Creating Graphics on the i.MX-RT 1060

Crank software inc.

MCUXpresso Environment

Storyboard FreeRTOS project

The project you have in your workspace is pre-configured and ready to run Storyboard. It's been built on top of the FreeRTOS hello world project included with MCUXpresso. Configuring FreeRTOS with Storyboard isn't part of this course. If you'd like to know more about this topic, please reach out to support@cranksoftware.com. They can provide you with documentation that walks you through configuring this project from scratch.

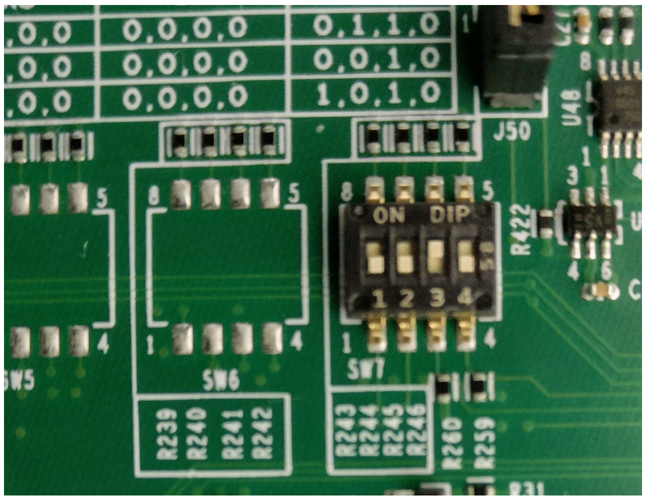
Storyboard specific files

There are a few files that we will be interacting with today that you should be aware of. The source directory is where we will spend our time.

- **freertos_sbengine.c**
  - Contains the application's main function, initializes the board and required spawn tasks for things like Storyboard, input, etc..
- **sbengine_plugins.h**
  - This is where you configure and link in the plugins that your storyboard application uses
- **sbengine_task.c**
  - This is where Storyboard integrates into the FreeRTOS application.
  - **run_storyboard_app() : line 120**, launch and bootstrap the application. Configure SBIO and it's callback handlers.
  - **sbengine_main_task() : line 157,** handle any plugin parameters and trigger the run method.
  - **gr_generic_display_init() : line 324**, configure the display and pass in framebuffer information to Storyboard.
  - **gr_generic_display_update() : line 363**, function that handles swapping and copying buffers to the display hardware.

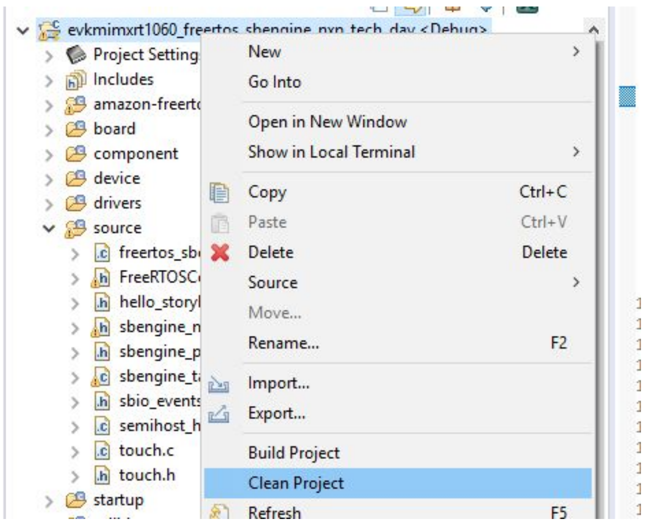
- **sbengine_input_task() : line 373**, handle input events from the touch panel on the lcd and inject them into storyboard.

Testing our setup

Step 1 - *Check hardware dip switches* Make sure the hardware is configured to boot from flash. Check the dip switches on the underside of the board and ensure that are set to "Off, Off, On, Off". Plug the i.MX RT-1060 into your laptop.	
Step 1 - *Launch MCUXpresso* Launch MCUXpresso from the desktop shortcut. Select the default workspace.	

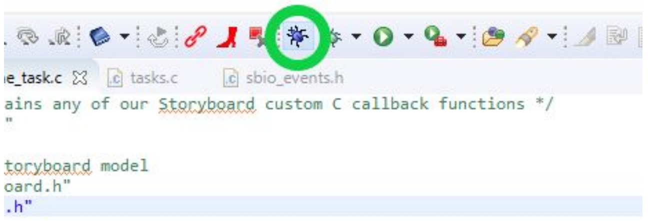
Step 1 - *Clean our Project*

Right click on the project labeled. *"Evkmimxrt1060_freertos_sbengine_nxp_tech_day"* and select clean.



Step 2 - *Build and flash*

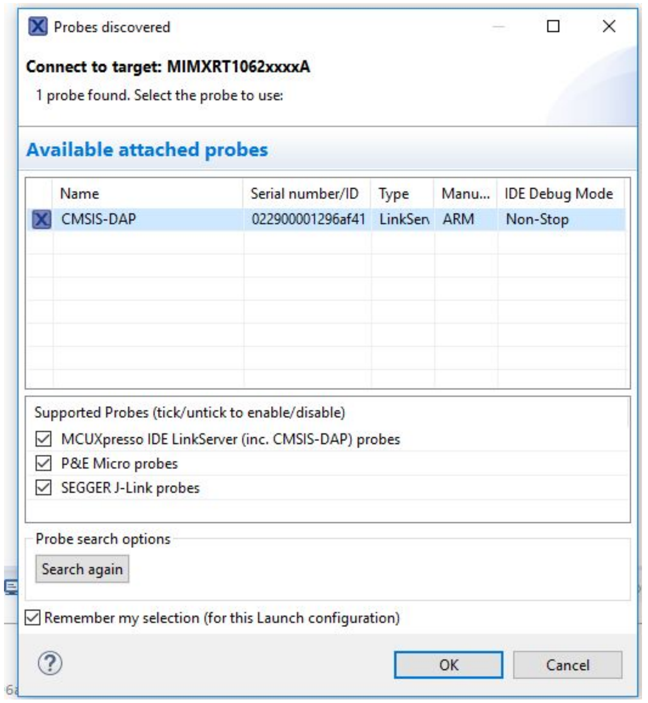
Select the blue debug icon in the toolbar.



```
ains any of our Storyboard custom C callback functions */
"

toryboard model
oard.h"
.h"
```

A dialog will appear asking you to select a debug probe. Leave the default selected one and press OK.



Probes discovered

Connect to target: MIMXRT1062xxxxA

1 probe found. Select the probe to use:

Available attached probes

Name	Serial number/ID	Type	Manu...	IDE Debug Mode
CMSIS-DAP	022900001296af41	LinkSen	ARM	Non-Stop

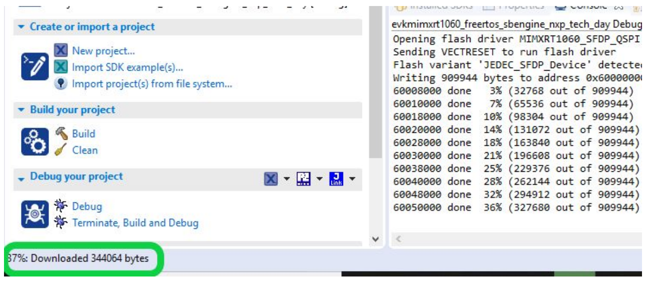
Supported Probes (tick/untick to enable/disable)

- [x] MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
- [x] P&E Micro probes
- [x] SEGGER J-Link probes

Probe search options

- [x] Remember my selection (for this Launch configuration)

You will see the application build and then the flash progress in the bottom left corner of the IDE status bar.



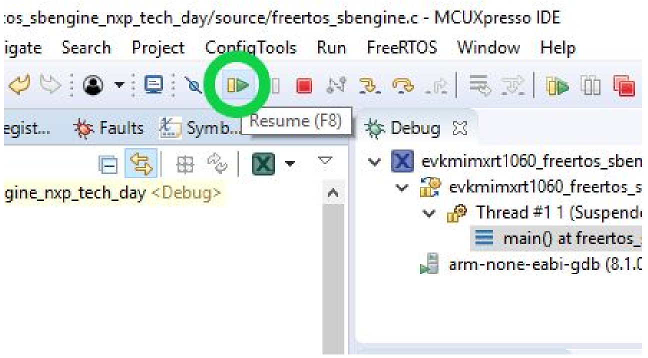
Create or import a project
New project...
Import SDK example(s)...
Import project(s) from file system...
Build your project
Build
Clean
Debug your project
Debug
Terminate, Build and Debug

```
evkmimxrt1060_freertos_sbengine_nxp_tech_day Debug
Opening flash driver MIMXRT1060_SFDP_QSPI
Sending VECTRESET to run flash driver
Flash variant 'JEDEC_SFDP_Device' detecte
Writing 909944 bytes to address 0x6000000
60008000 done   3% (32768 out of 909944)
60010000 done   7% (65536 out of 909944)
60018000 done  10% (98304 out of 909944)
60020000 done  14% (131072 out of 909944)
60028000 done  18% (163840 out of 909944)
60030000 done  21% (196608 out of 909944)
60038000 done  25% (229376 out of 909944)
60040000 done  28% (262144 out of 909944)
60048000 done  32% (294912 out of 909944)
60050000 done  36% (327680 out of 909944)
```

37%: Downloaded 344064 bytes

When flashing is complete the program will hit a breakpoint on the first line of main. Press the debugger resume button in the tool bar or F8 to continue.

You should now see “Hello Storyboard!” blinking on your RT-1060 display.



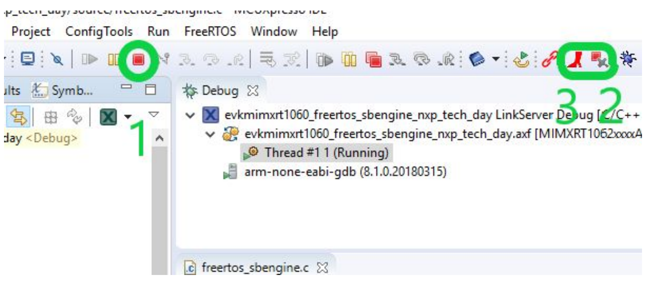
tos_sbengine_nxp_tech_day/source/freertos_sbengine.c - MCUXpresso IDE

gine_nxp_tech_day <Debug>

evkmimxrt1060_freertos_sben
evkmimxrt1060_freertos_s
Thread #1 1 (Suspend
main() at freertos_
arm-none-eabi-gdb (8.1.0

When you are done close the debug session with the stop button.

