Screen Navigation

Part	Version	Revision	Date	Status
en	5.1.1183.81	001	2021-03-17	Released

Content

FAQ COMBIVIS Studio HMI

Introduction

This document explains how to perform basic screen navigation in a Studio HMI project. The following sections will describe the tasks of (1) changing screens, (2) using popup screens, (3) using embedded screens, and (4) setting a startup screen.

Changing Screens

This section explains how to change the active screen with a command on click action.

1. Add Clickable Element to Screen

Add a clickable element to the root screen (“MainMenu” in this example). This example uses an XAML button. Add a descriptive label, “Open Screen” in this case, to help guide the user.

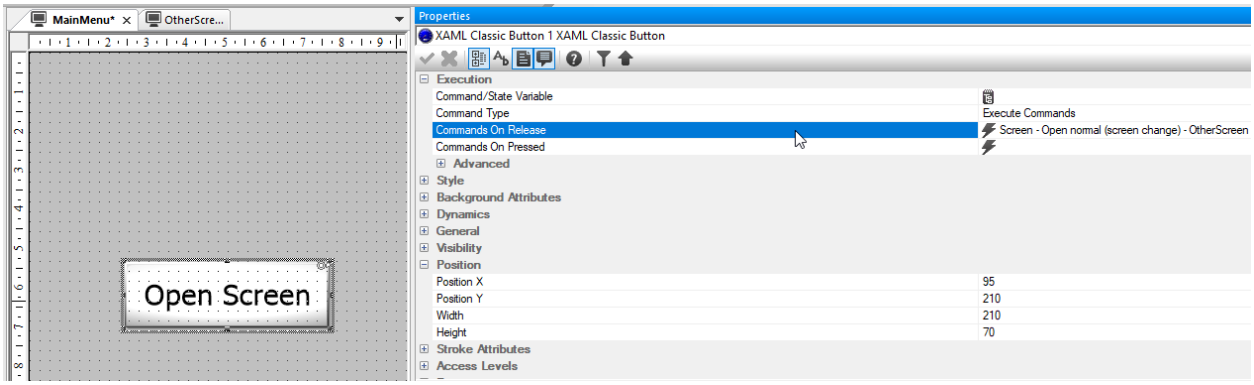

Fig. 1: XAML Button for Screen Change Command

In the *Execution* properties of the element, ensure that the *Command Type* is set to “Execute Commands”.

2. Define a Screen Change Command

Next, select either the *Commands on Release* or *Commands on Pressed* field and hit the ". . . " icon.

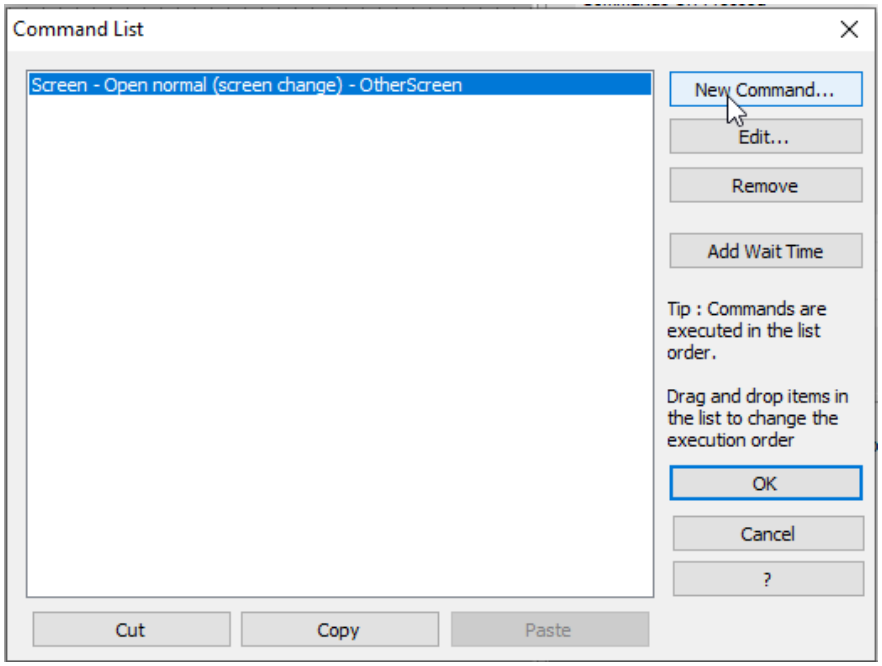

Fig. 2: Command List

A screen will be displayed with a command list that will be executed either upon the pressing or releasing of the element. Select *New Command. . .* and locate the *Screen* command type. The default *Action* of *Open normal (screen change)* will be used. Use the ". . . " icon in the Screen field to browse for the screen you would like to open. Select it, then select *OK* until the command is added.

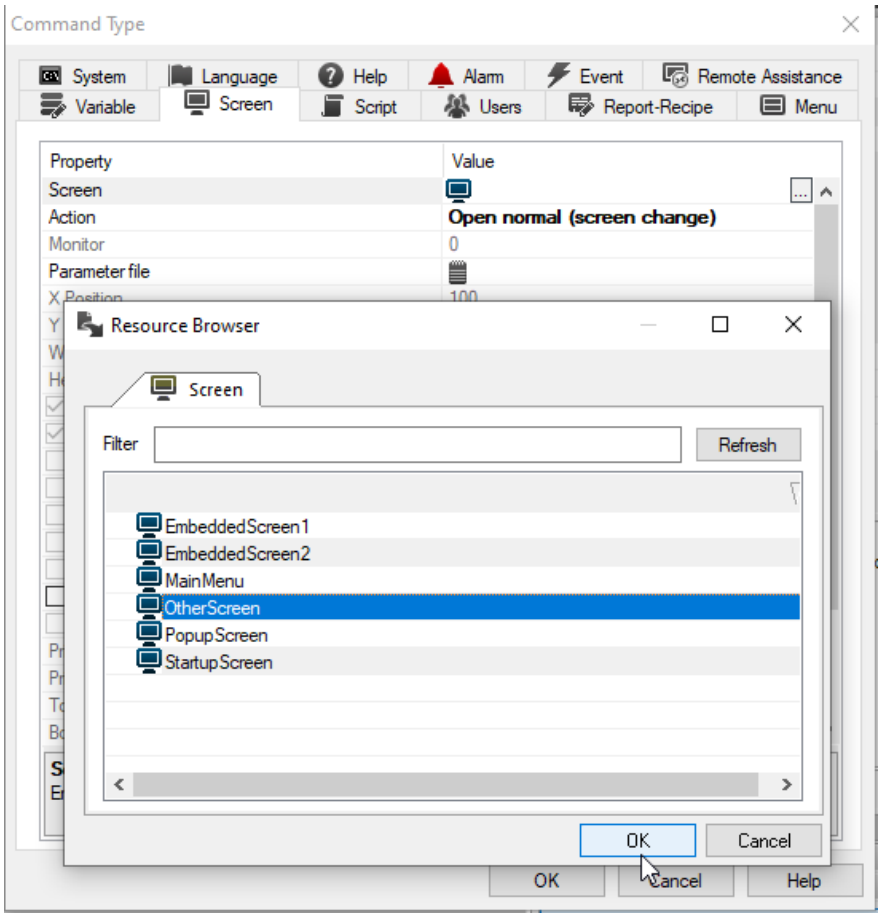

Fig. 3: Screen Change Command

3. Create a Return Command

You must configure a method to escape or return from a screen once it is opened. Otherwise, you may navigate to a screeen that you cannot return from. This can be done in many ways, the simplest being to create another button with a command to open the root screen that brought you to this other page. The button in this example is labeled “BACK”.

