BECKHOFF Automation: Vorwort

Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiter entwickelt.
Deshalb ist die Dokumentation nicht in jedem Fall vollständig auf die Übereinstimmung mit den beschriebenen Leistungsdaten, Normen oder sonstigen Merkmalen geprüft.
Falls sie technische oder redaktionelle Fehler enthält, behalten wir uns das Recht vor, Änderungen jederzeit und ohne Ankündigung vorzunehmen.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE® und XFC® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die TwinCAT Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP0851348, US6167425 mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

Copyright

TwinCAT MODBUS TCP Server

Übersicht

Der MODBUS ADS-Server ermöglicht es über eine MODBUS/TCP-Verbindung mit einem MODBUS-Client bzw. MODBUS-Server zu kommunizieren. Somit kann der Server sowohl als MODBUS-Client als auch als MODBUS-Server verwendet werden:

- MODBUS Client-Funktionalität: Mit Hilfe von SPS-Bausteinen kann per ADS mit einem MODBUS-Gerät kommuniziert werden, um den Inhalt von Registern bzw. digitalen Ein/Ausgänge auszulesen oder zu beschreiben.
- MODBUS Server-Funktionalität: Der Server kann Modbus-Funktionen über TCP/IP empfangen. Die Modbus-Register und Modbus-Ein/Ausgängen werden dann auf Bereiche der PLC gemappt.

TwinCAT MODBUS TCP Server : Modbus Client-Funktionalität

Übersicht

Der Modbus Ads-Server ermöglicht es per ADS mit einem Modbus-Gerät (Modbus-Server) über eine MODBUS/TCP-Verbindung zu kommunizieren. Die im Modbus-Protokoll definierten Funktionen sind als SPS-Bausteine in der Bibliothek TcModbusSrv.lib realisiert, die intern per ADS mit dem Server kommunizieren. Mit Hilfe der Modbus-Funktionen können dann die Register bzw. digitale Ein/Ausgänge eines Modbus-Gerätes ausgelesen bzw. beschrieben werden.

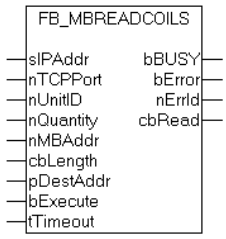
Modbus Funktion	Funktions-Code	SPS-Baustein
Read Coils	1	FB_MBReadCoils
Read Discreete Inputs	2	FB_MBReadInputs
Read Registers	3	FB_MBReadRegs
Read Input Registers	4	FB_MBReadInputRegs
Write Single Coil	5	FB_MBWriteSingleCoil
Write Single Register	6	FB_MBWriteSingleReg
Write Multiple Coils	15	FB_MBWriteCoils

Write Multiple Registers	16	FB_MBWriteRegs
Read/Write Multiple Registers	23	FB_MBReadWriteRegs
Diagnose	8	FB_MBDiagnose

Hinweis: Es ist derzeit nur möglich, auf das Laufzeitsystem 1 zuzugreifen.

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBReadCoils (Modbus-Funktion 1)



Diese Funktion wird zum Lesen von 1 bis 2000 digitalen Ausgängen (Coils) benutzt. Ein digitaler Ausgang entspricht einem Bit der gelesenen Bytes.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nQuantity       : WORD;
        nMBAddr         : WORD;
        cbLength        : UDINT;
        pDestAddr       : UDINT;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nQuantity: Anzahl der zu lesenden digitalen Eingänge (Bits).

nMBAdd: Startadresse der zu lesenden digitalen Eingänge.

cbLength: Enthält die Größe des Puffers, in den die Daten gelesen werden sollen.

pDestAdd: Enthält die Adresse des Puffers, in den die Daten gelesen werden sollen. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann. Der Puffer muss mindestens die Größe nQuantity/8 bzw. nQuantity/8 + 1, wenn der Rest ungleich 0 ist, besitzen.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
        cbRead          : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

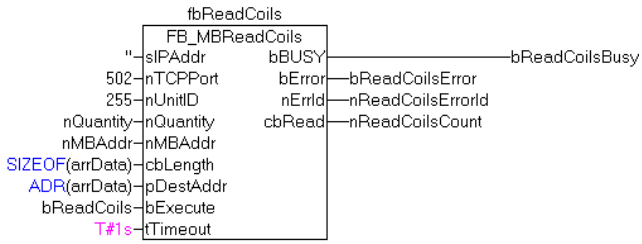
cbRead: Enthält die Anzahl der aktuell gelesenen Bytes.

