



Bridgetek
BRIDGING TECHNOLOGY

EVE Asset Builder 2.10 User Guide

Document Version: 2.3

Issue Date: 11-10-2023

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, 1 Tai Seng Avenue, Tower A, #03-05, Singapore 536464. Singapore Registered Company Number 201542387H. © Bridgetek Pte Ltd.

Contents

I. Preface	4
A. Purpose	4
B. Related documents.....	4
C. Acronyms and Abbreviations	4
II. Overview	5
A. Introduction.....	5
B. Key Features	5
C. Credits.....	5
III. Setup and Installation	6
A. System Requirements	6
B. Installing EVE Asset Builder	6
IV. Getting Started	9
A. EVE Chip Series	9
B. Session	9
1. Start Session.....	10
2. Load Session.....	10
3. Stop Session.....	10
C. Command Prompt.....	11
D. Log Window	12
E. Naming Convention and Usage of Output Files	12
F. Image Utilities.....	14
1. Image Converter	14
2. PNG/JPEG Validator	15
3. JPEG Optimizer.....	16
G. Video Converter	17
H. Audio Converter	18
I. Font Converter	19
1. Legacy Format	20
2. Extended Format [BT81X]	21
3. Font Encoder	22
J. Animation Converter	22
K. Flash Utilities	24
1. Flash Image Generator	24
2. Flash Programmer	26
3. Flash Diagnostic.....	30
4. Sample Code	31
L. Asset Compressor	32
1. Compressor	32
2. Validator/Decompressor	33
M. Custom Touch	34
N. Binary To C-Array	35
O. Disassembler.....	36
P. Display List Optimizer.....	38
Q. SampleApp	39
V. Example.....	42
A. Image Converter	42
B. Video Converter	43
C. Audio Converter	45
D. Font Converter	46
E. Font Encoder.....	47
F. Animation Converter and Generate Flash Image	49
Contact Information	52
Appendix A – List of Figures.....	53
Appendix B – List of Tables	54

Appendix C – Revision History	55
--------------------------------------------	-----------

I. Preface

A. Purpose

This document describes the features and procedures involved in using the **EVE Asset Builder (EAB)**.

B. Related documents

- [AN_391 EVE Platform Guide](#)
- [FT8xx and BT8xx Series Programming Guides](#)
 - BRT_AN_033_BT81X Series Programming Guide
 - BRT_AN_088_FT81x_BT88x Programming Guide
 - FT800 Series Programmers Guide
 - FT800 Programmers Guide in simplified Chinese

C. Acronyms and Abbreviations

Terms	Description
EVE	Embedded Video Engine
EAB	EVE Asset Builder
HAL	Hardware Abstraction Layer
MSVC	Microsoft Visual Studio C++
SPI	Serial Peripheral Interface
UI	User Interface
ASTC	Adaptive Scalable Texture Compression
MPSSE	Multi-Protocol Synchronous Serial Engine

II. Overview

A. Introduction

The EVE Asset Builder (EAB) is a software application designed for Windows-based operating systems. It is equipped with a range of utilities that enable the creation of resources for EVE series devices. The purpose of this document is to offer a comprehensive guide on how to effectively utilize the EAB tools.

B. Key Features

The following are the key features of EVE Asset Builder:

- ✚ Image Utilities: Convert bitmaps, validate PNG/JPEG, and optimize JPEG
- ✚ Video Converter: Convert a video to MJPEG format
- ✚ Audio Converter: Convert an audio to Eve-compatible format
- ✚ Font Converter: Convert a font to Legacy/Extended format
- ✚ Animation Converter: Convert a gif file or a sequence of images to an animation
- ✚ Flash Utilities: Generate/Program/Read a flash file
- ✚ Asset Compressor: Compress/Validate an asset to Eve-compatible format
- ✚ Custom Touch: Generate a custom touch firmware
- ✚ Bin2C: Convert the binary file into a C array
- ✚ Disassembler: Disassemble the Eve instructions into human-readable text
- ✚ Display List Optimizer: Optimize display list commands

C. Credits

Open-Source Software

- Qt: <https://www.qt.io/>
- Python: <https://www.python.org/>
- astcenc: <https://github.com/ARM-software/astc-encoder>
- FFmpeg: <https://www.ffmpeg.org/>
- OptiPNG: <http://optipng.sourceforge.net/>
- Jpegtran: <http://jpegclub.org/jpegtran/>
- QtProgressCircle: <https://github.com/mofr/QtProgressCircle>
- libraqm: <https://github.com/HOST-Oman/libraqm>

Icons Copyright

Icons made by [Freepik](#), [Smashicons](#), [Pixel perfect](#), [Dave Gandy](#), [Royyan Wijaya](#), [Recieve](#) icon by [Icons8](#)

Audio Copyright

Audio for testing by [Lesfm](#) from Pixabay

III. Setup and Installation

A. System Requirements

To install the application, ensure that your PC meets the requirements recommended below:

- ✓ Recommended Windows 10 or above; alternatively, Windows 8 or 7 with the latest windows updates
- ✓ 64-bit platform
- ✓ 1.6GHz or faster processor
- ✓ 1GB of RAM (1.5GB if running on a virtual machine)
- ✓ A multi-core CPU is highly recommended
- ✓ At least 512MB of hard disk space
- ✓ Display resolution 1080 x 800 pixels or higher
- ✓ "Write" permission to the installation folder

B. Installing EVE Asset Builder

The following steps will guide you through the EVE Asset Builder *Setup/Installation* process.

- i. Download the package from <https://brtchip.com/eab/>.
- ii. When prompted with a download dialog box. Click on **Save**.
- iii. Navigate to the folder under which the package files are downloaded.
- iv. Extract the zip file contents. Double click on the file **EVE-Asset-Builder-setup.exe**
- v. The EVE Asset Builder Setup Wizard is displayed along with a Welcome message.

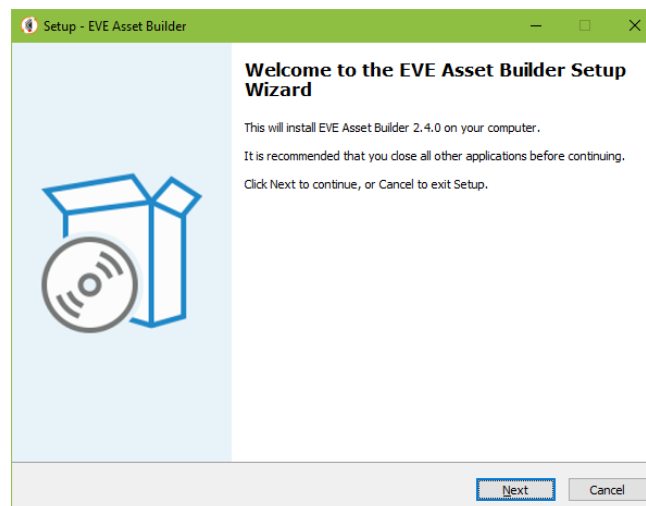


Figure 1 EVE Asset Builder Setup Wizard

- vi. Click **Next** to 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.

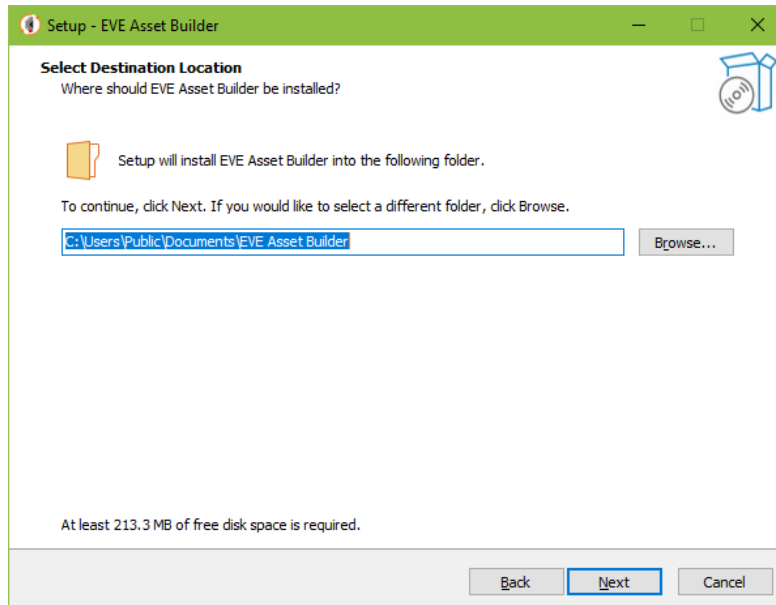


Figure 2 EVE Asset Builder Setup – Select the destination folder

- vii. In the **Select Additional Tasks** window, check the “**Create a desktop / Create Quick Launch shortcut**” boxes, to have the EVE Asset Builder icon and Quick Launch shortcut displayed on the desktop if required. Click **Next** to prepare for the installation.

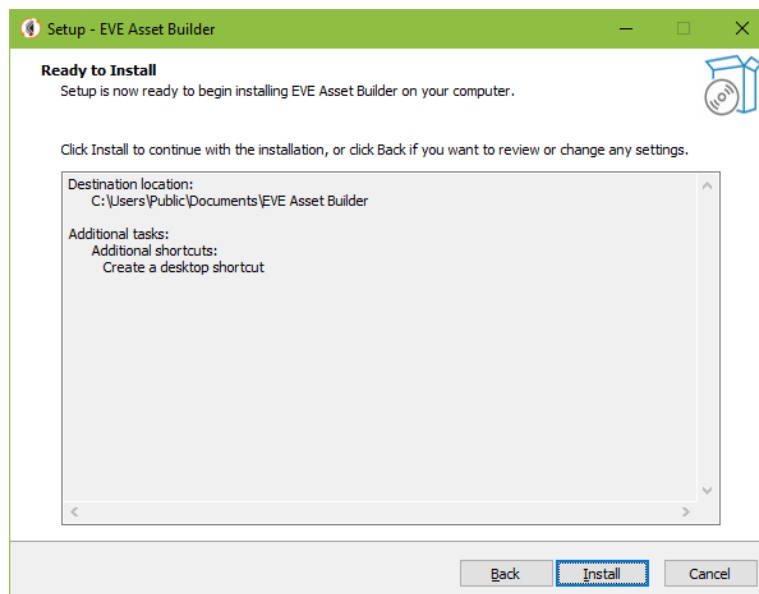


Figure 3 EVE Asset Builder Setup – Ready for Installation

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

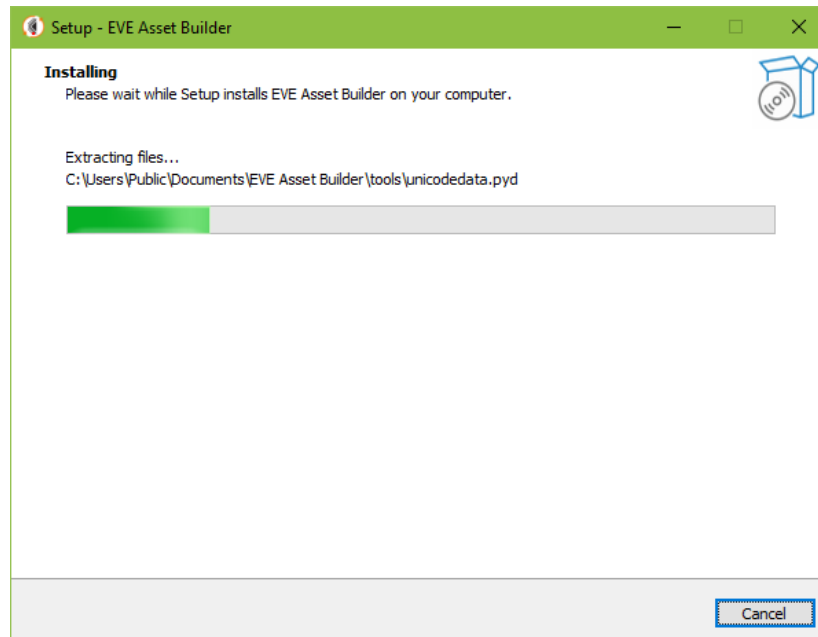


Figure 4 EVE Asset Builder Setup – Installing in Progress

- ix. Upon successful installation, click **Finish**. The EVE Asset Builder application UI is displayed.

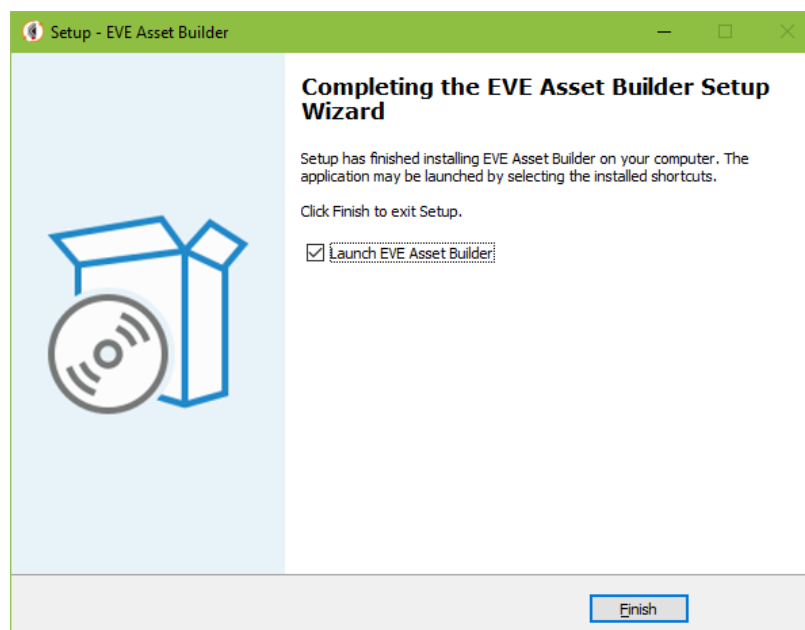


Figure 5 EVE Asset Builder Setup – Finish

IV. Getting Started

A. EVE Chip Series

The available EVE chip generations - BT81X, BT88X/FT81X, and FT80X - can affect which utilities are available once chosen.

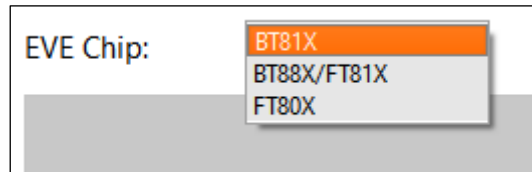


Figure 6 EVE chip series

B. Session

This feature enables the saving of user inputs for later use. Three buttons located on the left edge can be used to load, start, or stop sessions.

START SESSION: Start recording actions from the user, all information is stored in a session file.

STOP SESSION: Stop recording. The user has the option to save or discard the session.

LOAD SESSION: Read a session file and restore all inputs.

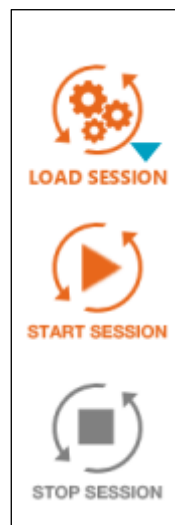
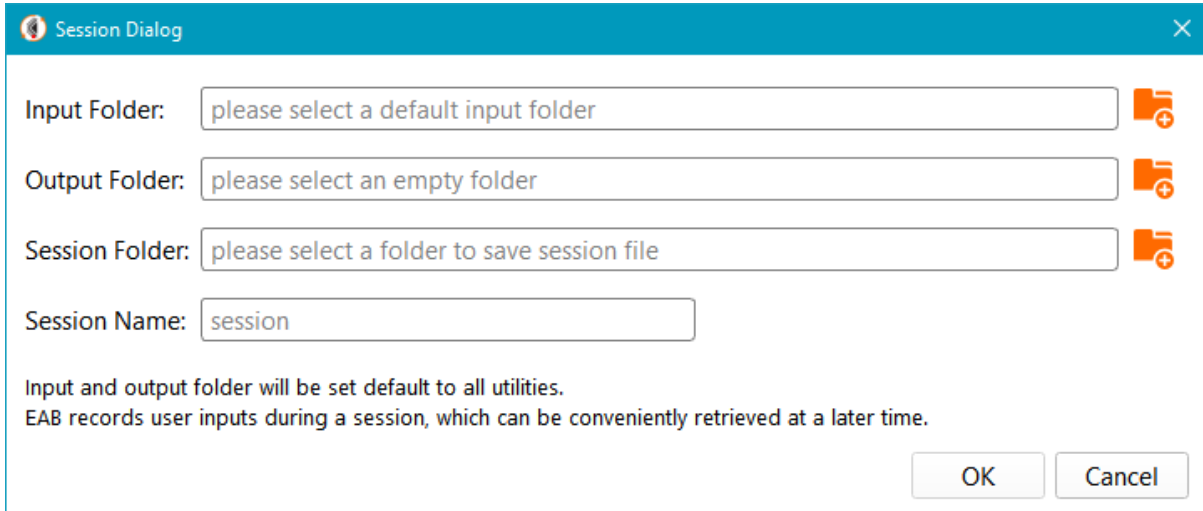


Figure 7 Session

Note: Upon starting EAB, a default session will be created. The user can then choose to either retain or discard it.

1. Start Session

Input and output folders will be set as specified in the session dialog below for all utilities of EAB. The output folder must be empty. A session should be named to find easily at the time of loading. Once a session has started, all inputs from users will be recorded and then saved to a file called *<session name>.eab*



The Session Dialog window contains the following fields and controls:

- Input Folder:** A text box with the placeholder "please select a default input folder" and a folder selection icon.
- Output Folder:** A text box with the placeholder "please select an empty folder" and a folder selection icon.
- Session Folder:** A text box with the placeholder "please select a folder to save session file" and a folder selection icon.
- Session Name:** A text box containing the text "session".
- Instructions:**
 - Input and output folder will be set default to all utilities.
 - EAB records user inputs during a session, which can be conveniently retrieved at a later time.
- Buttons:** "OK" and "Cancel" buttons at the bottom right.

Figure 8 Session Dialog

2. Load Session

Load a session that has previously been saved. Every user input will be filled in again.

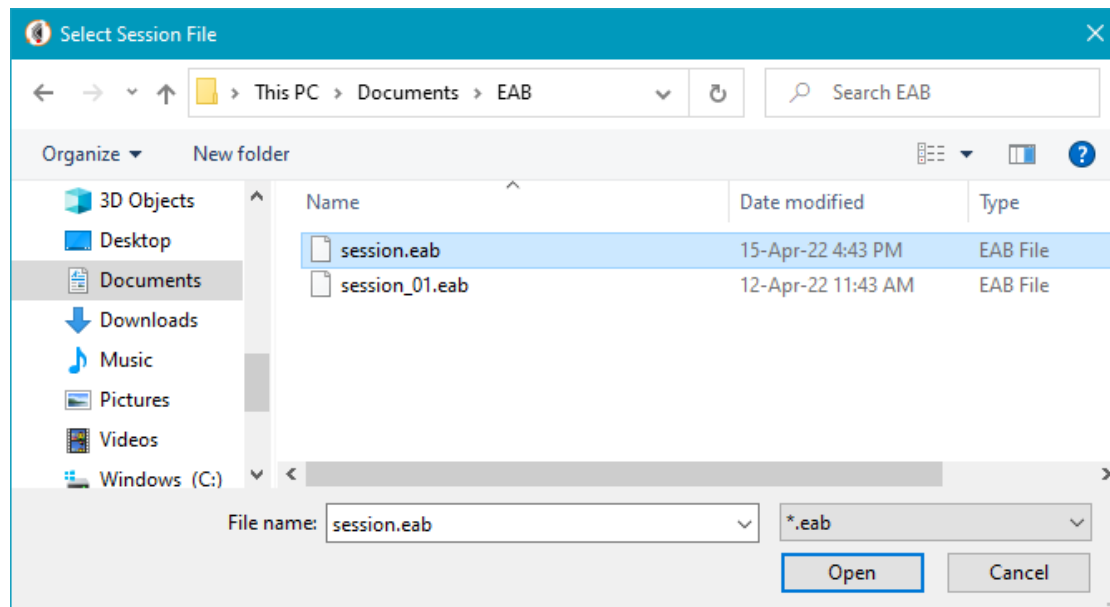


Figure 9 Select session file

3. Stop Session

All inputs from the user are saved to the session file. EAB will not collect user inputs from the time of stopping a session. The "Stop Session" button is enabled only when a session is started or loaded.

C. Command Prompt

Located at the left edge, the *Command Prompt* button will open a command window. The paths to *eab_tools.exe* and *ffmpeg.exe* are temporarily set to the **PATH** variable so that users can execute these tools without worrying about where they are located.

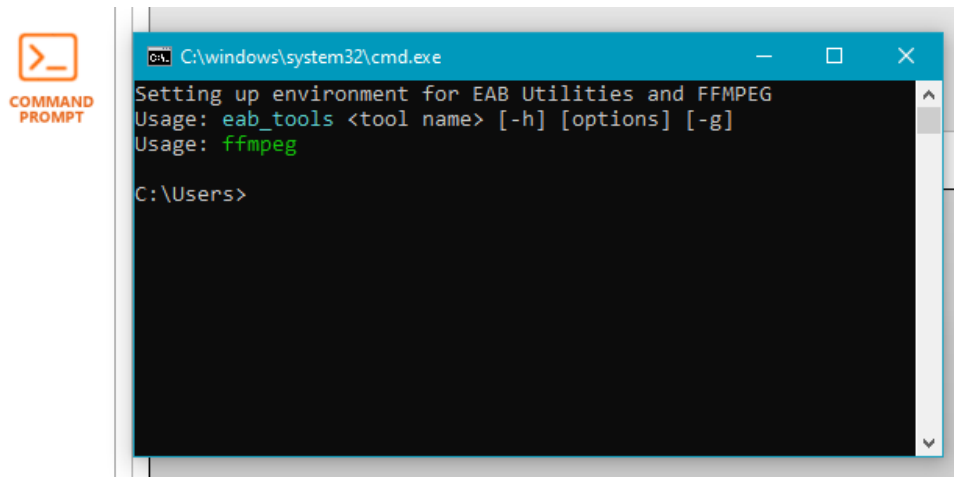


Figure 10 Command Prompt Window

Upon execution, every tool in the EVE Asset Builder (EAB) generates a command-line text in the Log Window. This text can be copied and pasted by users into the command window, allowing them to execute it from there.

