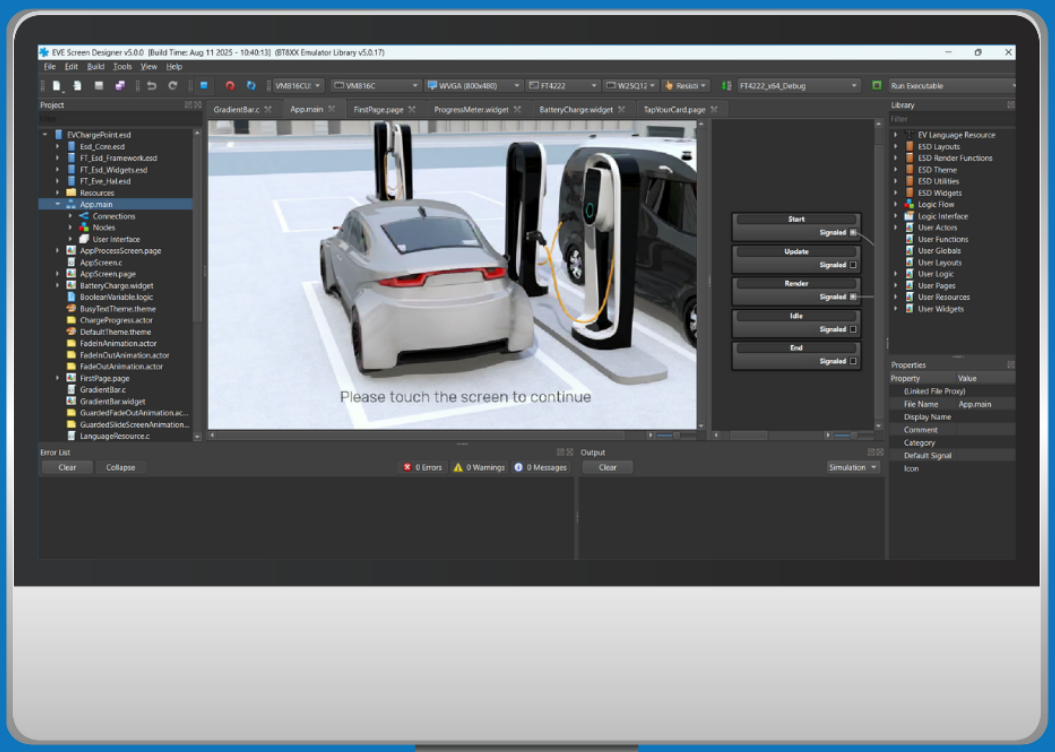
Bridgetek
BRIDGING TECHNOLOGY

EVE SCREEN DESIGNER 5.3.0

USER GUIDE



DOC. VER. 1.15

ISSUE DATE: 21 September 2026

NEITHER THE WHOLE NOR ANY PART OF THE INFORMATION CONTAINED IN, OR THE PRODUCT DESCRIBED IN THIS MANUAL, MAY BE ADAPTED, OR REPRODUCED IN ANY MATERIAL OR ELECTRONIC FORM WITHOUT THE PRIOR WRITTEN CONSENT OF THE COPYRIGHT HOLDER. THIS PRODUCT AND ITS DOCUMENTATION ARE SUPPLIED ON AN AS-IS BASIS AND NO WARRANTY AS TO THEIR SUITABILITY FOR ANY PARTICULAR PURPOSE IS EITHER MADE OR IMPLIED. BRIDGETEK PTE LTD WILL NOT ACCEPT ANY CLAIM FOR DAMAGES HOWSOEVER ARISING AS A RESULT OF USE OR FAILURE OF THIS PRODUCT MIGHT REASONABLY BE EXPECTED TO RESULT IN PERSONAL INJURY. THIS DOCUMENT PROVIDES PRELIMINARY INFORMATION THAT MAY BE SUBJECTED TO CHANGE WITHOUT NOTICE. NO FREEDOM TO USE PATENTS OR OTHER INTELLECTUAL PROPERTY RIGHTS IS IMPLIED BY THE PUBLICATION OF THIS DOCUMENT. BRIDGETEK PTE LTD, 1 TAISENG AVENUE, TOWER A, #03-05, SINGAPORE 536464. SINGAPORE REGISTERED COMPANY NUMBER 201542387H. © BRIDGETEK PTE LTD.

Contents

A. Preface 6
- Purpose 6
- Intended Audience 6
- Document Reference 6
- Feedback 6

B. Overview 7
- Introduction 7
- Key Features 7
- What's new in ESD 5.3.0 8
- Known Issues & Limitations 8
- Terms & Abbreviations 9
- Credits 10

C. Setup & Installation 10
- System Requirements 10
- Hardware Requirements 11
- Dependencies / Pre-Requisites 15
- Installing ESD Package 17
- Installation Folder 21

D. Working with ESD 22
- Major features 22
  - Support full BT81X features 22
  - Support for ASTC bitmap format 23
  - Support DXT1 bitmap format 24
  - Support extended font format 26
  - ASTC Encoder 28
  - Video Converter 28
  - Dynamic Bitmap Loading 30
  - Support Animation 30
  - Reconvert the resource file 31
  - Generate Flash file and write Flash file to device 32
  - LittleFS File System with EVE connect flash memory 33
  - Restructured EVE HAL Library 36
  - Memory management 36
  - UI enhancements 37
  - BT82x ESD Project Configuration 37
- Setting Platform Specific Properties 39
  - Migrating existing projects to new platform 39
- Advanced User Settings 40
  - Application Settings 40
  - Troubleshooting 43

E. Getting Started 44
- The Welcome Dialog 44
- The Graphical User Interface 46
  - Menu bar 47
  - Toolbar 49
  - Touch IC configuration 51
  - Project Explorer 51
  - Screen Layout Editor 52
  - Logic Node Editor 54

Copyright © Bridgetek Pte Ltd

Library Browser 54
Error List 55
Output 56
Property Editor 56
Status bar 57
Application Project Structure 59
Application Workflow 60
Design Layout using Layout Editor 61
Custom Widget 61
Add Bitmap Resource 61
Configure Bitmap Resource 64
Design Screen Logic using Logic Node Editor 65
Evaluate and choose the memory usage of the project 66
Simulation 66
Target selection 67
Build & Upload 69
Export 70
Exported Folder Structure 72
Assets 73
Bitmap Resources 73
SD card 73
EVE SD card support on the VM820C platform 74
Flash 75
Resource Metadata 75
Layout Editor 78
Page File 78
"Active" Property 78
Switch Page 79
Page Persistence 80
User Defined Function for Page File 80
Zoom In & Out 81
Main File 82
Logic File 82
C File 83
Logic Note Editor 83
Basic Logic Node 83
Composite Logic Node 84
Page Node 85
Widget Node 85
Layout Type Widget 87
Actor Node 88
Logic Object 89
Connections 90
Rendering Order 90
Logic Note Editor – User Interaction 91
Library Browser 92
ESD Layouts 94
Utilities 95
Switch Page 100
Dynamic Switch Page 101
Auto Switch Widget 103
Fixed Positioning 104
Fill Layout 105
Linear Layout 106
Stretch 108
Scroll Layout 109
Scroll Switch Page Layout 110

Copyright © Bridgetek Pte Ltd

Signal Switch Page Layout *112*
ESD Render Functions **113**
Primitives *114*
ESD Bitmap *115*
ESD Line *116*
ESD Rectangle *117*
Display List *117*
ESD Theme *118*
Built In Themes *118*
ESD Utilities **120**
ESD Idle Checker **121**
Logic Flow **121**
Condition **122**
Binary Condition **122**
Local Method **123**
Binary Operator **124**
Ternary Operator **125**
Unary Operator **125**
Sequence **126**
Switch Node **126**
Switch Value **127**
Set Variable **127**
Watch Variables **128**
Logic Interface **128**
Input **129**
Output **129**
Signal **130**
Slot **130**
Variable **131**
Writer **132**
Widget Interface **132**
Property Editor **133**
Common Properties **134**
Name **134**
Depth Sort **135**
Active **135**
Programming Features **136**
Macros **136**
ESD_TYPE **136**
ESD_ENUM **137**
ESD_PARAMETER **138**
ESD_FUNCTION **138**
ESD_METHOD **139**
ESD_VARIABLE **140**
ESD_GLOBAL **141**
ESD_INPUT **141**
ESD_OUTPUT **141**
ESD_UPDATE **141**
Pre-compiler options **142**
ESD_SIMULATION **142**
Add User Functions **142**
Creating Source File **142**
Editing the Source File **143**
F. Appendix A – List of Figures **144**

Copyright © Bridgetek Pte Ltd

G. Appendix B – List of Tables.....148
H. Appendix C – Revision History 149

Copyright © Bridgetek Pte Ltd

A. Preface

Purpose

This document outlines the functionality and procedures for using the **E**VE **S**creen **D**esigner (ESD).

Intended Audience

This document is intended audience for GUI application developers working with EVE products.

Document Reference

Document Name	Document Type	Document Format
FT81x-BT88x Programming Guide	Programming Guide	PDF
FT81x Datasheet	Datasheet	PDF
FT800/FT801 Programmer Guide	Programming Guide	PDF
FT9xx Toolchain Installation Guide	Installation Guide	PDF
BT81X Series Programming Guide	Programming Guide	PDF
BT815/6 Datasheet	Datasheet	PDF
BT817/8 Datasheet	Datasheet	PDF
BT82X Series Programming Guide	Programming Guide	PDF
BT820 Datasheet	Datasheet	PDF
EVE Screen Designer Widget Guide	User Reference specific for widgets (Built-in with ESD)	PDF
BT82X Firmware Update and Extension Guide	Programming Guide	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/.

Copyright © Bridgetek Pte Ltd

B. Overview

Introduction

EVE **S**creen **D**esigner (ESD) is a next-generation smart IDE for EVE that simplifies the development of EVE-based GUI applications. It enables users to create GUI applications through a **visual programming**[1] approach, eliminating the need to understand EVE-specific display list commands.

ESD offers 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 logic. It automatically generates **ANSI C code** for the target hardware platform. Users can also simulate their designs to preview the user interface (UI) before compiling and deploying the generated source code. Furthermore, ESD supports seamless development and deployment across multiple host platforms, with full integration for the following host platforms:

1. **PC** via **FT4222** (USB to SPI/QSPI)**.**
2. **PC** via **MPSSE** (USB to SPI).
3. **FT9XX**.
4. **Raspberry Pi Pico**[2] (RP2040).
5. **STM32F4**.

This integration enables users to compile, link, and upload generated source code directly within the ESD environment, streamlining development workflows across a wide range of embedded systems.

Key Features

The key features of **E**VE **S**creen **D**esigner include"

- **WYSIWYG** GUI design environment
- High-level widgets
- No requirement for EVE display list knowledge
- Widget-based GUI construction
- Drag-and-drop interface for creating screen layouts
- Inter-widget communication
- Screen logic creation without coding
- Simulation of screen logic and user touch input using mouse
- Ability to build and download generated "C" code (if FT9XX/STM32CubeIDE/Pico Toolchain installed).

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

Copyright © Bridgetek Pte Ltd

What's new in ESD 5.3.0

New Features & Improvements:

- ESD now reuses valid converted resources after a build-target switch and reconverts only missing, outdated, or incompatible resources.
- ESD now includes picotool for offline Pico SDK 2.0.0 or later project creation without rebuilding picotool for every project.
- Updated IDK-BT817-101A support for the DP-1011-02A LCD panel with an Ilitek touch controller.
- Updated the bundled example projects with the latest EVE library changes.
- Added a new memory allocator for BT820-based projects to improve memory management for the larger BT820 memory space, including allocation, garbage collection, and memory defragmentation.
- Add scanout format for BT820 project properties setting.
- Added support for accessing SD card resources in the ESD Asset Browser example project.
- Added simulation support for Array-type inputs
- Optimized STM32F4 platform support in ESD by retaining only publicly available STM32 Nucleo development boards and removing support for Bridgetek internal development boards (STM32F401CCU6).

Resolved Issues:

- Removed duplicate Get Delta Ms, Get EVE Host, and Get Milliseconds utilities from the ESD Library.
- Resolved STM32 exported-project build failures with GNU 14.3.Rel1 and STM32CubeIDE 2.1.1.
- Corrected .esdm output paths generated by project.
- Fixed the issue preventing video playback from working on the EVE SD card.

Known Issues & Limitations

The following are some known issues and limitations of ESD:

- On FT80x, RAM_PAL is limited to storing a single PALETTED bitmap resource due to hardware limitations.

- 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.

  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.

Copyright © Bridgetek Pte Ltd

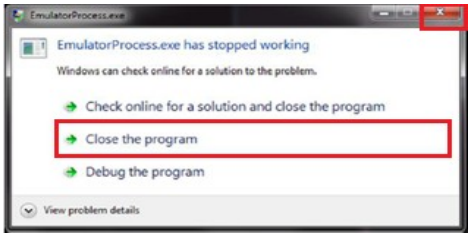


Figure 1 - EmulatorProcess.exe error message

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

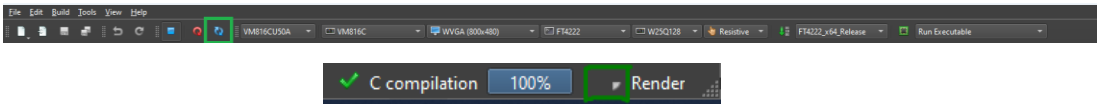


Figure 2 - Frozen simulation rotation icon

- EVE-connected flash image download is not supported for BT820-based modules (e.g., VM820C, VM820B) when used with STM32F401 or MM2040EV host platforms. In such cases, the flash image must be programmed using Eve Asset Builder.

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
Layout Editor	The interface for previewing a single page or the entire project
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

Table 1 - Terms and Abbreviations Description

Copyright © Bridgetek Pte Ltd

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 the ESD application, ensure that your system meets the following recommended requirements:

- ✓ Windows 11 (preferred); or Windows 10 with latest updates
- ✓ 1.6 GHz or faster processor
- ✓ Minimum 1 GB RAM (1.5G B if running on a virtual machine)
- ✓ Multi-Core CPU (highly recommended)
- ✓ At least 512 MB of available hard disk space
- ✓ Display resolution of 2048 x 2048 pixels or higher
- ✓ "Write" permission for the installation folder

Copyright © Bridgetek Pte Ltd

Hardware Requirements

The C source code exported from ESD is intended for **EVE-based** platforms, which typically include an EVE module paired with a host, such as an MCU or a PC-based system.

The supported platforms are listed below:

[Build Target]:

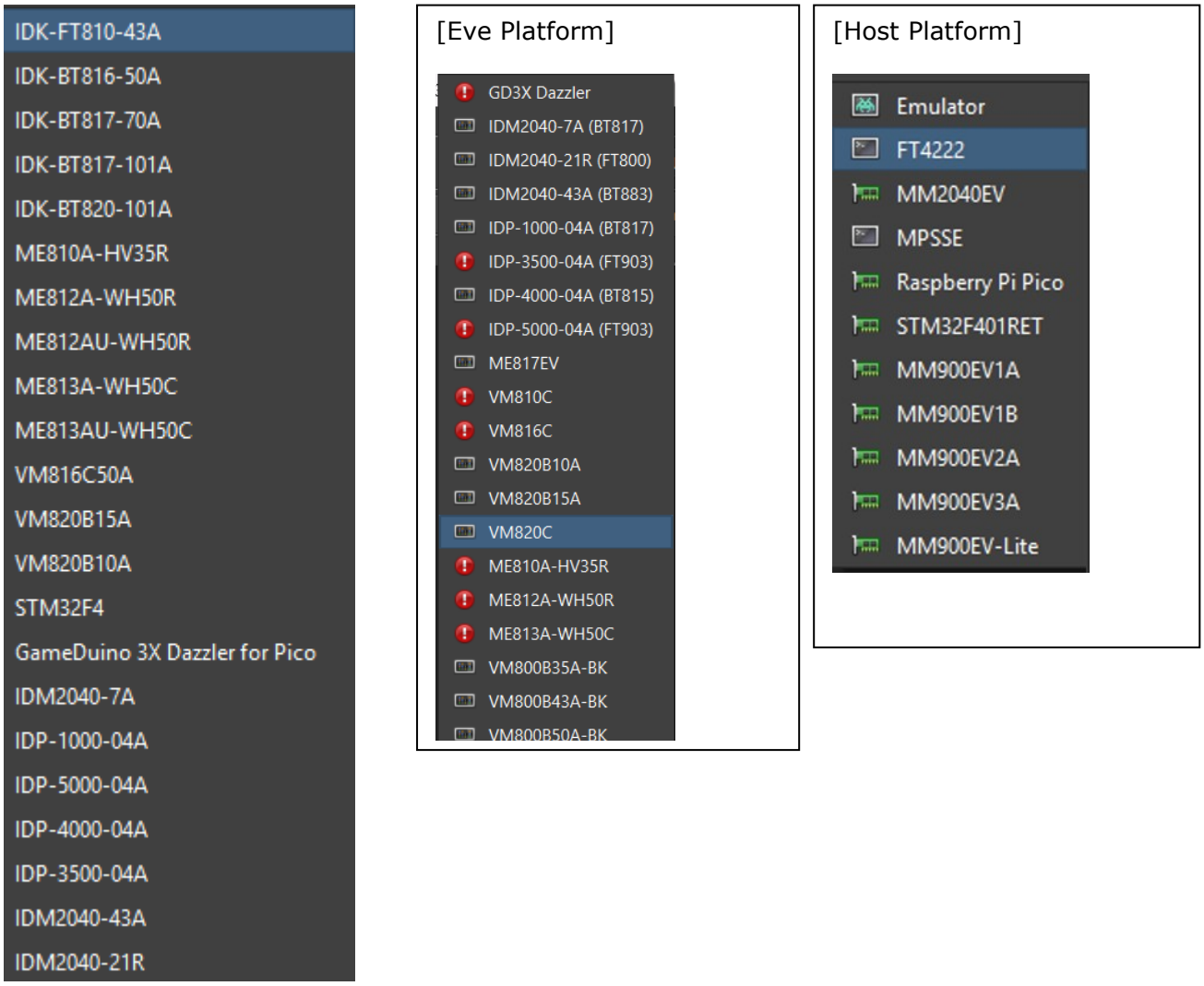


Figure 3 - ESD support platform

Note: The exclamation mark  indicates that this option is not compatible with the target hardware.

Copyright © Bridgetek Pte Ltd

Platform Name (Build Target)	Compatible EVE Module	Compatible MCU Module Name	Compatible PC based Module
IDK-FT810-43A[13]	VM810C50A[6] (480x272)	N.A.	FT4222H[8] (libFT4222.dll)
IDK-BT816-50A[14]	VM816CU[4] (800x480)	N.A.	FT4222H[8] (libFT4222.dll)
IDK-BT817-70A[15]	ME817EV[5] (800x480)	MM900EV[1] MM2040EV[2] STM32F4[11]	FT4222H[8] (libFT4222.dll)
IDK-BT817-101A[16]	ME817EV[5] (1280X800)	MM900EV[1] MM2040EV[2] STM32F4[11]	FT4222H[8] (libFT4222.dll)
IDK-BT820-101A[17]	VM820C[12] (1920x1200)	MM2040EV[2] STM32F4[11]	FT4222H[8] (libFT4222.dll)
ME810A-HV35R	ME810A-HV35R[6](320X480)	MM900EV[1] MM2040EV[2]	N.A.
ME812A-WH50R	ME812A-WH50R[6] (800x480)		N.A.
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)
VM816C50A	VM816C[4] (800x480)	N.A.	MPSSE [7] (libMPSSE.dll)
VM820B15A	VM820B (1920x1080)	N.A.	FT4222H[8] (libFT4222.dll)
VM820B10A	VM820B (1280x800)	N.A.	FT4222H[8] (libFT4222.dll)
STM32F4[11]	ME817EV[5] (1024X600) ME817EV[5] (1280X800)	STM32	N.A.
Gameduino 3X Dazzler for Pico	Gameduino 3X Dazzler[10] (HDMI output 1280x720)	Raspberry Pi Pico[3]	N.A.
IDM2040-7A	IDM2040-7A[9] (800x480)	Raspberry Pi Pico[3]	N.A.
IDP-1000-04A	IDP-1000-04A (1280x800)		N.A.
IDP-5000-04A	IDP-5000-04A (800x480)	MM900EV[1]	N.A.
IDP-4000-04A	IDP-4000-04A (480x480)	Raspberry Pi Pico[3]	N.A.
IDP-3500-04A	IDP-3500-04A (320x480)	MM900EV[1]	
IDM2040-43A	IDM2040-43A[9] (480x272)	Raspberry Pi Pico[3]	N.A.
IDM2040-21R	IDM2040-21R[9] (480x480)		N.A.

Table 2 - Platform vs Module Compatibility

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/

Copyright © Bridgetek Pte Ltd

7. MPSSE stands for "multi-protocol synchronous serial engine". The following Bridgetek or FTDI devices are supported:

   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/

   The MM4222-QSPI board includes an onboard FT4222H chip. Additionally, the MM4222-QSPI board includes multiple expansion sockets compatible with different MCU controller board. See: https://brtchip.com/product/mm4222-qspi/

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. EVE as GPU:
      - IDM2040-7A is based on BT817: BT817/8 Datasheet (brtchip.com).
      - IDM2040-43A is based on BT883: BT88X Datasheet (brtchip.com).
      - IDM2040-21R is based on FT800: FT800 Datasheet (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

11. The STM32F4 platform refers to the STM32F401RET-based platform supported by Bridgetek for EVE application development.

    The platform can use the STM32 Nucleo-64 (NUCLEO-F401RE) development board, connected to a supported Bridgetek EVE module, such as the ME817EV or VM820, via Bridgetek MM4222-QSPI adapter board. It should be noted that the STM32 Nucleo-64 board (NUCLEO-F401RE) does not include an on-board SD card. Therefore, applications that include resources with the **Location** property set to **ESD_RESOURCE_FILE** cannot run on this platform.

Please refer to the table for the pin connections between the STM32F4 MCU and the EVE module.

MCU Pin	EVE Signal	Description
PA1	PD#	Power Down Pin
PA4	CS1	SPI1 Chip Select
PA5	SCK	SPI1 Clock
PA6	MISO	SPI1 MISO
PA7	MOSI	SPI1 MOSI

Table 3 - Connection between STM32F401RET6 and EVE

Copyright © Bridgetek Pte Ltd

Please refer to the tables for the pin connections between the MCU and the SD card

MCU Pin	Signal	Description
PB12	CS1	SPI2 Chip Select
PB13	SCK	SPI2 Clock
PB14	MISO	SPI2 MISO
PB15	MOSI	SPI2 MOSI

Table 4 - Connection between STM32F401RET6 and SD CARD



To ensure proper operation of the hardware platform (FT90X modules, Raspberry Pi Pico, or STM32F4), copy the required resources to an SD card and insert it when EVE_RESOURCE_FILE is enabled. The application will use the SD card as external storage for EVE resources. If the SD card is not present, resource loading will fail, and the application may not function correctly.

12. VM820C is the EVE 5 development module: https://brtchip.com/wp-content/uploads/2025/07/DS_VM820C.pdf

13. IDK-FT810-43A is an FT810-based development kit with a 4.3-inch TFT display (480 × 272) and resistive touch support: https://brtchip.com/product/idk-ft810-43a/

14. IDK-BT816-50A is a BT816-based development kit with a 5.0-inch TFT display (800 × 480) and resistive touch support: https://brtchip.com/product/idk-bt816-50a/

15. IDK-BT817-70A is a BT817-based development kit with a 7.0-inch TFT display (800 × 480) and capacitive touch support: https://brtchip.com/product/idk-bt817-70a/

16. IDK-BT817-101A is a BT817-based development kit with a 10.1-inch TFT display (1280 × 800) and uses the DP-1011-02A LCD panel with an Ilitek capacitive touch controller: https://brtchip.com/product/idk-bt817-101a/

17. IDK-BT820-101A is a BT820-based development kit with a 10.1-inch TFT display (1920 × 1200) and capacitive touch support: https://brtchip.com/product/idk-bt820-101a/

Copyright © Bridgetek Pte Ltd

Dependencies / Pre-Requisites

- **Visual C++ Redistributable for Visual Studio 2022**

  If the PC does not have Microsoft Visual Studio 2022 installed, Visual C++ Redistributable is required. Users can download this from: https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170#latest-supported-redistributable-version

- **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:

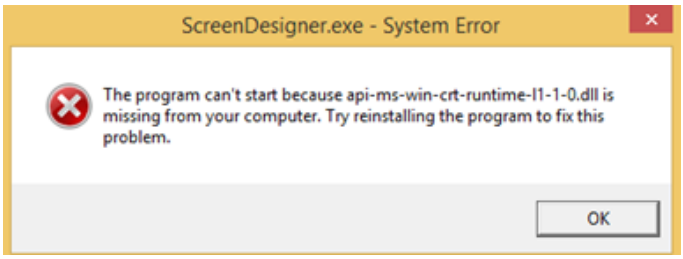


Figure 4 - Screen Designer - System Error

- **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/

  ESD 5.3.0 includes a prebuilt picotool, enabling offline creation of Pico SDK 2.0.0 or later projects without downloading or rebuilding picotool for each project.

Copyright © Bridgetek Pte Ltd

- **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:

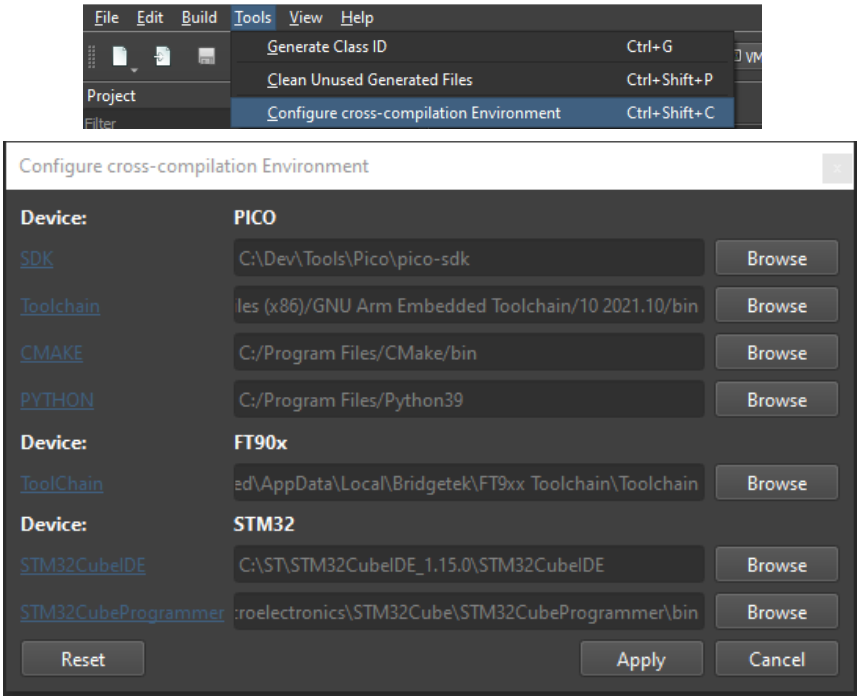


Figure 5 - Configure cross-compilation Environment

Please note that the expected version of STM32CubeIDE is 1.15.0 or higher, and the expected version of STM32CubeProgrammer is 2.15.0 or higher.

Copyright © Bridgetek Pte Ltd

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]**.

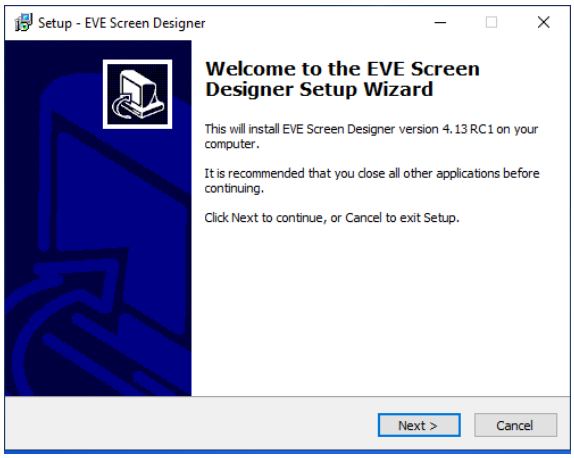


Figure 6 - EVE Screen Designer welcome page

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

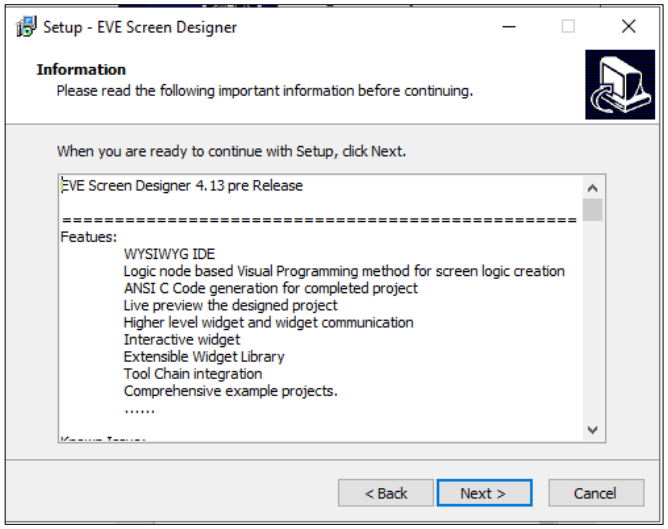


Figure 7 - EVE Screen Designer information page

Copyright © Bridgetek Pte Ltd

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]**.

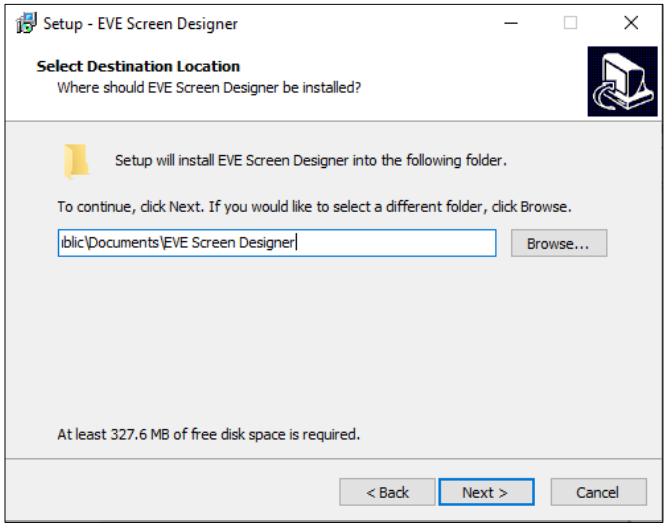


Figure 8 - EVE Screen Designer select destination location page

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.

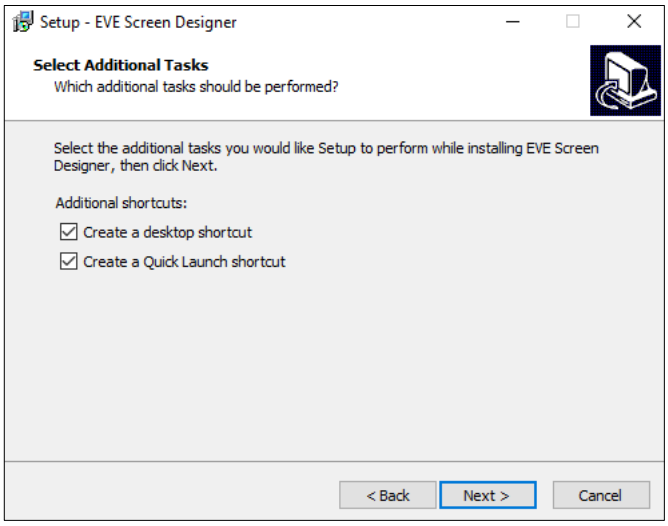


Figure 9 - EVE Screen Designer select additional tasks page

Copyright © Bridgetek Pte Ltd

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

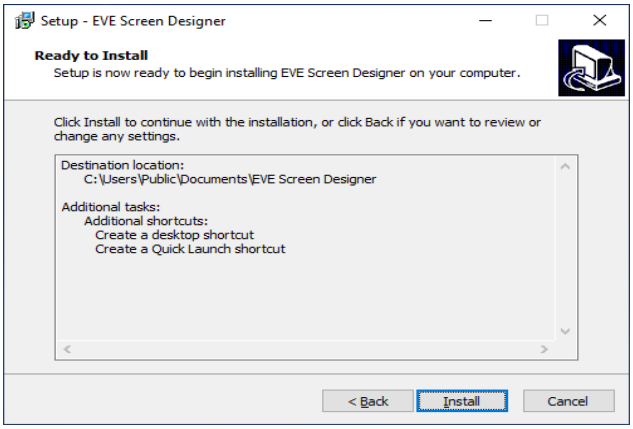


Figure 10 - EVE Screen Designer start installation

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

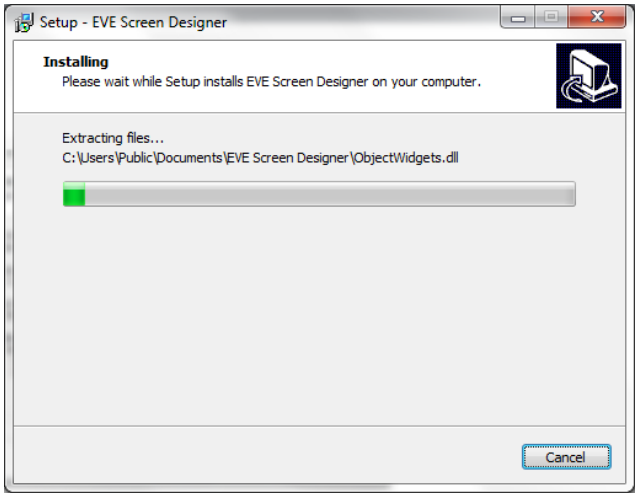


Figure 11 - Installation page

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

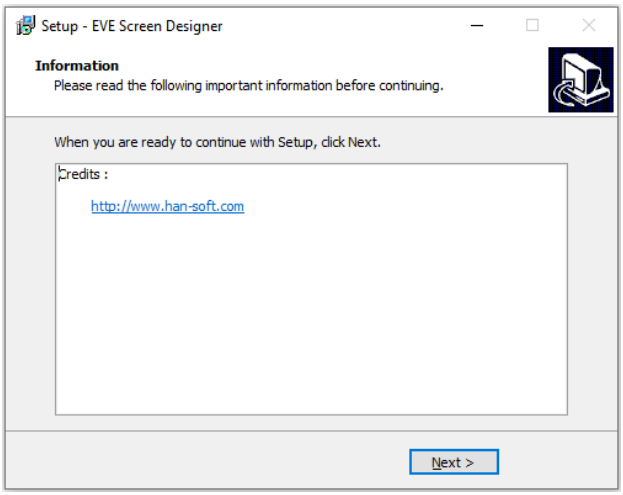


Figure 12 - EVE Screen Designer information page

Copyright © Bridgetek Pte Ltd

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

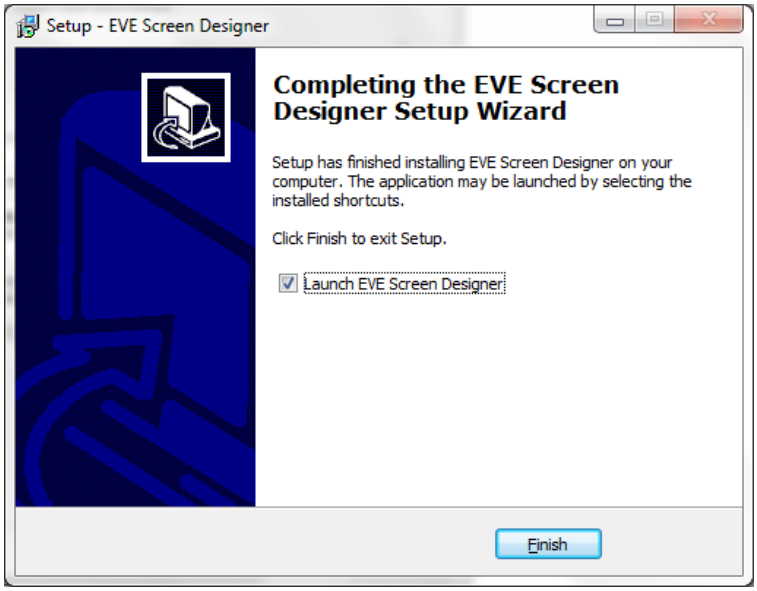


Figure 13 - EVE Screen Designer complete installation

Copyright © Bridgetek Pte Ltd

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 5 - Installation Folder

Copyright © Bridgetek Pte Ltd

D. Working with ESD

Major features

Support full BT81X features

Users can refer to the EVChargePoint examples at *<ESD Folder>/Examples/Advanced/EVChargePoint*. Please note that opening these examples may take over 5 minutes due to ASTC, bitmap and font conversion. This example showcases all the new features of BT81X. Here are the details:

1. Apply ASTC Bitmap Format for images and fonts.

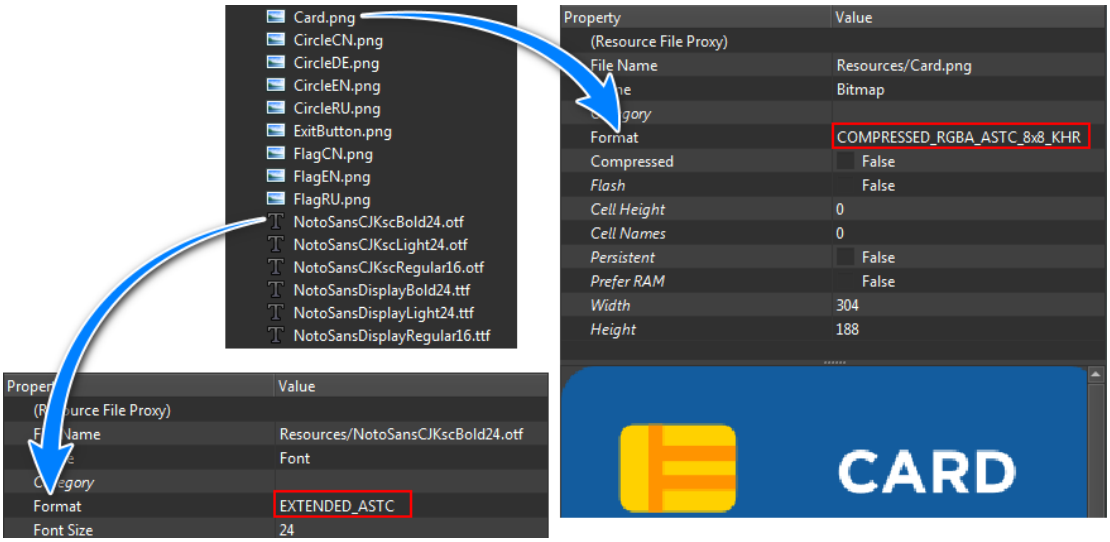


Figure 14 - ASTC applied for images and fonts

2. Resources are stored and loaded from flash.

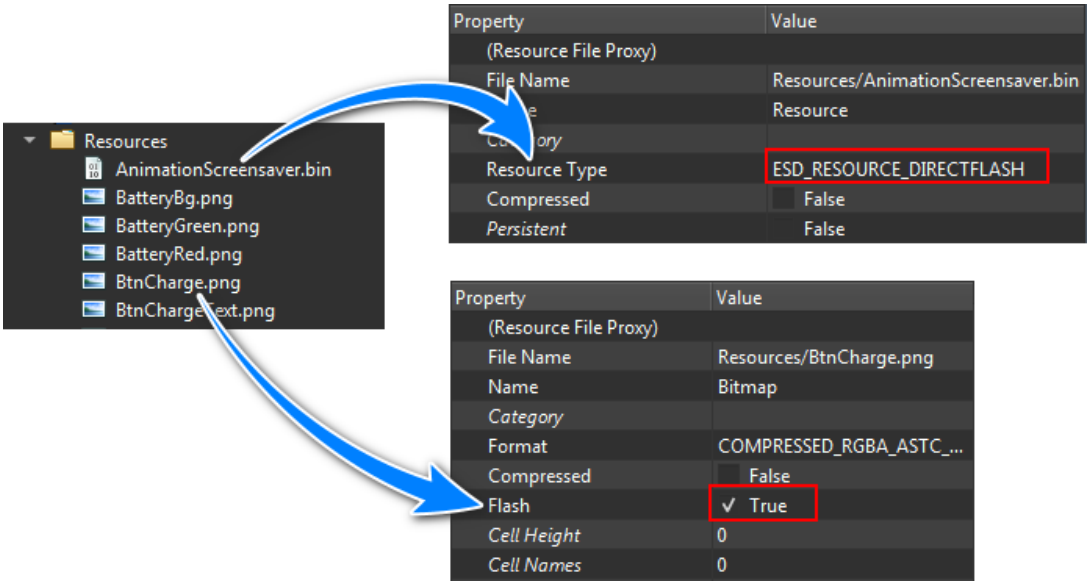


Figure 15 - Load resources from flash

Copyright © Bridgetek Pte Ltd

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

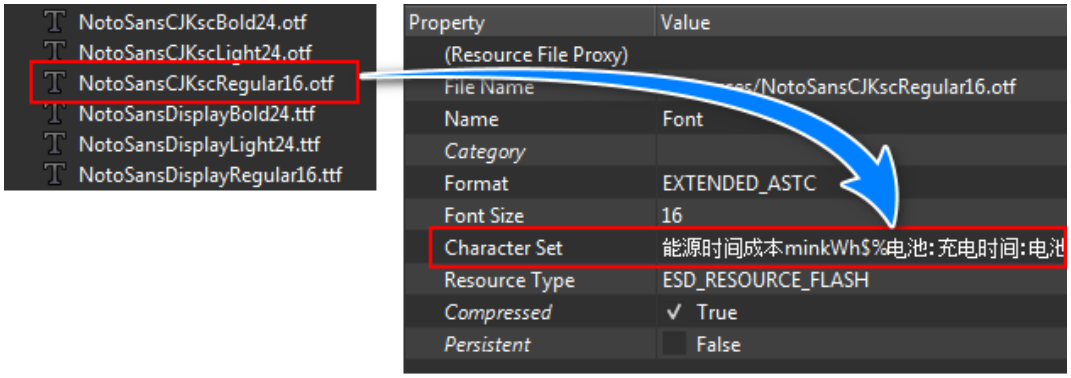


Figure 16 - Use Unicode font with pre-defined character set

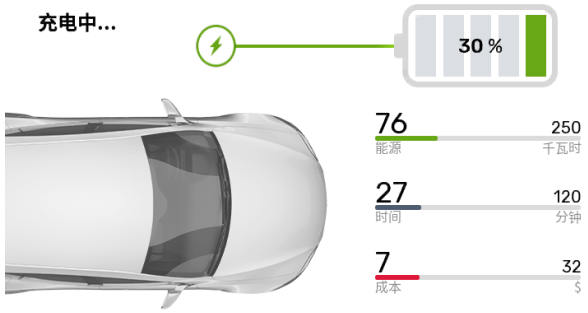


Figure 17 - Chinese font used in application

Support for ASTC bitmap format

ESD offers the option to convert images to the ASTC bitmap format, providing significant storage savings without compromising image quality.

