X-GO Flex Logic Control Documentation (Cluster 64 Bit)

Date: Sept,22.2022

X-GO Flex Logic Control Documentation

SYBERA GmbH Copyright © 2014

Table of content

1 INTRODUCTION 4
1.1 Features 4
1.2 Supported OS 6
1.3 Supported Platforms 6
1.4 Reference Hardware 6

2 FLEX MODULES 7
2.1 FlexProtocol 7
2.2 FlexMonitor 8
2.2.1 Sample: Drive Startup Sequence 9
2.3 FlexCoding 10
2.3.1 PLC Execution 11
2.3.2 PLC Language structure 12
2.3.3 Station Value 13
2.3.4 Constant Value 13
2.3.5 Variables 14
2.3.6 Buffers 16
2.3.7 Standard Operators 17
2.3.8 Conditional Operators 17
2.3.9 Jumps 18
2.3.10 Functions 20
2.4 FlexExtend 27
2.4.1 Function Encoder 27
2.4.2 Function ConvertEndian 28
2.4.3 Function Delay 28
2.4.4 DriveEnable 29
2.4.5 DriveDisable 29
2.4.6 DriveHoming 30
2.4.7 DriveJog 30
2.4.8 DriveIterate 31
2.4.1 DriveJog 33
2.4.2 PneumaticMotion 33
2.4.3 GetTemperatureLinear 34
2.5 FlexExchange 35
2.5.1 Buffer Exchange 35
2.5.2 Buffer Synchronization 36
2.6 FlexDebug 37
2.6.1 Setting a Breakpoint 37
2.6.2 Setting an execution line 39
2.6.3 Logic Check 40
2.7 FlexPanel (HmiDesignBox) 41
2.7.1 Panel parameter 42
2.7.2 Panel selection 43
2.8 Background Image 44

3 X-GO INSTALLATION 45
3.1 BIOS Adjustment 48
3.1.1 Disable Intel SpeedStep Technology 48
3.1.2 Disable C-States 48
3.1.3 Disable TurboMode 48
3.2 Jitter Control 49
3.3 AutoStart 50

4 ETHERCAT 51

4.1 XML Converter 53
4.2 PDO Configurator 54
4.3 Device Parameter File 58
4.3.1 Section [NAME] 59
4.3.2 Section [VENDOR] 59
4.3.3 Section [CODE] 59
4.3.4 Section [REVISION] 59
4.3.5 Section [SYNCMAN] 60
4.3.6 Section [FMMU] 62
4.3.7 Section [SDO] 63
4.3.8 Section [OUTPUT] / [INPUT] 65

5 PROFINET 66
5.1 Creating a configuration file 68
5.2 Accesspoint Module 69
5.3 Station Settings 70
5.4 Functional Module 72
5.5 Station Diagnostics 75

6 ETHERNET/IP 76
6.1 Creating a configuration file 78
6.1.1 Device Module 80
6.1.2 Device Settings 81
6.1.3 Connection Module 82
6.1.4 Connection Settings 83
6.1.5 Device Diagnostics 84

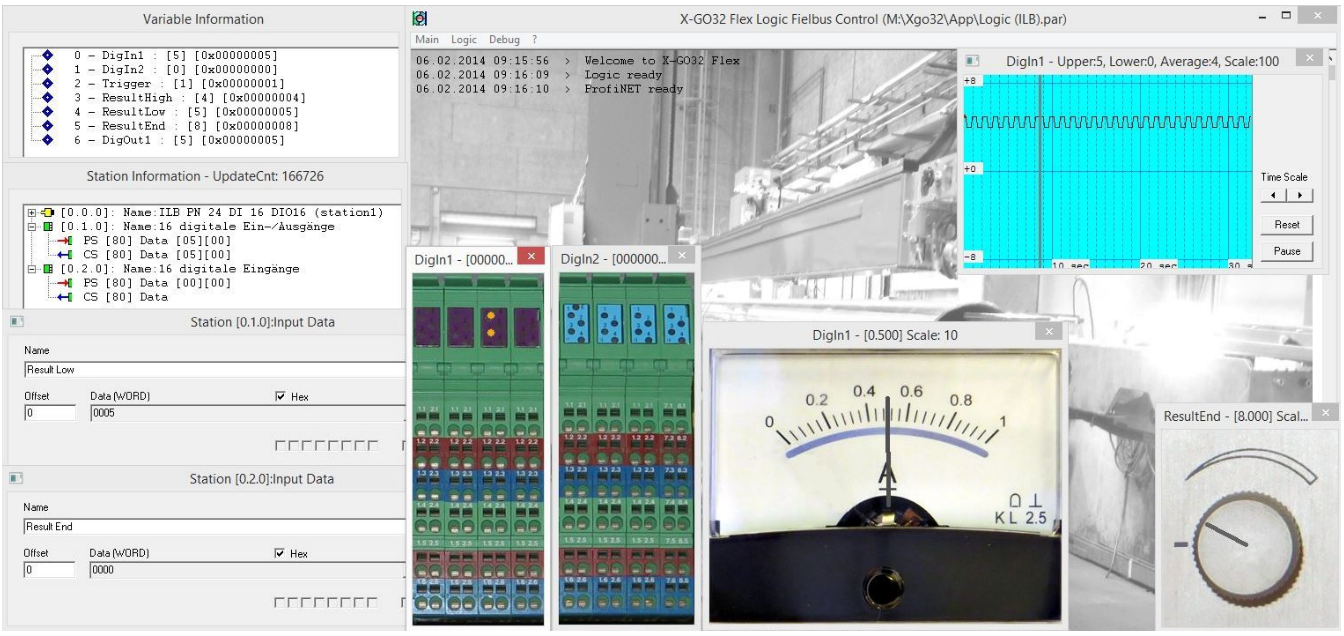
7 SERCOS III 87
7.1 Sercos III Check 89
7.2 Configuration Parameter File 90
7.2.1 Section [NAME] 91
7.2.2 Section [DEVICEID] 91
7.2.3 Section [PARAM] 91
7.2.4 Section [CONFIG] 91
7.2.5 Section [DESC] 92

8 ERROR HANDLING 94
8.1 Debug LOG File 94
8.2 Event File 94

X-GO Flex Logic Control Documentation

1 Introduction

The software X-GO Flex for Windows allows the control of fieldbus devices (EtherCAT, ProfiNET, SERCOS III and Ethernet/IP) under realtime condition, without programming. All stations can be controlled logically with a PLC programming language. On this occasion, different language elements are provided for evaluation of constants and station values, for timer, counter and conditional jumps. Additional function modules may be added to the system. Moreover, the input is checked for logical errors with a PLC program checker. X-GO Flex enables the complete administration of all fieldbus stations. The station monitor shows the current input and output conditions. Data exchange between PLC system and outside applications can be easily done with the X-GO Software Development Kit (SDK). The ability of offering a photorealistic desktop screen, simply designed by parameter files, allows a new dimension for control applications.


Variable Information
Station Information - UpdateCnt: 166726
X-GO32 Flex Logic Fielbus Control (M:\Xgo32\App\Logic (ILB).par)
Main Logic Debug ?
DigIn1 - Upper:5, Lower:0, Average:4, Scale:100
+8
+0
-8
10 sec
20 sec
30 s
Time Scale
Reset
Pause
DigIn1 - [00000...
DigIn2 - [000000...
DigIn1 - [0.500] Scale: 10
0
0.2
0.4
0.6
0.8
1
KL 2.5
ResultEnd - [8.000] Scal...

```
0 - DigIn1 : [5] [0x00000005]
1 - DigIn2 : [0] [0x00000000]
2 - Trigger : [1] [0x00000001]
3 - ResultHigh : [4] [0x00000004]
4 - ResultLow : [5] [0x00000005]
5 - ResultEnd : [8] [0x00000008]
6 - DigOut1 : [5] [0x00000005]
```

```
[0.0.0]: Name:ILB PN 24 DI 16 DIO16 (station1)
[0.1.0]: Name:16 digitale Ein-/Ausgänge
    PS [80] Data [05][00]
    CS [80] Data [05][00]
[0.2.0]: Name:16 digitale Eingänge
    PS [80] Data [00][00]
    CS [80] Data
```

Station [0.1.0]:Input Data

Name: Result Low
Offset: 0 Data (WORD): 0005 [✓ Hex]

Station [0.2.0]:Input Data

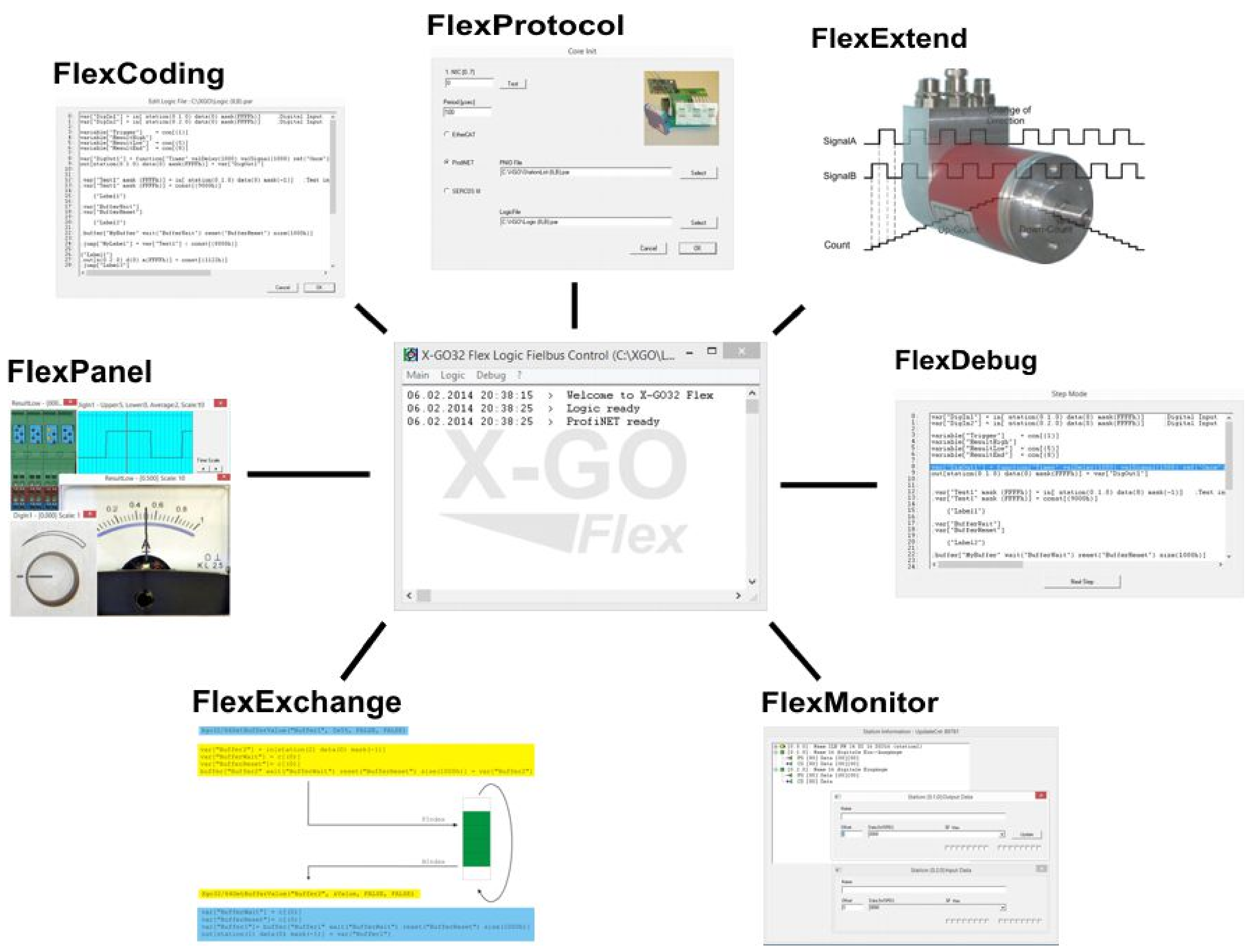
Name: Result End
Offset: 0 Data (WORD): 0000 [✓ Hex]

```
06.02.2014 09:15:56  >  Welcome to X-GO32 Flex
06.02.2014 09:16:09  >  Logic ready
06.02.2014 09:16:10  >  ProfiNET ready
```

1.1 Features

• Flex Protocol	EtherCAT, ProfiNET, Sercos III and Ethernet/IP support
• Flex Monitor	Input / Output monitoring with position mask
• Flex Coding	adjustable PLC programming language syntax
• Flex Debug	interactive PLC debugging system with code interpretation
• Flex Panel	photorealistic, parameter controlled HMI
• Flex Extend	programming interface for PLC function extensions
• Flex Exchange	programming interface for exchanging data with outside applications

X-GO Flex Logic Control Documentation



```
06.02.2014 20:38:15  >  Welcome to X-GO32 Flex
06.02.2014 20:38:25  >  Logic ready
06.02.2014 20:38:25  >  ProfiNET ready
```

1.2 Supported OS

- Windows 7
- Windows 8
- Windows 10

1.3 Supported Platforms

- INTEL Multi-Core (i3 / i5 / i7 / CoreDuo)
- AMD Multi-Core (x2 / x4)

1.4 Reference Hardware

X-GO was especially tested with following hardware platforms:

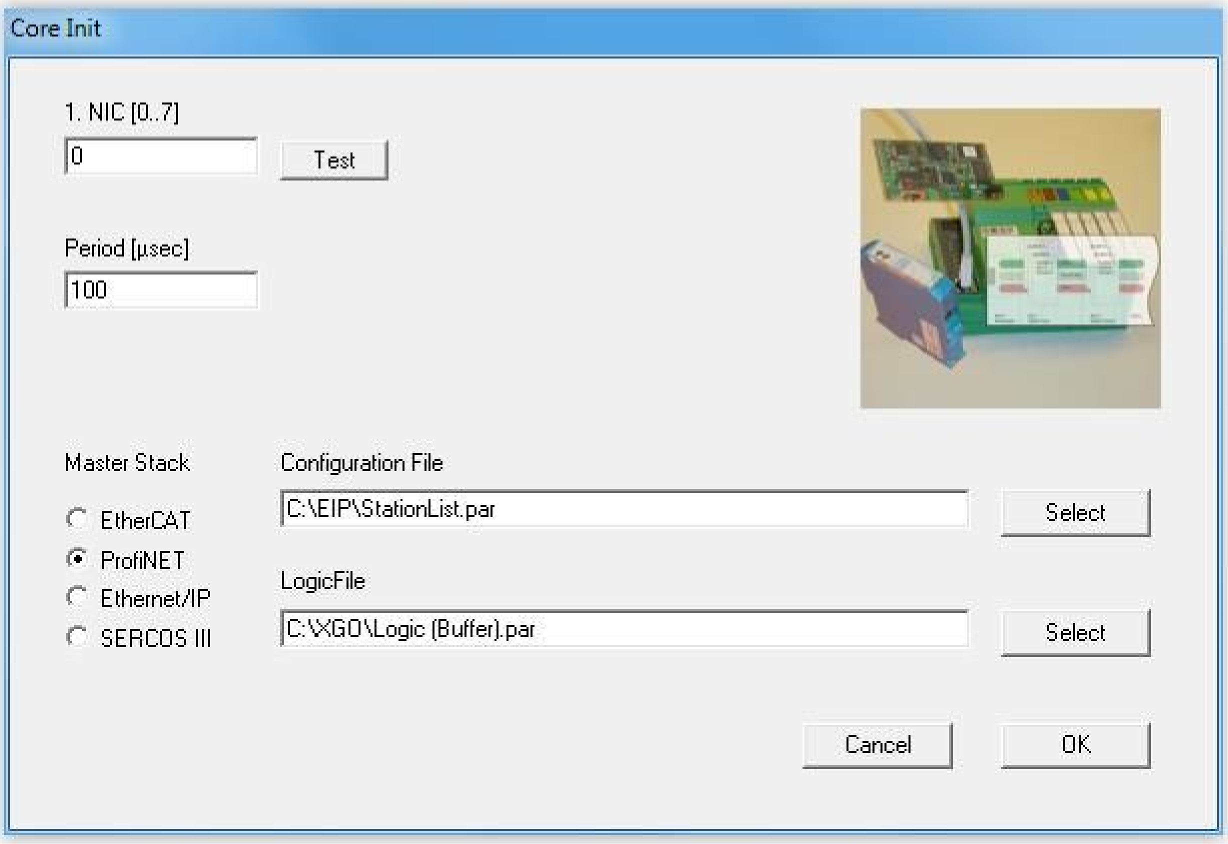
Mainboard:	DQ57TM
Processor:	INTEL I5
RAM:	4 GB
Graphics:	RADEON 9800 PRO
Mainboard:	ASROCK 785GM-S3
Processor:	ATHLON II X4 320
RAM:	4 GB
Graphics:	ATI Radeon HD 4200 (OnBoard)
Mainboard:	ASUS P5LD2-SE
Processor:	INTEL Core 2 Duo
RAM:	3 GB
Graphics:	ATI Radeon HD 4300
Mainboard:	Lenovo 44444XG
Processor:	INTEL Core i5 M480
RAM: 4 GB	
Graphics:	INTEL HD Graphics BR-1004 (OnBoard)

2 Flex Modules

2.1 FlexProtocol

Whether ProfiNET, EtherCAT, SERCOS III or Ethernet/IP, X-GO Flex based on the X-Realtime Engine for Windows controls the fieldbus protocols without complex programming. A multiple instantiation of X-GO Flex enables even the simultaneous operation of different protocols. All parameters are saved within the file XGO.PAR.

- EtherCAT
- ProfiNET
- Sercos III
- Ethernet/IP



Core Init

1. NIC [0..7]: 0

Period [µsec]: 100

Master Stack: ( ) EtherCAT [✓ ProfiNET] ( ) Ethernet/IP ( ) SERCOS III

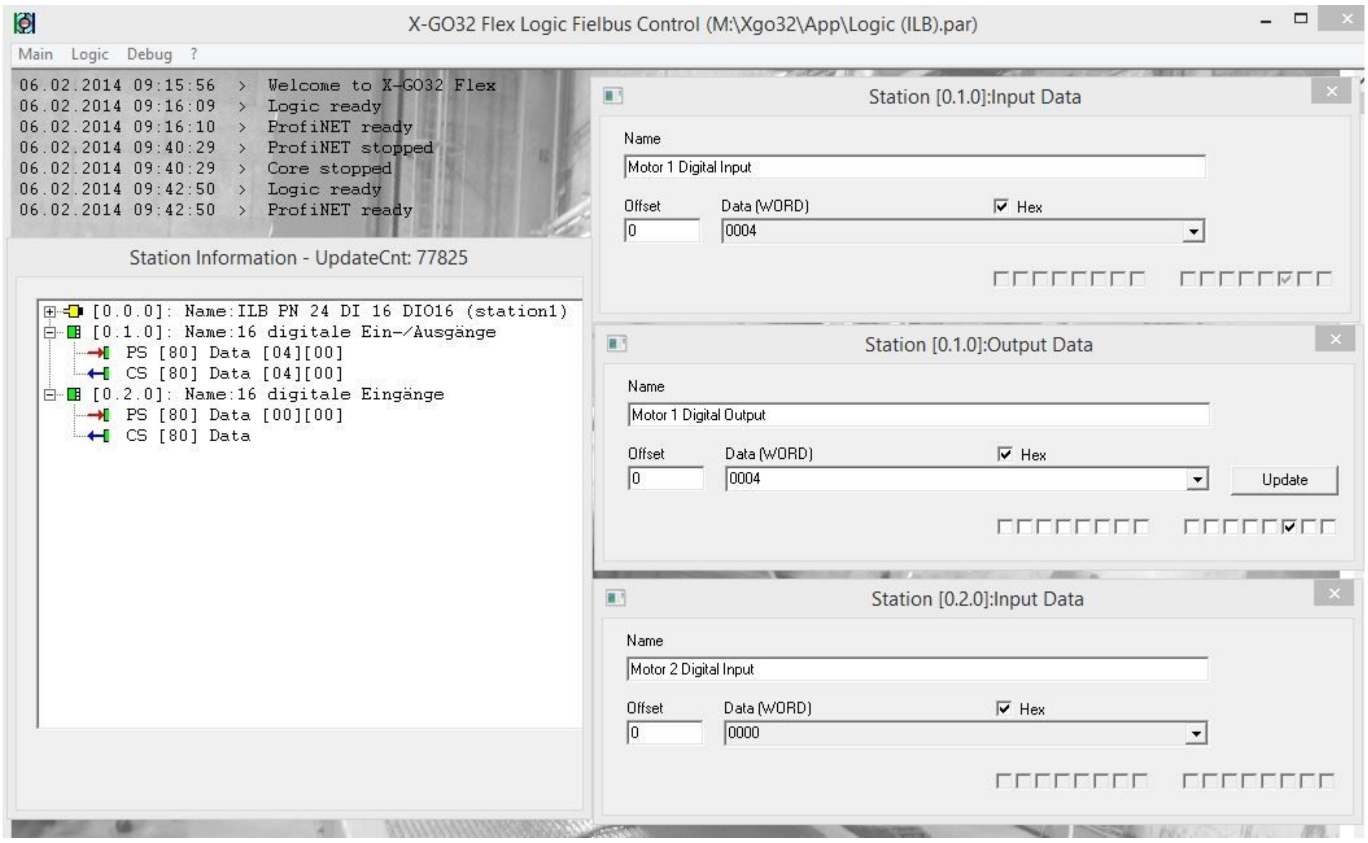
Configuration File: C:\EIP\StationList.par

LogicFile: C:\XGO\Logic (Buffer).par

Each protocol has its own initial parameter set, like sampling period, synchronisation period, network card index. Additionally the network connection may tested by pressing the [TEST] button. If so, the network link LED will turn off and on to signal the correct adapter choice.

2.2 FlexMonitor

The Flex Monitor provides you an easy to use station control system which is particularly suitable for service applications. It allows you to display the input data, and the change of output data with different data masks. All data masks are saved within the file "MASK.PAR" (or another given name) by leaving the software. The saved dialog location and name will be used by the Auto Start function..


X-GO32 Flex Logic Fielbus Control (M:\Xgo32\App\Logic (ILB).par)
Main Logic Debug ?

```
06.02.2014 09:15:56  >  Welcome to X-GO32 Flex
06.02.2014 09:16:09  >  Logic ready
06.02.2014 09:16:10  >  ProfiNET ready
06.02.2014 09:40:29  >  ProfiNET stopped
06.02.2014 09:40:29  >  Core stopped
06.02.2014 09:42:50  >  Logic ready
06.02.2014 09:42:50  >  ProfiNET ready
```

Station Information - UpdateCnt: 77825

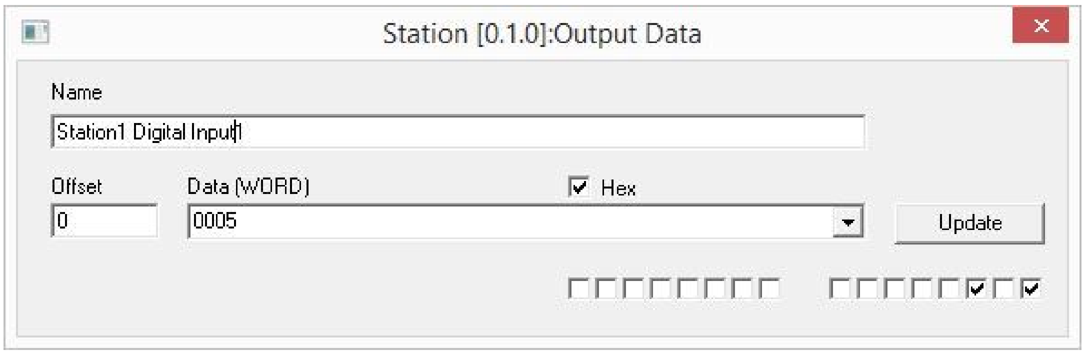
```
[0.0.0]: Name:ILB PN 24 DI 16 DIO16 (station1)
[0.1.0]: Name:16 digitale Ein-/Ausgänge
    PS [80] Data [04][00]
    CS [80] Data [04][00]
[0.2.0]: Name:16 digitale Eingänge
    PS [80] Data [00][00]
    CS [80] Data
```

Station [0.1.0]:Input Data
Name: Motor 1 Digital Input
Offset: 0 — Data (WORD): 0004 — [✓ Hex]

Station [0.1.0]:Output Data
Name: Motor 1 Digital Output
Offset: 0 — Data (WORD): 0004 — [✓ Hex] — Update

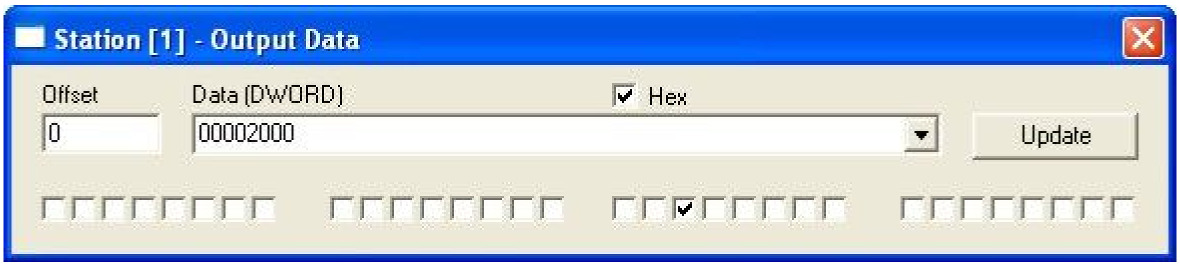
Station [0.2.0]:Input Data
Name: Motor 2 Digital Input
Offset: 0 — Data (WORD): 0000 — [✓ Hex]

The value dialog allows reading or writing of station values. Just click on a station input or output item inside the Station Monitor opens the dialog. The data format is adjustable (due to the data length) and provides BYTE, WORD, DWORD, ASCII format (either HEX or DEC). The data offset determines the beginning of the payload data (Sercos III without CCON). The Data may be entered or read binary as well as text with an adjustable data format


Station [0.1.0]:Output Data
Name: Station1 Digital Input
Offset: 0 — Data (WORD): 0005 — [✓ Hex] — Update

2.2.1 Sample: Drive Startup Sequence

1. Set Drive Control Word (Bit 13) : Drive Restart


Station [1] - Output Data

Offset: 0

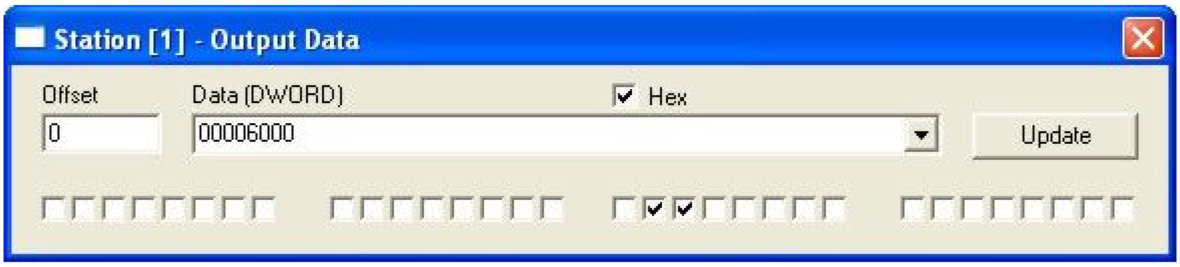
Data (DWORD): 00002000

[✓] Hex

Update

☐☐☐☐☐☐☐☐ ☐☐☐☐☐☐☐☐ ☐☐✓☐☐☐☐☐ ☐☐☐☐☐☐☐☐

2. Set Drive Control Word (Bit 14) : Drive Enable


Station [1] - Output Data

Offset: 0

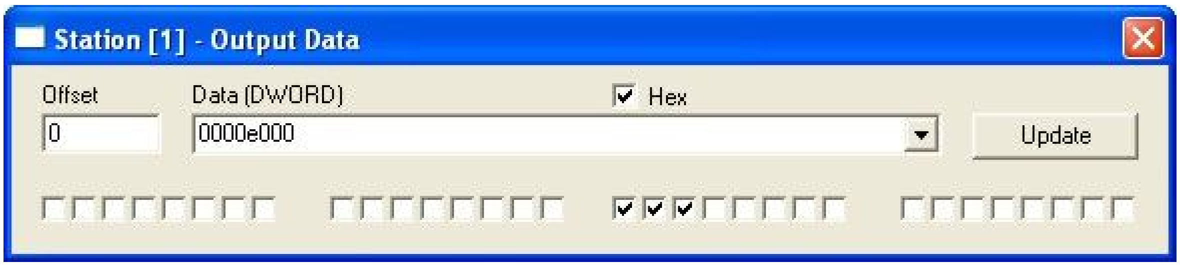
Data (DWORD): 00006000

[✓] Hex

Update

☐☐☐☐☐☐☐☐ ☐☐☐☐☐☐☐☐ ☐✓✓☐☐☐☐☐ ☐☐☐☐☐☐☐☐

3. Set Drive Control Word (Bit 15) : Drive ON


Station [1] - Output Data

Offset: 0

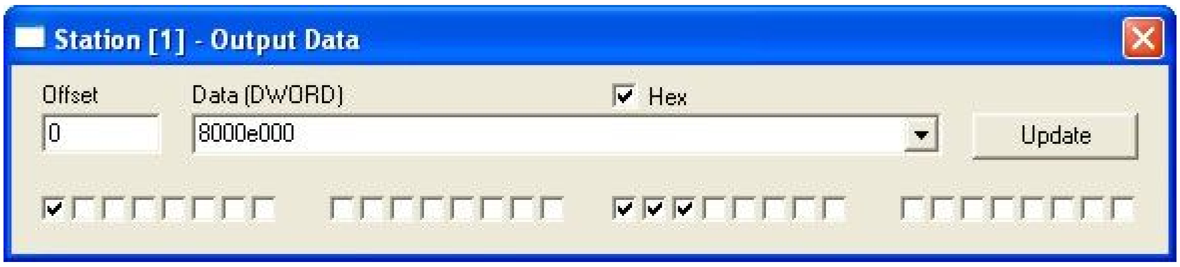
Data (DWORD): 0000e000

[✓] Hex

Update

☐☐☐☐☐☐☐☐ ☐☐☐☐☐☐☐☐ ✓✓✓☐☐☐☐☐ ☐☐☐☐☐☐☐☐

4. Set Drive Velocity (Bit 31)


Station [1] - Output Data

Offset: 0

Data (DWORD): 8000e000

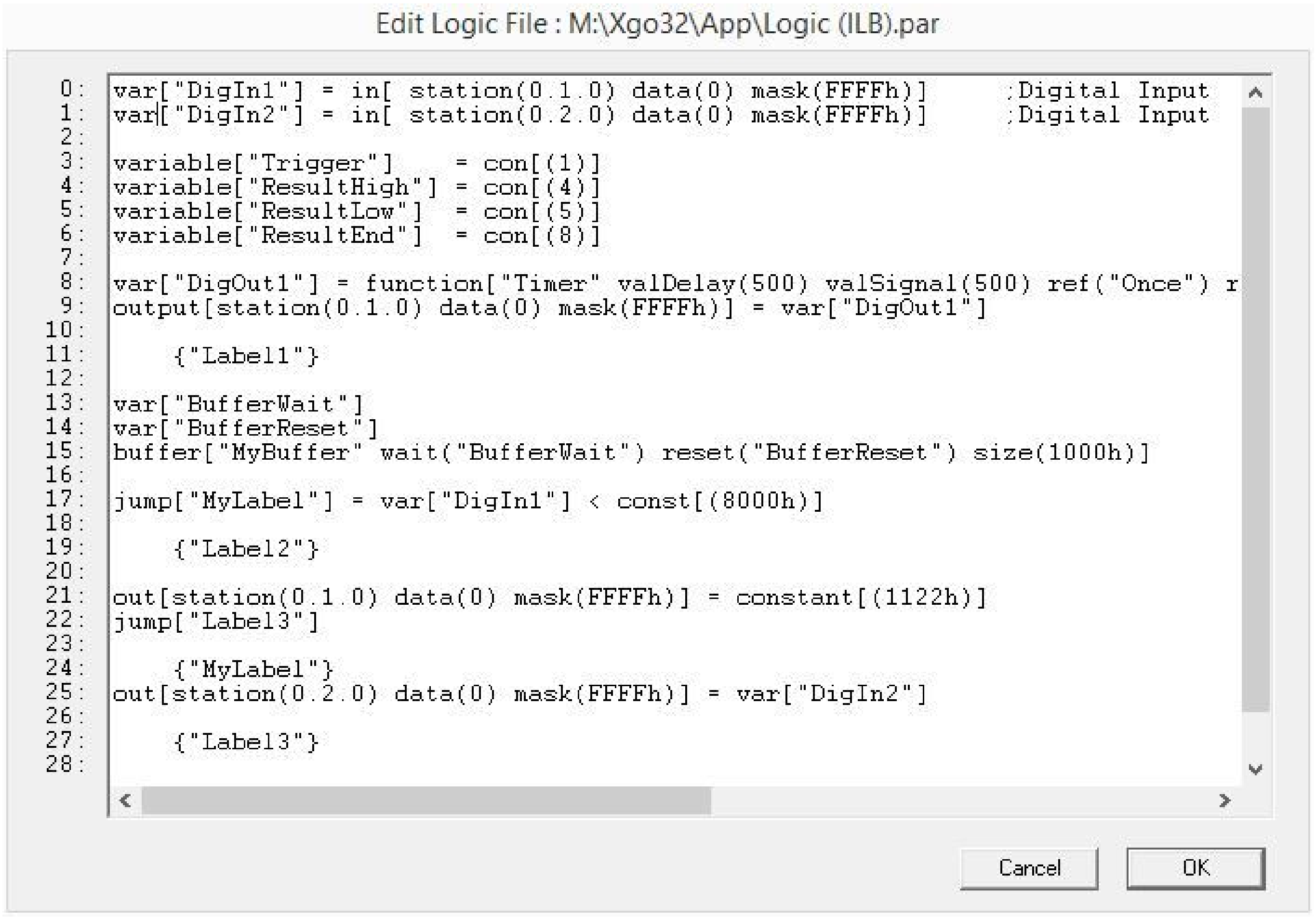
[✓] Hex

Update

✓☐☐☐☐☐☐☐ ☐☐☐☐☐☐☐☐ ✓✓✓☐☐☐☐☐ ☐☐☐☐☐☐☐☐

2.3 FlexCoding

A programming language adapting to your spelling preference ?
Write your PLC realtime code with your own familar syntax, with all the language elements and without crashes. A code checker checks the code and displays errors. X-GO Flex provides an editor to edit the PLC program. The PLC language provides all elements of a programming language and may be extended by own function modules (see FlexExtend). The PLC program is saved as text file (default name LOGIC.PAR)



```
var["DigIn1"] = in[ station(0.1.0) data(0) mask(FFFFh)]     ;Digital Input
var["DigIn2"] = in[ station(0.2.0) data(0) mask(FFFFh)]     ;Digital Input

variable["Trigger"]    = con[(1)]
variable["ResultHigh"] = con[(4)]
variable["ResultLow"]  = con[(5)]
variable["ResultEnd"]  = con[(8)]

var["DigOut1"] = function["Timer" valDelay(500) valSignal(500) ref("Once") r
output[station(0.1.0) data(0) mask(FFFFh)] = var["DigOut1"]

    {"Label1"}

var["BufferWait"]
var["BufferReset"]
buffer["MyBuffer" wait("BufferWait") reset("BufferReset") size(1000h)]

jump["MyLabel"] = var["DigIn1"] < const[(8000h)]

    {"Label2"}

out[station(0.1.0) data(0) mask(FFFFh)] = constant[(1122h)]
jump["Label3"]

    {"MyLabel"}
out[station(0.2.0) data(0) mask(FFFFh)] = var["DigIn2"]

    {"Label3"}
```

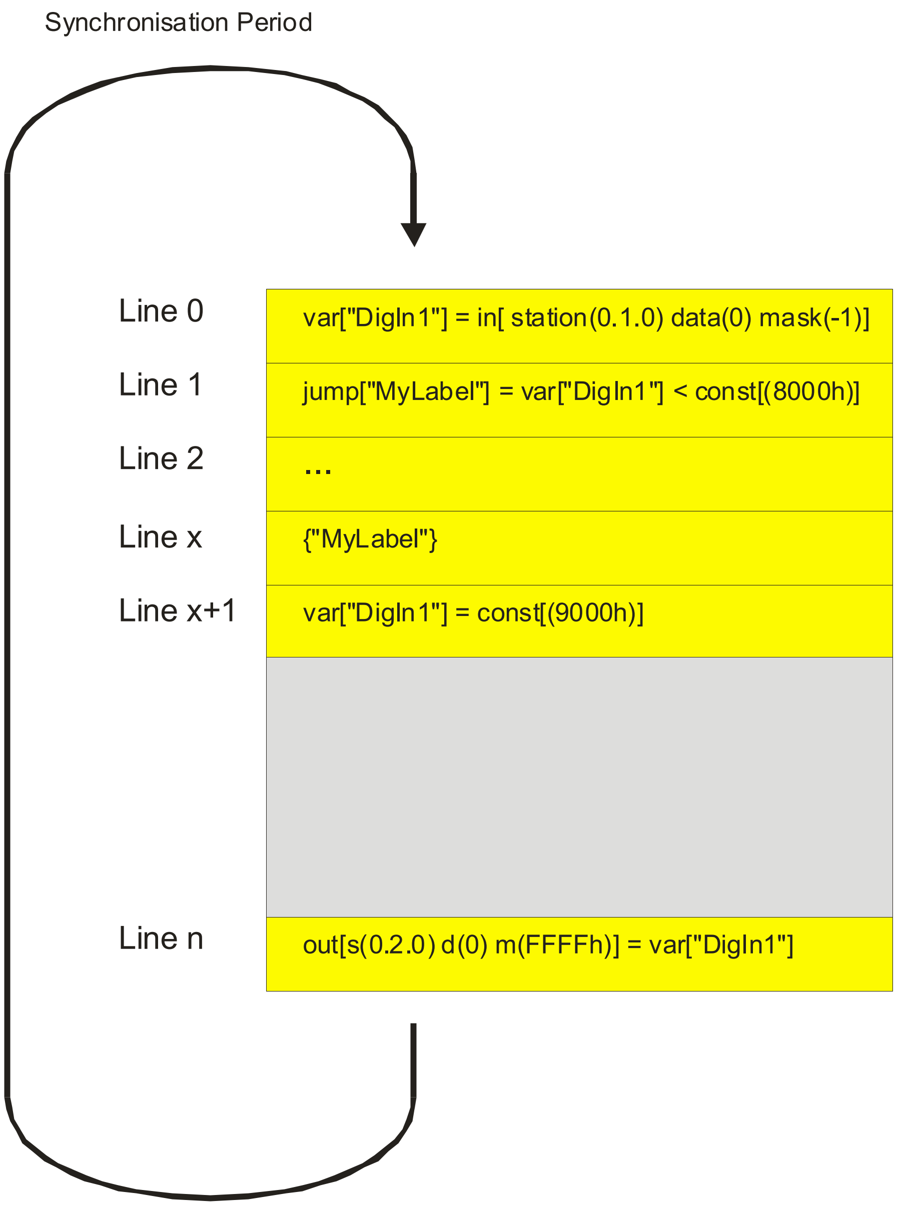
The adapting syntax allows using element declarations of your own preference. So these declarations are equal:

```
v["Sensor"]
var["Sensor"]
variable["Sensor"]
vMyVariable["Sensor"]
```

Only the first character (prefix) of the language element is fixed and must be common.

