USER GUIDE

EVE Screen Designer 4.19

Document Version: 1.8

Date: 29-07-2024

Contents

A. Preface

Purpose

This document describes the functionality and procedures involved in using the **E**VE **S**creen **D**esigner (ESD).

Intended Audience

The intended audience shall be any GUI application developer working with EVE products.

Document Reference

Document Name	Document Type	Document Format
FT81x Series Programmers Guide	Programming Guide	PDF
FT81x Datasheet	Datasheet	PDF
FT9xx Toolchain Installation Guide	Installation Guide	PDF
BT81X Series Programming Guide	Programming Guide	PDF
BT81X (815/6) Datasheet	Datasheet	PDF
BT817/8 Datasheet	Datasheet	PDF

Feedback

Every effort has been taken to ensure that the document is accurate and complete. However, any feedback on the document may be emailed to docufeedback@brtchip.com. For any additional technical support, refer to http://brtchip.com/contact-us/.

B. Overview

Introduction

EVE **S**creen **D**esigner (ESD) is the next generation of smart IDE for EVE, making EVE-based GUI development much easier to accomplish. This tool enables users to build one GUI application using a **visual programming**[1] method without needing to know any EVE-specific display list commands.

ESD provides a **WYSIWYG** ("**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et") environment for editing graphics, designing visual effects, and defining GUI application user logic, generating **ANSI C code** for the targeted hardware platform. Users can also choose to simulate the whole design to experience the UI before compiling and downloading the generated source code. Furthermore, ESD has the capability to work seamlessly with Bridgetek FT9XX Toolchain. Users can compile, link the generated source code with FT9XX Toolchain and upload it to the targeted platform without leaving ESD. From 4.13 onward, ESD has the same capability as FT90X platform to work with **Raspberry Pi Pico**[2]. From 4.19 onward, ESD also supports the **STM32F4** platform, similar to the FT90X platform.

A layout mechanism is introduced to manage widgets and pages in a more generic way. The layout mechanism will enable users to create more dynamic UI much easier than before. In addition, ESD 4.14 dramatically enhances the functionality of the logic nodes editor, layout editor and project browser, for better user experience.

Key Features

The following are some of the key features of **E**VE **S**creen **D**esigner:

- **WYSIWYG** GUI
- High level widgets
- No EVE display list knowledge required
- Widget based GUI construction
- Drag and drop widget to create screen layout
- Inter widget communication
- Screen logic creation without coding
- Simulation of screen logic and user touch input using mouse
- Building and downloading the generated "C" code (if FT9XX/STM32CubeIDE/Pico Toolchain is installed)

What's new in ESD 4.19

New Features:

- Improved widget functionality by enabling the selection of the corresponding radio button and checkbox widget through text-clicking.
- Added Scrolling Text Widget.
- Resolved the issue causing crashes in the ESD application when adding an ESD Input and selecting the 'esd_fonticon_t' type.

[1] https://en.wikipedia.org/wiki/Visual_programming_language
[2] https://www.raspberrypi.org/products/raspberry-pi-pico/

- Fixed the issue where the Text Input on the Property Browser was unable to input the '/' and '\' characters.
- Removed the redefined names and unnecessary definition names from some example projects .esd file.
- Fixed the Font Icon search function not recognizing case sensitivity.
- Replaced the graphics definitions EVE_GRAPHICS_VM816C50A and EVE_GRAPHICS_VM816CU50A with EVE_GRAPHICS_VM816C in the Target.conf file.
- Added Image Slideshow widget.
- Added QRCode to basic example projects.
- Moved .esdm files to a separate folder in the project folder.
- Added options for Gauge Widget: OPT_NOBACK , OPT_NOTICKS, OPT_NOPOINTER.
- Improved example project "AdvancedWidgetsDemo".
- Added support for STM32 platform.
- Resolved the issue preventing renaming of the 'Usr_Page' name in the project browser when creating a new project.
- Added the EVE flash memory feature into STM32 board.
- Added an example project for the STM32F4 platform named CoverFlow.
- Created Dynamic Array Input.
- Added e-Bike example project.

Known Issues & Limitations

The following are some known issues and limitations of ESD:

- Only the BT81X/BT88x/FT81X series **EVE** is supported. FT80X Series EVE is **NOT** supported.

- The C code editor in ESD does not support the following features:

    - Code completion
    - Column mode
    - Line bookmarking
    - Error and Warning Marks
    - Structure Visualizer
    - Word Wrap
    - Expand and collapse blocks of code.
    - Highlighting of multiple selected text portions

- If the project file path is too long (i.e., more than 512 bytes), ESD may have a problem opening it. The typical error message is shown below:

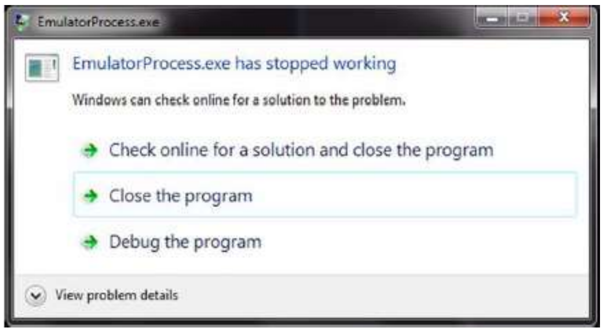
  ```
  "Unable to generate output files, check directory permission at: C:\Users\xxxx.xxxx\Project\ESD\......"
  ```

  Where

  `"C:\Users\xxxx.xxxx\Project\ESD\......"` refers to the project folder.

- Logic node editor background goes white after system hibernates.

- In some unusual cases, users may encounter a dialog box (shown below) which will not affect the functionality. In this case, just ignore the dialog box by closing it.



EmulatorProcess.exe

EmulatorProcess.exe has stopped working

Windows can check online for a solution to the problem.

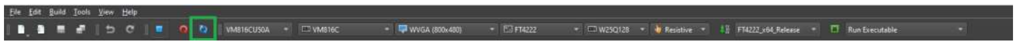
Check online for a solution and close the program

Close the program

Debug the program

View problem details

- In few cases, users may need to click **[Recompile]** at the toolbar to update the frozen simulation result.



- The Gameduino 3X dazzler platform does not currently support game controllers, which means that users are unable to interact with ESD projects. Additionally, the use of an SD card is necessary.

- Extended font is not supported in FT81X chips, but no error report will be generated if choose it in ESD

- When utilizing the ESD Circular Gradient Slider, certain scenarios may cause lines to appear on the Screen Layout Editor (Emulator), as depicted in Figure 1. However, no lines are ever displayed on the hardware side, as illustrated in Figure 2.

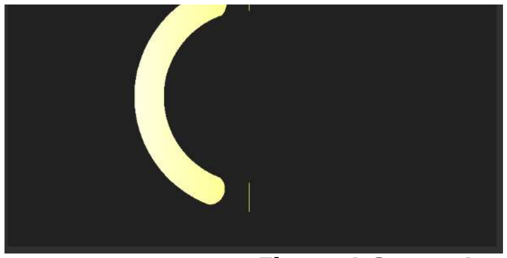


Figure 1 Screen Layout Editor

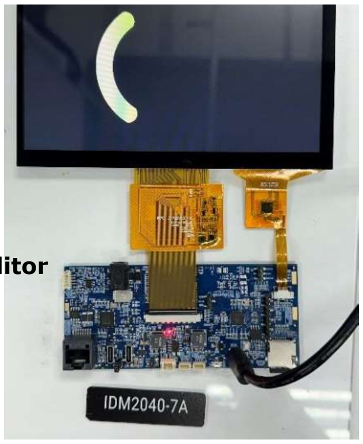


Figure 2 Hardware IDM2040-7A

Terms & Abbreviations

Terms/Abbreviations	Description
Actor	One type of logic node which is regularly updated but without visual appearance
DLL	Dynamic Link Library is a collection of small programs, any of which can be called when needed by a larger program that is running in the computer
ESD	EVE Screen Designer
EAB	EVE Asset Builder
EVE Emulator	Bridgetek behaviour-modelling software for EVE Series chip
HAL	Hardware Abstraction Layer is a software subsystem providing hardware abstraction.
IDE	Integrated Development Environment
Layout type widget	One type of widget which has no visual appearance rendered by EVE, but manages the associated widget
Logic Node	The node expressing certain logic. (Also referred to simply as "node" in this document)
Logic Node Editor	The place where you create logic by connecting the logic node
Page	One single screen in design
Simulation	Preview the project or page by running the generated C code on PC
Widget	One type of logic node which has visual appearance rendered by EVE

Credits

Open-Source Software

- Qt: https://doc.qt.io/qt-6/licensing.html under LGPL.
- TinyCC: http://bellard.org/tcc/ and http://repo.or.cz/tinycc.git under LGPL.
- Errorlist module: https://github.com/kaetemi/errorlist under MIT license
- QScintilla: part of PyQt under PyQt commercial license https://riverbankcomputing.com/commercial/license-faq

Icons Copyright

Some of the icons are from:
http://p.yusukekamiyamane.com/

They are used in compliance with the Creative Commons Attribution 3.0 License.

C. Setup & Installation

System Requirements

To install ESD application, ensure that your system meets the requirements recommended below:

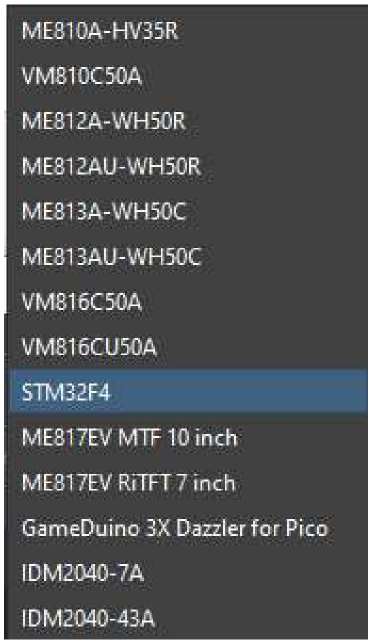
- ✓ Ideally Windows 10; alternatively, Windows 8 or 7 with the latest windows updates
- ✓ 1.6GHz or faster processor
- ✓ 1GB of RAM (1.5GB if running on a virtual machine)
- ✓ Multi-Core CPU is highly recommended
- ✓ At least 512MB hard disk space
- ✓ Display resolution 1080x800 pixels or higher
- ✓ "Write" permission to the installation folder

Hardware Requirements

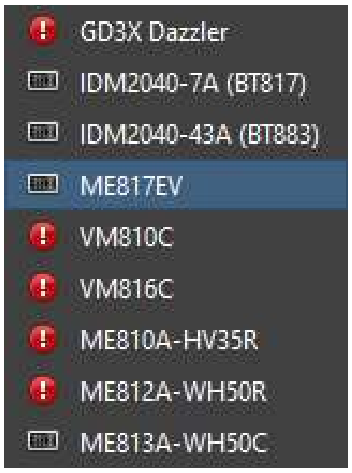
The exported C source code from ESD is targeted at **EVE based** platform, which typically consists of an Eve module and a host (MCU or PC based module).

The supported platforms are listed as below:

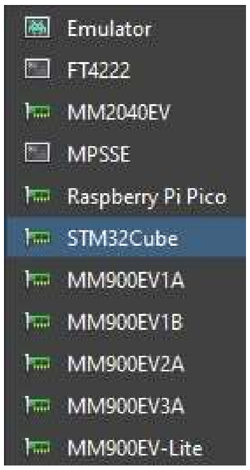
[Build Target]:


ME810A-HV35R
VM810C50A
ME812A-WH50R
ME812AU-WH50R
ME813A-WH50C
ME813AU-WH50C
VM816C50A
VM816CU50A
STM32F4
ME817EV MTF 10 inch
ME817EV RiTFT 7 inch
GameDuino 3X Dazzler for Pico
IDM2040-7A
IDM2040-43A

[Eve Platform]


GD3X Dazzler
IDM2040-7A (BT817)
IDM2040-43A (BT883)
ME817EV
VM810C
VM816C
ME810A-HV35R
ME812A-WH50R
ME813A-WH50C

[Host Platform]


Emulator
FT4222
MM2040EV
MPSSE
Raspberry Pi Pico
STM32Cube
MM900EV1A
MM900EV1B
MM900EV2A
MM900EV3A
MM900EV-Lite

Platform Name (Build Target)	Compatible EVE Module	Compatible MCU Module Name	Compatible PC based Module
ME810A-HV35R	ME810A-HV35R[6](320X480)	MM900EV[1] MM2040EV[2]	N.A.
ME812A-WH50R	ME812A-WH50R[6] (800x480)		
ME812AU-WH50R	ME812AU-WH50R[6] (800x480)	N.A.	FT4222H[8] (libFT4222.dll)
ME813A-WH50C	ME813A-WH50C[6] (800x480)	MM900EV[1] MM2040EV[2]	N.A.
ME813AU-WH50C	ME813AU-WH50C[6] (800x480)	N.A.	FT4222H[8] (libFT4222.dll)
VM810C50A	VM810C50A[6] (800x480)	N.A.	MPSSE[7] (libMPSSE.dll)
VM816C50A	VM816C[4] (800x480)	N.A.	MPSSE[7] (libMPSSE.dll)
VM816CU50A	VM816CU[4] (800x480)	N.A.	FT4222H[8] (libFT4222.dll)
ME817EV-WH70C	ME817EV[5] (1024X600)	MM900EV[1] MM2040EV[2]	FT4222H[8] (libFT4222.dll)
ME817EV-WH10C	ME817EV[5] (1280X800)		FT4222H[8] (libFT4222.dll)
Gameduino 3X Dazzler for Pico	Gameduino 3X Dazzler (HDMI output 1280x720)	Raspberry Pi Pico[3]	N.A.
IDM2040-7A	IDM2040-7A[9] (800x480)	Raspberry Pi Pico[3]	N.A.
IDM2040-43A	IDM2040-43A (480x272)	Raspberry Pi Pico[3]	N.A.
STM32F4	ME817EV[5] (800X480)	STM32	N.A.

1. MM900EV development modules:
    a. MM900EVxA – End of Life
    b. MM900EVxB - https://brtchip.com/product/mm900ev1b/
    c. MM900EV-Lite - https://brtchip.com/product/mm900ev-lite/

2. MM2040EV is a Raspberry Pi Pico adaptor board: https://brtchip.com/product/mm2040ev/

3. Raspberry Pi Pico is from: https://www.raspberrypi.com/products/raspberry-pi-pico/

4. VM816C is the EVE 3 development module: https://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_VM816C.pdf

5. ME817EV is the EVE 4 development module: https://brtchip.com/product-category/products/modules/eve4/

6. The EVE 2 development modules: check https://brtchip.com/product-category/products/modules/eve2/ for details

7. MPSSE stands for "Multi-Protocol synchronous serial engine". The following Bridgetek or FTDI devices support it:

    a. VA800A-SPI board
    b. C232HM-EDHSL-0(5V) cable
    c. C232HM-DDHSL-0(3.3V) cable

8. FT4222 refers to the bridge chip which converts from USB Hi-Speed transportation to Multi-Channel Serial SPI protocol. See: https://ftdichip.com/products/ft4222h/

9. IDM2040 is an embedded application platform from Bridgetek which consists of:

   a. RP 2040 as MCU: https://datasheets.raspberrypi.org/rp2040/rp2040-datasheet.pdf
   b. BT817 as GPU: BT817/8 Advanced Embedded Video Engine (brtchip.com)

10. Gameduino 3X Dazzler is an open-source project originally for Arduino: https://www.crowdsupply.com/excamera/gameduino-3x-dazzler
    ESD supports one variety – Gameduino 3X Dazzler for Pico, which consists of:

    - Raspberry Pi Pico as MCU module
    - BT815 as GPU

    You can get more details from:
    https://excamera.com/sphinx/store.html#gameduino-3x-dazzler-for-pico-39

⚠ To ensure proper functioning of the hardware platform (FT90X modules or Raspberry Pi Pico), users need to make sure that an SD card is inserted into the SD slot. This is because ESD is programmed to identify the presence of an SD card, and if it fails to do so, the application may not work correctly.

Dependencies / Pre-Requisites

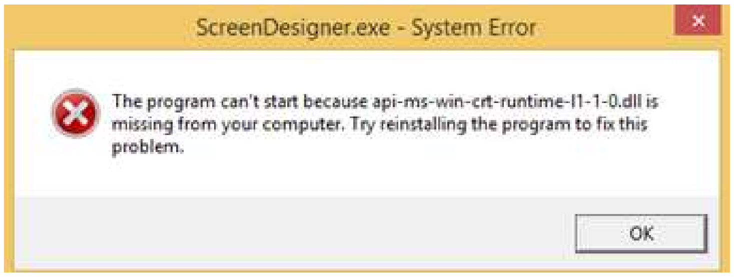
- **Visual C++ Redistributable for Visual Studio 2015**

  If the PC does not have Microsoft Visual Studio 2015 installed, Visual C++ Redistributable is required. Users can download this from:

  https://www.microsoft.com/en-sg/download/details.aspx?id=48145

- **Windows 10 Universe C Runtime**

  ESD has run-time dependency on Windows 10 Universe C Runtime (CRT). You may download it from https://www.microsoft.com/en-us/download/details.aspx?id=48234 and install on your PC should the following problem be encountered:


ScreenDesigner.exe - System Error

The program can't start because api-ms-win-crt-runtime-l1-1-0.dll is missing from your computer. Try reinstalling the program to fix this problem.

Figure 3 - Screen Designer - System Error

- **FT9XX Tool Chain Version 2.6.0 or later**

  To compile and build projects, the FT9XX Tool Chain 2.6.0 or later version must be installed on the PC. It is downloadable from https://brtchip.com/ft9xx-toolchain/ .

  Please ensure that the Tool Chain executable path is defined by the system **PATH** environment variable.

Users are advised to check the known issues and limitations (of FT9XX Toolchain) while building the ESD project with FT9XX Toolchain. The respective FT9XX Toolchain package version release note contains the list of known issues and limitations.

For ESD, we recommend users to install FT9XX Tool Chain version 2.6.0 or later version for the best results.

- **Raspberry Pi Pico Tool Chain**

  To compile and build Pico projects, several tool chains must be installed on the PC. They are: Pico-SDK, python 3.8, GNU Embedded Toolchain for Arm and Cmake. They can be downloaded/cloned from the following:
  - Pico-SDK - https://github.com/raspberrypi/pico-sdk
  - Python 3.8 - https://www.python.org/downloads/
  - GNU Embedded Toolchain - https://developer.arm.com/downloads/-/gnu-rm
  - CMake - https://cmake.org/download/

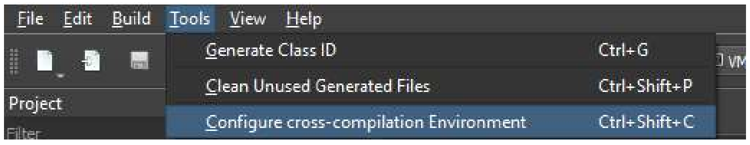
- **STM32Cube Tool Chain**

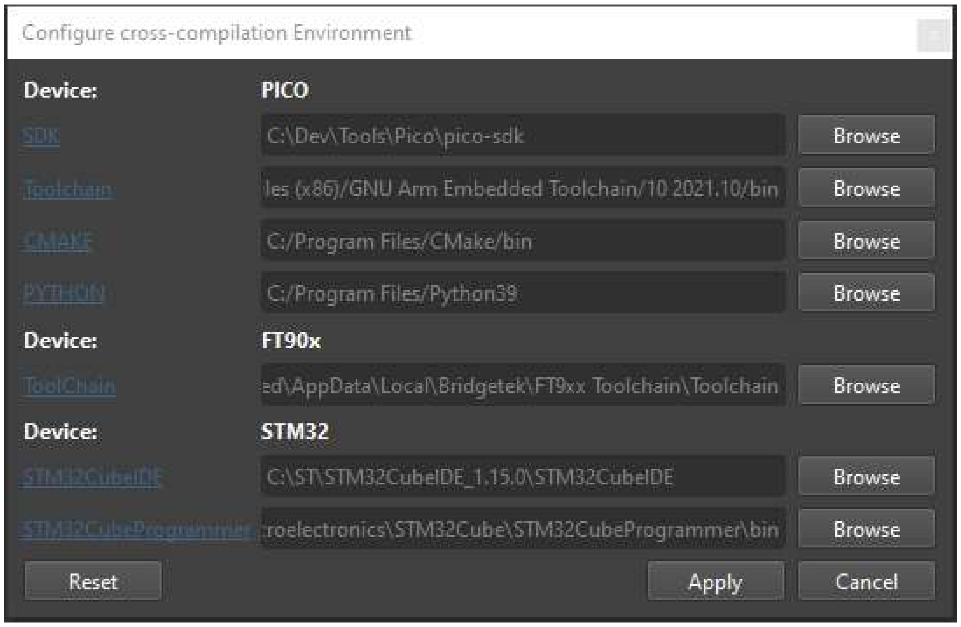
  To compile and build STM32 projects, you need to install several toolchains on your PC. These include STM32CubeIDE and STM32CubeProgrammer, which can be downloaded from the following links:

  - STM32CubeIDE:https://www.st.com/en/development-tools/stm32cubeide.html

  - STM32CubeProgrammer:https://www.st.com/en/development-tools/stm32cubeprog.html

ESD will check system environment variables to find these tool chains, and it also provides a user dialog to configure the tool chain path under `Tools` menu:


File
Edit
Build
Tools
View
Help
Generate Class ID Ctrl+G
Clean Unused Generated Files Ctrl+Shift+P
Configure cross-compilation Environment Ctrl+Shift+C
Project
Filter



Configure cross-compilation Environment

Device:	PICO	
SDK	C:\Dev\Tools\Pico\pico-sdk	Browse
Toolchain	les (x86)/GNU Arm Embedded Toolchain/10 2021.10/bin	Browse
CMAKE	C:/Program Files/CMake/bin	Browse
PYTHON	C:/Program Files/Python39	Browse
Device:	FT90x	
ToolChain	ed\AppData\Local\Bridgetek\FT9xx Toolchain\Toolchain	Browse
Device:	STM32	
STM32CubeIDE	C:\ST\STM32CubeIDE_1.15.0\STM32CubeIDE	Browse
STM32CubeProgrammer	roelectronics\STM32Cube\STM32CubeProgrammer\bin	Browse

Installing ESD Package

The following steps will guide you through the ESD *Setup/Installation* process.

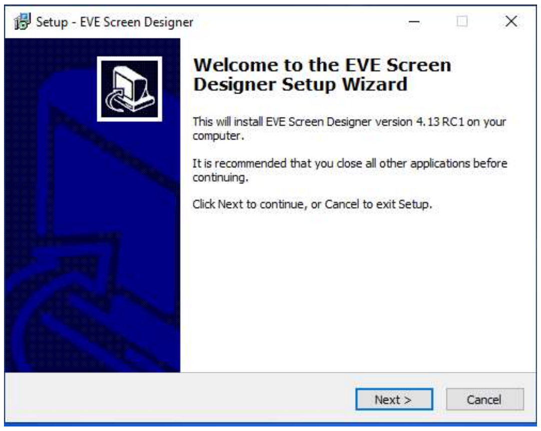
i. Download the package from https://brtchip.com/esd/.

ii. When prompted with a download dialog box. Click **[Save]**.

iii. Navigate to the folder under which the package files are downloaded.

iv. Extract the zip file contents. Double click on the executable file – **EVE Screen Designer.exe**

The EVE Screen Designer Setup Wizard is displayed along with a Welcome message. Click **[Next]**.



Setup - EVE Screen Designer

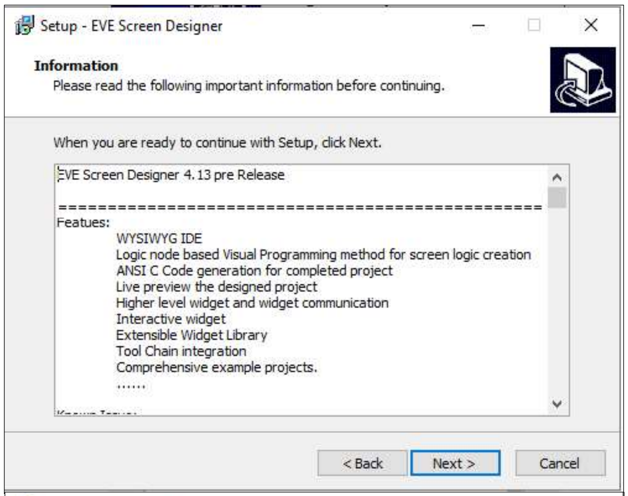
Welcome to the EVE Screen Designer Setup Wizard

This will install EVE Screen Designer version 4.13 RC1 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

v. End User information window is displayed. Click **[Next]**.



Setup - EVE Screen Designer

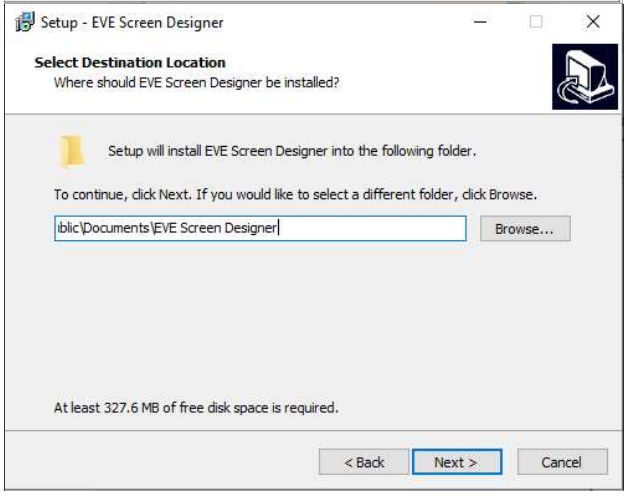
Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

```
EVE Screen Designer 4.13 pre Release

================================================
Featues:
        WYSIWYG IDE
        Logic node based Visual Programming method for screen logic creation
        ANSI C Code generation for completed project
        Live preview the designed project
        Higher level widget and widget communication
        Interactive widget
        Extensible Widget Library
        Tool Chain integration
        Comprehensive example projects.
        ......
```

vi. Select the "Destination Folder" for installing the files. Accept the default folder or click **Browse** to select a different destination folder. To confirm the destination folder and continue, click **[Next]**.



Setup - EVE Screen Designer

Select Destination Location

Where should EVE Screen Designer be installed?

Setup will install EVE Screen Designer into the following folder.

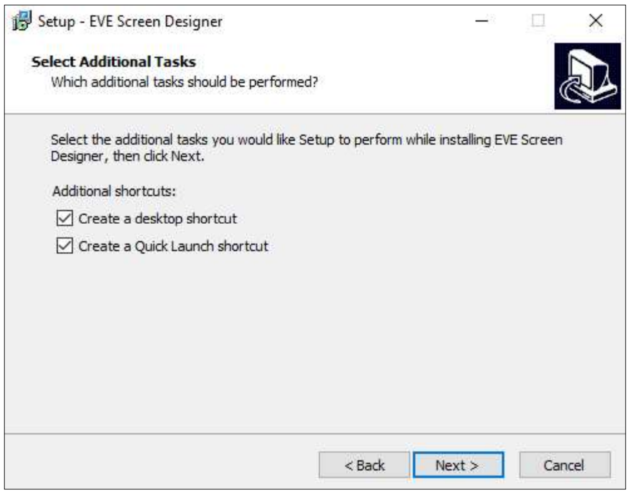
To continue, click Next. If you would like to select a different folder, click Browse.

ıblic\Documents\EVE Screen Designer

At least 327.6 MB of free disk space is required.

vii. In the **Select Additional Tasks** window, check **"Create a desktop / Create Quick Launch shortcut"** boxes, to have the ESD icon and Quick Launch shortcut displayed on the desktop if required. Click **[Next]** to prepare for the installation.

The initial setup is completed, and the application is ready to be installed.



Setup - EVE Screen Designer

Select Additional Tasks

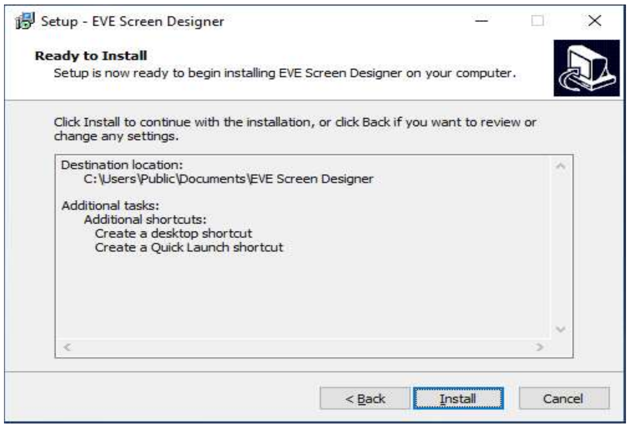
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing EVE Screen Designer, then click Next.

Additional shortcuts:

- [✓] Create a desktop shortcut
- [✓] Create a Quick Launch shortcut

viii. Click **[Install]** to start the installation.


Setup - EVE Screen Designer

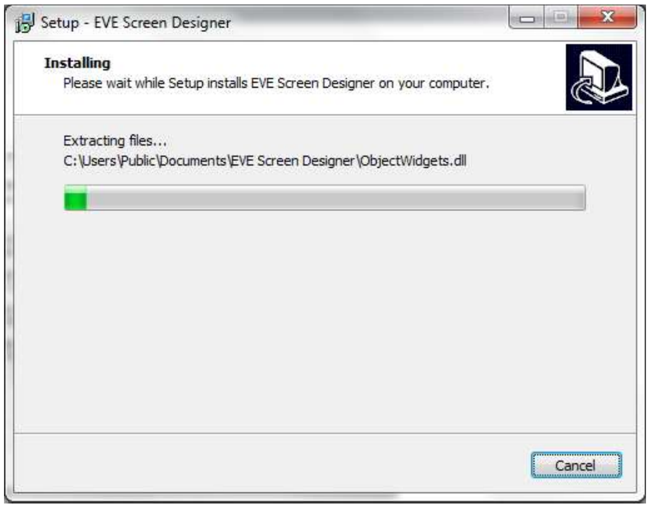
Ready to Install
Setup is now ready to begin installing EVE Screen Designer on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Users\Public\Documents\EVE Screen Designer

Additional tasks:
Additional shortcuts:
Create a desktop shortcut
Create a Quick Launch shortcut

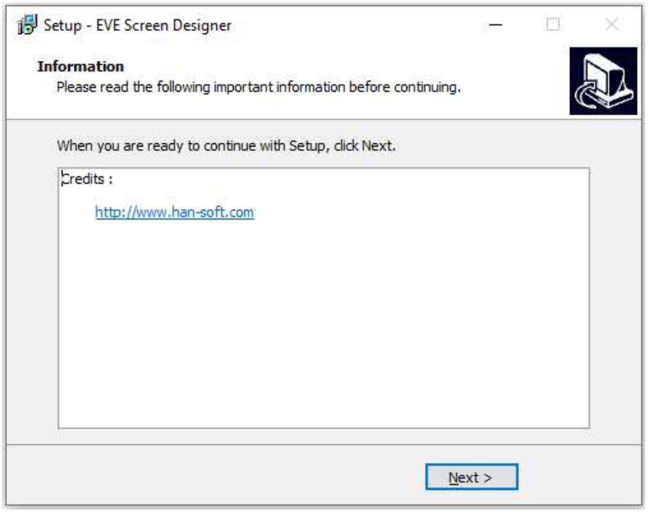
ix. The progress bar indicates that the installation is in progress.


Setup - EVE Screen Designer

Installing
Please wait while Setup installs EVE Screen Designer on your computer.

Extracting files...
C:\Users\Public\Documents\EVE Screen Designer\ObjectWidgets.dll

x. In the **Information** page, click **[Next]** to proceed ahead.

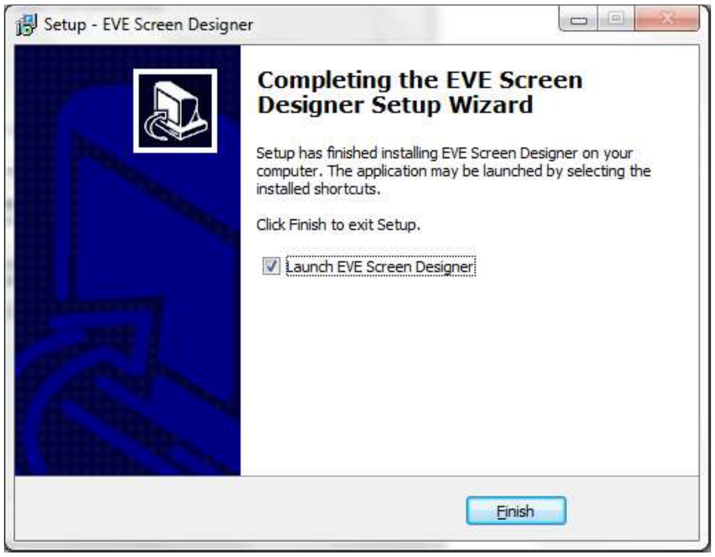

Setup - EVE Screen Designer

Information
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

Credits :
http://www.han-soft.com

xi. Upon successful installation, click **[Finish]**.



Completing the EVE Screen Designer Setup Wizard

Setup has finished installing EVE Screen Designer on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

[✓] Launch EVE Screen Designer

Installation Folder

The following table provides a list of folders that can be found under the installation path:

Folder Name	Description	Permission
Examples	The example projects created by ESD	Read/Write
Imageformats	Qt run-time DLLs for image format supporting	Read-Only
Styles	Qt run-time DLLs for UI styles	Read-Only
Libraries	Widget library, application framework and Hardware abstraction layer	Read-Only
Log	Stores the runtime logs for debug purpose	Read/Write
Manual	Contains this document	Read-Only
Platforms	Qt platform specific run-time DLLs	Read-Only
Settings	Configuration files for third-party utilities and tool chains as well as ESD settings. The files are in XML format.	Read-Only. Reserved for advanced users to change.
Templates	The template files used by ESD	Read-Only
TinyCC	TinyCC run-time used for ESD simulation purpose.	Read-Only
Tools	The utilities used in ESD for bitmap and font data generation purpose	Read-Only

Table 1 - Installation Folder

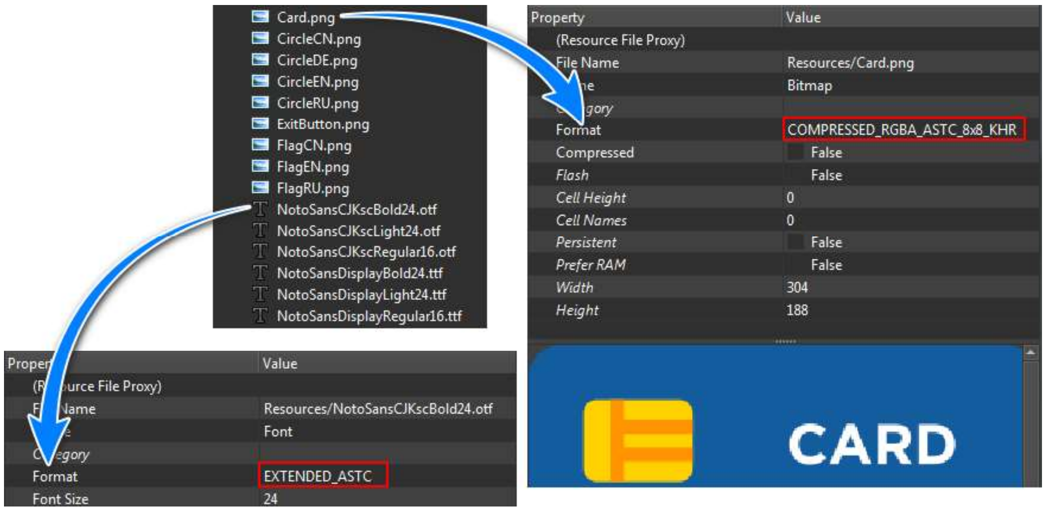
D. Working with ESD

Major features

Support full BT81X features

Users can refer to EVChargePoint examples in the path *<ESD Folder>/Examples/Advanced/EVChargePoint*. Opening it may take more than 5 minutes because of ASTC, bitmap and font conversion. This example contains all of the new BT81X features. Here are the details:

1. Apply ASTC Bitmap Format for images and fonts.



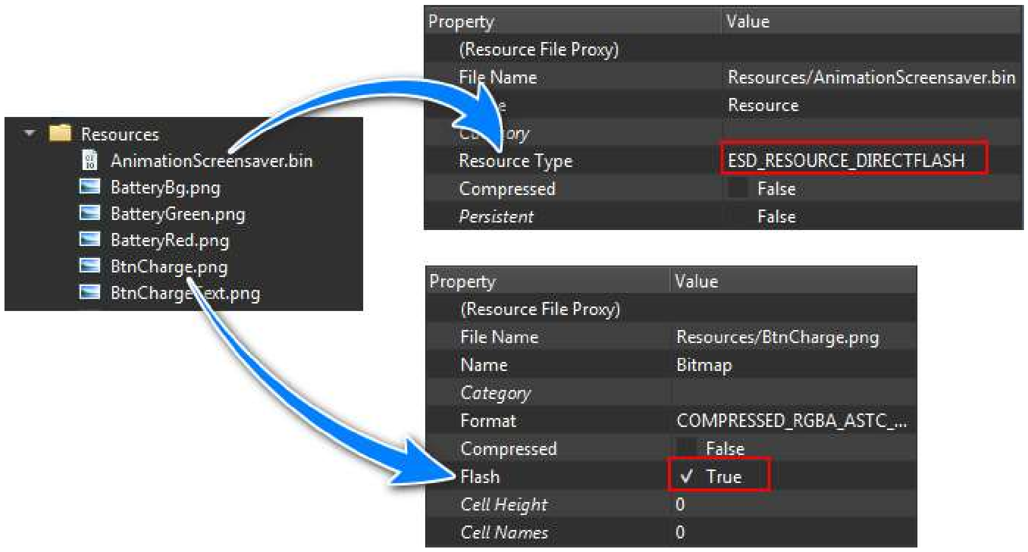
Card.png
CircleCN.png
CircleDE.png
CircleEN.png
CircleRU.png
ExitButton.png
FlagCN.png
FlagEN.png
FlagRU.png
NotoSansCJKscBold24.otf
NotoSansCJKscLight24.otf
NotoSansCJKscRegular16.otf
NotoSansDisplayBold24.ttf
NotoSansDisplayLight24.ttf
NotoSansDisplayRegular16.ttf

Property	Value
(Resource File Proxy)	
File Name	Resources/Card.png
ne	Bitmap
gory	
Format	COMPRESSED_RGBA_ASTC_8x8_KHR
Compressed	False
Flash	False
Cell Height	0
Cell Names	0
Persistent	False
Prefer RAM	False
Width	304
Height	188

Proper	Value
(R urce File Proxy)	
F Name	Resources/NotoSansCJKscBold24.otf
e	Font
C egory	
Format	EXTENDED_ASTC
Font Size	24

Figure 4 - ASTC applied for images and fonts

2. Resources are stored and loaded from flash.



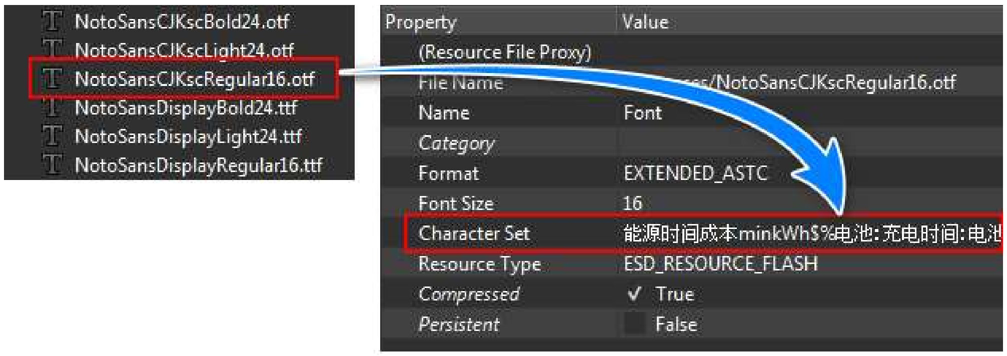
Resources
AnimationScreensaver.bin
BatteryBg.png
BatteryGreen.png
BatteryRed.png
BtnCharge.png
BtnCharge ext.png

Property	Value
(Resource File Proxy)	
File Name	Resources/AnimationScreensaver.bin
e	Resource
Ca ory	
Resource Type	ESD_RESOURCE_DIRECTFLASH
Compressed	False
Persistent	False

Property	Value
(Resource File Proxy)	
File Name	Resources/BtnCharge.png
Name	Bitmap
Category	
Format	COMPRESSED_RGBA_ASTC_...
Compressed	False
Flash	✓ True
Cell Height	0
Cell Names	0

Figure 5 - Load resources from flash

3. Use Unicode font with ASTC format and predefined character set


NotoSansCJKscBold24.otf
NotoSansCJKscLight24.otf
NotoSansCJKscRegular16.otf
NotoSansDisplayBold24.ttf
NotoSansDisplayLight24.ttf
NotoSansDisplayRegular16.ttf

Property	Value
(Resource File Proxy)	
File Name	…/NotoSansCJKscRegular16.otf
Name	Font
Category	
Format	EXTENDED_ASTC
Font Size	16
Character Set	能源时间成本minkWh$%电池:充电时间:电池
Resource Type	ESD_RESOURCE_FLASH
Compressed	✓ True
Persistent	False

Figure 6 - Use Unicode font with pre-defined character set

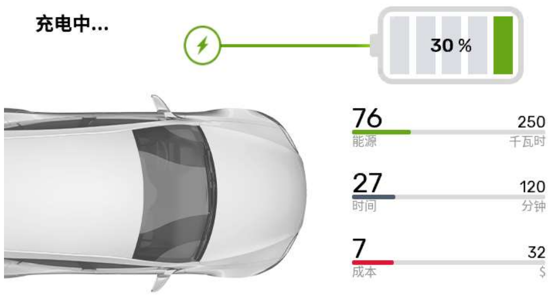
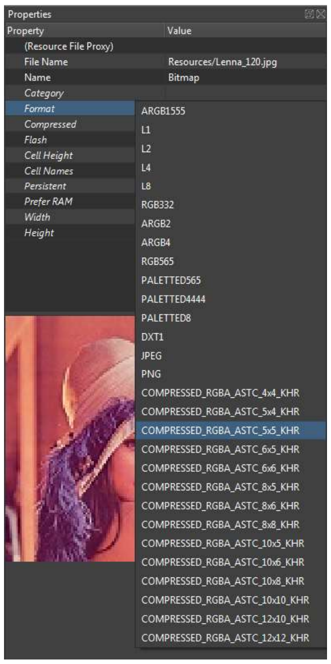


Figure 7 - Chinese font used in application

Support for ASTC bitmap format

ESD provides the option to convert images to the ASTC bitmap format, resulting in significant storage space savings without compromising image quality.



