



Product:	CDPSim
Product version:	V1.10
Document ID:	UM-CDPSim
Doc revision:	A
Written/Appr.:	RE / SL
Date:	8. Oct. 2008

## Industrial Control Design AS



# CDPSim V1.10

## 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The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.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. What is CDPSim.....	3
1.1.1. Integration with CDP.....	3
1.1.2. How CDPSim works.....	3
1.1.3. Integration algorithms.....	3
1.2. Creating simulator models.....	3
1.3. Changing the Visual Studio project to build with CDPSim.....	4
1.4. Modifying project files to enable CDPSim.....	4
1.5. Implementing the simulation model.....	5
1.6. XML configuration.....	5
1.7. More information.....	5
1.8. Known issues.....	6

# 1. Introduction

This document describes how to use the CDPSim add-on.

## 1.1. What is CDPSim

CDPSim is a toolkit for creating real-time simulations of systems described by differential equations. A high degree of accuracy is obtained by running the simulations at very small time steps, much smaller than the periodic processes of standard CDP components.

### 1.1.1. Integration with CDP

CDPSim uses state variables that are specialisations of standard CDP signals, allowing monitoring by standard CDP tools, and allowing routing to other CDP components on the network. The simulation model behaves like any other CDP component, and can run either as a stand-alone application, as part of a control application or a mix of both.

This means that CDPSim models can easily replace real hardware in Hardware In the Loop tests (HIL tests), simply by routing the signals from the simulated signals instead of from the i/o signals.

### 1.1.2. How CDPSim works

A simulation model is created by inheriting the class *DynamicSimComponent*, and then implementing the virtual method *EvaluateDiffEquations()*. This function defines the behaviour of the simulated sub(system) in form of differential equations. To define the new values of the derivatives, the model can use the state variables of the model itself, the state variables of other models, standard CDP signal, CDP parameters etc. The implementation is done in C++.

The simulation is performed by the SimulatorManager calling this virtual method in the simulator model to update the derivatives of the state variables. CDPSim then integrates the derivatives to calculate new values for the state variables, using the selected integration method and time step for that instance of the model.

### 1.1.3. Integration algorithms

CDPSim supports three different integration algorithms: Euler, Heun and Runge-Kutta. The Runge-Kutta algorithm supports adaptive time-step calculation.

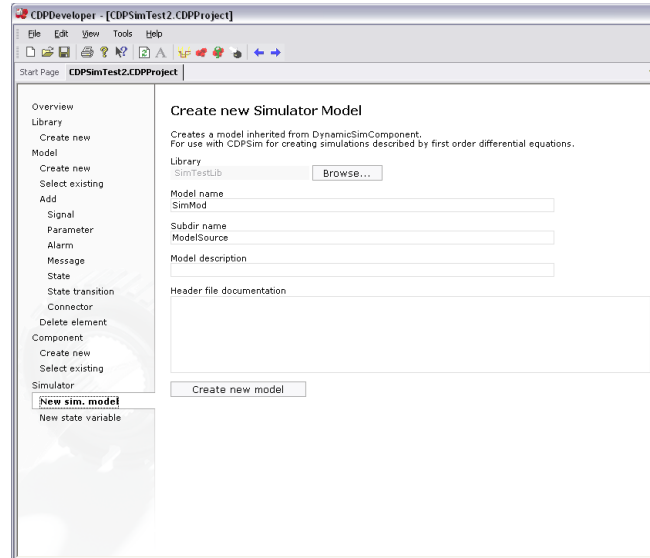
The integration method can be selected individually for each component instance. The choice of algorithm and simulation time step depends on the nature of the simulated system. Stiff systems require smaller time steps, and to prevent instability these (sub)systems should be implemented as separate models to enable running them at smaller time steps, especially when running the complete system at a sufficient small time step would be too CPU intensive for the host computer.

## 1.2. Creating simulator models

To set up the project for running dynamic simulations using CDPSim, follow the steps below.

