FLOWCODE 8

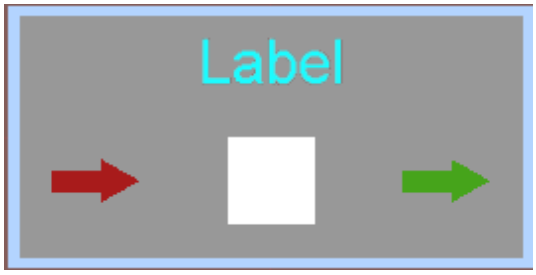
4

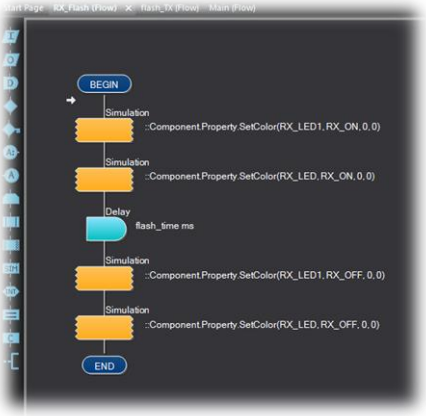
Component Creation

Advanced Sub-Component - Communication Flasher

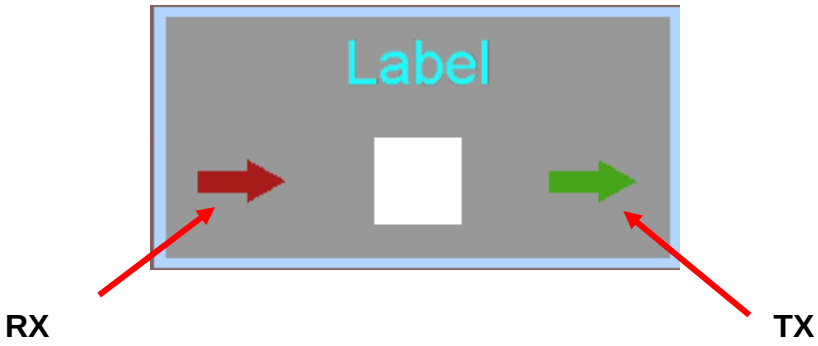
Build a Communication(s) Flasher

Advanced Sub_Components





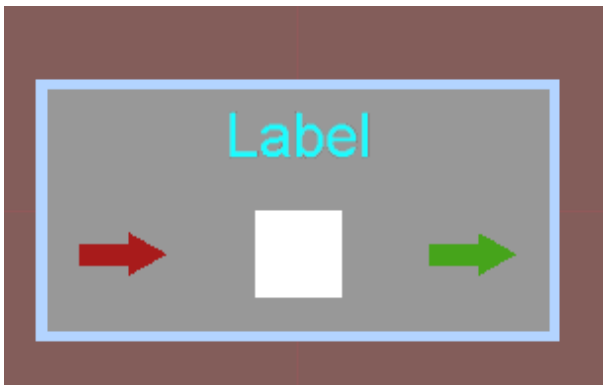
I would like to show you how to build an advanced **sub_component** that can be used for advanced component designs, such as those used in communications i.e. UART; I2C; SPI; One-Wire and CAN bus. This sub_component is the basis of simulation techniques that require a more advanced level of simulation. If you have a closer look at these components, what they have in common are simulation features that transition from one state to another automatically during communication they briefly flash (transition colour changes from bark-to-bright-to-dark). One feature for each communication port RX & TX. This technique can be used for 7 segment display designs if you add more shapes to your design!



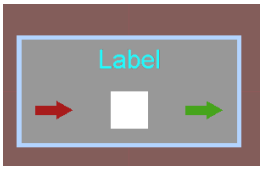
The advanced sub_component shown above has two arrows **RX** & **TX.** We need those arrows to briefly flash when a communication action is performed. No problem I

An advanced Sub_Component to build Yourself

Comm_flasher



1 - Com_flasher



Step 1 start a new project called **Comm_flasher**

Remember, it does not matter which embedded controller you use as none of the simulation code is downloaded onto the chip.

Step 2 The Canvas layout

Begin by placing the following shapes:

(i) Two rectangles onto the 2D Dashboard panel using the following sizes. You need to place the grey rectangle on top of the larger blue rectangle.

60mm x 30mm Blue **FE7350** outline same colour

55.5mm x 25.5 mm Grey **2D2D2D** outline same colour

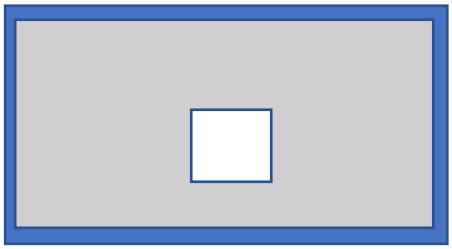
(ii) Next, place another but smaller rectangle on top. It needs to be white for best effect, but you can experiment. Make sure you arrange them correctly to achieve a layout that suits your style/requirements

10mm x 10mm White outline same colour

(iii) ***Label***

Place a text Label above the smaller square. We will be setting up how these can be accessed for use in your component designs that require two-way communication.