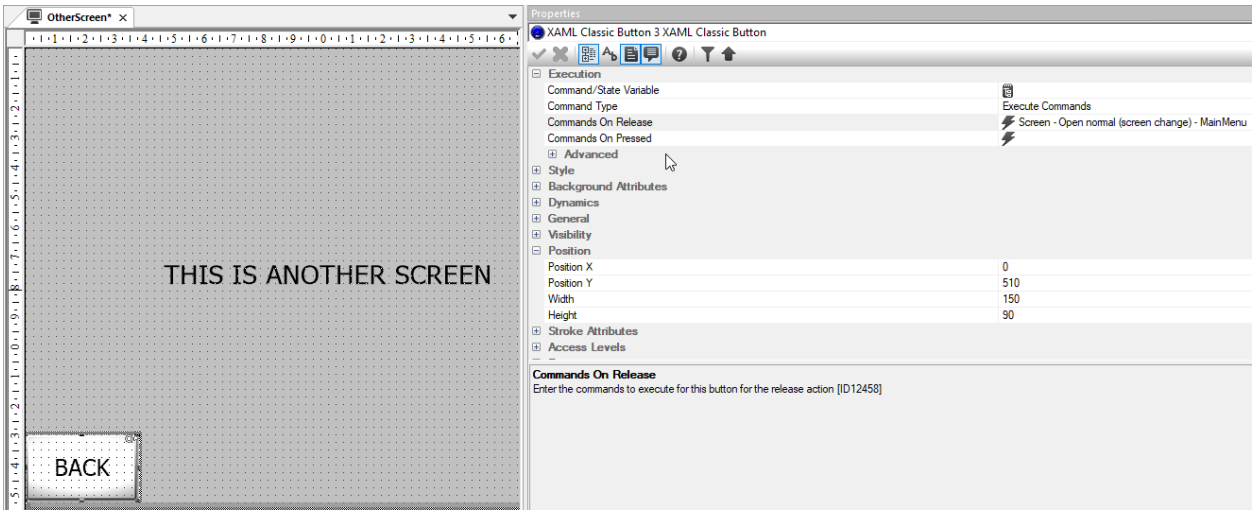

Fig. 4: Back Button with Command

Popup Screens

This section explains how to open and close popup screens on command. Popup screens are displayed in front of, and disable input to, the root screen until the popup screen is closed. These screens are often used to display important messages or prompt the user for input without fully changing screens.

1. Create a popup screen

Popup screens are typically smaller than the default screen size of the project. Create a new, smaller screen. Include a clickable element in the screen.

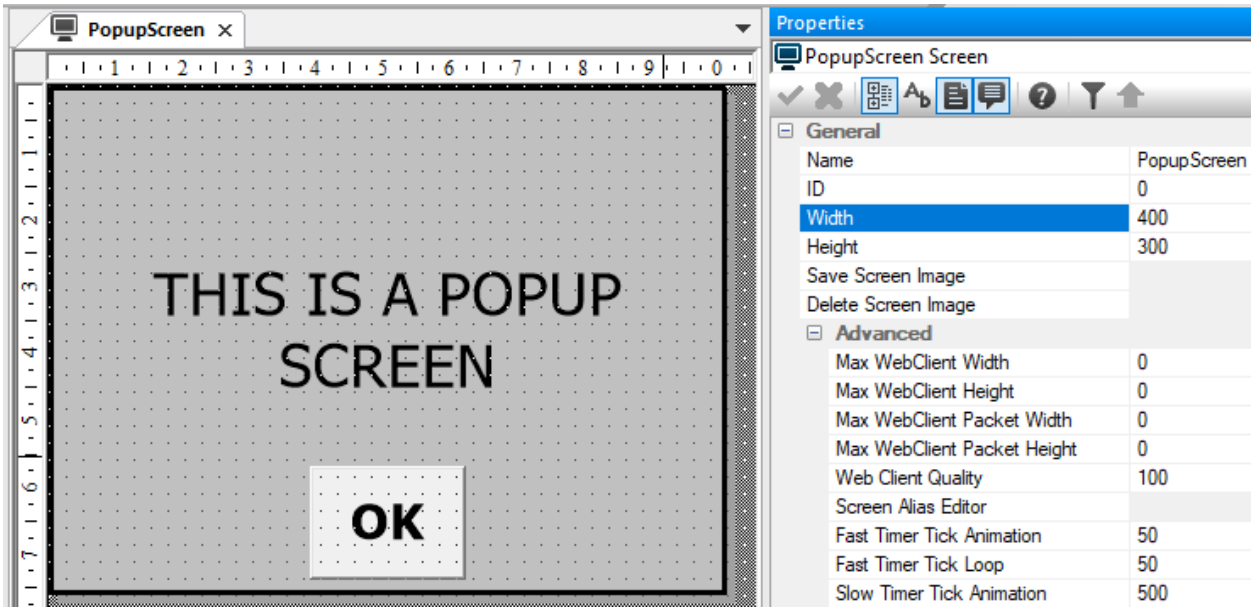

Fig. 5: Popup Screen with Button

2. Create a Close Command

Define a screen command for the clickable element in the same manner as the previous section. However, this time we will leave the *Screen* field empty and chose “Close and Return Back” for the *Action*.

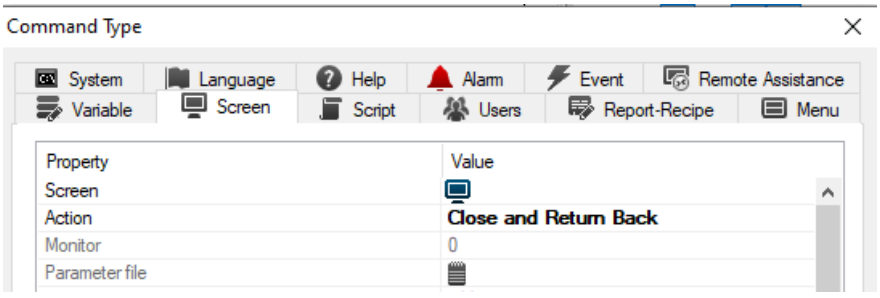

Fig. 6: Close Popup Screen Command

This action closes the popup screen and and returns the user to the screen running in the background.

