USER GUIDE

EVE Screen Editor 4.0

Document Version: 1.0

Issue Date: 03-12-2020

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 developer working with EVE products.

C. Related Documents

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

D. Feedback

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

II. Overview

A. Introduction

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

Co-processor commands and display list can also be provided as an 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 that the screen design can be created without taking the details of the MCU into consideration. Users have the option to export the design to hardware platform specific source code. This greatly reduces the effort to start up a new project on real hardware.

If users have EVE Series boards and FT4222/MPSSE cable, the screen design can be synchronized with the real 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's get started!

B. Supported Devices

ESE supports all series of EVE chips, including FT80X, FT81X and BT81X.
ESE also supports EVE modules which are listed out as below.

❖ For [***Exporting Feature***]:

EVE Chip Family	Platform	EVE Modules
FT80X	Arduino Projects	*ADAM_4DLCD_FT843, Breakout_4DLCD_FT843, VM800B35, VM800P35, VM800P43_50, VM801B43, VM801P43_50*
	EVE Hal Projects	*VM800B35, VM800B43_50, VM800BU35, VM800BU43_50, VM800C35, VM800C43_50, VM801B43_50A*
	GameDuino2 Projects	*GameDuino2*
FT81X	EVE Hal Projects	*ME810A_HV35R, ME810A_WH70R, ME810AU_WH70R, ME811A_WH70C, ME811AU_WH70C, ME812A_WH50R, ME812AU_WH50R, ME813A_WH50C, ME813A_WV7C, ME813AU_WH50C, VM810C50*
BT815/6	EVE Hal Projects	*VM816C50A, VM816CU50A*
BT817/8	EVE Hal Projects	*ME817EV*

❖ For [***Device Sync***]:

Host Platform	EVE Modules
FT4222, MPSSE	*ME817EV-WH70C*
FT4222, MPSSE	*ME817EV-WH10C*
FT4222, MPSSE	*VM816CU50A*
FT4222, MPSSE	*VM816C50A*
FT4222, MPSSE	*ME813AU-WH50C*
FT4222, MPSSE	*ME812A-WH50R*
FT4222, MPSSE	*ME810A-WH50R*
MPSSE	*VM800B*

C. Key Features

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

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

D. What's new in ESE 4.0?

ESE 4.0 Features Vs ESE 3.3 Features	
ESE 4.0	**ESE 3.3**
• Supports new EVE chip BT817/8 • Provides more user friendly value options for REG_OUTBITS • Updated Python to 3.8.0, FreeType to 2.10.2 • Added new resolutions when BT817/8 is selected • Updated the latest flash blob • Added BT817/8 specific commands to ESE UI (Toolbox) • Improvements on the register window • Improved device configuration window • Added a gauge example into BT81x subfolder of examples • Added ME817EV board in Device Manager • Set shortcut Ctrl+R for Reset Emulator menu • Improved CMD_GETPROPS and CMD_GETPTR commands in BT817/8 • Fixed issues related to opening ESE projects	• Constrain either horizontal or vertical positioning when dragging an object • Supports Riverdi EVE modules • Disabled Refresh, Device Manage, Device Display buttons upon connecting to a device • Added REG_OUTBITS to device configuration dialogue • Show the pixel value at status bar when mouse hover over the viewport window • Added Blend_Func example project • Added the circular progress bar widget example

E. Known Issues & Limitations

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

1. Device sync up feature: CMD_PLAYVIDEO without OPT_MEDIAFIFO is not supported
2. No Unicode support of file name for image files (PNG or JPG) in export feature, although the files can be imported in content manager successfully.
3. No prompt window is shown when connection has failed in device manager.
4. CMD_SNAPSHOT2 is not fully supported.
5. Properties window of VERTEX2F does not show correct value when VERTEX_FORMAT is not set to 4(1/16 pixel precision).
6. Properties window of CMD_LOADIMAGE and CMD_PLAYVIDEO does not show a warning message when an incorrect file name is entered into the stream line box.
7. Export feature may fail if project file path is more than 255 characters.
8. Playing invalid video stream or animation frame or graphic data may cause the ESE to hang.

F. Credits

1. Open Source Software

- Qt: http://doc.qt.io/qt-5/licensing.html under LGPL.
- Python: https://www.python.org under GPL- compatible.
- zlib: https://zlib.net/
- libpng: http://www.libpng.org/pub/png/libpng.html
- FreeType: https://www.freetype.org under GPL license.

2. Icons Copyright

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

III.Setup & Installation

A. System Requirements

To install ESE 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 free space
- OS: Windows 7 and above, 64-bit platform
- Display resolution: At least 1280 by 800 pixels

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

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

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

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

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

B. Hardware Requirements

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

Board	Host Platform
ME817EV-WH70C	FT4222, MPSSE
ME817EV-WH10C	FT4222, MPSSE
VM816CU50A	FT4222, MPSSE
VM816C50A	FT4222, MPSSE
ME813AU-WH50C	FT4222, MPSSE
ME812A-WH50R	FT4222, MPSSE
ME810A-WH50R	FT4222, MPSSE
VM800B	MPSSE

C. Dependencies / Pre-Requisites

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

  If the PC does not have Microsoft Visual Studio 2017 installed, Visual C++ Redistributable is required. Users can download this from: https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads

D. Installing ESE

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

i. Download the package from www.brtchip.com.

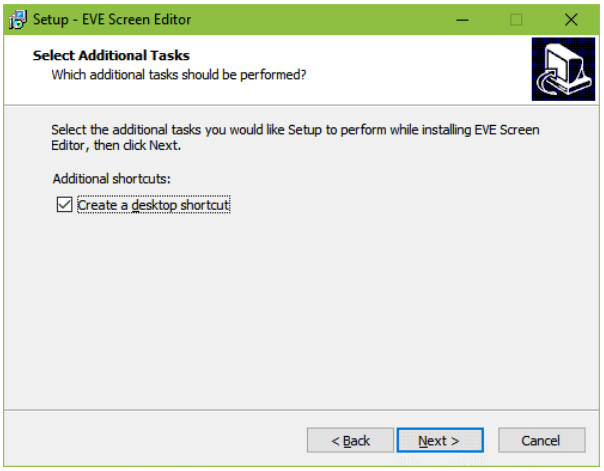
ii. When prompted with a download dialog box. Click on **Save**.

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

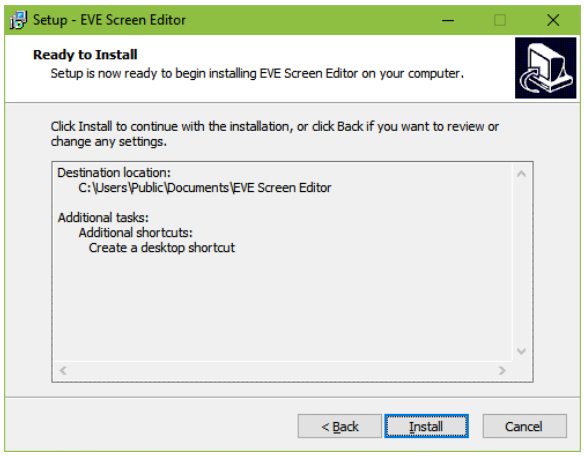
iv. Extract the zip file contents. Double click on the executable file – **EVE Screen Editor 4.0.exe**

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.

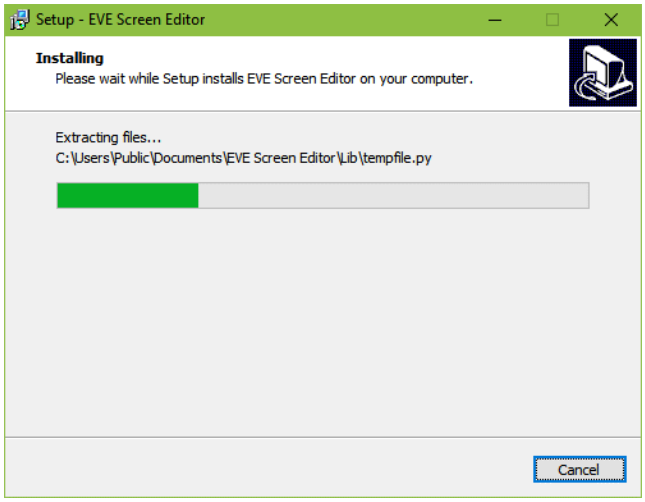
vi. In the **Select Additional Tasks** window, check **"Create a desktop"** boxes, to have the ESE icon displayed on the desktop if required. Click **Next** to prepare for the installation.



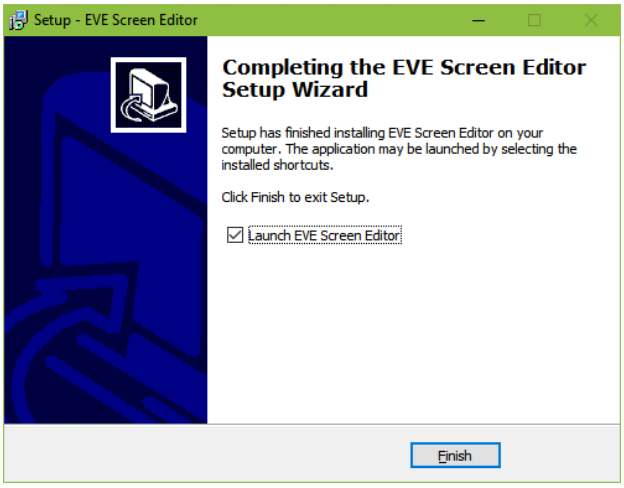
vii. The initial setup is completed and the application is ready to be installed.



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



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



E. Installation Folder

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

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

Table 1 - Installation Folder

IV. The Graphical User Interface

A. Overview

ESE user interface has the following components:

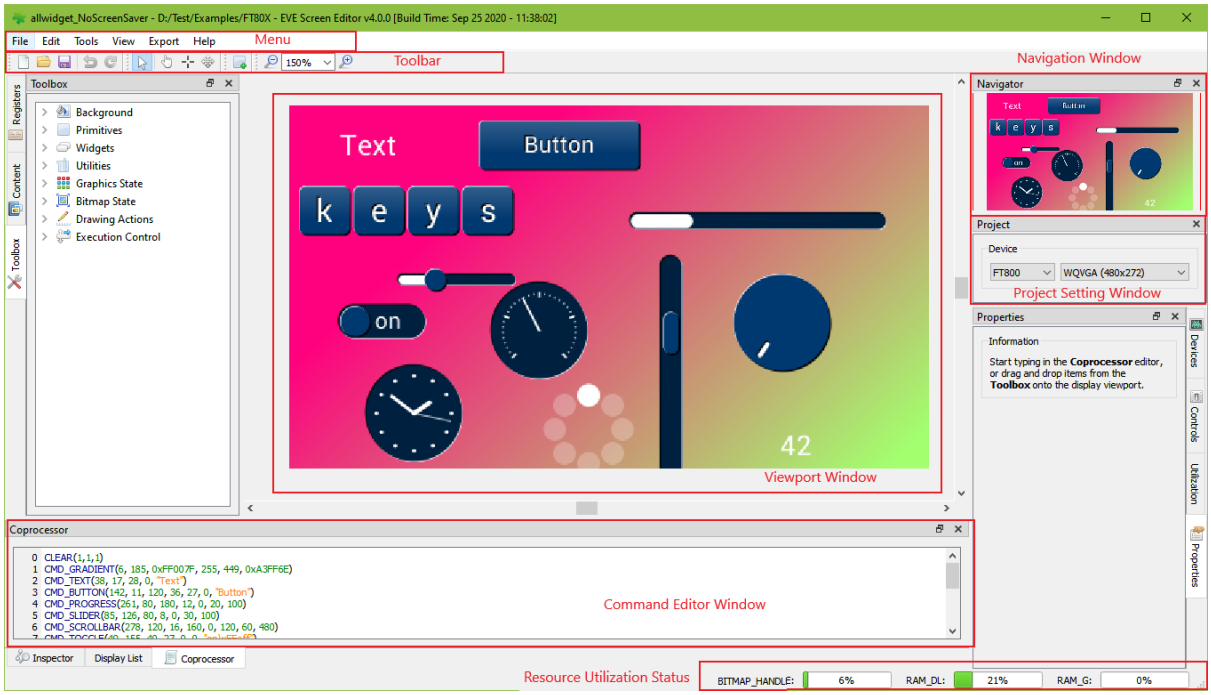


Figure 1 - User Interface Components

B. Menu bar, Toolbar and Status bar

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

File Edit Tools View Export Help

1. Menu bar

File Menu

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

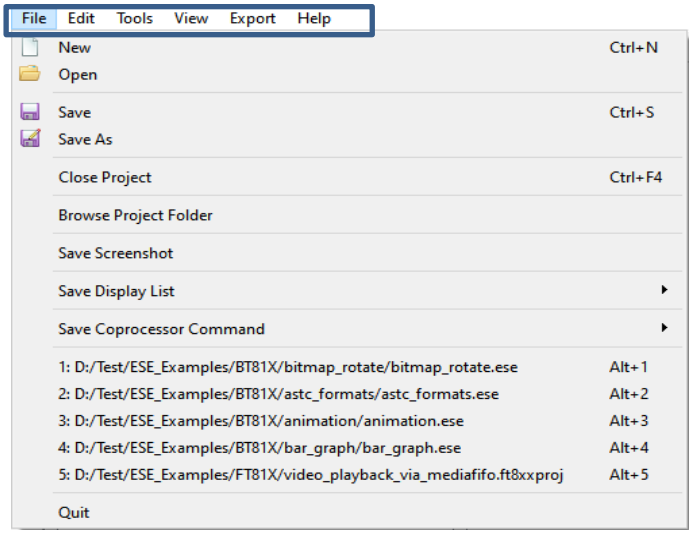


Figure 2 - File Menu

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

Edit Menu

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

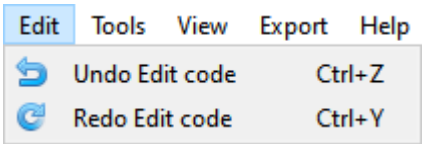


Figure 3 - Edit Menu

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

Tools menu

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

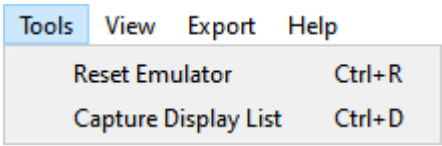


Figure 4 - Tool Menu

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

View Menu

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

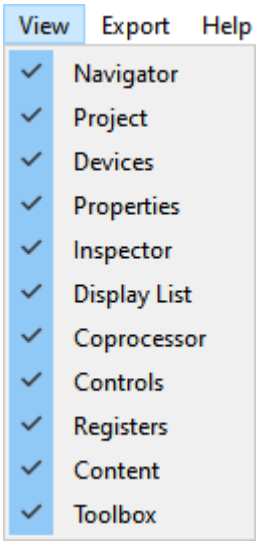


Figure 5 - View Menu

Export Menu

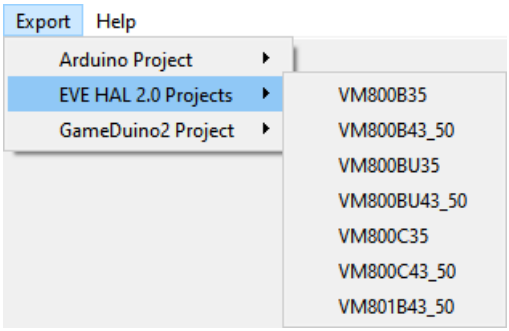


Figure 6 - Export Menu

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

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

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

Help Menu

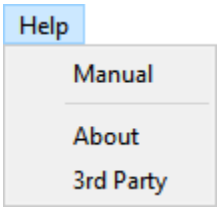


Figure 7 - Help Menu

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

2. Toolbar

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

Figure 8 - Toolbar

Menu 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.
Touch	Force touch cursor in viewport Enable mouse click on the viewport to simulate touch action on the touch panel connected to EVE touch engine. Therefore, the touch related registers are updated in the inspector. It is especially useful for CMD_SKETCH.
Trace	To force trace cursor in viewport. See **Trace the pixel** for more details.
Edit	To force widget editing cursor in viewport
Insert	To insert duplicates of currently selected widget or primitive at clicked position, overrides any current cursor selection
Zoom Out	To zoom out emulator view port
Zoom In	To zoom in emulator view port
Zoom Rate	To select concrete zooming rate

3. Status Bar

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

Figure 9 - Status Bar

C. Editors and Inspector

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

Editor

Editors provide individual windows to co-processor commands and display list commands which are sent to the EVE co-processor RAM_CMD and EVE graphics engine RAM_DL respectively.

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

Inspector

Inspector displays the content of RAM_DL and RAM_REG and cannot be edited. RAM_DL and RAM_REG can be selected line by line and then copied to another text editor.

