USER GUIDE

Bridgetek

BRIDGING TECHNOLOGY

EVE Screen Editor 4.7 User Guide

Document Version: 1.7

Issue Date: 11-08-2023

Contents

I. Preface

A. Purpose

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

B. Intended Audience

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

C. Related Documents

Document Name	Document Type	Document Format
FT81x/BT88x Programming Guide	Programming Guide	PDF
FT81x Datasheet	Datasheet	PDF
FT800 Series Programmers Guide	Programming Guide	PDF
FT800 Embedded Video Engine Datasheet	Datasheet	PDF
BT81x Datasheet	Datasheet	PDF
BT81X Series Programmer Guide	Programming Guide	PDF

D. 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/.

II. Overview

A. Introduction

EVE **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 innovative 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/MPSSE 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!

B. Supported Devices

ESE supports all series of EVE chips, including FT80X, FT81X, BT88x 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*
BT88X	EVE Hal Projects	*VM880C(480x272)*
BT815/6	EVE Hal Projects	*VM816C50A, VM816CU50A*
BT817/8	EVE Hal Projects	*ME817EV, IDM2040-7A*

❖ 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

C. 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

D. What's new in ESE 4.7?

- Add python-like editor to write a simple emulator based application.
- Add example brower dialog.
- Add one welcome screen.
- Add IDM2040-43A support in the submenu of export menu.
- Define "Unsaved"/"Saved" states for Save toolbutton.
- Add more custom device in the device manager.
- Add HSF support in Registers.
- Update the text of device manager syncup feature.
- Add example project for cmd_runanim in loop mode.
- Improve content manager.
- Improve the user manual.
- Update the copyright date to 2023.

E. Resolved issues

- ESE crashes when drag and drop cmd_fillwidth into view port.

F. Known Issues & Limitations

The following are some of the known issues and limitations:

G. Credits

1. Open-Source Software

- Qt: http://doc.qt.io/qt-6/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.

2. 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.

III. Setup & Installation

A. System Requirements

To install ESE 4.7 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 BT8xx 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

B. 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
IDM2040-7A	FT4222, MPSSE
VM816CU50A	FT4222
VM816C50A	FT4222, MPSSE
ME813AU-WH50C	FT4222
ME812A-WH50R	FT4222, MPSSE
VM810C50A	FT4222, MPSSE
VM800B	MPSSE
VM880C	MPSSE

Note: For VM816C50A, ME812A-WH50R, VM810C50A, VM880C**,** and VM800B, a separate USB-MPSSE adapter is required.

C. 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

D. Installing ESE

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

i. Download the installation package from the link https://brtchip.com/ic-module/toolchains/#EVEScreenEditor.

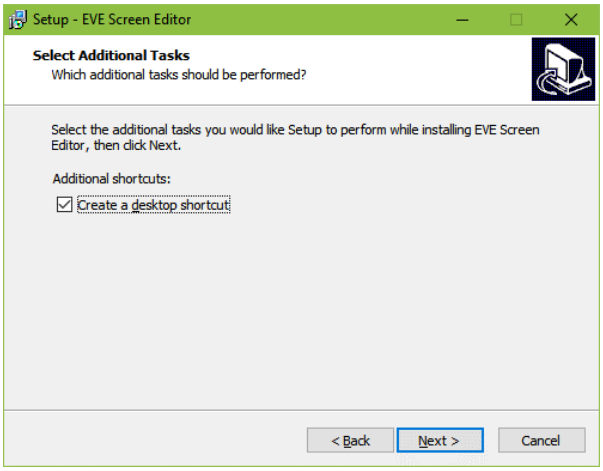
ii. When prompted with a download dialog box. Click on **Save**.

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

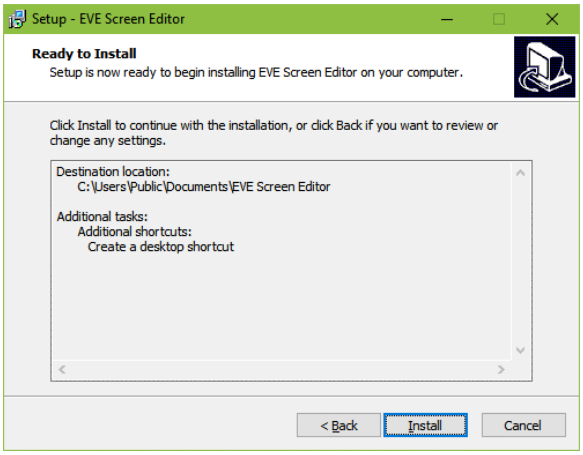
iv. Extract the zip file contents. Double click on the executable file

v. 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.

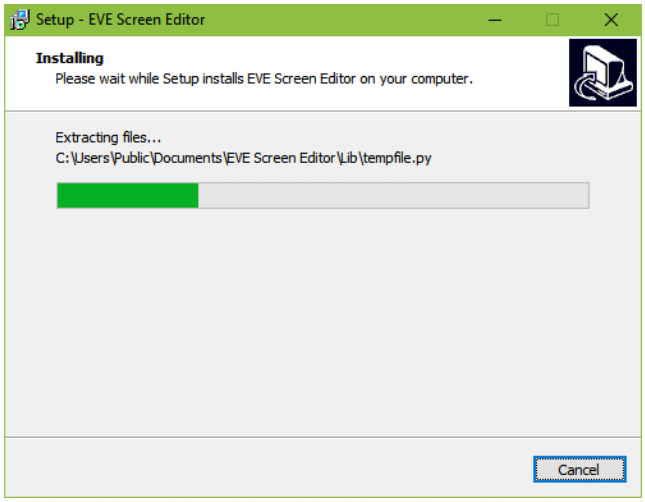
vi. In the **Select Additional Tasks** window, check the **"Create a desktop"** boxes, to have the ESE 4.7 icon displayed on the desktop if required. Click **Next** to prepare for the installation.



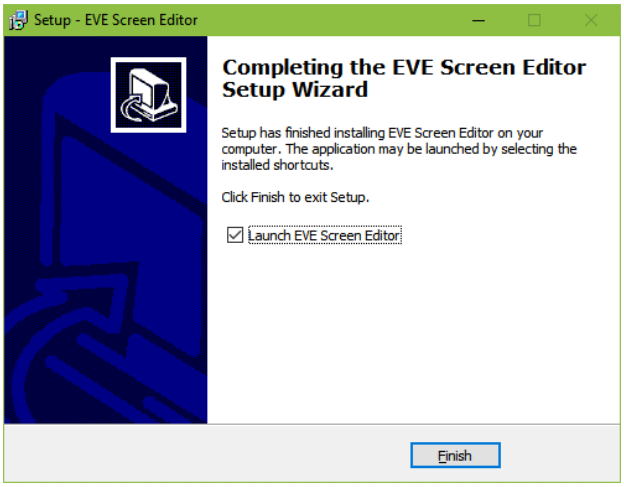
vii. The initial setup is completed, and the application is ready to be installed.



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



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



E. 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	this document	Read-Only
platforms	Qt platform run-time DLLs.	Read-Only
EVE_Hal_Library	The template project	Read-Only
untitled	The template project used by export feature	Read-Only
device_sync	information of built-in and custom boards	Read/Write
export_scripts	Python scripts to export project	Read-Only
firmware	flash **BLOB** for BT815/6 and BT817/8	Read-Only
iconengines	Qt DLL for supporting icon	Read-Only
__pycache__	Created by the Python interpreter	Read-Only

Table 1 - Installation Folder

IV. The Graphical User Interface

A. Overview

ESE user interface has the following components:

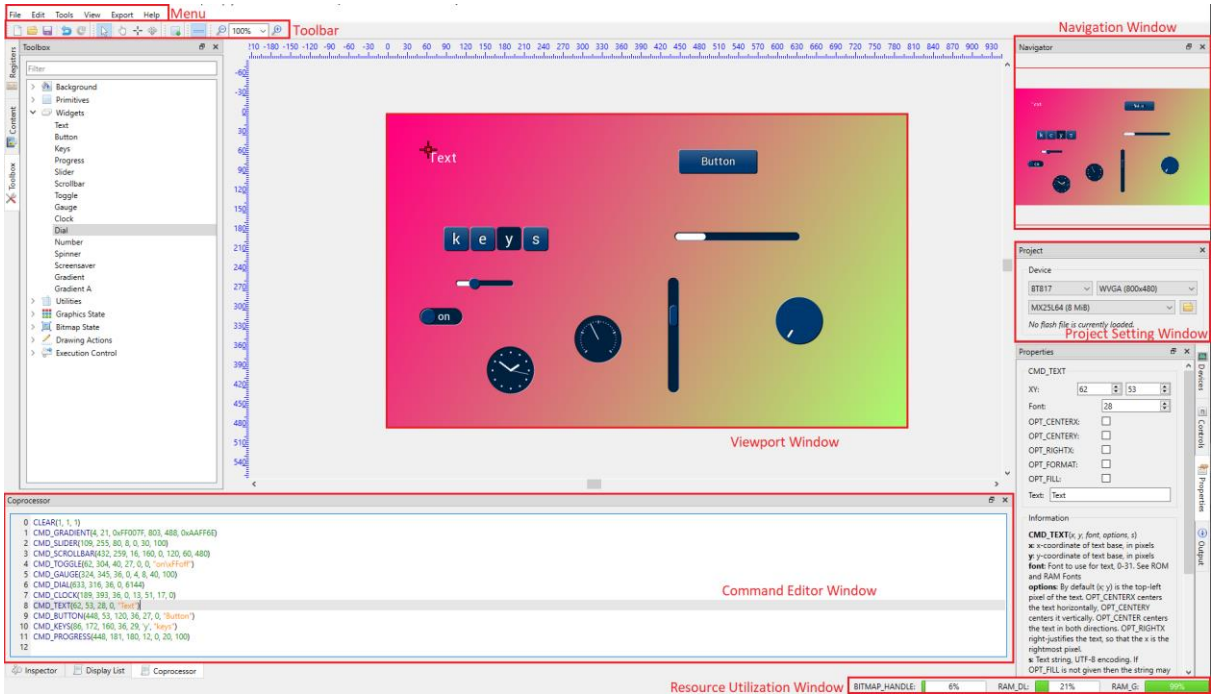


Figure 1 - User Interface Components

B. 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.

File Edit Tools View Export Help

1. 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.

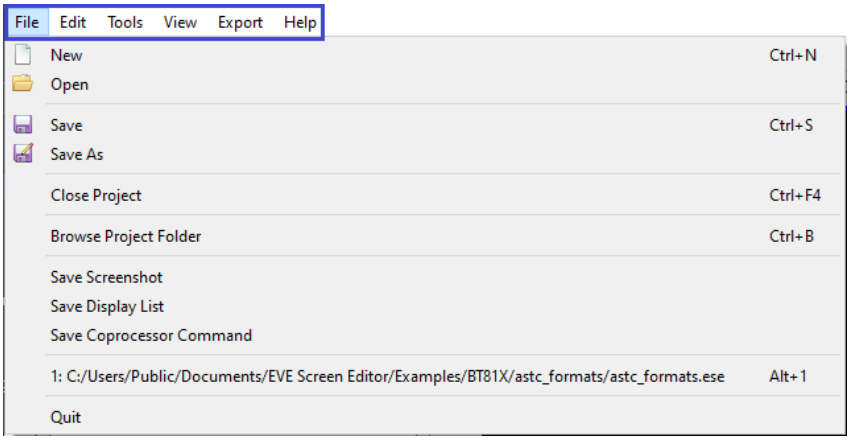


Figure 2 – File Menu

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 or a binary file, in Little Endian or Big-Endian format
Save Co-Processor Command	To save current coprocessor command to a text file or a binary 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*.

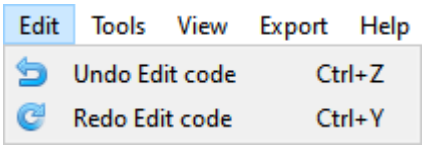


Figure 3 - Edit Menu

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.*

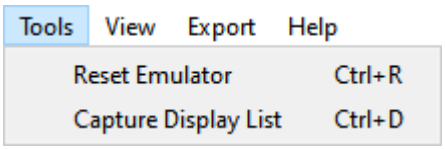


Figure 4 - Tool Menu

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.

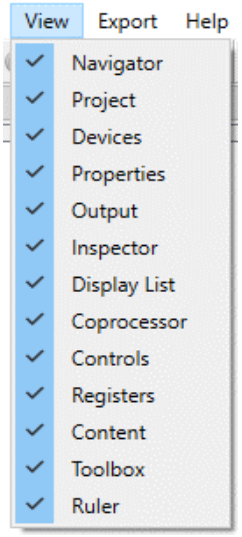


Figure 5 - View Menu

Export Menu

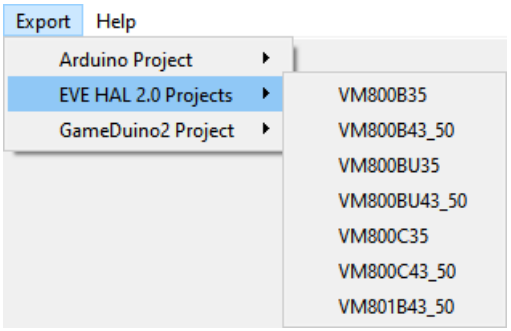


Figure 6 - Export Menu

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

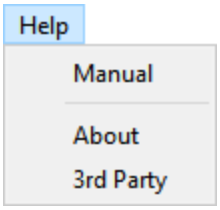


Figure 7 - Help Menu

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.

2. Toolbar

The toolbar defines shortcuts of mouse operation for *New*, *Open*, *Save*, *Undo*, *Redo*, *Cursor, Touch, Trace, Edit, Insert, Ruler, 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
Ruler	To trigger show/hide the ruler in emulator viewport
Zoom Out	To zoom out emulator viewport
Zoom In	To zoom in emulator viewport
Zoom Rate	To select concrete zooming rate

3. 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.

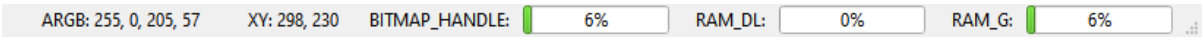


Figure 9 - Status Bar

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 hover

C. 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, RAM_REG, RAM_G and RAM_CMD and cannot be edited. RAM_DL, RAM_REG, RAM_G and RAM_CMD can be selected line by line and then copied to another text editor.

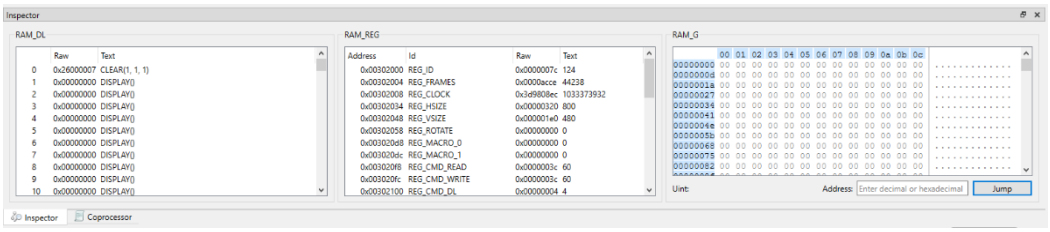


Figure 11 – Inspector

1. 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.

2. Display List Editor

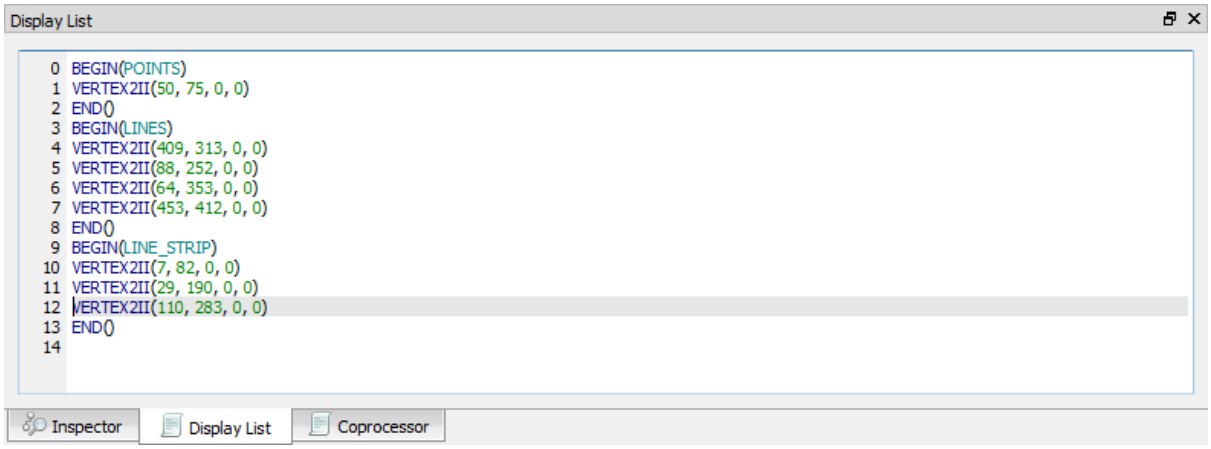


Figure 13 - Display List Editor

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.

3. Inspector

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

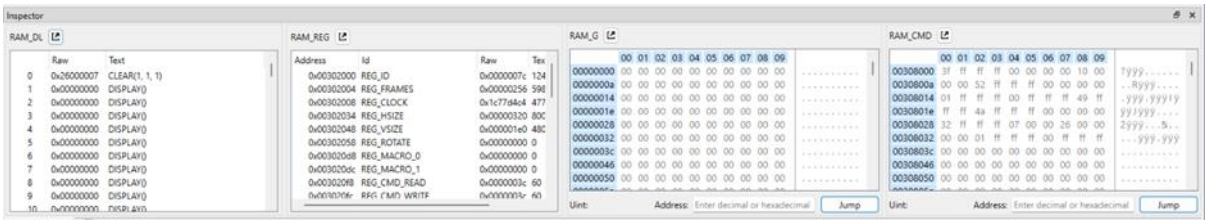


Figure 14 - Inspector

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.

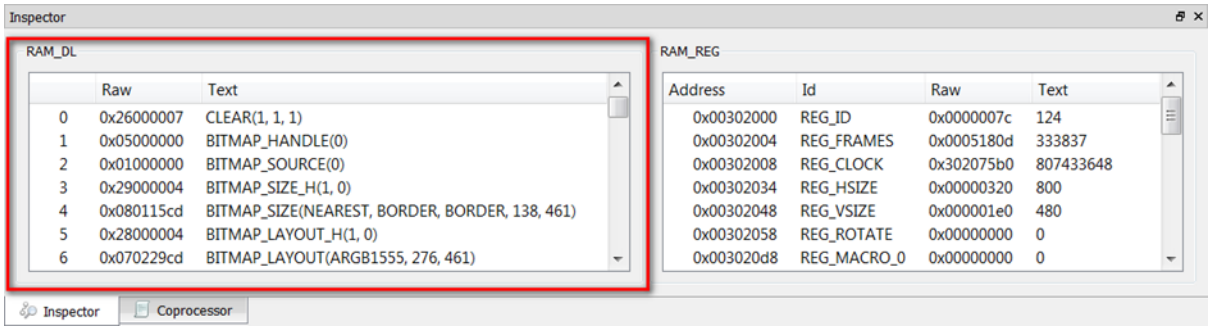


Figure 15 - RAM_DL

RAM_DL can be selected line by line and then copied to another text editor.

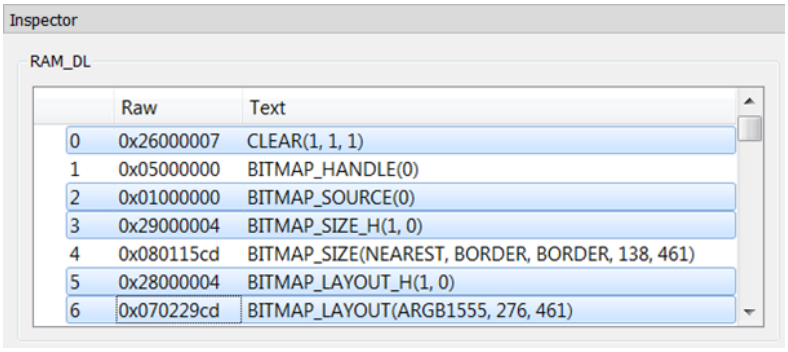


Figure 16 - Select rows in RAM_DL

	Raw	Text
0	0x26000007	CLEAR(1, 1, 1)
2	0x01000000	BITMAP_SOURCE(0)
3	0x29000004	BITMAP_SIZE_H(1, 0)
5	0x28000004	BITMAP_LAYOUT_H(1, 0)
6	0x070229cd	BITMAP_LAYOUT(ARGB1555, 276, 461)

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**.

