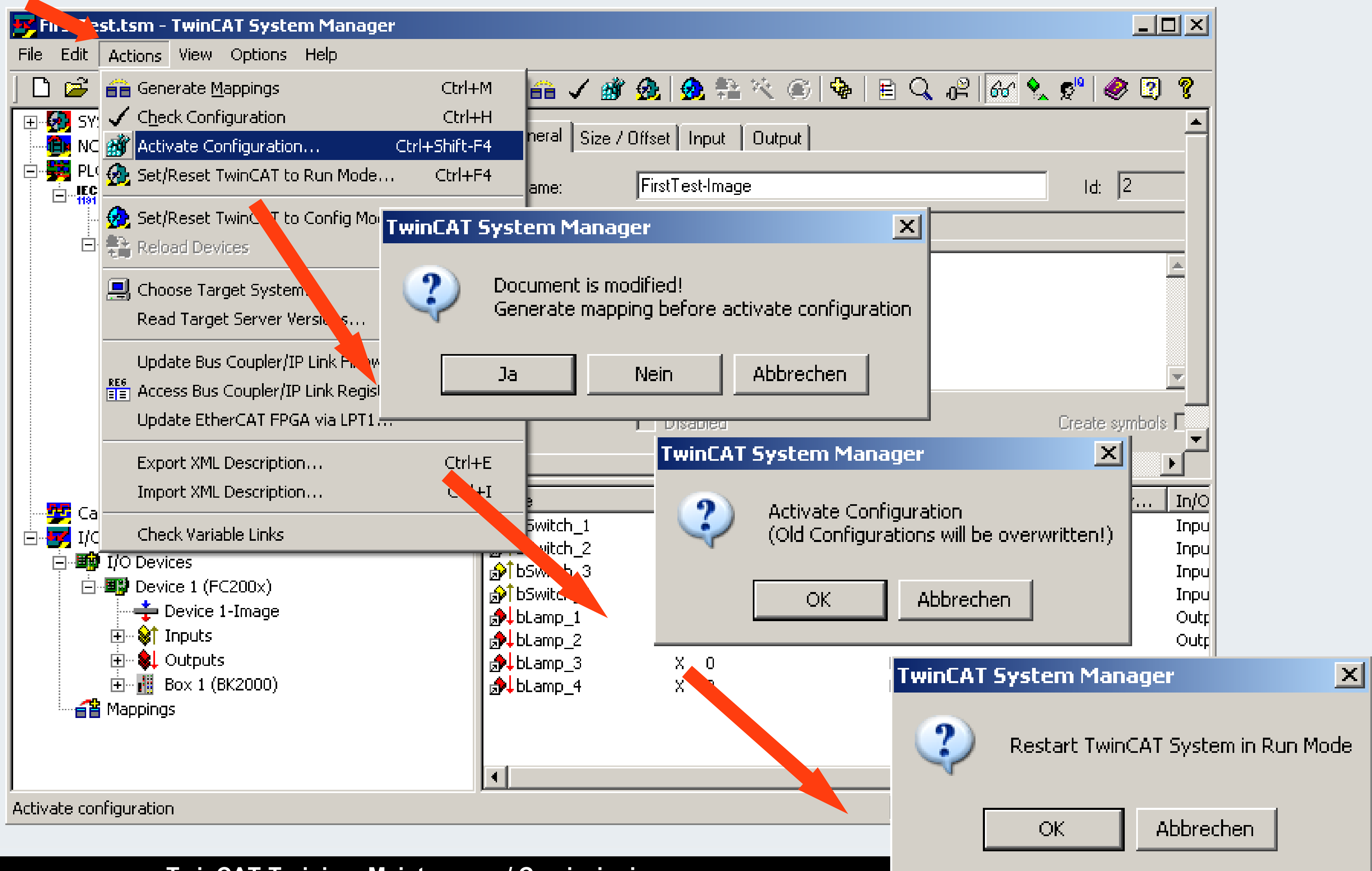
Link variables (repeat for all further variables)





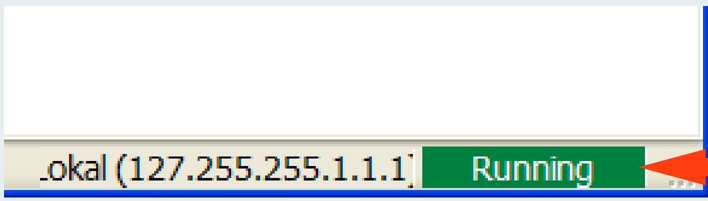
Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

Create new configuration



Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

Wait until TwinCAT operates in run mode

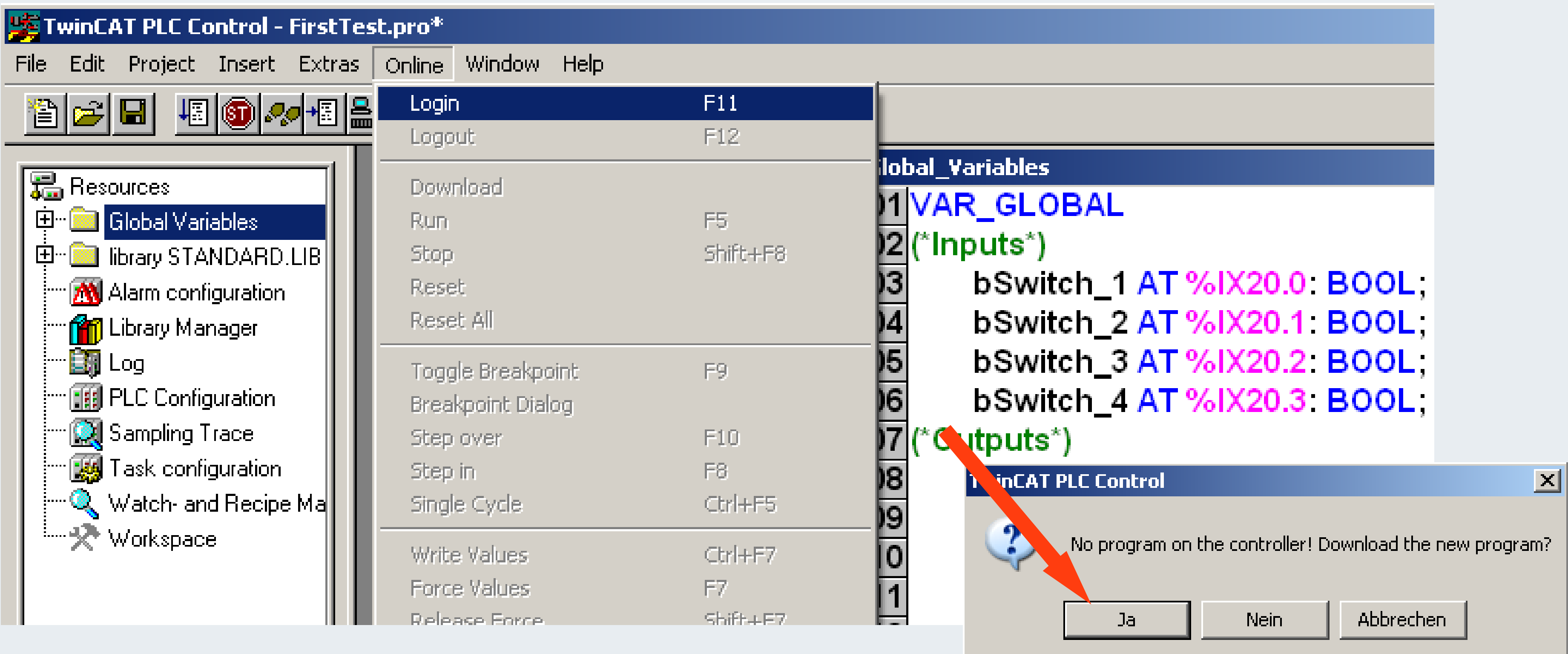


System Manager

TwinCAT system service in the taskbar

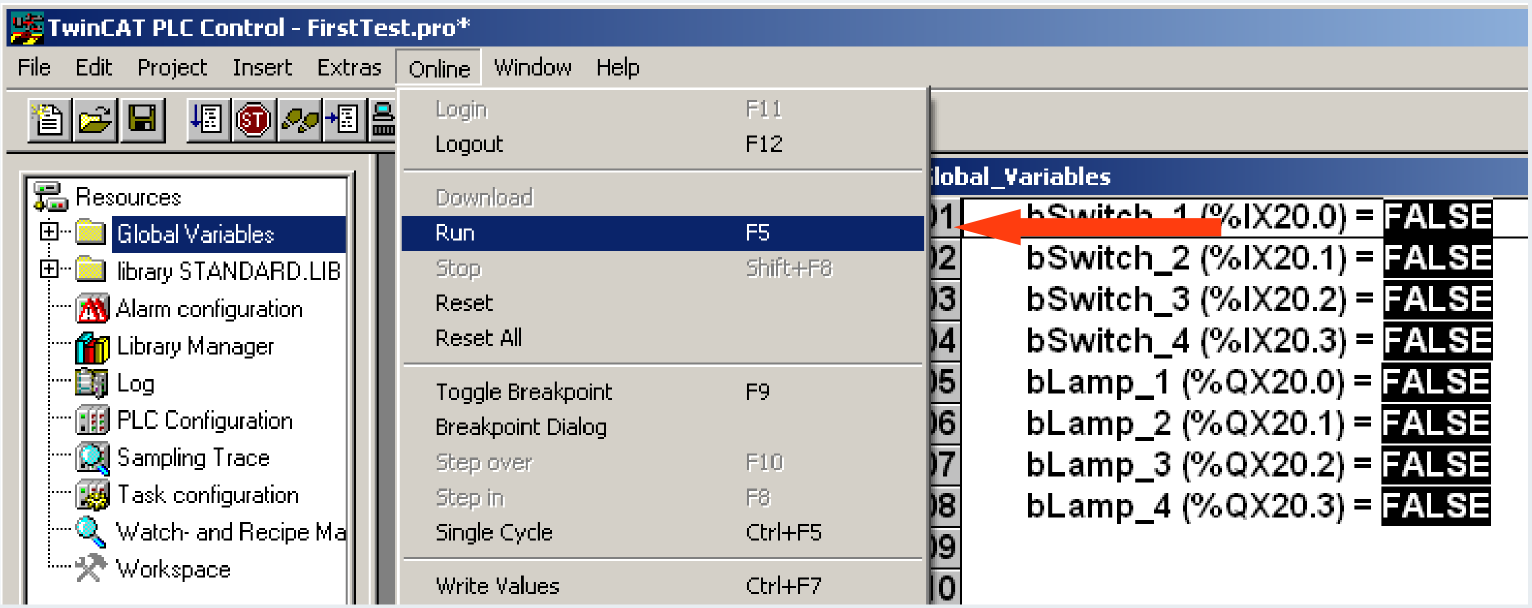
Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

Activate PLC Control, log in



Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

PLC Control, start PLC program



Expansion of an existing configuration with inputs and outputs in project FirstTest.pro

PLC Control, test variable online

Global_Variables

0001	bSwitch_1 (%IX20.0) = TRUE
0002	bSwitch_2 (%IX20.1) = FALSE
0003	bSwitch_3 (%IX20.2) = FALSE
0004	bSwitch_4 (%IX20.3) = FALSE
0005	bLamp_1 (%QX20.0) = FALSE
0006	bLamp_2 (%QX20.1) = FALSE
0007	bLamp_3 (%QX20.2) = FALSE
0008	bLamp_4 (%QX20.3) = FALSE
0009	

observe

through "Write Values"

Write values:

bSwitch_1 (%IX20.0) = TRUE
bSwitch_2 (%IX20.1) = FALSE
bSwitch_3 (%IX20.2) = FALSE
bSwitch_4 (%IX20.3) = FALSE
bLamp_1 (%QX20.0) = FALSE
bLamp_2 (%QX20.1) = FALSE < := TRUE>
bLamp_3 (%QX20.2) = FALSE
bLamp_4 (%QX20.3) = FALSE

Online Window Help

Login	F11
Logout	F12
Download	
Run	F5
Stop	Shift+F8
Reset	
Reset All	
Toggle Breakpoint	F9
Breakpoint Dialog	
Step over	F10
Step in	F8
Single Cycle	Ctrl+F5
Write Values	Ctrl+F7
Force Values	F7
Release Force	Shift+F7
Write/Force-Dialog	Ctrl+Shift+F7

bSwitch_1 (%IX20.0) = TRUE
bSwitch_2 (%IX20.1) = FALSE
bSwitch_3 (%IX20.2) = FALSE
bSwitch_4 (%IX20.3) = FALSE
bLamp_1 (%QX20.0) = FALSE
bLamp_2 (%QX20.1) = TRUE
bLamp_3 (%QX20.2) = FALSE
bLamp_4 (%QX20.3) = FALSE

IBK – T4

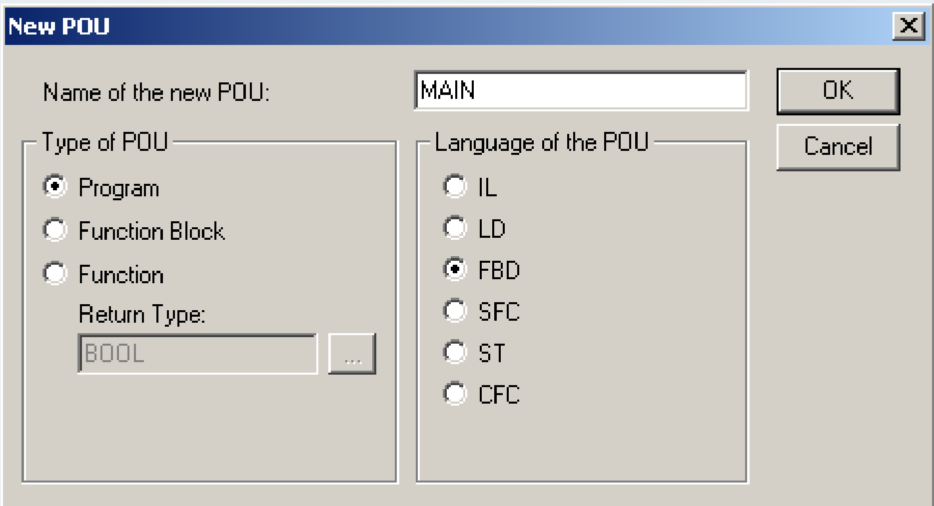
Where are the master cards entered in the System Manager?

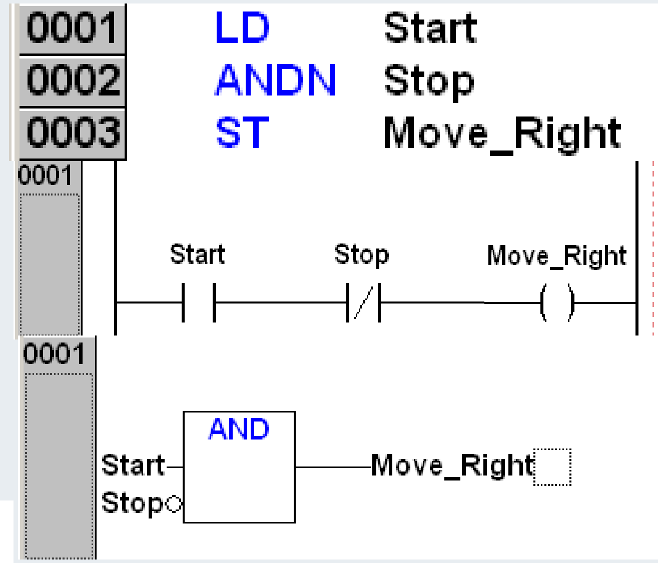
What menu item can be used to search for a master card that has not been entered?

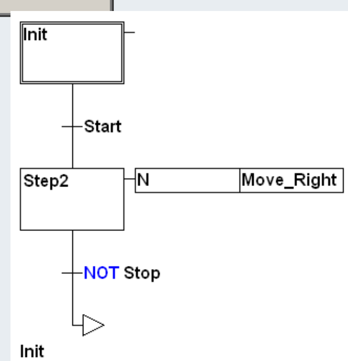
Which mouse button in the System Manager provides access to the Input Assistant for component selection?

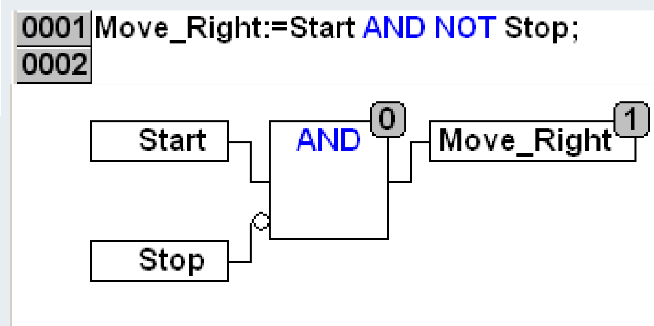
Where in the System Manager is the PLC Control interface?

PLC Control, further edit functions









IL: Instruction List
LD: Ladder Diagram
FBD: Function Block Diagram
SFC: Sequential Function Chart
ST: Structured Text
CFC: Continuous function chart

PLC Control toolbar (FBD)

File

New
Open
Save

Online

Start
Stop
Single Step
Breakpoint
Login
Logout
Global Search

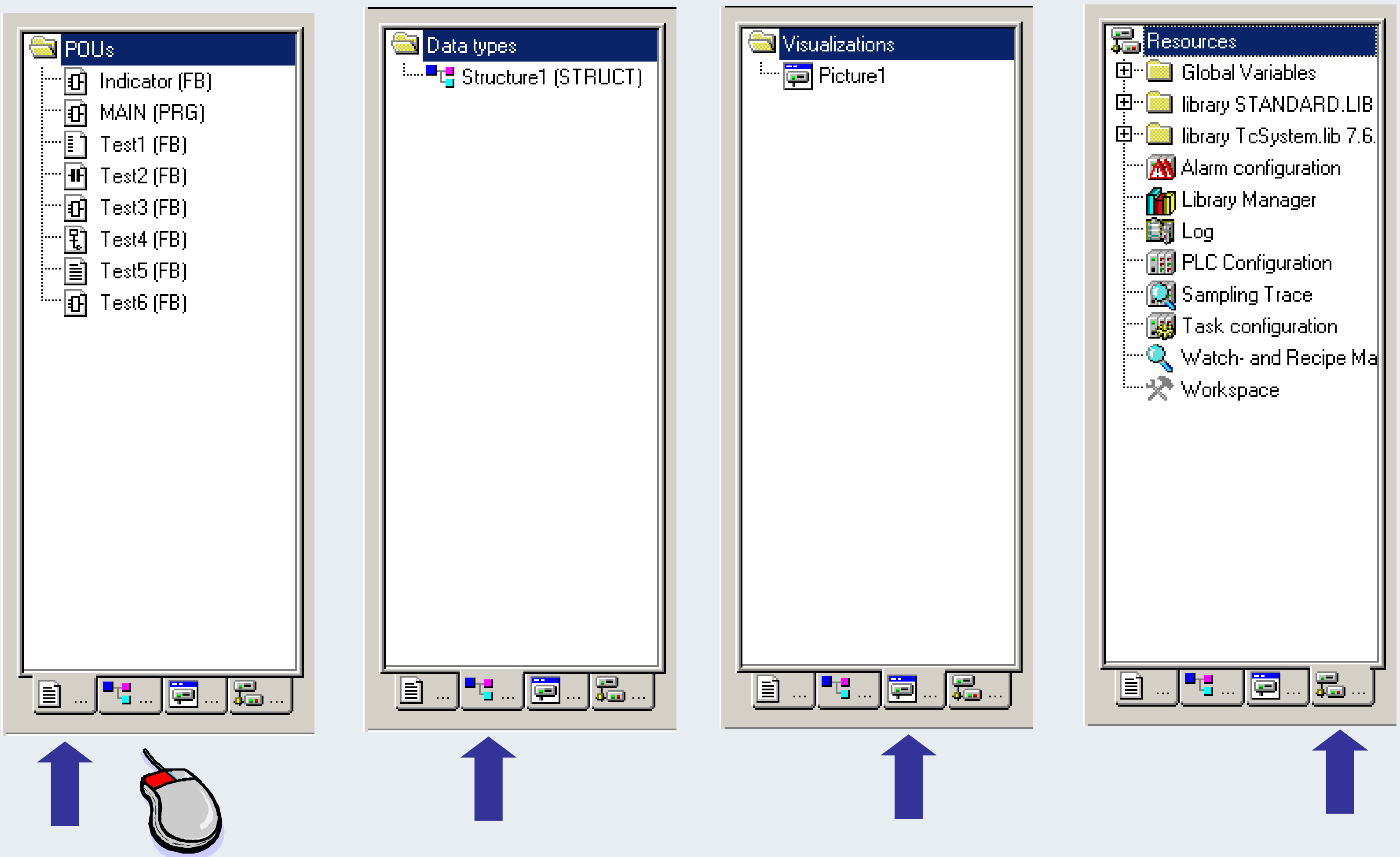
Element

Cut
Copy
Insert
Search
Find Next

Insert

Input
Output
Box
Assign
Jump
Return
Negate
Set/Reset

PLC Control, left window



PLC Control, right window

Local declaration window

MAIN (PRG-FBD)

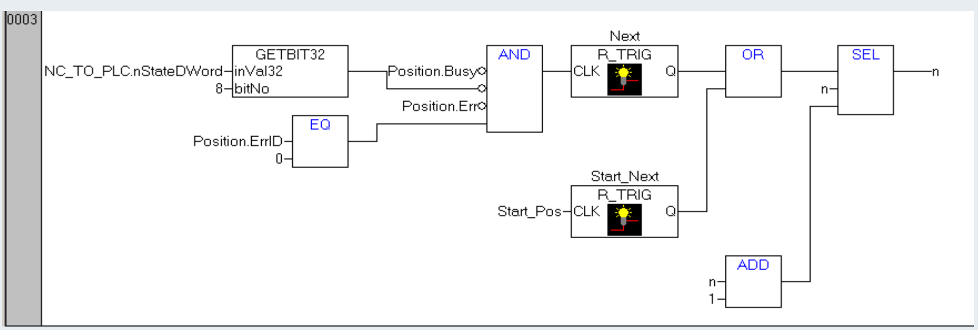
```
0001 PROGRAM MAIN
0002 VAR
0003     Indicator_1: Indicator;
0004     Help:BOOL;
0005 END_VAR
```

Global declaration window

Global_Variables

```
0001 VAR_GLOBAL
0002 (*Inputs*)
0003     bSwitch_1 AT %IX20.0: BOOL;
0004     bSwitch_2 AT %IX20.1: BOOL;
0005     bSwitch_3 AT %IX20.2: BOOL;
0006     bSwitch_4 AT %IX20.3: BOOL;
0007     iAnalog_1 AT %IB0:INT;
0008     iAnalog_2 AT %IB2:INT;
0009 (*Outputs*)
0010     bLamp_1 AT %QX20.0: BOOL;
0011     bLamp_2 AT %QX20.1: BOOL;
0012     bLamp_3 AT %QX20.2: BOOL;
0013     bLamp_4 AT %QX20.3: BOOL;
0014 END_VAR
```

Programming window CFC



IBK – T5

What block types are available?

How does a variable have to be declared so that it can be accessed from other blocks?

Between which keywords does a local variable have to be declared?

Where can an alphabetical list of the blocks that have created be found?

Under which Input Assistant category can the installed library blocks be displayed?

IBK – T6

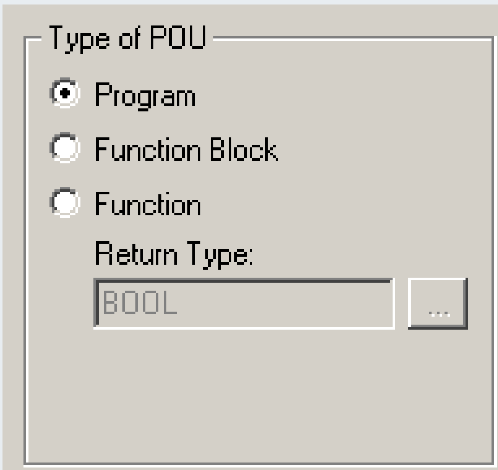
What steps have to be carried out in PLC Control after activation of System Manager for running the PLC program?

What colour does the TwinCAT icon have when the system is stopped?

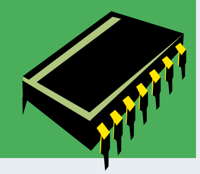
At what system time is it possible to log into the PLC and start the program?

Does the configuration have to be reactivated after a new variable link has been created?

Block types



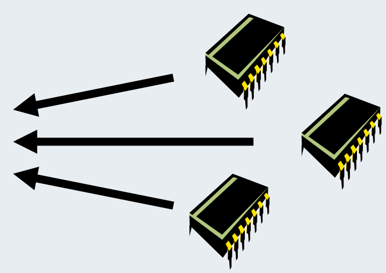
Program
1. Can call other programs, function blocks, and functions
2. Retains the state of local variables between program calls



Function Block
1. Can call other function blocks and functions
2. Retains the state of local variables between program calls
3. The function block program code can be used repeatedly, in each case with a different memory

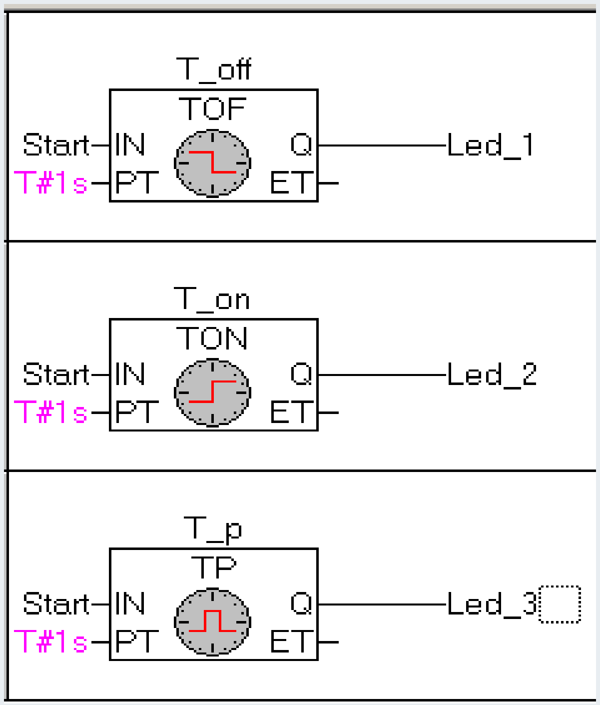
Function
1. Has no memory
2. Returns the result via the function name

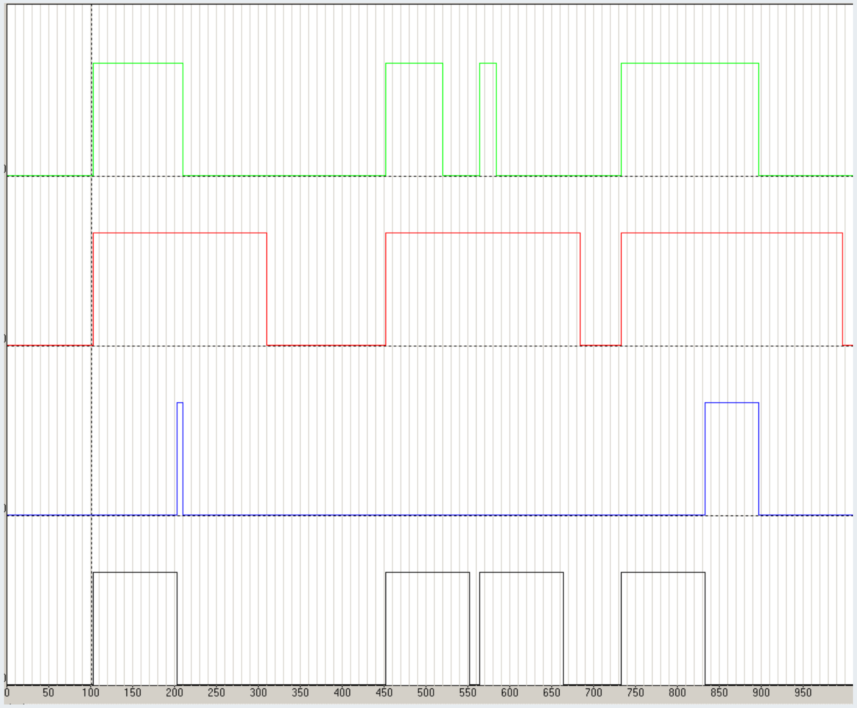
```
LD Var1
ADD Var2
GE limit
ST enable
```



Timer

Start



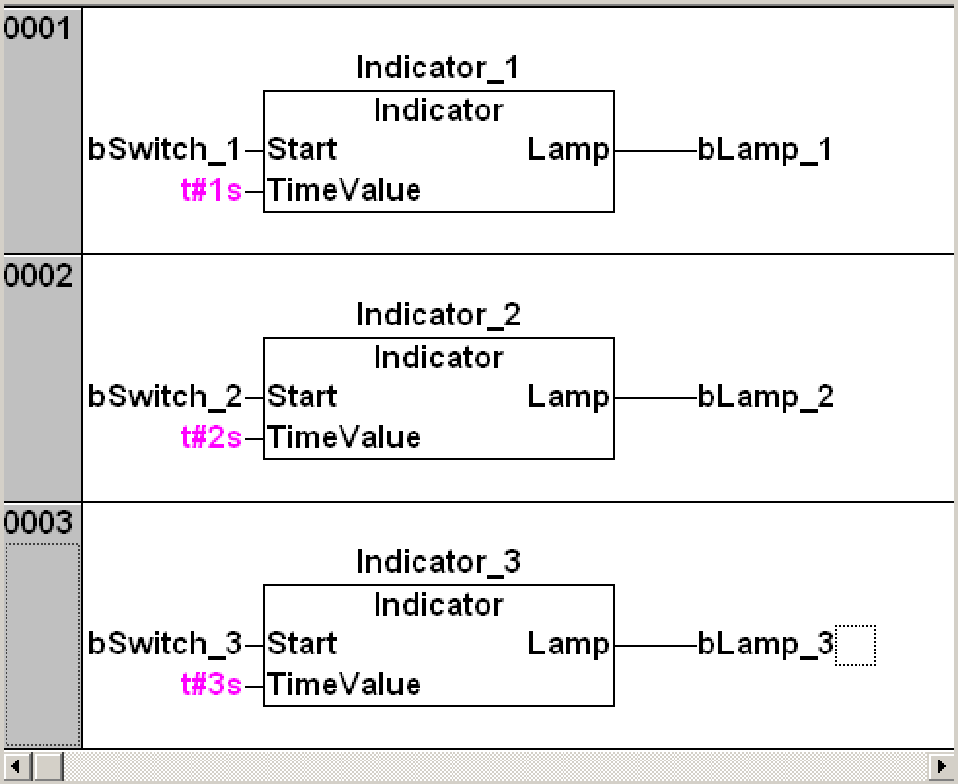


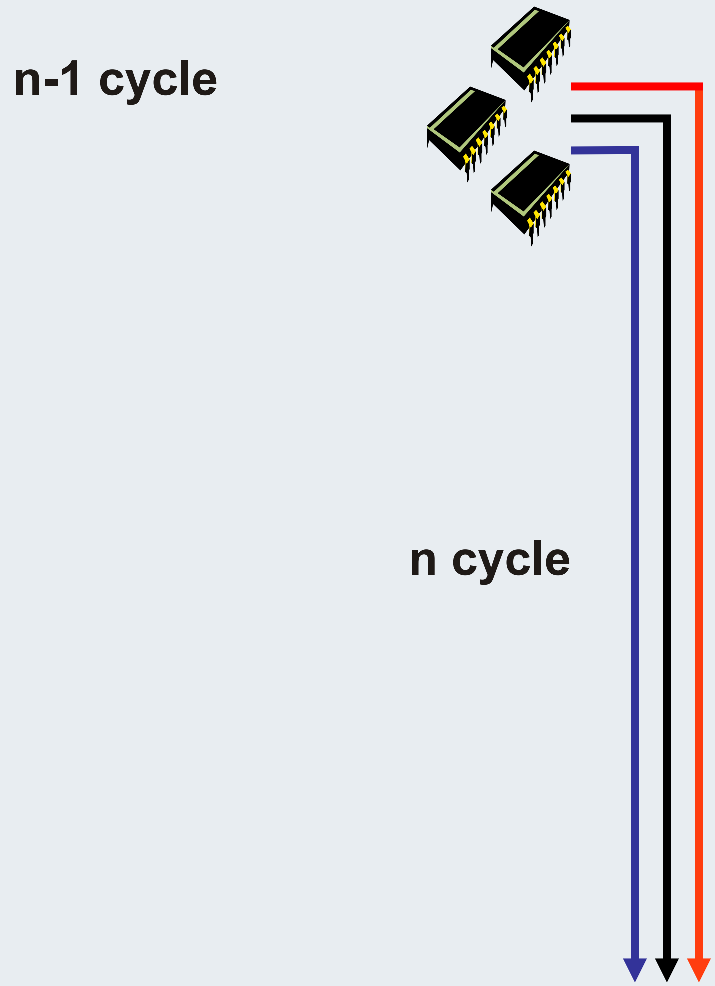
T/10ms

Instantiation of function blocks

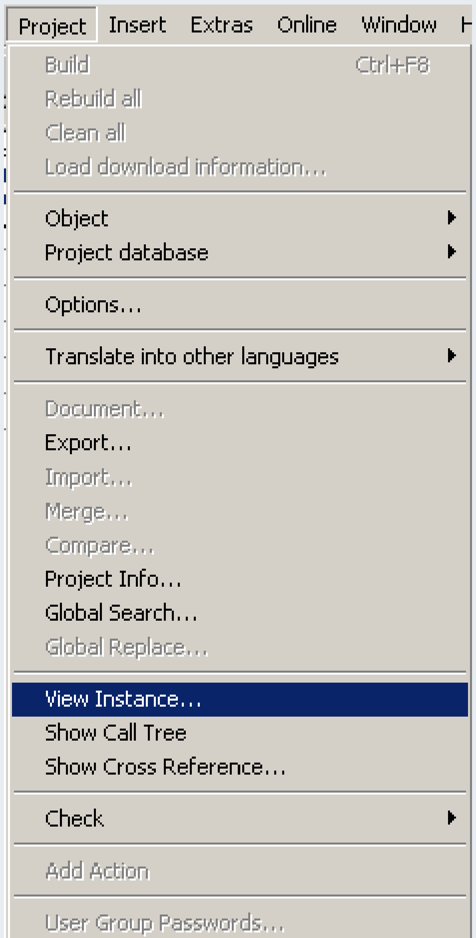
MAIN (PRG-FBD)

```
PROGRAM MAIN
VAR
    Indicator_1: Indicator;
    Indicator_2: Indicator;
    Indicator_3: Indicator;
END_VAR
```



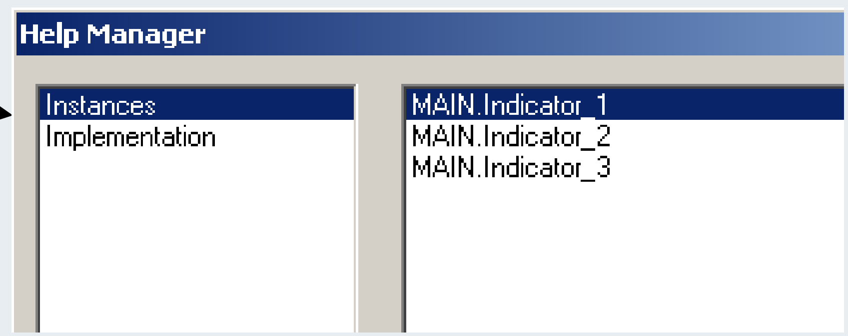


Display of individual instances



Note the following if this menu item is not selectable (greyed out):

1. **Project must be logged in.**
2. **The block should be marked in the left window and displayed in the right window.**



IBK – T7

Which block type offers multiple parameterisation based on the same logic?

Which block type returns the result in the block name?

How is a linked variable identified in the System Manager?

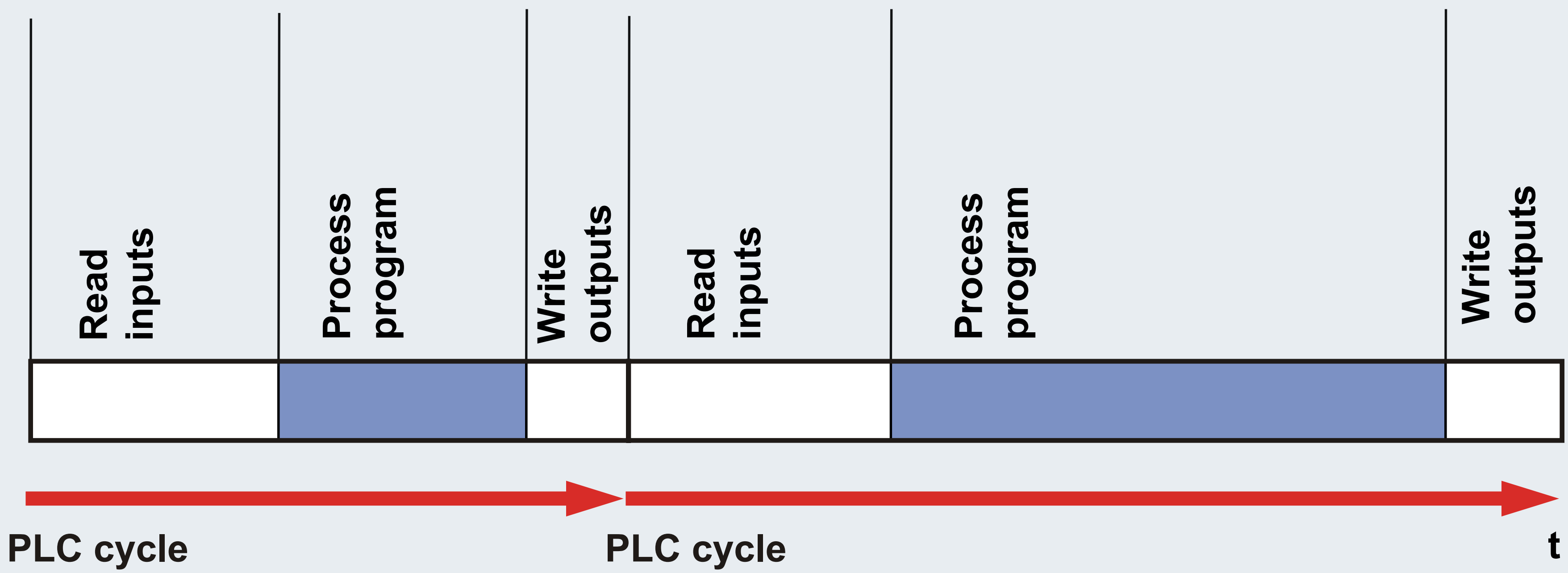
What options are available for querying the output of a function block?

What additional variable types that are not contained in the program block can be found in the function block?

SPS Task‘s

- Standard PLC: Programs are processed cyclically: fixed cycle time is one of the operating modes

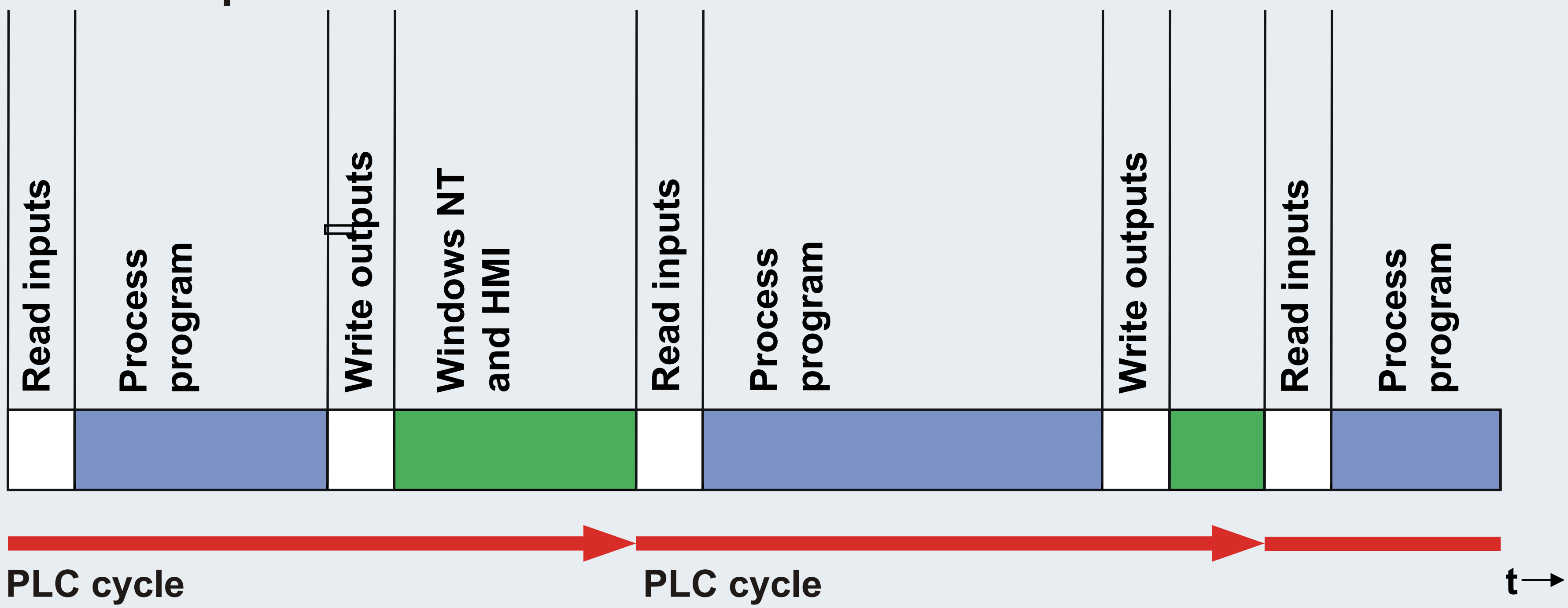
Real-time operation of PLC software in a classic PLC



Implementation: Basic tasks of PC control
Operation of a software PLC on the PC

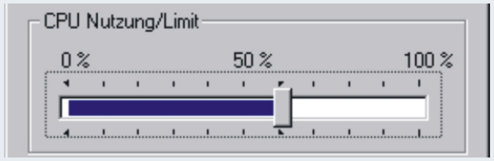
- Software PLC: Computing capacity is reserved for the PC operating system
- The software PLC operates with a fixed cycle, the PC operating system and the user interface in the period between cycles

Real-time operation of PLC software on a PC

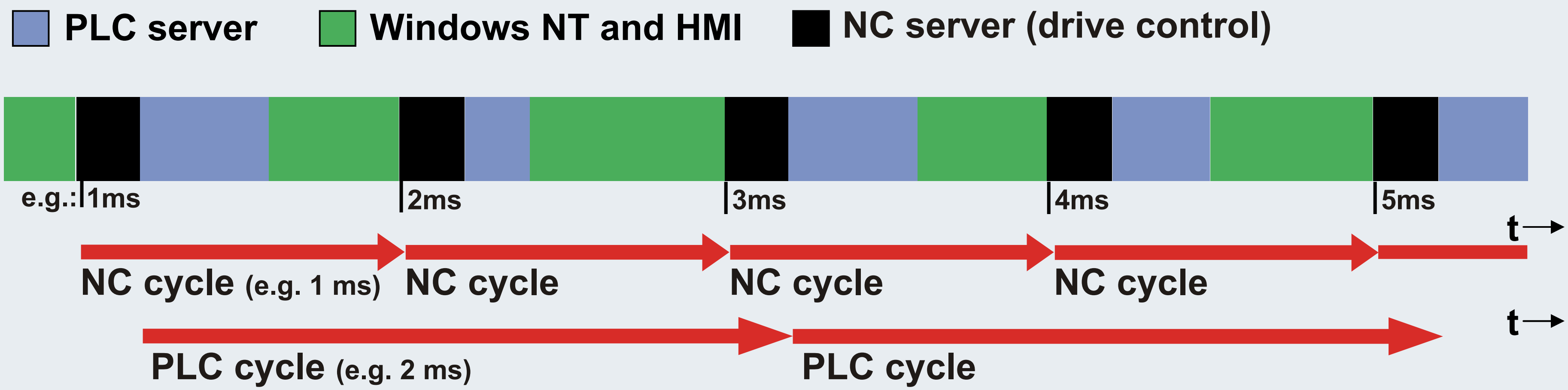


Implementation: Basic tasks of PC control Operation of a software PLC and software NC on the PC

- PLC tasks and NC drive control processed <u>deterministically</u> via multitasking
- Computing capacity is regularly made available for the operating system



Real-time operation of software for PLC and NC on a PC

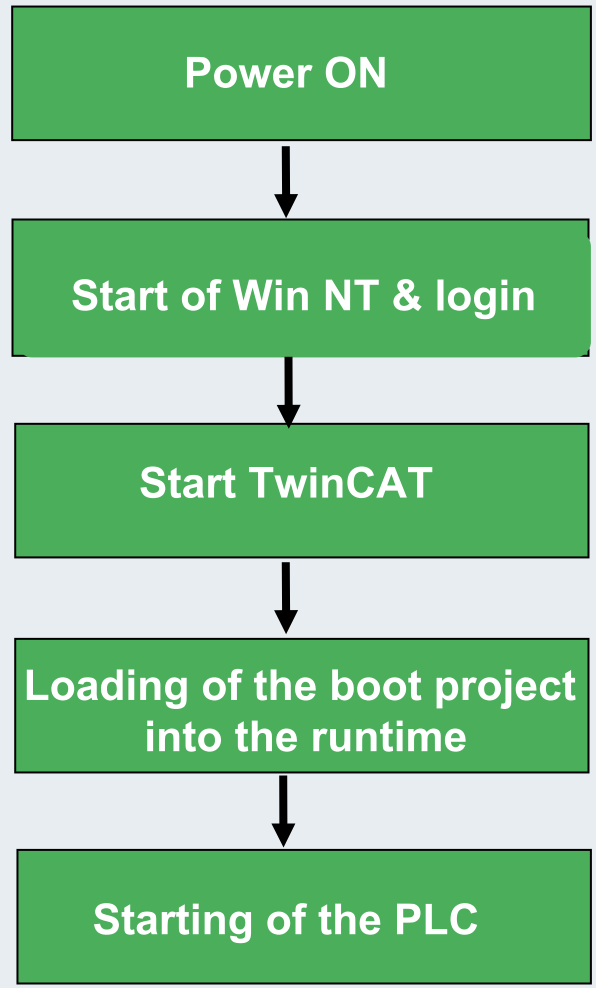


Automatic PLC start

Request:

Once the computer has been switched on, it should be possible to automate loading and starting of the PLC project.

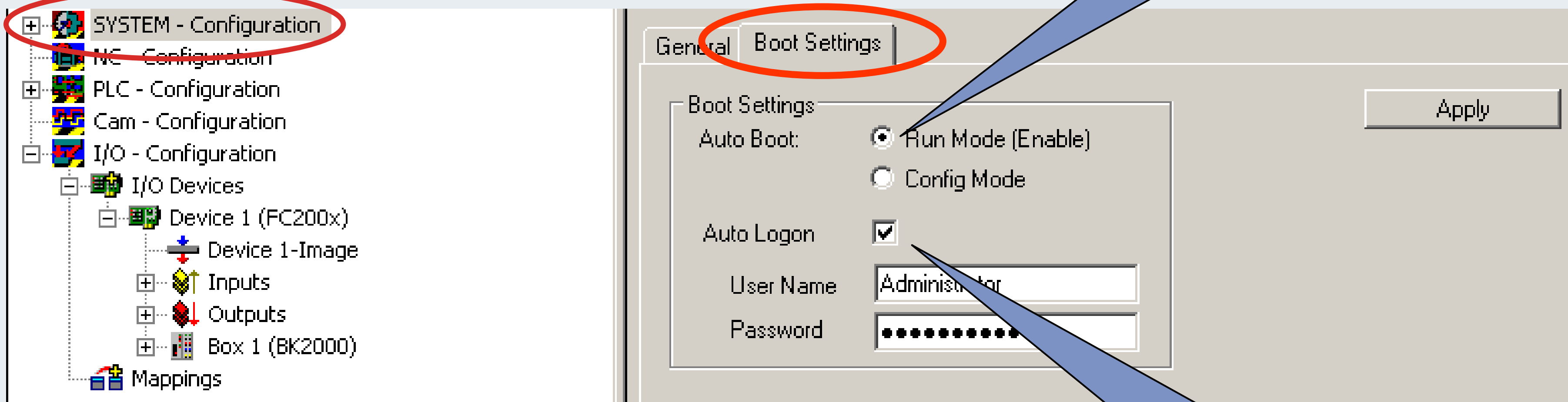
Procedure



TwinCAT autostart

Automatic start of TwinCAT

System Manager

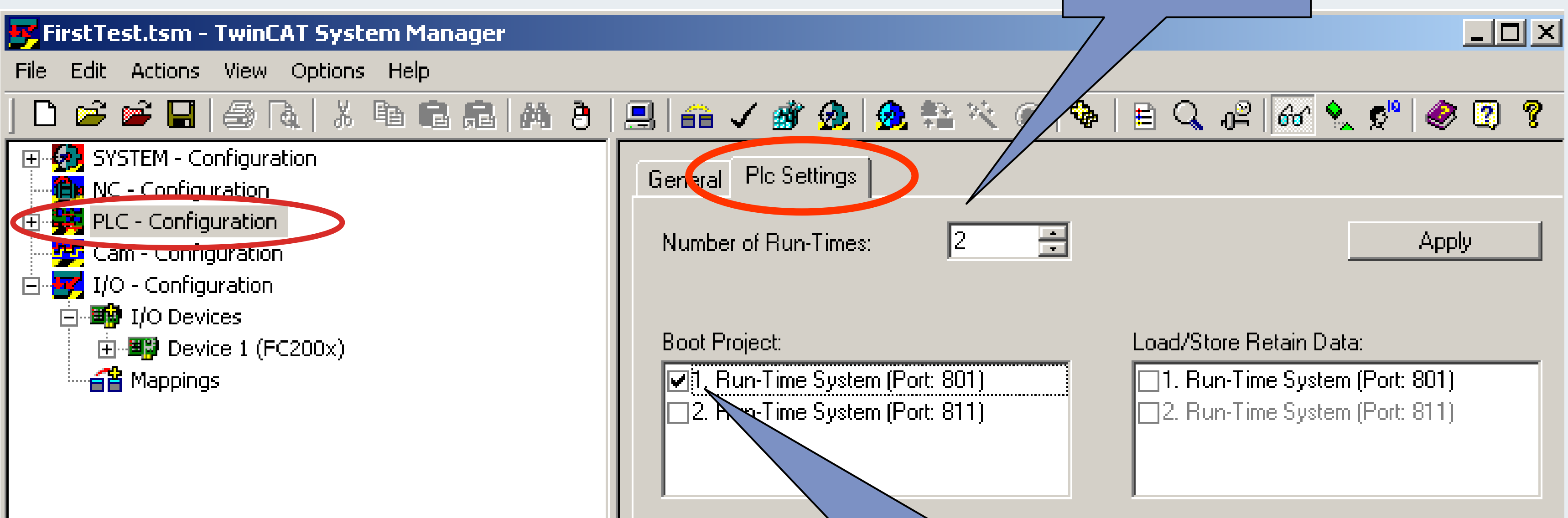


Automatic login under Windows with selected users and password

Selection of runtime [1..4]

1.
The user should know under which PLC (runtime) the project was configured in PLC Control

Number of Run-Times

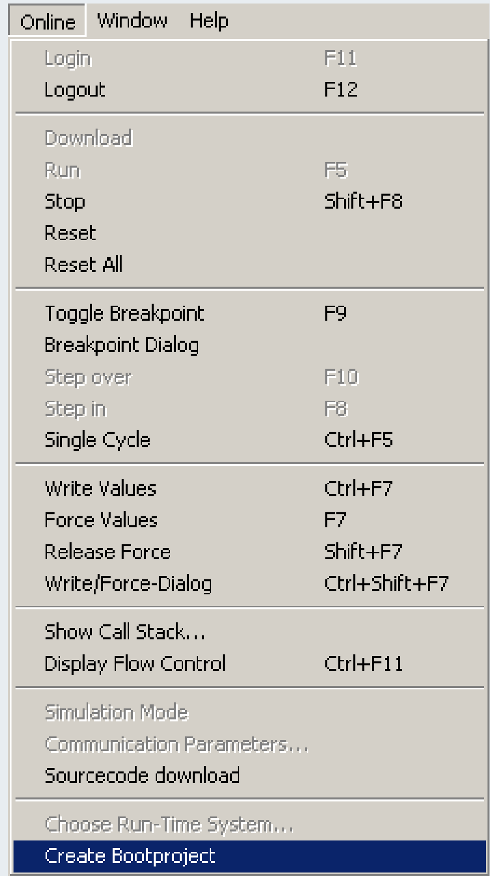


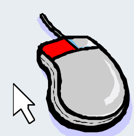
Run-Time selection and PLC start

Creation of a boot project in PLC Control

Requirements:

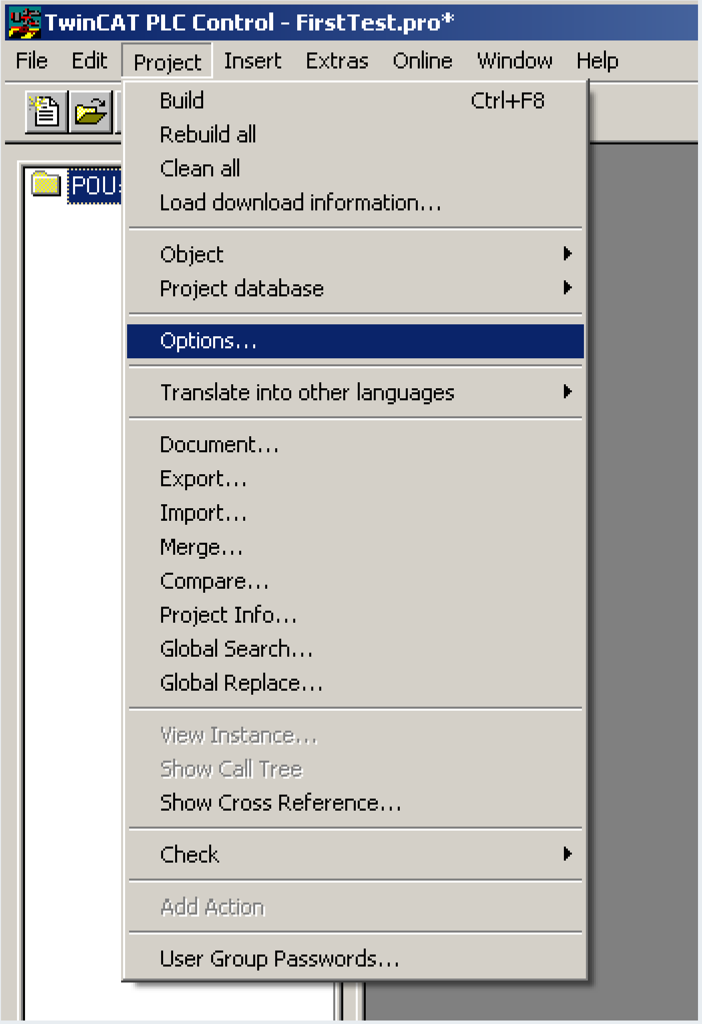
1. The machine should operate correctly.
2. Hardware, software and links are correct.
3. PLC Control is logged in





Saving of source code

- Select the „Project“ field in the menu bar.
- A selection window opens.
- Select „Options“.

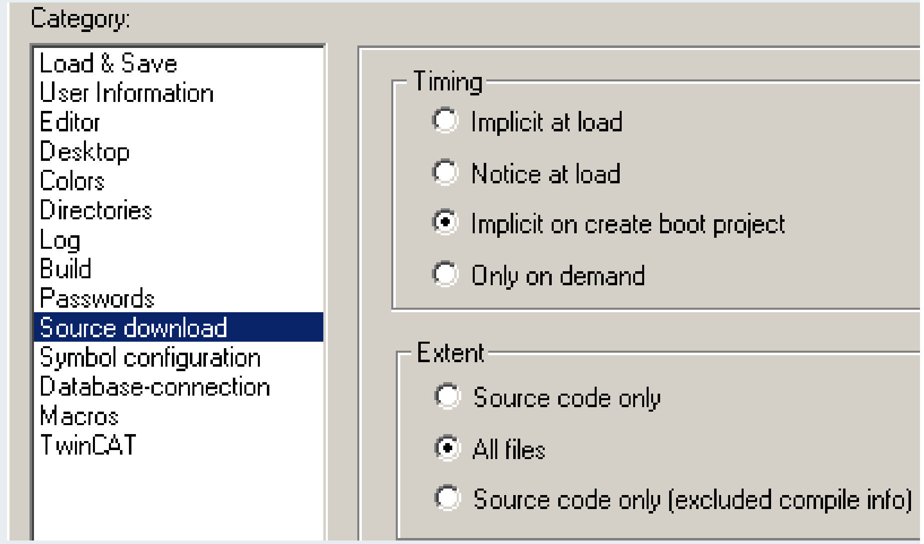


Time of source download

Implicit during loading:
The selected scope is written to the control computer each time the PLC project is opened.

Note on loading:
If the PLC project has been modified, a message box saying „Load source code into the controller?“ will appear during loading.

Create implicitly with a boot project:
The selected scope is written to the control computer each time a boot project is created.

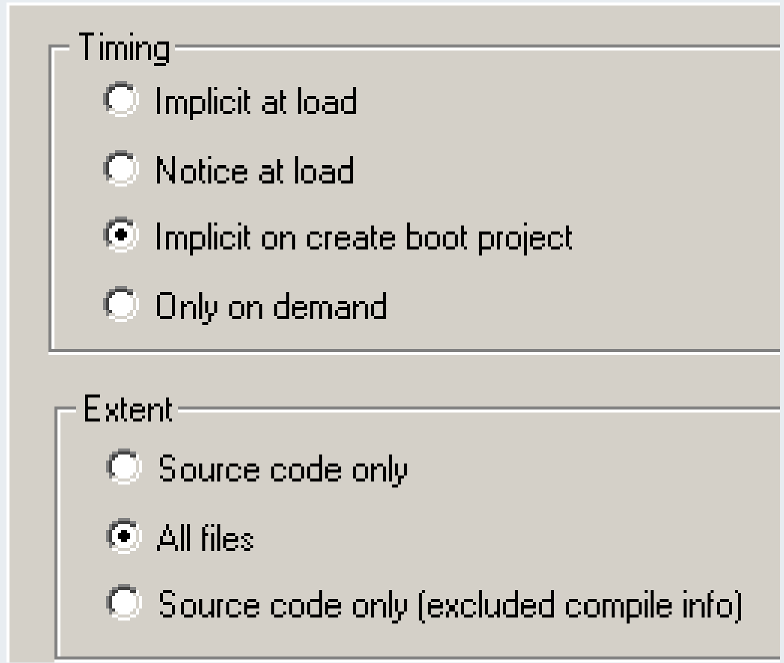


Time/scope of source download

On Demand:
The source code is only loaded into the controller on request.
Online/Load source code

Scope:
Source code only;
The PLC project is written to the control computer.

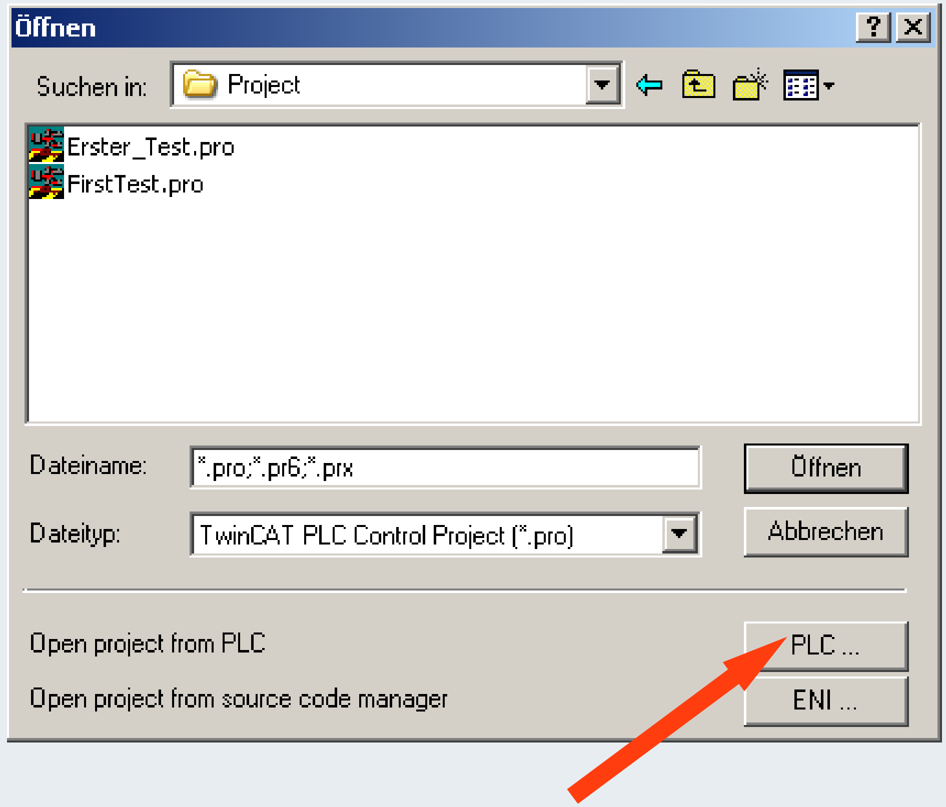
All files:
The PLC project including the libraries is written to the control computer.



Open PLC project from the controller

The current PLC project can be opened directly from the controller.

Select button „Open project from PLC“ under „Open File“.



Loading the source code (PLC project) from a different controller

Once the source code has been created on the controller, it can be loaded remotely

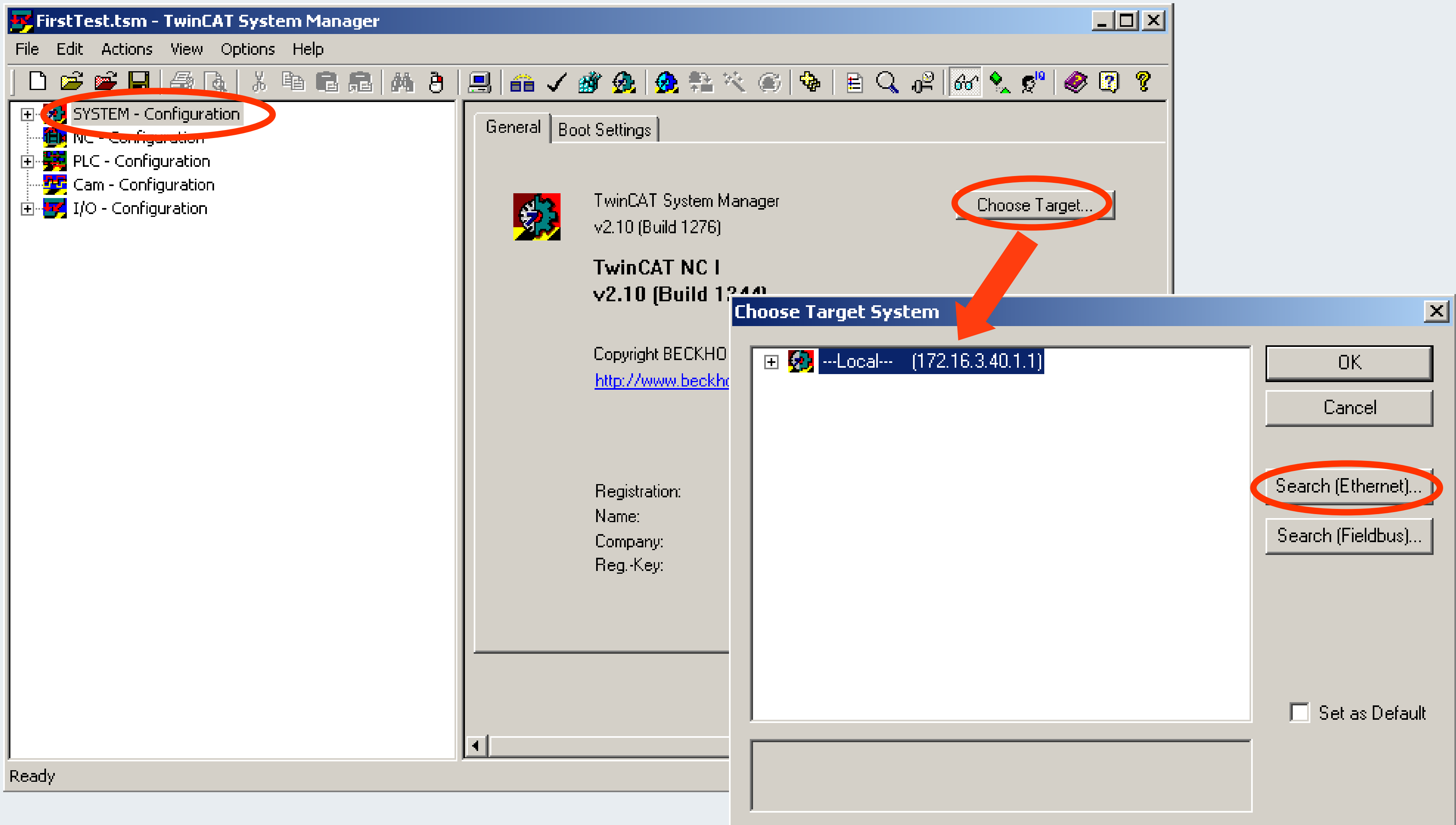
„Programming computer“ with Tc

Ethernet cross cable or network

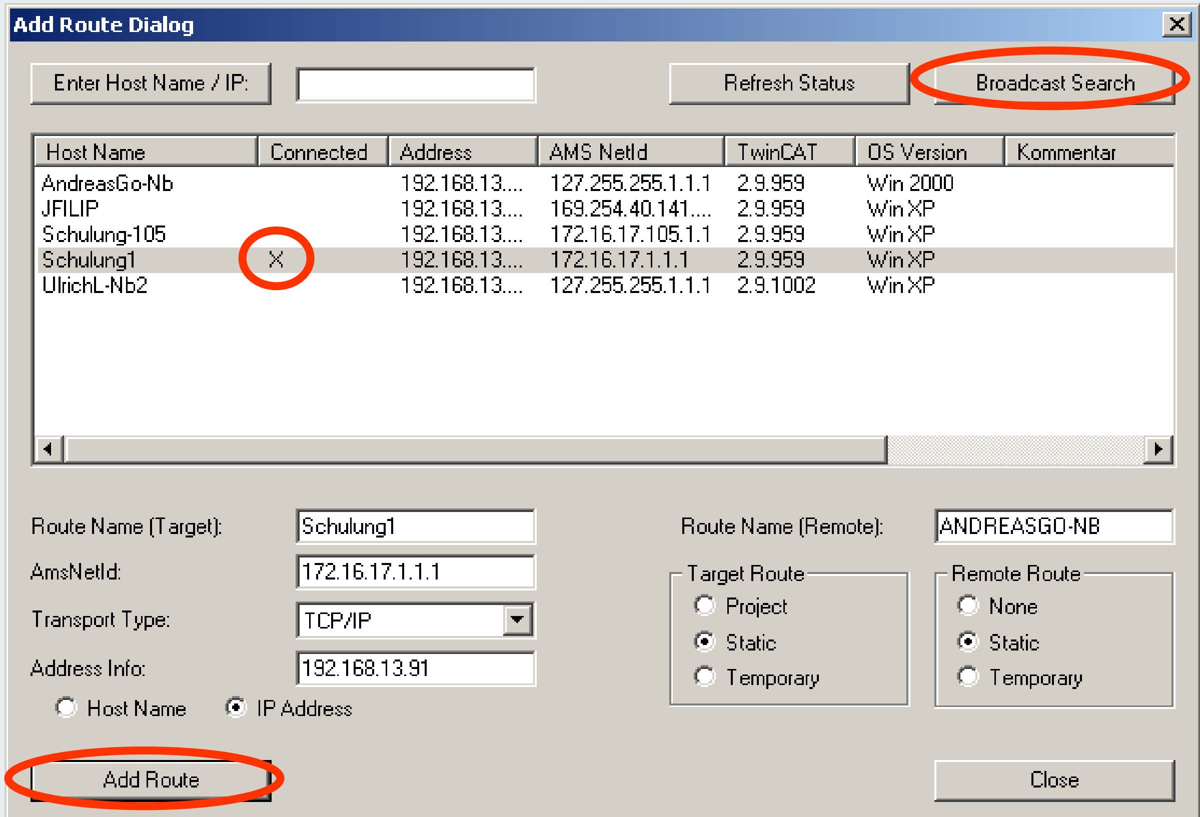
TwinCAT control computer with PLC source code

Establish remote connection

System Manager

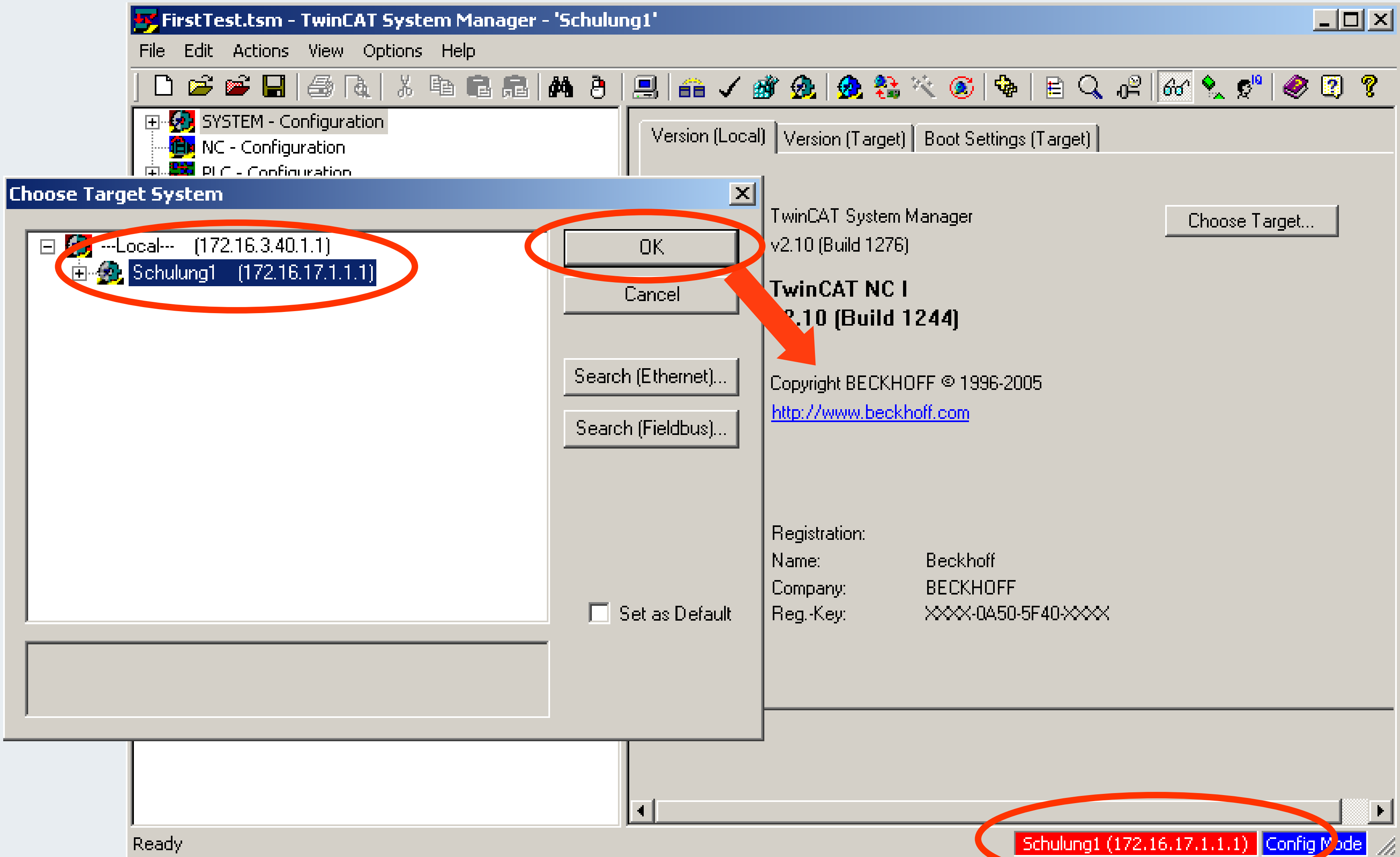


Establish remote connection
Search computer

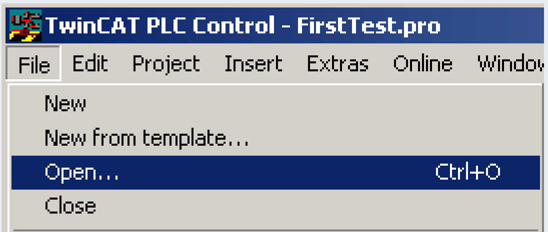


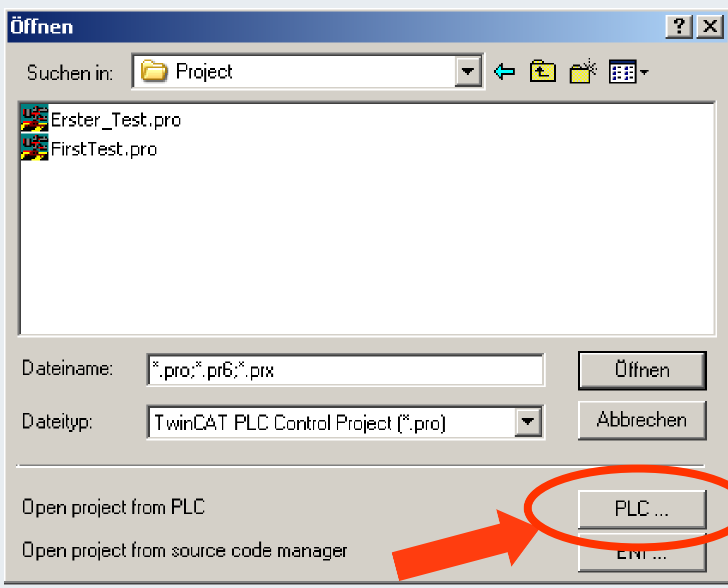
First search for all TwinCAT PCs in the network via the „Broadcast Search“ button. Then select the PC to be connected and press „Add Route“.
An X for „Connected“ appears once the password has been entered.
Should this not be the case, press „Refresh State“.

Select target system for System Manager connection



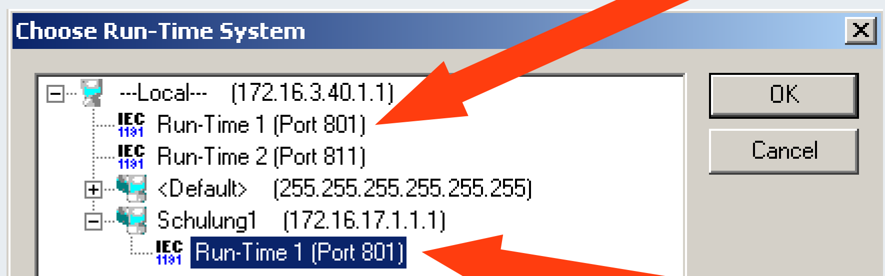
Open PLC project from the controller



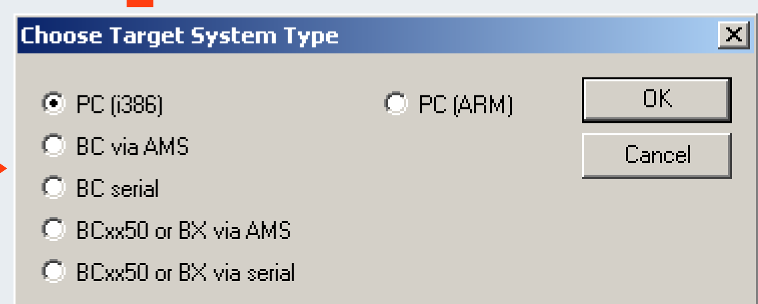


The current PLC project can be opened directly from the controller.

Lokal



Remote



Open PLC project from the controller

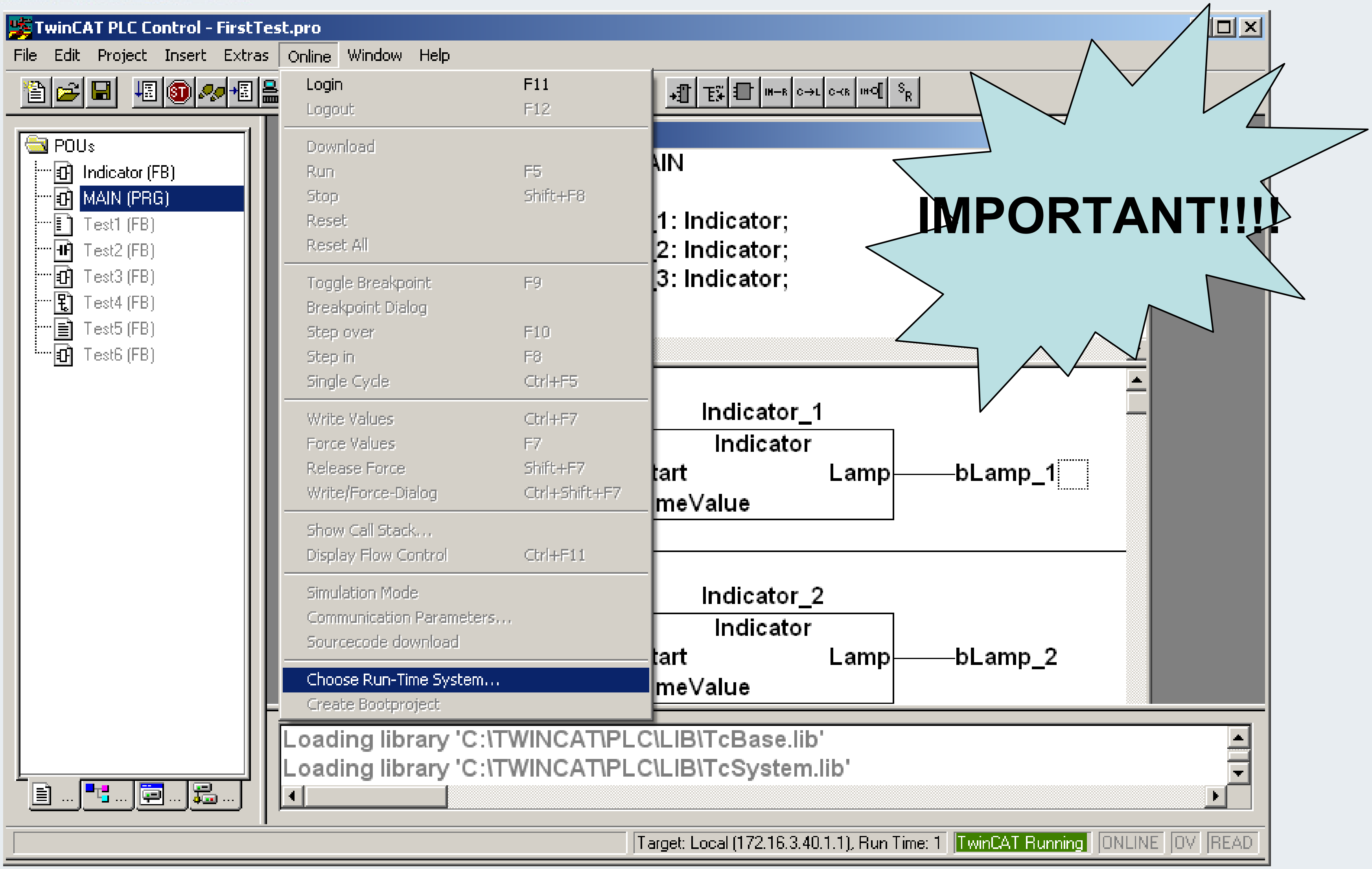
The reloaded project can be used for:

a) **Save as „Copy from controller“**

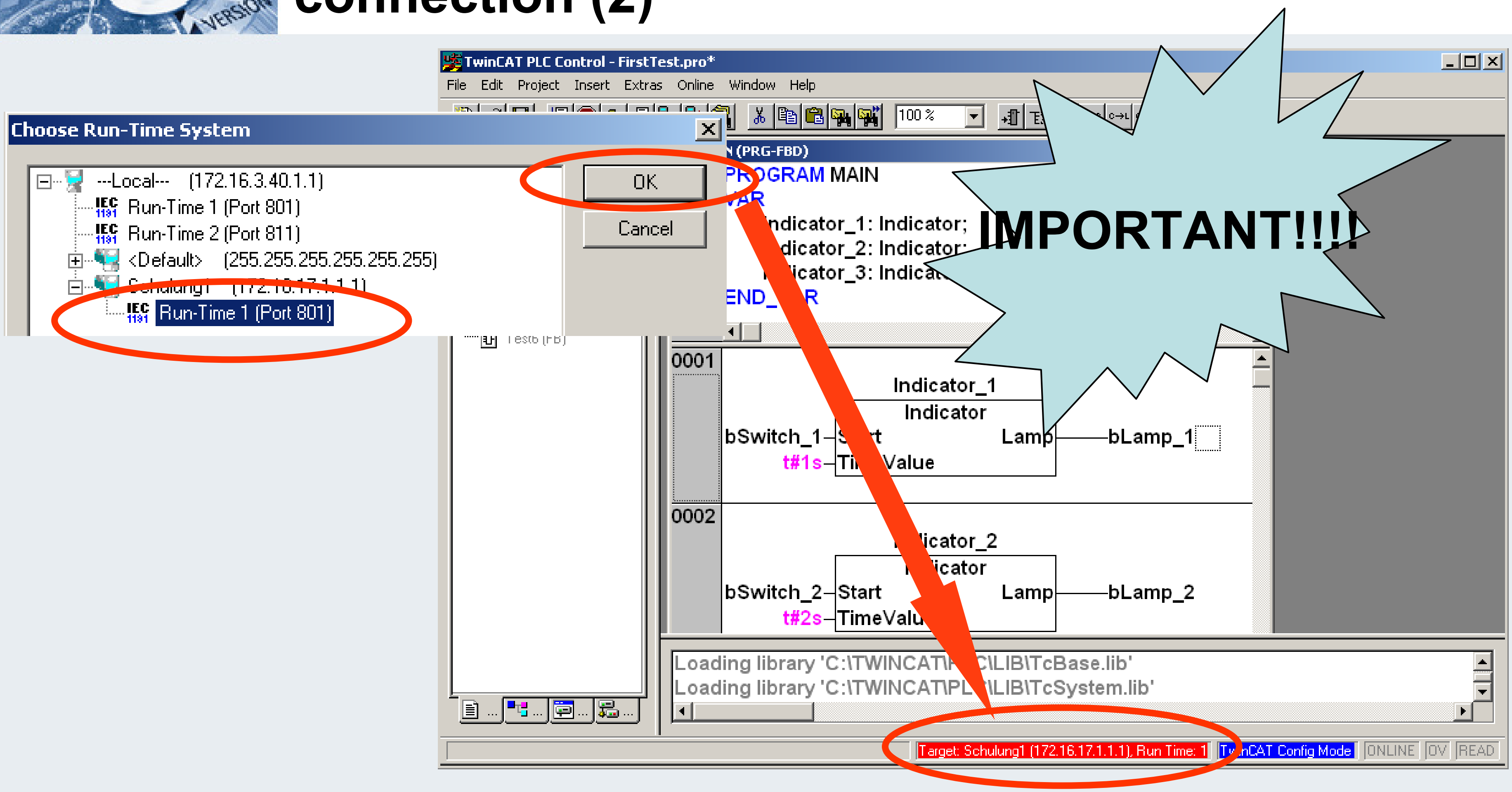
b) **Comparison with other projects**

c) **Continue operation directly on the other controller**

Select target system for PLC Control connection (1) (target computer + PLC runtime)



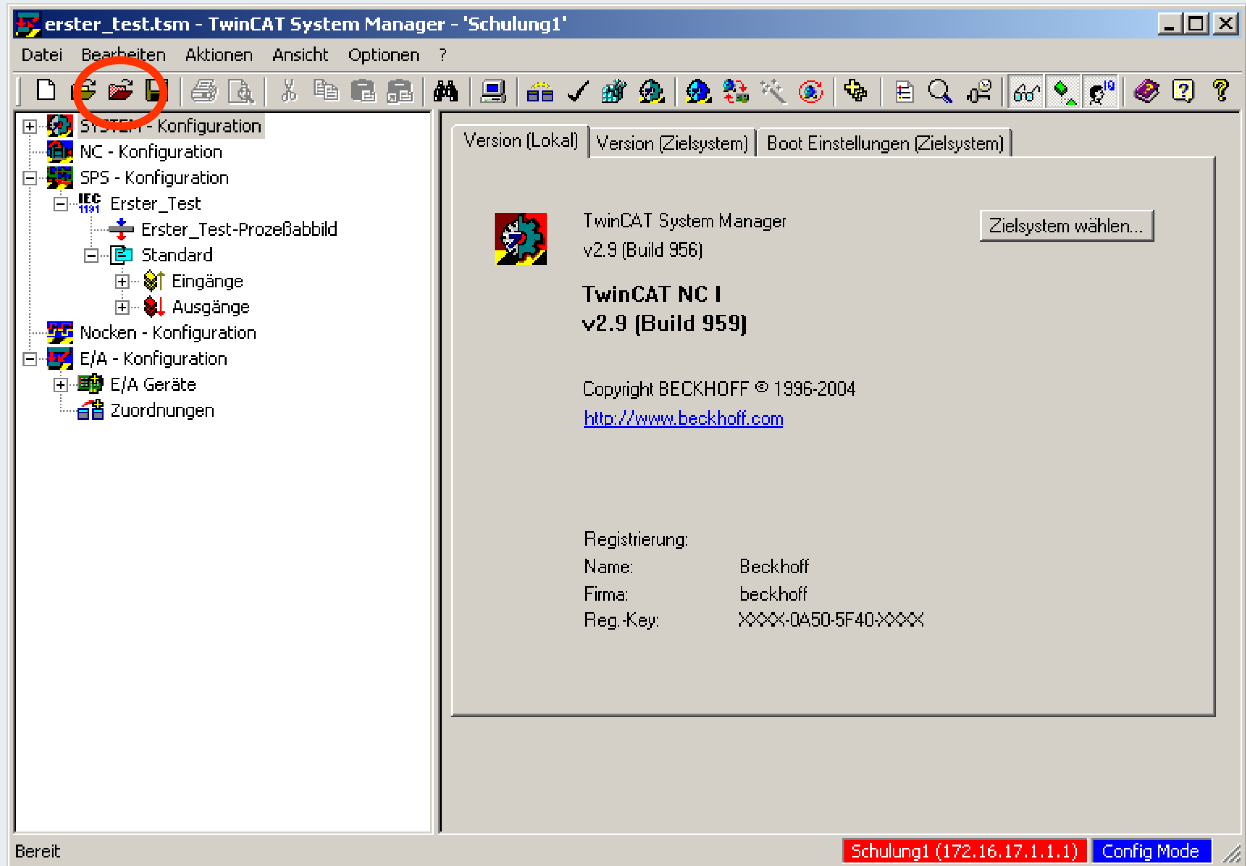
Select target system for PLC Control connection (2)



On login the target system displayed is used!

Opening a System Manager project from the target system

The System Manager project may be loaded from the controller.



IBK – T8

How can TwinCAT be started automatically?

Where are automatic login under Windows and automatic PLC start-up set?

What further steps have to be carried out in PLC Control for starting the PLC program automatically?

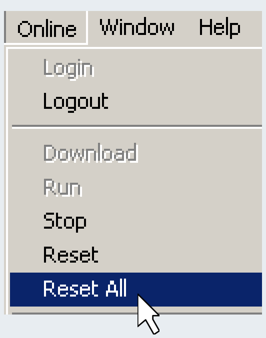
Where does runtime selection and PLC start have to be entered?

Data remanence

PERSISTENT

```
MAIN (PRG-FBD)
0001 PROGRAM MAIN
0002 VAR PERSISTENT
0003     Position:Calculate ;
0004     Value:INT;
0005     Input_1:BOOL;
0006 END_VAR
```

The data can only be deleted through a Reset All

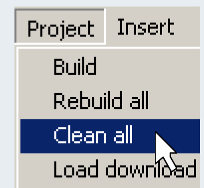


Enable archiving not required

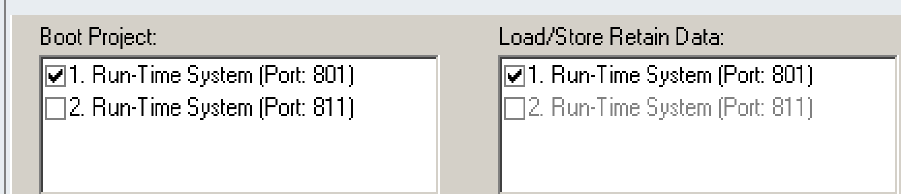
RETAIN

```
MAIN (PRG-FBD)
0001 PROGRAM MAIN
0002 VAR RETAIN
0003     Position:Calculate ;
0004     Value:INT;
0005     Input_1:BOOL;
0006 END_VAR
```

The data can be deleted through „Reset“ or „Clean all“



Enable archiving



Data remanence

Notes:

Remanent data (persistent and retain) are only stored if the TwinCAT system service is terminated cleanly. This usually involves using a UPS for shutting down the computer.

Data are also written if the computer is shut down manually

A new boot project has to be created if „persistent“ and „retain“ variables are modified.

IBK – T9

Which remanent variable types can be created?

How can persistent variables be deleted?

Do persistent variables have to be enabled?

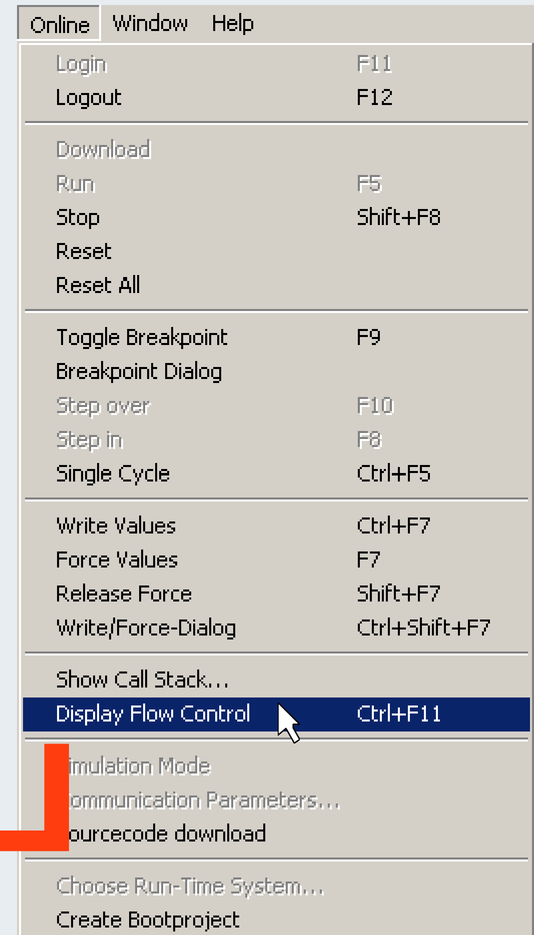
How can retain variables be deleted?

How are retain variables enabled?

Debugging and search functions in PLC Control and in the System Manager

Requirement:

- **Project must be logged in**
- **Project must be running**

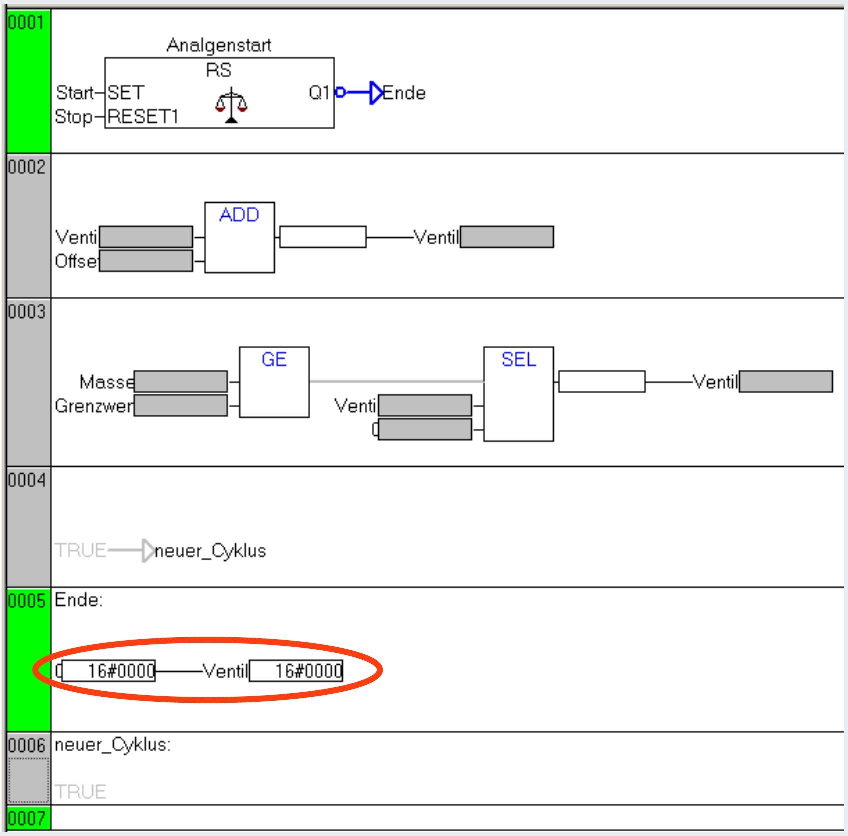


If flow control is activated, each line or network executed during the last control cycle is marked.

The number field of the active rows or networks are shown in green.

Flow Control

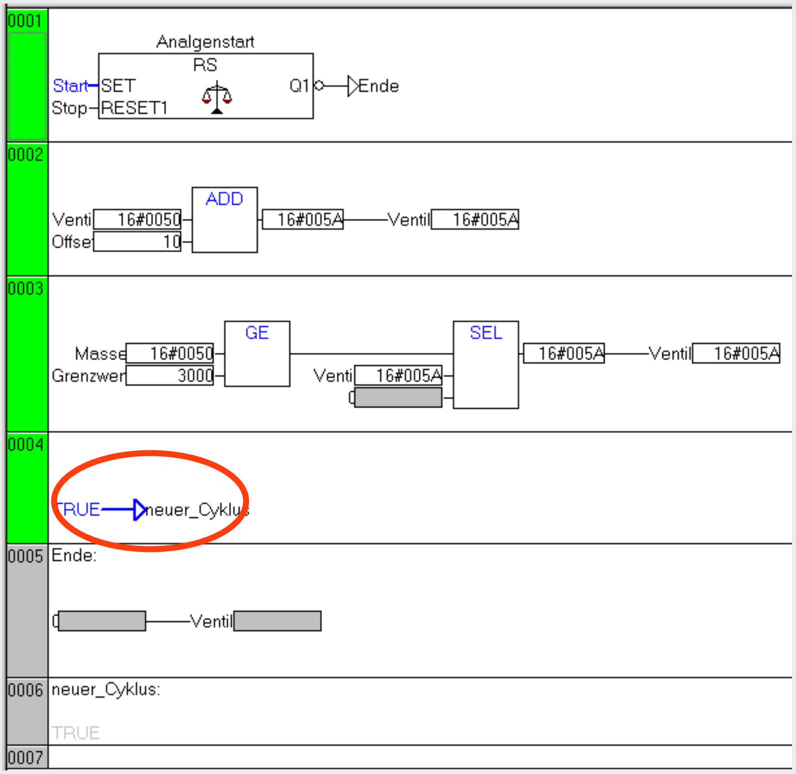
- A further field is inserted for all connecting lines not transporting Boolean values.
- If these outputs and inputs are assigned, the value transported via the connecting line is displayed in this field.



Flow Control

Connecting lines exclusively transporting Boolean values are only shown in blue if the transport is TRUE.

The flow of information can thus be monitored continuously.



IBK – T10

Where can flow control be set?

What precondition has to be met so that flow control can be switched on?

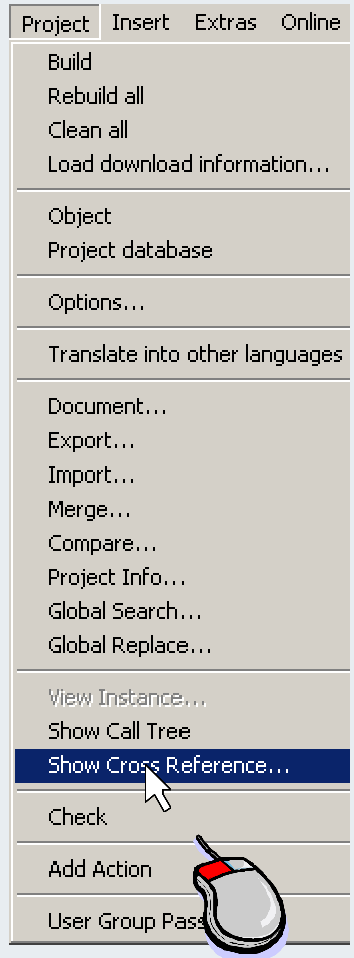
What can be controlled via flow control?

How is network processing indicated?

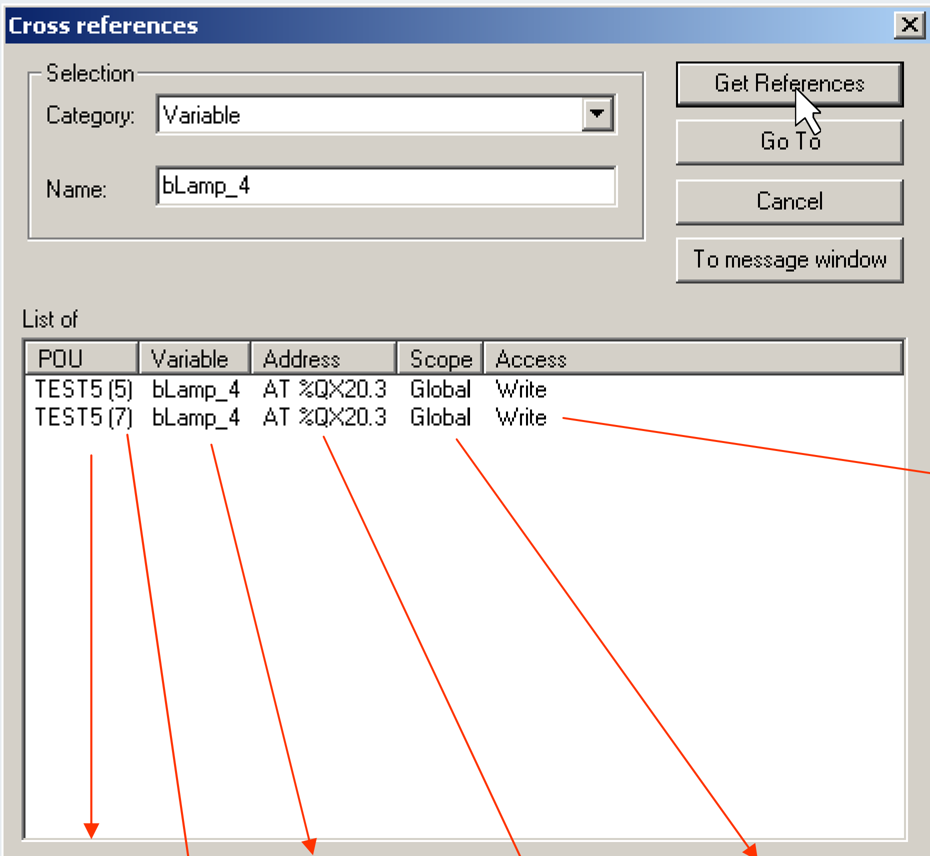
Cross-reference list 1

```
R_TRIG1(CLK:=Start , Q=> );
IF R_TRIG1.Q THEN
      bLamp_4:=FALSE;
ELSE
      bLamp_4:=TRUE;
END_IF
```

Mark variable 2x



Cross-reference list 2



Read/Write

Block name

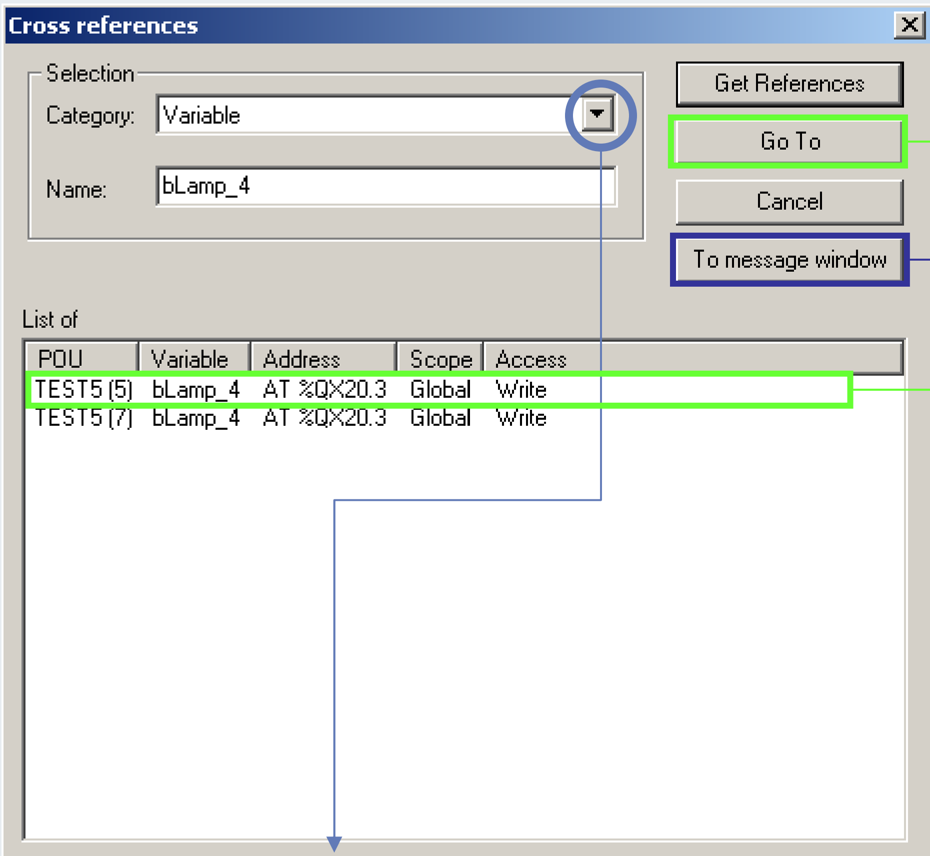
Variable name

Global/local

Network/line no.

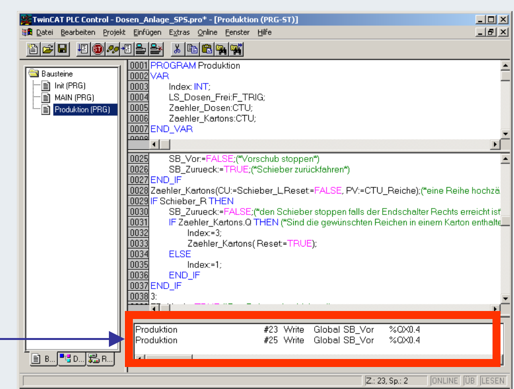
Variable address

Cross-reference list 3



Click on Line which you want to chose and press Goto

Change to selected network/line of code



Cross-references to:
Variables – bLamp_4
addresses - %QX20.3
blocks – Test5

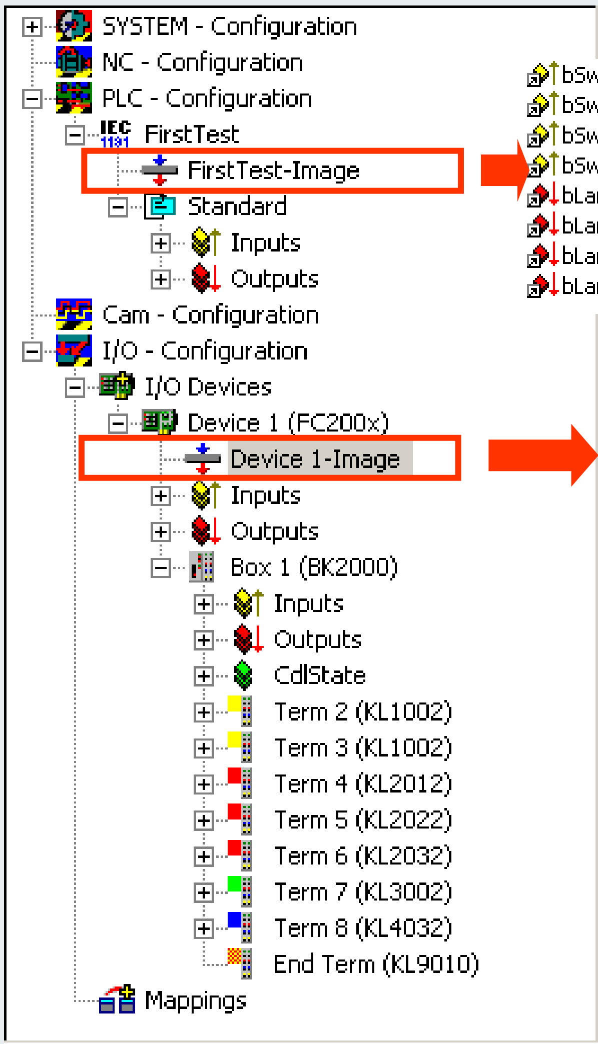
IBK – T11

Which menu item enables creation of a cross-reference list in the PLC?

What information can be derived from the cross-reference list?

What methods does the cross-reference list offer for finding the variables in the associated line of code?

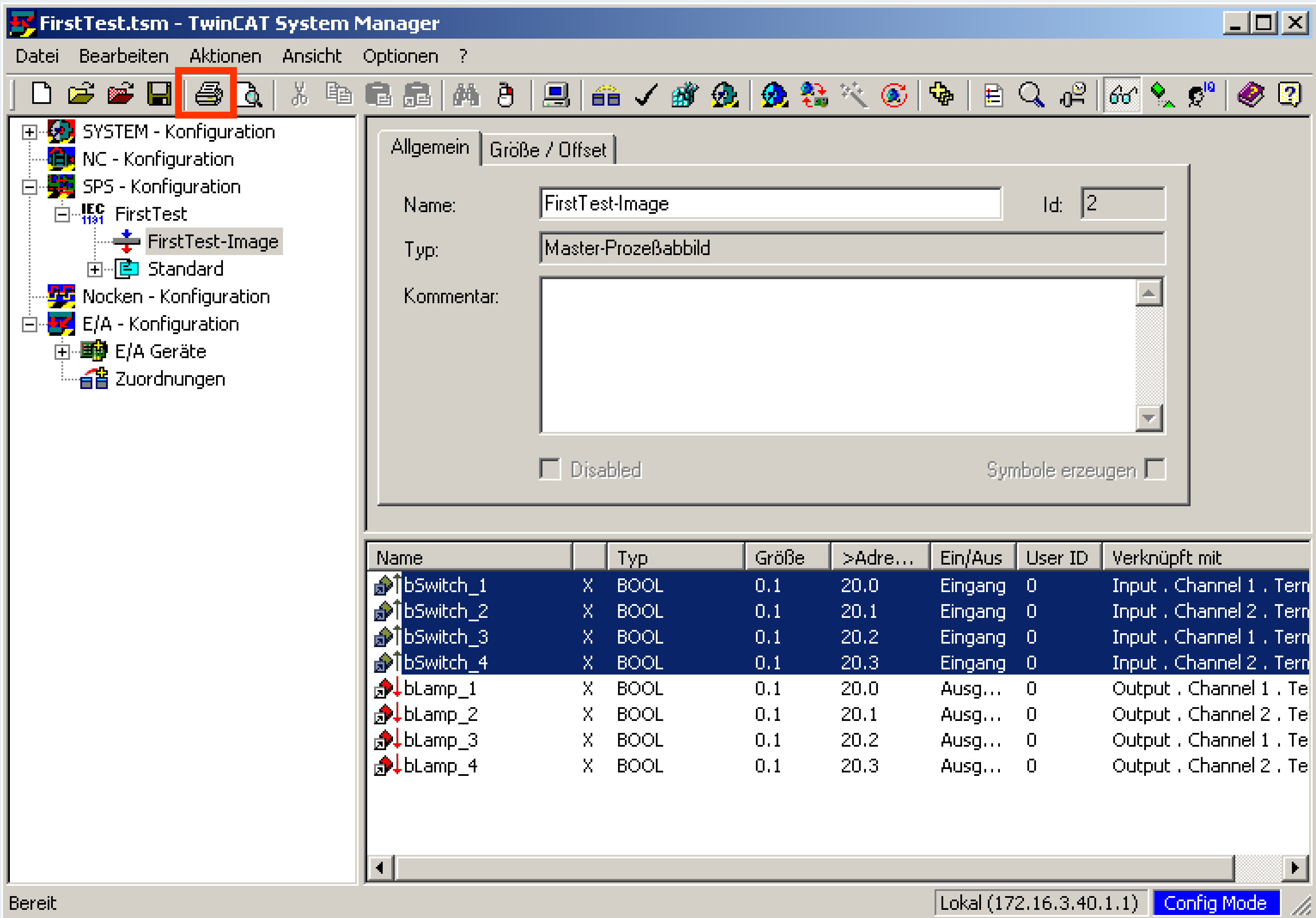
Cross-reference list in the System Manager (select)



bSwitch_1	X	0	BOOL	0.1	20.0	Input	0	Input . Channel 1 . Term 2 (KL10...
bSwitch_2	X	0	BOOL	0.1	20.1	Input	0	Input . Channel 2 . Term 2 (KL10...
bSwitch_3	X	0	BOOL	0.1	20.2	Input	0	Input . Channel 1 . Term 3 (KL10...
bSwitch_4	X	0	BOOL	0.1	20.3	Input	0	Input . Channel 2 . Term 3 (KL10...
bLamp_1	X	0	BOOL	0.1	20.0	Output	0	Output . Channel 1 . Term 4 (KL2...
bLamp_2	X	0	BOOL	0.1	20.1	Output	0	Output . Channel 2 . Term 4 (KL2...
bLamp_3	X	0	BOOL	0.1	20.2	Output	0	Output . Channel 1 . Term 5 (KL2...
bLamp_4	X	1	BOOL	0.1	20.3	Output	0	Output . Channel 2 . Term 5 (KL2...

State		USINT	1.0	8.0	Input	0	
Data In		INT	2.0	10.0	Input	0	
State		USINT	1.0	12.0	Input	0	
Data In		INT	2.0	14.0	Input	0	
Input	X	BOOL	0.1	16.0	Input	0	bSwitch_1 . Inputs . Stan...
Input	X	BOOL	0.1	16.1	Input	0	bSwitch_2 . Inputs . Stan...
Input	X	BOOL	0.1	16.2	Input	0	bSwitch_3 . Inputs . Stan...
Input	X	BOOL	0.1	16.3	Input	0	bSwitch_4 . Inputs . Stan...
Output	X	BOOL	0.1	16.0	Output	0	bLamp_1 . Outputs . Stan...
Output	X	BOOL	0.1	16.1	Output	0	bLamp_2 . Outputs . Stan...
Output	X	BOOL	0.1	16.2	Output	0	bLamp_3 . Outputs . Stan...
Output	X	BOOL	0.1	16.3	Output	0	bLamp_4 . Outputs . Stan...
Output		BOOL	0.1	16.4	Output	0	
Output		BOOL	0.1	16.5	Output	0	

Cross-reference list in the System Manager (print)



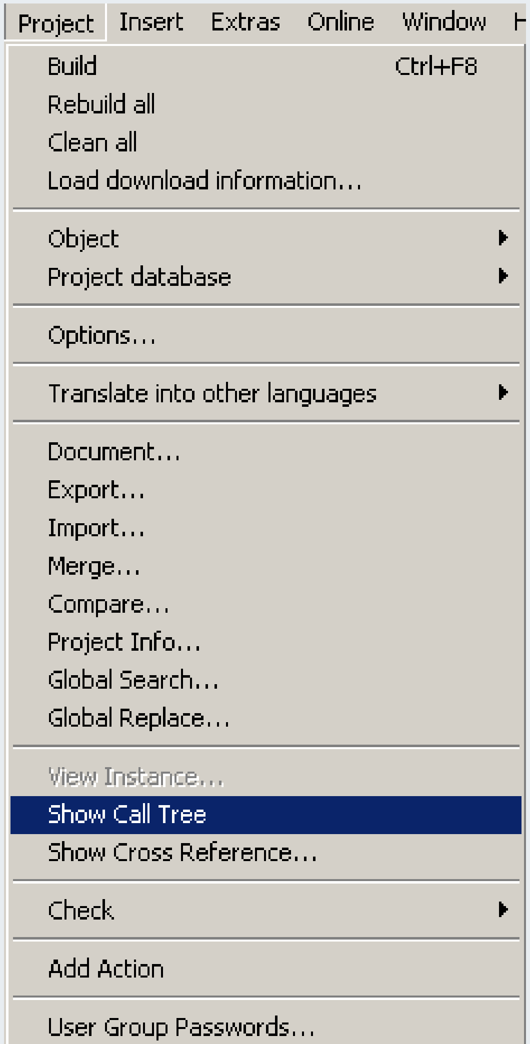
IBK – T12

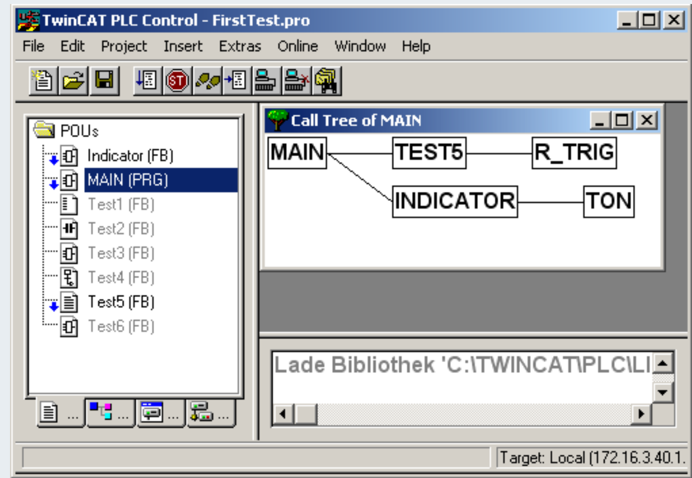
Where in the System Manager can the cross-reference list for the variables be displayed?

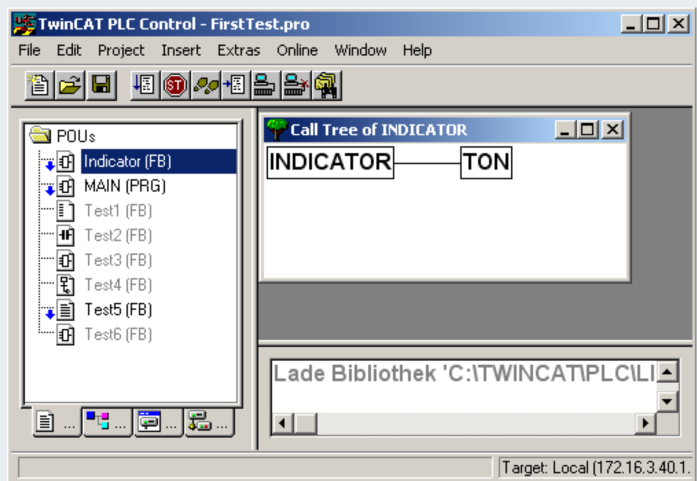
Where in the System Manager can the terminals assigned to the station be printed?