Function specific ADS error code	Possible reason
0x8001	Nicht implementierte Modbus-Funktion

0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbReadCoils     : FB_MBReadCoils;
        bReadCoils      : BOOL;
        bReadCoilsBusy  : BOOL;
        bReadCoilsError : BOOL;
        nReadCoilsErrorId       : UDINT;
        nReadCoilsCount : UDINT;
        nQuantity       : WORD := 10;
        nMBAddr         : WORD := 5;
        arrData         : ARRAY [1..2] OF BYTE;
END_VAR
```



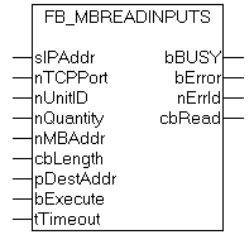
Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des ReadCoils-Befehls, wird der Inhalt der digitalen Ausgänge 6 - 15 in das Array arrData geschrieben:

Digitale Ausgänge	**Array-Offset**	**Status**
6-13	1	0x54 Status des Ausgangs 13 ist das MSB dieses Bytes (ganz links) Status des Ausgangs 6 ist das LSB dieses Bytes (ganz rechts)
14-15	2	0x02 da nur 10 Ausgänge gelesen werden sollen, werden die restlichen Bits (3-8) auf 0 gesetzt.

Entwicklungsumgebung	**Zielplattform**	**Einzubindende SPS Bibliotheken**
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBReadInputs (Modbus-Funktion 2)



Diese Funktion wird zum Lesen von 1 bis 2000 digitalen Eingängen benutzt.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nQuantity       : WORD;
        nMBAddr         : WORD;
        cbLength        : UDINT;
        pDestAddr       : UDINT;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über Tcp/Ip angesprochen wird, muss dieser Wert 16#FF entsprechen.

nQuantity: Anzahl der zu lesenden digitalen Eingänge (Bits).

nMBAdd: Startadresse der zu lesenden digitalen Eingänge.

pDestAddr: Enthält die Adresse des Puffers, in den die Daten gelesen werden sollen. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann. Der Puffer muss mindestens die Größe nQuantity/8 bzw. nQuantity/8 + 1, wenn der Rest ungleich 0 ist, besitzen.

cbLength: Enthält die Anzahl der zu lesenden Bytes.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
        cbRead          : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

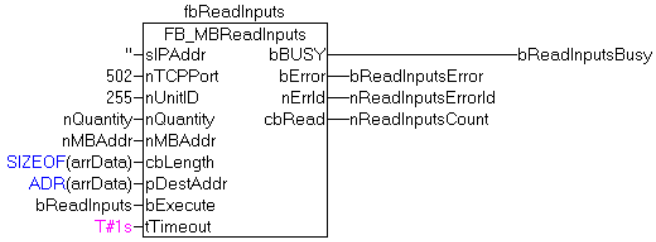
cbRead: Enthält die Anzahl der aktuell gelesenen Bytes.

Function specific ADS error code	**Possible reason**
0x8001	Nicht implementierte Modbus-Funktion
0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbReadInputs    : FB_MBReadInputs;
        bReadInputs     : BOOL;
```

```
        bReadInputsBusy : BOOL;
        bReadInputsError        : BOOL;
        nReadInputsErrorId      : UDINT;
        nReadInputsCount        : UDINT;
        nQuantity       : WORD := 20;
        nMBAddr         : WORD := 29;
        arrData         : ARRAY [1..3] OF BYTE;
END_VAR
```



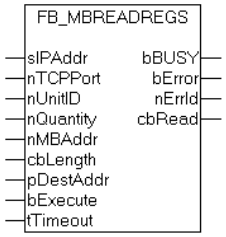
Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des ReadInputs-Befehls, wird der Inhalt der digitalen Ausgänge 30 - 49 in das Array arrData geschrieben:

Digitale Ausgänge	**Array-Offset**	**Status**
		0x34
29-36	1	Status des Ausgangs 36 ist das MSB dieses Bytes (ganz links) Status des Ausgangs 29 ist das LSB dieses Bytes (ganz rechts)
		0x56
37-44	2	Status des Ausgangs 44 ist das MSB dieses Bytes (ganz links) Status des Ausgangs 37 ist das LSB dieses Bytes (ganz rechts)
		0x07
45-49	3	da nur 20 Ausgänge gelesen werden sollen, werden die restlichen Bits (5-8) auf 0 gesetzt.

Entwicklungsumgebung	**Zielplattform**	**Einzubindende SPS Bibliotheken**
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBReadRegs (Modbus-Funktion 3)



Diese Funktion wird zum Lesen von 1 bis 125 Ausgangs-Register(16 Bit) benutzt. Das erste Byte enthält die unteren Bits und das zweite Byte die oberen acht Bits.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nQuantity       : WORD;
        nMBAddr         : WORD;
        cbLength        : UDINT;
        pDestAddr       : UDINT;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nQuantity: Anzahl der zu lesenden Ausgangs-Register (WORD).

nMBAdd: Startadresse der zu lesenden Ausgangs-Register.

pDestAddr: Enthält die Adresse des Puffers, in den die Daten gelesen werden sollen. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann. Der Puffer muss mindestens die Größe nQuantity * 2 besitzen.

cbLength: Enthält die Anzahl der zu lesenden Bytes.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
        cbRead          : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

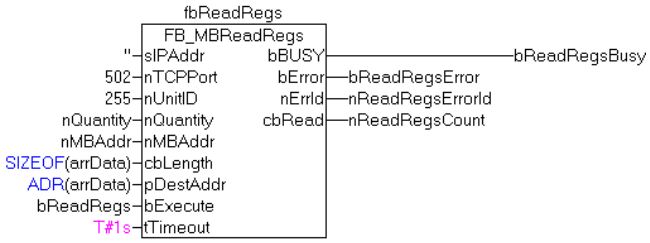
nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

cbRead: Enthält die Anzahl der aktuell gelesenen Bytes.