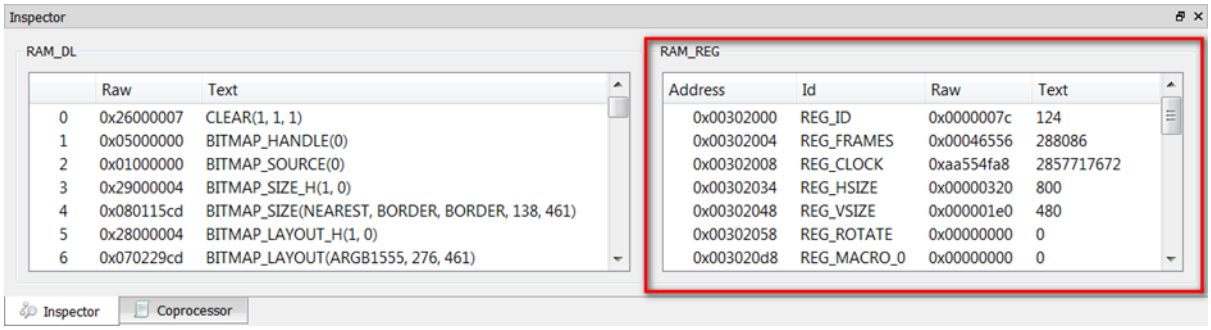


Figure 18 - RAM_REG

RAM_REG can be selected line by line and then copied to another text editor.

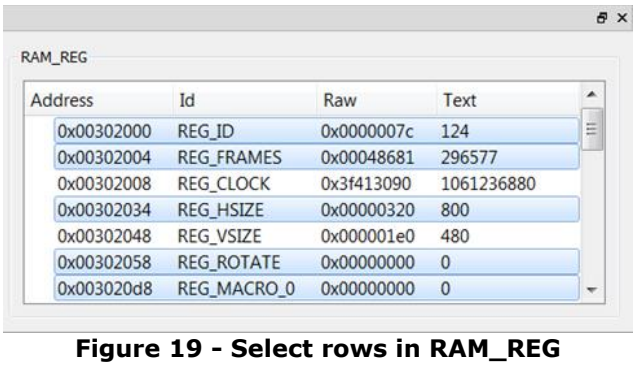


Figure 19 - Select rows in RAM_REG

Address	Id	Raw	Text
0x00302000	REG_ID	0x0000007c	124
0x00302004	REG_FRAMES	0x0000bd85	48517
0x00302008	REG_CLOCK	0x09b16888	162621576
0x00302034	REG_HSIZE	0x00000320	800
0x00302048	REG_VSIZE	0x000001e0	480
0x00302058	REG_ROTATE	0x00000000	0
0x003020d8	REG_MACRO_0	0x00000000	0

Figure 20 - Paste selected rows in RAM_REG

RAM_G

The **RAM_G** window in the Inspector tab shows the data of the RAM_G address and the corresponding character value. It provides the ability to jump to a specific address and view the Uint value of 4 consecutive bytes.

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

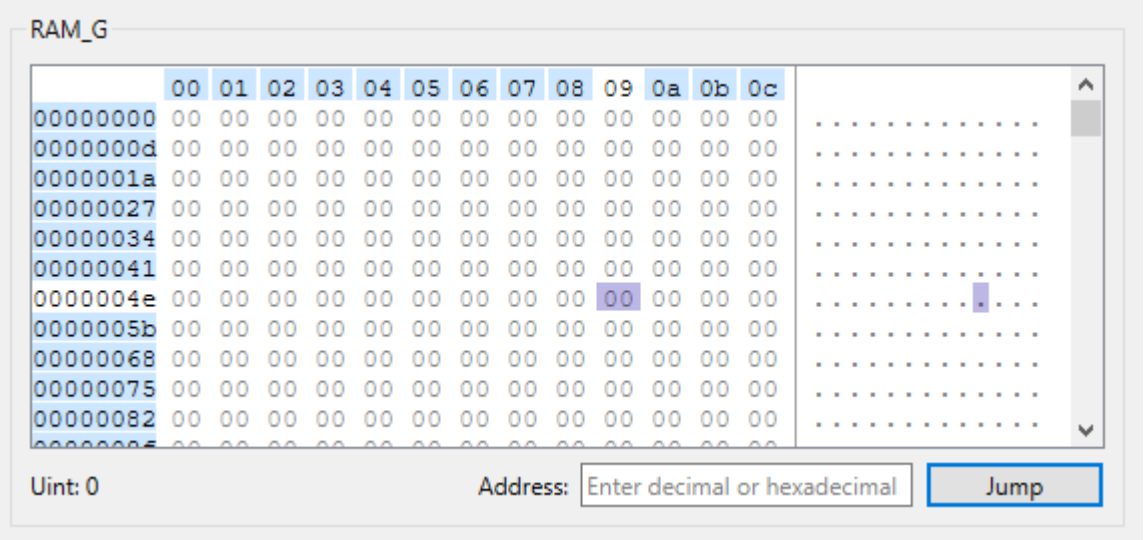


Figure 21 - RAM_G

RAM_G can be selected by area and then copied to another text editor.

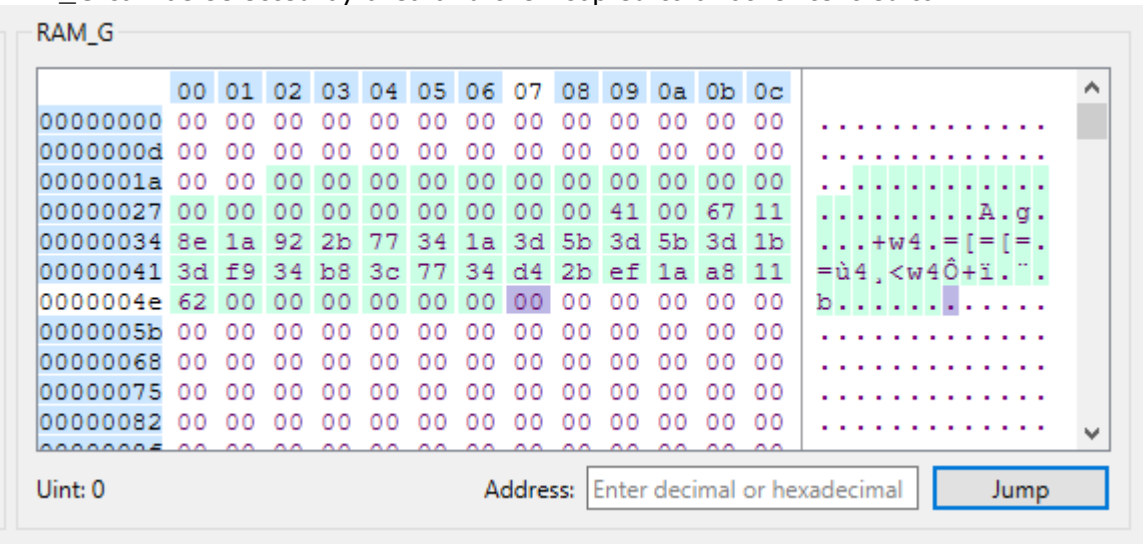


Figure 22 - Area selection

RAM_G can be jumped to a specific address based on the input of user

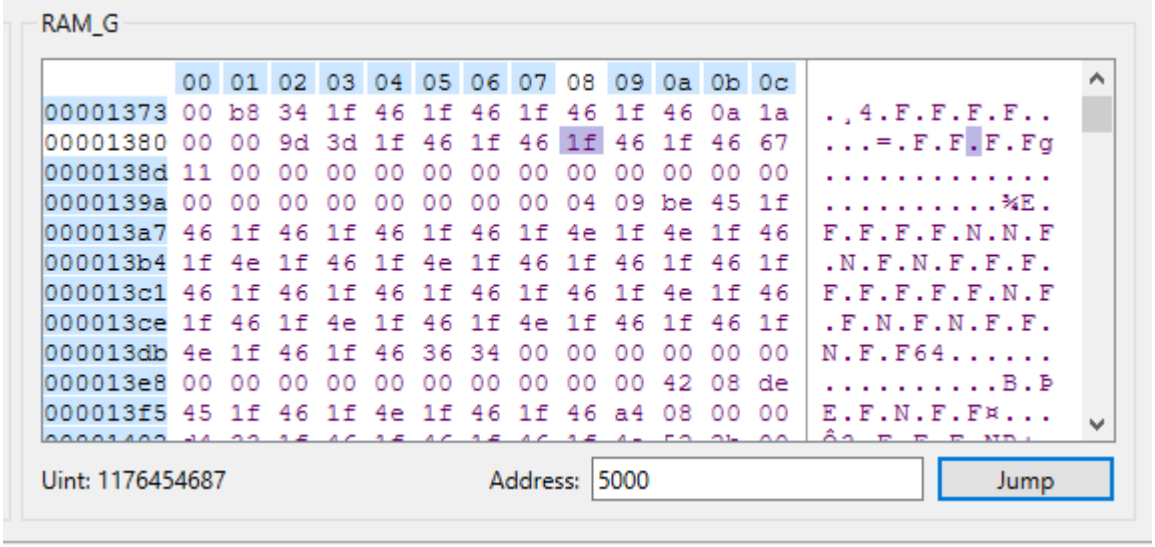


Figure 23 - Jump to a specific address with decimal

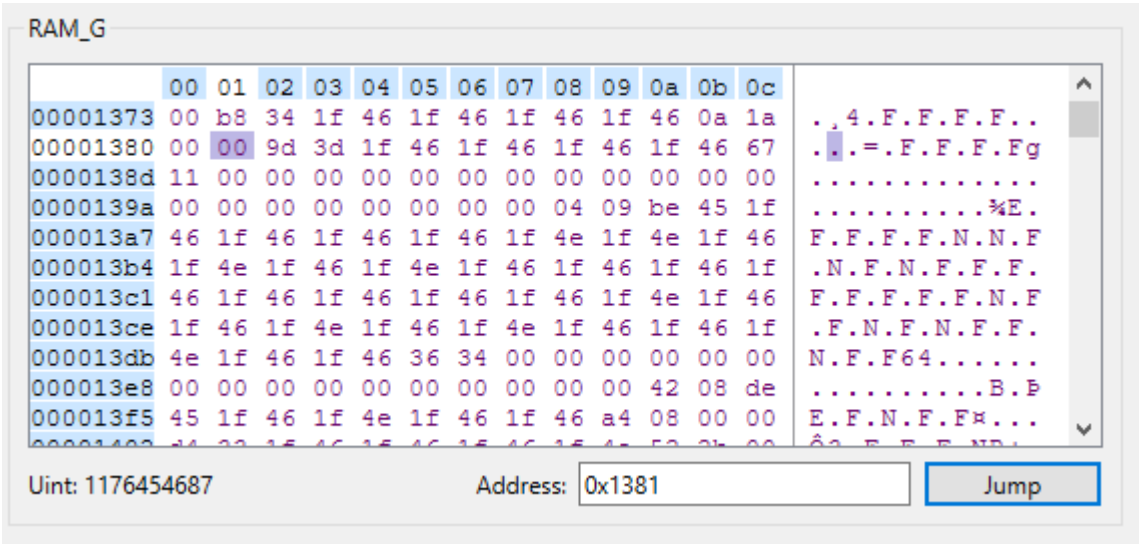


Figure 24 - Jump to a specific address with hexadecimal

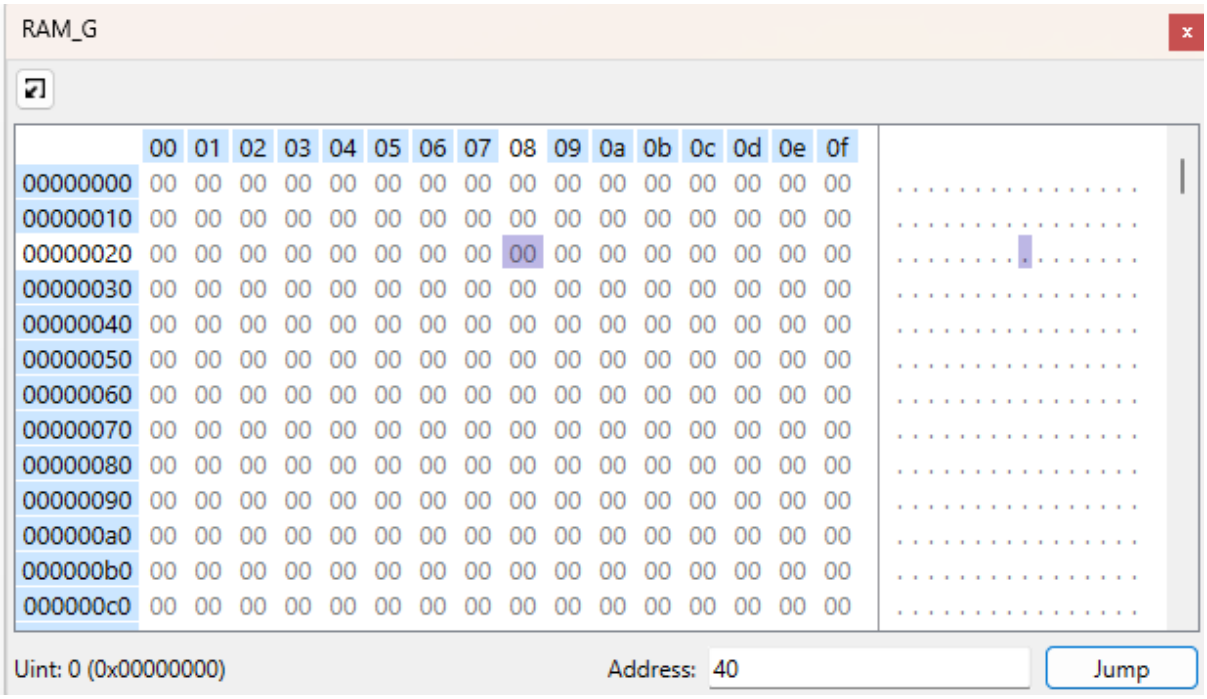


Figure 25 - Jump to a specific offset

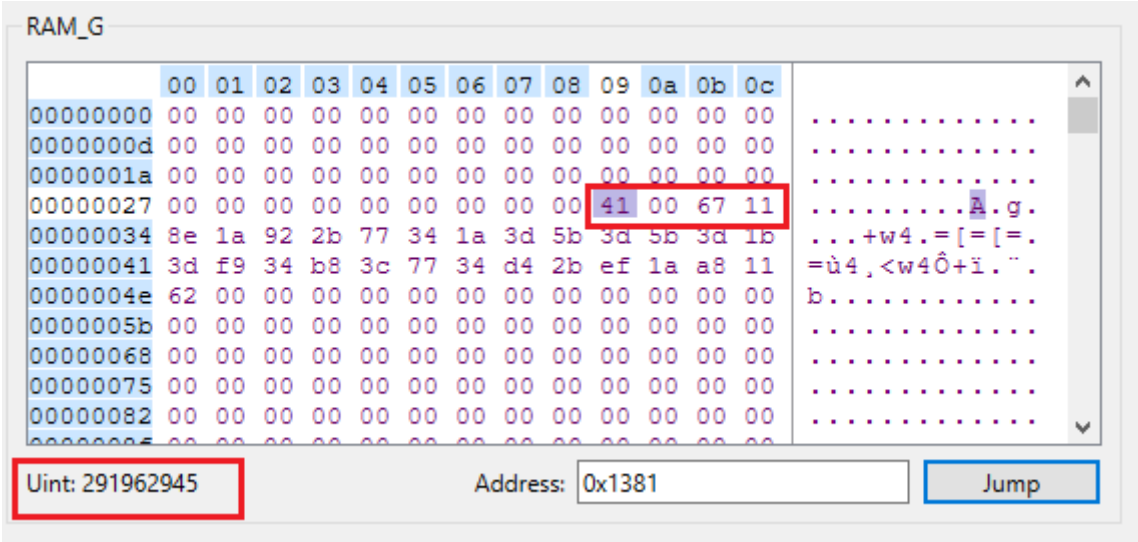


Figure 26 - The Uint value of 4 consecutive bytes

RAM_CMD

The **RAM_CMD** window in the Inspector tab shows the data of the RAM_CMD address and the corresponding character value. It provides the ability to jump to a specific address/offset and view the Uint value of 4 consecutive bytes.

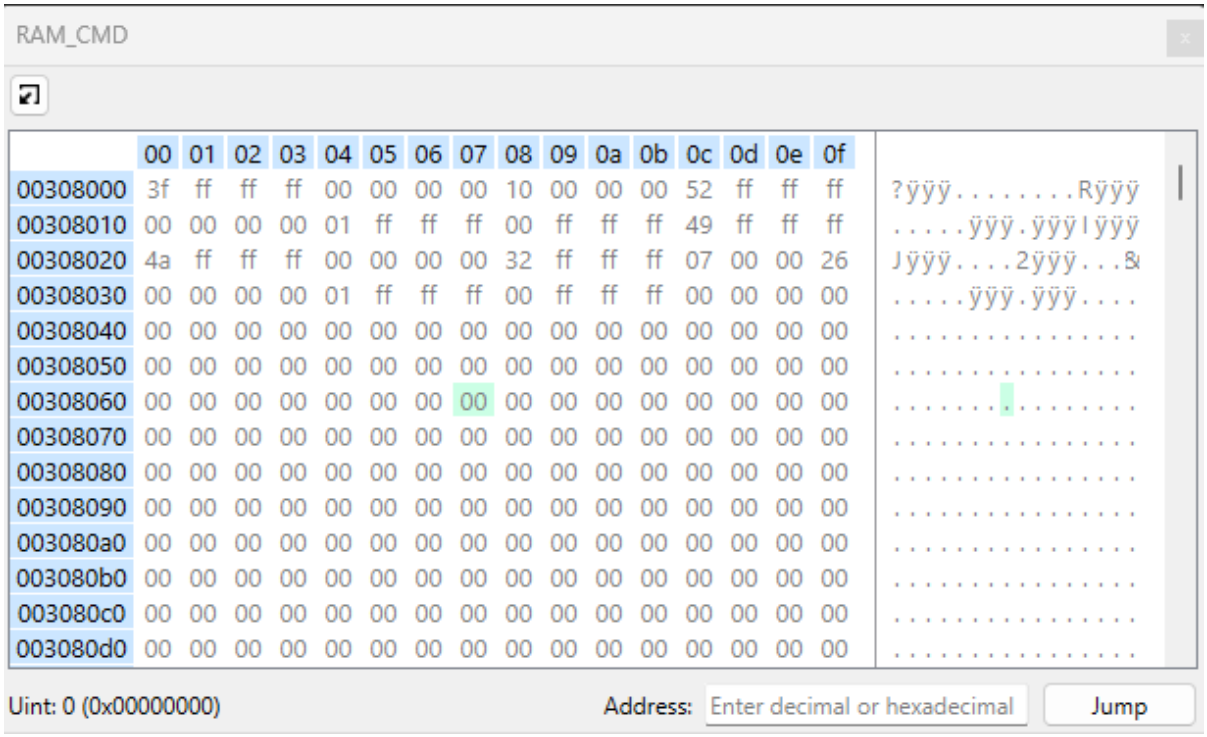


Figure 27 - RAM_CMD

RAM_CMD can be selected by area and then copied to another text editor.

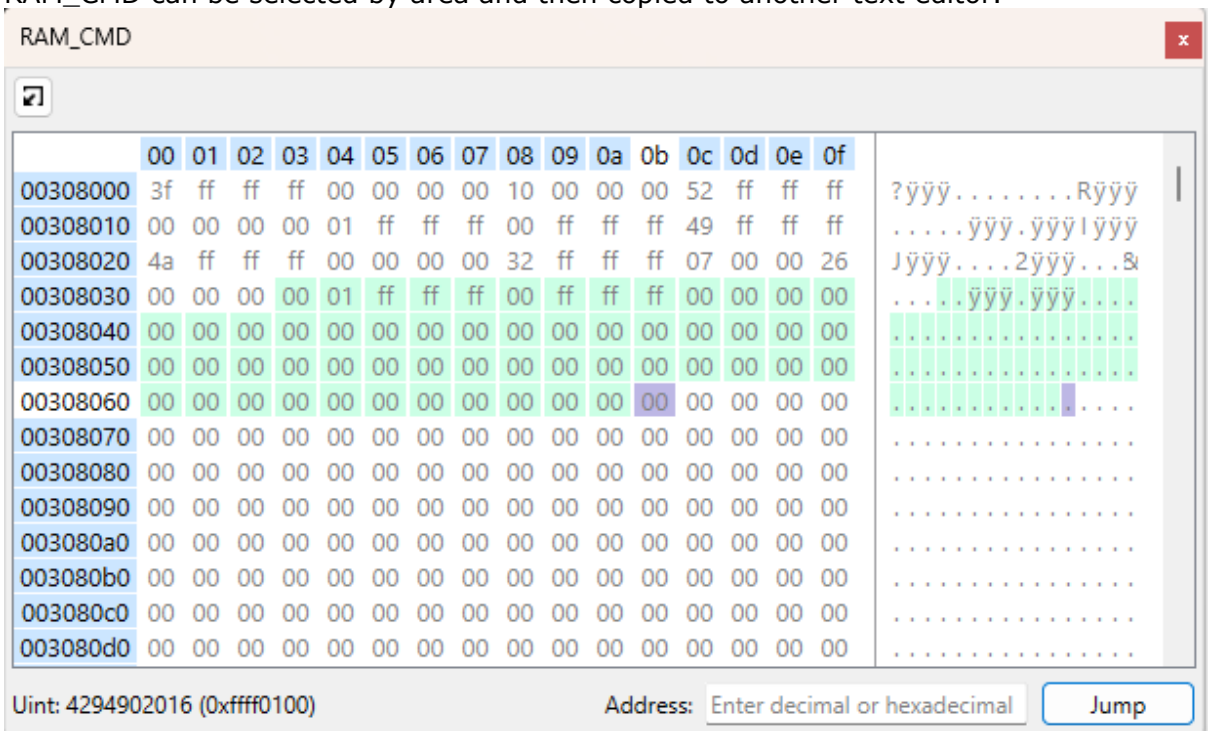


Figure 28 - Area selection

RAM_CMD can be jumped to a specific address/offset based on the input of user.