3. Create a Popup Open Command

On the root screen, add a clickable element.

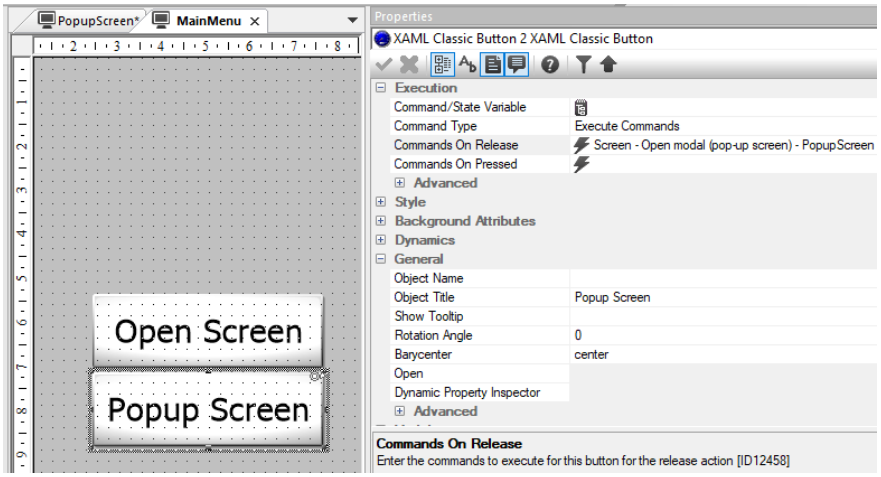

Fig. 7: Open Popup Screen Button

Create another screen command. Browse for and add your newly created popup screen in the *Screen* field. For *Action* type, select “Open modal (pop-up screen)”. You can also define where on the screen the popup screen will appear. The *X Position* and *Y Position* fields define the pixel distance from the top-left origin of the HMI screen to the top-left corner of the popup screen.

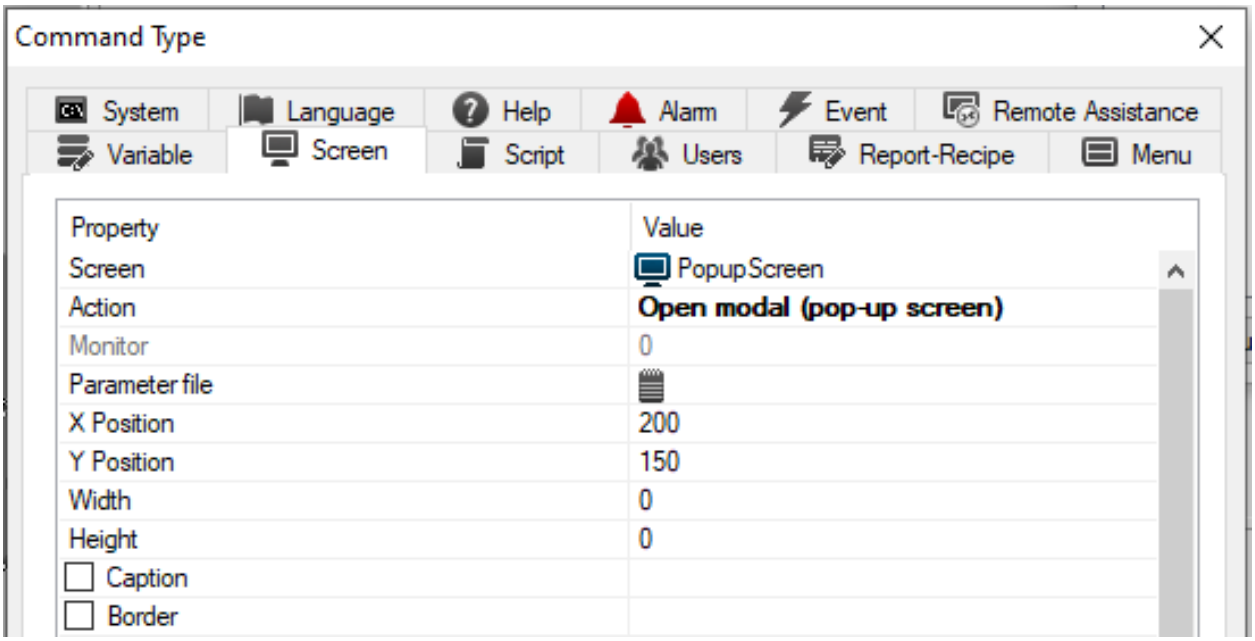

Fig. 8: Open Modal Command Properties

In addition, you can choose to keep or hide the screen border and caption which includes the screen name. In the example above, the screen caption and border have been turned off. In addition, the X Position, Y Position of 200, 150 have been chosen to center the popup screen of size Width = 400 and Height = 300 pixels on a screen of Width = 800 and Height = 600 pixels.

4. Open a Popup on an Event

Popup screens, as well as regular screens, may also be opened upon an event occuring rather than on user click. To accomplish this, define an event that will occur in your project's operation. See the example below where a local variable was created named "EventVar". This variable's value can be edited by the EditBox element on the root screen. The event in this example is set to trigger when EventVar = 5.

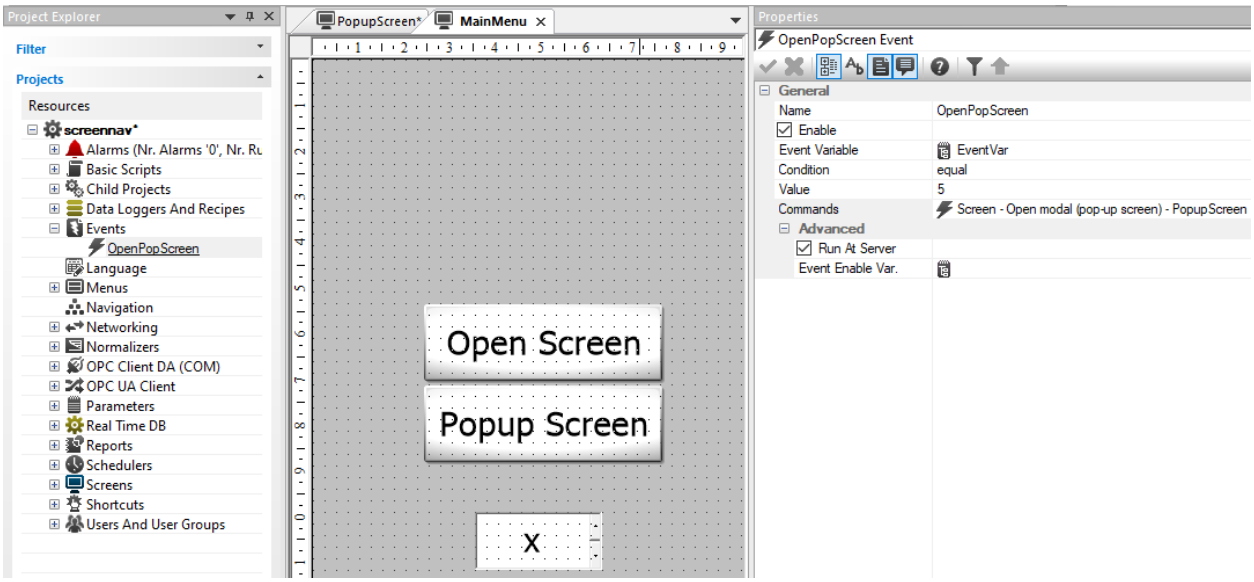

Fig. 9: Popup Triggered by Event Configuration

The *Commands* field of the event can then be configured identically to the previous section to open the popup when the event occurs.

