

ESE 3.0 User Guide

Version 1.0

Issue Date: 2018-05-25

This document is the FT Editor User Guide for the FT8XX and BT8XX series chip (where x stands for any value of 0 - 9). It describes the necessary information for programmers developing display, audio or touch applications with the FT8XX, BT8XX (EVE) series devices.

# 

# Introduction

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| The topics in this section provide basic information about the FTDI EVE Screen Editor (ESE), what it is for and what you can do with it.  The current version of ESE is 3.0.0       |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | How to get started?     |  |  | | --- | --- | |  | Study the Introduction chapter and [The User Interface](#topic_User_interface) sections to familiarize yourself with the program. |      |  |  | | --- | --- | |  | Work through the [Quick Start Tutorials](#topic_Quick_start_tutorials) to familiarize yourself with using this Help & Manual. |      |  |  | | --- | --- | |  | Check out the [Examples Project](#topic_Examples) for more information. | | |

## Introduction

|  |
| --- |
| FTDI EVE Screen Editor (ESE) is a GUI tool that provides an intuitive "drag&drop" user experience to construct screen design without programming.  Moreover, because it is empowered by FTDI cutting edge EVE emulator, it 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 FTDI VM800B Series board and MPSSE cable, the screen design shown in the tool can be synchronized with the real hardware immediately with only a few mouse clicks. Please check this [topic](#topic_Connect_with_hardware).    Last, but not least, there are more exciting features, such as "tracing and step by step", waiting to be discovered.  Just go ahead! |

## Major Features

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ESE has the following major features:     |  |  | | --- | --- | |  | Export project |   This feature exports the current project to FT800 Arduino Library, GameDuino 2 Library, and HAL/HAL 2.0(FTDI) Library.     |  |  | | --- | --- | |  | Import content |   This feature imports the bitmaps configuration from PNG and JPEG files and raw data.     |  |  | | --- | --- | |  | Export feature |   Users can export the current project into platform specific project by single mouse click.     |  |  | | --- | --- | |  | Device Manager |   The device manager displays the detected device (VM800B board) and allows it to synchronize with the detected device.  The VM800B board is connected to a Windows PC via the FTDI MPSSE cable. Please check [here](http://www.ftdichip.com/Products/Modules/VM800B.html) for details.     |  |  | | --- | --- | |  | Drag and Drop |   This feature allows the components and the commands to be placed directly in the view port to create graphics effects as well as automatically generating the command list. No programming is required at all!!! |

## What is New

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| In ESE 3.X, the following new features are introduced:   |  |  | | --- | --- | |  | BT815 support |   Support for the Adaptive Scalable Texture Compression (ASTC) algorithm  Support for the flash file which packages all resources such as: font, image, movie, etc.   New generic extension name for project file  ".ese" is introduced to obsolete ".ft8xxproj" extension name for project file.   Support unicode string  Support \uXXXX and \UXXXXXXXX in string  In ESE 2.X, the following new features are introduced:     |  |  | | --- | --- | |  | FT81X support |   Users are able to create large resolution screen up to 2048 by 2048 pixels.  Additionally, the full set of FT81X commands are supported.     |  |  | | --- | --- | |  | Project setting window |   Users can switch between FT81X and FT80X as well as display resolution dynamically for the project     |  |  | | --- | --- | |  | Navigator window |   Navigator window provides users the ability to view different regions of the screen.     |  |  | | --- | --- | |  | Enhanced toolbox |   Toolbox contains all supported commands ready for users to drag & drop, saving keyboard effort.     |  |  | | --- | --- | |  | Easy access to assets |   Load assets directly using relative file path of assets.   |  |  | | --- | --- | |  | Export projects |   Export menu replaces previous Scripts menu. Context sensitive export menu based on selected device type.   New generic extension name for project file  ".ft8xxproj" is introduced to obsolete ".ft800proj" extension name for project file.   Tools menu  New Tools menu enables emulator reset and display list capture.   Zoomable viewport  Viewport window supports zoom-in and zoom-out using mouse scroll wheel. |