Hint: The debugger uses sockets so if it doesn't close down properly you might need to use buttons labeled 2 and 3 to clean up all debug sessions and reboot the debug probe.

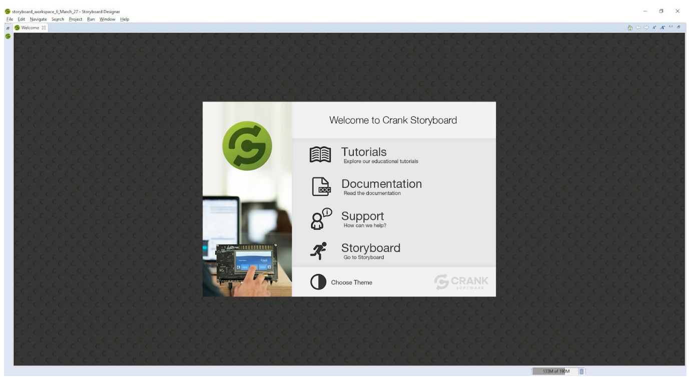

Project
ConfigTools
Run
FreeRTOS
Window
Help
Symb...
Debug
1
3
2
evkmimxrt1060_freertos_sbengine_nxp_tech_day LinkServer Debug [C/C++
evkmimxrt1060_freertos_sbengine_nxp_tech_day.axf [MIMXRT1062xxxxA
Thread #1 1 (Running)
arm-none-eabi-gdb (8.1.0.20180315)
day <Debug>
freertos_sbengine.c

Introducing Storyboard 6.0

Importing a Photoshop file and adding screen transitions.

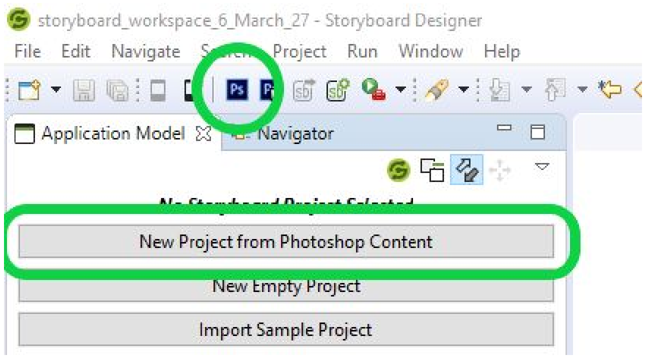
Step 1 - *Open Designer*

Launch Storyboard via the icon in the System Tray. You should see the welcome screen.



Step 2 - *Import a Photoshop file*

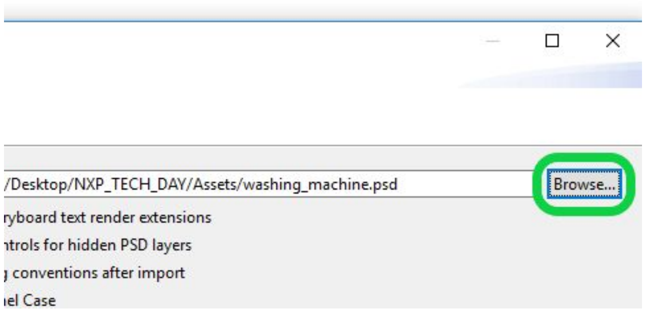
Once you close the welcome panel you should see the empty editor. Import a PSD file by selecting the PS icon in the toolbar or press the “New Project from Photoshop Content” button in the Application Model Panel.


storyboard_workspace_6_March_27 - Storyboard Designer
File Edit Navigate Search Project Run Window Help
Application Model
Navigator
No Storyboard Project Selected
New Project from Photoshop Content
New Empty Project
Import Sample Project

Step 3 - *Import PSD Options*

A dialog titled “Storyboard PSD Import” should open. Press the browse button and select the following PSD.

/Desktop/NXP_TECH_DAY/Assets/washing_machine.psd

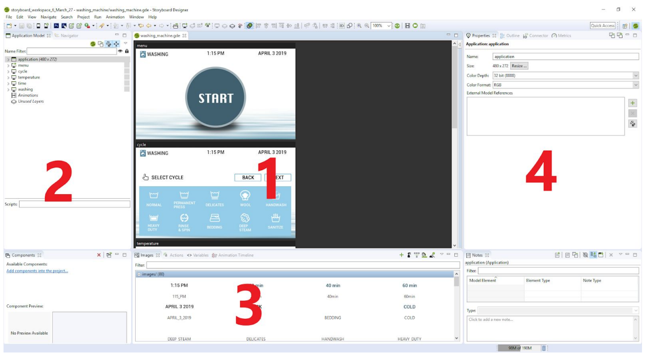

/Desktop/NXP_TECH_DAY/Assets/washing_machine.psd
Browse...
ryboard text render extensions
ntrols for hidden PSD layers
g conventions after import
el Case

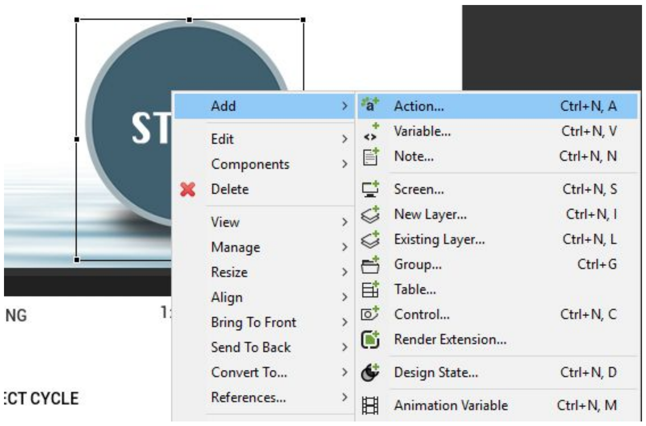
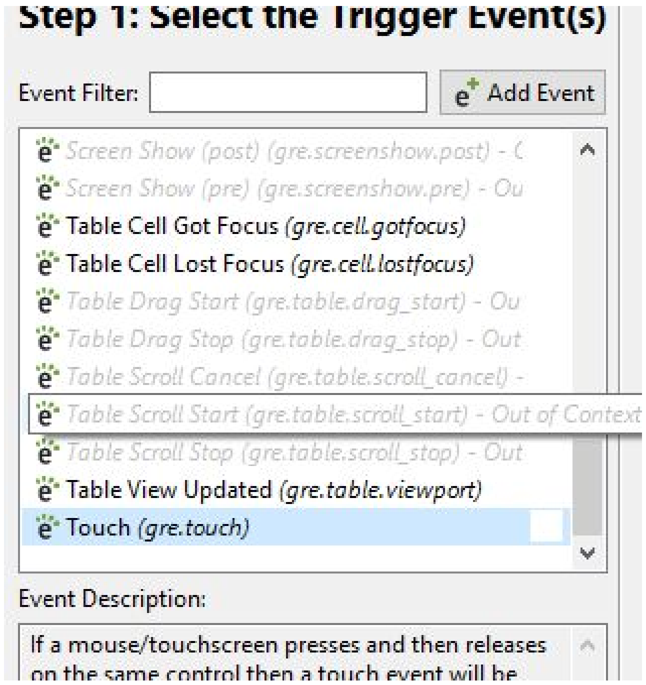
Leave everything else as is and press Finish.

Step 4 - *Explore your new project*

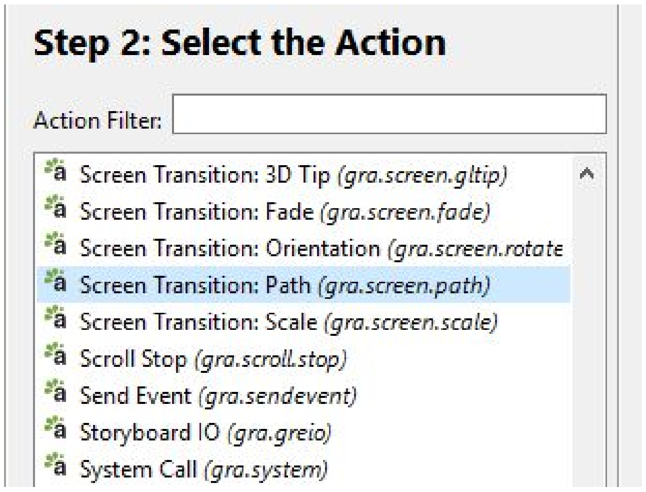
After the import process has completed you should a workspace like this. I will refer to the numbered sections as follows.

