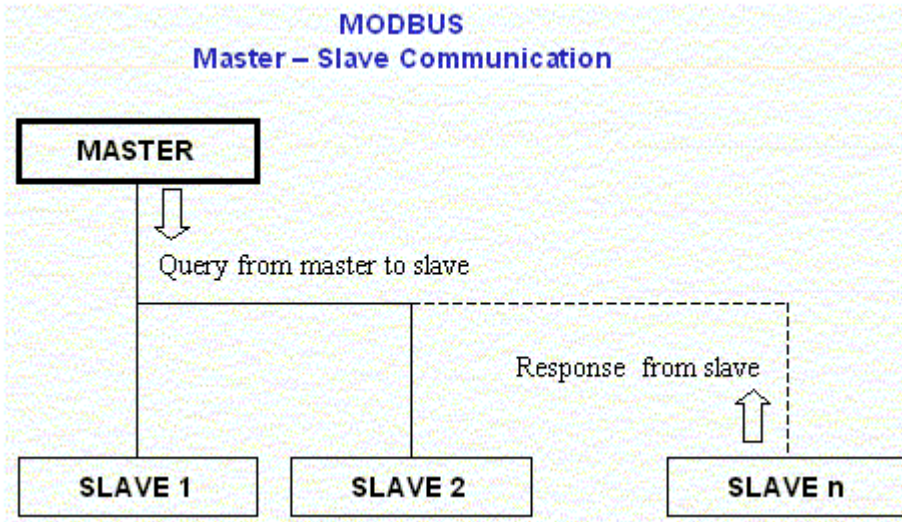




Product:	MODBUS Setup
Product version:	V1.1
Document ID:	UM-MODBUS Setup
Doc revision:	A
Written/Apr.:	RE / SL
Date:	8. Oct. 2008

Industrial Control Design AS



MODBUS Setup

User Manual

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

Contents

1. INTRODUCTION.....	3	3.0.2. XML Example.....	10
1.1. About.....	3	4. ERRORPACKETS.....	11
1.2. Create a ModbusIOServer.....	3	4.0.1. Description.....	11
1.2.1. Setting up Modbus on CDP.....	3	4.0.2. XML Example.....	11
1.2.2. Protocol selection.....	4	5. MODBUS ON TCP.....	12
2. GENERAL I/O-SERVER CONFIGURATION.....	5	5.0.1. Create a ModbusTCPIOServer.....	12
2.1. fs.....	5	5.0.2. Configure ModbusTCPIOServer.....	12
2.2. SendFrequency.....	5	5.1. NetworkInterface.....	13
2.2.1. Description.....	5	5.1.1. Description.....	13
2.2.2. XML Example.....	5	5.1.2. XML Example.....	13
2.3. IOConfig/Node.....	5	6. MODBUS ON UDP.....	14
2.3.1. Description.....	5	6.0.1. Create a ModbusUDPIOServer.....	14
2.3.2. Example XML.....	5	6.0.2. Configure ModbusUDPIOServer.....	14
2.4. Packet.....	6	6.1. NetworkInterface.....	15
2.4.1. Description.....	6	6.1.1. Description.....	15
2.4.2. Example XML.....	6	6.1.2. XML Example.....	15
2.4.3. Elements.....	6	7. MODBUS ON SERIAL.....	16
2.5. Inputs.....	6	7.0.1. About.....	16
2.5.1. Description.....	6	7.0.2. Create a ModbusSerialIOServer.....	16
2.5.2. Example XML.....	6	7.0.3. Configure ModbusSerialIOServer.....	16
2.5.3. Elements.....	6	7.1. Serial Configuration.....	17
2.6. Outputs.....	7	7.1.1. Description.....	17
2.6.1. Description.....	7	7.1.2. Example XML.....	17
2.6.2. Example XML.....	7	7.2. Unit.....	18
2.6.3. Elements.....	7	7.2.1. Description.....	18
2.7. AnalogChannel Scaling.....	8	7.2.2. XML Example.....	18
2.8. Alarms.....	8	7.3. Packet.....	19
2.9. Signals.....	8	7.3.1. Description.....	19
2.10. Parameters.....	9	7.3.2. XML Example.....	19
2.11. Routing.....	9	3. INITPACKETS.....	10
3. INITPACKETS.....	10	3.0.1. Description.....	10
3.0.1. Description.....	10		

1. Introduction

1.1. About

Modbus is a standard format for sending and retrieving data over a communication-medium like Ethernet and a Serial cable, but is not restricted to these mediums. The documentation for the Modbus protocol can be downloaded from <http://www.modbus.org/>

The modbus-protocol is a request-reply protocol and offers services specified by function codes. Modbus function codes are elements of Modbus request/reply protocol data units. The documentation from <http://www.modbus.org/> specifies the function-codes and the packet-formats. You might need this if you need to communicate using a modbus-function not specified in the file 'Models\ModbusFunctions.xml'.

