



EVE Screen Editor 3.4

Document Version: 0.7

Date: 08/07/2020

Neither the whole nor any part of the information contained in, or the product described in this manual, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. This product and its documentation are supplied on an as-is basis and no warranty as to their suitability for any particular purpose is either made or implied. Bridgetek Pte Ltd will not accept any claim for damages howsoever arising as a result of use or failure of this product. Your statutory rights are not affected. This product or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury. This document provides preliminary information that may be subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Bridgetek Pte Ltd, 178, Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number 201542387H. © Bridgetek Pte Ltd.

Contents

I	Preface.....	7
A.	Purpose.....	7
B.	Intended Audience	7
C.	Related Documents	7
D.	Feedback.....	7
II	Overview	8
A.	Introduction	8
B.	Key Features	8
C.	ESE 3.X Features	8
D.	Known Issues & Limitations	9
E.	Terms & Description	10
F.	Credits	10
1.	Open Source Software	10
2.	Icons Copyright	10
III	Setup & Installation	11
A.	System Requirements	11
B.	Hardware Requirements	12
C.	Dependencies / Pre-Requisites	12
D.	Installing ESE 3.3	12
E.	Installation Folder	15
IV	The Graphical User Interface	16
A.	Overview	16
B.	Menu, Toolbar and Status Bar	16
1.	Menu	16
2.	Toolbar	22
3.	Status Bar	23
C.	Editors and Inspector	24
1.	Coprocessor Command Editor	24
2.	Display List Editor	25
3.	Inspector	26

D. Toolbox, Content Manager and Registers	28
1. Toolbox	28
2. Registers.....	38
3. Content Manager	39
E. Devices, Controls and Properties	43
1. Device Manager	43
2. Controls.....	45
3. Properties.....	46
F. View Port	47
G. Navigator	47
H. Project settings	48
I. Keyboard Shortcuts	49
V Quick Start Tutorials	50
A. Change the color	50
B. Import the content.....	51
C. Import the flash	54
D. Open the project	55
E. Save your design	56
F. Export the project	56
G. Custom Fonts	58
H. Constrain either horizontal or vertical positioning when dragging an object.....	60
1. Constrain vertical	60
2. Constrain horizontal.....	61
VI Command Usage Examples	62
A. CMD_PLAYVIDEO	62
B. CMD_LOADIMAGE	63
C. CMD_SETBITMAP	64
D. CMD_SNAPSHOT	65
E. CMD_SKETCH	65
F. CMD_SNAPSHOT2	66

G. VERTEX_TRANSLATE_X/Y	67
H. CMD_MEDIAFIFO	68
I. CMD_SETBASE	69
J. CMD_ROMFONT	69
K. PALETTE_SOURCE.....	70
L. CMD_SETFONT2	71
M. CMD_TEXT	71
VII Working With EVE Screen Editor	72
A. Connect with Hardware.....	72
B. Export your project	75
C. Check your design	75
1. Step by Step	75
2. Trace the pixel.....	77
3. Drag and Drop.....	78
D. Example Project	78
VIII Disclaimer	79

Figure 1 - ESE 3.3 User Interface Components	16
Figure 2 - File Menu	17
Figure 3 - Edit Menu	18
Figure 4 - Tool Menu	18
Figure 5 - View Menu.....	20
Figure 6 - Export Menu	20
Figure 7 - Help Menu	22
Figure 8 - Toolbar	22
Figure 9 - Status Bar	23
Figure 10 - Inspector	24
Figure 11 - Coprocessor Command Editor	24
Figure 12 - Display List Editor.....	25
Figure 13 - Inspector.....	26
Figure 14 - RAM_DL	26
Figure 15 - Select rows in RAM_DL.....	26
Figure 16 - Paste selected rows in RAM_DL	27
Figure 17 - RAM_REG	27
Figure 18 - Select rows in RAM_REG.....	27
Figure 19 - Paste selected rows in RAM_REG	27
Figure 20 - Toolbox	28
Figure 21 - Display List mode	28
Figure 22 - Toolbox in Display List mode.....	29
Figure 23 - Background in Display List mode	29
Figure 24 - Primitives in Display List mode	30
Figure 25 - Graphics State in Display List mode	30
Figure 26 - Bitmap State in Display List mode	31
Figure 27 - Drawing Actions in Display List mode.....	31
Figure 28 - Execution Control in Display List mode.....	32
Figure 29 - Coprocessor mode	32
Figure 30 - Toolbox in Coprocessor mode.....	33
Figure 31 - Background in Coprocessor mode	33
Figure 32 - Primitives in Coprocessor mode	33
Figure 33 - Utilities in Coprocessor mode	34
Figure 34 - Graphics State in Coprocessor mode	35
Figure 35 - Bitmap State in Coprocessor mode	36
Figure 36 - Drawing Actions in Coprocessor mode.....	37
Figure 37 - Execution Control in Coprocessor mode.....	37
Figure 38 - Register.....	38
Figure 39 - Customize resolution in Register window	38
Figure 40 - Content Manager.....	39
Figure 41 - Load content dialog	40
Figure 42 - Content loaded.....	40
Figure 43 - Content properties.....	41
Figure 44 - Raw converter.....	41
Figure 45 - Drag content into Viewport	42
Figure 46 - Remove content	43
Figure 47 - Before connect.....	43
Figure 48 - Connected device	44
Figure 49 - Device type	44
Figure 50 - Controls tab.....	45
Figure 51 - Properties tab	46

Figure 52 – View port	47
Figure 53 – Navigator	47
Figure 54 - Project settings	48
Figure 55 - Flash supported.....	48
Figure 56 - Load flash file.....	48
Figure 57 - Select flash size	49
Figure 58 - Flash path	49
Figure 59 - Change the color	50
Figure 60 - Select color.....	51
Figure 61 - Import content.....	52
Figure 62 - Select converter image.....	52
Figure 63 – Drag & drop image.....	53
Figure 64 - Import flash.....	54
Figure 65 - Flash address.....	55
Figure 66 - Open project.....	55
Figure 67 - Save the project.....	56
Figure 68 - Export project	57
Figure 69 - Arduino IDE	58
Figure 70 - Custom font.....	59
Figure 71 - Property "Text" of custom font	59
Figure 72 - Press SHIFT for constrain vertical	60
Figure 73 - Slide up/down to set y-coordinate	60
Figure 74 - Press ALT for constrain horizontal	61
Figure 75 - Slide left/right to set x-coordinate	61
Figure 76 - Device connected	73
Figure 77 - Select correct device type.....	74
Figure 78 - Managing device.....	74
Figure 79 - Select Display List/Coprocessor	75
Figure 80 - Display List command is highlighted yellow	76
Figure 81 - Trace the pixel	78
Figure 82 - Open example project	79

I Preface

A. Purpose

This document describes the functionality and procedures involved in using the application **EVE Screen Editor (ESE)**.

B. Intended Audience

The intended audience shall be any 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
BT815/6 Advanced Embedded Video Engine	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 Screen Editor (ESE) is a GUI tool that provides an intuitive "drag & drop" user experience to construct screen design without programming. Empowered by cutting edge EVE emulator, ESE gives users the maximum fidelity of graphics effect.

Coprocessor commands and display list can also be input in the editor window of the tool 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 an 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.

Just go ahead!

B. 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

C. ESE 3.X Features

What's new in ESE 3.4?

- ✚ Add system clock configuration in the device manager
- ✚ Integrate latest Eve Emulator v4.0.1 to accelerate ASTC decoding
- ✚ Migrate ESE to 64 bit Windows release
- ✚ Fix bug: The assets under resource folder of ESE project are deleted after another project is opened

D. Known Issues & Limitations

The following are 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 hang.

E. Terms & Description

Abbreviations/Term	Description
DLL	Dynamic Link Library is a collection of small programs, any of which can be called when needed by a larger program that is running in the computer
ESE	EVE Screen Editor
EVE Emulator	Bridgetek behaviour-modelling software for EVE Series chip
HAL	Hardware Abstraction Layer is a software subsystem providing hardware abstraction.
IDE	Integrated Development Environment
Simulation	Preview the project or page by running the generated C code on PC
Widget	One type of logic node which has visual appearance rendered by EVE

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.
- SDL2: <https://www.libsdl.org> under zlib license.
- FreeType: <https://www.freetype.org> under GPL license.

2. Icons Copyright

Some of the icons used in ESE 3.X 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 3.3 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
- ✚ Display resolution: At least 1080 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 (MPSSE) projects. VM800B (3.5", 4.3" or 5.0" display) with MPSSE cable (USB to SPI cable) are required to run the project.
- ✚ 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:

- ✚ VM800B (3.5", 4.3" or 5.0" display) with MPSSE cable (USB to SPI cable). This is selected in the device manager of the screen editor tool.
- ✚ VM816C50A with LIBFT4222 or MPSSE cable

C. Dependencies / Pre-Requisites

- **Visual C++ Redistributable for Visual Studio 2015**

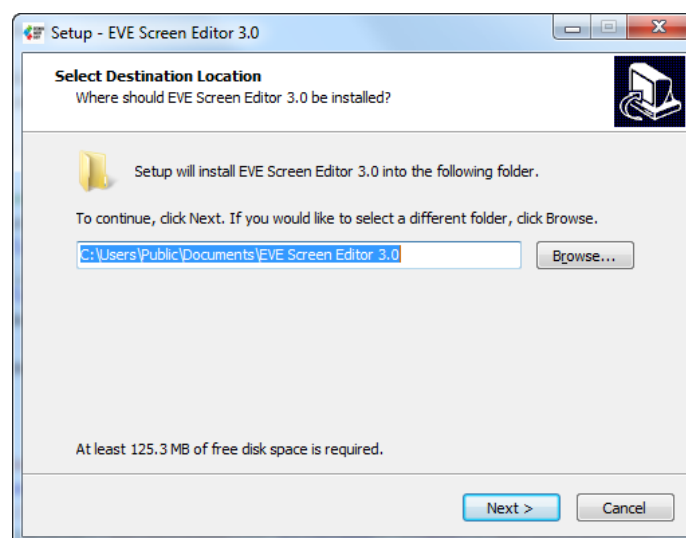
If the PC does not have Microsoft Visual Studio 2015 installed, Visual C++ Redistributable is required. Users can download this from:

<https://www.microsoft.com/en-sg/download/details.aspx?id=48145>

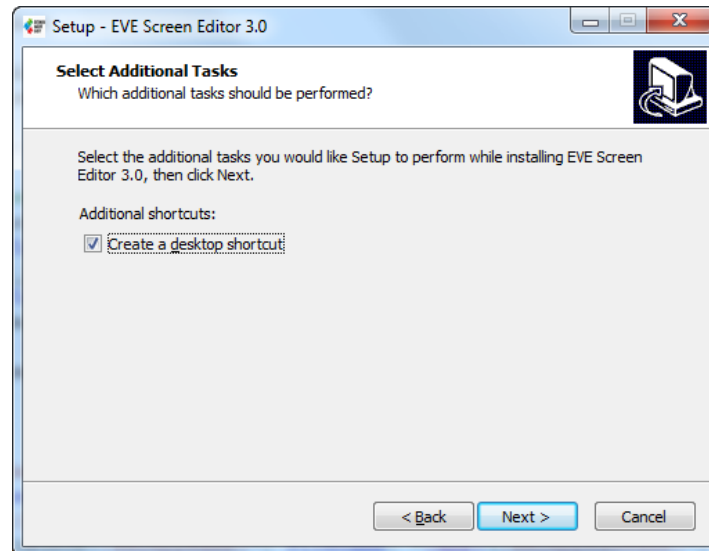
D. Installing ESE 3.3

The following steps will guide you through the ESE 3.3 *Setup/Installation* process.

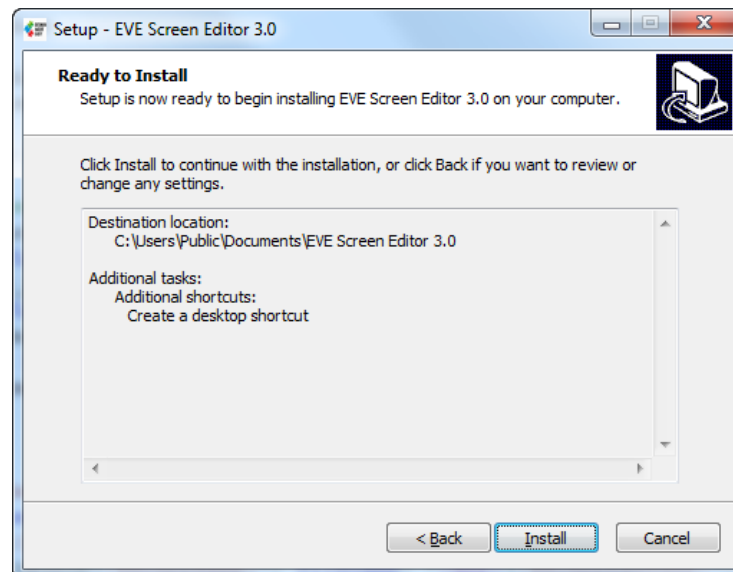
- Download the package from www.brtchip.com.
- When prompted with a download dialog box. Click on **Save**.
- Navigate to the folder under which the package files are downloaded.
- Extract the zip file contents. Double click on the executable file – **EVE Screen Editor 3.3.exe**
- Select a "Destination 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.



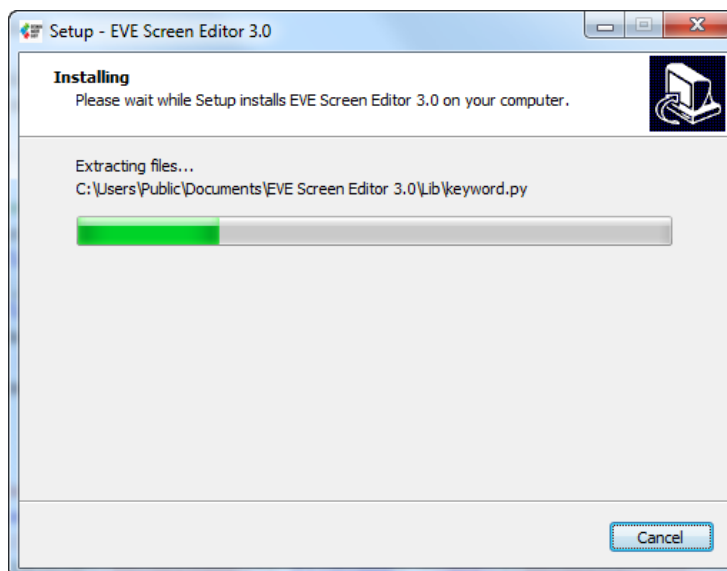
- vi. In the **Select Additional Tasks** window, check **"Create a desktop"** boxes, to have the ESE 3.3 icon displayed on the desktop if required. Click **Next** to prepare for the installation.



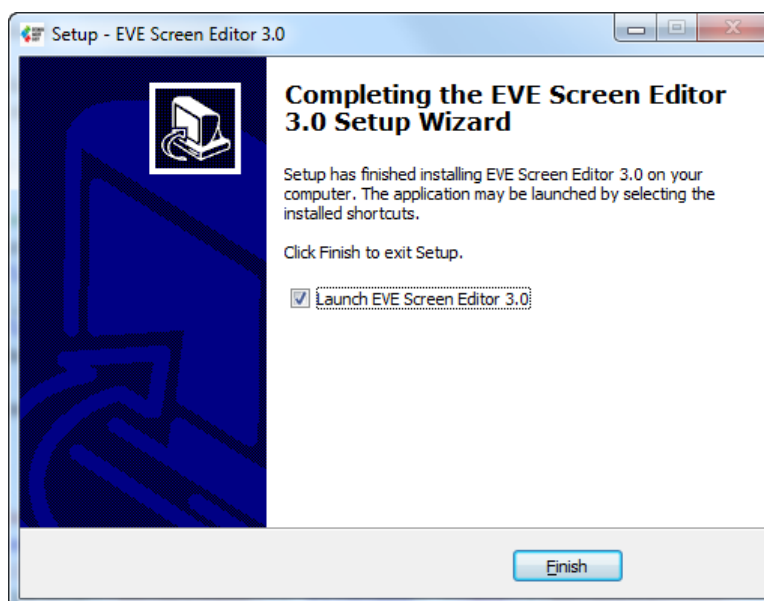
- vii. The initial setup is completed and the application is ready to be installed.



- 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 3.3 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 3.X.

