USER GUIDE



**EVE Screen Editor 4.4 User Guide3**

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. Your statutory rights are not affected. This product or any variant of it is not intended for use in any medical appliance, device, or system in which the failure of the product might reasonably be expected to result in personal injury. This document provides preliminary information that may be subject 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, 178, Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number 201542387H. © Bridgetek Pte Ltd.

**Document Version: 1.4**

**Issue Date: 05-07-2022**

**Contents**

[I. Preface 4](#_Toc107923904)

[A. Purpose 4](#_Toc107923905)

[B. Intended Audience 4](#_Toc107923906)

[C. Related Documents 4](#_Toc107923907)

[D. Feedback 4](#_Toc107923908)

[II. Overview 5](#_Toc107923909)

[A. Introduction 5](#_Toc107923910)

[B. Supported Devices 5](#_Toc107923911)

[C. Key Features 6](#_Toc107923912)

[D. What’s new in ESE 4.4? 6](#_Toc107923913)

[E. Known Issues & Limitations 7](#_Toc107923914)

[F. Credits 7](#_Toc107923915)

[1. Open-Source Software 7](#_Toc107923916)

[2. Icons Copyright 7](#_Toc107923917)

[III. Setup & Installation 8](#_Toc107923918)

[A. System Requirements 8](#_Toc107923919)

[B. Hardware Requirements 9](#_Toc107923920)

[C. Dependencies / Pre-Requisites 9](#_Toc107923921)

[D. Installing ESE 9](#_Toc107923922)

[E. Installation Folder 11](#_Toc107923923)

[IV. The Graphical User Interface 12](#_Toc107923924)

[A. Overview 12](#_Toc107923925)

[B. Menu bar, Toolbar, and Status bar 13](#_Toc107923926)

[1. Menu bar 13](#_Toc107923927)

[2. Toolbar 15](#_Toc107923928)

[3. Status Bar 16](#_Toc107923929)

[C. Editors and Inspector 17](#_Toc107923930)

[1. Coprocessor Command Editor 17](#_Toc107923931)

[2. Display List Editor 18](#_Toc107923932)

[3. Inspector 19](#_Toc107923933)

[D. Toolbox, Content Manager, and Registers 21](#_Toc107923934)

[1. Toolbox 21](#_Toc107923935)

[2. Registers 26](#_Toc107923936)

[3. Content Manager 27](#_Toc107923937)

[E. Devices, Controls, Properties, and Output 31](#_Toc107923938)

[1. Device Manager 31](#_Toc107923939)

[2. Controls 32](#_Toc107923940)

[3. Properties 33](#_Toc107923941)

[4. Output 33](#_Toc107923942)

[F. View Port 34](#_Toc107923943)

[G. Navigator 34](#_Toc107923944)

[H. Project settings 35](#_Toc107923945)

[I. Keyboard Shortcuts 38](#_Toc107923946)

[V. Quick Start Tutorials 39](#_Toc107923947)

[A. Capture Display List 39](#_Toc107923948)

[B. Change the color 39](#_Toc107923949)

[C. Import the content 41](#_Toc107923950)

[D. Import the flash 43](#_Toc107923951)

[E. Open the project 44](#_Toc107923952)

[F. Save your design 45](#_Toc107923953)

[G. Export the project 45](#_Toc107923954)

[H. Custom Fonts 48](#_Toc107923955)

[I. Constrain either horizontal or vertical positioning when dragging an object 50](#_Toc107923956)

[1. Constrain vertical positioning 50](#_Toc107923957)

[2. Constrain horizontal positioning 51](#_Toc107923958)

[J. Import/Export Memory Dump File 52](#_Toc107923959)

[VI. Command Usage Examples 53](#_Toc107923960)

[A. CMD\_PLAYVIDEO 53](#_Toc107923961)

[B. CMD\_LOADIMAGE 54](#_Toc107923962)

[C. CMD\_SETBITMAP 55](#_Toc107923963)

[D. CMD\_SNAPSHOT 56](#_Toc107923964)

[E. CMD\_SKETCH 56](#_Toc107923965)

[F. CMD\_SNAPSHOT2 57](#_Toc107923966)

[G. VERTEX\_TRANSLATE\_X/Y 58](#_Toc107923967)

[H. CMD\_MEDIAFIFO 59](#_Toc107923968)

[I. CMD\_SETBASE 59](#_Toc107923969)

[J. CMD\_ROMFONT 60](#_Toc107923970)

[K. PALETTE\_SOURCE 60](#_Toc107923971)

[L. CMD\_SETFONT2 61](#_Toc107923972)

[M. CMD\_TEXT 62](#_Toc107923973)

[VII. Working with EVE Screen Editor 63](#_Toc107923974)

[A. Connect with Hardware 63](#_Toc107923975)

[B. Check your design 65](#_Toc107923976)

[1. Step by Step 65](#_Toc107923977)

[2. Trace the pixel 66](#_Toc107923978)

[3. Drag and Drop 67](#_Toc107923979)

[C. Example Project 68](#_Toc107923980)

[VIII. Disclaimer 69](#_Toc107923981)

[IX. Contact Information 70](#_Toc107923982)

[Appendix A - References 71](#_Toc107923983)

[Document References 71](#_Toc107923984)

[Acronyms & Abbreviations 71](#_Toc107923985)

[Appendix B – List of Figures & Tables 72](#_Toc107923986)

[List of Figures 72](#_Toc107923987)

[List of Tables 74](#_Toc107923988)

[Appendix C – Revision History 75](#_Toc107923989)

# Preface

## Purpose

This document describes the functionality and procedures involved in using the application **E**VE **S**creen **E**ditor (ESE).

## Intended Audience

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

## Related Documents

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Document Type** | **Document Format** |
| [FT81x Series Programmers Guide](http://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT81X_Series_Programmer_Guide.pdf) | Programming Guide | PDF |
| [FT81x Datasheet](http://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_FT81x.pdf) | Datasheet | PDF |
| [FT800 Series Programmers Guide](http://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT800_Series_Programmer_Guide.pdf) | Programming Guide | PDF |
| [FT800 Embedded Video Engine Datasheet](http://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_FT800_Embedded_Video_Engine.pdf) | Datasheet | PDF |
| [BT81x Datasheet](https://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_BT81X.pdf) | Datasheet | PDF |
| [BT81X Series Programmer Guide](https://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/BRT_AN_033_BT81X_Series_Programming_Guide.pdf) | 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](mailto:docufeedback@brtchip.com?subject=BRT_AN_029%20Version%201.0). For any additional technical support, refer to <http://brtchip.com/contact-us/>.

# Overview

## Introduction

**E**VE **S**creen **E**ditor **(ESE)** is a GUI tool that provides an intuitive "drag & drop" user experience to construct screen design without programming. Empowered by the cutting-edge EVE emulator, ESE gives users the maximum fidelity of graphics effect.

Co-processor commands and display lists can also be provided as input in the editor window to construct the desired screen design. As a result, it dramatically lessens the learning curve of EVE features.

This tool is platform-independent so the screen design can be created without considering the details of the MCU. Users have the option to export the design to hardware platform-specific source code. This dramatically reduces the effort to start a new project on real hardware.

If users have EVE Series boards and an [FT4222](https://ftdichip.com/products/umft4222ev/)/[MPSSE](https://ftdichip.com/products/c232hm-edhsl-0/) cable, the screen design can be synchronized with the actual hardware with a few mouse clicks.

Last, but not least, there are more exciting features, such as "tracing and step by step", waiting to be discovered.

Let us get started!

## Supported Devices

ESE supports all series of EVE chips, including FT80X, FT81X, and BT81X.

ESE also supports EVE modules which are listed below.

* For [***Exporting Feature***]:

|  |  |  |
| --- | --- | --- |
| EVE Chip Family | Platform | EVE Modules |
| FT80X | Arduino Projects | *ADAM\_4DLCD\_FT843, Breakout\_4DLCD\_FT843, VM800B35, VM800P35, VM800P43\_50, VM801B43, VM801P43\_50* |
| EVE Hal Projects | *VM800B35, VM800B43\_50, VM800BU35, VM800BU43\_50, VM800C35, VM800C43\_50, VM801B43\_50A* |
| GameDuino2 Projects | *GameDuino2* |
| FT81X | EVE Hal Projects | *ME810A\_HV35R, ME810A\_WH70R, ME810AU\_WH70R, ME811A\_WH70C, ME811AU\_WH70C, ME812A\_WH50R, ME812AU\_WH50R, ME813A\_WH50C, ME813A\_WV7C, ME813AU\_WH50C, VM810C50* |
| BT815/6 | EVE Hal Projects | *VM816C50A, VM816CU50A* |
| BT817/8 | EVE Hal Projects | *ME817EV* |

* For [***Device Sync***]:

|  |  |
| --- | --- |
| Host Platform | EVE Modules |
| FT4222, MPSSE | ME817EV-WH70C |
| FT4222, MPSSE | ME817EV-WH10C |
| FT4222 | VM816CU50A |
| FT4222, MPSSE | VM816C50A |
| FT4222 | ME813AU-WH50C |
| FT4222, MPSSE | ME812A-WH50R |
| FT4222, MPSSE | VM810C50A |
| MPSSE | VM800B |

## Key Features

The following are some of the key features of EVE Screen Editor:

* **WYSIWYG** GUI
* High-level widgets
* No EVE display list knowledge required
* Widget-based GUI construction
* Drag and drop widget to create screen layout
* Exporting project

## What’s new in ESE 4.4?

* Add the appropriate bitmap setup when dragging and dropping CMD\_SKETCH into the emulator viewport
* Improve zoom-in precision
* Display the RGBA value along with the coordinate position of the cursor in the Emulator screen
* Enable drag and drop to open an existing project in ESE
* Update LibFT4222-64.dll to remove the dependency
* Show a warning message when the memory size required by CMD\_SNAPSHOT2 is too large
* Give more explanation for the escape character usage in CMD\_TEXT
* Removed glitched bitmap line in BT817 and BT818
* Fixed an issue that occurs when dragging and dropping an existing item in the Content manager

## Known Issues & Limitations

The following are some of the known issues and limitations in this release:

1. Device sync up feature: CMD\_PLAYVIDEO without OPT\_MEDIAFIFO is not supported
2. No Unicode support of file name for image files (PNG or JPG) in export feature, although the files can be imported in content manager successfully.
3. CMD\_SNAPSHOT2 is not fully supported.
4. Properties window of CMD\_LOADIMAGE and CMD\_PLAYVIDEO does not show a warning message when an incorrect file name is entered into the streamline box.
5. Export feature may fail if the project file path is more than 255 characters.
6. Playing an invalid video stream or animation frame or graphic data may cause the ESE to hang.

## Credits

### Open-Source Software

* Qt: <http://doc.qt.io/qt-5/licensing.html> under LGPL.
* Python: <https://www.python.org> under GPL- compatible.
* Pillow: <https://pillow.readthedocs.io/en/stable/index.html>
* zlib: <https://zlib.net/>
* libpng: <http://www.libpng.org/pub/png/libpng.html>
* FreeType: <https://www.freetype.org> under GPL license.

### Icons Copyright

Some of the icons used in ESE are from: <http://p.yusukekamiyamane.com/icons/search/fugue/> used in compliance with the Creative Commons Attribution 3.0 License.

# Setup & Installation

## System Requirements

To install ESE 4.4 application, ensure that your system meets the requirements recommended below:

* RAM: at least 1G RAM
* CPU: Multi-core is recommended
* Hard disk: More than 500MB of free space
* OS: Windows 7 and above, 64-bit platform
* Display resolution: At least 1280 by 800 pixels

We strongly recommend an administrator user account to run this application.

To work with the export feature, users are recommended to install the following software:

* Arduino IDE
* Gameduino 2 library
* EVE Arduino Library (1.2.0 and above)
* Microsoft Visual Studio C++ 2010 IDE or newer is required to compile the HAL MSVC projects.
* Microsoft Visual Studio C++ 2012 IDE is required to run HAL FT800 Emulator projects.

To build and verify projects on Arduino IDE, the following boards are needed:

* VM800P/VM801P (3.5", 4.3" or 5.0" display) for exported EVE Arduino library-based project
* Gameduino 2 board with Arduino Pro board for exported Gameduino2 library-based project

## Hardware Requirements

The following hardware can be used to verify the design in the device manager:

|  |  |
| --- | --- |
| Board | Host Platform |
| ME817EV-WH70C | FT4222, MPSSE |
| ME817EV-WH10C | FT4222, MPSSE |
| VM816CU50A | FT4222 |
| VM816C50A | FT4222, MPSSE |
| ME813AU-WH50C | FT4222 |
| ME812A-WH50R | FT4222, MPSSE |
| VM810C50A | FT4222, MPSSE |
| VM800B | MPSSE |

Note: For VM816C50A, ME812A-WH50R, VM810C50A, and VM800B, a separate USB-MPSSE adapter is required.

## Dependencies / Pre-Requisites

* **Visual C++ Redistributable for Visual Studio 2017**

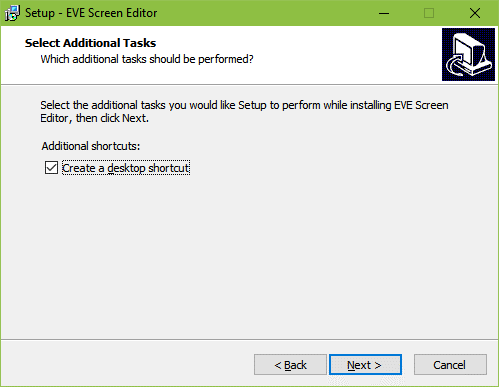
If the PC does not have Microsoft Visual Studio 2017 installed, Visual C++ Redistributable is required. Users can download this from:

<https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads>

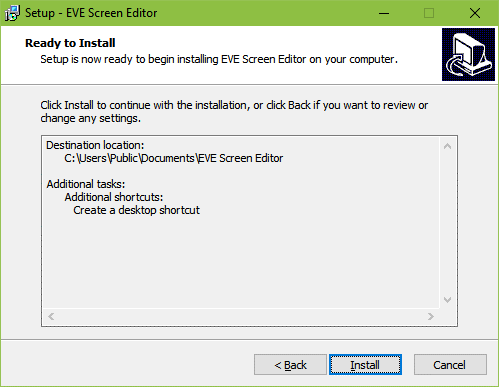
## Installing ESE

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

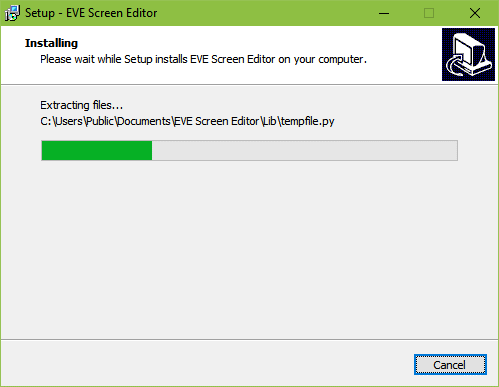
1. Download the installation package from the link <https://brtchip.com/ic-module/toolchains/#EVEScreenEditor>.
2. When prompted with a download dialog box. Click on **Save**.
3. Navigate to the folder under which the package files are downloaded.
4. Extract the zip file contents. Double click on the executable file
5. Select a “Destination Folder” for installing the files. Accept the default folder or click **Browse** to specify a different location. Click **Next** to confirm the destination folder and continue.
6. In the **Select Additional Tasks** window, check the **“Create a desktop”** boxes, to have the ESE icon displayed on the desktop if required. Click **Next** to prepare for the installation.



