EasyPort ActiveX control

License agreement / exclusion of liability

No guarantee whatsoever is made for use of EasyPort ActiveX control, in particular for use in industrial applications.

Internet: www.festo.com/didactic
e-mail: did@festo.com

Introduction

EasyPort ActiveX control provides an interfaces for programming Festo's EasyPort modules. It can be added to software packages which support this interface as a COM object.

Before using EasyPort ActiveX control, you should thoroughly familiarise yourself with use and programming of ActiveX control.

EasyPort ActiveX control offers easy to use methods and events for communication with one or more EasyPort modules.

These methods and events are described in the following pages, and the basic sequence for communicating with an EasyPort is depicted.

Preparation for programming

EasyPort ActiveX control can only be utilised after it has been successfully installed to the programming PC. After installation, EasyPort ActiveX control is registered at the PC and can be used in software projects. If an application created with EasyPort ActiveX control will be used on another PC, ActiveX control must be installed to it as well.

During the first step, ActiveX control is added to the project along with the programming environment. This option is provided for in most programming environments such as Visual C++, Excel and LabView. Please refer to the documentation for the programming environment to find out how control can be added to your project.

Even for different programming environments, the principle remains the same:
EasyPort control must be selected from a list of all ActiveX controls available at the PC.

This is usually done by selecting the corresponding name:

Festo EasyPort ActiveX Control

In some cases it may be necessary to address ActiveX control via its so-called class identifier:

CLSID = {2A89B520-8AC3-11D3-9143-400051C71400}

Typical EasyPort communication sequence

Essentially, communication with EasyPort modules comprises three phases.

1 Log on to EasyPort

The log-on phase is started with the Connect() method. *Connect* searches all serial ports within the system for EasyPort modules. In addition to physical serial ports, serial ports which are simulated by the EasyPort USB driver are also taken into consideration. The return value from the *Connect* method specifies the detected module numbers. Up to 4 EasyPort modules can be connected at once.

If EasyPort modules have been detected, the EasyPort type can be identified using the GetModuleType (short ModIndex) method. The following types are supported: digital with 16 inputs/outputs (D16), digital with additional analogue inputs/outputs (D8A), USB digital and analogue inputs/outputs (USB), digital with 6 inputs/outputs (D6).

EasyPort ActiveX control methods expect the module number as the first parameter. Alternatively, one EasyPort module can be made the standard module with the SetModule(short ModIndex) method. Module number 0 can then be transmitted with subsequent method invocations, and the standard module is used as a result.

2 Data exchange

Data exchange consists of reading and writing digital and analogue input signals with GetInputWord(short ModIndex, short WordIndex) and SetOutputWord(short ModIndex, short WordIndex, long Value). Individual digital bits are read and written with GetInput(short ModIndex, short ByteIndex, short BitIndex) and SetOutput(short ModIndex, short ByteIndex, short BitIndex, short Value).

EasyPort modules can autonomously transmit the analogue input values to the PC for quick acquisition of analogue measured values. One or several input channels can be additionally activated with the SetAutoSendMode(short ModIndex, short ChannelMask) method. ActiveX control saves current values, and in this case forwards them to the application without querying EasyPort again when the *GetInputWord* method is invoked.

When the *GetInputWord* method is used, the input values must be continuously queried by the application. An application can respond more conveniently to changes at EasyPort's inputs by implementing a handler for the ActiveX event:InputWordChanged(short ModIndex, short WordIndex, long Value). If changes occur at the digital or analogue inputs, assuming they have been activated for autonomous transmission with the *SetAutoSendMode* method, EasyPort ActiveX control triggers the *InputWordChanged* event.

3 Log off from EasyPort

The Disconnect() method is used to break off connection to the EasyPort modules. The modules are set to their basic state as a result, and the output signals are set to 0.

In addition to the mentioned basic functions, methods for the following are included with EasyPort ActiveX control:

- Set the measuring range
  SetMeasuringRange(short ModIndex, short InOut, short RangeIndex)
- Set the display,
  display channel selection: ForceDisplay(short ModIndex, short ChannelIndex)
  Unit of measure display: SetDisplayUnit(short ModIndex, short ChannelIndex, short UnitIndex)

Display value scaling: SetGain(short ModIndex, short ChannelIndex, float GainFactor)

and

- High-speed counter inputs
  Start and stop a counter: StartCounter(short ModIndex, short CounterIndex, short Activate)
  Read out a counter value: GetCounter(short ModIndex, short CounterIndex)