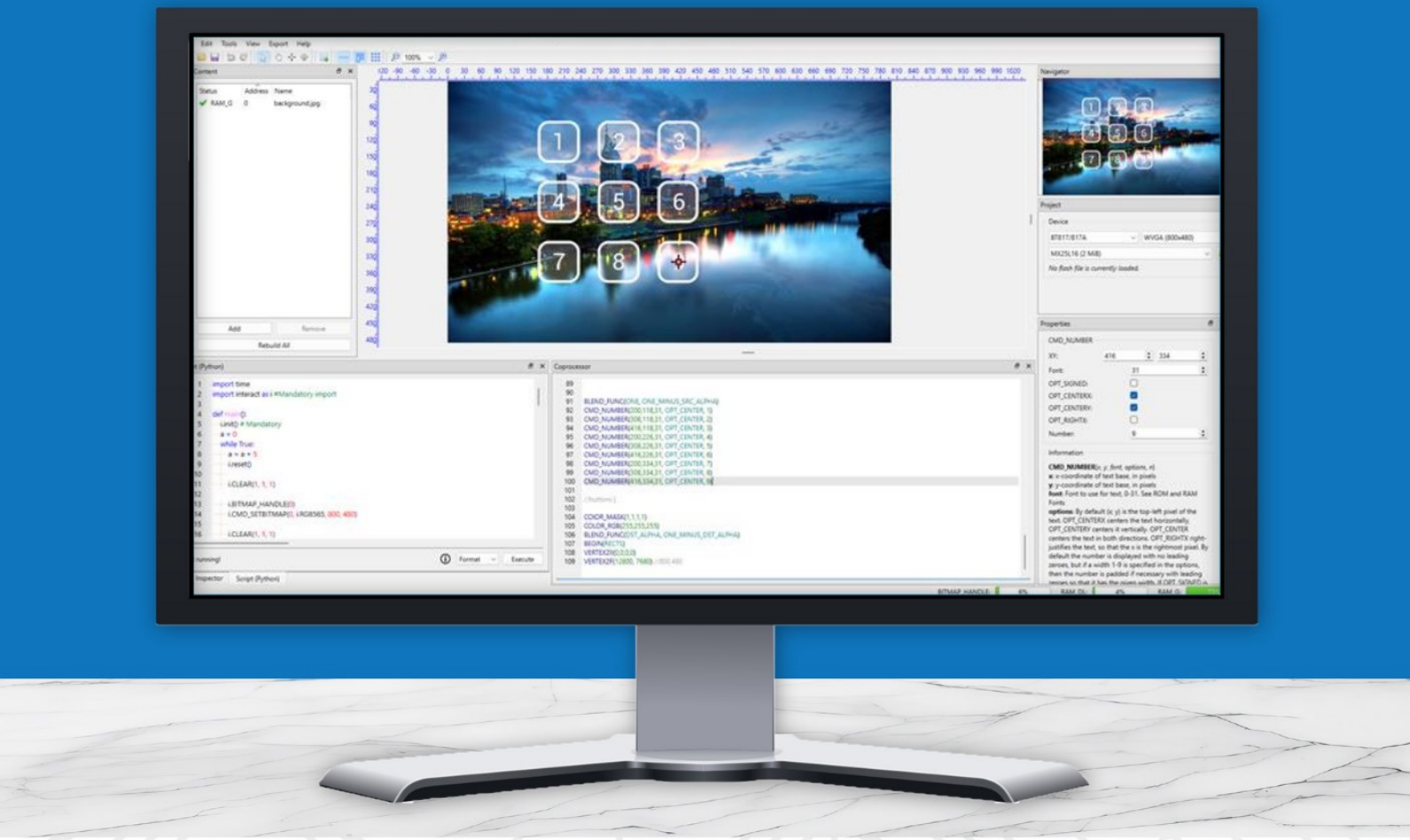
Bridgetek
BRIDGING TECHNOLOGY

EVE SCREEN EDITOR 5.0

USER GUIDE



DOC. VER. 2.2
ISSUE DATE: 21 MARCH 2025

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 Programming Guide.pdf	Programming Guide	PDF
BT82X Series Programming Guide	Programming Guide	PDF
BT82X Datasheet	Datasheet	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 board with a cable (FT4222 or MPSSE), 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, BT81X and BT82X. 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), IDM2040-43A(480x272)*
BT815/6	EVE Hal Projects	*VM816C50A, VM816CU50A*
BT817/8	EVE Hal Projects	*ME817EV, IDM2040-7A*

BT82X	EVE Hal Projects	*VM820C*

❖ 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
FT4222, MPSSE	VM820C

C. Key Features

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

- **WYSIWYG** GUI
- Support the editing, executing of display list, coprocessor commands
- Widget-based GUI construction
- Drag and drop to create single screen design
- Import EVE specific assets into the project
- Export project into source code, C or Python
- Sync up the screen design from PC to the connected device
- Integrated tools for asset conversion and loading
- Inspect the internal registers and memory

D. What's new in ESE 5.0?

- Add the support of BT820 and BT820 based module VM820C.
- Add CMD_NOP in ESE.
- Support dot character in the expression of coprocessor editor.
- Add ECG-Signals example project to showcase the usage of bargraph bitmap.

E. Resolved issues

- CMD_NOP is not shown on Toolbox.
- The Toolbox window is not displaying the correct items for the Display List editor.
- The application encountered a crash while attempting to open the most recent project via the Welcome screen.
- Error when displaying an image in ASTC format.
- The line stride and size are not correct on PALETTEDARGB8 format of BT820.
- LA1/LA2/LA4, PALETTEDARGB8 are not rendered correctly.
- Opening a project causes a crash.
- Unable to display an image in the PALETTED format on the FT800.
- The ARGB1555 format appears duplicated in the image format on the FT800.
- DXT1 formats are regenerating the commands incorrectly.

- In the Devices window, clicking "Write to RAM_G and RAM_CMD" does not render the screen on the BT8XX emulator unless "Write to RAM_G and RAM_DL" is clicked first.

F. Known Issues & Limitations

The following are some of the known issues and limitations:

1. **CMD_PLAYVIDEO** is unsupported in device sync up feature.
2. When the **REG_ROTATE** is changed, the emulator does not allow for item selection on the newly rotated screen. Instead, users must select items from the previous position.

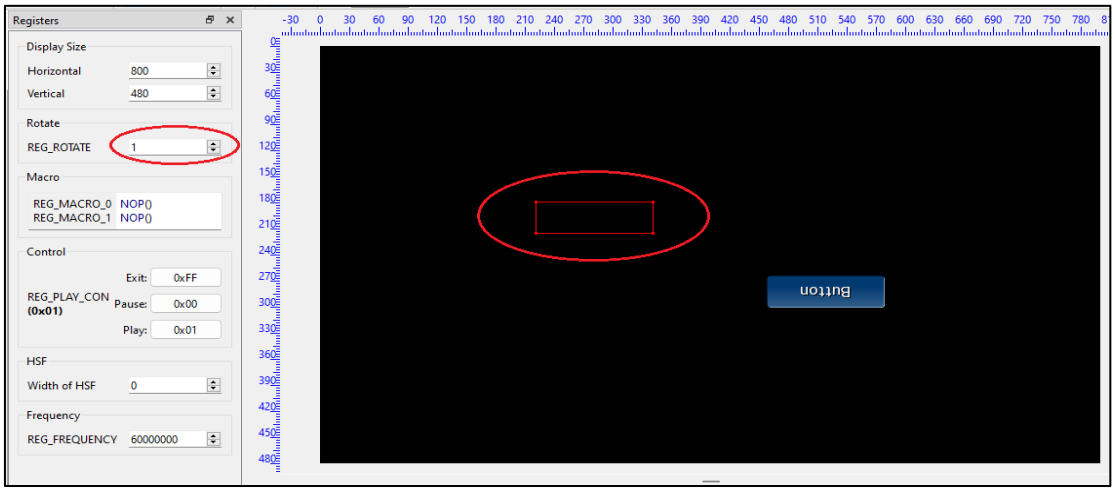


Figure 1 - Limitation for item selection on the rotated screen

3. The register **REG_RE_ROTATE** of BT82X is unsupported yet.

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.
- QScintilla: https://qscintilla.com/

2. Icons Copyright

ESE utilizes certain icons sourced from http://p.yusukekamiyamane.com/, in adherence to the Creative Commons Attribution 3.0 License.

H. System Requirements

To install ESE 5.0 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 using 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

I. Hardware Requirements

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

Board	Host Platform
VM820C	FT4222, MPSSE
ME817EV-WH70C	FT4222, MPSSE
ME817EV-WH10C	FT4222, MPSSE
VM816CU50A	FT4222
VM816C50A	FT4222, MPSSE
ME813AU-WH50C	FT4222
ME812A-WH50R	FT4222, MPSSE
VM810C50A	FT4222, MPSSE
VM800B	MPSSE

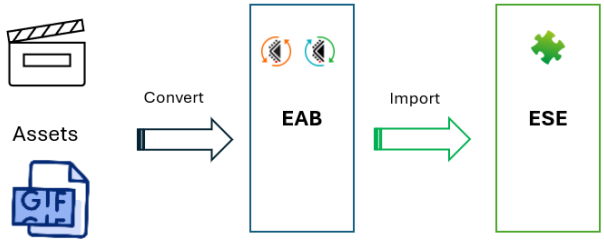
J. Dependencies / Pre-Requisites

- **The video and animation assets**

  ESE expects the video assets (.avi file) in MJPEG format conform to the EVE compatible format. Users shall make use of the video converter of **EAB** to ensure it.

  ESE expects the animation assets conform to the EVE compatible format. Users shall make use of the animation converter of **EAB** to ensure that.

  The following workflow is expected:



- **The assets name requirement**

  To avoid the compilation issue in the exported C code, the name of assets shall contain the ASCII code only.

- **The project file path**

  ESE project file path shall not be more than 256 characters to avoid the potential error when it is exported.

K. 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/toolchains/

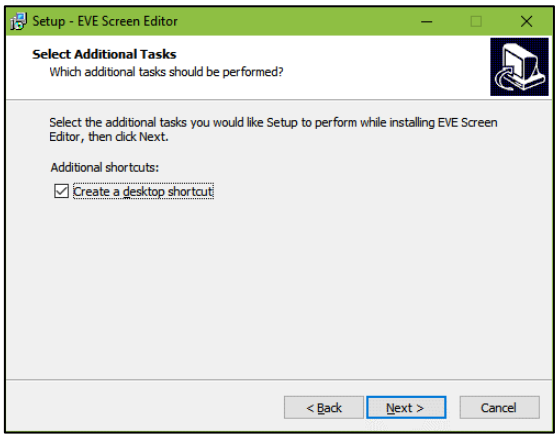
ii. When prompted by a download dialog box, click **[Save]**.

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

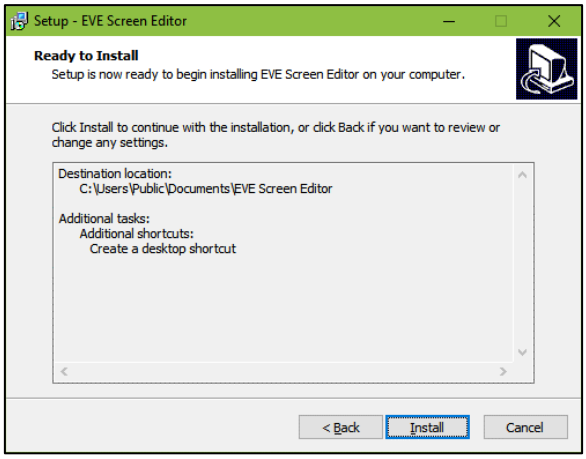
iv. Extract the zip file contents. Double-click the executable file.

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.

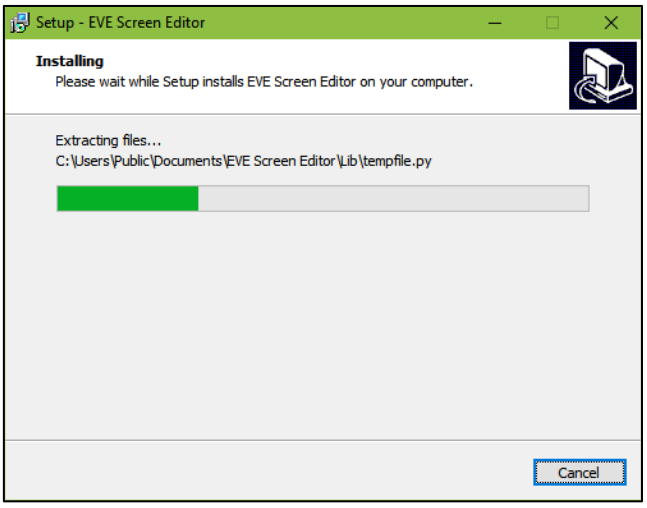
vi. In the **Select Additional Tasks** window, check the **"Create a desktop shortcut"** checkbox to create the ESE 5.0 icon on the desktop if required. Click **[Next]** to prepare for the installation.



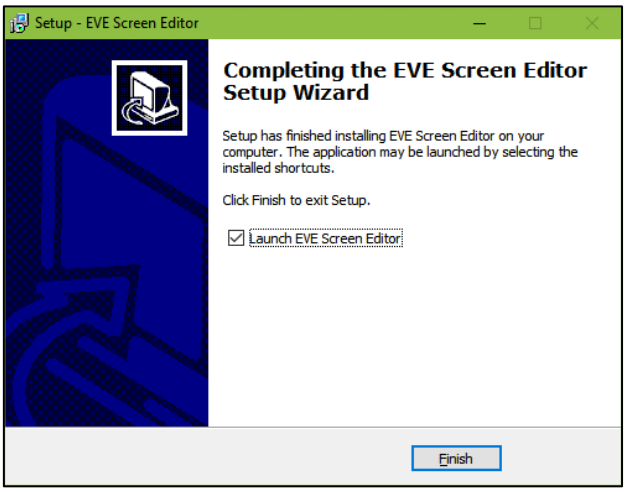
vii. Once the initial setup is completed, this screen informs the application is ready to be installed.



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.



L. 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	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
Export	Scripts and templates support the export feature	Read-Only
untitled	The template project used by export feature	Read-Only
device_sync	Information of built-in and custom boards	Read/Write

firmware	Flash **BLOB** for BT815/6, BT817/8 and BT820	Read-Only
iconengines	Qt DLL for supporting icon	Read-Only
__pycache__	A folder which is created by Python interpreter at run-time to store the compiled byte code, avoiding recompilation of source file during execution.	Read-Only

Table 1 - Installation Folder

III. The Graphical User Interface

A. Overview

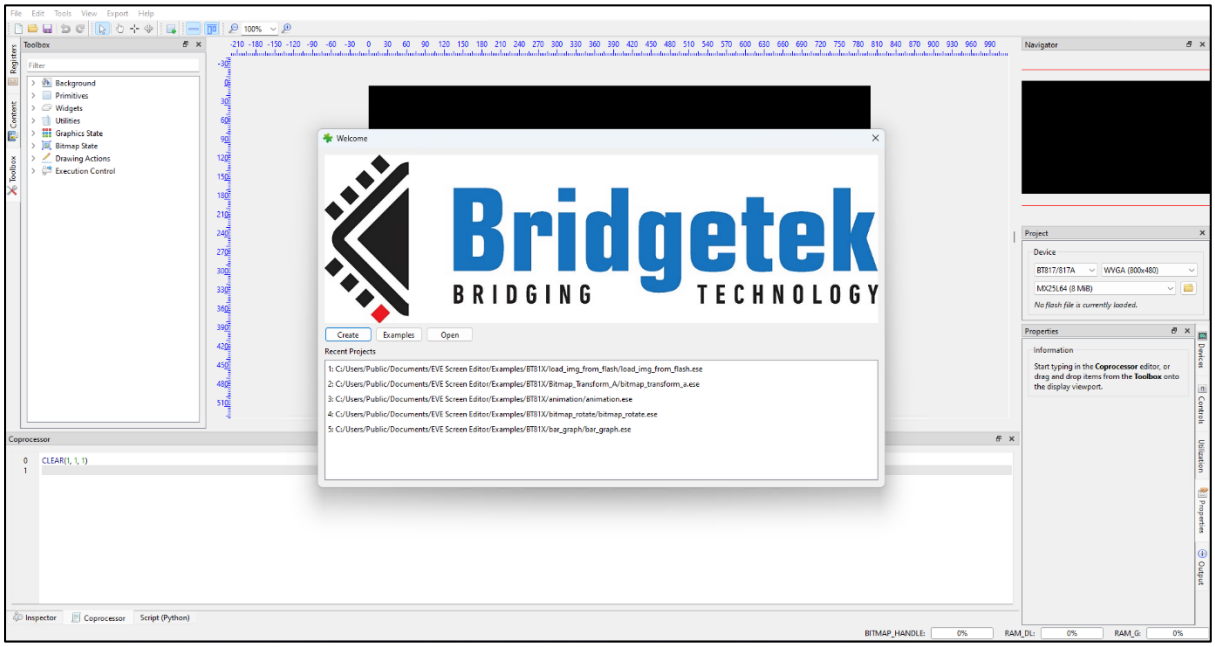


Figure 2 - Welcome Screen

ESE user interface has the following components:

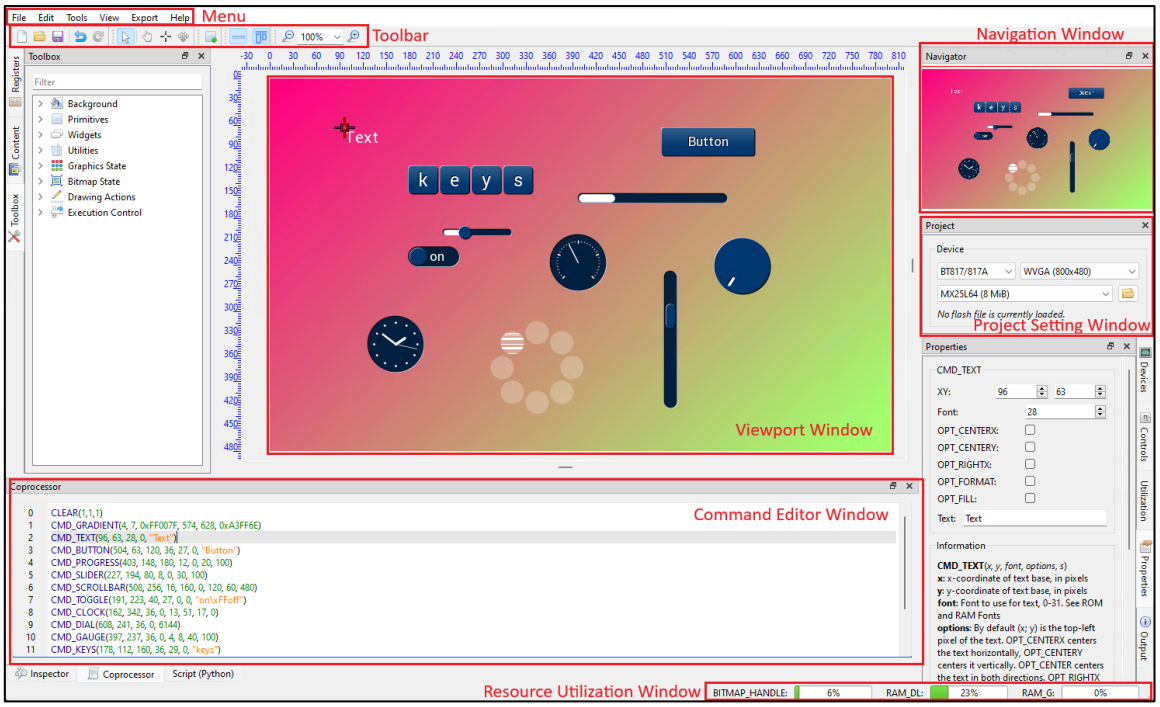


Figure 3 - User Interface Components

B. Menu bar, Toolbar, and Status bar

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

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.

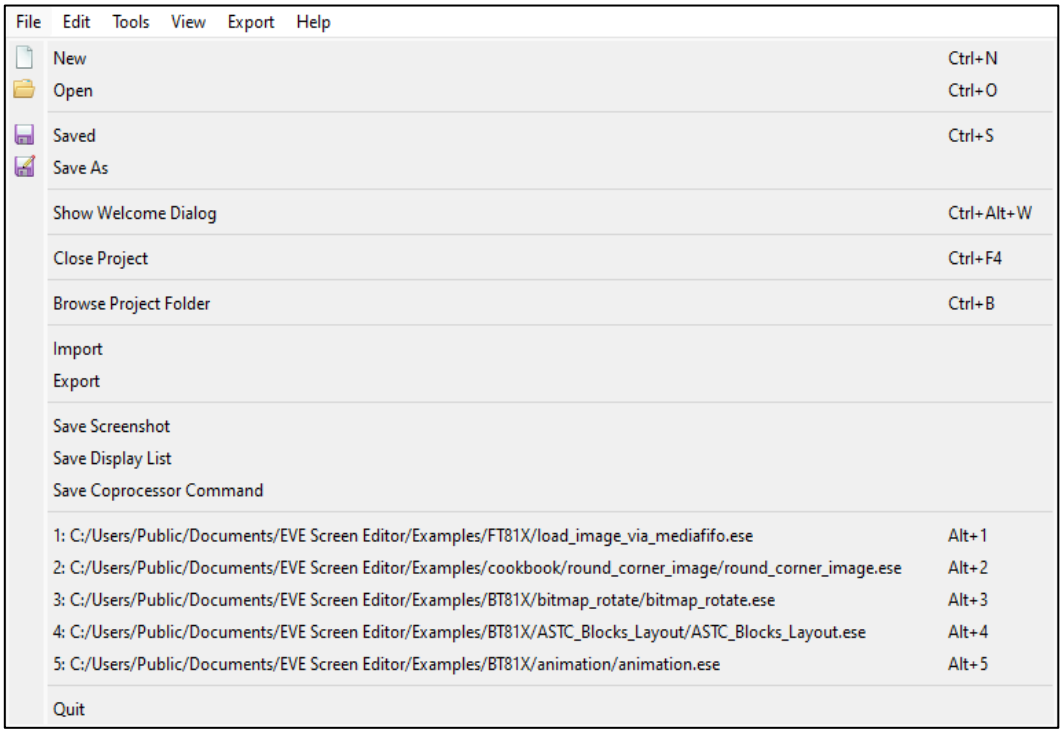


Figure 4 - File Menu

Menu Item	Description
New	To create a new project, clear the screen.
Open	To open/ retrieve the existing project. The file extension can be ".ese", ".ft8xxproj" or ".ft800proj". Example projects can be viewed.
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.
Show Welcome Dialog	To open the welcome dialog.
Close Project	To close the current project.
Browse Project Folder	To open the project folder where the current project file exists.
Import	To load the memory dump file which has an extension ".eve_dump."
Export	To save what is present in the memory of an EVE chip to a file.
Save Screenshot	To save the snapshot of the screen (i.e. what is 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 the 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 five opened projects are added into history list. Clicking 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*.

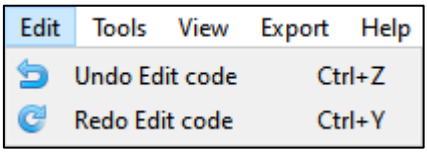


Figure 5 - Edit Menu

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.*

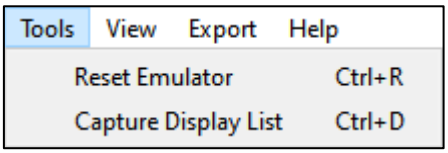


Figure 6 - Tools Menu

Menu Item	Description
Reset Emulator	To reset the emulator that is running in the background
Capture Display List	To capture the active display commands from the emulator into the 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.

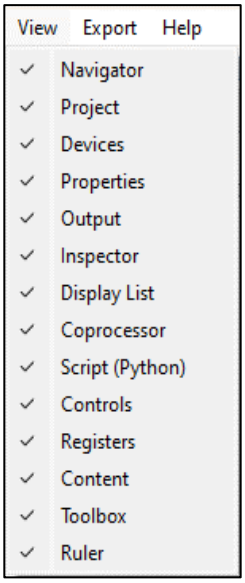


Figure 7 - View Menu

Export Menu

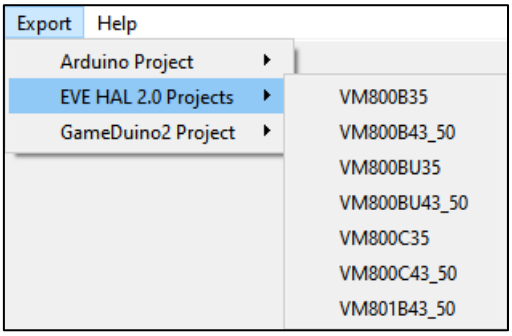


Figure 8 - Export Menu

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 them to Gameduino2, EVE Arduino, and HAL based projects.

For FT81X/BT81X/BT82X-based projects, there are scripts to export them to HAL 2.0 based projects.

Help Menu

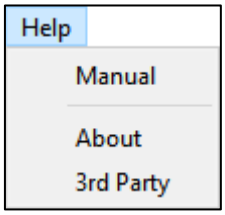


Figure 9 - Help Menu

Menu Item	Description
Manual	This document
About	Version and copyright disclaimer of ESE and EVE emulator, including support and update information.
3rd Party	Credits and copyright disclaimer of third-party software, assets used in ESE.

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 10 - 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	To 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 toggle show/ hide the ruler in emulator viewport.
Zoom Out	To zoom out emulator viewport.
Zoom In	To zoom in emulator viewport.
Zoom Rate	To choose a specific zoom level.

3. Status Bar

The status bar shows the consumption status of **RAM_G** and **RAM_DL** as well as bitmap handles in the form of percentage. It also shows the cursor position and pixel color in **(A, R, G, B)** format.

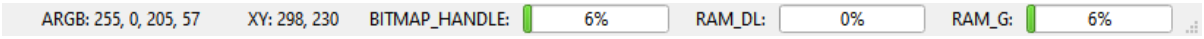


Figure 11 - Status Bar

When you hover over **BITMAP_HANDLE, RAM_DL, and RAM_G**, you'll see the ratio of utilized resources to the total available.

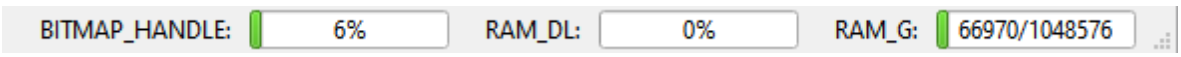


Figure 12 - RAM_G Usage on the Status Bar

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 the editing of 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** which 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.

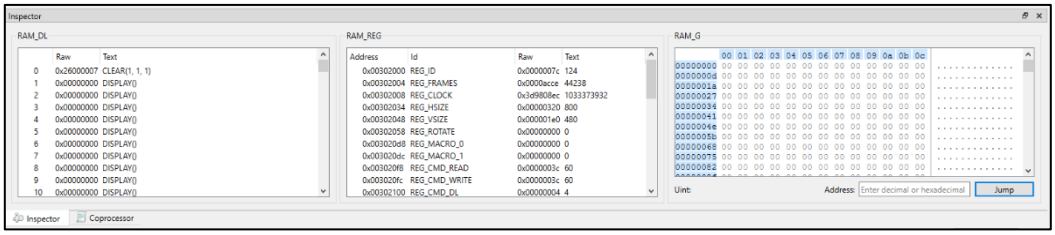


Figure 13 - Inspector

1. Coprocessor Command Editor

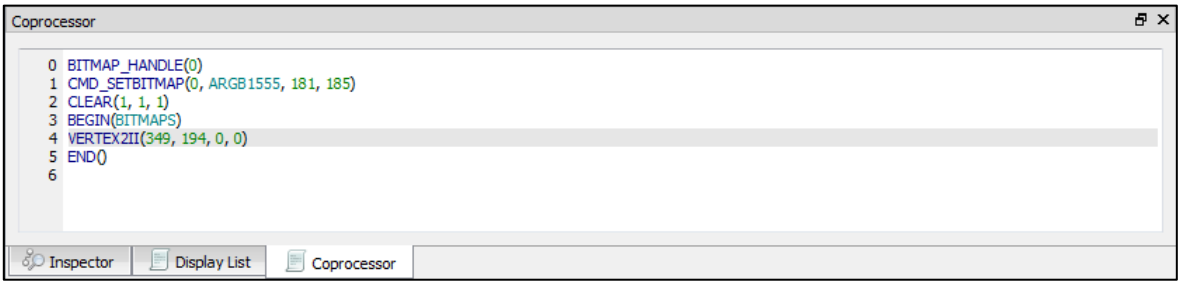


Figure 14 - Coprocessor Command Editor

The features are as below:

- Full set of commands support and auto-completion
- Decimal and hexadecimal values for parameters
- Error highlights
- Step-by-step emulation.
- Auto correct the user's input to proper upper/lower case

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.
- The terms "**RAM_G**" and "**RAM_DL**" are predefined keywords within the editor, and they are automatically recognized as referring to the base addresses of **RAM_G** and **RAM_DL**, respectively.

2. Display List Editor

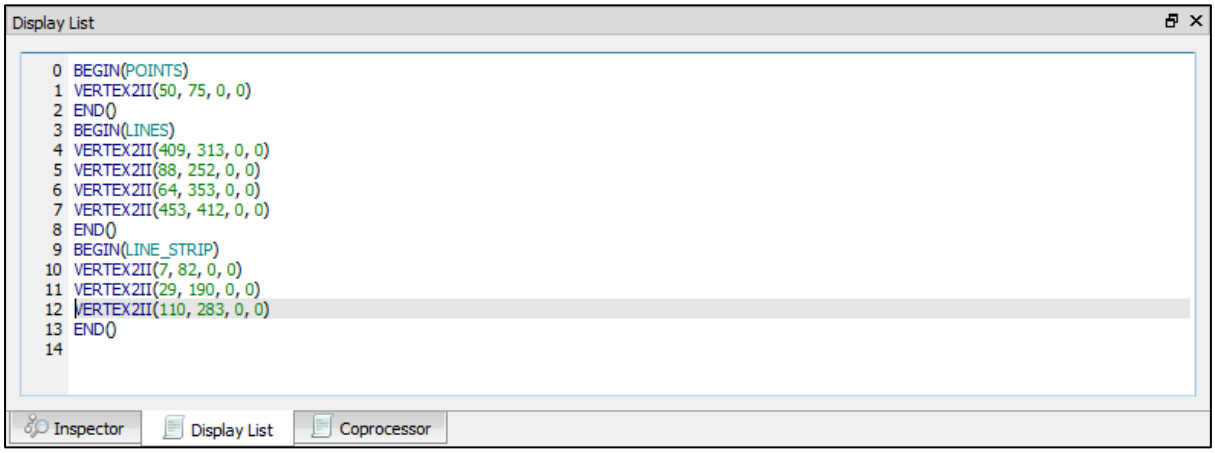


Figure 15 - Display List Editor

The features are as below:

- Display list commands auto-completion
- Decimal and hexadecimal values for parameters
- Error highlights
- Step-by-step emulation
- Auto correct the user's input to proper upper/lower case

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. Script Editor

This is an experimental feature that will help users write scripts to view dynamic/multiple screens.

Two types of input are supported:

String: Similar to the Coprocessor/Display List editor, ESE executes the commands in a string that you transfer.

Inline: Invoke the commands as Python lib. All commands are added to a list. The execute function performs the commands in the list and renders them on viewport.

To use it, please take note of the following:

- Python is the only language supported in the Script editor.
- Import "interact" module and use it as below:
  - void **init**(): Initialize the feature.
  - void **reset**(): Clear all prepared commands displayed on ESE.
  - int **transfer**(**string**): Transfer the commands as a string to the emulator. This function will return a negative number when the user has stopped executing the script. Therefore, please make sure to check the return value of this function to call deinit() and halt the script.
  - int **execute**(): Push the prepared commands into ESE to display. This function will return a negative number when the user has stopped executing the script. Therefore, please make sure to check the return value of this function to call deinit() and halt the script.
  - void **deinit**(): Deinitialize the feature.
  - void **print**(**string**): Print a message out. The message will be displayed in the Output window.

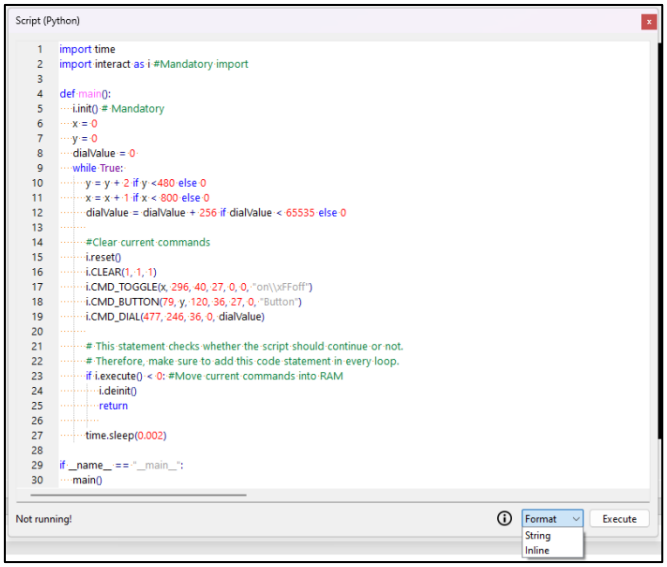


Figure 16 - Script Editor

Format: The project loads an example script from the "scripts" folder in the project directory. If there is no such file, it uses the default example script in the "Example/scripts" folder instead. String and Inline are the two options for loading an example in ESE.

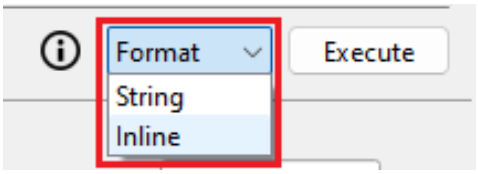


Figure 17 - Choose a format for loading example

Execute: Run the script from the Script editor and show the rendered result in the viewport.

4. Inspector

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

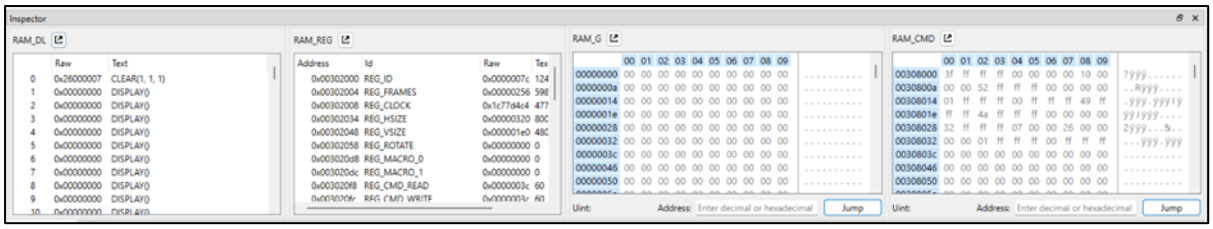


Figure 18 - Inspector

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.