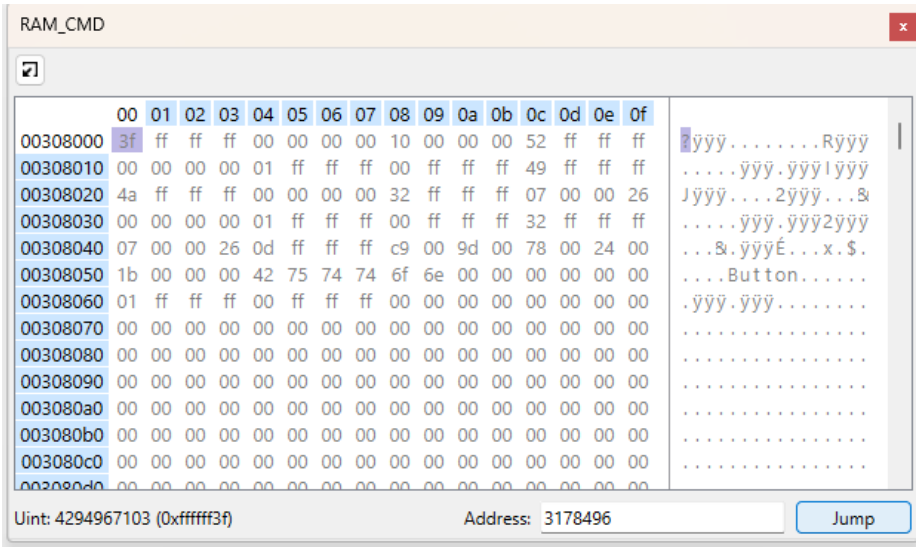


Figure 29 - Jump to a specific address with decimal

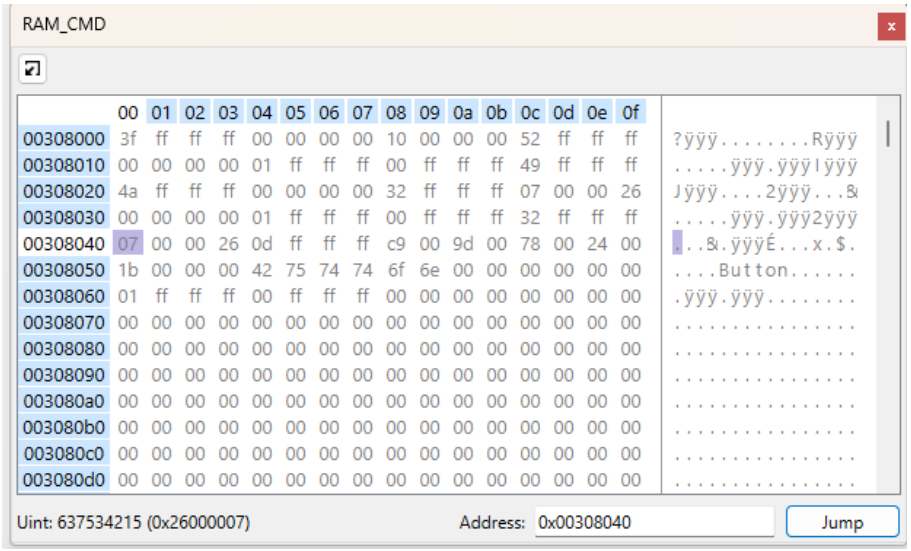


Figure 30 - Jump to a specific address with hexadecimal

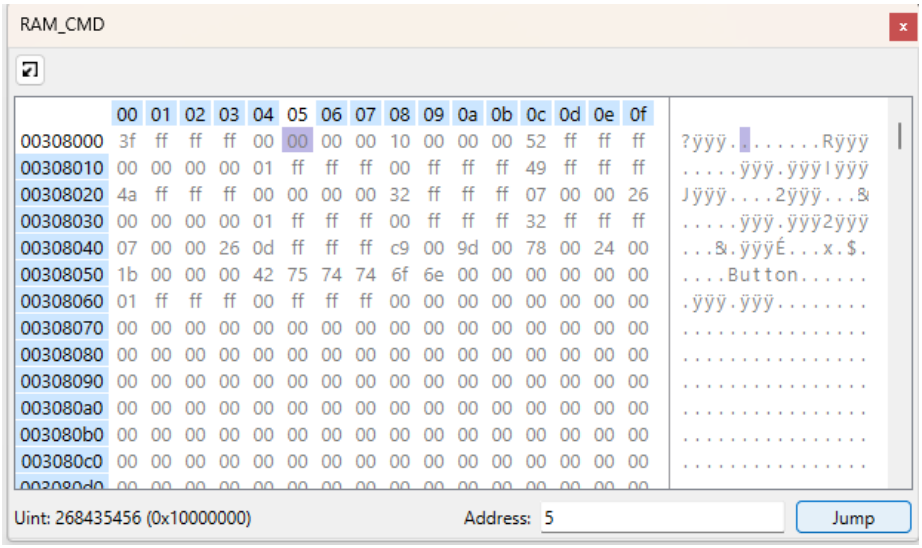


Figure 31 -Jump to a specific offset

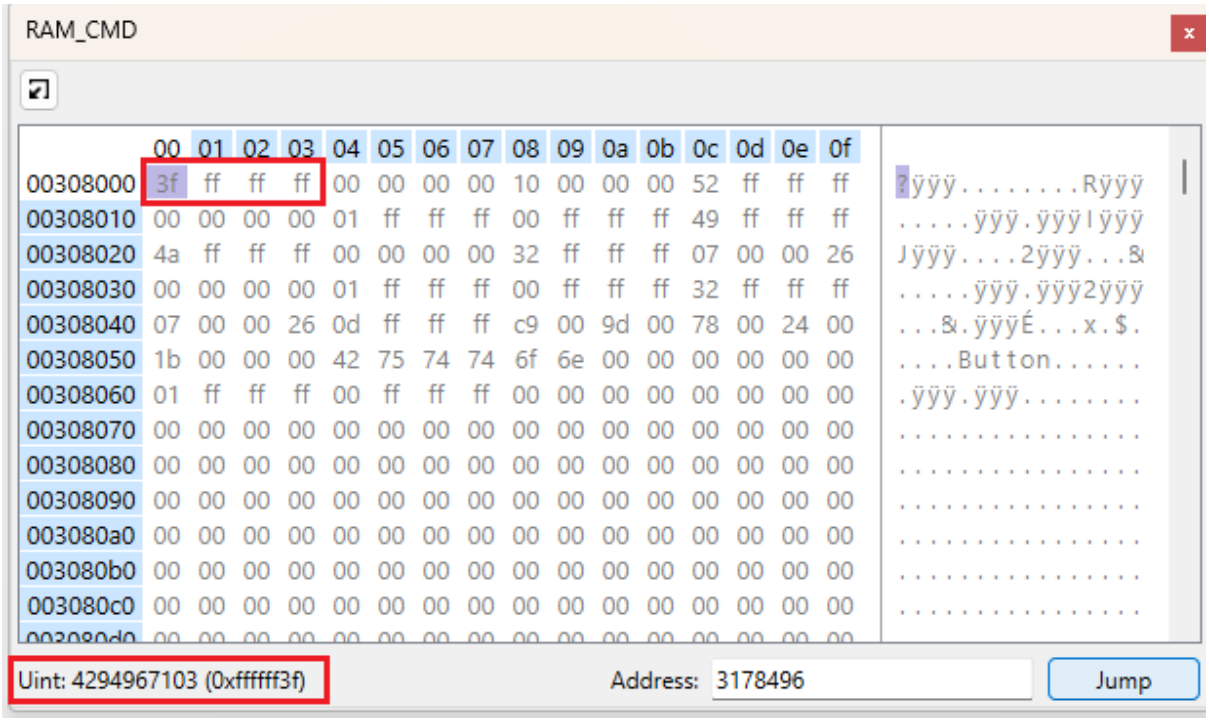


Figure 32 - The Uint value of 4 consecutive bytes

D. 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.

1. 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.

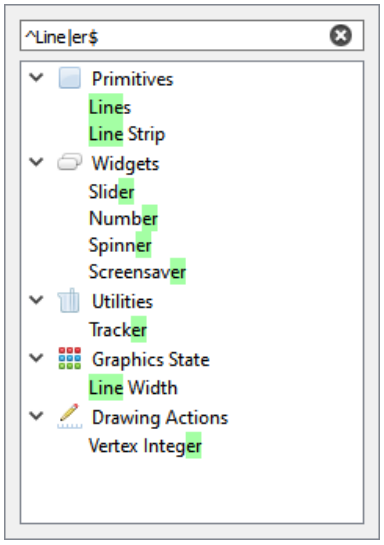


Figure 33 - Toolbox Filter

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.

Display list mode

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

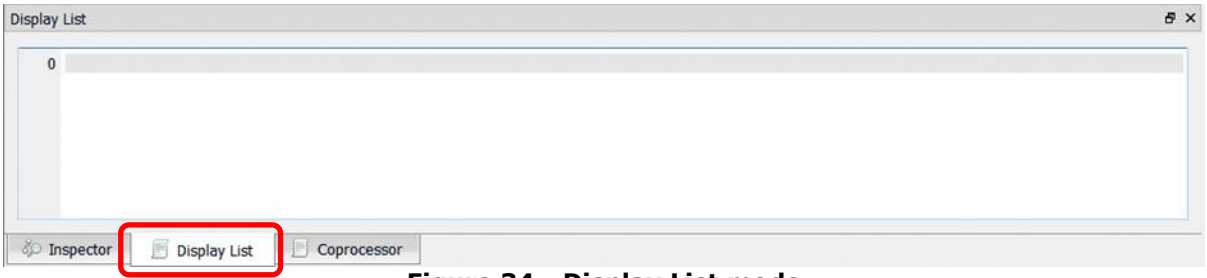


Figure 34 - Display List mode

The toolbox will be enabled as below:

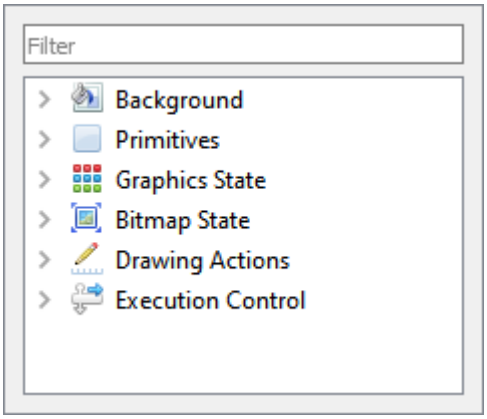


Figure 35 - Toolbox in Display List mode

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

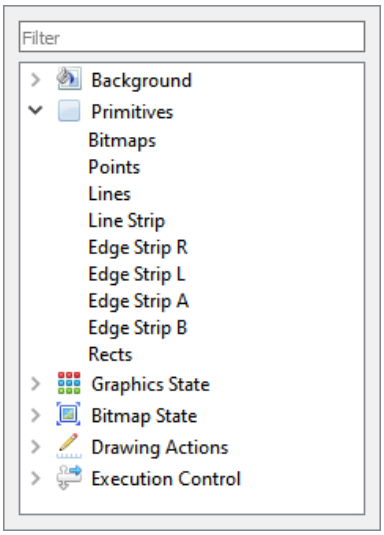


Figure 36 - Primitive in Display List Mode

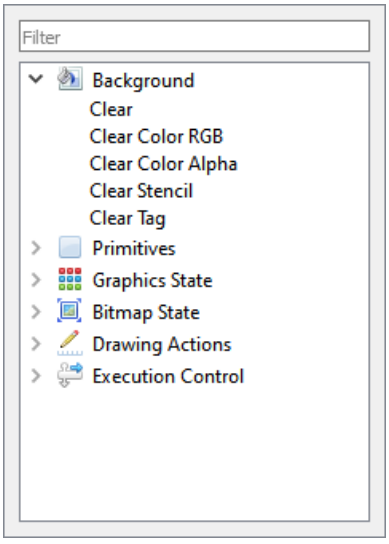


Figure 37 - Background in Display List mode

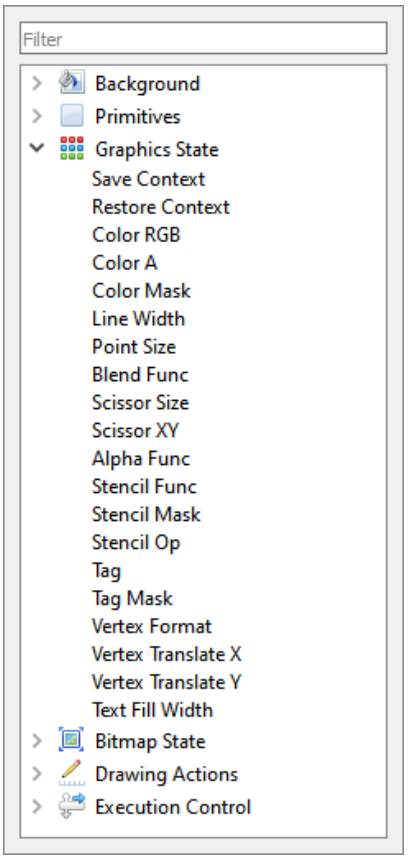


Figure 38 - Graphics State in Display List mode

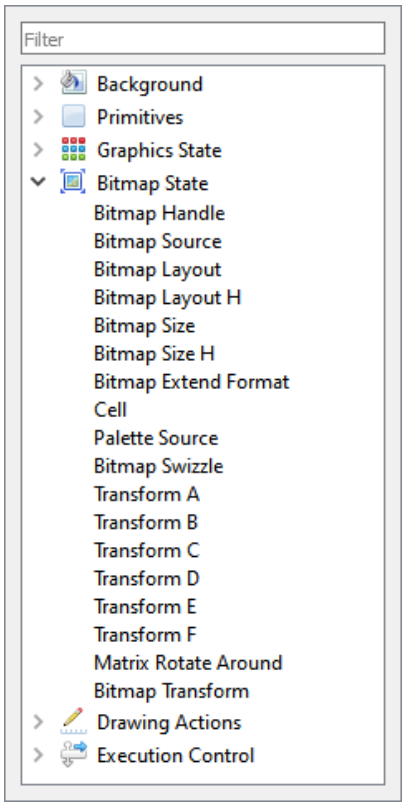


Figure 39 - Bitmap State in Display List mode

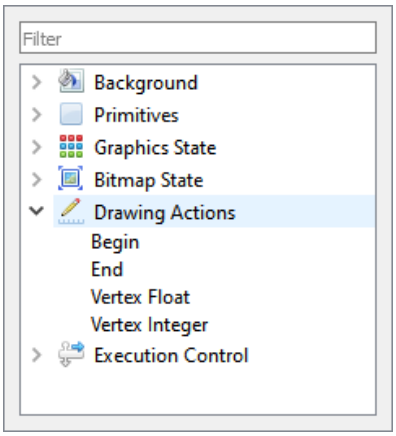


Figure 40 - Drawing Actions in Display List mode

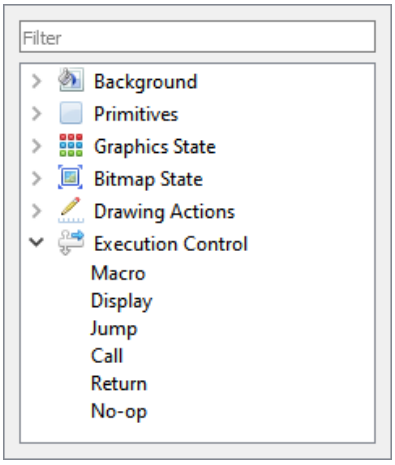


Figure 41 - Execution Control in Display List mode

Coprocessor mode

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

Figure 42 - Coprocessor mode

The toolbox will be enabled as below:

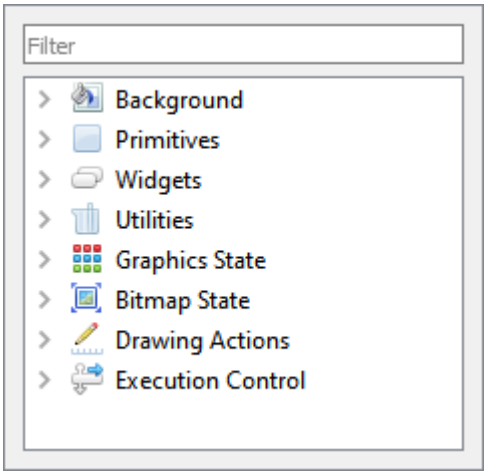


Figure 43 - Toolbox in Coprocessor mode

All commands are grouped into various categories based on functionality. In contrast to the display list mode, the inclusion of Widgets and Utilities categories has been introduced.

- Background
- Primitives
- **Widgets**
- **Utilities**
- Graphics State
- Bitmap State
- Drawing Actions
- Execution Control

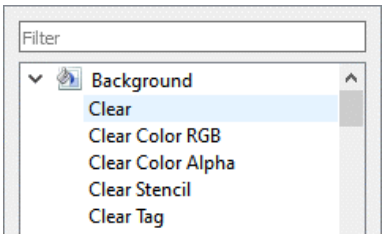


Figure 44 - Background in Co-processor mode

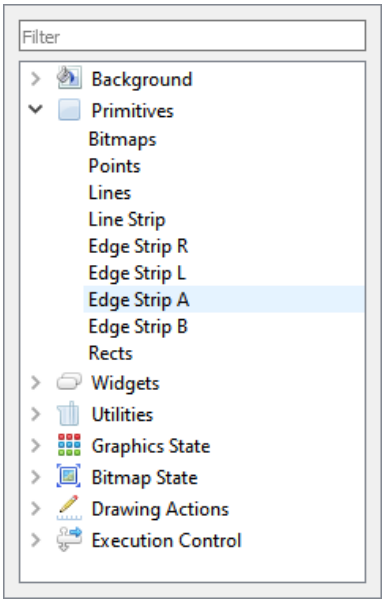


Figure 45 - Primitives in Co-processor mode

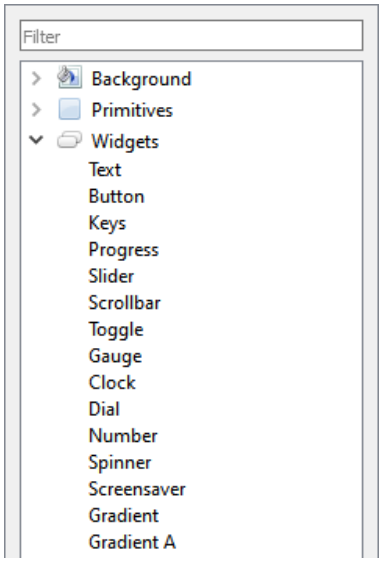


Figure 47 – Widgets in Co-processor mode

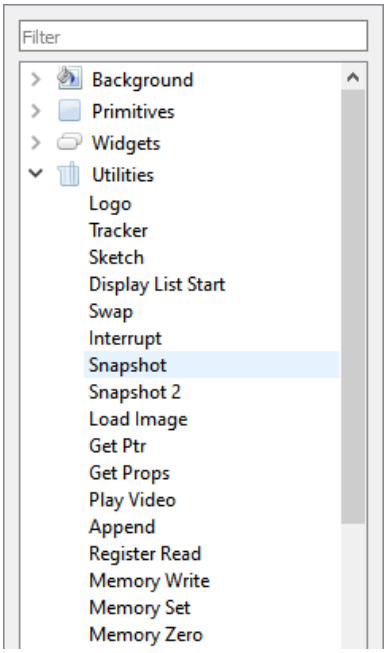


Figure 46 - Utilities in Co-processor mode

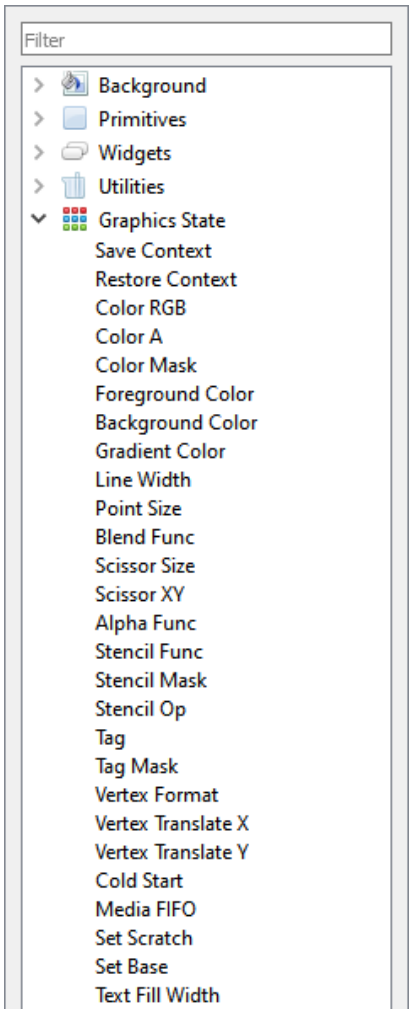


Figure 49 - Graphics State in Coprocessor mode

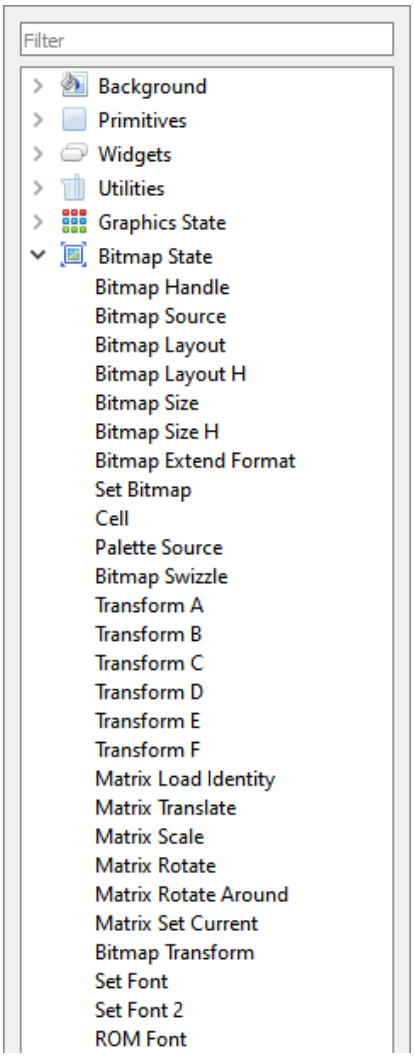


Figure 50 – Bitmap State in Coprocessor mode

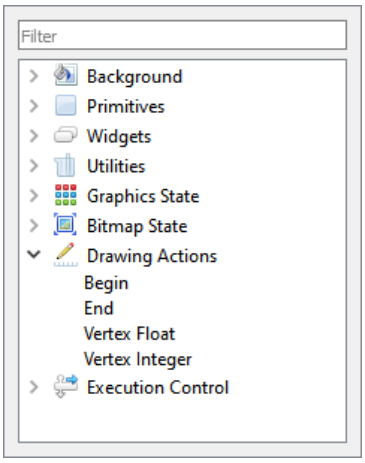


Figure 48 – Drawing Actions in Coprocessor mode

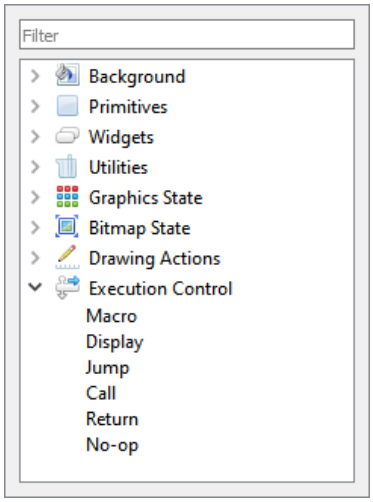


Figure 51 – Execution Control in Coprocessor mode

2. Registers

This tab is used to set up screen size and macro registers, control register's value such as **REG_ROTATE** and **REG_PLAY_CONTROL**. The **REG_MACRO_0** and **REG_MACRO_1** 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.