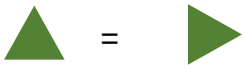
You should have something like this:



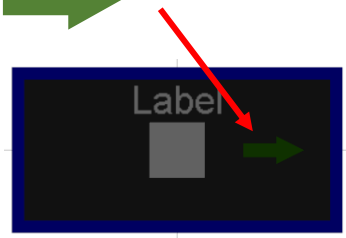
Step 3 – The Flashing Arrows

To construct the arrows, we need the following shapes:

Small rectangle 2.5mm x 6mm 

Small prism 5mm x 5mm rotate **Z** axis **-90.0** (rotate right) 

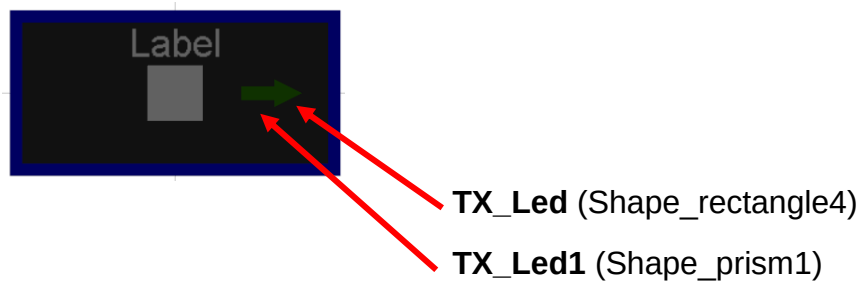
Place them next to each other to get 

It should look something like this: 

Before we do the next arrow, we need change the ***Component Properties HANDLE*** to the following for this arrow:

Shape_rectangle4 to TX_Led

Shape_prism1 to TX_Led1



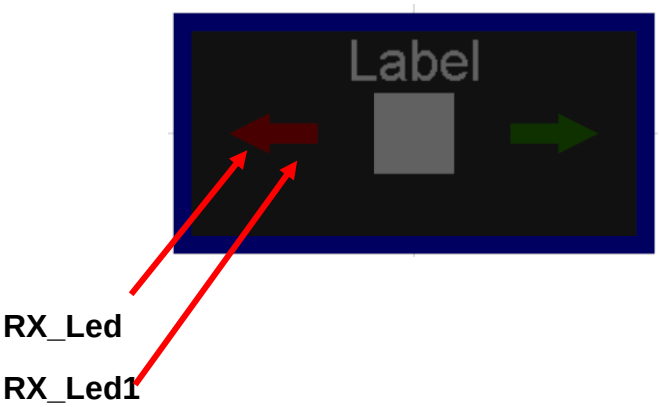
Step 4

Repeat the above for the next arrow. However, change the **Z** axis for:

prism to **z = 90.0** (should be pointing the other way rotate left)

Shown below is how it should look. Once you are happy with the arrangement, remember to change the names of the RED arrow shapes as shown below.

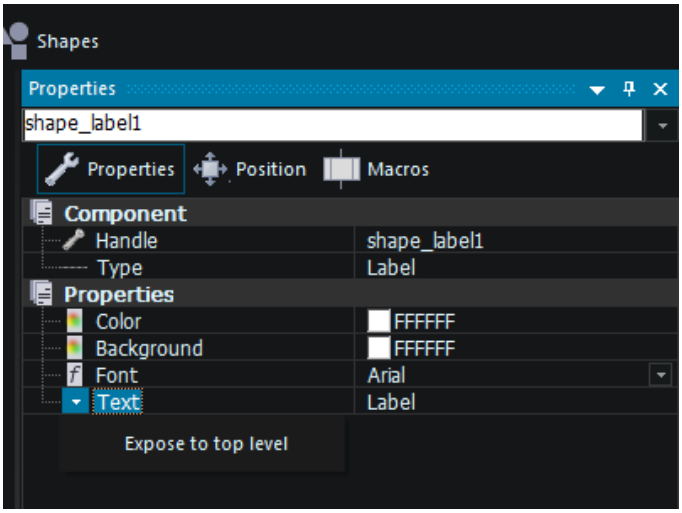
Change ***Component Properties HANDLE*** name to the following:



It is very important to change the **Handle** names as it makes referencing them much easier when we begin to write the code for the simulation **flash or blink effect.**

