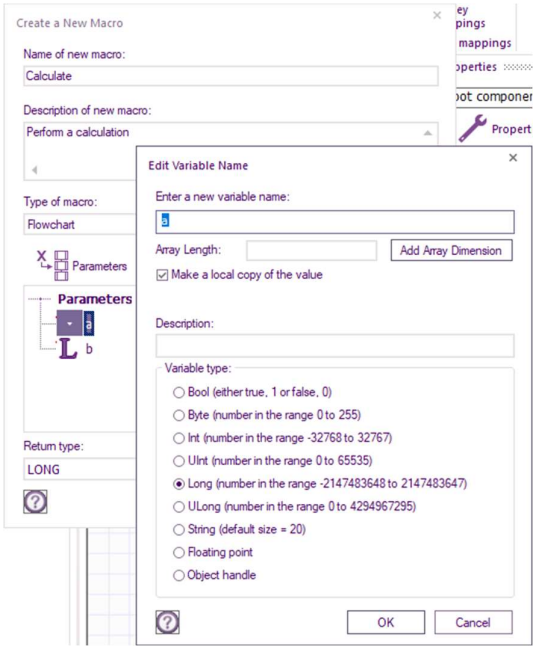
Step 1 – Create a new project

Select “New Project” and create a new embedded project. Any embedded target can be selected.

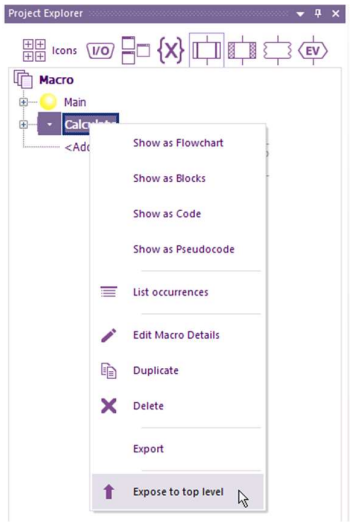
Step 2 – Create a new macro

Call the macro “Calculate” and give it two parameters (“a” and “b”, both of type “Long”). Set the return type to “Long” also.



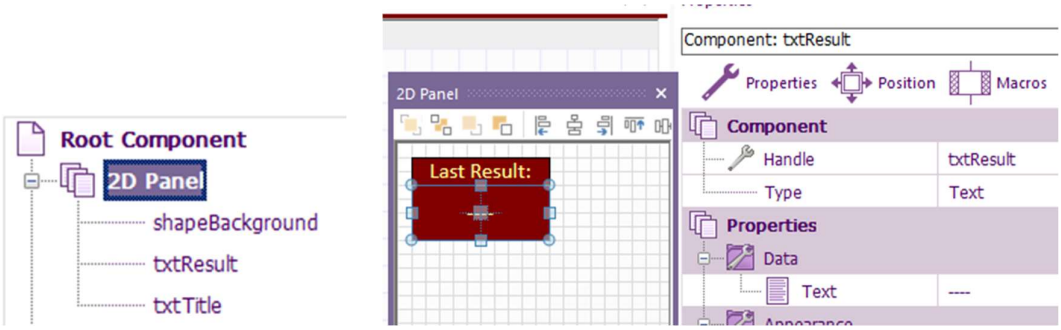
Step 3 – Expose the macro

Exposing a macro makes it available as a component macro when the component is created and used in other projects. In Project Explorer, right-click this macro and select “Expose to top level”.



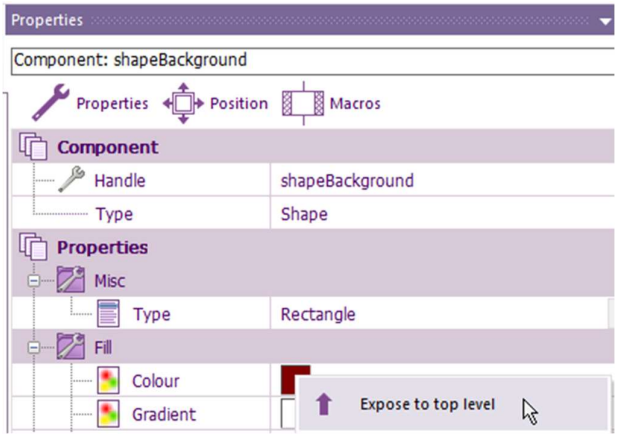
Step 4 – Create the visual of the component

