Session 8: Components

COGNEX

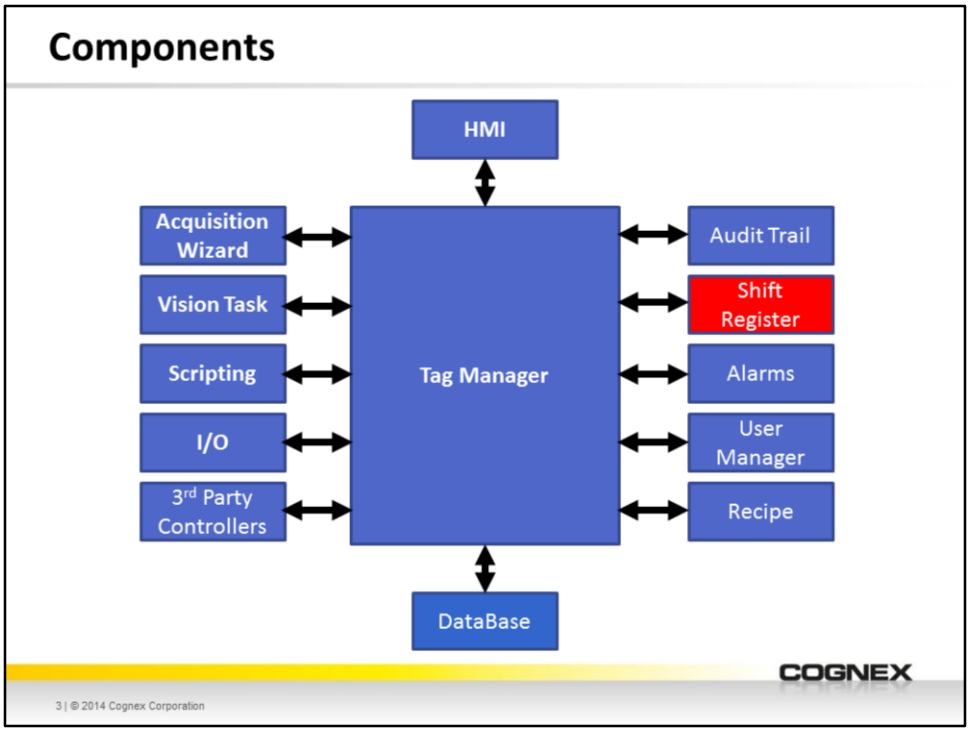
Objective

Components

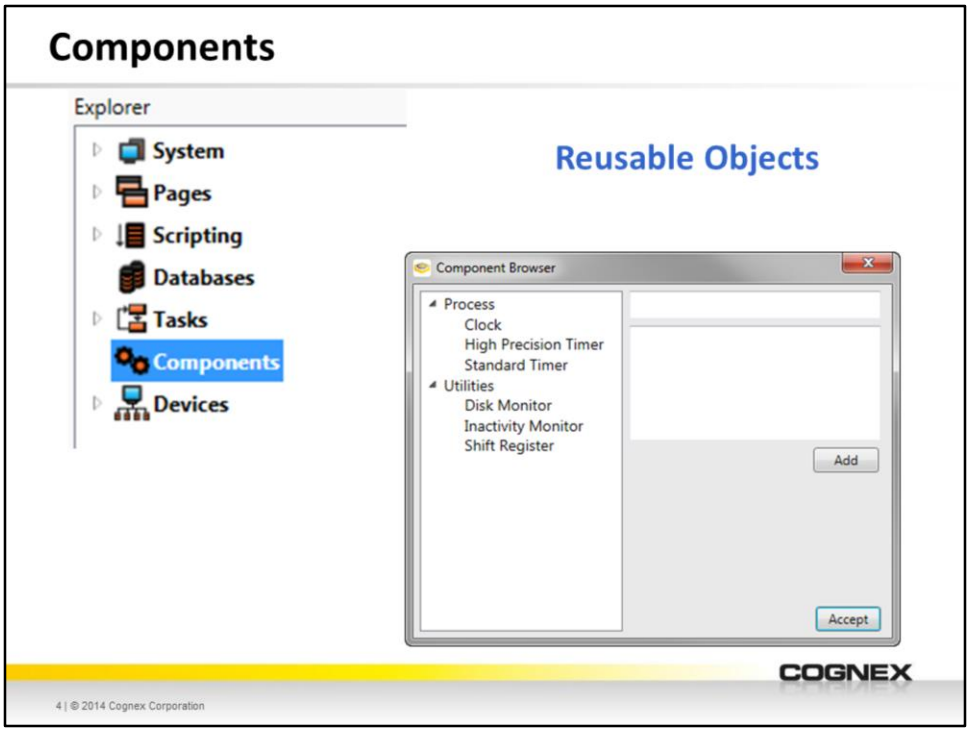
- Learn about the uses of Components
- Explore how a timer works
- Discover the use of shift registers

❖ Lab: Create the ability to continually acquire as well as display last 5 images on the HMI