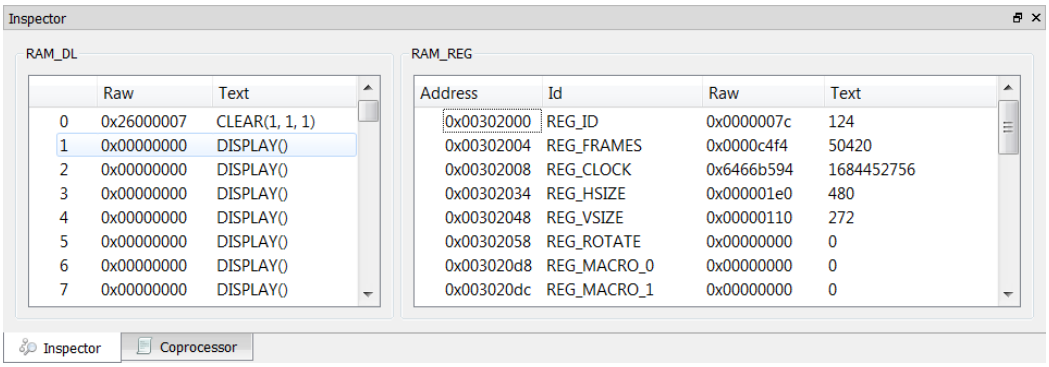


Figure 10 – Inspector

1. Coprocessor Command Editor

Figure 11 - Coprocessor Command Editor

The features are as below:

- Full set commands support and auto-completion
- Decimal and hexadecimal values for parameters
- Error highlights
- Step by step emulation

Note:

- CLEAR command is auto-inserted when ESE is launched.
- CMD_CALIBRATE/CMD_LOGO/CMD_SPINNER commands will pause the following commands and shall be the last command in the editor.

2. Display List Editor

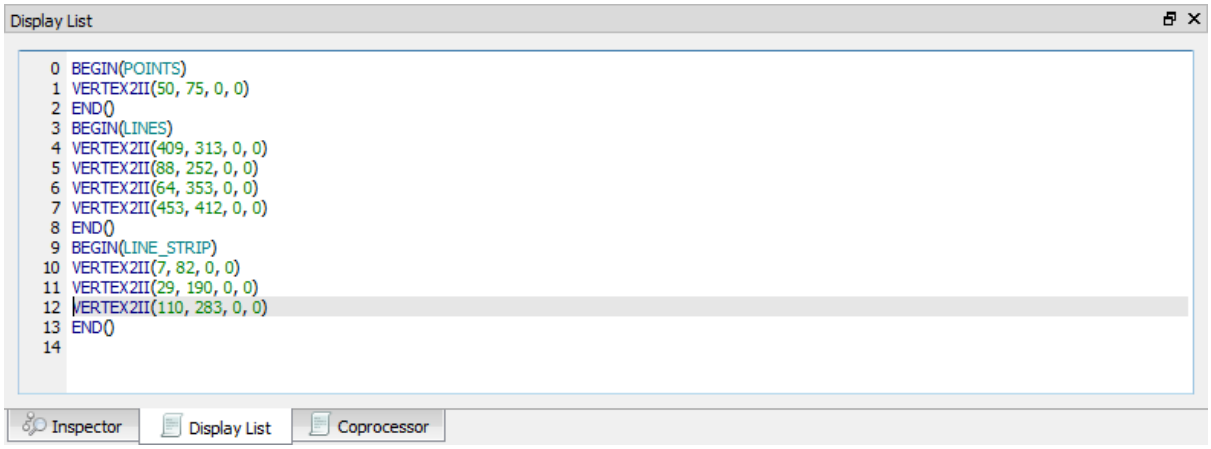


Figure 12 - Display List Editor

The features are as below:

- Display list commands auto-completion
- Decimal and hexadecimal values for parameters
- Error highlights
- Step by step emulation

Note:

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

- By default, the Display List editor is hidden.

3. Inspector

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

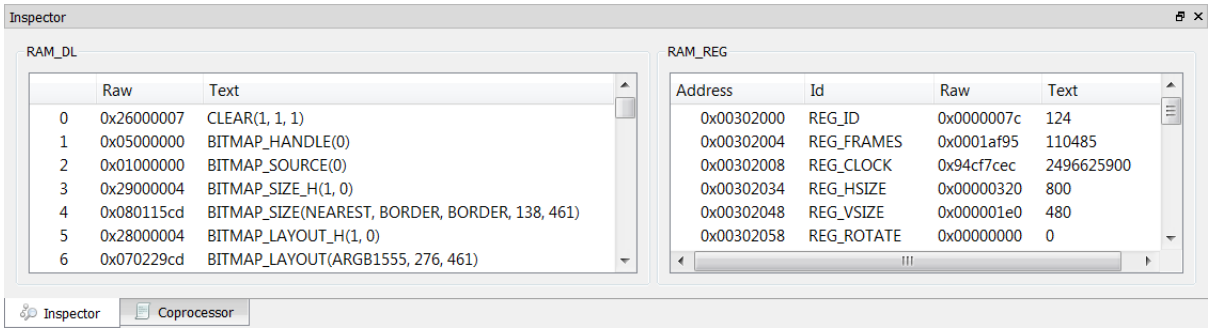


Figure 13 - Inspector

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.

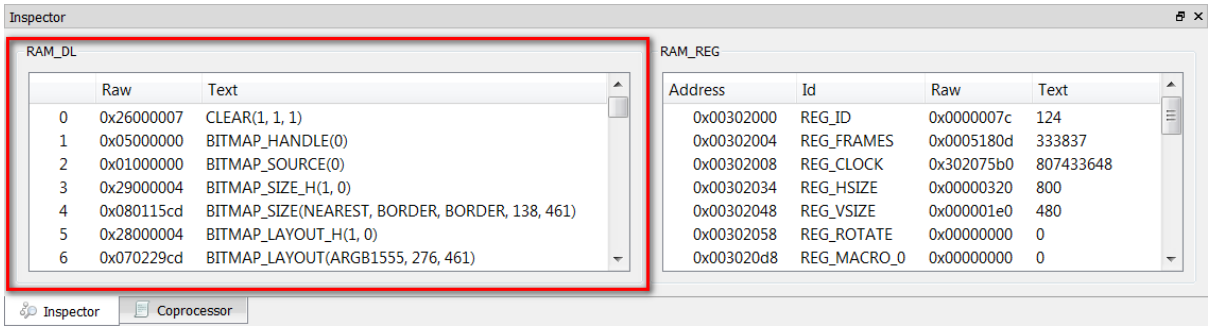


Figure 14 - RAM_DL

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

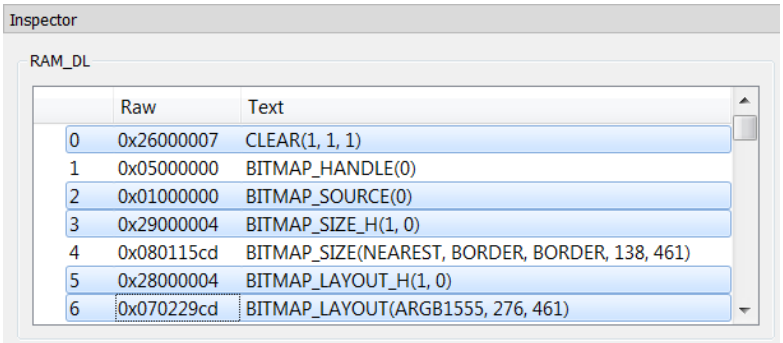


Figure 15 - Select rows in RAM_DL

	Raw	Text
0	0x26000007	CLEAR(1, 1, 1)
1	0x05000000	BITMAP_HANDLE(0)
2	0x01000000	BITMAP_SOURCE(0)
3	0x29000000	BITMAP_SIZE_H(0, 0)
4	0x0800dc55	BITMAP_SIZE(NEAREST, BORDER, BORDER, 110, 85)
5	0x28000000	BITMAP_LAYOUT_H(0, 0)
6	0x0701b855	BITMAP_LAYOUT(ARGB1555, 220, 85)

Figure 16 - Paste selected rows in RAM_DL

RAM_REG

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

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

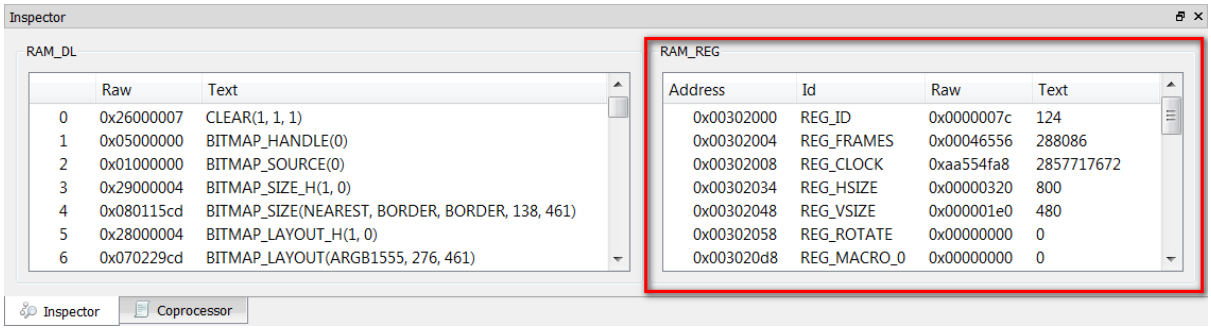


Figure 17 - RAM_REG

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

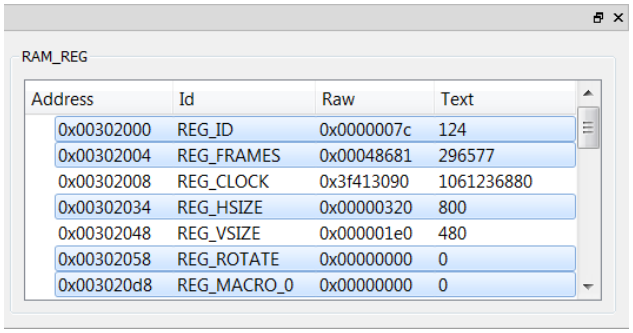


Figure 18 - Select rows in RAM_REG

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 19 - Paste selected rows in RAM_REG

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 at the left side of the viewport.

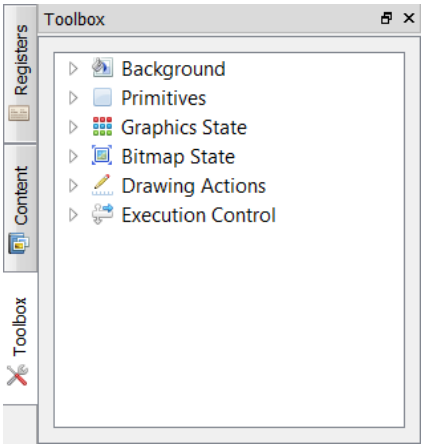


Figure 20 – Toolbox

1. Toolbox

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

Display list mode

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

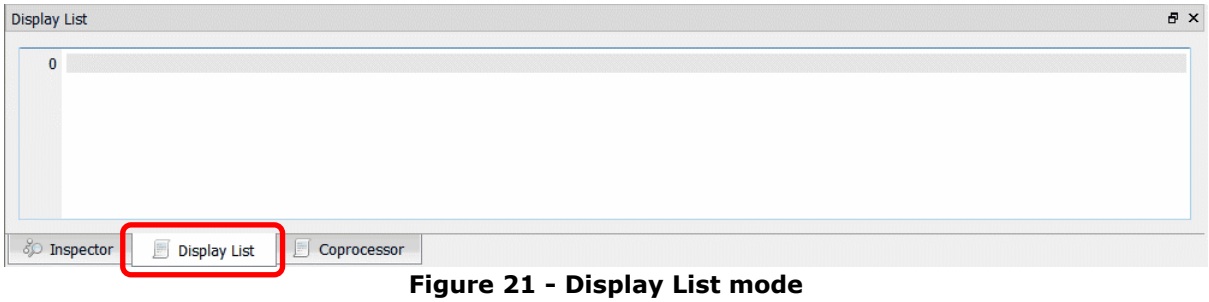


Figure 21 - Display List mode

The toolbox will be enabled as below:

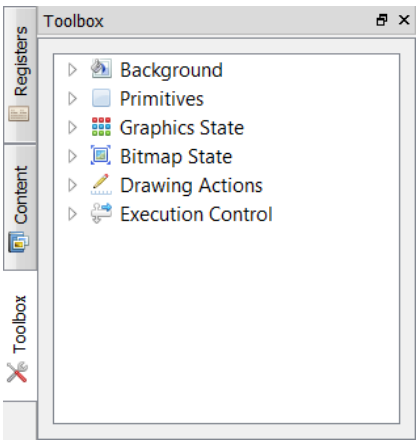


Figure 22 - Toolbox in Display List mode

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

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

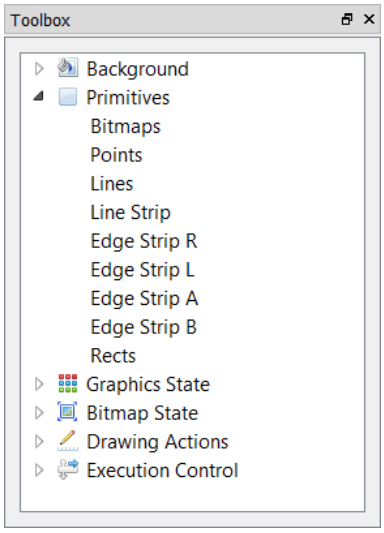


Figure 23 – Primitive in Display List Mode

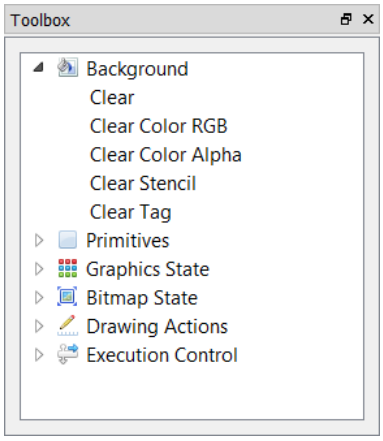


Figure 24 – Background in Display List mode

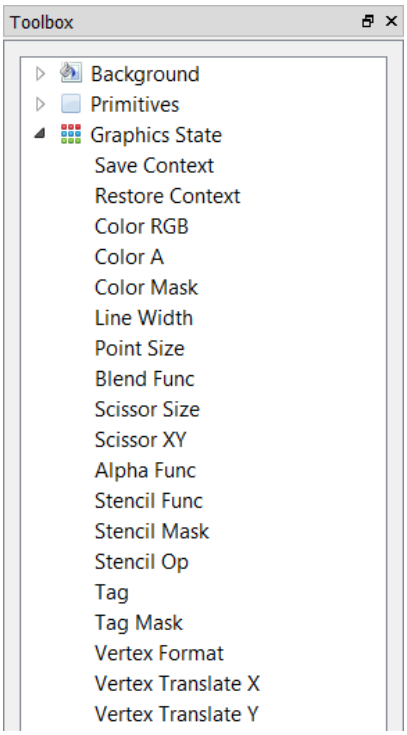


Figure 25 - Graphics State in Display List mode

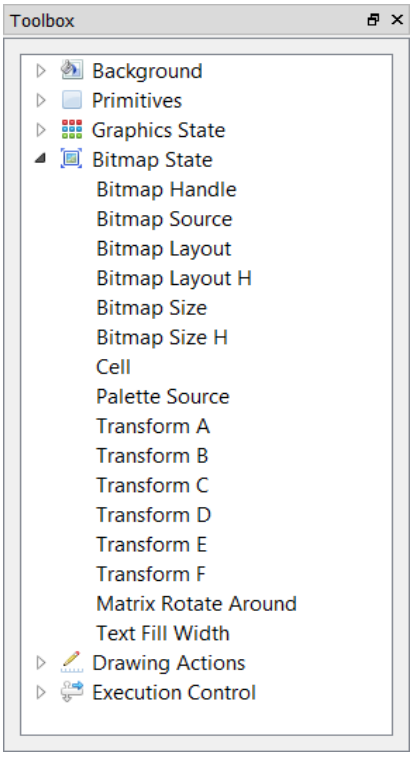


Figure 28 – Bitmap State in Display List mode

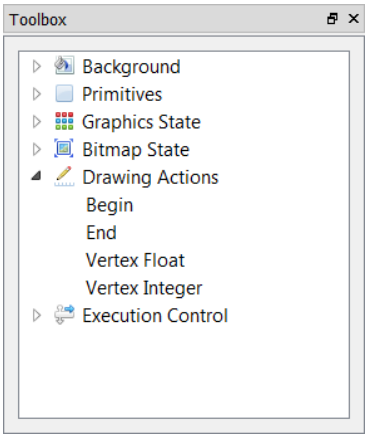


Figure 27 - Drawing Actions in Display List mode

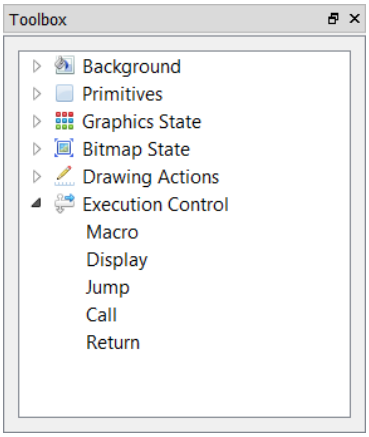


Figure 26 - Execution Control in Display List mode

Coprocessor mode

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

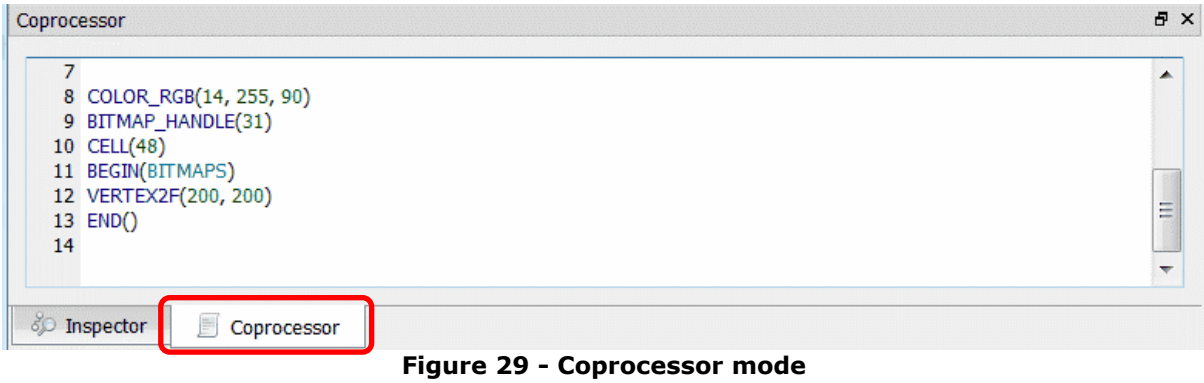


Figure 29 - Coprocessor mode

The toolbox will be enabled as below:

