

Cognex Designer Advanced – Alarms & Recipes



COGNEX

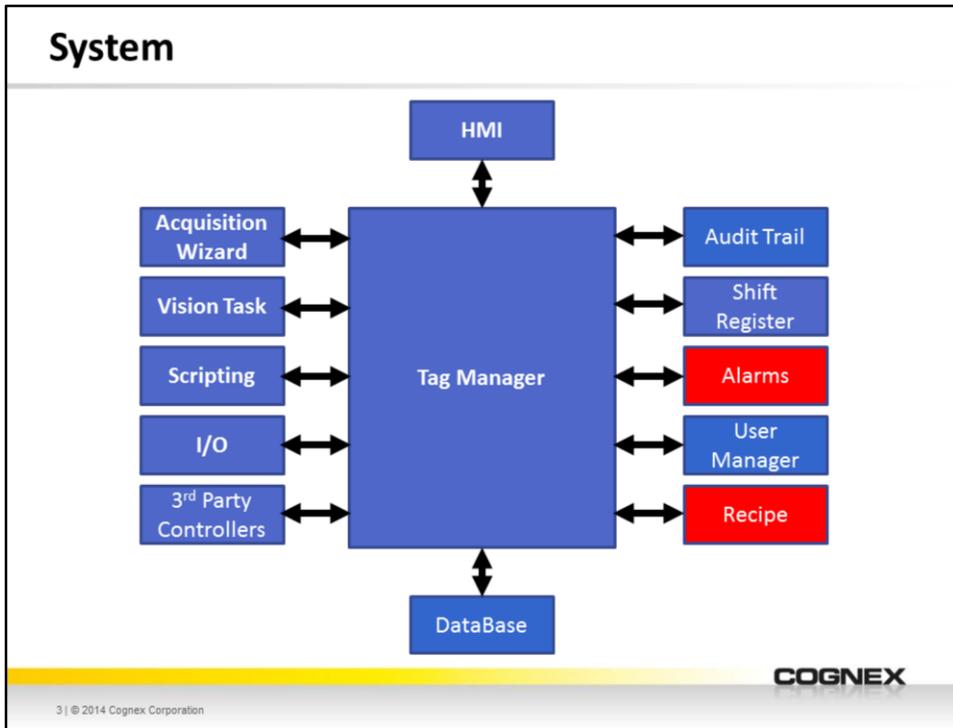
Objective

Cognex Designer Advanced – Alarms & Recipes

- Learn about what creates an alarm
- Explore variable changes through Recipes
- Understand uses for Resources

❖ **Lab: Create a Recipe to change tolerance settings**

COGNEX



Cognex Designer gives you the ability to use alarms and recipes through the use of Systems.

Systems

Ability to Address Change

- Alarms
- Recipes
- Audio/Video (Resources)
- User Access

Explorer

- System
 - Alarms
 - Recipes
 - Resources
 - Settings
 - Tag Manager
 - Users

4 | © 2014 Cognex Corporation

COGNEX

System gives access to:

- Alarms: Flags of issues in the program
- Recipes: Different Tag values based off of a collection of saved values
- Resources: Ability to add images or video to the HMI
- Settings: Control over the program and audit trail
- Tag Manager: Access to the User Defined tags
- Users: Privileges based on set of given criteria

Alarms

Alerts and Actions when Problem Arise

5 | © 2014 Cognex Corporation

COGNEX

Alarms are setup by defining a condition that causes an alarm (i.e. a Device is disconnected). An alarm can have one of four states:

- Inactive: The alarm condition is not active.
- Unacknowledged: The alarm condition is active but not acknowledged
- Acknowledged: The alarm condition is active and acknowledged
- Restored: The alarm condition is not active and not acknowledged.