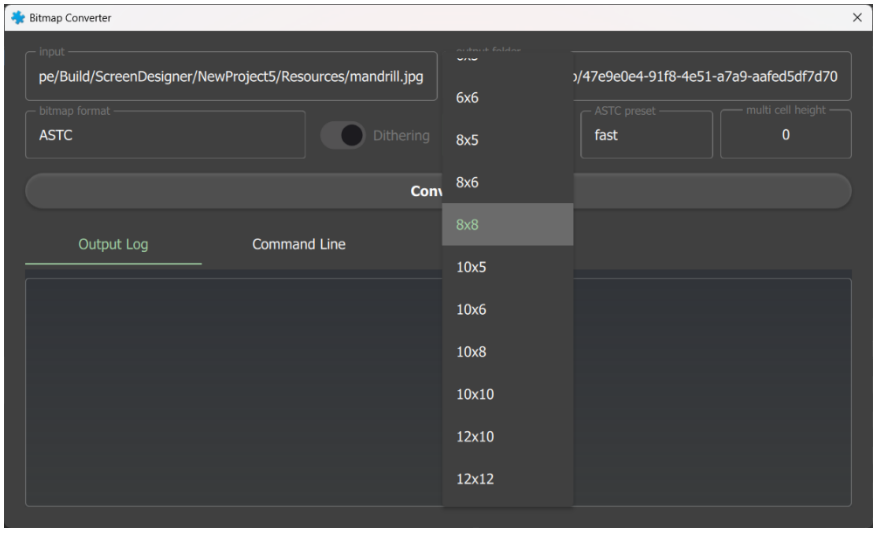


Figure 18 - ASTC Format

Copyright © Bridgetek Pte Ltd

Support DXT1 bitmap format

DXT1 (also known as Block Compression 1 or BC1) is the smallest variant 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 DXT1 bitmap format:

- **DXT1_L1_RGB565**: The 4x4 two-bit look up table uses **L1** format, while c0, are in **RGB565** format.
- **DXT1_L2_RGB565**: The 4x4 two-bit look up table uses **L2** format, while c0, c1 are in **RGB565** format.
- **DXT1_L1_PALETTED/DXT1_L2_PALETTED**: The 4x4 two-bit look up table uses **L1**/**L2** format, while c0 and c1 are in PALETTED565 format. Replacing RGB565 layers with **PALETTED565** reduces the size of the color layers to 2 bytes per block, resulting in approximately 3 bits per pixel plus a color palette of up to 512 bytes.

As with other bitmap formats, once the image is added to a project, users can set the "Format" property of input image to a DXT1 format. The example project "BitmapFormats" demonstrates the effect of DXT1 bitmaps by comparing them with other bitmap formats supported by EVE.

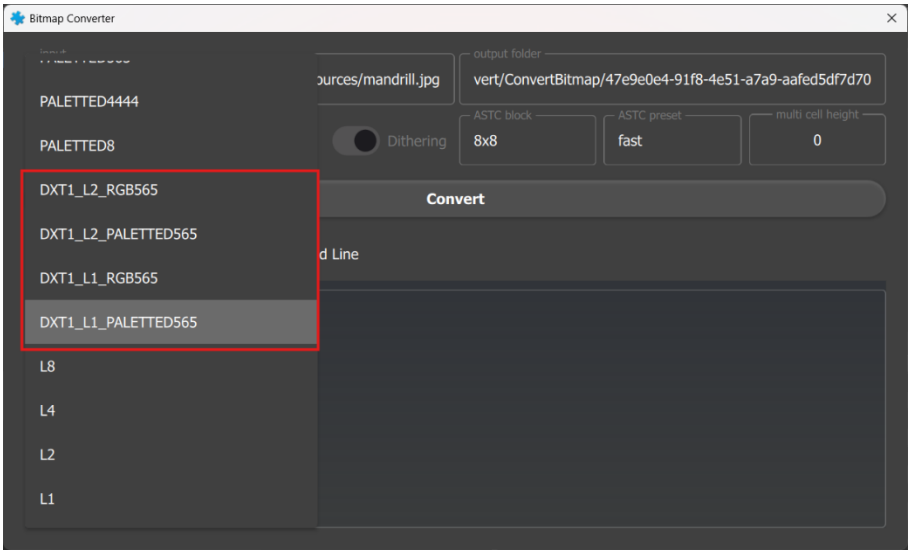


Figure 19 - DXT1 Bitmap formats

Copyright © Bridgetek Pte Ltd

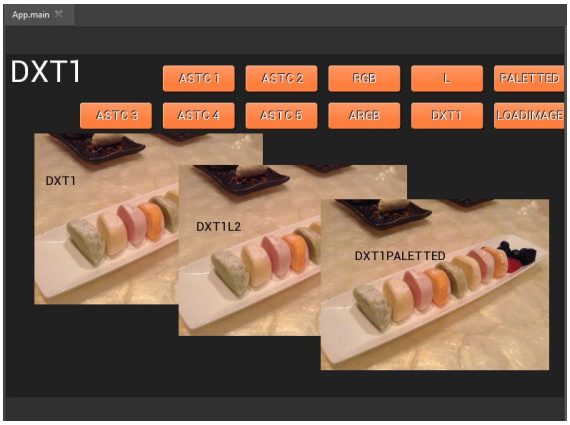


Figure 20 - DXT1 format bitmap in BitmapFormats example project

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).

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

Figure 21 - DXT1 output format

Users must download the converted bitmap resources, either *_b0.raw and *_c0.raw **or** *_b0.bin and *_b0.bin to the root directory of the SD card. Then, insert the SD card into the hardware module. The .esdm file does not need to be copied to the SD card unless the dynamic bitmap loading feature is used. For dynamic bitmap loading, the .esdm files must be placed in 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).

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

Figure 22 - DXT1L2 output format

To use the converted bitmap resources, transfer either *_b0.raw and *_c0.raw **or** *_b0.bin and *_b0.bin files to the root directory of the SD card. The .esdm file does not need to be copied to the SD card unless dynamic bitmap loading is used.

Copyright © Bridgetek Pte Ltd

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.

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	

Figure 23 - DXT1PALETTED output format

To utilize the converted bitmap resources, download either *_b0.raw, *_index.raw, and *_lut.raw or *_b0.bin, *_index.bin, and *_lut.raw files to the root directory of the SD card. Then, insert the SD card into the hardware module. The .esdm file does not need to be copied to the SD card unless dynamic bitmap loading is used. If dynamic bitmap loading feature is used, .esdm files must also be placed in the SD card's root directory.

Support extended font format

A Unicode font file can be added to a project by selecting ***File → Add*** from the menu. After choosing the file, the Font Converter dialog opens, allowing users to adjust the conversion settings before importing.



Figure 24 - Font Converter dialog for BT82X

Copyright © Bridgetek Pte Ltd

1 **Font To Convert**: Shows the complete file path of the selected input font and is automatically populated after browsing.

2 **Output Folder**: The folder where the converted font will be saved. This path is automatically generated by ESD.

3 **Input Character**: Select the character set to convert.

3.1 Select the character set from utf-8 file.

3.2 Select the character **Unicode** range to convert.

4 **Point Size:** Set the desired font point size. Any character wider than 255 pixels in width will be removed.

5 **DPI** (only BT82X): Set the DPI value for the dedicated rendering LCD panel. The formula to calculate pixels is:

Pixel = Point × DPI / 72.

6 **First Characters**: Set the first index for converted characters. The valid range is 1-127.

7 **Bitmap Format**:
- The Legacy structure supports bitmap formats L1, L2, L4, and L8.
- The Extend structure supports the same formats and adds ASTC support.

8 **Retain Space**: Fix the space character at code point 0x20.

9 **Kerning** (Only BT82X and font structure Extend 2): Enable kerning support for fonts with a GPOS table.

10 **Structure:**
- BT81X uses Legacy and Extend.
- BT82X uses Legacy, Extend 1, and Extend 2.

11 **Relocatable** (only BT82X): output is relocatable asset.

12 **Convert Button:** Starts the conversion process

Code Point Mode (only BT81X): Select either UTF-8 or Ordinal.
- UTF-8: Addressing the characters with UTF-8 code point.
- Ordinal: Addressing the characters with index 0, 1, 2, …, n

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

Copyright © Bridgetek Pte Ltd

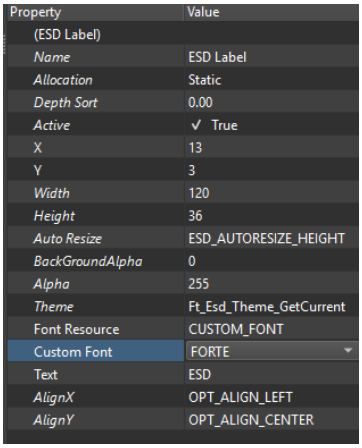


Figure 25 - Select Unicode Font Resource

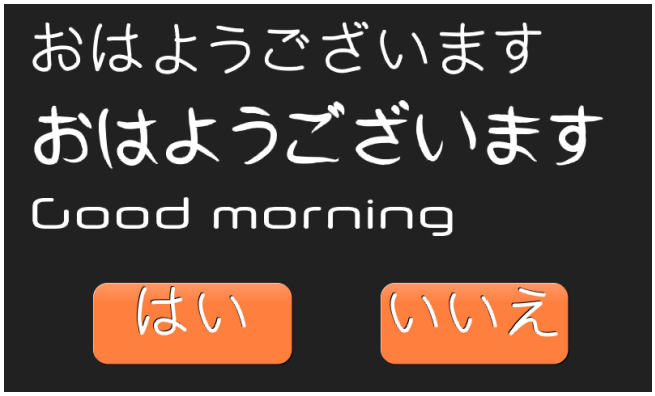


Figure 26 - Unicode font applied for Label and Button

ASTC Encoder

Since ESD 4.12, the ASTC encoder has been upgraded to version 2.0. This updated encoder, developed by ARM, handles the conversion of images into ASTC format and delivers up to 30% faster performance compared to ASTC encoder version 1.0.

Video Converter

ESD enables users to add video files to a project and converts them into an EVE-specific format using ffmpeg.

A video file can be added by selecting ***File*** → ***Add*** from the menu bar. After choosing a video file in the file browser, the **Video Converter** dialog will automatically appear, enabling you to configure the conversion settings before importing the video into the project.

Copyright © Bridgetek Pte Ltd

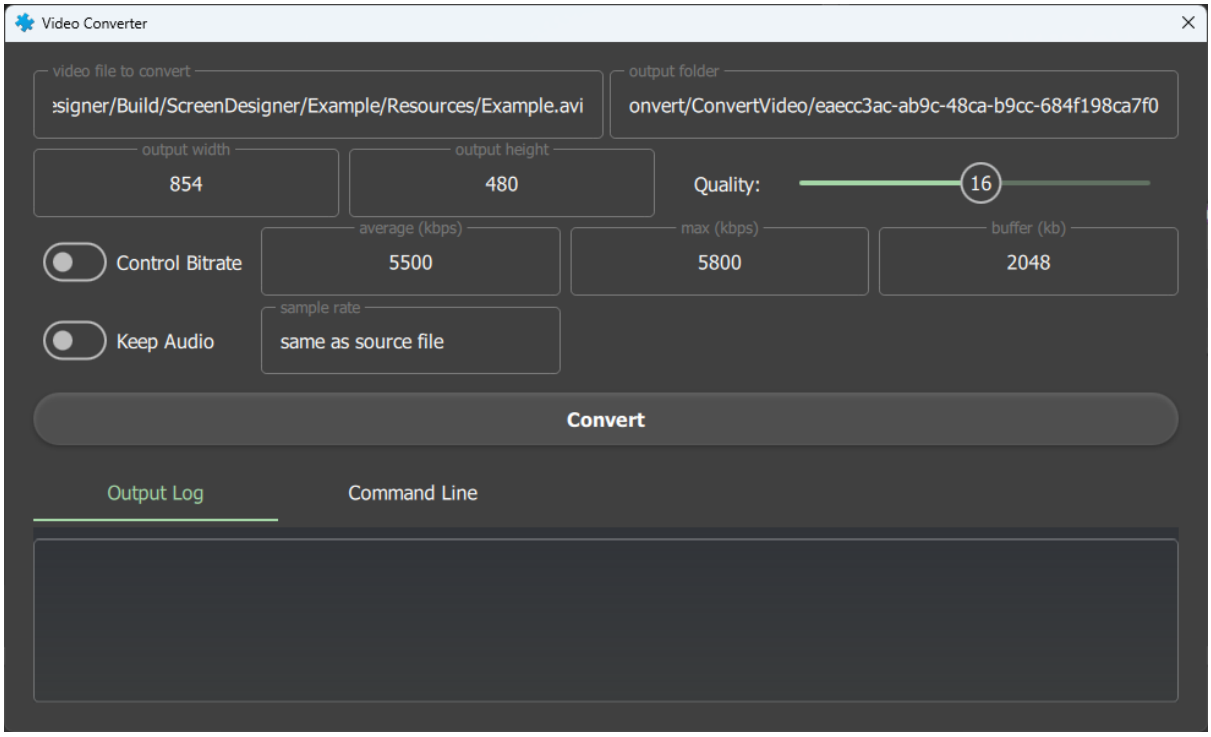


Figure 27 - Video Convert Dialog.

Video File To Convert: Displays the full path of the selected input video. Automatically filled after browsing.

Output Folder: Folder where the converted video will be saved. This path is auto generated by ESD.

Resolution: Specifies the output video resolution. The resolution will be generated based on original video resolution and this width and height can be adjusted.

Quality: Range from 1 (best quality) to 31 (worst quality).

Control Bitrate: Enables manual control of the output bitrate. When enabled, the Quality setting is ignored.

- Average (kbps): Set the average bit rate for the output video.
- Max (kbps): Set maximum bitrate for the output video.
- Buffer Size (kbps): Set buffer size for the output video.

Keep Audio Stream: Allows the user to keep or remove the audio stream in the output.

Sample Rate: Specifies the audio sample rate for the output. Users can retain the input sample rate or select from the available values: 48000, 44100, 32000, 22500, 16000, or 8000 Hz. This option is only editable if Keep Audio Stream is enabled.

Convert Button: Starts the conversion process.

Output Log: Shows status messages, progress, errors, and encoder feedback during conversion.

Copyright © Bridgetek Pte Ltd

Dynamic Bitmap Loading

Dynamic bitmap loading is a technique that loads bitmap images (graphical assets) into memory only when they are needed, instead of preloading them all during application startup. This approach optimizes memory usage, improves performance, and enables applications to handle large or numerous images without overloading system resources. The *.esdm* file format is one of the mechanisms used to support this feature.

For ESD users who want to utilize the dynamic bitmap loading at runtime, the .esdm files must be copied to the root directory of the SD card. ESD has the example project called "IconTheme". Review this example project for further details.

Support Animation

ESD enables users to import and manage animations within a project. Animations typically consist of sequences of images or video clips that are optimized for smooth playback on EVE displays.

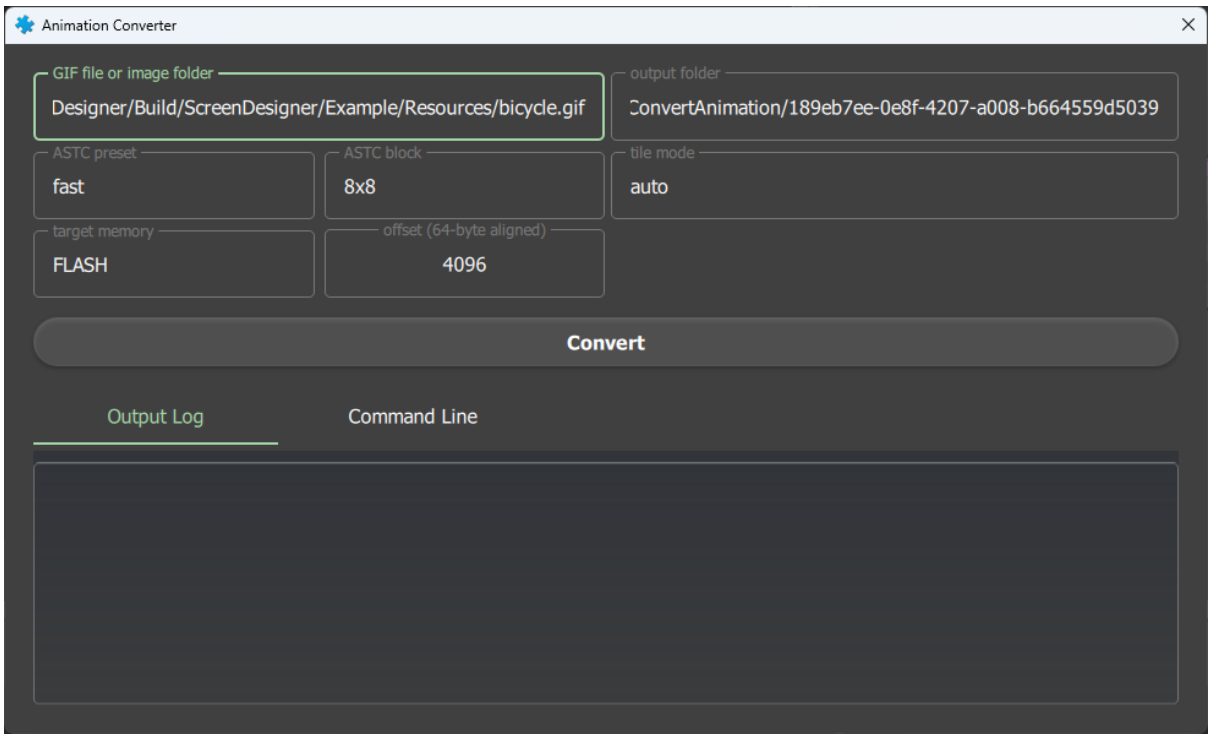


Figure 28 - Animation converter dialog.

GIF File or Image Folder: Displays the full path of the selected input video. Automatically filled after browsing.

Output Folder: Folder where the converted video will be saved. This path is auto generated by ESD.

ASTC Preset: Available options include fastest, fast, medium, thorough, and exhaustive. Encoding speed is inversely proportional to output quality:

- Fastest: Quickest performance, lowest quality
- Exhaustive: Highest quality, slowest processing

Copyright © Bridgetek Pte Ltd

ASTC Block: Block sizes range from 4×4 to 12×12. Smaller blocks give higher quality with lower compression; larger blocks give higher compression with lower quality.

Tile Mode: Defines the tiling method applied across frames. Options:

- Auto: Automatically selects the optimal tile size
- Not Apply: The entire frame is compressed as a single unit—there is no tiling used to subdivide it (Highest quality, slowest processing).
- Manual: Users set Tile Width and Height manually. Width × Height must be a multiple of 4 to meet ASTC rendering requirements.

Target Memory (BT81X only): Defines the animation destination memory. ESD supports animation storage only in flash memory for BT81X.

Offset (64-Byte Aligned): Specifies the memory location where the animation will be loaded. The address must be aligned to 64-byte boundaries, with the starting address set at 4096.

Convert Button: Starts the conversion process.

Output Log: Shows status messages, progress, errors, and encoder feedback during conversion.

Reconvert the resource file

Users can reconvert a resource file by following the steps below:

1. Select the resource file that needs to be reconverted in the Resource Folder.
2. Right-click the selected resource file and choose **Convert** from the context menu.

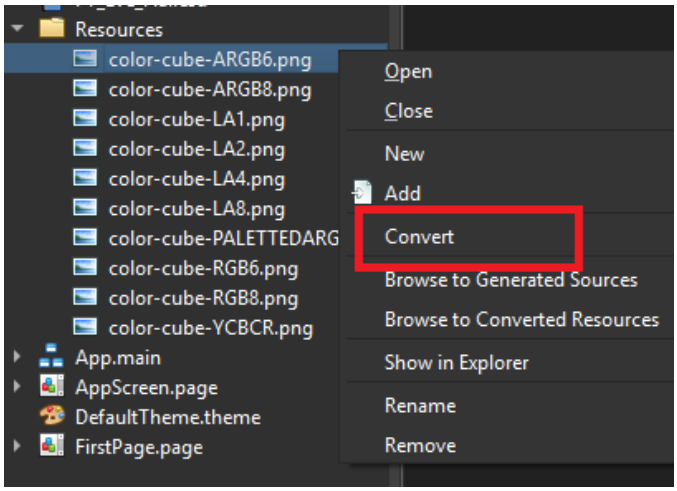


Figure 29 - Reconvert Added Resource Files

3. The Convert dialog will appear. Modify the conversion settings as needed.

Note: When the build target is switched, ESD reuses converted resources that remain valid for the selected target. Resources are converted again only when they are missing, outdated, or incompatible with the new target.

Copyright © Bridgetek Pte Ltd

Generate Flash file and write Flash file to device

Resources can be loaded from **EVE**-connected flash memory or **RAM_G**. When the  **[Upload Flash and Run Executable]** button is pressed, ESD generates the flash file for BT81X projects and uploads it to the device.

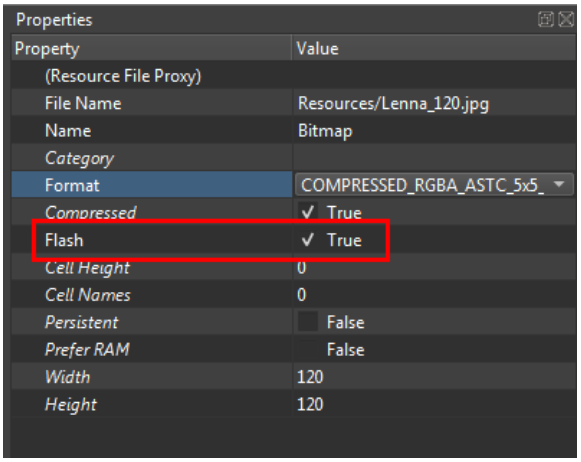


Figure 30 - Mark resource as storage in Flash

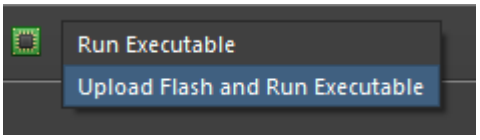


Figure 31 - Upload Flash file to device

Copyright © Bridgetek Pte Ltd

LittleFS File System with EVE connect flash memory

LittleFS is a fail-safe file system designed for embedded systems, particularly for microcontrollers that use external flash storage. Devices such as BT815 and above support flash memory. The AssetBrowser example project demonstrates how to use the LittleFS file system in ESD. Kindly review the following steps to understand how to integrate and use LittleFS within your project.

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

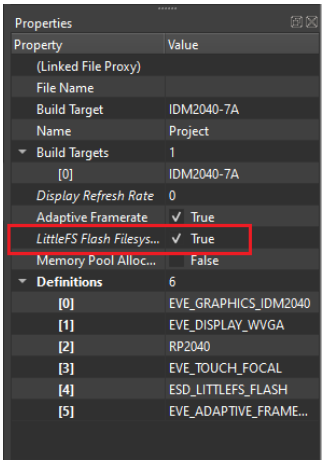


Figure 32 - LittleFS enable option in Project

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

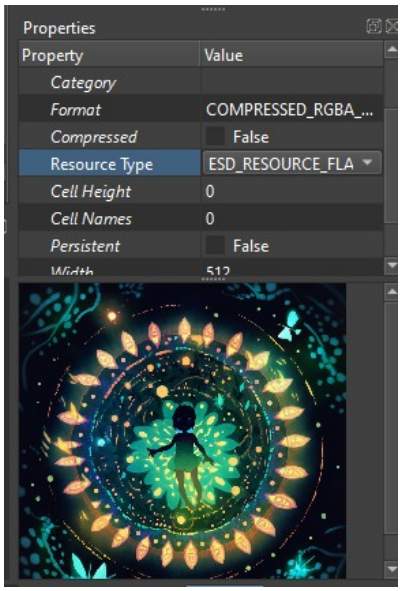


Figure 33 - Select ESD_RESOURCE_FLASH

Copyright © Bridgetek Pte Ltd

3. 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.

```
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 34 - C source file in the Project

```
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 35 - 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.

```
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);
}
```

Figure 36 - Reading each entry

Copyright © Bridgetek Pte Ltd

5. Generate rows consecutively using the "addWidget(context, info.name, info.size)" function, displaying the image file name, file size, and a Preview push button. During the function's validation process, rows will not be generated if the file name contains ".esdm". If the file name does not contain ".esdm", the rows will be created as expected. For more information ".esdm", refer to Resource Metadata section.

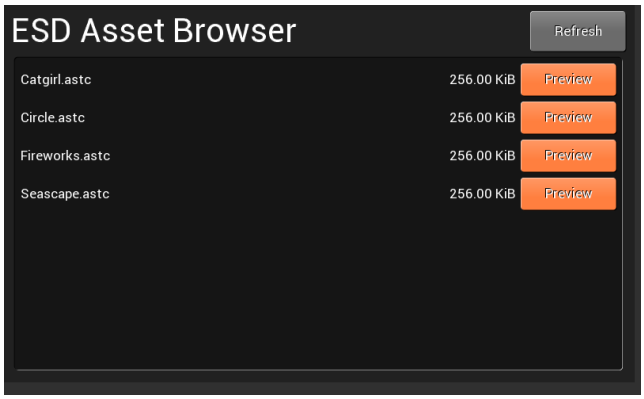


Figure 37 - Show all the image names in the project Asset browser

6. After clicking the Preview button, the loadResource(context) function is triggered. Within this function, the current image file name is assigned to bitmapinfo. Then, the flash address is retrieved using Esd_BitmapInfo_LoadFlashAddress(&context->BitmapInfo) and bitmapinfo, along with the obtained flash address, is passed to the Image Viewer for display.

```
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);
    }
}
```

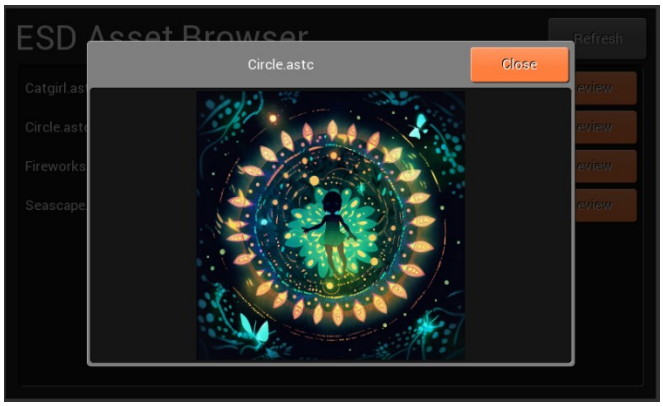


Figure 38 - Show preview image in the project AssetBrowser

Copyright © Bridgetek Pte Ltd

Restructured EVE HAL Library

EVE HAL Library includes C source files and a subfolder named "Hdr" containing the C header files. It provides the source code and header files for the EVE HAL platform, as well as coprocessor command definitions.

Memory management

Some widgets in ESD use heap memory, that is, they rely on `malloc` and `free`. In embedded systems, using heap memory is generally discourage due to its potential drawbacks. However, because ESD is a GUI design tool, fully relying on static memory is challenging. Starting with ESD4.16, a memory pool management module has been introduced. This module allows users to limit heap memory usage within a controlled range, as outlined below:

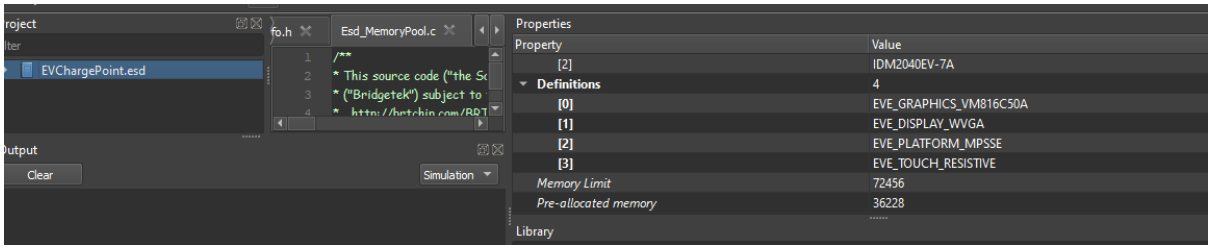


Figure 39 - Config the memory

Figure 39 shows that two variables - *Pre-allocated Memory* and *Memory Limit* are defined in the root project properties to manage memory allocation. *Pre-allocated Memory* represents the estimated memory usage of the project, while *Memory Limit* is the maximum memory that can be assigned to the project. If the *Pre-allocated Memory* is insufficient, ESD will try to extend memory usage up to the *Memory Limit*. The minimum value for both *Pre-allocated Memory* and *Memory Limit* is 0, since not all widgets use heap memory. The default values are 36,228 for *Pre-allocated Memory* and 72,456 for *Memory Limit*. Users should adjust these values based on the actual memory requirements of their project.

For BT820-based projects, EVE Screen Designer also includes a separate memory allocator for managing the larger EVE graphics memory space available on BT820 devices. This allocator is independent of the host-side memory pool described above. It manages EVE memory used by application resources and buffers, and supports allocation, garbage collection, and defragmentation to improve memory utilization when projects use large or multiple resources. Certain memory regions, such as reserved memory, framebuffer/swapchain buffers, and Media FIFO memory, may require dedicated memory areas and are handled separately from normal application resource allocations.

Copyright © Bridgetek Pte Ltd

UI enhancements

The following user interface enhancements have been implemented in ESD -

- Added filtering capabilities to the Library Browser and Project Browser windows
- Expanded toolbar buttons for a more comprehensive platform description
- Introduced a new menu option to clean up unused/generated project files
- Simplified the process of adding new resources to a project
- Added a context menu for document tabs

BT82x ESD Project Configuration

This section explains how to configure a BT82x project in ESD. It also describes the default values and where these values are hard-coded for the following configurations: Scanout, Video Playback, and LVDS RX.

1. Scanout Format Configuration

   Users can configure the scanout format in the ESD project settings. The available formats are YCbCr, RGB8, and RGB6.

   The default scanout format is YCbCr

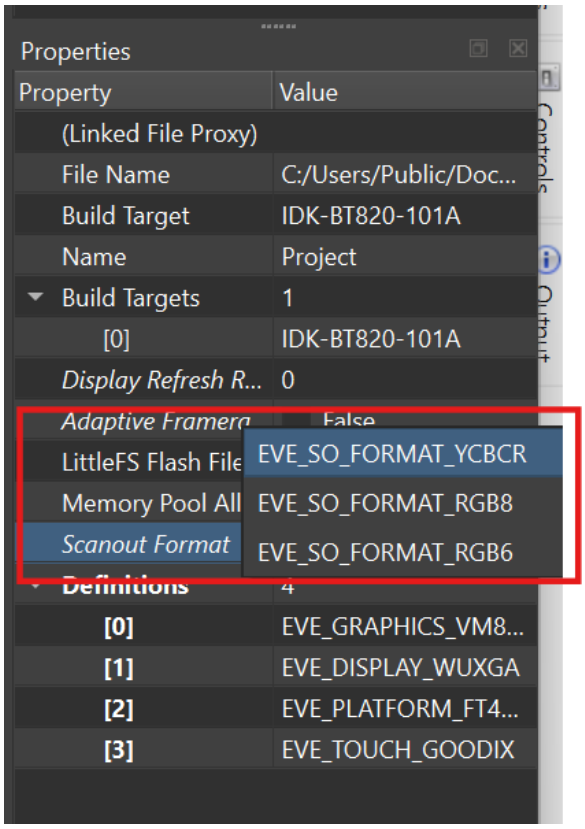


Figure 40 - Main Project properties

Copyright © Bridgetek Pte Ltd

2. Scanout Configuration

ESD configures SC0 with two scanout buffers. The buffer size and addresses are calculated automatically depend on the scanout format configuration.

Register	Configuration
REG_SC0_SIZE	2 (hardcoded)
REG_SC0_PTR0	Calculated by ESD
REG_SC0_PTR1	Calculated by ESD

The buffer addresses are calculated as follows:

```
uint32_t endAddr = phost->RamGSize - RAM_G_RESERVED;
int32_t bufferSize = scanoutSrcStride * config->Height;
uint32_t SC0_PTR0 = endAddr - bufferSize;
uint32_t SC0_PTR1 = SC0_PTR0 - bufferSize;
phost->RamGAvailable = buffer1;
```

The two SC0 buffers are allocated at the end of RAM_G. RamGAvailable indicates the remaining RAM_G available for other resources.

3. Video Playback Configuration

Swapchain 1 (SC1) is reserved for video playback.
By default, video playback is not configured, so no SC1 buffers are allocated.

4. LVDS RX Configuration

Swapchain 2 (SC2) is associated with the LVDS RX path.
The default BT82x ESD project has LVDS RX disabled and does not allocate SC2 buffers.