Step 5

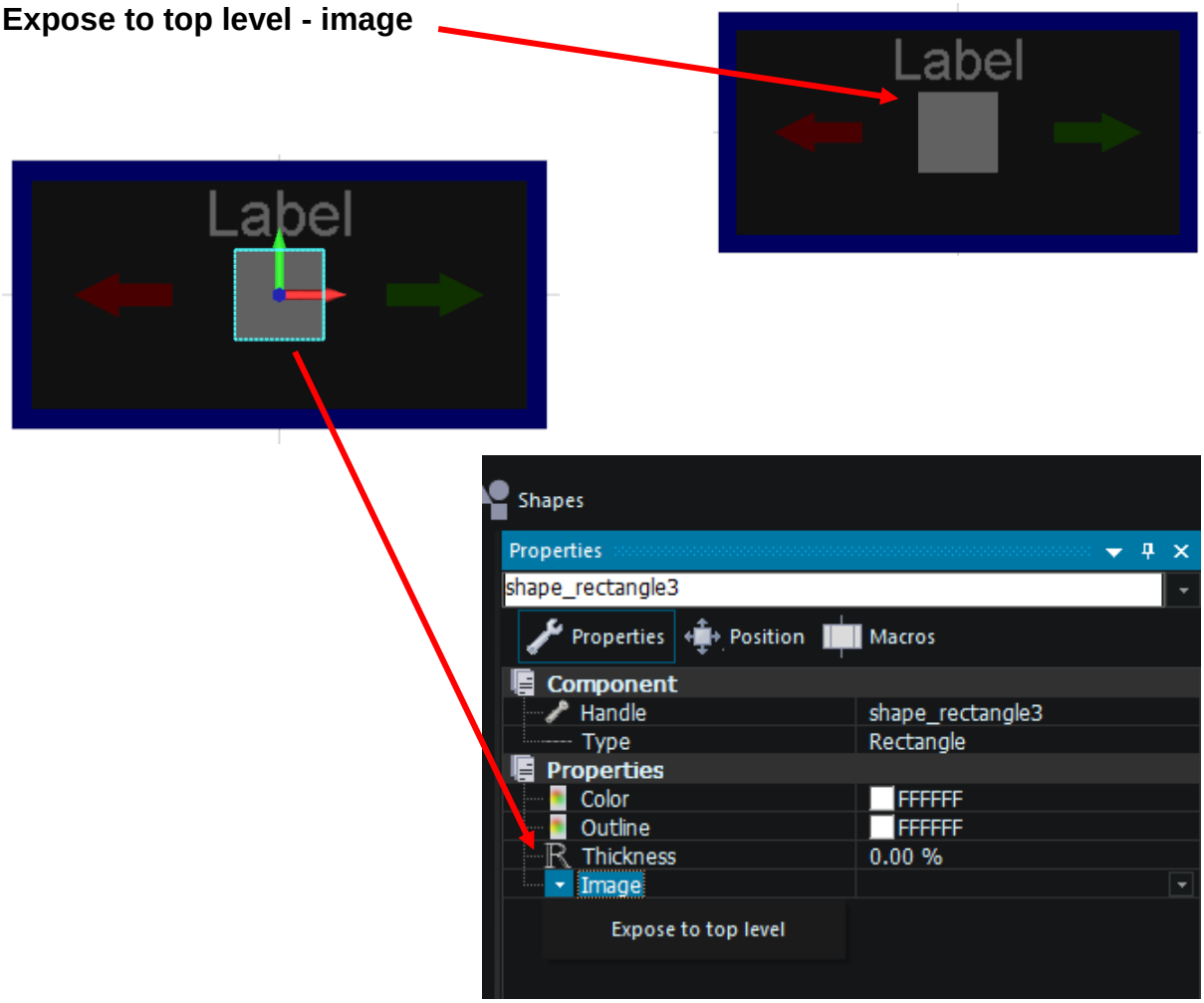
Next, click onto the **text *Label*** and from the drop-down menu, select **expose to top level.**



Step 6

Click onto middle 10mm x 10mm square, and from the drop-down menu select:

Expose to top level - image



These are all the shapes that are required to construct the communication flasher. Now we need to setup:

- colour selection for **RX** arrow (this MUST be different to the TX arrow)
- colour selection for **TX** arrow
- a way to control the flash period between colour transitions (*its not going to be on all the time*)

Step 7 Properties Panel Set Up

Now we need to set up our new sub_components properties. Click anywhere on the 2D Dashboard panel to begin, **not** on the actual new sub_component. Change the following Headings:

Component

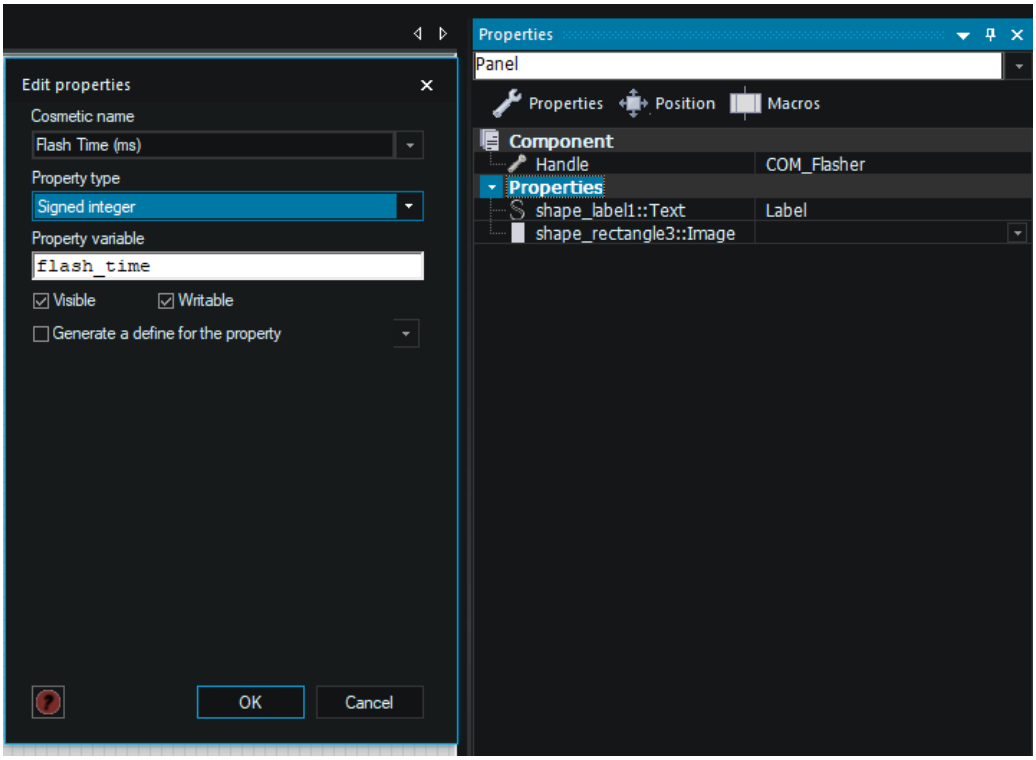
Handle ***COM_Flasher***

Properties

This should be blank

In order to get our automatic flashing arrows to work i.e. b **once** during data transfer of - *outgoing data and incoming data respectfully.* To set this time delay for the **ON** to **OFF period,** we need an adjustable variable for controlling the flash time period. We can achieve this with a delay macro that uses a variable the user can change from the properties panel, ***see*** below.