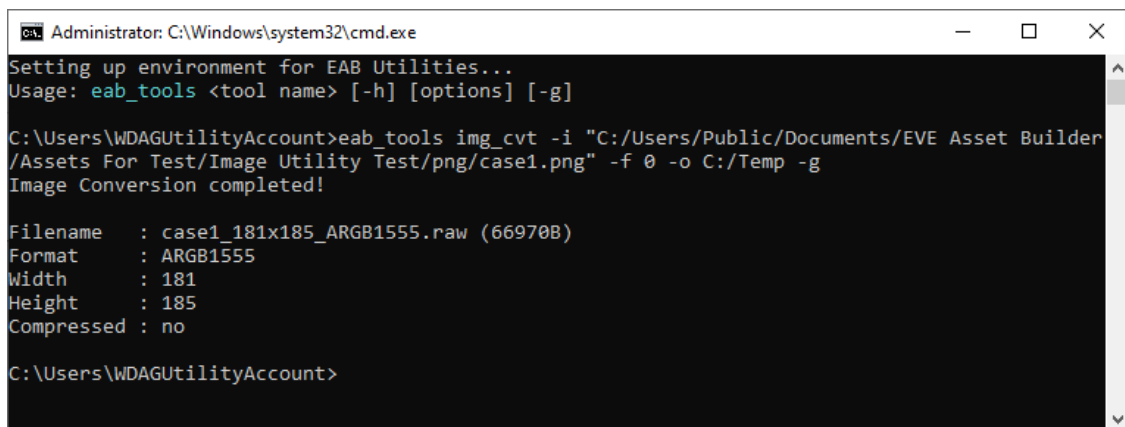


Figure 11 Example of converting an image

D. Log Window

The *Log Window* displays information related to inputs, outputs, warnings, or errors to assist users in effectively utilizing EAB tools.



Figure 12 Log Window

CLEAR: erase all content in Log Window

COPY: if there is any text selected, copy it to the clipboard. Otherwise, copy all the content.

Output Log Tab: show the log output

Command Line Tab: show the command-line texts which have been executed.

E. Naming Convention and Usage of Output Files

The output files generated by each tool in the EVE Asset Builder (EAB) adhere to a specific naming convention, as outlined in the following table. The convention and its intended use are described in detail. It should be noted that `<input>` refers to the base name of an input file, such as "lena" for an input file named "lena.png", while `<output>` is defined by the user.

Tool	Output Naming Convention	Example of input and output file	Usage
Image Converter	Uncompress: <code><input>_<width>x<height>_<output format>.<raw rawh></code>	lena.png -> lena_185x150_L8.raw, lena_185x150_L8.rawh	- Load the .raw or .rawh file to RAM_G - Decode it using cmd_setbitmap
	Compress: <code><input>_<width>x<height>_<output format>.<bin binh></code>	lena.png -> lena_185x150_L8.bin, lena_185x150_L8.binh	- Load the .bin or .binh file to RAM_CMD/Flash - Decompress it using cmd_inflate, cmd_inflate2 - Decode it using cmd_setbitmap
PNG/JPEG Validator	<code><input>.opt.<extension></code> Note: This tool may not generate any output file if an input image is already good for Eve	lena.png -> lena.opt.png	- Load the output file to RAM_G or command FIFO - Decode it using cmd_loadimage
JPEG Optimizer	<code><input>.opt.jpg</code>	lena.jpg -> lena.opt.jpg	- Load the output file to RAM_G or command FIFO - Decode it using cmd_loadimage
Video Converter	<code><input>.avi</code>	big_buck.avi -> big_buck.avi	- Load the output file to RAM_G (media FIFO) or command FIFO - Decode it using cmd_playvideo
Audio Converter	<code><input>.raw</code>	happy_summer.mp3 -> happy_summer.raw	- Load the output file to RAM_G (media FIFO) - Play it by controlling the appropriate registers

Font Converter	Legacy Format: <input>__ <bitmap format>. <raw rawh>	symbol.ttf -> symbol_25_L8.raw, symbol_25_L8.rawh	- Load the raw file to a predefined offset in RAM_G - Decode it using cmd_setfont, cmd_setfont2
	Extended Format: <input>__ <bitmap format>. <xfont glyph>	arial.ttf -> arial_39_ASTC.xfont, arial_39_ASTC.glyph	- Load .xfont file to RAM_G, .glyph file to RAM_G/FLASH - Decode them using cmd_setfont or cmd_setfont2
Animation Converter	FLASH: <input>.anim, <input>.anim.flash <u>Note:</u> .anim.flash is a flash image. It can be written to flash chip, then play.	bicycle.gif -> bicycle.anim, bicycle.anim.flash	- Use the .anim file to generate a flash image and take note of the offset address of .anim.object - Load the flash image to the flash chip - Use cmd_animstart, and cmd_animframe to play the animation, in which the flash address parameter is the offset noted in step 1 above
	RAM_G: <input>.anim, <input>.anim.ram_g	abstract.gif -> abstract.anim, abstract.anim.ram_g	- Load the .anim.ram_g file to RAM_G at the predefined offset - Use cmd_animstartram, and cmd_animframeram to play the animation, in which the address parameter is the offset of anim.object
Flash Image Generator	<output>.bin <output>.map <output>.edf	abstract.anim, lena.raw -> flash_16MB.bin,	- Load the flash .bin file to flash chip - See the offset of each asset in the map or the edf file
		flash_16MB.map	Each line contains the asset information in the format: <i>name:offset:length</i>
		flash_16MB.edf	Refer to section EDF Block
Asset Compressor	Zlib: <output>.zlib	arial.xfont, big_buck.avi -> inflate.zlib	- Load compressed file to flash or RAM_G
	Zopfli: <output>.zopfli	arial.xfont, big_buck.avi -> inflate.zopfli	- Use cmd_inflate, cmd_inflate2 to decompress the assets
Custom Touch Compiler	<input>.load.bin, <input>.load.h	firmware.c -> firmware.load.bin, firmware.load.h	- Write .load.bin file or .load.h file to RAM_CMD - Update the write pointer REG_CMD_WRITE - Wait till Eve completes all the commands
Bin2C Converter	<input>.c	img_argb2.bin -> img_argb2.c	Include the C file to use the converted C-array
Disassembler	<input>.da.txt	bt817_dump.bin -> bt817_dump.da.txt	Open .da.txt file to see the human-readable Eve commands

Table 1 Naming Convention and usage of output files

F. Image Utilities

Within this tab, there are three sub-tab windows that offer a range of image-related functionalities. The first sub-tab is dedicated to the conversion of PNG, JPEG, and BMP images to the EVE-compatible image format. The second sub-tab is designed for the purpose of assessing the compatibility of PNG and JPG files with CMD_LOADIMAGE. The third sub-tab hosts a JPEG optimizer that allows users to decrease the size of input JPEG files while maintaining their EVE-compatibility.

1. Image Converter

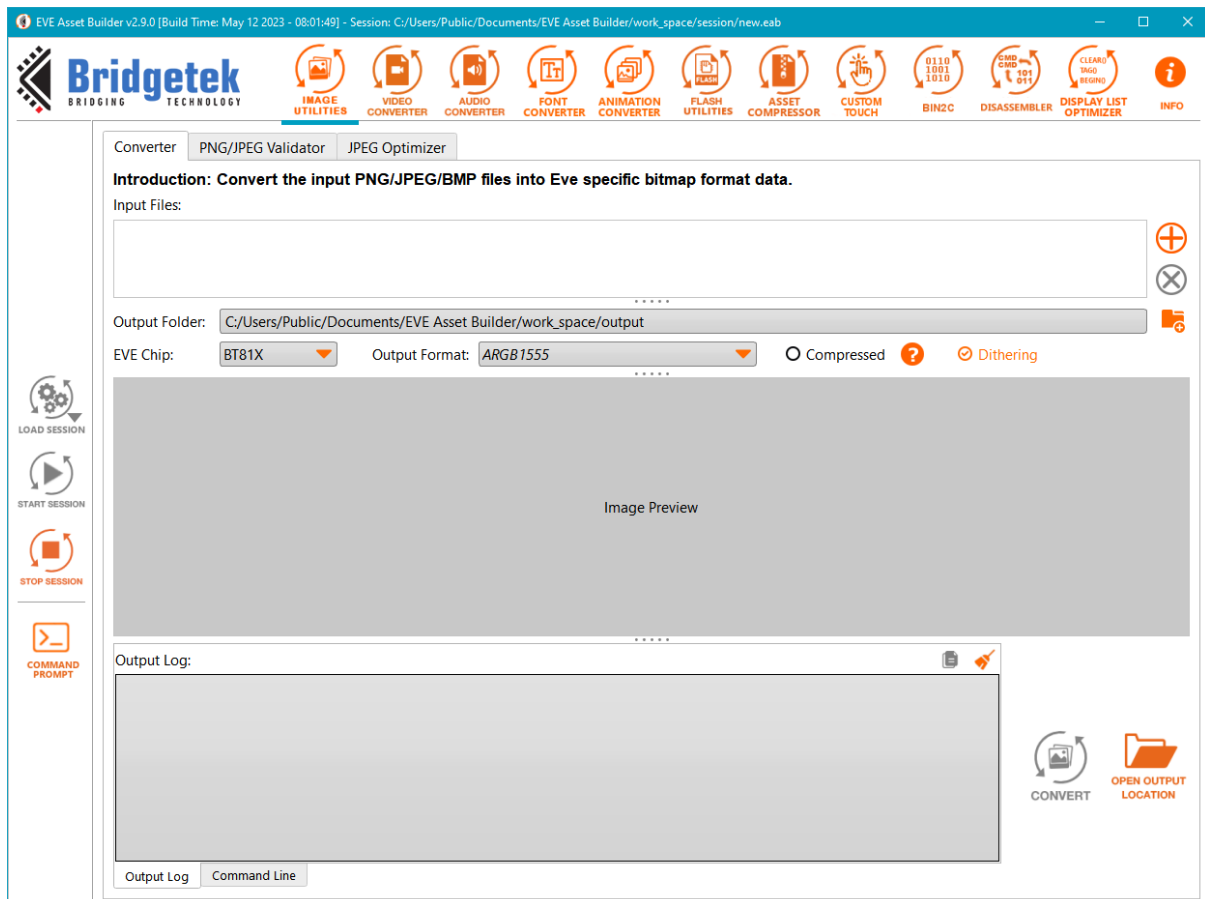


Figure 13 Image Converter tab

Input Files: PNG/JPG/BMP files to be converted. Users can change the order of images by using *Ctrl+Up* or *Ctrl+Down*.

Output Folder: The folder contains converted files.

EVE Chip: Refer to [EVE Chip Series](#)

Compressed: Deflate the output file, which file name is *.bin other than *.raw. It should be decompressed by cmd_inflate or cmd_inflate2 before use.

Output Format: The output format of images, is one of the following:

- ARGB1555, RGB565, ARGB4
- RGB232, ARGB2,
- L1, L2, L4, L8,
- PALETED565, PALETED4444, PALETED8

- ASTC formats
- DXT1

ASTC Preset: Only available for BT81X. Available options are *veryfast*, *fast*, *medium*, *thorough*, and *exhaustive*. Quality increases from *veryfast* to *exhaustive* while encoding speed decreases as well.

ASTC Options: More options for ASTC.

Multi-Cell Bitmap: Add padding data to ASTC-based multi-cell bitmap. Each cell needs 64-byte aligned in memory so that the EVE chip can render them correctly.

2. PNG/JPEG Validator

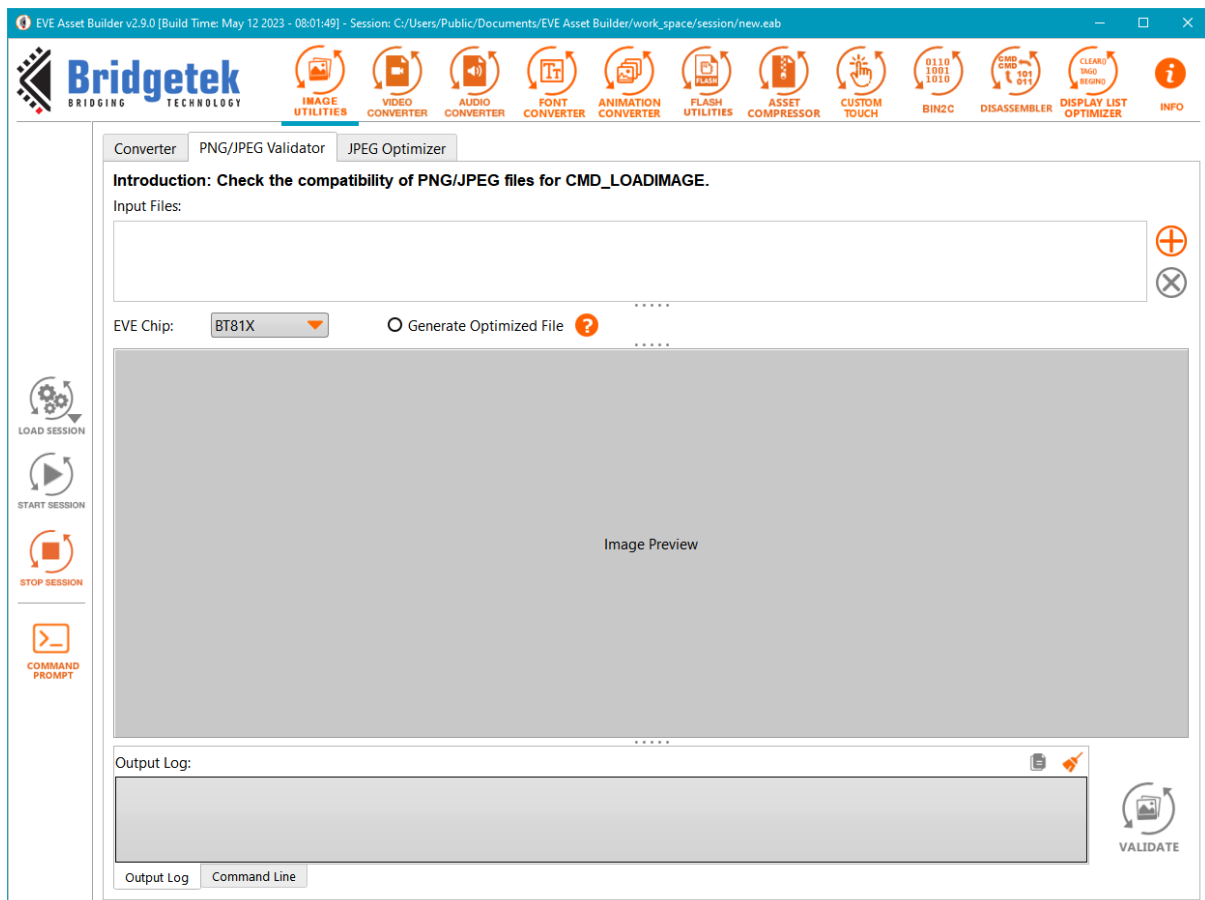


Figure 14 PNG/JPEG Validator tab

EVE Chip: Refer to [EVE Chip Series](#)

Generate Optimized File:

- If checked, input images are optimized and saved to the output folder. Then optimized images are validated.
- If unchecked, the original input images will be validated.

3. JPEG Optimizer

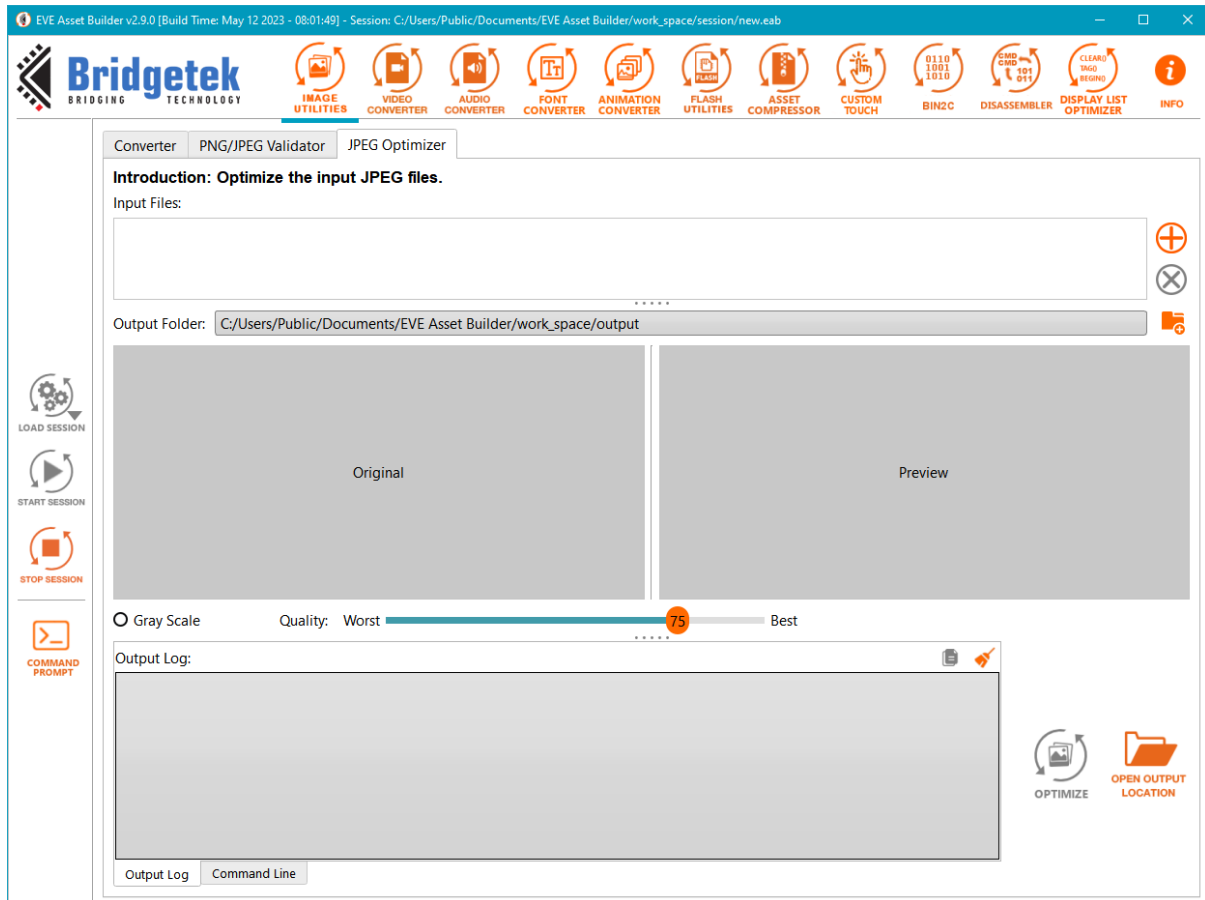


Figure 15 JPEG Optimizer tab

Gray Scale: Convert image to “L” format.

Quality: From 0-95. Reducing image size significantly also results in the worst quality.

G. Video Converter

Convert video file into **MJPEG** format and mono audio channel. Output is an AVI file and a sample project to guide how to playback video on EVE device.

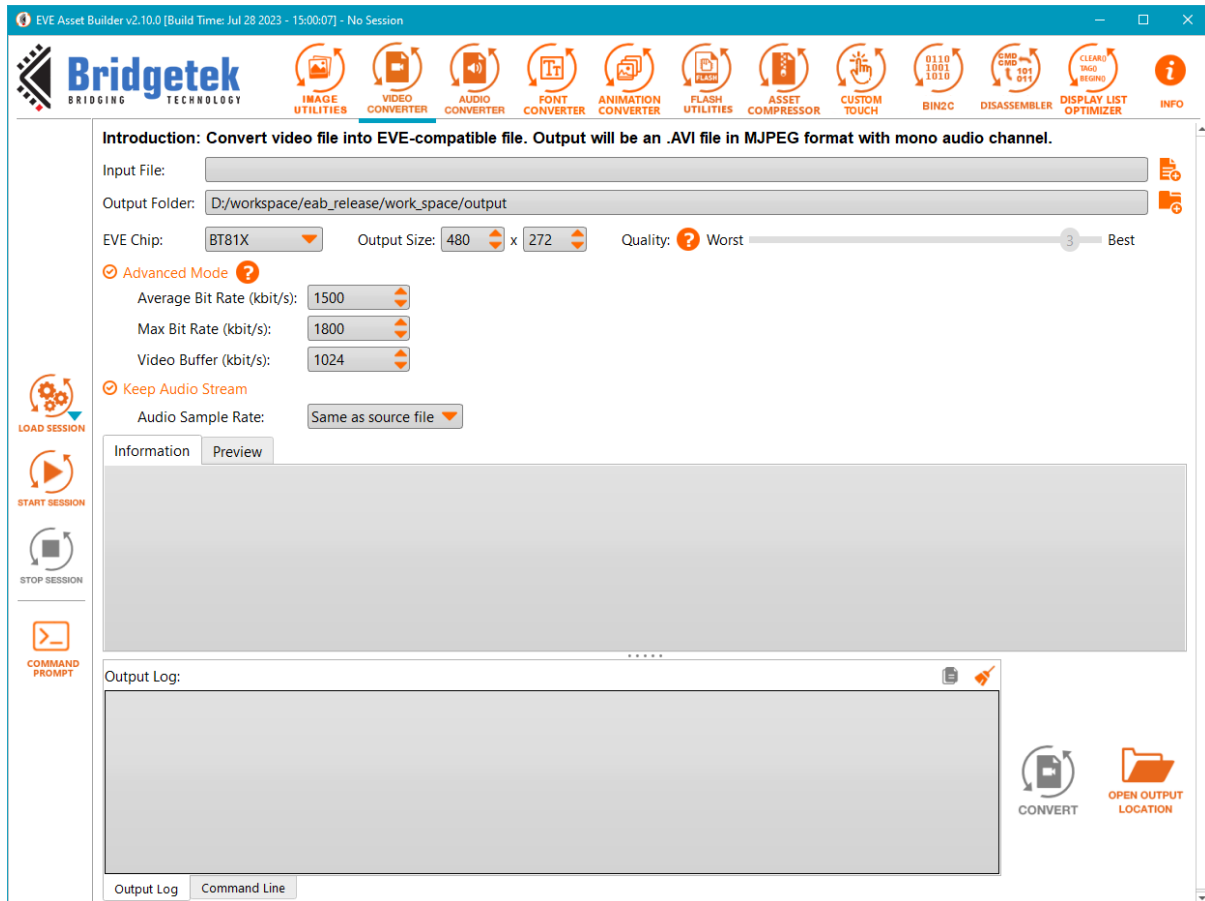


Figure 16 Video Converter tab

Input: The original video will be converted.