Copyright © Bridgetek Pte Ltd

Setting Platform Specific Properties

ESD allows users to create a single project that supports multiple platforms with different screen sizes. As a result, the same widget may have different property values depending on the target platform. For example, a button displayed on a platform with a screen size of 480 x 320 may be scaled down compated to its size on a platform with a screen size of 800 x 480.

Users can define property values specific to the current target platform through the Property Browser. This can be done by right-clicking on the desired property of the selected widget and accessing the context menu, as illustrated in the picture below –

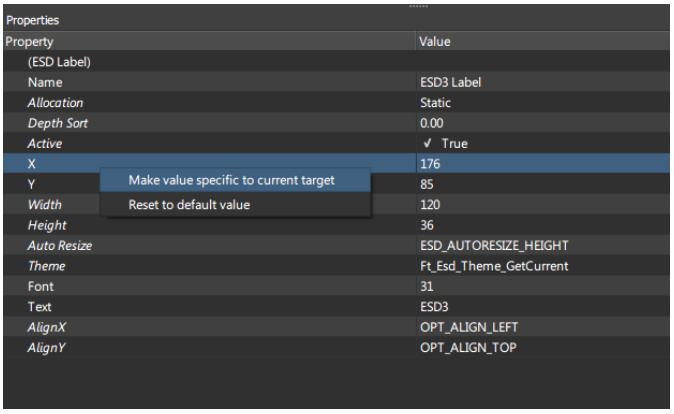


Figure 41 - Making Value Specific to Current Target

Users can remove a value specific to the current target platform through the Property Browser. This can be done by right-clicking the selected property of the current widget and using the context menu, as illustrated in the picture below –

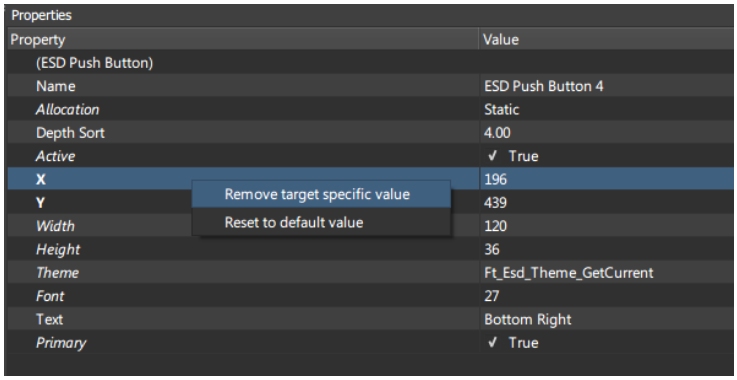


Figure 42 - Removing Target Specific Value

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

Migrating existing projects to new platform

Existing projects are typically developed for a specific platform. Users can migrate a project to a different platform by selecting the desired option from the "Build Target" combo box. If the required target is not available in the "Build Target" list, users can manually configure it by selecting the appropriate options for "Eve Platform", "Display Resolution", and "Host Platform" from their respective combo boxes.

Copyright © Bridgetek Pte Ltd

Advanced User Settings

Advanced users can configure ESD settings to better support GUI application development with EVE products. The Advanced User Settings allow users to enable or disable specific features and functions within ESD. This section provides a brief overview of these settings and their effects on the application. As these configurations involve modifying the application's configuration file, they should only be performed by users with sufficient knowledge. It is recommended to keep all advanced settings at their default values unless necessary.

Application Settings

The "Main.config" file, in XML format, is located in the "Settings" folder within the installation directory. This file contains application-wide settings that may affect the interface and behaviour of ESD. Users may modify these settings to suit their application needs. After making any changes, the application must be restarted for the modifications 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".

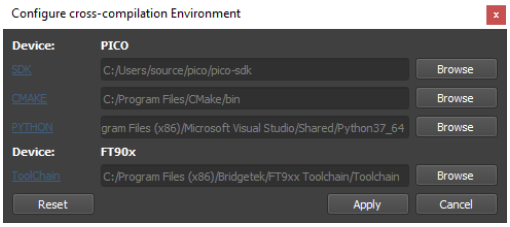


Figure 43 - Configure cross-compilation environments

- 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.

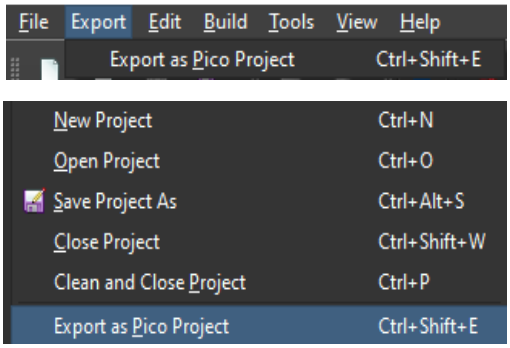


Figure 44 - Export option

Copyright © Bridgetek Pte Ltd

- 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 to be used by advanced users.

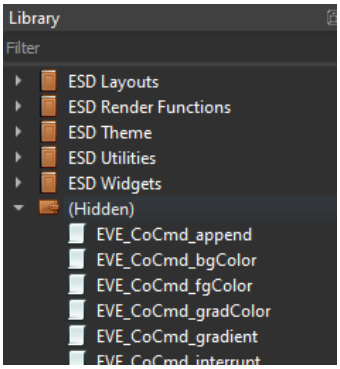


Figure 45 - Hidden folder

    - _WidgetDocument – Allows the user to create new customised widget.
    - d_LogicIncludeHeader – This allows the user to use "include header" logic node under the library which helps to add a header file into project.

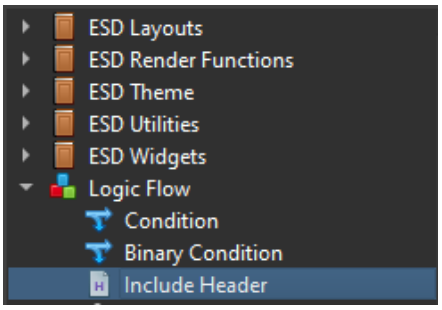


Figure 46 - Include Header

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

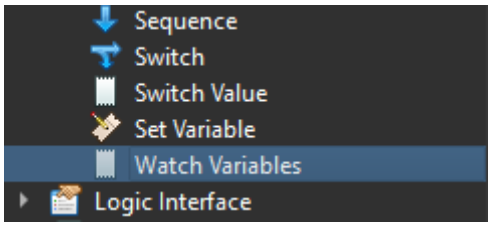


Figure 47 - Watch variables

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

Copyright © Bridgetek Pte Ltd

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

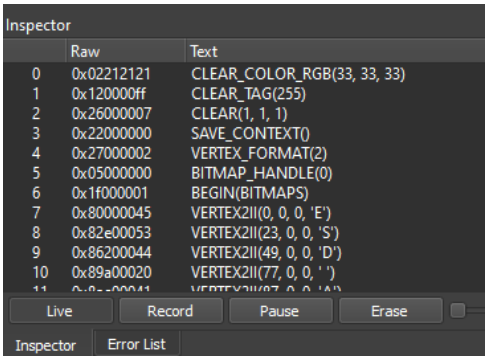


Figure 48 - Inspector dialog

- 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.

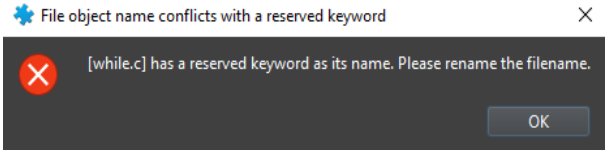


Figure 49 - File name keyword conflict error

Copyright © Bridgetek Pte Ltd

Troubleshooting

Building a project with an incorrect host platform may result in errors. Always ensure that you are using the correct EVE host platform. Refer to Table 2 - Platform vs Module Compatibility for further information.

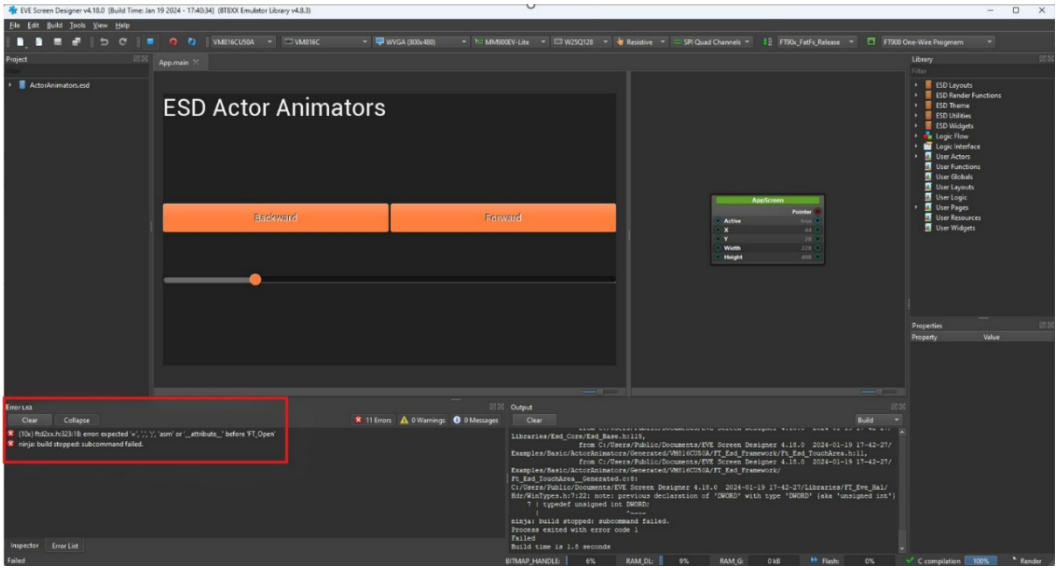


Figure 50 - Error list

As depicted in the figure above, a common error occurs when the EVE platform VM816C does not match the host platform MM900x. To resolve this issue, ensure that the correct host platform, FT4222, is selected.

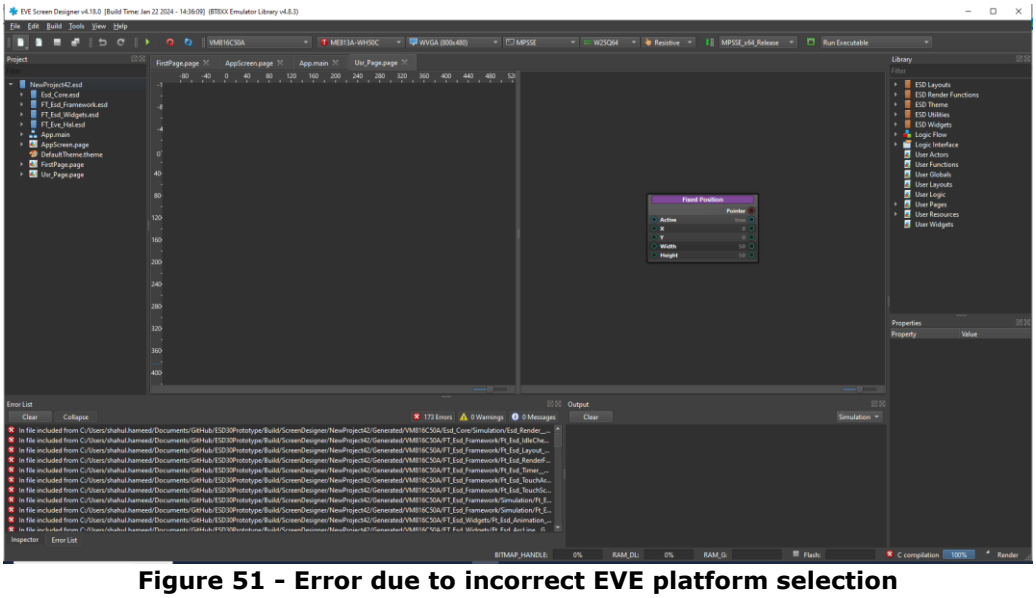


Figure 51 - Error due to incorrect EVE platform selection

As depicted in the figure above, an error may occur when creating a new project with the build targets VM816C50A or VM816CU50A if an incorrect EVE platform is selected. Ensure that the correct EVE platform, VM816C, is chosen to avoid this issue.

Copyright © Bridgetek Pte Ltd

E. Getting Started

The Welcome Dialog

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

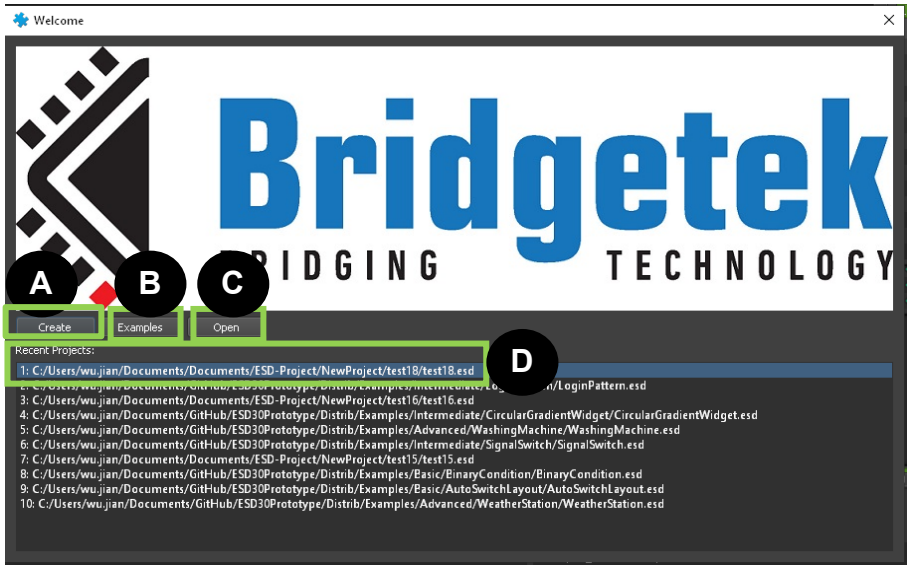


Figure 52 - ESD welcome dialog

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:

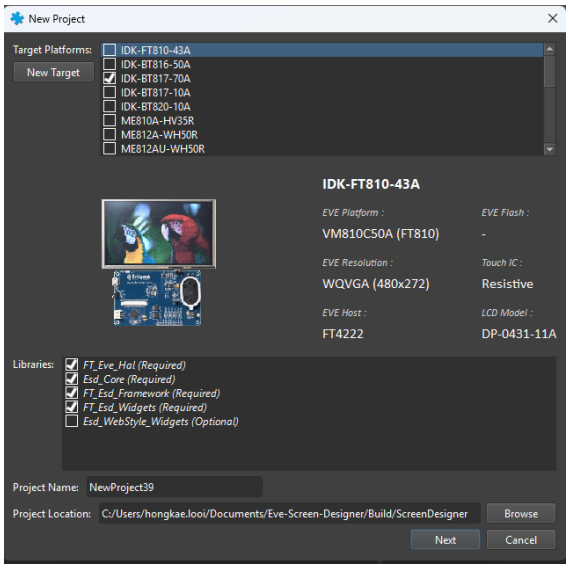


Figure 53 - ESD new project dialog

Copyright © Bridgetek Pte Ltd

The dialog allows users to create a new project using one of the predefined build targets shown in the picture, or to configure a custom build target by clicking the "New Target" button. When creating a new target, users can specify the platform, display resolution, host platform, flash chip, and touch IC.

Example Projects:

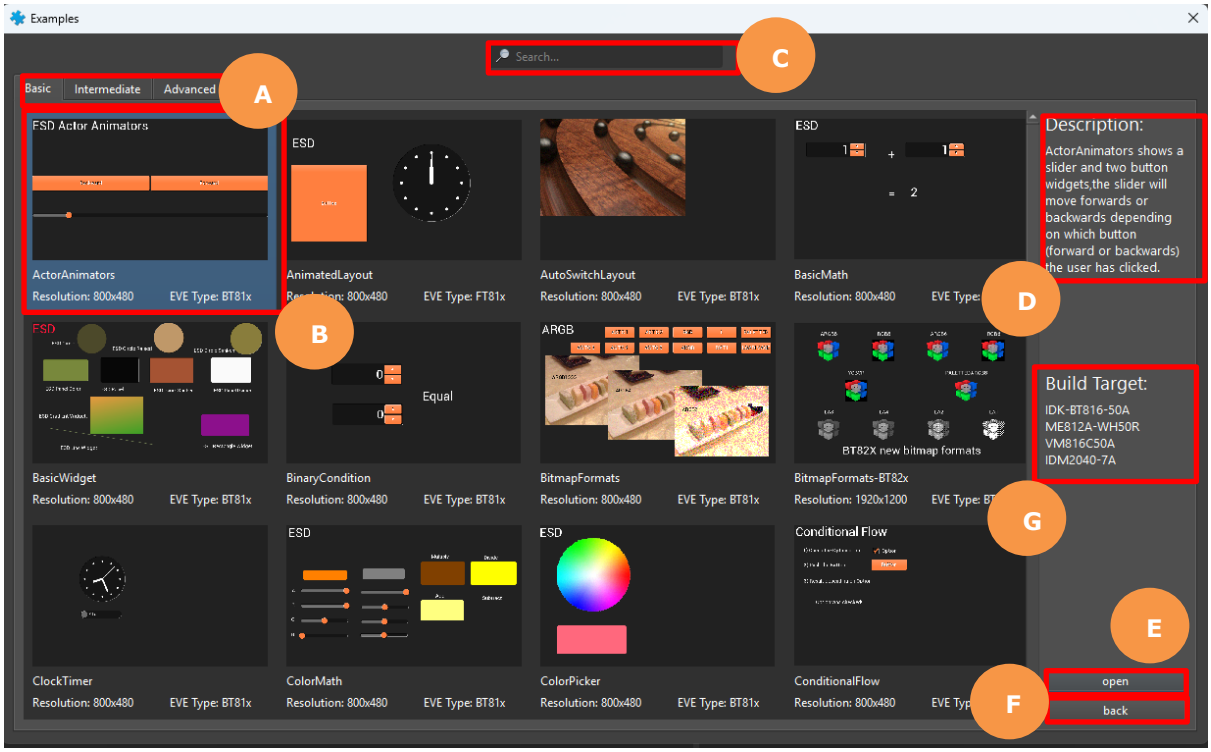


Figure 54 - ESD Example Projects Dialog

A: Tab representing 3 different categories of example projects. Users can switch between categories by clicking the tabs.
B: Thumbnail of the project, displaying the project name and its default resolution
C: Search bar that allows users to enter a keyword. The keyword will be matched against project names and descriptions, and the dialog will update to show only the matching projects.
D: Project description
E: Open button, which opens the selected project.
F: Back button, which returns to the original welcome dialog when clicked.
G: Project Build Target.

Copyright © Bridgetek Pte Ltd

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

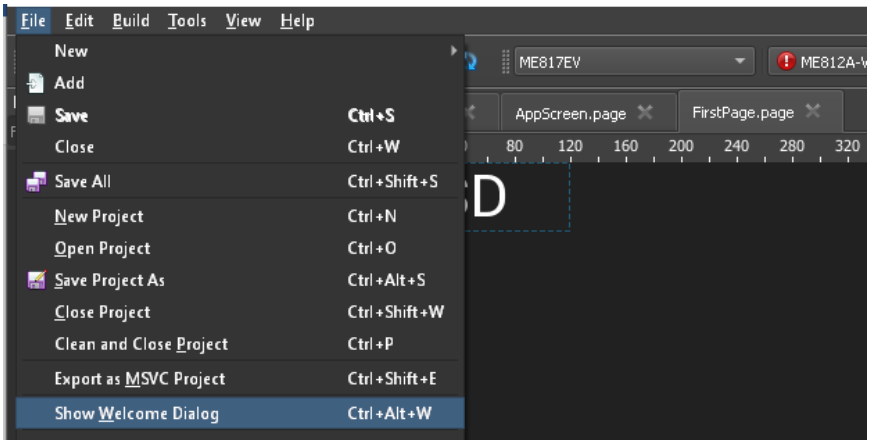


Figure 55 - ESD Welcome Dialog Menu

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*

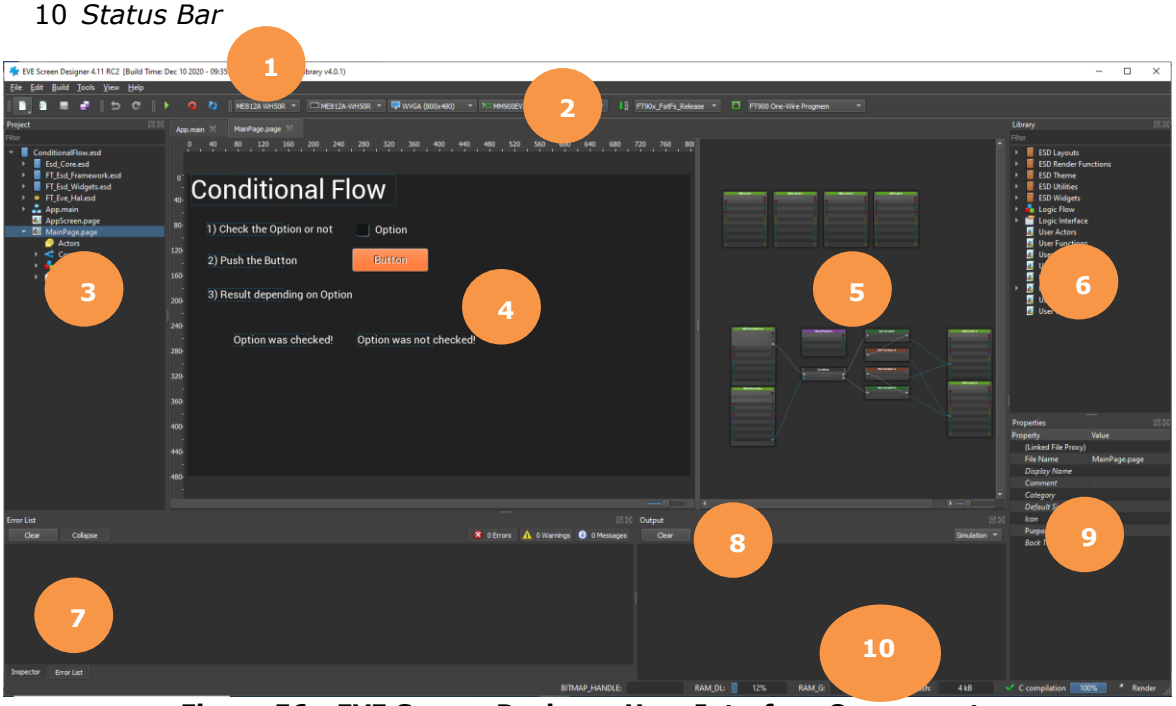


Figure 56 - EVE Screen Designer User Interface Components

Copyright © Bridgetek Pte Ltd

Menu bar

The *Menu bar* displays the headings for each drop-down menu. Commands are grouped under these headings according to their function.

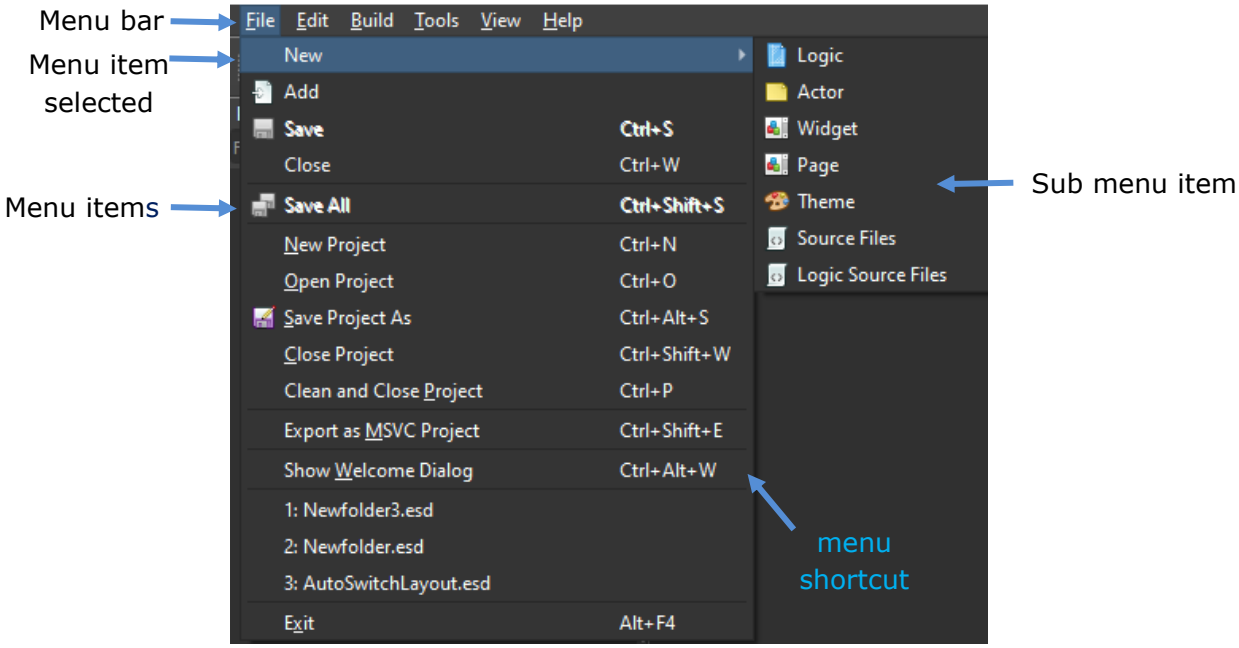
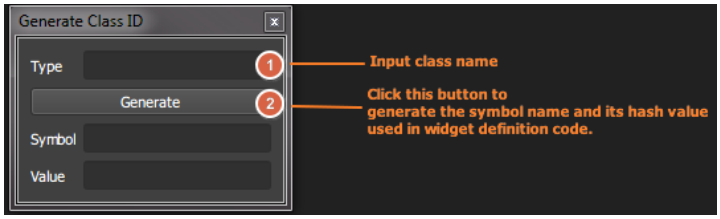
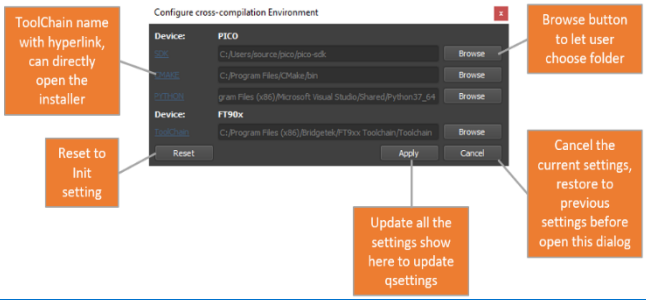


Figure 57 - Menu bar

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

Copyright © Bridgetek Pte Ltd

	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 useful when user prefers to write code in "C" language. 
	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. 
View	Project		To display or hide a component view. By default, all the components are displayed.
	Library		
	Properties		

Copyright © Bridgetek Pte Ltd

	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 third party software

Table 6 - 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.

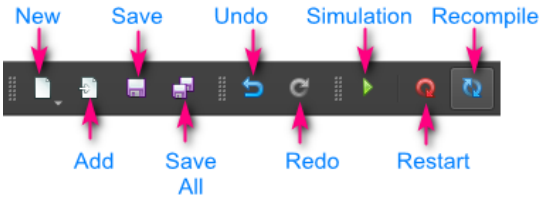


Figure 58 - Toolbar 1

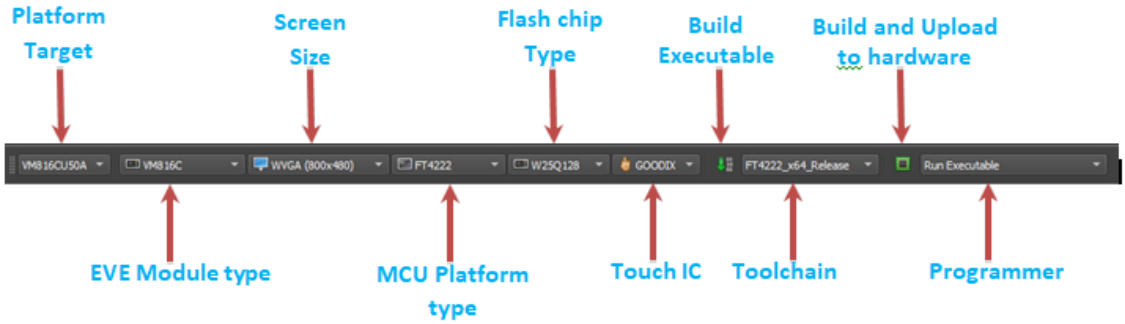


Figure 59 - Toolbar 2

Copyright © Bridgetek Pte Ltd

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 show EVE module and this EVE module is not recommend switching after project is created.
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 7 - Toolbar & Description

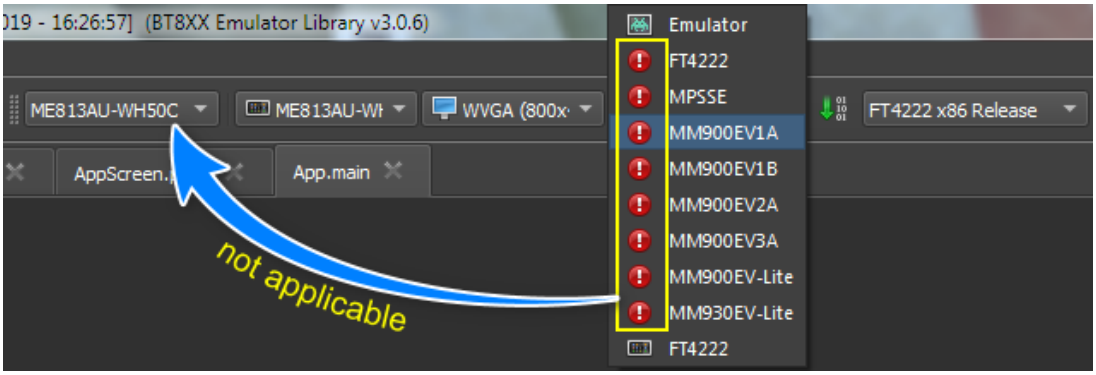


Figure 60 - MCU Type is not applicable to selected platform

The red exclamation mark shown in the image above indicates that these MCU types are not compatible with the currently selected platform.

Copyright © Bridgetek Pte Ltd

Touch IC configuration

The EVE chips FT811, FT813, BT881, BT883, BT815, BT817, and BT820 support capacitive touch controllers. Refer to the datasheet of the respective chip for a complete list of supported touch controllers.

In ESD, users can configure the touch IC controller 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.

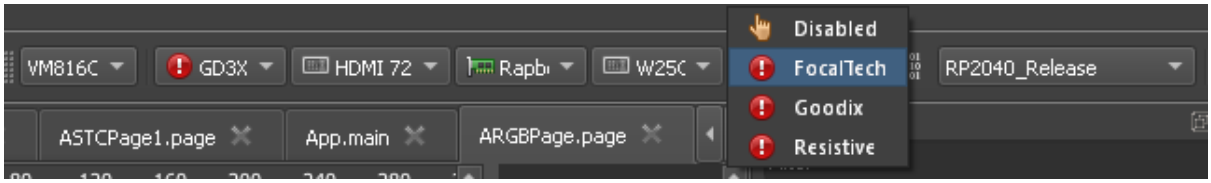


Figure 61 - CTP configuration

Project Explorer

The *Project Explorer* window organizes all project files in a tree view and displays the resources used by each file. It allows users to easily navigate through the pages of the project.

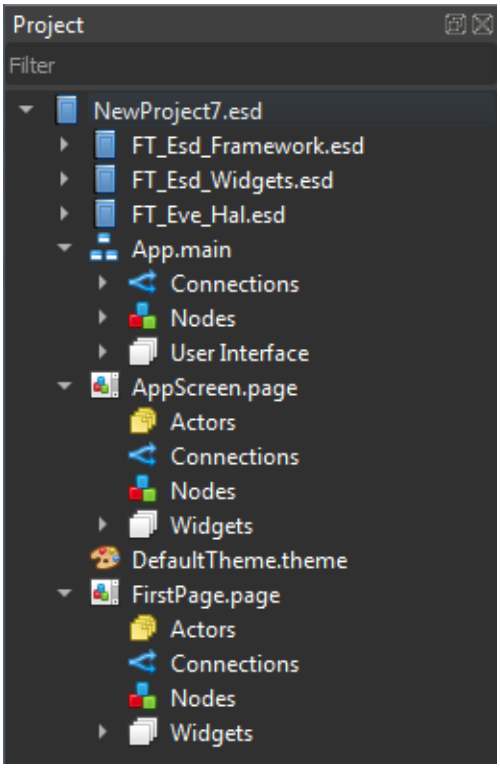


Figure 62 - Project Explorer window

Copyright © Bridgetek Pte Ltd

When a project contains many items, users can use the filter textbox to display a sorted list of matching items. Text that matches the filter is highlighted, and regular expressions are supported.

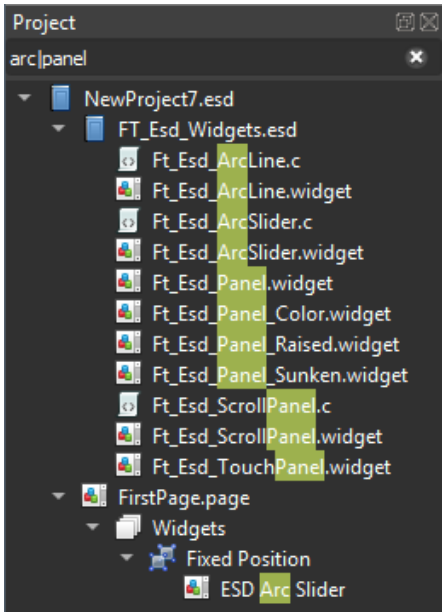


Figure 63 - Project Explorer Filter

Screen Layout Editor

The *Screen Layout Editor* displays the rendering output from EVE and allows users to edit C source code. Pages, widgets, and main files can be opened and modified within this editor.

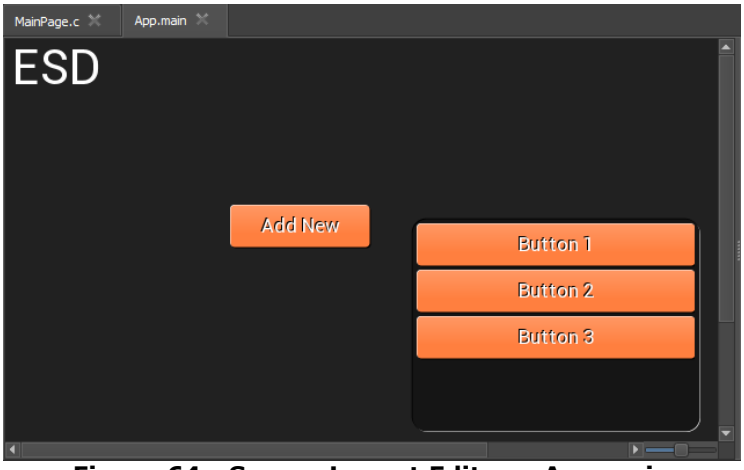


Figure 64 - Screen Layout Editor – App.main

Copyright © Bridgetek Pte Ltd

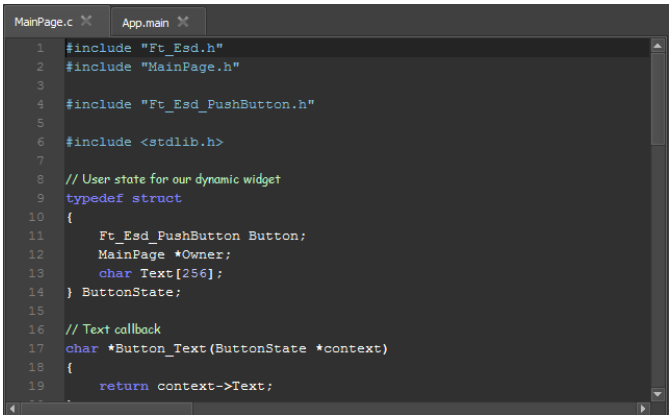


Figure 65 - Screen Layout Editor - MainPage.c (C Source Code)

The Screen Layout Editor shares a single viewport with the Logic Node Editor. Users can adjust its size by dragging the splitter handle.

When the simulations mode is *"off"*, users can drag and drop the widgets from the Library Browser to create the layout. For other types of logic nodes, if the EVE rendering process is not defined, the Screen Layout Editor will not allow them to be dropped.

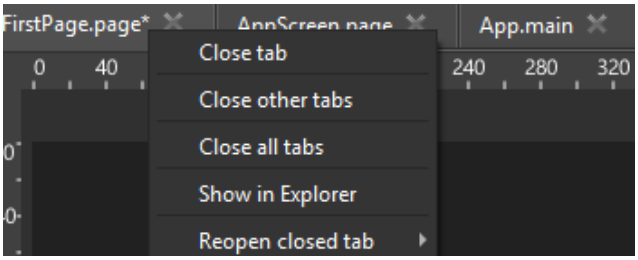


Figure 66 - Context menu on Tab bar

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
Show Explorer	Opens the project explorer panel.
Reopen Closed tab	Reopens the last closed tab.

Table 8 - Context Menu

Copyright © Bridgetek Pte Ltd

Logic Node Editor

The *Logic Note Editor* allows users to arrange logic nodes and establish connections to create logic maps. Users can drag and drop logic nodes from the Library Browser into the Logic Note Editor to form these connections.