Properties

Property	Value
(Resource File Proxy)	
File Name	Resources/Lenna_120.jpg
Name	Bitmap
Category	
Format	
Compressed	
Flash	
Cell Height	
Cell Names	
Persistent	
Prefer RAM	
Width	
Height	

ARGB1555
L1
L2
L4
L8
RGB332
ARGB2
ARGB4
RGB565
PALETTED565
PALETTED4444
PALETTED8
DXT1
JPEG
PNG
COMPRESSED_RGBA_ASTC_4x4_KHR
COMPRESSED_RGBA_ASTC_5x4_KHR
COMPRESSED_RGBA_ASTC_5x5_KHR
COMPRESSED_RGBA_ASTC_6x5_KHR
COMPRESSED_RGBA_ASTC_6x6_KHR
COMPRESSED_RGBA_ASTC_8x5_KHR
COMPRESSED_RGBA_ASTC_8x6_KHR
COMPRESSED_RGBA_ASTC_8x8_KHR
COMPRESSED_RGBA_ASTC_10x5_KHR
COMPRESSED_RGBA_ASTC_10x6_KHR
COMPRESSED_RGBA_ASTC_10x8_KHR
COMPRESSED_RGBA_ASTC_10x10_KHR
COMPRESSED_RGBA_ASTC_12x10_KHR
COMPRESSED_RGBA_ASTC_12x12_KHR

Figure 8 - ASTC Format

Support DXT1 bitmap format

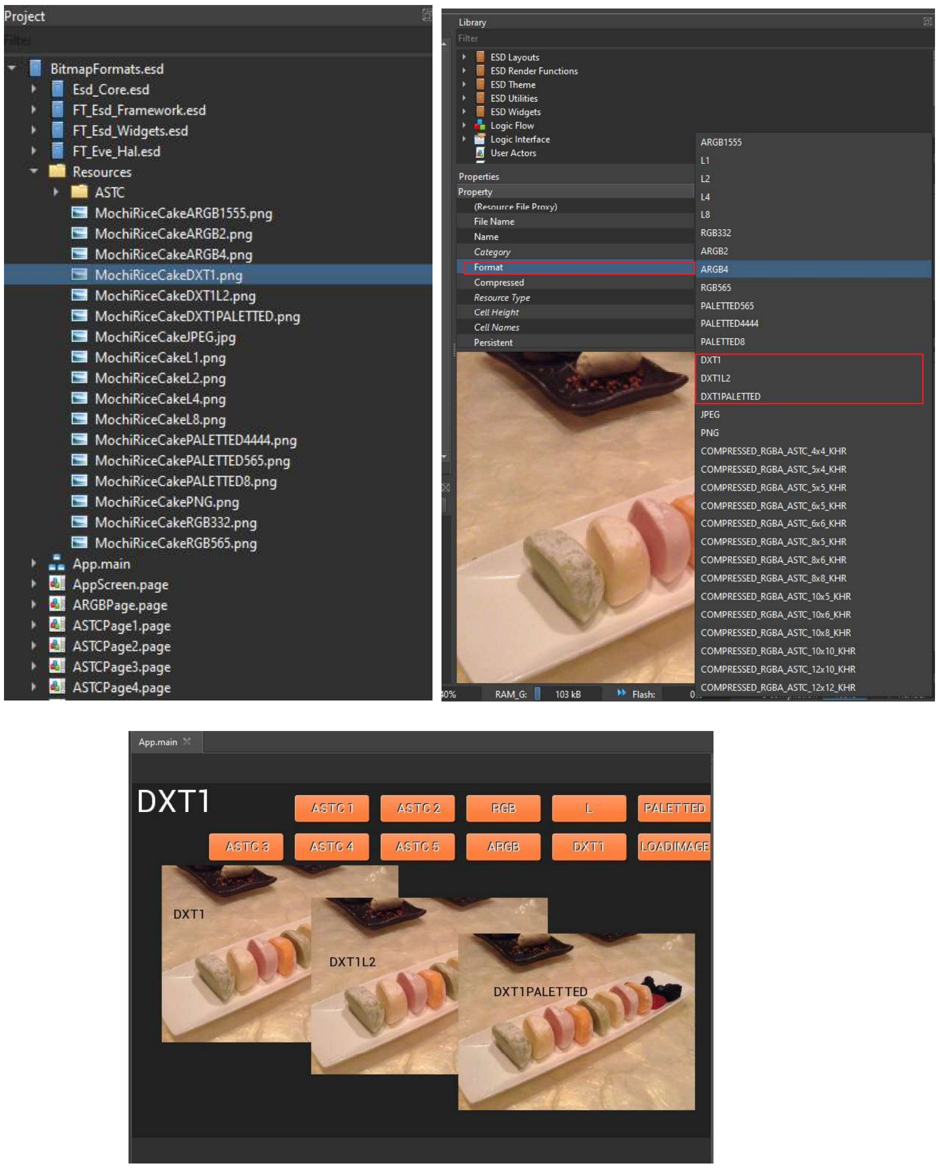
DXT1 (also known as Block Compression 1 or BC1) is the smallest variation of S3TC, storing 16 input pixels in 64 bits of output, consisting of two 16-bit RGB 5:6:5 color values(c0,c1), and a 4×4 two-bit lookup table. **ESD** supports the following 3 varieties of DXT1 format bitmap.

- **DXT1**: the 4x4 two-bit look up table is expressed in **L1** format, while c0,c1 is expressed in **RGB565** format.

- **DXT1L2**: the 4x4 two-bit look up table is expressed in **L2** format. while c0,c1 is expressed in **RGB565** format.

- **DXT1PALETTED**: the 4x4 two-bit look up table is expressed in **L2** format. while (c0,c1) is expressed in PALETTED565 format.
  By replacing the RGB565 layers with **PALETTED565** layers, the size is further reduced to 2 bytes per block total for the color layers, which results in a 3 bits per pixel plus the color palette of up to 512 bytes.

Just as with other bitmap format, once the image is added into project, users can select the "Format" property of input image and set it to the DXT1 related format. The example project "BitmapFormats" shows the effect of DXT1 format bitmap, by comparing with other bitmap formats which Eve supported.



Project

- BitmapFormats.esd
  - Esd_Core.esd
  - FT_Esd_Framework.esd
  - FT_Esd_Widgets.esd
  - FT_Eve_Hal.esd
  - Resources
    - ASTC
    - MochiRiceCakeARGB1555.png
    - MochiRiceCakeARGB2.png
    - MochiRiceCakeARGB4.png
    - MochiRiceCakeDXT1.png
    - MochiRiceCakeDXT1L2.png
    - MochiRiceCakeDXT1PALETTED.png
    - MochiRiceCakeJPEG.jpg
    - MochiRiceCakeL1.png
    - MochiRiceCakeL2.png
    - MochiRiceCakeL4.png
    - MochiRiceCakeL8.png
    - MochiRiceCakePALETTED4444.png
    - MochiRiceCakePALETTED565.png
    - MochiRiceCakePALETTED8.png
    - MochiRiceCakePNG.png
    - MochiRiceCakeRGB332.png
    - MochiRiceCakeRGB565.png
  - App.main
  - AppScreen.page
  - ARGBPage.page
  - ASTCPage1.page
  - ASTCPage2.page
  - ASTCPage3.page
  - ASTCPage4.page

Library

- ESD Layouts
- ESD Render Functions
- ESD Theme
- ESD Utilities
- ESD Widgets
- Logic Flow
- Logic Interface
- User Actors

Properties

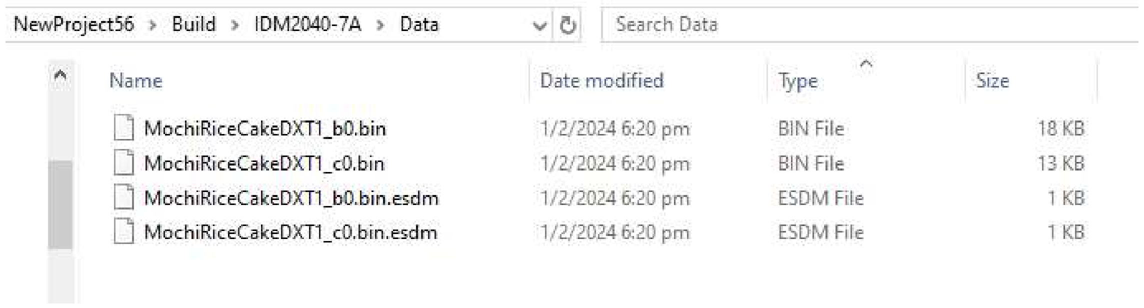
Property
(Resource File Proxy)
File Name
Name
Category
Format
Compressed
Resource Type
Cell Height
Cell Names
Persistent

ARGB1555
L1
L2
L4
L8
RGB332
ARGB2
ARGB4
RGB565
PALETTED565
PALETTED4444
PALETTED8
DXT1
DXT1L2
DXT1PALETTED
JPEG
PNG
COMPRESSED_RGBA_ASTC_4x4_KHR
COMPRESSED_RGBA_ASTC_5x4_KHR
COMPRESSED_RGBA_ASTC_5x5_KHR
COMPRESSED_RGBA_ASTC_6x5_KHR
COMPRESSED_RGBA_ASTC_6x6_KHR
COMPRESSED_RGBA_ASTC_8x5_KHR
COMPRESSED_RGBA_ASTC_8x6_KHR
COMPRESSED_RGBA_ASTC_8x8_KHR
COMPRESSED_RGBA_ASTC_10x5_KHR
COMPRESSED_RGBA_ASTC_10x6_KHR
COMPRESSED_RGBA_ASTC_10x8_KHR
COMPRESSED_RGBA_ASTC_10x10_KHR
COMPRESSED_RGBA_ASTC_12x10_KHR
COMPRESSED_RGBA_ASTC_12x12_KHR

Figure 9 - DXT1 format bitmap in BitmapFormats example project

Output Format for DXT1:

When the DXT1 format is selected in ESD, it is then converted to two file formats: *_b0.raw and *_c0.raw, **or** alternatively, *_b0.bin and *_b0.bin (if the compressed flag is selected).



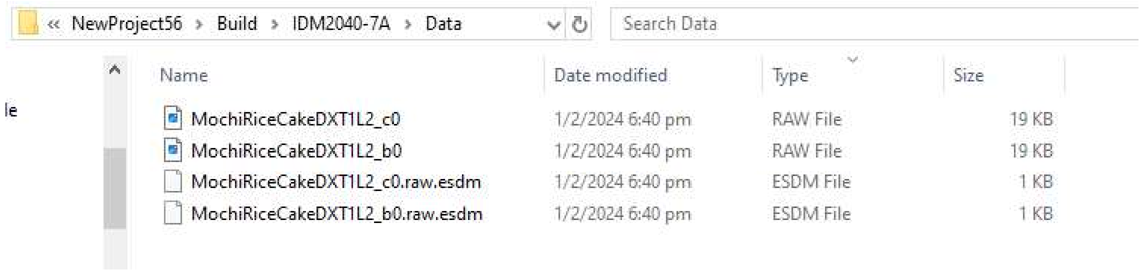
NewProject56 > Build > IDM2040-7A > Data

Name	Date modified	Type	Size
MochiRiceCakeDXT1_b0.bin	1/2/2024 6:20 pm	BIN File	18 KB
MochiRiceCakeDXT1_c0.bin	1/2/2024 6:20 pm	BIN File	13 KB
MochiRiceCakeDXT1_b0.bin.esdm	1/2/2024 6:20 pm	ESDM File	1 KB
MochiRiceCakeDXT1_c0.bin.esdm	1/2/2024 6:20 pm	ESDM File	1 KB

Users must download the converted bitmap resources, either *_b0.raw and *_c0.raw **or** *_b0.bin and *_b0.bin, into the root directory of an SD card. Subsequently, insert the SD card into the hardware module. There is no requirement to download the .esdm file format onto the SD card unless dynamic bitmap loading is used. For the dynamic bitmap loading feature, .esdm files must be copied to the root directory of the SD card.

Output Format for DXT1L2:

The DXT1L2 output format closely resembles DXT1. When ESD is converted, two file formats are produced: *_b0.raw and *_c0.raw, or alternatively, *_b0.bin and *_b0.bin (if the compressed flag is selected).



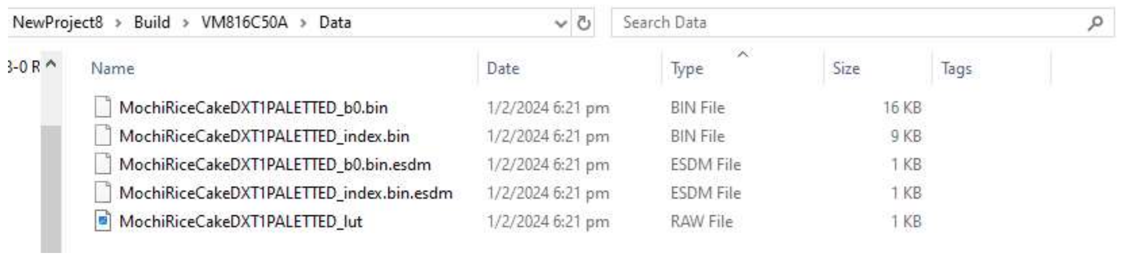
NewProject56 > Build > IDM2040-7A > Data

Name	Date modified	Type	Size
MochiRiceCakeDXT1L2_c0	1/2/2024 6:40 pm	RAW File	19 KB
MochiRiceCakeDXT1L2_b0	1/2/2024 6:40 pm	RAW File	19 KB
MochiRiceCakeDXT1L2_c0.raw.esdm	1/2/2024 6:40 pm	ESDM File	1 KB
MochiRiceCakeDXT1L2_b0.raw.esdm	1/2/2024 6:40 pm	ESDM File	1 KB

To use the converted bitmap resources, users need to transfer either *_b0.raw and *_c0.raw **or** *_b0.bin and *_b0.bin files to the root directory of an SD card. It's important to note that there is no need to download the .esdm file format onto the SD card unless dynamic bitmap loading is used.

Output Format for DXT1PALETTED:

If the DXT1PALETTED format is chosen in ESD, it undergoes conversion into three file formats: *_b0.raw, *_index.raw, and *_lut.raw. Alternatively, when the compressed flag is selected, the conversion results in *_b0.bin, *_index.bin, and *_lut.raw.



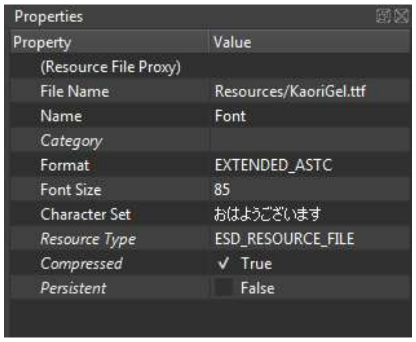
NewProject8 > Build > VM816C50A > Data

Name	Date	Type	Size	Tags
MochiRiceCakeDXT1PALETTED_b0.bin	1/2/2024 6:21 pm	BIN File	16 KB	
MochiRiceCakeDXT1PALETTED_index.bin	1/2/2024 6:21 pm	BIN File	9 KB	
MochiRiceCakeDXT1PALETTED_b0.bin.esdm	1/2/2024 6:21 pm	ESDM File	1 KB	
MochiRiceCakeDXT1PALETTED_index.bin.esdm	1/2/2024 6:21 pm	ESDM File	1 KB	
MochiRiceCakeDXT1PALETTED_lut	1/2/2024 6:21 pm	RAW File	1 KB	

To utilize the converted bitmap resources, users should download either *_b0.raw, *_index.raw, and *_lut.raw or *_b0.bin, *_index.bin, and *_lut.raw files into the root directory of an SD card. Following this, insert the SD card into the hardware module. It's important to note that there is no need to download the .esdm file format onto the SD card unless dynamic bitmap loading is used. For the dynamic bitmap loading feature, .esdm files must be copied to the root directory of the SD card.

Support extended font format

Unicode font file can be added to a project by clicking ***File -> Add*** from the menu bar and in the *Property* window, input font size and character set as desired.

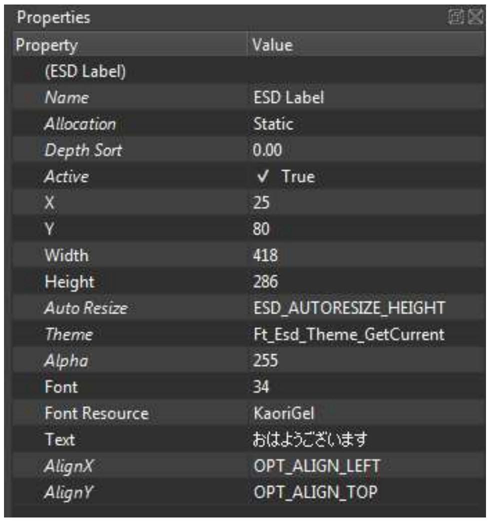


Properties

Property	Value
(Resource File Proxy)	
File Name	Resources/KaoriGel.ttf
Name	Font
Category	
Format	EXTENDED_ASTC
Font Size	85
Character Set	おはようございます
Resource Type	ESD_RESOURCE_FILE
Compressed	✓ True
Persistent	False

Figure 10 - Font size and Character set

Once the font resource is ready, add a widget, such as ESD Label, ESD Button to project. In *Property* window, select appropriate "*Font Resource*".



Properties

Property	Value
(ESD Label)	
Name	ESD Label
Allocation	Static
Depth Sort	0.00
Active	✓ True
X	25
Y	80
Width	418
Height	286
Auto Resize	ESD_AUTORESIZE_HEIGHT
Theme	Ft_Esd_Theme_GetCurrent
Alpha	255
Font	34
Font Resource	KaoriGel
Text	おはようございます
AlignX	OPT_ALIGN_LEFT
AlignY	OPT_ALIGN_TOP

Figure 11 - Select Unicode Font Resource

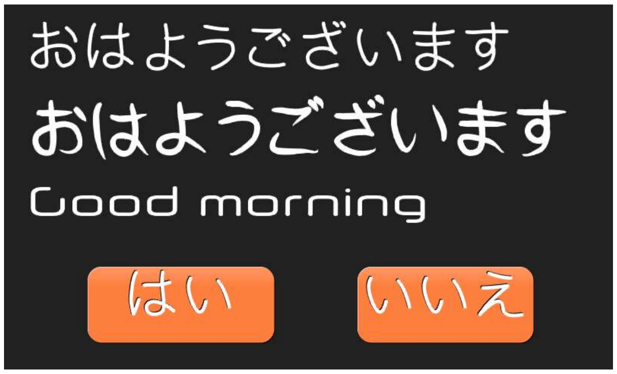


Figure 12 - Unicode font applied for Label and Button

ASTC Encoder

Since ESD 4.12, the ASTC encoder is upgraded to version 2.0. This is a new version of ASTC encoder, which is responsible for the image conversion process to ASTC format. It is from ARM and claims 30% faster performance than ASTC encoder 1.0.

Video Converter

ESD enables users to add video format into project and converts it to Eve specific format by employing ffmpeg.

It eases the procedure to add the video playback functionality in ESD project. In ESD examples project `PlayVideo` is added to play an mp4 video and avi video. After that, the videos are converted to mjpeq format and played.

Support Animation

ESD enables users to add **EVE** specific animation file (.anim) into project. The key features are:

- auto generation of c file and widget files
- thumbnail
- start and stop signals

User can convert .gif file to .anim file with **EAB** tool.
The steps to add an animation is as below:

1. Convert .gif to .anim with EAB tool
2. Add the .anim file into a ESD project
3. Drag the animation widget from user widget (automatically generated) to the page which includes the animation
4. Create customized start and end signal, and connect them to the corresponding slots on animation widget

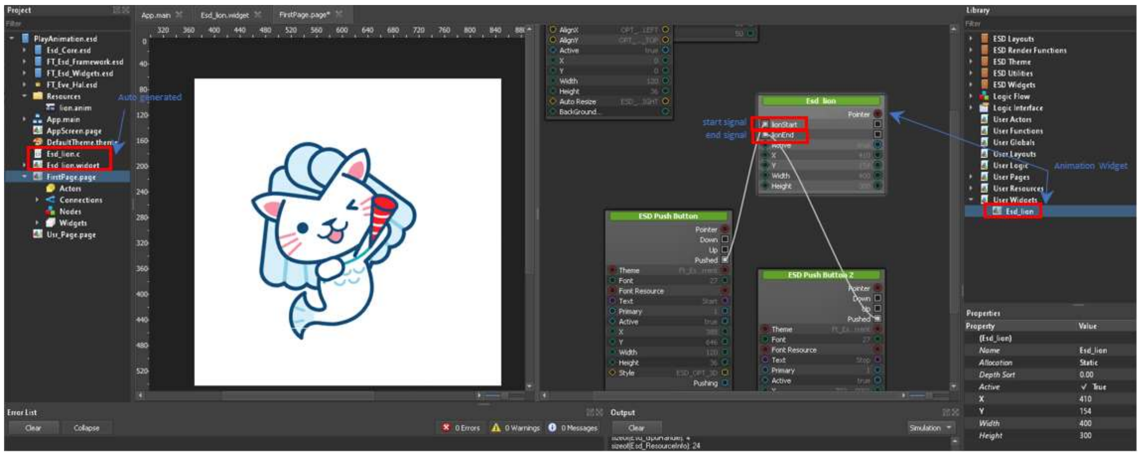
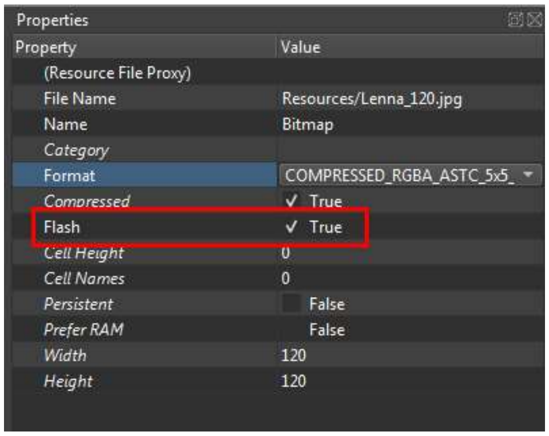


Figure 13 - Create animation in project

The example project PlayAnimation showcases how to play and stop an animation. The screencast video "playAnimationtutorial.gif" under the same folder records the procedure to design the example project.

Generate Flash file and write Flash file to device

Resources can be loaded from **EVE**-connected flash or **RAM_G**. ESD can generate Flash file for BT81X projects, then uploads the flash file to the device by pressing the **[Build and Upload to Hardware]** button.



Properties

Property	Value
(Resource File Proxy)	
File Name	Resources/Lenna_120.jpg
Name	Bitmap
Category	
Format	COMPRESSED_RGBA_ASTC_5x5_
Compressed	✓ True
Flash	✓ True
Cell Height	0
Cell Names	0
Persistent	False
Prefer RAM	False
Width	120
Height	120

Figure 14 - Mark resource as storage in Flash

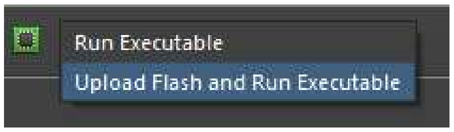
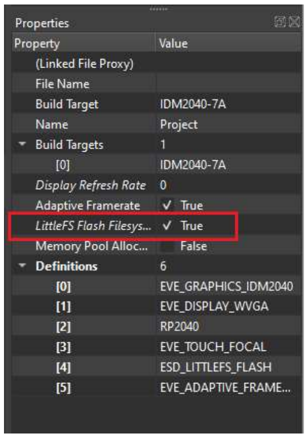


Figure 15 - Upload Flash file to device

LittleFS File System with EVE connect flash memory

The LittleFS is a fail-safe file system designed for embedded systems, specifically for microcontrollers that use external flash storage. BT815 and above devices support flash. The AssetBrowser example project demonstrates the utilization of the LittleFS file system in ESD. Kindly review the following steps outlining how to use LittleFS within the project.

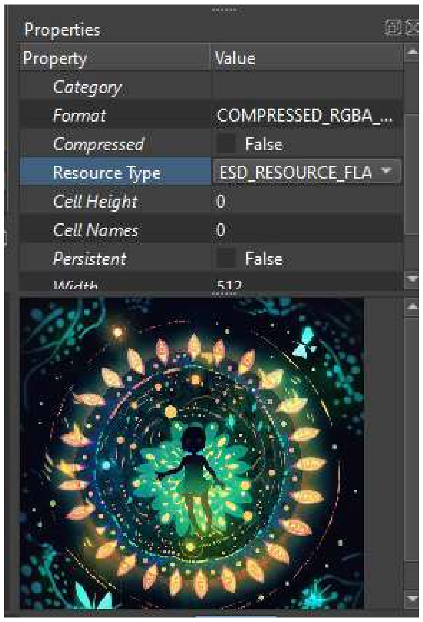
1) In the ESD Project properties, it's essential to ensure that the LittleFS Flash file system is selected as "True"



Property	Value
(Linked File Proxy)	
File Name	
Build Target	IDM2040-7A
Name	Project
Build Targets	1
[0]	IDM2040-7A
Display Refresh Rate	0
Adaptive Framerate	✓ True
LittleFS Flash Filesys...	✓ True
Memory Pool Alloc...	False
Definitions	6
[0]	EVE_GRAPHICS_IDM2040
[1]	EVE_DISPLAY_WVGA
[2]	RP2040
[3]	EVE_TOUCH_FOCAL
[4]	ESD_LITTLEFS_FLASH
[5]	EVE_ADAPTIVE_FRAME...

Figure 16 – LittleFS enable option in Project

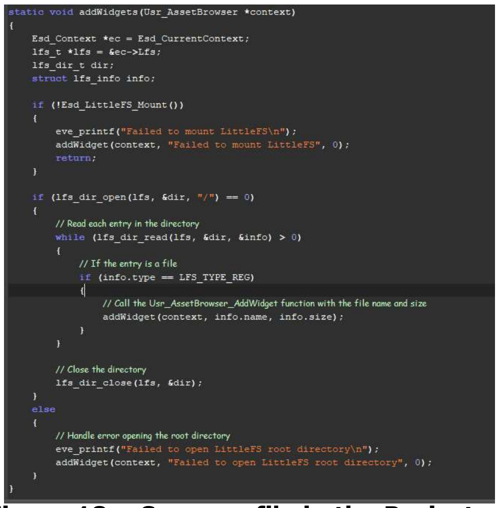
2) Within the Resource Image properties, it is imperative to choose the Resource type as "ESD_RESOURCE_FLASH".



Property	Value
Category	
Format	COMPRESSED_RGBA_...
Compressed	False
Resource Type	ESD_RESOURCE_FLA
Cell Height	0
Cell Names	0
Persistent	False
Width	512

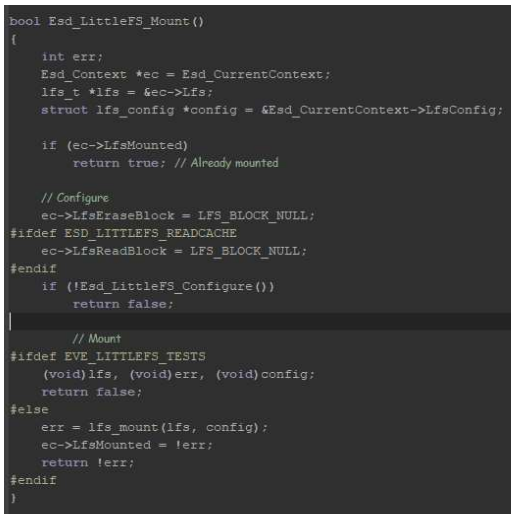
Figure 17 – Selected the ESD_RESOURCE_FLASH

3) Initially, call the Esd_LittleFS_Mount() API function from the C source code within the project. This API verifies if LittleFS is enabled. Upon verification, it proceeds to invoke the Esd_LittleFS_Configure() and lfs_mount() functions.



```c
static void addWidgets(Usr_AssetBrowser *context)
{
    Esd_Context *ec = Esd_CurrentContext;
    lfs_t *lfs = &ec->Lfs;
    lfs_dir_t dir;
    struct lfs_info info;

    if (!Esd_LittleFS_Mount())
    {
        eve_printf("Failed to mount LittleFS\n");
        addWidget(context, "Failed to mount LittleFS", 0);
        return;
    }

    if (lfs_dir_open(lfs, &dir, "/") == 0)
    {
        // Read each entry in the directory
        while (lfs_dir_read(lfs, &dir, &info) > 0)
        {
            // If the entry is a file
            if (info.type == LFS_TYPE_REG)
            {
                // Call the Usr_AssetBrowser_AddWidget function with the file name and size
                addWidget(context, info.name, info.size);
            }
        }

        // Close the directory
        lfs_dir_close(lfs, &dir);
    }
    else
    {
        // Handle error opening the root directory
        eve_printf("Failed to open LittleFS root directory\n");
        addWidget(context, "Failed to open LittleFS root directory", 0);
    }
}
```

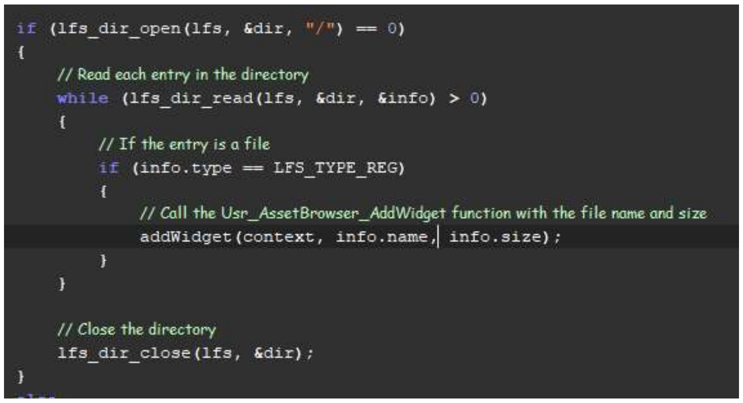
Figure 18 – C source file in the Project



```c
bool Esd_LittleFS_Mount()
{
    int err;
    Esd_Context *ec = Esd_CurrentContext;
    lfs_t *lfs = &ec->Lfs;
    struct lfs_config *config = &Esd_CurrentContext->LfsConfig;

    if (ec->LfsMounted)
        return true; // Already mounted

    // Configure
    ec->LfsEraseBlock = LFS_BLOCK_NULL;
#ifdef ESD_LITTLEFS_READCACHE
    ec->LfsReadBlock = LFS_BLOCK_NULL;
#endif
    if (!Esd_LittleFS_Configure())
        return false;

        // Mount
#ifdef EVE_LITTLEFS_TESTS
    (void)lfs, (void)err, (void)config;
    return false;
#else
    err = lfs_mount(lfs, config);
    ec->LfsMounted = !err;
    return !err;
#endif
}
```

Figure 19 – ESD_LittleFS.c file in ESD_Core

4) Open the directory using lfs_dir_open(lfs, &dir, "/") API function, and read each entry in the directory with while loop using lfs_dir_read(lfs, &dir, &info) API Function and get the output File Name and File Size.



```c
if (lfs_dir_open(lfs, &dir, "/") == 0)
{
    // Read each entry in the directory
    while (lfs_dir_read(lfs, &dir, &info) > 0)
    {
        // If the entry is a file
        if (info.type == LFS_TYPE_REG)
        {
            // Call the Usr_AssetBrowser_AddWidget function with the file name and size
            addWidget(context, info.name, info.size);
        }
    }

    // Close the directory
    lfs_dir_close(lfs, &dir);
}
```

5) Generate rows consecutively using the "addWidget(context, info.name, info.size)" function, including the image file name, file size, and Preview push button. Additionally, within this function's verification process, rows won't generate if the file name contains ".esdm". However, if the file name doesn't contain ".esdm", the rows will be created as planned. For further details about ".esdm", please refer to page 55.

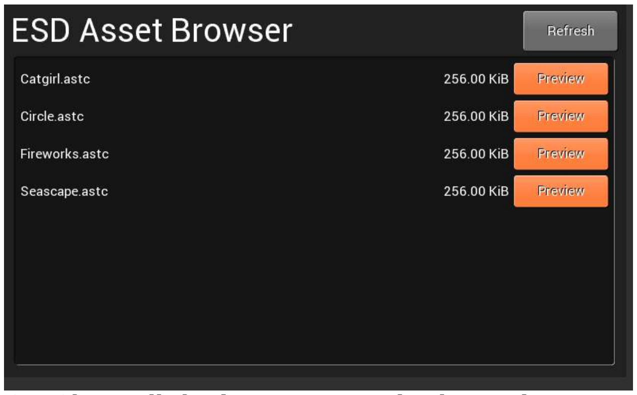
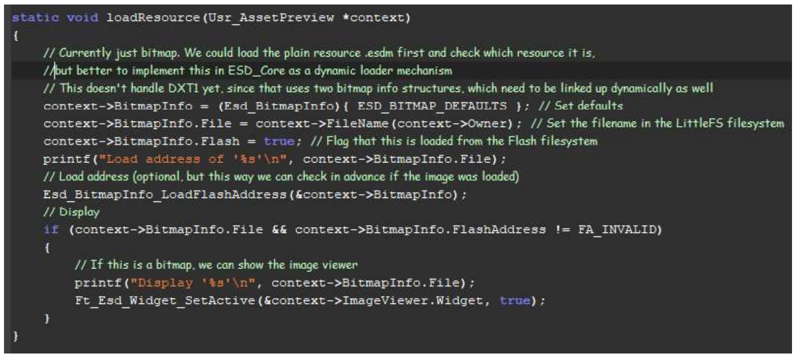


Figure 20 – Show all the images name in the project AssetBrowser

6) After clicking the Preview button, trigger the execution of the loadResource(context) function. Inside this function, set the current image file name to bitmapinfo. Proceed to retrieve the flash address using Esd_BitmapInfo_LoadFlashAddress(&context->BitmapInfo) and then assign bitmapinfo with the flash address into the Image Viewer.



```
static void loadResource(Usr_AssetPreview *context)
{
    // Currently just bitmap. We could load the plain resource .esdm first and check which resource it is,
    //but better to implement this in ESD_Core as a dynamic loader mechanism
    // This doesn't handle DXT1 yet, since that uses two bitmap info structures, which need to be linked up dynamically as well
    context->BitmapInfo = (Esd_BitmapInfo){ ESD_BITMAP_DEFAULTS }; // Set defaults
    context->BitmapInfo.File = context->FileName(context->Owner); // Set the filename in the LittleFS filesystem
    context->BitmapInfo.Flash = true; // Flag that this is loaded from the Flash filesystem
    printf("Load address of '%s'\n", context->BitmapInfo.File);
    // Load address (optional, but this way we can check in advance if the image was loaded)
    Esd_BitmapInfo_LoadFlashAddress(&context->BitmapInfo);
    // Display
    if (context->BitmapInfo.File && context->BitmapInfo.FlashAddress != FA_INVALID)
    {
        // If this is a bitmap, we can show the image viewer
        printf("Display '%s'\n", context->BitmapInfo.File);
        Ft_Esd_Widget_SetActive(&context->ImageViewer.Widget, true);
    }
}
```

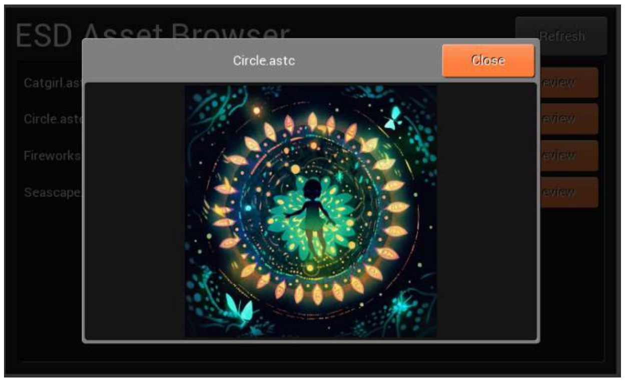


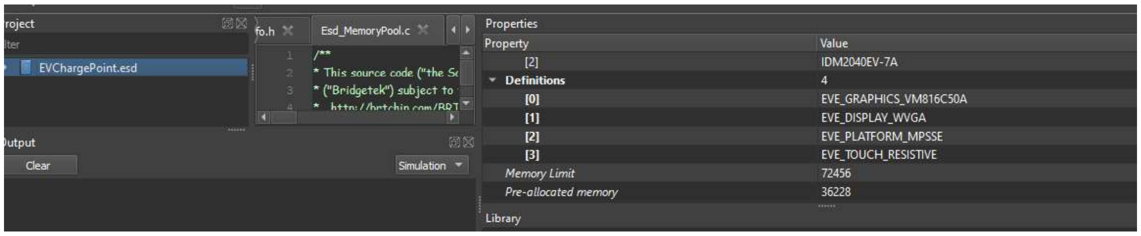
Figure 21 – Show preview image in the project AssetBrowser

Restructured EVE HAL Library

EVE HAL Library contains C source files, and a sub-folder “Hdr” that comprises of the C header files. The EVE HAL Library comprises the source code for the EVE HAL platform, the header files for the EVE HAL platform, and coprocessor commands.

Memory management

Some of the widgets in ESD will use heap memory, in other word, malloc and free will be used. In the embedded world, it is not recommended due to multiple drawbacks of using heap. However, it is not easy for ESD to fully use static memory because it is an UI design tool. So, a memory pool management module is developed in ESD4.16 onward, with this implementation, user can restrict the heap memory usage into a controlled range as below:



```
/**
* This source code ("the Sc
* ("Bridgetek") subject to
*   http://brtchip.com/BRT
```

Property	Value
[2]	IDM2040EV-7A
Definitions	4
[0]	EVE_GRAPHICS_VM816C50A
[1]	EVE_DISPLAY_WVGA
[2]	EVE_PLATFORM_MPSSE
[3]	EVE_TOUCH_RESISTIVE
Memory Limit	72456
Pre-allocated memory	36228

Figure 22 - Config the memory

Figure 22 shows that 2 variables pre-allocated memory and Memory limit are defined in the properties of the root project to configure memory allocation. Pre-allocated memory is the estimated memory usage of the project, and memory limit is the max value can be assigned to the project, if the pre-allocated memory is not enough due to some reasons, ESD will try to extend the memory up to the memory limit. The minimum values of pre-allocated and memory limit are 0 because not all the widgets use heap memory. The default values are 72456 for Memory Limit and 36228 for Pre-allocated Memory, and user should modify them according to the real memory usage of their projects.

UI enhancements

The following User Interface related enhancements have been added to ESD -

- Filter for Library Browser and Project Browser window
- More comprehensive toolbar buttons for platform description
- New menu to clean up unused/generated files in project
- Simplify the process of adding new resources into project
- Added context menu for document tab

Impacts of ESD 4.X updated features

Since the new/updated features of ESD 4.X may cause API changes, existing ESD 3.0 projects are not fully compatible with ESD. Although ESD 4.X has the utility inside to detect and migrate opening ESD 3.0 project, it may still not work as expected.

- *ESD Layouts:* A new category of widgets “ESD Layout” are introduced into the Library Browser, which enables the layout feature.

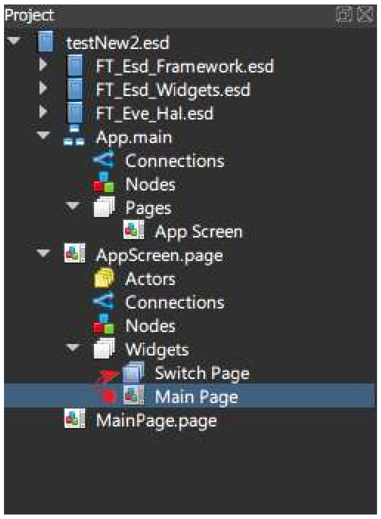
- *ESD Widgets:* A new set of widgets are introduced into "ESD Widgets" category in the library. These set of widgets are to be used in constructing the pages. The nodes under the "ESD Render Functions" are not supposed to construct page as it has no widget interface and the layout widget is unable to manage them.

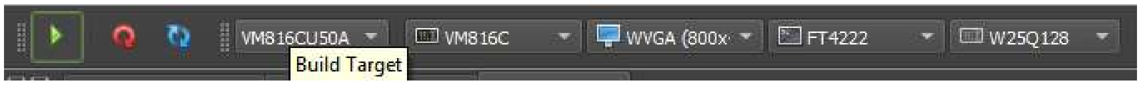
Migrating from ESD 3.0 to ESD 4.X project

When ESD 4.X opens an existing ESD 3.0 project, it will prompt users to migrate to the latest version. If user chooses not to migrate, ESD 4.X will not open ESD 3.0 project.

MIGRATION NOTES

1. Please back up your project if you are not sure about the migration. The migration process will **overwrite the project and it cannot be reversed**.
2. The migrated project file will be renamed from **"*.esd3"** to **"*.esd"**.
3. The target module of the opened project will be redirected to two built-in platforms after migration. So, if the target module is in 320 x 480 resolutions, users may select the platform **"ME810A HV35R"** from "Build Target" combo box from the toolbar and make a platform switch.





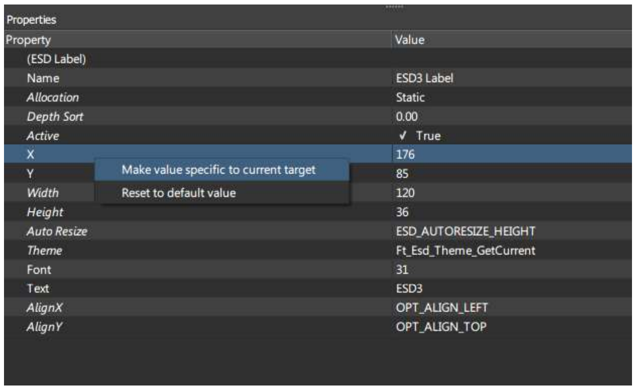
In some cases, if an existing ESD 3.0 project cannot be migrated to ESD 4.X, then users are required to migrate manually.

- The variable *"Parent"* of logic is renamed to *"Owner"* in ESD 4.X.
- *X, Y, Width, Height* properties in widgets is now accessible through the widget variable.
- Instead of pointer, struct *"Ft_Esd_BitmapCell"* shall be used by value in ESD 4.X.
- In order to restore the default behaviour of application logic *"App.main"*, users may need to manually move the existing pages into layout widget *"Switch Page"* in the Project Browser.

Setting Platform Specific Properties

ESD users are able to create one project supporting multiple platforms with various screen size. Therefore, on different platforms, the same widget may have a different property value. For instance, for a platform with screen size 480 x 320, the button size may shrink to a smaller size from the original value for a platform with screen size 800 x 480.

Users can make a property value specific to the current target through the context menu of the property browser. This can be done by right clicking the selected property of the current widget. This is illustrated in the picture given below –

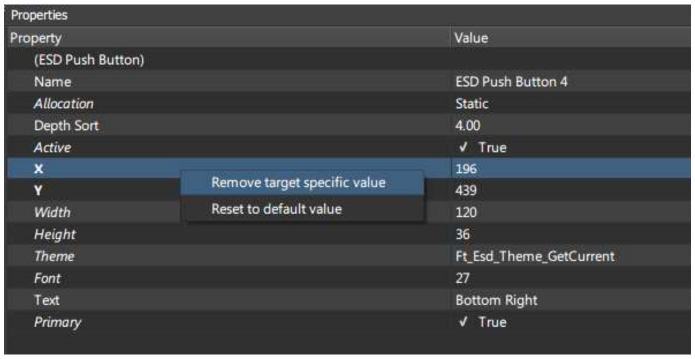


Properties

Property	Value
(ESD Label)	
Name	ESD3 Label
Allocation	Static
Depth Sort	0.00
Active	✓ True
X	176
Y	85
Width	120
Height	36
Auto Resize	ESD_AUTORESIZE_HEIGHT
Theme	Ft_Esd_Theme_GetCurrent
Font	31
Text	ESD3
AlignX	OPT_ALIGN_LEFT
AlignY	OPT_ALIGN_TOP

Figure 23 - Making Value Specific to Current Target

User can remove value specific to current target through context menu in property browser by right clicking the selected property of current widget. This is illustrated in the picture given below -



Properties

Property	Value
(ESD Push Button)	
Name	ESD Push Button 4
Allocation	Static
Depth Sort	4.00
Active	✓ True
X	196
Y	439
Width	120
Height	36
Theme	Ft_Esd_Theme_GetCurrent
Font	27
Text	Bottom Right
Primary	✓ True

Figure 24 - Removing Target Specific Value

Users can refer to the "ScreenResolution" project under the "Installation/Example" folder.

Migrating existing projects to new platform

All the existing projects may have been developed using a specific platform. Users can choose to migrate their projects to a different platform by selecting the appropriate "Build Target" combo box. If the desired target is not available in the "Build Target" Combo box, then the user can configure the individual specifications ("Eve Platform", "Display Resolution" and "Host Platform") of the target by selecting the appropriate combo boxes.

Advanced User Settings

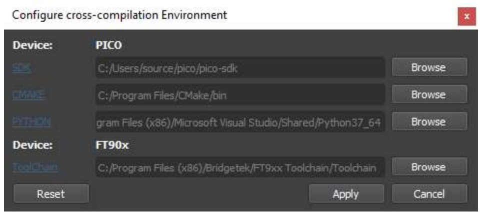
Advanced users are allowed to configure the ESD settings to better assist them in their GUI application development with EVE products. The Advanced User Settings allow the user to turn on/off specific features and functions in the ESD. This section shall provide a brief explanation on the usage and effect on the ESD. As it involves modifying the application configuration file, it should only be carried out by someone with adequate knowledge. It is recommended to leave any advanced settings at their default settings.

