Distributed Control and Automation Framework Hands-On Session

This hands-on covers the basics of implementing an application in the Distributed Control and Automation Framework, including using an existing module and developing a new control module. It doesn’t cover development of a new generic I/O or processing module. For this hands on, the framework downloads, and additional documentation, visit the [Reference Designs for Distributed Control Systems](https://decibel.ni.com/content/projects/reference-designs-for-distributed-control-systems) portal on [ni.com/referencedesigns](https://decibel.ni.com/content/groups/reference-designs).

## Set-up (Already configured on Alliance Day machines)

Navigate to <https://decibel.ni.com/content/docs/DOC-41727> and download the .vipc file attached to that page. This package can be installed to any LabVIEW version from 2013 to present.

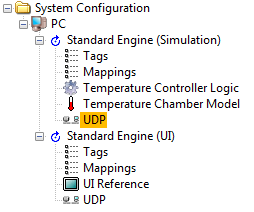
# Exercise 1: Configuring a Simple Temperature Control Application

**Introduction**

This exercise demonstrates the implementation of a simple temperature chamber controller application using the Distributed Control and Automation Framework (CDAF). This exercise makes use of a model of the chamber to simulate its I/O and allows users to define the setpoint and PID gains of the control algorithm through a simple user interface.

During the exercise you will identify how inputs and outputs from different modules are mapped into the DCAF Tag Bus to provide a synchronized communication between modules.

The Temperature Simulated Controller consists of 2 DCAF engines: the UI and the Temperature Controller Simulation.



*Figure 1.1*

This Simulation Engine is implemented in such a way that it will be easy to migrate from a simulation to a real time controller by adding a cRIO target and moving the Simulation Engine below it.

Notice the different components of each engine listed in Fig 1.1. Tags, Mappings and UDP repeat in both engines. The Temperature Controller Logic and the Temperature Chamber Model are Modules created specifically for the Simulation Engine as well as the UI Reference Module is only present in the UI Engine. Here is a brief description of each of these components.

**Tags:** Tags are scalar variables that can be mapped as inputs or outputs in any module within a DCAF engine. This pane allows the user to add or remove tags to the engine and configure tag properties such as Tag Name, Data Type, Default Value, and Description.

**Mappings:** This pane allows the user to connect tags as inputs or outputs in DCAF modules. For example, an Output Channel from the Temperature Controller Logic Module can be mapped into a Tag which can be mapped as an Input Channel to the Temperature Chamber Model Module.

**UDP:** This pane is also present in both components. It is designed to share tags between Engines by mapping each tag as an Engine Input or Output. All the tags that are intended to be shared between engines need to be defined in the Tags Pane of each engine with the same names.

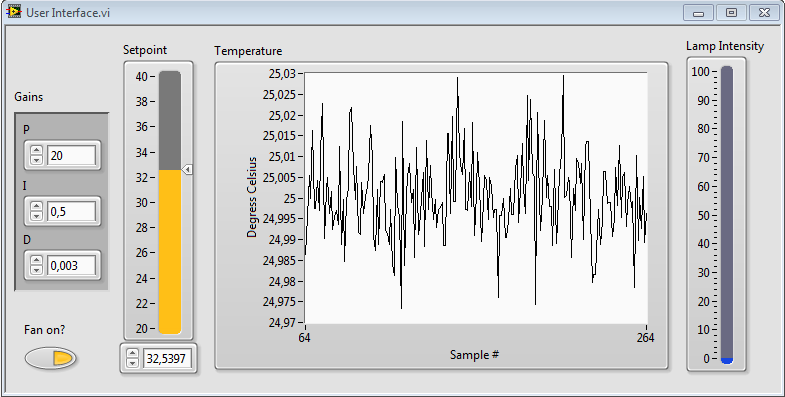
**UI Reference:** This module only works in the UI Engine. It provides a front panel with the necessary controls and indicators to allow the user interact with the DCAF Engine.

**Temperature Controller Logic:** This is a custom DCAF Model designed to provide a Temperature Control Logic. If the Simulation Engine is moved to a cRIO Target and the Temperature Chamber Model is replaced with real IO, this module could remain the same.

**Temperature Chamber Model:** This module provides a simulated model of a Temperature Chamber. This module could be replaced or overwritten to eventually provide IO from a real Temperature Chamber.

