



Product:	SerialStringDispatcher
Product version:	v1.1
Document ID:	UM-SerialStringDispatcher
Doc revision:	A1
Written/Appr.:	NPE / RE
Date:	21.10.2008

Industrial Control Design AS



SerialStringDispatcher v1.1

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The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.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
1.2. Overview.....	3
<hr/>	
2. APPLICATION CONFIGURATION.....	4
2.1. About.....	4
2.2. How to add the SerialStringDispatcher to a CDP Application.....	4
<hr/>	
3. CONFIGURATION.....	5
3.1. About.....	5
3.1.1. This chapter.....	5
3.1.2. SerialStringDispatcher configuration.....	5
3.2. DispatchString.....	5
3.2.1. Description.....	5
3.2.2. Example.....	5
3.2.3. Elements.....	5
3.2.4. String Attributes.....	6
<hr/>	
4. APPENDIX.....	7
4.1. Example SerialStringDispatcher Component XML-file.....	7

1. Introduction

1.1. About

This document describes how the SerialStringDispatcher component works, and how to set it up and use it with the CDP system. See the following section for a component overview.

1.2. Overview

The SerialStringDispatcher reads strings (or byte-sequences) from a serial COM port, and sends the string to a CDPCOMPONENT (or CDPOBJECT) as a CM_TEXTSTRING message.

The component also receives CM_TEXTSTRING messages which it will queue up and send out on the COM port. It will automatically handle discarding of invalid strings that are received in on the serial line if the received string header or footer does not match, or if the string is too long.

2. Application Configuration

2.1. About

This chapter describes how to instantiate the SerialStringDispatcher component within a CDP application.

2.2. How to add the SerialStringDispatcher to a CDP Application

Add the following to your project's Application.xml:

Inside the <Components> element, add an instance of a SerialStringDispatcher component, for instance:

```
<Component Name="SerialStringDispatcher" Type="SerialStringDispatcher"
src="Components/SerialStringDispatcher.xml"></Component>
```

Or, inside the <Subcomponents> element, add:

```
<Subcomponent Name="SerialStringDispatcher" Type="SerialStringDispatcher"
src="Components/SerialStringDispatcher.xml"> </Subcomponent>
```

This will tell CDP to initialize a component named “SerialStringDispatcher” from a component file located at “Components/SerialStringDispatcher.xml”. Make sure that your Models folder contains a SerialStringDispatcher.xml model file, or the component will not be initialized correctly.

Configuration is done by modifying the component xml file. It should not be necessary to modify the model xml file. An example of SerialStringDispatcher.xml file (component xml) is found in 4.1.

3. Configuration

3.1. About

3.1.1. This chapter

This chapter describes the various configuration parameters of the SerialStringDispatcher component.

3.1.2. SerialStringDispatcher configuration

The SerialStringDispatcher component is based on an SerialIOServer and has the usual serial setup elements. In addition you must specify a section of DispatchString elements to identify strings to receive and send to other component objects. The latter is described in the following sections, while the serial setup elements are described in the user manual for the SerialIOServer, named UM-SerialIOServer.

3.2. DispatchString

3.2.1. Description

The DispatchStrings element contains one or more String elements. The strings (or byte-sequences) that are received from the serial COM port are matched against the attributes of each String. When and if a match is found, the string is sent to the component specified in the MessageDestination attribute. The other attributes are described in the attribute list in section 3.2.4.

3.2.2. Example

```
<DispatchStrings>
  <String Header="$GPGGA" Footer="{0x0d}" MaxLength="82" MessageDestination="..SimpleNMEA"></String>
  <String Header="$*DCT" Footer="{0x0d}{0x0a}" MaxLength="82" MessageDestination="..SimpleNMEA"></String>
</DispatchStrings>
```

3.2.3. Elements

Element	Description
DispatchStrings	Includes String elements used when receiving serial strings.
String	Element for receiving a specific message

3.2.4. String Attributes

Attribute Name	Description
Header	Specifies the contents of the start of the string. You can specify '*' as a "don't care" character. If you need a non-printable character, this can be specified by typing the hex-value of the character enclosed inside "{}". For instance, if you want the character " to be accepted; specify {0x22} in it's place.
Footer	Specifies the contents of the end of the string. You can specify the same special characters here as in the Header.
MaxLength	The maximum length of the string. If this length is exceeded, the string is discarded.
MessageDestination	The destination CDPComponent (or CDPObject) that is to receive the string as a CM_TEXTSTRING message. The message is sent as soon as the Header and Footer matches, and the MaxLength is not exceeded.

4. Appendix

4.1. Example SerialStringDispatcher Component XML-file

```

<?xml version="1.0" encoding="iso-8859-1"?>
<Component Name="SerialStringDispatcher" Type="SerialStringDispatcher">
  <Activate>2</Activate>
  <Debug>0</Debug>

  <IOConfig>
    <Node Name="StringDispatcher">
      <IRQ>0</IRQ>          <!-- 0=use default IRQ setting;only used on RTOS -->
      <BaudRate>4800</BaudRate> <!-- The baudrate to use -->
      <Parity>None</Parity>    <!-- Even,Odd,Mark,Space,None-->
      <StopBits>1</StopBits>  <!-- 1 or 2; 1.5 stopbits is ONLY supported by setting DataBits to 5 and
                               StopBits to 2 (on RTOS)! -->
      <DataBits>8</DataBits>   <!-- 5,6,7 or 8 -->
      <ClockFrequencyMhz>1.8432</ClockFrequencyMhz> <!-- This is only used for RTOS, in case you have a
                                                       special board/PC104 -->
      <Protocol>None</Protocol> <!-- None,XonXoff,RtsCts or DtrDsr -->
      <BufferSize>1024</BufferSize> <!-- 1024 should be ok for most needs; only used on RTOS -->

      <!-- BaseAddress=0 means use default, networkconvert means use hton*() -->
      <ComPort Number="1" BaseAddress="0" NetworkConvert="0"></ComPort>

      <DispatchStrings>
        <String Header="$GPGGA" Footer="{0x0d}" MaxLength="82" MessageDestination="..SimpleNMEA"></String>
      </DispatchStrings>
    </Node>
  </IOConfig>

  <Alarms>
    <Alarm Name="Transmission Error" Text="SerialStringDispatcher transmission-error alarm" Level="Warning"
           Enabled="1" Set="0" Unacknowledged="0" Description="Transmission-error alarm"></Alarm>
  </Alarms>
</Component>

```