Application Settings

The "Main.config" file can be found in the "Settings" folder under the installation path. This file contains application wide settings which may affect the interface and behaviour of the ESD. Users are allowed to modify to suit their application needs. After the modification has been done, a restart on the application is required for it to take effect.

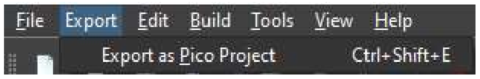
Below is the list of settings that can be found in the file.

- PortableSettings - Enabling/Disabling loading of Host Device Environment settings from the from Screen Designer config upon startup. The loaded settings will be reflected in "Config Host Device".

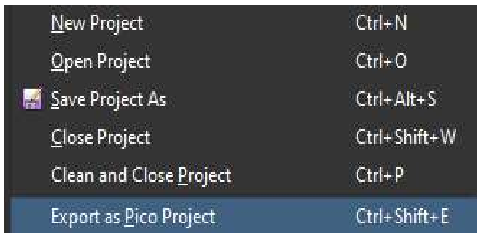

Configure cross-compilation Environment

Device: PICO
SDK: C:/Users/source/pico/pico-sdk
CMAKE: C:/Program Files/CMake/bin
PYTHON: gram Files (x86)/Microsoft Visual Studio/Shared/Python37_64
Device: FT90x
ToolChain: C:/Program Files (x86)/Bridgetek/FT9xx Toolchain/Toolchain

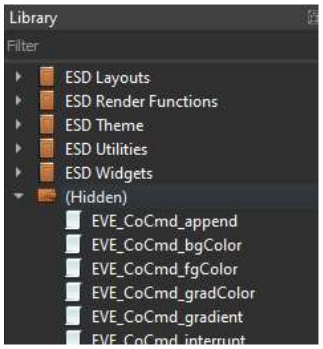
- SimplifiedExport - Enabling/disabling of the simplified export menu actions. An "Export" option will be displayed under the menu bar if it is disabled.



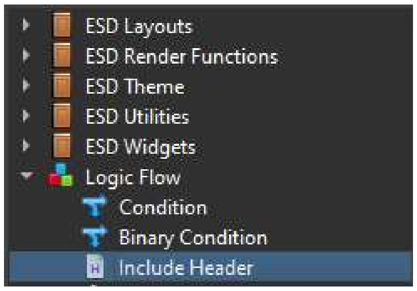
  The "Export" option will be shown under file menu if it is enabled.


New Project Ctrl+N
Open Project Ctrl+O
Save Project As Ctrl+Alt+S
Close Project Ctrl+Shift+W
Clean and Close Project Ctrl+P
Export as Pico Project Ctrl+Shift+E

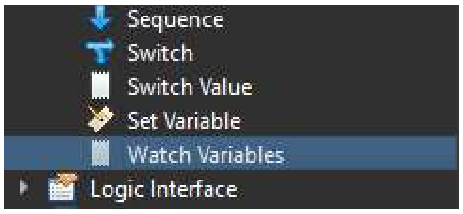
- Hiddentypes – Hide types from library and menu interface. The listed parameters under the "HiddenTypes" will be hidden.
  - _GroupHidden – A new set of commands will be made available to user when it is removed from the hidden state. These controls are mainly to be used by advanced users.
  - _WidgetDocument – Allows the user to create new customised widget.



- _LogicIncludeHeader – This allows the user to use "include header" logic node under the library which helps to add a header file into project.

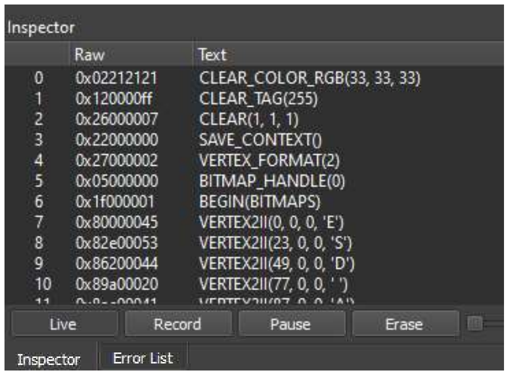


- _LogicWatcher – This allows the user to access "Watch Variables" logic node which helps to watch multiple variables for change.



- _SourceFile – This allows the user to add new empty .c file to the project.

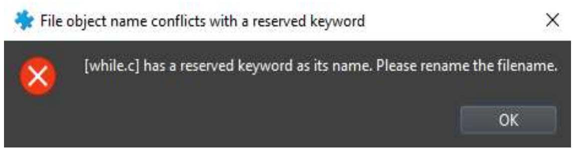
▪ EnableInspector – Enabling of Inspector window which is useful for debugging.



```
     Raw          Text
0    0x02212121   CLEAR_COLOR_RGB(33, 33, 33)
1    0x120000ff   CLEAR_TAG(255)
2    0x26000007   CLEAR(1, 1, 1)
3    0x22000000   SAVE_CONTEXT()
4    0x27000002   VERTEX_FORMAT(2)
5    0x05000000   BITMAP_HANDLE(0)
6    0x1f000001   BEGIN(BITMAPS)
7    0x80000045   VERTEX2II(0, 0, 0, 'E')
8    0x82e00053   VERTEX2II(23, 0, 0, 'S')
9    0x86200044   VERTEX2II(49, 0, 0, 'D')
10   0x89a00020   VERTEX2II(77, 0, 0, ' ')
```

▪ ShowBitmapHandleUsage – Displaying of bitmap handle usage in status bar.

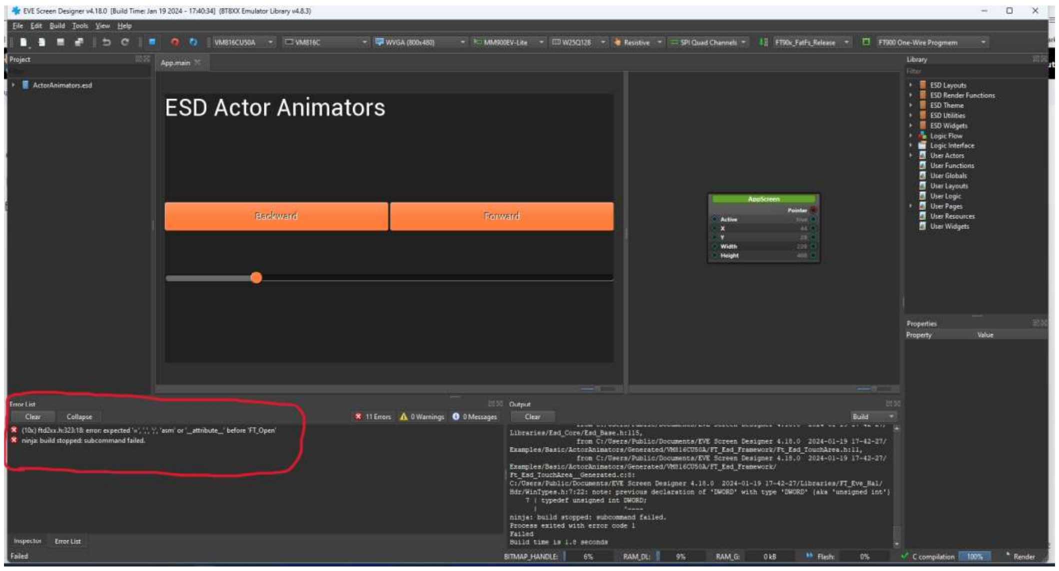
▪ Keywords – List of keywords to be avoided when naming ESD file object. An error display message box will be shown if there is a conflict with the reserved keyword.


File object name conflicts with a reserved keyword

[while.c] has a reserved keyword as its name. Please rename the filename.

Troubleshooting

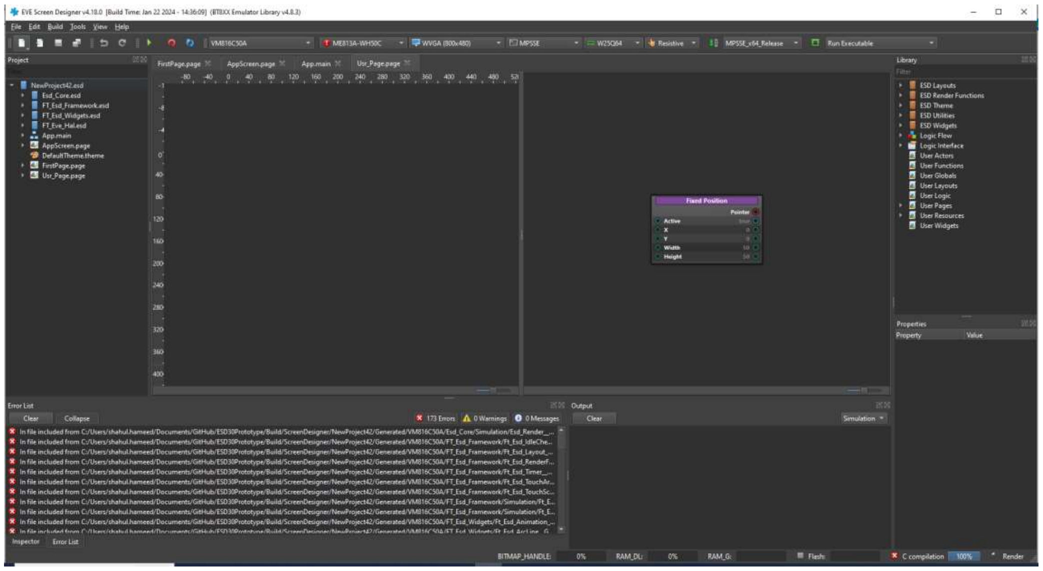
If you build the project with the incorrect host platform, errors may be displayed. Always ensure to choose the correct EVE host platform. Refer to additional details for further information. See the table



```
(10x) ftd2xx.h:323:18: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'FT_Open'
ninja: build stopped: subcommand failed.
```

```
Libraries/Esd_Core/Esd_Base.h:115,
                 from C:/Users/Public/Documents/EVE Screen Designer 4.18.0  2024-01-19 17-42-27/
Examples/Basic/ActorAnimators/Generated/VM816CU50A/FT_Esd_Framework/Ft_Esd_TouchArea.h:11,
                 from C:/Users/Public/Documents/EVE Screen Designer 4.18.0  2024-01-19 17-42-27/
Examples/Basic/ActorAnimators/Generated/VM816CU50A/FT_Esd_Framework/
Ft_Esd_TouchArea__Generated.c:8:
C:/Users/Public/Documents/EVE Screen Designer 4.18.0  2024-01-19 17-42-27/Libraries/FT_Eve_Hal/
Hdr/WinTypes.h:7:22: note: previous declaration of 'DWORD' with type 'DWORD' {aka 'unsigned int'}
    7 | typedef unsigned int DWORD;
      |                      ^~~~~
ninja: build stopped: subcommand failed.
Process exited with error code 1
Failed
Build time is 1.8 seconds
```

As depicted in the figure above, an error commonly arises when the Eve platform VM816C does not correspond to the host platform MM900x. Ensure that you choose the correct host platform, namely FT4222.



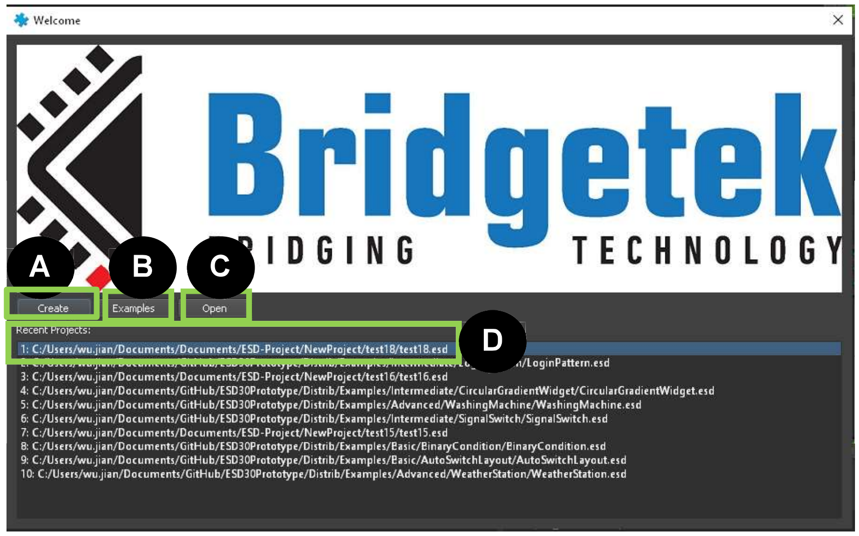
```
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/Esd_Core/Simulation/Esd_Render_...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Framework/Ft_Esd_IdleChe...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Framework/Ft_Esd_Layout_...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Framework/Ft_Esd_RenderF...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Framework/Ft_Esd_Timer__...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Framework/Ft_Esd_TouchAr...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Framework/Ft_Esd_TouchSc...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Framework/Simulation/Ft_E...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Framework/Simulation/Ft_E...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Widgets/Ft_Esd_Animation_...
In file included from C:/Users/shahul.hameed/Documents/GitHub/ESD30Prototype/Build/ScreenDesigner/NewProject42/Generated/VM816C50A/FT_Esd_Widgets/Ft_Esd_ArcLine_G...
```

As depicted in the figure above, if ESD encounter an error during the creation of a new project using the build targets VM816C50A or VM816CU50A due to an incorrect EVE platform. Please make sure to select the correct EVE platform, which is VM816C.

E. Getting Started

The Welcome Dialog

There is a welcome dialog which will be popped up when ESD is started.



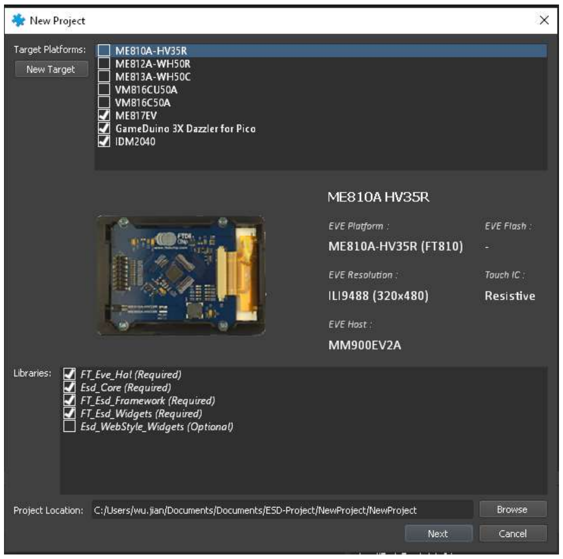
Recent Projects:

1: C:/Users/wu.jian/Documents/Documents/ESD-Project/NewProject/test18/test18.esd
2: [illegible]/LoginPattern.esd
3: C:/Users/wu.jian/Documents/Documents/ESD-Project/NewProject/test16/test16.esd
4: C:/Users/wu.jian/Documents/GitHub/ESD30Prototype/Distrib/Examples/Intermediate/CircularGradientWidget/CircularGradientWidget.esd
5: C:/Users/wu.jian/Documents/GitHub/ESD30Prototype/Distrib/Examples/Advanced/WashingMachine/WashingMachine.esd
6: C:/Users/wu.jian/Documents/GitHub/ESD30Prototype/Distrib/Examples/Intermediate/SignalSwitch/SignalSwitch.esd
7: C:/Users/wu.jian/Documents/Documents/ESD-Project/NewProject/test15/test15.esd
8: C:/Users/wu.jian/Documents/GitHub/ESD30Prototype/Distrib/Examples/Basic/BinaryCondition/BinaryCondition.esd
9: C:/Users/wu.jian/Documents/GitHub/ESD30Prototype/Distrib/Examples/Basic/AutoSwitchLayout/AutoSwitchLayout.esd
10: C:/Users/wu.jian/Documents/GitHub/ESD30Prototype/Distrib/Examples/Advanced/WeatherStation/WeatherStation.esd

Figure 25 - ESD welcome dialog

A: Create a new project
B: Open an example project
C: Open a project in the file system
D: Recent project, double click to open

Create a new project:



Target Platforms:
- [ ] ME810A-HV35R
- [ ] ME812A-WH50R
- [ ] ME813A-WH50C
- [ ] VM816CU50A
- [ ] VM816C50A
- [✓] ME817EV
- [✓] GameDuino 3X Dazzler for Pico
- [✓] IDM2040

ME810A HV35R

EVE Platform :	*EVE Flash :*
ME810A-HV35R (FT810)	-
EVE Resolution :	*Touch IC :*
ILI9488 (320x480)	Resistive
EVE Host :	
MM900EV2A	

Libraries:
- [✓] *FT_Eve_Hal (Required)*
- [✓] *Esd_Core (Required)*
- [✓] *FT_Esd_Framework (Required)*
- [✓] *FT_Esd_Widgets (Required)*
- [ ] *Esd_WebStyle_Widgets (Optional)*

Project Location: C:/Users/wu.jian/Documents/Documents/ESD-Project/NewProject/NewProject

Figure 26 - ESD new project dialog

The dialog allows user to create a new project with predefined build target listed in the picture, or configure a new build target (choose platform, display resolution, host platform, flash chip and touch IC) with New Target button.

Example projects:

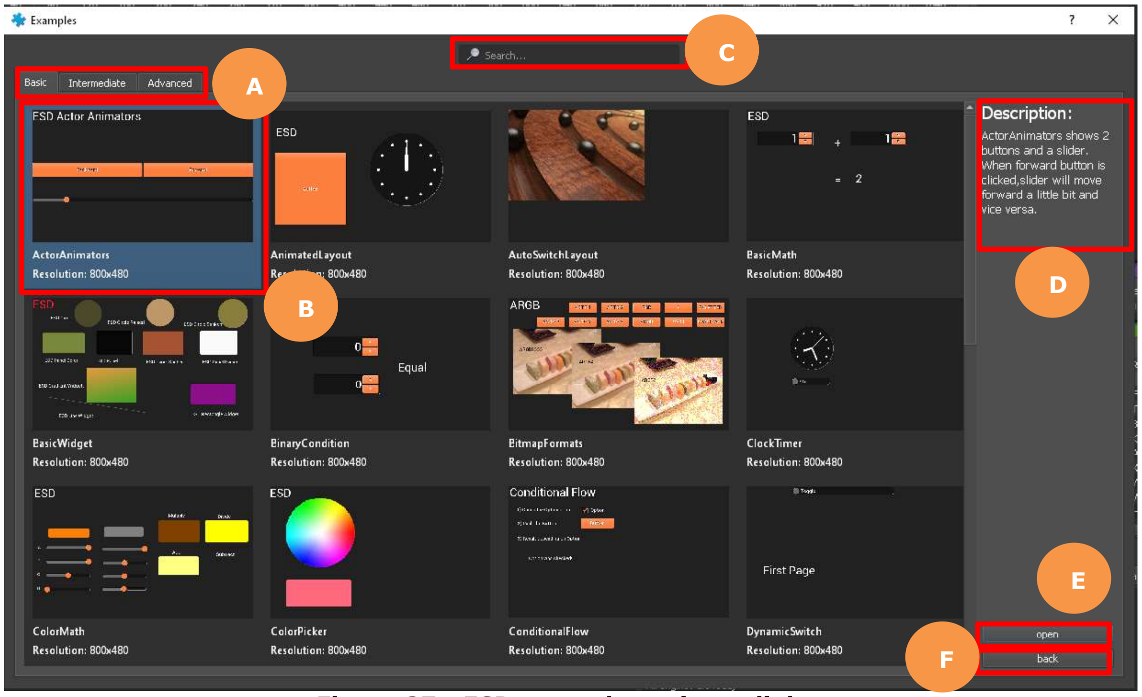


Figure 27 - ESD example projects dialog

A: Tab which represents 3 different categories of example projects, user can switch the category by clicking it.
B: Thumbnail of the project, project name and default resolution
C: Search bar which enable user to select keyword, keyword will match project name and project description. Dialog will update view and only show the matched projects when user input a keyword
D: Project description
E: Open button, it will open the selected project.
F: Back button, show original welcome dialog if this button is clicked.

User can choose to show welcome dialog again by below menu if it is closed:

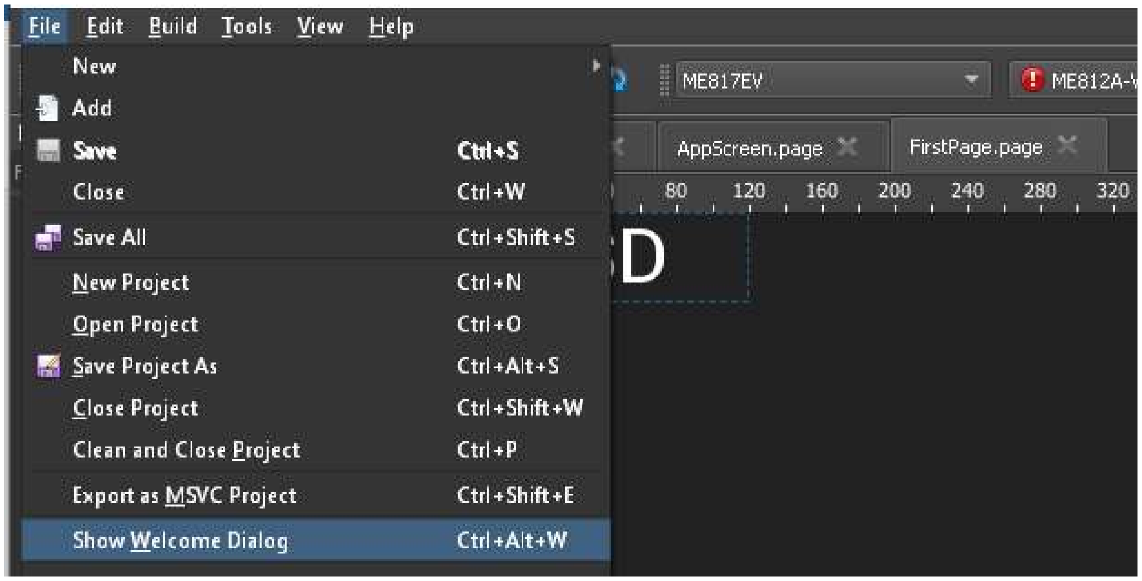


Figure 28 - ESD show welcome dialog menu

The Graphical User Interface

ESD user interface has the following components:

1 *Menu Bar*
2 *Toolbar*
3 *Project Explorer*
4 *Screen Layout Editor*
5 *Logic Node Editor*
6 *Library Browser*
7 *Error List/Inspector*
8 *Output* window
9 *Property Editor*
10 *Status Bar*

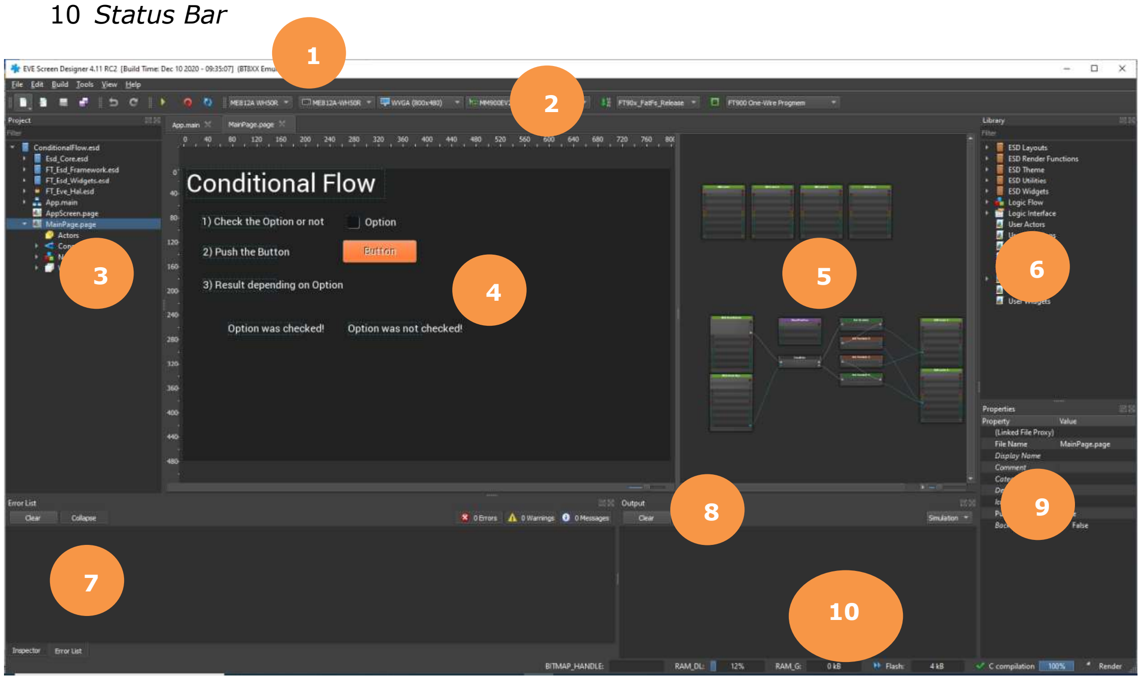


Figure 29 - EVE Screen Designer User Interface Components

Menu bar

The *Menu bar* displays the headings for each drop-down menu. According to the function, the commands are grouped under each of these menu headings.

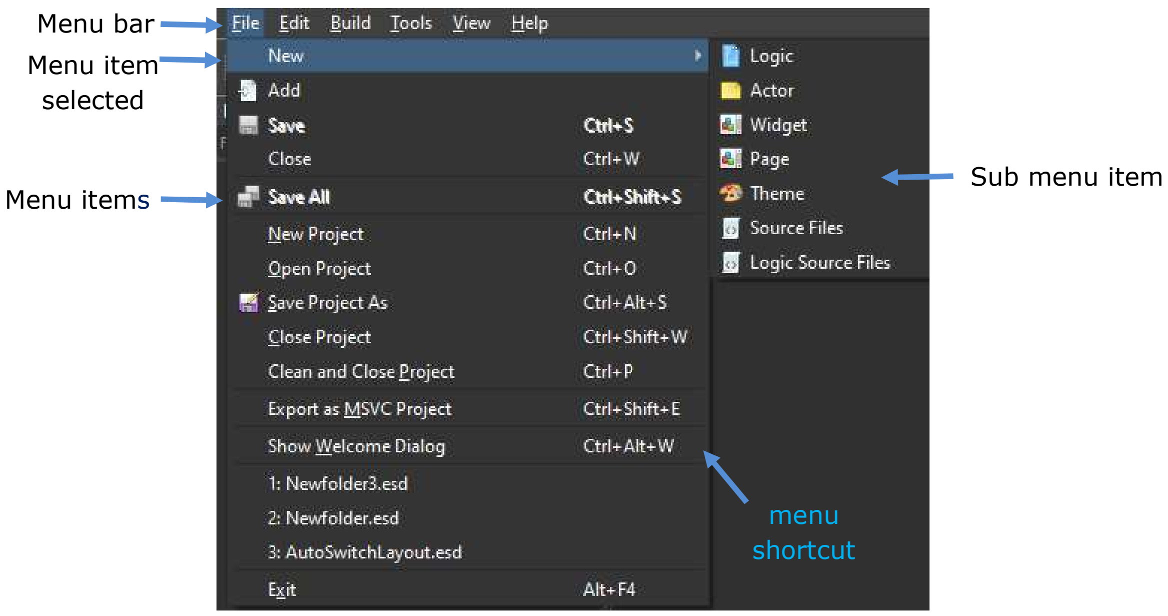
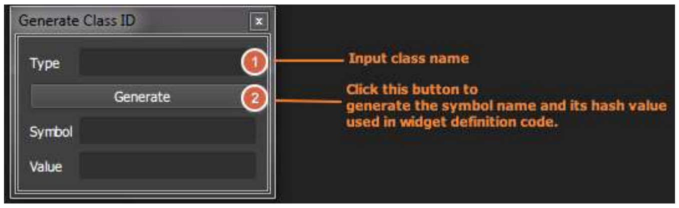


Figure 30 - Menu bar

The following table provides the list of Menu/Submenu and its description –

Menu	Submenu		Description
File	New	Logic	Creates new file for the current project
		Actor	
		Widget	
		Page	
		Theme	
		Source Files	
		Logic Source Files	
	Add		To add existing file(s)/resource (such as bitmap, C source file etc.) to the project
	Save		To save the current file
	Close		To close the current file
	Save All		To save all the files in the current project
	New Project		To open a new project
	Open Project		To open an existing project
	Close Project		To close a current project
	Clean And Close Project		To clean current project then close it

	Export as Eclipse Project		To export as Eclipse project. This will only be shown when supported platform is selected
	Export as MSVC Project		To export as MSVC project. This will only be shown when supported platform is selected
	Export as Pico Project		To export as Pico project. This will only be shown when supported platform is selected
	Show Welcome Dialog		To show welcome dialog which include create new project, show example projects, open project in file system and open recent projects options.
	1:<Project Path>		To open recent projects, up to 10 projects
	Exit		To close the ESD application
Edit	Undo		To reverse the action of a recently performed action
	Redo		To revert the effects of the undo action
	Cut		Cut the selected node(s) to clipboard
	Copy		Copy the selected node(s) to clipboard
	Paste		Paste the node(s) from clipboard
	Find		Find a specific text string in source editor
	Replace		Replace a specific text string in source editor
Build	Build Executable		To generate an executable file
	Build and Upload to Hardware		To generate an executable file and upload to hardware
	Browse to Executable		To navigate to the folder under which the executable file is located
Tools	Generate Class ID		To generate a Class ID for user widget or layout. It is a hash from the widget name and generally useful when user prefers to write code in "C" language.  Generate Class ID Type Generate Symbol Value 1 2 Input class name Click this button to generate the symbol name and its hash value used in widget definition code.
	Clean Unused Generated Files		To clean the generated files which are not in use
	Configure cross-compilation Environment		To show the current tool chain path for the host device, and user can change the paths according to their own configuration.

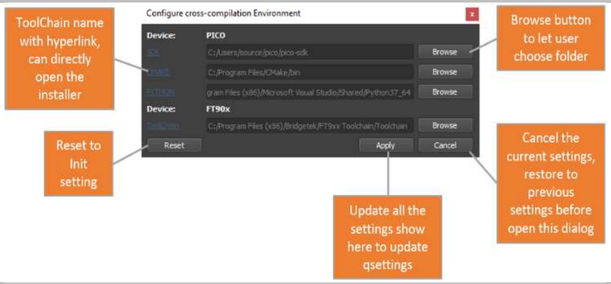
			 Configure cross-compilation Environment Device: PICO SDK C:/Users/source/pico/pico-sdk Browse CMAKE C:/Program Files/CMake/bin Browse PYTHON gram Files (x86)/Microsoft Visual Studio/Shared/Python37_64 Browse Device: FT90x ToolChain C:/Program Files (x86)/Bridgetek/FT9xx Toolchain/Toolchain Browse Reset Apply Cancel ToolChain name with hyperlink, can directly open the installer Browse button to let user choose folder Reset to Init setting Update all the settings show here to update qsettings Cancel the current settings, restore to previous settings before open this dialog
View	Project		To display or hide a component view. By default, all the components are displayed.
	Library		
	Properties		
	Inspector		
	Error List		
	Output		
Help	User Guide		Opens the User Guide with respect to current version
	Widget Introduction Guide		Opens the user guide which describes the in-built widgets in ESD.
	About		Displays the version details
	3rd Party		Displays the copyright, disclaimer and license information of the 3rd party software

Table 2 - Menu & Description

Toolbar

The *Toolbar* provides easy access to common functions (in the form of icons) such as *new file, save file, undo, redo* etc.

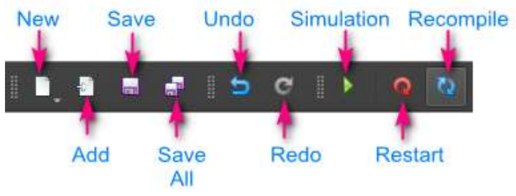
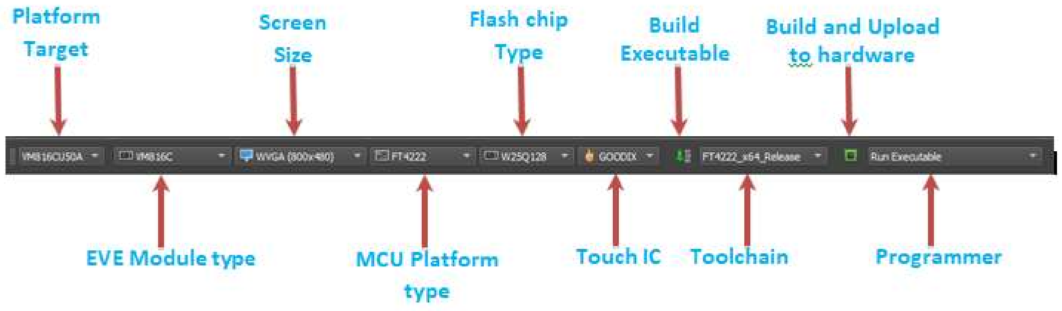


Figure 31 - Toolbar 1



VM816CU50A VM816C WVGA (800x480) FT4222 W25Q128 GOODIX FT4222_x64_Release Run Executable

Figure 32 - Toolbar 2

The following table provides the list of toolbar functions and its description –

Toolbar Function	Description
New	To open new *Logic/Actor/Widget/Page/Theme/Source Files/Logic Source Files*
Add	To add existing resource (such as bitmap, C source file etc.) to the project
Save	To save the currently open file
Save All	To save all the files in the current project
Undo	To reverse the action of a recently performed action
Redo	To revert the effects of the undo action
Simulation	A toggle button which starts or stops the simulation mode. ▶ - This state indicates that the ESD 4.14 is in simulation mode. Clicking this button stops the simulation. ▶ - This state indicates that the ESD 4.14 is out of simulation mode. Users can drag/drop widgets or edit them safely.
Restart	To automatically restart the EVE emulator. Clicking this button will force the simulated screens to be re-drawn.
Recompile	To recompile the whole project's source code using ESD built-in TinyCC compiler. It is a mandatory procedure for simulation.
Platform Target	Select the hardware platform as source code building target
EVE Module Type	To select EVE module type
Screen Size	To set screen size
MCU Platform Type	Select the building configuration when toolchain is invoked to build
Flash Chip Type	To select flash chip for BT81x based series platform
Touch IC	To select capacitive touch controller type
Build Executable	Build the project files
Toolchain	Select toolchain to build
Build and Upload to Hardware	Upload Flash file to device and/or run executable file
Programmer	Select executable mode

Table 3 - Toolbar & Description

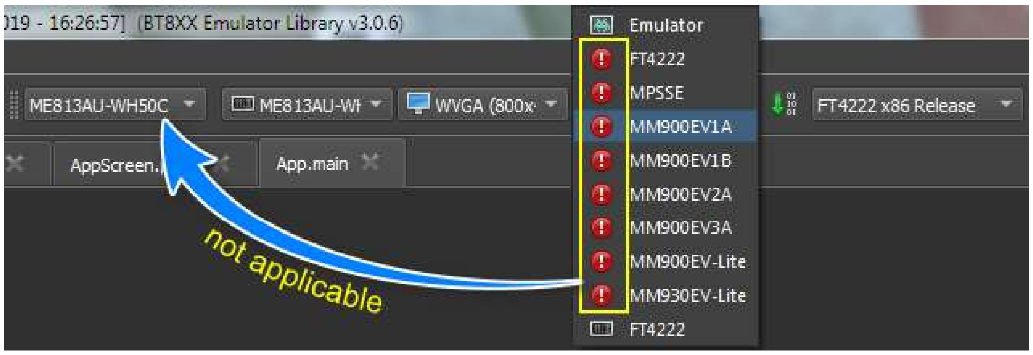


Figure 33 - MCU Type is not applicable to selected platform

The red exclamation mark in the image above means those MCU types are not applicable to the currently selected platform.

Touch IC configuration

FT811, BT815 & BT817 work with capacitive touch controllers. Currently, only the following capacitive touch controllers are supported by **EVE**:

FocalTech:	FT5xx6 / FT6xx6 series IC
Goodix:	GT911/GT9271 series IC.

ESD users can configure the touch IC controller by using the touch IC configuration combo box as shown below.

More information on the touch controllers can be found in AN_336 and BRT_AN_090.

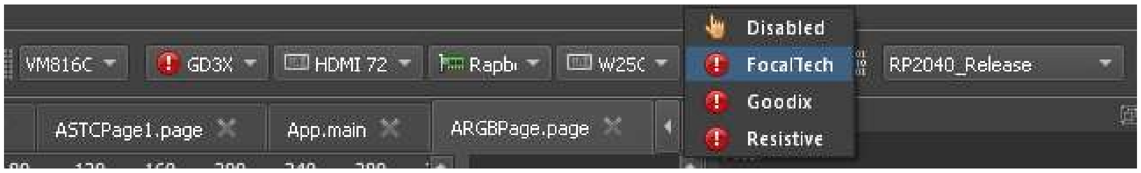


Figure 34 - CTP configuration

Project Explorer

The *Project Explorer* window organizes all the files used in the project in a tree view. It also lists out all the resources used by each file. Project explorer allows users to navigate each page of the project.

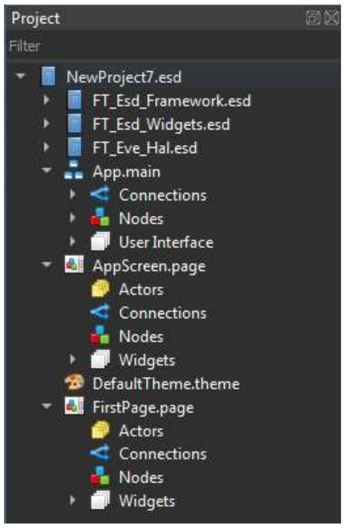


Figure 35 - Project Explorer window

In case of many items existing in a project, user can use filter textbox in order to get a sort list of items. Text that matches the filter is highlighted. Regular expressions are supported.

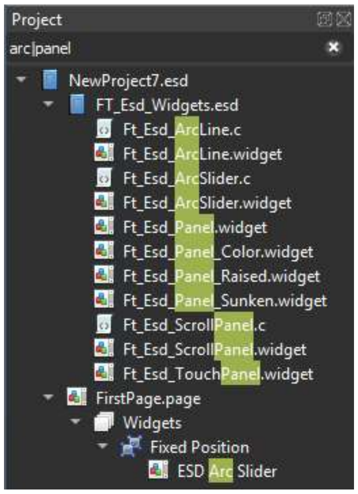


Figure 36 - Project Explorer Filter

Screen Layout Editor

The *Screen Layout Editor* displays the rendering output of EVE and allows users to edit C source code. The page/widget/main files can be opened and edited in this editor.

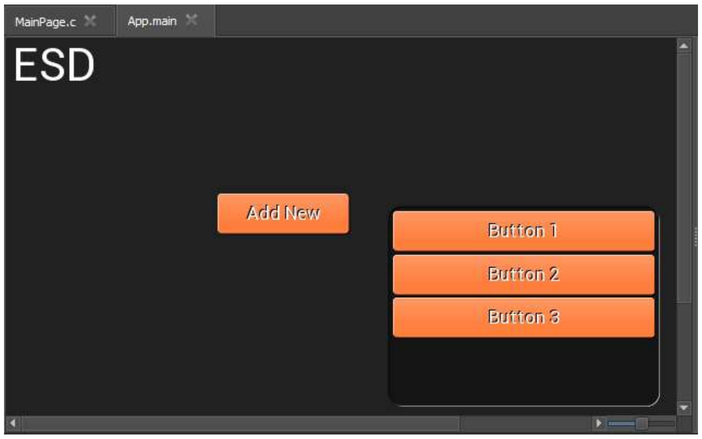
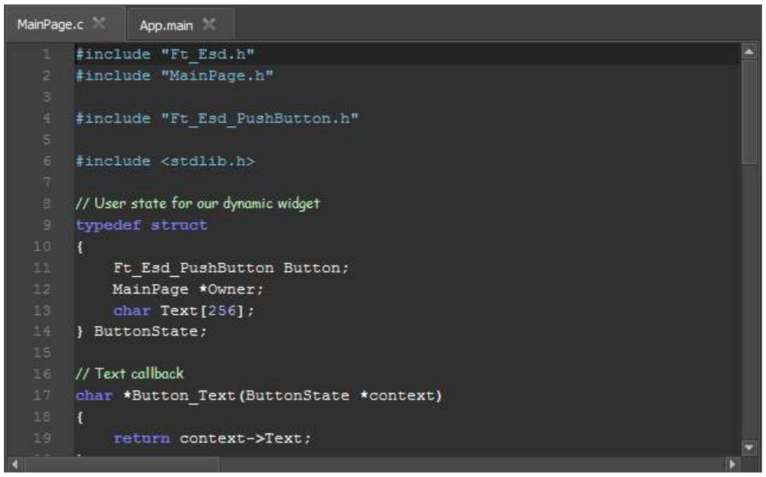


Figure 37 - Screen Layout Editor – App.main



```
#include "Ft_Esd.h"
#include "MainPage.h"

#include "Ft_Esd_PushButton.h"

#include <stdlib.h>

// User state for our dynamic widget
typedef struct
{
    Ft_Esd_PushButton Button;
    MainPage *Owner;
    char Text[256];
} ButtonState;

// Text callback
char *Button_Text(ButtonState *context)
{
    return context->Text;
```

Figure 38 - Screen Layout Editor - MainPage.c (C Source Code)

It shares one view port with the logic node editor. Users can adjust the size of the screen layout editor by dragging the splitter handle.

Users can drag and drop the widgets from the library browser to form the layout when simulation mode is *"off"*. For the other kinds of logic nodes, if the EVE rendering process is not defined, screen layout editor does not allow them to be dropped in.

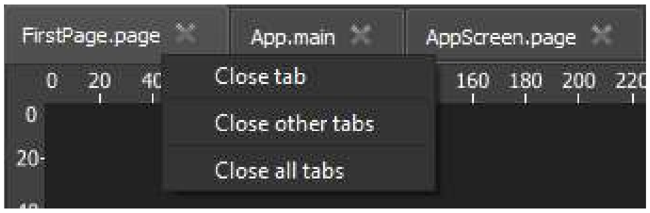


Figure 39 - Context menu on Tab bar

Users can right click on tab bar to active a context menu. Those menu items are described in table below.

Menu	Description
Close tab	Close the current tab
Close other tabs	Close all tabs except the current tab
Close all tabs	Close all tabs that were opened

Logic Node Editor

The *Logic Note Editor* allows users to layout logic nodes and to establish connections to create logic maps. Users can drag and drop a logic node from the Library Browser component to the Logic Note Editor in order to create connections.

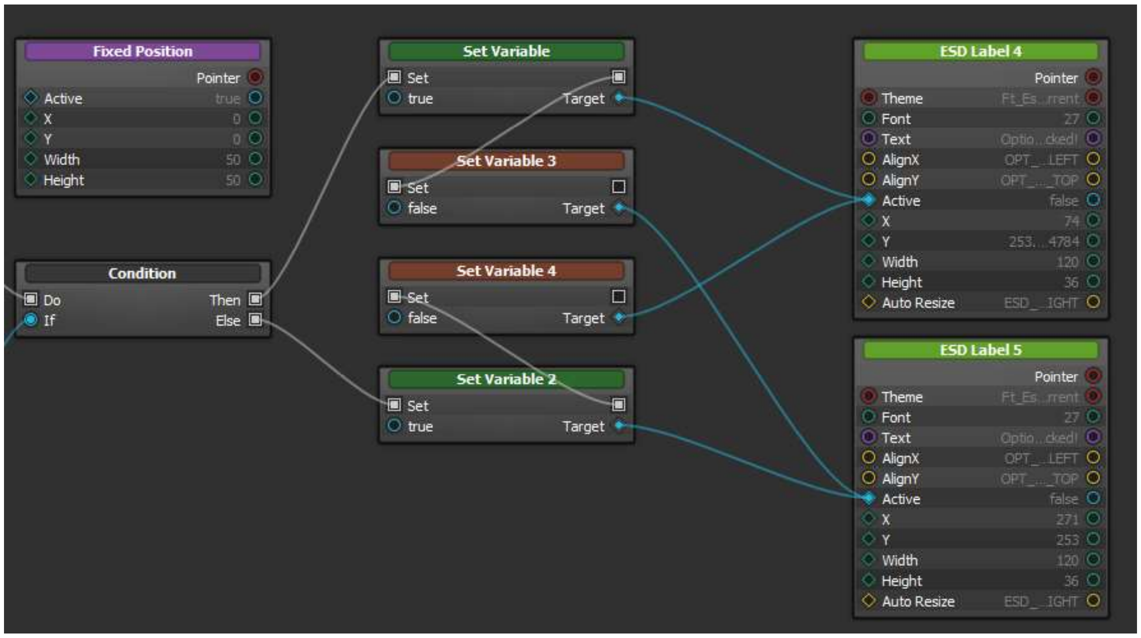


Figure 40 - Logic Node Editor

Library Browser

The *Library Browser* allows the storing of all the available logic nodes and resources in ESD that includes both built-in and user-defined ones. Users can view these logic nodes by category and select one for their project.