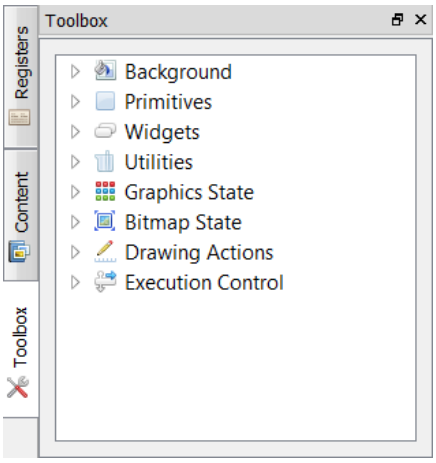


Figure 30 - Toolbox in Coprocessor mode

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

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

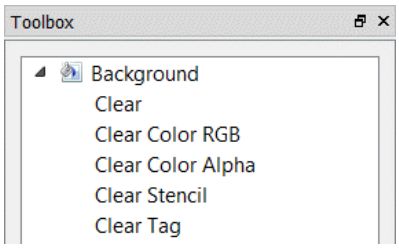


Figure 31 - Background in Coprocessor mode

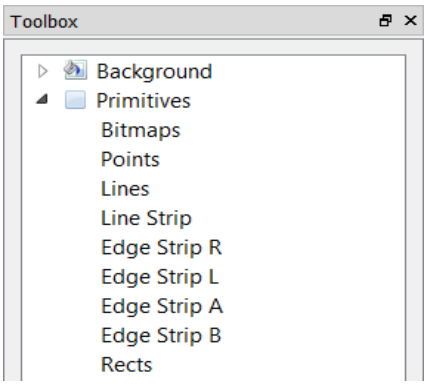


Figure 32 - Primitives in Coprocessor mode

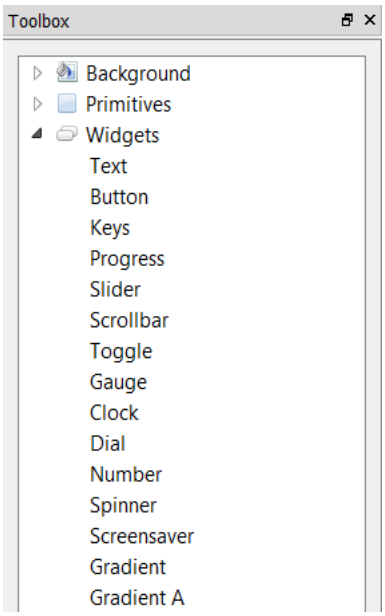


Figure 34 - Widgets Coprocessor mode

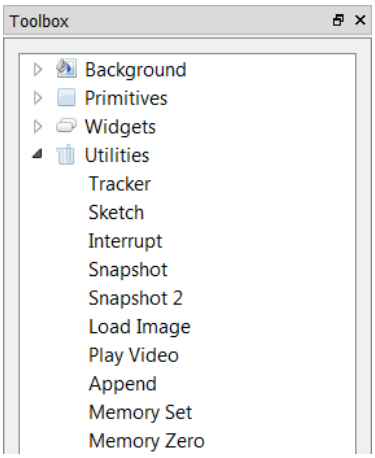


Figure 33 - Utilities in Coprocessor mode

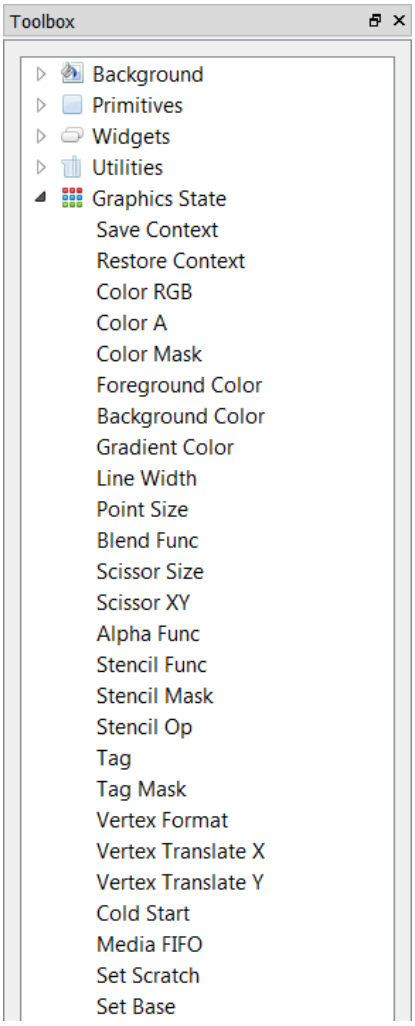


Figure 38 - Graphics State in Coprocessor mode

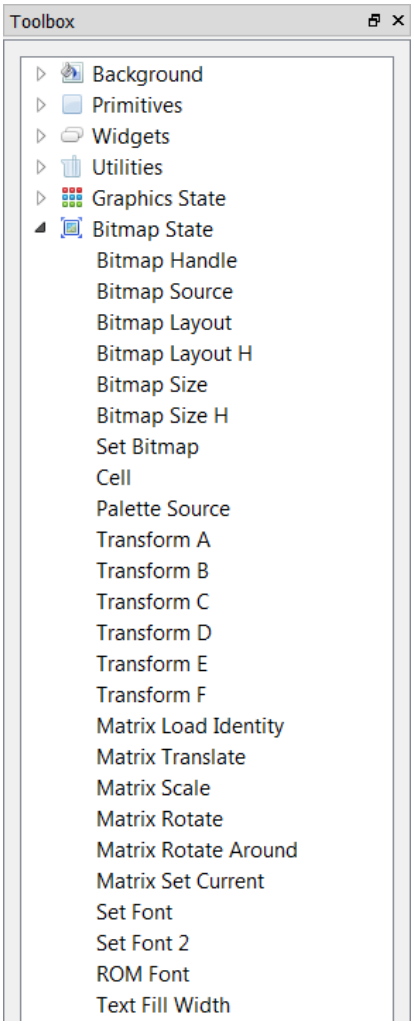


Figure 37 – Bitmap State in Coprocessor mode

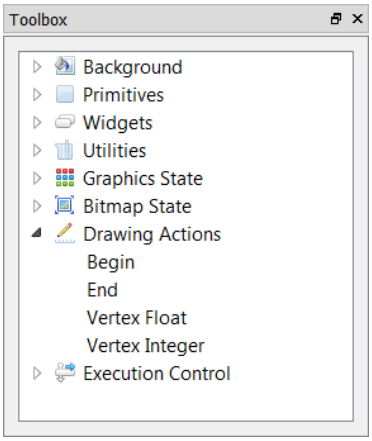


Figure 35 – Drawing Actions in Coprocessor mode

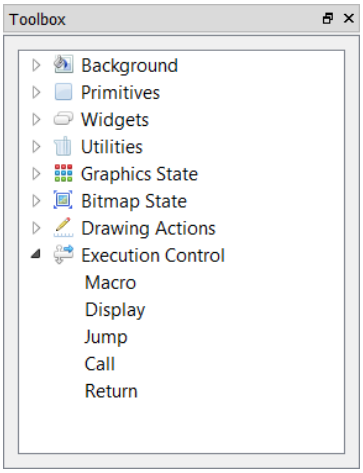


Figure 36 – Execution Control in Coprocessor mode

2. Registers

This tab is used to set up screen size and macro registers. The REG_MACRO0 and REG_MACRO1 registers can be edited in the editor box of Macro, the registers should be set using the display list command syntax. The vertical and horizontal size (REG_VSIZE and REG_HSIZE) of the screen can also be edited and the viewport will be updated accordingly.

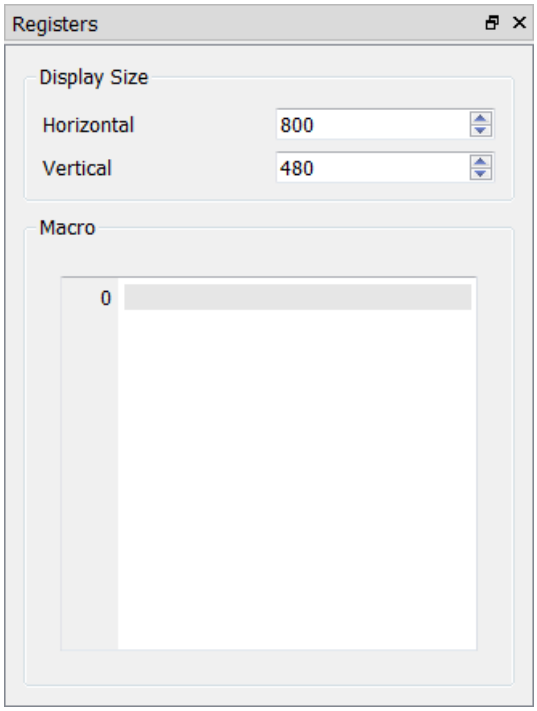


Figure 39 – Register

ESE has the ability to 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.

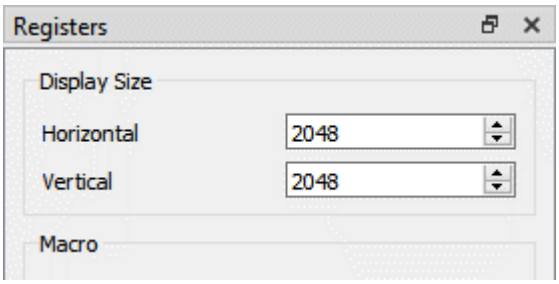


Figure 40 - Customize resolution in Register window

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 PC to RAM_G by converting the format behind the scenes.

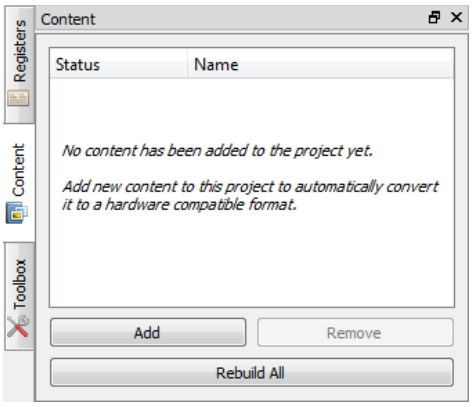


Figure 41 - Content Manager

Add the content

Content Manager enables users to add bitmap and raw data to be loaded into the specific addresses in RAM_G.

To perform this function, follow the steps below:

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

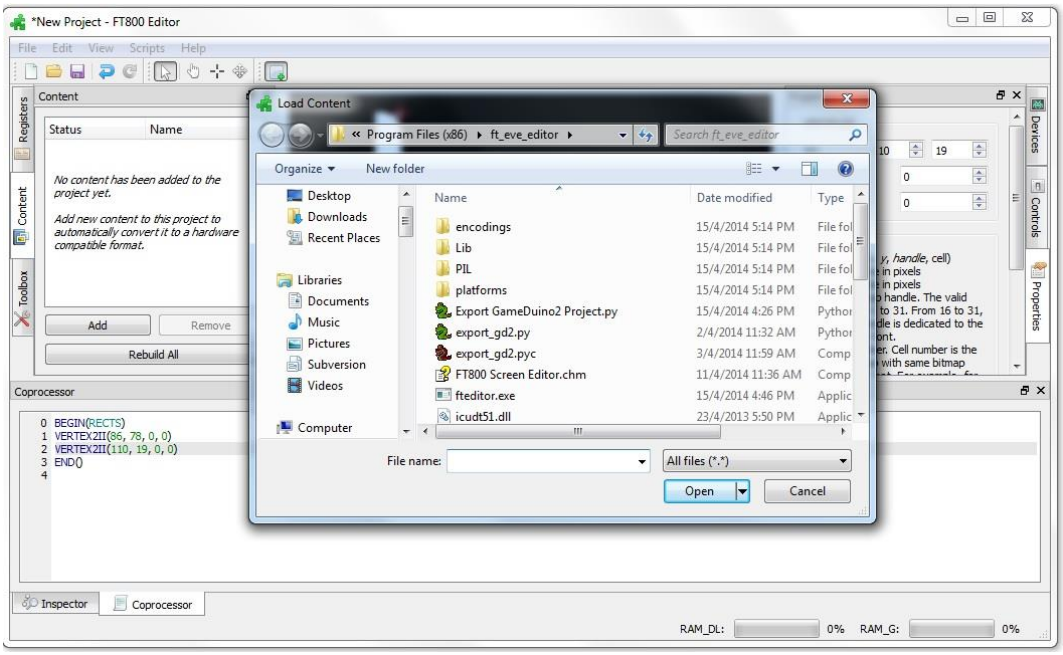


Figure 42 - Load content dialog

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.

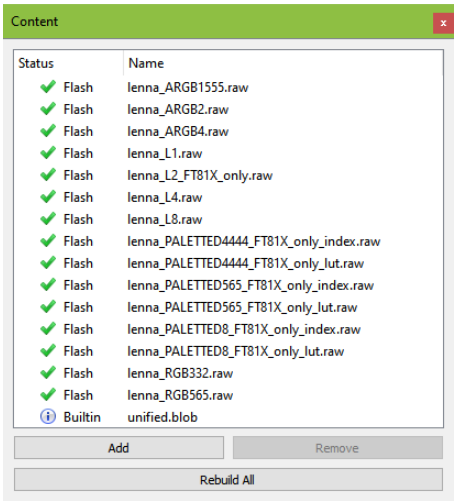


Figure 43 - Content loaded

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 option. Please note that the converted data is stored in the same directory as the original image.

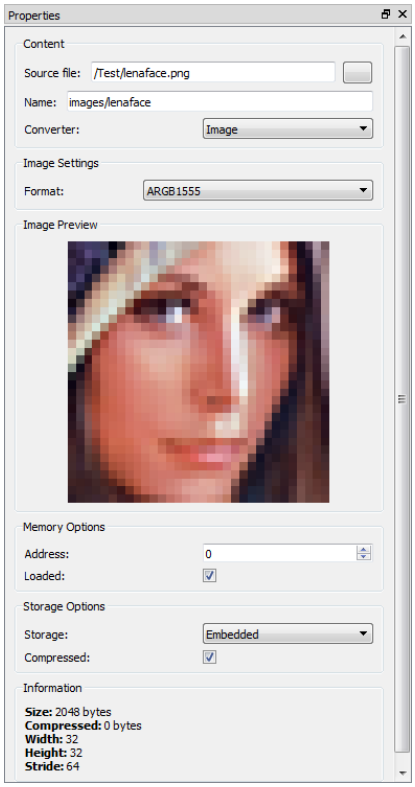


Figure 44 - Content properties

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

"Start" edit box as well as the length of data to be imported in the "Length" edit box.