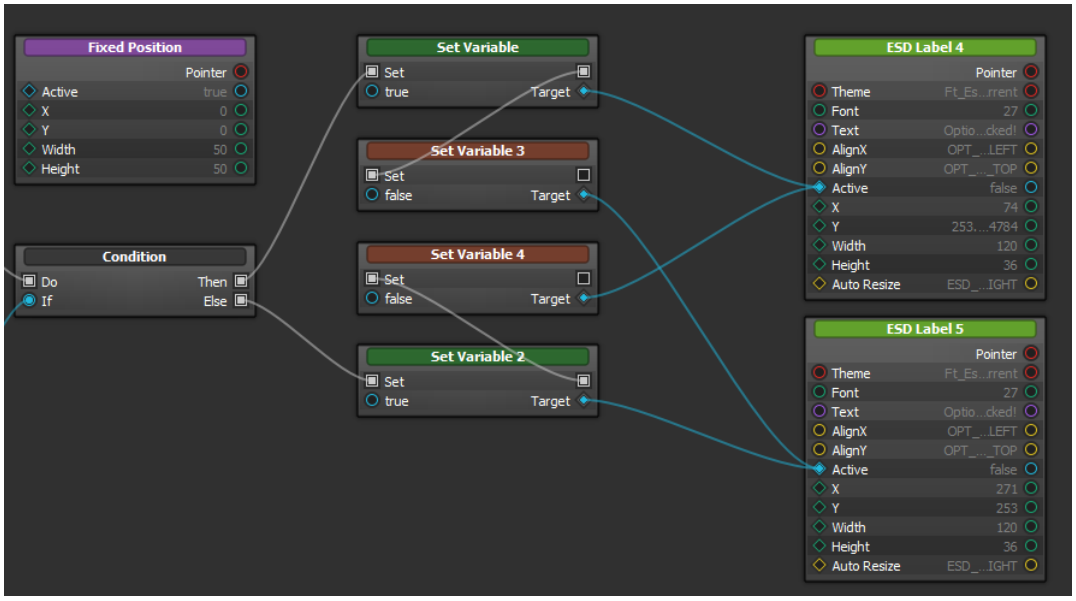


Figure 67 - Logic Node Editor

Library Browser

The *Library Browser* stores all available logic nodes and resources in ESD, including both built-in and user-defined items. Users can browse these logic nodes by category and select the ones they need for their project.

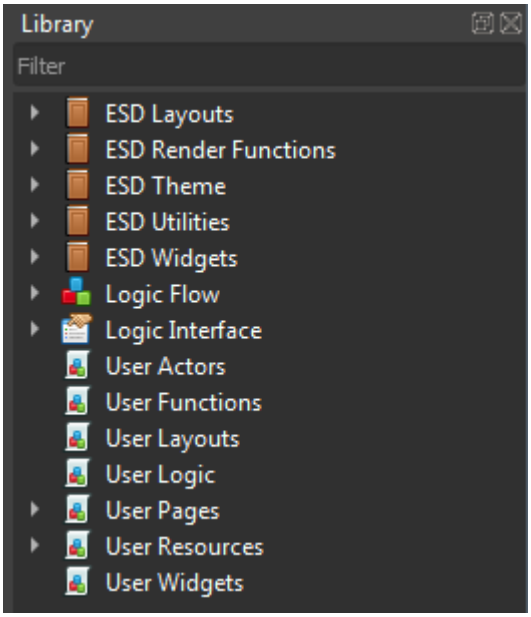


Figure 68 - Library Browser

Copyright © Bridgetek Pte Ltd

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 9 - ESD Libraries

Error List

The *Error List* is a dockable window that displays messages from ESD during project saving and recompilation. Any error shown indicates that the generated source code for the current project cannot be executed successfully. Users should review the logic defined in *Logic Node Editor* or check any user-defined source code in the project.

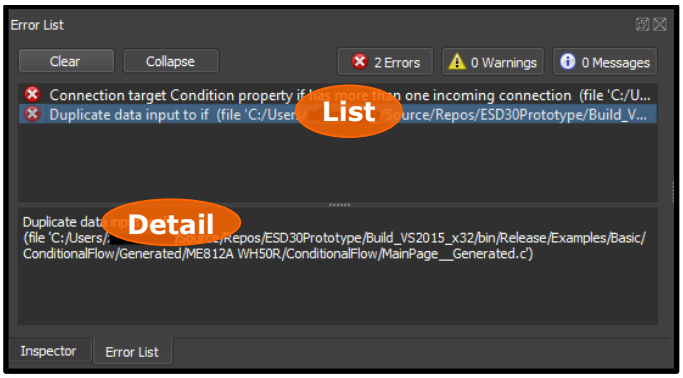


Figure 69 - Error List Window

Copyright © Bridgetek Pte Ltd

Output

The *Output* window is a docked panel that displays messages from the EVE emulator and the toolchain.

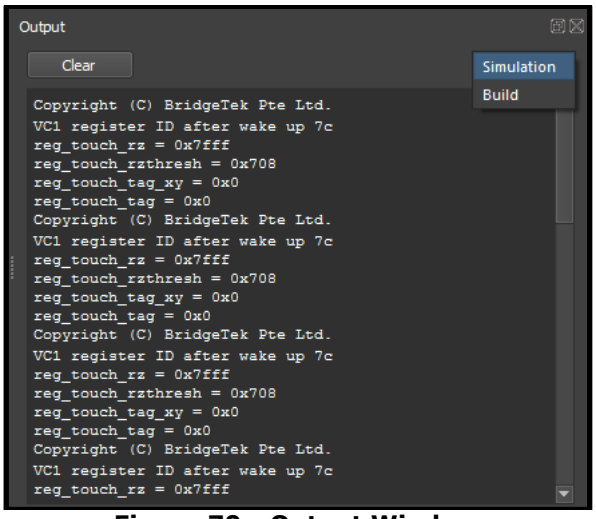


Figure 70 - Output Window

To view messages from the EVE emulator, select **"Simulation"** from the drop-down list. To view messages from the EVE toolchain during source code generation, select **"Build"** from the drop-down list. This window is automatically updated during simulation or build process.

Property Editor

The *Property Editor* allows users to edit the properties of selected logic nodes. The sample screenshot below shows the Property Editor for an ESD Push Button.

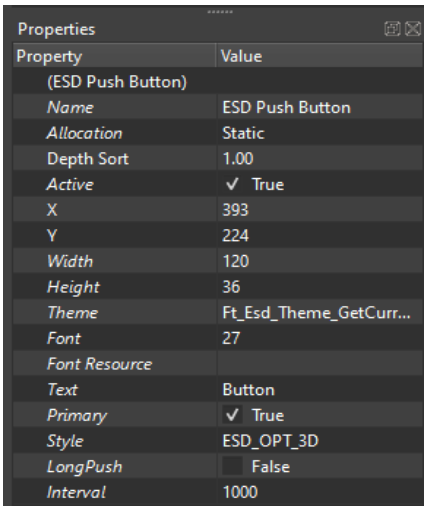


Figure 71 - Property Editor (ESD Push Button)

Copyright © Bridgetek Pte Ltd

Status bar

The *Status Bar,* located at the bottom of the user interface, displays the current status of ongoing processes or tasks. For example, when performing a C compilation, it shows the compilation status of the generated C code.

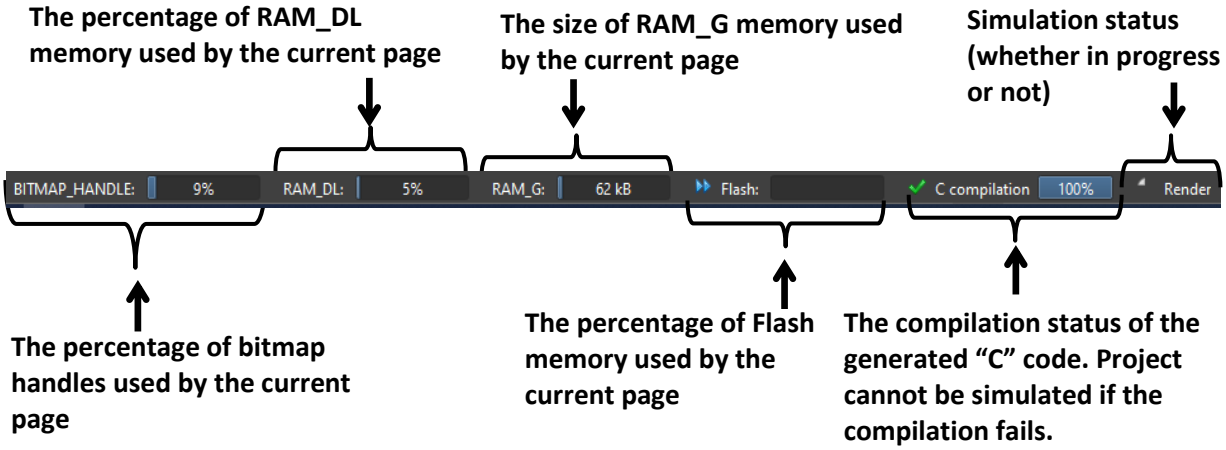


Figure 72 - Status bar

Explanation: Why does the available RAM_G on the VM820C platform vary with resolution and swapchain buffer format?

General formula:

RAM_G available = RAM_G total – RAM_G_RESERVED – Swapchain buffer size

- Total RAM_G: 128 MB
- RAM_G_RESERVED: 2.5 MB reserved for BT820
- Swapchain buffers: Two framebuffers allocated based on resolution and format

In **EVE_Util.c**, the available RAM_G is computed using the above formula.

```
config->SwapchainFormat = RGB8;
uint32_t endAddr = phost->RamGSize - RAM_G_RESERVED;
int swapchainFormat = config->SwapchainFormat;
switch (swapchainFormat)
{
case RGB6:
    scanoutSrcStride = ((config->Width * 9) + 3) >> 2;
    break;
case YCBCR:
    scanoutSrcStride = config->Width;
    break;
default:
    swapchainFormat = RGB8;
case RGB8:
    scanoutSrcStride = config->Width * 3;
    break;
}
```


Copyright © Bridgetek Pte Ltd

```
int32_t bufferSize = scanoutSrcStride * config->Height;
eve_printf_debug("Swapchain format: %i, stride: %i, buffer size: 2x %i
bytes\n", swapchainFormat, scanoutSrcStride, bufferSize);
uint32_t buffer0 = endAddr - bufferSize;
uint32_t buffer1 = buffer0 - bufferSize;
phost->RamGAvailable = buffer1;
```

Resolution **1920 × 1200** and format **RGB8**:

- Buffer size: Width x 3 x Height ≈ 6.9 MB *2 buffers ≈ 13.8 MB.
- **RAM_G available** = 128 MB(Total) – 2.5 MB(reserved) – 13.8 MB(buffer) ≈ 112 MB.

Resolution **800 × 480** and format **RGB8**:

- Buffer size: Width x 3 x Height ≈ 1.15 MB * 2 buffers ≈ 2.3 MB.
- **RAM_G available** = 128 MB – 2.5 MB – 2.3 MB ≈ 123 MB.

Resolution **1920 × 1200** and format **YCBCR**:

- Buffer size: Width x Height ≈ 2.3 MB * 2 buffers ≈ 4.6 MB.
- **RAM_G available** = 128 MB – 2.5 MB – 4.6 MB ≈ 120.9 MB.

Copyright © Bridgetek Pte Ltd

Application Project Structure

An application consists of one or more pages, and each page can contain one or more widgets. Widgets and pages are connected via connection lines, which define the screen logic.

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

By design, all widgets and pages are fully modular and self-contained. Any interaction between the widgets and pages must occur through the hierarchical chain, meaning that node connections exist only between the directly connected nodes within the hierarchy.

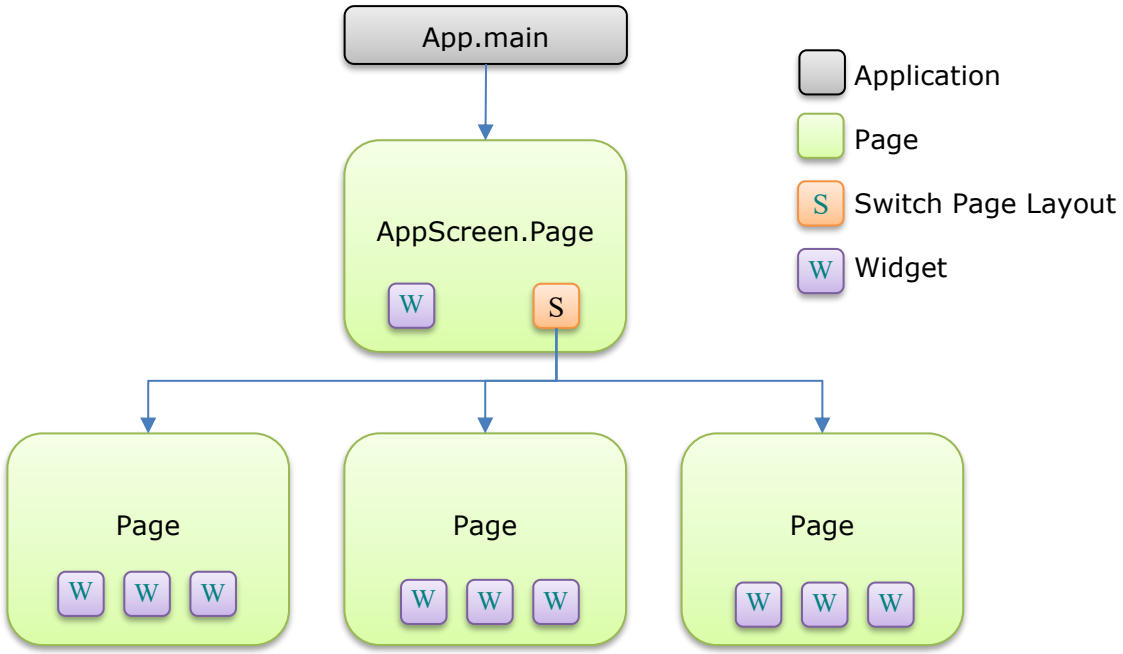


Figure 73 - Application Project Structure

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

Copyright © Bridgetek Pte Ltd

*.png, *.jpg, *.jpeg	Image resource files
*.anim4esd	Animation file converted by EAB tool 2.13 version
*.anim.reloc	Relocatable animation file converted by EAB tool 4.x version

Table 10 - Project File Type Description

Note: The file names listed above must **not conflict** with C standard library function or Windows header files. Examples include : *fopen, windows, winbase, math, printf, realloc.* If users assign these reserved names to widgets, pages, logics, images, or any other resources, the ESD simulation process may fail and produce errors. To prevent this, ESD enforces the 'Usr_' prefix for all new or renamed widgets, pages, functions, actors, source files, themes, and logic files. This ensures that the simulation process runs successfully without errors.

Application Workflow

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

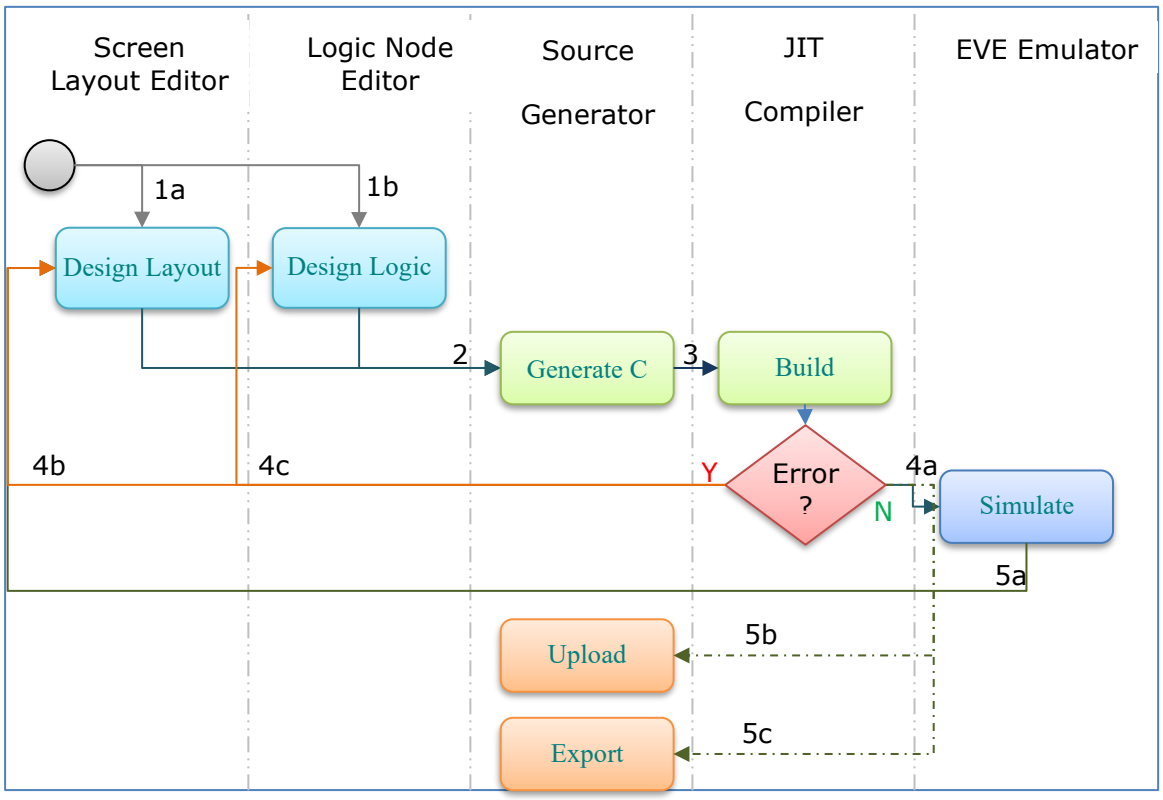


Figure 74 - Application Workflow

The application workflow consists of 5 main modules: the *Screen Layout Editor, Logic Node Editor, Source Generator, JIT Compiler(TinyCC)* and *EVE Emulator*. Among these, the *Source Generator*, *JIT Compiler* and *EVE Emulator* function as back-end modules with have no user-facing interface. The compiler is a JIT-style C compiler – specifically *TinyCC* , which compiles and links the generated code into an executable. Thi executable is then invoked by the emulator module for simulation.

Copyright © Bridgetek Pte Ltd

The application workflow begins with either designing the layout (1a) or defining the logic (1b). When the user saves the changes, it triggers C code generation (2). The compiler then attempts to build the generated C code; if any errors are encountered during build process, the user must return to either design layout (4b) or design logic (4c) to make corrections. If no errors are found, the emulator proceeds to simulate the project (4a). Once the simulation is successful, users can either upload the binary to the EVE chips(5b) or export the project as an Eclipse C project (5c).

Design Layout using Layout Editor

Users need to determine the number of pages in the project and the widgets to be included on each page. After creating a page, users can drag and drop widgets in the Screen Layout Editor to design its appearance. This process defines how many screens are included in the project and how they will appear overall.

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 requirements, users may define and design custom widgets.

Add Bitmap Resource

To display a bitmap on the page, users need to add a bitmap resource to the project. The following steps 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***).

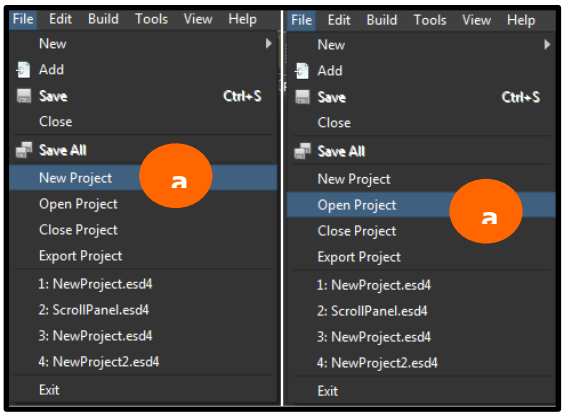


Figure 75 - New and Open Project Menu

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

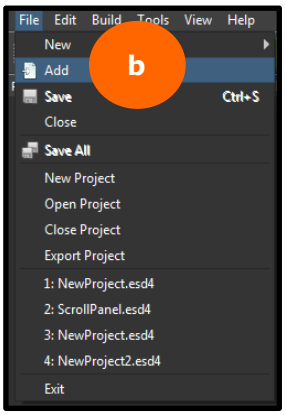


Figure 76 - Add Menu

Copyright © Bridgetek Pte Ltd

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

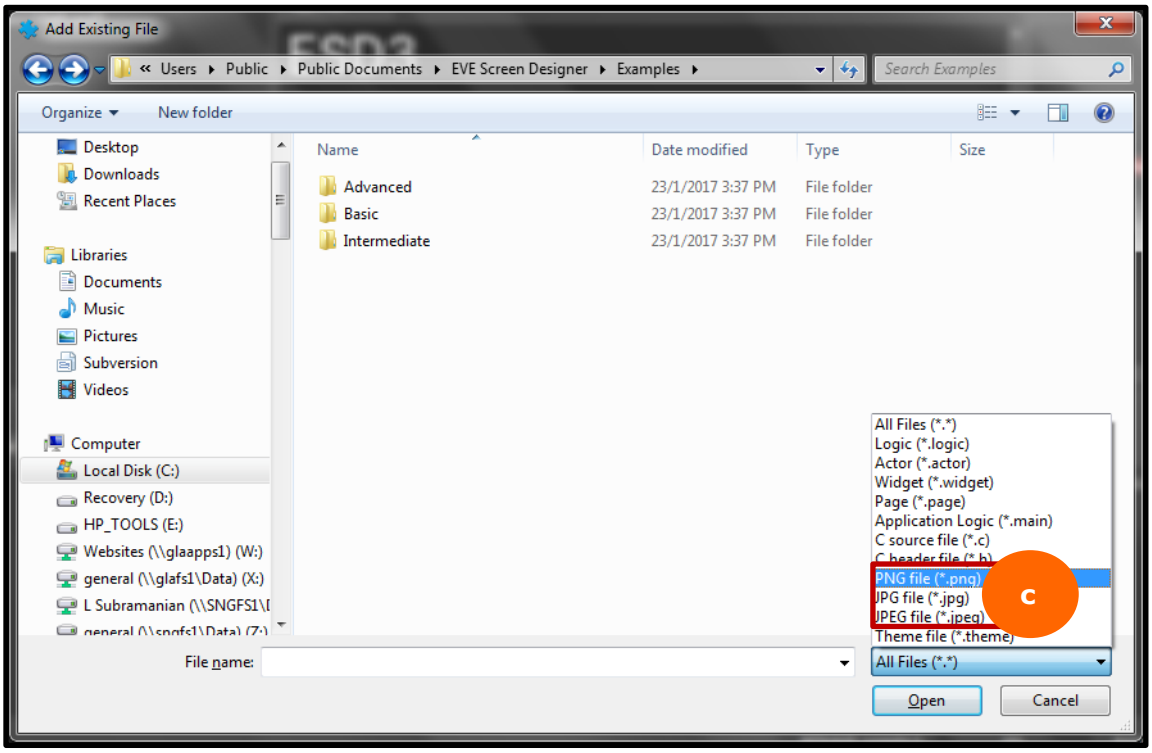


Figure 77 - Add Existing file dialog

After adding the file, the Bitmap Converter dialog appears, allowing users to adjust the conversion settings before importing the bitmap.

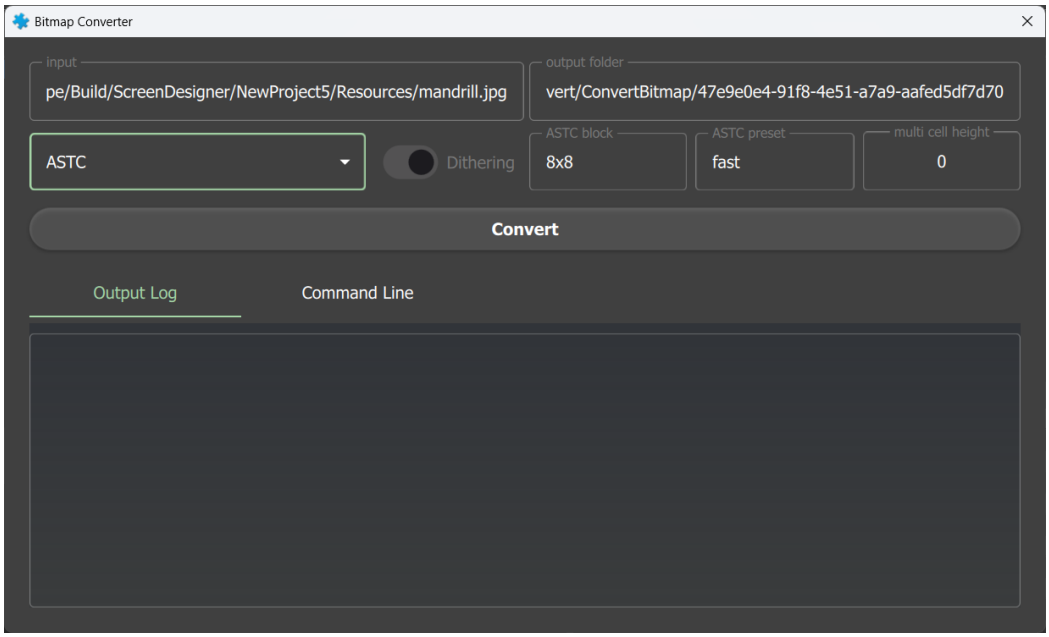


Figure 78 - Bitmap Converter dialog

Input: Displays the full path of the selected input font and is automatically populated after browsing.

Copyright © Bridgetek Pte Ltd

Output Folder: Specifies the folder where the converted bitmap will be saved. This path is automatically generated by ESD.

Bitmap Format: The output format of images, is one of the following:

BT82X:

- ARGB6, ARGB8, RGB6, RGB8
- LA1, LA2, LA4, LA8
- PALETTEDARGB8
- ASTC
- YCBCR

BT81X / FT81X / BT88X:

- ASTC
- ARGB1555, ARGB4, ARGB2
- RGB565, RGB232
- PALETTED565, PALETTED4444, PALETTED8
- L1, L2, L4, L8
- DXT1_L1_RGB565, DXT1_L1_PALETTED565
- DXT1_L2_RGB565, DXT1_L2_PALETTED565

FT80X:

- ARGB1555, ARGB4, ARGB2
- RGB565, RGB332
- L1, L4, L8
- PALETTED
- DXT1_L1_RGB565

ASTC Preset/Block

ASTC Extra Params: Allows users to explore diverse options to achieve the optimal balance based on specific requirements, such as memory usage, performance demands, and visual quality.

Cell Height: Adds padding data to ensure that each bitmap cell aligns to 64 bytes in memory, enabling proper rendering in ASTC format on the EVE chip. Setting a value greater than 0 enables this feature.

Dithering: When enabled, dithering can help reduce noise levels; however, it may negatively affect compression efficiency. Note that ASTC, YCBCR, and PALETTEDARGB8 bitmap formats do not support dithering.

Once the file conversion is complete, users can drag and drop the ESD Image widget onto the desired page and select the required bitmap cell (default is cell 0).

Copyright © Bridgetek Pte Ltd

Configure Bitmap Resource

Users can configure a bitmap resource using the *Property Editor*. Select the image file from the Project Browser and view its settings in the Property Editor.

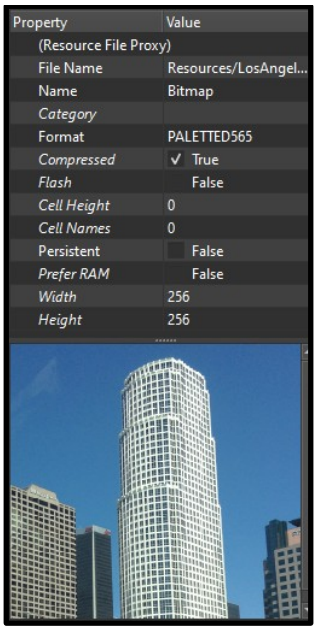


Figure 79 - Property Editor

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 • LA1/LA2/LA4/LA8 (BT82X) • RGB332/RGB565 • RGB6/RGB8 (BT82X) • ARGB1555/ARGB2/ARGB4 • ARGB6/ARGB8 (BT82X) • PALETTED4444/PALETTED565/PALETTED8 • PALETTEDARGB8 (BT82X) • 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 • YCBCR (BT82X)

Copyright © Bridgetek Pte Ltd

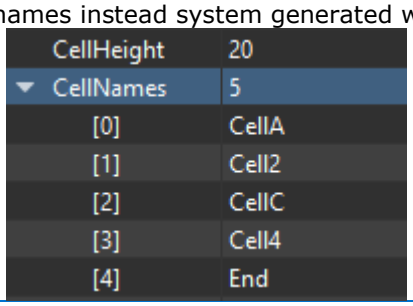
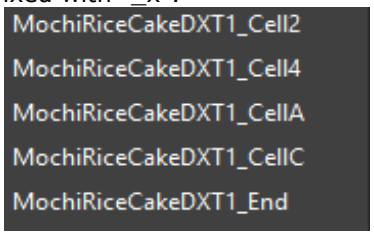
Compressed	Set bitmap compression on/off
Flash	Set this to store the resource in the Flash
CellHeight	The pixel number of one cell. For example, if the bitmap width is 240 and height is 240, then it can be defined as two cells with "CellHeight" - 120 or four cells with "CellHeight" – 60. Cell Height 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".  
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 11 - Bitmap Resource Properties

Design Screen Logic using Logic Node Editor

Users can define the dynamic behaviour of an application at this stage. The Logic Node Editor uses an innovative visual programming approach to allows users to define logic flows without coding.

At this stage, users can define both intra-page and inter-page behaviour. For intra-page behaviour, widgets on a page can be connected using the Logic Node Editor, with additional logic nodes from the Library Browser if required.

Inter-page logic is defined in the application logic, typically named as "AppScreen.page". Users drag and drop predefined pages into the Logic Node Editor and connect them using the logic interface. By linking pages in this way, behaviour that spans across page boundaries, such as screen transitions, is effectively defined.

Copyright © Bridgetek Pte Ltd

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 determine the memory usage before uploading the image to hardware. In debug mode, several useful messages are printed to assist users in making this decision.

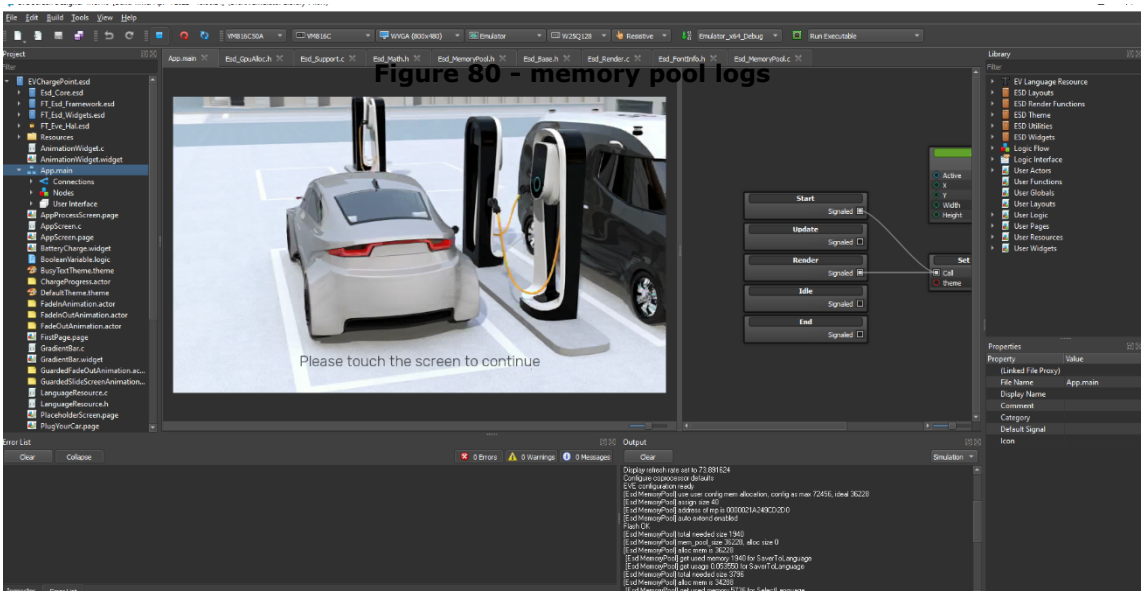


Figure 80 - memory pool logs

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

Simulation

To validate the screen logic behaviour, users may simulate the project on a PC before compiling and downloading the generated C code to the target device. When the application logic (AppScreen.page or App.main) is selected, users can simulate the entire project by clicking **[Simulation]** button on the toolbar. In simulation mode, users cannot drag and drop widgets in the layout editor. Instead, the PC mouse functions as a touch stylus, allowing direct interaction with the application as it would behave on a touch panel.

To validate the behaviour of a single page, users must select the page file and open it in the Logic Node Editor before clicking the **[Simulation]** button. Similarly, other logic nodes such as widgets or actors can be simulated by opening them in the Logic Node Editor and initiating simulation via the same button.

Before running a simulation, it is strongly recommended to save the current project using the **[Save all]** button. Once the simulation is complete, users should disable **[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.

Table 12 - Simulation Description

Copyright © Bridgetek Pte Ltd

Target selection

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

1. IDK-FT810-43A
2. IDK-BT816-50A
3. IDK-BT817-70A
4. IDK-BT817-101A
5. IDK-BT820-101A
6. ME810A-HV35R
7. ME812A WH50R
8. ME812AU-WH50R
9. ME813A-WH50C
10. ME813AU WH50C
11. VM816C50A
12. VM820B15A
13. VM820B10A
14. STM32F4
15. Gameduino 3X Dazzler for Pico
16. IDM2040-7A
17. IDP-1000-04A
18. IDP-5000-04A
19. IDP-4000-04A
20. IDP-3500-04PA
21. IDM2040-43A
22. IDM2040-21R

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 (800x480) and Flash set to 16MB MX25L16.Refer to Figure 81 - New Target button and Figure 82 for more details.

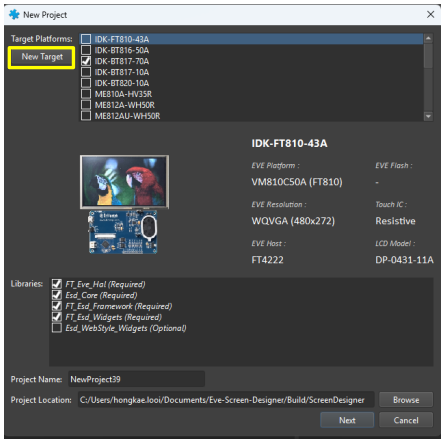


Figure 81 - New Target button

Copyright © Bridgetek Pte Ltd

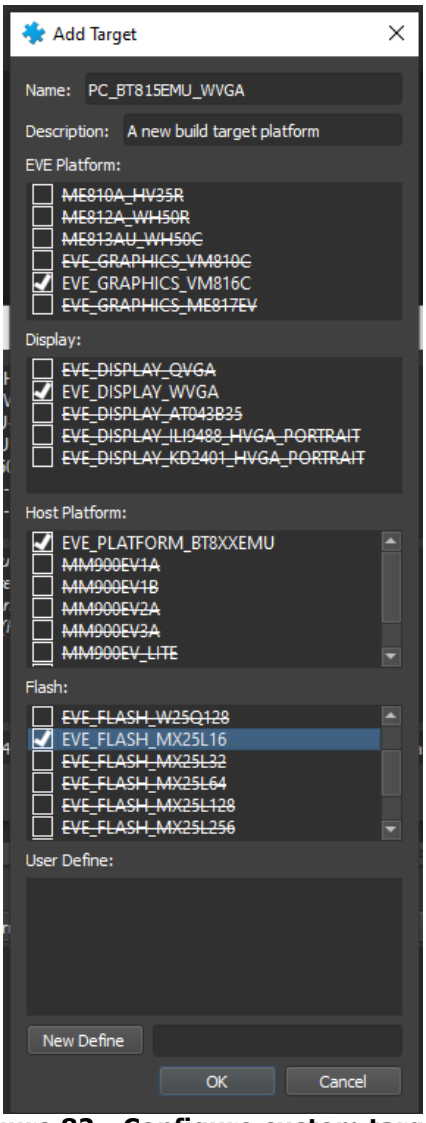


Figure 82 - Configure custom target

The newly created target can then be selected as the active build target. Refer to the screenshot below:

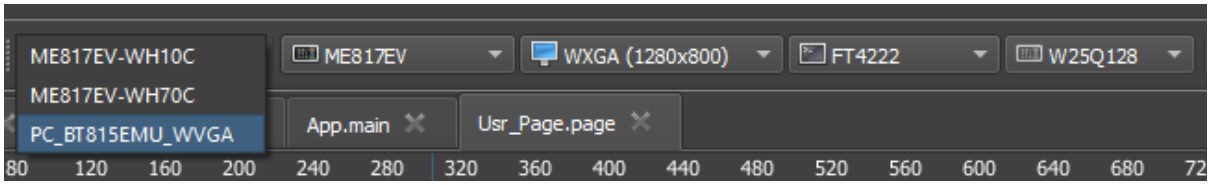


Figure 83 - Select custom build target

Copyright © Bridgetek Pte Ltd

Build & Upload

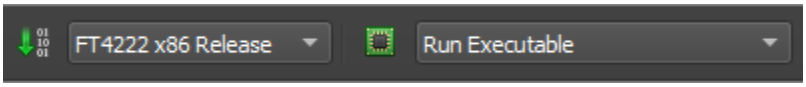


Figure 84 - Build & Upload

The "**Build Executable**" button builds the project.

The "**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.

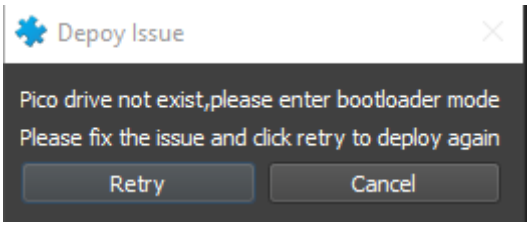


Figure 85 - Pico drive not connected alert

The respective output files generated by "Build Executable" option for different build configurations and host platforms are shown in Table 13.