Where in the System Manager is the setting for instructing the terminals assigned to the station to display the subvariables?

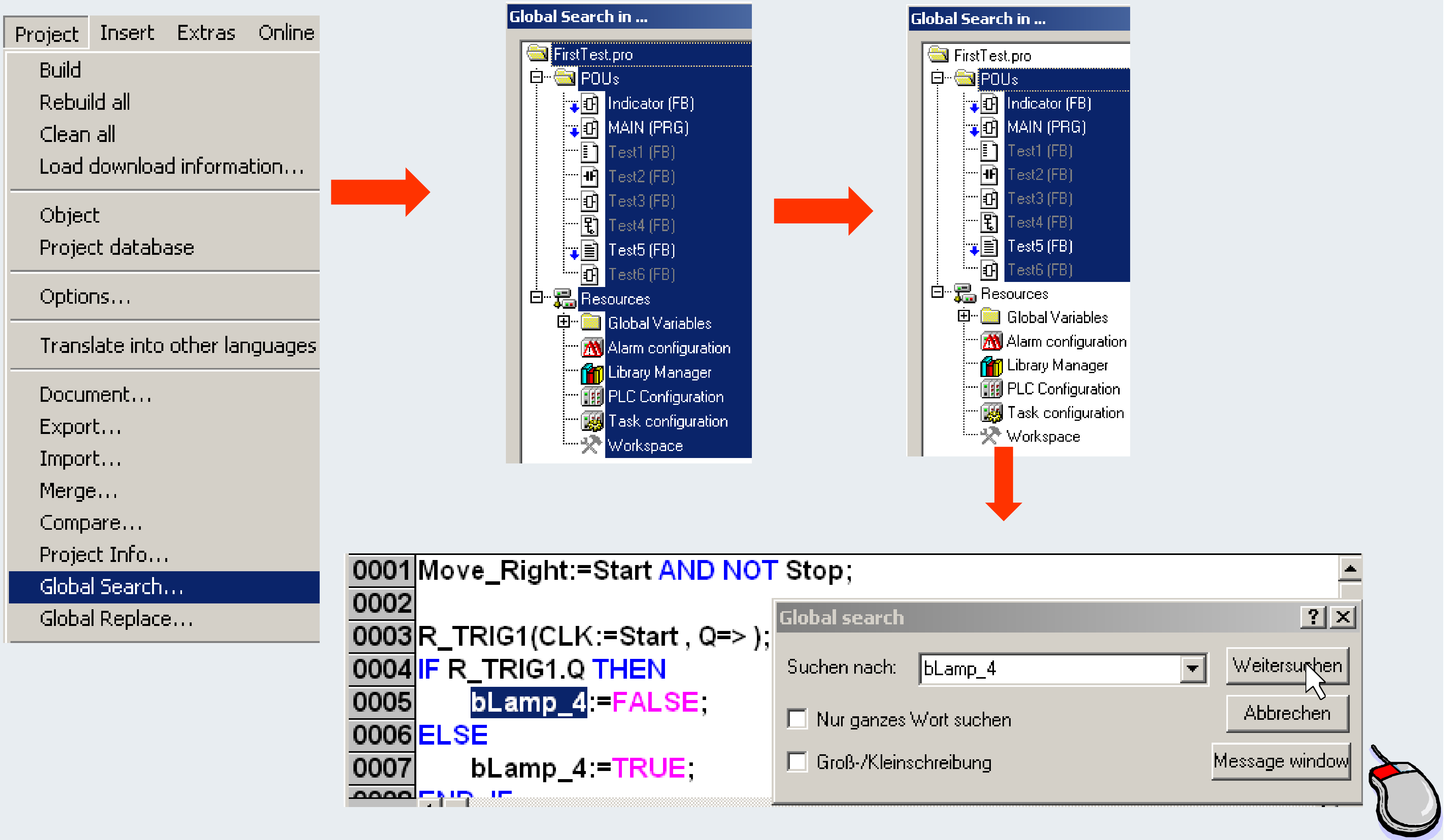
Show Call Tree







Global search and replace



IBK – T13

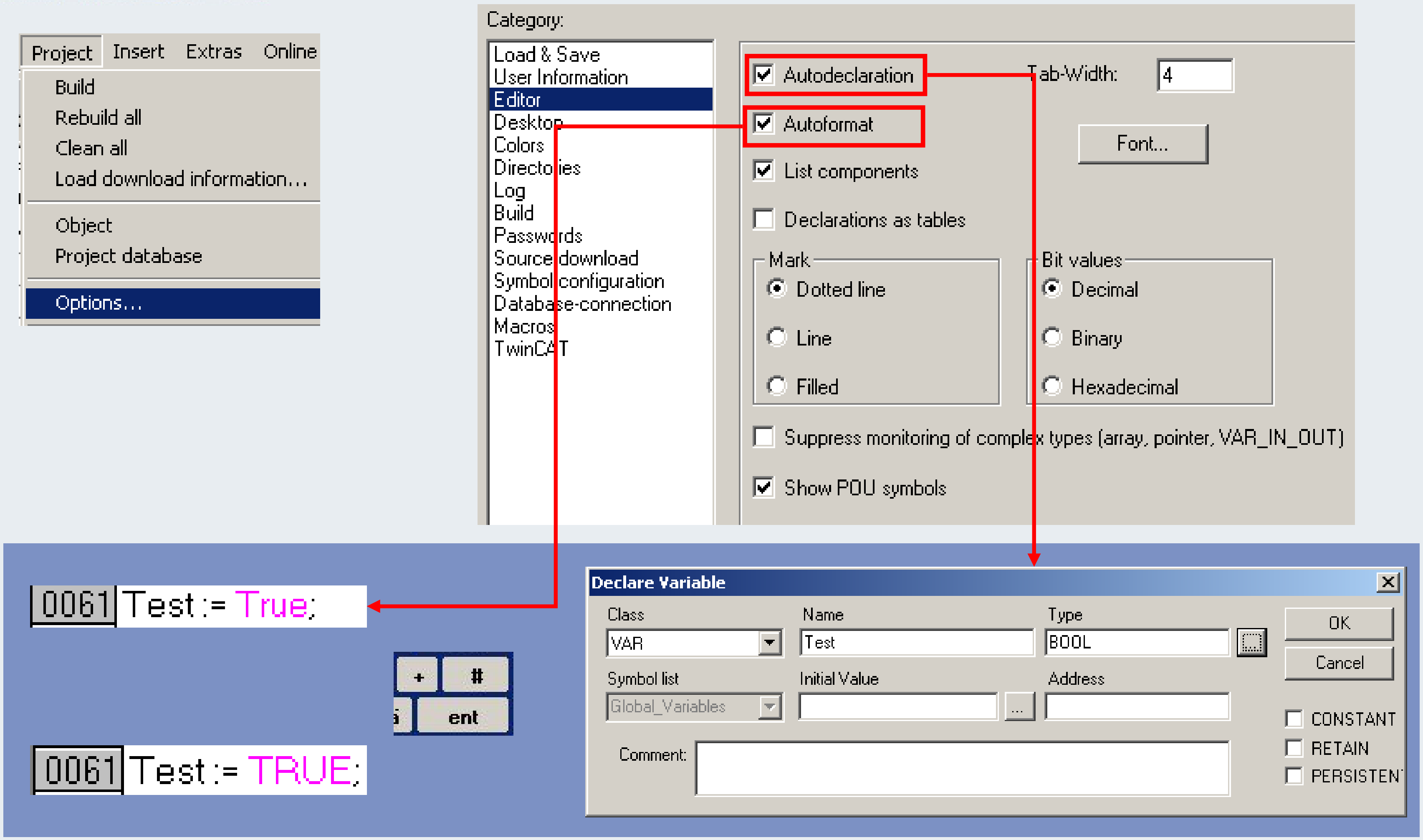
What is the name of the first block installed by default in the PLC?

How can the block processing sequence be displayed in the PLC program?

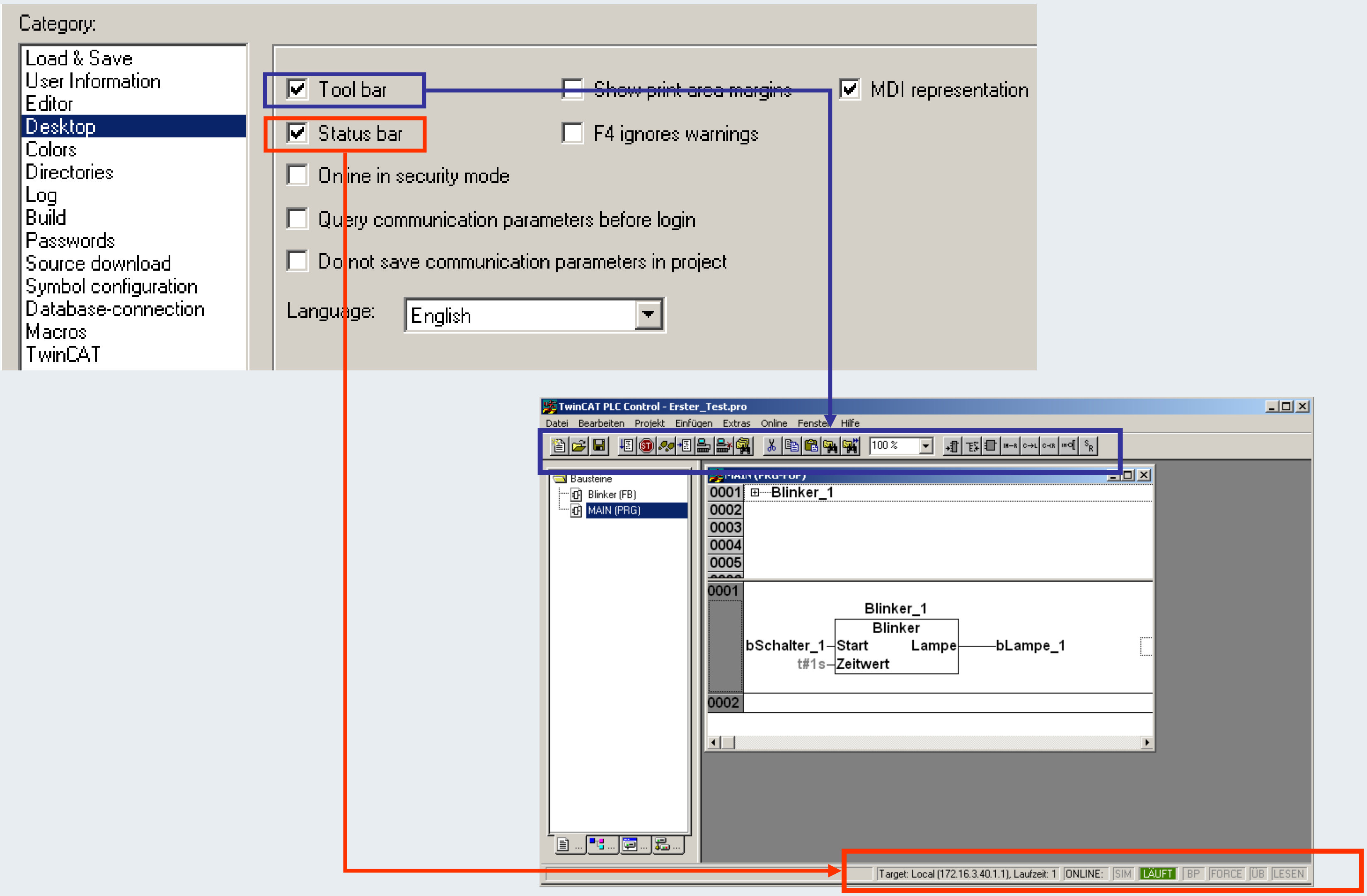
Where in the PLC can the blocks created be called up cyclically?

What software tool can be used for rewiring the PLC variables?

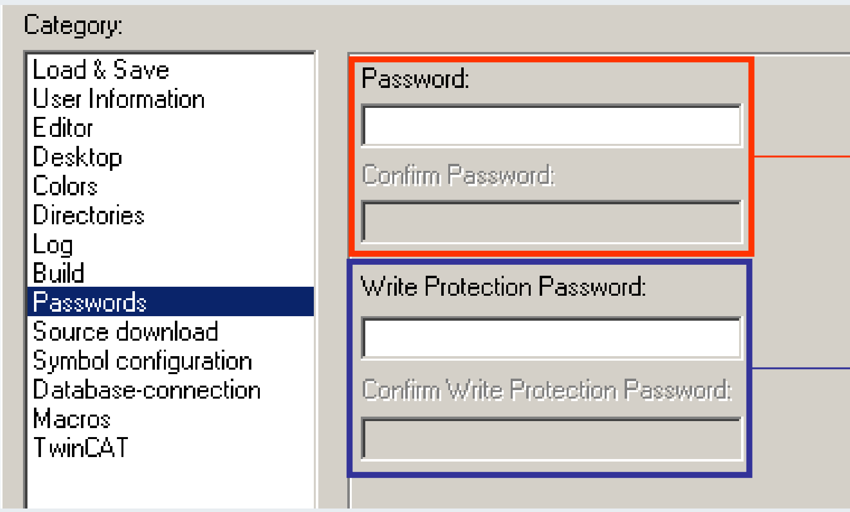
PLC Control options



Options (working area)



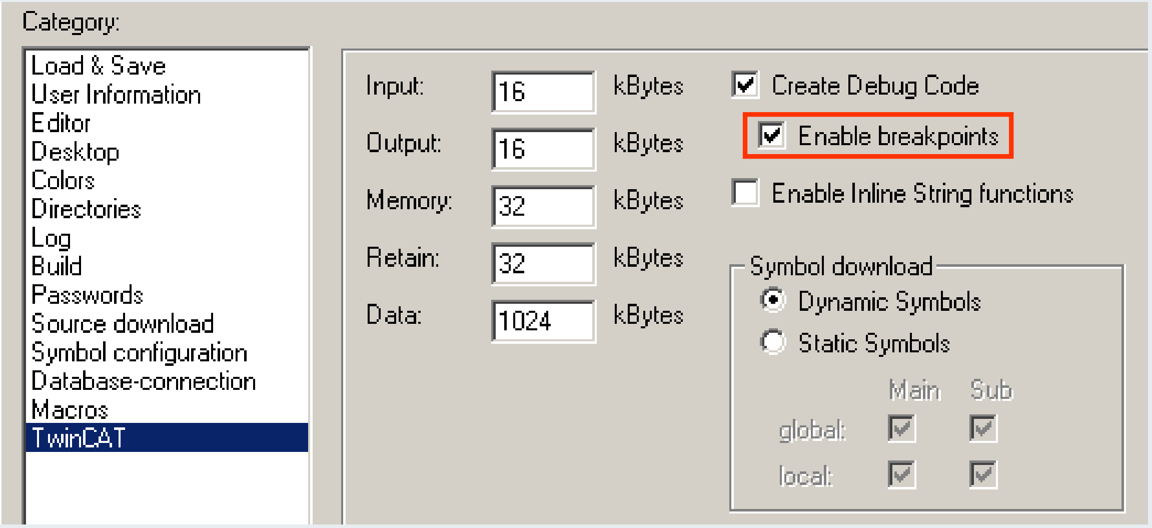
Options (password)



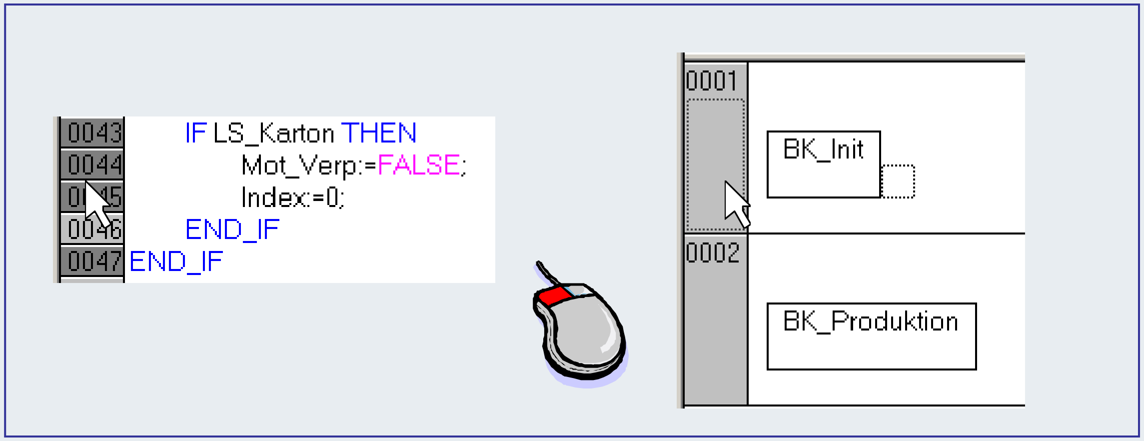
Absolute password (reading and writing)

View enabled; the program can only be changed with a password

Options (TwinCAT)



Break points can be set by clicking on the network field or a row number



IBK – T14

Which menu item can be used for project-specific setting of options in the PLC?

Which options category offers deactivation of automatic variable declaration?

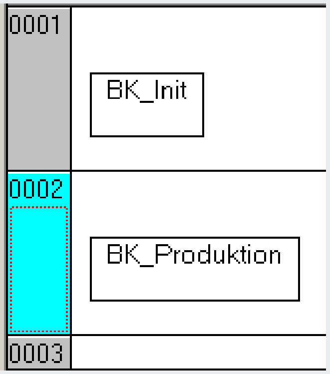
What effect does the „Automatic formatting“ options category have?

The PLC options can be used for allocating passwords.
Which passwords can be allocated and what effect do they have?

Break points

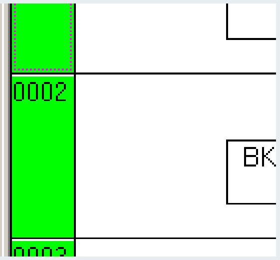
Setting a break point

```
0043      IF LS_Karton THEN
0044          Mot_Verp:=FALSE;
0045          Index:=0;
0046      END_IF
0047  END_IF
```



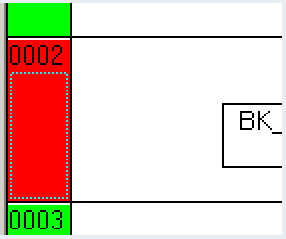
Program flow

```
0043      IF LS_Karton THEN
0044          Mot_Verp:=FALSE;
0045          Index:=0;
0046      END_IF
0047  END_IF
```



If the program overruns a break point, the colour changes to red, and program execution is stopped

```
0043      IF LS_Karton THEN
0044          Mot_Verp:=FALSE;
0045          Index:=0;
0046      END_IF
0047  END_IF
```



Break points

Data exchange is interrupted, since in most bus systems data exchange is requested by the PLC.

As a result, the slave module watchdog is deactivated after 100 ms, and the outputs are reset.

Lifting axes are held by the brakes.

Pneumatic valves drop out

IBK – T15

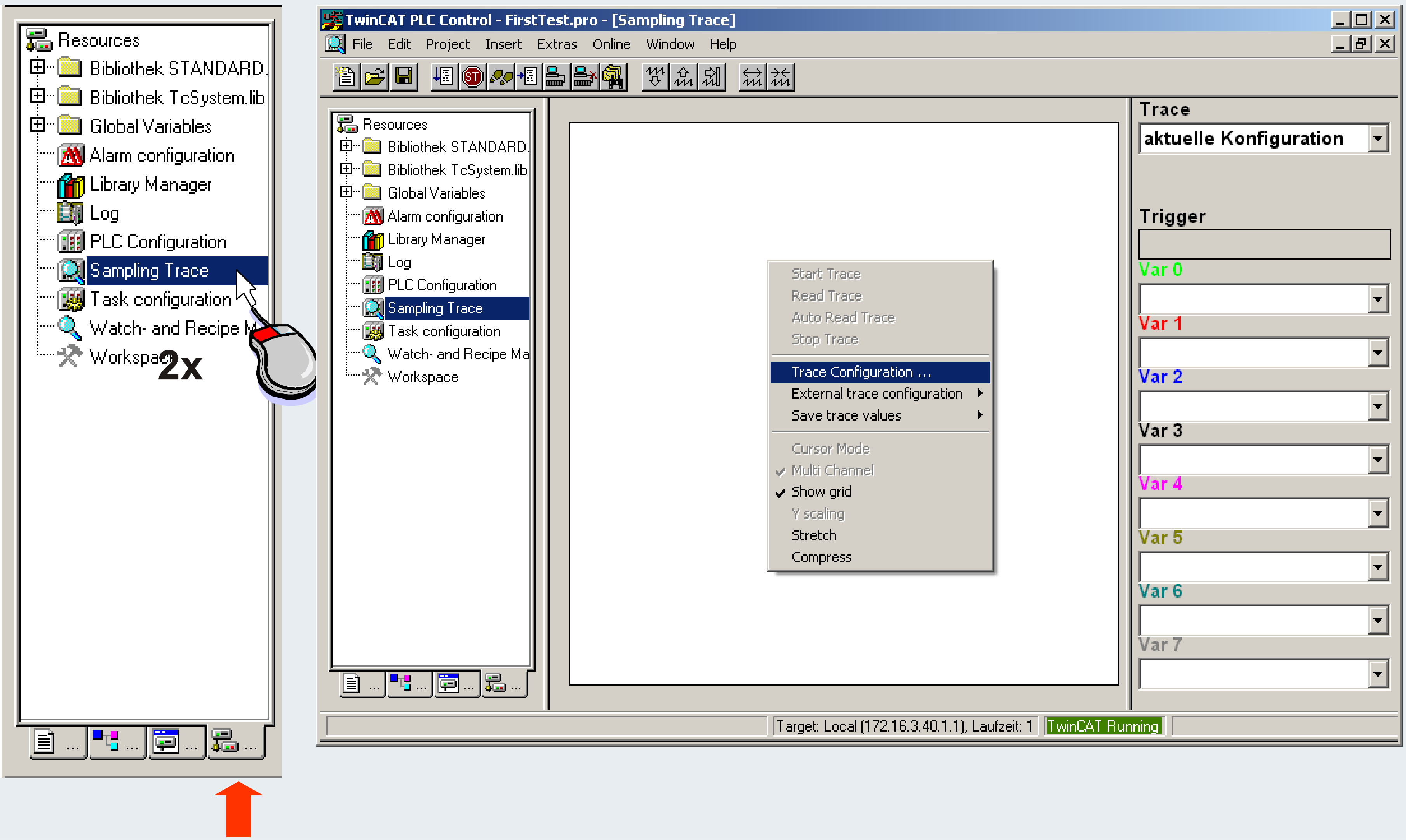
What effect does setting of a break point in the PLC have?

How is a set break point indicated?

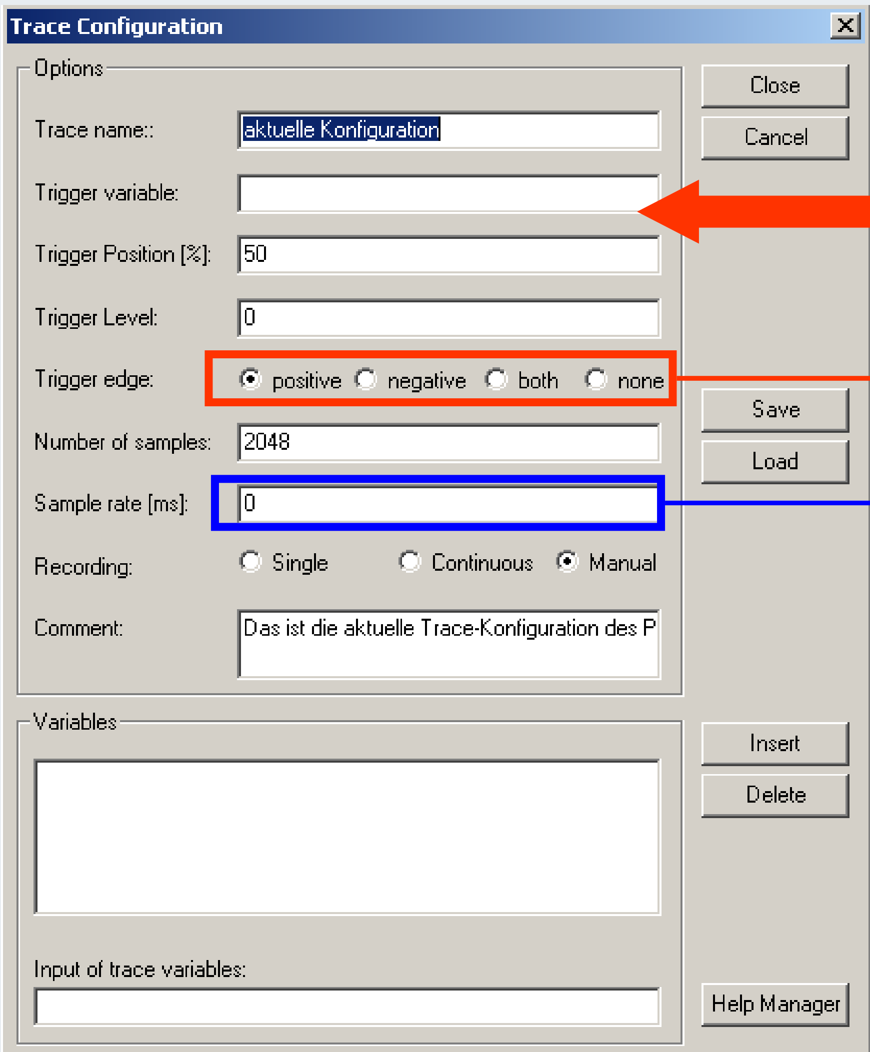
What happens if a break point is overriden?

Where can break point setting be switched off?

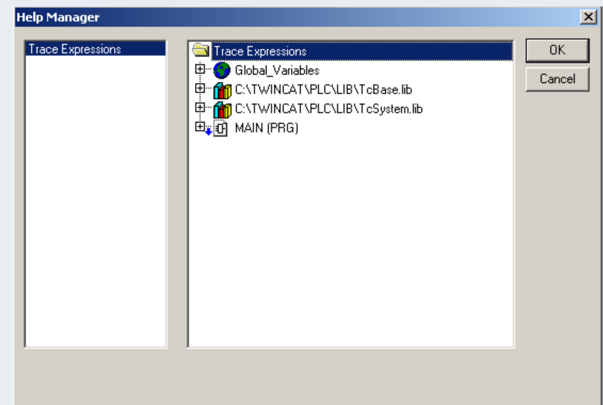
Trace selection



Trace configuration 1

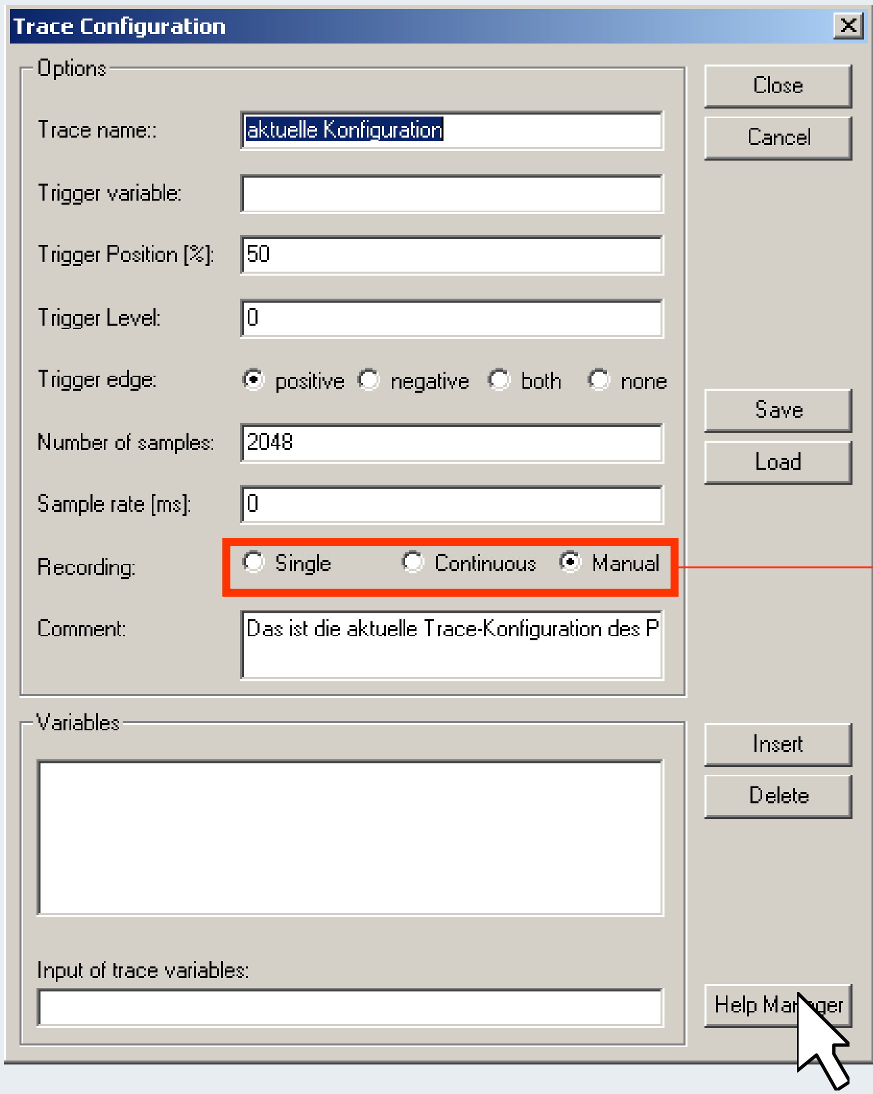


+



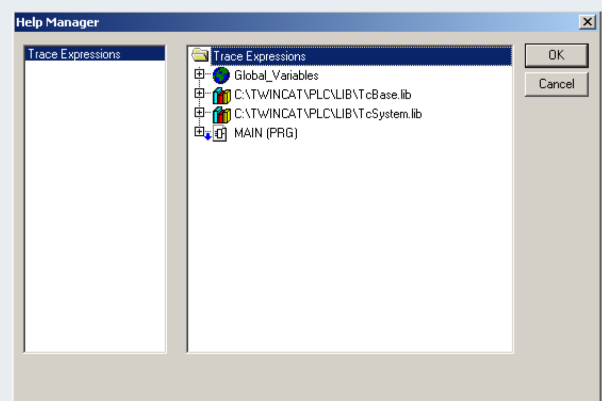
Zero corresponds to the cycle time, otherwise xx ms

Trace configuration 2



As soon as the trigger is active, an *individual* recording (or consecutive recording with each trigger) is started.

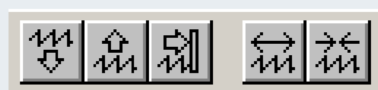
Insert Trace Variables



Trace configuration 3

The selected trigger variable is displayed here

The list of available variables is displayed here



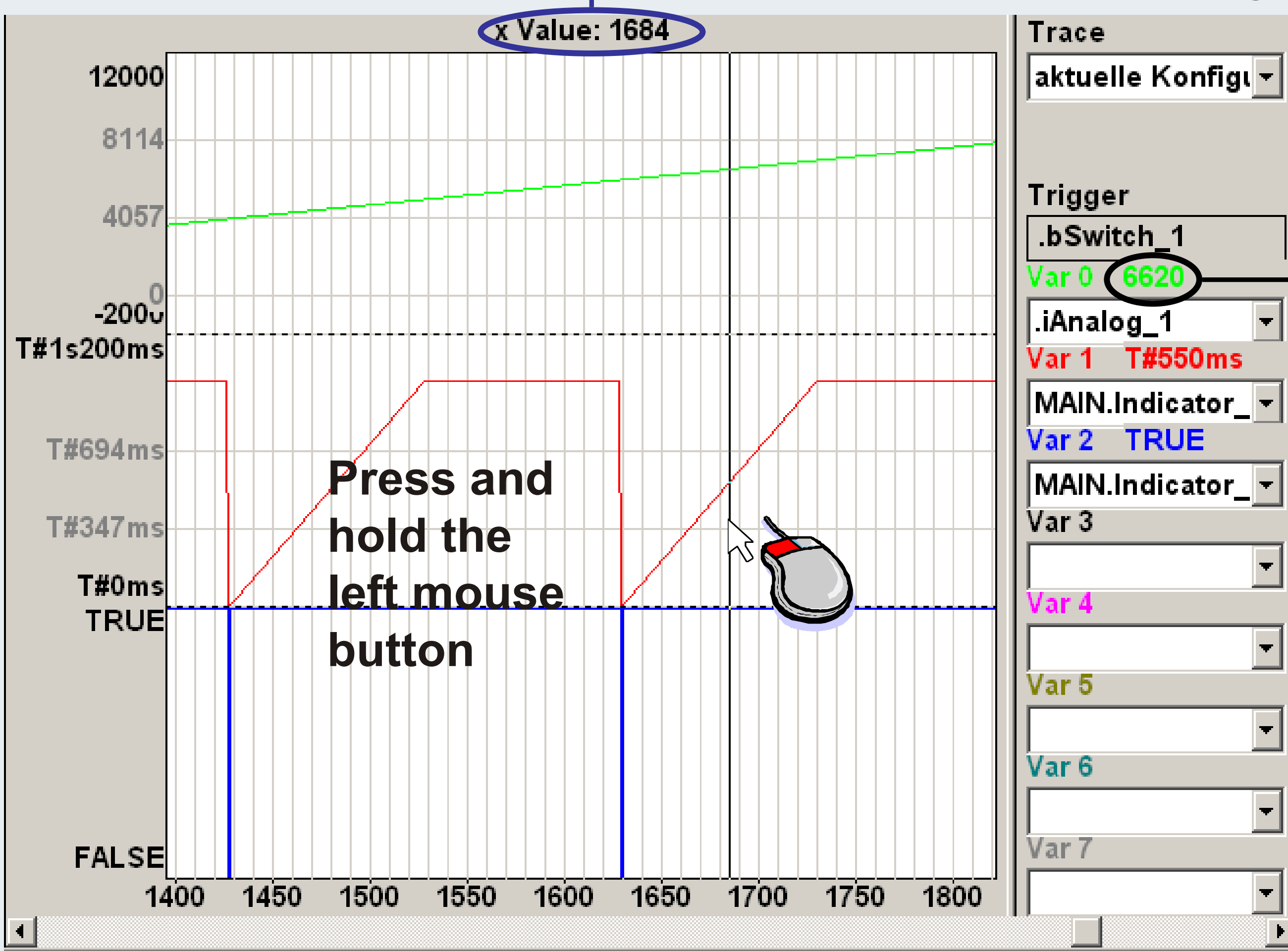
Start Read Stop Zoom

Cursor

Cycle number

Automatic scaling

Automatic colour selection



Variable value at the intersection with the cursor

Sampling Trace

Requirements:

- PLC Control should be online with the required project.
- The project is in „running“ mode.

Run Time: 1 | ONLINE: | SIM | RUN

Trace recording started. Waiting for trigger event

Status: Wait for trigger

The trigger event has occurred; trace recording running

Status: Trace record running, trigger reach

Once the selected number of recordings is reached (1-2048), recording is complete

Status: Trace record finished

IBK – T16

Which PLC tool offers graphic curve recording?

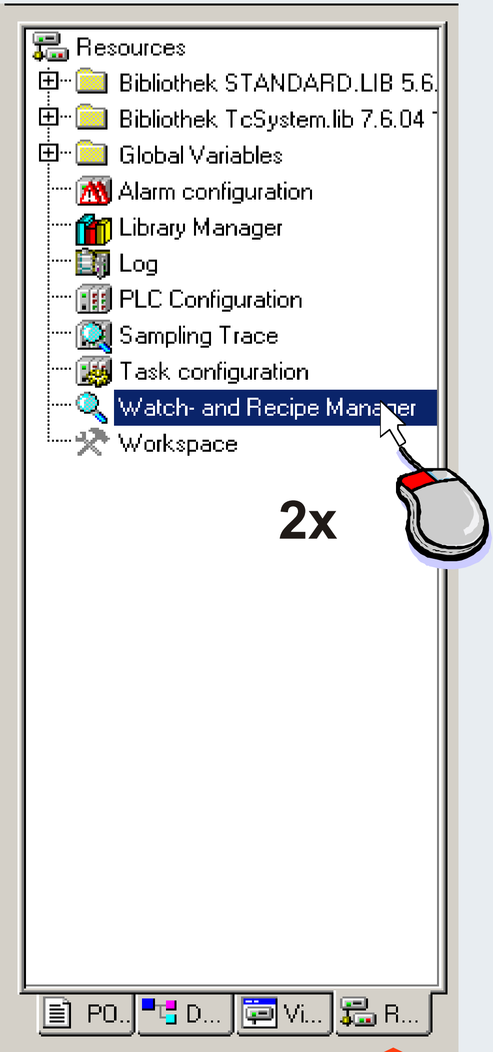
Where in the PLC is the recording tool located?

At what flanks of the trigger variable can recording be started?

In which table are the variables to be recorded created?

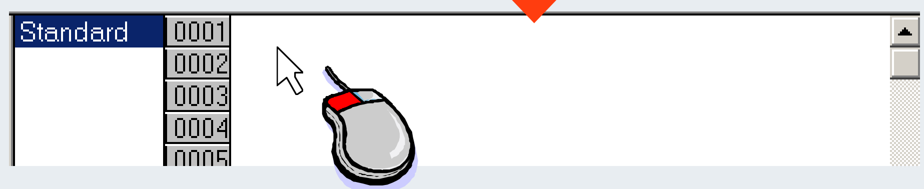
What precondition has to be met so that recording can be started?

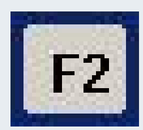
Watch window 1



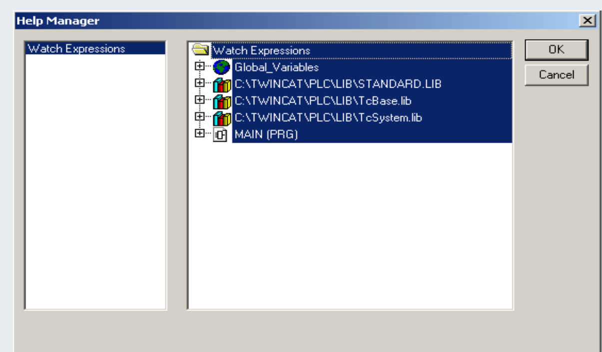
Precondition for entering variables in the required list:

- **PLC Control should be offline with the required project.**

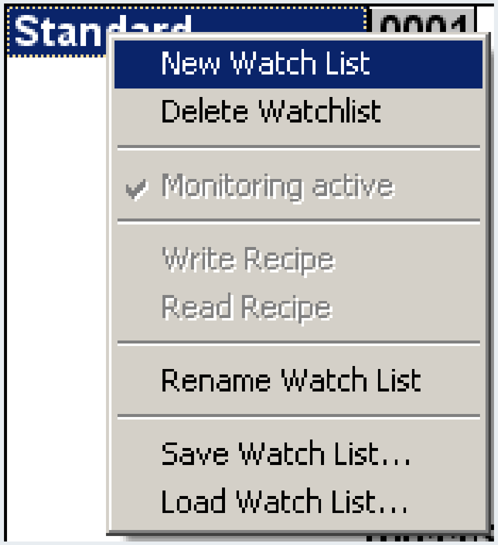




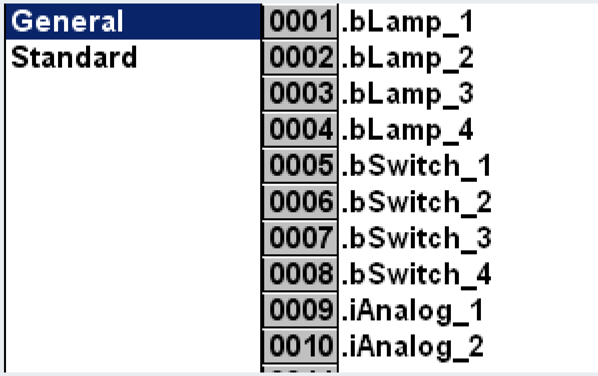
+



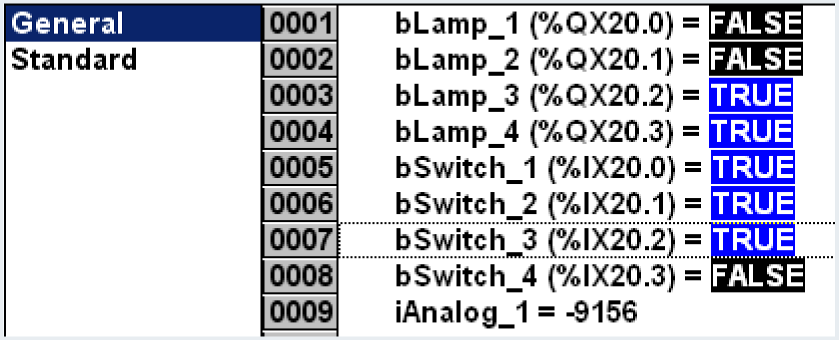
Watch window 2



Several lists can be created



Subsequent switching between lists is possible



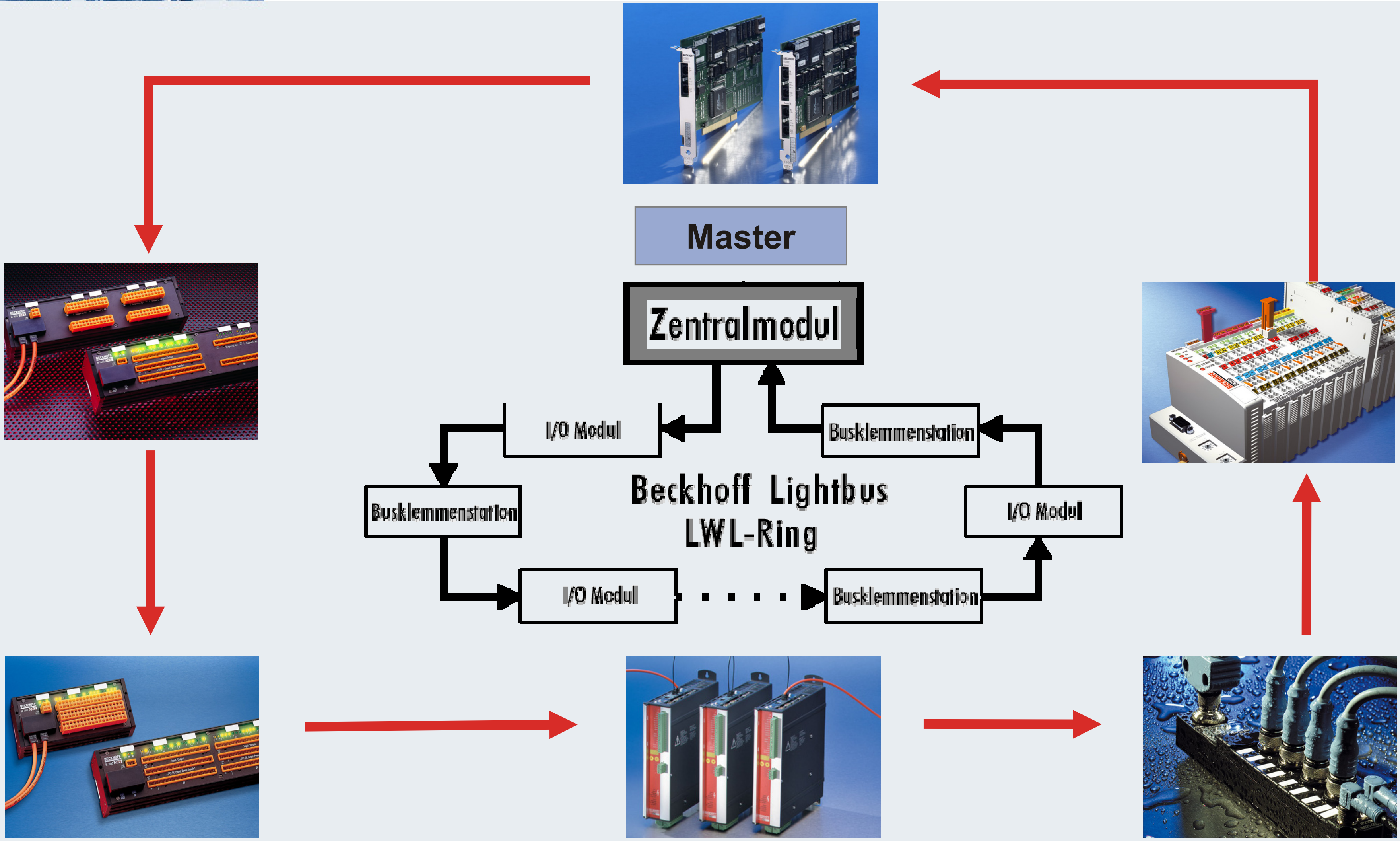
IBK – T17

Which tool can be created in the PLC for displaying the state of selected variables in the event of an error?

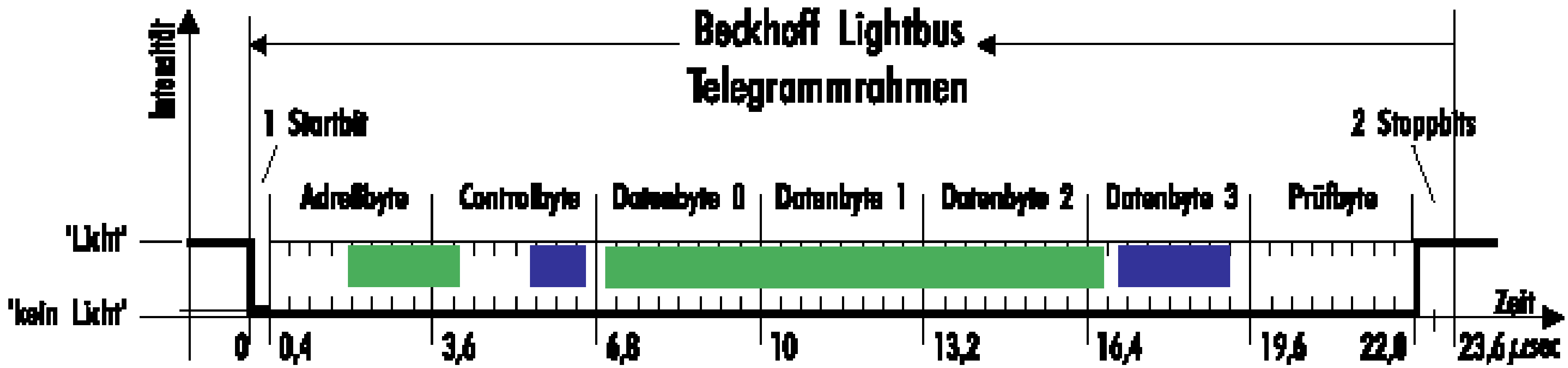
Which PLC window can be used for saving the „status list“?

Which function key offers access to the Input Assistant for entering variables in the „status list“?

II/O Lightbus



Lightbus telegram



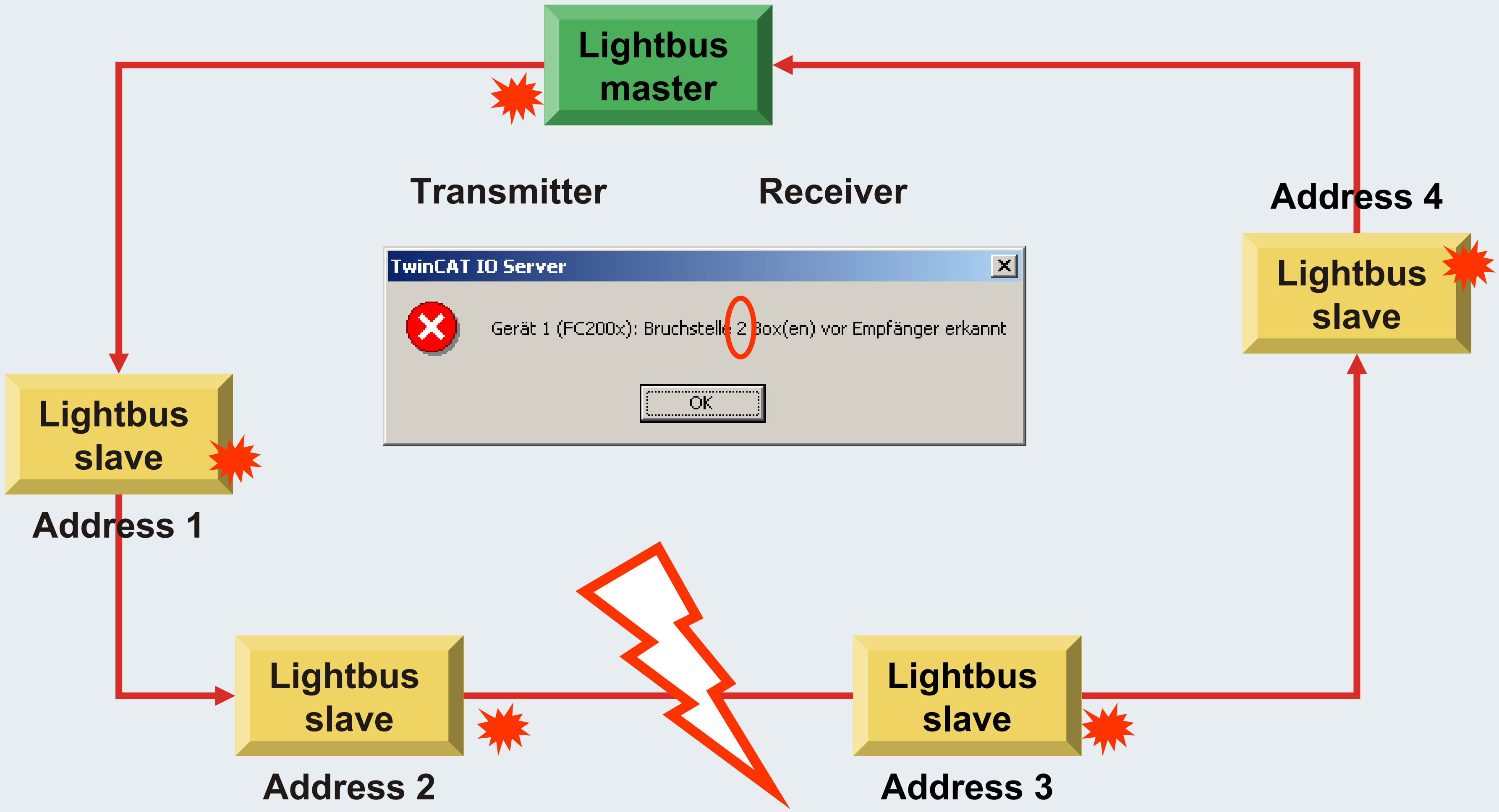
Address of slave assembly1 - 254

User data

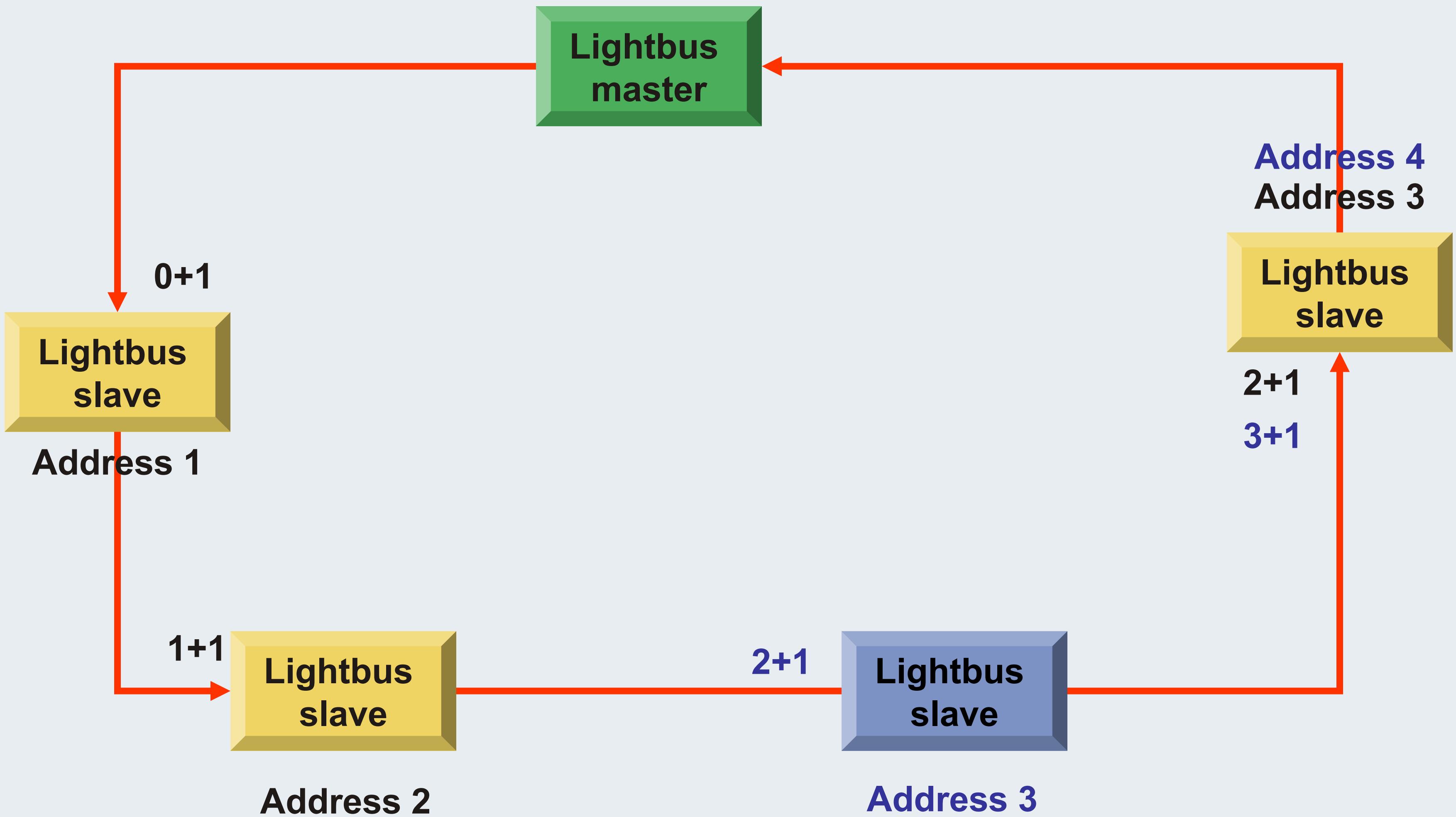
Telegram type
0 - Read
1 - Read/Write
2 - Address initialisation
...
9 - Attenuation test

6-bit CRC checksum

Break test


Lightbus
master
Transmitter
Receiver
Address 4
Lightbus
slave
TwinCAT IO Server
Gerät 1 (FC200x): Bruchstelle 2 Box(en) vor Empfänger erkannt
OK
Lightbus
slave
Address 1
Lightbus
slave
Address 2
Lightbus
slave
Address 3

Addressing of Lightbus slaves

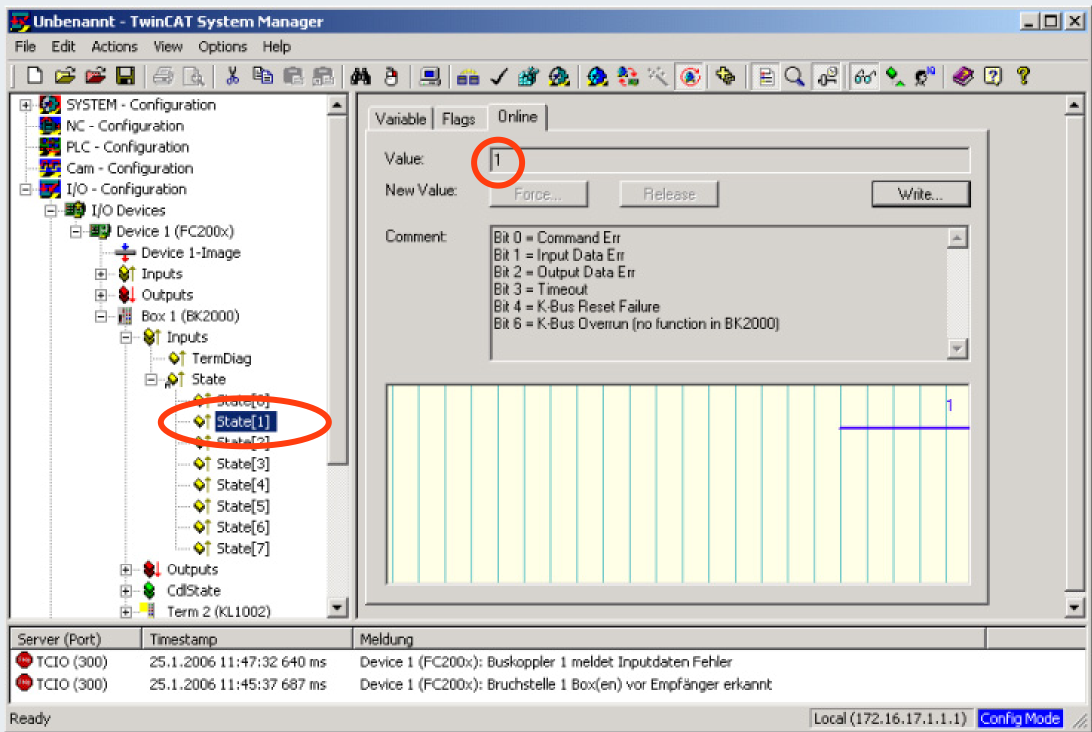


Diagnostics slave

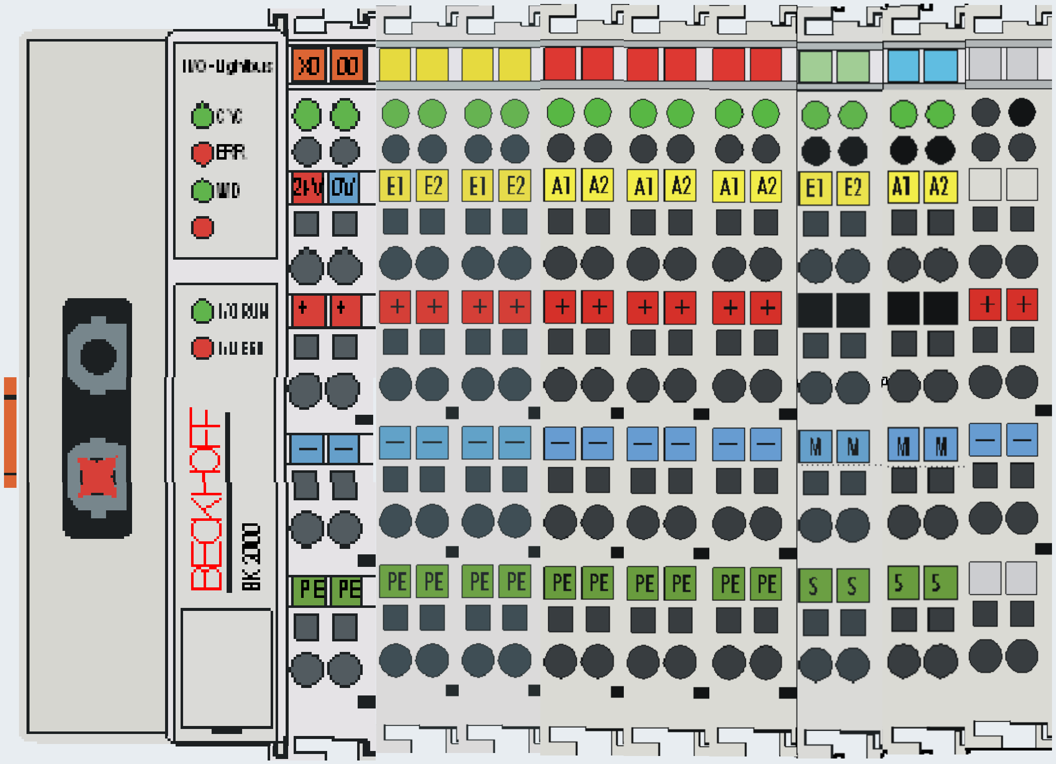
Coupler diagnostics:

- If an interruption occurs at the K-Bus, the status input of the coupler reports an input (0x2) or output data error (0x4).

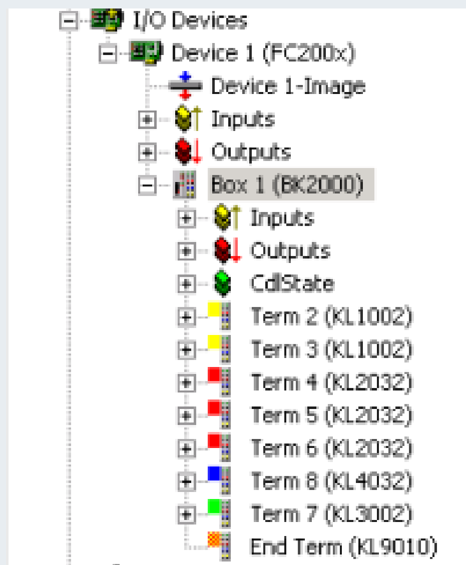
Bit 0 = Command Err
Bit 1 = Input Data Err
Bit 2 = Output Data Err
Bit 3 = Timeout
Bit 4 = K-Bus Reset Failure



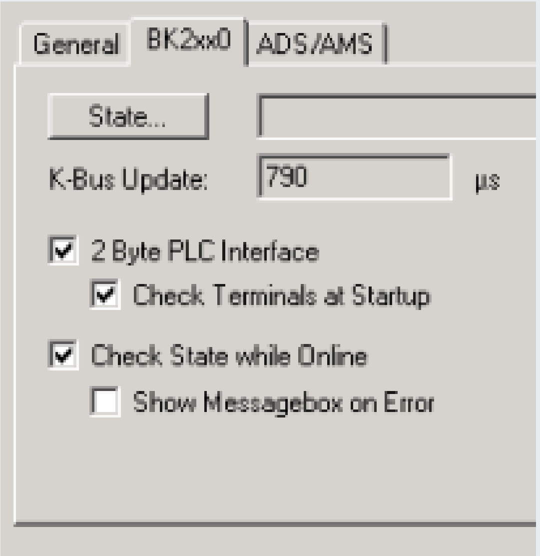
Bus Terminal configuration error



The position of terminal 6 does not correspond to the physical position at the coupler. This error generates an incorrect process image in the System Manager.

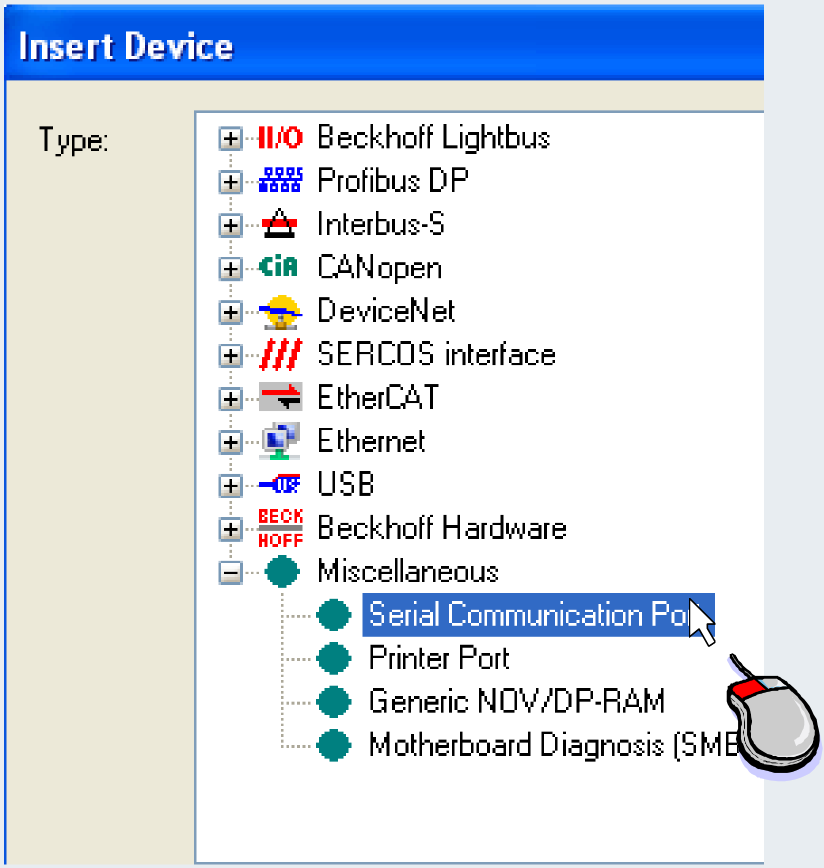


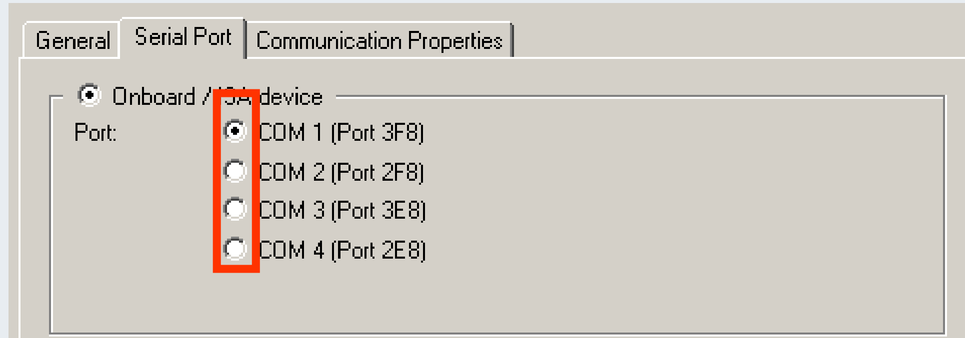
Remedy

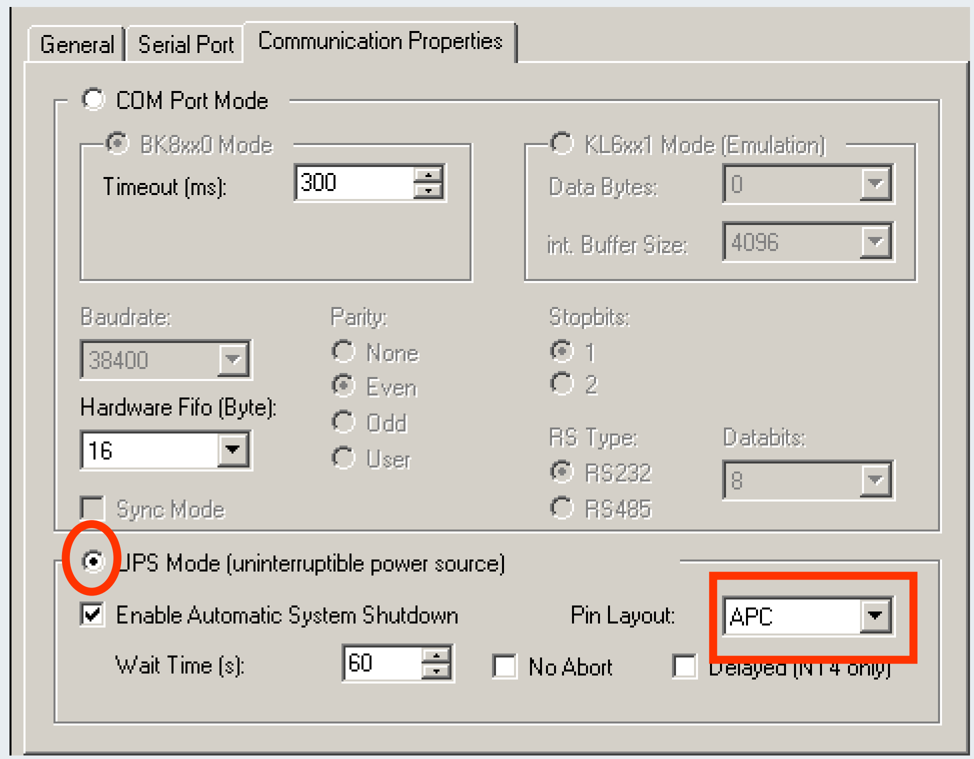


If both settings are made at the BK2000, the System Manager the compares the specified configuration with the coupler data. This enables process image size errors to be detected. The sequence is also checked.
Reversal of the terminals from the preceding example generates the following error message.

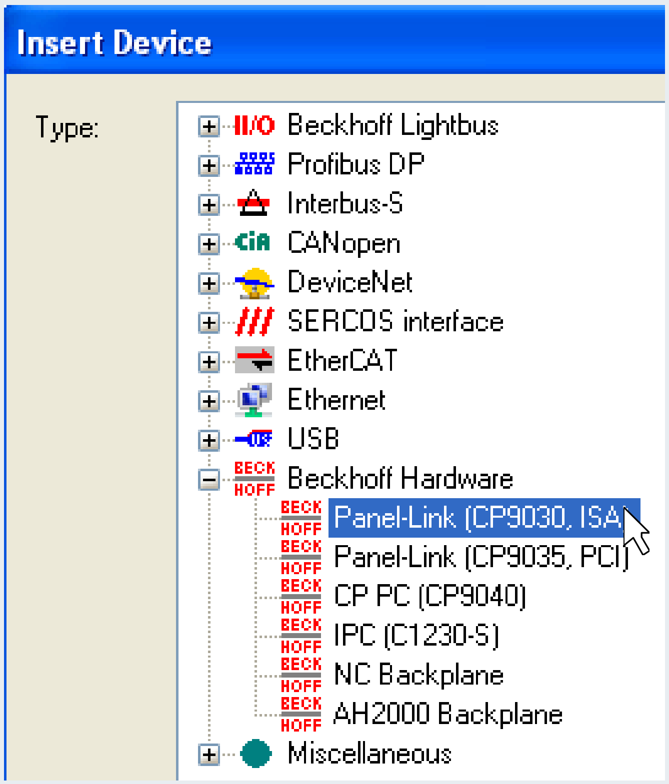
Hardware 230 V UPS

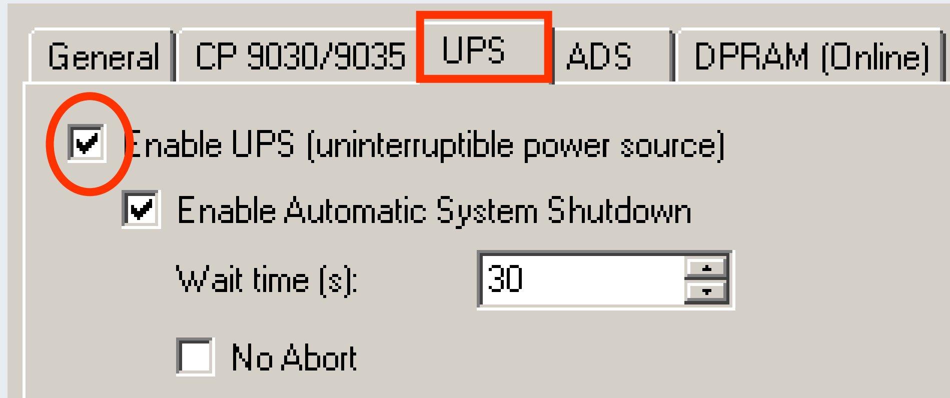


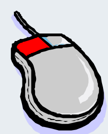




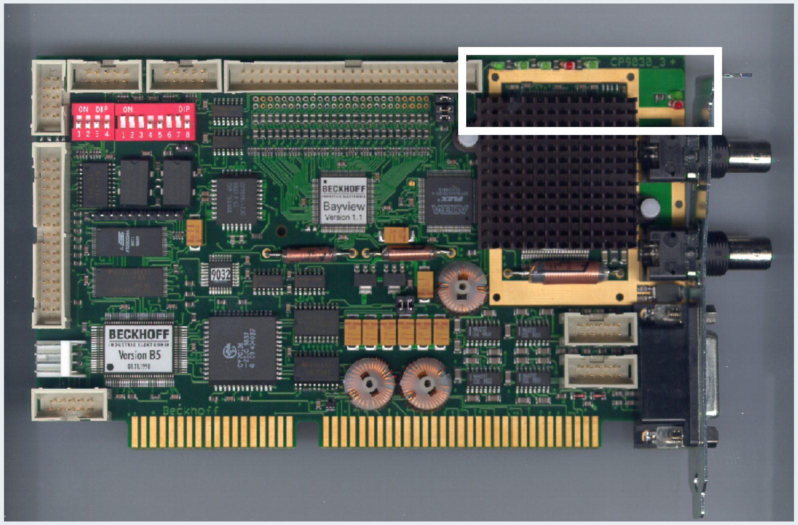
24-V-UPS

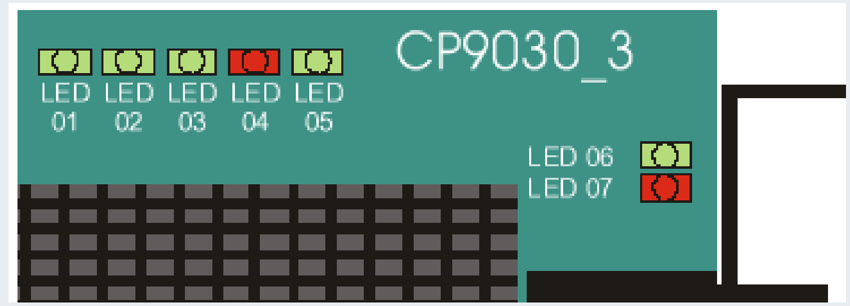

Insert Device
Type:
Beckhoff Lightbus
Profibus DP
Interbus-S
CANopen
DeviceNet
SERCOS interface
EtherCAT
Ethernet
USB
Beckhoff Hardware
Panel-Link (CP9030, ISA)
Panel-Link (CP9035, PCI)
CP PC (CP9040)
IPC (C1230-S)
NC Backplane
AH2000 Backplane
Miscellaneous


General
CP 9030/9035
UPS
ADS
DPRAM (Online)
Enable UPS (uninterruptible power source)
Enable Automatic System Shutdown
Wait time (s):
30
No Abort



CP9030 1





LED 01 - 12 V Supply voltage present (B channel). A short circuit may be present if the LED is off.

LED 02 - Transmitter PLL locked. If it is not on the video card does not work.

LED 03 - Receiver PLL engaged (A channel faulty).

LED 04 - Data error in the receiver; no connection to the Control Panel.

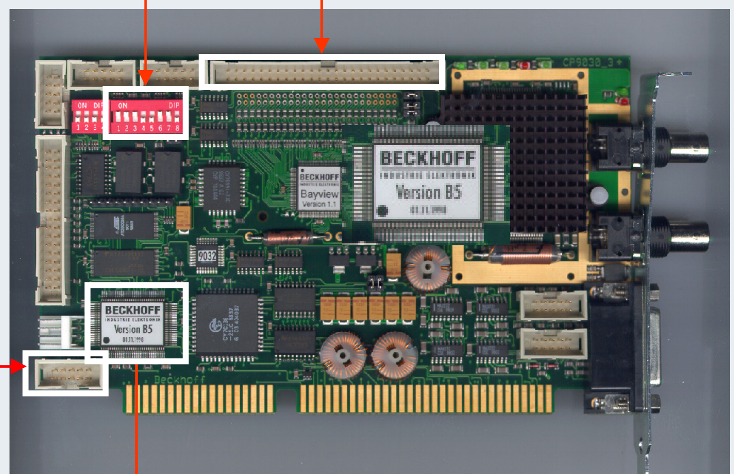
LED 06 - CP Link RUN, communication running.

LED 07 - CP-Link COMM-ERR. A continuously flashing LED either indicates a damaged coaxial cable or excessive interference from other devices.

CP9030 2

Address selection

Display connection (graphics card)

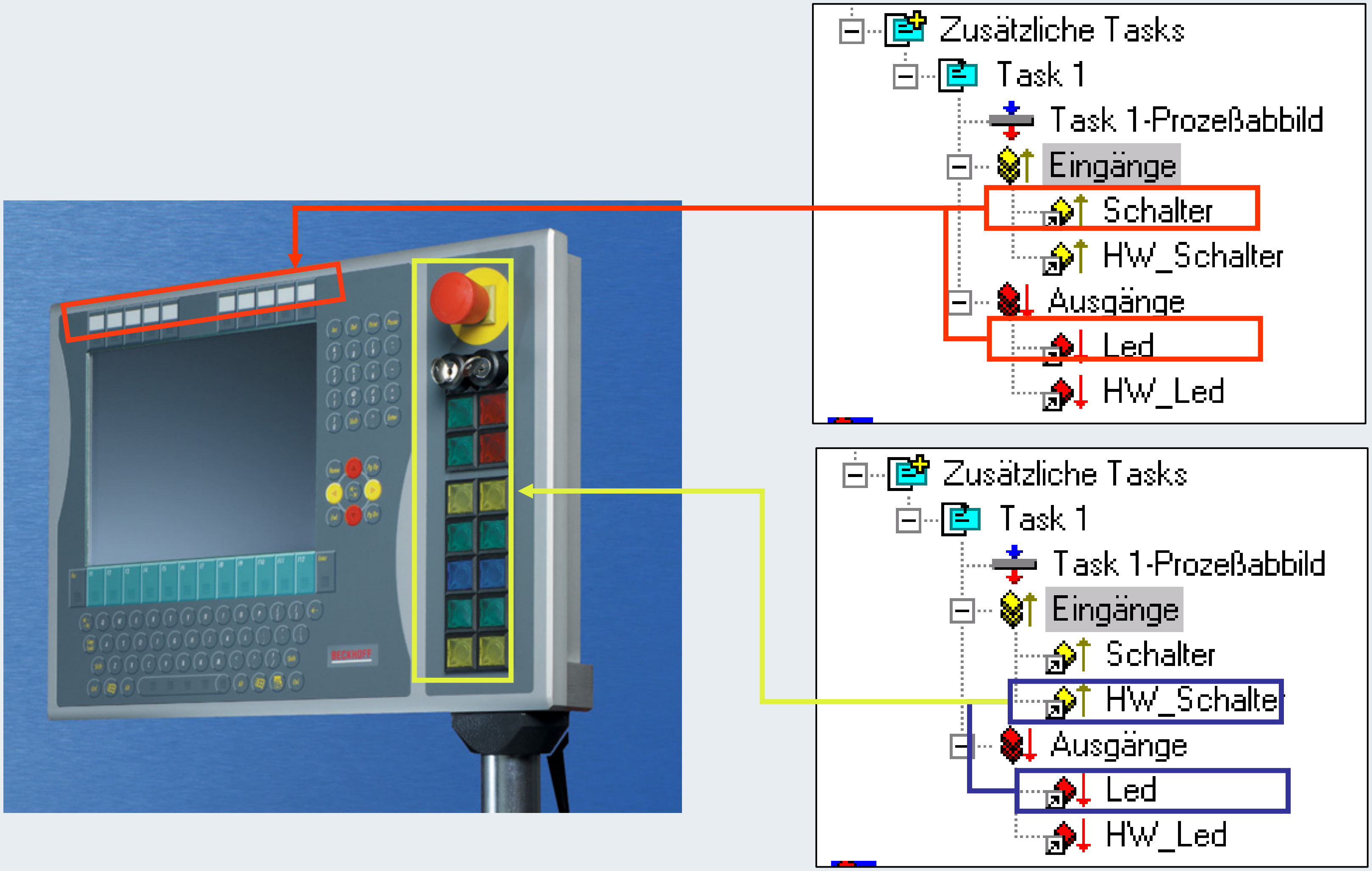


Channel A

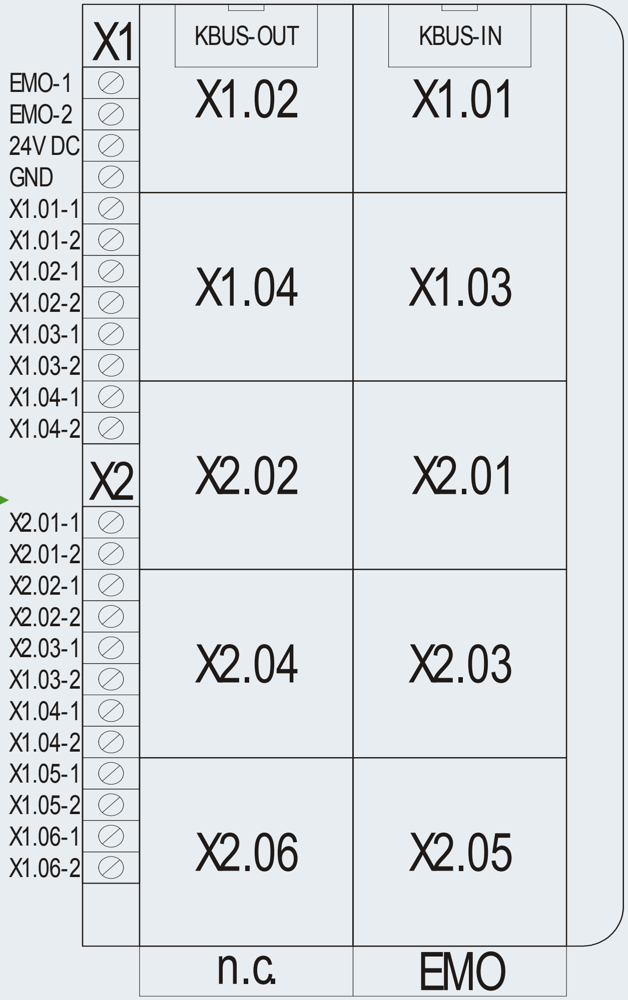
Channel B

24 V UPS control

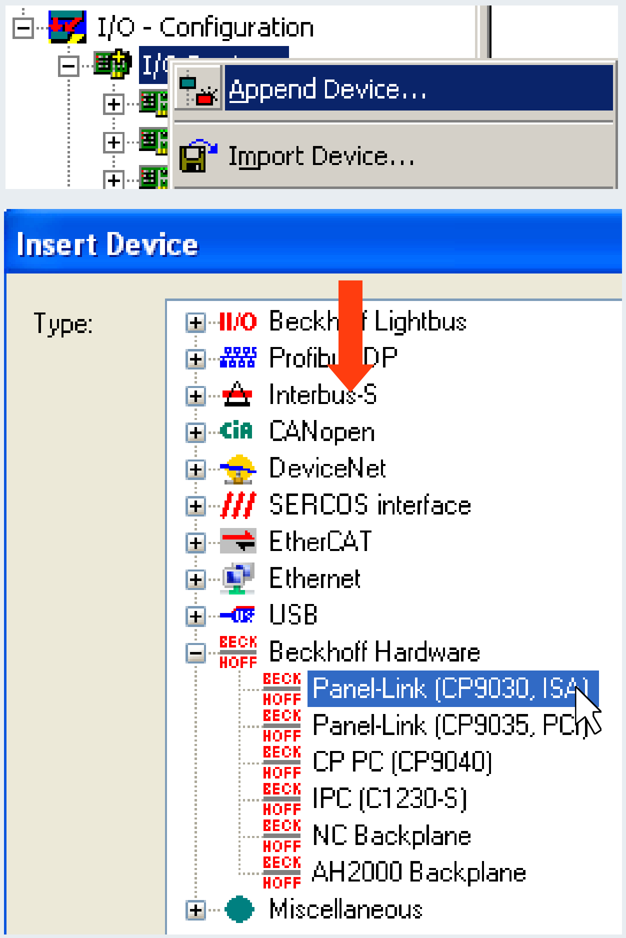
CP7031, additional tasks

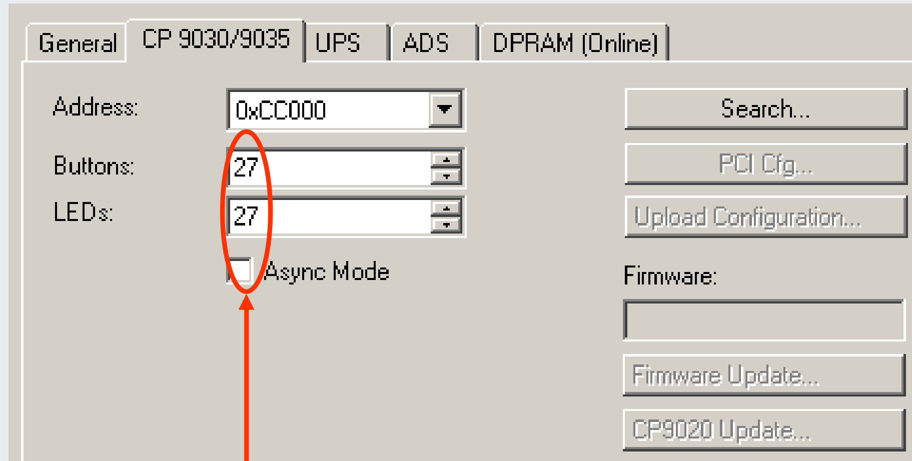


CP7031 button connection



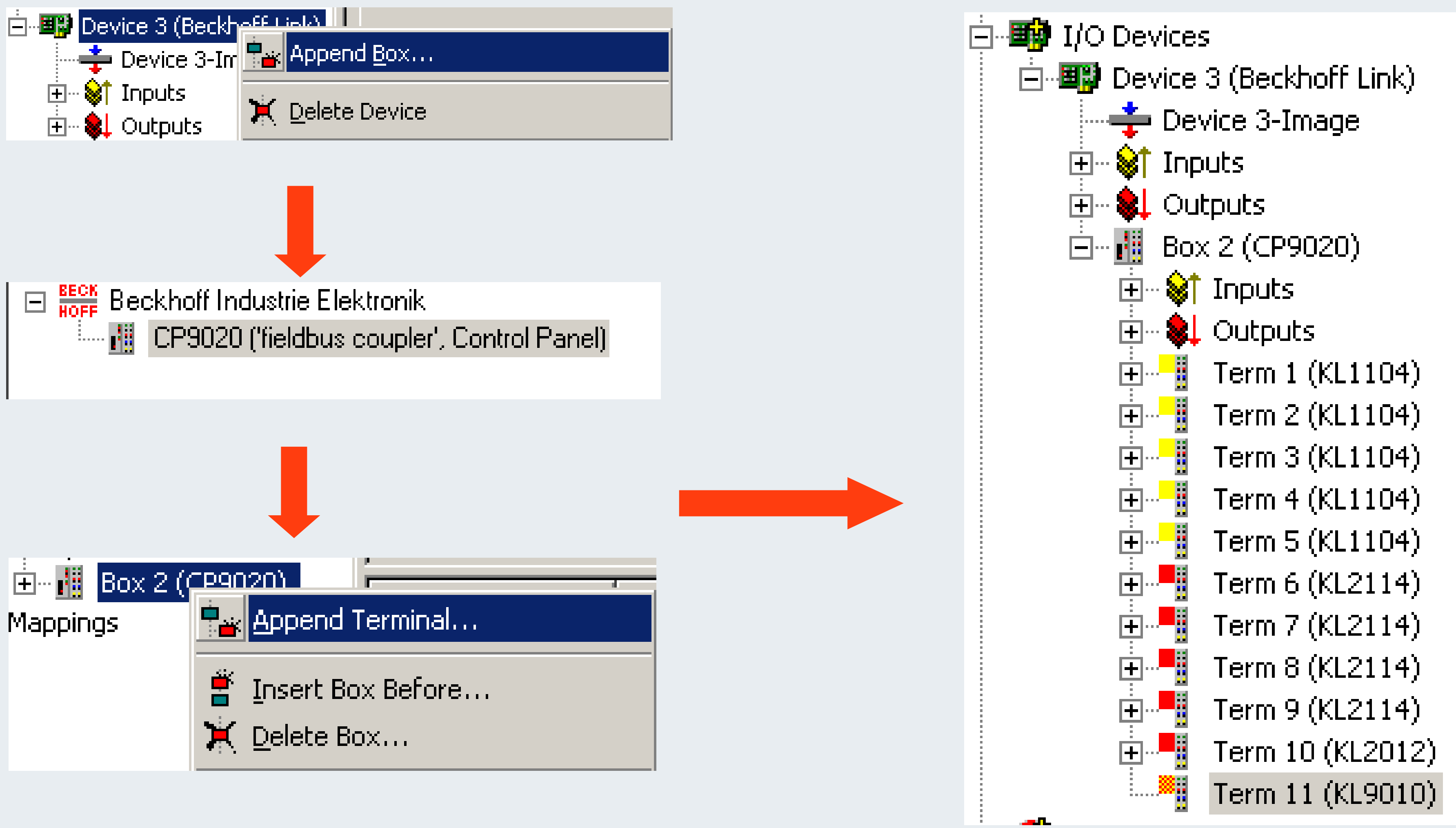
System Manager entry



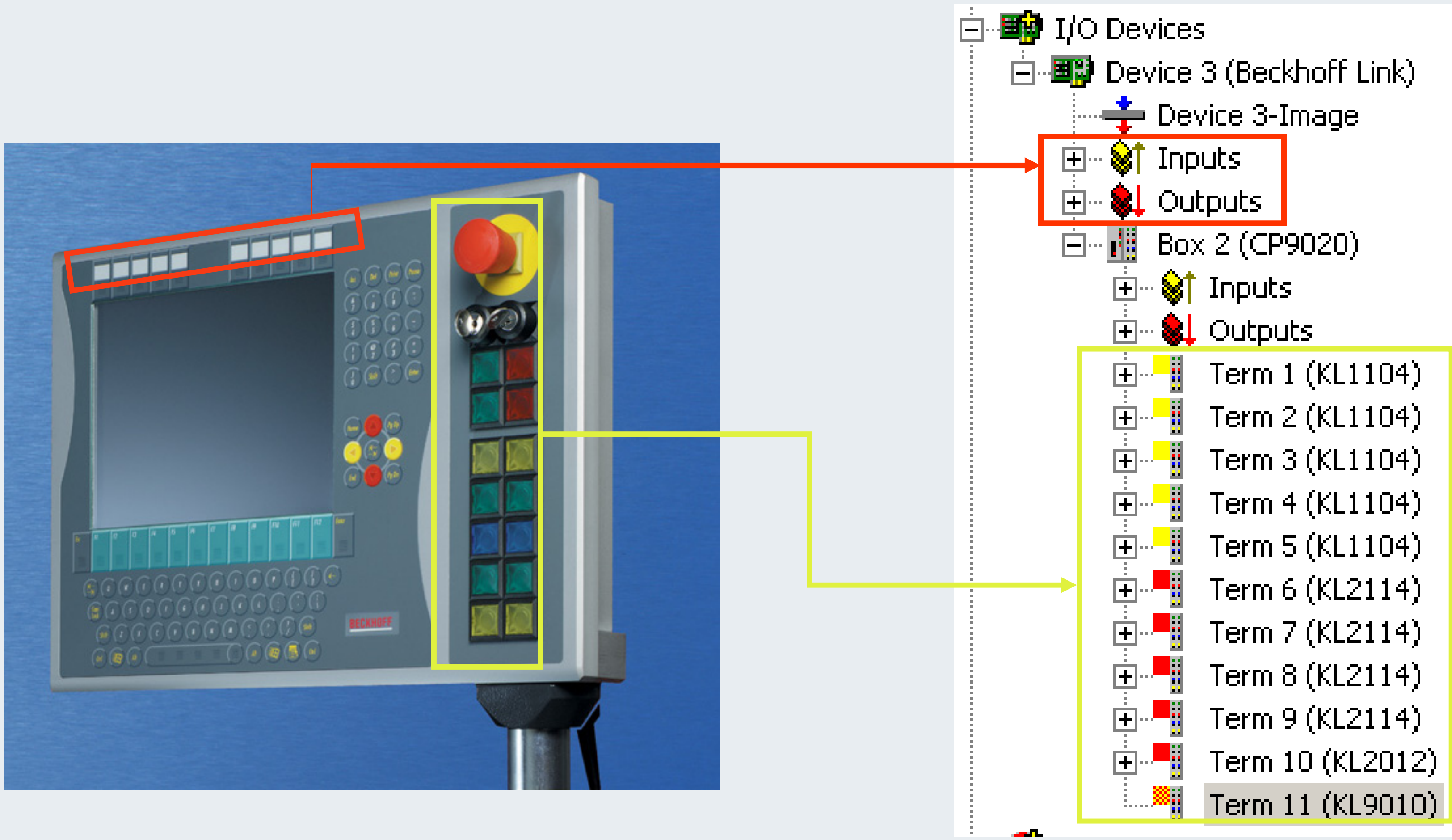


Enter 27 keys and LEDs, although only 10 are available in the hardware

System Manager entry



Bit distribution



I/O assignment at hardware buttons

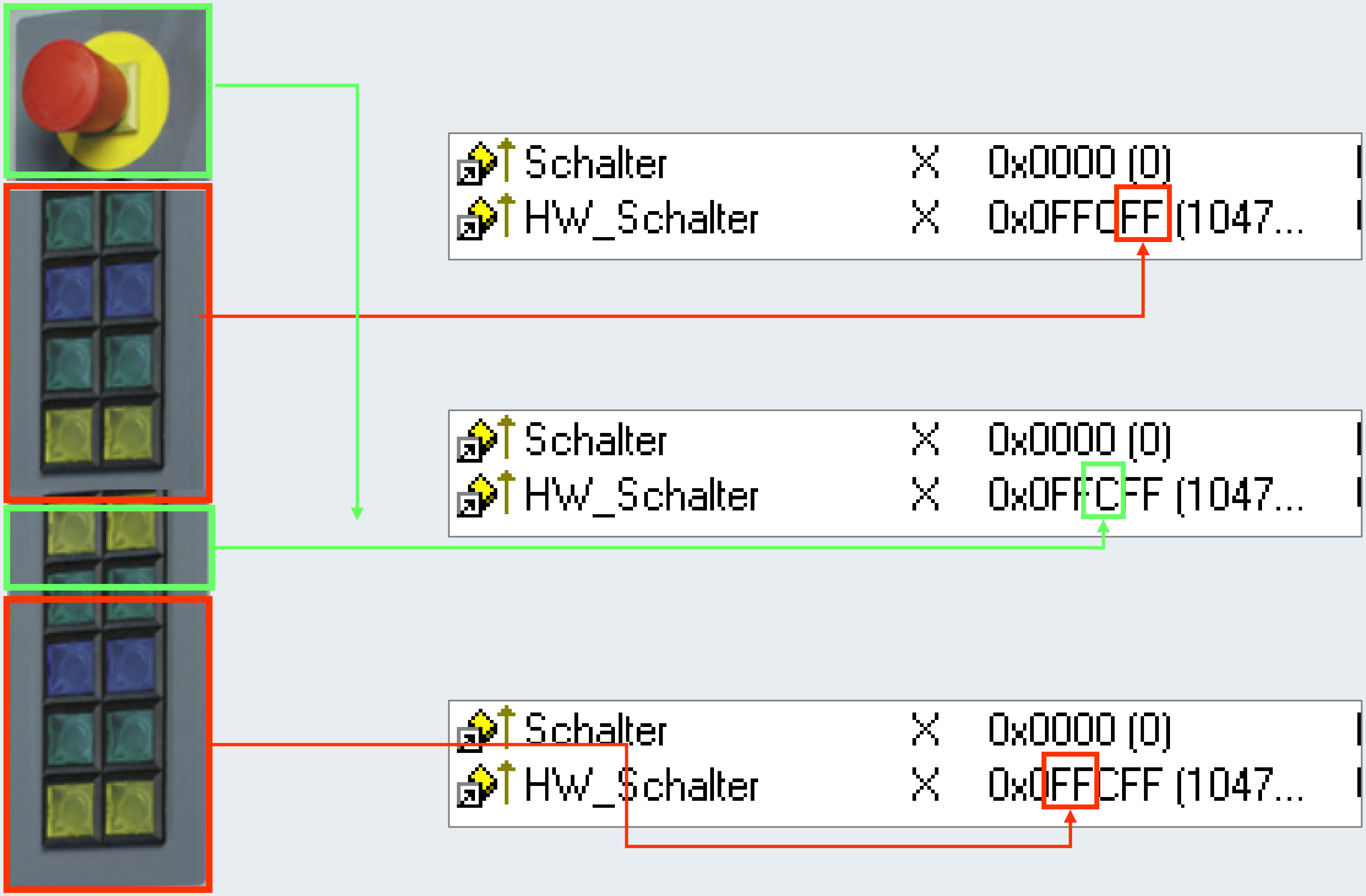


Table of Contents

Table of Contents

Table of Contents

DRAFT

TwinCAT

The Windows Control and Automation Technology

NCI

Numerical Control Interpolation

NCI Overview 1

- TwinCAT NCI consists of
- PLC
- NC-PTP (Point to Point)
- 3D Interpolation

NCI Overview 2

Interface for Interpolation

- **DIN66025 based interpreter (G-Code) or**
- **PLC Interpolation Library**

Limits :

3 drives per channel. In addition :

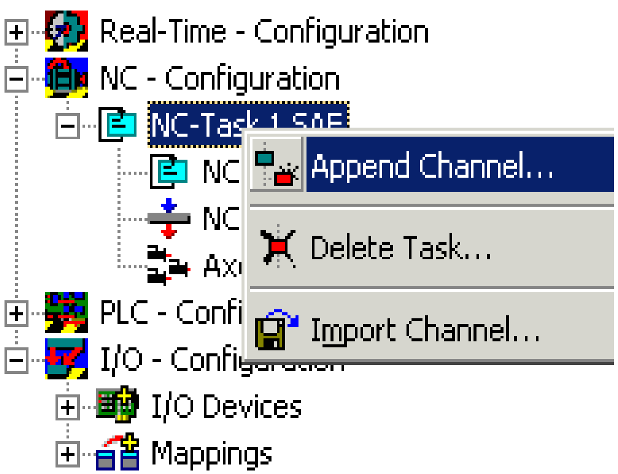
Master/Slave coupling

Online Reconfiguration of axes

Auxiliary axes

- **32 Interpolation channels**

Append interpolation channel



Configuration (Overview)

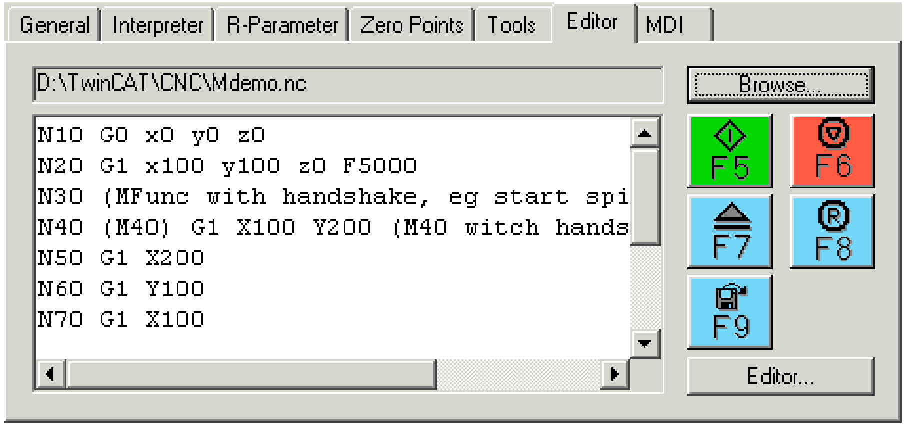
- All axes work correctly in PTP mode

  PTP axes are added to an interpolation group with :

  System Manager

  PLC Library

Interpreter I



NC program structure

Program name (optional)

No. of records

Program end

Example

```
% Test1 (Program begin)
N10 G0 X100 Y100 Z0
M30 (program end)
```

NC program structure

NC record

Each of the NC records consists of no (empty line), one or several NC codes, seperated by blank or tab. Inside Insode a code, blanks are not allowed.

NC code

The first character of the NC code (letter or special character) is the meaning of the code. Typically the first character is a letter or special character.

Example

```
% Test1 (Program begin)
N10 G0 X100 Y100 Z0
M30 (program end)
```

Execution time of codes

Execution time of codes

Codessuch as e.g. G0, G17, that execute throughout the end of the set are described as **modal** according to DIN 66025.

These codes execute until they are suspended or changed by a different code.

Comments

Comments

In order to avoid the interpretion of parts of an NC record or the complete record, the non-interpretion part can be put in parantheses.

A comment ends with the closing parantheses „)“, latest at the end of a record, which means that a comment cannot range over several pages. Nesting of comments are also not possible.

Example:

```
N10 G0 X100 (comment)
```

Record numbers

Main record and sub record

Two types of records are used in NC programs

- Main record
- Sub record

According to DIN 66025, a main record and where appropriate the following records must contain all words necessary to start the workflow with the program section startin at that point.

The main record and the sub record are differentiated in the NC program by the character for the record number.

Record number

Each record can be marked with a record number. The record number is marked with **"N" for sub records and with ":" for main records**.

Comment:

The record number is not necessarily required, anyhow a record not marked with a record number cannot be used as destination for jump codes. Furthermore, the occurance of errors can only be determined imprecisely (previous record number).

Programming motion

Referencing (homing)

Code	**G74**
Suspend	**End of Block**

By standard, drives should by referenced before creation of a 3D group from the PTP channel. Anyhow, referencing in an NC program is also possible.

If the drives are referenced in PTP mode, it is possible to do this for several drives simultaneausly. The NC program can only reference a single drive simulaneously.

Programming motion

Referencing (2)

Example:

```
N10 G74 X
N20 G74 Y
```

Programming motion

Fast motion

Code	**G0**
Suspend	**G01-G03**

Fast motion is used for fast positioning of the tool, but not for processing the workpiece. The drives are positioned with maximum speed.
If several drives have to be positioned in fast motion, the speed is determined by the drive requiring most time for its route.

Programming motion

Linear Interpolation (1)

Code	**G1**
Suspend	**G0, G3, G03**
Parameters	**F - feedrate**

Applying linear interpolation, a drive runs an even path with the feed rate F, that can be anywhere in space. The motion of the concerned drives is completed simultaneously.

The feedrate F describes the motion speed in millimetres per minute. This value is modally effective, i.e. the value need not be repeatedly programmed for later occurring geometrical forms wanting to use same feed rate (it remains applied to later occuring geometrical forms).

Programming motion

Linear Interpolation (2)
Example:

```
N10 G90
N20 G01 X100.1 Y200 F6000
```

Programming motion

Clockwise circular interpolation

Code	G2 and/or G02
Suspend	G0, G1, G3

Clockwise circular interpolation
The code G2 describes a clockwise circular path. Preliminarily it is required to determine the working level (by default G17).

An explicit description of the circle consists of the end point and further parameters. It is possible to choose centre programming and radius programming.

Programming motion

Radius programming

When using radius programming of circular motion, the ending point and the radius of the circle are programmed. The letters „B“ and/or „U“ can be used for the radius.

As G2 determines the direction, the circle is described explicitly. The coordinates of the starting point result from the previous geometry.

Example:

```
N10 G01 G17 X100 Y100 F6000
N20 G02 X200 B200
```

Programming motion

Centre programming

Centre programming is an alternative to radius prgramming. The advantage of centre programming is the possibility to describe full circles.

In the default setting, the centre is stated in relation to the starting point of the circle, using the the parameters I, J and K.

I stands for the X portion
J stands for the Y portion and
K stands for the Z portion.
At least one of these parameters is 0 and need not be specifically programmed

```
N10 G01 G17 X100 Y100 F6000
N20 G02 I50 J0 (J is optional) X200
N30 M30 (program end)
```

Programming motion

Anticlockwise circular interpolation

Code	**G03**
Suspend	**G0, G01, G02**

The code G3 describes an anticlockwise circular path. The parameters and coding possibilities are identical to G2.

Programming motion

The described circles can only operate in the main levels. The CIP circly allow to program a circle anywhere in space. In addition to the ending point, a point on the path is required.

In order to explicitly describe the circle, all 3 points (starting point is defined implicit) must not be collinear. Therefore a full circle cannot be programmed this way.

I, J and K are available for the description of the point on the path. By default they are described in relation to the starting point of the circle.

Code	CIP
Suspend	End of block

Programming motion

CIP Arc (2)
Example

```
N10 G01 X100 Y100 F6000
N20 CIP X200 Y200 I50 J50 K50
```

Programming motion

Helix(1)

Adding a vertical motion to a circular motion results in a helix A helix can only be programmed in the main levels. The same parameters used for the circular path in the main levels are used, in addition the drive in vertical direction is positioned.