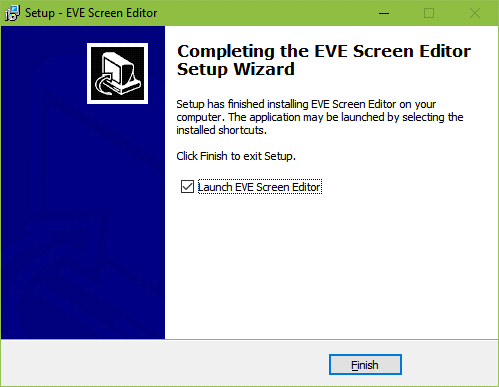
1. The initial setup is completed, and the application is ready to be installed.



1. Click **Install** to start the installation. A progress bar indicates that the installation is in progress.



1. Upon successful installation, click **Finish**. The ESE application UI is displayed.



## Installation Folder

The following table provides a list of folders that can be found under the installation path upon successful installation of ESE.

|  |  |  |
| --- | --- | --- |
| Folder Name | Description | Permission |
| *Examples* | The example projects created by ESE | Read/Write |
| *imageformats* | Qt run-time DLLs for image format supporting | Read-Only |
| *Styles* | Qt DLL for Windows style | Read-Only |
| *Lib* | Python library | Read-Only |
| *Manual* | Contains this document | Read-Only |
| *platforms* | Qt platform run-time DLLs. | Read-Only |
| *EVE\_Hal\_Library* | The template project for BT81X | Read-Only |
| *untitled* | The template project used by ESE | Read-Only |
| *device\_sync* | Contain information of built-in and custom boards | Read/Write |
| *export\_scripts* | Contain Python scripts to export project | Read-Only |
| *firmware* | Contain flash blob for BT815 and BT817 | Read-Only |

Table 1 - Installation Folder

# The Graphical User Interface

## Overview

ESE user interface has the following components:

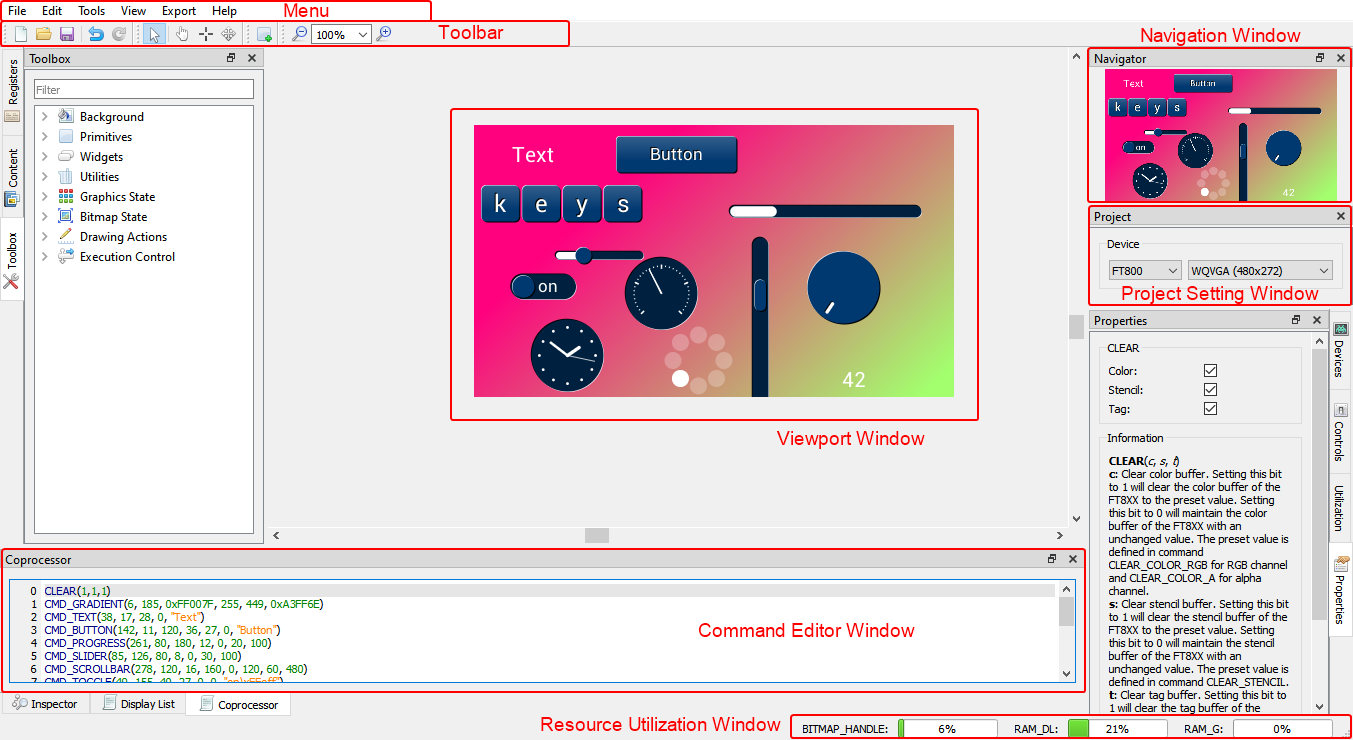


Figure 1 - User Interface Components

## Menu bar, Toolbar, and Status bar

The menu bar consists of *File, Edit, View, Tools, Export,* and *Help*, each with a drop-down list of available functions.



### Menu bar

#### File Menu

The *File* menu is the first item that appears as part of the menu bar. It contains a list of commands that are used for handling files such as *New, Open, Save* etc.



Fig2 - File Menu

|  |  |
| --- | --- |
| **Menu Item** | **Description** |
| *New* | To create a new project, clear the screen |
| *Open* | To open / retrieve the existing project. The file extensions can be ".ese", ".ft8xxproj" or ".ft800proj". Example projects can be viewed here |
| *Save* | To save the screen design in the user-specified location. The file is saved with an “.ese” extension. |
| *Save As* | To choose a different destination and file name to save the current project. The file is saved with an ".ese" extension. |
| *Browse Project Folder* | To open the project folder where the current project file exists |
| *Save Screenshot* | To save the snapshot of the screen (i.e. currently in use) into the local PC |
| *Save Display List* | To save the current display list to a text file, in Little Endian or Big-Endian format |
| *Save Co-Processor Command* | To save current coprocessor command to a text file, in Little Endian or Big-Endian format |
| *Recent Projects History* | To view the recently opened projects. The latest 5 opened projects are added into history list. Clicking on the history item will reopen the corresponding project again. |
| *Quit* | To exit the application. |

#### Edit Menu

The *Edit* menu contains a list of commands that are used for handling information within a file such as *Undo* and *Redo*.

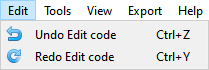


Figure 3 - Edit Menu

|  |  |
| --- | --- |
| **Menu Item** | **Description** |
| *Undo* | To Undo the last action done in the editor. |
| *Redo* | To Redo the last action done in the editor. |

#### Tools menu

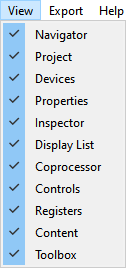
The *Tools* menu contains a list of commands that are used for configuring software such as *Reset Emulator* and *Capture Display List.*



Figure 4 - Tool Menu

|  |  |
| --- | --- |
| **Menu Item** | **Description** |
| *Reset Emulator* | To reset the emulator that is running in the background |
| *Capture Display List* | To extend the EVE co-processor commands in order to display the list commands in the display list editor |

#### View Menu



The view menu enables users to *hide* or *show* the sub windows in the editor. Each of the sub windows can be docked to a different side of the main window as well as can be a stand-alone floating window. Selecting an option ensures that the corresponding window is displayed. Clearing the selection hides the corresponding window.

Figure 5 - View Menu

#### Export Menu

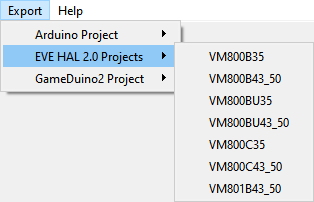


Figure 6 - Export Menu

Internally, ESE has a Python engine built-in and employs the Python script to export the co-processor commands to a project.

For FT80X based projects, there are scripts to export it to Gameduino2, EVE Arduino, and HAL based projects.

For FT81X / BT81X based projects, there are scripts to export it to a HAL 2.0 based project.

#### Help Menu

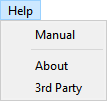


Figure 7 - Help Menu

|  |  |
| --- | --- |
| **Menu Item** | **Description** |
| *Manual* | To display the EVE Screen Editor user guide |
| *About* | To view the version information of EVE Screen Editor |
| *3rd Party* | To view information about the copyright of 3rd party software or artefact including Qt Software and Figure icons. |

### Toolbar

The toolbar defines shortcuts of mouse operation for *New*, *Open*, *Save*, *Undo*, *Redo*, *Cursor, Touch, Trace, Edit, Insert, Zoom Out and Zoom In* functions.



Figure 8 - Toolbar

|  |  |
| --- | --- |
| **Toolbar Item** | **Description** |
| *New* | To create a new project. (Clears the editor and starts a new project in a temporary directory) |
| *Open* | To open an existing project |
| *Save* | To save the current project |
| *Undo* | To revoke the last operation |
| *Redo* | To redo a revoked / undone operation |
| *Cursor* | Automatic context-dependent cursor switching in viewport.  Cursor mode will automatically switch between *Touch/Trace/Edit* cursors depending on the context and exit a special context-specific mode upon right clicking.  Most cursor actions (such as inserting points or trace) can be ended by right-clicking in the viewport.  Shortcut keys: Alt + C |
| *Touch* | Force touch cursor in viewport  Enable mouse click on the viewport to simulate touch action on the touch panel connected to EVE touch engine.  Therefore, the touch-related registers are updated in the inspector. It is especially useful for CMD\_SKETCH.  Shortcut keys: Alt + T |
| *Trace* | To force trace cursor in viewport. See **Trace the pixel** for more details.  Shortcut keys: Alt + R |
| *Edit* | To force widget editing cursor in viewport  Shortcut keys: Alt + E |
| *Insert* | To insert duplicates of currently selected widget or primitive at clicked position, overrides any current cursor selection |
| *Zoom Out* | To zoom out emulator viewport |
| *Zoom In* | To zoom in emulator viewport |
| *Zoom Rate* | To select concrete zooming rate |

### Status Bar

The status bar shows the consumption status of **RAM\_G** and **RAM\_DL** as well as bitmap handles. It also shows the cursor position and pixel color in **(A, R, G, B)** format.



Figure 9 - Status Bar

For BITMAP\_HANDLE, RAM\_DL, and RAM\_G, the ratio of the exact number of resources used and the total will be displayed when a mouse hovers over it.



Figure 10 - Status Bar when a mouse hovers

## Editors and Inspector

This section discusses in detail how the Editors and Inspector window, which is located at the bottom of the main window, functions.

**Editor**

Editors provide individual windows to co-processor commands and display list commands which are sent to the EVE co-processor RAM\_CMD and EVE graphics engine RAM\_DL, respectively.

Please note that the co-processor command editor is the primary editor window since it supports editing the full command set of EVE, including co-processor commands and display list commands.

**Inspector**

Inspector displays the content of RAM\_DL and RAM\_REG and cannot be edited. RAM\_DL and RAM\_REG can be selected line by line and then copied to another text editor.



Figure 11 – Inspector

### Coprocessor Command Editor



Figure 12 - Coprocessor Command Editor

The features are as below:

• Full set commands support and auto-completion

• Decimal and hexadecimal values for parameters

• Error highlights

• Step-by-step emulation

**Note:**

* CLEAR command is auto inserted when ESE is launched.
* CMD\_CALIBRATE/CMD\_LOGO/CMD\_SPINNER commands will pause the following commands and shall be the last command in the editor.

### Display List Editor



Figure 13 - Display List Editor

The features are as below:

• Display list commands auto-completion

• Decimal and hexadecimal values for parameters

• Error highlights

• Step-by-step emulation

**Note:**

* Co-processor Command Editor has higher priority, and its content overrides the content of the display list editor. To validate the input of the display list editor, ensure that the co-processor command editor window does not contain any commands.
* By default, the Display List editor is hidden.

### Inspector

This section discusses the functions of the Inspector in EVE Screen Editor.



Figure 14 - Inspector

#### RAM\_DL

This window reflects the content of the RAM\_DL. It shows each 4-byte command in hexadecimal as well as text format, from lower to high address.

Please note they are read-only.



Figure 15 - RAM\_DL

RAM\_DL can be selected line by line and then copied to another text editor.



Figure 16 - Select rows in RAM\_DL

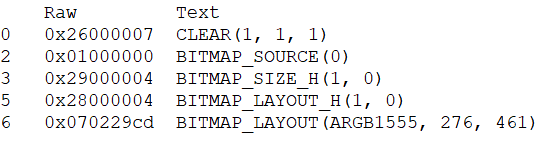


Figure 17 - Paste selected rows in RAM\_DL

#### RAM\_REG

The RAM\_REG window in the Inspector tab shows the register address, register name, and current register value in hexadecimal and decimal.

Please note that they are **read-only**.



Figure 18 - RAM\_REG

RAM\_REG can be selected line by line and then copied to another text editor.

****

Figure 19 - Select rows in RAM\_REG



Figure 20 - Paste selected rows in RAM\_REG

## Toolbox, Content Manager, and Registers

This section illustrates the Toolbox, Content Manager, and Register features of ESE. These features are available in the window on the left side of the viewport.

### Toolbox

The Toolbox is the portal to access the co-processor or display list commands. When the Display List editor is in focus, the display list commands are available in the Toolbox. When the Co-processor editor is in focus, the full set of display list and co-processor commands are available in the Toolbox. Users may drag and drop the commands from the Toolbox into the viewport.

#### Filter

The toolbox supports a filter powered by regular expression.