Please read the 'Packets and FunctionDefinitions' CDP Manual to learn more about packets and function-definitions, as this is a requirement for setting up the MODBUS configuration correctly.

CDP 2.3.1.0 and earlier supports Modbus acting on the controller as a Master only. This means that external devices can not connect to CDP using modbus; it is the CDP Application that must connect to the external device.

To communicate with slave devices on Modbus protocol from a CDP Application, you must use a Modbus IOserver which will translate the Modbus packets to CDP signals (input) and CDP signals into Modbus packets (output).

1.2. Create a ModbusIOserver

1.2.1. Setting up Modbus on CDP

To set up Modbus communication on CDP, these criteria must be met:

1. In the Application.xml file, you must include (after the </Console> tag, before the <Components> tag):

```
<Definitions>
  <Definition src="Models\ModbusFunctions.xml"></Definition>
</Definitions>
```

This will tell the CDP DefinitionManager about the name of, and how, the various modbus protocol data units are built up. If you have other definitions needed by your IOserver, add them here.

2. You must activate a ModbusIOserver component, and set up a packet and function-code (specified by name, see Models\ModbusFunctions.xml) to use for sending/receiving the packet. CDP Supports Modbus via TCP, UDP and Serial RTU and ASCII. This will instantiate a TCP Modbus IOserver:

```
<Subcomponents>
  <!-- Subcomponents are put here... -->
  <!-- ..... -->
  <Subcomponent Name="WagoIO" Model="ModbusTCPIOserver"></Subcomponent>
  <!-- more Subcomponents are put here... -->
</Subcomponents>
```

1.2.2. Protocol selection

CDP supports MODBUS on TCP, UDP and Serial.

TCP is a connection-oriented protocol. Due to the nature of its implementation, the socket layer will retry faulty transmissions up to a certain point. Depending on your transmission timeout, this means that old data might be re-transmitted due to network congestion. This might impact the real-time behavior of your application.

UDP is a connection-less protocol. There is no “layer” keeping track of the packets that are sent, so if a packet does not arrive at the recipient, it's lost.

Modbus on Serial is a variation of the MODBUS protocol, specific for use on a Serial line.

Take care when choosing the protocol to use. Depending on your application's needs, one protocol might be more suitable than another.

Different Modbus IO Servers cover the protocols:

IO Server name	Protocol
ModbusTCPIO Server	Modbus-TCP (On Ethernet)
ModbusUDPIO Server	Modbus-UDP (On Ethernet)
ModbusSerialIO Server	Modbus-serial, RTU or ASCII (on serial RS232/ RS422/ RS485)

2. General I/O-Server configuration

I/O-Server Configuration is done by modifying the component xml file inside the Application\Components\ folder. It should not be necessary to modify the model xml file.

2.1. fs

The <fs> element specifies the frequency of which to run the Online/Offline checking in the IOserver. <fs>100</fs> means 100 Hz, so a check to see if data has been received is then done every 10 milliseconds.

2.2. SendFrequency

2.2.1. Description

SendFrequency specifies how often to send packets.

2.2.2. XML Example

```
<SendFrequency>5</SendFrequency>
```

Element	Description
SendFrequency	The frequency in Hz to send packets (requests).

2.3. IOConfig/Node

2.3.1. Description

IOConfig is an XML element that wraps the packet and Input / Output configuration. A minimal configuration is shown below.

2.3.2. Example XML

```
<IOConfig>
  <Node Name="TestIO">
    <Packet>
      <Inputs>
      </Inputs>
      <Outputs>
      </Outputs>
    </Packet>
  </Node>
</IOConfig>
```

2.4. Packet

2.4.1. Description

Contains the Inputs and outputs to send.

2.4.2. Example XML

```
<Packet Name="IOsignals" FunctionCode="MODBUSReadWriteMultipleRegisters" NetworkConvert="1">
  <Inputs>
    <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="3">
      <Channel Nr="0" Type="short" Name="750-454 AI1" Min="0" Max="32767"></Channel>
      <Channel Nr="1" Type="short" Name="750-454 AI2" Min="0" Max="32767"></Channel>
    </ChannelGroup>
    <ChannelGroup Type="Digital" NumberOf="2" Offset="2" ModuleNr="1" SizeInBytes="2">
      <Channel Nr="0" Type="bool" Name="750-403 DI1"></Channel>
      <Channel Nr="1" Type="bool" Name="750-403 DI2"></Channel>
    </ChannelGroup>
  </Inputs>
  <Outputs>
    <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="0">
      <Channel Nr="0" Type="short" Name="750-554 AO1" Min="0" Max="32767"></Channel>
      <Channel Nr="1" Type="short" Name="750-554 AO2" Min="0" Max="32767"></Channel>
    </ChannelGroup>
    <ChannelGroup Type="Digital" NumberOf="2" Offset="2" ModuleNr="2" SizeInBytes="2">
      <Channel Nr="0" Type="bool" Name="750-501 DO1"></Channel>
      <Channel Nr="1" Type="bool" Name="750-501 DO2"></Channel>
    </ChannelGroup>
  </Outputs>
</Packet>
```

2.4.3. Elements

Packet Attributes	Description
Name	Application-wide unique name for the packet
FunctionCode	A function code to look up in the DefinitionManager. The FunctionCode defines how the input and output data are put into the data-packet that is sent to the device. See the document Packets and FunctionDefinitions for more information.
NetworkConvert	Set to 1 to do byte conversion (network conversion) on data bigger than one byte.