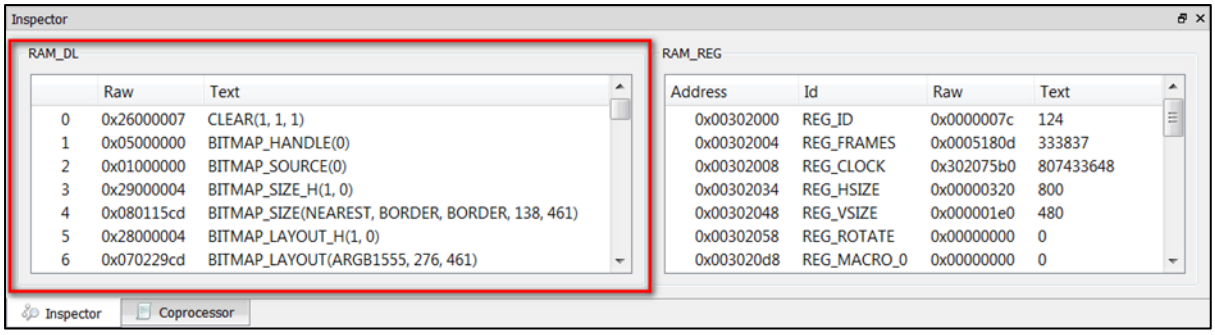


Figure 19 - RAM_DL

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

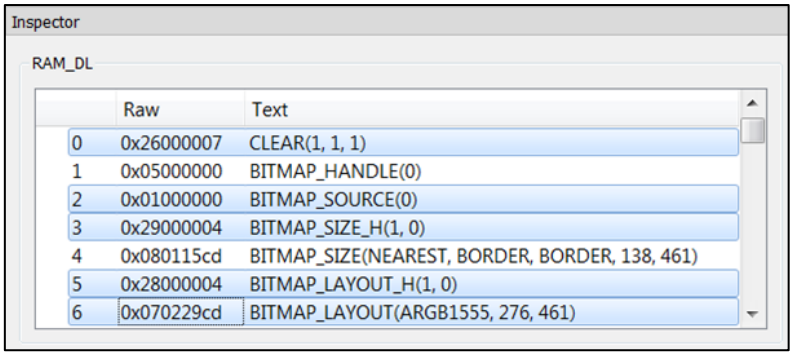


Figure 20 - Select Rows in RAM_DL

	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 21 - 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**.

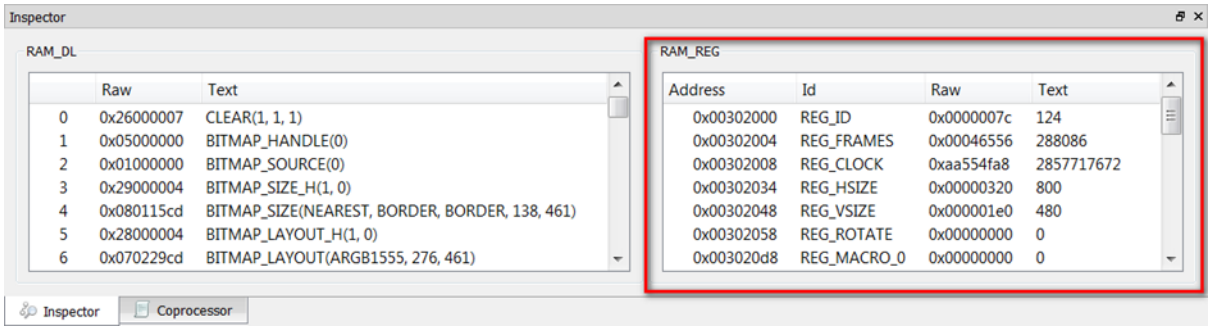


Figure 22 - RAM_REG

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

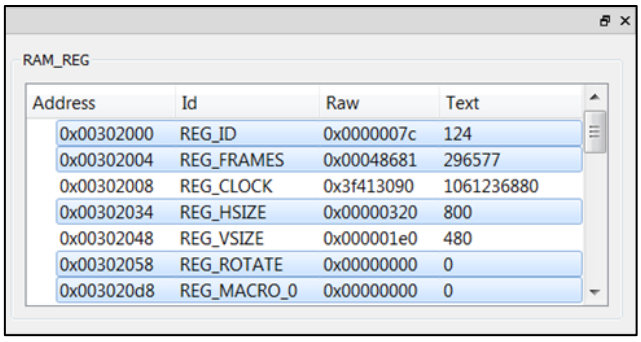


Figure 23 - Select Rows in RAM_REG

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 24 - 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**.

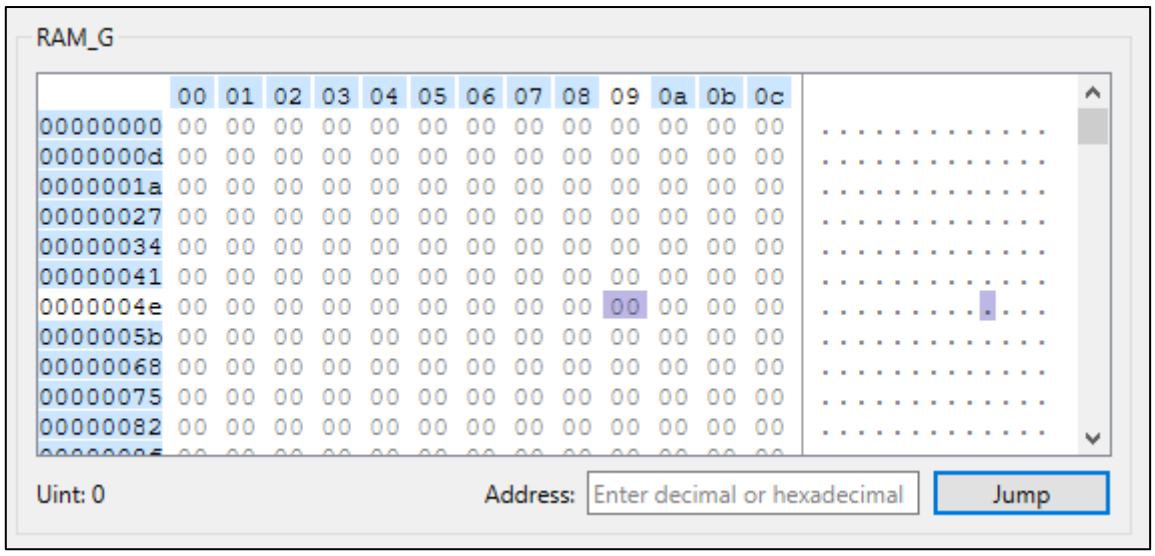


Figure 25 - RAM_G

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

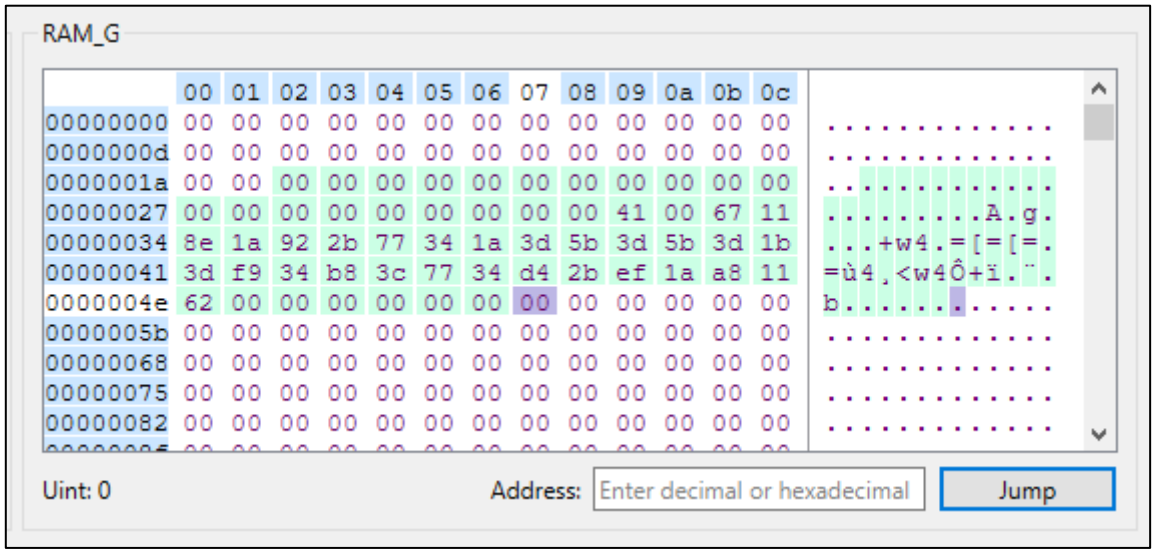


Figure 26 - Area Selection

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

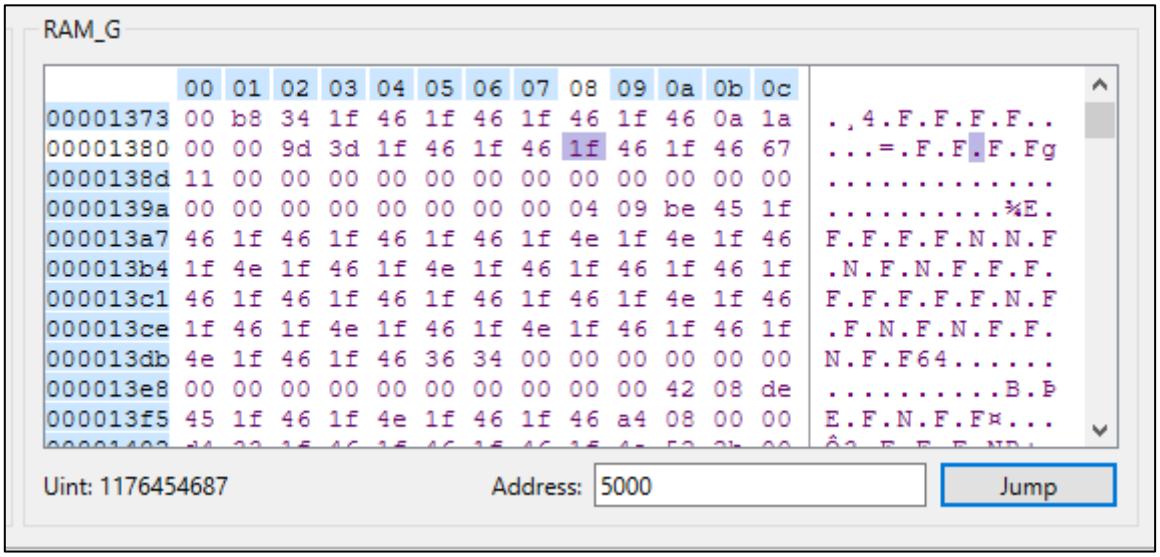


Figure 27 - Jump to a Specific Address with Decimal

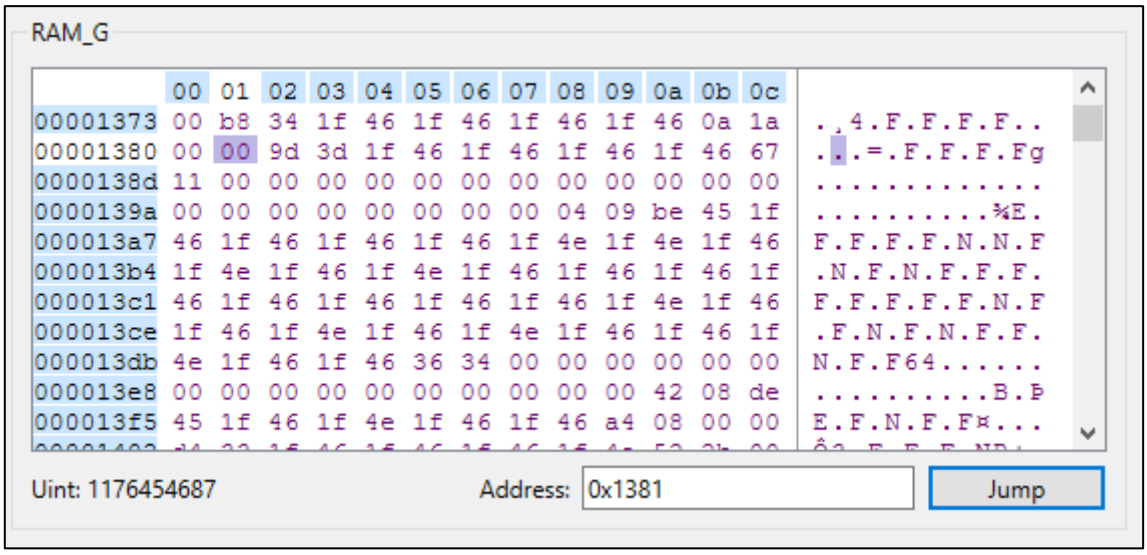


Figure 28 - Jump to a Specific Address with Hexadecimal

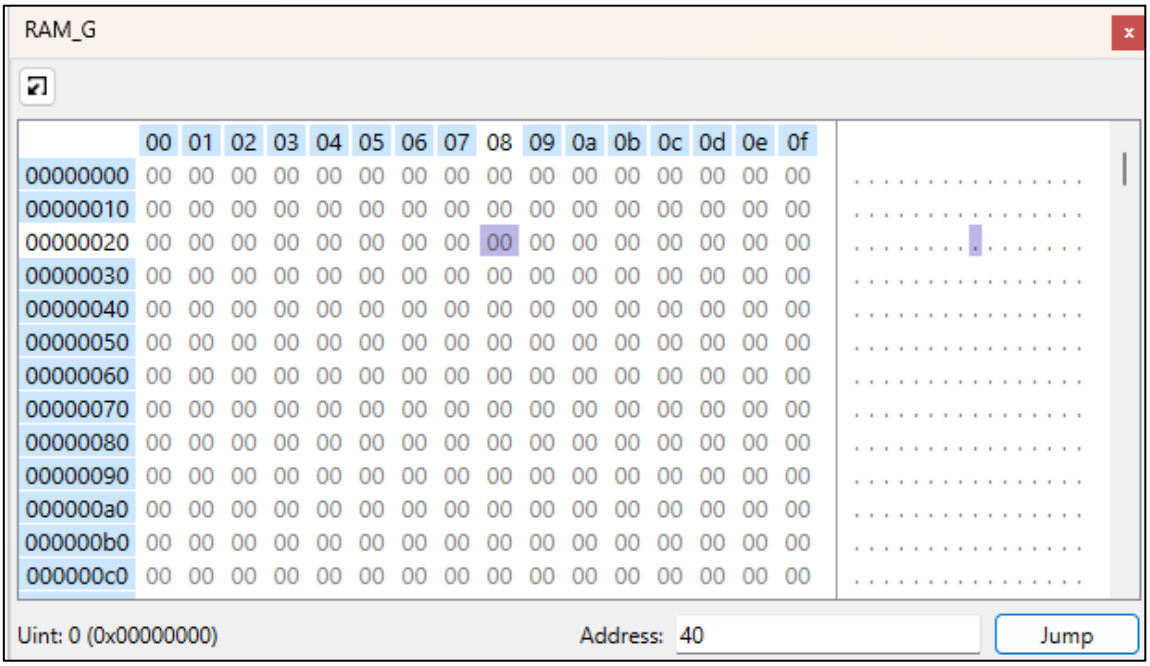


Figure 29 - Jump to a Specific Offset

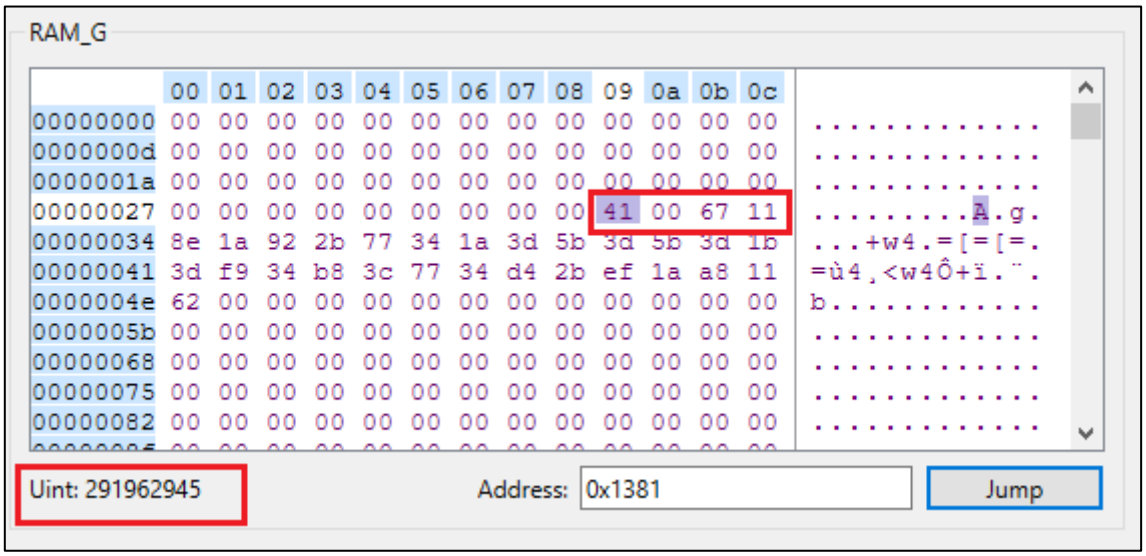


Figure 30 - Unsigned integer value of 4 consecutive bytes (little endian)

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 unsigned integer value of 4 consecutive bytes in decimal, assuming little endian.

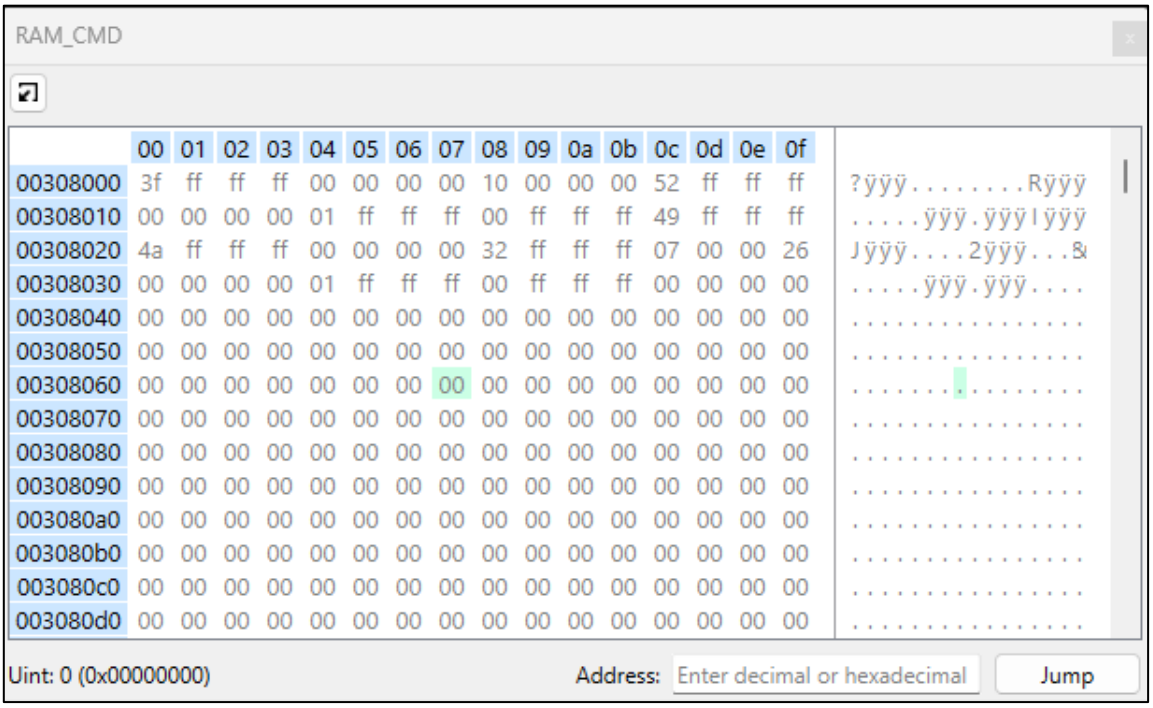


Figure 31 - RAM_CMD

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

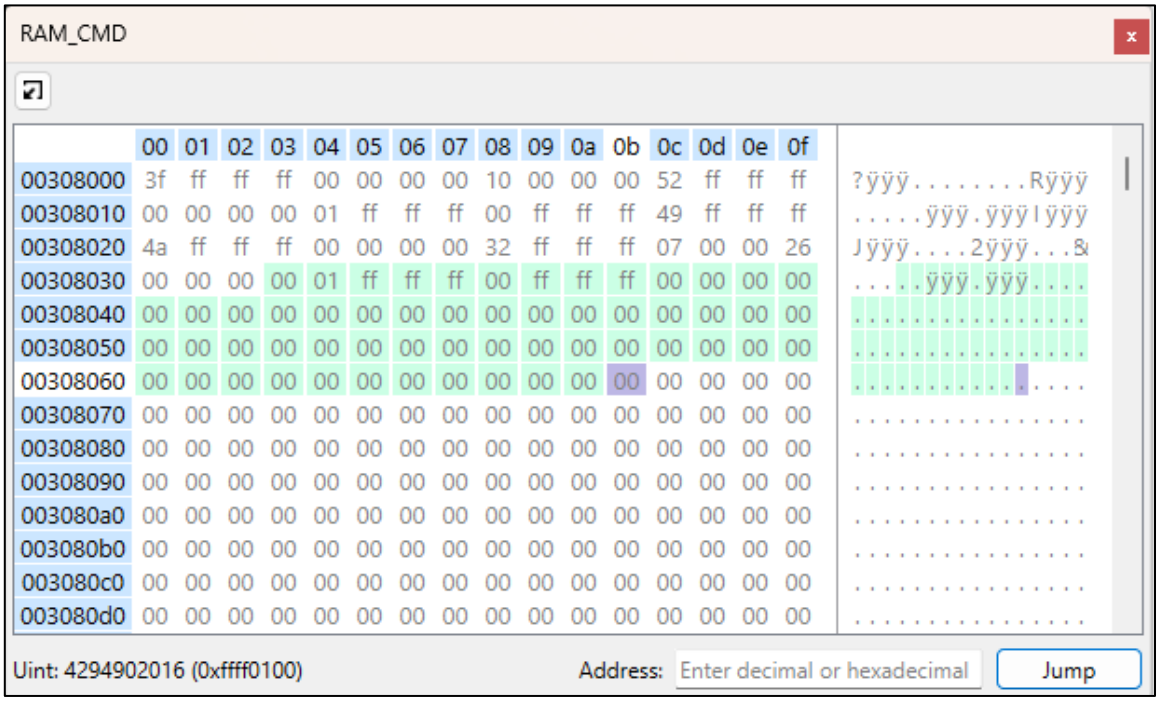


Figure 32 - Area Selection

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

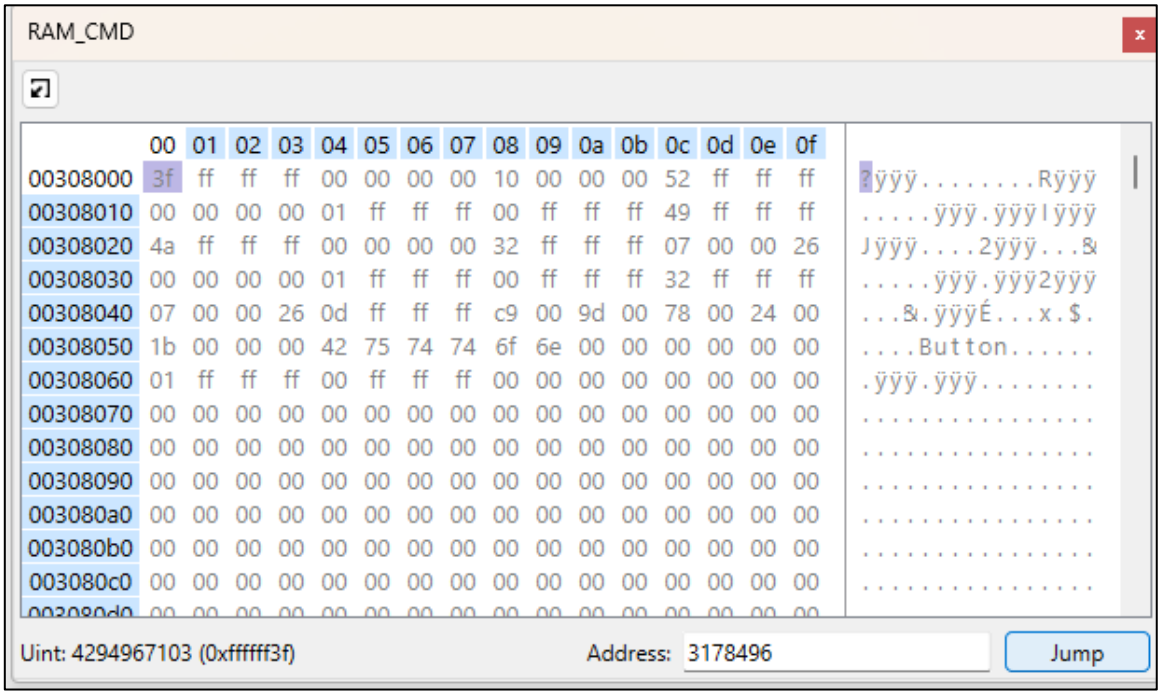


Figure 33 - Jump to a Specific Address with Decimal

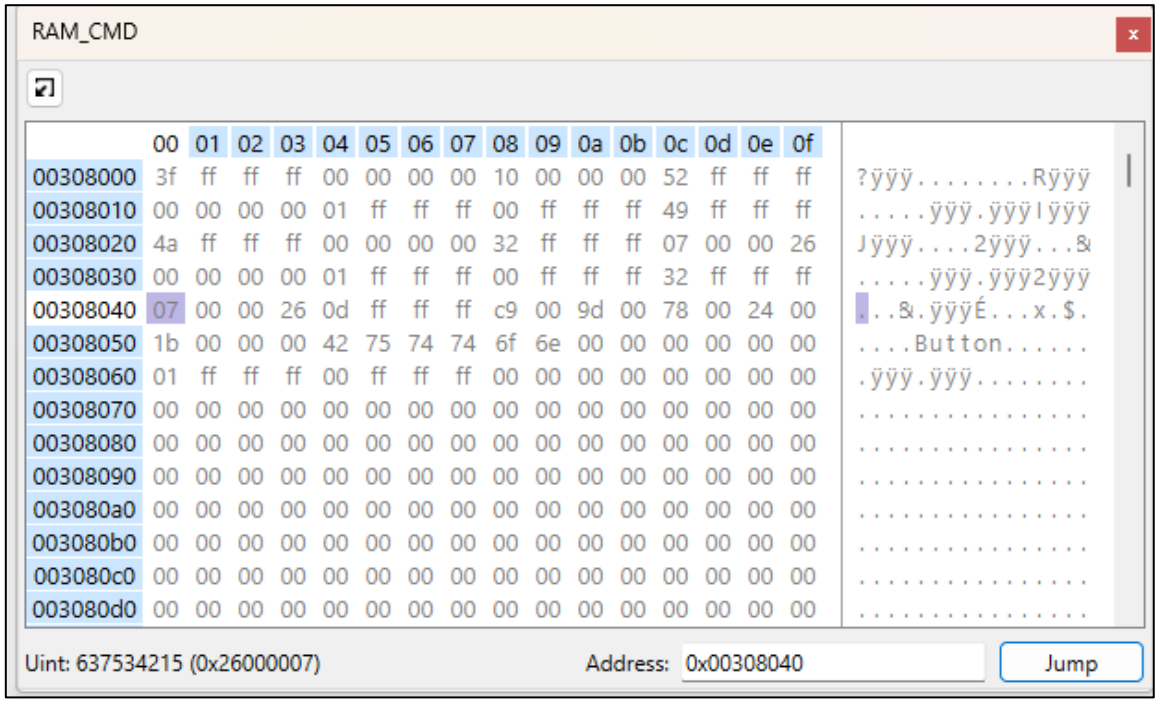


Figure 34 - Jump to a Specific Address with Hexadecimal

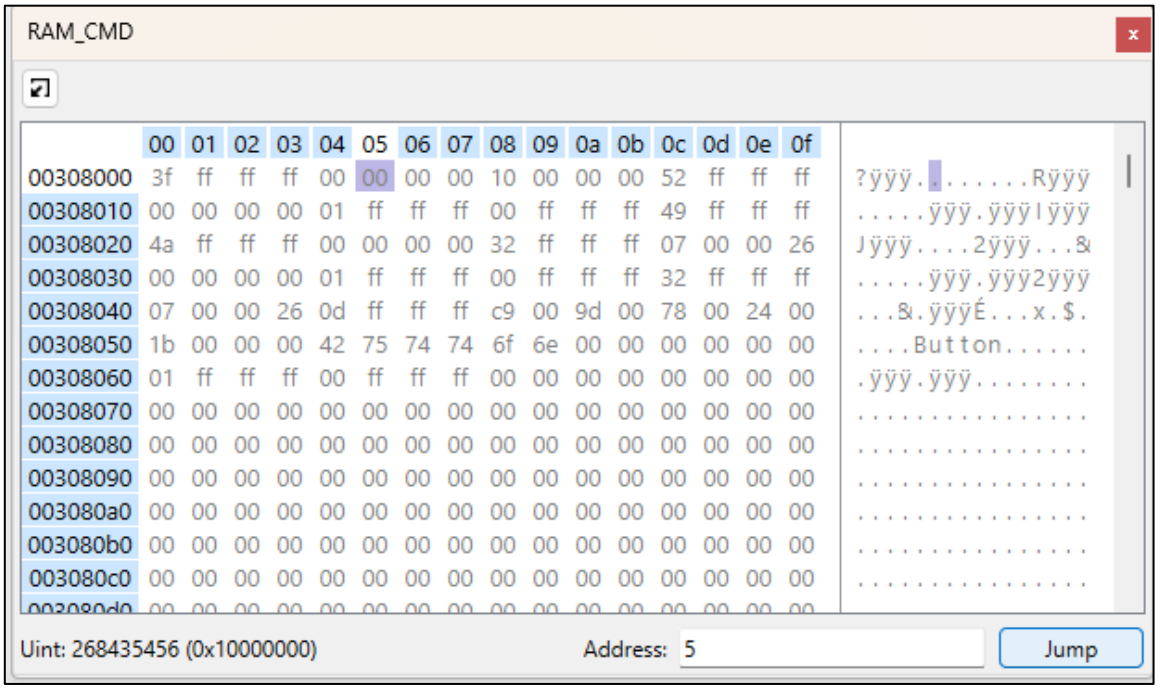


Figure 35 -Jump to a Specific Offset

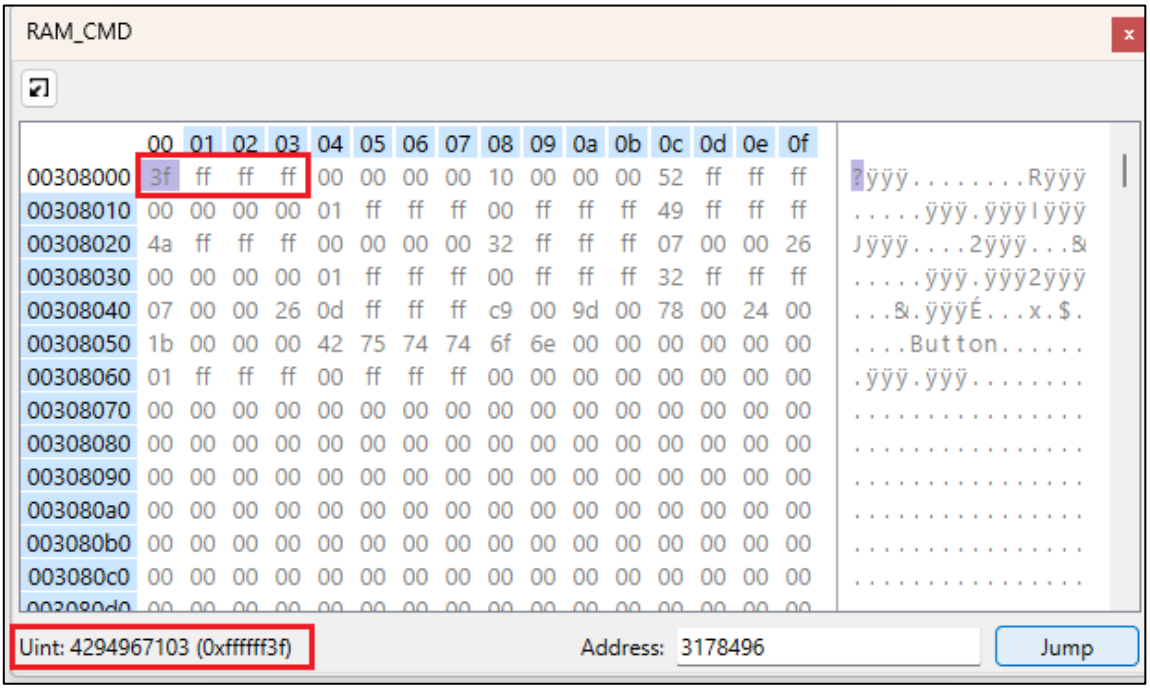


Figure 36 - Unsigned integer value of 4 consecutive bytes (little endian)

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 a 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.

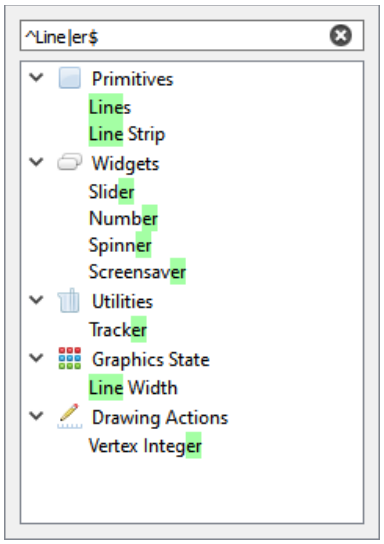


Figure 37 - Toolbox Filter

Add an item

The command in the toolbox can be dragged and dropped into the emulator viewport. Users can also double-click to add an item. The editor will be updated with the corresponding commands.