2.3.1 PLC Execution

All PLC program lines are executed at realtime, starting from the first line to the last and starting again in a loop by the given realtime synchronization period:


Synchronisation Period
Line 0
var["DigIn1"] = in[station(0.1.0) data(0) mask(-1)]
Line 1
jump["MyLabel"] = var["DigIn1"] < const[(8000h)]
Line 2
...
Line x
{"MyLabel"}
Line x+1
var["DigIn1"] = const[(9000h)]
Line n
out[s(0.2.0) d(0) m(FFFFh)] = var["DigIn1"]

2.3.2 PLC Language structure

A PLC program line consists of 1, 3 or 5 language elements:

Element1	Element2	Element3	Element4	Element5
Variable / Buffer / Station /Jump	Operator [=]	Variable / Buffer / Station / Constant / Function	Logical Operator [+,-,/.*,<,>,&,|,^,?,:,#]	Variable / Buffer / Station /Constant / Function

1 element PLC line:

Element1 = Variable / Buffer / Station /Jump

```
jump["MyLabel"]    ;jump to MyLabel

var["BufferWait"]  ;global variable
```

3 elements PLC line:

Element1 : Variable / Buffer / Station /Jump
Element2 : Operator [=]
Element3 : Variable / Buffer / Station / Constant / Function

```
variable[“DigIn1”] = input[station(0.1.0 data(0) mask(-1)]

var[“ResultLow”] = const[(9000h)]    ;set result low for timer

jump["MyLabel"] = var["Cond"]
```

5 language elements line:

Element1 : Variable / Buffer / Station /Jump
Element2 : Operator [=]
Element3 : Variable / Buffer / Station / Constant / Function
Element4 : Logical Operator [+,-,/.*,<,>,&,|,^,?,:,#]
Element3 : Variable / Buffer / Station / Constant / Function

```
variable[“DigIn1”] = input[station(0.1.0 data(0) mask(-1)] + constant[(5)]

jump["MyLabel"] = var["Test1"]< const[(8000h)]
```

The size of each value is upto 4 bytes. Each program line may be decorated optionally by comments. Comments are separated by [;]. Even if a line contains only comments, it`s included to the logic cycle as NOP program line.

2.3.3 Station Value

The station value defines a location inside the station data input or output area. The size of each element is upto 4 bytes. The station element may consists of a combination of the parameters StationAddress (see FlexMonitor), DataOffset, BitMask and DataType. Due to the controlled fieldbus, the station address contains of the station index (EtherCAT, Sercos III) or the station index , module index and submodule index (ProfiNET). All station indexes starting from zero and have decimal format.

```
output[station(0.2.0) data(9) mask(FFh) type(4)] ;output station address 0.2.0
                                                 ;data Offset 9
                                                 ;bit mask FFh
                                                 ;data type string

out[s(3) d(0) m(2)]          ;output station address 0
;data offset 0
;bit mask 2

input[station(1) d(2) m(-1)]  ;input station address 1
;data offset 2
;bit mask 0xFFFFFFFF
```

Data Types:
0 : None (DWORD)
1 : BYTE
2 : WORD
3 : DWORD
4 : WORD STRING
5 : DWORD STRING

2.3.4 Constant Value

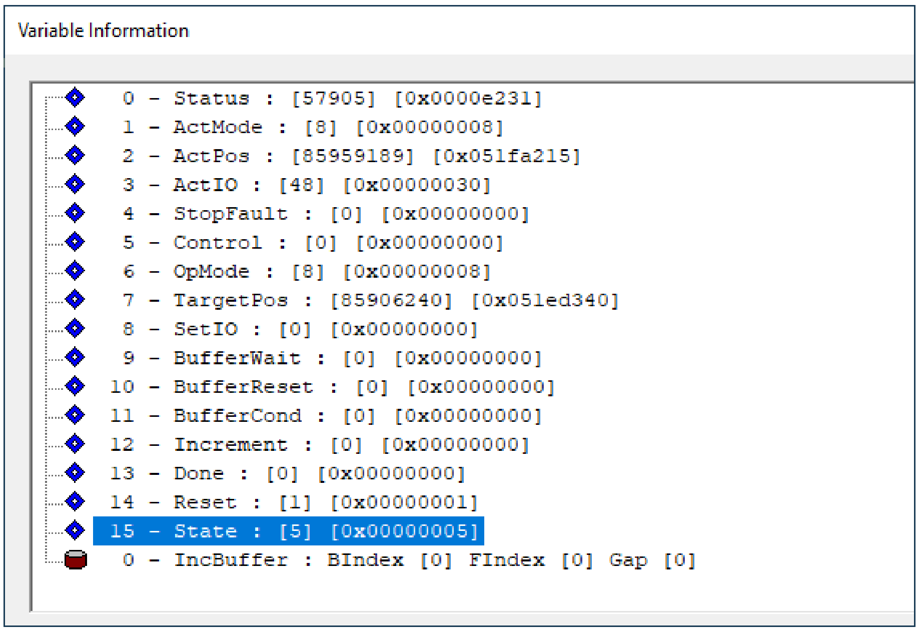
Constant values may have decimal or hexadecimal format. The size of each element is upto 4 bytes.

```
constant[(3)]    ;constant value decimal 3
con[(-1)]         ;constant value decimal -1 (0xFFFFFFFF)
c[(15h)]          ;constant value hexadecimal 0x15
```

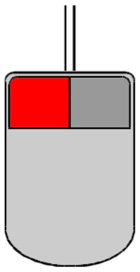
2.3.5 Variables

Variables are used as place holders for any language element. The size of each element is upto 4 bytes. A variable element may consists of the parameters Name in combination of the parameter BitMask.

```
v["Sensor1"]      ;sensor 1
var["DigIn1"]     ;digital input 1
variable["Motor2"]            ;motor 2
vMyVariable["Test"]     ;test variable
```

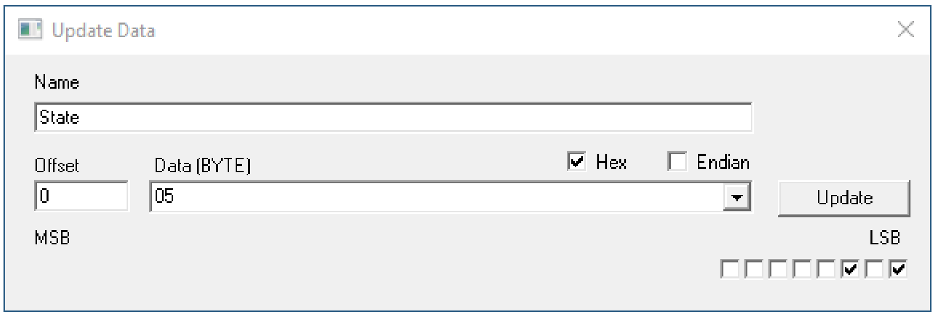

Variable Information

```
 0 - Status : [57905] [0x0000e231]
 1 - ActMode : [8] [0x00000008]
 2 - ActPos : [85959189] [0x051fa215]
 3 - ActIO : [48] [0x00000030]
 4 - StopFault : [0] [0x00000000]
 5 - Control : [0] [0x00000000]
 6 - OpMode : [8] [0x00000008]
 7 - TargetPos : [85906240] [0x051ed340]
 8 - SetIO : [0] [0x00000000]
 9 - BufferWait : [0] [0x00000000]
10 - BufferReset : [0] [0x00000000]
11 - BufferCond : [0] [0x00000000]
12 - Increment : [0] [0x00000000]
13 - Done : [0] [0x00000000]
14 - Reset : [1] [0x00000001]
15 - State : [5] [0x00000005]
 0 - IncBuffer : BIndex [0] FIndex [0] Gap [0]
```



2.3.5.1 Changing a variable value

On pressing the left button at the selected variable, the value dialog appears. In debug mode, the value can be changed immediately


Update Data

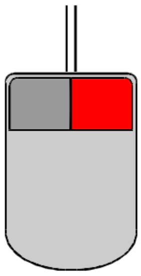
Name: State

Offset: 0

Data (BYTE): 05

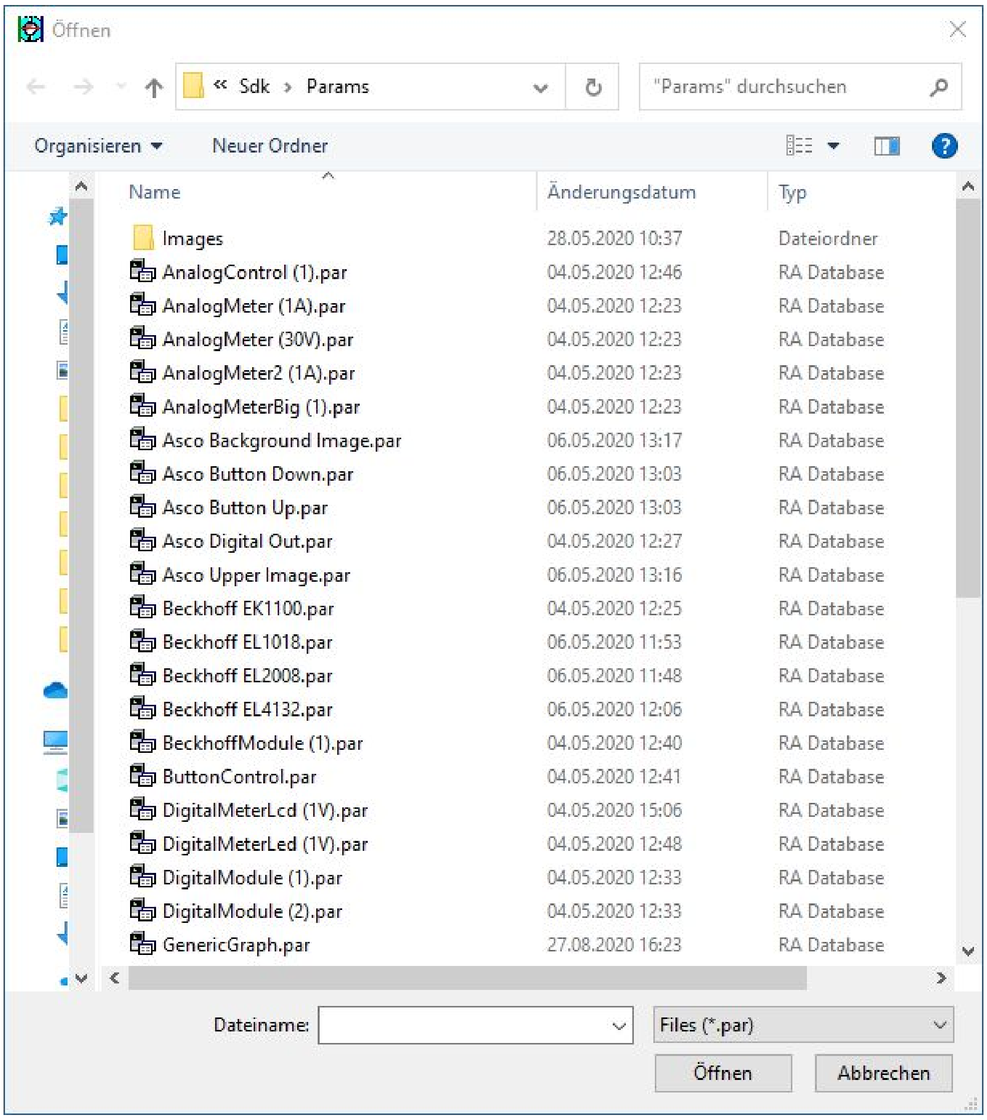
[✓] Hex [ ] Endian

☐ ☐ ☐ ☐ ☐ ☑ ☐ ☑



2.3.5.2 Selecting a panel

On pressing the right mouse button, a panel can be selected for the variable (see FlexPanel)



Name	Änderungsdatum	Typ
Images	28.05.2020 10:37	Dateiordner
AnalogControl (1).par	04.05.2020 12:46	RA Database
AnalogMeter (1A).par	04.05.2020 12:23	RA Database
AnalogMeter (30V).par	04.05.2020 12:23	RA Database
AnalogMeter2 (1A).par	04.05.2020 12:23	RA Database
AnalogMeterBig (1).par	04.05.2020 12:23	RA Database
Asco Background Image.par	06.05.2020 13:17	RA Database
Asco Button Down.par	06.05.2020 13:03	RA Database
Asco Button Up.par	06.05.2020 13:03	RA Database
Asco Digital Out.par	04.05.2020 12:27	RA Database
Asco Upper Image.par	06.05.2020 13:16	RA Database
Beckhoff EK1100.par	04.05.2020 12:25	RA Database
Beckhoff EL1018.par	06.05.2020 11:53	RA Database
Beckhoff EL2008.par	06.05.2020 11:48	RA Database
Beckhoff EL4132.par	06.05.2020 12:06	RA Database
BeckhoffModule (1).par	04.05.2020 12:40	RA Database
ButtonControl.par	04.05.2020 12:41	RA Database
DigitalMeterLcd (1V).par	04.05.2020 15:06	RA Database
DigitalMeterLed (1V).par	04.05.2020 12:48	RA Database
DigitalModule (1).par	04.05.2020 12:33	RA Database
DigitalModule (2).par	04.05.2020 12:33	RA Database
GenericGraph.par	27.08.2020 16:23	RA Database

2.3.6 Buffers

Buffers can be used to share information with the programming interface between the PLC program and an outside application (see FlexAttach, part of the X-GO SDK). The buffer elements consist of the parameters Name, Size and a reference to the control variables BufferWait, BufferReset and BufferCond.

```
var["BufferWait"]  = const[(0)]
var["BufferReset"] = const[(0)]
var["BufferCond"]  = const[(0)]
buffer["MyBuffer" wait("BufferWait") reset("BufferReset") cond("BufferCond")
size(1000h)]
```

Prameters:

[“buffer_name”]	buffer name
wait[“name”] (in):	if variable is set (1), buffering is suspended, else (0) buffering is active
reset[“name”] (in):	if variable is set (1), buffering is reset, else (0) buffering is active

condition[“name”] (out):	return buffer condition:
	
	BUFFER_EMPTY = 0
	BUFFER_BUSY = 1
	BUFFER_FULL = 2

size(x) (in):	buffer size. If buffer is located multiple times within the PLC program, only the first location specifies the buffer size

2.3.7 Standard Operators

X-GO provides almost all standard operators, like addition, subtraction, multiplication, division, negation, greater than, less than.

```
var["Limit”] = in[station (1) d(0) m(1)] + c[(1000)]        ;Addition
var["Hold”]  = in[station (2) d(0) m(FFFFh)] < c[(2000)]    ;if less than 2000 the
                                                            ;result becomes 1 else 0
```

Operator	Type	Result
&	LOGIC_TYPE_AND	Element1 & Element2
|	LOGIC_TYPE_OR	Element1 | Element2
^	LOGIC_TYPE_XOR	Element1 ^ Element2
~	LOGIC_TYPE_NOT	Element1 & (~Element2)
+	LOGIC_TYPE_PLUS	Element1 + Element2
-	LOGIC_TYPE_MINUS	Element1 - Element2
<	LOGIC_TYPE_LESS	(Element1 < Element2) ? 1 : 0
>	LOGIC_TYPE_GREATER	(Element1 > Element2) ? 1 : 0
*	LOGIC_TYPE_MUL	Element1 * Element2
/	LOGIC_TYPE_DIV	Element1 / Element2
?	LOGIC_TYPE_COND	Element1 ? Element2 : 0
:	LOGIC_TYPE_NOTCOND	Element1 ? 0 : Element2
#	LOGIC_TYPE_CMP	(Element1 == Element2) ? 1 : 0
!	LOGIC_TYPE_NOTCMP	(Element1 != Element2) ? 1 : 0

2.3.8 Conditional Operators

Conditional operators result with element2 on condition of element1, if the result is logical true, and 0 if the result is false

```
jump["MyLabel"] = var["Hold”] ? const[(1)]      ;Jump if var["Hold”] > 0
jump["MyLabel"] = var["Hold”] ? const[(0)]      ;Jump if var["Hold”] = 0
```

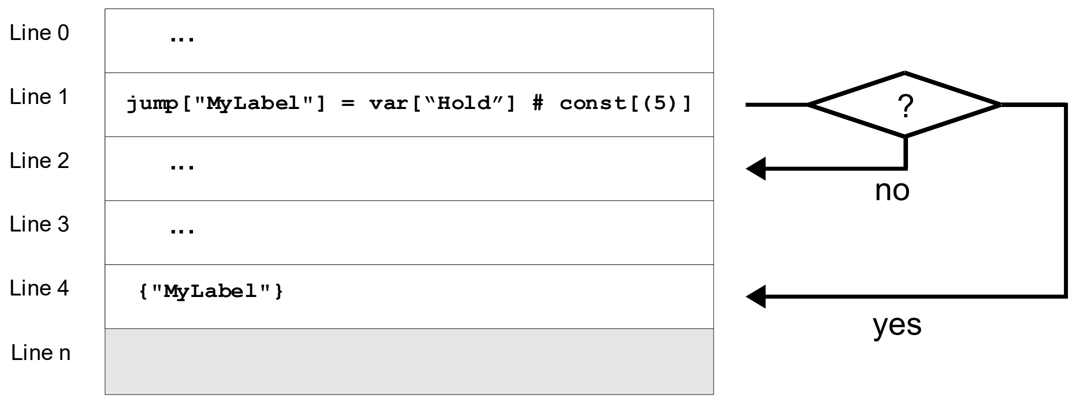
2.3.9 Jumps

Typically all programming languages provide conditional or unconditional jumps. X-GO Flex provides 3 jump types, conditional, unconditional and conditional wait.

2.3.9.1 Conditional Jump

Conditional jumps are proceeded due to the result of a conditional. The destination of the jump is the label line

jump["MyLabel"] = var[“Hold”] # const[(5)]

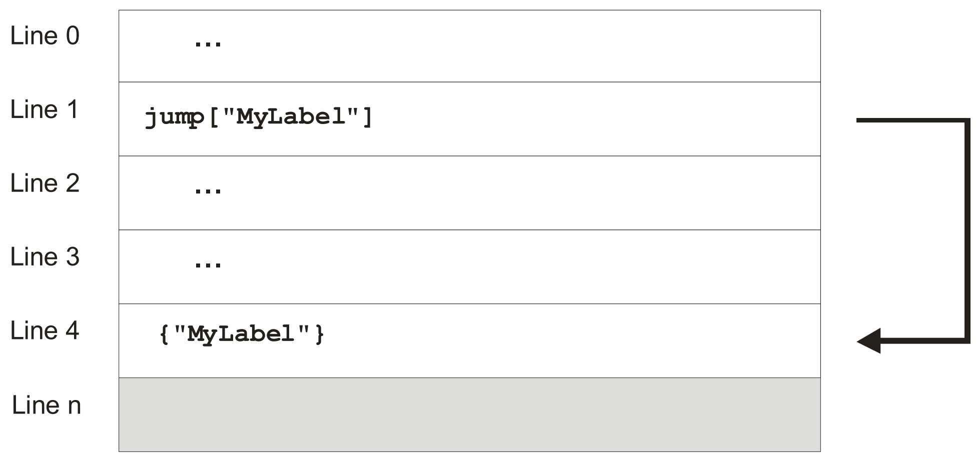


```
...
jump["MyLabel"] = var["Hold”] # const[(5)]
...
...
{"MyLabel"}
```

2.3.9.2 Unconditional jump

A unconditional jump is immediately executed. The destination of the jump is the label line

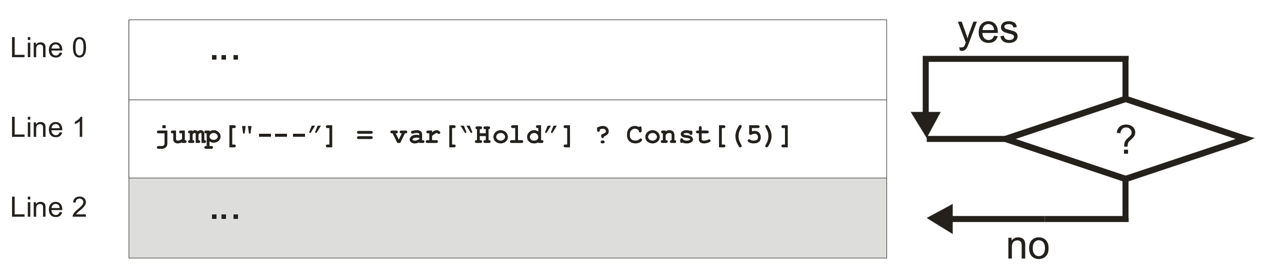
jump["MyLabel"]



Line 0	...
Line 1	`jump["MyLabel"]`
Line 2	...
Line 3	...
Line 4	`{"MyLabel"}`
Line n	

2.3.9.3 Conditional Wait

Conditional wait keeps the line position while“result == TRUE”. On “result == FALSE” the destination of the jump is the next line



Line 0	...
Line 1	`jump["---”] = var[“Hold”] ? Const[(5)]`
Line 2	...

2.3.10 Functions

X-GO Flex provides a set of PLC functionality, like TIMER, COUNTER, THRESHOLD, … .Each function may be used as language element inside the PLC program. A function consists of the FunctionName followed by parameters.

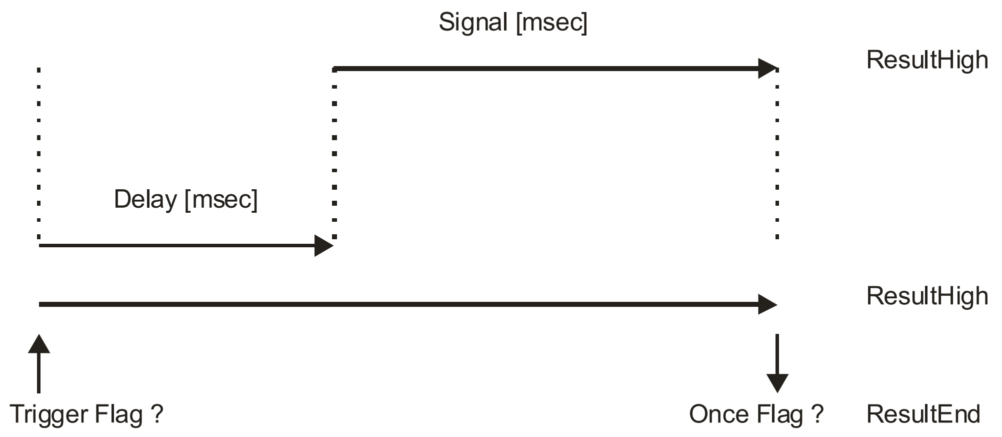
```
Var[“Result”] = function[“Name”, Param1, Param2, Param3, …)
```

Each function parameter may be a value or a reference. Each function returns a value.

```
Value:      val***(x)              e.g. valPosistion(2)
Reference:  ref***(“VarName”)      e.g. ref (“Trigger”)
```

2.3.10.1 Function Timer

Timers may be used as periodic or single-shot timer. The timer function is defined by the parameter order FunctionName, DelayTime, SignalTime, OnceFlag, ResultLow, ResultHigh, ResultEnd and TriggerFlag. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v). The timer resolution is 1 msec.



```
var[“Once”] = const[(0)]
var[“ResultLow”]   = const[(2)]
var[“ResultHigh”] = const[(8)]
var[“ResultEnd”]   = const[(0)]
var[“Trigger”]     = const[(1)]


var["Result"] = function["Timer" valDelay(500) valSignal(500) ref("Once")
                ref("ResultLow") ref("ResultHigh") ref("ResultEnd") ref("Trigger")]
```

equal

```
var["Result"] = function["Timer" vDelay(500) vSignal(500) vOnce(0)
                vResultLow(2) vResultHigh(8) vResultEnd(0) vTrigger(1)]
```

2.3.10.2 Function Counter

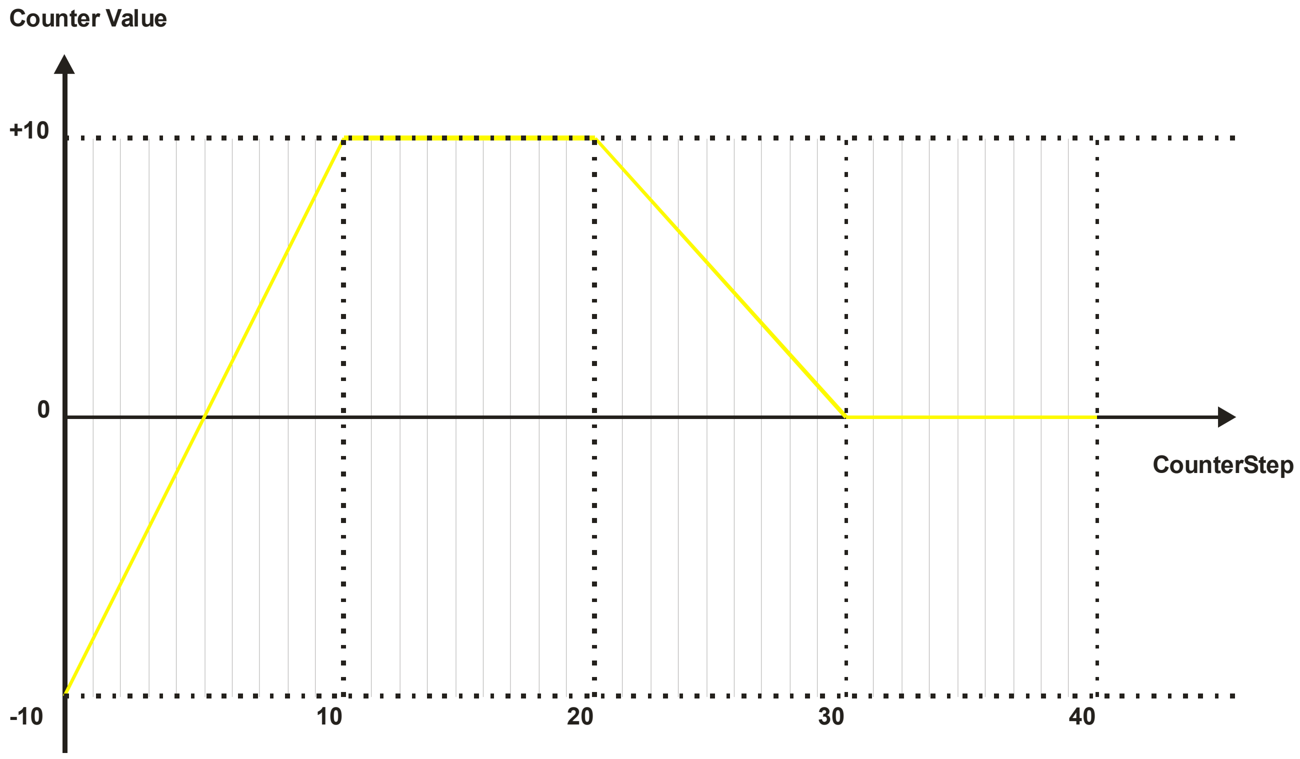
The counter function is defined by the parameter order FunctionName, IntervalCount, RepeatCount, TriggerFlag, and upto 5 positions. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v). The position parameter is define with X and Y steps. The counter function makes use of a matrix which allows a wide range of use. The matrix is defined by counter positions (x/y). The gap between 2 positions is divided into steps given from the x distance:

e.g. pos1 = (0, -10) pos2 = (10, 10) -> 10 Steps resulting (counter = -10, -8, -6, -4, -2, 0, 2, 4, 6, 8, final 10)

Counters may be used as periodic or single-shot counter. The step time of a counter is based on the given sampling time (typically 100 µsec). This time may be multiplied by the parameter IntervalCount. The RepeatCount is the number of the periodically repeation of the matrix. A RepeatCount of -1 means endless repeation.

e.g.

```
var["Trigger"] = const[(1)]
var["Result"]  = function["Counter" valInterval(100) valRepeat(3) ref("Trigger")
                           pos(0, -10) pos(10,10) pos(20,10) pos(30,0)]
```



Note: Calculation of Y_{step} = (long) $\frac{(Y_n - Y_{n-1})}{(X_n - X_{n-1})}$

2.3.10.3 Function Threshold

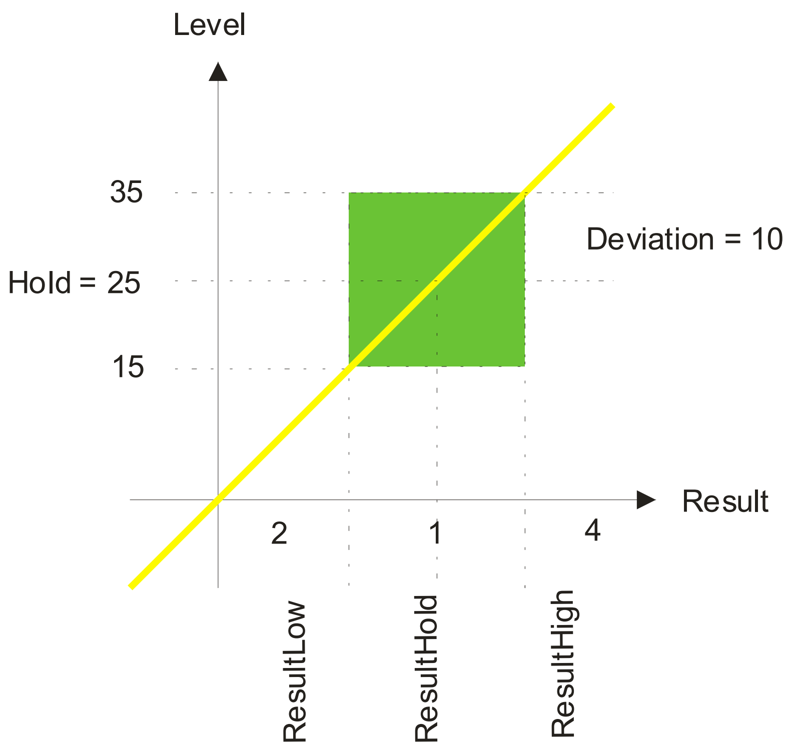
The threshold function is defined by the parameter order FunctionName, Level, HoldLevel, Deviation, ResultUp, ResultDown and ResultHold. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
var["Level"] = in[station (2) data(0) mask(FFFFh)]
var["Result"]  = function["Threshold" ref(“Level”) valHoldLevel(25) valDeviation(10)
                                valResultUp(4), valResultDown(1), valResultHold(2)]
```

Definition:

if (Level < HoldLevel - Deviation)	Result = ResultDown
if (Level > HoldLevel + Deviation)	Result = ResultUp
if (Level >= HoldLevel - Deviation) and (Level <= HoldLevel + Deviation)	Result = ResultHold



2.3.10.4 Function Shift(Left/Right)

The function Shift(Left/Right) is defined by the parameter order FunctionName, Value, and Shift. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
var["Value"] = const[(11223344h)]
var["Result"]  = function["ShiftLeft" ref(“Value”) valShift(8)]
var["Result"]  = function["ShiftRight" ref(“Value”) valShift(16)]
```

Definition: Result = Value << Shift

2.3.10.5 Function Compare

The function Compare is defined by the parameter order FunctionName, Comperand1, Comperand2, CompareMask, ResultEqual, ResultLower and ResultHigher. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
variable["Value1"] = const[(1)]
variable["Value2"] = const[(2)]
var["Result"] = function["Compare" ref("Value1") ref("Value2") valMask((-1)
                             valEqual(4) valLower(5) valHigher(6)]
```

```
Definition:          if ((Value1 & Mask) == (Value2 & Mask)) { Result = valEqual; }
                     if ((Value1 & Mask) <  (Value2 & Mask)) { Result = valLower; }
                     if ((Value1 & Mask) >  (Value2 & Mask)) { Result = valHigher; }
```

2.3.10.6 Function BitSet

The function BitSet is defined by the parameter order FunctionName, Value and Bit. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.
```
variable["Value"] = const[(4)]
var["Result"] = function["BitSet" ref("Value") valBit(1)] ;Result = 6
```

Definition: Result = (Value | (1<<Bit))

2.3.10.7 Function BitReset

The function BitReset is defined by the parameter order FunctionName, Value and Bit. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.
```
variable["Value"] = const[(4)]
var["Result"] = function["BitReset" ref("Value") valBit(2)]       ;Result = 0
```

Definition: Result = (Value & (~(1<<Bit)))

2.3.10.8 Function BitTst

The function BitTst is defined by the parameter order FunctionName, Value, Bit, ResultEqual and ResultNotEqual. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.
```
variable["Value"] = const[(4)]
var["Result"] = function["BitTst" ref("Value") valBit(2) valEqual(4) valNotEqual(5)]
```

```
Definition:     if (Value & (1<<Bit))     { Result = valEqual; }
                else                      { Result = valNotEqual; }
```

2.3.10.9 Function AbsVal

The function AbsVal is defined by the parameter order FunctionName, Value and returns the absolute value. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.
```
variable["Value"] = const[(-4)]
var["Result"] = function["AbsVal" ref("Value")]       ;Result = 4
```

Definition: Result = (ULONG)((Value < 0) ? (-Value) | Value))

2.3.10.10 Function MultiProbe

The function MultiProbe compares several values to each other. If all values are equal it returns the parameter EQAUL, otherwise it returns the parameter NOTEQUAL. At maximum 5 values can be compared to each other. The parameter order is FunctionName, Comperand, Value1, Value2, Value3, Value4, Value5, ValueNum, Equal, NotEqual. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
variable["Value1"] = const[(4)]
variable["Value2"] = const[(4)]
variable["Value3"] = const[(3)]
variable["Value4"] = const[(2)]
var["Result"] = function["MultiProbe" valComperand(4) ref("Value1") ref("Value2")
            ref("Value3") ref("Value4") valEmpty(0) ValNum(4) valEqual(1) valNotE-
            qual(0)]

;Result = 0
```

```
Definition:     Result = for (LONG i=0; i<Num; i++)
                            if (Val[i] != Tst)
                                return NotEqual;

                      return Equal;
```

2.3.10.11 Function Range

The function Range compares a value to its upper and lower limit. If the values is within the range, it returns the parameter ResultHold. If the value is outside the lower limit, it returns the parameter ResultDown. If the value is outside the upper limit it returns the the parameter ResultUp.

The parameter order is FunctionName, Comperand, LimitLow, LimitHigh, ResultUp, ResultDown, ResultHold. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
variable["Level"]     = const[(4)]
variable["LimitLow"]  = const[(2)]
variable["LimitHigh"] = const[(5)]

var["Result"] = function["Range" ref("Level") ref("LimitLow") ref("LimitHigh")
            valResultUp(0) valResultDown(0) valResultHold(1)]

;Result = 1
```

```
Definition:       //Return Exeeding Limit Result
                  if (Level < LimitLow)      { return ResultDown;  }
                  if (Level > LimitHigh)     { return ResultUp; }

                  //Return Hold Result
            return ResultHold;
```

2.4 FlexExtend

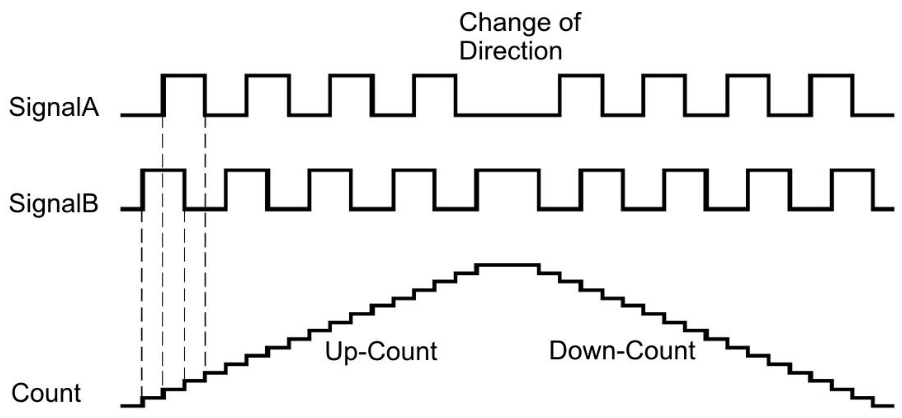
The X-GO Software Development Kit (SDK) allows the extension of the X-GO PLC language with new functional modules. With a simple programming interface new functional modules can be developed and used in the PLC program. For example Drive modules for complex curve control, or encoder modules may be added.

2.4.1 Function Encoder

The encoder function of FlexExtend provides a counter value for the encoder signals A, B and Reset. The encoder function is defined by the parameter order SignalA, SignalB and Reset. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
var["SignalA"] = in[station (2) data(0) mask(1)] ;Get SignalA of incremental encoder
var["SignalB"] = in[station (2) data(0) mask(2)] ;Get SignalB of incremental encoder
var["Reset"]   = in[station (2) data(0) mask(4)] ;Get Reset signal
var["Result"]  = function["Encode" ref(“SignalA”) ref(“SignalB”) ref(“Reset”)]
```



For more information about FlexExtend see the manual of the X-GO Software Development Kit (SDK)

Sample Program:

```
{"Start"}

var["DigIn1"] = in[ station(0.1.0) data(0) mask(FFFFh)]     ;Digital Input
var["DigIn2"] = in[ station(0.2.0) data(0) mask(FFFFh)]     ;Digital Input

variable ["Trigger"]    = const[(1)]
variable ["ResultHigh"] = const[(4)]
variable ["ResultLow"]  = const[(5)]
variable ["ResultEnd"]  = const[(8)]
variable ["Timer1"] = function["Timer" valDelay(1000) valSignal(1000) ref("Once")
                ref("ResultLow") ref("ResultHigh") ref("ResultEnd") ref("Trigger")]

jump["MyLabel"] = var["Timer1"] ? const[(4)]     ;Jump if timer result is 4

var["BufferWait"]  = const[(0)]
var["BufferReset"] = const[(0)]
buffer["Buffer1" wait("BufferWait") reset("BufferReset") size(1000h)] = var["Timer1"]

{"MyLabel"}

var["Counter1"] = function["Counter" valInterval(10) valRepeat(-1) valTrigger(1)
                    pos(0, -10) pos(10,10) pos(20,0)]

out[station(0.1.0) data(0) mask(FFFFh)] = var["Counter1"]
```

2.4.2 Function ConvertEndian

The function “ChangeEndian” changes a value to Big/LittleEndian. The function is defined by the parameter order Value and Size. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
var["Value"]  = in[station (2) data(0) mask(-1)]
var["Result"] = function["ConvertEndian" ref(“Value”) varSize(4)]
```

2.4.3 Function Delay

The function “Delay” implies a timer which is used as functional delay in units of [msec] by returning a binary result value (TRUE / FALSE). Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
var["Reset"] = const[(0)]
var["Done"]  = function["Delay" ref("Reset") valDelay(10000)]
```

2.4.4 DriveEnable

The function “DriveEnable” provides a state machine to bring up the drive into operational mode:

1. Reset Error
2. Enable Voltage
3. SwitchON
4. EnableOP

Therefore it uses the controlword as well as the statusword of the drive.