Function specific ADS error code	**Possible reason**
0x8001	Nicht implementierte Modbus-Funktion
0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbReadRegs      : FB_MBReadRegs;
        bReadRegs       : BOOL;
        bReadRegsBusy   : BOOL;
        bReadRegsError  : BOOL;
        nReadRegsErrorId        : UDINT;
        nReadRegsCount  : UDINT;
        nQuantity       : WORD:=2;
        nMBAddr         : WORD:=24;
        arrData         : ARRAY [1..2] OF WORD;
END_VAR
```



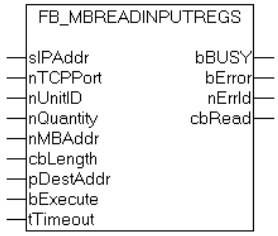
Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des ReadRegs-Befehls, befinden sich der Inhalt der Register 25 und 26 in dem Array arrData:

Register	**Array-Offset**	**Status**
25	1	0x1234 (als Byte 0x34 0x12)
26	2	0x5563 (als Byte 0x63 0x55)

Entwicklungsumgebung	**Zielplattform**	**Einzubindende SPS Bibliotheken**
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBReadInputRegs (Modbus-Funktion 4)



Diese Funktion wird zum Lesen von 1 bis 125 Eingangs-Register(16Bit) benutzt. Endian

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nQuantity       : WORD;
        nMBAddr         : WORD;
        cbLength        : UDINT;
        pDestAddr       : UDINT;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nQuantity: Anzahl der zu lesenden Eingangs-Register (WORD).

nMBAdd: Startadresse der zu lesenden Eingangs-Register.

pDestAddr: Enthält die Adresse des Puffers, in den die Daten gelesen werden sollen. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann. Der Puffer muss mindestens die Größe nQuantity * 2 besitzen.

cbLength: Enthält die Anzahl der zu lesenden Bytes.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
```

```
        bError          : BOOL;
        nErrId          : UDINT;
        cbRead          : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

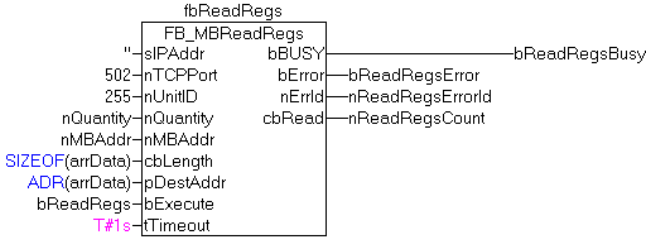
nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

cbRead: Enthält die Anzahl der aktuell gelesenen Bytes.

Function specific ADS error code	Possible reason
0x8001	Nicht implementierte Modbus-Funktion
0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbReadRegs      : FB_MBReadRegs;
        bReadRegs       : BOOL;
        bReadRegsBusy   : BOOL;
        bReadRegsError  : BOOL;
        nReadRegsErrorId        : UDINT;
        nReadRegsCount  : UDINT;
        nQuantity       : WORD := 3;
        nMBAddr         : WORD:= 2;
        arrData         : ARRAY [1..3] OF WORD;
END_VAR
```



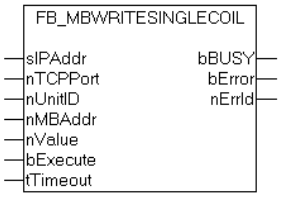
Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des ReadRegs-Befehls, befinden sich der Inhalt der Register 3-5 in dem Array arrData:

Register	Array-Offset	Status
3	1	0x4543 (als Byte 0x43 0x45)
4	2	0x5234 (als Byte 0x34 0x52)
5	2	0x1235 (als Byte 0x35 0x12)

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBWriteSingleCoil (Modbus-Funktion 5)



Diese Funktion wird zum Beschreiben eines digitalen Ausgangs(Coil) benutzt. Dabei handelt es sich um einen Bit-Zugriff.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nMBAddr         : WORD;
        nValue          : WORD;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nMBAddr: Adresse des digitalen Ausgangs.

nValue: Wert, der in den digitalen Ausgang geschrieben werden soll. Der Wert 16#FF00 schalten den Ausgang ein und der Wert 16#0000 schaltet den Ausgang ab.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

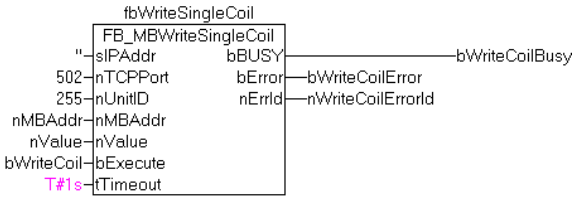
nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

Function specific ADS error code	Possible reason
0x8001	Nicht implementierte Modbus-Funktion
0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbWriteSingleCoil       : FB_MBWriteSingleCoil;
        bWriteCoil      : BOOL;
        bWriteCoilBusy  : BOOL;
        bWriteCoilError : BOOL;
        nWriteCoilErrorId       : UDINT;
        nMBAddr         : WORD := 3;
        nValue          : WORD := 16#FF00;
```