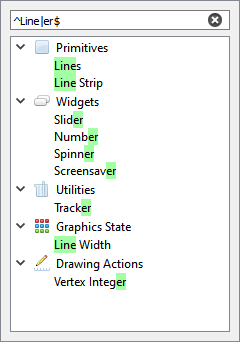


Figure 21 - Toolbox Filter

#### Display list mode

Select the display list editor window to use the display list mode:



Figure 22 - Display List mode

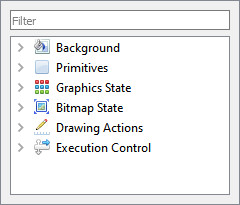
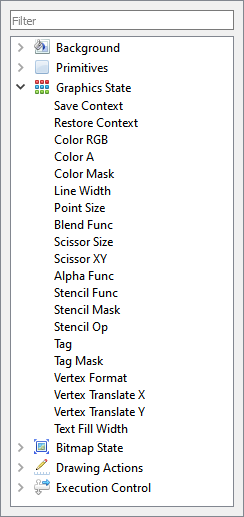
The toolbox will be enabled as below:

Figure 23 - Toolbox in Display List mode



All display list commands are grouped into various categories based on functionality (as in the FT81X project):

* Background
* Primitives
* Graphic State
* Bitmap State
* Drawing Actions
* Execution Control

Figure - Graphics State in Display List mode

Figure – Primitive in Display List Mode

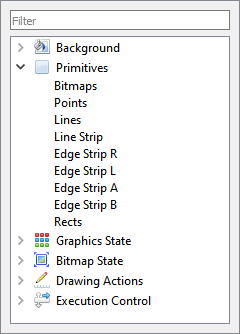
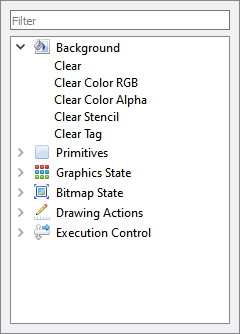


Figure – Background in Display List mode

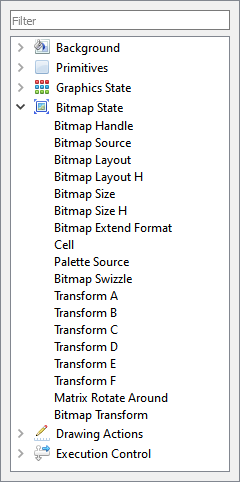
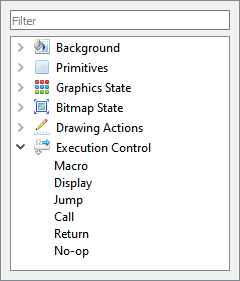
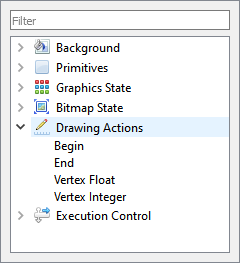


Figure - Drawing Actions in Display List mode

mode

Figure - Execution Control in Display List mode

Figure – Bitmap State in Display List mode

#### Coprocessor mode

To use the coprocessor mode, the coprocessor editor window shall be selected as below:



Figure 30 - Coprocessor mode

The toolbox will be enabled as below:

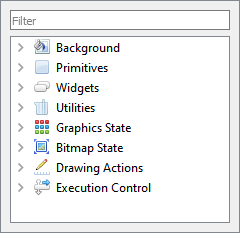
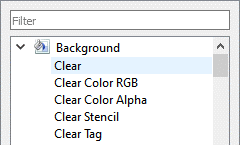
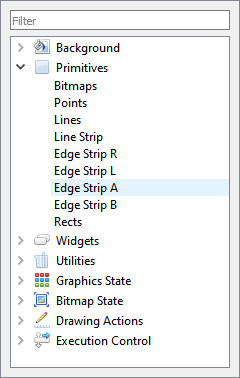


Figure 31 - Toolbox in Coprocessor mode

All commands are grouped into various categories based on functionality (as in the FT81X project):

• Background

• Primitives

• Widgets

• Utilities

• Graphics State

• Bitmap State

Figure - Background in Co-processor mode

• Drawing Actions

• Execution Control

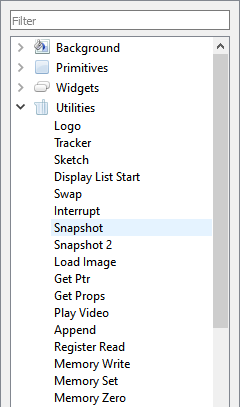
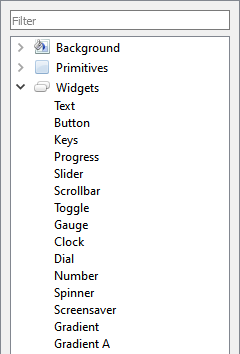


Figure - Primitives in Co-processor mode

Figure - Utilities in Co-processor mode

Figure - Widgets Co-processor mode

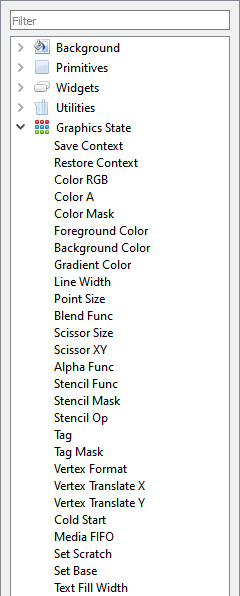
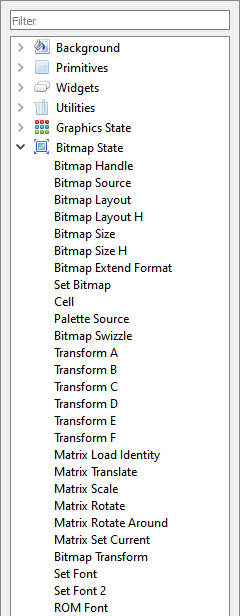
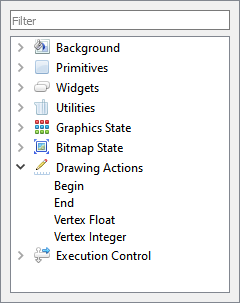
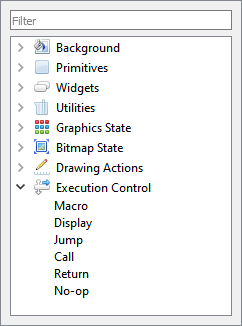


Figure – Execution Control in Coprocessor mode

Figure – Drawing Actions in Coprocessor mode

Figure – Bitmap State in Coprocessor mode

Figure - Graphics State in Coprocessor mode

### Registers

This tab is used to set up screen size and macro registers. The REG\_MACRO0 and REG\_MACRO1 registers can be edited in the editor box of Macro, the registers should be set using the display list command syntax. The vertical and horizontal size (REG\_VSIZE and REG\_HSIZE) of the screen can also be edited and the viewport will be updated accordingly.



Figure 40 – Register

ESE can simulate the custom resolution up to 2048 by 2048, which can be done using the register window. However, users shall note that this is for simulation purposes only and not for the physical hardware platform.

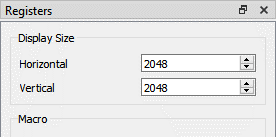


Figure 41 - Customize resolution in the Register window

Top of Form

Bottom of Form

### Content Manager

This section provides information about the content manager feature of ESE. The Content Manager allows users to import the assets (PNG, JPG files, or raw data) on the PC to RAM\_G by converting the format behind the scenes.



Figure 42 - Content Manager

#### Add the content

Content Manager enables users to add bitmaps and raw data to be loaded into the specific addresses in RAM\_G.

To perform this function, follow the steps below:

1. Click **[Add]** in the *Content* tab.
2. The Load Content dialog pops up. Browse and select the file to be added.

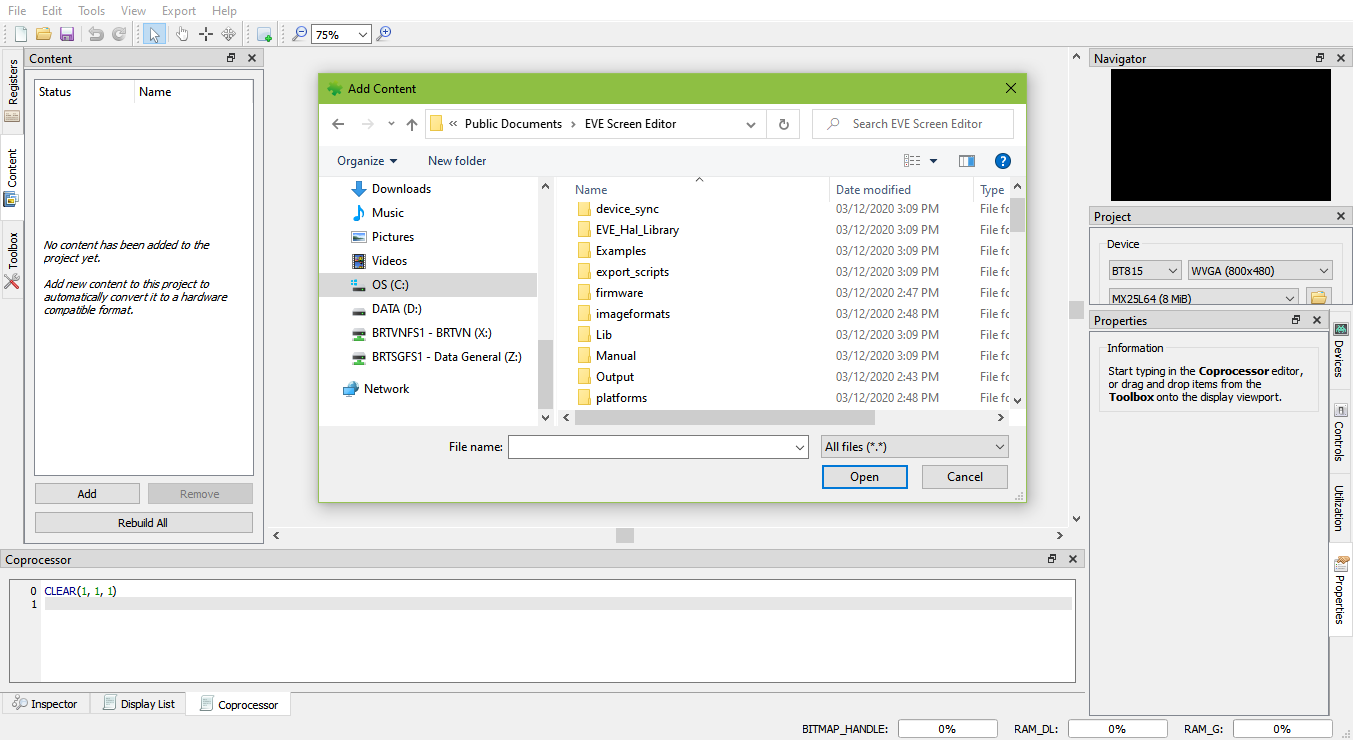


Figure 43 - Load content dialog

**Note:** The user can drag a content file from the PC and drop it into the Content Manager window.

1. Upon adding the content successfully, a green check mark will appear next to the item name indicating that the content is available for configuration in the *Properties* tab.



Figure 44 - Content loaded



1. If the content is an image, the user must specify the converter type as "Image” and specify the desired output format for conversion.

The user can also specify where to store the converted image data in RAM\_G through the memory options, or in flash storage through the storage options. Please note that the converted data is stored in the same directory as the original image.

NOTE: only ASTC format can be loaded from flash storage directly.

1. If the content is raw data, simply select the "Raw" option in the Converter drop-down menu since already converted raw data does not need further processing. Upon loading the data successfully, users can specify the offset of raw data in the "Start" edit box as well as the length of data to be imported in the "Length" edit box.

Figure - Content properties



Figure 46 - Raw converter

1. Upon image conversion, users can drag the image from the content manager and drop it into the viewport.

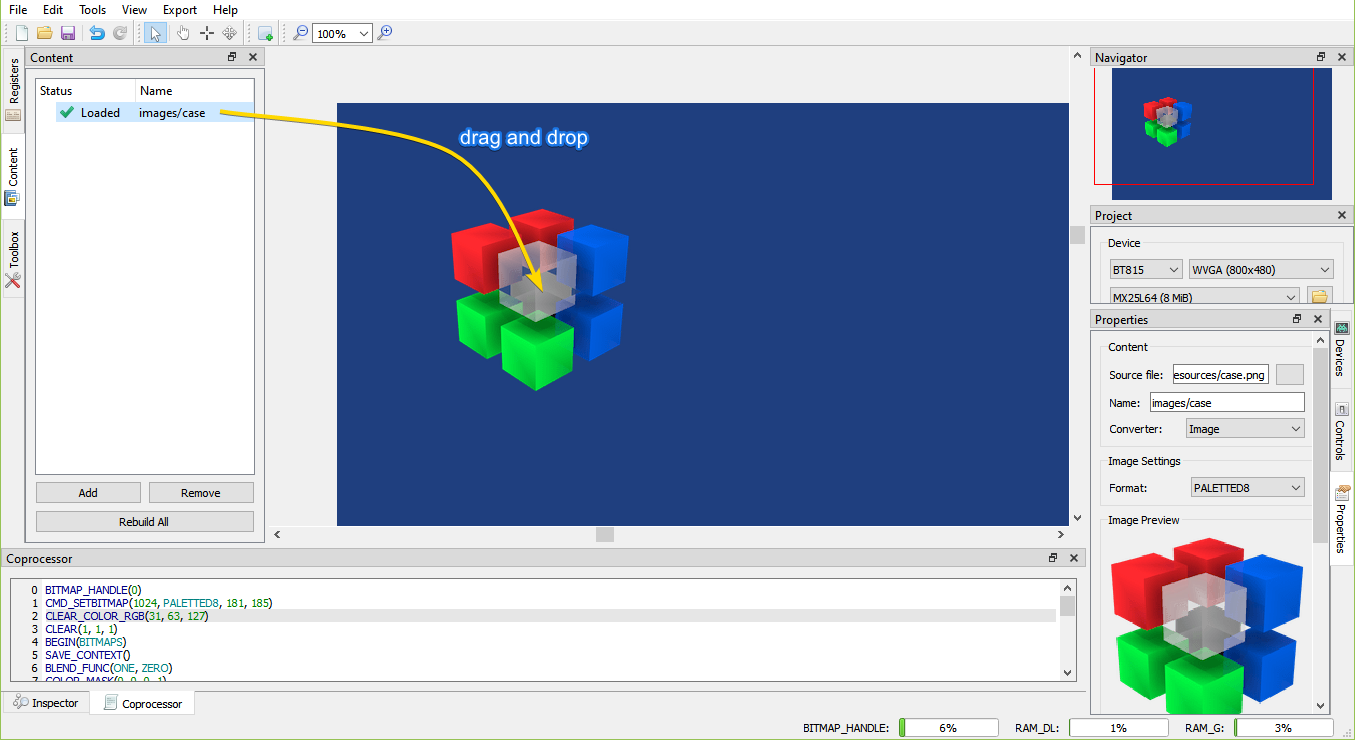


Figure 47 - Drag content into Viewport

After placing the image, the display list will be generated in the editor automatically, appended after the currently focused command.

#### Remove content

Users may remove the selected bitmap or raw data in the content manager and clear the content manager.

To remove added content:

* Select the content to be removed and click **[Remove]**.



Figure 48 - Remove content

* The selected content is then removed from the list.

#### Rebuild the content

If the source file of the content has been marked out of date, users need to rebuild it.

## Devices, Controls, Properties, and Output

This section illustrates the *Controls* and *Properties* tab in the EVE Screen Editor.

### Device Manager

The Device Manager enables the user to connect the EVE board with the PC and observe the design directly on the hardware.



Figure 49 - Before connecting



Figure 50 - Connected device



The device type can be changed by clicking this icon and then selecting the correct display device type. Built-in devices are displayed in bold font and custom devices are displayed in regular font.



Figure 51 - Device type

**Note:** The Horizontal and Vertical input fields in the Registers dock change the View Port dimensions only. The display configurations when syncing with the device are determined by the selected display device type.

### Controls

In the *controls* tab, users may execute the code step by step in the granularity of the display list command or coprocessor command.

See the "Steps" grouped widgets below.

As a result, the step-by-step construction of the screen can be viewed by increasing or decreasing the value of the display list or coprocessor input box.

Only one option can be selected at any given point of time and the respective tab must be focused. Refer to the topic **Step by Step** for more details.

Figure 52 - Controls tab

Users may also trace, which commands are involved to render the pixel at the specified coordinator.

See the "Trace" grouped widgets below. Refer to the topic **Trace the pixel** for more details.