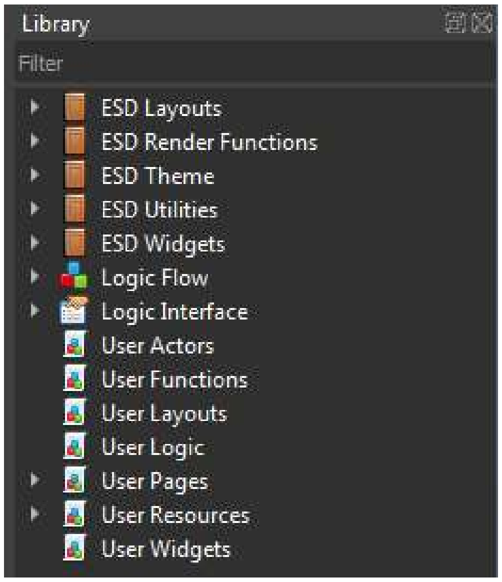


Figure 41 - Library Browser

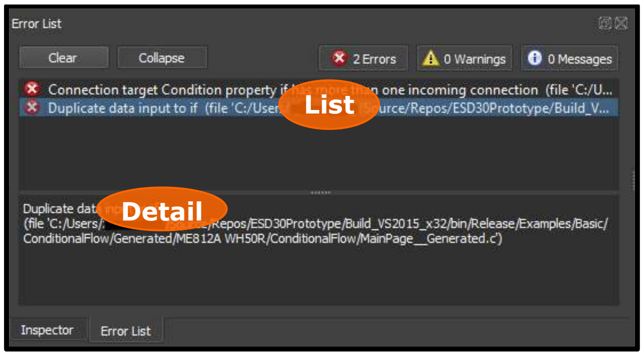
The following table provides the ESD library name and its description –

Library Name	Description
ESD Layouts	ESD built-in layout widgets and relevant utilities
ESD Render Functions	ESD built-in Render functions
ESD Theme	ESD built-in basic theme manipulation functions
ESD Utilities	ESD built-in utilities
ESD Widgets	ESD built-in widgets
Logic Flow	ESD built-in logic node for control flow
Logic Interface	ESD built-in logic node for interface
User Actors	User defined actor logic node. It is empty by default.
User Functions	User defined functions. It is empty by default.
User Layouts	User defined layouts. It is empty by default.
User Logic	User defined logics. It is empty by default.
User Pages	Pages added by user. It is empty by default if no page is created.
User Resources	Resources added by users (For example: bitmap). It is empty by default.
User Widgets	Widgets created and added by users. It is empty by default.

Table 4 - ESD Libraries

Error List

The *Error List* is a dock window which shows the message output from ESD, while saving and recompiling the project files. Any error message displayed in this window indicates that the generated source code for the current project is unable to be executed successfully. Users need to double check the logic defined in *logic node editor* or the user-defined source code in the project.



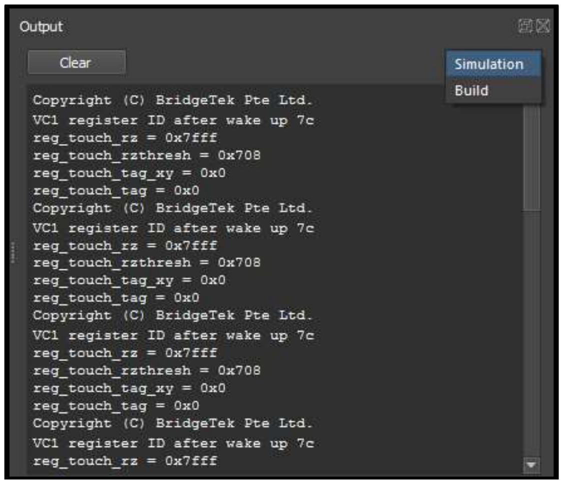
```
Connection target Condition property if has more than one incoming connection  (file 'C:/U...
Duplicate data input to if  (file 'C:/Users/[redacted]/Source/Repos/ESD30Prototype/Build_V...
```

```
Duplicate data input to if
(file 'C:/Users/[redacted]/Source/Repos/ESD30Prototype/Build_VS2015_x32/bin/Release/Examples/Basic/
ConditionalFlow/Generated/ME812A WH50R/ConditionalFlow/MainPage__Generated.c')
```

Figure 42 - Error List Window

Output

The *Output* component is a docked window which shows the message output from the EVE emulator and Toolchain.



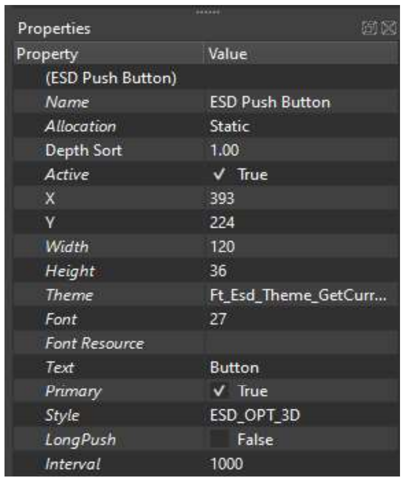
```
Copyright (C) BridgeTek Pte Ltd.
VC1 register ID after wake up 7c
reg_touch_rz = 0x7fff
reg_touch_rzthresh = 0x708
reg_touch_tag_xy = 0x0
reg_touch_tag = 0x0
Copyright (C) BridgeTek Pte Ltd.
VC1 register ID after wake up 7c
reg_touch_rz = 0x7fff
reg_touch_rzthresh = 0x708
reg_touch_tag_xy = 0x0
reg_touch_tag = 0x0
Copyright (C) BridgeTek Pte Ltd.
VC1 register ID after wake up 7c
reg_touch_rz = 0x7fff
reg_touch_rzthresh = 0x708
reg_touch_tag_xy = 0x0
reg_touch_tag = 0x0
Copyright (C) BridgeTek Pte Ltd.
VC1 register ID after wake up 7c
reg_touch_rz = 0x7fff
```

Figure 43 - Output Window

To check a message from the EVE emulator, click **"Simulation"** from the drop-down list. To check a message from the EVE Toolchain while building generated source code, click **"Build"** from the drop-down list. This window is automatically updated during simulation or building.

Property Editor

The *Property Editor* allows user to edit the properties of selected logic nodes. The sample screenshot given below shows the property editor of an ESD Push button.



Properties

Property	Value
(ESD Push Button)	
Name	ESD Push Button
Allocation	Static
Depth Sort	1.00
Active	✓ True
X	393
Y	224
Width	120
Height	36
Theme	Ft_Esd_Theme_GetCurr...
Font	27
Font Resource	
Text	Button
Primary	✓ True
Style	ESD_OPT_3D
LongPush	False
Interval	1000

Figure 44 - Property Editor (ESD Push Button)

Status bar

The *Status bar* is found at the bottom of the user interface. It primarily displays the current status of any process or job that is being handled by the application. For example, if the user is performing a C compilation, then the compilation status of the generated C code is displayed.

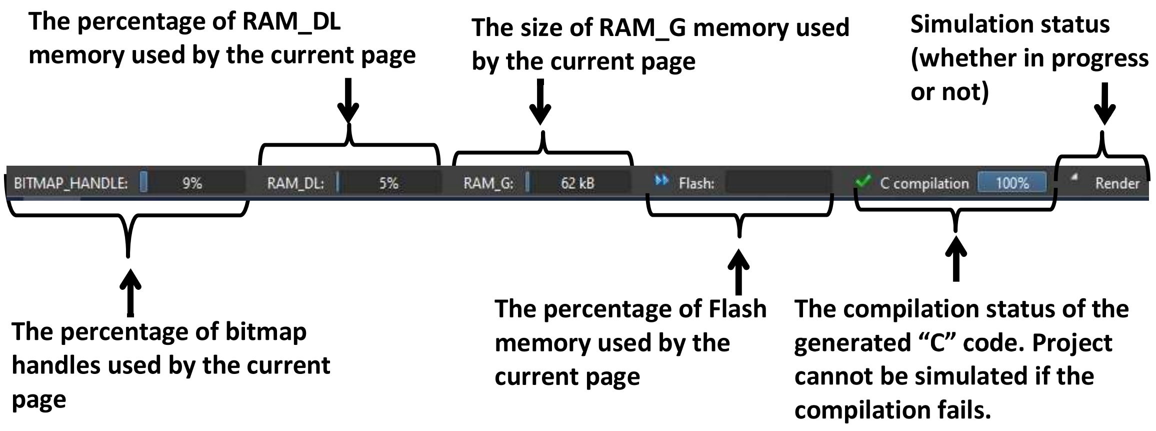

The percentage of RAM_DL memory used by the current page
The size of RAM_G memory used by the current page
Simulation status (whether in progress or not)
BITMAP_HANDLE: 9%
RAM_DL: 5%
RAM_G: 62 kB
Flash:
C compilation 100%
Render
The percentage of bitmap handles used by the current page
The percentage of Flash memory used by the current page
The compilation status of the generated "C" code. Project cannot be simulated if the compilation fails.

Figure 45 - Status bar

Application Project Structure

An application consists of one or more pages. Each page in turn may contain one or more widgets. Widgets and pages are connected via connection lines so that the screen logic is defined.

The application model follows a strict hierarchical structure where all the pages are owned and managed by the application, and all the widgets are owned by their respective pages. The visibility and memory lifetime of the widgets are thus managed on a page-by-page basis.

By design, all widgets and pages are generated to be entirely modular and self-sufficient. Any interaction between the widgets and pages is therefore required to be routed through the hierarchical chain. (Only node connections exist between the directly connected nodes in the hierarchy.)

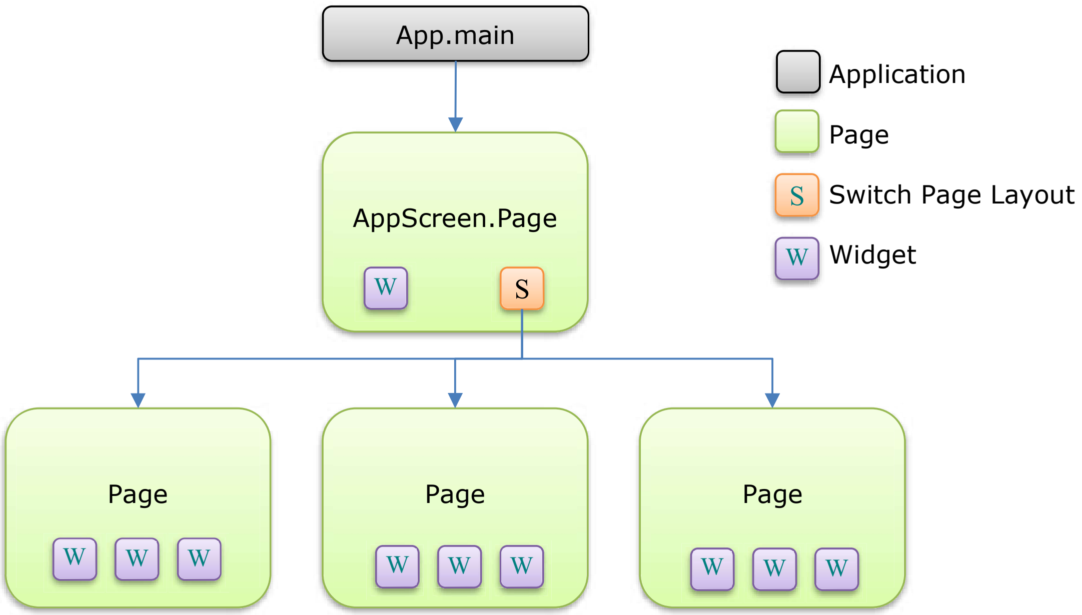


Figure 46 - Application Project Structure

File Name	Description
*.esd	Project file - This XML file describes what files are included in this project. This file is unique for the whole project.
*.main	Logic file - This XML file describes the application-level logic. By default, it is named as "App.main" and is unique project wide.
*.page	Page Node Definition file - This XML file describes what widgets are contained and connected in each page. By default: "AppScreen.Page" is added, where a switch page may be included for control on page transition. A page can contain other pages, but switch page is required for page transition control.
*.widget	Widget Node Definition file. This XML file describes a widget detail. A widget can contain other nodes, which may include widget node, actor node and logic node.
*.actor	Actor Node Definition file. This XML file describes the actor node details.
*.logic	Logic Object Definition file. This XML file describes the logic node details.
*.theme	Application Theme Definition file. This XML file describes the theme detail.

*.h	C language source header file
*.c	C language source body file
*.png, *.jpg, *.jpeg	Image resource files
*.anim	Animation file converted by EAB tool

Note: The file names listed above shall **NOT** be conflicting with the C standard library function as well as windows header files. Here are some examples: *fopen, windows, winbase, math, printf, realloc.* ESD simulation process may give errors and fail, if user gives the file name of their widgets, logics, images, or any resources with these reserved names. Therefore, ESD enforces the 'Usr_' prefix to all the new/renamed widgets, pages, functions, actors, as well as source file, theme, logic source file etc, in order to ensure that the simulation process succeeds without error.

Application Workflow

The workflow given below illustrates how to create an EVE based GUI application:

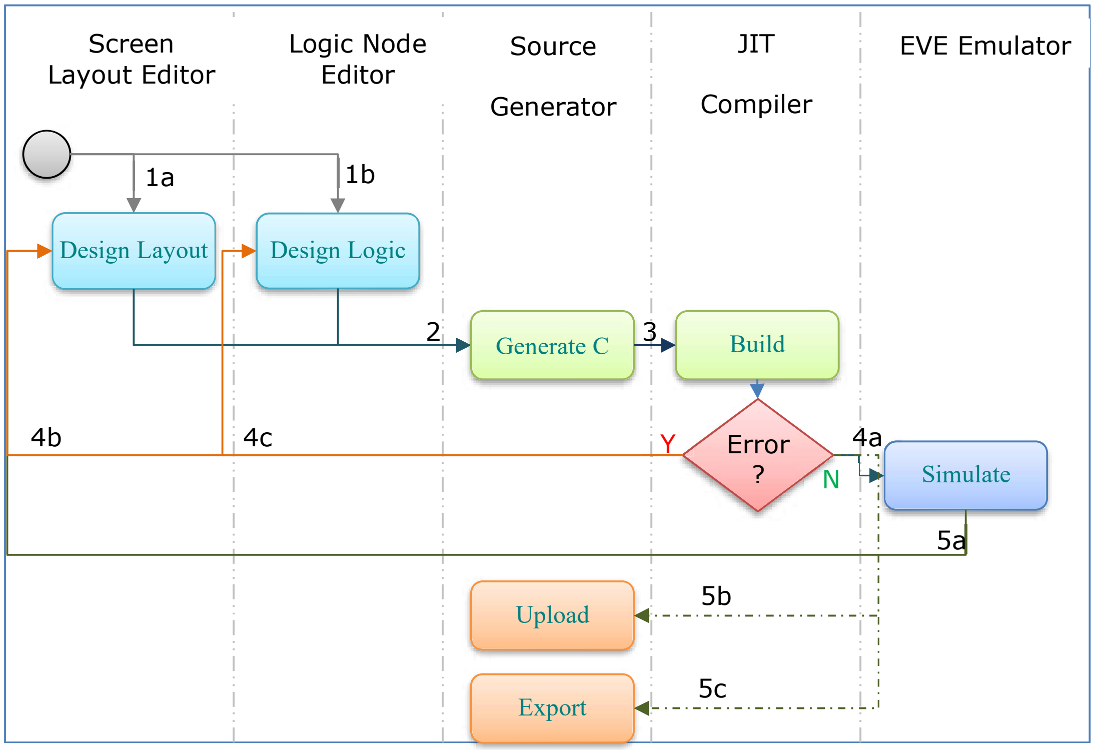


Figure 47 - Application Workflow

The 5 main modules that comprise the application workflow are the *Screen Layout Editor, Logic Node Editor, Source Generator, JIT Compiler(TinyCC)* and *EVE Emulator*. Among these, *Source Generator*, *JIT Compiler* and *EVE Emulator* are the back-end modules which have no UI exposed to end user. The compiler is a JIT style C compiler, i.e., TinyCC compiler, which compiles and links the generated code into executable. The executable is invoked by the emulator module for simulation.

The application workflow starts by design layout (1a) or design logic (1b); Upon user saving the changes, it will trigger C code generation (2), then the compiler will try to build the generated c code; if any error encountered during building, then user needs to continue on design layout (4b) or design logic (4c); or if no error found, then the emulator will start simulating (4a) the project. Users will be able to upload (5b) the binary to the EVE chips or export (5c) the project as eclipse C project.

Design Layout using Layout Editor

Users need to determine how many pages are included in the project and what widgets are to be contained in each page. After a page is created, users can drag and drop the widgets to design the appearance of the page in the screen layout editor. This process determines how many screens are shown in the project and how they look like together.

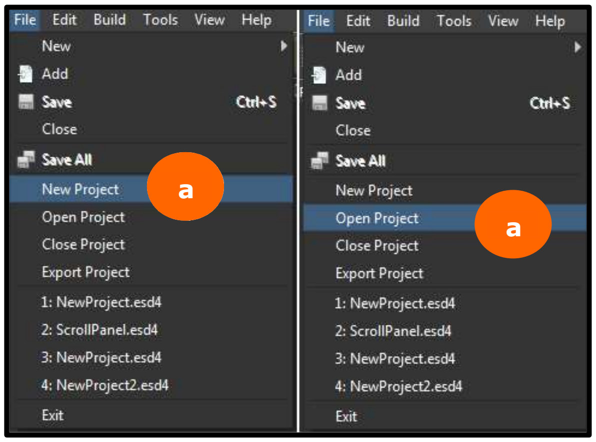
Custom Widget

ESD provides a set of built-in widgets for users to construct screen layouts. Additionally, if the built-in widgets do not meet the design requirement, users may define and design custom widgets.

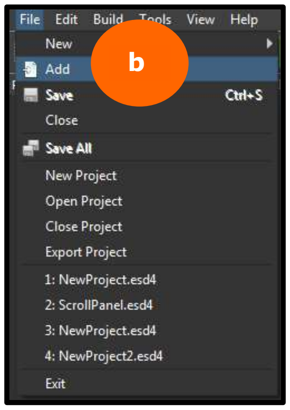
Add Bitmap Resource

To display a bitmap on the page, users need to add a bitmap resource into the project. The following steps will provide guidance on how to add a bitmap resource:

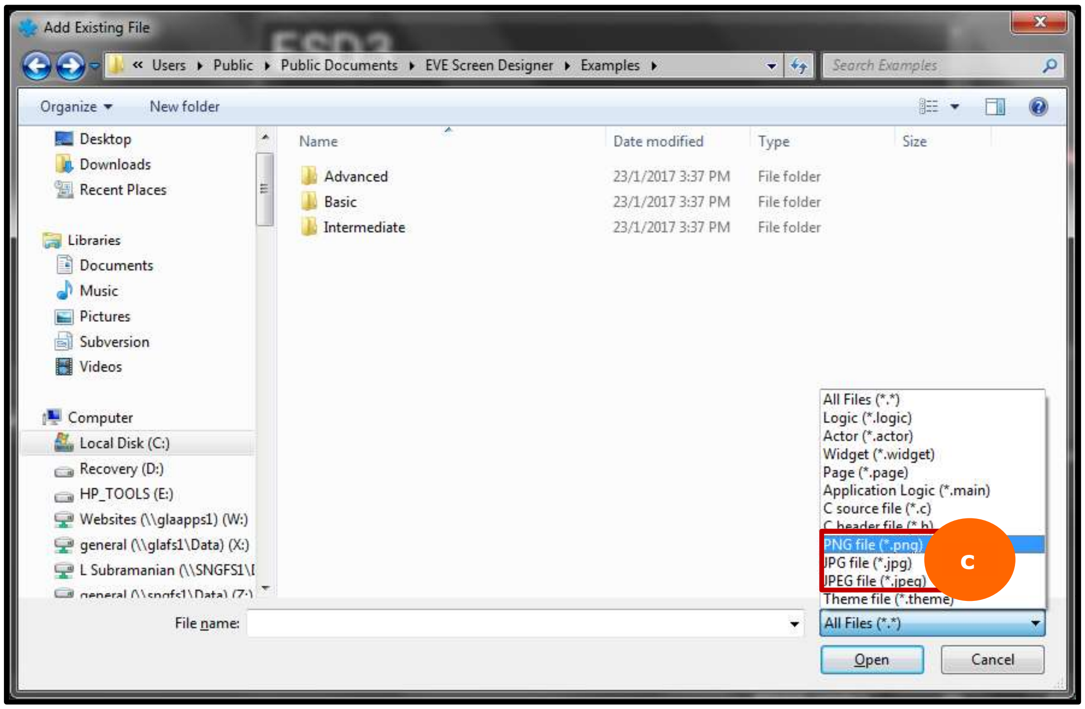
a. Create a New Project ***(File → New Project)*** or open an existing Project ***(File → Open Project)***.



b. Click **File → Add** from the menu.



c. Browse and select the image file(s) from the file explorer window.

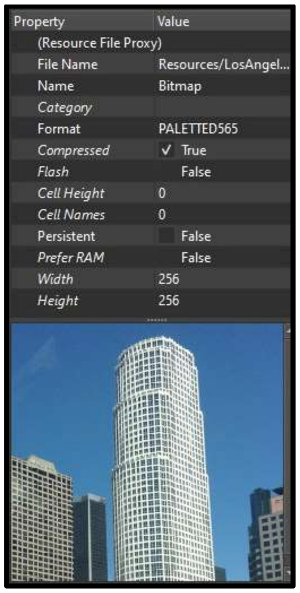


Name	Date modified	Type	Size
Advanced	23/1/2017 3:37 PM	File folder	
Basic	23/1/2017 3:37 PM	File folder	
Intermediate	23/1/2017 3:37 PM	File folder	

Upon adding the file, users can drag and drop the ESD Image widget to the desired page and select the bitmap cell (which is default at cell 0) accordingly.

Configure Bitmap Resource

Users can configure a bitmap resource via the *Property Editor*. Select the image file from the project browser and check the Property Editor.



Property	Value
(Resource File Proxy)	
File Name	Resources/LosAngel...
Name	Bitmap
Category	
Format	PALETTED565
Compressed	✓ True
Flash	False
Cell Height	0
Cell Names	0
Persistent	False
Prefer RAM	False
Width	256
Height	256

Figure 48 - Property Editor

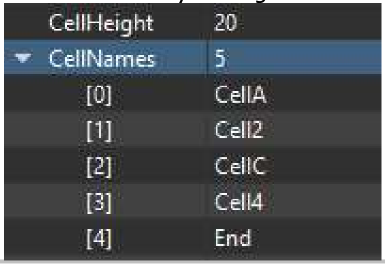
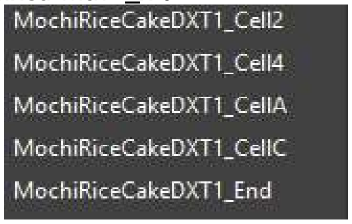
Property	Value/Description
File Name	Name of the image file
Name	Resource Type Name. In this case, the selected resource type is Bitmap.
Category	Category of the selected object
Format	Image Format – The target format of bitmap can be – • L1/L2/L4/L8 • RGB332/RGB565 • ARGB1555/ARGB2/ARGB4 • PALETTED4444/PALETTED565/PALETTED8 • DXT1 • JPEG (Decoded into RGB565 format only) • PNG (Decoded into RGB565 format only) • COMPRESSED_RGBA_ASTC_4x4_KHR • COMPRESSED_RGBA_ASTC_5x4_KHR • COMPRESSED_RGBA_ASTC_5x5_KHR • COMPRESSED_RGBA_ASTC_6x5_KHR • COMPRESSED_RGBA_ASTC_6x6_KHR • COMPRESSED_RGBA_ASTC_8x5_KHR • COMPRESSED_RGBA_ASTC_8x6_KHR • COMPRESSED_RGBA_ASTC_8x8_KHR • COMPRESSED_RGBA_ASTC_10x5_KHR • COMPRESSED_RGBA_ASTC_10x6_KHR • COMPRESSED_RGBA_ASTC_10x8_KHR • COMPRESSED_RGBA_ASTC_10x10_KHR • COMPRESSED_RGBA_ASTC_12x10_KHR • COMPRESSED_RGBA_ASTC_12x12_KHR
Compressed	Set bitmap compression on/off
Flash	Set this to store the resource in the Flash
CellHeight	The pixel number of one cell. For example, if the bitmap width is 240 and height is 240, then it can be defined as two cells with "CellHeight" - 120 or four cells with "CellHeight" – 60. Cell Height default at 0 which means there will be only one cell no matter how large the bitmap it is.
CellNames	This field will enable system to create a list of n number of user defined cell names instead system generated which is suffixed with "_x".  CellHeight 20; CellNames 5; [0] CellA; [1] Cell2; [2] CellC; [3] Cell4; [4] End  MochiRiceCakeDXT1_Cell2; MochiRiceCakeDXT1_Cell4; MochiRiceCakeDXT1_CellA; MochiRiceCakeDXT1_CellC; MochiRiceCakeDXT1_End
Persistent	Keeps the memory persistent even if the bitmap is not displayed when it is true.
Width	The width of bitmap image in pixels
Height	The height of bitmap image in pixels
Prefer RAM	For ASTC images, resources can be rendered directly from Flash without the need to store it to RAM_G. If Flash flag is True to load ASTC image from Flash directly. If Flash flag is False, Resources will be rendered from RAM_G. **Prefer RAM feature is exclusively available for *ASTC* format alone.**

Table 5 - Bitmap Resource Properties

Design Screen Logic using Logic Node Editor

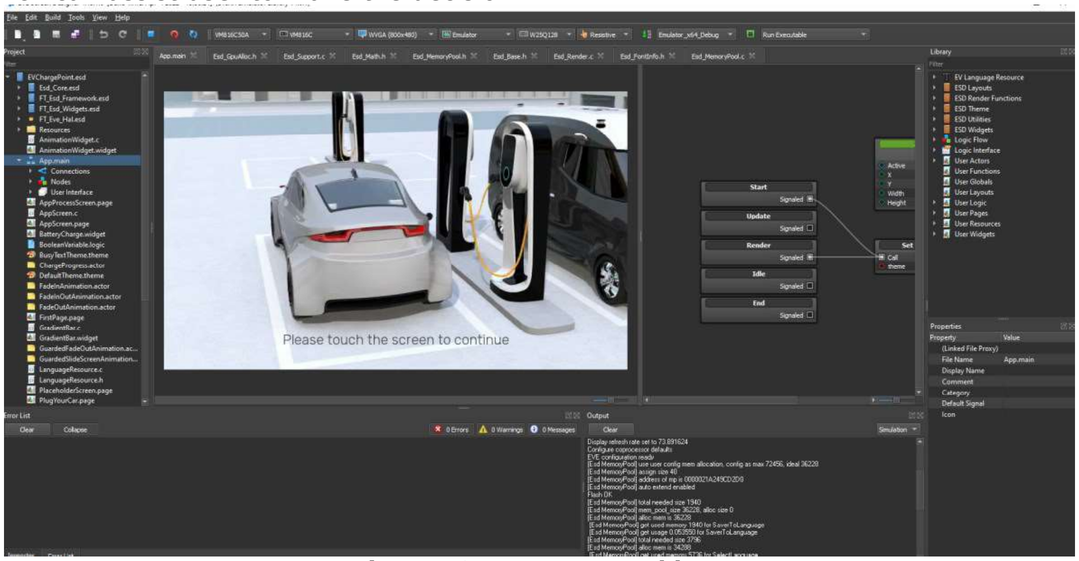
Users can define the dynamic behaviour of an application in this stage. The Logic Node Editor employs the innovative visual programming idea to help users define logic flows without coding.

During this stage, users can decide the behaviour mode of both inner-page and inter-pages. For inner-page behaviour, users can connect the widgets on the page via the logic node editor adding more logic nodes from the library browser, if necessary.

Inter-page logic behaviour is captured in the application logic, which is normally named as "AppScreen.page". Users are required to drag and drop the predefined pages into the logic node editor and connect these pages via the logic interface. By establishing connections between these pages using the logic interface, the behaviour that transcends page boundaries, such as screen logic, is effectively defined.

Evaluate and choose the memory usage of the project

As mentioned earlier, ESD allows users to configure heap memory usage in root project properties. It is necessary to decide the memory usage before uploading the image to hardware. There are several useful debug messages which will be printed in debug mode which can assist user to make the decision.



```
Display refresh rate set to 73.891624
Configure coprocessor defaults
EVE configuration ready
[Esd MemoryPool] use user config mem allocation, config as max 72456, ideal 36228
[Esd MemoryPool] assign size 40
[Esd MemoryPool] address of mp is 0000021A249CD2D0
[Esd MemoryPool] auto extend enabled
Flash OK
[Esd MemoryPool] total needed size 1940
[Esd MemoryPool] mem_pool_size 36228, alloc size 0
[Esd MemoryPool] alloc mem is 36228
 [Esd MemoryPool] get used memory 1940 for SaverToLanguage
 [Esd MemoryPool] get usage 0.053550 for SaverToLanguage
[Esd MemoryPool] total needed size 3796
[Esd MemoryPool] alloc mem is 34288
```

Figure 49 - memory pool logs

The above figure illustrates the use of keyword [Esd MemoryPool] for decision making.

Simulation

To validate the behaviour of the screen logic, users may need to simulate the project on the PC before compiling and downloading the generated C code into the target device. When application logic (AppScreen.page or App.main) is selected, users can simulate the whole project by clicking **[Simulation]** button on the toolbar. When the application is in simulation mode, users cannot drag and drop the widgets into the layout editor. The PC mouse will act as a touch stylus on the touch panel to interact with the application directly.

To validate the behaviour of single page, users need select the page file and open it with logic node editor before clicking the **[Simulation]** button on the toolbar. Similarly, users can simulate the behaviour of other logic node, such as widget, actor by opening it with logic node editor and clicking **[Simulation]** button on the toolbar.

Before performing a simulation, it is strongly recommended to save the current project by clicking **[Save all]** button. Once the simulation is completed, ensure to switch off the simulation mode by clicking the **[Simulation]** button again.

Simulation State	Description
▶	ON State - Indicates that the simulation is in progress. Clicking on it will switch off the simulation. In this state, users will not be able to edit the page or project.
▶	OFF State - Indicates that currently simulation is not in progress. Clicking on it will switch on the simulation.

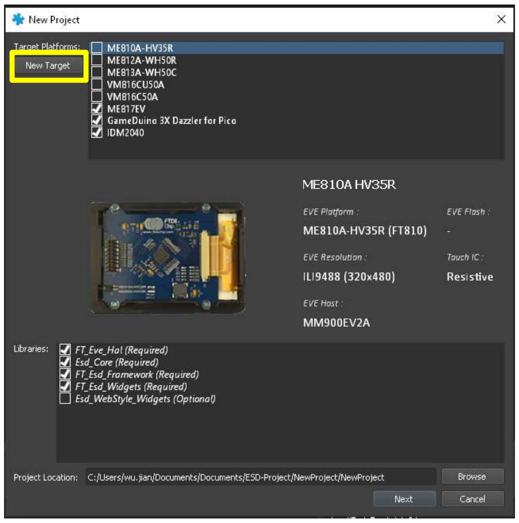
Target selection

ESD provides support for the following built-in build targets -

1. ME810A-HV35R
2. VM810C50A
3. ME812A WH50R
4. ME812AU-WH50R
5. ME813A-WH50C
6. ME813AU WH50C
7. VM816CU50A
8. VM816C50A
9. ME817EV-WH10C
10. ME817EV-WH70C
11. Gameduino 3X Dazzler for Pico
12. IDM2040EV-7A
13. IDM2040EV-43A
14. STM32F4

These build targets typically come with standard configuration which the user can modify within certain limits as per their requirements.

ESD also provides a mechanism to create new custom targets for new ESD projects. For example, if the user wants to create a new build target "PC_BT815EMU_WVGA" with PC as Host platform, BT815 as EVE platform, Screen Resolution as WVGA (800 X 480) and Flash set to 16MB MX25L16. Refer to the Figure 50 and Figure 51.



- [ ] ME810A-HV35R
- [ ] ME812A-WH50R
- [ ] ME813A-WH50C
- [ ] VM816CU50A
- [ ] VM816C50A
- [✓] ME817EV
- [✓] GameDuino 3X Dazzler for Pico
- [✓] IDM2040

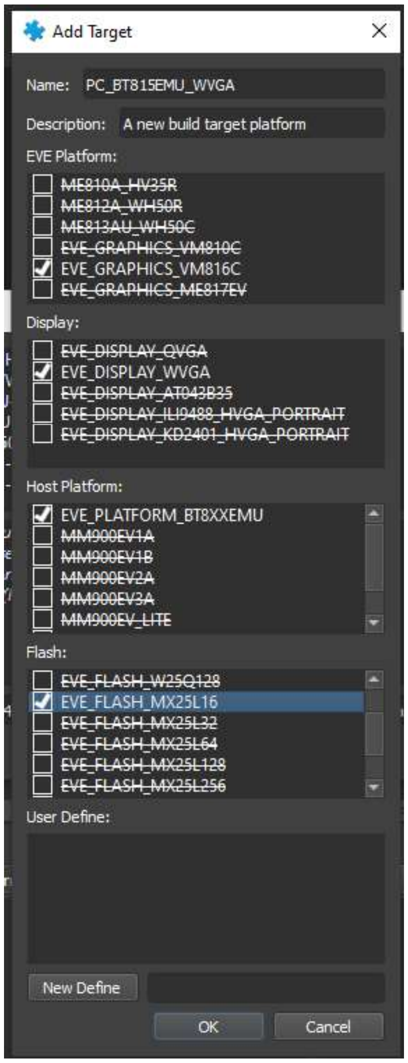
ME810A HV35R

EVE Platform : ME810A-HV35R (FT810)
EVE Flash : -
EVE Resolution : ILI9488 (320x480)
Touch IC : Resistive
EVE Host : MM900EV2A

- [✓] FT_Eve_Hal (Required)
- [✓] Esd_Core (Required)
- [✓] FT_Esd_Framework (Required)
- [✓] FT_Esd_Widgets (Required)
- [ ] Esd_WebStyle_Widgets (Optional)

Project Location: C:/Users/wu.jian/Documents/Documents/ESD-Project/NewProject/NewProject

Figure 50 - New Target button


Add Target

Name: PC_BT815EMU_WVGA

Description: A new build target platform

EVE Platform:
- [ ] ~~ME810A_HV35R~~
- [ ] ~~ME812A_WH50R~~
- [ ] ~~ME813AU_WH50C~~
- [ ] ~~EVE_GRAPHICS_VM810C~~
- [✓] EVE_GRAPHICS_VM816C
- [ ] ~~EVE_GRAPHICS_ME817EV~~

Display:
- [ ] ~~EVE_DISPLAY_QVGA~~
- [✓] EVE_DISPLAY_WVGA
- [ ] ~~EVE_DISPLAY_AT043B35~~
- [ ] ~~EVE_DISPLAY_ILI9488_HVGA_PORTRAIT~~
- [ ] ~~EVE_DISPLAY_KD2401_HVGA_PORTRAIT~~

Host Platform:
- [✓] EVE_PLATFORM_BT8XXEMU
- [ ] ~~MM900EV1A~~
- [ ] ~~MM900EV1B~~
- [ ] ~~MM900EV2A~~
- [ ] ~~MM900EV3A~~
- [ ] ~~MM900EV_LITE~~

Flash:
- [ ] ~~EVE_FLASH_W25Q128~~
- [✓] EVE_FLASH_MX25L16
- [ ] ~~EVE_FLASH_MX25L32~~
- [ ] ~~EVE_FLASH_MX25L64~~
- [ ] ~~EVE_FLASH_MX25L128~~
- [ ] ~~EVE_FLASH_MX25L256~~

User Define:

Figure 51 - Configure custom target

The newly created target can then be set via build target. Refer to the screenshot given below:

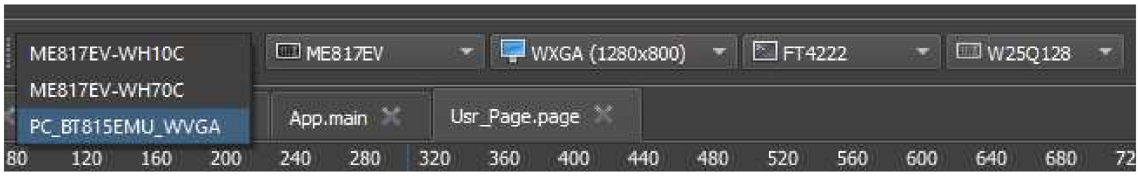


Figure 52 - Select custom build target

Build & Upload

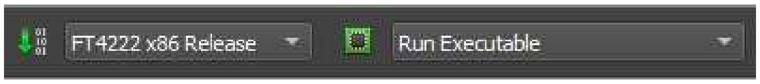
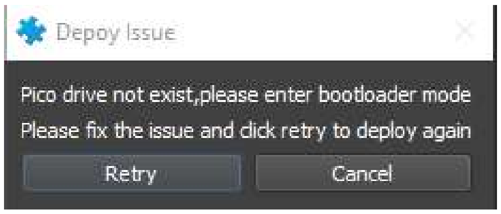


Figure 53 - Build & Upload

"**Build Executable**" button builds a project.

"**Build and Upload to Hardware**" button will invoke the compiler and programmer, so that the current project can be build and uploaded into the target hardware. If user selects x64 toolchain, this button will build and generate an '.exe' application which can be run on a PC as host environment.

For Pico RP2040 platform, ESD will set up the build environment and trigger cmake and nmake to build Pico project. An Uf2 image file will be generated and flashed to the Pico MCU, so user needs to enter Pico flash mode before triggering this option. If user has failed to do this, a dialog will pop up and requesting user to enter flash mode.


Depoy Issue

Pico drive not exist,please enter bootloader mode
Please fix the issue and click retry to deploy again

Figure 54 - Pico drive not connected alert

The respective output files generated by "Build Executable" for different build configuration and host platforms are shown in Table 6.

Build Target	Host Platform	Output files
ME810A-HV35R	FT90x toolchain	*.bin; *.elf
	Pico toolchain	*.uf2
VM810C50A	X64 toolchain (MPSSE)	*.exe
ME812A WH50R	FT90x toolchain	*.bin; *.elf
	Pico toolchain	*.uf2
ME812AU-WH50R	X64 toolchain (FT4222)	*.exe
ME813A WH50R	FT90x toolchain	*.bin; *.elf
	Pico toolchain	*.uf2
ME813AU WH50C	X64 toolchain (FT4222)	*.exe
VM816CU50A	X64 toolchain (FT4222)	*.exe
VM816C50A	X64 toolchain (MPSSE)	*.exe
ME817EV-WH10C	FT90x toolchain	*.bin; *.elf
	Pico toolchain	*.uf2
	X64 toolchain (FT4222)	*.exe
ME817EV-WH70C	FT90x toolchain	*.bin; *.elf
	Pico toolchain	*.uf2
	X64 toolchain (FT4222)	*.exe
Gameduino 3X Dazzler for Pico	Pico toolchain	*.uf2
IDM2040EV-7A	Pico toolchain	*.uf2
IDM2040EV-43A	Pico toolchain	*.uf2
STM32F4	STM32 toolchain	*.elf; *.bin

Table 6 - Output files for different Build Configuration & Host Platforms

Export

Upon completing the screen design, it’s time to move the project to the next phase of the workflow – export project. There are four ways of exporting a project. They are **“Export as Eclipse Project”, “Export as MSVC Project”, “Export as STM32CubeIDE Project”** and **“Export as Pico Project”**. This function is used to prepare all the necessary files for the MCU tool chain so that users can make further enhancements such as connecting sensors or external hardware to a full HMI solution.

On exporting, ESD copies the generated “C” source code, the necessary bitmap resources and the ESD application framework code into a user defined folder. In ESD, all the generated files are named as “*_generated.c”. For eclipse export projects, “.cproject” and “.project” files are generated so that the users can open the project in FT90X eclipse IDE[3]. For MSVC export projects, “Project” folder is generated containing the MSVC compatible project. For Pico export projects, “CMakeLists.txt” and “pico_sdk_import.cmake” are generated for user to build the project. Additionally, an optional batch file “pico_build.cmd” is also generated. This allows the user to build Pico export project in a single step. And in order to use this script, system environment variable PICO_SDK_PATH needs to be set as the pico-sdk path. Another script “pico_flash.cmd” will be generated only if ESD project has assets in Flash, run this script will flash both the flash image and application into Pico board, Pico board needs to be under flash mode before running this script. Table 7 gives more information on the Build Targets, Host platform and the corresponding export projects that can be created.

Build Target	Host Platform	Export Project
ME810A-HV35R	Emulator	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
	MM2040EV	Pico Project
VM810C50A	Emulator, MPSSE*	MSVC Project
ME812A-WH50R	Emulator	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
	MM2040EV	Pico Project
ME812AU-WH50R	Emulator, FT4222*	MSVC Project
ME813A WH50C	Emulator	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
	MM2040EV	Pico Project
ME813AU WH50C	Emulator, FT4222	MSVC Project
VM816CU50A	Emulator, FT4222	MSVC Project
VM816C50A	Emulator, MPSSE*	MSVC Project
ME817EV-WH10C	Emulator, FT4222	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
	MM2040EV	Pico Project
ME817EV-WH70C	Emulator, FT4222	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
	MM2040EV	Pico Project
Gameduino 3X dazzler	Emulator	MSVC Project

[3] https://brtchip.com/ft9xx-toolchain/

	Raspberry pi Pico	Pico Project
IDM2040EV-7A	Emulator	MSVC Project
	IDM2040EV	Pico Project
STM32F4	STM32Cube	STM32 Project

Table 7 - Build Targets, Host platform and the corresponding export projects

(* Requires separate MPSSE adapter or FT4222H circuit)

Exported Folder Structure

Table 8 provides the list of folders and files that are created upon exporting the project:

Folder / Files	Description
	(Common export files)
Esd_Core	Contains the application core files. ⚠ **It is recommended not to make any changes to this folder/ files**
FT_Esd_Framework	Contains the application framework files. ⚠ **It is recommended not to make any changes to this folder/ files**
FT_Esd_Widgets	Contains the widget files ⚠ **It is recommended not to make any changes to this folder/ files**
FT_Esd_Hal	Contains the hardware abstraction layer files. These files may be changed to support a different MCU.
Data	Contains the converted bitmap resource data and the flash data which is used by the current project
$(ProjectName)	Contains the application specific source code
	(Eclipse export files)
.cproject (for Eclipse export)	Eclipse project file
.project (for Eclipse export)	Eclipse project file
ThirdPartyLib	Contains third party files such fat filesystem library handler files
	(MSVC export files)
Project/$(ProjectName)_MSVC/ $(ProjectName)_MSVC.sln	MSVC project solution file
Project/$(ProjectName)_MSVC/ $(ProjectName)_MSVC /$(ProjectName)_MSVC.vcxproj	MSVC visual studio C++ project file
	(Pico export files)
CMakeLists.txt(for Pico export)	cmake file
pico_sdk_import.cmake	cmake file
ThirdPartyLib	Contains third party files such fat filesystem library handler files
pico_build.cmd	Build batch to trigger Pico build
pico_flash.cmd	Batch to flash both assets and application
	(STM32 export files)
.cproject (for STM32 export)	STM32Cube project file
.project (for STM32 export)	STM32Cube project file
ThirdPartyLib	Contains third party files such fat filesystem library handler files
Core Folders	STM32 driver startup main files
Drivers Folder	STM32F4xx HAL Driver files

Table 8 - List of Folder & File created upon exporting project

Assets

The assets typically include various resources and data files that are bundled with the application to support its functionality and appearance.