2.5. Inputs

2.5.1. Description

Contains the ChannelGroups and Channels / signals to receive input from the physical Input Modules.

2.5.2. Example XML

```
<Inputs>
  <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="0">
    <Channel Nr="0" Type="short" Name="Input module 0.0"></Channel>
    <Channel Nr="1" Type="short" Name="Input module 0.1"></Channel>
  </ChannelGroup>
</Inputs>
```

2.5.3. Elements

Element	Description
ChannelGroup	An enclosing element for a group of channels/signals.
Channel	A Signal / Channel that correspond to one value received in.

ChannelGroup	Description
Type	Can be 'Analog' or 'Digital'

ChannelGroup	Description
NumberOf	The Number of channels in this ChannelGroup
ModuleNr	The physical module number
Offset	The offset for this channel group. Modbus uses this for specifying the destination register.
SizeInBytes	Only for Digital ChannelGroups: Specifies the size in bytes of the ChannelGroup. This is usually set to 2, but please note that for digital packets sent with the MODBUSReadCoils /MODBUSWriteCoils FunctionCodes, you must set SizeInBytes to the actual size in bytes that the digital I/O's use (i.e. <=8 bits requires 1 byte, >8<=16 bits requires 2 bytes etc.).

Channel Attribute	Description
Number	The number in a sequence beginning at 0, must be last channel number+1. On Digital channels this also signifies the bit position.
Type	The c++ data type, can be bool, char, byte, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, float or double. Typical values are bool and short.
Name	The signal name for this channel. Feel free to use more understandable names than what is used in the example, like 'Pressure', 'Oil Level' and so on.

2.6. Outputs

2.6.1. Description

Contains the ChannelGroups and Channels/signals to send as output from the physical modules.

2.6.2. Example XML

```
<Outputs>
  <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="0">
    <Channel Nr="0" Type="short" Name="module out 0.0"></Channel>
    <Channel Nr="1" Type="short" Name="module out 0.1"></Channel>
  </ChannelGroup>
</Outputs>
```

2.6.3. Elements

Element	Description
ChannelGroup	An enclosing element for a group of channels/signals.
Channel	A Signal / Channel that correspond to one value received in.

ChannelGroup	Description
Type	Can be 'Analog' or 'Digital'
NumberOf	The Number of channels in this ChannelGroup
ModuleNr	The physical module number
Offset	The offset for this channel group. Modbus uses this for specifying the destination register.
SizeInBytes	Only for Digital ChannelGroups: Specifies the size in bytes of the ChannelGroup. This is usually set to 2, but please note that for digital packets sent with the MODBUSReadCoils /MODBUSWriteCoils FunctionCodes, you must set SizeInBytes to the actual size in bytes that the digital I/O's use (i.e. <=8 bits requires 1 byte, >8<=16 bits requires 2 bytes etc.).

Channel Attribute	Description
Number	The number in a sequence, starting at 0, must be last number+1. On Digital channels this also signifies the bit position.

Channel Attribute	Description
Type	The c++ data type, can be bool, char, byte, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, float or double. Typical values are bool and short.
Name	The signal name for this channel.

2.7. AnalogChannel Scaling

Analog Channels can do multipoint scaling each time they are read or written. Please see the document 'AnalogChannel with Multipoint scaling.pdf' in the Doc folder where you installed CDP for more information about this. On a Windows install, there is also a shortcut placed on 'Start Menu'-'>CDP'-'>Doc'-'>IOServers'.

2.8. Alarms

The following alarms can trigger from this IOserver:

Alarm Name	Description
Transmission Error	An error is causing the transmission of signals to fail

In addition, you can set up alarms to trigger directly on a Channel mask.

A Channel Alarm is set up in XML like this:

```
<Channel Nr="0" Type="short" Name="4-20 mA input" ErrorMask="0x0003"
AlarmMessageCommand="Stop" AlarmMessageDestination=".ControlCode" AlarmText="Cable break on
Valve feedback" Description="A Cable break was detected on the first input module in the Winch
IO Cabinet (cable from winch speed valve feedback)"></Channel>
```

The Alarm will get the name of the channel, and “ Alarm” is appended to that name. For this reason, make sure the Channel Name is less than 25 characters to avoid a problem with ShortName being greater than 31 characters.

