



Product:	SimpleNMEA
Product version:	v1.2
Document ID:	UM-SimpleNMEA
Doc revision:	A1
Written/Appr.:	NPE / RE
Date:	17.10.2008

## Industrial Control Design AS



# SimpleNMEA v1.2

## 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The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.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](http://www.icd.no), [support@icd.no](mailto:support@icd.no), [forum.icd.no](http://forum.icd.no)

# Contents

<b>1. INTRODUCTION.....</b>	<b>3</b>
1.1. About.....	3
1.2. Overview.....	3
<b>2. APPLICATION CONFIGURATION.....</b>	<b>4</b>
2.1. About.....	4
2.2. How to add the SimpleNMEA component to a CDP Application.....	4
<b>3. CONFIGURATION.....</b>	<b>5</b>
3.1. About.....	5
3.2. Activate.....	5
3.2.1. Description.....	5
3.2.2. Example.....	5
3.3. fs.....	5
3.3.1. Description.....	5
3.3.2. Example.....	5
3.4. Debug.....	5
3.4.1. Description.....	5
3.4.2. Example.....	5
3.5. ReceiveStrings.....	6
3.5.1. Description.....	6
3.5.2. Example XML.....	6
3.5.3. Elements.....	6
3.5.4. String Attributes.....	7
3.6. SendStrings.....	7
3.6.1. Description.....	7
3.6.2. Example XML.....	7
3.6.3. Elements.....	8
3.6.4. String Attributes.....	8
3.7. Signals.....	8
3.8. Parameters.....	8
<b>4. APPENDIX.....</b>	<b>9</b>
4.1. Send and receive example.....	9
4.1.1. Description.....	9
4.1.2. Application xml file.....	9
4.1.3. Component xml file for sending strings.....	9
4.1.4. Component xml file for receiving strings.....	10

# 1. Introduction

## 1.1. About

This document describes how the SimpleNMEA 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 SimpleNMEA component is designed to accept `CM_TEXTSTRING` messages containing (assumed NMEA) strings as input. These strings are matched against strings set up in the xml file. If a match is found, the string is checksummed. This checksum is compared against the checksum embedded in the string. If the checksum is correct, signals are updated from data in the string according to specifications in the XML file.

The SimpleNMEA component can also send NMEA strings as `CM_TEXTSTRING` messages to other CDPComponents (or CDPObjects). Signals are put into the string per specification in the component xml, the string is checksummed and then sent to the component specified in `MessageDestination`.

# 2. Application Configuration

## 2.1. About

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

## 2.2. How to add the SimpleNMEA component to a CDP Application

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

Inside the <Components> element, add an instance of a CS1000Decoder component, for instance:

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

Or, inside the <Subcomponents> element, add:

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

This will tell CDP to initialize a component named “SimpleNMEA” from a component file located at “Components/SimpleNMEA.xml”. Make sure that your Models folder contains a SimpleNMEA.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 SimpleNMEA.xml file (component xml) is found in Error: Reference source not found.

# 3. Configuration

## 3.1. About

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

## 3.2. Activate

### 3.2.1. Description

Specifies the time to delay startup of the component, after configure has been run.

### 3.2.2. Example

```
<Activate>1</Activate>
```

## 3.3. fs

### 3.3.1. Description

In this specific component, fs represents the frequency for sending strings listed in SendStrings. Note that only one string is sent every ts. Thus, sending two different strings once every second requires fs = 2.

### 3.3.2. Example

```
<fs>2</fs>
```

## 3.4. Debug

### 3.4.1. Description

Use this element to set the debug level of the component. When set to a non-zero value, various debug information about messages and so on will get printed.

### 3.4.2. Example

```
<Debug>1</Debug>
```

## 3.5. ReceiveStrings

### 3.5.1. Description

The ReceiveStrings element contains one or more String elements that the SimpleNMEA component will use to check received messages against. When a match is found, output signals will be updated based on the information provided in the received message.