### Properties

The properties tab provides the information as well as the available editable parameters of the selected commands and components. Different commands have different properties. These parameters can be edited either in the properties tab or in the code editor.



Figure 53 - Properties tab

This tab also provides information about the content item in the Content window.

### Output

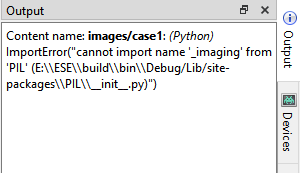
Warning and error messages will be displayed in the Output tab.

Figure - Output tab

## View Port

This is the significant area in the center of the screen. When the user selects any components or commands in the Toolbox, those components can be visually seen in the viewport. The viewport has the same resolution as specified in REG\_HSIZE and REG\_VSIZE.

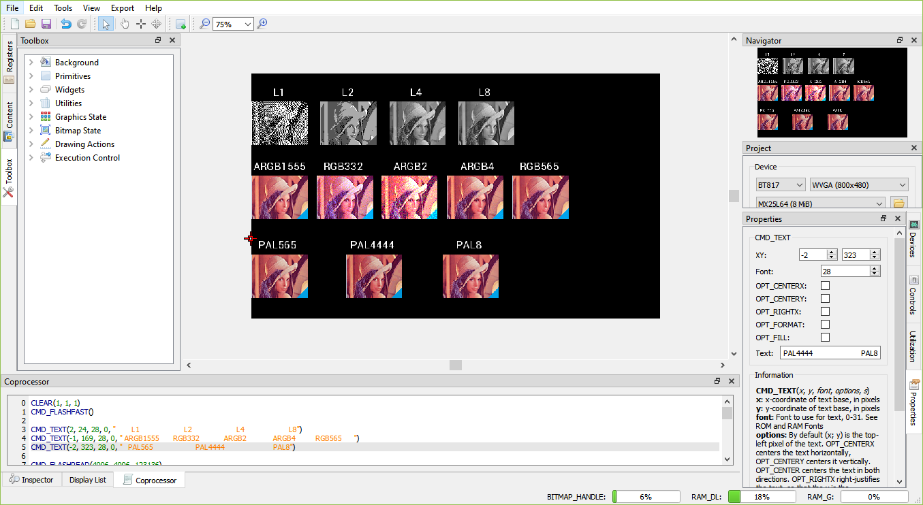


Figure 55 – Viewport

## Navigator

The viewport navigator provides a convenient way to move the viewport, especially for large resolutions.

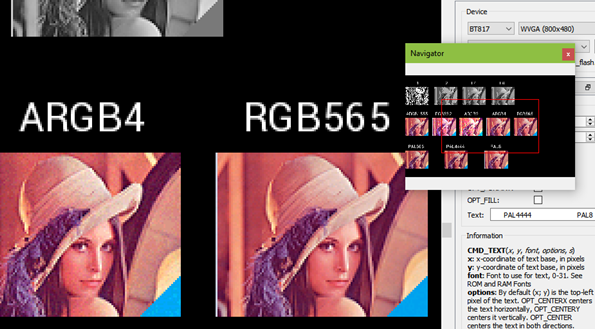


Figure 56 – Navigator

## Project settings

Within this tab, users can select chip type and corresponding screen resolution for the current project.



Figure 57 - Project settings

From BT81X, flash images are supported. To load a flash image, click the Browse button and select a flash map file on a local PC.



Figure 58 - Flash supported

Flash image (.bin) and flash map (.map) are generated by another tool called *EVE Asset Builder*. The latest version is available at this link - <https://brtchip.com/ic-module/toolchains/#EVEAssetBuilder>.

Follow these steps to generate a flash image:

1. Open the EAB tool, switch to the tab **Flash Utilities**
2. Add necessary asset files
3. Set output folder and output name for the flash image
4. Press the button **Generate**
5. Generated files are saved in the output folder



Figure 59 - Flash image generation by EAB



Figure 60 - Generated files in the output folder

A “flash.bin” file is a binary file that can be loaded into an emulator as well as a flash chip. Both “flash.map” and “flash.edf” are human-readable text files. Each line shows the asset name, the beginning address, and the length.



Figure 61 - Content of a .map file

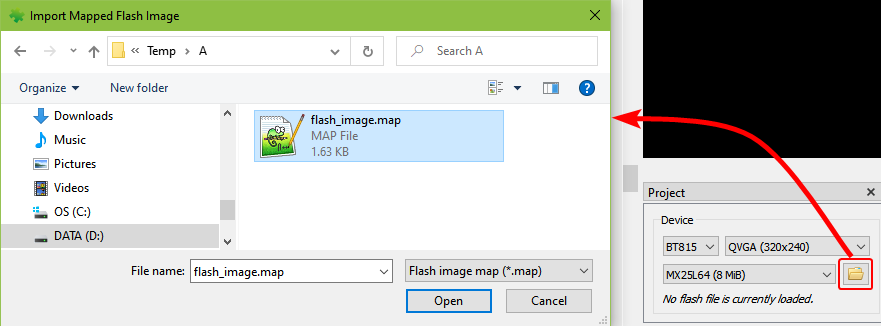


Figure 62 - Load flash file

Users can select flash memory from 8MB to 256 MB.



Figure 63 - Select flash size

Upon loading the flash file, its path is displayed.



Figure 64 – Display flash path

All the assets in a flash file are loaded and shown in the Content window.



Figure 65 - Flash assets are shown in the Content window

## Keyboard Shortcuts

The following keyboard shortcuts can be used in the screen editor:

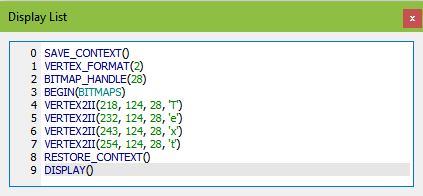
|  |  |
| --- | --- |
| Item | Shortcut |
| New | *Ctrl + N* |
| Save | *Ctrl + S* |
| Undo | *Ctrl + U* |
| Redo | *Ctrl + Y* |
| Cut | *Ctrl + X* |
| Copy | *Ctrl + C* |
| Paste | *Ctrl + V* |
| Zoom In/Out of Viewport | *Ctrl + Mouse wheel* |
| Close Project | *Ctrl + F4* |
| Reset Emulator | *Ctrl + R* |
| Capture Display List | *Ctrl + D* |
| Open Recent Project 1->5 | *Alt + 1, Alt + 2, …, Alt + 5* |
| Toolbar Cursor | *Alt + C* |
| Toolbar Touch | *Alt + T* |
| Toolbar Trace | *Alt + R* |
| Toolbar Edit | *Alt + E* |

# Quick Start Tutorials

This section explains how to use the EVE screen editor. They are intentionally kept brief so that the user can start using the editor as quickly as possible. The objective is not to teach the user every single detail, but to help the user to get familiarized with the basic principles and the way the editor works.

## Capture Display List

To capture the display list, select menu **Tools -> Capture Display List**, or use the shortcut *Ctrl+D*. For example, users can type "CMD\_TEXT" in the co-processor editor, then press *Ctrl+D*. The display list commands will be displayed.



## Change the color

Subsequent drawing color can be changed by the drag and drop method of the Color RGB command, under the *Graphics State* group in the **Toolbox** to the viewport and then by choosing the desired color in the Command Properties or by editing the command values in the command output.



Figure 66 - Change the color

The Properties tab of the Color RGB command can change the color visually by clicking on the color bar and selecting a color.

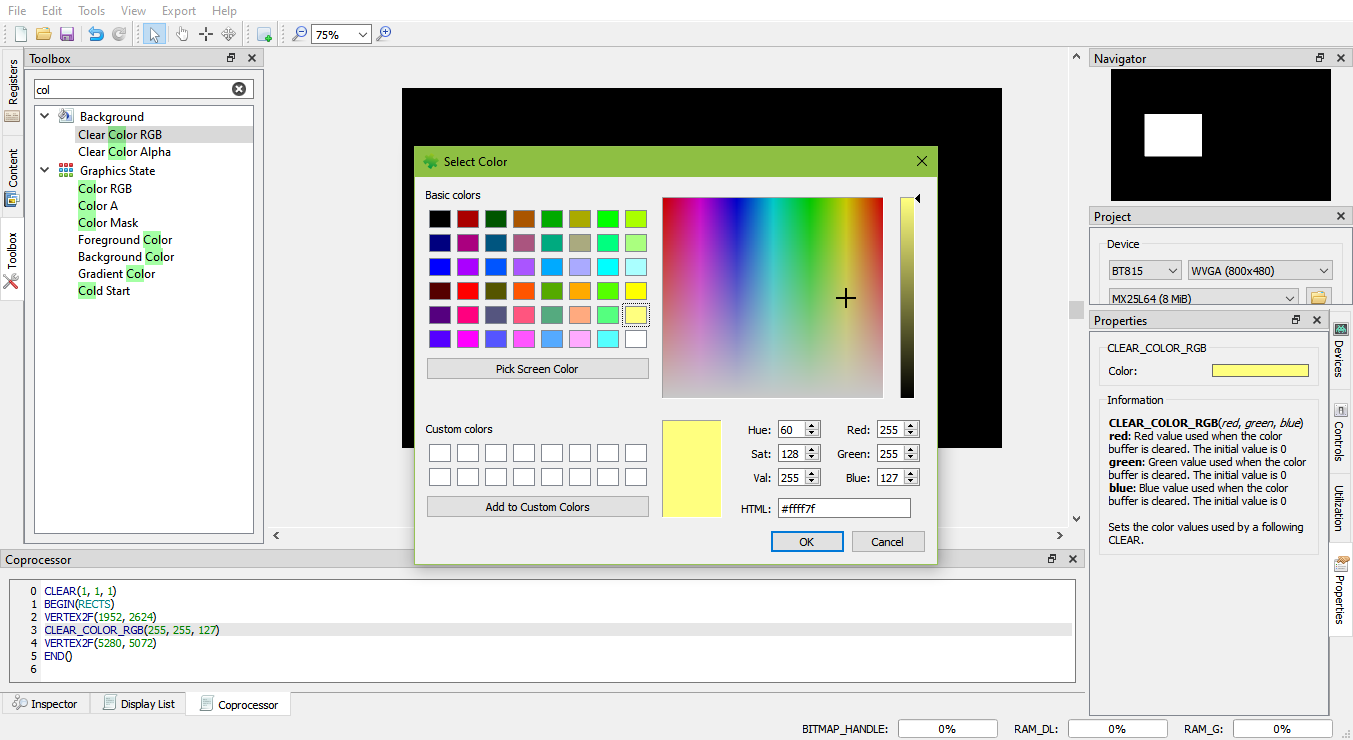


Figure 67 - Select color

In the EVE code syntax, the following commands have the color channels as their parameters (in the order of red, green, and blue):

• COLOR\_RGB

• CLEAR\_COLOR\_RGB

• CMD\_GRADIENT

• CMD\_BGCOLOR

• CMD\_FGCOLOR

• CMD\_GRADCOLOR

## Import the content

Importing the content adds the bitmap or raw data to the content tab. The data added will be listed in the content tab and can be used in the construction of display screens by dragging and dropping the data into the viewport. The raw and bitmap data can be added to the list as explained in Add Content. The added data can be removed by selecting an entry and clicking **[Remove]** in the Content tab.



Figure 68 - Import content

If the content added is an image, select the "Image" mode of Converter in the properties tab:



Figure 69 - Select converter image

Upon adding the image data successfully, the image can be dropped in the viewport by dragging the content name in the Content Manager to the viewport. The display commands are automatically generated.

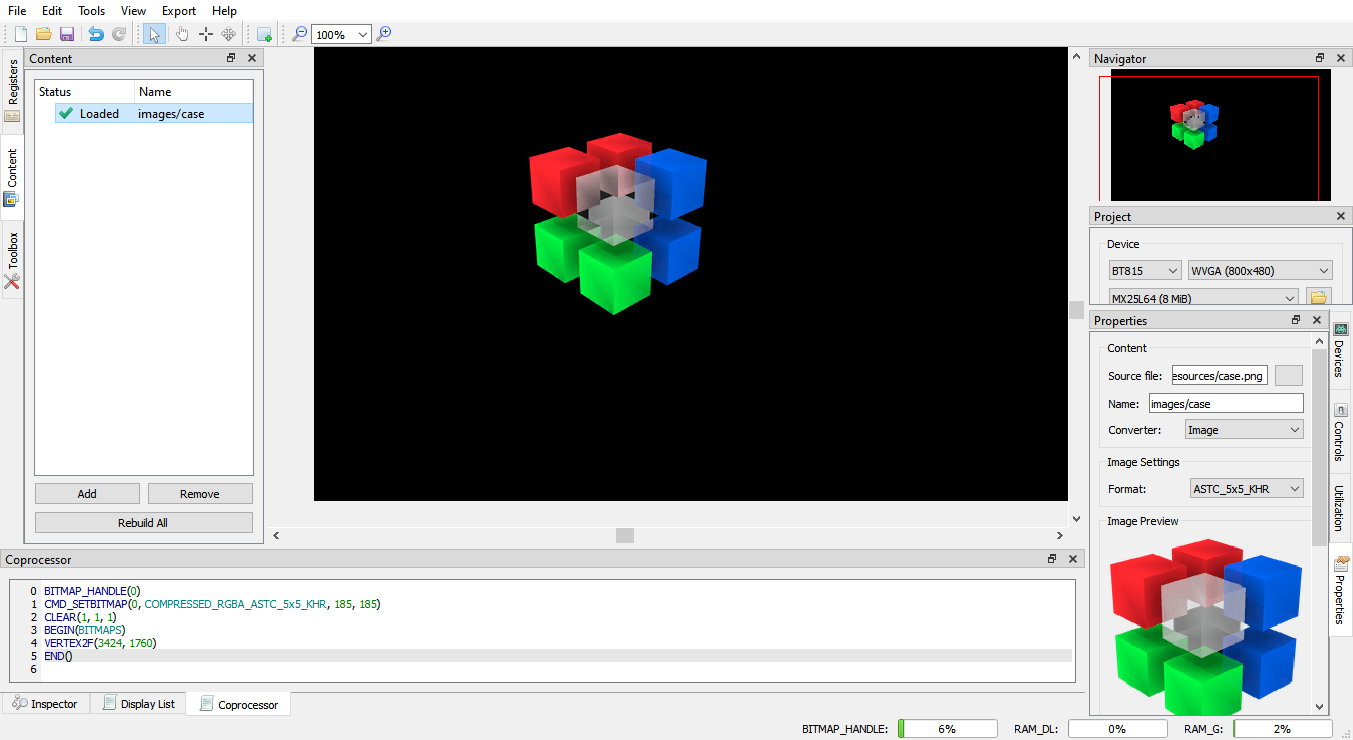


Figure 70 – Drag & drop image

The following information is for users who wish to program EVE directly, it is not required to use this utility.

For each valid resource in the Resource Manager, the utility converts it to the below file formats (except for \*.raw resources):

|  |  |
| --- | --- |
| \*.raw | The binary format of the converted file can be downloaded into RAM\_G directly. |
| \*.rawh | The header file of the converted file is in the text representation. Programmers can include this file into their program and build it into the final binary. |
| \*.bin | The compressed binary format of converted file in ZLIB algorithm. Programmers need to download it into RAM\_G and use CMD\_INFLATE to inflate them before using it. |
| \*.binh | The header file of compressed binary format, which is in the text representation of \*.bin. Programmers can include this file into their program and build it into the final binary. |

If the palette image format was chosen, files with the “.lut” text in the file name are generated and the appropriate file should be downloaded into RAM\_PAL for FT80X or idle area in RAM\_G for FT81X and later.