Display List Mode

Select the Display List Editor window to use the Display List mode:

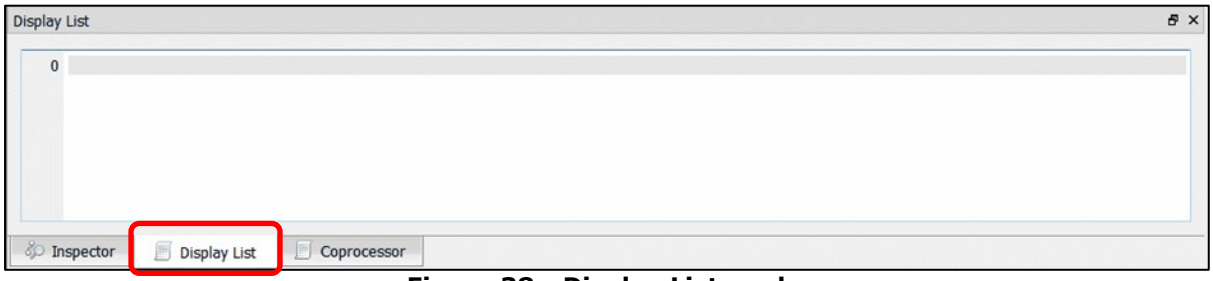


Figure 38 - Display List mode

The toolbox will be enabled as below:

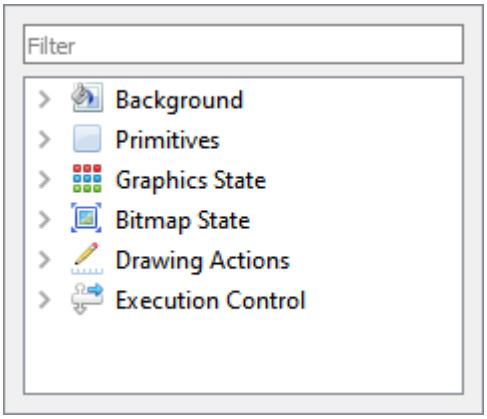


Figure 39 - Toolbox in Display List mode

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

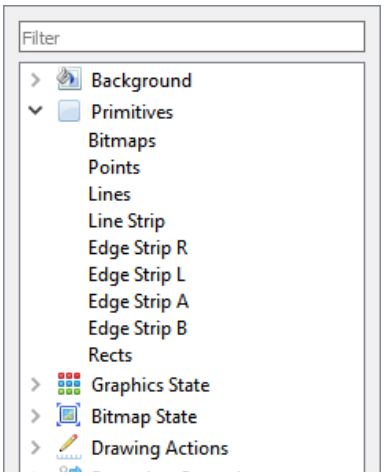


Figure 40 - Primitives in Display List Mode

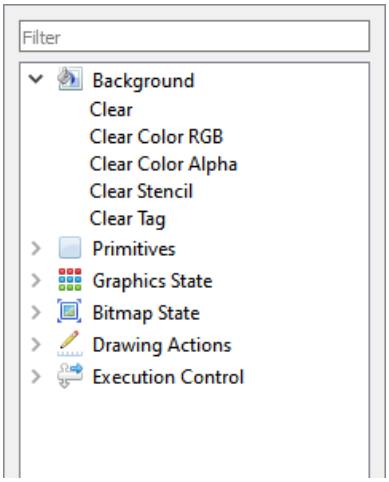


Figure 41 - Background in Display List Mode

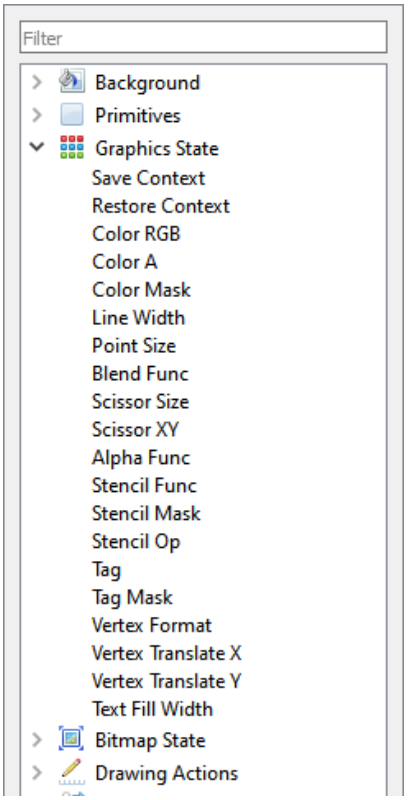


Figure 42 - Graphics State in Display List Mode

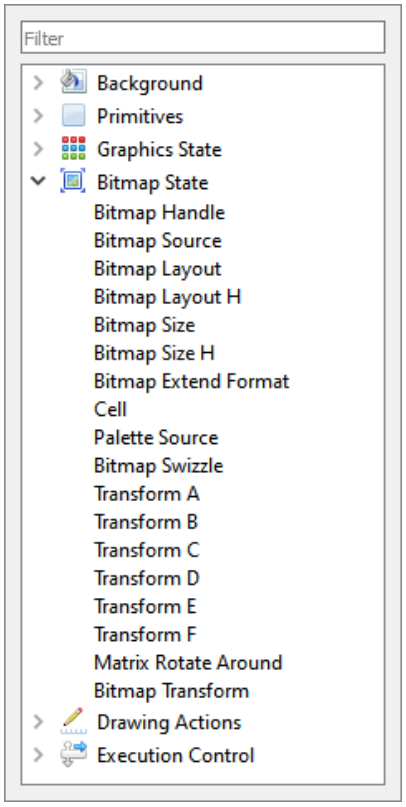


Figure 43 - Bitmap State in Display List Mode

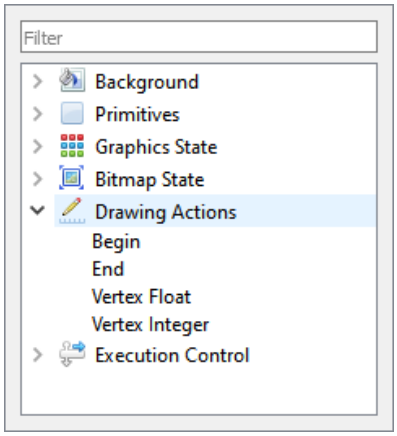


Figure 44 - Drawing Actions in Display List Mode

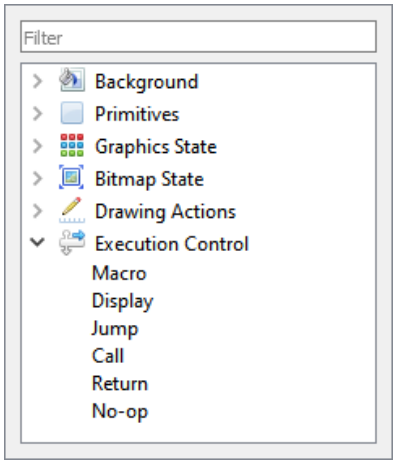


Figure 45 - Execution Control in Display List Mode

Coprocessor Mode

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

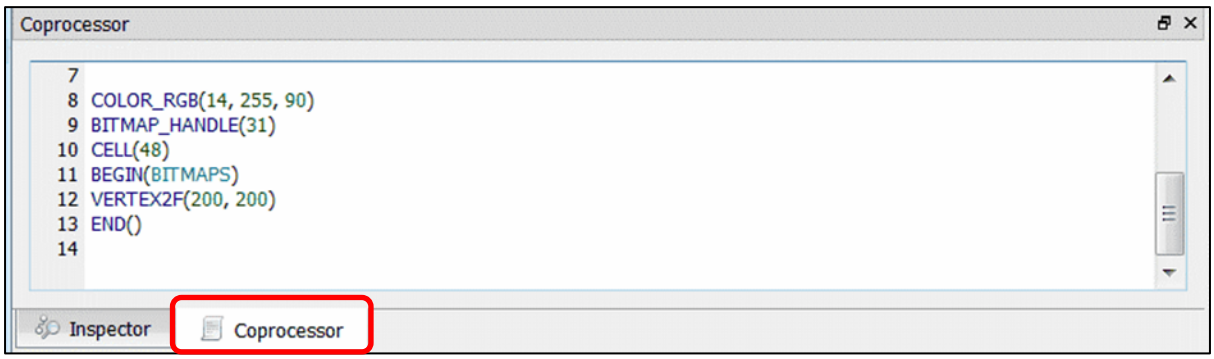


Figure 46 - Coprocessor Mode

The toolbox will be enabled as below:

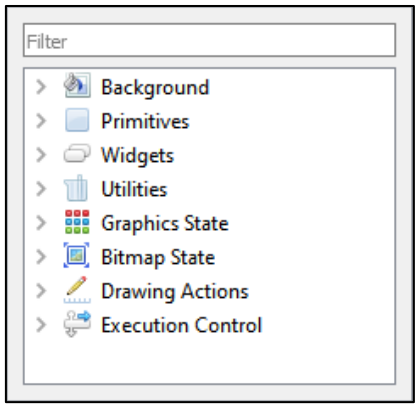


Figure 47 - Toolbox in Coprocessor Mode

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

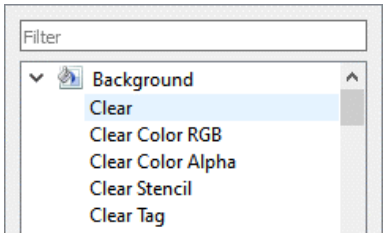


Figure 48 - Background in Co-processor Mode

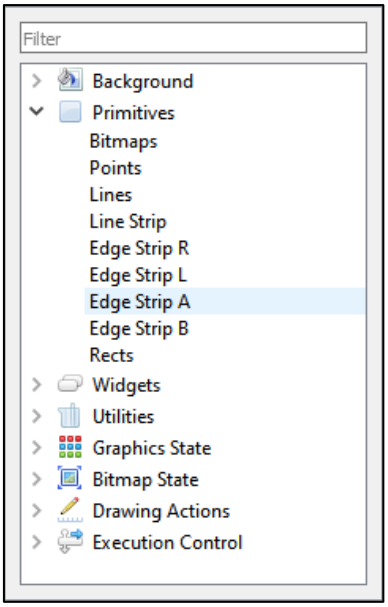


Figure 49 - Primitives in Co-processor Mode

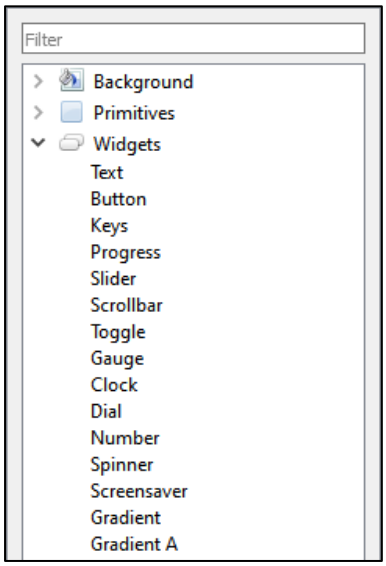


Figure 50 - Widgets in Co-processor Mode

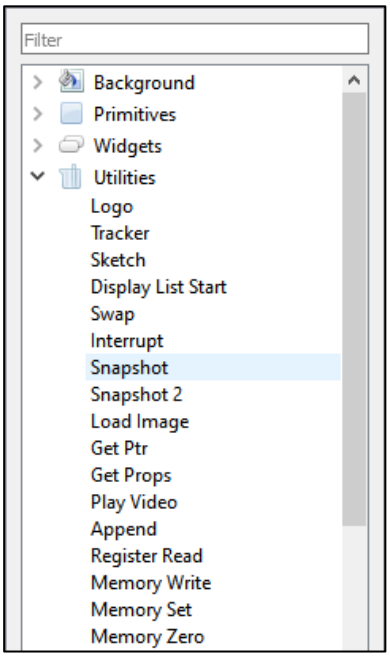


Figure 51 - Utilities in Co-processor Mode

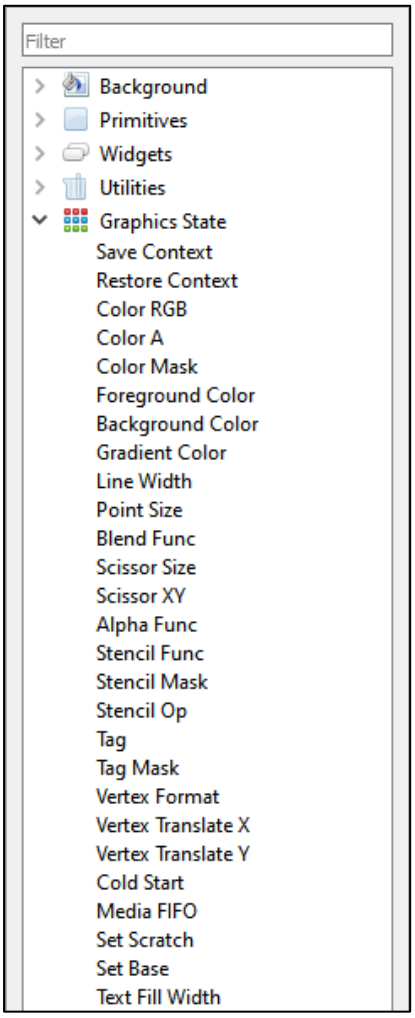


Figure 53 - Graphics State in Coprocessor Mode

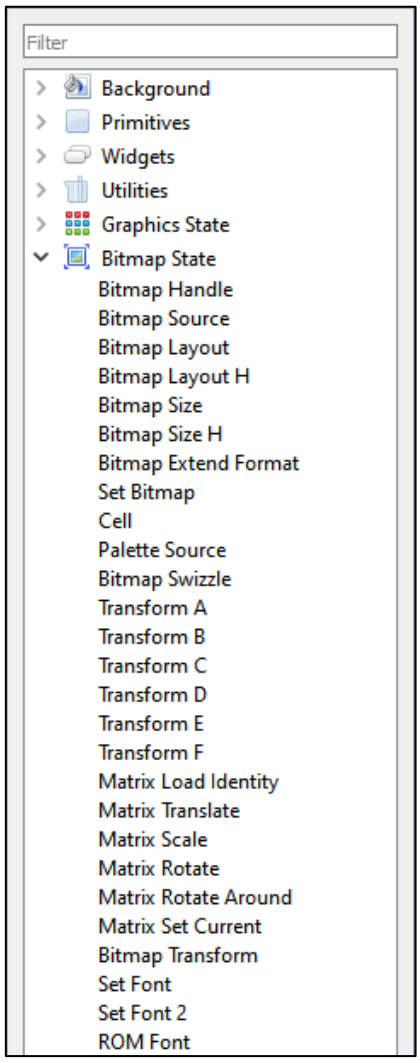


Figure 54 - Bitmap State in Coprocessor Mode

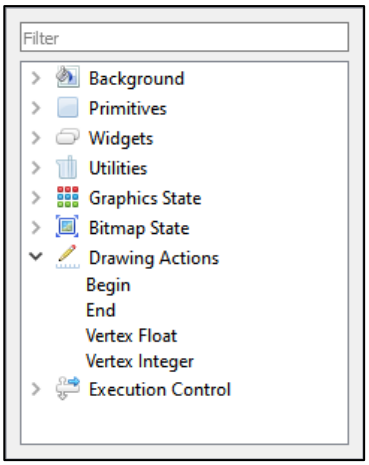


Figure 52 - Drawing Actions in Coprocessor Mode

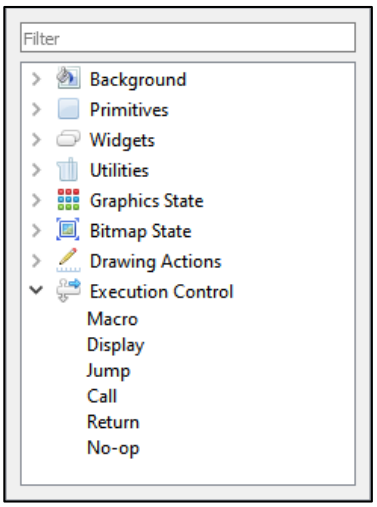


Figure 55 - Execution Control in Coprocessor Mode

2. Registers

This tab is used to set up screen size and macro registers, control **Width of HSF** and other register's value such as **REG_ROTATE, REG_PLAY_CONTROL** and **REG_FREQUENCY**. The **REG_MACRO_0** and **REG_MACRO_1** register can be edited in the editor box of Macro and 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.

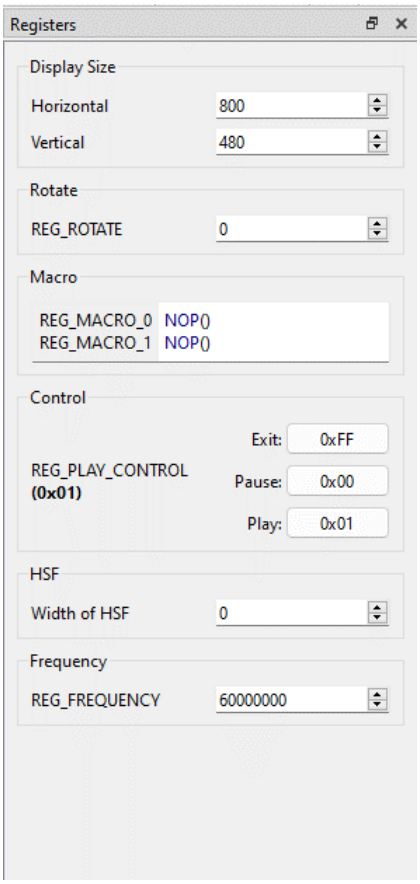


Figure 56 - Register

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.

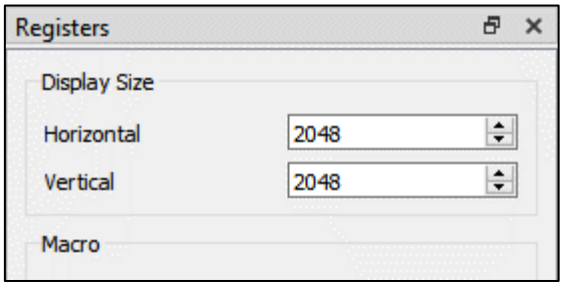


Figure 57 - Customize Resolution in Register Window

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

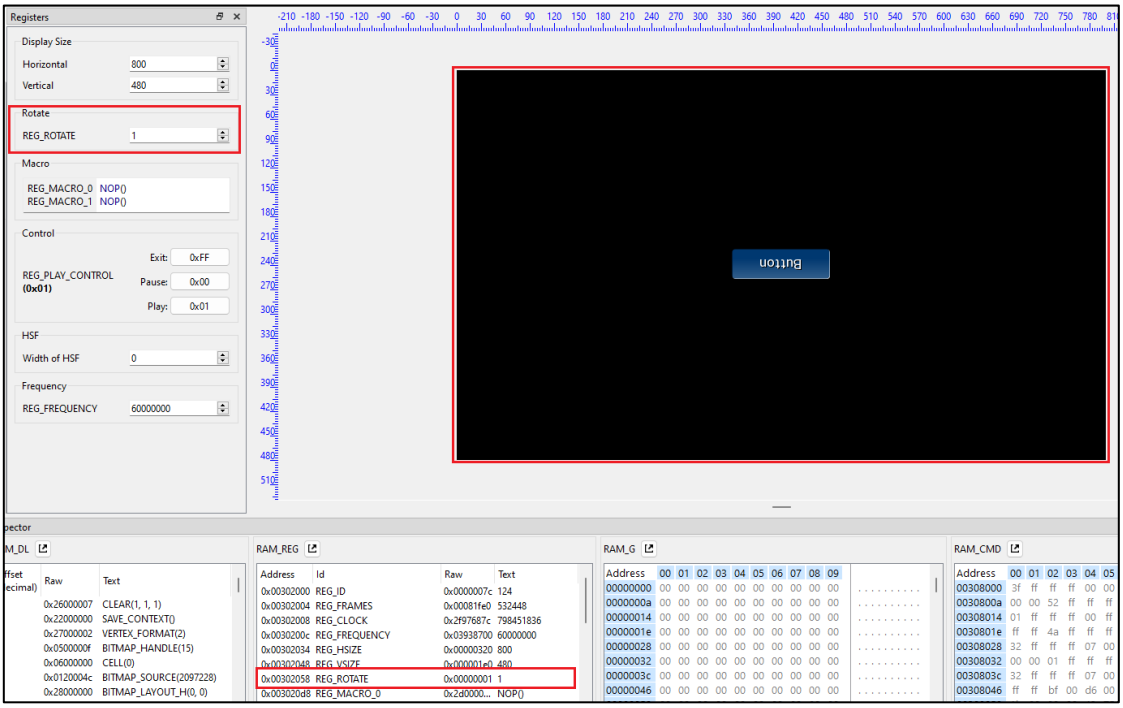


Figure 58 - Change Screen Rotation using REG_ROTATE

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

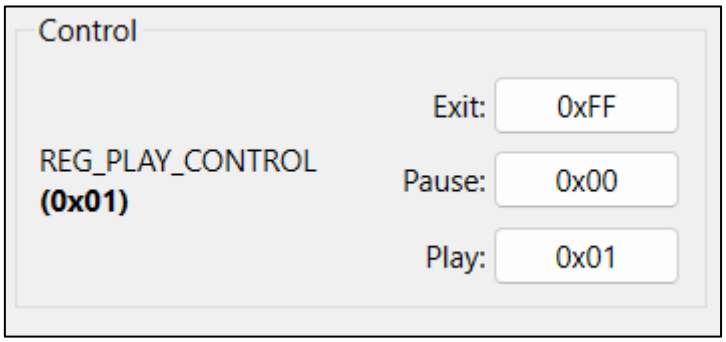


Figure 59 - Control Section

There is an option to enable HSF (Horizontal Scanout Filter). The default value should be zero, which will disable HSF. Changing the value of this box has the same effect as using **CMD_HSF.**

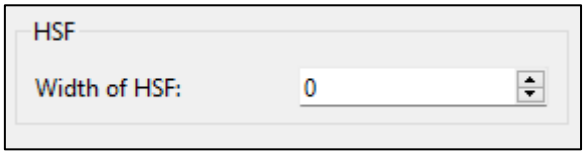


Figure 60 - Width of HSF Section

The main clock frequency can be changed by modifying **REG_FREQUENCY.** The value is in Hz and by default is 60MHz.

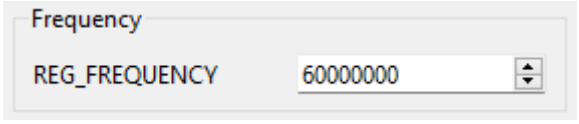


Figure 61 - The main clock frequency using REG_FREQUENCY

On BT82X, we support changes to REG_SC0_PTR0, REG_SC0_PTR1, and REG_SO_FORMAT.

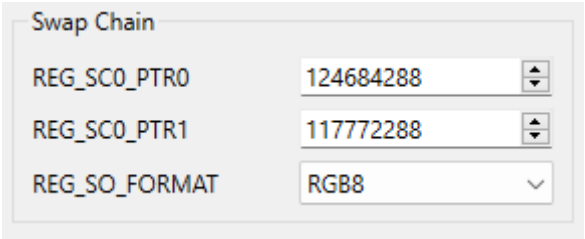


Figure 62 - Change Swap Chain registers

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.

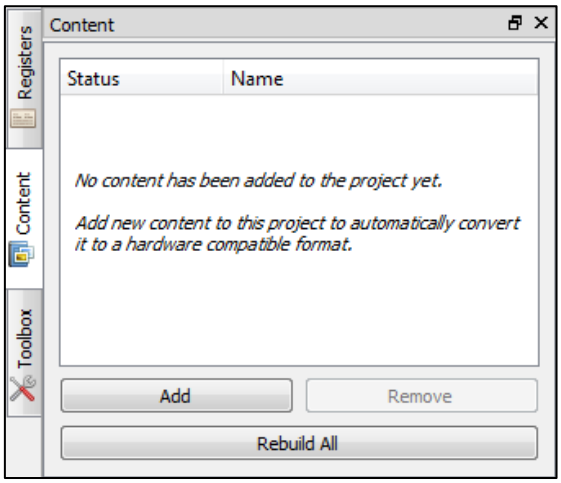


Figure 63 - Content Manager

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 Add Content dialog pops up. Browse and select the file to be added.

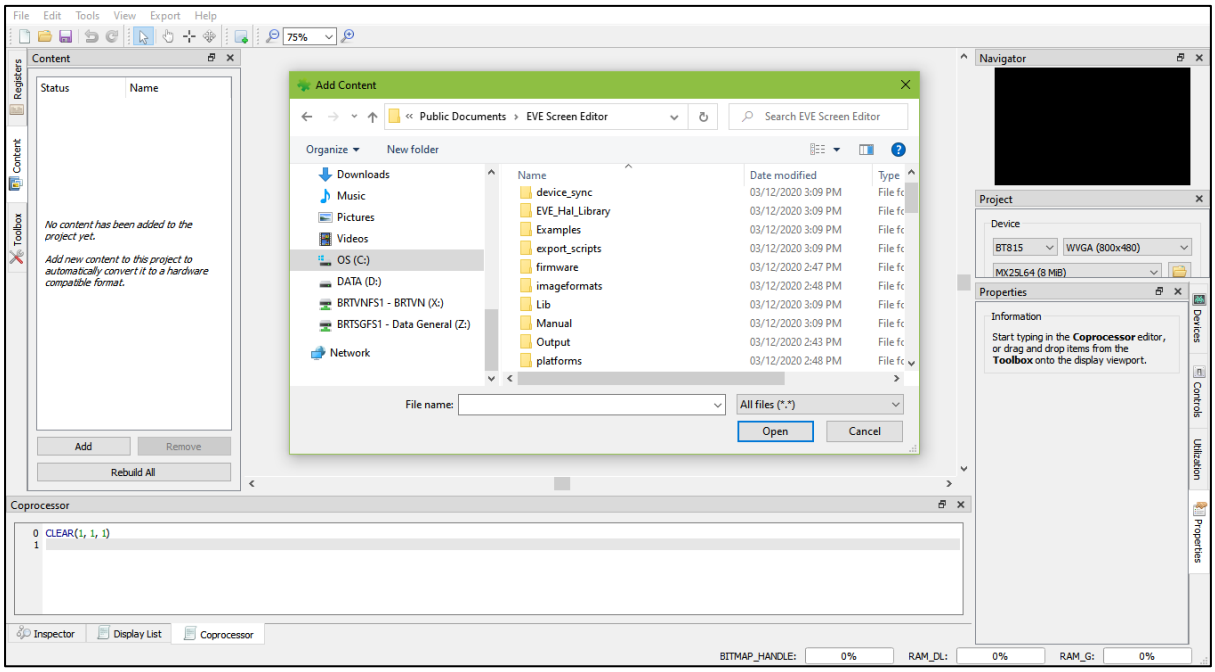


Figure 64 - Add Content Dialog

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.

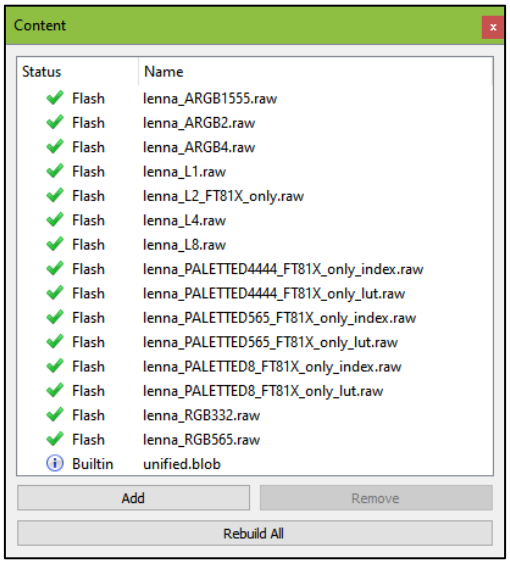


Figure 65 - Content Loaded

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.

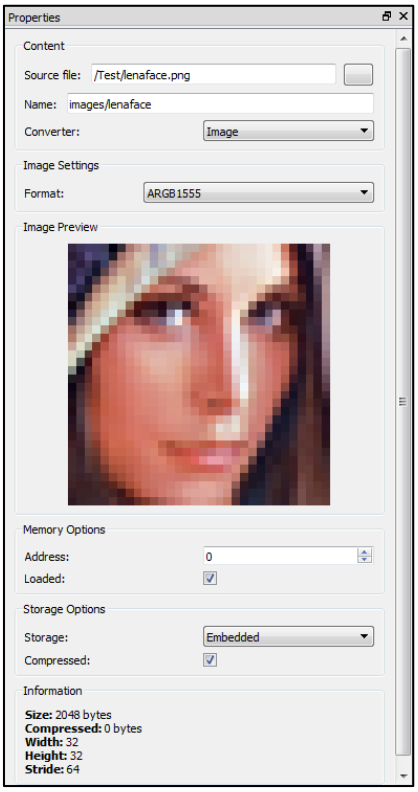


Figure 66 - Content Properties

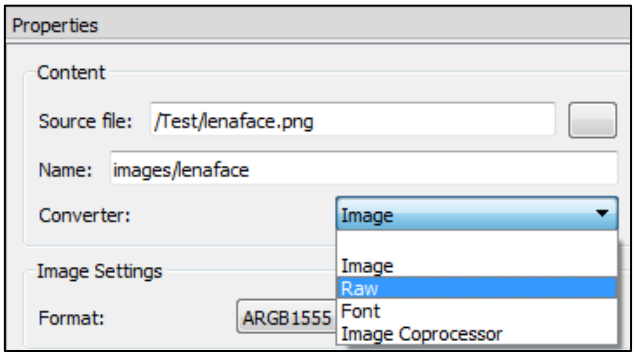


Figure 67 - Raw Converter

6. When the converter is selected as 'Image Coprocessor', no conversion is triggered for the input image. Instead, ESE will load the input image with CMD_LOADIMAGE, which decodes the input image into the loaded address. With this option, users must ensure the input image is an EVE-compatible PNG or JPEG file.

7. Upon image conversion, users can drag the image from the Content Manager and drop it into the viewport.

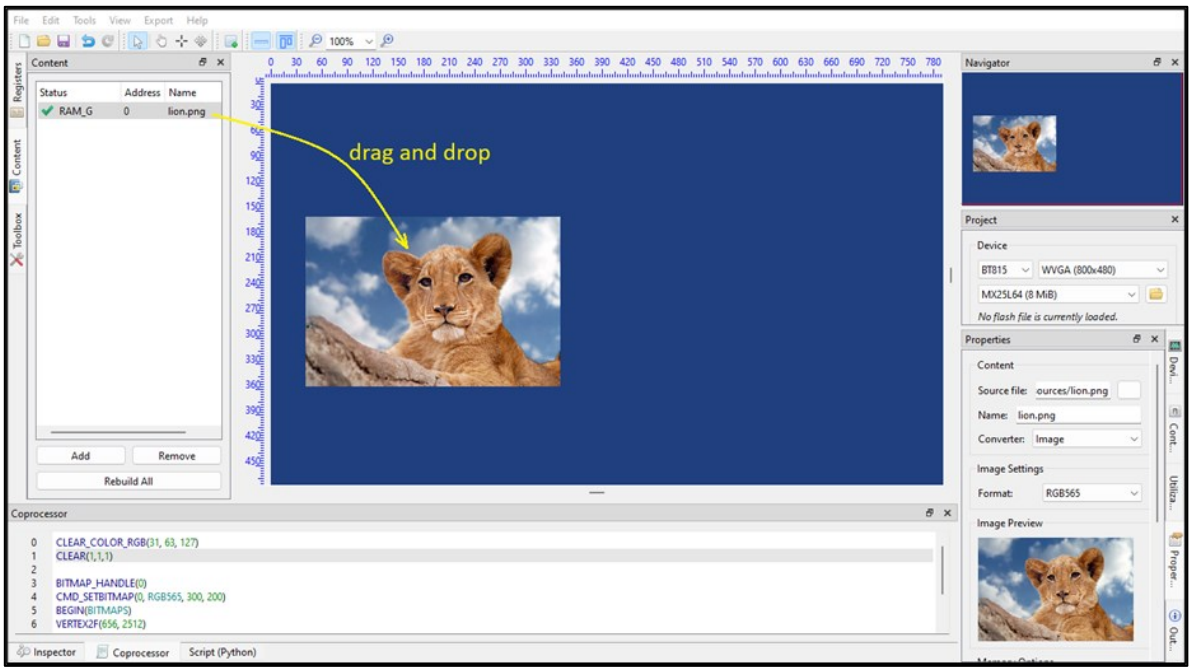


Figure 68 - Drag Content into Viewport

After placing the image, the display list will be generated in the editor automatically and 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]**.

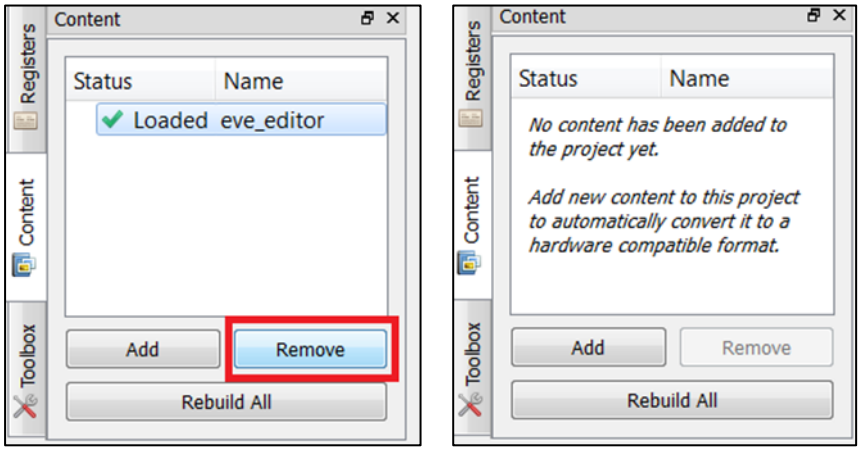


Figure 69 - Remove Content

- The selected content is then removed from the list.

Rebuild the Content

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

Convert the Content

The following information is for users who want to program EVE directly. Currently, ESE is using the utilities: FreeType, libpng, pngquant, ASTC Encoder. These utilities are not required to use in the ESE, but they help convert the resource to the necessary files that EVE can understand.

For each valid resource in the Content Manager, the utility converts it to the file format below:

**.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 folders:

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

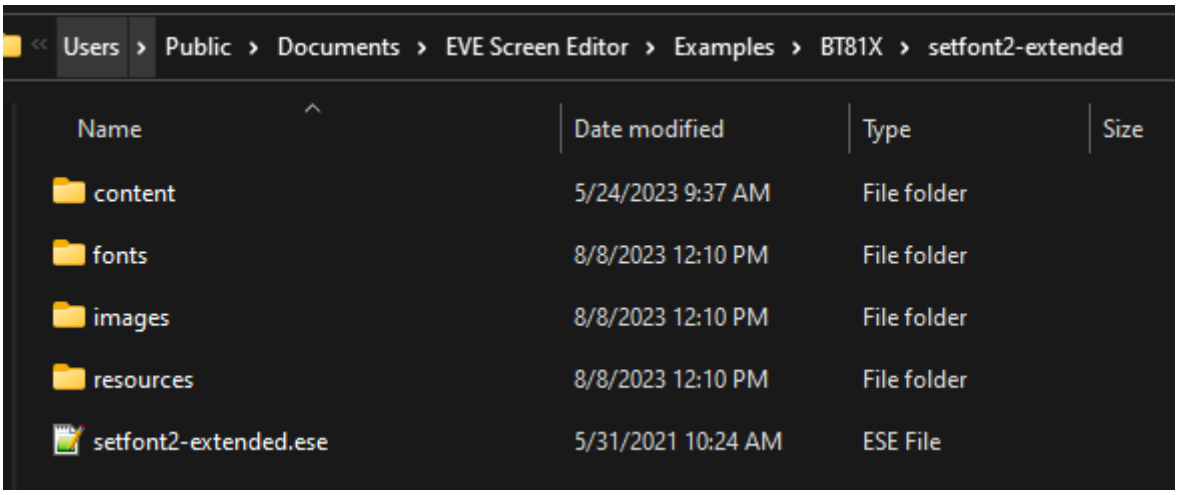


Figure 70 - Structure of a Project

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 directly observe the user's design on the EVE board hardware.