COGNEX

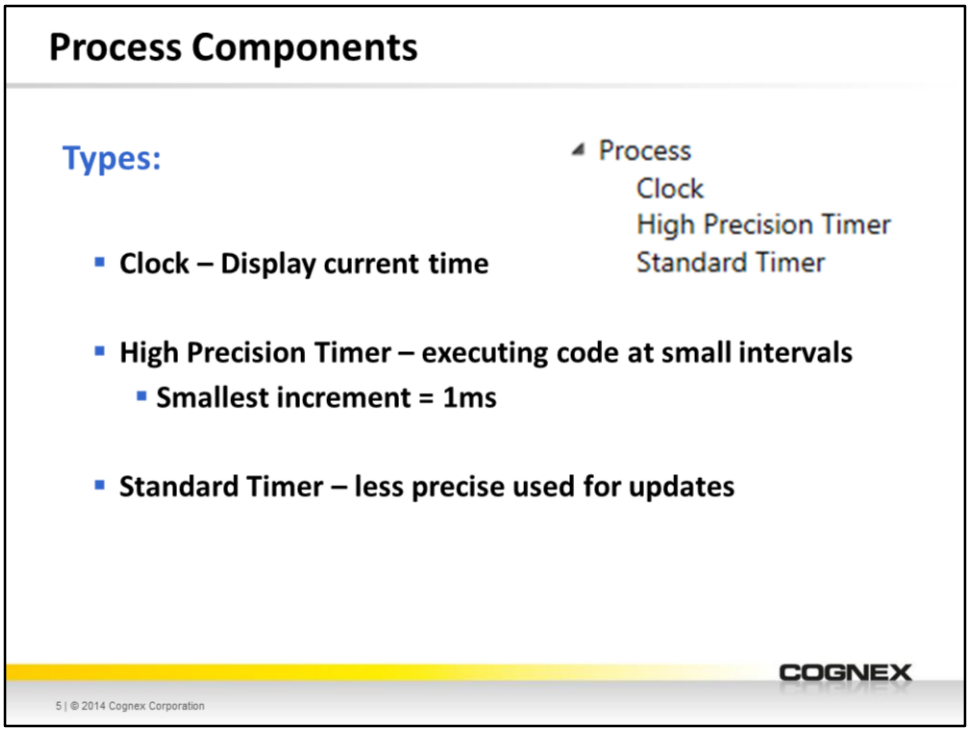


Cognex Designer gives user the ability to user timers, monitors, and shift registers through the use of Components.

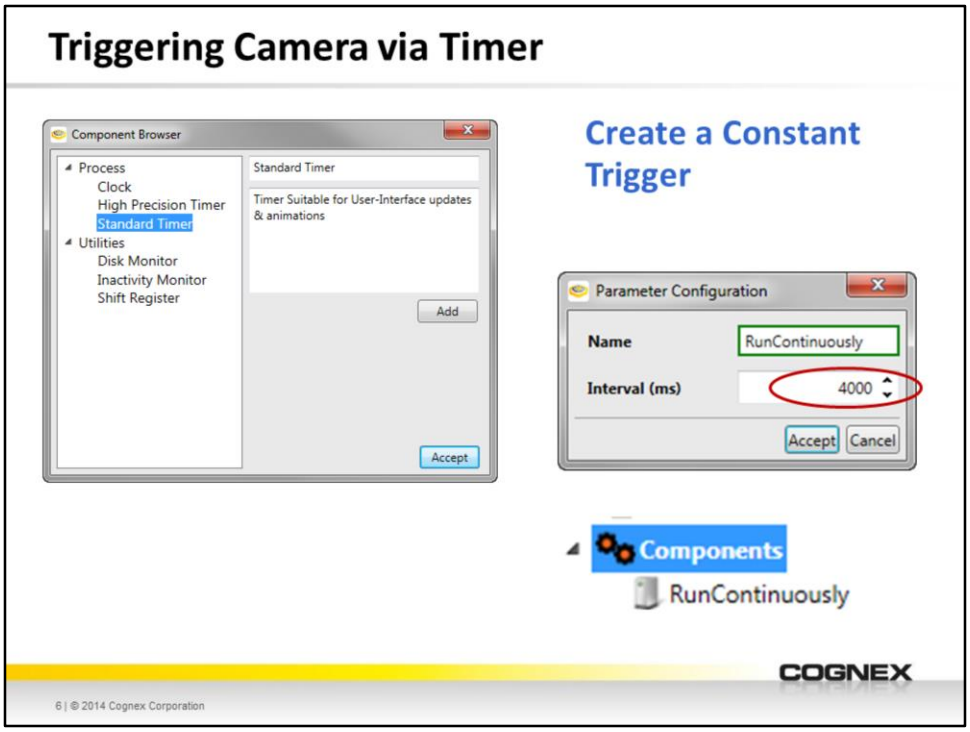


Components are reusable objects that provide specific application functionality. They are separated into Process Components and Utility Components.

Right-click on Components to configure a new component for the application.



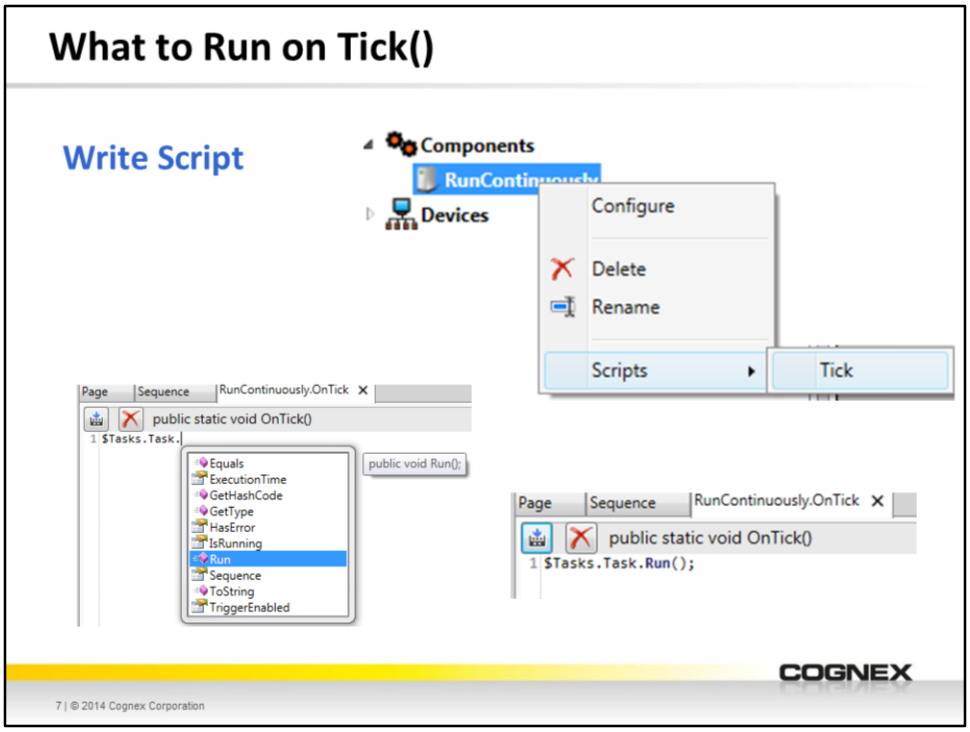
Clock – display current time. Allows user to input an offset of hour to account for time zone differences.
High Precision Timer - useful for executing code at very small intervals, with a higher degree of accuracy. It can be set to an interval of 1 millisecond.
Standard Timer - is most suitable for user-interface updates and other non-critical items.