The generated files are in the directory mentioned in the "Information" section of the resource "Properties" tab.

## Import the flash

Importing the flash adds resources such as movies, images, font, etc. The added data is formatted as raw and loaded into flash memory.

Their flash address can be used in the display list and co-processor command.

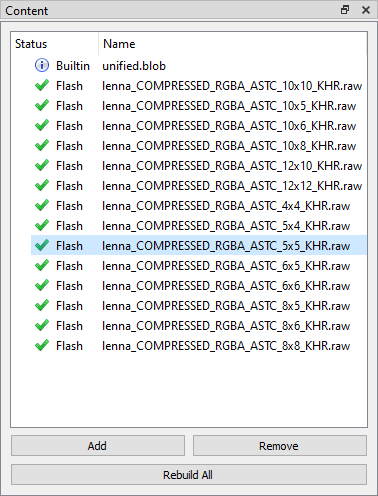
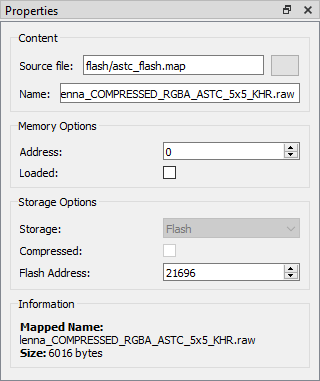


Figure - Import flash

Figure - Flash address

## Open the project

To open a saved project, simply click ***Open*** on the toolbar or select **File > Open** from the menu bar and browse for the saved project.

In 2.X, ESE is still able to open 1.X project file with the ".ft800proj" extension name.

In 3.X or above, ESE is still able to open 1.X and 2.X project files with ".ft800proj" and ".ft8xxproj" extension names.

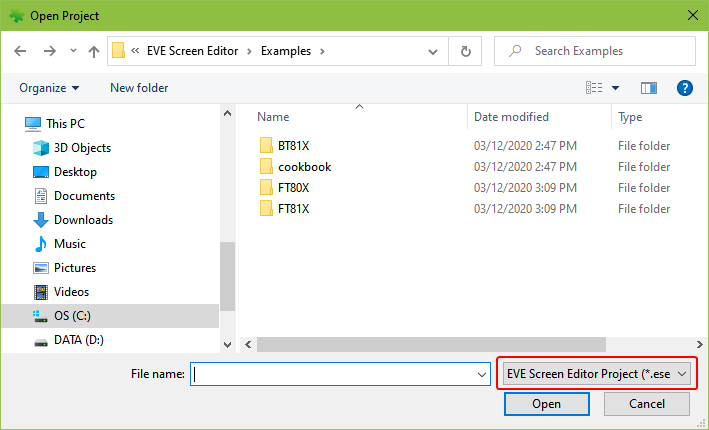


Figure 73 - Open project

From version 4.4, users can drag and drop a project folder or a project file (\*.ese) into ESE to reopen that project.

## Save your design

The current project can be saved by clicking *Save* on the toolbar or by selecting **File > Save** from the menu bar or pressing the *Ctrl + S* keyboard shortcut. Users can also save the current project under a different name and/or in a different directory by selecting **File > Save As**.

The EVE Screen Editor can only open the saved project.

Please note the saved projects have an extension of ***.ese***.

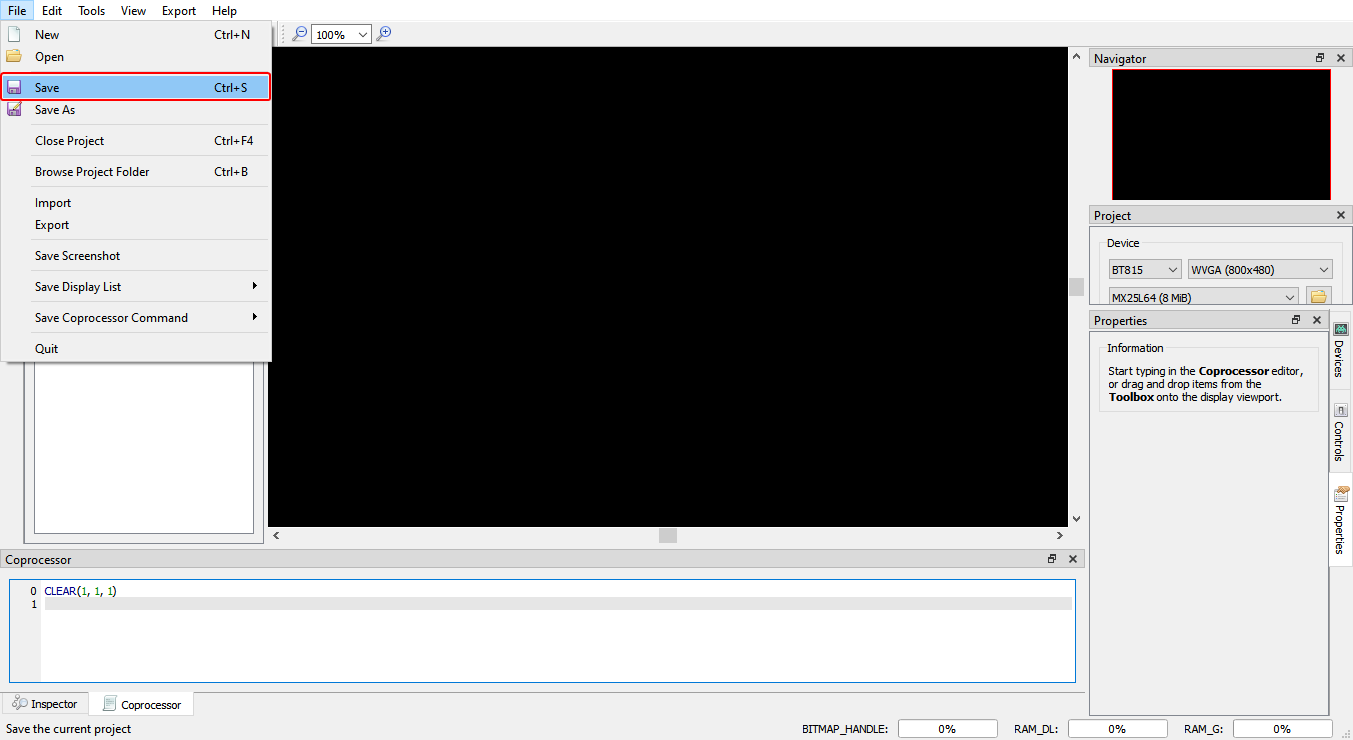


Figure 74 - Save the project

## Export the project

Upon creating the screen design, users can export the design in the form of a Gameduino 2 project, EVE Arduino project, EVE HAL2.0 project (C code based), and Raspberry Pi Pico project.

After the respective project is successfully exported, the Properties dock continues to show the project output information till the subsequent user interaction with the application.

Since the export feature requires writing files to the disk, a user needs to ensure that the EVE Screen Editor has the proper privileges. This may require running the tool as an Administrator.

Note that the Content Manager does not apply a strict naming convention on the loaded items, but while exporting, they need to be distinct and follow the C programming language variable naming convention.

To export the project, follow the steps below:

1. Click the **Export** menu in the menu bar.
2. Select the option to which the project is to be exported.

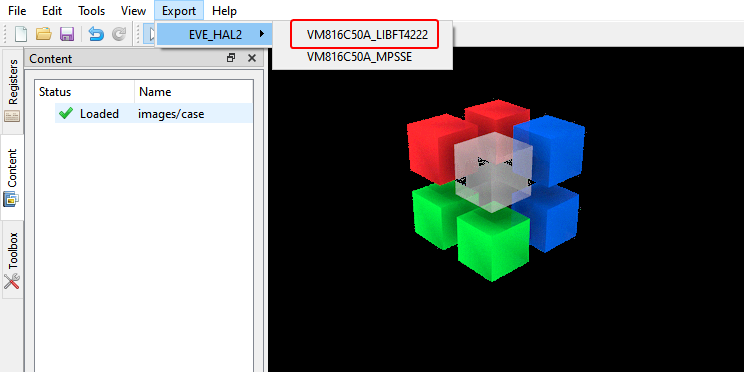


Figure 75 - Export project

For the "EVE HAL2.0 project", a file browser opens after the generation, and the ReadMe.txt in the project folder details the project directory files.

For "Gameduino2" and "Arduino" projects, the project files will get generated and opened by Arduino IDE, if the Arduino IDE is installed, then the Arduino project file extension ".ino" is associated with the Arduino IDE. Please note GameDuino2 EVE library and Arduino library are required to compile and build the project.

Users are required to read the datasheet of GameDuino2 for hardware connection information.



Figure 76 - Arduino IDE

For Raspberry Pi Pico projects, the project files will get generated in a separate folder which contains two sub folders “assets,” “lib” and two files “code.py”, and “readme.md”. Users need to read the “readme.md” file for the explanation of each item as well as the circuitPython firmware version, hardware connection information, and some useful links.

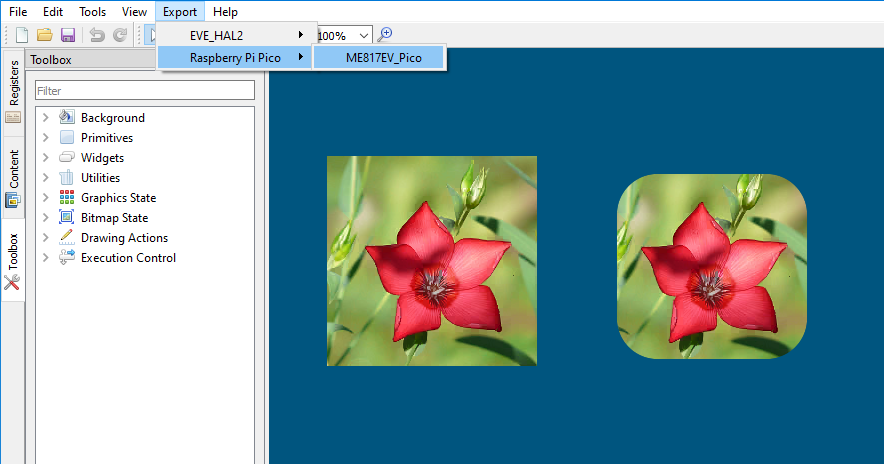


Figure 77 – Export Raspberry Pi Pico project

## Custom Fonts

ESE supports widely used custom font types such as *TrueType fonts (TTF)* and *OpenType fonts (OTF)*. The widgets in the Toolbox can only support non-kerned fonts. Kerned fonts can still be displayed if they are drawn individually as bitmaps. The Examples folder contains multiple custom fonts and using non-kerned fonts is as simple as loading bitmaps.

To use custom fonts:

1. Load the custom font in the Content manager. Successfully loaded fonts have the "Loaded" status next to the font name.
2. Set the font format and size attributes.
3. Drag and drop the font to the Viewport.
4. Click the font object in the Viewport to edit the display text.
5. "CMD\_SETFONT" and "CMD\_SETFONT2" are generated accordingly for FT80X and FT81X/BT81X devices to assign the new custom fonts with one unused bitmap handle. By default, the first unused bitmap handle is zero.



Figure 78 - Custom font



Figure 79 - Property "Text" of custom font

## Constrain either horizontal or vertical positioning when dragging an object

### Constrain vertical positioning

1. Prepare two objects which need to be aligned in the same x-coordinate
2. Press SHIFT and left-click on the aligned object
3. Slide it up and down following the dashed red line



Figure 80 - Press SHIFT to constrain vertical



Figure 81 - Slide up/down to set y-coordinate

### Constrain horizontal positioning

1. Prepare two objects which need to be aligned in the same y-coordinate
2. Press ALT and left-click on the aligned object
3. Slide it left and right following the dashed red line



Figure 82 - Press ALT to constrain horizontal



Figure 83 - Slide left/right to set x-coordinate

## Import/Export Memory Dump File

Users can save what is present in the memory of an Eve chip to a file, then load it back. These actions can be done via menu item *File -> Export* and *File -> Import*. The memory dump file which has an extension “.eve\_dump” is a binary file. Its format is differentiated from the Eve chip series, as described in the following sections.

#### Header

uint32\_t dump\_version; // 100 -> FT80X; 110 -> FT81X/BT81X  
 uint32\_t display\_width;  
 uint32\_t display\_height;  
 uint32\_t macro\_command\_1;  
 uint32\_t macro\_command\_2;  
 uint32\_t reserved;

#### Content

|  |  |
| --- | --- |
| Eve Chip | Memory Dump File Content |
| FT80X | 256 KBytes of RAM\_G  1 KBytes of RAM\_PALETTE  8 KBytes of RAM\_DL |
| FT81X | 1024 KBytes of RAM\_G  8 KBytes of RAM\_DL |
| BT81X | 1024 KBytes of RAM\_G  8 KBytes of RAM\_DL |

Table 2 - Memory Dump File Content

# Command Usage Examples

This section explains the usage of some of the new and advanced commands. As not all the commands are supported by all the devices, the description of commands indicates which devices they can run on.

## CMD\_PLAYVIDEO

CMD\_PLAYVIDEO plays back an MJPEG-encoded AVI video. This command can only be used in FT81X/BT81X devices.

***Prototype:***

**CMD\_PLAYVIDEO(Option, Stream)**

***Parameter Description:***

*Option* (one or more of the following) -

OPT\_MONO - video playback in greyscale, L8, mode.

OPT\_NOTEAR - Attempt to avoid horizontal "tearing" artifacts.

OPT\_FULLSCREEN - Zoom the video so that it fills as much of the screen as possible.

OPT\_MEDIAFIFO - Instead of sourcing the AVI video data from the command buffer, source it from the media FIFO. If this option is checked, CMD\_MEDIAFIFO must be specified before CMD\_PLAYVIDEO.

OPT\_SOUND - Video playback with sound.

*Stream* -Absolute or relative path to the MJPEG-encoded AVI video.

***Example:***

CMD\_PLAYVIDEO(OPT\_FULLSCREEN | OPT\_SOUND, "../chickens-4.avi")

CMD\_DLSTART()

CMD\_TEXT(154, 212, 31, 0, "Video playback has ended.")

***Note:***

CMD\_PLAYVIDEO is a blocking command which it initiates the video playback till the end of the input .avi file. All display objects before and after the CMD\_PLAYVIDEO are not shown while the video back is in progress. CMD\_DLSTART() should be specified right after CMD\_PLAYVIDEO to continue to display the subsequent display commands.

## CMD\_LOADIMAGE

CMD\_LOADIMAGE decompresses the specified JPEG or PNG data into a bitmap, in RAM\_G. Note: FT80X does not support PNG image sources.

***Prototype:***

**CMD\_LOADIMAGE(Address, Options, Stream)**

***Parameters Description:***

*Address* - The starting location in RAM\_G where the command will put the decoded data.

Options (one or more of the following):

*OPT\_MONO* - Decode the image to mono, L8, format.

*OPT\_NODL* - The command will not insert the default display commands in the display list buffer. The command will simply decode the file to the specified location and format.

*OPT\_MEDIAFIFO* - Use a mediafifo in RAM\_G as a buffer for decoding, instead of the coprocessor buffer. Otherwise, Mediafifo is not required to decode a bitmap file.