```
END_VAR
```

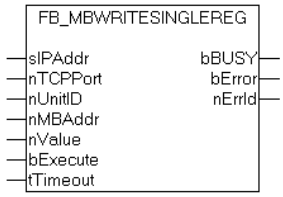


Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des WriteSingleCoil-Befehls, wird der digitale Ausgang 4 angeschaltet.

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBWriteSingleReg (Modbus-Funktion 6)



Diese Funktion wird zum Beschreiben eines einzelnen Registers benutzt. Dabei handelt es sich um einen 16Bit-Zugriff.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nMBAddr         : WORD;
        nValue          : WORD;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nMBAddr: Adresse des Ausgangs-Registers.

nValue: Wert, der in das Register geschrieben werden soll.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

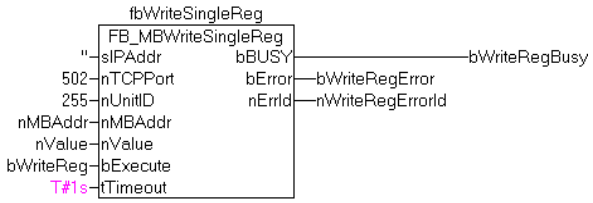
nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

Function specific ADS error code	Possible reason
0x8001	Nicht implementierte Modbus-Funktion

0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbWriteSingleReg        : FB_MBWriteSingleReg;
        bWriteReg       : BOOL;
        bWriteRegBusy   : BOOL;
        bWriteRegError  : BOOL;
        nWriteRegErrorId: UDINT;
        nMBAddr         : WORD := 4;
        nValue          : WORD := 16#1234;
END_VAR
```

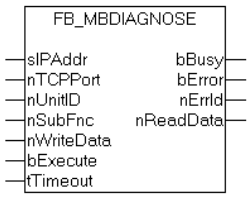


Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des WriteSingleReg-Befehls, wird in das Register 5 der Wert 16#1234 geschrieben.

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBDiagnose (Modbus-Funktion 8)



Die Diagnose-Funktion stellt eine Reihe von Tests, für die Überprüfung des Übertragungssystems zwischen dem Master und dem Slave oder für die Überprüfung der verschiedenen internen Fehlerzustände innerhalb des Slaves, zur Verfügung.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nSubFnc         : WORD;
        nWriteData      : WORD;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nSubFnc: Die Subfunktion, die ausgeführt werden soll.

nWriteData: Das Datenwort, das geschrieben werden soll.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBusy           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
        nReadData       : WORD;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

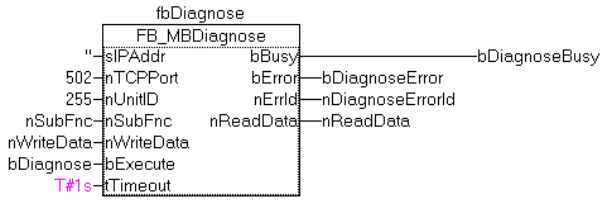
nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

nReadData: Liefert das gelesene Datenwort.

Function specific ADS error code	Possible reason
0x8001	Nicht implementierte Modbus-Funktion
0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbDiagnose      : FB_MBDiagnose;
        bDiagnose       : BOOL;
        bDiagnoseBusy   : BOOL;
        bDiagnoseError  : BOOL;
        nDiagnoseErrorId        : UDINT;
        nSubFnc         : WORD;
        nReadData               : WORD;
        nWriteData      : WORD;
END_VAR
```

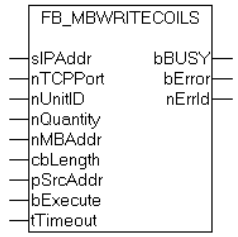


Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des Dianose-Befehls, befinden sich in nReadData das gelesene Datenwort.

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBWriteCoils (Modbus-Funktion 15)



Diese Funktion wird zum Beschreiben mehrerer digitaler Ausgänge benutzt.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nQuantity       : WORD;
        nMBAddr         : WORD;
        cbLength        : UDINT;
        pSrcAddr                : UDINT;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nQuantity: Anzahl der digitalen Ausgänge, die beschrieben werden sollen (Bits).

nMBAddr: Startadresse der digitalen Ausgänge , die beschrieben werden sollen.

cbLength: Enthält die Größe des Puffer, der die zu schreibenden Daten enthält .

pDestAddr: Enthält die Adresse des Puffers, der die zu schreibenden Daten enthält. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann. Der Puffer muss mindestens die Größe nQuantity/8 bzw. nQuantity/8 + 1, wenn der Rest ungleich 0 ist, besitzen.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
        cbRead          : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

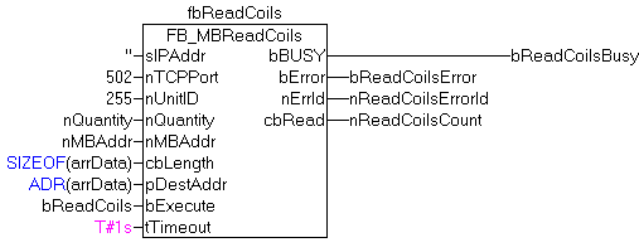
bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

Function specific ADS error code	**Possible reason**
0x8001	Nicht implementierte Modbus-Funktion
0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbWriteCoils    : FB_MBWriteCoils;
        bWriteCoils     : BOOL;
        bWriteCoilsBusy : BOOL;
        bWriteCoilsError        : BOOL;
        nWriteCoilsErrorId      : UDINT;
        nWriteCoilsCount        : UDINT;
        nQuantity       : WORD := 10;
        nMBAddr         : WORD := 14;
        arrData         : ARRAY [1..2] OF BYTE := 16#75,16#03;
END_VAR
```