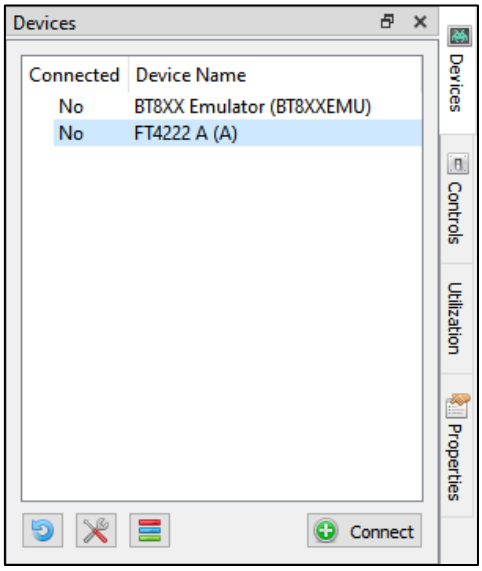


Figure 71 - Before Connecting

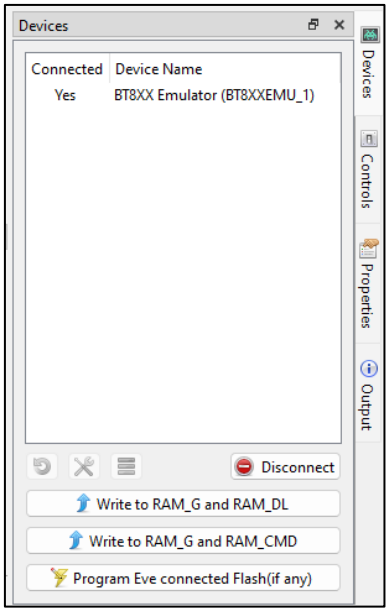


Figure 72 - Connected Device

The device type can be changed by clicking the Settings 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.

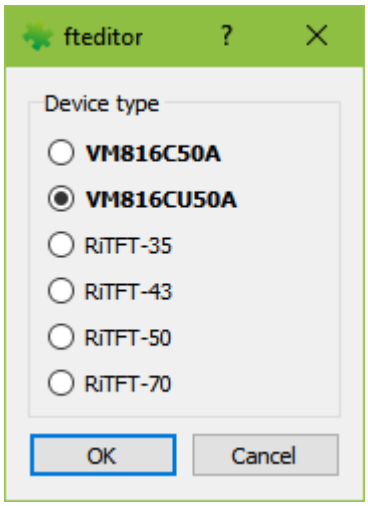


Figure 73 - Device Type

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.

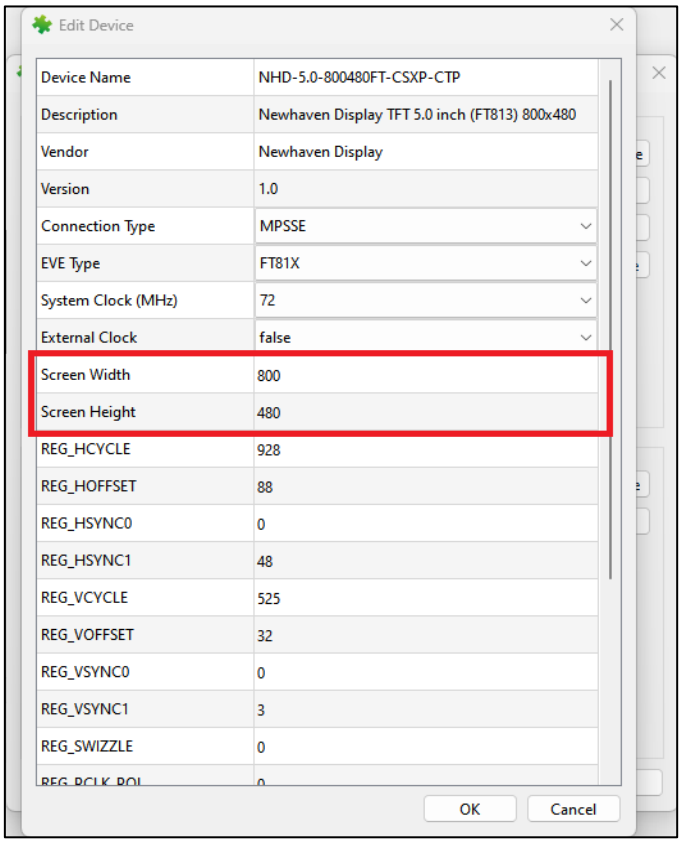


Figure 74 - Device Settings

2. Controls

In the *Controls* tab, users may execute the code step by step according to the granularity of the display list command or coprocessor command.

See the "Steps" grouped widgets in Figure 75.

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 **Step by Step** section for more details.

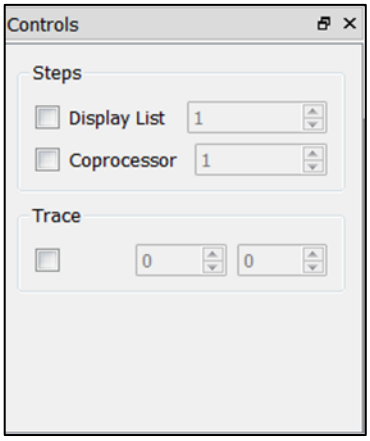


Figure 75 - Controls Tab

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

See the "Trace" grouped widgets in Figure 75. Refer to **Trace the Pixel** section for more details.

3. Properties

The Properties tab provides 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.

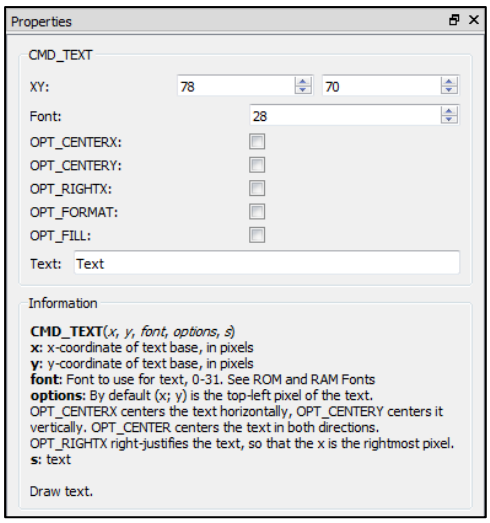
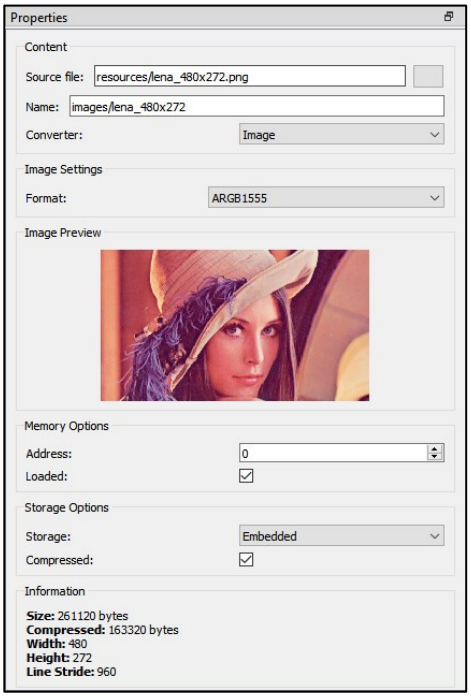


Figure 76 - Properties Tab



The Properties 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.

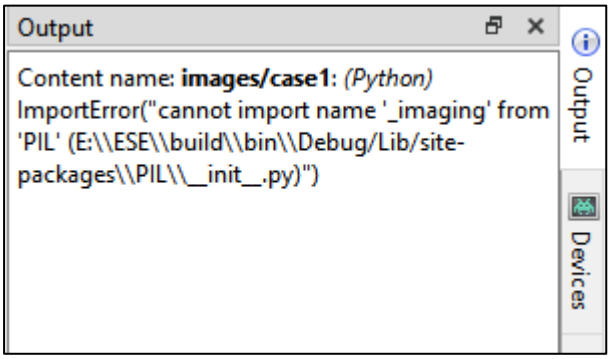


Figure 77 - Output Tab

F. Viewport

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.

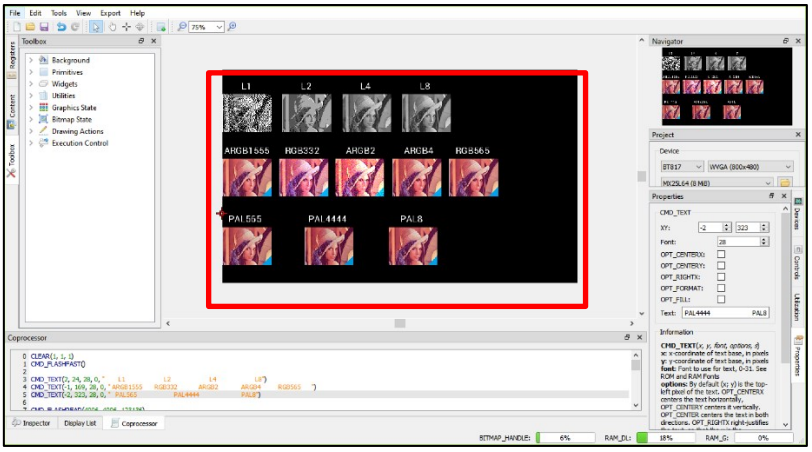


Figure 78 - Viewport

G. Navigator

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

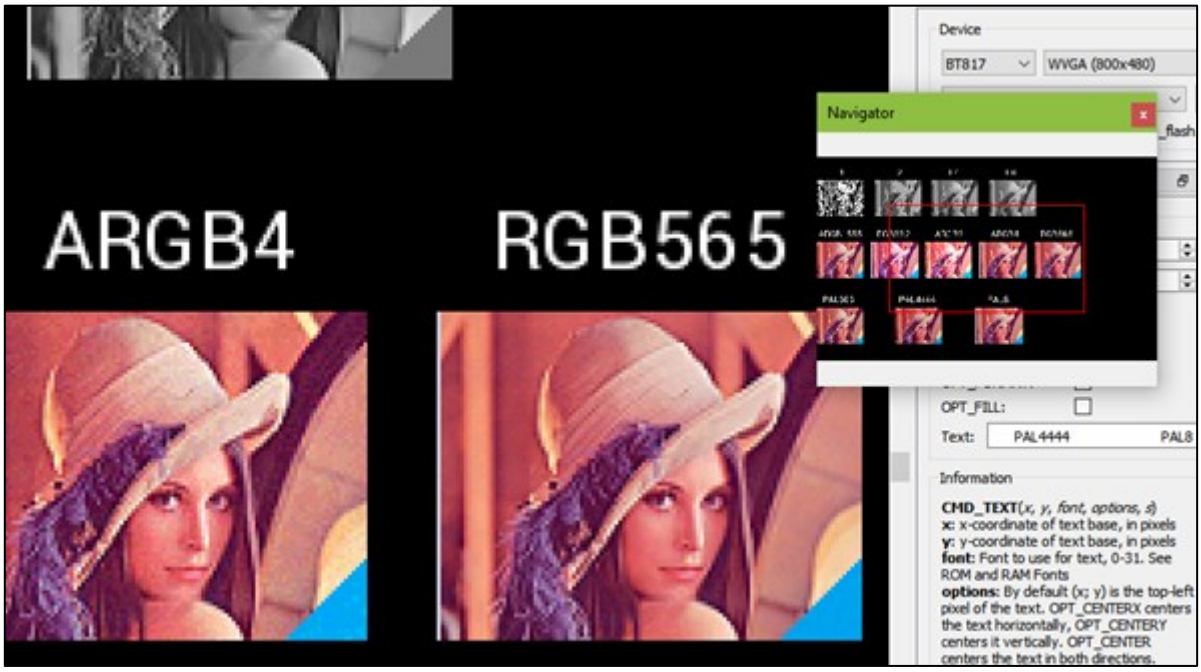


Figure 79 - Navigator

H. Project Settings

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

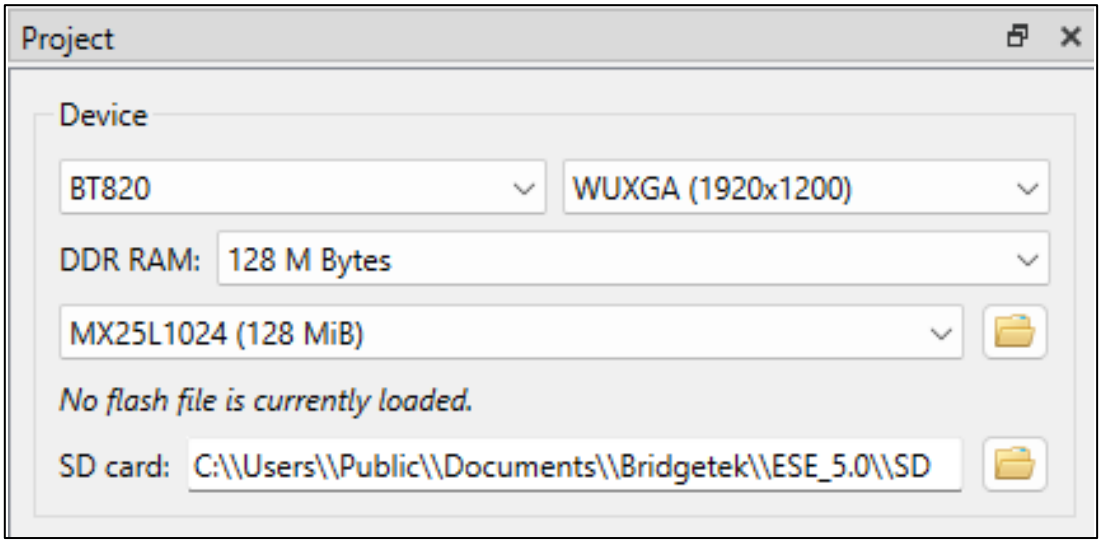


Figure 80 - Project Settings

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

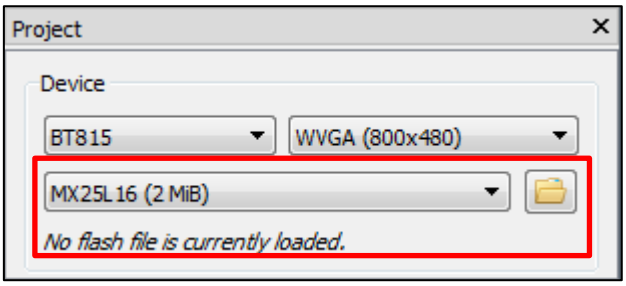


Figure 81 - Flash Is Supported

DDR RAM has been supported since BT82X. To change the DDR RAM size, click the drop-down box and select the desired size.

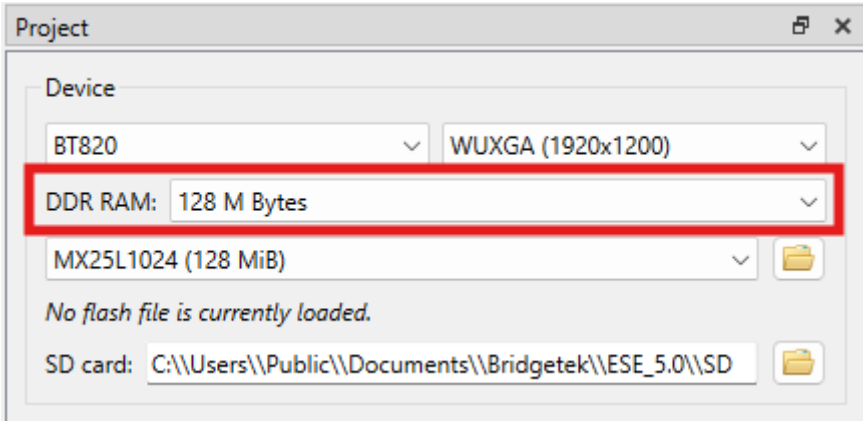


Figure 82 - DDR RAM size

ESE allows a folder to be used as an SD card on BT82X. Users can specify which folder to use as the SD card.

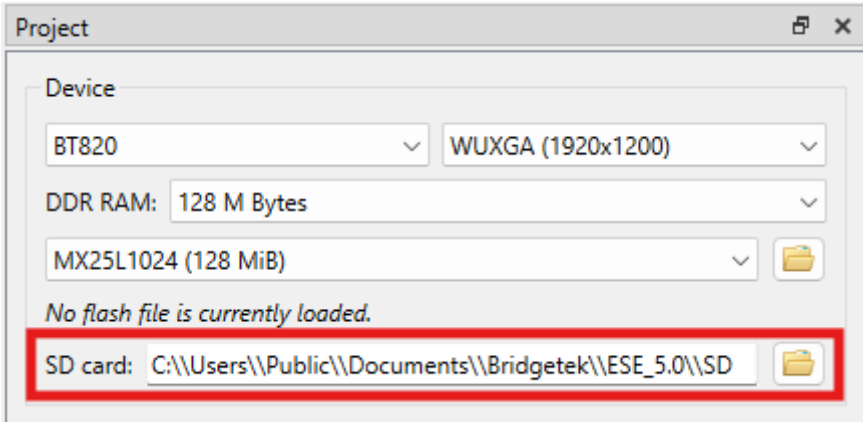


Figure 83 - SD card folder

***Note*:** The red text indicates that the folder cannot be inserted as an SD card. Try resetting the emulator or selecting another folder.

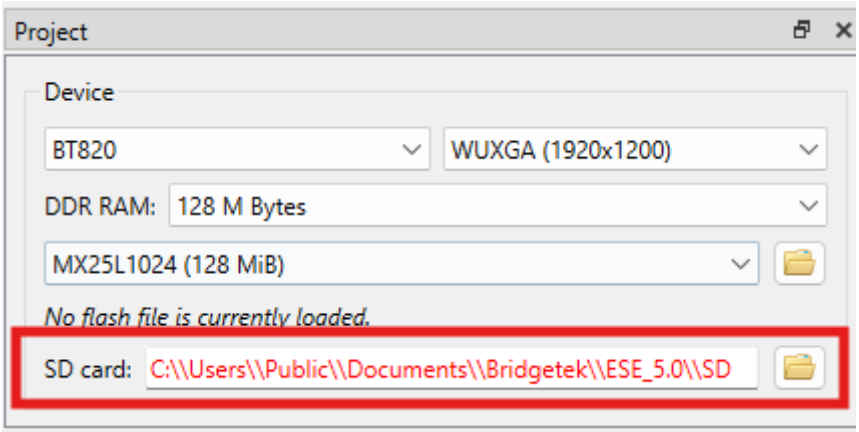


Figure 84 - Cannot insert SD card folder

I. Keyboard Shortcuts

The following keyboard shortcuts can be used in the Eve Screen Editor:

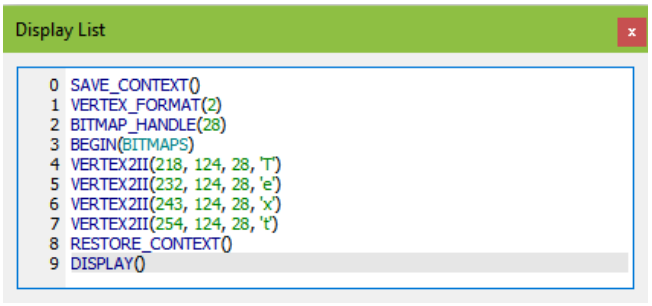
Item	Shortcut
New	*Ctrl + N*
Save	*Ctrl + S*
Undo	*Ctrl + U*
Redo	*Ctrl + Y*
Cut	*Ctrl + X*
Copy	*Ctrl + C*
Paste	*Ctrl + V*
Zoom In/Out of Viewport	*Ctrl + Mouse wheel*
Close Project	*Ctrl + F4*
Reset Emulator	*Ctrl + R*
Capture Display List	*Ctrl + D*
Open Recent Project 1->5	*Alt + 1, Alt + 2, …,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

IV.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 **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 Color

Subsequent drawing color can be changed by several methods:

- 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.
- Input the commands and then change directly in the editor.

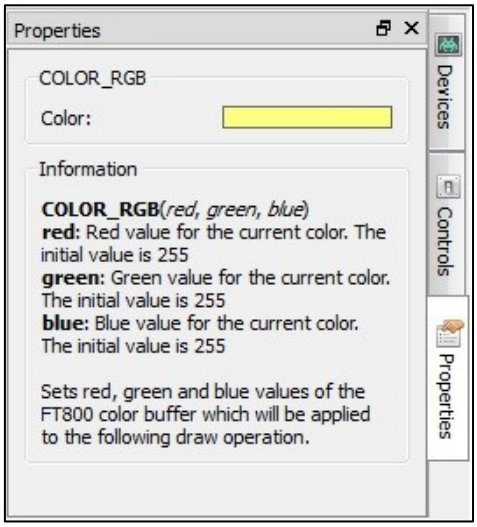


Figure 85 - Change Drawing Color

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

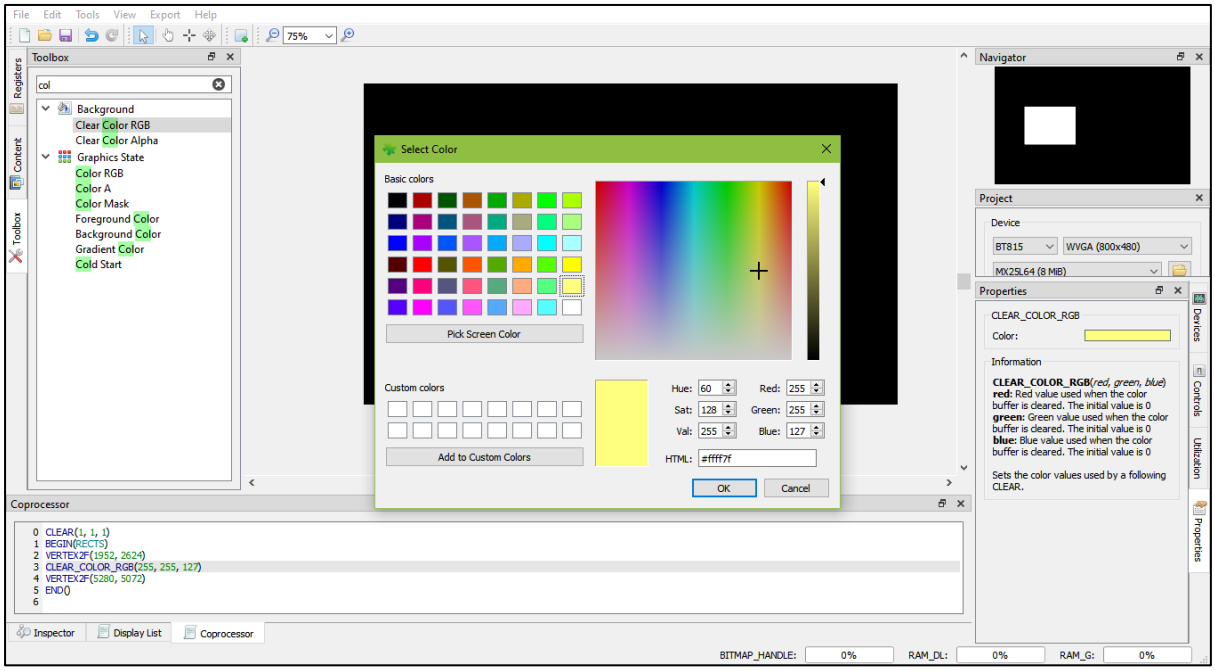


Figure 86 - Select Color

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 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 the Add Content section. The added data can be removed by selecting an entry and clicking **[Remove]** in the Content tab.

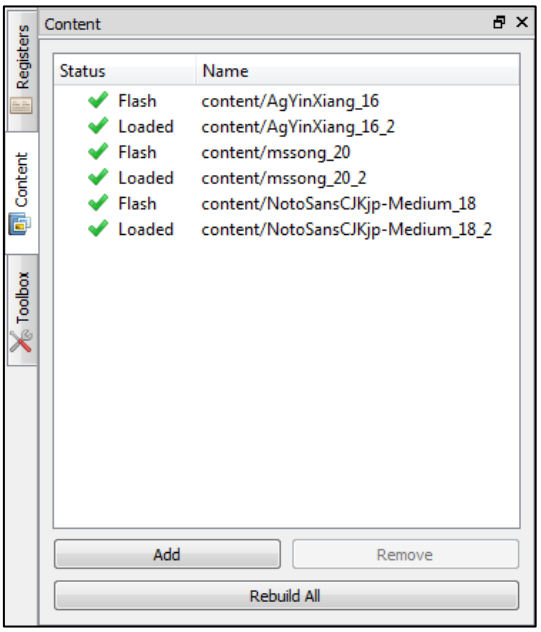


Figure 87 - Import Content

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

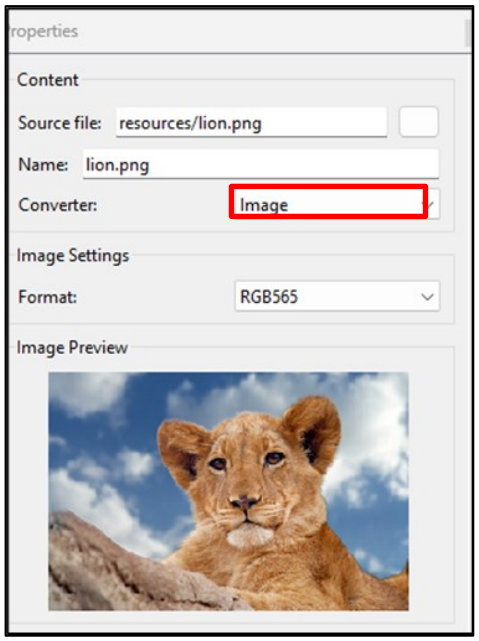


Figure 88 - Select Converter Image

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

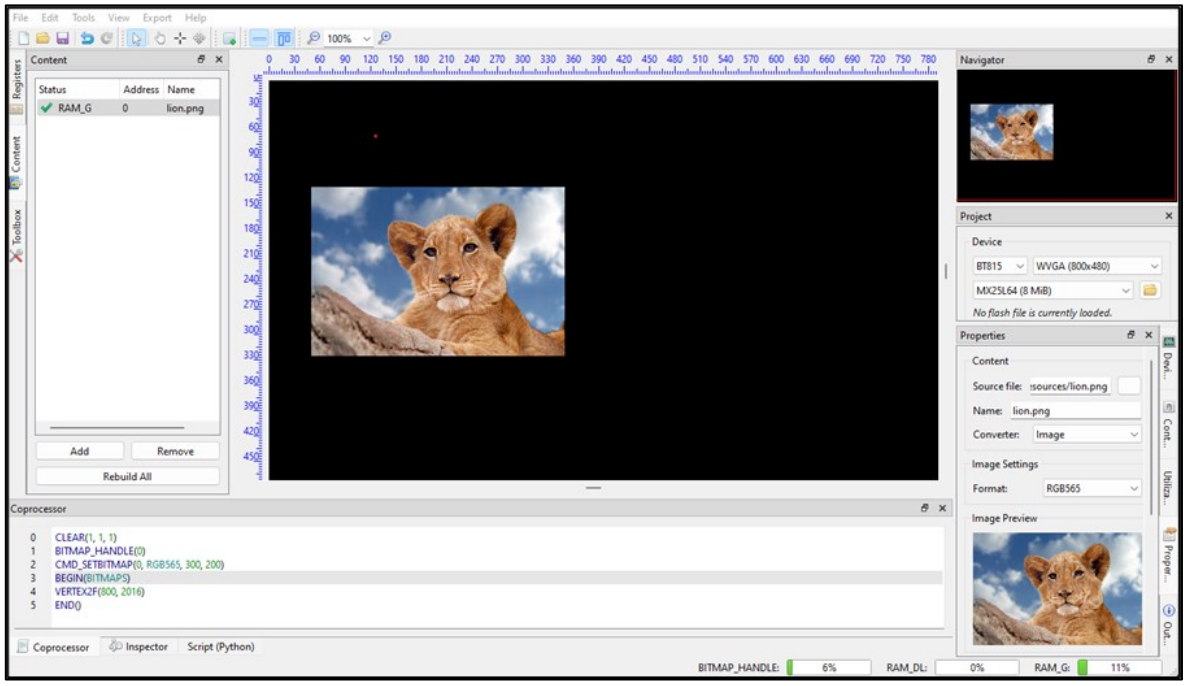


Figure 89 - Drag & Drop Image

D. Import 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 addresses can be used in the Display List and Co-Processor commands.

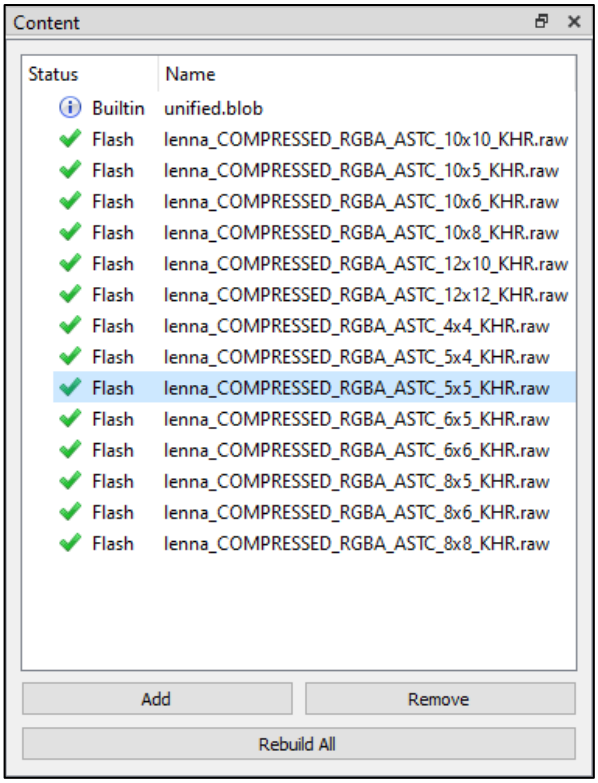


Figure 91 - Import Flash

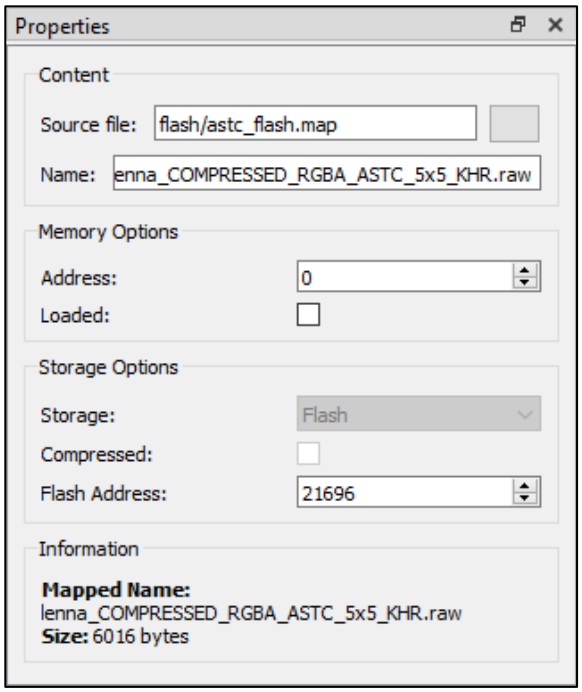


Figure 90 - Flash Address

E. Open a 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.

Note:

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

ESE Versions 3.X and higher are still able to open 1.X and 2.X project files with ".ft800proj" and ".ft8xxproj" extension names.

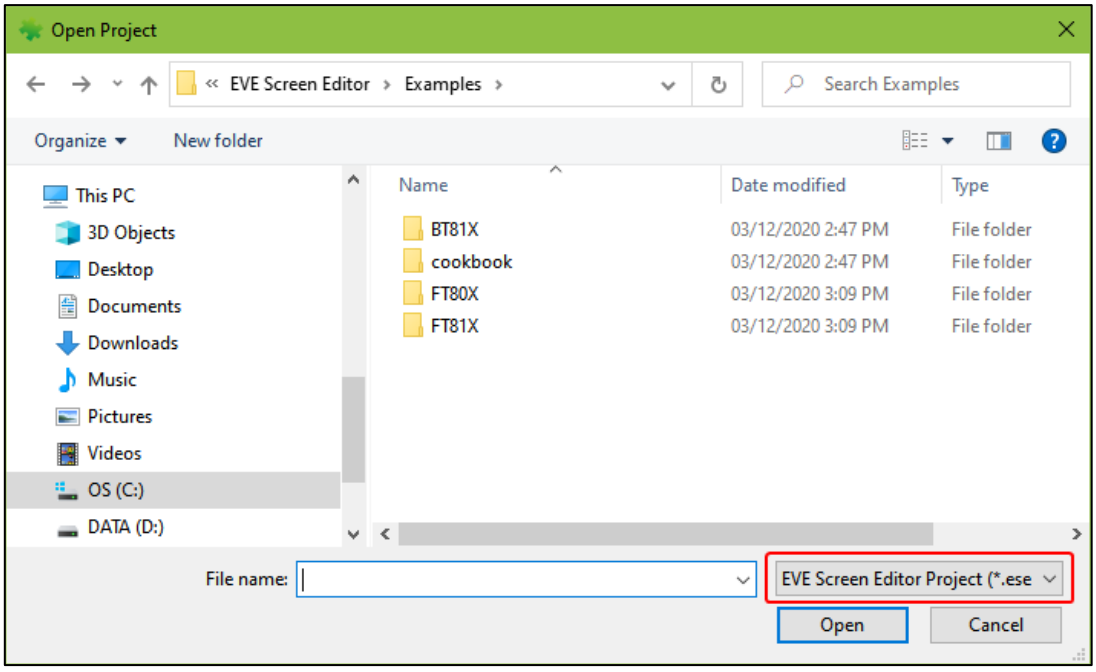


Figure 92 - Open Project

ESE Versions 4.4 and higher support opening a project using drag and drop events. Users can drag and drop a project folder or a project file (*.ese) into ESE to open that project.