Build Target	Host Platform	Output files
IDK-FT810-43A	X64 toolchain (FT4222)	*.exe
IDK-BT816-50A	X64 toolchain (FT4222)	*.exe
IDK-BT817-70A	X64 toolchain (FT4222)	*.exe
	FT90x toolchain	*.bin; *.elf
	Pico toolchain	*.uf2
	STM32 toolchain	*.elf; *.bin
IDK-BT817-101A	X64 toolchain (FT4222)	*.exe
	FT90x toolchain	*.bin; *.elf
	Pico toolchain	*.uf2
	STM32 toolchain	*.elf; *.bin
IDK-BT820-101A	STM32 toolchain	*.elf; *.bin
	Pico toolchain	*.uf2
	X64 toolchain (FT4222)	*.exe
ME810A-HV35R	FT90x toolchain	*.bin; *.elf
	Pico toolchain	*.uf2
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
VM816C50A	X64 toolchain (MPSSE)	*.exe
VM820B15A	X64 toolchain (FT4222)	*.exe

Copyright © Bridgetek Pte Ltd

VM820B10A	X64 toolchain (FT4222)	*.exe
STM32F4	STM32 toolchain	*.elf; *.bin
Gameduino 3X Dazzler for Pico	Pico toolchain	*.uf2
IDM2040EV-7A	Pico toolchain	*.uf2
IDP-1000-04A	Pico toolchain	*.uf2
IDP-4000-04A	Pico toolchain	*.uf2
IDP-5000-04A	FT90x toolchain	*.bin; *.elf
IDP-3500-04A	FT90x toolchain	*.bin; *.elf
IDM2040EV-43A	Pico toolchain	*.uf2

Table 13 - Output files for different Build Configuration & Host Platforms

Export

Upon completing the screen design, the project can be moved to the next phase of the workflow – exporting the project. There are four export options: **"Export as Eclipse Project", "Export as MSVC Project", "Export as STM32CubeIDE Project"** and **"Export as Pico Project"**. This function prepares all necessary files for the MCU tool chain, enabling further enhancements such as integrating sensors or external hardware into a complete HMI solution.

During export, ESD copies the generated "C" source code, required bitmap resources, and the ESD application framework code into a user-defined folder. In ESD, all generated files are named in the format "*_generated.c".

For eclipse export projects, ".cproject" and ".project" files are generated, users to open the project in the FT90X eclipse IDE[3]. For MSVC export projects, a "Project" folder is created containing an MSVC-compatible project structure. For Pico export projects, "CMakeLists.txt" and "pico_sdk_import.cmake" are generated to support building the project. Additionally, an optional batch file "pico_build.cmd", is generated to allow a single-step build process. To use this script, the system environment variable PICO_SDK_PATH must be set to the location of the Pico SDK.

Another script "pico_flash.cmd" is generated only if the ESD project contains flash assets. Executing this script flashes both the application and flash image to the Pico board, which must be placed in flash mode beforehand. Table 14 provides more information on the build targets, host platforms, and corresponding export projects that can be created.

Copyright © Bridgetek Pte Ltd

Build Target	Host Platform	Export Project
IDK-FT810-43A	Emulator, FT4222	MSVC Project
IDK-BT816-50A	Emulator, FT4222	MSVC Project
IDK-BT817-70A	Emulator, FT4222	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
	MM2040EV	Pico Project
	STM32F401RET	STM32 Project
IDK-BT817-101A	Emulator, FT4222	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
	MM2040EV	Pico Project
	STM32F401RET	STM32 Project
IDK-BT820-101A	Emulator, FT4222	MSVC Project
	MM2040EV	Pico Project
	STM32F401RET	STM32 Project
ME810A-HV35R	Emulator	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
	MM2040EV	Pico 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
VM816C50A	Emulator, MPSSE*	MSVC Project
VM820B15A	Emulator, FT4222	MSVC Project
VM820B10A	Emulator, FT4222	MSVC Project
STM32F4	STM32F401RET	STM32 Project
Gameduino 3X dazzler	Emulator	MSVC Project
	Raspberry pi Pico	Pico Project
IDM2040EV-7A IDM2040EV-43A IDM2040EV-21R IDP-1000-04A IDP-4000-04A	Emulator	MSVC Project
	RP2040	Pico Project
IDP-5000-04A IDP-3500-04A	Emulator	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project

Table 14 - Build Targets, Host platform, and the corresponding export projects

(* Requires separate MPSSE adapter or FT4222H circuit)

Copyright © Bridgetek Pte Ltd

Exported Folder Structure

Table 15 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 15 - List of Folders & Files created upon Exporting Project

Copyright © Bridgetek Pte Ltd

Assets

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

Bitmap Resources

The bitmap resources used in an ESD project can be stored on a SD card or in flash, depending on the flash flag configuration set in ESD.

.astc : The ASTC Bitmap Format for images. The corresponding .astc file must be copied to either the SD card or the flash directory.

.z : The .z bitmap format used for compressed images. The .z file must also be copied to either the SD card or the flash directory.

SD card

If bitmap resources intended for the SD card are used in a project, they are converted into *.astc (for ASTC file format), *.raw (for other file formats), or *.bin (if the compression flag is enabled). Users must download the converted bitmap resources to the **root** directory of the SD card and then 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.

Copyright © Bridgetek Pte Ltd

EVE SD card support on the VM820C platform

BT82X supports SD cards managed directly by the EVE chip, rather than the host MCU. The usage remains similar to that of host-managed SD card:

1. Add resource to your project.
2. Set the Resource type to **ESD_RESOURCE SD** to store the resource on the EVE SD card.

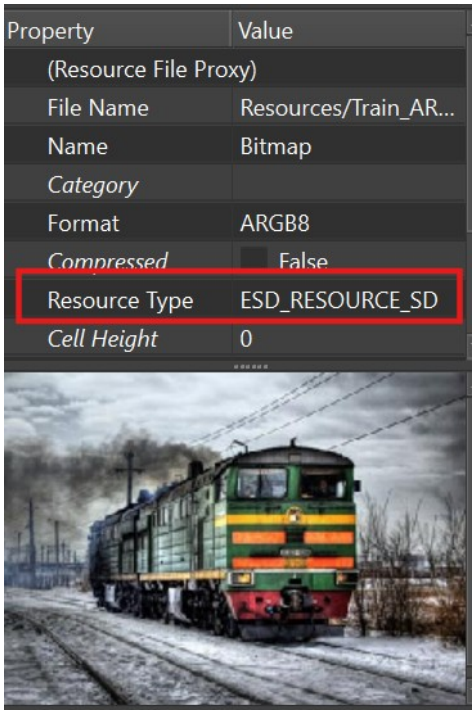


Figure 86 - Store resource in EVE SD card

3. Copy the converted resources to the SD card and insert it into the EVE SD card slot on the hardware module.

ESD supports all BT82X-compatible bitmap formats, AVI video files, custom fonts, and relocatable animation files (.anim.reloc).

The converted resources are located at:
$(ExportFolder)\data

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

Copyright © Bridgetek Pte Ltd



Figure 87 - SD Card BT82X Example project

The example project SDCard-BT82X demonstrates how to use resources stored on an SD card managed by the BT82X. Supported resource types include images, videos, and fonts.

Flash

If bitmap resources intended for flash are used in a project, they are converted into a Flash.bin file. Users must download the converted bitmap resources 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 requiring binary recompilation due to hardcoded asset information, ESD generates additional metadata files with the **.esdm** extension appended to the original filename for each converted resource.

The contents of the metadata file correspond to the Esd_BitmapInfo, Esd_ResourceInfo, Esd_FontInfo, and Esd_AnimationInfo data structures, packed into a fixed format of 64 bytes. This size limitation ensures the metadata can fit within the directory record of the LittleFS filesystem when used with flash storage and also simplifies loading.

Copyright © Bridgetek Pte Ltd

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

Table 16 - Common Resource Metadata Format

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

Table 17 - Bitmap Resource Metadata Format

Copyright © Bridgetek Pte Ltd

Animation Resource Metadata Format:

The animation resource metadata format specifies the address offsets used when the animation is assembled into the flash image. It is currently used only to verify that the animation matches its expected location in flash. During the build process, ESD adjusts these offsets on the fly while assembling the flash image. At present, ESD supports animation playback only 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

Table 18 - Animation Resource Metadata Format

Copyright © Bridgetek Pte Ltd

Font Resource Metadata Format:

Specifies font measurement characteristics that are not included in the font block and is 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	

Table 19 - Font Resource Metadata Format

Layout Editor

The *Layout Editor* allows users to preview both a single page and the entire project. ESD E VE emulator to render an EVE display within the 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

A page represents a single screen which will be rendered. An application consists of at least one page. Each page may have its own life cycle and manages all widgets contained within it. Only widgets can be dragged and dropped into pages in the Page Layout Editor. In contrast, all the nodes from the library can be dragged and dropped into the Logic Node Editor.

"Active" Property

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

Copyright © Bridgetek Pte Ltd

Switch Page

Users can define multiple pages within a project. A *"Switch Page"* layout must be added to manage these pages. User can drag and drop pages from the Project Browser into the Switch Page. The picture below illustrates how the Logic Editor view appears for the Switch Page Logic.

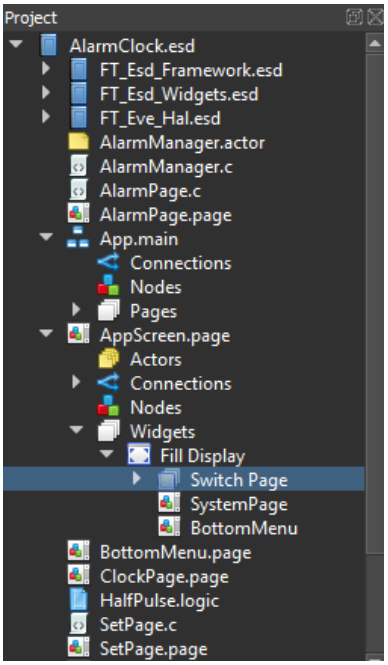


Figure 88 - Switch Page

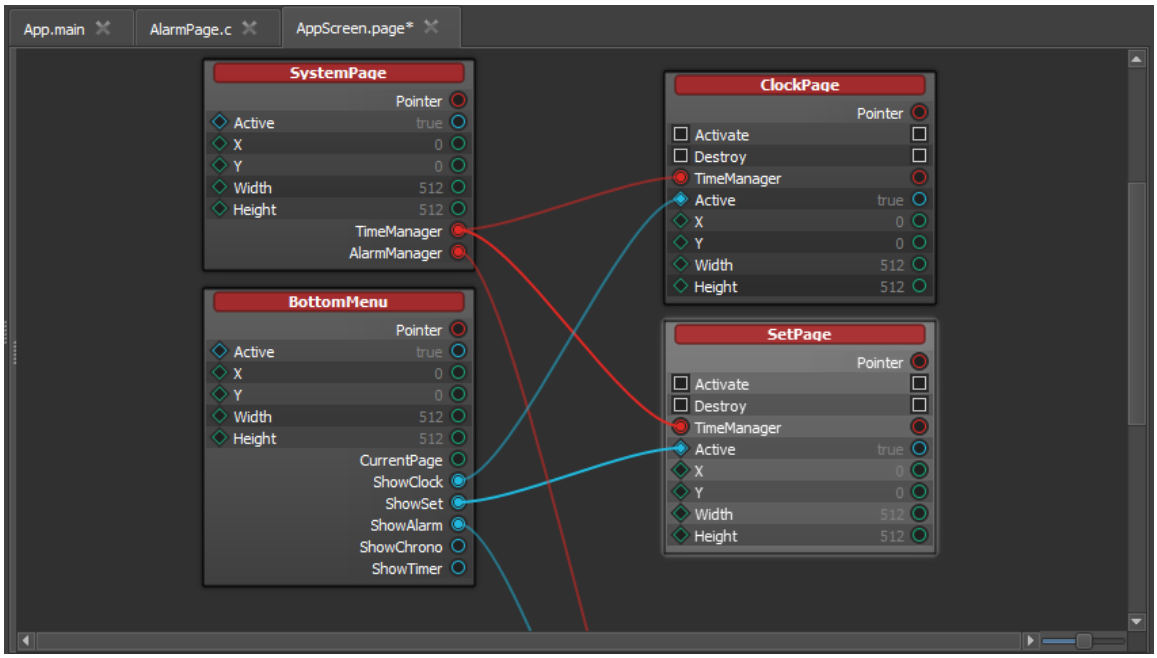


Figure 89 - Sample Switch Page UI

Copyright © Bridgetek Pte Ltd

Page Persistence

To allow a page to remain in memory with its current state even when it is set to inactive, the allocation mode can be set to "Lazy Persistent" or "Static" in the page's Properties under the application logic. This feature enables users to retain page data even when the page not visible.

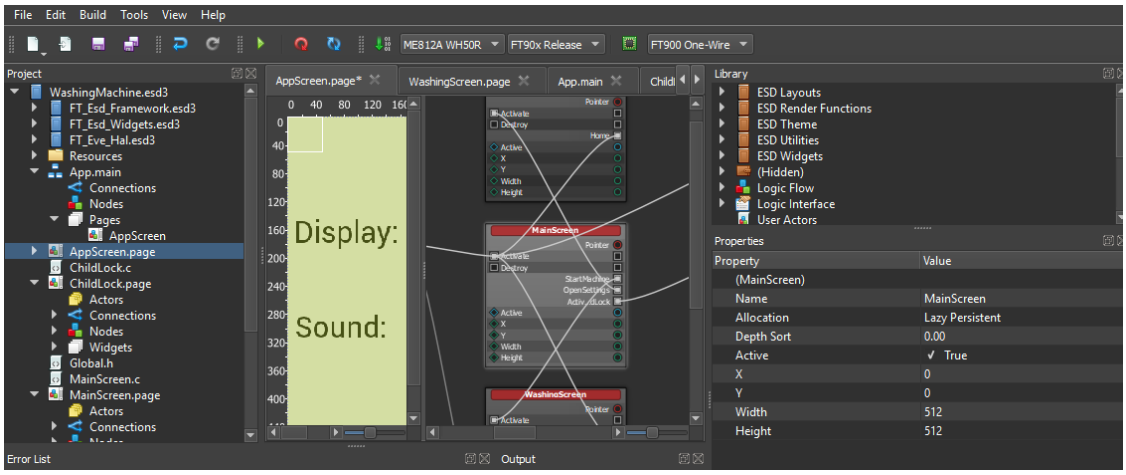


Figure 90 - Properties Editor – Page Allocation

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.

Table 20 - Page Allocation Modes Overview

User Defined Function for Page File

Users can write their own code to handle the "Pushed" signal of the "ESD Push Button" widget. To do so, users must select the "ESD Push Button" widget in the Logic Node Editor and double-click it. A function will then be generated, replacing 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.

Copyright © Bridgetek Pte Ltd

MainPage.page* | App.main | MainPage.c

```
1  #include "Ft_Esd.h"
2  #include "MainPage.h"
3
4  ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context = MainPage)
5  void MainPage_ESD_Push_Button_Pushed(MainPage *context)
6  {
7      // …
8      printf("This push button is pushed\n");
9  }
10
```

Figure 91 - 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 provides *zoom-in* and *zoom-out* functionality, allowing users to view the screen design in greater detail. Refer to the sample picture below:

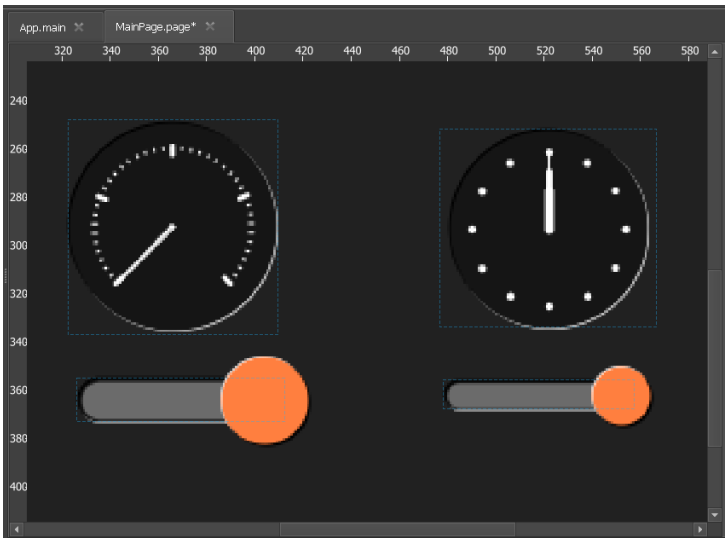


Figure 92 - Zoom In & Out

Copyright © Bridgetek Pte Ltd

Main File

To view the application logic file (*.main), double-click the file in the Project Explorer. The Screen Layout Editor will display the first page defined in the Logic Node Editor. To inspect how the application logic is defined, click the **"Simulation"** button and review it in the Screen Layout Editor.

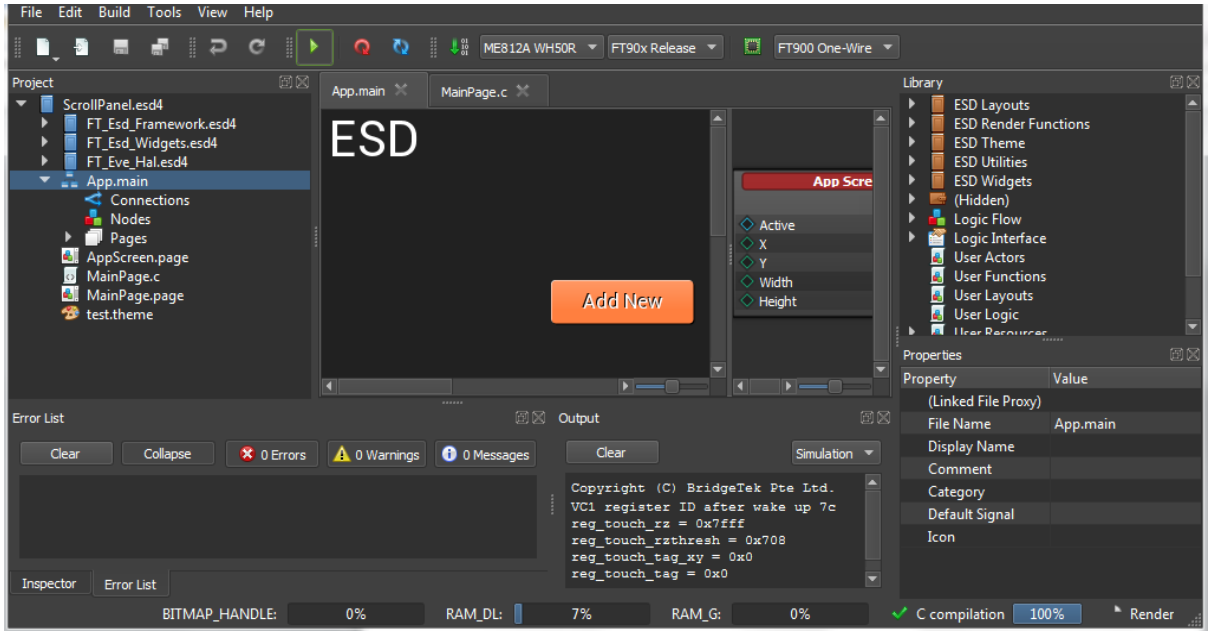


Figure 93 - Main File

Logic File

The logic file contains logic used to encapsulate user-defined behaviour. It functions similarly to a C function and does not render any user interface.

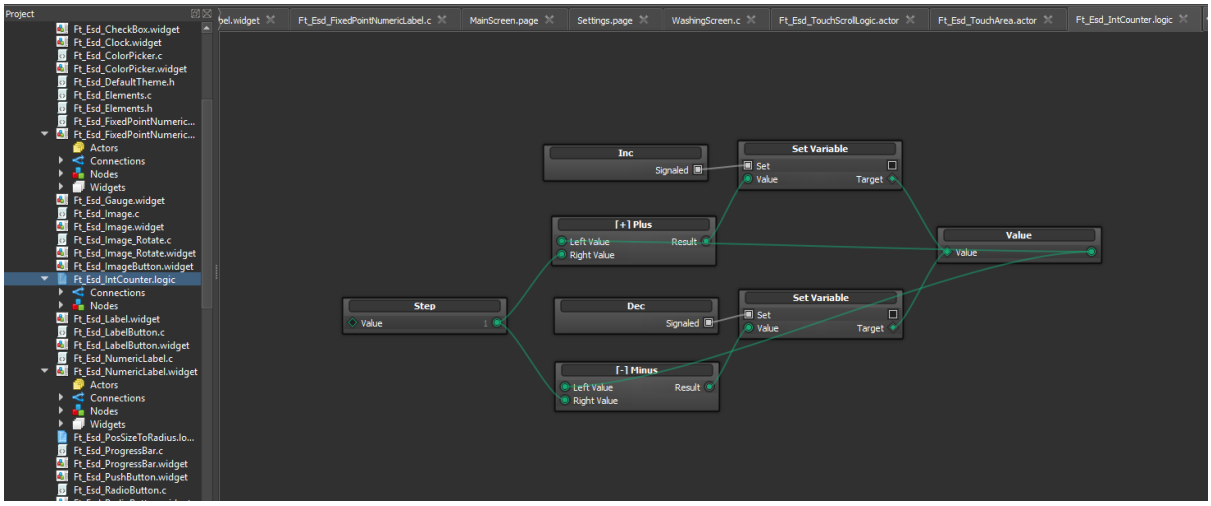


Figure 94 - Sample Logic File

Copyright © Bridgetek Pte Ltd

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 additional features may be added in future updates.

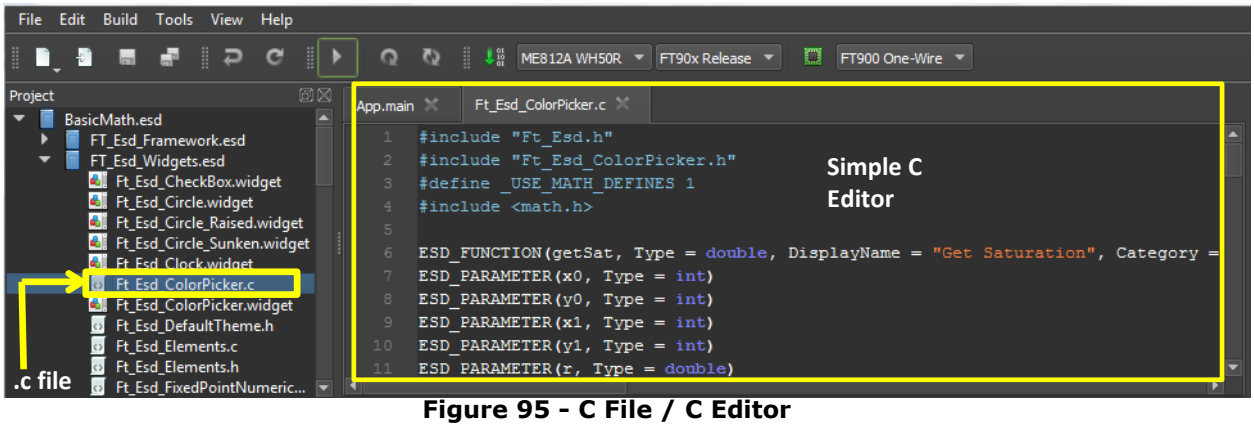


Figure 95 - C File / C Editor



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 provides a set of built-in predefined *basic logic nodes*. These nodes implement fundamental control flow, logic operations, and interface functions. In the logic node editor, a typical basic logic node is displayed as a block, with its name show in White text.

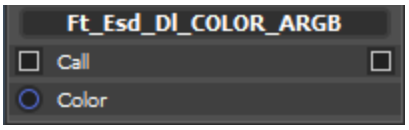


Figure 96 - Logic Node Editor - Basic Logic Node

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

Copyright © Bridgetek Pte Ltd

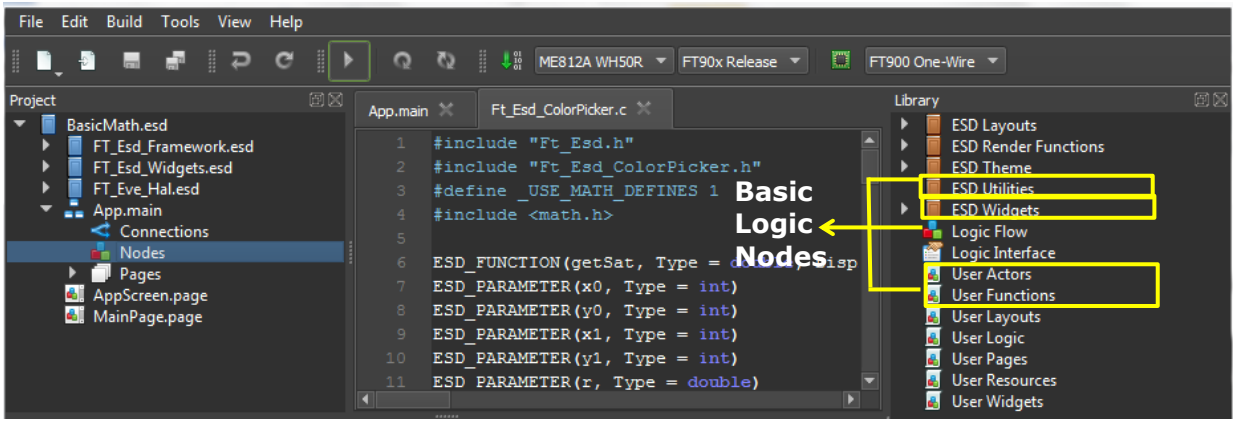


Figure 97 - Library Browser - Basic Logic Nodes

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 -

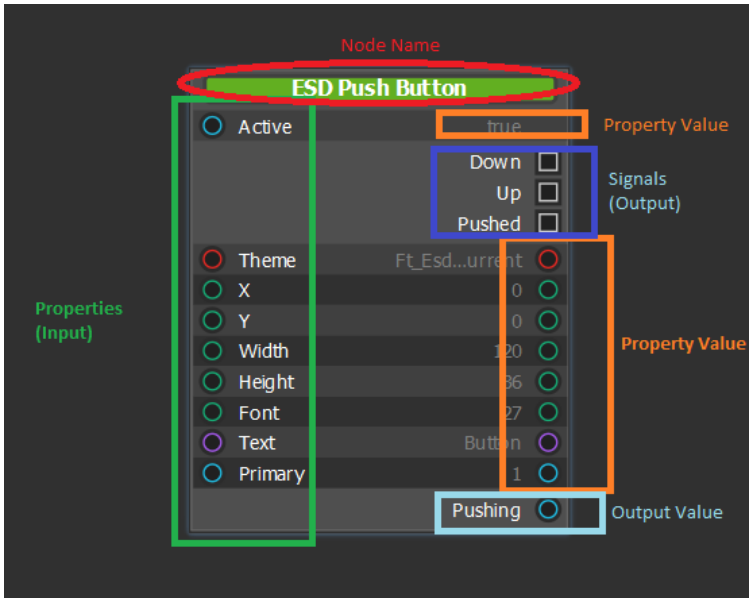


Figure 98 - Composite Logic Node

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*

Copyright © Bridgetek Pte Ltd

Page Node

When users create a single page, it is represented as a logic node under the "User Page" category in the library browser. Each page node can be connected to the other page nodes within the logic node editor, enabling navigation or flow between pages. Additionally, each page node has its own set of properties and defined outputs, which control its behaviour and interactions with other nodes.

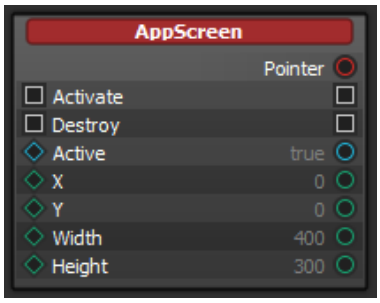


Figure 99 - Composite Logic Node - Page Node

Widget Node

A *Widget Node* is a type of logic node that has a visual representation and can handle user input. It includes built-in slots such as *Start*, *Update*, *Idle*, *Render* and *End*, which define its lifecycle and behaviour.

For more details on these built-in slots, refer to the Built-in Slot section.

In the logic node editor, a widget node is visually distinguished by its name displayed in a green-coloured title bar, as shown in the sample picture below -

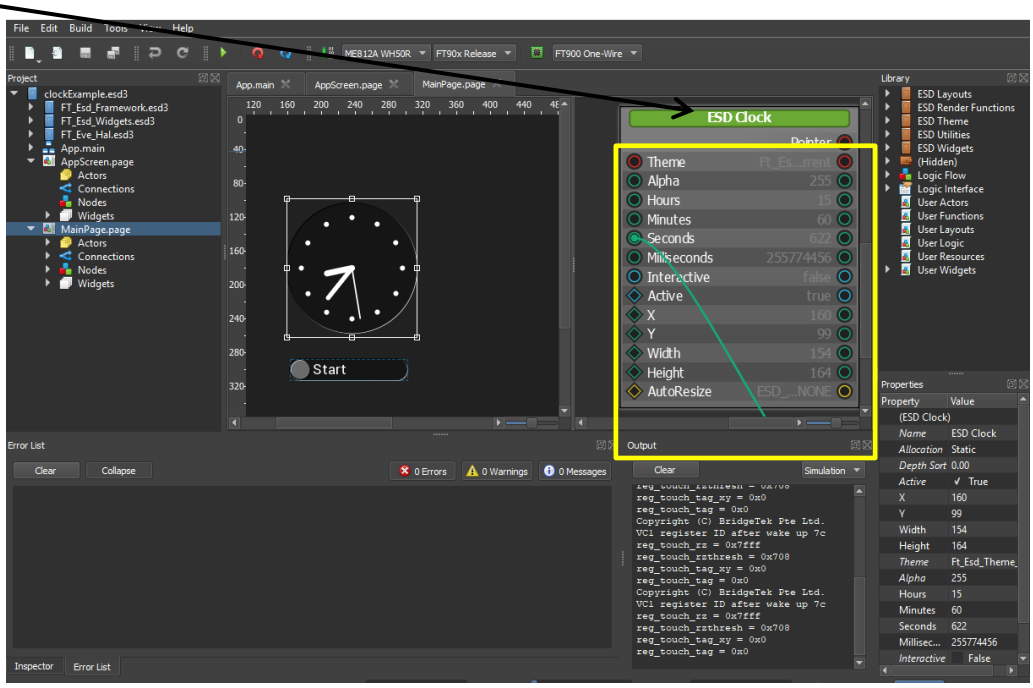


Figure 100 - Logic Node Editor – Widget Node

The highlighted sections provide information on adding user widgets, configuring their position and size properties, rendering the widgets, and applying widget themes.

Copyright © Bridgetek Pte Ltd

Adding User Widgets

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

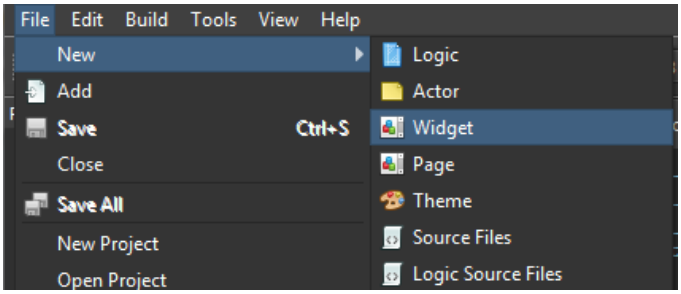


Figure 101 - User Widget

To utilize these slots, users need to drag them from the library browser and drop them into the logic node editor. By renaming the slots to match the pre-defined slot names, users can connect the widget control flow to the overall application control flow.

Position & Size Properties

As user interface elements, widgets must have position and size properties so that users can manipulate them using visible handles. The position is defined by "Input" nodes of type "ft_int16_t", named as "X" and "Y". The size is defined by "Input" nodes of type "ft_int16_t", named as "Width" and "Height". Figure 102 shows an example of the "X" property.

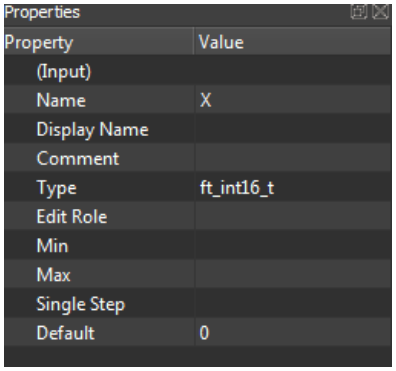


Figure 102 - User Widget - Position & Size Properties

Rendering a widget

The rendering of a widget can be implemented entirely within the logic editor, or partially in code, with the logic editor used to integrate the code with the generated structures.

To create a render function, add a *Slot* from the Logic Interface category in the library browser, rename it to "Render" and double-click it to create the user-defined method for handling rendering. The widget is expected to restore any graphics state it modifies, except for the specific state values listed below.

It is recommended to use the Ft_Esd_Dl_ utility functions for display list drawing whenever possible. These functions internally optimize the display list by duplicating frequently used state parameters, reducing redundancy and improving efficiency. The de-duplicated state values include: *Palette source, Color RGB, Color A, Handle, Cell,* and *Vertex Format*. When using Ft_Esd_Dl_ functions are used, these state values do not need to restore to their original settings. However, they should be considered undefined at the start of the

Copyright © Bridgetek Pte Ltd

rendering function. To access the value of the X input property, refer to the example 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 a property input that allows overriding the default application theme. To achieve this, an input property of type *Ft_Esd_Theme* * should be added by dragging an input node from the "Logic Interface" category in the library browser. Theme colours can then be accessed within the logic editor using the utility functions available under the ESD Theme category, or directly from user code, as shown 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

To handle *touch input*, add a Touch Tag node from the *"ESD Utilities"* category in the library browser to the logic editor. Users can then either handle its signals directly or access its state through the Touch Tag interface.

During the Render function, the "Tag" value provided by the "Touch Tag" node must be used when rendering the relevant portion of the widget. After rendering, the "Tag" values should be reset to 0.

To handle Touch Tag signals in user code, create a function using the ESD_METHOD format, similar to the Render function implementation. Once defined, this function will appear under the User Functions category in the library browser, allowing it to be dragged into the logic editor wo user code function, and it will be available from the User Functions category in the library to drag into the logic editor and connected to the desired signals from the Touch Tag node.

Layout Type Widget

A Layout widget is a layout container for organizing widgets. Unlike widget nodes, which have a visual appearance, a layout widget is used to manage the layout specifications of its child widgets. Layout widgets were introduced since ESD 4.0. Figure 103 & Figure 104 shows the various layout widgets supported in ESD.

Copyright © Bridgetek Pte Ltd

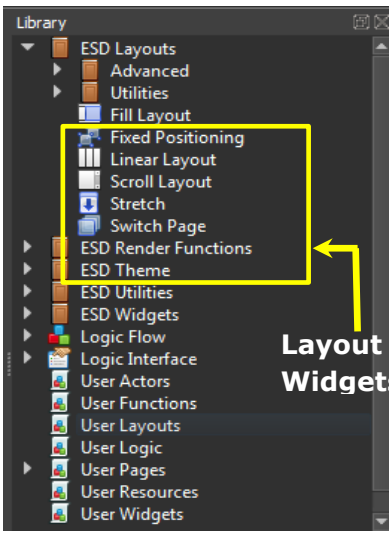


Figure 103 - Layout Widgets

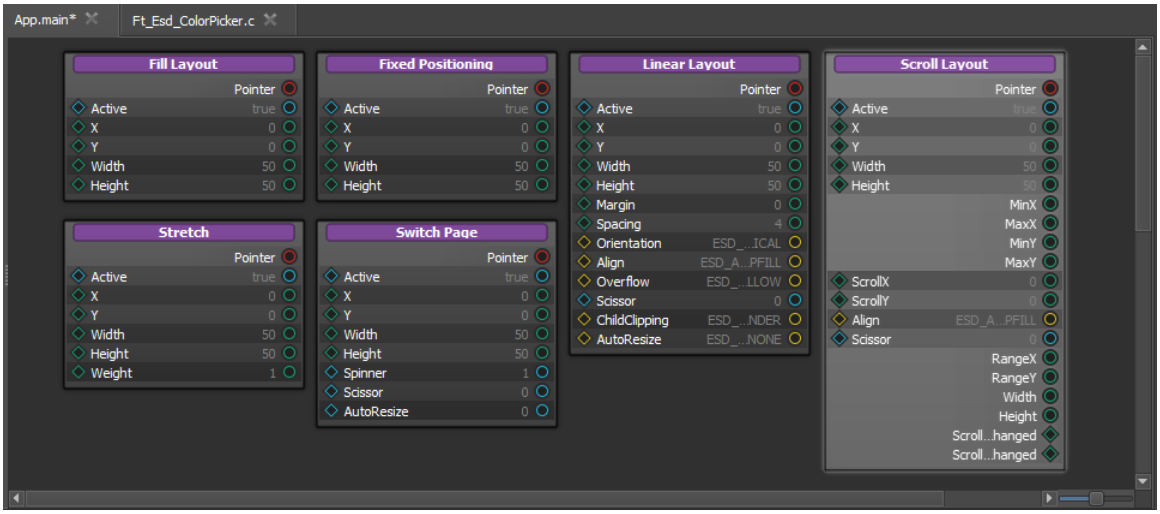


Figure 104 - Layout Widgets

Actor Node

An *Actor* type is an extended form of logic node that functions similarly to a widget node but does not include a *Render* slot. Therefore, it has no visual representation. When a new actor node is created, it does not appear in the layout editor. It includes the built-in slots such as *Start, Update, Idle* and *End,* which are not available in regular logic nodes. For more information on built-in slots, refer to the Built-in Slot section.

Actor nodes are useful for implementing continuous or background behaviours, such as animations, timers, and input data polling. In the logic node editor, an actor node is visually distinguished by its name highlighted in yellow colour.

Copyright © Bridgetek Pte Ltd

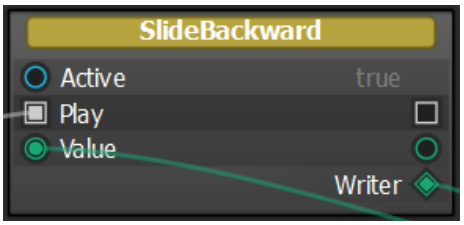


Figure 105 - Actor Node

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

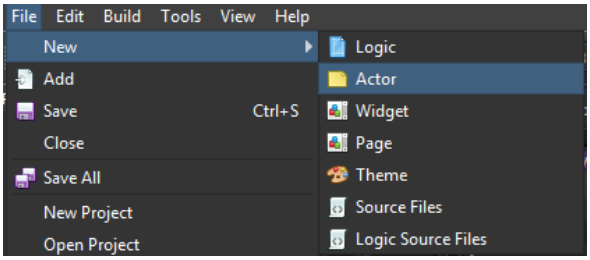


Figure 106 - Actor folder

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 represent specific logic without any built-in slots. As a result, it cannot be invoked through built-in slots such as Render or Update. However, it may define its own private or public properties, outputs, signals, and slots, depending on the users' implementation.