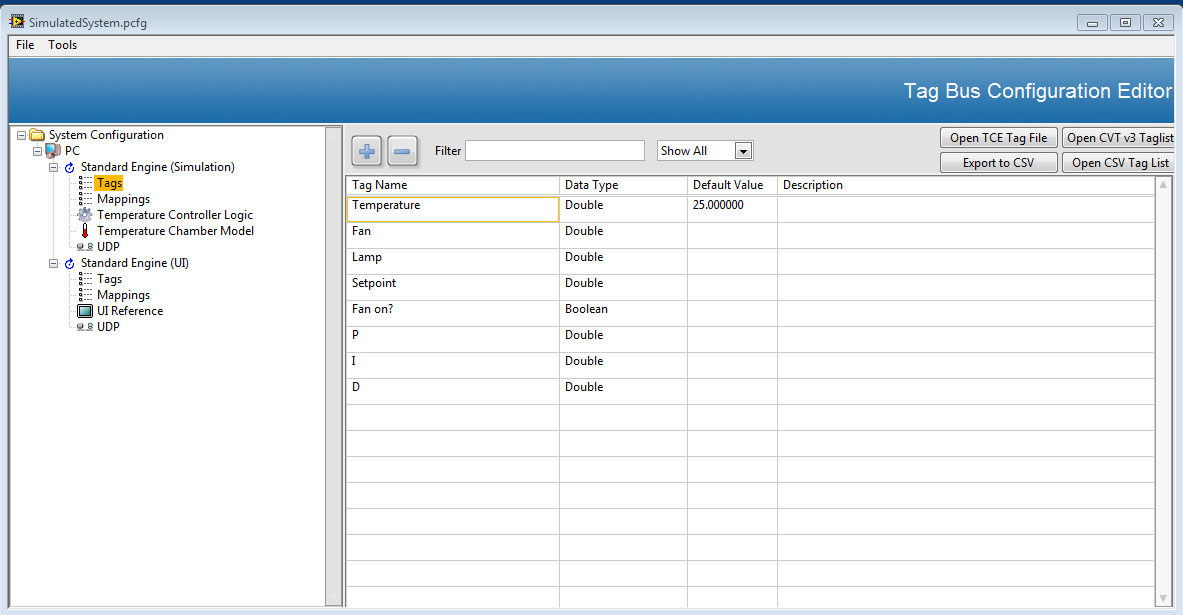
**Execution Steps**

1. Open up the Standard Configuration Editor for TBD by navigating in LabVIEW to **Tools>>TBDF>>Launch Standard Configuration Editor…**
2. Navigate within the editor to **Tools>>Edit Plugin Search Paths**.
3. Add a search path to the TBM plugins for this example located at **\\Desktop\DCAF Hands On\Temperature Controller\Exercise 1** if it’s not already there. Also confirm that the standard vi.lib file paths are specified as shown below for the version of LabVIEW that you are using.
4. In the DCAF Configuration Editor go to **File>>Open** and search for the SimulatedSystem.pcfg Configuration File located **at** [**\\Desktop\DCAF Hands On\Temperature Controller\Exercise 1**](file:///\\Desktop\DCAF%20Hands%20On\Temperature%20Controller\Exercise%201).
5. Take a couple of minutes to go through each component in the Simulation and UI Engines.
6. Open the Temperature Controller project located at [**\\Desktop\DCAF Hands On\Temperature Controller\Exercise 1**](file:///\\Desktop\DCAF%20Hands%20On\Temperature%20Controller\Exercise%201)**.**
7. Open and run **Host Main.vi**. Try changing the Setpoint and the other controls. Do you see any change in the temperature value displayed in the Graph?