*CMD\_MEDIAFIFO* must be specified before this command.

*Stream* - The absolute or relative path from the project of the image to be decoded.

***Example:***

CLEAR(1, 1, 1)

/\*decode Eiffel Tower jpeg image. Put data to the 0th offset in RAM\_G and use default options\*/

CMD\_LOADIMAGE(0, 0, "../EiffelTower\_800\_480.jpg")

BEGIN(BITMAPS)

VERTEX2II(0, 0, 0, 0)

END()

/\*decode lenna256 png image. Put data after the Eiffel Tower decoded data and use mediafifo buffer for decoding\*/

CMD\_LOADIMAGE(768000, 0, "../lenna256.png")

BEGIN(BITMAPS)

VERTEX2II(413, 170, 0, 0)

END()

***Additional Information:***

Currently, CMD\_LOADIMAGE is a standalone command where the location of the decoded data is manually specified, but the application does not know the offset and amount of space that the decoded data will occupy. Users can "reserve" the RAM\_G space, so other assets in the content manager will not overwrite the data by loading the intended image in the content manager first with the final decoding format and RAM\_G offset.

If BITMAP\_HANDLE is used for other assets before the CMD\_LOADIMAGE command, then it might overwrite the bitmap handle properties when the OPT\_NODL is not selected, because the BITMAP\_HANDLE value is part of the context and CMD\_LOADIMAGE does not insert a BITMAP\_HANDLE command. Manually adding a BITMAP\_HANDLE with an unused handle before CMD\_LOADIMAGE might be needed to prevent re-association of the last specified bitmap handle.

For JPEG images, only the regular baseline JPEGs are supported. The default format is RGB565, or L8 if the OPT\_MONO option is selected.

For PNG images, only bit-depth 8 is supported; bit-depths 1, 2, 4, and 16 are not supported. The PNG standard defines several image colors format. Each format is loaded as a bitmap as follows:

Grayscale loads as L8,

Truecolor loads as RGB565,

Indexed loads as PALETTED4444, if the image contains transparency, or PALETTED565 otherwise,

Grayscale with alpha is not supported,

Truecolor with alpha loads as ARGB4

## CMD\_SETBITMAP

This command generates the corresponding display list commands (BITMAP\_SOURCE, BITMAP\_LAYOUT, BITMAP\_SIZE) for the given bitmap information, sparing the effort of writing the display list manually. This command is supported in FT81X or BT81X devices.

***Prototype:***

**CMD\_SETBITMAP(Address, Format, Width, Height)**

***Parameter Description:***

*Address* - The address in RAM\_G where the bitmap data starts.

*Format* - One of the device’s supported bitmap formats.

*Width* - The width of the bitmap.

*Height* - The height of the bitmap.

***Example:***

BITMAP\_HANDLE(0)

CMD\_SETBITMAP(0, RGB565, 800, 480)

BEGIN(BITMAPS)

VERTEX2II(0, 0, 0, 0)

END()

***Additional Information:***

If the bitmap is bigger than 512 pixels in either dimension, CMD\_SETBITMAP will also insert BITMAP\_LAYOUT\_H and/or BITMAP\_SIZE\_H command(s) with the appropriate parameter values.

The parameters filter/wrapx/wrapy in BITMAP\_SIZE is always set to NEAREST/BORDER/BORDER value in the generated display list commands.

## CMD\_SNAPSHOT

Capture the current screen and put the bitmap data in the specified RAM\_G location, the capturing bitmap format is always ARGB4. This command is supported on all devices.

***Prototype:***

**CMD\_SNAPSHOT(Address)**

***Parameter Description:***

*Address* - The address in RAM\_G where the device will put the captured bitmap data.

***Example:***

CLEAR(1, 1, 1)

CMD\_BUTTON(10, 14, 120, 36, 27, 0, "Button")

CMD\_KEYS(8, 65, 160, 36, 29, 0, "keys")

CMD\_TEXT(145, 22, 28, 0, "Text")

CMD\_SNAPSHOT(0)

BITMAP\_HANDLE(1)

BITMAP\_SOURCE(0)

BITMAP\_LAYOUT(ARGB4, 960, 200)

BITMAP\_SIZE(NEAREST, BORDER, BORDER, 480, 200)

BEGIN(BITMAPS)

VERTEX2II(197, 116, 1, 0)

END()

***Additional Information:***

Users can also use **[Capture Snapshot]** in the "Properties" tab of the **CMD\_SNAPSHOT** command to save the captured bitmap as an ARGB4 raw file, JPEG, or PNG image.

## CMD\_SKETCH

CMD\_SKETCH is one co-processor command which tracks the user's touch input and updates the memory content accordingly.

***Prototype:***

**CMD\_SKETCH(X, Y, W, H, Address, Format)**

***Parameter Description:***

*X, Y* - The coordinates of the top-left pixel of the sketching area

*W, H* - The width and height of the sketching area.

*Address* - The address in RAM\_G where the device will put the bitmap data.

*Format* - L8 or L1

***Example:***

//To run on Screen Editor:

//Click the hand button in the menu bar then "draw" on the display area.

//To run on hardware:

//1] Perform a screen calibration after setup

//2] Make sure the following generated code will only run once.

CLEAR(1,1,1)

CMD\_MEMZERO(0,130560)

BITMAP\_HANDLE(0)

BITMAP\_LAYOUT(L8,480,272)

BITMAP\_SIZE(NEAREST,BORDER,BORDER,480,272)

BITMAP\_SOURCE(0)

CMD\_SKETCH(0,0,480,272,0,L8)

BEGIN(BITMAPS)

VERTEX2F(0,0)

END()

***Additional Information:***

Note that the mouse shall be switched to touch mode by clicking the toolbar before sketching on the viewport.

From version 4.4, users can drag CMD\_SKETCH from Toolbox and drop it into the emulator viewport. ESE will then generate the appropriate bitmap setup commands.

## CMD\_SNAPSHOT2

Capture a specific screen region and put the bitmap data in the specified RAM\_G location, the capturing bitmap format can be RGB565, ARGB4, or ARGB8\_SNAPSHOT. This command is supported in FT81X or BT81X devices.

***Prototype:***

**CMD\_SNAPSHOT2(Format, Address, X, Y, Width, Height)**

***Parameter Description:***

*Format* - Captured data format, either RGB565, ARGB4, or ARGB8\_SNAPSHOT.

*Address* - The address in RAM\_G where the device will put the captured bitmap data.

*X* - The x-coordinate of the top-left vertex of the intended capturing region.

*Y* - The y-coordinate of the top-left vertex of the intended capturing region.

*Width* - The width of the intended capturing region.

*Height* - The height of the intended capturing region.

***Example:***

CLEAR(1, 1, 1)

CMD\_BUTTON(10, 14, 120, 36, 27, 0, "Button")

CMD\_KEYS(8, 65, 160, 36, 29, 0, "keys")

CMD\_TEXT(145, 22, 28, 0, "Text")

CMD\_SNAPSHOT2(RGB565,0,0,0,300,300)

BITMAP\_HANDLE(1)

BITMAP\_SOURCE(0)

BITMAP\_LAYOUT(RGB565, 600, 300)

BITMAP\_SIZE(NEAREST, BORDER, BORDER, 300, 300)

BEGIN(BITMAPS)

VERTEX2II(300, 300, 1, 0)

END()

***Additional Information:***

Users can also use the **[Capture Snapshot]** in the "Properties" tab of the **CMD\_SNAPSHOT2** command to output the captured bitmap to a specified format raw file or as a JPEG or PNG image. ESE will show a warning message if the memory size required by CMD\_SNAPSHOT2 is too large.

## VERTEX\_TRANSLATE\_X/Y

If the user wants to shift the position of multiple objects but does not want to manually change the position of the objects, then VERTEX\_TRANSLATE\_X and/or VERTEX\_TRANSLATE\_Y commands can be used to translate all subsequent display commands with the specified amount of offset. These commands can only be used in FT81X or BT81X devices.

***Prototype:***

**VERTEX\_TRANSLATE\_X(Value) //translation in the x-axis**

**VERTEX\_TRANSLATE\_Y(Value) //translation in the y-axis**

***Parameter Description:***

*Value* - The amount of offset added to the respective coordinates, in 1/16 pixel. Negative values are permitted, and the initial value is 0.

***Example:***

CLEAR(1, 1, 1)

VERTEX\_TRANSLATE\_X(320) //translate all subsequent display objects by 20 pixels in the x-axis

VERTEX\_TRANSLATE\_Y(800) //translate all subsequent display objects by 50 pixels in the y-axis

CMD\_BUTTON(35, 45, 120, 36, 27, 0, "Button")

CMD\_KEYS(34, 141, 160, 36, 29, 0, "keys")

CMD\_TEXT(316, 56, 28, 0, "Text")

CMD\_GAUGE(305, 135, 36, 0, 4, 8, 40, 100)

CMD\_TOGGLE(205, 53, 40, 27, 0, 0, "on\xFFoff")

CMD\_DIAL(201, 226, 36, 0, 6144)

VERTEX\_TRANSLATE\_X(0) //change back to default

VERTEX\_TRANSLATE\_Y(0) //change back to default

***Additional Information:***

Both VERTEX\_TRANSLATE\_X and VERTEX\_TRANSLATE\_Y are part of the graphics context which means the value specified to the commands will affect all subsequent drawing objects till the value has changed or the *Tools->Reset Emulator* option is selected.

## CMD\_MEDIAFIFO

When a project is in FT81X/BT81X mode, both CMD\_PLAYVIDEO and CMD\_LOADIMAGE commands have the option to utilize a mediafifo buffer in RAM\_G to speed up the data loading process. If option **OPT\_MEDIAFIFO** is not selected, all data is expected to be loaded into the co-processor buffer which is limited to a maximum of 4 kilobytes per transfer. The performance increase can be noticeably faster when running the exported project on the hardware.

***Prototype:***

CMD\_MEDIAFIFO(ptr, size)

***Parameter Description:***

*ptr* - The starting address of the memory block which will be used as a media fifo.

*size* - The size of the memory block.

***Example:***

CLEAR(1, 1, 1)

CMD\_MEDIAFIFO(768000, 20000)

CMD\_LOADIMAGE(0, OPT\_MEDIAFIFO, "..//EiffelTower\_800\_480.png")

BEGIN(BITMAPS)

VERTEX2II(0, 0, 0, 0)

END()

## CMD\_SETBASE

CMD\_SETBASE sets the numeric base for CMD\_NUMBER. This command can only be used in FT81X or BT81X devices.

***Prototype:***

CMD\_SETBASE(Base)

***Parameter Description:***

*Base* - The numeric base, valid values are from 2 to 36. Common bases are:

2 - binary

8 - octal

10 - decimal

16 – hexadecimal

***Example:***

CLEAR(1, 1, 1)

CMD\_SETBASE(2) //set base to binary

CMD\_NUMBER(88, 193, 29, 0, 65536)

## CMD\_ROMFONT

CMD\_ROMFONT sets any device ROM fonts to one bitmap handle. FT81X offers a couple of bigger ROM fonts, and this command is required to utilize those fonts for built-in widgets. This command can only be used in FT81X or BT81X devices.

***Prototype:***

CMD\_ROMFONT(Font, RomSlot)

***Parameter Description:***

*Font* - The bitmap handle is to be associated with the specified ROM font, a valid value range is 0 to 31.

*RomSlot* - The ROM fonts to be associated.

***Example:***

CLEAR(1, 1, 1)

CMD\_ROMFONT(1, 31); //associate bitmap font handle 31 to 1

CMD\_TEXT( 0, 0, 1, 0, "31");

CMD\_ROMFONT(1, 32); //associate bitmap font handle 32 to 1

CMD\_TEXT( 0, 60, 1, 0, "32");

CMD\_ROMFONT(1, 33); //associate bitmap font handle 33 to 1

CMD\_TEXT(80,-14, 1, 0, "33");

CMD\_ROMFONT(1, 34); //associate bitmap font handle 34 to 1

CMD\_TEXT(60, 32, 1, 0, "34");

***Additional Information:***

Bitmap font handles 32 - 34 are only available on FT81X or BT81X devices. Bitmap handle parameter is limited to 0-31. Other than the ability to re-associate a bitmap font handle to a different handle, ROM fonts 32-34 must use CMD\_ROMFONT to associate itself to handle 0-31.

## PALETTE\_SOURCE

Palette lookup tables are in the RAM\_G. PALETTED\_SOURCE allows the user to specify the lookup table for the paletted asset. As a result, multiple look-up tables and index data files can be loaded.

PALETTED\_SOURCE can only be used for FT81X or BT81X devices.

***Prototype:***

PALETTE\_SOURCE(Addr)

***Parameter Description:***

*addr* - The starting address of the look-up table in RAM\_G.

***Example:***

// drawing of an arbitrary paletted bitmap

PALETTE\_SOURCE(0)

BITMAP\_SOURCE(1024)

BITMAP\_LAYOUT(PALETTED565, 80, 40)

BITMAP\_SIZE(NEAREST, BORDER, BORDER, 40, 40)

BEGIN(BITMAPS)

VERTEX2F(0,0)

END()

***Additional Information:***

The palette lookup table has a maximum size of 1024 bytes. The value specified to PALETTE\_SOURCE is part of the context.

## CMD\_SETFONT2

To use a custom font with the co-processor objects, create the font definition data in RAM\_G and issue CMD\_SETFONT2, as described in ROM and RAM Fonts. Note that CMD\_SETFONT2 sets up the font’s bitmap parameters by appending commands BITMAP\_SOURCE, BITMAP\_LAYOUT, and BITMAP\_SIZE to the current display list. Please note this command is only applicable to FT81X or BT81X devices.

***Prototype:***

CMD\_SETFONT2(Font, Ptr, Firstchar)

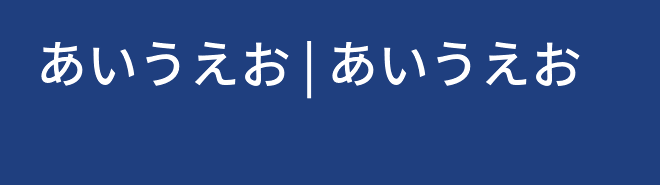
***Parameter Description:***

*font* - The bitmap handle from 0 to 31

*ptr* - 32-bit aligned memory address in RAM\_G of font metrics block

f*irstchar* - The ASCII value of the first character in the font. For an extended font block, this should be zero.

***Example:***



With a suitable font metrics block loaded in RAM\_G at address 0, appropriate glyph file is loaded in Flash, first character’s ASCII value 32, to use it for font 0:

CLEAR(1, 1, 1)

CMD\_FLASHFAST()

CMD\_SETFONT2(0, 0, 32)

CMD\_TEXT(50, 100, 0, 0, "あいうえお | \u3042\u3044\u3046\u3048\u304a")

## CMD\_TEXT

Draw text, can use string specifier.

***Prototype:***

CMD\_TEXT(X, Y, Font, Options, Text)

***Parameter Description:***

*x* - x-coordinate of text base, in pixels

*y* - y-coordinate of text base, in pixels

*font* - Font to use for text, 0-31.

*options -* By default (x, y) is the top-left pixel of the text and the value of options is zero.

OPT\_CENTERX centers the text horizontally, and OPT\_CENTERY centers it vertically.