Output Folder: The folder contains converted files.

EVE Chip: To show warning if the output resolution reaches the limit. Refer to [EVE Chip Series](#)

Size: Must be specified, the video will be converted to this size.

Quality: From 1-31. 1 is best and 31 is the worst.

Advanced Mode: Let user control output bitrate. *Video Quality* will be ignored if this mode is turned on.

- ❖ **Average Bit Rate (kbit/s):** Set the average bit rate for the output video
- ❖ **Max Bit Rate (kbit/s):** Set max bit rate for the output video
- ❖ **Video Buffer (kbit/s):** Set buffer size for the output video

Keep Audio Stream: Users can keep or remove the audio stream

Audio Sample Rate: Select the sample rate for the output audio stream, which can be the same as the input video (default) or any value from the list. They are: 8000Hz, 11025Hz, 16000 Hz, 22050 Hz, 32000 Hz, 32780 Hz, 44056 Hz, 44100 Hz, and 48000 Hz.

H. Audio Converter

Convert WAV/MP3 file into EVE-compatible file.

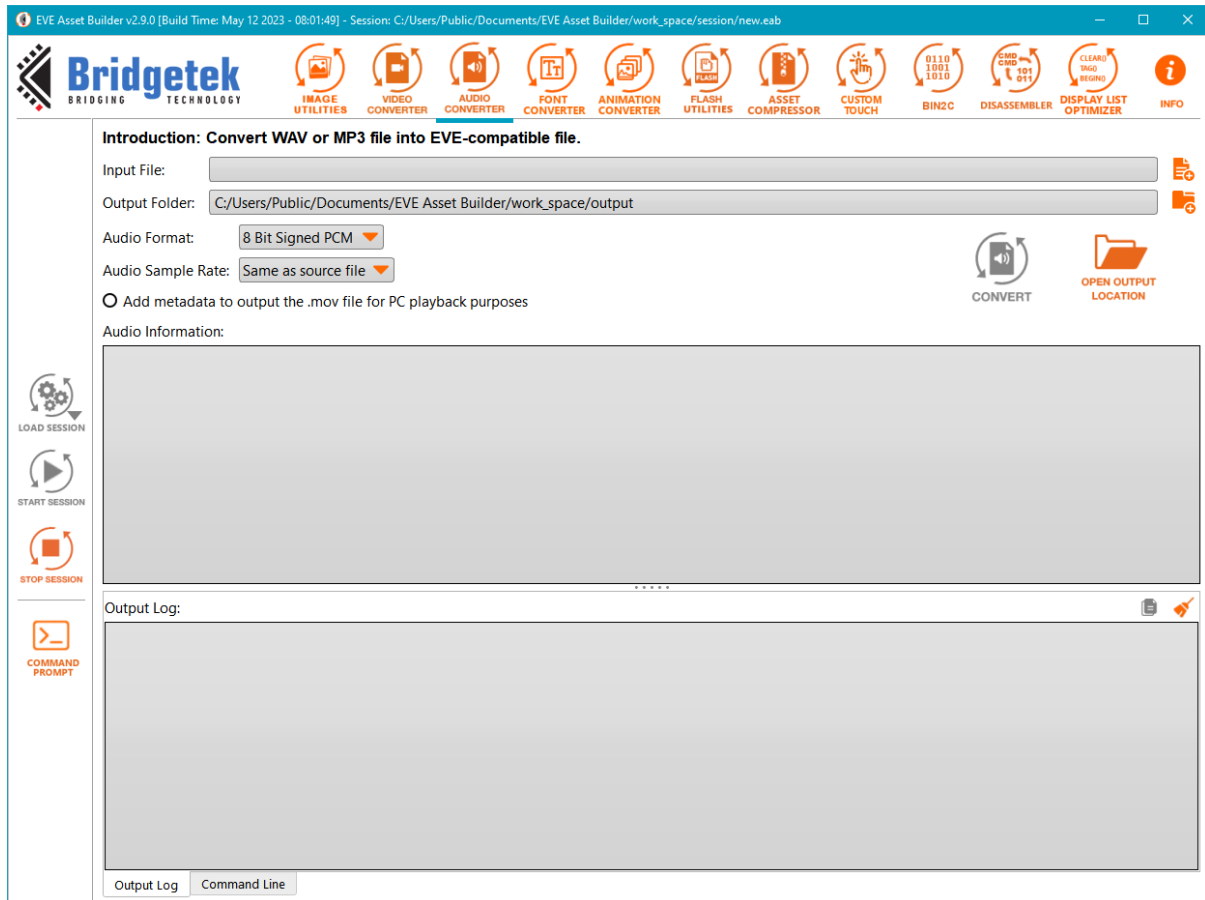


Figure 17 Audio Converter tab

Input File: The audio file will be converted.

Output folder: The folder contains converted files.

Audio Format: Audio format of the output

- 8-bit signed PCM
- 8-bit u-Law
- 4-bit IMA ADPCM

Audio Sample Rate: Users can keep the same sample rate as the input stream or select any value from the drop-down list.

Add metadata to output the .mov file for PC playback purposes: The converted file needs a header to be played back on PC. But the header will make it unplayable on Eve devices.

I. Font Converter

Extract characters from the OpenType or TrueType font file into a specific font metric block as well as the raw bitmap data. Unprintable characters will be filtered out.

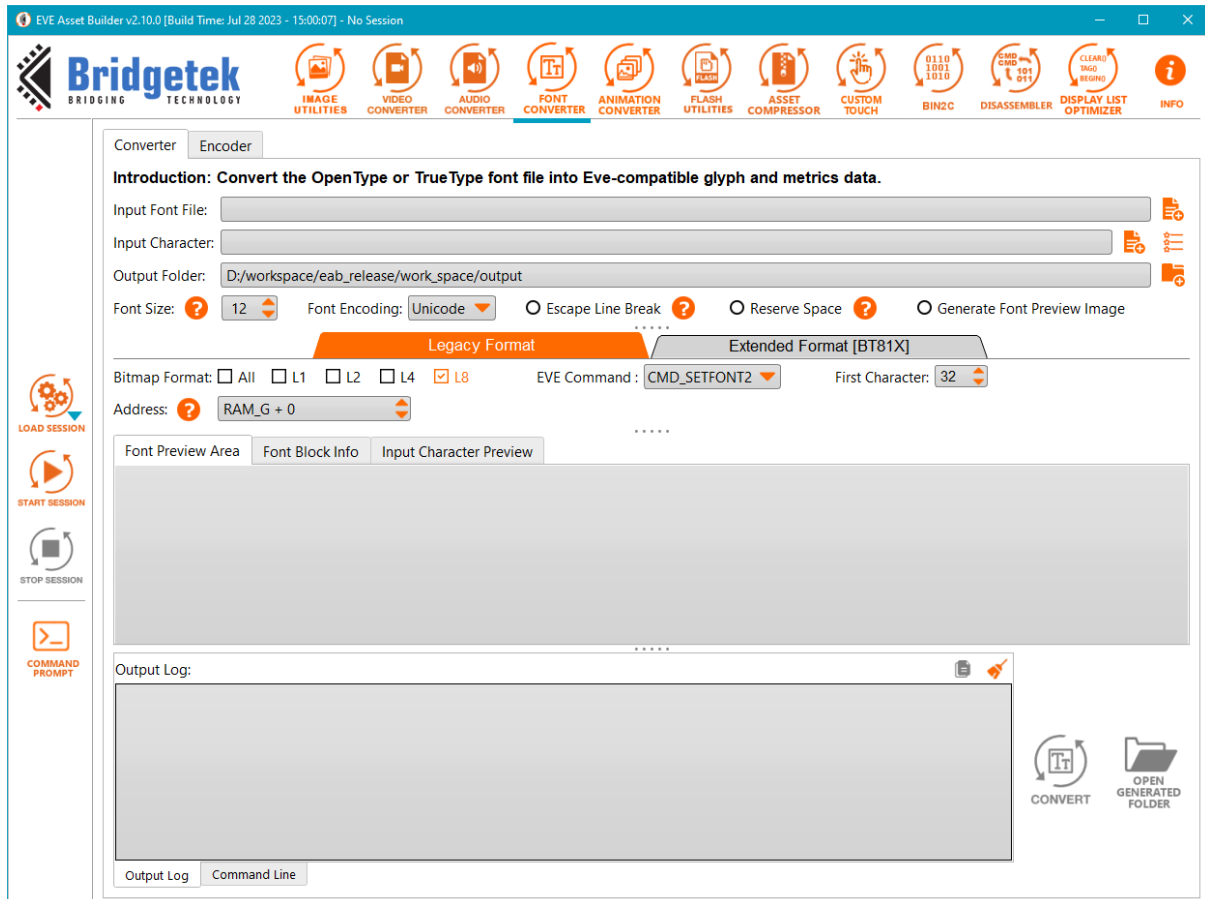


Figure 18 Font Converter tab

Input Font File: Font file to be converted.

Input Character: Choose the character set that will be converted.

Output Folder: The folder contains converted files.

Font Size: Any character whose width exceeds 255 pixels will be removed.

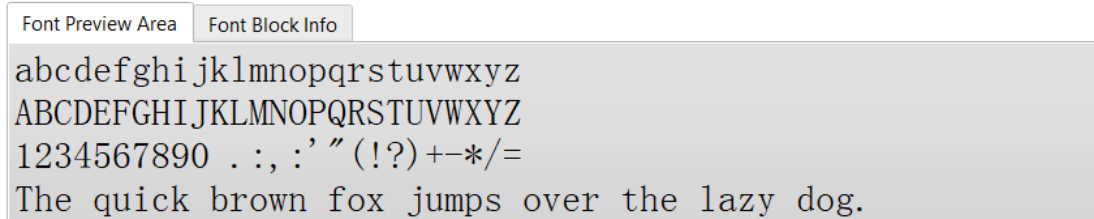
Font Encoding: To ensure proper conversion, the user must select an encoding that is compatible with the input font. Currently, EAB offers support for two encodings: Unicode and Symbol.

Escape Line Break: BT817/8 chip support to overwrite line break. The ASCII 0x0A is interpreted as the index, other than the line break.

Reserve Space: Allocate the code point \x20 for the space character, in order for cmd_fillwidth to function properly.

Generate Font Preview Image: These images help users check if the characters are rendered as expected

Font Preview Area: Display sample characters by using the selected font.

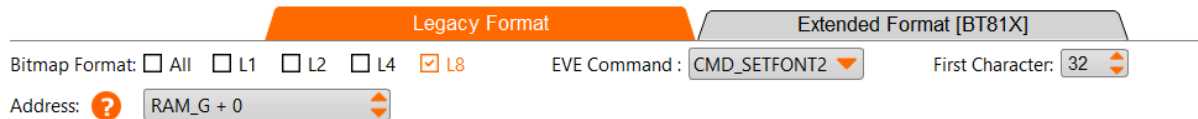

Figure 19 Font Preview Area
Font Block Info:

- Show the number of glyphs that are valid for the selected font.
- Show a statistic of the selected font, including Block Name, Unicode Range, and Valid Glyphs/All Glyphs.

Font Preview Area		Font Block Info
Number Of Glyph: 21982		
Font Block Name	Unicode Range	Valid Glyphs/All Glyphs
Basic Latin	U+0000-U+007F	95/128
Latin-1 Supplement	U+0080-U+00FF	21/128
Latin Extended-A	U+0100-U+017F	8/128
Latin Extended-B	U+0180-U+024F	8/208
IPA Extensions	U+0250-U+02AF	2/96

Figure 20 Font Block Info

1. Legacy Format


Figure 21 Legacy Format

Bitmap Format: Any combination of bitmap types L1, L2, L4, L8

EVE Command: Select CMD_SETFONT or CMD_SETFONT2. Converted files will be compatible with the chosen command.

Address: The converted font will be loaded to the offset address in RAM_G, which must be divisible by 4.

First Characters: Set the first index for converted characters. The valid range is 1-127. Apply to CMD_SETFONT2 only.

Output:

- The output consists of a 148-byte metric block along with raw bitmap data.
- In addition, this tool generates example C code to demonstrate usage.
- The output data is formatted for a single bitmap handle.

2. Extended Format [BT81X]

Handle fonts with a full range of Unicode code points. This feature only supports the BT81X EVE series.

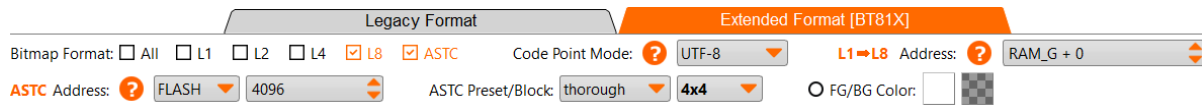


Figure 22 Extended Format

Bitmap Format: Any combination of bitmap types L1, L2, L4, L8 and ASTC

Code Point Mode:

- UTF-8: Addressing the characters with UTF-8 code point.
- Ordinal: Addressing the characters with index 0, 1, 2, ..., n

Address: Address of glyph data. For ASTC, the location can be RAM_G or FLASH. For L1->L8, the location must be in RAM_G.

Note: Due to the size of RAM_G, the maximum number of input characters will be reduced.

- **ASTC:** 8192 characters
- **L8:** 8192 characters
- **L4:** L8 * 2
- **L2:** L8 * 4
- **L1:** L8 * 8

ASTC Preset: veryfast, fast, medium, thorough, and exhaustive. The encoding speed decreases while the quality increases from 'veryfast' to 'exhaustive'.

ASTC Block: 4x4, 5x4, 5x5, 6x5, 6x6, 8x5, 8x6, 8x8, 10x5, 10x6, 10x8, 10x10, 12x10, 12x12

Note: For *auto* option, block footprint depends on font size:

- If font size > 52: block footprint = 10x8
- If font size < 19: block footprint = 8x5
- Otherwise: block footprint = 8x8

FG/BG Color: set foreground/background color for the converted characters.

Output:. **glyph** file contains graphic data and. **xfont** file contains the font metrics block, as well as the sample C code demonstrating the usage.

Limitation: For **CMD_BUTTON** and its relatives, the converted characters are not vertically aligned. This is because the pixel height calculation accommodates all input characters. Thus, it is recommended to employ a limited set of characters for **CMD_BUTTON** text to achieve superior vertical alignment. A potential solution would be to convert only the appropriate number of characters utilized in **CMD_BUTTON** to ensure that the pixel height aligns accordingly.

3. Font Encoder

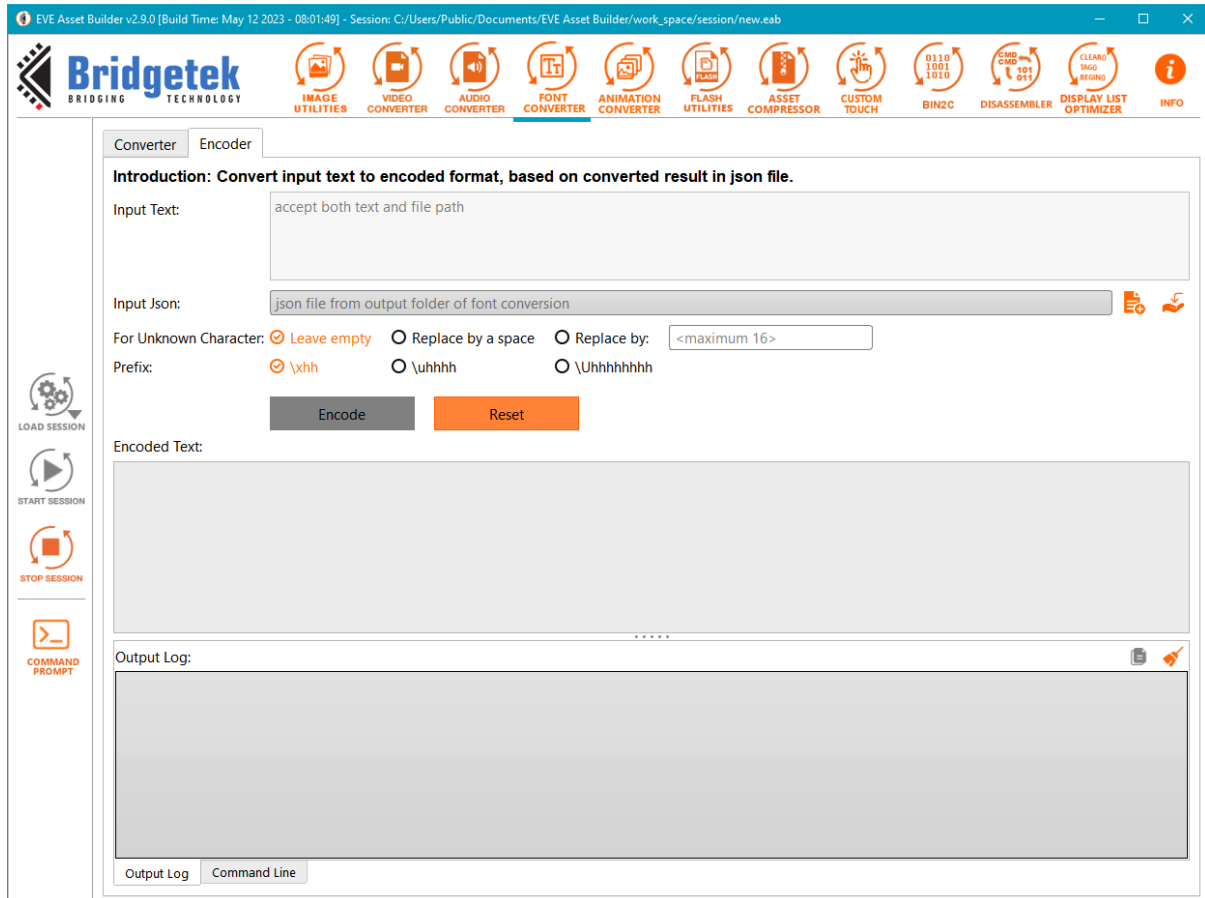


Figure 23 Encoder tab

Input Text: Set text to encode.

Input Json: Set the .json file which is the output of a font conversion.

For Unknown Character: Set the option to process with unknown characters.

Prefix: Set the prefix of encoded output

J. Animation Converter

Convert a **GIF** file or a series of **PNG/JPEG/BMP** files into EVE-compatible animation files. Animation is supported by the BT81X chip and above.

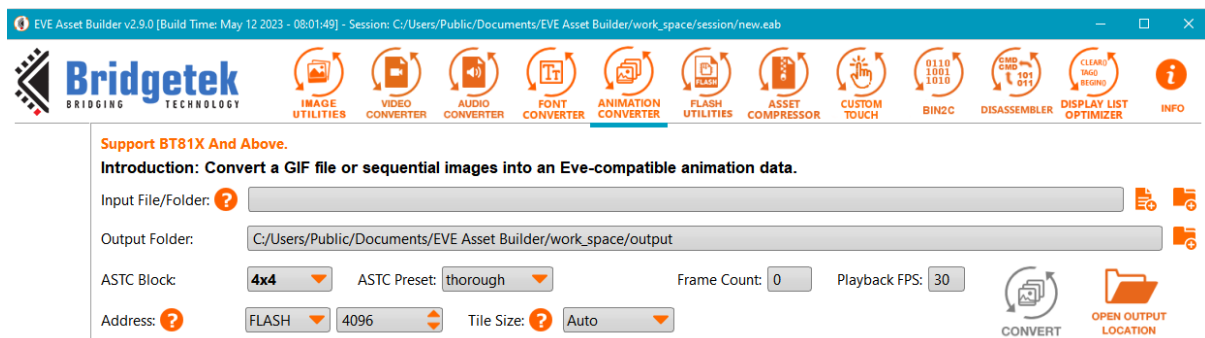


Figure 24 Animation Converter

Input File/Folder: Users can select a **GIF** file or image folder. Image folder must contain PNG/JPEG/BMP files which have the sequential name as a regular expression `".*[0-9]+\.(png|jpeg|jpg|bmp)"` defined. The number denotes the sequence of animated frames. For example: "001.png," "002.png."

Output Folder: The folder contains the converted files.