From the drop-down menu **Properties**:



Select ***add new*** and set a variable:

Cosmetic name: flash_time (ms)

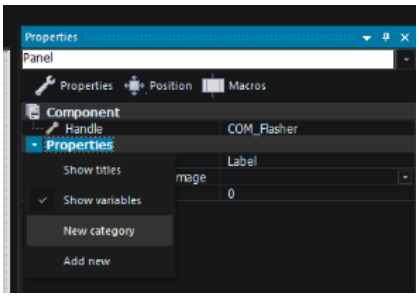
Property type: Signed integer

Property variable: flash_time

Next

Use the drop-down menu and select **New category**

create a new category called **Colour**

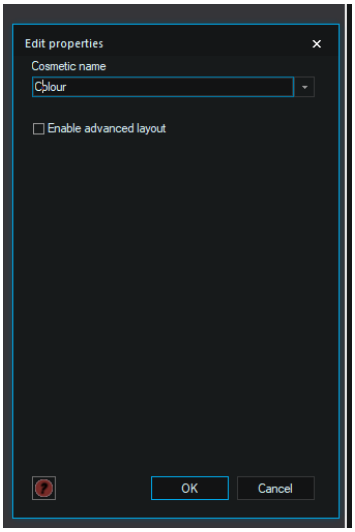


Click OK

Step 8 - Colour Pickers

We need to be able to select the colours for each arrow separately for the **ON** colour and the **OFF** colour. Also, we need to place them in a suitable category shown here. To achieve this, from the **Properties** drop down menu select:

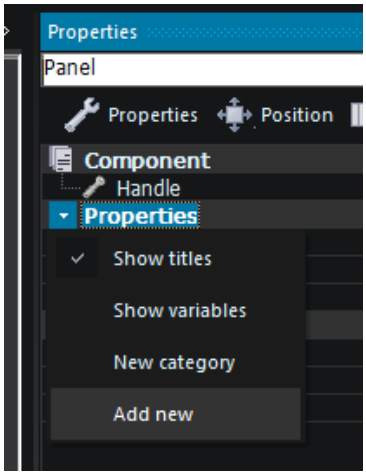
- **New category**



Name it **Colour**.

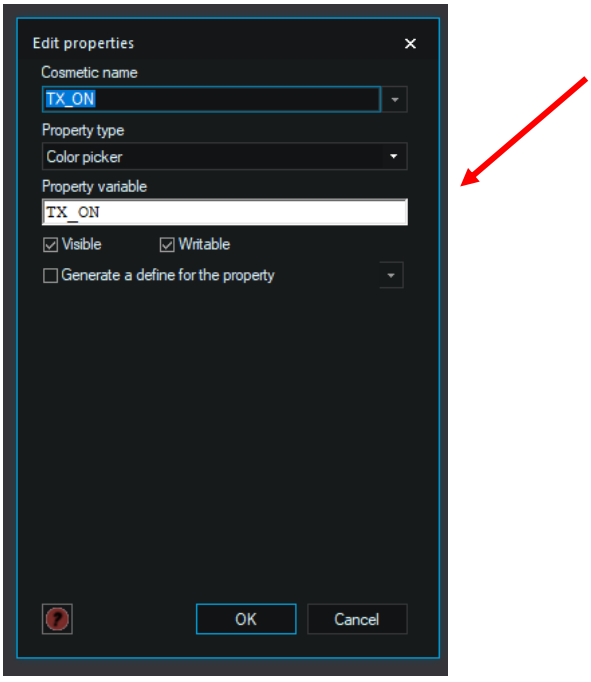
Next

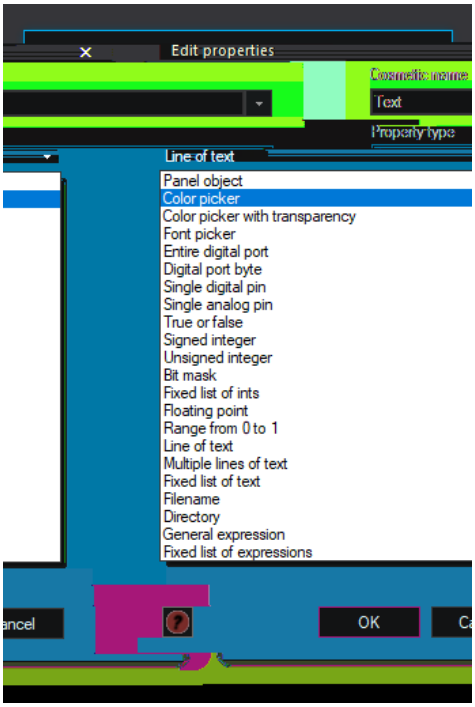
Create four colours using the **Properties** drop down menu **Add New** shown here:



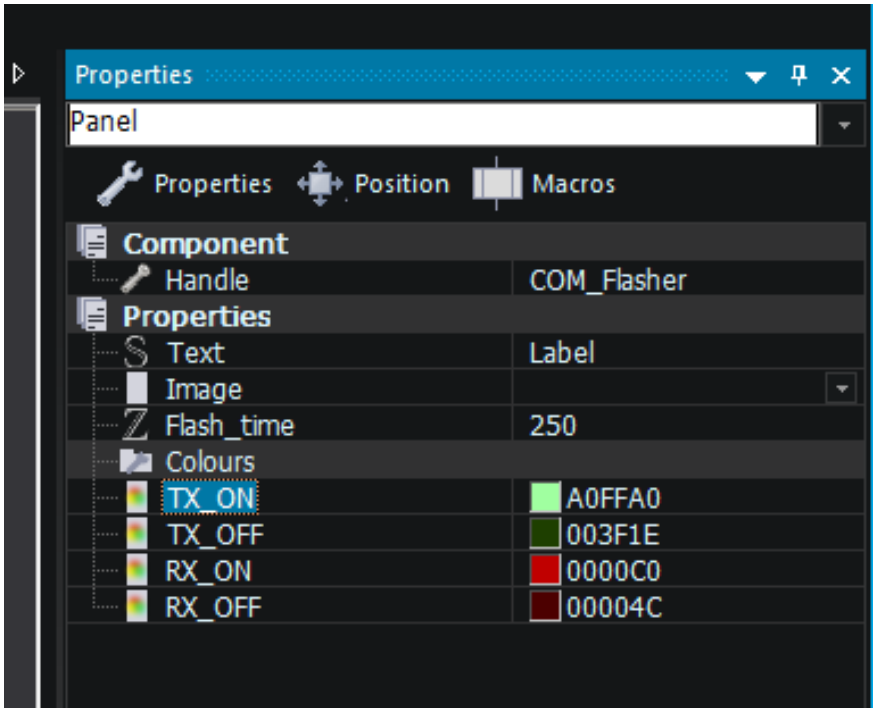
Using the **Properties** color picker, create four colours properties:

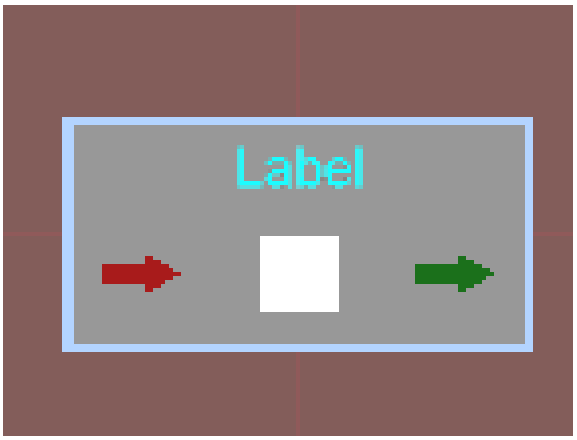
- **TX_ON**
- **TX_OFF**
- **RX_ON**
- **RX_OFF**





You should now have something that looks like this:





Step 9 – Component Simulation Macros – TX & RX

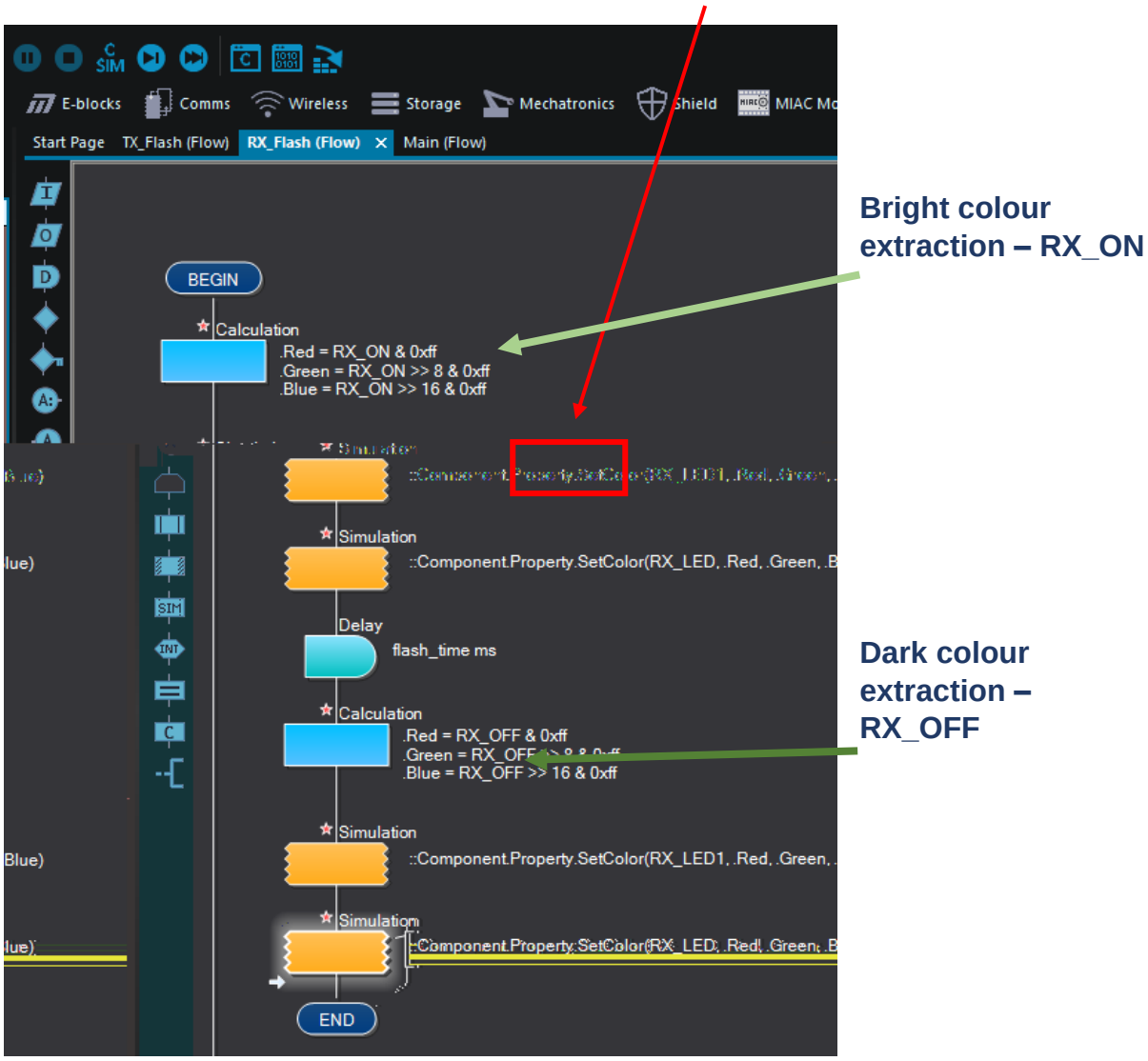
Create two macros called;