1. Main editor
2. Application Model View
3. Actions/Animation/Images Panel
4. Properties View



Step 5 - *Adding a Screen Transitions* Let's add a screen transition, select the Start button. Right click > Add Action.	 ST NG CT CYCLE Add Edit Components Delete View Manage Resize Align Bring To Front Send To Back Convert To... References... Action... Ctrl+N, A Variable... Ctrl+N, V Note... Ctrl+N, N Screen... Ctrl+N, S New Layer... Ctrl+N, I Existing Layer... Ctrl+N, L Group... Ctrl+G Table... Control... Ctrl+N, C Render Extension... Design State... Ctrl+N, D Animation Variable Ctrl+N, M
Step 6 - *Configure your transition* Nothing in Storyboard happens without an event to trigger an action. So we need to pick an event to trigger the screen transition. Select the "Touch" event in the event list. Then select	Step 1: Select the Trigger Event(s) Event Filter: Screen Show (post) (gre.screenshow.post) - (Screen Show (pre) (gre.screenshow.pre) - Ou Table Cell Got Focus (gre.cell.gotfocus) Table Cell Lost Focus (gre.cell.lostfocus) Table Drag Start (gre.table.drag_start) - Ou Table Drag Stop (gre.table.drag_stop) - Out Table Scroll Cancel (gre.table.scroll_cancel) - Table Scroll Start (gre.table.scroll_start) - Out of Context Table Scroll Stop (gre.table.scroll_stop) - Out Table View Updated (gre.table.viewport) Touch (gre.touch) Event Description: If a mouse/touchscreen presses and then releases on the same control then a touch event will be

Select the “Screen Transition: Path” action in the action list. *Note: The MCUXpresso project is configured to use this transition type so it must be selected. Animated effect require extra memory, on systems without extra memory available these should be avoided.*

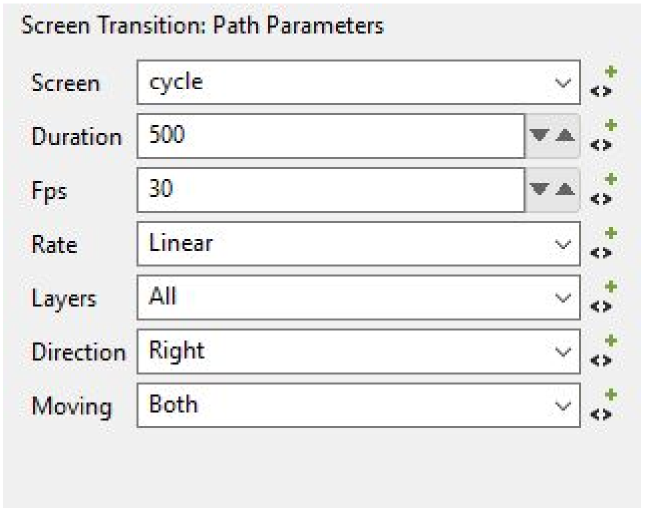


Step 2: Select the Action

Action Filter:

- Screen Transition: 3D Tip *(gra.screen.gltip)*
- Screen Transition: Fade *(gra.screen.fade)*
- Screen Transition: Orientation *(gra.screen.rotate*
- Screen Transition: Path *(gra.screen.path)*
- Screen Transition: Scale *(gra.screen.scale)*
- Scroll Stop *(gra.scroll.stop)*
- Send Event *(gra.sendevent)*
- Storyboard IO *(gra.greio)*
- System Call *(gra.system)*

Fill out the transition parameters. Here we can choose where we want to transition to and how. Select “cycle” as the screen and direction as “Right”.



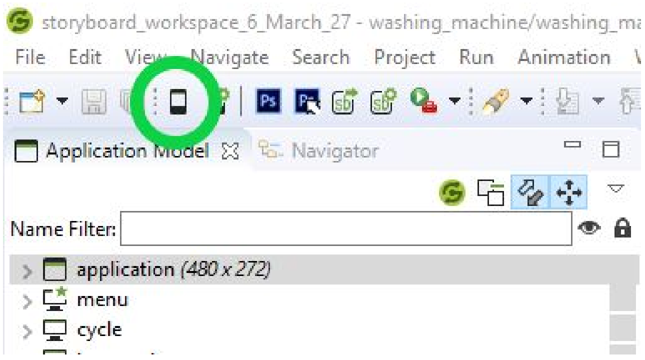
Screen Transition: Path Parameters

Screen	cycle
Duration	500
Fps	30
Rate	Linear
Layers	All
Direction	Right
Moving	Both

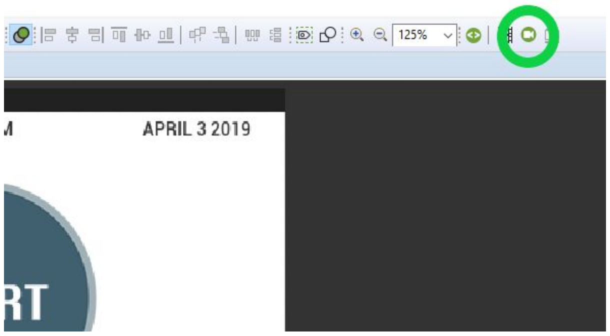
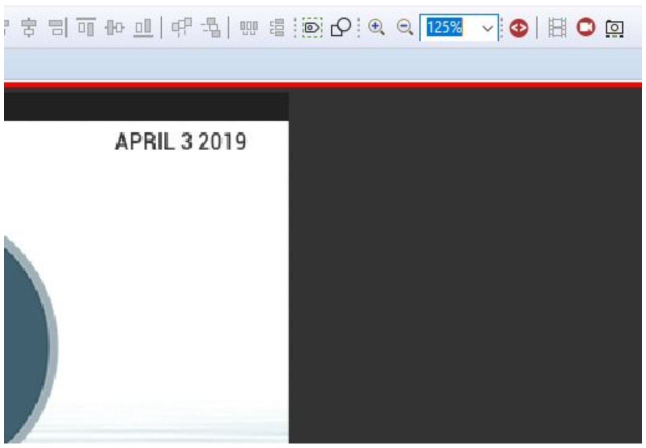
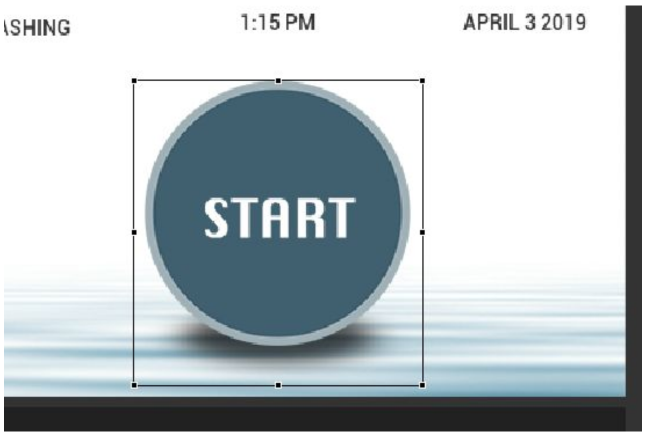
Step 8 - *Simulate and test*

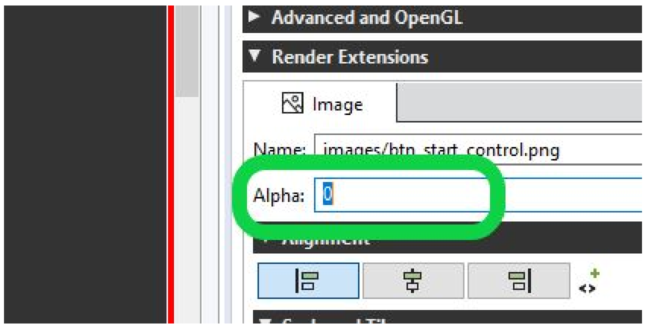
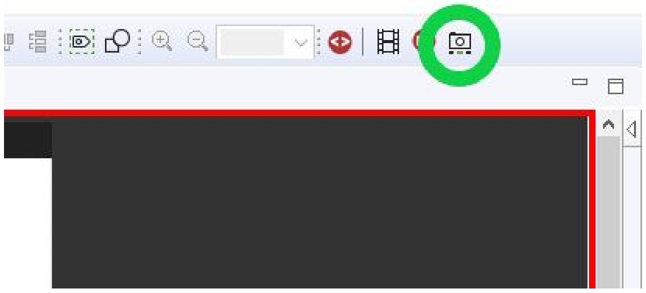
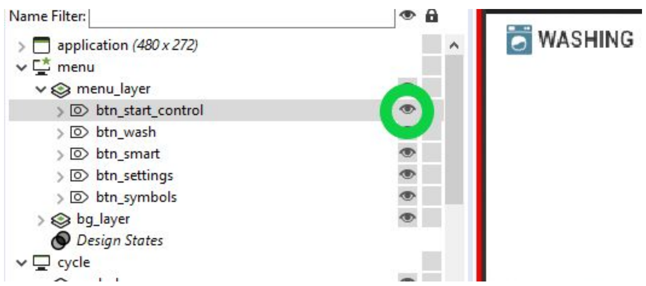
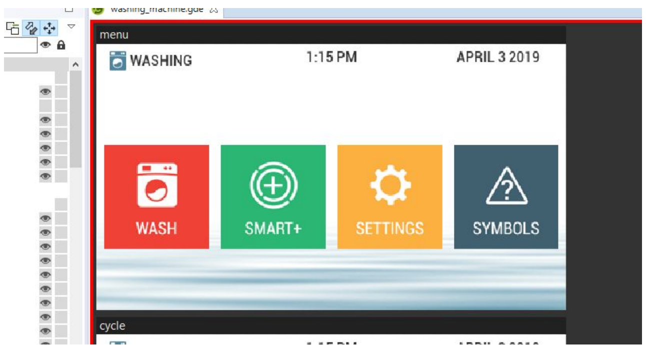
Once you have completed the transitions let's test it out. Press the simulate button in the toolbar.

If you are done early hook up transitions to move backward through your application. Use the “BACK” buttons and in the transition parameters use the direction “left”.


storyboard_workspace_6_March_27 - washing_machine/washing_ma
File Edit View Navigate Search Project Run Animation
Application Model
Navigator
Name Filter:
application (480 x 272)
menu
cycle

Creating an Animation

Step 1 - *Create an animation* Storyboard allows you to bootstrap an animation with the handy “Animation Record” feature. Press the green camera icon situated at the right end of the toolbar. The icon will turn red and the main editor window will have a red highlight.	 
Step 2 - *Create an animation step* Storyboard will now record all the changes you make and convert them to an animation. Select the start button on the menu screen.	

Set the alpha property to 0	
Press the Animation snapshot icon to turn this change into an animation "step". You will notice a label in the bottom right corner of the main editor update to say "Recording.. [Frame 2]"	
Now that the Start button is transparent we want to make sure it' marked as hidden so it doesn't receive any touch input. In the Application Model View toggle the "visible" property (eye icon). Then take another animation snapshot. You will now be on "Frame 3".	
Now that we've hidden the start button we can do something interesting with the buttons underneath. Slide them out horizontally. Since this is the last step we don't need to take another snapshot. Press the animation record button and it will finalize your animation.	