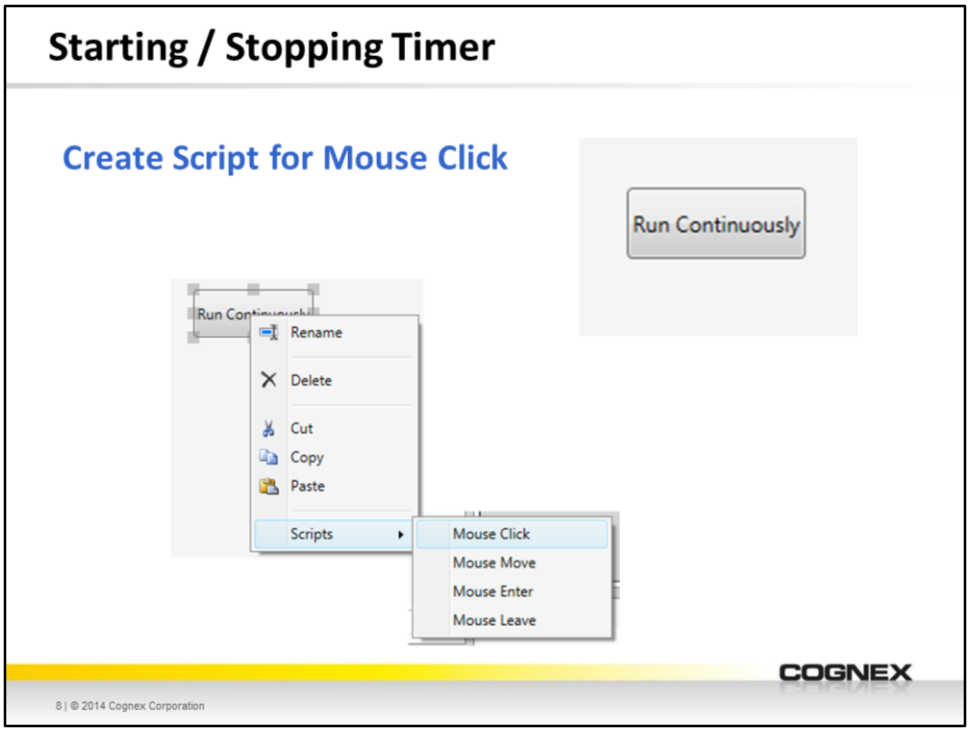
We can use a timer to constantly send triggers to the camera at a constant interval. This allows for a presentation mode or a consistent trigger within the application.

> **Note:** Make sure that the **Interval** is greater than the cycle time otherwise you will start missing images. We are choosing a large number in this case because the DS1000 acquisition can take some time.

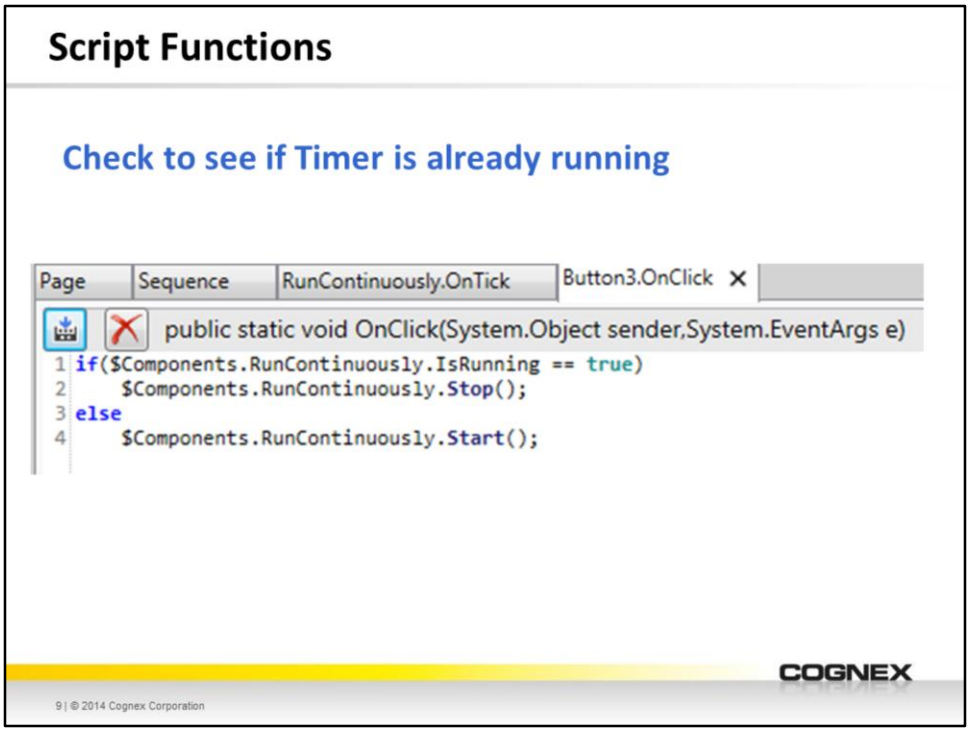
The new component will now appear in the Explorer tree under **Component**.



Script needs to be written to let the application know what function(s) it should run when the timer “ticks” to it’s next interval. In this case, we want our sequence to run so we add the function “$Tasks.Task.Run();” to the timer’s OnClick() sub-routine.