Channel Alarm Attribute	Description
ErrorMask	Required. Specifies a Mask to AND (&) with the channel. If the result is non-zero, the alarm is set (see Timeout below)
AlarmMessageCommand	MessageCommand to send when alarm is set.
AlarmMessageDestination	MessageDestination for the AlarmMessageCommand.
AlarmText	Text to set in the Alarm, visible in a visualizer
AlarmDescription	Description to set in the alarm, visible in a visualizer
Timeout	Time in seconds that the error condition must be active, before the alarm is set. Default value 0.

2.9. Signals

The following signals are in the IOserver:

Signal Name	Description
Send-Receive Roundtrip time	The time used for outputting and inputting data, including signal conversion.
OutputDisabled	Only used in conjunction with CDP Redundancy: Is set to 1 if parameter 'RD output disable control' is set to 1 and the RDmanager is not in the Active state, else it is 0. This signals tells if the outputs are written out to the physical modules(if value=0) or not (value=1).

2.10. Parameters

Parameter Name	Description
RD output disable control	Only used in conjunction with CDP Redundancy: Set to 1 to make RDManager disable output if RDManager is not in Active State.
SignalTimeout	Decimal number. When communication fails, this is the number of seconds to wait after a failure before the IOServer goes Offline.

2.11. Routing

To get the IOServer to output sensible signals, you will have to route the signals from another component. In CDP version 2.3.1.0 and earlier you can **not** set routing directly on an IOServer such as this. You will have to 'push' the routing from a signal on the same controller as the IOServer is located. 'Push' routing only works on components running inside the same controller. For instance, a 'Sinus' component could have routing on its 'Output' signal set to 'WAGOIPCI0.Analog Output 1', this will effectively 'write' the sinus out to the Module containing that signal.

3. InitPackets

3.0.1. Description

InitPackets are used when communication is (re)established. The packets listed are then sent sequentially.

3.0.2. XML Example

```

<!-- Packets to send to I/O after a successful connect (i.e. set up watchdog...) -->
<InitPackets>
  <Packet Name="StopWAGOWD1" FunctionCode="MODBUSStopWAGOWatchDogStep1" NetworkConvert="1"></Packet>
  <Packet Name="StopWAGOWD2" FunctionCode="MODBUSStopWAGOWatchDogStep2" NetworkConvert="1"></Packet>
  <Packet Name="SetWAGOTimeOut" FunctionCode="MODBUSSetWAGOTimeout500ms" NetworkConvert="1"></Packet>
  <Packet Name="StartWAGOWatchDog" FunctionCode="MODBUSStartWAGOWatchDog" NetworkConvert="1"></Packet>
</InitPackets>
    
```

Element	Description
InitPackets	<p>Surrounding element for packets to send on first-time communication with a device, or when a device has been lost and reconnected. Each packet listed here is sent sequentially in the order they are defined in the XML file, with approximately 100 milliseconds time-delay between each packet.</p> <p>The Initpackets can for instance be used on the WAGO I/O, where it might be required to set up the watchdog so that it sets its outputs to 0 when it receives no communication from the controller. If the external hardware connected to the I/O is set up accordingly, this will prevent the equipment from continuing without control if for instance the Ethernet cable to the I/O node is severed during operation.</p> <p>If you don't need any InitPackets, this element can be removed.</p>

Please note that the FunctionCode(s) specified in the packets in <InitPackets> **must** have been read by the DefinitionManager, as described in 1.2.

4. ErrorPackets

4.0.1. Description

Errorpackets are sent sequentially when a modbus error response has been received.

4.0.2. XML Example

```
<ErrorPackets>
  <Packet Name="ClearPhoenixError" FunctionCode="MODBUSClearPhoenixError" NetworkConvert="1"></Packet>
</ErrorPackets>
```

Element	Description
ErrorPackets	<p>Surrounding element for packets to send to the IONode when a modbus error occurs. Each packet listed here is sent sequentially in the order they are defined in the XML file, with approximately 100 milliseconds time-delay between each packet.</p> <p>The ErrorPackets can for instance be used on a Phoenix I/O module, where a 'clear error status' packet is required on first-time-communication after a communication failure.</p> <p>If you don't need any ErrorPackets, this element can be removed.</p>

Please note that the FunctionCode(s) specified in the packets in <ErrorPackets> **must** have been read by the DefinitionManager, as described in 1.2.

5. Modbus on TCP

5.0.1. Create a ModbusTCPIOServer

To use Modbus on Ethernet / TCP, you must instantiate a component of type ModbusTCPIOServer in your Application.xml file.

Example:

```
<Subcomponents>
  <Subcomponent Name="WagoIO" Model="ModbusTCPIOServer"></Subcomponent>
</Subcomponents>
```

This will tell CDP to look for a file called “WagoIO.xml” in the application subfolder. If the Application name is “Test”, CDP will look for the file “Components\Test\WagoIO.xml”. If you specify src=“Components\IOServer.xml” then it will open that file instead.