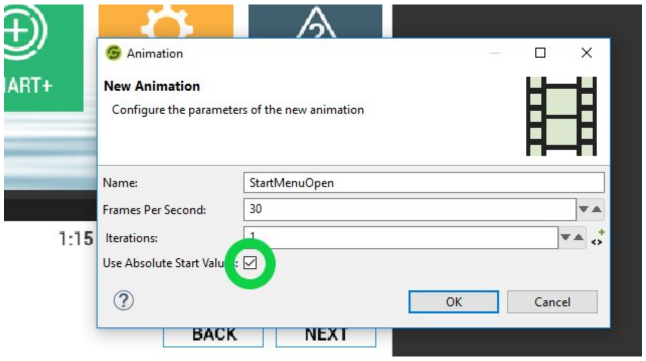
Image

Name: images/btn_start_control.png

Alpha: 0

Name Filter:
- application (480 x 272)
- menu
  - menu_layer
    - btn_start_control
    - btn_wash
    - btn_smart
    - btn_settings
    - btn_symbols
  - bg_layer
  - Design States
- cycle

You will be presented with a dialog. Give your animation a meaningful name and select the "Use Absolute Start Values" option. Then press OK.



Animation

New Animation

Configure the parameters of the new animation

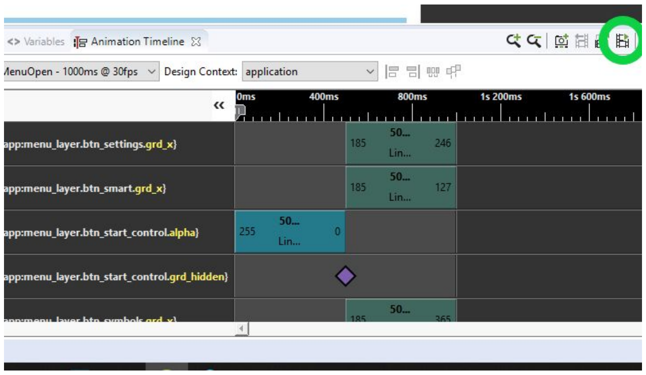
Name: StartMenuOpen

Frames Per Second: 30

Iterations: 1

Use Absolute Start Valu[✓]

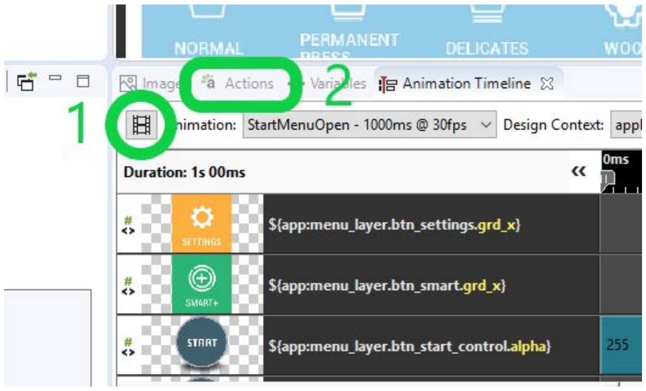
Your animation will automatically open in the timeline (Action/Animation Panel area). You can preview it by pressing the animation preview button. Each step can be selected, and edited in the properties panel if you want to make changes.



Variables
Animation Timeline
lenuOpen - 1000ms @ 30fps
Design Context: application
0ms
400ms
800ms
1s 200ms
1s 600ms
app:menu_layer.btn_settings.grd_x}
app:menu_layer.btn_smart.grd_x}
app:menu_layer.btn_start_control.alpha}
app:menu_layer.btn_start_control.grd_hidden}

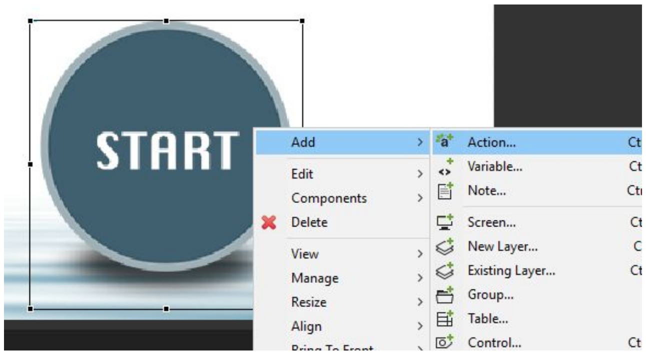
Step 3 - *Trigger the animation*

When you are happy with your animation leave the timeline by pressing the filmstrip icon in the timeline panel. Then the Actions tab.

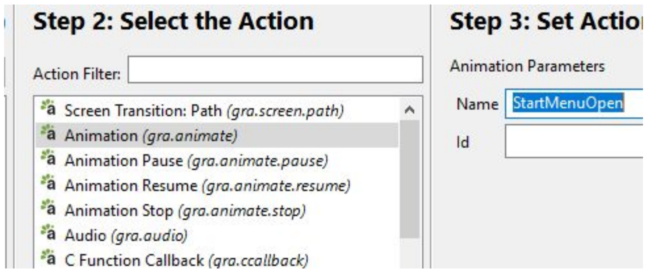


NORMAL
PERMANENT PRESS
DELICATES
Image
Actions
Variables
Animation Timeline
1
2
nimation: StartMenuOpen - 1000ms @ 30fps
Design Context: appl
Duration: 1s 00ms
0ms
${app:menu_layer.btn_settings.grd_x}
${app:menu_layer.btn_smart.grd_x}
${app:menu_layer.btn_start_control.alpha}
SETTINGS
SMART+
START
255

Select the Start button, right click and add an action.



Select a “Touch” event and an “Animation” as the action. Using the drop down for the name property of the animation select your newly created animation (StartMenuOpen). Click finish.


Step 2: Select the Action

Action Filter:

Screen Transition: Path *(gra.screen.path)*
Animation *(gra.animate)*
Animation Pause *(gra.animate.pause)*
Animation Resume *(gra.animate.resume)*
Animation Stop *(gra.animate.stop)*
Audio *(gra.audio)*
C Function Callback *(gra.ccallback)*

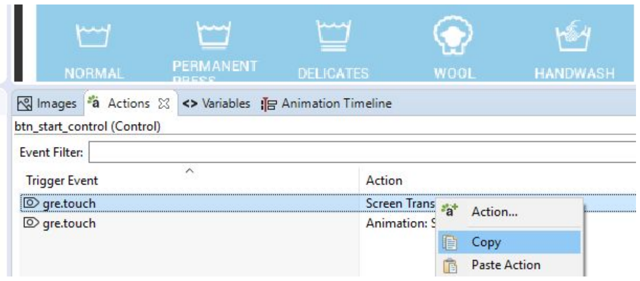
Step 3: Set Actio

Animation Parameters

Name StartMenuOpen

Id

In the actions panel you will notice that you have two touch events bound to the start button. Right click on the Screen transition event and select delete. We will move this action to wash button that will be visible at the end of the animation.

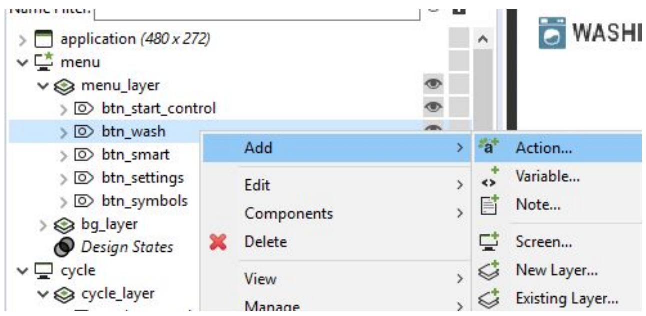


btn_start_control (Control)

Event Filter:

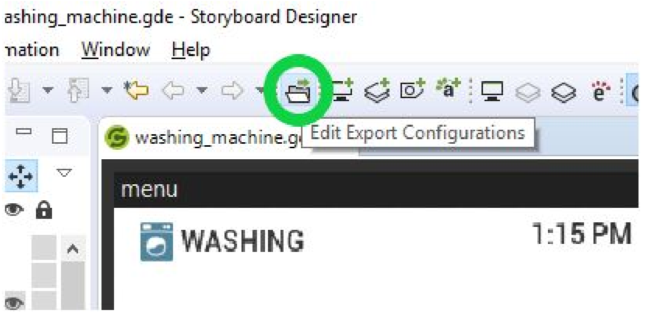
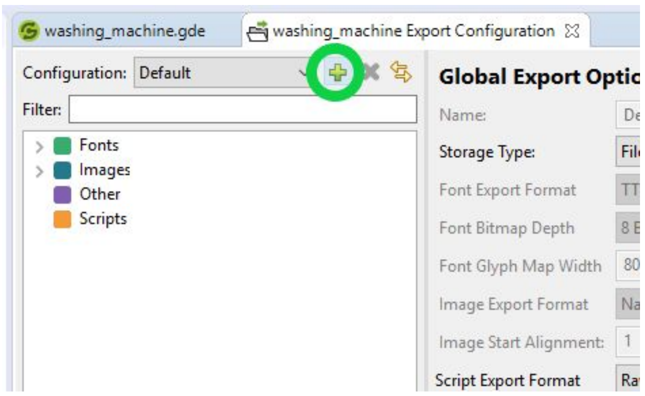
Trigger Event	Action
gre.touch	Screen Trans
gre.touch	Animation: S

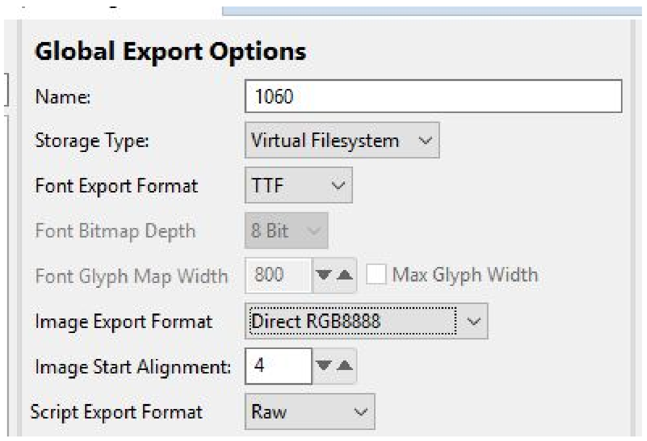
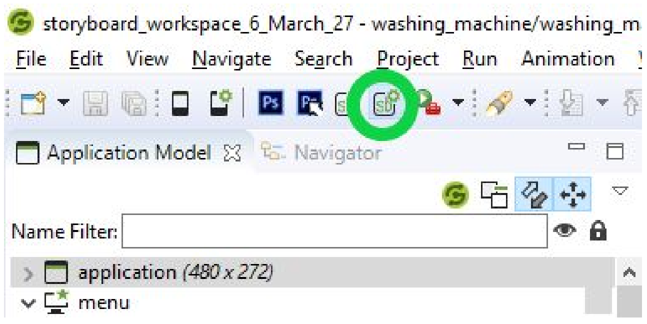
Select the wash button in the Application Model view. Right click and add the screen path transition action. Refer to the notes in the “adding a screen transition”.