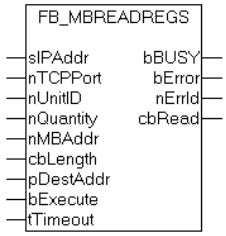
Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des ReadCoils-Befehls, wird der Inhalt des Arrays arrData in die Ausgänge 15-24 geschrieben:

Bit	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
Output	22	21	20	19	18	17	16	15	X	X	X	X	X	X	24	23

Entwicklungsumgebung	**Zielplattform**	**Einzubindende SPS Bibliotheken**
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBWriteRegs (Modbus-Funktion 16)



Diese Funktion wird zum Beschreiben mehrerer Ausgangs-Register(16 Bit) benutzt.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nQuantity       : WORD;
        nMBAddr         : WORD;
        cbLength        : UDINT;
        pSrcAddr                : UDINT;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nQuantity: Anzahl der zu schreibenden Ausgangs-Register (WORD).

nMBAdd: Startadresse der zu schreibenden Ausgangs-Register.

pSrcAddr: Enthält die Adresse des Puffers, der die tu schreibenden Daten enthält. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann. Der Puffer muss mindestens die Größe nQuantity * 2 besitzen.

cbLength: Enthält die Anzahl der zu schreibenden Bytes.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
```

```
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

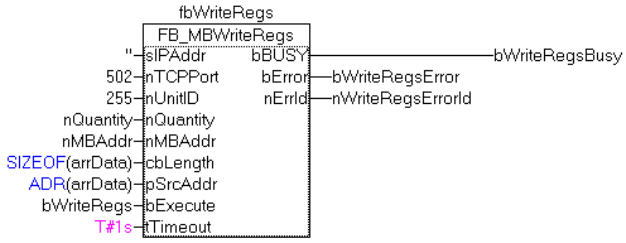
bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

Function specific ADS error code	Possible reason
0x8001	Nicht implementierte Modbus-Funktion
0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbWriteRegs     : FB_MBWriteRegs;
        bWriteRegs      : BOOL;
        bWriteRegsBusy  : BOOL;
        bWriteRegsError : BOOL;
        nWriteRegsErrorId       : UDINT;
        nWriteRegsCount : UDINT;
        nQuantity       : WORD := 3;
        nMBAddr         : WORD := 4;
        arrData         : ARRAY [1..3] OF WORD;
END_VAR
```

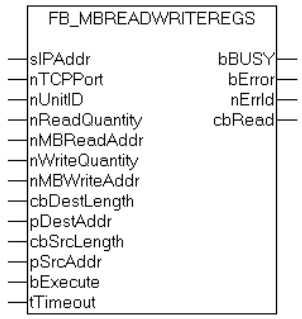


Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des ReadRegs-Befehls, wird der Inhalt des Arrays arrData in die Register 5-7 geschrieben.

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT PLC Library: TcModbusSrv

FUNCTION_BLOCK FB_MBReadWriteRegs (Modbus-Funktion 23)



Diese Funktion liest zuerst mehrere Register und beschreibt danach mehrer Register.

VAR_INPUT

```
VAR_INPUT
        sIPAddr         : STRING(15);
        nTCPPort                : UINT:= MODBUS_TCP_PORT;
        nUnitID         : BYTE:=16#FF;
        nReadQuantity   : WORD;
        nMBReadAddr     : WORD;
        nWriteQuantity  : WORD;
        nMBWriteAddr    : WORD;
        cbDestLength    : UDINT;
        pDestAddr       : UDINT;
        cbSrcLength     : UDINT;
        pSrcAddr                : UDINT;
        bExecute                : BOOL;
        tTimeout                : TIME;
END_VAR
```

sIPAddr: Ist ein String, der die IP-Adresse des Zielgerätes enthält.

nTCPPort: Portnummer des Zielgerätes.

nUnitID: Identifizierungsnummer eines Gerätes eines seriellen Sub-Netzwerkes. Wenn ein Gerät direkt über TCP/IP angesprochen wird, muss dieser Wert 16#FF entsprechen.

nRdQuantity: Anzahl der zu lesenden Ausgangs-Register (WORD).

nRdMBAdd: Startadresse der zu lesenden Ausgangs-Register.

nWrQuantity: Anzahl der zu schreibenden Ausgangs-Register (WORD).

nWrMBAdd: Startadresse der zu schreibenden Ausgangs-Register.

pDestAddr: Enthält die Adresse des Puffers, in den die Daten gelesen werden sollen. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann. Der Puffer muss mindestens die Größe nQuantity * 2 besitzen.

cbDestLength: Enthält die Anzahl der zu lesenden Bytes.

pSrcAddr: Enthält die Adresse des Puffers, der die zu schreibenden Daten enthält. Der Puffer kann eine Einzelvariable, ein Array oder eine Struktur sein, dessen Adresse man mit dem ADR - Operator ermitteln kann. Der Puffer muss mindestens die Größe nQuantity * 2 besitzen.

cbSrcLength: Enthält die Anzahl der zu schreibenden Bytes.

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Funktionsbaustein aktiviert.

tTimeout: Gibt die Timeout-Zeit an, die bei der Ausführung des ADS-Kommandos nicht überschritten werden darf.

VAR_OUTPUT