F. Save Your Design

The current project can be saved by:

- clicking **[Save]** on the toolbar
- or selecting **File -> Save**
- or pressing *Ctrl+S* keyboard shortcut

Users can also save the current project as a different filename and/or in a different directory by selecting **File- > Save As**.

Please note saved project files have an extension of ***.ese***.

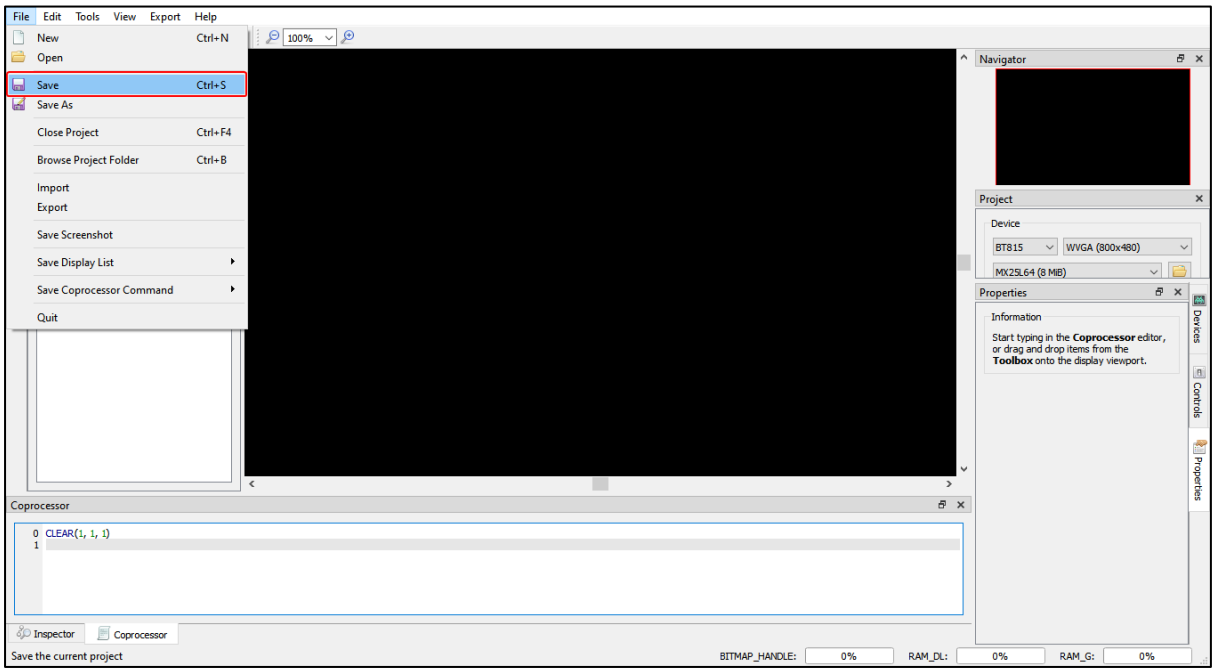


Figure 93 - Save a Project

G. Export a Project

Users can export their screen designs in various formats such as Gameduino 2 project, EVE Arduino project, EVE HAL2.0 project (C code based), and Raspberry Pi Pico project. 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 to use 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 **Export** menu in the menu bar.
2. Select the option to which the project is to be exported.

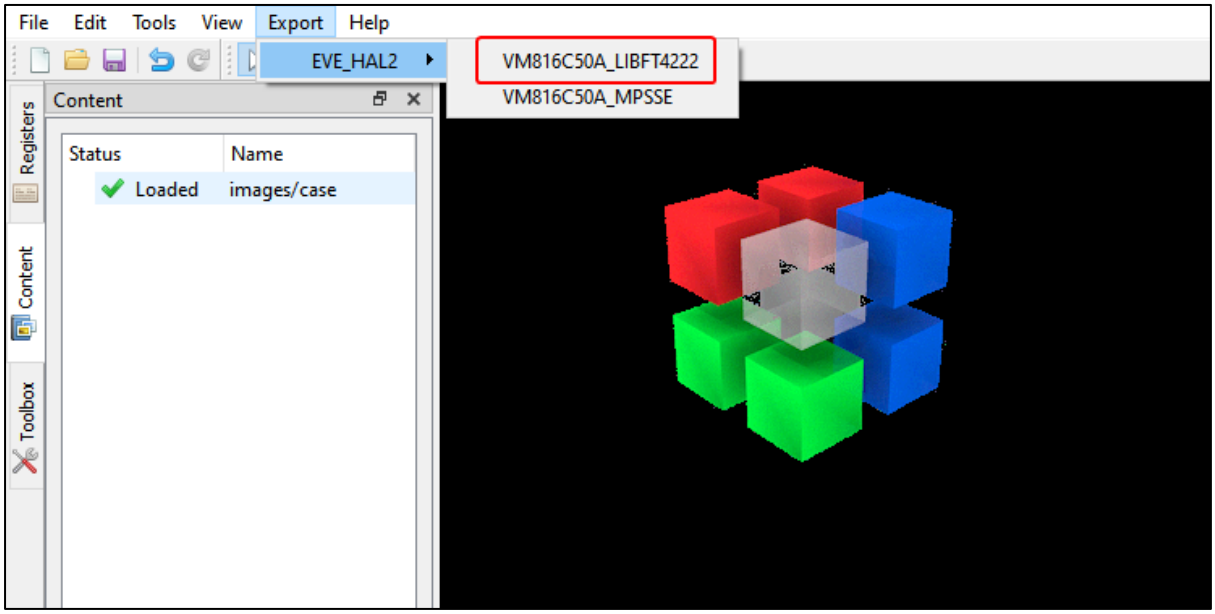


Figure 94 - Export Project

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. 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.

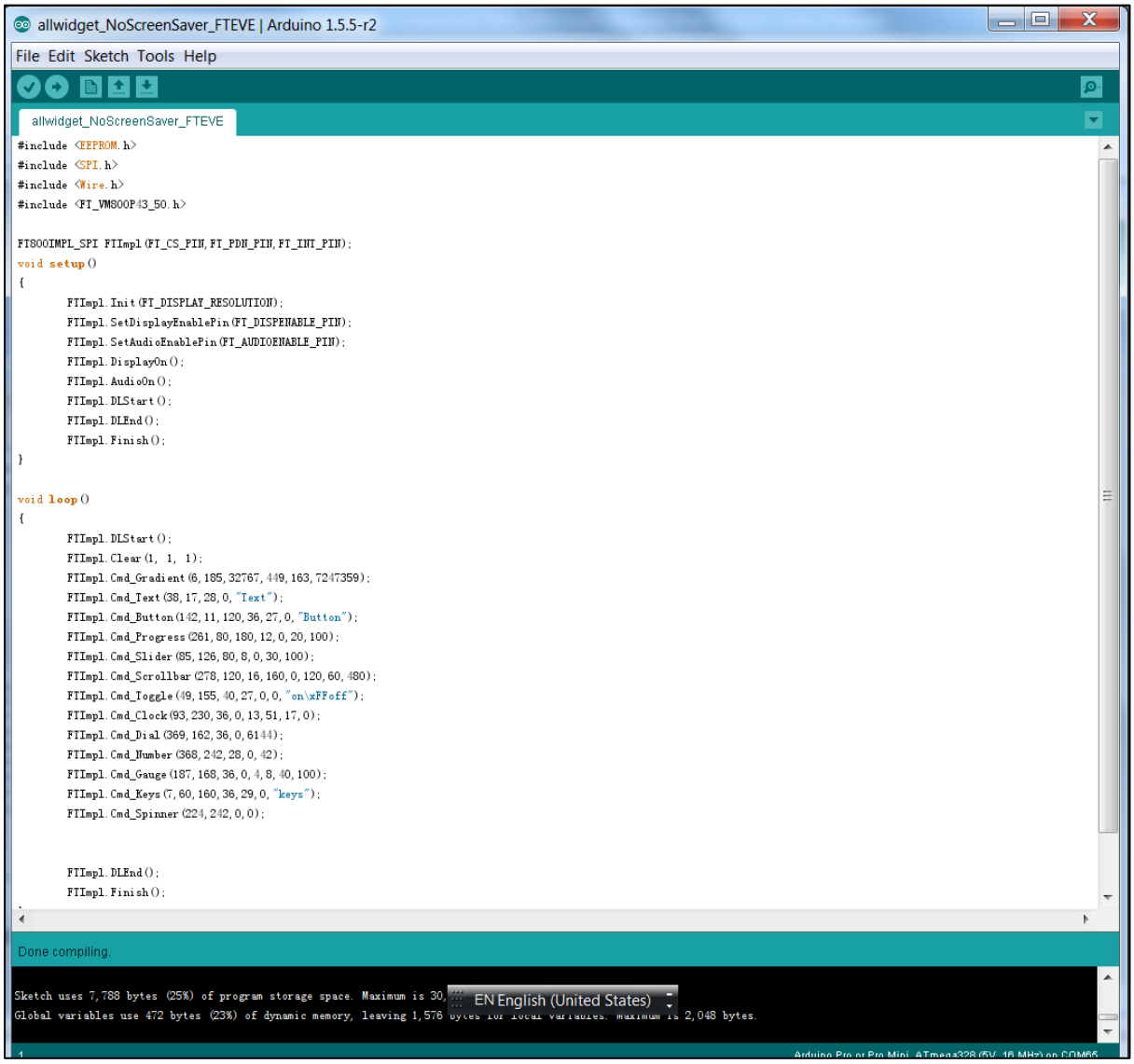


Figure 95 - Arduino IDE

For Raspberry Pi Pico projects, the project files will be generated in a separate folder which will contain two sub folders “assets,” “lib” and two files “code.py”, and “readme.md”.

Users need to read the “readme.md” file that explains each item as well as the circuitPython firmware version, hardware connection information, and some useful links.

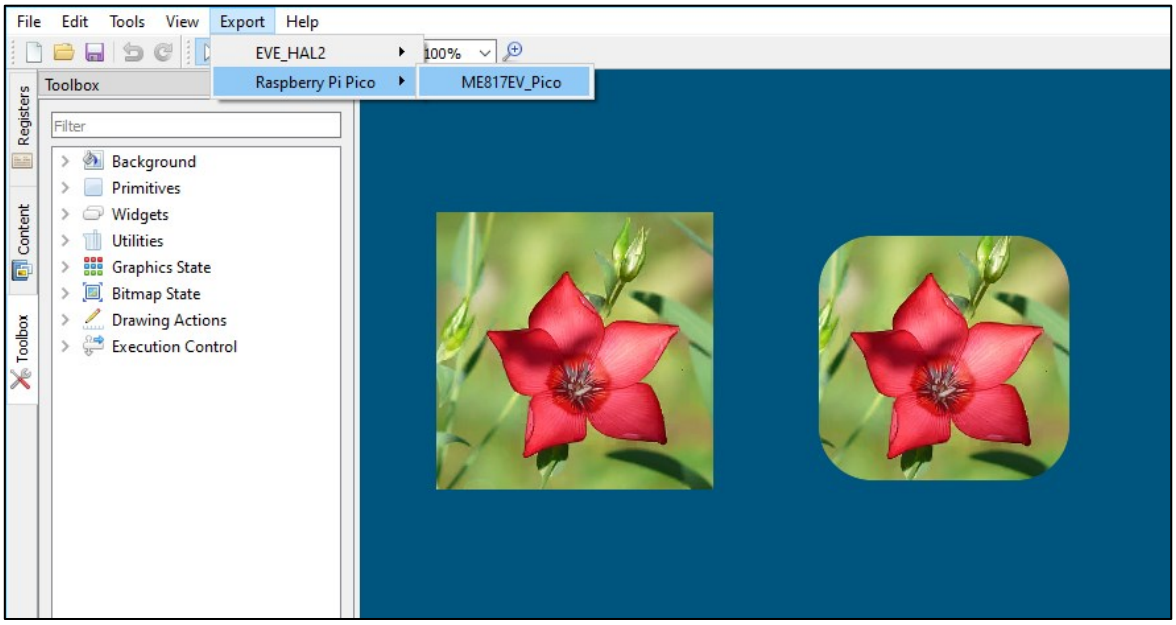


Figure 96 - Export Raspberry Pi Pico Project

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, refer to the following two figures and these steps:

1. Load the custom font in the Content manager. Successfully loaded fonts have the "Loaded" status indicated beside their font names.
2. Set the font format and size attributes.
3. Drag and drop the font into Viewport.
4. Click the font object in the Viewport to edit the commands.
5. "CMD_SETFONT" and "CMD_SETFONT2" are generated accordingly for FT80X/BT82X and FT81X/BT81X devices to assign the new custom fonts with one unused bitmap handle. By default, the first unused bitmap handle is zero.

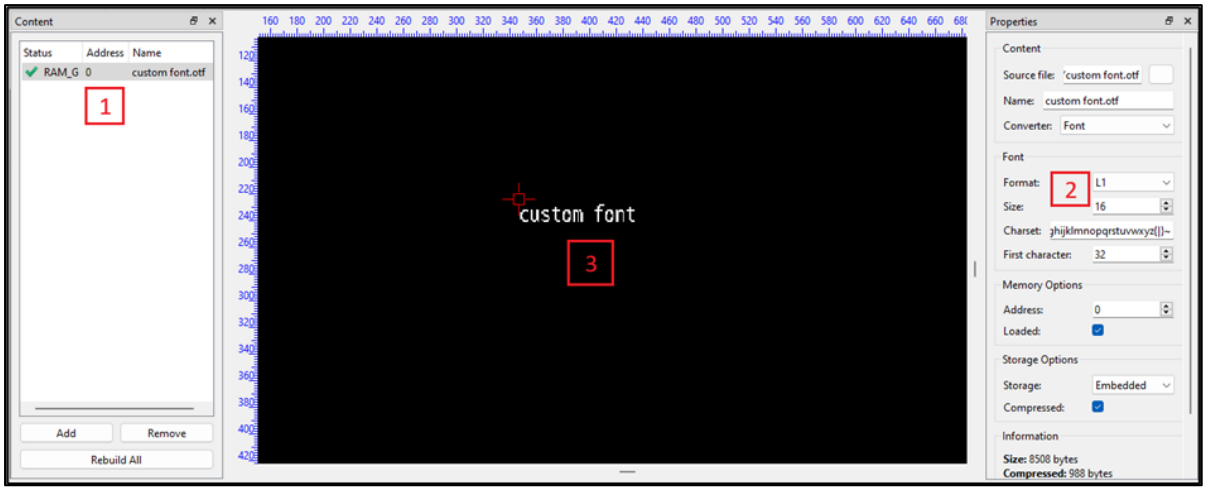


Figure 97 - Custom Font

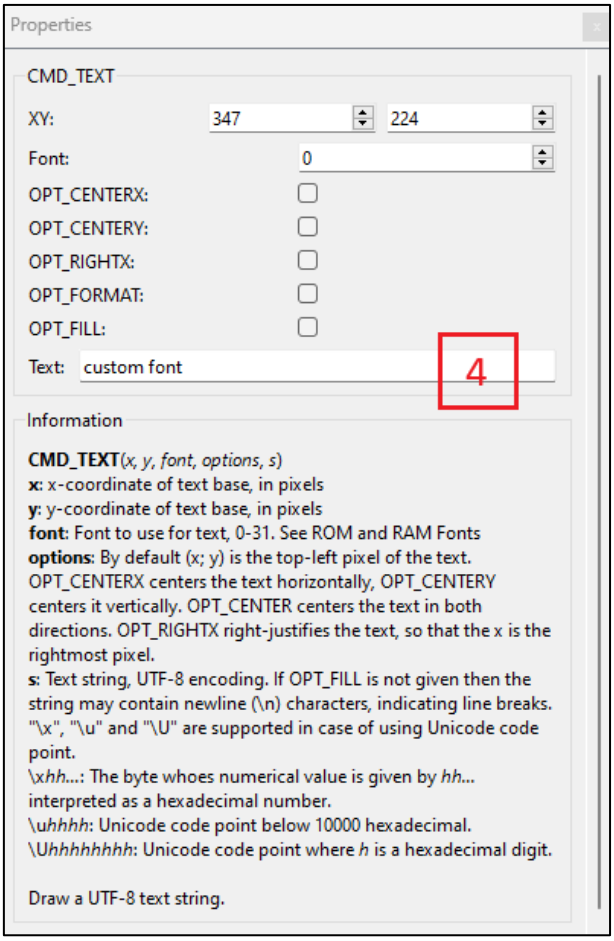


Figure 98 - "Text" Property of a Custom Font

I. Placement Assistance

1. Restrict Vertical Placement

1. Align two objects that should share the same x-coordinate alignment.
2. Hold down the **SHIFT** key and click the aligned object.
3. Move it vertically along the dashed red line.

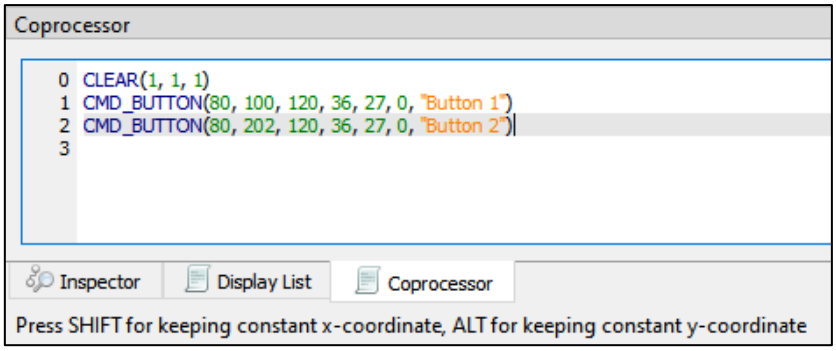


Figure 99 - Hold Down the SHIFT Key to Lock Vertical Movement

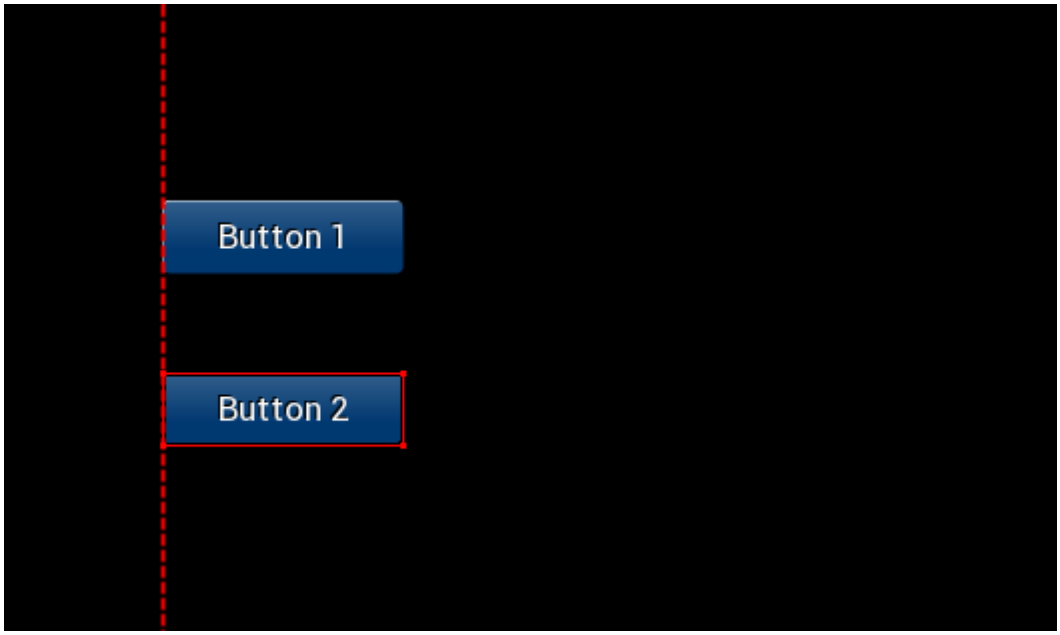


Figure 100 - Move the Selected Object Vertically Along the Dashed Red Line

2. Restrict Horizontal Placement

1. Align two objects that should share the same y-coordinate alignment.
2. Hold down the **ALT** key and click the aligned object.
3. Move it horizontally along the dashed red line.

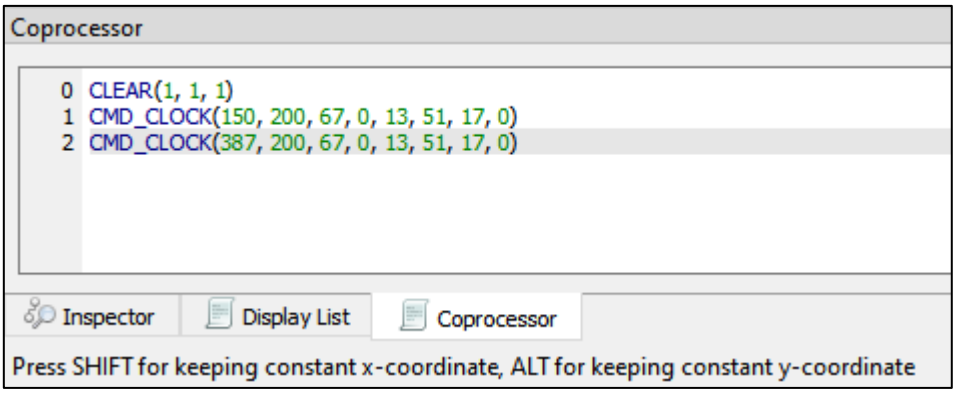


Figure 101 - Hold Down the ALT Key to Lock Horizontal Movement

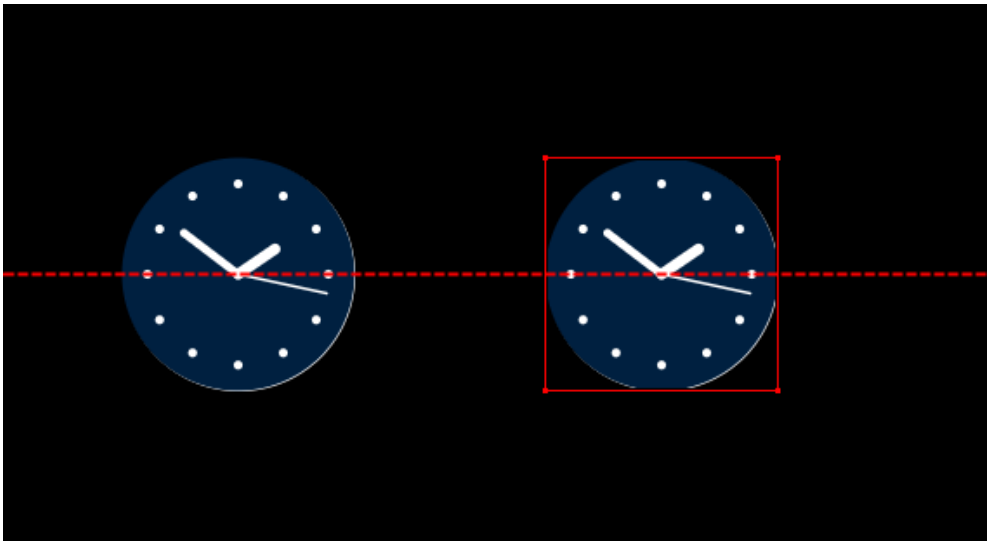


Figure 102 - Move the Selected Object Horizontally Along the Dashed Red Line

J. Automatically Detect and Load Content

When a user adds an asset from the list of supported files to the Content window, ESE will automatically read the corresponding file (.json/.readme) located in the same folder to gather essential information. This information will then be configured in the Properties window.

The supported files are listed as below:

- .ram_g (animation), .flash (animation)
- .raw (image), .raw (font legacy format)
- .glyph (font extended format), .xfont (font extended format)

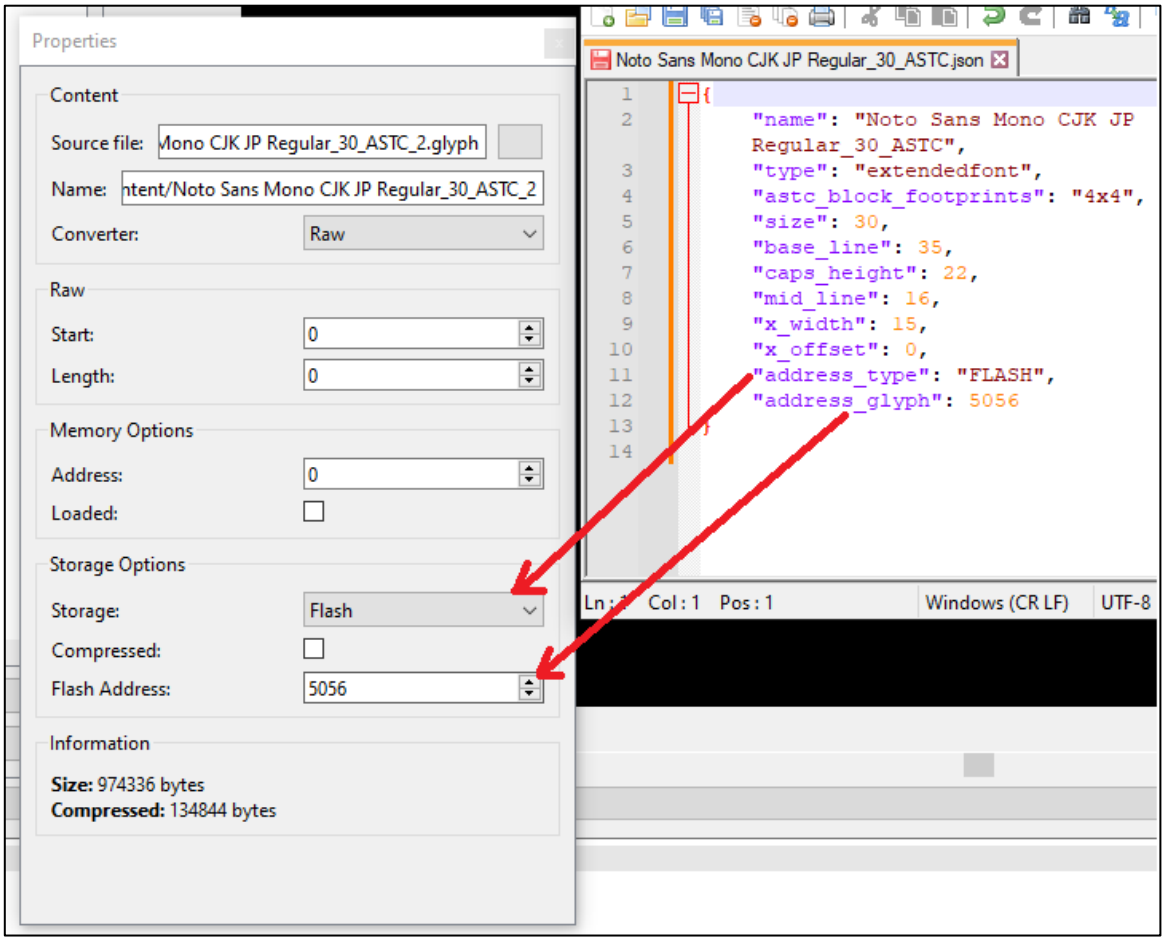


Figure 103 – Retrieve Meta Data of Asset

Note:

- For image with PALETTED format, ESE will try to add both the LUT file and the INDEX file.
- For font with Extended format: ESE 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 in the original folder structure generated by EAB tool.

K. Generate Coprocessor Commands for Assets

After the user has dragged and dropped an asset from the Content Manager into the viewport, ESE extracts the relevant information from the JSON file (located in the same directory). Subsequently, ESE generates the corresponding coprocessor commands to render them.

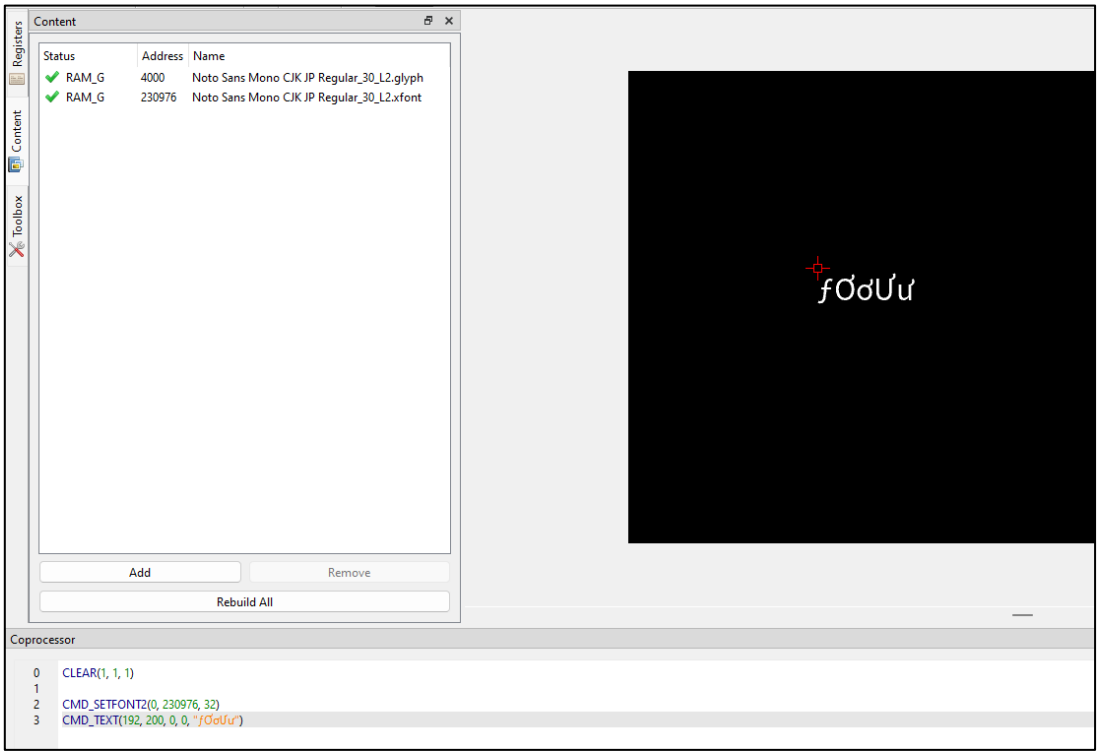


Figure 104 - Automatically Generated Coprocessor Commands

Supported assets:

- Video (.avi file)
- Bitmap (.raw file)
- Font (.raw or .xfont file)

L. Dotted Lines overlaid on the Viewport to assist in Aligning Graphics Objects

ESE displays dotted lines to help the user align a graphic item when its position is close to the vertical or horizontal edges of other graphic items.

Green: Vertically aligned.
Cyan: Horizontally aligned.
Red: Closely aligned in a vertical orientation.
Dark yellow: Closely aligned in a horizontal orientation.

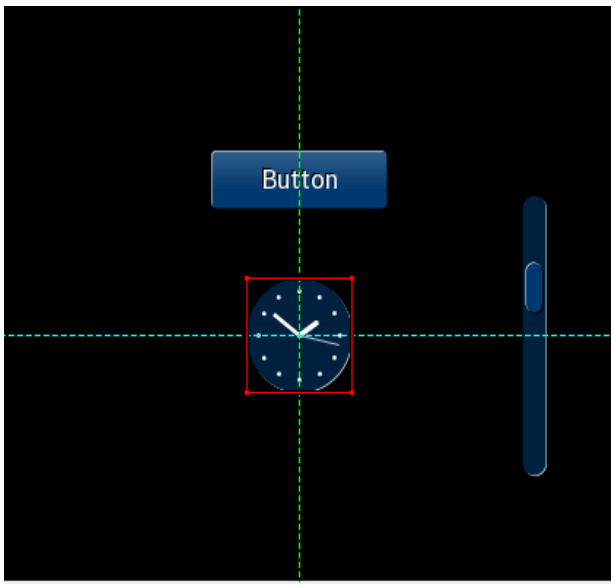


Figure 105 - Vertical Match and Horizontal Match

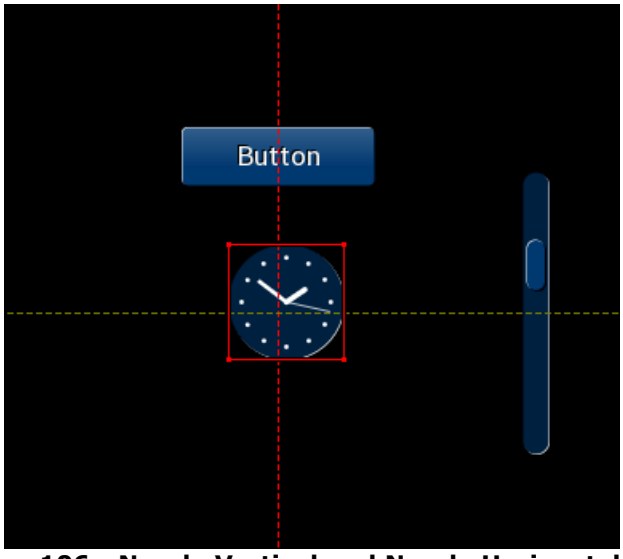


Figure 106 - Nearly Vertical and Nearly Horizontal

V. Example Project Explanation

ESE provides a set of example projects to showcase the different features of EVE. Users are encouraged to explore these example projects to learn more about the functionality of EVE as well as the usage of ESE. To browse the example projects, click the "Examples" button in the Welcome dialog box as below:

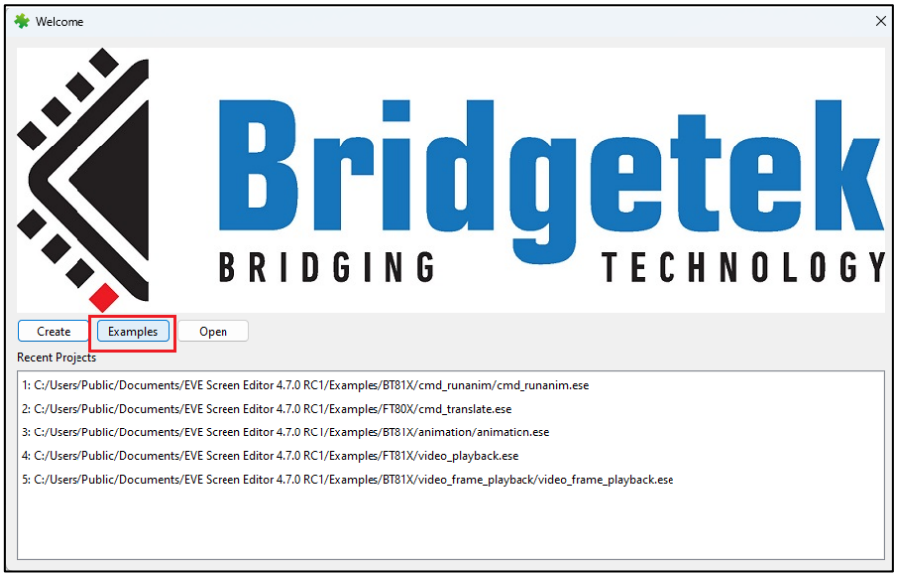


Figure 107 - Overview Dialog

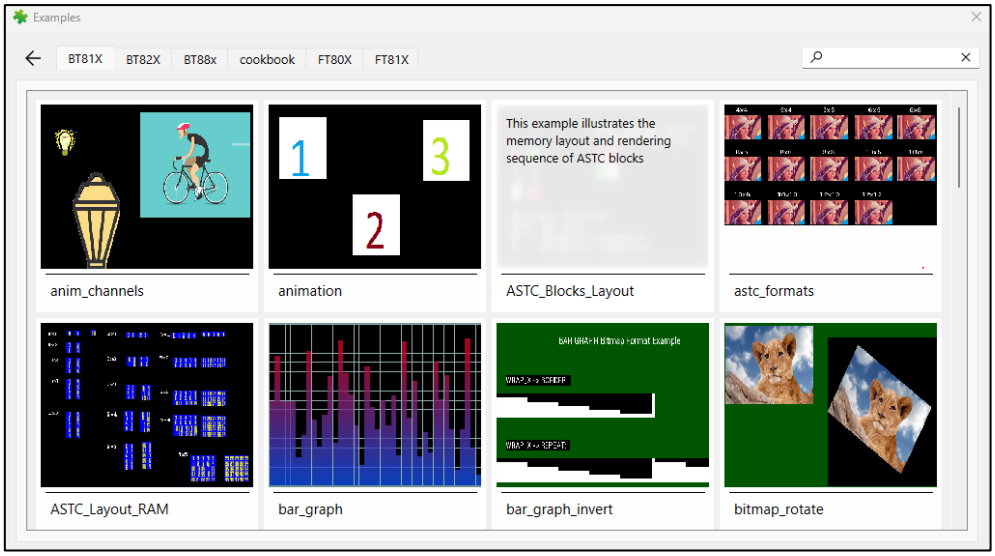


Figure 108 - Example Dialog

When the mouse hovers over the screenshot of an example project, its brief introduction will be shown. Click the screenshot and the example project will be launched.