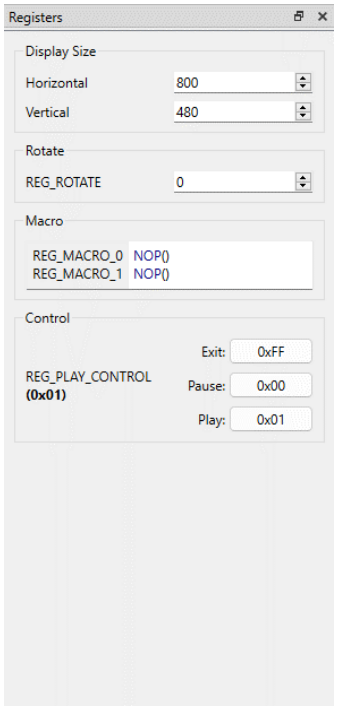


Figure 52 – Register

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.

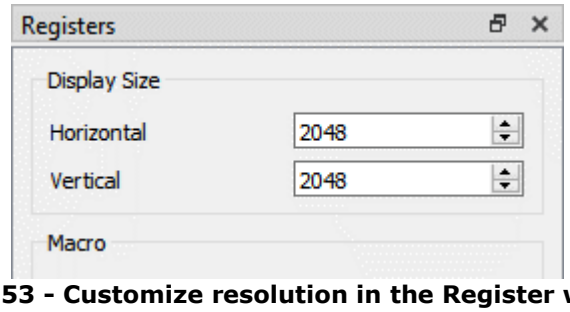


Figure 53 - Customize resolution in the Register window

The screen rotation can be changed using **REG_ROTATE**.

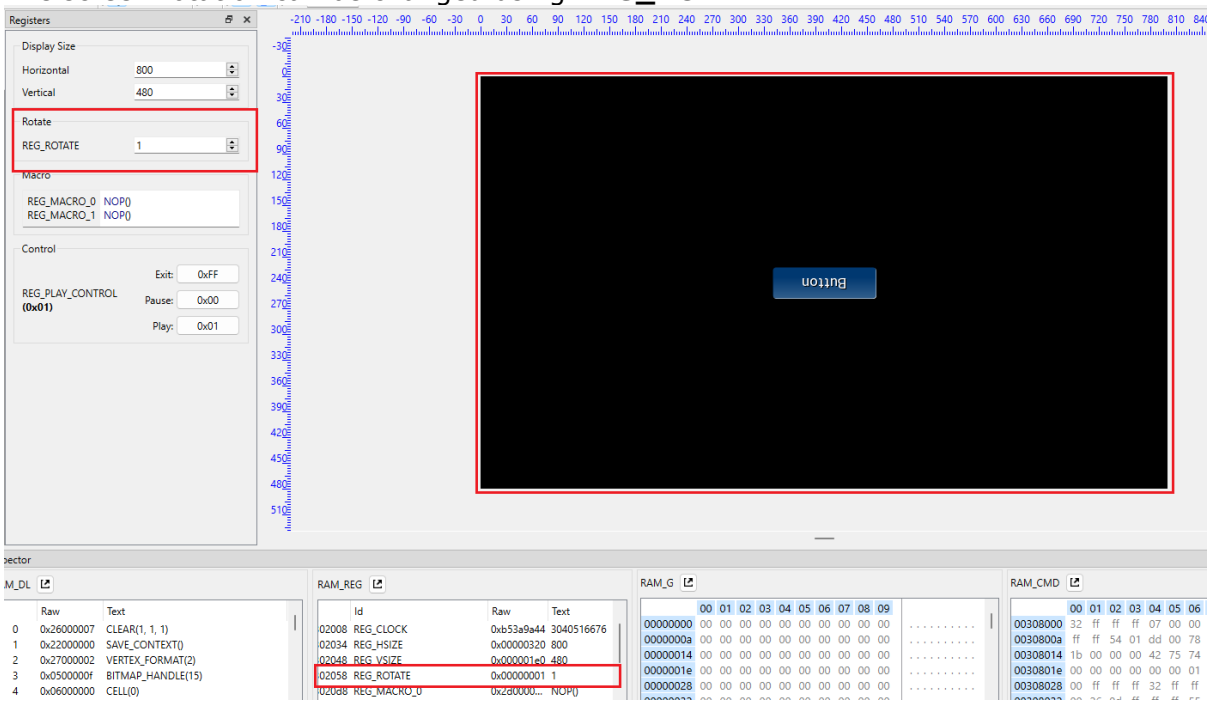


Figure 54 –Change screen rotation using REG_ROTATE

The playback may be paused or terminated by writing to **REG_PLAY_CONTROL**. This register's value can be controlled in the Control section.

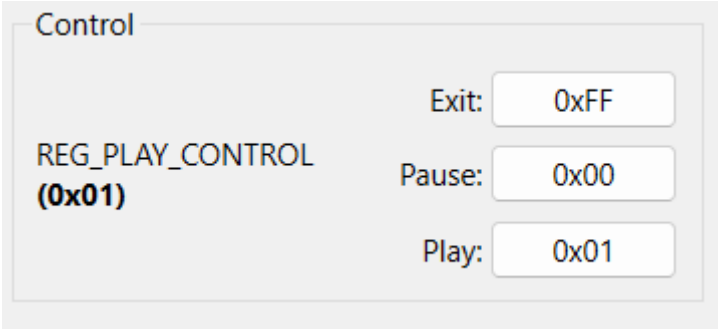


Figure 55 - Control section

3. 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.

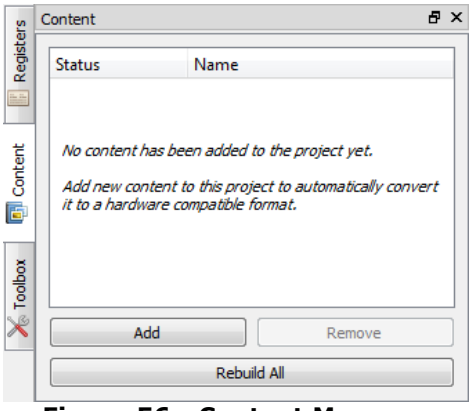


Figure 56 - Content Manager

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.

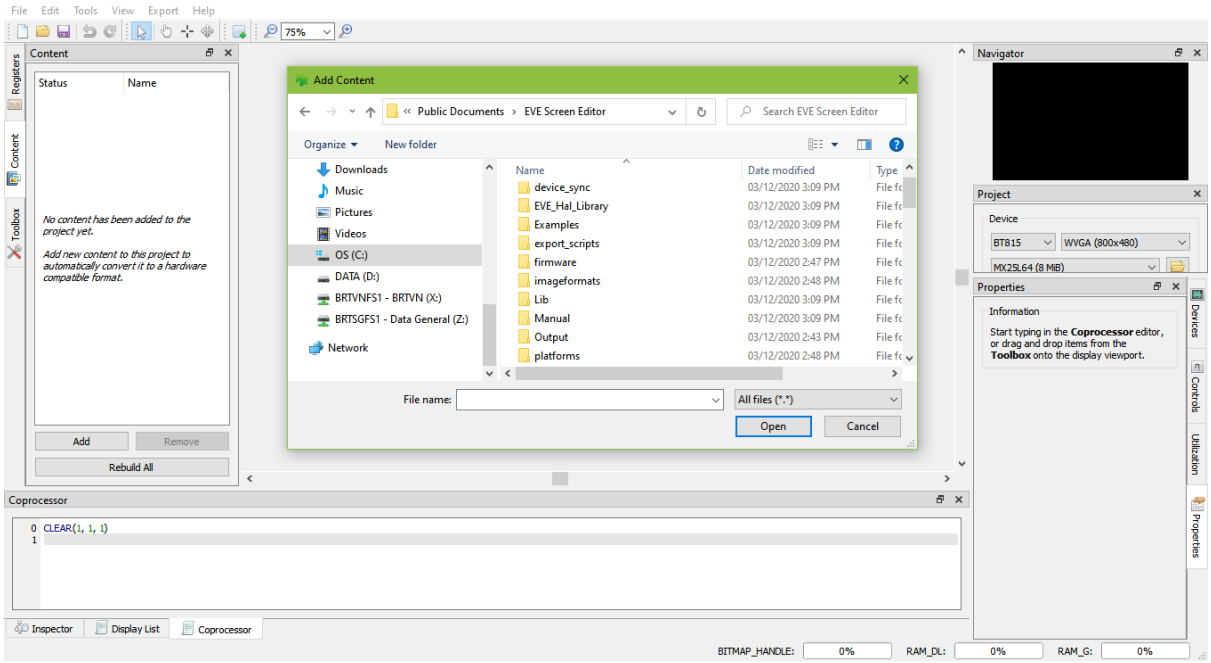


Figure 57 - Load content dialog

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

3. 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.

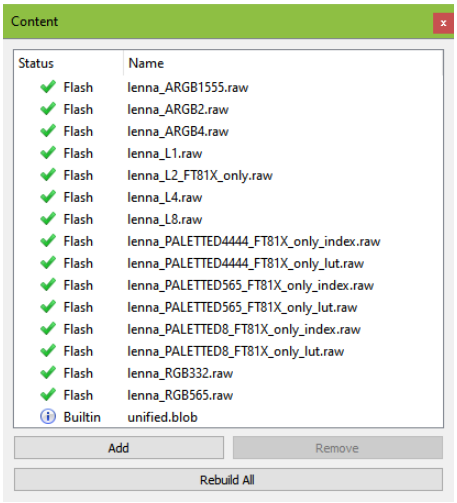


Figure 58 - Content loaded

4. 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.

5. 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.

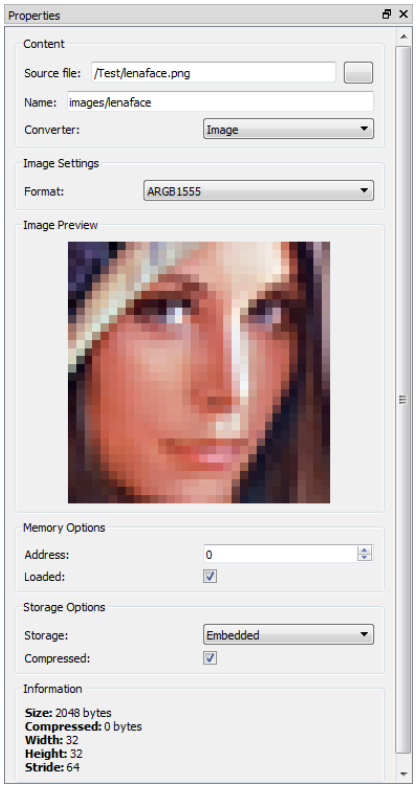


Figure 59 - Content properties

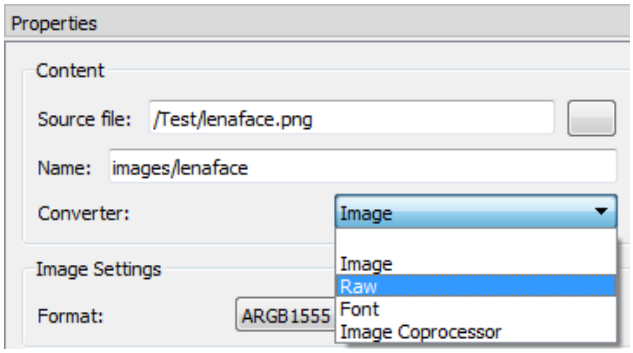


Figure 60 - Raw converter

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

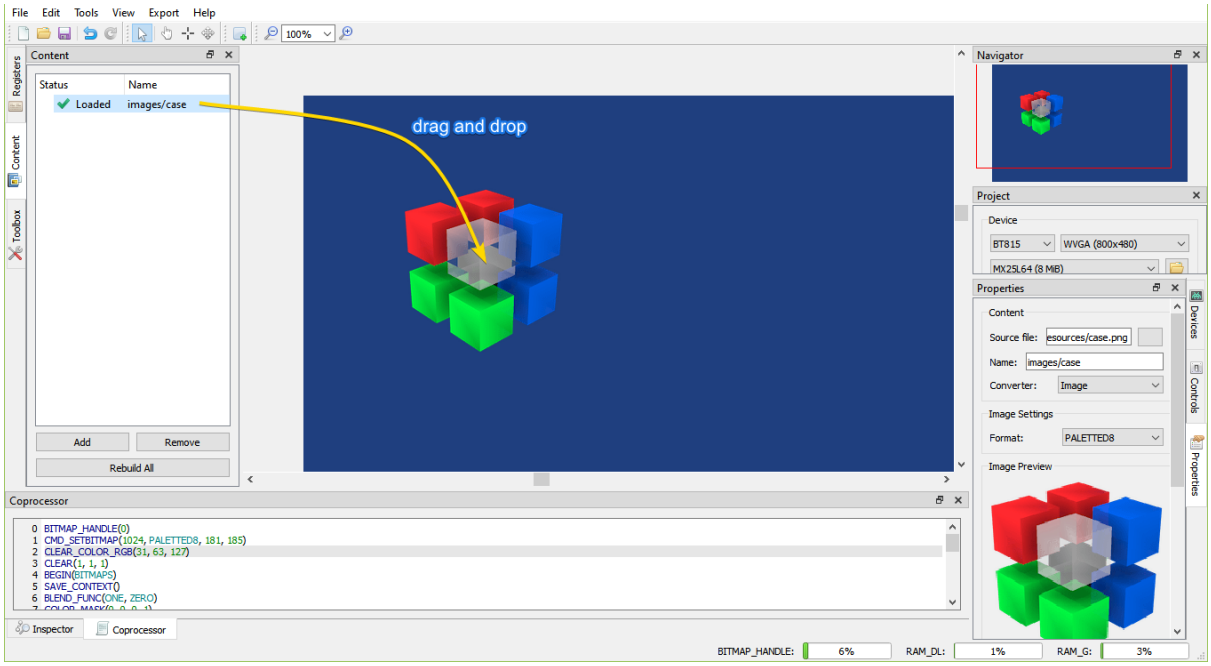


Figure 61 - Drag content into Viewport

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]**.

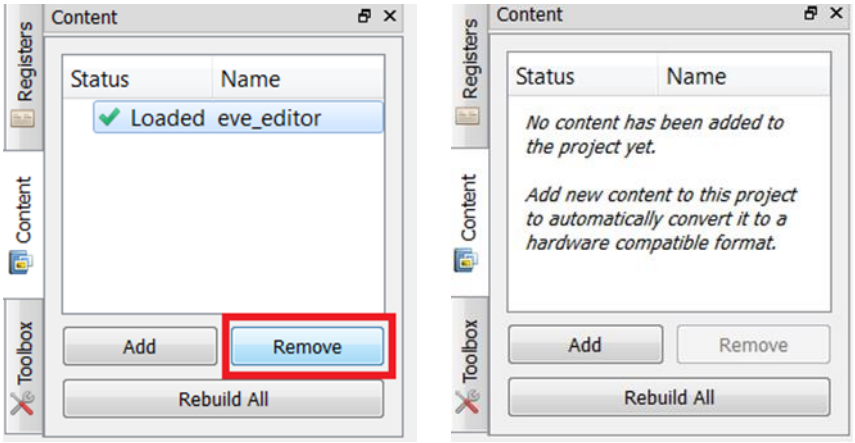


Figure 62 - Remove content

- 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.

Convert the content

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 Content Manager, the utility converts it to the below file formats:

**.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.
**.meta*	The file stores the basic information of a content item. It helps to check whether the utility should convert it again.
**.json*	The file stores the information of the converted image.
**_Converted.png, *_fs8.png*	The file is generated from the image through the utility when the converter is Image.
**_converted.png*	The file is generated from the font through the utility when the converter is Font.

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.

These original and converted files are stored in the following folder:

resources	Stores the original files
images	Stores the converted files when the converter is Image, Image Coprocessor
font	Stores the converted files when the converter is Font
content	Stores the converted files when the converter is Raw

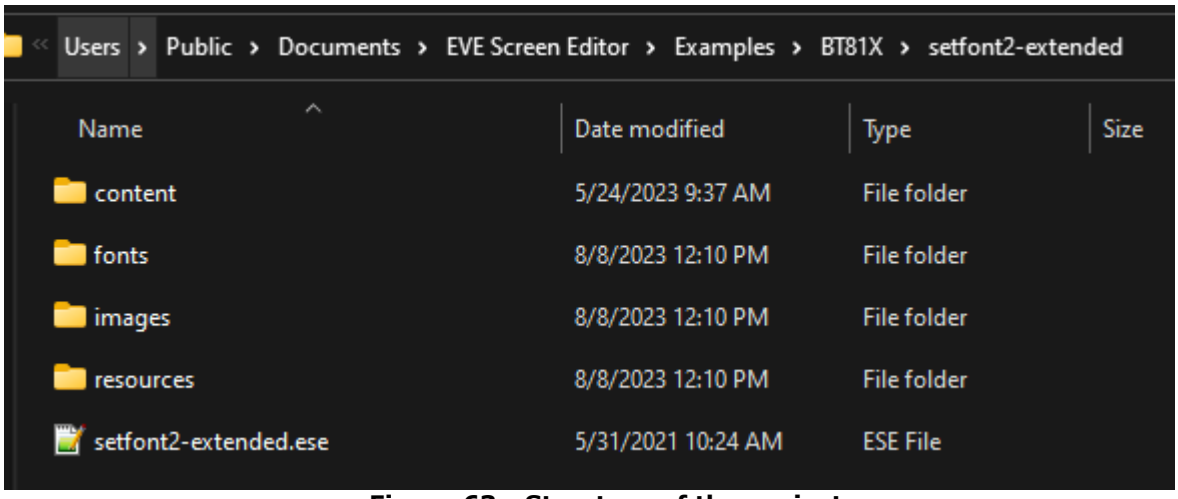


Figure 63 - Structure of the project

E. Devices, Controls, Properties, and Output

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

1. Device Manager

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

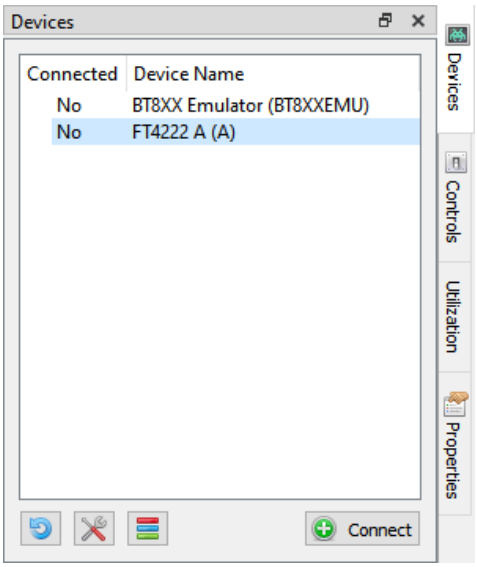


Figure 64 - Before connecting

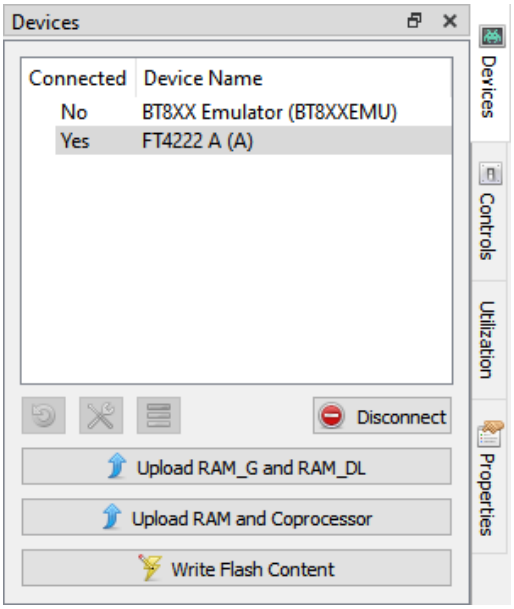


Figure 65 - Connected device

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.