```
VAR_OUTPUT
        bBUSY           : BOOL;
        bError          : BOOL;
        nErrId          : UDINT;
        cbRead          : UDINT;
END_VAR
```

bBusy: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt und bleibt gesetzt, bis eine Rückmeldung erfolgt.

bError: Sollte ein ADS-Fehler bei der Übertragung des Kommandos erfolgen, dann wird dieser Ausgang gesetzt, nachdem der bBusy-Ausgang zurückgesetzt wurde.

nErrId: Liefert bei einem gesetzten bError-Ausgang die ADS-Fehlernummer.

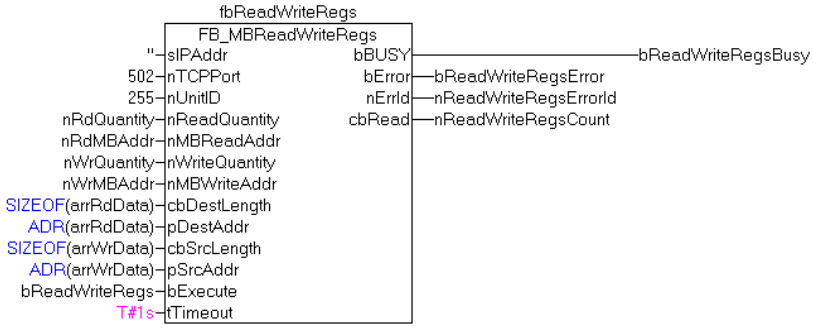
cbRead: Enthält die Anzahl der aktuell gelesenen Bytes.

Function specific ADS error code	Possible reason
0x8001	Nicht implementierte Modbus-Funktion
0x8002	Ungültige Adresse oder Länge
0x8003	Ungültige Parameter: - falsche Registeranzahl
0x8004	Fehler im Modbus-Server

Beispiel für den Aufruf des Bausteins in FBD:

```
PROGRAM Test
VAR
        fbReadWriteRegs         : FB_MBReadWriteRegs;
        bReadWriteRegs          : BOOL;
        bReadWriteRegsBusy      : BOOL;
        bReadWriteRegsError     : BOOL;
        nReadWriteRegsErrorId   : UDINT;
        nReadWriteRegsCount     : UDINT;
        nRdQuantity             : WORD;
        nRdMBAddr                       : WORD;
        nWrQuantity             : WORD;
        nWrMBAddr               : WORD;
        arrRdData               : ARRAY [1..9] OF WORD;
```

```
        arrWrData               : ARRAY [1..9] OF WORD;
END_VAR
```



Nach steigender Flanke von "bExecute" und erfolgreicher Ausführung des ReadWriteRegs-Befehls, befinden sich in arrRdData die gelesenen Daten der Register und die Daten aus arrWrData werden in die Register geschrieben.

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8.0	PC (i386)	TcModbusSrv.Lib

TwinCAT MODBUS TCP Server : Modbus Server-Funktionalität

Übersicht

Der Server kann Modbus-Funktionen über TCP/IP empfangen. Die Modbus-Register und Modbus-Ein/Ausgänge werden dann auf Bereiche der PLC gemappt. Für die Konfiguration dieses Mappings wird der TwinCAT Modbus TCP/IP-Server Konfigurator verwendet.

TwinCAT MODBUS TCP Server : Modbus Server-Funktionalität

Mapping zwischen Modbus und Ads

In Modbus sind die folgenden vier Adressierungs-Bereiche definiert:

Modbus-Bereiche	Datentyp	Zugriff
digitale Eingänge (Discreete Inputs)	1 Bit	nur Lesen
digitale Ausgänge (Coils)	1 Bit	Lesen und Schreiben
Eingangs-Register	16 Bit	nur Lesen
Ausgangs-Register	16 Bit	Lesen und Schreiben

Die einzelnen Bereiche können mit den Adressen 0 - 0xFFFF angesprochen werden. Der Ads-Server mappt diese auf die einzelnen Ads-Bereiche. Die Standardeinstellung wird in der folgenden Tabelle dargestellt:

Modbus-Bereiche	Modbus-Adress	Ads-Bereich	
digitale Eingänge (Inputs)	0x0000 - 0x7FFF	**Indexgruppe** 0xF021 - Prozessabbild der physikalischen Eingänge (Bit-Zugriff)	**Indexoffset** 0x0
	0x8000 - 0x80FF	**Name der Variablen im SPS-Programm** .mb_Input_Coils	**Datentyp** ARRAY [0..255] OF BOOL
digitale Ausgänge (Coils)	0x0000 - 0x7FFF	**Indexgruppe** 0xF031 - Prozessabbild der physikalischen Ausgänge (Bit-Zugriff)	**Indexoffset** 0x0
	0x8000 - 0x80FF	**Name der Variablen im SPS-Programm** .mb_Output_Coils	**Datentyp** ARRAY [0..255] OF BOOL
Eingangs-Register (Input Registers)	0x0000 - 0x7FFF	**Indexgruppe** 0xF020 - Prozessabbild der physikalischen Eingänge	**Indexoffset** 0x0
	0x8000 - 0x80FF	**Name der Variablen im SPS-Programm** .mb_Input_Registers	**Datentyp** ARRAY [0..255] OF WORD
Ausgangs-Register (Registers)	0x0000 - 0x2FFF	**Indexgruppe** 0xF030 - Prozessabbild der physikalischen Ausgänge	**Indexoffset** 0x0
	0x3000 - 0x5FFF	0x4020 - SPS-Memory-Bereich	0x0
	0x6000 - 0x7FFF	0x4040 - SPS-Daten-Bereich	0x0
	0x8000 -	**Name der Variablen im SPS-Programm**	**Datentyp** ARRAY [0..255] OF

0x80FF	.mb_Output_Registers	WORD

Wie man in der Tabelle sieht, werden die Adressen 0x8000-0x80FF in allen vier Bereichen auf eine Variable in der SPS gemappt. Damit dies funktioniert, müssen die Variable zu den globalen Variablen des SPS-Programms hinzugefügt werden:

```
VAR_GLOBAL
   mb_Input_Coils       : ARRAY [0..255] OF BOOL;
   mb_Output_Coils      : ARRAY [0..255] OF BOOL;
   mb_Input_Registers   : ARRAY [0..255] OF WORD;
   mb_Output_Registers  : ARRAY [0..255] OF WORD;
END_VAR
```

Beispiel zum Schreiben mehrerer Register:

```
PROGRAM Test
VAR
   M1 AT%MB10 : ARRAY [0..3] OF WORD;
   fbWriteRegs : FB_MBWriteRegs;
   bWriteRegs : BOOL;
   arrValue : ARRAY [0..3] OF WORD  := 123,234,567,889;
END_VAR

fbWriteRegs.sIPAddr := '';
fbWriteRegs.nQuantity := 4;
fbWriteRegs.nMBAddr := 16#3005;
fbWriteRegs.cbLength := SIZEOF(arrValue);
fbWriteRegs.pSrcAddr := ADR(arrValue);
fbReadCoils.tTimeout := T#1s;
fbWriteRegs.bExecute := bWriteRegs;
fbWriteRegs();
```

Um einen Modbus-Befehl an den Server zu verschicken, verwenden wird den FB_MBWriteRegs Baustein. Dabei muss der Parameter sIPAddr der lokalen IP-Adresse oder einem Leerstring entsprechen. In diesem Beispiel wurde die Adresse 0x3005 als Startadresse gewählt. In der vorherigen Tabelle kann man nun unter Ausgangs-Register und der entsprechenden Adresse die Ads-Indexgruppe 0x4020 ablesen. Dabei handelt sich um den Merkerbereich. Der Byte-Offset des Merkerbereichs lässt sich wie folgt berechnen:

Byte-Offset im Merkerbereich = (0x3005 - 0x3000) * 2 = 10

Dies entspricht somit der Variable M1 AT%MB10. Nachdem man bWriteRegs aufruft, wird also das Array arrValue in den Merkerbereich und somit in die Variable M1 geschrieben. Wenn die Adresse 0x8000 in fbWriteRegs.MBAddr eingetragen wird, werden die Daten in die globale Variable mb_OutputRegisters geschrieben.

Beispiel zum Schreiben mehrerer digitaler Ausgänge :

```
PROGRAM TestReadCoilsIO
VAR
QX1 AT%QX0.5 : BOOL;
QX2 AT%QX0.6 : BOOL;
QX3 AT%QX0.7 : BOOL;
QX4 AT%QX1.0 : BOOL;
QX5 AT%QX1.1 : BOOL;
QX6 AT%QX1.2 : BOOL;
QX7 AT%QX1.3 : BOOL;
QX8 AT%QX1.4 : BOOL;
Q1 AT%QB0 : ARRAY [0..1] OF BYTE;