The state of an alarm will usually change in the following order:

Inactive > Unacknowledged > Acknowledged > Inactive

or

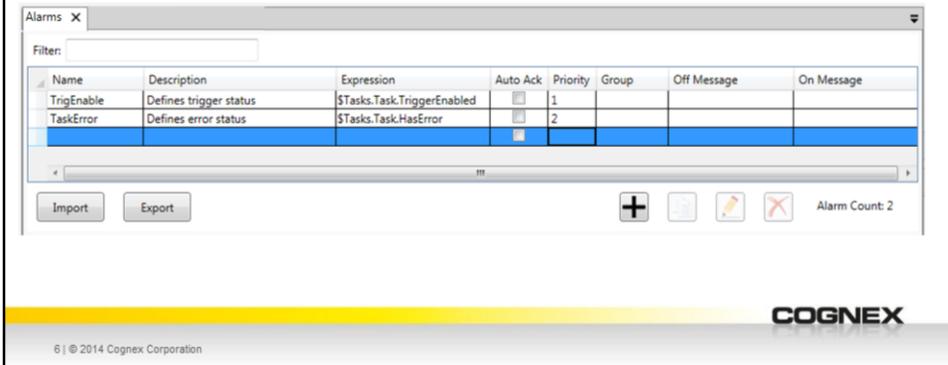
Inactive > Unacknowledged > Restored > Inactive

To acknowledge an Alarm, the Acknowledge function must be called from a script

Alarm Editor

Access to Alarms

- Create new alarms
- Edit existing alarms
- Ability to acknowledge alarms



The Alarm Editor allows the user to create their set of alarms.

- Name: Specifies the name of the alarm; the name must be unique and only contain alphanumeric or underscore characters.
- Description: Specifies additional information about the alarm.
- Expression: Specifies the activation conditions for the alarm.
- Auto Ack: Determines whether or not the alarm should be automatically acknowledged once it has been cleared.
- Priority: Specifies the alarm's priority level (this is an optional setting).
- Group: Specifies the group name in which the alarm resides (this is an optional setting).
- Off Message: Specifies the message that the alarm displays when the Expression condition is false.
- On Message: Specifies the message that the alarm displays when the Expression condition is true.

Implementing Alarms

Check to see if Sequence has an Error

Name	Description	Expression	Auto Ack	Priority	Group	Off Message	On Message
SequenceError		\$Tasks.Task.Sequence.HasErrc	<input type="checkbox"/>	5		Issue Resolved	Cam not Connected

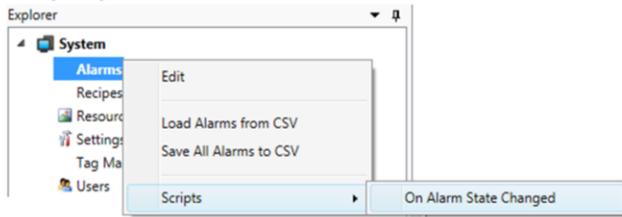
COGNEX

7 | © 2014 Cognex Corporation

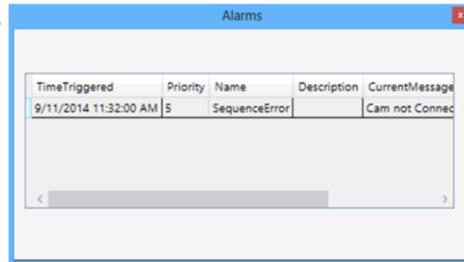
An alarm can be created to check to see if a Sequence Error has taken place. We can give it a priority in case we have multiple alarms and we can state the message to be shown when it goes on an goes off.

Issue in Sequence

Display the Alarms



Write a script that displays a special HMI page.
The HMI page shows a table
with query results from the Audit Database



COGNEX

8 | © 2014 Cognex Corporation

In this case we had created a separate page to display a table that contains the Alarm information.

The script for OnAlarmStateChanged has a call to show the Alarm page that contains the alarm.

Further integration can be done to accept the alarm as well as close the window.

