Connect

short Connect()

Return value

0 if no EasyPort is found, otherwise the module numbers of the detected EasyPorts

Module 1:	Return value 1
Module 2:	Return value 2
Module 3:	Return value 4
Module 4:	Return value 8

The values are added up if several EasyPorts are detected.

Description

All of the interfaces within the system are checked for connected EasyPorts. When connection is established with the *Connect* method, the EasyPort modules are initialised with the following EasyPort commands:

setup0	Basic initialisation
MT<x>=01	Activate the event mode for the digital inputs
MS<x>=00	Deactivate automatic transmission of the analogue inputs
MME=4	Activate the binary measuring mode
MM<x>=20	Set the filter mask

Example

ModNo = Connect()

In the case of return value *ModNo == 9*, EasyPort modules 1 and 4 have been detected.

Disconnect

void Disconnect()

Description

All interfaces opened with the connect method are once again closed, and communication with the EasyPorts is ended. The EasyPort modules, and thus the output signals, are reset with the *setup0* command.

Example

Disconnect()

Communication is ended.

GetComPort

short GetComPort(short *ModIndex*)

Return value

Number of the serial interface or 0.

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

Description

Reads out the number of the COM port to which the EasyPort with the transmitted number is connected. If a value of 0 is returned, the EasyPort with module number *ModIndex* is not available.

The COM port number is not required for use of EasyPort ActiveX control. It can be displayed, for example, for information purposes.

Example

ComNo = GetComPort(2)

If a value of *ComNo == 15* is returned, module 2 is connected to COM15.

GetModuleType

short GetModuleType(short *ModIndex*)

Return value

EasyPort module type:

• Return value 0:	No EasyPort with specified module number
• Return value 1:	Digital EasyPort D16
• Return value 2:	Analogue EasyPort DA8
• Return value 3:	EasyPort USB

Parameters

Module number: 0..4.
The standard module is addressed with module number 0.

Description

Reads out the type of EasyPort for the transmitted module number. If a value of 0 is returned, the EasyPort with module number *ModIndex* is not available.

Example

ModType = GetModuleType(2)

If a value of *ModType == 3* is returned, module 2 is an EasyPort USB.

SetModule

void SetModule(short *ModIndex*)

Parameters

ModIndex
Module number: 1..4

Description

EasyPort module *ModIndex* is made the standard module with the SetModule method. All following methods with module number 0 make reference to this module.

Example

SetModule(2)

InpWord = GetInputWord(0, 0)

InpWord1 = GetInputWord(1, 0)

The *InpWord* variable contains the value of input word 0 from module 2, which has been made the standard module with the *SetModule(2)* method. The *InpWord1* variable contains the value of input word 0 from module 1.

GetInput

short GetInput(short *ModIndex*, short *ByteIndex*, short *BitIndex*)

Return value

Input signal as 0 or 1.

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

ByteIndex

Byte number: 0..1

BitIndex
Bit number: 0..15

Description

Reads out the signal of a digital input. Bit numbers 0 through 15 are permissible for byte 0, and bit numbers 0 through 7 are permissible for byte 1.

Internally,*GetInput* uses the GetInputWord(short ModIndex, short WordIndex) method.

Example

Inp0_8 = GetInput(0, 0, 8)

Inp1_0 = GetInput(0, 1, 0)

Both methods refer to the same digital input, because bit 8 of the first byte corresponds to bit 0 of the second byte. *Inp0_8 and Inp1_0* thus contain the same value.

GetInputWord

long GetInputWord(short *ModIndex*, short *WordIndex*)

Return value

The value of an input channel as a word.

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

WordIndex
Word number: 0..4

Word number	Input channel	EasyPort input word
0	Digital input channel	EW<x>.0
1	Analogue input channel 0	EW<x>.2
2	Analogue input channel 1	EW<x>.4
3	Analogue input channel 2	EW<x>.6
4	Analogue input channel 3	EW<x>.8

Description

The current value of the digital, as well as the analogue input channels, can be queried with this method.

When the digital input signal is queried, the *GetInputWord* method accesses values saved to EasyPort ActiveX control. The event mode for digital inputs is activated by means of initialisation in Connect(). When EasyPort indicates that a value has changed at a digital input channel, the value is saved to ActiveX control.

In order to query the analogue input channels, ActiveX control must first get the value from the EasyPort with the *DEW<x>.<y>* command. When an analogue channel is activated for automatic transmission with the SetAutoSendMode(short ModIndex, short ChannelMask) method, channel querying can be accelerated with the *GetInputWord* method: The EasyPort module continuously transmits the current measured value to ActiveX control, which temporarily stores the value and makes it available via the *GetInputWord* method.