On the 2D Panel, add primitive components from the “Creation” components list to create a visual representation of the component. For this example, add 2 Text primitives and 1 Shape primitive:



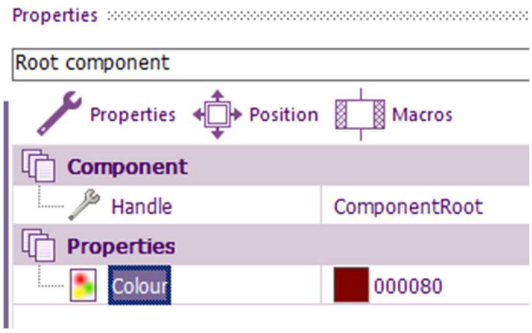
Step 5 – Expose the background colour property

Exposing a property of a subcomponent makes that property available in the created component. Open the properties for the background shape, right-click the colour property and select “Expose to top level”.

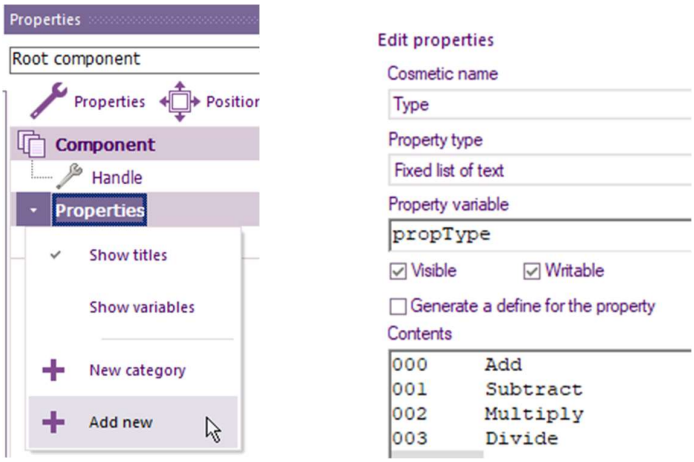


Step 6 – Add a new property for the calculation type

Properties for this new component can also be created directly by creating new properties. Select the “Root component”. The “Colour” property from the previous step should already be there:

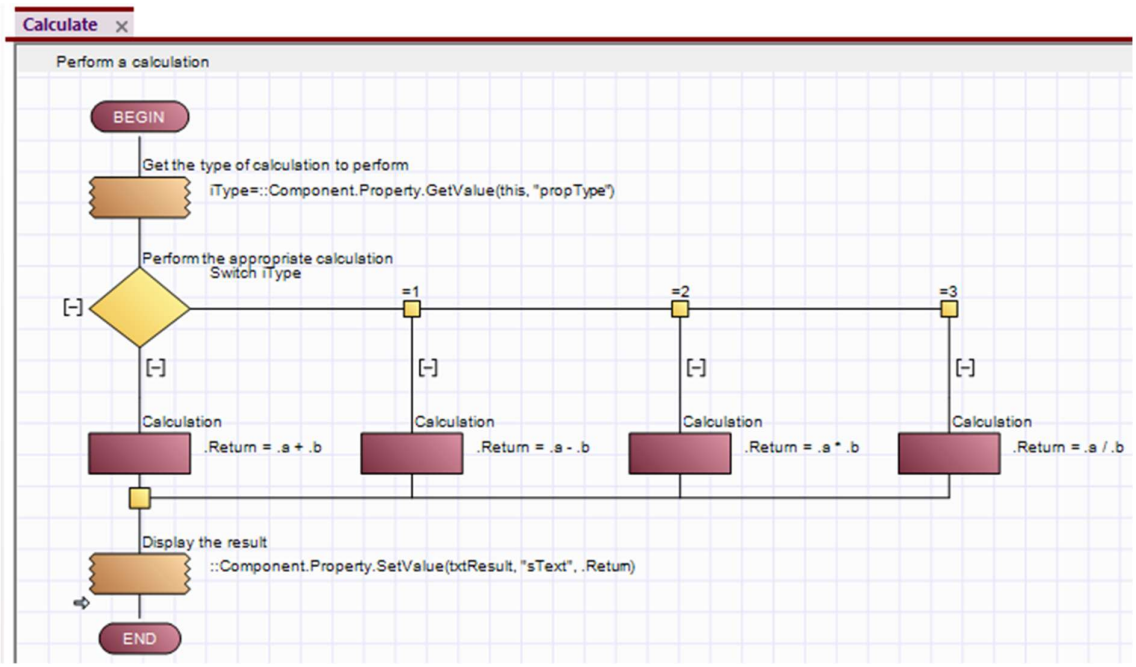


Click the dropdown arrow next to “Properties” and select “Add new”. In the resulting “Edit properties” window, create a “fixed list of text” property called “Type” with the property variable “propType” and the values as shown below:



Step 7 – Create the macro code for the calculation

Add the following icons to the “Calculate” macro:

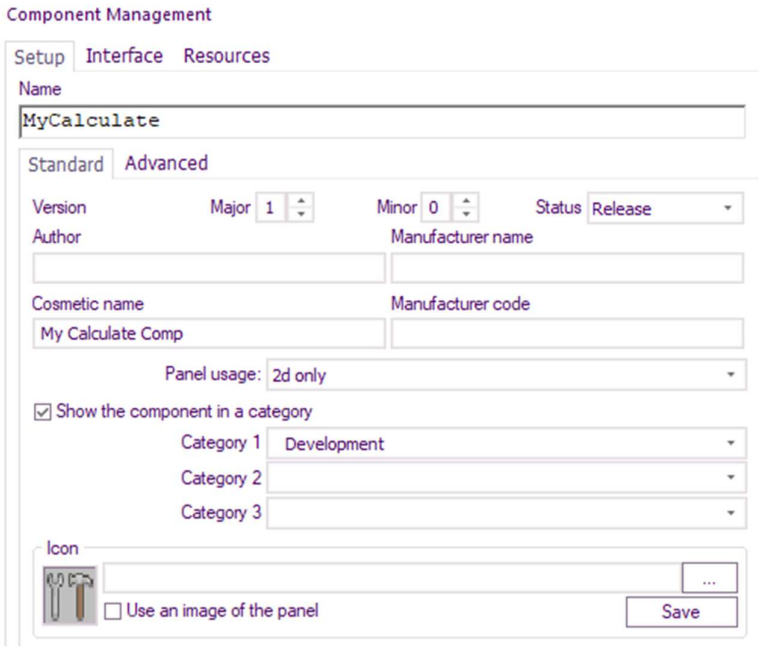


The first and last icons in this macro are Built-in Functions and are available via a Project Explorer tab. If this tab is not available, it can be added within Global Settings.

Note that the name of the “Text” property in the final icon is “sText”. This is because the visual name of the property is not necessarily the same as the actual property name. There is no error checking within Flowcode when this property name is used in the built-in “SetValue” function, so it is important to use the correct property name.

Step 8 – Edit the export settings for the project

Select File…Export and then “Edit the export settings for your component”. Give the project an appropriate name, for example “MyCalculate”, and enter this in the “Name” field. The “Cosmetic name” field should have a similar entry.



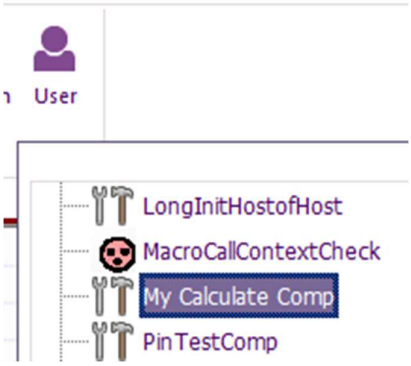
Once done, save your project.