It is assumed that a project and a library has already been created using CDPDeveloper:

1. Create a new simulator model using the menu **Simulator->New sim. model** in CDPDeveloper.
2. Add state variables to the new model.  
The state variables are special for CDPSim compared to CDP. When a state variable is added to a model, two *Signals* are added, the state variable itself, and its derivative. See below for how to use these in the code.
3. Create an instance of the simulator model using the menu **Component->Create new**



### 1.3. Changing the Visual Studio project to build with CDPSim

4. Open the project in Visual Studio by selecting **Tools->Open current project in DevStudio** in CDPDeveloper.
5. Add you new library to the solution (right click on solution in the solution explorer and select **Add->Existing project, then select *YourLibName.vcproj***). The set project dependencies so that CDP\_Application depends on *YourLibName*.
6. In Visual Studio, add  $\$(CDPBase)\CDPSim$  to the **Additional include directories for you library**. (*YourLibName->Properties->C/C++->General->Additional include directories*).
7. Add  $\$(CDPBase)\CDPSim$  to the **Additional include directories** in the **CDP\_Application** project (*CDP\_Application->Properties->C/C++->General->Additional include directories*).
8. Add *CDPSim\_Debug.lib* or *CDPSim\_Release.lib* to the **Linker->Input->Additional dependencies** section in the **CDP\_Application** project. (*CDP\_Application->Properties->Linker->Input->Additional dependencies*)

### 1.4. Modifying project files to enable CDPSim

9. In CDP\_Application, libraries.h, add:  
`#include <SimulatorLib.h>`
10. Instantiate a **SimulatorManager** component by adding this line inside the <Components> section in Application.xml:  
`<Component Name="Simulator" src="Components/CDP/Simulator.xml"></Component>`
11. Copy the **Simulator.xml** file from the XML templates folder ( $\$(CDPBase)\Templates\XML\Templates\CDP$ ) to your *Application/Components/CDP* folder.
12. Build and start the application. Remember to set the CDP\_Application as the default/startup project, and to set the working directory to  $\dots\Application$  (under *CDP\_Application->Properties->Debug->Working directory*).
13. Send a **Start message** to the simulator using the web browser or CDPBrowser.

## 1.5. Implementing the simulation model

Example: Mass hanging from a spring.

First, add two state variables, position ( $x$ ) and speed ( $v$ ) using the CDPDeveloper, as described in the previous chapter.

Then, add constants or CDPParams for the spring constant ( $k$ ), mass ( $m$ ) and gravity ( $g$ ) also using the CDPDeveloper.

The implementation of EvaluateDiffEquations() would then be something like this:

```
void MySpringModel::EvaluateDiffEquations(double t)
{
    // Typically, something like this should be done here:
    x_ddt = v;
    v_ddt = g - x * k/m;
}
```

## 1.6. XML configuration

Excerpt from the generated component file for the simulation model. See comments for explanation of each parameter.

```
<!-- Not simulation fs, but normal state machine fs.
      Use MinTimeStep to set simulation time step.-->
<fs>10</fs>

<!-- IntegrationMethods: 'Euler', 'Heun' or 'RungeKutta4'.
      They produce errors in the order dt, dt^3 and dt^5, respectively.-->
<IntegrationMethod>RungeKutta4</IntegrationMethod>

<!-- Set AdaptiveTimeStep to 1 to use adaptive time stepping (RungeKutta4 int. only).-->
<AdaptiveTimeStep>0</AdaptiveTimeStep>

<!-- The parameters KappaLow and KappaHigh are used only if AdaptiveTimeStep is enabled.
      When using adaptive time steps with RungeKutta4, an error measure, Kappa is calculated.
      KappaLow and KappaHigh define when to increase or decrease the time step depending on
      this error measure.-->

<!--The time step is increased if Kappa is smaller than this value.-->
<KappaLow>0.04</KappaLow>

<!--The time step is decreased if Kappa is larger than this value.-->
<KappaHigh>0.2</KappaHigh>

<!-- The default time step (for all integration methods).
      When using adaptive time step, it will never be decreased below this limit.-->
<MinTimeStep>1e-3</MinTimeStep>

<!-- The time step will never be increased above this limit (valid only when using adaptive
      time step). If not set, 1/fs will be used.-->
<MaxTimeStep>0.1</MaxTimeStep>
```

## 1.7. More information

The [CDP Forum](http://forum.icd.no) has a separate section for discussing the CDPSim add-on, located here:

<http://forum.icd.no/viewforum.php?f=15> .

## 1.8. Known issues

ICD Bug #	Description
<a href="#">Bug #232</a>	SimulatorManager must be top-level component if other simulator components are top-level components. If all are subcomponents, the SimulatorManager may also be a subcomponent, but must be listed first.

*Table 1 - Known issues in CDPSim V1.10*