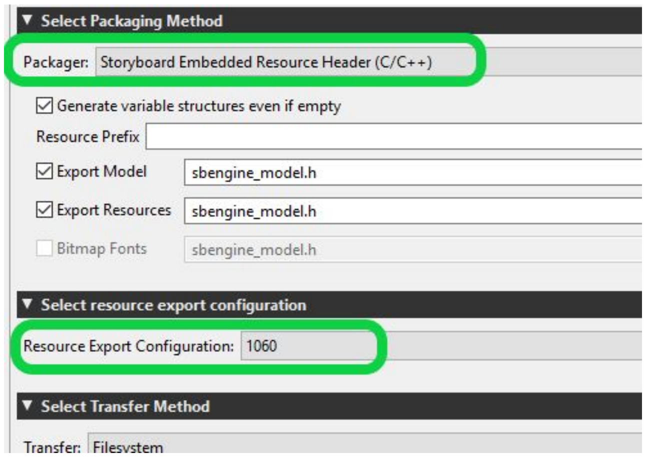
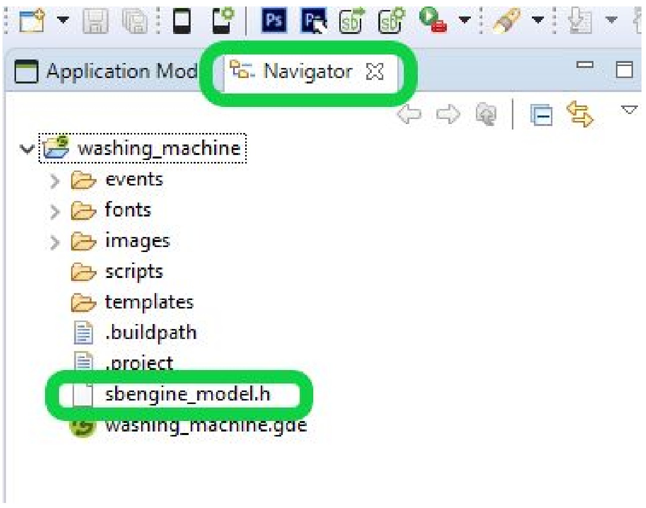


Simulate and test. You should now have an animation and screen transitions!

Preparing Your application for export

Step 1 - *Create a resource export configuration.* Now that we have an application with screen transitions and animations we should test it out on the RT 1060. To do this we need to create an export configuration. Press the Export Configurations icon in the toolbar.	 ashing_machine.gde - Storyboard Designer nation Window Help washing_machine.g Edit Export Configurations menu WASHING 1:15 PM
Step 2 - *Create a new resource export configuration.* The resource export configuration panel will open in the Main editor section. Create a new configuration for the RT 1060 by pressing the new configuration button. Give it a meaningful name (eg: 1060) and press OK.	 washing_machine.gde washing_machine Export Configuration Configuration: Default Filter: Fonts Images Other Scripts Global Export Optic Name: De Storage Type: Fil Font Export Format TT Font Bitmap Depth 8 E Font Glyph Map Width 80 Image Export Format Na Image Start Alignment: 1 Script Export Format Ra

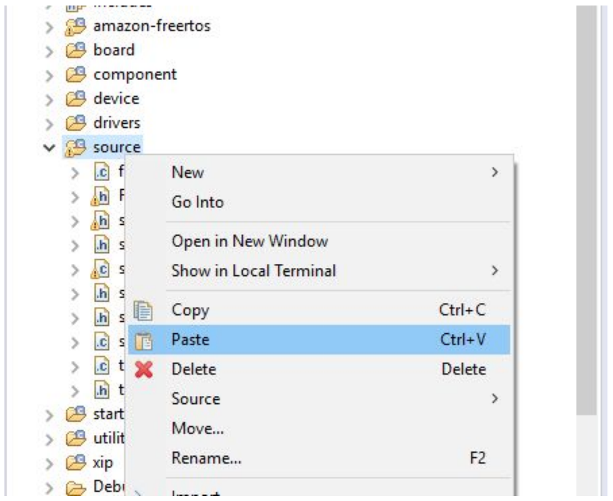
Step 3 - *Resource Export Options.* Now that we have a new configuration we can setup how the application and its assets will be exported for the 1060. Select the following options, notice how the metrics update based on what you change. ● Storage Type:Virtual Filesystem ● Font Export Format: TTF ● Image ExportFormat: Direct RGB8888 ● Image Start Alignment: 4 ● Script Export Format: Raw Press CTRL + S to save the configuration and close the panel.	 **Global Export Options** Name: 1060 Storage Type: Virtual Filesystem Font Export Format: TTF Font Bitmap Depth: 8 Bit Font Glyph Map Width: 800 ☐ Max Glyph Width Image Export Format: Direct RGB8888 Image Start Alignment: 4 Script Export Format: Raw
Step 4 - *Configure Application export.* With this new configuration we can now export our application into a C header file for inclusion in our FreeRTOS project. Press the "Storyboard Application Export configurations" button in the toolbar. Or Run > Storyboard Application Export configurations.	 storyboard_workspace_6_March_27 - washing_machine/washing_m File Edit View Navigate Search Project Run Animation Application Model Navigator Name Filter: application (480 x 272) menu

Step 5 - *Export the application.* The Storyboard Application Export Configurations dialog will have opened up. Select the following options. In the “Packager” field select “Storyboard Embedded Resource Header”. In the “Resource Export Configurations” field select your new created configuration. (eg. 1060). Press Apply and then Run. the dialog will close and you should see a success dialog in a few seconds.	 Select Packaging Method Packager: Storyboard Embedded Resource Header (C/C++) ☑ Generate variable structures even if empty Resource Prefix ☑ Export Model: sbengine_model.h ☑ Export Resources: sbengine_model.h ☐ Bitmap Fonts: sbengine_model.h Select resource export configuration Resource Export Configuration: 1060 Select Transfer Method Transfer: Filesystem
Step 6 - *Find the exported header file.* The export will have created a header file called “*sbengine_model.h*”. This can be found in your project directory. Select the Navigator tab in the Application Model view then expand your project. Right click on “*sbengine_model.h*” and copy it.	 Application Mod Navigator washing_machine events fonts images scripts templates .buildpath .project sbengine_model.h washing_machine.gde

Deploying your application to hardware

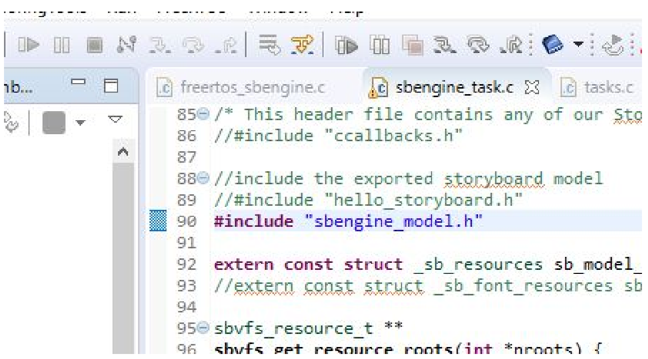
Step 1 - *Add the model to the FreeRTOS Project.*

Switch back to MCUXpresso and expand the project directory. Paste ***sbengine_model.h*** into the source directory.

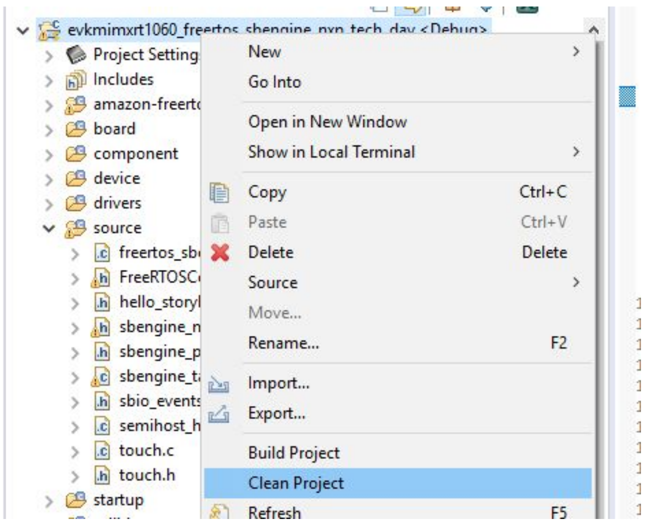


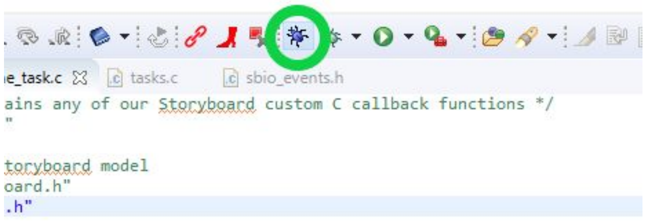
Step 2 - *Include your new header*

Open sbengine_task.c and go to line 89. Comment out the include for the hello_storyboard.h file and add your new sbengine_model.h.



```
85 /* This header file contains any of our Sto
86 //#include "ccallbacks.h"
87
88 //include the exported storyboard model
89 //#include "hello_storyboard.h"
90 #include "sbengine_model.h"
91
92 extern const struct _sb_resources sb_model_
93 //extern const struct _sb_font_resources sb
94
95 sbvfs_resource_t **
96 sbvfs_get_resource_roots(int *nroots) {
```