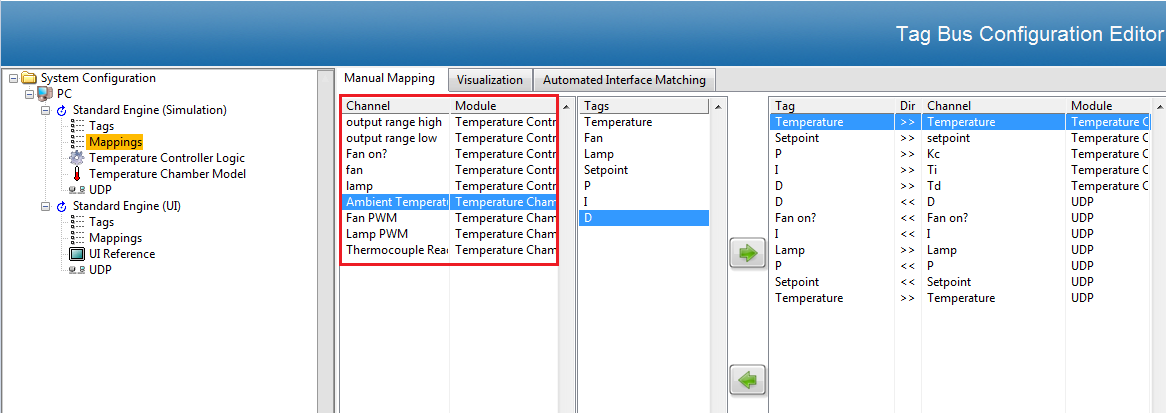
*Figure 1.2*

1. You shouldn’t see any change in the signal since there are some tags that haven’t been connected.
2. Stop the VI and return to the Configuration Editor. We will review the connections in each component on both engines to understand the tag dataflow and connect the tags that are missing to make it run.
3. We will start with the Simulation Engine. First select the Tags component and take a look to the tags.



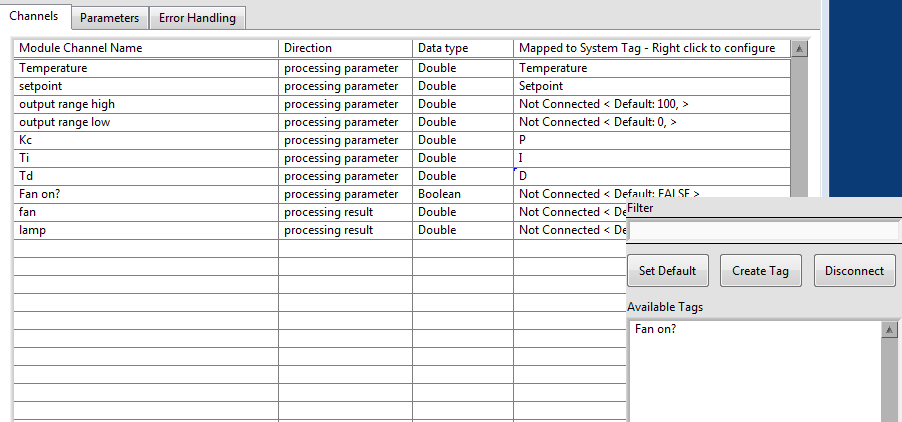
*Figure 1.3*

1. This tags are used as inputs or outputs in the rest of Simulation Engine modules: Temperature Controller Logic, Temperature Chamber Model and UDP. Notice all of them are Doubles except for ***Fan on?****.*
2. Go to ***Mappings*** and select the ***Manual Mapping*** tab. This section will allow you to have a better look of the tag flow in this application. In the left pane you will see all the channels that haven’t been mapped. Just take a look, don’t make changes.



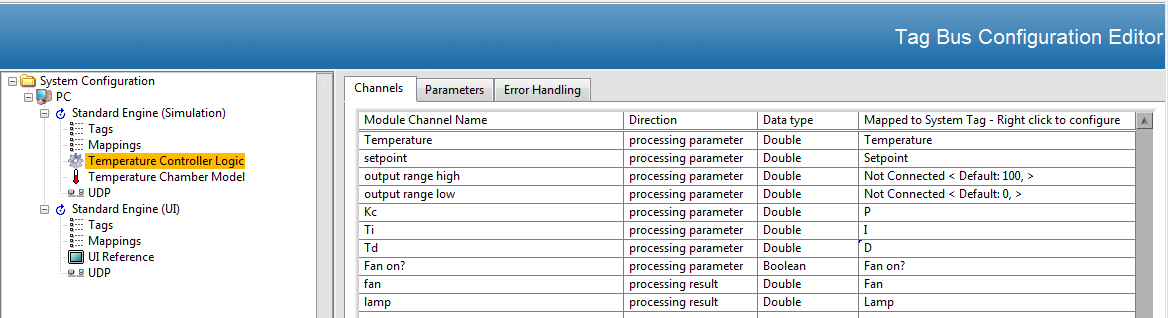
*Figure 1.4*

1. Go to the Temperature Controller Logic Module. Notice there are two variables that don’t appear in the Tag list: ***output range high*** and ***output range low***). This are internal variables with constant values defined statically. The rest should be mapped to a tag.
2. The last 3 channels should be connected to a tag (***Fan on?***, ***fan***, and ***lamp***). To connect a channel to a tag take the cursor to the corresponding cell in the ***Mapped to System Tag*** column, right click, and select the corresponding tag from the ***Available Tags*** list.