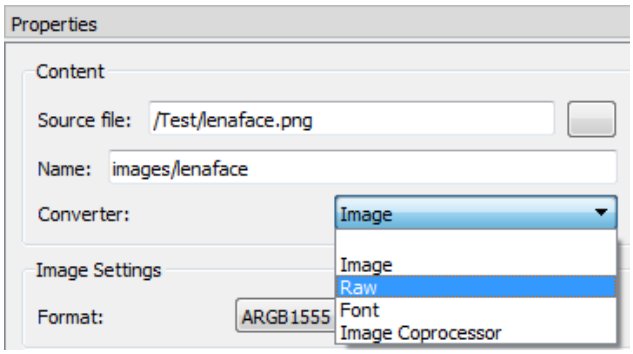


Figure 45 - Raw converter

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

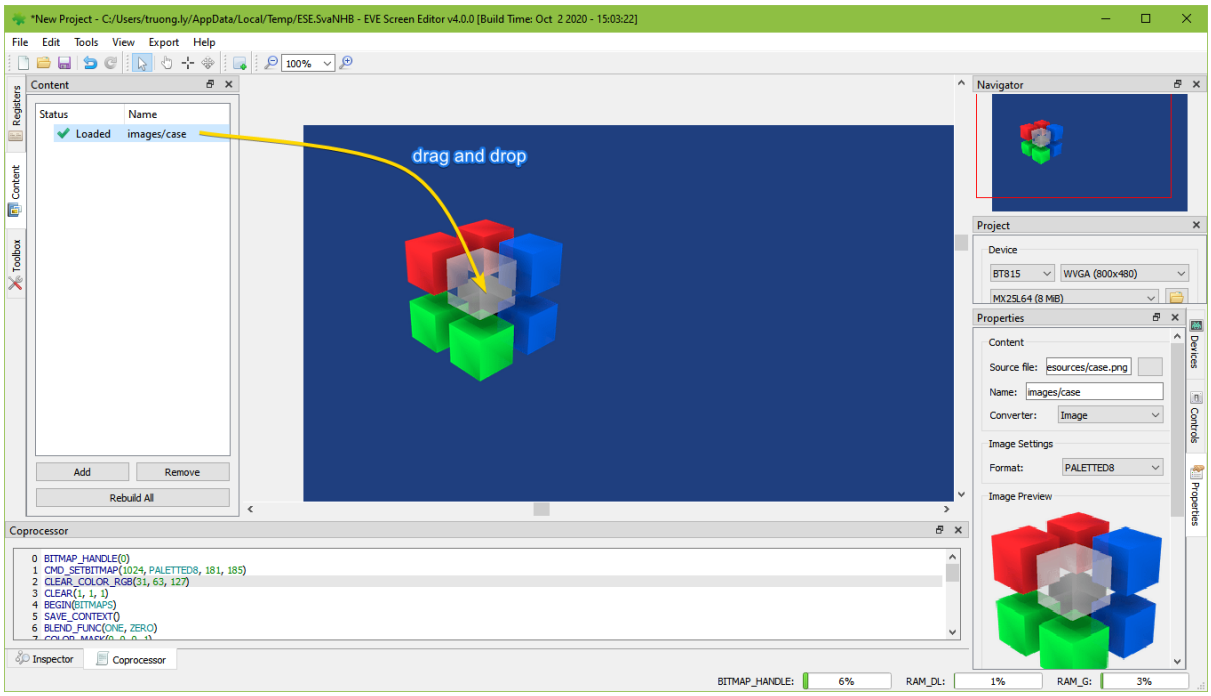


Figure 46 - Drag content into Viewport

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

Remove added 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]**.

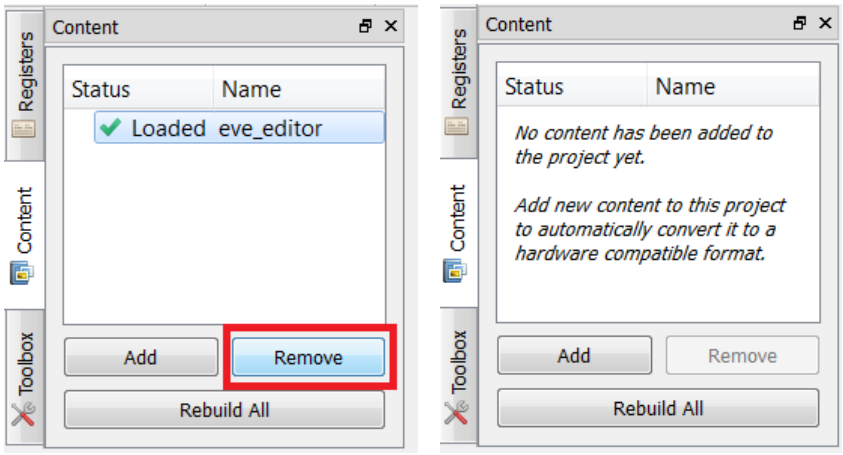


Figure 47 - Remove content

- The selected content is then removed from the list.

Rebuild the content

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

E. Devices, Controls and Properties

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

1. Device Manager

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

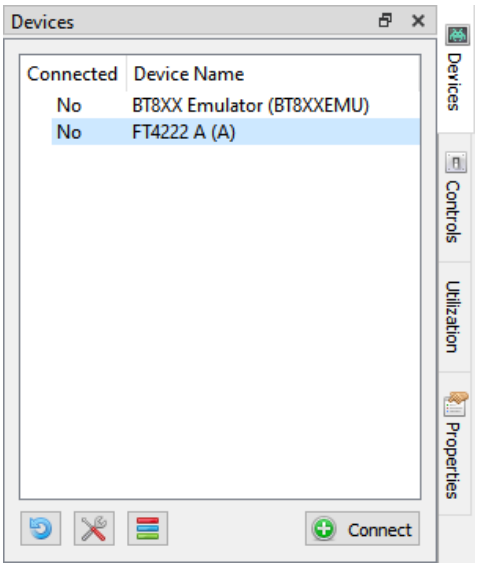


Figure 48 - Before connect

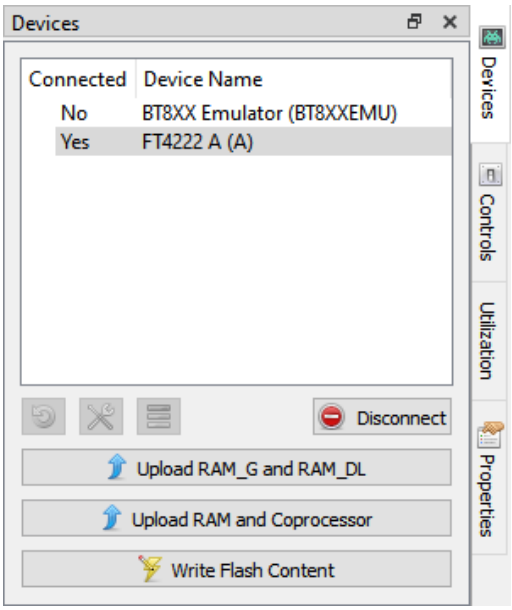


Figure 49 - Connected device

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

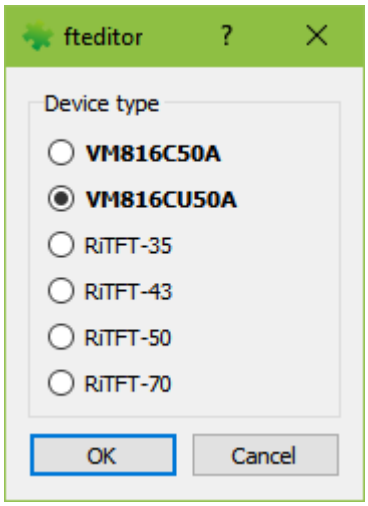


Figure 50 - Device type

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

2. Controls

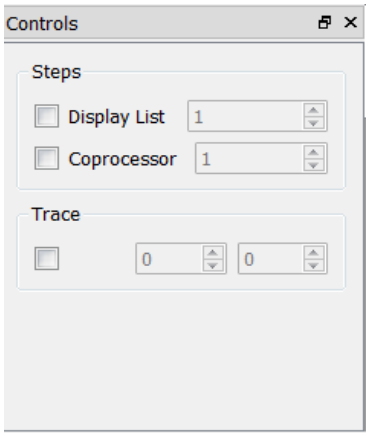


Figure 51 - Controls tab

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

See the "Steps" grouped widgets below.

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

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

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

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

3. Properties

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

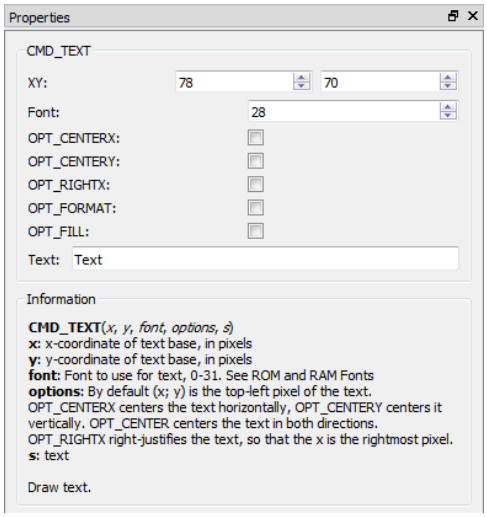
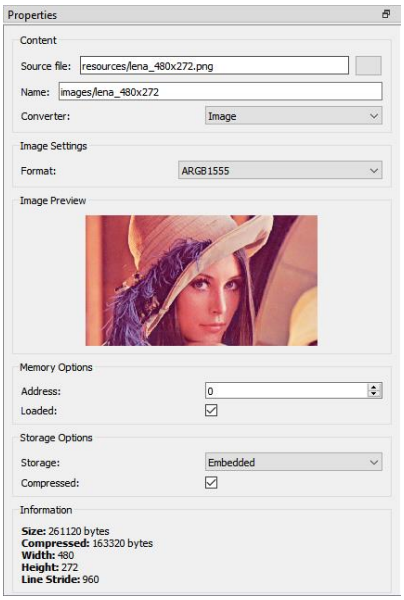


Figure 52 - Properties tab

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



F. View Port

This is the significant area in the center of the screen. When the user selects any components or commands in the Toolbox, those components can be visually seen in the view port. The view port has the same resolution as specified in REG_HSIZE and REG_VSIZE.

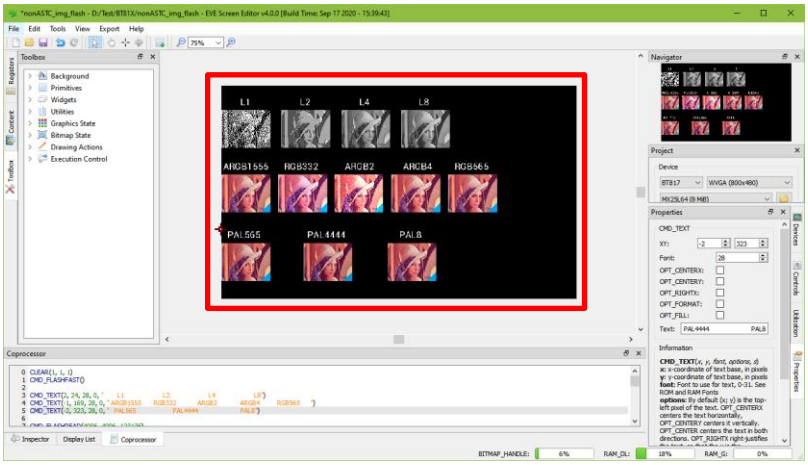


Figure 53 – View port

G. Navigator

Viewport navigator provides a convenient way to move the view port, especially for large resolutions.

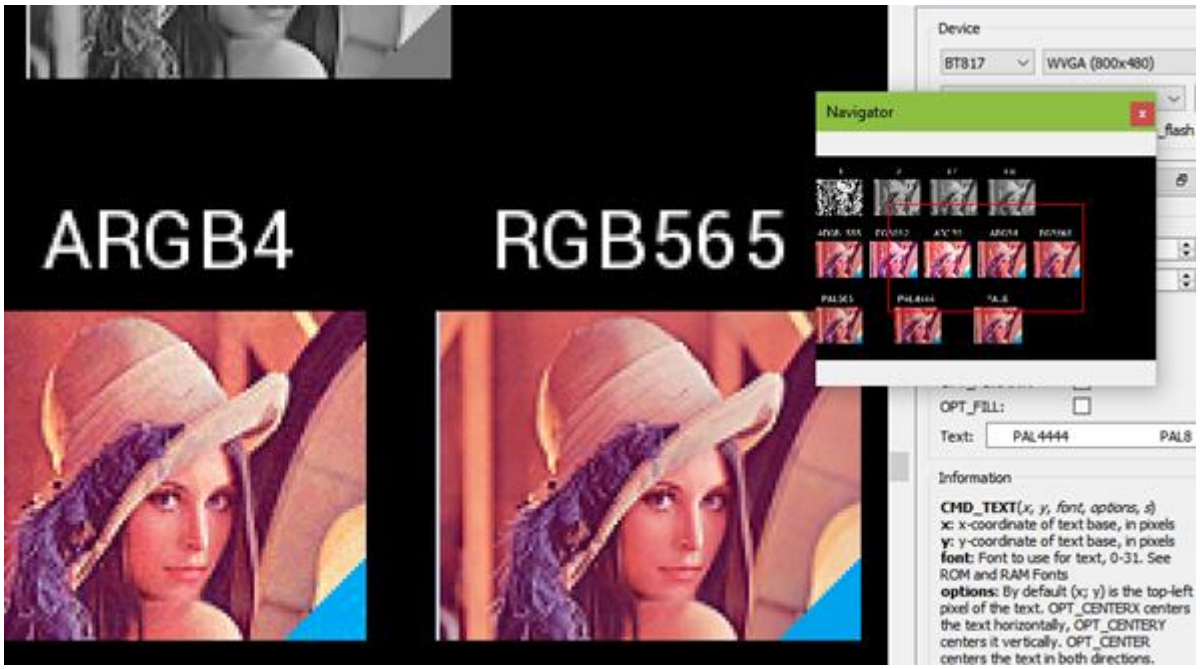


Figure 54 – Navigator

H. Project settings

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

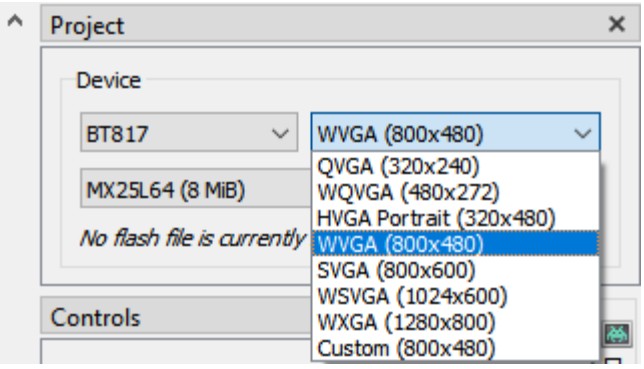


Figure 55 - Project settings

From BT815 device an above, flash image is supported.

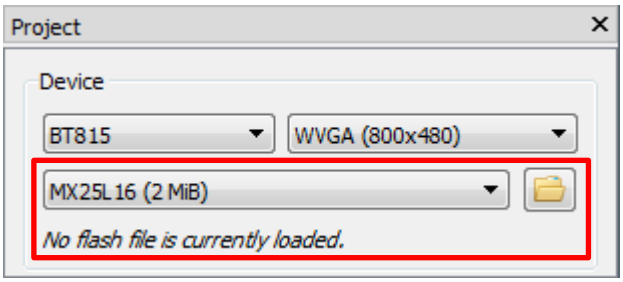


Figure 56 - Flash supported

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

Follow these steps to generate a flash image:

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

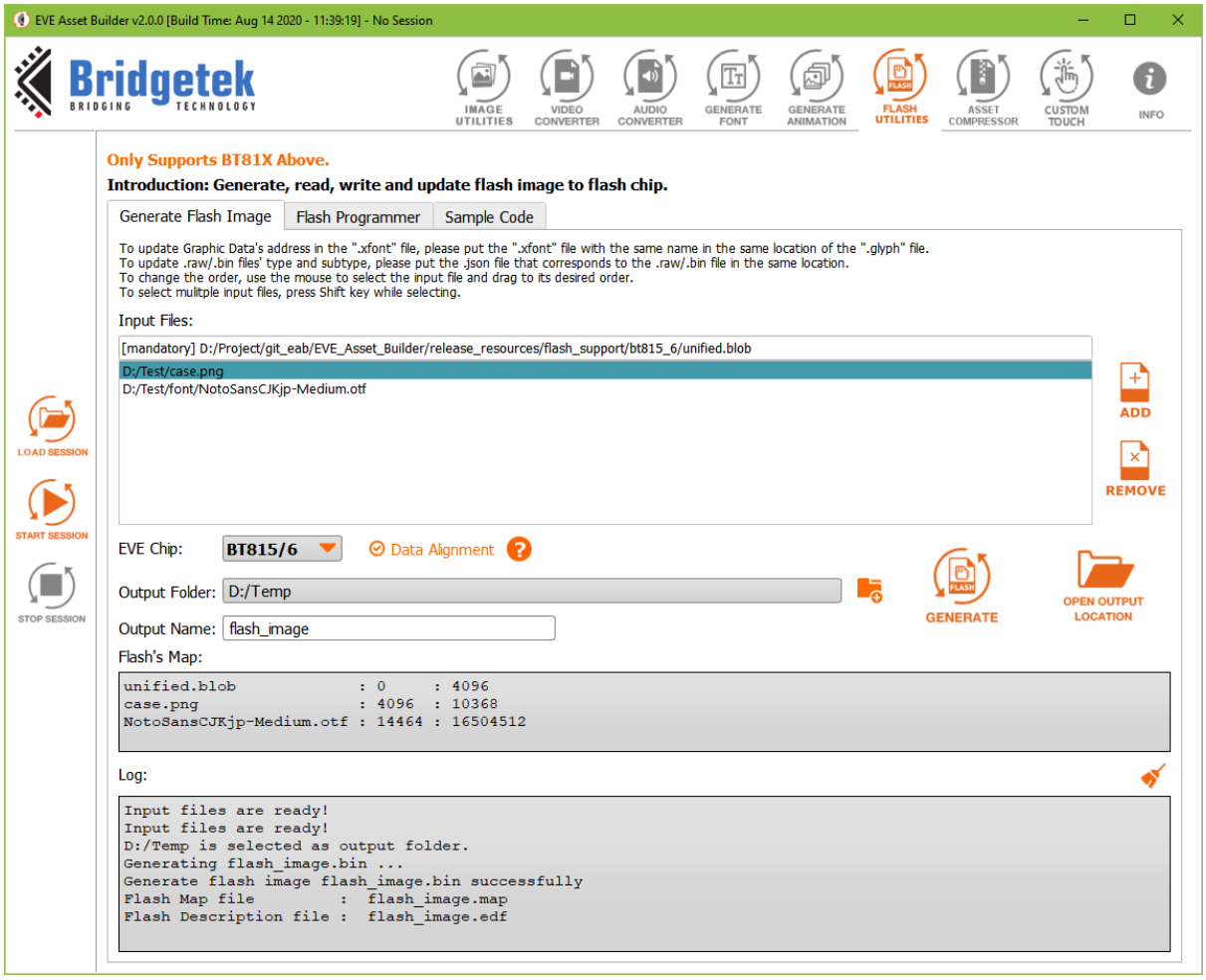


Figure 57 - Flash image generation by EAB

Name	Ext	Size
[..]		<DIR>
flash_image	bin	16,519,168
flash_image	edf	231
flash_image	map	128

Figure 58 - Generated files in output folder

Flash .bin file is the binary file which can be loaded into emulator as well as flash chip. Flash .map and .edf are human-readable text file. Each line shows the asset name, its beginning address and length.

```
flash_image.map
1  unified.blob              : 0      : 4096
2  case.png                  : 4096   : 10368
3  NotoSansCJKjp-Medium.otf  : 14464  : 16504512
```

Figure 59 - Content of a .map file

To load a flash image, click **File > Open** and select flash .map file in local PC.