Folder Name	Description	Permission
Examples	The example projects created by ESE	Read/Write
imageformats	Qt run-time DLLs for image format supporting	Read-Only
Libraries	Widget library, application framework and Hardware abstraction layer	Read-Only
Manual	Contains this document	Read-Only
platforms	Qt platform run-time DLLs.	Read-Only
EVE_Hal_Library	The template project for BT81X used by ESE	Read-Only
untitled	The template project used by ESE	Read-Only
device_sync	Contain information of custom boards	Read/Write
device_sync/buildin	Contain information of build-in boards	Read-Only

Table 1 - Installation Folder

IV The Graphical User Interface

A. Overview

ESE 3.X user interface has the following components:

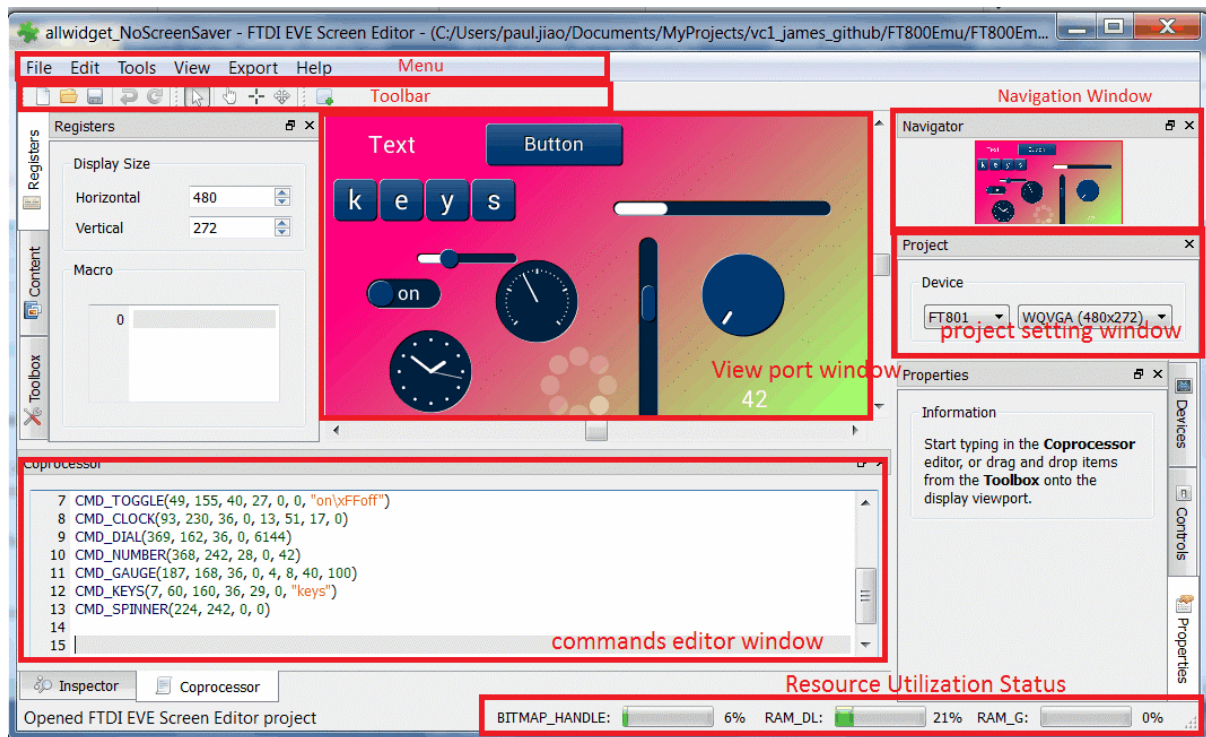


Figure 1 - ESE 3.3 User Interface Components

B. Menu, Toolbar and Status Bar

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

1. Menu

File Menu

The file menu consists of menu options as image below.

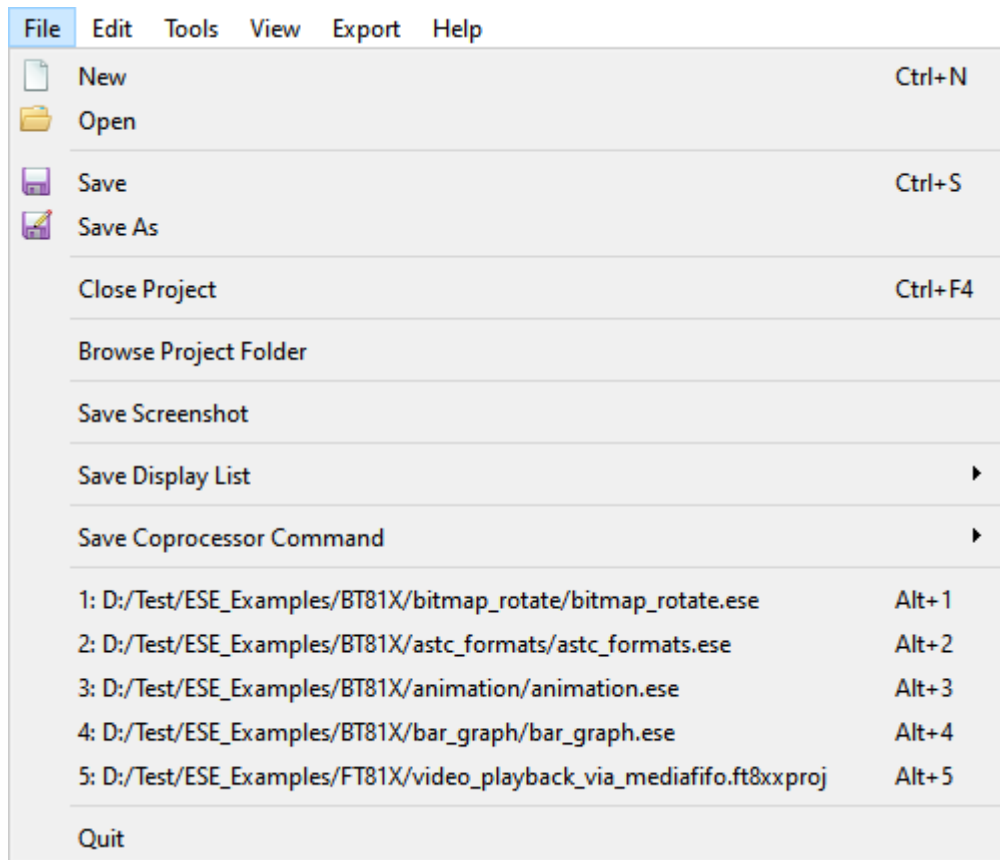


Figure 2 - File Menu

New

Creates a new project, clears the screen.

Open

Displays the open dialog box which retrieves the existing project. The file extension is in ".ese", ".ft8xxproj" or ".ft800proj". Example projects can be viewed here.

Save

Saves the screen design in the user specified location. The file extension is in ".ese" format.

Save As

Choose a different destination and file name to save the current project. The file extension is in ".ese" format.

Browse Project Folder

Open the project folder where current project file locates in one click.

Save Screenshot

Save the current screen of viewport window into local PC.

Save Display List

Save current display list to a text file, in Little Endian or Big Endian format

Save Coprocessor Command

Save current coprocessor command to a text file, in Little Endian or Big Endian format

Recent Projects History

The latest 5 opened projects are added into history list. Click on history item will reopen the related project again.

Quit

This closes the application.

Edit Menu

The edit menu allows users to revert or redo certain actions.

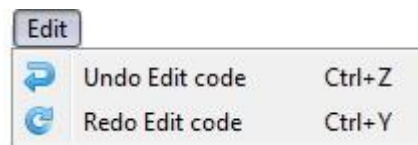


Figure 3 - Edit Menu

Undo

Undo the last action done in the editor.

Redo

Redo the last action done in the editor.

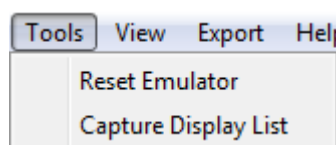
Tools menu

Figure 4 - Tool Menu

Reset Emulator

Reset the background running emulator.

Capture Display List

Extend the EVE coprocessor commands to display list and show it in display list editor window.

For example, to see what display list is generated for coprocessor command "CMD_BUTTON", users can type "CMD_BUTTON" in coprocessor editor and click the menu here. The display list commands will be shown in the display list editor as below:

```
SAVE_CONTEXT()
VERTEX_FORMAT(2)
BITMAP_HANDLE(15)
CELL(0)
BITMAP_SOURCE(-2097076)
BITMAP_LAYOUT_H(0, 0)
BITMAP_LAYOUT(L8, 1, 25)
BITMAP_SIZE_H(0, 0)
BITMAP_SIZE(NEAREST, REPEAT, BORDER, 120, 36)
COLOR_MASK(0, 0, 0, 1)
BLEND_FUNC(ZERO, ZERO)
BEGIN(BITMAPS)
VERTEX2F(192, 484)
COLOR_MASK(1, 1, 1, 1)
BLEND_FUNC(SRC_ALPHA, ONE_MINUS_SRC_ALPHA)
LINE_WIDTH(60)
BEGIN(RECTS)
COLOR_RGB(255, 255, 255)
VERTEX2F(205, 497)
VERTEX2F(655, 611)
COLOR_RGB(0, 0, 0)
VERTEX2F(211, 503)
VERTEX2F(661, 617)
COLOR_RGB(0, 56, 112)
VERTEX2F(207, 499)
VERTEX2F(657, 613)
BEGIN(BITMAPS)
COLOR_MASK(0, 0, 0, 1)
BLEND_FUNC(DST_ALPHA, ZERO)
VERTEX2F(192, 484)
COLOR_MASK(1, 1, 1, 0)
BLEND_FUNC(DST_ALPHA, ONE_MINUS_DST_ALPHA)
COLOR_RGB(255, 255, 255)
VERTEX2F(192, 484)
COLOR_MASK(1, 1, 1, 1)
BLEND_FUNC(SRC_ALPHA, ONE_MINUS_SRC_ALPHA)
COLOR_RGB(0, 0, 0)
BITMAP_HANDLE(27)
VERTEX2II(82, 128, 27, 'B')
VERTEX2II(92, 128, 27, 'u')
VERTEX2II(101, 128, 27, 't')
VERTEX2II(108, 128, 27, 't')
VERTEX2II(115, 128, 27, 'o')
VERTEX2II(125, 128, 27, 'n')
RESTORE_CONTEXT()
SAVE_CONTEXT()
VERTEX_FORMAT(2)
BITMAP_HANDLE(27)
VERTEX2II(83, 129, 27, 'B')
VERTEX2II(93, 129, 27, 'u')
VERTEX2II(102, 129, 27, 't')
VERTEX2II(109, 129, 27, 't')
VERTEX2II(116, 129, 27, 'o')
```

```

VERTEX2II(126, 129, 27, 'n')
RESTORE_CONTEXT()
DISPLAY()
  
```

View Menu

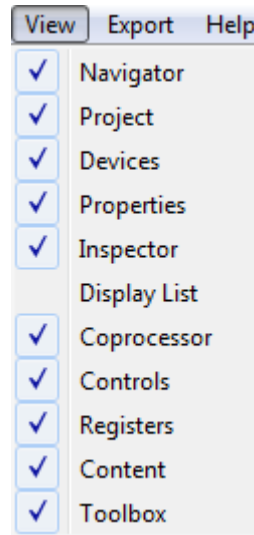


Figure 5 - View Menu

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

Export Menu

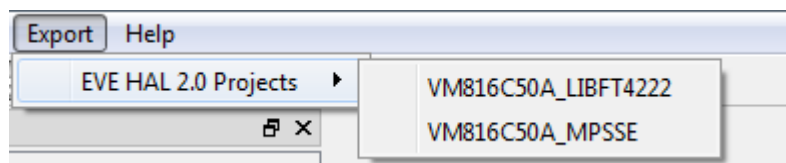


Figure 6 - Export Menu

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

For FT80X based project, there are scripts to export it to Gameduino2, EVE Arduino, and HAL (FTDI) based project.

For FT81X based project, there are scripts to export it to HAL 2.0 (FTDI) based project.

For BT81X based project, there are scripts to export it to HAL 2.0 (FTDI) based project.

Assuming the commands in the editor are

```
BEGIN(RECTS)
VERTEX2II(34, 75, 0, 0)
COLOR_RGB(255, 85, 0)
VERTEX2II(112, 138, 0, 0)
END()
BEGIN(POINTS)
POINT_SIZE(320)
COLOR_RGB(85, 170, 127)
VERTEX2II(238, 105, 0, 0)
END()
CMD_KEYS(199, 196, 160, 36, 29, 0, "keypad")
```

The equivalent exported gameduino2 project is

```
#include <EEPROM.h>
#include <SPI.h>
#include <GD2.h>

void setup()
{
  GD.begin();
}

void loop()
{
  GD.Clear(1, 1, 1);
  GD.Begin(RECTS);
  GD.Vertex2ii(34, 75, 0, 0);
  GD.ColorRGB(255, 85, 0);
  GD.Vertex2ii(112, 138, 0, 0);
  GD.End();
  GD.Begin(POINTS);
  GD.PointSize(320);
  GD.ColorRGB(85, 170, 127);
  GD.Vertex2ii(238, 105, 0, 0);
  GD.End();
  GD.cmd_keys(199, 196, 160, 36, 29, 0, "keypad");

  GD.swap();
}

/* end of file */
```

If the user has the Arduino IDE installed and associated with the Arduino project files, the Arduino IDE will be invoked with this generated project. In order to run the generated project, the respective Arduino library has to be installed. HAL (FTDI) project script opens up a file browser window and the ReadMe.txt in the project sub folder details the file structure of the projects.

Help Menu

This gives information about the software and help and manual.

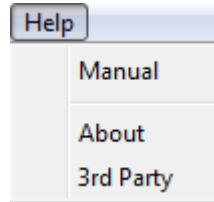


Figure 7 - Help Menu

Manual

This document is displayed.

About

The about window of the FTDI EVE Screen Editor is displayed.

3rd Party

This gives information about the copyright of third party software or artifacts, including Qt software and Fugue icon.

2. Toolbar

This toolbar defines shortcuts of mouse operation for New, Open, Save, Undo, Redo, Cursor, Touch, Trace, Edit and Insert functionality.

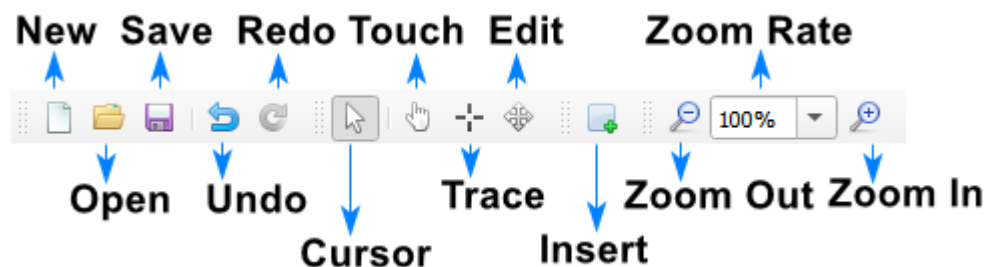


Figure 8 - Toolbar

New

Create a new project. (Clears the editor and starts a new project in a temporary directory.)

Open

Open an existing project.

Save

Save the current project to file

Undo

Revoke the last operation.

Redo

Redo 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

Force trace cursor in viewport. See Trace the pixel for more details.

Edit

Force widget editing cursor in viewport

Insert

Insert duplicates of currently selected widget or primitive at clicked position, overrides any current cursor selection

Zoom Out

Zooming out emulator view port

Zoom In

Zooming in emulator view port

Zoom Rate

Select concrete zooming rate

3. Status Bar

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



Figure 9 - Status Bar

C. Editors and Inspector

The topics in this section provide information about the editors and inspector window, which locates at bottom of main window.

Editors provides individual window to coprocessor commands and display list commands, which are sent to EVE coprocessor RAM_CMD and EVE graphics engine RAM_DL, respectively.

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

Inspector shows the content of RAM_DL and RAM_REG and no editing is allowed. RAM_DL and RAM_REG can be selected line by line then copy them to another text editor.

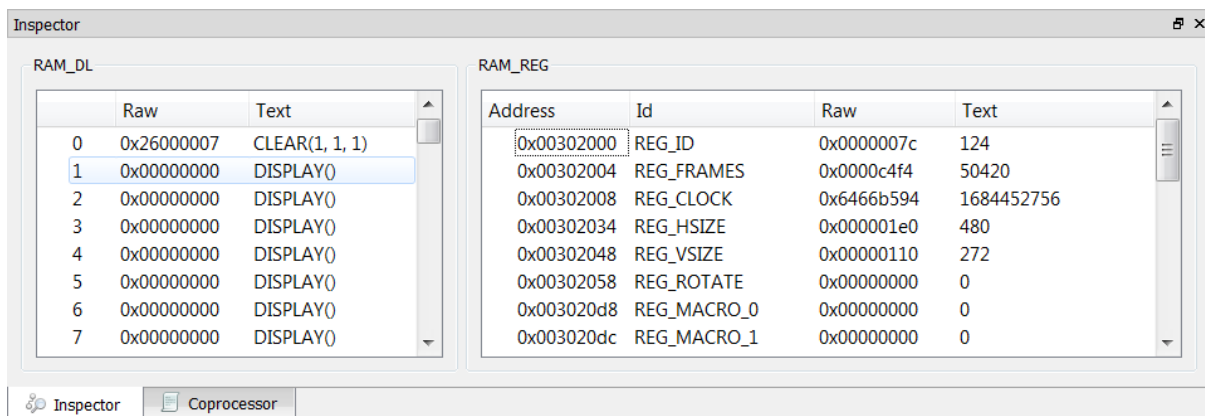


Figure 10 – Inspector

1. Coprocessor Command Editor

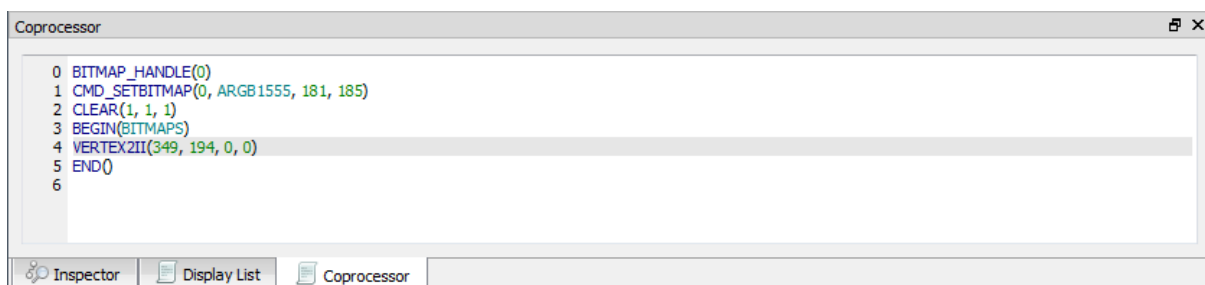


Figure 11 - Coprocessor Command Editor

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 its following commands and shall be the last commands in editor.