In this chapter, several example projects have been carefully chosen to explain in more detail, which could help users understand more about the **EVE** chip and **ESE** tool.

A. ASTC format Example project

This example showcases the same image in different ASTC formats.

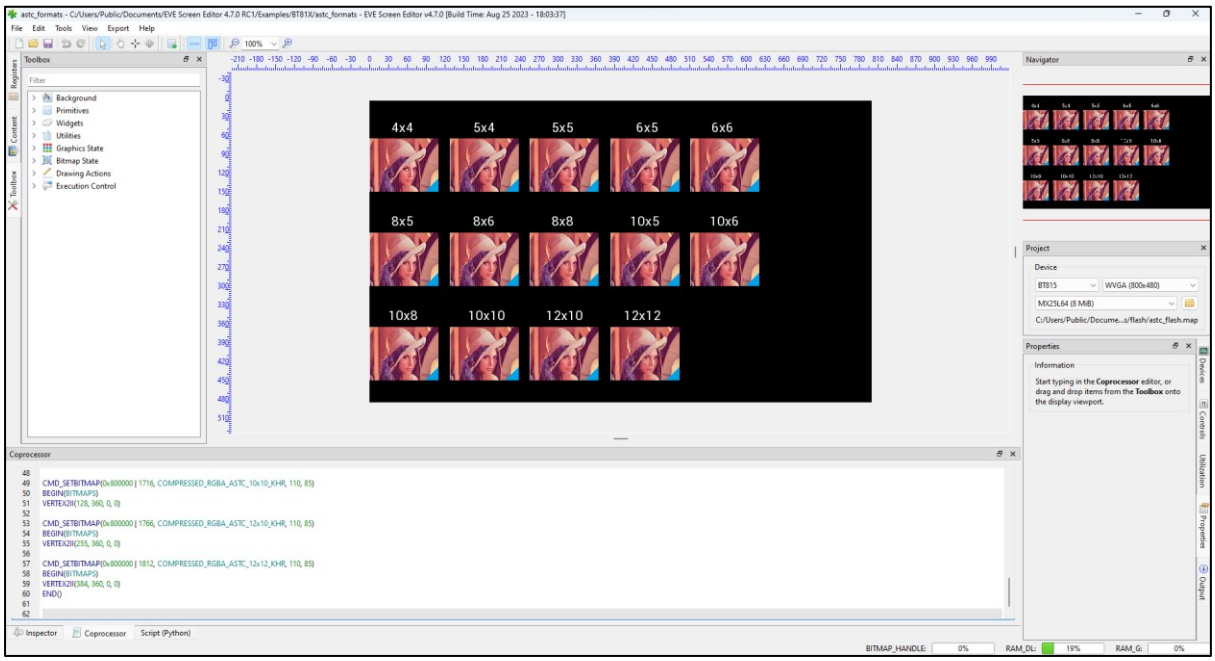


Figure 109 - ASTC Format Example Project

The example loads the mapped flash image to get a list of items as shown below.

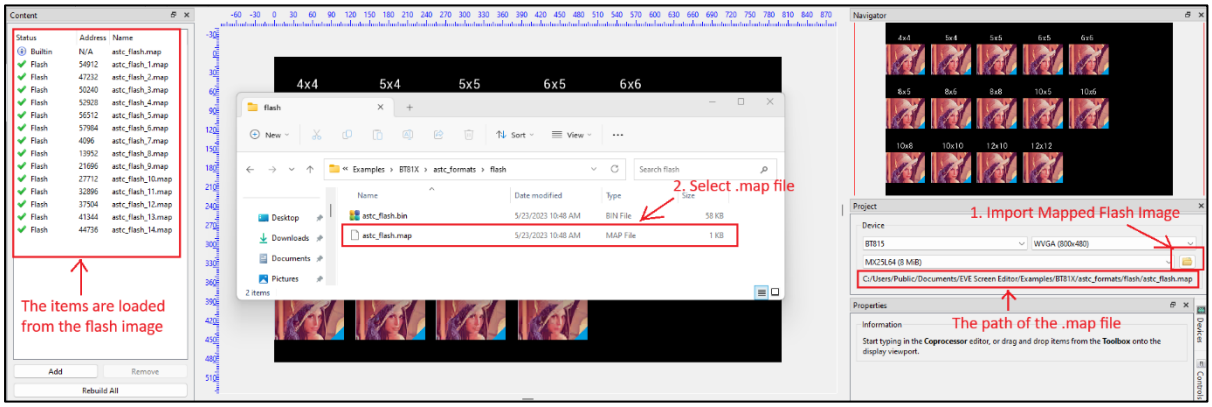


Figure 110 - Construction of ASTC Format Example Project

Based on the information of the content item, the example uses the appropriate commands to display the image.

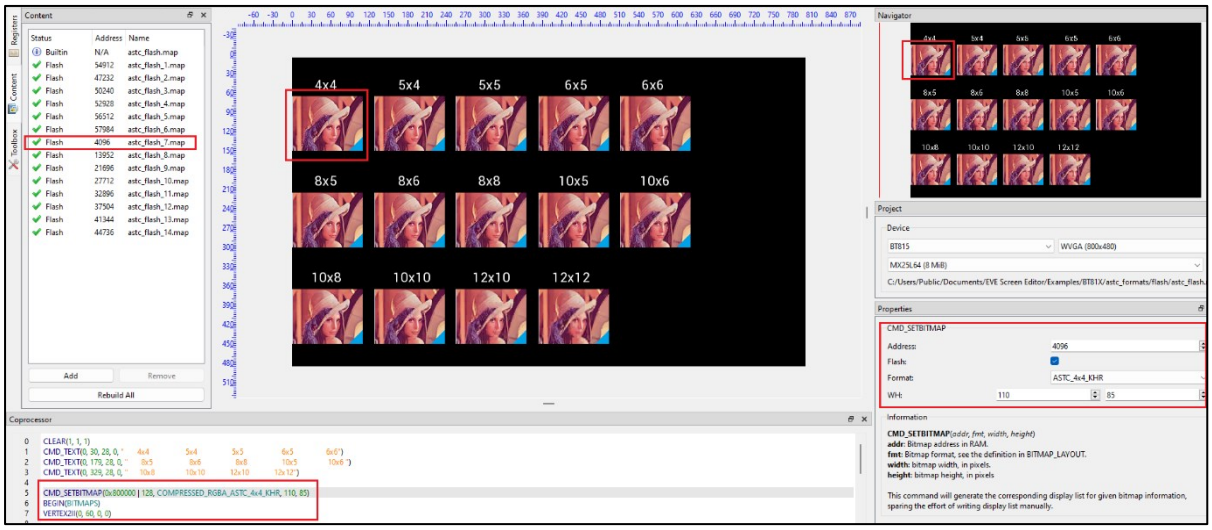


Figure 111 - Use Commands and Assets to Display Content with ASTC format

B. Setfont2 Example Project

This example displays the usage of cmd_setfont2.

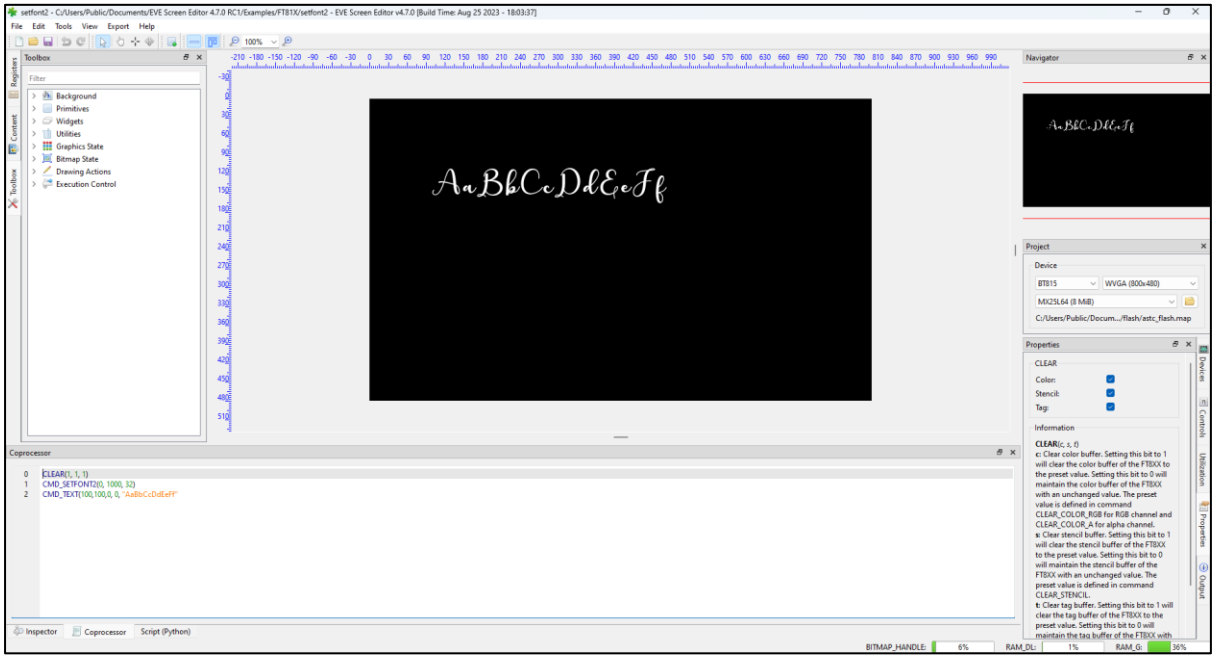


Figure 112 - Setfont2 Example Project

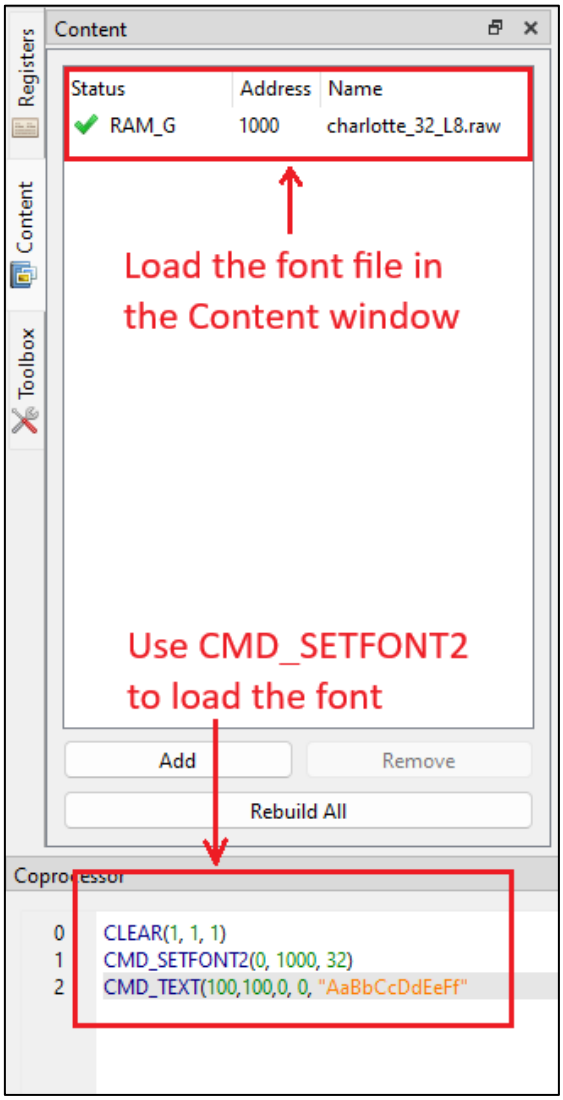


Figure 113 - Use Commands and Assets to Display Font

C. Transparent Button Group Example Project

This example displays a group of transparent buttons.

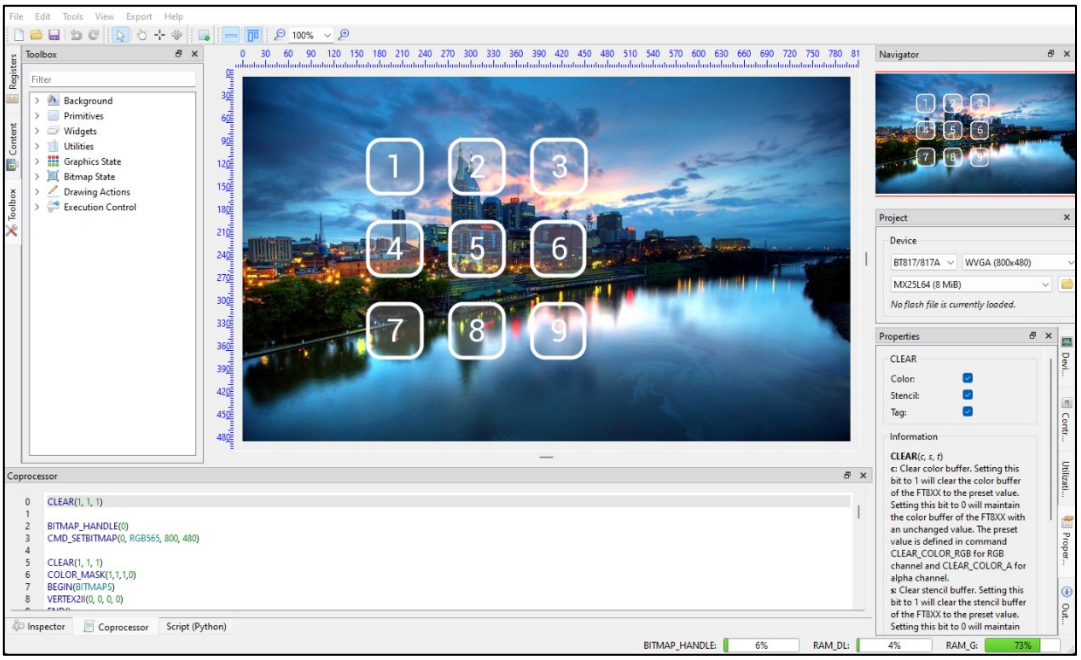


Figure 114 - Transparent Button Group Example Project

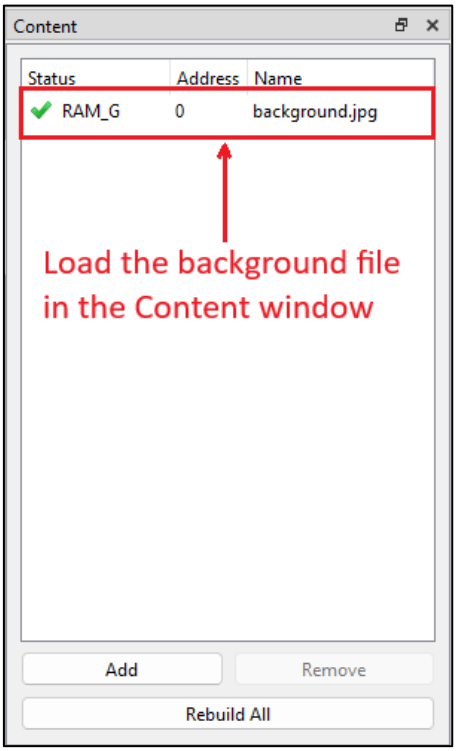


Figure 115 - Construction of Transparent Button Group Example Project

```
CLEAR(1, 1, 1)

BITMAP_HANDLE(0)
CMD_SETBITMAP(0, RGB565, 800, 480)

CLEAR(1, 1, 1)
COLOR_MASK(1,1,1,0)
BEGIN(BITMAPS)
VERTEX2II(0, 0, 0, 0)
END()
```

Figure 116 - Create Background

```
BLEND_FUNC(ONE, ONE_MINUS_SRC_ALPHA)
BEGIN(RECTS)

LINE_WIDTH(320)
//button 1
VERTEX2II(182, 100,0,0)
VERTEX2II(218, 136,0,0)
//button 2
VERTEX2II(290, 100,0,0)
VERTEX2II(326, 136,0,0)
//button 3
VERTEX2II(398, 100,0,0)
VERTEX2II(434, 136,0,0)
//button 4
VERTEX2II(182, 208,0,0)
VERTEX2II(218, 244,0,0)
//button 5
VERTEX2II(290, 208,0,0)
VERTEX2II(326, 244,0,0)
//button 6
VERTEX2II(398, 208,0,0)
VERTEX2II(432, 244,0,0)
//button 7
VERTEX2II(182, 316,0,0)
VERTEX2II(218, 352,0,0)
//button 8
VERTEX2II(290, 316,0,0)
VERTEX2II(326, 352,0,0)
//button 9
VERTEX2II(398, 316,0,0)
VERTEX2II(432, 352,0,0)

BLEND_FUNC(ZERO, ONE_MINUS_SRC_ALPHA)
COLOR_A(200)
COLOR_A(200)
LINE_WIDTH(240)
```

Figure 117 - Create Button Background

```
BLEND_FUNC(ONE, ONE_MINUS_SRC_ALPHA)
CMD_NUMBER(200,118,31, OPT_CENTER, 1)
CMD_NUMBER(308,118,31, OPT_CENTER, 2)
CMD_NUMBER(416,118,31, OPT_CENTER, 3)
CMD_NUMBER(200,226,31, OPT_CENTER, 4)
CMD_NUMBER(308,226,31, OPT_CENTER, 5)
CMD_NUMBER(416,226,31, OPT_CENTER, 6)
CMD_NUMBER(200,334,31, OPT_CENTER, 7)
CMD_NUMBER(308,334,31, OPT_CENTER, 8)
CMD_NUMBER(416,334,31, OPT_CENTER, 9)

//buttons }

COIOR_MASK(',1,1,1)
COLOR_RGB(255,255,255)
BLEND_FUNC(DST_ALPHA, ONE_MINUS_DST_ALPHA)
BEGIN(RECTS)
VERTEX2II(0,0,0,0)
VERTEX2F(12800, 7680) //800,480
```

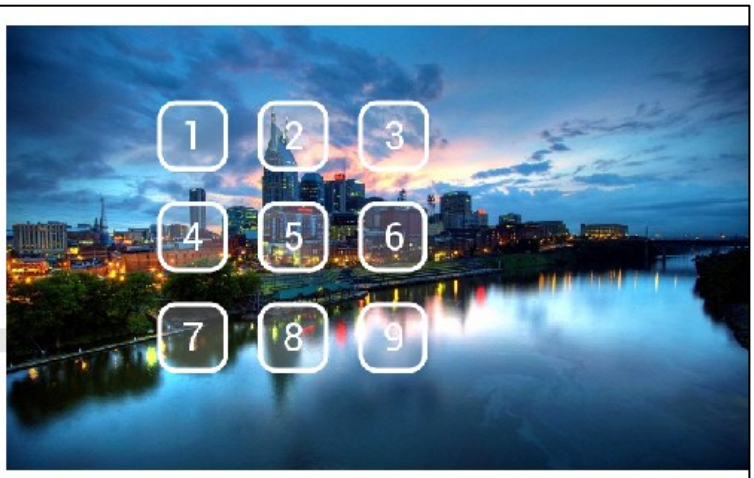


Figure 118 - Complete the Example

D. Video Frame Playback Example Project

This example displays how to render the video frame by frame.

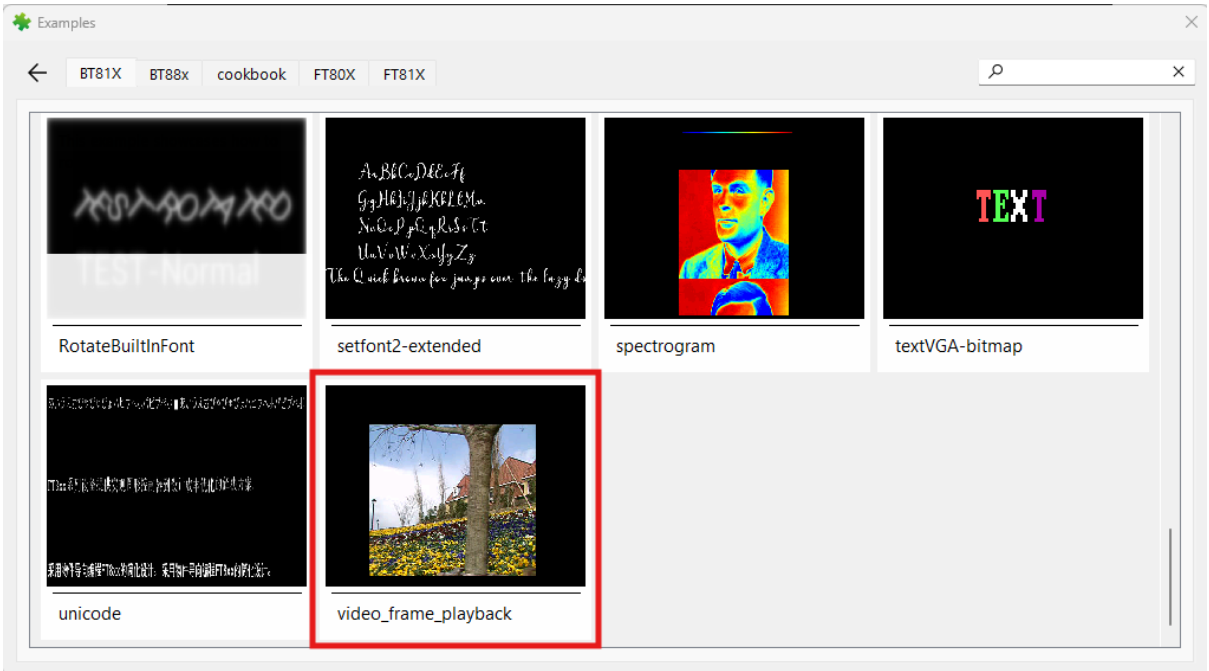


Figure 119 - Project thumbnail in the Examples window

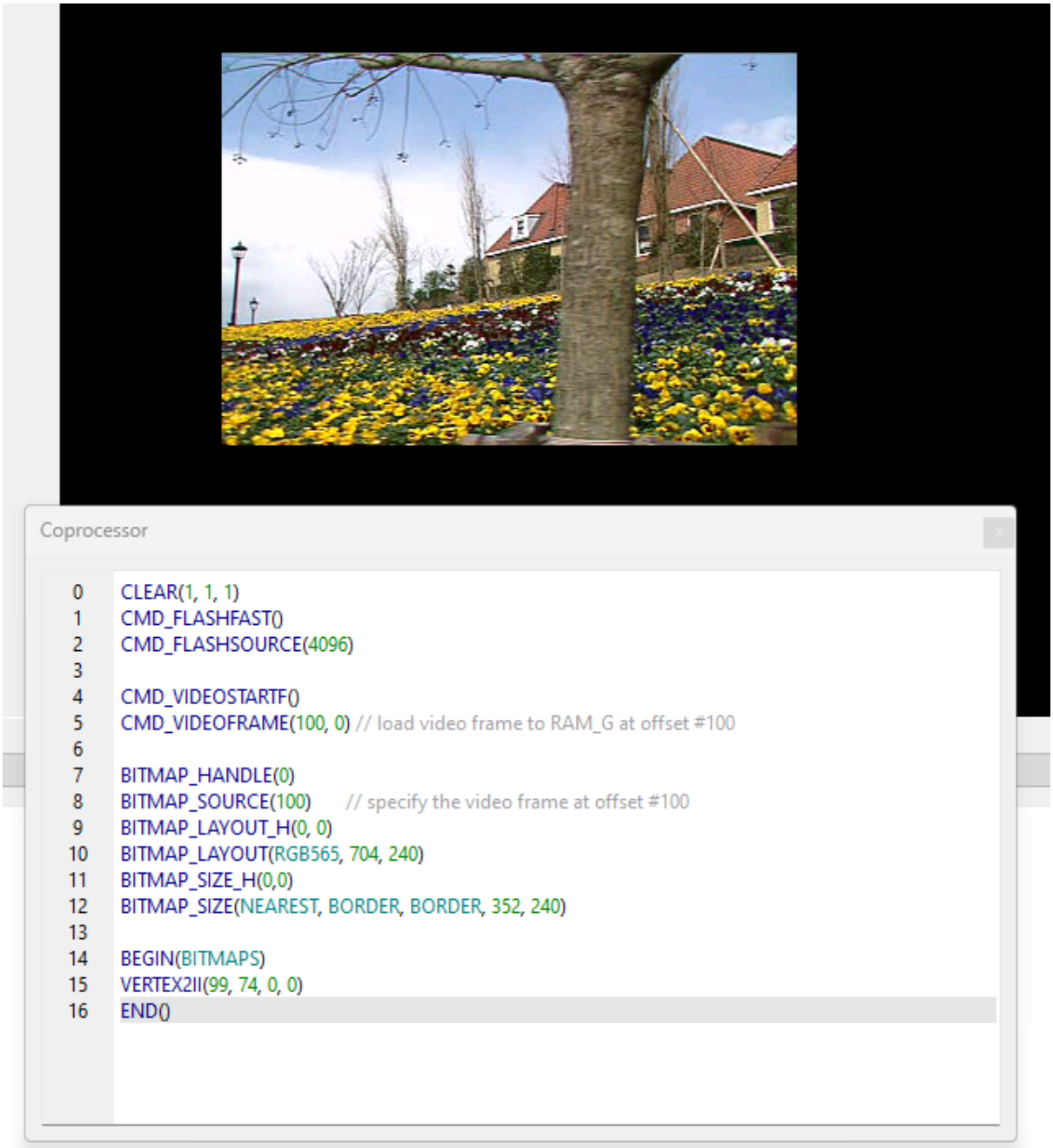


Figure 120 - Coprocessor commands to display a frame

This project includes examples that demonstrate how to display multiple frames. Choose 'Format' to select the type you desire.

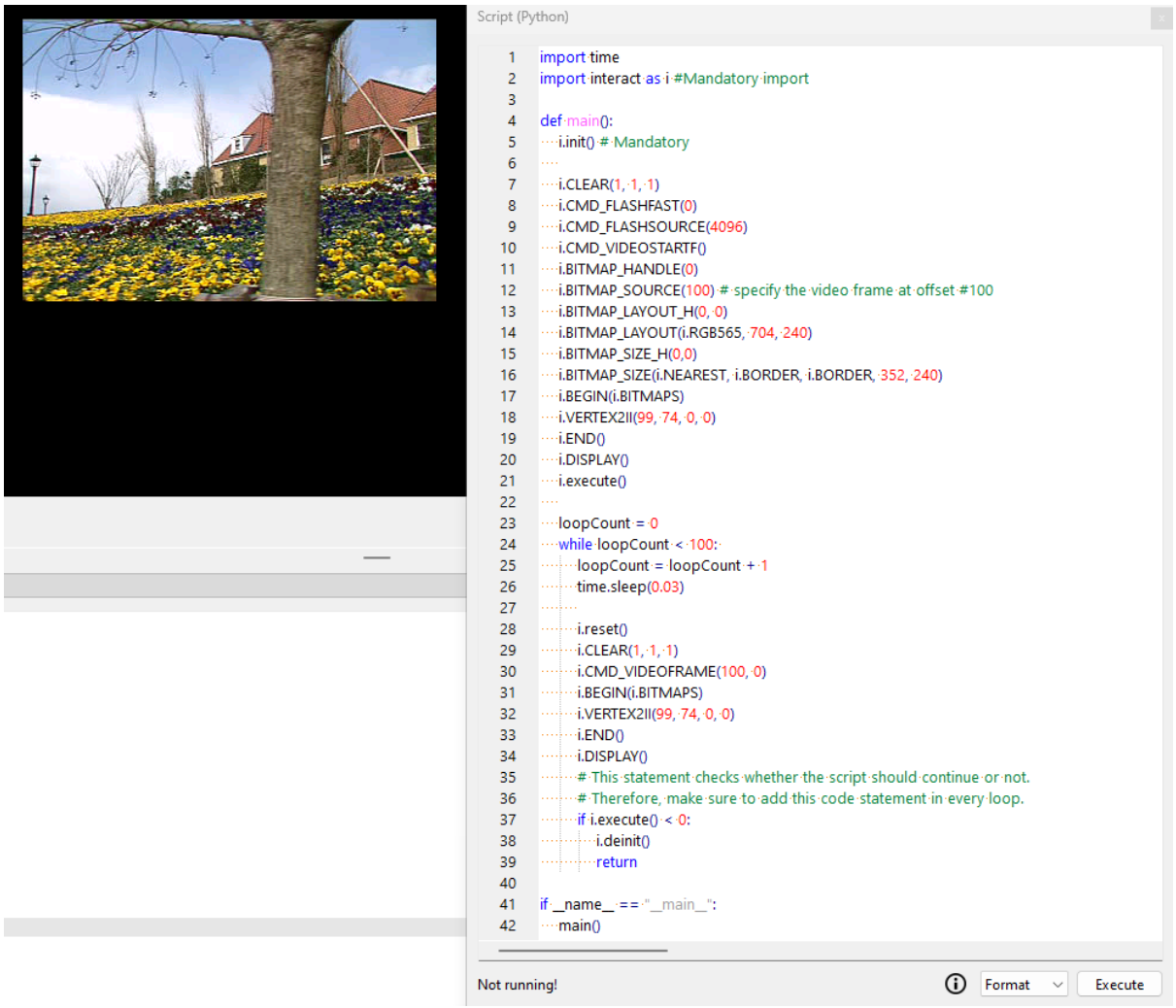


Figure 121 - Example of a Script editor displaying multiple frames

VI.Working with ESE

This section provides information about the usability of 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.

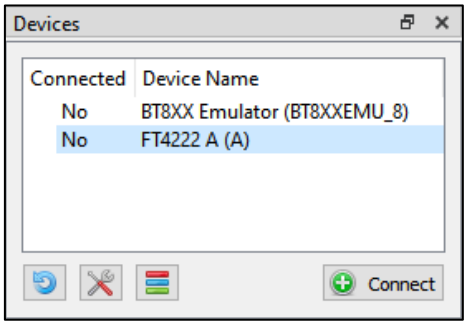


Figure 122 - Connect Device

Upon clicking **[Connect]** button, 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.

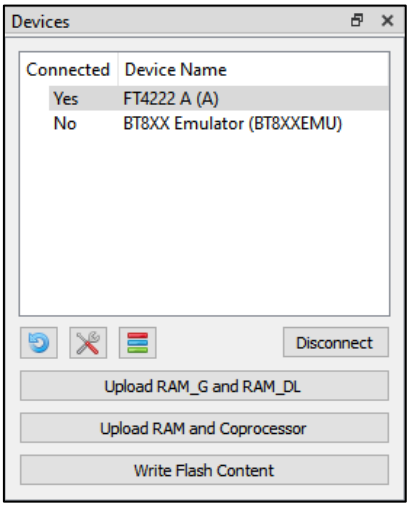


Figure 123 - Device Connected

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 while **[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). See the figure below.

Note 2: In case there are content items currently stored in flash, it is better to **[Write Flash Content]** before any update.

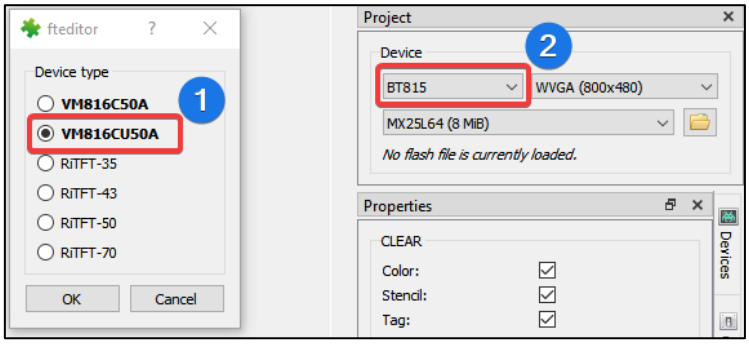


Figure 124 - Select the Correct Device Type

Managing Device

There are two kinds of devices:

- *Built-in Device:* Users can examine and clone these devices. They cannot be modified.
- *Custom Device:* Users can Add/Edit/Clone/Remove a device.

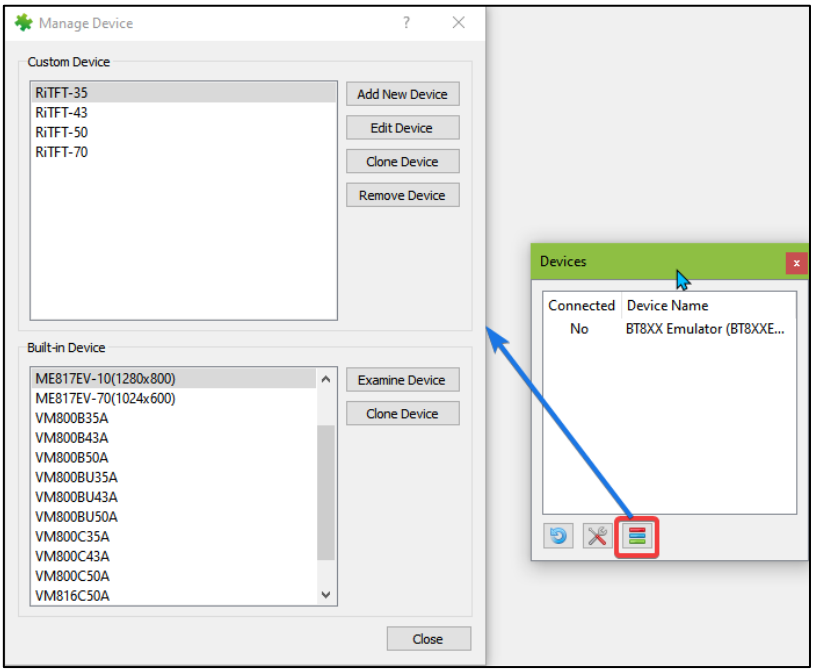


Figure 125 - Managing Device

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 input box will execute the specified steps and highlight the specific display list or co-processor commands.