5.0.2. Configure ModbusTCPIOServer

Example ModbusTCPIOServer file:

```
<?xml version="1.0" encoding="utf-8"?>
<Component Name="WagoIO" Model="ModbusTCPIOServer">
  <Activate>1</Activate>
  <IOConfig>
    <Node Name="10.0.2.13">
      <Type>ModbusTCP</Type>
      <Number>1</Number>
      <NetworkInterface LocalName="ETH0" RemoteIP="10.0.2.13" RemotePort="502"></NetworkInterface>
      <SendFrequency>200</SendFrequency>

      <!-- Packets to send to I/O after a successful connect (i.e. set up watchdog...) -->
      <InitPackets>
        <Packet Name="StopWAGOWD1" FunctionCode="MODBUSStopWAGOWatchDogStep1" NetworkConvert="1"></Packet>
        <Packet Name="StopWAGOWD2" FunctionCode="MODBUSStopWAGOWatchDogStep2" NetworkConvert="1"></Packet>
        <Packet Name="SetWAGOWTimeOut" FunctionCode="MODBUSSetWAGOWTimeOut500ms" NetworkConvert="1"></Packet>
        <Packet Name="StartWAGOWatchDog" FunctionCode="MODBUSStartWAGOWatchDog" NetworkConvert="1"></Packet>
      </InitPackets>

      <ErrorPackets>
        <!-- No packets are sent on error -->
      </ErrorPackets>

      <!-- Below are the signals to be sent by modbus functioncode MODBUSReadWriteMultipleRegisters -->
      <Packet Name="IOSignals" FunctionCode="MODBUSReadWriteMultipleRegisters" NetworkConvert="1">
        <Inputs>
          <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="3">
            <Channel Nr="0" Type="short" Name="750-454 AI1" Min="0" Max="32767"></Channel>
            <Channel Nr="1" Type="short" Name="750-454 AI2" Min="0" Max="32767"></Channel>
          </ChannelGroup>
          <ChannelGroup Type="Digital" NumberOf="2" Offset="2" ModuleNr="1" SizeInBytes="2">
            <Channel Nr="0" Type="bool" Name="750-403 DI1"></Channel>
            <Channel Nr="1" Type="bool" Name="750-403 DI2"></Channel>
          </ChannelGroup>
        </Inputs>
        <Outputs>
          <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="0">
            <Channel Nr="0" Type="short" Name="750-554 AO1" Min="0" Max="32767"></Channel>
            <Channel Nr="1" Type="short" Name="750-554 AO2" Min="0" Max="32767"></Channel>
          </ChannelGroup>
          <ChannelGroup Type="Digital" NumberOf="2" Offset="2" ModuleNr="2" SizeInBytes="2">
            <Channel Nr="0" Type="bool" Name="750-501 DO1"></Channel>
            <Channel Nr="1" Type="bool" Name="750-501 DO2"></Channel>
          </ChannelGroup>
        </Outputs>
      </Packet>
    </Node>
  </IOConfig>
  <Parameters>
    <Parma Name="SignalTimeout" Value="1" DefaultValue="0" PreviousValue="0" TimeLastChanged="Thu Nov 20 16:39:45 2003"
    Description=""></Parma>
  </Parameters>
</Component>
```

Note: The Component Name and Type of the component xml file must match that which you set up in the Application.xml file.

5.1. NetworkInterface

5.1.1. Description

The NetworkInterface tells the IOserver where to send and receive its packets.

5.1.2. XML Example

```
<NetworkInterface LocalName="ETH0" RemoteIP="10.0.2.13" RemotePort="502"></NetworkInterface>
```

NetworkInterface Attributes	Description
LocalName	Refers to the name of a NetworkInterface as set up in Application.xml.
RemoteIP	The IPAddress of the (remote) device to connect to
RemotePort	The port on the destination device to communicate with. Port 502 is defined for Modbus communication on ethernet.

6. Modbus on UDP

Modbus on Ethernet / UDP is set up almost identical to Modbus on Ethernet / TCP. Nonetheless, the complete setup for Modbus UDP is shown below, modified from the Modbus TCP setup shown in the previous chapter.

6.0.1. Create a ModbusUDPIOServer

To use Modbus on Ethernet / UDP, you must instantiate a component of type ModbusUDPIOServer in your Application.xml file.

Example:

```
<Subcomponents>
  <Subcomponent Name="MyIO" Model="ModbusUDPIOServer"></Subcomponent>
</Subcomponents>
```

This will tell CDP to look for a file called “MyIO.xml” in the application subfolder. If the Application name is “Test”, CDP will look for the file “Components\Test\MyIO.xml”. If you specify src=“Components\IOServer.xml” then it will open that file instead.