2. Display List Editor

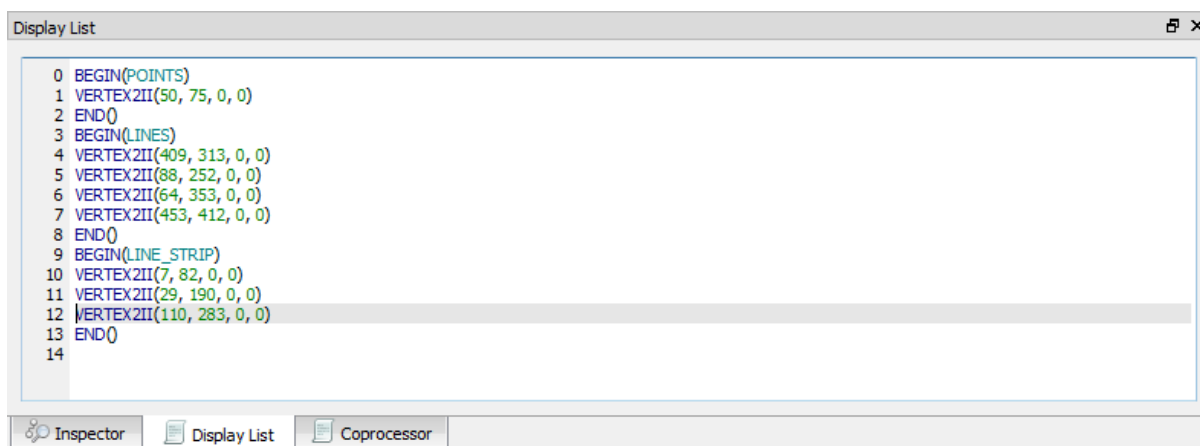


Figure 12 - Display List Editor

The features are as below:

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

Note:

- Coprocessor Command Editor has higher priority and its content overrides the content of display list editor. To validate the input of display list editor, make sure the coprocessor command editor window contains no any commands.
- By default, the Display List editor is hide.

3. Inspector

The topics in this section provide information about the inspector feature of the EVE Screen Editor.

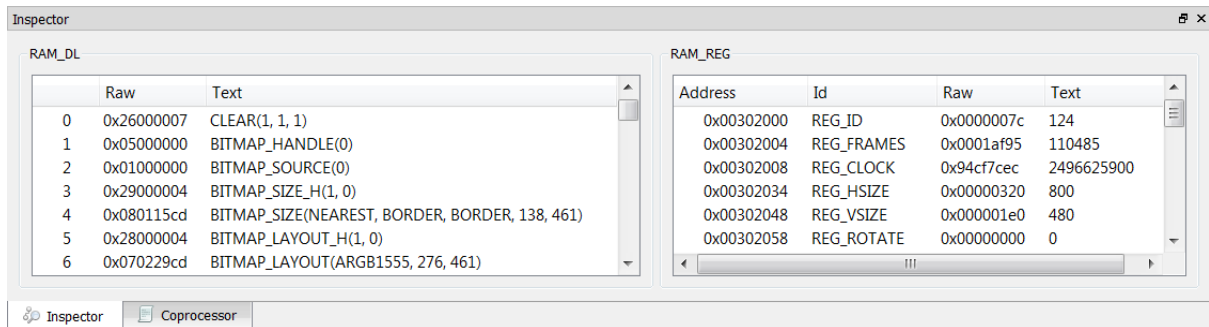


Figure 13 - Inspector

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.

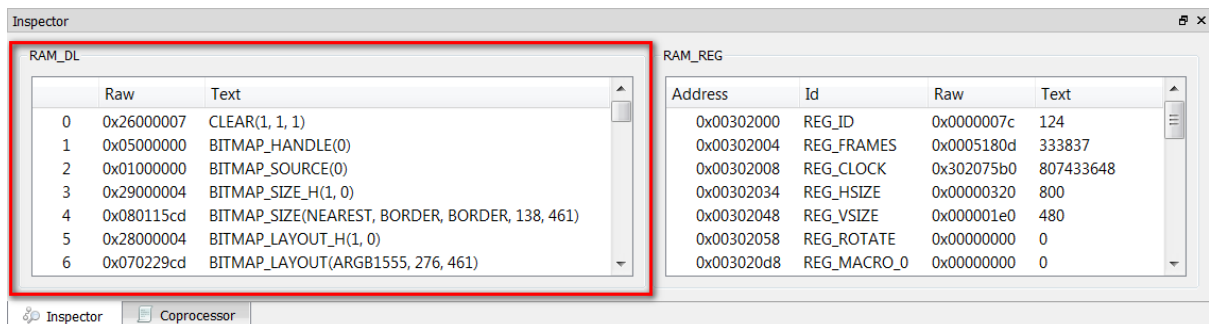


Figure 14 - RAM_DL

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

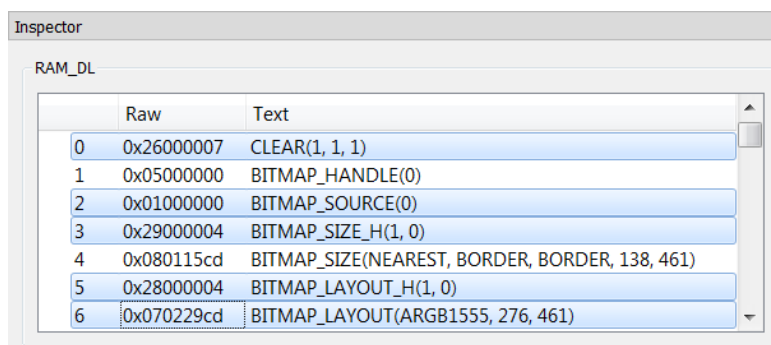


Figure 15 - Select rows in RAM_DL

	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 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 they are read-only.

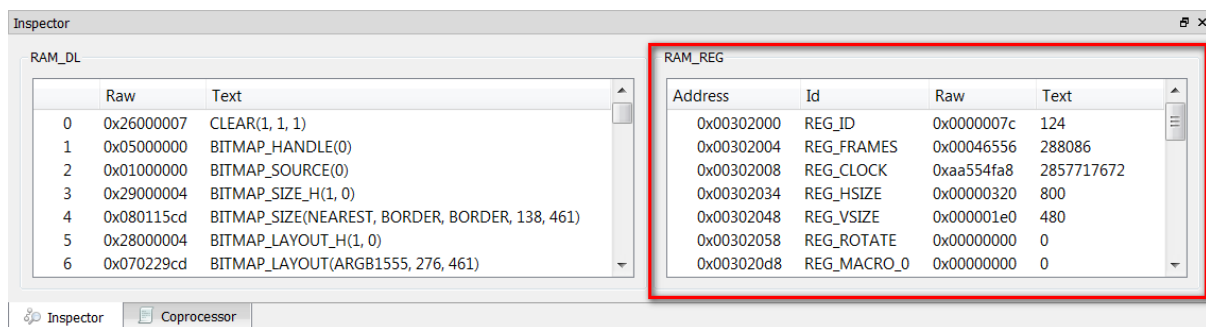


Figure 17 - RAM_REG

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

Address	Id	Raw	Text
0x00302000	REG_ID	0x0000007c	124
0x00302004	REG_FRAMES	0x00048681	296577
0x00302008	REG_CLOCK	0x3f413090	1061236880
0x00302034	REG_HSIZE	0x00000320	800
0x00302048	REG_VSIZE	0x000001e0	480
0x00302058	REG_ROTATE	0x00000000	0
0x003020d8	REG_MACRO_0	0x00000000	0

Figure 18 - Select rows in RAM_REG

Address	Id	Raw	Text
0x00302000	REG_ID	0x0000007c	124
0x00302004	REG_FRAMES	0x00048681	296577
0x00302034	REG_HSIZE	0x00000320	800
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

The topics in this section provide information about the windows at the left side of viewport.

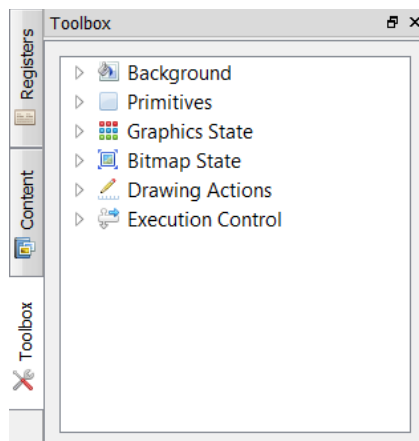


Figure 20 - Toolbox

1. Toolbox

The Toolbox is the portal to access the coprocessor commands or display list commands. When the Display List editor is in focus, the display list commands are available in the Toolbox.

When the Coprocessor editor is in focus, the full set of display list and coprocessor commands are available in the Toolbox.

Users may drag and drop the commands from the Toolbox into the viewport.

Display list mode

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

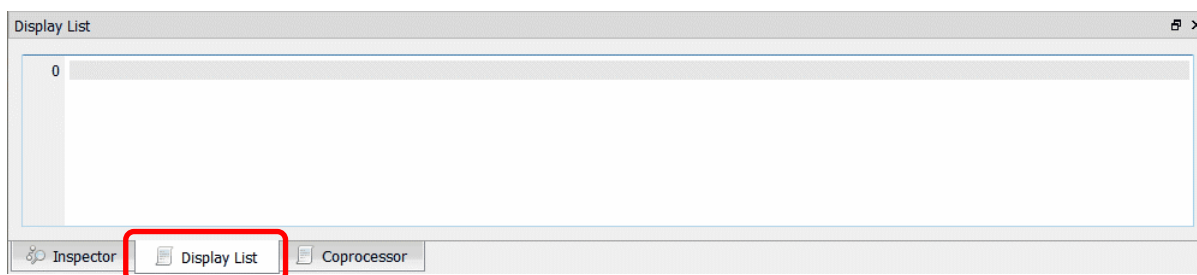


Figure 21 - Display List mode

The toolbox will be enabled as below:

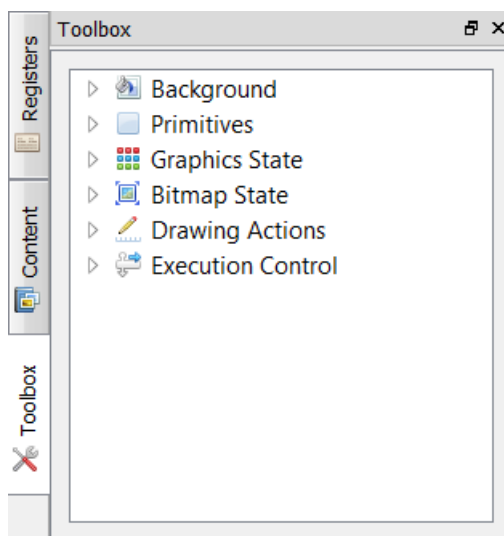


Figure 22 - Toolbox in Display List mode

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

- Background

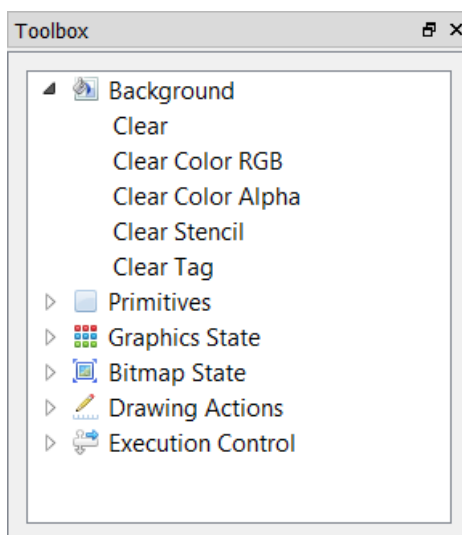


Figure 23 - Background in Display List mode

-

- Primitives

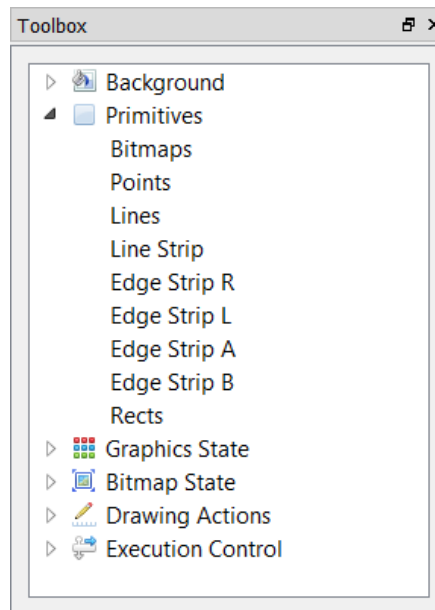


Figure 24 - Primitives in Display List mode

- Graphics State

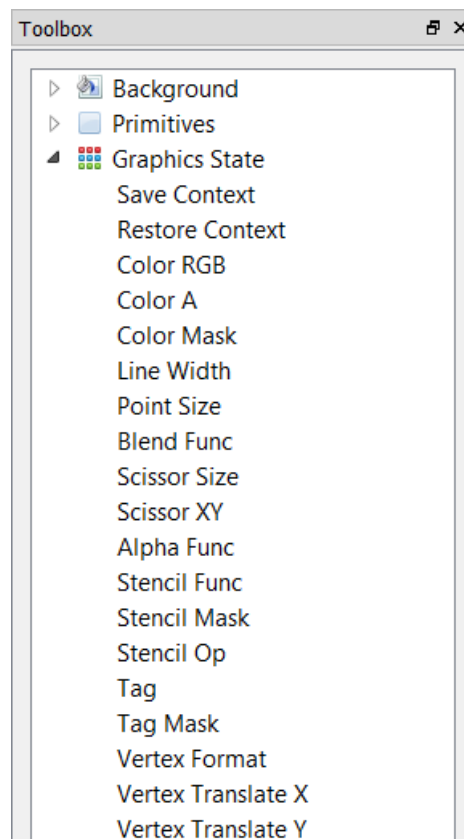


Figure 25 - Graphics State in Display List mode

- Bitmap State

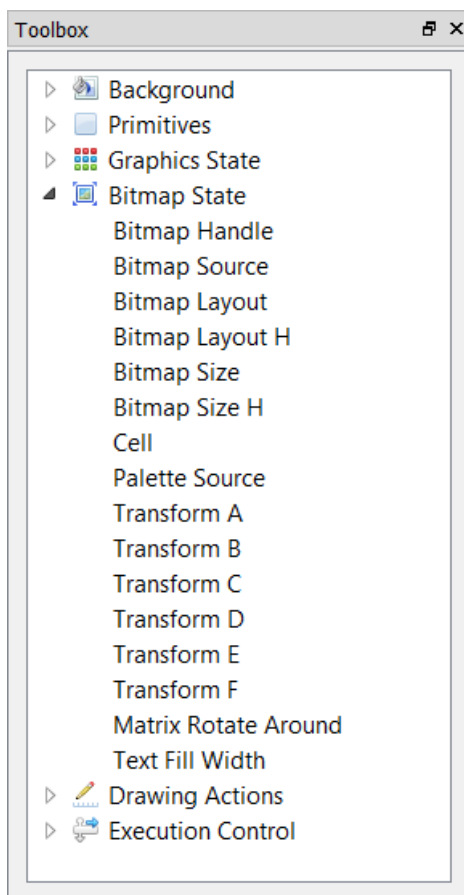


Figure 26 - Bitmap State in Display List mode

- Drawing Actions

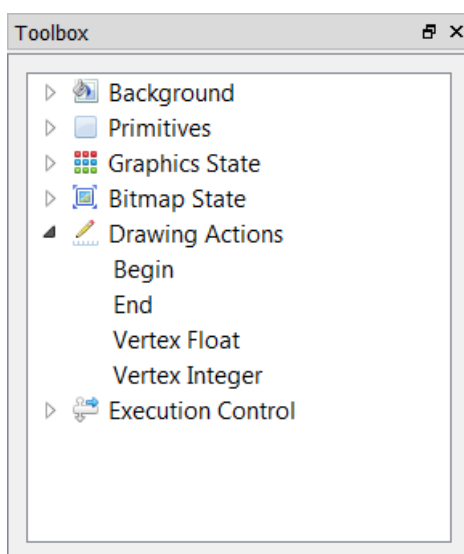


Figure 27 - Drawing Actions in Display List mode

- Execution Control

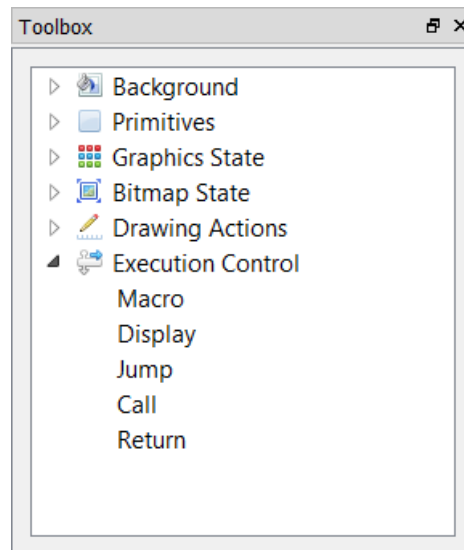


Figure 28 - Execution Control in Display List mode

Coprocessor mode

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

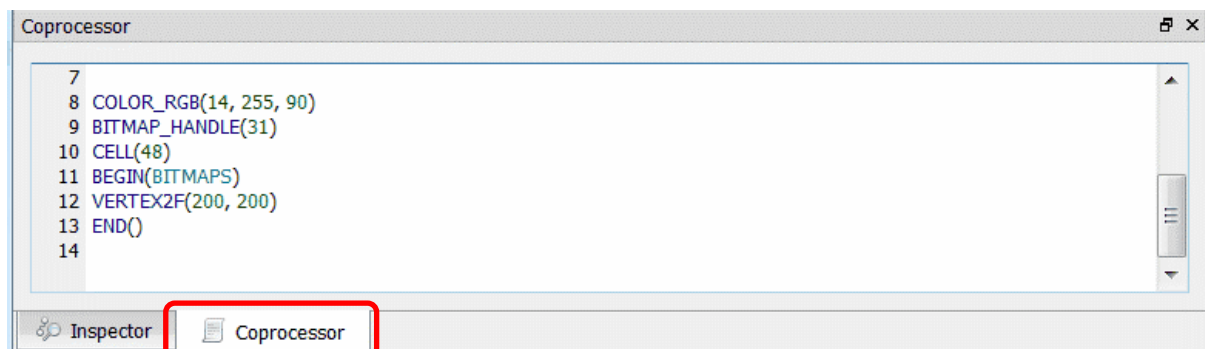


Figure 29 - Coprocessor mode

The toolbox will be enabled as below:

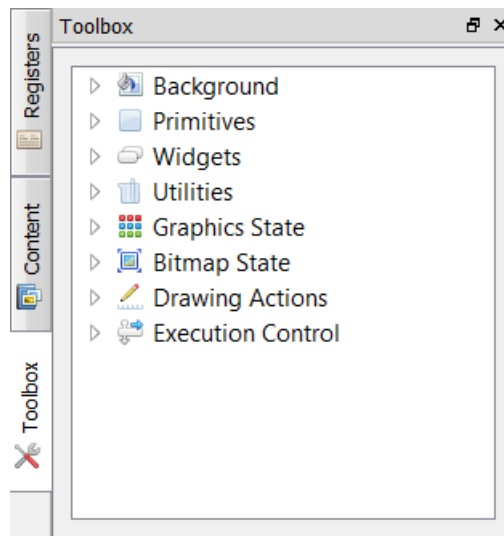


Figure 30 - Toolbox in Coprocessor mode

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

- Background

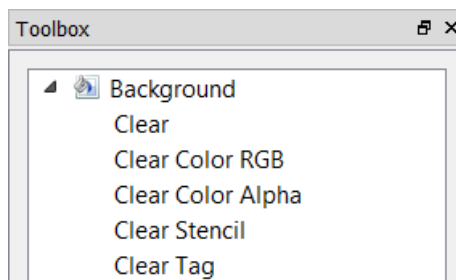


Figure 31 - Background in Coprocessor mode

- Primitives

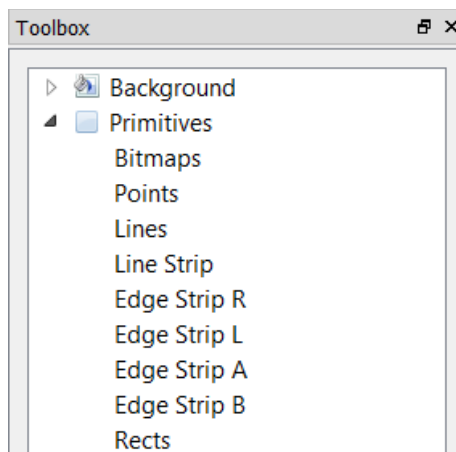
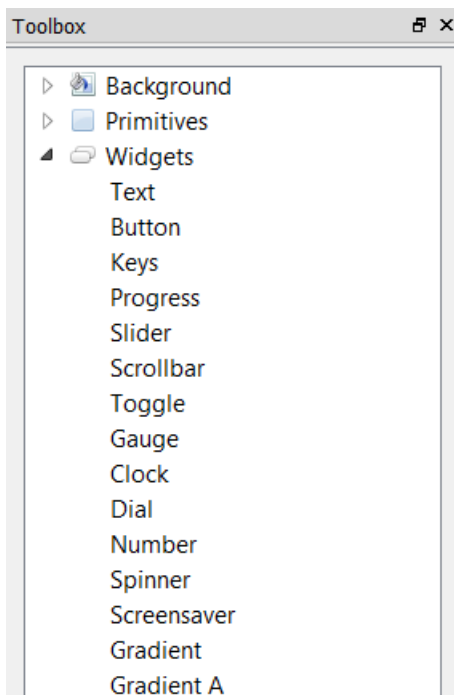


Figure 32 - Primitives in Coprocessor mode

- Widgets



- Utilities

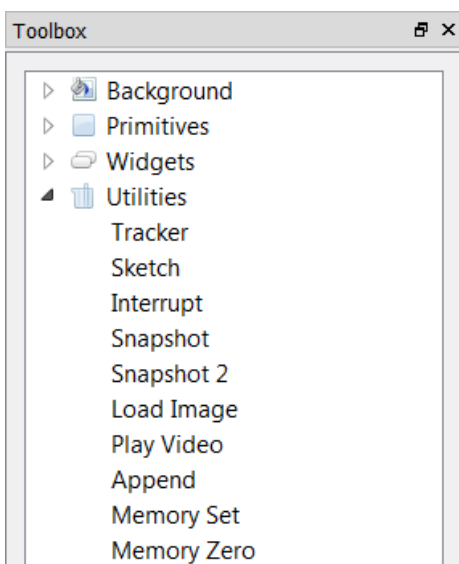


Figure 33 - Utilities in Coprocessor mode

- Graphics State

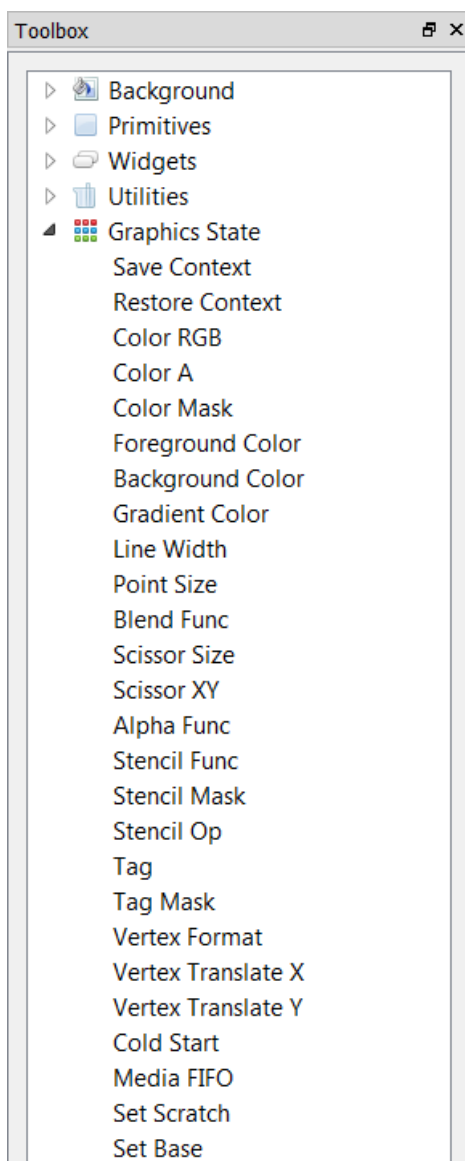


Figure 34 - Graphics State in Coprocessor mode

- Bitmap State

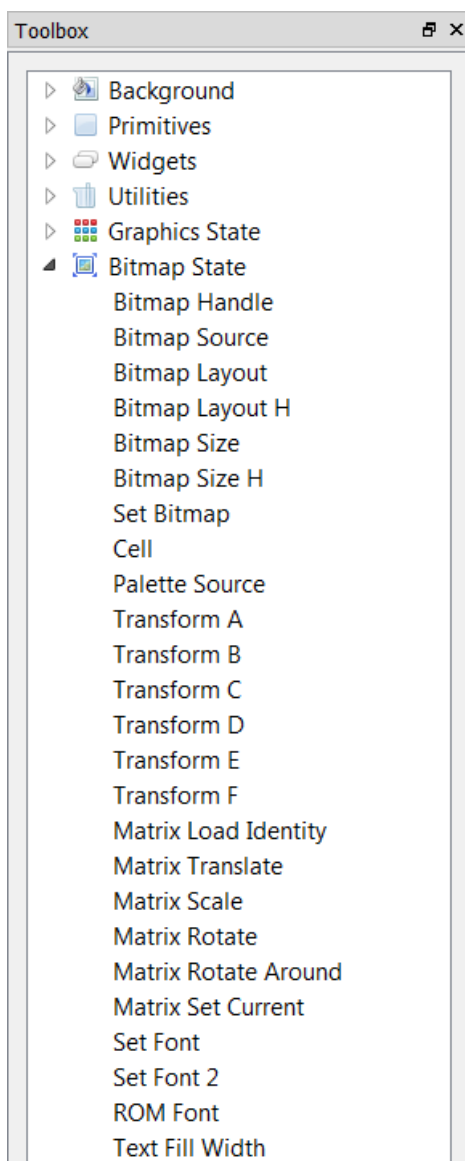


Figure 35 - Bitmap State in Coprocessor mode

- Drawing Actions

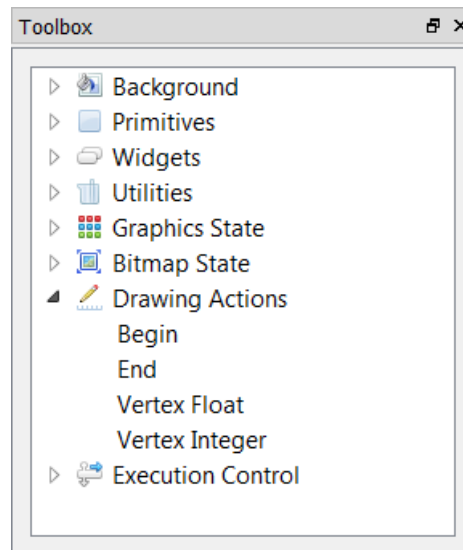


Figure 36 - Drawing Actions in Coprocessor mode

- Execution Control

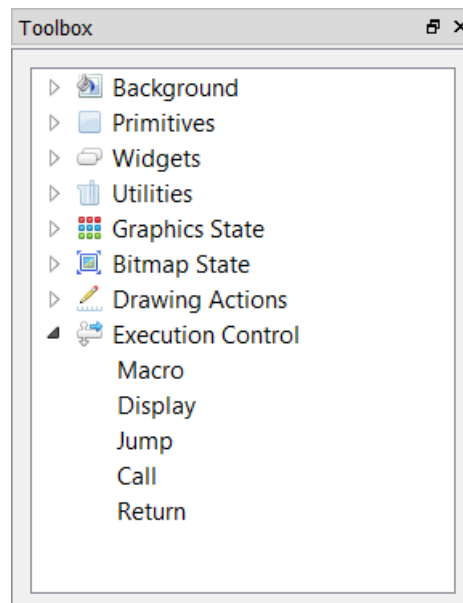


Figure 37 - Execution Control in Coprocessor mode

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 in 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 immediately.

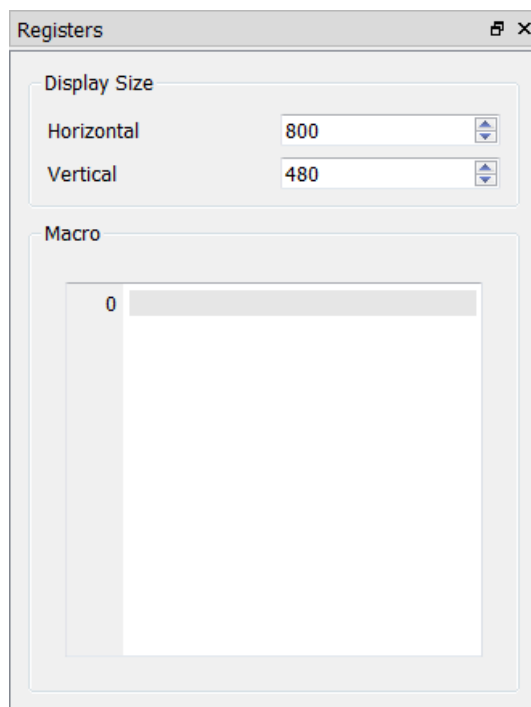


Figure 38 – Register

ESE is able to simulate the custom resolution up to 2048 by 2048, which can be done by register window:

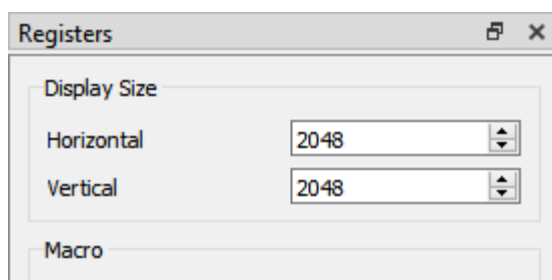


Figure 39 - Customize resolution in Register window

However, users shall notice that is for simulation purpose only, not for physical hardware platform.

3. Content Manager

The topics in this section provide information about the content manager of the ESE. 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 scene.

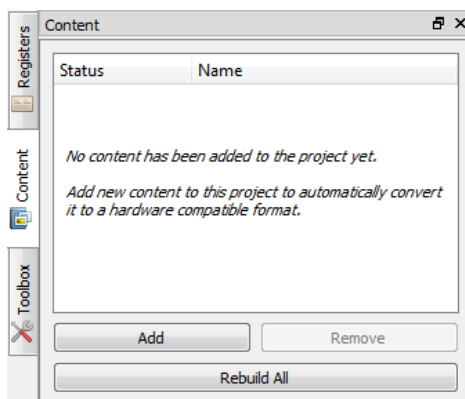


Figure 40 - Content Manager

Add the content

Via content manager, the user can 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 the Add button in Content Tab.
2. The Load Content dialog opens up to browse to the file to be added.

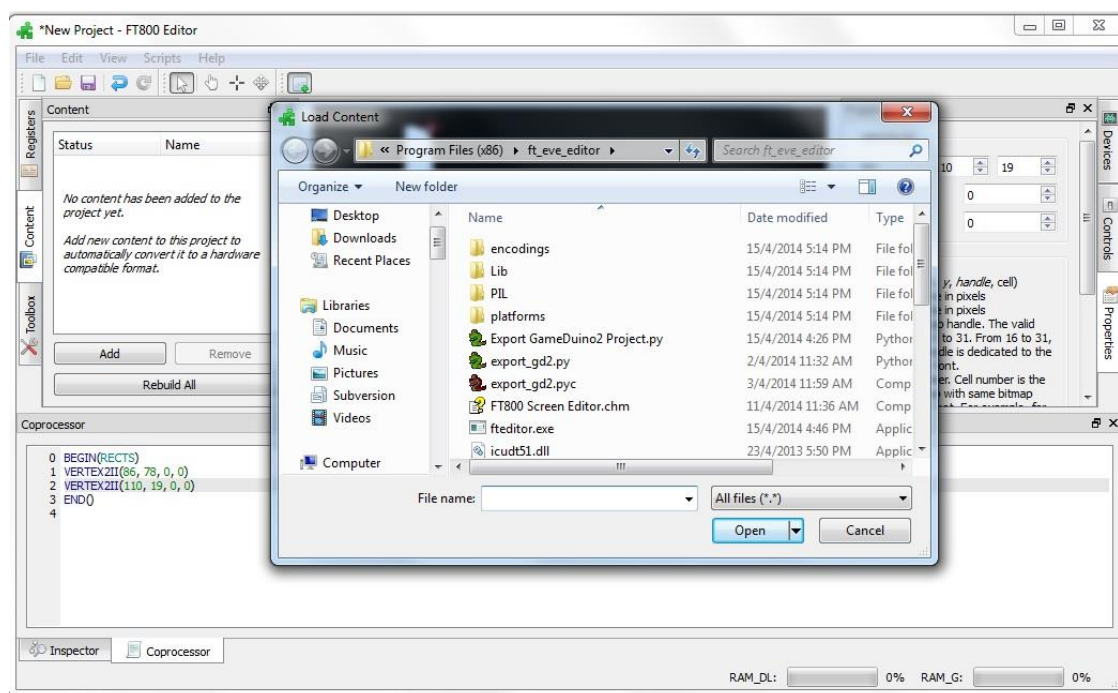
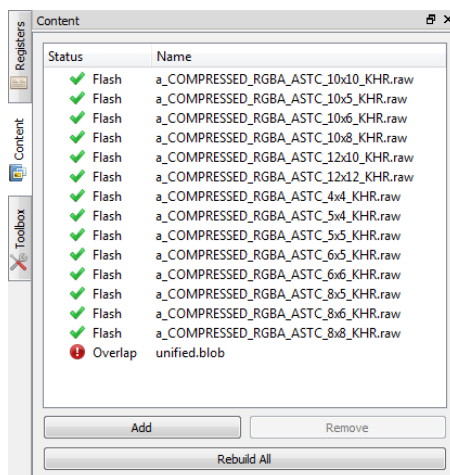


Figure 41 - Load content dialog

3. After the content is added successfully, a green check mark will be next to the item name, the content is available for configuration in the Properties tab.


Figure 42 - 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.

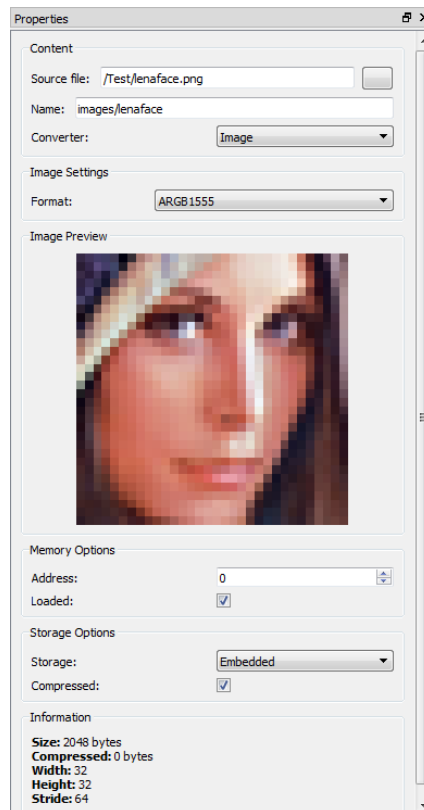


Figure 43 - Content properties

5. If the content is raw data, simply select the "Raw" option in the Converter drop down menu because the already converted raw data does not need further processing. After the data has been successfully loaded, the user 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.