Embedded Screens

This section describes how to configure basic embedded screens. Embedded screens are similar to popup screens in that they are typically smaller than the default screen size of a project. However, embedded screens exist on the same layer as the root screen and are always visible. Embedded screens are ideal for things like screen navigation menus and screen headers that are re-used on multiple screens. Embedded screens are also very useful because they can be created once, but used on any number of screens. Refer to the end of this section for an example of embedded screen usage.

1. Create an Embedded Screen

Create a screen that is smaller than the default screen size of the project.

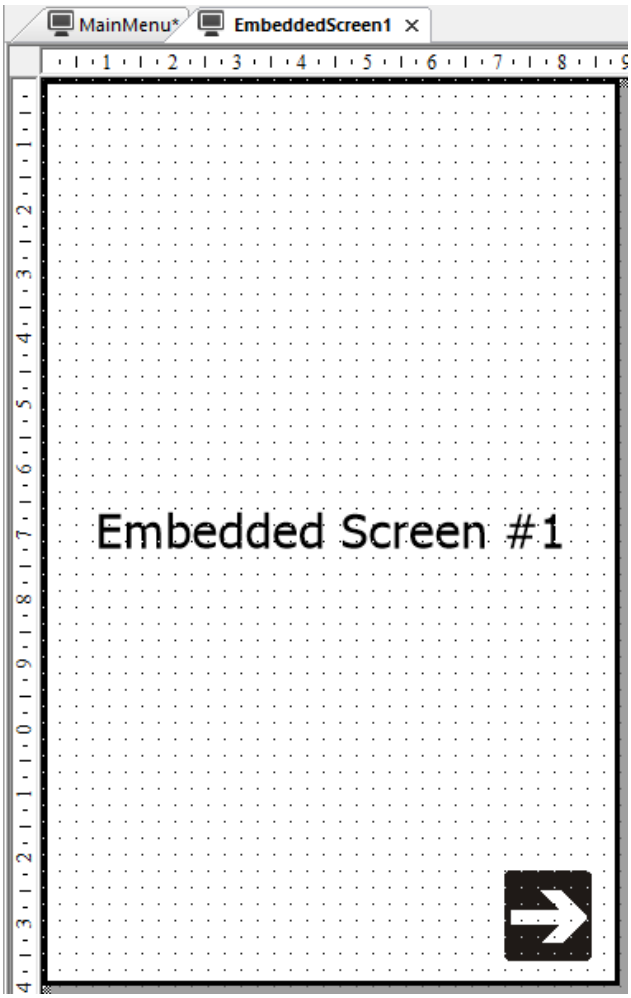

Fig. 10: Example Embedded Screen

2. Create an Embedded View

Insert an *Embedded View* element into the screen where the embedded screen is to be displayed. This element can be found in the Studio HMI Toolbox under *Advanced Objects* > *Embedded View*. Make the embedded view the same size as the embedded screen to avoid distortion of elements. Assign the embedded screen to the embedded view in the *Style* properties.

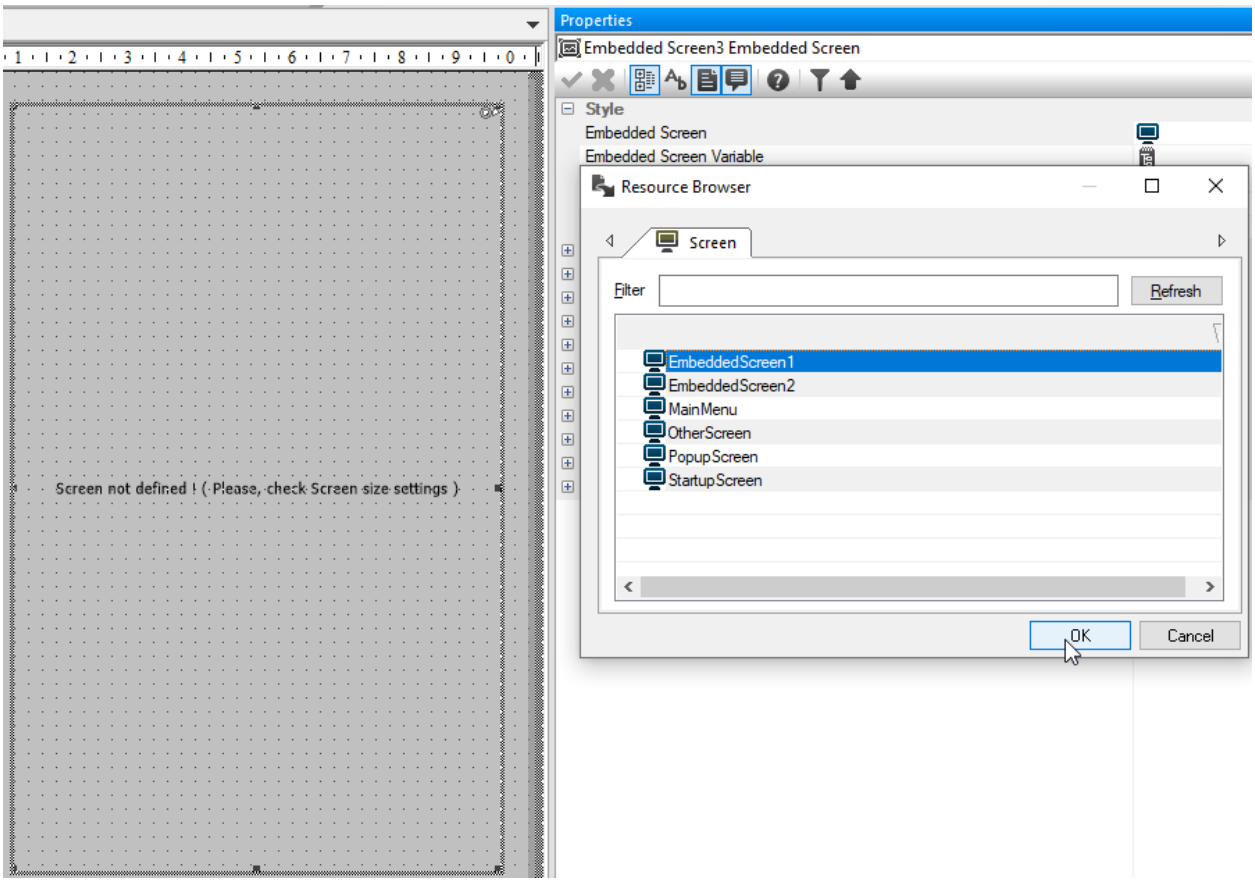

Fig. 11: Configure Embedded View

The embedded screen can now be view and interacted with through the embedded view.