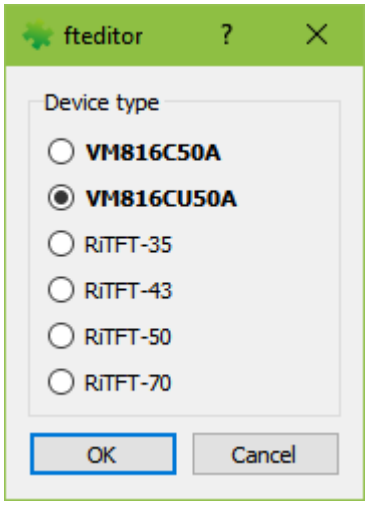


Figure 66 - Device type

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.

2. 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.

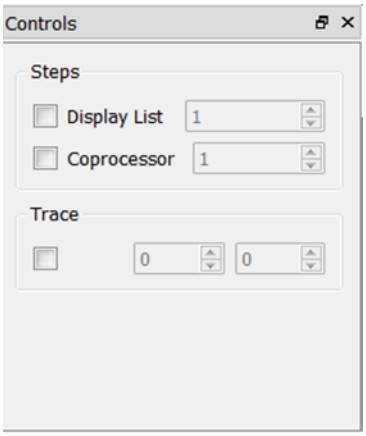


Figure 67 - Controls tab

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.

3. 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.

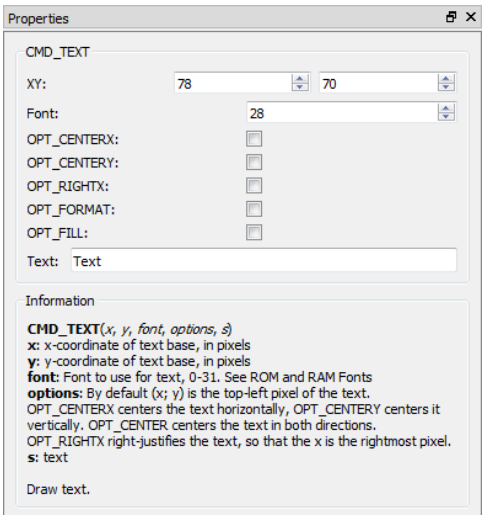
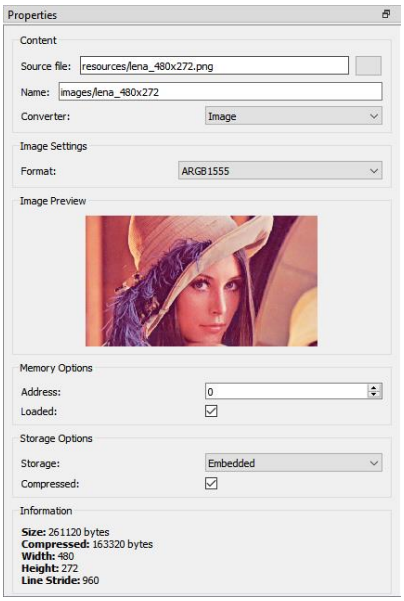


Figure 68 - Properties tab



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

4. Output

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

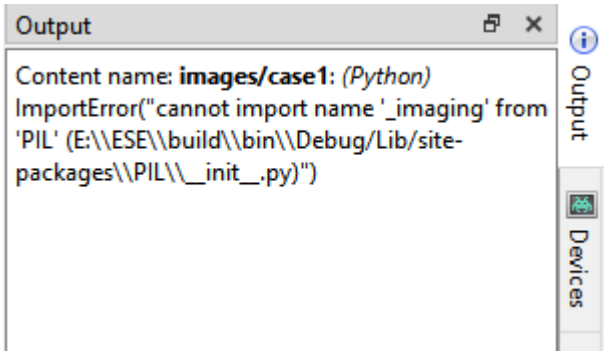


Figure 69 - Output tab

F. 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.

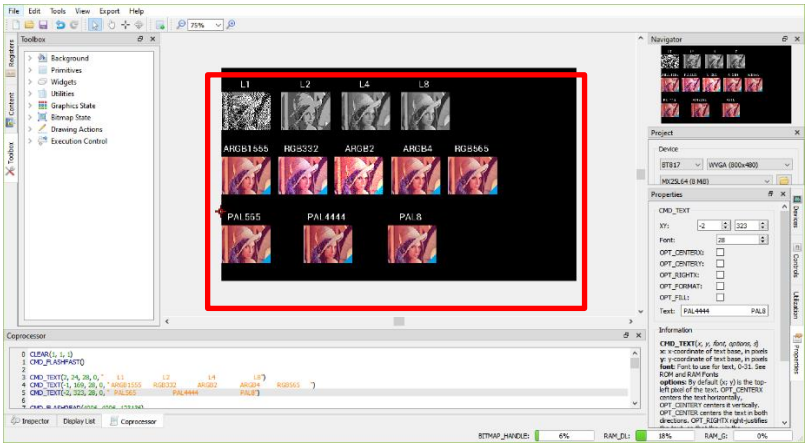


Figure 70 – Viewport

G. Navigator

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

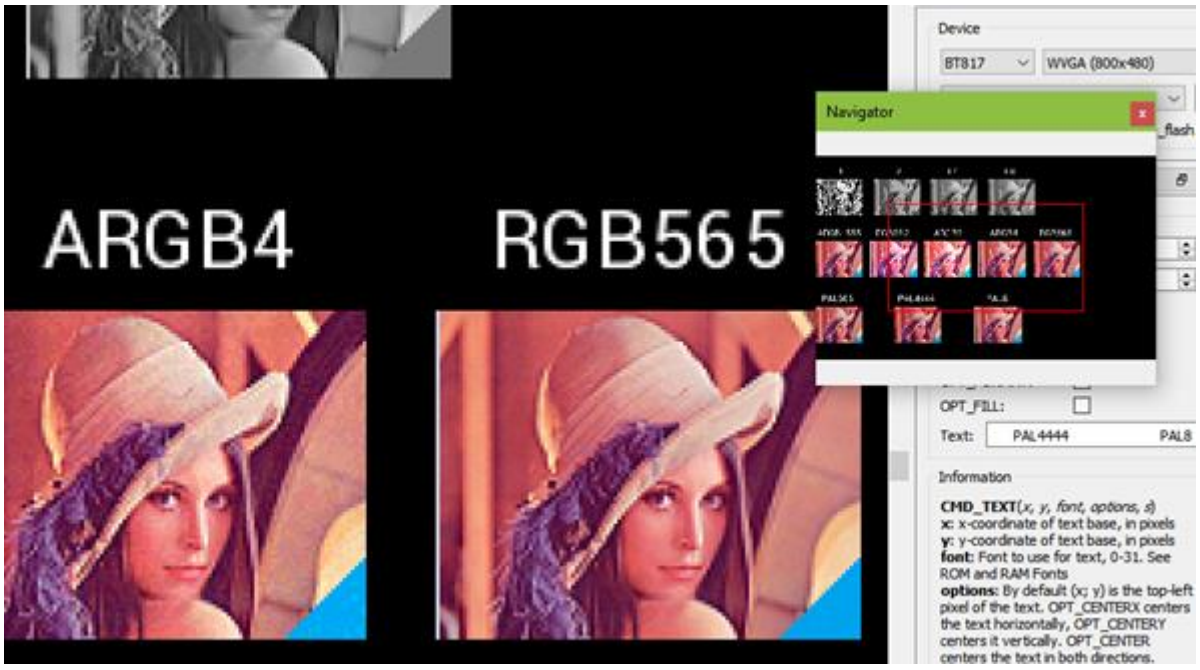


Figure 71 – Navigator

H. Project settings

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

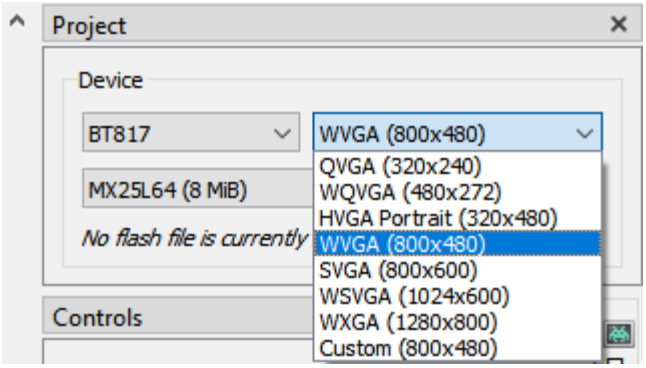


Figure 72 - Project settings

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.

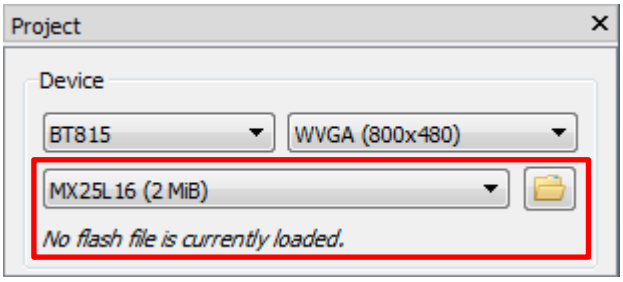


Figure 73 - Flash supported

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

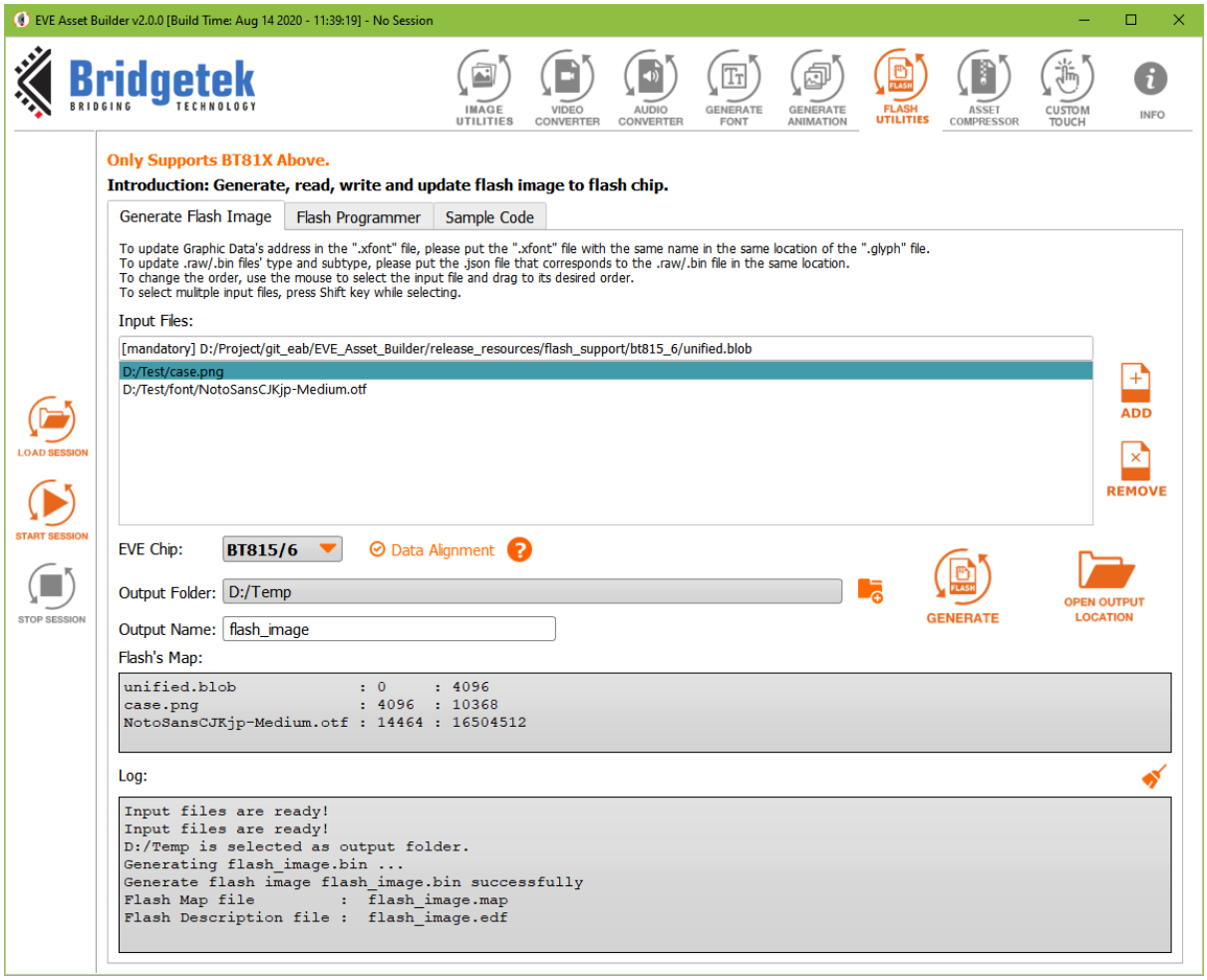


Figure 74 - Flash image generation by EAB

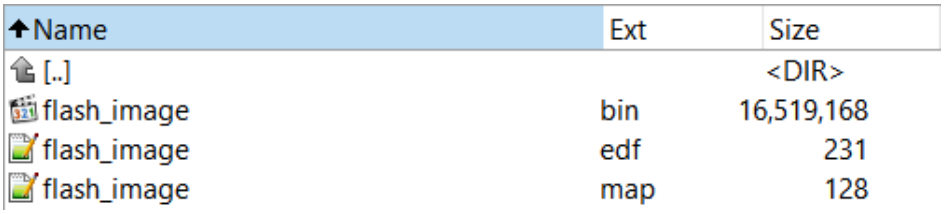


Figure 75 - Generated files in the output folder

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.

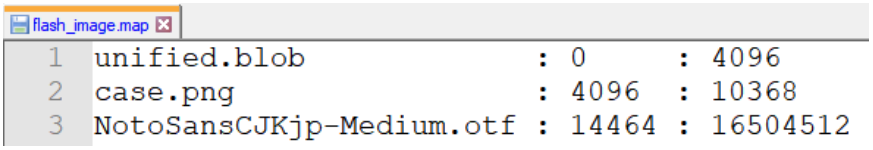


Figure 76 - Content of a .map file

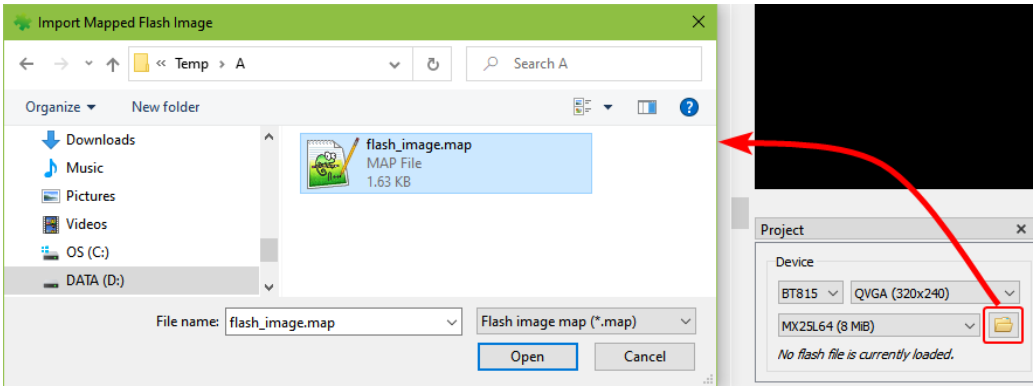


Figure 77 - Load flash file

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

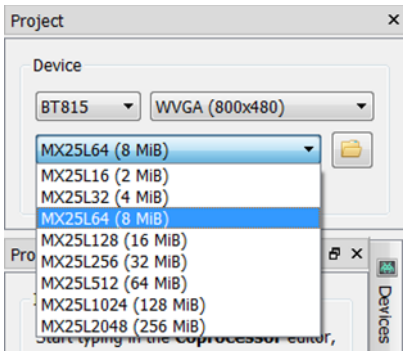


Figure 78 - Select flash size

Upon loading the flash file, its path is displayed.

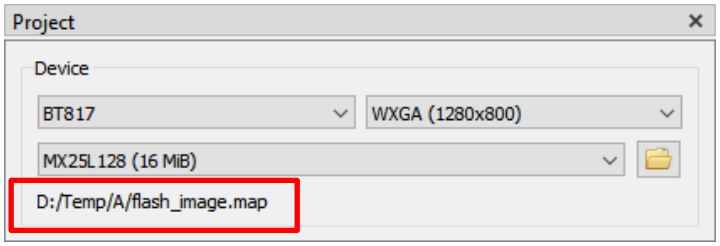


Figure 79 – Display flash path

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

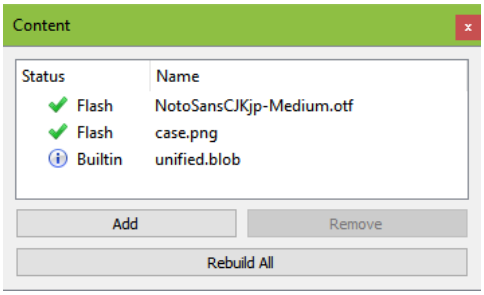


Figure 80 - Flash assets are shown in the Content window

I. Keyboard Shortcuts

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

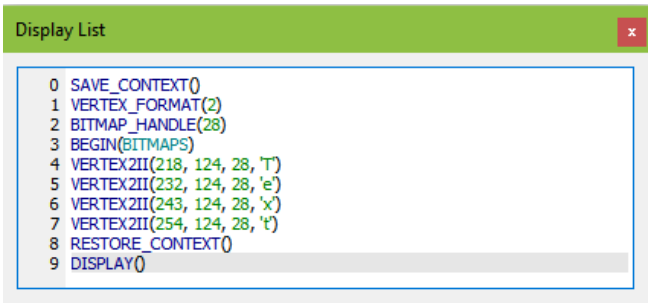
Item	Shortcut
New	*Ctrl + N*
Save	*Ctrl + S*
Undo	*Ctrl + U*
Redo	*Ctrl + Y*
Cut	*Ctrl + X*
Copy	*Ctrl + C*
Paste	*Ctrl + V*
Zoom In/Out of Viewport	*Ctrl + Mouse wheel*
Close Project	*Ctrl + F4*
Reset Emulator	*Ctrl + R*
Capture Display List	*Ctrl + D*
Open Recent Project 1->5	*Alt + 1, Alt + 2, …,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