Example

DigInpWord = GetInputWord(1, 0)

AnaInpWord0 = GetInputWord(1, 1)

Variables *DigInpWord* and *AnaInpWord0* contain the values of the digital input channel and the first analogue input channel.

FESTO

GetOutputWord

long GetOutputWord(short *ModIndex*, short *WordIndex*)

Return value

The value of an output channel as a word.

Parameters

ModIndex

Module number: 0..4.
The standard module is addressed with module number 0.

WordIndex
Word number: 0..2

Description

The current value of the digital, as well as the analogue output channels, can be queried with this method. The word number is assigned to the output channels as follows:

Word number	Input channel	EasyPort input word
0	Digital output channel	AW<x>.0
1	Analogue output channel 0	AW<x>.2
2	Analogue output channel 1	AW<x>.4

The *GetOutputWord* method transmits EasyPort command *DAW<x>.<y>.*

Example

DigOutpWord = GetOutputWord(1, 0)

AnaOutpWord0 = GetOutputWord(1, 1)

Variables *DigOutpWord* and *AnaOutpWord0* contain the values of the digital output channel and the first analogue output channel.

SetAutoSendMode

void SetAutoSendMode(short *ModIndex*, **short** *ChannelMask***)**

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

ChannelMask
Contains the input channels which should be transmitted automatically by the EasyPort in the form of a binary pattern:

Bit number	Decimal value	Input channel
0	1	Analogue input channel 0
1	2	Analogue input channel 1
2	4	Analogue input channel 2
3	8	Analogue input channel 3

Several input channels can be activated by setting several bits in the *ChannelMask*.

Description

The values of the analogue input channel are read out automatically in a cyclical fashion from the EasyPort to ActiveX control after activation with the *SetAutoSendMode* method. In the event of a value change, ActiveX control triggers the InputWordChanged(short ModIndex, short WordIndex, long Value) event. In addition to this, the last transmitted value can be read out with the GetInputWord(short ModIndex, short WordIndex) method. When the *ChannelMask = 0* parameter is transferred, transmission is ended for all channels.

Example

SetAutoSendMode(1, 3)

AnaOutpWord0 = GetInputWord(1, 1)

AnaOutpWord1 = GetInputWord(1, 2)

AnaOutpWord2 = GetInputWord(1, 3)

SetAutoSendMode(1, 0)

First of all, automatic transmission is activated for analogue input channels 0 and 1 (*ChannelMask = 3*). The EasyPort then starts cyclically transmitting the values of the two channels. The *GetInputWord* method accesses the temporarily saved values for the first two invocations. Due to the fact that input channel 2 is not activated, the current value is first queried at the EasyPort with the *GetInputWord(1, 3)* method. *SetAutoSendMode(1, 0)* ends the transmission mode.

SetOutput

void SetOutput(short *ModIndex*, **short** *ByteIndex*, **short** *BitIndex*, **short** *Value***)**

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

ByteIndex

Byte number: 0..1

BitIndex
Bit number: 0..15

Value
Value: 0 or 1

Description

Sets the signal of a digital output. Bit numbers 0 through 15 are permissible for byte 0, and bit numbers 0 through 7 are permissible for byte 1.

Internally, *GetOutput* uses the method: SetOutputWord(short ModIndex, short WordIndex, long Value).

Example

SetOutput(0, 0, 8, 1)

SetOutput(0, 1, 0, 0)

Both methods refer to the same digital output, because bit 8 of the first byte corresponds to bit 0 of the second byte. The first invocation sets a 1-signal, and the second invocation sets a 0-signal at the respective output.

SetOutputWord

void SetOutputWord(short *ModIndex*, **short** *WordIndex*, **long** *Value***)**

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

WordIndex

Word number: 0..2

Word number	Input channel	EasyPort input word
0	Digital output channel	AW<x>.0
1	Analogue output channel 0	AW<x>.2
2	Analogue output channel 1	AW<x>.4

Value
Value: 0..32767 (0x7FFF)

Description

The value of the digital, as well as the analogue output channels, can be set with this method. The value is only transmitted to the EasyPort with the *MAW<x>.<y>=<zzzz>* command if the *Value* differs from the last set value.

Example

SetOutputWord(1, 0, 255)

SetOutputWord(1, 0, 255)

SetOutputWord(1, 1, 32767)

The first command sets the first 8 bits of the digital outputs to the 1-signal. The second invocation of *SetOutputWord(1, 0, 255)* has no effect, because the output value is already set to 255. The third invocation of the method results in maximum voltage at output 0.