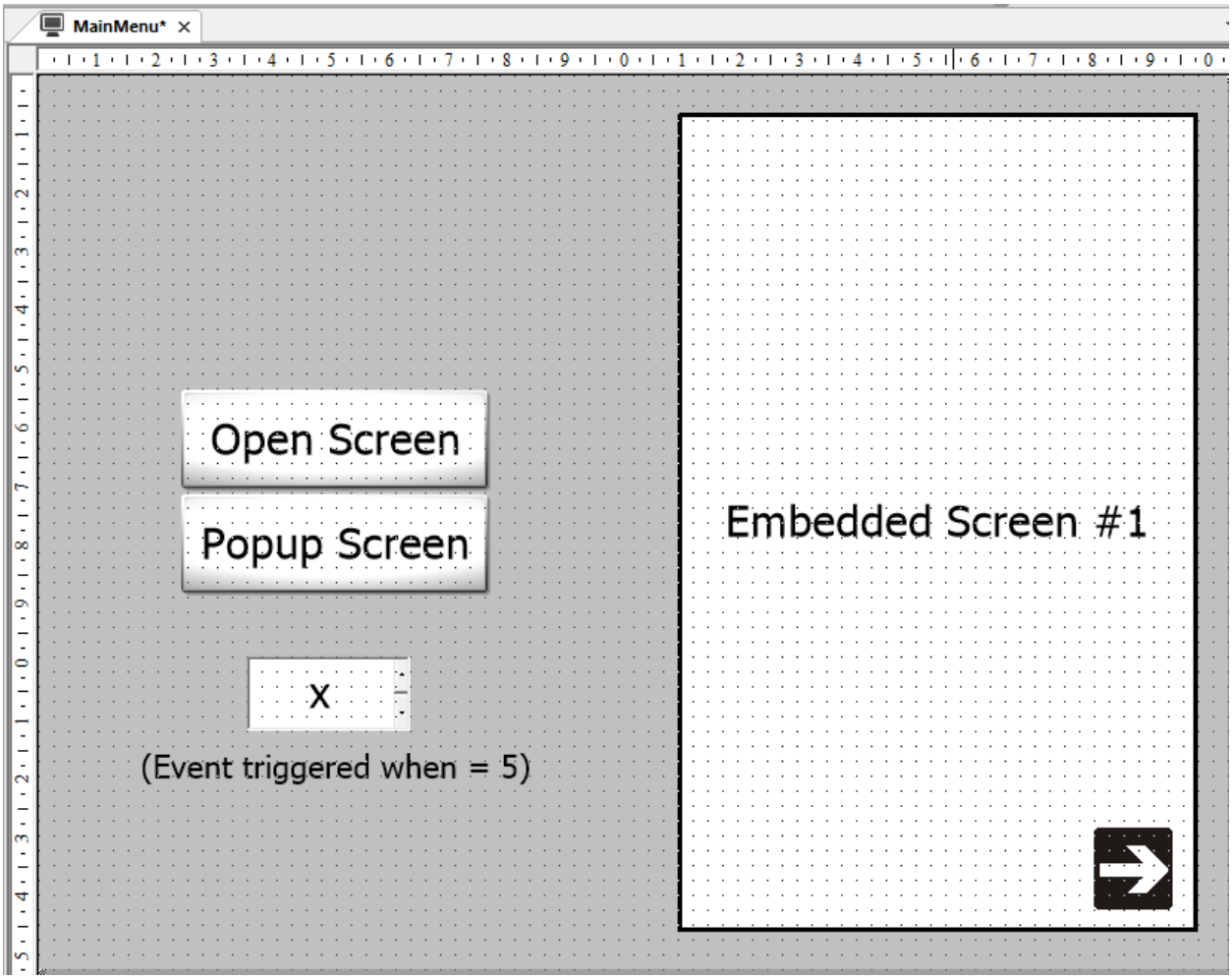

Fig. 12: Embedded Screen now Visible Within Embedded View

3. Make the Embedded Screen Dynamic

If you would like to have a single embedded view display multiple embedded screens, you can assign an *Embedded Screen Variable* in the *Style* properties of the embedded view. This variable will be of type string. To display an embedded screen in the embedded screen view, assign the name of the screen to the embedded screen variable.

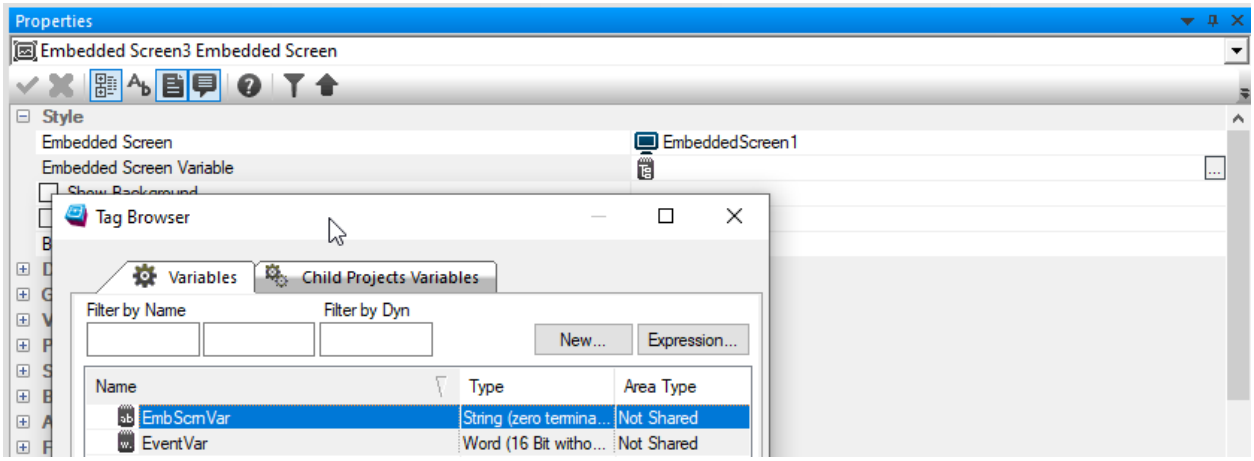

Fig. 13: Embedded Screen Variable

In this example, we have two screens named “EmbeddedScreen1” and “EmbeddedScreen2”. Each embedded screen has an arrow symbol element which when clicked, executes a variable command that sets the value to the name of the other screen.