ASTC Block: Select one of the following: 4x4, 5x4, 5x5, 6x5, 6x6, 8x5, 8x6, 8x8, 10x5, 10x6, 10x8, 10x10, 12x10, 12x12

ASTC Preset: veryfast, fast, medium, thorough, and exhaustive. The quality increases from very fast to exhaustive, while the encoding speed decreases as well.

Frame Count: Display the number of available frames.

Playback FPS: For previewing the speed of animation

Address: Select **FLASH** or **RAM_G**. Address must be a multiple of 64.

Tile Size: Tile is the unit to determine the overlap area across frames. The valid options are: Auto, Manual or Disabled

- Auto: EAB will select the best tile size automatically
- Manual: user selects *Tile Width* and *Tile Height*

Note: Result of *Tile Width* x *Tile Height* must be a multiple of 4, due to the constraint of rendering ASTC image from flash.

- Disabled: Do not apply Tile to the conversion process

Output

- A *SampleApp* folder illustrates how to showcase an animation.
- An *.anim* file that is employed to generate a flash image file, enabling the rendering of animations through flash.
- With BT817/8 onwards, the utilization of **RAM_G** can be leveraged to improve the animation rendering speed. To achieve this, an *"anim.ram_g"* file is generated.
- An *.anim.flash* file that can be directly written to flash and then used to render animations.
- An *.anim.json* that contains the conversion information.

K. Flash Utilities

1. Flash Image Generator

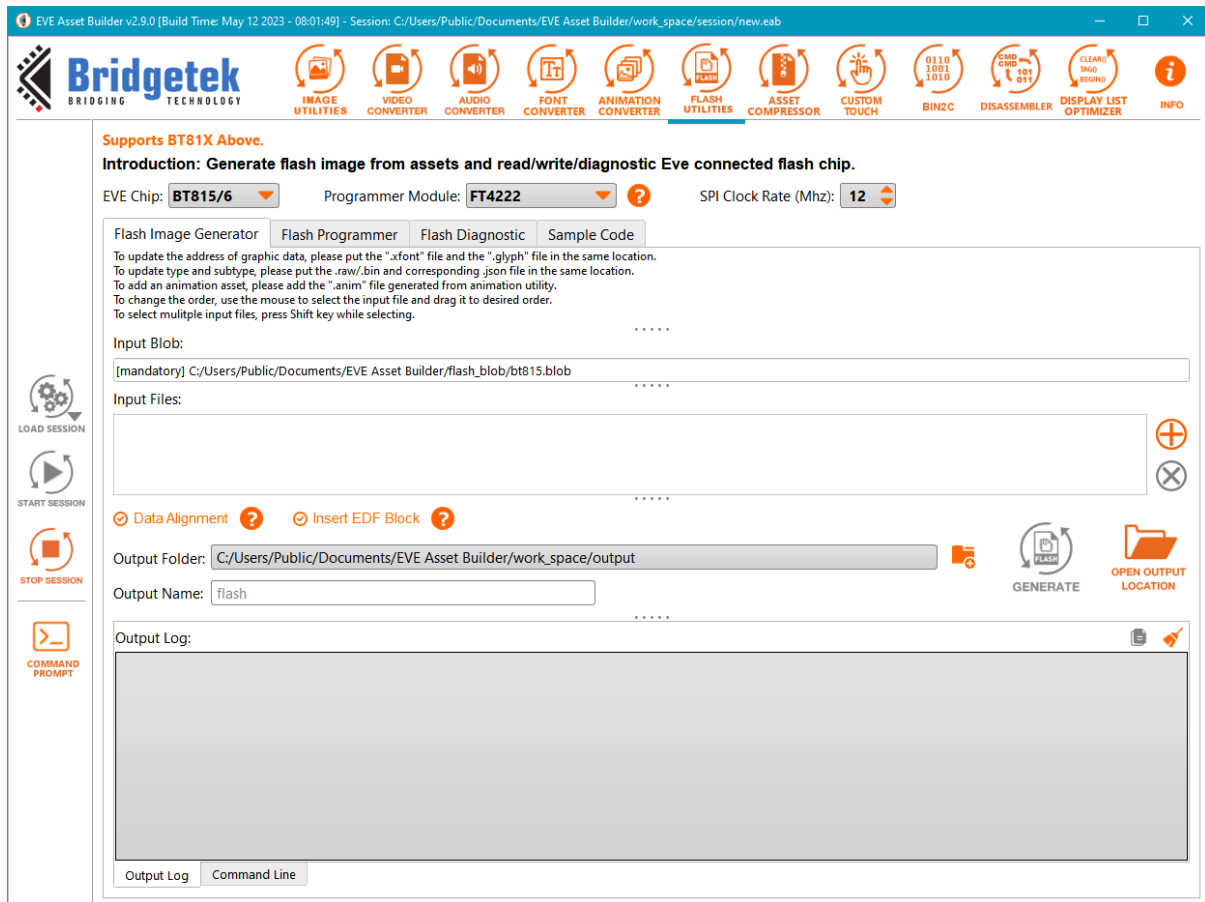


Figure 25 Generate Flash Image

Input Blob: The flash driver used by firmware, always on the first block (4096 bytes).

Input Files: Select the files of flash content.

EVE Chip: Select **BT815/6** or **BT817/8**, which leads to the respective blob file in the flash image.

Data Alignment: Each asset is 64-byte aligned, and the whole flash file is 256-byte aligned. The alignment will not be applied if unchecked.

Insert EDF Block: EDF block will be generated and inserted after the blob driver.

Output Folder: Folder containing the generated files.

Output Name: The generated flash image file name. Default is "flash".

Output: A .bin file to write to the flash chip,
 A .map file keeps the address and size of each asset in the flash as the format:
<asset name> : <address> : <size>,
 An .edf file has the details of each assets, as described in [EDF Block](#).

Note:

- To update the address of the graphic data in the xfont file, users should place it in the same folder as the glyph file.
- To include xfont in the flash bin, the user should place xfont after the corresponding glyph.
- To update the type and subtype of a raw/bin file, users should place the json file that corresponds to the raw/bin file in the same folder.

EDF Block:

The "Eve Flash Description File" (EDF) contains asset data in sequential order and is located immediately after the **blob** driver. In order to obtain detailed information about the asset, a '.json' file is created with the same name as the output flash image file, and it is placed in the same directory.

EDF structure is defined as below:

```
struct EDF {
    uint32_t block_length;           // the length of this EDF
    Eve_Flash_Asset_Info assets[0...n-1]; // the information of n assets
};

struct Eve_Flash_Asset_Info {
    uint16_t assetID;               // the sequential ID of the asset
    uint32_t startAddress;          // the offset of the asset
    uint32_t size;                  // the size of the asset, in byte
    uint8_t compressionMethod;     // 0 = no compression, 1 = compressed
    uint8_t type;                  // the type of the asset, see below
    uint16_t subType;              // the subtype of the asset, see below
    uint16_t width;                // the width of bitmap/video/animation
    uint16_t height;               // the height of bitmap/video/animation
};
```

assetID denotes the order of an asset in flash image

width and **height** are only meaningful for bitmap, video, and animation. For other types of assets, they will be set to zero.

type and **subType** are defined in the following tables

Type	Subtype	Description
0x01	-	Flash Driver (Blob)
0x02	See Table 3 subType of Bitmap	Bitmap
0x03	-	Bitmap PALETTED
0x04	-	Font Glyph (ASTC)
0x05	-	Extended Font
0x06	-	Legacy Font
0x07	-	Animation Data
0x08	-	Animation Object
0x09	-	PNG/JPEG File
0x0A	-	DXT1 File
0x0B	See Table 4 subType of Audio	Audio File
0x0C	-	Video file
0xFC	-	Padding Data
0xFD	-	EDF Block
0xFE	-	User Data

Table 2 Asset Type

SubType of Bitmap	Description
0	ARGB1555
1	L1
2	L4
3	L8
4	RGB332
5	ARGB2
6	ARGB4
7	RGB565
9	TEXT8X8
10	TEXTVGA
11	BARGRAPH
14	PALETTED565
15	PALETTED4444
16	PALETTED8
17	L2
37808	COMPRESSED_RGBA_ASTC_4x4_KHR
37809	COMPRESSED_RGBA_ASTC_5x4_KHR
37810	COMPRESSED_RGBA_ASTC_5x5_KHR
37811	COMPRESSED_RGBA_ASTC_6x5_KHR
37812	COMPRESSED_RGBA_ASTC_6x6_KHR
37813	COMPRESSED_RGBA_ASTC_8x5_KHR
37814	COMPRESSED_RGBA_ASTC_8x6_KHR
37815	COMPRESSED_RGBA_ASTC_8x8_KHR
37816	COMPRESSED_RGBA_ASTC_10x5_KHR
37817	COMPRESSED_RGBA_ASTC_10x6_KHR
37818	COMPRESSED_RGBA_ASTC_10x8_KHR
37819	COMPRESSED_RGBA_ASTC_10x10_KHR
37820	COMPRESSED_RGBA_ASTC_12x10_KHR
37821	COMPRESSED_RGBA_ASTC_12x12_KHR

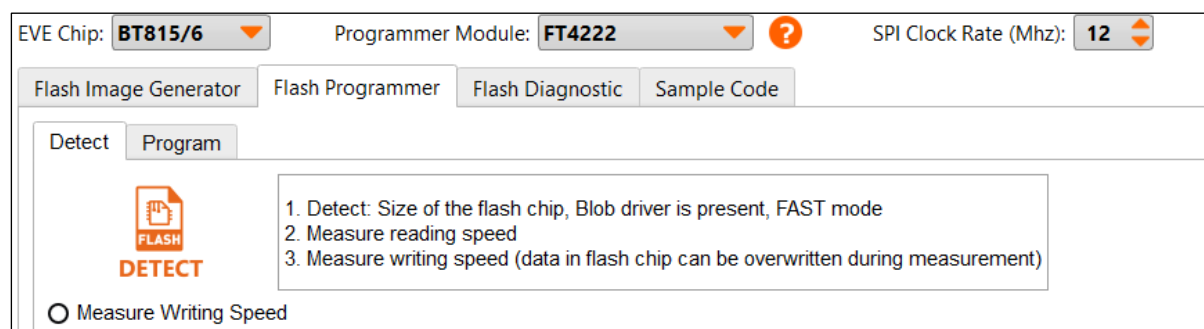
Table 3 subType of Bitmap

SubType of Audio	Description
0	Linear Sample format
1	uLaw Sample format
2	4-bit IMA ADPCM Sample format

Table 4 subType of Audio

2. Flash Programmer

Detect Flash


Figure 26 Detect Flash

EVE Chip: Select BT815/6 or BT817/8 based on the board.

Programmer Module: can be **FT4222**, **MPSSE**, or **Raspberry Pi Pico**.

Make sure the selected module is present on your device and all required drivers are installed. If your device has no module present, you may need to purchase these modules and connect them with your device properly.

Limitation: At any given time, only a single Programmer Module can be utilized.

SPI Clock Rate (Mhz): set SPI clock rate that is used to communicate from programmer module to Eve chip

Detect: Detect flash information.

Measure Writing Speed: The process will involve the writing of dummy bytes to the flash chip, which will result in the overwriting of any pre-existing data.

Program Flash



Figure 27 Program Flash

Start Address: Must be 64-byte aligned and within the range 0-256 MBytes

Binary File: Select file to write to flash chip or to verify flash content.

ERASE: Erase all data in the flash chip.

UPDATE: Write to the flash chip, erasing if necessary. Flash is not cleared out completely, only updates the required partitions.

UPDATE & VERIFY: Same as Update, then compare the updated data to source data.

VERIFY: compare the input binary file with flash content.

PROGRAM: Write into factory state flash chip. It is two or three times faster because it does not compare data before writing. This feature is supported for BT817/8 and above.

Read Flash

Read all content of flash chip into .bin file.

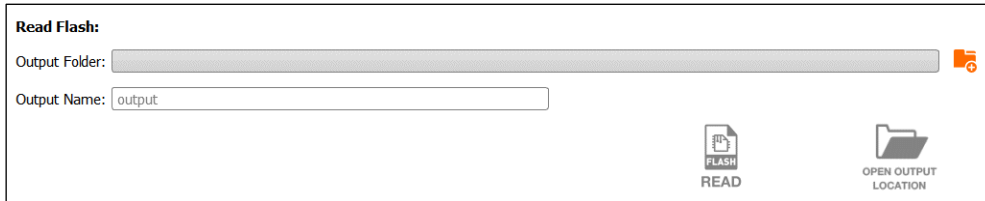


Figure 28 Read Flash

Output Folder: Select a folder to save the output file

Output Name: Select the name for the output file

READ: Start reading all data in the flash chip to the output file

Note: The whole flash chip is read out. As a result, the output file size is the same as the flash chip size.

INSTALL BLOB (for advanced users)

This feature is hidden by default. The following steps below will enable it:

1. Go to the EAB installation folder
2. Open file *ui_config.json*
3. Change the value of "install_blob" from **false** to **true**, then save it
4. Reopen EAB

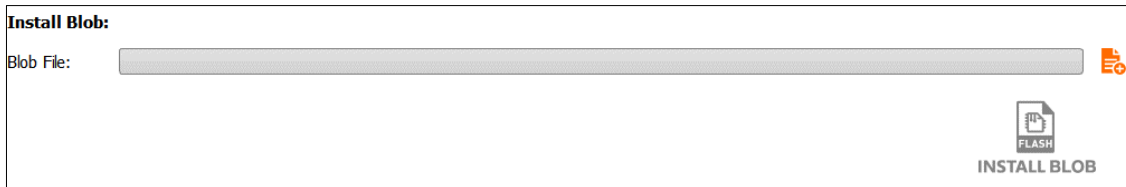


Figure 29 Enable "Install Blob"

Blob File: Select .blob file to install.

INSTALL BLOB: Start installing the blob driver to the flash chip

Programming Flash Using Raspberry Pi Pico

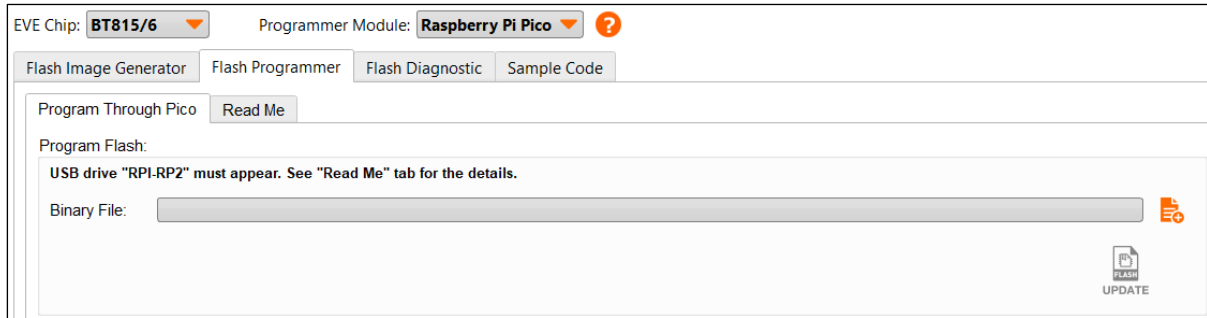


Figure 30 Programming flash using Raspberry Pi Pico

Binary File: select the file to write to the flash chip.

UPDATE: write .bin into the flash chip, erasing if necessary. Flash is not cleared out completely, only update the partition which is being updated.

Read Me Tab: show the steps to program EVE Flash through Pico.

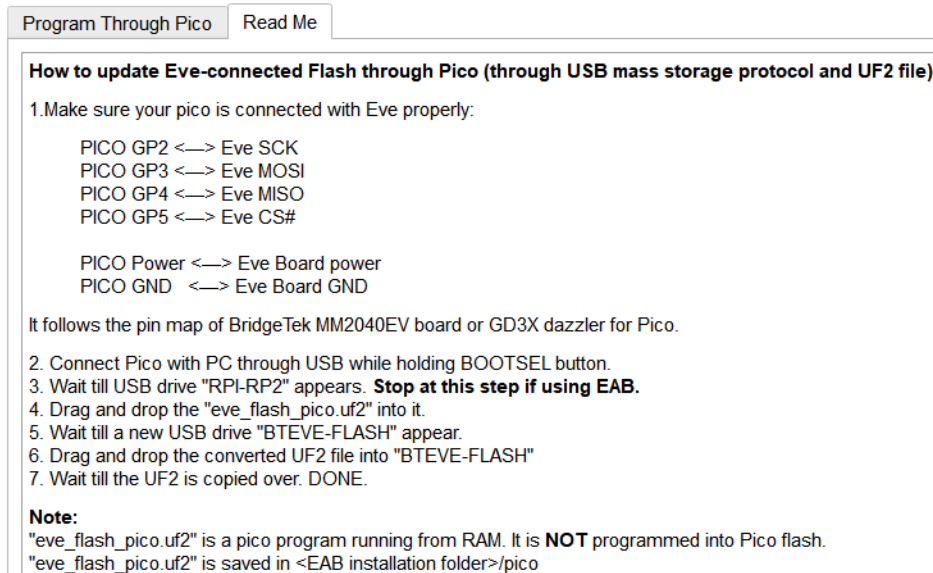


Figure 31 Raspberry Pi Pico Readme

Convert Binary File To Uf2 Format




Figure 32 Convert binary file to Uf2 format

Binary File: select a binary file to convert

Uf2 File: choose a file path to save the converted output

CONVERT: start the converting process

3. Flash Diagnostic

The purpose of this feature is to troubleshoot the flash chip. The "Raspberry Pi Pico" cannot currently be used for diagnosis, so you have to use another programmer module like FT4222 or MPSSE.

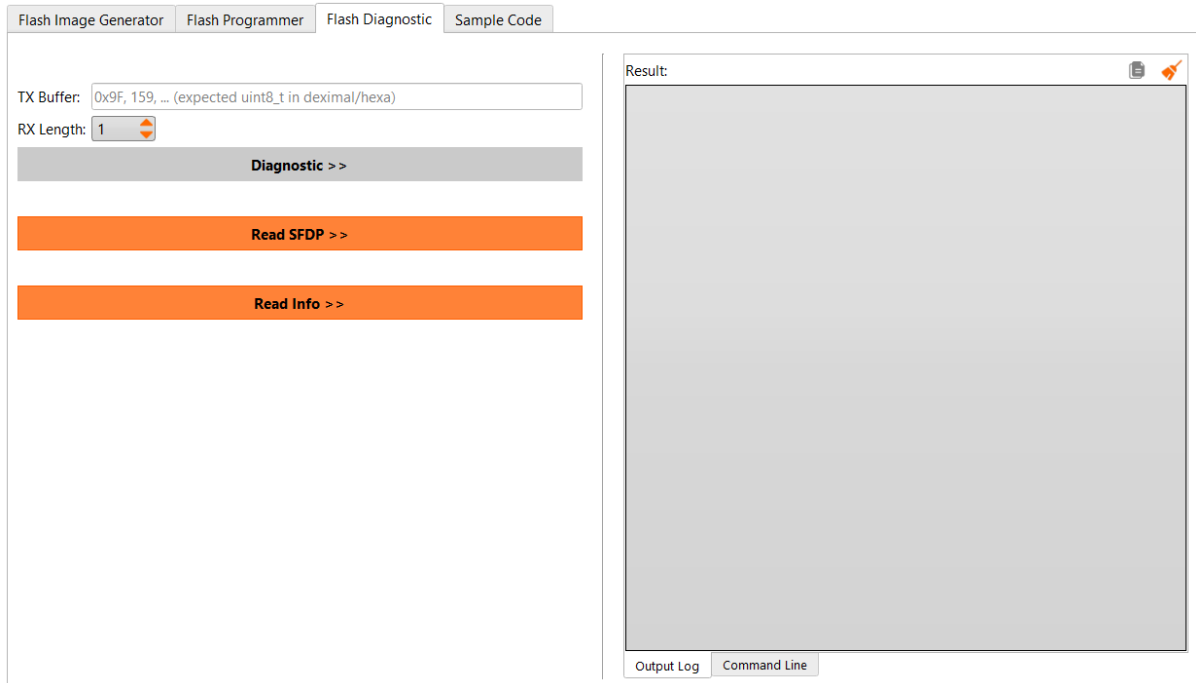


Figure 33 Flash Diagnosis

Diagnostic

TX Buffer: a buffer that will be sent to flash chip, with commas between each byte

RX Length: enter the length that you expect to receive from the flash chip

Note: for the TX and RX commands, please consult the manual of the flash chip that you are using.