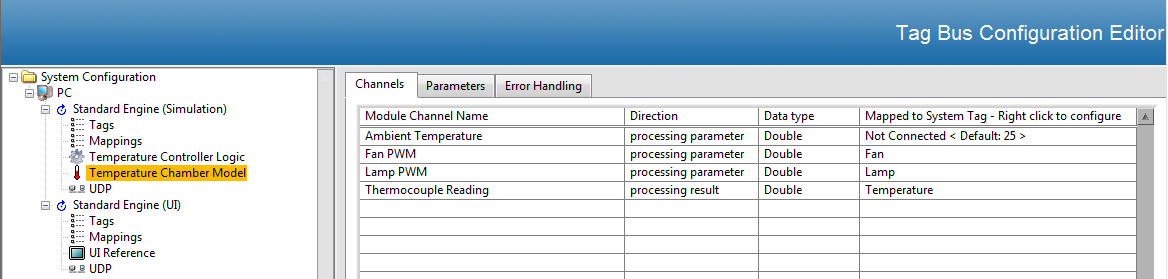
*Figure 1.5*

1. Verify your table looks like the following image:



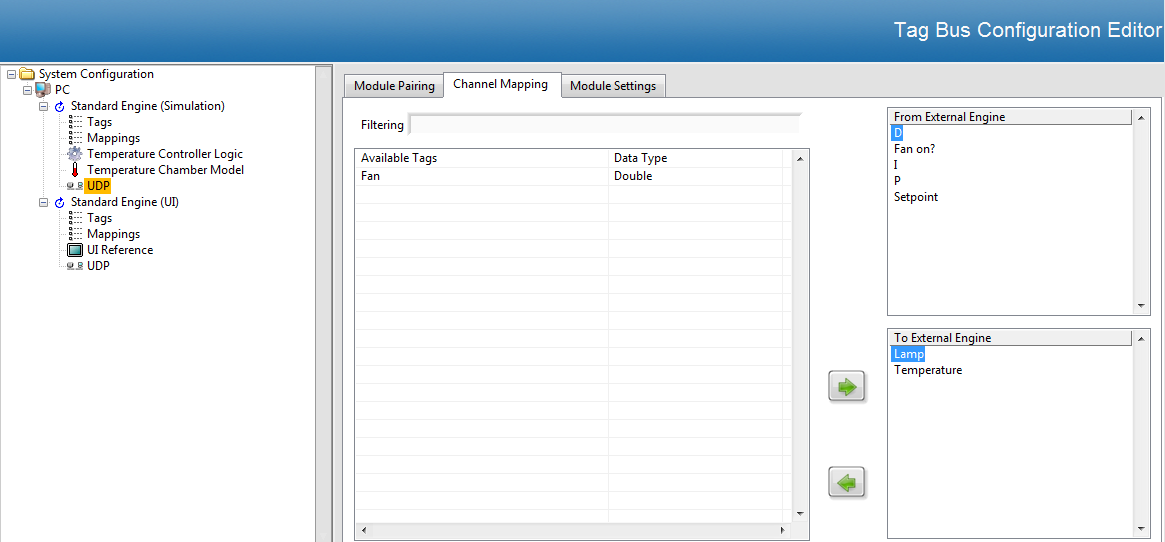
*Figure 1.6*

1. Before going to the next module notice the ***Direction*** column. ***Processing parameters*** are module inputs while ***processing results*** are module outputs. Some of the ***processing parameters*** in this module come from the ***UI Engine*** and others come from the ***Temperature Controller Logic Module***. The two ***processing results*** in this module will go through the ***Tag Bus*** as inputs in the ***Temperature Channel Model***.
2. Go to the ***Temperature Chamber Model Module***. Notice all the channels are disconnected from any tag. The only disconnected channel should be ***Ambient Temperature***. ***Fan PWM*** and ***Lamp PWM*** channels are ***processing parameters*** in this module that should come from *the* ***Temperature Controller Logic Module***. ***Thermocouple Reading*** is a ***processing result*** that should be used as the feedback signal in the ***Temperature Controller Logic Module*** and will also be sent to the ***UI Engine*** to be displayed in the graph. Following the same instructions as in step 14 to map ***Fan PWM***, ***Lamp PWM***, and ***Thermocouple Reading*** channels to ***Fan***, ***Lamp***, and ***Thermocouple*** tags. Verify your table looks like the following image:



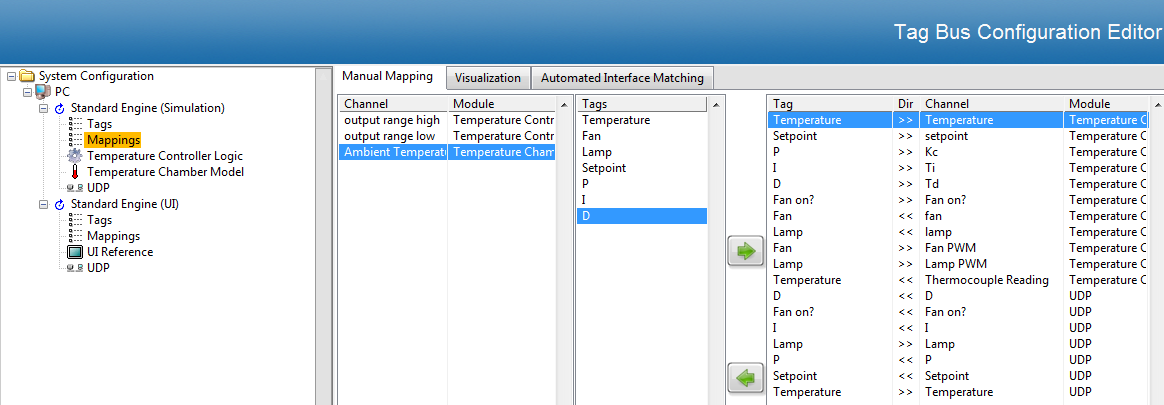
*Figure 1.7*

1. Go to the ***UDP Module*** in the ***Simulation Engine***. Go to the ***Channel Mapping Tab***. Notice the tags in the ***From External Engine*** (Inputs) and ***To External Engine*** (Outputs) boxes. Notice the ***Fan*** tag is still as an ***Available Tag***. There is no need to move it since it is not needed in the ***UI Engine,*** it is only used internally in the ***Simulation Engine.***



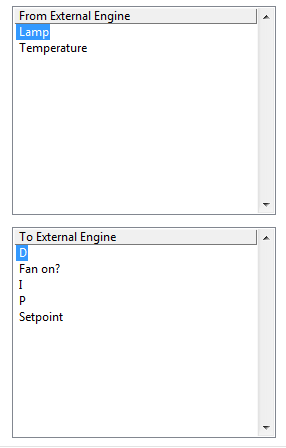
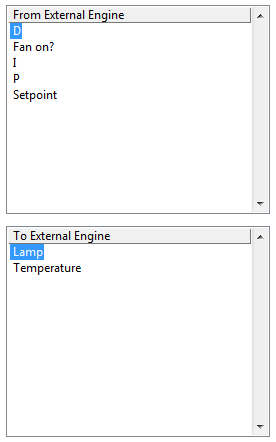
*Figure 1.8*

1. Go back to ***Mappings*** in the ***Simulation Engine***. Notice now there are only 3 channels that haven’t been mapped. There are no tags for those channels since they are configured statically in their corresponding modules or set as default. All the channels that originally were unmapped now appear mapped in the right pane. Take some time to review the mapping directions to have a better understanding of the data flow.



*Figure 1.9*

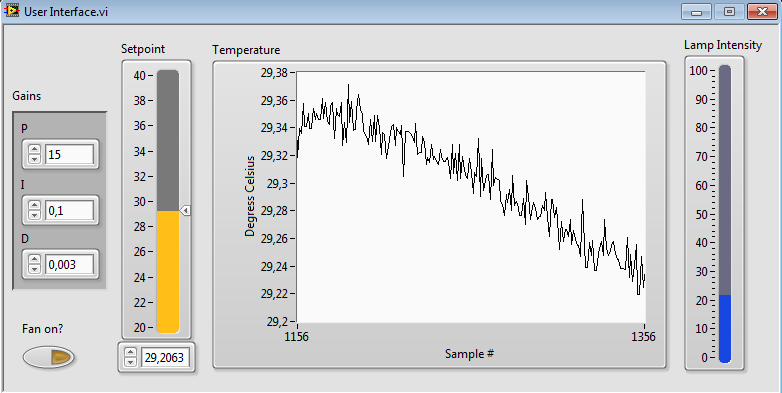
1. Now you have understood how the ***Simulation Engine*** works, step into the different components of the ***UI Engine*** to understand how it interacts with the ***Simulation Engine***. Notice that the Inputs for the ***UI Engine UDP Module*** are the Outputs for the ***Simulation Engine UDP Module*** and viceversa.



Simulation Engine UDP UI Engine UDP

*Figure 1.10*

1. Go to ***File >> Save*** and close the ***Configuration Editor***.
2. Open the ***Temperature Controller Example Project*** if not already open. Open and run ***Host Main.vi***.
3. Modify the ***Setpoint*** and the other controls in the ***UI***. You should now see the temperature being controlled by the ***Simulation Engine***.



*Figure 1.11*