Code	**G02 / G03**
Suspend	**G0, G1**

Programming motion

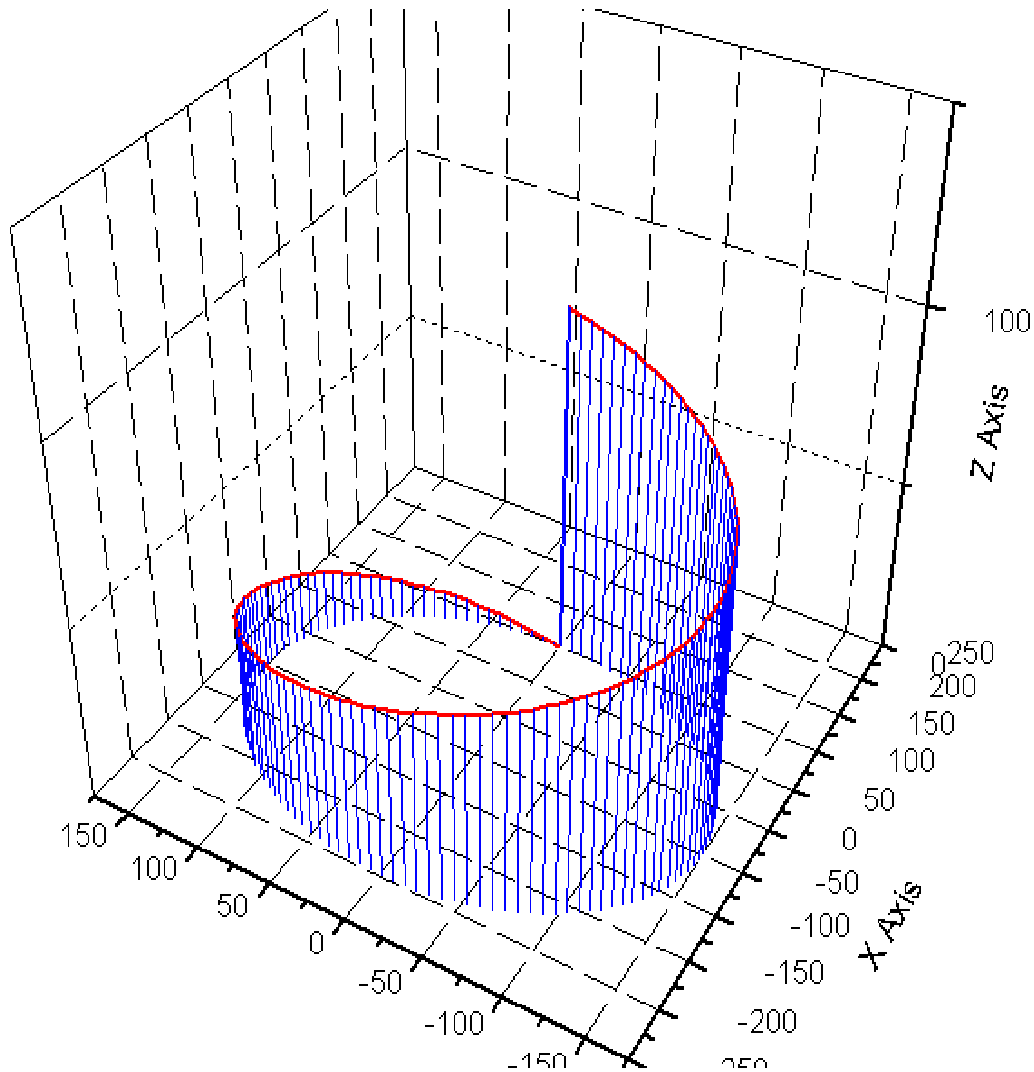
Helix (2)

Example:

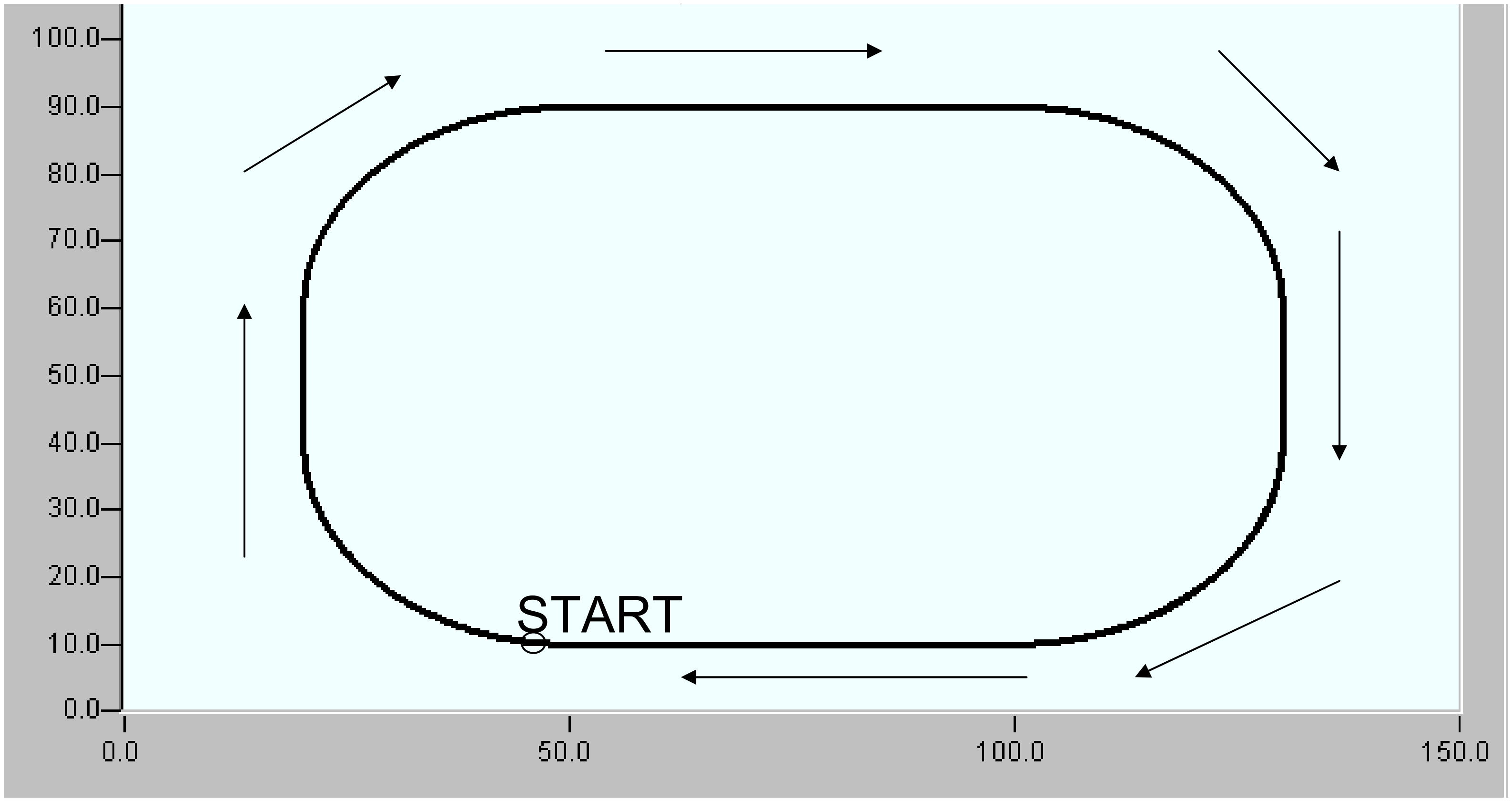
```
N10 G01 G17 X100 Y0 Z0 F6000
N20 G03 I-50 Z100
M30
```

Programming motion

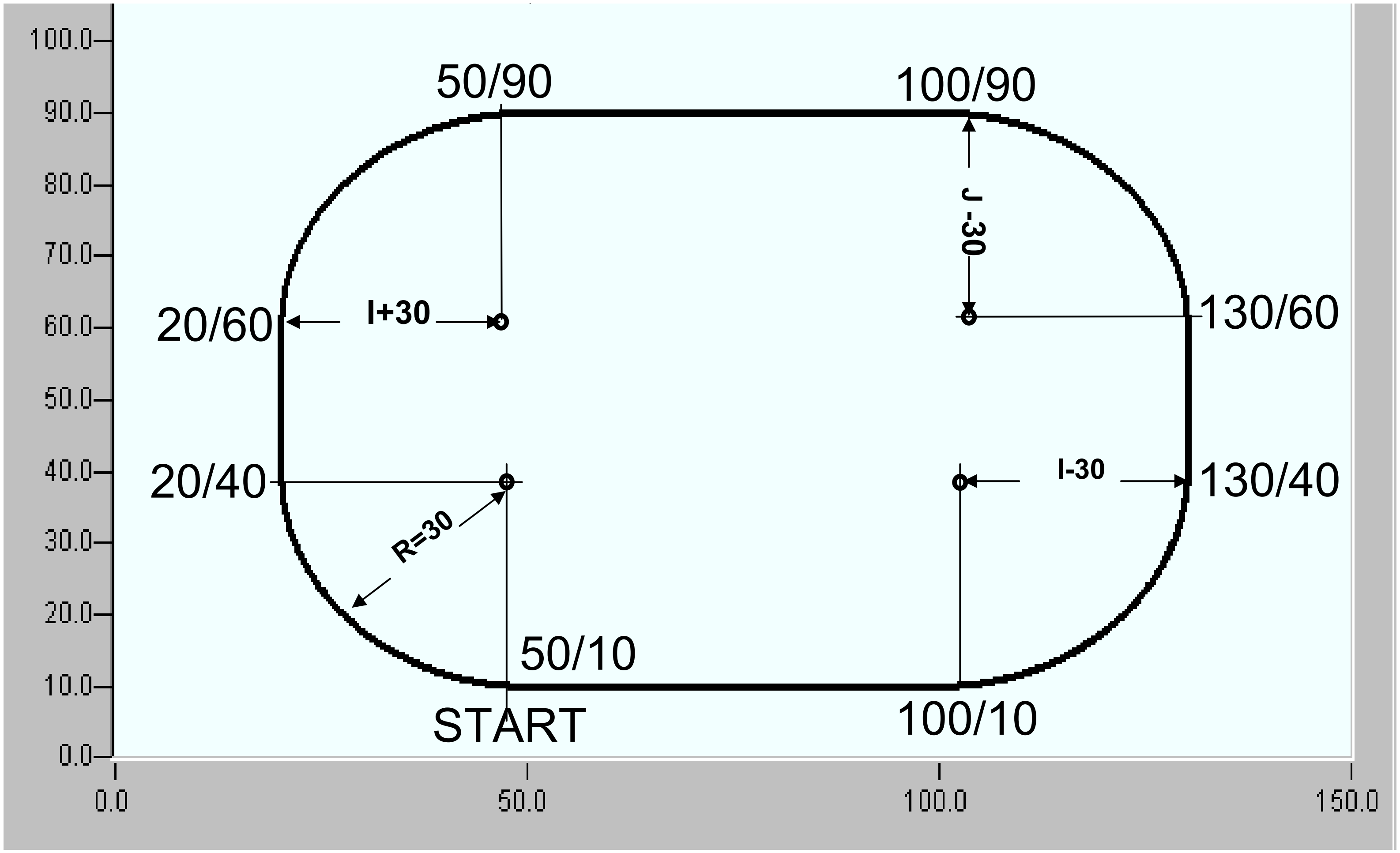
Helix (3)



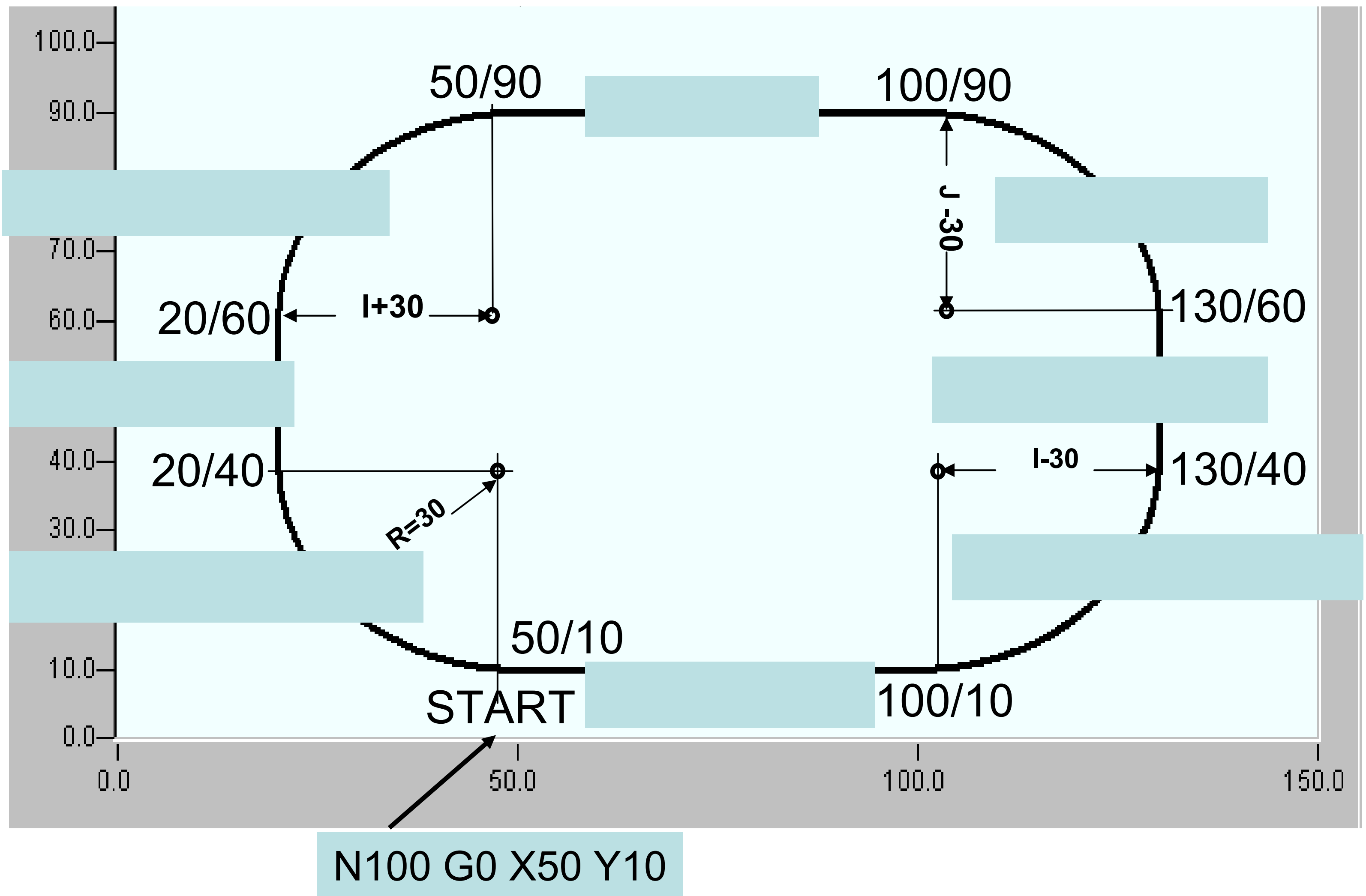
Example G- CODE



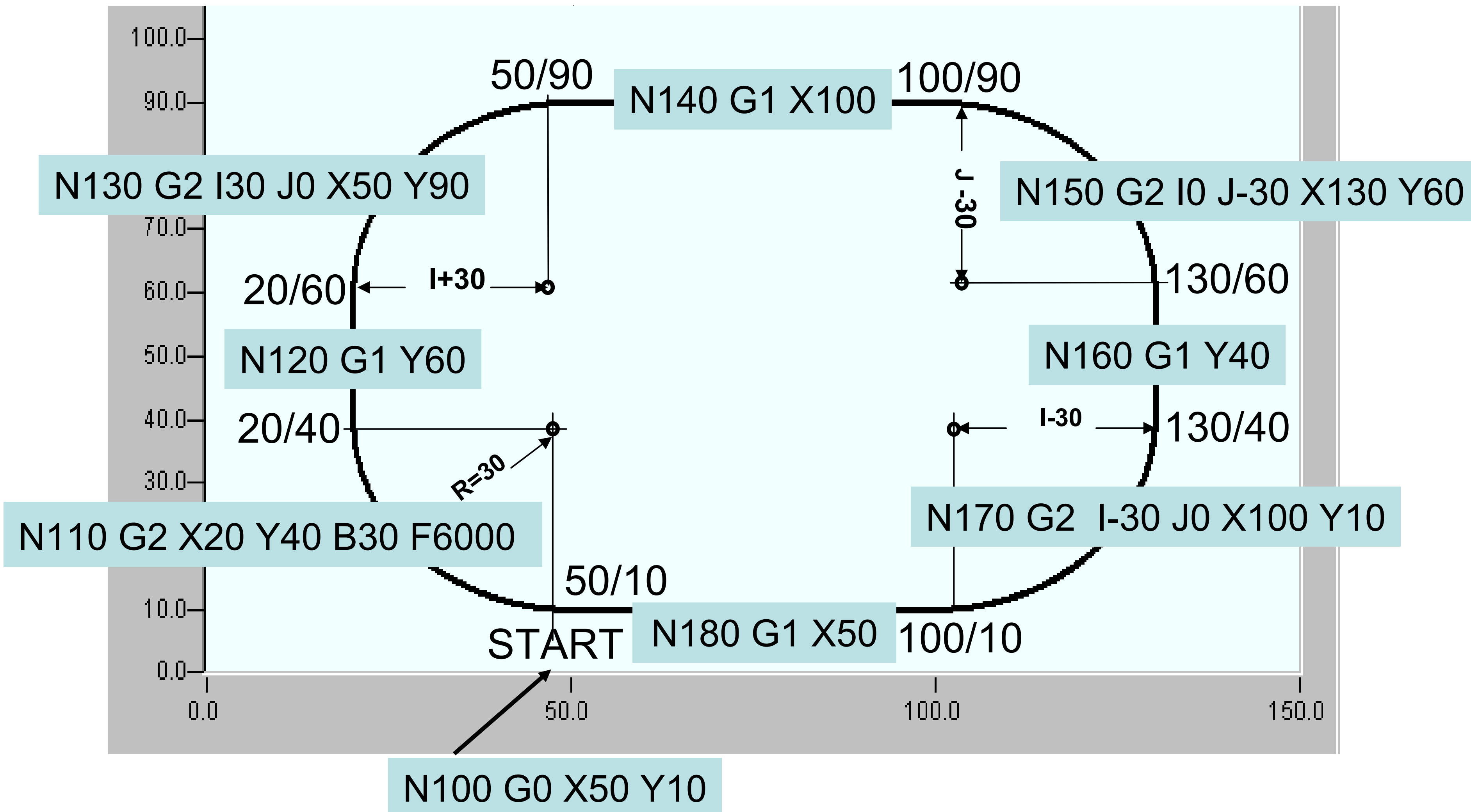
Example G- CODE



Example G- CODE

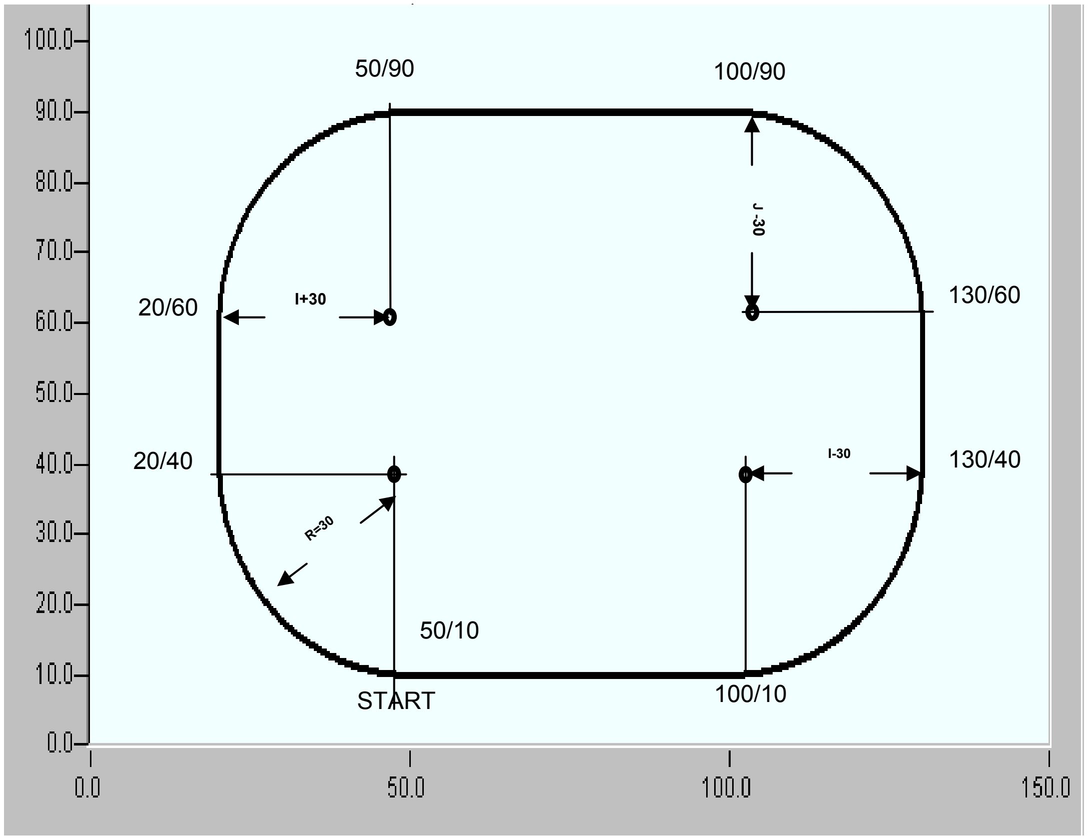


Example G- CODE



Example G- CODE

N100 G0 X50 Y10
N110 G2 X20 Y40 B30 F6000
N120 G1 Y60
N130 G2 I30 J0 X50 Y90
N140 G1 X100
N150 G2 I0 J-30 X130 Y60
N160 G1 Y40
N170 G2 I-30 J0 X100 Y10
N180 G1 X50
N190 M30



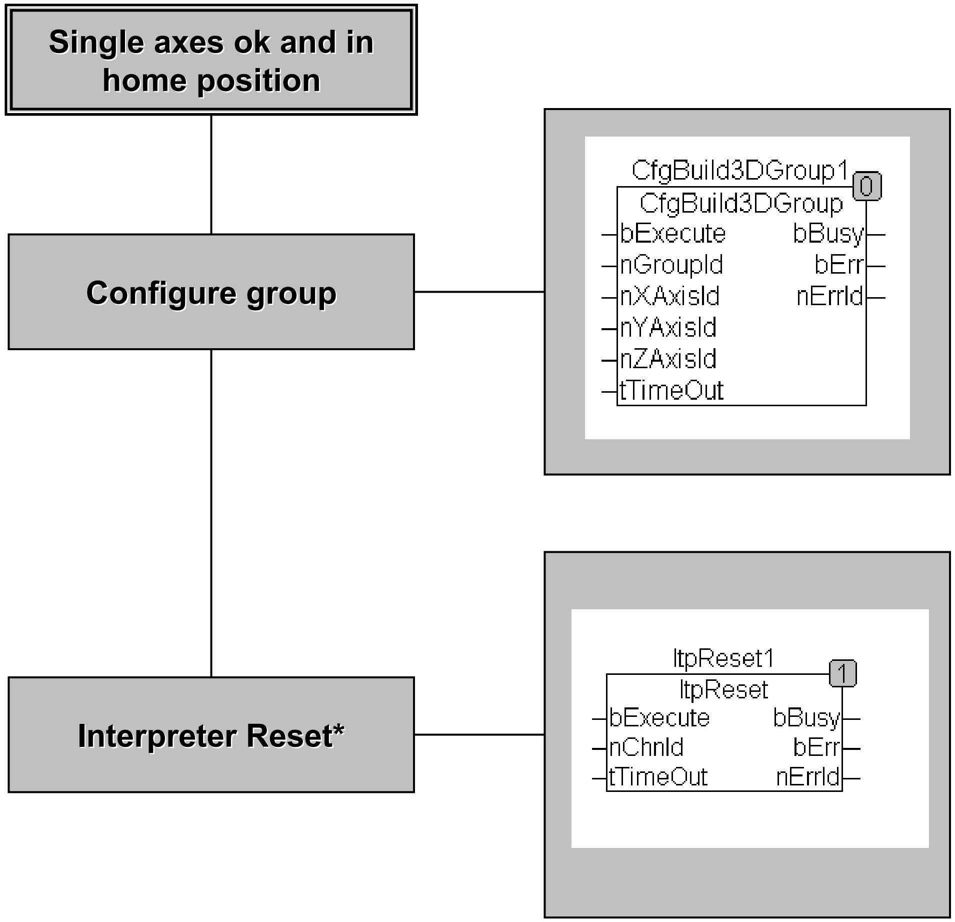
PLC NCI Libraries

TwinCAT PLC Library NC Configuration	Function blocks for the configuration of the interpolation group (i.e. Definition of 3D group) (delivered with TwinCAT NC I)
TwinCAT PLC Library NCI Interpreter	Function Blocks for operating the interpreter (i.e. Load, start, etc.) (delivered with TwinCAT NC I)
TwinCAT PLC Library NCI Interpolation	Function Blocks for the interpolation of geometries **without** using the interpreter (delivered with TwinCAT NC I)

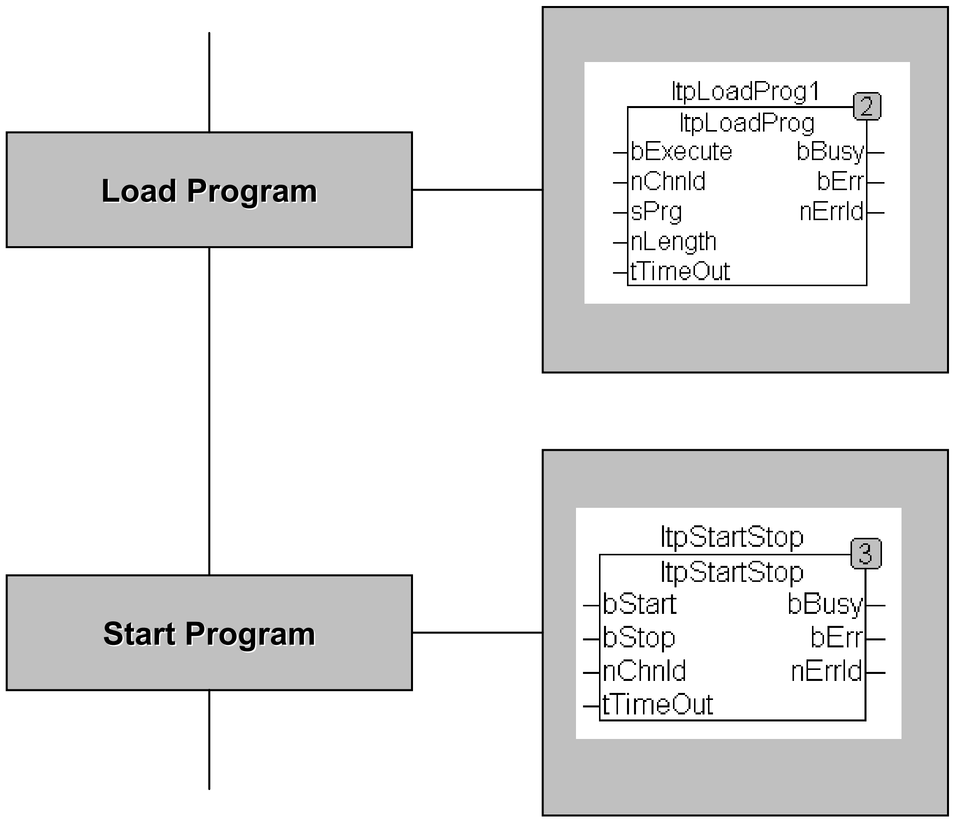
PLC NCI Libraries

TwinCAT PLC Library NCI Interpreter	
TcNciItp.lib	Development kit TwinCAT v2.7.0
TcNci.lib	**Development kit TwinCAT > v2.8.0**

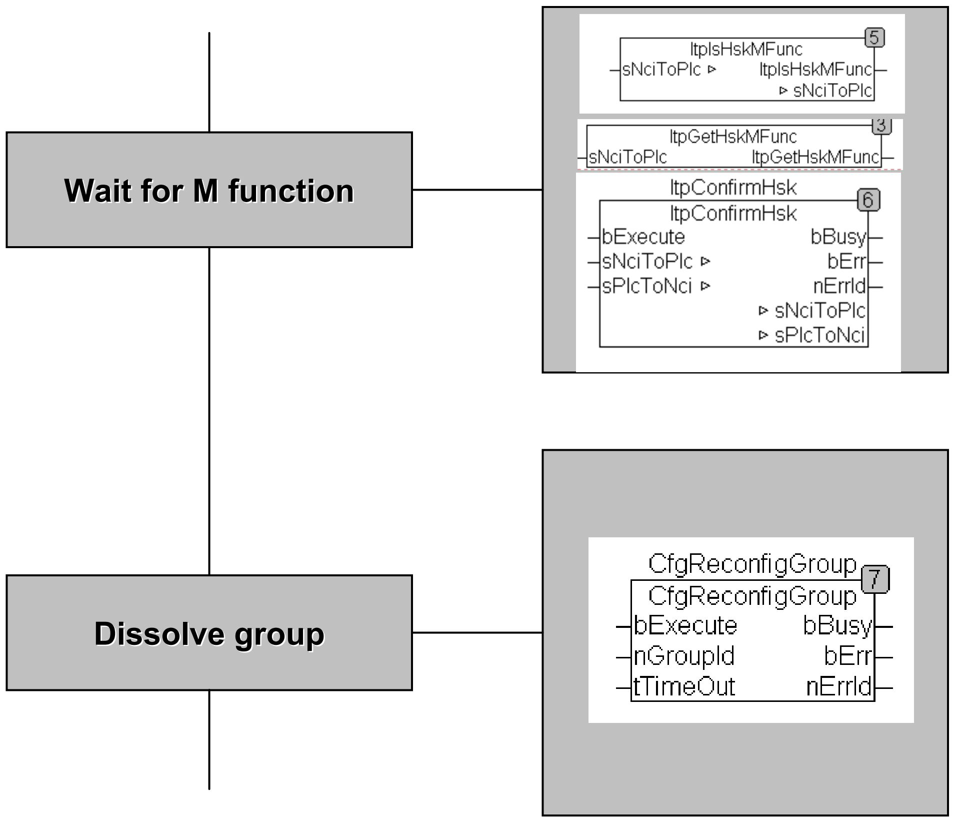
PLC NCI Libraries execution method



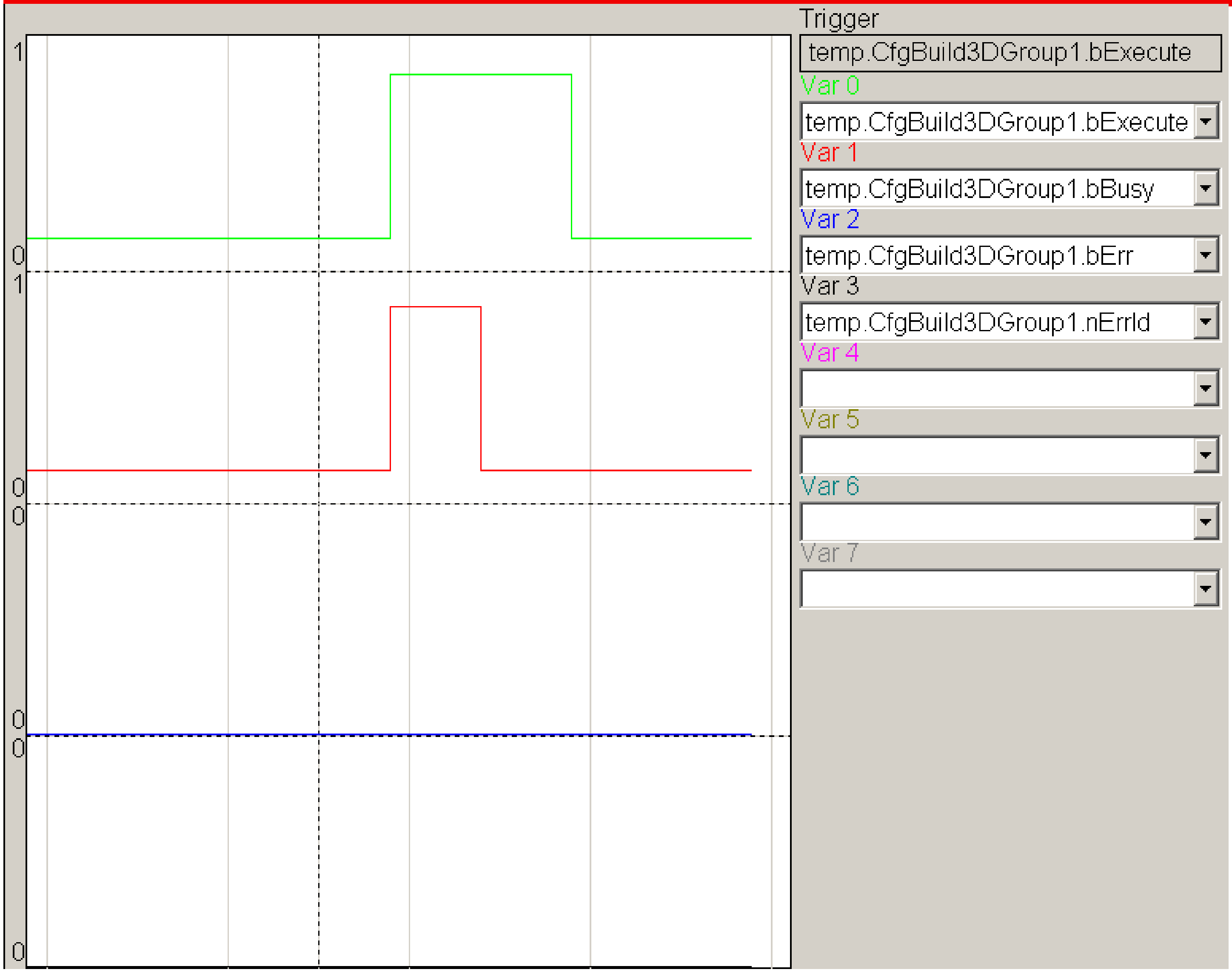
PLC NCI Libraries execution method



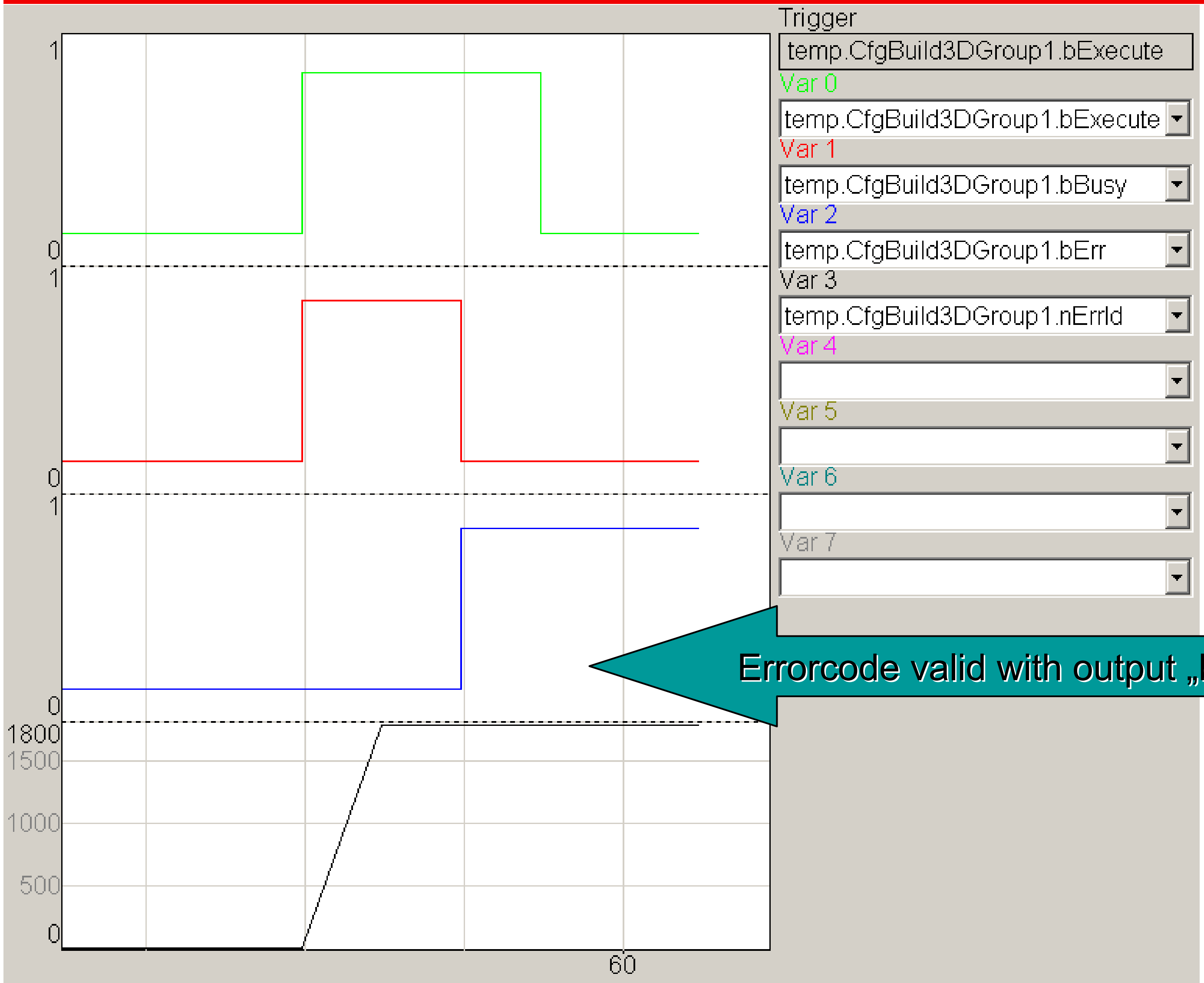
PLC NCI Libraries execution method



PLC NCI Libraries general handshake without error

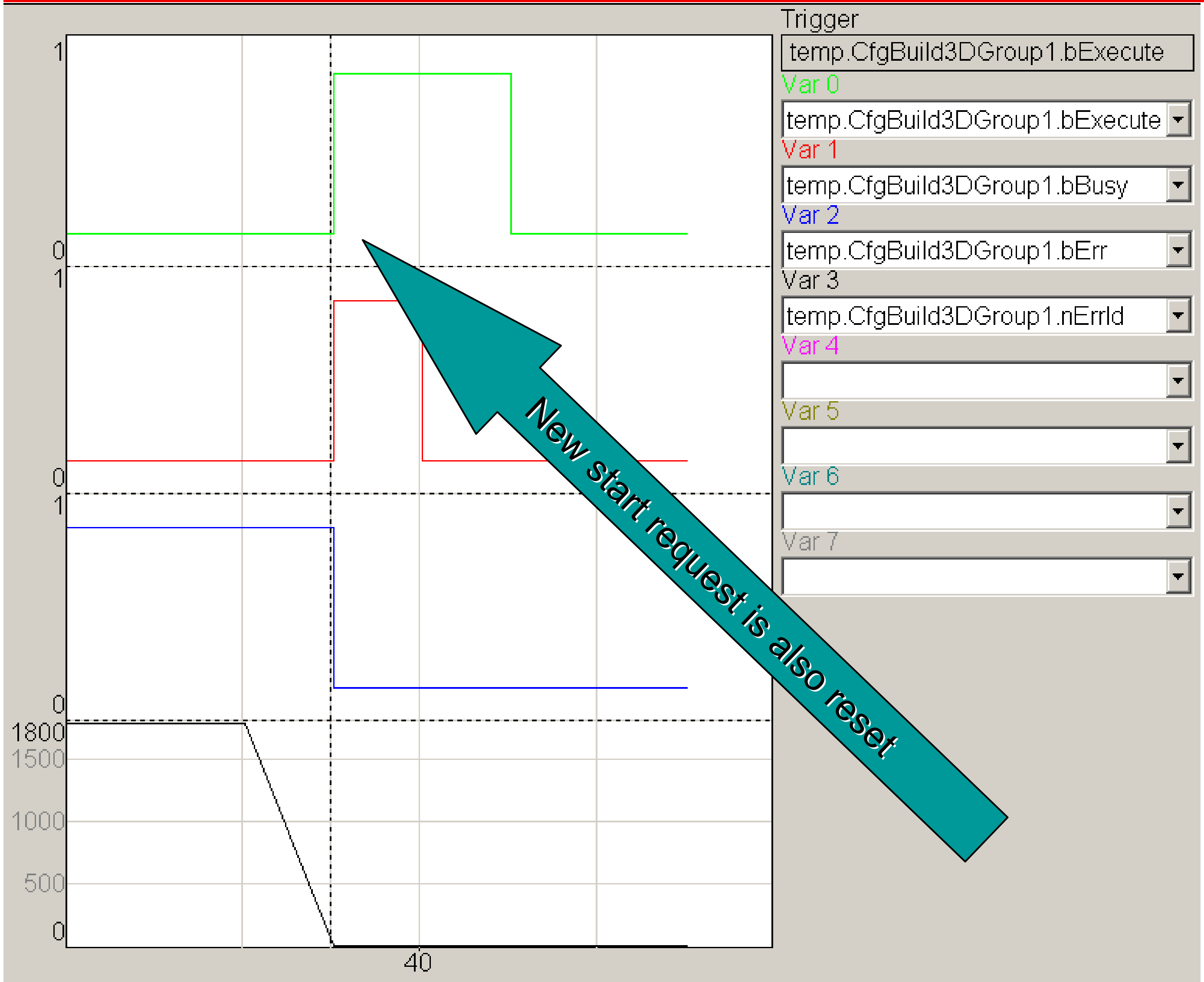


PLC NCI Libraries general handshake in case of error



Errorcode valid with output „bErr“

PLC NCI Libraries general handshake in case of error, reset of error with new rising edge signal



PLC NCI FB‘s

Which FB in which library?

TcNcCfg.Lib	CfgBuild3DGroup bExecute bBusy nGroupId bErr nXAxisId nErrId nYAxisId nZAxisId tTimeOut	TcNci.lib	ItpLoadProg bExecute bBusy nChnId bErr sPrg nErrId nLength tTimeOut	TcNcCfg.Lib	CfgReconfigGroup bExecute bBusy nGroupId bErr tTimeOut nErrId
TcNci.lib	ItpReset bExecute bBusy nChnId bErr tTimeOut nErrId	TcNci.lib	ItpStartStop bStart bBusy bStop bErr nChnId nErrId tTimeOut		

M - functions

M-functions

Why M-functions?

Signal communication NC <-> PLC

It is an advantage fo a variety of constructions, e.g. tongs, drills, transport devices, etc. to not be directly controlled by the NC, but indirectly using a PLC as adaptation and/or link control. That easily enables to consider acknowledgement signals or security conditions, without the NC program or without having to adapt an NC program. M-functions of an NC provide methods of exchanging signals in a digital form: functions are activated or deactivated. It is not considered to pass values (numerical expressions) as working parameters, but this can be achieved in a different way (H-function, T-number, etc.).

Basically there are two methods of exchanging signals: fast signal bits or safe transmission with handshake.

M - functions

Set signal bits

If the function to be controlled does not respond any feedback at all, and it is definitely not necessary to wait, it is possible to make use of fast signal bits.

Signal bits are a field of bits in the PLC / NC channel interface. Each M function is represented with a signal bit, which can be set or reset by the NC, not considering or expecting any signals from the PLC.

An advantage of this method is the performance. As there is definitely no delay by waiting, it is even possible to realize flying M-functions (depending on the type of group executing), that are issued without intermediate stop and without reduction in speed.

Reset signal-bits zurücksetzen

The signal bits remain set in the channel interface until they are explcitly reset or the NC is restarted. Resetting the bits can occur automatically at the end of a record or by calling a different M-function.

M - functions

Safe Handshake

Functions requiring response must be processed with a bi-directional signal exchange between NC and PLC, making use of a sub-interface of the PLC / NC channel interface. Every single M-function is registered as a function number with a request signal. The next function is only executed, if the PLC has acknowledged the M-function as „ready“ and the signal bits have returned to an idle position (no request, no acknowledgement).

This method allows to safely coordinate the NC controlled parts and the PLC controlled parts of the machine.

Only one single M-function using handshake can be active in the NC program, as this type of M-functions are synchronous functions.

In order to make use of the benefits of both methods, parameters for each M-function can be set defining a sequence of default rules per M function by setting bits in a control mask.

M - functions

Setup of an M-Defs File (edited manually before TwinCAT 2.9)

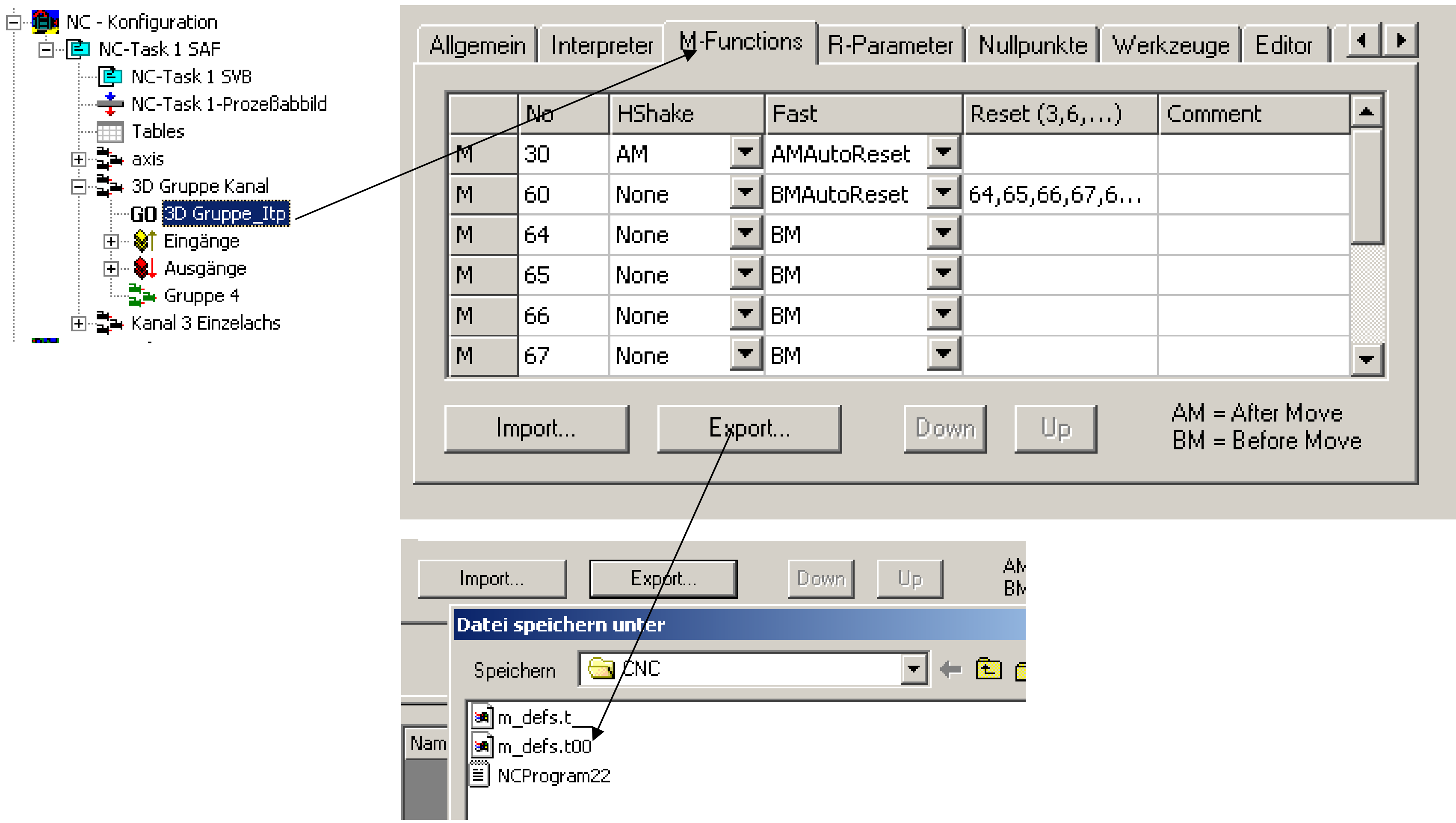
```
//---------------------------------------
// Beispiel für eine Datei zur Festlegung
// der M-Funktionsregeln.
//---------------------------------------
//
// M-Nummer     00 ... 159
//
// Regel-Bits   01h =   1 = Handshake before Move
//              02h =   2 = Set FastBit before Move
//              04h =   4 = Handshake after Move
//              08h =   8 = Set FastBit after Move
//              10h =  16 = Reset FastBit before Move
//              20h =  32 = Reset FastBit after Move
//              40h =  64 = donot use
//              80h = 128 = AutoReset FastBit at Line End
//
// Abgeloescht[10]
//
//---------------------------------------
30 140 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
60 130 64 65 66 67 68 69 70 71 -1 -1
64   2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
65   2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
66   2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
67   2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
68   2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
69   2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
70   2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
71   2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

M - functions explanation of values

Bit	Value	Description
0	1	Transfer with handshake. In case the same record contains codes for motion, the handshake occurs before the motion
1	2	Output as signal bit. In case the same record contains codes for motion, the output occurs before the motion
2	4	Transfer with handshake. In case the same record contains codes for motion, the handshake occurs after the motion
3	8	Output as signal bit. In case the same record contains codes for motion, the output occurs after the motion
4	16	Reset signal- bit before the motion. In case the same record contains codes for motion, the resetting occurs before the motion
5	32	Reset signal- bit after the motion. In case the same record contains codes for motion, the resetting occurs before the motion
6	64	reserved
7	128	The signal bit of the M-Function is automatically deleted at the end of a record, i.e. it is only effective per record

M - functions

Editing possible via system manager



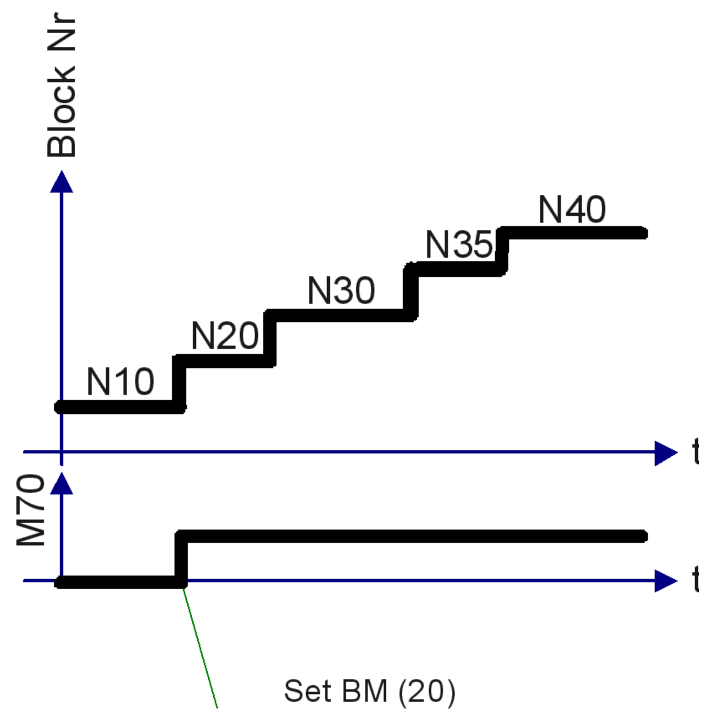
Handshake NCI-PLC

Example fast M-functions

M	70	None	BM

```
N10 G1 X100 Y100 F5000    (Kommentar)
N20 X100 Y110 M70
N30 X10  Y50
N35 X20
N40 M30
```

Fast M-Funktion



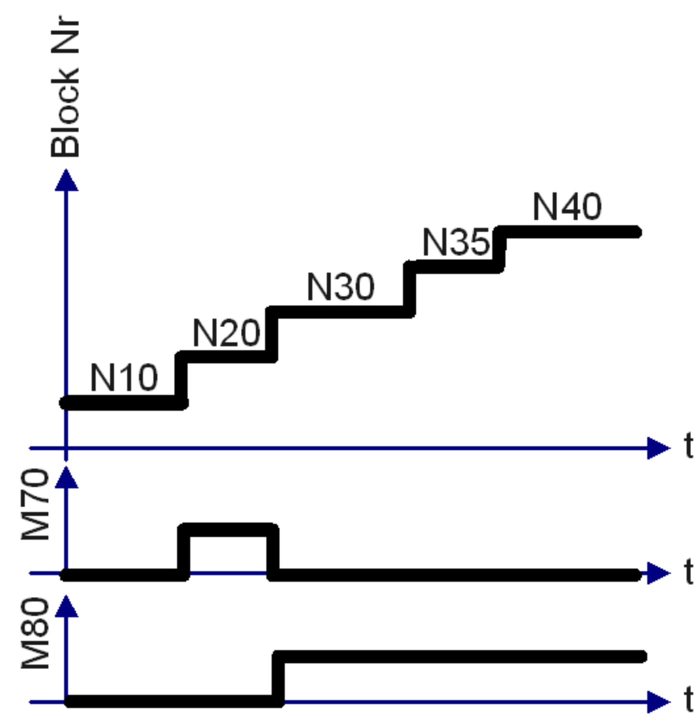
Handshake NCI-PLC

Example fast M-functions

	No	HShake	Fast	Reset (3,6,...)
M	70	None	BM	
M	80	None	BM	70

```
N10 G1 X100 Y100 F5000    (Kommentar)
N20 X100 Y110 M70
N30 X10  Y50 M80
N35 X20
N40 M30
```

Fast M-Funktion

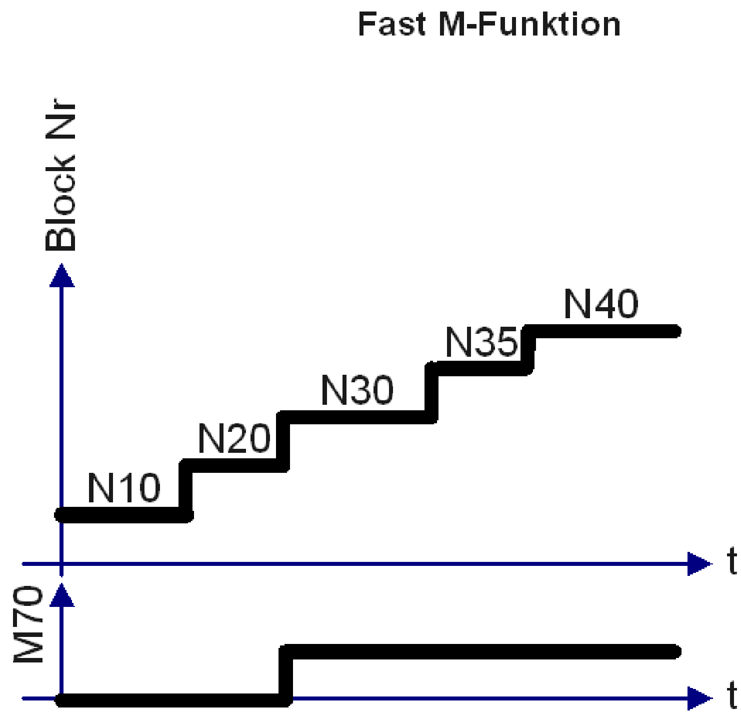


Handshake NCI-PLC

Example fast M-functions

	No	HShake	Fast	Reset (3,6,...)
M	70	None	AM	

```
N10 G1 X100 Y100 F5000    (Kommentar)
N20 X100 Y110 M70
N30 X10  Y50
N35 X20
N40 M30
```



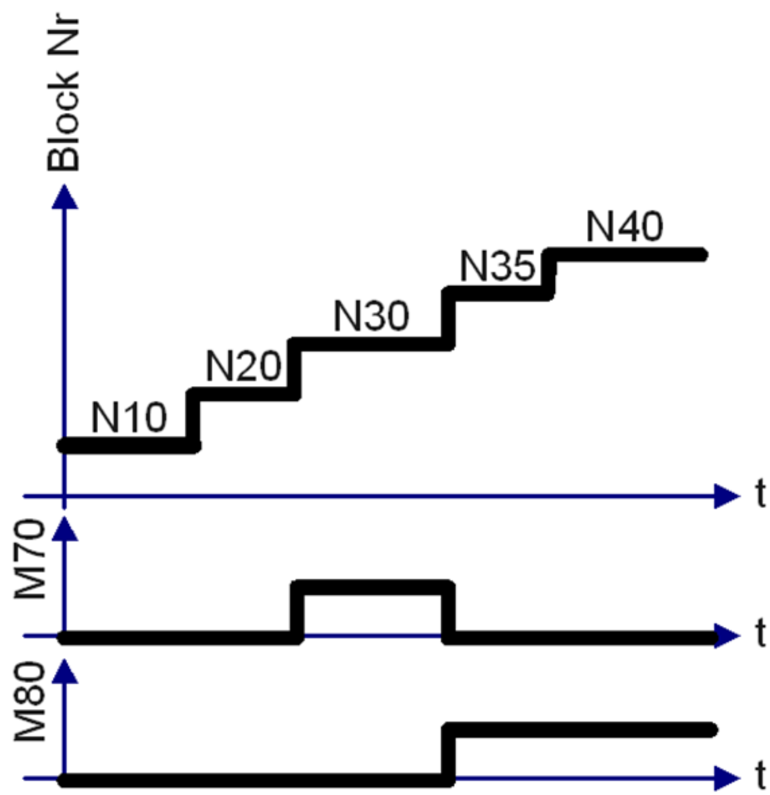
Handshake NCI-PLC

Example fast M-functions

	No	HShake	Fast	Reset (3,6,...)
M	70	None	AM	
M	80	None	BM	70

```
N10 G1 X100 Y100 F5000    (Kommentar)
N20 X100 Y110 M70
N30 X10  Y50
N35 X20 M80
N40 M30
```

Fast M-Funktion



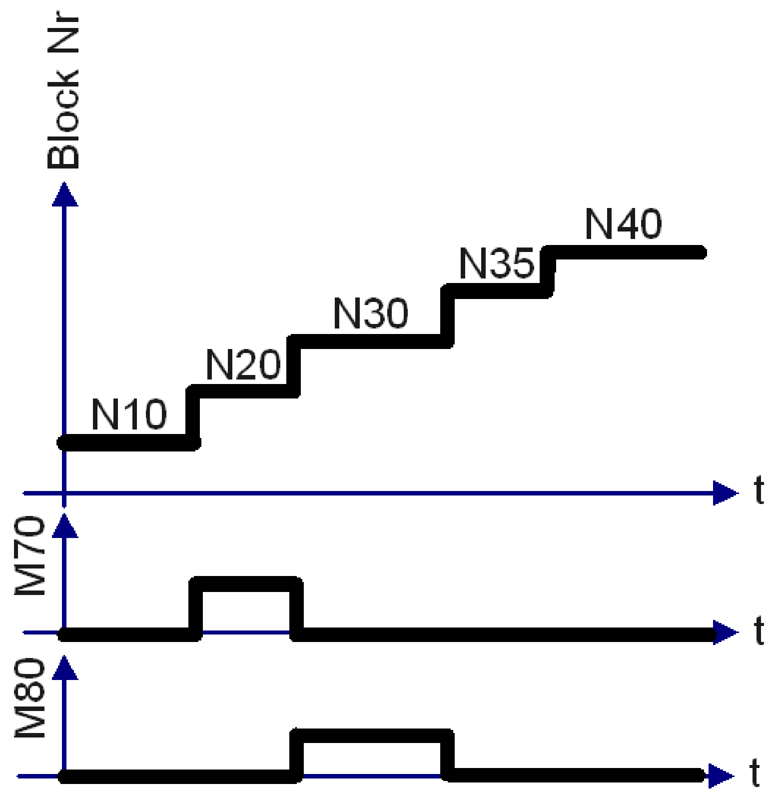
Handshake NCI-PLC

Example fast M-functions

	No	HShake	Fast	Reset (3,6,...)
M	70	None	BMResetAM	
M	80	None	BMResetAM	

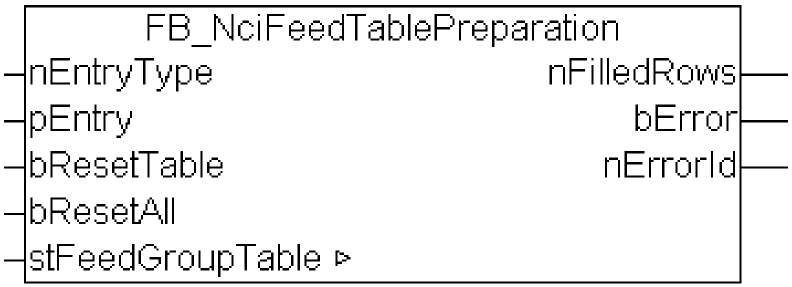
```
N10 G1 X100 Y100 F5000    (Kommentar)
N20 X100 Y110 M70
N30 X10  Y50
N35 X20 M80
N40 M30
```

Fast M-Funktion



TcPlcInterpolation

The NC Codes transfer directly from the PLC



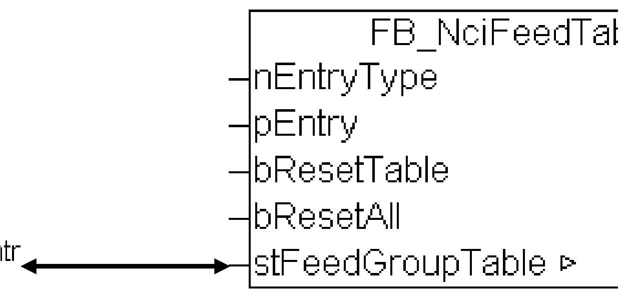
FB inscribes Entrys to the NC in a data buffer.

Module works synchron at call, no Busy.

At longer transfers, the call can be distribute in more cycles.

Buffer to NC

```
TYPE ST_NciFeedGroupTable : ARRAY[1..NciMaxTableEntries]OF ST_NciFeedGroupEntr
END_TYPE
```



TcPlcInterpolation

The NC Codes transfer directly from the PLC

```
TYPE E_NciEntryType :
(
    E_NciEntryTypeNone := 0,
    E_NciEntryTypeGeoStart  := 1,
    E_NciEntryTypeGeoLine   := 2,
    E_NciEntryTypeGeoCirclePlane := 3,
    E_NciEntryTypeGeoBezier3 := 10,
    E_NciEntryTypeMFuncHsk      := 20,
    E_NciEntryTypeMFuncFast     := 21,
    E_NciEntryTypeMFuncResetAllFast      := 23,
    E_NciEntryTypeHParam   := 24,
    E_NciEntryTypeSParam   := 25,
    E_NciEntryTypeTParam   := 26,
    E_NciEntryTypeDynOvr := 50,
    E_NciEntryTypeVertexSmoothing   := 51,
    E_NciEntryTypeTfDesc     := 100,
    E_NciEntryTypeEndOfTables := 1000
);
END_TYPE
```

G1

G2 / G3

```
TYPE ST_NciGeoLine :
STRUCT
    nEntryType: E_NciEntryType  := E_NciEntryTypeGeoLine;
    nDisplayIndex:     UDINT;
    fEndPosX:    LREAL;
    fEndPosY:    LREAL;
    fEndPosZ:    LREAL;
    fEndPosQ1:  LREAL;
    fEndPosQ2:  LREAL;
    fEndPosQ3:  LREAL;
    fEndPosQ4:  LREAL;
    fEndPosQ5:  LREAL;
    fVelo:          LREAL;
    bRapidTraverse:  BOOL;
    bAccurateStop:          BOOL;   (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

```
FB_NciFeedTal
nEntryType
pEntry
bResetTable
bResetAll
stFeedGroupTable ⊳
```

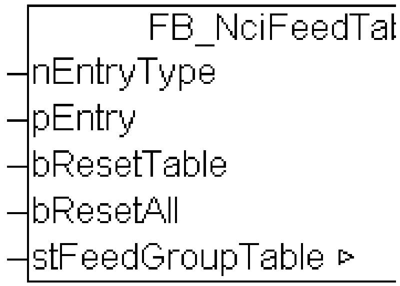
Don`t change

Xpos, YPos,
Feed..

TcPlcInterpolation

Fill the list FB invoice same Instance

Startpositions E_NciEntryTypeGeoStart



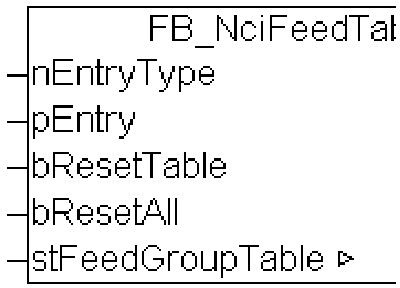
If so parameter Splineinterpolation entering

Geometry 1 programming G01 / G02

Geometry 2 programming G01 / G02

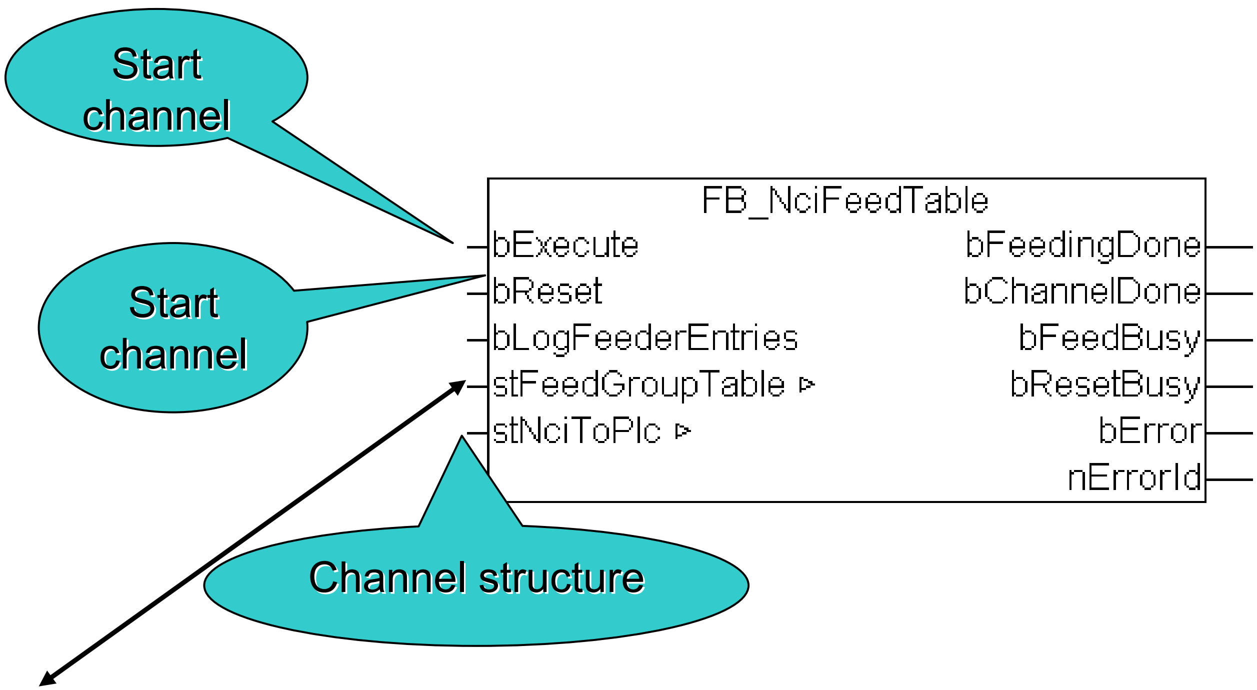
Geometry n programming G01 / G02

Identification E_NciEntryTypeEndOfTables



TcPlcInterpolation

Start the NC with




```
TYPE ST_NciFeedGroupTable : ARRAY[1..NciMaxTableEntries]OF ST_NciFeedGroupEntr
END_TYPE
```

Buffer to NC filled with FbFeedtablepreparation

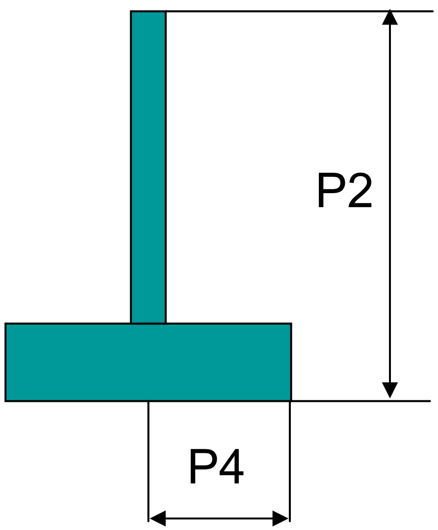
Milling cutter radius correction

Editing tool data:

-System Manager

-PLC program

-Piece program



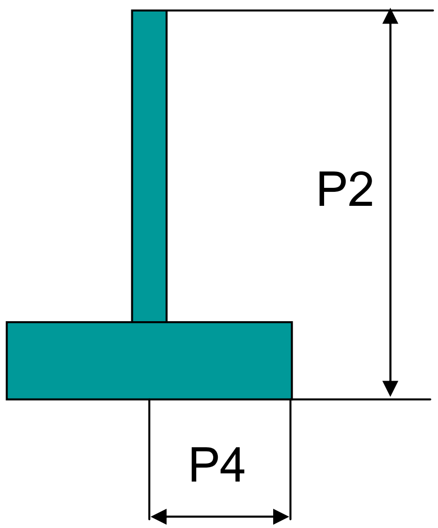
Example parameters of shaft milling cutter:

Type shaft milling cutter 20	(P1)
Length of milling cutter	(P2)
Radius	(P4)
Wear: length	(P5)
Wear: radius	(P7)

cartesian tool adjustment in X-direction (P8)
cartesian tool adjustment in Y-direction (P9)
cartesian tool adjustement in Z-direction (P10)

Milling cutter radius correction

Example



Example parameters of shaft milling cutter:

Type shaft milling cutter 20	(P1=20)
Length of milling cutter	(P2=10)
Radius	(P4=5)
Wear: length	(P5=0)
Wear: radius	(P7=0)

cartesian tool adjustment in X-direction (P8=0)
cartesian tool adjustment in Y-direction (P9=0)
cartesian tool adjustement in Z-direction (P10=0)

Milling cutter radius correction

- Kanal2
  - GO 3D Gruppe_Itp
  - Eingänge
  - Ausgänge
  - Gruppe 4

Entries in System Manager (saved in <channelid>.WZ)

Allgemein | Interpreter | M-Functions | R-Parameter | Nullpunkte | Werkzeuge | Editor

	TNr.(P0)	Typ(P1)	Geom.(P2)	Geom.(P3)	Geom.(P4)	Verschl.(P5)	Verschl.(P6)	Verschl.(P7)	P8
D 1	1	20	10.000000	0.000000	5.000000	0.000000	0.000000	0.000000	0
D 2	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
D 3	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
D 4	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
D 5	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
D 6	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
D 7	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0

Allgemein | Interpreter | M-Functions | R-Parameter | Nullpunkte | Werkzeuge | Editor

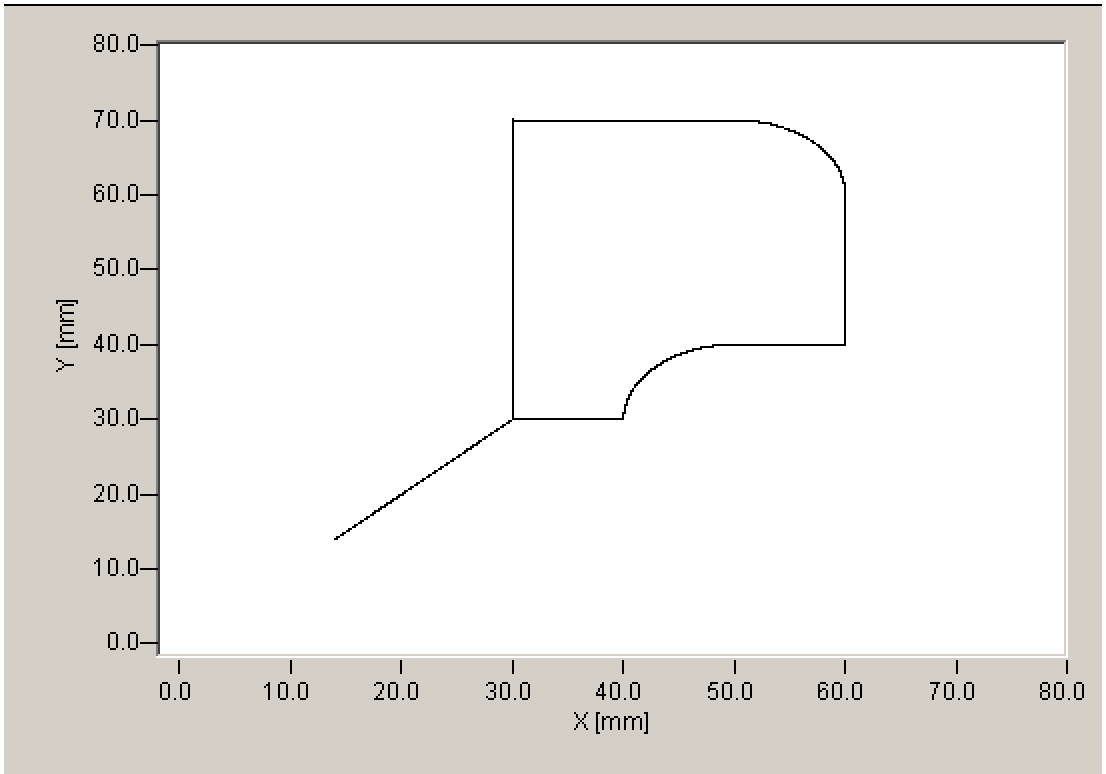
	P8	P9	P10	P11	P12	P13	P14	P15
D 1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
D 2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
D 3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
D 4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
D 5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
D 6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
D 7	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Milling cutter radius correction

NC Program without tool correction

NC Editor

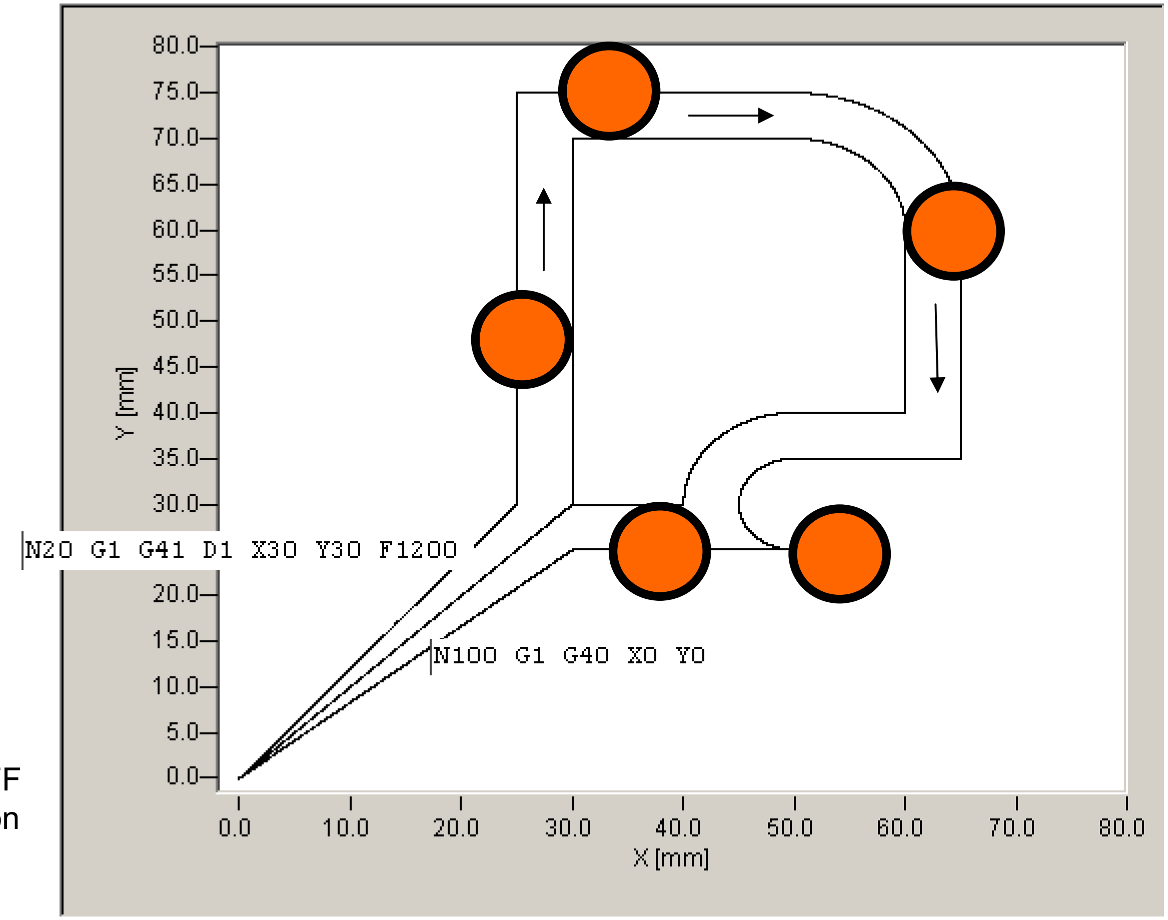
```
%Programm Start
N10 G17 G0 X0 Y0 (Arbeitsebene XY)
N20 X30 Y30 F1200
N30 X30 Y70
N40 X50 Y70
N50 G2 X60 Y60  J-10 F1200
N60 G1 x60 Y40
N70 G1 X50 Y40
N80 G3 X40 Y30  J-10
N90 G1 X30 Y30
```



Milling cutter radius correction

NC Program with tool correction

```
(----------Korrektur-----------)

N10 G17 G0 X0 Y0 (Arbeitsebene XY)
N20 G1 G41 D1 X30 Y30 F1200
N30 X30 Y70
N40 X50 Y70
N50 G2 X60 Y60  J-10 F1200
N60 G1 x60 Y40
N70 G1 X50 Y40
N80 G3 X40 Y30  J-10
N90 G1  X30 Y30
N100 G1 G40 D0 X0 Y0
```

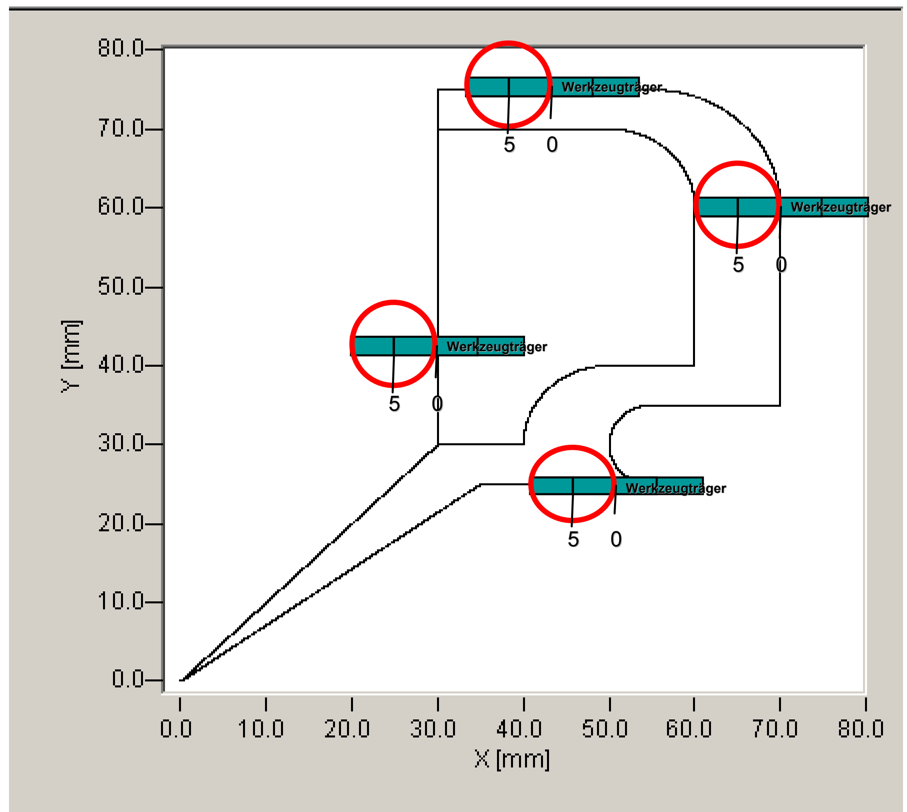


N20:
G41 Milling cutter radius correction left of workpiece
(G0 or G1 must be active)

D1 select tool memory D1
N100: G40 Milling cutter radius correction OFF
D0 selection of tool memory OFF (tool position adjustment)

Milling cutter radius correction

```
N10 G17 G0 X0 Y0 (Arbeitsebene XY)
N20 G1 G41 D1 X30 Y30 F1200
N30 X30 Y70
N40 X50 Y70
N50 G2 X60 Y60  J-10 F1200
N60 G1 x60 Y40
N70 G1 X50 Y40
N80 G3 X40 Y30  J-10
N90 G1  X30 Y30
N100 G1 G40 D0 X0 Y0
```



Tool is 5 mm further left:

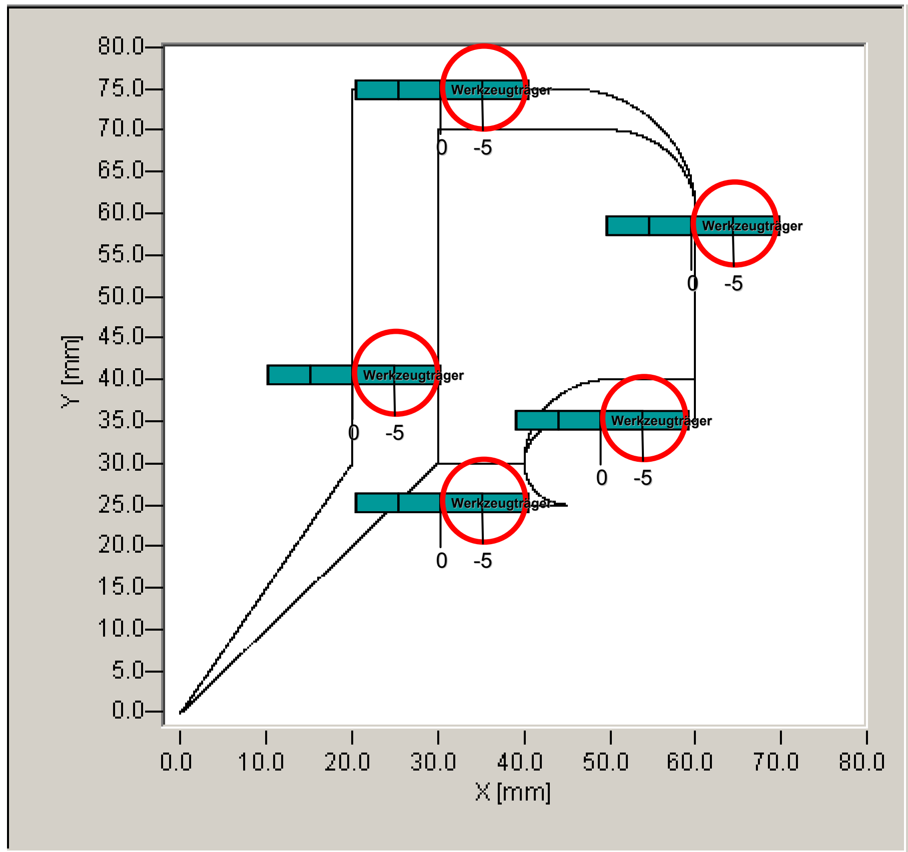
X Offset 5 mm

Milling cutter radius correction

```
N10 G17 G0 X0 Y0 (Arbeitsebene XY)
N20 G1 G41 D1 X30 Y30 F1200
N30 X30 Y70
N40 X50 Y70
N50 G2 X60 Y60  J-10 F1200
N60 G1 x60 Y40
N70 G1 X50 Y40
N80 G3 X40 Y30  J-10
N90 G1  X30 Y30
N100 G1 G40 D0 X0 Y0
```

Allgemein | Interpreter | M-Functio

	P8
D 1	-5.000000



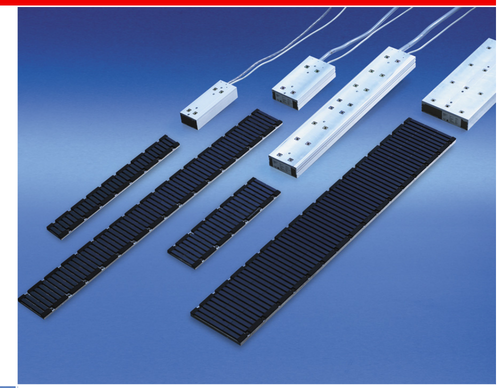
Tool 5 mm further right:

X Offset –5mm

Drive Technology

The drive system for high dynamic positioning tasks

Linear
Servomotors
ALxxxx



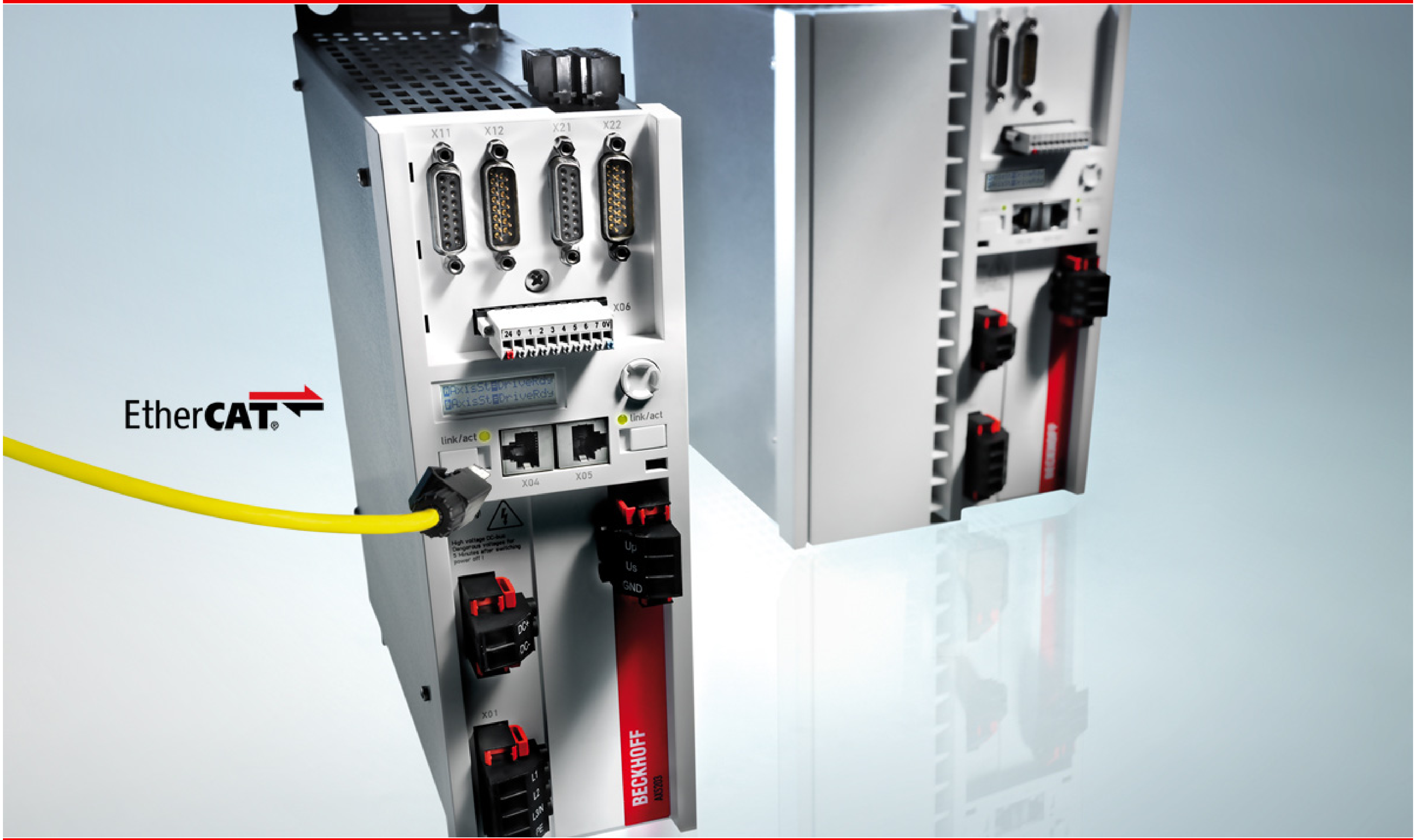
Synchronous
Servomotors
AM2000/AM3xxx

Digital Compact
Servo Drive
AX5xxx

Digital Compact
Servo Drive
AX20xx

Digital Compact
Servo Drive
AX25xx

AX5000 | Digital Compact Servo Drive



Technical highlights

- fast control technology
  - current control: min. 31.25 µs
  - speed control: min. 125 µs
  - position control: min. 125 µs
- high-speed EtherCAT system communication
- 1- or 2-channel Servo Drive
  - optimised for multi-axis applications
  - variable motor output allocation in 2-channel drives
- active DC-Link and brake energy management
- variable motor interface with
  - multi-feedback interface
  - flexible motor type selection
  - scalable, wide range motor current measurement

Technical highlights

- high-speed capture inputs
- wide range voltage 100…480 V AC (up to 40 A)
- integrated mains filter
- integration of safety functions (optional)
  - restart lock
  - TwinSAFE: intelligent safety functions for Motion Control
- compact design for simple control cabinet installation
  (for 300 mm depth) (up to 40 A)
- AX-Bridge – the quick connection system for power supply, DC-Link (up to 40 A)
- variable cooling concept
  (fanless, forced cooling, cold plate)

Features

Motor feedback:
Sin/Cos 1Vss, single- or multi-turn, EnDat, Hiperface, BiSS

Motor feedback: Resolver, TTL

8 digital I/Os, e.g. enable, limit switch, capture input, error message

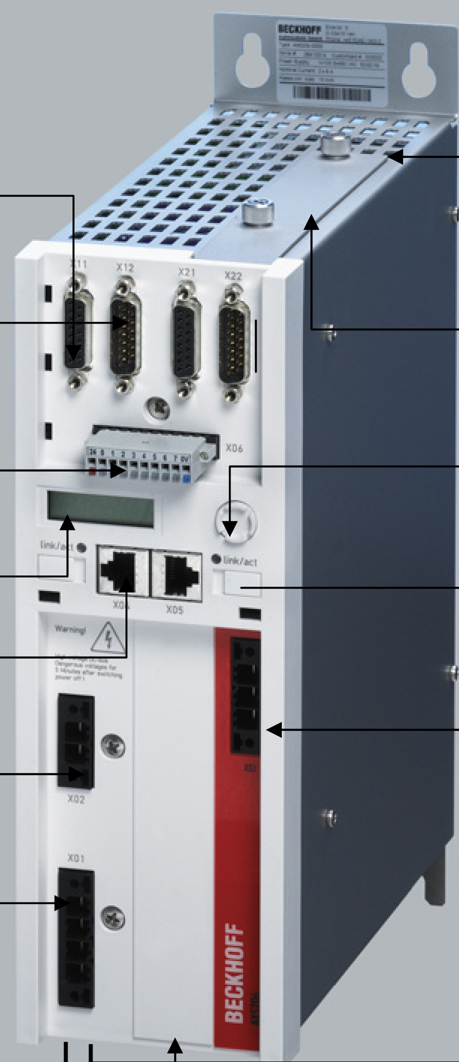
Status display, e.g. axis identifier or error message

EtherCAT system bus

DC-Link

DC power supply/Power supply 100 V AC…480 V AC

Brake control/motor temperature monitoring



Optional slot for interface boards, e.g. additional feedback

Optional slot for restart lock or optional TwinSAFE safety cards

Navigation buttons (Enter, Up/Down)

Operating material identification

24 V DC control and braking voltage

Motor circuits

AX51xx | Rated output current of 1,5 A, 3 A, 6 A and 12 A

- 1-axis Servo Drive for motors up to 12 A rated current

AX51xx | 1-channel Servo Drive

AX5118, AX5125 | Rated output current of 18 A, 25 A and 40 A

- 1-axis Servo Drive for motors up to 40 A rated current

AX51xx | Rated output from 60 A up to 170 A

NEW

AX51xx | Rated output from 60A up to 170 A

NEW

- Enlargement of the Servo drive product line AX5000 with Servo Drives from 60 A up to 170 A.
- three sizes with rated currents of 60 A, 72 A, 90 A, 110 A, 143 A, 170 A

- Features
  - Highspeed EtherCAT system communication
  - Connection voltage: 400…480 V AC +-10%
  - Multi feedback interface
  - flexible selection of motor type
  - Highspeed capture inputs
  - Diagnosis and parameter display
  - Integrated mains filter up to 72 A rated current acc. to Cat. C3, acc. to EN61800-3
  - optional safety functions:
    - restart lock
    - intelligent TwinSAFE safety function

AX51xx | 1-channel Servo Drive

Technical data at 50 °C ambient temperature

NEW

Technische Daten	AX5160	AX5172	AX5190	AX5191	AX5192	AX5193
Rated output current	1 x 60 A	1 x 72 A	1 x 90 A	1 x 110 A	1 x 143 A	1 x 170 A
Rated supply voltage	3 x 400 V AC – 10%... 3 x 480 V AC + 10 %					
DC-Link voltage	max. 890 V DC					
Peak output current (1)	120 A	142 A	135 A	165 A	215 A	221 A
Rated connected load for 480 V AC	42 kVA	50 kVA	62kVA	76 kVA	99 kVA	118 kVA
Continuous braking power	external					
Max. braking power	external					

(1) I_{eff} für max. 3 s

AX52xx | 2-channel Servo Drive

- 2-axis Servo Drive for two motors with a total current up to 12 A

AX52xx | 2-channel Servo Drive

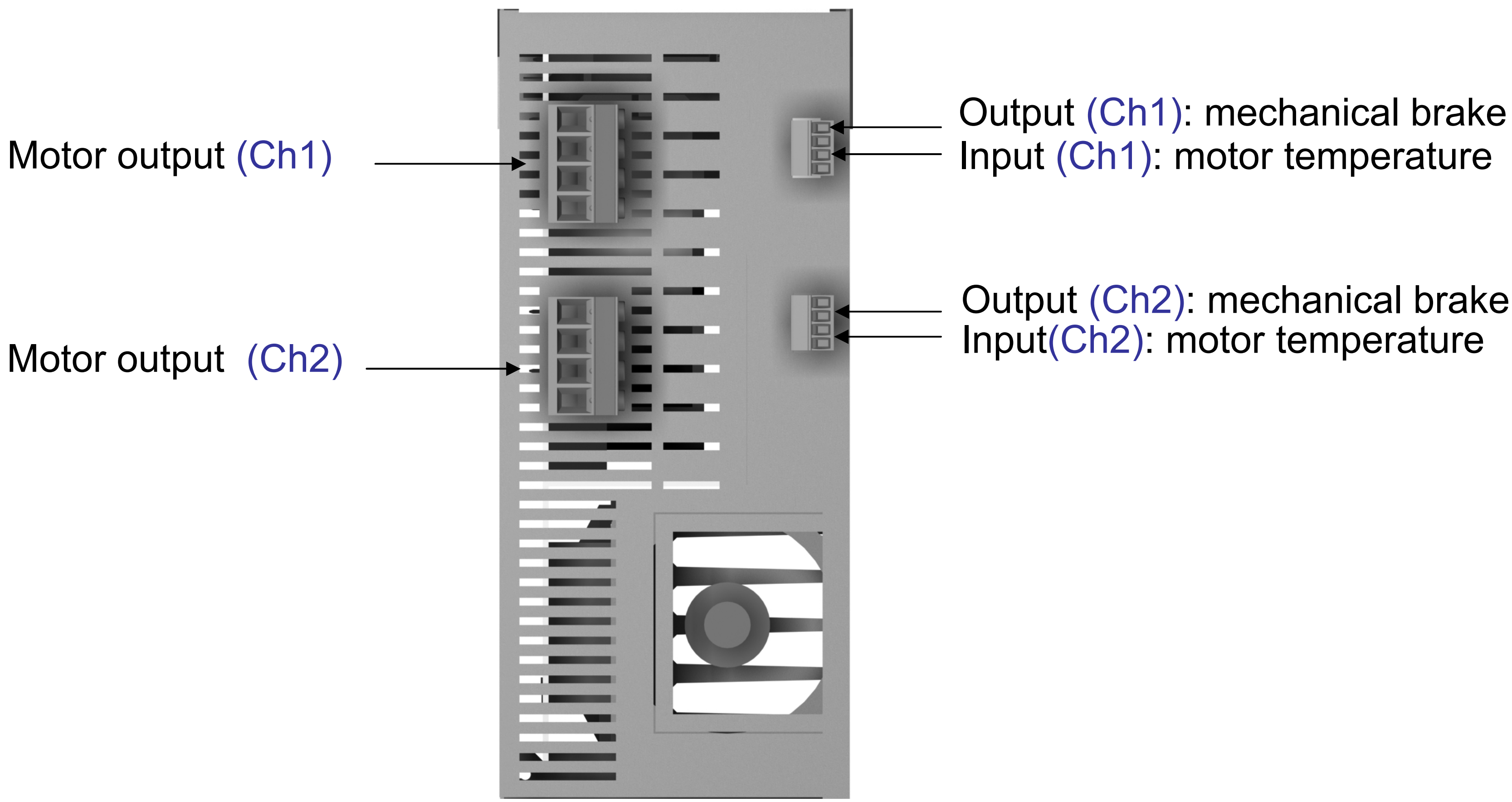
Technical data at 50 °C ambient temperature

Technical data	AX5201	AX5203	AX5206
Rated output current	2 x 1.5 A	2 x 3 A	2 x 6 A (*)
Rated supply voltage	3 x 100 V AC – 10 %... 3 x 480 V AC + 10% 1 x 100 V AC – 10 %... 1 x 240 V AC + 10 %		
DC-Link voltage	max. 890 VDC		
Peak output current (1)	10 A	20 A	26 A
Rated connected load for S1-operation (2)	2.5 kVA	5 kVA	10 kVA
Continuous braking power (2)	50 W	150 W	90 W
Max. braking power (2)	14 kW		

(1) I_{eff} for max. 7 seconds
(2) internal brake resistor
(*) With a 1-phase mains, the total current is limited to 9 A.

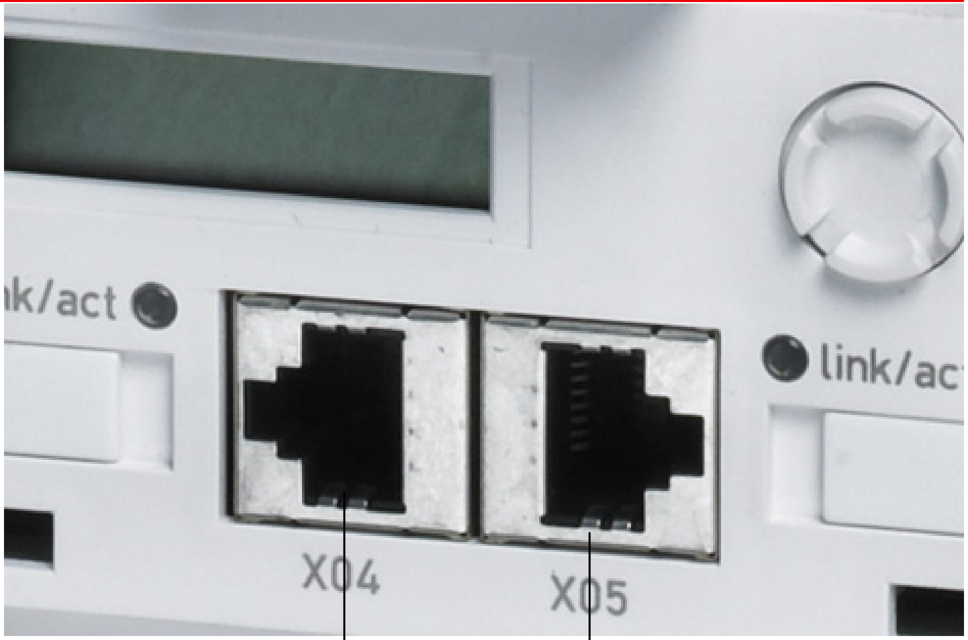
AX5000 | The features in detail

Bottom


Motor output (Ch1)
Motor output (Ch2)
Output (Ch1): mechanical brake
Input (Ch1): motor temperature
Output (Ch2): mechanical brake
Input(Ch2): motor temperature

Communication

- high-speed EtherCAT system bus
- SERCOS profile for drive technology as implemented per IEC 61491
- other fieldbus systems over external gateways



Cycle times and clock frequencies

- different cycle times for various application requirements
  - 62,5 µs current control loop
  - 4 kHz frequency for minimum power dissipation

- example configurations:

EtherCAT (minimum)	Position loop	Speed loop	Current loop	IGBT switching	Motor cable
62,5 µs	NC/PLC	NC/PLC	31,25 µs	16 kHz	32 kHz
62,5 µs	NC/PLC	NC/PLC	62,5 µs	8 kHz	16 kHz
125 µs	125 µs	125 µs	31,25 µs	16 kHz	32 kHz
125 µs	125 µs	125 µs	62,5 µs	8 kHz	16 kHz
125 µs	125 µs	125 µs	125 µs	4 kHz	8 kHz

Wide voltage range

- same drive for all common power supply systems – no options, no variants, e.g.
  - 1 x 100 V AC for Asia
  - 1 x 115 V AC for North America
  - 3 x 200 V AC for Asia
  - 1 x 230 V AC for Europe
  - 3 x 230 V AC for North America
  - 3 x 400 V AC for Europe
  - 3 x 480 V AC for North America

Multi feedback interface

- all common feedback systems on-board – no additional interface necessary
    - resolver
    - TTL encoder
    - Sinus/Cosinus 1 Vss
    - EnDAT, single- and multi-turn
    - Hiperface, single- and multi-turn
    - BiSS, single- and multi-turn

Variable motor interface

- brushless synchronous servomotors
- asynchronous servomotors
- asynchronous AC motors in servo operation with sensor feedback up to 6,000 rpm
- standard motors (DASM) in frequency mode up to 60,000 rpm
- linear motors (iron core and ironless)
- torque motors

Scalable output current

- high resolution measuring range spread at full current resolution
- advantages
  - A 6 A drive can run a 1 A motor.
  - flexible power balancing within a 2-channel module by utilising total device current:
    12 A = 2 x 6 A or 1 x 3 A + 1 x 9 A
  - minimum type variation, minimum inventory
  - device-specific factory setting, afterwards automatic application scaling via motor parameters

Active DC-Link

- DC-Link automatically connected only for regenerative energy flow
- short-circuit-proof DC-Link connection
- distributed braking by using all connected braking resistors
- external chopper module for high regenerative energy

Digital inputs

- Number
  - 7 inputs per device
- Functions
  - limit switches pos./neg. enable
  - amplifier lock with stator short cut braking
  - capture (2 x)

Digital output (programmable functions)

- Number
  - 1 output per channel + 1 device output
- Functions
  - control of the mechanical brake
  - error messages regarding external dynamic emergency stop functionalities
  - ready for operation

Status display

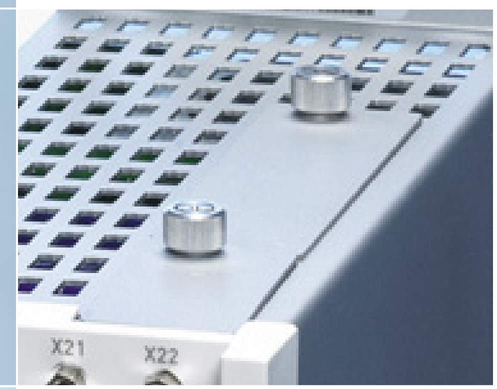
- Advantages
  - comfortable device diagnosis with output of the axis identifier
  - display of axis status and errors, also without EtherCAT communication
  - error messages as plain text

2 rows x 16 characters with backlight

Cooling concept

- max. operation temperature: 50 °C
- fanless operation up to 2 x 3 A or 1 x 6 A
- forced air cooling with regulated fan from 2 x 6 A/1 x 12 A
- internal air flow channel separated from electronic parts, thus no contamination
- cold plate
  - plane back plane for cold plate assembly
  - thereby realisation of protection class IP 65

AX5000 | System modules and accessories



System modules

- AX5001 | DC-Link expansion
  - for buffering of regenerated energy (brake energy)
  - short-circuit-proof
  - generation of 24 V auxiliary supply from intermediate circuit including power management
  - can be combined with multi-axis systems through AX-Bridge
  - EtherCAT interface for parameterisation and diagnosis
- AX5021 | Brake module
  - with internal 250 W braking resistor and active cooling
  - integrated brake chopper for external braking resistor up to 6 kW
  - EtherCAT interface for parameterisation and diagnosis
- AX5041 | Energy recovery module
  - mains inverter for feeding brake energy back into the supply network
  - EtherCAT interface for parameterisation and diagnosis

AX5000 | System modules and accessories

AX59xx | AX-Bridge quick connection system

- Connection module with power rail system for multi-axis systems, current carrying capacity up to 85 A

Supply module AX5901 with snap-on connection for the Servo Drive

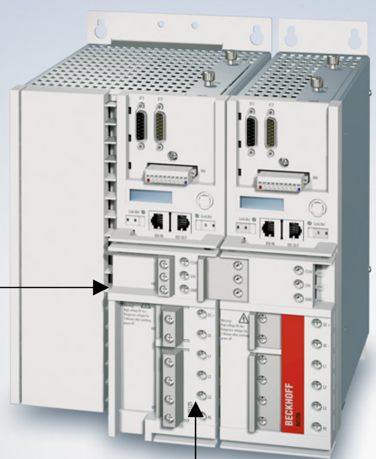
Power distribution module AX5911 with snap-on connection for power supply, DC-Link and control voltage

AX51xx | 1-channel Servo Drive

AX5118, AX5125 | Rated output current of 18 A and 25 A

NEW

Power supply module AX5901 with snap-on connection for Servo Drive

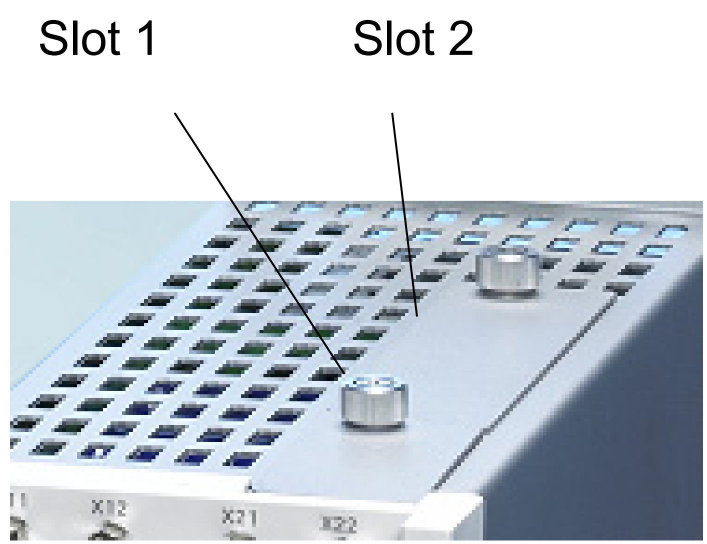


Power distribution module AX5911 with snap-on connection for power supply, DC-Link and control voltage

Power distribution module AX5921 for AX5118, AX5125 with snap-on connection

Accessories

- optional slots
  - safety for Motion Control (slot 1)
  - additional encoder interface, e.g. SSI (slot 2)
  - I/Os (capture, etc.) (slot 2)
  - customer-specific cards (slot 2)



Optimised for EtherCAT

- EtherCAT – the optimum drive bus
  - short cycle time
  - synchronicity
  - simultaneity
- Ethernet right down to the drives
- high-precision system synchronising through distributed clocks
- high-speed capture with time stamp, e.g. for print mark control
- ultra high-speed communication with update times of:
  - 100 axes in 100 µs
  - 1,000 distributed I/Os in 30 µs

Optimised for EtherCAT

- high-speed control algorithms
  - current controller with cycle times down to 31.25 µs for highly dynamic regulation of ironless linear motors
  - speed controller 125 µs
  - position controller 125 µs
- transparent line topology with flexible branches
- simple system wiring using standard patch cable
- simple diagnosis
  - breaking point detection and localisation
  - Protocol, physical characteristics and topology enable individual quality monitoring of all transmission links.

Safety integrated with TwinSAFE

Option cards for various safety categories

- AX5801 | Restart lock
  - personal protection against inadvertent restart of the drive axis
    - meets EN 954-1
    - STO Safe Torque Off (IEC 61800-5-2)
    - SS1 Safe Stop 1 (IEC 61800-5-2)
    - control through digital input
    - Mains voltage and motor line remain connected.

Safety integrated with TwinSAFE

Option cards for various safety categories

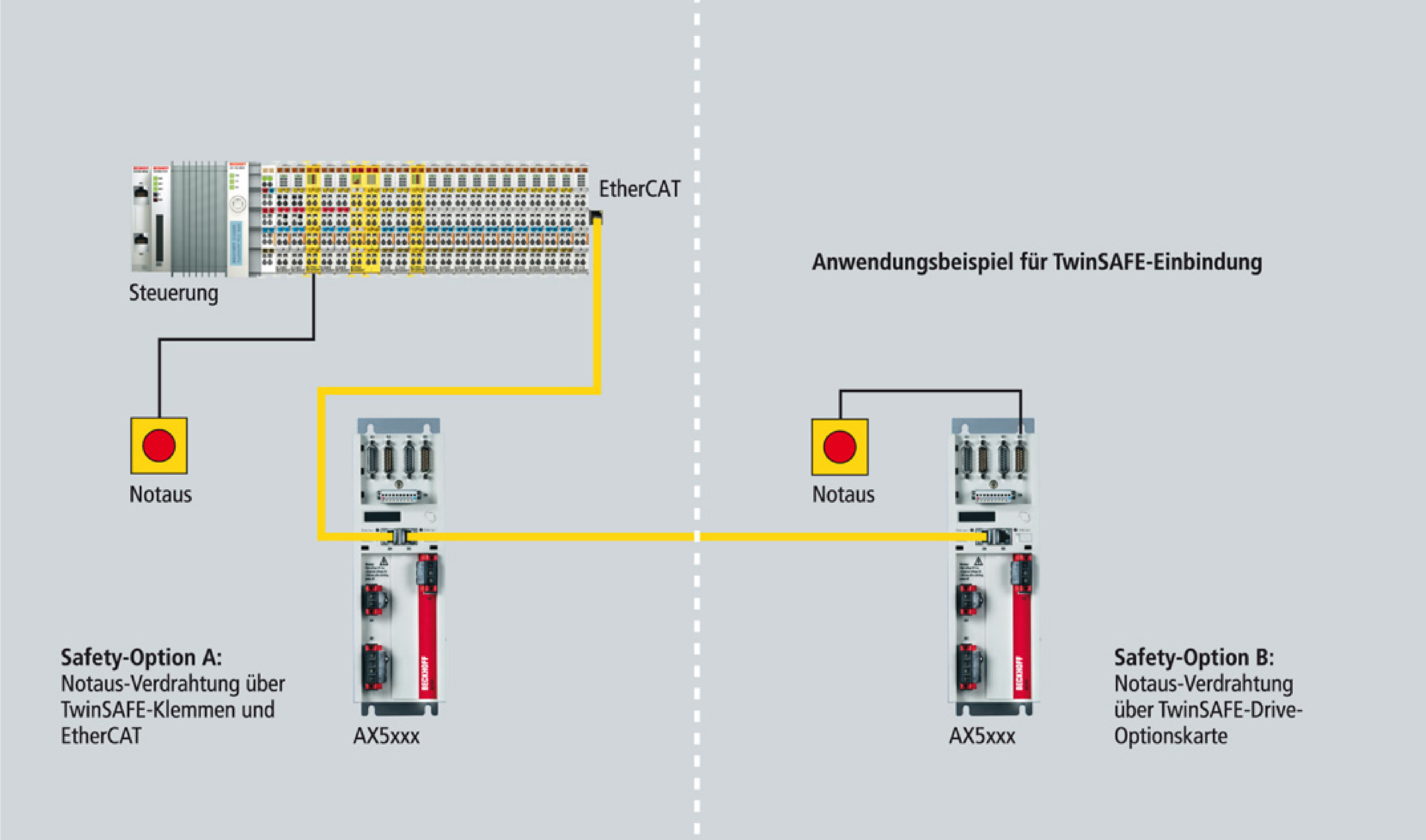
- AX5805 | TwinSAFE drive option cards
  - meets safety category 3 (EN 954)
  - realisation of the following functions, acc. to IEC 61800-5-2
    - STO Safe Torque Off
    - SS1 Safe Stop 1
    - SS2 Safe Stop 2
    - SOS Safe Operating Stop
    - SLA Safely Limited Accel.
    - SLS Safely Limited Speed
    - SSR Safe Speed Range
    - SLT Safely Limited Torque
    - STR Safe Torque Range
    - SLP Safely Limited Position
    - SLI Safely Limited Increment
    - SDI Safe Direction
    - SCA Safe CAM

Safety integrated with TwinSAFE

- Safety over EtherCAT
  - The protocol developed according to IEC 61508 can be transferred via EtherCAT.
  - Fieldbus gateways enable the drives to be integrated into traditional fieldbus systems: PROFIBUS, DeviceNet, CANopen, SERCOS interface or Ethernet.
  - The integration into the TwinSAFE product family allows the realisation of safety technology without sophisticated safety control.

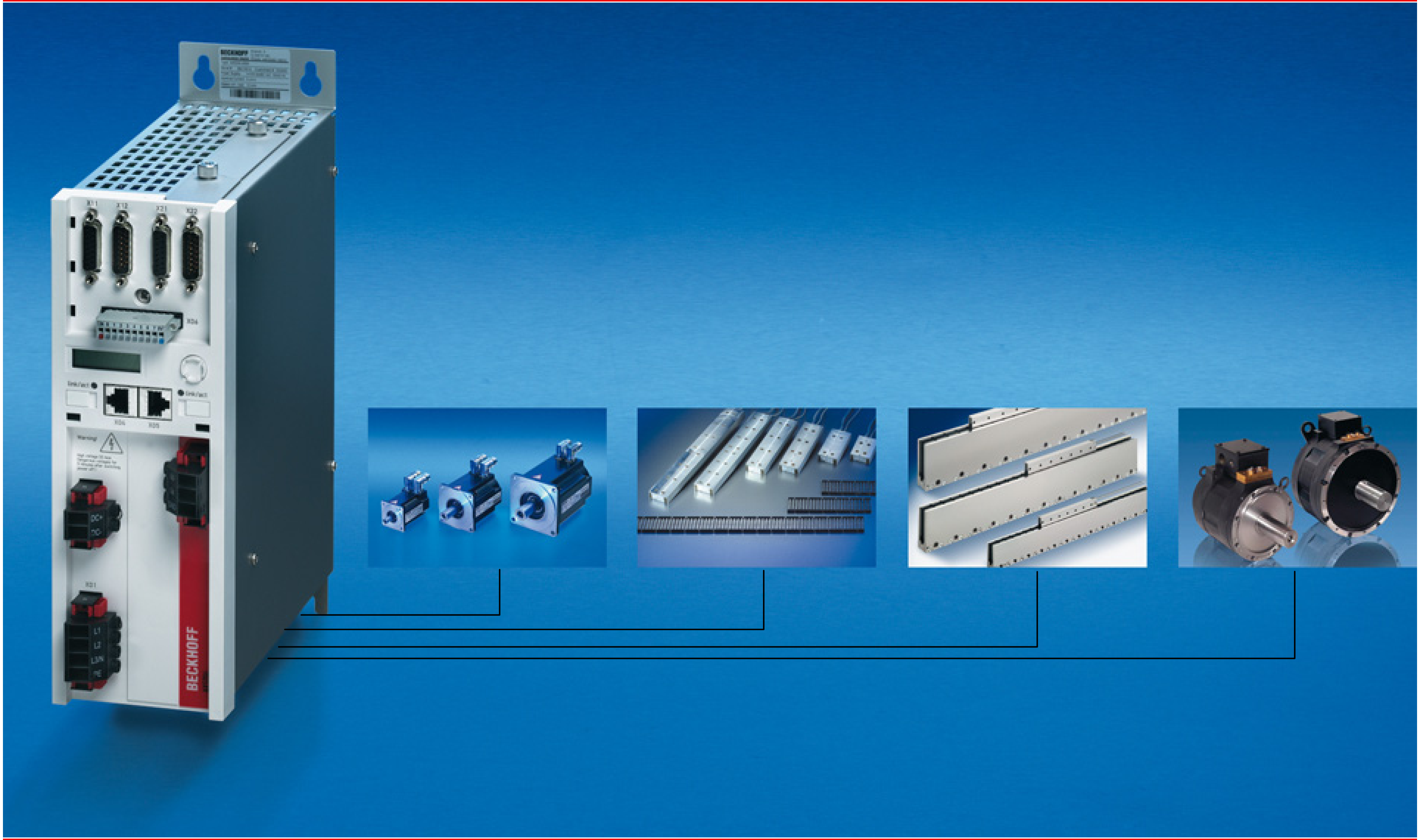
Safety over EtherCAT®

Emergency stop wiring via TwinSAFE and EtherCAT



AX5000

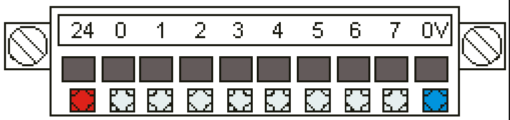
Variable motor interface



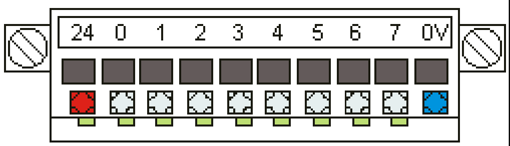
X06: Digitale I/Os

X06: Digitale I/Os

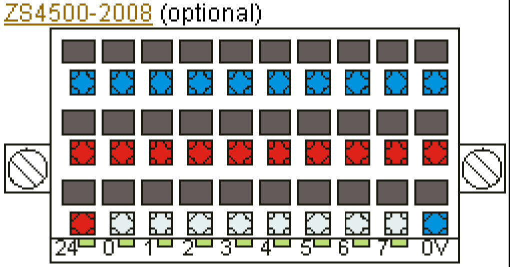
I/O-Steckverbinder ohne LEDs
ZS4500-2006



I/O-Steckverbinder mit LEDs
ZS4500-2007 (optional)



ZS4500-2008 (optional)



Terminal	Signal	Factory setting	
		AX51xx	AX52xx
24	Output 24V DC !!!		
0	Input 1	Enable	Enable Achse 1
1	Input 2	P-Stop	
2	Input 3	N-Stop	
3	Input 4		
4	Input 5		
5	Input 6	Capture	Capture Axis 1
6	Input 7	Capture	Capture Axis 2
7	Input 8 or Output	Error	
0V	Ground/DC 0V		

Multi-Feedback-Interface

Pin	Signal: high resolution Feedback		
	EnDAT/Biss	Hiperface	Sinus/ Cosinus 1Vss
1	Cos B+	Cos B+	Cos B+
2	GND UP_5V	GND UP_9V	GND UP_5V
3	SIN A+	SIN A+	SIN A+
4	UP_5V	n.c.	UP_5V
5	DX+ (Data)	DX+ (Data)	n.c.
6	n.c.	UP_9V	n.c.
7	REF N-	UP_9V	REF N-
8	CLK+ (Clock)	n.c.	n.c.
9	REFCOS B-	REFCOS B-	REFCOS B-
10	GND_Sense	n.c.	GND_Sense
11	REFSIN A-	REFSIN A-	REFSIN A-
12	UP_5V_Sense	n.c.	UP_5V_Sense
13	DX- (Data)	DX- (Data)	n.c.
14	N+	N+	N+
15	CLK- (Clock)	n.c.	n.c.

X03: 24 VDC Supply

- Control voltage supply by connector X3. The 24V supply has two lines, in this way brake and control supply can be handled separately. In case of unused Up, please connect Up-Us. By connecting motor holding brake , please pay attention to voltage tolerance.

Connector	Signal
Up	24 VDC -0 / +15% : Peripherie (for example, Motor break voltage)
Us	24 VDC +/-15% : control unit voltage
GND	GND

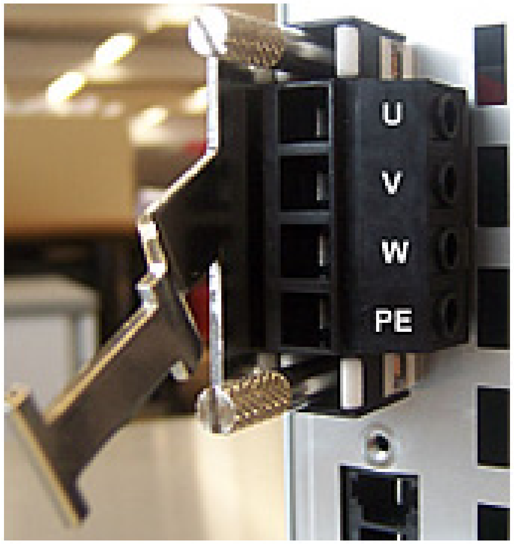
AX5xxx

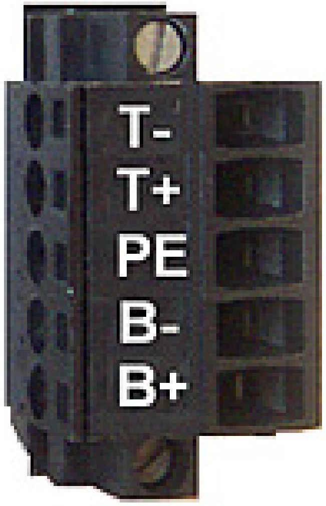
Main power

- X01: Power input-
- from single phase 100 VAC up to 3-phase 480 VAC. In case of single phase supply connect phase to L1 and N to L3.

Terminal	Connection	
	3-phase	1-phase
L1	Phase L1	Phase L1
L2	Phase L2	n.c.
L3/ N	Phase L3	Neutral wire
PE	Protective earth	Protective earth

X13 (A), X23 (B): Motor terminal

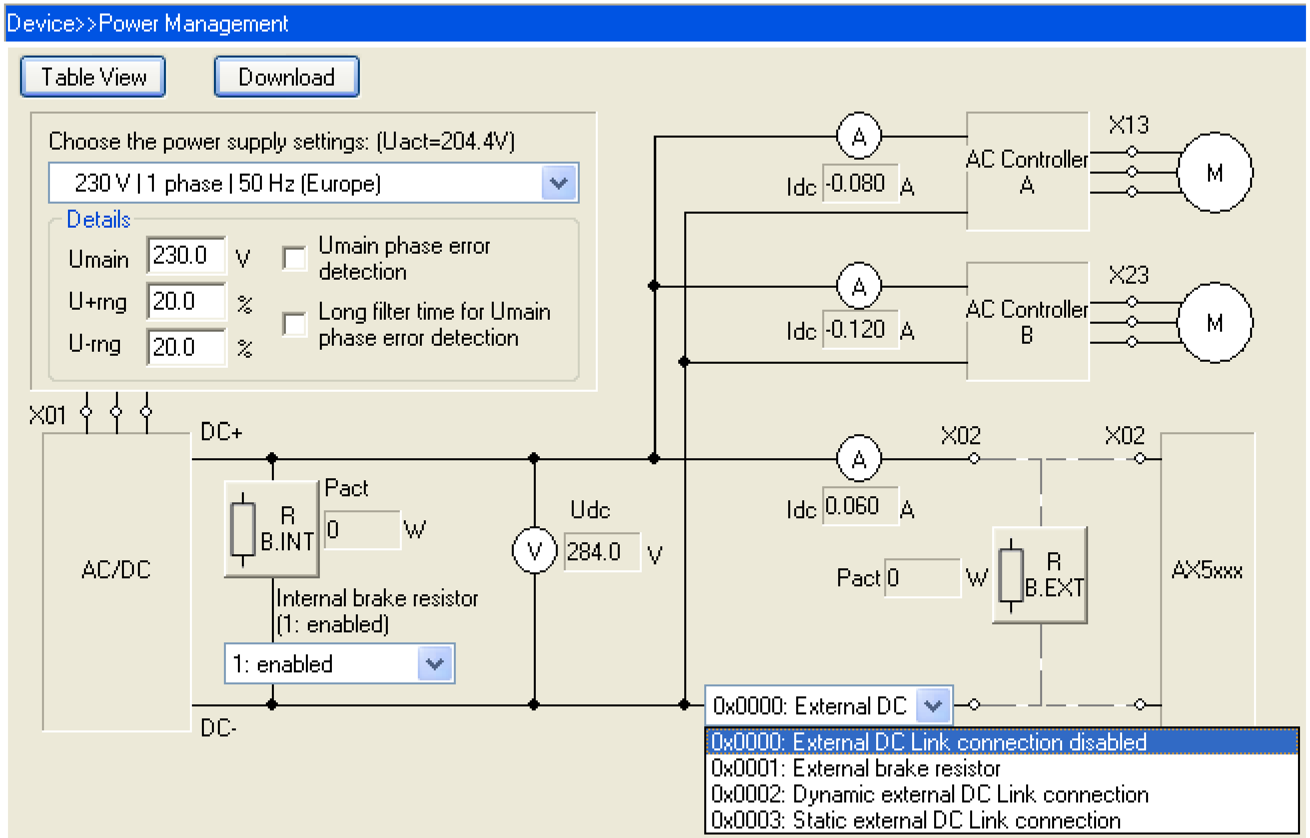




Terminal	Signal
U	Motor U
V	Motor V
W	Motor W
PE	Protect earth
Shield	Shield

X02: DC Link Bus

- By terminal X2, DC bus coupling or direct DC power supply is possible.



Active DC link

- DC link automatically connected only for regenerative energy flow
- short circuit proof DC link connection
- distributed braking by using all connected braking resistors
- external chopper module for high regenerative energy *in prep.*

Variable cooling concept

- max. operation temperature: 50°C
- fanless operation up to 2 x 3 A or 1 x 6 A
- temperature controlled forced cooling, starting at 2 x 6 A or 1 x 12 A
- internal air flow channel separated from electronic parts, by thus no contamination
- Cold Plate *in prep.*
  - plane back plane for cold plate assembly
  - thereby realisation of protection class IP 65

System modules

- AX5001 | DC link expansion *in prep.*
- brake energy can be stored and reused for
- next acceleration process
- Short circuit proof
- can be combined with multi-axis systems through AX-Bridge
- EtherCAT interface for parameterisation and diagnosis

- AX5021 | Brake module *in prep.*
- with internal 250 W braking resistor and active cooling
- integrated brake chopper for external braking resistor up to 6 kW
- EtherCAT interface for parameterisation and diagnosis

- AX5041 | Energy recovery module *in prep.*
- mains inverter for feeding brake energy back into the supply network
- EtherCAT interface for parameterisation and diagnosis

Status display

2 rows x 16 characters with backlight

- Advantages:
- comfortable device diagnosis and maintenance
- axis identifier for two channel devices
- display of axis status and errors, also without EtherCAT communication
- error messages as plain text

SERCOS-Profile for servo drives

- To bring the motion control to an existing standard the SERCOS – Profile IEC 61491 was implemented.
- This offer the user an easy and optimal setup.

- Sercos S- and P- Parameter:

- This SERCOS profile differs two main groups of parameter.