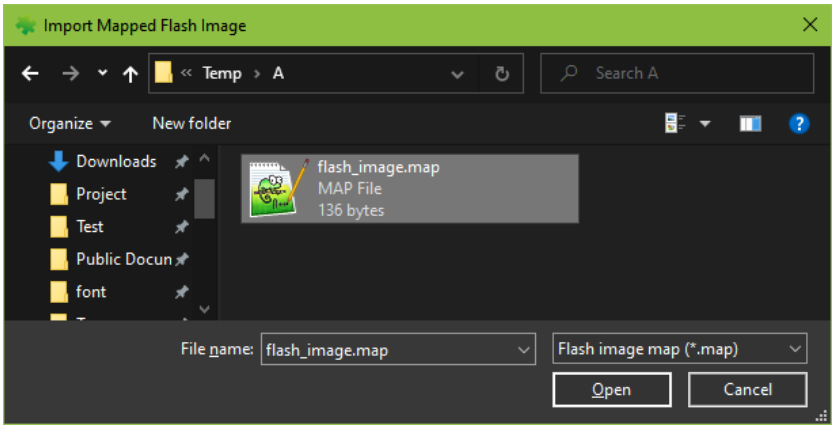


Figure 60 - Load flash file

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

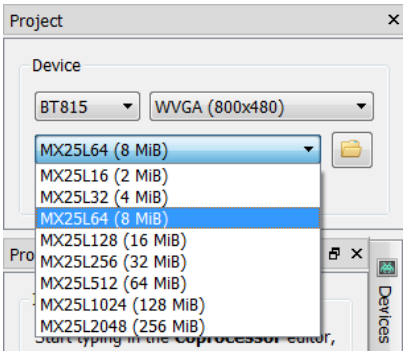


Figure 61 - Select flash size

Upon loading the flash file, its path is displayed.

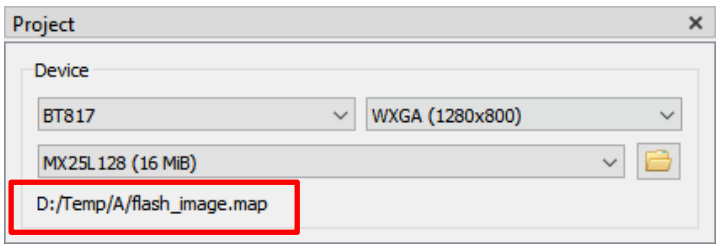


Figure 62 – Display flash path

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

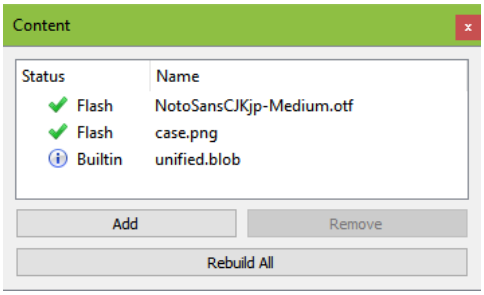


Figure 63 - Flash assets show in Content window

I. Keyboard Shortcuts

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

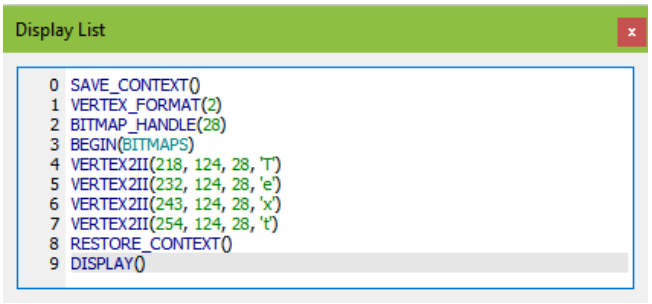
Item	Shortcut
New	Ctrl + N
Save	Ctrl + S
Undo	Ctrl + U
Redo	Ctrl + Y
Cut	Ctrl + X
Copy	Ctrl + C
Paste	Ctrl + V
Zoom In/Out of Viewport	Ctrl + Mouse wheel
Close Project	Ctrl + F4
Reset Emulator	Ctrl + R
Capture Display List	Ctrl + D
Open Recent Project 1->5	Alt + 1, Alt + 2, …, Alt + 5

V. Quick Start Tutorials

This section explains how to use the EVE screen editor. They are intentionally kept brief so that the user can actually 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 display list, select menu **Tools -> Capture Display List**, or use shortcut *Ctrl+D*. For example, users can type "CMD_TEXT" in co-processor editor, then press *Ctrl+D*. The display list commands will be displayed -



B. Change the color

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

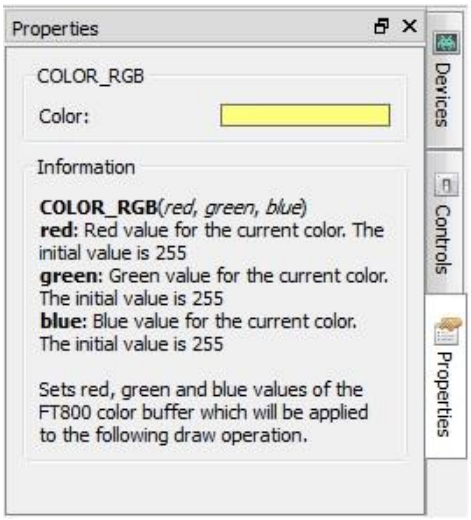


Figure 64 - Change the color

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

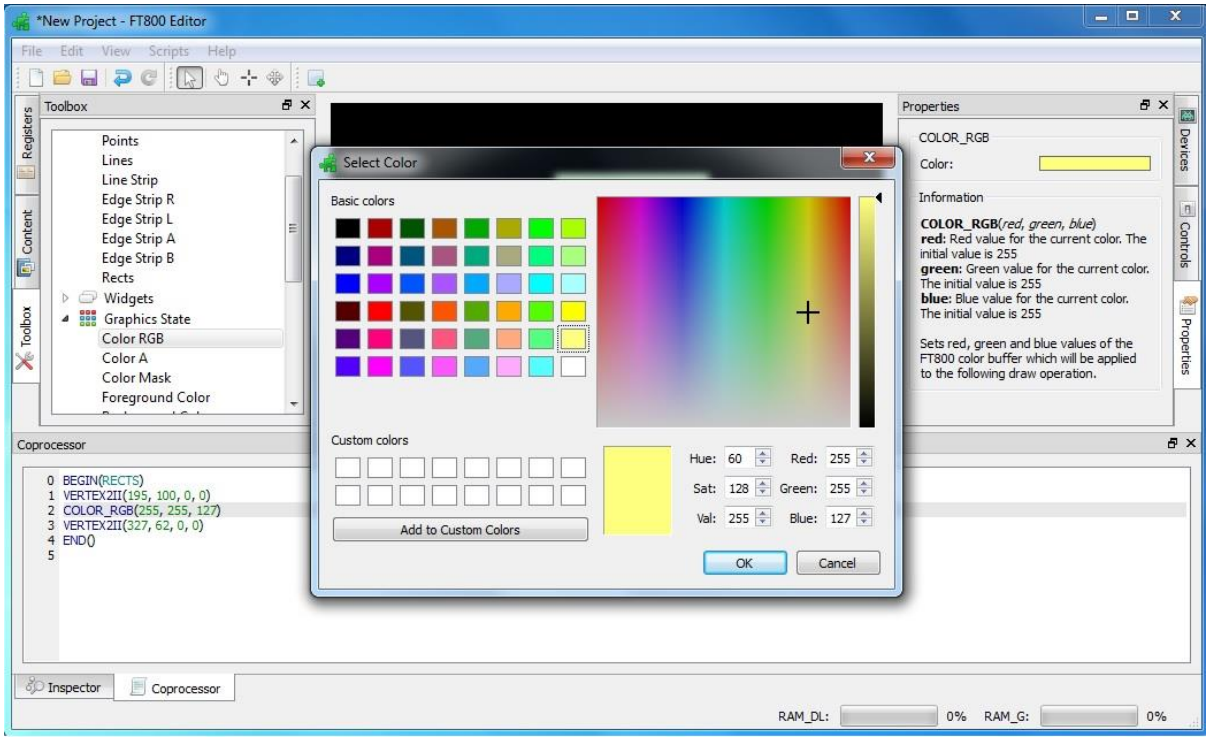


Figure 65 - Select color

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

- COLOR_RGB
- CLEAR_COLOR_RGB
- CMD_GRADIENT
- CMD_BGCOLOR
- CMD_FGCOLOR
- CMD_GRADCOLOR

C. Import the content

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

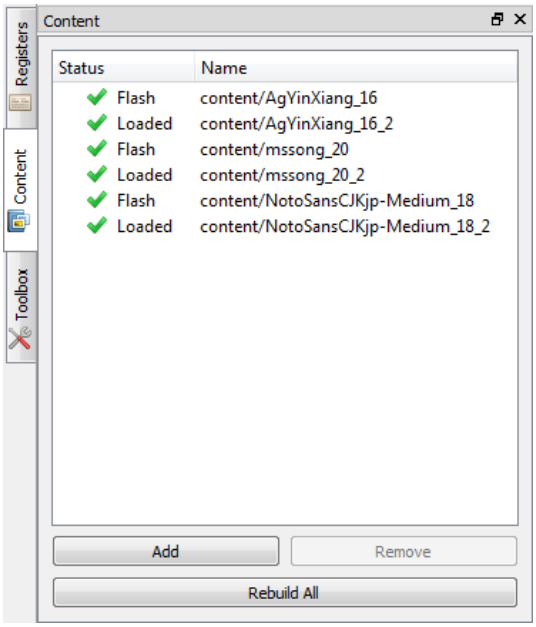


Figure 66 - Import content

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

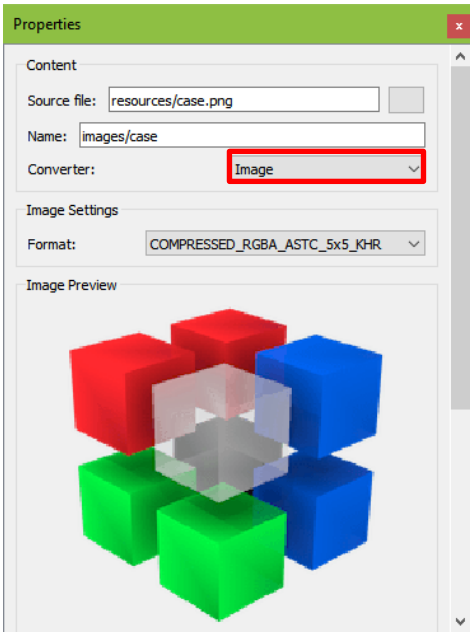


Figure 67 - Select converter image

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

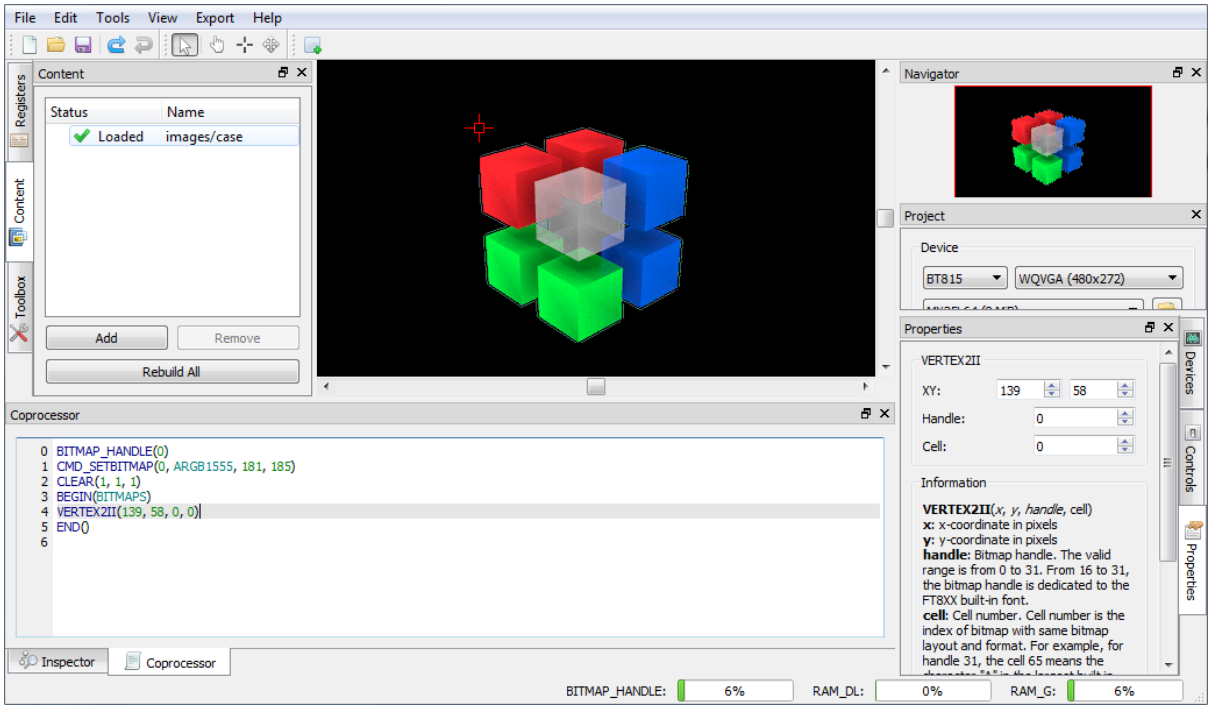


Figure 68 – Drag & drop image

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

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

*.raw	The binary format of converted file, which can be downloaded into RAM_G directly.
*.rawh	The header file of converted file, which is in text representation. Programmer can include this file into their program and build it into final binary.
*.bin	The compressed binary format of converted file in ZLIB algorithm. Programmer needs 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 text representation of *.bin. Programmer can include this file into their program and build it into final binary.

If the palette image format was chosen, files with the ".lut" text in the file name are generated and the appropriate file should be downloaded into RAM_PAL for FT80X or idle area in RAM_G for FT81X.