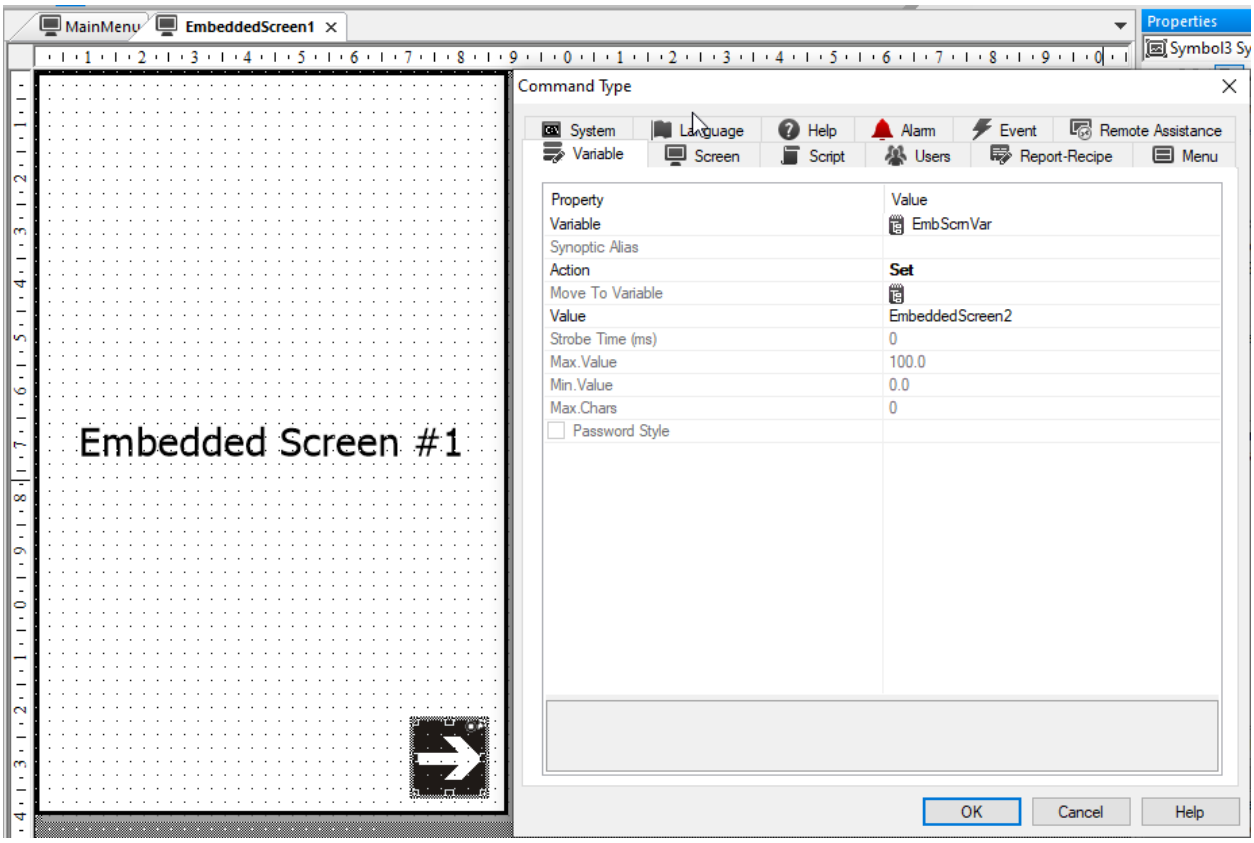

Fig. 14: Embedded Screen Variable Execution

A click of this element will now switch from EmbeddedScreen1 to EmbeddedScreen2.

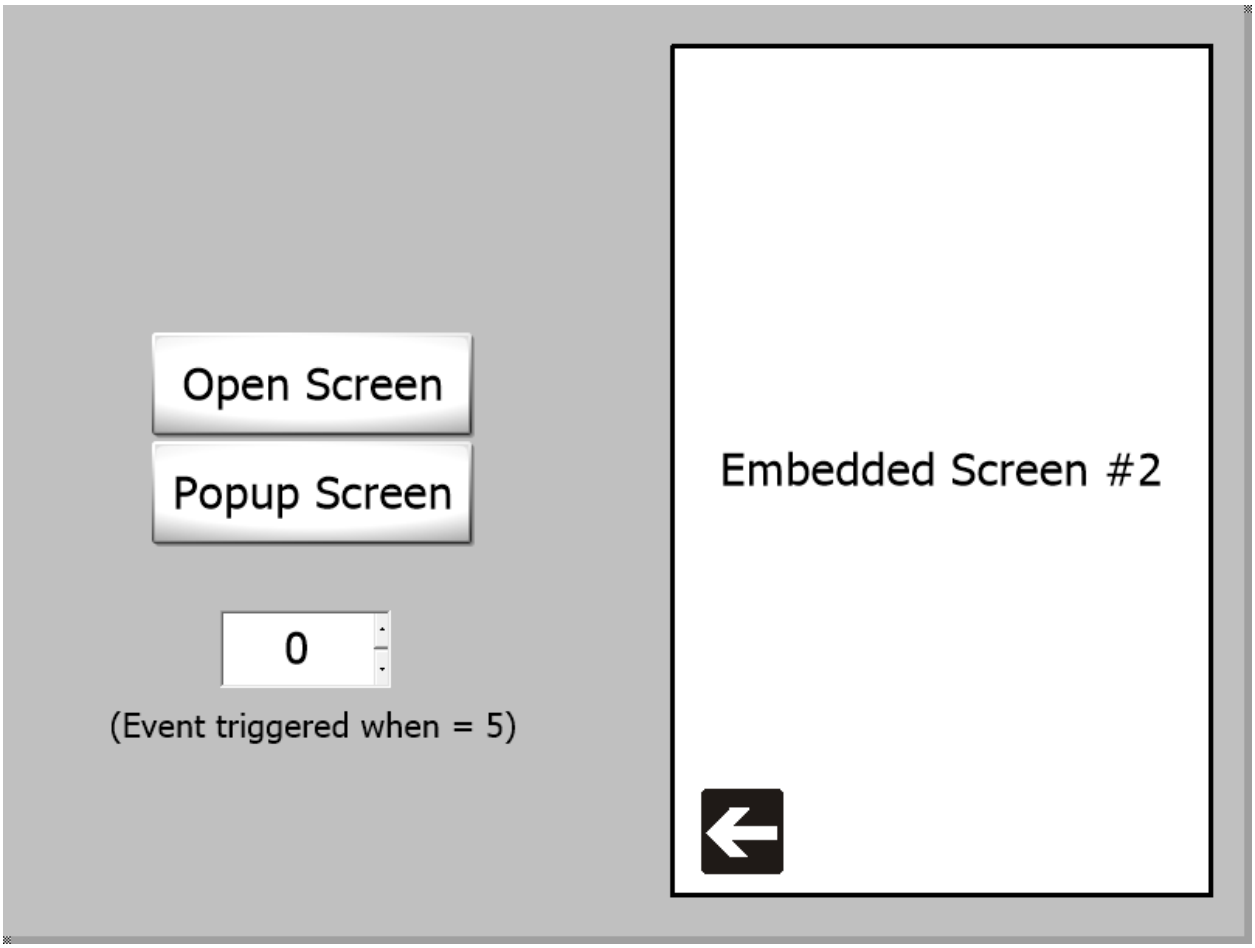

Fig. 15: New Embedded Screen Visible.