Read SFDP

Read SFDP structure from flash chip

Read Info

This will report:

1. Flash status
2. Flash size (in Mbytes and Mbits)
3. DTR(double transfer rate) is supported or not

4. Sample Code

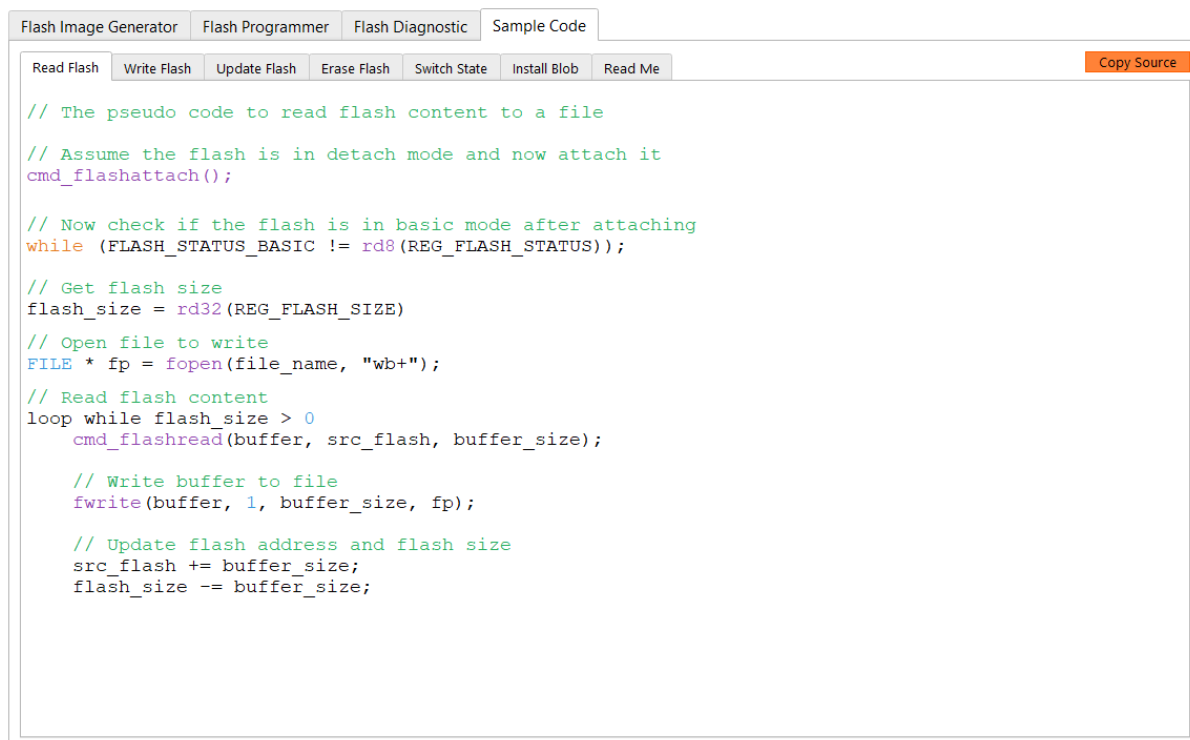


Figure 34 Flash Sample Code

Copy Source: Copy pseudo code in the current tab to the clipboard.

L. Asset Compressor

1. Compressor

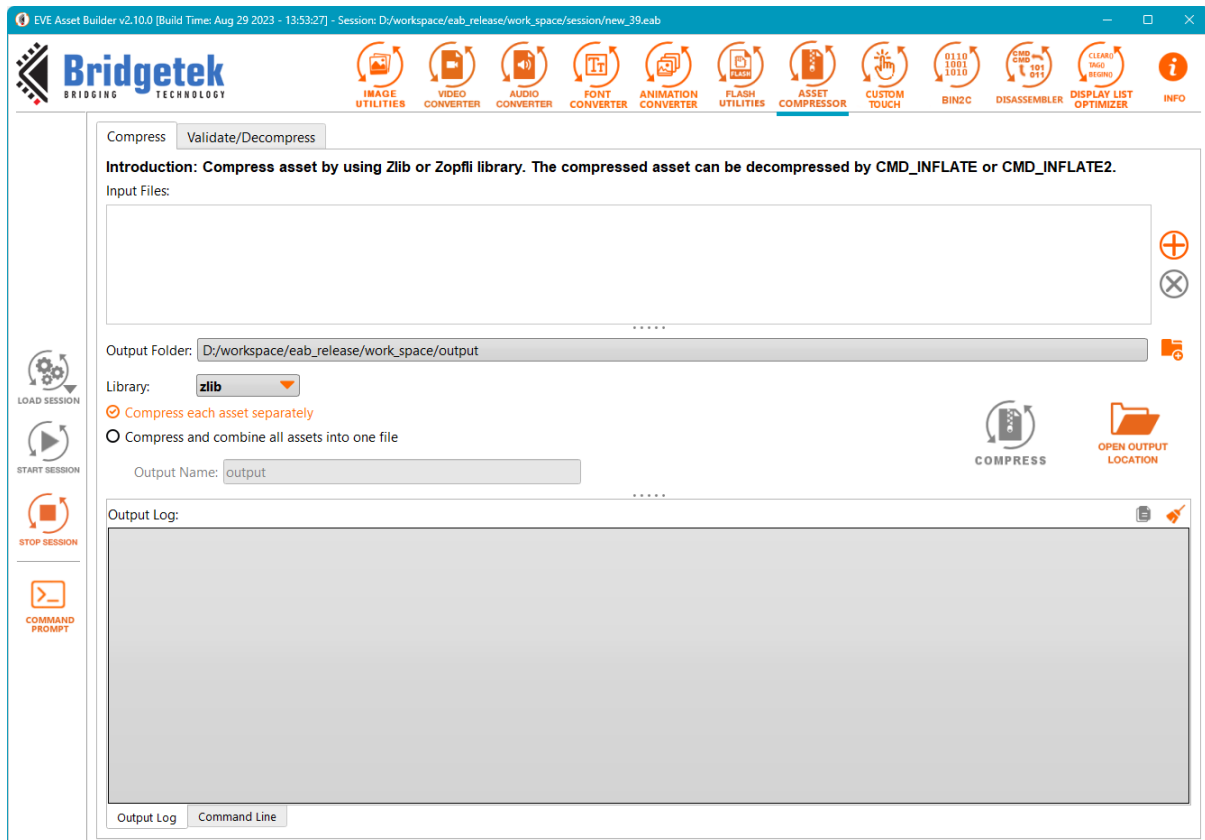


Figure 35 Asset Compressor

Add: Add asset files to compress

Remove: Remove asset file from the list

Output Folder: Folder to save compressed files

Library: Select library to compress. They are **zlib** and **zopfli**.

Compress each asset separately: each input will have one compressed output.

Compress and combine all assets into one file: all inputs will be compressed and combined into only one file.

Note: Zopfli achieves higher compression than other DEFLATE/Zlib implementations but takes much longer to perform the compression. The speed of decompressing Zopfli's output versus zlib's output is practically unaffected.

2. Validator/Decompressor

Check if the input assets are valid for **cmd_inflate** and **cmd_inflate2**. This tool supports to decompress the assets if required.

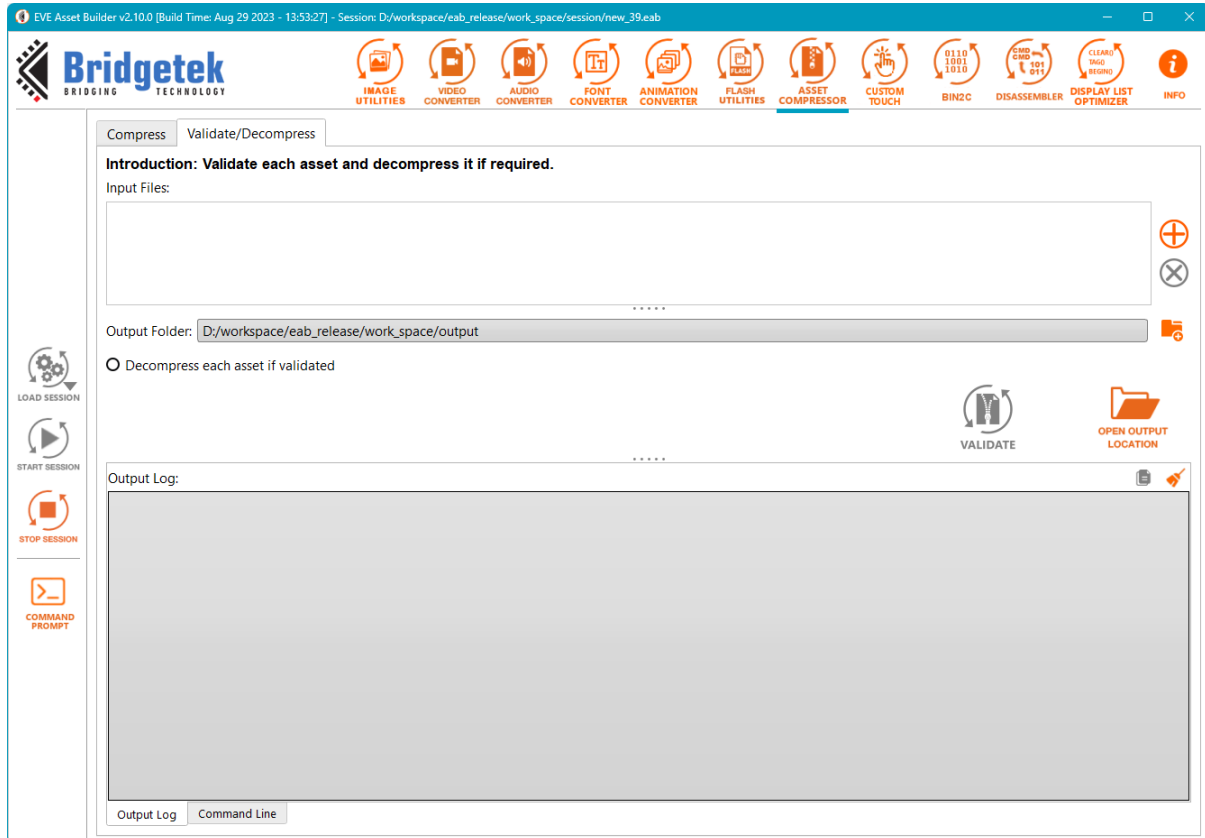


Figure 36 Compressed asset validator

Add: Add files to validate

Remove: Remove files from the list

Output Folder: Folder to save decompressed files

Decompress each asset if validated: Unpack an asset that passed validation when ticked

Validate: Verify the validity of the input files for "cmd_inflate" and "cmd_inflate2", and proceed with decompressing them if required

M. Custom Touch

Compile a small program in a tiny C-like language and produces a loadable firmware image for touch functionality.

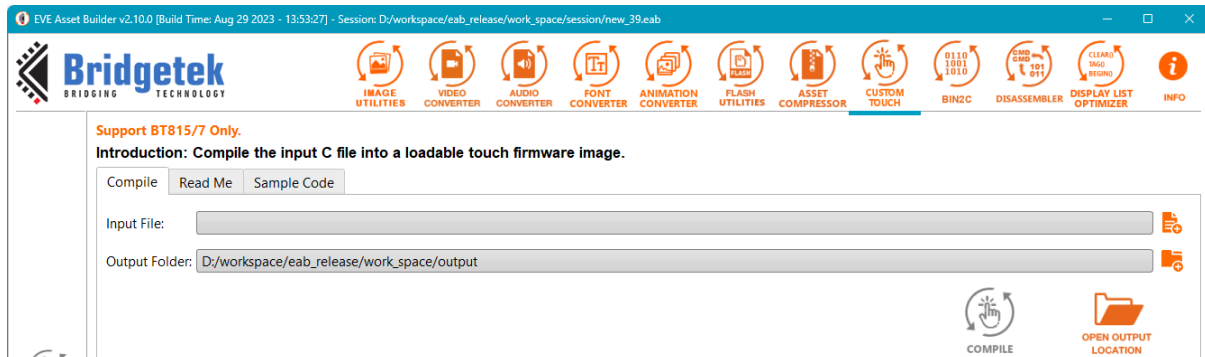


Figure 37 Compile Custom Touch Firmware

Input File: Set a tiny C-like language file to compile

Output Folder: Set a folder to save output files

Read Me Tab: Introduction to the Custom Touch Firmware compiler

Sample Code Tab: Provide source code for some LCD controller ICs

Note: There is the underlying command-line tool to compile the input file, which is called *jtc.exe*, located at <EAB Installation folder>/tools.

Usage: jtc.exe <input.c>

It generates two output files <input>.load.bin and <input>.load.h.

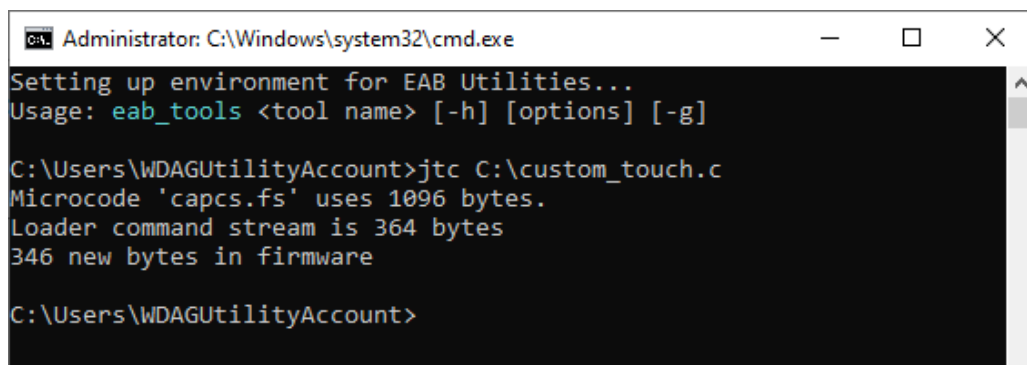


Figure 38 Example of using jtc.exe

Pseudo-code on how to use custom firmware

```

/****
Assume that EVE boot-up sequence is properly done before this routine.
The custom firmware is a piece of code that can be executed by the Eve
coprocessor, so the routine will do:
1) Read the firmware .load.bin into local memory or Use the C array from
the .load.h file
2) Write the firmware into RAM_CMD
3) Update the register REG_CMD_WRITE to start execution
4) Wait till the execution is done (REG_CMD_READ==REG_CMD_WRITE)
****/

```

```
// Read the custom touch firmware 'output.load.bin' into local memory
custom_touch_content = read("output.load.bin", "b");

// Write the custom touch firmware into RAM_CMD
EVE_Hal_wrMem(RAM_CMD, custom_touch_content, sizeof(custom_touch_content));

// Update the write pointer register to start the execution
EVE_Hal_wr32(REG_CMD_WRITE, sizeof(custom_touch_content));

// Wait till Eve completes all the commands
while (EVE_Hal_rd32(REG_CMD_READ) == EVE_Hal_rd32(REG_CMD_WRITE));
```

N. Binary To C-Array

Bin2C is a tool to convert a binary file into a C array.

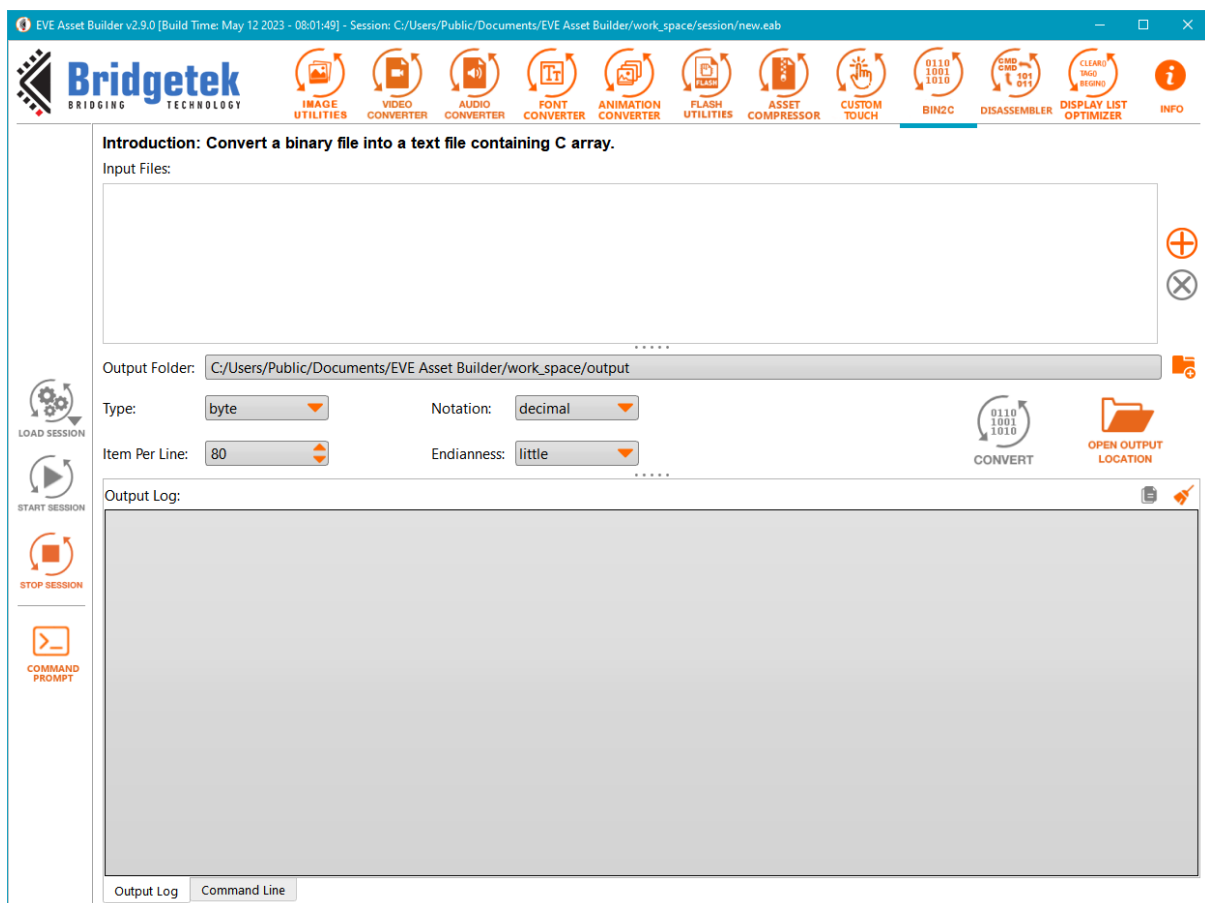


Figure 39 Bin2C Converter

Input Files: List of binary files to be converted

Output Folder: A folder that will contain C array files

Type: Data type of C array, one of byte/word/long

Notation: Set number notation in decimal/hexadecimal

Item Per Line: Number of array items in one line

Endianness: Set endian little/big when reading binary file

O. Disassembler

This tool disassembles a binary file or a text file containing a string of hexadecimal digits to human-readable command text. All display list and coprocessor commands are supported.

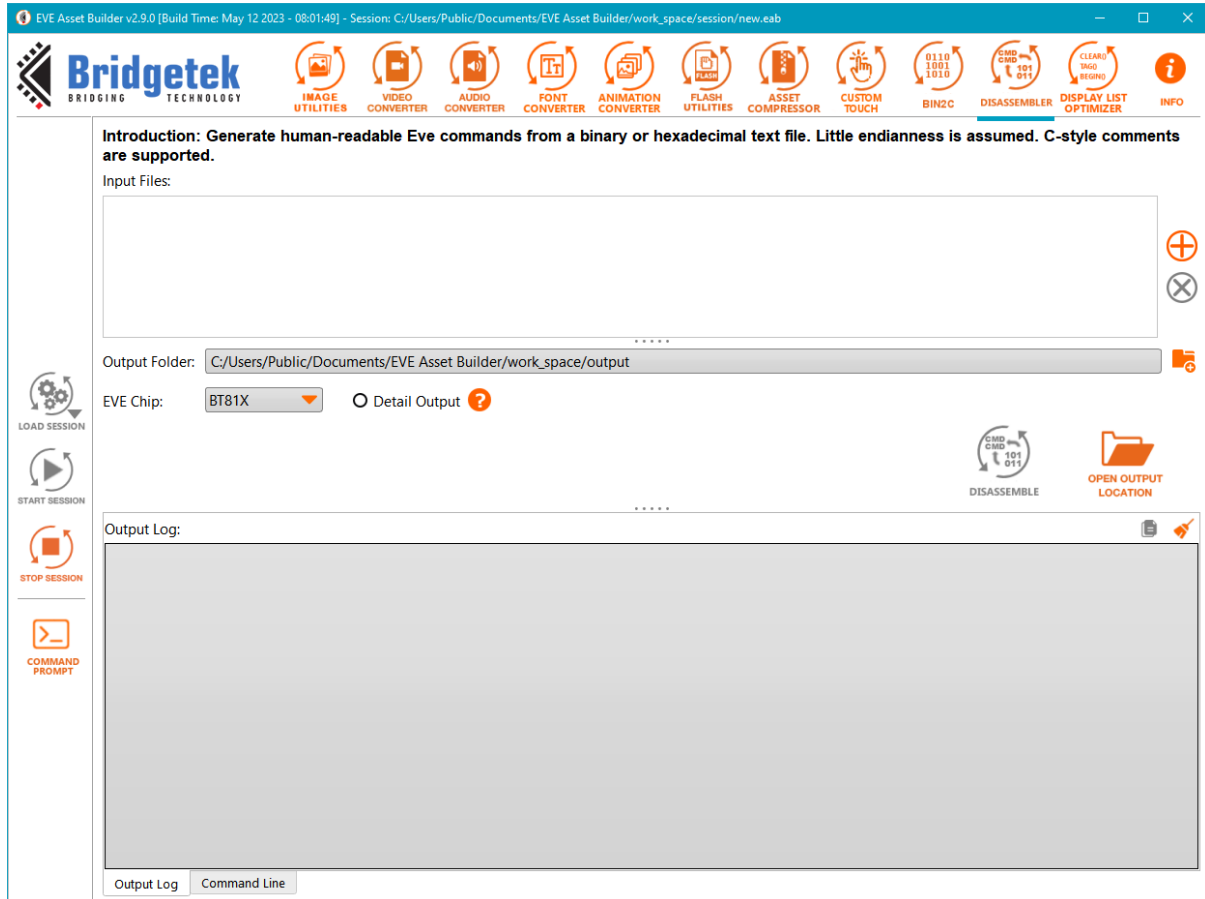


Figure 40 Eve-command Disassembler

Input Files: List of files to be disassembled. Little-endianness is assumed. C-style comments allowed.

Output Folder: A folder to save disassembled files

EVE Chip: Refer to [EVE Chip Series](#)

Detail Output: When checked, the output format will be the same as screen-shot below.