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

D. Import the flash

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

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

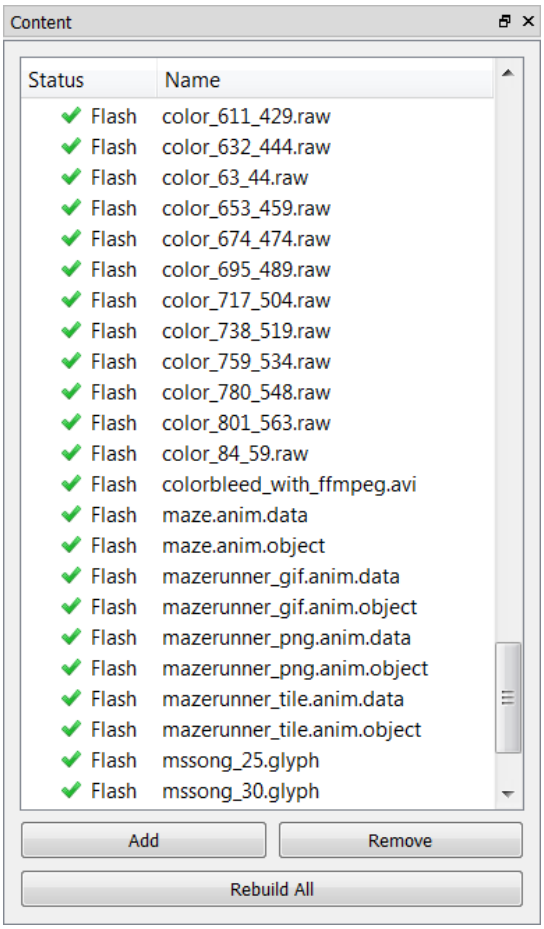


Figure 70 - Import flash

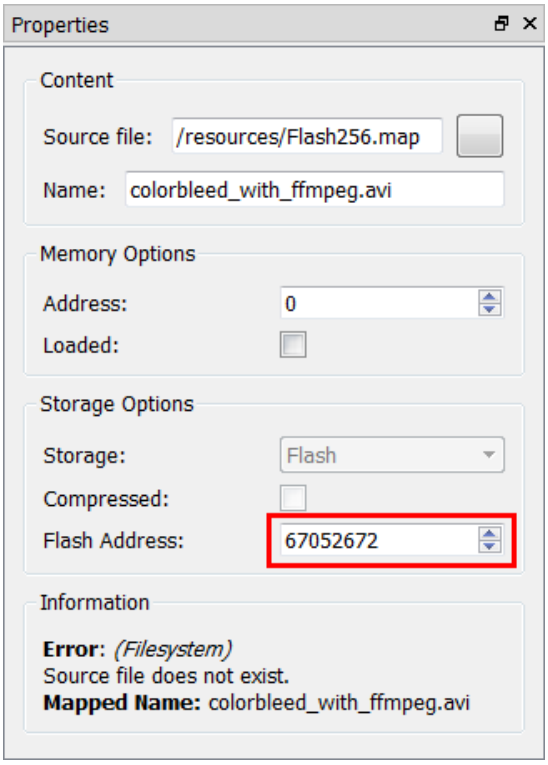


Figure 69 - Flash address

E. Open the project

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

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

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

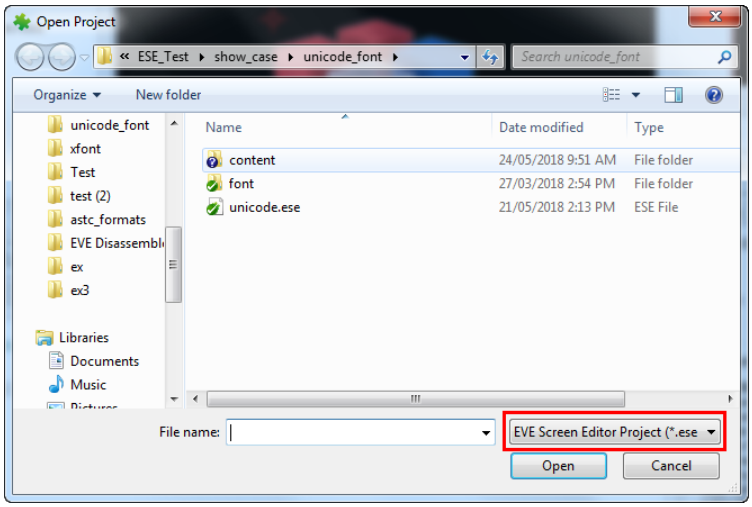


Figure 71 - Open project

F. Save your design

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

The saved project can be opened only by the EVE Screen Editor.

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

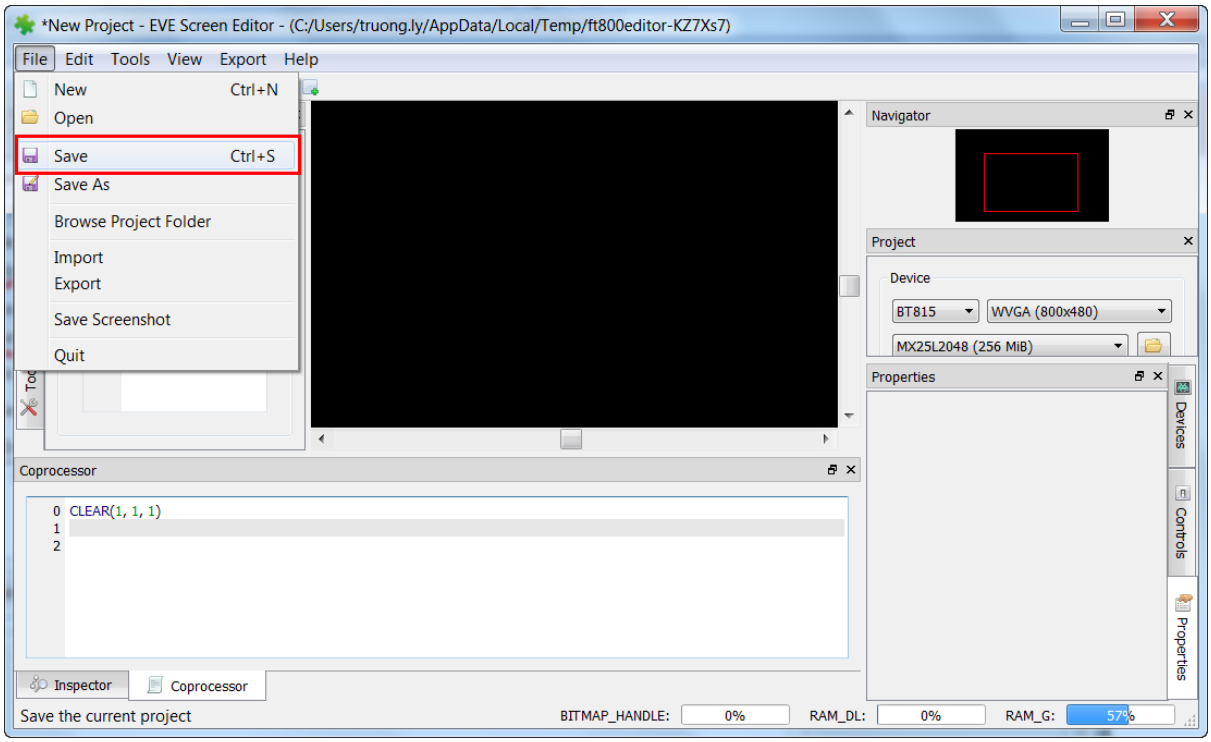


Figure 72 - Save the project

G. Export the project

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

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

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

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

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.

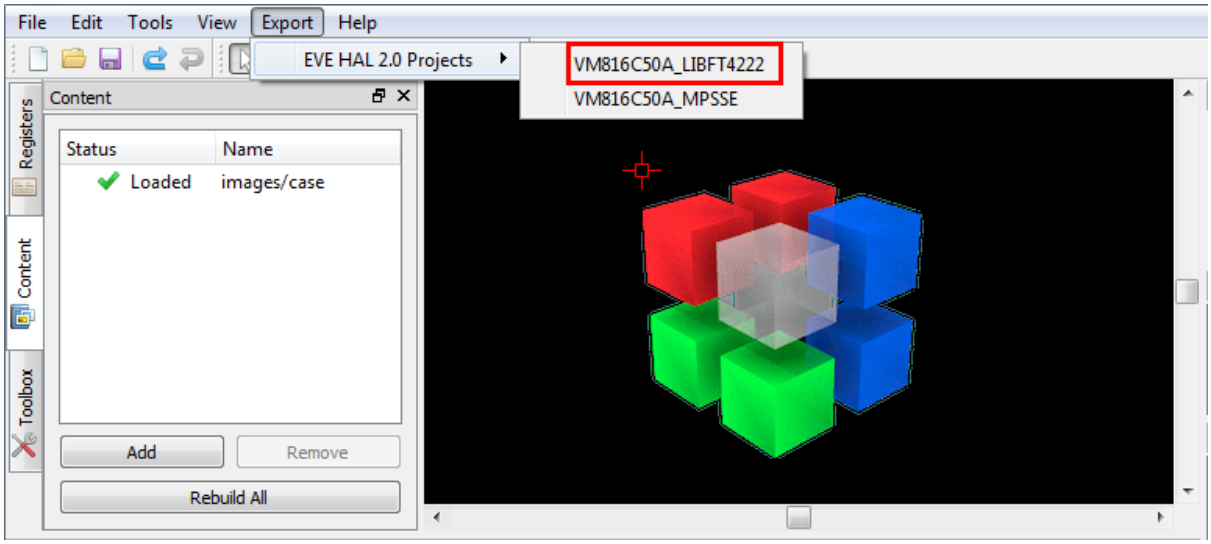


Figure 73 - Export project

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

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

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

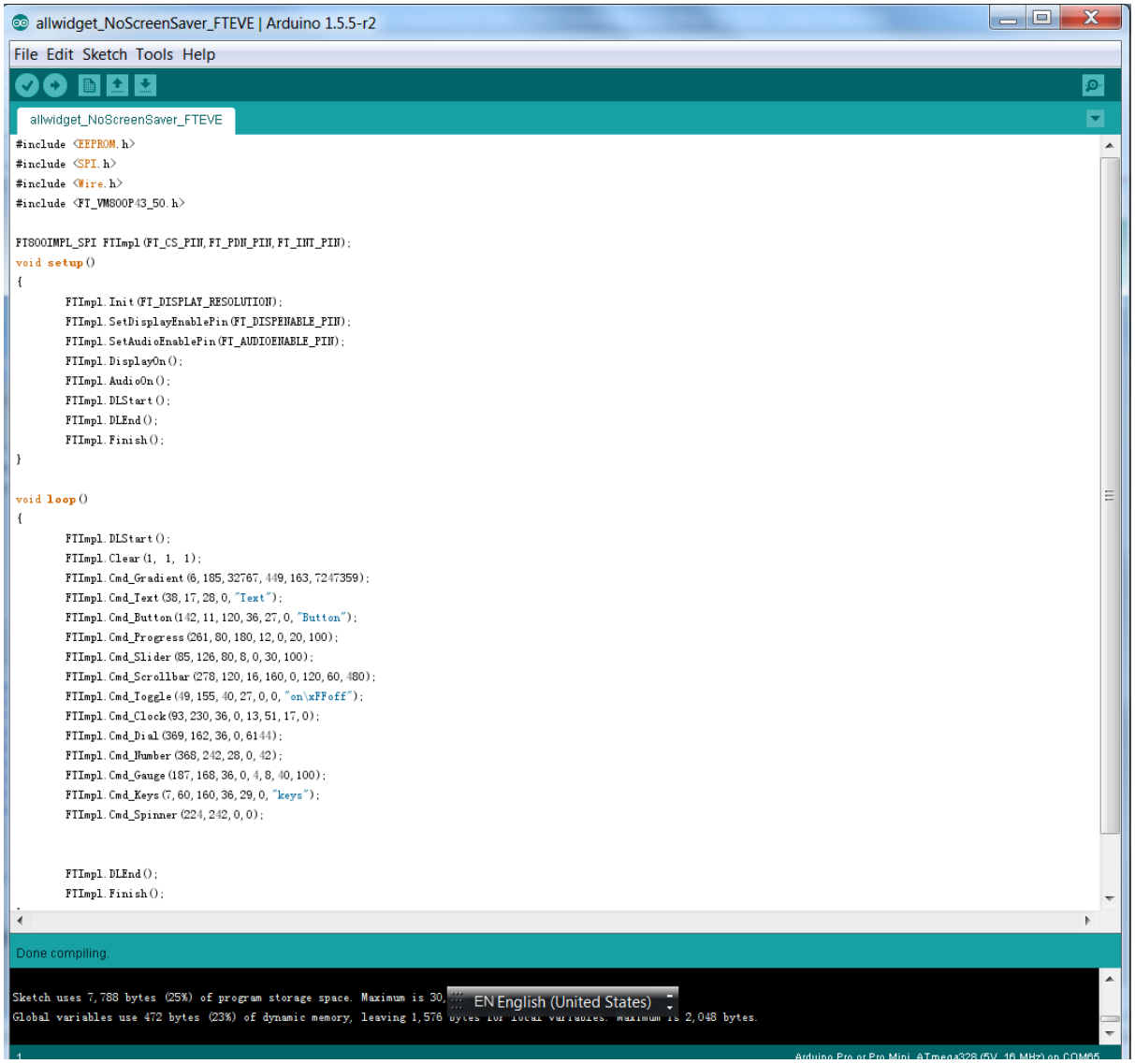


Figure 74 - Arduino IDE

H. Custom Fonts

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

To use custom fonts:

1. Load the custom font in the Content manager. Successfully loaded fonts have the "Loaded" status next to the font name.
2. Set the font format and size attributes.
3. Drag and drop the font to the Viewport.
4. Click the font object in the Viewport to edit the display text.

5. "CMD_SETFONT" and "CMD_SETFONT2" are generated accordingly for FT80X and FT81X/BT81X device to assign the new custom fonts with one unused bitmap handle. By default, the first unused bitmap handle is zero.

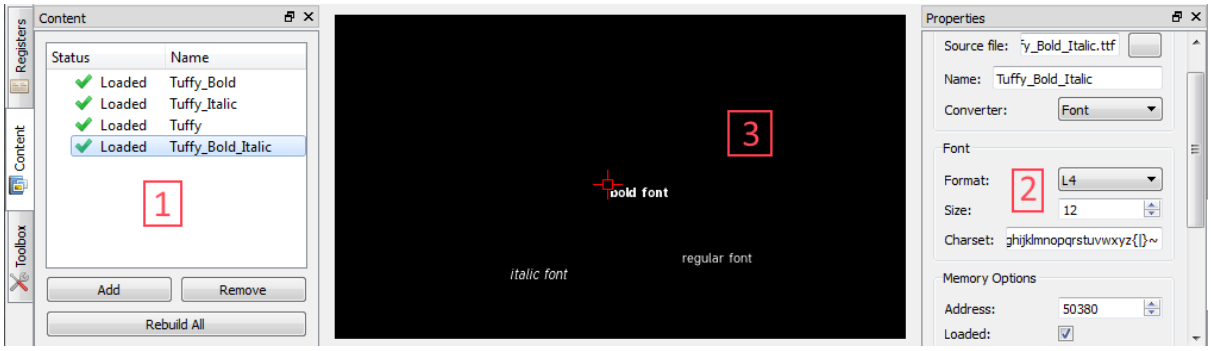


Figure 75 - Custom font

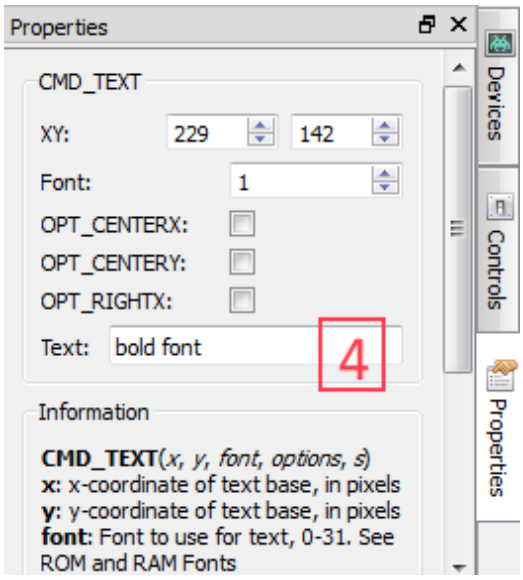


Figure 76 - Property "Text" of custom font

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

1. Constrain vertical positioning

- Prepare two objects which needs to align in the same x-coordinate
- Press SHIFT and left click to the aligned object
- Slide it up and down following the dashed red line

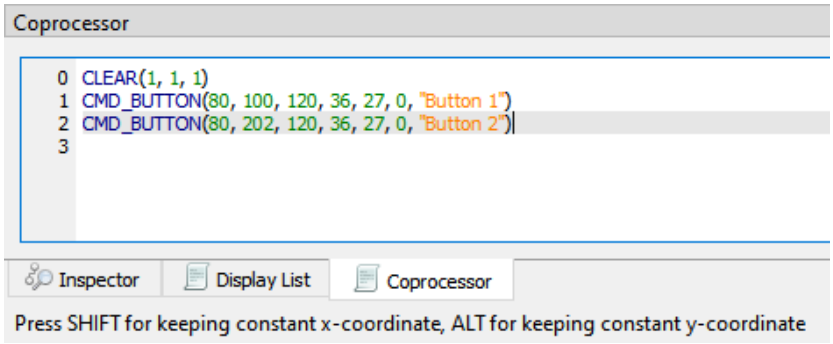


Figure 77 - Press SHIFT for constrain vertical

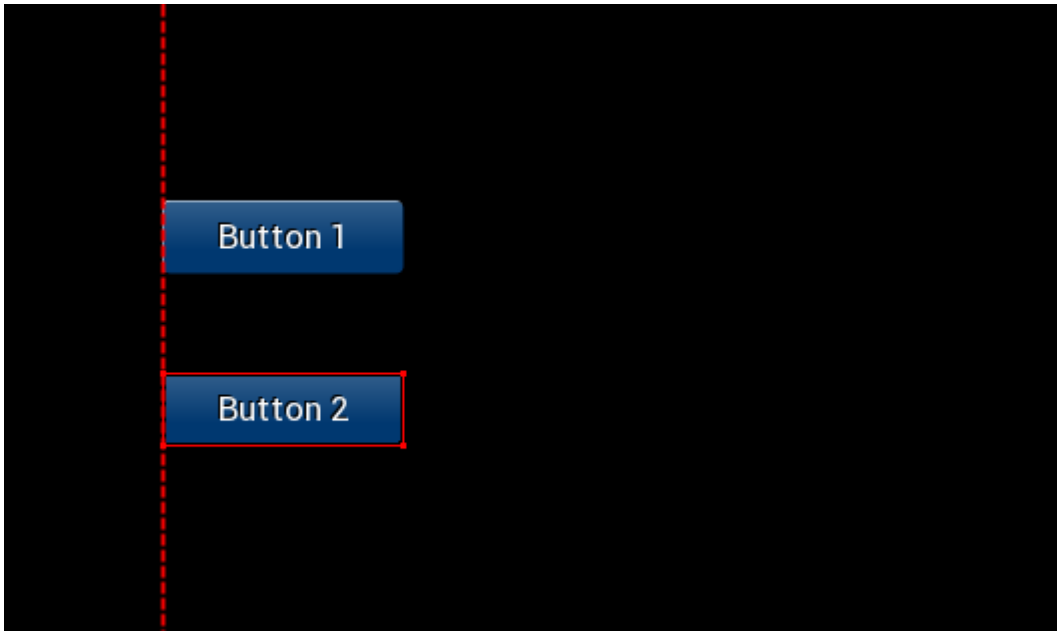


Figure 78 - Slide up/down to set y-coordinate

2. Constrain horizontal positioning

- Prepare two objects which needs to align in the same y-coordinate
- Press ALT and left click to the aligned object
- Slide it left and right follow the dashed red line

