



| | |
|------------------|-------------------|
| Product: | SerialIOServer |
| Product version: | V1.1 |
| Document ID: | UM-SerialIOServer |
| Doc revision: | A |
| Written/Apr.: | RE / SL |
| Date: | 13. Oct. 2008 |

Industrial Control Design AS



SerialIOServer

User Manual

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

Industrial Control Design AS, www.icd.no, support@icd.no, forum.icd.no

Contents

| | |
|---|----------|
| 1. INTRODUCTION..... | 3 |
| 1.1. About..... | 3 |
| <hr/> | |
| 2. CONFIGURATION..... | 4 |
| 2.1. Serial Configuration..... | 4 |
| 2.1.1. Description..... | 4 |
| 2.1.2. Example XML..... | 4 |
| 2.2. IOConfig/Node..... | 5 |
| 2.2.1. Description..... | 5 |
| 2.2.2. Example XML..... | 5 |
| 2.3. Port..... | 5 |
| 2.3.1. Description..... | 5 |
| 2.3.2. Example XML..... | 5 |
| 2.3.3. Elements..... | 5 |
| 2.4. ChannelGroup..... | 5 |
| 2.4.1. Description..... | 5 |
| 2.4.2. Example XML..... | 5 |
| 2.4.3. Elements..... | 6 |
| 2.5. AnalogChannel Scaling..... | 6 |
| 2.6. Alarms..... | 6 |
| 2.7. Signals..... | 7 |
| 2.8. Setting up a Header to match on..... | 7 |
| <hr/> | |
| 3. EXAMPLE XML FILE..... | 9 |

1. Introduction

1.1. About

This document describes how to set up a SerialIOserver Configuration in XML. A SerialIOserver is used to send and receive periodic binary serial data that has a fixed header.

2. Configuration

Configuration is done by modifying the component .xml file inside the Application\Components\ folder.

2.1. Serial Configuration

The serial configuration is explained in the document 'UM-Serial Setup', but is included here for your convenience.

2.1.1. Description

The serial configuration is the same in all components using a serial connection. It contains several elements to manipulate the properties of the Serial communication, and has XML similar to this:

2.1.2. 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 Name | 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 the serial port crystal. If a non-standard crystal frequency is used on your controller, specify the frequency here to get correct baud rate calculation. The default clock-frequency on a serial port is 1.8432 MHz. If you specify another clock frequency here, the baudrate is actually recalculated according to the formula: $\text{newBaudRate} = (1.8432 / \text{ClockFrequencyMhz}) * \text{BaudRate};$ |
| 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).MultiDrop will only function as intended when Protocol is set to None. On RS232, MultiDrop should be set to 'None'. On RS485 2-wire setup, use 'HalfDuplex', and on RS485 4-wire setup use 'FullDuplex'. (Please note that the actual physical wires are actually 3 and 5 respectively). |
| 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. |

| ComPort Attributes | Description |
|--------------------|--|
| 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. |

2.2. IOConfig/Node

2.2.1. Description

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

2.2.2. Example XML

```
<IOConfig>
  <Node Name="TestIO">
    <Port>
    </Port>
  </Node>
</IOConfig>
```

2.3. Port

2.3.1. Description

Contain the ChannelGroups to send.

2.3.2. Example XML

```
<Port Name="MRU0" Alias="Berechnung" Type="Binary">
  <ChannelGroup Type="BinaryInput" PacketDescriptorValue="0x9090" PacketDescriptorOffsetBytes="0"
  PacketDescriptorSizeBytes="2">
    <Channel Nr="0" Type="byte" Name="Status"></Channel>
    <Channel Nr="1" Type="byte" Name="Header"></Channel>
    <Channel Nr="2" Type="short" Name="Roll"></Channel>
    <Channel Nr="3" Type="short" Name="Pitch"></Channel>
    <Channel Nr="4" Type="short" Name="Heave"></Channel>
    <Channel Nr="5" Type="short" Name="Heading"></Channel>
  </ChannelGroup>
  <ChannelGroup Type="BinaryOutput">
    <Channel Nr="0" Type="byte" Name="Out_Start"></Channel>
  </ChannelGroup>
</Port>
```

2.3.3. Elements

| Packet Attributes | Description |
|-------------------|---|
| Name | A name for the Port. Currently not used. |
| Alias | An Alias for the port. Currently not used. |
| Type | A Type for the port. Should be set to Binary. |

2.4. ChannelGroup

2.4.1. Description

Contains the Channels / signals

2.4.2. Example XML

```
<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>
```

2.4.3. 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 Attributes | Description |
|-----------------------------|--|
| Type | Can be 'BinaryInput' or 'BinaryOutput'. 'BinaryInput' are packets that are received in on the Serial Port, while Binary Output packets are sent out from the serial port as often as specified by SendFrequency. |
| PacketDescriptorValue | This is used for BinaryInput packets, to detect the start of packet data. This value should be set to the value of the byte(s) in the packet that are fixed (i.e. they stay the same for each packet, for example the packet header). A maximum of four contiguous bytes can be defined as a packetdescriptor. |
| PacketDescriptorOffsetBytes | This is used for BinaryInput packets, and is the offset (in bytes) from the start of the packet to where the PacketDescriptor is located. |
| PacketDescriptorSizeBytes | This is used for BinaryInput packets, and is the size in bytes of the packet descriptor. Note that legal values for this are 0,1,2,3 or 4. |

| 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. |

2.5. 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'.