V. 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.

A. 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.



B. 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.

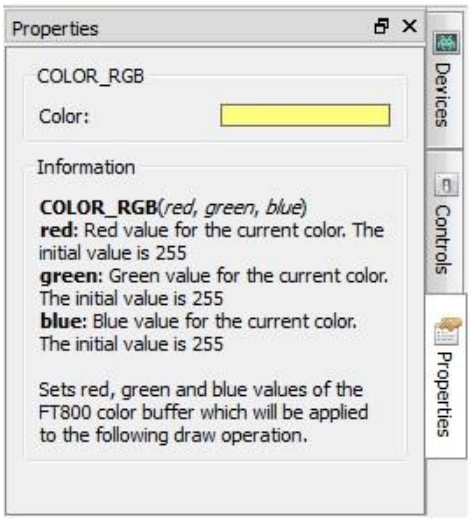


Figure 81 - Change the color

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

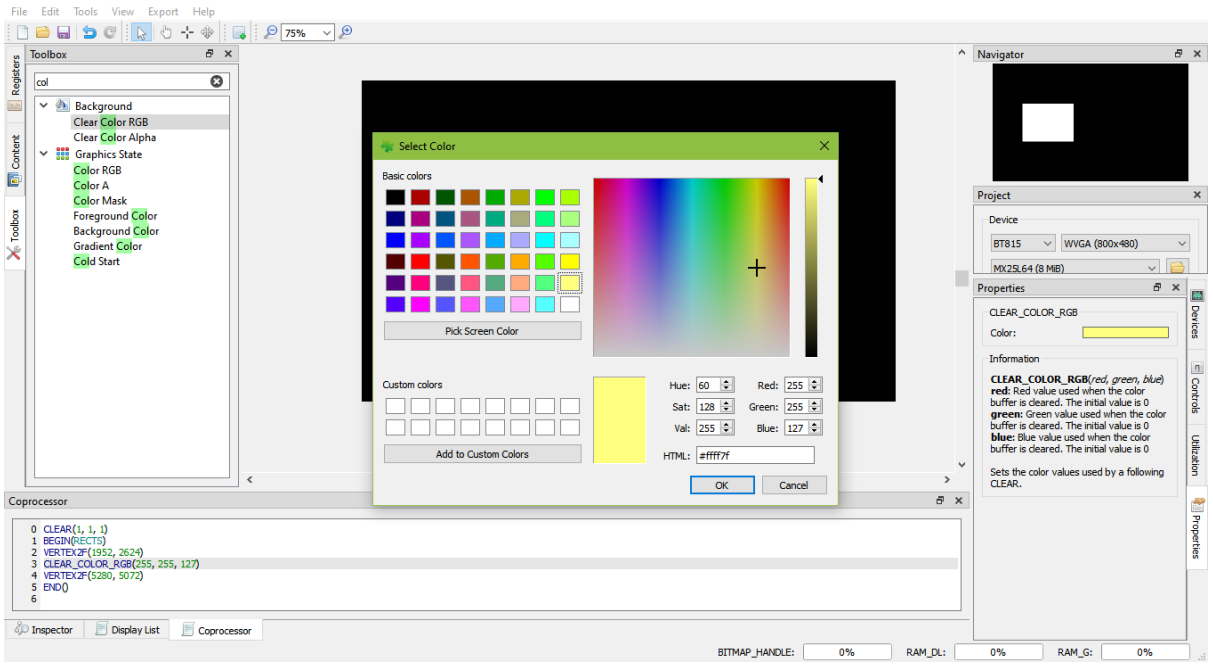


Figure 82 - Select color

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**

C. 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.

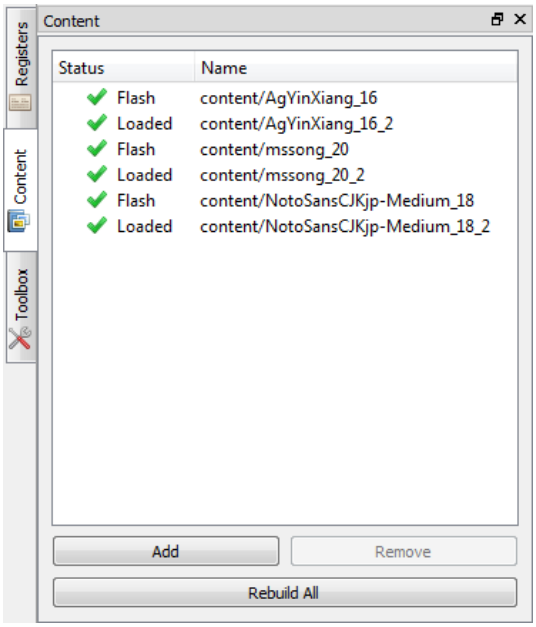


Figure 83 - Import content

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

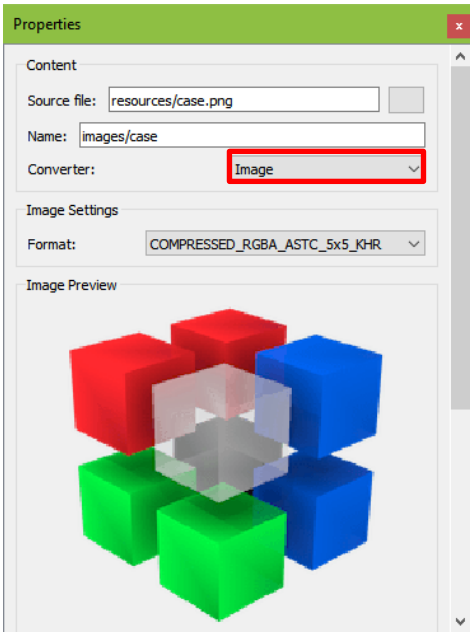


Figure 84 - Select converter image

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.

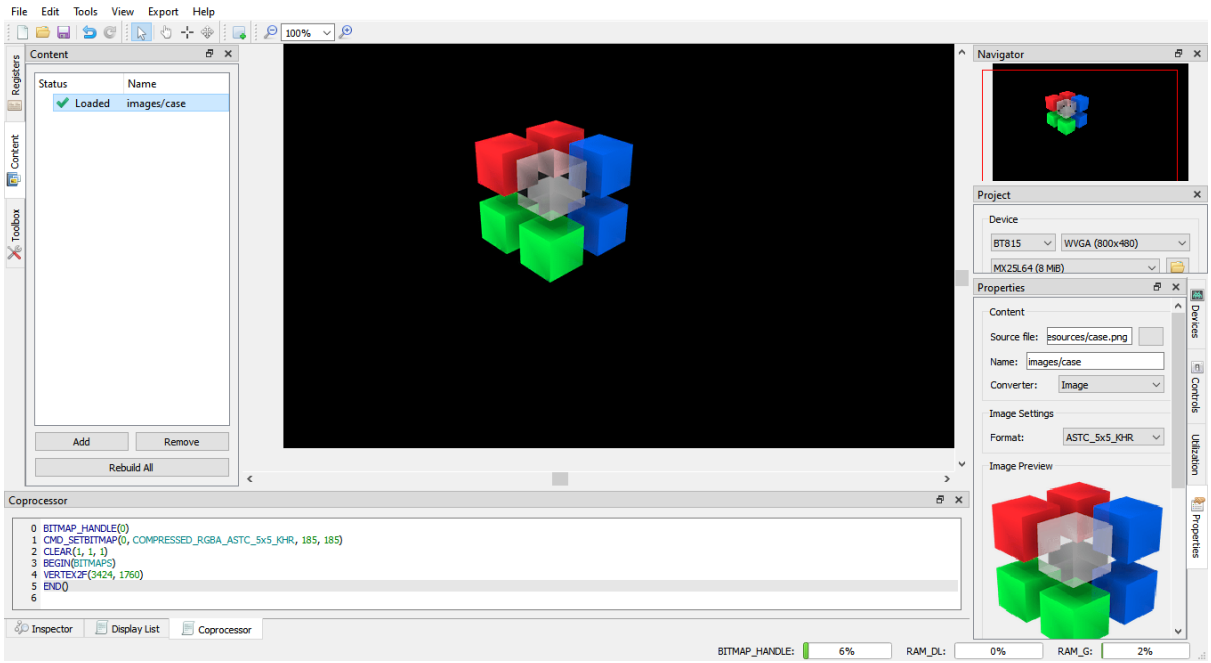


Figure 85 – Drag & drop image

D. 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.

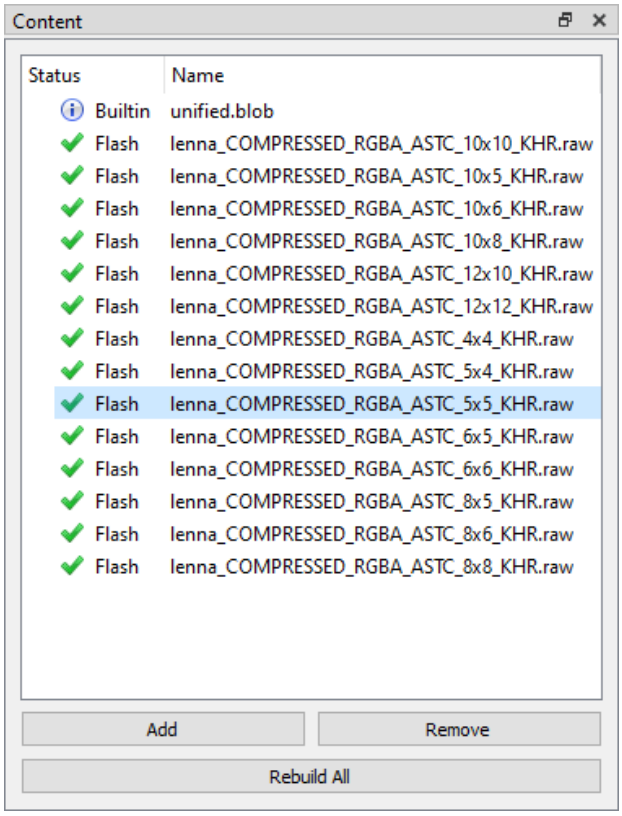


Figure 87 - Import flash

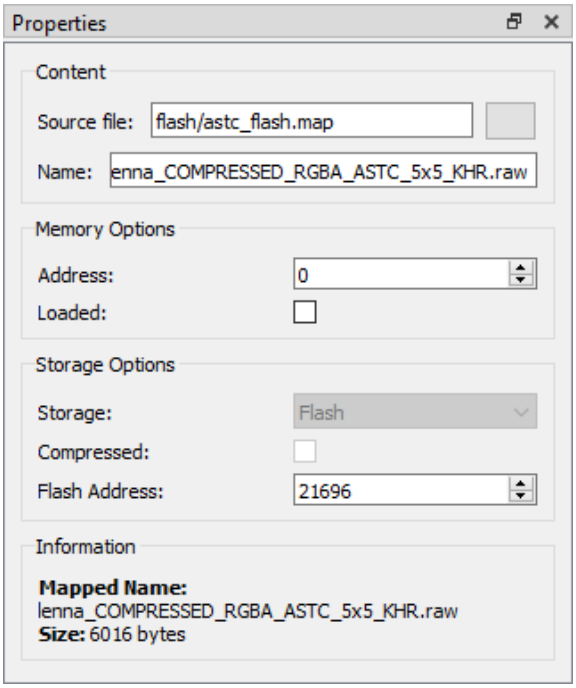


Figure 86 - Flash address

E. 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.

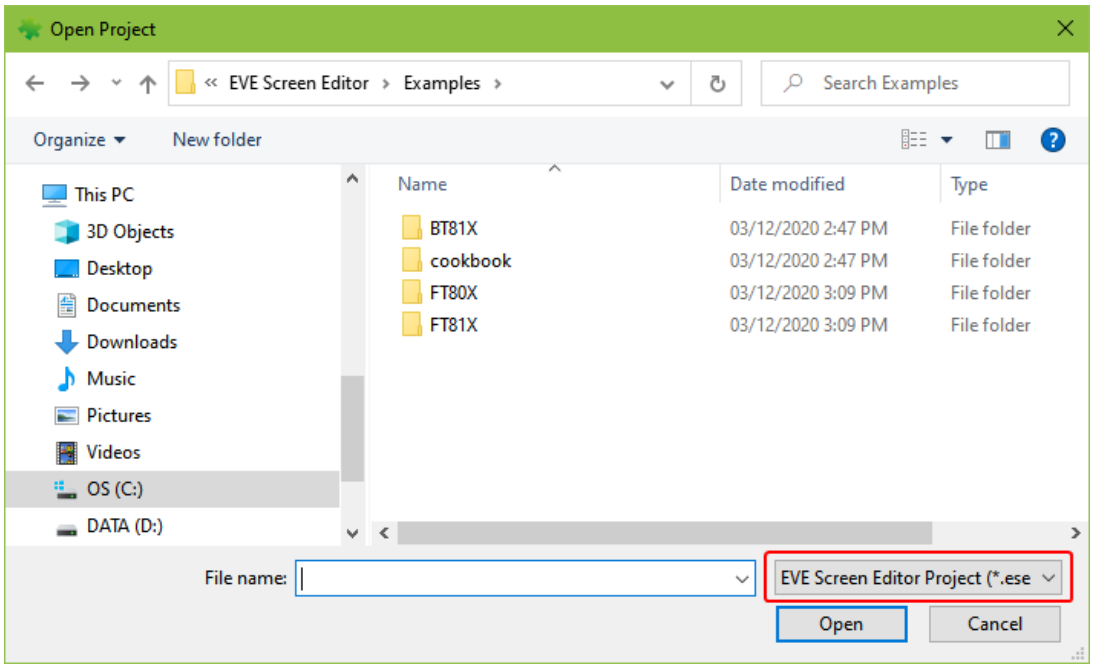


Figure 88 - Open project

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

F. 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***.

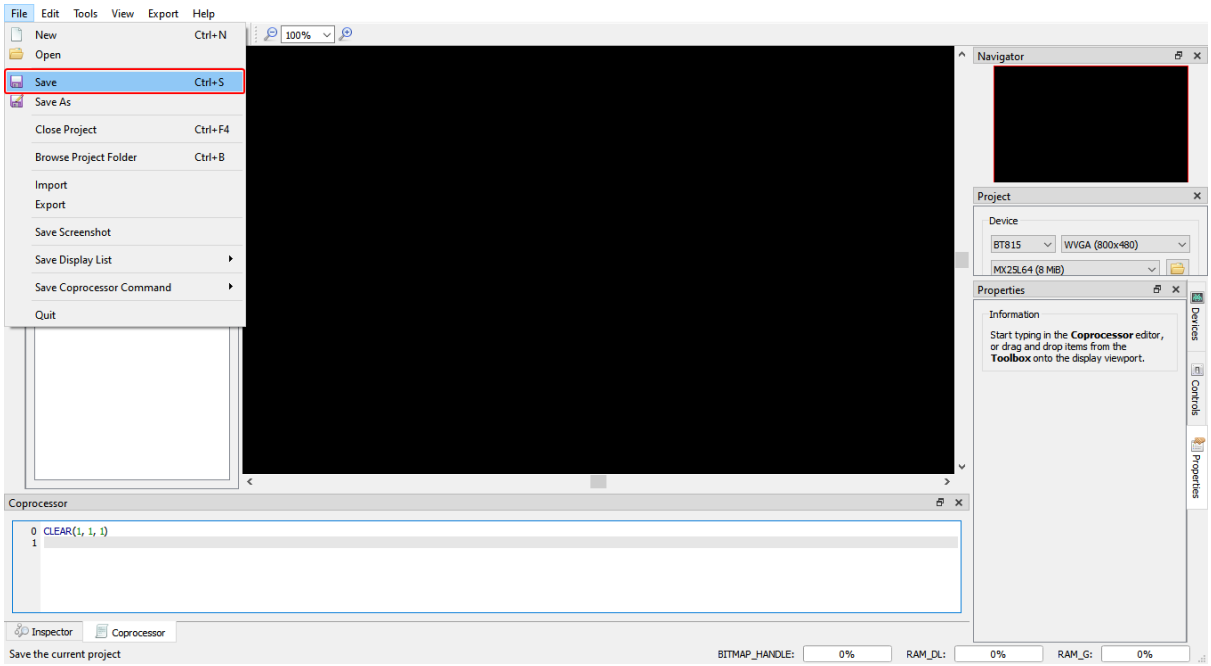


Figure 89 - Save the project

G. Export the project

Users can export their screen design in various formats such as Gameduino 2 project, EVE Arduino project, EVE HAL2.0 project (C code based), and Raspberry Pi Pico project, upon creating the screen design. Once the export process is complete, the Properties dock continues to display the project output information until the user interacts with the application again.
As the export feature involves writing files to the disk, users must ensure that they have appropriate privileges while using the EVE Screen Editor. This may require running the tool as an Administrator.
It should be noted that while the Content Manager does not enforce a strict naming convention on loaded items, during the export process, the names should be distinct and follow the variable naming convention used in the C programming language.

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.

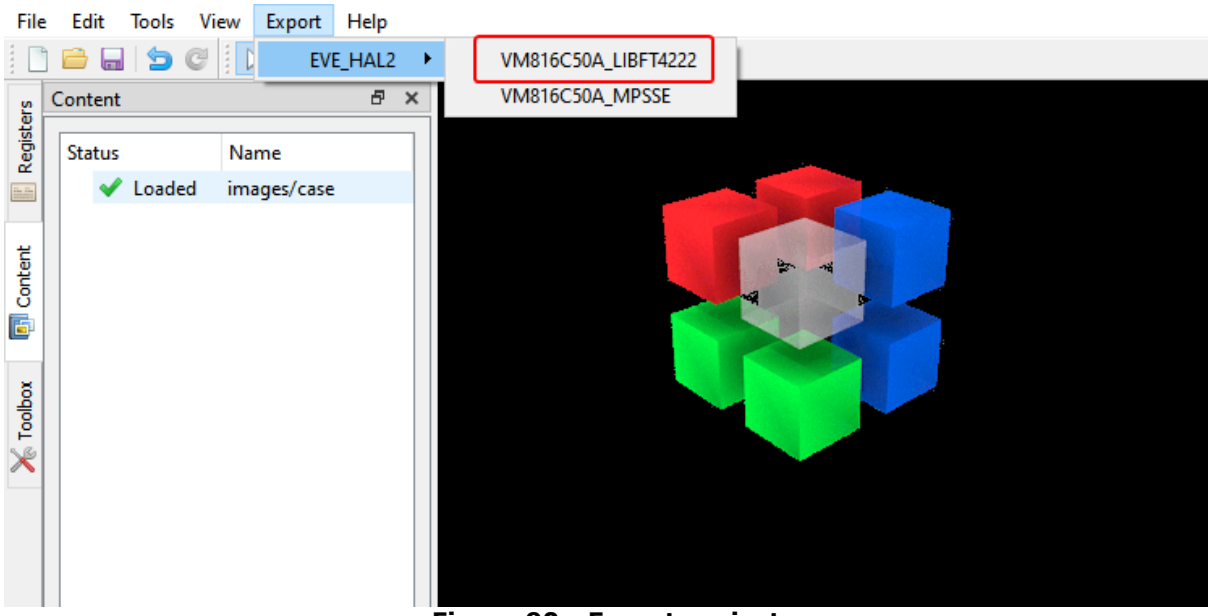


Figure 90 - Export project

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.

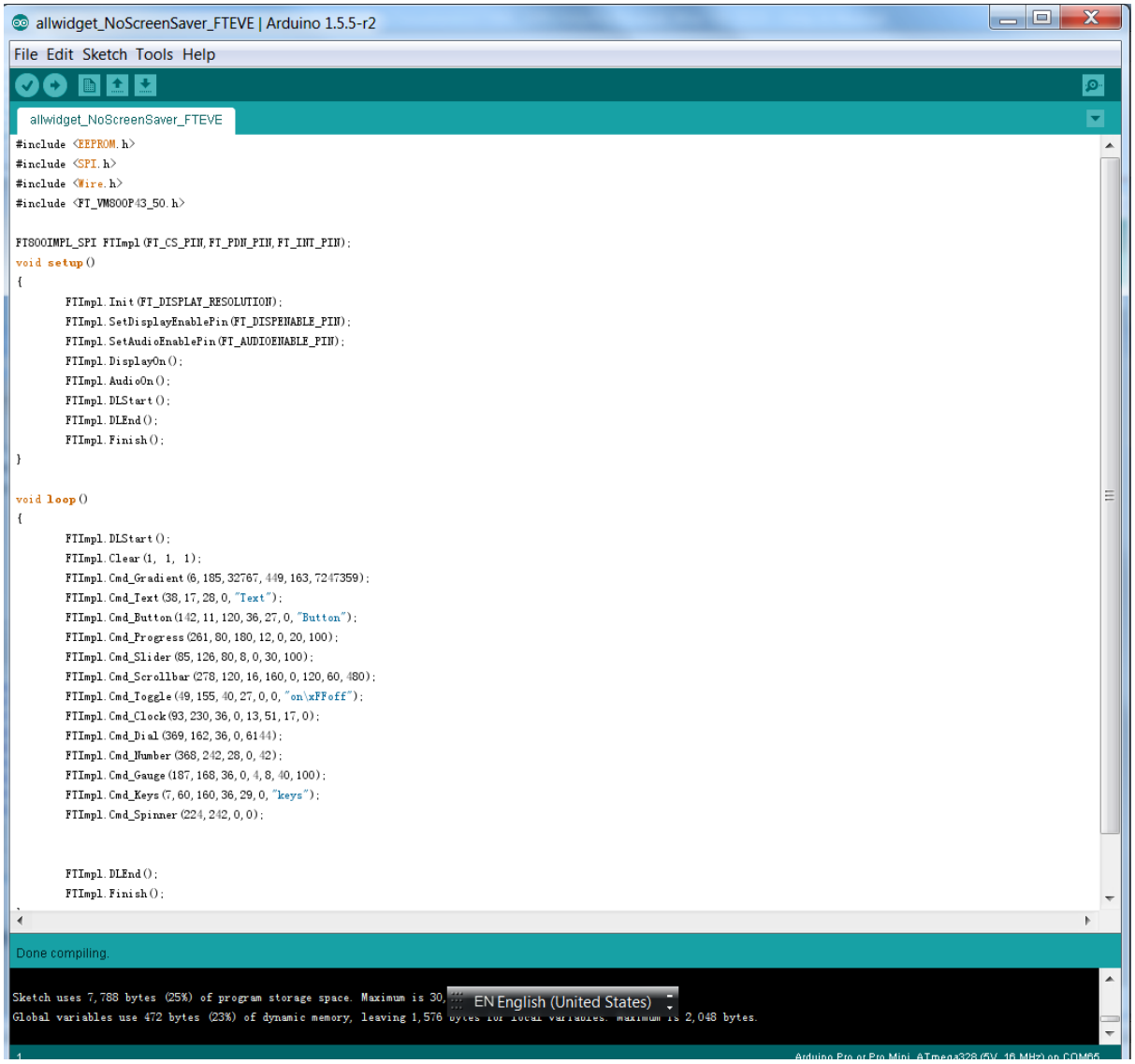


Figure 91 - Arduino IDE

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.