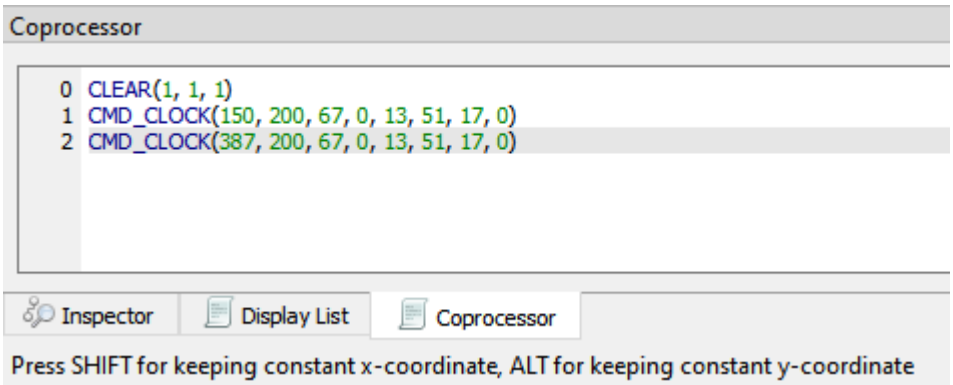


Figure 79 - Press ALT for constrain horizontal

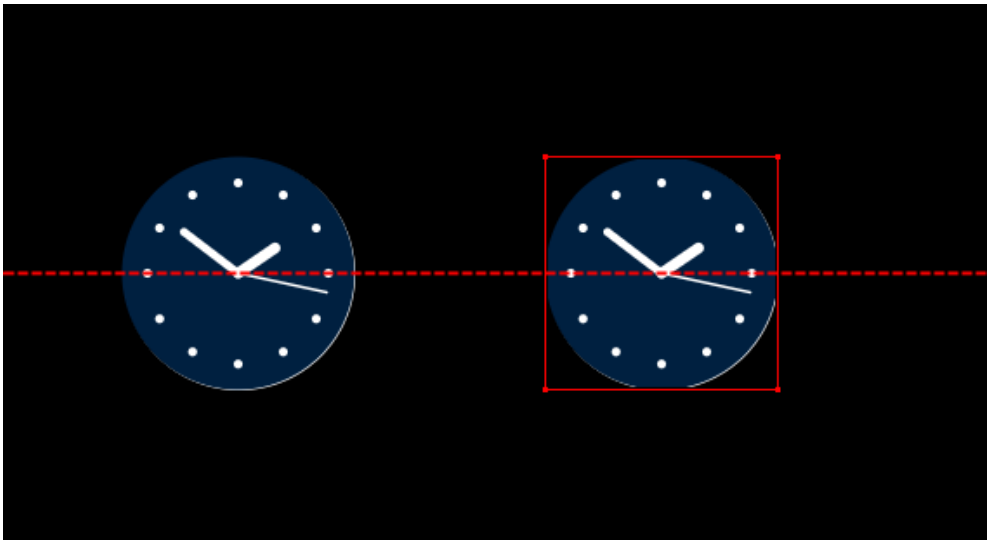


Figure 80 - Slide left/right to set x-coordinate

VI. Command Usage Examples

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

A. CMD_PLAYVIDEO

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

Prototype:

CMD_PLAYVIDEO(Option, Stream)

Parameter Description:

Option (one or more of the following) -
OPT_MONO - video playback in greyscale, L8, mode.
OPT_NOTEAR - Attempt to avoid horizontal "tearing" artifacts.
OPT_FULLSCREEN - Zoom the video so that it fills as much of the screen as possible.
OPT_MEDIAFIFO - Instead of sourcing the AVI video data from the command buffer, source it from the media FIFO. If this option is check, CMD_MEDIAFIFO must be specified before CMD_PLAYVIDEO.
OPT_SOUND - Video playback with sound.

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

Example:

```
CMD_PLAYVIDEO(OPT_FULLSCREEN | OPT_SOUND, "../chickens-4.avi")
CMD_DLSTART()
CMD_TEXT(154, 212, 31, 0, "Video playback has ended.")
```

Note:

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

B. CMD_LOADIMAGE

CMD_LOADIMAGE decompresses the specified JPEG or PNG data into an FT81X bitmap, in RAM_G. PNG image source can only be specified in FT81X or BT81X devices.

Prototype:

CMD_LOADIMAGE(Address, Options, Stream)

Parameters description:

Address - The starting location in RAM_G where the command will put the decoded data.
Options (one or more of the following):

OPT_MONO - Decode the image to mono, L8, format.
OPT_NODL - The command will not insert the default display commands in the display list buffer. The command will simply decode the file to the specified location and format.
OPT_MEDIAFIFO - Use a mediafifo in RAM_G as a buffer for decoding, instead of the coprocessor buffer. Otherwise, Mediafifo is not required to decode a bitmap file.
CMD_MEDIAFIFO must be specified prior to this command.
Stream - The absolute or relative path from the project of the image to be decoded.

Example:

```
CLEAR(1, 1, 1)
/*decode Eiffel Tower jpeg image.  Put data to the 0th offset in RAM_G and
use default options*/
CMD_LOADIMAGE(0, 0, "../EiffelTower_800_480.jpg")

BEGIN(BITMAPS)
VERTEX2II(0, 0, 0, 0)
END()

/*decode lenna256 png image.  Put data after the Eiffel Tower decoded data
and use mediafifo buffer for decoding*/
CMD_LOADIMAGE(768000, 0, "../lenna256.png")
BEGIN(BITMAPS)
VERTEX2II(413, 170, 0, 0)
END()
```

Additional Information:

Currently, CMD_LOADIMAGE is a standalone command where the location of the decoded data is manually specified, but the application doesn't know the offset and amount of the space which the decoded data will occupy. Users can "reserve" the RAM_G space, so other assets in the content manager will not overwrite the data by loading the intended image in the content manager first with the final decoding format and RAM_G offset.

If BITMAP_HANDLE is used for other assets before the CMD_LOADIMAGE command, then it might overwrite the bitmap handle properties when the OPT_NODL is not selected, because the BITMAP_HANDLE value is part of the context and CMD_LOADIMAGE don't insert a BITMAP_HANDLE command. Manually adding a BITMAP_HANDLE with an unused handle before CMD_LOADIMAGE might be needed to prevent re-association of the last specified bitmap handle.

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

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

Grayscale loads as L8,
Truecolor loads as RGB565,
Indexed loads as PALETTED4444, if the image contains transparency, or PALETTED565 otherwise,
Grayscale with alpha is not supported,
Truecolor with alpha loads as ARGB4

C. CMD_SETBITMAP

This command generate the corresponding display list commands (BITMAP_SOURCE, BITMAP_LAYOUT, BITMAP_SIZE) for the given bitmap information, sparing the effort of writing the display list manually. This command is supported in FT81X or BT81X devices.

Prototype:

CMD_SETBITMAP(Address, Format, Width, Height)

Parameter Description:

Address - The address in RAM_G where the bitmap data starts.
Format - One of the device supported bitmap format.
Width - The width of the bitmap.
Height - The height of the bitmap.

Example:

```
BITMAP_HANDLE(0)
CMD_SETBITMAP(0, RGB565, 800, 480)
BEGIN(BITMAPS)
VERTEX2II(0, 0, 0, 0)
END()
```

Additional Information:

If the bitmap is bigger than 512 pixels in either dimension, CMD_SETBITMAP will also insert BITMAP_LAYOUT_H and/or BITMAP_SIZE_H command(s) with the appropriate parameter values.

The parameters filter/wrapx/wrapy in BITMAP_SIZE are always set to NEAREST/BORDER/BORDER value in the generated display list commands.

D. CMD_SNAPSHOT

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

Prototype:

CMD_SNAPSHOT(Address)

Parameter Description:

Address - The address in RAM_G where the device will put the captured bitmap data.

Example:

```
CLEAR(1, 1, 1)
CMD_BUTTON(10, 14, 120, 36, 27, 0, "Button")
CMD_KEYS(8, 65, 160, 36, 29, 0, "keys")
CMD_TEXT(145, 22, 28, 0, "Text")
CMD_SNAPSHOT(0)
BITMAP_HANDLE(1)
BITMAP_SOURCE(0)
BITMAP_LAYOUT(ARGB4, 960, 200)
BITMAP_SIZE(NEAREST, BORDER, BORDER, 480, 200)
BEGIN(BITMAPS)
VERTEX2II(197, 116, 1, 0)
END()
```

Additional Information:

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

E. CMD_SKETCH

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

Prototype:

CMD_SKETCH(X, Y, W, H, Address, Format)

Parameter Description:

X, Y - The coordinates of top left pixel of sketching area
W, H - The width and height of sketching area.
Address - The address in RAM_G where the device will put the bitmap data.
Format - L8 or L1

Example:

```
//To run on Screen Editor:
//Click the hand button in the menu bar then "draw" on the display area.

//To run on hardware:
//1] Perform a screen calibration after setup
//2] Make sure the following generated code will only run once.

CLEAR(1,1,1)
CMD_MEMSET(0,0,130560)
BITMAP_HANDLE(0)
BITMAP_LAYOUT(L8,480,272)
BITMAP_SIZE(NEAREST,BORDER,BORDER,480,272)
BITMAP_SOURCE(0)

CMD_SKETCH(0,0,480,272,0,L8)

BEGIN(BITMAPS)
VERTEX2II(0, 0, 0, 0)
END()
```

Additional Information:

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

F. CMD_SNAPSHOT2

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

Prototype:

CMD_SNAPSHOT2(Format, Address, X, Y, Width, Height)

Parameter Description:

Format - Captured data format, either RGB565, ARGB4, or ARGB8_SNAPSHOT.
Address - The address in RAM_G where the device will put the captured bitmap data.
X - The x-coordinate of the top left vertex of the intended capturing region.
Y - The y-coordinate of the top left vertex of the intended capturing region.
Width - The width of the intended capturing region.
Height - The height of the intended capturing region.

Example:

```
CLEAR(1, 1, 1)
CMD_BUTTON(10, 14, 120, 36, 27, 0, "Button")
CMD_KEYS(8, 65, 160, 36, 29, 0, "keys")
CMD_TEXT(145, 22, 28, 0, "Text")
CMD_SNAPSHOT2(RGB565,0,0,0,300,300)
BITMAP_HANDLE(1)
BITMAP_SOURCE(0)
BITMAP_LAYOUT(RGB565, 600, 300)
BITMAP_SIZE(NEAREST, BORDER, BORDER, 300, 300)
BEGIN(BITMAPS)
VERTEX2II(300, 300, 1, 0)
END()
```

Additional Information:

User can also use the **[Capture Snapshot]** in the "Properties" tab of the command to output the capture bitmap to a specified format raw file or as a JPEG or PNG image.

G. VERTEX_TRANSLATE_X/Y

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

Prototype:

```
VERTEX_TRANSLATE_X(Value)  //translation in the x-axis
VERTEX_TRANSLATE_Y(Value)  //translation in the y-axis
```

Parameter Description:

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

Example:

```
CLEAR(1, 1, 1)
VERTEX_TRANSLATE_X(320)  //translate all subsequent display objects by 20
pixels in the x-axis
VERTEX_TRANSLATE_Y(800)  //translate all subsequent display objects by 50
pixels in the y-axis
CMD_BUTTON(35, 45, 120, 36, 27, 0, "Button")
CMD_KEYS(34, 141, 160, 36, 29, 0, "keys")
CMD_TEXT(316, 56, 28, 0, "Text")
CMD_GAUGE(305, 135, 36, 0, 4, 8, 40, 100)
CMD_TOGGLE(205, 53, 40, 27, 0, 0, "on\xFFoff")
CMD_DIAL(201, 226, 36, 0, 6144)
VERTEX_TRANSLATE_X(0) //change back to default
VERTEX_TRANSLATE_Y(0)  //change back to default
```

Additional Information:

Both VERTEX_TRANSLATE_X and VERTEX_TRANSLATE_Y are part of the graphics context which means the value specified to the commands will have an effect for all subsequent drawing objects till the value has changed or *Tools->Reset Emulator* option is selected.

H. CMD_MEDIAFIFO

When project is in FT81X/BT81X mode, both CMD_PLAYVIDEO and CMD_LOADIMAGE commands have the option to utilize a mediafifo buffer in RAM_G to speed up the data loading process. Due to the fact that if option OPT_MEDIAFIFO is not selected then all data will be loaded to the co-processor buffer which is limited to a maximum of 4 kilobytes per transfer. The performance increase can be noticeably faster when running the exported project on the hardware.

Prototype:

CMD_MEDIAFIFO(ptr, size)

Parameter Description:

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

Example:

```
CLEAR(1, 1, 1)
CMD_MEDIAFIFO(768000, 20000)
CMD_LOADIMAGE(0, OPT_MEDIAFIFO, "..//EiffelTower_800_480.png")
BEGIN(BITMAPS)
VERTEX2II(0, 0, 0, 0)
END()
```

I. CMD_SETBASE

CMD_SETBASE sets the numeric base for CMD_NUMBER. This command can only be used in FT81X or BT81X devices.

Prototype:

CMD_SETBASE(Base)

Parameter Description:

Base - The numeric base, valid values are from 2 to 36. Common bases are:
2 - binary
8 - octal
10 - decimal
16 – hexadecimal

Example:

```
CLEAR(1, 1, 1)
CMD_SETBASE(2)  //set base to binary
CMD_NUMBER(88, 193, 29, 0, 65536)
```

J. CMD_ROMFONT

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

Prototype:

CMD_ROMFONT(Font, RomSlot)

Parameter Description:

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

RomSlot - The ROM fonts to be associated.

Example:

```
CLEAR(1, 1, 1)
CMD_ROMFONT(1, 31);  //associate bitmap font handle 31 to 1
CMD_TEXT( 0, 0, 1, 0, "31");
CMD_ROMFONT(1, 32);  //associate bitmap font handle 32 to 1
CMD_TEXT( 0, 60, 1, 0, "32");
CMD_ROMFONT(1, 33);  //associate bitmap font handle 33 to 1
CMD_TEXT(80,-14, 1, 0, "33");
CMD_ROMFONT(1, 34);  //associate bitmap font handle 34 to 1
CMD_TEXT(60, 32, 1, 0, "34");
```

Additional Information:

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

K. PALETTE_SOURCE

Palette look-up tables are located in the RAM_G. PALETTED_SOURCE allows the user to specify the look-up table for paletted asset. As a result, multiple look-up tables and index data files can be loaded.

PALETTED_SOURCE can only be use for FT81X or BT81X devices.

Prototype:

PALETTE_SOURCE(Addr)

Parameter Description:

addr - The starting address of the look-up table in RAM_G.

Example:

```
// drawing of an arbitrary paletted bitmap
PALETTE_SOURCE(0)
BITMAP_SOURCE(1024)
BITMAP_LAYOUT(PALETTED565, 80, 40)
BITMAP_SIZE(NEAREST, BORDER, BORDER, 40, 40)
BEGIN(BITMAPS)
VERTEX2F(0,0)
END()
```

Additional Information:

Palette look-up table has a maximum size of 1024 bytes. The value specified to PALETTE_SOURCE is part of the context.

L. CMD_SETFONT2

To use a custom font with the co-processor objects, create the font definition data in RAM_G and issue CMD_SETFONT2, as described in ROM and RAM Fonts. Note that CMD_SETFONT2 sets up the font's bitmap parameters by appending commands BITMAP_SOURCE, BITMAP_LAYOUT and BITMAP_SIZE to the current display list. Please note this command is only applicable to FT81X or BT81X devices.

Prototype:

CMD_SETFONT2(Font, Ptr, Firstchar)

Parameter Description:

font - The bitmap handle from 0 to 31
ptr - 32 bit aligned memory address in RAM_G of font metrics block
firstchar - The ASCII value of first character in the font. For an extended font block, this should be zero.

Example:

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

```
CLEAR(1, 1, 1)
CMD_FLASHFAST()
CMD_SETFONT2(0, 0, 32)
CMD_TEXT(50, 100, 0, 0, "あいうえお | \u3042\u3044\u3046\u3048\u304a")
```

M. CMD_TEXT

Draw text, can use string specifier.

Prototype:

CMD_TEXT(X, Y, Font, Options, Text)

Parameter Description:

x - x-coordinate of text base, in pixels
y - y-coordinate of text base, in pixels
font - Font to use for text, 0-31.

options - By default (x, y) is the top-left pixel of the text and the value of options is zero.
OPT_CENTERX centers the text horizontally, OPT_CENTERY centers it vertically.
OPT_CENTER centers the text in both directions.
OPT_RIGHTX right-justifies the text, so that the x is the rightmost pixel.
OPT_FORMAT processes the text as a format string, see String formatting.
OPT_FILL breaks the text at spaces into multiple lines, with maximum width set by CMD_FILLWIDTH.

text – Text string, UTF-8 encoding. If OPT_FILL is not given, then the string may contain newline (\n) characters, indicating line breaks.