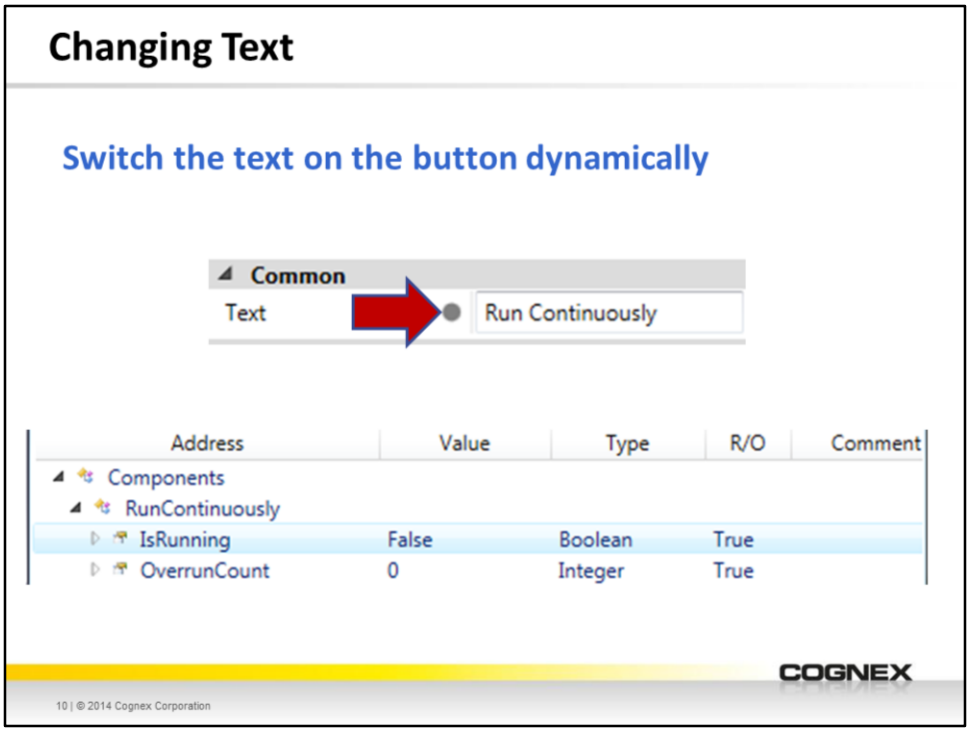


Now we need to start the timer as well as stop it. One way is to tie the functionality to a button.
We cannot use the property parameter **Click Command** in this case as we need to do a few things – not just a single command.
Right click of the Button and choose **Scripts -> Mouse Click** as we want to add a set of commands when we click on the button.

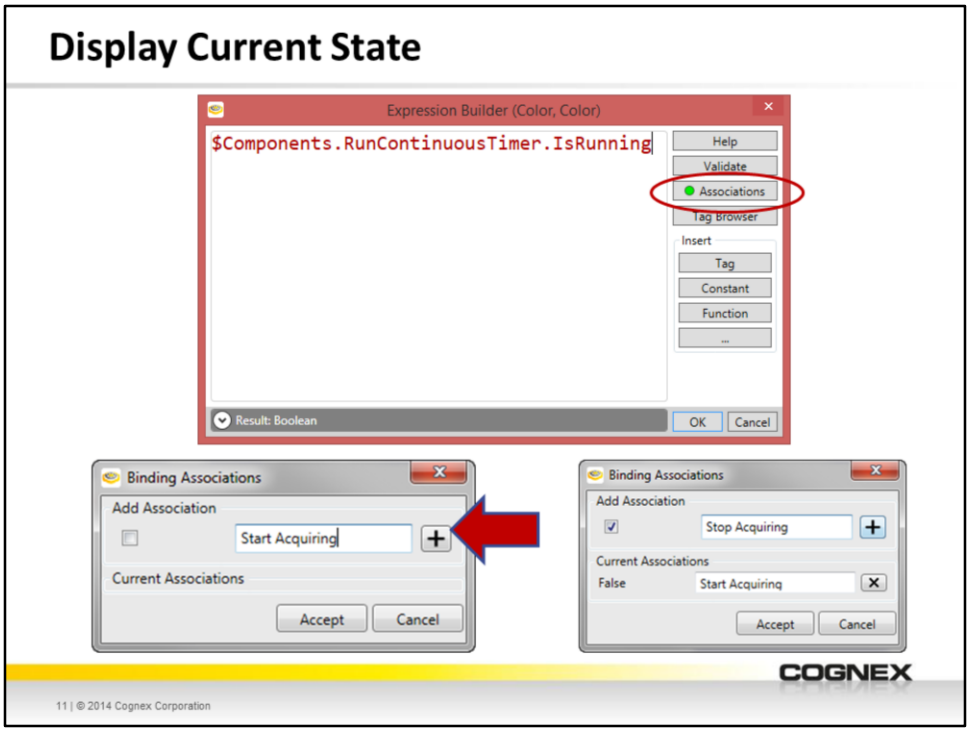


This script is checking to see if the timer is already running.

- If it is, clicking on this button will stop the timer.
- If it is not, clicking the button will start the timer.