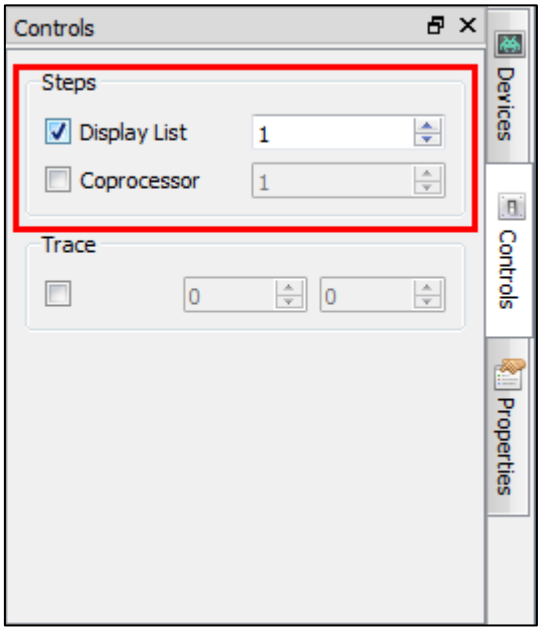


Figure 126 - Select Display List/Coprocessor

Refer to the figure below. The 2nd display list command is highlighted by a yellow line but there is nothing drawn in the viewport because the **VERTEX2II** command has not been executed yet.

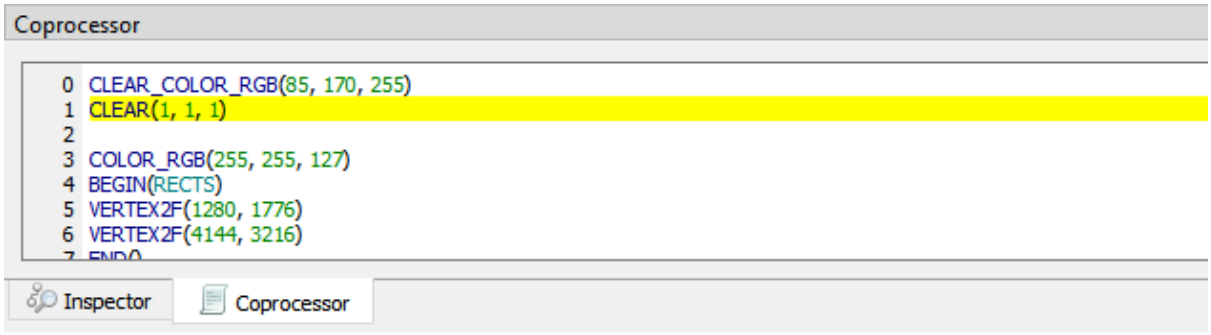


Figure 127 - Display List Command is Highlighted in Yellow

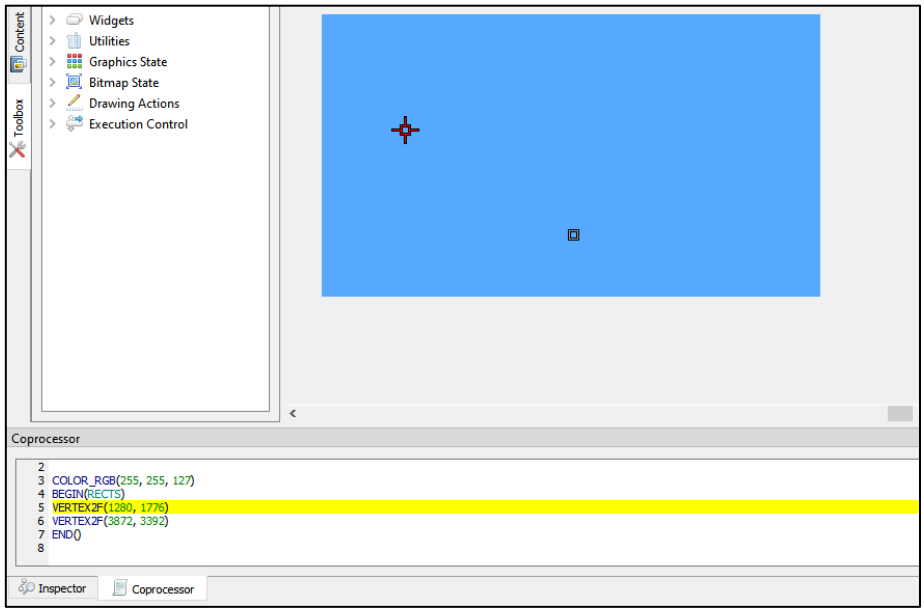


Figure 128 - Prepare to Draw a Rectangle

If the user increases the value in the Display List input textbox, the highlighted line will be moved, and the effect of the drawing is shown below:

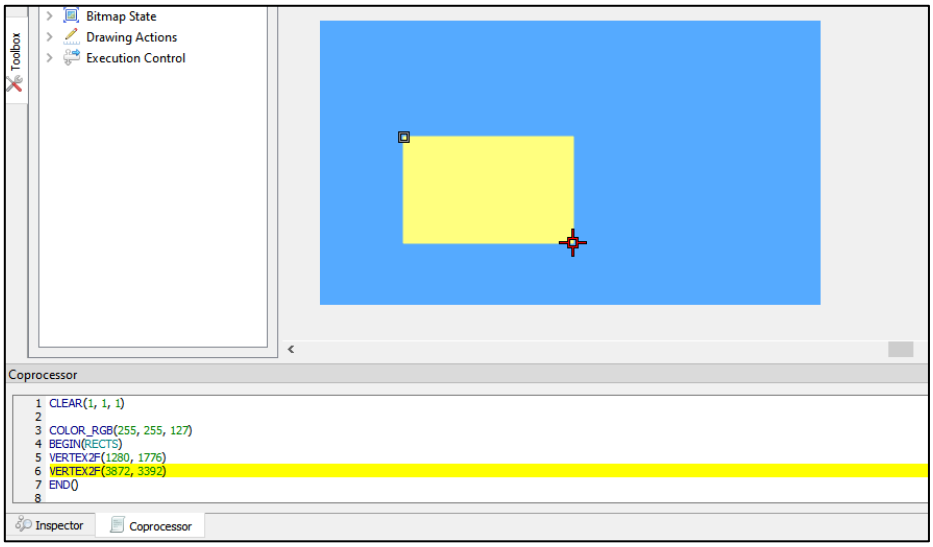


Figure 129 – Draw Rectangle

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** instead of those under it.

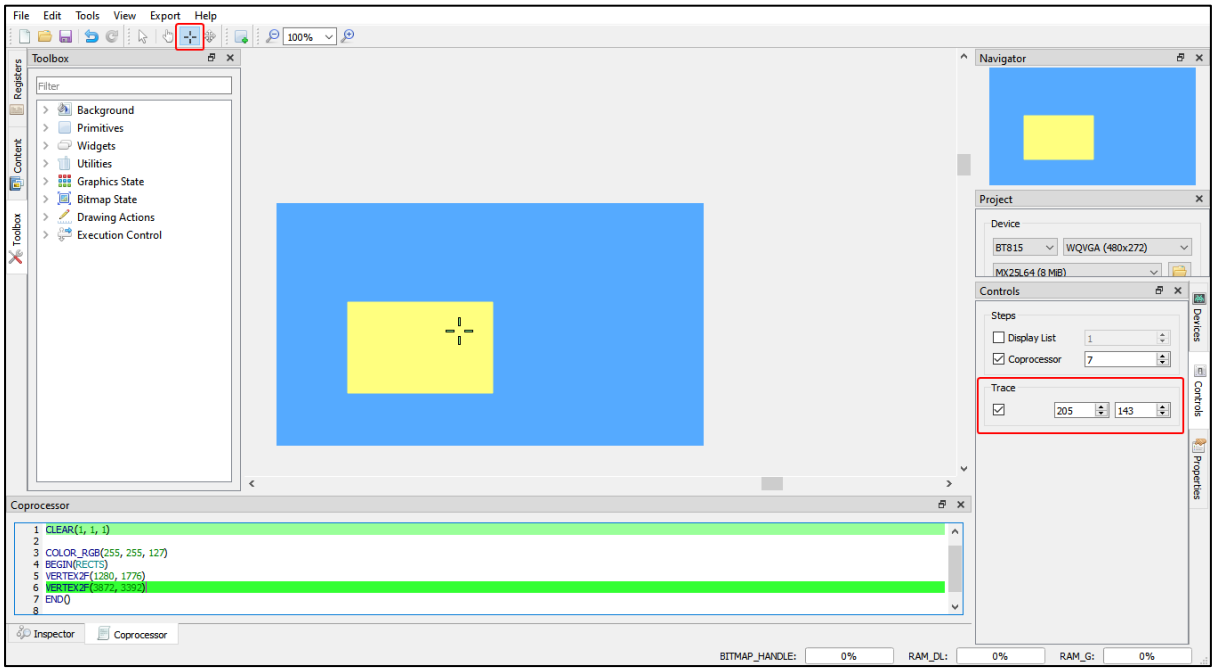


Figure 130 - Trace the Pixel

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", "BT82X", "BT88X", "cookbook", "FT80X", "FT81X", and "scripts"* for specific example projects:

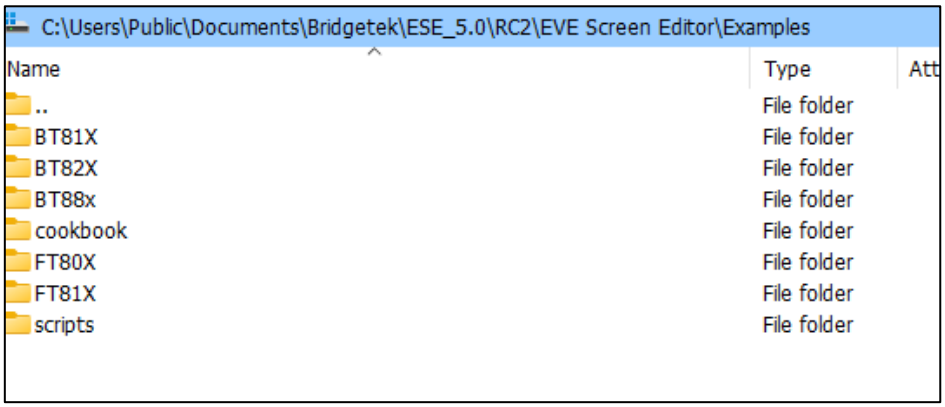


Figure 131 - Examples

Open "allwidget_NoScreenSaver" project under FT80X folder and it will show:

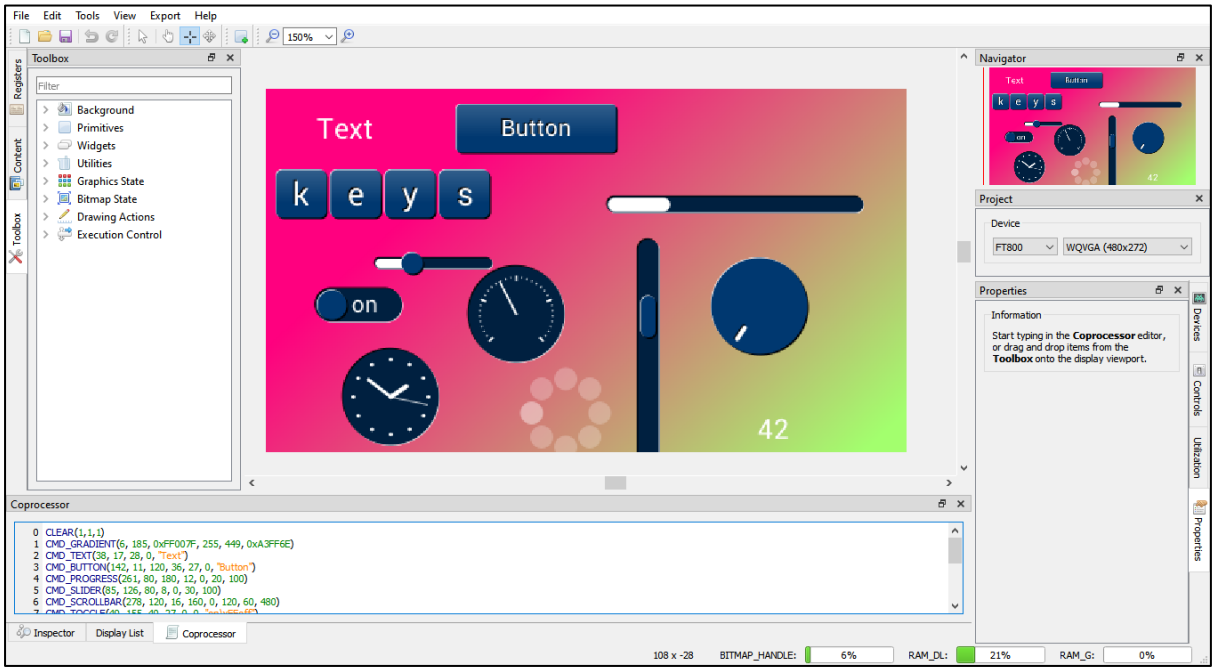


Figure 132 - Open an Example Project

D. Add Software Library for Exporting Projects

You can expand the functionality of **ESE** by including an additional Python file to integrate your custom software library for your specific 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"**

Note 1: If the device type is labeled with a generic name, the modules will be visible to all models falling under that type. For instance, if the **<device_type>** is configured as "**bt88x**," selecting "**bt881**" in the device type section of the project configuration window in ESE will also reveal the export menu. Please make sure to specify the chipset name as the device type when working with a particular chipset.

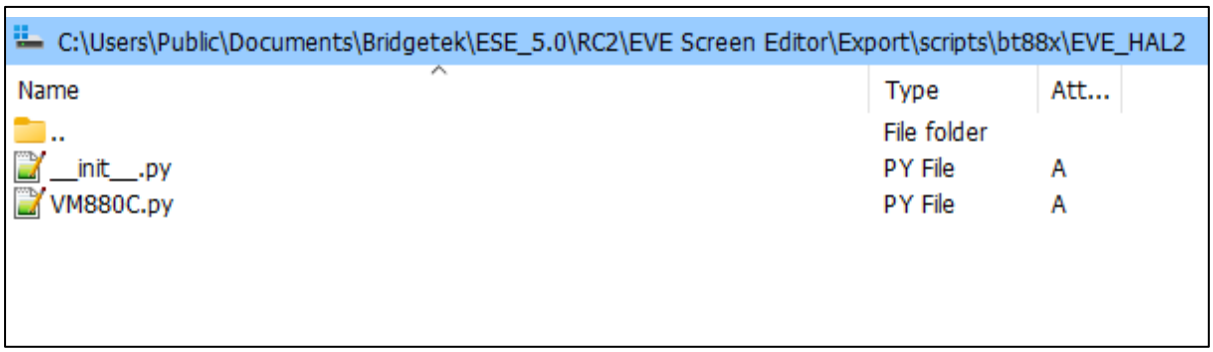


Figure 133 - Directory structure for a software library containing modules

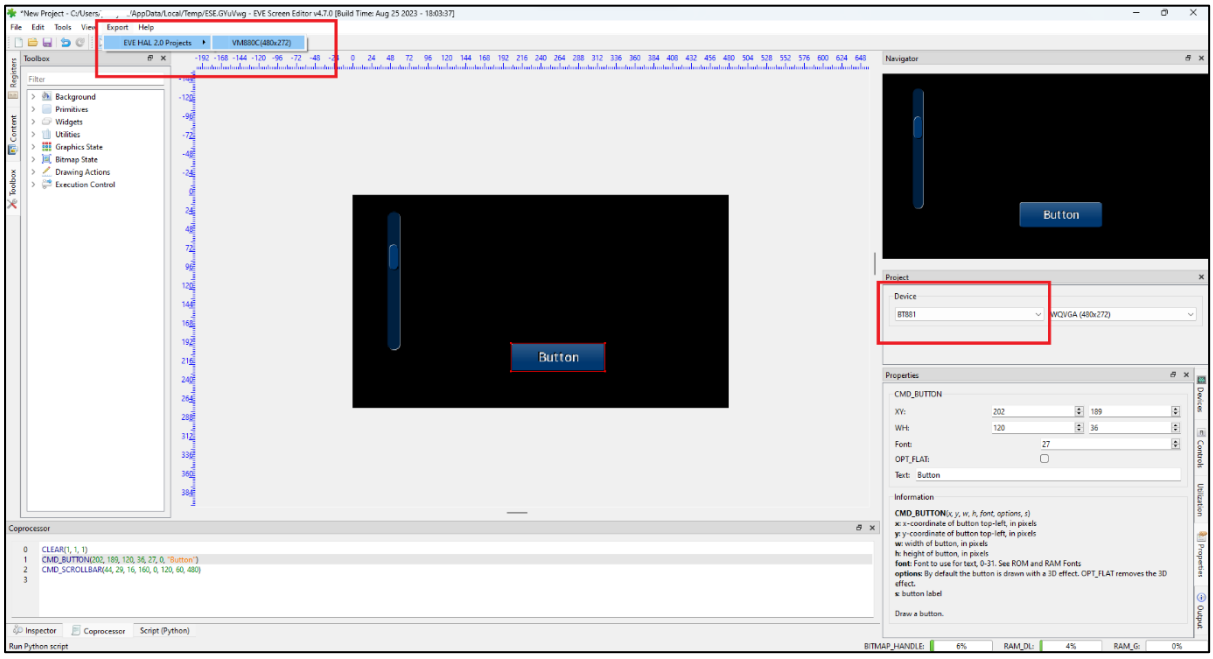


Figure 134 - The software libraries of Export feature are displayed on the ESE interface

Note 2: Define the callback functions for ESE to invoke.

```
import os, shutil, subprocess, sys, re, imp

supportedPlatforms = [2071, 2072]
exportModuleName = "export_bt8xx"
deviceModuleName = "ME817EV_MPSSE"

missingFileMessage = 'Unable to locate the required export file in the installation directory.'
wrongPlatformMessage = 'The project can not be exported to the selected platform.  The selected platform is incompatible with the current project device type.'

def displayName():
    return deviceModuleName

def run(name, document, ram, screnResolution):
    device_type = document["project"]["device"]
    if device_type not in supportedPlatforms:
        return wrongPlatformMessage
    else:
        exportScriptPath = os.path.abspath(__file__).rsplit("Export", 1)
        if len(exportScriptPath):
            exportScriptName = exportScriptPath[0] + exportModuleName + ".py"
            if os.path.exists(exportScriptName):
                exportScript = imp.load_source(exportModuleName, exportScriptName)
                return exportScript.run(name, document, ram, deviceModuleName, screnResolution)
            else:
                return missingFileMessage
        else:
            return missingFileMessage
```

Figure 135 - An export script

In your custom module file **<module_name>.py**, the following 2 functions are mandatory and must be defined properly:

- **displayName():** The function that returns a string, normally the module name, which will display on the submenu of "Export" menu of ESE. It will be triggered when a user clicks the Export menu and hovers the mouse over the submenu.

- **run(.....):** The entry function that ESE invokes to carry out the export task.

  When BT81X is chosen as the device type, the following function signature is anticipated:

  ```
  run(name, document, ram, screenResolution)
  ```

  Otherwise, the function below is expected:

  ```
  run(name, document, ram)
  ```

  where,

  name : Project name. A string object.
  document : All information about the current device (device model, register, display list, coprocessor). A string object in json format.
  ram : RAM of the device. A byte array object.
  screenResolution: Resolution of the screen (Example: 480x272). A string object.

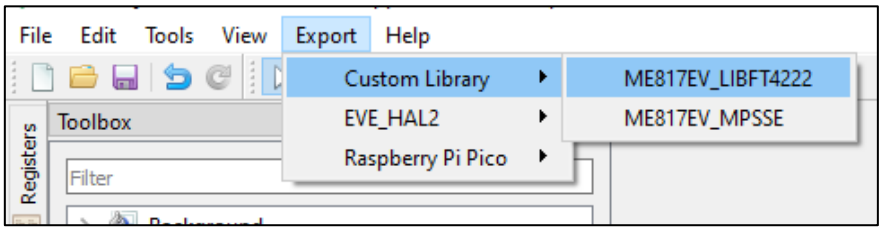


Figure 136 - Export Projects from User's Custom Software Library

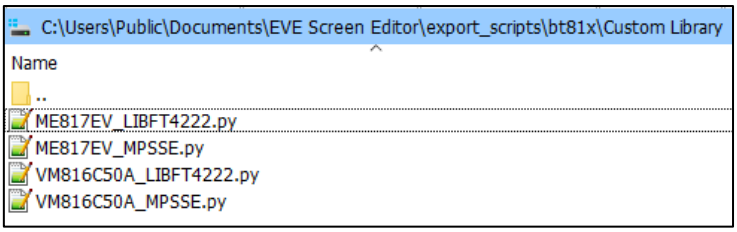


Figure 137 - Add Module Name Files to Installation Directory

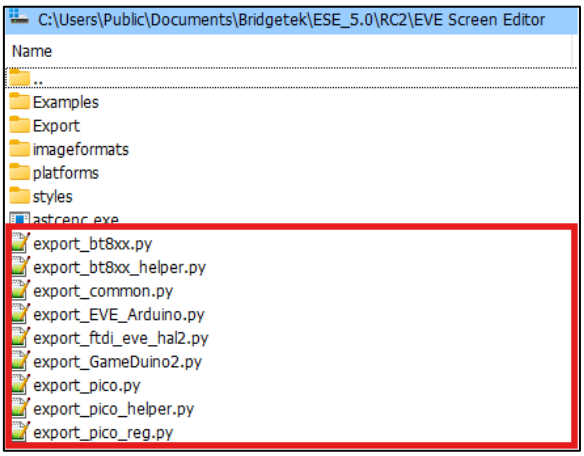


Figure 138 - Export Scripts

E. Working with EVE Asset Builder

When a user adds an image, a font or an animation generated by EVE Asset Builder (EAB) to the Content window, ESE will automatically gather the necessary information from the 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.

1. Load Image

ESE will automatically retrieve the required information from the .json file, which was created by EAB. The JSON file must have the same name as the .raw file and be located in the same folder.

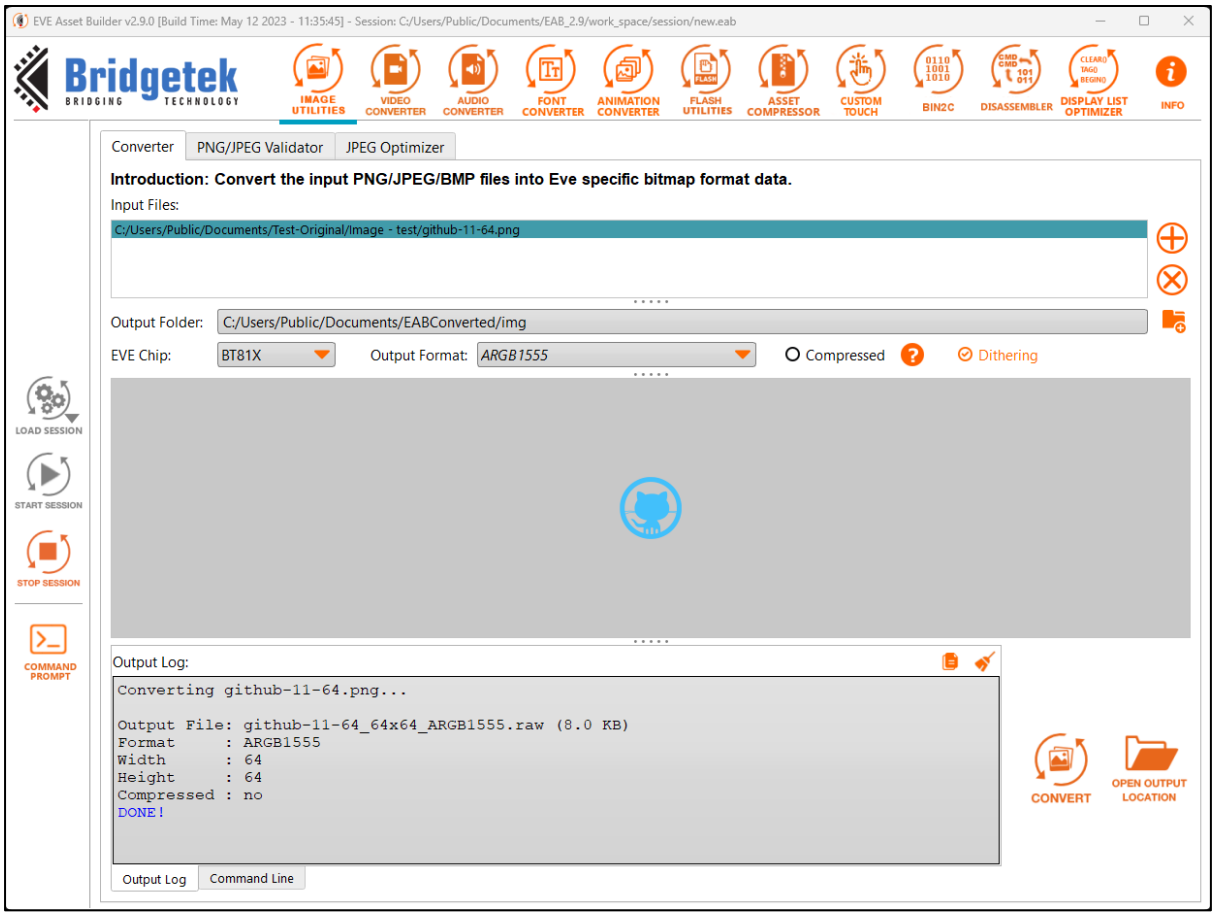


Figure 139 - EAB - Image Converter

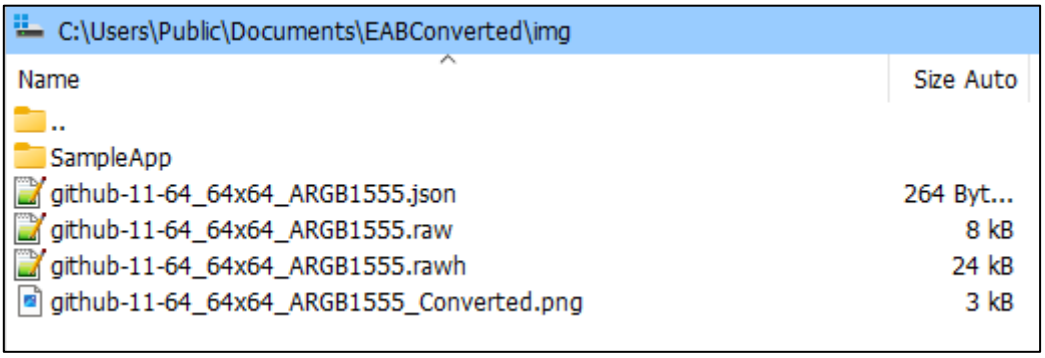


Figure 140 - EAB - Output of Image Converter

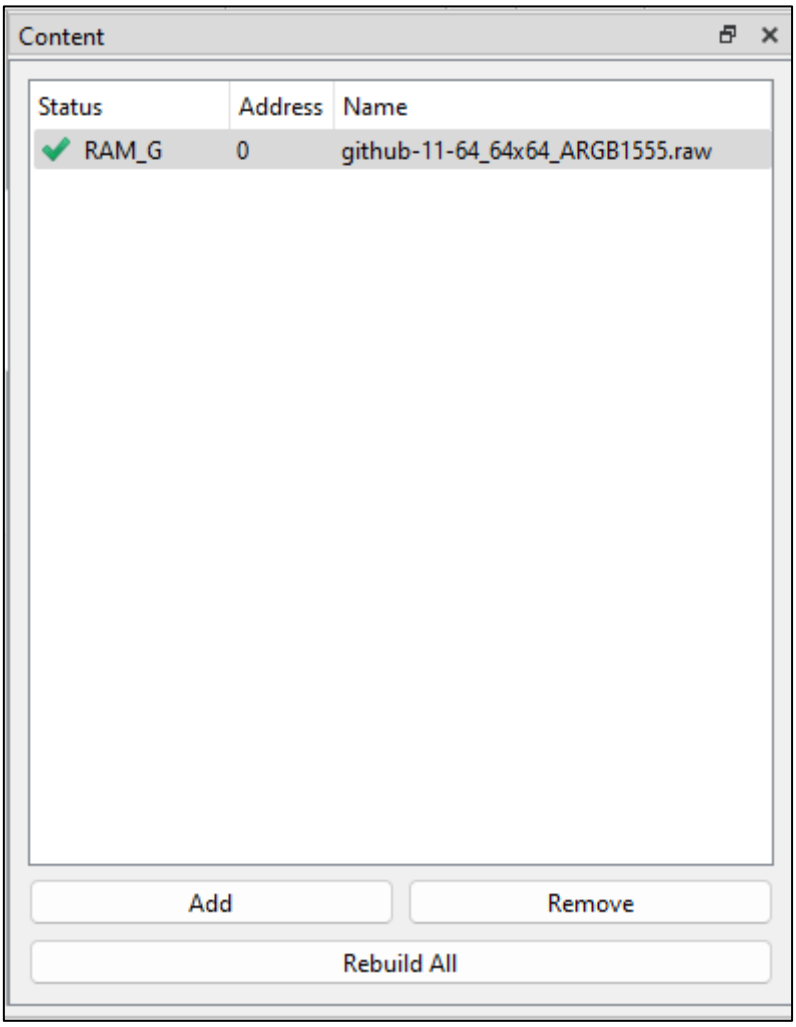


Figure 141 - Add the Image into Content Window

2. Load Font

Add a font with **legacy format (.raw)** generated by EAB to the Content window. ESE will automatically gather the necessary information from the .json file.

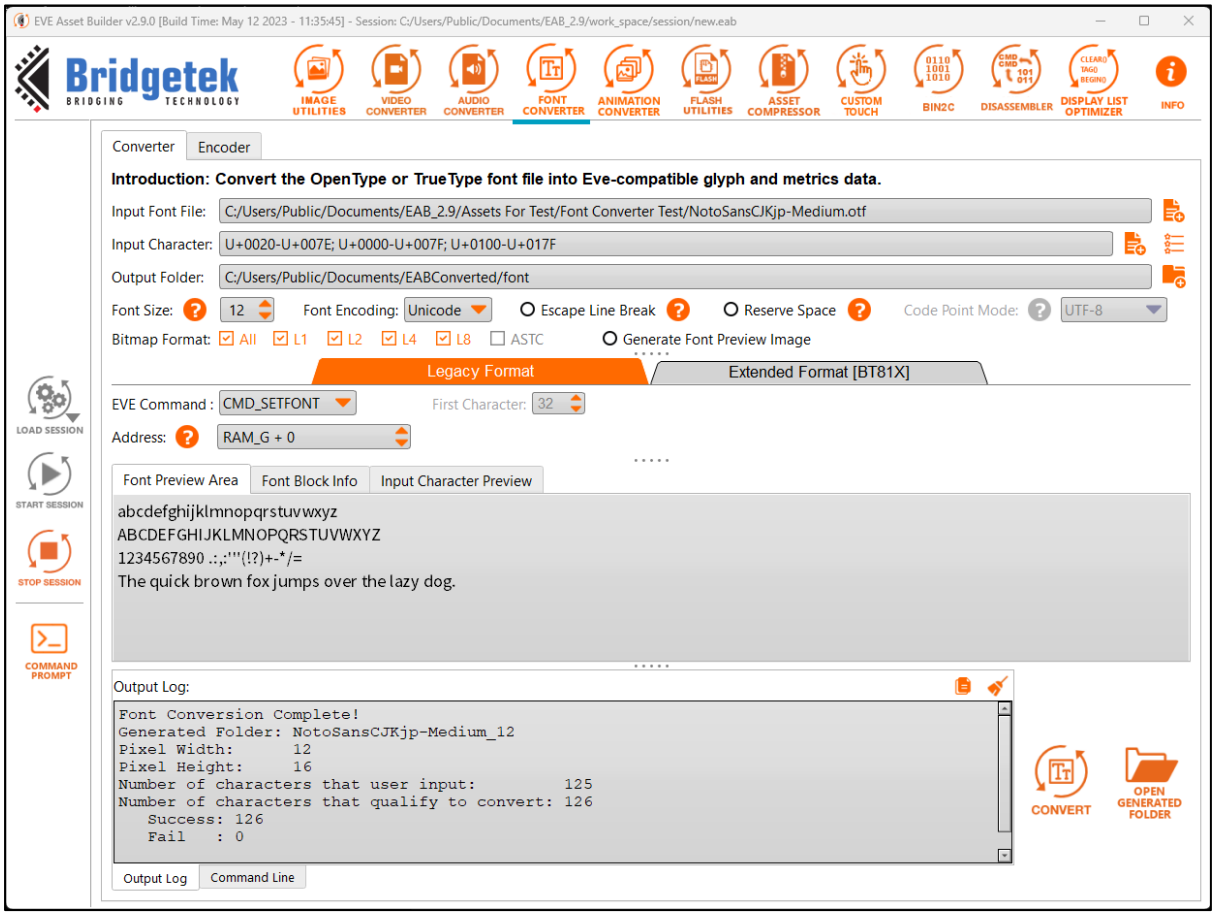


Figure 142 - EAB - Font Converter with Legacy Format

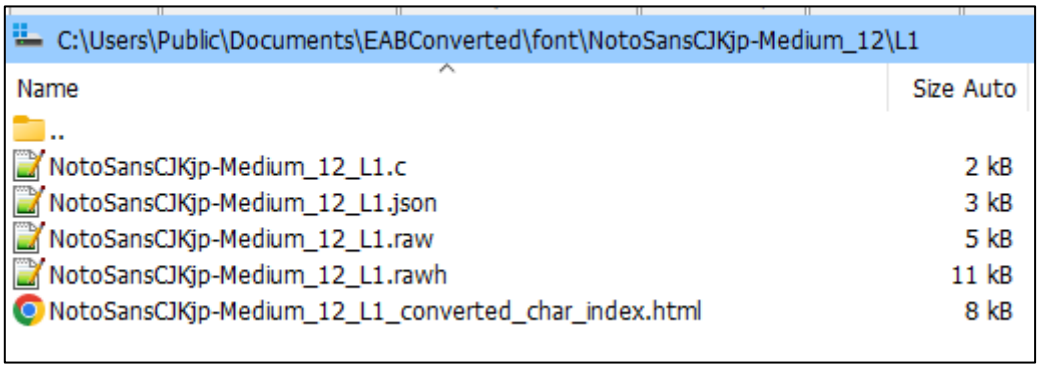


Figure 143 - EAB - Output of Font Converter with Legacy Format

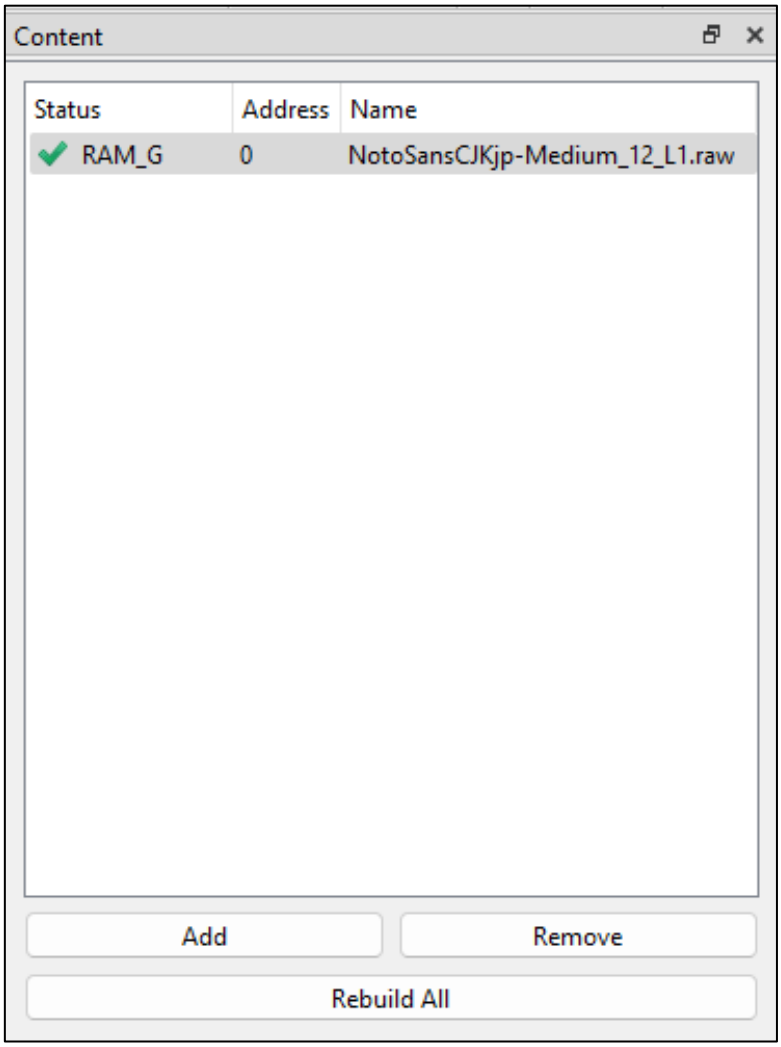


Figure 144 - Add a Legacy Font into the Content Window

Add a font with extended format (.xfont and .glyph) generated by EAB to Content window. ESE will automatically gather the necessary information from the .json file.