|  |  |
| --- | --- |
|  |  |

## System Requirement

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| To install and run the FTDI EVE Screen Editor, the recommended system requirement is as 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](#topic_Export_the_project), users are recommended to install the following software:   |  |  | | --- | --- | |  | [Arduino IDE](http://arduino.cc/en/main/software) | |  | [Gameduino 2 library](http://excamera.com/sphinx/gameduino2/code.html) | |  | [EVE Arduino Library](http://www.ftdichip.com/Support/SoftwareExamples/EVE/FTDI_V1.2.0_28042014.zip) (1.2.0 and above) |    Microsoft Visual Studio C++ 2010 IDE or newer is required to compile the HAL MSVC(MPSSE) projects. [VM800B](http://www.ftdichip.com/Products/Modules/VM800B.html) (3.5" , 4.3'' or 5.0'' display) with [MPSSE cable(USB to SPI cable)](http://www.ftdichip.com/Products/Cables/USBMPSSE.htm) are required to run the project.   Microsoft Visual Studio C++ 2012 IDE is required to run HAL FT800 Emulator projects.  The following hardware can be used to verify the design in device manager:   |  |  | | --- | --- | |  | [VM800B](http://www.ftdichip.com/Products/Modules/VM800B.html) (3.5" , 4.3'' or 5.0'' display) with [MPSSE cable(USB to SPI cable)](http://www.ftdichip.com/Products/Cables/USBMPSSE.htm) .This is selected in the device manager of the screen editor tool. |     To build and verify projects on Arduino IDE, the following boards are needed:   |  |  | | --- | --- | |  | [VM800P/VM801P (3.5" , 4.3" or 5.0" display)](http://www.ftdichip.com/Products/Modules/VM800P.html) for exported EVE Arduino library based project | |  | [Gameduino 2 board](http://excamera.com/sphinx/gameduino2/) with [Arduino Pro board](http://arduino.cc/en/Main/ArduinoBoardPro) for exported Gameduino2 library based project | |

## Reference Documents

|  |
| --- |
| These documents provide the background knowledge to work with the FTDI EVE Screen Editor efficiently.They are available at the following links:  1) [FT800 Data Sheet](http://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_FT800_Embedded_Video_Engine.pdf)  2) [FT800 Programmers Guide](http://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT800_Series_Programmer_Guide.pdf)  3) [FT81X Data Sheet](http://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_FT81x.pdf)  4) [FT81X Programmers Guide](http://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT81X_Series_Programmer_Guide.pdf) |

## Known Issues and Limitations

|  |
| --- |
| The following are known issues and limitations in this release:  1. 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.  2. Both FT80X based and FT81X based modules are supported.  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. |

# User Interface

|  |
| --- |
| The topics in this section provide information about the user interface of the FTDI EVE Screen Editor. |

## Overview

|  |
| --- |
| This is the main layout of the graphical user interface. |

## Menu,Toolbar and Status Bar

|  |
| --- |
| The topics in this section provide information about the menus, toolbar and status bar in ESE. |

### Menu

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

#### File Menu

|  |
| --- |
| The file menu consists of New, Open, Save, SaveAs and Quit menu options.          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" format.  Example projects can be viewed [here](#topic_Examples).    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.  Import  Import memory dump (\*.vc1dump)  Export  Export memory dump (\*.vc1dump)    Save Screenshot  Save the current screen of viewport window into local PC.  Quit  This closes the application. |

#### Edit Menu

|  |
| --- |
| The edit menu allows users to revert or redo certain actions.        Undo  Undo the last action done in the editor.    Redo  Redo the last action done in the editor. |

#### Tools menu

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

|  |
| --- |
| 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

|  |
| --- |
| Internally, the Screen Editor 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.              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.      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. |

### ToolBar

|  |
| --- |
| This toolbar defines shortcuts of mouse operation for New, Open, Save, Undo, Redo, Cursor, Touch, Trace, Edit and Insert functionality.      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](#topic_CMD_SKETCH).    Trace  Force trace cursor in viewport. See [Trace the pixel](#topic_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 |

### Status Bar

|  |
| --- |
| This status bar shows the consumption status of RAM\_G and RAM\_DL as well as bitmap handles. |

## 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. |

### Coprocessor Command Editor

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The features are as below:     |  |  |  | | --- | --- | --- | |  | | Full set commands support and autocompletion | |  | | 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. | |

### Display List Editor

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The features are as below:     |  |  | | --- | --- | |  | Display list commands autocompletion | |  | 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. | |  |  | | |

### Inspector

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

#### 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.    RAM\_DL can be selected line by line then copy to another text editor. |