A similar variable command can then be created for EmbeddedScreen2.

4. Examples of Embedded Screen Usage

See the following example that includes two different embedded screens - one for the screen header (top), and one for screen navigation (left). The header is used to show status messages as well as date and time information on every screen in the project.

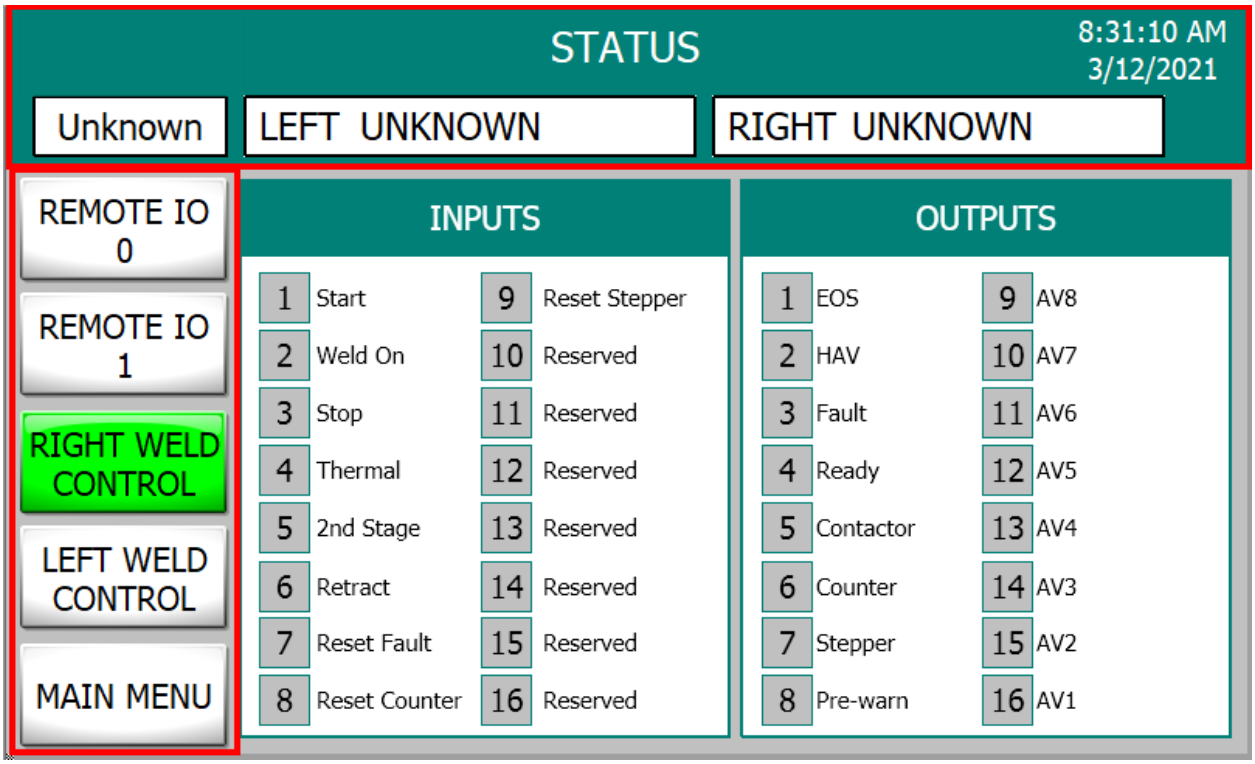

Fig. 16: Example of Embedded Screen Uses

Startup Screens

A startup screen must be defined in the HMI project. Any screen may be defined as a startup screen. Sometimes, however, it is desirable to create a dedicated startup screen with the machine manufacturer's logo and contact information for the operator's reference. To define a normal operation screen as the startup screen, skip to step 3.

1. Create Startup Screen

Create a new screen. In that screen, include a clickable element. In this example, we will use a simple rectangle that is the size of the entire screen and is brought to the front of the screen. This way, a click by the operator anywhere on the screen will take them to the next screen.