- The standard parameter e.g. :
- S-0-0001 ⇨ NC ⇨ Cycle time (TNcyc)

- Product specific parameter e.g. :
- P-0-0001 Switching frequency of the IGBT module

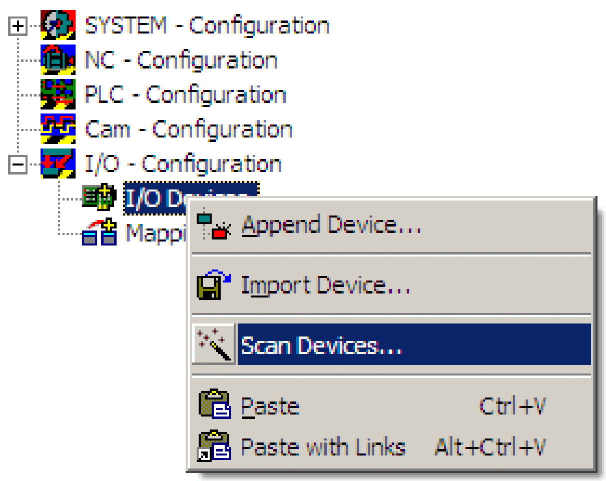
The storage concept

- Compared to the AX2000 the modified drive parameters are not stored inside the Drive, there is only the default setup as part of the Drive firmware.
- e.g. by changing the parameter „Motor“ , the new setup has to be added to the „Startup List“.
- After „saving“ the „Startup List“ and “Activate configuration”, it becomes a part of the System Manager file *.tsm and will be handled from the system manager.

Drives linking – First motion

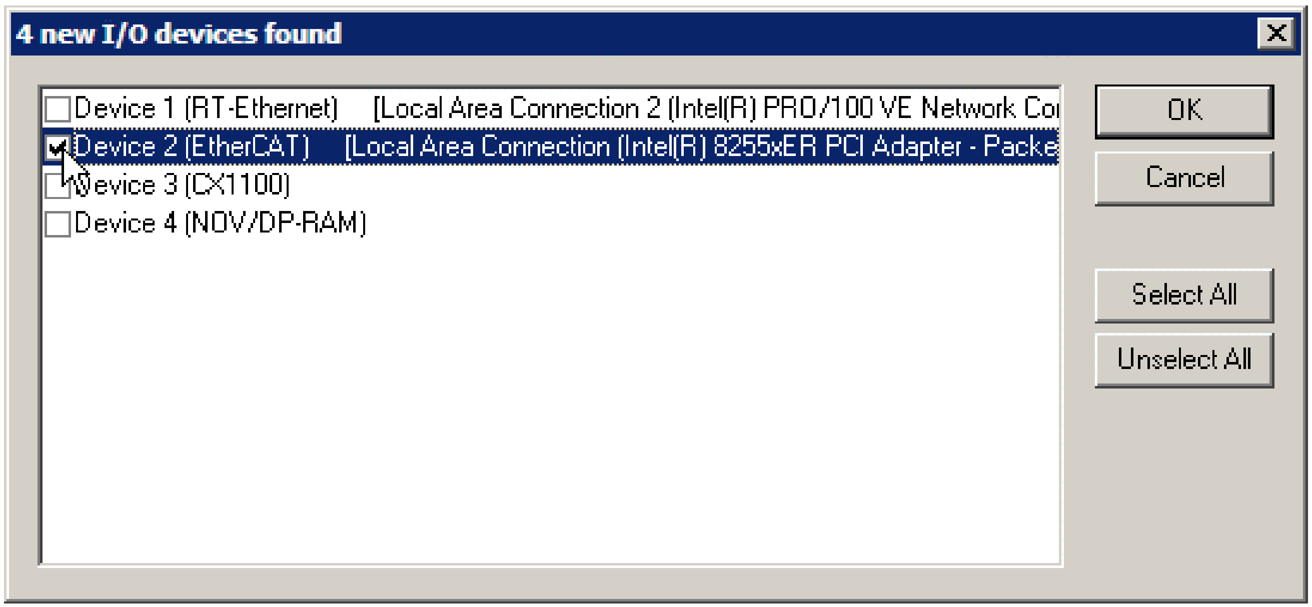
- Requirements:
  - Control voltage: 24 VDC
  - EtherCAT- master connection
  - TwinCAT Config Mode

- The first step is to scan the bus for EtherCAT devices:



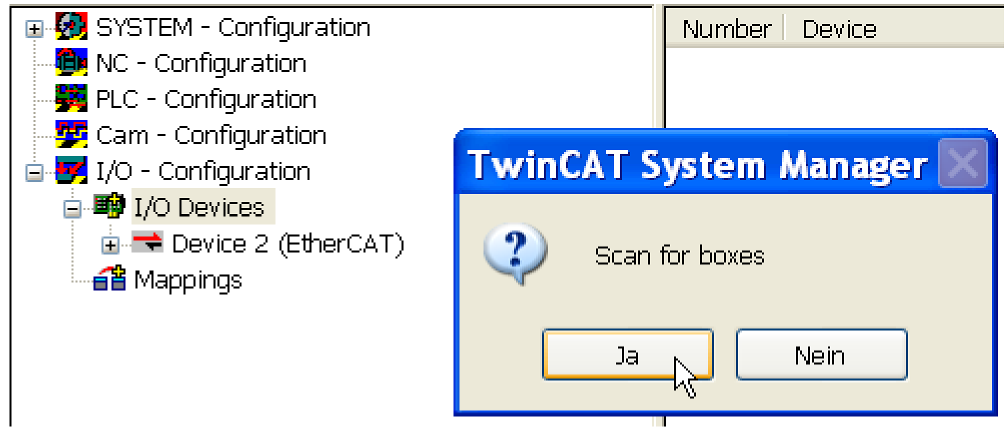
Drives linking – First motion

Select the EtherCAT-Interface

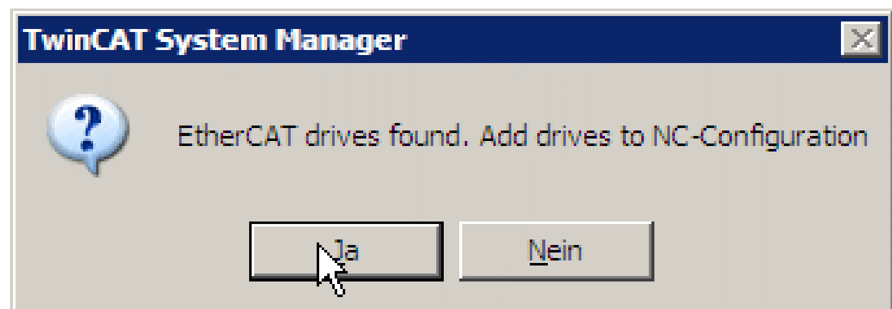


Drives linking – First motion

- Scan for boxes



- Add the Drives to the NC



- No “Free Run”

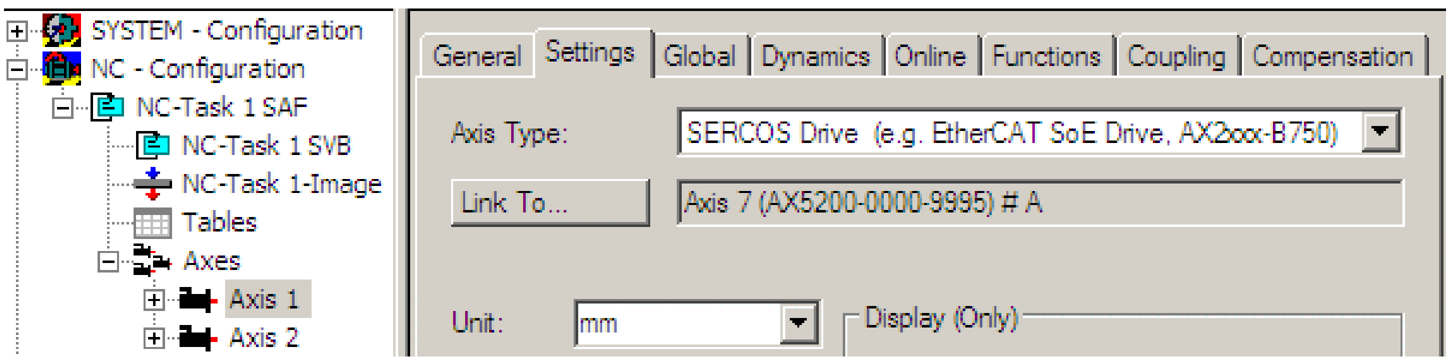
Ja = Yes

Nein = No

Drives linking – First motion

- Tap "Settings"

- All detected Axis are displayed under NC- Configuration.
- The AX5000 is shown as “SERCOS Drive”.
- The communication profile is SoE (Sercos over EtherCAT).

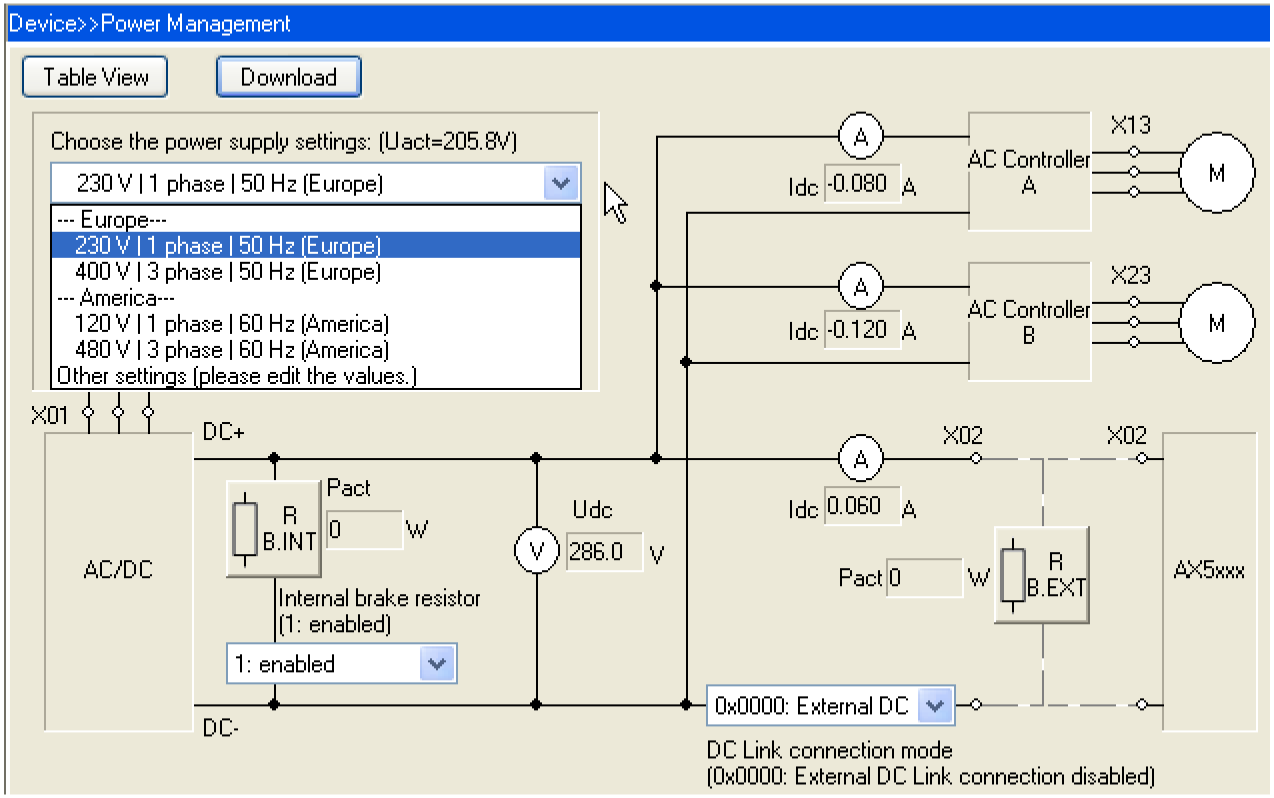


Drives linking – First motion

- The TCDriveManager.
- The TCDriveManager gives all the resources to handle the drive setup and the parameter. By the menu tree you have access to device and drive data's. In case of twin axis like AX52xx axis data's selectable as canal A and channel B.

Setup in the „Power Management“
Umain
U+rng
U-rng
Disable „Phase Error Detection“
Press „Download“

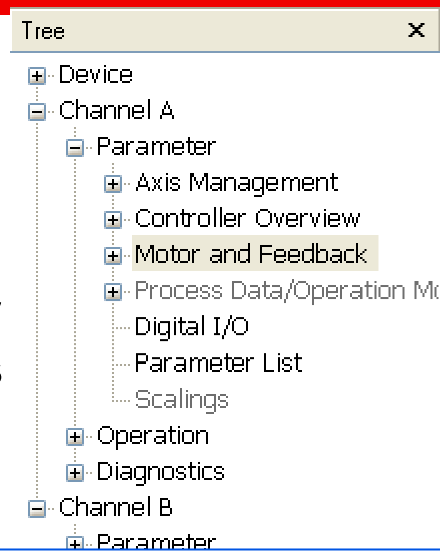
Only needed in case of manual Power supply setup.

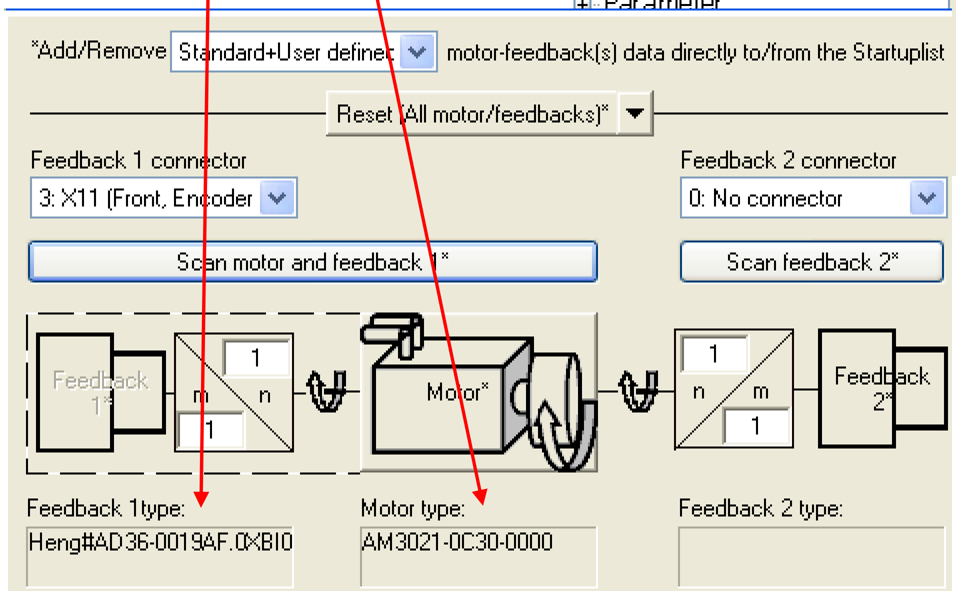


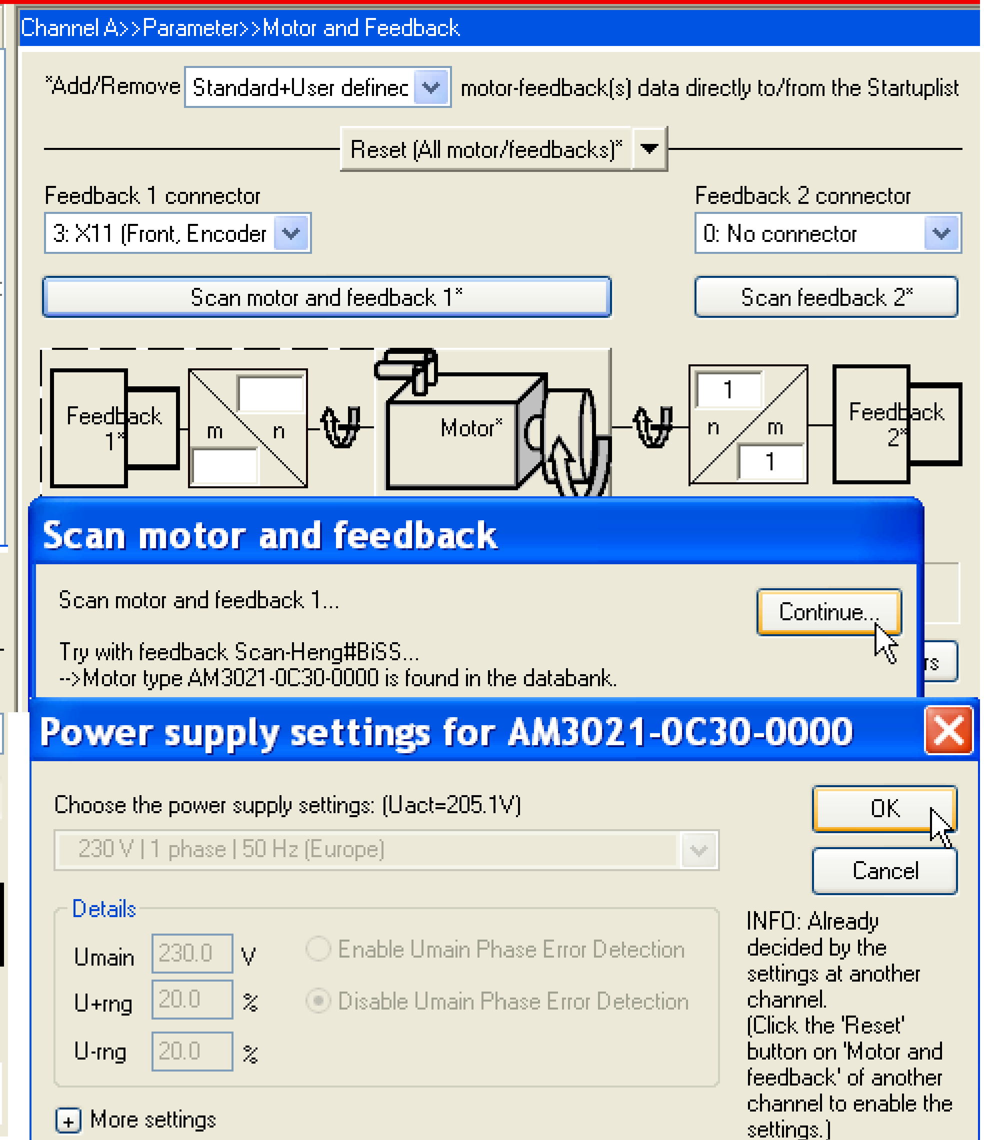
Drives linking – First motion

Scan motor and feedback in mode Pre-Op.

After this steps motor and Feedback type is shown

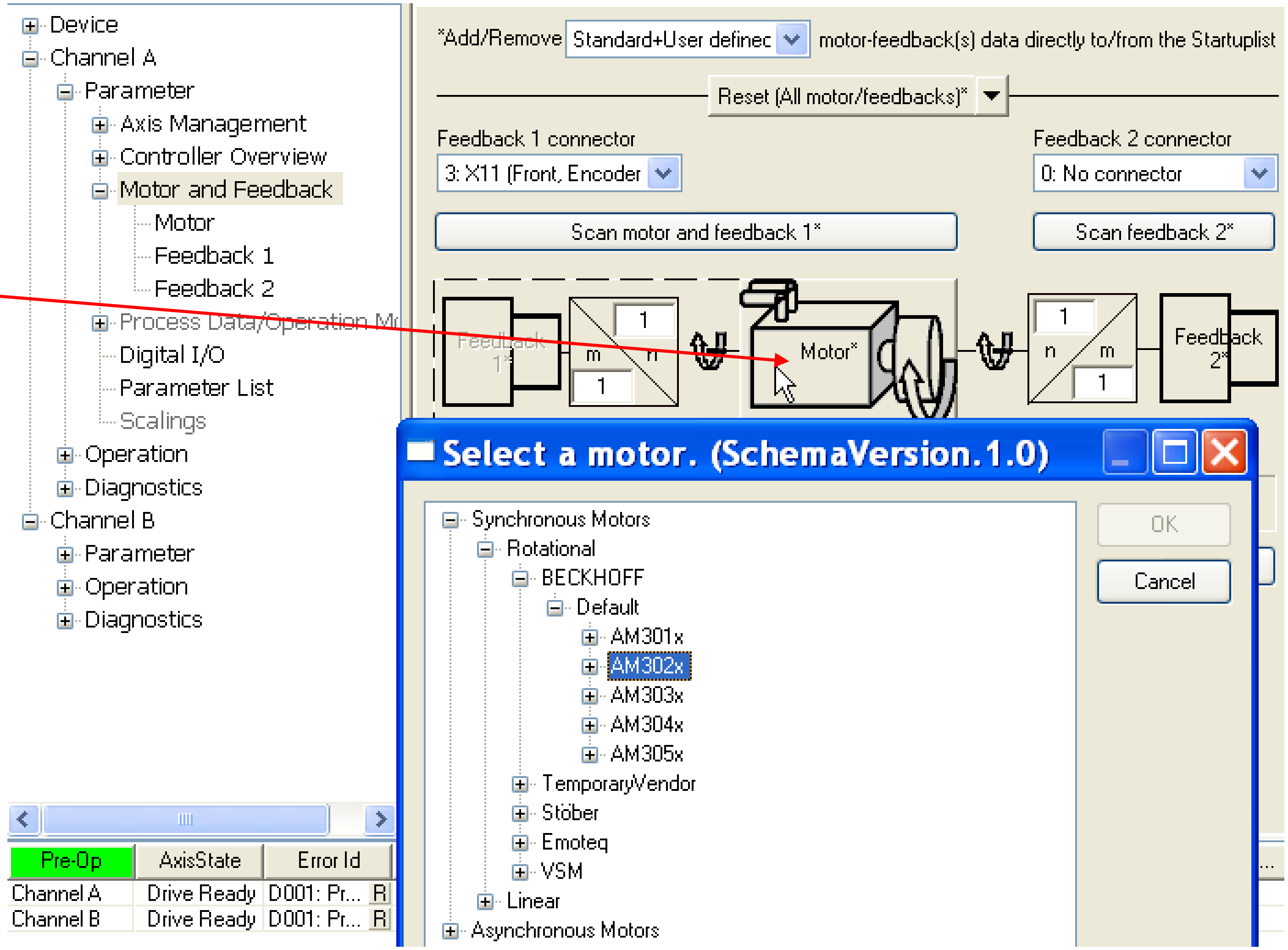






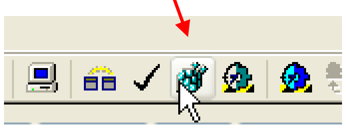
Drives linking – First motion

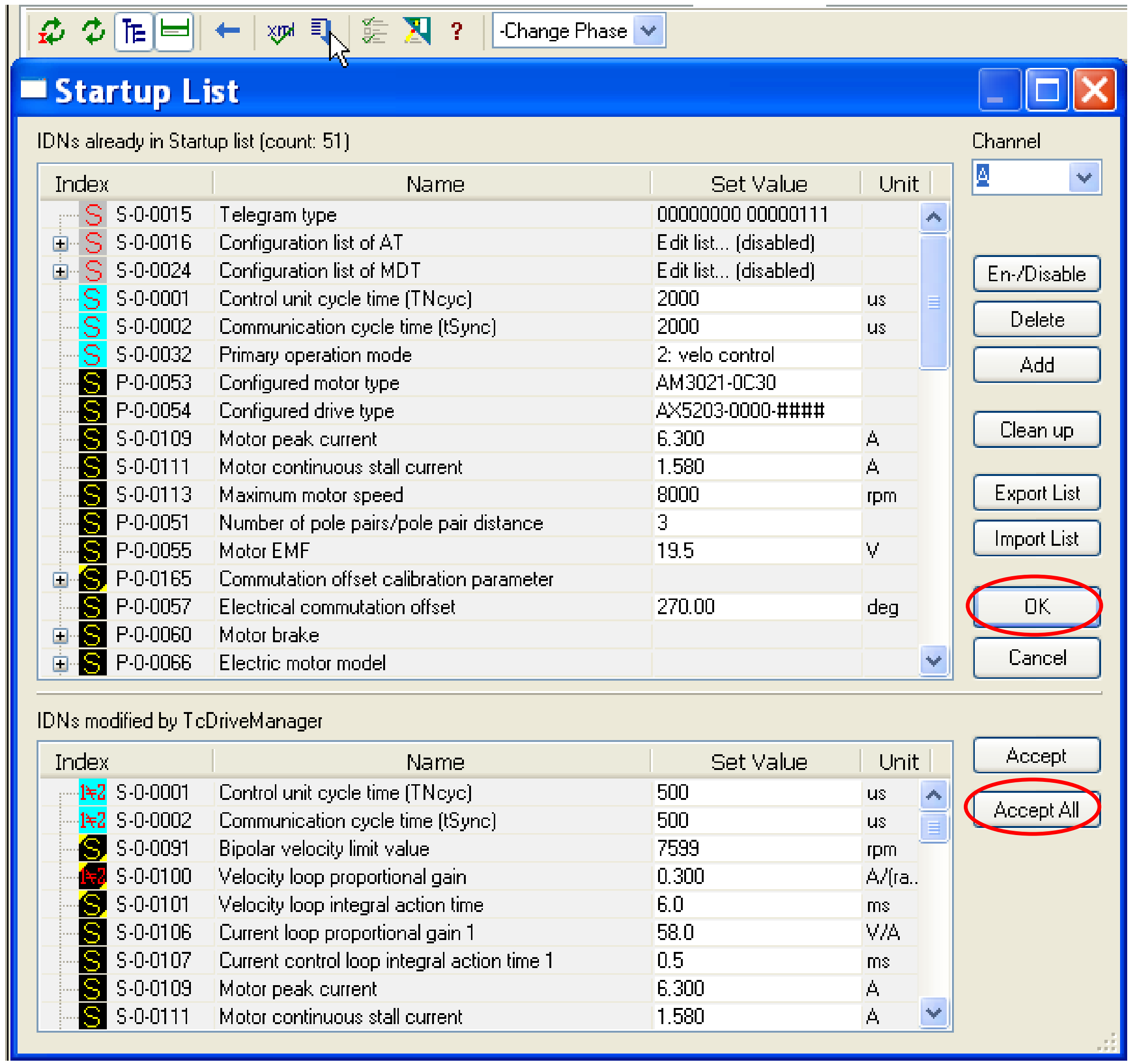
- Manual „Motor and Feedback“ selection from the database



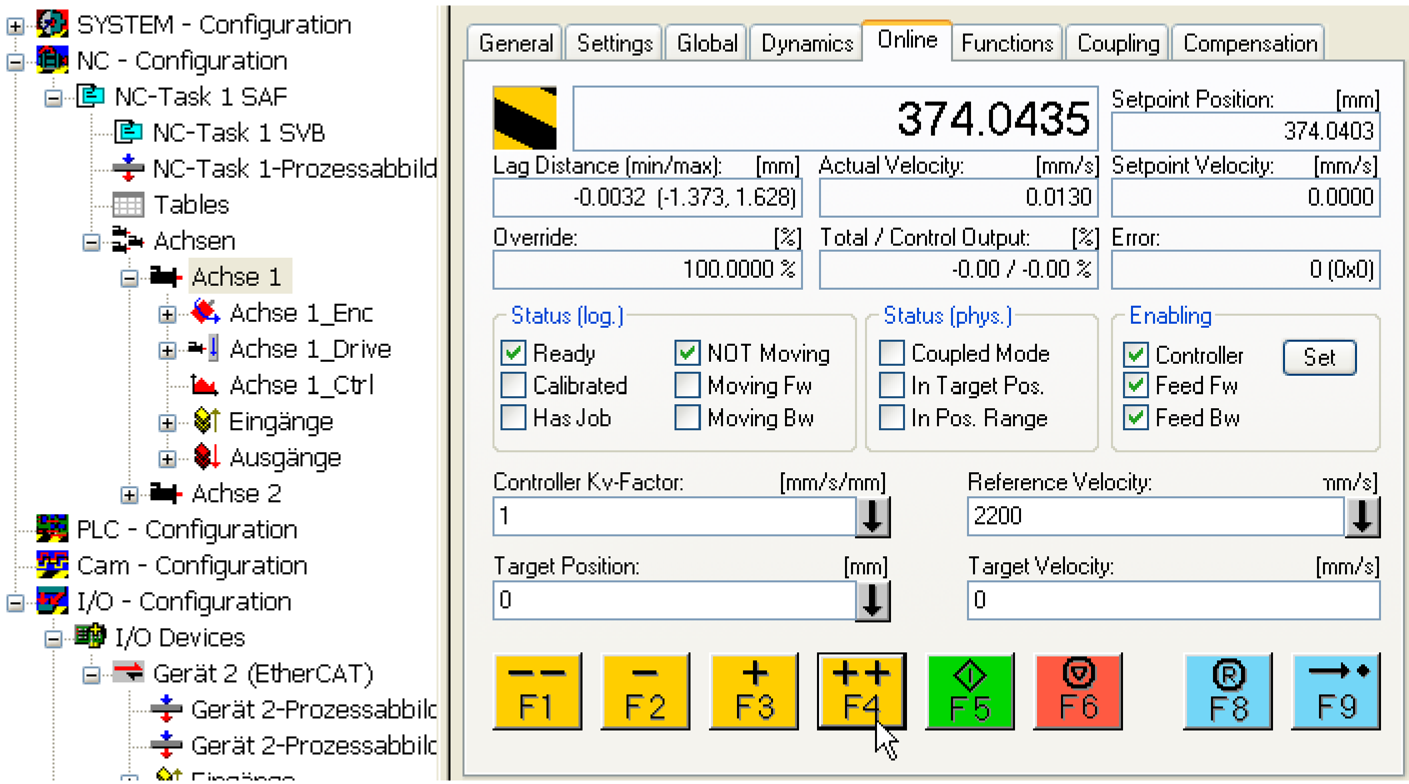
Drives linking – First motion

- The upper part of the Startup List shows default and changed parameters / IDNs.
- The lower part shows all the IDNs modified by the TCDriveManager.
- Add this by „Accept All“ and “OK” to the Startup List.
- And “Activate configuration”



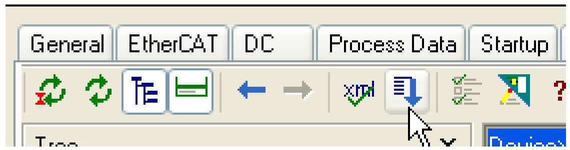


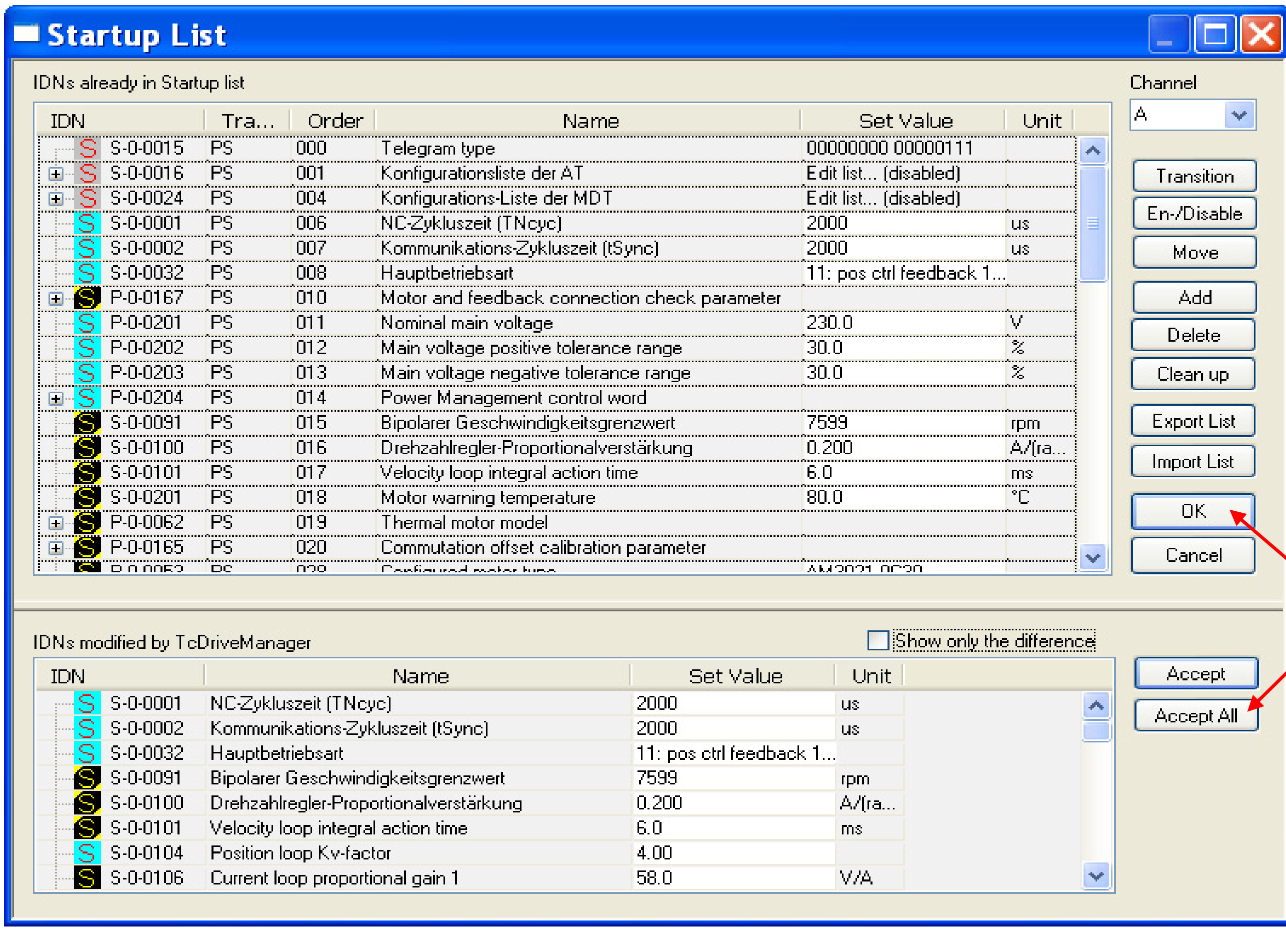
Drives linking – First motion



Now the first move is possible !

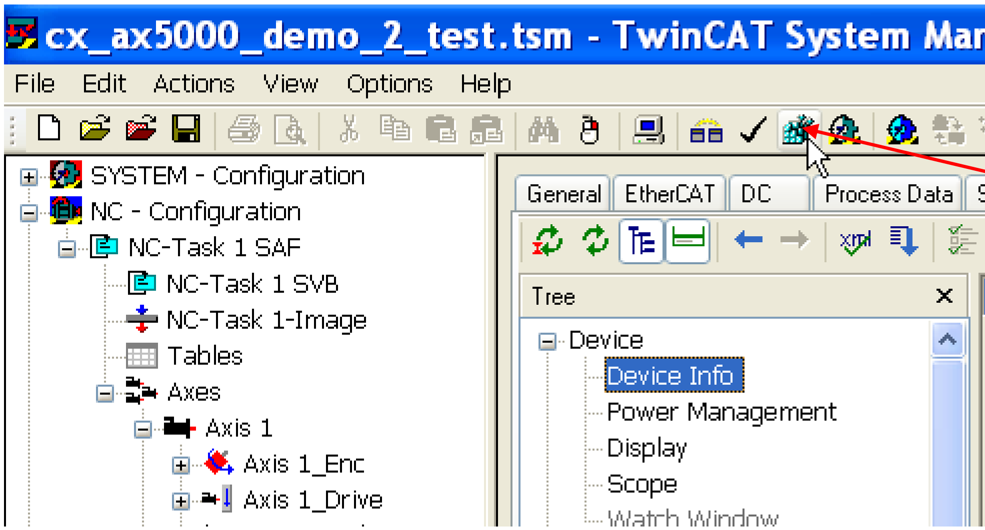
Setup saving in three steps: 1. in „Startup List“



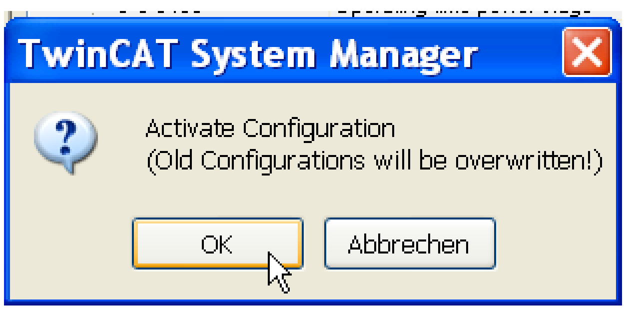


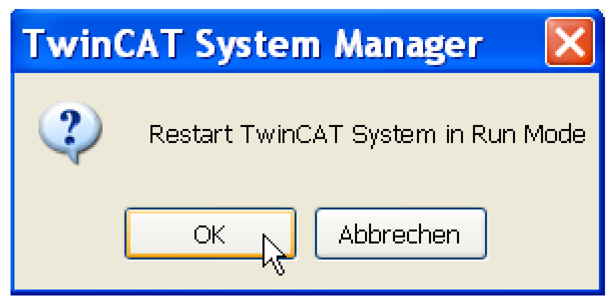
1. “Accept All”
2. “OK”

Setup saving in three steps: 2.„Activate configuration“

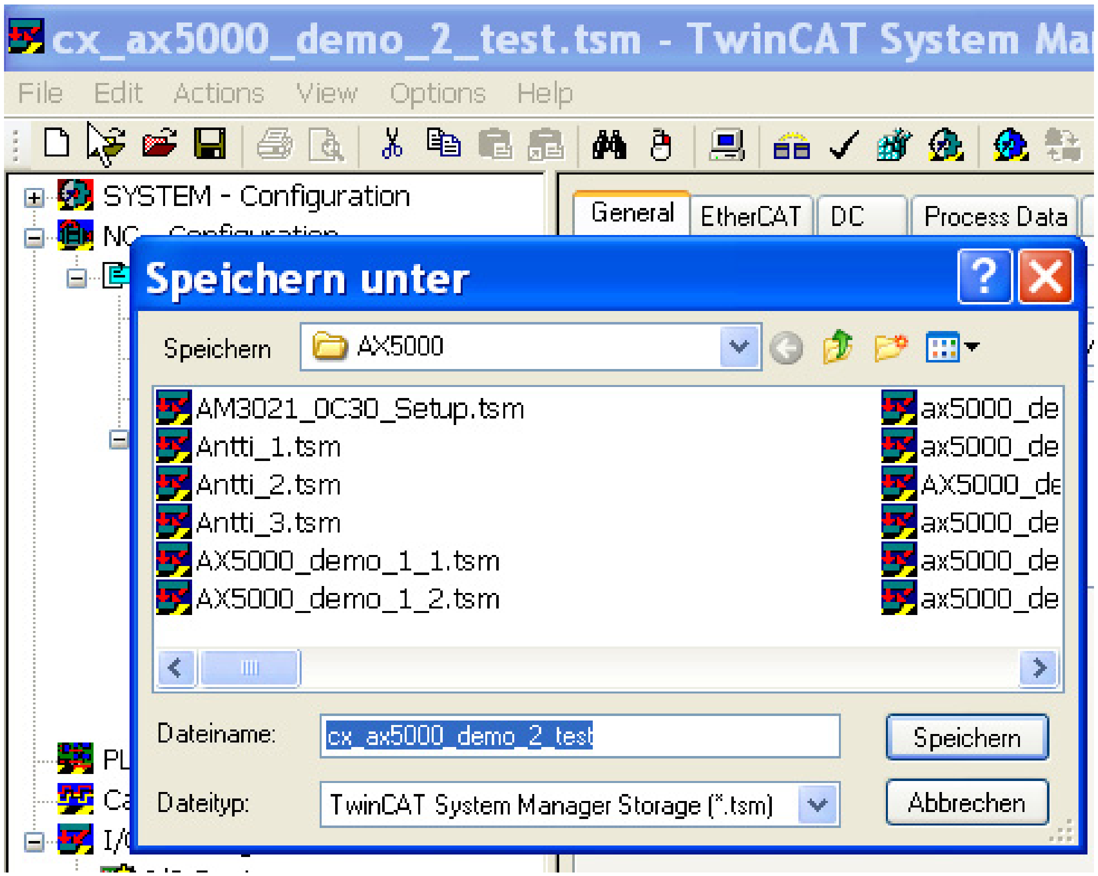


Activate configuration

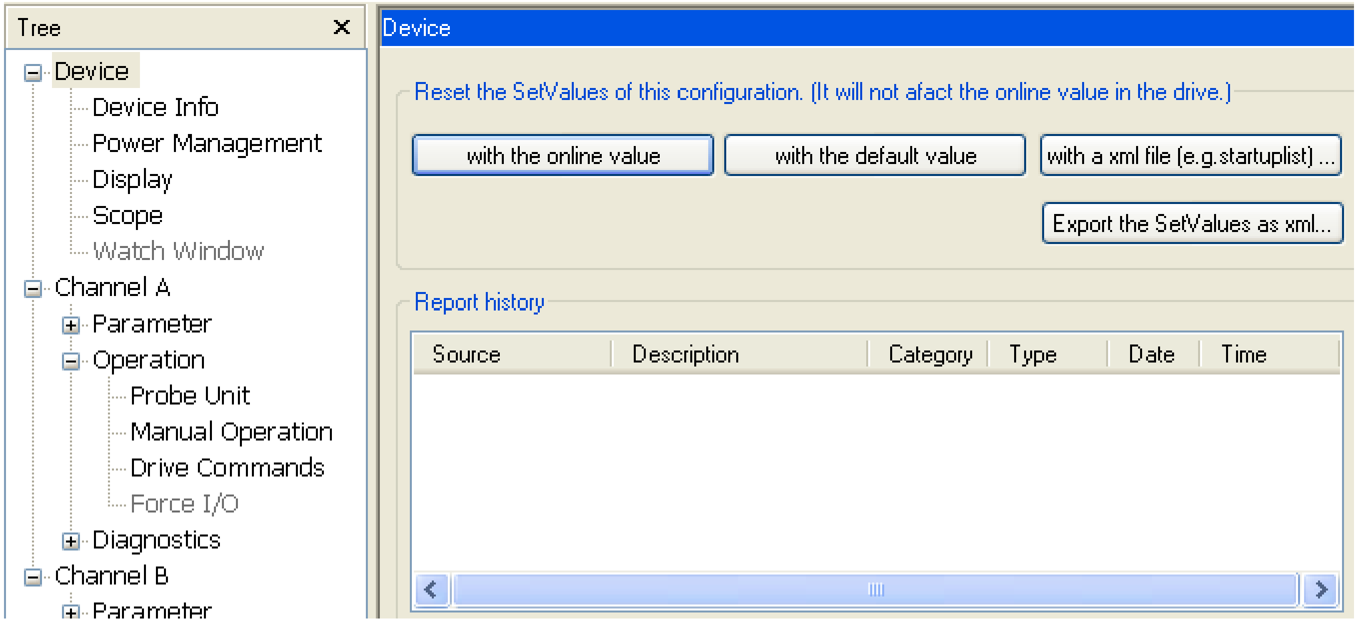




Setup saving in three steps: 3. in TSM file

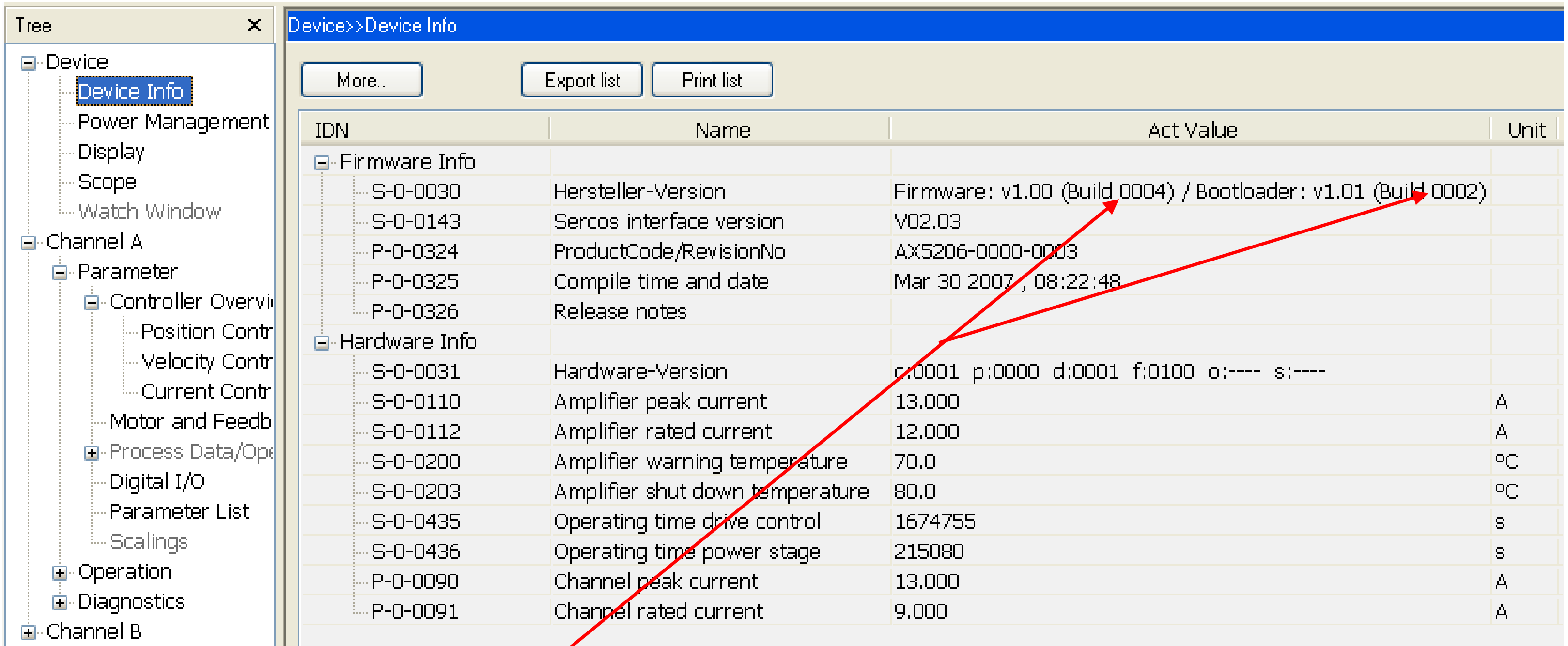


Device- Function



Parameter handling of the AX5000

Device- Info

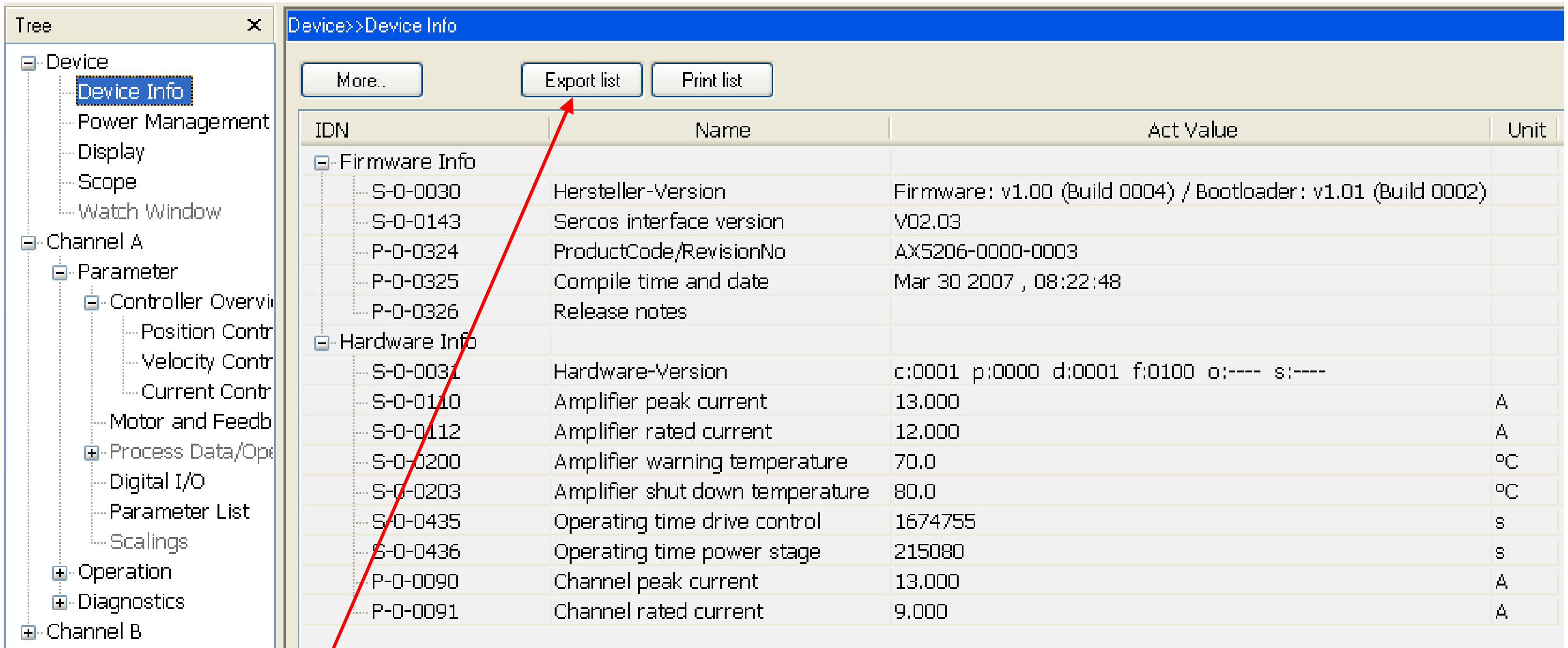


Drive "Firmware" - 0 = released 9 = test version

Hardware Version c= control - board; p = power-; d = driver-; f = frond-; o = option; s = safety

AX5xxx

Device- Info

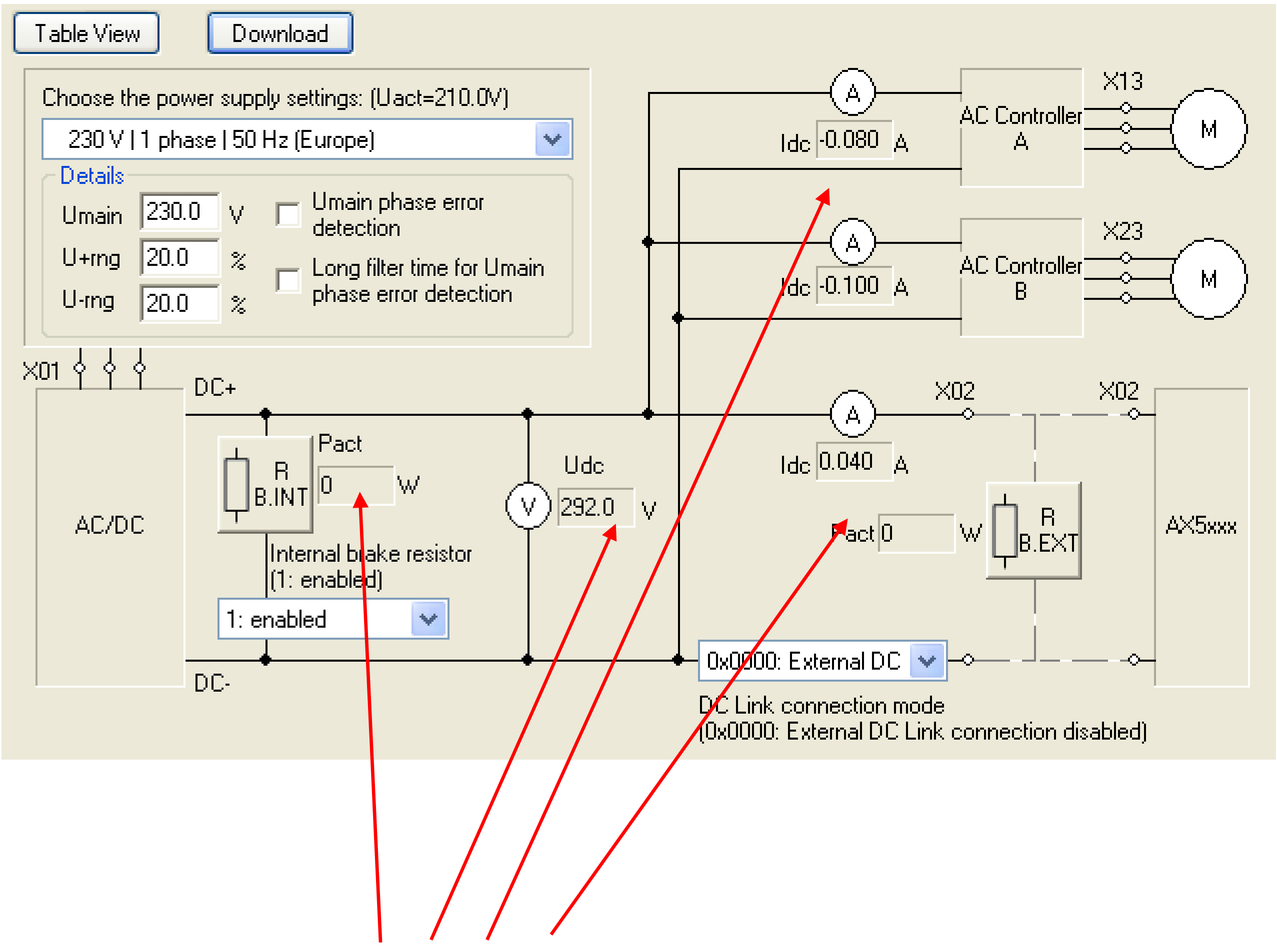


Export function for device info, **please save it for each drive!**

Device- Info

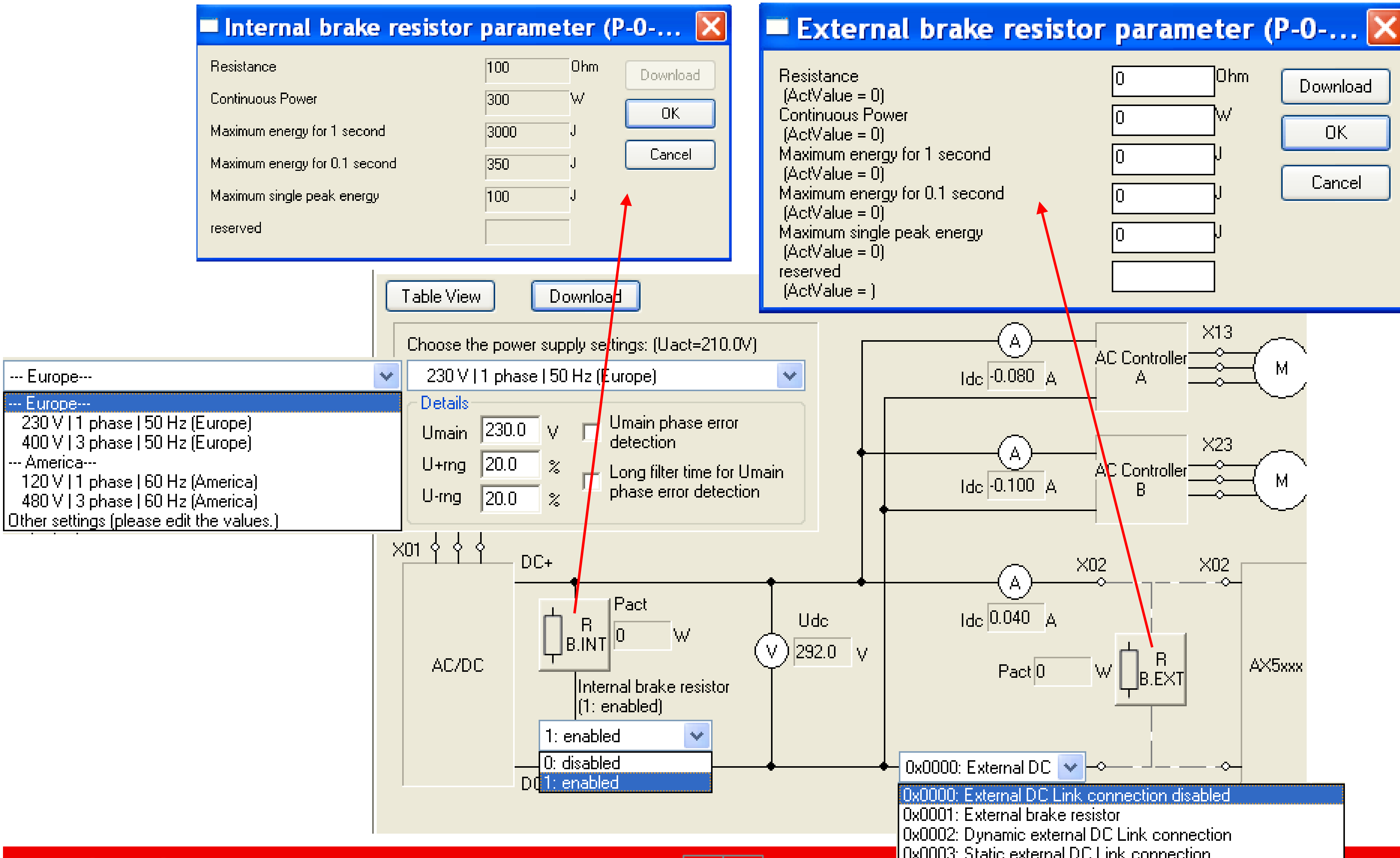
IDN;Name;ActValue;SetValue;Unit
Firmware Info;;;;
S-0-0030;Hersteller-Version;Firmware: v1.01 (Build 0002) / Bootloader: v1.01 (Build 0002);;
S-0-0143;Sercos interface version;V02.03;;
P-0-0324;ProductCode/RevisionNo;AX5203-0000-0006;;
P-0-0325;Compile time and date;Sep 27 2007 , 12:36:48;;
P-0-0326;Release notes;;;
Hardware Info;;;;
S-0-0031;Hardware-Version;c:0001 p:0001 d:0001 f:0100 o:---- s:----;;
S-0-0110;Amplifier peak current;12.000;;A
S-0-0112;Amplifier rated current;6.000;;A
S-0-0200;Amplifier warning temperature;70.0;;°C
S-0-0203;Amplifier shut down temperature;80.0;;°C
S-0-0435;Operating time drive control;854046;;s
S-0-0436;Operating time power stage;45888;;s
P-0-0090;Channel peak current;12.000;;A
P-0-0091;Channel rated current;6.000;;A

Power- Management



Different information of actual values

Power- Management



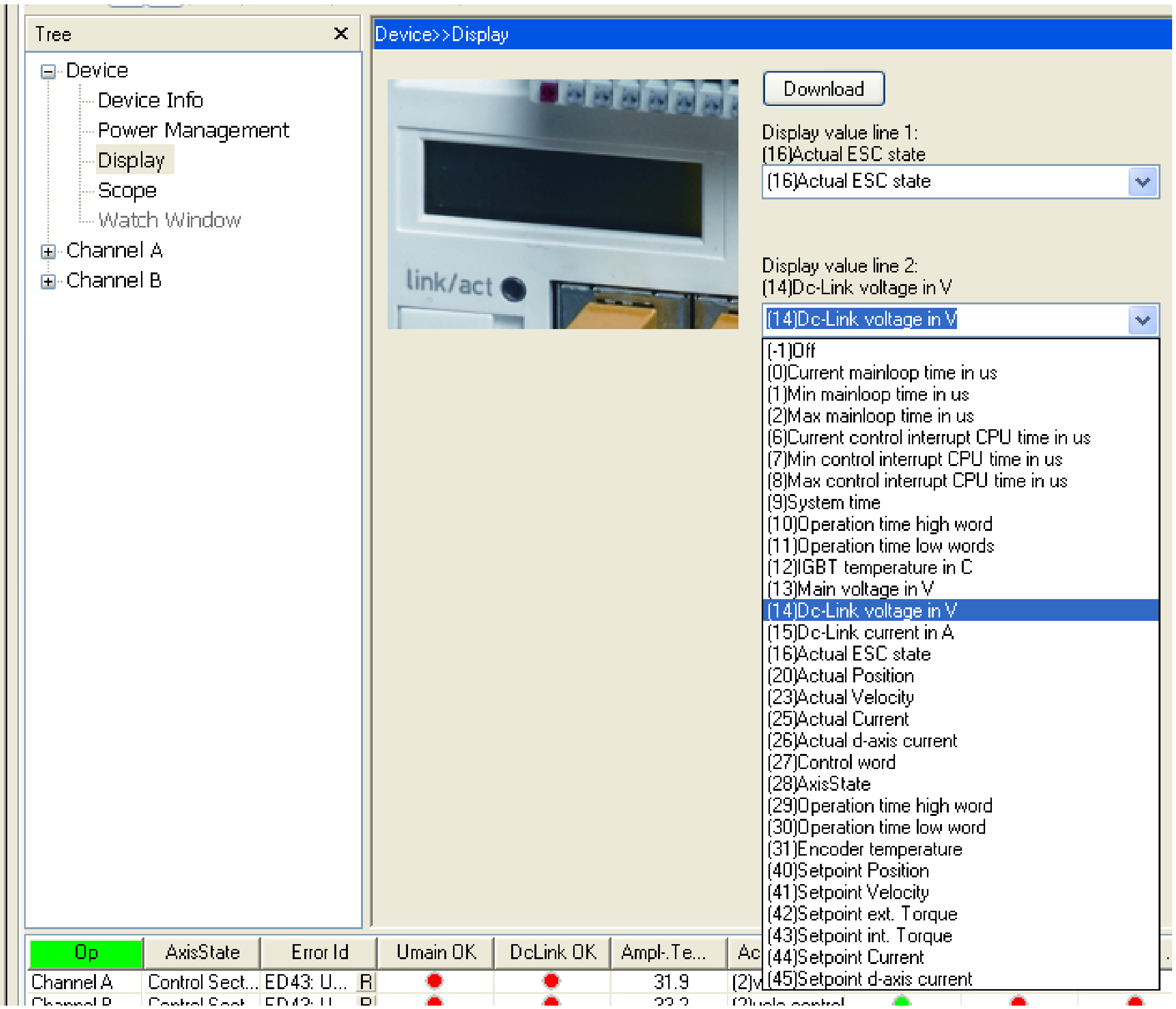
Power- Management in „Table View“

Device>>Power Management

Graphic View   Download

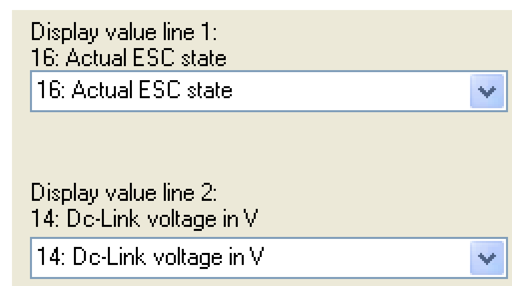
IDN	Name	Act Value	Set Value	Unit
S-0-0380	DC bus voltage	280.0		V
S-0-0381	DC bus current	-0.080		A
P-0-0200	Actual main voltage peak value	284.0		V
P-0-0201	Nominal main voltage	230.0	230.0	V
P-0-0202	Main voltage positive tolerance range	30.0	30.0	%
P-0-0203	Main voltage negative tolerance range	30.0	30.0	%
P-0-0204	Power Management control word			
P-0-0205	Power Management status word			
P-0-0206	Power management switching thresholds			
P-0-0207	Internal brake resistor parameter			
P-0-0208	External brake resistor parameter			
P-0-0209	Actual power internal brake resitor	0		W
P-0-0210	Actual power external brake resitor	0		W
P-0-0211	Warning level: Actual power internal brake resitor	100	100	W
P-0-0212	Warning level: Actual power external brake resitor	500	500	W
P-0-0213	External DC link current	0.040		A
P-0-0214	DC Link connection mode	0x0000: External DC Li...	0x0000: External DC Li...	
P-0-0215	Actual Periphery Voltage	26.688		V

Display



Different display modes are possible

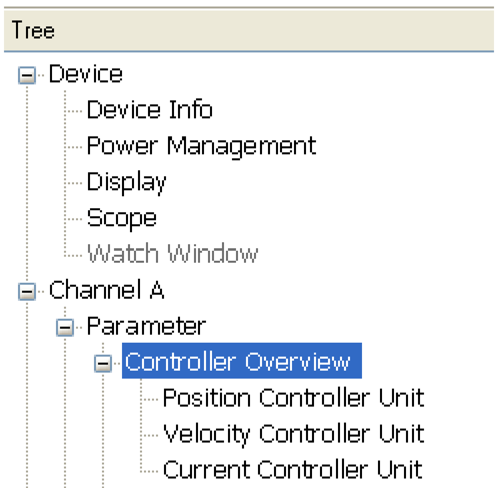
Default setting:



Parameter

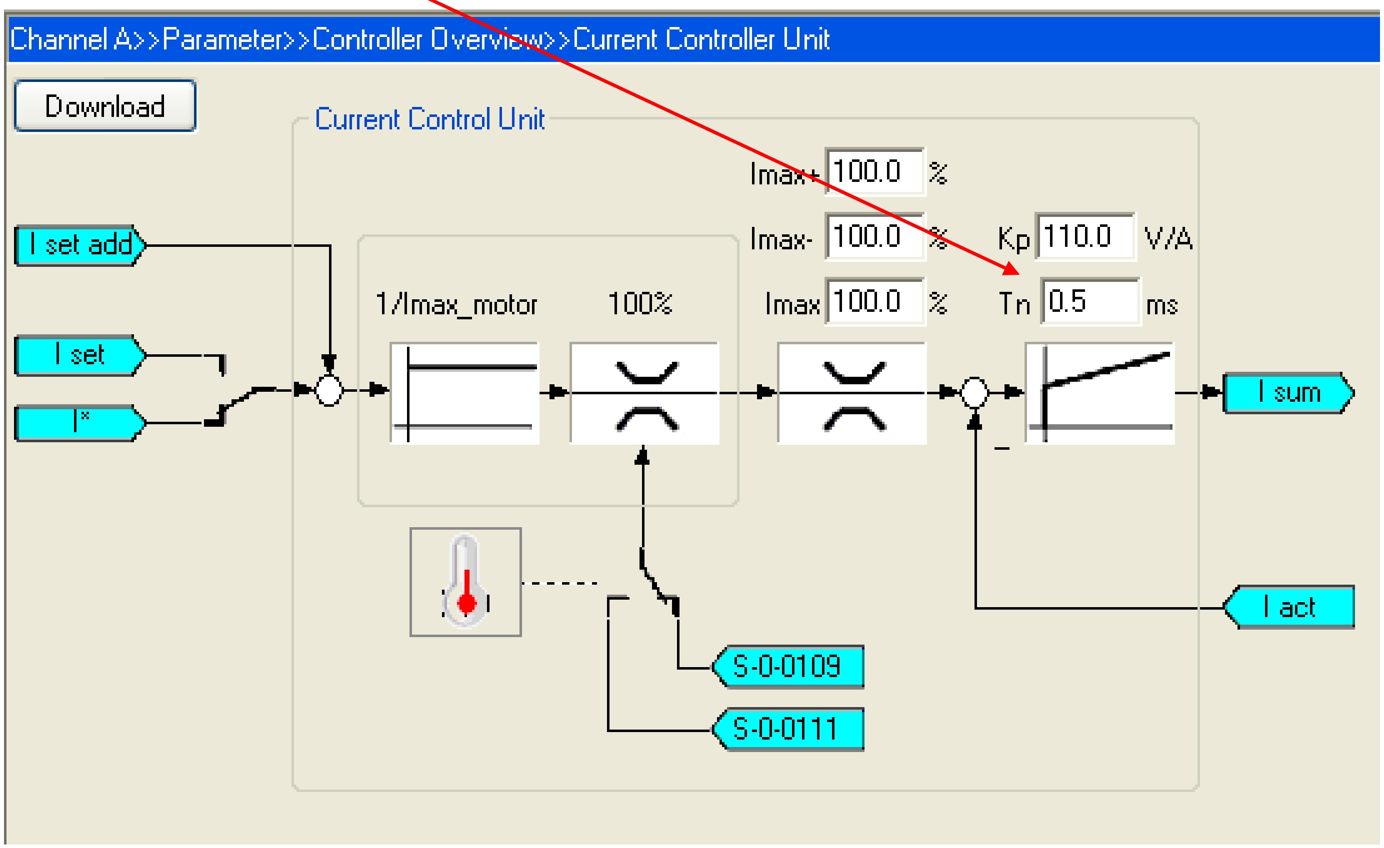
- Controller Overview

- The setup of the controllers goes from the “inside” (Current Controller) to the “outside” (Position Controller).

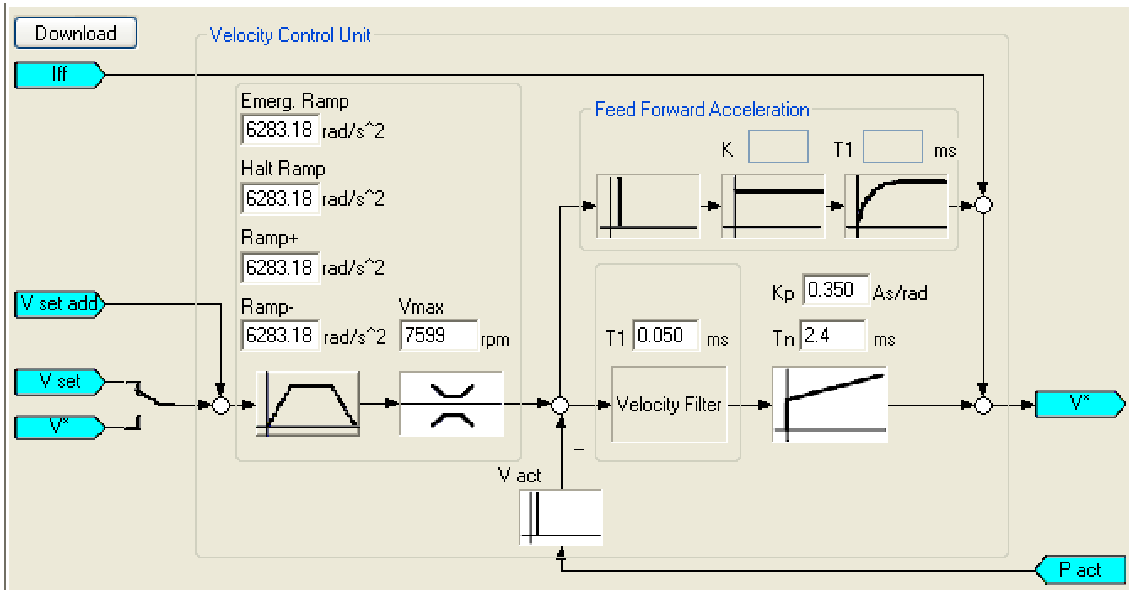


Current Controller Unit

- Kp and Tn of the current controller are set by the Motor default parameters.



Velocity Controller Unit



Acceleration limit value "S-0-0137" to Ramp in [sec]

T = V / Acceleration limit value S-0-0137 [rad/sec2] / 2π

Example:

Vmax = 6000 rpm => 100 rev/sec

S-0-0137 = 20971 [rad/sec2] =>

20971 [rad/sec2] / 2π = 3337,64 [rev/sec2]

t = V / a

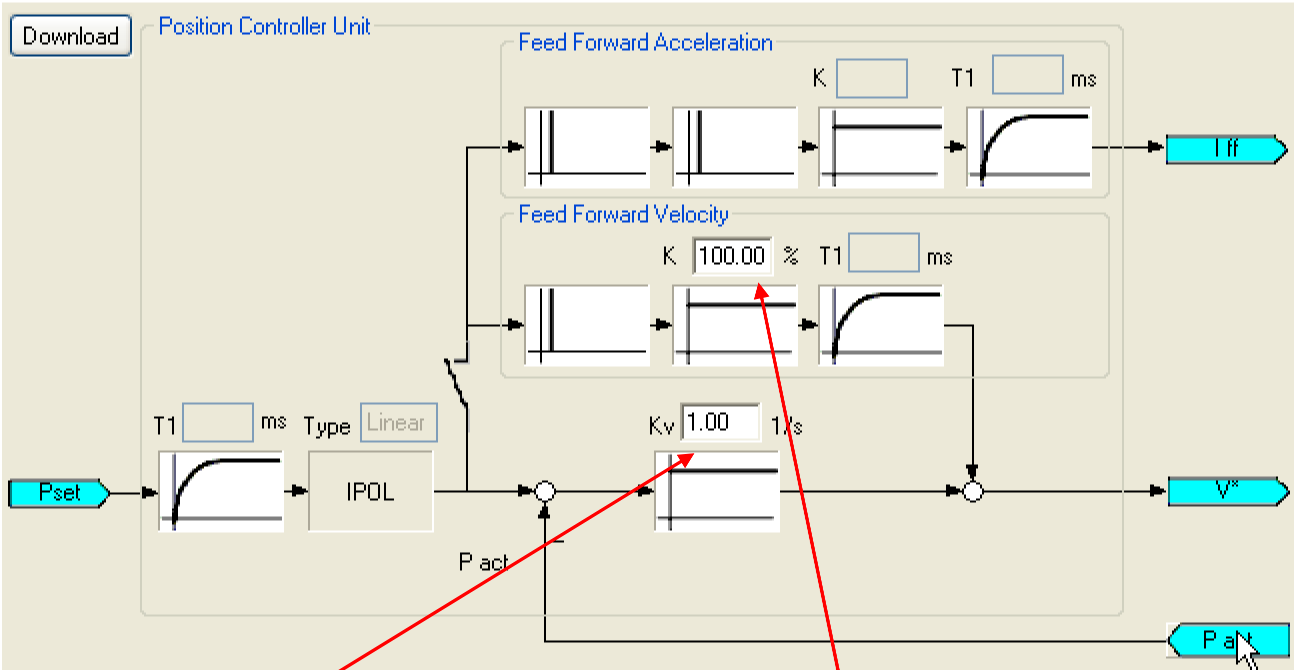
t = 100[rev/sec] / 3338 [rev/sec2]

t = 30 ms

- The scaling “rad” is fix.
- The perigon is 2 π radian or 360 degree; That is:

$$1\,\mathrm{rad} = \frac{360^\circ}{2\pi} = \frac{180^\circ}{\pi} \approx 57{,}29577951^\circ$$

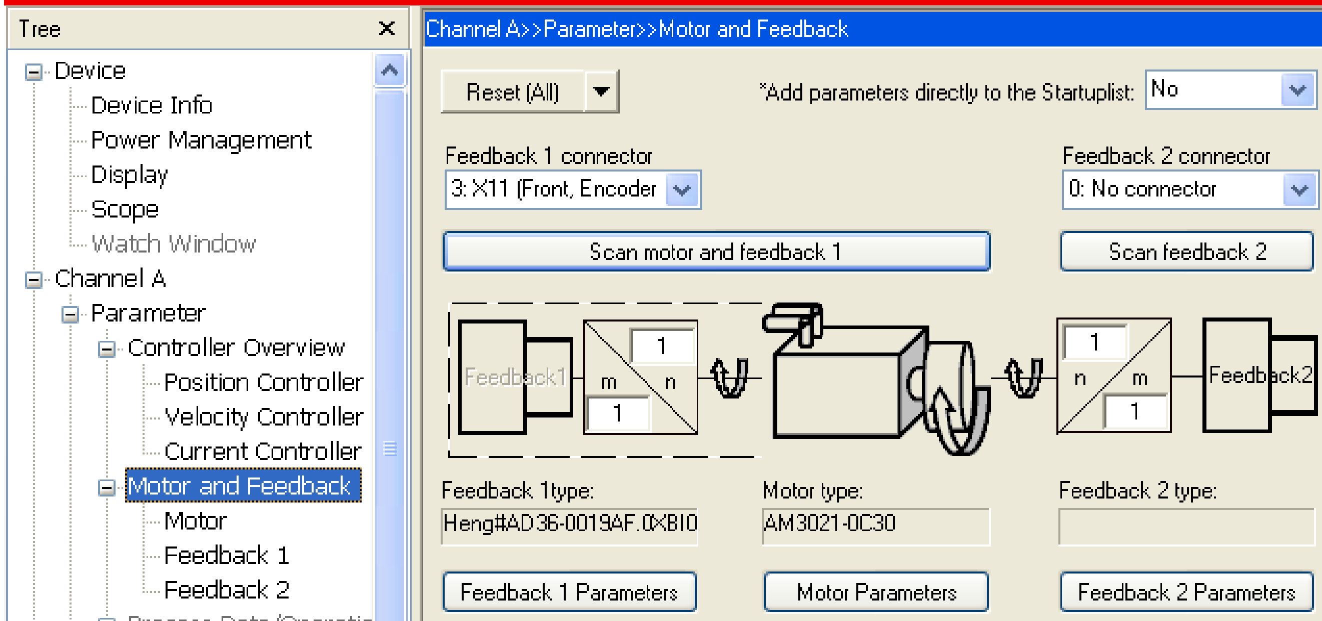
Position Controller Unit



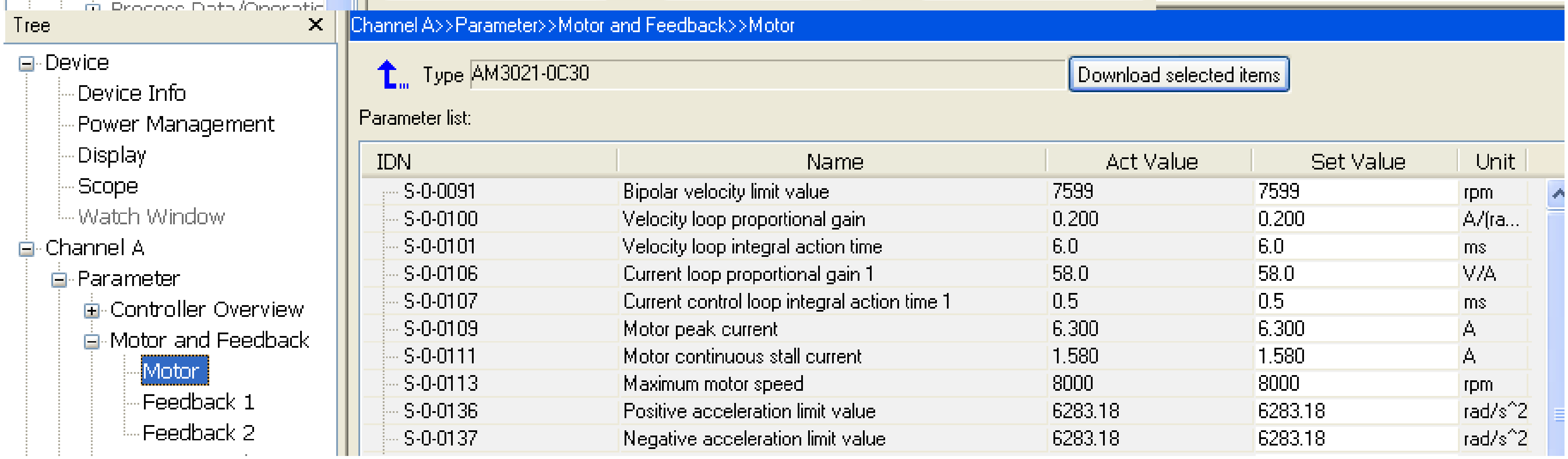
"Proportional gain" and "Velocity Feed Forward" in the position controller

AX5xxx

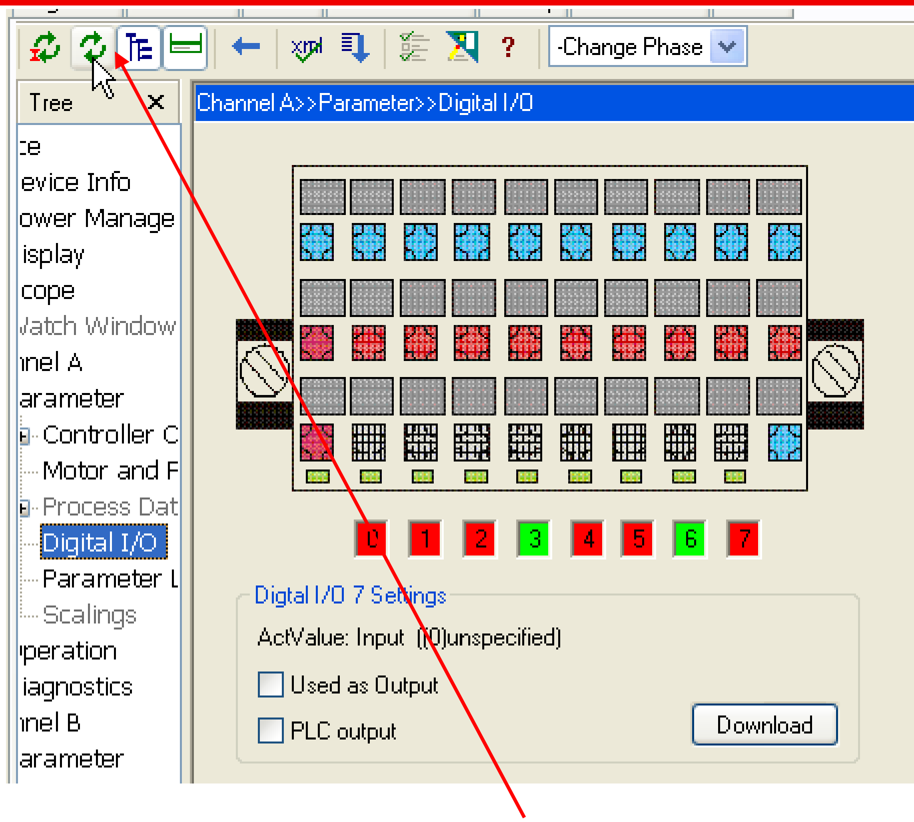
Motor and Feedback



13 Feedback options



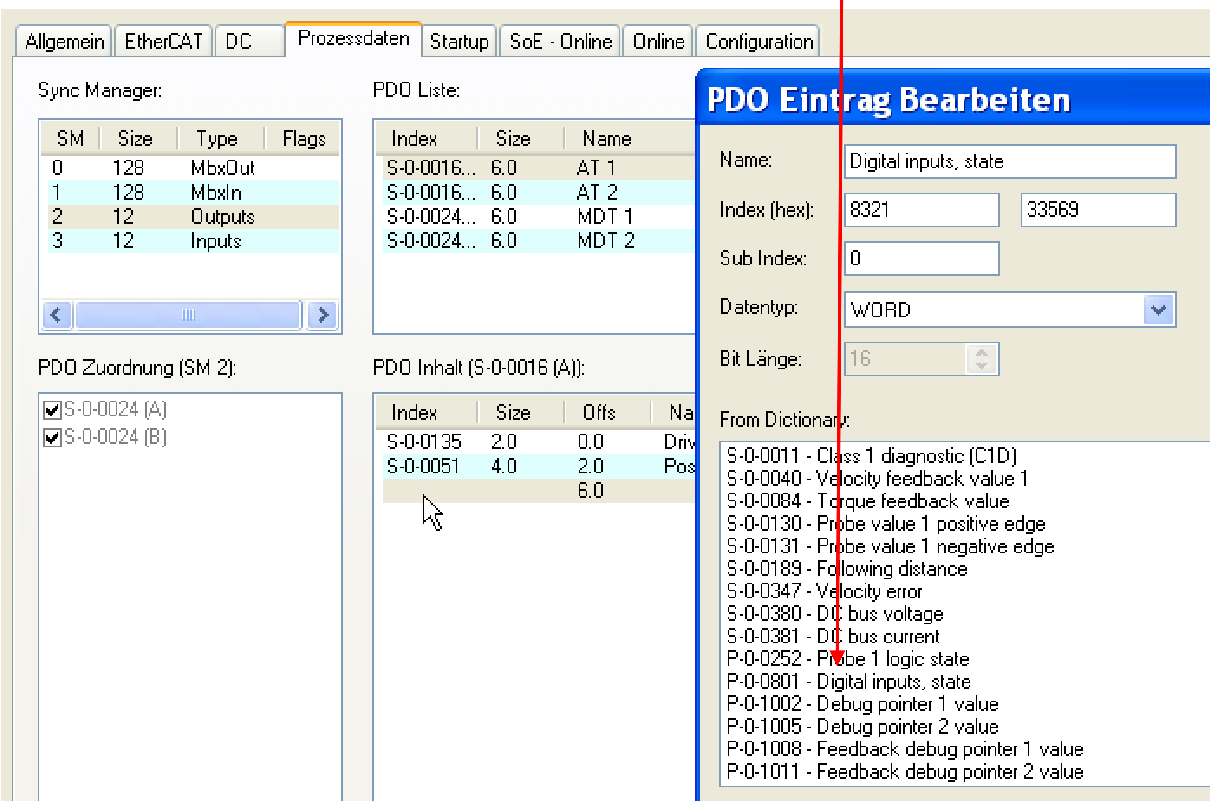
Digital I/O Link



- After running „Update IDN`s“ the input online state is displayed.

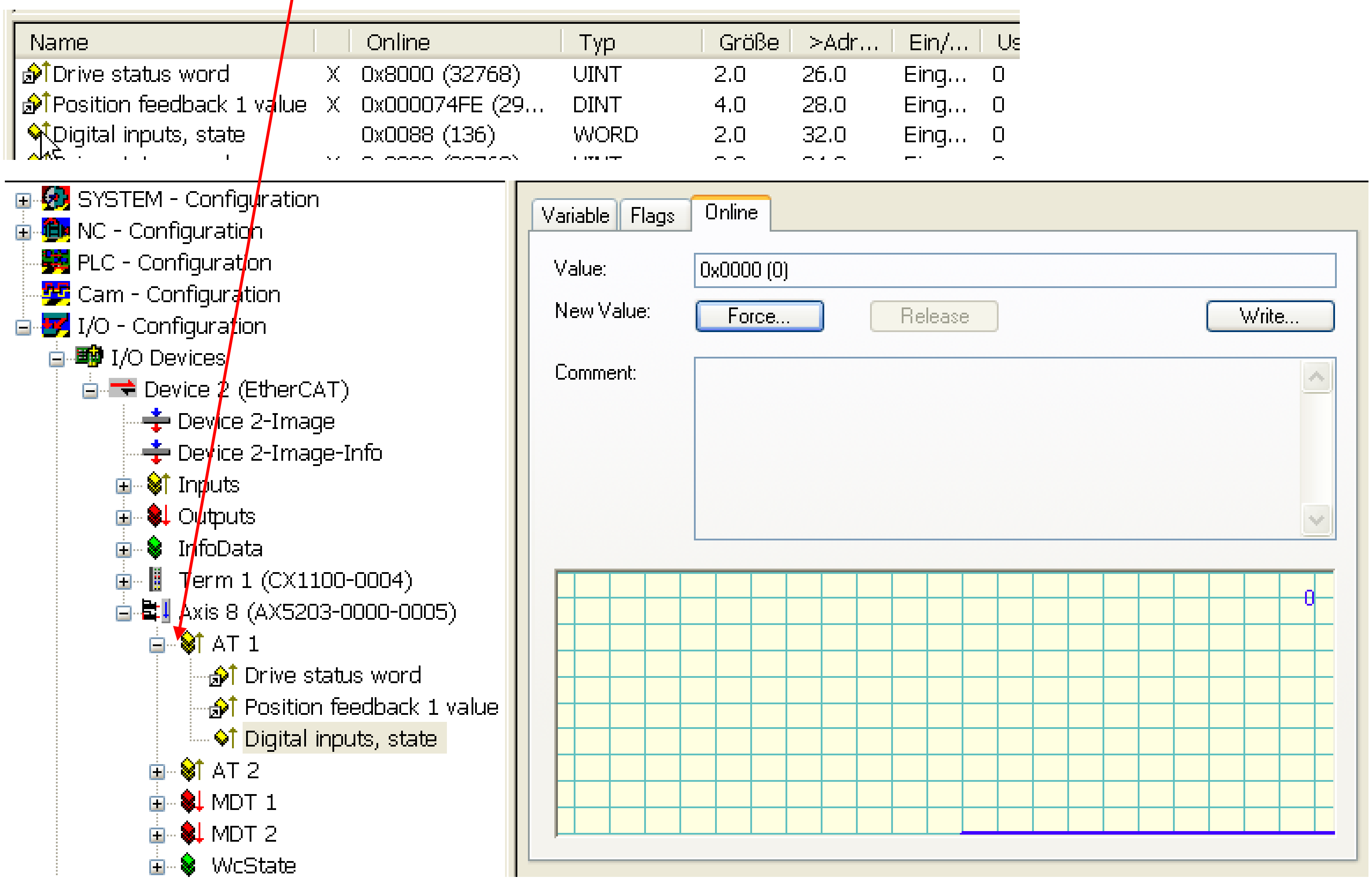
Digital I/O Link

- By the folder „Process data“ e.g. the I/O state can be add.
- Maximum is: 12 input words and 20 output words by 62,5 µsec.
- One Word = 2Byte



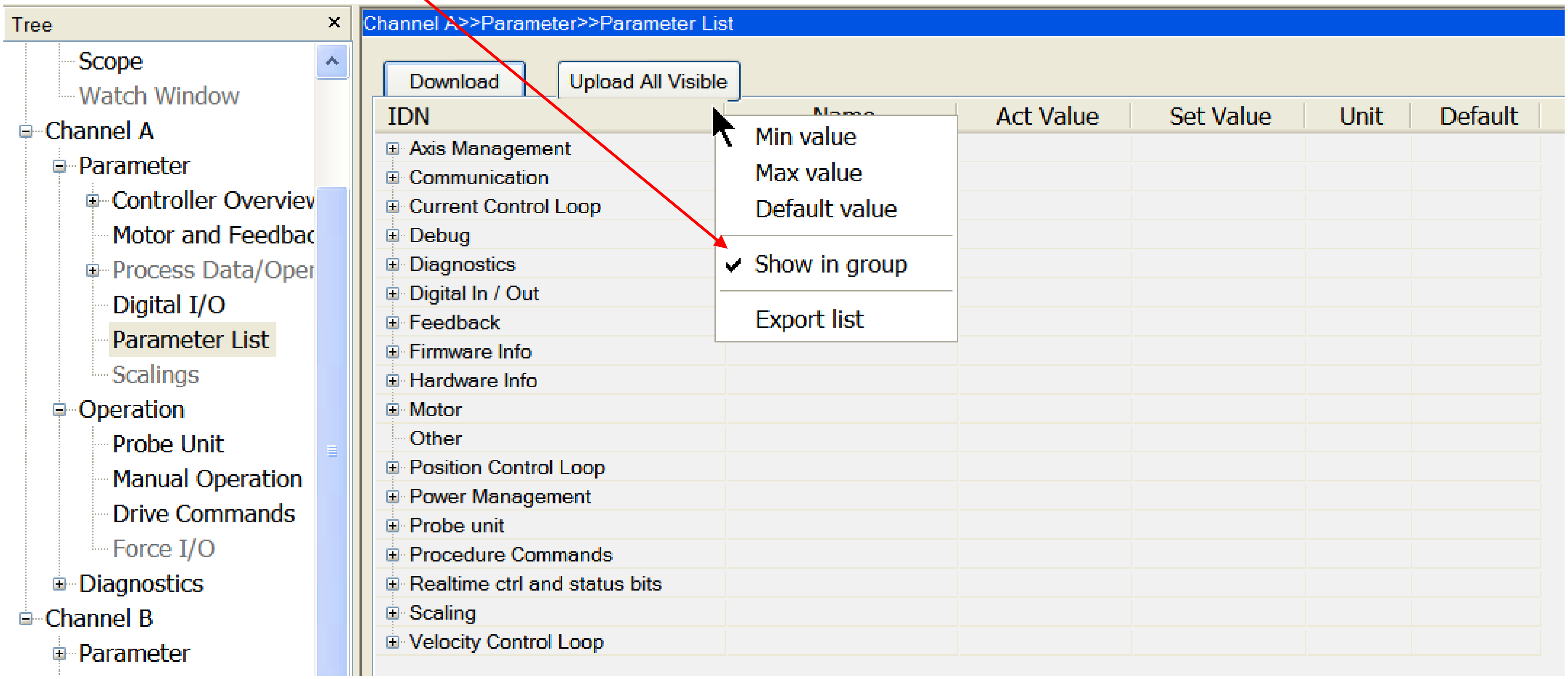
Digital I/O Link

- Now „Digital inputs, state“ is a part of AT1.



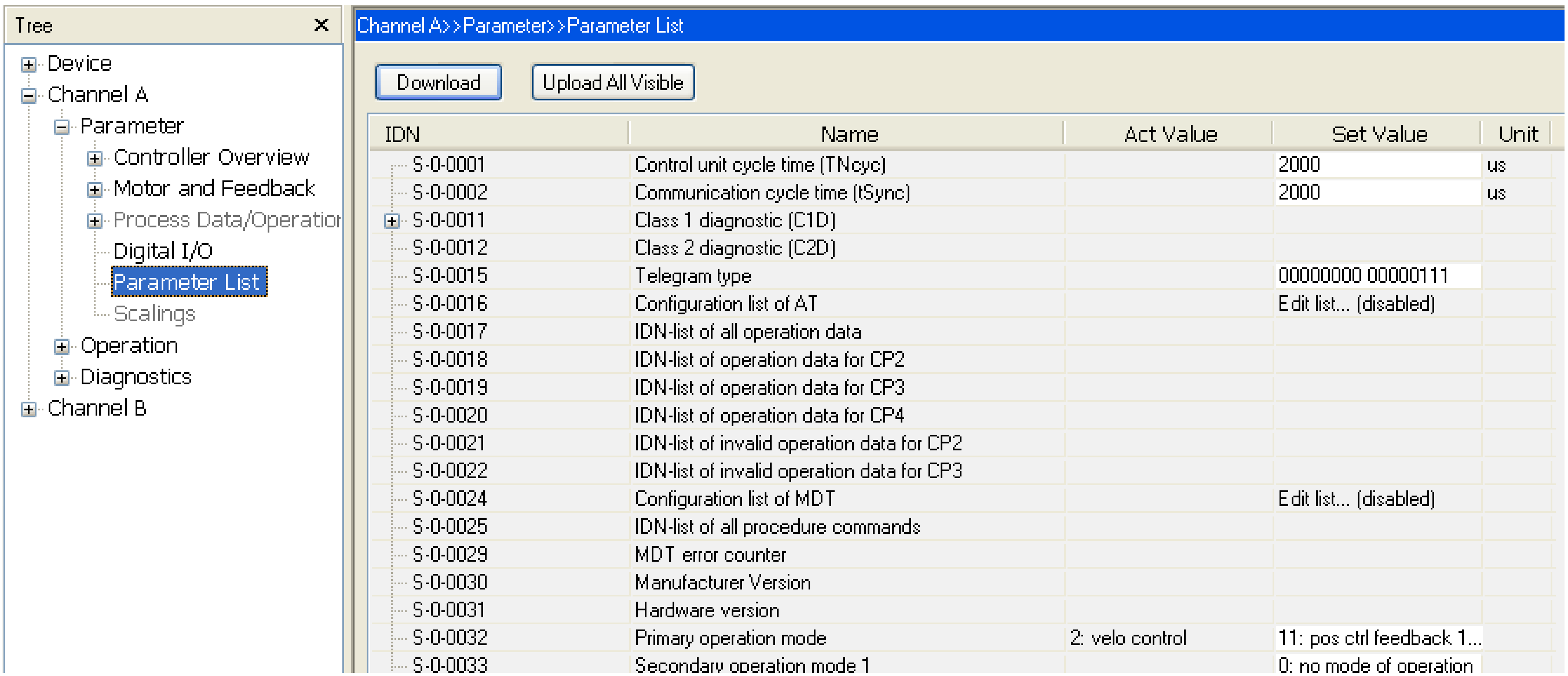
Parameter List

- By the „Parameter List“ there is access to the axis parameter.
- Two forms are possible.
- Show in groups:



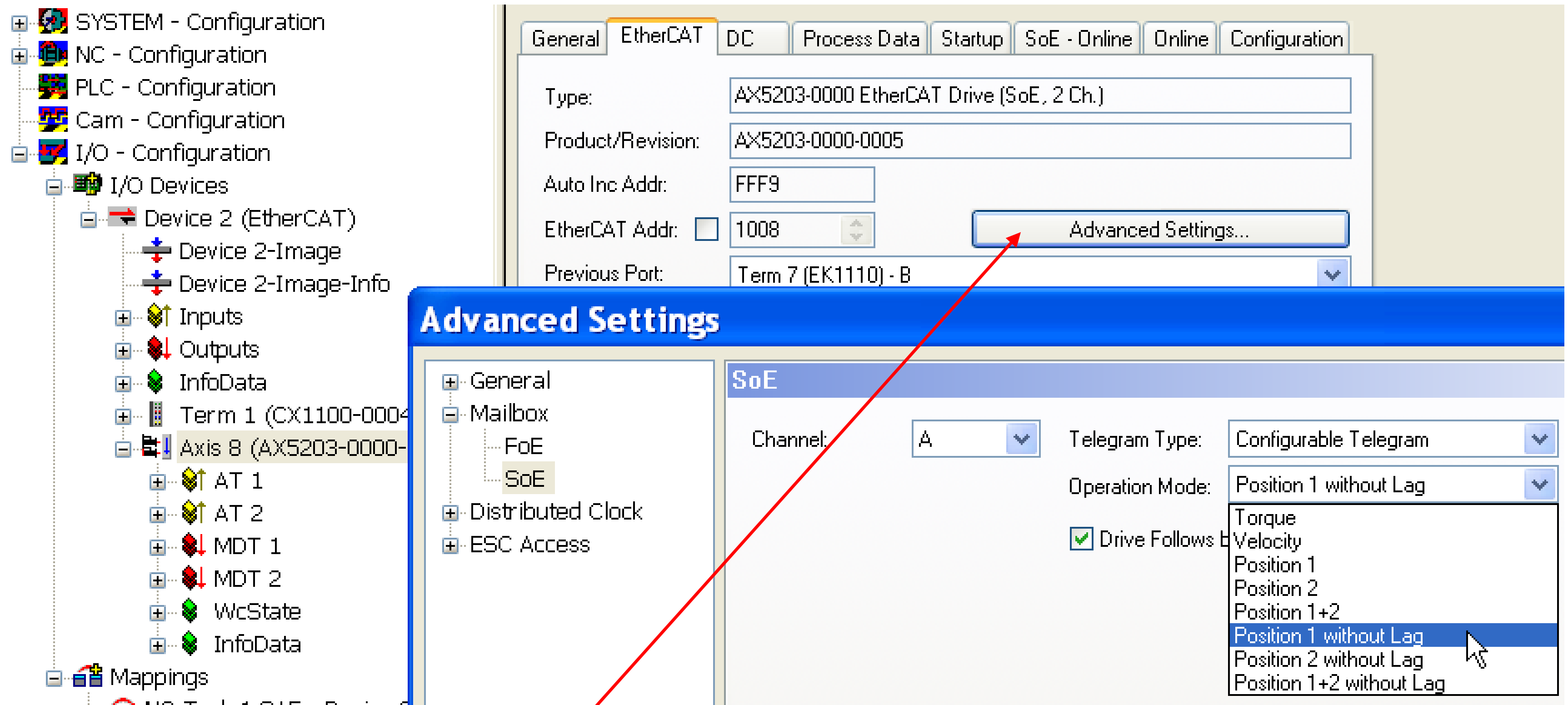
Parameter List

- Or IDN listed.



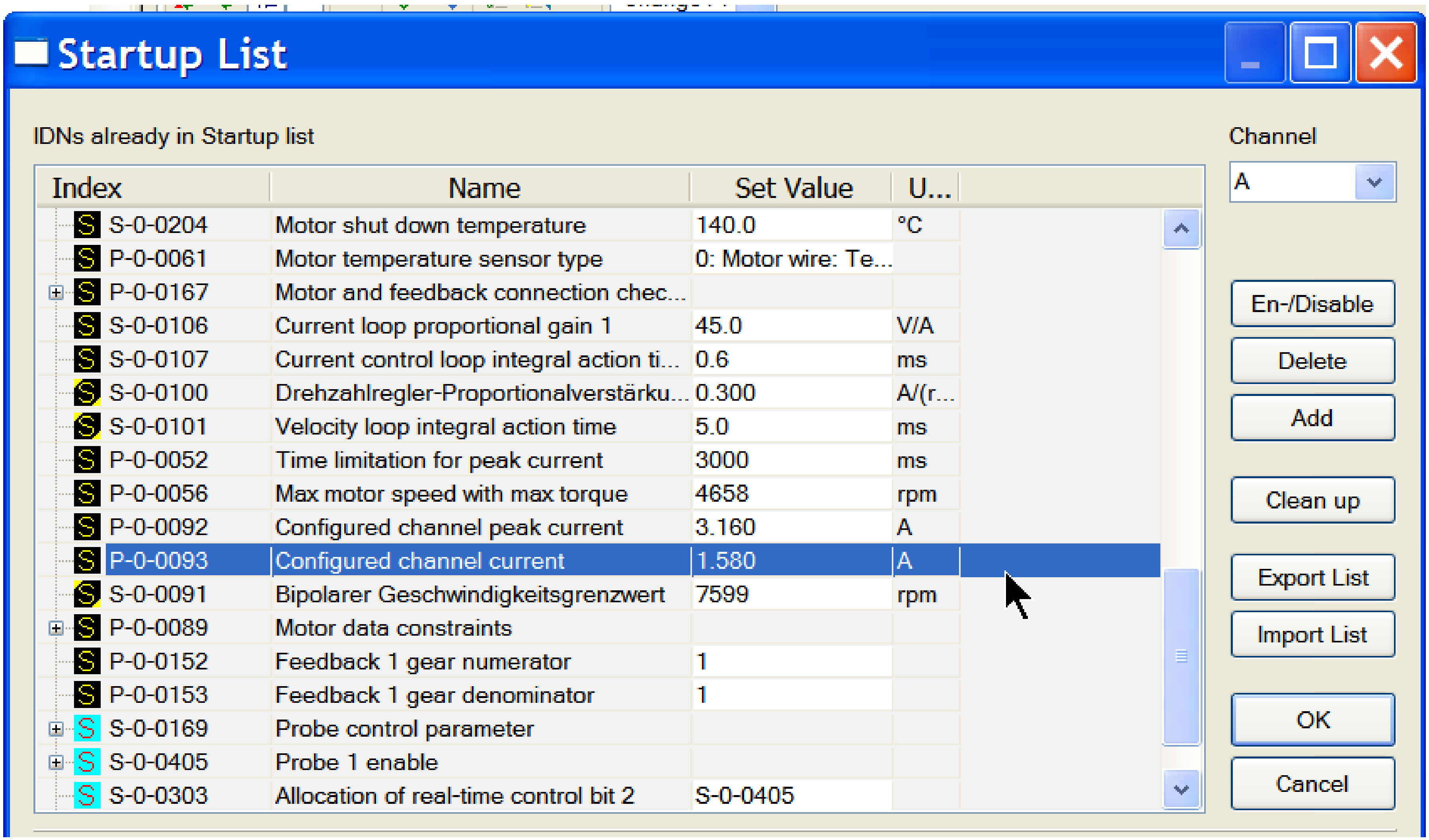
AX5xxx

AX5000 Position controller



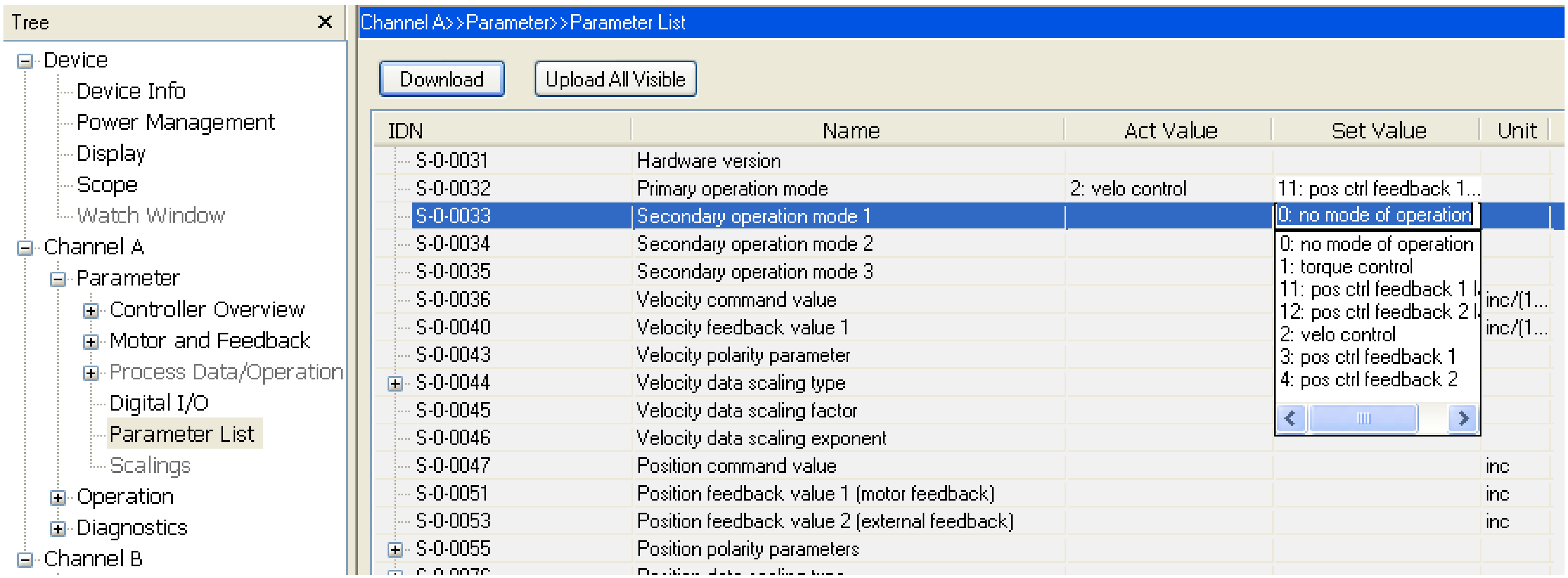
Activated by “Advanced Settings”. After this setting the “Position Controller” is done by the AX5000.

Channel current configuration



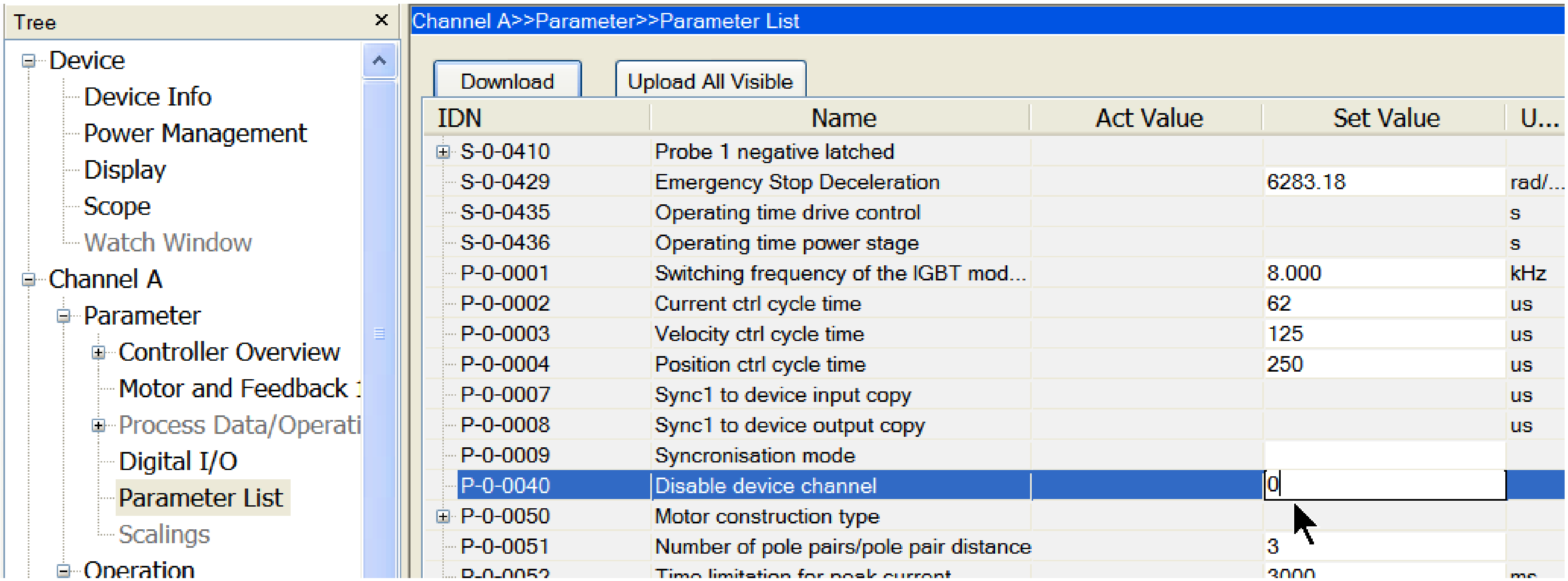
Index	Name	Set Value	U...
S-0-0204	Motor shut down temperature	140.0	°C
P-0-0061	Motor temperature sensor type	0: Motor wire: Te...	
P-0-0167	Motor and feedback connection chec...		
S-0-0106	Current loop proportional gain 1	45.0	V/A
S-0-0107	Current control loop integral action ti...	0.6	ms
S-0-0100	Drehzahlregler-Proportionalverstärku...	0.300	A/(r...
S-0-0101	Velocity loop integral action time	5.0	ms
P-0-0052	Time limitation for peak current	3000	ms
P-0-0056	Max motor speed with max torque	4658	rpm
P-0-0092	Configured channel peak current	3.160	A
P-0-0093	Configured channel current	1.580	A
S-0-0091	Bipolarer Geschwindigkeitsgrenzwert	7599	rpm
P-0-0089	Motor data constraints		
P-0-0152	Feedback 1 gear numerator	1	
P-0-0153	Feedback 1 gear denominator	1	
S-0-0169	Probe control parameter		
S-0-0405	Probe 1 enable		
S-0-0303	Allocation of real-time control bit 2	S-0-0405	

S-0-0033 Secondary operation mode



Secondary operation modes are selectable by the Controlword! in prep.

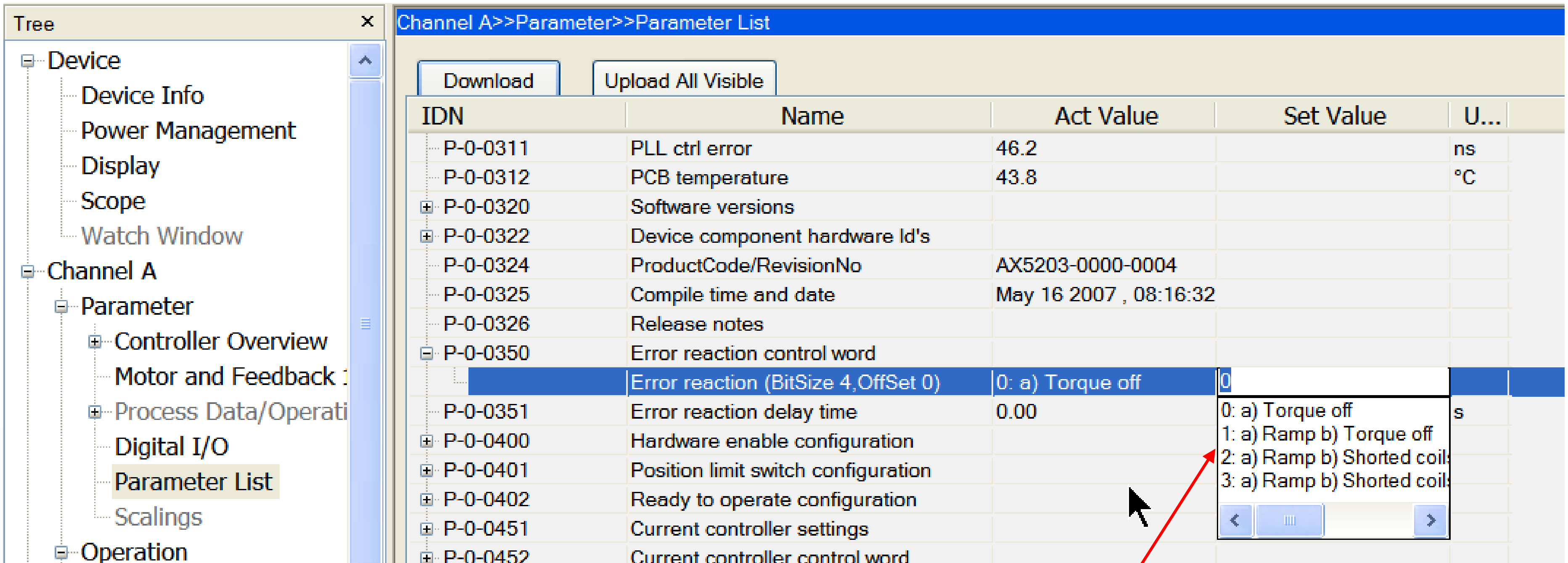
Disable device channel



e.g. to use only channel 2 feedback

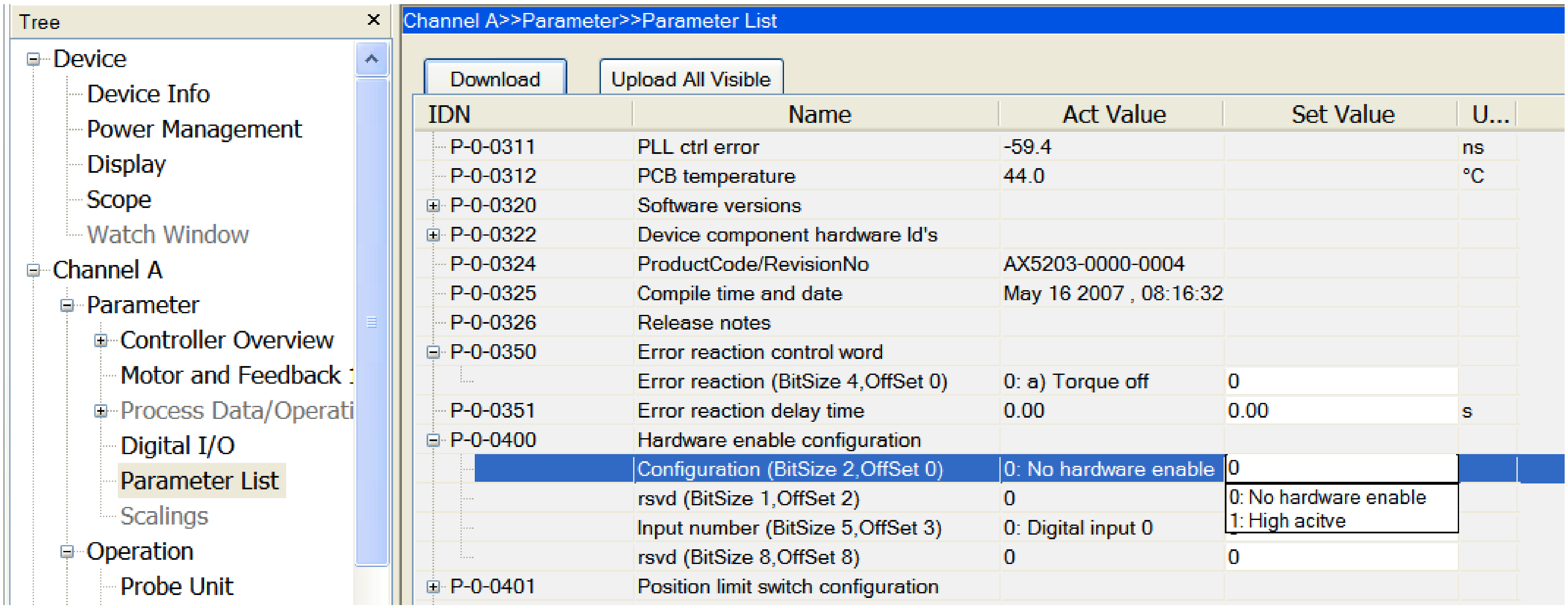
AX5xxx

Error reaction



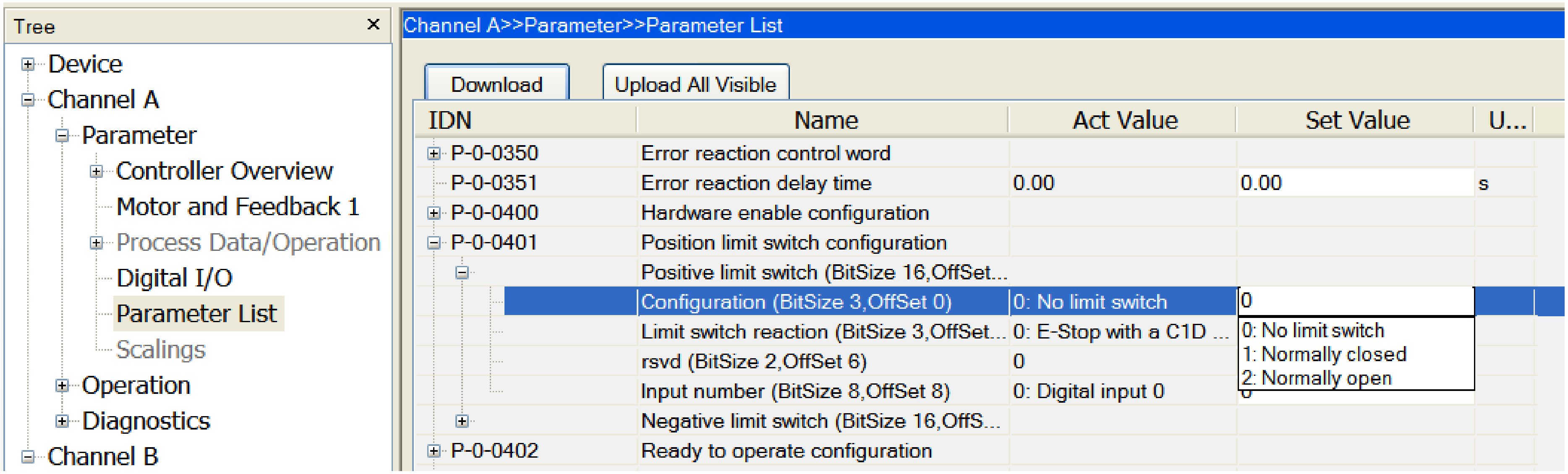
What should happen after error detection.