OPT\_CENTER centers the text in both directions.

OPT\_RIGHTX right-justifies the text so that the x is the rightmost pixel.

OPT\_FORMAT processes the text as a format string, see String formatting.

OPT\_FILL breaks the text at spaces into multiple lines, with a maximum width set by

CMD\_FILLWIDTH.

*text* – Text string, UTF-8 encoding. If OPT\_FILL is not given, then the string may contain newline (\n) characters, indicating line breaks. \x, \u, and \U are supported in the case of using Unicode code point.

\xhh...: The byte whose numerical value is given by hh... interpreted as a hexadecimal number.  
[\uhhhh](file:///\\uhhhh): Unicode code point below 10000 hexadecimal.

\Uhhhhhhhh: Unicode code point where h is a hexadecimal digit.

***Example:***



CMD\_TEXT(12, 49, 31, OPT\_FORMAT, "BT%dX Programmer Guide", 81)

CMD\_TEXT(11, 110, 31, OPT\_FORMAT, "0x%X = %d", 2069, 2069)

# Working with EVE Screen Editor

This section provides information about the usability of the EVE Screen Editor.

## Connect with Hardware

Once the user has designed the screen on the PC, if the user has the board connected to the PC, the device manager can be enabled to observe the effect on actual hardware.



Figure 84 - Connect Device

Upon clicking **[Connect]**, the device is ready to be synchronized with the Screen Editor.

**Note 1:** USB Hub will cause a failure in connection with the FT4222 module. So, in case of using the FT4222 module, please do not use USB Hub for connection.



**Note 2:** Do not forget to click this icon to select the correct device type.



Figure 85 - Device connected

If the project loads content from a flash image, we need to write flash first by clicking **[Write Flash Content]**. **[Upload RAM\_G and RAM\_DL]** is used for synchronizing Display List and **[Upload RAM and Coprocessor]** is used for synchronizing Coprocessor command.

**Note 1:** Ensure that the connected device (1) has the same EVE chip as the selected emulator(2).

**Note 2:** In case of having content items stored in flash, it is better to “Write Flash Content” before any update.

Check the picture below:

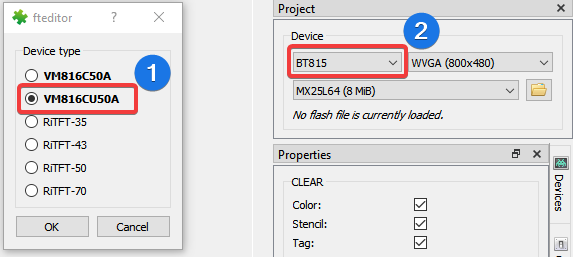


Figure 86 - Select the correct device type

**Managing device**

There are two kinds of devices: Built-in and Custom.

*Built-in Device:* Users can examine and clone these devices. They cannot be modified.

*Custom Device:* Users can Add/Edit/Clone/Remove a device.

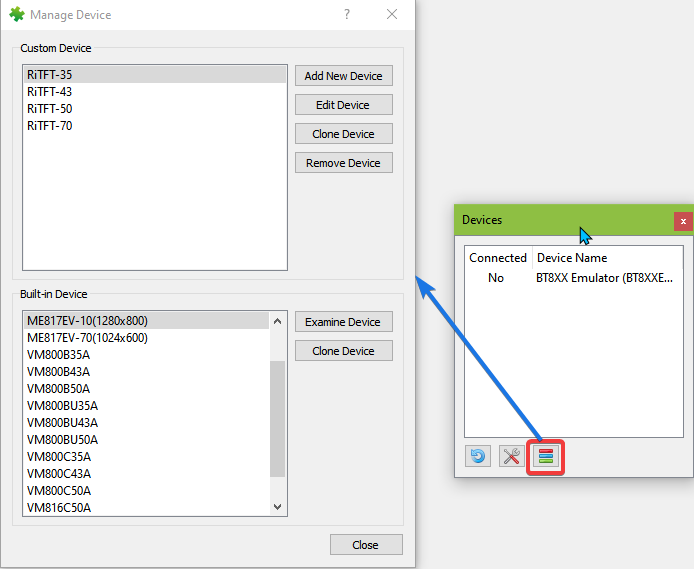


Figure 87 - Managing device

## Check your design

This section explains how to check a design.

### Step by Step

Users can select to execute the display list or co-processor command step by step to observe the effects of the commands up to that point. The increase or decrease in the value of the display list and co-processor editor box will execute the specified steps and highlight the specific display list or co-processor commands.



Figure 88 - Select Display List/Coprocessor

The 2nd display list command is highlighted in the yellow bar and there is nothing at the viewport because the VERTEX2II command is not executed yet.

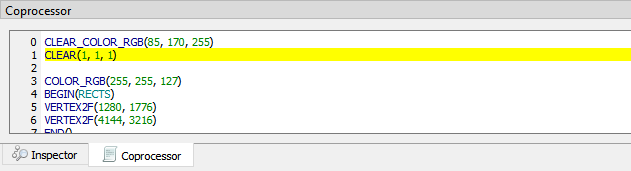


Figure 89 - Display List command is highlighted in yellow

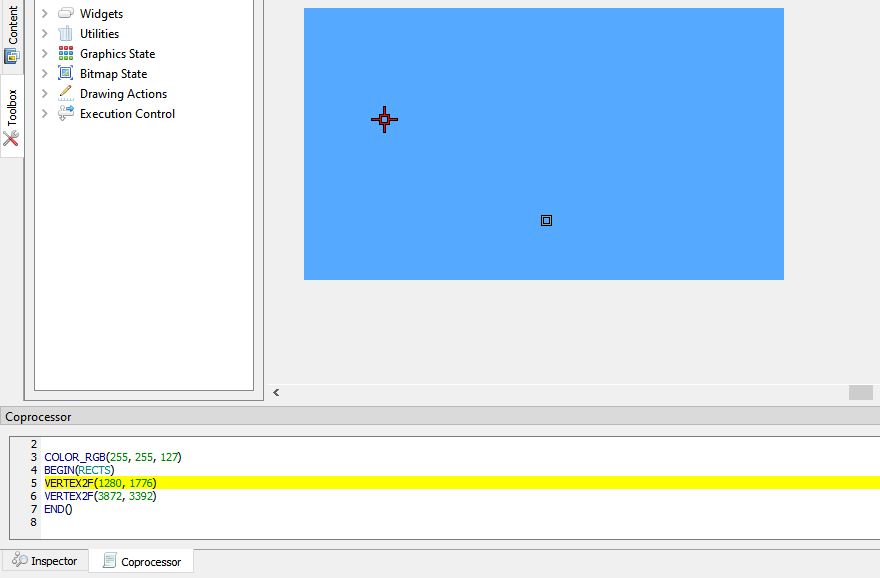


Figure 90 - Prepare to draw a rectangle

If the user increases the value in the editor box, the highlighted line will be moved, and the effect of the drawing is shown below:

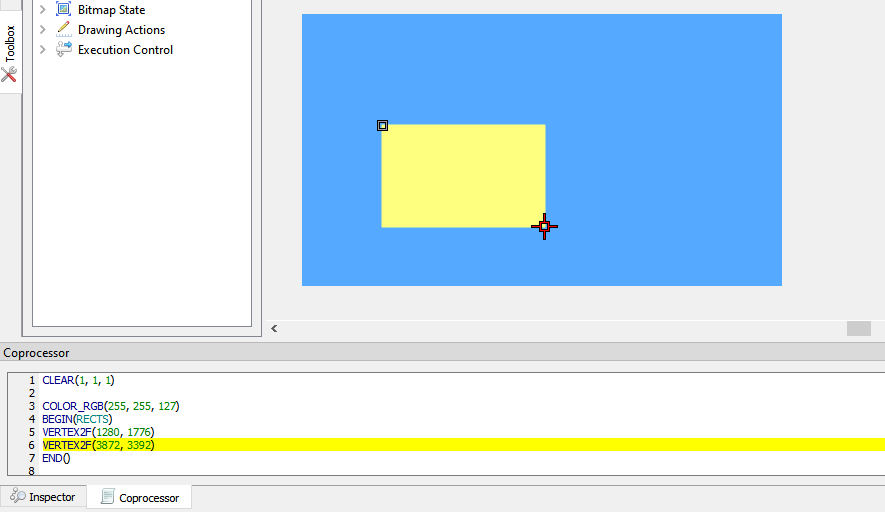


Figure 91 – Draw rectangle

### Trace the pixel

Users can check what commands or display lists are involved in the drawing of the pixel by selecting a coordinate in the viewport with the Trace mouse command. The movement of the Trace in the viewport is updated in the Trace section of the Control tab. If an object(s) occupies the tracing coordinate, then the respective command(s) in the commands editor is highlighted in **green**. If there is more than 1 object occupying the trace coordinate, the topmost object will get highlighted in a **brighter green** than those under it.

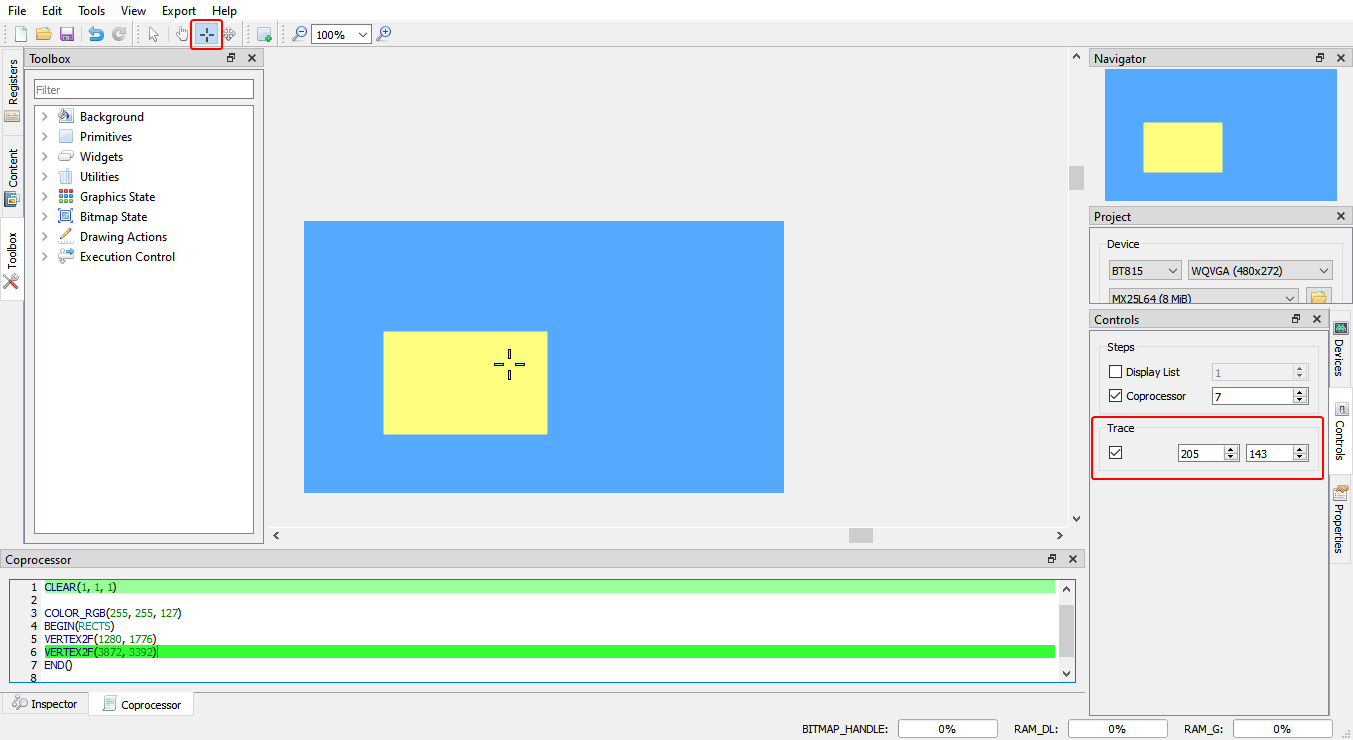


Figure 92 - Trace the pixel

### Drag and Drop

The commands in the toolbox can be dragged and dropped in the emulator viewport. The editor will be updated with the corresponding commands.

## Example Project

The examples are easily accessible from the "Examples" folder in the installation directory. They can be opened in the screen editor using **File > Open**.

The *Examples* folder contains three sub-folders *BT81X, FT81X, and* *FT80X* for specific example projects.

Opened "allwidget\_NoScreenSaver" project.

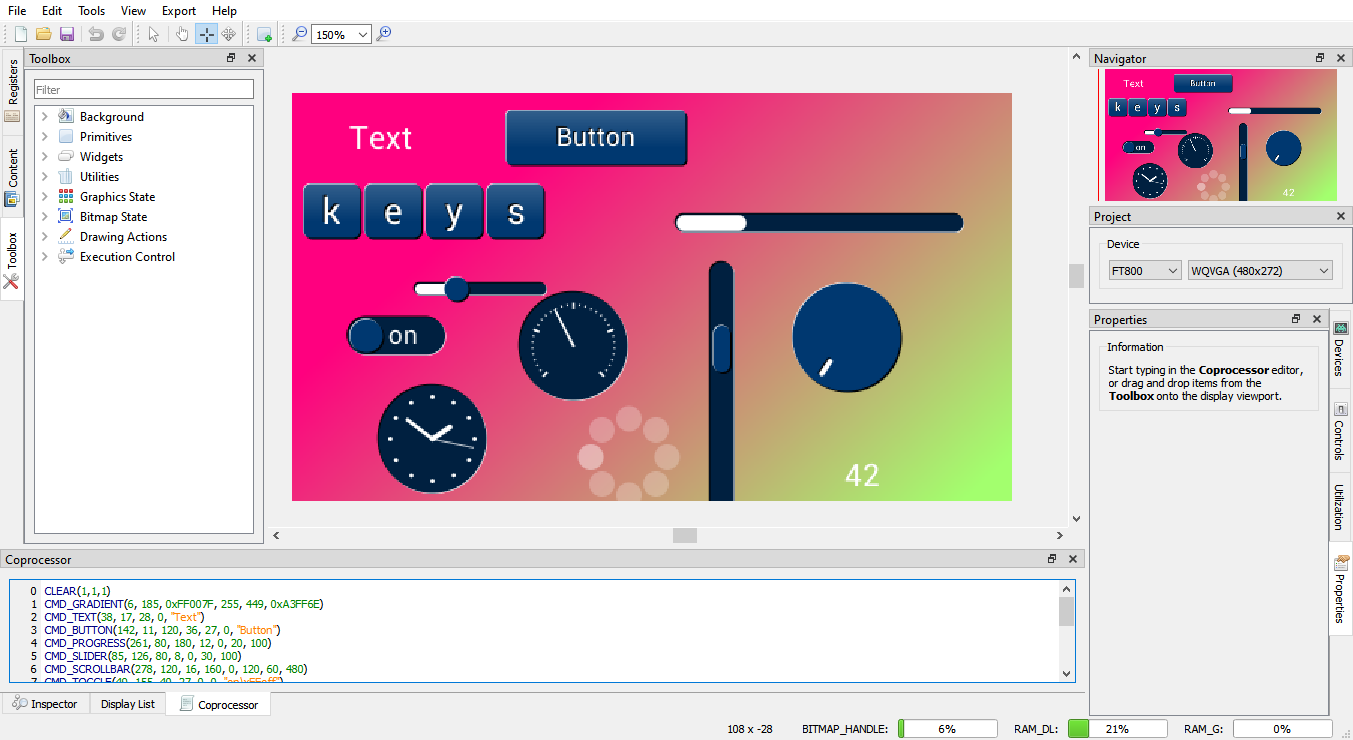


Figure 93 - Open example project

# Disclaimer

This section contains the license agreements for the EVE Screen. Any other third-party software or artifacts included in the software are listed under **Help > 3rd party**.