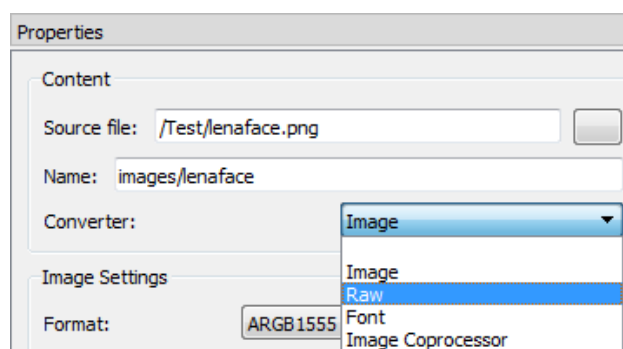


Figure 44 - Raw converter

6. After conversion is done with image data, the user can drag the image from the content manager and drop it into the viewport:

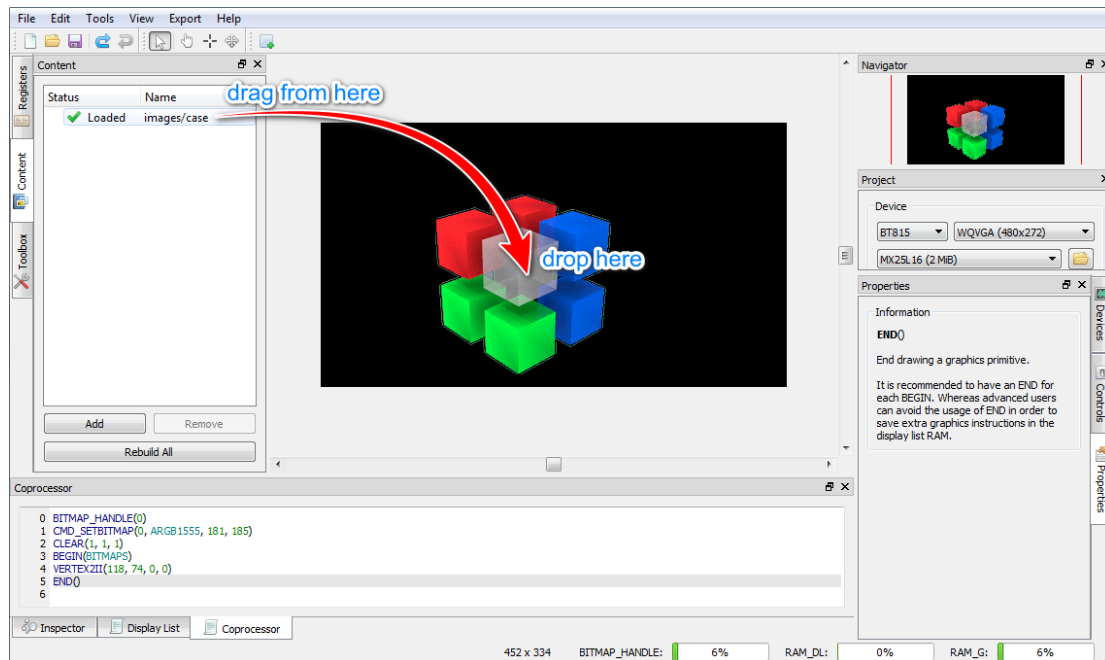


Figure 45 - Drag content into Viewport

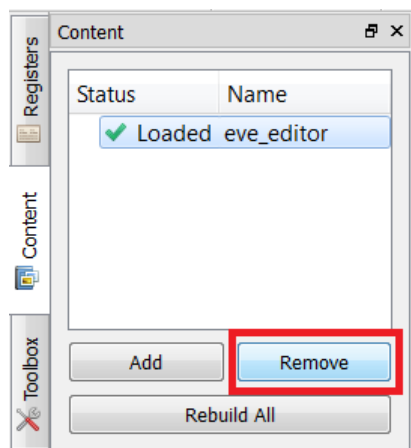
After the user places the image, the display list will be generated in the editor automatically, appended after the current focused command.

Remove added content

This button removes the selected bitmap or raw data in the content manager and thus clears the content manager.

To perform this function:

- Select the content to be removed and click the Remove button



- The selected content is then removed from the list

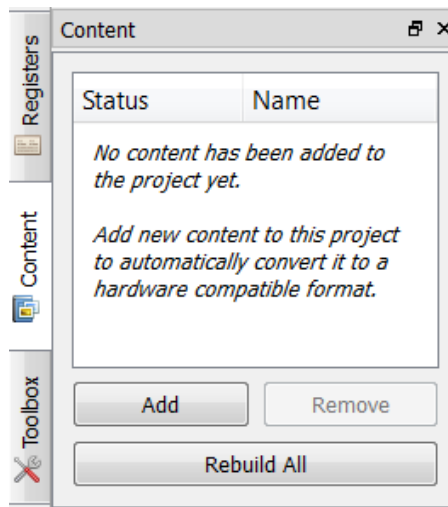


Figure 46 - Remove content

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

The topics in this section provide information about the Controls and Properties tab in the FTDI 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.

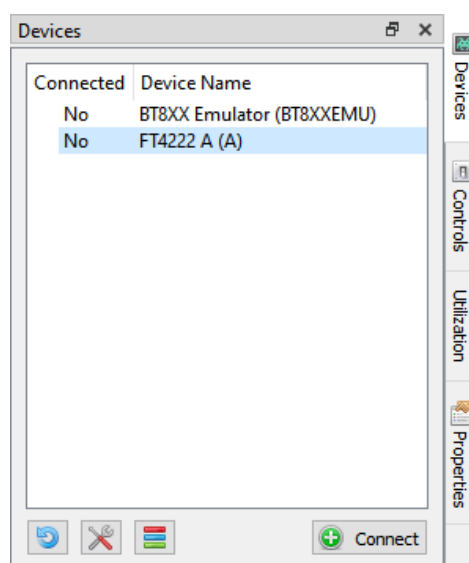


Figure 47 - Before connect

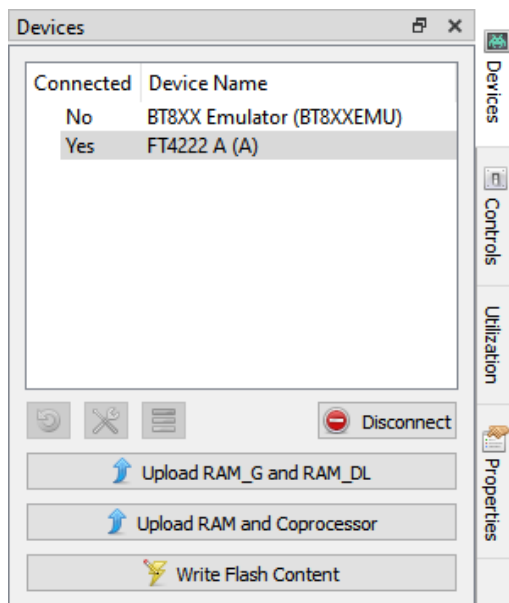


Figure 48 - Connected device

Device type can be changed by clicking the wrench and screw driver button and then selecting the correct display device type. Built-in device is displayed in bold font. Custom device is displayed in regular font.

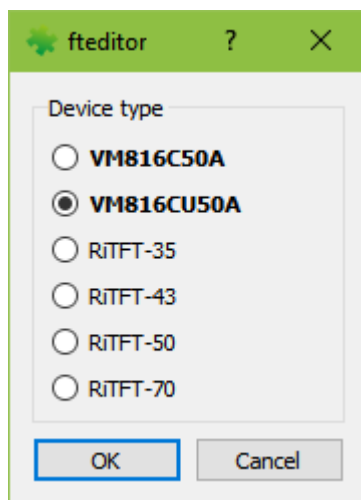


Figure 49 - Device type

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

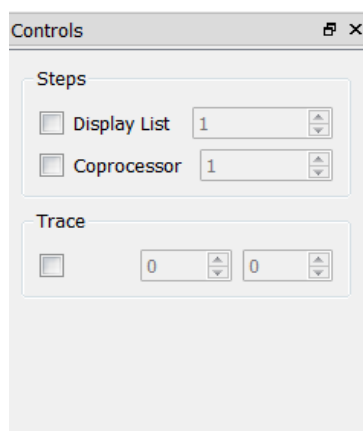


Figure 50 - Controls tab

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 time and the respective tab has to be focused.

Please check 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.

Please check 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.

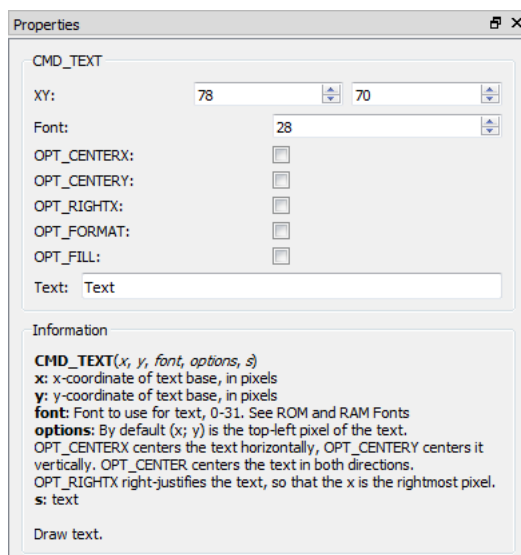
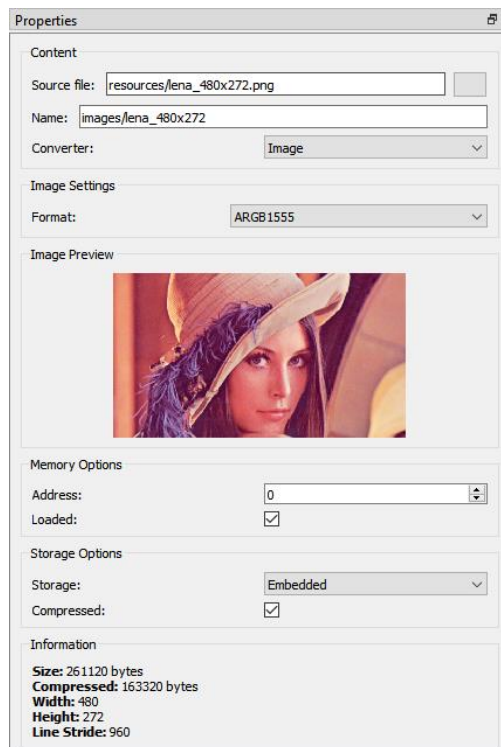


Figure 51 - Properties tab

This tab also provides information of 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.

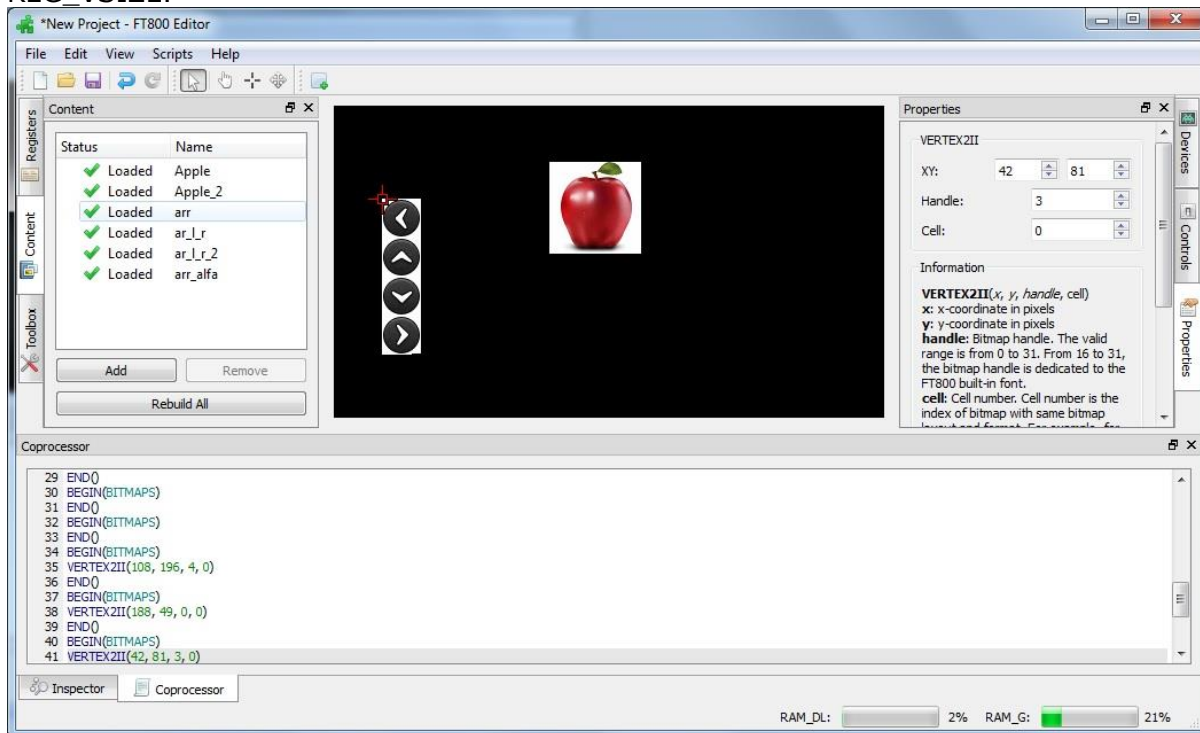


Figure 52 – View port

G. Navigator

Viewport navigator provides the convenient way to edit the whole screen of view port, especially for large resolution.

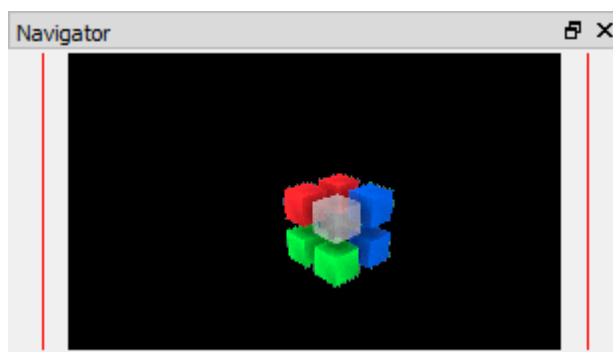


Figure 53 – Navigator

H. Project settings

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

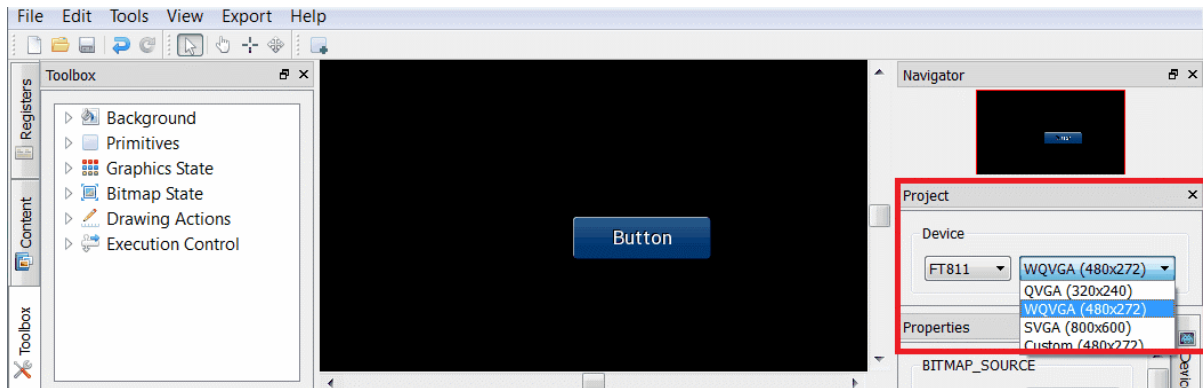


Figure 54 - Project settings

From BT815 device an above, flash is supported.

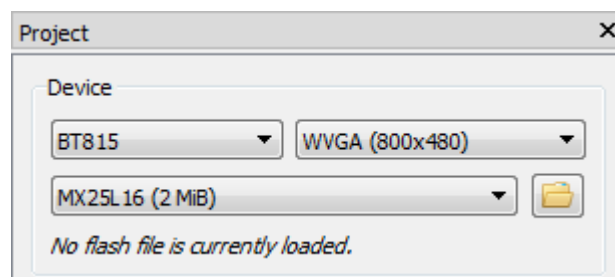


Figure 55 - Flash supported

Open file button is used to select flash file is local PC and load it.

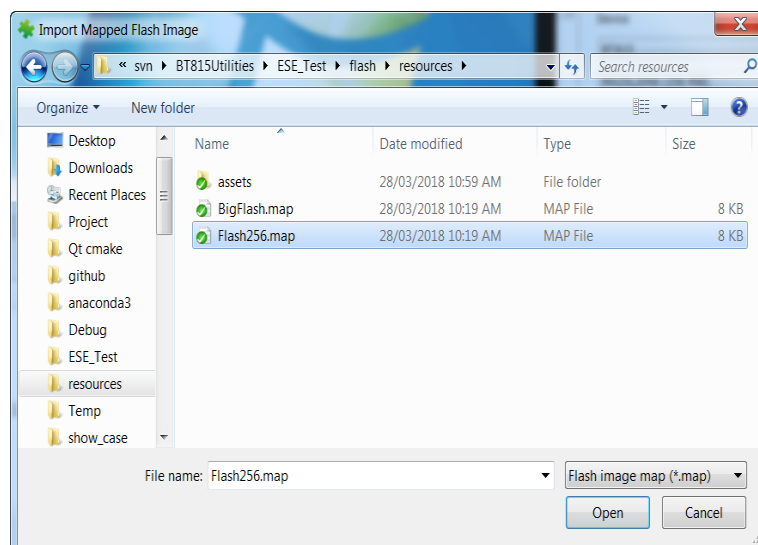


Figure 56 - Load flash file

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

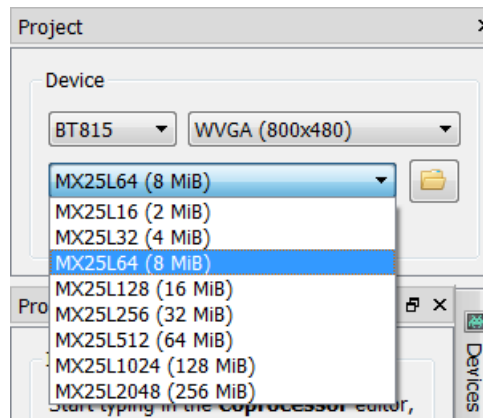


Figure 57 - Select flash size

After flash file is loaded, its path is show for convenient.

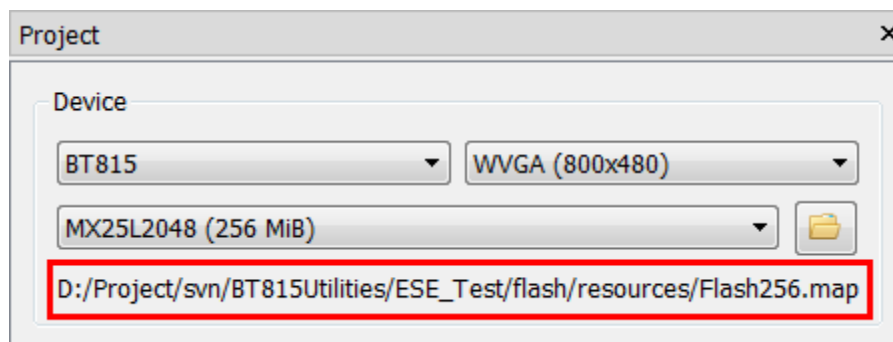


Figure 58 - Flash path

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
Open Recent Project 1->5	Alt + 1, Alt + 2, ..., Alt + 5

V Quick Start Tutorials

The tutorials in this section provide a brief guide on 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 familiarize the user with the basic principles and the way the editor works. For full details on the procedures described in the tutorials please refer to the Basic Working Procedures section.