Hardware Enable



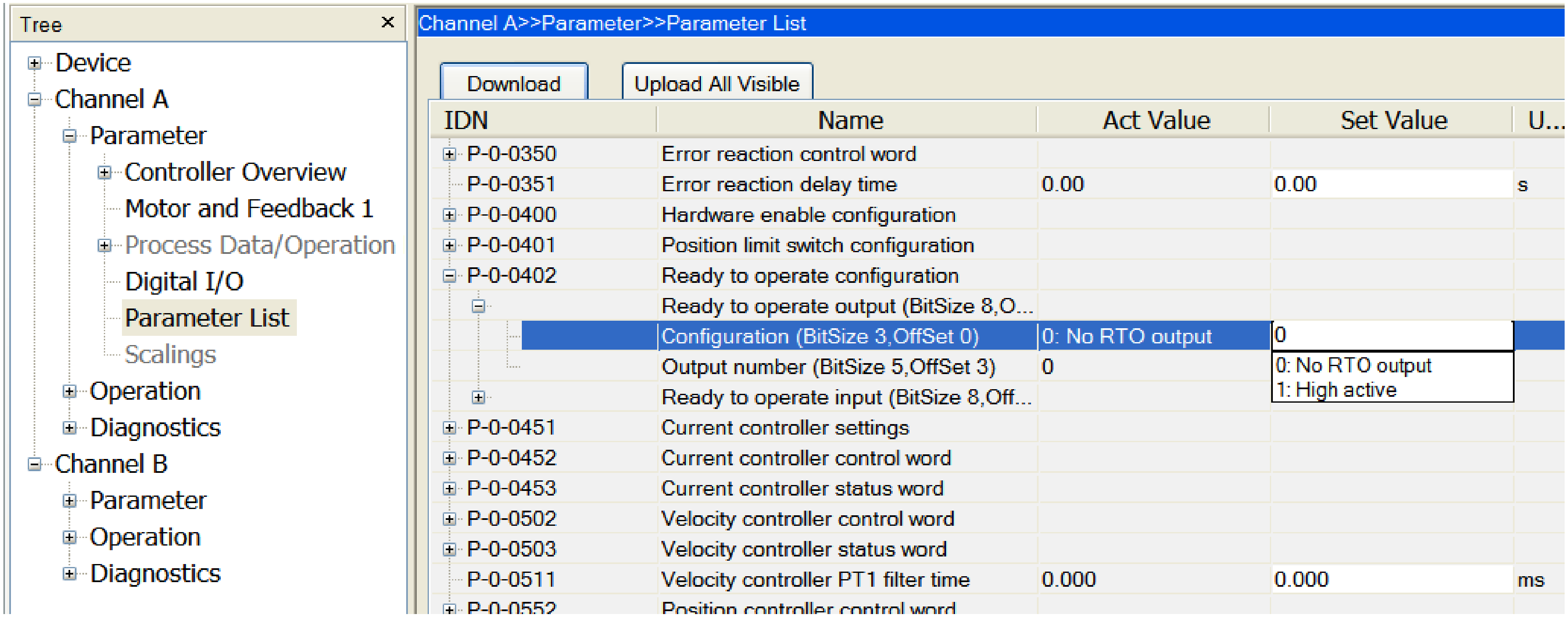
AX5xxx

Limit switch configuration

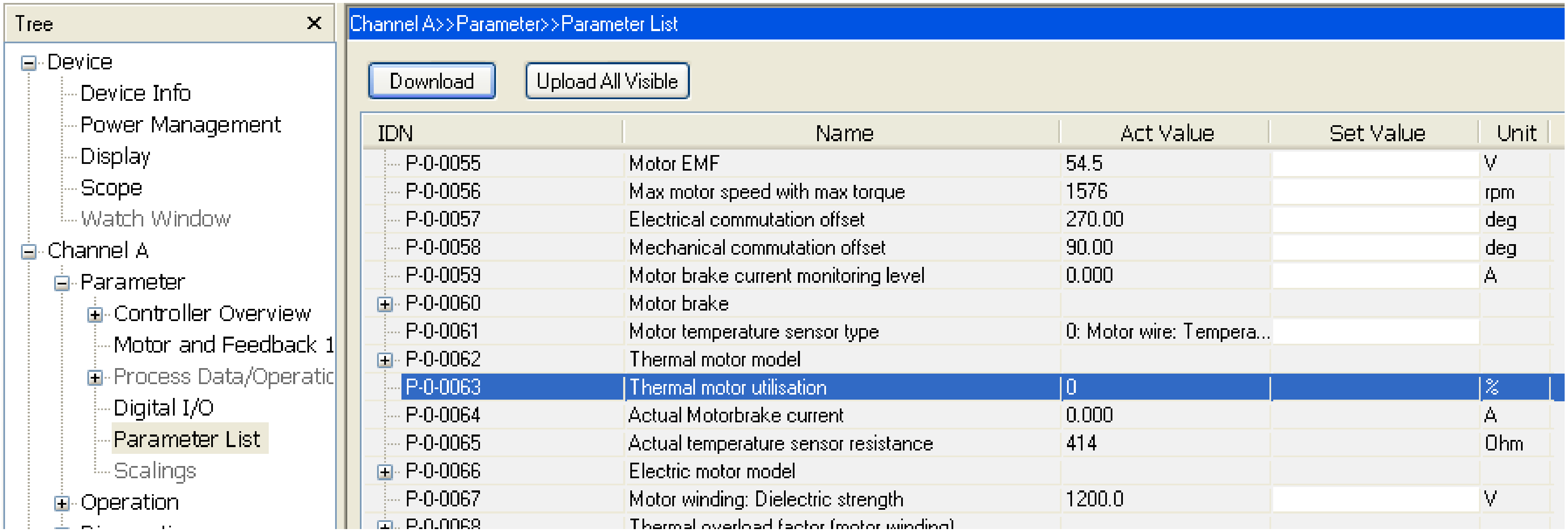


AX5xxx

RTO (BTB) Function

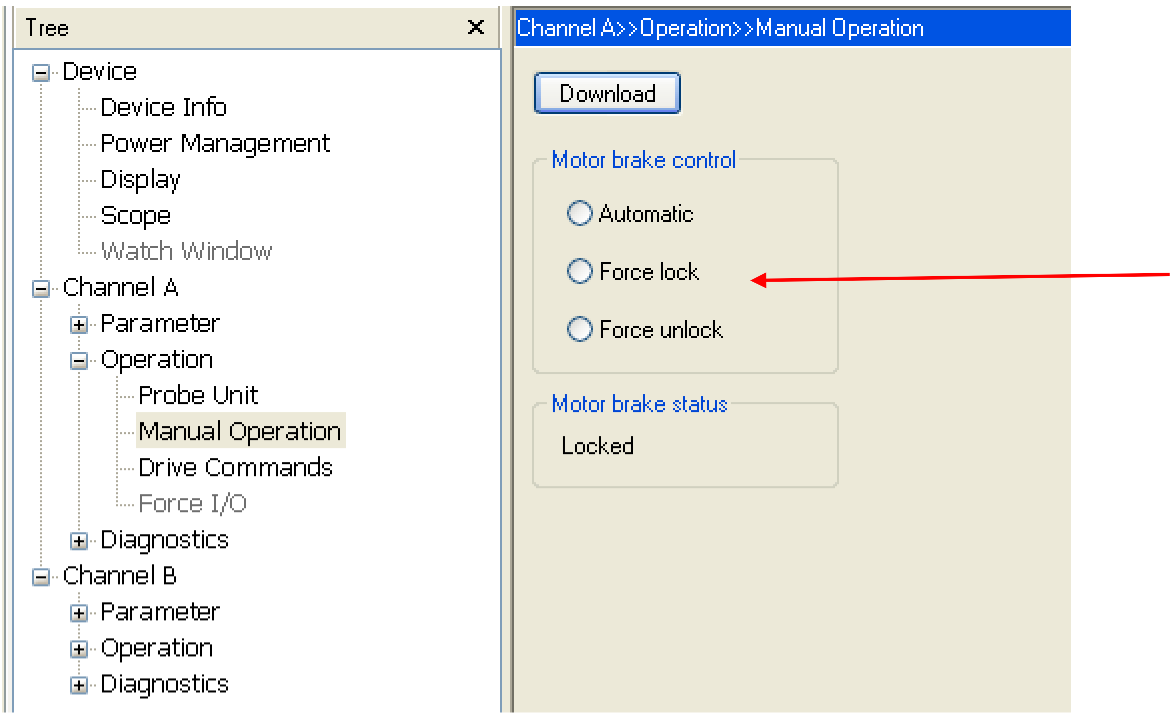


Display Motor working load by P-0-0063



This function has to be enabled by IDN P-0-0062 (Reaction =1)

Manual Operation

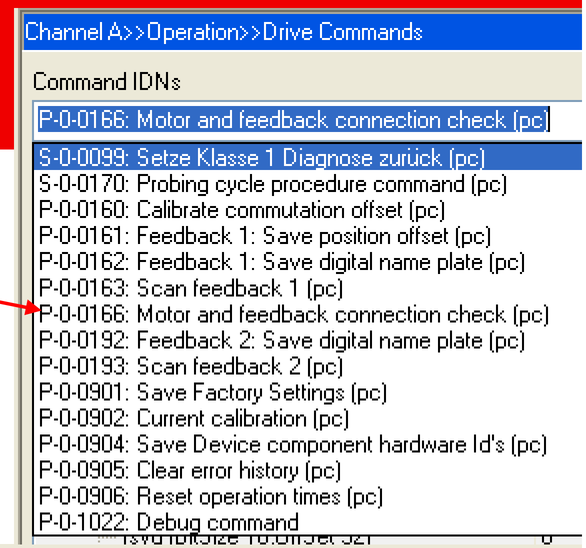


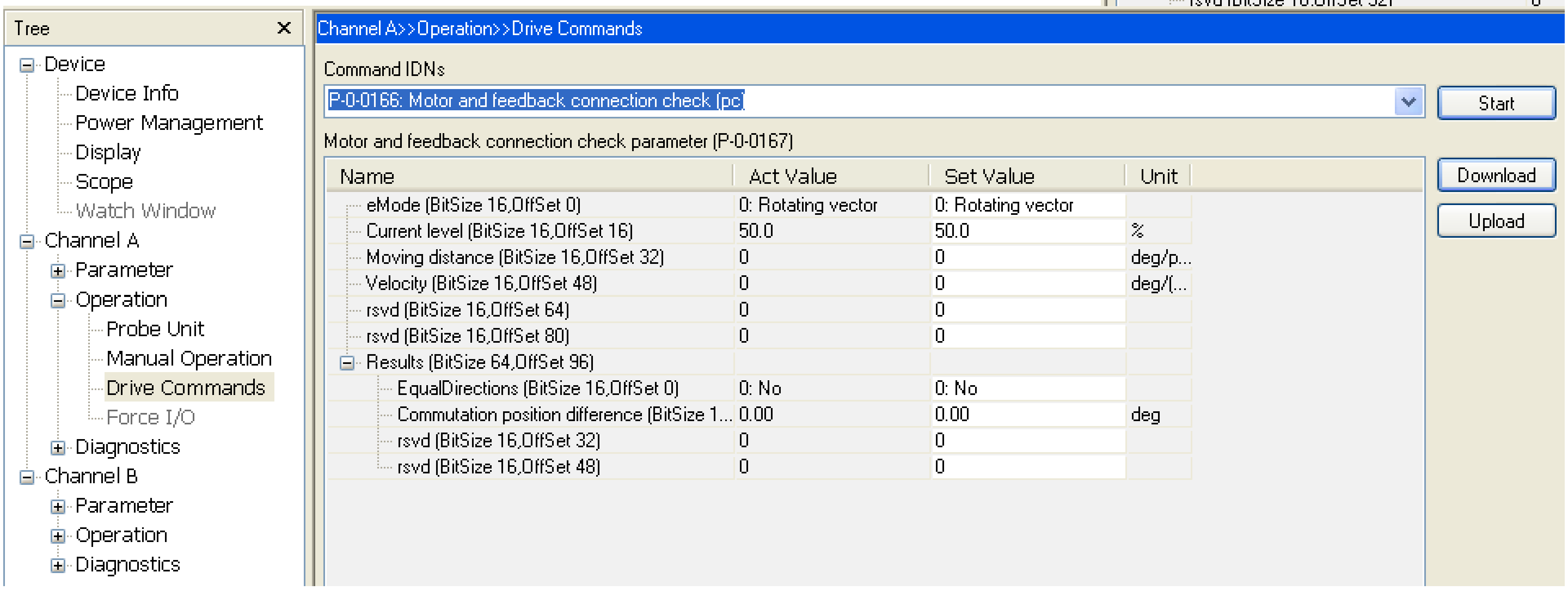
- Brake operations

AX5xxx

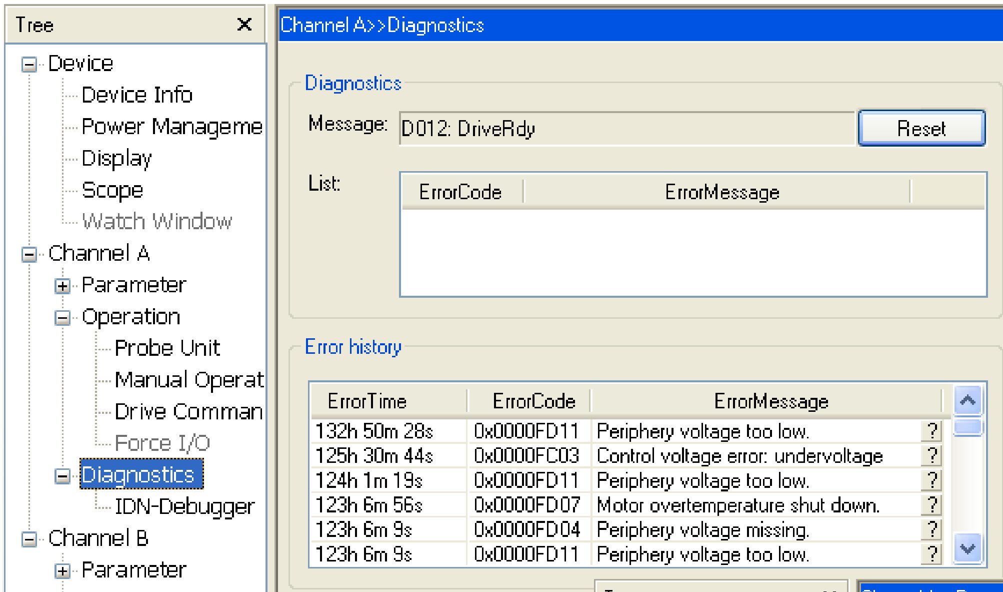
Drive Commands

- e.g. Motor feedback connection check

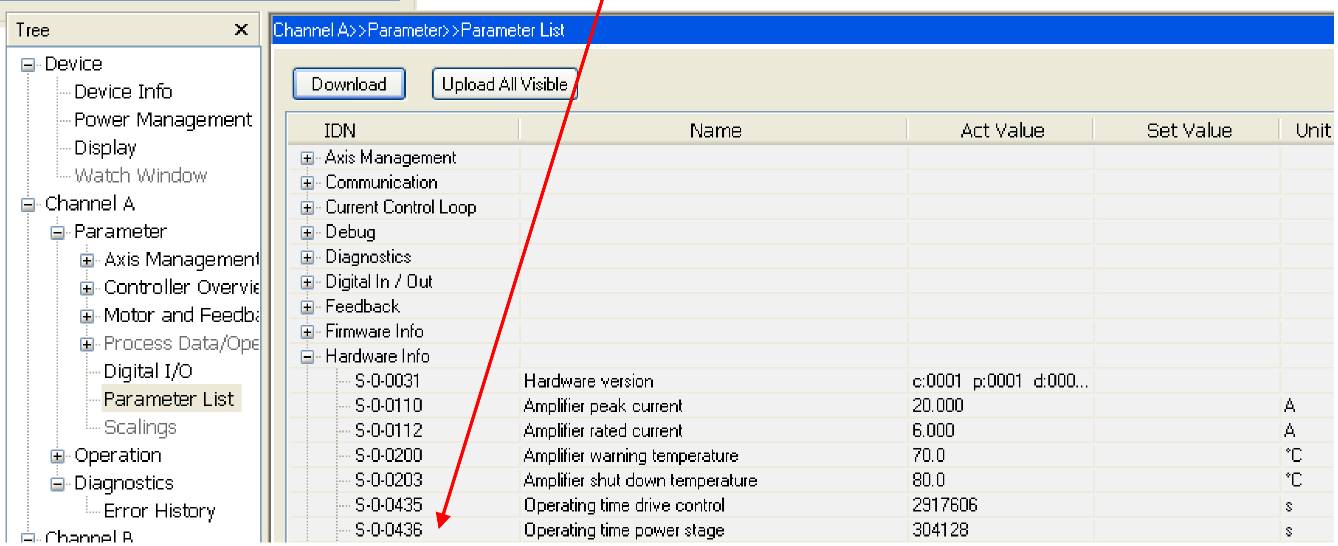




Diagnostics and error history

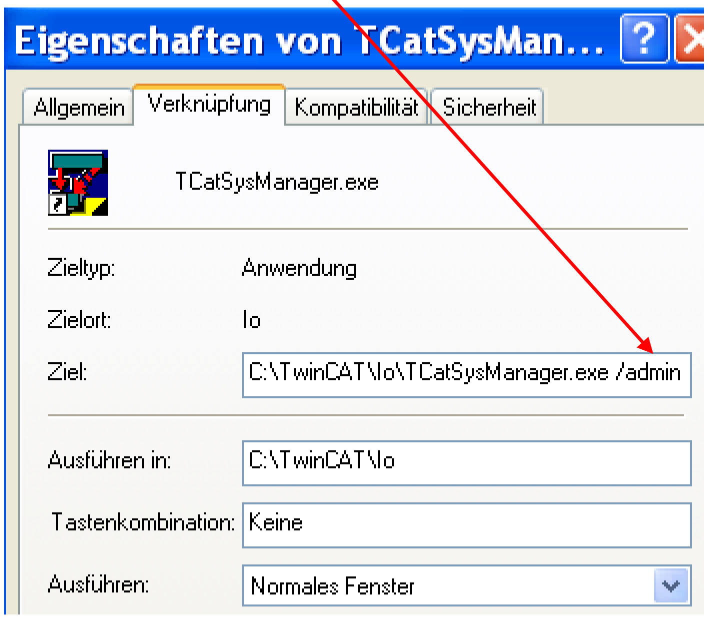


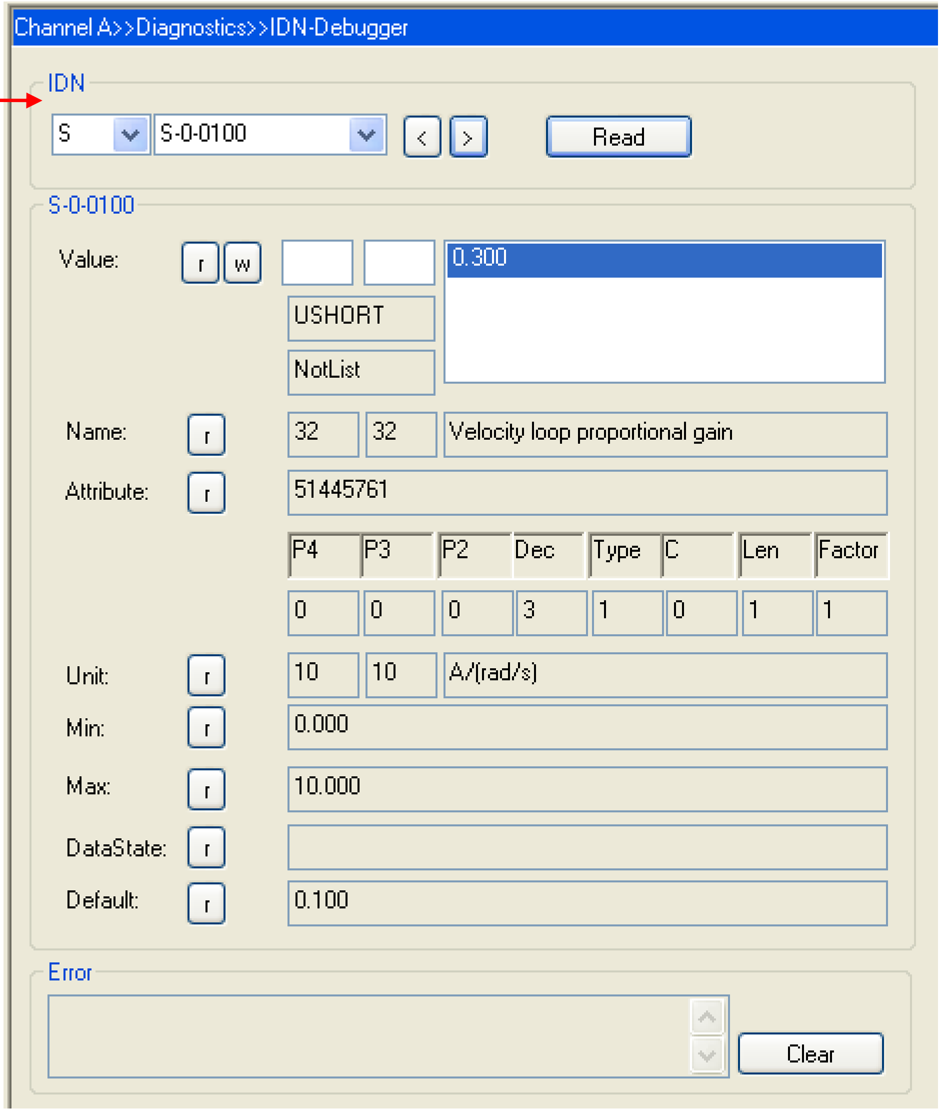
Operating time in S-0-0435 and S-0-0436



IDN-Debugger

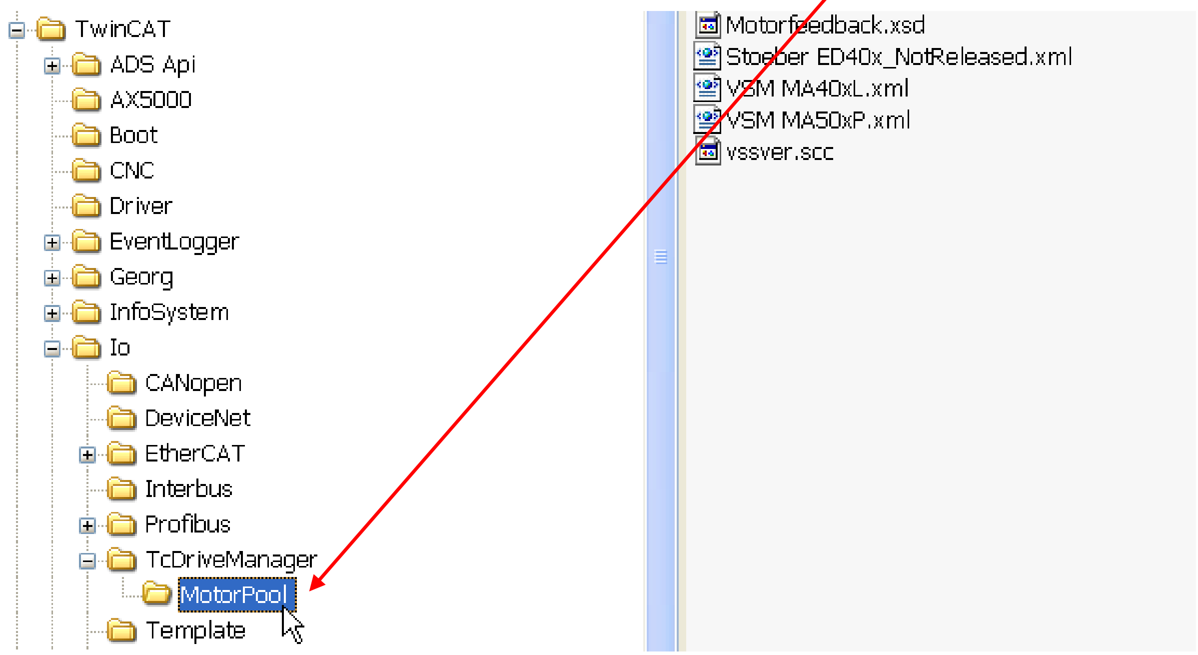
- Direct IDN access after /admin start





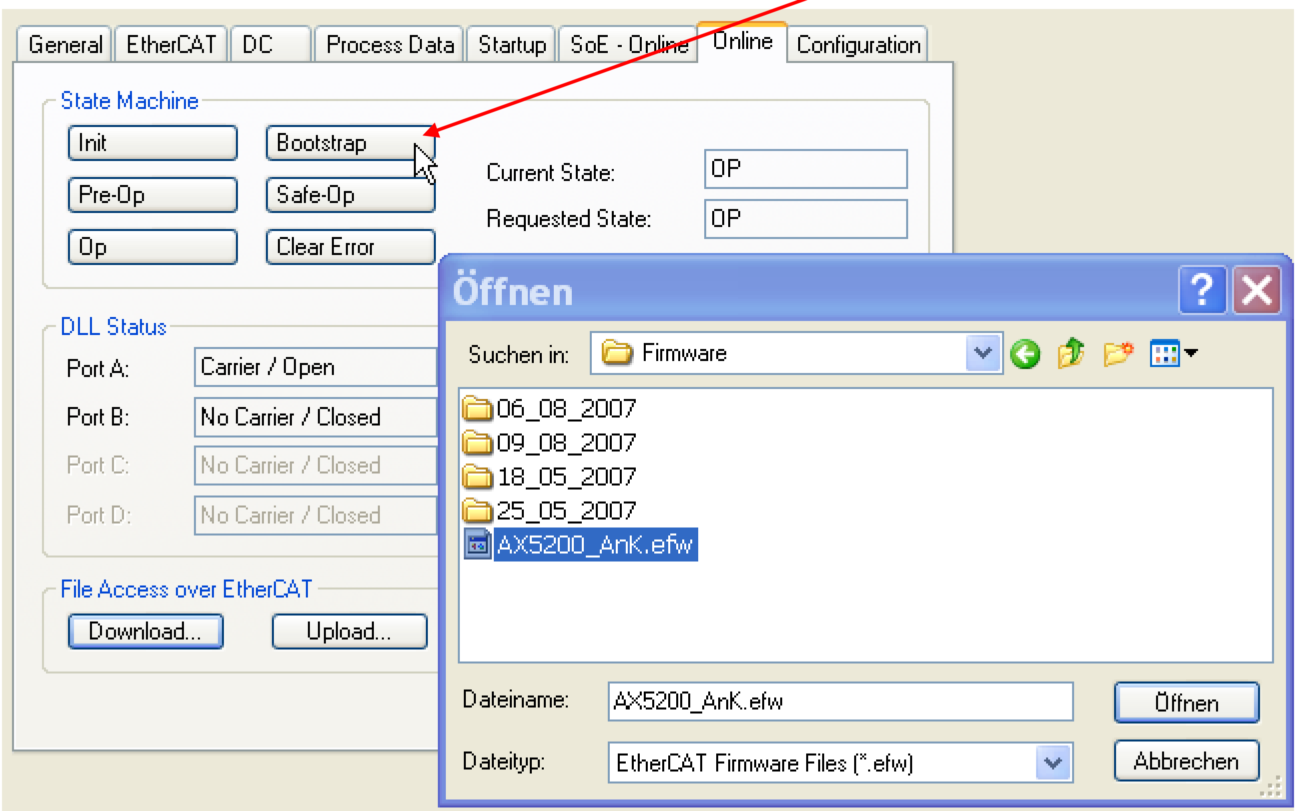
XML handling

- Location of motor default parameter file (motor.xml).



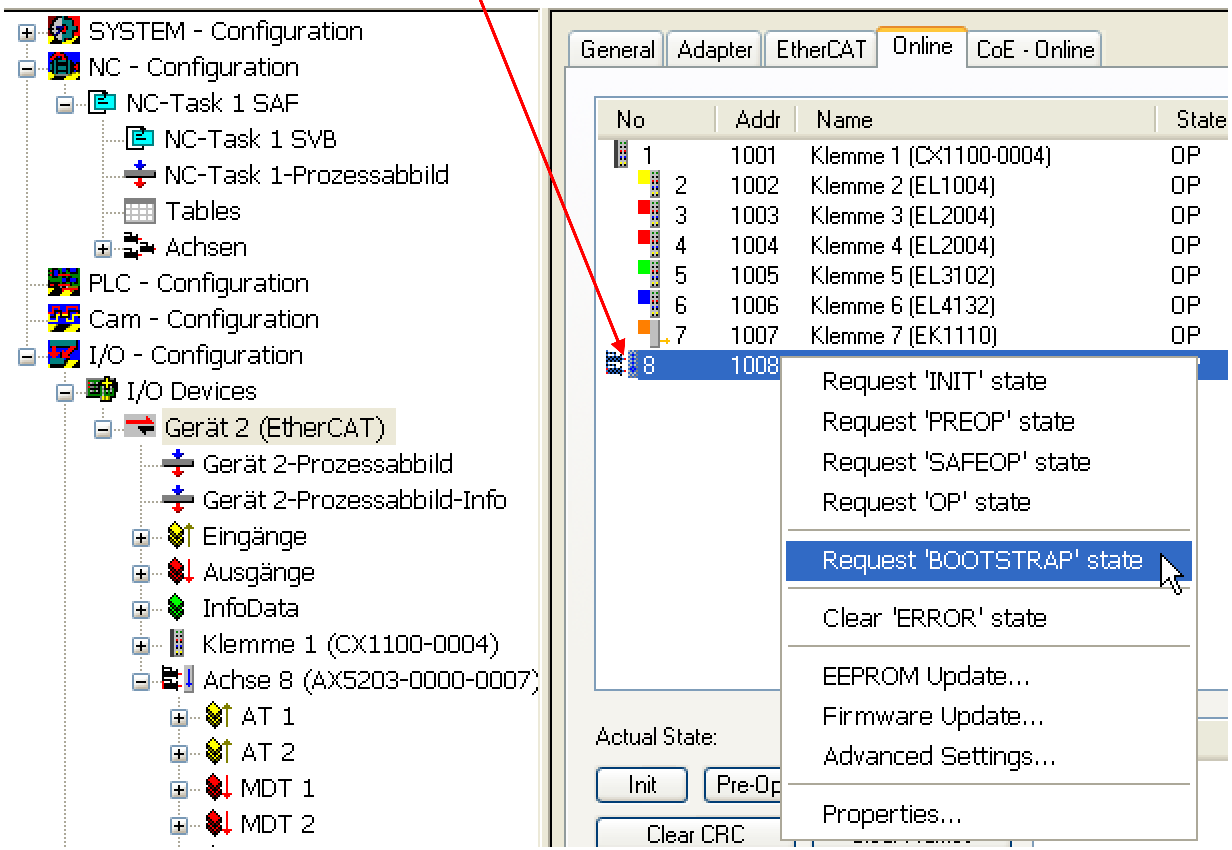
Softwareupdate

- To load a new firmware (xxx.efw file), please bring drive into „Bootstrap“ mode.



Multi- Softwareupdate

- Mark all Drives and “START” by right mouse click



Feedback setup

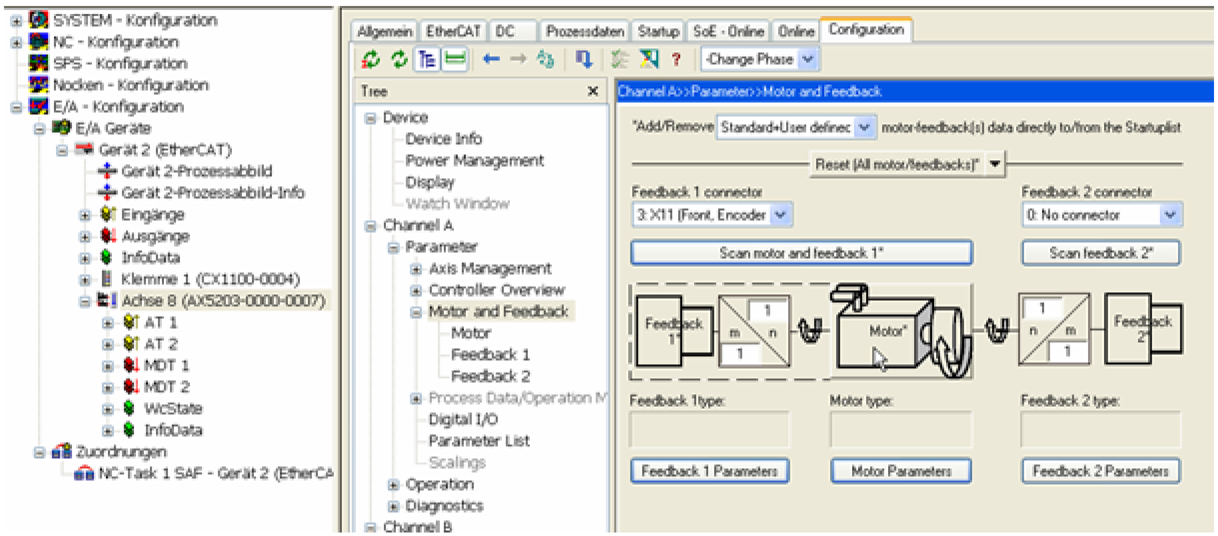
05.05.2008
V1.2
Rudolf W. Meier

Please see the document “Set_Motornameplate_2.doc”

Set digital name plate in AX5000.

The motor has to be free of load by running this procedure.

1. **Select the motor manually by click on “Motor”.**



Preparations

Three Steps to tune a Servo Controller

1. **Tune the current controller**

   The current loop PI controller parameters are tuned by the engineers of Beckhoff Drive Technology. In most case, default parameters is ok for the application.

2. **Tune the velocity controller**

   Switch off the filter and Tn of the velocity loop .

   Raise up Kp to the final point without overshoot

   Raise Tn up to 10-20 % overshoot.

   Activate the filter according to requirements

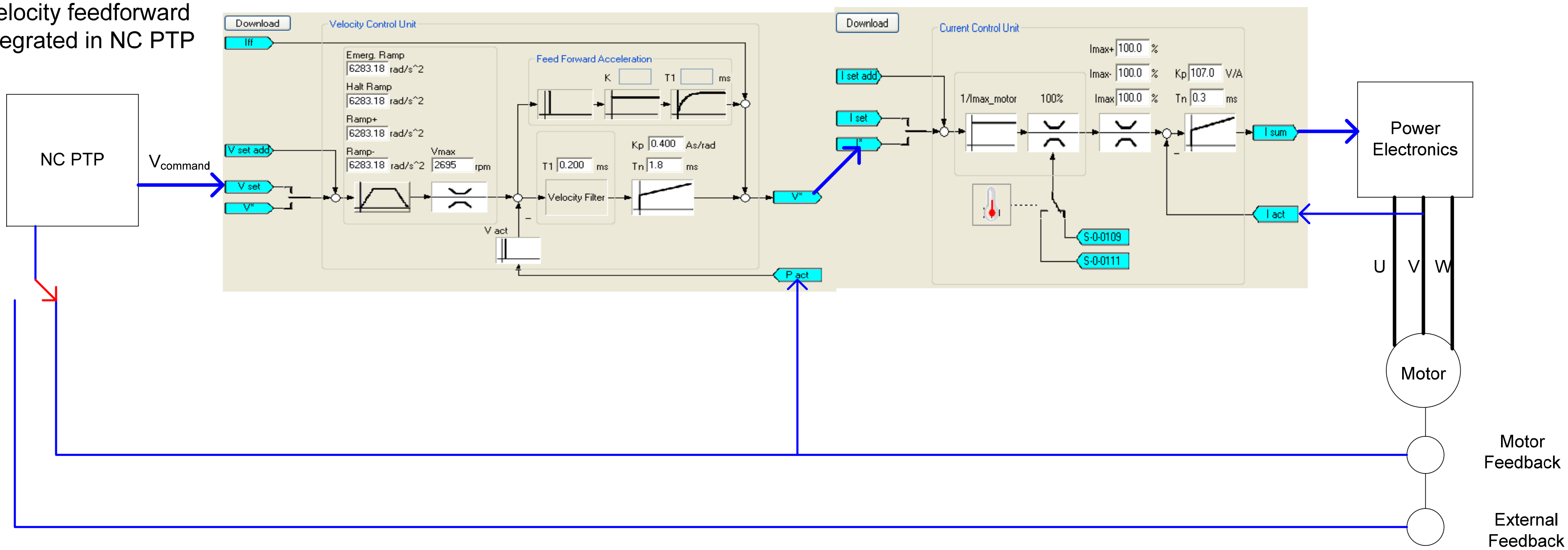
3. **Tune the position controller**

   The position controller gain should be scaled to reach less following error and specified settling time. This procedure can be watched by„TwinCAT Scop View".The System Manager generates the position setpoint.

Preparations

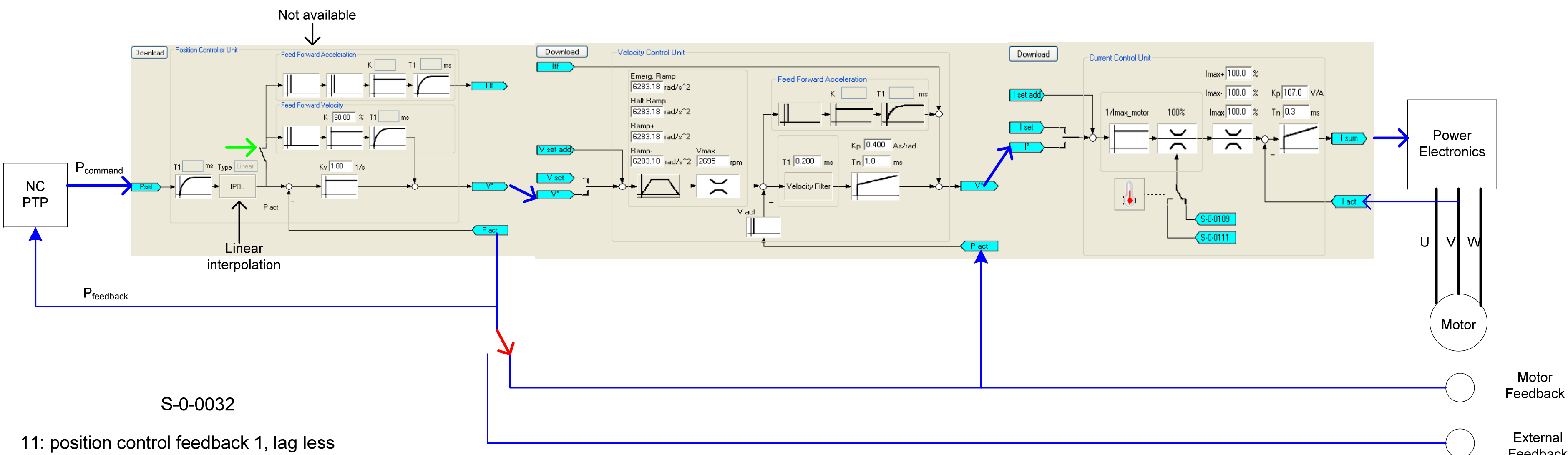
- **AX5000 velocity control and current control S-0-0032 =2**
- **NC PTP position control**

Position Controller P
Interpolation
Velocity feedforward
integrated in NC PTP



Preparations

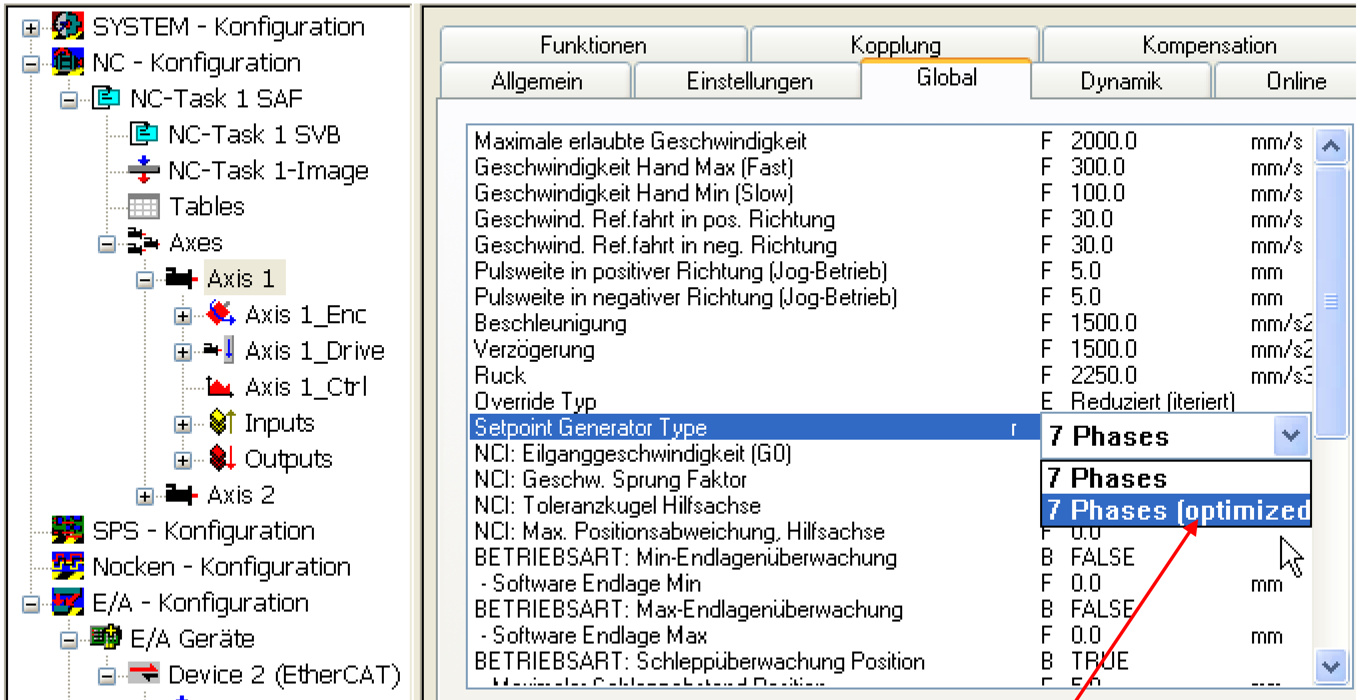
- AX5000 position control, velocity control, current control



S-0-0032

11: position control feedback 1, lag less
12: position control feedback 2, lag less
3: position control feedback 1
4: position control feedback 2
Velocity feedforward is important to reduce the position following error

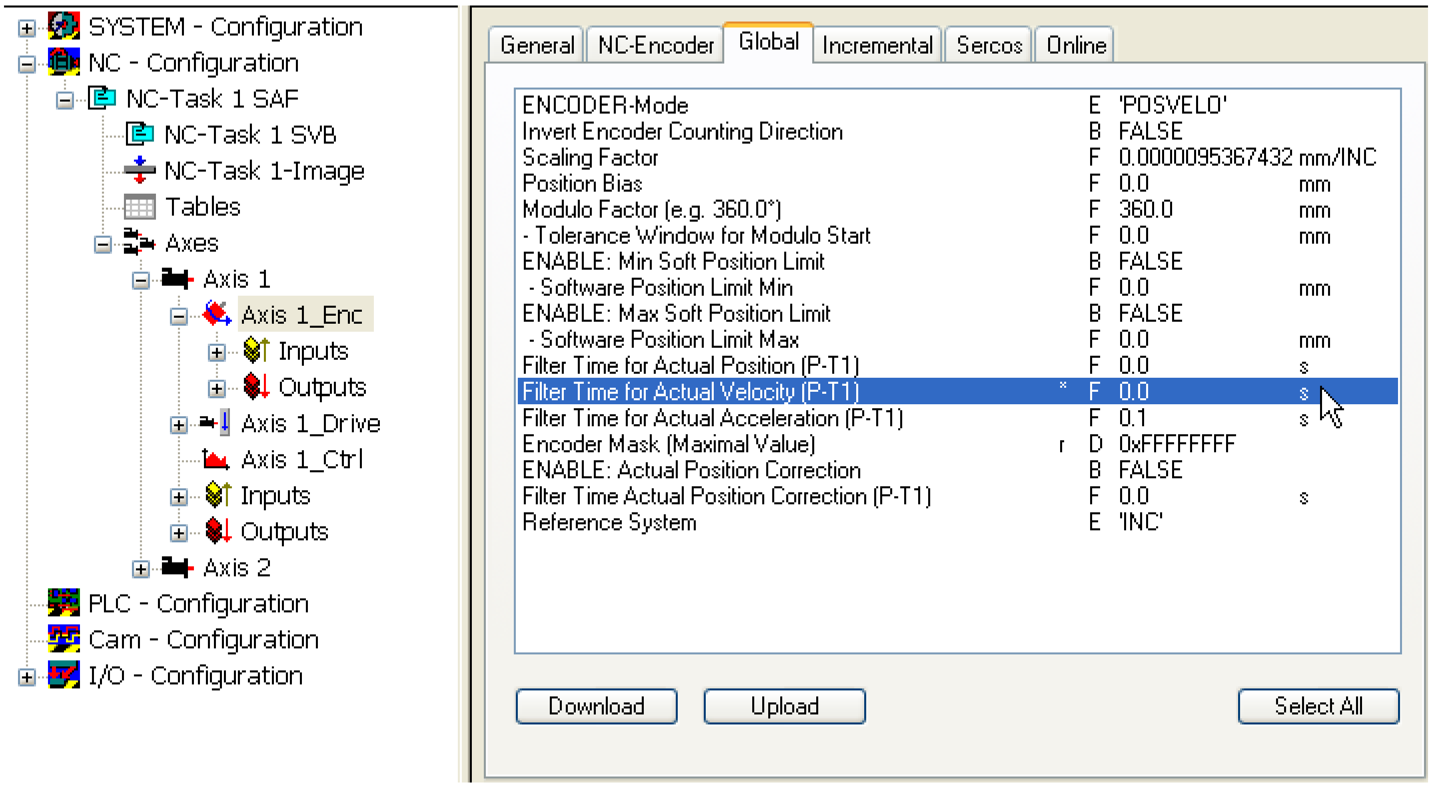
Velocity controller tuning



Setup preparations for velocity controller tuning:

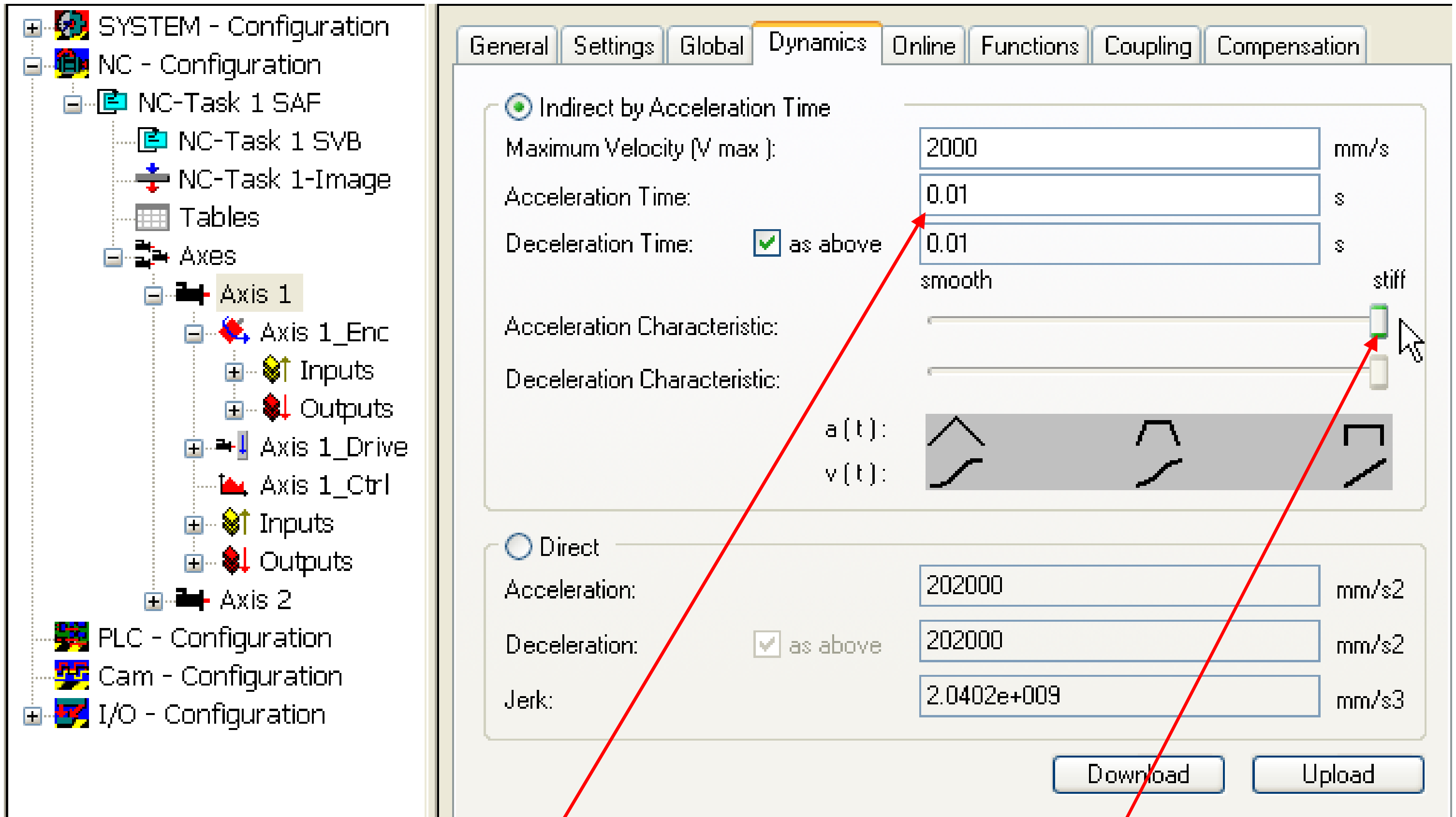
Very short ramps (< 20ms) are possible by this option of “Setpoint Generator Type”.

Velocity controller tuning



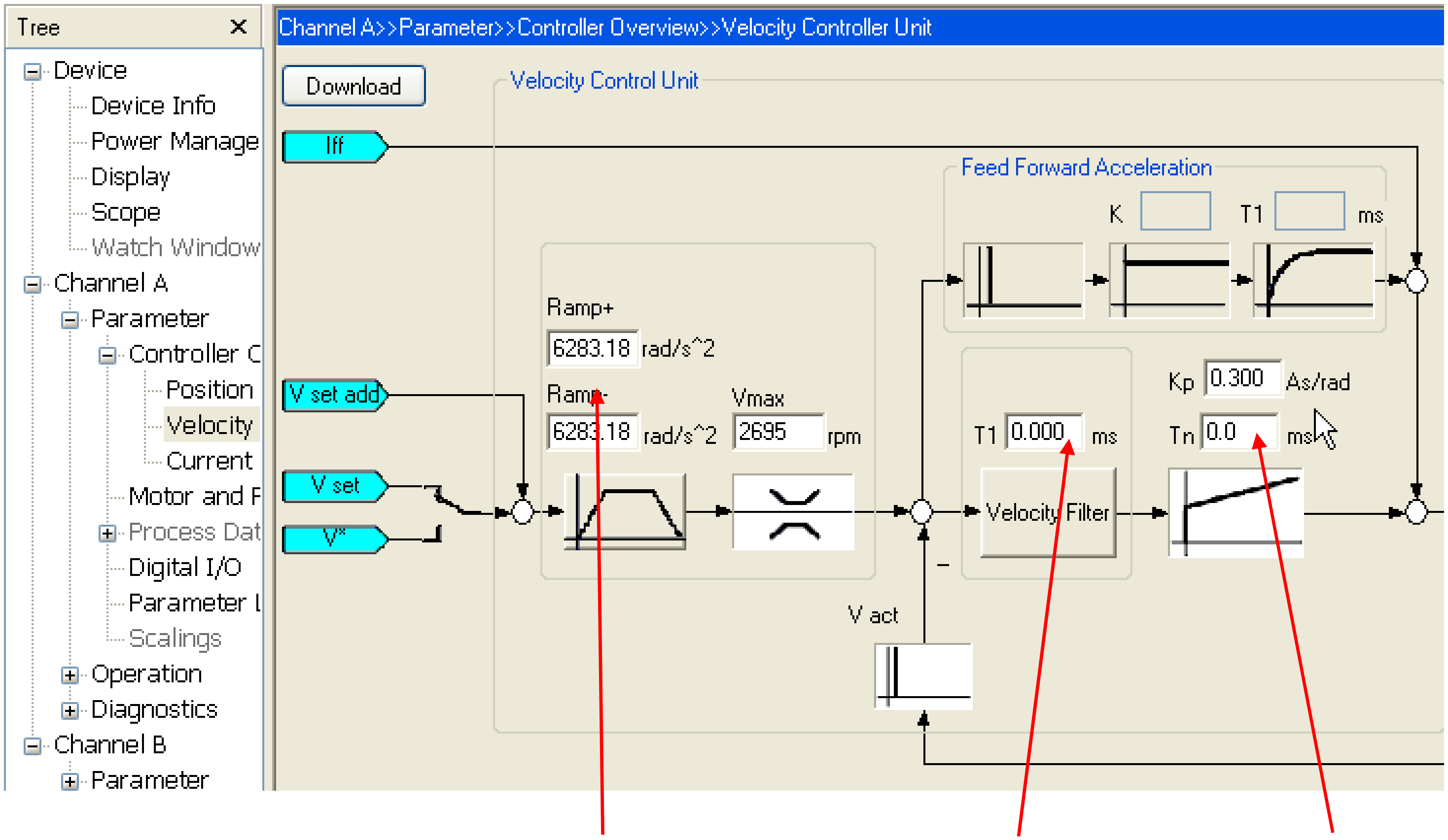
Set “Filter Time Actual Velocity (P-T1)” to 0.

Velocity controller tuning



Selection of short ACC and DEC ramps and nearly no Jerk limitation.

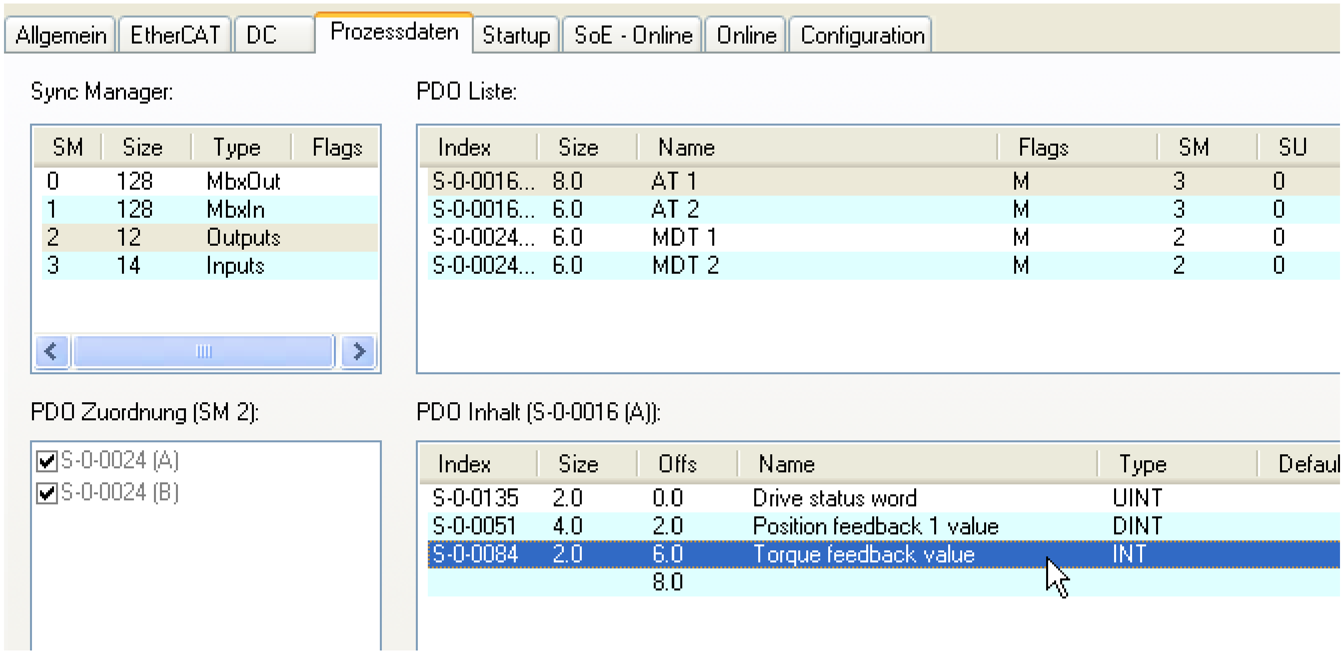
Velocity controller tuning



Selection of short Ramp+ and Ramp- . Switch off tacho filter and integral part.

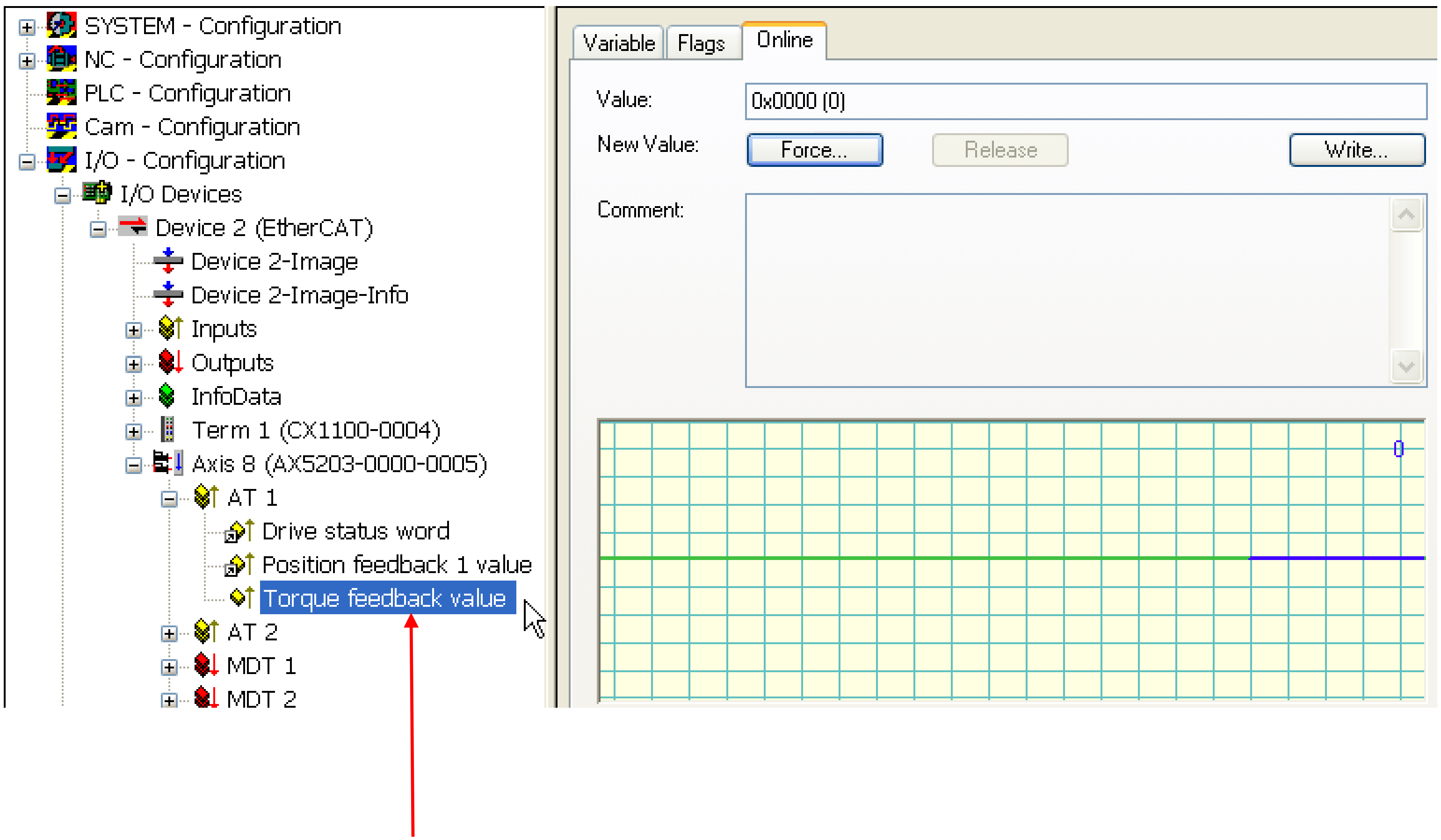
Drive tuning

Velocity controller tuning



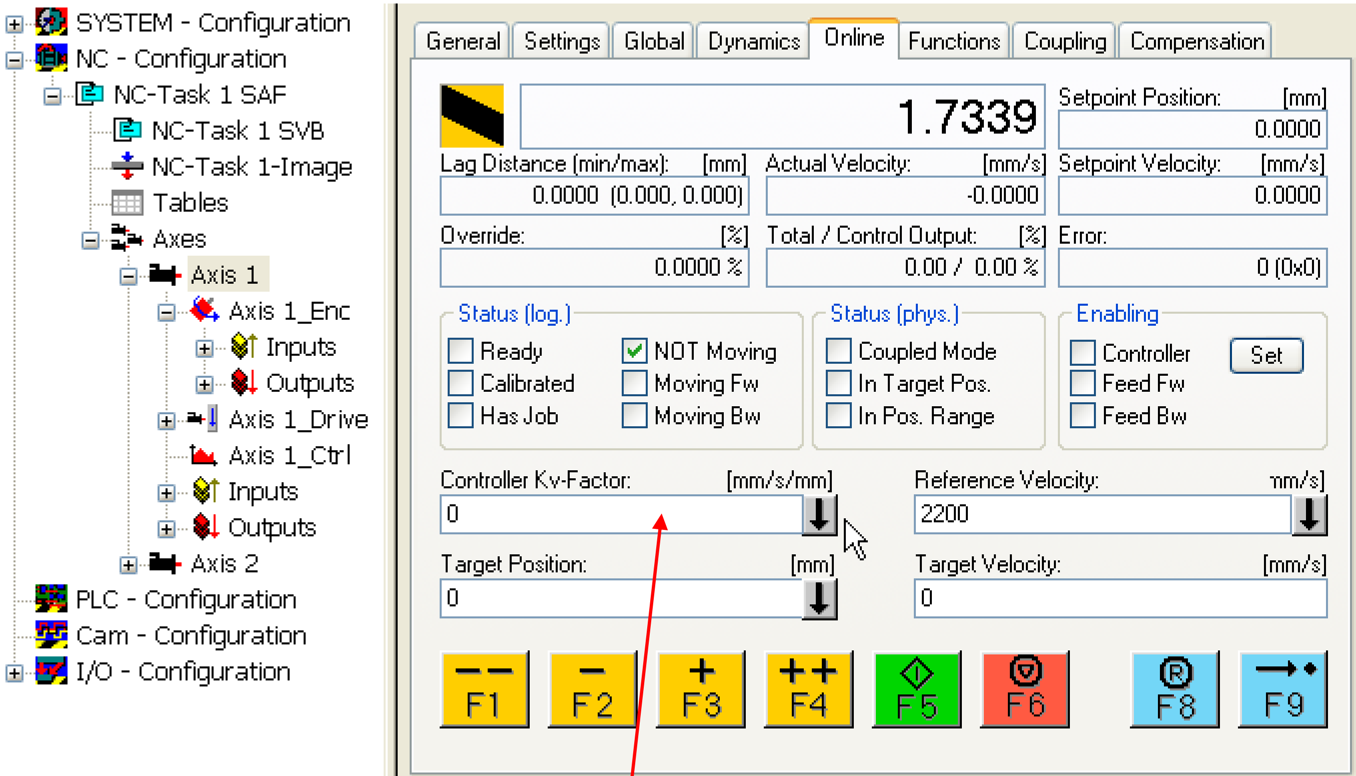
For „Step response“ estimation we have to map the actual current (Torque feedback value) into the Process data's.

Velocity controller tuning



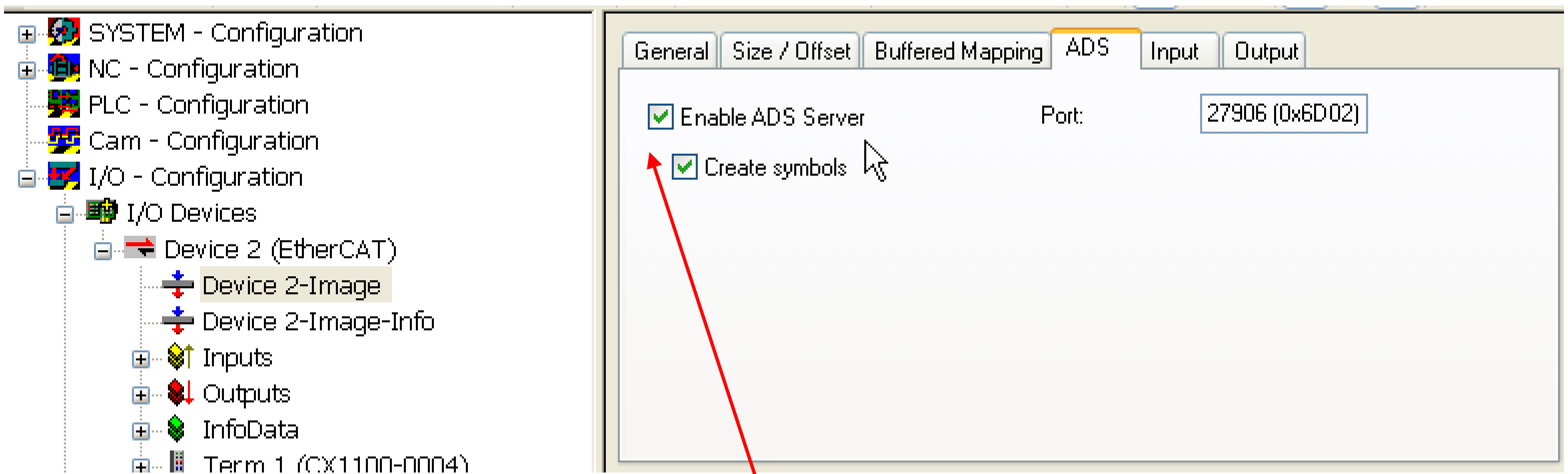
The actual current is a part of the Process Data.

Velocity controller tuning



Switch off the position controller Kv=0

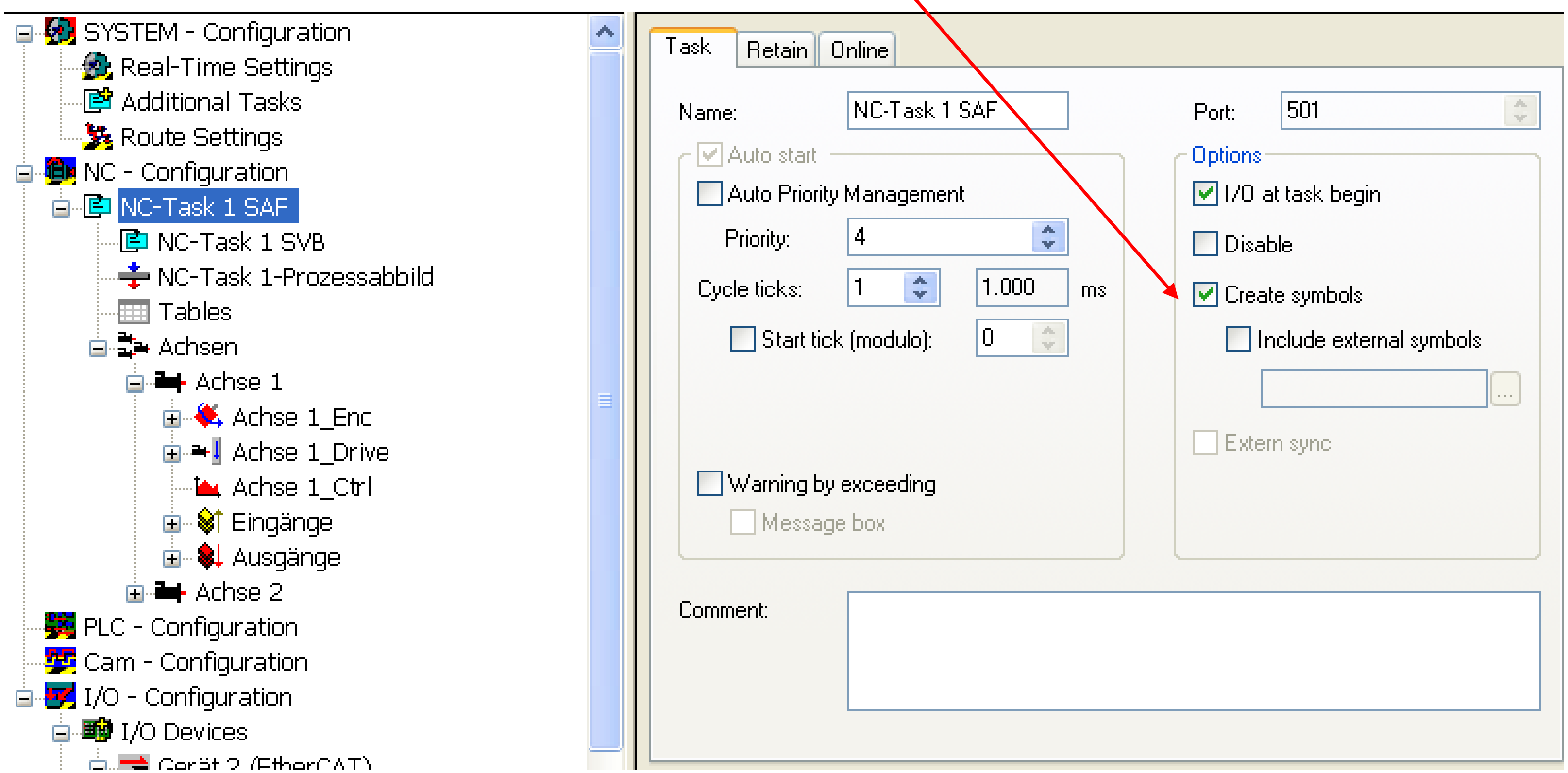
Velocity controller tuning



- For the scope function please select: “Enable ADS Server“ and “Create symbols“.

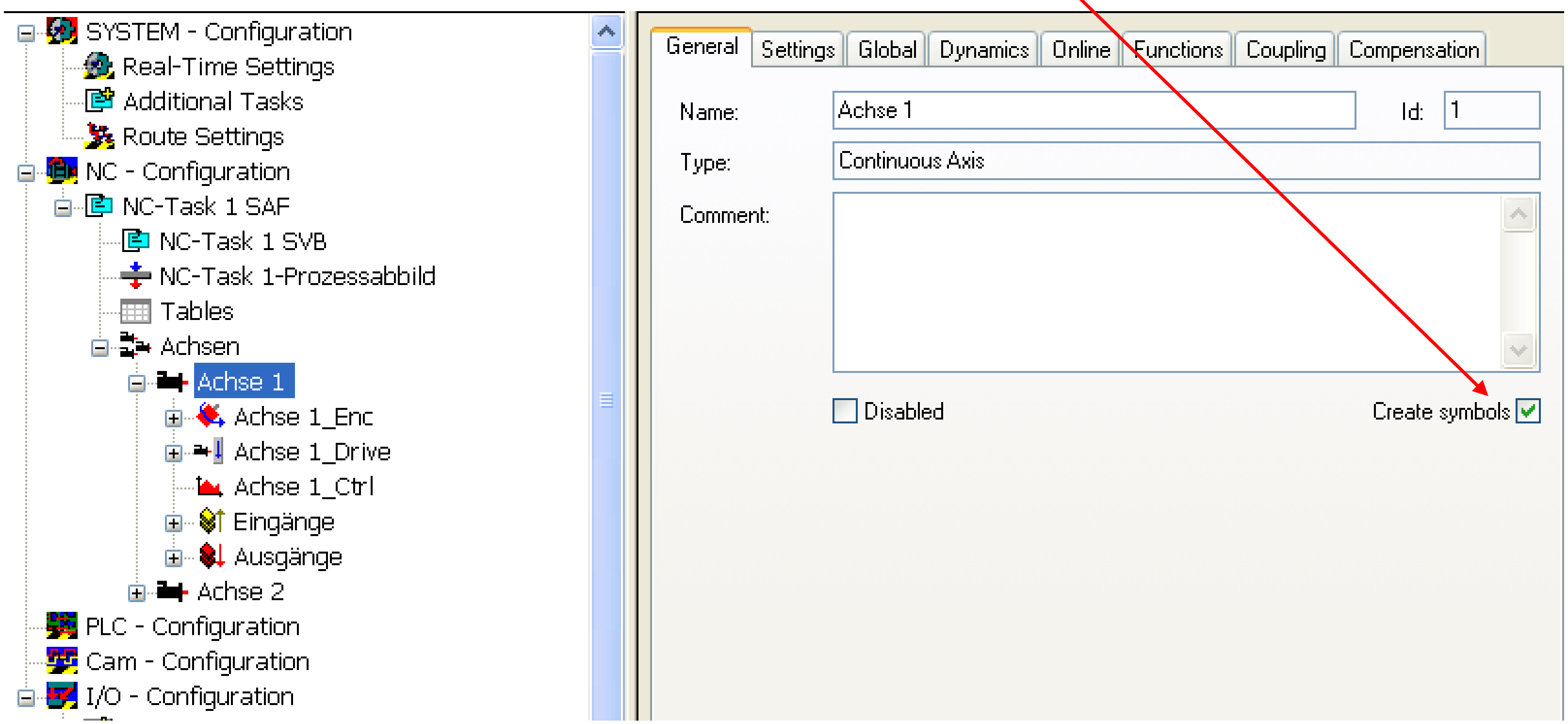
Velocity controller tuning

- Scope view configuration: Select “Create symbols”



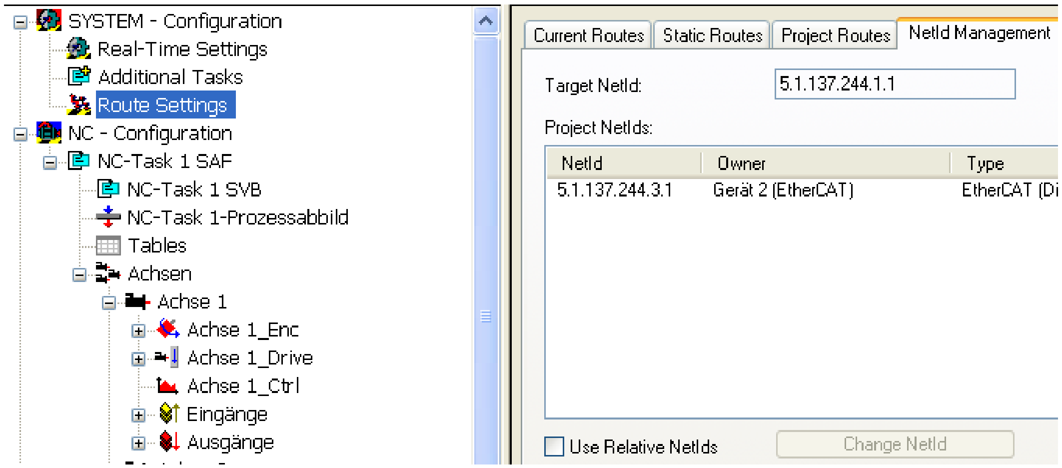
Velocity controller tuning

Scope view configuration: Select “Create symbols”

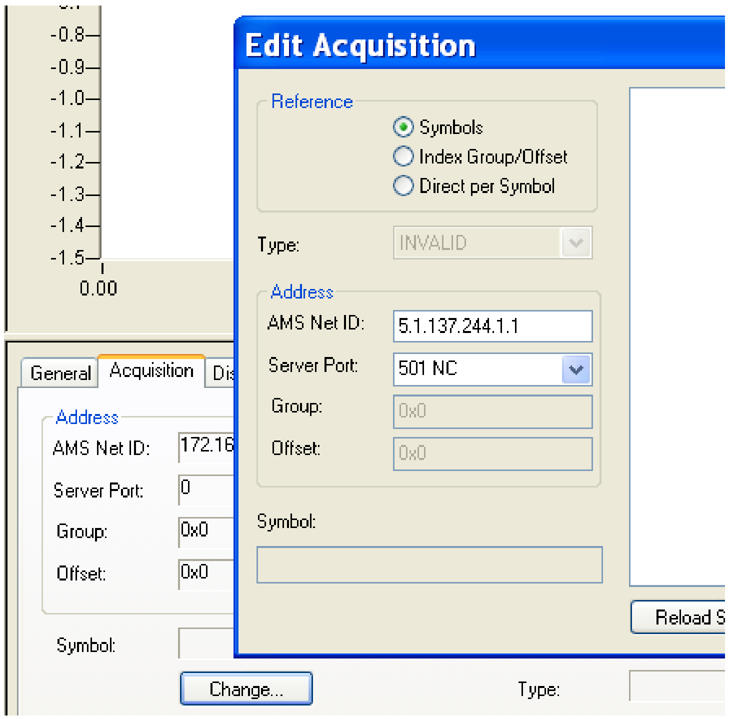


Drive tuning

Velocity controller tuning



- Scope view configuration:
- Read the "AMS Net ID" from here:
- And give it there:

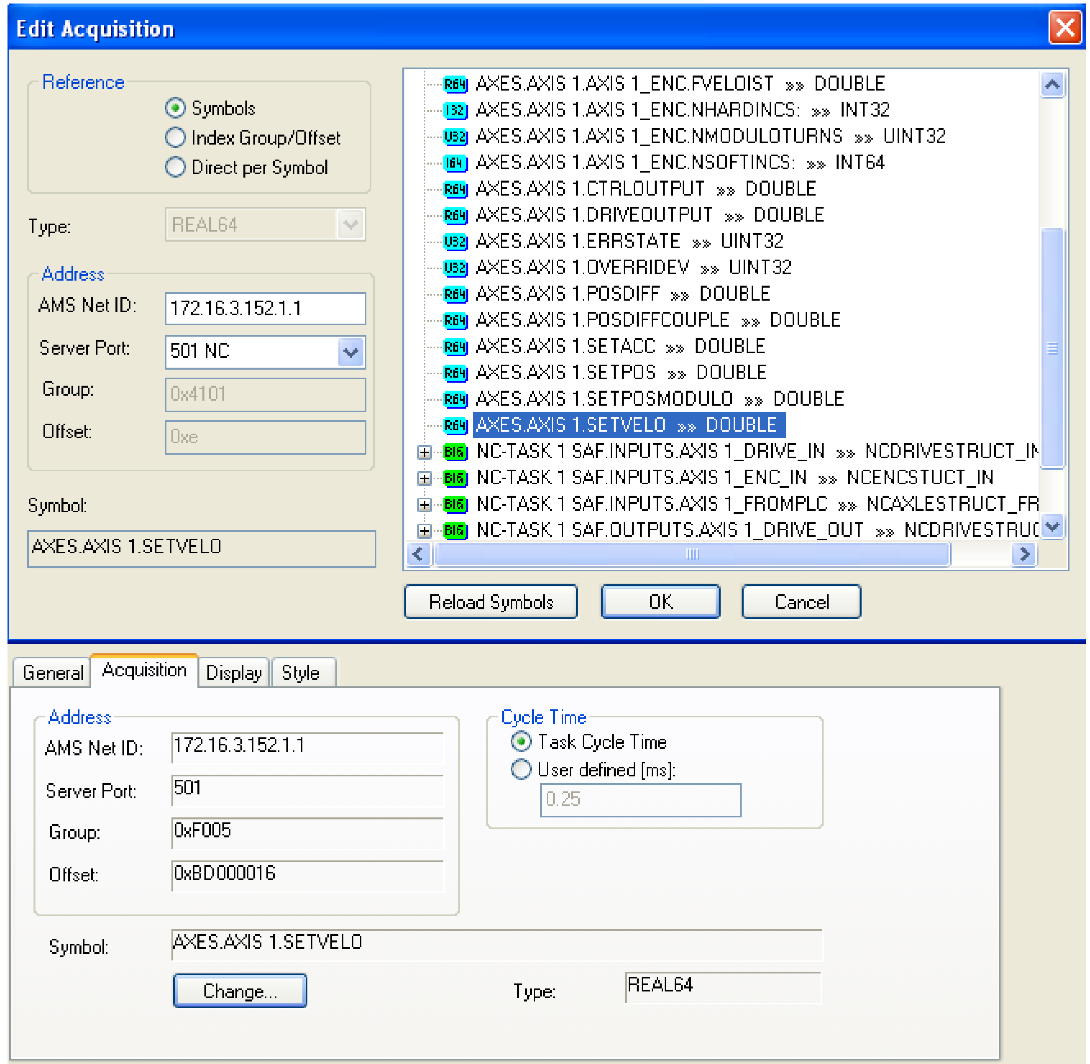


Drive tuning

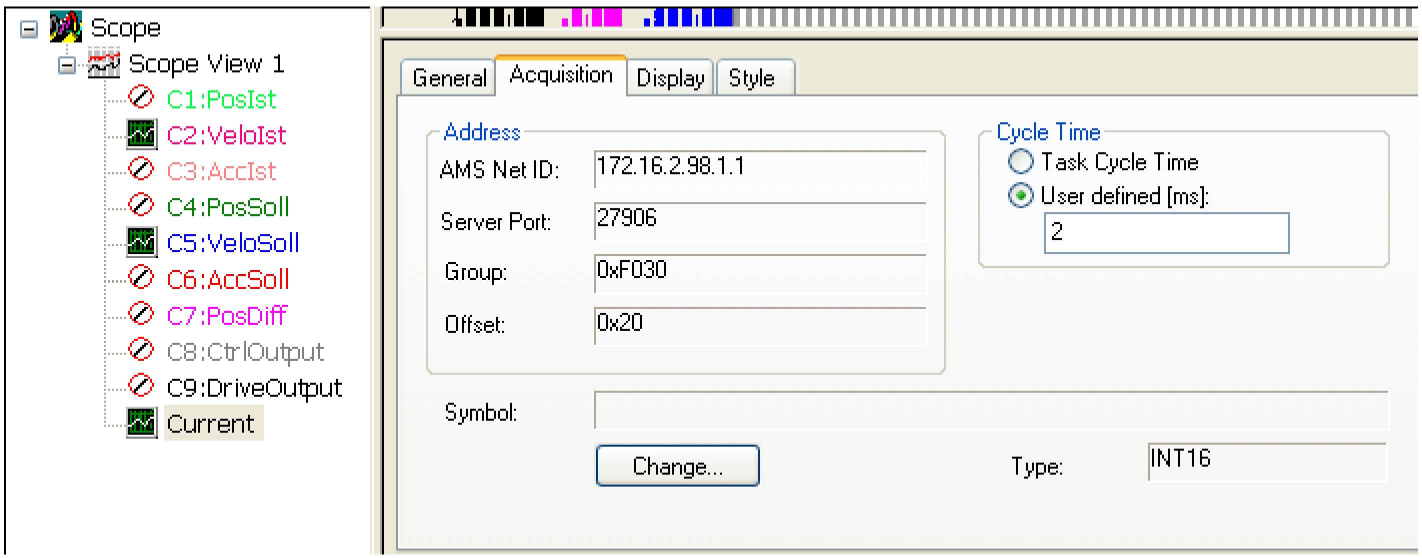
Velocity controller tuning

Scope view configuration:

- Add the set velocity and
- actual velocity to the scope
- view channel

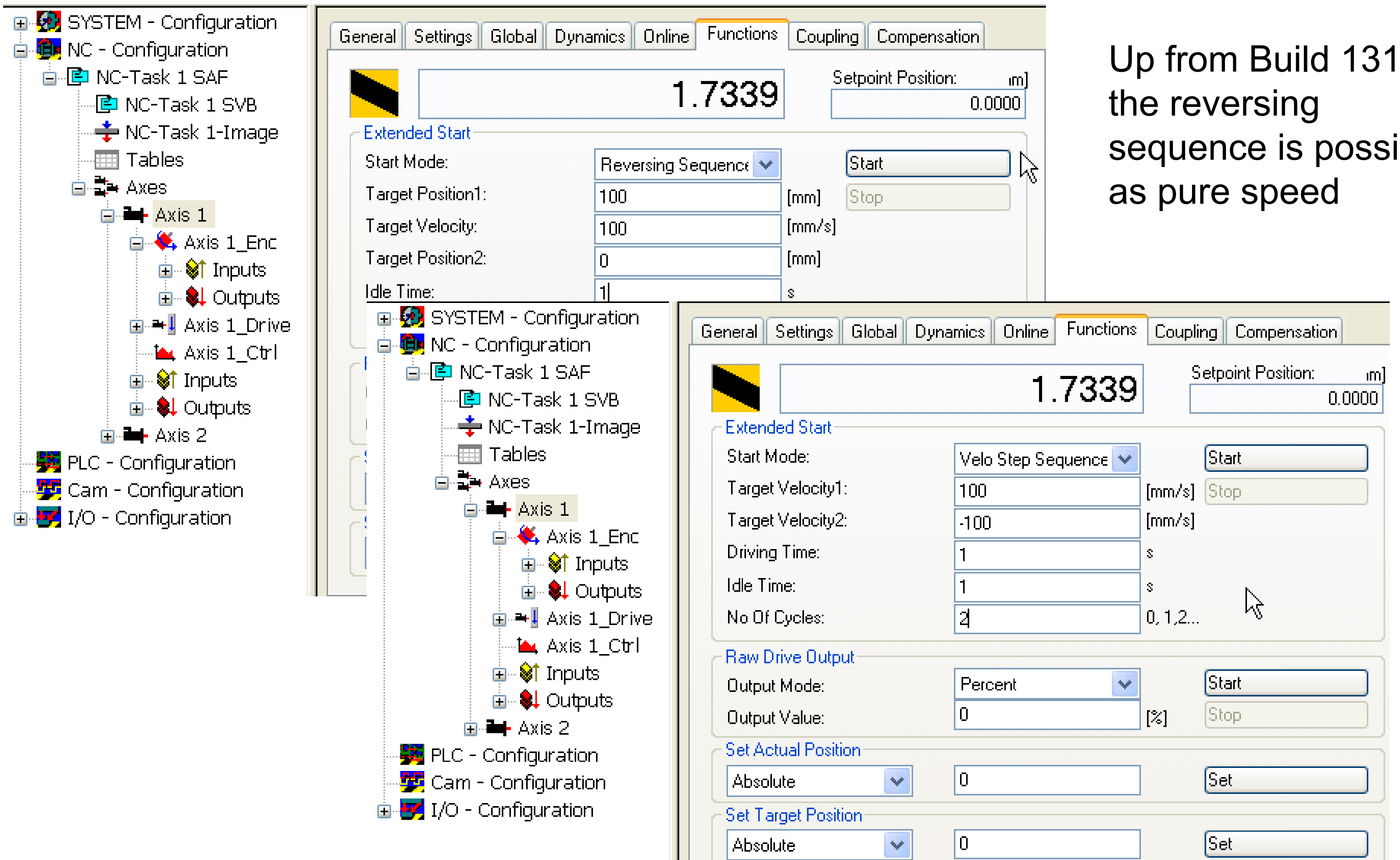


Velocity controller tuning



Or take basic „Scope View“ selection by „Achse1.scp“ and add actual current to the Scope.

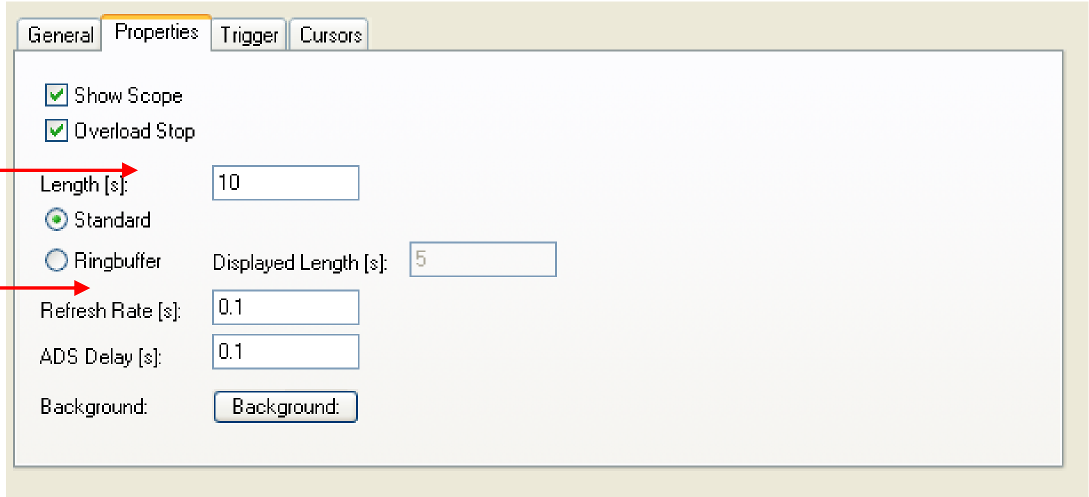
Velocity controller tuning



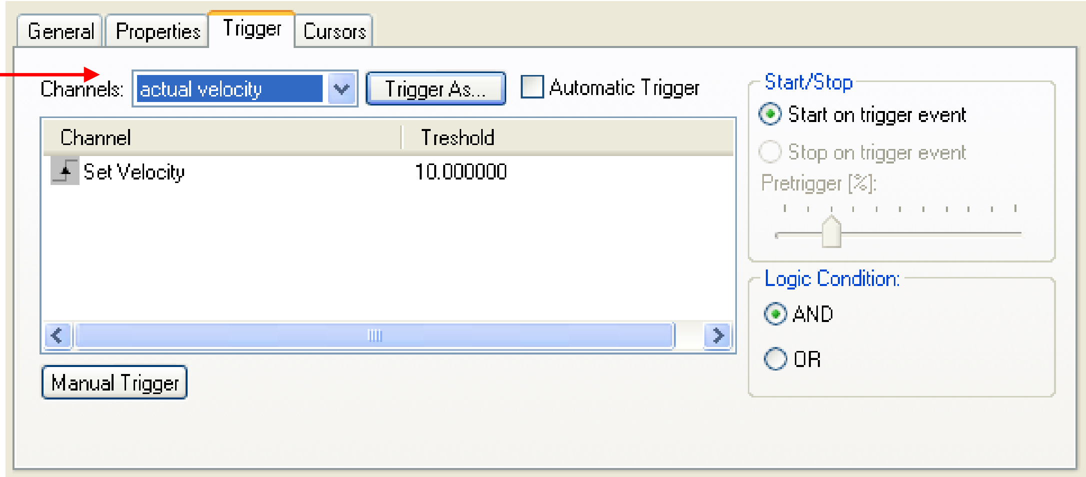
Up from Build 1316 the reversing sequence is possible as pure speed

Velocity controller tuning

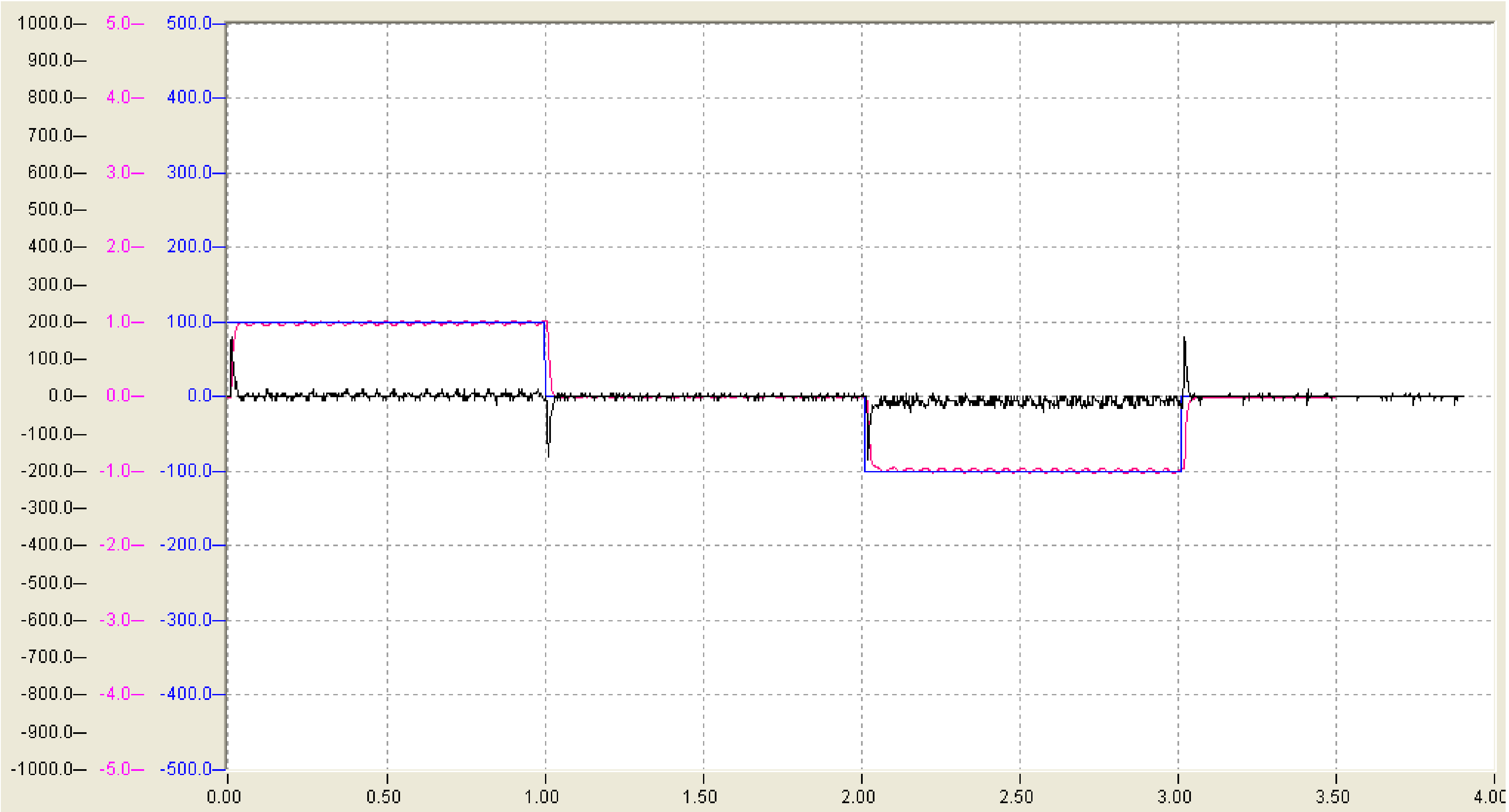
- Set the scope record length
- Set the scope refresh rate



- Set the trigger of the scope

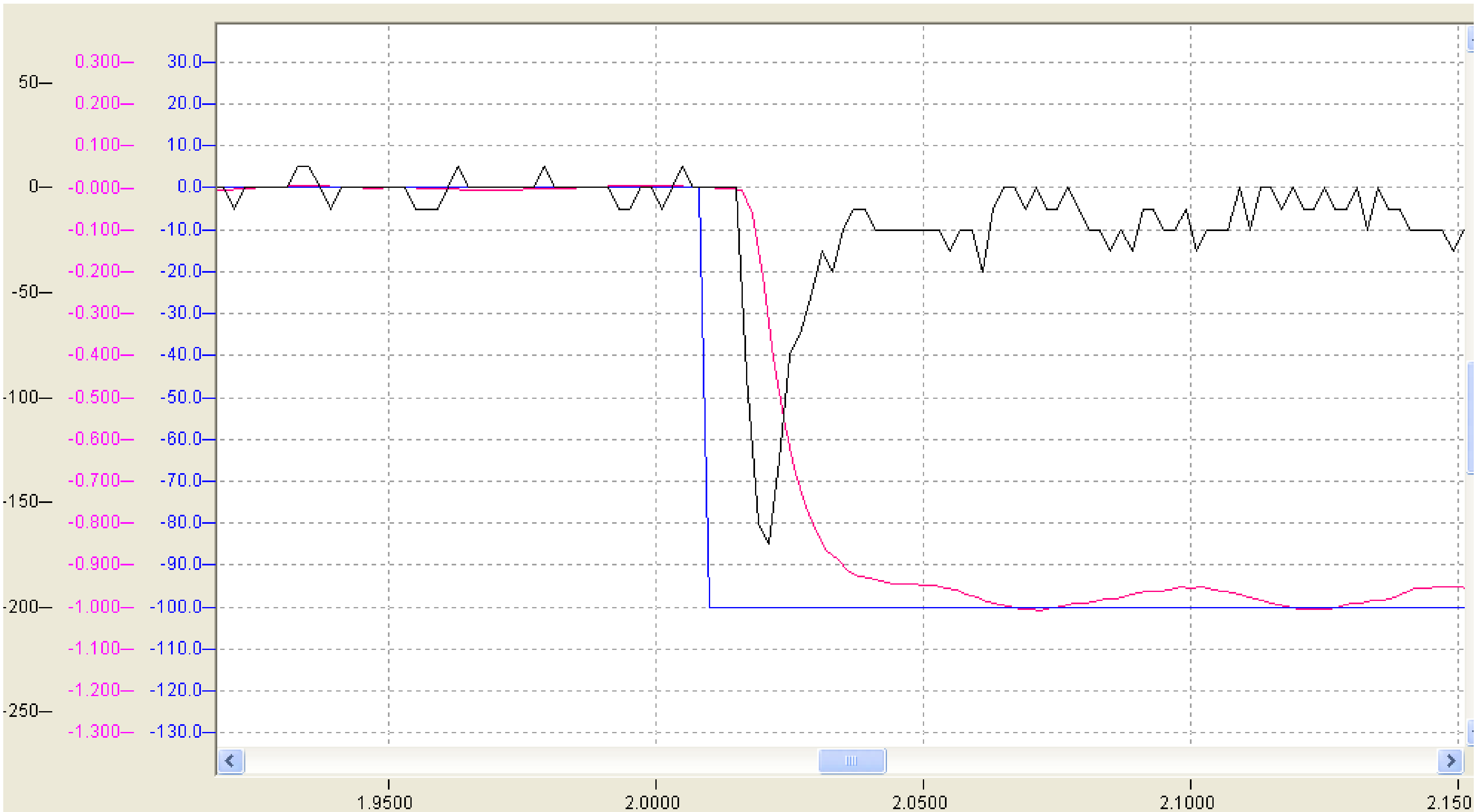


Velocity loop “step response”



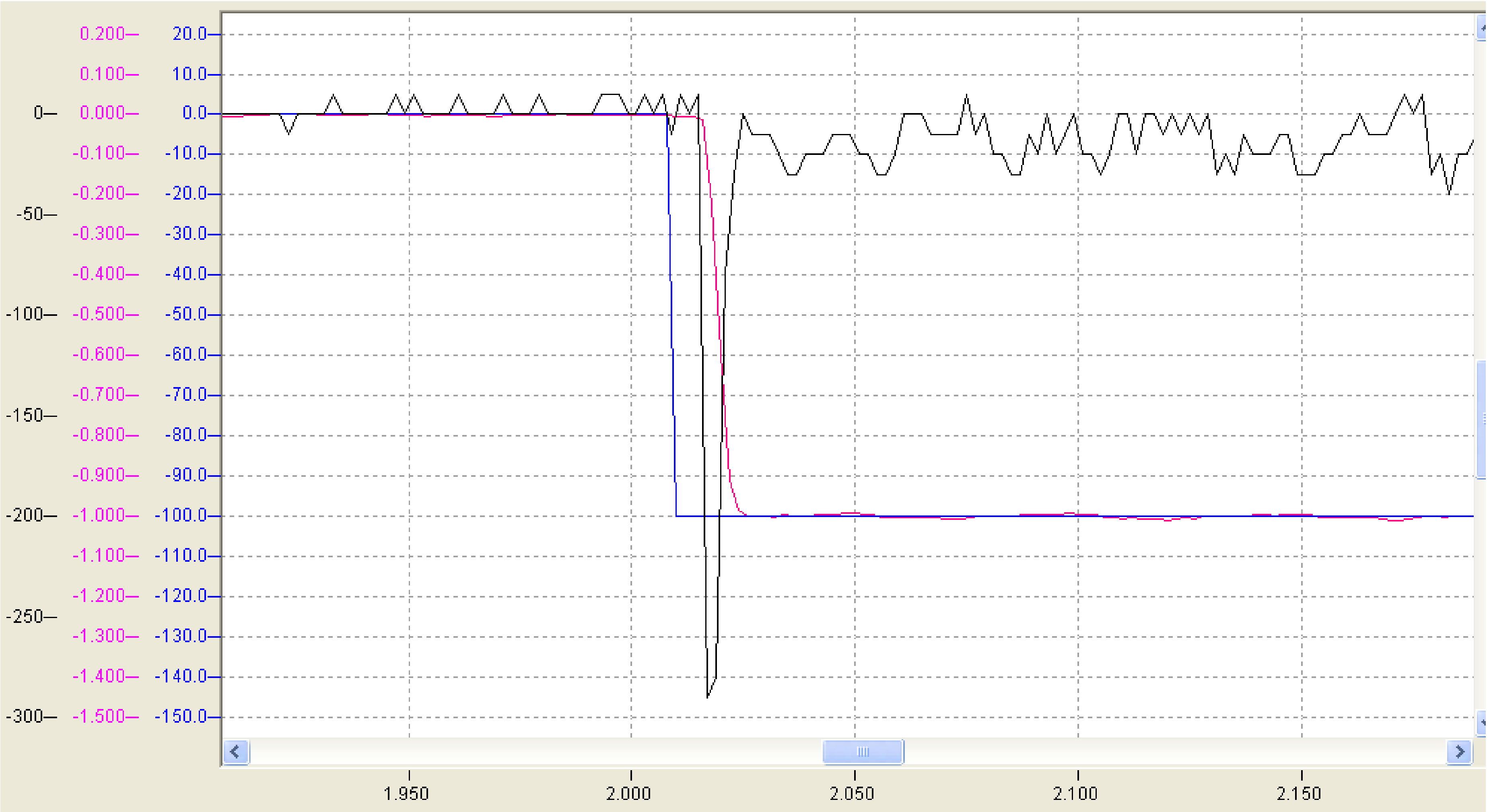
Tn = 0 Kp = 0,2

Velocity loop “step response”



Tn = 0 Kp = 0,2

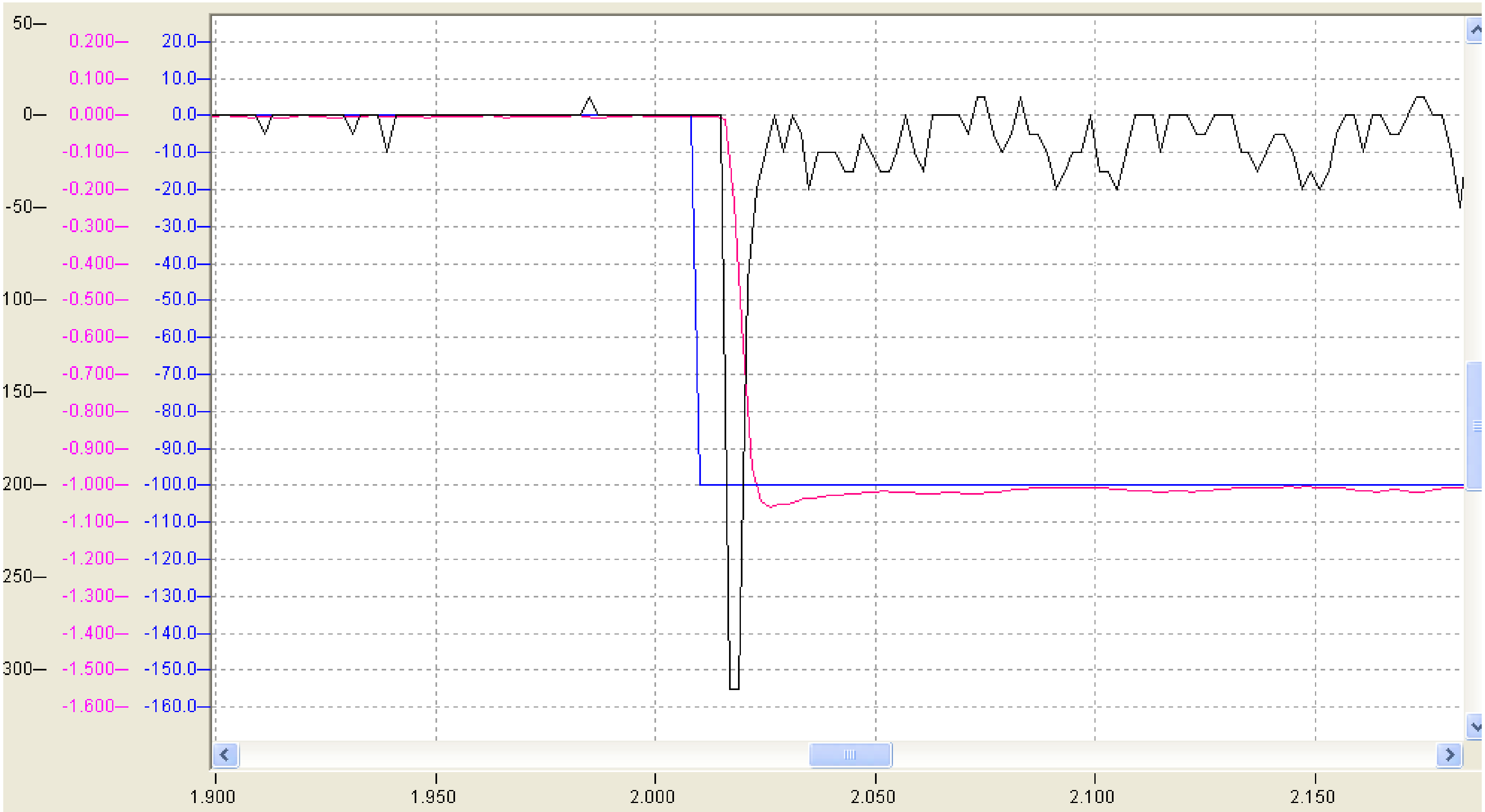
Velocity loop “step response”



Tn = 0 Kp = 1,0

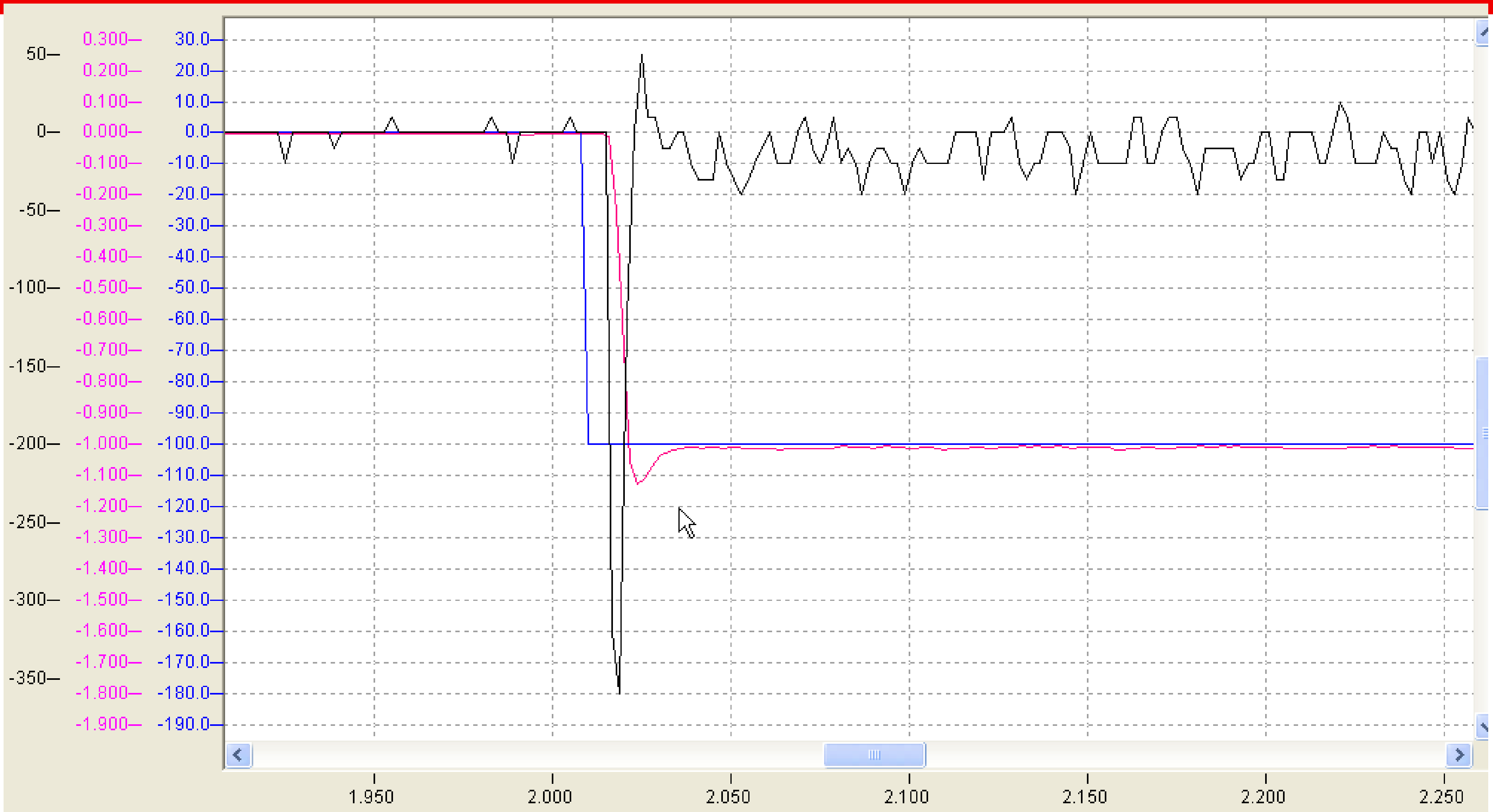
Drive tuning

Velocity loop “step response”



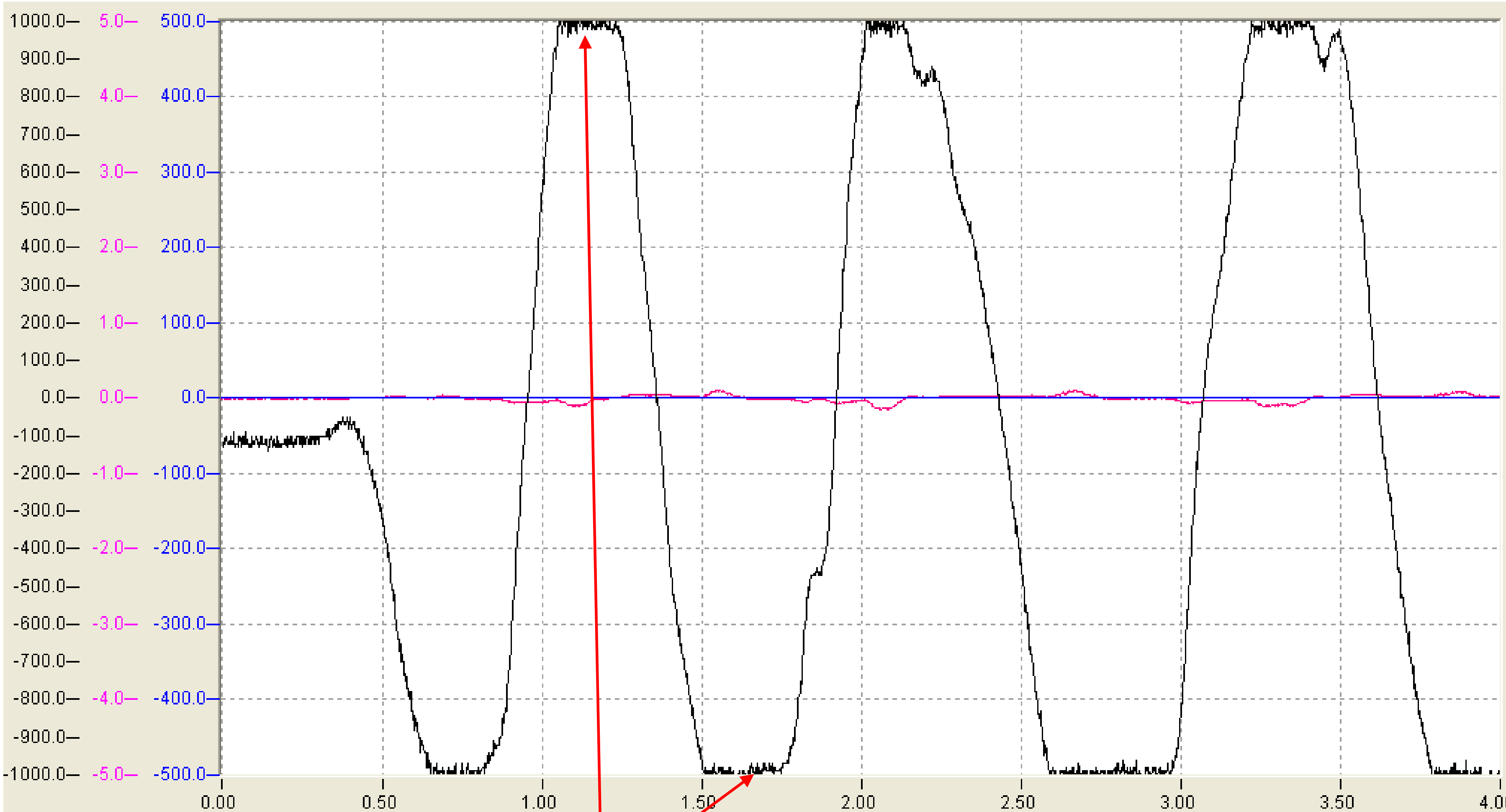
Tn = 20 Kp = 1,0

Velocity loop “step response”



Tn = 5 Kp = 1,0

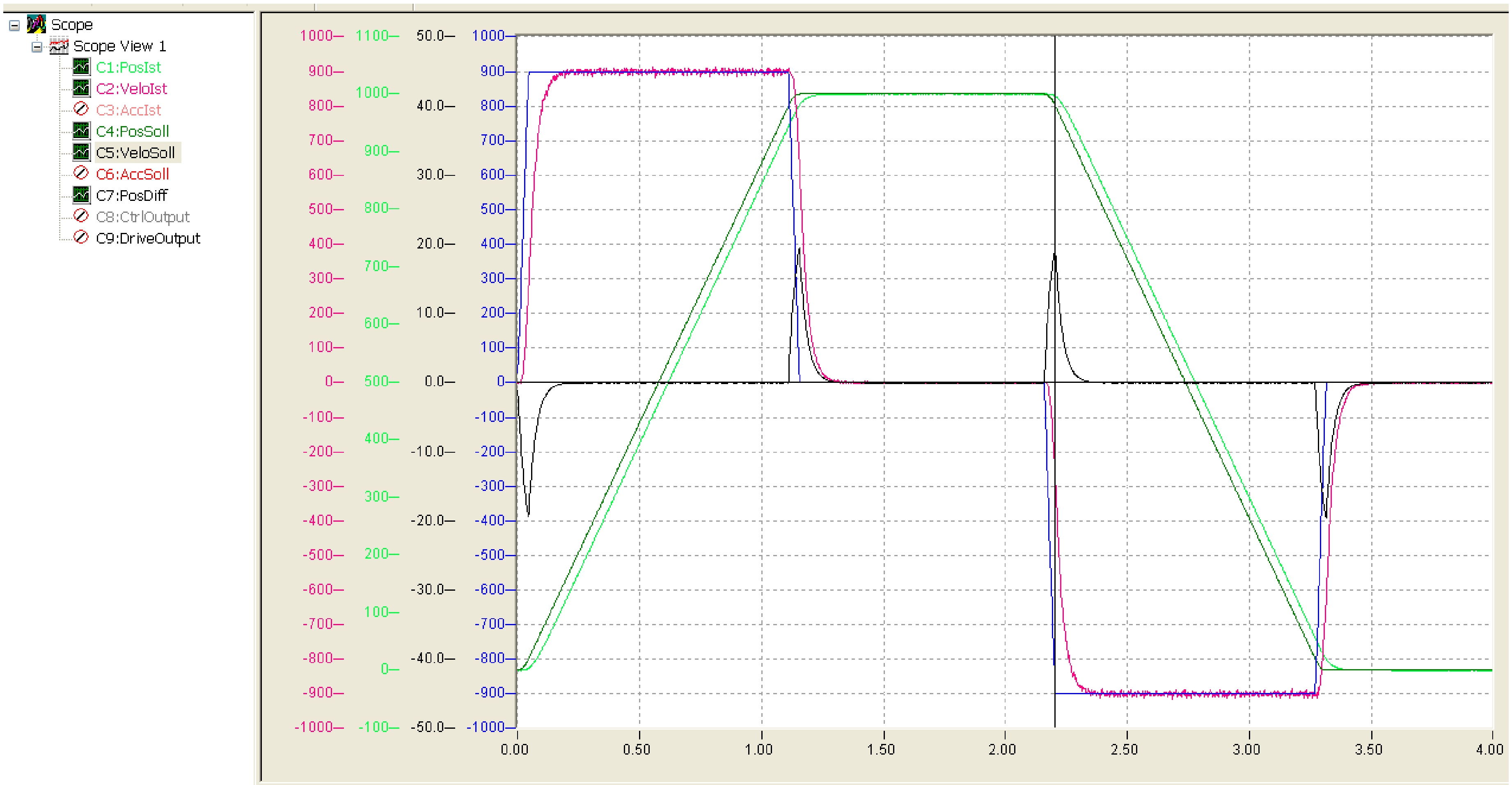
Velocity loop “step response”



During the step response keep the current away from saturation.