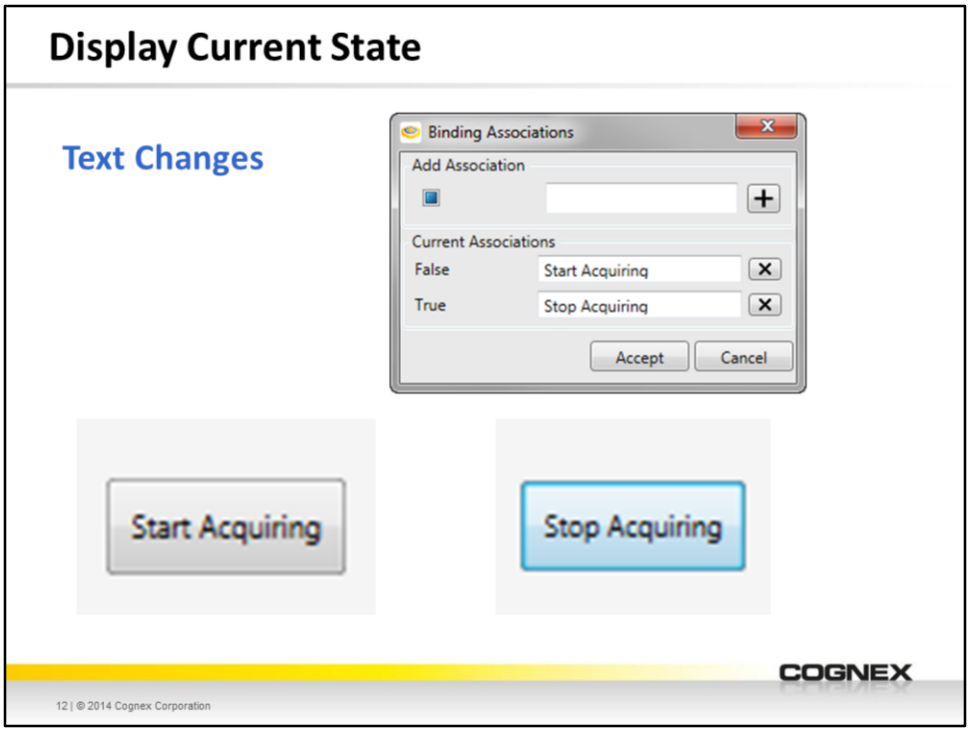
To switch the text dynamically in the button, we can add the Tag for $Components.RunContinuously.IsRunning to the Text parameter of the button. This will change the text depending on whether the timer is running or not.



Associating needs to be added to let the system know what should be shown depending on whether the timer is running or not.

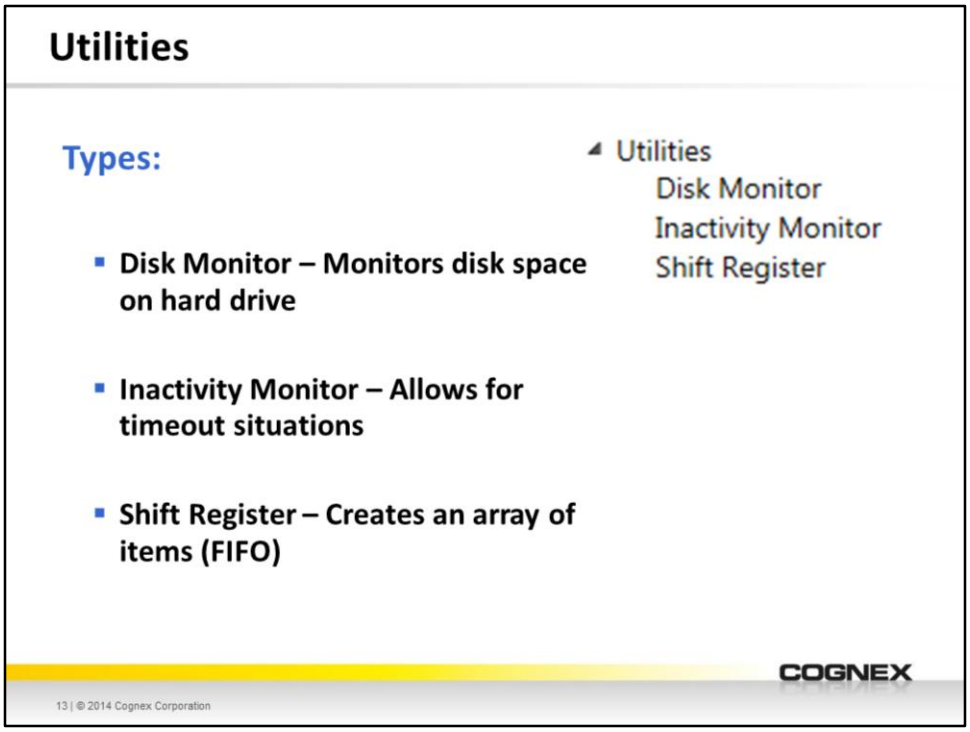
If the timer is not running, then the text “Start Acquiring” will appear to let the user know to push it to start the sequence.
If the timer is running, then the text “Stop Acquiring” will appear to let the user know to push the button to stop the automatic acquisition.

Note: Make sure you press the “+” for the second entry so that both states show under the associations.



Now it is set-up that if the timer is running, the button will say “Stop Acquiring”. If the timer is not running, then the button will say to “Start Acquiring”.