A. 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 choose the desired color in the Properties of the command or edit the values of the command in the command output.

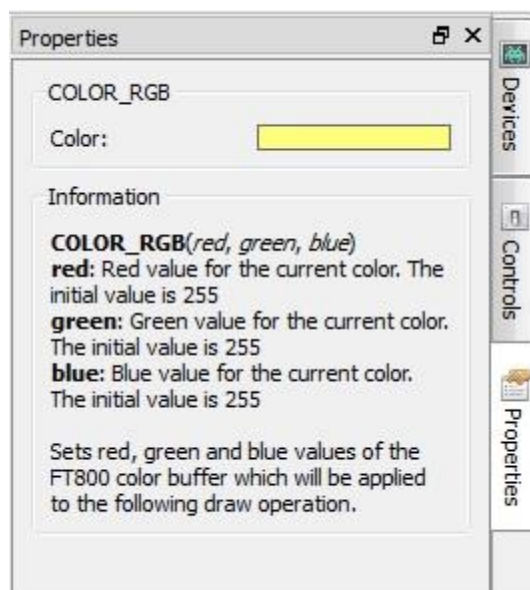


Figure 59 - Change the color

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

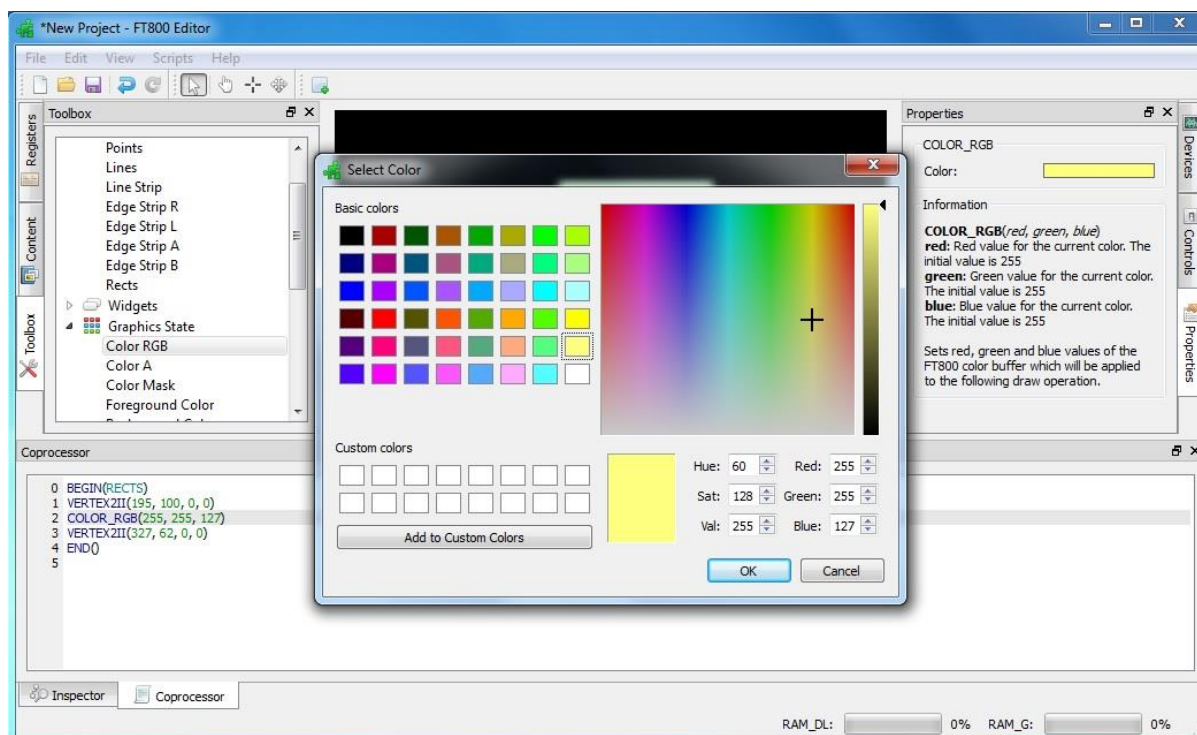


Figure 60 - Select color

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

B. 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 the Remove button in the Content tab.

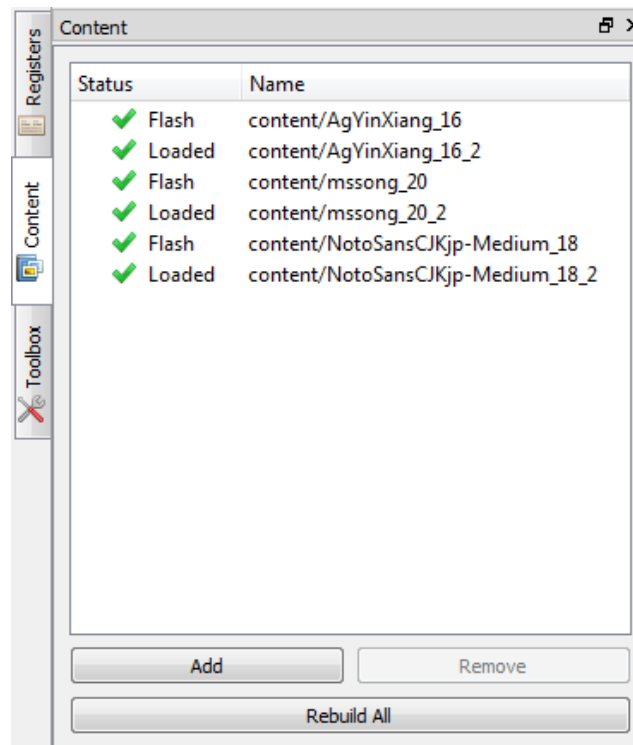


Figure 61 - Import content

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

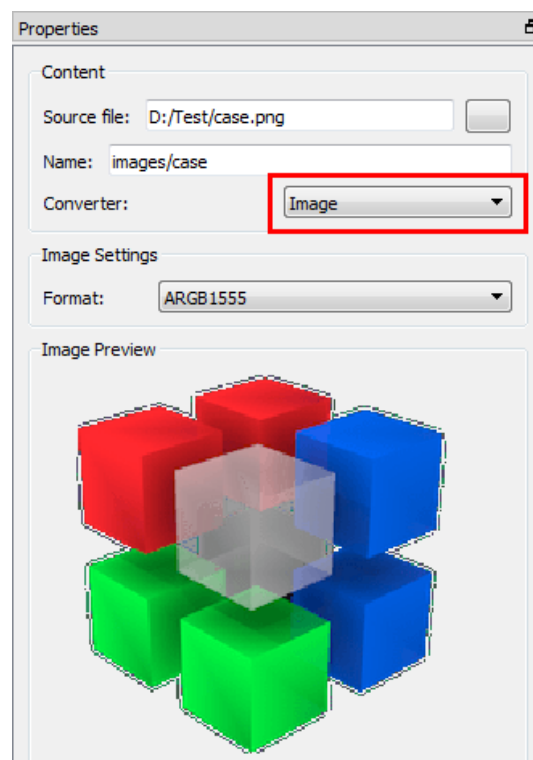


Figure 62 - Select converter image

After the image data has been successfully added, 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.

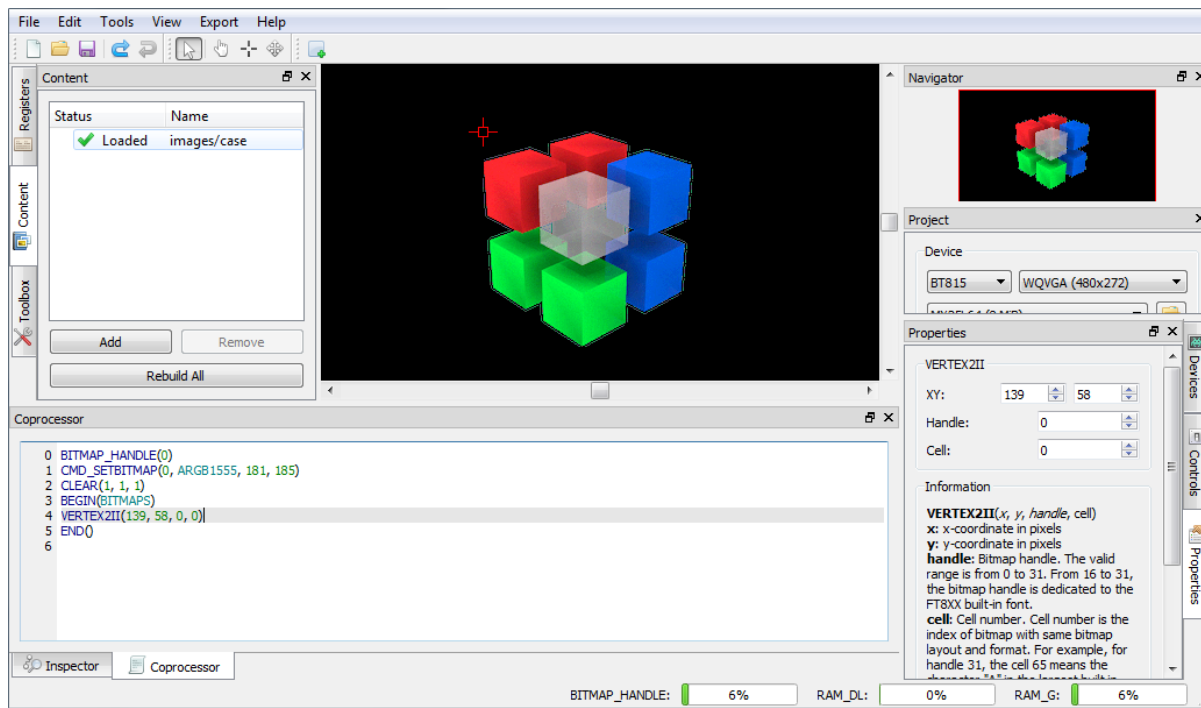


Figure 63 – Drag & drop image

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.

C. 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 coprocessor command.

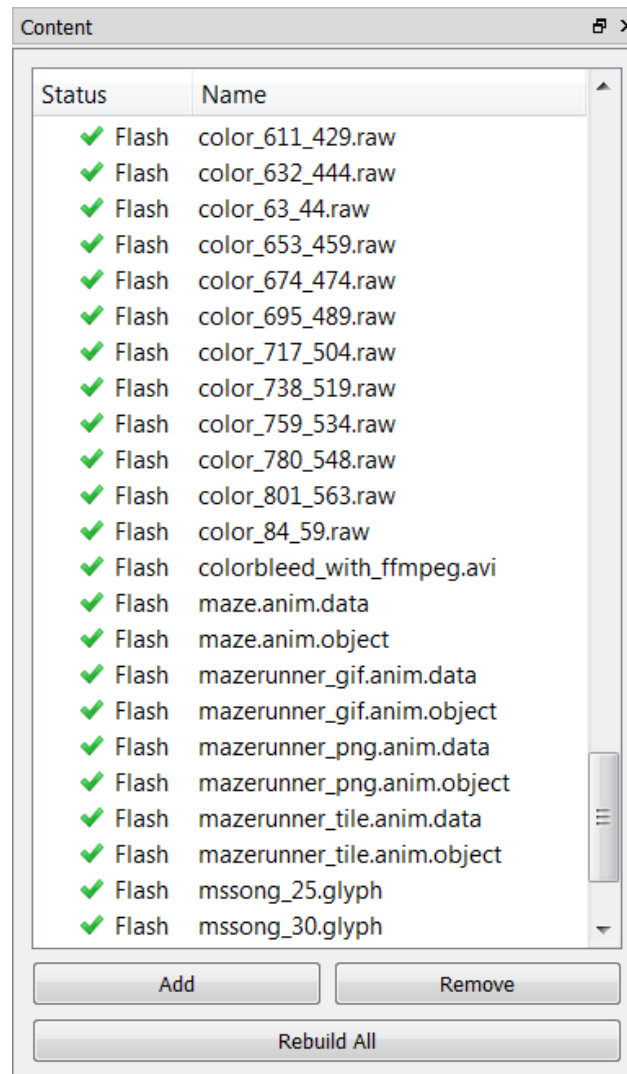


Figure 64 - Import flash

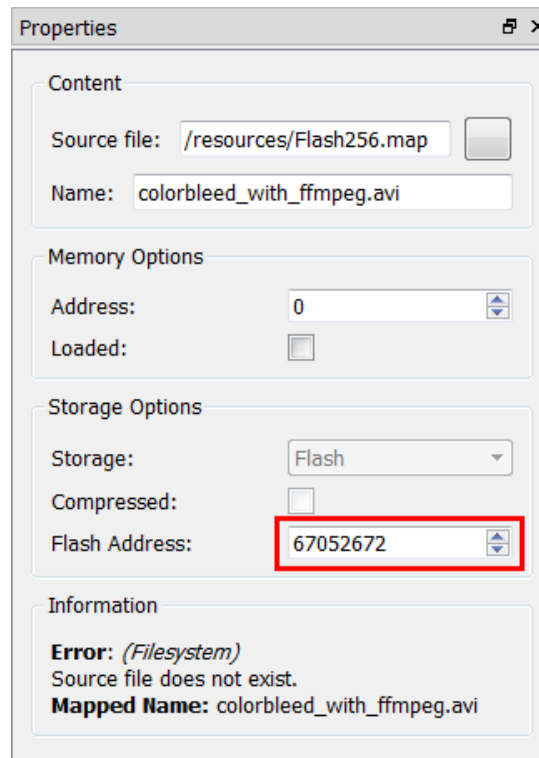


Figure 65 - Flash address

D. Open the project

To open a saved project, simply click the open button on the toolbar or select File->Open then browse to the saved project in the pop up file browser window.

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

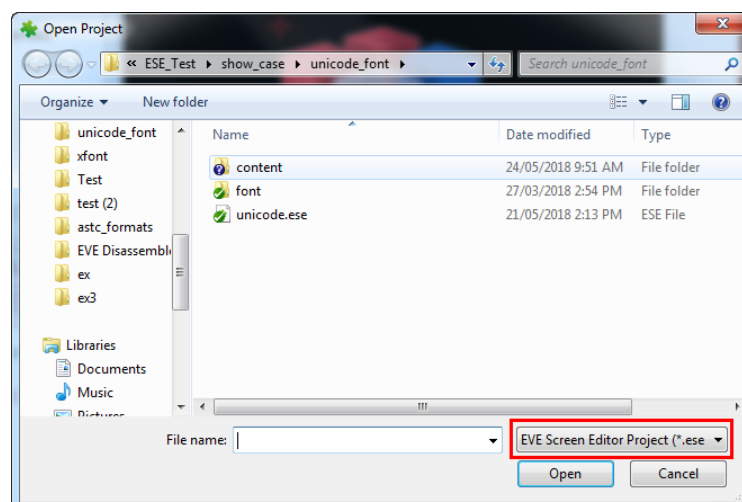


Figure 66 - Open project

E. Save your design

The current project can be saved by clicking the save button on the toolbar, File->Save in the menu, or 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 menu then Save As.

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

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

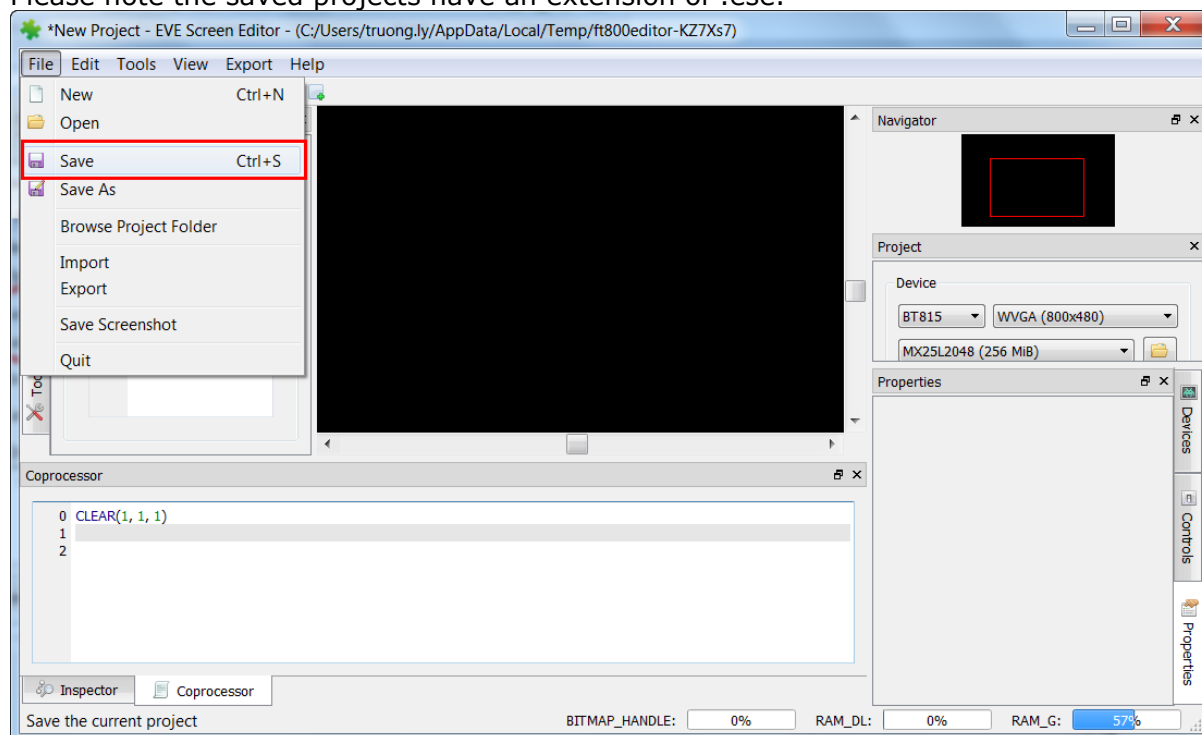


Figure 67 - Save the project

F. Export the project

After creating the screen design, the user 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.