Address	Value (Hex)	Instruction	Parameter	Interpretation Of Parameter
0	0x26000007	CLEAR	0x0000 0007	CLEAR(1, 1, 1)
4	0x22000000	SAVE_CONTEXT	-	SAVE_CONTEXT()
8	0x27000002	VERTEX_FORMAT	0x0000 0002	VERTEX_FORMAT(2)
12	0x0500001C	BITMAP_HANDLE	0x0000 001C	BITMAP_HANDLE(28)
16	0x1F000001	BEGIN	0x0000 0001	BEGIN(BITMAPS)
20	0x95C87E54	VERTEX2II	0x15C8 7E54	VERTEX2II(174, 135, 28, 84)
24	0x97887E65	VERTEX2II	0x1788 7E65	VERTEX2II(188, 135, 28, 101)
28	0x98E87E78	VERTEX2II	0x18E8 7E78	VERTEX2II(199, 135, 28, 120)
32	0x9A487E74	VERTEX2II	0x1A48 7E74	VERTEX2II(210, 135, 28, 116)
36	0x23000000	RESTORE_CONTEXT	-	RESTORE_CONTEXT()
40	0x00000000	DISPLAY	-	DISPLAY()

Figure 41 Detail output of Disassembler tool

Sample inputs

❖ Binary:

copro.bin	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
0000014b	07	00	00	26	4e	ff	ff	ff	00	10	00	00	50	ff	ff	ff
00000000	00	00	00	00	40	00	00	00	43	ff	ff	ff	00	00	00	00
00000010	00	00	b5	00	b9	00	00	00	01	00	00	1f	00	60	23	91
00000020																



```

copro.decode.txt - WordPad
File Home View

CLEAR(1, 1, 1)
CMD_FLASHSOURCE(4096)
CMD_INFLATE2(0, OPT_FLASH)
CMD_SETBITMAP(ARGB1555, 0, 181, 185)
BEGIN(BITMAPS)
VERTEX2II(137, 54, 0, 0)
  
```

❖ Hexadecimal string:

```

hexa_string.txt
File Home View

0x26000007
0x22000000
0x27000002
0x0500001c
0x1f000001
0x95c87e54
0x97887e65
0x98e87e78
0x9a487e74
0x23000000
0x00000000
  
```



```

hexa_string.decode.txt - WordPad
File Home View

CLEAR(1, 1, 1)
SAVE_CONTEXT()
VERTEX_FORMAT(2)
BITMAP_HANDLE(28)
BEGIN(BITMAPS)
VERTEX2II(174, 135, 28, 84)
VERTEX2II(188, 135, 28, 101)
VERTEX2II(199, 135, 28, 120)
VERTEX2II(210, 135, 28, 116)
RESTORE_CONTEXT()
DISPLAY()
  
```

P. Display List Optimizer

This tool helps to optimize display list commands.

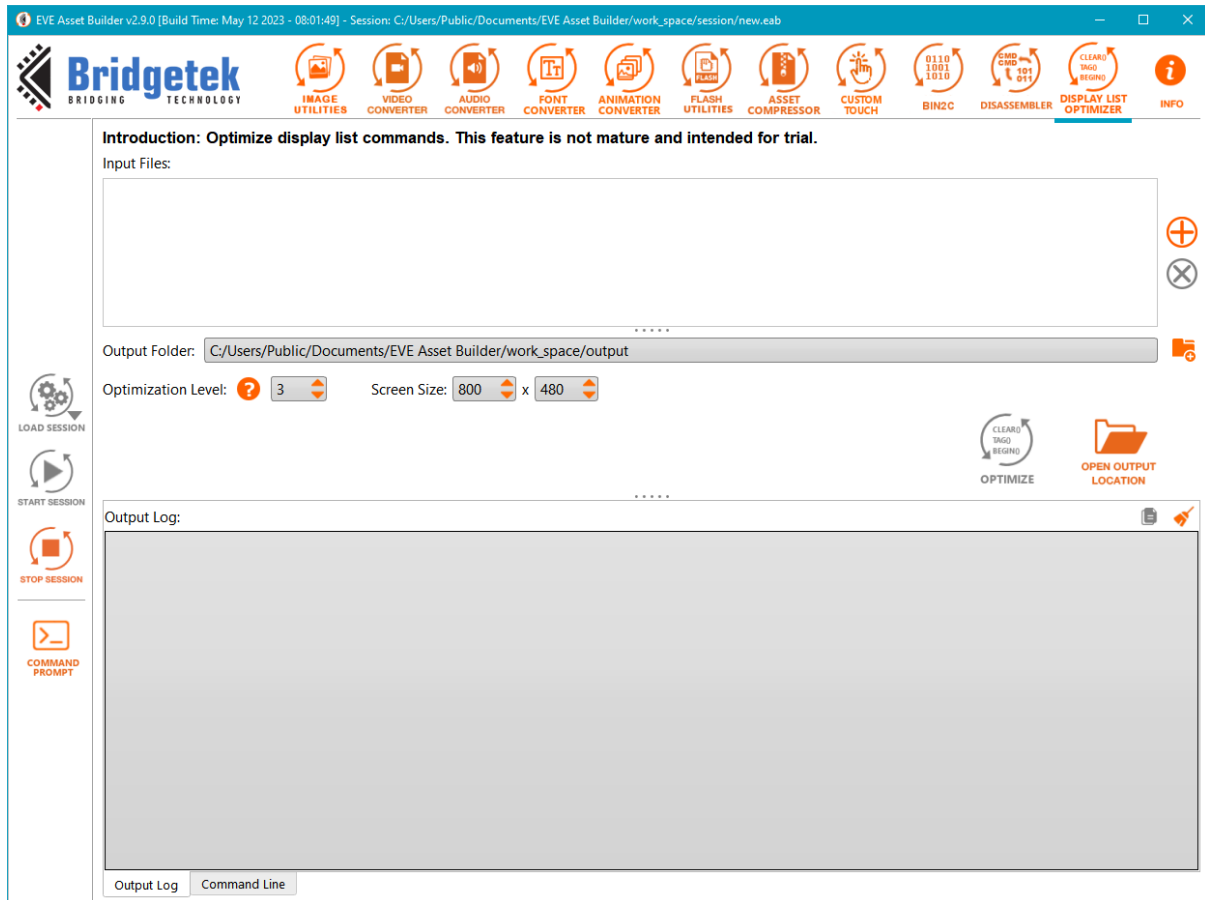


Figure 42 Display List Optimizer

Input Files: List of files to be optimized.

Output Folder: A folder to save optimized files

Optimization Level: Range from 0-3

- 0 - No optimize
- 1 - Remove VERTEX_FORMAT, VERTEX_TRANSLATE_X/Y, END
- 2 - LEVEL_1 + remove the off-screen drawing primitive
- 3 - LEVEL_2 + remove hidden objects partially/totally

Screen Size: Set screen resolution so that the tool knows which drawing object is off-screen

Optimize: Start optimizing each input file

Q. SampleApp

After converting EVE resources, EAB will generate a **SampleApp** project. It contains sample code that demonstrates the usage of converted resources. The SampleApp project is prepared for the Bridgetek modules VM816CU50A or ME817EV which have built-in FT4222H.

Example: Convert image **lenaface.png** into ARGB1555 format.

```
void Load_Image_lenaface_32x32_ARGB1555(EVE_HalContext * phost)
{
    Gpu_Hal_WaitCmdfifo_empty(phost);

    Gpu_CoCmd_Dlstart(phost);
    App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));
    App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));

    uint16_t iw = 32;
    uint16_t ih = 32;
    uint16_t format = ARGB1555;

#ifdef USE_BT81X_FLASH
#define BITMAP_ADDRESS_ON_FLASH <address> // address of bitmap file from Flash Map
#define BITMAP_SIZE_ON_FLASH <size> // size of bitmap file from Flash Map
    /* Switch Flash to FULL Mode */
    Gpu_CoCmd_FlashHelper_SwitchFullMode(phost);

    if (format >= compressedED_RGBA_ASTC_4x4_KHR && format <= compressedED_RGBA_ASTC_12x12_KHR) {
        Gpu_CoCmd_SetBitmap(phost, (0x800000 | BITMAP_ADDRESS_ON_FLASH / 32), format, iw, ih);
    } else {
        Gpu_CoCmd_FlashRead(phost, RAM_G, BITMAP_ADDRESS_ON_FLASH, BITMAP_SIZE_ON_FLASH);
        Gpu_CoCmd_SetBitmap(phost, RAM_G, format, iw, ih);
    }
#else
    //load bitmap file into graphics RAM
    //RAM_G is starting address in graphics RAM, for example 00 0000h
    Gpu_Hal_LoadImageToMemory(phost, "../../lenaface_32x32_ARGB1555.raw", RAM_G, LOAD);
    Gpu_CoCmd_SetBitmap(phost, RAM_G, format, iw, ih);
#endif
    //Start drawing bitmap
    App_WrCoCmd_Buffer(phost, BEGIN(BITMAPS));
    App_WrCoCmd_Buffer(phost, VERTEX2II(0, 0, 0, 0));
    App_WrCoCmd_Buffer(phost, END());
    App_WrCoCmd_Buffer(phost, RESTORE_CONTEXT());
    App_WrCoCmd_Buffer(phost, DISPLAY());
    Gpu_CoCmd_Swap(phost);
    App_Flush_Co_Buffer(phost);
    Gpu_Hal_WaitCmdfifo_empty(phost);
}
```

Figure 43 Sample code: load ARGB1555 image

The `main()` function will call `eab_test_feature()`. Then `eab_test_feature()` calls the dedicated `Load_Image_lenaface_32x32_ARGB1555()`.

```
void eab_test_feature(EVE_HalContext * phost)
{
    Load_Image_lenaface_32x32_ARGB1555(phost);
    /* INSERT GENERATED FUNCTION */
}

int32_t main(int32_t argc, char8_t *argv[])
{
    phost = &ph;
    Gpu_Init(phost);
    EVE_Util_clearScreen(phost);

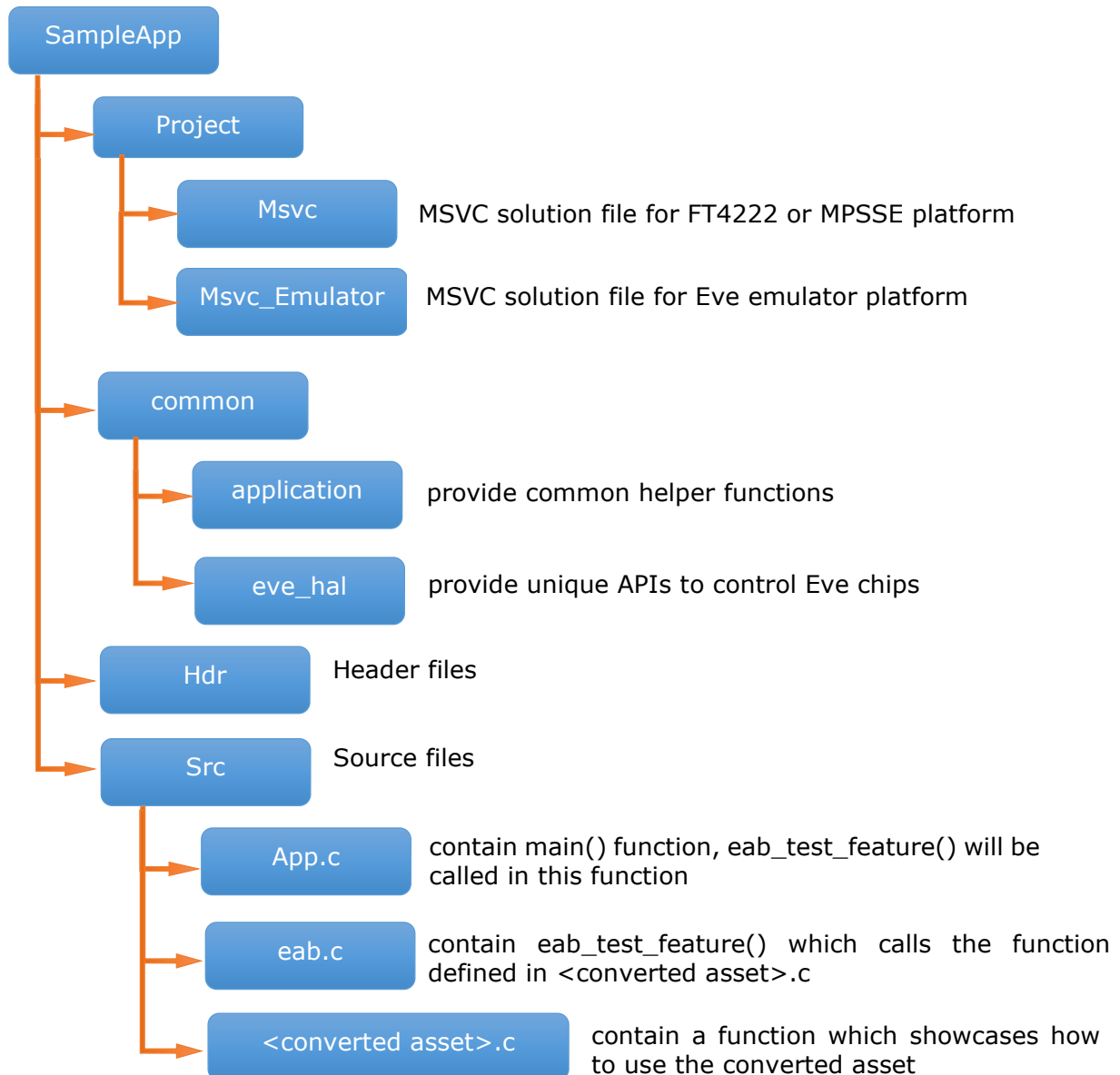
    eab_test_feature(phost);

    system("pause");
    Gpu_Release(phost);
    return 0;
}
```

Figure 44 SampleApp: main() function

Usually, most of the parameters are well-prepared in SampleApp, users just need to open the solution file at *SampleApp/Project/MSVC/ SampleApp_MSVC.sln*, then build and run in the connected Eve board. The converted resource should be rendered in the hardware LCD.

SampleApp folder structure



V. Example

A. Image Converter

Below is an example of converting a **PNG** file into an ASTC format 4x4 file and showcasing how the output file is to be utilized.

Step 1: Convert the input **PNG** file into ASTC

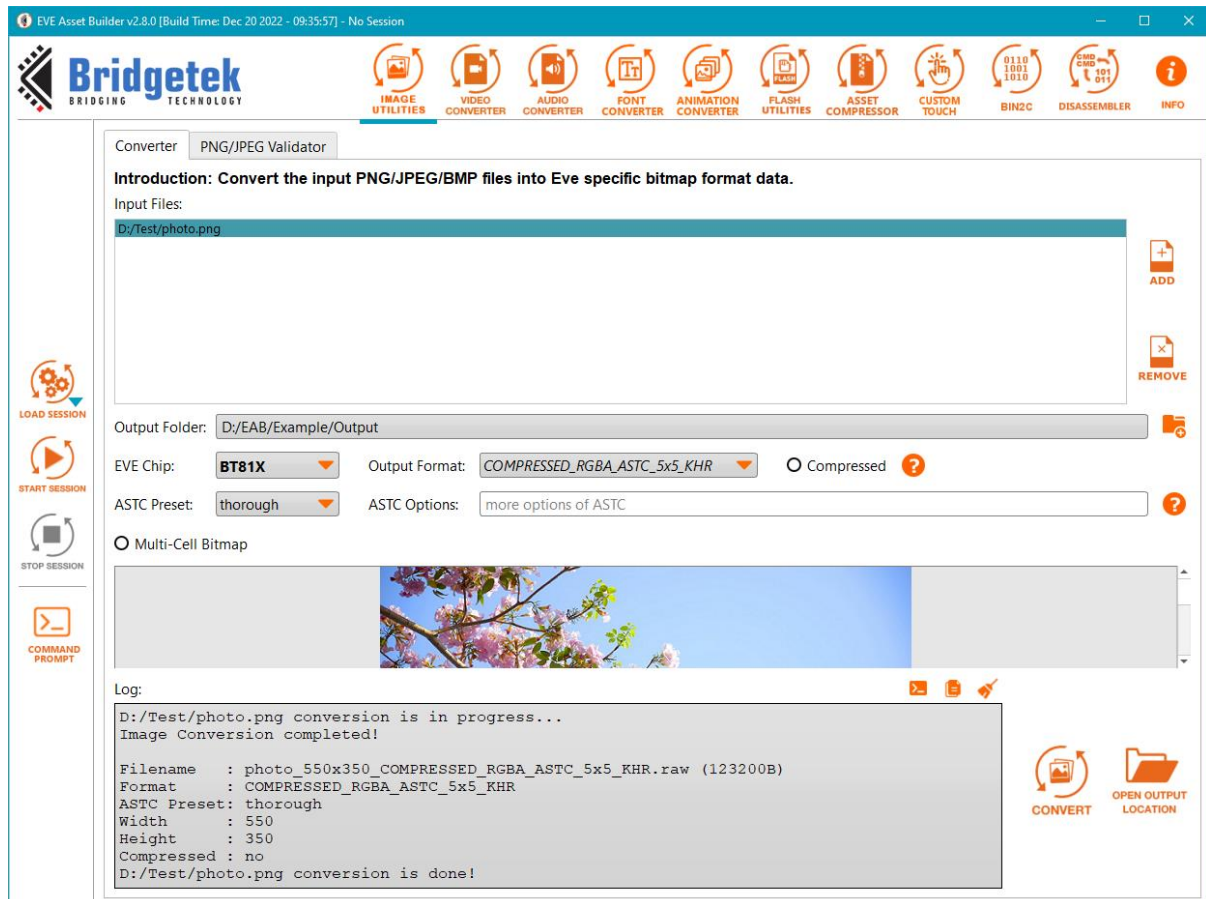


Figure 45 Image Converter example on EAB

Output folder:

↑ Name	Ext	Size	Date
↑ [..]		<DIR>	25/05/2020 11:11
photo_800x480_COMPRESSED_RGBA_A.. c	c	1,455	25/05/2020 11:10
photo_800x480_COMPRESSED_RGBA_A.. json	json	234	25/05/2020 11:10
photo_800x480_COMPRESSED_RGBA_A.. raw	raw	245,760	25/05/2020 11:10
photo_800x480_COMPRESSED_RGBA_A.. rawh	rawh	878,293	25/05/2020 11:10
photo_800x480_COMPRESSED_RGBA_A.. png	png	389,331	25/05/2020 11:10

Figure 46 Image Converter example: Output folder

The generated photo_800x480_COMPRESSED_RGBA_ASTC_5x5_KHR.c contains a function Load_Image() to load the generated .raw file:

Step 2: Call that function inside the SampleApp MSVC project, images will be displayed on LCD:



Figure 47 Image Converter example: LCD screen

B. Video Converter

Below is an example to convert an AVI file and shows how the output file is to be utilized.