Bitmap Resources

The bitmap resources used in an ESD project can be saved in a SD card or to flash based on the selection of flash flag configuration in the ESD.

.astc : The ASTC Bitmap Format for images, there is requirement to copy the .astc file to the SD card or Flash directory.

.z : The .z Bitmap Format, used for compressed images, necessitates copying the .z file to either the SD card or the Flash directory.

SD card

If Bitmap resources meant for SD card are used in a project, it is converted to *.astc (for ASTC file format)/*.raw (for other file formats) or *.bin file format (if compressed flag is selected). Users are required to download the converted bitmap resource into an SD card **root** directory and insert the SD card into the hardware module.

The converted bitmap resource is located at:

$(ExportFolder)\data

The bitmap resource file with .bin extension is called to decompress using the EVE engine command *CMD_INFLATE*.

As per the Hardware Platform requirements, the SD card must be formatted as a FAT32 file system.

Flash

If bitmap resources meant for flash are used in a project, it is converted to a Flash.bin file. Users are required to download the converted bitmap resource into flash memory of the hardware module.

The converted bitmap resource is located at:

$(ExportFolder)\data

The bitmap resource file with .bin extension is called to decompress using the EVE engine command *CMD_INFLATE*.

Resource Metadata

To support dynamically changing and loading new assets at runtime without binary recompilations due to hardcoded asset information, EVE Screen Designer generates additional metadata files with the file extension **.esdm** appended to the full original filename for each converted resource.

The contents of the metadata file correspond to the contents of the Esd_BitmapInfo, Esd_ResourceInfo, Esd_FontInfo, and Esd_AnimationInfo data structures, packed in a fixed format to fit within 64 bytes. This size restriction allows the metadata to fit within the directory record of the LittleFS filesystem when in use with flash storage, and simplifies loading.

Common Resource Metadata Format:

The first 12 bytes contain the common fields used for all assets.

Identifier	Offset	Type	Name	Description
ESD_METADATA_SIGNATURE	0	uint32	Signature	Serialized byte sequence equals "RES", "ANI", "BMP" or "FNT", NUL-terminated
ESD_METADATA_VERSION	4	uint8	Version	Version of the metadata format
ESD_METADATA_SIZE	5	uint8	Size	Size of the metadata structure
ESD_METADATA_COMPRESSION	6	uint8	Compression	The Compression option in ResourceInfo
ESD_METADATA_EXTLEN	7	uint8	Extension Length	The string length of the complete file extension of the resource
ESD_METADATA_RAWSIZE	8	uint32	Raw Size	The uncompressed size of the resource, RawSize in ResourceInfo

Bitmap Resource Metadata Format:

The bitmap resource metadata format builds on the common format, and has all the necessary data to populate the Esd_BitmapInfo structure.

Identifier	Offset	Type	Name	Description
ESD_METADATA_WIDTH	12 + 0	int32_t	Width	Display width in pixels
ESD_METADATA_HEIGHT	12 + 4	int32_t	Height	Display height in pixels
ESD_METADATA_STRIDE	12 + 8	int32_t	Stride	Stride of the bitmap data in bytes
ESD_METADATA_FORMAT	12 + 12	uint32_t	Format	Format of the bitmap data
ESD_METADATA_PALETTESIZE	12 + 16	uint16_t	Palette Size	Uncompressed size of the palette file
ESD_METADATA_PALETTEFILEEXT	12 + 18	uint8_t[10]	Palette File Extension	Filename suffix of the palette. Concatenated to the bitmap filename trimmed by Extension Length (Paletted)
ESD_METADATA_ADDTLRESEXT	12 + 28	uint8_t[12]	Additional Resource Extension	Filename suffix of the additional bitmap info resource. Concatenated to the bitmap filename trimmed by Extension Length (DXT1)
ESD_METADATA_CELLS	12 + 40	uint16	Cells	Number of cells usable by the user (up to 256 pages of 256 cells), excluding hidden cells (e.g., hidden DXT1 layers)

ESD_METADATA_SWIZZLE	12 + 42	uint16_t	Swizzle	Swizzle, same 12-bit format as BITMAP_SWIZZLE, plus one bit to enable the option

Animation Resource Metadata Format:

The animation resource metadata format specifies the address offsets that the animation was built with when assembling the flash image. It is currently only used to verify that the animation matches its current location in flash. The ESD build process adjusts these offsets on-the-fly while assembling the flash image. At this moment, ESD only supports displaying animations directly from flash.

Identifier	Offset	Type	Name	Description
ESD_METADATA_FRAMESPTR	12 + 0	uint32	Frames Ptr	The address of the frames file that the object file was built with. Refers to a RAM or flash address depending on whether the header object is loaded in RAM or flash
ESD_METADATA_FRAMESSIZE	12 + 4	uint32	Frames Size	Size of the display list frames file, if 0 both frames and atlas files are ignored
ESD_METADATA_ATLASBITMAPSOURCE	12 + 8	uint32	Atlas Bitmap Source	The address in bitmap source display list format (can be flash or RAM) of the atlas file that the frames file was built with
ESD_METADATA_ATLASSIZE	12 + 12	uint32	Atlas Size	Size of the ASTC bitmap atlas file, if 0 the atlas file is ignored
ESD_METADATA_RECTX	12 + 16	int16	Rect X	Display list bounds of the display list frames
ESD_METADATA_RECTY	12 + 18	int16	Rect Y	Display list bounds of the display list frames
ESD_METADATA_RECTWIDTH	12 + 20	int16	Rect Width	Display list bounds of the display list frames
ESD_METADATA_RECTHEIGHT	12 + 22	int16	Rect Height	Display list bounds of the display list frames

Font Resource Metadata Format:

Specifies font measurement characteristics that are not present in the font block, used by ESD to correctly align and centre fonts.

Identifier	Offset	Type	Name	Description
ESD_FONT_EXTENDED	12 + 0	uint8	Type	Type of font to load (Legacy or Extended)
FontHeight	12 + 4	uint8	First Char	First character in the glyph map (for legacy fonts)
BaseLine	12 + 8	uint16	Base Line	Baseline, distance from top of font glyph
CapsHeight	12 + 16	uint16	Caps Height	Caps height of font, measured from baseline
XOffset	12 + 20	uint16	X Offset	Offset of left edge of characters in the bitmap
FontResource			… Font	
GlyphResource			… Glyphs	

Layout Editor

The *Layout Editor* allows users to preview a single page as well as the whole project. ESD employs the EVE emulator to render an EVE display onto the layout editor. It provides users, the interface to view the result of the screen design and its logic.

The following file types are displayed in ESD layout editor:

- *Application logic file* - generally with ".main" extension
- *Page file* - generally with ".page" extension

Page File

Page stands for a single screen which will be rendered. One application consists of at least one page. Page may have its own life cycle and manages all the widgets on it. Only widgets can be dragged and dropped into pages at Page layout editor. However, all the nodes from a library can be dragged and dropped into the logic node editor.

"Active" Property

The *"Active" property* is a Boolean value to control the page, and is created or released while the application is updating the pages. By default, the "Active" property is True. The page is created and shown, unless "Switch" end is connected.

Switch Page

Users can define multiple pages within the project. *"Switch Page"* ESD layout should add to manage the pages. User can drag and drop the pages into this switch page from project browser. The picture below displays how the logic editor view looks like for the switch page logic.

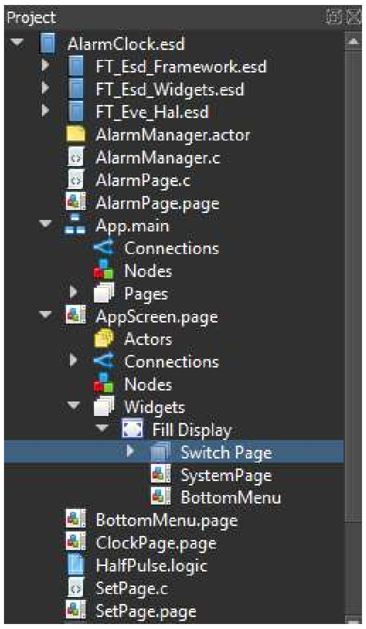


Figure 55 - Switch Page

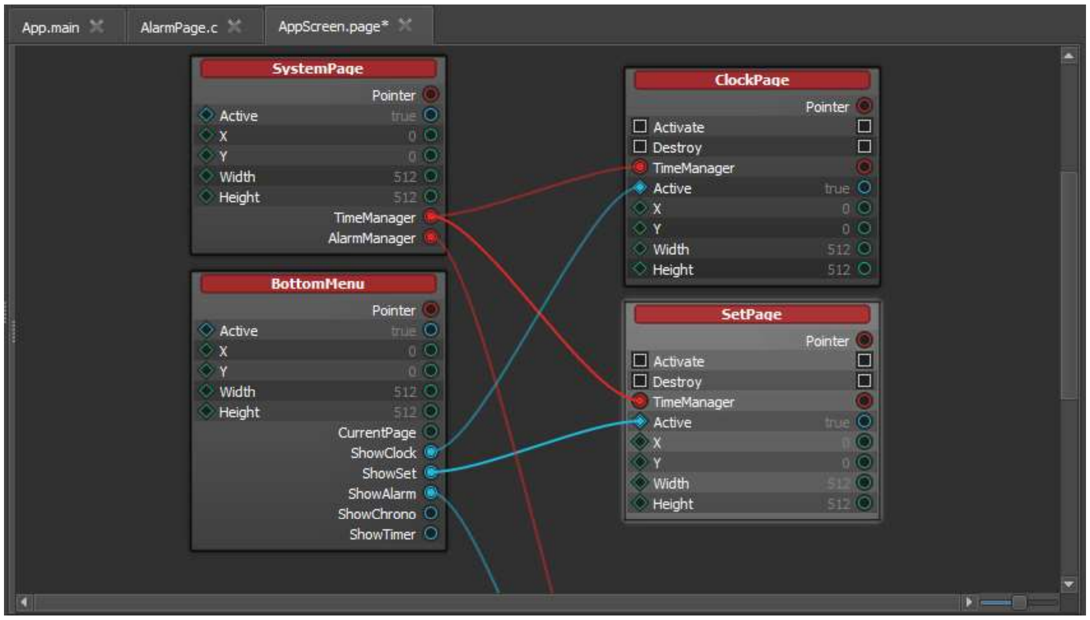
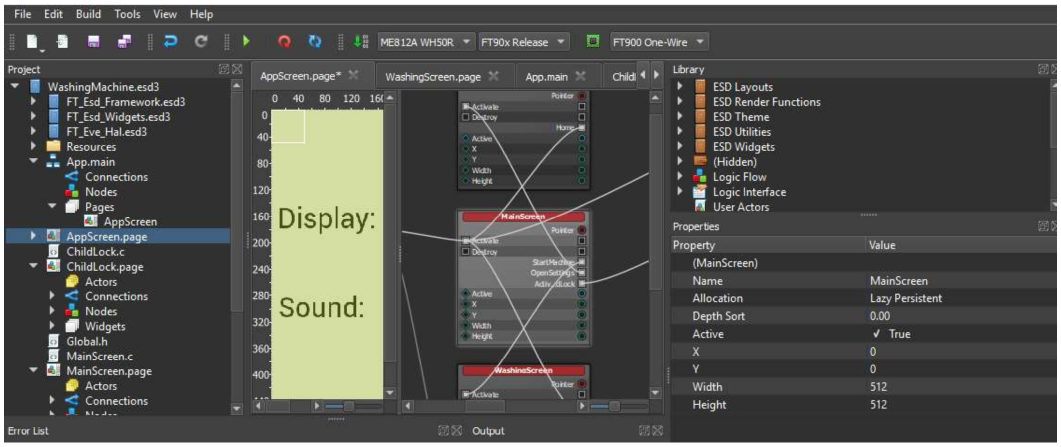


Figure 56 - Sample Switch Page UI

Page Persistence

To allow a page to remain in memory with its current state even when it is made inactive, set the allocation mode to "Lazy Persistent" or "Static" from the application logic in the page's Properties. This feature can help users to reserve the information of a page even when it is not visible.



Property	Value
(MainScreen)	
Name	MainScreen
Allocation	Lazy Persistent
Depth Sort	0.00
Active	✓ True
X	0
Y	0
Width	512
Height	512

Figure 57 - Properties Editor – Page Allocation

The page allocation modes supported in ESD are listed in the table given below:

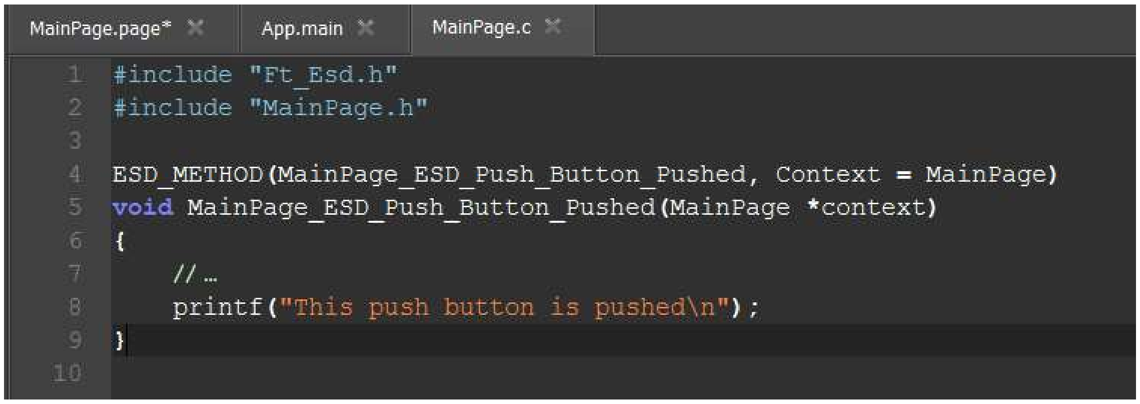
Allocation Mode	Description
Static	Page is always persistent in memory after application starts
Lazy	Page is created when it is in "active" state and destroyed when it is inactive
Lazy Persistent	Page is created when it is in "active" state but persistent in memory when it is "inactive" until application is closed.

User Defined Function for Page File

Users can write their own code to handle the "Pushed" signal of the "ESD Push Button" widget. Users are required to select the "ESD Push Button" widget in the logic node editor and double click it. A function will be generated to replace the user defined function.

```
ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context = MainPage)
void MainPage_ESD_Push_Button_Pushed(MainPage *context)
{
        // Users can write their code here
}
```

The code can be located at ***$(PageFileName).c*** file.



```
#include "Ft_Esd.h"
#include "MainPage.h"

ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context = MainPage)
void MainPage_ESD_Push_Button_Pushed(MainPage *context)
{
    //…
    printf("This push button is pushed\n");
}
```

Figure 58 - Sample Source File

Zoom In & Out

The visual area of the screen layout editor can be zoomed when the current file is a page file, i.e., *".page"* file. The mouse wheel button provides *zoom-in* and *zoom-out* functionality, allowing users to view more details about the screen design. Refer to the sample picture given below:

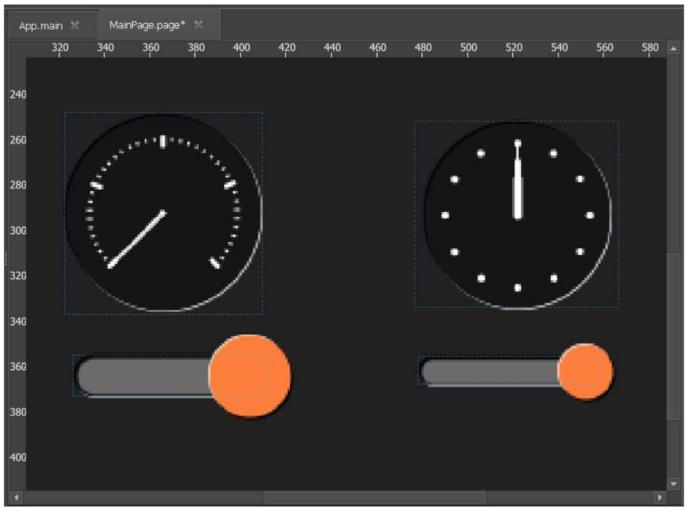
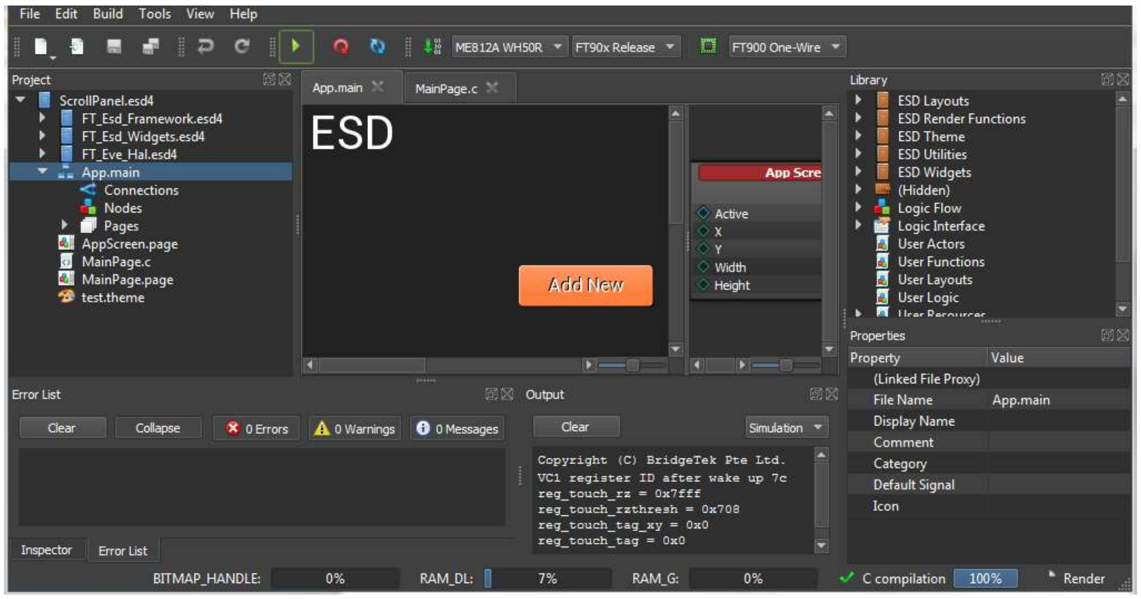


Figure 59 - Zoom In & Out

Main File

To view the application logic file (*.main) file, double click the file within the project explorer. The screen layout editor will show the first page defined in the logic node editor. To view how the application logic is defined, click the **"Simulation"** button and check it in the screen layout editor.



Property	Value
(Linked File Proxy)	
File Name	App.main
Display Name	
Comment	
Category	
Default Signal	
Icon	

```
Copyright (C) BridgeTek Pte Ltd.
VC1 register ID after wake up 7c
reg_touch_rz = 0x7fff
reg_touch_rzthresh = 0x708
reg_touch_tag_xy = 0x0
reg_touch_tag = 0x0
```

Figure 60 - Main File

Logic File

The logic file contains certain logic that is used to encapsulate the user defined logic. It works similar to a C function. A logic file does not render any user interface.

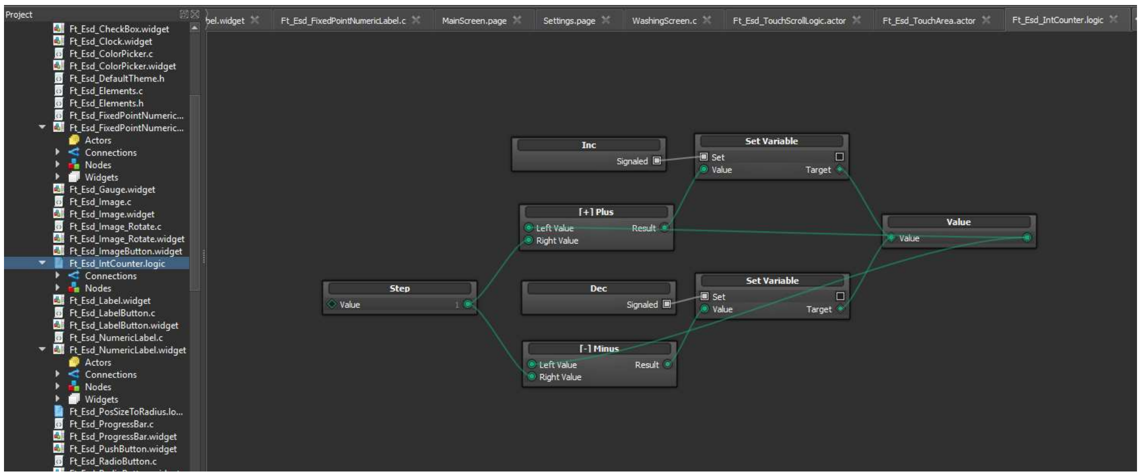
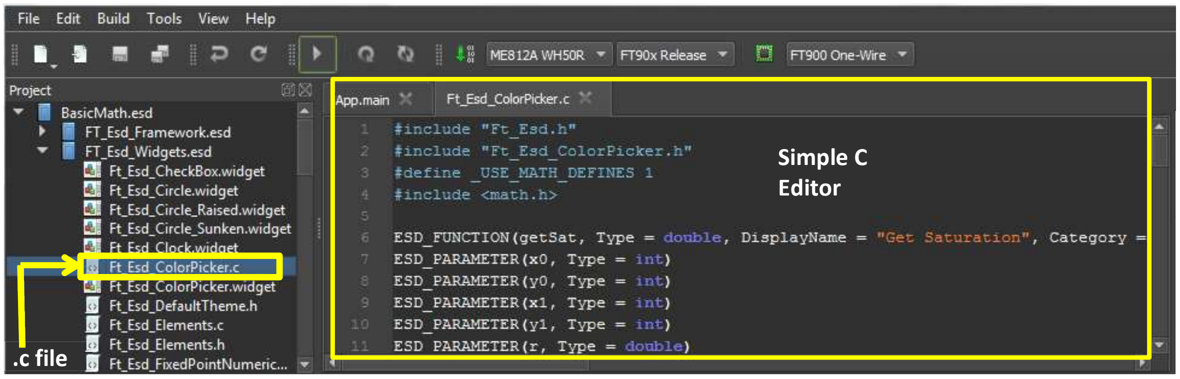


Figure 61 - Sample Logic File

C File

The screen layout editor provides a simple C editor when users double click a ".c" file in the project browser. Note that it is not a full-fledged C editor and many features may be implemented in the future.



```
#include "Ft_Esd.h"
#include "Ft_Esd_ColorPicker.h"
#define _USE_MATH_DEFINES 1
#include <math.h>

ESD_FUNCTION(getSat, Type = double, DisplayName = "Get Saturation", Category =
ESD_PARAMETER(x0, Type = int)
ESD_PARAMETER(y0, Type = int)
ESD_PARAMETER(x1, Type = int)
ESD_PARAMETER(y1, Type = int)
ESD_PARAMETER(r, Type = double)
```

Figure 62 - C File / C Editor

To effect the changes made, ensure that all the changes are saved to the project by clicking **"File → Save All"** from the menu or from the toolbar.

Logic Note Editor

Using the *Logic Node Editor* (LNE), users compose screen logic by connecting logic nodes. The Logic Note Editor is designed to open the following file formats:

- Application logic file , *.main file
- Screen logic file, *.page file
- Widget file, *.widget
- Actor file, *.actor
- User defined logic file, *.logic file

Basic Logic Node

ESD has some built-in predefined *basic logic nodes*. These basic logic nodes provide basic control flow and logic interfaces as well as basic functions. A typical basic logic node is represented in the logic node editor as below, the name of which is shown in white.

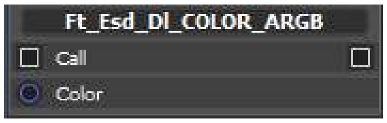
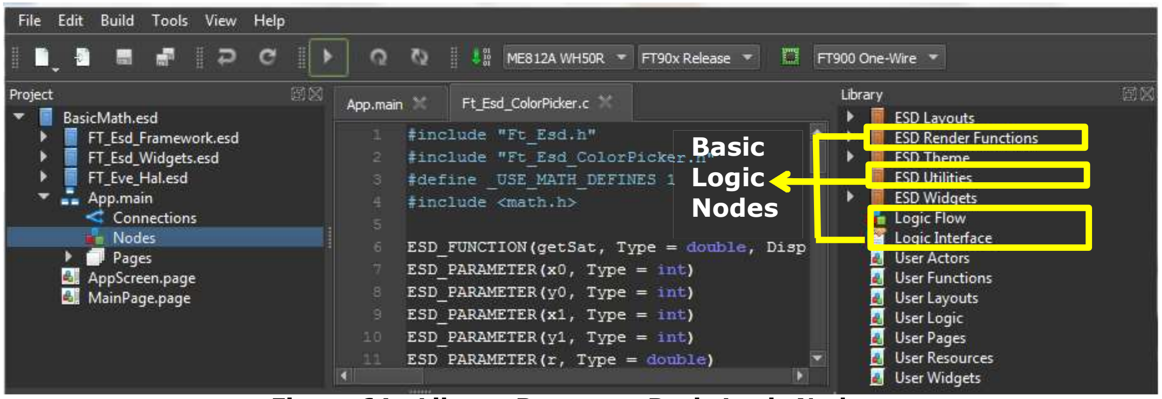


Figure 63 - Logic Node Editor - Basic Logic Node

Within the library browser, the basic logic nodes are located at *"Logic Flow"*, *"Logic Interface"*, *"ESD Utilities"* and *"EVE Render Functions"*.



```
#include "Ft_Esd.h"
#include "Ft_Esd_ColorPicker.h"
#define _USE_MATH_DEFINES 1
#include <math.h>

ESD_FUNCTION(getSat, Type = double, Disp
ESD_PARAMETER(x0, Type = int)
ESD_PARAMETER(y0, Type = int)
ESD_PARAMETER(x1, Type = int)
ESD_PARAMETER(y1, Type = int)
ESD_PARAMETER(r, Type = double)
```

Figure 64 - Library Browser - Basic Logic Nodes

Composite Logic Node

A *Composite Logic Node* is made up by connecting multiple basic logic nodes. A typical composite logic node is defined in a standalone document in XML format and is shown in the logic node. Refer to the picture given below -

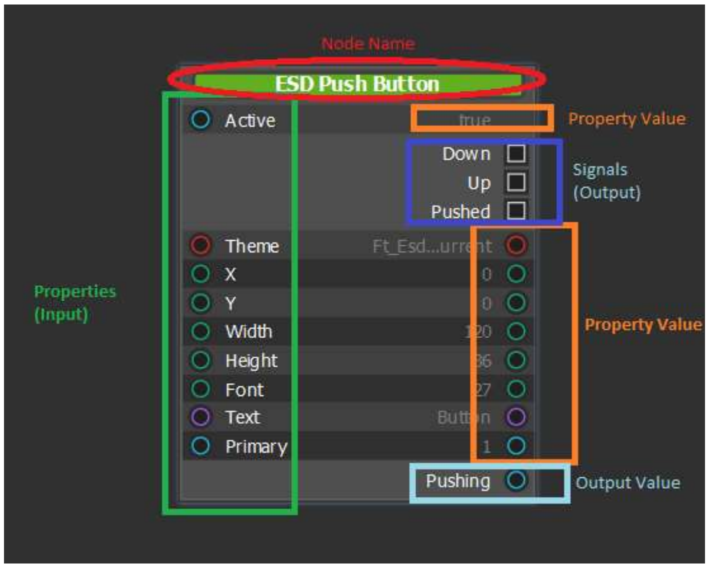


Figure 65 - Composite Logic Node

The following are the different types of composite logic nodes defined in ESD 4.14:

- *Page Node*
- *Widget Node*
- *Layout Widget*
- *Actor Node*
- *Logic Object*

Page Node

When users create a single page, it is represented by a logic node under the "User Page" category in the library browser. Users can connect one page node to another within the logic node editor. It has its own property and defined output.

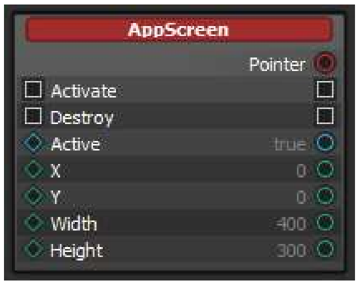


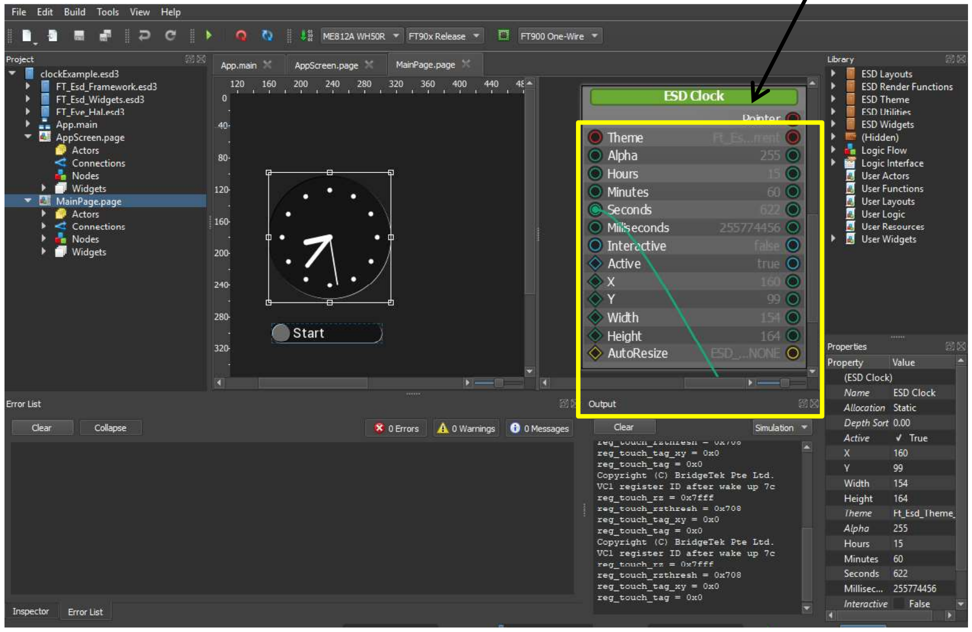
Figure 66 - Composite Logic Node - Page Node

Widget Node

The *Widget Node* is one type of logic node which has visual appearance and is able to handle user input. It has the built-in *Start*, *Update*, *Idle*, *Render* and *End slots*.

For more information on built-in slots, please refer to Built-in Slot.

Within the logic node editor, the widget name is highlighted with the green colour title as shown in the sample picture below -



```
reg_touch_tag_xy = 0x0
reg_touch_tag = 0x0
Copyright (C) BridgeTek Pte Ltd.
VC1 register ID after wake up 7c
reg_touch_rz = 0x7fff
reg_touch_rzthresh = 0x708
reg_touch_tag_xy = 0x0
reg_touch_tag = 0x0
Copyright (C) BridgeTek Pte Ltd.
VC1 register ID after wake up 7c
reg_touch_rz = 0x7fff
reg_touch_rzthresh = 0x708
reg_touch_tag_xy = 0x0
reg_touch_tag = 0x0
```

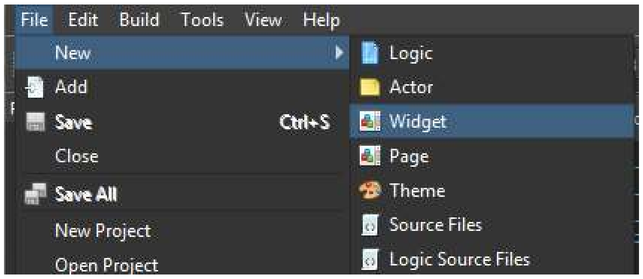
Property	Value
(ESD Clock)	
Name	ESD Clock
Allocation	Static
Depth Sort	0.00
Active	✓ True
X	160
Y	99
Width	154
Height	164
Theme	Ft_Esd_Theme_
Alpha	255
Hours	15
Minutes	60
Seconds	622
Millisec...	255774456
Interactive	False

Figure 67 - Logic Node Editor – Widget Node

The highlighted sections provide information on adding user widgets; the position and size properties; rendering the widget and widget theme.

Adding User Widgets

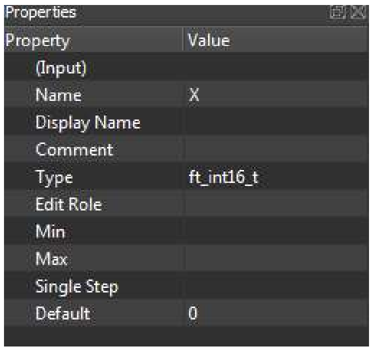
A *User Widget*, also called as *custom widget* is a reusable user interface element created by a user. To create a widget, from the menu, click and select **"File → New → Widget"**.



In order to make use of these slots, users need to drag the slots from the library browser and drop them into the logic node editor. By renaming it to the slot defined above, users can connect widget control flow to application control flow.

Position & Size Properties

As a user interface element, widgets have at least position and size properties so that users can control them using a visible handle. Position property is defined by an "Input" node with "ft_int16_t" type named as "X" or "Y". Size property is defined by an "Input" node with "ft_int16_t" type named as "Width" or "Height". Figure 68 provides a sample of the "X" property.



Properties

Property	Value
(Input)	
Name	X
Display Name	
Comment	
Type	ft_int16_t
Edit Role	
Min	
Max	
Single Step	
Default	0

Figure 68 - User Widget - Position & Size Properties

Rendering a widget

Rendering of a widget may be implemented fully in the logic editor, or directly in the code using the logic editor to glue together the code with the generated structures. To create the render function, simply add a *Slot* from the Logic Interface category of the library browser, name it “Render” and double click to create the user method for handling the rendering. The widget is expected to restore any graphics state it changes, with the exception of the state values specified below.

Preferably use the Ft_Esd_Dl_ utility functions for display list drawing when available. Several state parameters which may be expected to be changed often to the same value implement de-duplication here to shorten the display list. The de-duplicated values are: *Palette source, Color RGB, Color A, Handle, Cell,* and *Vertex Format*. These graphic state values are not required to be restored to their original values when the Ft_Esd_Dl_ functions are used. These values should be expected to be any undefined value at the start of the Rendering function. Access the value of the X input property as given below:

```
int x = context->X(context->Owner);
```

This calls the `get()` function assigned by the parent to the X property, passing the owner’s context to the function to handle it there.

Theme

Widgets should provide property input to override the default application theme. An input property with type *Ft_Esd_Theme* * should be added by dragging one input node from the “Logic Interface” category of the library browser. Colours from the theme may be accessed from the logic editor using the utility functions under the ESD Theme category, or from the user code as given below:

```
Ft_Esd_Theme *theme = context->Theme(context->Owner);
if (theme == NULL) theme = Ft_Esd_Theme_GetCurrent();
ft_rgb32_t primaryColor = theme->PrimaryColor;
```

Touch Input

In order to handle the *touch input*, add a Touch Tag node from *“ESD Utilities”* category of the library browser to the logic editor, and either handle the signals or access its state through the Touch Tag interface. Users need to make use of the “Tag” value provided by the “Touch Tag” node during the Render function, and restore the “Tag” to value 0 after the relevant portion of the rendering.

To handle any of the signals from Touch Tag in user code, simply create an ESD_METHOD style function in the same format as the Render user code function, and it will be available from the User Functions category in the library to drag into the logic editor and connect to any of the signals from the Touch Tag node.

Layout Type Widget

Layout widget is a layout container for widgets. Unlike widget nodes which have visual appearance, a layout widget manages the layout specification for its child widgets. Layout widgets are introduced since ESD 4.0. Figure 69 & Figure 70 displays the various layout widgets supported in ESD.

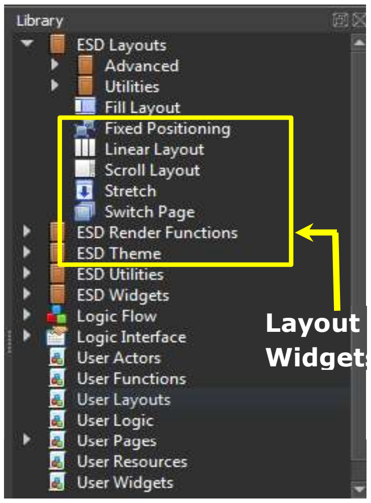


Figure 69 - Layout Widgets

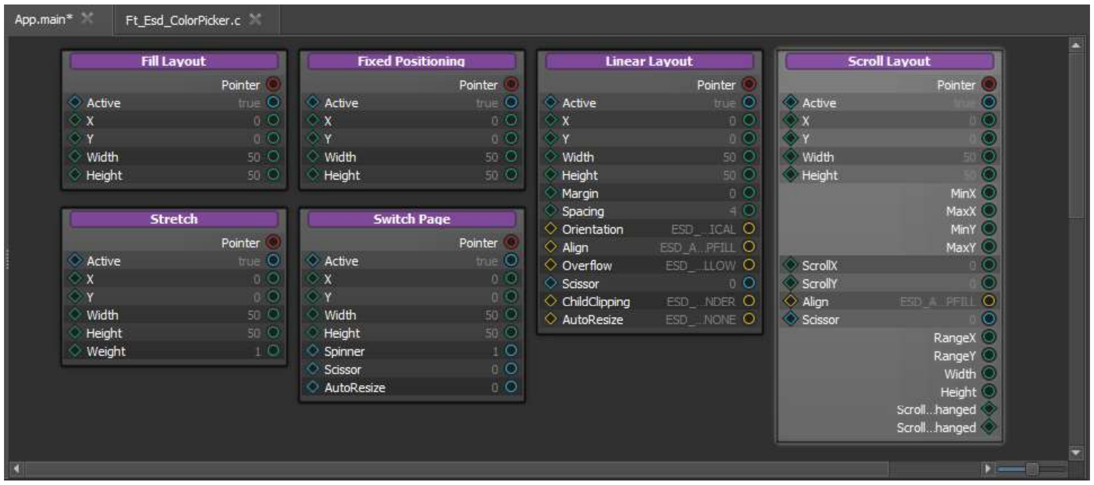


Figure 70 - Layout Widgets

Actor Node

Actor type is an extended type of logic node which functions similar to the widget node, but without the *Render* slot. Therefore, it has no visual appearance. When users create a new actor node, it does not show in the layout editor. It has the built-in *Start, Update,* *Idle* and *End* slots which are not available in a regular logic node. For more information about built-in slots, please refer to Built-in Slot.

An Actor node is useful for implementing both continuous and background running behaviours, such as animations, timers, and input data polling. The actor node name is highlighted with the yellow colour within the logic node editor.

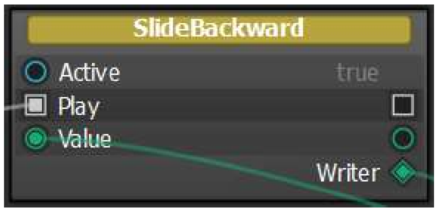
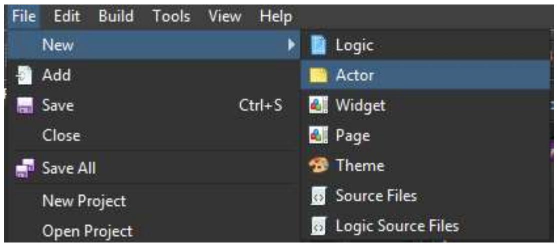


Figure 71 - Actor Node

To create an actor, from the menu, click and select **"File → New → Actor"**.



Refer to the example project **"ActorAnimator.esd"** under the **"EVE Screen Designer → Examples → Basic → ActorAnimators"** folder for adding and using actor node.

Logic Object

A *Logic Object* is a basic logic block designed to express certain logic which has NO built-in slot. Hence, it is not possible to invoke it through the built-in render slot or update slot. However, it may have its own private or public property and output, signal and slot, depending on the users' implementation.

A logic object is generally used to simplify complex logic by splitting it into multiple smaller and simpler logic objects. It is defined and saved in a "*.logic" file.

Within the logic node editor, the logic object name is highlighted with the grey colour.

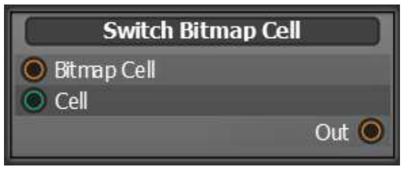


Figure 72 - Logic Object

To add a logic object, from the menu, click and select **"File → New → Logic"**.

Connections

Connections are lines between two nodes. The connection line's colour is determined by the ends. The following table provides the different types of ends and their description.

Connection Line End Type	Description
Event slots and Signals ■	Event slots and signals are implemented as functions which only take a pointer to the node context as a parameter. The node context contains the state of the node as well as a pointer to the parent node to signal back.
Data property bindings ◎ / ◎ / ◎	Data property input and output bindings are implemented as get-functions which only take a pointer to the node context as a parameter and return a value. This means that these connections are completely evaluated every time the property is read, and consequently the property always reads as the latest value. A buffer variable can be used to cache values if necessary. The different colours denote the different data types.
Variable write ◇	The write interface is translated into a function pointer during the code generation. Variables are plain variables, which can be written using the Set Variable node. They may also be set through the Writer interface which is implemented using a pointer to a set-function.

Table 9 - Connection End Types

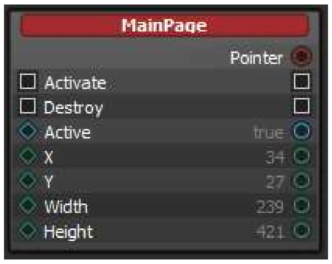
In general, all of the input properties that can be connected can also be specified from the property editor when they are constant values; this is sometimes referred to as the "Default" value. In addition, it is possible to specify the Minimum and Maximum values for the input bindings and variables.

Rendering Order

In the application logic file "App.main", the pages added on the logic node editor are to be represented with the symbols shown in the sample picture.

When there are more than two pages, the order in which the pages were created determines the rendering order.

The following implicit rule of the logic node editor is worthy to be noted, since this rule applies to all the logic nodes.


MainPage
Pointer
Activate
Destroy
Active true
X 34
Y 27
Width 239
Height 421

"Depth Sort" is used to determine the rendering order of the logic nodes. The greater the value of "Depth Sort", the later the logic node is rendered and higher the layer in which it is shown. When logic nodes have the same "Depth Sort" value, the logic node's "X"/ "Y" coordinate (within layout widget) determines the rendering order. The page node with the lowest value of the "X"/ "Y" coordinate is rendered first (as lower depth value).