fbReadCoils : FB_MBReadCoils;
bReadCoils : BOOL;
nValue : BYTE;
END_VAR

fbReadCoils.sIPAddr := '';
fbReadCoils.nQuantity := 8;
fbReadCoils.nMBAddr := 16#0005;
fbReadCoils.cbLength := SIZEOF(nValue);
fbReadCoils.pDestAddr := ADR(nValue);
fbReadCoils.tTimeout := T#1s;
fbReadCoils.bExecute := bReadCoils;
fbReadCoils();
```

In diesem Beispiel wird der FB_MBReadCoils Baustein verwendet und die Adresse 0x0005 als Startadresse gewählt. In der Tabelle kann man nun unter digitale Ausgänge und der entsprechenden Adresse die Ads-Indexgruppe 0xF031 ablesen. Dabei handelt sich um das Prozessabbild der physikalischen Ausgänge (Bit-Zugriff). Aus der Adresse ergibt sich der Indexoffset 0x0005. Nachdem man bWriteCoils aufruft, wird also das Byte nValue in die Bits 5 -12 dieses Bereiches geschrieben und somit auch in die Variablen QX1-QX8.

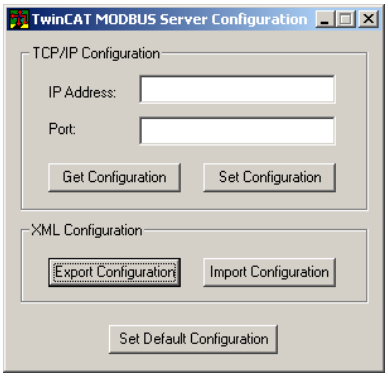
TwinCAT MODBUS TCP Server : Modbus Server-Funktionalität

Konfiguration

Bemerkung: Unter CE kann die Konfiguration nicht geändert werden.

Für die Konfiguration des Modbus Servers wird der TwinCAT Modbus TCP/IP-Server Konfigurator verwendet.
Mit dem Konfigurator kann die Ip-Adresse und der TCP-Port eingestellt werden. Außerdem ist es möglich das Mapping zu verändern. Dazu kann man eine XML-Konfigurationsdatei, die die Mapping-Information enthält, importieren.

Zu beachten ist, dass für die Verwendung des Konfigurators das TwinCAT system gestoppt sein muss. Die Konfigurationsdaten werden in der Datei Default.tps im TwinCAT Verzeichnis gespeichert. Die Konfiguration kann also mit dieser Datei gesichert oder auf einen Zielrechner kopiert werden.



Auswahl der IP-Adresse und des TCP-Ports

Mit Hilfe des Buttons "Get Configuration" können die aktuellen Werte aus der Datei Default.tps gelesen werden. Diese werden dann in die jeweiligen Edit-Felder "IP Address" und "Port" eingetragen. Standardmäßig wird die lokale IP-Adresse(Leer-String) und der Port 502 verwendet. Änderungen der Werte werden mit "Set Configuration" übernommen.

Auslesen der Mapping-Information

Mit "Export Configuration" kann die aktuelle Mapping-Information als XML-Datei gespeichert werden. Außerdem enthält die XML-Datei noch die TCP/IP-Konfiguration des Servers.

Beispiel eines einfachen Mappings:

```
<Configuration>
   <Port>502</Port>
   <IpAddr>172.16.3.217</IpAddr>
   <Mapping>
      <InputRegisters>
         <MappingInfo>
            <StartAddress>0</StartAddress>
            <EndAddress>4095</EndAddress>
            <IndexGroup>61472</IndexGroup>
            <IndexOffset>0</IndexOffset>
         </MappingInfo>
      </InputRegisters>
      <OutputRegisters/>
      <InputCoils/>
      <OutputCoils/>
   </Mapping>
</Configuration>
```

Im Beispiel ist der Port als 502 und die IP-Adresse als 172.16.3.217 definiert. Außerdem wird nur ein einziger Mapping-Bereich definiert. Die Modbus-Eingangsregister 0 bis 4095 werden auf die ADS-Indexgruppe 61472 (=0xF020 -Prozessabbild der physikalischen Eingänge) mit dem Indexoffset 0 gemappt.

Einlesen der Mapping-Information

Mit "Import Configuration" kann die Mapping-Information aus einer XML-Datei importiert werden. Am einfachsten ist es zuvor "Export Configuration" aufzurufen und die damit erstellte XML-Datei zu editieren und dann wieder zu importieren.

So könnte man an dem vorherigen Beispiel einigen Änderungen vornehmen:

```
<Configuration>
   <Port>502</Port>
   <IpAddr>172.16.3.217</IpAddr>
   <Mapping>
      <InputRegisters>
         <MappingInfo>
```

```
            <StartAddress>0</StartAddress>
            <EndAddress>4095</EndAddress>
            <IndexGroup>61472</IndexGroup>
            <IndexOffset>0</IndexOffset>
          </MappingInfo>
        </InputRegisters>
        <OutputRegisters>
          <MappingInfo>
            <StartAddress>0</StartAddress>
            <EndAddress>4095</EndAddress>
            <IndexGroup>61488</IndexGroup>
            <IndexOffset>0</IndexOffset>
          </MappingInfo>
        </OutputRegisters>
        <InputCoils/>
        <OutputCoils/>
    </Mapping>
</Configuration>
```

Nachdem diese Datei mit "Import Configuration" importiert worden ist, steht die neue Mapping-Information zur Verfügung. Zusätzlich werden nun die Modbus-Ausgangsregister 0 bis 4095 auf die Ads-Indexgruppe 61488 (=0xF030 -Prozessabbild der physikalischen Ausgänge) mit dem Indexoffset 0 gemappt. Anstatt den Ads-Bereich per Adresse festzulegen kann auch ein Variablenname verwendet werden. Bei Registern muss dieser vom Typ WORD oder ARRAY OF WORD und bei digitalen Ein/Ausgängen vom Typ BOOL oder ARRAY OF BOOL sein.