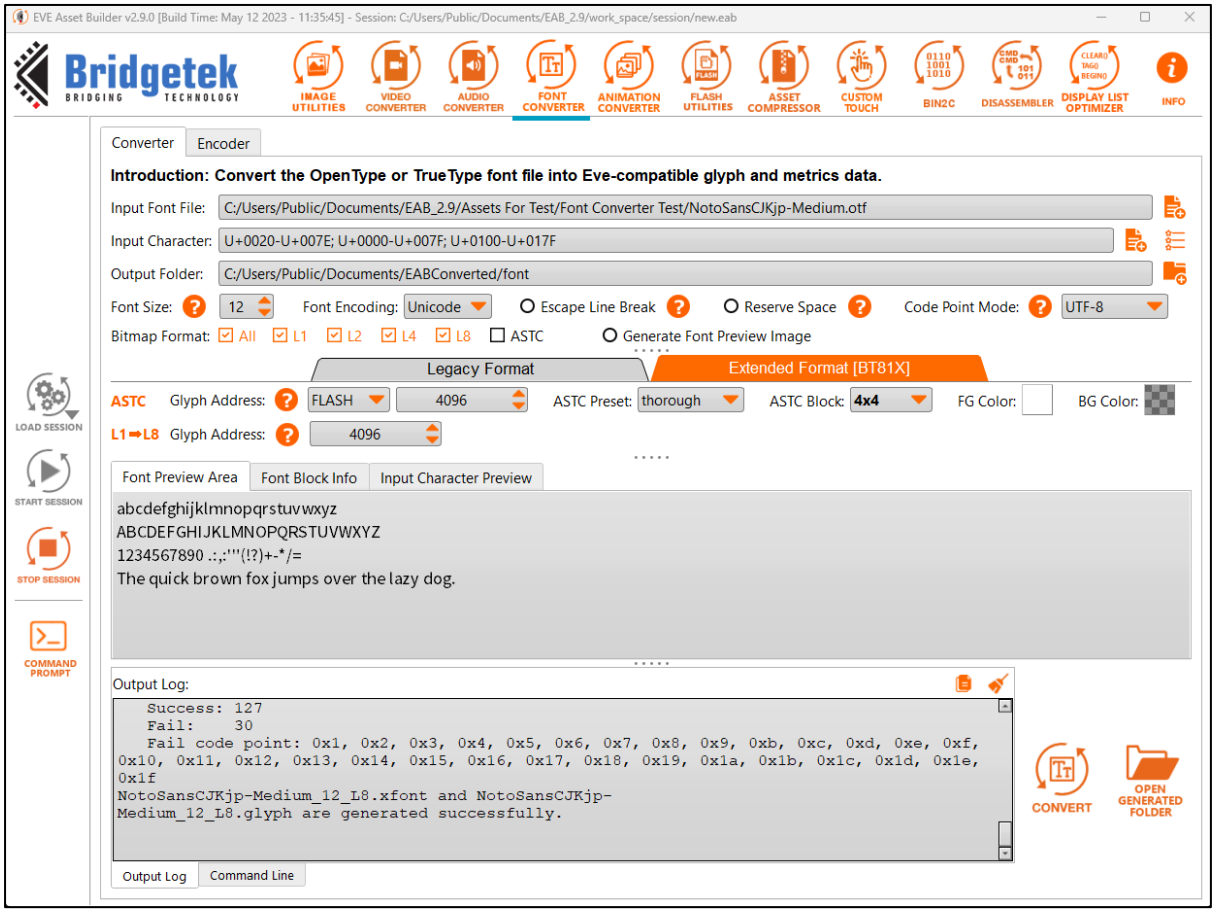


Figure 145 - EAB - Font Converter with Extended Format

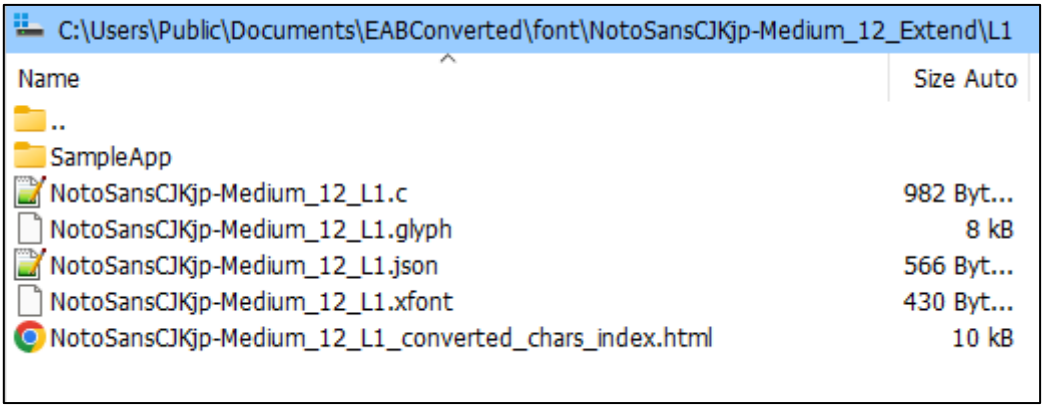


Figure 146 - EAB - Output of Font Converter with Extended Format

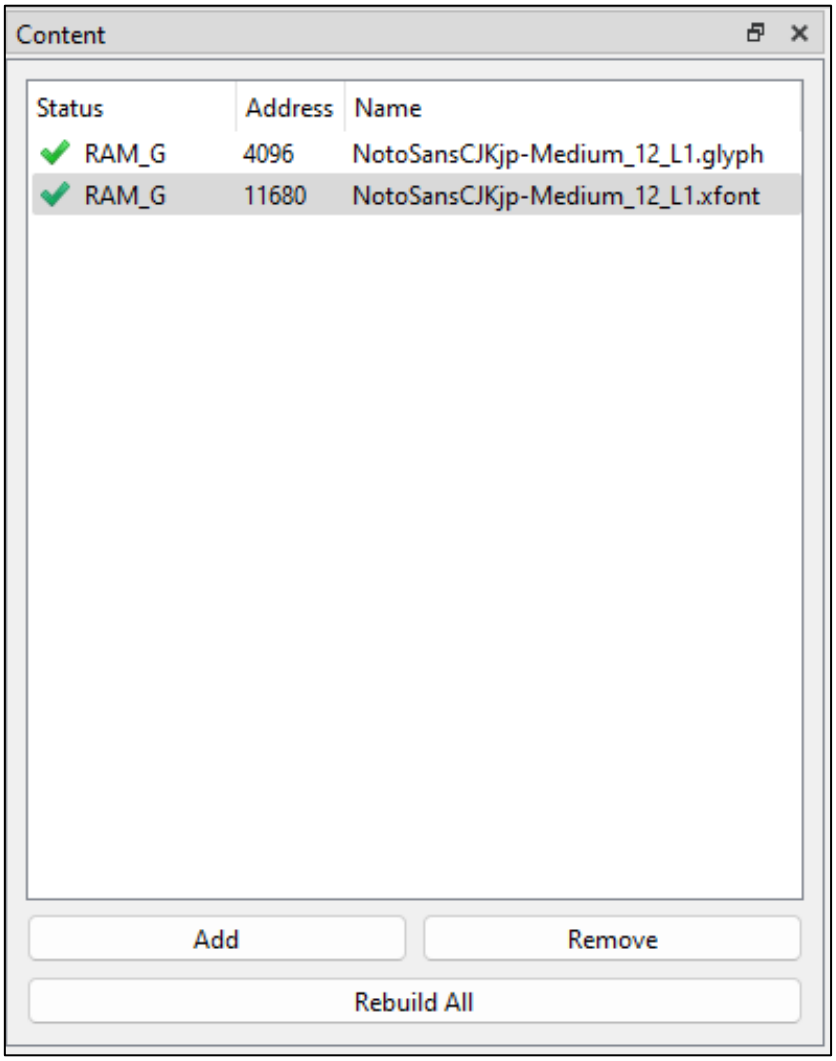


Figure 147 - Add the Extended Font into Content Window

3. Load Animation

Add an animation (.anim.flash and .anim.ram_g) generated by EAB to the Content window. ESE will automatically gather the necessary information from the .json file (or .readme with older versions of EAB files).

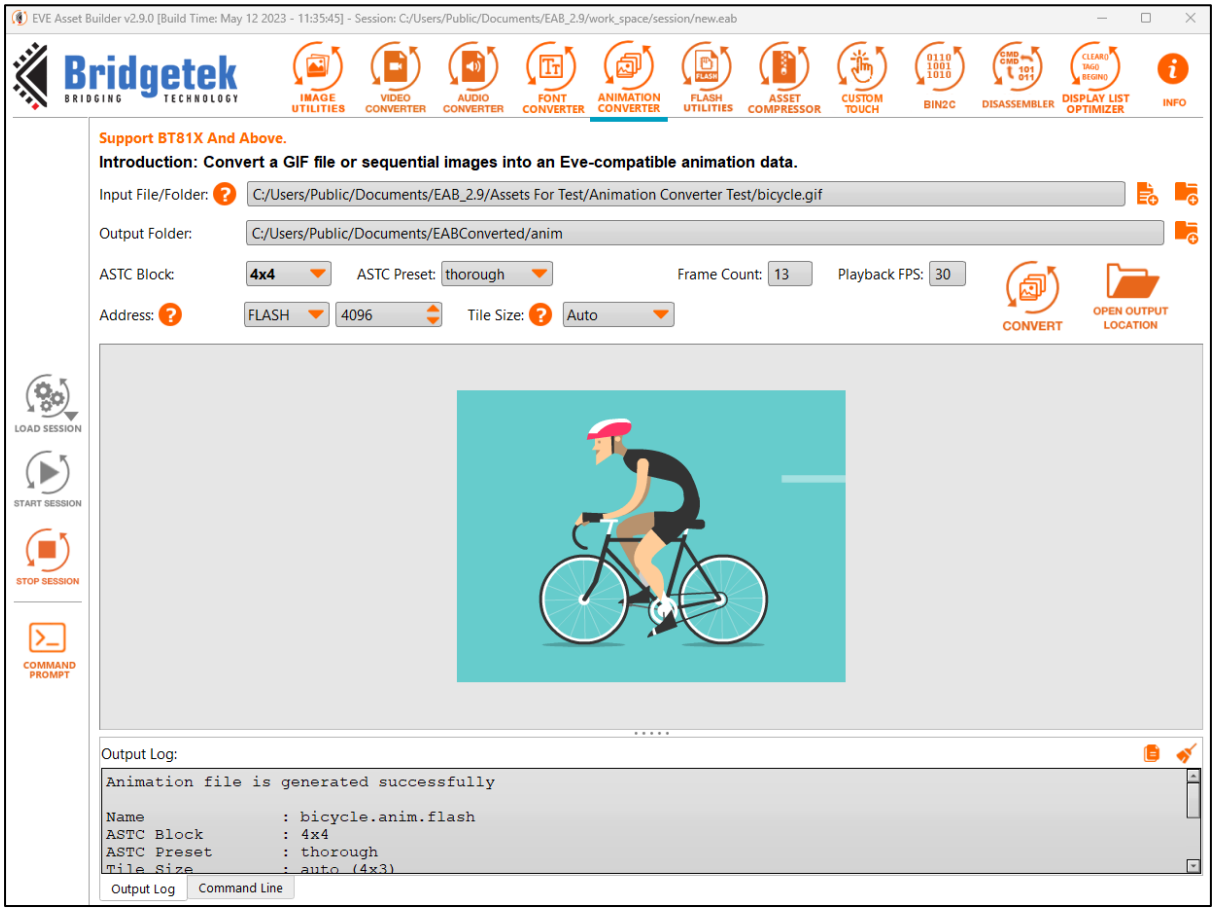


Figure 148 - EAB - Animation Converter with Flash Address

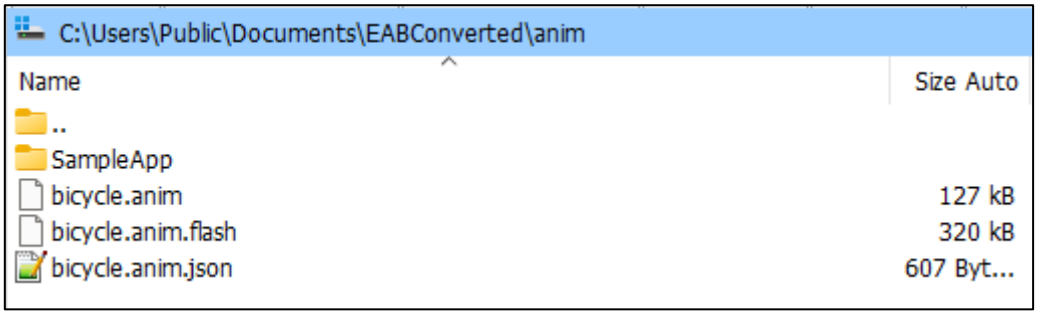


Figure 149 - EAB - Output for Animation Converter with Flash Address

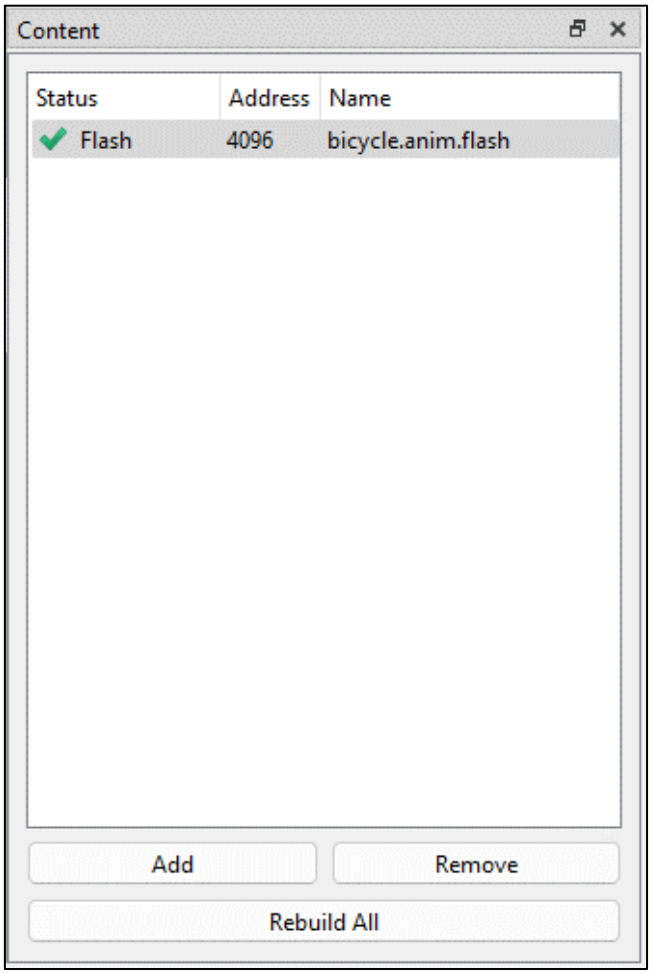


Figure 150 - Add the Animation with Flash Address into Content Window

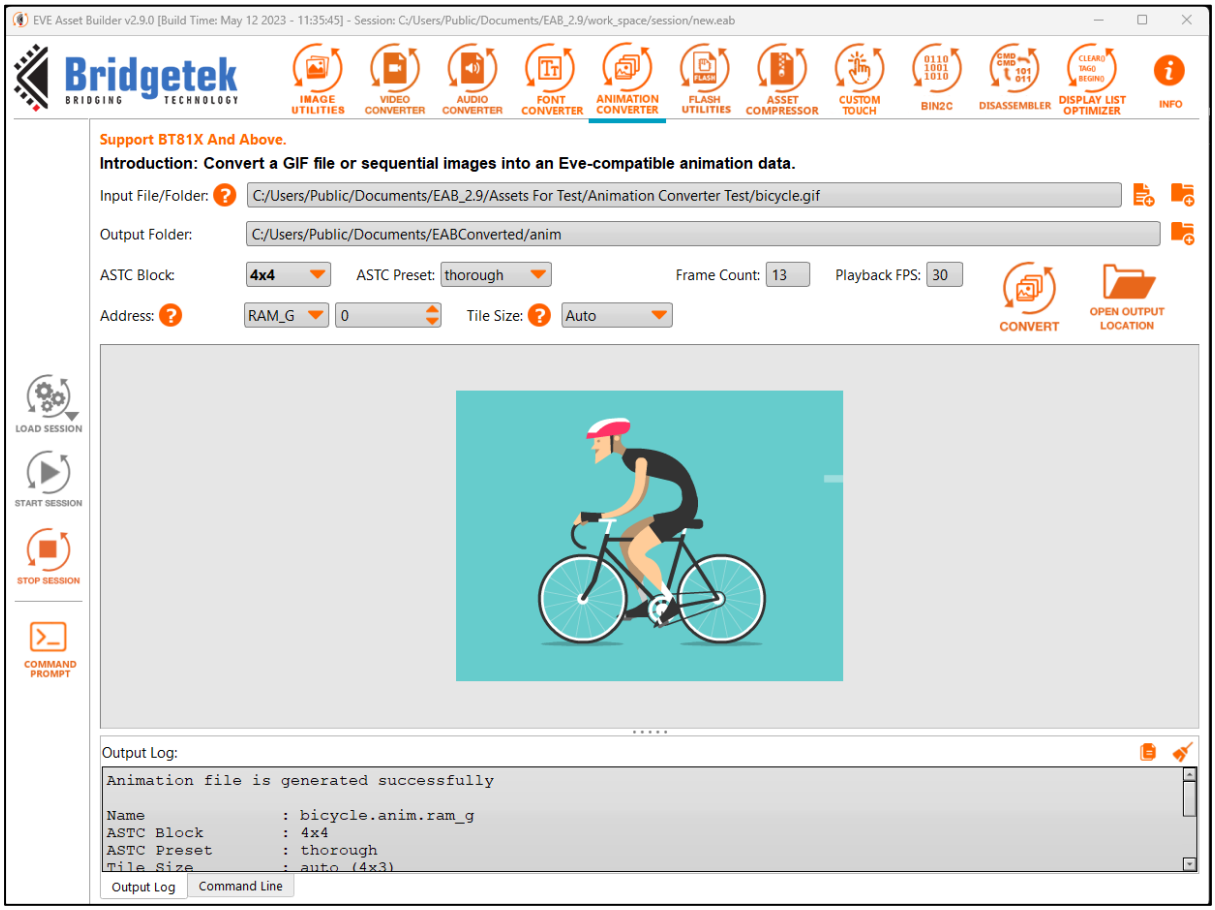


Figure 151 - EAB - Animation Converter with RAM_G Address

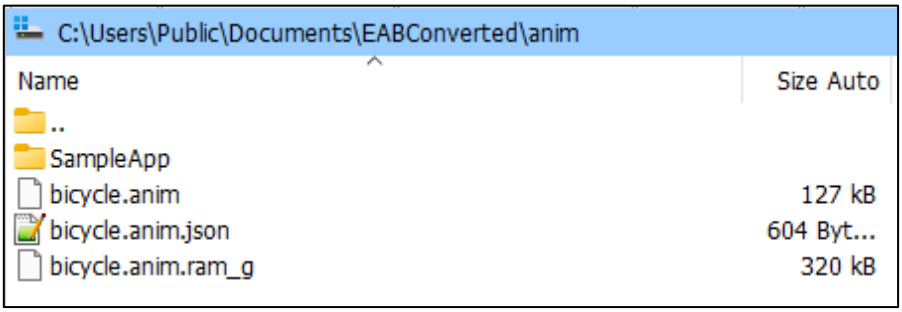


Figure 152 - EAB - Output for Animation Converter with RAM_G Address

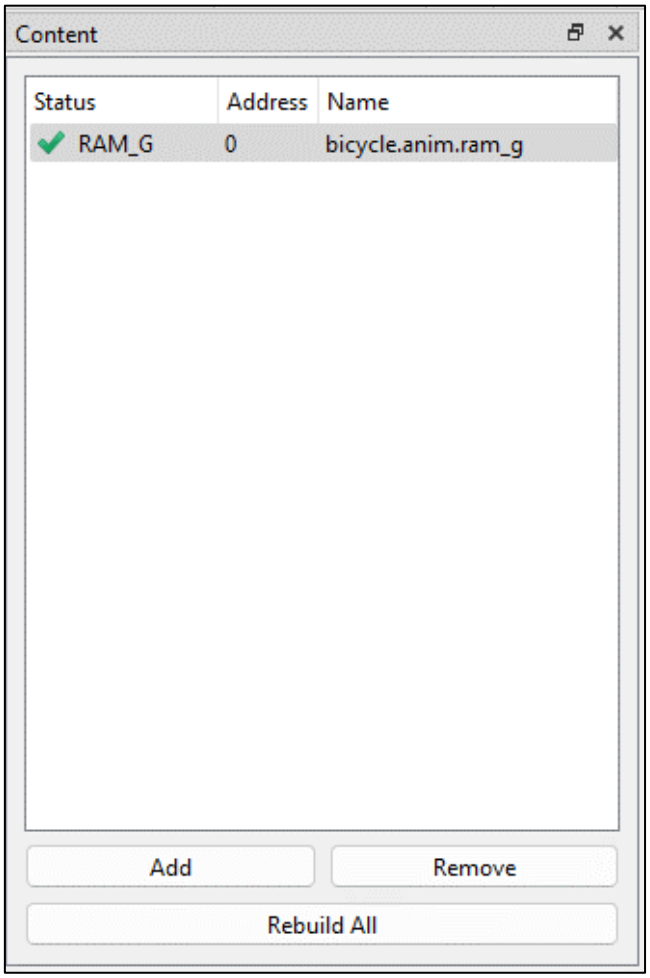


Figure 153 - Add the Animation with RAM_G Address into Content Window

4. Load Flash Image

Flash image (.bin) and flash map (.map) are generated by another tool called *EVE Asset Builder* whose 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 **Flash Utilities** tab.
2. Add necessary asset files.
3. Set output folder and output name for the flash image.
4. Press **[Generate]** button.
5. Generated files are saved in the output folder.

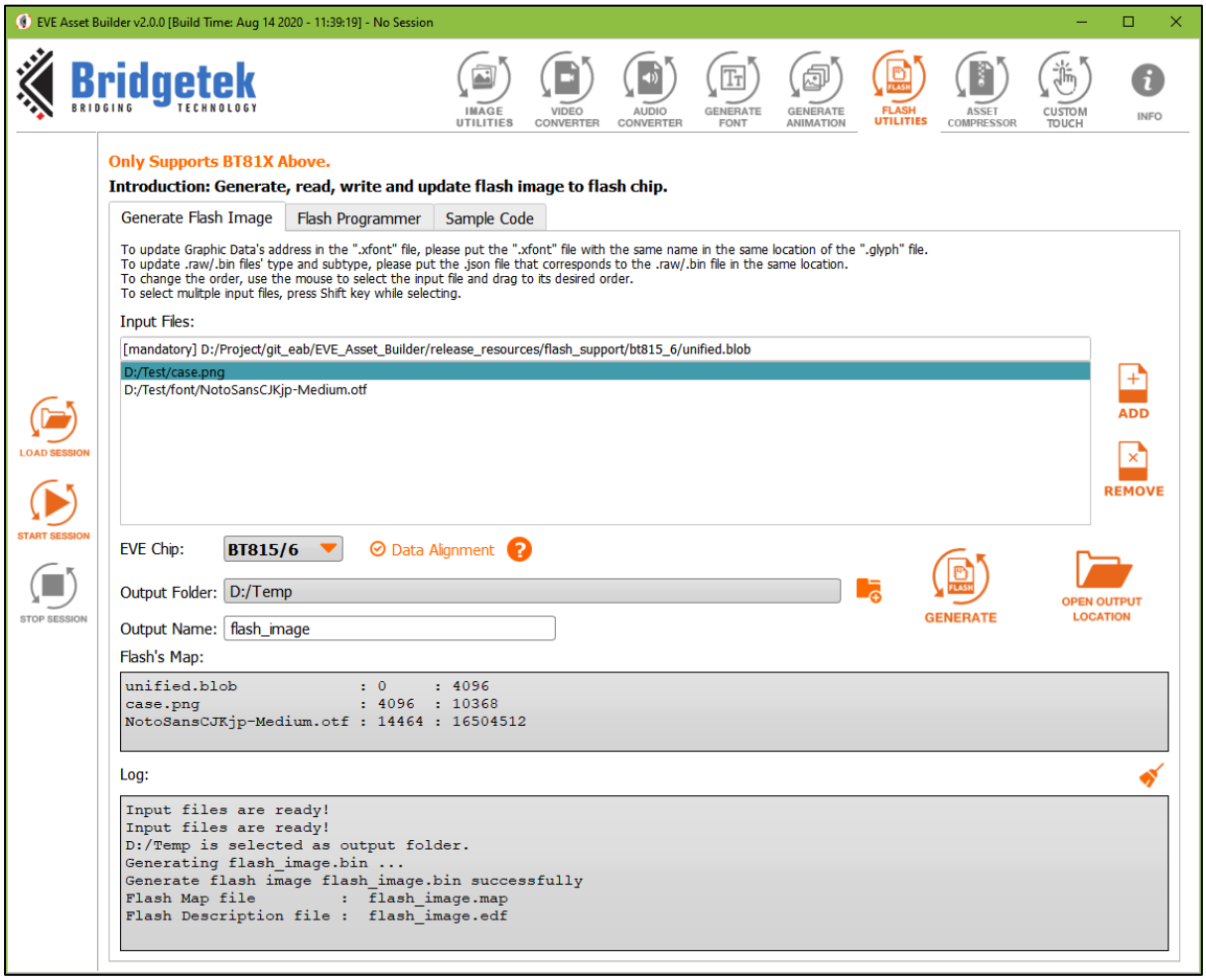


Figure 154 - Flash Image Generation by EAB

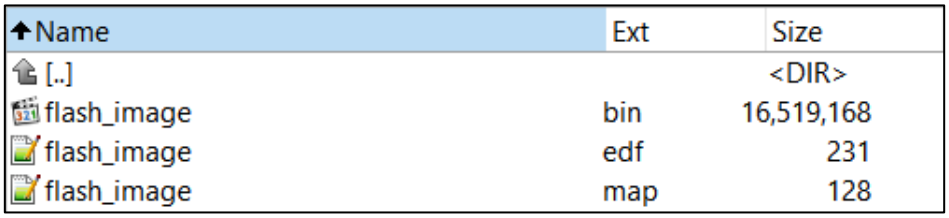


Figure 155 - Generated Files in Output Folder

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.

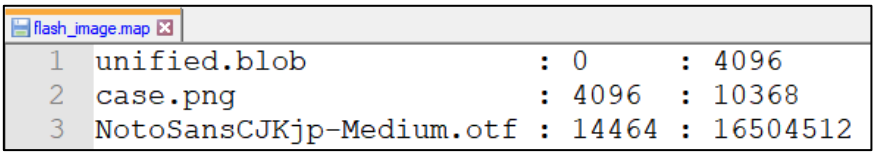


Figure 156 - Content of .map File

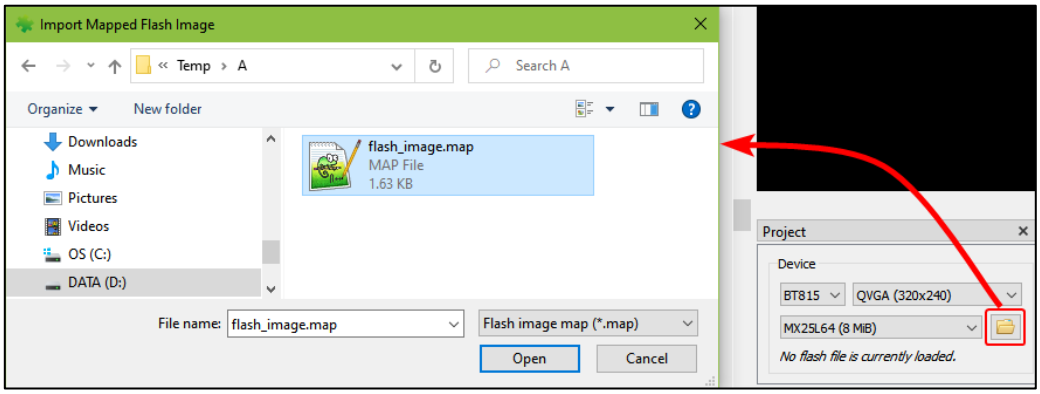


Figure 157 - Load Flash File

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

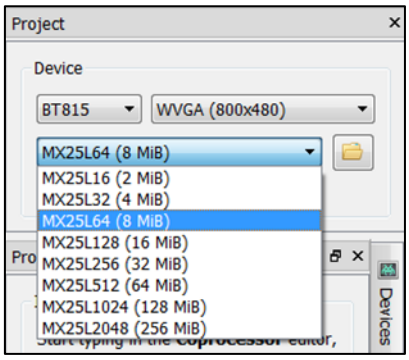


Figure 158 - Select Flash Size

Upon loading the flash file, its path is displayed.

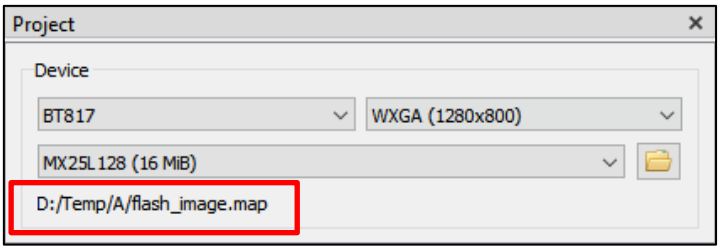


Figure 159 - Display Flash Path

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

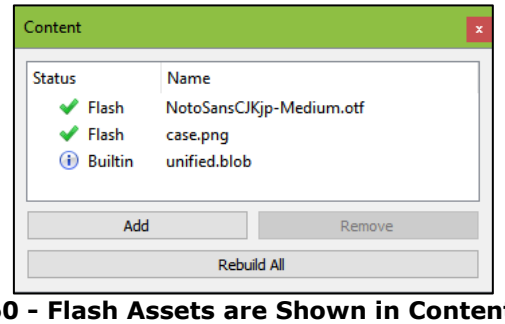


Figure 160 - Flash Assets are Shown in Content Window

VII. 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

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.

VIII. Contact Information

Refer to https://brtchip.com/contact-us/ for contact information.

Distributor and Sales Representatives

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

Appendix A - References

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
DDR RAM	Double Data Rate Random-Access Memory

Appendix B – List of Figures & Tables

List of Figures

List of Tables

Appendix C – Revision History

Document Title : BRT_AN_037 EVE Screen Editor 5.0 User Guide
Document Reference No. : BRT_000231
Clearance No. : BRT#160
Product Page : https://brtchip.com/toolchains/#EVEScreenEditor
Document Feedback : Send Feedback

Revision	Changes	Date
Draft Version 0.6	Initial Release of ESE 3.3	15-11-2019
Version 1.0	Updated release (content updated with respect to ESE 4.0 version)	03-12-2020
Version 1.1	Updated as per ESE Version 4.1	10-12-2020
Version 1.2	Updated as per ESE Version 4.2	27-10-2021
Version 1.3	Updated as per ESE Version 4.3	05-04-2022
Version 1.4	Updated as per ESE Version 4.4	25-11-2022
Version 1.5	Updated as per ESE Version 4.5	08-03-2023
Version 1.6	Updated as per ESE Version 4.6	05-07-2023
Version 1.7	Updated as per ESE Version 4.7	16-10-2023
Version 1.8	Updated as per ESE Version 4.8	10-01-2024
Version 1.9	Updated as per ESE Version 4.9	08-05-2024
Version 2.0	Updated as per ESE Version 4.10	03-07-2024
Version 2.1	Updated as per ESE Version 4.11	24-10-2024
Version 2.2	Updated as per ESE Version 5.0	21-03-2025