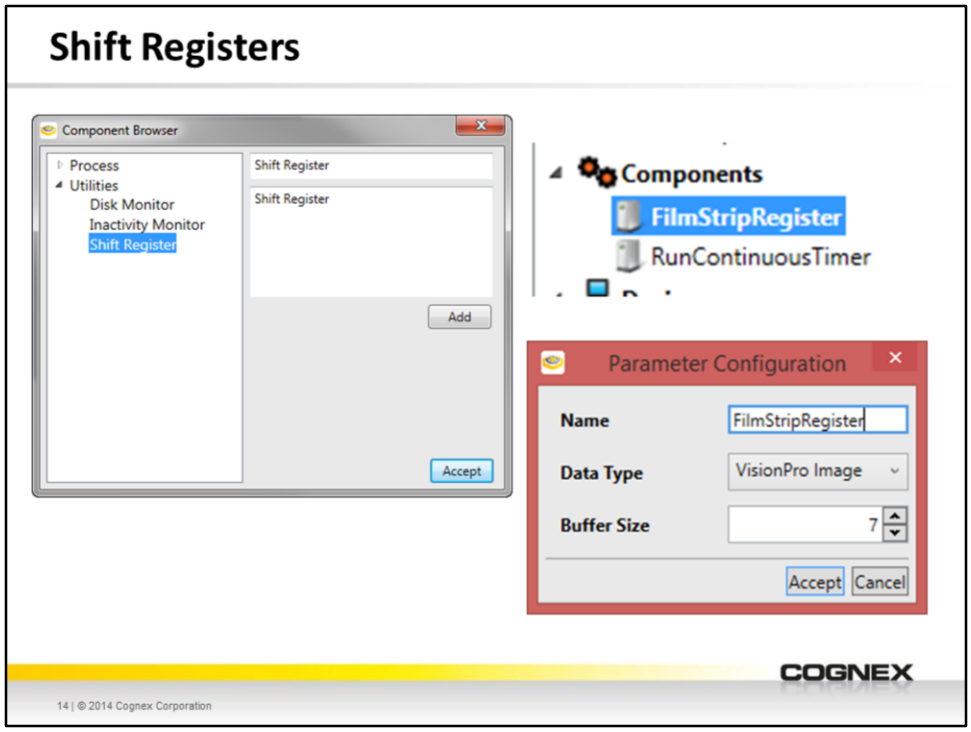
This is not very practical in a our application as we have to move the gantry each time. In the lab, we will add a signal to let us know to start moving again so that it appears to be constant.



Disk Monitor – monitors disk space on hard drive. Can be configured to return total capacity, amount used / available, percent used / available. Interval of minutes can be selected to state how often to check.

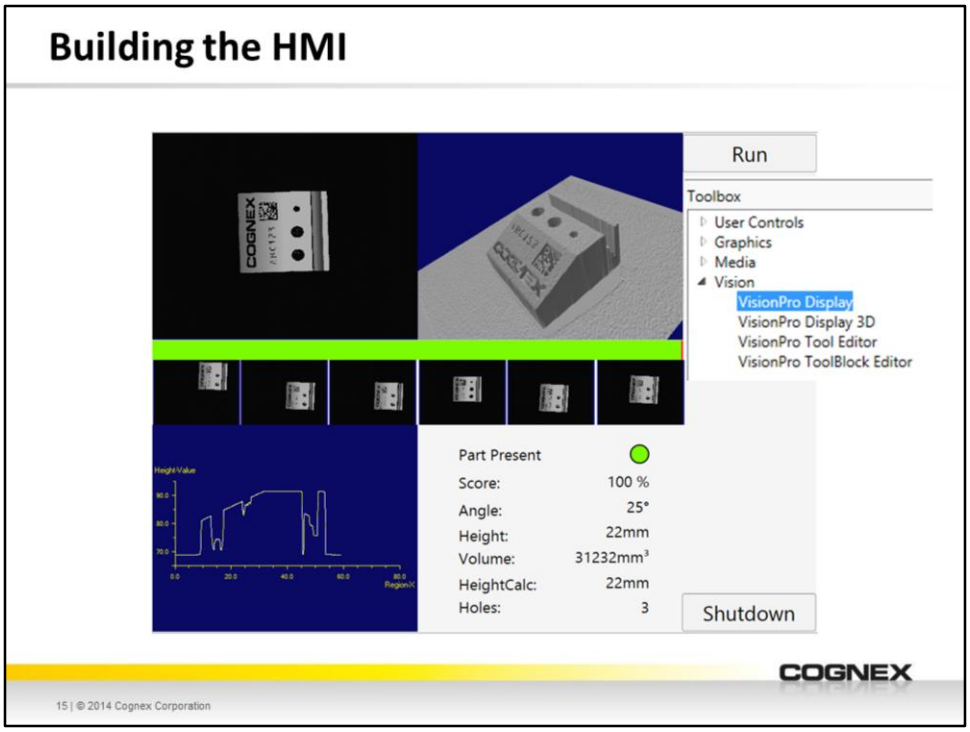
Inactivity Monitor – monitors user input and calculates the length of the inactivity time. The monitor provides a configurable timeout script that can be executed when the inactivity time exceeds the specified amount.

Shift Register - provides an array of tags of a particular data type. When a new item is added to the shift register, previous items in the register are shifted in position. Number of items is inputted when created. The Add() function is used to add the new data type to the existing array.



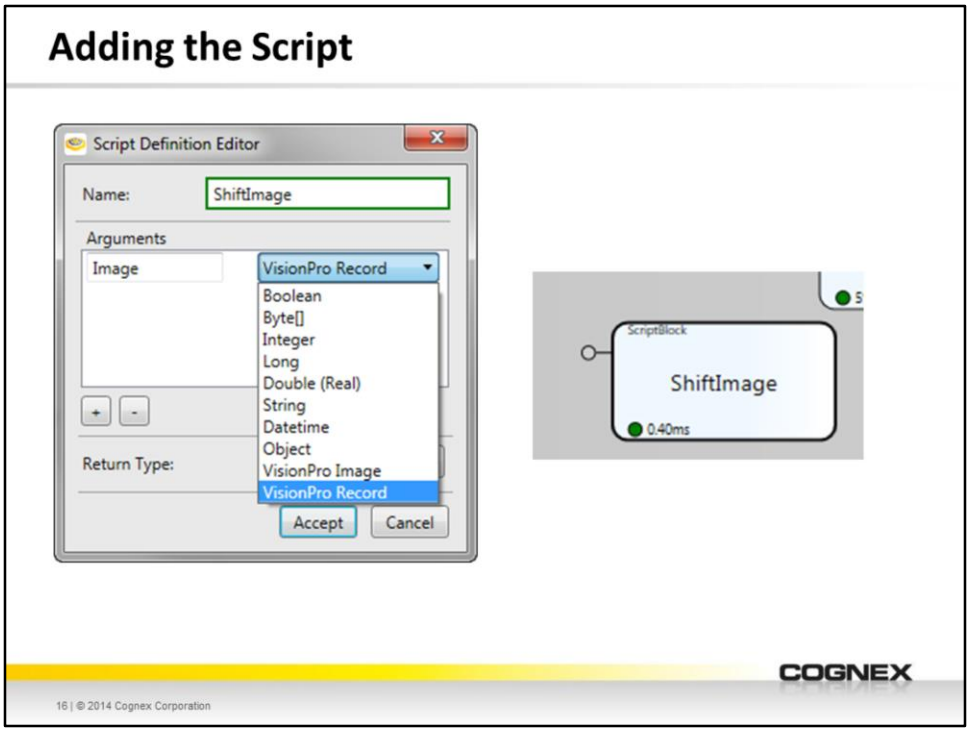
We are going to create a filmstrip at the bottom of our HMI that will display the last 5 images acquired by the camera. The Data Type being used is VisionPro Image.