A logic object is typically used to simplify complex logic by breaking it into smaller, more manageable components. It is defined and saved in a "*.logic" file.

In the logic node editor, a logic object is visually distinguished by its name highlighted in grey colour.

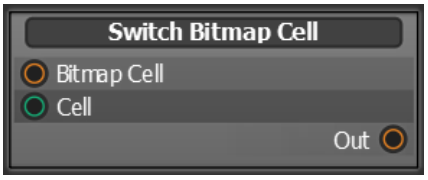


Figure 107 - Logic Object

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

Copyright © Bridgetek Pte Ltd

Connections

Connections are lines drawn between two nodes. The color of a connection line is determined by the types of its endpoints. The following table describes the different types of endpoints and their meanings.

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 21 - Connection End Types

In general, all input properties that support connections can also be set directly in the property editor when using constant values. This is sometimes referred to as the "Default" value. In addition, it is possible to define the Minimum and Maximum constraints for input bindings and variables.

Rendering Order

In the application logic file "App.main", pages added in the logic node editor is represented using the symbols shown in the sample picture.

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

The following implicit rule of the logic node editor is important to note, as it applies to all logic nodes.

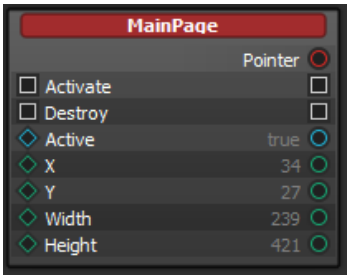


Figure 108 - Main Page



"Depth Sort" is used to determine the rendering order of 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 share the same "Depth Sort" value, the logic node's "X"/ "Y" coordinates (within layout widget) determine the rendering order. In this case, the page node with the lowest "X" or "Y" coordinate is rendered first (i.e. assigned a lower depth value).

Copyright © Bridgetek Pte Ltd

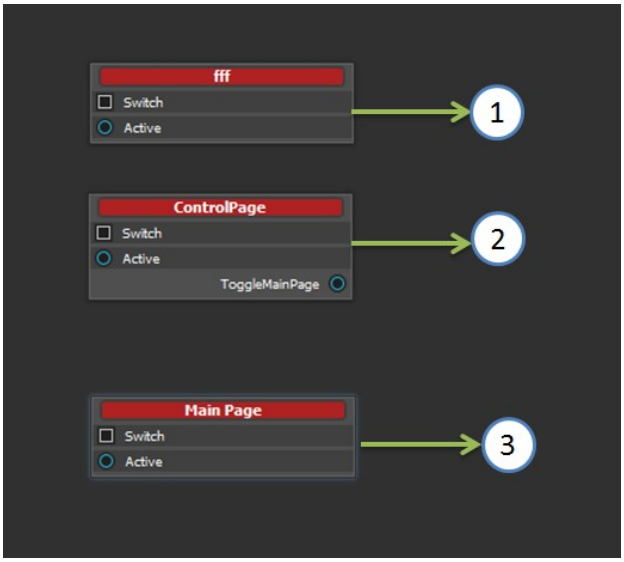


Figure 109 - Rendering Order Example

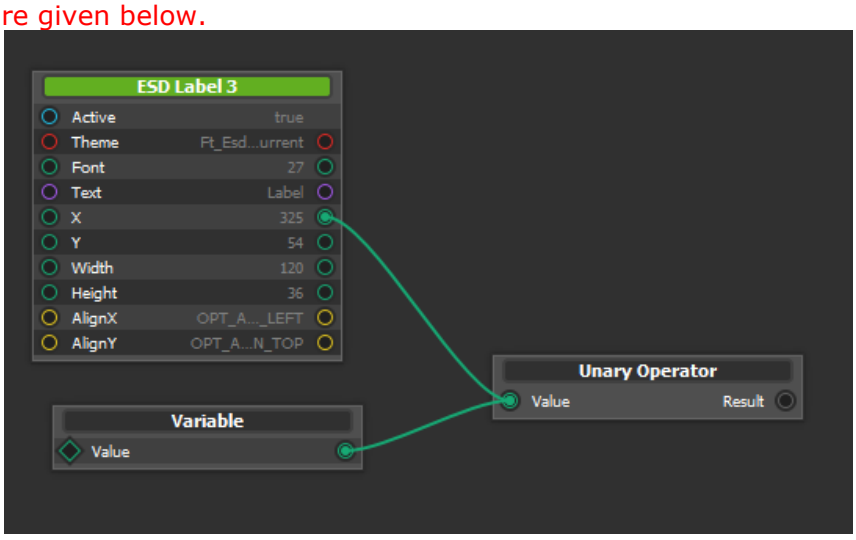
Logic Note Editor – User Interaction

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

"Panning" the Logic Node Editor allows users to view areas of the workspace that do not fit within a single screen. Panning can be performed in two ways:

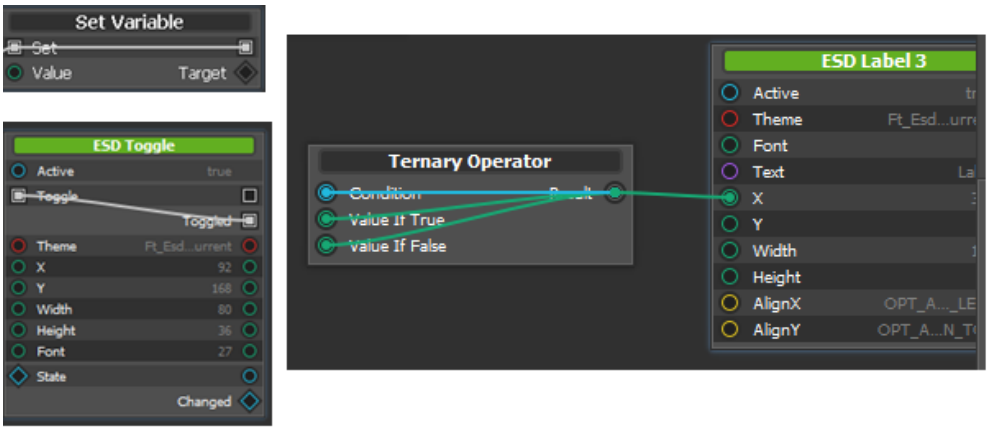
(i) Press *Alt Key + Left mouse button* and move the mouse
(ii) Press the *middle mouse* button and move the mouse.

Users can select multiple node items in the Logic Node Editor by holding the Control key and clicking the desired objects with the left mouse button. Single or multiple objects can then be moved by dragging them while holding the left mouse button.

✕ **DON'TS**	Avoid two connections connecting to the same port. Refer to the sample picture given below. 

Copyright © Bridgetek Pte Ltd

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.



To apply changes made in the Logic Note Editor, ensure that all modifications are saved to the project by selecting **"File → Save All"** from the menu or toolbar. In some cases, it may also be necessary to click the **"Recompile"** button to rebuild the source code generated by ESD.

Library Browser

The *Library Browser* consists of basic components that are predefined and built into ESD 4.14, as well as user-defined components and/or resources. The available components 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 contents displayed in the Library Browser depend on the currently selected node. It only shows items that are relevant to the node currently being edited.

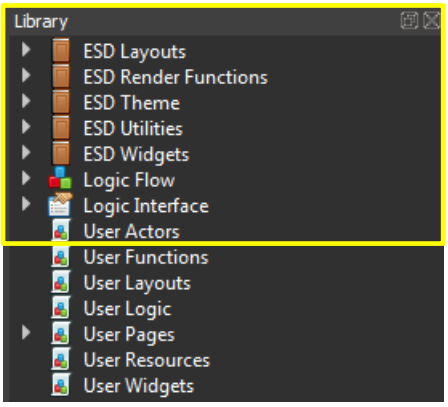


Figure 110 - Library Browser

Copyright © Bridgetek Pte Ltd

In the Library Browser, items can be filtered by entering search text in the textbox. Regular expressions are supported, and matching results are highlighted, as shown in the sample picture below -

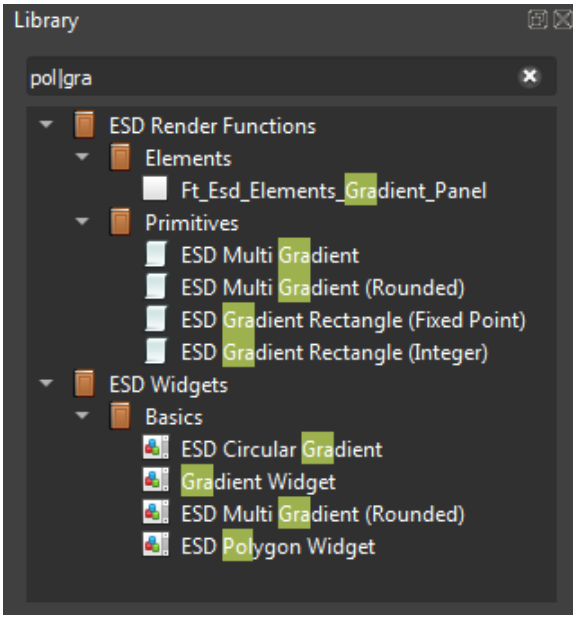


Figure 111 - Library Filter

The forthcoming sections provide information about the different components of the Library Browser.

Copyright © Bridgetek Pte Ltd

ESD Layouts

ESD Layouts contains a collection of ESD layout widgets and layout utility functions.

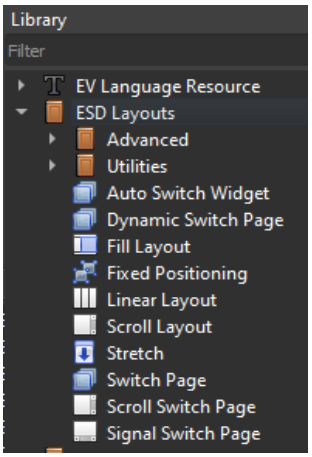


Figure 112 - ESD Layouts

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.

Table 22 - ESD Layouts Description

The ESD Layouts collection also includes layout utilities and advanced layout features. These sections are intended for advanced users only.

Layout utilities are a set of helpers functions designed to simplify layout development. Users can use these utilities to build their own layout widgets. Severa items are provided to support widget sorting and arrangement purposes.

Copyright © Bridgetek Pte Ltd

Utilities

ESD provides various layout utility functions for designing layouts.

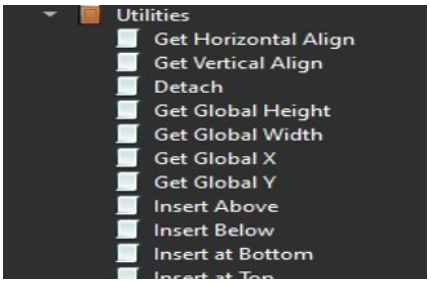


Figure 113 - Utilities Layouts

Get Horizontal Align

The *Get Horizontal Align* utility is used to determine or set the horizontal alignment of a widget.

The function can be called as shown below:

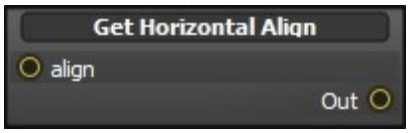


Figure 114 - Get Horizontal Align

```
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 determine or set the vertical alignment of a widget.

The function can be called like below:

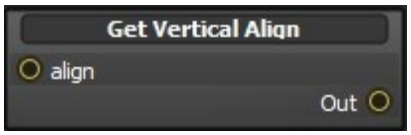


Figure 115 - Get Vertical Align

```
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 remove a widget from its current parent layout.

The function can be called like below:



Figure 116 - Detach

```
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);
```


Copyright © Bridgetek Pte Ltd

Get Global Height

The *Get Global Height* utility is used to obtain the global height of a widget. The function can be called like below:



Figure 117 - Get Global Height

```
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:



Figure 118 - Get Global Width

```
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:

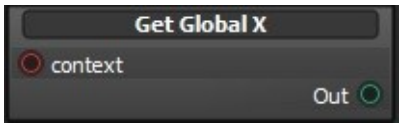


Figure 119 - Get Global X

```
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:

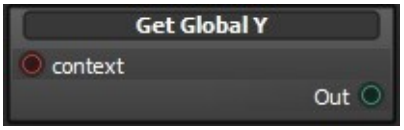


Figure 120 - Get Global Y

```
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)
```


Copyright © Bridgetek Pte Ltd

Insert Above

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



Figure 121 - Insert Above

```
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:



Figure 122 - Insert 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:



Figure 123 - Insert Bottom

```
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);
```


Copyright © Bridgetek Pte Ltd

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:



Figure 124 - Insert at Top

```
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:

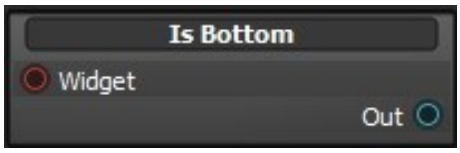


Figure 125 - Is Bottom

```
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:



Figure 126 - Is Top

```
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:

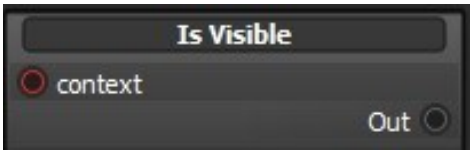


Figure 127 - Is Visible

```
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);
```


Copyright © Bridgetek Pte Ltd

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:



Figure 128 - Sort Highter

```
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:



Figure 129 - Sort Lower

```
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:



Figure 130 - Sort to Bottom

```
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:

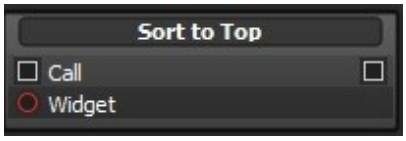


Figure 131 - Sort to Top

```
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);
```


Copyright © Bridgetek Pte Ltd

Switch Page

The *Switch Page* is a special layout that handles page and widget transitions. By default, each ESD project includes one Switch Page layout in "AppScreen.Page". However, this is optional, and users may remove or change the Switch Page layout as needed. For an example of its usage, refer to the sample project: **"EVE Screen Designer → Examples → Basic → MenuPage → MenuPage.esd → AppScreen.page"**.

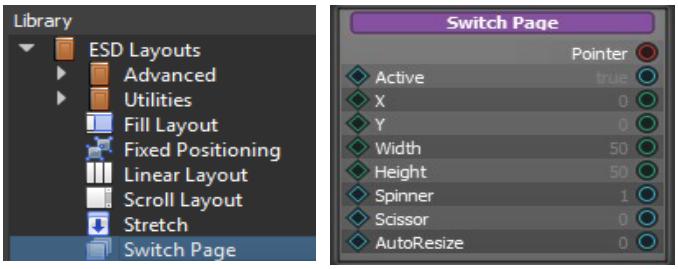


Figure 132 - Switch Page

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 23 - Switch Page Layout Properties

Switch Page Implementation

1. Setting a Page/Widget into Switch Page

Unlike widgets, a Switch Page cannot be dragged into the Screen Layout Editor or Logic Editor. To place a page under the control of the Switch Page Layout, users must use the Project Browser and drag the widget into the Switch Page layout.

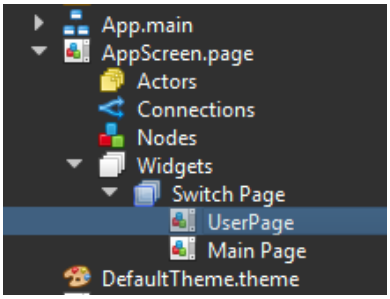


Figure 133 - Switch Page Layout

Copyright © Bridgetek Pte Ltd

2. Page Resource

Since ESD 4.0, page resource allocation can be configured as follows:

- Static – Always load at startup
- Lazy – Allocated when the page becomes active and freed when it becomes inactive.
- Lazy Persistent – Allocated the first time the page becomes active and remains in memory until system reset.

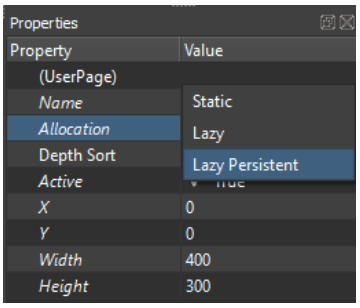


Figure 134 - Switch Page Resource Allocation

When a new page is activated, its parent object, the Switch Page, marks the current page as inactive and frees it from memory if it is using "Lazy" allocation mode.

Dynamic Switch Page

The *Dynamic Switch Page* is a special layout designed to handle page and widget transitions. Its primary purpose is to provide fade-in and fade-out animations when transitioning from one page to another. Similar to the Switch Page widget, only one child of the Dynamic Switch Page is visible at any given time. However, unlike the Switch Page widget, which switches visibility abruptly, the Dynamic Switch Page smoothly fades out the current page while fading in the newly active page.

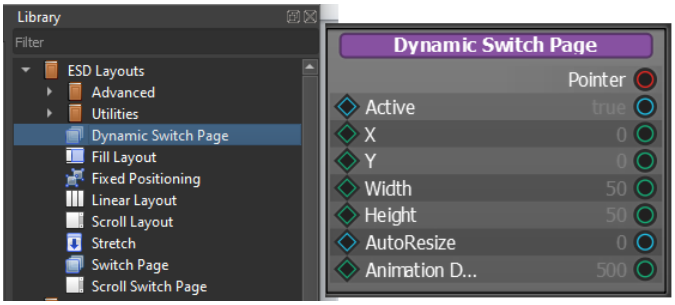


Figure 135 - Dynamic Switch Page

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 24 - Dynamic Switch Page Layout Properties

Copyright © Bridgetek Pte Ltd

Dynamic Switch Page Implementation

1. Setting a Page/Widget into Dynamic Switch Page

Unlike regular widgets, a Dynamic Switch Page cannot be dragged into Screen Layout Editor or Logic Editor. To place a page under the control of a Dynamic Switch Page layout, users must use the Project Browser and drag the widget into the Dynamic Switch Page layout.

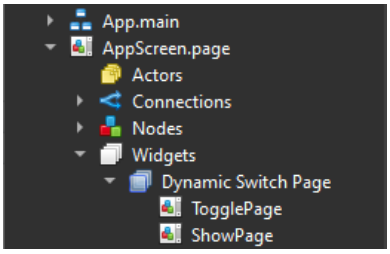


Figure 136 - Dynamic Switch Page Layout

2. Page Resource

Since ESD 4.0, page resource allocation can be configured as follows:

- Static – Always load at startup.
- Lazy – Allocated when the page becomes active and freed when it becomes inactive.
- Lazy Persistent – Allocated the first time the page becomes active and remains in memory until system reset.

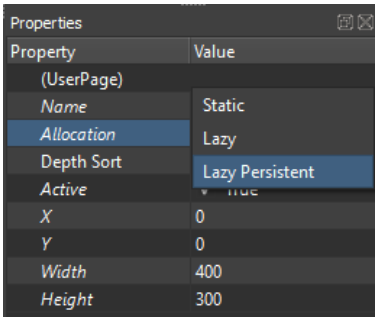


Figure 137 - Dynamic Page Resource Allocation

When a new page is activated, its parent object, the Dynamic Switch Page, marks the current page as inactive and frees it from memory if it is using "Lazy" allocation mode.

Copyright © Bridgetek Pte Ltd

Auto Switch Widget

The *Auto Switch Widget* is a layout designed for rapid transitions between its child widgets. Its primary purpose is to sequentially activate each child widget based on their depth order. The transition rate is configurable by the user, allowing it to create a GIF-like animation effect. Unlike Switch Page and Dynamic Switch Page, transitions between child widgets in the Auto Switch Widget are automatic and not trigger manually. A typical use case can be found in Examples/Basic/AutoSwitchLayout.

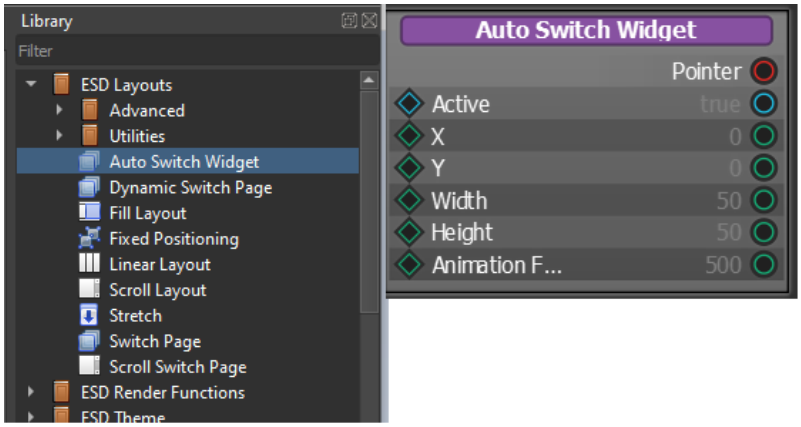


Figure 138 - Auto Switch Widget

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 25 - Auto Switch Page Layout Properties

Auto Switch Page Implementation

1. Setting Widgets into Auto Switch Page

The Auto Switch Widget can be dragged into the Logic Editor. To place a widget under the control of the Auto Switch Widget layout, users must use the Project Browser and drag the widget into the Auto Switch Widget layout.

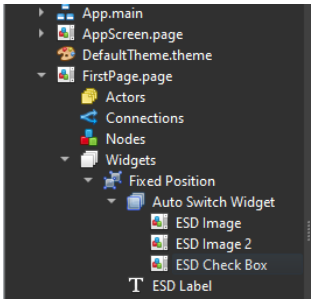


Figure 139 - Auto Switch Page Layouts

2. Widget Rendering Order

All child widgets are activated sequentially and rendered according to their depth order. To change the activation sequence, users must update the depth order of the child widgets.

Copyright © Bridgetek Pte Ltd

Fixed Positioning

The *Fixed Positioning* layout widget allows its child widgets to be fixed at design time. By default, every page starts with a Fixed Positioning layout so that all widgets within the page are positioned during the design time. The Fixed Positioning layout does not update the dimensions or positions of its child widgets at runtime; therefore, the runtime appearance remains the same as the design-time layout.

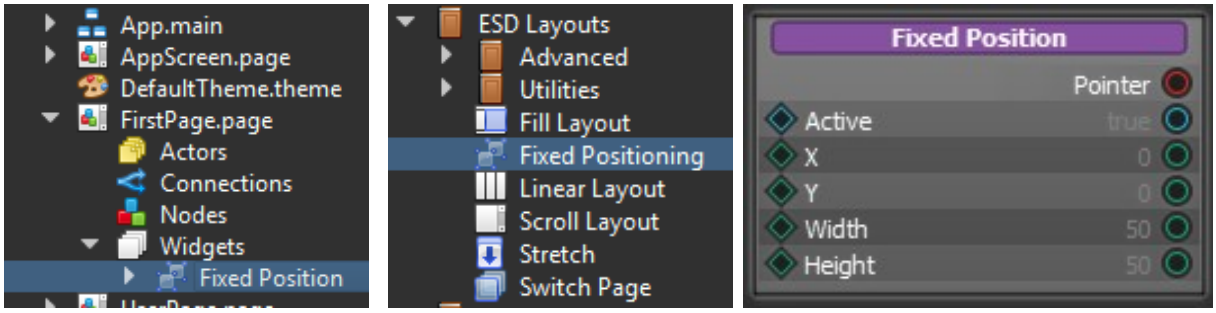


Figure 140 - Fixed Positioning

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 26 - Fixed Positioning Layout Properties

Fixed Positioning Implementation

1. Setting a Page/Widget into Fixed Positioning Layout

When a new widget is dragged and dropped into the Screen Layout Editor, it will be assigned the highest depth-sort value in the layout by default. However, the depth-sort values are user-configurable and can be adjusted manually. If a widget is not set to fixed positioning, the Screen Layout Editor will not be able to reposition it as expected. Always refer to the Project Browser (as shown in the picture) to verify the widget's structure and hierarchy.

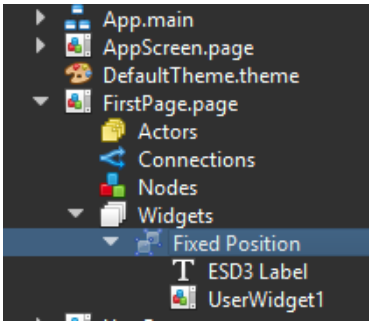


Figure 141 - Fixed Positioning Layouts

Copyright © Bridgetek Pte Ltd

2. Change Child Widgets' Depth Sort

Changing the depth-sort value of child widgets affects their rendering sequence (i.e. which widgets appear in the front of others). This functionality is consistent with the behaviour in ESD 3.0.

Fill Layout

The *Fill Layout* is the base layout for every widget and page in ESD. Each widget is created with a Fill Layout by default. For pages, the Fill Layout has a fixed size determined by the screen resolution defined by the hardware. All widgets within a Fill Layout automatically resize to fill the entire available layout area.



Figure 142 - Fill Layout

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 27 - Fill Layout Properties

Fill Layout Implementation

1. Change Widget Sequence

The sequence of widgets within a Fill Layout is determined by their depth-sort values. Changing a widget to a higher depth-sort value increases its rendering priority, causing it to appear above other widgets with lower values. For example, consider a Clock widget and an Image widget within the same Fill Layout. Both widgets occupy the full layout area. By assigning a higher depth-sort value to the Clock widget than the Image widget, the Clock widget will be rendered on top of the Image widget.

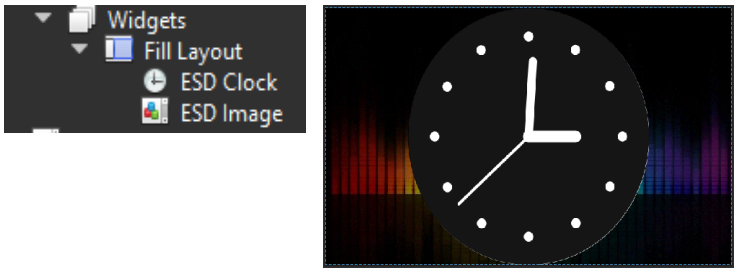


Figure 143 - Widget rendering based on depth sort in Fill Layout

Copyright © Bridgetek Pte Ltd

Linear Layout

The *Linear Layout* in ESD supports both horizontal and vertical orientations. All child widgets are arranged sequentially based on the selected orientation. In the Screen Layout Editor, users cannot directly change the sequence of widgets within a Linear Layout. However, the sequence can be modified indirectly by adjusting the widget's depth-sort values in the Property Browser.

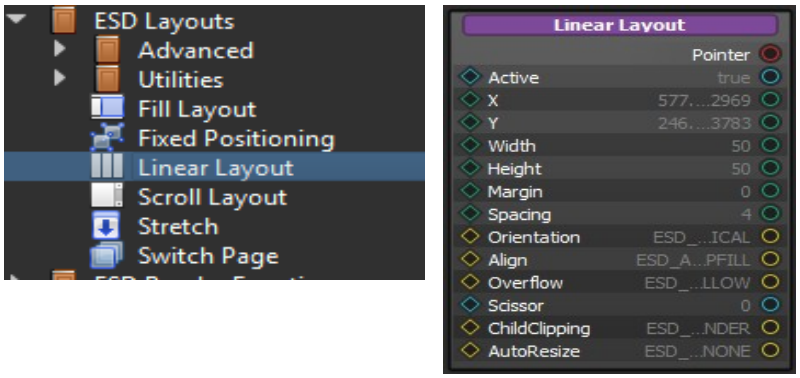


Figure 144 - Linear Layout

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 28 - Linear Layout Properties

Copyright © Bridgetek Pte Ltd

Linear Layout Implementation

1. Horizontal Layout

All child widgets within a Linear Layout configured with the orientation "ESD_ORIENTATION_HORIZONTAL" are arranged horizontally. They are distributed from left to right with spacing applied between adjacent widgets. In the example below, the Linear Layout uses the following parameters:

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

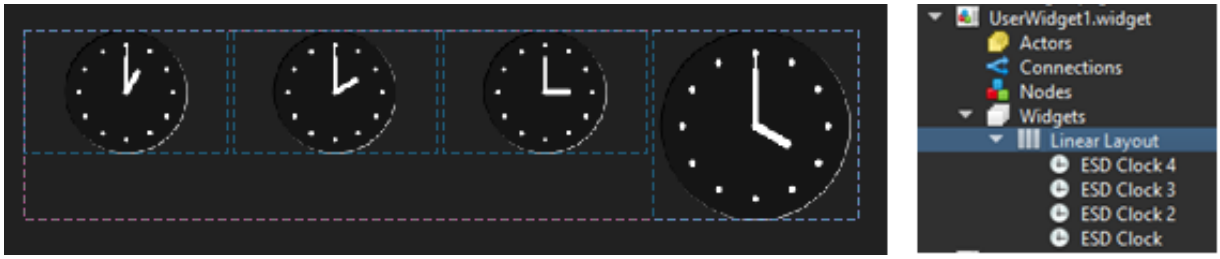


Figure 145 - Example of ESD_ORIENTATION_HORIZONTAL layout

2. Vertical Layout

All child widgets within a Linear Layout configured with the orientation "ESD_ORIENTATION_VERTICAL" are arranged vertically. They are distributed from top to bottom with spacing applied between adjacent widgets. In the example below, the Linear Layout uses the following parameters:

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

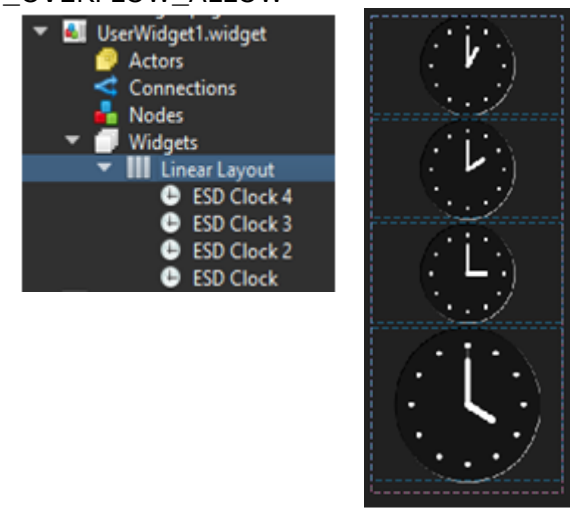


Figure 146 - Example of ESD_ORIENTATION_VERTICAL layout

Copyright © Bridgetek Pte Ltd

Stretch

The *Stretch Layout* is designed to be used in conjunction with the Linear Layout. It allows certain widgets to occupy more space within the layout by assigning them a higher weight. When some widgets need greater emphasis or proportion with the available space, Stretch Layouts can be applied to distribute space accordingly based on their assigned weight values.

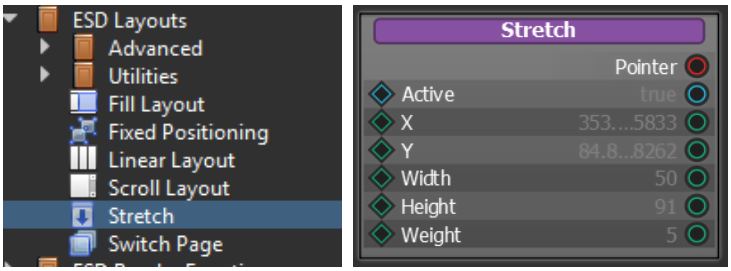


Figure 147 - Stretch Layout

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 29 - Stretch Layout Properties

Stretch Layout Implementation

1. Space Stretching

The example below demonstrates how unused space can be stretched horizontally. A Stretch Layout has been added within the Linear Layout to absorb the remaining space. In this case, the weight factor is set to 1.

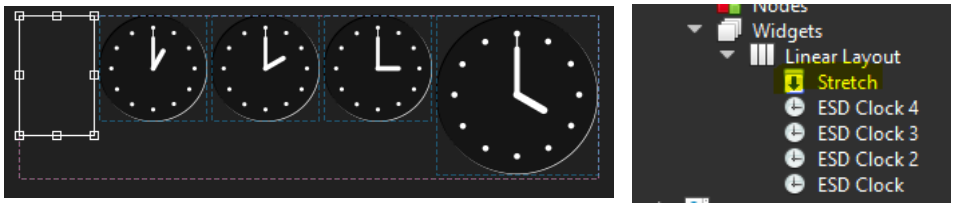


Figure 148 - Example of horizontal space stretching using a stretch layout

2. Widget Weight Stretching

The example below demonstrates widget stretching behaviour. In this case, each Clock widget is assigned a different stretch weight value. As a result, each widget occupies a proportion of the available width according to its assigned weight, as shown below.

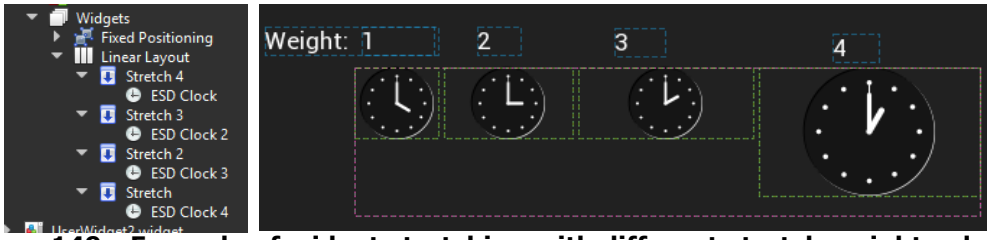


Figure 149 - Example of widget stretching with different stretch weight values

Copyright © Bridgetek Pte Ltd

Scroll Layout

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

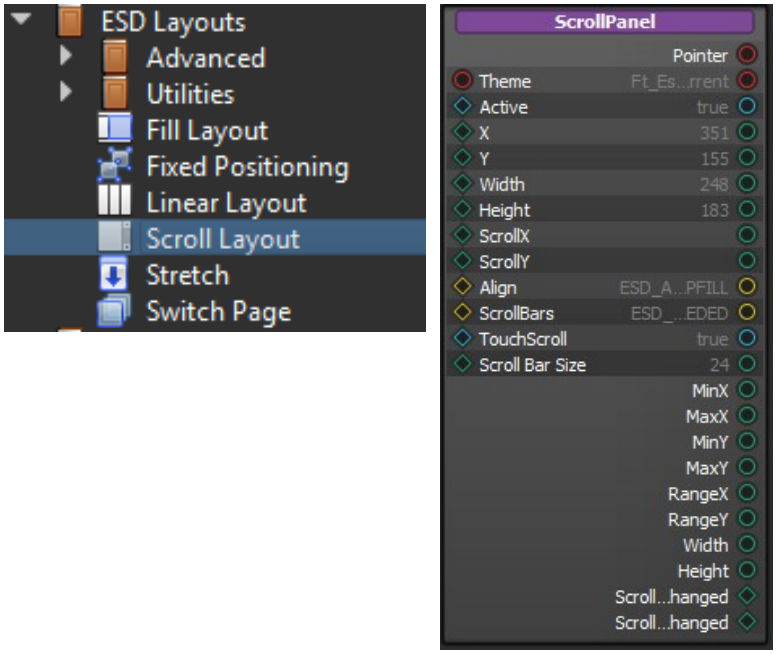


Figure 150 - Scroll Layout

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 30 - Scroll Layout Properties

Scroll Layout Implementation

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

Copyright © Bridgetek Pte Ltd

Scroll Switch Page Layout

The *Scroll Switch Page Layout* consists of a Switch Page Manager and a Scroll Panel to support runtime scrolling functionality.

This layout requires three lazy-loaded page instances and a starting index, namely: the previous page instance, the current page instance, and the next page instance.

When a user performs a touch swipe gesture, the page transitions based on the swipe direction. The behaviour of the Scroll Switch Page Layout is as follows:



Figure 151 - Scroll Switch Page Layout

- During non-swiping runtime, only the current page is rendered, while update logic for all three pages (previous, current, and next) continues to execute.
- During swipe, both the current page and the target page (previous or next instance) are displayed.
- On touch release, the view either reverts to the original position if the swipe threshold is not met, or transitions to the new page. Touch scrolling is temporally disabled until the action completes.
- After a successful page switch, a "NewPage" event is triggered, and a new page instance is requested and assigned 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

Copyright © Bridgetek Pte Ltd

Current Index	The output of current index
New Index	The output of new index

Table 31 - Scroll Switch Page Layout Properties

Scroll Switch Page Layout Implementation

In a typical Scroll Switch Page Layout configuration, the setup follows the rules below:

- Connect the Start slot to the "Initialise" slot of Scroll Switch Page Layout instance.
- All switchable pages must be configured as child widgets of the Scroll Switch Page.
- Each page should use lazy allocation and have the same width as the Scroll Switch Page.
- The initial Previous Page, Current, and Next pages are active by default, while all other child pages remain inactive.
- The initial index should match the index of the initial current page.
- It is recommended to use a "Switch" node to trigger the target page's "Activate" signal.
- It is recommended to use a "Switch Value" node to retrieve the corresponding page instance from the available pages.
- Header and footer pages are recommended and should be added as sibling widgets of the Scroll Switch Page Layout instance.