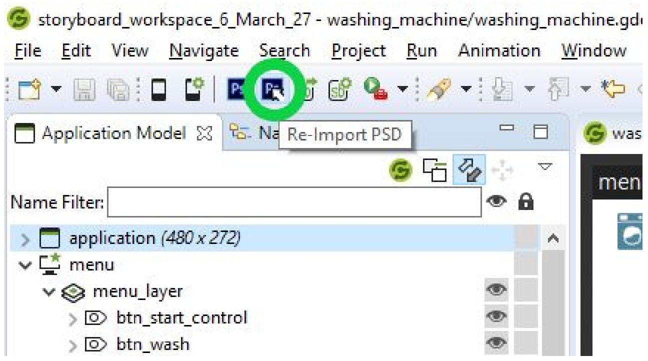
Step 3 - *Clean and flash to hardware* Clean your project and launch with debugging.	 evkmimxrt1060_freertos_sbengine_nxp_tech_day <Debug> Project Settings Includes amazon-freerto board component device drivers source freertos_sb FreeRTOSC hello_storyb sbengine_n sbengine_p sbengine_t sbio_events semihost_h touch.c touch.h startup New Go Into Open in New Window Show in Local Terminal Copy Ctrl+C Paste Ctrl+V Delete Delete Source Move... Rename... F2 Import... Export... Build Project Clean Project Refresh F5

The flashing process will take longer since we are transferring the application and all assets this time.	
	`ains any of our Storyboard custom C callback functions */`
	`"`
	`toryboard model`
	`oard.h"`
	`.h"`

Reskinning your UI

Step 1 - *Re-import a PSD*

Storyboard allows you to quickly and easily reskin/rebrand your GUI via the PSD Re-import function. Select the PSD reimport button to begin.

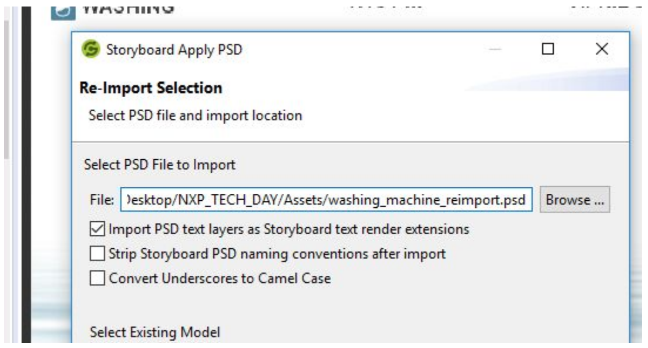


Step 2 - *Pick your PSD*

Select the following PSD in the NXP_TECH_DAY folder in the desktop.

\Desktop\NXP_TECH_DAY\Assets\washing_machine_reimport.psd

Leave the other settings default and select Next.



Storyboard Apply PSD

Re-Import Selection

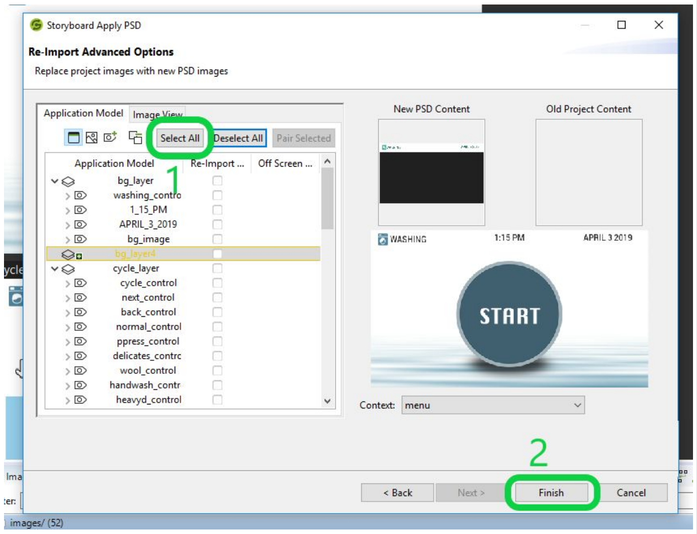
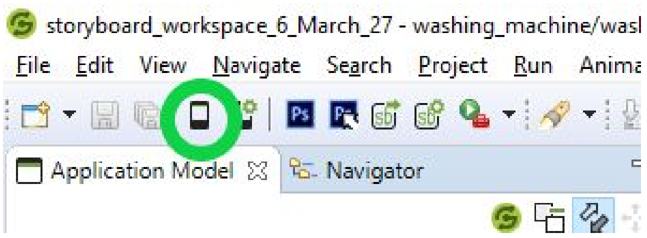
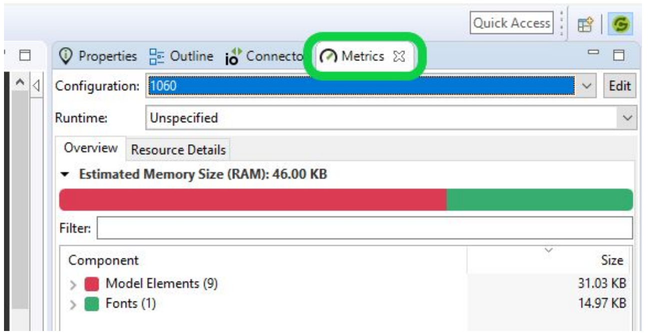
Select PSD file and import location

Select PSD File to Import

File:)esktop/NXP_TECH_DAY/Assets/washing_machine_reimport.psd Browse ...

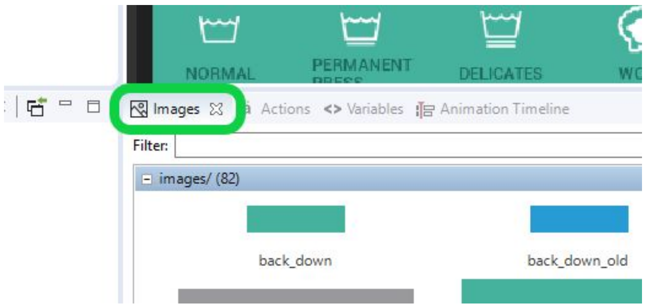
- [x] Import PSD text layers as Storyboard text render extensions
- [ ] Strip Storyboard PSD naming conventions after import
- [ ] Convert Underscores to Camel Case

Select Existing Model

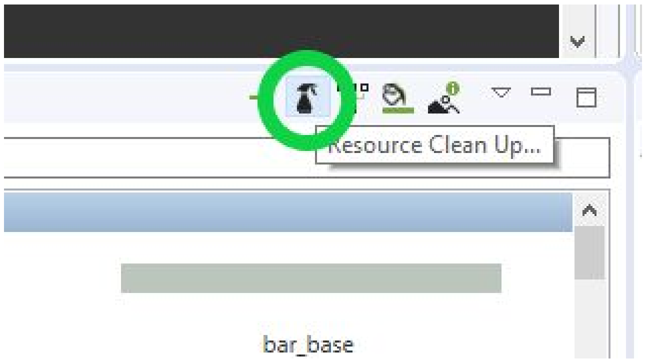
Step 2 - *Manage incoming content* Storyboard gives you the ability to manage how the new content is coming into your application. This is useful if the new PSD has different naming then the original. Storyboard will do its best to match content based on names however sometimes you will need to manually pair elements. In this case our PSD files follow the same conventions so you can press “Select All” then “Finish	 Storyboard Apply PSD Re-Import Advanced Options Replace project images with new PSD images Application Model Image View Select All Deselect All Pair Selected 1 Re-Import ... Off Screen ... bg_layer washing_contro 1_15_PM APRIL_3_2019 bg_image cycle_layer cycle_control next_control back_control normal_control ppress_control delicates_contrc wool_control handwash_contr heavyd_control New PSD Content Old Project Content WASHING 1:15 PM APRIL 3 2019 START Context: menu 2 < Back Next > Finish Cancel images/ (52)
Step 3 - *Simulate* You now have a new look and feel for your application and it was imported in only a few clicks. You’ve also retained all the work and behaviour you implemented. Simulate your project and test it out.	 storyboard_workspace_6_March_27 - washing_machine/wasl File Edit View Navigate Search Project Run Anima Application Model Navigator
Step 4 - *Optimize your assets* Now that you’ve reskinned your UI we should take a moment to optimize our assets so we take up less space on the RT-1060. This will also help speed up deployment. Open the metrics tab in the Properties Panel to get a view of our current resource consumption. Select the 1060 configuration.	 Quick Access Properties Outline Connecto Metrics Edit Overview Resource Details Filter: Configuration: 1060 Runtime: Unspecified Estimated Memory Size (RAM): 46.00 KB Component / Size 🟥 Model Elements (9) 31.03 KB 🟩 Fonts (1) 14.97 KB

Step 4 - *Condense project images*

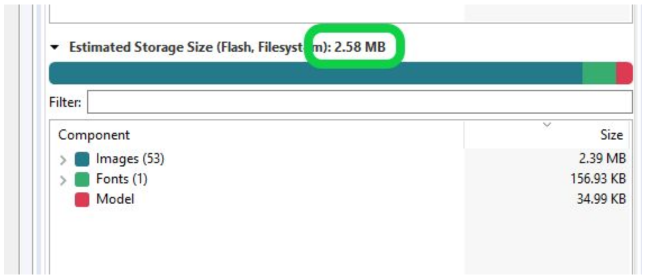
Now open your images tab in the Action Panel area. Notice that there are excess images from the original design. In the metrics view you can see how much space they take up in flash memory.



Select the resource clean up icon in the panel toolbar. This will open up a wizard that will scan your project for unreferenced images. Note: this tool will not catch images referenced in scripts. So you may need to remove some images that are suggested.



This application doesn't have any images referenced via script so we can press Next. The tool will also scan for unused fonts but won't find any in this project. Press finish and opt to delete the images. Notice all the space you've saved.