Each String element in the ReceiveStrings element has a tag, named 'Text'. The 'Text' tag contains the string data to receive. To extract data into a signal, specify the local signalname inside [], for instance [Heading]. This will take the text (assumed numeric) at the specified position and convert it into a double, then put it into the signal with name 'Heading'. To accept any character at a position in the string, specify '\*'. Note that a single '\*' equals one character and not a sequence. To accept a special character (i.e. Non-printable character), specify the hexadecimal character value inside {}, for instance {0x22}. Note that if a received string is shorter or longer than the one specified, all matching elements will still get converted. When receiving empty elements in a string, "", the local signal that is specified at that position will get updated with value "0.0". Finally, do not include checksum field and crlf in string, it's assumed to follow after the string you define.

In addition to the Text tag, each string can/should also include attributes to update signals with NMEA information like telegram periods, time since the last telegram was received, and so on. There are also attributes for specifying a signal to flag timeout, and a parameter for specifying when this signal should be set. The attributes are described in the attribute list in section 3.5.4 and illustrated in the following example, showing a string for receiving a GPS command. Note that a local signal must be added for each attribute.

Finally, it should be noted that strings inside ReceiveStrings are converted to signals as soon as possible after they are received. While strings within the SendStrings element are only generated and sent each fs.

### 3.5.2. Example XML

```
<ReceiveStrings>
  <!--
    Strings follow here.
    [] encloses the local signalname (specified in the <Signals> tag below,
    * means accept any (single) character,
    {} encloses hexadecimal number.
    Do not include checksum field and crlf in string, it's assumed to follow after the string you define.
  -->
  <String Text="$GPGGA,[GPGGA.UTC],[GPGGA.Altitude],N,[GPGGA.Latitude],E" TimeoutParma="GPGGA.Timeout"
    TimedOutSignal="GPGGA.TimedOut" TimeSinceLastTelegramSignal="GPGGA.TimeSinceLast"
    TelegramPeriodSignal="GPGGA.TelegramPeriod">
  </String>
</ReceiveStrings>

<!-- Signals for the GPGGA string -->
<Signals>
<Signal Name="GPGGA.TimedOut" Type="double" Description="Set to one if timeout is reached."/>
<Signal Name="GPGGA.TimeSinceLast" Type="double" Description="Time since previous telegram"/>
<Signal Name="GPGGA.TelegramPeriod" Type="double" Description="Time between last 2 telegrams"/>

<Signal Name="GPGGA.UTC" Type="double" Description="GPGGA.UTC"/>
<Signal Name="GPGGA.Altitude" Type="double" Description="GPGGA.Altitude"/>
<Signal Name="GPGGA.Latitude" Type="double" Description="GPGGA.Latitude"/>
</Signals>

<Parameters>
<Parma Name="GPGGA.Timeout" Value="10" DefaultValue="0" PreviousValue="0" Unit="Sec"
  TimeLastChanged="Thu Oct 16 15:37:02 2008" Description="Maximum time since last telegram received.">
</Parma>
<Parma Name="DiscardCRC" Value="0" DefaultValue="0" PreviousValue="0"
  TimeLastChanged="Thu Oct 16 15:37:06 2008" Description="Disable checksum control.">
</Parma>
</Parameters>
```

### 3.5.3. Elements

Element	Description
ReceiveStrings	Includes String elements used when receiving messages.
String	Element for receiving a specific message

### 3.5.4. String Attributes

Attribute Name	Description
Text	The main attribute within String. See description and example for more details.
TimeoutParma	The TimeoutParma attribute is used for specifying a local parameter. The parameter can then be used for viewing and changing the maximum time to wait for a new telegram/message. If the timeout is reached, the signal specified in the TimedOutSignal is set to 1. See the example above for an illustration.
TimedOutSignal	The signal specified here is set to 1 if we don't receive a message within the time provided by the TimeoutParma.
TimeSinceLastTelegramSignal	The signal that is specified here will be updated with the time since we received a message of the type specified in the Text attribute. Note that this signal is required for the timeout handling to work. The signal is updated every fs. Thus, fs should be set at least two times the rate at which the messages are received.
TelegramPeriodSignal	Shows the time period between the 2 previously received signals of type Text.