TX_Flash (flow)

RX_Flash (flow)

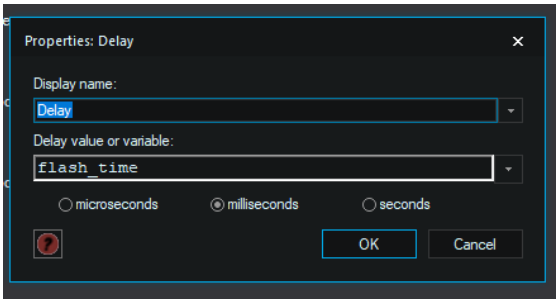
Place the following code into each macro. The **Main MUST** be left empty (no code).

For the **RX_Flash** macro:



How this is achieved is shown below.

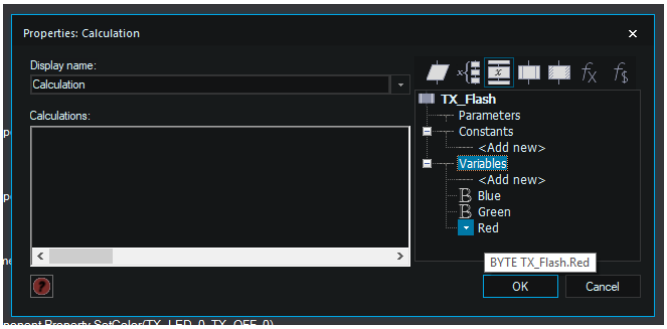
The delay is achieved using variable **flash_time** as shown here:



Create three **Local** variables for each macro;

Red; Green; Blue

All byte 0 to 255



Do this for each macro TX & RX

The calculation

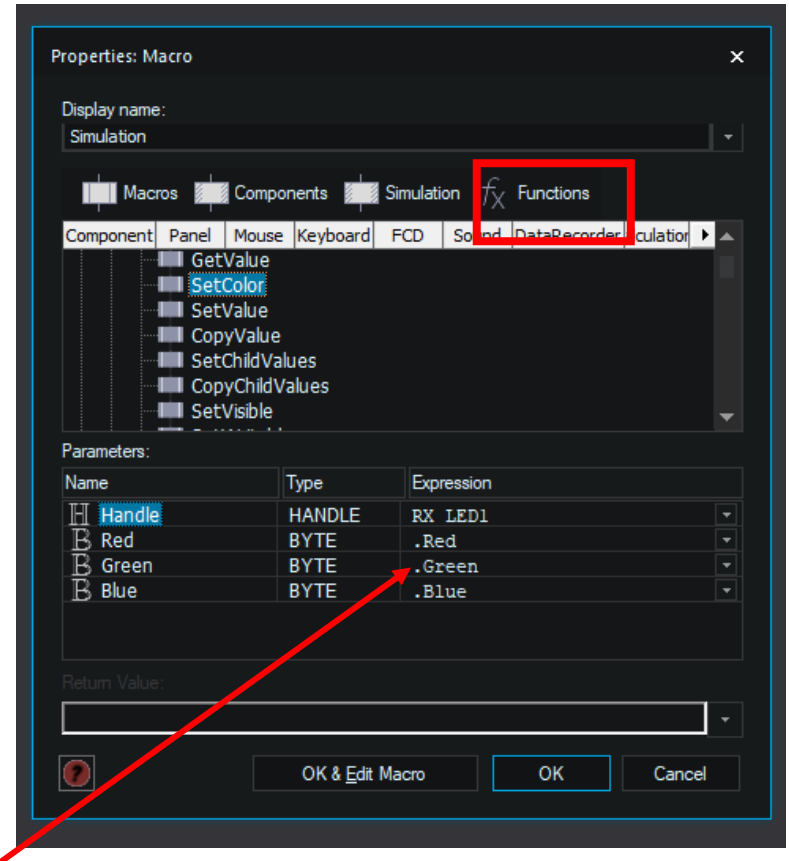
.ON_colour = ON_flash

.Red = .ON_colour & 0xff

.Green = .ON_colour >> 8 & 0xff

.Blue = .ON_colour >> 16 & 0xff

The simulation macros are setup as follows **TAB Functions - SetColor**:



The FULL STOP indicates a local variable with a list of local variables!

Place a simulation macro for each individual shapes and select:

- **Functions**
  - **SetColor**
    - **HANDLE** (choose each shape that makes the arrow LED0; LED1)
      - Select the colour property for the colour change

What we are trying to achieve, the first two simulation icons turn each shape (LED0; LED1) used to construct the **TX** arrow, a bright colour (**GREEN**). Then wait a while. Then turn them back again to a darker colour.