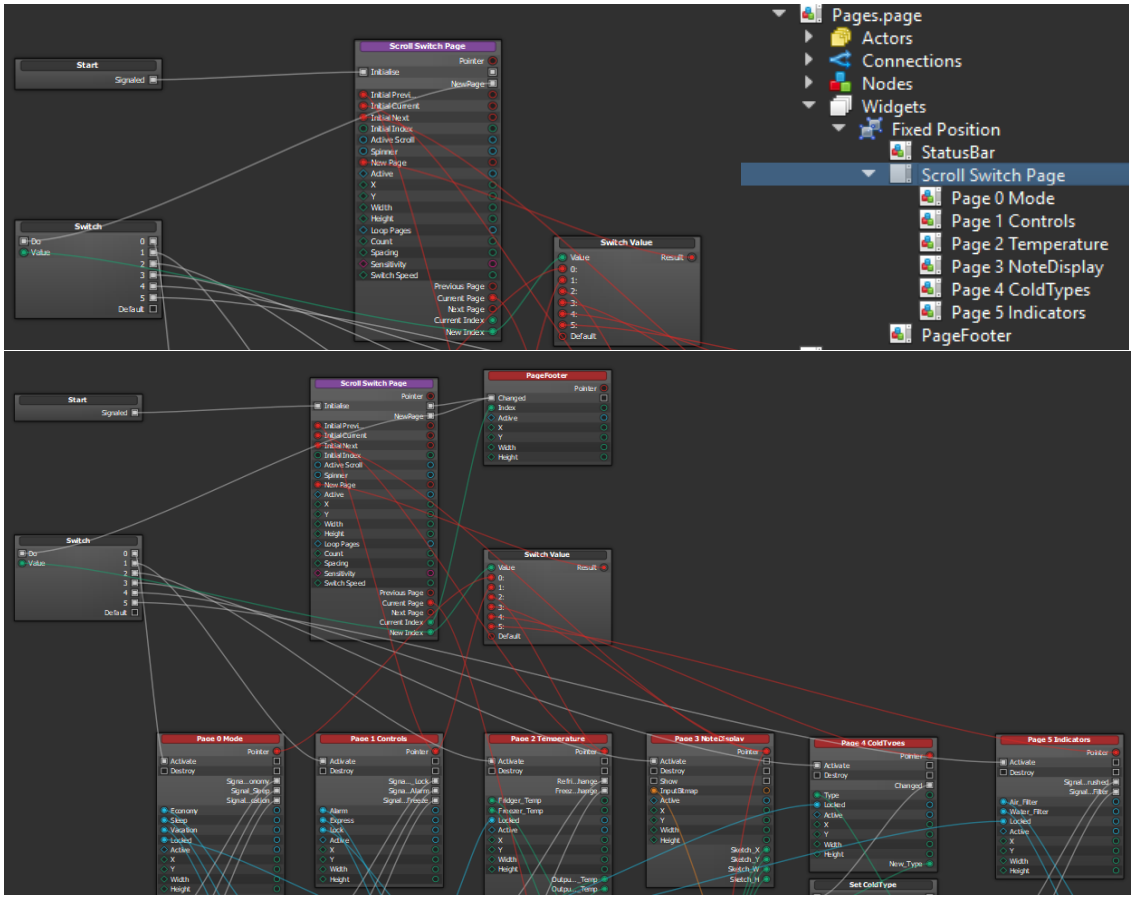


Figure 152 - Sample Scroll Switch Page Implementation

Copyright © Bridgetek Pte Ltd

Signal Switch Page Layout

The *Signal Switch Page Layout* is a simplified version of the *Scroll Switch Page Layout* designed for use cases where touch input is not supported.

Like the *Scroll Switch Page Layout*, it requires three lazy-loaded page instances and a starting index: previous, current and the next page instances.

Two slots, namely NextPageSignal and PreviousPageSignal are provided to switch between pages. Page transitions can be triggered by buttons or other external signals.

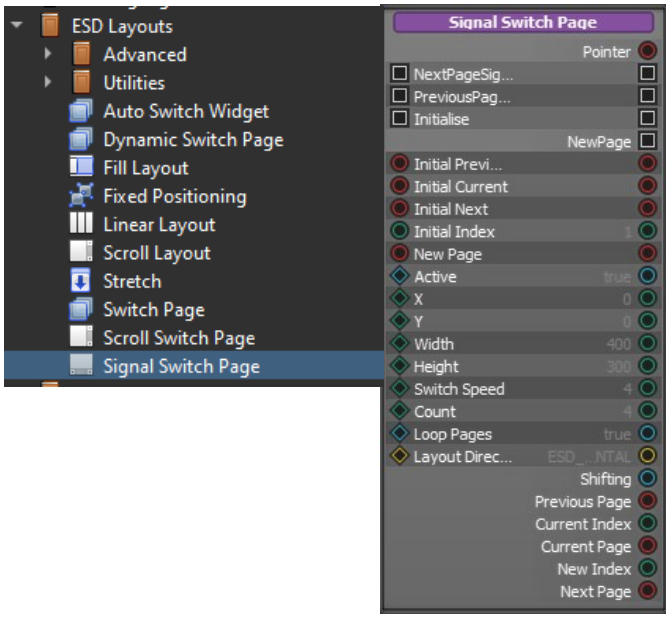


Figure 153 - Signal Switch Page Layout

Similar to *Scroll Switch Page Layout,* when a swipe event is triggered via signal, both the current page and the target page (can be previous or next page instance) are displayed.

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 32 - Signal Switch Page Layout Properties

Copyright © Bridgetek Pte Ltd

ESD Render Functions

All render functions are non-widget components and are not managed by the layout system in ESD 4.14. They can only be used withing a widget and are intended as utility functions to help users create custom widgets.

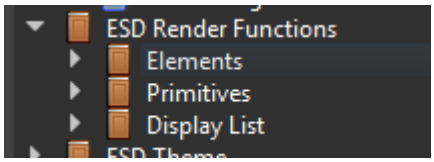


Figure 154 - ESD Render Functions

Note: To preview render function result at design time, a Render Slot is required in the custom widget.

Elements

Elements are simple, basic widgets typically used to build more complex, higher-level widgets. They usually do not handle touch input.

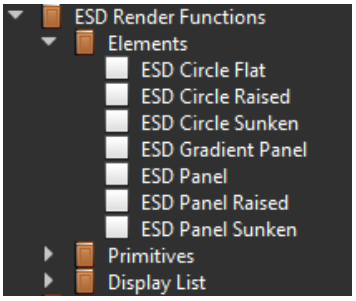


Figure 155 - ESD Elements

ESD Circle

The *Circle* element allows users to draw circles on the screen. It supports the following styles: *ESD Circle Flat; ESD Circle Raised* and *ESD Circle Sunken*.

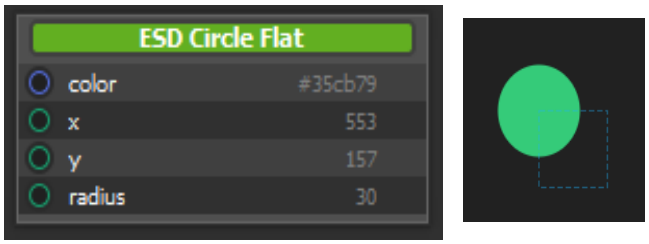


Figure 156 - ESD Circle Element

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 33 - ESD Circle Element Properties

ESD Panel

The *ESD Panel* is an adjustable rectangular widget used to define a coloured area. Its colour can be set via a theme or user's selection. Internally, it is built using EVE primitives such as RECTS and SCISSORS. It supports two styles: *ESD Panel Raised* and *ESD Panel Sunken*.

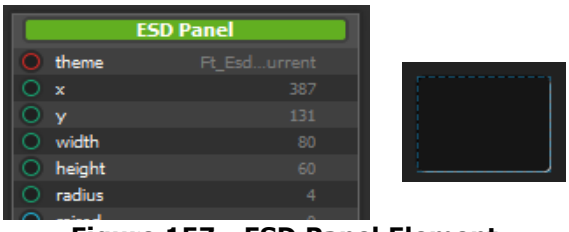


Figure 157 - ESD Panel Element

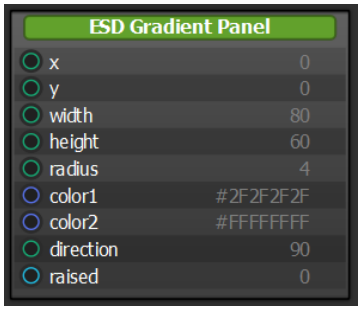
Copyright © Bridgetek Pte Ltd

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 34 - ESD Panel Element Properties

ESD Gradient Panel

The *Gradient Panel* element is an adjustable rectangular widget used to define a gradient-coloured area. The gradient is defined using two colours: a start color and an end color. Its direction can be adjusted by configuring the gradient direction.



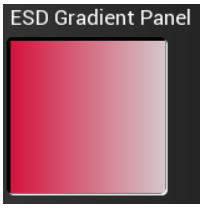


Figure 158 - ESD Gradient Panel Element

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 35 - ESD Gradient Panel Element Properties

Primitives

A *Primitive* is a special type of logic node that wraps a rendering function called through a Render slot. It does not return any value, and its properties cannot be passed to other logic nodes. Like widgets, it is rendered on the screen but does not handle user input.

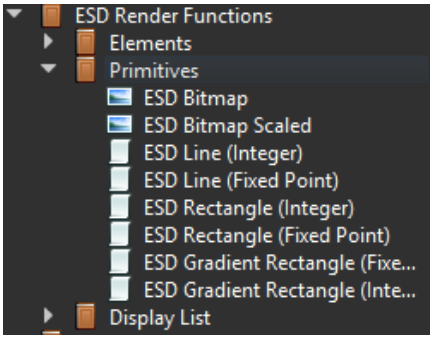


Figure 159 - Primitives

Copyright © Bridgetek Pte Ltd

ESD Bitmap

The *ESD Bitmap* object allows users to display a bitmap resource. A bitmap resource is created from an image file added to the project. By default, it is treated as a single bitmap cell named "(Bitmap Resource)_0".

For example, if the resource is named "photo.jpg", the default bitmap cell name will be "photo_0". Users can define multiple bitmap cells by specifying the *"CellHeight"* property.

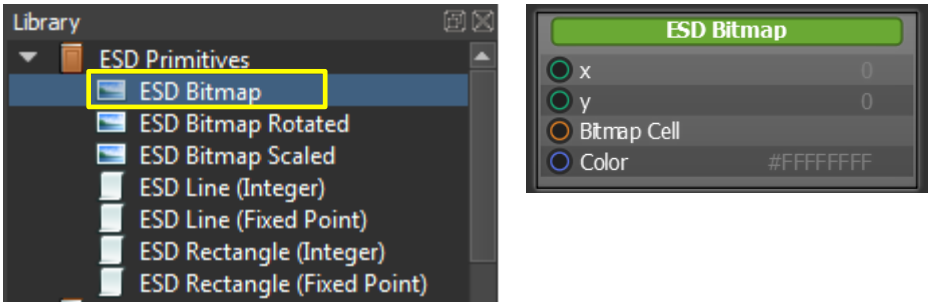


Figure 160 - Bitmap Primitive

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 36 - 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"*.

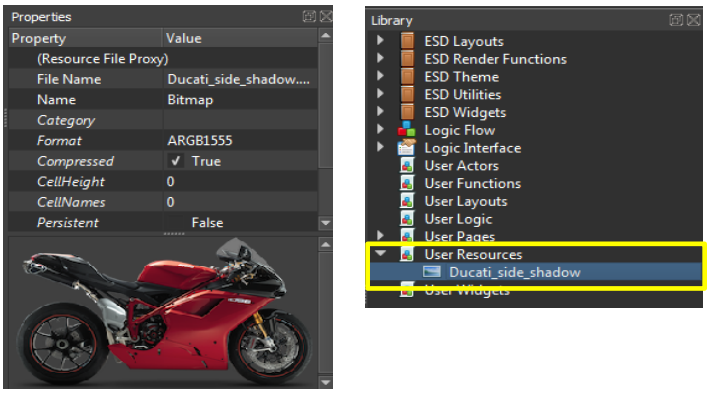


Figure 161 - Example of bitmap resource added to the library

Copyright © Bridgetek Pte Ltd

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.

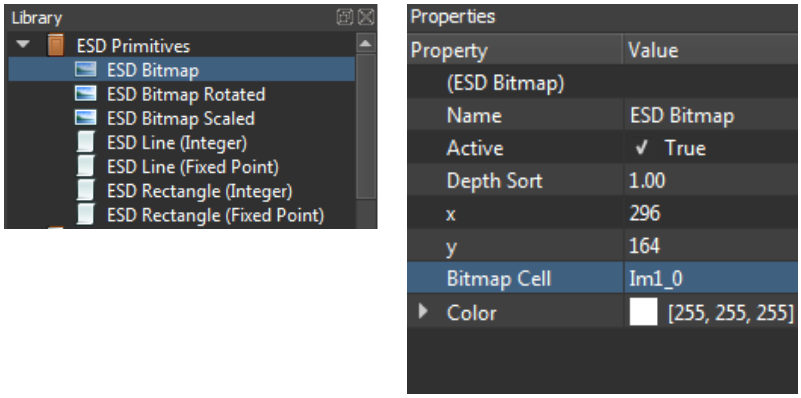


Figure 162 - Example of using bitmap cell

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.

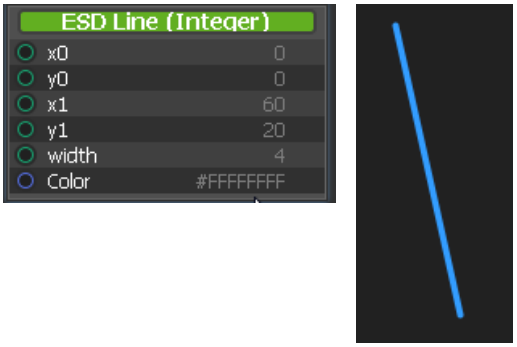


Figure 163 - ESD Line

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.

Copyright © Bridgetek Pte Ltd

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 37 - 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.

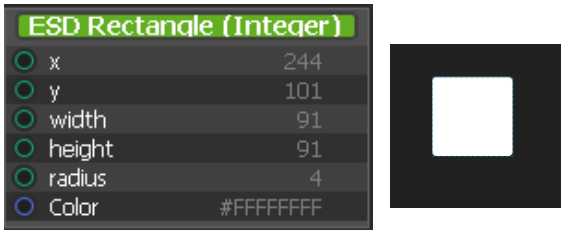


Figure 164 - ESD Rectangle

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 38 - 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.

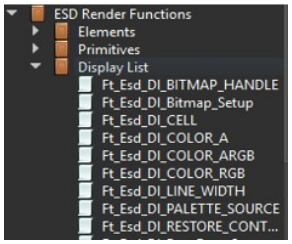


Figure 165 - ESD Display List

Copyright © Bridgetek Pte Ltd

ESD Theme

A *Theme* is a collection of colours used across the application to maintain a consistent style. The Library Browser provides the necessary logic nodes to make full use of theme functionality.

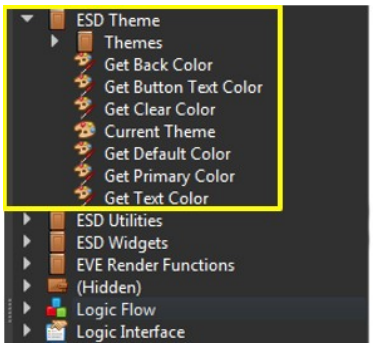


Figure 166 - Library Browser - ESD Theme

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.

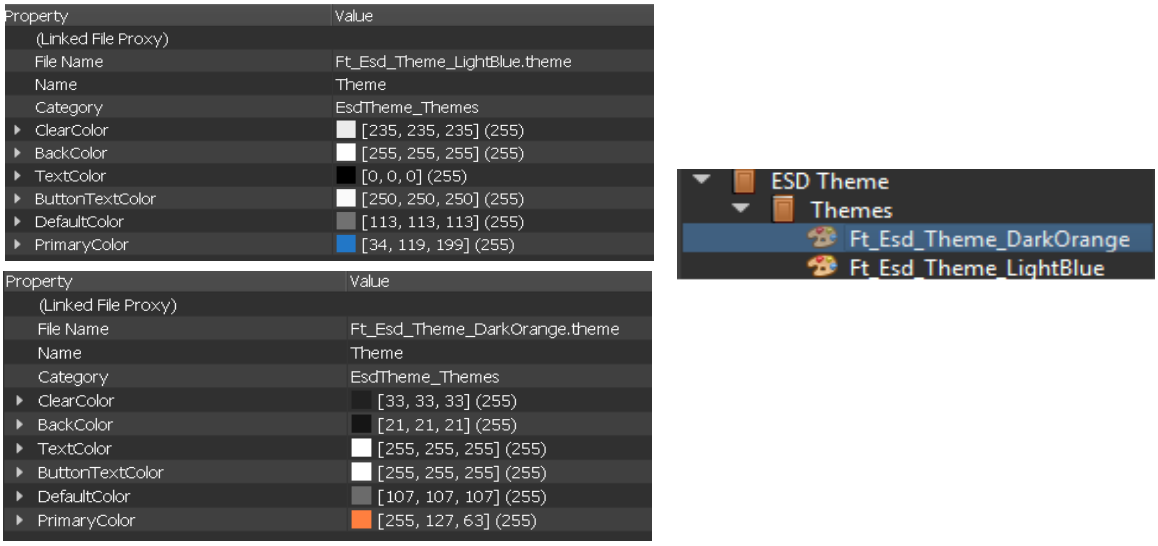


Figure 167 - ESD Built-in Themes

Table 39 and Table 40 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 39 - Ft_Esd_Theme_LightBlue Theme

Copyright © Bridgetek Pte Ltd

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 40 - 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.

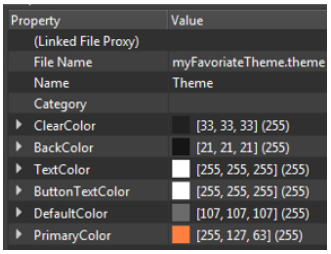


Figure 168 - User Theme Property

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

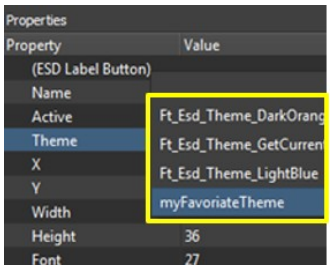


Figure 169 - Example of applying a theme to widgets

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.

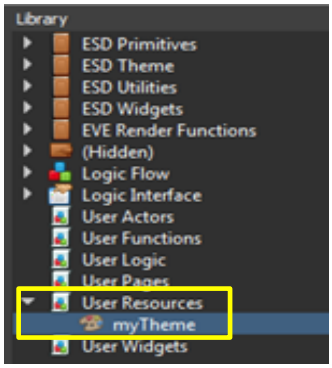


Figure 170 - User Resources in the Library Browser

Copyright © Bridgetek Pte Ltd

ESD Utilities

ESD Utilities is a collection of utility functions provided by ESD. The table below lists the available ESD Utilities and their functionalities.



Figure 171 - ESD Utilities

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 41 - ESD Utilities & Description

Copyright © Bridgetek Pte Ltd

ESD Idle Checker

The ESD Idle Checker Actor monitors application activity to determine when the system should enter or exit sleep mode. A common use case is detecting idle state and triggering a screen saver page. In the example shown in "AppScreen.page", the Idle Checker is used to switch between the main application page and the idle page. It operates with 250 ms interval and a count of 60, resulting in a 15-second timeout (60 x 250 ms = 15,000 ms) when no touch events are detected.

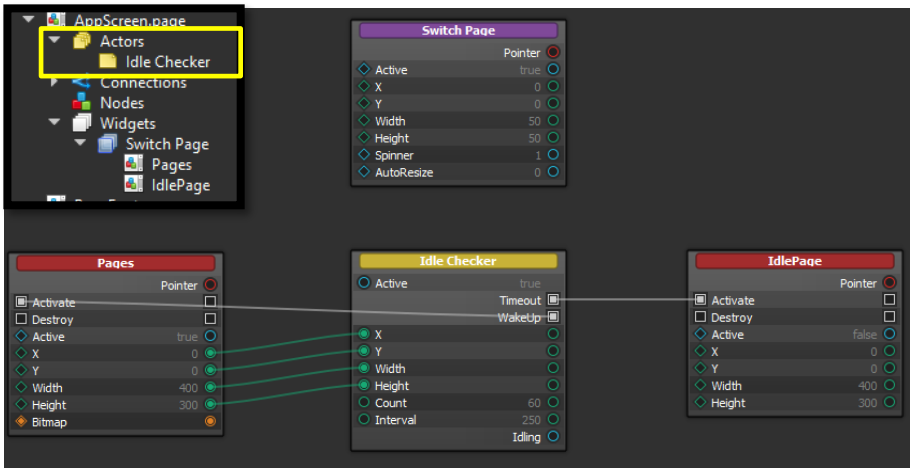


Figure 172 - ESD Idle Checker

Logic Flow

A *Logic Flow* is a built-in ESD logic node used to define application logic and control flow. It consists of the following nodes:

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

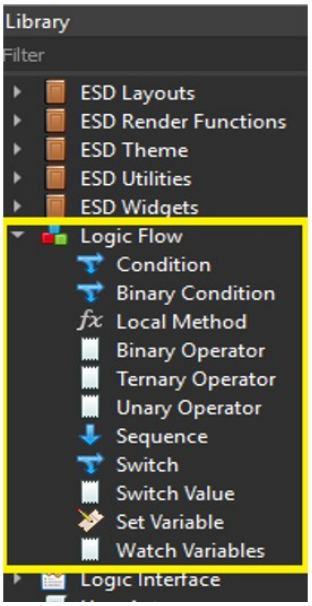


Figure 173 - Logic Flow

Copyright © Bridgetek Pte Ltd

Condition

A *Condition* node defines a set of rules that are executed when a specified condition is met. In other words, it is used to introduce decision points in the logic flow.

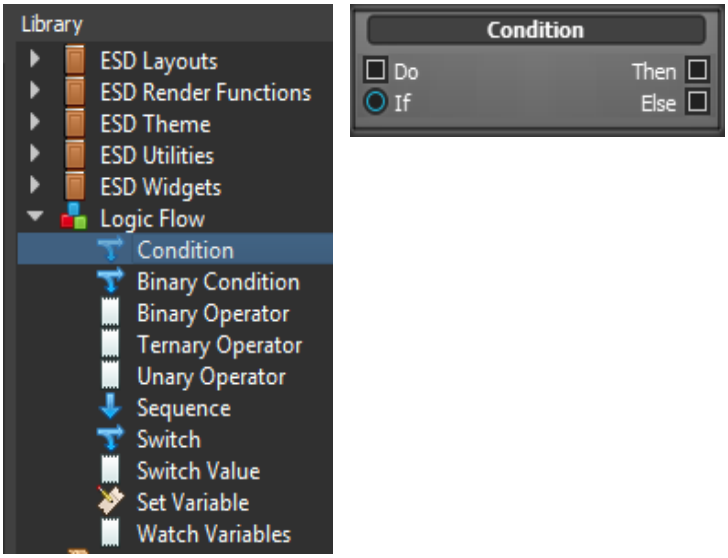


Figure 174 - Logic Flow - Condition Node

Whenever a Condition node is executed, the *"Then"* condition is triggered if the *"If"* expression evaluates to True; otherwise, the *"Else"* condition is executed.

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

Binary Condition

A *Binary Condition* node defines a set of rules that are executed based on the result of a binary operation between two input values. In other words, it introduces decision points in the logic flow.

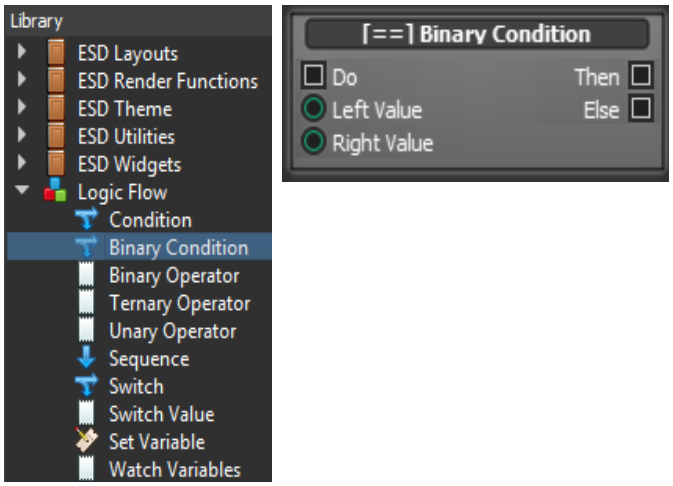


Figure 175 - Logic Flow - Binary Condition Node

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.

Copyright © Bridgetek Pte Ltd

Local Method

A Local Method node is used to create a visual function node. It acts as a place holder where users define input parameters and implement the function logic in the Logic Editor. It allows configuration of the number of parameters, along with their types and names. User can adjust the parameter count using a scroll bar and define each parameter in the Properties tab.

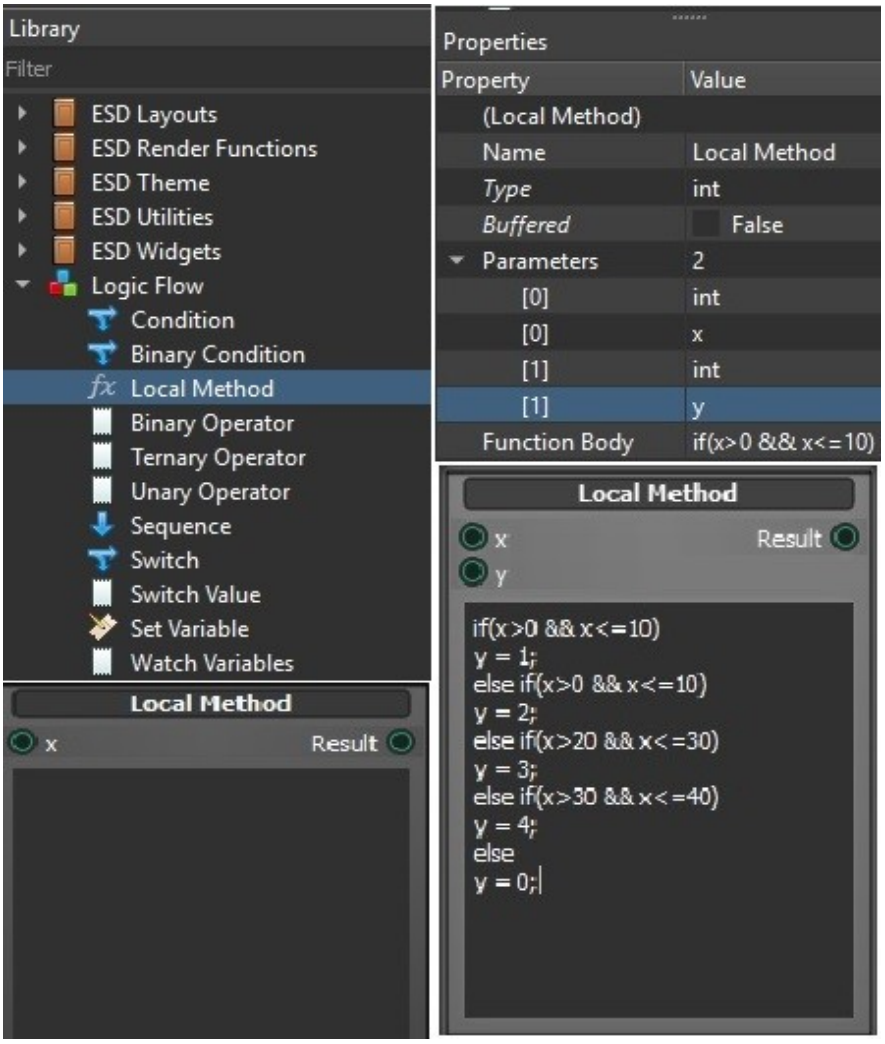


Figure 176 - Logic Flow - Local Method

- "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.

Copyright © Bridgetek Pte Ltd

Binary Operator

This node contains *Binary Operator* that operates on two input operands and returns a computed output. For example: *Result=Left Value x Right Value*.

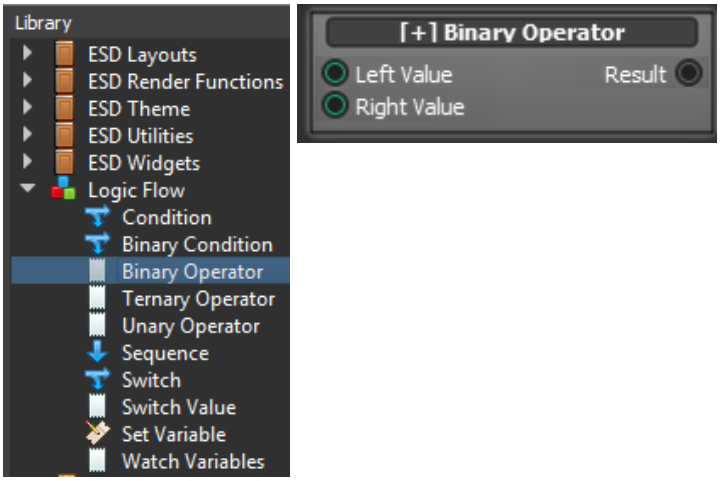


Figure 177 - Logic Flow - Binary Operator Node

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 42 - Binary Operators Supported by ESD 4.14

Copyright © Bridgetek Pte Ltd

Ternary Operator

This node contains a *Ternary Operator* that works with 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".*

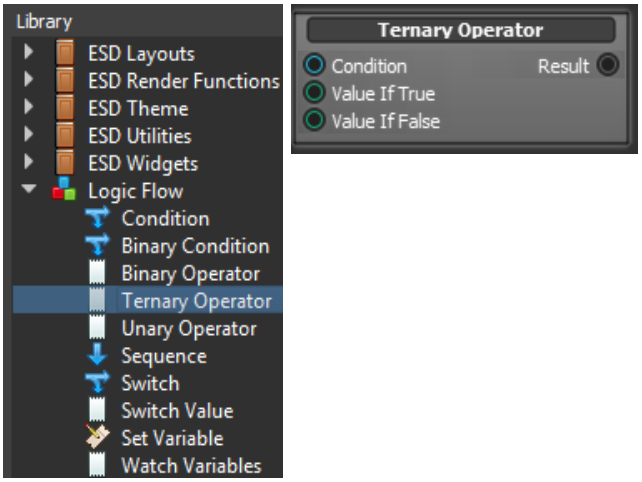


Figure 178 - Logic Flow - Ternary Operator Node

Unary Operator

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

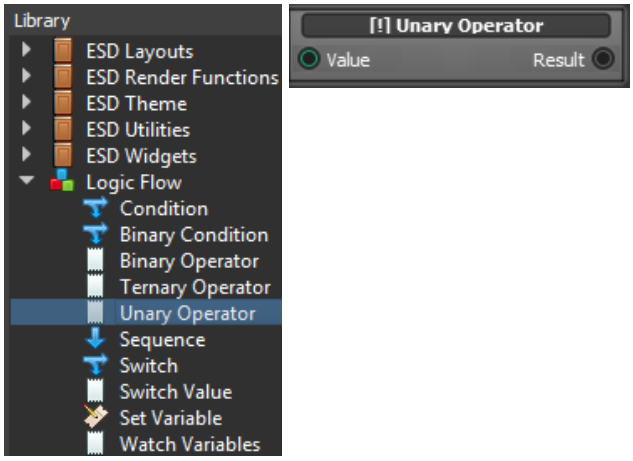


Figure 179 - Logic Flow - Unary Operator Node

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 43 - Unary Operators Supported by ESD

Copyright © Bridgetek Pte Ltd

Sequence

The "*Sequence*" node allows users to define a series of actions to be executed in order. When the "Do" input is triggered, the calls are executed sequentially. The number of calls is defined by the "Count" property.

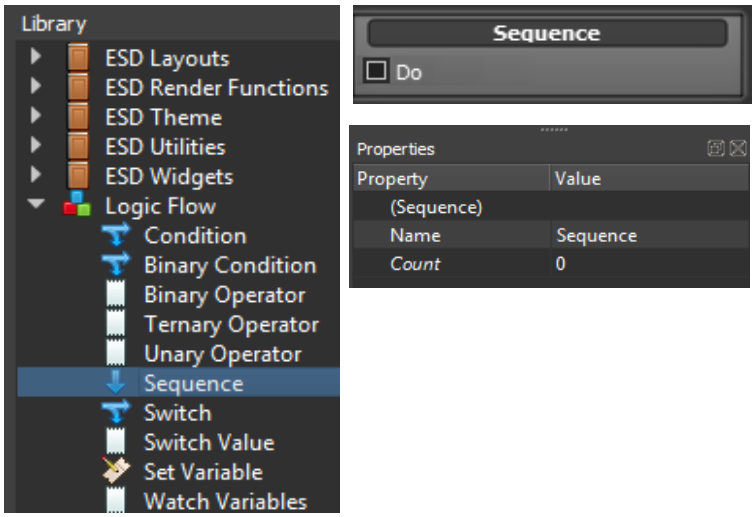


Figure 180 - Logic Flow - Sequence Node

Switch Node

The "*Switch*" node routes execution to different branches based on an input value, similar to a "Switch-Case" statement in C language. Users can define multiple branches and specify the comparison values using the "Cases" property. In the example shown in Figure 181, three branches are defined and triggered when the input value matches 1 or 2 or 3 respectively. Since the input value is set to 4 and does not match any case, the default branch is executed.

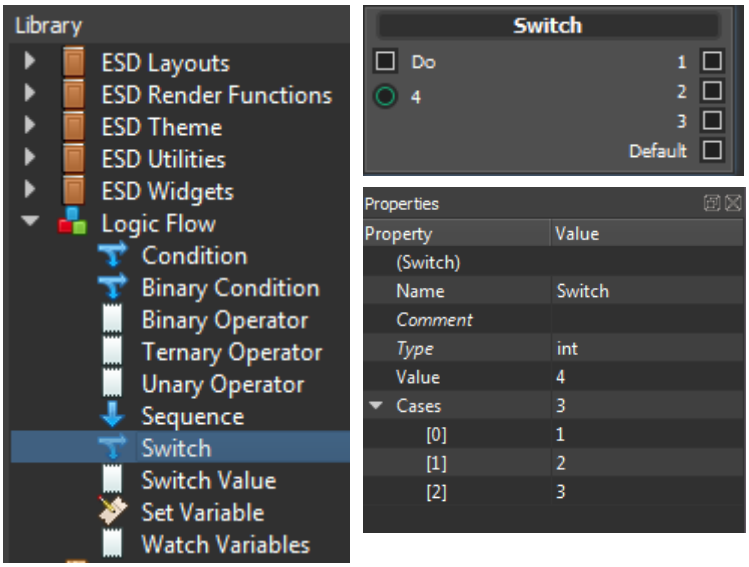


Figure 181 - Logic Flow - Switch Node

Copyright © Bridgetek Pte Ltd

Switch Value

The "*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 182 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.

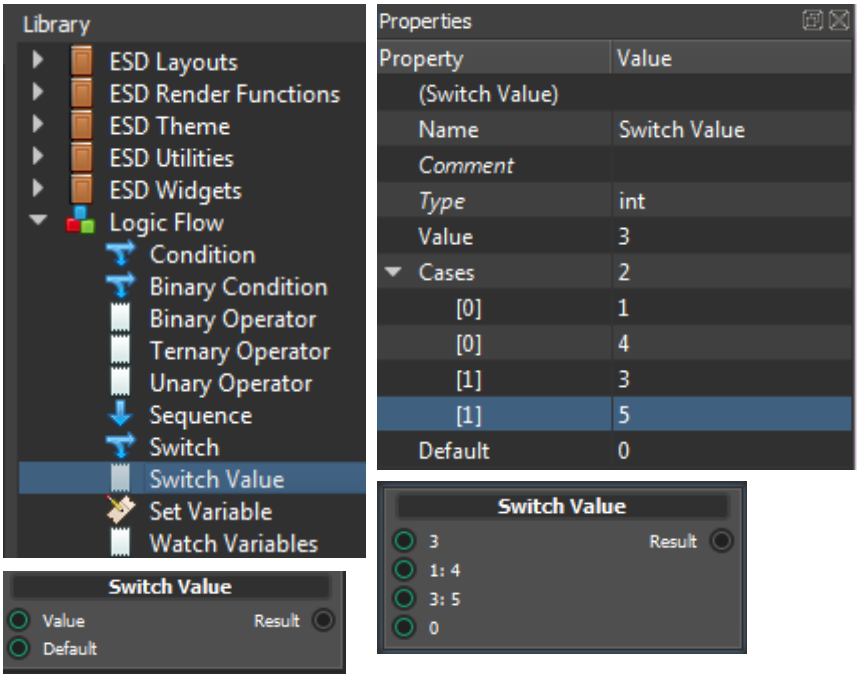


Figure 182 - Logic Flow - Switch Value

Set Variable

The *Set Variable* node is used to assign values to 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).

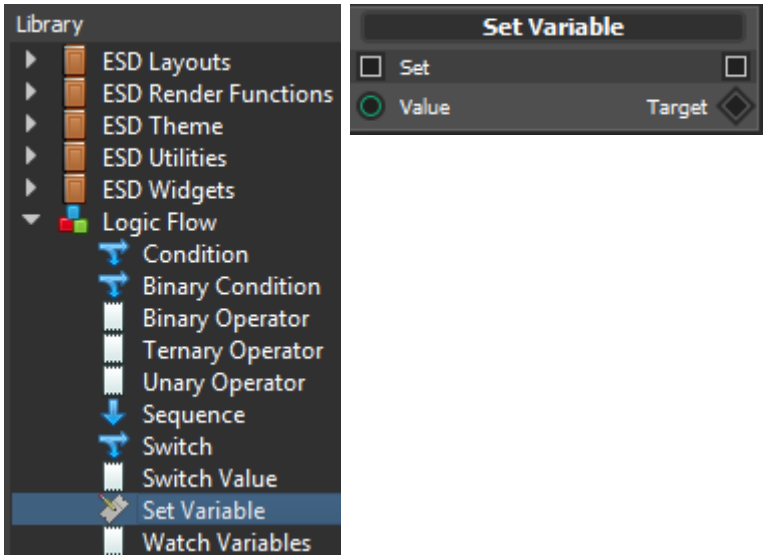
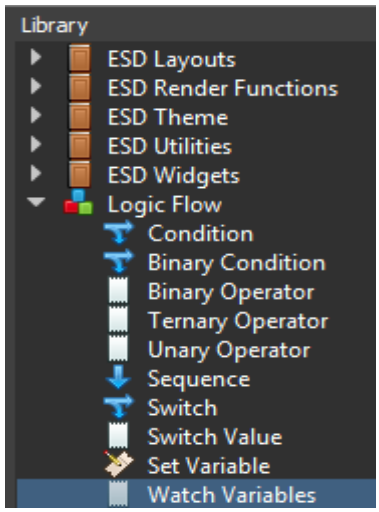


Figure 183 - Logic Flow - Set Variable Node

Copyright © Bridgetek Pte Ltd

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:



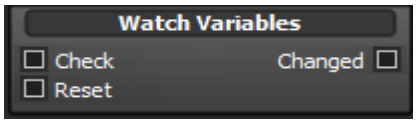


Figure 184 - Logic Flow - Watch Variable Node

- "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*

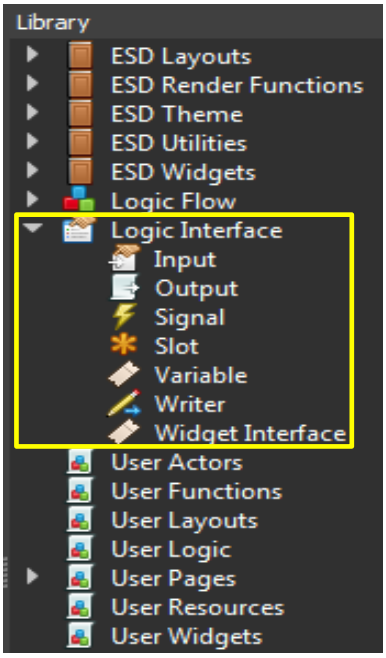


Figure 185 - Logic Interface

Copyright © Bridgetek Pte Ltd

Input

The *Input* node is used to define an input interface for a parent logic node to receive and bind incoming data. For example, in widget creation, an input node defines a property whose value is provided through external connections. Its configuration can be set via the Property Editor.