ForceDisplay

void ForceDisplay(short *ModIndex*, short *ChannelIndex*)

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

ChannelIndex
Channel number: 1, 2, 4, 8, 16, 32 or 0

Channel number	Input channel
1	Analogue input channel 0
2	Analogue input channel 1
4	Analogue input channel 2
8	Analogue input channel 3
16	Analogue output channel 0
32	Analogue output channel 1
0	Deactivate force

Description

Which analogue input or output channel will appear at the EasyPort display is determined with the *ForceDisplay* method. The display can no longer be switched using the keys at the EasyPort. Keypad disabling can be cancelled with the *ChannelIndex = 0* method. EasyPort command*MF<x>=<yy>* is transmitted.

Example

ForceDisplay(0, 32)

ForceDisplay(0, 0)

The first command switches the display at the EasyPort to analogue output channel 1. The second command cancels keypad disabling without influencing the display.

SetDisplayUnit

void SetDisplayUnit(short *ModIndex*, **short** *ChannelIndex*, **short** *UnitIndex***)**

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

ChannelIndex
Channel number: 1, 2, 4, 8, 16 or 32

Channel number	Input channel
1	Analogue input channel 0
2	Analogue input channel 1
4	Analogue input channel 2
8	Analogue input channel 3
16	Analogue output channel 0
32	Analogue output channel 1

UnitIndex
Unit number: 0, 1, 2, 3, 4 or 5

Unit number	Unit of measure
0	V
1	bar
2	PSI
3	MPa
4	l/min
5	ºC

Description

Sets the unit of measure for the display of the respective analogue input/output channel. Toggling is still possible with the help of the keypad at the EasyPort. EasyPort command*MU<x>.<yy>=<zz>* is transmitted.

Example

SetDisplayUnit(0, 16, 4)

ForceDisplay(0, 16)

The unit of measure for analogue output channel 0 is switched to l/min, and the display is switched to output channel 0 with the *ForceDisplay* method.

SetGain

void SetGain(short *ModIndex*, **short** *UnitIndex*, **float** *GainFactor***)**

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

UnitIndex
Unit number: 0, 1, 2, 3, 4 or 5

Unit number	Unit of measure	Preset
0	V	10.0
1	bar	1.0
2	PSI	1.0
3	MPa	1.0
4	l/min	1.0
5	ºC	1.0

GainFactor
Gain factor as floating-point number

Description

The display at the EasyPort can be appropriately scaled to match the utilised sensor with the help of the gain factor. If the preset values are used, 1 V corresponds to a display value of 0.1 bar, PSI, MPa, l/min orºC. Conversion can be adjusted to the sensor's characteristic curve by changing the gain factor.

EasyPort command *MG<x>.<yy>=<zz>* is transmitted.

Example

SetGain(0, 1, 20.0)

SetGain(0, 2, 290.01)

SetGain(0, 3, 2.0)

If a gain factor of 20.0 is selected for the unit of measure bar, an applied voltage of 1 V results in a display value of 2 bar.
A factor of 29.01 is set for unit of measure PSI, and a factor of 2.0 for MPa, resulting in the following display values at the EasyPort: 1 V => 2 bar => 29.01 PSI => 0.2 MPa.

FESTO

SetMeasuringRange

void SetMeasuringRange(short *ModIndex*, **short** *InOut*, **short** *RangeIndex*)

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

InOut
Selection as to whether the input or the output channels will be set: 0 or 1

Channel selection	Channel group
0	Input channels
1	Output channels

RangeIndex

Measuring range: 0 or 1

Range selection	Measuring range
0	0..10V
1	+/-10V

Description

Sets the measuring range for the analogue inputs or outputs.
For analogue-digital conversion, voltage at the input/output channels is always displayed within a value range of 0..32767. In accordance with the selected measuring range, the GetInputWord(short ModIndex, short WordIndex) method reads out different digital values for the same applied voltage. In the same way, different voltages are generated for the same digital value by means of the SetOutputWord(short ModIndex, short WordIndex, long Value) method. After switching the EasyPort on, the measuring range is 0..10V.

EasyPort command *MRE<x>=<y>* is transmitted in order to set the input channels, and the *MRA<x>=<y>* command is transmitted for the output channels.

Example

AnaInput = GetInputWord(0, 1)

SetOutputWord(0, 1, 3277)

SetMeasuringRange(0, 0, 1)

SetMeasuringRange(0, 1, 1)

AnaInput = GetInputWord(0, 1)

The first invocation of *GetInputWord* reads out a value of 3277 (32767 / 10 V) for 1 V at input channel 0. *SetOutputWord*, on the other hand, generates 1 V at output channel 0. After changing the measuring range to +/-10V, the*GetInputWord* method reads out a value of 18022 (32767 / 20V * 11) and a voltage of -8 V is read out at output channel 0, because the digital value of 3277 is unchanged.

GetCounter

long GetCounter(short *ModIndex*, short *CounterIndex*)

Return value

Current counter value.