Step 1: Convert video file into EVE-compatible file by EAB

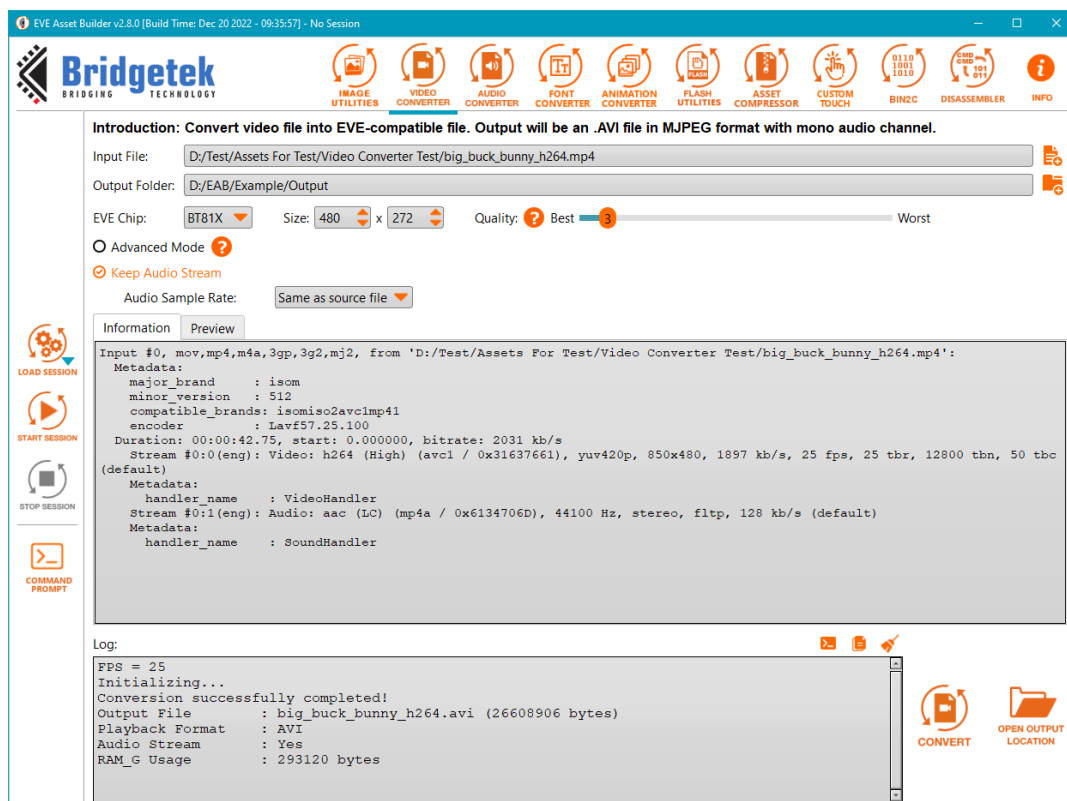


Figure 48 Video Converter example

Output folder:



Name	Date modified	Type	Size
 big_buck_bunny_h264.avi	20-Dec-22 1:30 PM	AVI Video File (VLC)	25,986 KB
 SampleApp	12-Dec-22 2:17 PM	File folder	

Figure 49 Video Converter example: Output folder

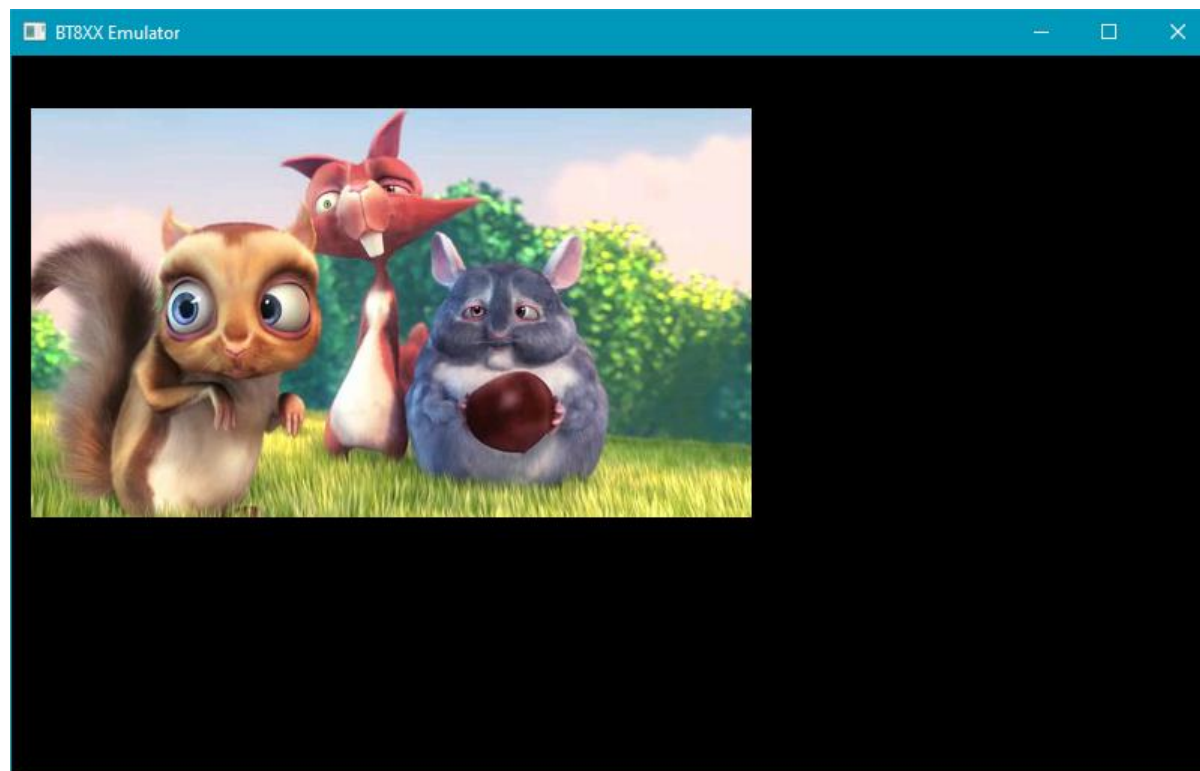
The generated video playback.c contains a C function to load the generated .raw file:

```
void VideoPlayback()
{
    #if USE_BT81X_FLASH
        /* Switch Flash to FULL Mode */
        Gpu_CoCmd_FlashHelper_SwitchFullMode(&host);
        Gpu_Hal_Wr32(phost, REG_PLAY_CONTROL, 1);
        Gpu_CoCmd_Dlstart(phost);
        App_WrCoCmd_Buffer(phost, CLEAR(1, 1, 1));
        App_WrCoCmd_Buffer(phost, COLOR_RGB(255, 255, 255));

        Gpu_CoCmd_FlashSource(phost, ADDR_VIDEO);
        App_WrCoCmd_Buffer(phost, CMD_PLAYVIDEO);
        App_WrCoCmd_Buffer(phost, OPT_FLASH | OPT_SOUND | OPT_NOTEAR);
        App_Flush_Co_Buffer(phost);
        Gpu_Hal_WaitCmdfifo_empty(phost);
    #endif
}
```

Figure 50 Video Converter example: sample code

Step 2: Call function VideoPlayback from sample app MSVC_Emulator project, video will be played back.


Figure 51 Video Converter example: play back

C. Audio Converter

Below is an example of convert a WAV file and how the output file is to be utilized.

Step 1: Convert the audio file into an EVE-compatible file

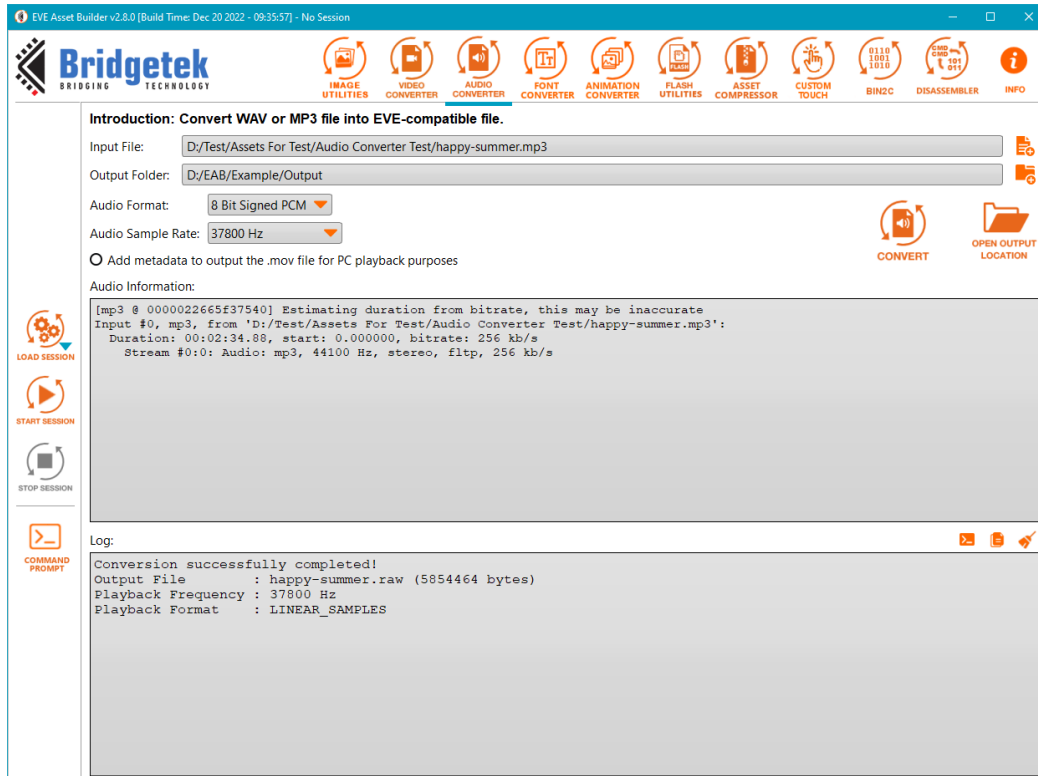


Figure 52 Audio Converter example on EAB

Output folder:




Name	Date modified	Type	Size
 happy-summer.raw	20-Dec-22 1:45 PM	RAW File	5,718 KB
 happy-summer.json	20-Dec-22 1:45 PM	JSON File	1 KB
 SampleApp	12-Dec-22 2:17 PM	File folder	

Figure 53 Audio Converter example: Output folder

The generated SampleApp contains a C function to load the generated .raw file:

```
void Audio_Playback(EVE_HalContext *host)
{
    phost = host;
    Fifo_Init(&MediaFifo, MEDIA_FIFO_ADDR, MEDIA_FIFO_BUFFER, REG_MEDIAFIFO_READ, REG_MEDIAFIFO_WRITE);
    Gpu_CoCmd_MediaFifo(phost, MEDIA_FIFO_ADDR, MEDIA_FIFO_BUFFER);
    printf("Mediafifo: Start address and length %d %d\n", MEDIA_FIFO_ADDR, MEDIA_FIFO_BUFFER);
    App_Flush_Co_Buffer(phost);
    Gpu_Hal_WaitCmdfifo_empty(phost);
    MediaFile.pfile = fopen(".././../happy-summer.raw", "rb");
    if (MediaFile.pfile == NULL) {
        return;
    }

    fseek(MediaFile.pfile, 0, SEEK_END);
    MediaFile.FILE_SIZE = MediaFile.file_len = ftell(MediaFile.pfile);
    MediaFile.played_len = 0;
}
```

Figure 54 Audio Converter example: Sample code

Step 2: Call function Audio_Playback () from SampleApp MSVC project, audio will play LCD.

D. Font Converter

Below is an example of converting a TrueType Font File(.ttf) and how the output file is utilized.

Step 1: Convert font file into EVE-compatible file

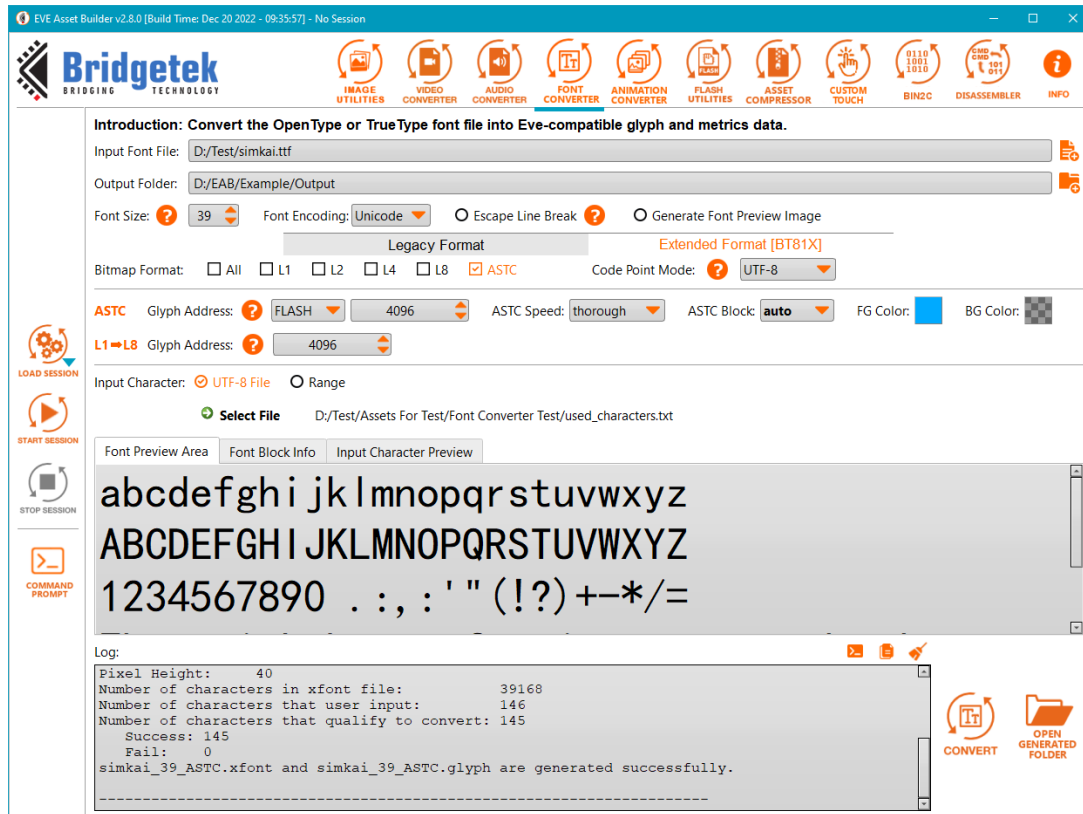


Figure 55 Font Converter example

Output folder:

Name	Date modified	Type	Size
SampleApp	12-Dec-22 2:17 PM	File folder	
simkai_39_ASTC.c	20-Dec-22 1:53 PM	C File	1 KB
simkai_39_ASTC.glyph	20-Dec-22 1:53 PM	GLYPH File	621 KB
simkai_39_ASTC.json	20-Dec-22 1:53 PM	JSON File	1 KB
simkai_39_ASTC.xfont	20-Dec-22 1:53 PM	XFONT File	4 KB
simkai_39_ASTC_converted_chars_index.html	20-Dec-22 1:53 PM	Chrome HTML Document	9 KB

Figure 56 Font Converter example: Output folder

The generated simkai_39_ASTC.c contains a C function to load the generated .raw file:

```
void LoadXfont()
{
    uint32_t fontAddr = RAM_G;

    // NOTE: Remember to write glyph file into BT815's flash at address 4096
    // Switch Flash to FULL Mode
    Gpu_CoCmd_FlashHelper_SwitchFullMode(phost);

    // Load xfont file into graphics RAM
    Gpu_Hal_LoadImageToMemory(phost, "path\\to\\simkai_39_ASTC.xfont", fontAddr, LOAD);
    Gpu_Hal_WaitCmdfifo_empty(phost);
}
```

Figure 57 Font Converter example: sample code

Step 2: Call function LoadXfont() from SampleApp MSVC project, text will be displayed on LCD:

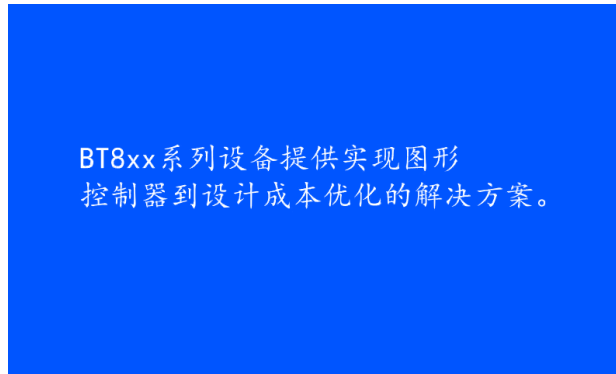
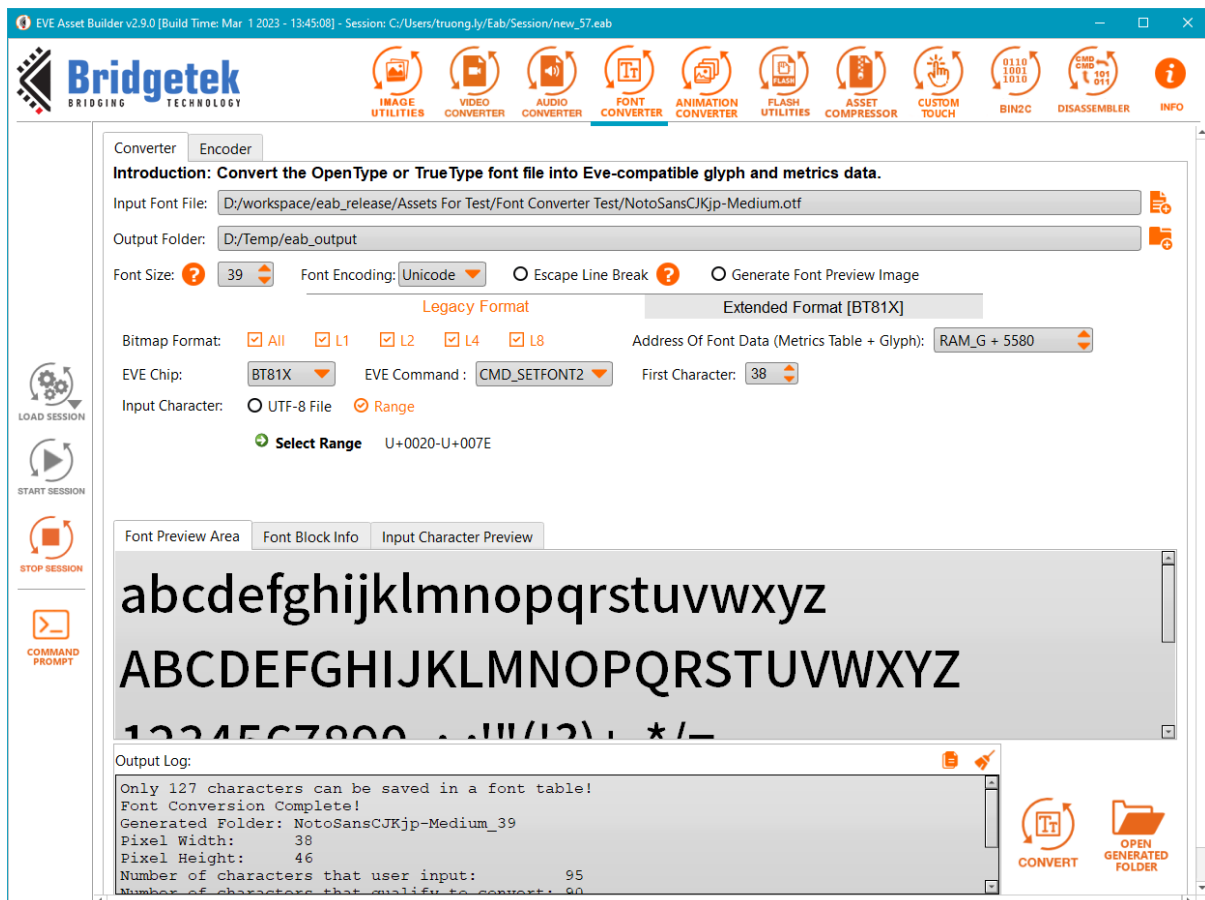


Figure 58 Font Converter example: LCD screen

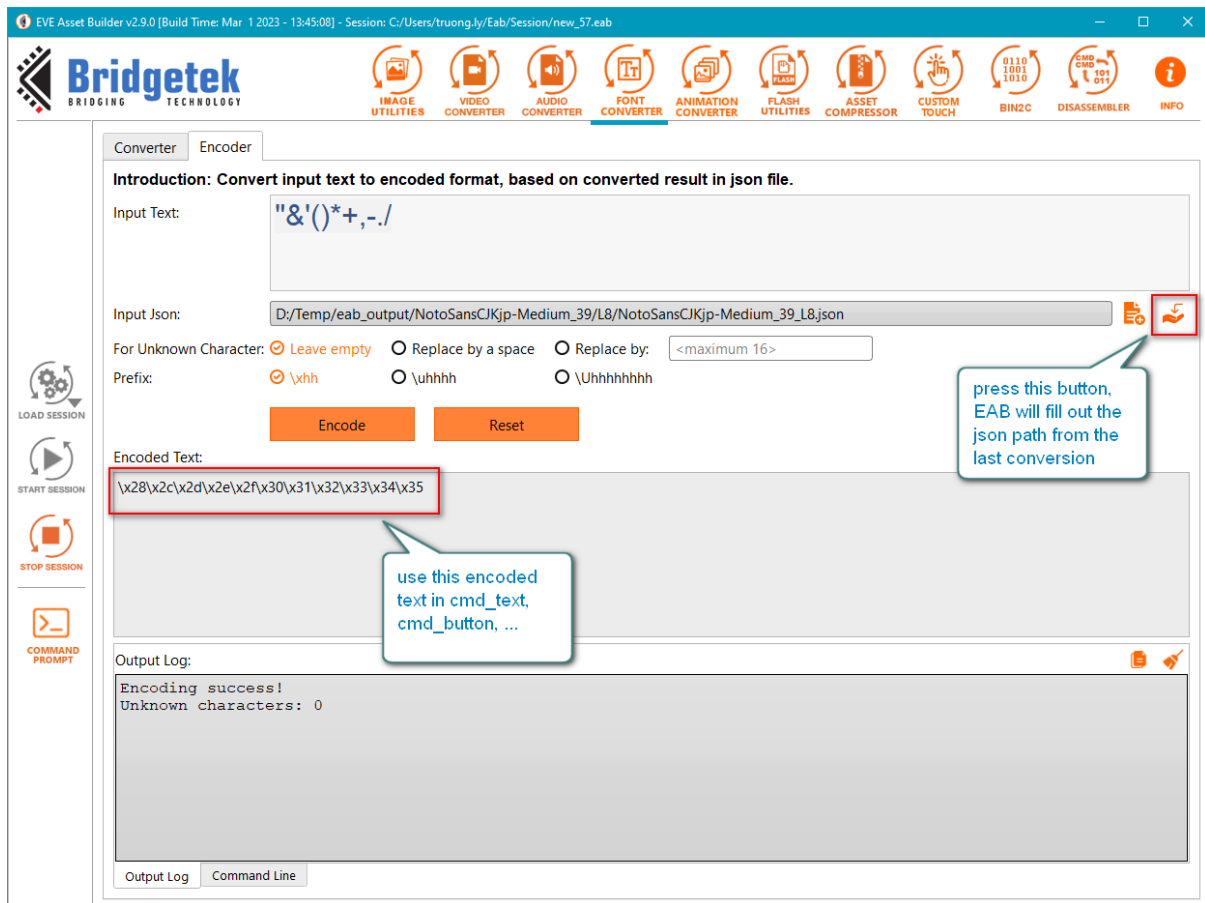
E. Font Encoder

Font encoder helps to encode an input string into the format of `\xhh`, `\uhhhh`, or `\Uhhhhhhhhh`. The `hh` value is retrieved from the json file. The encoded string can be used in coprocessor command such as `CMD_TEXT`, `CMD_BUTTON`.