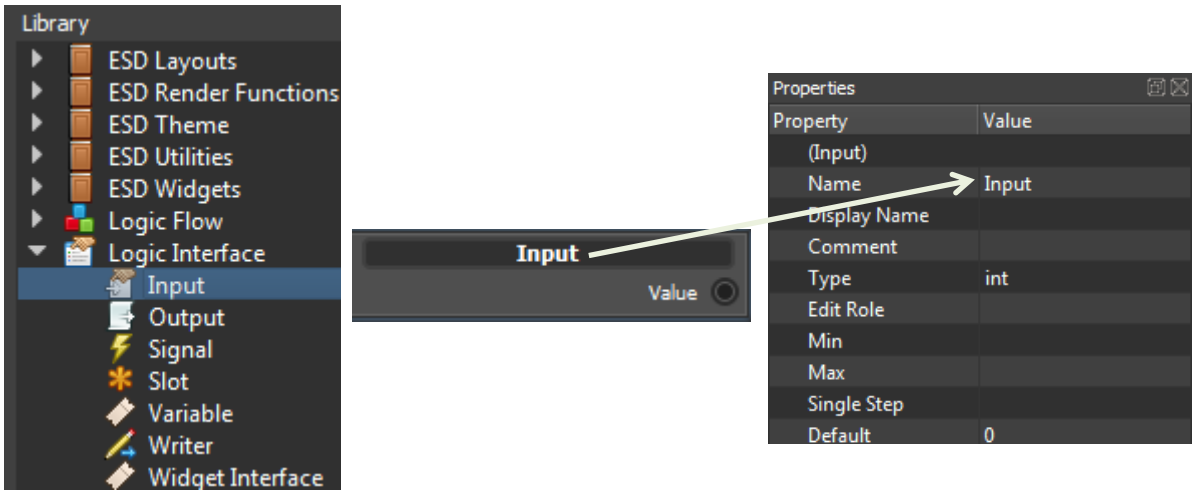


Figure 186 - Logic Interface - Input Node

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.

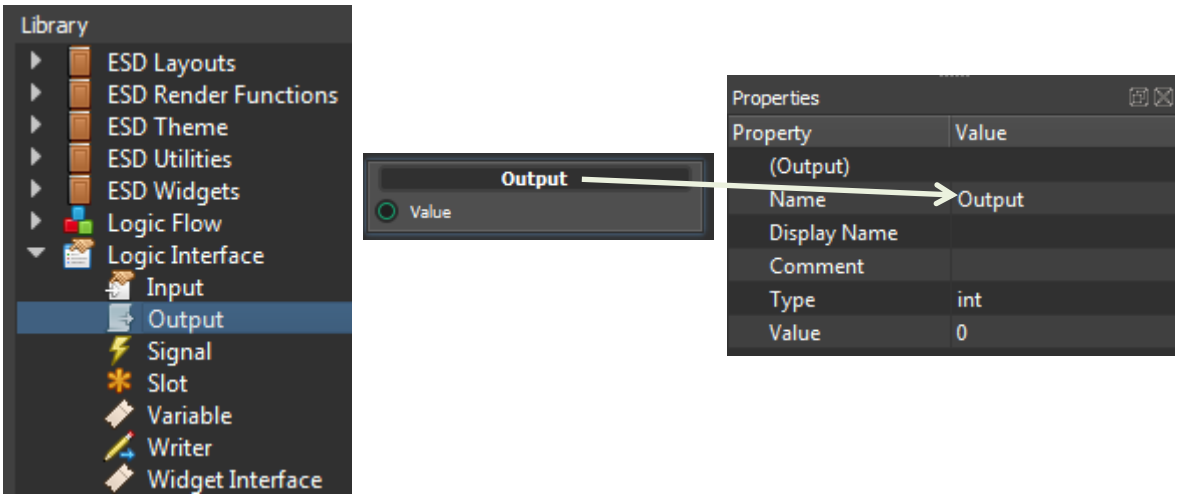


Figure 187 - Logic Interface - Output Node

Copyright © Bridgetek Pte Ltd

Signal

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



Figure 188 - Logic Interface - Signal Node

Slot

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

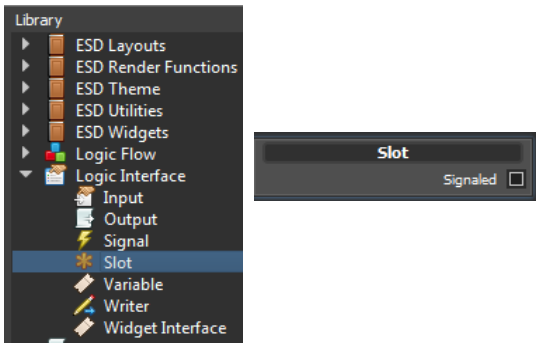


Figure 189 - Logic Interface - Slot Node

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 44 - Built-in Slots

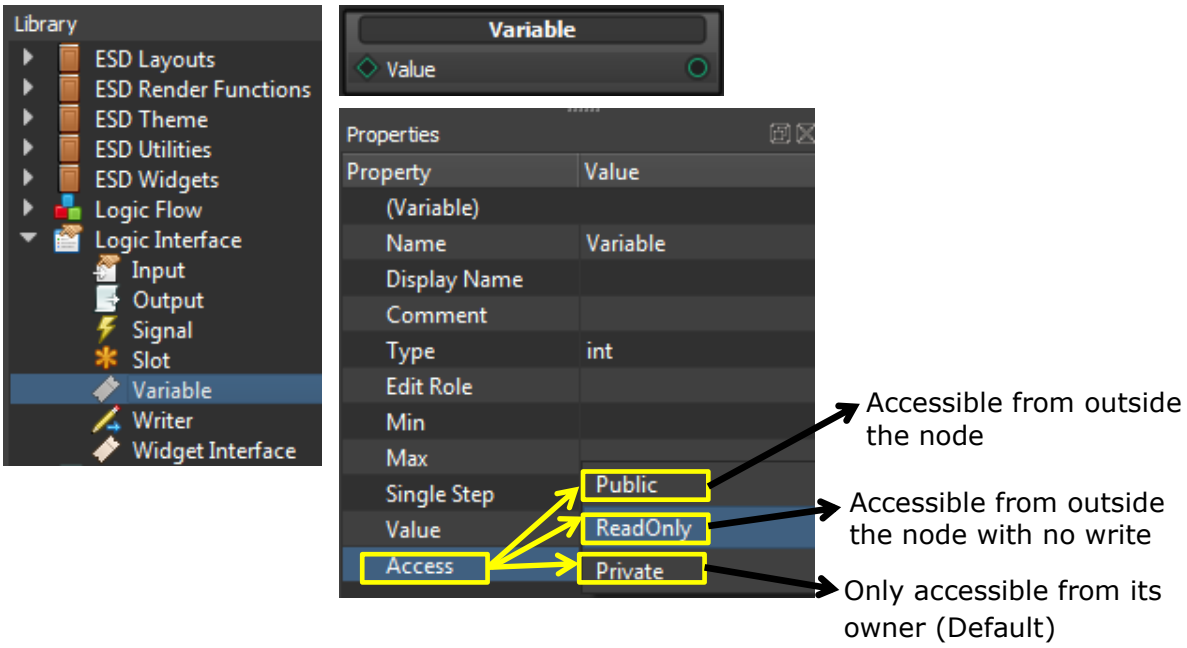
Copyright © Bridgetek Pte Ltd

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 Editor

Figure 190 - Logic Interface - Variable Node

Copyright © Bridgetek Pte Ltd

Writer

The *Writer* node defines a value-writing interface for its parent node. It is typically used to update external variable. For example, in the ESD Spinbox widget, the "Changed" writer updates connected external variables whenever the Spinbox value changes.

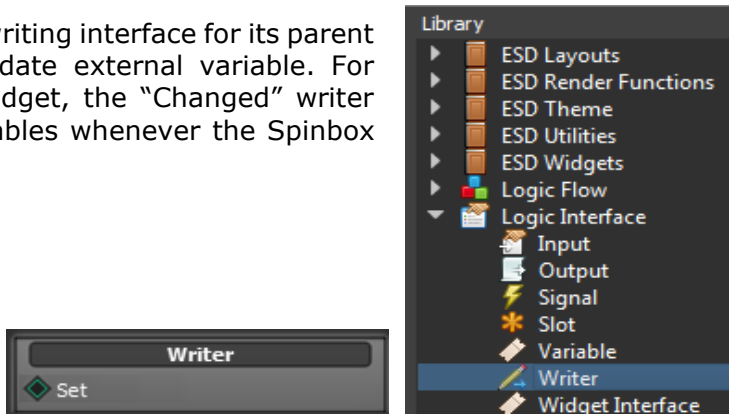


Figure 191 - Logic Interface - Writer Node

Widget Interface

The *Widget Interface* node represents the base interface shared by all widgets. Since ESD 4.0, layouts, pages, and widgets are implemented based on this interface. It is primarily intended for advanced users to design custom widgets and access core information from the ESD widget framework.

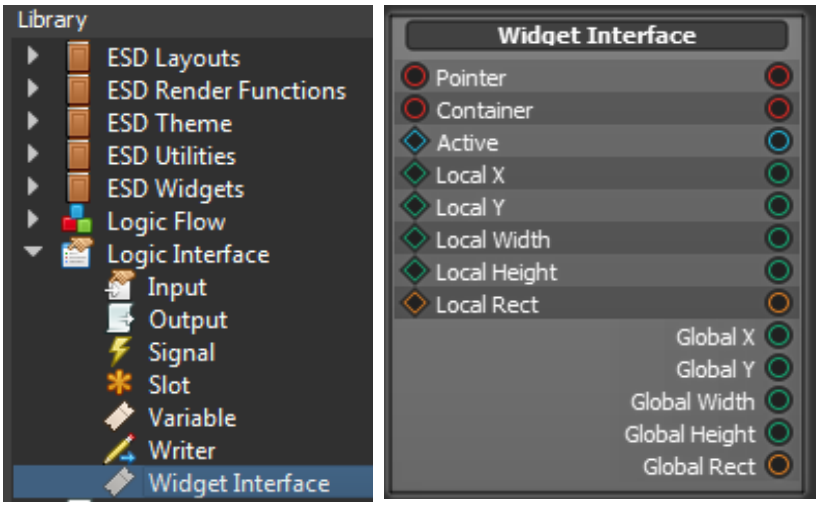


Figure 192 - Widget Interface

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

Copyright © Bridgetek Pte Ltd

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 45 - 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 a unified interface for editing properties. In ESD, each logic node, including widgets, has predefined properties. Users can edit these values through the Property Editor to customize the behaviour of logic nodes.

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

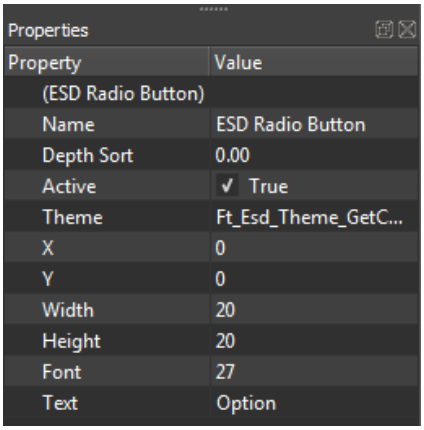


Figure 193 - Property Editor

Copyright © Bridgetek Pte Ltd

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*

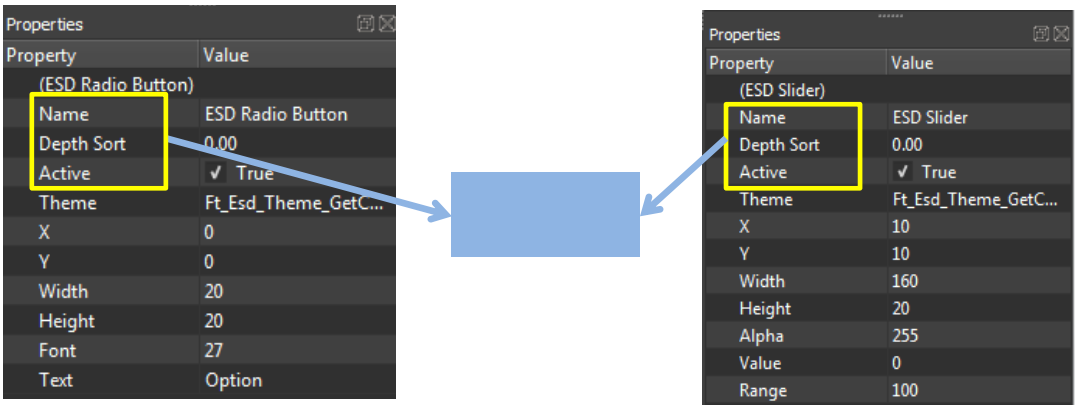


Figure 194 - ESD Common Properties

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.

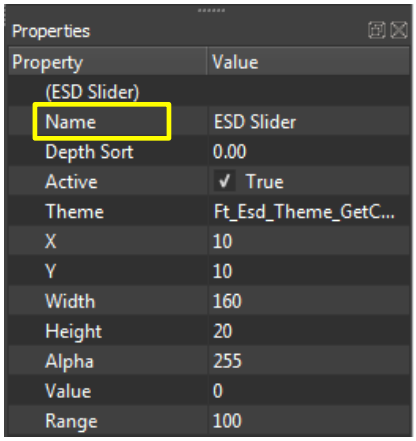


Figure 195 - Name Property

Copyright © Bridgetek Pte Ltd

Depth Sort

Each widget and page has a common property called *Depth Sort*. It is a floating-point value that defines the rendering order. The default value is "0.00" and users can adjust it to control the sequence of rendering.

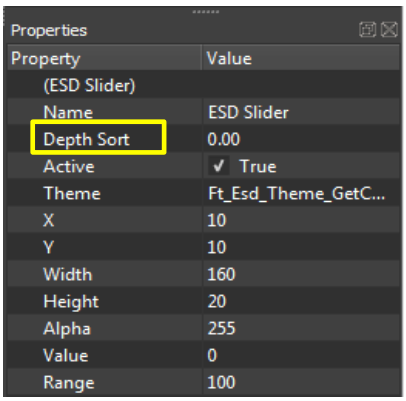


Figure 196 - Depth Sort Property

The context menu allows users to adjust the rendering sequence without manually changing depth-sort values. It can be accessed in the Layout Editor by selecting a widget or page and right-clicking it.

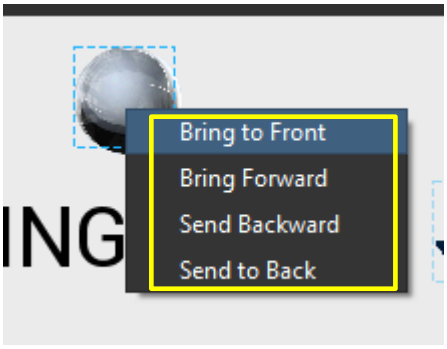


Figure 197 - Right Click Menu

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

Active

Each widget has an *Active* property of Boolean type that controls whether it is rendered on the screen. When set to false, the widget is not rendered but remains in memory as part of the application state.

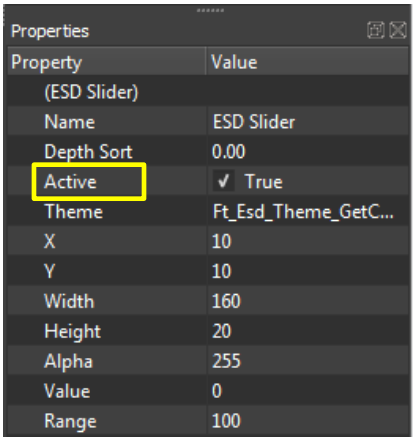


Figure 198 - Active Property



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

Copyright © Bridgetek Pte Ltd

Programming Features

This section describes the programming features in ESD. ESD allows users to write C code and expose it as logic nodes, enabling code reuse across multiple projects.

Macros

ESD provides a set of macros to inform the development environment about available functionality in user code,. These macros are required when exposing custom C code to the *Logic Node Editor* or *Layout Editor*.

These macros are typically used alongside the source code, and it is recommended to place all associated definitions immediately after the macro declarations in the same file. Failure to do so may result in unexpected behaviour. For e.g., the *ESD_FUNCTION* macro must be immediately followed by its function definition. Comments are allowed between the macro and its definition.

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

Copyright © Bridgetek Pte Ltd

	ColorRGB	Color, rgb without alpha
	Library	Select a function specified with ESD_FUNCTION to provide the value (useful for pointer values)

Table 46 - Macros - ESD_TYPE – Parameters

ESD_ENUM

Enumeration types can be used as data types in the Logic Editor. The specified type is applied in the generated code. Internally, only enumeration identifiers are parsed, while their values are ignored. When used in properties, enumerations are serialized as identifier strings and are edited through a drop-down menu in the *Property 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 checkboxes 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.

Copyright © Bridgetek Pte Ltd

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 used to register a user-defined function in ESD, allowing it to appear as a User Function node in the Library Browser. It is not intended for static functions, even though ESD technically allows any function to be declared with the *ESD_FUNCTION* macro.

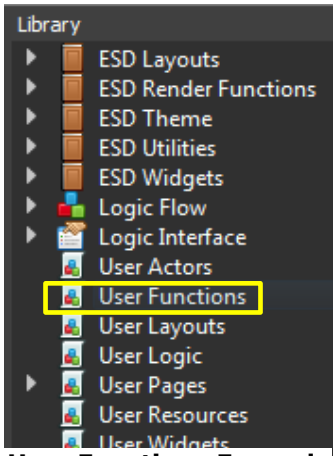


Figure 199 - User Functions Example

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)
```


Copyright © Bridgetek Pte Ltd

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

Parameters	Description
Type	The function's return value type

Table 47 - 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 will have a function with just one output value which is called whenever the output value is used. Here is an example:

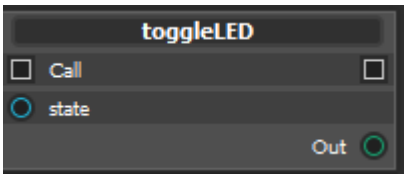


Figure 200 - Toggle LED

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

ESD_METHOD

ESD_METHOD works like *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 –

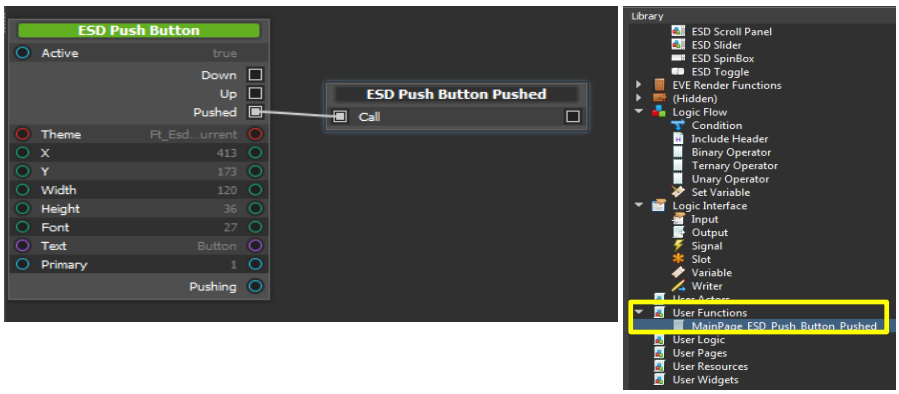


Figure 201 - ESD_METHOD Example

Copyright © Bridgetek Pte Ltd

Parameters	Description
Context	Pointing to the caller's context

Table 48 - Meta Macros - ESD_METHOD - Parameters

NOTE

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

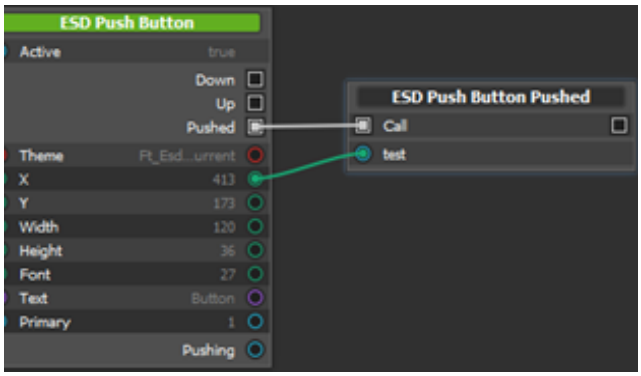


Figure 202 - Define ESD_FUNCTION

```
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;
```


Copyright © Bridgetek Pte Ltd

ESD_GLOBAL

ESD_GLOBAL is a built-in macro used to define global variables that can be accessed across different pages of a logic node. The variable declaration should be placed in the header file, while its definition must be placed in the 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.

Copyright © Bridgetek Pte Ltd

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 code is executed in the ESD simulation engine. If certain parts of the user's C code should not run in simulation, such as hardware-specific register access, users can use this macro to conditionally exclude those sections.

Add User Functions

Users can define custom functions by writing C source code directly. If the functions are defined using the predefined "ESD_FUNCTION" macro, they will appear under the "User Functions" category in the Library Browser. These functions can then be dragged and dropped into the Logic Node Editor for use.

Creating Source File

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

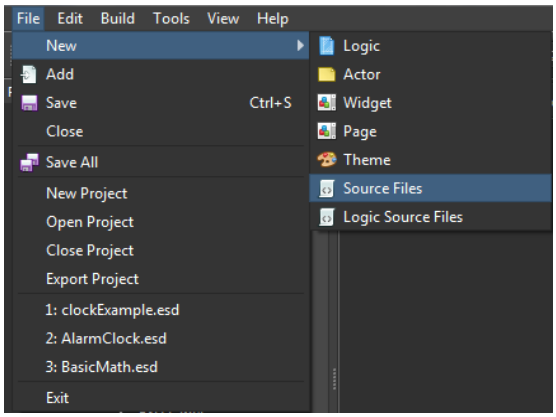


Figure 203 - Creating Source File

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

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


Copyright © Bridgetek Pte Ltd

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(){}
```

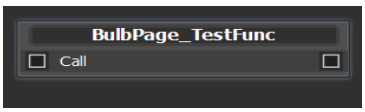


Figure 204 - ESD_FUNCTION Without Parameter Input and Return Value

- 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();
```

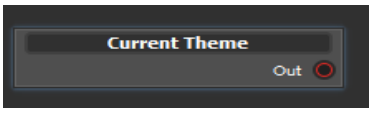


Figure 205 - ESD_FUNCTION Without Parameter Input

- 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) { //.. }
```

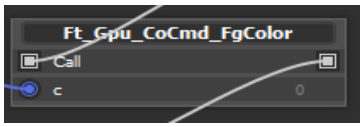


Figure 206 - ESD_FUNCTION Without Return Value

- 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)
{
    //..
}
```

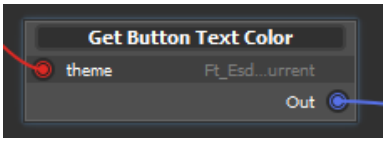


Figure 207 - ESD_FUNCTION With Parameter Input and Return Value

Copyright © Bridgetek Pte Ltd

F. Appendix A – List of Figures

Figure 1 - EmulatorProcess.exe error message 9
Figure 2 - Frozen simulation rotation icon 9
Figure 3 - ESD support platform 11
Figure 4 - Screen Designer - System Error 15
Figure 5 - Configure cross-compilation Environment 16
Figure 6 - EVE Screen Designer welcome page 17
Figure 7 - EVE Screen Designer information page 17
Figure 8 - EVE Screen Designer select destination location page 18
Figure 9 - EVE Screen Designer select additional tasks page 18
Figure 10 - EVE Screen Designer start installation 19
Figure 11 - Installation page 19
Figure 12 - EVE Screen Designer information page 19
Figure 13 - EVE Screen Designer complete installation 20
Figure 14 - ASTC applied for images and fonts 22
Figure 15 - Load resources from flash 22
Figure 16 - Use Unicode font with pre-defined character set 23
Figure 17 - Chinese font used in application 23
Figure 18 - ASTC Format 23
Figure 19 - DXT1 Bitmap formats 24
Figure 20 - DXT1 format bitmap in BitmapFormats example project 25
Figure 21 - DXT1 output format 25
Figure 22 - DXT1L2 output format 25
Figure 23 - DXT1PALETTED output format 26
Figure 24 - Font Converter dialog for BT82X 26
Figure 25 - Select Unicode Font Resource 28
Figure 26 - Unicode font applied for Label and Button 28
Figure 27 - Video Convert Dialog. 29
Figure 28 - Animation converter dialog. 30
Figure 29 - Reconvert Added Resource Files 31
Figure 30 - Mark resource as storage in Flash 32
Figure 31 - Upload Flash file to device 32
Figure 32 - LittleFS enable option in Project 33
Figure 33 - Select ESD_RESOURCE_FLASH 33
Figure 34 - C source file in the Project 34
Figure 35 - ESD_LittleFS.c file in ESD_Core 34
Figure 36 - Reading each entry 34
Figure 37 - Show all the image names in the project Asset browser 35
Figure 38 - Show preview image in the project AssetBrowser 35
Figure 39 - Config the memory 36
Figure 40 - Main Project properties 37
Figure 41 - Making Value Specific to Current Target 39
Figure 42 - Removing Target Specific Value 39
Figure 43 - Configure cross-compilation environments 40
Figure 44 - Export option 40
Figure 45 - Hidden folder 41
Figure 46 - Include Header 41
Figure 47 - Watch variables 41
Figure 48 - Inspector dialog 42
Figure 49 - File name keyword conflict error 42
Figure 50 - Error list 43

Copyright © Bridgetek Pte Ltd

Figure 51 - Error due to incorrect EVE platform selection 43
Figure 52 - ESD welcome dialog 44
Figure 53 - ESD new project dialog 44
Figure 54 - ESD Example Projects Dialog 45
Figure 55 - ESD Welcome Dialog Menu 46
Figure 56 - EVE Screen Designer User Interface Components 46
Figure 57 - Menu bar 47
Figure 58 - Toolbar 1 49
Figure 59 - Toolbar 2 49
Figure 60 - MCU Type is not applicable to selected platform 50
Figure 61 - CTP configuration 51
Figure 62 - Project Explorer window 51
Figure 63 - Project Explorer Filter 52
Figure 64 - Screen Layout Editor – App.main 52
Figure 65 - Screen Layout Editor - MainPage.c (C Source Code) 53
Figure 66 - Context menu on Tab bar 53
Figure 67 - Logic Node Editor 54
Figure 68 - Library Browser 54
Figure 69 - Error List Window 55
Figure 70 - Output Window 56
Figure 71 - Property Editor (ESD Push Button) 56
Figure 72 - Status bar 57
Figure 73 - Application Project Structure 59
Figure 74 - Application Workflow 60
Figure 75 - New and Open Project Menu 61
Figure 76 - Add Menu 61
Figure 77 - Add Existing file dialog 62
Figure 78 - Bitmap Converter dialog 62
Figure 79 - Property Editor 64
Figure 80 - memory pool logs 66
Figure 81 - New Target button 67
Figure 82 - Configure custom target 68
Figure 83 - Select custom build target 68
Figure 84 - Build & Upload 69
Figure 85 - Pico drive not connected alert 69
Figure 86 - Store resource in EVE SD card 74
Figure 87 - SD Card BT82X Example project 75
Figure 88 - Switch Page 79
Figure 89 - Sample Switch Page UI 79
Figure 90 - Properties Editor – Page Allocation 80
Figure 91 - Sample Source File 81
Figure 92 - Zoom In & Out 81
Figure 93 - Main File 82
Figure 94 - Sample Logic File 82
Figure 95 - C File / C Editor 83
Figure 96 - Logic Node Editor - Basic Logic Node 83
Figure 97 - Library Browser - Basic Logic Nodes 84
Figure 98 - Composite Logic Node 84
Figure 99 - Composite Logic Node - Page Node 85
Figure 100 - Logic Node Editor – Widget Node 85
Figure 101 - User Widget 86
Figure 102 - User Widget - Position & Size Properties 86
Figure 103 - Layout Widgets 88

Copyright © Bridgetek Pte Ltd

Figure 104 - Layout Widgets 88
Figure 105 - Actor Node 89
Figure 106 - Actor folder 89
Figure 107 - Logic Object 89
Figure 108 - Main Page 90
Figure 109 - Rendering Order Example 91
Figure 110 - Library Browser 92
Figure 111 - Library Filter 93
Figure 112 - ESD Layouts 94
Figure 113 - Utilities Layouts 95
Figure 114 - Get Horizontal Align 95
Figure 115 - Get Vertical Align 95
Figure 116 - Detach 95
Figure 117 - Get Global Height 96
Figure 118 - Get Global Width 96
Figure 119 - Get Global X 96
Figure 120 - Get Global Y 96
Figure 121 - Insert Above 97
Figure 122 - Insert Below 97
Figure 123 - Insert Bottom 97
Figure 124 - Insert at Top 98
Figure 125 - Is Bottom 98
Figure 126 - Is Top 98
Figure 127 - Is Visible 98
Figure 128 - Sort Highter 99
Figure 129 - Sort Lower 99
Figure 130 - Sort to Bottom 99
Figure 131 - Sort to Top 99
Figure 132 - Switch Page 100
Figure 133 - Switch Page Layout 100
Figure 134 - Switch Page Resource Allocation 101
Figure 135 - Dynamic Switch Page 101
Figure 136 - Dynamic Switch Page Layout 102
Figure 137 - Dynamic Page Resource Allocation 102
Figure 138 - Auto Switch Widget 103
Figure 139 - Auto Switch Page Layouts 103
Figure 140 - Fixed Positioning 104
Figure 141 - Fixed Positioning Layouts 104
Figure 142 - Fill Layout 105
Figure 143 - Widget rendering based on depth sort in Fill Layout 105
Figure 144 - Linear Layout 106
Figure 145 - Example of ESD_ORIENTATION_HORIZONTAL layout 107
Figure 146 - Example of ESD_ORIENTATION_VERTICAL layout 107
Figure 147 - Stretch Layout 108
Figure 148 - Example of horizontal space stretching using a stretch layout 108
Figure 149 - Example of widget stretching with different stretch weight values 108
Figure 150 - Scroll Layout 109
Figure 151 - Scroll Switch Page Layout 110
Figure 152 - Sample Scroll Switch Page Implementation 111
Figure 153 - Signal Switch Page Layout 112
Figure 154 - ESD Render Functions 113
Figure 155 - ESD Elements 113
Figure 156 - ESD Circle Element 113

Copyright © Bridgetek Pte Ltd

Figure 157 - ESD Panel Element 113
Figure 158 - ESD Gradient Panel Element 114
Figure 159 - Primitives 114
Figure 160 - Bitmap Primitive 115
Figure 161 - Example of bitmap resource added to the library 115
Figure 162 - Example of using bitmap cell 116
Figure 163 - ESD Line 116
Figure 164 - ESD Rectangle 117
Figure 165 - ESD Display List 117
Figure 166 - Library Browser - ESD Theme 118
Figure 167 - ESD Built-in Themes 118
Figure 168 - User Theme Property 119
Figure 169 - Example of applying a theme to widgets 119
Figure 170 - User Resources in the Library Browser 119
Figure 171 - ESD Utilities 120
Figure 172 - ESD Idle Checker 121
Figure 173 - Logic Flow 121
Figure 174 - Logic Flow - Condition Node 122
Figure 175 - Logic Flow - Binary Condition Node 122
Figure 176 - Logic Flow - Local Method 123
Figure 177 - Logic Flow - Binary Operator Node 124
Figure 178 - Logic Flow - Ternary Operator Node 125
Figure 179 - Logic Flow - Unary Operator Node 125
Figure 180 - Logic Flow - Sequence Node 126
Figure 181 - Logic Flow - Switch Node 126
Figure 182 - Logic Flow - Switch Value 127
Figure 183 - Logic Flow - Set Variable Node 127
Figure 184 - Logic Flow - Watch Variable Node 128
Figure 185 - Logic Interface 128
Figure 186 - Logic Interface - Input Node 129
Figure 187 - Logic Interface - Output Node 129
Figure 188 - Logic Interface - Signal Node 130
Figure 189 - Logic Interface - Slot Node 130
Figure 190 - Logic Interface - Variable Node 131
Figure 191 - Logic Interface - Writer Node 132
Figure 192 - Widget Interface 132
Figure 193 - Property Editor 133
Figure 194 - ESD Common Properties 134
Figure 195 - Name Property 134
Figure 196 - Depth Sort Property 135
Figure 197 - Right Click Menu 135
Figure 198 - Active Property 135
Figure 199 - User Functions Example 138
Figure 200 - Toggle LED 139
Figure 201 - ESD_METHOD Example 139
Figure 202 - Define ESD_FUNCTION 140
Figure 203 - Creating Source File 142
Figure 204 - ESD_FUNCTION Without Parameter Input and Return Value 143
Figure 205 - ESD_FUNCTION Without Parameter Input 143
Figure 206 - ESD_FUNCTION Without Return Value 143
Figure 207 - ESD_FUNCTION With Parameter Input and Return Value 143

Copyright © Bridgetek Pte Ltd

G. Appendix B – List of Tables

Table 1 - Terms and Abbreviations Description 9
Table 2 - Platform vs Module Compatibility 12
Table 3 - Connection between STM32F401RET6 and EVE 13
Table 4 - Connection between STM32F401RET6 and SD CARD 14
Table 5 - Installation Folder 21
Table 6 - Menu & Description 49
Table 7 - Toolbar & Description 50
Table 8 - Context Menu 53
Table 9 - ESD Libraries 55
Table 10 - Project File Type Description 60
Table 11 - Bitmap Resource Properties 65
Table 12 - Simulation Description 66
Table 13 - Output files for different Build Configuration & Host Platforms 70
Table 14 - Build Targets, Host platform, and the corresponding export projects 71
Table 15 - List of Folders & Files created upon Exporting Project 72
Table 16 - Common Resource Metadata Format 76
Table 17 - Bitmap Resource Metadata Format 76
Table 18 - Animation Resource Metadata Format 77
Table 19 - Font Resource Metadata Format 78
Table 20 - Page Allocation Modes Overview 80
Table 21 - Connection End Types 90
Table 22 - ESD Layouts Description 94
Table 23 - Switch Page Layout Properties 100
Table 24 - Dynamic Switch Page Layout Properties 101
Table 25 - Auto Switch Page Layout Properties 103
Table 26 - Fixed Positioning Layout Properties 104
Table 27 - Fill Layout Properties 105
Table 28 - Linear Layout Properties 106
Table 29 - Stretch Layout Properties 108
Table 30 - Scroll Layout Properties 109
Table 31 - Scroll Switch Page Layout Properties 111
Table 32 - Signal Switch Page Layout Properties 112
Table 33 - ESD Circle Element Properties 113
Table 34 - ESD Panel Element Properties 114
Table 35 - ESD Gradient Panel Element Properties 114
Table 36 - ESD Bitmap Properties 115
Table 37 - ESD Line Properties 117
Table 38 - ESD Rectangle Properties 117
Table 39 - Ft_Esd_Theme_LightBlue Theme 118
Table 40 - Ft_Esd_Theme_DarkOrange Theme 119
Table 41 - ESD Utilities & Description 120
Table 42 - Binary Operators Supported by ESD 4.14 124
Table 43 - Unary Operators Supported by ESD 125
Table 44 - Built-in Slots 130
Table 45 - Widget Interface Properties 133
Table 46 - Macros - ESD_TYPE – Parameters 137
Table 47 - Macros - ESD_FUNCTION - Parameters 139
Table 48 - Meta Macros - ESD_METHOD - Parameters 140

Copyright © Bridgetek Pte Ltd

H. Appendix C – Revision History

Document Title	:	BRT_AN_029 EVE Screen Designer 5.3.0 User Guide
Document Reference No.	:	BRT_000218
Clearance No.	:	BRT#159
Product Page	:	ESD - IC & Module
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 Raspberry 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

Copyright © Bridgetek Pte Ltd

Version 1.9	User Guide updated as per ESD4.19.2 version features and enhancements	23-10-2024
Version 1.10	User Guide updated as per ESD 4.19.6 version features and enhancements	26-03-2025
Version 1.11	User Guide updated as per ESD 5.0.0 version features and enhancements	09-09-2025
Version 1.12	User Guide updated as per ESD 5.1.0 version features and enhancements	16-01-2026
Version 1.13	User Guide updated as ESD 5.2.0 version features and enhancements	22-04-2026
Version 1.14	User Guide updated as ESD 5.2.1 version features and enhancements	12-05-2026
Version 1.15	User Guide updated as ESD 5.3.0 version features and enhancements	21-09-2026

Copyright © Bridgetek Pte Ltd