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

CounterIndex
Counter index: 0..1

Description

Counters 0 and 1 can be queried with this method. The counter inputs are connected to bit 0 of the two digital input bytes. The counter must first be activated with the StartCounter(short ModIndex, short CounterIndex, short Activate) method.

EasyPort command*DC<x>.<y>* is transmitted.

Example

StartCounter(0, 0, 1)

...

PosCounter = GetCounter(0, 0)

After counter 0 has been activated, its current value is saved to *PosCounter*.

StartCounter

void StartCounter(short *ModIndex*, short *CounterIndex*, short *Activate*)

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

CounterIndex
Counter index: 0..1

Activate
Counter reading: 0..1

Description

Counters 0 and 1 can be started and stopped with this method. The counter is started with *Activate = 0*, and the counter is stopped and reset with *Activate = 1*. Uses the EasyPort MC command

EasyPort command *MC<x>.<y> =<z>* is transmitted.

Example

StartCounter(0, 0, 1)

...

PosCounter = GetCounter(0, 0)

StartCounter(0, 0, 0)

After counter 0 has been activated, its current value is saved to *PosCounter*. The counter is then stopped again.

SendAndGetString

BSTR SendAndGetString(short *ModIndex*, LPCTSTR *str*)

Return value

Response string from the EasyPort module.

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

str
An EasyPort command.

Description

Any desired command can be transmitted to an EasyPort module. The command is ended automatically with a carriage return.

After transmission, the method waits until the EasyPort module transmits a response, and sends it back.

Example

ResStr = SendAndGetString(0, "DS")

The DS "get status" command has been introduced along with the EasyPort USB. The response in *ResStr* appears in the following format for the EasyPort USB: *"S=<xx>"*. Older types of EasyPorts return the *"DS"* command unchanged in order to indicate that it couldn't be interpreted.

FESTO

SetFilterMask

void SetFilterMask(short *ModIndex*, **short** *FilterMask***)**

Parameters

ModIndex
Module number: 0..4.
The standard module is addressed with module number 0.

FilterMask
Filter mask: 0..255

Description

Individual bits of the digital value can be suppressed for analogue-digital conversion. A more stable signal can thus be achieved at the expense of accuracy.

EasyPort command*MM<x> =<z>* is transmitted.

Example

SetFilterMask(0, 7F)

After setting the filter mask to 127 (binary 01111111), the last 7 bits of the digital value of an analogue signal are set to 0, so that changes within this range are not digitised. As a result, digitised analogue input values make larger jumps.

FESTO

AboutBox

void AboutBox()

Description

The information dialog with the version number.

FESTO

ShowDebugWnd

void ShowDebugWnd()

Description

Opens a window within which communication between ActiveX control and the connected EasyPorts is displayed.

FESTO

WriteDebugString

void WriteDebugString(LPCTSTR *Message*)

Parameters

Message
Single-line text.

Description

Reads out a line in the debug window. The window must first be opened with the ShowDebugWnd method.

Outdated methods

EasyPort ActiveX control was entirely revised for the introduction of EasyPort USB.
Main differences include:

- Simultaneous support of up to 4 EasyPort modules
- Automatic search at serial ports
- Support for new EasyPort commands

This has resulted in a collection of new methods which have replaced technically outdated methods. In order to assure compatibility with older applications, these methods have been retained in ActiveX control. However, they shouldn't be used any more.

The following table shows how the outdated methods can be replaced:

Outdated method	**Alternative method**
short CloseSerial()	void Disconnect()
short GetAnalogInput0() short GetAnalogInput1() short GetAnalogInput2() short GetAnalogInput3()	long GetInputWord(short ModIndex, short WordIndex)
short GetDigitalInput0() short GetDigitalInput1()	long GetInputWord(short ModIndex, short WordIndex)
short OpenSerial(short PortNo)	short Connect()
void SendString(short ModIndex, LPCTSTR str)	BSTR SendAndGetString(short ModIndex, LPCTSTR str)
void SetAnalogOutput0(short Data) void SetAnalogOutput1(short Data)	void SetOutputWord(short ModIndex, short WordIndex, long Value)
void SetChannelMask(short Mask)	void SetAutoSendMode(short ModIndex, short ChannelMask)
void SetDigitalOutput0(short Data) void SetDigitalOutput1(short Data)	void SetOutputWord(short ModIndex, short WordIndex, long Value)
void StartTimer()	The timer is started automatically by means of the Connect() method.
void StopTimer()	Stopping the timer is no longer necessary.
void DoBackgroundProcessing(short Count)	Runs through the Windows message loop Count times
void DoProcess()	Waits 2 ms. The Windows message loop is run through during waiting time.