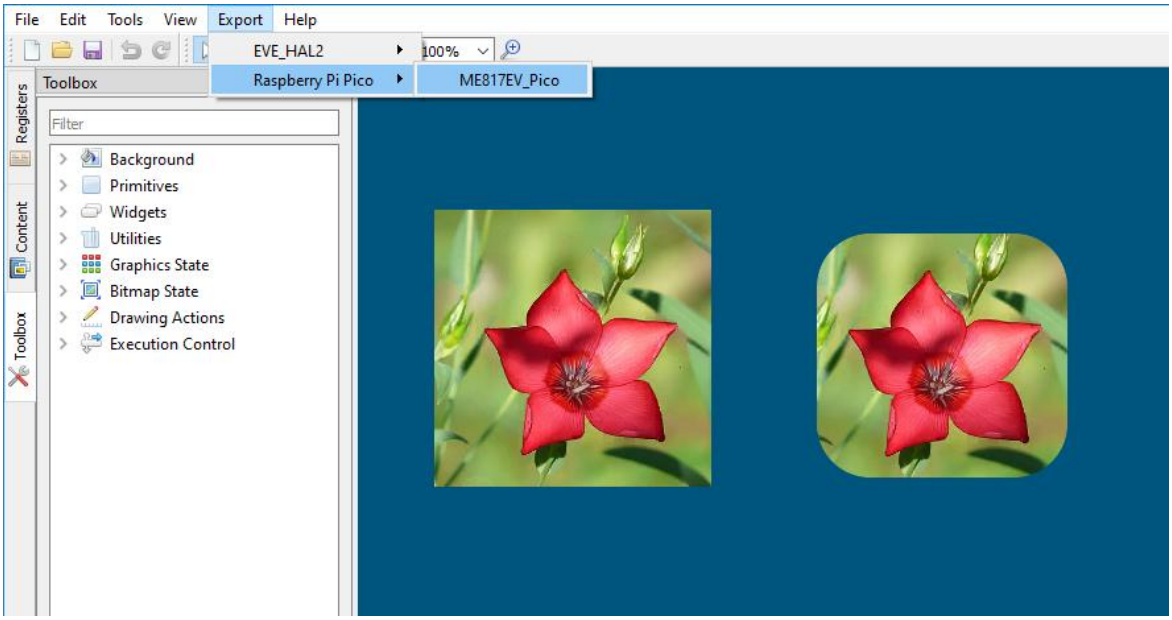


Figure 92 – Export Raspberry Pi Pico project

H. Custom Fonts

ESE supports generating EVE specific custom font by importing the widely used fonts, such as TrueType fonts (TTF) and OpenType font(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.

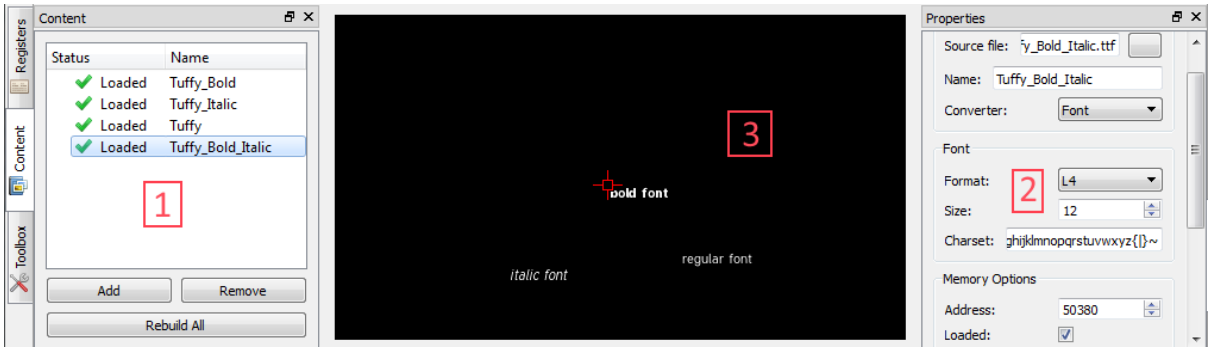


Figure 93 - Custom font

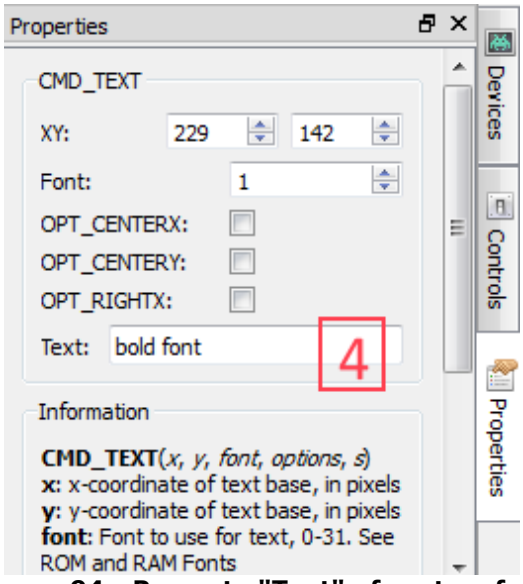


Figure 94 - Property "Text" of custom font

I. Constrain either horizontal or vertical positioning when dragging an object

1. 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

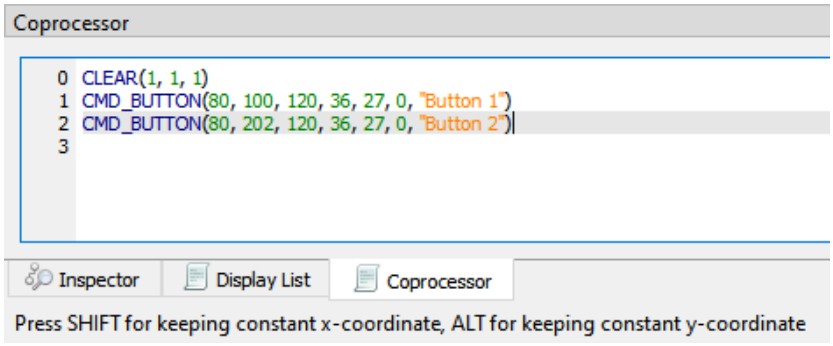


Figure 95 - Press SHIFT to constrain vertical

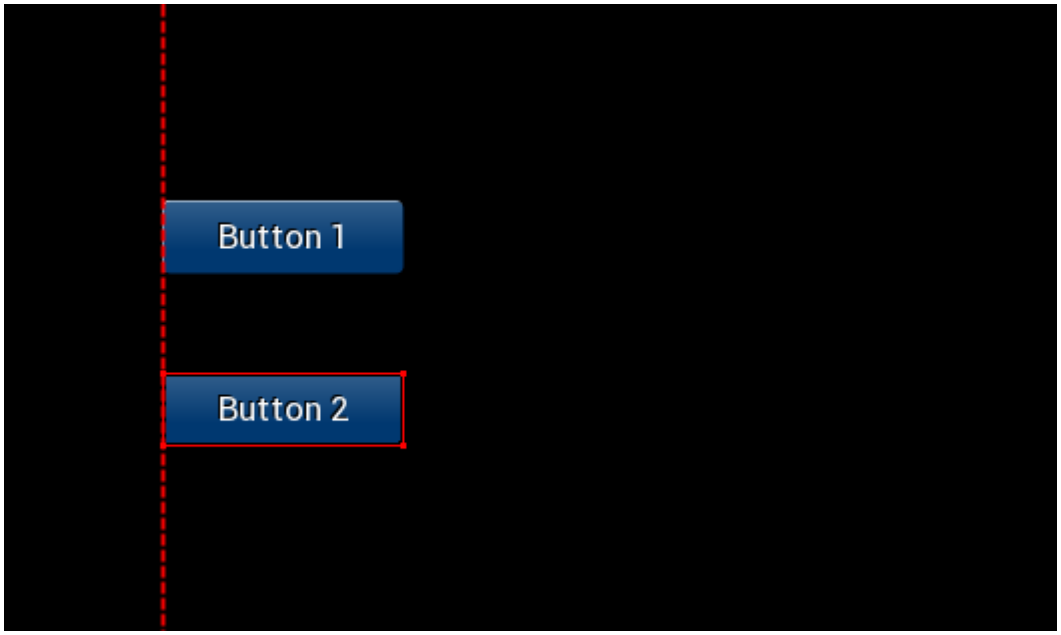


Figure 96 - Slide up/down to set y-coordinate

2. 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

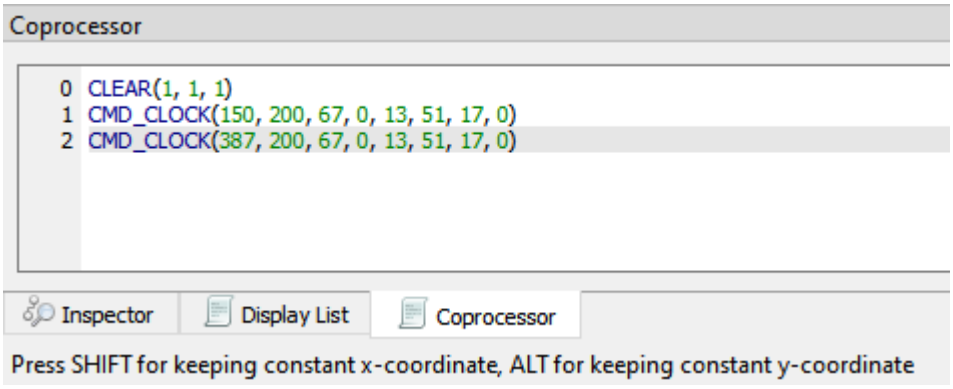


Figure 97 - Press ALT to constrain horizontal

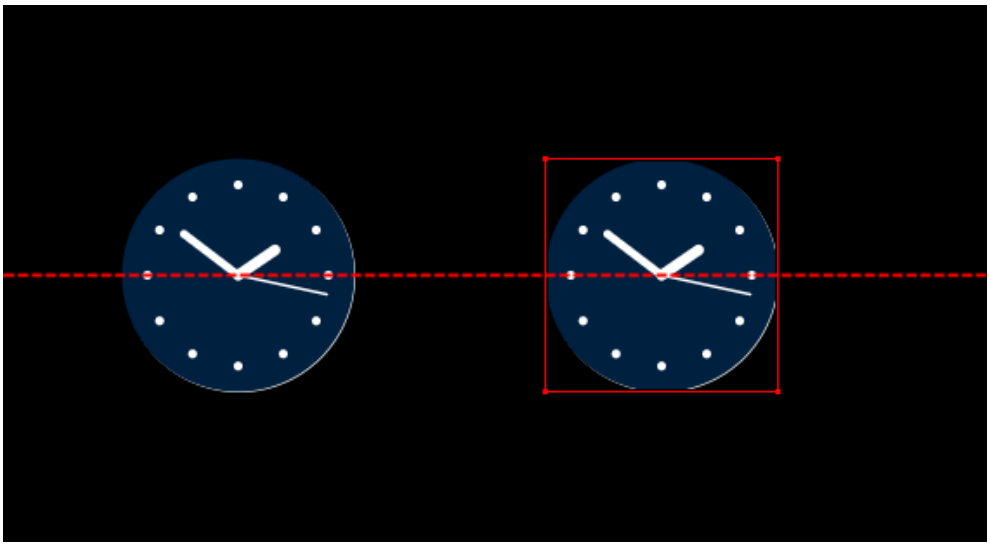


Figure 98 - Slide left/right to set x-coordinate

J. Automatically detect and load the content

When the user adds a content file that is in the list of supported files into the Content window, the app will automatically read the related file (.json/.readme) in the same folder as it to collect necessary information, then set them to Properties windows.

Supported files: .ram_g (animation), .flash (animation), .raw (image), .raw (font legacy format), .glyph (font extended format), .xfont (font extended format).

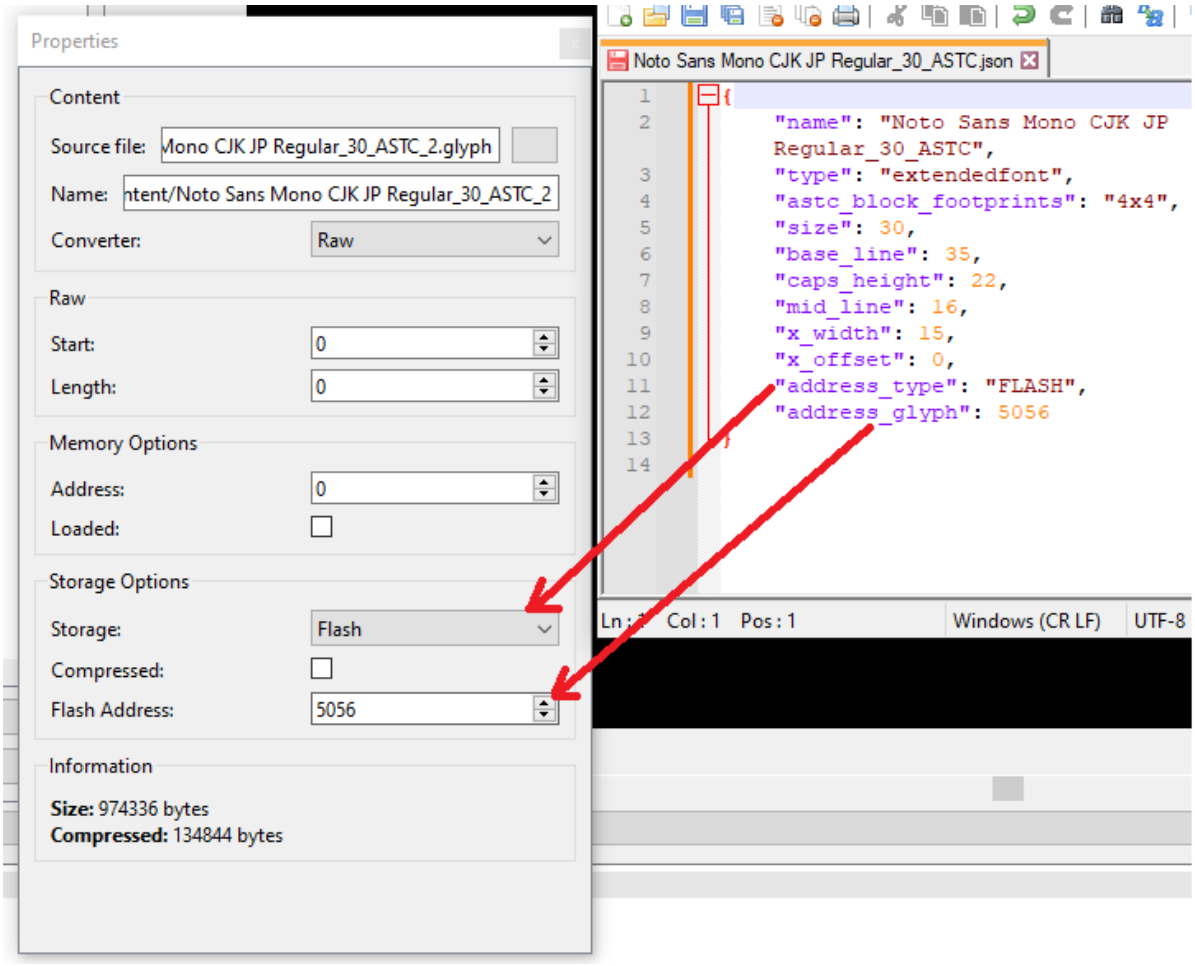


Figure 99 - Map data from the related file to Properties window

Notes:
Image with PALETTED format: The app will try to add both the LUT file and the INDEX file.
Font with Extended format: The app will try to add both the .glyph file and the .xfont file.
Therefore, please make sure that you are keeping them in the same folder or keeping the structure generated by EAB tool.

K. Generate coprocessor commands when users drop a content item

Once the user drags and drops a content item into the viewport, we extract the content item's info from JSON file, then generate related coprocessor commands to render them. Support types: .avi (video), .flash (animation), .ram_g (animation), .raw (image), .raw (font legacy format), .xfont (font extended format).

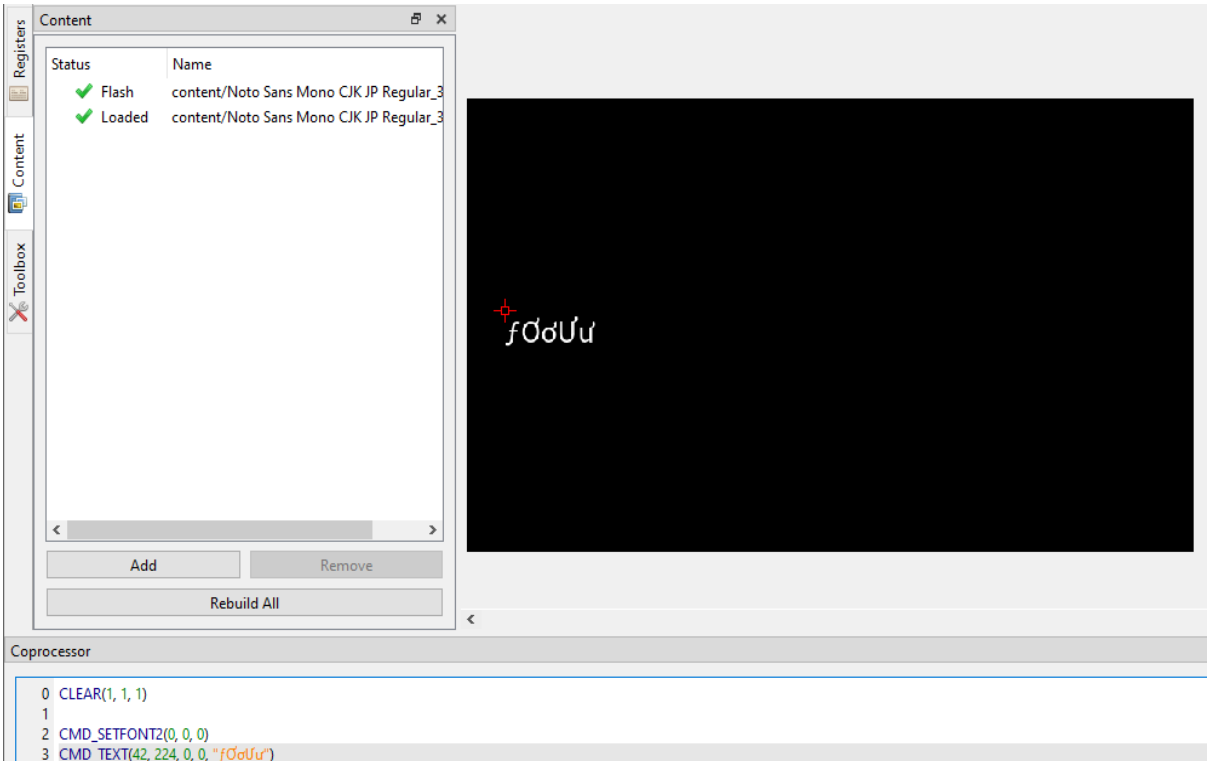


Figure 100 - Automatically generate coprocessor commands

L. Overlay extra lines in viewport to assist the alignment of graphics object.

Display alignment lines to help user align the item when its position is close to the vertical or horizontal edges of other items.