The parameter order is FunctionName, Reset, Control, Status. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.
```
{"Init"}
var["Reset"] = const[(1)]
var["State"] = var["State"] + const[(1)]
jmp["End"]

{"Enable"}
var["Status"] = in[station(5) data(0) mask(FFFFh)]
var["Done"]   = function["DriveEnable" ref("Reset") ref("Control") ref("Status")]
var["Reset"] = var["Done"]
var["State"] = var["State"] + var["Done"]
jmp["End"]

{"End"}
out[station(5) data(0) mask(FFFFh)]= var["Control"]
```

2.4.5 DriveDisable

The function “DriveDisable” provides a state machine to stop the drive from operational mode. Therefore it uses the controlword as well as the statusword of the drive.
The parameter order is FunctionName, Reset, Control, Status. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.
```
{"Init"}
var["Reset"] = const[(1)]
var["State"] = var["State"] + const[(1)]
jmp["End"]

{"Disable"}
var["Status"] = in[station(5) data(0) mask(FFFFh)]
var["Done"]   = function["DriveDisable" ref("Reset") ref("Control") ref("Status")]
var["Reset"] = var["Done"]
var["State"] = var["State"] + var["Done"]
jmp["End"]

{"End"}
out[station(5) data(0) mask(FFFFh)]= var["Control"]
```

2.4.6 DriveHoming

The drive homing procedure is required to provide a reference position, for all further movements. Typically, several homing methods are available, described in the drive manual of the vendor. The function “DriveHoming” brings the drive to a reference position, due to its PDO’s:

Index	SubIndex	Name
0x6098	0	Homing Method
0x6060	0	Modes of Operation

Therefore it uses the controlword as well as the statusword of the drive with BitMasks and Flags.
The parameter order is FunctionName, Reset, Control, Status, BitMask, Flags. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
{"Init"}
var["Reset"] = const[(1)]
var["State"] = var["State"] + const[(1)]
jmp["End"]

{"Disable"}
var["Status"] = in[station(5) data(0) mask(FFFFh)]
var["Done"]   = function["DriveHoming" ref("Reset") ref("Control_EMCA_X")
            ref("Status_EMCA_X") valBitMask(00009400h) valFlags(00009400h)]
var["Reset"]  = var["Done"]
var["State"]  = var["State"] + var["Done"]
jmp["End"]

{"End"}
out[station(5) data(0) mask(FFFFh)]= var["Control"]
```

2.4.7 DriveJog

The function “DriveJog” moves the drive in Synchronous Cyclic Position mode safely (with limit check).
The parameter order is FunctionName, Control, Status, ActualPos, TargetPos, Increment, MinPos and MaxPos.
Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

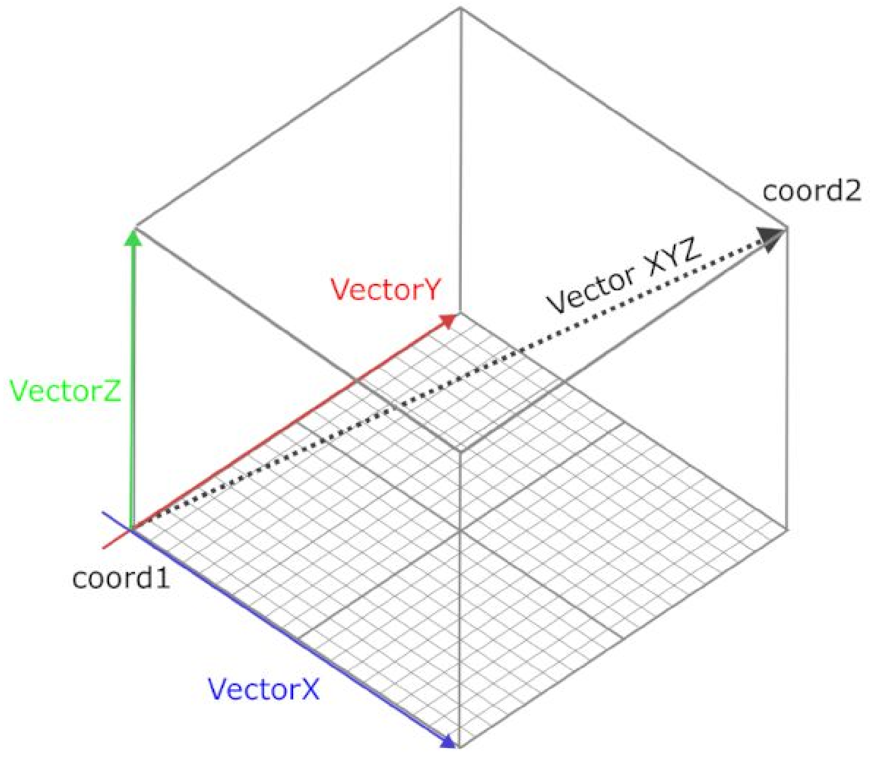
```
var["Status_EMCA_X"]    = in[station(2) data(0) mask(FFFFh)]
var["ActualPos_EMCA_X"] = in[station(2) data(2) mask(-1)]

var["Done"] = function["DriveJog" ref("Control_EMCA_X") ref("Status_EMCA_X")
            ref("ActualPos_EMCA_X") ref("TargetPos_EMCA_X") ref("Inc_EMCA_X")
            valMinPos(-100) valMaxPos(300000)]

{"End"}
out[station(2) data(0) mask(FFFFh)]= var["Control"]
out[station(2) data(3) mask(-1)]   = var["TargetPos_EMCA_X"]
```

2.4.8 DriveIterate

One of the biggest challenges in motion control is the interpolation of coordinate points. In order to get a handle on this challenge, the initial situation must first be considered.

In 3-dimensional space, 2 coordinates describe a vector that defines a smooth movement for the X/Y/Z-axes, so that the tool head is moved from coordinate 1 to coordinate 2 in a smooth linear movement (Note: in addition to linear path descriptions, the GCODE can also describe curved paths). The speed of the respective axis movement is thus also defined via the spatial vector.

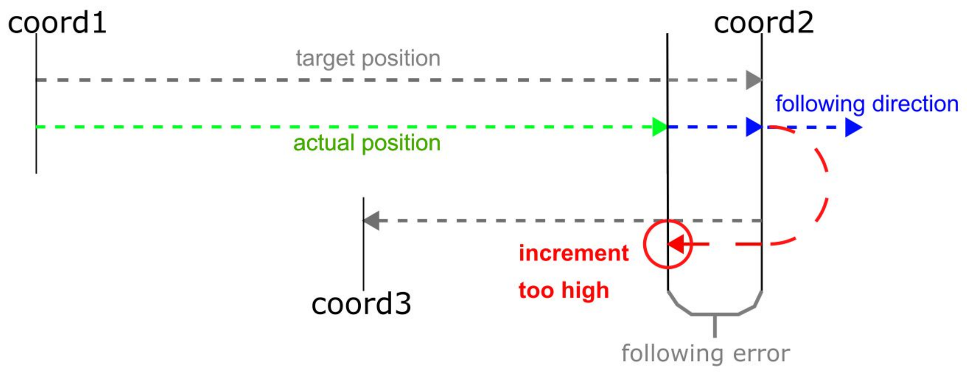
In the cyclic synchronous position mode of a drive, the axis movement between 2 coordinate points is divided into iteration steps (waypoints). The speed of the drive is defined by increments (the larger the increment, the higher the speed). Since the vector of two 3D coordinates determines the speed of the overall movement (of the tool head), partial vectors must be formed for each axis in order to ensure a coordinated linear movement of all axes to the target coordinate. This partial vectors are calculated using trigonometric arithmetic operations, which are carried out either statically (in advance) or dynamically (during the course of the process).



Since the number of iteration steps of a vector of two 3D coordinates also applies to all partial vectors, the iteration step size (increment) between the coordinate points can now be calculated for each axis. This results in a separate list of points (waypoints) for each axis, which describes the coordinated movement of the axis to the target point. A 3D model can contain several million waypoints per axis (depending on the resolution). This can require enormous amounts of memory.

Therefore, the dynamic calculation of the waypoints using predefined iterations makes more sense, since the calculation can be done using simple arithmetic operations. Thus, the coordinate tables per axis are created in advance, but the iteration steps are calculated dynamically.

The problem, however, is that an axle drive is a highly dynamic mechanism in which the moments of inertia must also be taken into account. If a movement is started, a distance (following error) is formed between the desired position (target position) and the actual position (actual position) due to the mass inertia, which, like a rubber cable, must be tracked by the drive control. If a path is now driven between 2 coordinate points via the iteration points (waypoints), the dynamic distance between the target and actual position is at its highest value when the target speed is reached, which is then reduced again in the course of the uniform movement.



If the movement between 2 coordinate points is continued in the same direction to the following coordinate points, the dynamic distance between the target and actual position remains almost constant. On the other hand, when changing direction, the dynamic distance can increase, since the target position is incremented in the opposite direction. The resulting following error can now exceed the dynamic limits of the drive and thus put the drive in an error state.

A change of direction must therefore be taken into account when calculating the subsequent waypoints. The dynamic limits of the drive (tolerances) must be known. If the tolerance specifications are too low, there will be "choppy" movements due to constant movement standstill, if the tolerances are set too high, the movement will become smoother, but the following error can exceed the dynamic limits.

The function “DriveIterate” is a powerful kinetic calculation between 2 points to move in Synchronous Cyclic Position mode. It calculates from given iteration steps (iteration vector) the increment of each move, by considering the following error.

The parameter order is FunctionName, Control, Status, ActualPos, TargetPos, StartPos, FinalPos, Iteration, Tolerance, Reset. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
var["Status_EMCA_X"]     = in[station(2) data(0) mask(FFFFh)]
var["ActualPos_EMCA_X"] = in[station(2) data(2) mask(-1)]

var["DoneX"] = function["DriveIterate" ref("Control_EMCA_X") ref("Status_EMCA_X")
              ref("ActualPos_EMCA_X") ref("TargetPos_EMCA_X") ref("Last_Coord_XAxis")
              ref("Coord_XAxis") ref("Coord_Iteration") valTolerance(50)
              ref("IterationReset")]

{"End"}
out[station(2) data(0) mask(FFFFh)]= var["Control"]
out[station(2) data(3) mask(-1)]    = var["TargetPos_EMCA_X"]
```

2.4.1 DriveJog

The function “DriveJog” moves the drive in Synchronous Cyclic Position mode safely (with limit check).
The parameter order is FunctionName, Control, Status, ActualPos, TargetPos, Increment, MinPos and MaxPos.
Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
var["Status_EMCA_X"]    = in[station(2) data(0) mask(FFFFh)]
var["ActualPos_EMCA_X"] = in[station(2) data(2) mask(-1)]

var["Done"] = function["DriveJog" ref("Control_EMCA_X") ref("Status_EMCA_X")
            ref("ActualPos_EMCA_X") ref("TargetPos_EMCA_X") ref("Inc_EMCA_X")
            valMinPos(-100) valMaxPos(300000)]

{"End"}
out[station(2) data(0) mask(FFFFh)]= var["Control"]
out[station(2) data(3) mask(-1)]   = var["TargetPos_EMCA_X"]
```

2.4.2 PneumaticMotion

The function “PneumaticMotion” is used to drive a pneumatics cylinder due to a proportional valve. Therefore the parameter “ParamDist” is measured by a sensor, while the parameter “ParamProp” drives a proportional valve. The parameter mode defines the motion ramp (1: linear, 2: progressive, 3: degressive), defined by 3 points:

x1 , pressure1
X2 , pressure2
X3 , pressure3

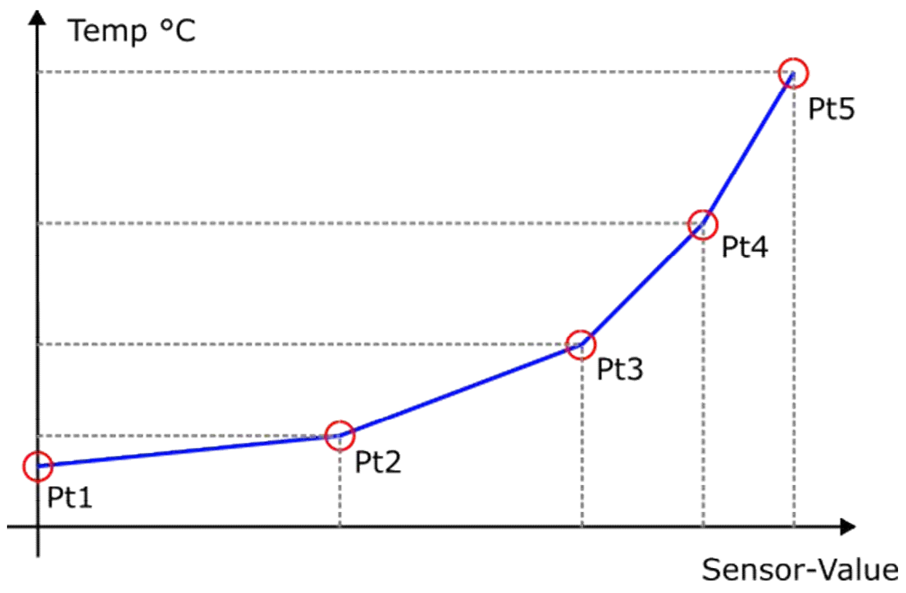
Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

e.g.

```
var["ParamDist"] = in[station(3) data(5) mask(FFh)]
var["ParamProp"]
var["Done"]  = function["PneumaticMotion" ref("Reset") valMode(1) ref("ParamDist")
                                          pos1(20,5500) pos2(15,4000) pos3(10,5000)
                                          ref("ParamProp")]

out[station(5) data(2) mask(FFFFh)] = var["ParamProp"]
```

2.4.3 GetTemperatureLinear

The function „GetTemperatureLinear“ gives back a temperature value defined by a linear curve, described with 5 coordinates (x/y).



e.g. MUH-HS-U PT100

```
var["Result"] = function["GetTemperatureLinear" ref("AnalogIn ") ref("ActualTemperature")
                 p1(0h,0) p2(1FFFh,62) p3(3FFFh,124) p4(5FFFh,186) p5(7FFFh,248)]
```

The parameter order is FunctionName, AnalogValue, TemperatureValue, Point1, Point2, Point3, Point4, Point5. Each parameter may be a reference to a variable (prefix r), or a constant value (prefix v).

2.5 FlexExchange

To make use of the FlexExchange Module the X-GO Software Development Kit (SDK) is required. The FlexExchange module allows easily connecting external applications to the X-GO PLC program. The whole logic stack is accessible via shared memory, so that PLC data can be processed directly with your C # or C + + program, via exported PLC variables and multiple buffers.

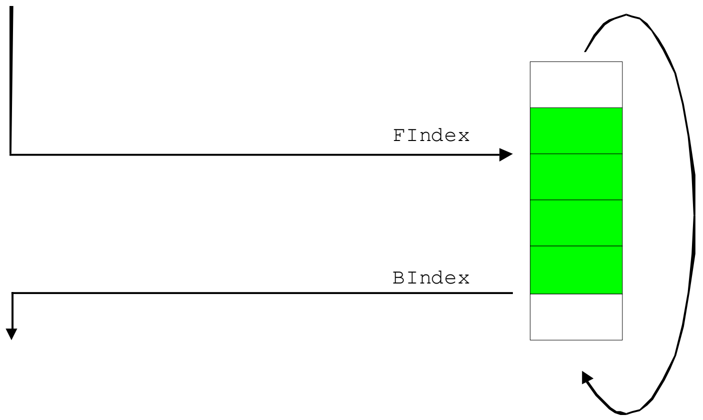
2.5.1 Buffer Exchange

Buffers can be used to share information with the programming interface between PLC program and any outside application. Therefore 2 interface functions are available:

```
Xgo32/64AttachToBuffer("Buffer1", &pBuffer, &hBuffer, &hDesc);
Xgo32/64AttachToBuffer("Buffer2", &pBuffer, &hBuffer, &hDesc);
```

```
Xgo32/64SetBufferBlock(pBuffer, pBlock, BlockLen, FALSE, FALSE, hDesc, &ValuesWritten)
```

```
var["Buffer2"] = in[station(2) data(0) mask(-1)]
var["BufferWait"] = c[(0)]
var["BufferReset"]= c[(0)]
buffer["Buffer2" wait("BufferWait") reset("BufferReset") cond("CondFlag") size(1000h)] = var["Buffer2"]
```



```
Xgo32/64GetBufferBlock(pBuffer, pBlock, BlockLen, FALSE, FALSE, hDesc, &ValuesRead)
```

```
var["BufferWait"] = c[(0)]
var["BufferReset"]= c[(0)]
var["Buffer1"]= buffer["Buffer1" wait("BufferWait") reset("BufferReset") cond("CondFlag") size(1000h)]
out[station(1) data(0) mask(-1)] = var["Buffer1"]
```

```
Xgo32/64DetachFromBuffer(hBuffer);
Xgo32/64DetachFromBuffer(hBuffer);
```

e.g. set value to buffer

```
var["BufferVal"]  = const [(11223344h)]
var["BufferWait"] = const [(0)]
var["BufferReset"]= const [(0)]
var["BufferCond"]

buffer["InBuffer" wait("BufferWait")
                  reset("BufferReset")
                  cond("BufferCond")
                  size(10000h)] = var["BufferVal"]
```

e.g. get value from buffer

```
var["BufferVal"]
var["BufferWait"] = const [(0)]
var["BufferReset"]= const [(0)]
var["BufferCond"]

var["BufferVal"]= buffer["OutBuffer" wait("BufferWait")
                                     reset("BufferReset")
                                     cond("CondFlag")
                                     size(1000h)]
```

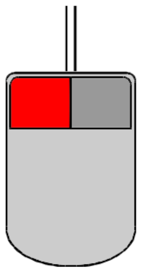
2.5.2 Buffer Synchronization

With the BufferWait flag buffers can be synchronized between PLC program and outside application. Also the buffer can be resetted from both sides using the flag BufferReset. The condition indicates following buffer states:

- BUFFER_FULL
- BUFFER_BUSY
- BUFFER_EMPTY

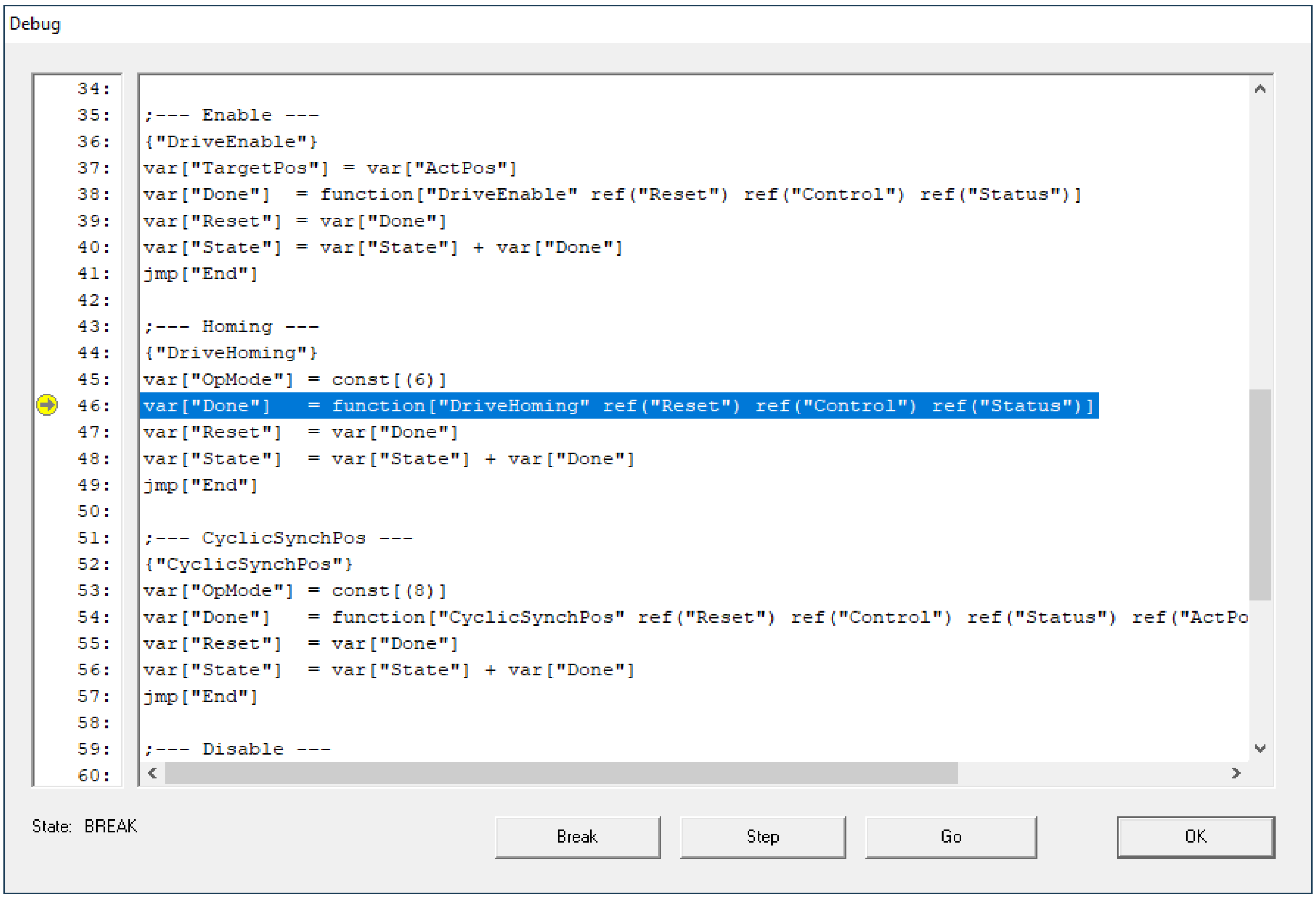
2.6 FlexDebug

The FlexDebug allows breakpoints and single line execution of the PLC program. Each program line will be evaluated separately, while its result is updated inside FlexMonitor and FlexPanel. Flex Debug allows constant repositioning of the current execution line. When the program is running, it's interrupted on pressing the BREAK button. On BREAK, you can change or display values associated to FlexPanel elements.



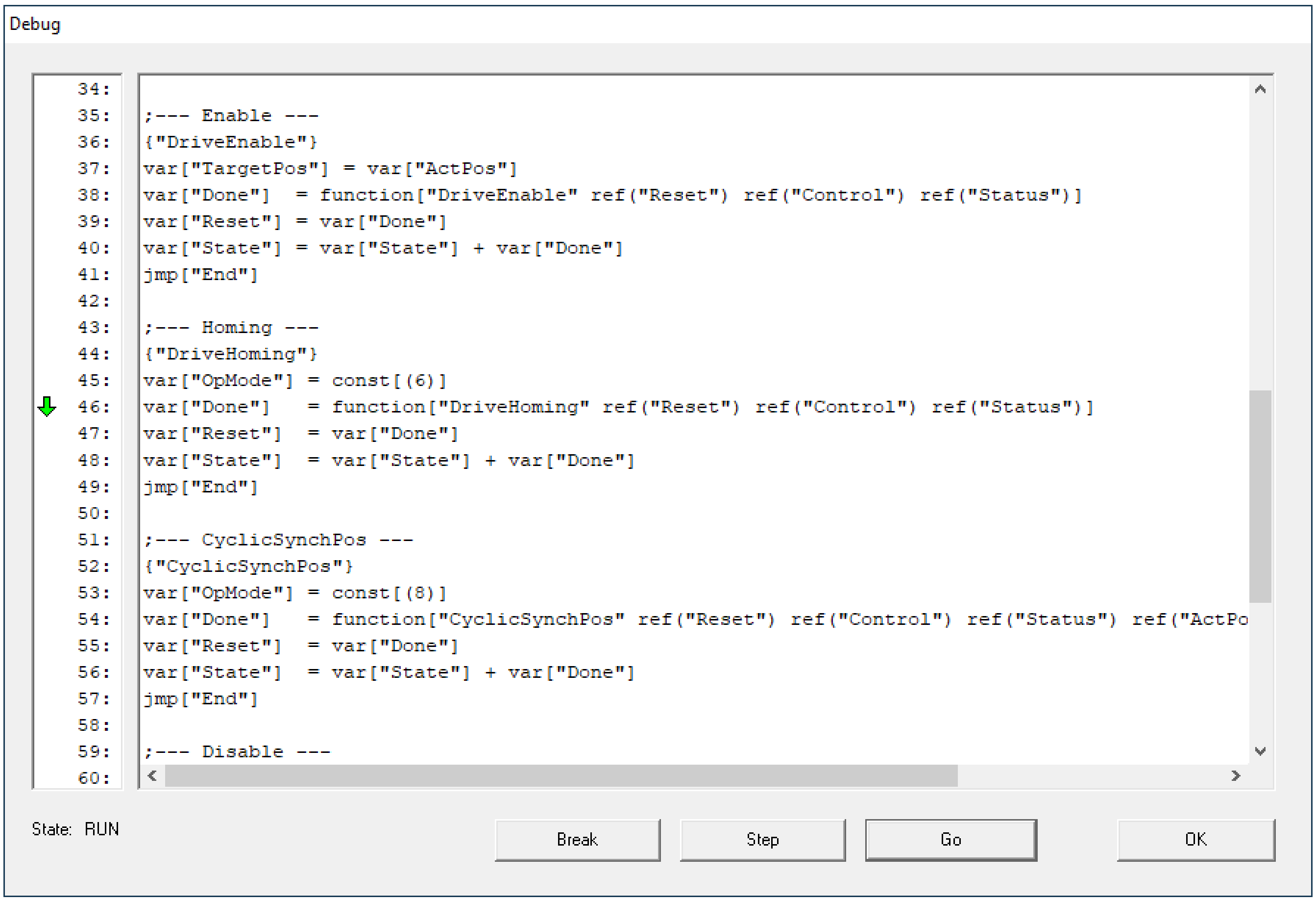
2.6.1 Setting a Breakpoint

A breakpoint is set on pressing the left mouse button at the selected line. Press the GO button to run the program until the breakpoint is reached



```
;--- Enable ---
{"DriveEnable"}
var["TargetPos"] = var["ActPos"]
var["Done"]  = function["DriveEnable" ref("Reset") ref("Control") ref("Status")]
var["Reset"] = var["Done"]
var["State"] = var["State"] + var["Done"]
jmp["End"]

;--- Homing ---
{"DriveHoming"}
var["OpMode"] = const[(6)]
var["Done"]   = function["DriveHoming" ref("Reset") ref("Control") ref("Status")]
var["Reset"]  = var["Done"]
var["State"]  = var["State"] + var["Done"]
jmp["End"]

;--- CyclicSynchPos ---
{"CyclicSynchPos"}
var["OpMode"] = const[(8)]
var["Done"]   = function["CyclicSynchPos" ref("Reset") ref("Control") ref("Status") ref("ActPo
var["Reset"]  = var["Done"]
var["State"]  = var["State"] + var["Done"]
jmp["End"]

;--- Disable ---
```

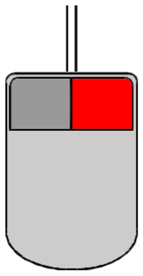
State: BREAK



```
;--- Enable ---
{"DriveEnable"}
var["TargetPos"] = var["ActPos"]
var["Done"]  = function["DriveEnable" ref("Reset") ref("Control") ref("Status")]
var["Reset"] = var["Done"]
var["State"] = var["State"] + var["Done"]
jmp["End"]

;--- Homing ---
{"DriveHoming"}
var["OpMode"] = const[(6)]
var["Done"]   = function["DriveHoming" ref("Reset") ref("Control") ref("Status")]
var["Reset"]  = var["Done"]
var["State"]  = var["State"] + var["Done"]
jmp["End"]

;--- CyclicSynchPos ---
{"CyclicSynchPos"}
var["OpMode"] = const[(8)]
var["Done"]   = function["CyclicSynchPos" ref("Reset") ref("Control") ref("Status") ref("ActPo
var["Reset"]  = var["Done"]
var["State"]  = var["State"] + var["Done"]
jmp["End"]

;--- Disable ---
```

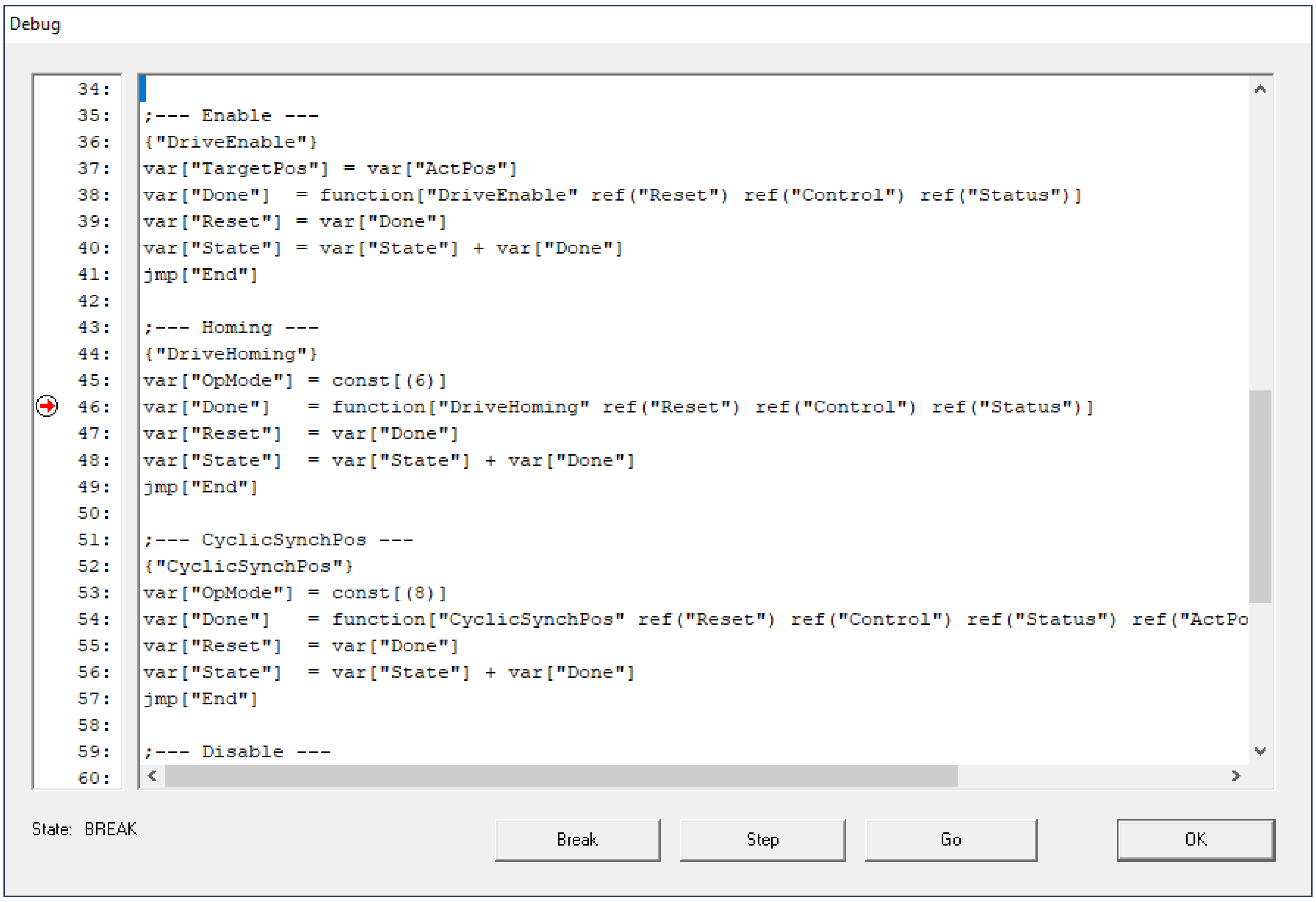
Note:

When pressing the STEP button the line will be executed. Comment lines are stepped as NOP command



2.6.2 Setting an execution line

When pressing the right mouse button at any line, the execution will be set to this line

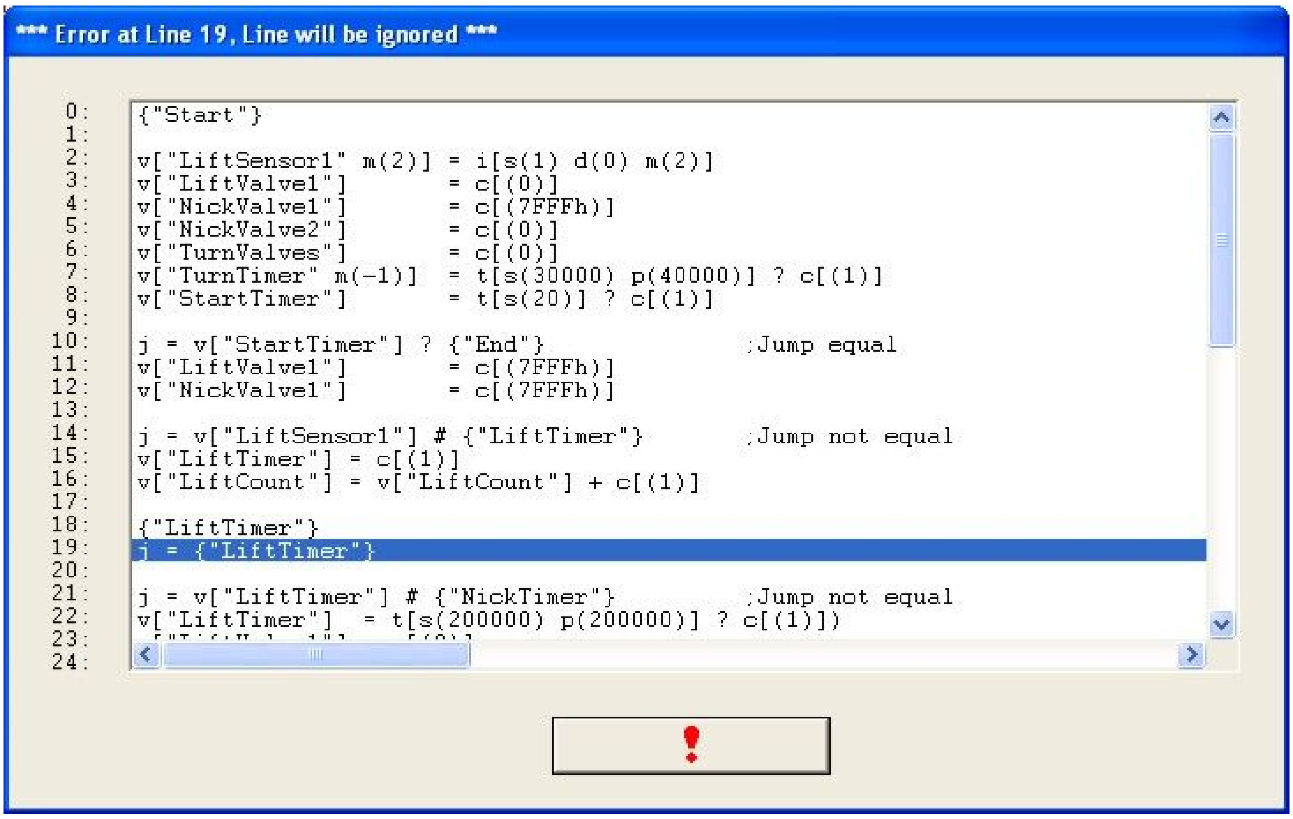


```

;--- Enable ---
{"DriveEnable"}
var["TargetPos"] = var["ActPos"]
var["Done"]  = function["DriveEnable" ref("Reset") ref("Control") ref("Status")]
var["Reset"] = var["Done"]
var["State"] = var["State"] + var["Done"]
jmp["End"]

;--- Homing ---
{"DriveHoming"}
var["OpMode"] = const[(6)]
var["Done"]   = function["DriveHoming" ref("Reset") ref("Control") ref("Status")]
var["Reset"]  = var["Done"]
var["State"]  = var["State"] + var["Done"]
jmp["End"]

;--- CyclicSynchPos ---
{"CyclicSynchPos"}
var["OpMode"] = const[(8)]
var["Done"]   = function["CyclicSynchPos" ref("Reset") ref("Control") ref("Status") ref("ActPo
var["Reset"]  = var["Done"]
var["State"]  = var["State"] + var["Done"]
jmp["End"]