#### 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.    RAM\_REG can be selected line by line then copy to another text editor. |

## Toolbox ,Content Manager and Registers

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

### Tool box

|  |
| --- |
| 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:  The toolbox will be enabled as below:      All display list commands are grouped into different categories based on functionality(as in FT81X project):   Background   Primitives   Graphics State   Bitmap State   Drawing Actions   Execution Control |

#### Coprocessor mode

|  |
| --- |
| To use the coprocessor mode, coprocessor editor window shall be selected as below:  The toolbox will be enabled as below:        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 |

### 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. |

### Content Manager

|  |
| --- |
| The topics in this section provide information about the content manager of the FTDI EVE Screen Editor.  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. |

#### 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. |           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.            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.        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.  edit box.    6. After conversion is done with image data, the user can drag the image from the content manager and drop it into the 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 in the Content Tab. |        |  |  | | --- | --- | |  | The selected content is then removed from the list | |

#### Rebuild the content

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

## Devices,Controls and Properties

|  |
| --- |
| The topics in this section provide information about the Controls and Properties tab in the FTDI EVE Screen Editor. |

### Device Manager

|  |
| --- |
| The Device Manager enables user to connect the FTDI [VM800B](http://www.ftdichip.com/Products/Modules/VM800B.html) series board with the PC and observe the design directly on hardware.      The V800B boards with 3.5", 4.3" and 5.0" display are supported. The default display configuration works on both V800B 4.3" and 5.0" displays. Display device type can be changed by clicking the wrench and screw driver button and then selecting the correct display device type.  Note: The Horizontal and Vertical input fields in the Registers dock change the viewPort dimensions only, the display configurations when syncing with the device are determined by the selected display device type. |

### Controls

|  |
| --- |
| 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](#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](#topic_Trace_the_pixel)" for more details |

### Properties

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

## ViewPort

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

## Navigator

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

## Project settings

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

With BT815 device, flash is supported.

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

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

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

## Keyboard Shortcuts

|  |
| --- |
| The following keyboard shortcuts can be used in the screen editor:    New -> Ctrl + N  Save -> Ctrl + S  Undo -> Ctrl + Z  Redo -> Ctrl + Y  Cut -> Ctrl + X  Copy -> Ctrl + C  Paste -> Ctrl + V  Zoom In/Out of Viewport -> Ctrl + wheel of mouse |

# 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. |

## Change the color

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Subsequent drawing color can be changed by the [drag and drop](#topic_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.      The Properties tab of the Color RGB command can change the color visually by clicking on the color bar and select a color..          In the EVE code syntax, the following commands have the color channels as their parameters (in the order of red, green and blue):     |  |  | | --- | --- | |  | COLOR\_RGB | |  | CLEAR\_COLOR\_RGB | |  | CMD\_GRADIENT | |  | CMD\_BGCOLOR | |  | CMD\_FGCOLOR | |  | CMD\_GRADCOLOR | |

## Import the content

|  |
| --- |
| Importing the content adds the bitmap or raw data to the content tab. The data added will be listed in the content tab and can be used in the construction of display screens by dragging and dropping the data into the view port. The raw and bitmap data can be added to the list as explained in [Add Content](#topic_Add_image_and_raw_data_into_ram_g).The added data can be removed by selecting an entry and clicking the Remove button in the Content tab.          If the content added is an image, select the "Image" mode of Converter in properties tab:        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.    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. |

## 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.

## Open the project

|  |
| --- |
| To open a saved project, simply click the open button on the toolbar or select File->Open and 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. |

## 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 FTDI EVE Screen Editor.  Please note the saved projects have an extension of .ese . |

## 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. |         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. |

## 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.

# 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.

## 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 videoback is in progress. CMD\_DLSTART() should be specified right after CMD\_PLAYVIDEO to continue display the subsequent display commands.

## 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.

Usage:

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 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)

CMD\_LOADIMAGE(0, 0, "../EiffelTower\_800\_480.jpg") //decode EiffelTower jpeg image. Put data to the 0th offset in RAM\_G and use default options.

BEGIN(BITMAPS)

VERTEX2II(0, 0, 0, 0)

END()

CMD\_LOADIMAGE(768000, 0, "../lenna256.png") //decode lenna256 png image. Put data after the EiffleTower decoded data and use mediafifo buffer for decoding.