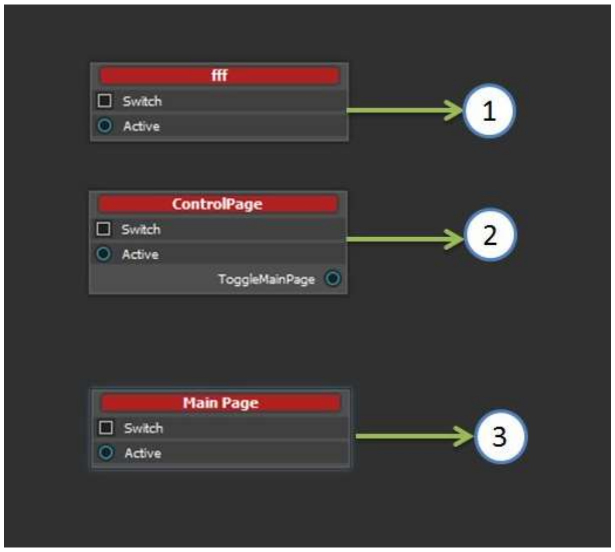


Figure 73 - Rendering Order Example

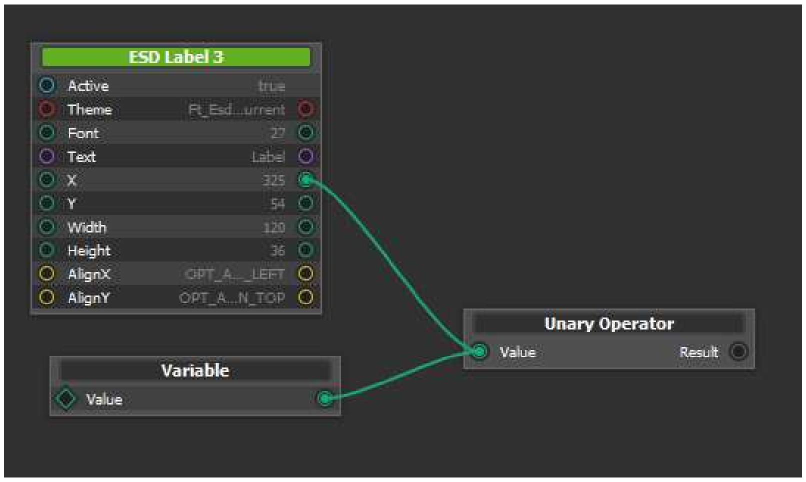
Logic Note Editor – User Interaction

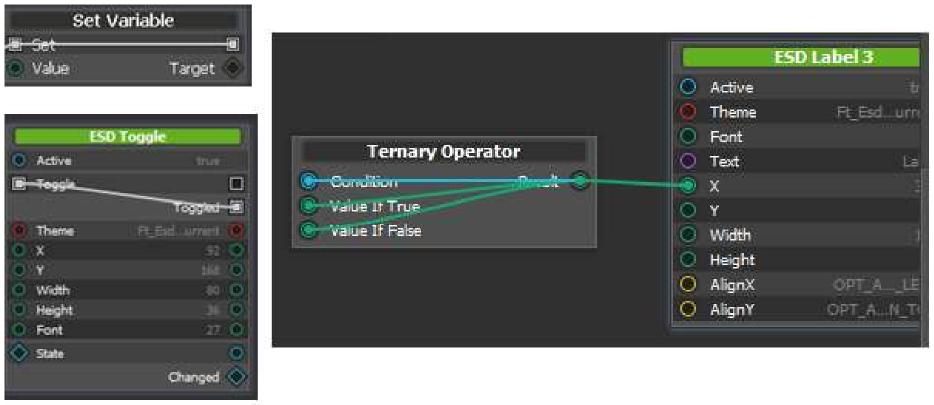
The Logic Node Editor enables *Zoom-in* and *Zoom-out* functionality. Users can make use of the mouse wheel button to zoom in and zoom out.

"Panning" the logic node editor screen helps reveals the parts of the logic node editor which could not have been fit in a single screen. Panning can be done by 2 ways:

(i) Press *Alt Key* + *Left mouse button* and move the mouse
(ii) Press the *middle mouse* button and move the mouse.

User can select multiple node items in the logic node editor by pressing Control key and selecting the desired objects by left mouse button click. Single/multiple objects can thus be moved if needed by moving the mouse while pressing the Left mouse button.

✗ **DON'T S**	Avoid two connections connecting to the same port. Refer to the sample picture given below. 

	Avoid endless dead loops caused by connecting the input and output of the same object. It will cause an unexpected error. Refer to the sample pictures given below.  Set Variable Set Value Target ESD Toggle Active Toggle Toggled Theme X Y Width Height Font State Changed Ternary Operator Condition Result Value If True Value If False ESD Label 3 Active Theme Font Text X Y Width Height AlignX AlignY

To effect the changes made to the logic note editor, ensure that all the changes are saved to the project, by clicking **"File → Save All"** from the menu or toolbar.

In some cases, click the **"Recompile"** button to recompile the source code generated by ESD.

Library Browser

The *Library Browser* consists of the basic components which are predefined and built in as part of ESD 4.14, as well as user defined components and/or resources. The components in the library include *Theme*, *Primitive, Widgets*, *Logic Flow* and *Logic Interface*. Users can select the appropriate components and drag them into the logic node editor or layout editor.

The library view depends on the currently selected node. The library browser will only show the contents which applies to the currently opened node.

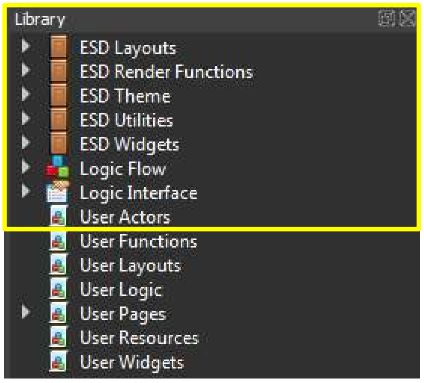


Figure 74 - Library Browser

In the Library Browser, items can be filtered by entering the search text in the textbox. Regular expression is supported. The matching text is highlighted as shown in the sample picture given below -

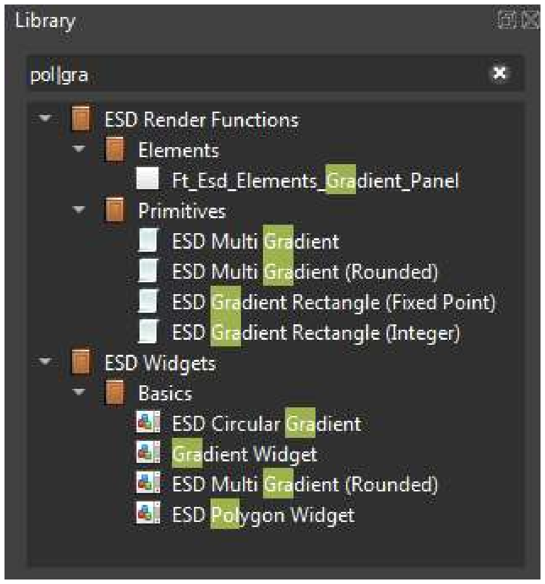
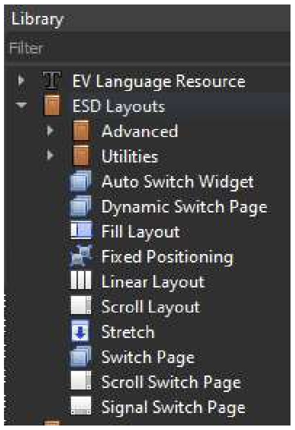

Library
pol|gra
- ESD Render Functions
  - Elements
    - Ft_Esd_Elements_Gradient_Panel
  - Primitives
    - ESD Multi Gradient
    - ESD Multi Gradient (Rounded)
    - ESD Gradient Rectangle (Fixed Point)
    - ESD Gradient Rectangle (Integer)
- ESD Widgets
  - Basics
    - ESD Circular Gradient
    - Gradient Widget
    - ESD Multi Gradient (Rounded)
    - ESD Polygon Widget

Figure 75 - Library Filter

The forthcoming sections provide information about the different components of the library browser.

ESD Layouts

ESD Layouts contains the collection of ESD layout widgets and layout utilities functions.


Library
Filter
- EV Language Resource
- ESD Layouts
  - Advanced
  - Utilities
  - Auto Switch Widget
  - Dynamic Switch Page
  - Fill Layout
  - Fixed Positioning
  - Linear Layout
  - Scroll Layout
  - Stretch
  - Switch Page
  - Scroll Switch Page
  - Signal Switch Page

Layout	Description
Dynamic Switch Page	This Layout is used to achieve a fade-in/fade-out animation from one page to another page
Fill Layout	This layout is used to fill up the whole widget display region
Fixed Positioning	This layout is for fixed coordinate layouts
Linear Layout	This layout supports both horizontal and vertical layouts
Scroll Layout	Specific layout for scrollable panel
Stretch	This layout supports both horizontal and vertical layouts
Switch Page	To manage page switching logic

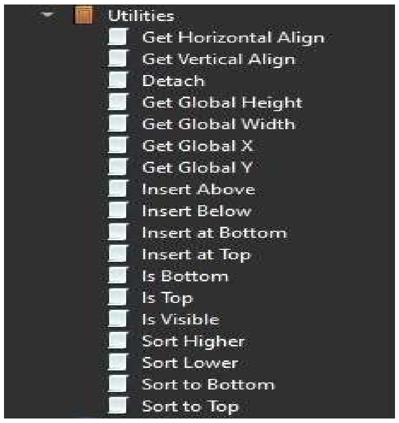
Scroll Switch Page	This layout supports scroll between pages according to the swipe direction.
Signal Switch Page	This layout support switch between pages via signals, can be set as vertical switch or horizontal switch.

There are also layout utilities and advanced layout features provided in ESD Layouts collection. These sections are for advanced user only.

Layout utilities are a set of helping functions to ease the development of layout. Users can build their own layout widget by making use of them. There are several items defined for sorting widget's purpose.

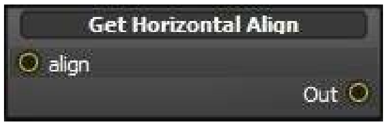
Utilities

ESD provides various Layout utility functions while designing layouts.



Get Horizontal Align

The *Get Horizontal Align* utility is used to align the widget horizontally.

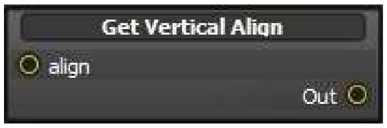


The function can be called as shown below:

```
ESD_FUNCTION(ESD_ALIGN_HORIZONTAL, Type = Esd_AlignHorizontal, DisplayName
= "Get Horizontal Align", Category = EsdLayoutUtilities, Macro)

ESD_PARAMETER(align, Type = Esd_Align)

#define ESD_ALIGN_HORIZONTAL(align) (align & 3)
```

Get Vertical Align

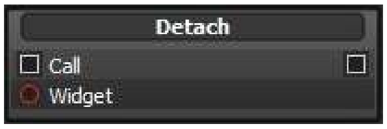
The *Get Vertical Align* utility is used to align the widget vertically.



The function can be called like below:

```
ESD_FUNCTION(ESD_ALIGN_VERTICAL, Type = Esd_AlignVertical, DisplayName =
"Get Vertical Align", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(align, Type = Esd_Align)
#define ESD_ALIGN_VERTICAL(align) (align & 12)
```

Detach

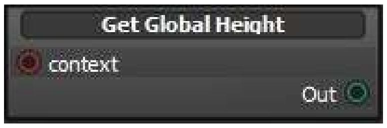


The *Detach* utility is used to detach the widget from the current parent layout

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_Detach,  DisplayName  =  "Detach",  Category  =
EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_Detach(Ft_Esd_Widget *context);
```

Get Global Height

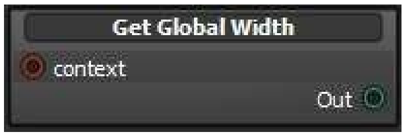


The *Get Global Height* utility is used to get the global X co-ordinate of widget.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_GetGlobalHeight, Type = ft_int16_t, DisplayName
= "Get Global Height", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
#define       Ft_Esd_Widget_GetGlobalHeight(context)        (((Ft_Esd_Widget
*)context)->GlobalHeight)
```

Get Global Width

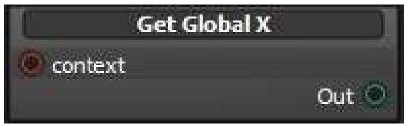


The *Get Global Width* utility is used to get the width of the widget.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_GetGlobalWidth, Type = ft_int16_t, DisplayName =
"Get Global Width", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
#define Ft_Esd_Widget_GetGlobalWidth(context) (((Ft_Esd_Widget
*)context)->GlobalWidth)
```

Get Global X

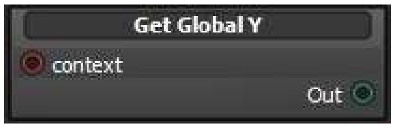


The *Get Global X* utility is used to get the global X co-ordinate of the widget.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_GetGlobalX, Type = ft_int16_t, DisplayName =
"Get Global X", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
#define Ft_Esd_Widget_GetGlobalX(context) (((Ft_Esd_Widget
*)context)->GlobalX)
```

Get Global Y

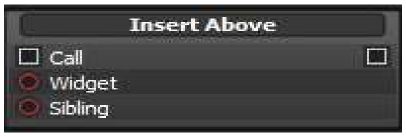


The *Get Global Y* utility is used to get the global Y co-ordinate of the widget.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_GetGlobalY, Type = ft_int16_t, DisplayName =
"Get Global Y", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
#define Ft_Esd_Widget_GetGlobalY(context) (((Ft_Esd_Widget
*)context)->GlobalY)
```

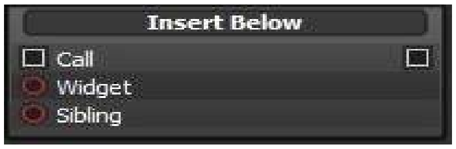
Insert Above



The *Insert at Above* utility is used to insert a widget above the specified sibling widgets in the sibling widgets parent layout.

```
ESD_FUNCTION(Ft_Esd_Widget_InsertAbove, DisplayName = "Insert Above",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ESD_PARAMETER(sibling, DisplayName = "Sibling", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_InsertAbove(Ft_Esd_Widget *context, Ft_Esd_Widget
*sibling);
```

Insert Below

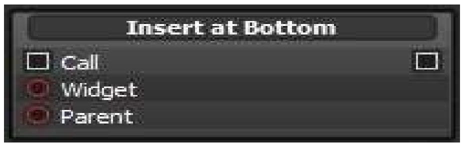


The *Insert at Below* utility is used to insert a widget below the specified sibling widgets in the sibling widgets parent layout.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_InsertBelow, DisplayName = "Insert Below",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ESD_PARAMETER(sibling, DisplayName = "Sibling", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_InsertBelow(Ft_Esd_Widget *context, Ft_Esd_Widget
*sibling);
```

Insert at Bottom



The *Insert at Bottom* utility is used to insert a widget in the bottom position of the specified parent layout.

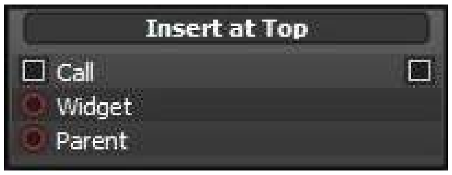
The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_InsertBottom, DisplayName = "Insert at Bottom",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ESD_PARAMETER(parent, DisplayName = "Parent", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_InsertBottom(Ft_Esd_Widget *context, Ft_Esd_Widget
*parent);
```

Insert at Top

The *Insert at Top* utility is used to insert a widget in the top position of the specified parent layout.

The function can be called like below:

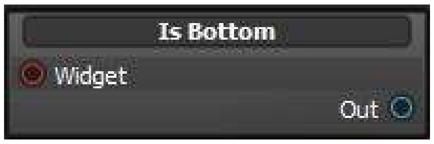


```
ESD_FUNCTION(Ft_Esd_Widget_InsertTop, DisplayName = "Insert at Top",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ESD_PARAMETER(parent, DisplayName = "Parent", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_InsertTop(Ft_Esd_Widget *context, Ft_Esd_Widget
*parent);
```

Is Bottom

The *Is Bottom* utility returns FT_TRUE if the widget is the bottom widget within the current parent layout.

The function can be called like below:

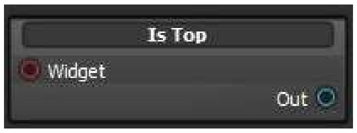


```
ESD_FUNCTION(Ft_Esd_Widget_IsBottom, Type = ft_bool_t, DisplayName = "Is
Bottom", Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ft_bool_t Ft_Esd_Widget_IsBottom(Ft_Esd_Widget *context);
```

Is Top

This *Is Top* utility returns FT_TRUE if the widget is the topmost widget within the current parent layout

The function can be called like below:

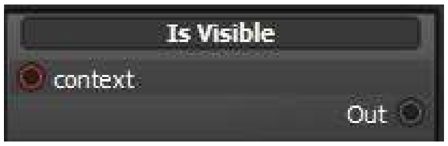


```
ESD_FUNCTION(Ft_Esd_Widget_IsTop, Type = ft_bool_t, DisplayName = "Is Top",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ft_bool_t Ft_Esd_Widget_IsTop(Ft_Esd_Widget *context);
```

Is Visible

The *Is Visible* utility is used to check if the widget is within the current screen scissor area.

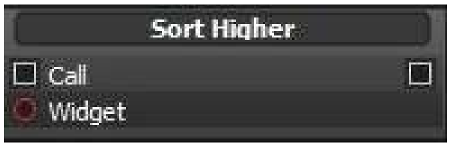
The function can be called like below:



```
ESD_FUNCTION(Ft_Esd_Widget_IsVisible, Type = ft_bool + t, DisplayName = "Is
Visible", Category = EsdLayoutUtilities)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
ft_bool_t Ft_Esd_Widget_IsVisible(Ft_Esd_Widget *context);
```

Sort Higher

The *Sort Higher* utility is used to sort the widget above its current sibling within the current parent layout. The function can be called like below:

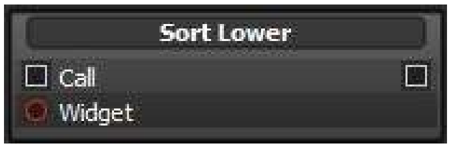


```
ESD_FUNCTION(Ft_Esd_Widget_SortHigher,
DisplayName = "Sort Higher", Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_SortHigher(Ft_Esd_Widget *context);
```

Sort Lower

The *Sort Lower* utility is used to sort the widget below its current sibling within the current parent layout.

The function can be called like below:

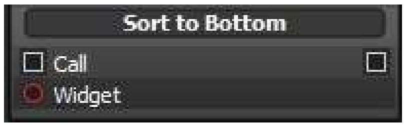


```
ESD_FUNCTION(Ft_Esd_Widget_SortLower, DisplayName = "Sort Lower", Category
= EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_SortLower(Ft_Esd_Widget *context);
```

Sort to Bottom

The *Sort to Bottom* utility is used to sort the widget to the bottom within the current parent layout.

The function can be called like below:

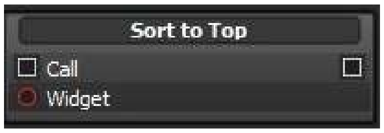


```
ESD_FUNCTION(Ft_Esd_Widget_SortToBottom, DisplayName = "Sort to Bottom",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_SortToBottom(Ft_Esd_Widget *context);
```

Sort to Top

The *Sort to Top* utility is used to sort the widget to the top within the current parent layout.

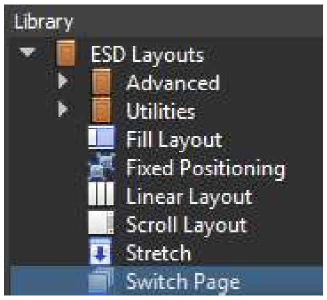
The function can be called like below:

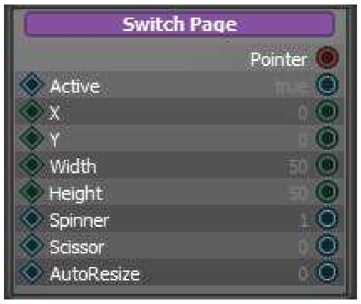


```
ESD_FUNCTION(Ft_Esd_Widget_SortToTop, DisplayName = "Sort to Top", Category
= EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_SortToTop(Ft_Esd_Widget *context);
```

Switch Page

The *Switch Page* is a special layout which focuses on page/widget transitions. By default, there will be only one Switch Page layout in "AppScreen.Page" in every ESD project. However, this is optional and the users can remove or change the switch page layout in the project. Refer to the example project **"EVE Screen Designer → Examples → Basic → MenuPage → MenuPage.esd → AppScreen.page"** for a switch page demo.

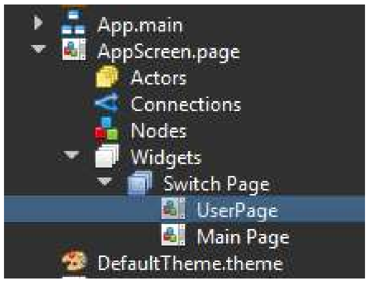




Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Spinner	Set true to show spinner/loading icon when switching page. This is useful user indication when a page has heavy resource.
Scissor	Set true, to trim off any content is drawn outside the layout region.
Auto Resize	Set true, to enable auto resize for this layout.

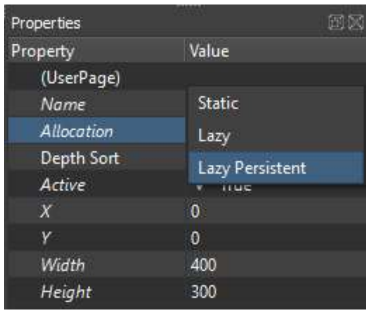
Table 10 – Switch Page Layout Properties

Switch Page Implementation



1. Set page/widget into Switch Page

Unlike widgets, Switch Page cannot be dragged on Screen Layout Editor or Logic Editor. In order to put a page under the control of the switch page layout, the user has to use Project Browser instead, drag the widget into the switch page layout.



Property	Value
(UserPage)	
Name	Static
Allocation	Lazy
Depth Sort	Lazy Persistent
Active	True
X	0
Y	0
Width	400
Height	300

2. Page Resource

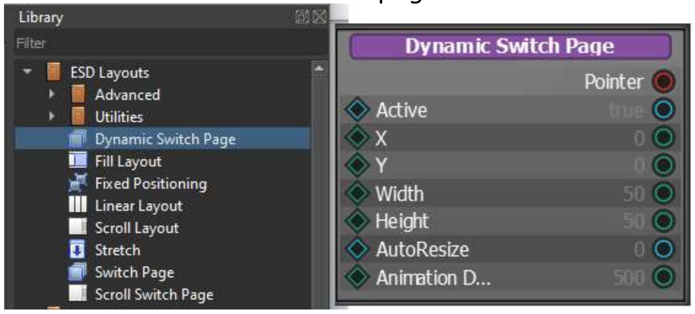
Since ESD 4.0, a page resource allocation can be:

- Static – always load at the start
- Lazy – allocate when it is active, and free it when it is inactive.
- Lazy persistent – allocate when it is first time active, and it will remain in system until reset.

When a new page is activated, its parent object, the Switch Page will also mark the current page as inactive and free it from memory when it has “Lazy” mode of allocation.

Dynamic Switch Page

The *Dynamic Switch Page* is a special layout which focuses on page/widget transitions. The primary purpose of this widget is to achieve a fade-in/fade-out animation while transiting from one page to another page. Like Switch Page widget, only one of the children of Switch Page would be visible at any point of time. But unlike Switch Page widget which abruptly changes the visibility between the transiting pages, Dynamic Switch Page widget will fade-in and fade-out the new to-be-visible page and old not-to-visible page.



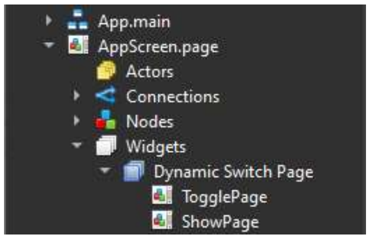
Dynamic Switch Page

Pointer	
Active	true
X	0
Y	0
Width	50
Height	50
AutoResize	0
Animation D...	500

Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Auto Resize	Set true, to enable auto resize for this layout. -
Animation Duration	Sets the time within which the animation needs to be finished.

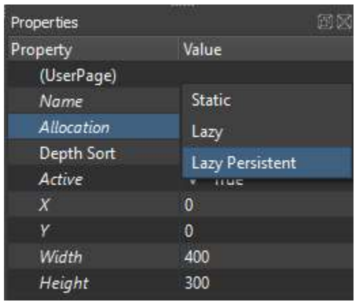
Table 11 - Dynamic Switch Page Layout Properties

Dynamic Switch Page Implementation



1. Set page/widget into Dynamic Switch Page

Unlike widgets, Dynamic Switch Page cannot be dragged on Screen Layout Editor or Logic Editor. In order to put a page under the control of the switch page layout, the user has to use Project Browser instead, drag the widget into the Dynamic switch page layout.



Property	Value
(UserPage)	
Name	
Allocation	
Depth Sort	
Active	true
X	0
Y	0
Width	400
Height	300

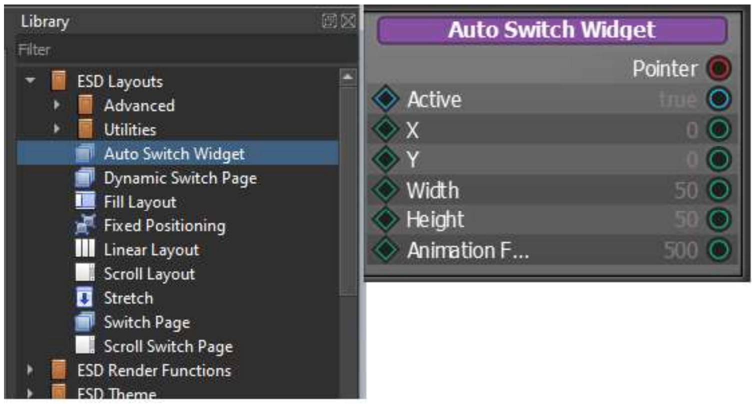
2. Page Resource

Since ESD 4.0, a page resource allocation can be:

- Static – always load at the start
- Lazy – allocate when it is active, and free it when it is inactive.
- Lazy persistent – allocate when it is first time active, and it will remain in system until reset.

When a new page is activated, its parent object, the Dynamic Switch Page will also mark the current page as inactive and free it from memory when it has "Lazy" mode of allocation.

Auto Switch Widget

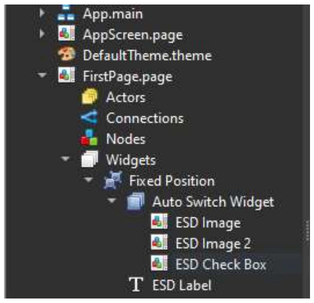


The *Auto Switch Widget* is a layout which focuses on rapid transitions between its child widgets. The primary purpose of this widget is to rapidly activate its child widget one by one in the order of its depth. The rate of change is set by the user. This can be used to generate a gif like animation effect. A sub. But unlike Switch Page and Dynamic Switch Page, the transition between child widget is automatic and not triggered. Use case of the widget can be found in Examples/Basic/AutoSwitchLayout.

Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinates of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Animation Frequency	Sets the rate of change of child widget activations

Table 12 - Auto Switch Page Layout Properties

Auto Switch Page Implementation



1. Set widgets into Dynamic Switch Page

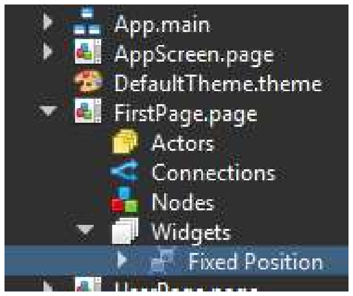
Auto Switch widget can be dragged on Logic Editor. In order to put a widget under the control of the switch page layout, the user has to use Project Browser instead, drag the widget into the Auto Switch Widget layout.

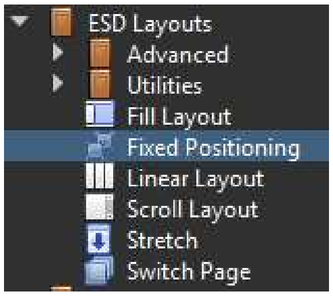
2. Widget Rendering order

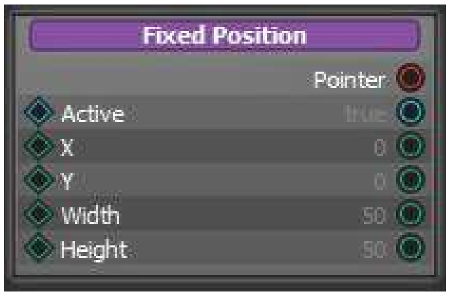
All child widgets are activated one by one and rendered in a sequence according to their depth orders. Thus, to rearrange the order of activation, user needs to update the depth order of the children widgets.

Fixed Positioning

The *Fixed Positioning* is a layout widget which allows its child widgets to be fixed at design time. By default, every page starts with one fixed positioning layout, such that all widgets under the page are placed at design time. Fixed Positioning layout does not update its child widget's dimensions on runtime. Users should get what they see on design time, on runtime as well.





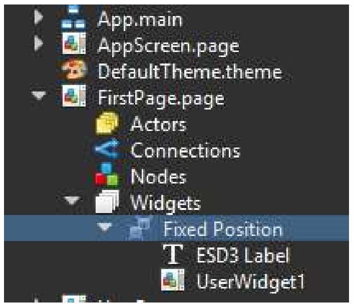


Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height

Table 13 - Fixed Positioning Layout Properties

Fixed Positioning Implementation

1. Set page/widget into Fixed Positioning Layout

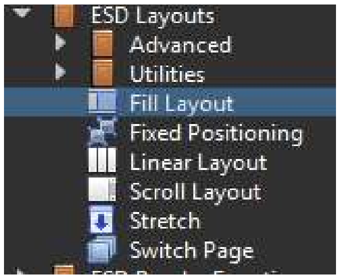


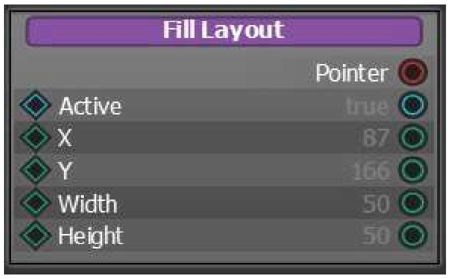
Drag and drop a new widget into the Screen Layout Editor will be the highest depth sort widget in the layout by default. However, the depth sort values are configurable by the users. If a widget is not in fixed positioning layout, the Screen layout editor will not able to change it is position as it desired. Always check on project browser as picture on the right.

2. Change Child Widgets' depth sort

Changing the depth sort value of child widgets will affect their rendering sequence. This functionality is same as in ESD 3.0.

Fill Layout





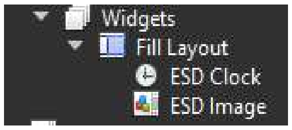
The *Fill Layout* is the base layout for every widget/page since in ESD. A widget always comes with a Fill Layout in it. However, a fill layout in a page has a fixed size due to the screen size defined from the hardware. All widgets in the Fill Layout will auto resize to fit the whole layout area.

Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height

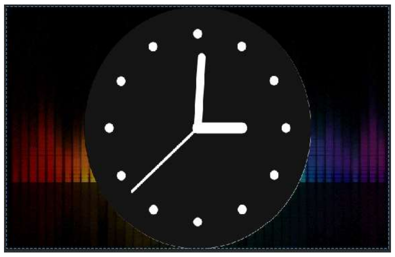
Table 14 - Fill Layout Properties

Fill Layout Implementation

1. Change Widget sequence

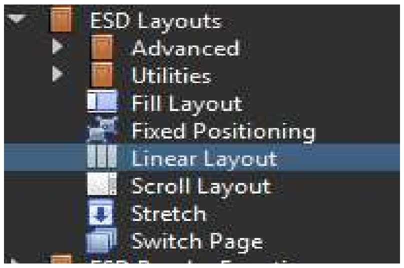


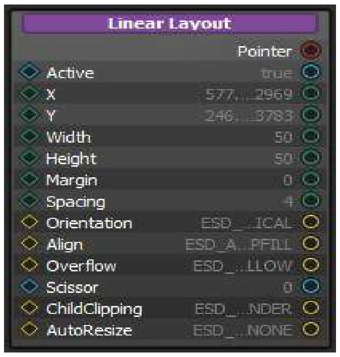
The fill layout sequence is ordered in the sequence of widgets' depth sort. Changing a particular widget to a higher depth sort value, will increase the priority widget's rendering. The example below shows a clock widget and an image widget under the same Fill Layout. Both widgets are filling up the full "Fill Layout" area. However, by setting the depth value of clock widget bigger than the image widget's depth sort, the clock widget will be rendered on top of the image widget.



Linear Layout

The *Linear Layout* supports both horizontal and vertical alignment in ESD. All child widgets will be distributed in the selected orientation. Users cannot change the sequence of widgets in the linear layout from Screen Layout Editor in ESD. However, changing their depth sort value from the property browser will change the sequence.





Linear Layout

	Pointer
Active	true
X	577...2969
Y	246...3783
Width	50
Height	50
Margin	0
Spacing	4
Orientation	ESD_...ICAL
Align	ESD_A...PFILL
Overflow	ESD_...LLOW
Scissor	0
ChildClipping	ESD_...NDER
AutoResize	ESD_...NONE

Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Spacing	Set the spacing gap between adjacent child widgets.
Orientation	Set Linear Layout orientation. It supports: • ESD_ORIENTATION_HORIZONTAL • ESD_ORIENTATION_VERTICAL
Alignment	Set the alignment setting for the child widgets in the layout
Overflow	Set child widget overflow mode. It supports: • ESD_OVERFLOW_ALLOW • ESD_OVERFLOW_CLIP • ESD_OVERFLOW_FILL
Scissor	Set true to enable scissor to clip off overflow

ChildClipping	Set the flags when child clipping should check. It supports: • ESD_CLIP_RENDER • ESD_CLIP_UPDATE • ESD_CLIP_IDLE
Auto Resize	Set auto resize mode for this layout.

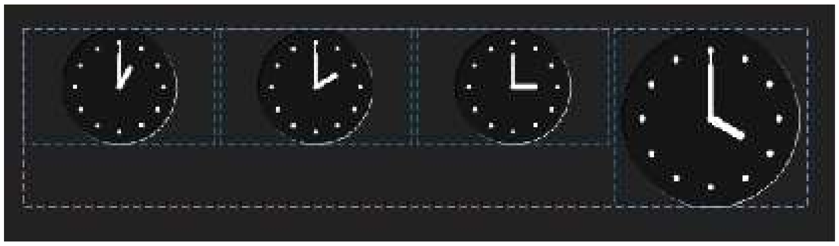
Table 15 - Linear Layout Properties

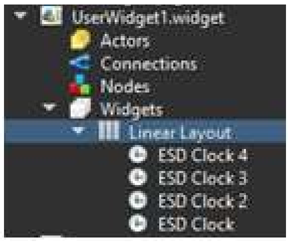
Linear Layout Implementation

1. Horizontal Layout

All child widgets of a linear layout with its orientation are set to "ESD_ORIENTATION_HORIZONTAL". They will be distributed horizontally with some spacing in between adjacent widgets. The layout used as in the example here has parameters:

- Orientation = ESD_ORIENTATION_HORIZONTAL
- Spacing = 4
- Align = ESD_ALIGN_TOPFILL
- Overflow = ESD_OVERFLOW_ALLOW

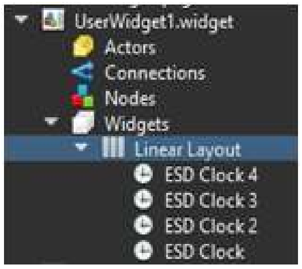


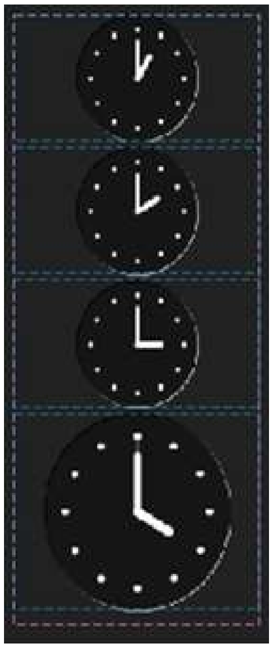


2. Vertical Layout

All child widgets of a linear layout with its orientation set to "ESD_ORIENTATION_VERTICAL". They will be distributed vertically with some spacing in between adjacent widgets. The layout used as in the example here has parameters:

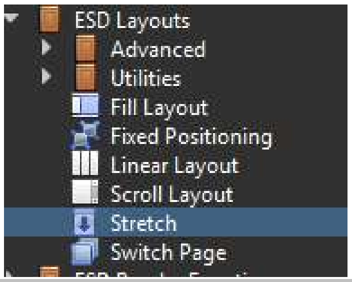
- Orientation = ESD_ORIENTATION_VERTICAL
- Spacing = 4
- Align = ESD_ALIGN_TOPFILL
- Overflow = ESD_OVERFLOW_ALLOW

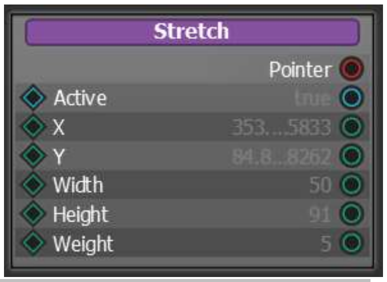




Stretch

The *Stretch Layout* is designed to work together with the linear layout. In the event some widgets need to have higher weightage in the layout, Stretch Layouts are used.





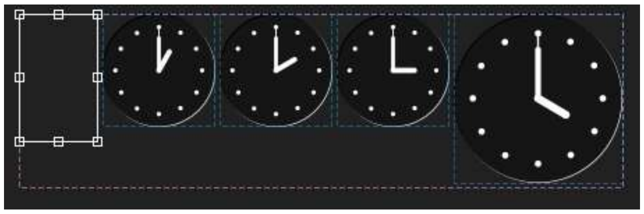
Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Weight	The weight factor when determine the stretch scale in the selected orientation

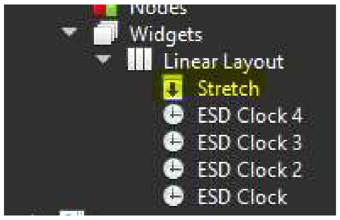
Table 16 - Stretch Layout Properties

Stretch Layout Implementation

1. Space Stretching

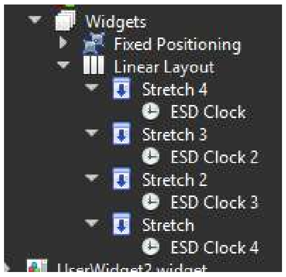
The example here demonstrates how to stretch unused space horizontally. A stretch layout has been added into the Linear Layout. The weight factor used here is 1.

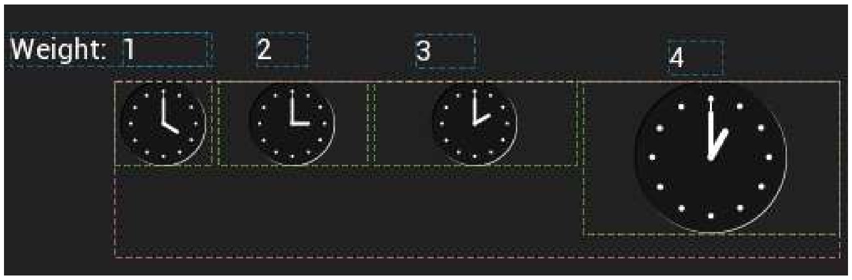




2. Widget Weight Stretching

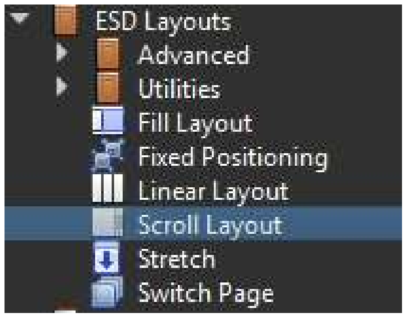
The example here demonstrates widget stretching. In this example, every clock widget has different stretch weight values. The result every widget gains their proportion of width as shown below -

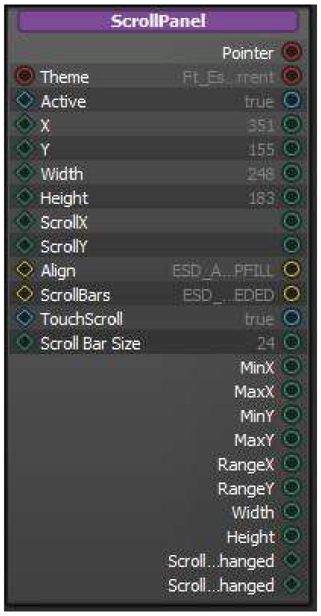




Scroll Layout

The *Scroll Layout* is a support runtime resizable and scrollable layout in ESD.





Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
ScrollX	Getter and Setter for scroll value along X-axis
ScrollY	Getter and Setter for scroll value along Y-axis
Align	Set alignment for its child widgets
Scrollbars	Set when scrollbar should be visible
TouchScroll	Set true to enable touch scroll
Scroll Bar Size	To set scroll bar size when it is visible

Table 17 - Scroll Layout Properties

Scroll Layout Implementation

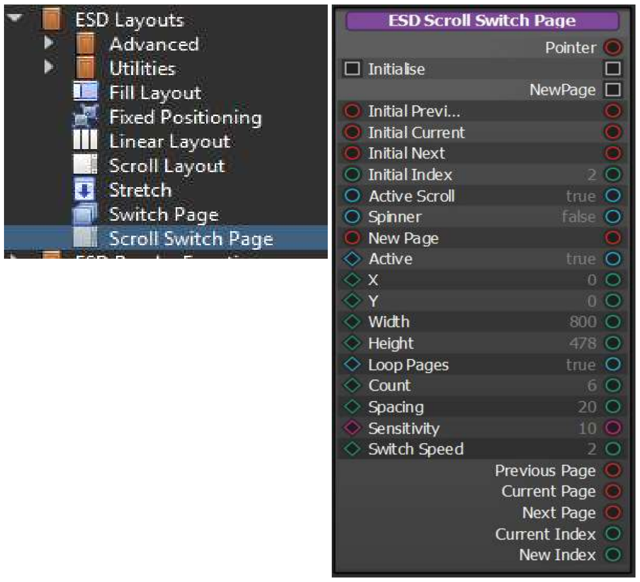
For the implementation of scroll layout, refer to the example project - **"EVE Screen Designer → Examples → Intermediate → ScrollPanel.**

Scroll Switch Page Layout

The *Scroll Switch Page Layout* consists of a switch page manager and a scroll panel for runtime scrollable support.

Scroll Switch Page layout requires 3 lazy page instances and a current index to start with, namely - the previous page instance, the current page instance and the next page instance.

Upon user touch swipe event, the page will switch according to the swipe direction. The behaviour of this scroll switch page layout is as below:



ESD Layouts
- Advanced
- Utilities
- Fill Layout
- Fixed Positioning
- Linear Layout
- Scroll Layout
- Stretch
- Switch Page
- Scroll Switch Page

ESD Scroll Switch Page

		Pointer
Initialise		
		NewPage
Initial Previ...		
Initial Current		
Initial Next		
Initial Index	2	
Active Scroll	true	
Spinner	false	
New Page		
Active	true	
X	0	
Y	0	
Width	800	
Height	478	
Loop Pages	true	
Count	6	
Spacing	20	
Sensitivity	10	
Switch Speed	2	
		Previous Page
		Current Page
		Next Page
		Current Index
		New Index

Figure 76 - Scroll Switch Page Layout

- On non-swiping runtime, it only renders the current page, but runs the update logic of all the 3 pages.
- On swiping event, it displays both the current page and the swiping to page (can be previous and next page instance).
- On touch up event, it will either recover to the original position if the swiped range has not reached the minimum threshold or switch to the new page. The touch scroll will be temporally disabled until it has recovered or switched to the target page.
- Upon switching page complete, the scroll switch page will fire "NewPage" event and requests a new page instance to be allocated and assigns it as the "New Page" input.

Property Name	Description
Pointer	The pointer reference of the widget object
Initialise	The "initialise" slot for initialling the scroll switch page layout
Initial Previous	Initial Previous page instance at start
Initial Current	Initial Current page instance at start
Initial Next	Initial Next page instance at start
Initial Index	Initial Current page index at start
Active Scroll	Set true to override child widgets touch tag event where tag is default(255)
Spinner	Set true to show spinner upon loading the new page instance.
New Page	New page instance input
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Loop Pages	Set true to loop the pages
Count	Set pages count/limit
Spacing	Defines the spacing between pages
Sensitivity	Define the scroll switch sensitivity, range from: 0.05 to 20.00
Switch Speed	Define the switch speed during page transition
Previous Page	The output of previous page instance

Current Page	The output of current page instance
Next Page	The output of next page instance
Current Index	The output of current index
New Index	The output of new index

Table 18 - Scroll Switch Page Layout Properties

Scroll Switch Page layout Implementation

In typical configuration for a scroll switch page layout, the figure below demonstrates a sample setup. It should have the following rules:

- Connect start slot to the "Initialise" slot of scroll switch page layout instance.
- All switching pages should be configured as child widgets of Scroll Switch Page.
- All child page widgets should be lazy allocation and has same width as the Scroll Switch Page.
- Initial Previous Page, Initial Current Page and the Initial Next Page should be active by default while the rest of child pages should be inactive.
- Initial index should be the index of initial current page.
- Recommend using the "Switch" node to trigger the specific "Activate" signal of the target page.
- Recommend using the "Switch Value" node to get the specific target page instances from all the pages.
- Recommend adding header and footer pages which are Sibling widgets of the Scroll Switch Page Layout instance.

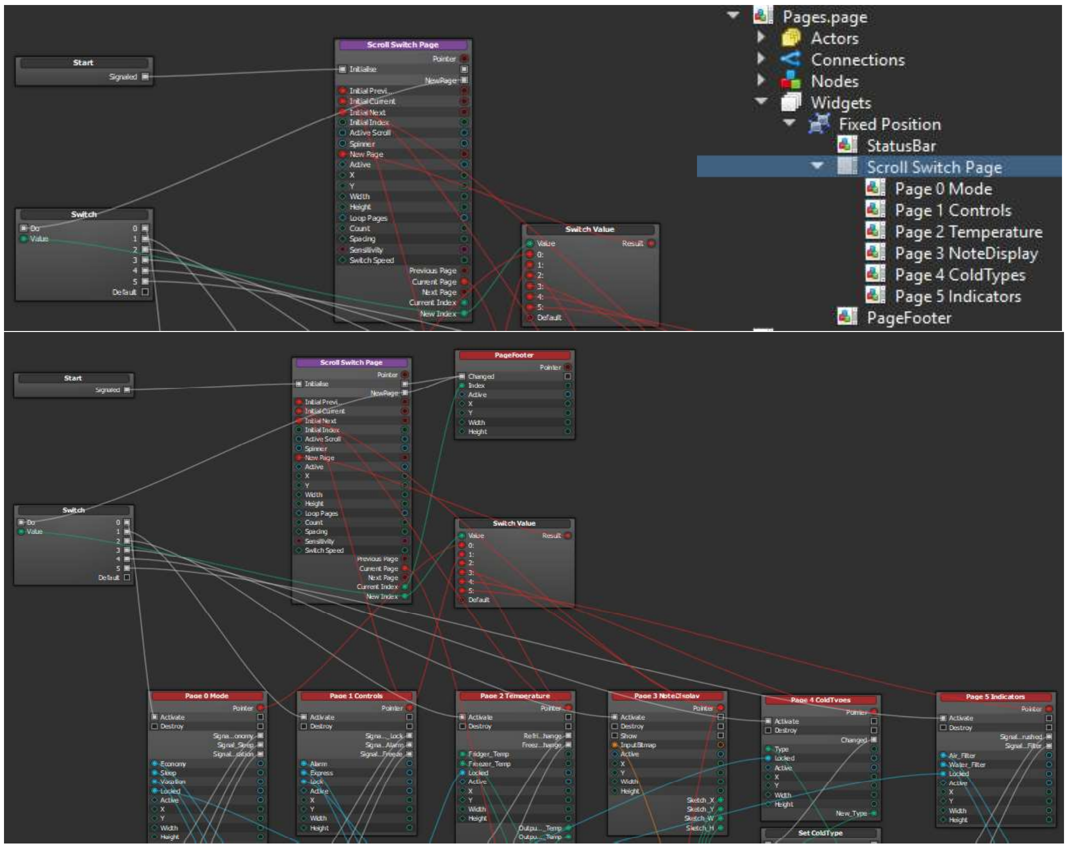


Figure 77 - Sample Scroll Switch Page Implementation

Signal Switch Page Layout

The *Signal Switch Page Layout* is a simplified layout of *Scroll Switch Page Layout* which designed for the user case which touch screen is not supported.

The same as *Scroll Switch Page Layout*, *Signal Switch Page Layout* requires 3 lazy page instances and a current index to start with, namely - the previous page instance, the current page instance and the next page instance.

2 slots, namely NextPageSignal and PreviousPageSignal are provided for the user to switch to next or previous page. The signals to trigger the switch page can be a button trigger, or another signals.

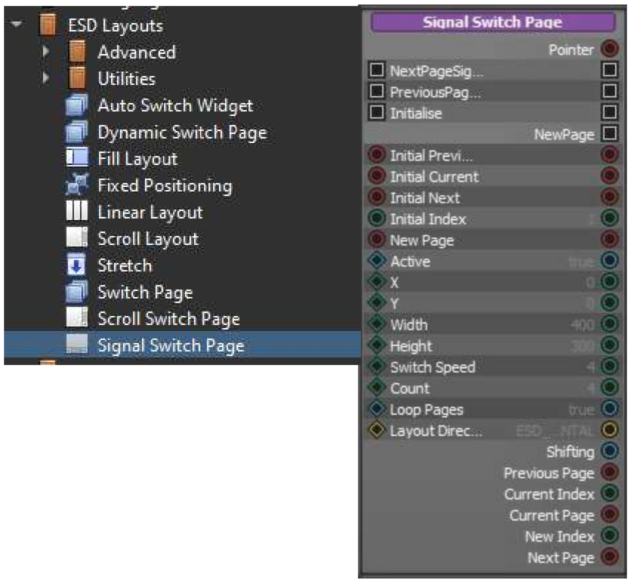


Figure 78 - Signal Switch Page Layout

Similar to *Scroll Switch Page Layout,* on swiping event (trigger by signal), it displays both the current page and the swiping to page (can be previous and next page instance).

Property Name	Description
Pointer	The pointer reference of the widget object
Initialise	The "initialise" slot for initialling the scroll switch page layout
Initial Previous	Initial Previous page instance at start
Initial Current	Initial Current page instance at start
Initial Next	Initial Next page instance at start
Initial Index	Initial Current page index at start
New Page	New page instance input
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Loop Pages	Set true to loop the pages
Count	Set pages count/limit
Spacing	Defines the spacing between pages
Switch Speed	Define the switch speed during page transition
Previous Page	The output of previous page instance
Current Page	The output of current page instance
Next Page	The output of next page instance
Current Index	The output of current index
New Index	The output of new index
Layout Direction	Choose between horizontally or vertically
Shifting	Boolean value to show if page is in the middle of shifting or not

Table 19 - Signal Switch Page Layout Properties

ESD Render Functions

All render functions are non-widget. They are not managed by layout in ESD 4.14. Non-widget functions are to be used only within a widget. These functions are provided as utilities for user to make their own custom widgets.

Note: In order to preview the render functions result in design time, render slot in the custom widget is required.

Elements

The *Elements* are both simple and basic widgets, usually used to construct higher level and more complex widgets. Normally, they do not handle any touch input.

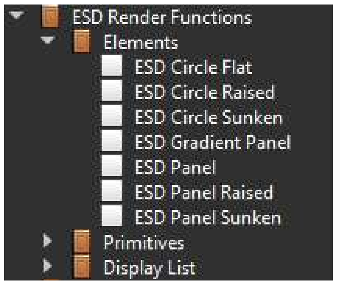
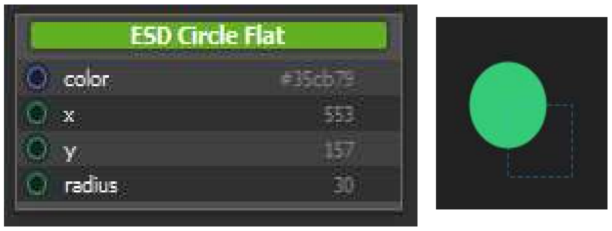


Figure 79 - ESD Elements

ESD Circle

The *Circle* element allows the user to draw circles on the screen. The following styles are available – *ESD Circle Flat; ESD Circle Raised* and *ESD Circle Sunken*.



color #35cb79
x 553
y 157
radius 30

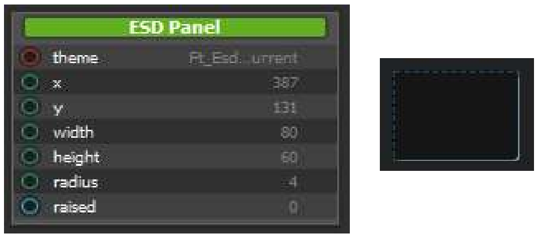
Figure 80 - ESD Circle Element

Property Name	Description
Color	RGB value to be rendered inside the circle
X	x coordinate of central point, in pixels
Y	Y coordinate of central point, in pixels
radius	Radius of the point

Table 20 - ESD Circle Element Properties

ESD Panel

The *ESD Panel* is an adjustable rectangular widget that defines a coloured area. The colour can be specified by a theme or user's selection. Internally, it is constructed by EVE primitives RECTS and SCISSORS. Two styles of ESD Panel are available namely – *ESD Panel Raised* and *ESD Panel Sunken*.



theme Pt_Esd...urrent
x 387
y 131
width 80
height 60
radius 4
raised 0

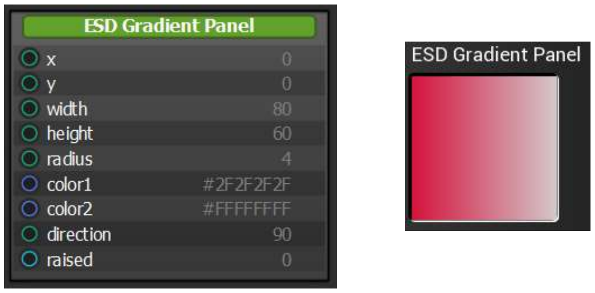
Figure 81 - ESD Panel Element

Property Name	Description
Theme	Theme to be applied on the panel
x	x coordinate of the top-left point, in pixels
y	Y coordinate of the top-left point, in pixels
width	Panel width
height	Panel Height
radius	radius of the corners
raised	Set to true to make panel in raised style

Table 21 - ESD Panel Element Properties

ESD Gradient Panel

The *Gradient Panel* element is an adjustable rectangular widget that defines a gradient-coloured area. The gradient colour can be specified by two colours (one at start and one at the end). The gradient direction is adjustable by changing the direction.


ESD Gradient Panel

x	0
y	0
width	80
height	60
radius	4
color1	#2F2F2F2F
color2	#FFFFFFFF
direction	90
raised	0

Figure 82 - ESD Gradient Panel Element

Property Name	Description
x	x coordinate of the top-left point, in pixels
y	Y coordinate of the top-left point, in pixels
width	Panel width
height	Panel Height
radius	radius of the corners
color1	Start colour for gradient effect
color2	End colour for gradient effect
direction	Gradient effect direction
raised	Set to true to make panel in raised style

Table 22 - ESD Gradient Panel Element Properties

Primitives

A *Primitive* is a special type of logic node which is a wrapper for a rendering function called by Render slot. It has no return value and its properties cannot be output to other logic nodes. Similar to widgets, it appears on the screen, but is unable to handle user input.

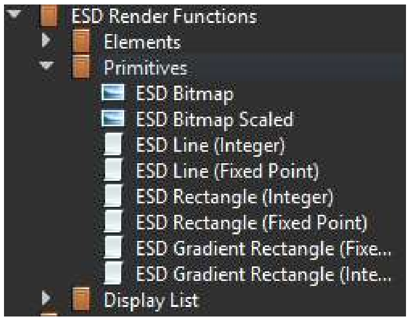
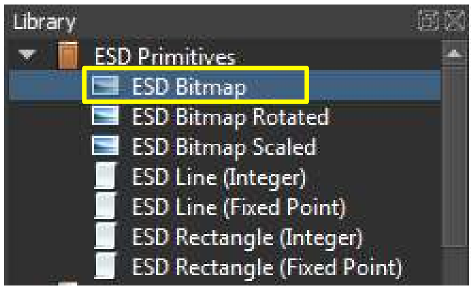


Figure 83 - Primitives

ESD Bitmap

The *ESD Bitmap* object allows users to display a bitmap resource. A bitmap resource is generated from an image file by adding it into a project. A bitmap resource is treated as a single bitmap cell by default with the name "(Bitmap Resource)_0".

For example, if the bitmap resource is named "photo.jpg", the default bitmap resource will be named as "photo_0". Users can define how many bitmap cells they want by specifying the *"CellHeight"* property.



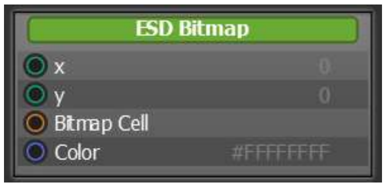


Figure 84 - Bitmap Primitive

Property Name	Description
x	x coordinate of Bitmap, top-left, in pixels
y	Y coordinate of Bitmap, top-left, in pixels
Bitmap Cell	Bitmap cell to display
Color	Color to be applied to bitmap

Table 23 - ESD Bitmap Properties

To add a bitmap

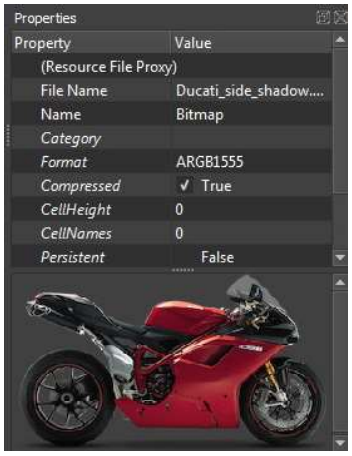
Click and select **"File → Add"** or click from the toolbar. A bitmap cell is then assigned. ESD 4.14 supports the following image file formats as input – *".png", ".jpg"* and *".jpeg"*. Bitmap resources can contain a single or multiple cells based on the *CellHeight* value.

If the value of the *"CellHeight"* is zero, then the number of cells is one.

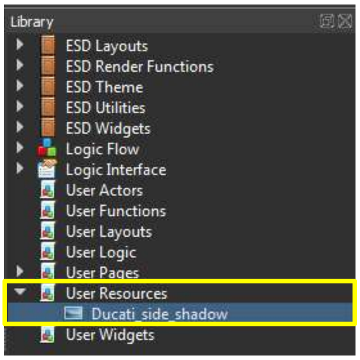
If the value of the *"CellHeight"* is non-zero, then the number of cells value is calculated as shown below –