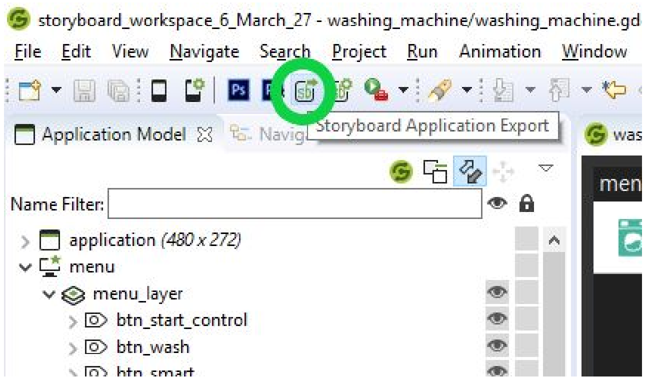
Estimated Storage Size (Flash, Filesystem): 2.58 MB

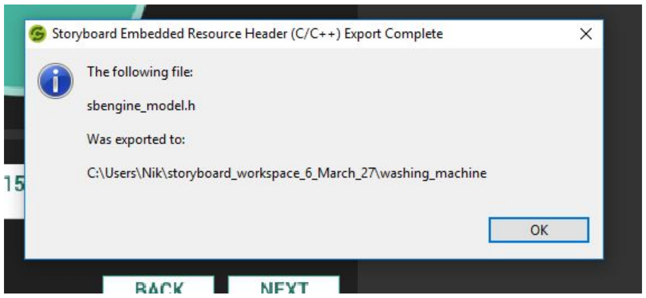
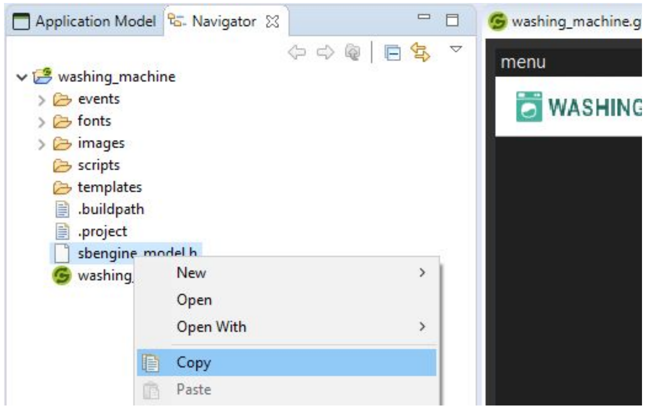
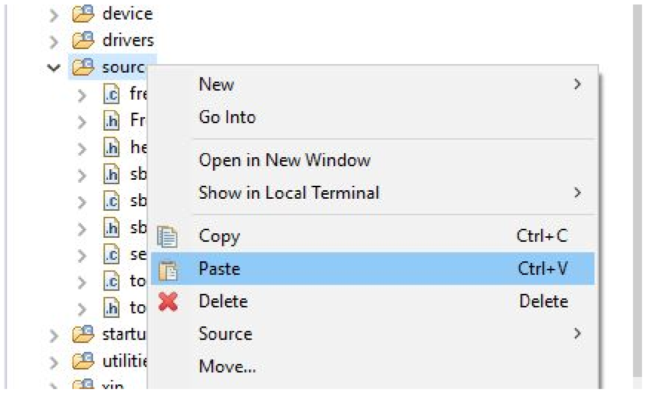
Filter:

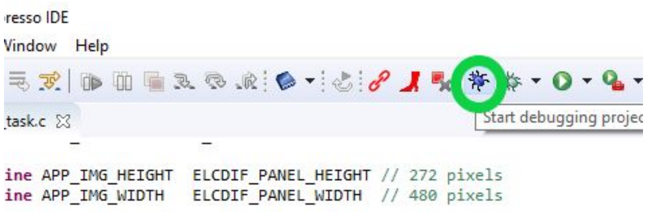
Component	Size
Images (53)	2.39 MB
Fonts (1)	156.93 KB
Model	34.99 KB

Step 4 - *Export the application*

Re-export your application for deployment to hardware. You can do this with a single click by pressing the Storyboard Application Export button in the toolbar. This will run the last export configuration.



You should see a message saying sbengine_model.h was successfully exported.	 Storyboard Embedded Resource Header (C/C++) Export Complete The following file: sbengine_model.h Was exported to: C:\Users\Nik\storyboard_workspace_6_March_27\washing_machine
Copy the header file from the Project directory.	 Application Model Navigator washing_machine.g washing_machine events fonts images scripts templates .buildpath .project sbengine_model.h washing New Open Open With Copy Paste menu WASHING
Step 4 - *Update the MCUXpresso project* Switch back to MCUXpresso and paste ***"sbengine_model.h"*** into the source directory. Overwrite the previous file.	 device drivers sourc New Go Into Open in New Window Show in Local Terminal Copy Ctrl+C Paste Ctrl+V Delete Delete Source Move... startu utilitie

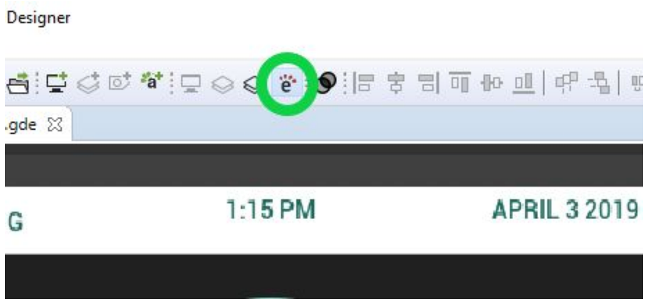
Clean and build then flash the project to the i.MX RT-1060.	 resso IDE
	Vindow Help
	task.c
	Start debugging projec
	`ine APP_IMG_HEIGHT   ELCDIF_PANEL_HEIGHT // 272 pixels`
	`ine APP_IMG_WIDTH    ELCDIF_PANEL_WIDTH  // 480 pixels`

Interacting with Hardware

Storyboard allows you to communicate with external tasks or processes through the use of the Storyboard IO plugin and library. This gives you the ability to send and receive events with data payloads to and from the UI. Let's create a custom event in Storyboard now.

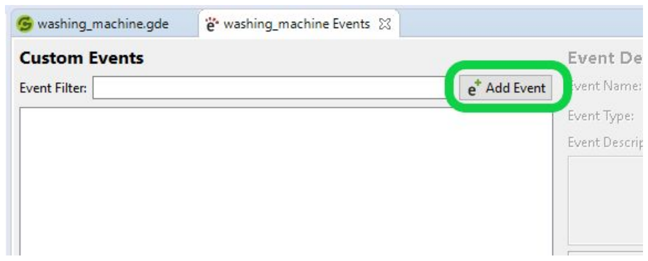
Step 1 - *Open the event editor*

Press the event editor button in the toolbar. This will open an editor that allows you to create and edit custom events.

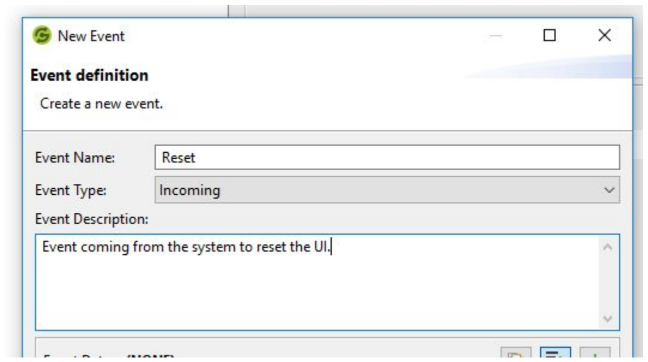


Step 2 - *Create a custom event.*

In the event editor panel press the "Add Event" button.



Give your new event a name of "Reset" and a description if you wish. Note: the name must be exactly as shown, case sensitivity matters.



Event definition

Create a new event.

Event Name: Reset

Event Type: Incoming

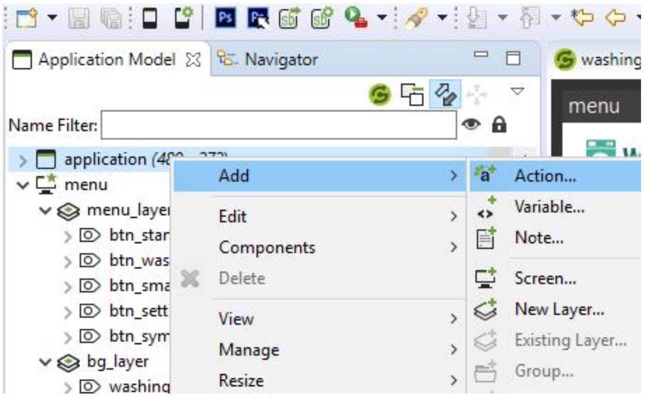
Event Description:

Event coming from the system to reset the UI.

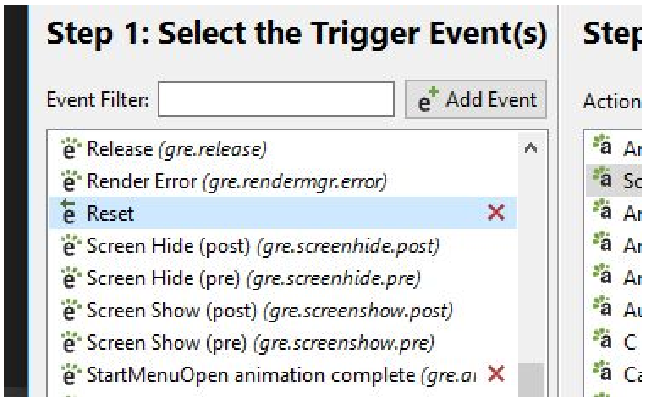
Select Finish. Save this event by pressing *CTRL + s* and close the event editor tab.

Step 3 - *Hook up the Reset Event.*

We can now use this new event to return the user to the main screen. Let's hook it up, right click on the Application node in the Application Model viewer and add an action. We choose the application because we want this event to trigger regardless of what screen we are on.

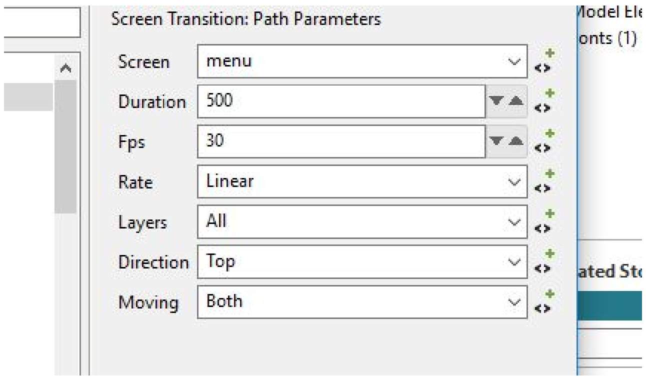


Find our new created“Reset” event in the event list. Select it and attach a “Screen Transition:Path” as the action.