Recipes

Offer Different Configurations

- Part Changeover
- Different set of Requirements
- Flexibility

Saves or Loads a set of Tag values

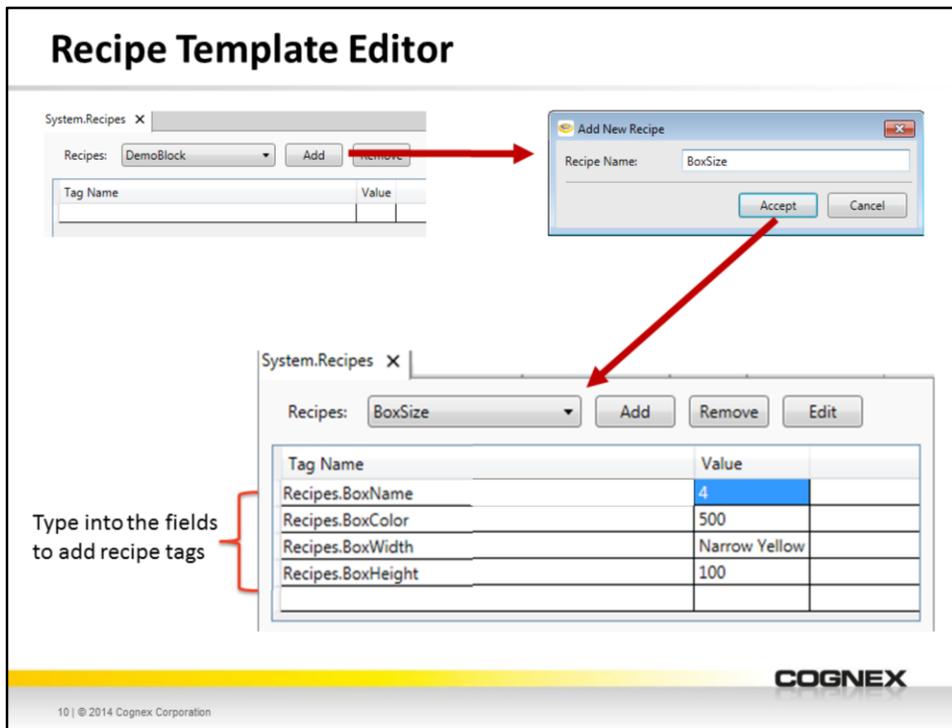


COGNEX

9 | © 2014 Cognex Corporation

Recipes can be used to provide different settings for different configurations. Cognex Designer supports multiple recipes and each recipe can have unlimited configurations. A *Recipe* provides functionality to save and restore values of sets tags.

- The Recipe **Save** function takes the current values of all the tags in the *Recipe* configuration and saves them to disk using the specified name.
- The Recipe **Load** function reads all the values from the specific *Recipe* configuration on disk and loads them into the current tag values.

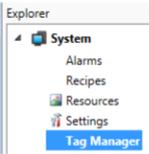


The Recipe Editor is used to create and build recipes. Add a new Recipe and specify a name and the location where the recipe data should be saved.

- **Add** will prompt for a new type of recipe and which folder to save it in.
- **Remove** deletes the Recipe Type.
- **Edit** allows you to change the location of the stored recipes.

Creating Recipe Templates

Define the Tags first



Recipes.BoxColor		Integer	1
Recipes.BoxHeight		Integer	300
Recipes.BoxName		String	Large Green
Recipes.BoxWidth		Integer	400

Then add to tags to your Recipe template

Recipes:

Tag Name	Value
Recipes.BoxName	
Recipes.BoxColor	
Recipes.BoxWidth	
Recipes.BoxHeight	

COGNEX

11 | © 2014 Cognex Corporation

Before creating a recipe, the tags need to be defined. These will be variables used with our program so that the values change as the recipe changes.

After the tags are created, they can be added to the Recipe.

Create an HMI Recipe Page

Add a way to Save / Load different Recipes

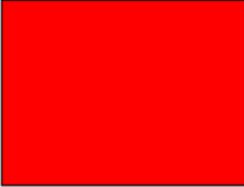
Demonstrates how to Save & Load tags as part of a recipe/product configuration.