2.6. 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. |

2.7. Signals

The following signals are in this IO Server:

| 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). |

2.8. Setting up a Header to match on

Assume that we want to receive the following binary data:

HDER<16 bit value><16 bit value>

“HDER” is the ascii header, and it's followed by two 16-bit values of data. To know where to find the data, we must find the “HDER” header. We do this the following way:

H D E R (ASCII)

converted to HEX:

0x48 0x44 0x45 0x52 (HEX)

The header will then be the 4-byte number that these letters make up:

$(0x48 \ll 24) + (0x44 \ll 16) + (0x45 \ll 8) + (0x52)$

= 0x48444552 (hex)

or 1212433746(decimal).

Since 'HDER' is the first thing in the packet, the `PacketDescriptorOffsetBytes="0"`.
(If a packet consist of 2 bytes of “garbage” or void data *before* the header, `PacketDescriptorOffsetBytes` should be set to 2). Since 'HDER' is four bytes long, set `PacketDescriptorSizeBytes="4"`.

The `PacketDescriptorValue` can be specified as hexadecimal (with a '0x' prefix) or decimal value.

We can then set up the `ChannelGroup` as follows:

```
<ChannelGroup Type="BinaryInput" PacketDescriptorValue="0x48444552"  
              PacketDescriptorOffsetBytes="0" PacketDescriptorSizeBytes="4">  
  <Channel Nr="0" Type="byte" Name="Corresponds to H"></Channel>  
  <Channel Nr="1" Type="byte" Name="Corresponds to D"></Channel>  
  <Channel Nr="2" Type="byte" Name="Corresponds to E"></Channel>  
  <Channel Nr="3" Type="byte" Name="Corresponds to R"></Channel>  
  <Channel Nr="4" Type="short" Name="First 16bit value"></Channel>  
  <Channel Nr="5" Type="short" Name="Second 16bit value"></Channel>  
</ChannelGroup>
```

3. Example XML File

Below is an example XML for a SerialIOserver file:

```
<?xml version="1.0" encoding="utf-8"?>
<Component Name="Serial" Type="SerialIOserver">
  <Activate>1</Activate>
  <IOConfig>
    <Node Name="MRUIONode">
      <IRQ>0</IRQ>
      <BaudRate>9600</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>
      <!-- BaseAddress=0 means use default, networkconvert means use hton*() -->
      <SendFrequency>100</SendFrequency> <!-- Send output packets 100 times per second.-->
      <Port Name="MRU0" Alias="SteuerBerechnung" Type="Binary">
        <ChannelGroup Type="BinaryInput" PacketDescriptorValue="0x9090"
          PacketDescriptorOffsetBytes="0" PacketDescriptorSizeBytes="2">
          <Channel Nr="0" Type="byte" Name="Status"></Channel>
          <Channel Nr="1" Type="byte" Name="Header"></Channel>
          <Channel Nr="2" Type="short" Name="Roll">
            <Unit Name="degrees" HWLow="-1000" HWHHigh="1000" UnitLow="-10" UnitHigh="10"></Unit>
          </Channel>
          <Channel Nr="3" Type="short" Name="Pitch">
            <Unit Name="degrees" HWLow="-1000" HWHHigh="1000" UnitLow="-10" UnitHigh="10"></Unit>
          </Channel>
          <Channel Nr="4" Type="short" Name="Heave">
            <Unit Name="m" HWLow="-1000" HWHHigh="1000" UnitLow="-10" UnitHigh="10"></Unit>
          </Channel>
          <Channel Nr="5" Type="short" Name="Heading">
            <Unit Name="degrees" HWLow="0" HWHHigh="36000" UnitLow="0" UnitHigh="360"></Unit>
          </Channel>
        </ChannelGroup>
        <ChannelGroup Type="BinaryOutput">
          <Channel Nr="0" Type="byte" Name="Out_Start"></Channel>
        </ChannelGroup>
      </Port>
    </Node>
  </IOConfig>
  <Parameters>
    <Parma Name="SignalTimeout" Value="1" DefaultValue="0" PreviousValue="0.1" TimeLastChanged="Wed Aug 04 13:37:13 2004"
      Description=""></Parma>
  </Parameters>
</Component>
```