;--- Disable ---
```

State: BREAK

2.6.3 Logic Check

X-GO provides a PLC program check while stepping. Thus, e.g. endless loops will be recognized.



*** Error at Line 19, Line will be ignored ***

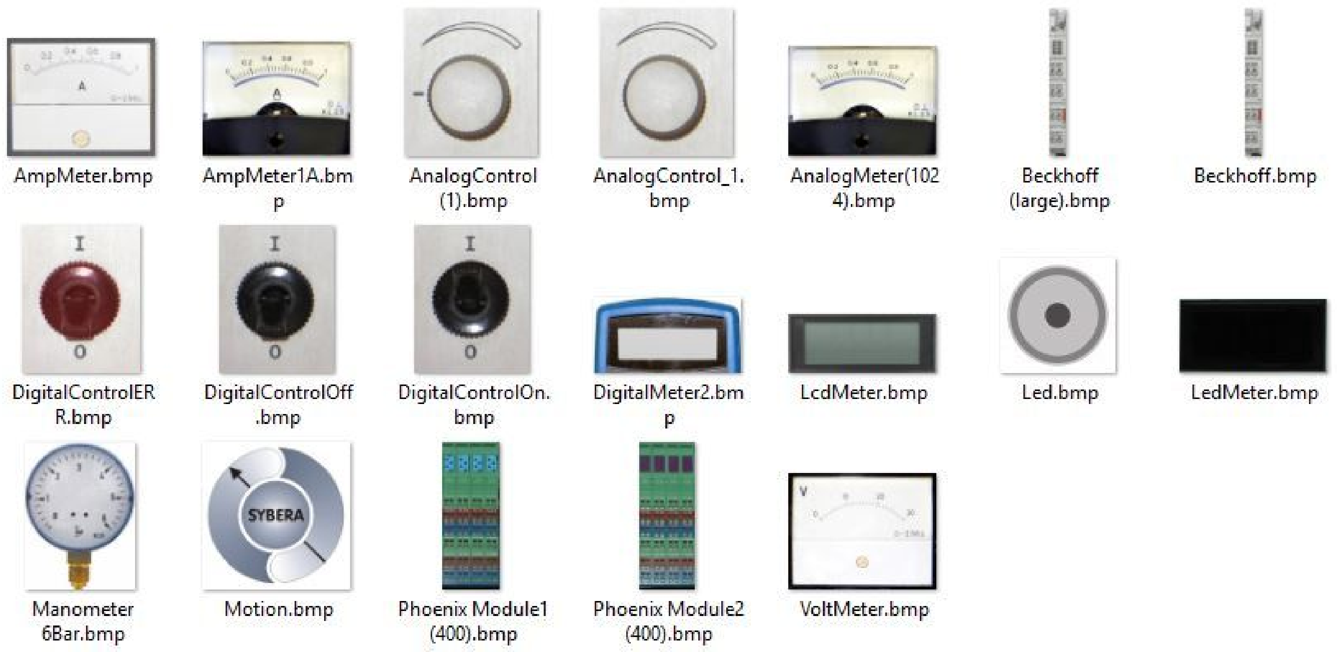
```
 0: {"Start"}
 1:
 2: v["LiftSensor1" m(2)] = i[s(1) d(0) m(2)]
 3: v["LiftValve1"]       = c[(0)]
 4: v["NickValve1"]       = c[(7FFFh)]
 5: v["NickValve2"]       = c[(0)]
 6: v["TurnValves"]       = c[(0)]
 7: v["TurnTimer" m(-1)]  = t[s(30000) p(40000)] ? c[(1)]
 8: v["StartTimer"]       = t[s(20)] ? c[(1)]
 9:
10: j = v["StartTimer"] ? {"End"}               ;Jump equal
11: v["LiftValve1"]       = c[(7FFFh)]
12: v["NickValve1"]       = c[(7FFFh)]
13:
14: j = v["LiftSensor1"] # {"LiftTimer"}        ;Jump not equal
15: v["LiftTimer"] = c[(1)]
16: v["LiftCount"] = v["LiftCount"] + c[(1)]
17:
18: {"LiftTimer"}
19: j = {"LiftTimer"}
20:
21: j = v["LiftTimer"] # {"NickTimer"}          ;Jump not equal
22: v["LiftTimer"]  = t[s(200000) p(200000)] ? c[(1)])
23:
24:
```

The error line will be ignored when pressing the button twice.

2.7 FlexPanel (HmiDesignBox)

The Flex Panel module allows the photorealistic design and integration of machine panel instruments and controls, like meters, switches, modules and backgrounds - without programming. You can use images and graphics of your machine instruments, overlaid by corresponding software templates and simply linked to PLC program variables. The overlay of the software templates is carried out through simple adjustment of the parameter files. Additionally new templates can be programmed and integrated with the X-GO Software Development Kit (SDK).

Pre-defined panel images:



Pre-defined panel functions:

- AnalogMeter
- DigitalMeter
- LinearMeter
- AnalogControl
- DigitalControl
- DigitalModule
- GenericGraph
- Monitor
- Motion

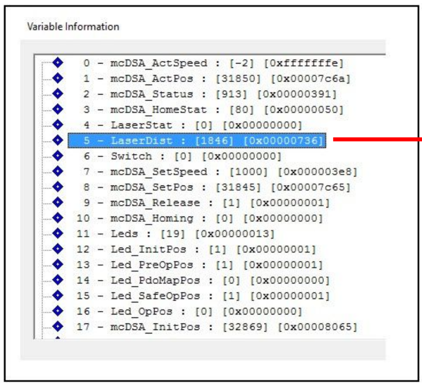
Sample: Parameter File for AnalogMeter Panel:

```
String [Images\\VoltMeter.bmp] //Image file
String [AnalogMeter]           //Function name
String [ ]                     //Unit name
Value [0]           //ShowTitle;
Value [0]           //MinVal;
Value [30]          //MaxVal;
Value [1000]        //Scale;
Value [203]         //MagXCentre;
Value [240]         //MagYCentre;
Value [0]           //MagXOffs;
Value [60]          //MagYOffs;
Value [180]         //MagLen;
Value [45]          //AngleOffs;
Value [88]          //AngleSpan;
Value [0]           //UnitXOffs
Value [0]           //UnitYOffs
Value [200]         //UnitFontSize
Value [0x00646464]  //ColorOk
Value [0x000000FF]  //ColorErr
```

2.7.1 Panel parameter

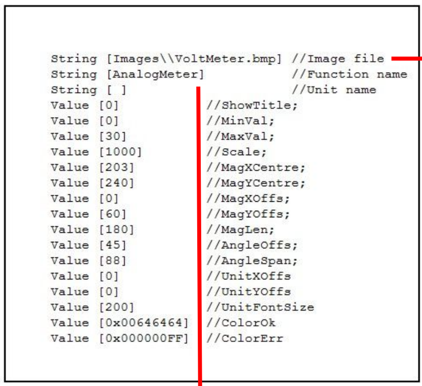
To overlay a software panel template on a given image, simply adjust the parameter set. This can be done with any editor while X-GO is running. The default parameters set files are stored within the directory \Params, while corresponding images are stored within \Params\Images as files of Windows bitmap type BMP. To adjust the template element positions, like magnitude, unit offset, font size, by pressing the key [F5] the coordinates are shown in the left-top corner inside the panel dialog. When pressing the key [F4] the movement of the panel is adjusted to a grid.

Variable Information Dialog



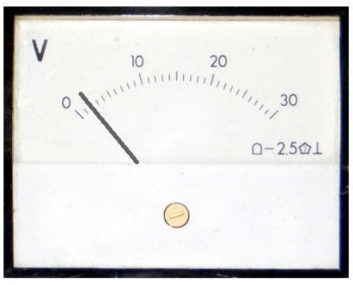
```
Variable Information

0 - mcDSA_ActSpeed : [-2] [0xfffffffe]
1 - mcDSA_ActPos : [31850] [0x00007c6a]
2 - mcDSA_Status : [913] [0x00000391]
3 - mcDSA_HomeStat : [80] [0x00000050]
4 - LaserStat : [0] [0x00000000]
5 - LaserDist : [1846] [0x00000736]
6 - Switch : [0] [0x00000000]
7 - mcDSA_SetSpeed : [1000] [0x000003e8]
8 - mcDSA_SetPos : [31845] [0x00007c65]
9 - mcDSA_Release : [1] [0x00000001]
10 - mcDSA_Homing : [0] [0x00000000]
11 - Leds : [19] [0x00000013]
12 - Led_InitPos : [1] [0x00000001]
13 - Led_PreOpPos : [1] [0x00000001]
14 - Led_PdoMapPos : [0] [0x00000000]
15 - Led_SafeOpPos : [1] [0x00000001]
16 - Led_OpPos : [0] [0x00000000]
17 - mcDSA_InitPos : [32869] [0x00008065]
```

Parameter File



```
String [Images\\VoltMeter.bmp] //Image file
String [AnalogMeter]           //Function name
String [ ]                     //Unit name
Value [0]           //ShowTitle;
Value [0]           //MinVal;
Value [30]          //MaxVal;
Value [1000]        //Scale;
Value [203]         //MagXCentre;
Value [240]         //MagYCentre;
Value [0]           //MagXOffs;
Value [60]          //MagYOffs;
Value [180]         //MagLen;
Value [45]          //AngleOffs;
Value [88]          //AngleSpan;
Value [0]           //UnitXOffs
Value [0]           //UnitYOffs
Value [200]         //UnitFontSize
Value [0x00646464]  //ColorOk
Value [0x000000FF]  //ColorErr
```

Image File

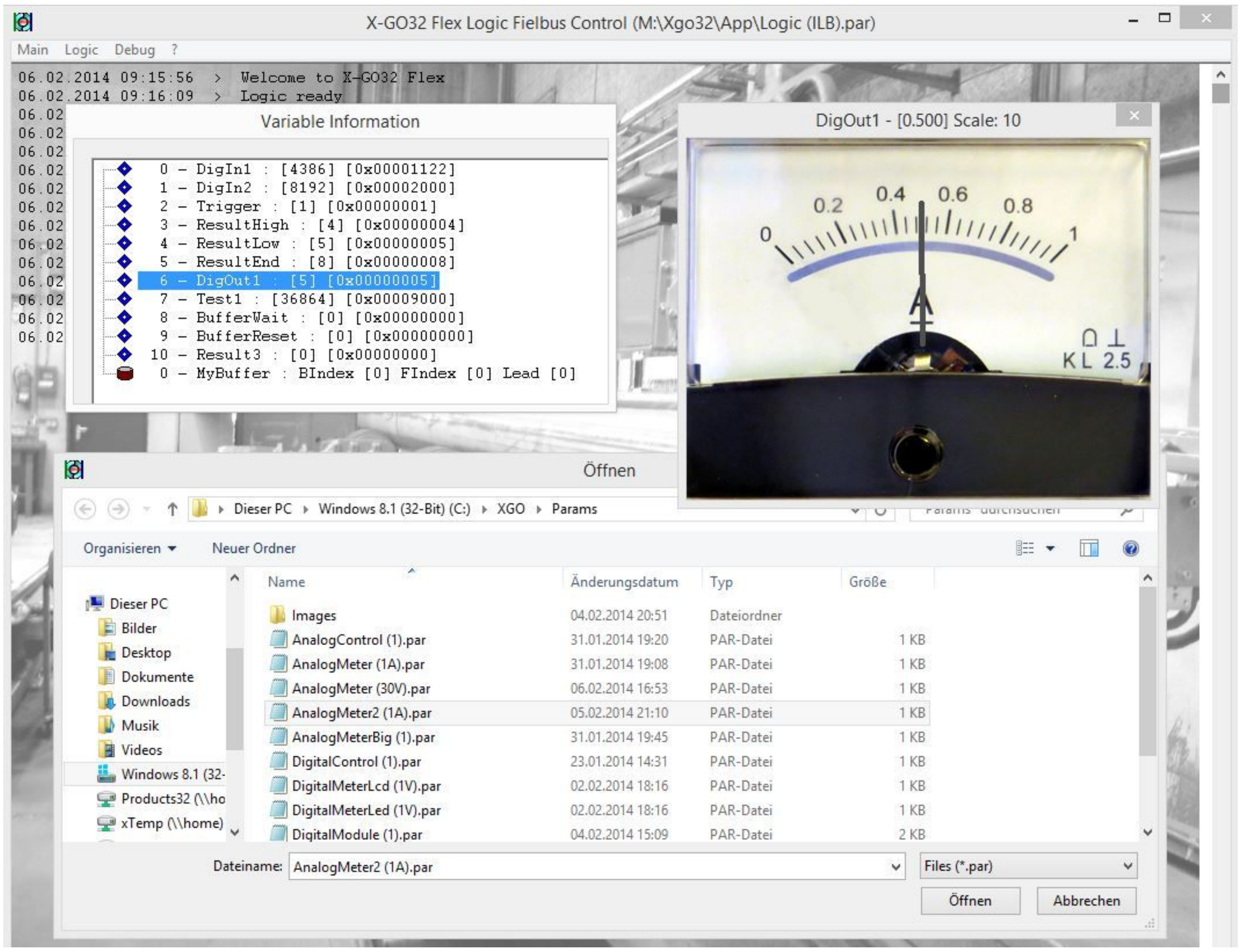


XgoPanelDll64.DLL

Panel Function DLL

2.7.2 Panel selection

For selecting a panel just right-click on a variable inside the Variable Information dialog.



```
06.02.2014 09:15:56  >  Welcome to X-GO32 Flex
06.02.2014 09:16:09  >  Logic ready
```

```
 0 - DigIn1 : [4386] [0x00001122]
 1 - DigIn2 : [8192] [0x00002000]
 2 - Trigger : [1] [0x00000001]
 3 - ResultHigh : [4] [0x00000004]
 4 - ResultLow : [5] [0x00000005]
 5 - ResultEnd : [8] [0x00000008]
 6 - DigOut1 : [5] [0x00000005]
 7 - Test1 : [36864] [0x00009000]
 8 - BufferWait : [0] [0x00000000]
 9 - BufferReset : [0] [0x00000000]
10 - Result3 : [0] [0x00000000]
 0 - MyBuffer : BIndex [0] FIndex [0] Lead [0]
```

Name	Änderungsdatum	Typ	Größe
Images	04.02.2014 20:51	Dateiordner	
AnalogControl (1).par	31.01.2014 19:20	PAR-Datei	1 KB
AnalogMeter (1A).par	31.01.2014 19:08	PAR-Datei	1 KB
AnalogMeter (30V).par	06.02.2014 16:53	PAR-Datei	1 KB
AnalogMeter2 (1A).par	05.02.2014 21:10	PAR-Datei	1 KB
AnalogMeterBig (1).par	31.01.2014 19:45	PAR-Datei	1 KB
DigitalControl (1).par	23.01.2014 14:31	PAR-Datei	1 KB
DigitalMeterLcd (1V).par	02.02.2014 18:16	PAR-Datei	1 KB
DigitalMeterLed (1V).par	02.02.2014 18:16	PAR-Datei	1 KB
DigitalModule (1).par	04.02.2014 15:09	PAR-Datei	2 KB

Dateiname: AnalogMeter2 (1A).par — Files (*.par)

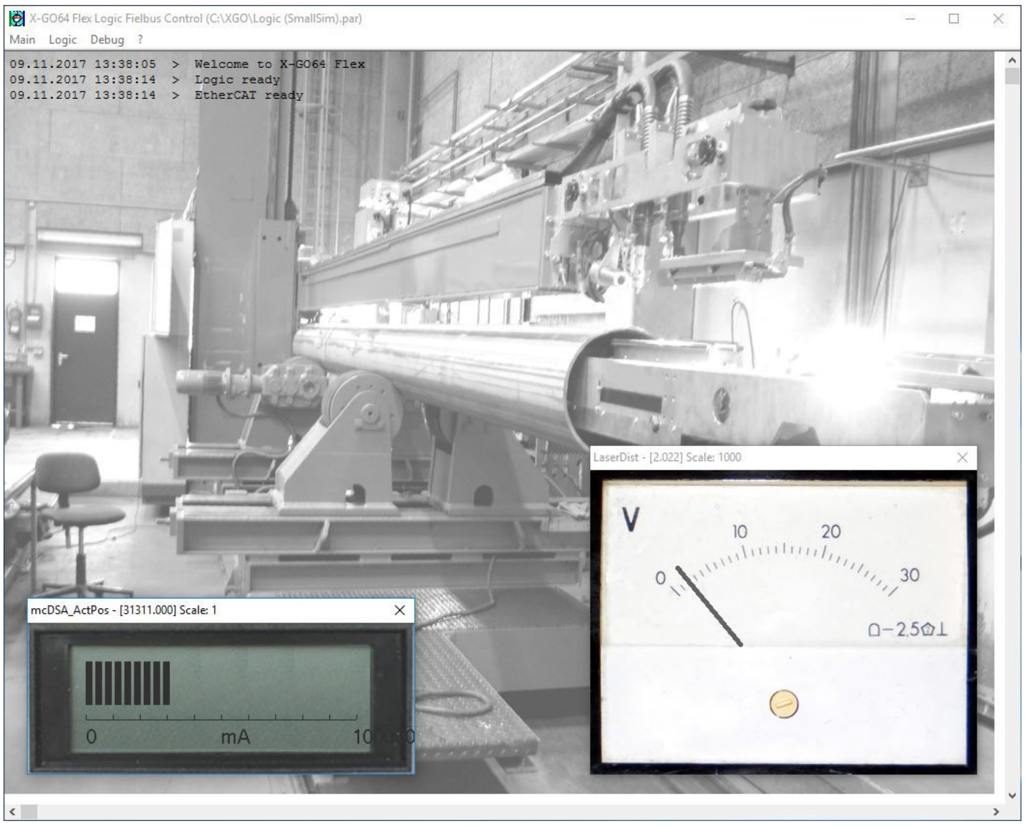
Function Keys:

F4: Lock to Grid. When pressing F4 the selected panel is moving along a grid.
F5: Print Grid Position. When pressing F5, the grid position appears within the selected
F6: Unlock for moving. The selected panel appears at the upper left corner and must be unlocked for moving by pressing the Key F6.
F7: Delete the selected panel
F8: Decrease Z-Order
F9: Increase Z-Order

Note:
For the development of new panel functions, Sybera offers the X-GO Software Development Kit

2.8 Background Image

If an image named “”MAINIMG.BMP” is present within the XGO directory, the image is used as background image for XGO. It may be used as panel background to build up a photo-realistic control desktop in combination with the AUTOSTART feature..


X-GO64 Flex Logic Fielbus Control (C:\XGO\Logic (SmallSim).par)
Main
Logic
Debug
?
LaserDist - [2.022] Scale: 1000
V
0
10
20
30
mcDSA_ActPos - [31311.000] Scale: 1
0
mA
10

```
09.11.2017 13:38:05  >  Welcome to X-GO64 Flex
09.11.2017 13:38:14  >  Logic ready
09.11.2017 13:38:14  >  EtherCAT ready
```

3 X-GO Installation

For installation of the X-GO software following steps are required:

Preparation

1. Provide a PC with INTEL or REALTEK Ethernet adapter and Windows operating system with administrator
2. rights
3. Make shure, the latest Updates are installed (especially Windows 7)
4. Check the installed Ethernet adapter has given a correct IP address
5. Install the MFC redistributables DLLs (in Folder MISC \ REDIST)
6. Install Fonts (in Folder MISC \ FONTS)
7. Install MSXML redistributables (in Folder MISC \ MSXML)

Installation

8. Run the program SYSETUP32/64
9. (make sure the directory path has no space characters)

10. On Installation the PEC information (PID, SERNUM and KEYCODE) must be
11. entered. The SERNUM for the evaluation version is: 12345678,
12. the KEYCODE is: 00001111-22223333

13. Select Network card
14. Reboot the System

Optional: Install network card manually (see chapter 2.1)
Optional: Check license with SYLICENCECHECK(32/64).EXE

Operation

15. Use additional Tools for device configuration

ProfiNET: Use PNIOVERIFY to generate device configuration file

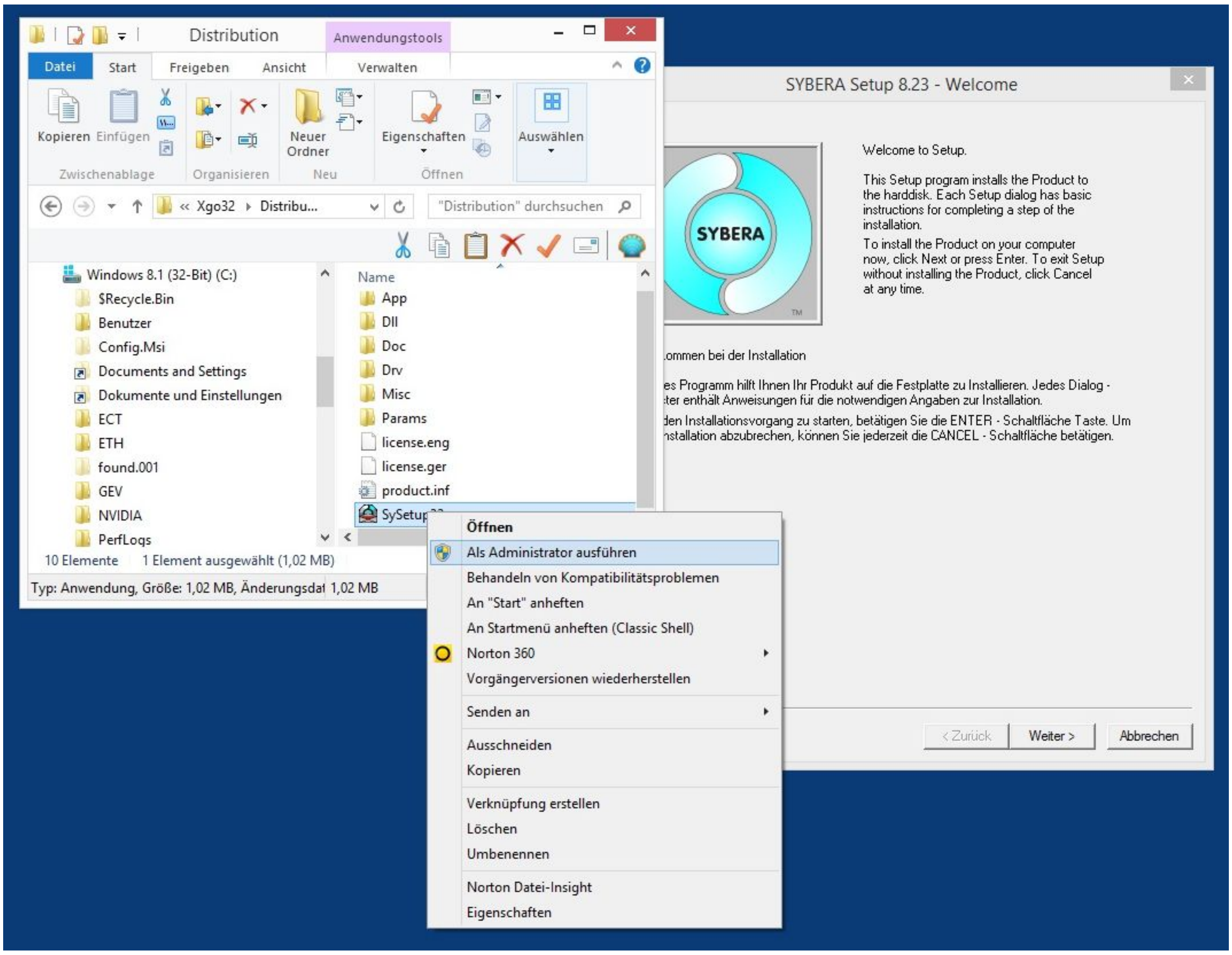
EtherCAT: Use ECATVERIFY to generate device configuration file ECATDEVICE.PAR
(to be placed in \WINDOWS\SYSTEM32)

Ethernet/IP: Use EIPCONFIG to generate device configuration file

Sercos III: Use editor to generate device configuration file SC3CONFIG
(to be placed in \WINDOWS\SYSTEM32)

16. Run X-GO(32/64)

Start installation program SySetup32/64 with administrator privileges


SYBERA Setup 8.23 - Welcome

Welcome to Setup.

This Setup program installs the Product to the harddisk. Each Setup dialog has basic instructions for completing a step of the installation.

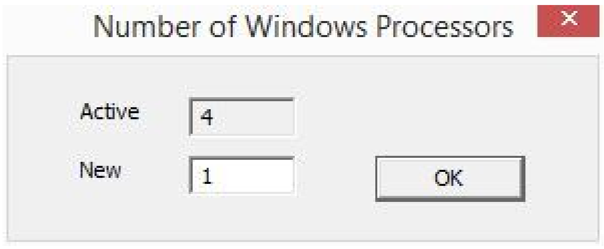
To install the Product on your computer now, click Next or press Enter. To exit Setup without installing the Product, click Cancel at any time.

ommen bei der Installation

es Programm hilft Ihnen Ihr Produkt auf die Festplatte zu Installieren. Jedes Dialog - ter enthält Anweisungen für die notwendigen Angaben zur Installation.

den Installationsvorgang zu starten, betätigen Sie die ENTER - Schaltfläche Taste. Um nstallation abzubrechen, können Sie jederzeit die CANCEL - Schaltfläche betätigen.

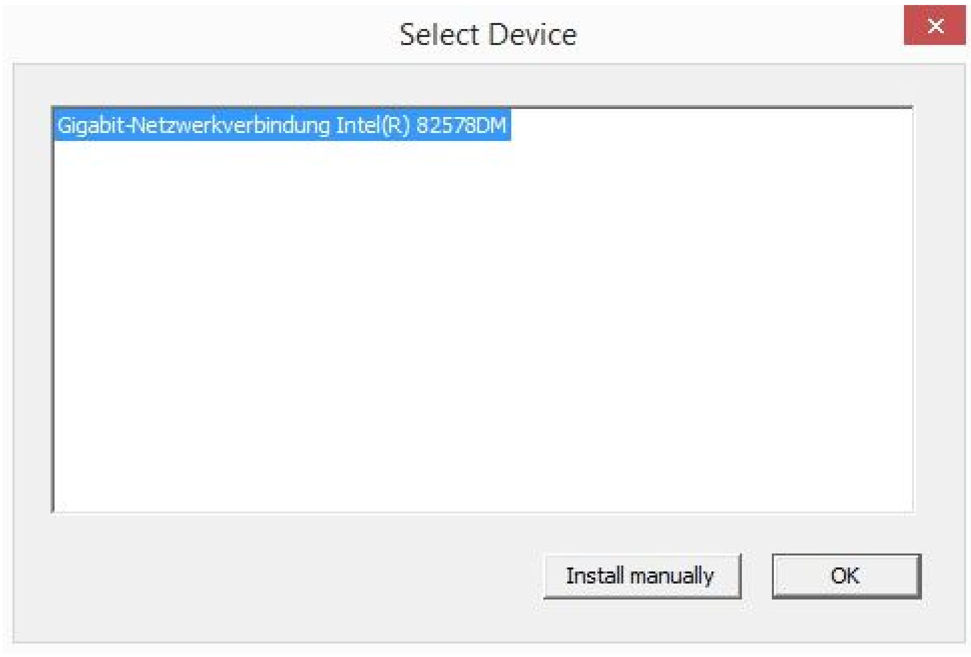
Select windows processor count (typically Active - 1 is best)


Number of Windows Processors

Active: 4

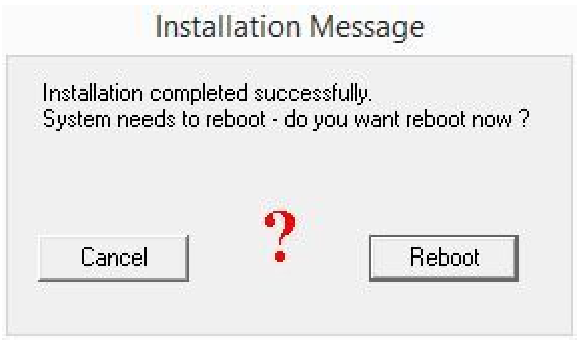
New: 1

Select Ethernet device and quit with button [OK]


Select Device

Gigabit-Netzwerkverbindung Intel(R) 82578DM

Reboot the system


Installation Message

Installation completed successfully.
System needs to reboot - do you want reboot now ?

3.1 BIOS Adjustment

Note:

For proper operation, make sure within the BIOS the INTEL SpeedStep Technology, the INTEL TurboBoost Technology as well as the INTEL C-STATE Technology is turned off.

3.1.1 Disable Intel SpeedStep Technology

Enhanced SpeedStep — SpeedStep also modulates the CPU clock speed and voltage according to load, but it is invoked via another mechanism. The operating system must be aware of SpeedStep, as must the system BIOS, and then the OS can request frequency changes via ACPI. SpeedStep is more granular than C1E halt, because it offers multiple rungs up and down the ladder between the maximum and minimum CPU multiplier and voltage levels.

3.1.2 Disable C-States

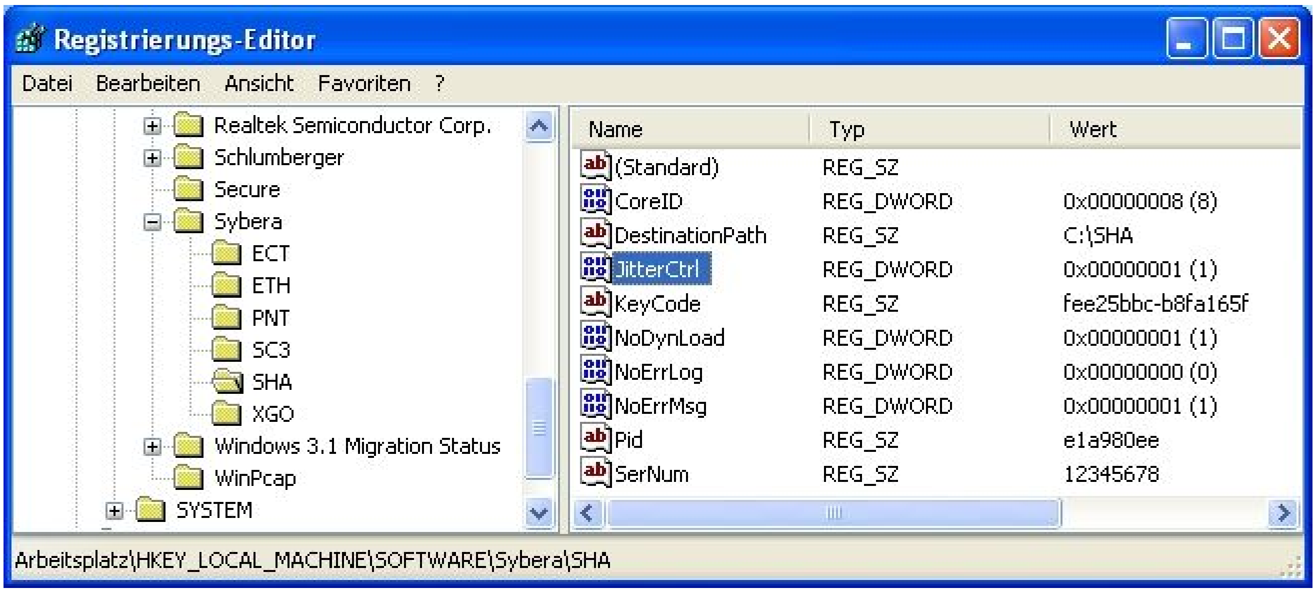
C1E enhanced halt state — Introduced in the Pentium 4 500J-series processors, the C1E halt state replaces the old C1 halt state used on the Pentium 4 and most other x86 CPUs. The C1 halt state is invoked when the operating system's idle process issues a HLT command. (Windows does this constantly when not under a full load.). C0 is the operating state. C1 (often known as Halt) is a state where the processor is not executing instructions, but can return to an executing state essentially instantaneously. All ACPI-conformant processors must support this power state. Some processors, such as the *Pentium 4*, also support an Enhanced C1 state (C1E or Enhanced Halt State) for lower power consumption. C2 (often known as Stop-Clock) is a state where the processor maintains all software-visible state, but may take longer to wake up. This processor state is optional. C3 (often known as Sleep) is a state where the processor does not need to keep its *cache* coherent, but maintains other state. Some processors have variations on the C3 state (Deep Sleep, Deeper Sleep, etc.) that differ in how long it takes to wake the processor. This processor state is optional.

3.1.3 Disable TurboMode

Intel® Turbo Boost Technology automatically allows processor cores to run faster than the base operating frequency, increasing performance. Under some configurations and workloads, Intel® Turbo Boost technology enables higher performance through the availability of increased core frequency. Intel® Turbo Boost technology automatic allows processor cores to run faster than the base operating frequency if the processor is operating below rated power, temperature, and current specification limits. Intel® Turbo Boost technology can be engaged with any number of cores or logical processors enabled and active. This results in increased performance of both multi-threaded and single-threaded workloads.

3.2 Jitter Control

Note: Since a notebook has a quiet different jitter behaviour than desktop systems, an enhanced jitter control mechanism is required. Therefore SYBERA provides a registry entry called “JitterCtrl”. This entry allows an adaptive iteration to the best jitter behaviour of the notebook.


Registrierungs-Editor
Datei Bearbeiten Ansicht Favoriten ?
Realtek Semiconductor Corp.
Schlumberger
Secure
Sybera
ECT
ETH
PNT
SC3
SHA
XGO
Windows 3.1 Migration Status
WinPcap
SYSTEM

Name	Typ	Wert
(Standard)	REG_SZ	
CoreID	REG_DWORD	0x00000008 (8)
DestinationPath	REG_SZ	C:\SHA
JitterCtrl	REG_DWORD	0x00000001 (1)
KeyCode	REG_SZ	fee25bbc-b8fa165f
NoDynLoad	REG_DWORD	0x00000001 (1)
NoErrLog	REG_DWORD	0x00000000 (0)
NoErrMsg	REG_DWORD	0x00000001 (1)
Pid	REG_SZ	e1a980ee
SerNum	REG_SZ	12345678

Arbeitsplatz\HKEY_LOCAL_MACHINE\SOFTWARE\Sybera\SHA

Following values are valid:

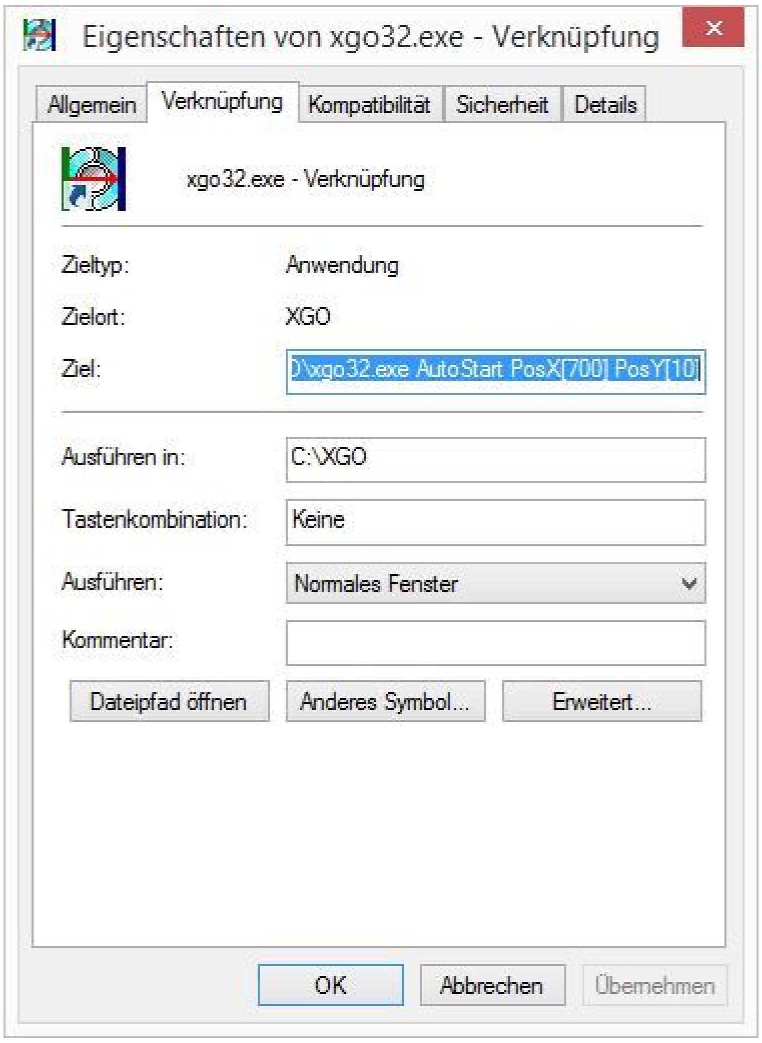
0:	No enhanced jitter control
1:	Enhaced Jitter Control, Step 1 (first choice together with BIOS settings)
2:	Enhaced Jitter Control, Step 2 (for INTEL platforms only)
3:	Enhaced Jitter Control, Step 3 (for INTEL platforms only, together with BIOS settings)

<u>Note:</u>
In order to operate SYBERA software under VISTA or Windows 7, the program must be started with Administrator privilleges. It's not sufficient to be logged on as administrator, since VISTA or Windows 7 does not share the rights with the processes automatically.

3.3 AutoStart

Note: For frequent operation X-GO may be started with command line options:

AutoStart	X-GO uses last saved startup parameters


Eigenschaften von xgo32.exe - Verknüpfung

xgo32.exe - Verknüpfung

Zieltyp: Anwendung
Zielort: XGO
Ziel: D\xgo32.exe AutoStart PosX[700] PosY[10]
Ausführen in: C:\XGO
Tastenkombination: Keine
Ausführen: Normales Fenster
Kommentar:

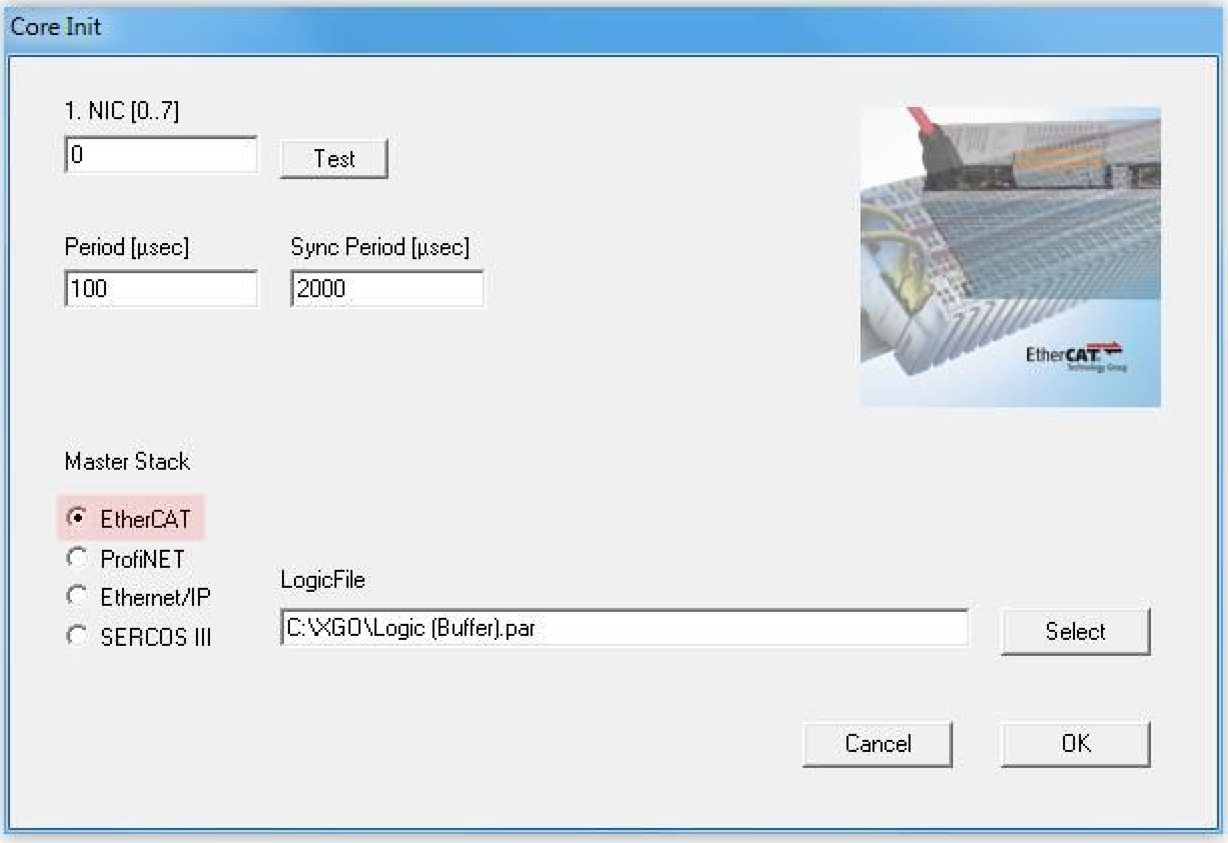
PosX[Pixel]	X position of the main window
PosY[Pixel]	Y position of the main window

Note:

On Autostart, all Dialogs (Station Information, Variable Information and Value Dialogs) are hidden. The Main windows is minimized. Only Panel elements are displayed

4 EtherCAT

When using X-GO EtherCAT following parameters have to be used:



1. NIC [0..7]: 0

Period [µsec]: 100

Sync Period [µsec]: 2000

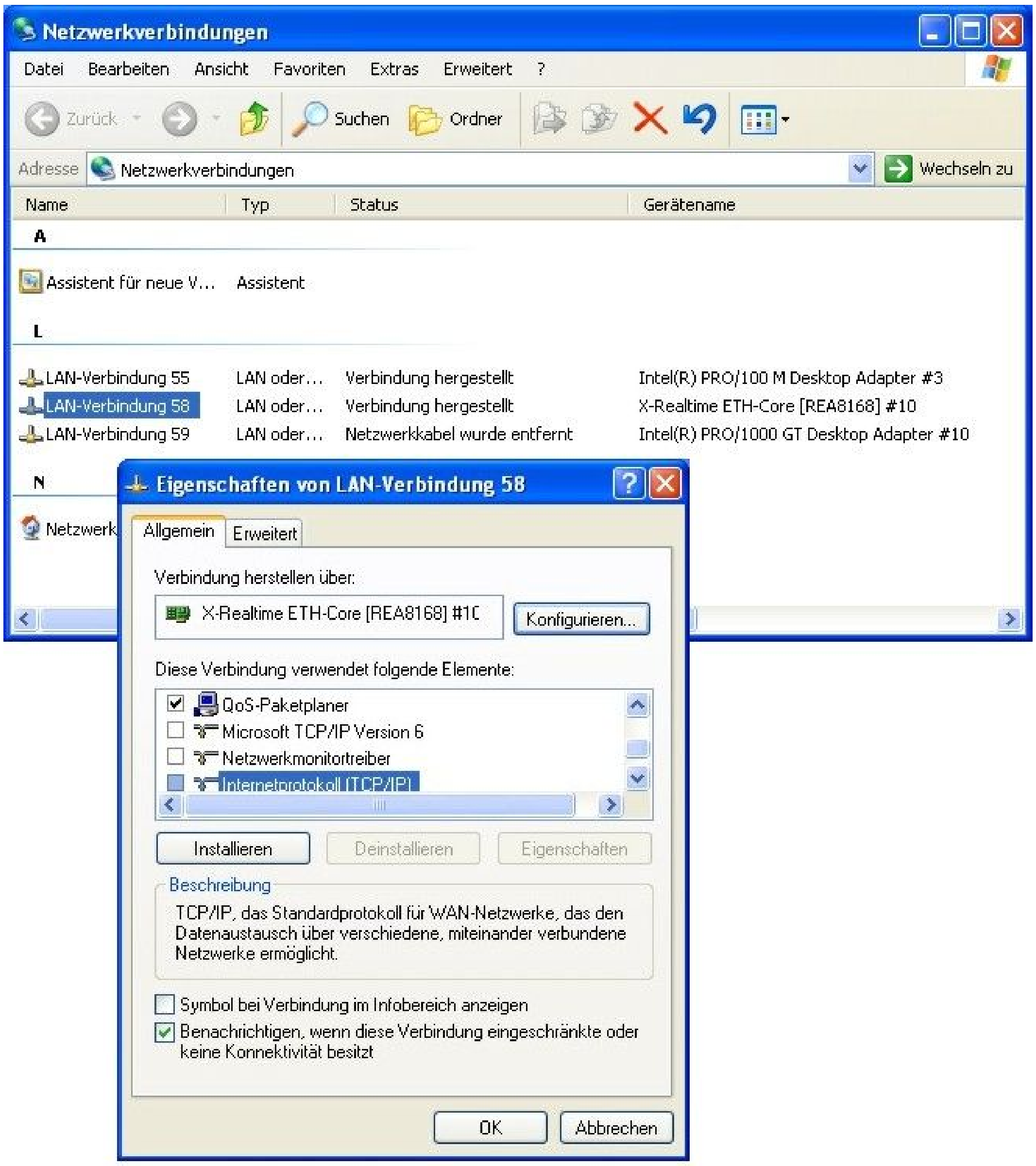
Master Stack: [✓ EtherCAT] ProfiNET Ethernet/IP SERCOS III

LogicFile: C:\XGO\Logic (Buffer).par

1.NIC	Index of the first realtime Ethernet dirver (typically 0)
Period	Sampling period (typically 100 µsec)
Sync Period	Synchronisation (Update) Period (typically 2000 µsec)

Note:

For proper EtherCAT operation its recommended to use the SERCOS III network as standalone network. This requires to turn off the Windows protocols for this network connection:



Netzwerkverbindungen
Datei Bearbeiten Ansicht Favoriten Extras Erweitert ?
Zurück Suchen Ordner
Adresse Netzwerkverbindungen
Wechseln zu

Name	Typ	Status	Gerätename
A			
Assistent für neue V...	Assistent		
L			
LAN-Verbindung 55	LAN oder...	Verbindung hergestellt	Intel(R) PRO/100 M Desktop Adapter #3
LAN-Verbindung 58	LAN oder...	Verbindung hergestellt	X-Realtime ETH-Core [REA8168] #10
LAN-Verbindung 59	LAN oder...	Netzwerkkabel wurde entfernt	Intel(R) PRO/1000 GT Desktop Adapter #10
N			
Netzwerk			

Eigenschaften von LAN-Verbindung 58

Allgemein | Erweitert

Verbindung herstellen über:
X-Realtime ETH-Core [REA8168] #10
Konfigurieren...

Diese Verbindung verwendet folgende Elemente:
- [✓] QoS-Paketplaner
- [ ] Microsoft TCP/IP Version 6
- [ ] Netzwerkmonitortreiber
- [ ] Internetprotokoll (TCP/IP)

Installieren | Deinstallieren | Eigenschaften

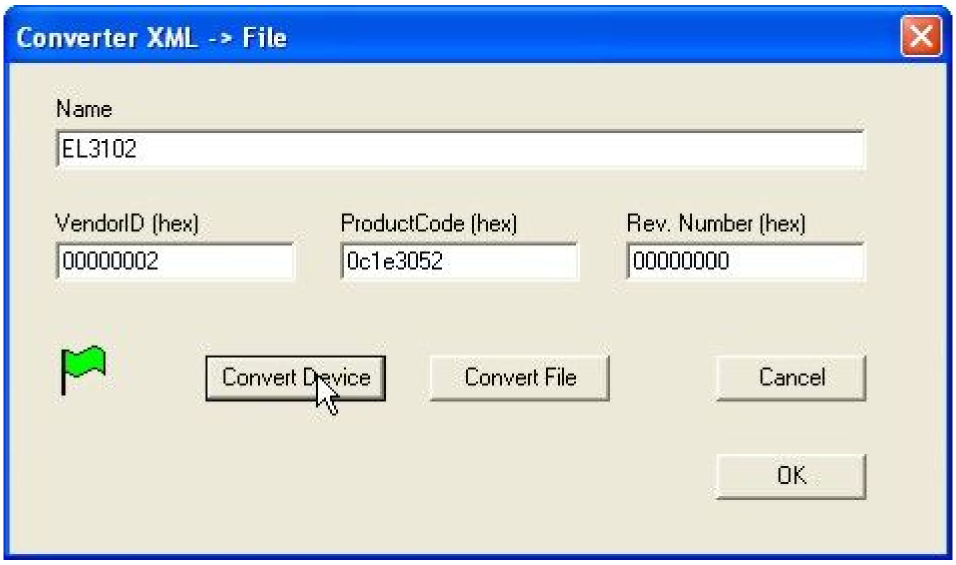
Beschreibung
TCP/IP, das Standardprotokoll für WAN-Netzwerke, das den Datenaustausch über verschiedene, miteinander verbundene Netzwerke ermöglicht.

- [ ] Symbol bei Verbindung im Infobereich anzeigen
- [✓] Benachrichtigen, wenn diese Verbindung eingeschränkte oder keine Konnektivität besitzt

OK | Abbrechen

4.1 XML Converter

Since the X-GO EtherCAT Master uses a native parameter format, the XML (ESI) device information has to be converted. The tool ECATVERIFY has implemented a XML parser which allows converting XML device information into native parameter information and save it into the file ECATDEVICE.PAR (to be placed in the directory \WINDOWS\SYSTEM32). For conversion the XML files must be placed in the directory where ECATVERIFY is located. The device which is to be converted may be searched within an XML file by its Name, Product Code, Vendor ID or Revision Number. It is also possible to convert the whole XML file to the native format. Devices which are already present in ECATDEVICE.PAR will be updated.


Converter XML -> File

Name: EL3102

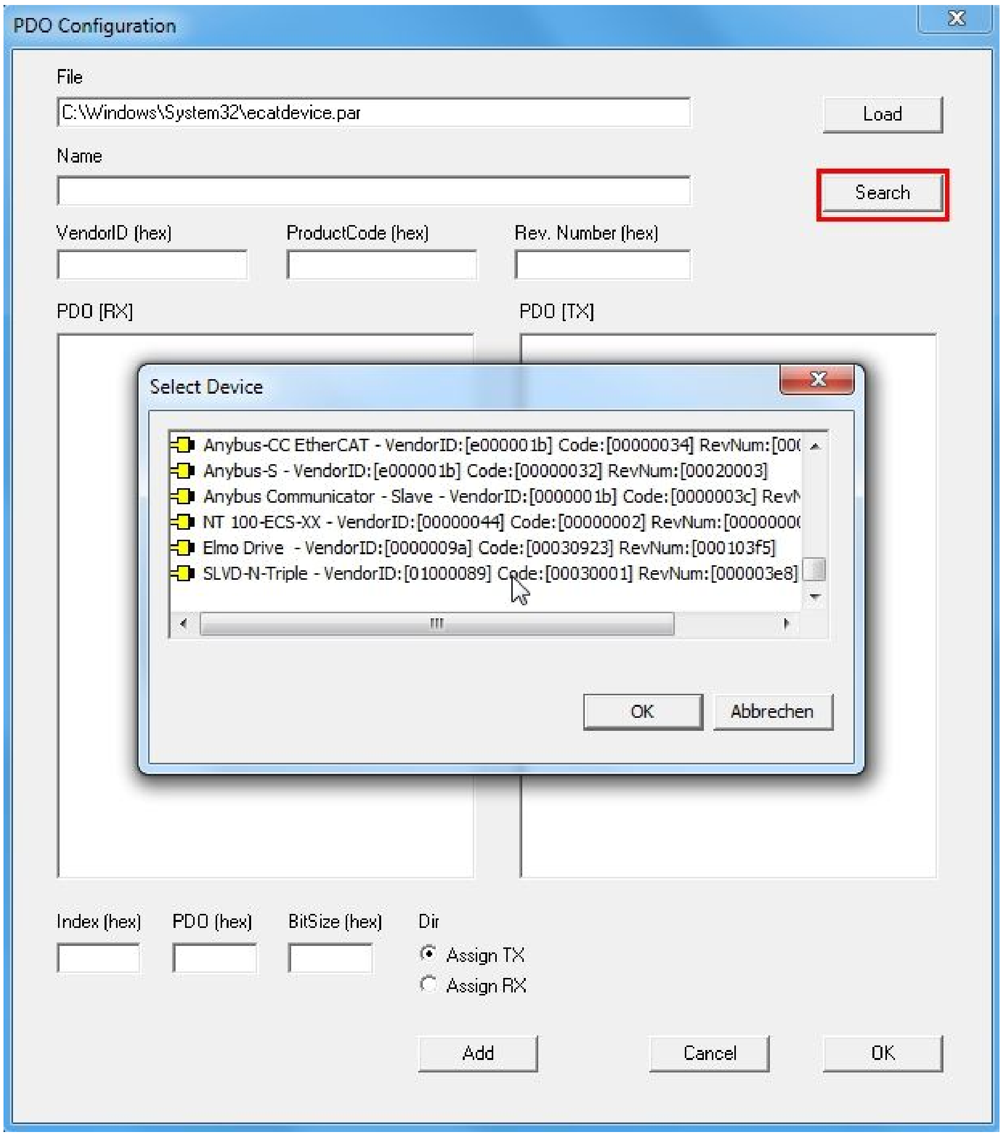
VendorID (hex): 00000002

ProductCode (hex): 0c1e3052

Rev. Number (hex): 00000000

4.2 PDO Configurator

The integrated PDO configurator allows easy determination of the EtherCAT PDO mapping. The PDO Configurator allows adding, removing, and deleting PDO mapping objects. With the PDO-Configurator devices located in the file ECATDEVICE.PAR can be listed or searched for editing the PDO mappings.


PDO Configuration

File: C:\Windows\System32\ecatdevice.par

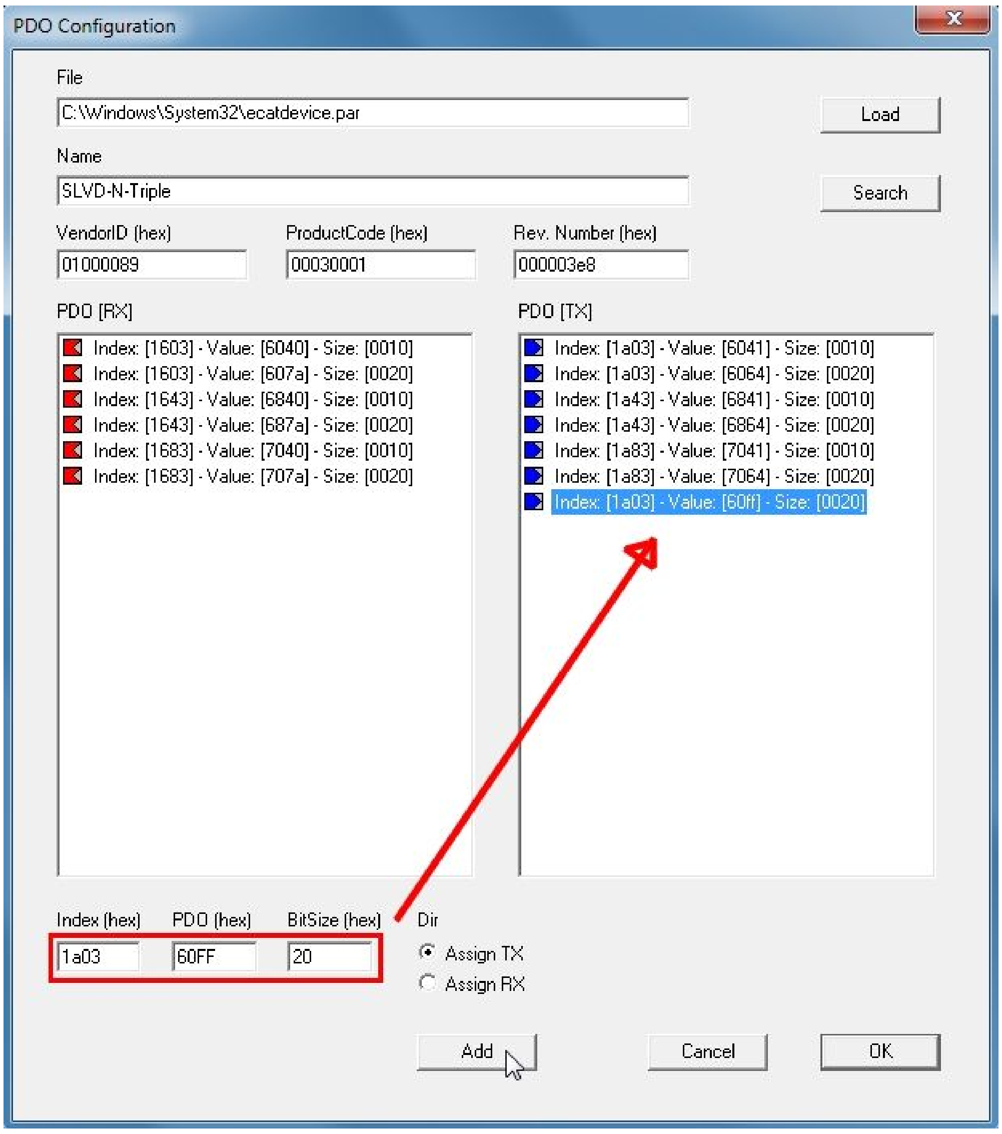
Select Device