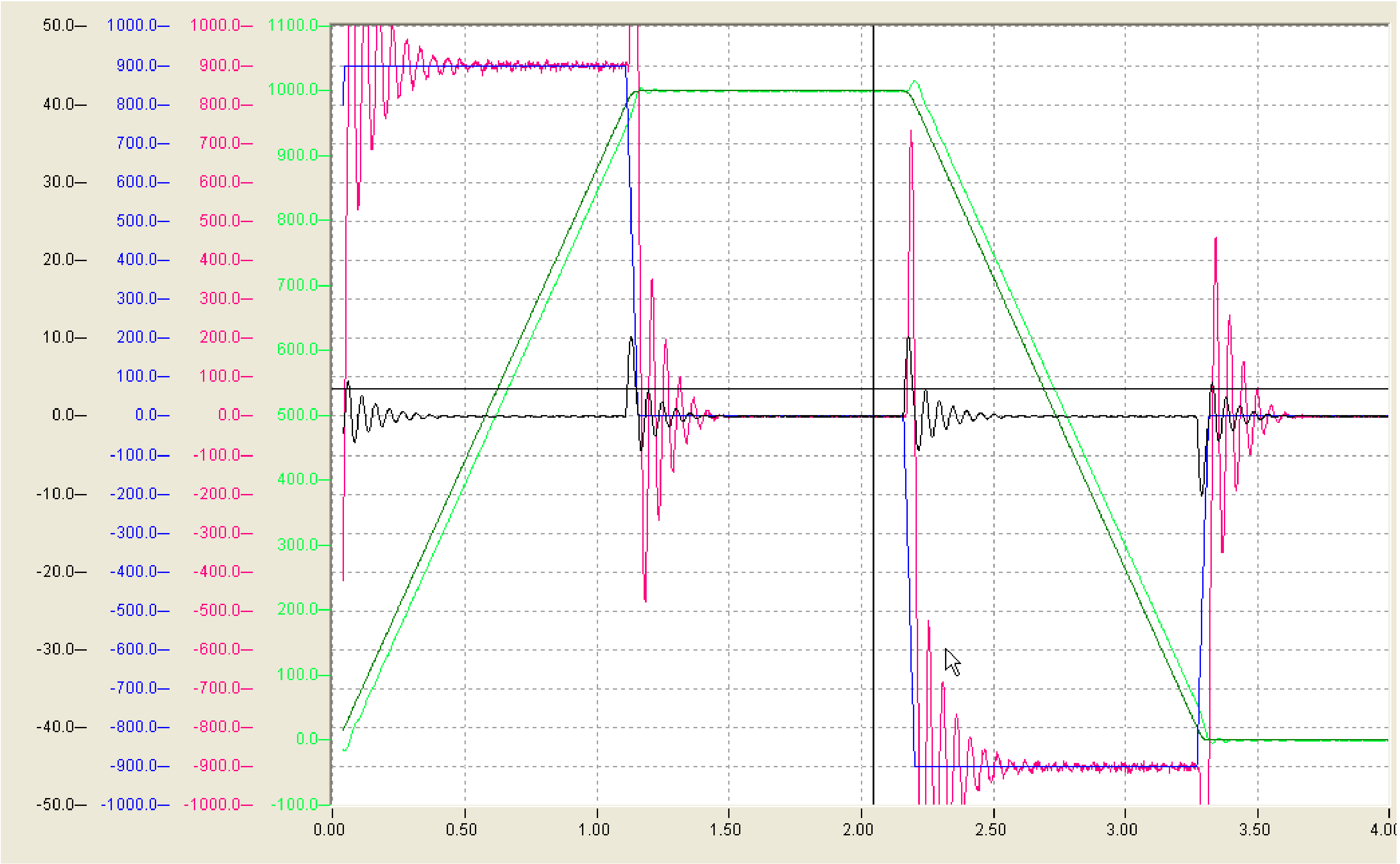
Position controller tuning

- Right scaling of KV



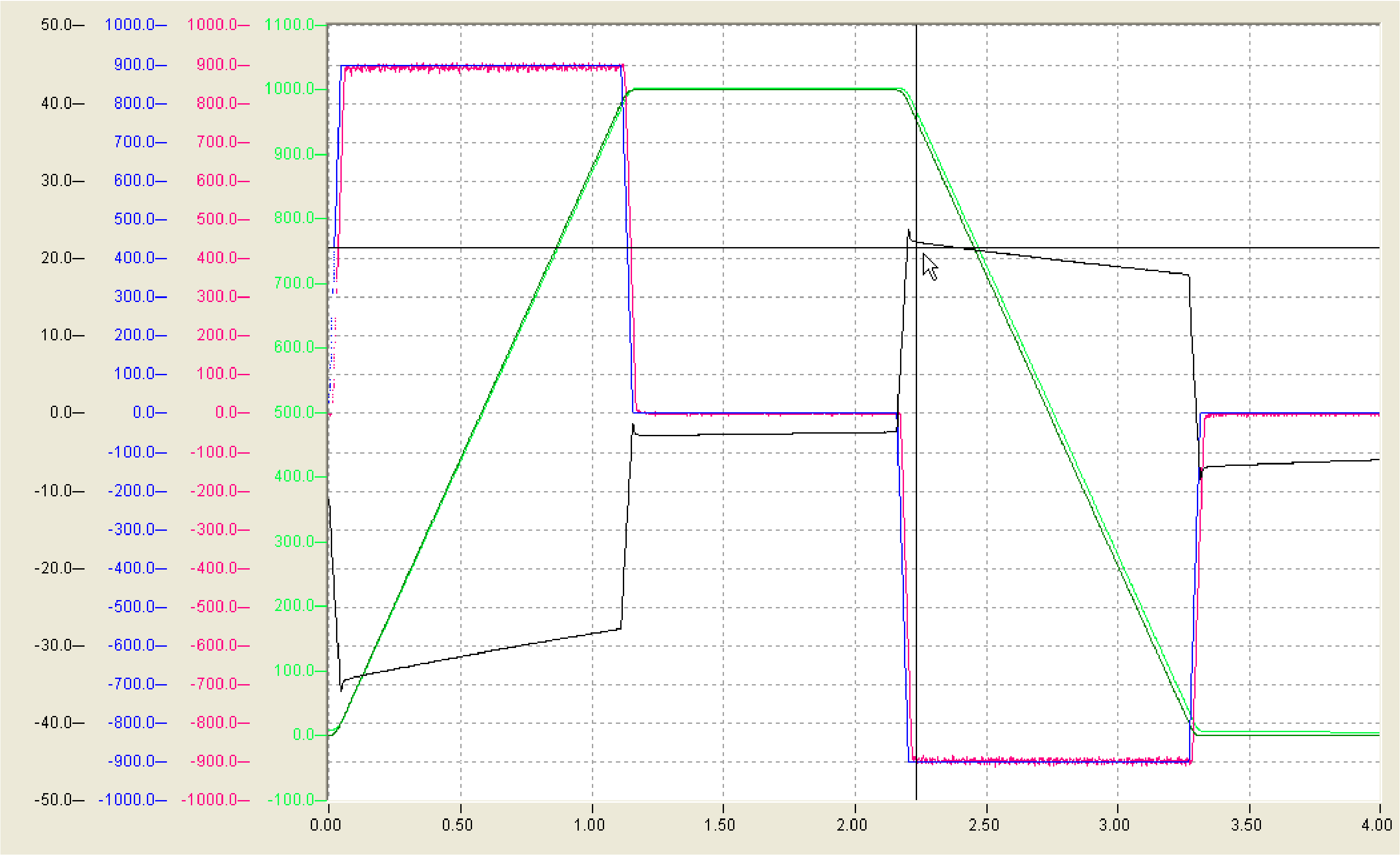
Position controller tuning

- KV to high



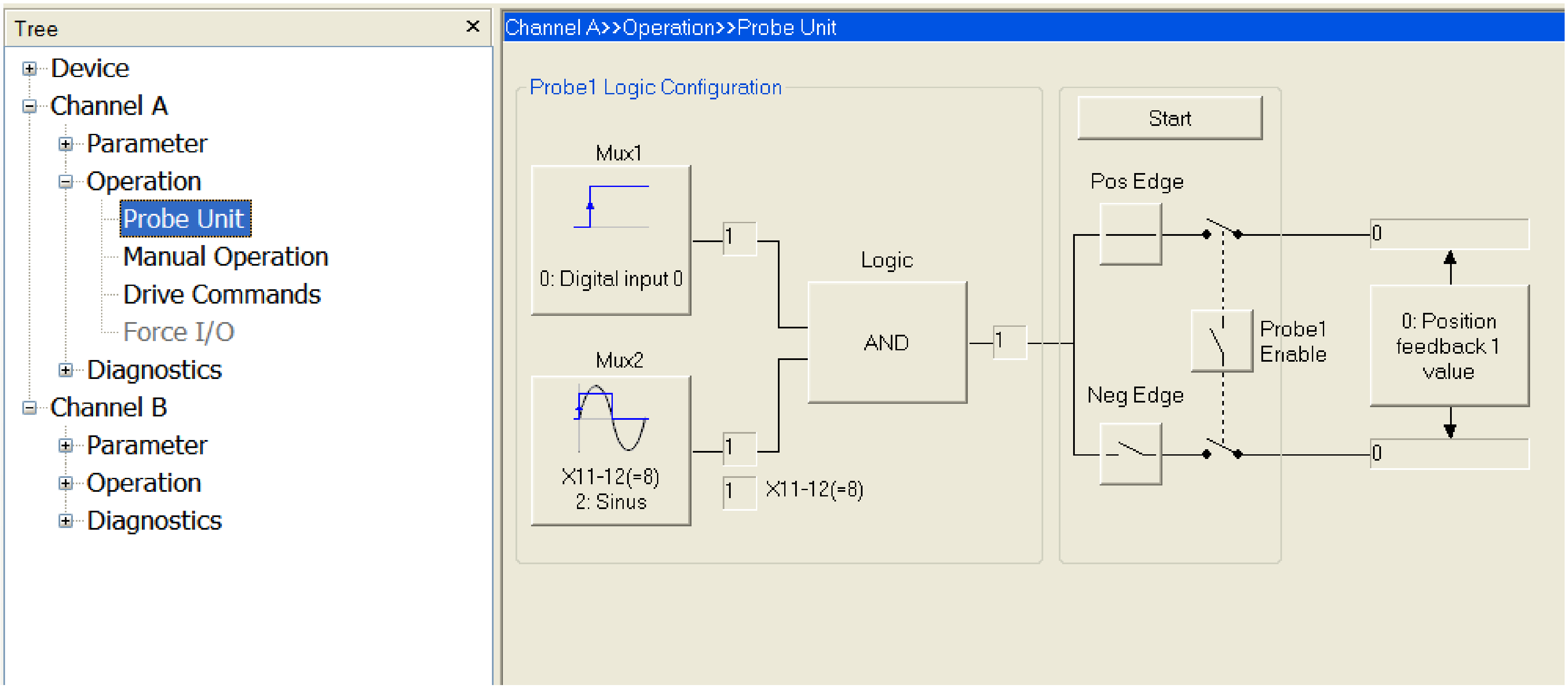
Position controller tuning

- KV to low



Probe Unit

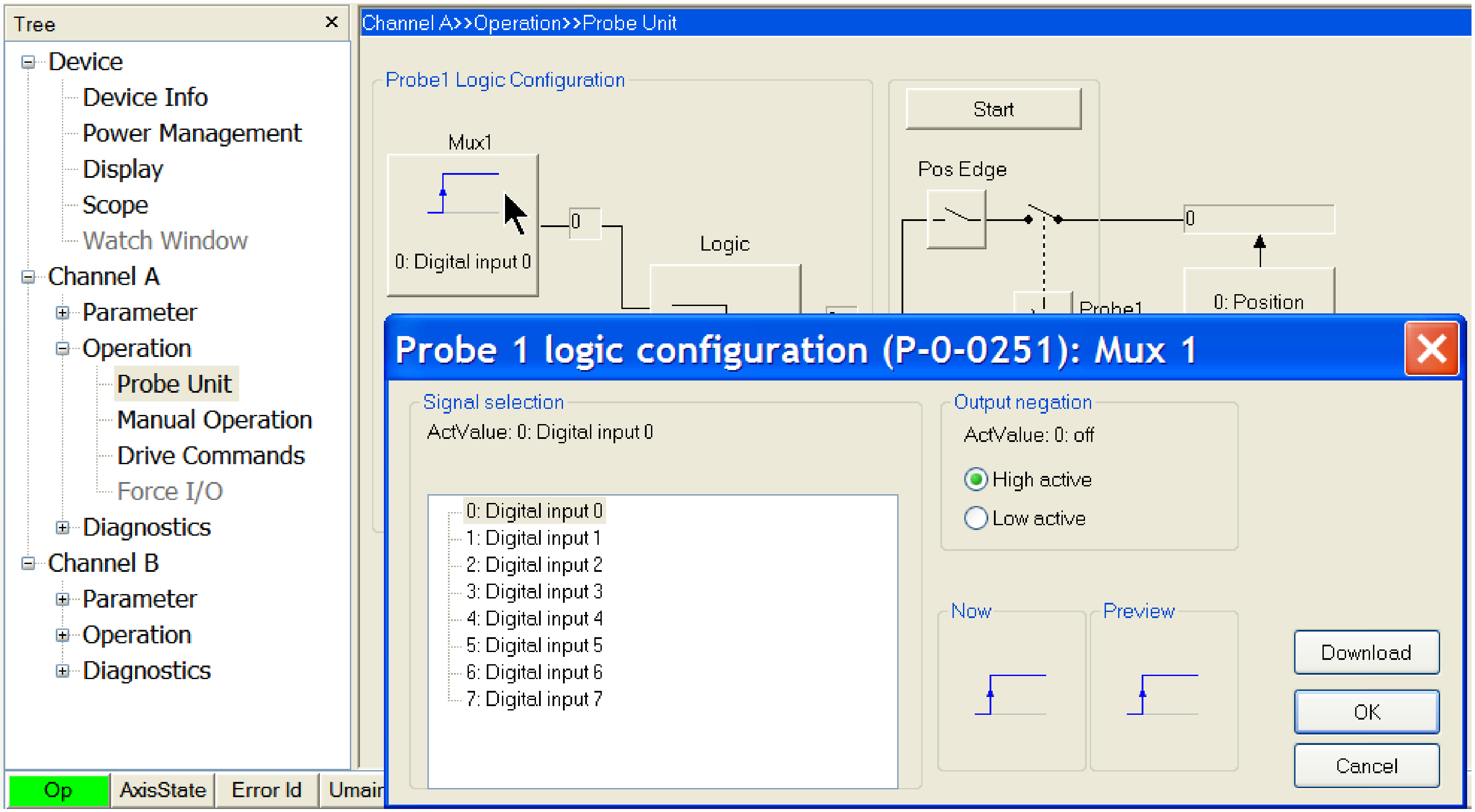
Operation



The Probe Unit gives the possibility to select different latch and "Homing" alternatives.

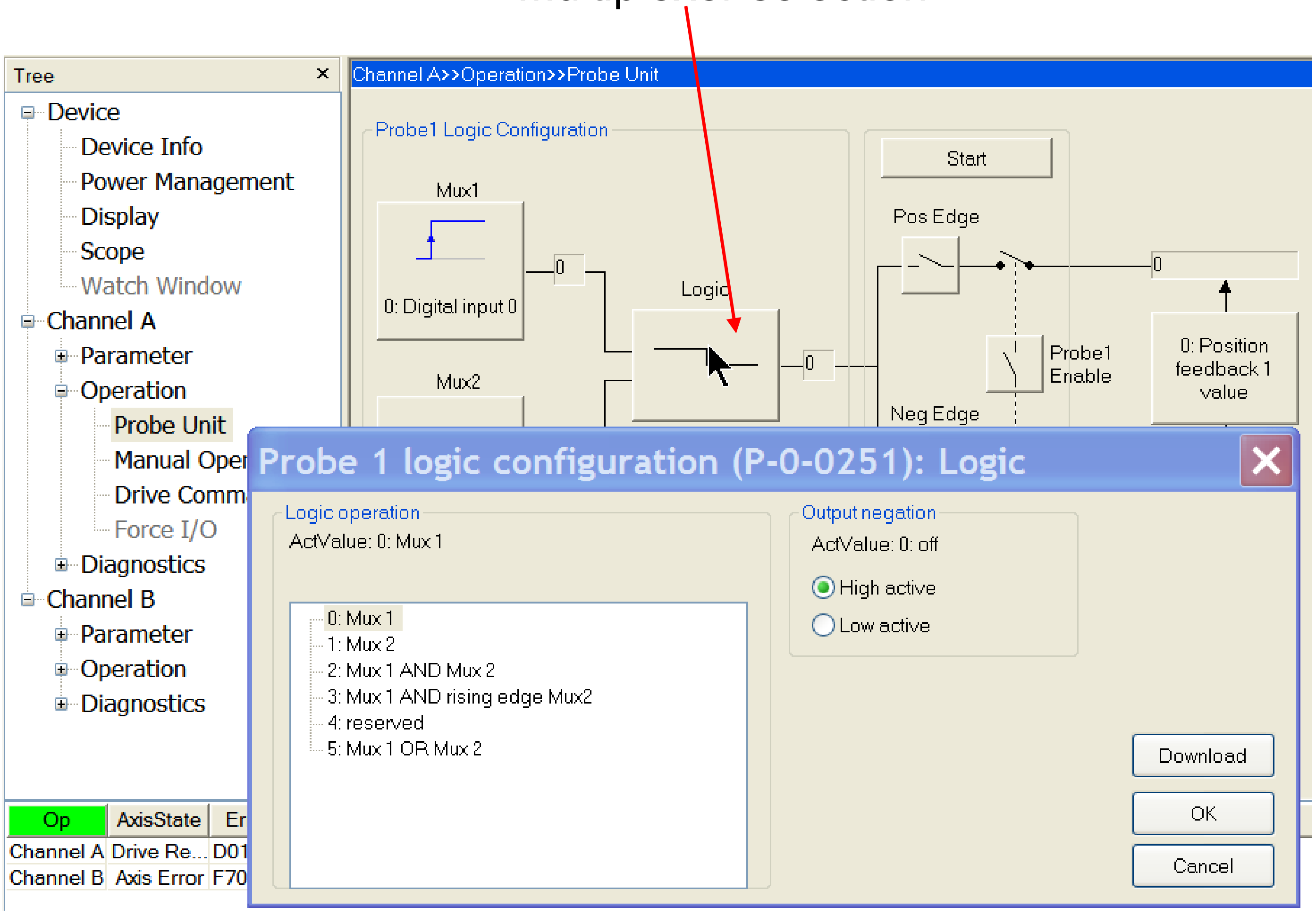
Operation, Position Latch

In case of „position latch“ the Probe Unit can configured in that way:

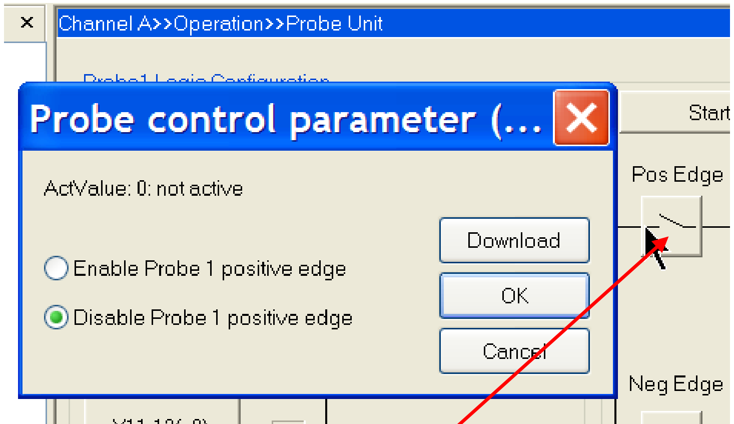


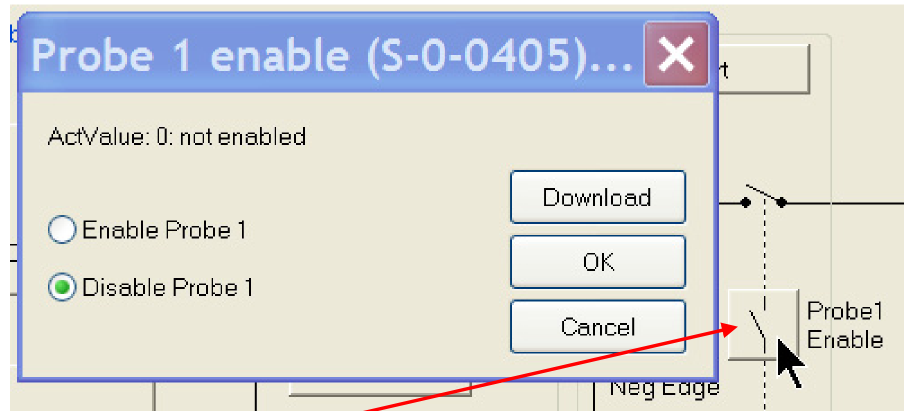
Operation, Position Latch

Multiplexer selection

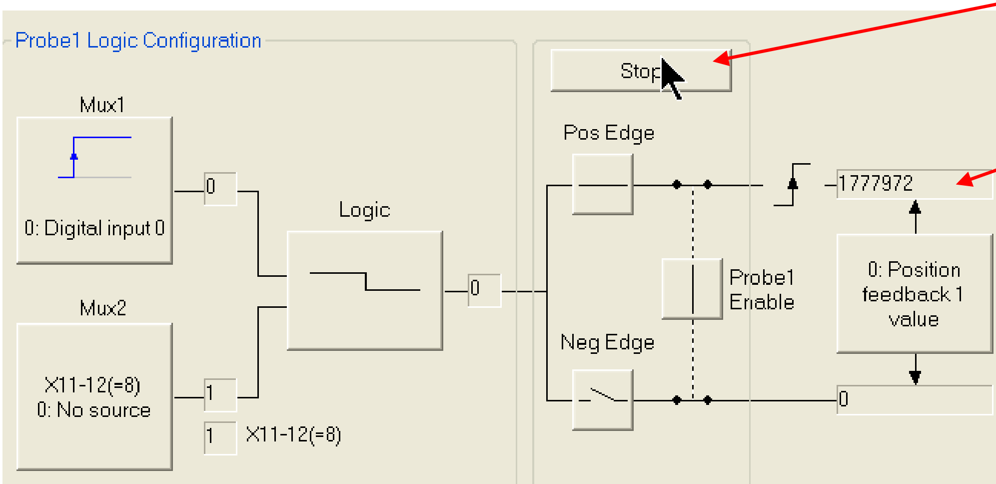


Operation, Position Latch





Select „Probe control“ and „Probe 1 enable“, now the latch start/start is possible.

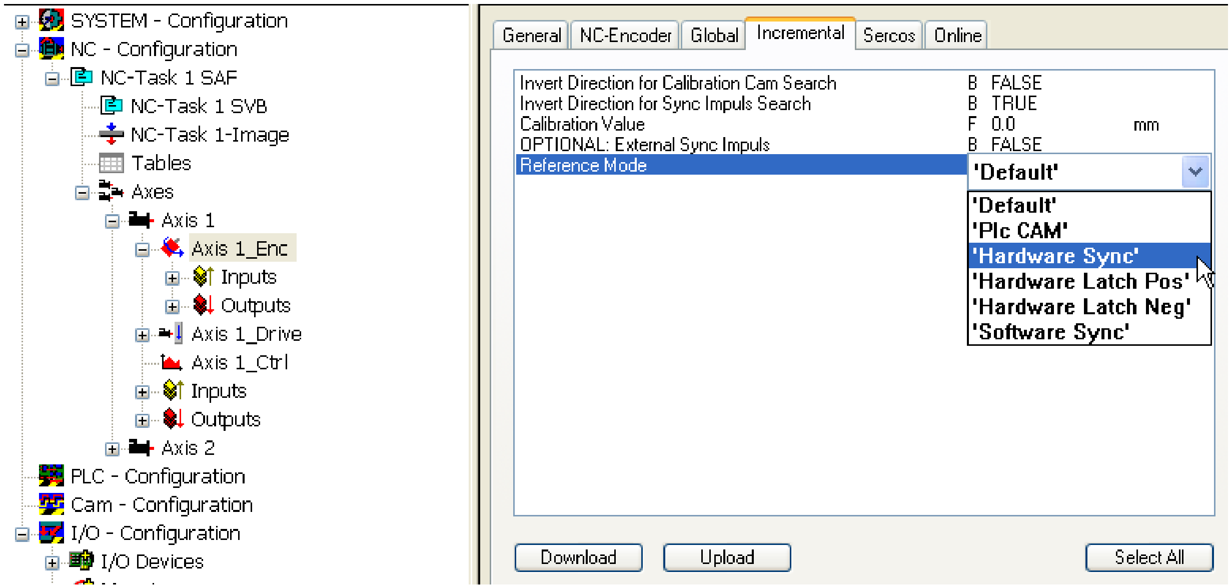


The latched value is displayed here.

Operation, Position Latch

- It is possible to do Position Latch through NC PTP and PLC.
- In order that NC PTP and PLC could control the probe unit, IDN S-0-0405 and S-0-0409 have to be mapped into the real-time-control status bit 2. This is done by the IDN S-0-0303 and S-0-0307 entry.
- Add latched value to „ Process data“
- The Probe Unit configuration could be done by the “Startup List” or during the axis operation by the FB of TcSOE,
- Start /Stop executing Probe (S-0-0170) could be operated by the FB of TcSOE,
- The latched position could be read by the FB MC_TouchProbe

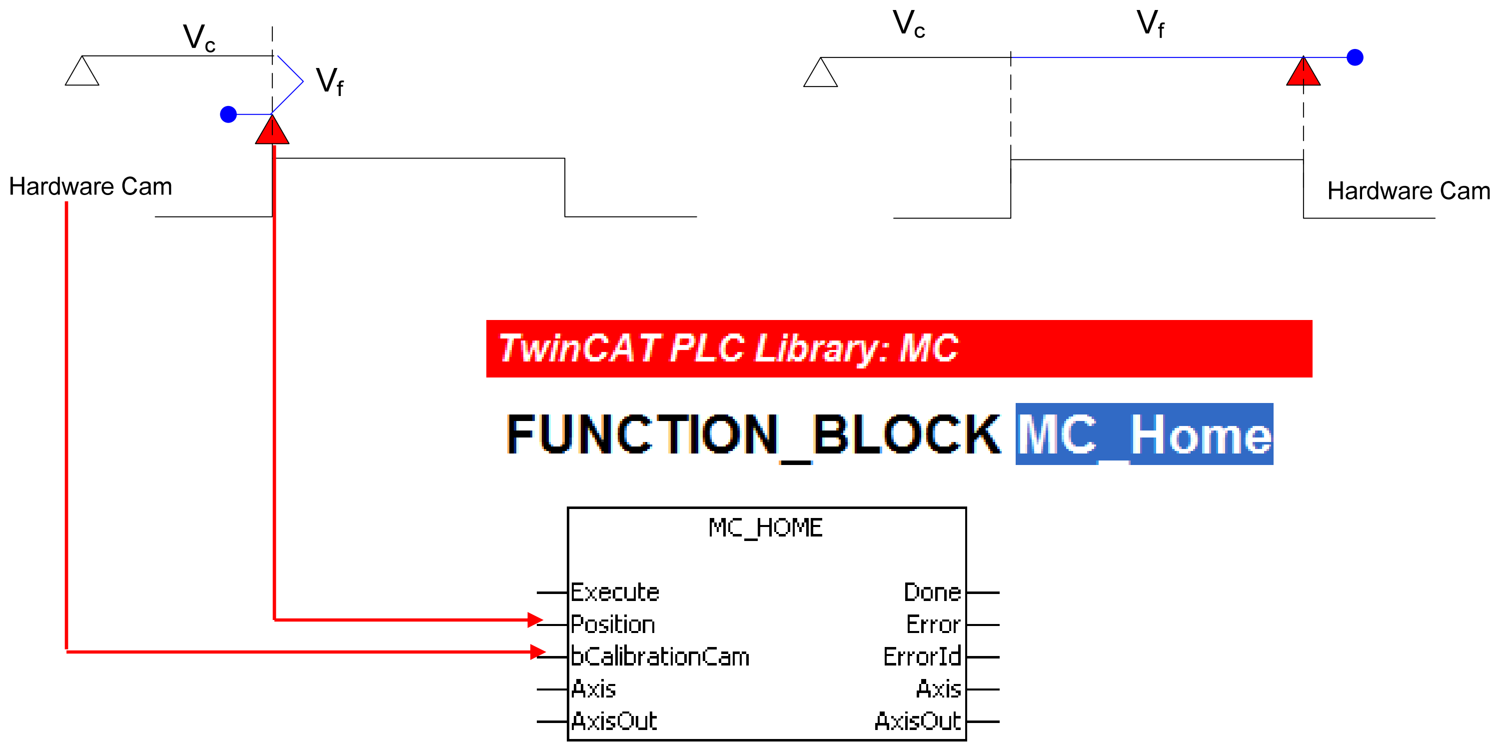
Homing Types



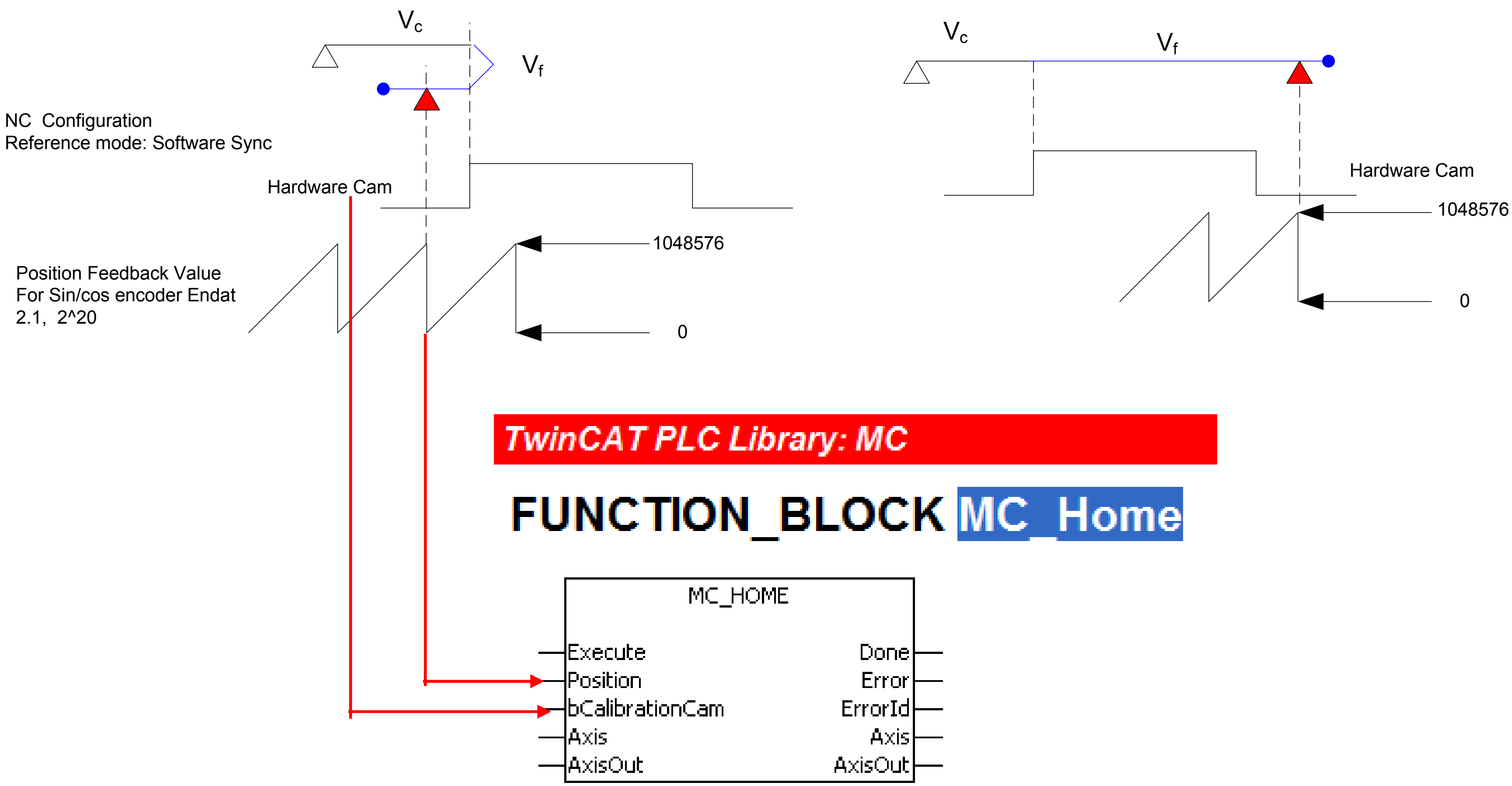
1. PLC Cam
2. Software Sync
3. Hardware Sync (Hardware Latch Pos, Hardware Latch Neg)

PLC Cam, “Homing”

NC Configuration
Reference mode: PLC Cam



Software Sync, "Homing"

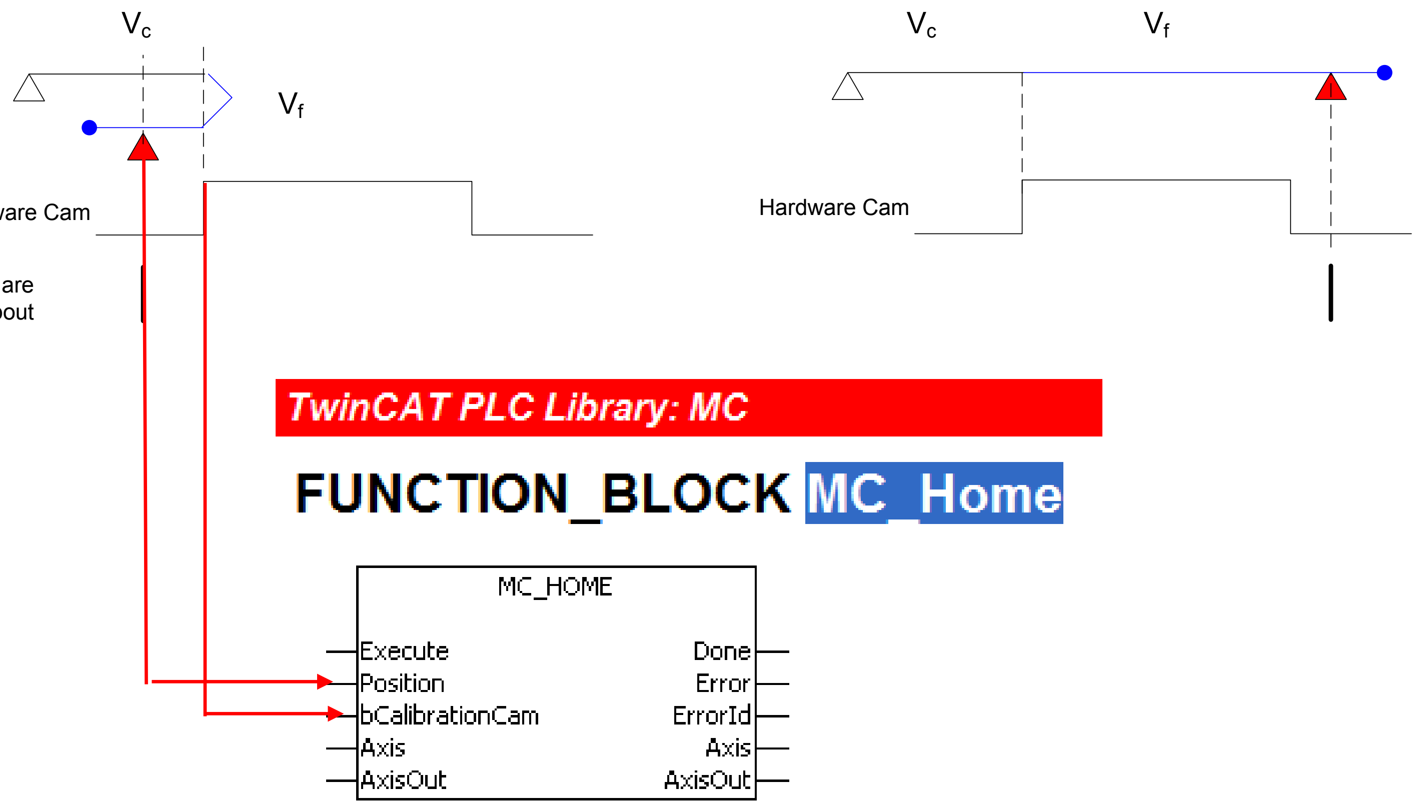


Probe Unit

Operation, “Homing”

NC Configuration
Reference mode: Hardware Sync

Ax5000 Probe valid signal. There are many combination possibilities about the Probe unit.



Operation, “Homing”

Reference Movement

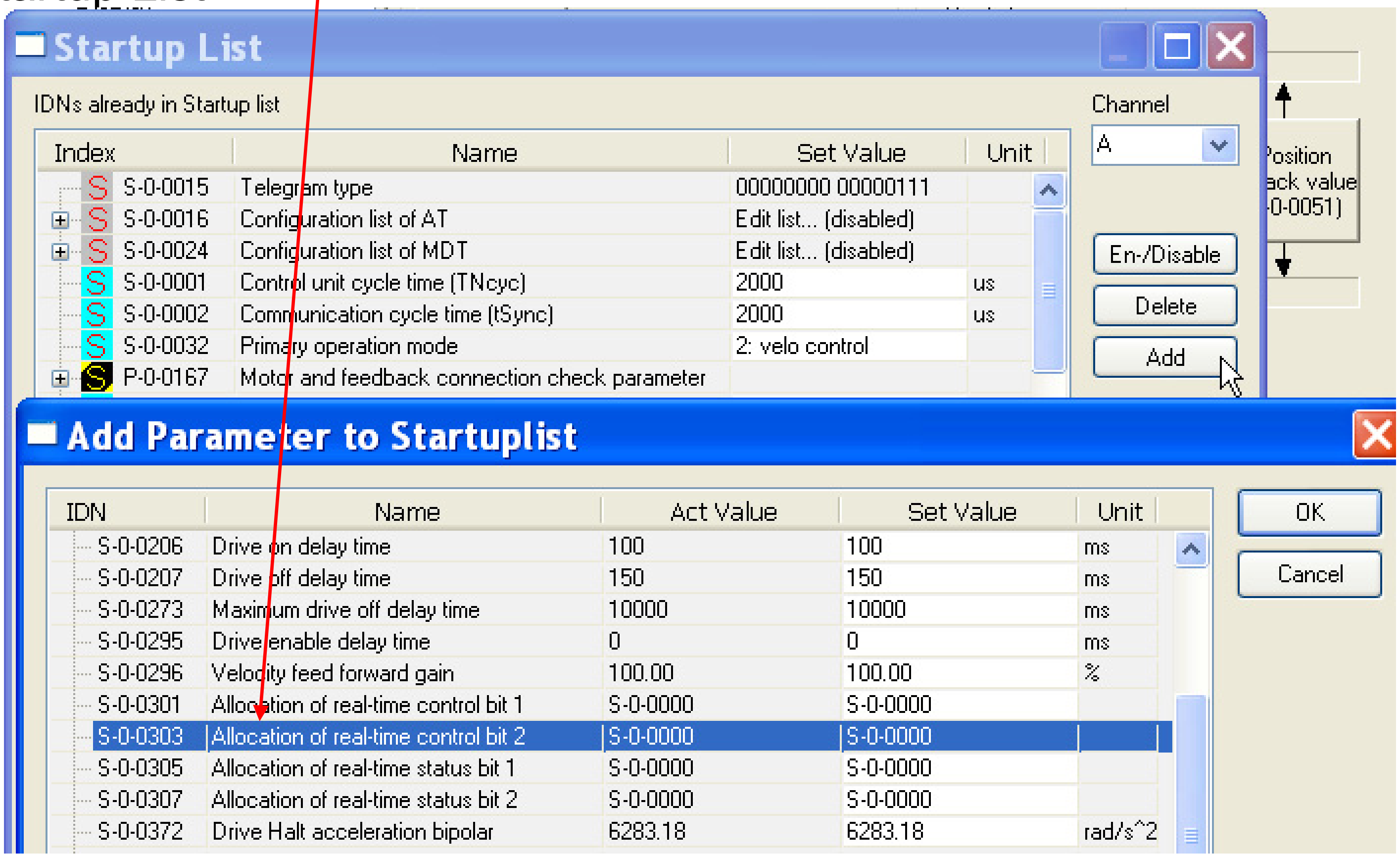
It is possible to do Homing by TwinCAT using the probe unit of AX5000. The probe unit configuration should be done in the “Startup List”, it is also possible during axis operation.

So that TwinCAT is able to control the probe unit by IDN S-0-0405 and S-0-0409, these has to be mapped into the real-time-control and status bit 2. This is done by the IDN S-0-0303 and S-0-0307 entry.

Configuration of real-time-control and status-bit: ¬

Operation, “Homing”

Add IDN S-0-0303 und IDN S-0-0307 to “Startup List”

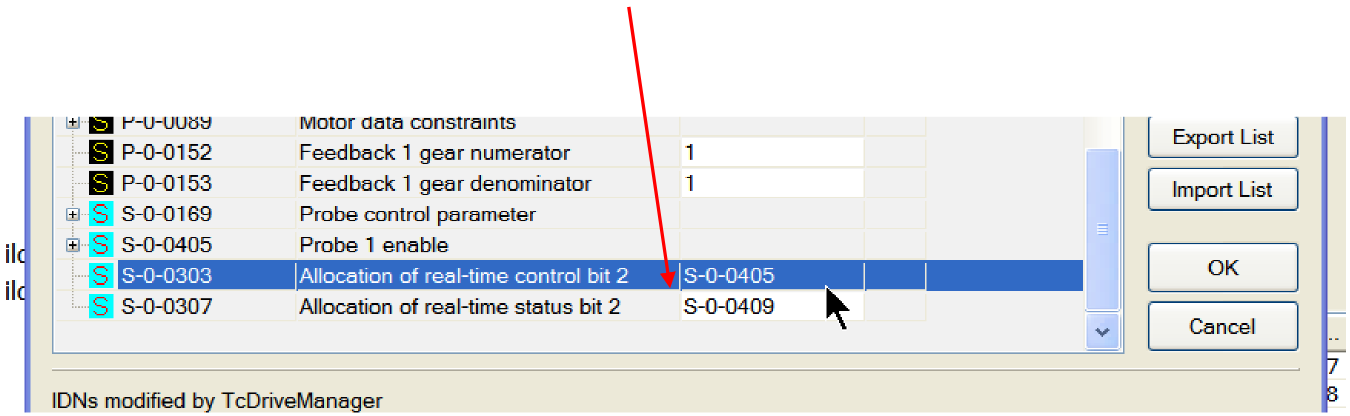


Probe Unit

Operation, “Homing”

The latched drive position is stored in IDN S-0-0130 “Probe value 1 positive edge“ or in IDN S-0-0131 „Probe value 1 negative edge. One of this selected value is cyclic (by the AT-Telegram) assigned to the NC.

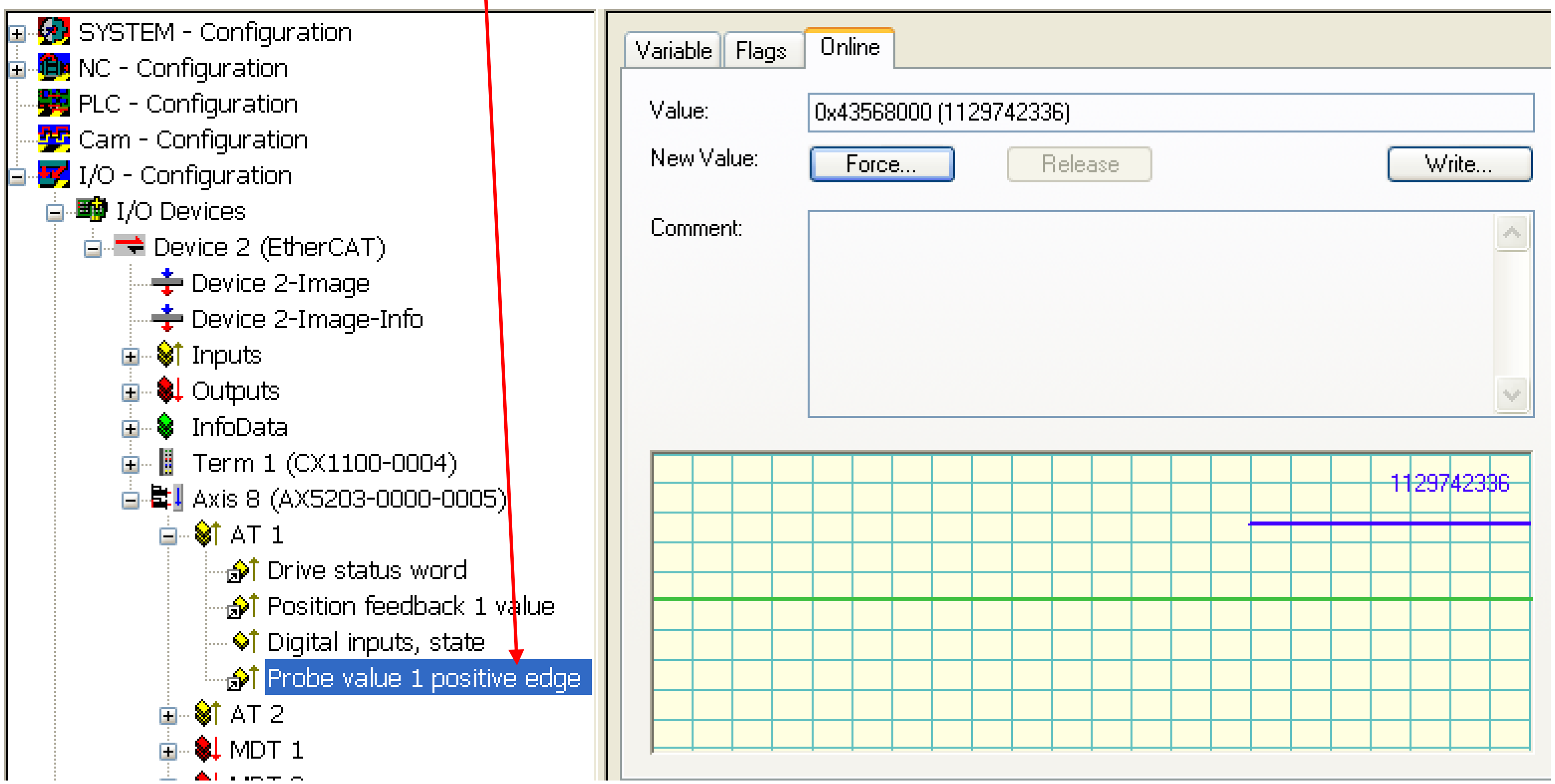
Configuration of S-0-0303 and S-0-0307 with:



Probe Unit

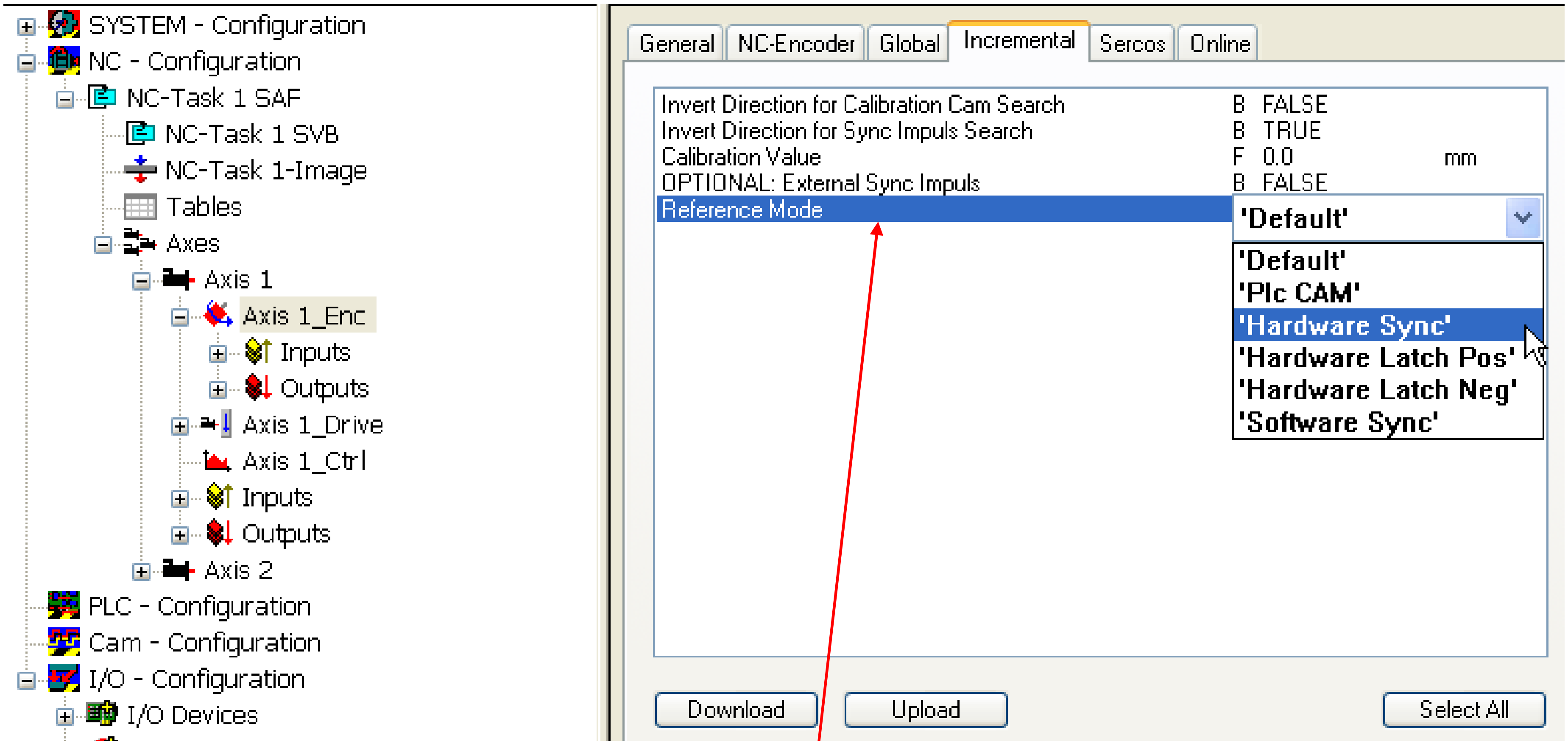
Operation, Position Latch

Add latched value to „ Process data's“



Probe Unit

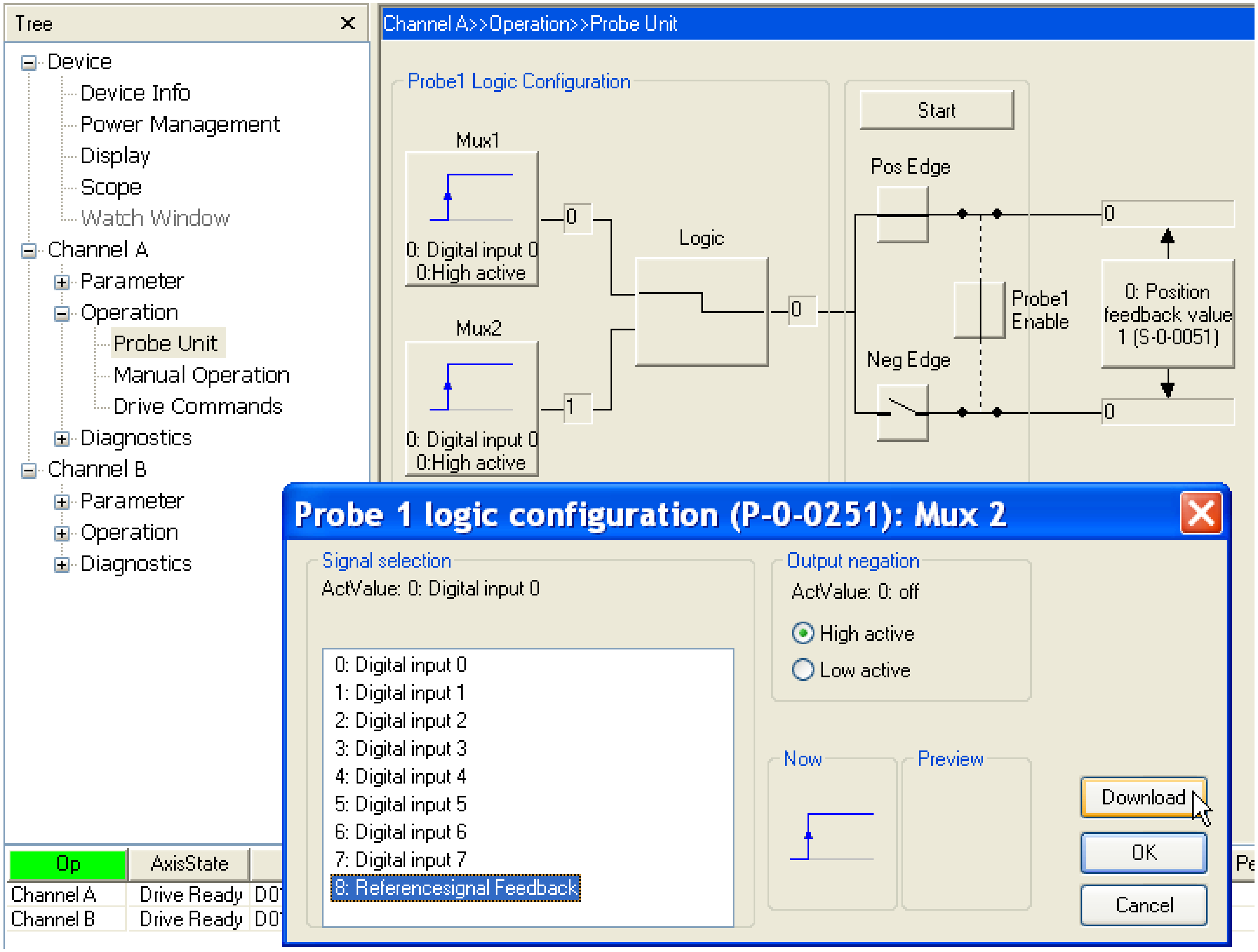
Operation, “Homing”



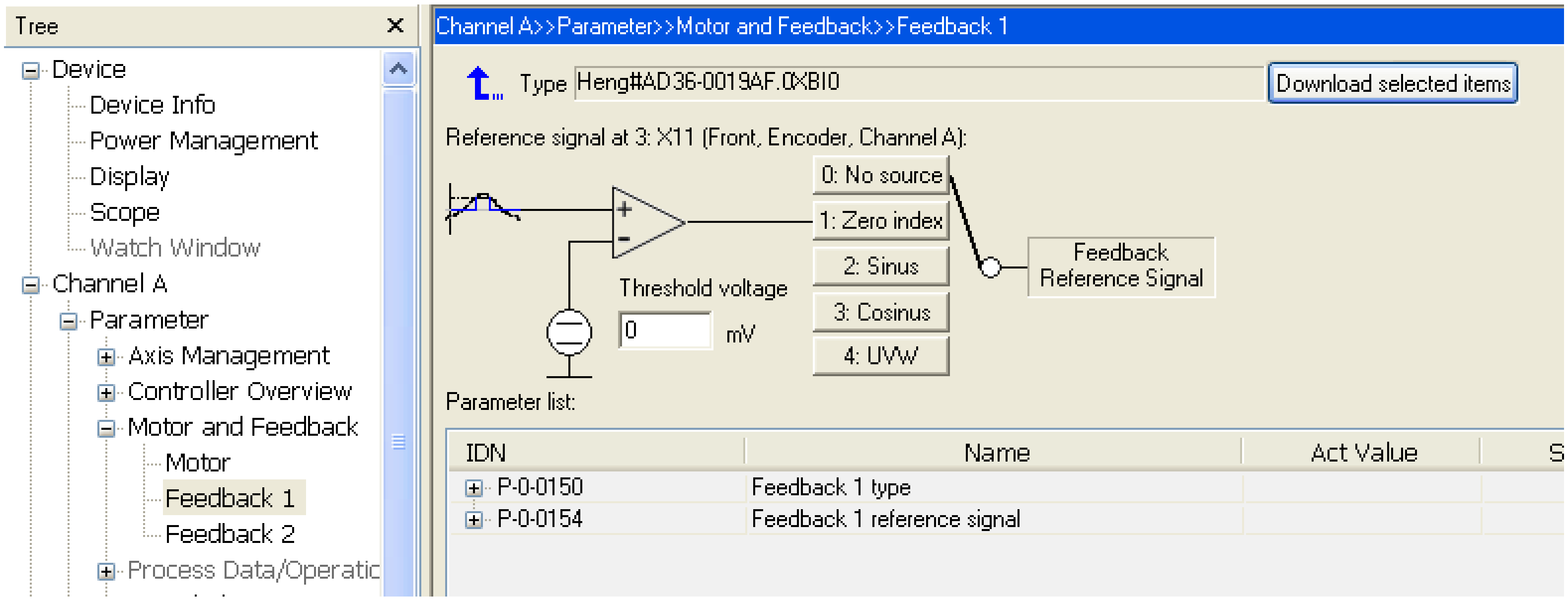
Selection of Reference Mode

Probe Unit

Operation, “Homing”



Operation, “Homing”

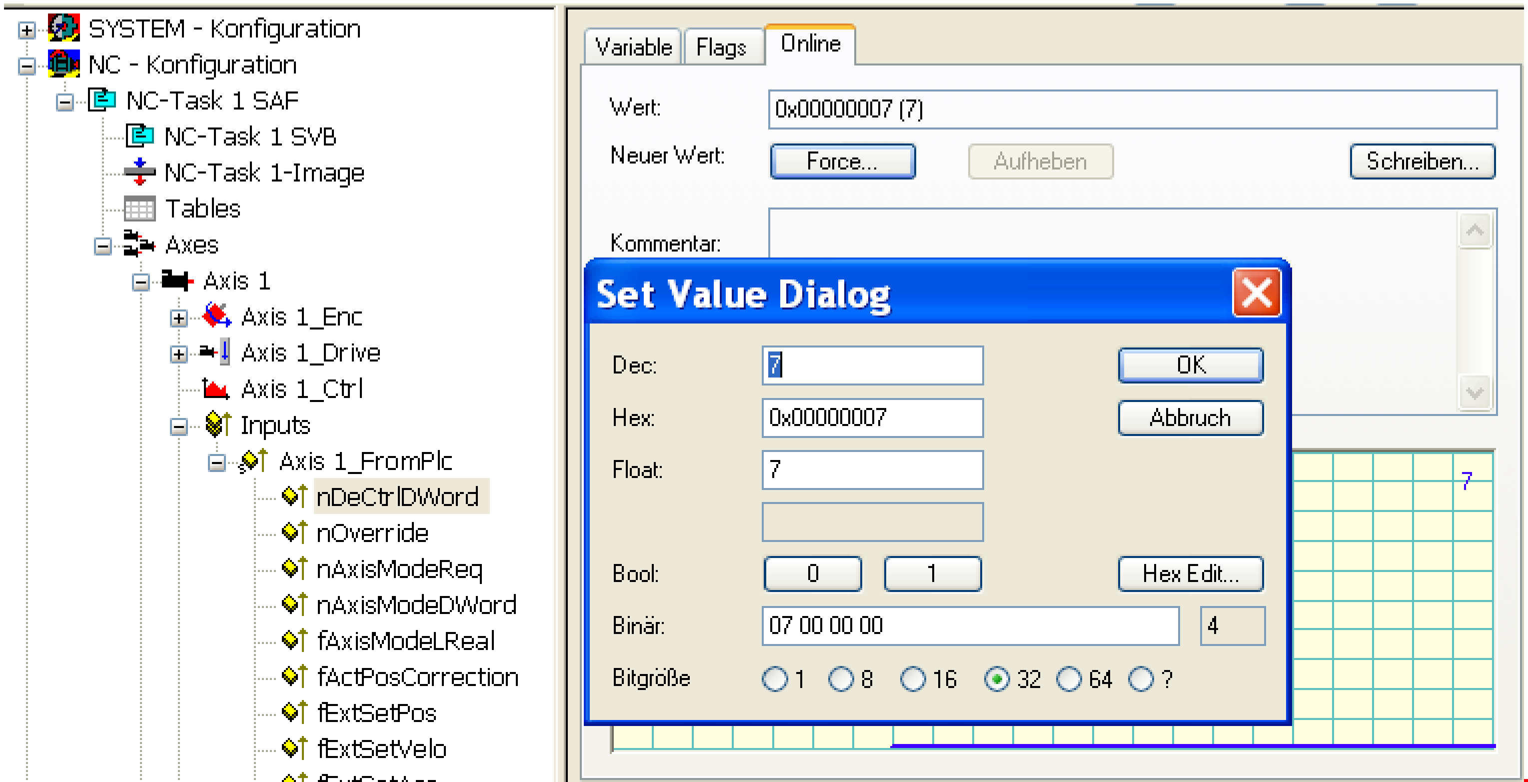


1: Feedback „Zero index“ detection.

2,3 Sin/Cos zero detection

4: Digital commutation for linear motors   i.prep.

Operation, “Homing”

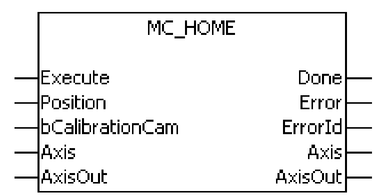


TwinCAT PLC Library: MC

FUNCTION_BLOCK MC_Home

Change direction by Bit 5; Input Hex 27.

Handled by „bCalibrationCam“ in MC_HOME.



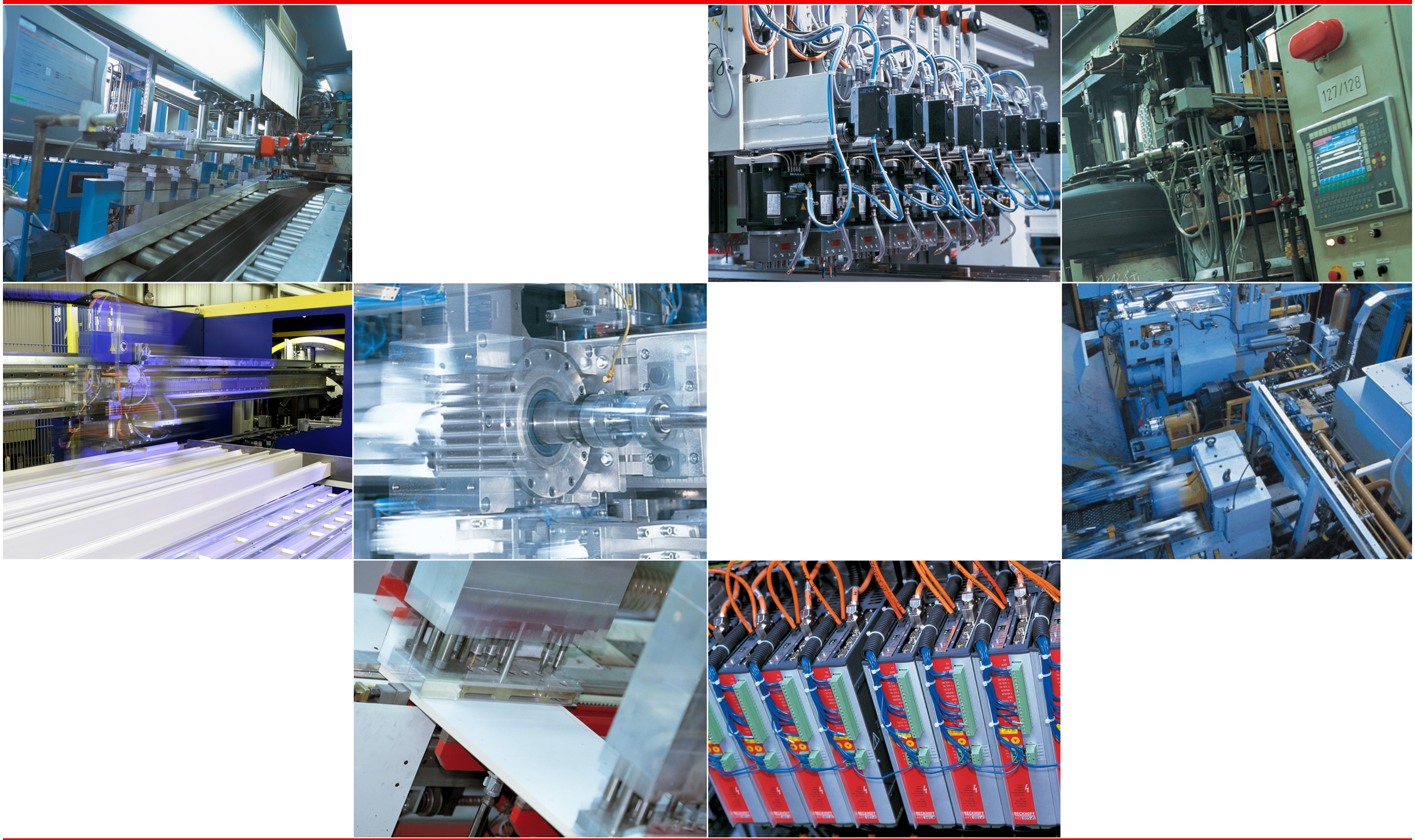
Programming example

Move Axis 1 and 2 by giving analog setpoint.

```
VAR_GLOBAL
	Ax_to_Plc AT%I*: ARRAY[1..2] OF NCTOPLC_AXLESTRUCT;
	Plc_To_Ax AT%Q*:ARRAY[1..2] OF PLCTONC_AXLESTRUCT;
	bEnable AT%I*:BOOL;		(* Input Digital1 Enable and Disable Axis 1+2 *)
	bMove AT%I*:BOOL;		(*  Input Digital2 start move *)
	bReset AT%I*:BOOL;		(* Input Digital 3, Reset; Stop Move; double click change axis *)
	bRefCam AT%I*:BOOL;		(* Reference Cam *)

	bReady AT%Q*:ARRAY[1..2] OF  BOOL;
	bError AT%Q*:BOOL;
	bAxIsCalibr AT%Q*:BOOL;
	rAbsPos AT %I*: INT;		(* Analog in 1 ; Position  *)
	rVelo AT %I*: INT;		(* Analog in 2 ; Veolcity  *)
	SetAnalogOut AT %Q*: WORD;	(* Set 10 V *)
END_VAR
```

Beckhoff Drive Technology

Thank you for your attention.



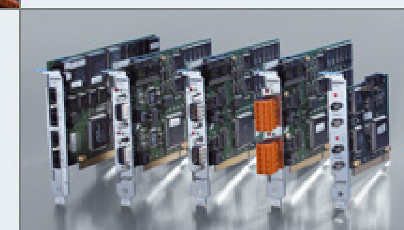
TwinCAT-Training: NC Point-to-Point

Beckhoff
Industrie-PC

Beckhoff
Lightbus

Beckhoff
TwinCAT

Beckhoff
Embedded-PC



Beckhoff
Busklemmen

Beckhoff
Feldbus Box

Beckhoff
PC-Feldbuskarten,
Switche

Beckhoff
EtherCAT

Beckhoff
Antriebstechnik

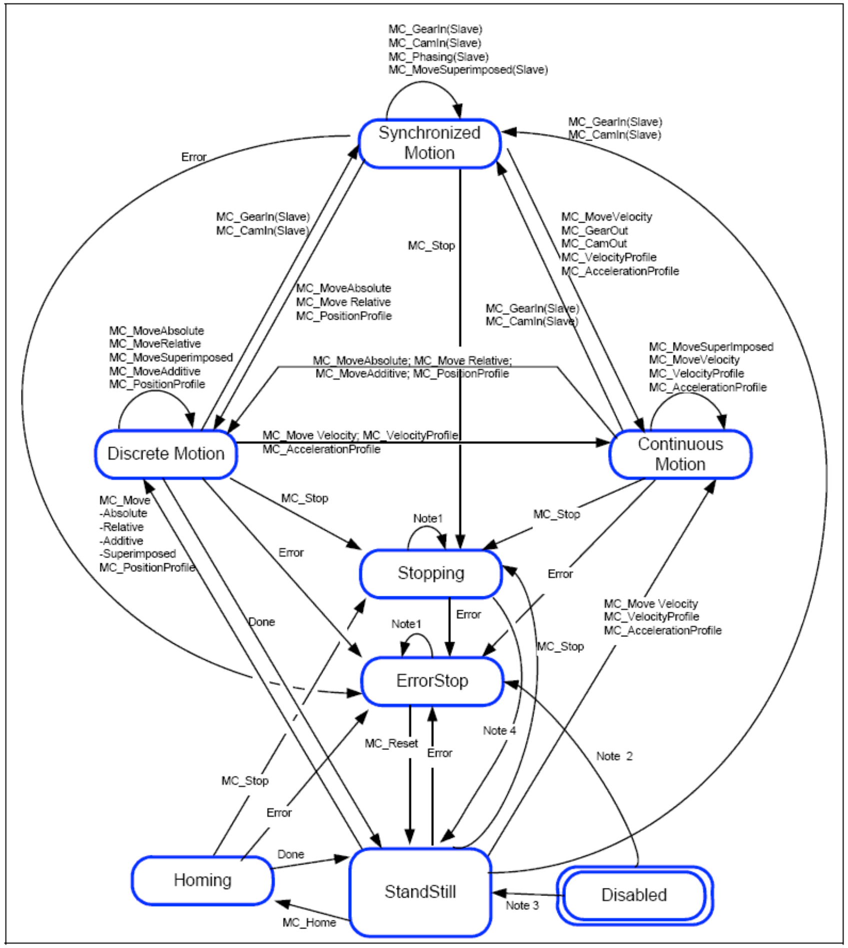
Target: IEC61131-3 compatible programmig interface for motion tasks

TwinCAT Motion

- NC PTP – TcMc2.lib
  - Why?
    - New FB‘s by PLCopen
    - Interfaces (IN/OUT/Handshakebits) are changed by the PLCopen
    - Easier handling of axis structures (Beckhoff)
    - That means:
    - TcMC2 is not downwards compatible (changed interface)
    - TcMc.lib is futhermore supported by Beckhoff
    - For new projects TCMc2 is recommended

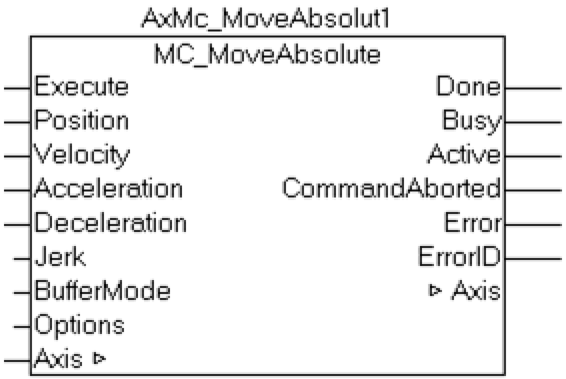
TwinCAT Motion

- **NC PTP – TcMc2.lib**
  - **PLCopen changes**
    - **New FB`s**
    - **New functions**
      - **Buffering and Blending**
      - **Extended inputs/outputs**



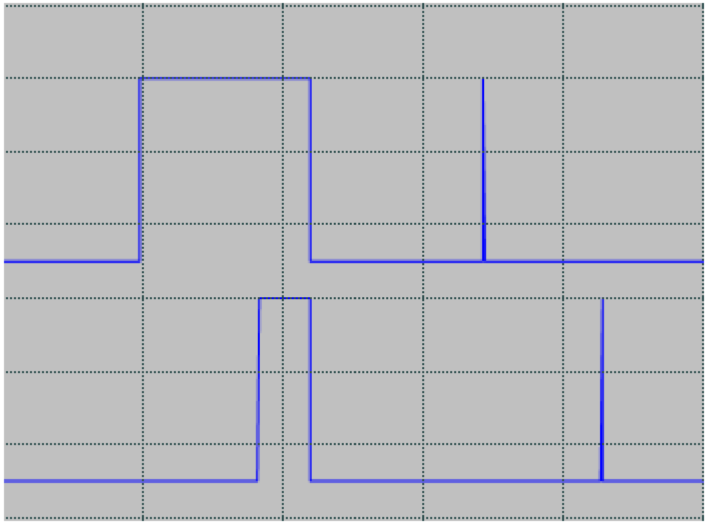
General Handshake

- Easy message by done



Execute

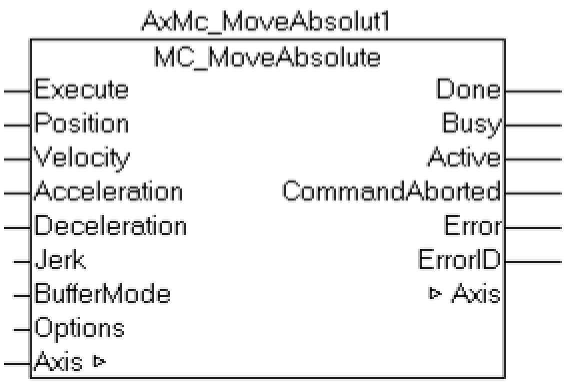
Done

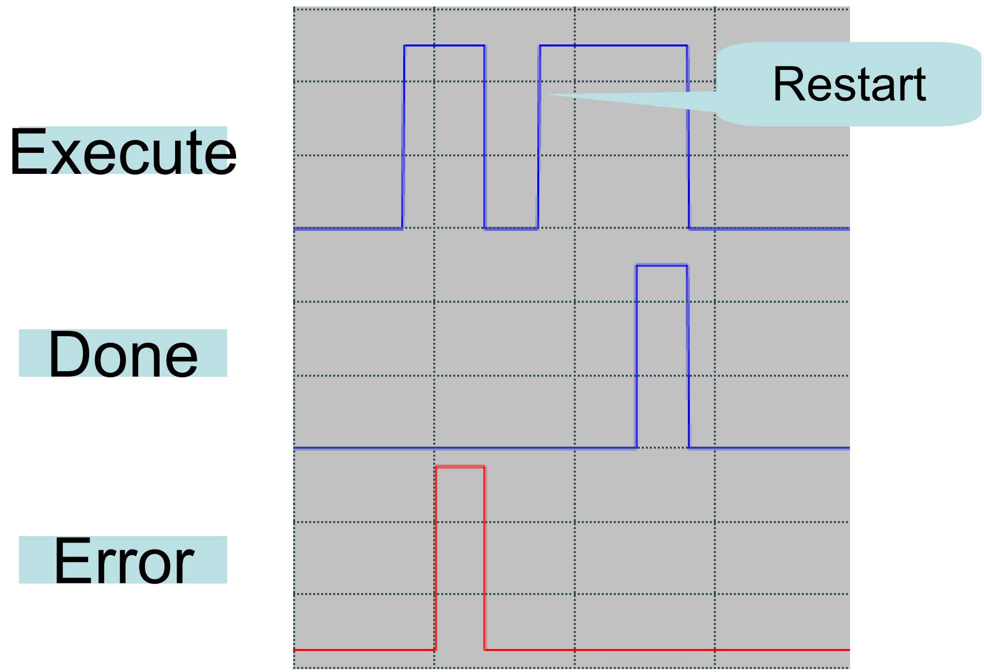


- If Execute is disabled before Done, Done is active for 1 cycle

General Handshake

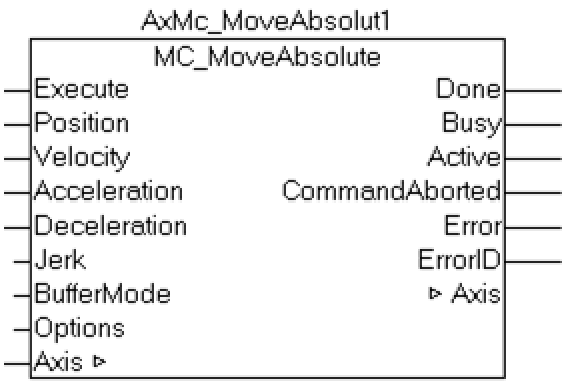
- Error

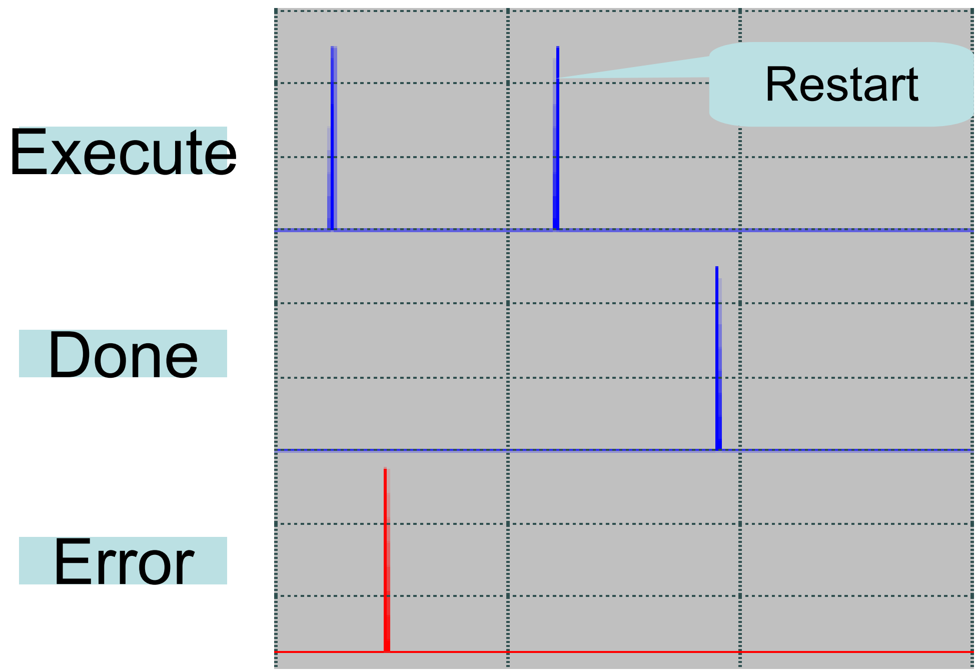




- Error is set until „End Execute“. Reset from error state of FB with new start trigger.

- Error

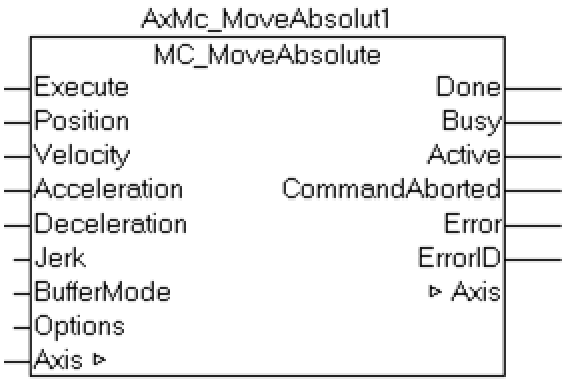


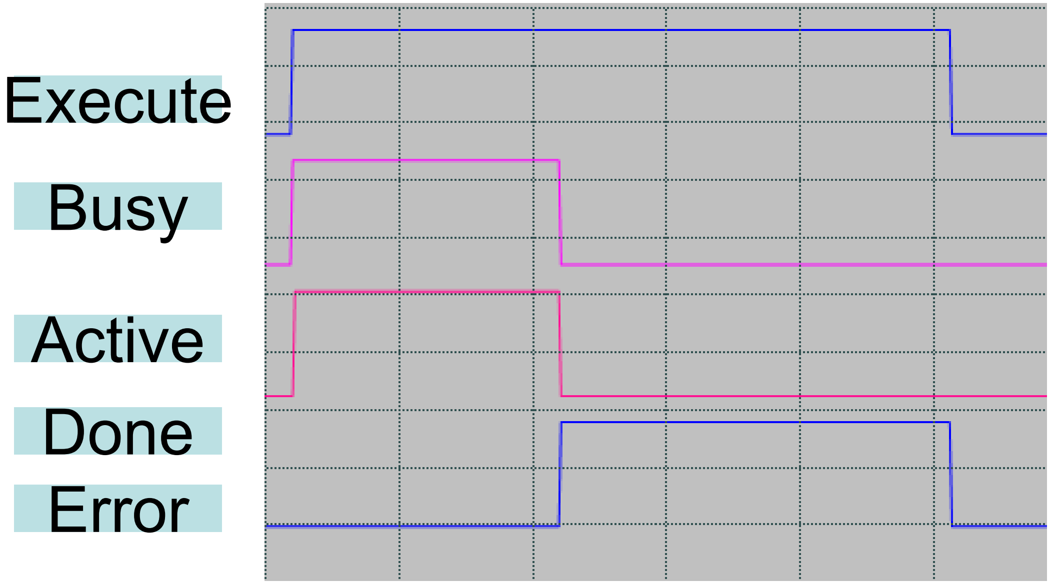


- Error becomes active if execute is disabled, error is announced for one cycle

Busy and active

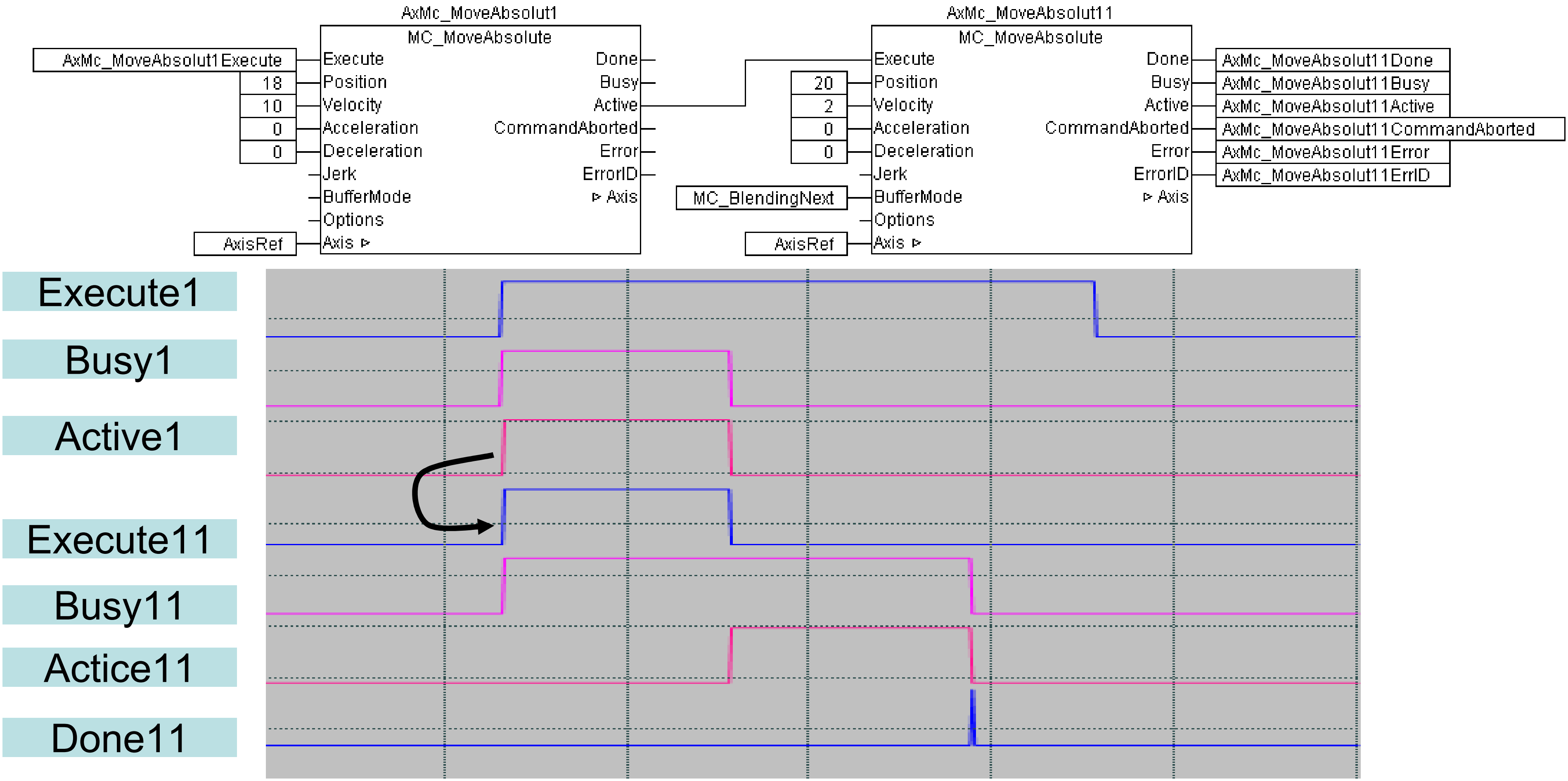
- Busy: FB is active, (is carrying instructions out)
- Active : FB is active AND axis has instructions
- Active and busy are not accurate simultaneous, active becomes active after busy





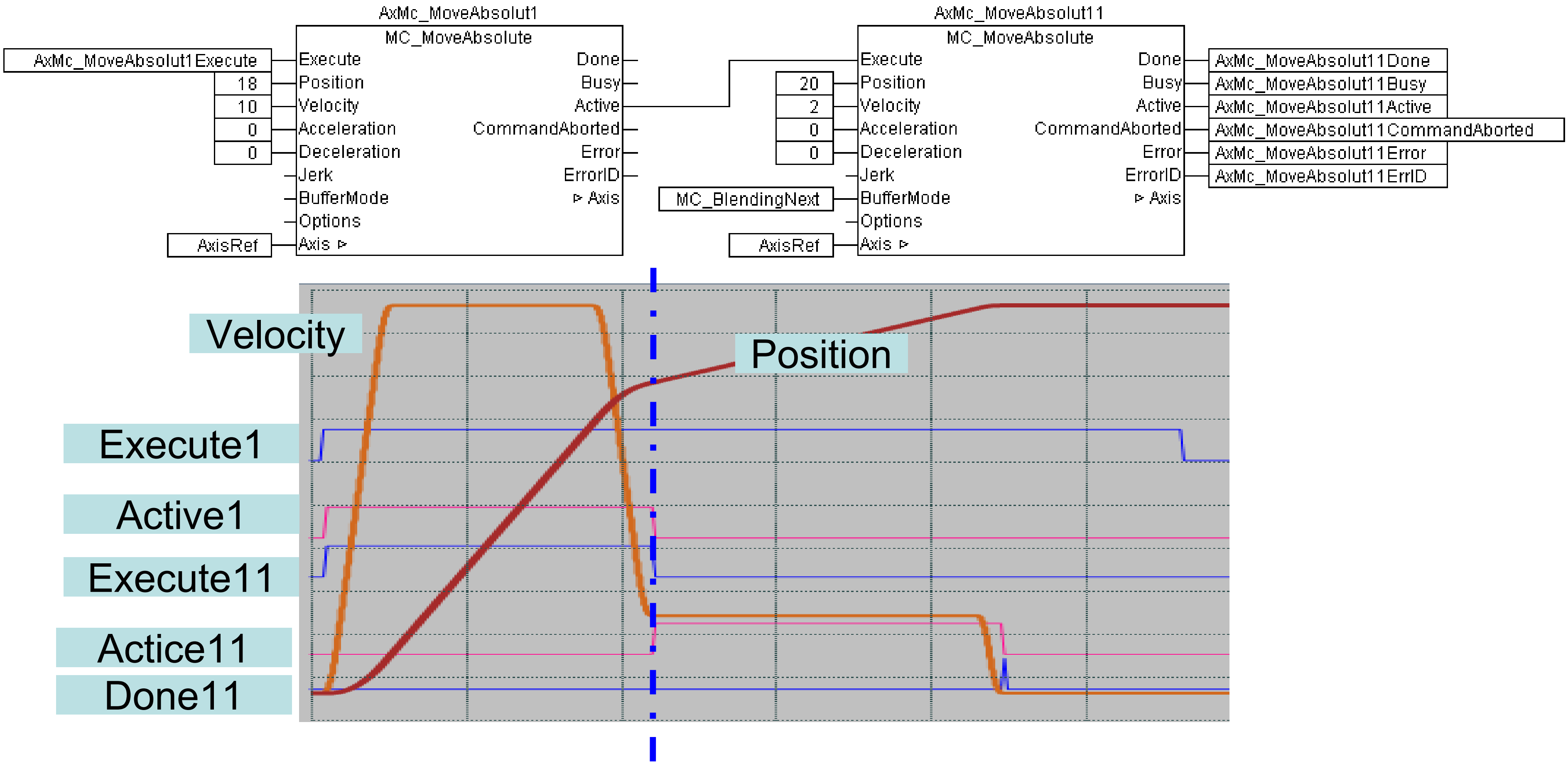
Busy and active in buffered mode

- Active can be used to set a follow-up instruction (buffered) for the same axis while the instruction is active

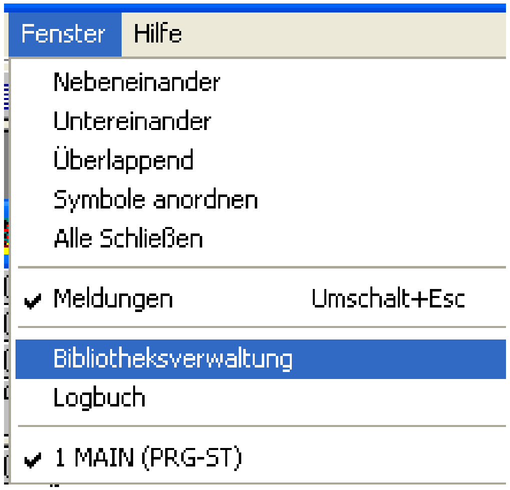


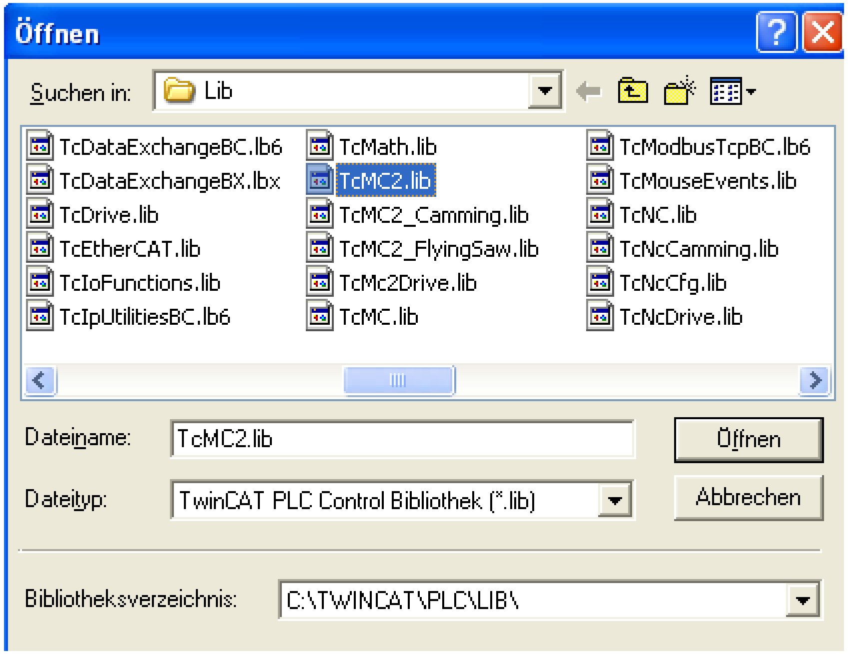
Busy and active in buffered mode

- Effect on the axis



- Linking of TcMc2.Lib





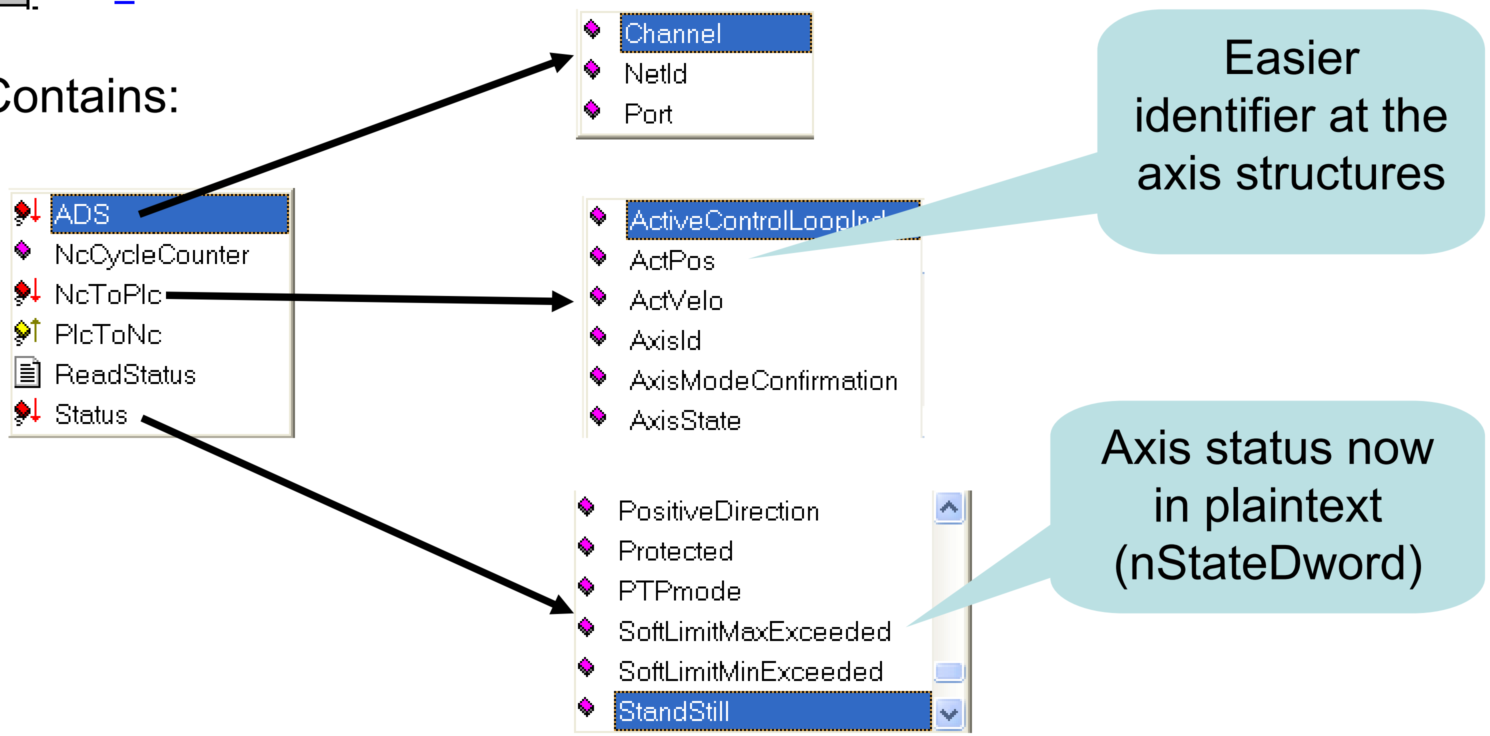
AxisRef

- NcToPlc and PlcToNc now in one object

Globale_Variablen

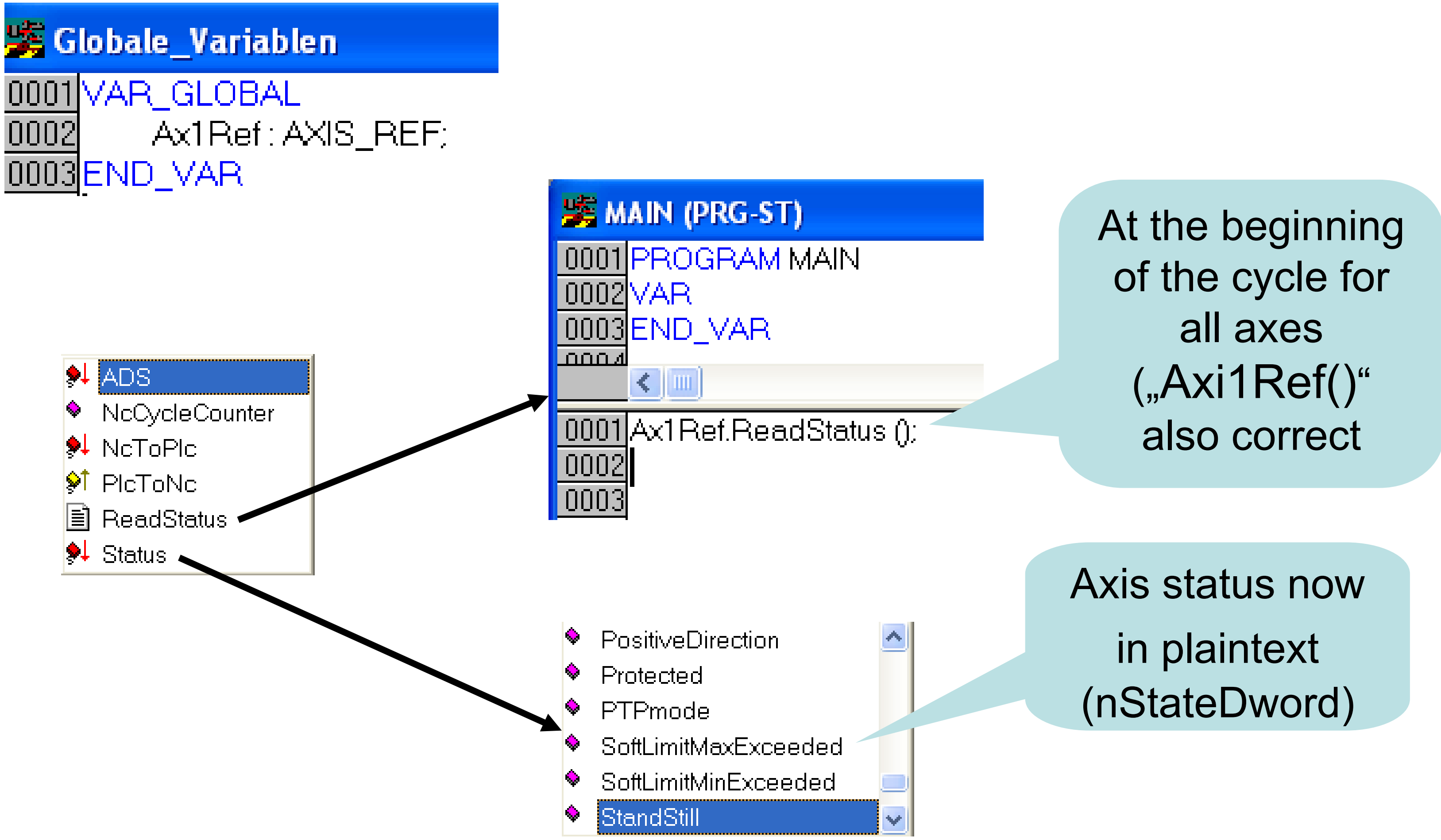
```
0001 VAR_GLOBAL
0002     Ax1Ref : AXIS_REF;
0003 END_VAR
```

- Contains:



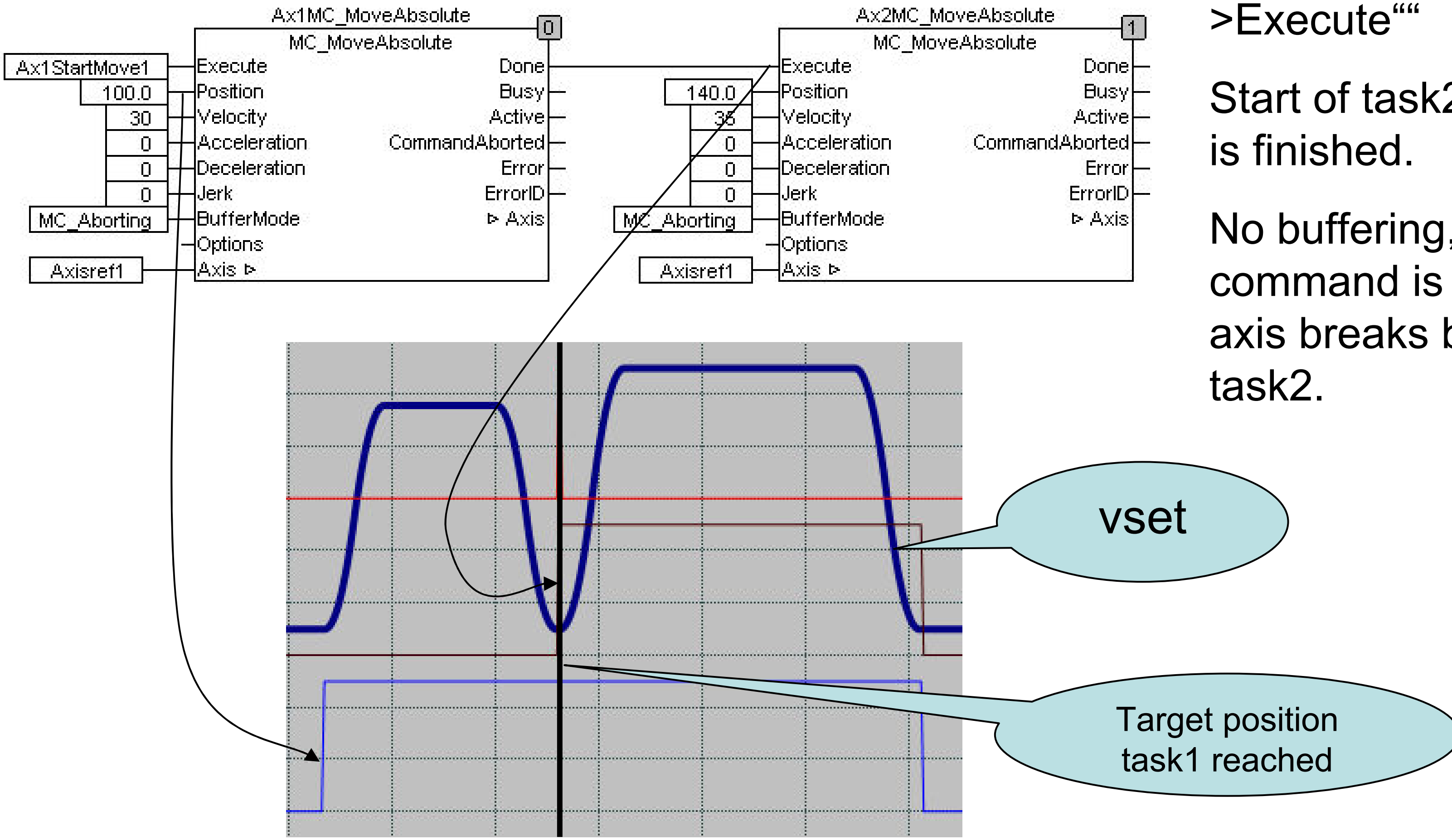
AxisRef

- Call the action ReadStatus for update the state



Examples about Buffered and Superimposed

TcMC II BufferMode

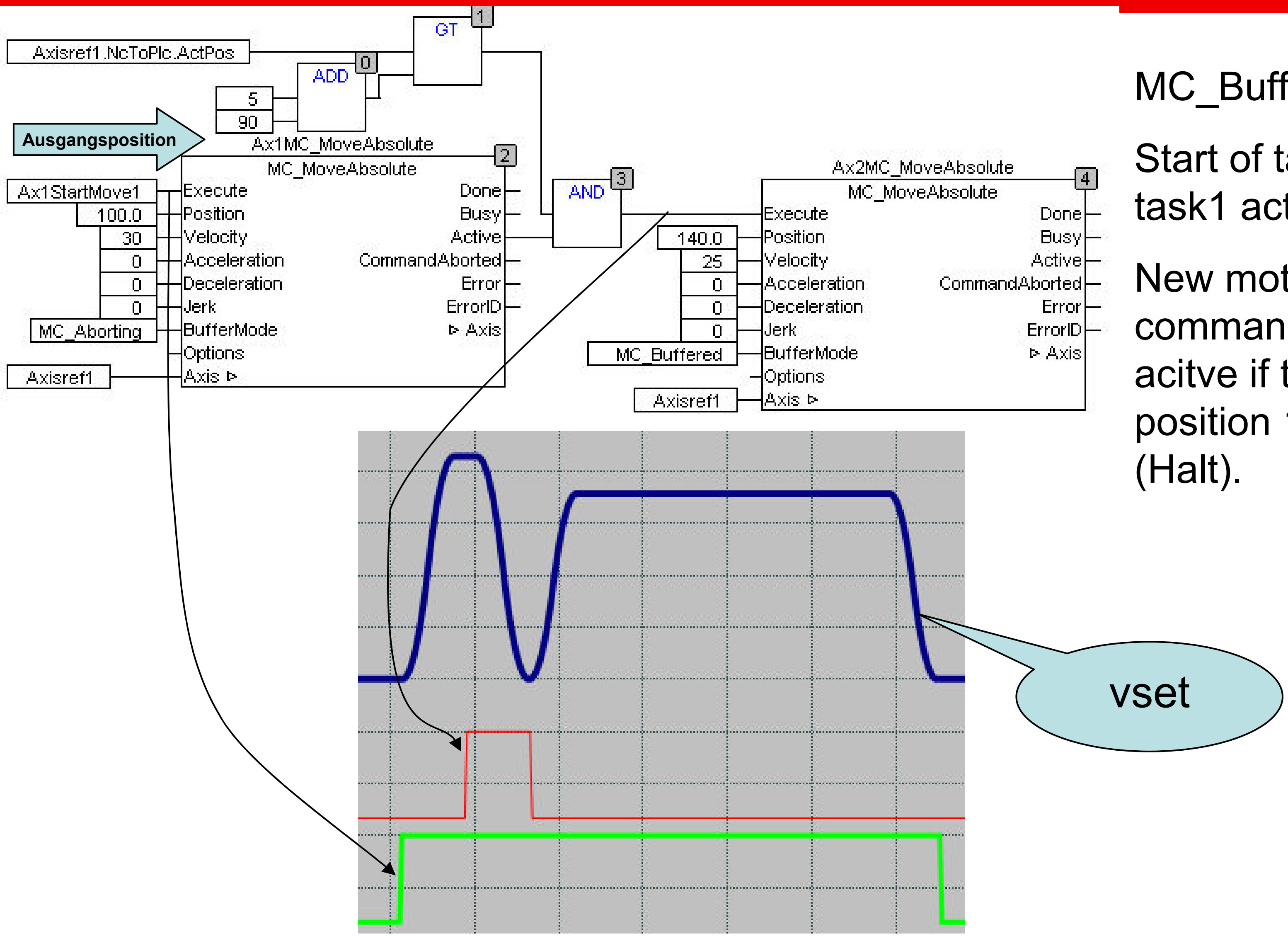


„Normal nesting Done->Execute““

Start of task2 if task1 is finished.

No buffering, motion command is finished, axis breaks before task2.

TcMC II BufferMode

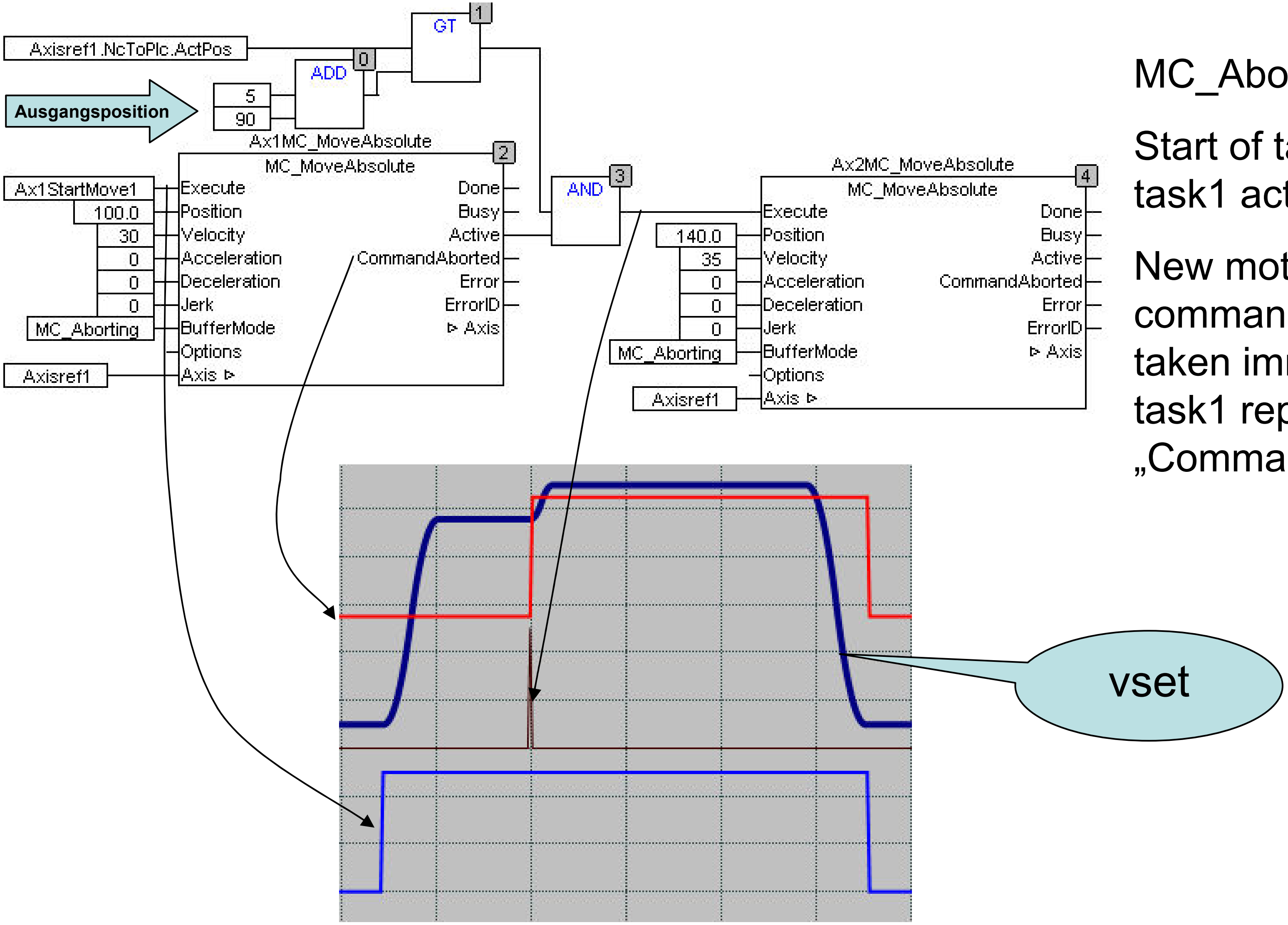


MC_Buffered

Start of task2 during task1 active.

New motion command becomes acitve if target position 1 reached (Halt).