6.0.2. Configure ModbusUDPIOServer

Example ModbusUDPIOServer file:

```
<?xml version="1.0" encoding="utf-8"?>
<Component Name="MyIO" Model="ModbusUDPIOServer">
  <Activate>1</Activate>
  <IOConfig>
    <Node Name="10.0.2.13">
      <Type>ModbusUDP</Type>
      <Number>1</Number>
      <NetworkInterface LocalName="ETH0" RemoteIP="10.0.2.13" RemotePort="502"></NetworkInterface>
      <SendFrequency>200</SendFrequency>

      <!-- Packets to send to I/O after a successful connect (i.e. set up watchdog...) -->
      <InitPackets>
        <Packet Name="StopWAGOWD1" FunctionCode="MODBUSStopWAGOWatchDogStep1" NetworkConvert="1"></Packet>
        <Packet Name="StopWAGOWD2" FunctionCode="MODBUSStopWAGOWatchDogStep2" NetworkConvert="1"></Packet>
        <Packet Name="SetWAGOTimeout" FunctionCode="MODBUSSetWAGOTimeout500ms" NetworkConvert="1"></Packet>
        <Packet Name="StartWAGOWatchDog" FunctionCode="MODBUSStartWAGOWatchDog" NetworkConvert="1"></Packet>
      </InitPackets>

      <ErrorPackets>
        <!-- No packets are sent on error -->
      </ErrorPackets>

      <!-- Below are the signals to be sent by modbus functioncode MODBUSReadWriteMultipleRegisters -->
      <Packet Name="IOSignals" FunctionCode="MODBUSReadWriteMultipleRegisters" NetworkConvert="1">
        <Inputs>
          <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="3">
            <Channel Nr="0" Type="short" Name="750-454 AI1" Min="0" Max="32767"></Channel>
            <Channel Nr="1" Type="short" Name="750-454 AI2" Min="0" Max="32767"></Channel>
          </ChannelGroup>
          <ChannelGroup Type="Digital" NumberOf="2" Offset="2" ModuleNr="1" SizeInBytes="2">
            <Channel Nr="0" Type="bool" Name="750-403 DI1"></Channel>
            <Channel Nr="1" Type="bool" Name="750-403 DI2"></Channel>
          </ChannelGroup>
        </Inputs>
        <Outputs>
          <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="0">
            <Channel Nr="0" Type="short" Name="750-554 AO1" Min="0" Max="32767"></Channel>
            <Channel Nr="1" Type="short" Name="750-554 AO2" Min="0" Max="32767"></Channel>
          </ChannelGroup>
          <ChannelGroup Type="Digital" NumberOf="2" Offset="2" ModuleNr="2" SizeInBytes="2">
            <Channel Nr="0" Type="bool" Name="750-501 DO1"></Channel>
            <Channel Nr="1" Type="bool" Name="750-501 DO2"></Channel>
          </ChannelGroup>
        </Outputs>
      </Packet>
    </Node>
  </IOConfig>
  <Parameters>
    <Parma Name="SignalTimeout" Value="1" DefaultValue="0" PreviousValue="0" TimeLastChanged="Thu Nov 20 16:39:45 2003"
    Description=""></Parma>
  </Parameters>
```

```
</Component>
```

Note: The Component Name and Type of the component xml file must match that which you set up in the Application.xml file.

6.1. NetworkInterface

6.1.1. Description

The NetworkInterface tells the IO Server where to send and receive its packets.

6.1.2. XML Example

```
<NetworkInterface LocalName="ETH0" RemoteIP="10.0.2.13" RemotePort="502"></NetworkInterface>
```

NetworkInterface Attributes	Description
LocalName	Refers to the name of a NetworkInterface as set up in Application.xml.
RemoteIP	The IP Address of the (remote) device to connect to
RemotePort	The port on the destination device to communicate with. Port 502 is defined for Modbus communication on ethernet.

7. Modbus on Serial

7.0.1. About

CDP supports using MODBUS on a serial connection (RS232/422/485). There are two types of MODBUS protocols available on Serial connections; RTU mode and ASCII mode. RTU mode is the most efficient of these two, as it transmits binary data. ASCII mode is a human-readable format, and has twice the overhead of RTU mode. The ASCII mode is primarily used for compatibility with very old devices.

The actual packet setup of Modbus Serial is similar to that of MODBUS UDP/TCP.

7.0.2. Create a ModbusSerialIOServer

To use Modbus on a Serial Connection, you must instantiate a component of type ModbusSerialIOServer in your Application.xml file.

Example:

```
<Subcomponents>
  <Subcomponent Name="SerIO" Model="ModbusSerialIOServer"></Subcomponent>
</Subcomponents>
```

This will tell CDP to look for a file called "SerIO.xml" in the application subfolder. If the Application name is "Test", CDP will look for the file "Components\Test\SerIO.xml". If you specify src="Components\IOServer.xml" then it will open that file instead.