Step 9 – Export the component

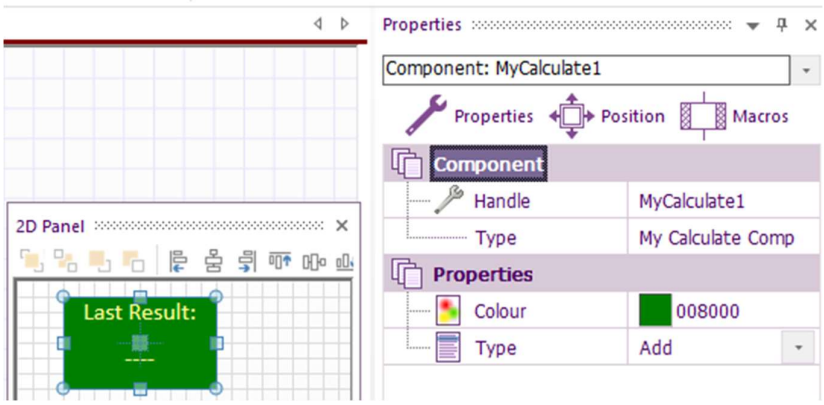
In the same export screen, select “Export this project as a component”. Save the exported component (*.fcpx file) to an appropriate location (for example, your “component dev” folder). This folder should be set up as a location in the “Look for components in…” box within File…Global Settings…Locations.

Step 10 – Use the new component in a test project

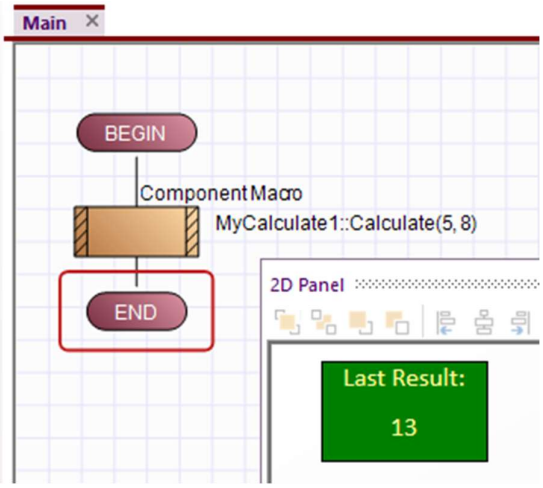
Create a new embedded project and add the new component to the 2D Panel. It will be in the “User” component list with the name you gave in the export settings.



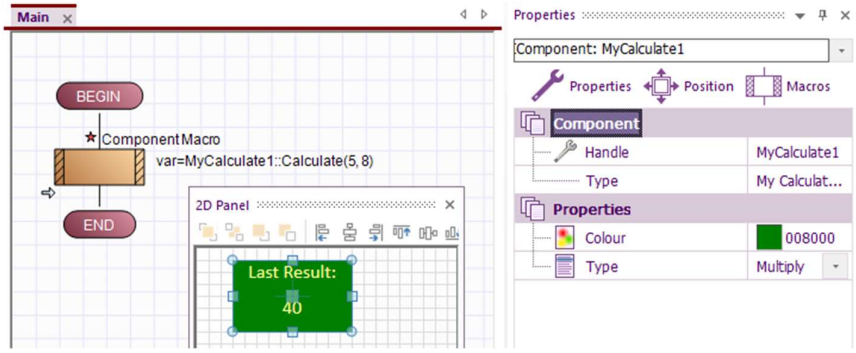
This component should have 2 properties – “Colour” and “Type” – as shown below:



Once added, call the component macro and run to simulate the project.



Try changing the calculation type and running again.



Notes

This component is set to use the “Long” integer type. This means the result will potentially be truncated. For example, when the result of a division would not produce a whole number.

Before exporting the component, you can use the “Main” macro in the component to test any functionality. The “Main” macro is never exported.

This component is designed for simulation. It will produce appropriate microcontroller code to perform the calculation, but the “propType” property cannot be changed once a project is compiled. If similar functionality was required in an embedded component, then another method of setting the calculation type should be used. For example, separate macros for each type or pass the type of calculation as a parameter to the “Calculate” macro.