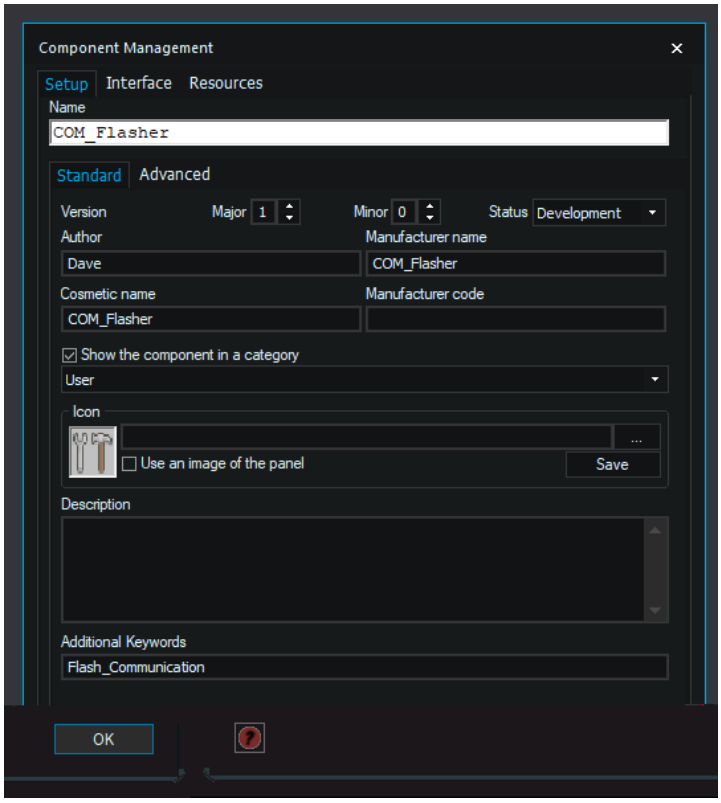
It will all happen so quickly as to look like the arrow is a single item.

Repeat for the **TX** macro using the correct shapes.

You will have noticed that choosing the appropriate names for each of the shapes is important. Even though I used the names LED & LED1, Flowcode labelled them LED0 & LED1. Takes a bit of getting used to.

Step 10 – Component Configuration and Export

From the task bar **– File** select: - **Component configuration**

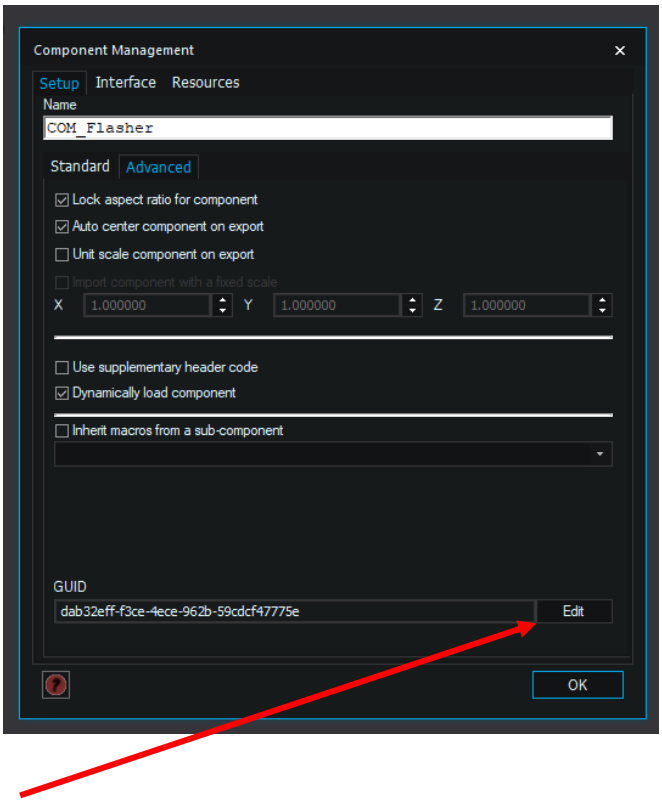


You have two Tabs to setup:

(i) Standard

Complete the above as shown including a description the suits your chosen application. At the moment it is left blank until the flasher has been used to make an actual working component for a device.

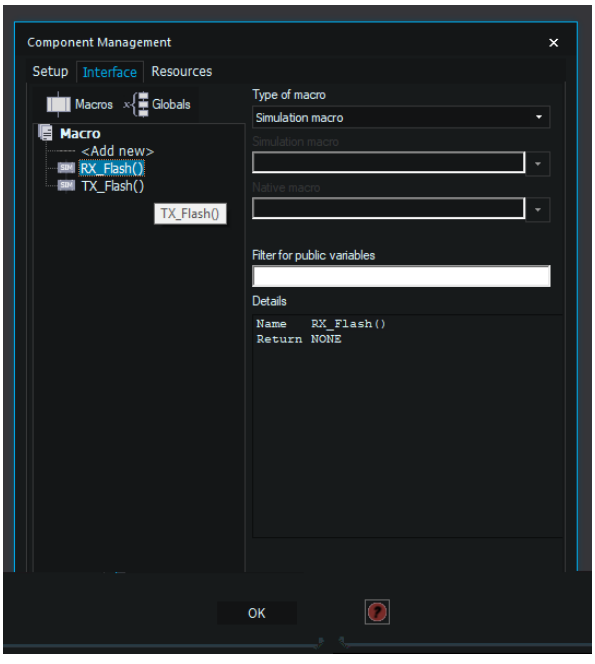
(ii) Advanced



You must make sure you set a new **GUID** for this advanced sub_component. To do this, select **Edit** and then **New**.