7.0.3. Configure ModbusSerialIOServer

Example ModbusSerialIOServer file:

```
?xml version="1.0" encoding="utf-8"?>
<Component Name="SerIO" Model="ModbusSerialIOServer">
  <Activate>1</Activate>
  <IOConfig>
    <Node Name="ModbusSerialNode">
      <Type>WAGO ModbusSerial</Type>
      <SendFrequency>5</SendFrequency>

      <IRQ>0</IRQ>
      <BaudRate>115200</BaudRate>
      <Parity>None</Parity>
      <StopBits>1</StopBits>
      <DataBits>8</DataBits>
      <ClockFrequencyMhz>1.8432</ClockFrequencyMhz>
      <Protocol>None</Protocol>
      <BufferSize>1024</BufferSize>

      <ComPort Number="1" BaseAddress="0" NetworkConvert="1"></ComPort>
      <!-- MaxWaitTimeMS : The max time to wait for a character before trying next packet -->
      <MaxWaitTimeMS>1000</MaxWaitTimeMS>
      <!-- AdditionalPacketDelayMS: Additional time to wait between packets -->
      <AdditionalPacketDelayMS>200</AdditionalPacketDelayMS>
      <!--
      TelegramTimeoutMS: This is used for non-compliant MODBUS devices. After receiving first byte,
      this is the max time to wait for the remaining bytes. Set to 0 to conform
      to MODBUS standard.
      -->
      <TelegramTimeoutMS>1000</TelegramTimeoutMS>

      <ModbusMode>ASCII</ModbusMode><!-- Available modes are: RTU or ASCII -->

      <InterCharacterTimeout>25</InterCharacterTimeout> <!-- 0.0 = disable (not recommended), or else interval (ms)
      multiplied by number of characters to get timeout value (for drop packet) -->
      <Ignore_LRC_CRC_Error>0</Ignore_LRC_CRC_Error>

      <InitPackets>
      </InitPackets>

      <Unit Nr="2"> <!-- The unit to communicate with (unitID/SlaveID), add more Unit elements to communicate with more
      units-->
      <Packet Name="AnalogInput1" FunctionCode="MODBUSReadHoldingRegisters" NetworkConvert="1">
        <Inputs>
```

```

<ChannelGroup Type="Analog" NumberOf="4" Offset="0" ModuleNr="1">
  <Channel Nr="0" Type="short" Name="AI_0"></Channel>
  <Channel Nr="1" Type="short" Name="AI_1"></Channel>
  <Channel Nr="2" Type="short" Name="AI_2"></Channel>
  <Channel Nr="3" Type="short" Name="AI_3"></Channel>
</ChannelGroup>
</Inputs>
</Packet>
<Packet Name="AnalogOutput1" FunctionCode="MODBUSWriteMultipleRegisters" NetworkConvert="1">
  <Outputs>
    <ChannelGroup Type="Analog" NumberOf="2" Offset="0" ModuleNr="2">
      <Channel Nr="0" Type="short" Name="AO_0"></Channel>
      <Channel Nr="1" Type="short" Name="AO_1"></Channel>
    </ChannelGroup>
    <ChannelGroup Type="Analog" NumberOf="2" Offset="2" ModuleNr="3">
      <Channel Nr="0" Type="short" Name="AO_2"></Channel>
      <Channel Nr="1" Type="short" Name="AO_3"></Channel>
    </ChannelGroup>
  </Outputs>
</Packet>

<Packet Name="DigitalInput1" FunctionCode="MODBUSReadCoils" NetworkConvert="1">
  <Inputs>
    <!-- Note that SizeInBytes is 1 due to the fact that MODBUSReadCoils function expects only one byte of data in this
case -->
    <ChannelGroup Type="Digital" NumberOf="8" Offset="0" ModuleNr="0" SizeInBytes="1">
      <Channel Nr="0" Type="bool" Name="DI_0" Invert="0"></Channel>
      <Channel Nr="1" Type="bool" Name="DI_1" Invert="0"></Channel>
      <Channel Nr="2" Type="bool" Name="DI_2" Invert="0"></Channel>
      <Channel Nr="3" Type="bool" Name="DI_3" Invert="0"></Channel>
      <Channel Nr="4" Type="bool" Name="DI_4" Invert="0"></Channel>
      <Channel Nr="5" Type="bool" Name="DI_5" Invert="0"></Channel>
      <Channel Nr="6" Type="bool" Name="DI_6" Invert="0"></Channel>
      <Channel Nr="7" Type="bool" Name="DI_7" Invert="0"></Channel>
    </ChannelGroup>
  </Inputs>
</Packet>

<Packet Name="DigitalOutput1" FunctionCode="MODBUSWriteCoils" NetworkConvert="1" SendInterval="2">
  <Outputs>
    <!-- Note that SizeInBytes is 1 due to the fact that MODBUSReadCoils function expects only one byte of data in
this case -->
    <ChannelGroup Type="Digital" NumberOf="2" Offset="0" ModuleNr="0" SizeInBytes="1">
      <Channel Nr="0" Type="bool" Name="DO_0" Invert="0"></Channel>
      <Channel Nr="1" Type="bool" Name="DO_1" Invert="0"></Channel>
    </ChannelGroup>
  </Outputs>
</Packet>
</Unit>
</Node>
</IOConfig>
<Alarms>
  <Alarm Name="Transmission Error" Text="This alarm triggers when a transmission-error occurs." Level="Warning" Enabled="1"
Set="0" Unacknowledged="0" Description="Transmission-error alarm"></Alarm>
</Alarms>
<Parameters>
  <Parma Name="SignalTimeout" Value="3" DefaultValue="1" PreviousValue="2" TimeLastChanged="Thu Feb 17 18:24:51 2005"
Description="If the timestamps are not updated in more than this timeinterval, the IOserver will go Offline."></Parma>
</Parameters>
<Signals>
  <Signal Name="Send-Receive Roundtrip time" Type="double" Description="The time taken to send and receive all packets for
this IONode."></Signal>
</Signals>
</Component>