```
Number of Cell = Bitmap Height / CellHeight
```

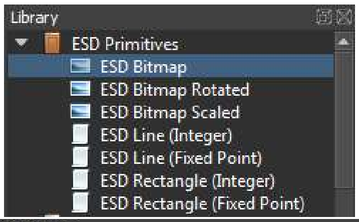
Upon adding a Bitmap, the new bitmap resource will be added into the library under the category – *"User Resources"*.



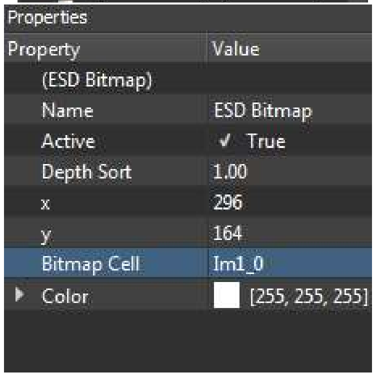
Property	Value
(Resource File Proxy)	
File Name	Ducati_side_shadow....
Name	Bitmap
Category	
Format	ARGB1555
Compressed	✓ True
CellHeight	0
CellNames	0
Persistent	False



To use an *ESD Bitmap*, drag and drop the ESD Bitmap Node into the *Layout Editor* or *Logic Node Editor*.



Choose a Bitmap Cell to display from the dropdown list.


Properties

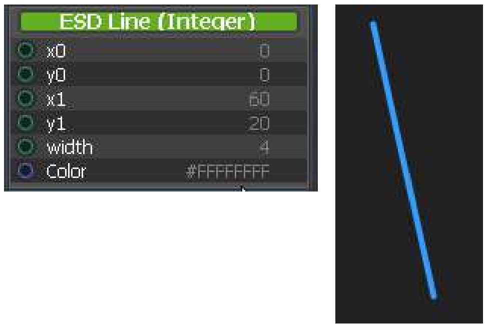
Property	Value
(ESD Bitmap)	
Name	ESD Bitmap
Active	✓ True
Depth Sort	1.00
x	296
y	164
Bitmap Cell	Im1_0
Color	[255, 255, 255]

The list of Bitmap Cell values is updated automatically after a new image file is added into a project.

Use ESD Image to display bitmap as ESD bitmap is non-widget, and used only as a rendering function in ESD 4.14. Users are required to upgrade ESD Bitmap to ESD Image from ESD 3.X.

ESD Line

The *ESD Line* allows users to draw lines on the screen.


ESD Line (Integer)

x0	0
y0	0
x1	60
y1	20
width	4
Color	#FFFFFFFF

Figure 85 - ESD Line

Use ESD Widgets – Line Widget to display line as ESD line is non-widget, and used only as a rendering function in ESD 4.14. Users are required to upgrade ESD Line to Line Widget.

Property Name	Description
x0	x – coordinate of the start point, in pixels
y0	y – coordinate of the start point, in pixels
x1	x – coordinate of the end point, in pixels
y1	y – coordinate of the end point, in pixels
Width	Line width, in pixel
Color	Line color, in RGBA format

Table 24 - ESD Line Properties

Fixed Point Variant

ESD Line (Fixed point) expresses the properties with pixel attribute in fixed point format. It provides more control to lines.

ESD Rectangle

The *ESD Rectangle* allows users to draw rectangle on the screen.

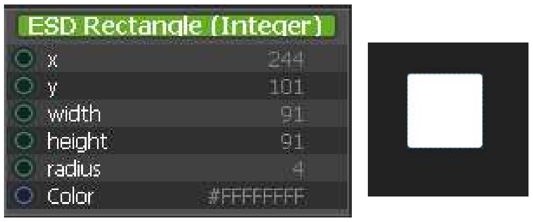

ESD Rectangle (Integer)
x 244
y 101
width 91
height 91
radius 4
Color #FFFFFFFF

Figure 86 - ESD Rectangle

Property Name	Description
X	x coordinate of Bitmap, top-left, in pixels
Y	Y coordinate of Bitmap, top-left, in pixels
width	width, in pixels
height	height, in pixels
radius	radius of the round corner, in pixels
Color	color of the rectangle

Table 25 - ESD Rectangle Properties

Fixed Point Variant

ESD Rectangle (Fixed point) expresses the properties with pixel attribute in fixed point format. It provides more control to rectangles.

Display List

The Display List provides a collection of utilities functions related to EVE Display List.

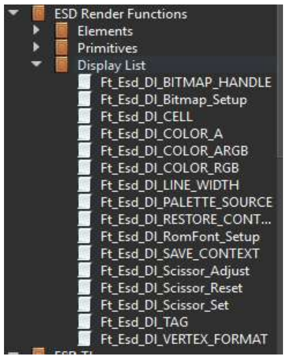

ESD Render Functions
Elements
Primitives
Display List
Ft_Esd_Dl_BITMAP_HANDLE
Ft_Esd_Dl_Bitmap_Setup
Ft_Esd_Dl_CELL
Ft_Esd_Dl_COLOR_A
Ft_Esd_Dl_COLOR_ARGB
Ft_Esd_Dl_COLOR_RGB
Ft_Esd_Dl_LINE_WIDTH
Ft_Esd_Dl_PALETTE_SOURCE
Ft_Esd_Dl_RESTORE_CONT...
Ft_Esd_Dl_RomFont_Setup
Ft_Esd_Dl_SAVE_CONTEXT
Ft_Esd_Dl_Scissor_Adjust
Ft_Esd_Dl_Scissor_Reset
Ft_Esd_Dl_Scissor_Set
Ft_Esd_Dl_TAG
Ft_Esd_Dl_VERTEX_FORMAT

Figure 87 - ESD Display List

ESD Theme

A *Theme* is a collection of colours which can be used across the application in order to maintain a consistent style. The library browser provides the necessary logic nodes to make most use of the theme function.

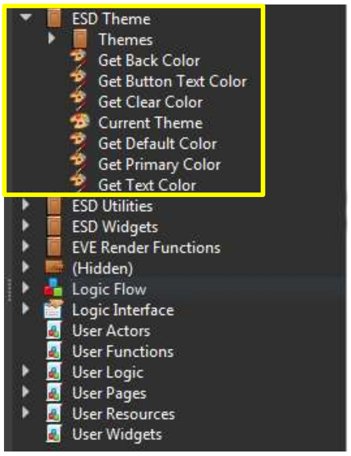
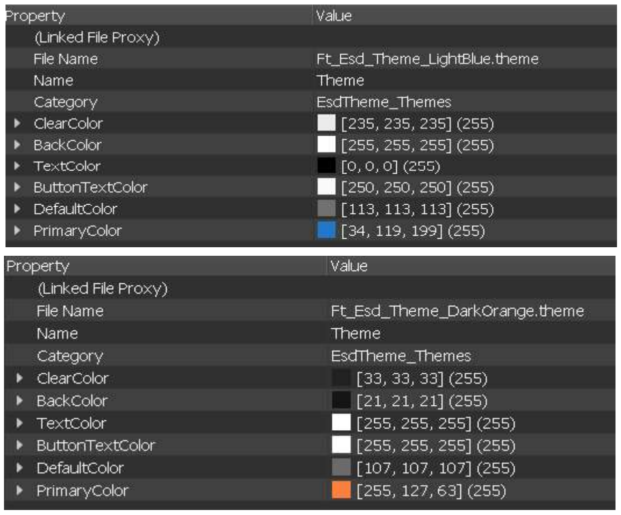


Figure 88 - Library Browser - ESD Theme

Built In Themes

ESD provides two *Built-in Themes* for users to configure the colour scheme of project. They are: *Ft_Esd_Theme_LightBlue* theme and *Ft_Esd_Theme_DarkOrange* theme.



Property	Value
(Linked File Proxy)	
File Name	Ft_Esd_Theme_LightBlue.theme
Name	Theme
Category	EsdTheme_Themes
ClearColor	[235, 235, 235] (255)
BackColor	[255, 255, 255] (255)
TextColor	[0, 0, 0] (255)
ButtonTextColor	[250, 250, 250] (255)
DefaultColor	[113, 113, 113] (255)
PrimaryColor	[34, 119, 199] (255)

Property	Value
(Linked File Proxy)	
File Name	Ft_Esd_Theme_DarkOrange.theme
Name	Theme
Category	EsdTheme_Themes
ClearColor	[33, 33, 33] (255)
BackColor	[21, 21, 21] (255)
TextColor	[255, 255, 255] (255)
ButtonTextColor	[255, 255, 255] (255)
DefaultColor	[107, 107, 107] (255)
PrimaryColor	[255, 127, 63] (255)

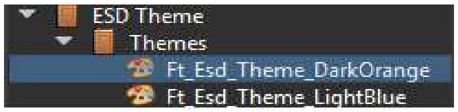


Figure 89 - ESD Built-in Themes

Table 26 and Table 27 provides information about the built-in theme's property -

Property Name	Type	Value (RGBA)	Description
Name	String	Theme	Ft_Esd_Theme_LightBlue
ClearColor	RGBA	235,235,235(255)	Clear Color
BackColor	RGBA	255,255,255(255)	Background Color
TextColor	RGBA	0,0,0(255)	Text Color
ButtonTextColor	RGBA	250,250,250(255)	Button Text Color
DefaultColor	RGBA	113,113,113(255)	Default Color
PrimaryColor	RGBA	34,119,199(255)	Primary Color

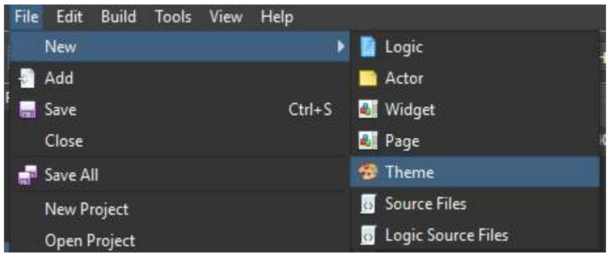
Table 26 - Ft_Esd_Theme_LightBlue Theme

Property Name	Type	Value (RGBA)	Description
Name	String	Theme	Ft_Esd_Theme_DarkOrange
ClearColor	RGBA	33,33,33(255)	Clear Color
BackColor	RGBA	21,21,21(255)	Background Color
TextColor	RGBA	255,255,255(255)	Text Color
ButtonTextColor	RGBA	255,255,255(255)	Button Text Color
DefaultColor	RGBA	107,107,107(255)	Default Color
PrimaryColor	RGBA	255,127,63(255)	Primary Color

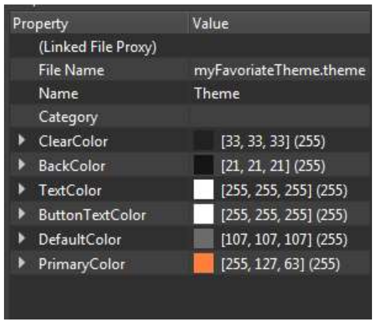
Table 27 - Ft_Esd_Theme_DarkOrange Theme

To create a new theme:

1. Click and select **"File → New → Theme"**.

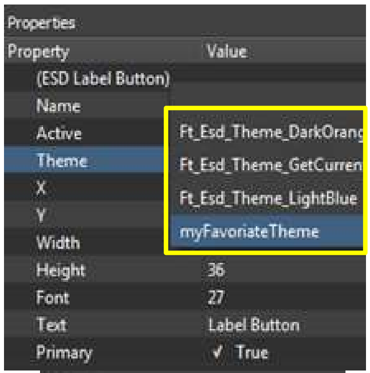


2. A new "*.*theme"* file is created within the Project folder. Using the Property Editor, users can configure it as per their requirement.



Property	Value
(Linked File Proxy)	
File Name	myFavoriateTheme.theme
Name	Theme
Category	
ClearColor	[33, 33, 33] (255)
BackColor	[21, 21, 21] (255)
TextColor	[255, 255, 255] (255)
ButtonTextColor	[255, 255, 255] (255)
DefaultColor	[107, 107, 107] (255)
PrimaryColor	[255, 127, 63] (255)

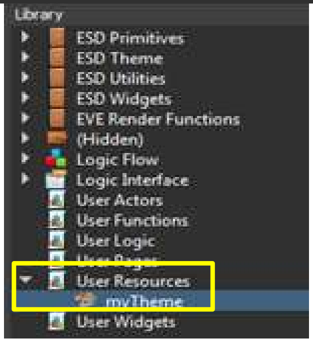
3. The newly-created theme will be available across the project and may be applied it to other widgets with the theme property.



Properties

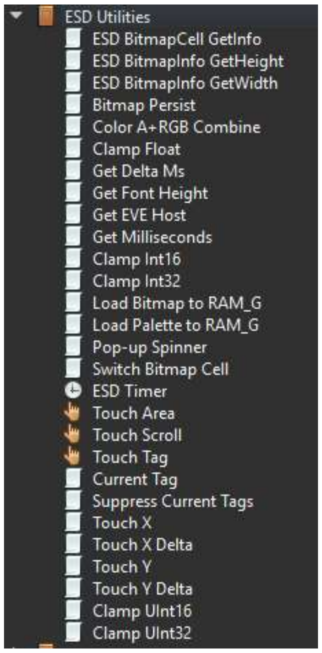
Property	Value
(ESD Label Button)	
Name	
Active	Ft_Esd_Theme_DarkOrang
Theme	Ft_Esd_Theme_GetCurren
X	Ft_Esd_Theme_LightBlue
Y	myFavoriateTheme
Width	
Height	36
Font	27
Text	Label Button
Primary	✓ True

4. The newly created theme will also be available under **User Resources** in the *Library Browser*. Users can drag and drop the theme to the logic editor.



ESD Utilities

ESD Utilities contains the collection ESD utilities functions. The following table provides the list of ESD Utilities and its functionality.

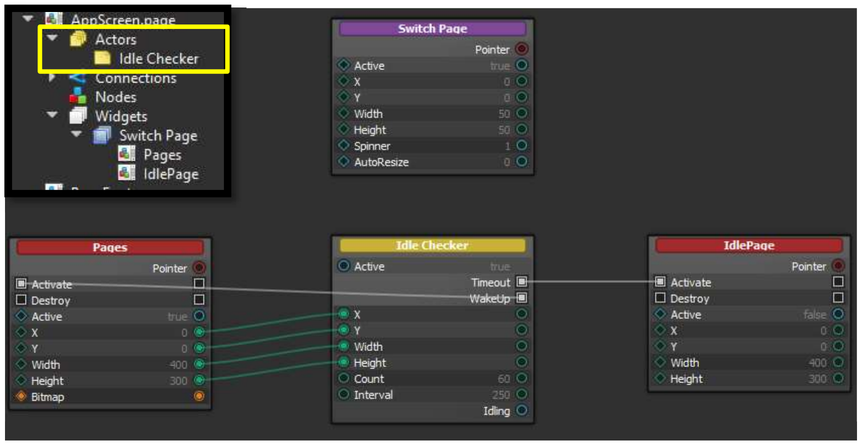


ESD Utility	Description
ESD BitmapCell GetInfo	Retrieves the BitmapCell information with given bitmap reference.
ESD BitmapInfo GetHeight	Returns the BitmapCell Height of given bitmap reference.
ESD BitmapInfo GetWidth	Retrieves the BitmapCell Width of given bitmap reference.
Bitmap Persist	To persist the given bitmap object
Color A+RGB Combine	Merge Alpha to RGB, to become RGBA colour
Clamp Float	Clamp float value with min and max limits
Get Delta Ms	Get Delta time difference in milliseconds since last frame update called
Get Font Height	Return Font Height
Get EVE Host	Get EVE Host
Get Milliseconds	Get time in milli-seconds for current frame
ESD Idle Checker	Check when to sleep and wake up based on the touch events.
Clamp Int16/UInt16	Clamp Int16/UInt16 value with min and max limits
Clamp Int32/UInt32	Clamp Int32/UInt32 value with min and max limits
Load Bitmap to RAM_G	Load Bitmap to RAM_G
Load Palette to RAM_G	Load Palette to RAM_G
Pop-up Spinner	Pop-up Spinner when the frame is rendered
Switch Bitmap Cell	Switch Bitmap Cell
ESD Timer	ESD Timer Actor
Touch Area	Touch Area Actor
Touch Scroll	Touch Scroll Actor
Touch Tag	Touch Tag Actor
Current Tag	Current Touch Tag
Suppress Current Tags	Suppress Current Tags
Touch X/Y	Touch X/Y coordinate
Touch X/Y Delta	Touch X/Y Delta

Table 28 - ESD Utilities & Description

ESD Idle Checker

ESD Idle Checker Actor provides an actor to check when the application should go to sleep mode or wake up from sleep mode. One of the main use cases of the idle checker actor is to check for idling and wake up event for displaying the screen saver page. The picture on the right demonstrated in "AppScreen.page" uses Idle Checker to switch between application pages and the idle page. The idle checker has an interval of 250 ms and count of 60 intervals. In other words, it may timeout when there is no touch event for 15 secs (60 * 250 = 15000 ms).



Logic Flow

A *Logic Flow* is an ESD built-in logic node. It is used to define the logic or control flow and contains the following nodes:

- *Condition*
- *Binary Condition*
- *Local Method*
- *Binary Operator*
- *Ternary Operator*
- *Unary Operator*
- *Sequence*
- *Switch*
- *Switch Value*
- *Set Variable*
- *Watch Variables*

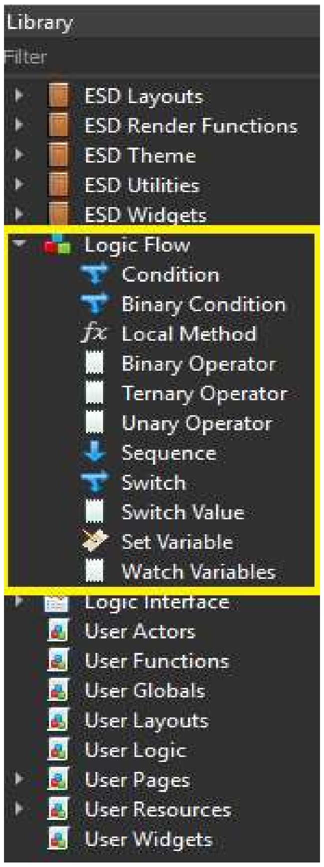
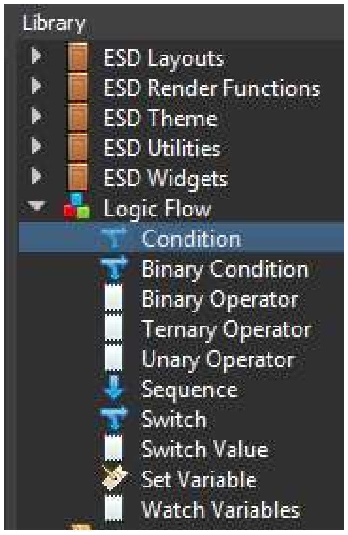


Figure 90 - Logic Flow

Condition

Condition node provides a set of rules to be performed if a certain condition is met. In other words, it allows applying decision points.



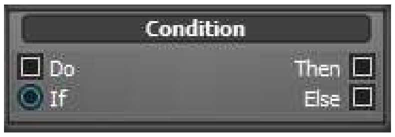


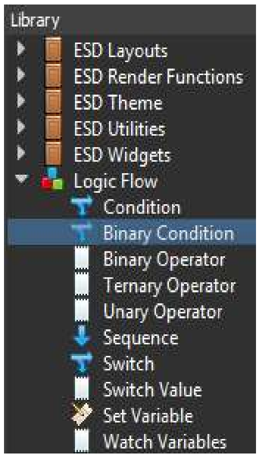
Figure 91 - Logic Flow – Condition Node

Whenever a condition node is called, a *"Then"* condition will be called when the *"If"* expression is True, otherwise an *"Else"* condition will be called.

Ensure that the input for the "If" connection is a Boolean variable (i.e., *True* or *False*).

Binary Condition

Binary Condition node provides a set of rules to be performed based on the result of a binary operator on two input values. In other words, it allows applying decision points.



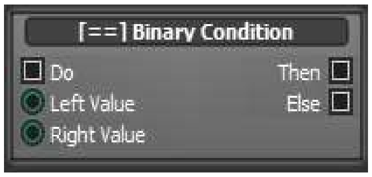


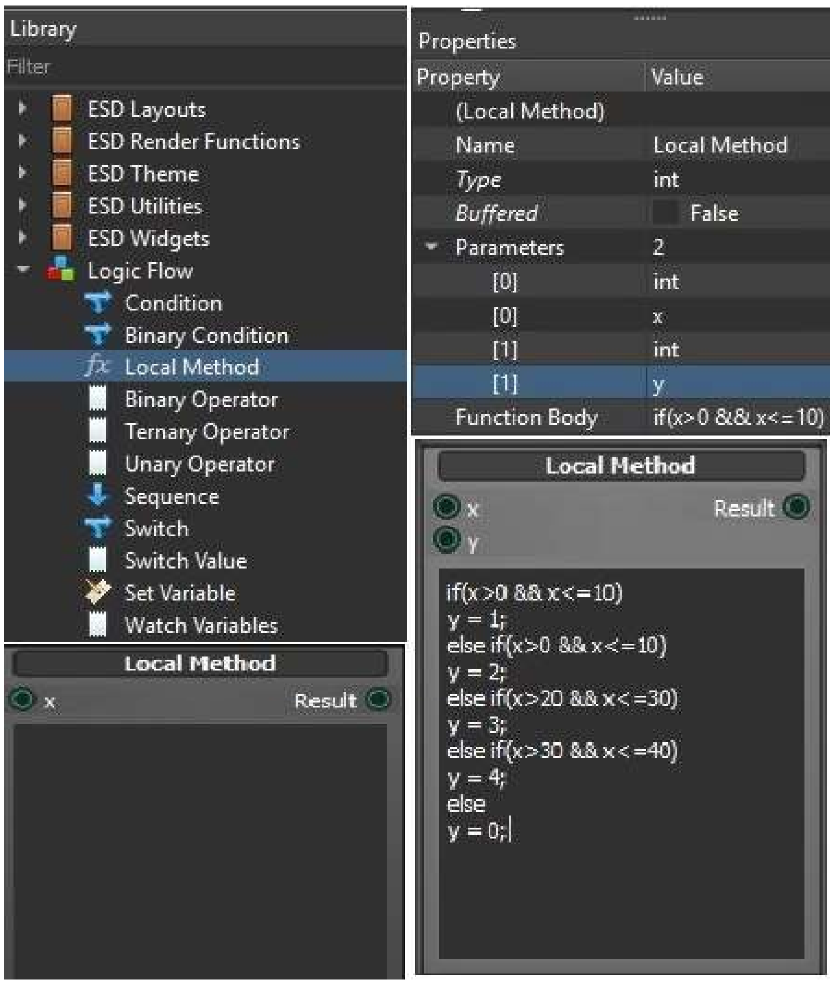
Figure 92 - Logic Flow – Binary Condition Node

Whenever a binary condition node is called, a *"Then"* condition will be called when the result of binary operator on left and right value is True, otherwise an *"Else"* condition will be called.

The binary operator is specified by user and it will be evaluated when the node is called through "*Do*" input call.

Local Method

Local Method node is used to add a visual function node. It works as a place holder where user will provide input using parameters and put the function body in logic editor. It gives the option to specify the no. of parameters, parameter type and parameter name to the function as input. User can select the parameter count using scroll bar and add parameters (type and name) from the properties tab.



Property	Value
(Local Method)	
Name	Local Method
Type	int
Buffered	False
Parameters	2
[0]	int
[0]	x
[1]	int
[1]	y
Function Body	if(x>0 && x<=10)

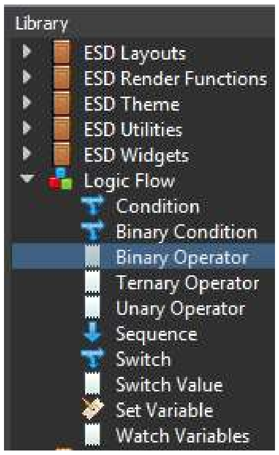
```
if(x>0 && x<=10)
y = 1;
else if(x>0 && x<=10)
y = 2;
else if(x>20 && x<=30)
y = 3;
else if(x>30 && x<=40)
y = 4;
else
y = 0;
```

Figure 93 - Logic Flow – Local Method

- "Parameter Type" – Type of input parameter to be selected from drop down menu i.e., int, float, char* or Ft_Esd_Theme* etc.
- "Parameter Name" – Name of the variable should be according to ANSI C coding standard.
- "Result" – Return value from this function which will be assigned to a variable or input to new widget or assigned to output directly.

Binary Operator

This node contains *Binary Operator* that operates on two operands (inputs) and manipulates them to return an output. For example: *Result=Left Value * Right Value*.



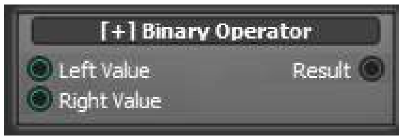


Figure 94 - Logic Flow - Binary Operator Node

The following table provides a list of binary operators supported by ESD 4.14.

Binary Operator	Description
==	Equal to comparison operator
!=	Not Equal to comparison operator
&&	Logical AND operator
&	Binary AND operator
||	Logical OR operator
|	Logical OR operator
*	Multiplication operator
/	Division operator
%	Modulus operator
^	Binary XOR operator
+	Addition operator
-	Subtraction operator
,	Comma operator
>	Greater than operator
<	Less than operator
>=	Greater than or equal to
<=	Less than or equal to
<<	Binary Left Shift Operator
>>	Binary Right Shift Operator

Table 29 - Binary Operators Supported by ESD 4.14

Ternary Operator

This node contains *Ternary Operator* that operates on three arguments. First is the comparison argument; second is the result upon a true comparison and the third is the result upon a false comparison.

For example: *Result= condition ? "Value if true": "Value if false".*

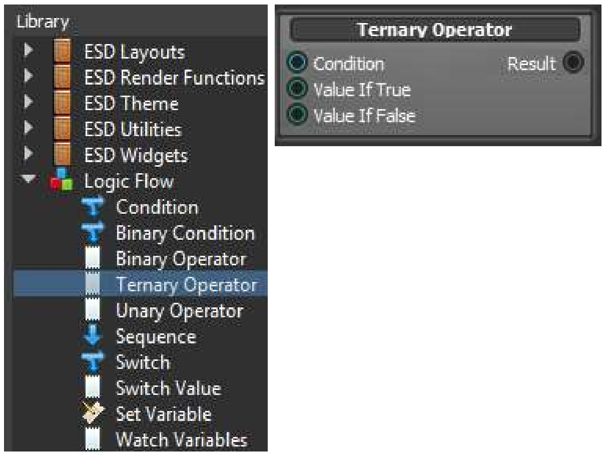


Figure 95 - Logic Flow - Ternary Operator Node

Unary Operator

This node contains *Unary Operator* that operates on a single operand (input). For example: *~.*

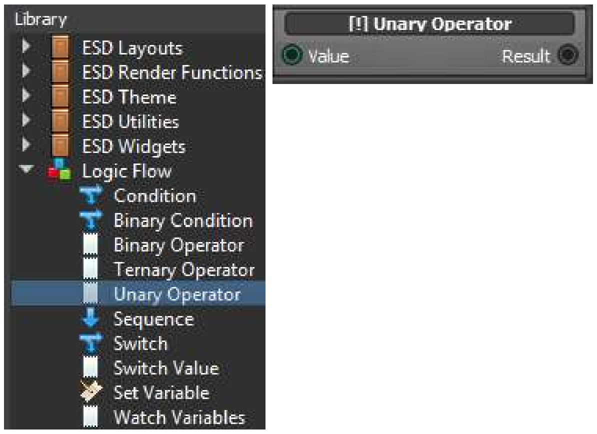


Figure 96 - Logic Flow - Unary Operator Node

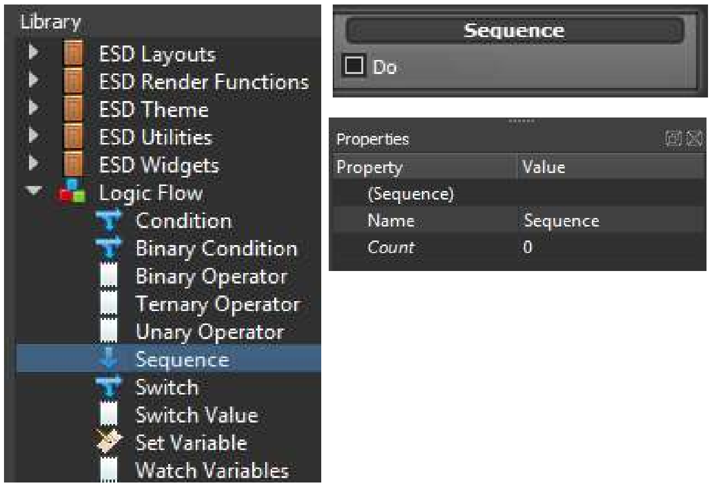
The following table provides a list of unary operators supported by ESD.

Unary Operator	Description
!	Logic Negation
&	Address of
*	De-reference
+	Positive operator
-	Negative operator
~	One's complement
!!	Non-zero value converts to 1and other values converts to 0.

Table 30 - Unary Operators Supported by ESD

Sequence

"*Sequence*" node allows users to define a series of calling actions. When the "Do" input call is invoked, the calls will be triggered sequentially. The number of calls is specified by users in property "Count".

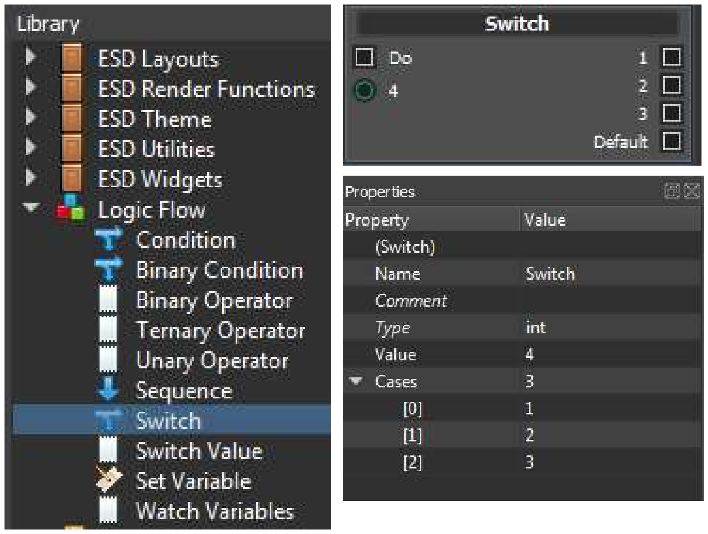


Property	Value
(Sequence)	
Name	Sequence
Count	0

Figure 97 - Logic Flow – Sequence Node

Switch Node

"*Switch*" node will call different branches based on the input value, like a "Switch-Case" statement in C language. Users can define any number of branches and the values to be compared with input value through "Cases" property. Figure 98 shows an example of 3 branches defined and called respectively if input value is equal to 1 or 2 or 3. Since, the input value is pre-set to 4; the default branch is called if no connection is made on it.

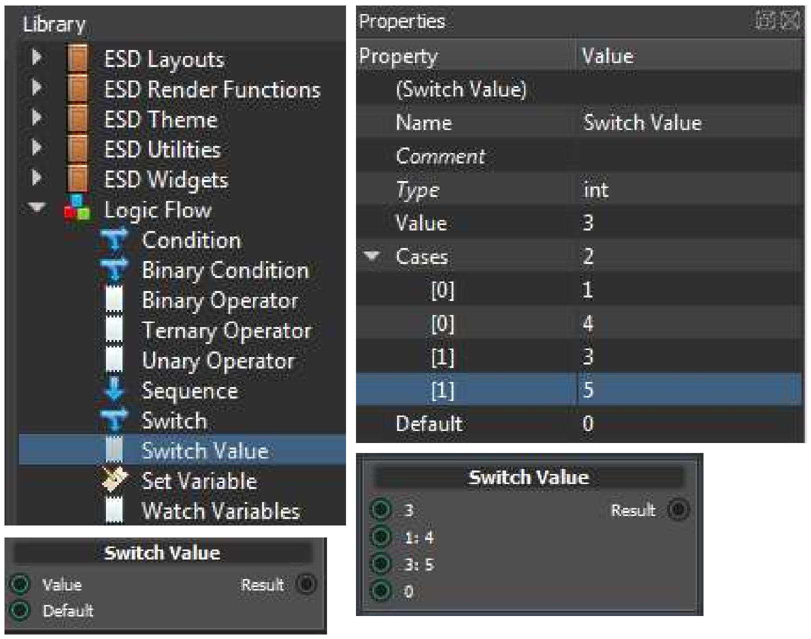


Property	Value
(Switch)	
Name	Switch
Comment	
Type	int
Value	4
Cases	3
[0]	1
[1]	2
[2]	3

Figure 98 - Logic Flow – Switch Node

Switch Value

"*Switch Value*" is a node to select one output value from multiple predefined values in "Cases" property based on input value. The example in Figure 99 shows 2 "*Cases*" branch and property "*Cases*" 0 defines such logic: The "*Result*" value is 4 if input value is 1. Therefore, the result value is 5 when the input value is 3.



Properties

Property	Value
(Switch Value)	
Name	Switch Value
Comment	
Type	int
Value	3
Cases	2
[0]	1
[0]	4
[1]	3
[1]	5
Default	0

Figure 99 - Logic Flow – Switch Value

Set Variable

The *Set Variable* node is used to set the values of a variable. It links an event call with a data binding to a Writer (output a functional call through "writer") or a *Variable* (sets a variable to the value).

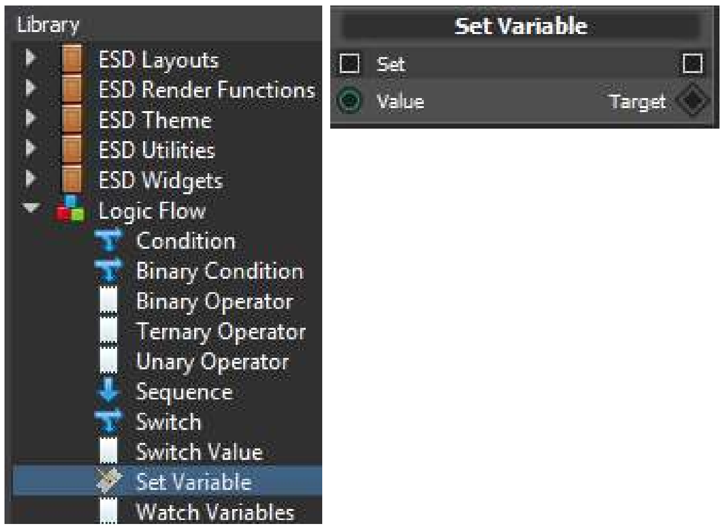
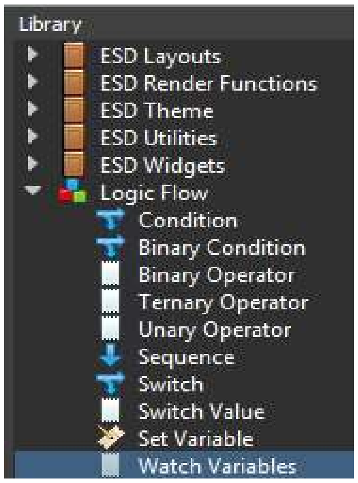


Figure 100 - Logic Flow - Set Variable Node

Watch Variables

The Watch variables node is used to monitor a set of variable updates. A single watch variable node can monitor 'n' number of variables which can be different (primitive) types. However, the more the input variables have connected, the more frequent the output "Changed" will be triggered. User can also connect input slots:



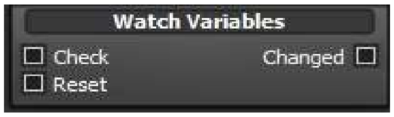


Figure 101 - Logic Flow - Watch Variable Node

- "Check" – used to trigger an immediate check for the input variables against previously checked values.
- "Reset" – set base values as current input values, and monitors changes from there.

This watch variable could be very useful during the debugging, or multiple input trigger events. It simplifies nested-if conditional flows.

Logic Interface

A *Logic Interface* is an ESD built-in logic node. It is used to define an interface and contains the following nodes:

- *Input*
- *Output*
- *Signal*
- *Slot*
- *Variable*
- *Writer*
- *Widget Interface*

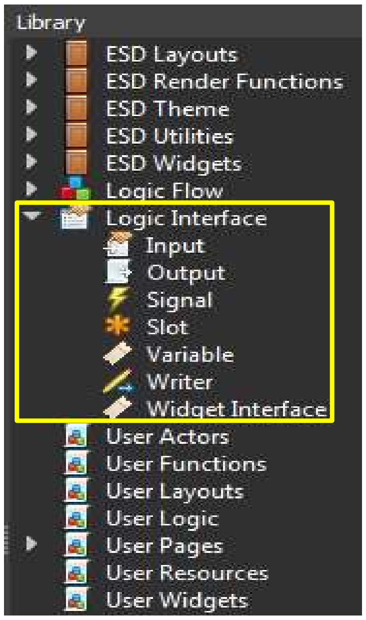
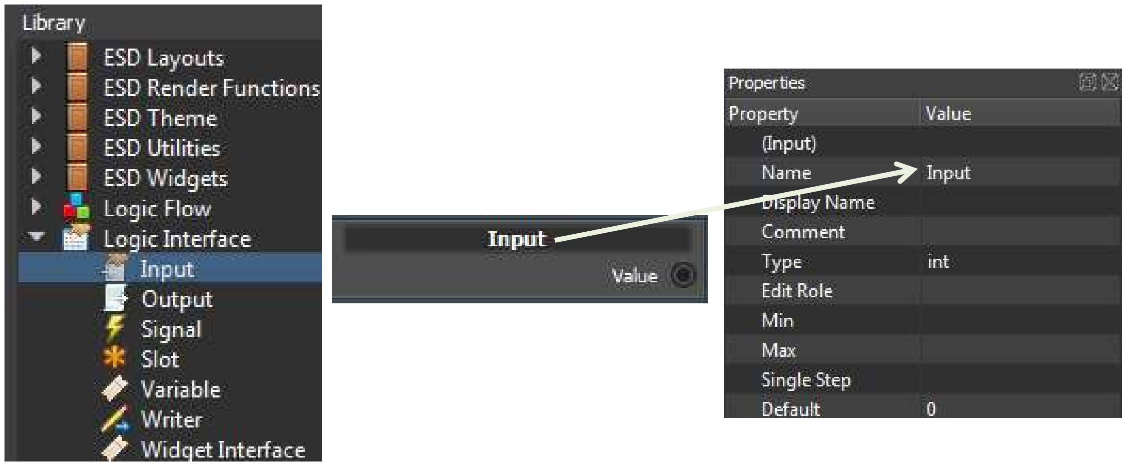


Figure 102 - Logic Interface

Input

The *Input* node is used to create an input interface for its parent logic node to bind incoming data. For instance, while creating a widget, an input node will define a property for the widget whose value depends on the connection. Users can configure its property through the Property Editor.

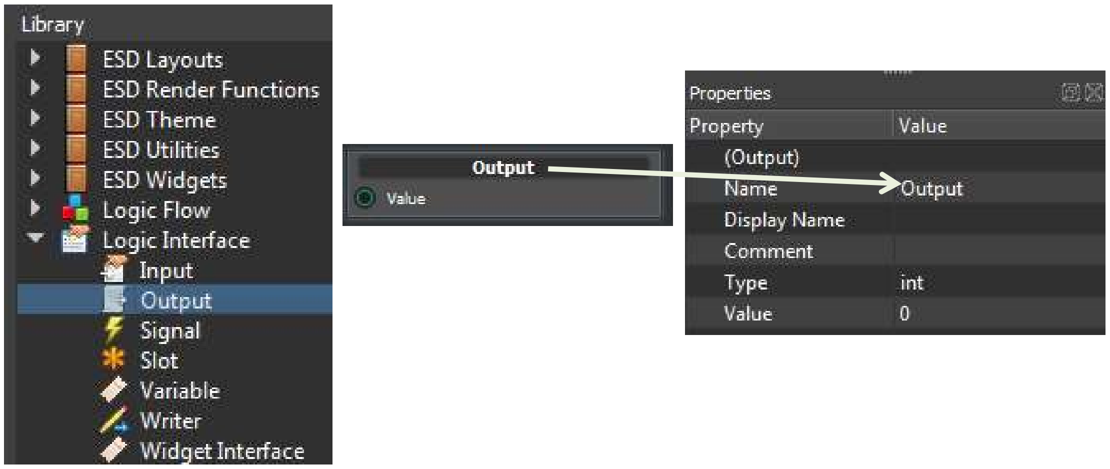


Property	Value
(Input)	
Name	Input
Display Name	
Comment	
Type	int
Edit Role	
Min	
Max	
Single Step	
Default	0

Figure 103 - Logic Interface - Input Node

Output

The *Output* node is used to create an output interface for its parent logic to bind the outgoing data. For instance, while creating a widget, an output node will define an interface for the widget to display values to the users. Users can configure its property through the Property Editor.



Property	Value
(Output)	
Name	Output
Display Name	
Comment	
Type	int
Value	0

Figure 104 - Logic Interface - Output Node

Signal

The *Signal* node is used to create an outgoing event function call. Users can configure its property through the Property Editor.

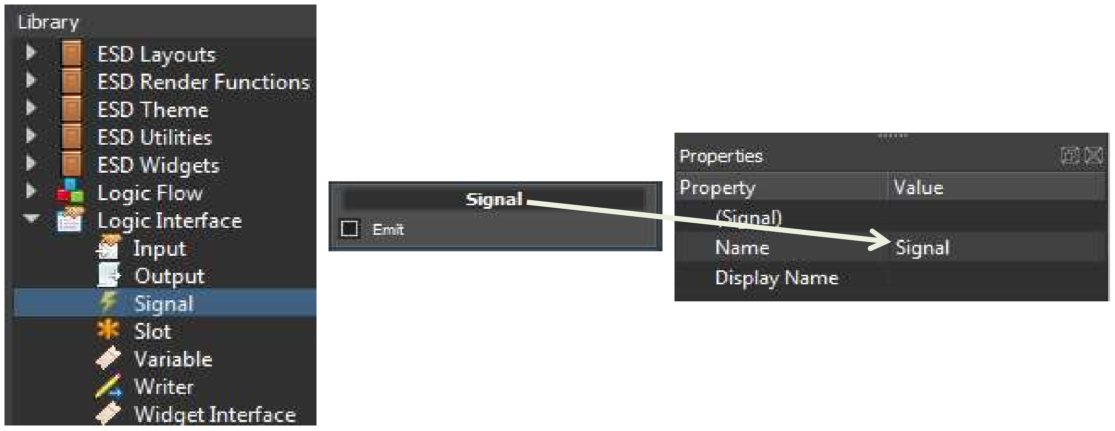


Figure 105 - Logic Interface - Signal Node

Slot

The *Slot* node is used to create an incoming event function call.

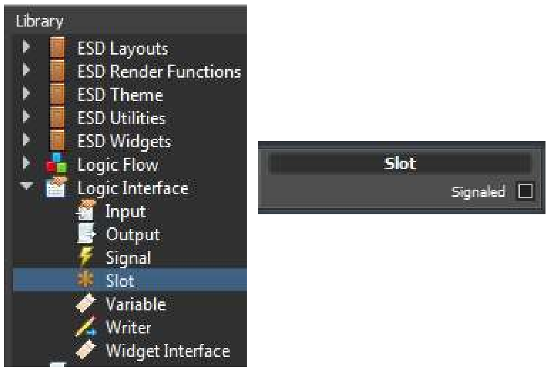


Figure 106 - Logic Interface - Slot Node

Built-in Slot

There are a number of *built-in slots* available for Widgets, Pages and Applications which are automatically called through the node hierarchy. The following table provides the list of built-in slots.

Built-in Slot	Description
Start	Called when the node is created in memory
Render	Writing rendering calls to the co-processor command buffer. It is repeatedly called.
Update	Called once before the frame render. It is called frequently.
Idle	Called repeatedly while waiting for the frame to flip
End	Called when the node is removed from the memory

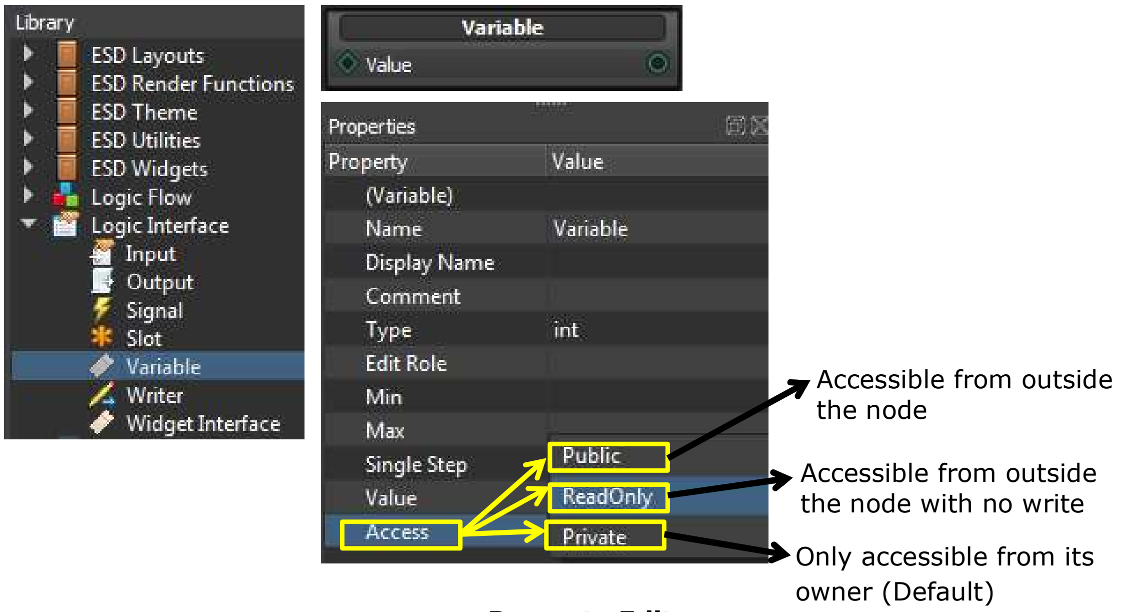
Table 31 - Built-in Slots

The following pseudo-code may help you understand how these slots are invoked:

```
//Application is booting up and started
Start()
//Now UI is launched
while(UI is running)
{
    Update();
    Render();
    Idle();
}
//Application is ending
End()
```

Variable

The *Variable* node is used to define a plain C variable. Users can drag and drop the Variable from the Library browser to the logic node editor and configure its property through the Property Editor.



Property	Value
(Variable)	
Name	Variable
Display Name	
Comment	
Type	int
Edit Role	
Min	
Max	
Single Step	
Value	
Access	

Property Editor

Figure 107 - Logic Interface - Variable Node

Writer

The *Writer* node is used to define a value writing interface for its parent node. Generally, it is used to setup the value for the external variable. For instance, while creating the ESD Spinbox widget, the "Changed" writer is defined to update the connected external variables of Spinbox widget once Spinbox value is changed.

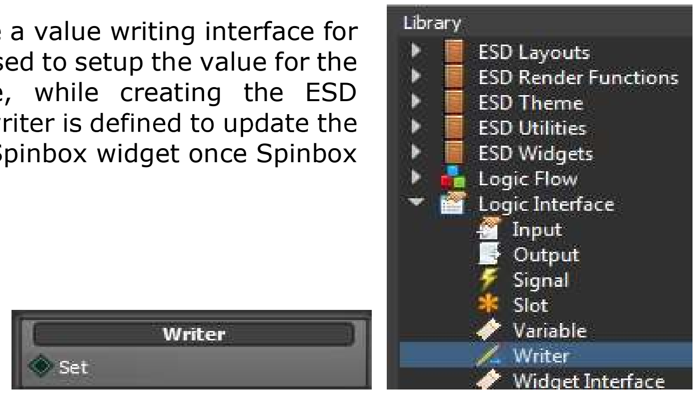


Figure 108 - Logic Interface - Writer Node

Widget Interface

The *Widget Interface* node shows the interface of base widget interface. Since ESD 4.0, all layouts, pages and widgets have been implemented from this widget interface. For advanced users, this widget interface node could be useful for designing user customized widget in order to use its base interface for accessing information from ESD widget framework.

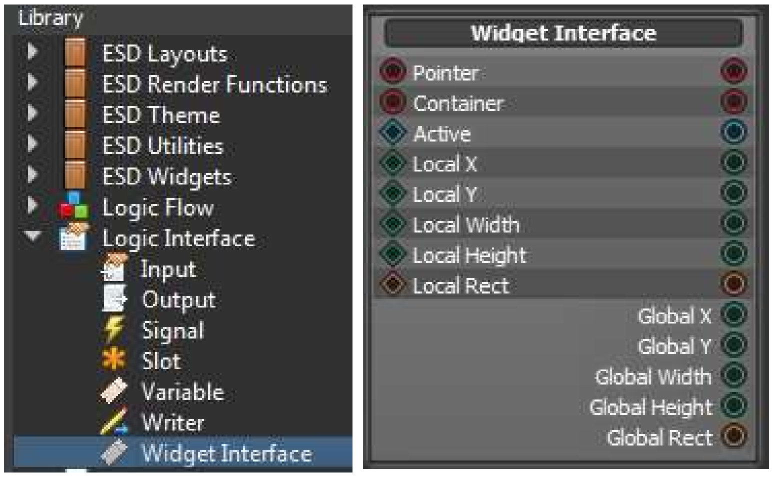


Figure 109 - Widget Interface

Property Name	Description
Pointer	Provides the pointer of this widget, allow other widget/layout/page to refer
Container	Provides the pointer of this widget's immediate container. Normally, a widget's immediate container is a layout
Active	Set and get the current widget active
Local X	Set and get the current widget local X. Local X refers to the local x-coordinate with reference to its container
Local Y	Set and get the current widget local Y. Local Y refers to the local y-coordinate with reference to its container
Local Width	Set and get the current widget local width. Local width refers to the designed width with reference to its container
Local Height	Set and get the current widget local height. Local height refers to the designed height with reference to its container
Local Rect	Set and get the current widget local dimension with reference to its container
Global X	Set and get the current widget global X. Global X refers to the global x-coordinate with reference to screen

Global Y	Set and get the current widget global Y. Global Y refers to the global y-coordinate with reference to screen
Global Width	Set and get the current widget global width. Global width refers to the designed width with reference to screen
Global Height	Set and get the current widget global height. Global height refers to the designed height with reference to screen
Global Rect	Set and get the current widget global dimension with reference to its screen

Table 32 - Widget Interface Properties

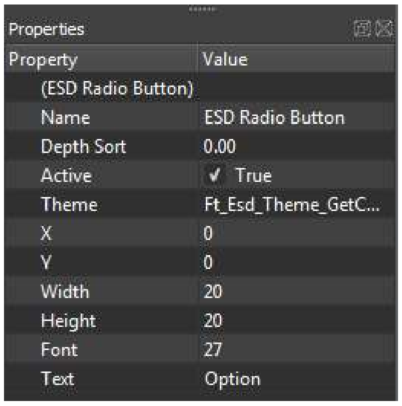
Advanced users are encouraged to refer the example project **"WindowLayout"** for details.

- All Local properties are subjected to difference from the respective Global properties. As the global values are determined by the layers of the layouts as containers.
- Global properties are the values determined at the runtime. While Local properties are values determined at design time.
- Global properties are eventually stable in runtime, as layers of layouts may require multiple rendering cycle to stabilize.

Property Editor

The *Property Editor* provides user a unified interface to edit properties. Each logic node in ESD, including a widget, has its own predefined properties. Users can edit the property values using the Property Editor to customize logic nodes.

The sample picture given below shows an example of the ESD widget "ESD Radio Button".



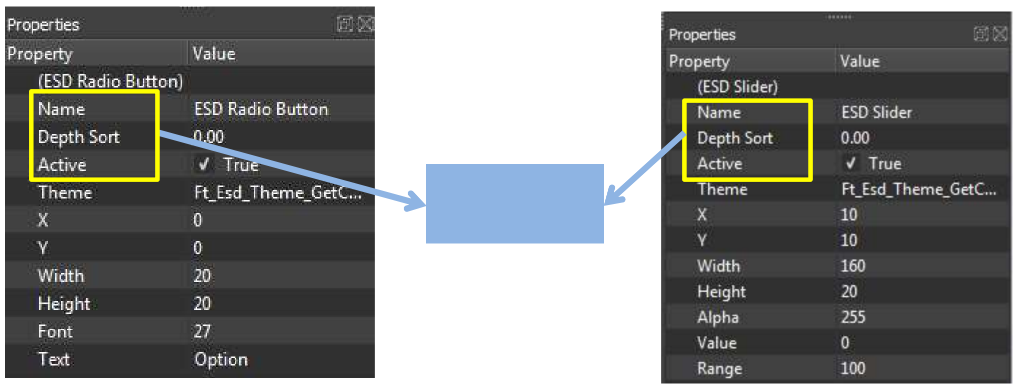
Property	Value
(ESD Radio Button)	
Name	ESD Radio Button
Depth Sort	0.00
Active	✓ True
Theme	Ft_Esd_Theme_GetC...
X	0
Y	0
Width	20
Height	20
Font	27
Text	Option

Figure 110 - Property Editor

Common Properties

While each widget has its own properties defined for its own specific features, the following are some of the properties that are commonly found in all the widgets of ESD application -

- *Name*
- *Depth Sort*
- *Active*



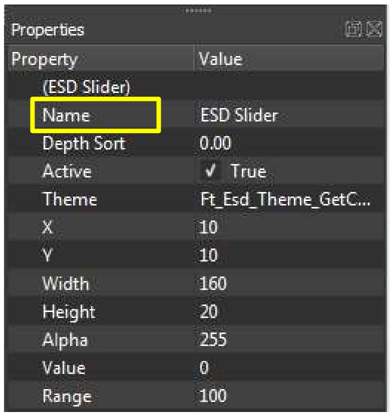
Property	Value
(ESD Radio Button)	
Name	ESD Radio Button
Depth Sort	0.00
Active	✓ True
Theme	Ft_Esd_Theme_GetC...
X	0
Y	0
Width	20
Height	20
Font	27
Text	Option

Property	Value
(ESD Slider)	
Name	ESD Slider
Depth Sort	0.00
Active	✓ True
Theme	Ft_Esd_Theme_GetC...
X	10
Y	10
Width	160
Height	20
Alpha	255
Value	0
Range	100

Figure 111 - ESD Common Properties

Name

Each node must have one unique *name* as an identifier. Users are required to define a name that is valid C identifier, since these names will be used by ESD for generating the source code.



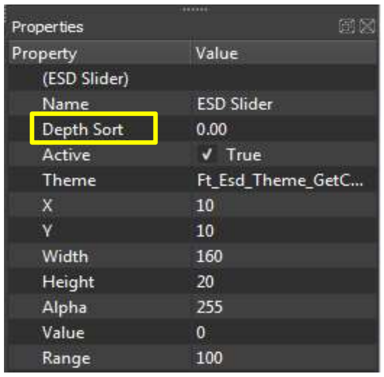
Properties

Property	Value
(ESD Slider)	
Name	ESD Slider
Depth Sort	0.00
Active	✓ True
Theme	Ft_Esd_Theme_GetC...
X	10
Y	10
Width	160
Height	20
Alpha	255
Value	0
Range	100

Figure 112 - Name Property

Depth Sort

Each widget and page have one common property called *Depth Sort*. It is of floating-point type, and defines the sequence of rendering. The default value is "0.00" and users may change it accordingly to adjust the sequence of the rendering process.



Properties

Property	Value
(ESD Slider)	
Name	ESD Slider
Depth Sort	0.00
Active	✓ True
Theme	Ft_Esd_Theme_GetC...
X	10
Y	10
Width	160
Height	20
Alpha	255
Value	0
Range	100

Figure 113 - Depth Sort Property

The context menu helps users adjust the rendering sequence without dealing with the values and can be accessed within the layout editor by selecting the widget or page and right-clicking it.

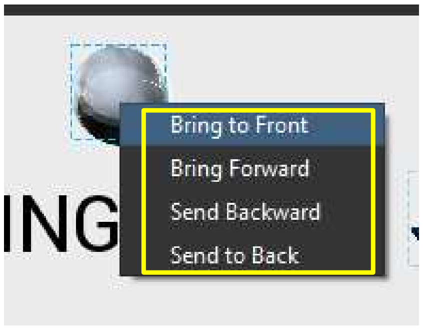
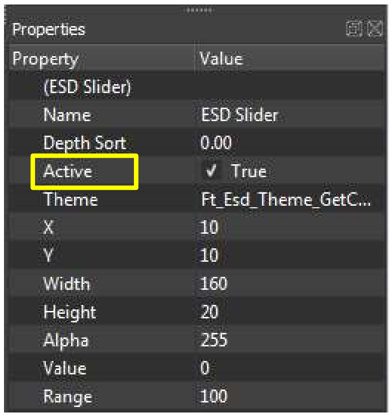


Figure 114 - Right Click Menu

The widget or page with the **greatest** positive "Depth Sort" value will be rendered **last** and appear on the **topmost** layer.

Active

Each widget has one *Active* property of Boolean type which controls whether or not it is rendered on the screen. Active property in false state does not mean that the widget is released from the memory. Instead, it means that the widget is not rendered on the screen, but still is part of the memory.


Properties

Property	Value
(ESD Slider)	
Name	ESD Slider
Depth Sort	0.00
Active	✓ True
Theme	Ft_Esd_Theme_GetC...
X	10
Y	10
Width	160
Height	20
Alpha	255
Value	0
Range	100

Figure 115 - Active Property

Users must ensure that the Name property follows the standard **C identifier** syntax in order to make the generated source code more readable.

Programming Features

This section explains about the programming features. ESD enables users to write C code and make this code available in the form of logic nodes so that the code can be re-used across multiple projects.

Macros

In order to provide the development environment with necessary information about the available functionality in a user's code, ESD provides a set of macros. These macros are necessary and useful when users add their own code that needs to be used from the *logic node editor* or *layout editor*.

These macros are generally used together with the source code. It is recommended that the user defines all the definitions associated with macros immediately following the macros declarations where ever needed and in the same file. Not doing so, may result in unexpected behaviour. For e.g., *ESD_FUNCTION* macro immediately needs to be followed with function definition. Comments are acceptable between the macro and the definitions.

ESD_TYPE

ESD_TYPE provides information about a primitive type. All types are assumed to have a default value of 0.

Syntax Example:

```
ESD_TYPE(ft_uint32_t, Native = UInt32, Edit = Int)
ESD_TYPE(char *, Native = Latin1, Edit = String)
ESD_TYPE(Esd_BitmapCell *, Native = Pointer, Edit = Library)
```

Parameter Type	Parameters	Description
Native		Type used internally for formatting C literals and binary representations
	Void	Not a type (default)
	Int64, Int32, Int16, Int 8,IntPtr	Signed integer
	UInt64, UInt32, UInt16, UInt8, UIntPtr	Unsigned integer
	Double	64bit floating point
	Float	32bit floating point
	Char	An 8bit text character
	Bool	C99 Bool
	Utf8	String with UTF-8 encoding
	Latin1	String with Latin 1 (ISO8859-1) encoding
	Pointer	Pointer type
	Struct	Struct type (not fully supported yet)
Edit		Used by the Properties Editor to specify the control to edit the value and by code generation when using the fixed-point formats
	None	No editor control (default)
	Int	Integer, spin-box integer control
	Real	Floating point, spin-box floating point control
	Boolean	One or zero, checkbox
	String	Text, single-line entry
	Fixed1, Fixed2, …, Fixed32	Fixed point, spin-box floating point control
	ColorARGB	Color, rgb with alpha
	ColorRGB	Color, rgb without alpha
	Library	Select a function specified with ESD_FUNCTION to provide the value (useful for pointer values)

Table 33 - Macros - ESD_TYPE - Parameters

ESD_ENUM

Enumeration types are usable as types from the logic editor. It will use the specified Type in the generated code. Internally, only the enumeration identifiers are parsed; values are ignored. Enumerations used in the properties are serialized as their identifier string. Enumerations are edited using a drop-down control in the *Properties Editor*.

Syntax Example:

Using pre-processor definitions, the symbol right after each *#define* statement is used.

```
ESD_ENUM(Ft_BitmapFormat, Type = ft_uint32_t)
#define PALETTED             8UL
#define PALETTED4444         15UL
#define PALETTED565          14UL
#define PALETTED8            16UL
ESD_END()
```

Parsing from an enum requires the identifiers to be on separate lines, the first identifier of each line is used.

```
ESD_ENUM(Ft_BitmapFormat)
enum Ft_BitmapFormat
{
    PALETTED,
    PALETTED565,
    PALETTED8
};
ESD_END()
```

In case the enumeration is defined in another file, a macro is available to manually specify the identifiers.

```
ESD_ENUM(Ft_BitmapFormat, Type = ft_uint32_t)
ESD_IDENTIFIER(PALETTED)
ESD_IDENTIFIER(PALETTED565)
ESD_END()
```

When the Flag value is specified, the enumeration can combine multiple values, and will show as a series of check-boxes in the Properties Editor instead of a drop-down menu.

```
ESD_ENUM(Ft_CoPro_Opt, Type = ft_uint16_t, Flags)
#define OPT_CENTERX          512UL
ESD_END()
```

To allow users to select 0 as a valid value, the Zero flag can be set. This flag has no effect when the Flag value is specified.

ESD_PARAMETER

ESD_PARAMETER is a macro designed for the registration of user-defined parameters within ESD. Whenever a user creates an ESD_FUNCTION or ESD_METHOD, they can establish the parameter by utilizing ESD_PARAMETER.

Syntax Example:

```
ESD_PARAMETER(value, Type = Int)
ESD_PARAMETER(test, Type = ft_bool_t)
ESD_PARAMETER(color, Type = ft_argb32_t)
```

ESD_FUNCTION

ESD_FUNCTION is a macro to register a user defined function to ESD, so that the application can add that function into the *User Functions* category node which is located in the Library browser. *ESD_FUNCTION* is not applicable for static functions even though ESD permits to make any function with *ESD_FUNCTION* macro.

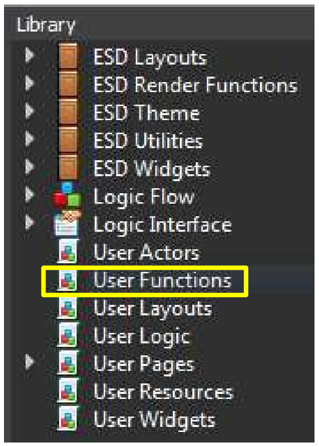


Figure 116 - User Functions Example

Syntax Example:

The following syntax example is to register a function called *"hsvToRgb"* to ESD.

```
ESD_FUNCTION(hsvToRgb, Type = ft_argb32_t )
ESD_PARAMETER(h, Type = double)
ESD_PARAMETER(s, Type = double)
ESD_PARAMETER(v, Type = double)
ft_argb32_t hsvToRgb(double h, double s, double v)
```

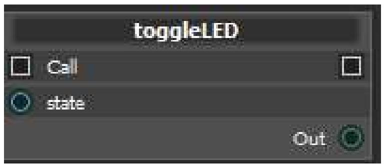
Upon registering the function, it is added to the *User Functions* category.

Parameters	Description
Type	The function's return value type

Table 34 - Macros - ESD_FUNCTION - Parameters

NOTE

- Each ESD_PARAMETER defines only one function parameter. It must be kept in a single line for each parameter.
- "Buffered" flag can be attached after "Type"**.** It will create the output of the function stored in a member variable, whenever the function is called. Without the Buffered flag, you'll have a function with just one output value which is called whenever the output value is used. Here is an example:



```
ESD_FUNCTION(toggleLED, Type=int,Buffered)
ESD_PARAMETER(state,Type=ft_bool_t)
```

ESD_METHOD

ESD_METHOD works similar to *ESD_FUNCTION* with an enhanced feature. It takes a logic/actor/widget/page/app context pointer as first parameter. It can be used only within the logic editor of that logic/actor/widget/page/app.

Syntax Example:

The following syntax example defines one method to handle "Pushed" event for an ESD Push Button widget contained in "MainPage.page".

```
ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context = MainPage)
void MainPage_ESD_Push_Button_Pushed(MainPage *context)
{
        // ...
}
```

The logic node connection and the corresponding output is shown in the sample picture below –

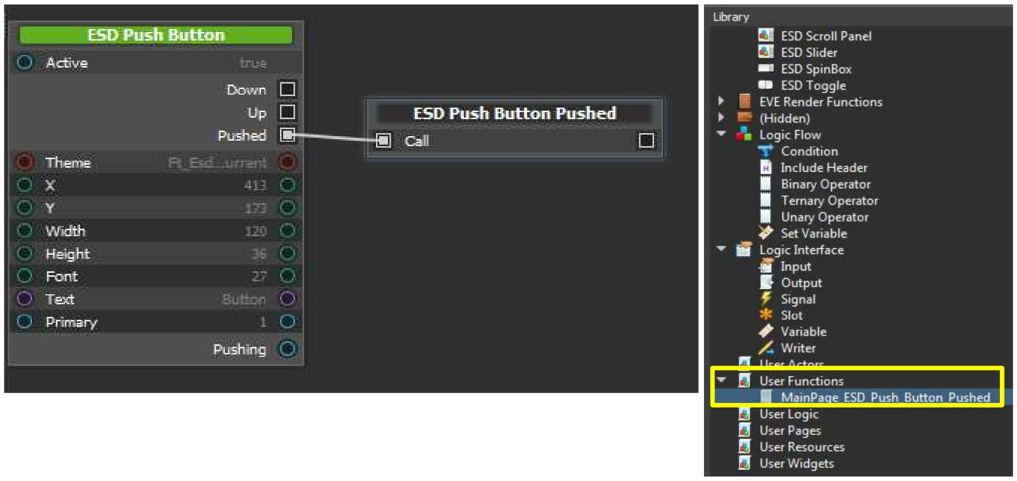


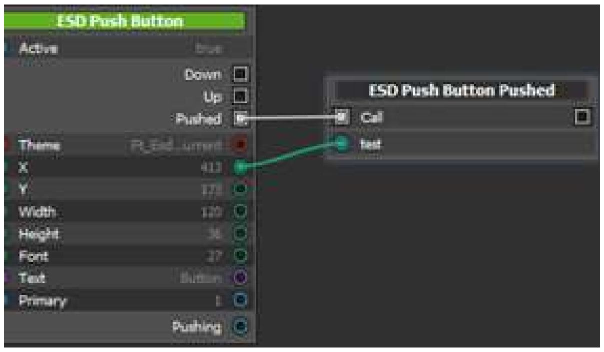
Figure 117 - ESD_METHOD Example

Parameters	Description
Context	Pointing to the caller's context

Table 35 - Meta Macros - ESD_METHOD - Parameters

NOTE

- Additional parameters can be defined similar to ESD_FUNCTION. But these parameters must be added only after the Context parameter. Here is an example:



```
ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context =
MainPage)
ESD_PARAMETER(test,Type = ft_bool_t)
void MainPage_ESD_Push_Button_Pushed(MainPage
*context,ft_bool_t test)
{
    // ...
}
```

NOTE

- If the return value of method function is to be defined, "Buffered" keyword needs to be added after the "Type" definition. Refer to the syntax example -

```
ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context =
MainPage,Type=int,Buffered)
ESD_PARAMETER(test,Type = ft_bool_t)
int MainPage_ESD_Push_Button_Pushed(MainPage
*context,ft_bool_t test)
{
    // ...
}
```

ESD_VARIABLE

ESD_VARIABLE is a built-in macro used to define a local variable of a logic node.

Syntax Example:

```
ESD_VARIABLE(Margin, Type = ft_int16_t, Default = 0, Public)
ft_int16_t Margin;
```

ESD_GLOBAL

ESD_GLOBAL is a built-in macro used to define a global variable accessed across different pages of a logic node. Global variable declaration should present in the header file and definition should be in source file.

Syntax Example:

```
In header file:
ESD_GLOBAL(Var, Type = int)      // type must be specified
Extern int Var;

In C file:
Int Var = 0xFFFF;    // Assign initial value
```

ESD_INPUT

ESD_INPUT is a built-in macro used to define an input variable of a logic node.

Syntax Example:

```
ESD_INPUT(Margin, Type = int)
int(* Margin)(void *context);
```

ESD_OUTPUT

ESD_OUTPUT is a built-in macro used to define an output variable of a logic node.

Syntax Example:

```
ESD_OUTPUT(FirstPos, Type = int)
int LayoutSplit_FirstPos(LayoutSplit *context);
```

ESD_UPDATE

ESD_UPDATE macro is used to register a function that is called repeatedly by built-in "Update" slot. This function shall not have any return value.

Syntax Example:

```
ESD_UPDATE(Ft_Esd_BitmapPersist)
ESD_PARAMETER(bitmapCell, Type = Ft_Esd_BitmapCell *)
void Ft_Esd_BitmapPersist(Ft_Esd_BitmapCell *bitmapCell);
```

Parameters:

Users can define their own parameter used by the registered function by using the ESD_PARAMETER.

Pre-compiler options

Within the ESD C code simulation engine, the following macros are predefined in generated source code:

- *ESD_SIMULATION*

ESD_SIMULATION

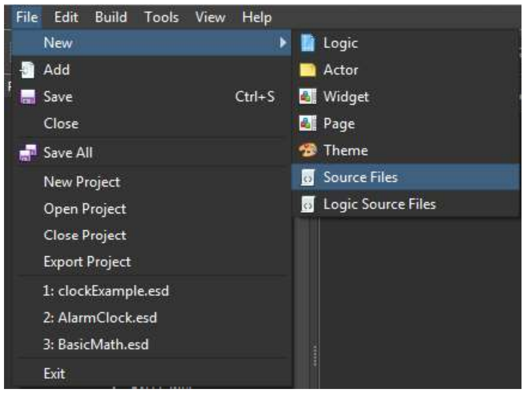
ESD_SIMULATION macro is defined when the code is running through the ESD simulation engine. If some part of the user's C source code is not supposed to be run under the ESD simulation engine, for example, accessing some hardware specific registers, users may choose to skip the code by using this macro.

Add User Functions

Users can define their own functions by writing the C source code directly. If these functions are defined by using the predefined "ESD_FUNCTION", they will be shown under the "User Functions" category in the library browser. Users can drag and drop these nodes into the logic node editor to use them.

Creating Source File

To add user functions, users need to create a C source file by clicking **File → New → Source Files**.



Upon adding the source file, include the following header **(mandatory)** in the source file.

```
#include "Ft_Esd.h"
```

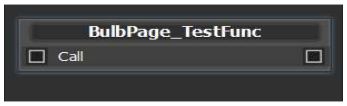
Editing the Source File

To make ESD responsive of the user functions, the ESD specific macro *"ESD_FUNCTION or ESD_METHOD"* needs to be added before the function definition. Refer to the examples given below –

Syntax Example:

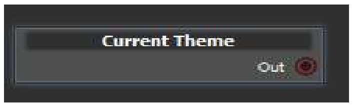
- Without parameter input and return value

```
ESD_FUNCTION(BulbPage_TestFunc)
void BulbPage_TestFunc(){}
```



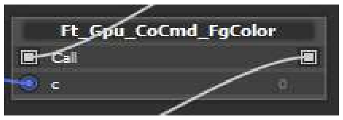
- Without parameter input, but with the return value

```
ESD_FUNCTION(Ft_Esd_Theme_GetCurrent, Type = Ft_Esd_Theme *)
Ft_Esd_Theme *Ft_Esd_Theme_GetCurrent();
```



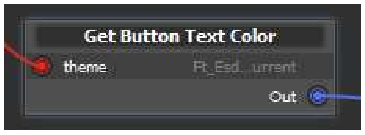
- Without return value defined, but with the parameter input

```
ESD_FUNCTION(Ft_Esd_Theme_GetButtonTextColor, Type = ft_rgb32_t)
ESD_PARAMETER(theme, Type = Ft_Esd_Theme *)
ft_rgb32_t Ft_Esd_Theme_GetButtonTextColor(Ft_Esd_Theme *theme) { //.. }
```



- Without return value and parameter input

```
ESD_FUNCTION(Ft_Esd_Theme_GetButtonTextColor, Type = ft_rgb32_t)
ESD_PARAMETER(theme, Type = Ft_Esd_Theme *)
ft_rgb32_t Ft_Esd_Theme_GetButtonTextColor(Ft_Esd_Theme *theme)
{
    //..
}
```



F. Appendix A – List of Figures

G. Appendix B – List of Tables

H. Appendix C – Revision History

Document Title	:	BRT_AN_029 EVE Screen Designer 4.19 User Guide
Document Reference No.	:	BRT_000218
Clearance No.	:	BRT#159
Product Page	:	https://brtchip.com/eve-toolchains
Document Feedback	:	Send Feedback

Revision	Changes	Date
Version Draft 0.3	Initial user guide release for ESD 4.5 version features & enhancements	22-06-2018
Version Draft 0.6	Updated user guide as per ESD 4.8 version features and enhancements	27-09-2019
Version 1.0	User Guide updated as per the following changes for 4.10 – 1. Overview/What's new in ESD 4.10 2. ESD Workflow/Target Selection 3 .ESD Workflow/Build & Upload 4. ESD Workflow/Export	03-12-2020
Version 1.1	User Guide updated as per the following changes for 4.12 – 1. Overview/What's new in ESD 4.12 2. ESD Layouts: add Signal Switch page layout guide 3. Add a section on video converter 4. Other minor improvements	12-04-2021
Version 1.2	User Guide updated as per the following changes for 4.13– 1. Overview/What's new in ESD 4.13 2. Modify Hardware Requirements session to add Pico 3. Modify the outdated figures such as menu bar, installation wizard 4. Add new Tools menu description in Table 2 5. Add Pico export description in Table 8 6. Added Gameduino 3X dazzler support 7. Added Advanced User Setting Section	30-07-2021
Version 1.3	User Guide updated as per the following changes for 4.14 – 1. Add play animation	06-10-2021
Version 1.4	User Guide updated as per the following changes for 4.15 – 1. Overview/What's new in ESD 4.15 2. Introduction for welcome screen 3. Add platform support list for IDM2040	11-02-2022
Version 1.5	User Guide updated as per the following changes for 4.16 – 1. Overview/What's new in ESD 4.16 2. Add memory management explanation	06-04-2023
Version 1.6	User Guide updated as per the following changes for 4.17 – 1. Overview/What's new in ESD 4.17 2. Updated Known Issues & Limitations for the list of features not supported by the C code editor in ESD	01-11-2023

Version 1.7	User Guide updated as per the following changes for 4.18 – 1. Overview/What's new in ESD 4.18 2. Added GNU ARM toolchain for Rapsperry Pi Pico project building 3. Revised section - "Support DXT1 format bitmap"	06-03-2024
Version 1.8	User Guide updated as per ESD4.19 version features and enhancements	29-07-2024