Click ok when done.

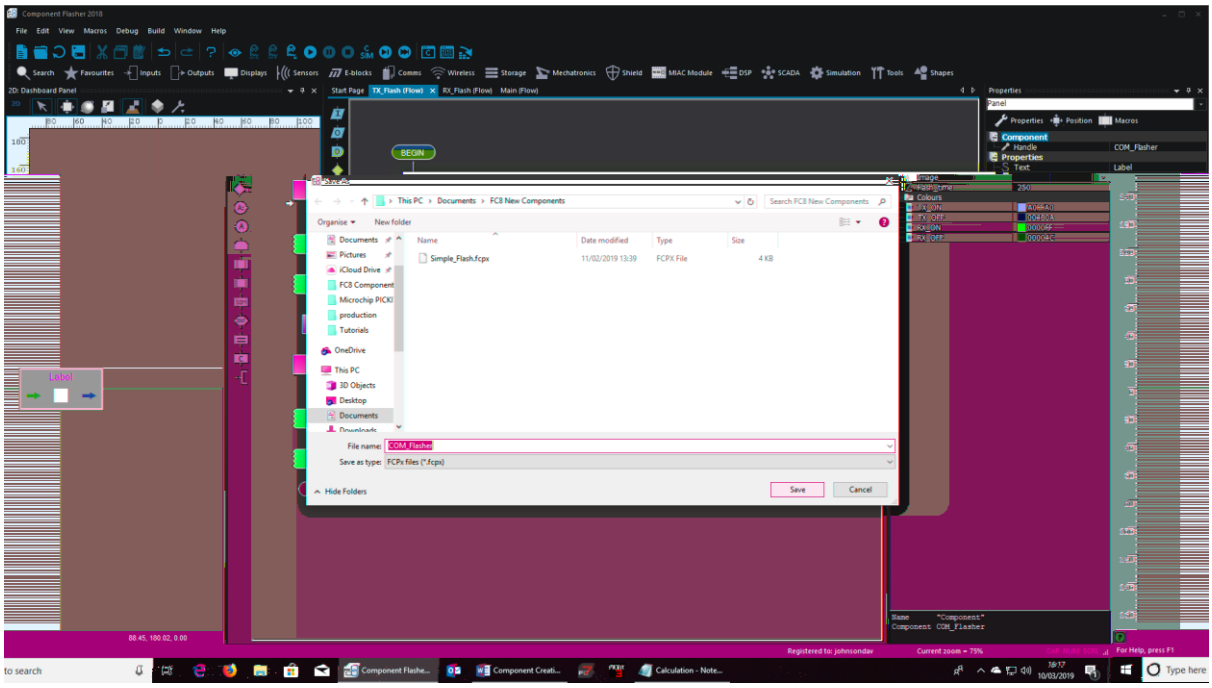
- **(III) Interface**



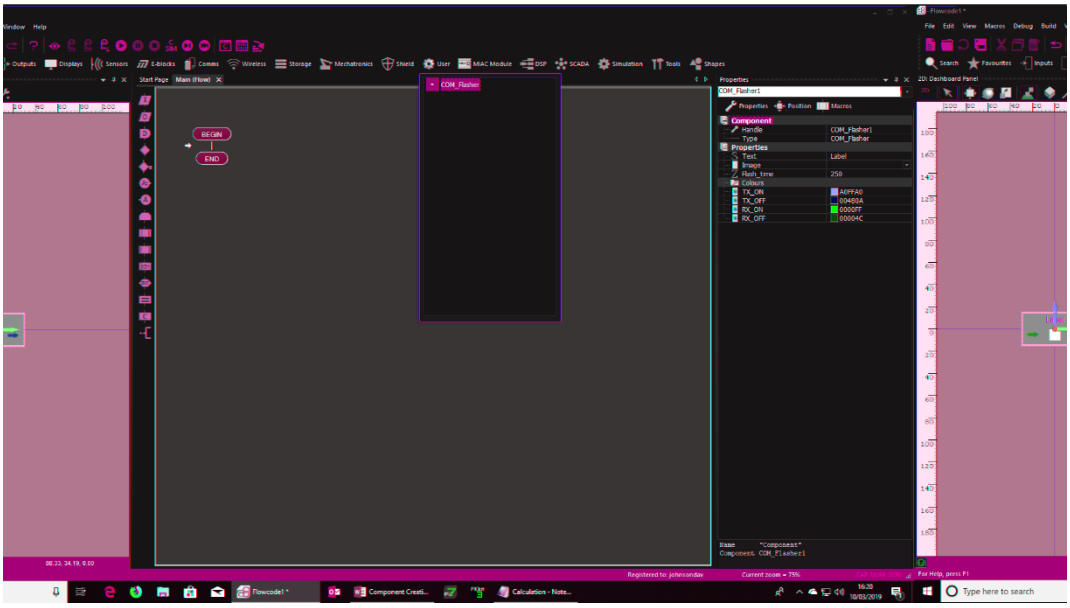
Both Macros must be set to – Type: Simulation

Next Export advanced sub_component

To do this, select File and then Export Component:



Click OK when done and save all your work. Close Flowcode and restart FC. If all is ok you should be able to start a new project and load your advanced sub_component from the TB **User** as shown:



The new sub_component will appear wherever you set it in the component configuration. I selected **user** as shown.