Example:


```
CMD_TEXT(12, 49, 31, OPT_FORMAT, "BT%dX Programmer Guide", 81)
CMD_TEXT(11, 110, 31, OPT_FORMAT, "0x%X = %d", 2069, 2069)
```

VII. Working with EVE Screen Editor

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

A. Connect with Hardware

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

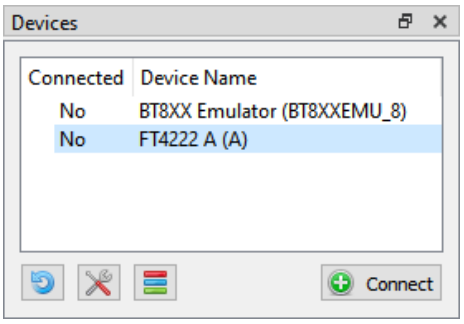


Figure 81 - Connect Device

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

Note 1: USB Hub will cause a failure in connection with FT4222 module. So, in case of using 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.

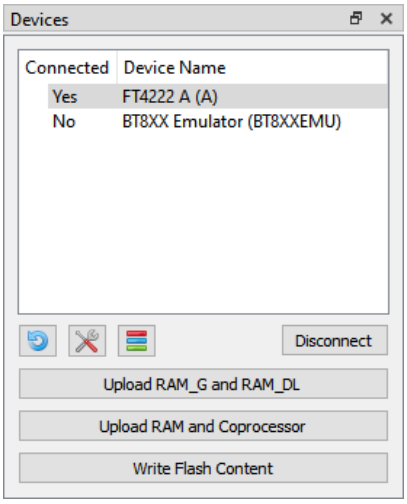


Figure 82 - Device connected

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

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

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

Check the picture below:

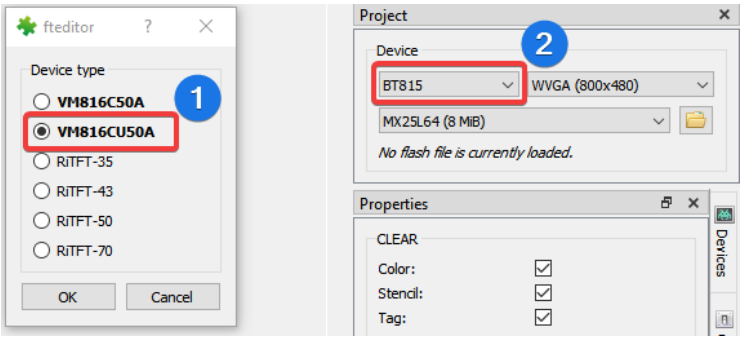


Figure 83 - Select correct device type

Managing device

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

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

Custom Device: User can Add/Edit/Clone/Remove device.

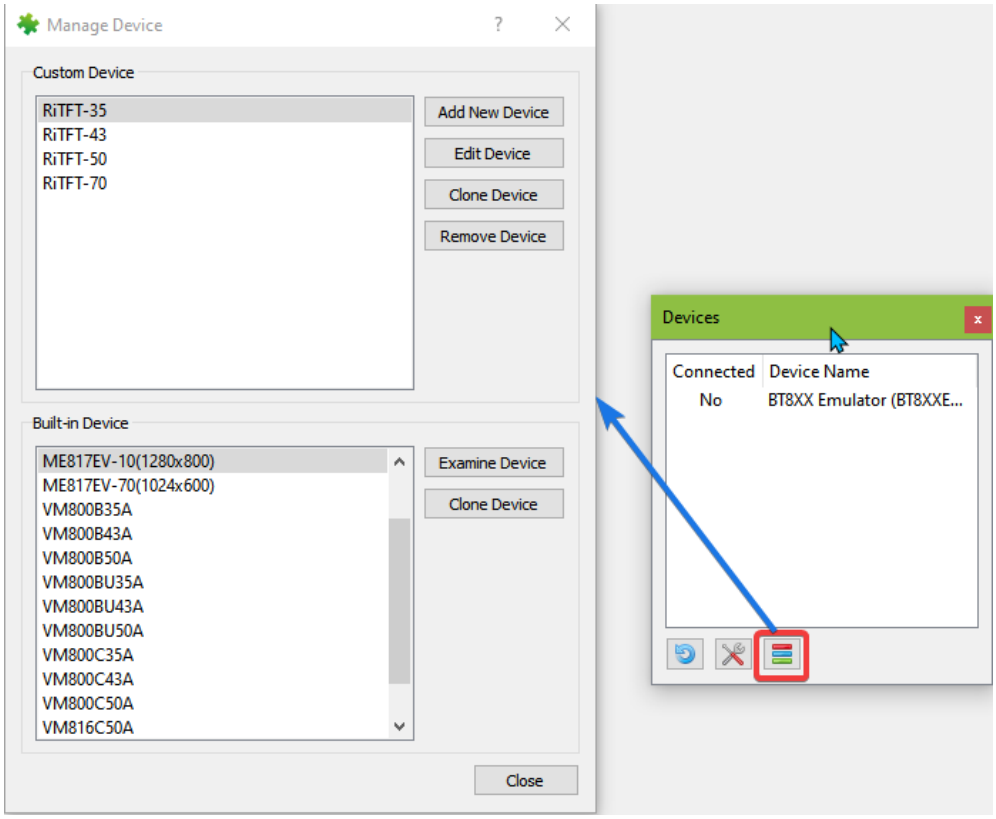


Figure 84 - Managing device

B. Check your design

This section explain how to check the design.

1. Step by Step

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

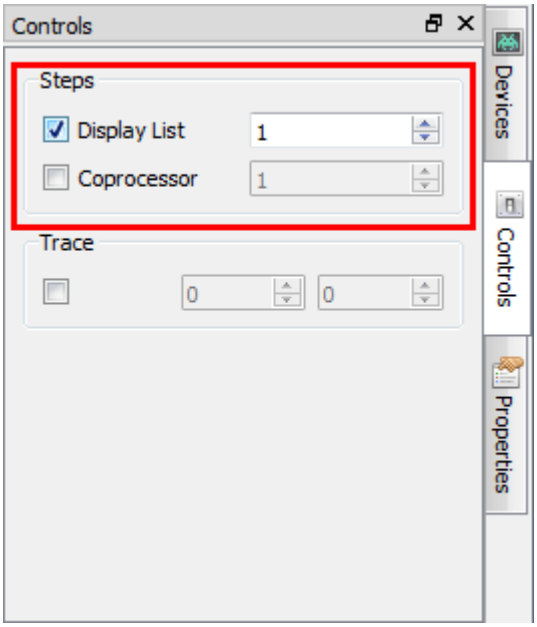


Figure 85 - Select Display List/Coprocessor

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

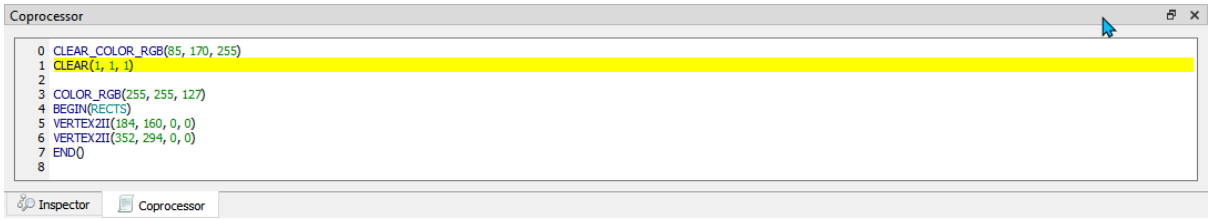


Figure 86 - Display List command is highlighted yellow

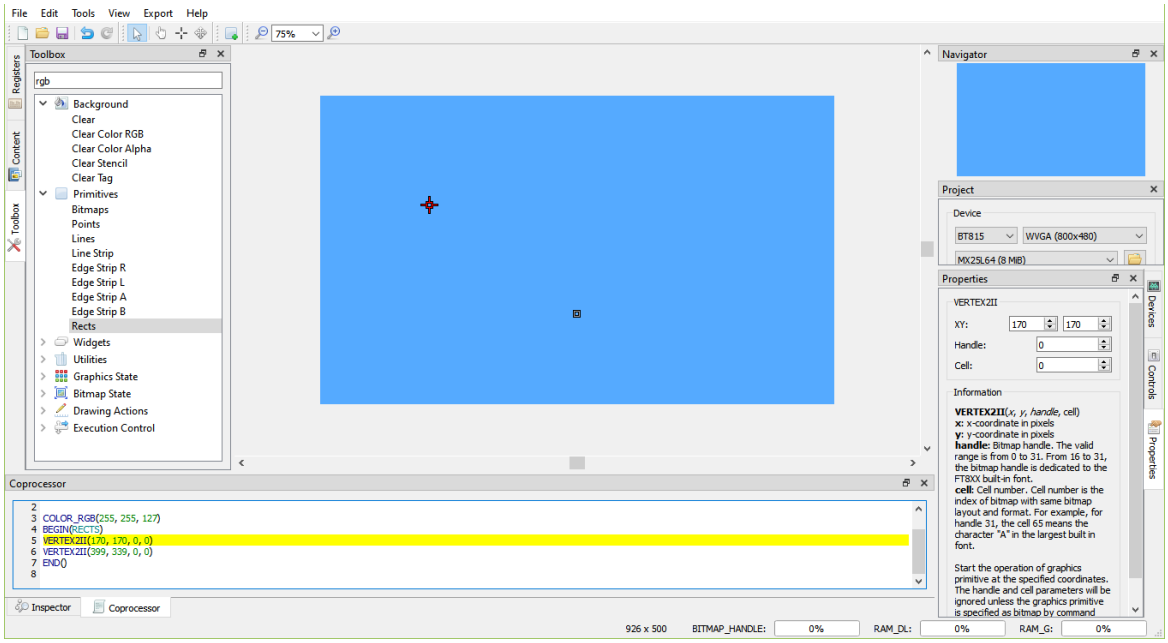


Figure 87 - Prepare to draw rectangle

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

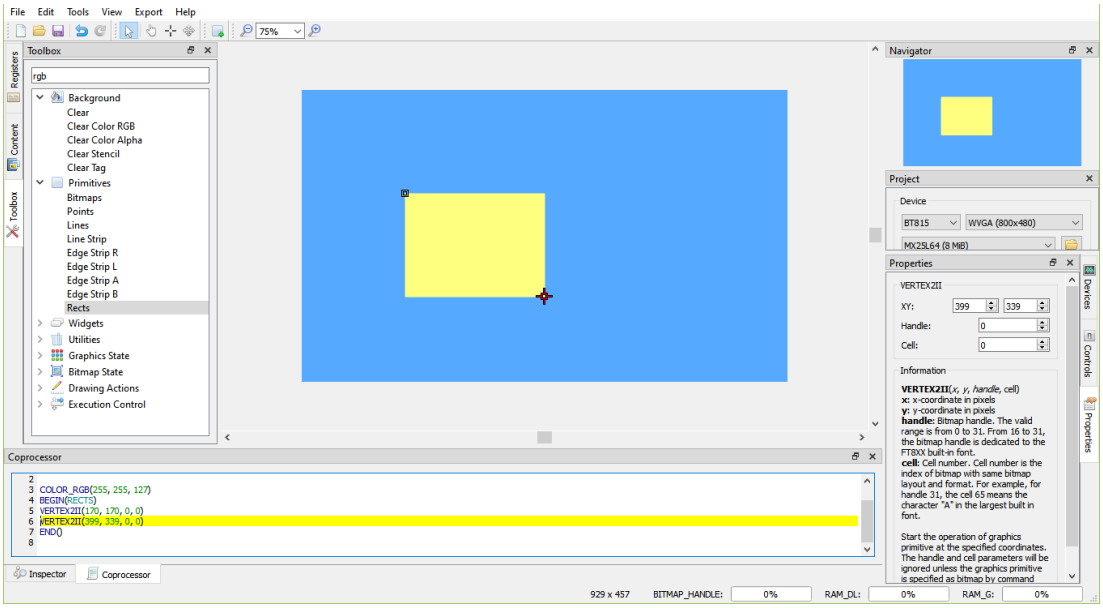


Figure 88 – Draw rectangle

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 object(s) occupy the tracing coordinate, then the respective command(s) in the

commands output are highlighted in **green**. If there is more than 1 object occupying the trace coordinate, the top most object will get highlighted in a **brighter green** than those under it.

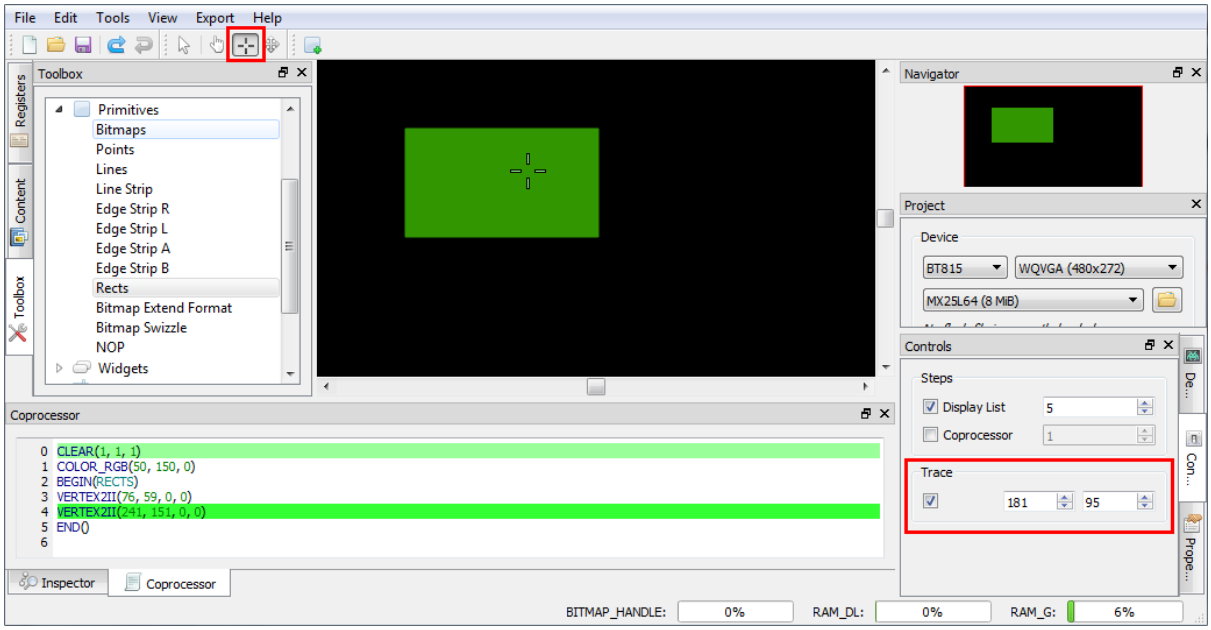


Figure 89 - Trace the pixel

3. Drag and Drop

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

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**.

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

Opened "allwidget_NoScreenSaver" project.

Figure 90 - Open example project

VIII. Disclaimer

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

Disclaimer

Neither the whole nor any part of the information contained in, or the product described in this manual, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder.

This product and its documentation are supplied on an as-is basis and no warranty as to their suitability for any particular purpose is either made or implied. Bridgetek Pte Ltd will not accept any claim for damages howsoever arising as a result of use or failure of this product.

Your statutory rights are not affected. This product or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury. This document provides preliminary information that may be subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document.

Bridgetek Pte Ltd
178 Paya Lebar Road, #07-03
Singapore 409030

IX. Contact Information

Head Quarters – Singapore

Bridgetek Pte Ltd
178 Paya Lebar Road, #07-03
Singapore 409030
Tel: +65 6547 4827
Fax: +65 6841 6071

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

Branch Office – Taipei, Taiwan

Bridgetek Pte Ltd, Taiwan Branch
2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District
Taipei 114
Taiwan, R.O.C.
Tel: +886 (2) 8797 5691
Fax: +886 (2) 8751 9737

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

Branch Office - Glasgow, United Kingdom

Bridgetek Pte. Ltd.
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

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

Branch Office – Vietnam

Bridgetek VietNam Company Limited
Lutaco Tower Building, 5th Floor, 173A Nguyen Van Troi,
Ward 11, Phu Nhuan District,
Ho Chi Minh City, Vietnam
Tel : 08 38453222
Fax : 08 38455222

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

Web Site

http://brtchip.com/

Distributor and Sales Representatives

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

Appendix A - References

Document References

https://brtchip.com/eve-toolchains/

FT81x Series Programmers Guide

FT81x Datasheet

FT800 Series Programmers Guide

FT800 Embedded Video Engine Datasheet

BT81x Datasheet

BT81X Series Programmer Guide

Acronyms & Abbreviations

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

Appendix B – List of Figures & Tables

List of Figures

List of Tables

Appendix C – Revision History

Document Title : BRT_AN_037 EVE Screen Editor 4.0 User Guide

Document Reference No. : BRT_000231

Clearance No. : BRT#160

Product Page : https://brtchip.com/eve-screen-editor/

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