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

## 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.

## 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 output the capture bitmap to a ARGB4 raw file or as a JPEG or PNG image.

## 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:

## 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.

## 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 the value has changed or Tools->"Reset Emulator" option is selected.

## 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()

## 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)

Extra Information:

None.

## 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 rang 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

## 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.

# Working With EVE Screen Editor

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

## Connect with Hardware

|  |
| --- |
| After the user has designed the screen on the PC, if user has the VM800B board connected with the PC, the device manager can be enabled to observe the effect on actual hardware.    The picture below shows how the board connection was made.              After clicking the refresh button, the device shall be discovered and appears in the list box. The user needs to select the desired device to connect.      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.        After the user clicks "Sync With Device" button, the designed screen will be shown on hardware immediately.  Note: Make sure the connected module(1) has the same type of EVE chip on board as the emulator(2) runs on.  Check the picture below: |

## 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](http://www.ftdichip.com/Support/SoftwareExamples/Eve/FTDI_V1.2.0_28042014.zip) | |  | [GameDuino 2 Library](http://excamera.com/sphinx/gameduino2/code.html#gd2code) |    FTDI EVE HAL   FTDI EVE HAL 2.0 (FT81X only)  Users may refer to section [Export the project](#topic_Export_the_project) for more details. |
|  |

## Check your design

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

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

### 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. |

### Drag and Drop

|  |
| --- |
| The commands in the [toolbox](#topic_Coprocessor_mode_toolbox) may be dragged and dropped in the [viewport](#topic_Viewport). The editor will be updated with the corresponding commands. |

## Example Project

|  |
| --- |
| The examples are easily accessible from the "Examples" folder in the installation directory, they can be opened in the screen editor using the Open option in File menu.  "Examples" folder contains two sub-folders "FT81X" and "FT80X" for FT81X specific and FT80X specific example projects.    Opened "allwidget\_NoScreenSaver" project. |

# Disclaimer and License Agreement

|  |
| --- |
| The topics in this section contain the license agreements for the FTDI EVE Screen Editor and Python.  The EVE screen editor has the python 2.7.9 interpreter built-in to make it extensible.  Any other third party software or artifacts included in the software are listed in "3rd party" sub-menu of menu "[Help](#topic_Help_menu)". |

## 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. Future Technology Devices International Ltd will not accept any claim  for damages howsoever arising as a result o f 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 |

## Python License Agreement

|  |
| --- |
| Python License (Python-2.0)    PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2  --------------------------------------------    1. This LICENSE AGREEMENT is between the Python Software Foundation  ("PSF"), and the Individual or Organization ("Licensee") accessing and  otherwise using this software ("Python") in source or binary form and  its associated documentation.    2. Subject to the terms and conditions of this License Agreement, PSF  hereby grants Licensee a nonexclusive, royalty-free, world-wide  license to reproduce, analyze, test, perform and/or display publicly,  prepare derivative works, distribute, and otherwise use Python  alone or in any derivative version, provided, however, that PSF's  License Agreement and PSF's notice of copyright, i.e., "Copyright (c)  2001, 2002, 2003, 2004, 2005, 2006 Python Software Foundation; All Rights  Reserved" are retained in Python alone or in any derivative version  prepared by Licensee.    3. In the event Licensee prepares a derivative work that is based on  or incorporates Python or any part thereof, and wants to make  the derivative work available to others as provided herein, then  Licensee hereby agrees to include in any such work a brief summary of  the changes made to Python.    4. PSF is making Python available to Licensee on an "AS IS"  basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR  IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND  DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS  FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON WILL NOT  INFRINGE ANY THIRD PARTY RIGHTS.    5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON  FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS  A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON,  OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.    6. This License Agreement will automatically terminate upon a material  breach of its terms and conditions.    7. Nothing in this License Agreement shall be deemed to create any  relationship of agency, partnership, or joint venture between PSF and  Licensee. This License Agreement does not grant permission to use PSF  trademarks or trade name in a trademark sense to endorse or promote  products or services of Licensee, or any third party.    8. By copying, installing or otherwise using Python, Licensee  agrees to be bound by the terms and conditions of this License  Agreement. |

Your index page goes here...

In MS-Word, select INDEX AND CONTENTS from the INSERT menu.

Select INDEX and click OK.