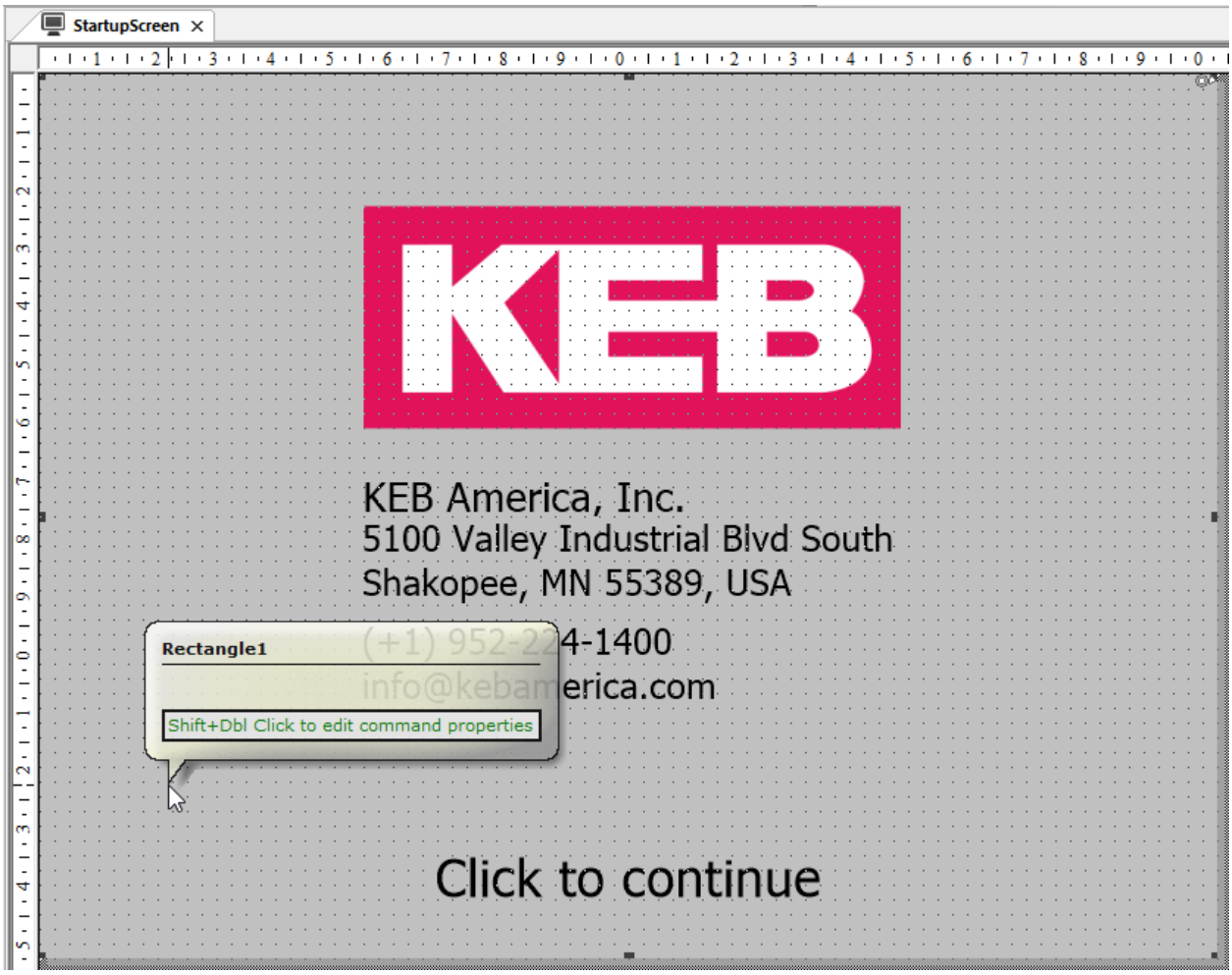

Fig. 17: Startup Screen with Clickable Element

2. Define a Click Command

Define a command for a clickable element in the *Execution* > *Advanced* > *Command On Click* field to bring the user to a home page from the startup screen.

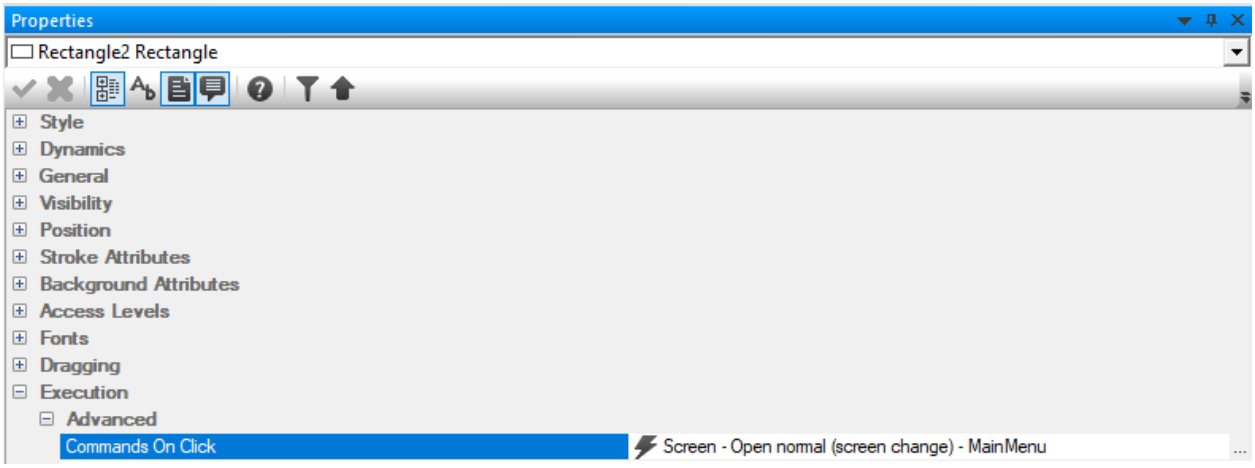

Fig. 18: Startup Screen Click Command

3. Set the Startup Screen

In the *Project* properties, locate the *Startup Screen* setting in the *Execution* properties. Browse for the screen that you would like to be your startup screen and select it.

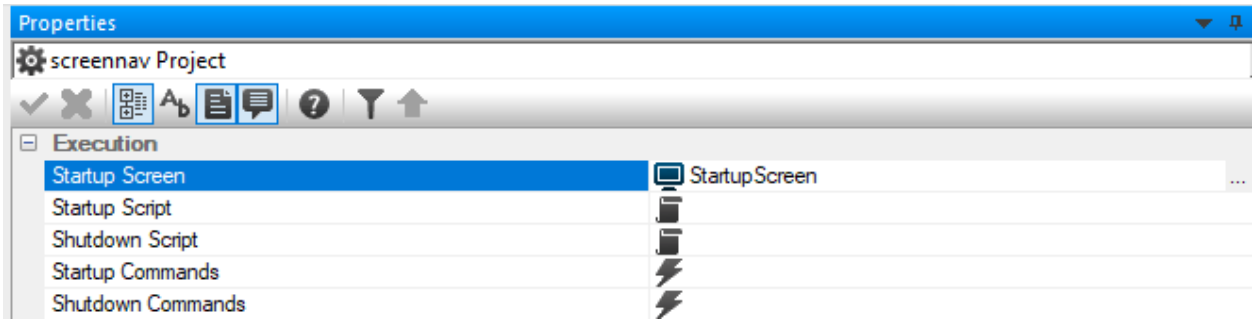

Fig. 19: Startup Screen Project Setting

Disclaimer

KEB America, Inc. reserves the right to change/adapt specifications and technical data without prior notification. The safety and warning reference specified in this manual is not exhaustive. Although the manual and the information contained in it is made with care, KEB does not accept responsibility for misprint or other errors or resulting damages. The marks and product names are trademarks or registered trademarks of the respective title owners.

The information contained in the technical documentation, as well as any user-specific advice in verbal or in written form are made to the best of our knowledge and information about the application. However, they are considered for information only without responsibility. This also applies to any violation of industrial property rights of a third-party.

Inspection of our units in view of their suitability for the intended use must be done generally by the user. Inspections are particular necessary, if changes are executed, which serve for the further development or adaption of our products to the applications (hardware, software or download lists). Inspections must be repeated completely, even if only parts of hardware, software or download lists are modified.

Application and use of our units in the target products is outside of our control and therefore lies exclusively in the area of responsibility of the user.

Americas:

KEB America, Inc.
5100 Valley Industrial Blvd South
Shakopee, MN 55379, USA
(+1) 952-224-1400
info@kebamerica.com

Headquarters:

KEB Automation KG
Suedstrasse 38
D - 32683 Barntrup, Germany
(+49) 5263 401-0
info@keb.de