## 3.6. SendStrings

### 3.6.1. Description

The SendStrings element is very similar to the ReceiveStrings element. The only difference is the signal specification inside the String Text tag in addition to a 'MessageDestination' tag. Since the signal value has to be converted into text before it's put into the string, the component needs to know how to do this conversion. This is done by specifying a format-string before the signal name. The format-string should be “%A.Bf” where A is number of digits before the '.', and B is number of digits after the '.'. The 'f' indicates that the value is a floating point value (double). For instance, if you want the value of the signal 'Speed' to be printed as an integer, specify [%0.f Speed] in the string at the position where you want the speed signal (it's not a good idea to specify %i if the signal is a double value; the conversion will give unpredictable results at best). Skipping the format-string is identical to adding %.1f, thus only adding a single decimal to the value.

The destination component to send the string to is specified in the 'MessageDestination' tag. To get an alive signal for the destination component, use the NoContactSignal attribute to specify a local signal. This signal can be used to check if the destination component is connected

The strings inside SendStrings are generated and sent each fs of the component, while the strings inside ReceiveStrings are converted to signals as soon as possible after they are received.

### 3.6.2. Example XML

```
<SendStrings>
  <!--
    Strings follow here.
    [] encloses the format string (see sprintf), a space and the local signalname (specified<Signals> tag
    below), Note that you should only use %f in different variations. If you want the signal as an
    integer, use %.0f as format string.If an error occurs when converting the signal into the buffer,
    the signal value will get the text "ERROR".
    * means accept any (single) character,
    {} encloses hexadecimal number.
    Do not include checksum field and crlf in string, its calculated on the position after the string you
    define, and <CR><LF> is appended.
    MessageDestination specifies the component to send the CM_TEXTSTRING to.
  -->
  <String Text="$GDGDT, [%0.2f Direction] Head, [%0.2f Heading]" MessageDestination="..SerialStringDispatcher"
    NoContactSignal="NoContact"></String>
</SendStrings>

<!-- The local output signals -->
<Signals>
  <Signal Name="Heading" Type="double" Value="18.23" Description="Heading"></Signal>
  <Signal Name="Direction" Type="double" Value="20.13" Description="Direction"></Signal>
  <Signal Name="NoContact" type="double" Value="1" Description="No contact with MessageDest."></Signal>
</Signals>
```

### 3.6.3. Elements

Element	Description
ReceiveStrings	Includes String elements used when receiving messages.
String	Element for receiving a specific message.

### 3.6.4. String Attributes

Attribute Name	Description
Text	Contains the text string to send including format-strings for any signals. See description for details.
MessageDestination	Specifies the full or relative path to the destination component.
NoContactSignal	The signal provided here will be set to 1 if the destination component is offline (not connected).

## 3.7. Signals

The SimpleNMEA component contains no default signals. However, local signal must be added when wanting additional information and control. This is described in the SendStrings and ReceiveStrings sections.

## 3.8. Parameters

The following parameters are in the SimpleNMEA component:

Parameter Name	Description
DiscardCRC	Disable checksum control.

# 4. Appendix

## 4.1. Send and receive example

### 4.1.1. Description

The following example describes how to send and receive strings between two SimpleNMEA components, SimpleNMEATestSend and SimpleNMEATestReceive.