TcMC II BufferMode

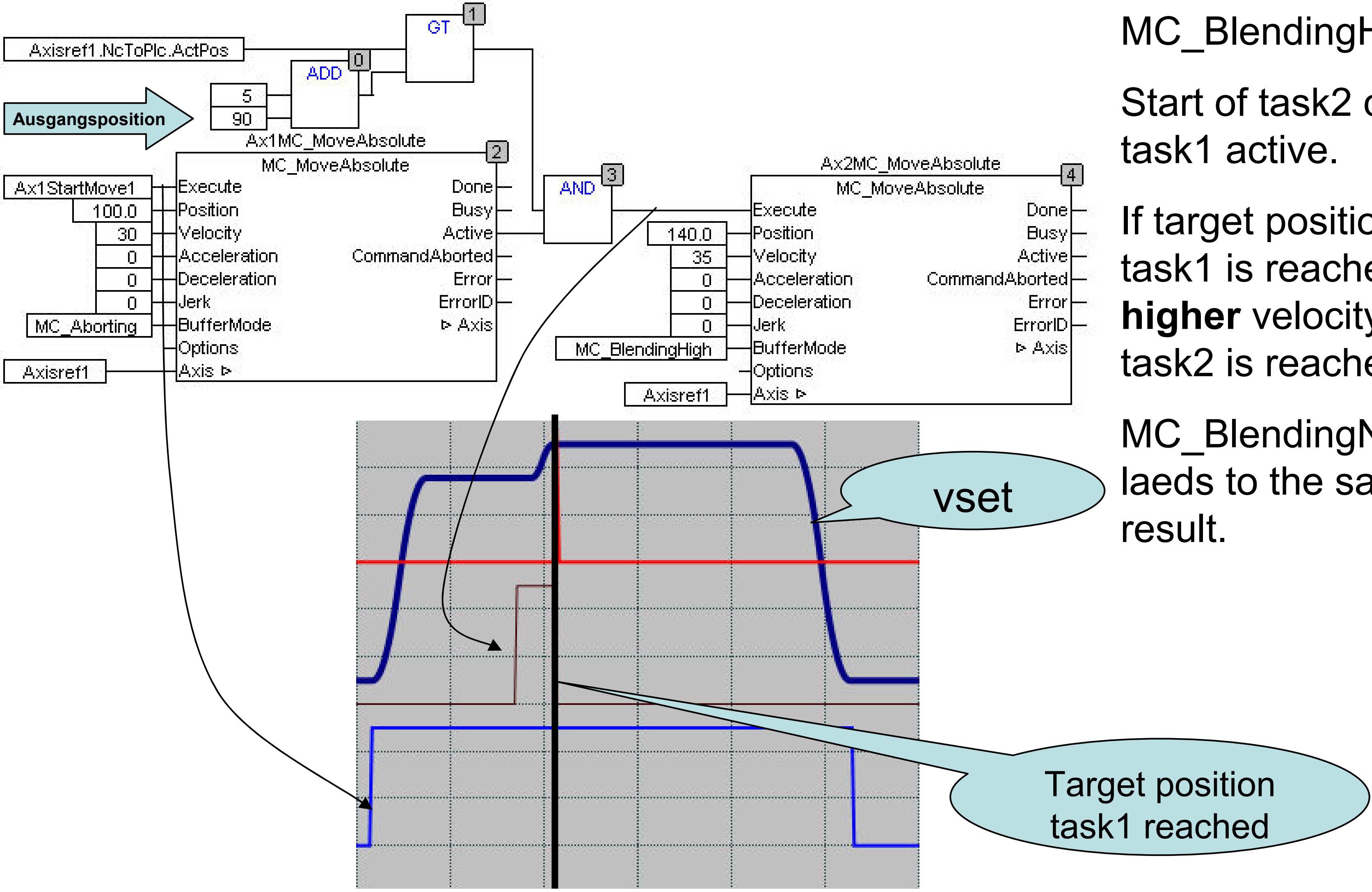


MC_Aborting

Start of task2 during task1 active.

New motion command will be taken immediately, task1 reports „CommandAborted“.

TcMC II BufferMode



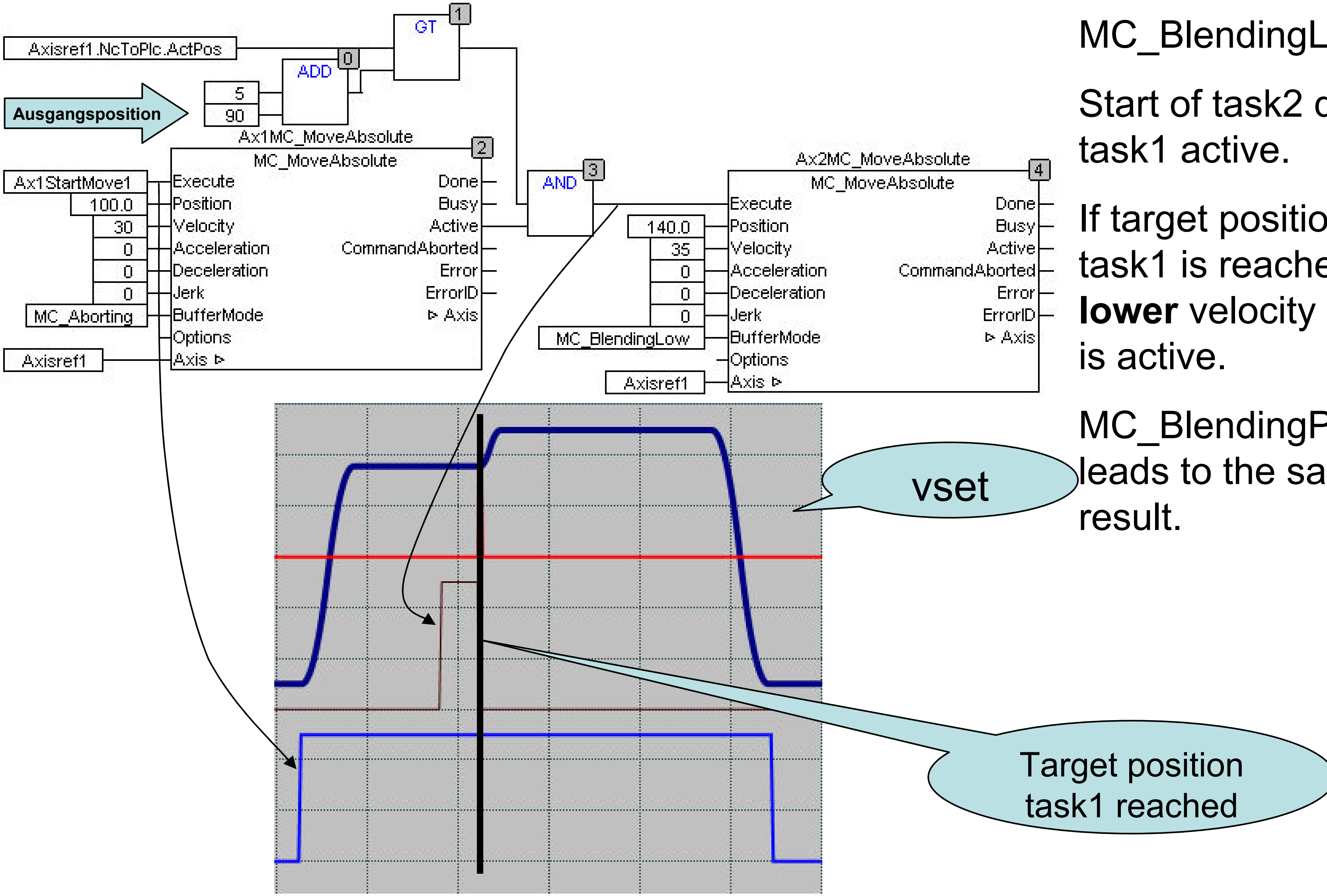
MC_BlendingHigh

Start of task2 during task1 active.

If target position of task1 is reached, the **higher** velocity of task2 is reached.

MC_BlendingNext laeds to the same result.

TcMC II BufferMode



MC_BlendingLow

Start of task2 during task1 active.

If target position of task1 is reached, the **lower** velocity of task1 is active.

MC_BlendingPrevious leads to the same result.

TcMC II BufferMode

Example :

Eine Achse soll auf die Endposition 130 fahren.

Auf dem Verfahrweg „kreuzt“ eine andere Bewegung. Spätestens bei der Achsposition 55 muss das Freigabesignal vorhanden sein.

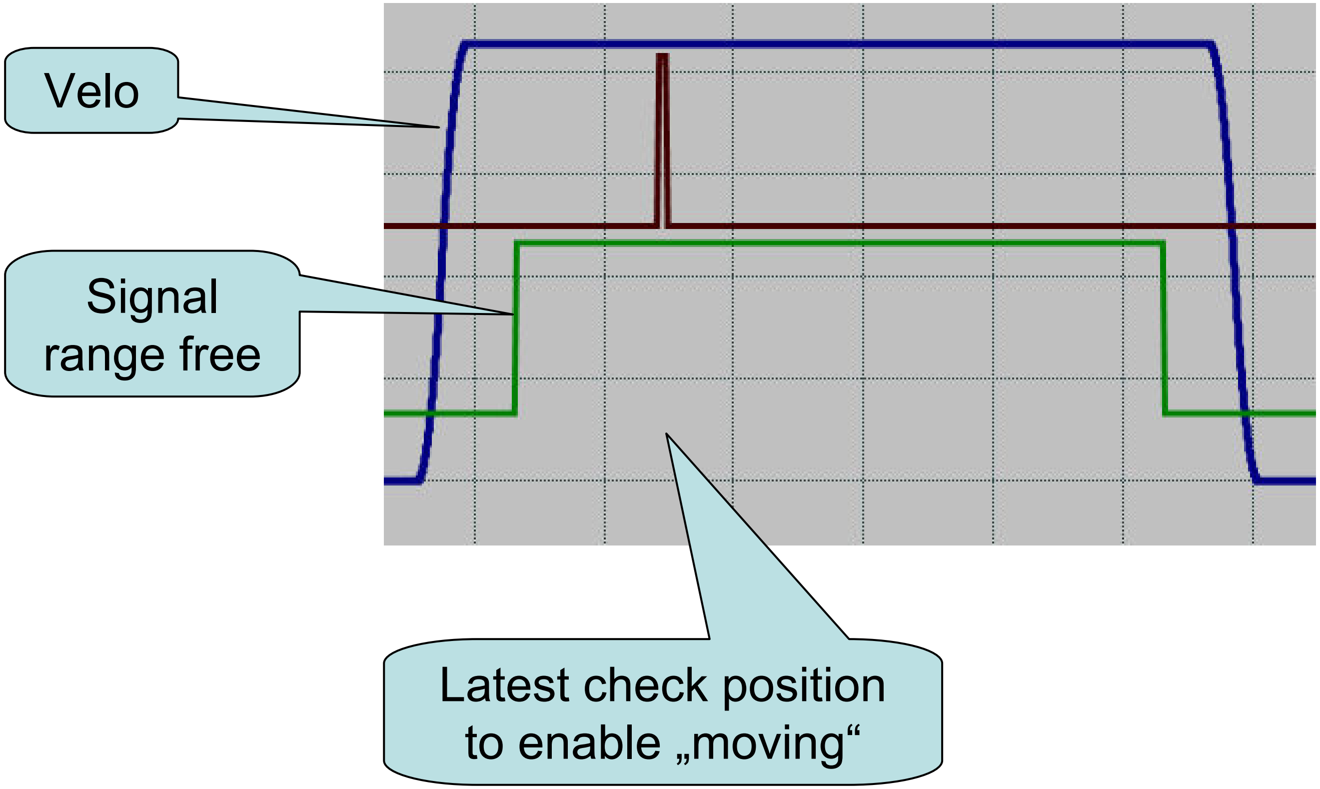
-> ist der Bereich frei soll die Achse die Endposition ohne Stopp anfahren

-> ist der Bereich noch belegt soll die Achse definiert auf Position 75 anhalten und erst starten wenn eine Freimeldung kommt.

Solution: MC_Moveabsolute with Blending

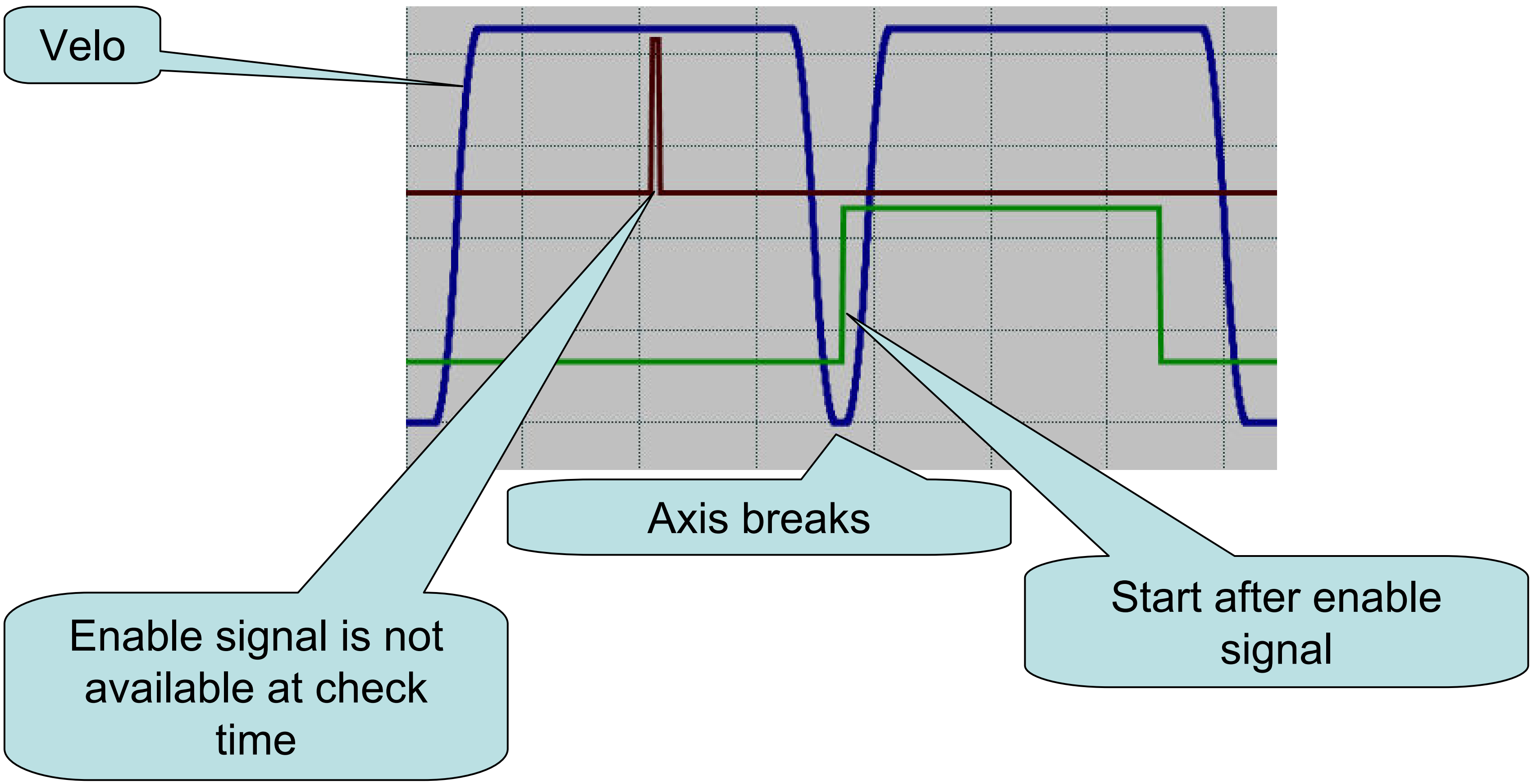
TcMC II BufferMode

Behaviour1: enable signal „early enough“



TcMC II BufferMode

Behaviour1 : enable signal „too late “



TcMC II BufferMode

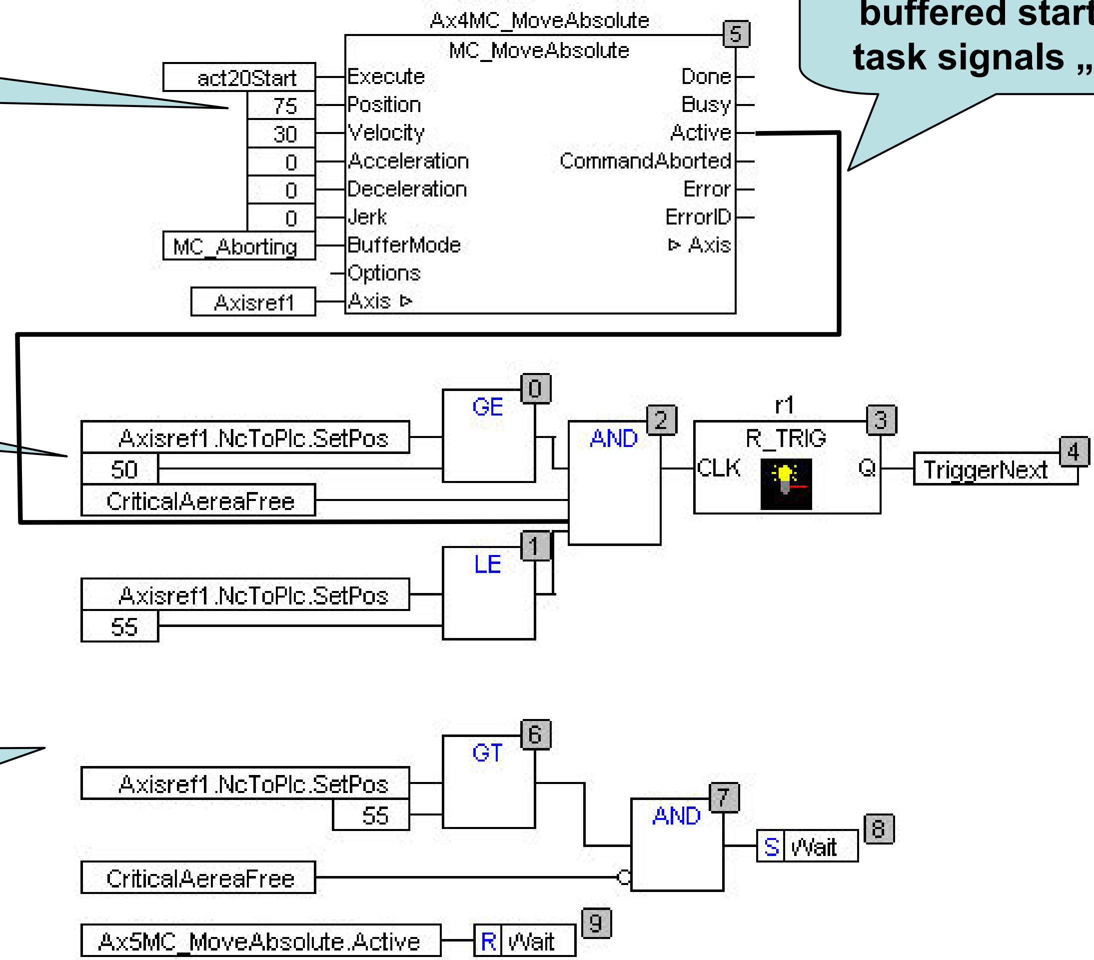
Flow

Start at first on inclined position

Axis can „moving“

Enable signal too late, axis needs to break

Earliest time to buffered start if first task signals „active“



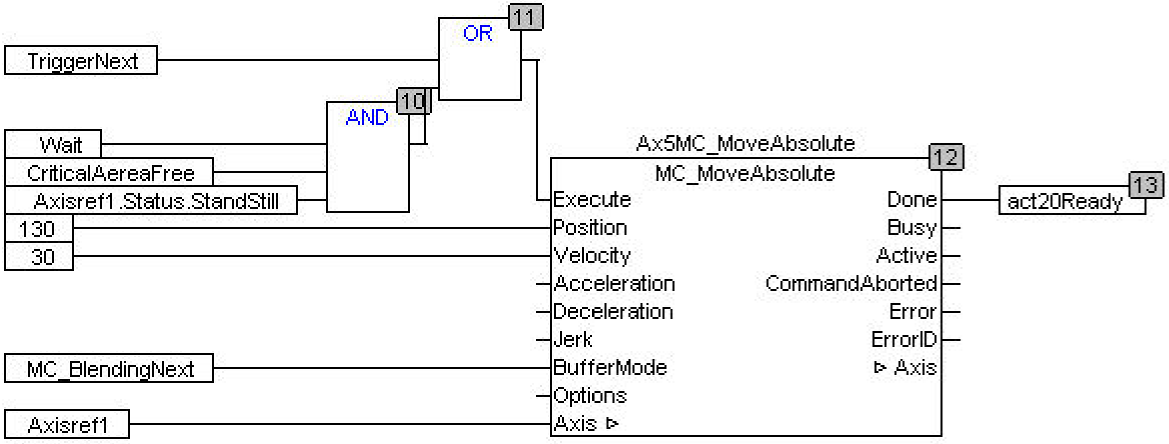
TcMC II BufferMode

Restart

Along without halt

Start after halt and enable signal

Hint: Don‘t forget action call „ReadStatus“. Invisible here.



TcMC II :MC_STOP vs. MC_HALT

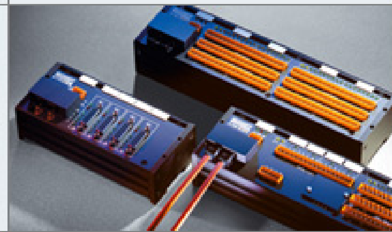
MC_STOP ,MC_HALT stops an axis with a defined braking ramp.

In contrast to MC_STOP, the axis is not locked against further motion commands. The axis can therefore be restarted through a further command during the braking ramp or after it has come to a halt.

TwinCAT-Training: NC Point-to-Point
VORLÄUFIGE DOKU

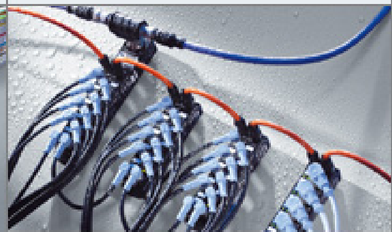
Beckhoff
Industrie-PC

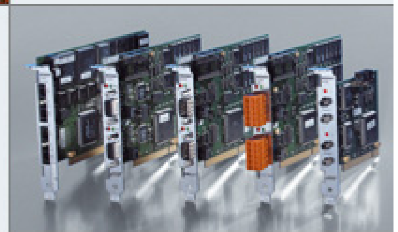
Beckhoff
Lightbus



Beckhoff
TwinCAT

Beckhoff
Embedded-PC





Beckhoff
Busklemmen

Beckhoff
Feldbus Box

Beckhoff
PC-Feldbuskarten,
Switche

Beckhoff
EtherCAT

Beckhoff
Antriebstechnik

TwinCAT NC PTP - Contents

- Part I General
- Overview
- Axis types
- Functional principle
- Referencing
- Motion Control Function blocks

- Teil II Practical Part:
- Setting up NC axes in the System Manager
- Starting NC axes from the PLC

TwinCAT NC PTP

Programming:	**Performed using function blocks for TwinCAT PLC according to IEC61131-3, convenient axis commissioning menus**
Debugging:	**Online monitoring of all axis state variables such as actual/set value, enable, controller values, online axis tuning, forcing axis variables**
Runtime system:	**NC Point-to-Point (NC PTP) including TwinCAT PLC**
Number of axes:	**up to 255**
Axis types:	**Electrical and hydraulic servo drives, frequency converter drives, stepper motor drives, switched drives (fast/crawl axes)**
Cycle time:	**Min. 50 µs, typ. 1 ms (freely adjustable)**
Axis functions:	**Standard axis functions: start/stop/reset/reference, Velocity override, target override**
Special functions:	**master-slave cascading, electronic gearboxes, online distance compensation of segments**

Camshafts, Flying saw

Camshafts :	Software solution for electronic camshafts, obviating the need to use mechanical camshafts and special hardware assemblies. A table relates the position of the master axis (mainshaft) to the associated position to which the slave axis is driven.
Flying Saw:	The "flying saw" (diagonal slave) is a special kind of slave coupling. The slave axis is brought from standstill to a speed synchronous with the master.
Universal Flying Saw (Ufs)	The "Universal Flying Saw" is able to start synchronisation of the slave even when the slave has already started, and is therefore no longer stationary.

FIFO, external set value generator

FIFO :	**Instead of using internal generation of standard set values, an axis can also obey an externally calculated sequence of set values that can be supplemented as the movement of the axis proceeds (FIFO buffer).**
External Set value generator	**Enables the implementation of individual set value generators. Superposition of existing internal generators and external set value sources is possible**

TwinCAT NC Interpolation (NC I)

TwinCAT NC Interpolation (NC I) is the NC system for linear or circular interpolated path movements of axis groups each involving two or three drives.

TwinCAT NC I offers

- **2D and 3D interpolation (interpreter, set point generation, position controller),**
- **an integrated PLC with an NC-I interface and**
- **an I/O connection for axes via the field bus.**

TwinCAT NC I

- **Programming:** DIN 66025 programs for NC interpolation, access via function blocks for TwinCAT PLC according to IEC61131-3
- **Debugging:** Online-Monitoring in the TwinCAT System Manager with the following displays: present set/actual positions, following errors of all axes, NC program line presently being executed/interpreted, channel status
- **Runtime system:** NC PTP + NC interpolation, including TwinCAT PLC
- **Number of axes:** Per channel: 3 axes interpolated, 5 auxiliary axes max. 31 channel
- **Axis types:** Servo axes
- **Interpreter-funktionen:** Subroutines and jumps, programmed loops, zero shifts, tool compensations, M and H functions
- **Geometrien:** Straight lines and circular paths in 3D space, circular paths in all main planes, helixes with base circles in all main planes
- **Axis functions:** Online reconfiguration of axes in groups, path override, slave coupling to path axes

Axis types continuous

Continuous axes

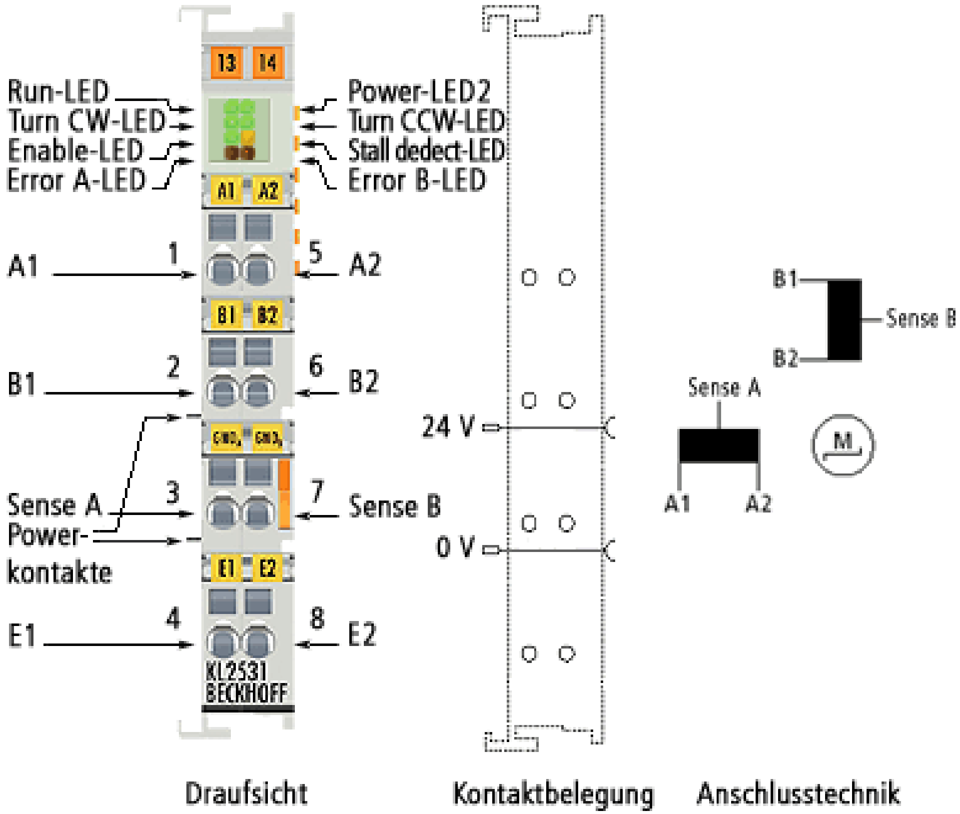
The axis responds to a continuously changeable set value

The set value is generated by TwinCAT NC,

- **e.g. servo with +/- 10 V,**
- **Sercos drive,**
- **frequency converter,**
- **linearised hydraulic axis,**
- **Stepper motor at stepper motor terminals,**
- **stepper motor drive with amplifier**

Axis types

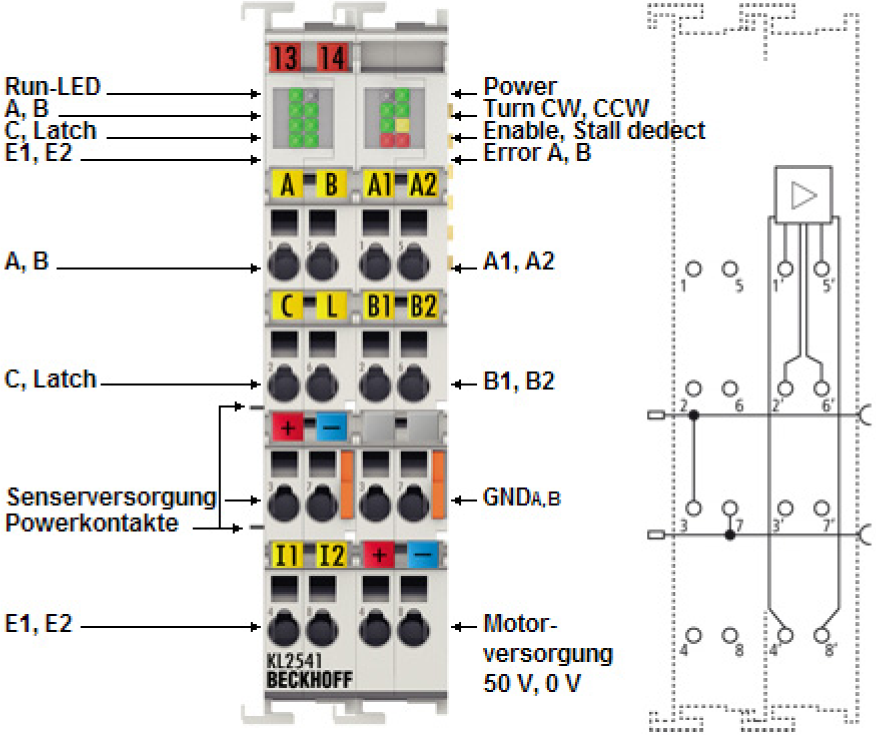
KL2531



- **Output current** 2 x 1 A, 2 x 1,5 A peak current, overload and short circuit protected
- **Maximum step frequency** > 125.000 steps/s
- **Step pattern** full step, half step, up to 64-fold microstepping
- **Current controller frequency** approx. 25 kHz

Axis types

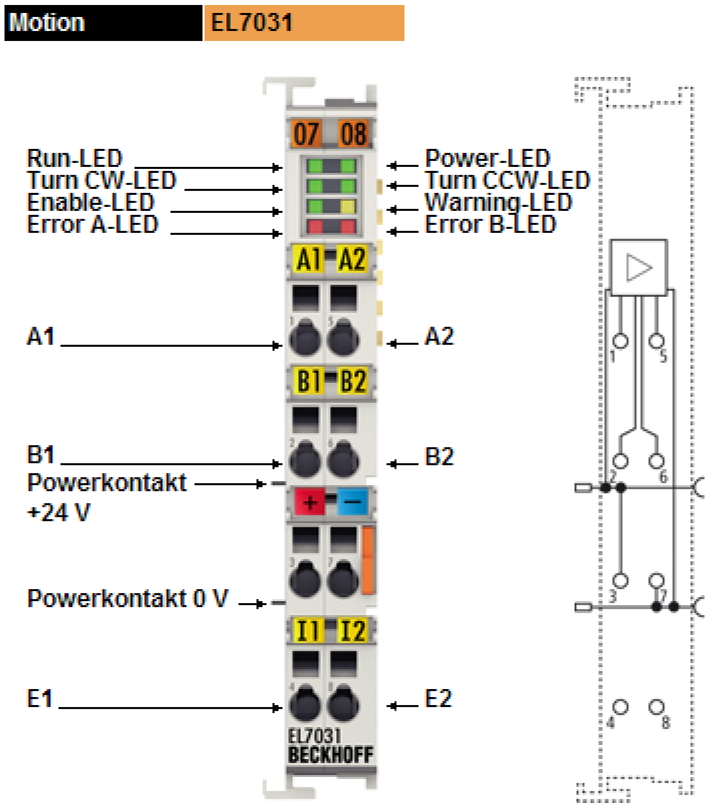
KL2541

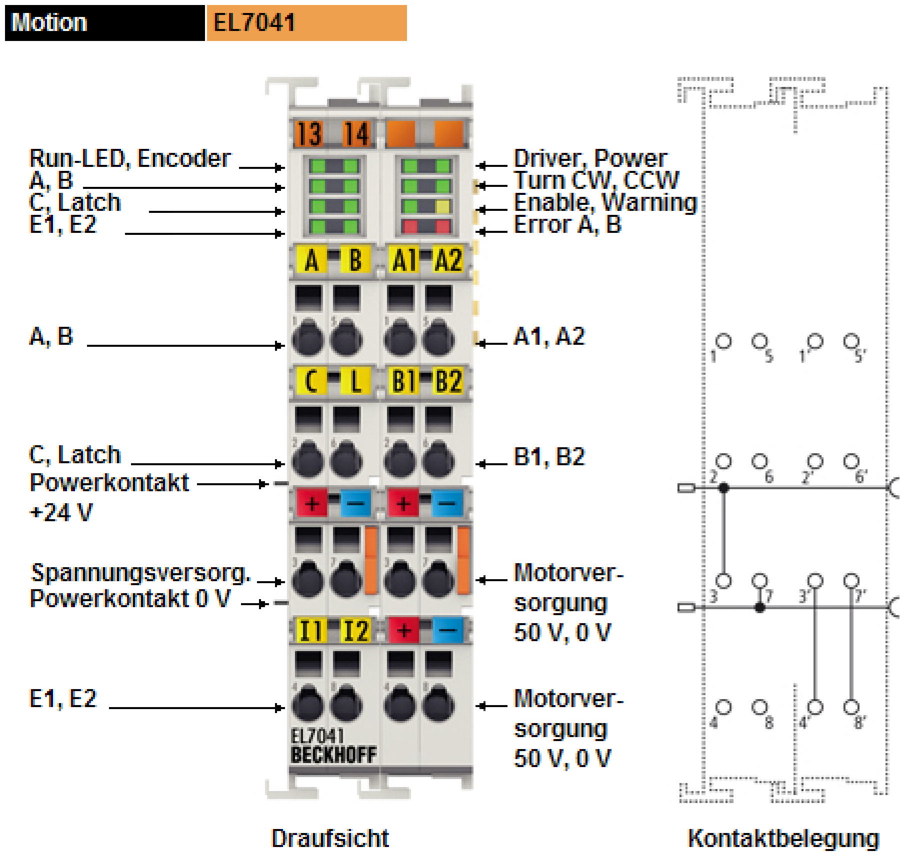


- 8…50 V DC
- **Output current**
  2 x 3,5 A, 2 x 5-A peak current
- **Maximum step frequency**
  > 125.000 steps/s
- **Step pattern** full step, half step, up to 64-fold microstepping
- **Current controller frequency**
  approx. 25 kHz

Axis types

EtherCAT stepper motor terminals, differences to KL25XX see data sheets





Axis types Low Cost stepper motor

Low cost stepper motor

The axis consists of a stepper motor which is connected to digital outputs and reacts to pulses (A/B from the terminals)

Fast pulse sequence -> motor turns quickly
Slow pulse sequence -> motor turns slowly

The set value (= pulse pattern) is generated by TwinCAT NC.

Axis types high /low speed

High/low speed axes

The axis responds to a two-stage set speed value including direction of rotation:

FAST/SLOW and FORWARDS/REVERSE

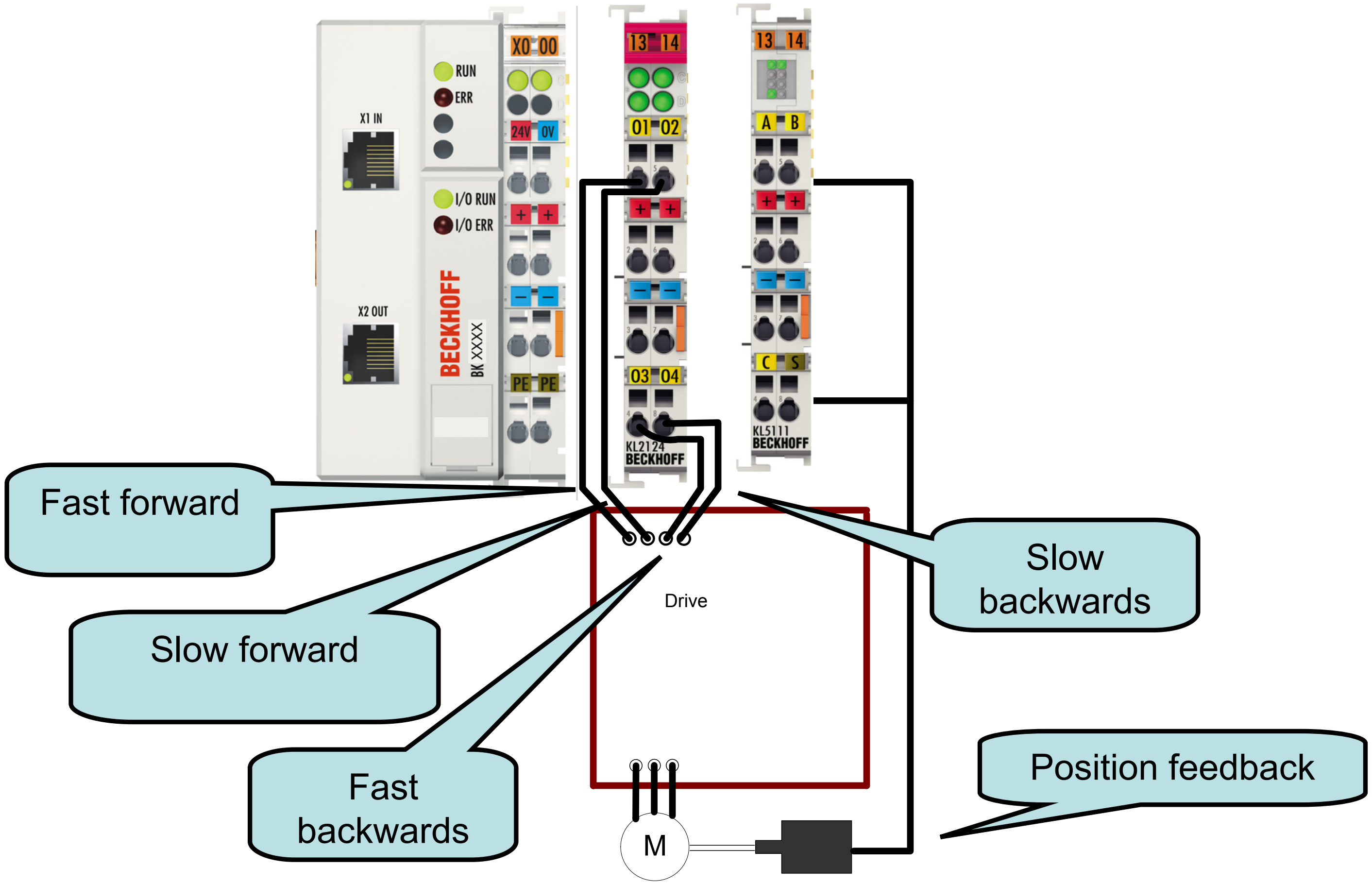
The set value is generated by TwinCAT NC,

e.g. frequency converter with fast/slow inputs, combination interlock.

Warning: Acquisition of actual value (Encoder is necessary)

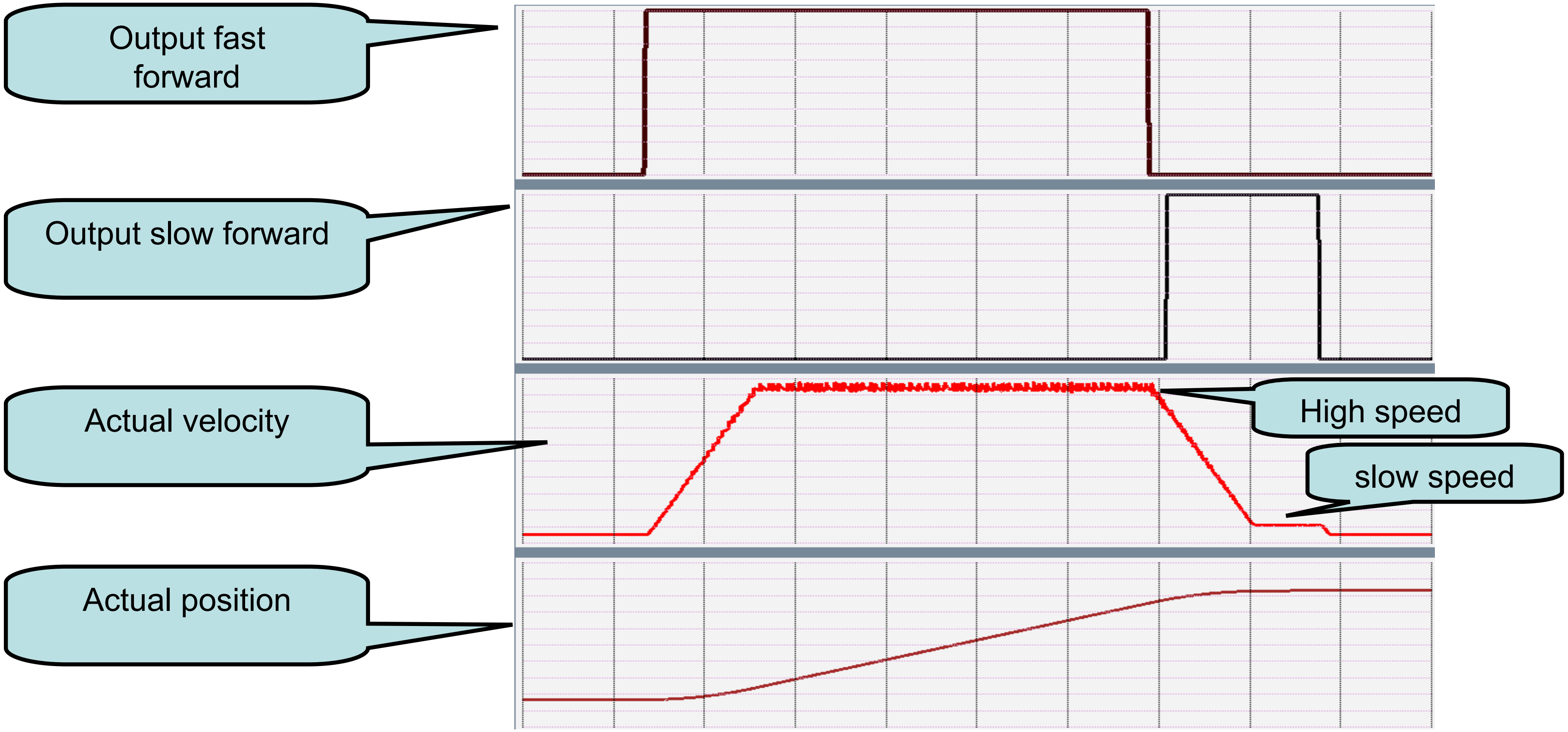
Axis types high /low speed

Possible scheme, further combinations and assignments adjustable



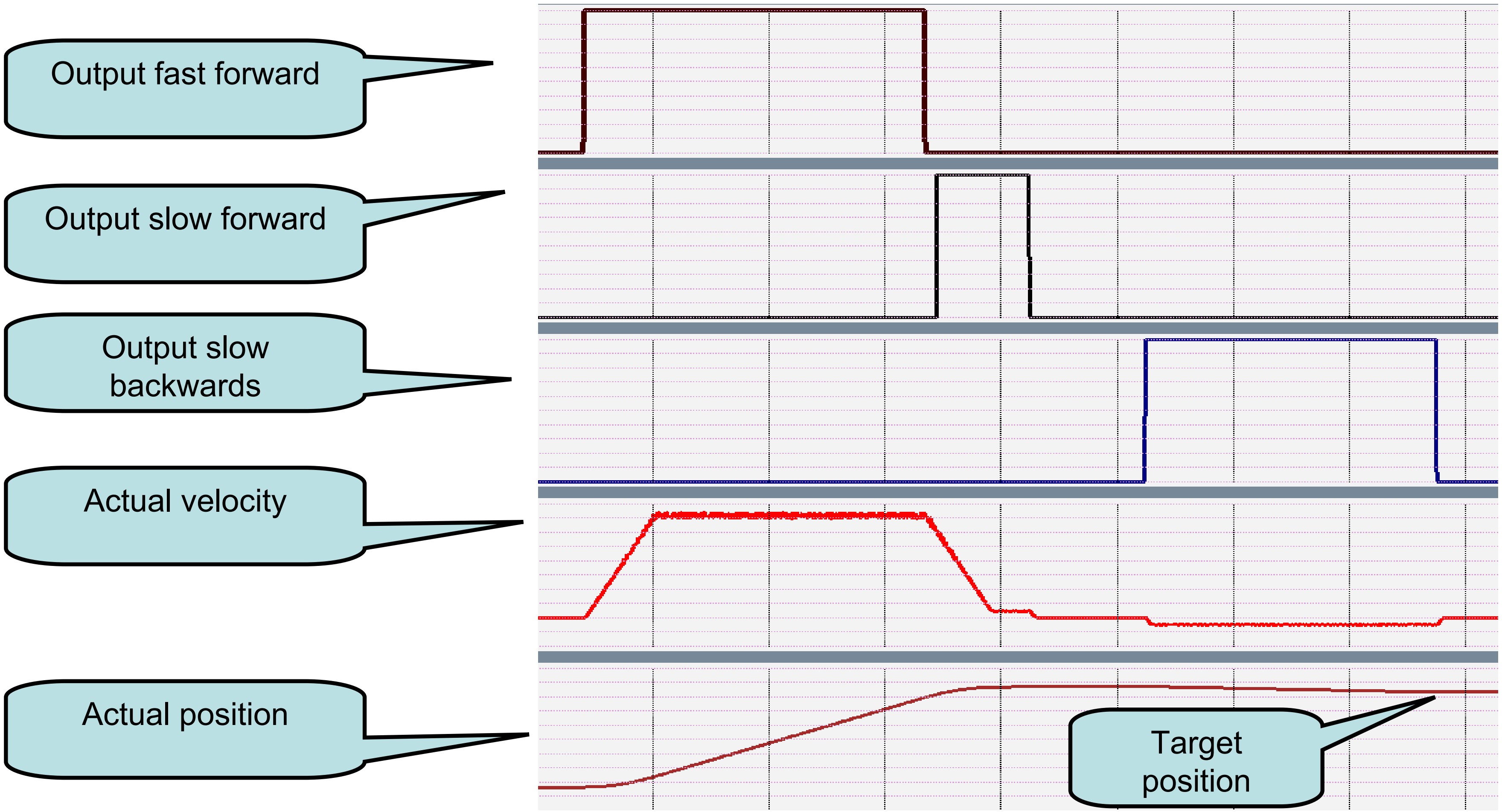
Axis types high /low speed

Example positioning in positvie direction



Axis types high /low speed

Example positioning in positive direction with loop movement (move to target from one direction, always)



Axis types

Stepper motor terminal (axis type continuous):

Stepper motor terminal, 24 V DC, 1,5 A
The KL2531 Bus Terminal is intended for the direct connection of different small stepper motors. The slimline PWM output stages for two motor coils are located in the Bus Terminal together with two digital inputs for limit switches. The KL2531 can be adjusted to the motor and the application by changing just a few parameters. 64-fold microstepping ensures particularly quiet and precise motor operation. In many applications, integrated monitoring of the mechanical load makes an encoder system or limit switch unnecessary.

Axis types

Low cost stepper motor, Hardware

e.g. 24 Volt stepper motor with 2A output terminals

An encoder is NOT required
for acquisition of the actual value, since the pulses that are output are counted.
! The mechanical design and/or maximum rotary speed/torque should be examined to ensure that the motor will be able to "keep up", since an output terminal cannot provide an increased voltage at higher frequency

Axis types

Virtual encoder axis,

An axis that only consists of an encoder.

"Normal" (continuous) axes can be coupled to this axis as slaves, and follow the set encoder value of the virtual encoder axis. (Gear ration possible)

HAND WHEEL FUNCTION

Functional principle of the TwinCAT NC

Output is a speed value
The actual position is monitored.

Output:

Speed pre-control
+ controller output
(acceleration pre-control also is optional)

Feedback:

Actual position value
At specific axis types e.g. SERCOS is also a direct output of the Setposition in NC time possible.

Functional principle of the TwinCAT NC

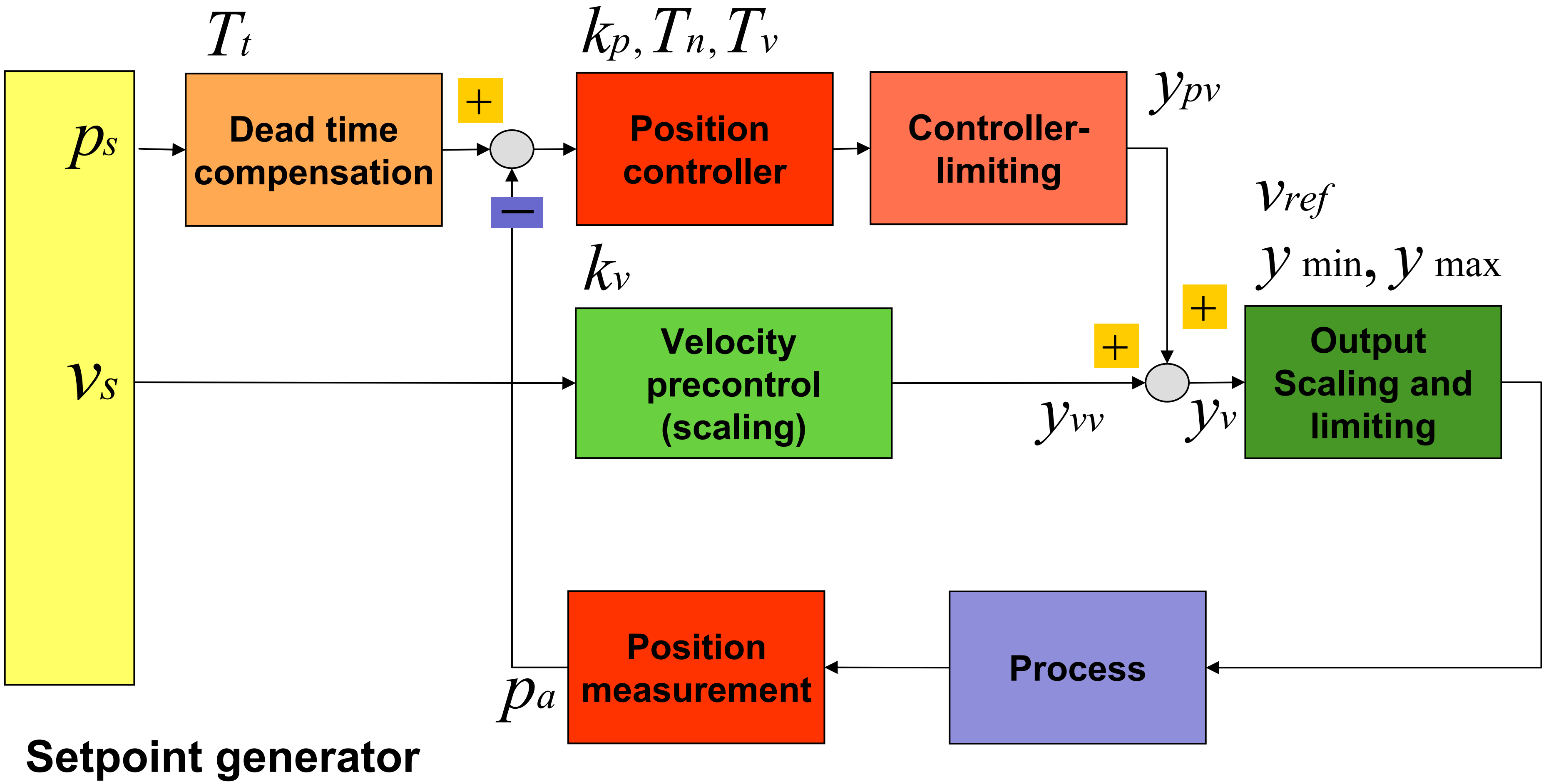
TwinCAT NC works with a velocity pre control.

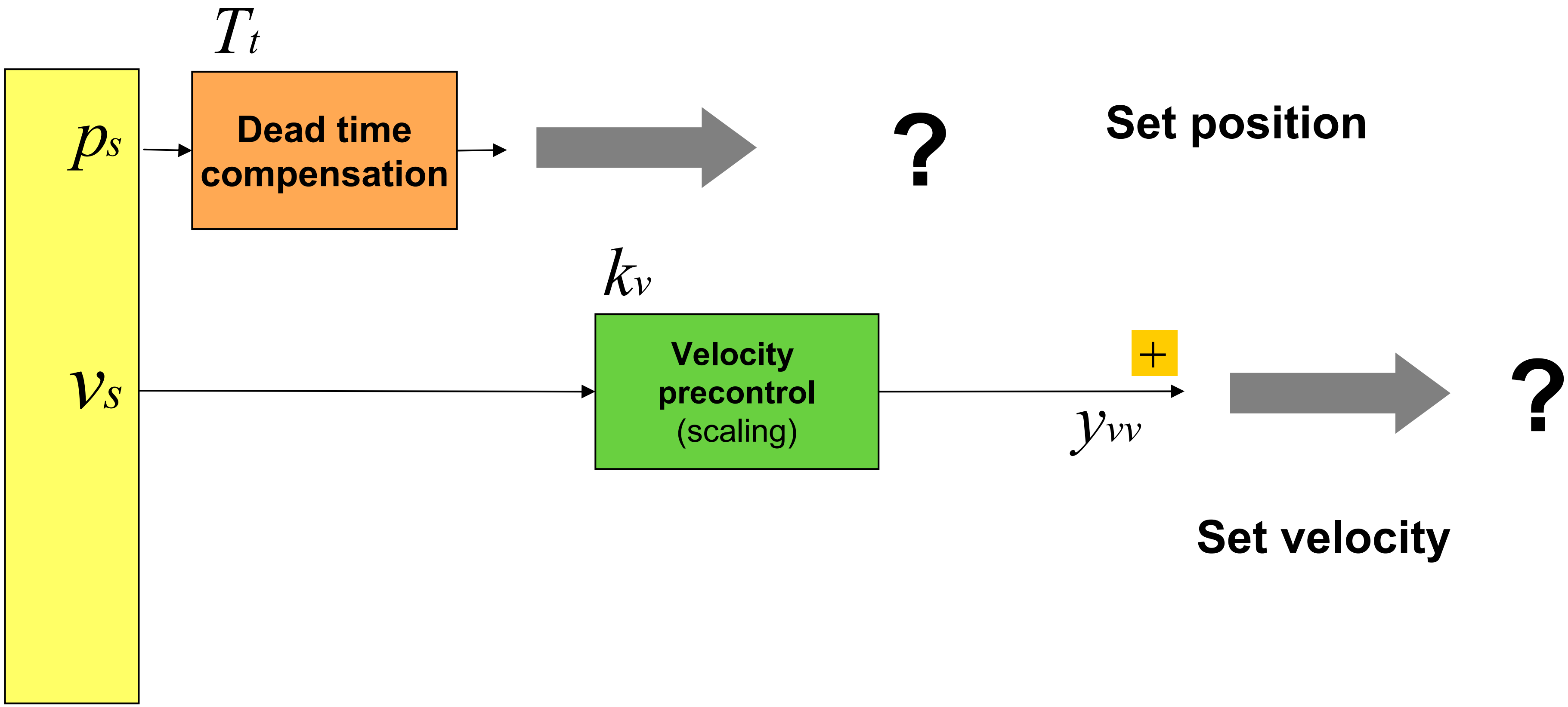
The Position controller controls the observance of the set position („Motion“ and position control).

Further available options:
-Acceleration pre control
-Position control with two P constants
-direct output of the position. (Sercos Axes)

-High / low speed controller
-Stepper motor controller
-External Setpoint generation (ab TwinCAT 2.9)
-Linearisation of pre control for non linear axes (Hydraulic axes).

Functional principle of the TwinCAT NC





Setpoint generator

Set value profiles

The profile of the velocity output can be variied during an defined brake time

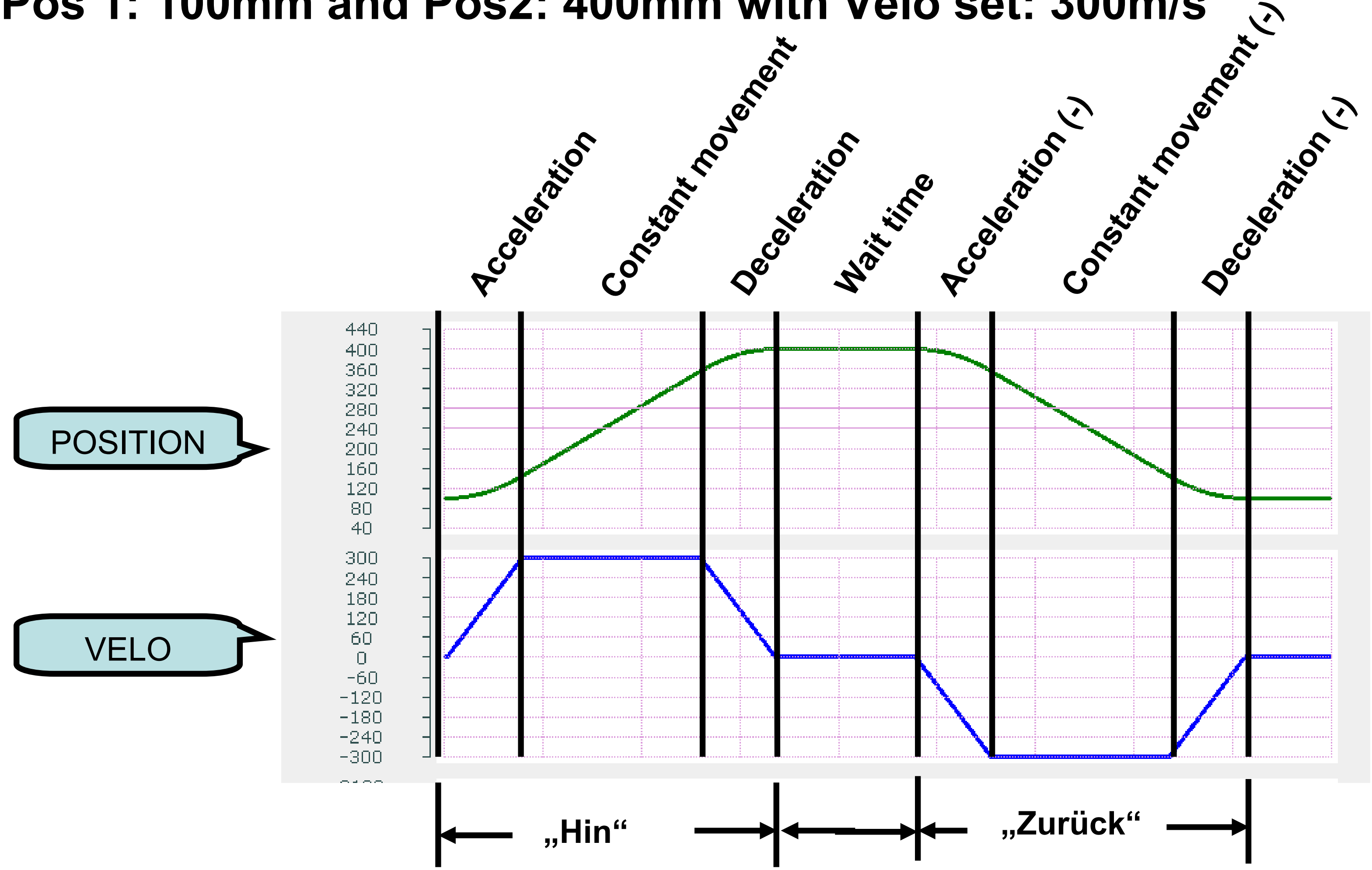
Thereby the acceleration change (jerk) can be reduced considerably.

This works out on in the short run mechanical burdens and commensurate with as well on the electric burden of the drive.

A smaller jerk requires higher acceleration, if the axis should reach the demanded velocity within the same time.

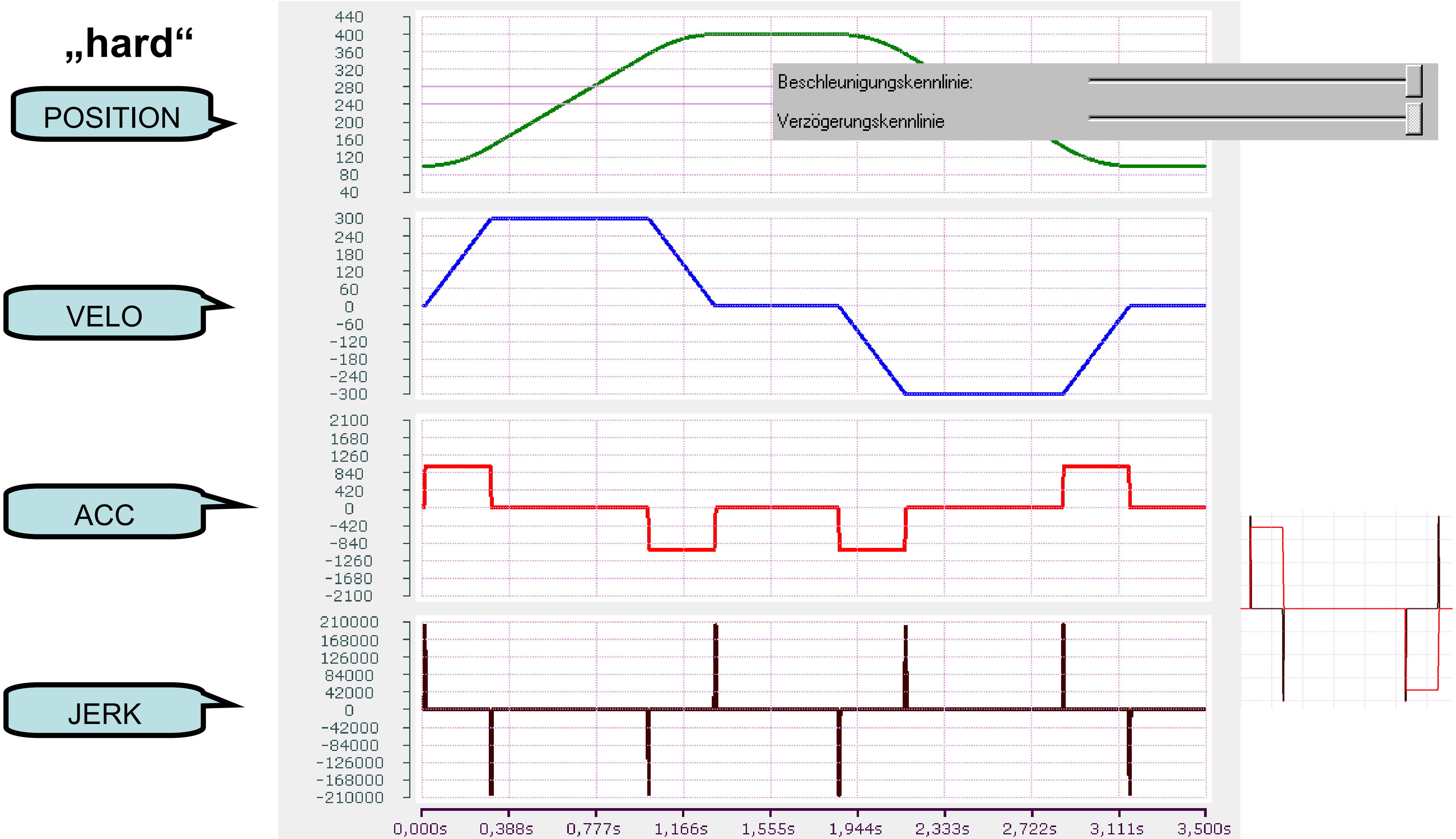
Set value profiles

Example reverse axis between
Pos 1: 100mm and Pos2: 400mm with Velo set: 300m/s



Set value profiles

„hard“



Set value profiles

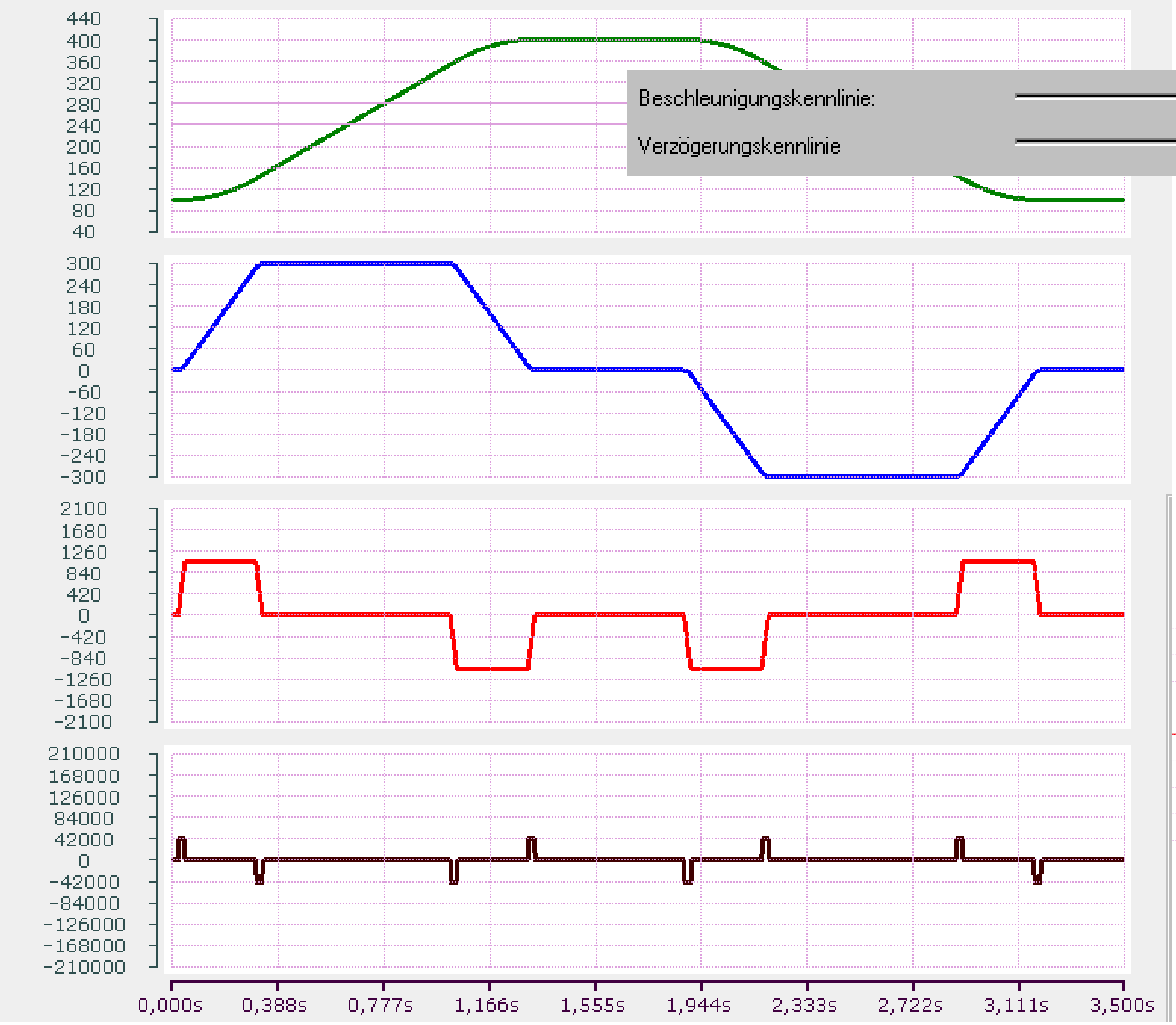
„hard (2)“

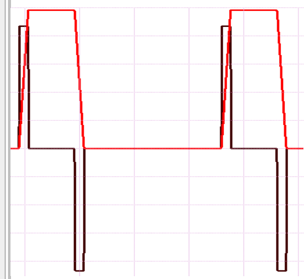
POSITION

VELO

ACC

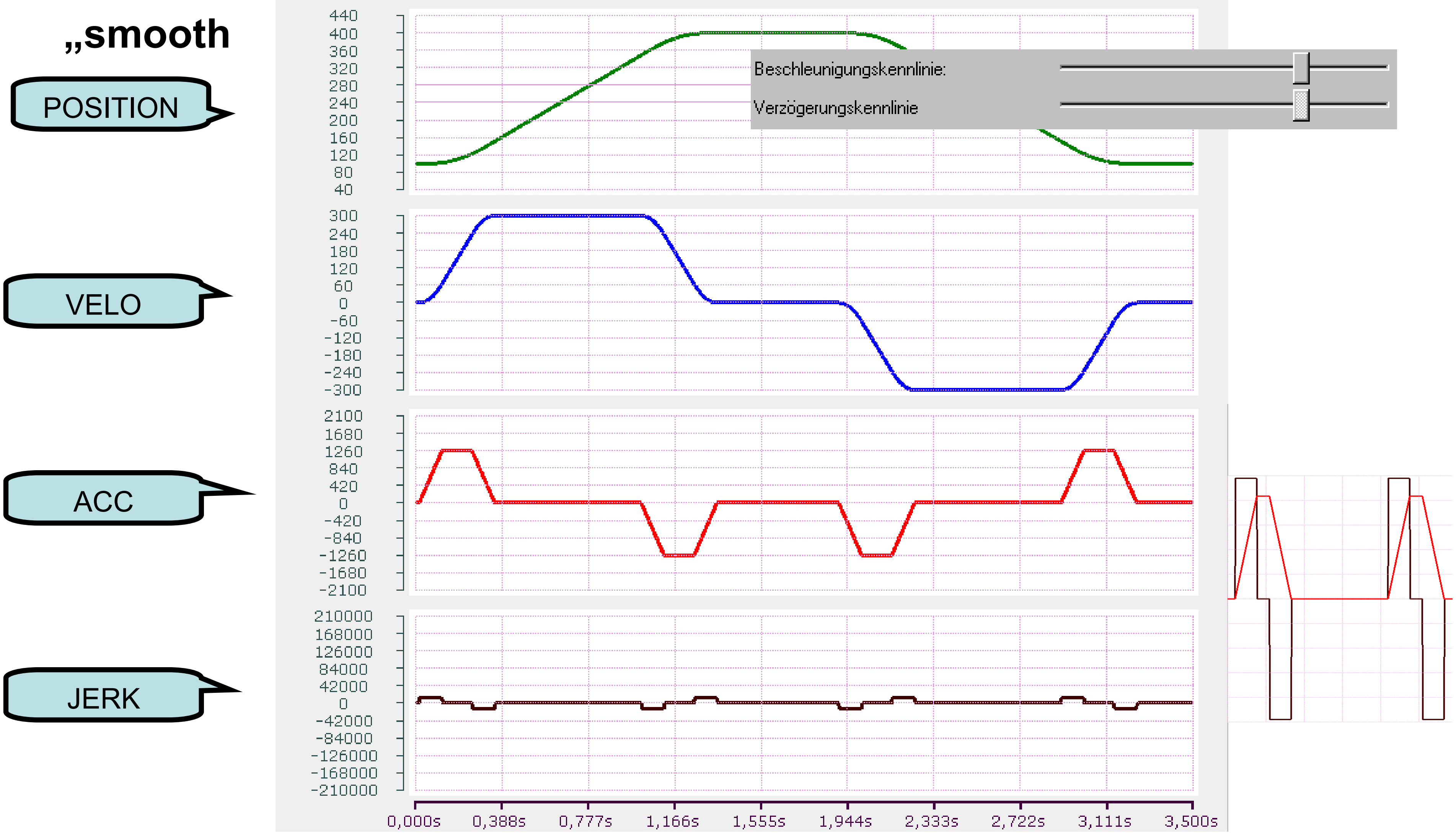
JERK





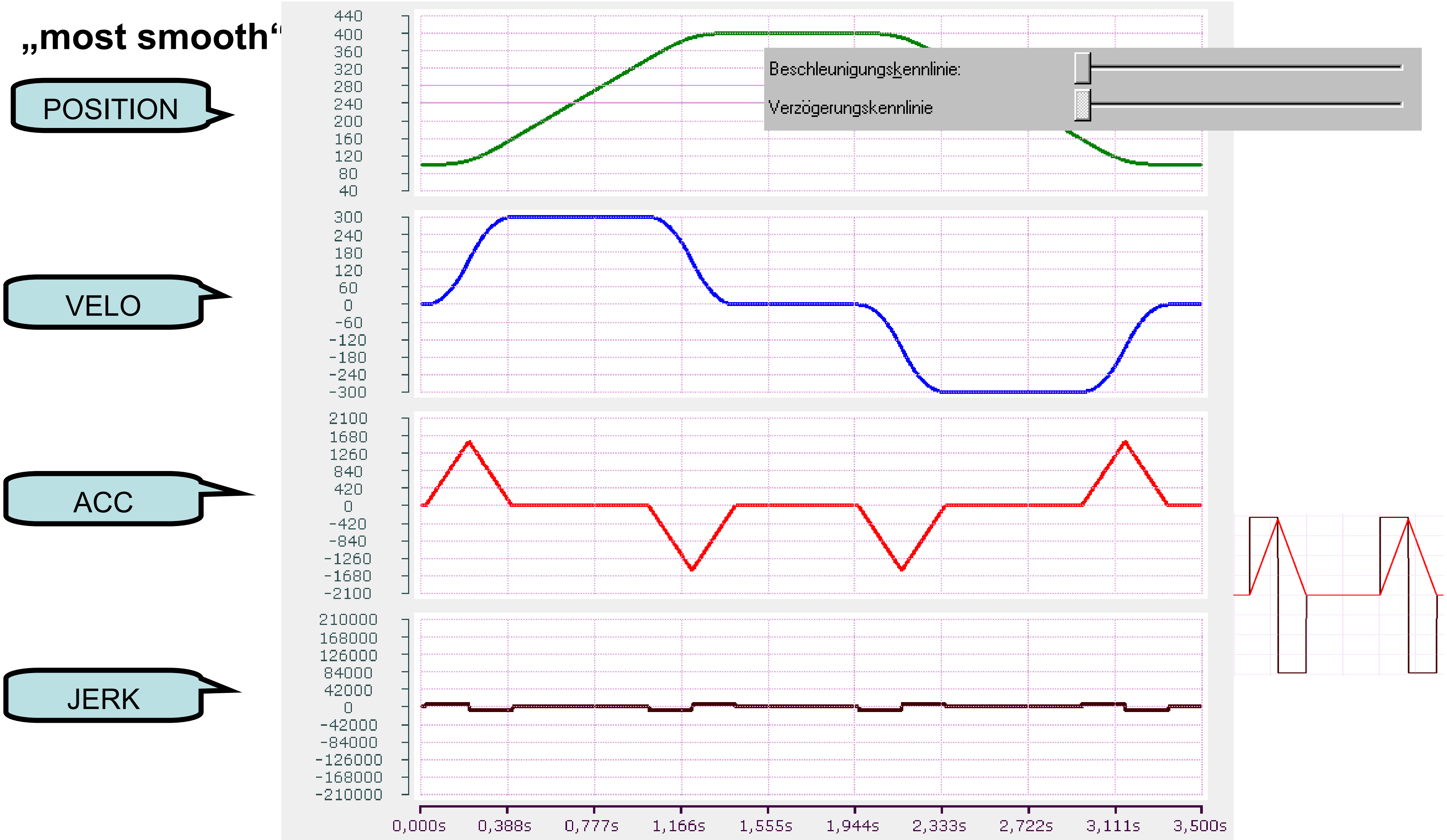
Set value profiles

„smooth



Set value profiles

„most smooth“



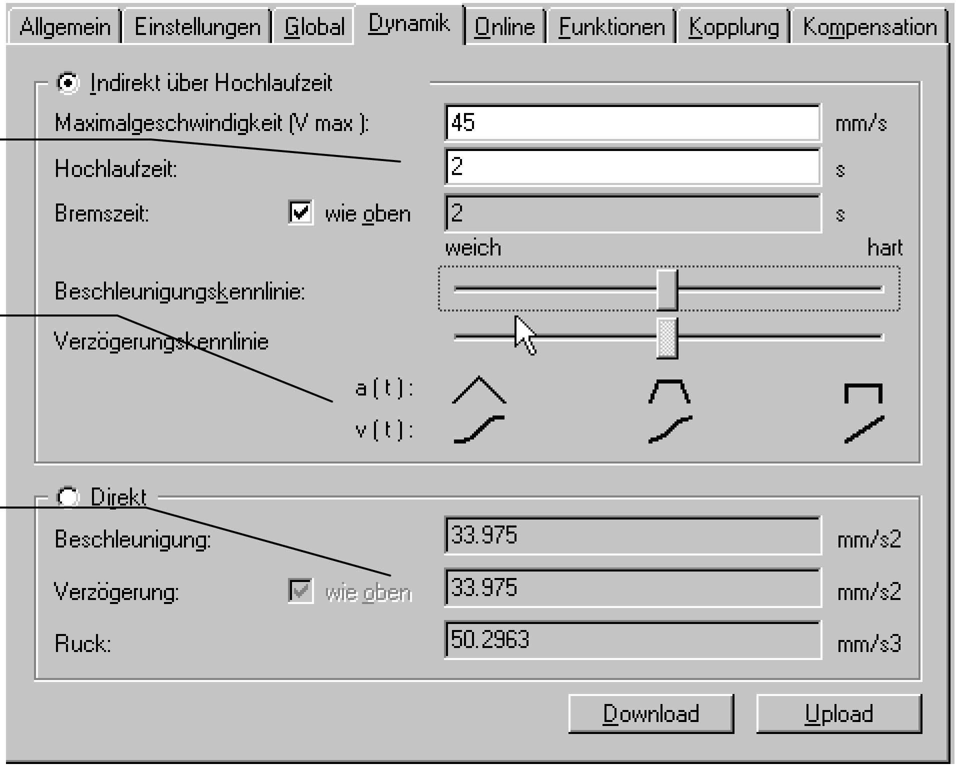
Set value profiles

The presetting can be done easily via the presetting of the run-up time and the selcetion of the profile in the System Manager!

Input via run-up time

Preselect profile

Calculation by the TwinCAT System Manager

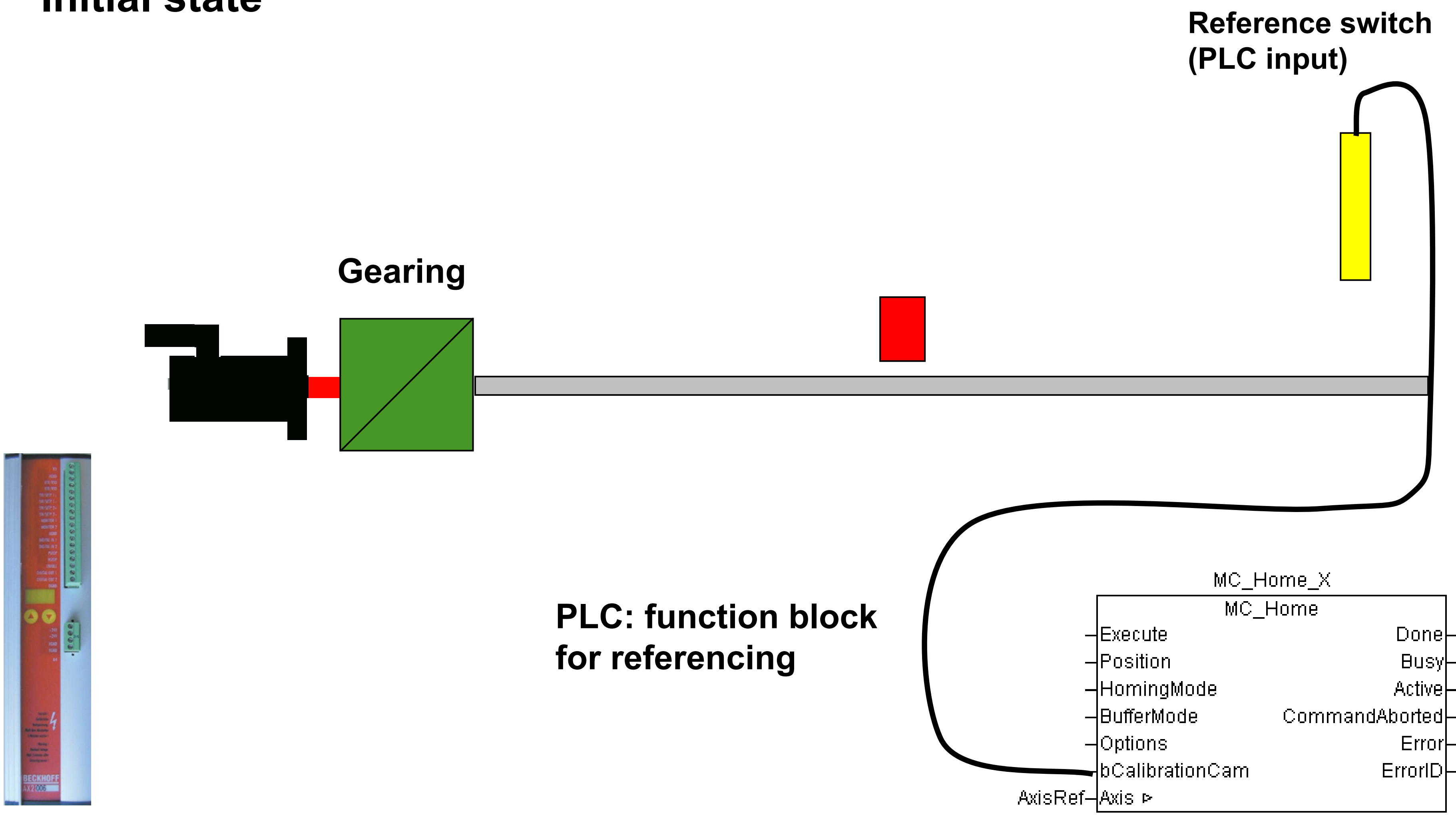


Referencing

- Referencing (calibrate) is necessary for axis with not absolute encoder systems.
  Incremental Encoder, Single Turn
- Absolute Encoder, or not absolute encoder systems direct from the drive, (e.g. actual position value of AX2000).
- At referencing the axis is lead to a fix reference position and the encoder is set to the current actual position.

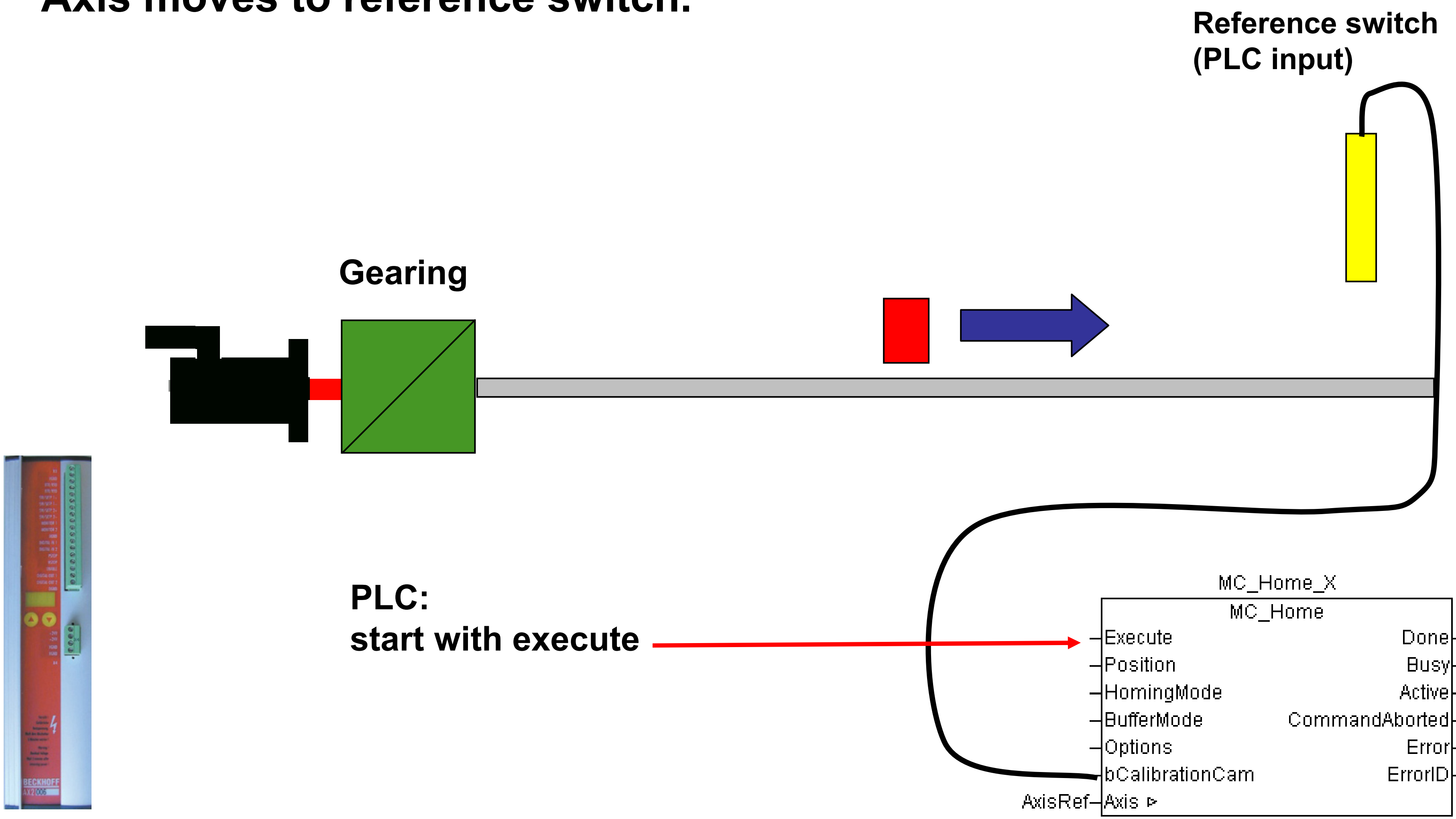
Referencing

Initial state



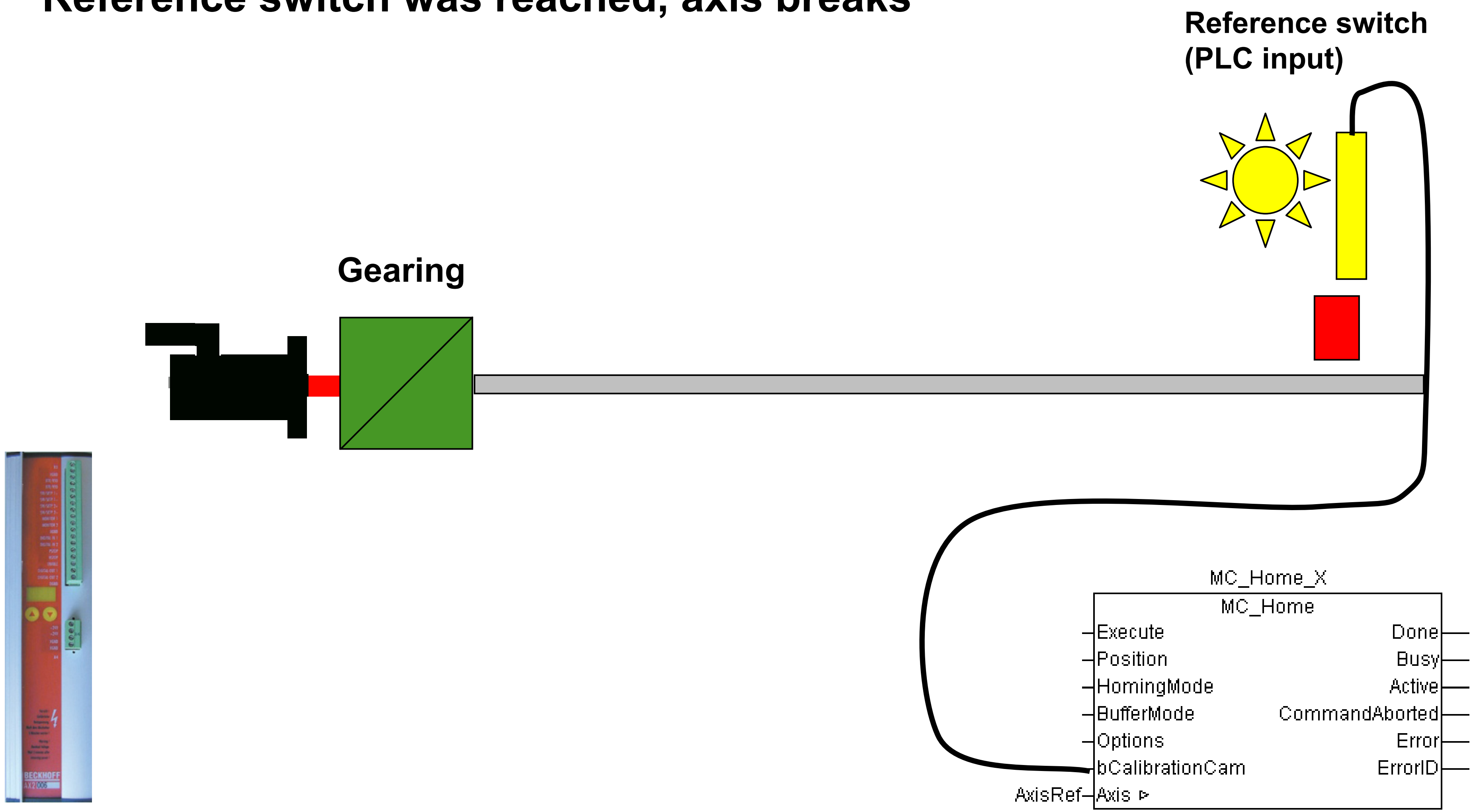
Referencing

Axis moves to reference switch.



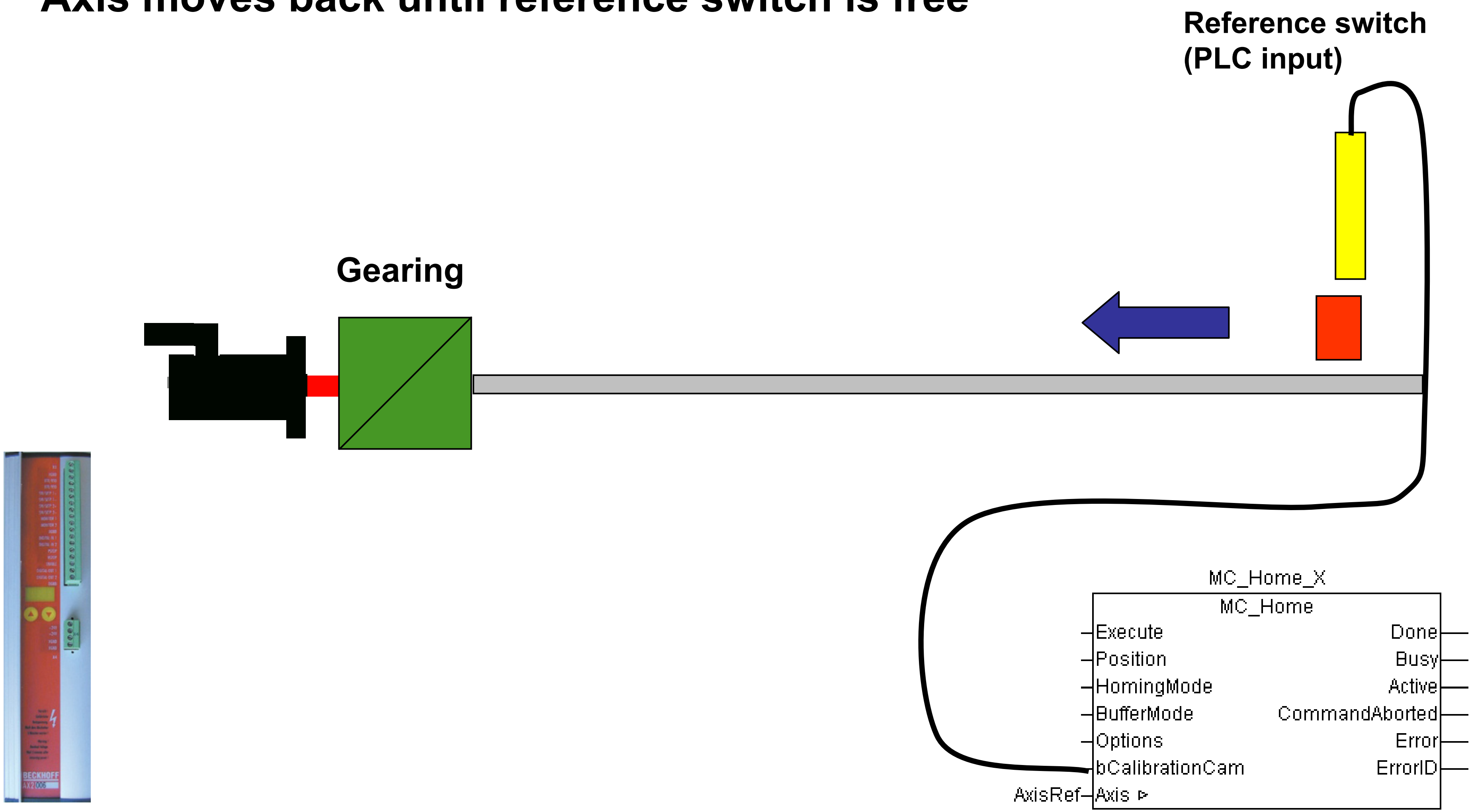
Referencing

Reference switch was reached, axis breaks



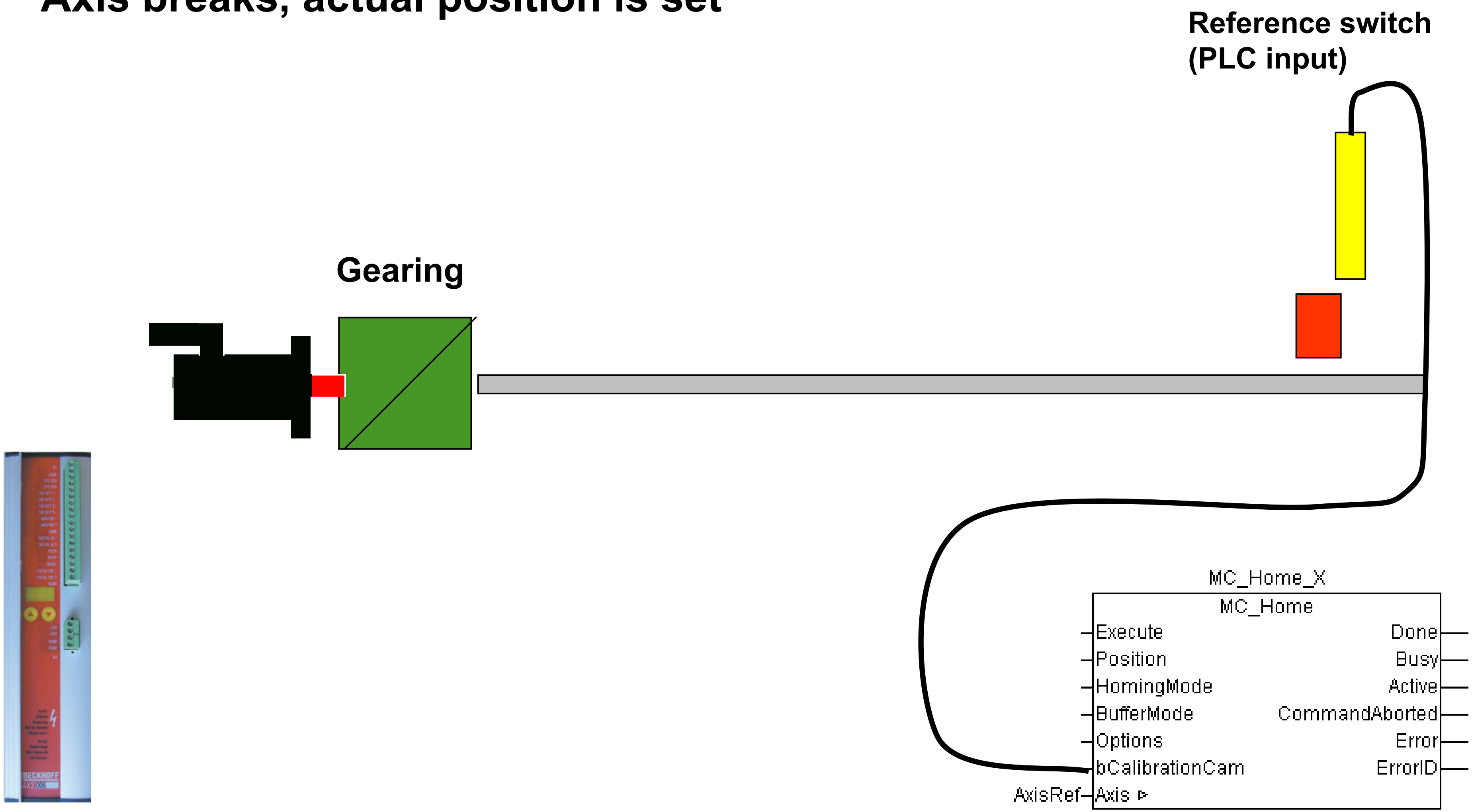
Referencing

Axis moves back until reference switch is free



Referencing completed (a)

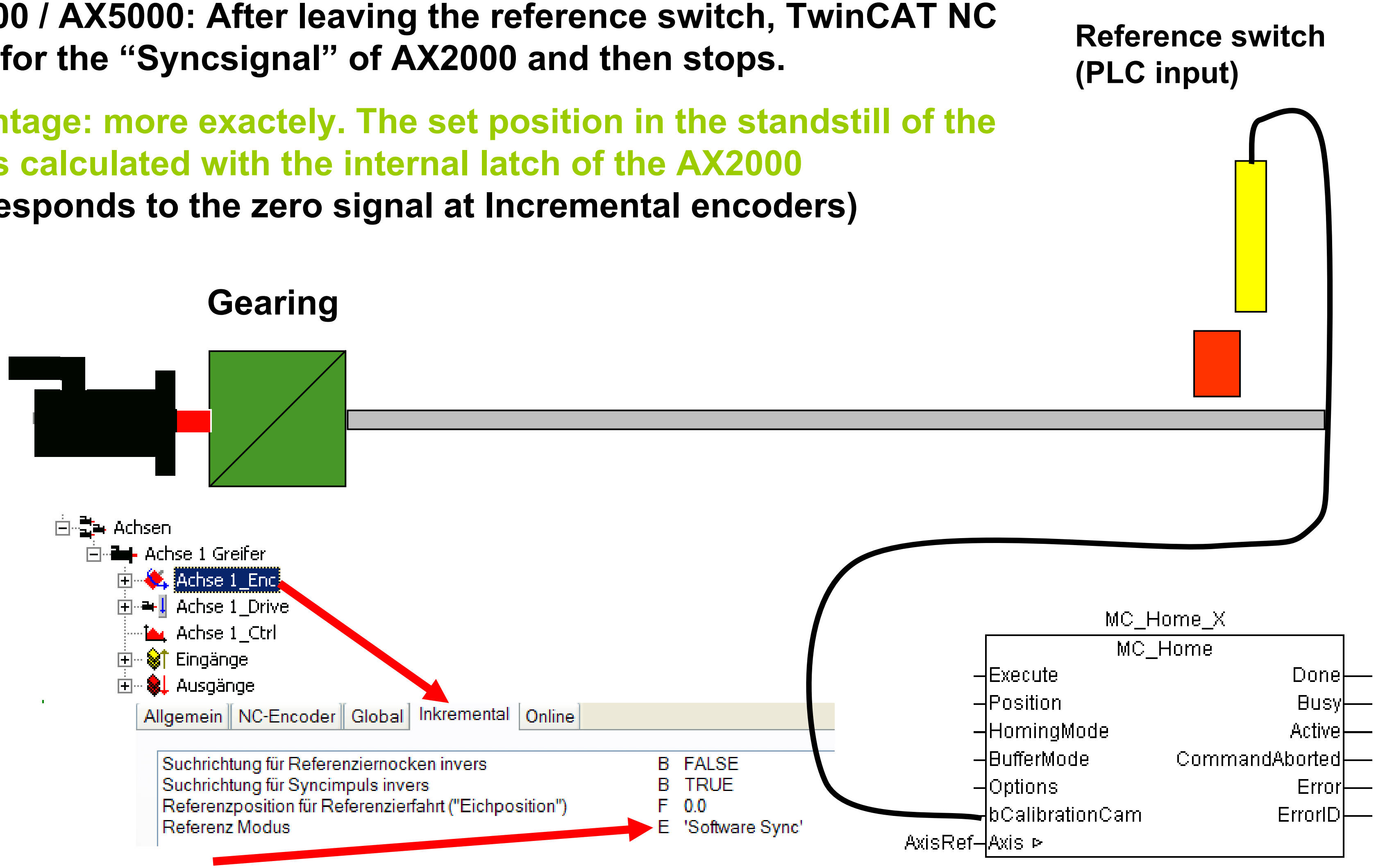
Axis breaks, actual position is set



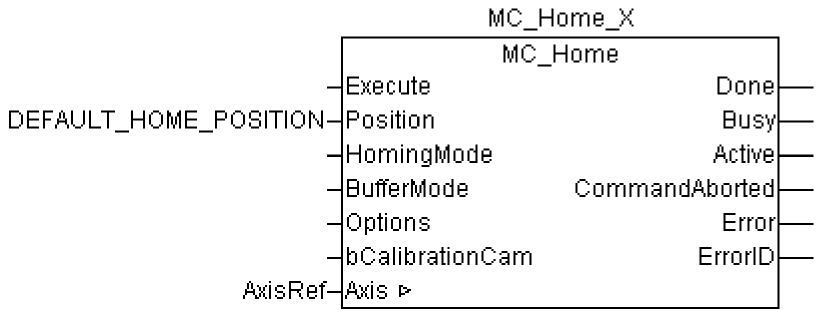
Referencing completed (b)

- **AX2000 / AX5000: After leaving the reference switch, TwinCAT NC waits for the "Syncsignal" of AX2000 and then stops.**

  Advantage: more exactly. The set position in the standstill of the axis is calculated with the internal latch of the AX2000
  (corresponds to the zero signal at Incremental encoders)

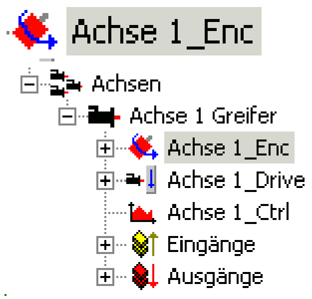


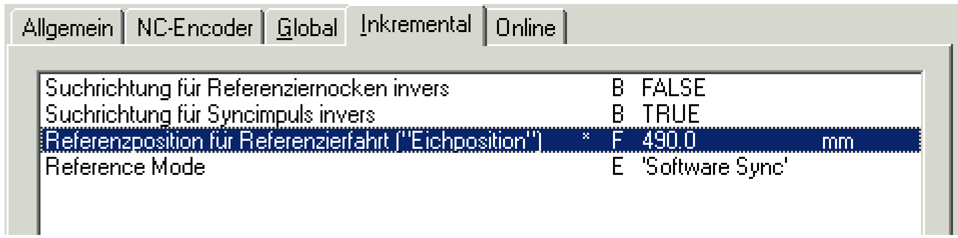
Referencing completed. Which position is set?



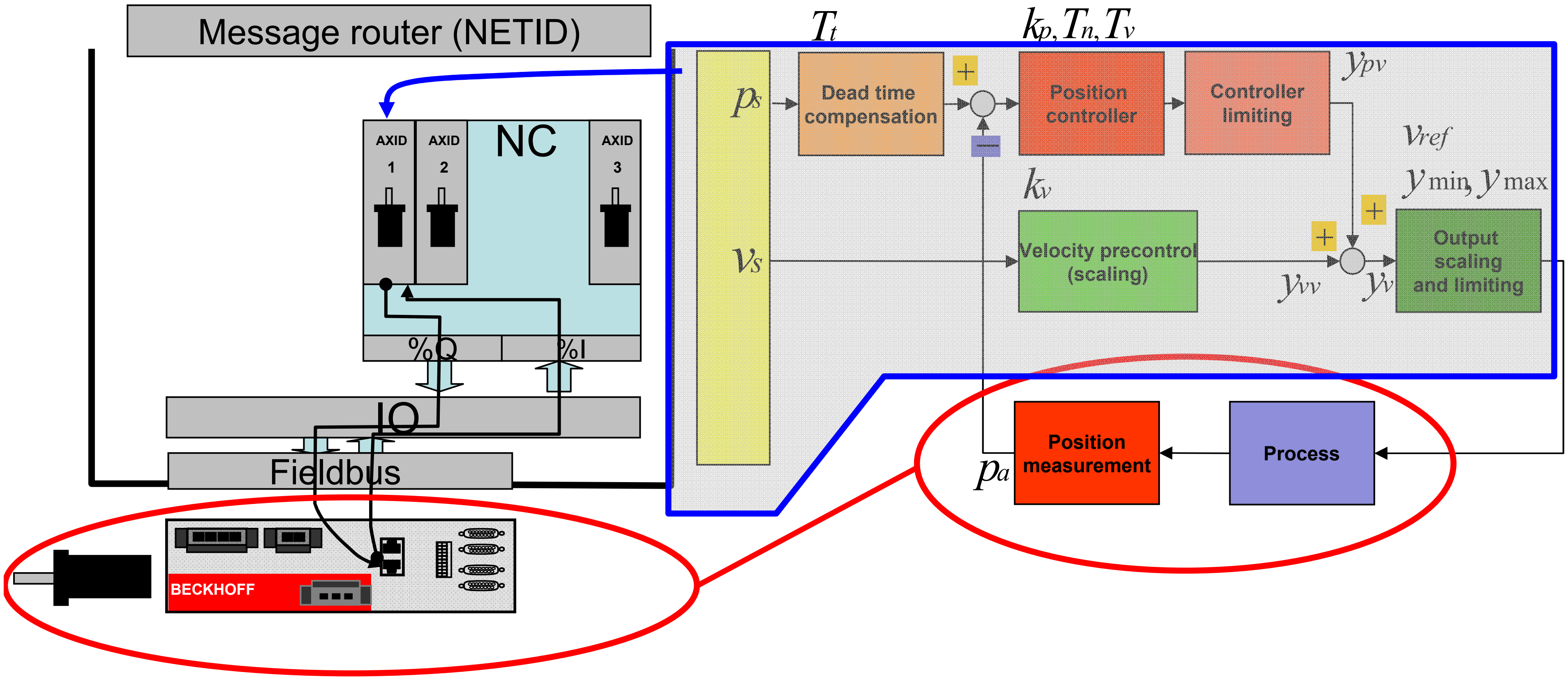
If „Position“DEFAULT_HOME_POSITION (global variable from TCMC.LIB) is submitted at the Fb input, the value is taken out of the System Manager.