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

Please note that the Content Manager does not apply strict naming convention on the loaded items, but when 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 the Export menu in the menu bar.
2. Select the option to which the project is to be exported.

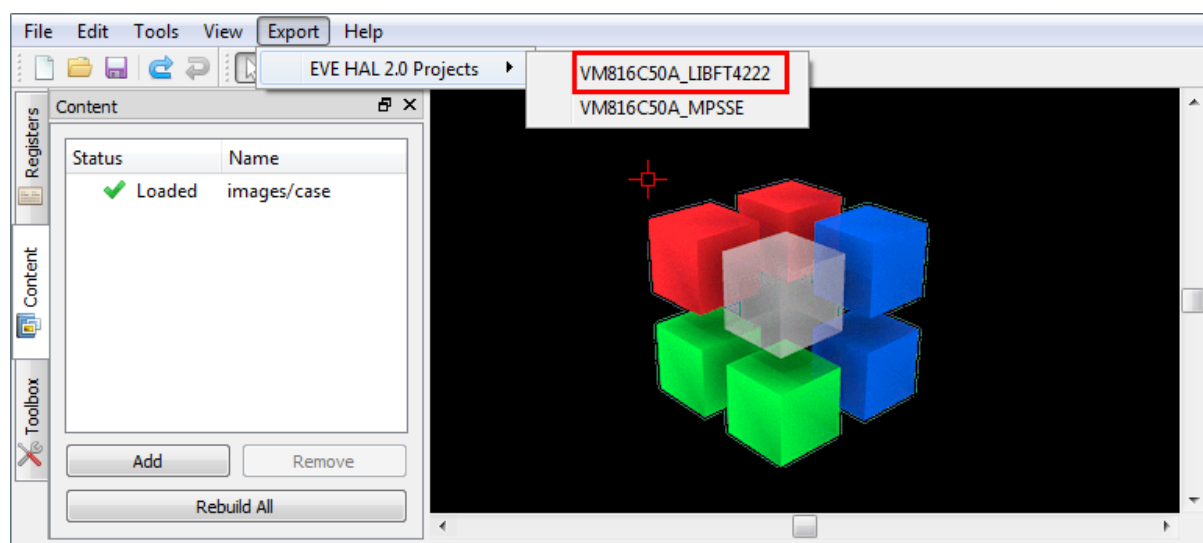


Figure 68 - Export project

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 and 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 module data sheet for hardware connection information.

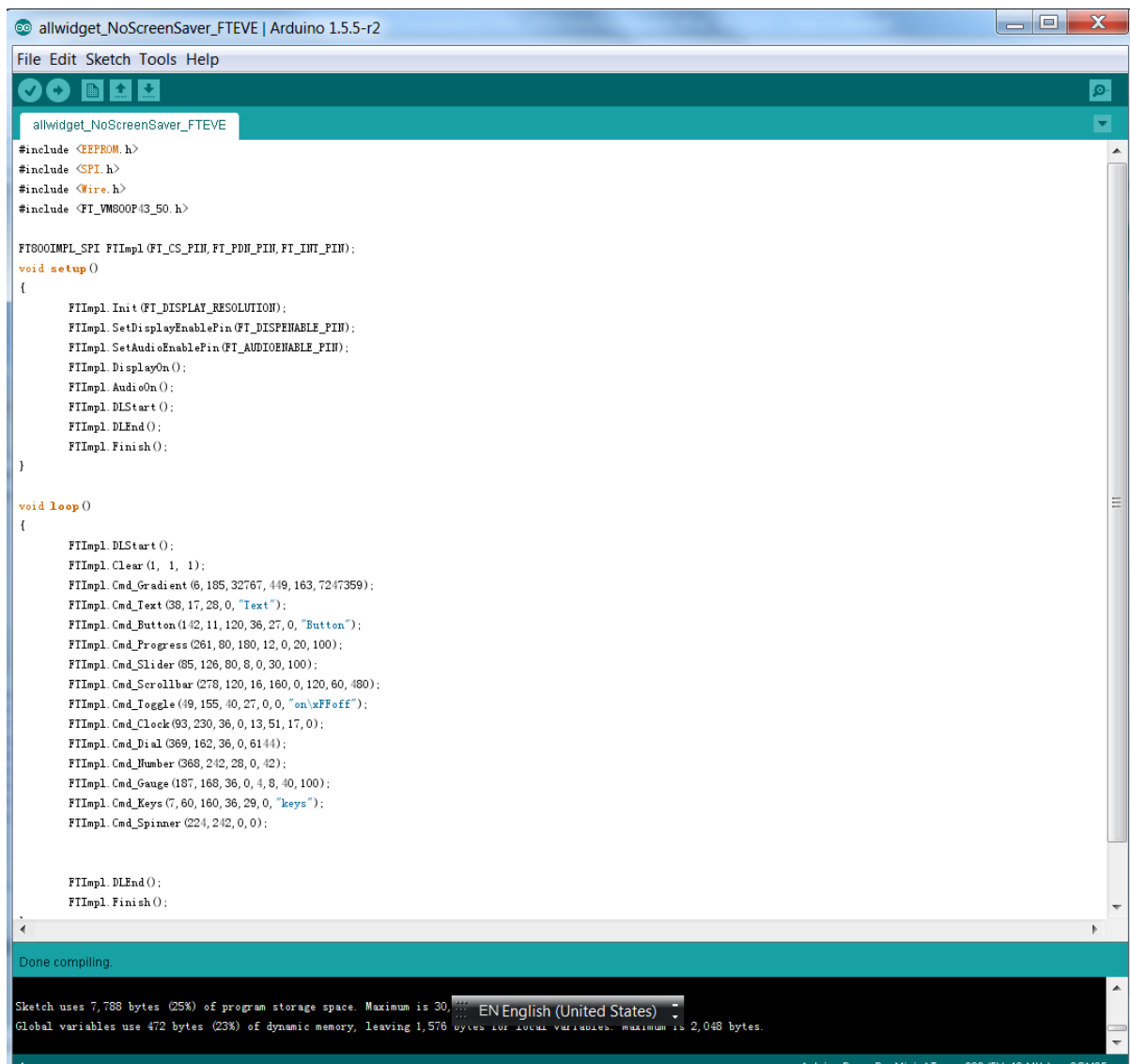


Figure 69 - Arduino IDE

G. 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 device to assign the new custom fonts with one unused bitmap handle. By default, the first unused bitmap handle is zero.

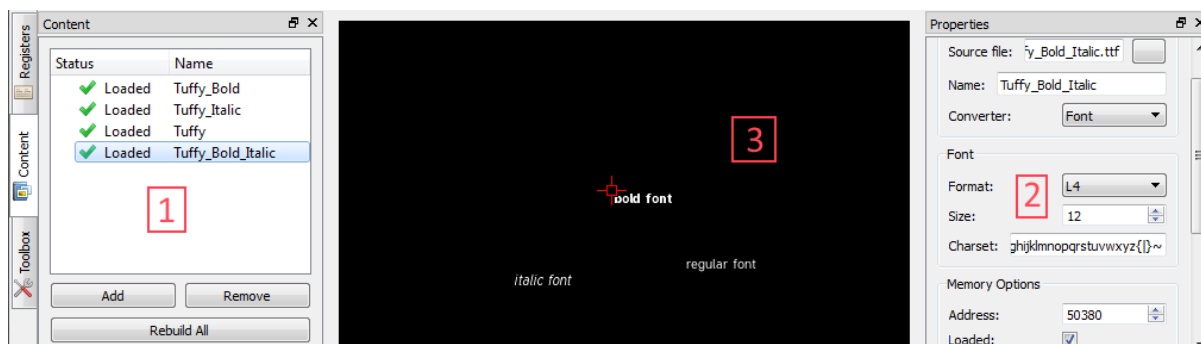


Figure 70 - Custom font

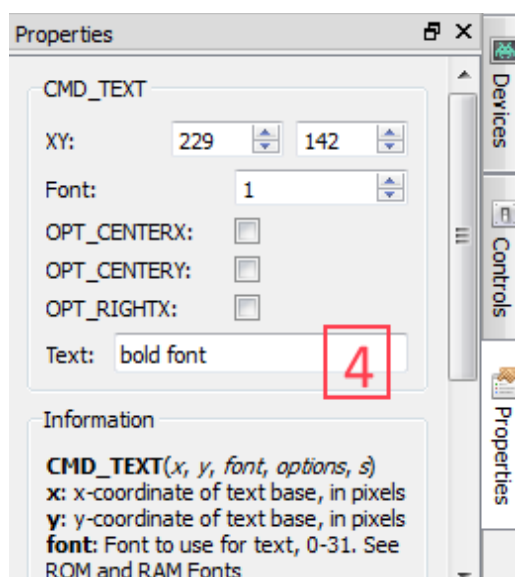


Figure 71 - Property "Text" of custom font

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

1. Constrain vertical

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

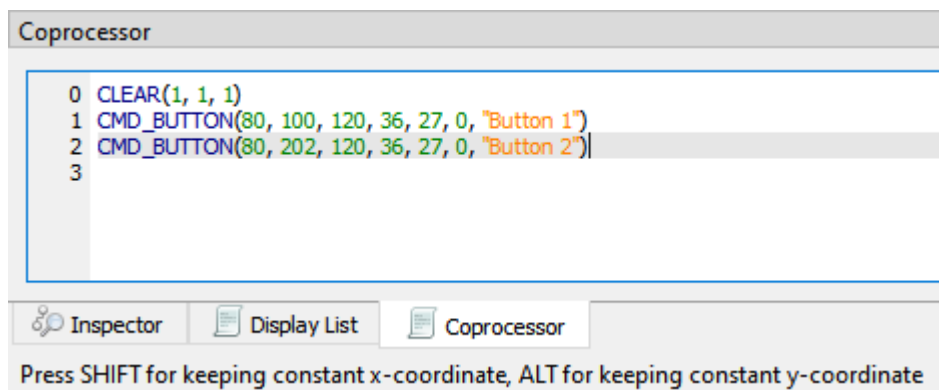


Figure 72 - Press SHIFT for constrain vertical



Figure 73 - Slide up/down to set y-coordinate

2. Constrain horizontal

- Prepare two objects which need 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

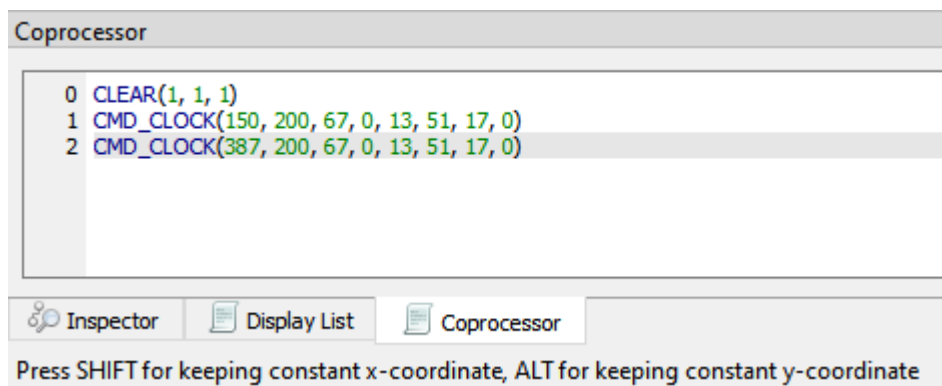


Figure 74 - Press ALT for constrain horizontal

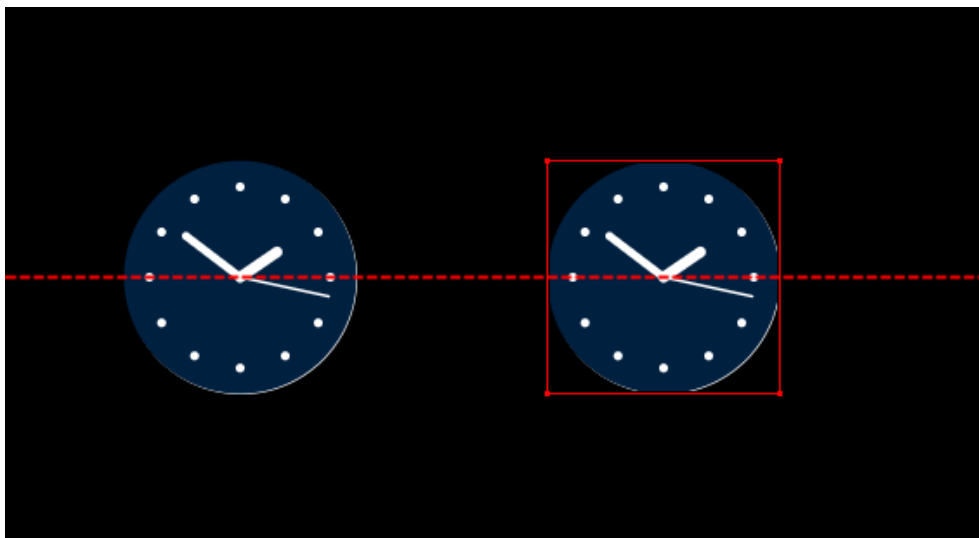


Figure 75 - Slide left/right to set x-coordinate

VI Command Usage Examples

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

A. CMD_PLAYVIDEO

CMD_PLAYVIDEO plays back an MJPEG-encoded AVI video. This command can only be used in FT81X 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 decompress the specified JPEG or PNG data into an FT81X bitmap, in RAM_G. PNG image source can only be specified in FT81X 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 no 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. Mediafifo is not required to decode a bitmap file, otherwise 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()
```

Extra 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 no overwrite the data by load 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 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. 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 display list manually. This command is only supported in FT81X 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()
```


Extra 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()
```

Extra Information:

User can also use the "Capture Snapshot" button 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 coprocessor command which tracks user's touch input and updates the memory content accordingly. It applies FT80X and FT81X devices.

Prototype:

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

Parameter Description:

X, Y - The coordinator 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()
```

Extra Information:

Please note the mouse shall be touch mode by clicking 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 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()
```

Extra Information:

User can also use the "Capture Snapshot" button 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 reluctant to manually change each 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 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
```

Extra 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 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 coprocessor 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 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 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");
```

Extra Information:

Bitmap font handles 32 - 34 are only available on FT81X 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 for FT81X devices 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 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()
```

Extra 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.

Prototype:

CMD_SETFONT2(font, ptr, firstchar)

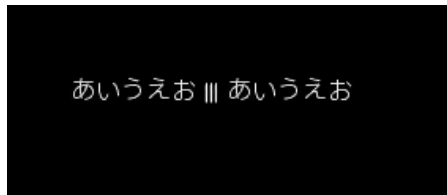
Parameter Description:

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

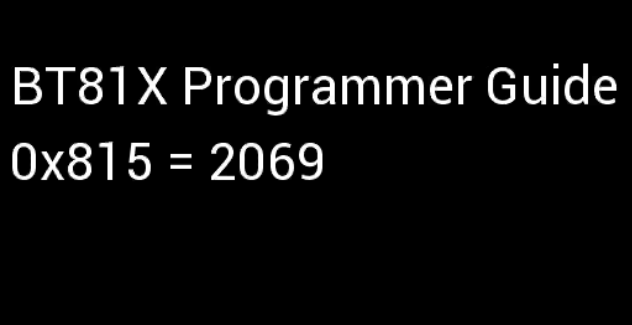
font - Font to use for text, 0-31.

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:



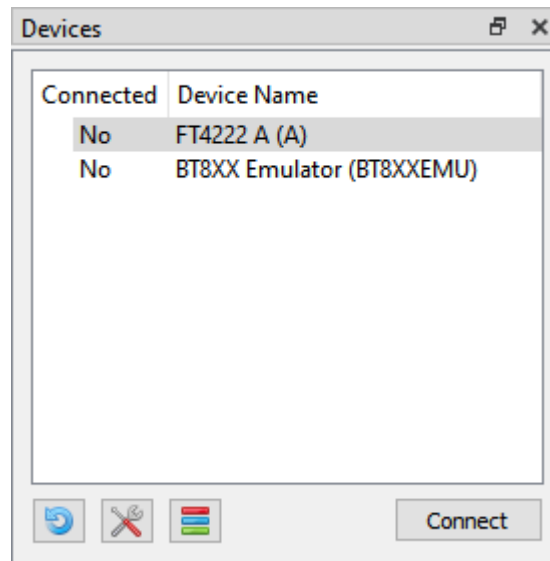
```
CMD_TEXT(12, 49, 31, OPT_FORMAT, "BT%dX Programmer Guide", 81)  
CMD_TEXT(11, 110, 31, OPT_FORMAT, "0x%X = %d", 2069, 2069)
```

VII Working With EVE Screen Editor

The topics in this section provide information about usability of the EVE Screen Editor.

A. Connect with Hardware

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



After the user clicks the "Connect" button, the device is ready to be synchronized with the Screen Editor.

Note: Don't forget to click the wrench and screw driver button to select the correct device type.

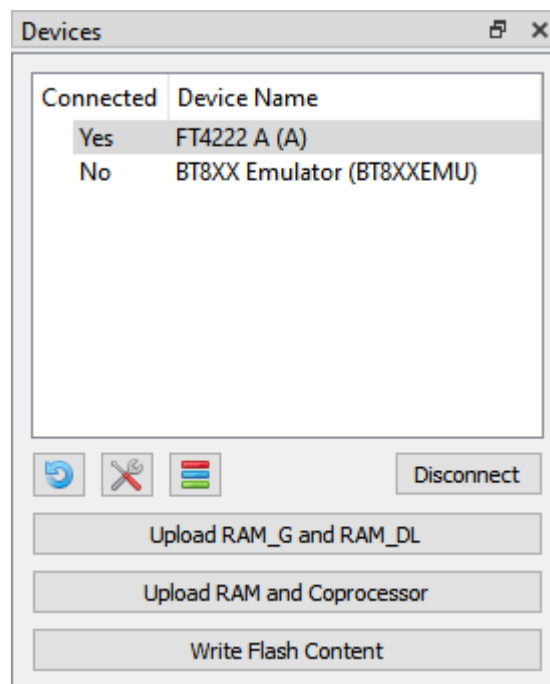


Figure 76 - Device connected

If the project loads content from flash image, we need to write flash first by pressing button "Write Flash Content". Button "Upload RAM_G and RAM_DL" is used for

synchronizing Display List and button "Upload RAM and Coprocessor: is used for synchronizing Coprocessor command.