Green: Vertical match
Cyan: Horizontal match
Red: Nearly vertically
Dark yellow: Nearly horizontally

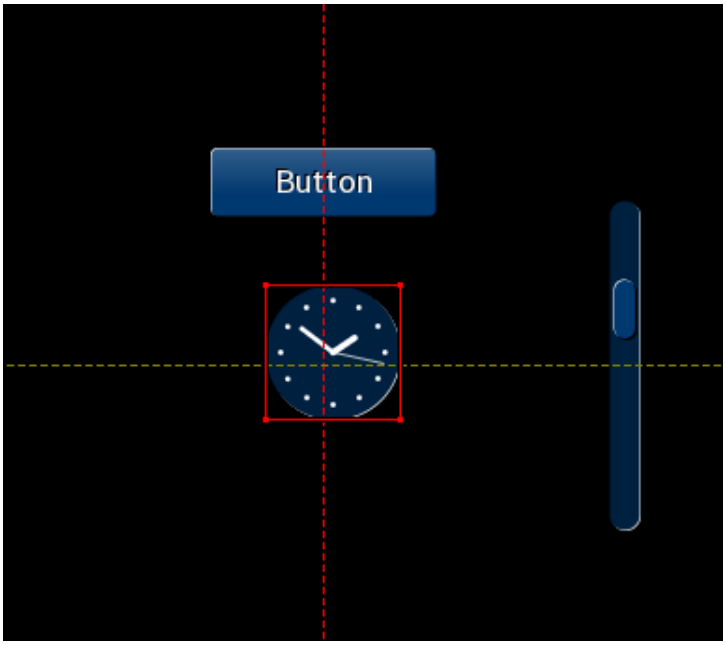


Figure 101 - Nearly vertically and nearly horizontally

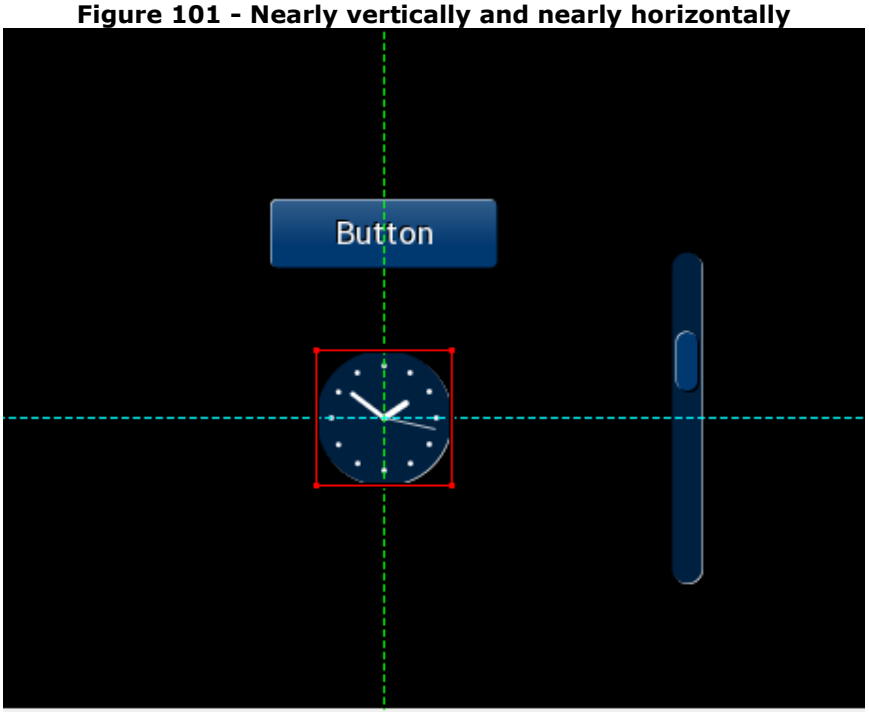


Figure 102 - Vertical match and horizontal match

M. 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* -> *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

VI. 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.

Constant	Value
RAM_G	The start addresses of RAM_G: 0
RAM_DL	The start addresses of RAM_DL: Compiled for each device

Table 3 - Predefined constants

A. 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.

B. 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

C. 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.

D. 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.

E. 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.

F. 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.

G. 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.

H. 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()
```

I. 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)
```

J. 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.

K. 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.

L. 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
firstchar - 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")
```

M. 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: 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)
```

VII. Working with EVE Screen Editor

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

A. 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.

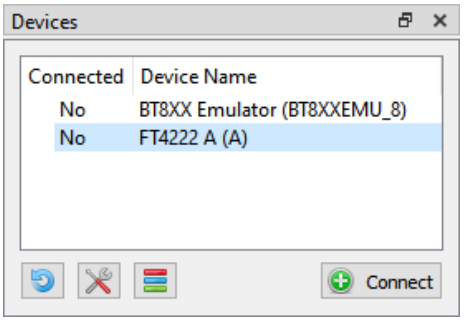


Figure 103 - Connect Device

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

Note 1: USB Hub may 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.

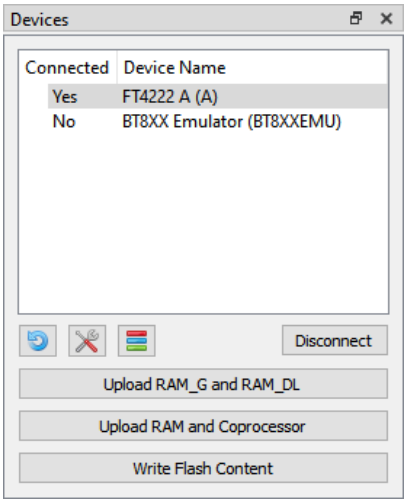


Figure 104 - Device connected

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:

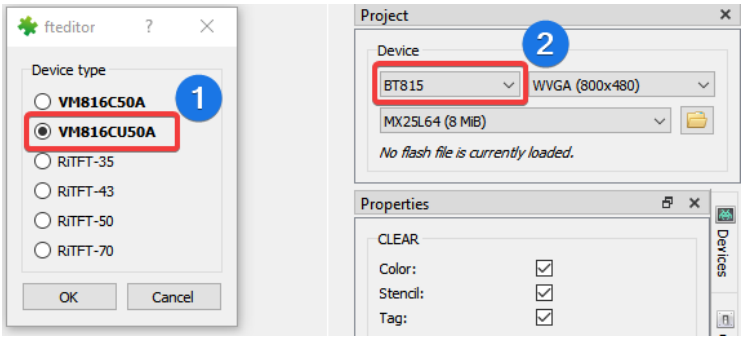


Figure 105 - Select the correct device type

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.

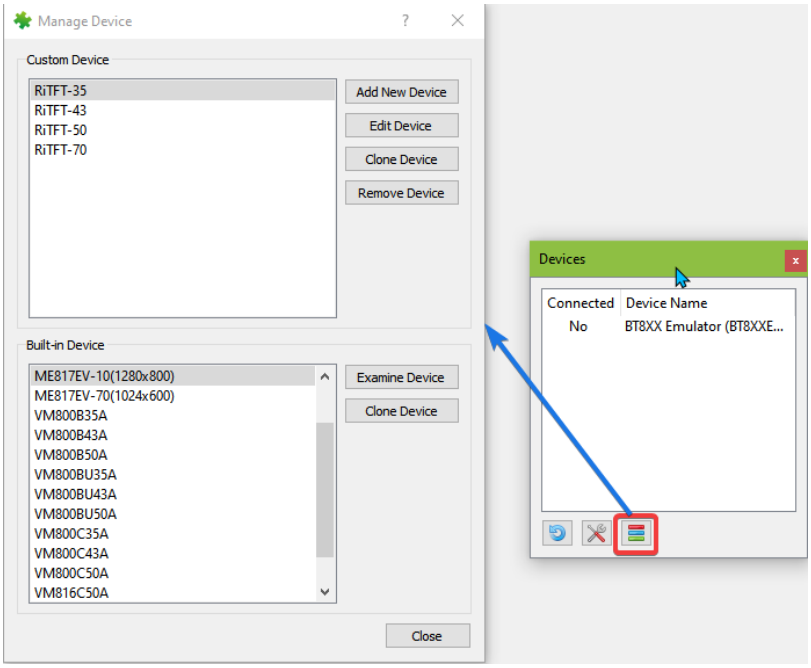


Figure 106 - Managing device

B. Check your design

This section explains how to validate your design.

1. 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.

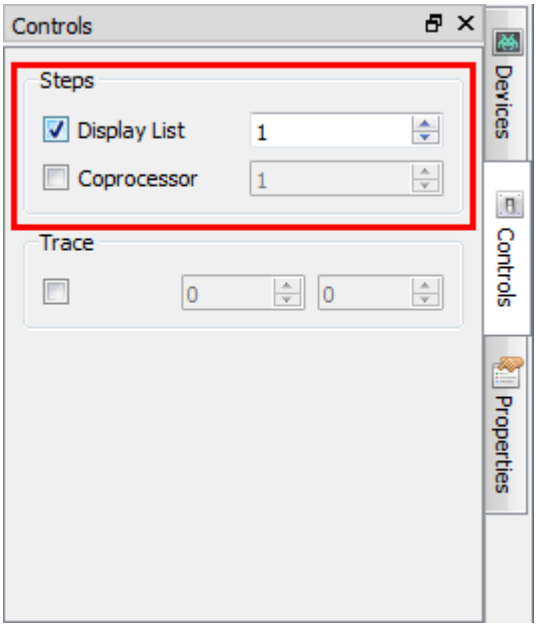


Figure 107 - Select Display List/Coprocessor

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.

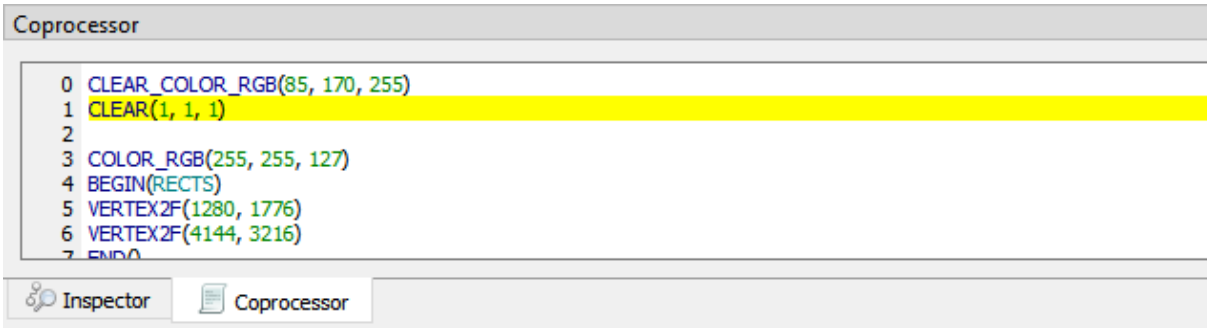


Figure 108 - Display List command is highlighted in yellow

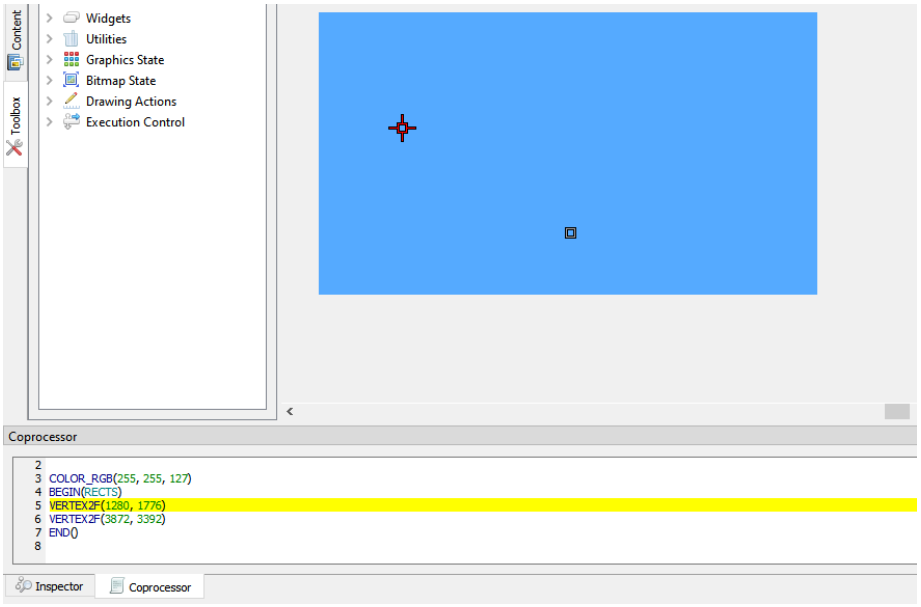


Figure 109 - Prepare to draw a rectangle

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:

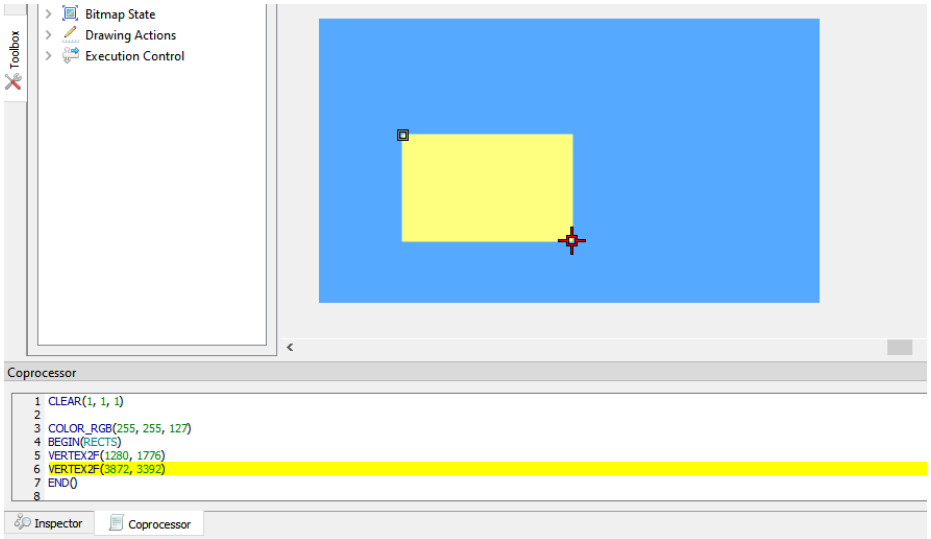


Figure 110 – Draw rectangle

2. 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.

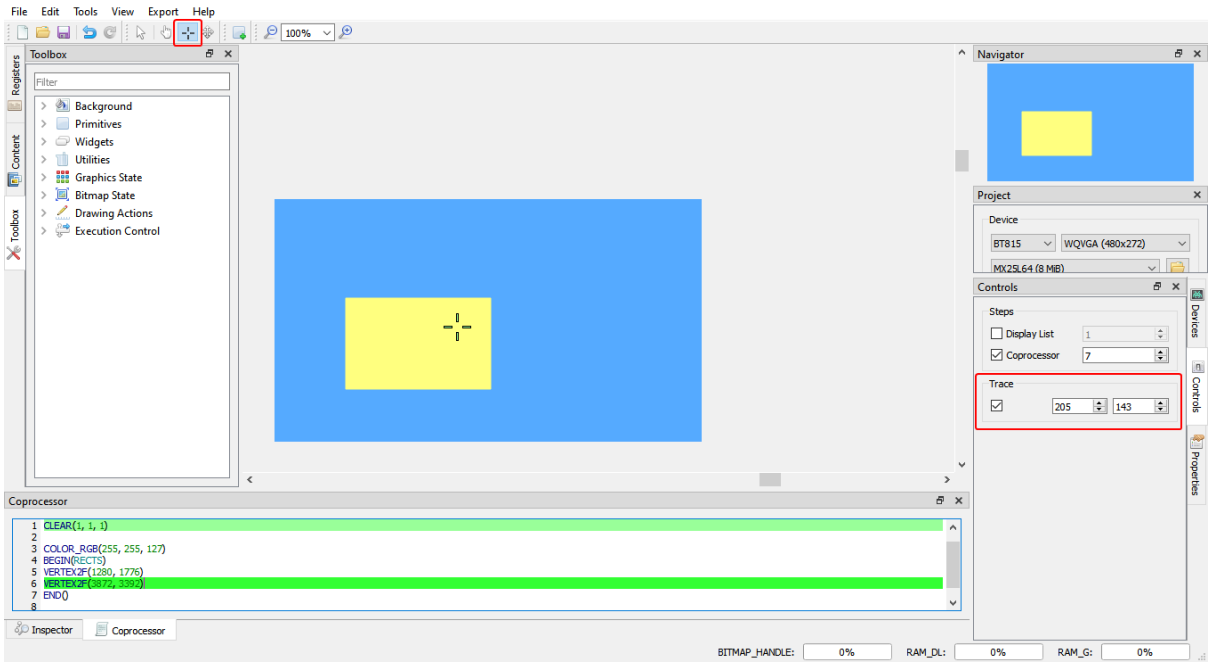


Figure 111 - Trace the pixel

C. 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 four sub-folders *"BT81X", "FT81X", "cookbook" and "FT80X"* for specific example projects:

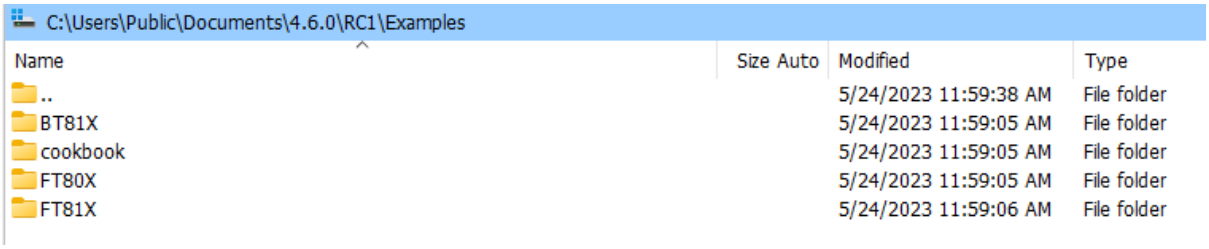


Figure 112 - Examples

Open "allwidget_NoScreenSaver" project under FT80X folder and it shows:

Figure 113 - Open example project

D. Add software library for exporting projects

You may extend **ESE** by adding one python file to support your own software library for your hardware platform.

To incorporate a custom library into the Export menu, simply place the files of your module with the module name in the following directory within the installation directory: "**/export_scripts/<device_type>/<library_name>/<module_name>.py"**
Currently, the existing modules use "exportModuleName" as a reference to call the corresponding Python files in the installation directory for exporting projects. Users can use this as a basis to customize their own needs.

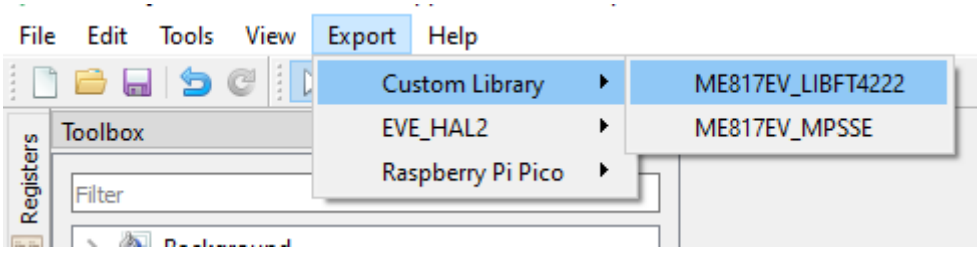


Figure 114 - Export projects from own software library

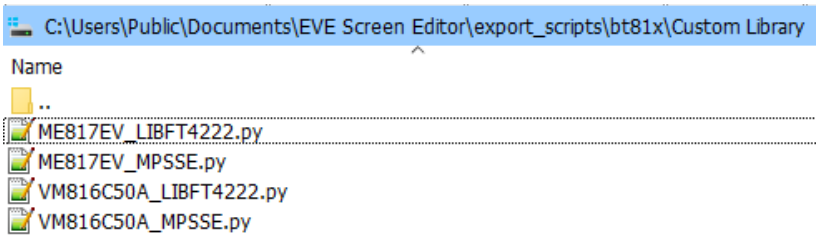


Figure 115 - Add module name files to the installation directory

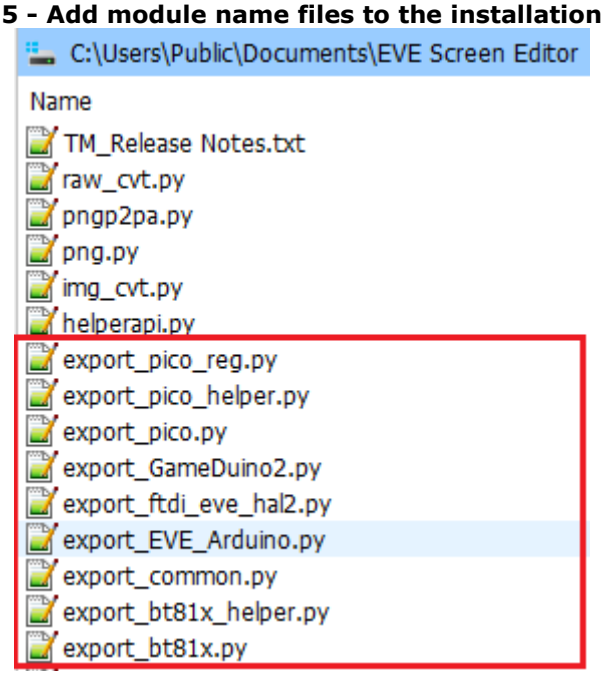


Figure 116 - Export scripts

Note 1: If the device type is a generic name, the modules will be displayed to all models of that type (**bt88x**: bt880, bt881, bt882, bt883). Therefore, please use the device type as the chipset name when using a specific chipset.

Note 2: Please keep the structure of the module name files the same as the existing library. With BT815 and later, the **run** function needs 4 arguments, other devices it needs 3 arguments. With **supportedPlatforms** variable, please convert the number of your target device from hexadecimal to decimal to get the number you need (Example: BT817: Hex: 817 -> Dec: 2071).

E. Working with EVE Asset Builder

When a user adds a font or flash asset generated by EVE Asset Builder (EAB) to the Content window, ESE will automatically gather the necessary information from the corresponding information file. This information facilitates the modification of values in the Properties window and the generation of commands when the user places the content item into the viewport.

With .raw and .glyph files, ESE will read .json file.

With .ram_g and .flash files, ESE will read .readme file.

VIII. Disclaimer

This section contains the license agreements of ESE. 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.

IX.Contact Information

Head Quarters – Singapore

Bridgetek Pte Ltd
1 Tai Seng Avenue, Tower A #03-05
Singapore 536464
Tel: +65 6547 4827
Fax: +65 6841 6071

E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Branch Office – Taipei, Taiwan

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
E-mail (Support) support.apac@brtchip.com

Branch Office - Glasgow, United Kingdom

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

E-mail (Sales) sales.emea@brtichip.com
E-mail (Support) support.emea@brtchip.com

Branch Office – Vietnam

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.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Web Site

http://brtchip.com/

Distributor and Sales Representatives

Please visit the Sales Network page of the Bridgetek Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A - References

Document References

FT81x Series Programmers Guide

FT81x Datasheet

FT800 Series Programmers Guide

FT800 Embedded Video Engine Datasheet

BT81x Datasheet

BT81X Series Programmer Guide

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
EAB	EVE Asset Builder

Appendix B – List of Figures & Tables

List of Figures

List of Tables