Otherwise the value is taken at the input „Position“





NC scheme in the TwinCAT architecture



NC settings demo-rack

- Configuration:
- PC as TwinCAT Controller with EtherCAT interface
- AX 5203 in operating mode velocity interface
- AX base configuration is finished in the System Manager

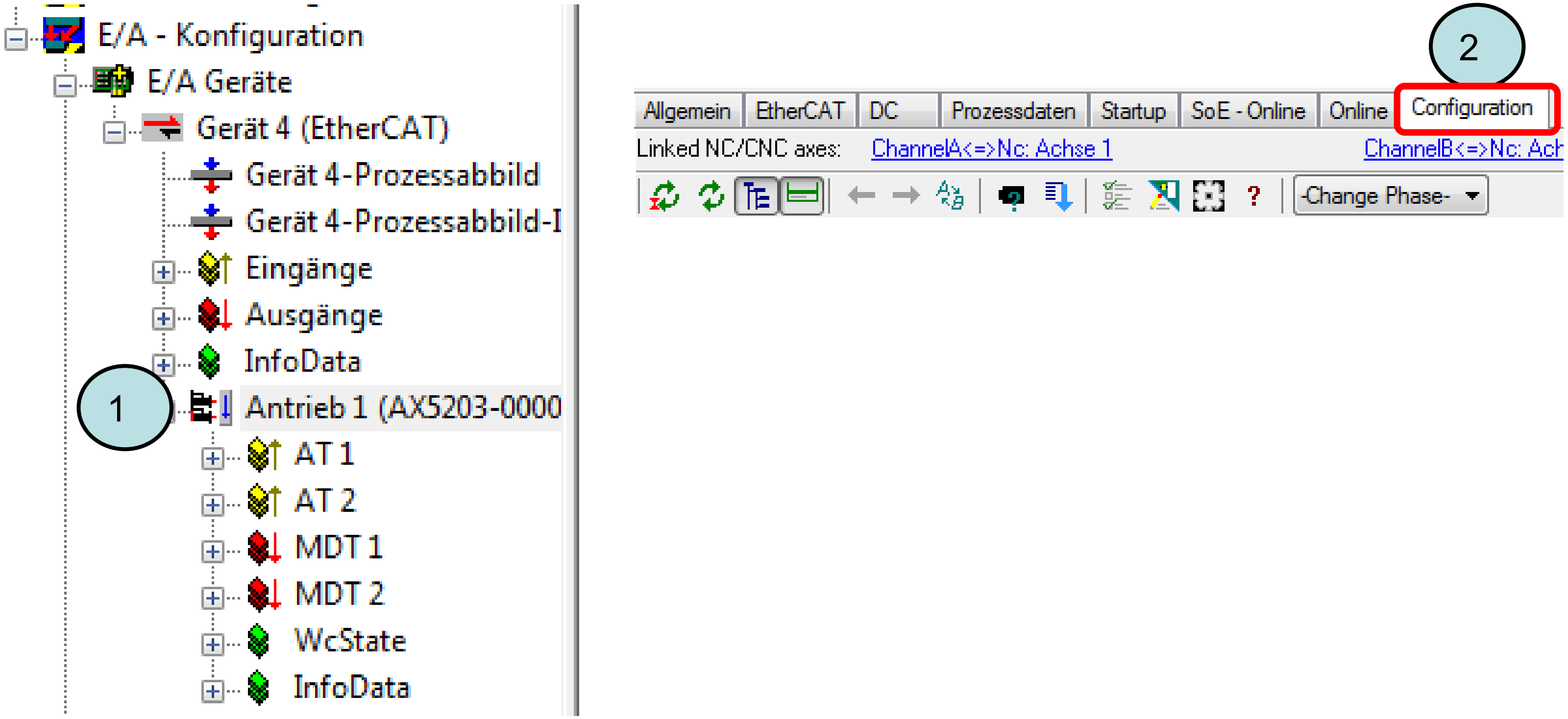
- Data mechanical / electrical
- Assumption: **1 rotation motor corresponds to 10 mm path** (linear movement)

- Drive data:
- Maximum speed velocity controller in the AX5000: 1500 U/min
- Feedback resolution: 20bit : 1 rotation cornensponds to 1 048 576 increments

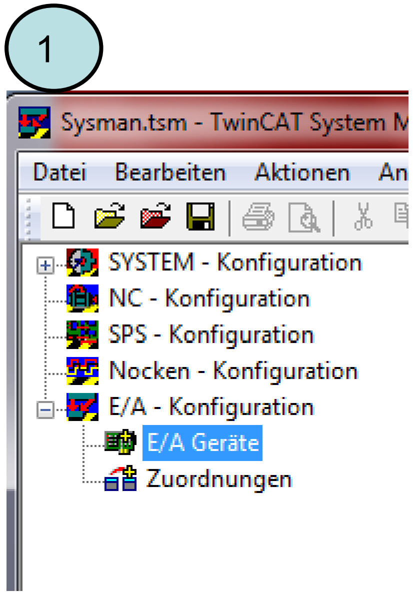
Settings are only valid for „Demorack Training“

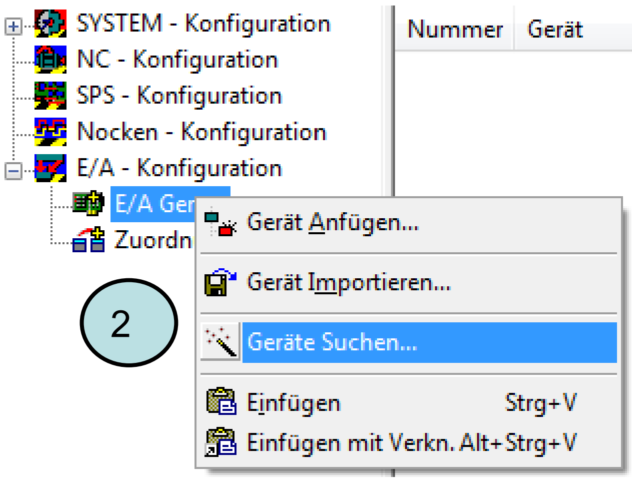
NC Settings Demorack

- Bei AX5000: basicconfiguration Drivetool (only valid for trainingrack)

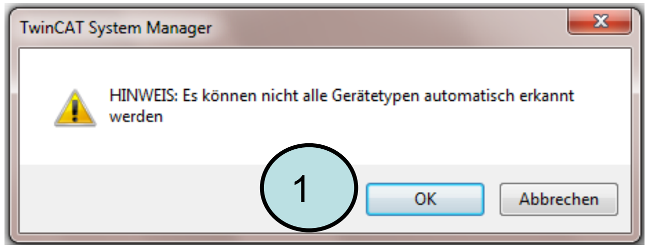


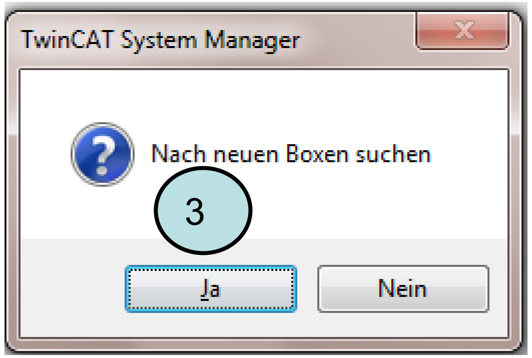
NC Settings Demorack

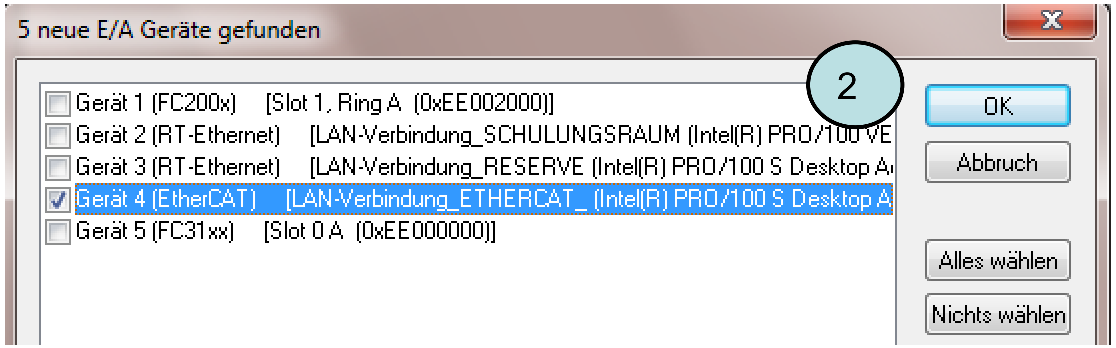




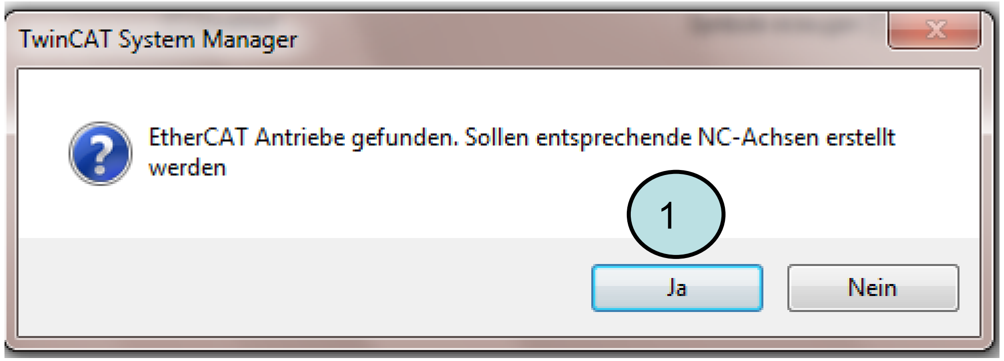
NC Settings Demorack

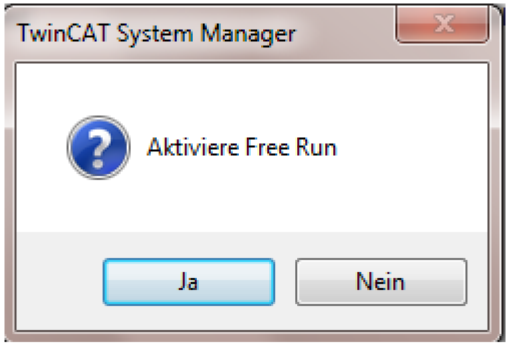




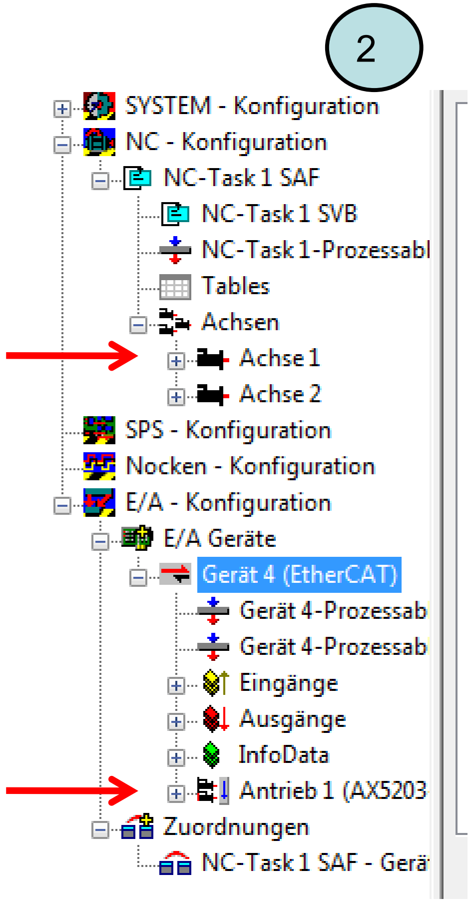


NC Settings Demorack

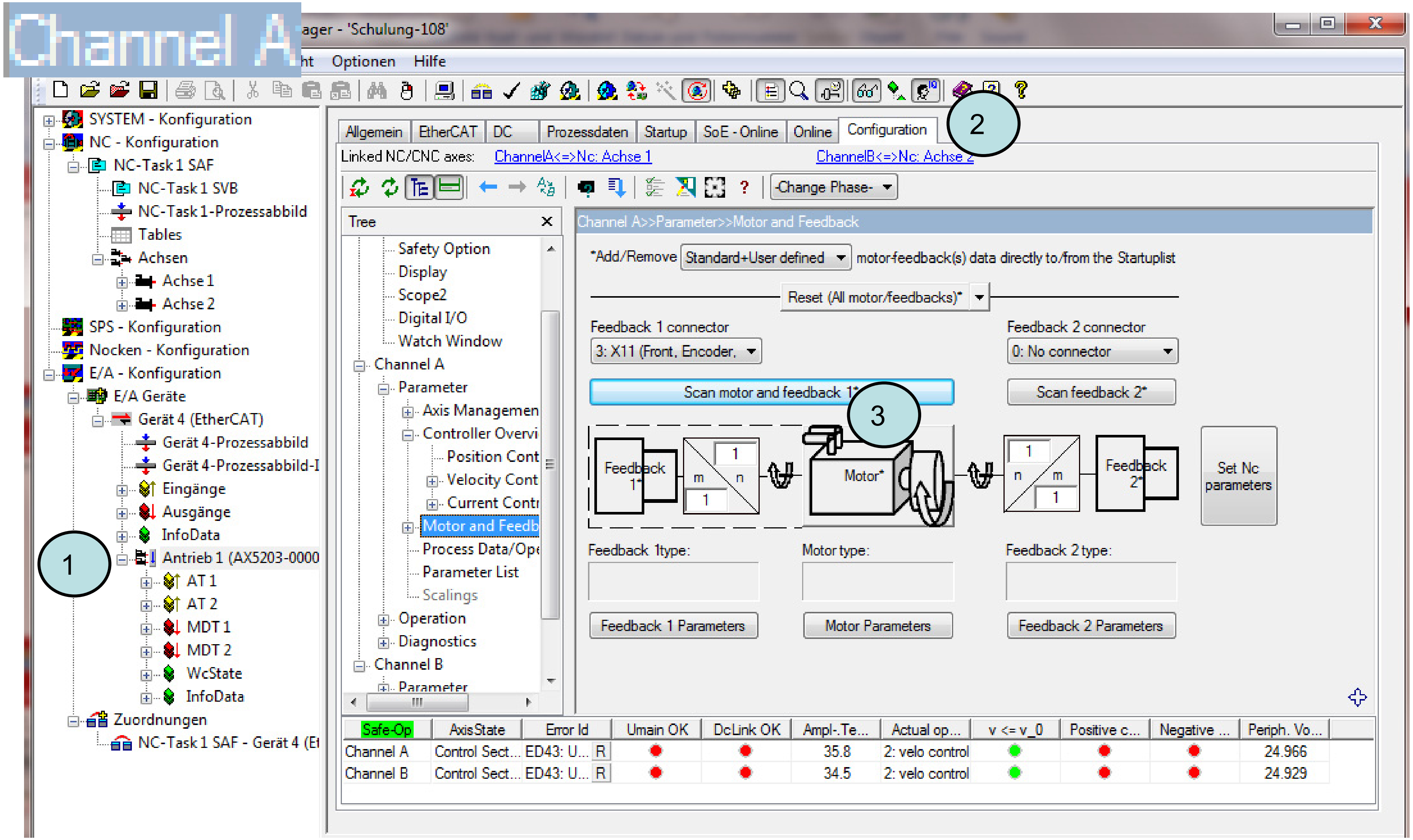




2

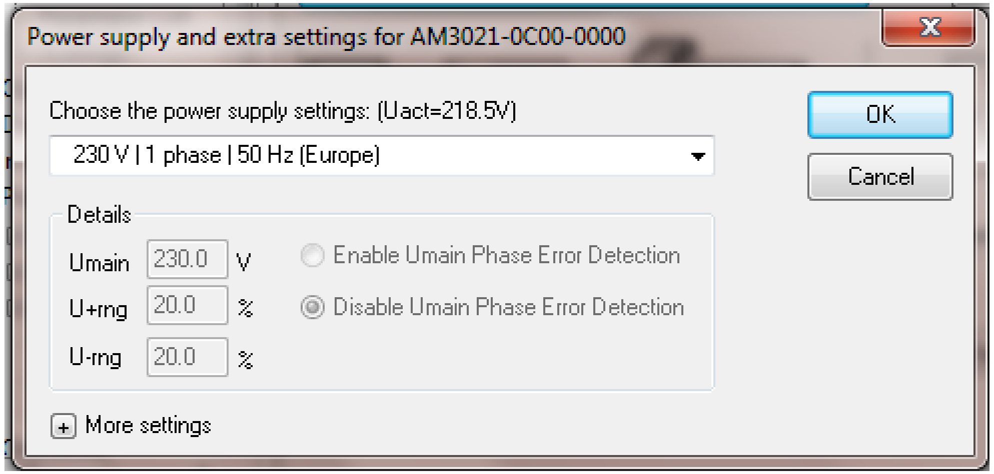


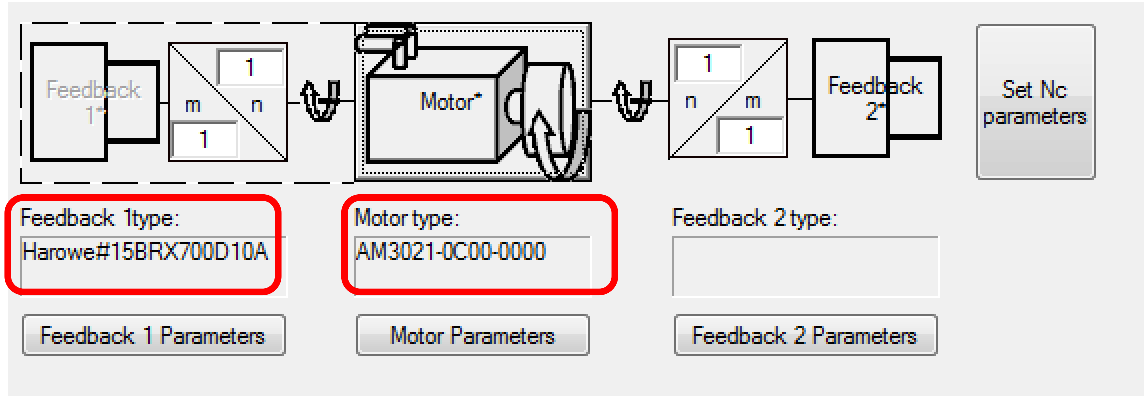
NC Settings Demorack



NC Settings Demorack

Channel A

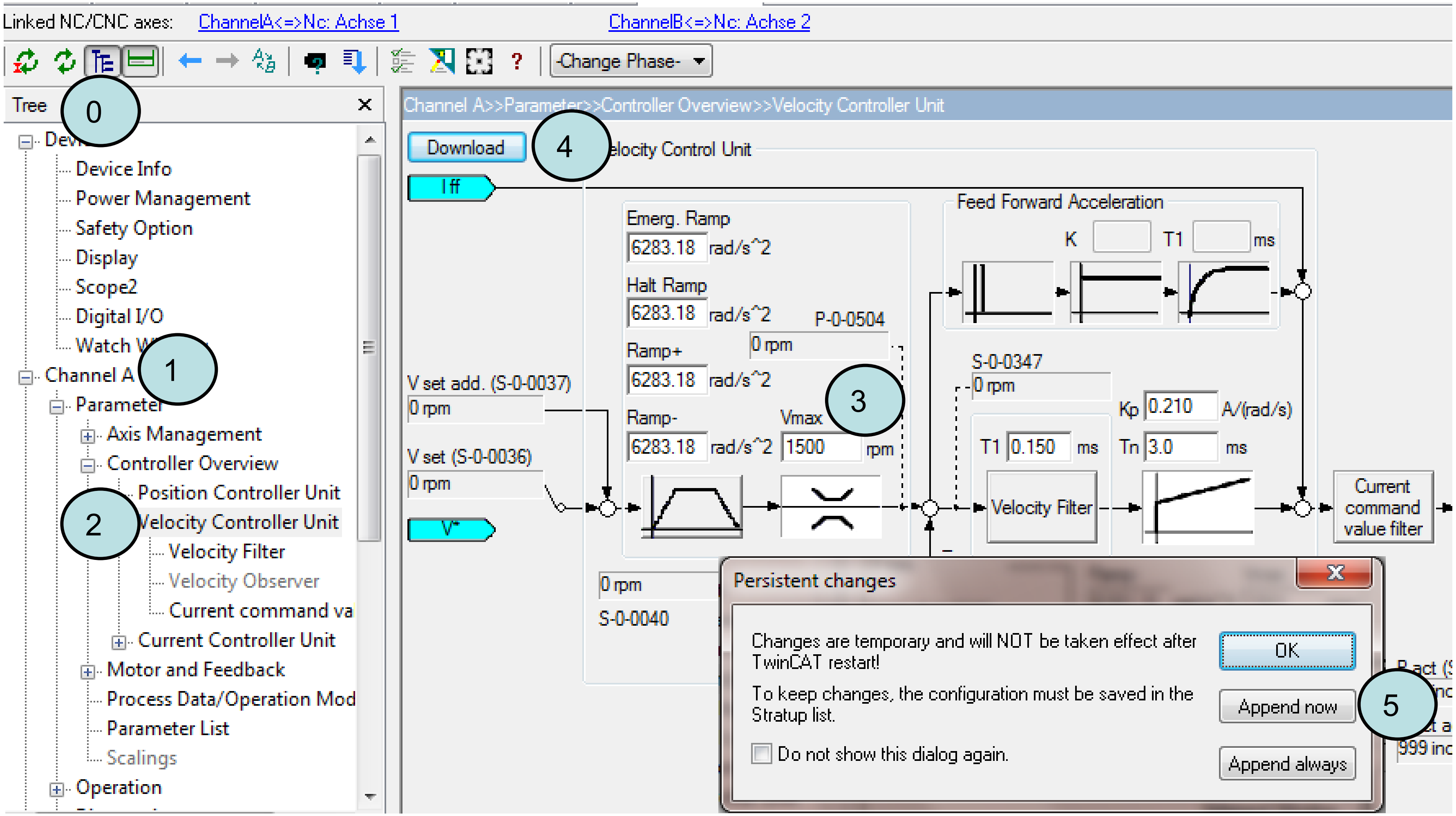




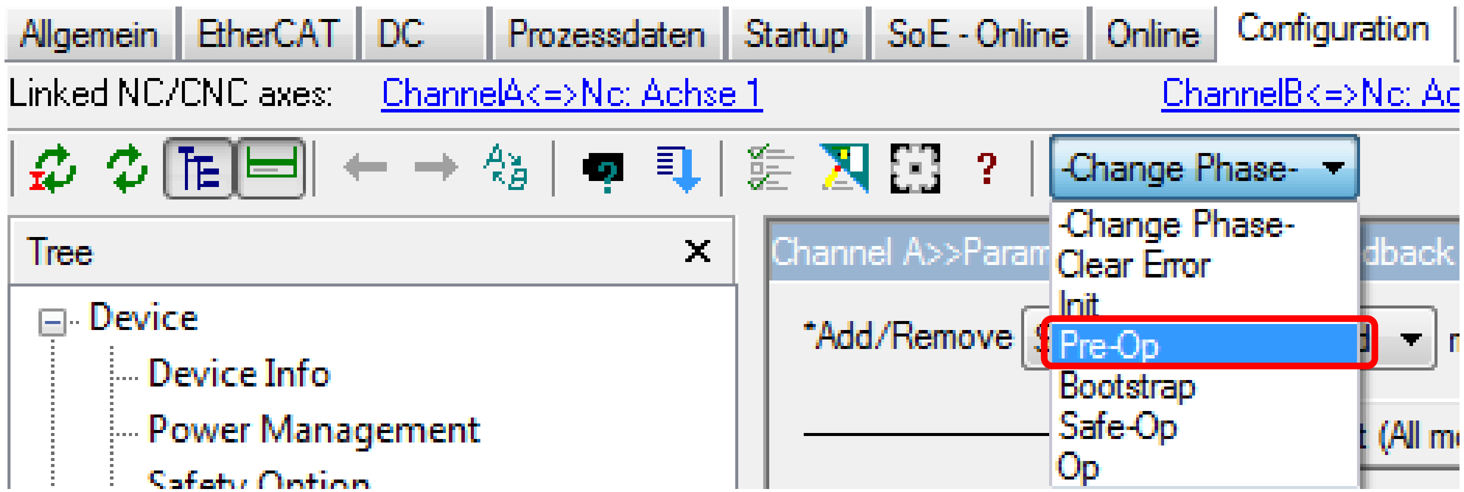
NC Settings Demorack

Channel A

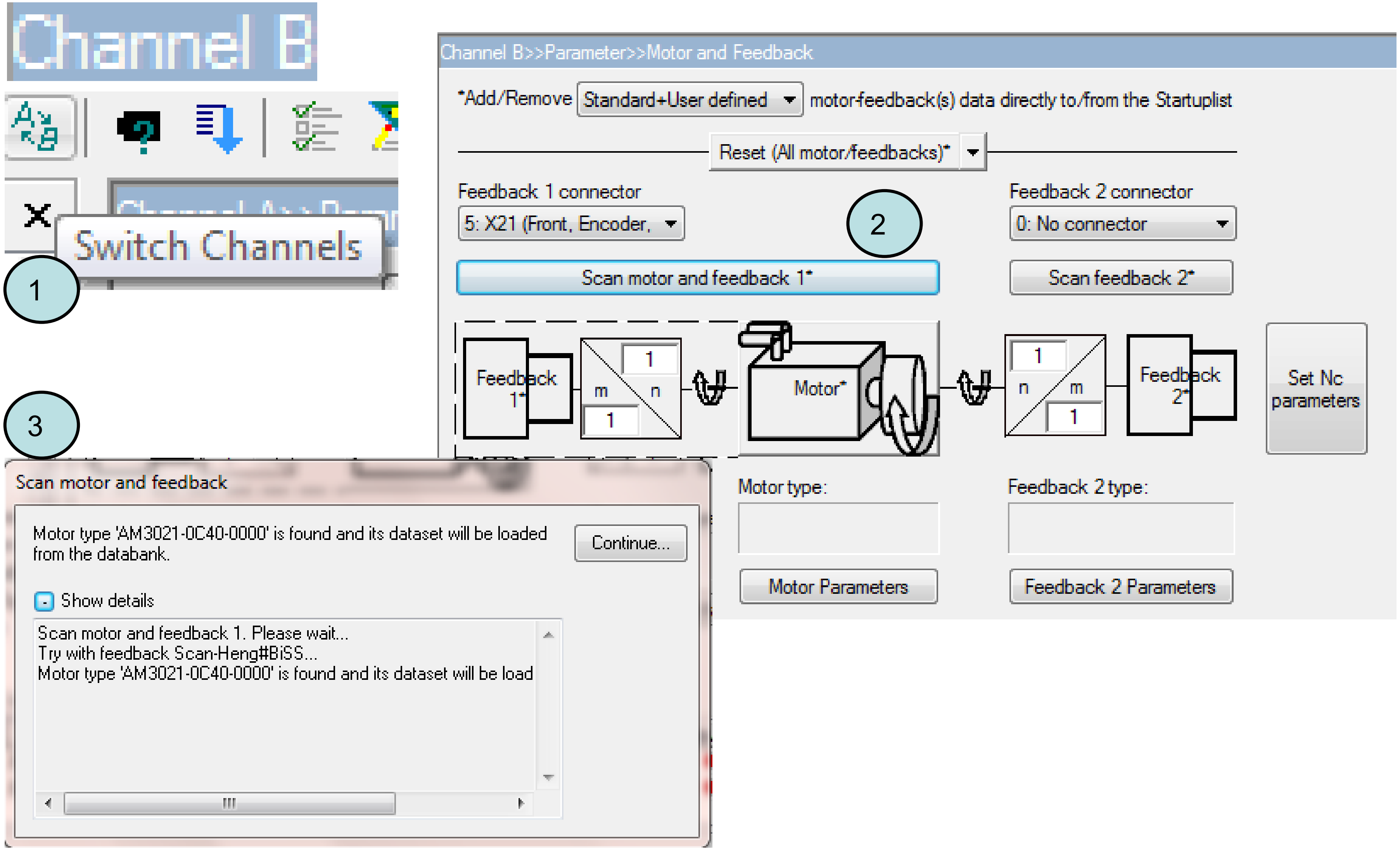
Limit nmax



NC Settings Demorack

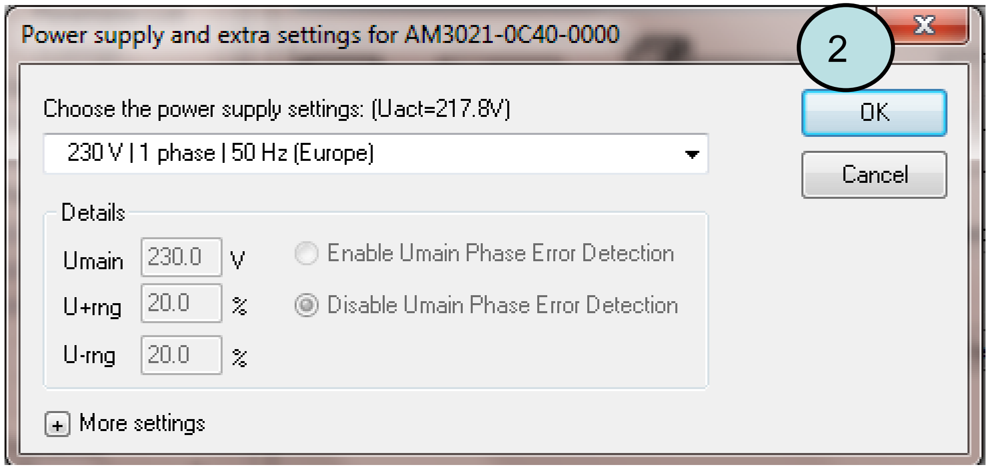


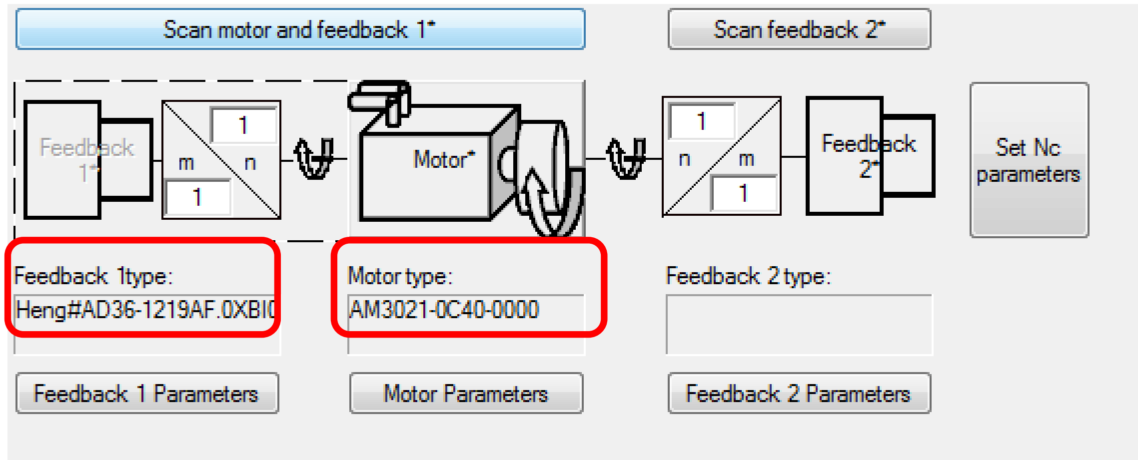
NC Settings Demorack



NC Settings Demorack

Channel B

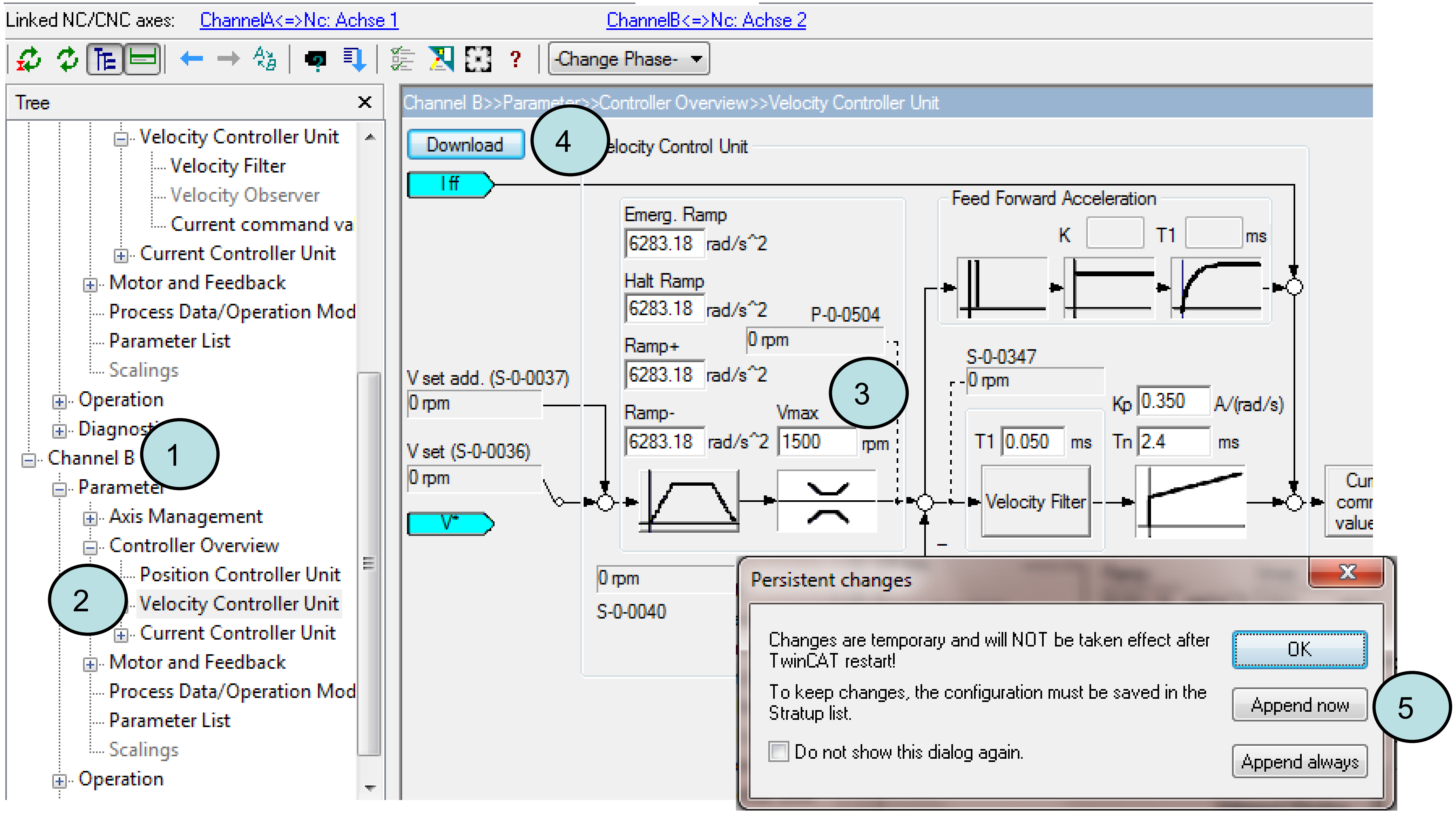




NC Settings Demorack

Channel B

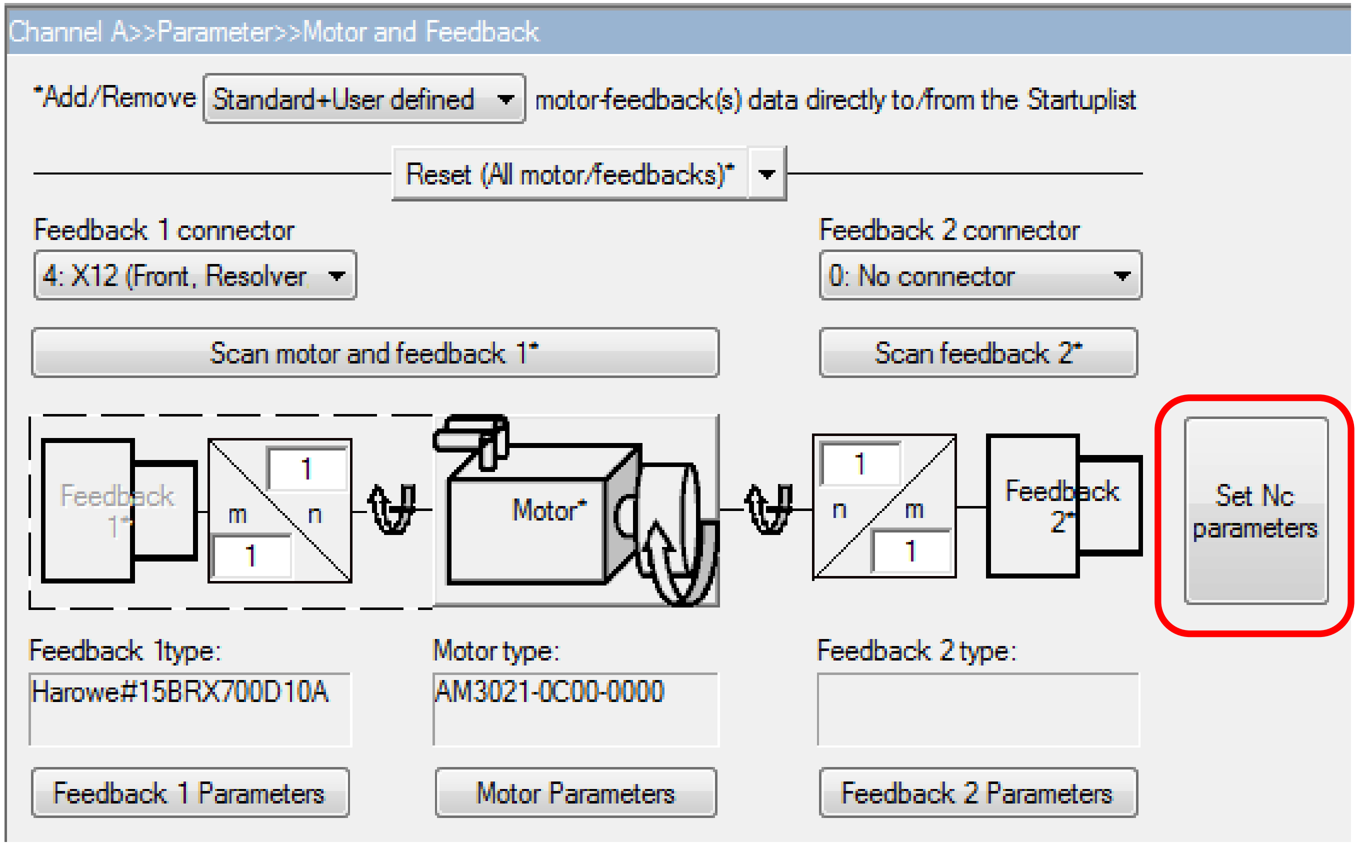
Limit nmax



New:
Auto calculation
of
NC basicparameter

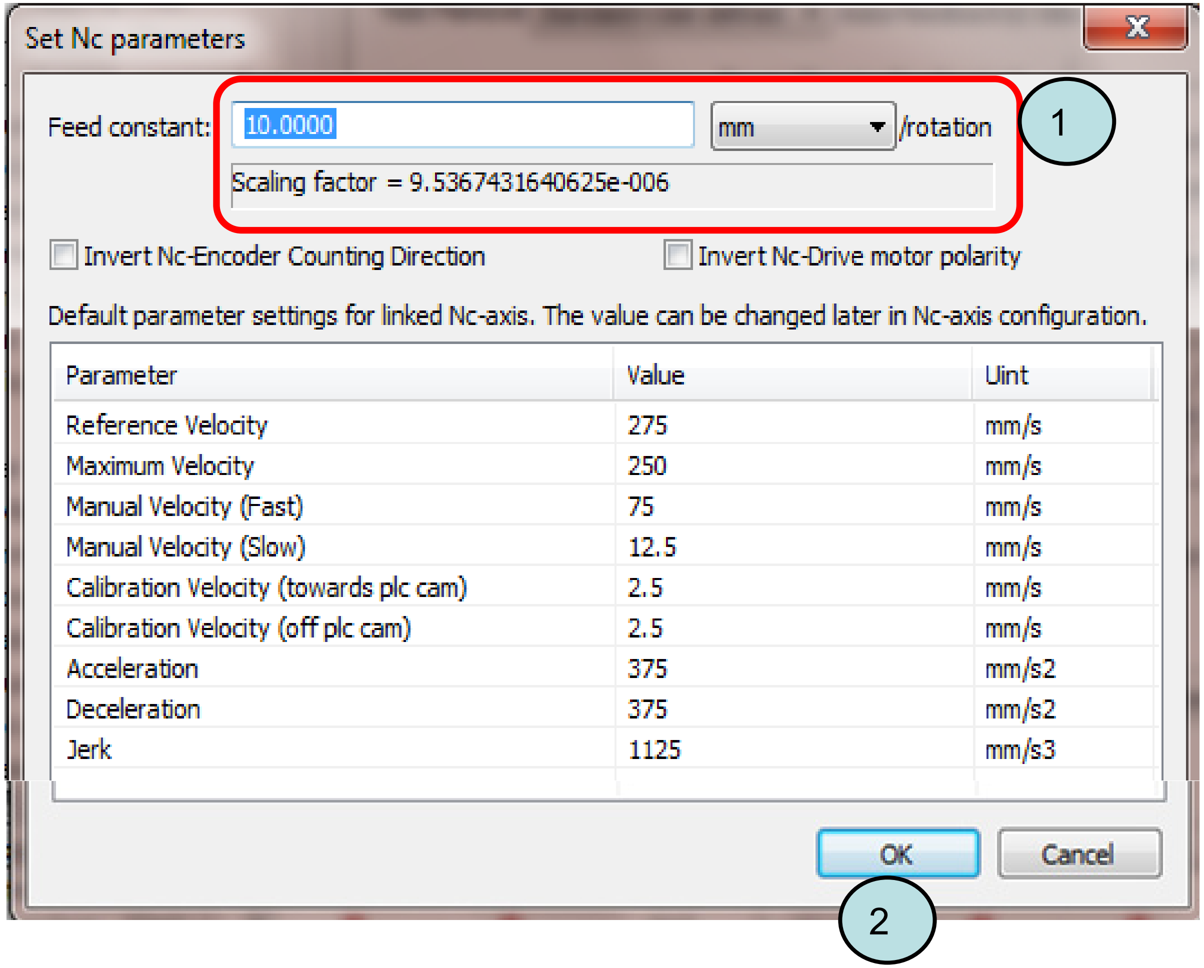
NC Settings Demorack

Channel A



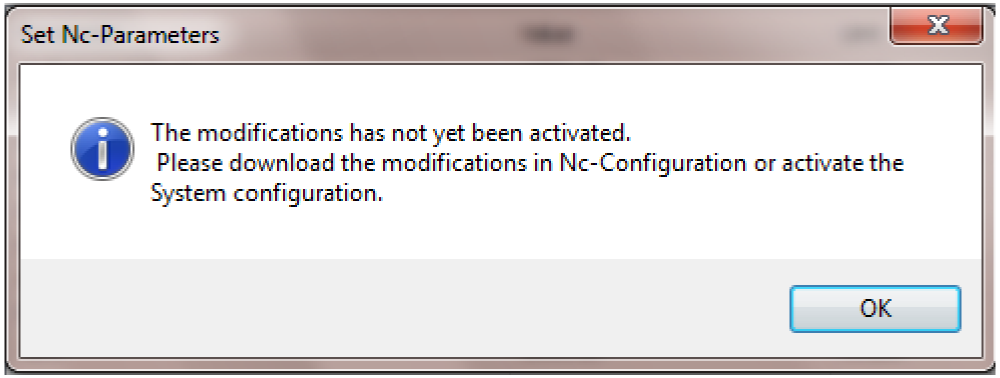
NC Settings Demorack

Channel A



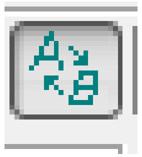
NC Settings Demorack

Channel A

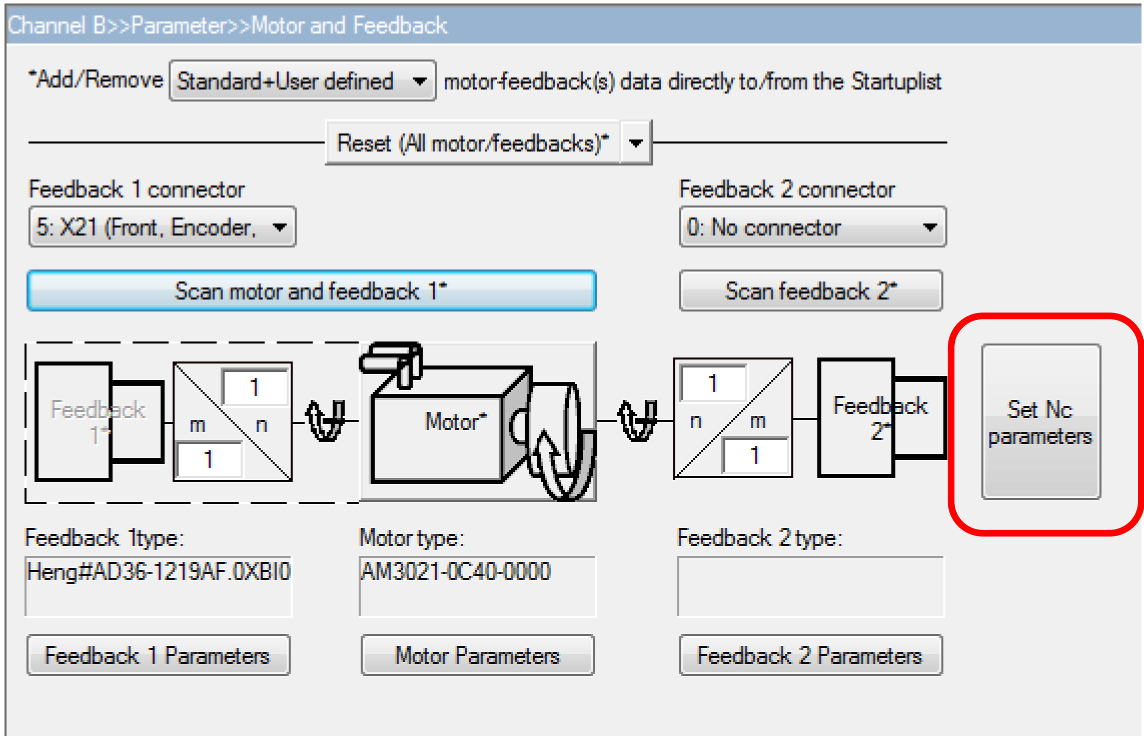


NC Settings Demorack

Channel B



1

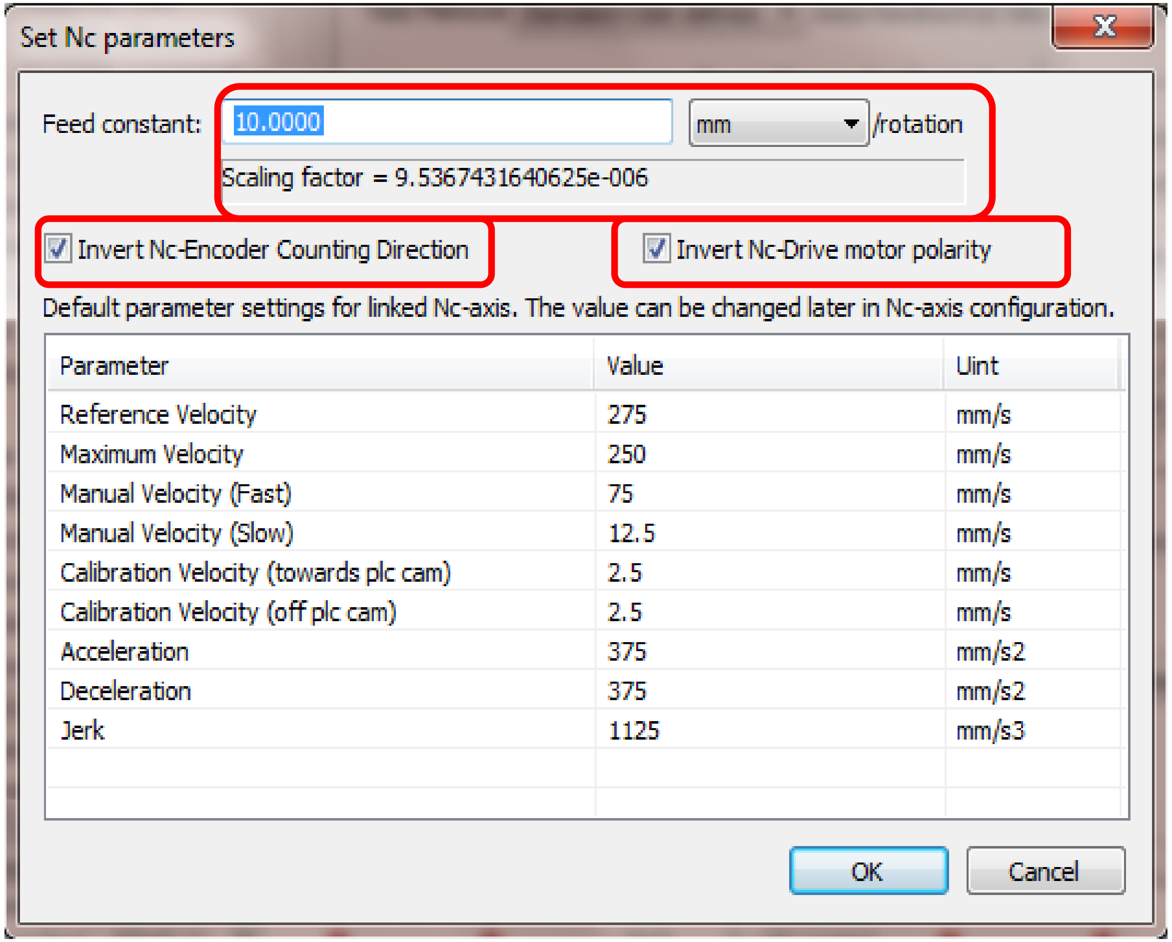


2

NC Settings Demorack

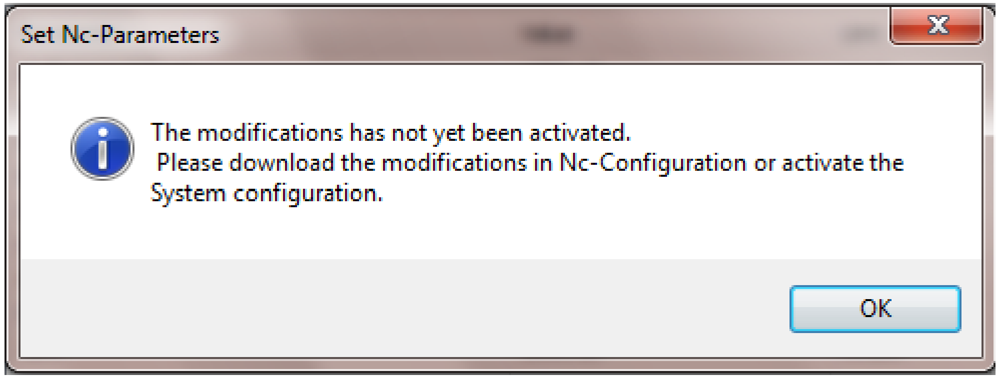
Channel B

Channel B
invers



NC Settings Demorack

Channel B

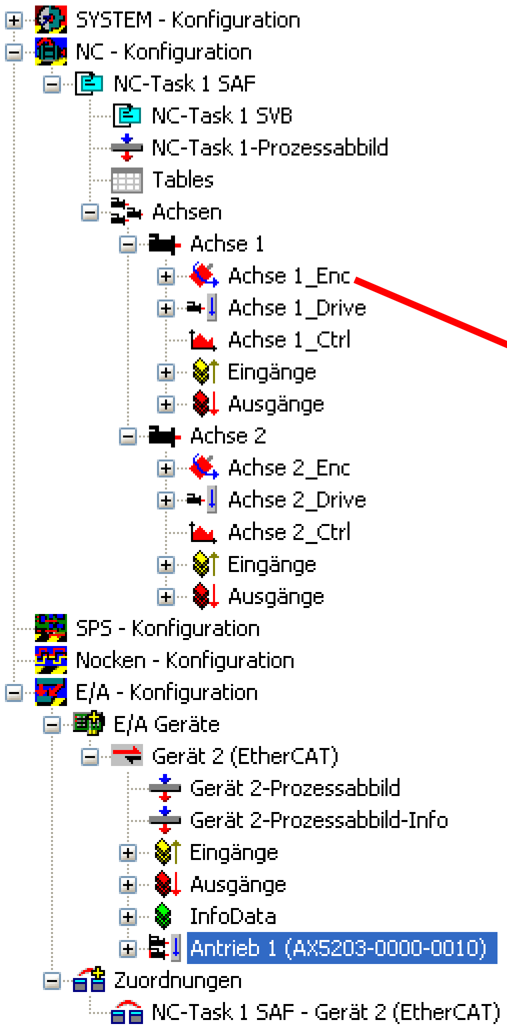


NC Settings Democrack

NC

Parameter

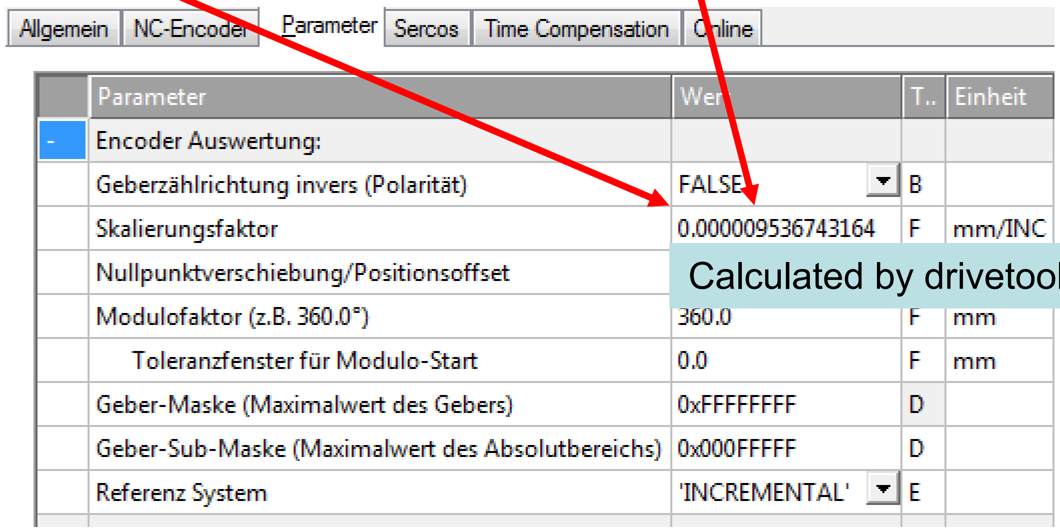
NC Settings Demorack



- System Manager contains hardware configuration and NC axes

Scaling factor: adjustment position feedback (Encoder increments / path)

$$Scalierungsfaktor = \frac{10mm}{1048576\,INC} = 9{,}5367431640625e\text{-}6 \text{ mm/INC}$$

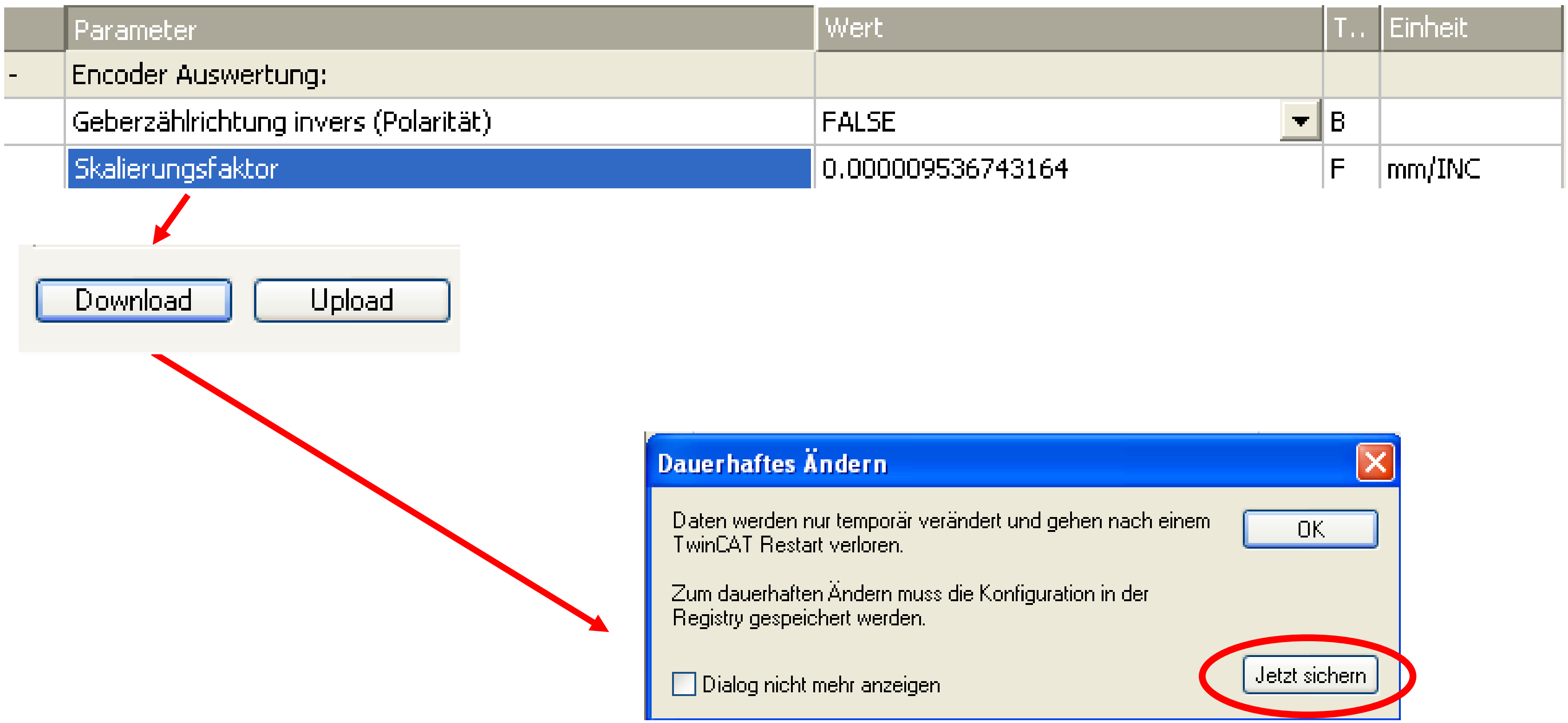


Parameter	Wert	T..	Einheit
Encoder Auswertung:			
Geberzählrichtung invers (Polarität)	FALSE	B	
Skalierungsfaktor	0.000009536743164	F	mm/INC
Nullpunktverschiebung/Positionsoffset			
Modulofaktor (z.B. 360.0°)	360.0	F	mm
Toleranzfenster für Modulo-Start	0.0	F	mm
Geber-Maske (Maximalwert des Gebers)	0xFFFFFFFF	D	
Geber-Sub-Maske (Maximalwert des Absolutbereichs)	0x000FFFFF	D	
Referenz System	'INCREMENTAL'	E	

Calculated by drivetool

NC Settings Demorack

- Parameter können bei Download zur NC in die aktuelle Konfiguration gesichert werden



NC Settings Demorack

- Paramameter können bei Download zur NC in die aktuelle Konfiguration gesichert werden

-	Endschalter:			
	Software Endlagenüberwachung Minimum	TRUE	B	
	Software Endlage Minimum	10.0	F	mm
	Software Endlagenüberwachung Maximum	TRUE	B	
	Software Endlage Maximum	6000.0	F	mm
+	Filter:			
-	Referenzfahrt:			
	Suchrichtung für Referenziernocken invers	FALSE	B	
	Suchrichtung für Syncimpuls invers	TRUE	B	
	Referenzposition für Referenzierfahrt ("Eichposition")	5900.0	F	mm
	Referenz Modus	'Default'	E	

'Default'
'Plc CAM'
'Hardware Sync'
'Hardware Latch Pos'
'Hardware Latch Neg'
'Software Sync'

Grenzen des Fahrweges. Überwachung durch NC

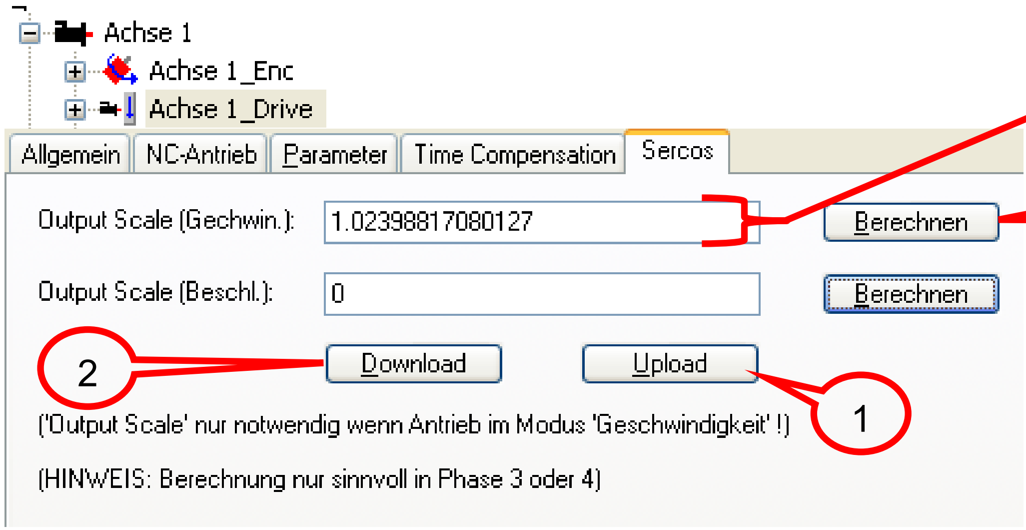
Richtungen Eichvorgang.
Bsp:
Referenznocken suchen (+)
Referenznocken verlassen (-)

Referenzposition

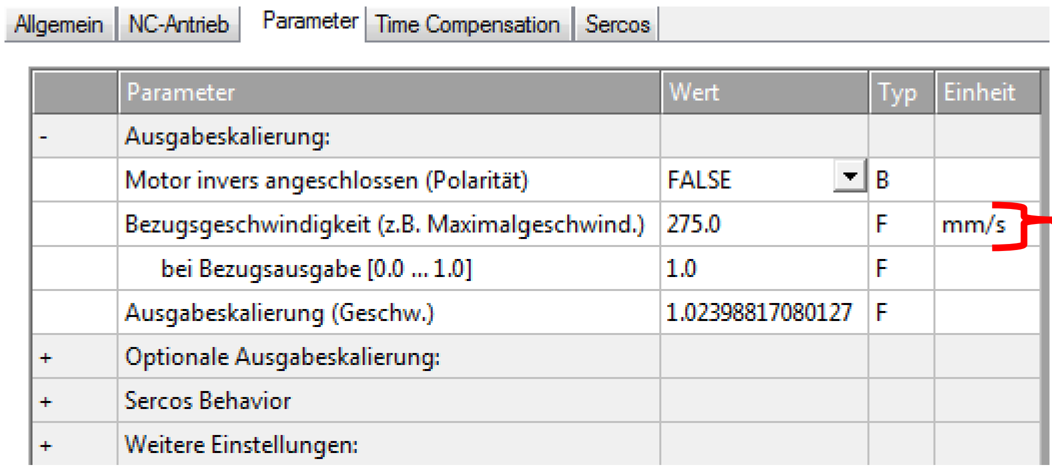
Nulldurchgang im Antrieb auswerten

NC Settings Demorack

- Ausgabescalierung prüfen lassen (online, Betriebsart Geschwindigkeitsausgabe



Output scaling



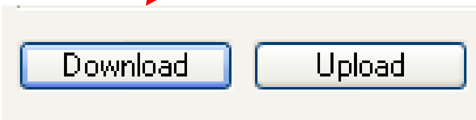
	Parameter	Wert	Typ	Einheit
-	Ausgabeskalierung:			
	Motor invers angeschlossen (Polarität)	FALSE	B	
	Bezugsgeschwindigkeit (z.B. Maximalgeschwind.)	275.0	F	mm/s
	bei Bezugsausgabe [0.0 ... 1.0]	1.0	F	
	Ausgabeskalierung (Geschw.)	1.02398817080127	F	
+	Optionale Ausgabeskalierung:			
+	Sercos Behavior			
+	Weitere Einstellungen:			

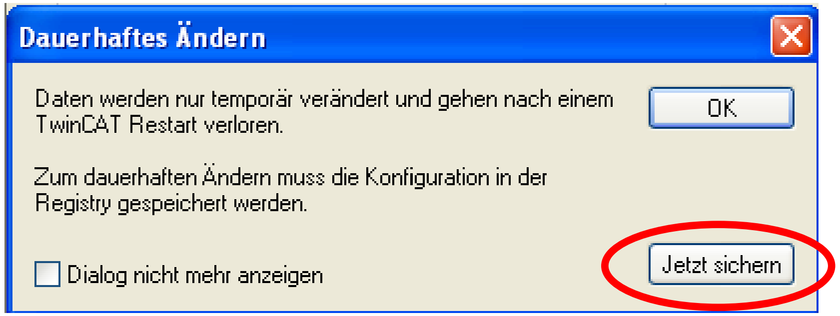
Reference velocity:
Output scaling for further drive types: analog, stepper motor and further.
Here: max. possible velocity.
Final limit velocity.

NC settings demo-rack

- Parameter can be stored by download to NC in the active configuration.

	Parameter	Wert	T..	Einheit
-	Encoder Auswertung:			
	Geberzählrichtung invers (Polarität)	FALSE	B	
	Skalierungsfaktor	0.000009536743164	F	mm/INC





NC settings demo-rack

- Parameter can be stored by download to NC in the active configuration.

-	Endschalter:			
	Software Endlagenüberwachung Minimum	TRUE	B	
	Software Endlage Minimum	10.0	F	mm
	Software Endlagenüberwachung Maximum	TRUE	B	
	Software Endlage Maximum	6000.0	F	mm
+	Filter:			
-	Referenzfahrt:			
	Suchrichtung für Referenziernocken invers	FALSE	B	
	Suchrichtung für Syncimpuls invers	TRUE	B	
	Referenzposition für Referenzierfahrt ("Eichposition")	5900.0	F	mm
	Referenz Modus	'Default'	E	

'Default'
'Plc CAM'
'Hardware Sync'
'Hardware Latch Pos'
'Hardware Latch Neg'
'Software Sync'

Limits of path.
Monitoring by NC

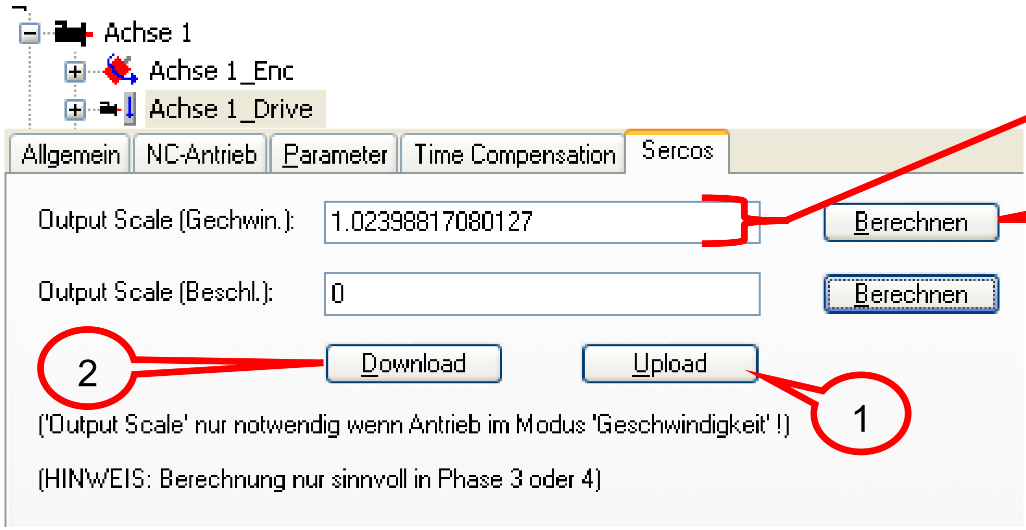
Direction referencing.
e.g.:
Search reference cam (+)
Leave reference cam (-)

Reference position

Analyse zero point in the drive

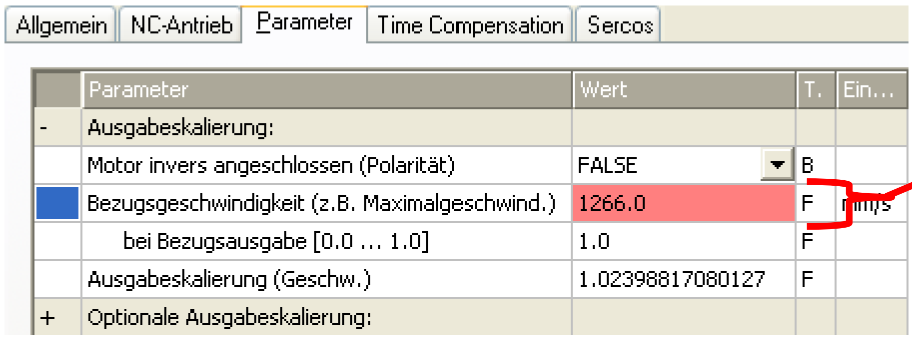
NC settings demo-rack

- Output values to be checked (online, operating mode velocity output)



Output scaling

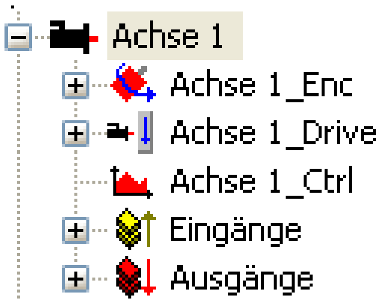
$$v_{bezug} = \frac{n\,max}{60s} * \frac{Weg}{Umdrehung} = \frac{7599U}{60s} * \frac{10mm}{U} = \frac{1266,5\ mm}{s}$$

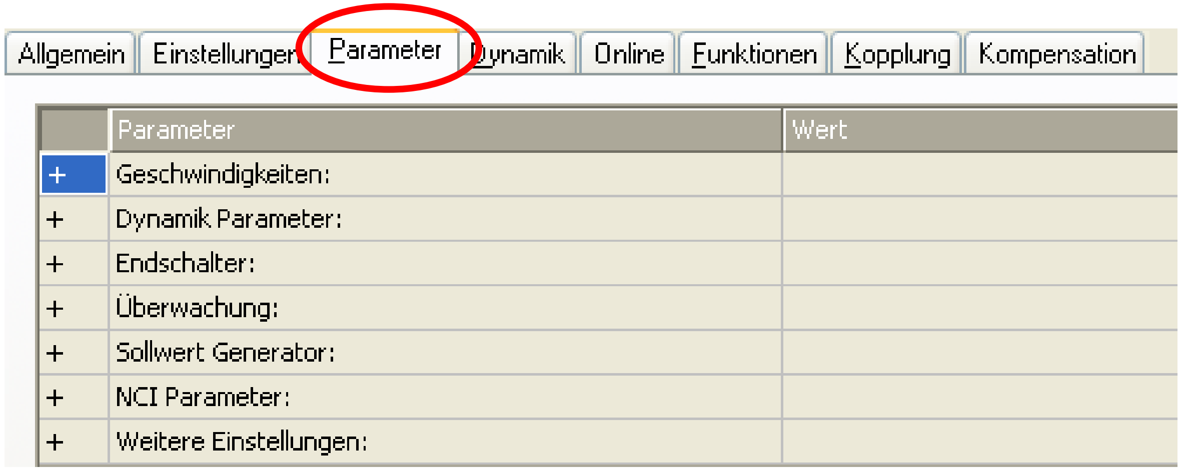


Reference velocity:
Output scaling for further drive types:
analog, stepper motor and further.
Here: max. possible velocity.
Final limit velocity.

NC settings demo-rack

- Axis, parameter





NC settings demo-rack

- Axis, parameter

	Parameter	Wert	T.	Einheit
-	Geschwindigkeiten:			
	Bezugsgeschwindigkeit (z.B. Maximalgeschwind.)	1266.0	F	mm/s
	Maximale erlaubte Geschwindigkeit	250.0	F	mm/s
	Geschwindigkeit Hand Max (Fast)	300.0	F	mm/s
	Geschwindigkeit Hand Min (Slow)	100.0	F	mm/s
	Geschwind. Ref.fahrt in pos. Richtung	30.0	F	mm/s
	Geschwind. Ref.fahrt in neg. Richtung	30.0	F	mm/s
	Pulsweite in positiver Richtung (Jog-Betrieb)	5.0	F	mm
	Pulsweite in negativer Richtung (Jog-Betrieb)	5.0	F	mm
+	Dynamik Parameter:			
+	Endschalter:			
+	Überwachung:			
+	Sollwert Generator:			
+	NCI Parameter:			
+	Weitere Einstellungen:			

Allgemein | NC-Encoder | Parameter | Sercos | Time Compensation | Online

-	Referenzfahrt:			
	Suchrichtung für Referenziernocken invers	FALSE	B	
	Suchrichtung für Syncimpuls invers	TRUE	B	
	erenzposition für Referenzierfahrt ("Eichposition")	5900.0	F	mm
	erenz Modus	'Default'	E	

Achse 1
Achse 1_Enc

See „Drive“

Limitation of the max. allowed velocity (Contoller reserve)

Hand velocities

-- F1 ++ F4

Hand velocities

- F2 + F3

Referencing velocity positive. Here „search cams“

Referencing velocity negative Here :„Leave cams“

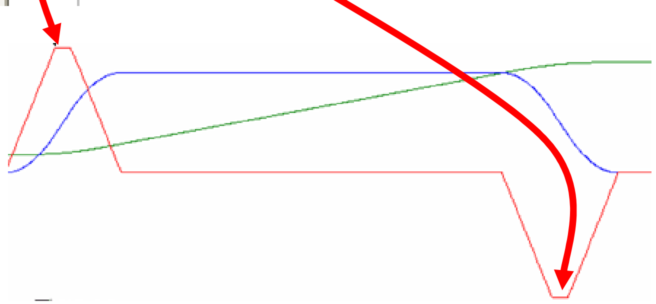
CAN BE CHANGED!!

NC settings demo-rack

- Axis, parameter

Allgemein | Einstellungen | Parameter | Dynamik | Online | Funktionen | Kopplung

	Parameter	Wert	T.	Einheit
-	Geschwindigkeiten:			
	Bezugsgeschwindigkeit (z.B. Maximalgeschwind.)	1266.0	F	mm/s
	Maximale erlaubte Geschwindigkeit	250.0	F	mm/s
	Geschwindigkeit Hand Max (Fast)	300.0	F	mm/s
	Geschwindigkeit Hand Min (Slow)	100.0	F	mm/s
	Geschwind. Ref.fahrt in pos. Richtung	30.0	F	mm/s
	Geschwind. Ref.fahrt in neg. Richtung	30.0	F	mm/s
	Pulsweite in positiver Richtung (Jog-Betrieb)	5.0	F	mm
	Pulsweite in negativer Richtung (Jog-Betrieb)	5.0	F	mm
-	Dynamik Parameter:			
	Beschleunigung	1500.0	F	mm/s2
	Verzögerung	1500.0	F	mm/s2
	Ruck	2250.0	F	mm/s3



Travel path pos. for MC_JOGG

Travel path neg. for MC_JOGG

Maximum occuring acceleration within 7 phase profile.

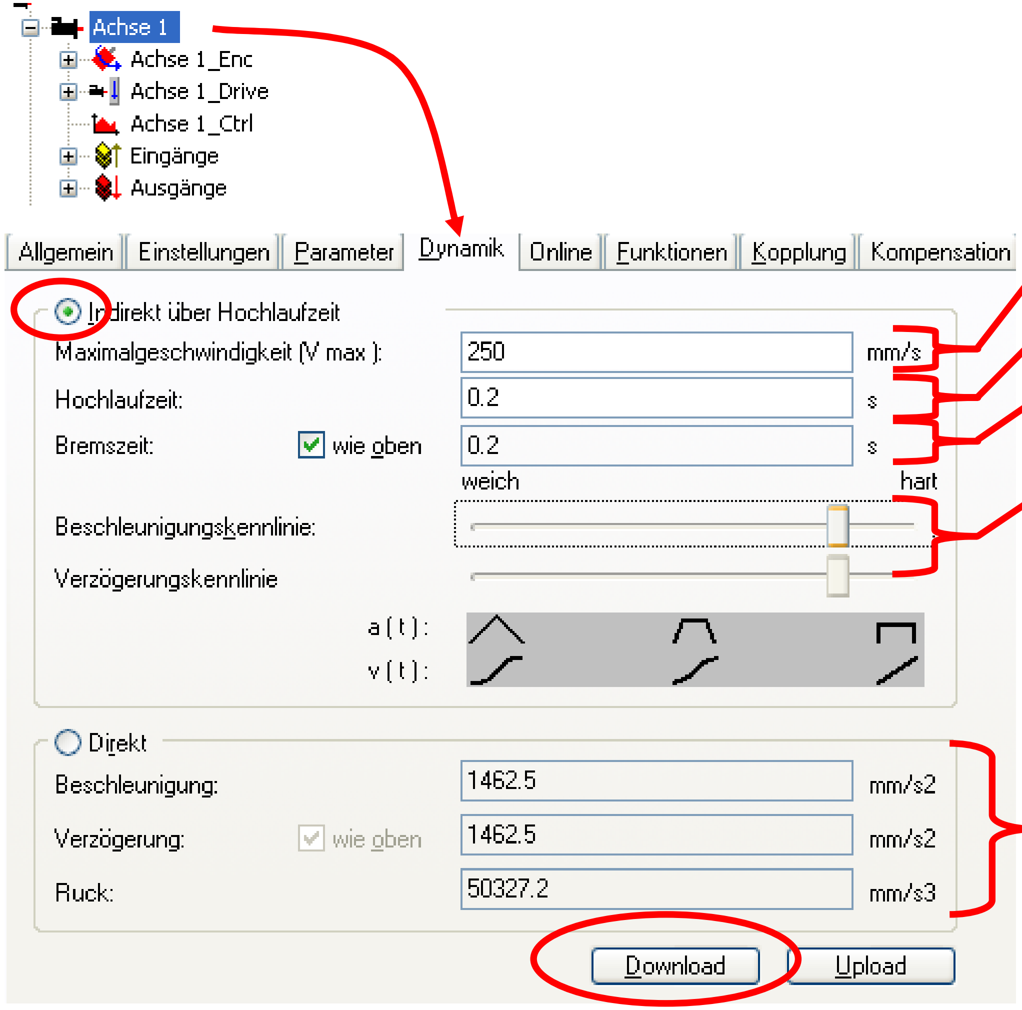
Maximum occuring decceleration within 7 phase profile.

Maximum occuring acceleration change (jerk)

Settings can be done via „run-up time“ and acceleration/deceleration characteristic curves.

NC settings demo-rack

- Achse , dynamics



Maximum velocity from „parameter“

Specify Run-up time to Vmax

Deceleration time

Acceleration profile:

Hard : small acceleration but high acceleration change. (jerk)

Smooth : higher acceleration but samller acceleration change (jerk)

Calculated maximum acceleration /deceleration values

NC settings demo-rack

- Axis in online mode

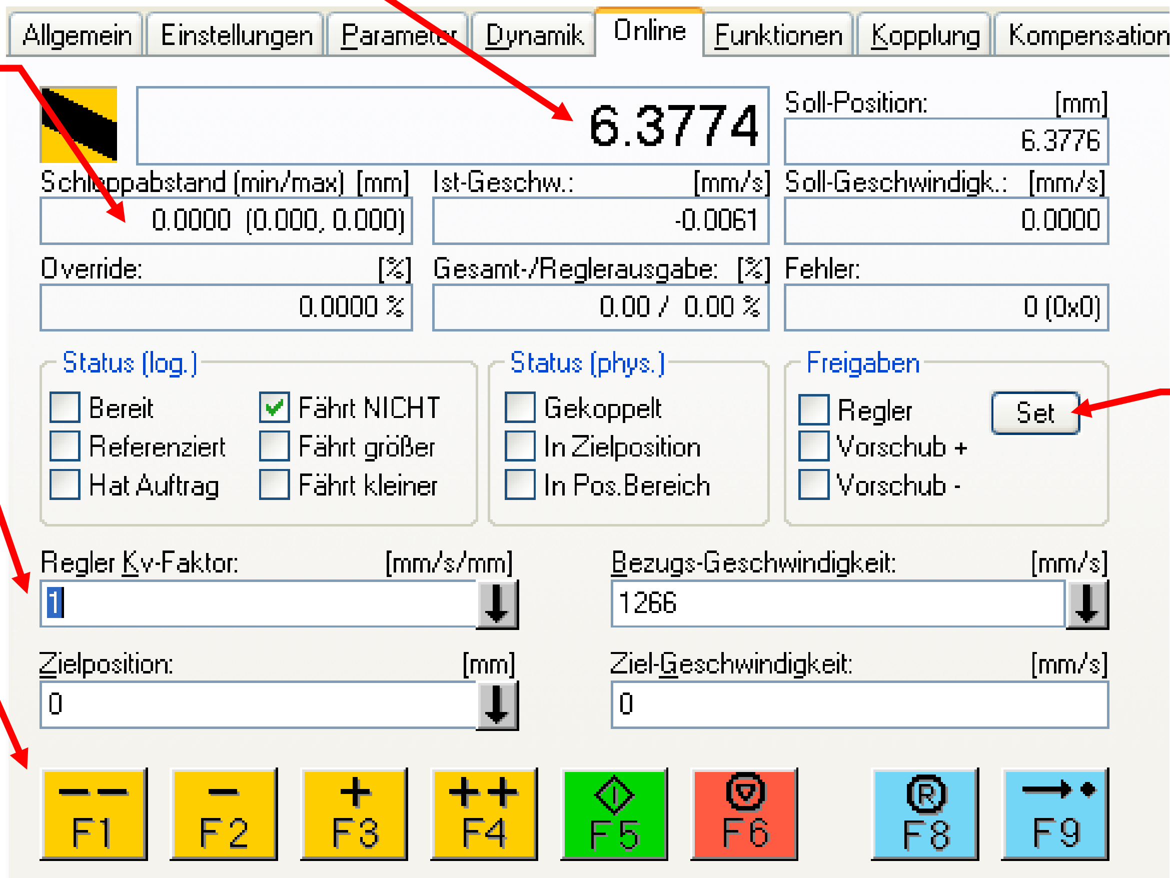
Actual position

Position deviation

Position-controller Kv factor

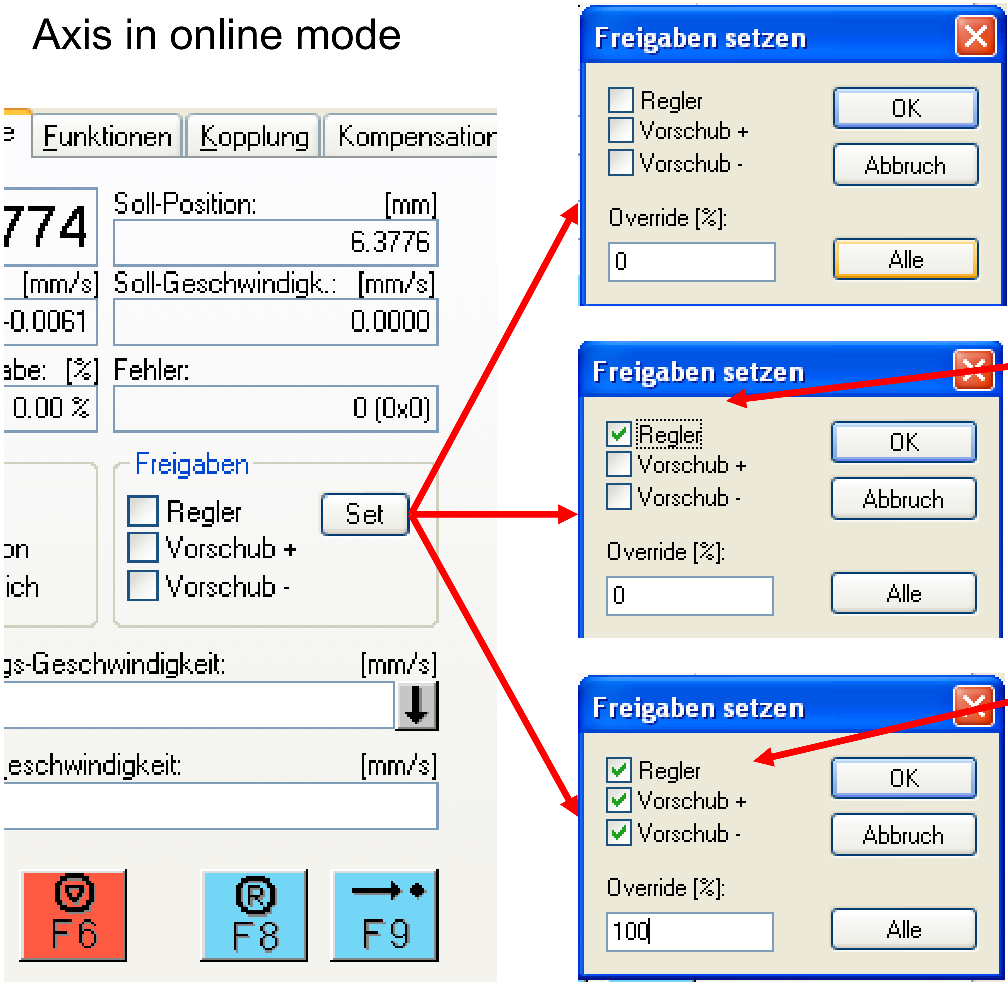
Move axis in incremental jog

Controller enable



NC settings demo-rack

- Axis in online mode



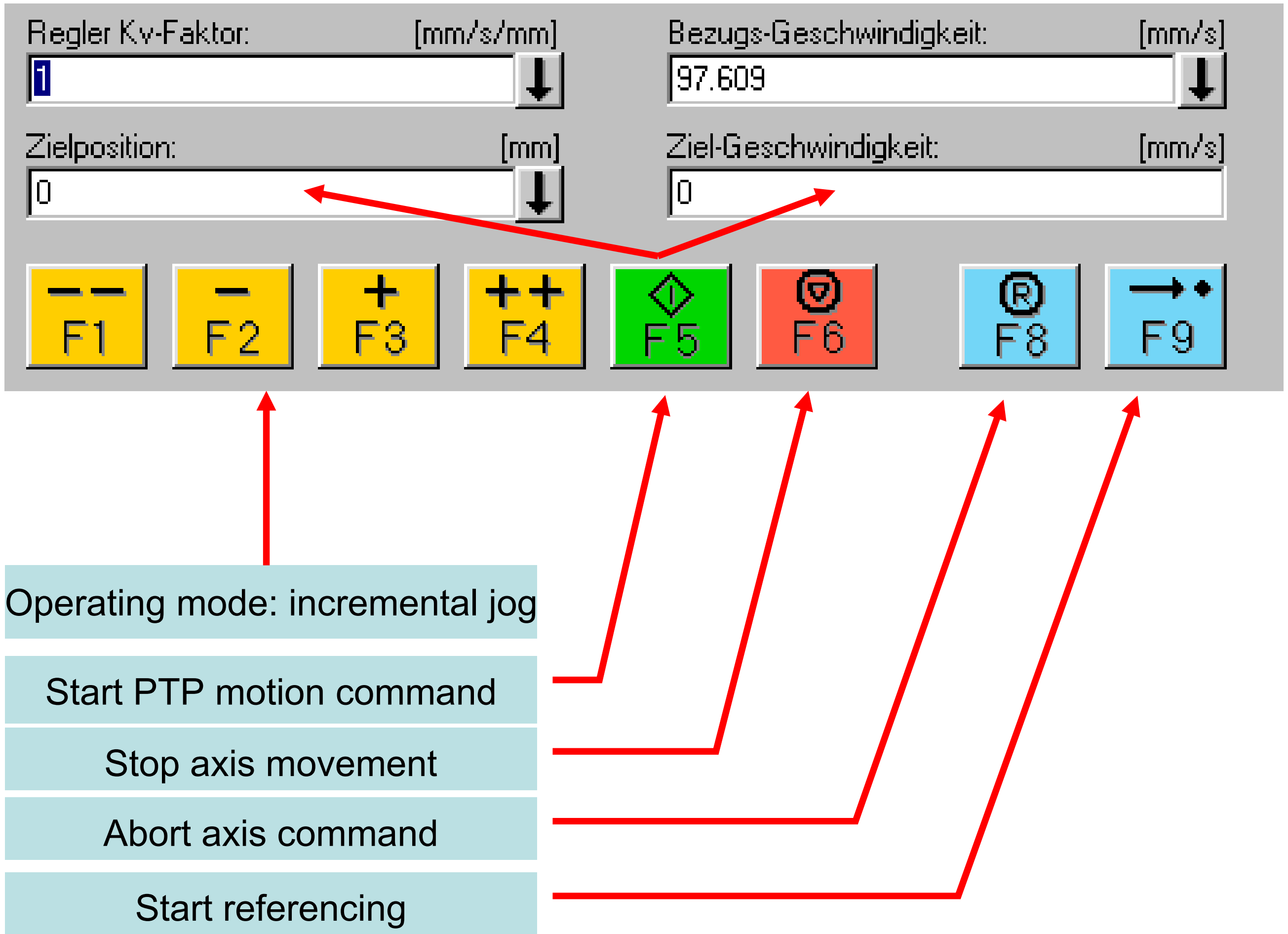
No enabling

Position control, motion locked

Position control + motion enabled. WARINING: consider override!

NC settings demo-rack

- Axis in online mode



NC settings demo-rack with KL2531

Data stepper motor and KL2531 (test-rack):

KL2531 operation mode : velocity direct without ramps

Stepper motor: 1 full step 1.8 degree

KL2531 switched to 64-fold microstepping

Supposed mechanical transformation : **1 rotation corresponds to 10 mm path**

Maximum speed of stepper motor **585,6514** U/min:

Note: The test rack works without encoder. The position feedback is delivered from input „Position“ of KL2531.

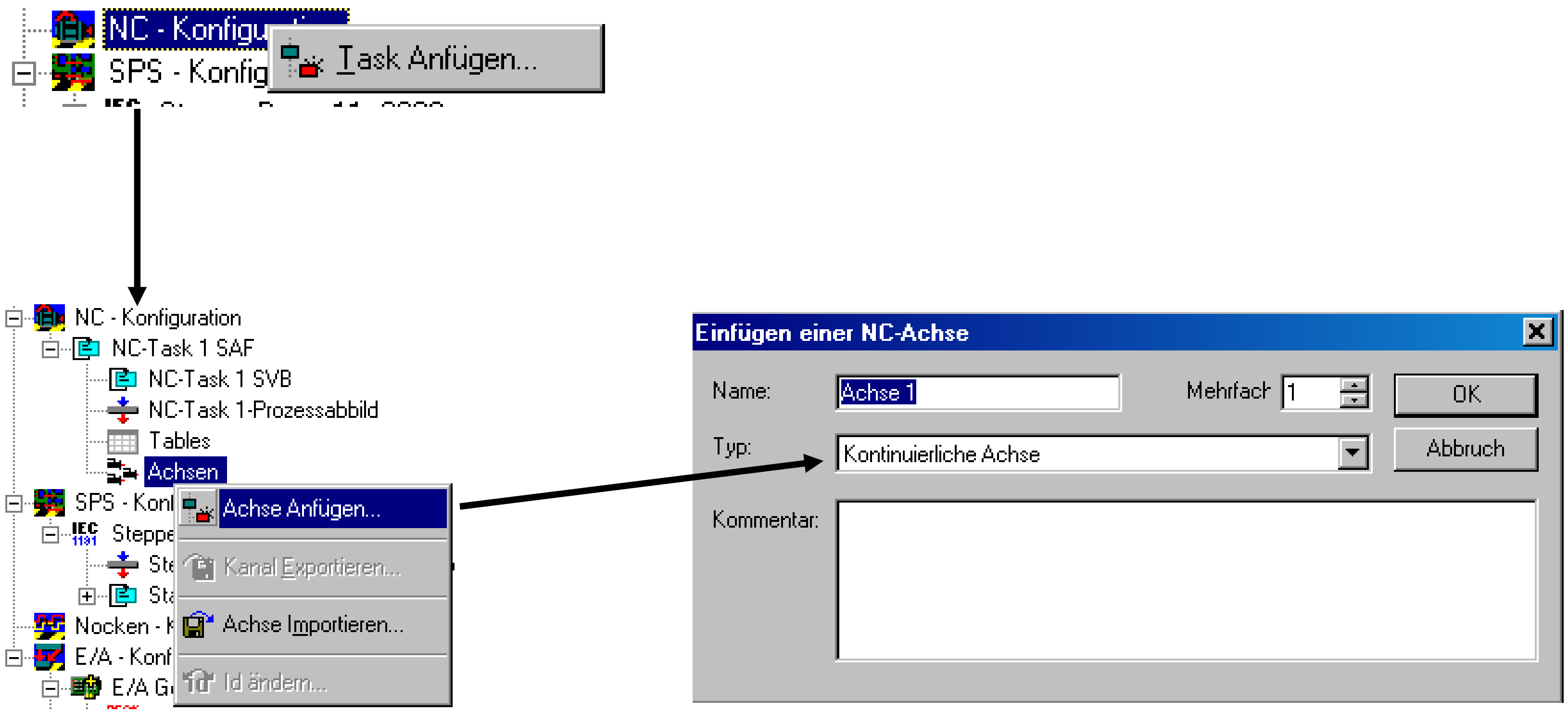
$$Incremente = \frac{360°}{Vollschritte} * Microstepping = \frac{360}{1,8} * 64 = 12800\,INC$$

$$Scalierungsfaktor = \frac{mechWeg\,/\,Umdrehung}{AnzahlIncremente} = \frac{10\frac{mm}{U}}{12800\frac{Inc}{U}} = 0,00078125\,\frac{\mathrm{mm}}{\mathrm{Inc}}$$

$$Bezugsgeschwindigkeit = MaxDrehzahl * \frac{mech.Weg}{Umdrehung} = 585,6514\,\frac{\mathrm{U}}{60\mathrm{s}} * 10\frac{mm}{U} = 97,609\,\frac{\mathrm{mm}}{\mathrm{s}}$$

NC settings demo-rack with KL2531

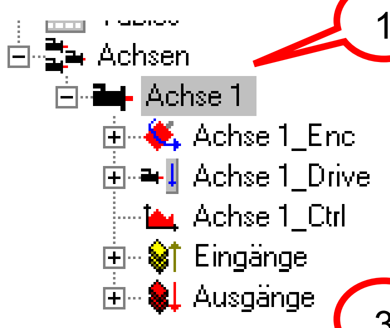
- Append axis manually



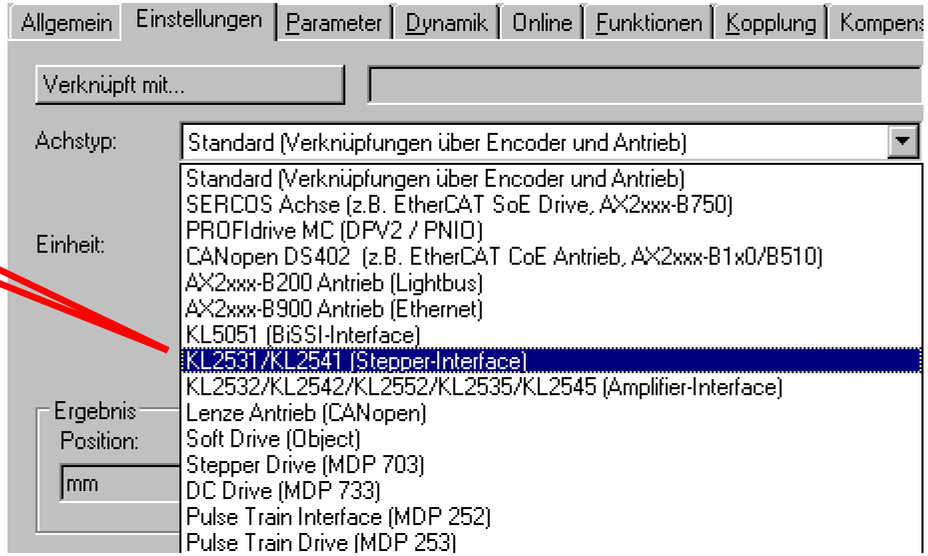
NC settings demo-rack with KL2531

- Link axis

1



2

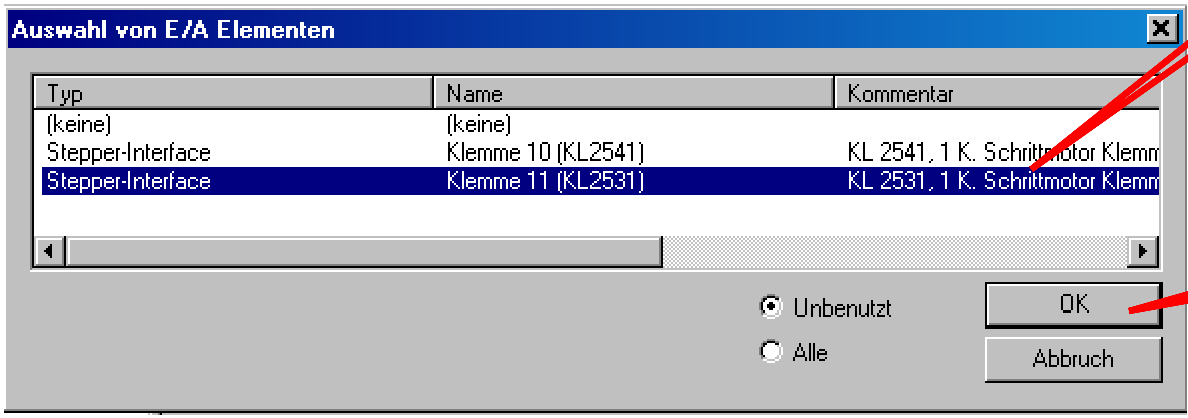


3

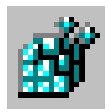
4

Don‘t forget „Activate Configuration if online mode is used

Axis parameter can be changed at first offline

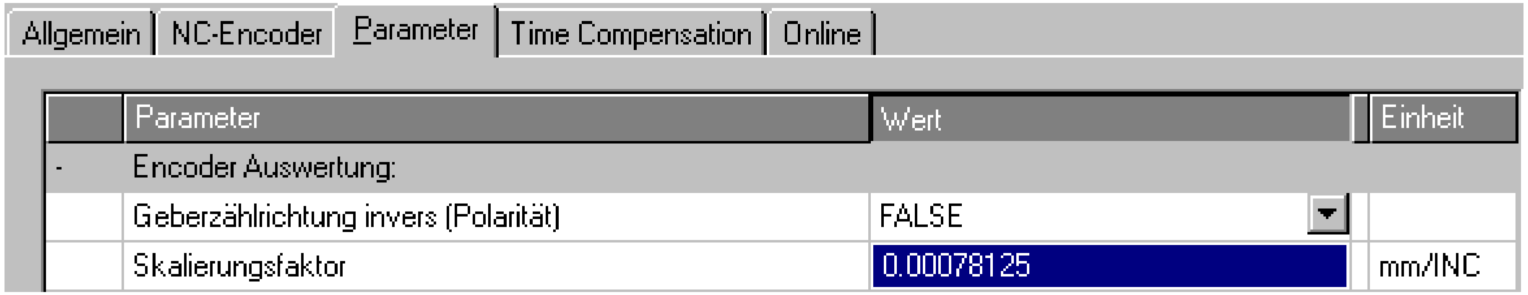


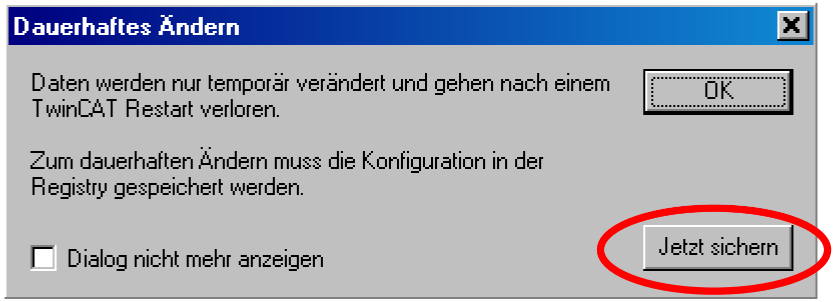
5



NC settings demo-rack with KL2531

- Scaling factor :





NC settings demo-rack with KL2531

- Scaling factor:

-	Endschalter:			
	Software Endlagenüberwachung Minimum	TRUE	B	
	Software Endlage Minimum	10.0	F	mm
	Software Endlagenüberwachung Maximum	TRUE	B	
	Software Endlage Maximum	1000.0	F	mm
+	Filter:			
-	Referenzfahrt:			
	Suchrichtung für Referenziernocken invers	FALSE	B	
	Suchrichtung für Syncimpuls invers	TRUE	B	
	Referenzposition für Referenzierfahrt ("Eichposition")	1010.0	F	mm
	Referenz Modus	'Default'	E	
+	Weitere Einstellungen:			

'Default'
'Plc CAM'
'Hardware Sync'
'Hardware Latch Pos'
'Hardware Latch Neg'
'Software Sync'

5900.0

'Default'

'Default'
'Plc CAM'
'Hardware Sync'
'Hardware Latch Pos'
'Hardware Latch Neg'
'Software Sync'

Limits of travel path.
Monitoring by NC

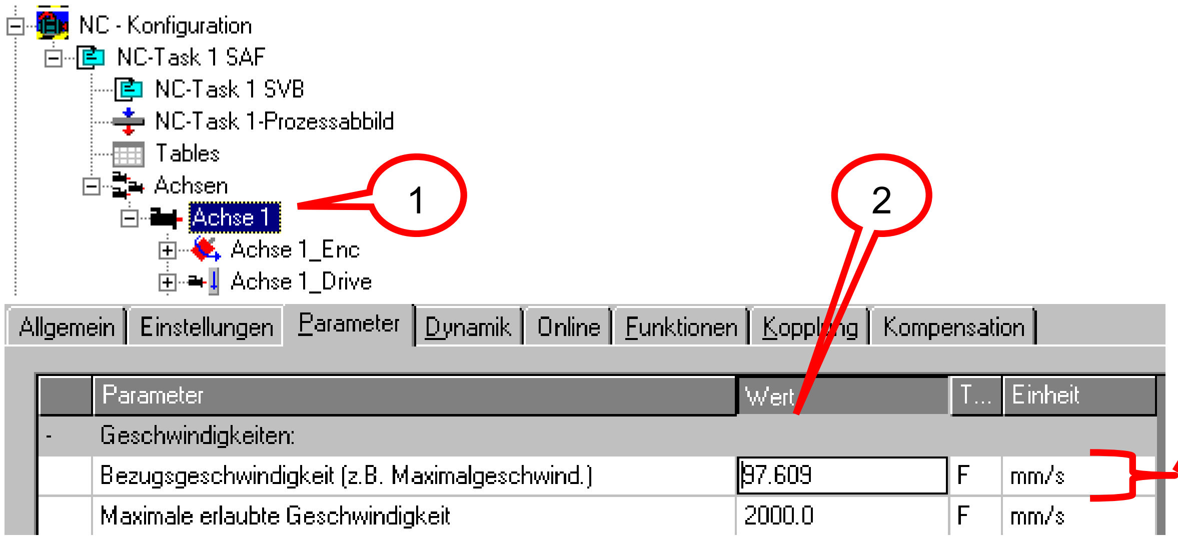
Directions referencing.
e.g.:
Search reference cams (+)
Leave reference cams (-)

Reference position

No zero signal wothout encoder:
Reference signal is 1-0 edge
of reference switch

NC settings demo-rack with KL2531

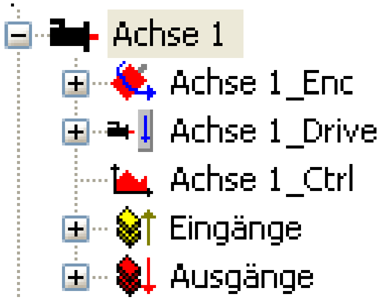
- Calculate output scaling (formula see introduction)

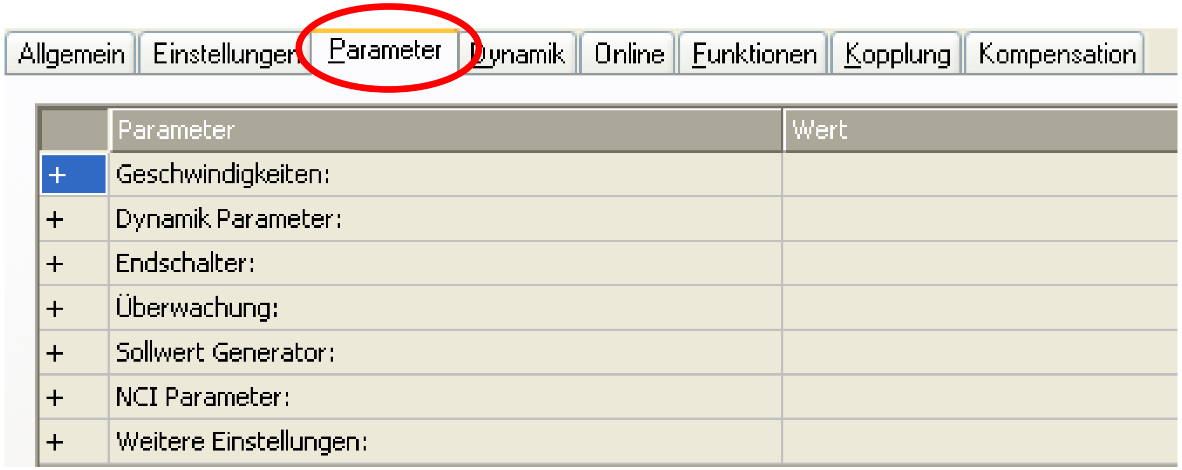


Reference velocity

NC settings demo-rack with KL2531

- Axis, parameter





NC settings demo-rack with KL2531

- Axis, parameter

	Parameter	Wert	T...	Einheit
-	Geschwindigkeiten:			
	Bezugsgeschwindigkeit (z.B. Maximalgeschwind.)	97.609	F	mm/s
	Maximale erlaubte Geschwindigkeit	90.0	F	mm/s
	Geschwindigkeit Hand Max (Fast)	50.0	F	mm/s
	Geschwindigkeit Hand Min (Slow)	20.0	F	mm/s
	Geschwind. Ref.fahrt in pos. Richtung	10.0	F	mm/s
	Geschwind. Ref.fahrt in neg. Richtung	5.0	F	mm/s
	Pulsweite in positiver Richtung (Jog-Betrieb)	5.0	F	mm
	Pulsweite in negativer Richtung (Jog-Betrieb)	5.0	F	mm
+	Dynamik Parameter:			

Allgemein | NC-Encoder | Parameter | Sercos | Time Compensation | Online

-	Referenzfahrt:			
	Suchrichtung für Referenziernocken invers	FALSE	B	
	Suchrichtung für Syncimpuls invers	TRUE	B	
	Referenzposition für Referenzierfahrt ("Eichposition")	1010.0	F	mm
	Referenz Modus	'Default'	E	

Achse 1
Achse 1_Enc

See „Drive“

Limitation of the max. allowed velocity (Contoller reserve)

Hand velocities

– – F1 | + + F4

Hand velocities

– F2 | + F3

Referencing velocity positive. Here „search cams“

Referencing velocity negative Here :„Leave cams“

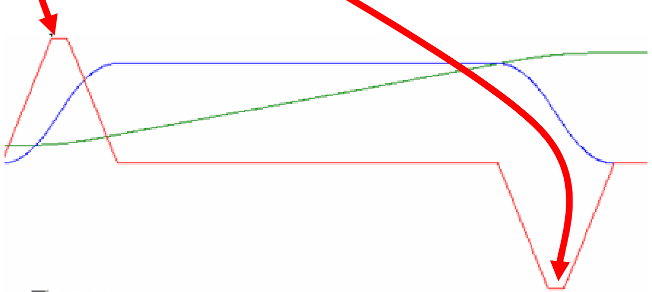
CAN BE CHANGED!!

NC settings demo-rack with KL2531

- Axis, parameter

Allgemein | Einstellungen | Parameter | Dynamik | Online | Funktionen | Kopplung | Kompensation

Parameter	Wert	T...	Einheit
- Geschwindigkeiten:			
Bezugsgeschwindigkeit (z.B. Maximalgeschwind.)	97.60899...	F	mm/s
Maximale erlaubte Geschwindigkeit 250.0	90.0	F	mm/s
Geschwindigkeit Hand Max (Fast)	50.0	F	mm/s
Geschwindigkeit Hand Min (Slow)	20.0	F	mm/s
Geschwind. Ref.fahrt in pos. Richtung	10.0	F	mm/s
Geschwind. Ref.fahrt in neg. Richtung	5.0	F	mm/s
Pulsweite in positiver Richtung (Jog-Betrieb)	5.0	F	mm
Pulsweite in negativer Richtung (Jog-Betrieb)	5.0	F	mm
- Dynamik Parameter:			
Beschleunigung	141.1930...	F	mm/s2
Verzögerung	141.1930...	F	mm/s2
Ruck	1582.170...	F	mm/s3



Travel path pos. for MC_JOGG

Travel path neg. for MC_JOGG

Maximum occuring acceleration within 7 phase profile.

Maximum occuring decceleration within 7 phase profile.

Maximum occuring acceleration change (jerk)

Settings can be done via „run-up time“ and acceleration/deceleration characteristic curves.

NC settings demo-rack with KL2531

- Axis in online mode

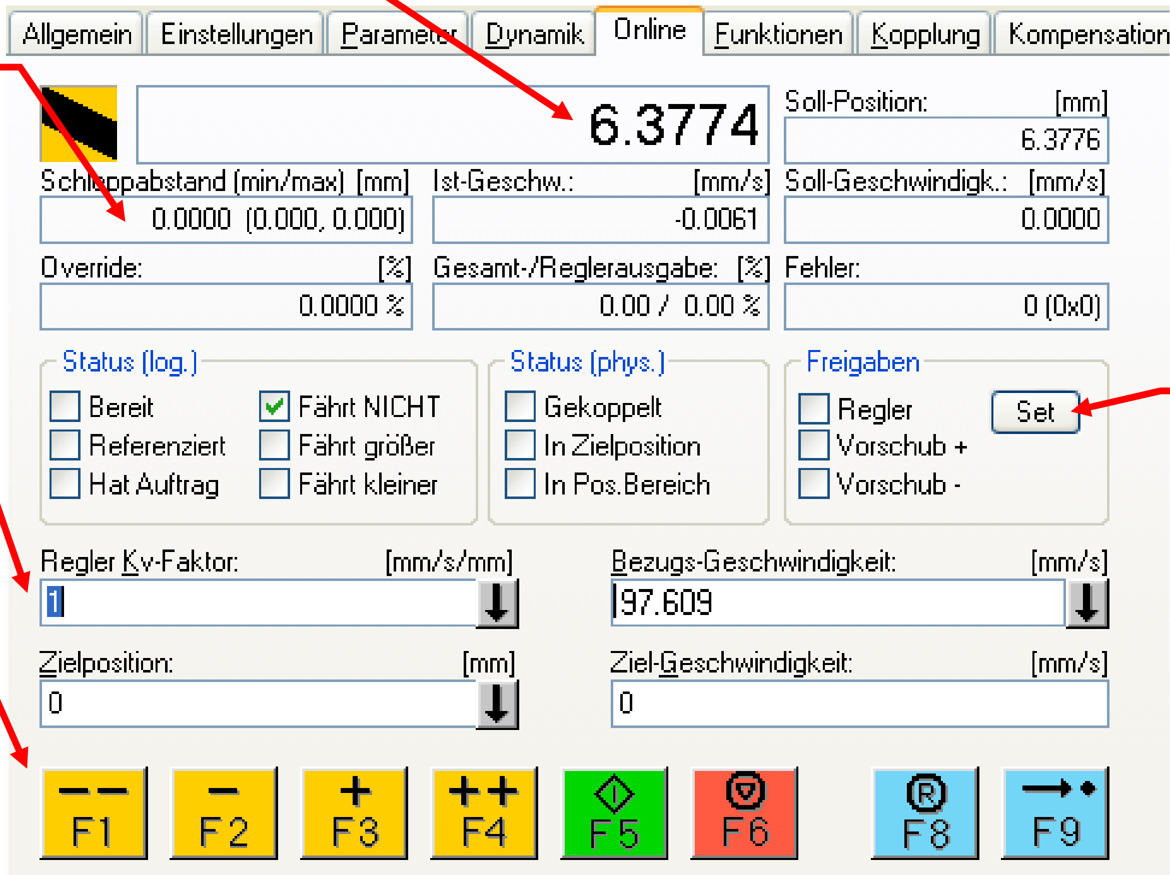
Actual position

Position deviation

Position-controller Kv factor

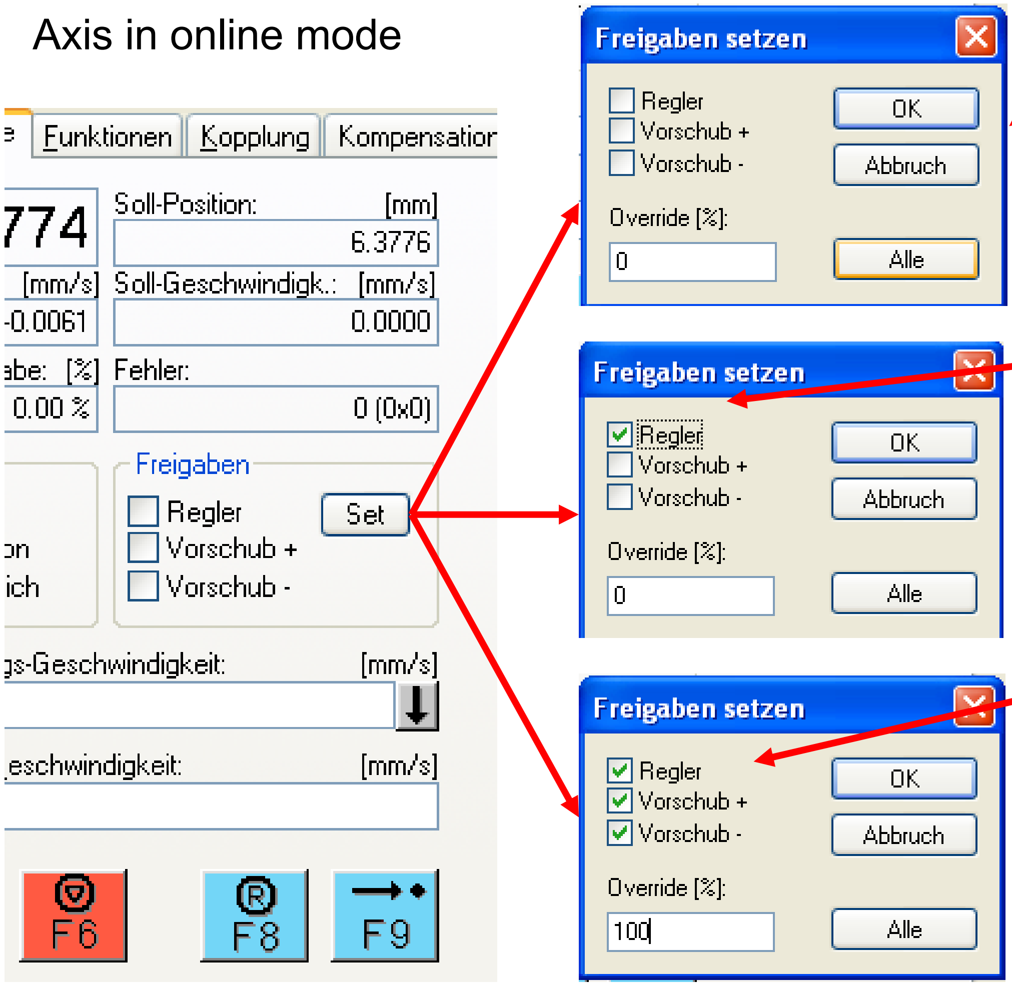
Move axis in incremental jog

Controller enable



NC settings demo-rack with KL2531

- Axis in online mode



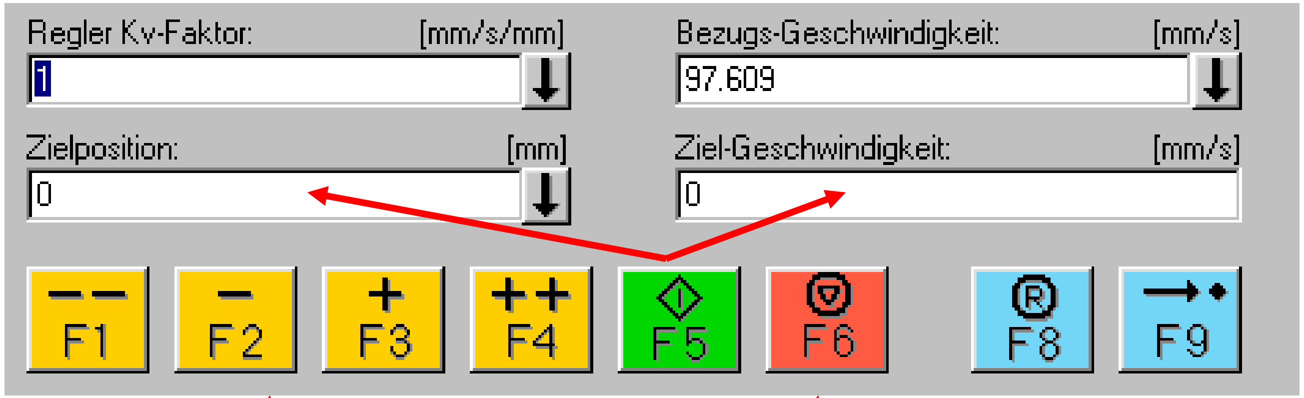
No enabling

Position control, motion locked

Position control + motion enabled. WARINING: consider override!

NC settings demo-rack with KL2531

- Axis in online mode



Operating mode: incremental jog

Start PTP motion command

Stop axis movement

Abort axis command

Start referencing

NC settings demo-rack with KL2531

Further information about Micro-stepping:

The resulting step width can be adjusted via the function micro-stepping.

At the terminal KL2531 / 2541 this setting can be changed with a access to register 46.

This access can be done in several ways:

K bus coupler:

- **KS2000 terminal configuration software**

PLC program via the PLC interface of buscoupler (CX K bus master)

- **Library: TcPlcCoupler.LIB (FB_ReadCouplerRegs/ FB_WriteCouplerRegs)**

EtherCAT with K- bus coupler and KL2531/ 2541 or EL7031 / 7041 stepper motor terminal:

- **System Manager startup list of bus coupler / EL terminal**
- **PLC program with parameter access via CoeRead and CoeWrite at EtherCAT BK**

Library: TcEtherCAT.LIB (FB_EcCoESdoWrite FB_EcCoESdoRead)

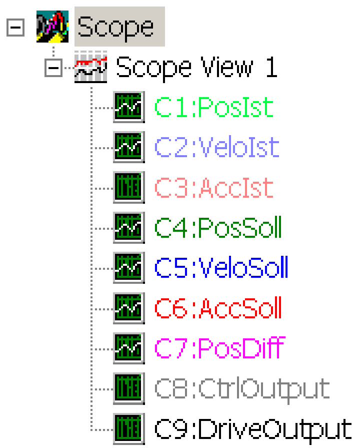
Testing the axis with the TwinCAT oscilloscope functions

Two Scope packages available:

1. **TwinCAT ScopeView as part of the TwinCAT installation**
2. **TwinCAT Scope2 as supplement**

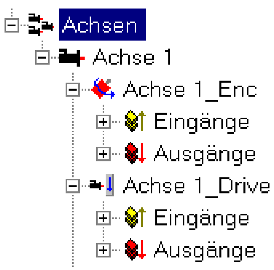
ScopeView

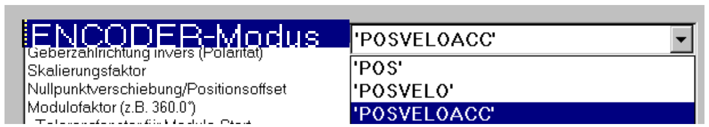
Prepared ScopeView data for axis are installed with TwinCAT under …\TwinCAT\Scope\Achse1.scp



C1: actual position
C2: actual velocity
C3: actual acceleration
C4: set position
C5: set velocity
C6: set acceleration
C7: following error
C8: cotroller output in %
C9: total output

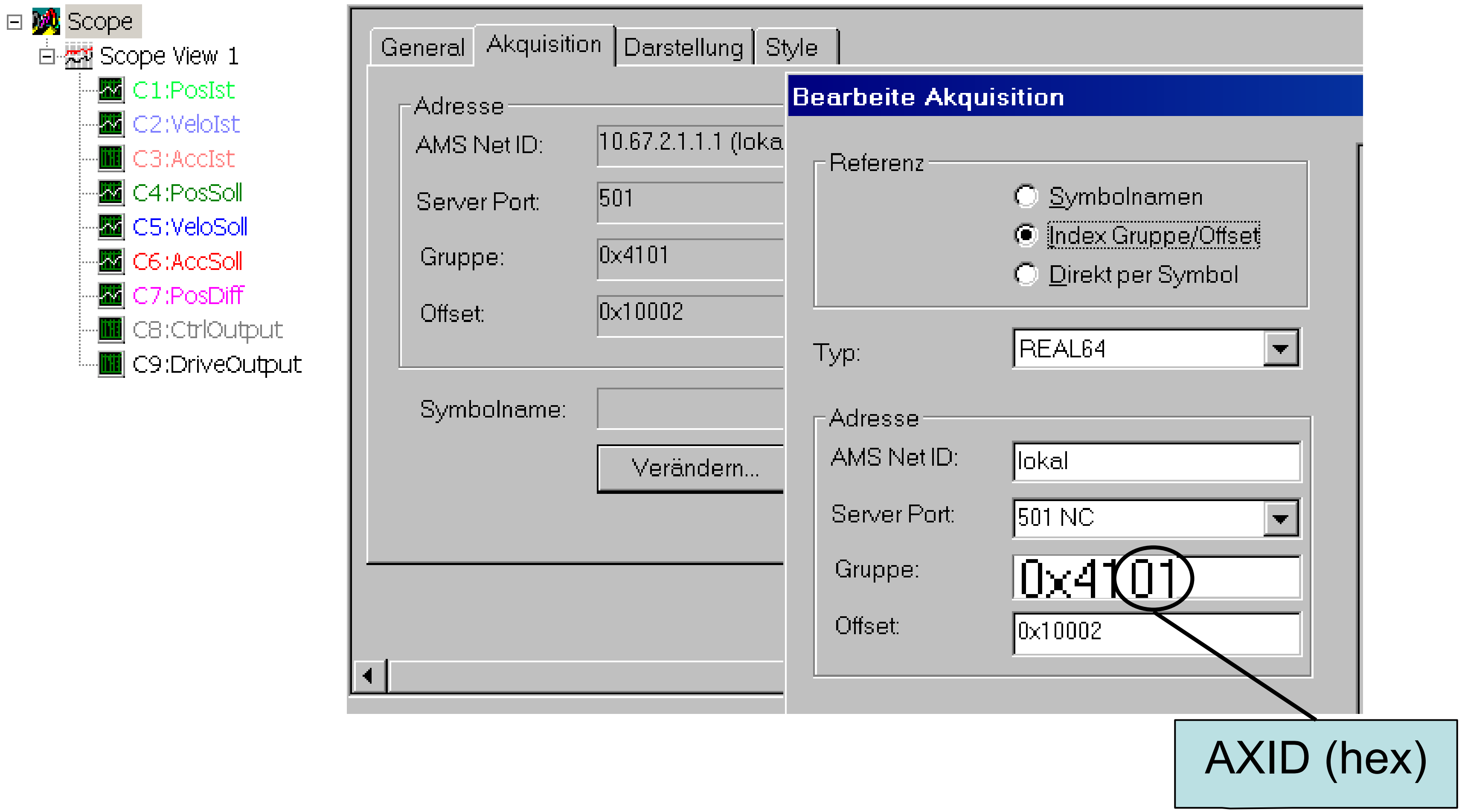
Consider Encoder Mode for „AccIst“ record





ScopeView

Change aquisition for further axes:



ScopeView „measurement points“