Note: Make sure the connected device(1) has the same type of EVE chip on board as the emulator(2) runs on.
 Check the picture below:

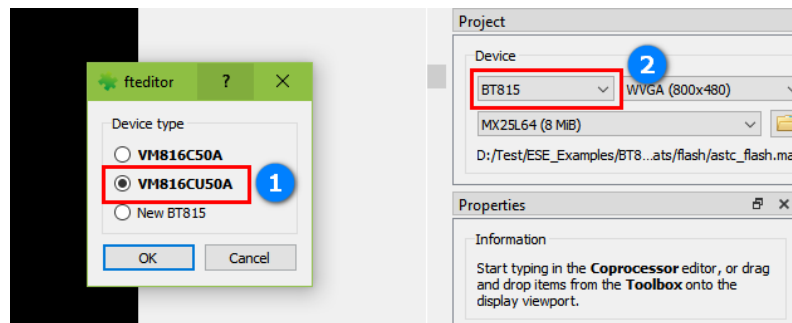


Figure 77 - Select correct device type

Managing device

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

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

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

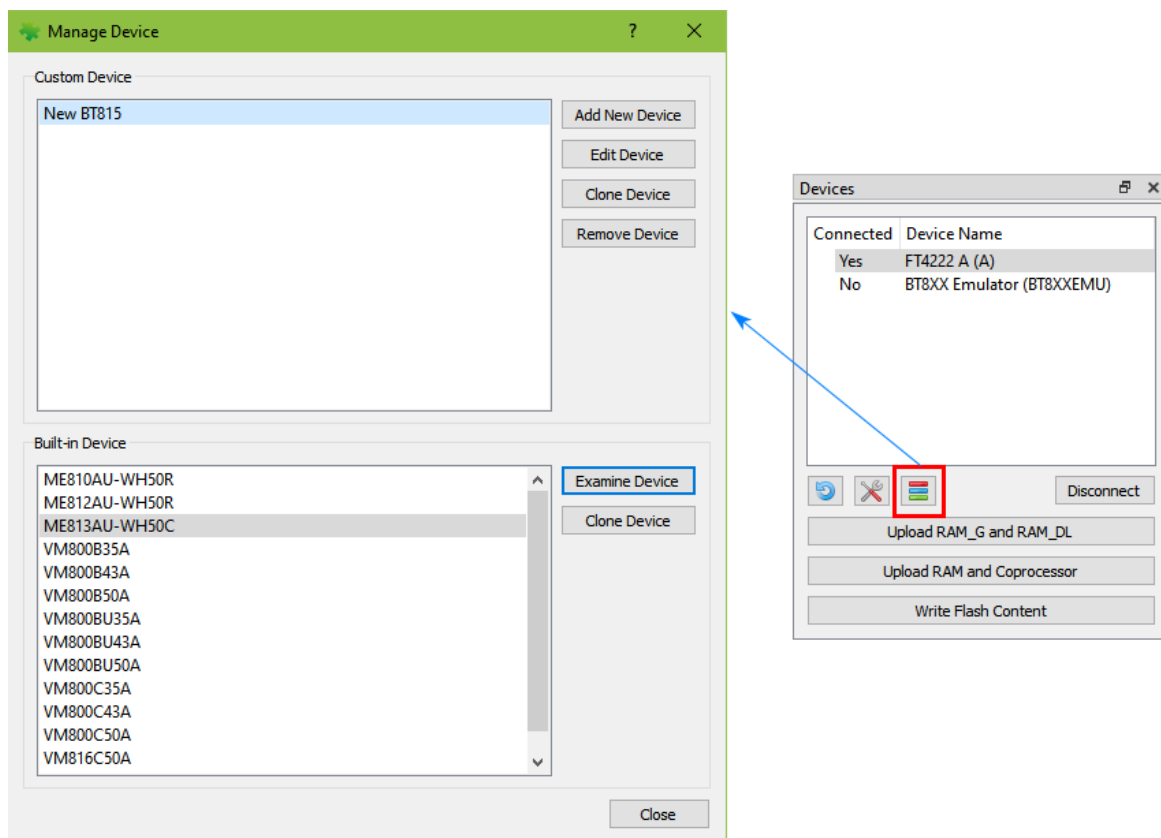


Figure 78 - Managing device

B. Export your project

To export the current screen to certain platform specific API based source code, users may click "Export" menu to choose the corresponding platform name.

Currently, ESE has following built-in capability to generate the code for:

- EVE Arduino Library
- GameDuino 2 Library
- FTDI EVE HAL
- FTDI EVE HAL 2.0 (FT81X only)

C. Check your design

The topics in this section explain how to check the design.

1. Step by Step

The user can select to execute the display list or coprocessor 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 coprocessor editor box will execute the specified steps and highlight the specific display list or coprocessor commands.

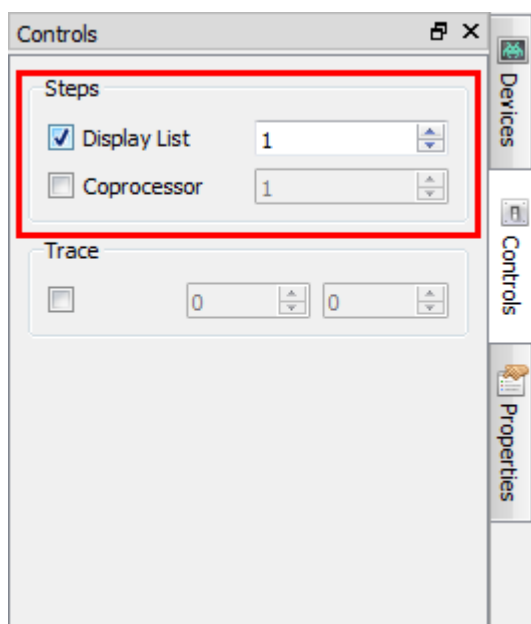


Figure 79 - Select Display List/Coprocessor

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.

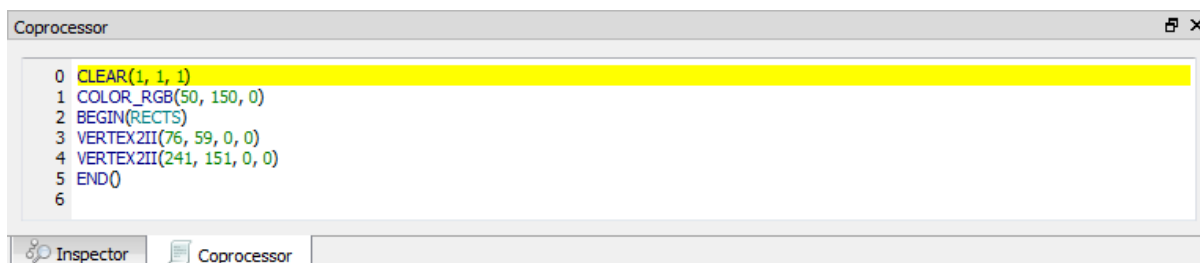
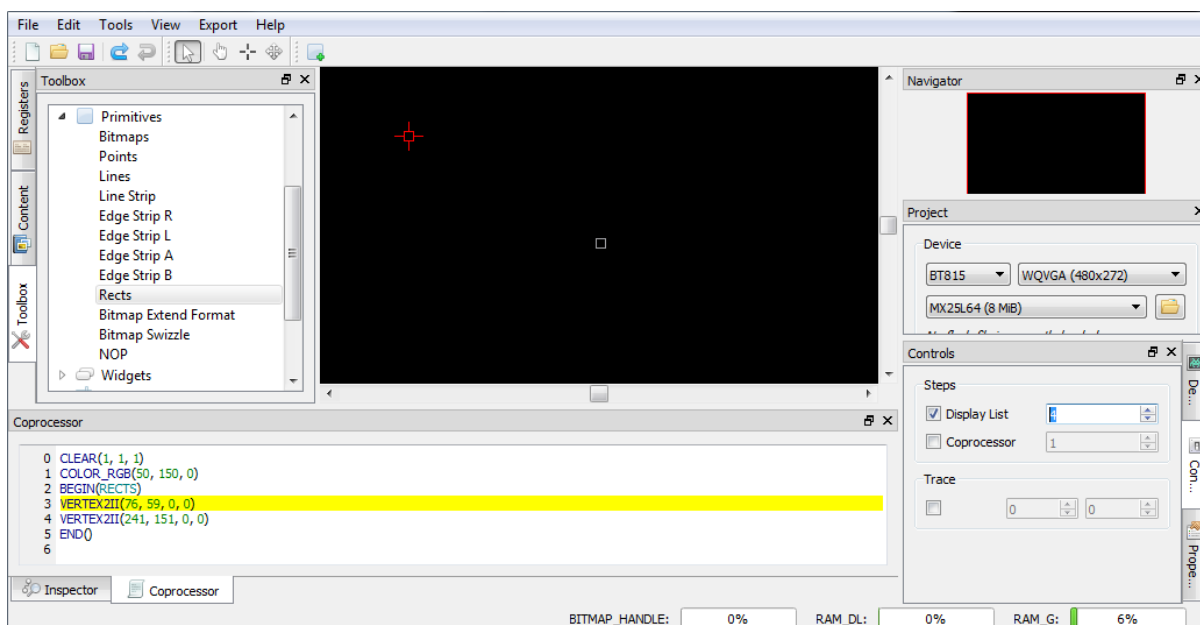
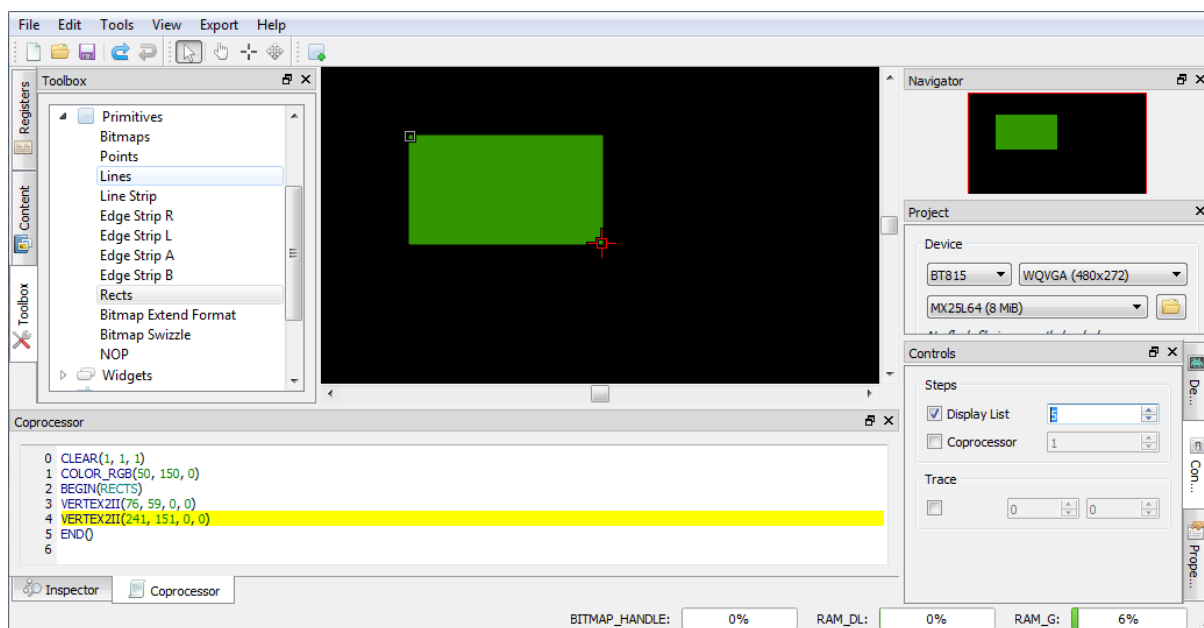


Figure 80 - Display List command is highlighted yellow



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



2. Trace the pixel

The user 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 are more than 1 object occupying the trace coordinate, the top most object will get highlighted in a brighter green than those under it.

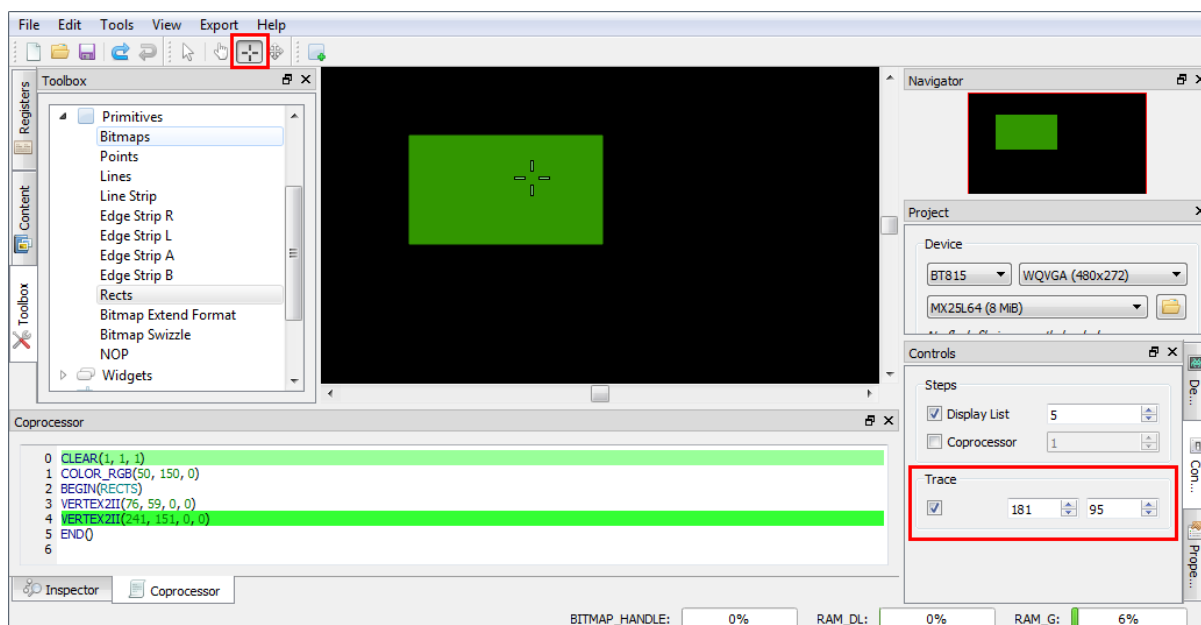


Figure 81 - Trace the pixel

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.

D. Example Project

The examples are easily accessible from the "Examples" folder in the installation directory, they can be opened in the screen editor using the Open option in File menu.

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

Opened "allwidget_NoScreenSaver" project.

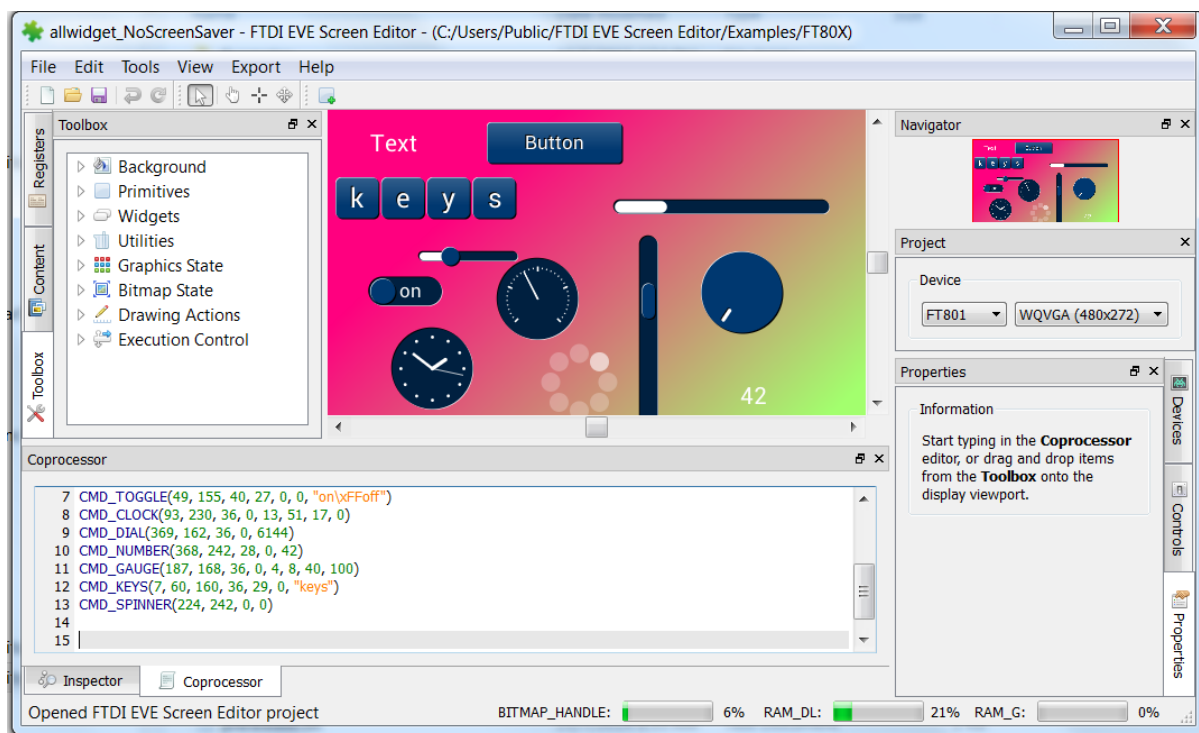


Figure 82 - Open example project

VIII Disclaimer

The topics in this section contain the license agreements for the EVE Screen. Any other third party software or artifacts included in the software are listed in "3rd party" sub-menu of menu "Help".

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