[Help Link](#)

Choose a Box Type Medium Red

	Current	Recipe
<code>\$\$Sample.Recipes.BoxName</code>	Medium Red	Medium Red
<code>\$\$Sample.Recipes.BoxColor</code>	Red <input type="button" value="Load"/>	Red
<code>\$\$Sample.Recipes.BoxWidth</code>	320	320
<code>\$\$Sample.Recipes.BoxHeight</code>	240	240

Load/Save By File



COGNEX

12 | © 2014 Cognex Corporation

This application is being pulled from the FunctionSample project that is installed with the Cognex Designer software.

This page allows the user to save and load different recipes. It also provides the means to swap Recipe files instead of using just configurations.

Behind the Scenes

Recipes stored as XML on the PC

The image shows a file explorer window on the left with a list of files: Directory.xml, Large Green.xml, Large Purple.xml, Medium Red.xml, Small Blue.xml, and Small Yellow.xml. To the right, XML code is displayed. Red arrows point from the file names to their corresponding XML snippets. The XML code is structured as follows:

```
<?xml version="1.0"?>
<RecipeConfiguration>
  - <RecipeConfiguration>
    <Name>Large Green</Name>
    <Description/>
  </RecipeConfiguration>
  - <RecipeConfiguration>
    <Name>Medium Red</Name>
    <Description/>
  </RecipeConfiguration>
  - <RecipeConfiguration>
    <Name>Small Blue</Name>
    <Description/>
  </RecipeConfiguration>
  - <RecipeConfiguration>
    <Name>Small Yellow</Name>
    <Description/>
  </RecipeConfiguration>
  - <RecipeConfiguration>
    <Name>Large Purple</Name>
    <Description/>
  </RecipeConfiguration>
</RecipeConfiguration>
```

The XML code continues with a <Recipe> element containing several <Item> elements, each with a <TagName> and <Data64> attribute. The <Data64> attributes contain Base64 encoded values. The <Item> elements are:

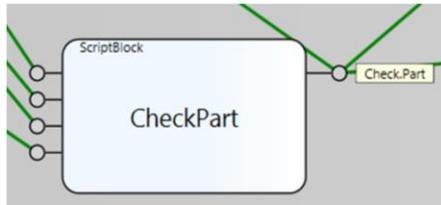
- <Item> with <TagName> Recipes.BoxName</TagName>, <SerializerID> proto</SerializerID>, <TypeID> String</TypeID>, and <Data64> CgTMYXJnZSBHcmVibg==</Data64>
- <Item> with <TagName> Recipes.BoxColor</TagName>, <SerializerID> proto</SerializerID>, <TypeID> Integer</TypeID>, and <Data64> CAE=</Data64>
- <Item> with <TagName> Recipes.BoxWidth</TagName>, <SerializerID> proto</SerializerID>, <TypeID> Integer</TypeID>, and <Data64> ClR=</Data64>
- <Item> with <TagName> Recipes.BoxHeight</TagName>, <SerializerID> proto</SerializerID>, <TypeID> Integer</TypeID>, and <Data64> CJIE=</Data64>

Recipes are stored as .XML files. It is recommended that the Recipe files be saved in the same directory as the project – in this case ...\\Documents\\Cognex Designer\\Samples\\FunctionSample\\Recipes\\BoxSize

The Directory .XML contains the names and description of the individual configurations. This is list of sorts that can be used as Source for a Listbox in order to choose among the different saved configurations. Note that the files are not encrypted though the data is stored as a Base64 value so that any type of variable can be saved including tools or images.