We could make it easy and just grab the 2D image though that may not help us with troubleshooting the application from the HMI.

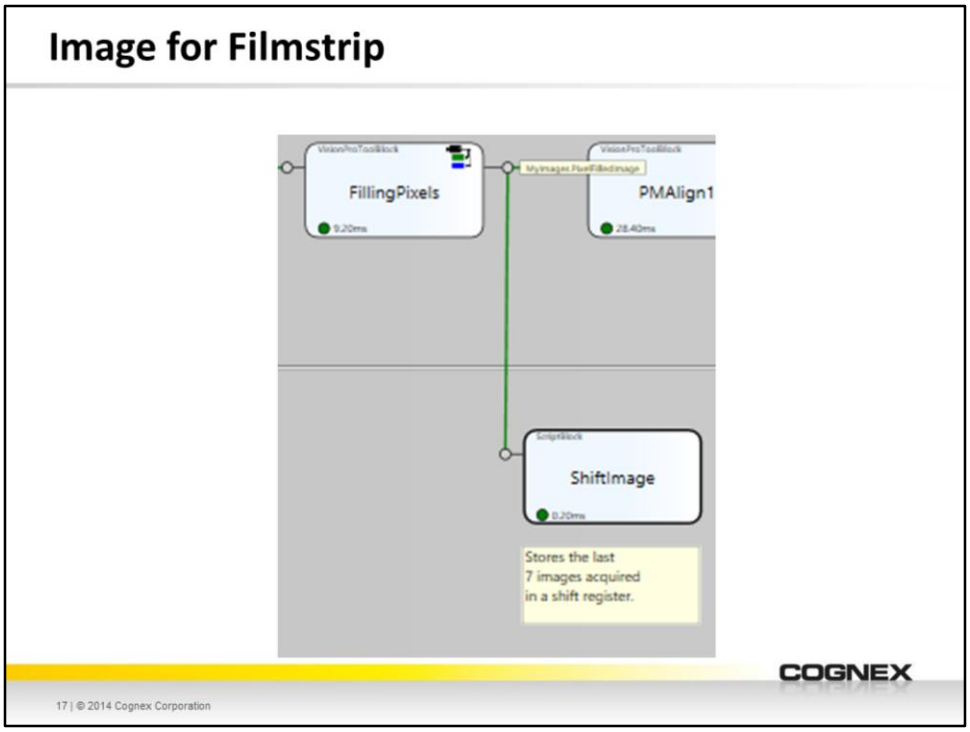


Add 6 VisionPro displays.

Note: Use the Arrange buttons on the taskbar at the top of Cognex Designer to line up each display, distribute horizontally, and other options to perfect the position on the HMI

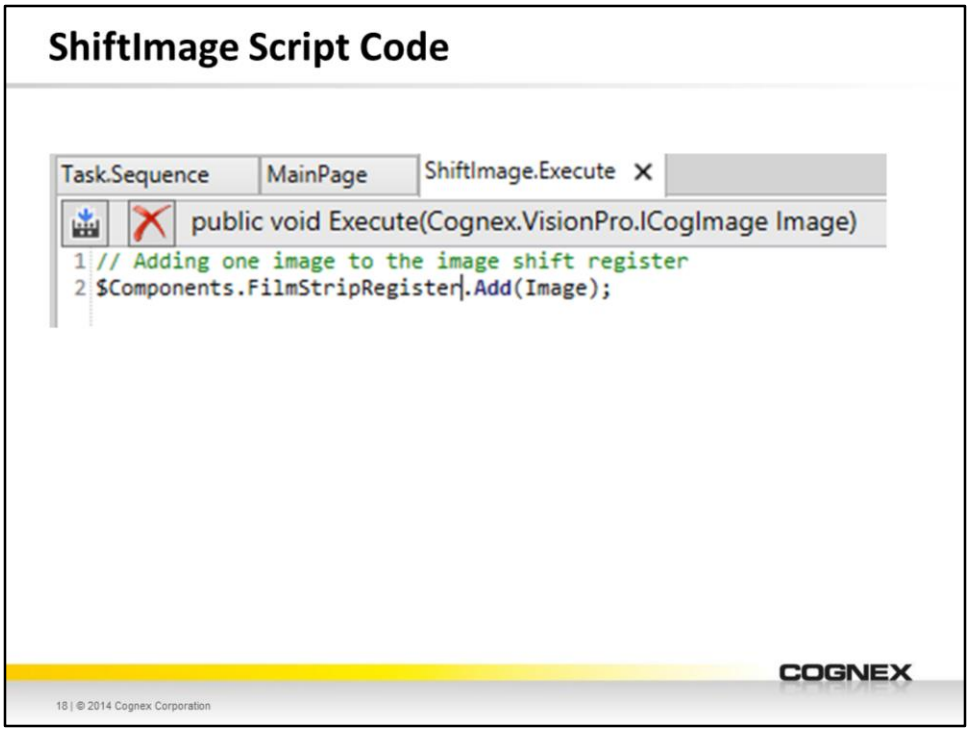


Create a new ScriptBlock in the Sequence to add the images to the shift register. The input argument would be VisionPro.ICogImage and the Return Type would be None. The creation of the array will be happening in the Script so no additional output is needed.