### 4.1.2. Application xml file

Add the following two lines to the subcomponents section in Application.xml to instantiate the two components:

```
<Subcomponent Name="SimpleNMEATestSend" Type="SimpleNMEA" src="Components\SimpleNMEATestSend.xml">
</Subcomponent>
<Subcomponent Name="SimpleNMEATestReceive" Type="SimpleNMEA" src="Components\SimpleNMEATestReceive.xml">
</Subcomponent>
```

### 4.1.3. Component xml file for sending strings

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Component Name="SimpleNMEATestSend" Type="SimpleNMEA">
  <Activate>1</Activate>
  <fs>2</fs>
  <Debug>0</Debug>
  <ReceiveStrings>
  </ReceiveStrings>
  <SendStrings>
    <!--
    Strings follow here.
    [] encloses the format string (see sprintf), a space and the local signalname (specified<Signals> tag
    below), Note that you should only use %f in different variations. If you want the signal as an
    integer, use %.0f as format string.If an error occurs when converting the signal into the buffer,
    the signal value will get the text "ERROR".
    * means accept any (single) character,
    {} encloses hexadecimal number.
    Do not include checksum field and crlf in string, its calculated on the position after the string you
    define, and <CR><LF> is appended.
    MessageDestination specifies the component to send the CM_TEXTSTRING to.
    -->
    <String Text="$GDGDT, [%2f Direction] myHeading, [Heading]" MessageDestination="..SimpleNMEATestReceive"
    NoContactSignal="NoContact">
    </String>
  </SendStrings>
  <!-- The local signals to send and receive -->
  <Signals>
    <Signal Name="Heading" Type="double" Value="32.12345" Description="Heading"></Signal>
    <Signal Name="Direction" Type="double" Value="16.789" Description="Direction"></Signal>
    <Signal Name="NoContact" Type="double" Description="No contact with destination."></Signal>
  </Signals>
  <Parameters>
    <Parma Name="DiscardCRC" Value="0" DefaultValue="0" PreviousValue="0" TimeLastChanged="0"
    Description="Disable checksum control."></Parma>
  </Parameters>
</Component>
```

#### 4.1.4. Component xml file for receiving strings

```

<?xml version="1.0" encoding="iso-8859-1"?>

<Component Name="SimpleNMEATestReceive" Type="SimpleNMEA">
  <Activate>1</Activate> <!-- Activate is called from bottom of ConfigureNode to prevent starting the
server before it's configured -->
  <fs>4</fs>
  <Debug>0</Debug>

  <ReceiveStrings>
    <!--
      Strings follow here.
      [] encloses the local signalname (specified in the <Signals> tag below,
      * means accept any (single) character,
      {} encloses hexadecimal number.
      Do not include checksum field and crlf in string, it's assumed to follow after the string you define.
    -->
    <String Text="$GDGDT,[Direction] myHeading,[Heading]" TimeoutParma="GDGDT_Timeout"
      TimedOutSignal="GDGDT_TimedOut" TimeSinceLastTelegramSignal="GDGDT_TimeSinceLast"
      TelegramPeriodSignal="GDGDT_TelegramPeriod">
    </String>
  </ReceiveStrings>

  <SendStrings>
</SendStrings>

  <!-- The local signals to send and receive -->
  <Signals>
    <!-- Signals for the GDGDT string -->
    <Signal Name="GDGDT_TimedOut" Type="double" Description="Set to 1 if timeout is reached."></Signal>
    <Signal Name="GDGDT_TimeSinceLast" Type="double" Description="Time since last telegram."></Signal>
    <Signal Name="GDGDT_TelegramPeriod" Type="double" Description="Time between last 2 telegrams.">
    </Signal>

    <Signal Name="GDGDT_Heading" Type="double" Description="Heading"></Signal>
    <Signal Name="GDGDT_Direction" Type="double" Description="Direction"></Signal>
  </Signals>

  <Parameters>
    <Parma Name="NMEATimeout" Value="5" DefaultValue="5" PreviousValue="5" TimeLastChanged="0"
      Description="Timeout before setting Contact/NoContact signals."></Parma>
    <Parma Name="DiscardCRC" Value="0" DefaultValue="0" PreviousValue="0" TimeLastChanged="0"
      Description="Disable checksum control."></Parma>
  </Parameters>
</Component>

```