Use Recipes to check part for pass/fail

Integrate into the application using a scriptblock

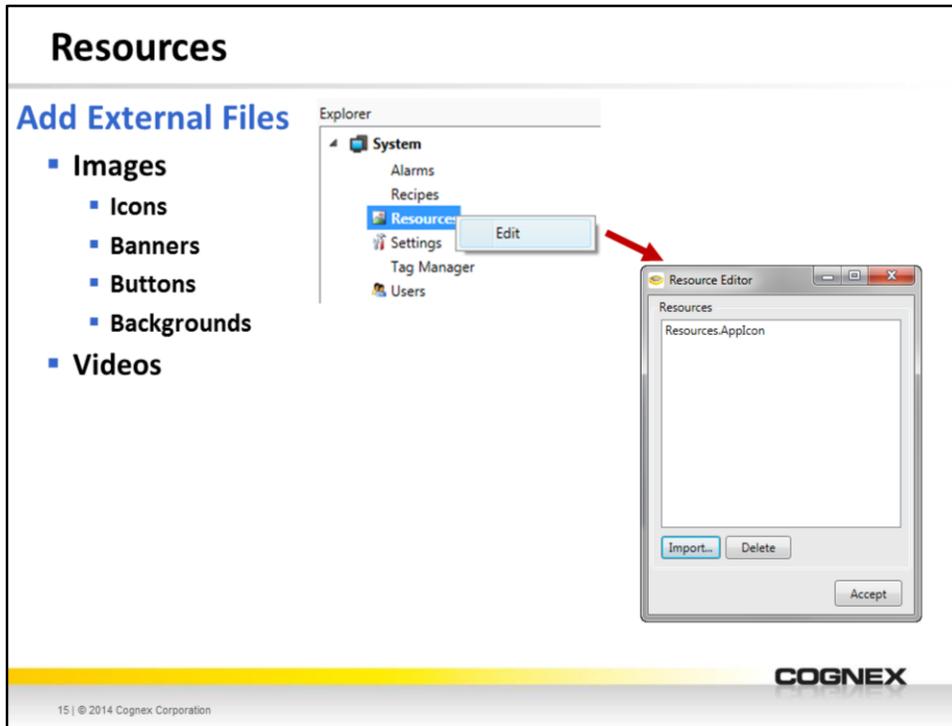


```
public static bool Execute(double Height,double Volume,int Count,double Length)
1 if (Height < ($Recipe.Height))
2
3
4
```

COGNEX

14 | © 2014 Cognex Corporation

Now the Recipe variables can be used within the program for areas such as tolerance checking. As the Recipe changes, so does the tolerance settings and the application is now using the new values.



Resources allows you to bring in images or videos to be displayed within your application.

You would first Import the desired Media file. In this case we are bringing in the generic Cognex Designer Icon to display on our HMI. This could be a Company Logo as well.

Creating Buttons

Use the image as a button

- Create a Script for Mouse Down event

The screenshot illustrates the steps to create a button using an image. On the left, the 'Toolbox' shows the 'Image' control selected under the 'Media' category. A red arrow points from the 'Image' control to a yellow circular icon placed on a cyan background. A context menu is open over the icon, with the 'Scripts' option selected. A second red arrow points from the 'Scripts' menu to a code editor window. The code editor shows a script for the 'OnMouseDown' event:

```
public static void OnMouseDown  
1 SHMI.ShowMessage(" Add your message.");
```

16 | © 2014 Cognex Corporation

COGNEX

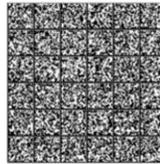
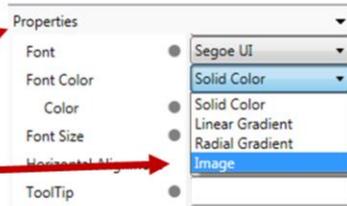
Image is found in the Toolbox for Pages. Place Image on the page and have the Source (Resource) Property reference the imported image from Resources.

The Script for OnMouseDown can now mimic functionality of a button.

Get Funky

Use images as a pattern

- Select a Label on your HMI
- In the Properties for the label select Font Color: Image
- Select the image source
- See the result



COGNEX

In this case we are using a 2D matrix image as the Font Color for the label. This allows for many options for an HMI.

Summary

Alarms & Recipes

- **Learned about what creates an alarm**
- **Understood variable changes through Recipes**
- **Explored some of the uses for Resources**

COGNEX