By attaching it to the PixelFilledImage, we will get the same clean image the other displays are seeing.

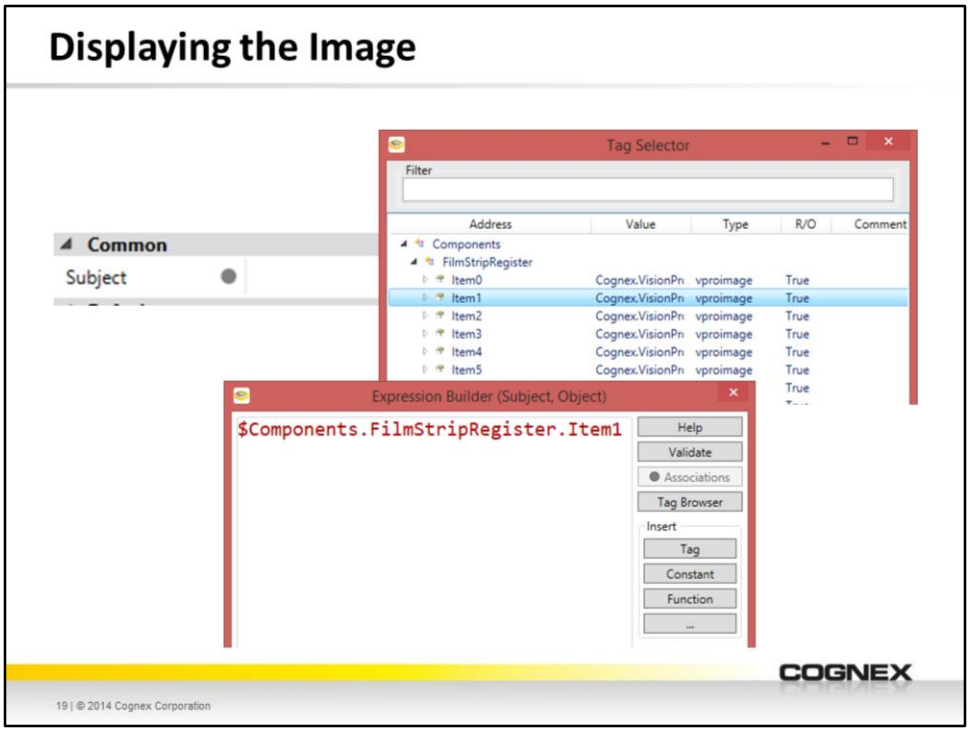
This might assist in trying to troubleshoot potential issues as we see the images displayed in the filmstrip.



Simple and clean.

Every time we run the sequence, we add a PixelFilledImage to the FilmstripRegister.

It operates on a first in, first out (FIFO) method.



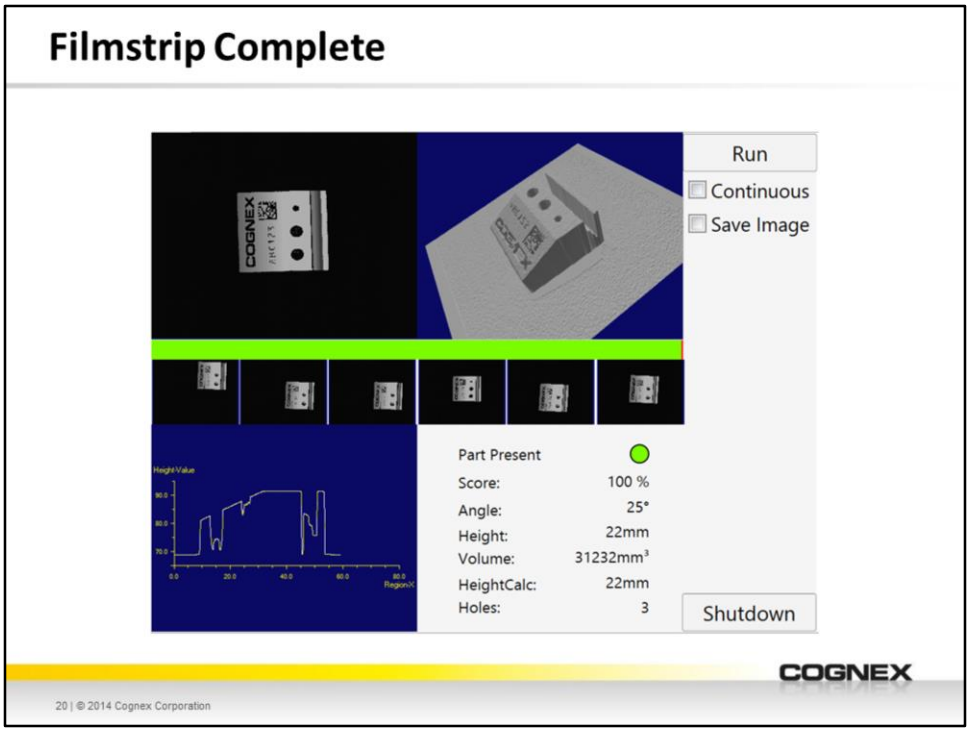
The next step is to assign the image to each of the displays.

Starting with the one on the left, go to the **Subject** and use the **Tag Browser** to reference the shift register's 2nd item which is Item1 (zero-based).

The reason we go to the 2nd item is because we are already be displaying the 1st item (Item0) in the large original display.

> **Note:** VisionPro (as is most Cognex products) is zero based which means the first item is referenced as Item0. The second would be Item1 and so forth.

Do the same in the next display except choose Item2. Continue do this for each of the remaining displays so that the right most one is referencing the last available image in the shift register.



Start the application and note the images that are displayed in the filmstrip each time that an acquisition (or rather Execute Sequence) happens which could be accomplished through our Run Once button or through our timer button.

Summary

Components

- Explored the uses of Process and Utility Components
- Learned about using a timer to generate a trigger
- Discovered the method of creating a filmstrip via implementing a shift register

COGNEX