```
Anybus-CC EtherCAT - VendorID:[e000001b] Code:[00000034] RevNum:[00(
Anybus-S - VendorID:[e000001b] Code:[00000032] RevNum:[00020003]
Anybus Communicator - Slave - VendorID:[0000001b] Code:[0000003c] RevN
NT 100-ECS-XX - VendorID:[00000044] Code:[00000002] RevNum:[0000000(
Elmo Drive  - VendorID:[0000009a] Code:[00030923] RevNum:[000103f5]
SLVD-N-Triple - VendorID:[01000089] Code:[00030001] RevNum:[000003e8]
```

Dir: [✓ Assign TX] Assign RX

Note:
Existing PDO-Mappings need to have an already listed PDO assignment (1C12 / 1C13). Otherwise the PDO mapping has to setup newly.

New PDO mappings are entered by index, PDO and bit size for assigning it to the corresponding PDO mapping list (TX / RX).


PDO Configuration

File: C:\Windows\System32\ecatdevice.par

Name: SLVD-N-Triple

VendorID (hex): 01000089
ProductCode (hex): 00030001
Rev. Number (hex): 000003e8

PDO [RX]
```
Index: [1603] - Value: [6040] - Size: [0010]
Index: [1603] - Value: [607a] - Size: [0020]
Index: [1643] - Value: [6840] - Size: [0010]
Index: [1643] - Value: [687a] - Size: [0020]
Index: [1683] - Value: [7040] - Size: [0010]
Index: [1683] - Value: [707a] - Size: [0020]
```

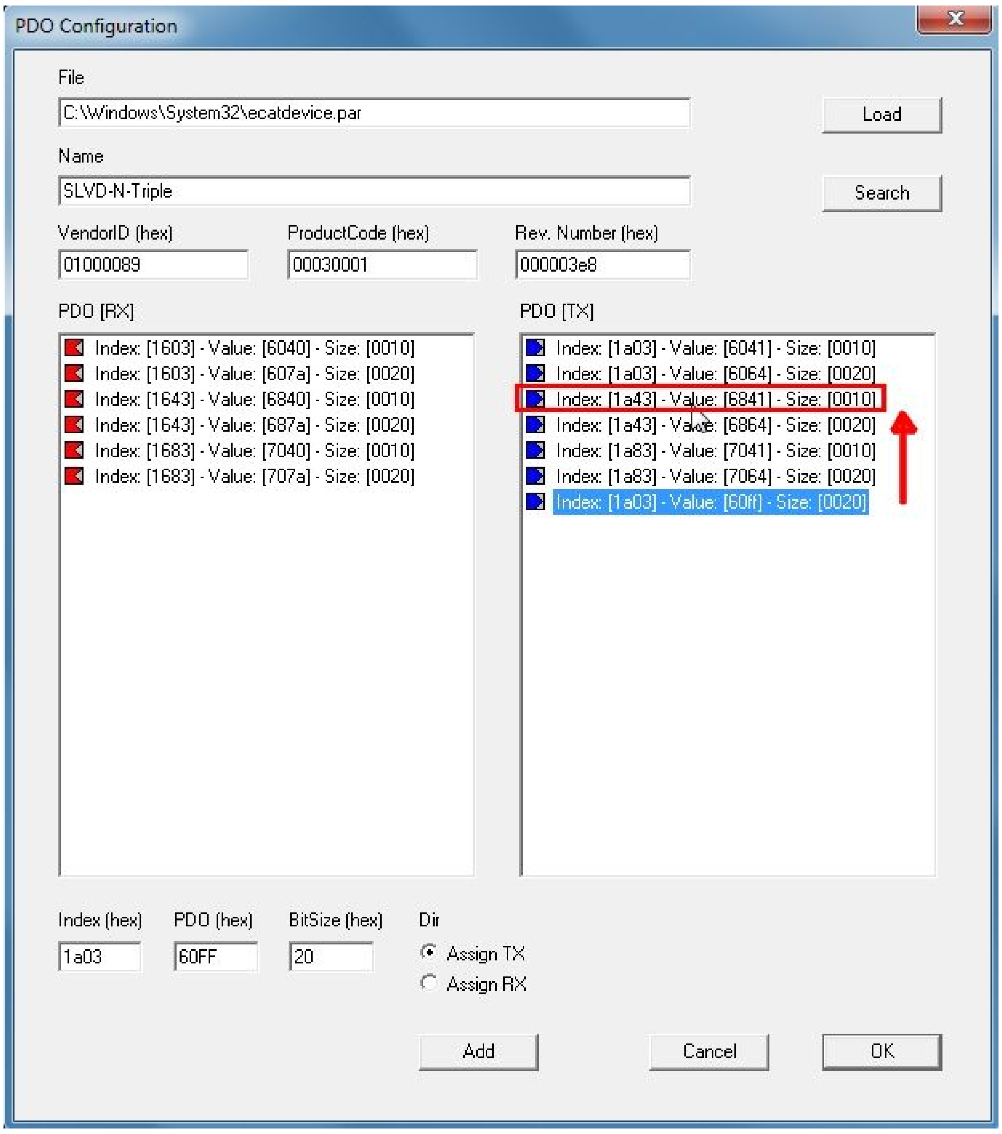
PDO [TX]
```
Index: [1a03] - Value: [6041] - Size: [0010]
Index: [1a03] - Value: [6064] - Size: [0020]
Index: [1a43] - Value: [6841] - Size: [0010]
Index: [1a43] - Value: [6864] - Size: [0020]
Index: [1a83] - Value: [7041] - Size: [0010]
Index: [1a83] - Value: [7064] - Size: [0020]
Index: [1a03] - Value: [60ff] - Size: [0020]
```

Index (hex): 1a03
PDO (hex): 60FF
BitSize (hex): 20

Dir: [✓ Assign TX] Assign RX

Selected PDO mappings may be deleted by pressing the key „DELETE“.

The new PDO mapping entries can be moved to the appropriate position. For this, the corresponding entry is selected to be moved and swapped with the entry of the desired position by clicking on it.



File: C:\Windows\System32\ecatdevice.par

Name: SLVD-N-Triple

VendorID (hex): 01000089

ProductCode (hex): 00030001

Rev. Number (hex): 000003e8

PDO [RX]

```
Index: [1603] - Value: [6040] - Size: [0010]
Index: [1603] - Value: [607a] - Size: [0020]
Index: [1643] - Value: [6840] - Size: [0010]
Index: [1643] - Value: [687a] - Size: [0020]
Index: [1683] - Value: [7040] - Size: [0010]
Index: [1683] - Value: [707a] - Size: [0020]
```

PDO [TX]

```
Index: [1a03] - Value: [6041] - Size: [0010]
Index: [1a03] - Value: [6064] - Size: [0020]
Index: [1a43] - Value: [6841] - Size: [0010]
Index: [1a43] - Value: [6864] - Size: [0020]
Index: [1a83] - Value: [7041] - Size: [0010]
Index: [1a83] - Value: [7064] - Size: [0020]
Index: [1a03] - Value: [60ff] - Size: [0020]
```

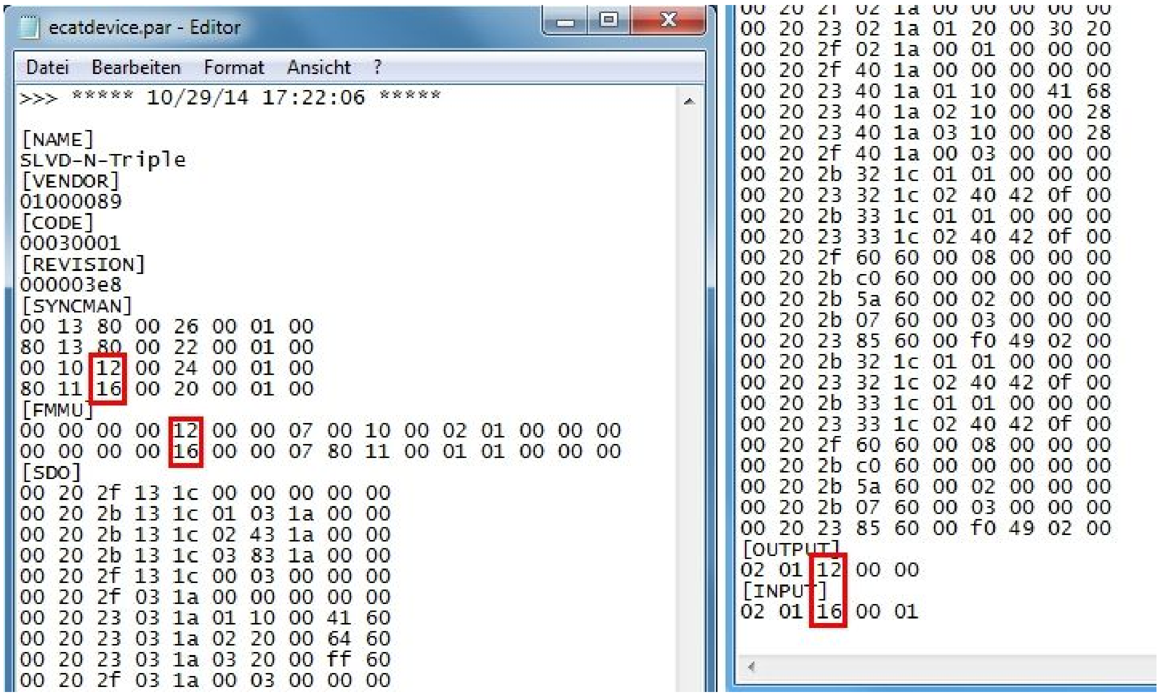
Index (hex): 1a03

PDO (hex): 60FF

BitSize (hex): 20

Dir: [✓ Assign TX] Assign RX

Once configured, the device located in the file ECATDEVICE.PAR file is automatically updated and the value "length" of the corresponding FMMU-, SYNCMAN- and INPUT / OUTPUT descriptor entries is automatically updated.



```
>>> ***** 10/29/14 17:22:06 *****

[NAME]
SLVD-N-Triple
[VENDOR]
01000089
[CODE]
00030001
[REVISION]
000003e8
[SYNCMAN]
00 13 80 00 26 00 01 00
80 13 80 00 22 00 01 00
00 10 12 00 24 00 01 00
80 11 16 00 20 00 01 00
[FMMU]
00 00 00 00 12 00 00 07 00 10 00 02 01 00 00 00
00 00 00 00 16 00 00 07 80 11 00 01 01 00 00 00
[SDO]
00 20 2f 13 1c 00 00 00 00 00
00 20 2b 13 1c 01 03 1a 00 00
00 20 2b 13 1c 02 43 1a 00 00
00 20 2b 13 1c 03 83 1a 00 00
00 20 2f 13 1c 00 03 00 00 00
00 20 2f 03 1a 00 00 00 00 00
00 20 23 03 1a 01 10 00 41 60
00 20 23 03 1a 02 20 00 64 60
00 20 23 03 1a 03 20 00 ff 60
00 20 2f 03 1a 00 03 00 00 00
```

```
00 20 2f 02 1a 00 00 00 00 00
00 20 23 02 1a 01 20 00 30 20
00 20 2f 02 1a 00 01 00 00 00
00 20 2f 40 1a 00 00 00 00 00
00 20 23 40 1a 01 10 00 41 68
00 20 23 40 1a 02 10 00 00 28
00 20 23 40 1a 03 10 00 00 28
00 20 2f 40 1a 00 03 00 00 00
00 20 2b 32 1c 01 01 00 00 00
00 20 23 32 1c 02 40 42 0f 00
00 20 2b 33 1c 01 01 00 00 00
00 20 23 33 1c 02 40 42 0f 00
00 20 2f 60 60 00 08 00 00 00
00 20 2b c0 60 00 00 00 00 00
00 20 2b 5a 60 00 02 00 00 00
00 20 2b 07 60 00 03 00 00 00
00 20 23 85 60 00 f0 49 02 00
00 20 2b 32 1c 01 01 00 00 00
00 20 23 32 1c 02 40 42 0f 00
00 20 2b 33 1c 01 01 00 00 00
00 20 23 33 1c 02 40 42 0f 00
00 20 2f 60 60 00 08 00 00 00
00 20 2b c0 60 00 00 00 00 00
00 20 2b 5a 60 00 02 00 00 00
00 20 2b 07 60 00 03 00 00 00
00 20 23 85 60 00 f0 49 02 00
[OUTPUT]
02 01 12 00 00
[INPUT]
02 01 16 00 01
```

4.3 Device Parameter File

Usually device information is provided by a corresponding XML configuration file. Since the development of software with the EtherCAT Master Library has special needs for programming, the XML file must be parsed and translated into a native format. Therefore a configuration file named **ECATDEVICE.PAR** has to be generated with the tool **ECATVERIFY** and placed into the directory **\windows\system32**. The ECATDEVICE.PAR is a text based file with sections for Product Code, Name, SYNC Manager, FMMU Manager, SDO and Data Description. A new device description must start with the signature **">>>"**

Sample:

```
>>> ***** 09/15/10 14:56:37 *****

[NAME]
EL3102
[VENDOR]
00000002
[CODE]
0c1e3052
[REVISION]
00100000
[SYNCMAN]
00 10 80 00 26 00 01 00
80 10 80 00 22 00 01 00
00 11 00 00 04 00 00 00
80 11 06 00 20 00 01 00
[FMMU]
00 00 00 00 06 00 00 07 80 11 00 01 01 00 00 00
00 00 00 00 01 00 00 00 0d 08 00 01 01 00 00 00
[SDO]
00 20 2f 13 1c 00 00 00 00 00
00 20 2b 13 1c 01 00 1a 00 00
00 20 2b 13 1c 02 01 1a 00 00
00 20 2f 13 1c 00 02 00 00 00
[OUTPUT]
[INPUT]
02 01 01 00 00
02 06 02 00 00
02 01 01 00 00
02 06 02 00 00
```

Note: With newer devices the configuration is stored inside the EEPROM. The EtherCAT Master Library is able to configure the devices by parsing the EEPROM information, even without XML file or Native file. But without using the configuration file, the configuration time increases by parsing EEPROM information. The Software ECATVERIFY parses XML information and EEPROM information and converts it into the native format and gives additional help for configuration.

4.3.1 Section [NAME]

This section contains the name of the device:

```
[NAME]
EL3102
```

4.3.2 Section [VENDOR]

This section contains the vendor ID of the device:

```
[VENDOR]
00000002
```

4.3.3 Section [CODE]

This section contains the product code of the device:

```
[CODE]
0C1E3052
```

4.3.4 Section [REVISION]

This section contains the revision number of the device:

```
[CODE]
00100000
```

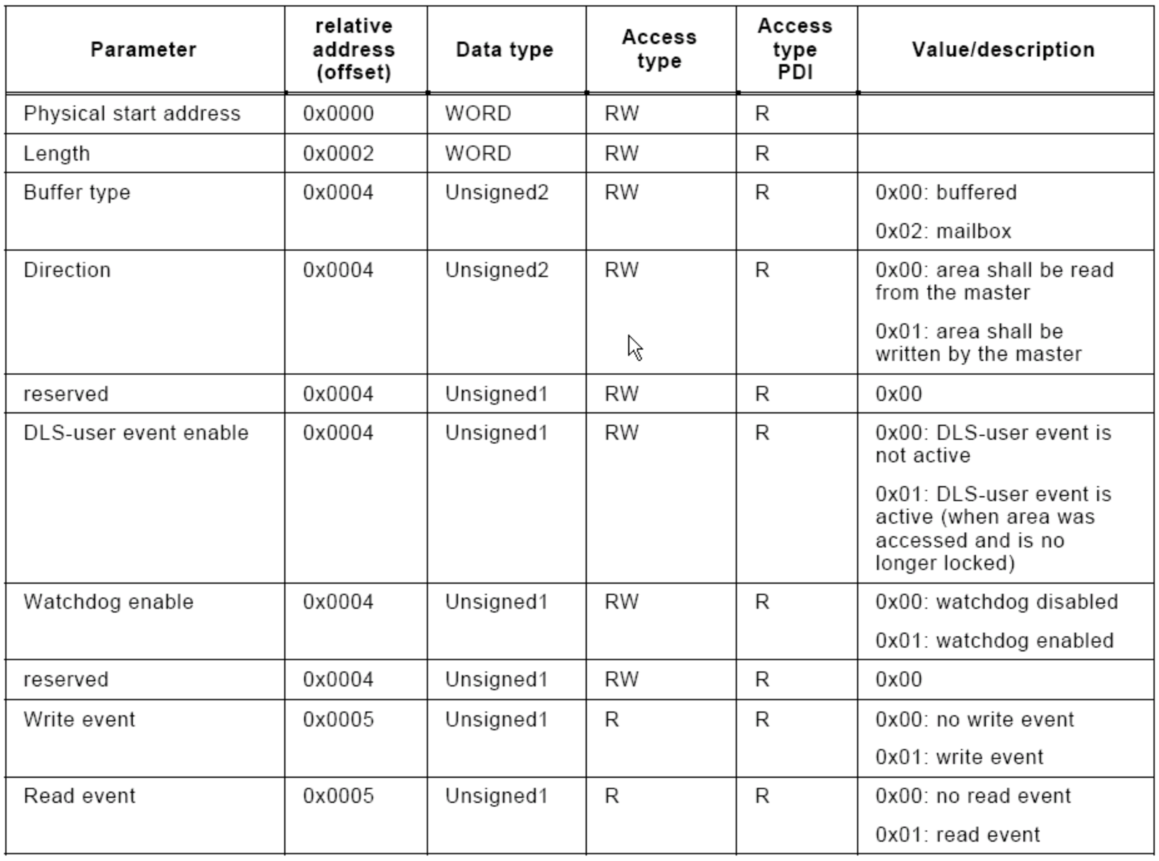
4.3.5 Section [SYNCMAN]

This section contains the binary data for the synchronisation manager of the device:

```
[SYNCMAN]
00 18 F6 00 26 00 01 00
F6 18 F6 00 22 00 01 00
00 10 00 00 24 00 00 00
00 11 06 00 20 00 01 00
```

Meaning:

```
| Addr   |   Len   |Cntr    |ChEn
                        |Stat    |Res
|        |         |    |    |    |    |
 00  18  F6  00  26  00  01  00     <- SYNMAN0
 F6  18  F6  00  22  00  01  00     <- SYNMAN1
 00  10  00  00  24  00  00  00     <- SYNMAN2
 00  11  06  00  20  00  01  00     <- SYNMAN3
```



Parameter	relative address (offset)	Data type	Access type	Access type PDI	Value/description
Physical start address	0x0000	WORD	RW	R	
Length	0x0002	WORD	RW	R	
Buffer type	0x0004	Unsigned2	RW	R	0x00: buffered 0x02: mailbox
Direction	0x0004	Unsigned2	RW	R	0x00: area shall be read from the master 0x01: area shall be written by the master
reserved	0x0004	Unsigned1	RW	R	0x00
DLS-user event enable	0x0004	Unsigned1	RW	R	0x00: DLS-user event is not active 0x01: DLS-user event is active (when area was accessed and is no longer locked)
Watchdog enable	0x0004	Unsigned1	RW	R	0x00: watchdog disabled 0x01: watchdog enabled
reserved	0x0004	Unsigned1	RW	R	0x00
Write event	0x0005	Unsigned1	R	R	0x00: no write event 0x01: write event
Read event	0x0005	Unsigned1	R	R	0x00: no read event 0x01: read event

reserved	0x0005	unsigned1	R	R	0x00
Mailbox state	0x0005	Unsigned1	R	R	0x00: mailbox empty 0x01: mailbox full
Buffered state	0x0005	Unsigned2	R	R	0x00: first buffer 0x01: second buffer 0x02: third buffer 0x03: buffer locked
reserved	0x0005	Unsigned2	R	R	0x00
Channel enable	0x0006	Unsigned1	RW	R	0x00: channel disabled 0x01: channel enabled
Repeat	0x0006	Unsigned1	RW	R	
reserved	0x0006	Unsigned4	RW	R	0x00
DC Event 0 with Bus write	0x0006	Unsigned1	RW	R	0x00: no Event 0x01: DC Event if master writes complete buffer
DC Event 0 with local write	0x0006	Unsigned1	RW	R	0x00: no Event 0x01: DC Event if DL-user writes complete buffer
Channel enable PDI	0x0007	Unsigned1	R	RW	0x00: channel disabled 0x01: channel enabled
RepeatAck	0x0007	Unsigned1	R	RW	shall follow repeat after data recovery
reserved	0x0007	Unsigned6	R	RW	0x00

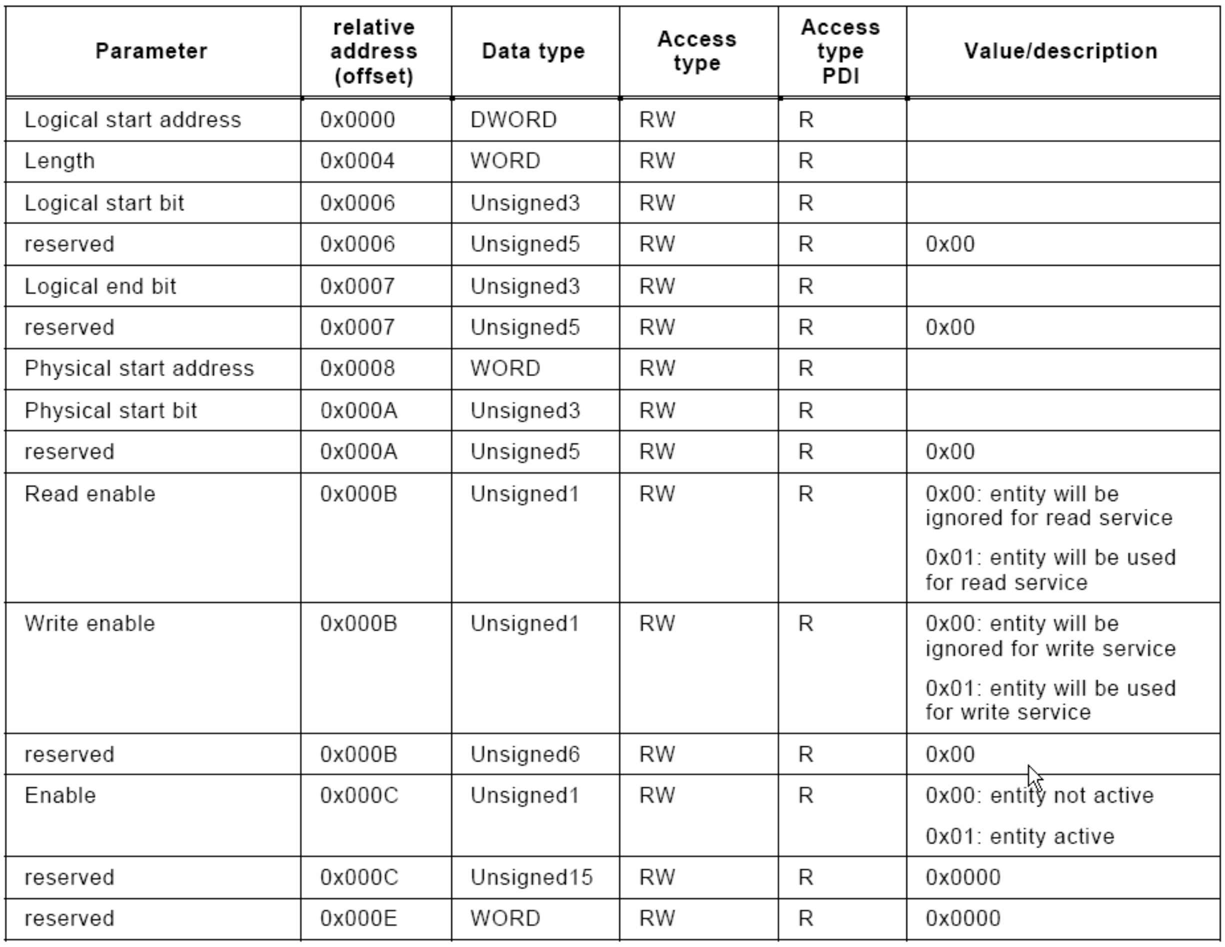
4.3.6 Section [FMMU]

This section contains the binary data for the FMMU manager of the device:

```
[FMMU]
06 00 00 00 01 00 00 00 0D 08 00 01 01 00 00 00
00 00 00 00 06 00 00 07 00 11 00 01 01 00 00 00
```

Meaning:

```
| LogAddr(Offs) |  Len  |LogStartBit    |PhysStartBit
                            |LogEndBit      |RdWrEnable
                                |PhysAddr       |ChEnable
|               |       |   |   |       |   |   |               |
 00  00  00  00  01  00  00  00  0D  08  00  01  01  00  00  00 <- FMMU0
 00  00  00  00  06  00  00  07  00  11  00  01  01  00  00  00 <- FMMU1
```



Parameter	relative address (offset)	Data type	Access type	Access type PDI	Value/description
Logical start address	0x0000	DWORD	RW	R	
Length	0x0004	WORD	RW	R	
Logical start bit	0x0006	Unsigned3	RW	R	
reserved	0x0006	Unsigned5	RW	R	0x00
Logical end bit	0x0007	Unsigned3	RW	R	
reserved	0x0007	Unsigned5	RW	R	0x00
Physical start address	0x0008	WORD	RW	R	
Physical start bit	0x000A	Unsigned3	RW	R	
reserved	0x000A	Unsigned5	RW	R	0x00
Read enable	0x000B	Unsigned1	RW	R	0x00: entity will be ignored for read service 0x01: entity will be used for read service
Write enable	0x000B	Unsigned1	RW	R	0x00: entity will be ignored for write service 0x01: entity will be used for write service
reserved	0x000B	Unsigned6	RW	R	0x00
Enable	0x000C	Unsigned1	RW	R	0x00: entity not active 0x01: entity active
reserved	0x000C	Unsigned15	RW	R	0x0000
reserved	0x000E	WORD	RW	R	0x0000

4.3.7 Section [SDO]

This section contains the binary SDO data of the device:

```
[SDO]
00 20 2F 12 1C 00 00 00 00 00
00 20 2F 13 1C 00 00 00 00 00
00 20 2B 13 1C 01 00 1A 00 00
00 20 2B 13 1C 02 01 1A 00 00
00 20 2F 13 1C 00 02 00 00 00
```

Meaning:

```
|NumServ|Cmd|Index   |SubIndex
                         |Data
|       |   |       |   |              |
 00   20   2F   12   1C   00   00   00   00   00 <- COE Cmd0
 00   20   2F   13   1C   00   00   00   00   00 <- COE Cmd1
 00   20   2B   13   1C   01   00   1A   00   00 <- COE Cmd2
 00   20   2B   13   1C   02   01   1A   00   00 <- COE Cmd3
 00   20   2F   13   1C   00   02   00   00   00 <- COE Cmd4
```

SDO Header Word and Command Byte

Frame part	Data Field	Data Type	Value/Description
CANopen Header	Number	Unsigned9	0x00
	Reserved	Unsigned3	0x00
	Service	Unsigned4	0x02: SDO Request
SDO	Size Indicator	Unsigned1	0x00: size of Data (1..4) unspecified 0x01: size of Data in Data Set Size specified
	Transfer Type	Unsigned1	0x01: Expedited transfer
	Data Set Size	Unsigned2	0x00: 4 Octet Data 0x01: 3 Octet Data 0x02: 2 Octet Data 0x03: 1 Octet Data
	Complete Access	Unsigned1	0x00
	Command	Unsigned3	0x01: Initiate Download Request

<u>Sample:</u>

COE Header 2000h :	SDO Request
SDO Cmd 2Fh :	Data in Data Set Size, exp. Transfer, 1 Oct. Data, Download Req.
Index 1C10h :	Sync Manager 0 PDO Assignment (UNSIGNED16)
Index 1C11h :	Sync Manager 1 PDO Assignment (UNSIGNED16)
Index 1C12h :	Sync Manager 2 PDO Assignment (UNSIGNED16)
Index 1C13h :	Sync Manager 3 PDO Assignment (UNSIGNED16)

4.3.8 Section [OUTPUT] / [INPUT]

This section contains the output/input data description of the device:

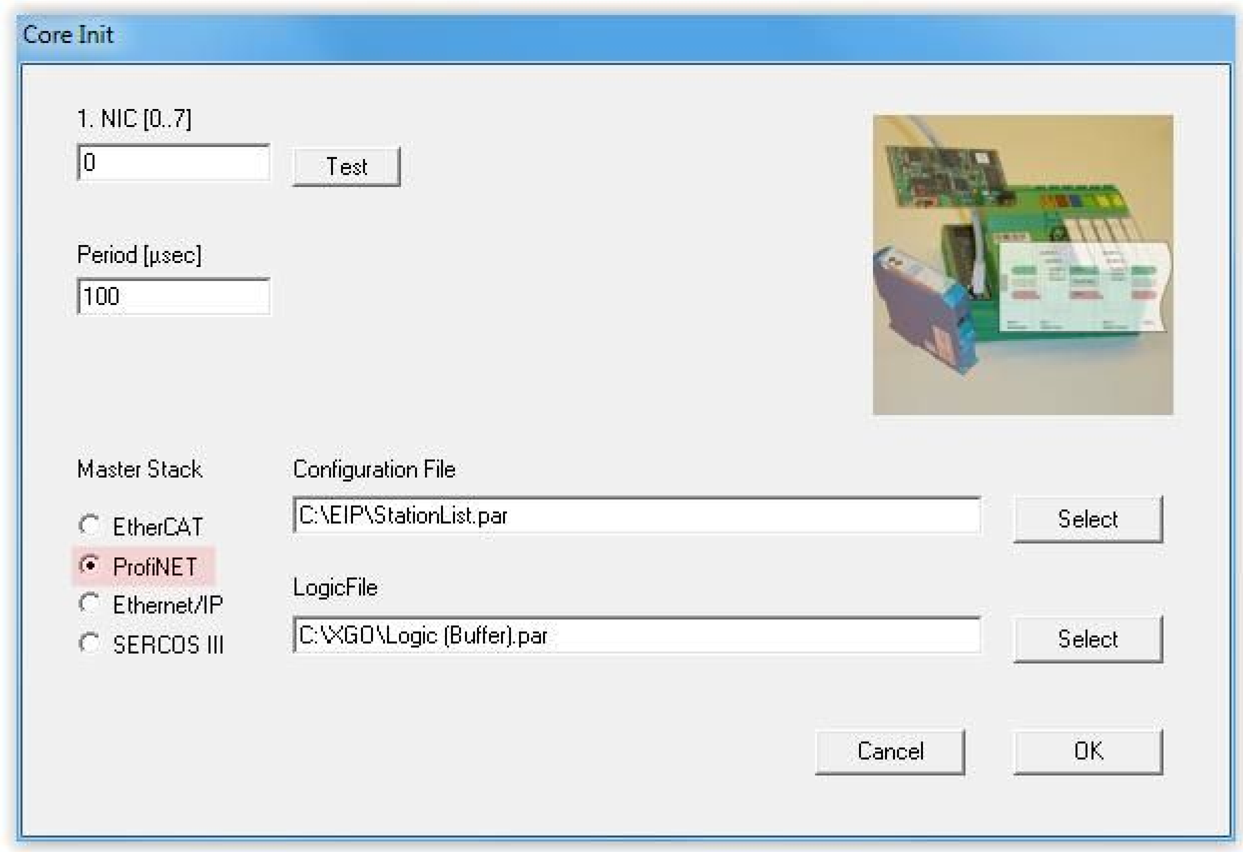
```
[OUTPUT]
01 01 01 00 00
02 02 02 00 00
01 01 01 00 00
02 02 02 00 00
03 02 02 00 00
03 02 02 00 00
```

Meaning:

```
The output data is based on 6 data items:

Item Type   (01 : DATA_ITEM_STATUS)
(02 : DATA_ITEM_VALUE)
(03 : DATA_ITEM_SCALE)

Data Type   (01 : DATA_TYPE_U8)
                (02 : DATA_TYPE_U16)


Item Type
    Data Type
        Data Len
                FMMU Index
|   |   |       |   |
 01  01  01  00  00 <- Item 0
 02  02  02  00  00 <- Item 1
 01  01  01  00  00 <- Item 2
 02  02  02  00  00 <- Item 3
 03  02  02  00  00 <- Item 4
 03  02  02  00  00 <- Item 5
```

5 ProfiNET

When using X-GO ProfiNET following parameters have to be used:



Core Init

1. NIC [0..7]: 0

Period [µsec]: 100

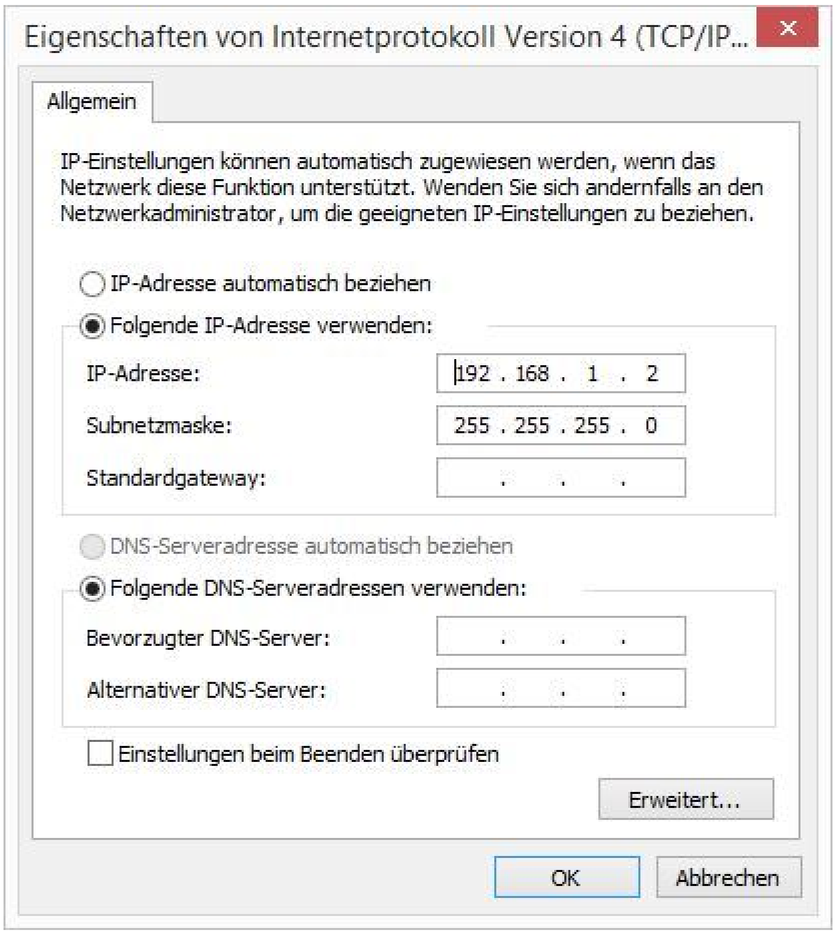
Master Stack: ( ) EtherCAT [✓ ProfiNET] ( ) Ethernet/IP ( ) SERCOS III

Configuration File: C:\EIP\StationList.par

LogicFile: C:\XGO\Logic (Buffer).par

1.NIC	Index of the first realtime Ethernet driver (typically 0)
Period	Sampling period (typically 100 µsec)
Configuration File	Path to configuration file (e.g. STATIONLIST.PAR)

Note: ProfiNET requires the Host PC to have a valid IP-Address. So, check the netadapter setting.



Eigenschaften von Internetprotokoll Version 4 (TCP/IP...

Allgemein

IP-Einstellungen können automatisch zugewiesen werden, wenn das Netzwerk diese Funktion unterstützt. Wenden Sie sich andernfalls an den Netzwerkadministrator, um die geeigneten IP-Einstellungen zu beziehen.

○ IP-Adresse automatisch beziehen
● Folgende IP-Adresse verwenden:

IP-Adresse:	192 . 168 . 1 . 2
Subnetzmaske:	255 . 255 . 255 . 0
Standardgateway:	. . .

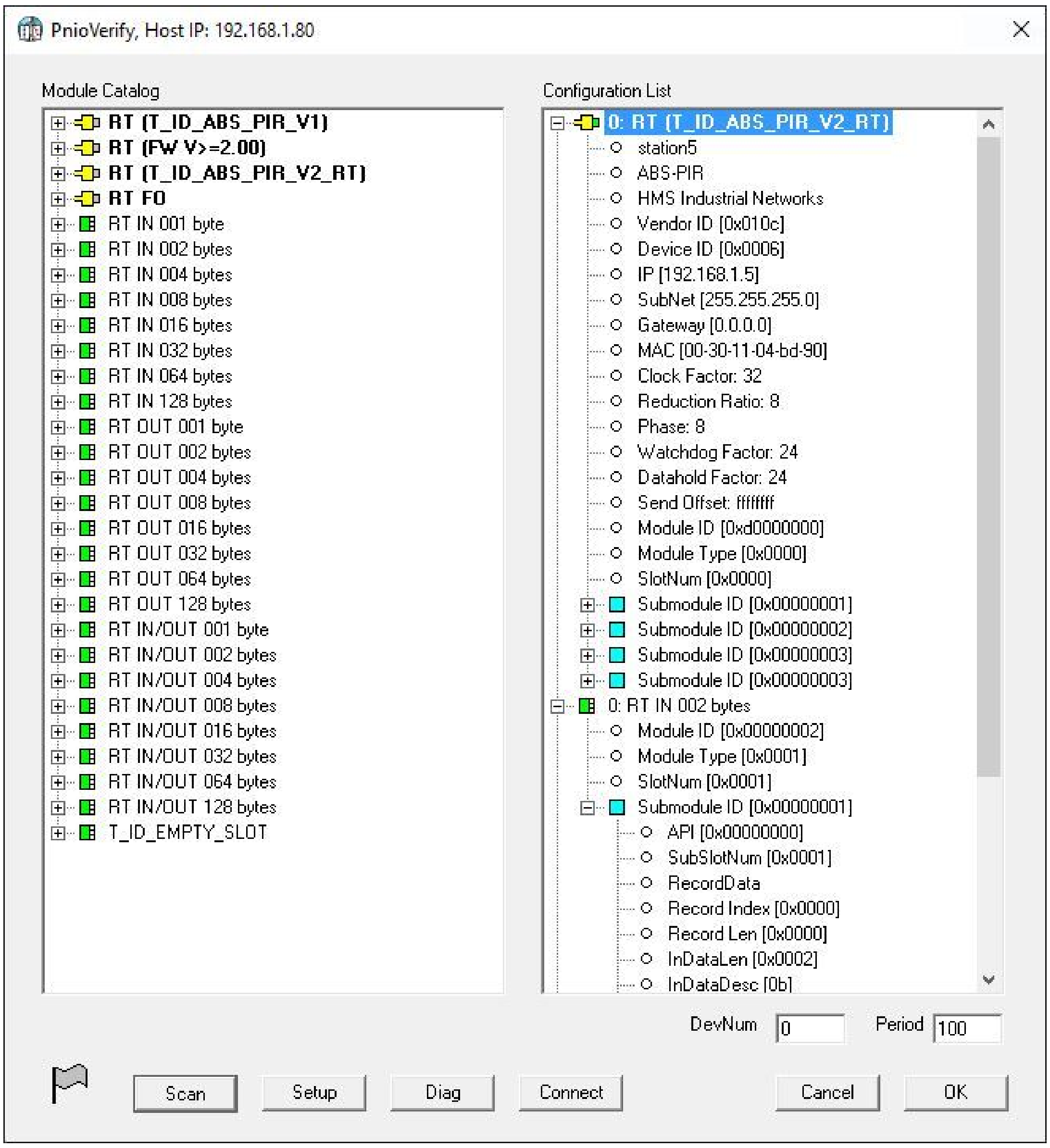
○ DNS-Serveradresse automatisch beziehen
● Folgende DNS-Serveradressen verwenden:

Bevorzugter DNS-Server:	. . .
Alternativer DNS-Server:	. . .

☐ Einstellungen beim Beenden überprüfen

5.1 Creating a configuration file

A ProfinetIO fieldbus system consists of several station devices (typically buscoupler devices). A station consists at least of one module (SLOT) and a module consists at least of one submodule (SUBSLOT). For proper operation the ProfinetIO devices needs first to be configured (by Station Name and IP) and a native STATIONLIST for operating the ProfiNET realtime library has to be created. Therefore SYBERA provides a program called PNIOVERIFY.EXE.


PnioVerify, Host IP: 192.168.1.80

Module Catalog

```
⊞ RT (T_ID_ABS_PIR_V1)
⊞ RT (FW V>=2.00)
⊞ RT (T_ID_ABS_PIR_V2_RT)
⊞ RT FO
⊞ RT IN 001 byte
⊞ RT IN 002 bytes
⊞ RT IN 004 bytes
⊞ RT IN 008 bytes
⊞ RT IN 016 bytes
⊞ RT IN 032 bytes
⊞ RT IN 064 bytes
⊞ RT IN 128 bytes
⊞ RT OUT 001 byte
⊞ RT OUT 002 bytes
⊞ RT OUT 004 bytes
⊞ RT OUT 008 bytes
⊞ RT OUT 016 bytes
⊞ RT OUT 032 bytes
⊞ RT OUT 064 bytes
⊞ RT OUT 128 bytes
⊞ RT IN/OUT 001 byte
⊞ RT IN/OUT 002 bytes
⊞ RT IN/OUT 004 bytes
⊞ RT IN/OUT 008 bytes
⊞ RT IN/OUT 016 bytes
⊞ RT IN/OUT 032 bytes
⊞ RT IN/OUT 064 bytes
⊞ RT IN/OUT 128 bytes
⊞ T_ID_EMPTY_SLOT
```

Configuration List

```
⊟ 0: RT (T_ID_ABS_PIR_V2_RT)
    station5
    ABS-PIR
    HMS Industrial Networks
    Vendor ID [0x010c]
    Device ID [0x0006]
    IP [192.168.1.5]
    SubNet [255.255.255.0]
    Gateway [0.0.0.0]
    MAC [00-30-11-04-bd-90]
    Clock Factor: 32
    Reduction Ratio: 8
    Phase: 8
    Watchdog Factor: 24
    Datahold Factor: 24
    Send Offset: ffffffff
    Module ID [0xd0000000]
    Module Type [0x0000]
    SlotNum [0x0000]
  ⊞ Submodule ID [0x00000001]
  ⊞ Submodule ID [0x00000002]
  ⊞ Submodule ID [0x00000003]
  ⊞ Submodule ID [0x00000003]
⊟ 0: RT IN 002 bytes
    Module ID [0x00000002]
    Module Type [0x0001]
    SlotNum [0x0001]
  ⊟ Submodule ID [0x00000001]
      API [0x00000000]
      SubSlotNum [0x0001]
      RecordData
      Record Index [0x0000]
      Record Len [0x0000]
      InDataLen [0x0002]
      InDataDesc [0b]
```

DevNum: 0 Period: 100

Note: Make shure a valid IP address is provided for the network connection.

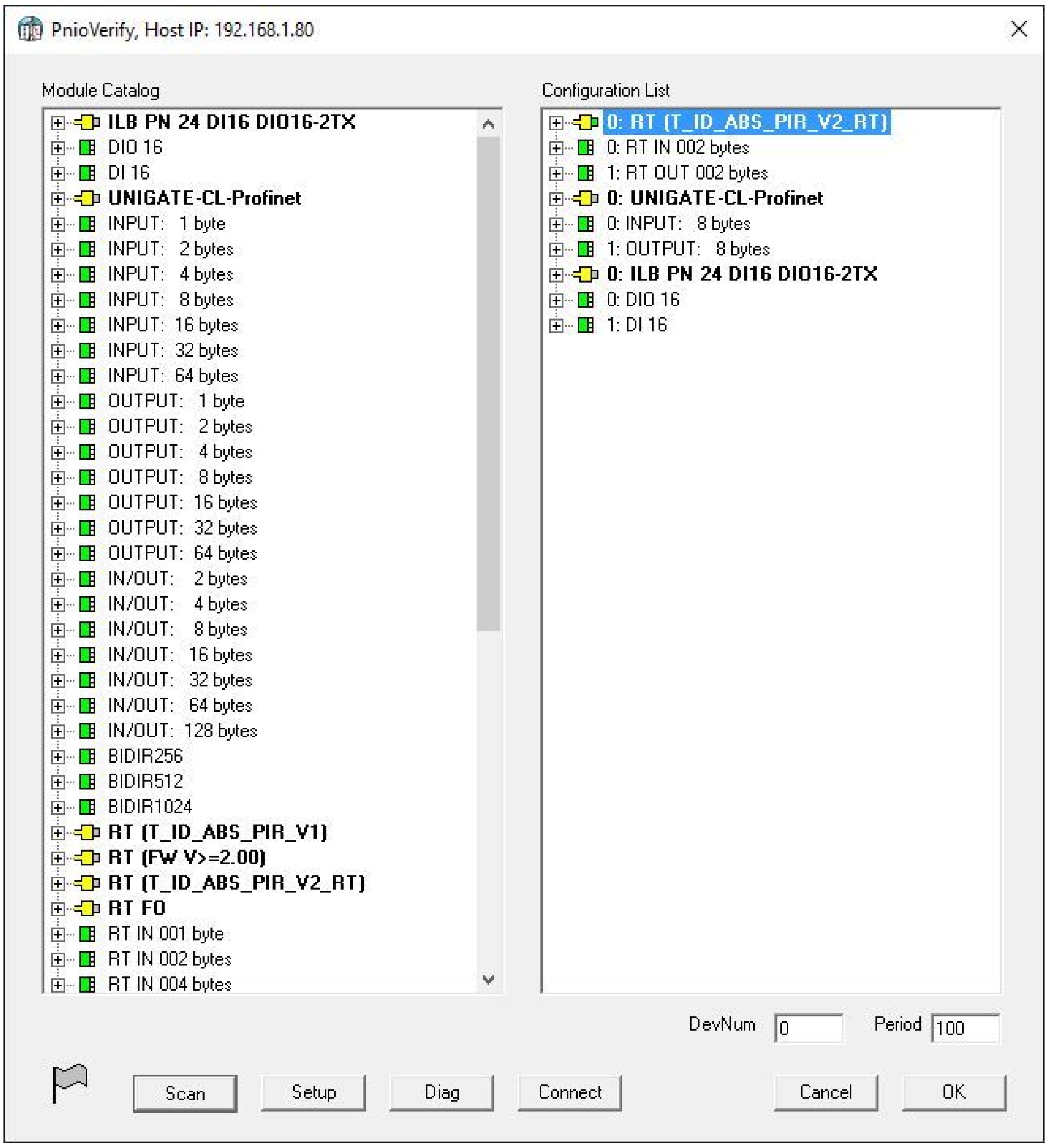
Note: If the application fails to run, check if the lastest Microsoft XML Parser has been installed. If not, install in the directory \APP\MSXML\MSXML6

PNIOVERIFY allows creating a native stationlist by selecting modules from a module catalog (leftside view). The catalog get its entries by the provides GSDML files which must be present in the same directory as PNIOVERIFY. A module is inserted to the station list configuration (rightside view) by a DRAG and DROP operation (just drag a module from the catalog to the station list configuration). There are two types of modules:

🟨	Accesspoint Module	(SLOT 0)
🟩	Functional Module	(SLOT 1 .. n)

5.2 Accesspoint Module

The accesspoint module keeps all information required for connecting to the fieldbus, as station name, IP parameters, MAC address, timing parameters. Therefore first task is to collect information about the ProfinetIO configuration by scanning the bus.



PnioVerify, Host IP: 192.168.1.80

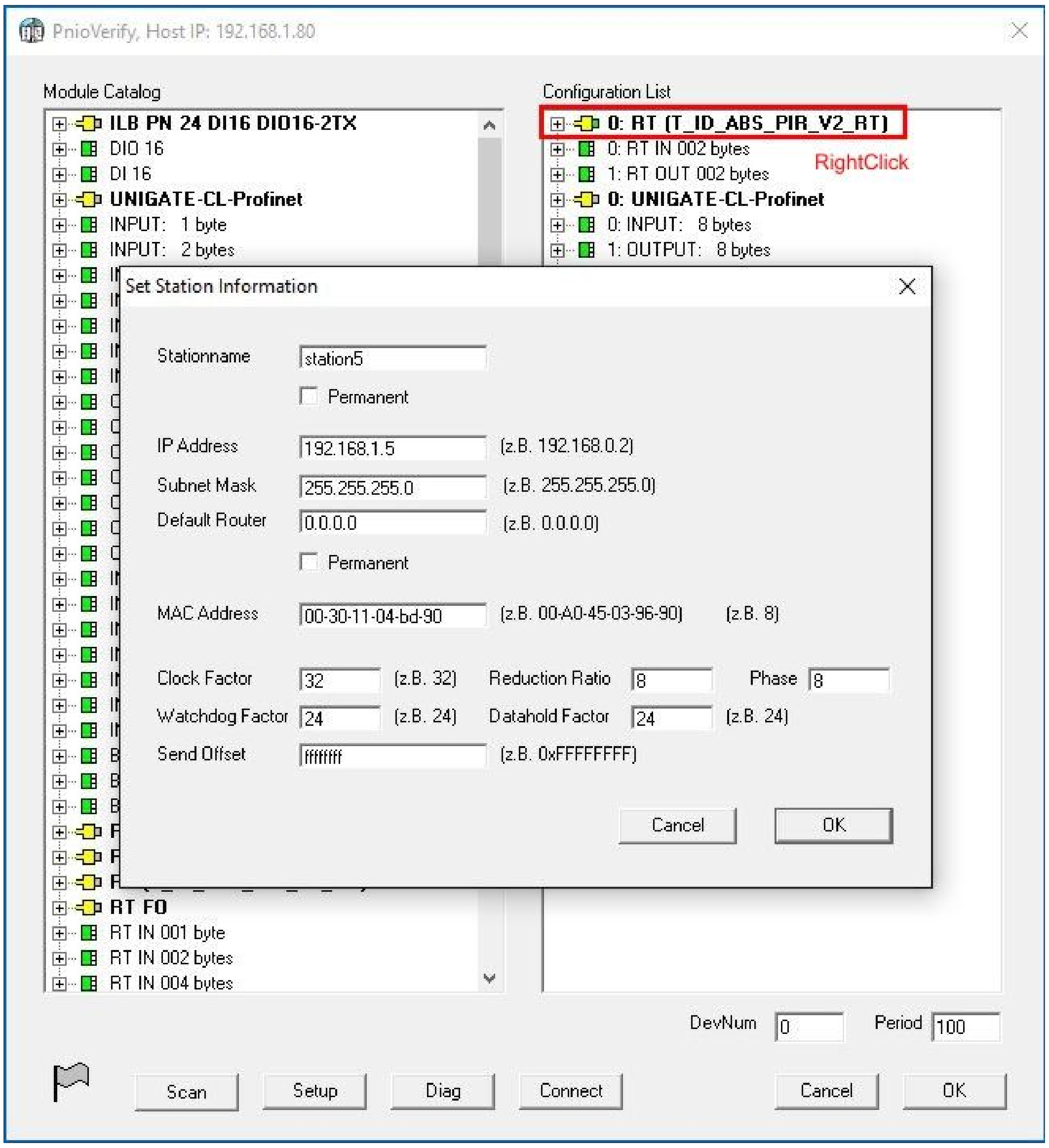
Module Catalog
- ILB PN 24 DI16 DIO16-2TX
- DIO 16
- DI 16
- UNIGATE-CL-Profinet
- INPUT: 1 byte
- INPUT: 2 bytes
- INPUT: 4 bytes
- INPUT: 8 bytes
- INPUT: 16 bytes
- INPUT: 32 bytes
- INPUT: 64 bytes
- OUTPUT: 1 byte
- OUTPUT: 2 bytes
- OUTPUT: 4 bytes
- OUTPUT: 8 bytes
- OUTPUT: 16 bytes
- OUTPUT: 32 bytes
- OUTPUT: 64 bytes
- IN/OUT: 2 bytes
- IN/OUT: 4 bytes
- IN/OUT: 8 bytes
- IN/OUT: 16 bytes
- IN/OUT: 32 bytes
- IN/OUT: 64 bytes
- IN/OUT: 128 bytes
- BIDIR256
- BIDIR512
- BIDIR1024
- RT (T_ID_ABS_PIR_V1)
- RT (FW V>=2.00)
- RT (T_ID_ABS_PIR_V2_RT)
- RT FO
- RT IN 001 byte
- RT IN 002 bytes
- RT IN 004 bytes

Configuration List
- 0: RT (T_ID_ABS_PIR_V2_RT)
- 0: RT IN 002 bytes
- 1: RT OUT 002 bytes
- 0: UNIGATE-CL-Profinet
- 0: INPUT: 8 bytes
- 1: OUTPUT: 8 bytes
- 0: ILB PN 24 DI16 DIO16-2TX
- 0: DIO 16
- 1: DI 16

DevNum 0 Period 100

5.3 Station Settings

The scan gets information about manufacturer name and MAC address. Now individual assignment must set (e.g. IP address, station name, timings). On a right button click at the accesspoint module a dialog appears, which allows setting of station name , IP and timing parameters.



Stationname	station5				
	☐ Permanent				
IP Address	192.168.1.5	(z.B. 192.168.0.2)			
Subnet Mask	255.255.255.0	(z.B. 255.255.255.0)			
Default Router	0.0.0.0	(z.B. 0.0.0.0)			
	☐ Permanent				
MAC Address	00-30-11-04-bd-90	(z.B. 00-A0-45-03-96-90)	(z.B. 8)		
Clock Factor	32	(z.B. 32)	Reduction Ratio	8	Phase 8
Watchdog Factor	24	(z.B. 24)	Datahold Factor	24	(z.B. 24)
Send Offset	ffffffff	(z.B. 0xFFFFFFFF)			

DevNum 0 Period 100

The timing settings of each station are based on a clock unit of 31,25 µsec. The synchronisation period is calculated as follow:

SyncTime = 31,25 µsec * ClockFactor * ReductionRatio

(e.g. 31,25 µsec * 32 * 8 = 8000 µsec = 8 msec)

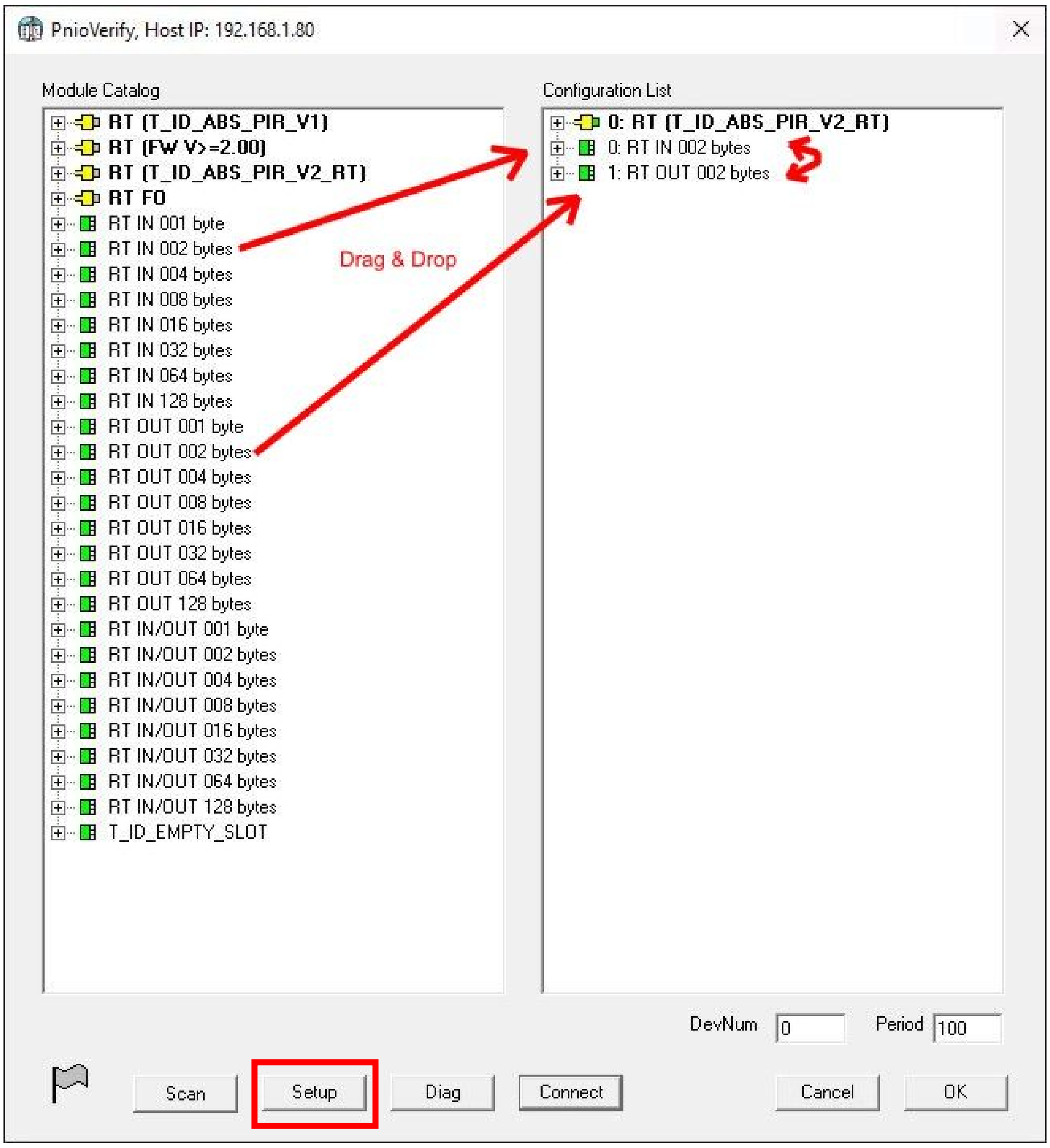
WatchdogTime = SyncTime * WatchdogFactor

(e.g. 8 msec * 24 = 192 msec)

The SendOffset must be set to 0xFFFFFFFF

5.4 Functional Module

Each station typically consists of multiple functional modules (SLOT 1..n). Function Modules have to be inserted from the catalog by DRAG and DROP operations. As well the nmodules may be sorted below the AccessPoint. A station configuration should contain all functional modules (in the order these modules are physically connected). When inserting a new module from the catalog, after dropping, it appears at the end of the configuration list and may be pushed to the correct slot location.


PnioVerify, Host IP: 192.168.1.80

Module Catalog

- RT (T_ID_ABS_PIR_V1)
- RT (FW V>=2.00)
- RT (T_ID_ABS_PIR_V2_RT)
- RT FO
- RT IN 001 byte
- RT IN 002 bytes
- RT IN 004 bytes
- RT IN 008 bytes
- RT IN 016 bytes
- RT IN 032 bytes
- RT IN 064 bytes
- RT IN 128 bytes
- RT OUT 001 byte
- RT OUT 002 bytes
- RT OUT 004 bytes
- RT OUT 008 bytes
- RT OUT 016 bytes
- RT OUT 032 bytes
- RT OUT 064 bytes
- RT OUT 128 bytes
- RT IN/OUT 001 byte
- RT IN/OUT 002 bytes
- RT IN/OUT 004 bytes
- RT IN/OUT 008 bytes
- RT IN/OUT 016 bytes
- RT IN/OUT 032 bytes
- RT IN/OUT 064 bytes
- RT IN/OUT 128 bytes
- T_ID_EMPTY_SLOT

Configuration List

- 0: RT (T_ID_ABS_PIR_V2_RT)
- 0: RT IN 002 bytes
- 1: RT OUT 002 bytes

DevNum: 0
Period: 100

When the settings are done, the station my be initialized by pressing the button [Setup]

The resulting stationlist is stored to a choosen text file (sample):

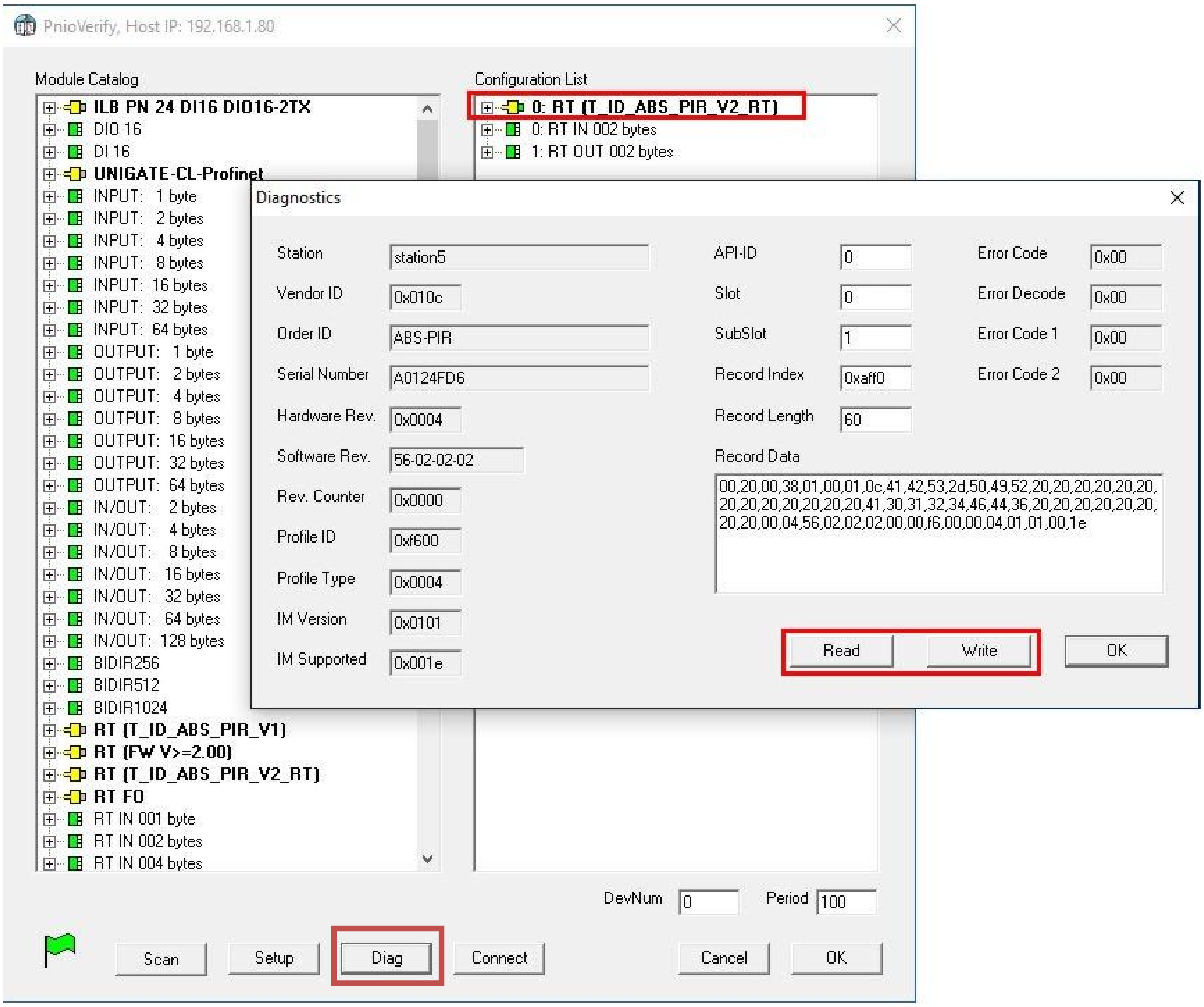
Sample:

```
> Station

[NAME]
station8
[MFG_NAME]
ABS-PIR
[VENDOR_NAME]
HMS Industrial Networks
[STATION_ID]
0c 01 06 00 01 00
[MAC_ADDR]
00 30 11 04 bd 90
[IP_PARAMS]
c0 a8 01 08 ff ff ff 00 00 00 00 00
[TI_PARAMS]
20 00 08 00 08 00 18 00 18 00 ff ff ff ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00

>> Module

[NAME]
T_ID_ABS_PIR_V2_RT
[MOD_ID]
00 00 00 d0
[MOD_TYPE]
00 00
[SLOT_NUM]
00 00

>>> Submodule

[SUBMOD_ID]
01 00 00 00
[SUBSLOT_NUM]
01 00
[OBJ_INPUT]
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 01 00 00 00 01 00 00 00

>>> Submodule

[SUBMOD_ID]
02 00 00 00
[SUBSLOT_NUM]
00 80
[OBJ_INPUT]
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 01 00 00 00 01 00 00 00

>>> Submodule

[SUBMOD_ID]
```

```
03 00 00 00
[SUBSLOT_NUM]
01 80
[OBJ_INPUT]
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                              00 00 00 00 01 00 00 00 01 00 00 00

>>> Submodule

[SUBMOD_ID]
03 00 00 00
[SUBSLOT_NUM]
02 80
[OBJ_INPUT]
01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                              00 00 00 00 01 00 00 00 01 00 00 00

>> Module

[NAME]
T_ID_RT_IN2
[MOD_ID]
02 00 00 00
[MOD_TYPE]
01 00
[SLOT_NUM]
01 00

>>> Submodule

[SUBMOD_ID]
01 00 00 00
[SUBSLOT_NUM]
01 00
[OBJ_INPUT]
01 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                              02 00 00 00 01 00 00 00 01 00 00 00

>> Module

[NAME]
T_ID_RT_OUT2
[MOD_ID]
20 00 00 00
[MOD_TYPE]
01 00
[SLOT_NUM]
02 00

>>> Submodule

[SUBMOD_ID]
01 00 00 00
[SUBSLOT_NUM]
01 00
[OBJ_OUTPUT]
01 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                              02 00 00 00 01 00 00 00 01 00 00 00
```

5.5 Station Diagnostics

The station diagnostics allows gathering of I&M data, as well as reading and writing acyclic information (by API, SlotNum, SubSlotNum and RecordIndex). Therefore select an AccessPoint and press button [Diag].



Module Catalog:

```
ILB PN 24 DI16 DIO16-2TX
DIO 16
DI 16
UNIGATE-CL-Profinet
INPUT:   1 byte
INPUT:   2 bytes
INPUT:   4 bytes
INPUT:   8 bytes
INPUT:  16 bytes
INPUT:  32 bytes
INPUT:  64 bytes
OUTPUT:  1 byte
OUTPUT:  2 bytes
OUTPUT:  4 bytes
OUTPUT:  8 bytes
OUTPUT: 16 bytes
OUTPUT: 32 bytes
OUTPUT: 64 bytes
IN/OUT:  2 bytes
IN/OUT:  4 bytes
IN/OUT:  8 bytes
IN/OUT: 16 bytes
IN/OUT: 32 bytes
IN/OUT: 64 bytes
IN/OUT: 128 bytes
BIDIR256
BIDIR512
BIDIR1024
RT (T_ID_ABS_PIR_V1)
RT (FW V>=2.00)
RT (T_ID_ABS_PIR_V2_RT)
RT FO
RT IN 001 byte
RT IN 002 bytes
RT IN 004 bytes
```

Configuration List:

```
0: RT (T_ID_ABS_PIR_V2_RT)
0: RT IN 002 bytes
1: RT OUT 002 bytes
```

Diagnostics:

Station	station5	API-ID	0	Error Code	0x00
Vendor ID	0x010c	Slot	0	Error Decode	0x00
Order ID	ABS-PIR	SubSlot	1	Error Code 1	0x00
Serial Number	A0124FD6	Record Index	0xaff0	Error Code 2	0x00
Hardware Rev.	0x0004	Record Length	60		
Software Rev.	56-02-02-02				
Rev. Counter	0x0000				
Profile ID	0xf600				
Profile Type	0x0004				
IM Version	0x0101				
IM Supported	0x001e				

Record Data

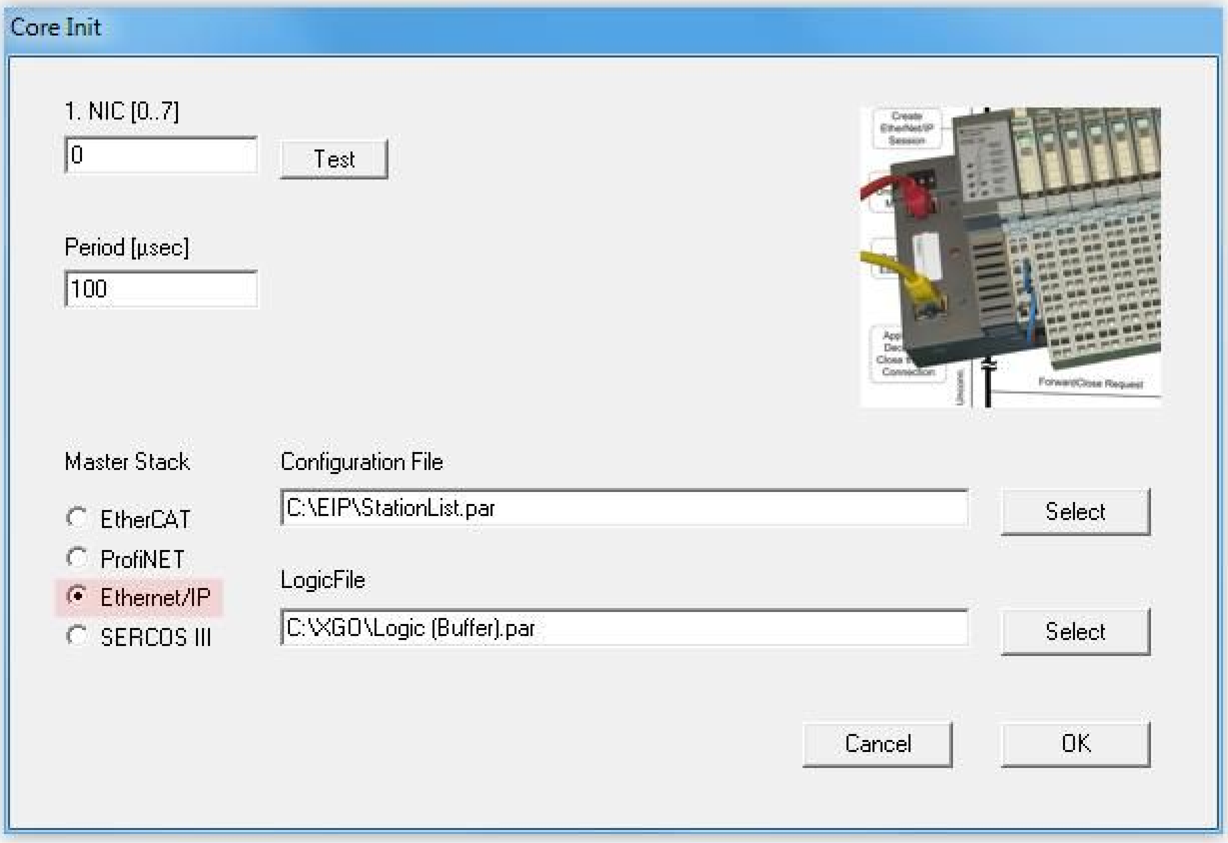
```
00,20,00,38,01,00,01,0c,41,42,53,2d,50,49,52,20,20,20,20,20,20,
20,20,20,20,20,20,20,41,30,31,32,34,46,44,36,20,20,20,20,20,20,
20,20,00,04,56,02,02,02,00,00,f6,00,00,04,01,01,00,1e
```

DevNum 0   Period 100

To read or write acyclic information, put in the API-ID, Slot, SubSlot and Record Index. If the function fails, you'll get the corresponding PNIO error code.

6 Ethernet/IP

When using X-GO SERCOS III following parameters have to be used:


Core Init

1. NIC [0..7]: 0

Period [µsec]: 100

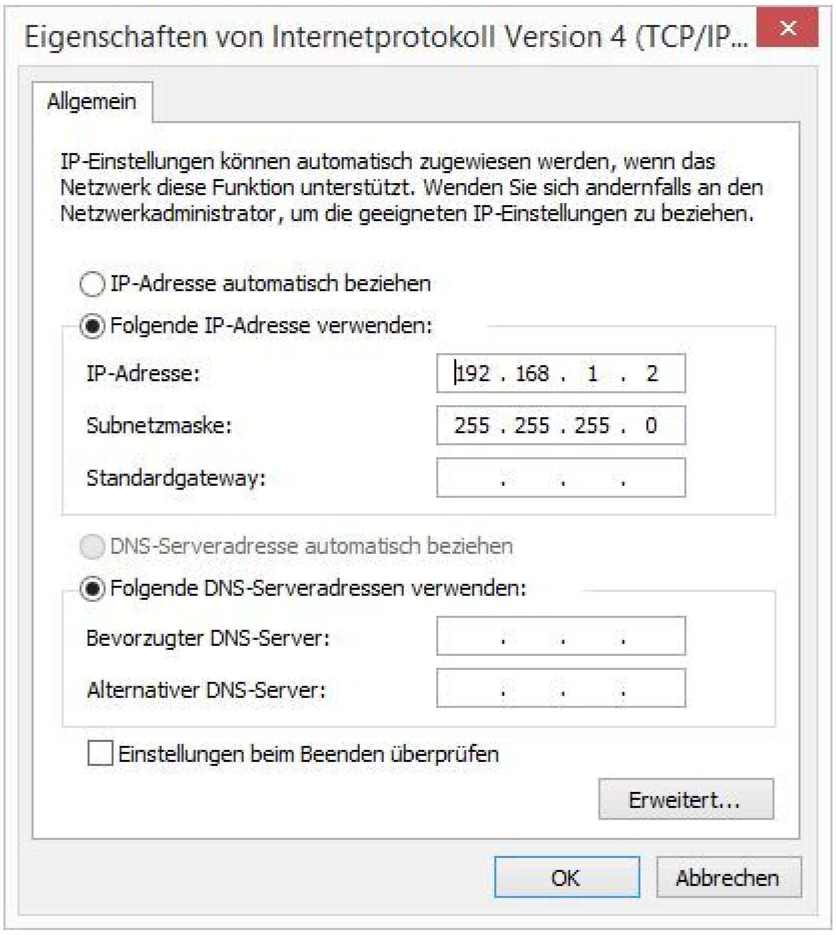
Master Stack: EtherCAT, ProfiNET, [✓ Ethernet/IP], SERCOS III

Configuration File: C:\EIP\StationList.par

LogicFile: C:\XGO\Logic (Buffer).par

1.NIC	Index of the first realtime Ethernet driver (typically 0)
Period	Sampling period (typically 100 µsec)
Configuration File	Path to configuration file (e.g. STATIONLIST.PAR)

Note: ProfiNET requires the Host PC to have a valid IP-Address. So, check the ethnet adapter setting.



Eigenschaften von Internetprotokoll Version 4 (TCP/IP...

Allgemein

IP-Einstellungen können automatisch zugewiesen werden, wenn das Netzwerk diese Funktion unterstützt. Wenden Sie sich andernfalls an den Netzwerkadministrator, um die geeigneten IP-Einstellungen zu beziehen.

○ IP-Adresse automatisch beziehen
● Folgende IP-Adresse verwenden:

IP-Adresse:	192 . 168 . 1 . 2
Subnetzmaske:	255 . 255 . 255 . 0
Standardgateway:	. . .

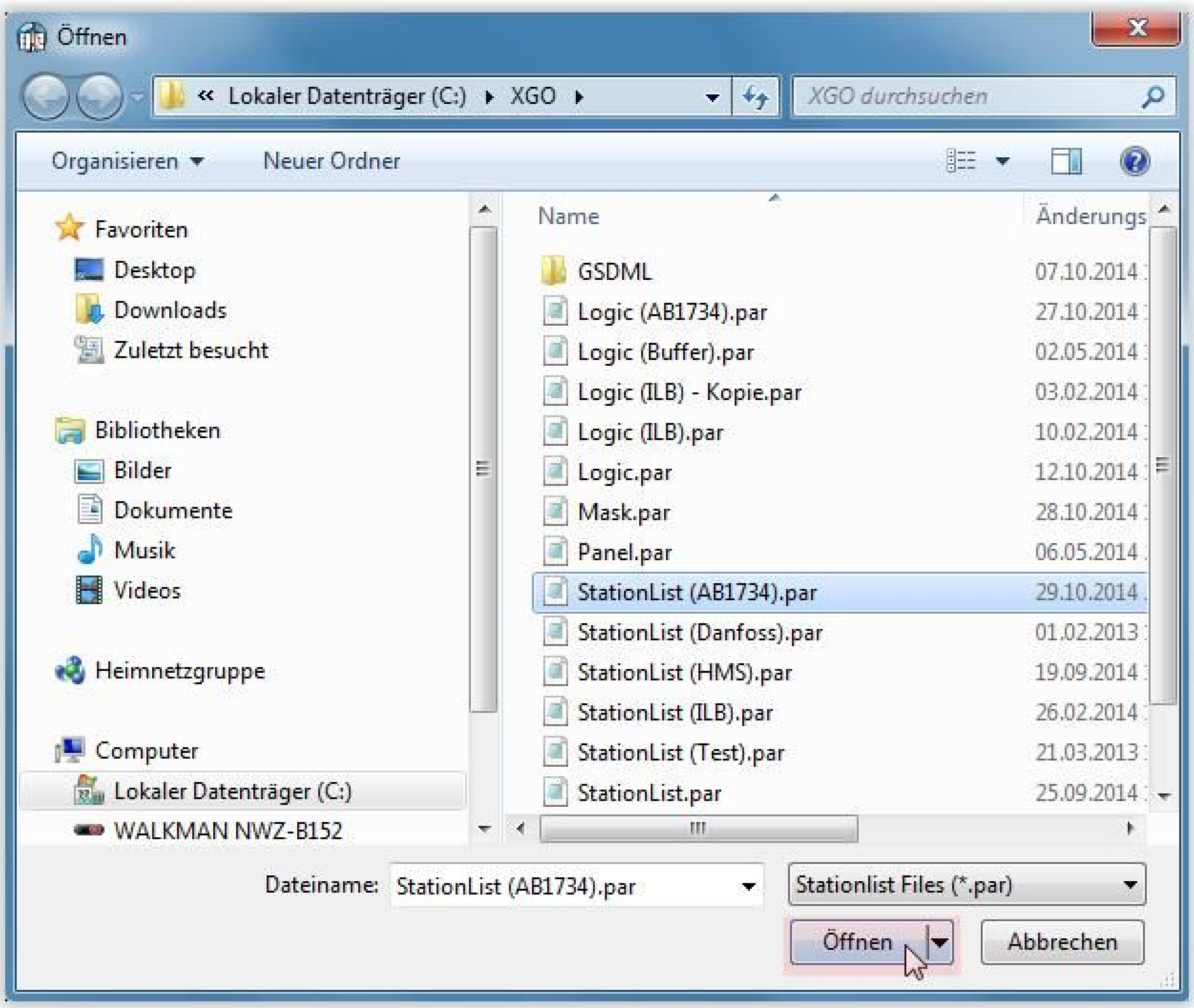
○ DNS-Serveradresse automatisch beziehen
● Folgende DNS-Serveradressen verwenden:

Bevorzugter DNS-Server:	. . .
Alternativer DNS-Server:	. . .

☐ Einstellungen beim Beenden überprüfen

6.1 Creating a configuration file

A Ethernet/IP fieldbus configuration consists typically of several station devices. A station device owns at least one connection. For proper operation the Ethernet/IP device needs first to be configured and a configuration file must be created. Therefore SYBERA provides a program called EIPCONFIG.EXE. When starting the program you will be asked for a already existing configuration file. If a file is present, choose it, otherwise just cancel the dialog.

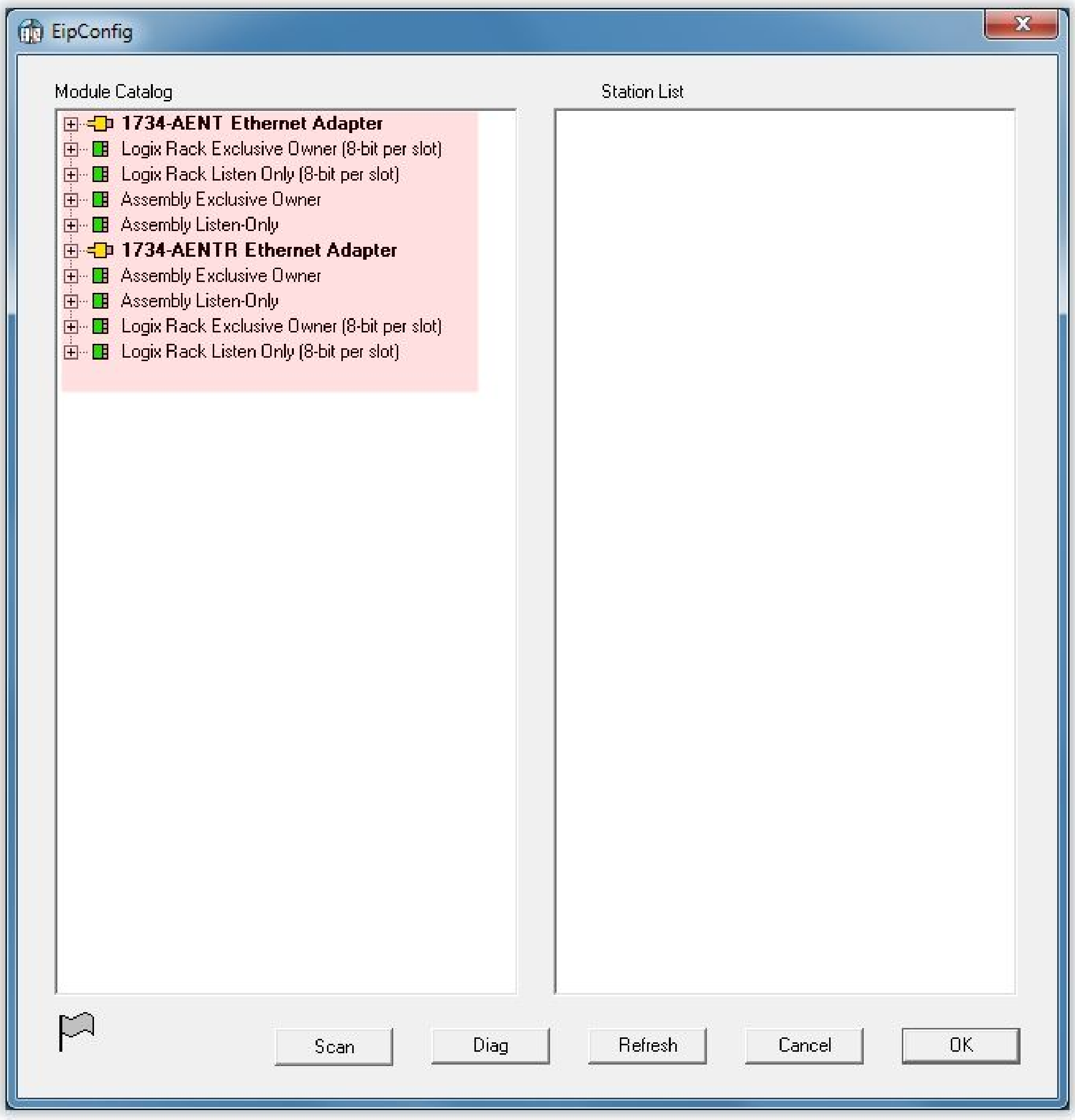


Name	Änderungs
GSDML	07.10.2014
Logic (AB1734).par	27.10.2014
Logic (Buffer).par	02.05.2014
Logic (ILB) - Kopie.par	03.02.2014
Logic (ILB).par	10.02.2014
Logic.par	12.10.2014
Mask.par	28.10.2014
Panel.par	06.05.2014
StationList (AB1734).par	29.10.2014
StationList (Danfoss).par	01.02.2013
StationList (HMS).par	19.09.2014
StationList (ILB).par	26.02.2014
StationList (Test).par	21.03.2013
StationList.par	25.09.2014

Dateiname: StationList (AB1734).par
Stationlist Files (*.par)

Note: If the program fails to run, check if the latest Microsoft XML Parser has been installed. If not, install in the directory \APP\MSXML\MSXML6

EIPCONFIG allows creating a configuration file by selecting devices and corresponding connections from a module catalogue (left side view). The catalogue gets its entries by the provided ESD files. A module is inserted to the configuration (right side view) by a DRAG and DROP operation (just drag a module from the catalogue to the station list configuration). There are two types of modules:



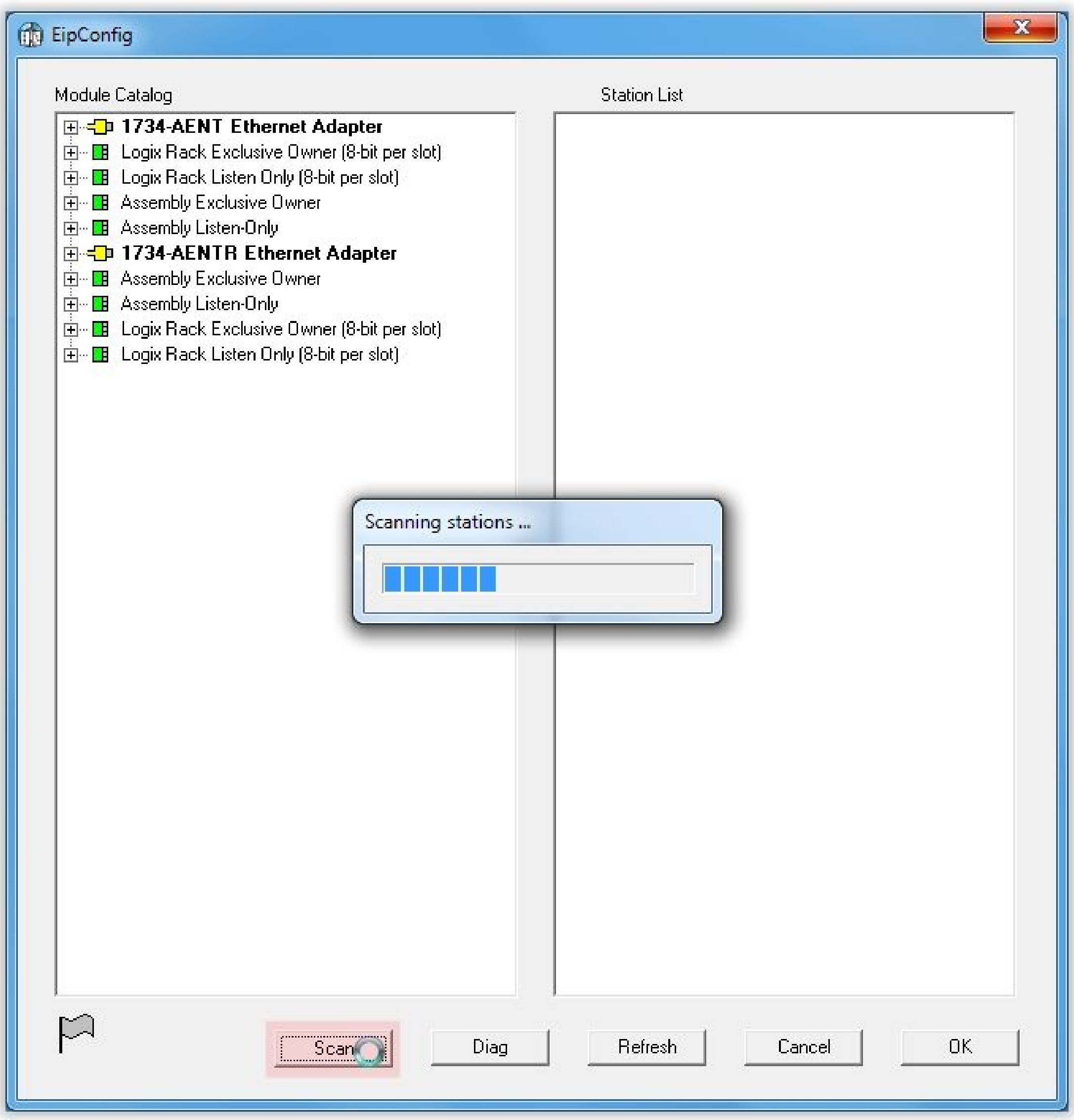
1734-AENT Ethernet Adapter
Logix Rack Exclusive Owner (8-bit per slot)
Logix Rack Listen Only (8-bit per slot)
Assembly Exclusive Owner
Assembly Listen-Only
1734-AENTR Ethernet Adapter
Assembly Exclusive Owner
Assembly Listen-Only
Logix Rack Exclusive Owner (8-bit per slot)
Logix Rack Listen Only (8-bit per slot)

dule

Module

6.1.1 Device Module

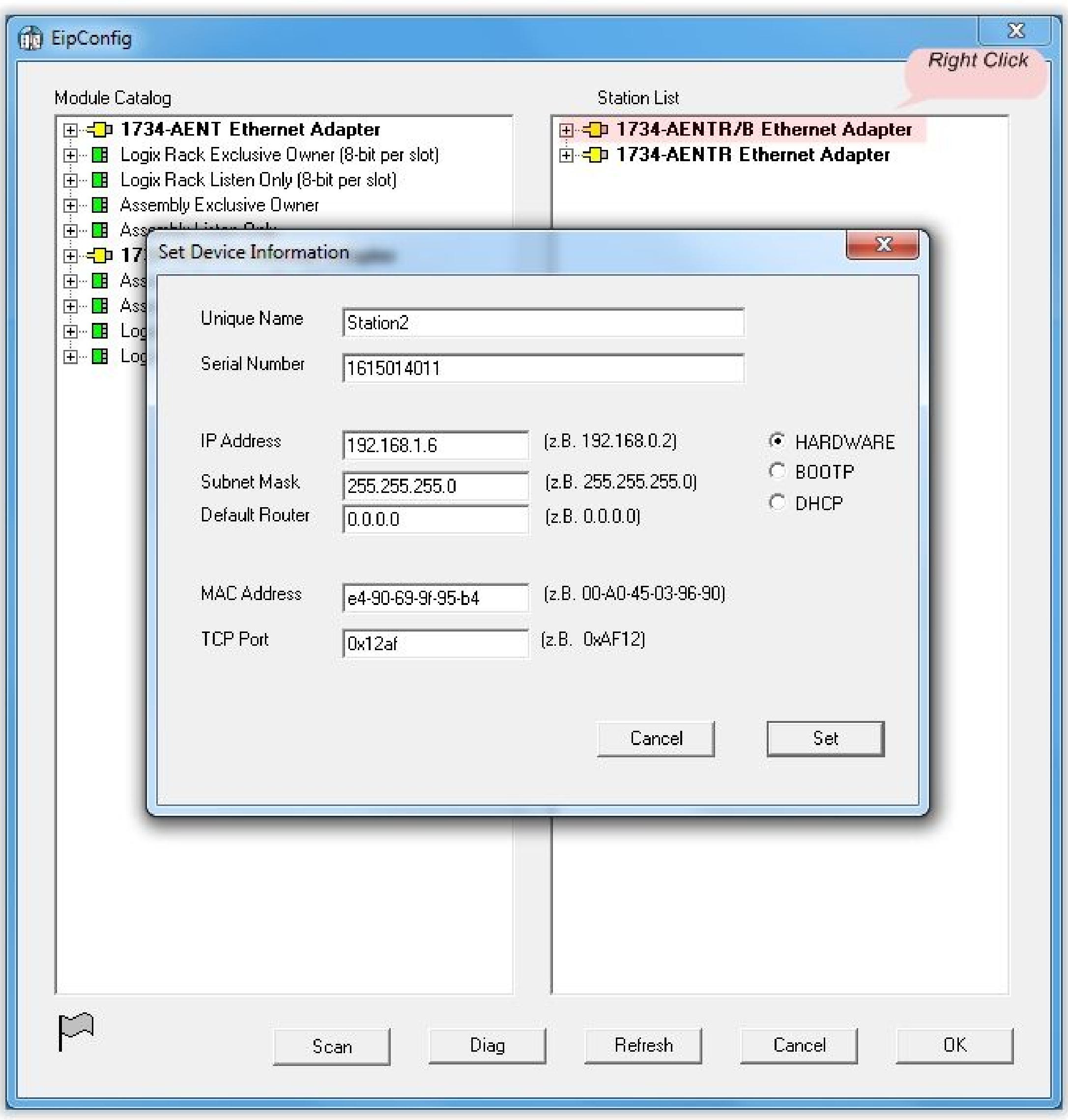
The device module keeps all information required for identifying to the fieldbus, such as station name, product code IP parameters, MAC address. Therefore, the first task is to collect information about the available devices by scanning the bus.



1734-AENT Ethernet Adapter
Logix Rack Exclusive Owner (8-bit per slot)
Logix Rack Listen Only (8-bit per slot)
Assembly Exclusive Owner
Assembly Listen-Only
1734-AENTR Ethernet Adapter
Assembly Exclusive Owner
Assembly Listen-Only
Logix Rack Exclusive Owner (8-bit per slot)
Logix Rack Listen Only (8-bit per slot)

6.1.2 Device Settings

The scan gets all information to identify the device. Now individual assignment must set (e.g. IP address and station name). On a right button click at the device module the "Set Device Information" dialog appears.

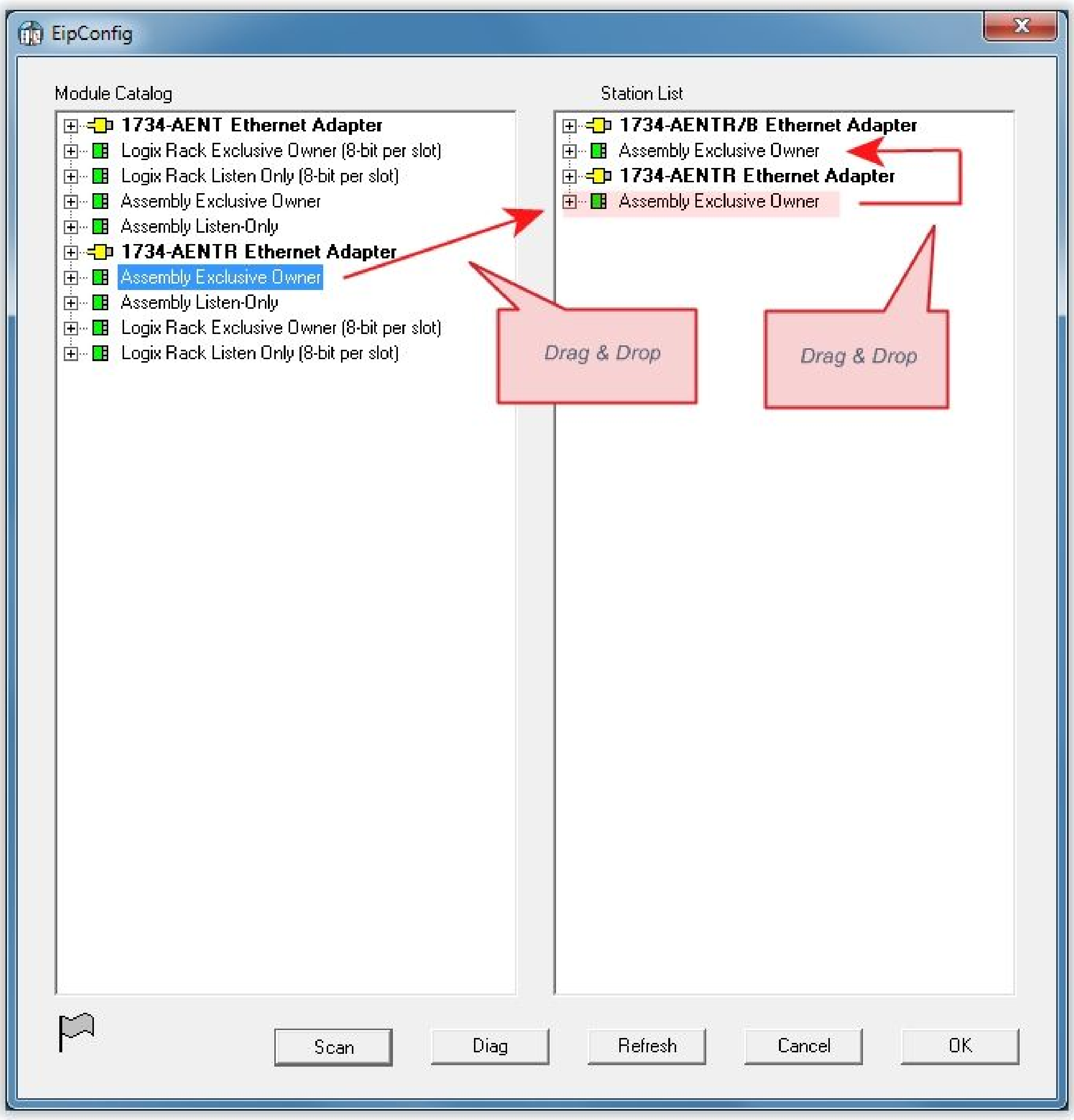

EipConfig
Right Click
Module Catalog
1734-AENT Ethernet Adapter
Logix Rack Exclusive Owner (8-bit per slot)
Logix Rack Listen Only (8-bit per slot)
Assembly Exclusive Owner
Station List
1734-AENTR/B Ethernet Adapter
1734-AENTR Ethernet Adapter
Set Device Information
Cancel
Set
Scan
Diag
Refresh
Cancel
OK

Unique Name	Station2	
Serial Number	1615014011	
IP Address	192.168.1.6	(z.B. 192.168.0.2)
Subnet Mask	255.255.255.0	(z.B. 255.255.255.0)
Default Router	0.0.0.0	(z.B. 0.0.0.0)
MAC Address	e4-90-69-9f-95-b4	(z.B. 00-A0-45-03-96-90)
TCP Port	0x12af	(z.B. 0xAF12)

[✓ HARDWARE] BOOTP DHCP

6.1.3 Connection Module

Each device typically owns at least one connection module. Connection modules have to be inserted from the catalogue by DRAG and DROP operations. A station configuration must contain all functional modules in the correct order (the modules will be addressed in this order). When inserting a module from the catalogue, after dropping, it appears at the end of the configuration list and must be pushed to the correct location.



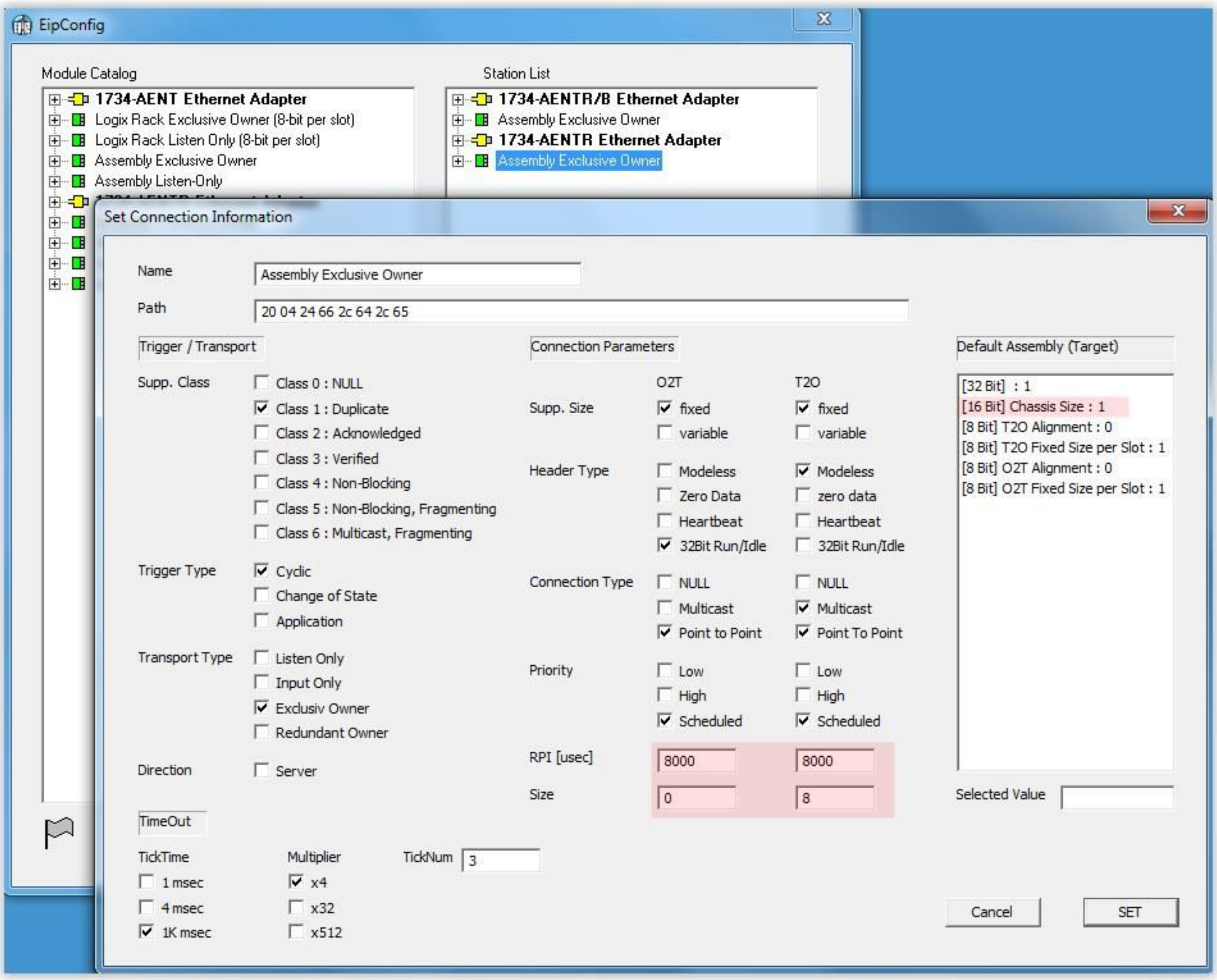
Module Catalog:
- 1734-AENT Ethernet Adapter
- Logix Rack Exclusive Owner (8-bit per slot)
- Logix Rack Listen Only (8-bit per slot)
- Assembly Exclusive Owner
- Assembly Listen-Only
- 1734-AENTR Ethernet Adapter
- Assembly Exclusive Owner
- Assembly Listen-Only
- Logix Rack Exclusive Owner (8-bit per slot)
- Logix Rack Listen Only (8-bit per slot)

Station List:
- 1734-AENTR/B Ethernet Adapter
- Assembly Exclusive Owner
- 1734-AENTR Ethernet Adapter
- Assembly Exclusive Owner

6.1.4 Connection Settings

On a right button click at the device module the "Set Connection Information" dialog appears. Within the connection typically 3 elements need to be adjusted:

- RPI (Requested Packet Interval)
- Payload Size
- Chassis Size



EipConfig

Module Catalog
- 1734-AENT Ethernet Adapter
- Logix Rack Exclusive Owner (8-bit per slot)
- Logix Rack Listen Only (8-bit per slot)
- Assembly Exclusive Owner
- Assembly Listen-Only

Station List
- 1734-AENTR/B Ethernet Adapter
- Assembly Exclusive Owner
- 1734-AENTR Ethernet Adapter
- Assembly Exclusive Owner

Set Connection Information

Name: Assembly Exclusive Owner

Path: 20 04 24 66 2c 64 2c 65

Trigger / Transport

Supp. Class: Class 0 : NULL; [✓ Class 1 : Duplicate]; Class 2 : Acknowledged; Class 3 : Verified; Class 4 : Non-Blocking; Class 5 : Non-Blocking, Fragmenting; Class 6 : Multicast, Fragmenting

Trigger Type: [✓ Cyclic]; Change of State; Application

Transport Type: Listen Only; Input Only; [✓ Exclusiv Owner]; Redundant Owner

Direction: Server

Connection Parameters

	O2T	T2O
Supp. Size	[✓ fixed] variable	[✓ fixed] variable
Header Type	Modeless, Zero Data, Heartbeat, [✓ 32Bit Run/Idle]	[✓ Modeless], zero data, Heartbeat, 32Bit Run/Idle
Connection Type	NULL, Multicast, [✓ Point to Point]	NULL, [✓ Multicast], [✓ Point To Point]
Priority	Low, High, [✓ Scheduled]	Low, High, [✓ Scheduled]
RPI [usec]	8000	8000
Size	0	8

Default Assembly (Target)

[32 Bit] : 1
[16 Bit] Chassis Size : 1
[8 Bit] T2O Alignment : 0
[8 Bit] T2O Fixed Size per Slot : 1
[8 Bit] O2T Alignment : 0
[8 Bit] O2T Fixed Size per Slot : 1

Selected Value:

TimeOut

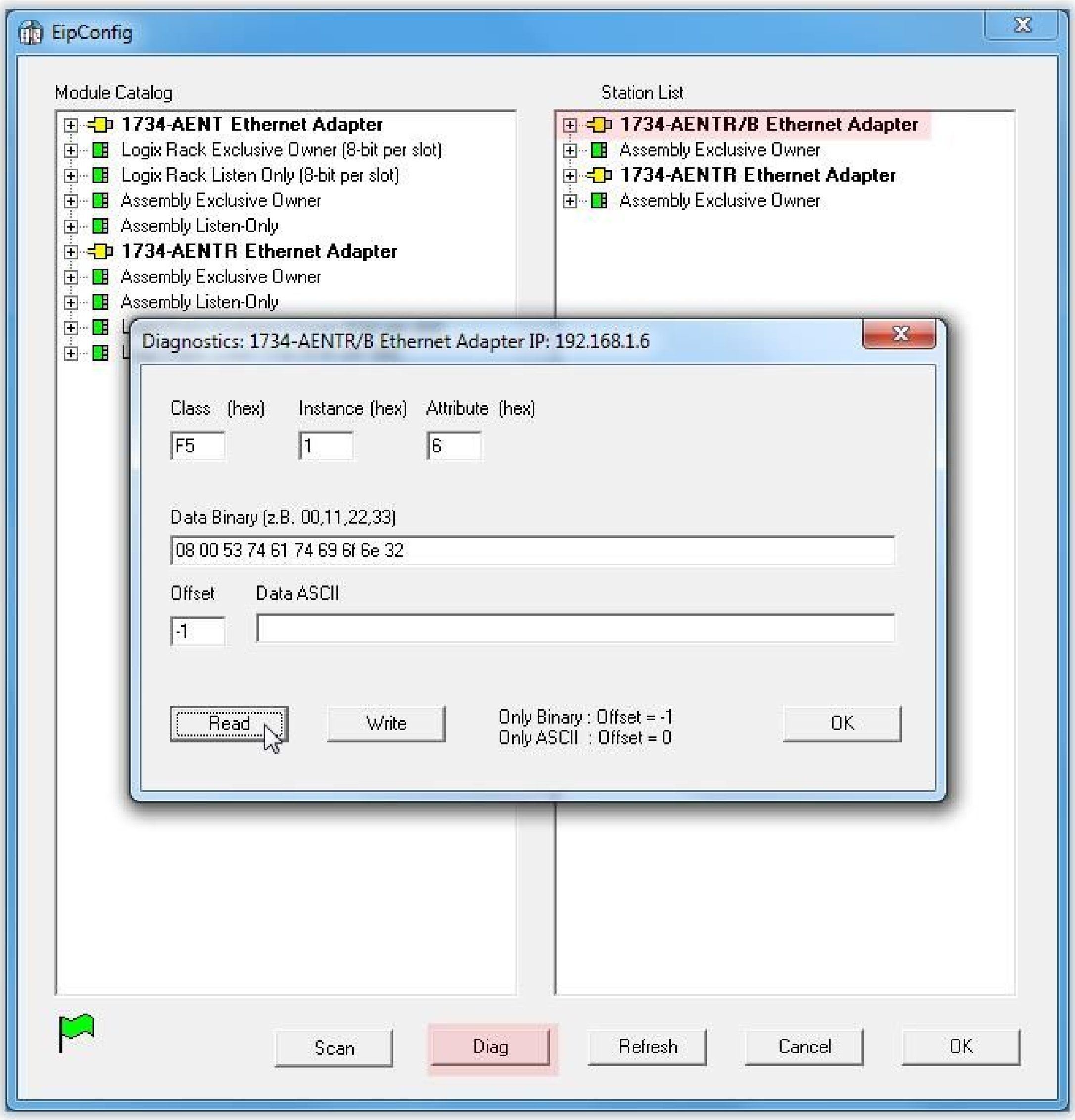
TickTime: 1 msec; 4 msec; [✓ 1K msec]

Multiplier: [✓ x4]; x32; x512

TickNum: 3

6.1.5 Device Diagnostics

The device diagnostics allows gathering all available attributes. Therefore, select a device within the configuration and press button [DIAG]. Now, the attribute information can be read or write to the device by its class ID, instance ID and attribute ID. For instance, reading or writing the device name has class ID : 0xF5, instance ID: 0x01 and attribute ID: 0x06 (see Ethernet/IP specification).



Diagnostics: 1734-AENTR/B Ethernet Adapter IP: 192.168.1.6

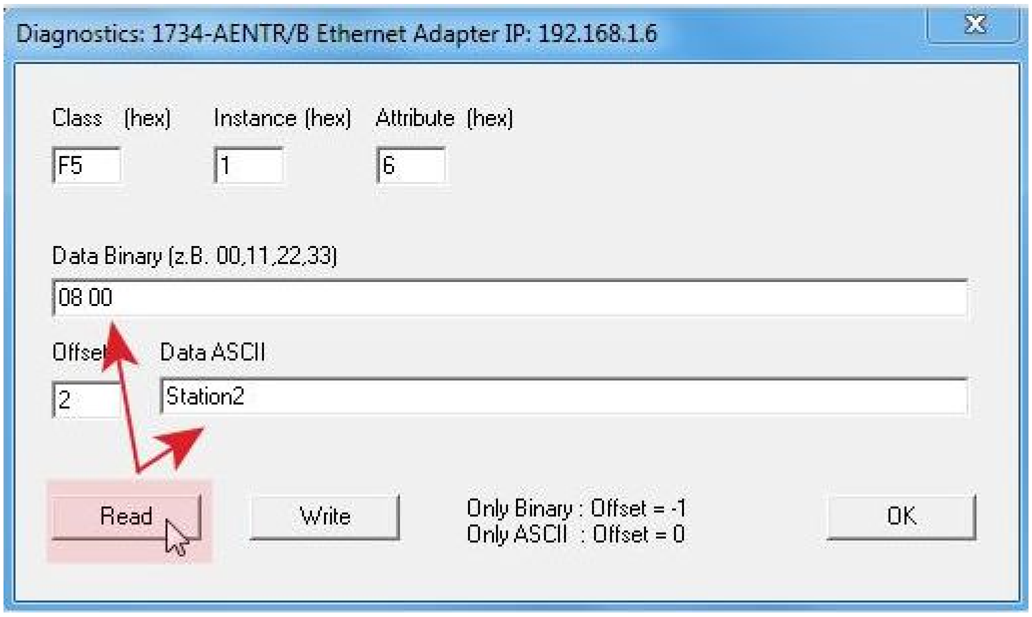
Class (hex): F5
Instance (hex): 1
Attribute (hex): 6

Data Binary (z.B. 00,11,22,33): 08 00 53 74 61 74 69 6f 6e 32

Offset: -1
Data ASCII:

Only Binary : Offset = -1
Only ASCII : Offset = 0

Reading of attributes will be shown as binary data. Some attributes consist of a structured data. For instance string data consist typically of 2 bytes size trailing information and the ASCII chars following. Therefore an offset can be set, to separate these structure elements.



Diagnostics: 1734-AENTR/B Ethernet Adapter IP: 192.168.1.6

Class (hex): F5

Instance (hex): 1

Attribute (hex): 6

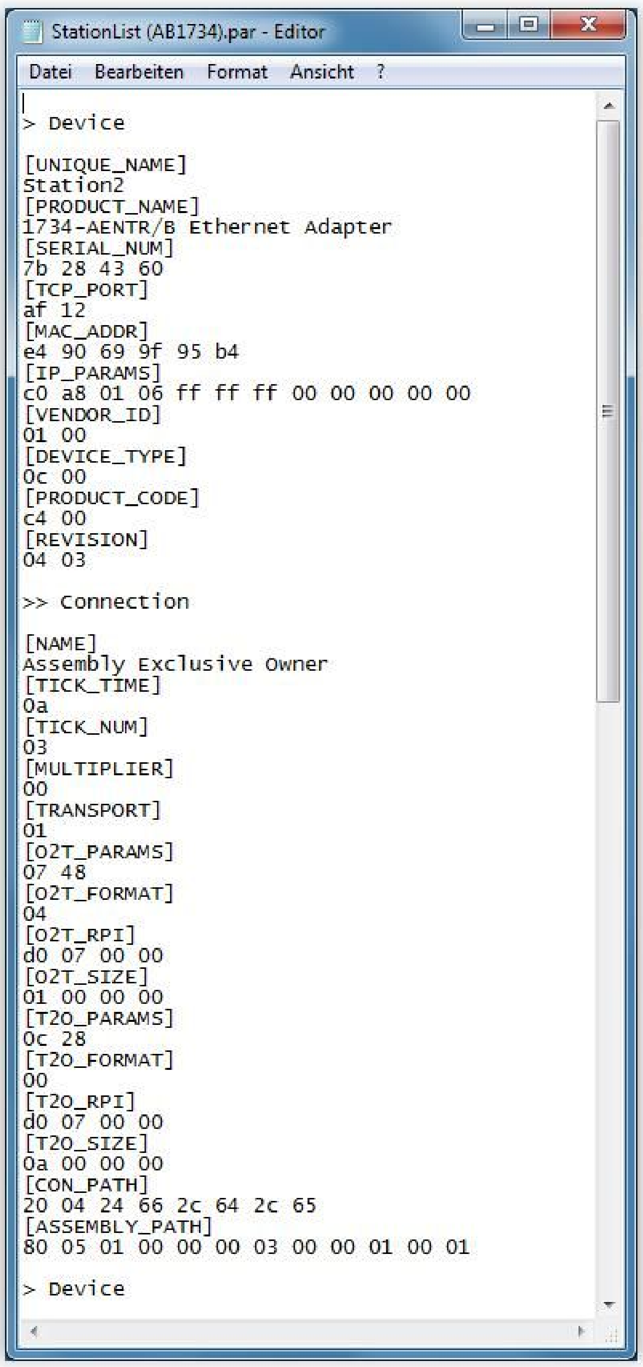
Data Binary (z.B. 00,11,22,33): 08 00

Offset: 2

Data ASCII: Station2

Only Binary : Offset = -1
Only ASCII : Offset = 0

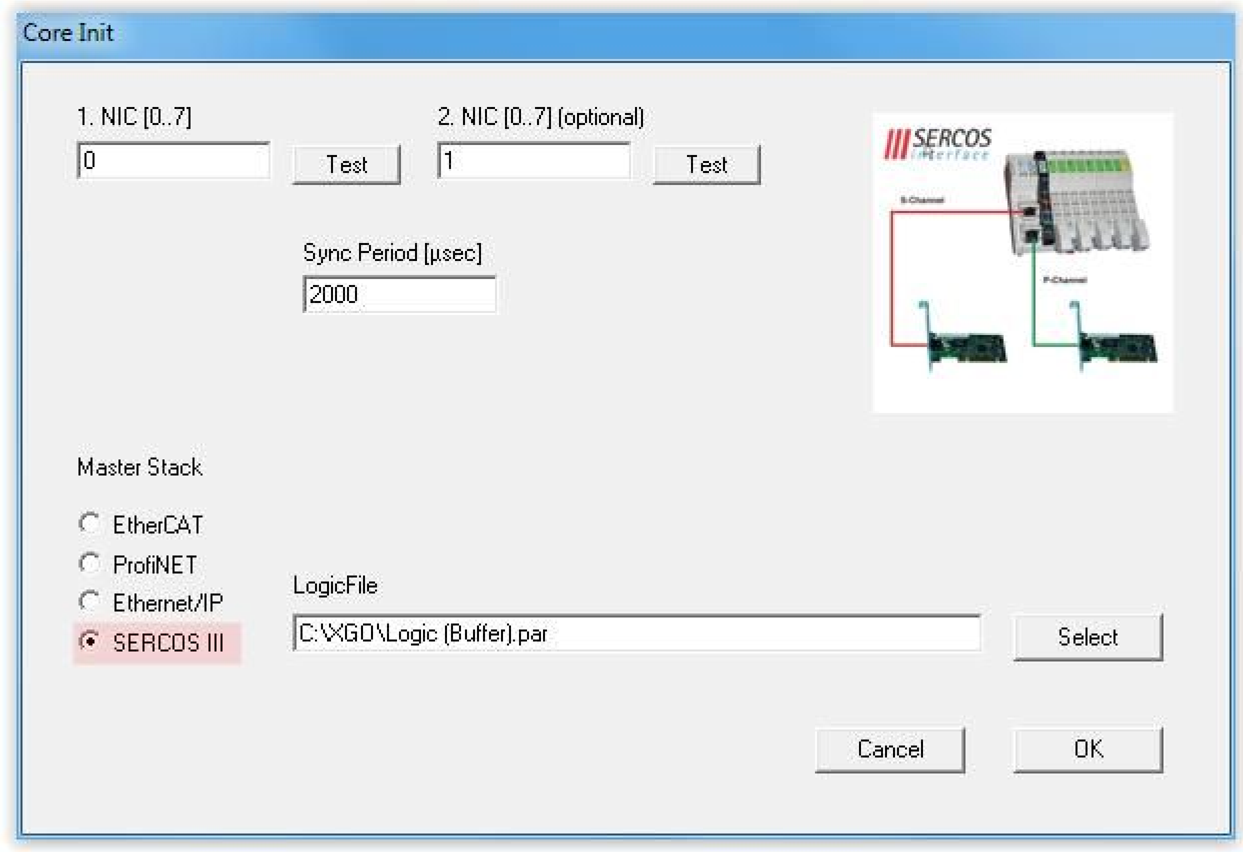
When pressing the button [OK] the configuration file will be saved. The resulting configuration is stored in a text file which must be provided to the Ethernet/IP Master for operation.


StationList (AB1734).par - Editor
Datei Bearbeiten Format Ansicht ?