***Disclaimer***

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 because of use or failure of this product.

Your statutory rights are not affected. This product or any variant of it is not intended for use in any medical appliance, device, or system in which the failure of the product might be expected to result in personal injury. This document provides preliminary information that may be subject 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

178 Paya Lebar Road, #07-03

Singapore 409030

# Contact Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Head Quarters – Singapore** | | **Branch Office – Taipei, Taiwan** | |
|  | |  | |
| Bridgetek Pte Ltd  178 Paya Lebar Road, #07-03  Singapore 409030  Tel: +65 6547 4827  Fax: +65 6841 6071 | | Bridgetek Pte Ltd, Taiwan Branch  2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District  Taipei 114  Taiwan, R.O.C.  Tel: +886 (2) 8797 5691  Fax: +886 (2) 8751 9737 | |
|  | |  | |
| E-mail (Sales) | [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com) | E-mail (Sales) | [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com) |
| E-mail (Support) | [support.apac@brtchip.com](mailto:support.apac@brtchip.com) | E-mail (Support) | [support.apac@brtchip.com](mailto:support.apac@brtchip.com) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Branch Office - Glasgow, United Kingdom** | | **Branch Office – Vietnam** | |
|  | |  | |
| Bridgetek Pte. Ltd.  Unit 1, 2 Seaward Place, Centurion Business Park  Glasgow G41 1HH  United Kingdom  Tel: +44 (0) 141 429 2777  Fax: +44 (0) 141 429 2758 | | Bridgetek VietNam Company Limited  Lutaco Tower Building, 5th Floor, 173A Nguyen Van Troi, Ward 11, Phu Nhuan District, Ho Chi Minh City, Vietnam  Tel : 08 38453222 Fax: 08 38455222 | |
|  | |  | |
| E-mail (Sales) | [sales.emea@brtichip.com](mailto:sales.emea@brtichip.com) | E-mail (Sales) | [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com) |
| E-mail (Support) | [support.emea@brtchip.com](mailto:support.emea@brtchip.com) | E-mail (Support) | [support.apac@brtchip.com](mailto:support.apac@brtchip.com) |

**Web Site**

<http://brtchip.com/>

**Distributor and Sales Representatives**

Please visit the Sales Network page of the [Bridgetek Web site](http://brtchip.com/contact-us/) for the contact details of our distributor(s) and sales representative(s) in your country.

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Bridgetek Pte Limited (BRTChip) devices incorporated in their systems, meet all applicable safety, regulatory, and system-level performance requirements. All application-related information in this document (including application descriptions, suggested Bridgetek devices, and other materials) is provided for reference only. While Bridgetek has taken care to assure it is accurate, this information is subject to customer confirmation, and Bridgetek disclaims all liability for system designs and any application assistance provided by Bridgetek. Use of Bridgetek devices in life support and/or safety applications is entirely at the user’s risk, and the user agrees to defend, indemnify, and hold harmless Bridgetek from any damages, claims, suits, or expenses resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Bridgetek Pte Limited, 178 Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number: 201542387H.

# Appendix A - References

## Document References

<https://brtchip.com/ic-module/toolchains/>

[FT81x Series Programmers Guide](http://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT81X_Series_Programmer_Guide.pdf)

[FT81x Datasheet](http://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_FT81x.pdf)

[FT800 Series Programmers Guide](http://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT800_Series_Programmer_Guide.pdf)

[FT800 Embedded Video Engine Datasheet](http://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_FT800_Embedded_Video_Engine.pdf)

[BT81x Datasheet](https://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_BT81X.pdf)

[BT81X Series Programmer Guide](https://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/BRT_AN_033_BT81X_Series_Programming_Guide.pdf)

## Acronyms & Abbreviations

|  |  |
| --- | --- |
| Terms | Description |
| CPU | Central Processing Unit |
| DLL | Dynamic Link Library |
| ESE | EVE Screen Editor |
| EVE | Embedded Video Engine |
| GUI | Graphical User Interface |
| GPL | General Public License |
| HAL | Hardware Abstraction Layer |
| IDE | Integrated Development Environment |
| LGPL | Lesser General Public License |
| MCU | Micro Controller Unit |
| MPSSE | Multi-Protocol Synchronous Serial Engine |
| OS | Operating System |
| RAM | Random Access Memory |
| UI | User Interface |
| WYSIWYG | What You See Is What You Get |

# Appendix B – List of Figures & Tables

## List of Figures

[Figure 1 - User Interface Components 12](#_Toc107923811)

[Fig2 - File Menu 13](#_Toc107923812)

[Figure 3 - Edit Menu 14](#_Toc107923813)

[Figure 4 - Tool Menu 14](#_Toc107923814)

[Figure 5 - View Menu 14](#_Toc107923815)

[Figure 6 - Export Menu 15](#_Toc107923816)

[Figure 7 - Help Menu 15](#_Toc107923817)

[Figure 8 - Toolbar 15](#_Toc107923818)

[Figure 9 - Status Bar 16](#_Toc107923819)

[Figure 10 - Status Bar when a mouse hovers 16](#_Toc107923820)

[Figure 11 – Inspector 17](#_Toc107923821)

[Figure 12 - Coprocessor Command Editor 17](#_Toc107923822)

[Figure 13 - Display List Editor 18](#_Toc107923823)

[Figure 14 - Inspector 19](#_Toc107923824)

[Figure 15 - RAM\_DL 19](#_Toc107923825)

[Figure 16 - Select rows in RAM\_DL 19](#_Toc107923826)

[Figure 17 - Paste selected rows in RAM\_DL 20](#_Toc107923827)

[Figure 18 - RAM\_REG 20](#_Toc107923828)

[Figure 19 - Select rows in RAM\_REG 20](#_Toc107923829)

[Figure 20 - Paste selected rows in RAM\_REG 20](#_Toc107923830)

[Figure 21 - Toolbox Filter 21](#_Toc107923831)

[Figure 22 - Display List mode 21](#_Toc107923832)

[Figure 23 - Toolbox in Display List mode 22](#_Toc107923833)

[Figure 24 - Graphics State in Display List mode 22](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923834)

[Figure 25 – Primitive in Display List Mode 22](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923835)

[Figure 26 – Background in Display List mode 22](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923836)

[Figure 27 - Drawing Actions in Display List mode 23](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923837)

[Figure 28 - Execution Control in Display List mode 23](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923838)

[Figure 29 – Bitmap State in Display List mode 23](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923839)

[Figure 30 - Coprocessor mode 23](#_Toc107923840)

[Figure 31 - Toolbox in Coprocessor mode 24](#_Toc107923841)

[Figure 32 - Background in Co-processor mode 24](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923842)

[Figure 33 - Primitives in Co-processor mode 24](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923843)

[Figure 34 - Utilities in Co-processor mode 24](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923844)

[Figure 35 - Widgets Co-processor mode 24](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923845)

[Figure 36 – Execution Control in Coprocessor mode 25](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923846)

[Figure 37 – Drawing Actions in Coprocessor mode 25](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923847)

[Figure 38 – Bitmap State in Coprocessor mode 25](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923848)

[Figure 39 - Graphics State in Coprocessor mode 25](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923849)

[Figure 40 – Register 26](#_Toc107923850)

[Figure 41 - Customize resolution in the Register window 26](#_Toc107923851)

[Figure 42 - Content Manager 27](#_Toc107923852)

[Figure 43 - Load content dialog 27](#_Toc107923853)

[Figure 44 - Content loaded 28](#_Toc107923854)

[Figure 45 - Content properties 28](#_Toc107923855)

[Figure 46 - Raw converter 29](#_Toc107923856)

[Figure 47 - Drag content into Viewport 29](#_Toc107923857)

[Figure 48 - Remove content 30](#_Toc107923858)

[Figure 49 - Before connecting 31](#_Toc107923859)

[Figure 50 - Connected device 31](#_Toc107923860)

[Figure 51 - Device type 32](#_Toc107923861)

[Figure 52 - Controls tab 32](#_Toc107923862)

[Figure 53 - Properties tab 33](#_Toc107923863)

[Figure 54 - Output tab 33](#_Toc107923864)

[Figure 55 – Viewport 34](#_Toc107923865)

[Figure 56 – Navigator 34](#_Toc107923866)

[Figure 57 - Project settings 35](#_Toc107923867)

[Figure 58 - Flash supported 35](#_Toc107923868)

[Figure 59 - Flash image generation by EAB 36](#_Toc107923869)

[Figure 60 - Generated files in the output folder 36](#_Toc107923870)

[Figure 61 - Content of a .map file 36](#_Toc107923871)

[Figure 62 - Load flash file 37](#_Toc107923872)

[Figure 63 - Select flash size 37](#_Toc107923873)

[Figure 64 – Display flash path 37](#_Toc107923874)

[Figure 65 - Flash assets are shown in the Content window 37](#_Toc107923875)

[Figure 66 - Change the color 39](#_Toc107923876)

[Figure 67 - Select color 40](#_Toc107923877)

[Figure 68 - Import content 41](#_Toc107923878)

[Figure 69 - Select converter image 41](#_Toc107923879)

[Figure 70 – Drag & drop image 42](#_Toc107923880)

[Figure 71 - Import flash 43](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923881)

[Figure 72 - Flash address 43](file:///E:\ESE\git-ese\deployment\ese\Manual\BRT_AN_037_EVE%20Screen%20Editor%204.4%20User%20Guide.docx#_Toc107923882)

[Figure 73 - Open project 44](#_Toc107923883)

[Figure 74 - Save the project 45](#_Toc107923884)

[Figure 75 - Export project 46](#_Toc107923885)

[Figure 76 - Arduino IDE 47](#_Toc107923886)

[Figure 77 – Export Raspberry Pi Pico project 48](#_Toc107923887)

[Figure 78 - Custom font 49](#_Toc107923888)

[Figure 79 - Property "Text" of custom font 49](#_Toc107923889)

[Figure 80 - Press SHIFT to constrain vertical 50](#_Toc107923890)

[Figure 81 - Slide up/down to set y-coordinate 50](#_Toc107923891)

[Figure 82 - Press ALT to constrain horizontal 51](#_Toc107923892)

[Figure 83 - Slide left/right to set x-coordinate 51](#_Toc107923893)

[Figure 84 - Connect Device 63](#_Toc107923894)

[Figure 85 - Device connected 63](#_Toc107923895)

[Figure 86 - Select the correct device type 64](#_Toc107923896)

[Figure 87 - Managing device 64](#_Toc107923897)

[Figure 88 - Select Display List/Coprocessor 65](#_Toc107923898)

[Figure 89 - Display List command is highlighted in yellow 65](#_Toc107923899)

[Figure 90 - Prepare to draw a rectangle 66](#_Toc107923900)

[Figure 91 – Draw rectangle 66](#_Toc107923901)

[Figure 92 - Trace the pixel 67](#_Toc107923902)

[Figure 93 - Open example project 68](#_Toc107923903)

## List of Tables

[Table 1 - Installation Folder 11](#_Toc107923805)

[Table 2 - Memory Dump File Content 52](#_Toc107923806)

# Appendix C – Revision History

Document Title : BRT\_AN\_037 EVE Screen Editor 4.4 User Guide

Document Reference No. : BRT\_000231

Clearance No. : BRT#160

Product Page : https://brtchip.com/ic-module/toolchains/#EVEScreenEditor

Document Feedback : [Send Feedback](mailto:docufeedback@brtchip.com?subject=BRT_AN_037%20Version%20Versiont%201.3)

|  |  |  |
| --- | --- | --- |
| Revision | Changes | Date |
| Draft Version 0.6 | Initial Release of ESE 3.3 | 15-11-2019 |
| Version 1.0 | Updated release (content updated with respect to ESE 4.0 version) | 03-12-2020 |
| Version 1.1 | Updated as per ESE Version 4.1 | 10-12-2020 |
| Version 1.2 | Updated as per ESE Version 4.2 | 27-10-2021 |
| Version 1.3 | Updated as per ESE Version 4.3 | 05-04-2022 |

**Internal Revision History**

Revision history (internal use only, please clearly state all changes here before saving the file)

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date**  **YYYY-MM-DD** | **Changes** | **Editor** |
| Draft 0.1 | 08-10-2020 | Initial Draft prepared using BRT\_AN-037 EVE Screen Editor 3.3 User Guide as a base; Content updated with respect to the latest version (ESE 4.0) | Ly Viet Truong |
| Draft 0.2 | 20-10-2020 | Reviewed the document; Formatted the document as required; Applied the BRT document standards; Performed grammatical check and updated as required | L Subramanian |
| Draft 0.3 | 05-11-2020 | Update after document is reviewed by L Subramanian and Aaron Jones | Ly Viet Truong |
| Draft 0.3 | 10-11-2020 | Reviewed, completed minor formatting changes | A Jones |
| Draft 0.3 | 10-11-2020 | Reviewed and commented | G Brown |
| Draft 0.3 | 12-11-2020 | Update after document is reviewed | Ly Viet Truong |
| Draft 0.3 | 25-11-2020 | Reviewed and recommend for approval | Gavin Moore |
| Draft 0.3 | 02-12-2020 | Add section “Supported Devices”; replace "FT81X or later devices" to "FT81X or BT81X devices" | Ly Viet Truong |
| 1.0 | 03-12-2020 | Approved LCE | L Subramanian |
| 1.1 | 10-12-2020 | Updated with respect to ESE v4.1 | Ly Viet Truong |
| 1.2 | 16-09-2021 | Updated with respect to ESE v4.2 | Ly Viet Truong |
| 1.2 | 14-10-2021 | Reviewed, grammatical edits | A Jones |
| 1.2 | 14-10-2021 | Reviewed, some minor edits and comments | G Brown |
| 1.2 | 26-10-2021 | Reviewed; minor formatting | L Subramanian |
| 1.2 | 26-10-2021 | Address the CES Comments | Ly Viet Truong |
| 1.2 | 26-10-2021 | Document ready for Approval Recommendation | L Subramanian |
| 1.2 | 26-10-2021 | Reviewed and recommend for approval | Gavin Moore |
| 1.2 | 27-10-2021 | Approved LCE | L Subramanian |
| 1.3 | 10-01-2022 | Updated with respect to ESE v4.3 | Ly Viet Truong |
| 1.3 | 21-03-2022 | Reviewed with some comments and questions about cmd\_loadimage | Jiao Shouwu Paul |
| 1.3 | 21-03-2022 | Reviewed and Commented | A Jones |
| 1.3 | 22-03-2022 | Reviewed and commented | G Brown |
| 1.3 | 31-03-2022 | Reviewed, recommended for approval | A Jones |
| 1.3 | 01-04-2022 | Reviewed, recommended for approval | Gavin Moore |
| 1.3 | 05-04-2022 | Approved LCE | L Subramanian |
| 1.4 | 05-07-2022 | Updated with respect to ESE v4.4 | Ly Viet Truong |