```

Note: The Component Name and Type of the component xml file must match that which you set up in the Application.xml file.

7.1. Serial Configuration

7.1.1. Description

The serial configuration is the same in all components using a serial connection. It has XML similar to this:

7.1.2. Example XML

```

<IRQ>0</IRQ>
<BaudRate>19200</BaudRate>
<Parity>None</Parity>
<StopBits>1</StopBits>
<DataBits>8</DataBits>
<ClockFrequencyMhz>1.8432</ClockFrequencyMhz>
<Protocol>None</Protocol>
<BufferSize>1024</BufferSize>
<MultiDrop>None</MultiDrop>
<ComPort Number="1" BaseAddress="0" NetworkConvert="0"></ComPort>

```

Element	Description
IRQ	Serial Port IRQ number. '0' means 'use default' but only works for COM1-COM4.
BaudRate	Serial Port Baudrate to set
Parity	Character Parity. Can be Even, Odd, Mark, Space and None
StopBits	Character Stop bits, can be 1 or 2 bits per character. Note that on RTOS32 you can specify 1.5 stop bits by specifying DataBits to 5 and StopBits to 2
DataBits	Number of Data bits per character. Can be 5,6,7 or 8. Note that DataBits = 8 and StopBits=2 is not allowed.
ClockFrequencyMhz	Clock Frequency of serial port crystal. If a non-standard crystal frequency is used on your controller, specify the frequency here to get correct baud rate calculation.
Protocol	The protocol to use. Supported protocols are: None, XonXoff, RtsCts and DtrDsr
MultiDrop	Valid choices are 'None', 'HalfDuplex' and 'FullDuplex'. This is used to enable RS485 communication, and set whether it should be half duplex (uni-directional) or full duplex (bi-directional).
BufferSize	Used On RTOS32 as the BufferSize to use. Default to 1024 bytes.
ComPort	The Com port settings, see table below

ComPort Attributes	Description
Number	The com port number, '1' as in 'COM1'. Must be 1 or bigger.
BaseAddress	The Base Address for the COM port, typically listed in the BIOS for the controller. A BaseAddress of '0' means 'use default'
NetworkConvert	Set to 1 if you want automatic network byte conversion for values.

7.2. Unit

7.2.1. Description

Modbus Serial can send to several Modbus slaves sequentially. To do this, a Unit element is used to enclose the packets to send to one device. Several units can exist in one ModbusSerial file.

7.2.2. XML Example

```
<Unit Nr="2">
  <Packet Name="AnalogInput1" FunctionCode="MODBUSReadHoldingRegisters" NetworkConvert="1">
    <Inputs>
      <ChannelGroup Type="Analog" NumberOf="4" Offset="0" ModuleNr="1">
        <Channel Nr="0" Type="short" Name="AI_0"></Channel>
        <Channel Nr="1" Type="short" Name="AI_1"></Channel>
        <Channel Nr="2" Type="short" Name="AI_2"></Channel>
        <Channel Nr="3" Type="short" Name="AI_3"></Channel>
      </ChannelGroup>
    </Inputs>
  </Packet>
</Unit>
```

Unit Attributes	Description
Nr	The unit number, or Slave ID, of the device to send the packets enclosed by the Unit.

7.3. Packet

7.3.1. Description

The Packet setup is specified in chapter 2.4. In the ModbusSerialIOServer, you can specify an additional attribute 'SendInterval' on the Packet Element, as described below.

7.3.2. XML Example

Additional Packet Attribute	Description
SendInterval	This packet is sent each SendFrequency*Number. If no SendInterval is specified, SendInterval is set to '1'.