```
> Device

[UNIQUE_NAME]
Station2
[PRODUCT_NAME]
1734-AENTR/B Ethernet Adapter
[SERIAL_NUM]
7b 28 43 60
[TCP_PORT]
af 12
[MAC_ADDR]
e4 90 69 9f 95 b4
[IP_PARAMS]
c0 a8 01 06 ff ff ff 00 00 00 00 00
[VENDOR_ID]
01 00
[DEVICE_TYPE]
0c 00
[PRODUCT_CODE]
c4 00
[REVISION]
04 03

>> Connection

[NAME]
Assembly Exclusive Owner
[TICK_TIME]
0a
[TICK_NUM]
03
[MULTIPLIER]
00
[TRANSPORT]
01
[O2T_PARAMS]
07 48
[O2T_FORMAT]
04
[O2T_RPI]
d0 07 00 00
[O2T_SIZE]
01 00 00 00
[T2O_PARAMS]
0c 28
[T2O_FORMAT]
00
[T2O_RPI]
d0 07 00 00
[T2O_SIZE]
0a 00 00 00
[CON_PATH]
20 04 24 66 2c 64 2c 65
[ASSEMBLY_PATH]
80 05 01 00 00 00 03 00 00 01 00 01

> Device
```

7 SERCOS III

When using X-GO SERCOS III following parameters have to be used:



Core Init

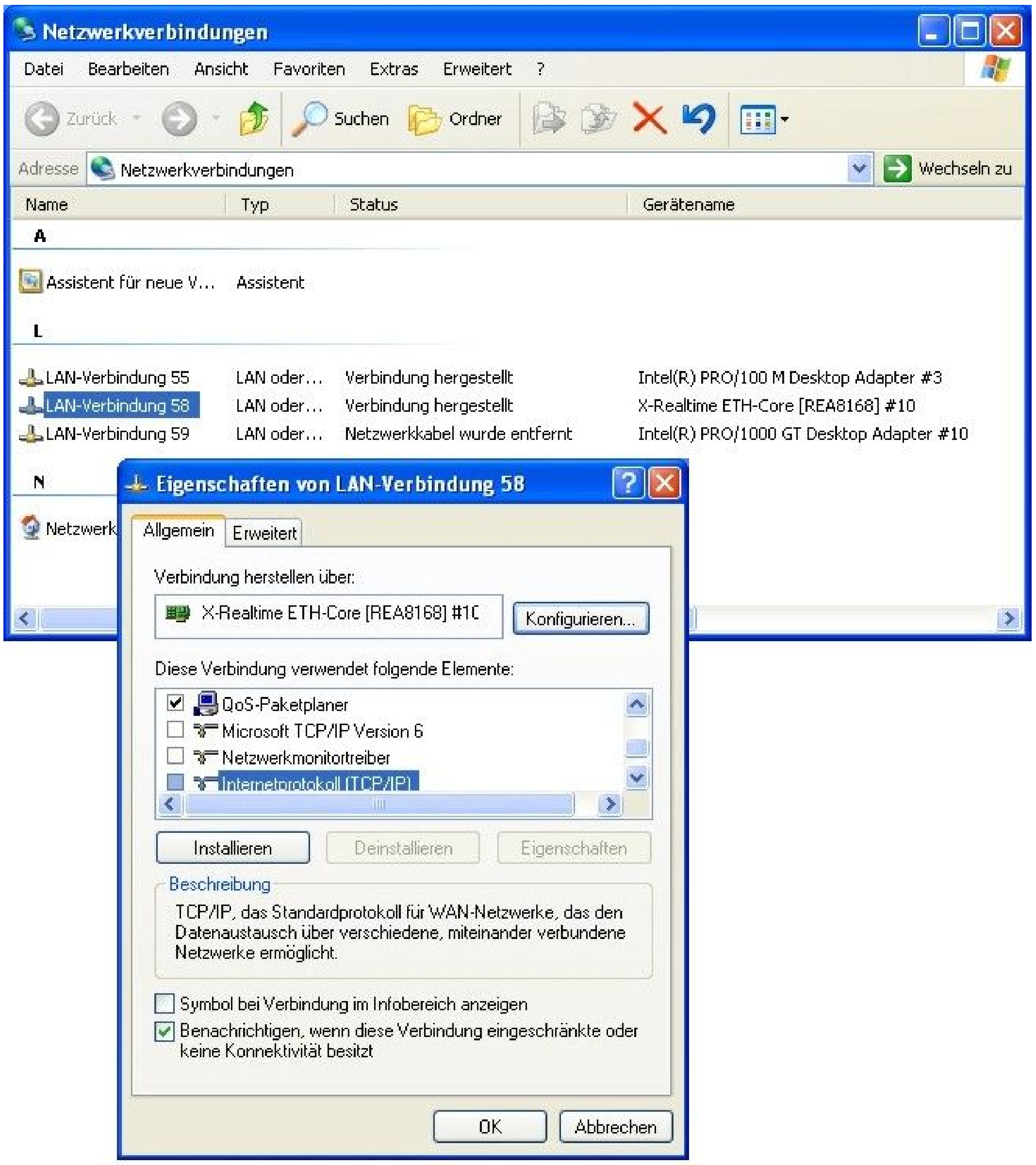
1. NIC [0..7]: 0
2. NIC [0..7] (optional): 1
Sync Period [µsec]: 2000

Master Stack
( ) EtherCAT
( ) ProfiNET
( ) Ethernet/IP
(● SERCOS III)

LogicFile: C:\XGO\Logic (Buffer).par

1.NIC	Index of the first realtime Ethernet driver (typically 0)
2.NIC	Index of the second realtime Ethernet driver (typically 1)
Sync Period	Synchronisation (Update) Period (typically 2000 µsec)

Note: For proper operation of Sercos III its recommended to use the SERCOS III network as a standalone network. This requires to turn off the Windows protocols for this network connection:


Netzwerkverbindungen
Datei Bearbeiten Ansicht Favoriten Extras Erweitert ?
Zurück Suchen Ordner
Adresse Netzwerkverbindungen Wechseln zu
Eigenschaften von LAN-Verbindung 58
Allgemein Erweitert
Konfigurieren...
Installieren Deinstallieren Eigenschaften
OK Abbrechen

Name	Typ	Status	Gerätename
A			
Assistent für neue V...	Assistent		
L			
LAN-Verbindung 55	LAN oder...	Verbindung hergestellt	Intel(R) PRO/100 M Desktop Adapter #3
LAN-Verbindung 58	LAN oder...	Verbindung hergestellt	X-Realtime ETH-Core [REA8168] #10
LAN-Verbindung 59	LAN oder...	Netzwerkkabel wurde entfernt	Intel(R) PRO/1000 GT Desktop Adapter #10
N			
Netzwerk			

Verbindung herstellen über:
X-Realtime ETH-Core [REA8168] #1C

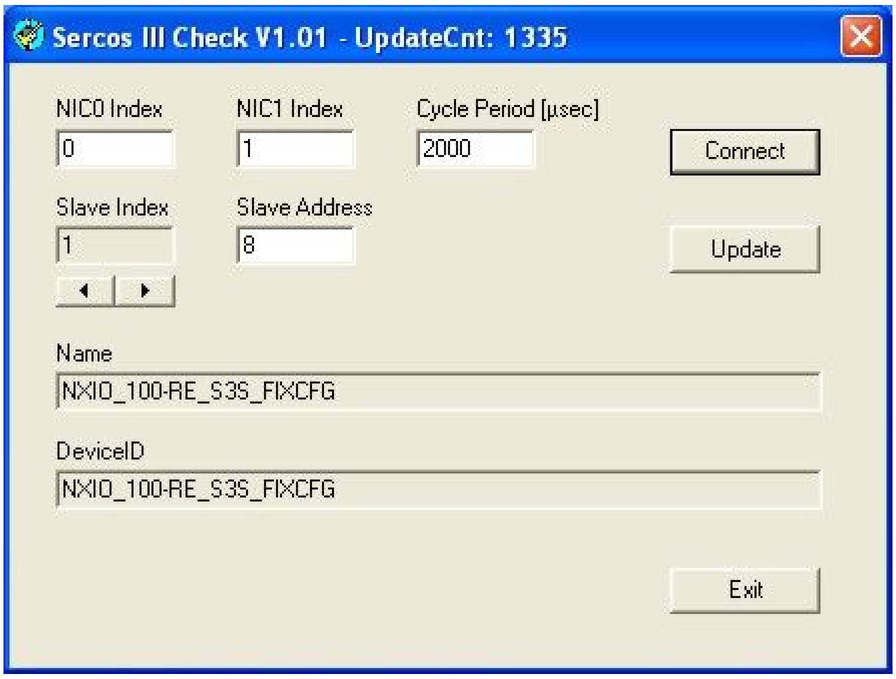
Diese Verbindung verwendet folgende Elemente:
- [✓] QoS-Paketplaner
- [ ] Microsoft TCP/IP Version 6
- [ ] Netzwerkmonitortreiber
- [ ] Internetprotokoll (TCP/IP)

Beschreibung
TCP/IP, das Standardprotokoll für WAN-Netzwerke, das den Datenaustausch über verschiedene, miteinander verbundene Netzwerke ermöglicht.

- [ ] Symbol bei Verbindung im Infobereich anzeigen
- [✓] Benachrichtigen, wenn diese Verbindung eingeschränkte oder keine Konnektivität besitzt

7.1 Sercos III Check

The tool SC3CHECK allows the adjustment of Addresses for Sercos III modules


Sercos III Check V1.01 - UpdateCnt: 1335

NIC0 Index: 0
NIC1 Index: 1
Cycle Period [µsec]: 2000
Slave Index: 1
Slave Address: 8
Name: NXIO_100-RE_S3S_FIXCFG
DeviceID: NXIO_100-RE_S3S_FIXCFG

7.2 Configuration Parameter File

Usually device information is provided by a corresponding XML configuration file. Since the development of software with the SERCOS III Master Library has special needs for programming, the XML file must be parsed and translated into a native format. Therefore the SERCOS III Master Library provides a configuration file called **SC3CONFIG.PAR**, which is located in the directory **\windows\system32** after installation. The SC3CONFIG.PAR is a text based file with sections for Device Name, Device ID, configuration parameters and data description. All sections of a device must follow the item order:

NAME -> DEVICEID -> PARAM -> CONFIG -> DESC

```
[NAME]
HCS01.1E-W0003-A-02-E-S3-EC-NN-NN-NN-FW
[DEVICEID]
FWA-INDRV*-MPE-16
[PARAM]
S-0-32.0.0 (02 00)
[CONFIG]
S-0-0134.0.0  <0,0>
S-0-0036.0.0  <0,0>
S-0-0047.0.0  <0,0>
S-0-0135.0.0  <1,1>
S-0-0040.0.0  <1,1>
S-0-0386.0.0  <1,1>
[DESC]
00 00 01 01 02 00
00 00 04 02 02 00
00 00 04 03 04 00
00 00 04 03 04 00
00 01 01 01 02 00
00 01 04 02 02 00
00 01 04 03 04 00
00 01 04 03 04 00
```

7.2.1 Section [NAME]

This section contains the name of the device:

```
[NAME]
HCS01.1E-W0003-A-02-E-S3-EC-NN-NN-NN-FW
```

7.2.2 Section [DEVICEID]

This section contains the device ID of the station:

```
[DEVICEID]
FWA-INDRV*-MPE-16
```

7.2.3 Section [PARAM]

This section contains the IDN and binary data for configuration:

```
[PARAM]
S-0-32.0.0 (02 00)
```

7.2.4 Section [CONFIG]

This section contains the IDN list for the connections:

```
[CONFIG]
S-0-0134.0.0  <0,0>
S-0-0036.0.0  <0,0>
S-0-0047.0.0  <0,0>
S-0-0135.0.0  <1,1>
S-0-0040.0.0  <1,1>
S-0-0386.0.0  <1,1>


IDN
|              Connection Index
|              | Telegam Type
|              | |
S-0-0135.0.0  <1,1>
```

7.2.5 Section [DESC]

This section contains the output/input data description of the device:

```
[DESC]
00 00 01 01 02 00
00 00 04 02 02 00
00 00 04 03 04 00
00 00 04 03 04 00
00 01 01 01 02 00
00 01 04 02 02 00
00 01 04 03 04 00
00 01 04 03 04 00
```

```
Telegram Type   00 : MDT (Output)
     01 : AT (Input)

Data Item   00 : DATA_ITEM_NONE
     01 : DATA_ITEM_CCON
     02 : DATA_ITEM_IOCTRL
     03 : DATA_ITEM_IOSTAT
     04 : DATA_ITEM_VALUE
     05 : DATA_ITEM_SCALE
     06 : DATA_ITEM_DIAG
     07 : DATA_ITEM_NAME

Data Type   00 : DATA_TYPE_NONE
     01 : DATA_TYPE_U8
     02 : DATA_TYPE_U16
     03 : DATA_TYPE_U32
     04 : DATA_TYPE_U64
     05 : DATA_TYPE_I8
     06 : DATA_TYPE_I16
     07 : DATA_TYPE_I32
     08 : DATA_TYPE_I64
     09 : DATA_TYPE_F32
     10 : DATA_TYPE_F64



Connection Index
|  Telegram Type
|  |  Data Item
|  |  |  Data Type
|  |  |  |  Data Len
|  |  |  |  |     |
 00 00 01 01 02 00 <- Desc 0
 00 00 02 01 02 00 <- Desc 1
 00 00 04 02 02 00 <- Desc 2
 00 01 01 01 02 00 <- Desc 3
 00 01 03 01 02 00 <- Desc 4
 00 01 04 02 02 00 <- Desc 5
 00 01 04 02 02 00 <- Desc 6
```

8 Error Handling

X-GO and all implemented Master Stacks provides an error handling and tracing mechanism.

8.1 Debug LOG File

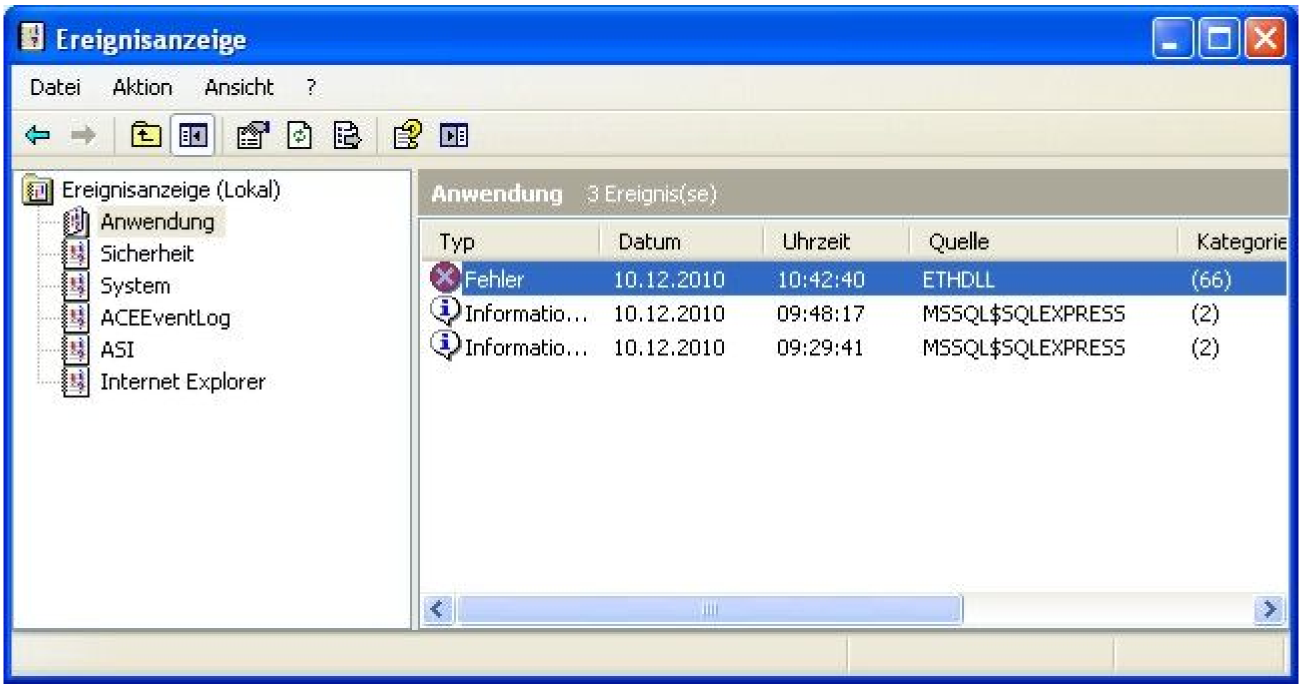
On execution X-GO creates sequence files in Text-Format:

EtherCAT:	ECATDBG.LOG
ProfiNET:	PNTDBG.LOG
Ethernet/IP:	EIPDBG.LOG
Sercos III:	SC3DBG.LOG

Note: These files are not accessible while X-GO is running

8.2 Event File

On execution X-GO logs error event to the Windows Event Manager. The X-GO software logs Application and System events. These events can be exported to a file and provided for support purposes.



Typ	Datum	Uhrzeit	Quelle	Kategorie
Fehler	10.12.2010	10:42:40	ETHDLL	(66)
Informatio...	10.12.2010	09:48:17	MSSQL$SQLEXPRESS	(2)
Informatio...	10.12.2010	09:29:41	MSSQL$SQLEXPRESS	(2)