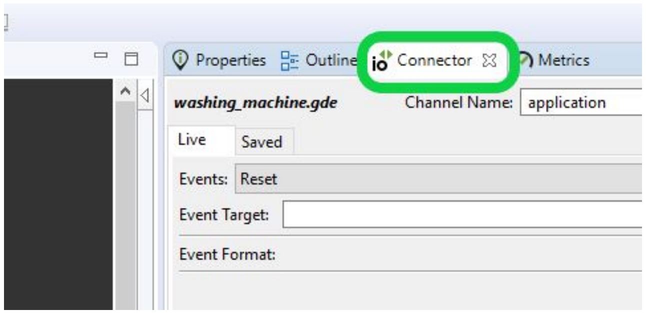
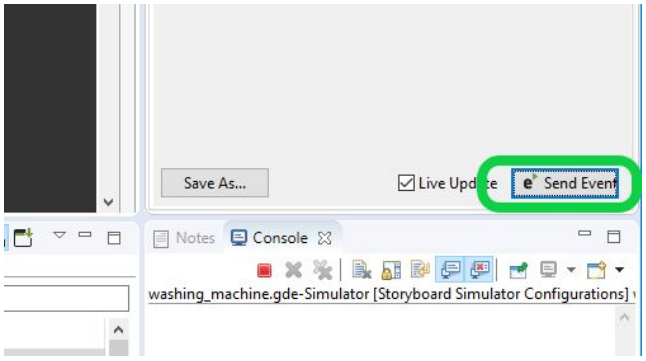
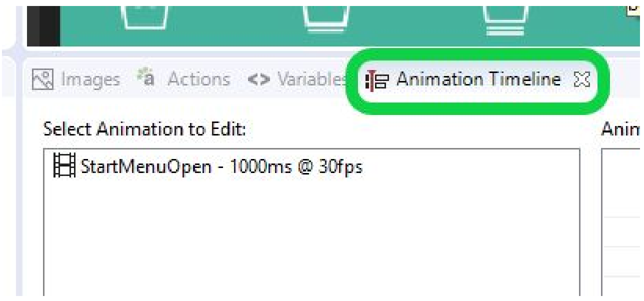
Release *(gre.release)*
Render Error *(gre.rendermgr.error)*
Reset
Screen Hide (post) *(gre.screenhide.post)*
Screen Hide (pre) *(gre.screenhide.pre)*
Screen Show (post) *(gre.screenshow.post)*
Screen Show (pre) *(gre.screenshow.pre)*
StartMenuOpen animation complete *(gre.a*

Set the properties of the transition to return to the menu screen using a direction of your choice. Press Finish.



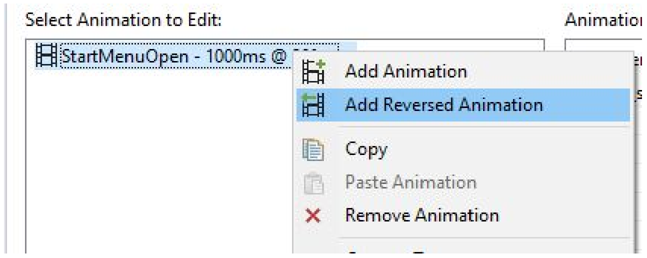
Screen Transition: Path Parameters

Screen	menu
Duration	500
Fps	30
Rate	Linear
Layers	All
Direction	Top
Moving	Both

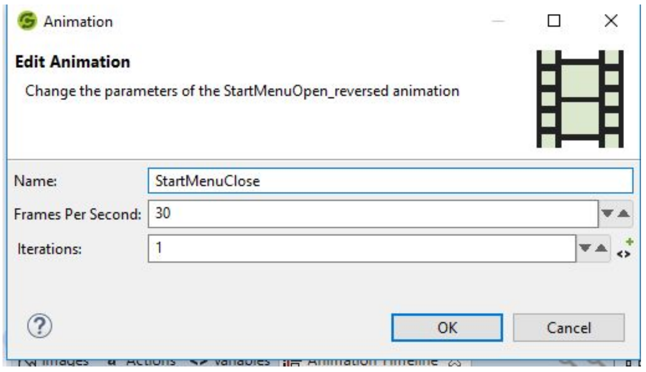
Step 4 - *Testing our event* Let's test our new functionality out. Fortunately we don't have to export and flash the application to hardware to test. We can simulate this event using a tool called the ***"Connector"***. Select the "Connector" tab in the Properties panel region. You will see the "Reset" event available in the "Events" drop down list. Simulate your application and navigate through a few screens. Then press the "Send Event" button in the Connector tab. *Note: You may need to resize your Storyboard Designer window to see your application and editor on the same screen.*	 Properties Outline Connector Metrics washing_machine.gde Channel Name: application Live Saved Events: Reset Event Target: Event Format:  washing_machine.gde-Simulator [Storyboard Simulator Configurations]
Step 5 - *Resetting the Main Menu animation* You probably noticed that although the Reset event returns us to the main screen. Our menu is still left expanded. Let's resolve this before exporting to hardware for the final time. Select the animation timeline panel.	 Images Actions Variables Animation Timeline Select Animation to Edit: StartMenuOpen - 1000ms @ 30fps

Step 6 - *Reverse our animation*

We can create the reverse of our animation. Right click on the "StartMenuOpen" animation and select "Add Reversed Animation"



Give it a more meaningful name like "StartMenuClose". Press OK.

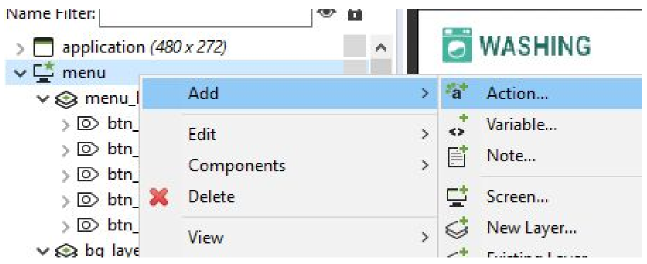


Change the parameters of the StartMenuOpen_reversed animation

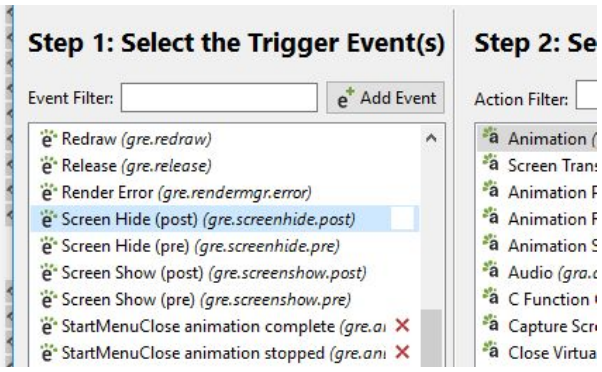
Name: StartMenuClose
Frames Per Second: 30
Iterations: 1

Step 7 - *Trigger the close animation.*

Right click on the "menu" screen in the Application model view and add and Action.



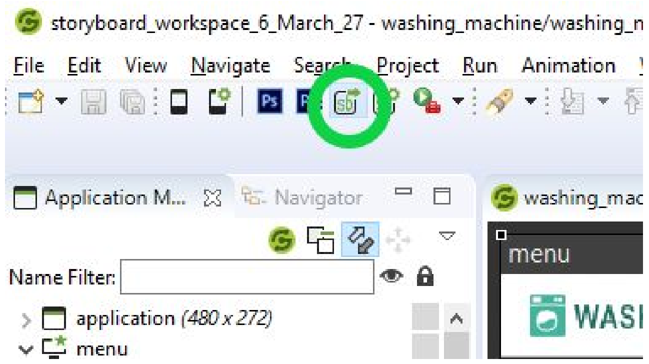
Select "Screen Hide (post)" in the "Trigger Event(s)" list. Select Animation as the action and select our new "StartMenuClose" animation. Press finish.


Step 1: Select the Trigger Event(s)
Step 2: Se
Event Filter:
Add Event
Action Filter:
Redraw (gre.redraw)
Release (gre.release)
Render Error (gre.rendermgr.error)
Screen Hide (post) (gre.screenhide.post)
Screen Hide (pre) (gre.screenhide.pre)
Screen Show (post) (gre.screenshow.post)
Screen Show (pre) (gre.screenshow.pre)
StartMenuClose animation complete (gre.a
StartMenuClose animation stopped (gre.an
Animation (
Screen Tran
Animation F
Animation F
Animation S
Audio (gra.
C Function
Capture Scr
Close Virtua

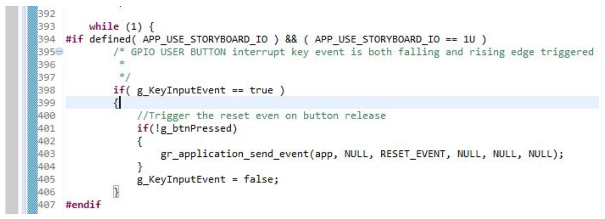
Simulate and test with the connector. Your Reset event should now bring you back to the main screen with the Start button.

Step 8 - *Export and Test on hardware.*

Export your application as a header file and drop it into your MCUXpresso Project.



The project already has code in place to generate a Reset event when SW8 is pressed. Line 403 of “*sbengine_task.c*” sends the event. The file “*sbio_events.h*” contains the event definition.



```
392
393     while (1) {
394 #if defined( APP_USE_STORYBOARD_IO ) && ( APP_USE_STORYBOARD_IO == 1U )
395         /* GPIO USER BUTTON interrupt key event is both falling and rising edge triggered
396          *
397          */
398         if( g_KeyInputEvent == true )
399         {
400             //Trigger the reset even on button release
401             if(!g_btnPressed)
402             {
403                 gr_application_send_event(app, NULL, RESET_EVENT, NULL, NULL, NULL);
404             }
405             g_KeyInputEvent = false;
406         }
407 #endif
```

Clean then build and flash your project to hardware and test.