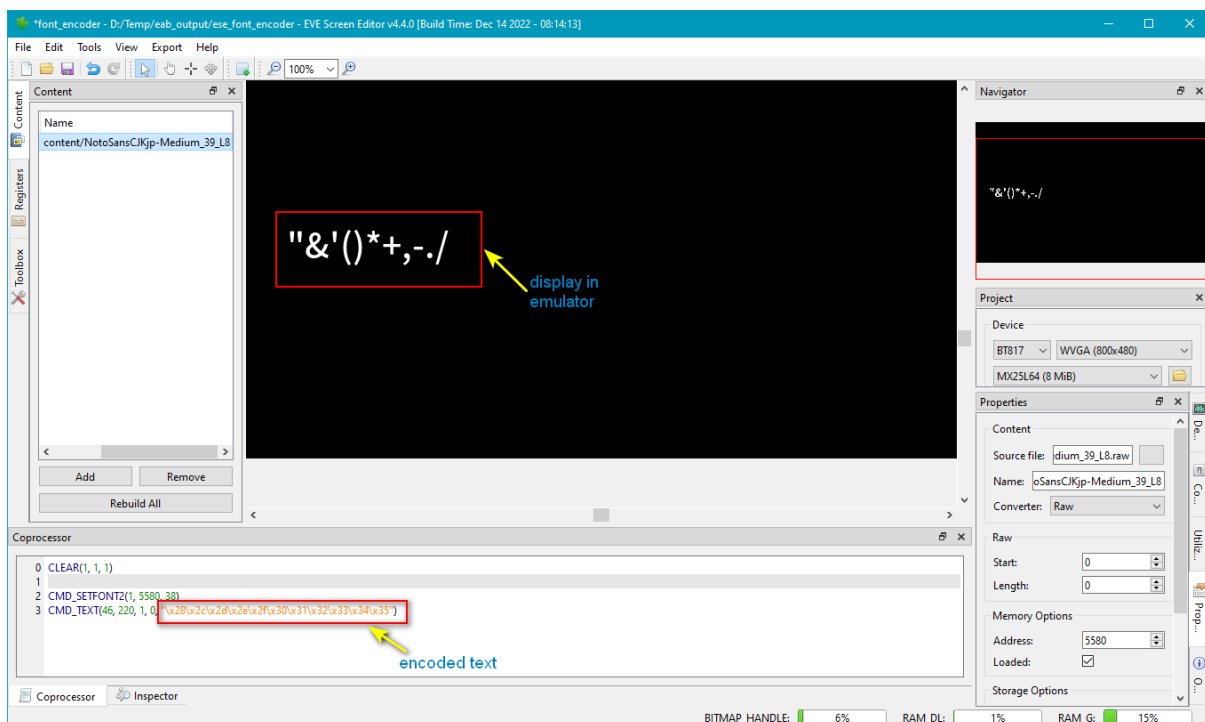
Step 1: Convert a font



Step 2: Switch to Font Encoder tab, then input some text



Step 3: Use the encoded text in ESE



F. Animation Converter and Generate Flash Image

Below is an example to convert a gif file into an animation, generate a flash image, and display that animation on LCD.

Step 1: Convert a gif file into EVE-compatible file

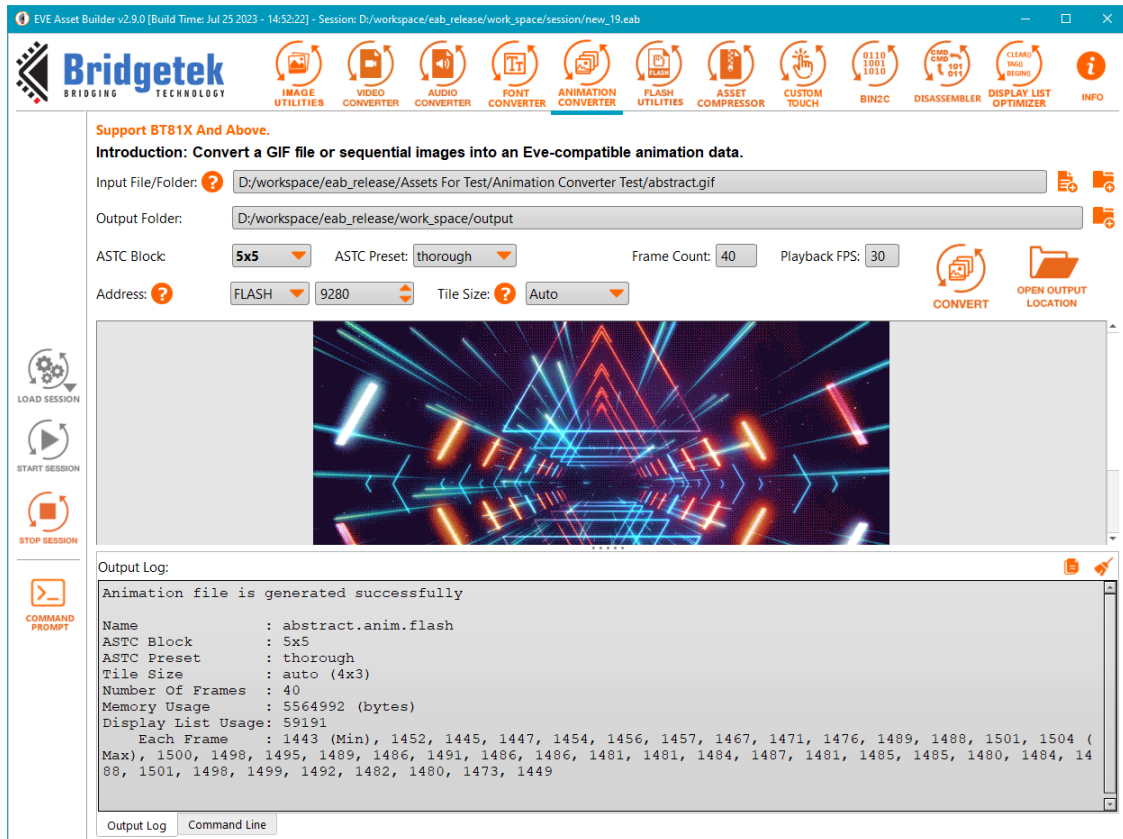


Figure 59 Animation Converter example

Output folder:

Name	Date	Type	Size
SampleApp	26-Jul-23 2:54 PM	File folder	
abstract.anim	26-Jul-23 2:54 PM	ANIM File	3,250 KB
abstract.anim.flash	26-Jul-23 2:54 PM	FLASH File	5,435 KB
abstract.anim.json	26-Jul-23 2:54 PM	JSON File	1 KB

Figure 60 Animation Converter example: Output folder

Note:

- .anim is a zip format which contains a series of ASTC images and one config file. The user needs to generate a flash image from .anim, so that Eve chip can utilize it. Bitmap addresses can be changed while generating flash image.
- .anim.flash or .anim.ram_g is a flash image, which is prepared for convenience. The user can load it to flash or ram_g, then it is ready to use. Bitmap address are fixed.
- in flash image, animation is separated into two parts: *anim.data* and *anim.object*. The offset of anim.object will be use in CMD_ANIMxxx.

The generated "abstract_anim.c" in SampleApp folder contains a function to load the .anim.flash file:

```
void LoadAnimation(Gpu_Hal_Context_t *phost, uint8_t frame_rate)
{
    double duration = 1000.0 / frame_rate;
    double remain = 0;
    double frame_time = 0;

    // Switch Flash to FULL Mode
    Gpu_CoCmd_FlashHelper_SwitchFullMode(phost);
}
```

Figure 61 Animation Converter example: Sample code

Step 2: Generate a flash image from .anim file:

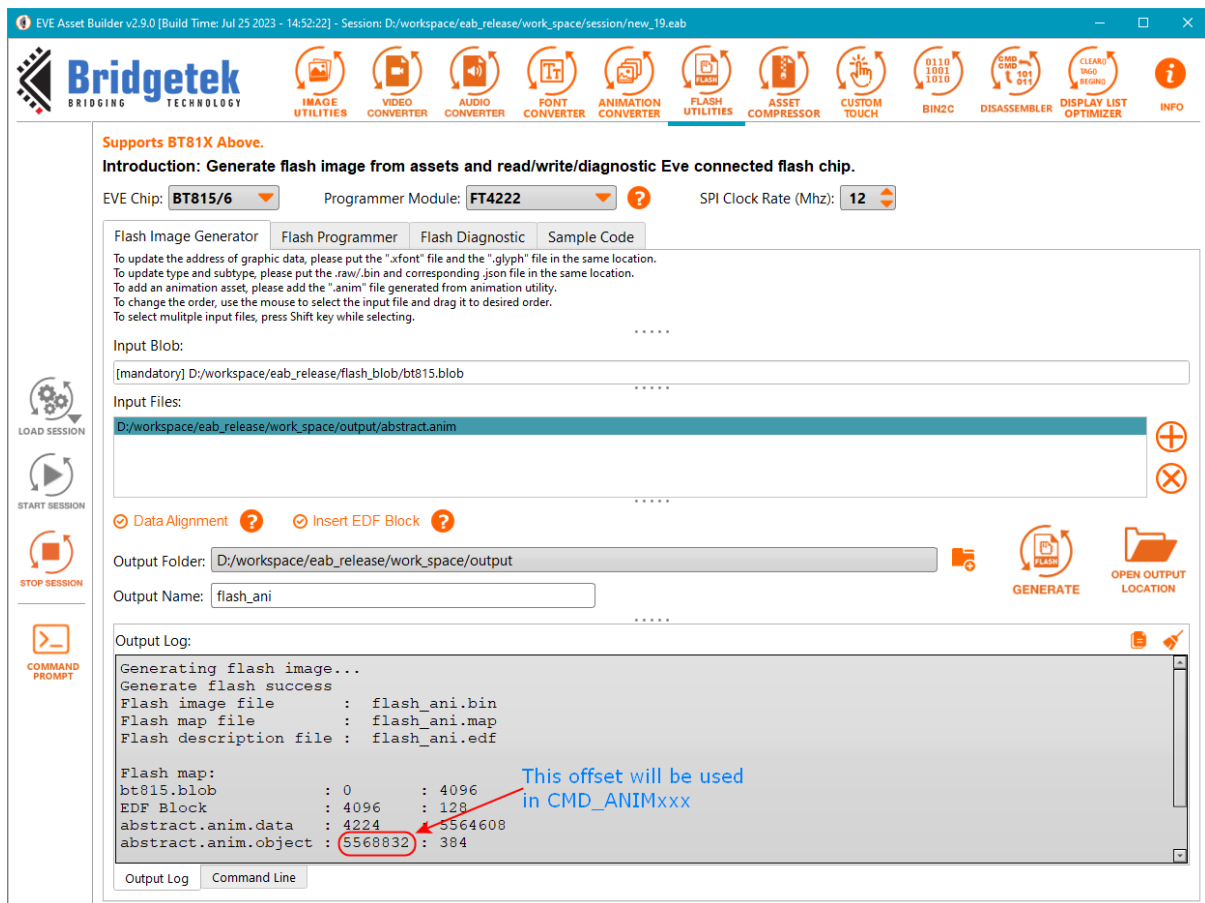
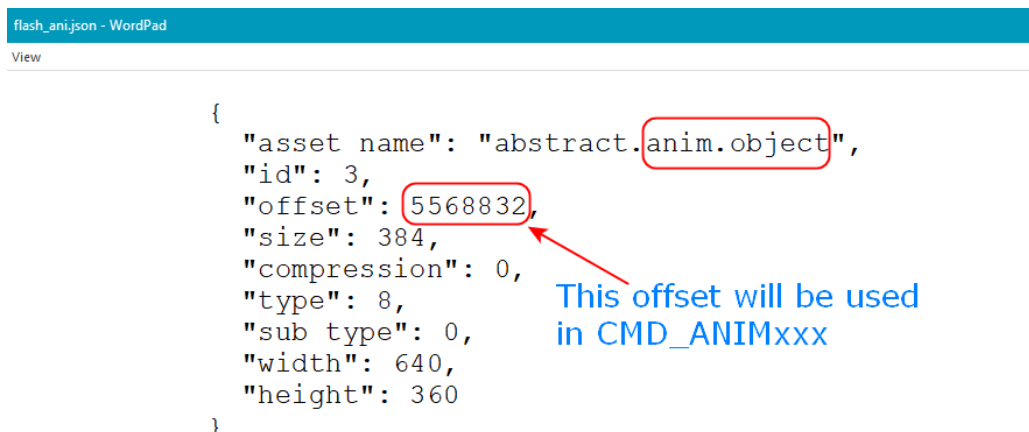


Figure 62 Animation Converter example: Generate flash



Step 3: Write the flash image into EVE's flash chip:

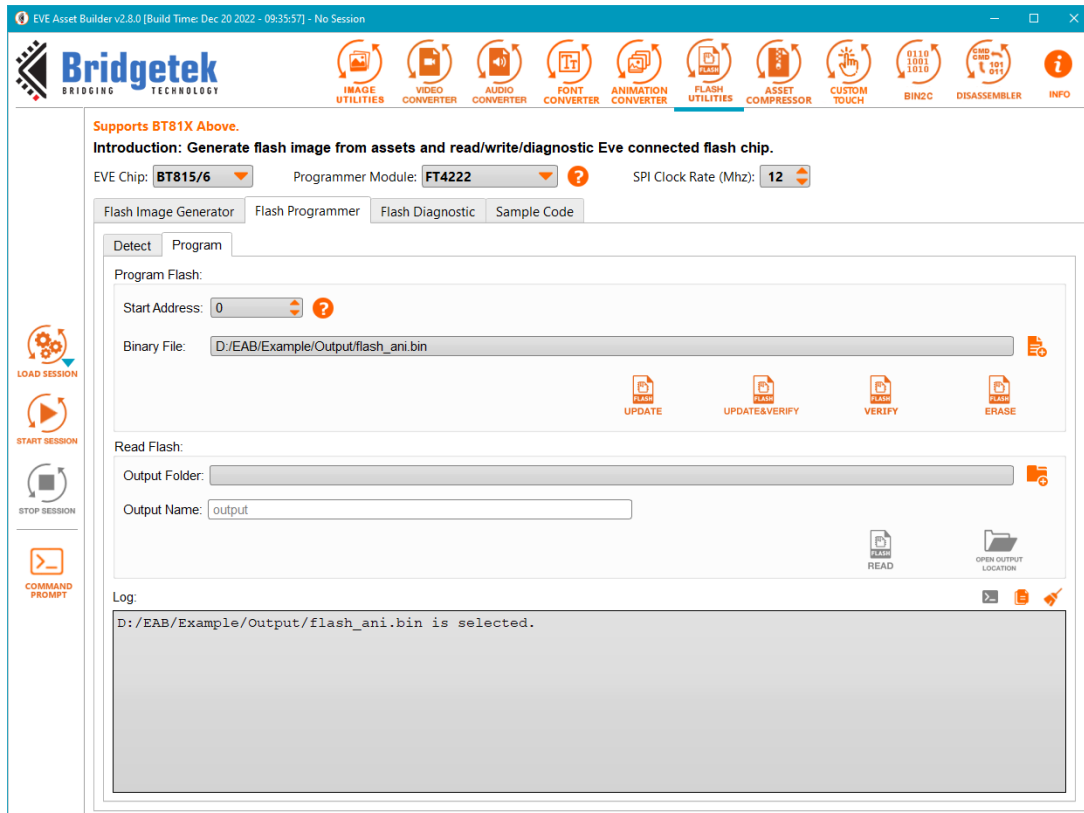


Figure 63 Animation Converter example: Program flash

Step 4: Call function LoadAnimation() from SampleApp MSVC_Emulator project, update <ANIM_ADDR> with the *offset* of *abstract.anim.object* in .json file in step 2, and animation will be played.

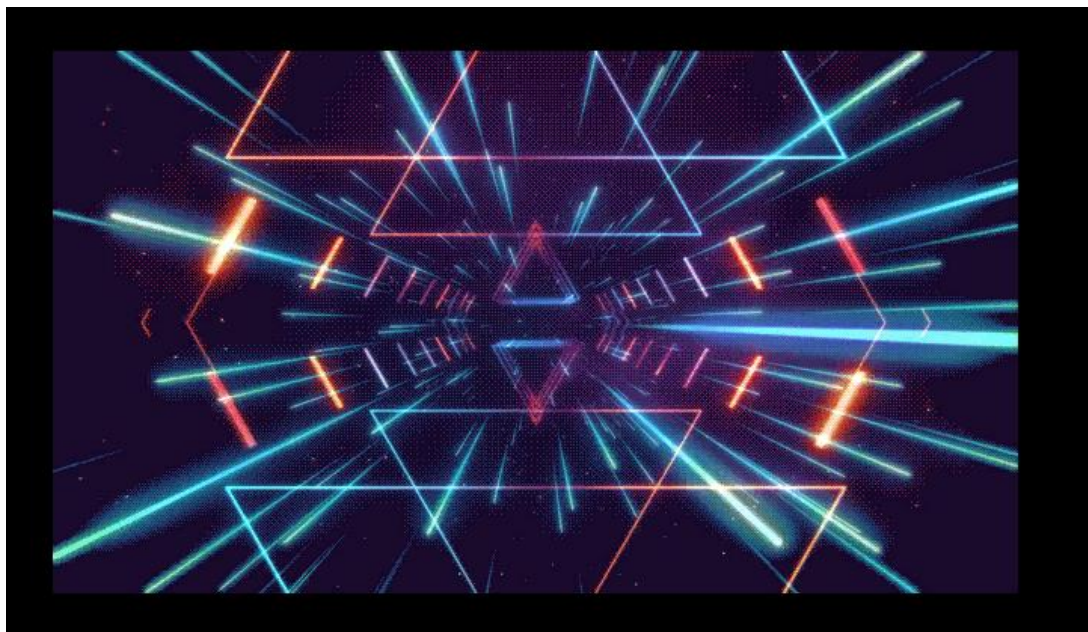


Figure 64 Animation Converter example: play back

Contact Information

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

Distributor and Sales Representatives

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

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Bridgetek Pte Ltd (BRTChip) devices incorporated in their systems, meet all applicable safety, regulatory, and system-level performance requirements. All application-related information in this document (including application descriptions, suggested Bridgetek devices, and other materials) is provided for reference only. While Bridgetek has taken care to assure it is accurate, this information is subject to customer confirmation, and Bridgetek disclaims all liability for system designs and any application assistance provided by Bridgetek. Use of Bridgetek devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify, and hold harmless Bridgetek from any damages, claims, suits, or expenses resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted, or reproduced in any material or electronic form without the prior written consent of the copyright holder. Bridgetek Pte Ltd, 1 Tai Seng Avenue, Tower A, #03-05, Singapore 536464. Singapore Registered Company Number: 201542387H.

Appendix A – List of Figures

Figure 1 EVE Asset Builder Setup Wizard.....	6
Figure 2 EVE Asset Builder Setup – Select the destination folder	7
Figure 3 EVE Asset Builder Setup – Ready for Installation	7
Figure 4 EVE Asset Builder Setup – Installing in Progress	8
Figure 5 EVE Asset Builder Setup – Finish	8
Figure 6 EVE chip series.....	9
Figure 7 Session	9
Figure 8 Session Dialog.....	10
Figure 9 Select session file	10
Figure 10 Command Prompt Window	11
Figure 11 Example of converting an image	11
Figure 12 Log Window	12
Figure 13 Image Converter tab	14
Figure 14 PNG/JPEG Validator tab	15
Figure 15 JPEG Optimizer tab	16
Figure 16 Video Converter tab	17
Figure 17 Audio Converter tab	18
Figure 18 Font Converter tab.....	19
Figure 19 Font Preview Area.....	20
Figure 20 Font Block Info	20
Figure 21 Legacy Format.....	20
Figure 22 Extended Format	21
Figure 23 Encoder tab	22
Figure 24 Animation Converter	22
Figure 25 Generate Flash Image	24
Figure 26 Detect Flash.....	26
Figure 27 Program Flash	27
Figure 28 Read Flash	28
Figure 29 Enable "Install Blob".....	28
Figure 30 Programming flash using Raspberry Pi Pico	29
Figure 31 Raspberry Pi Pico Readme.....	29
Figure 32 Convert binary file to Uf2 format	29
Figure 33 Flash Diagnosis.....	30
Figure 34 Flash Sample Code.....	31
Figure 35 Asset Compressor	32
Figure 36 Compressed asset validator	33
Figure 37 Compile Custom Touch Firmware	34
Figure 38 Example of using jtc.exe.....	34
Figure 39 Bin2C Converter	35
Figure 40 Eve-command Disassembler.....	36
Figure 41 Detail output of Disassembler tool.....	36
Figure 42 Display List Optimizer.....	38
Figure 43 Sample code: load ARGB1555 image.....	39
Figure 44 SampleApp: main() function	40
Figure 45 Image Converter example on EAB.....	42
Figure 46 Image Converter example: Output folder.....	42
Figure 47 Image Converter example: LCD screen.....	43
Figure 48 Video Converter example.....	43
Figure 49 Video Converter example: Output folder.....	44
Figure 50 Video Converter example: sample code	44
Figure 51 Video Converter example: play back	44
Figure 52 Audio Converter example on EAB.....	45
Figure 53 Audio Converter example: Output folder.....	45

Figure 54 Audio Converter example: Sample code.....	45
Figure 55 Font Converter example	46
Figure 56 Font Converter example: Output folder.....	46
Figure 57 Font Converter example: sample code.....	46
Figure 58 Font Converter example: LCD screen	47
Figure 59 Animation Converter example	49
Figure 60 Animation Converter example: Output folder.....	49
Figure 61 Animation Converter example: Sample code	50
Figure 62 Animation Converter example: Generate flash	50
Figure 63 Animation Converter example: Program flash.....	51
Figure 64 Animation Converter example: play back.....	51

Appendix B – List of Tables

Table 1 Naming Convention and usage of output files	13
Table 2 Asset Type.....	25
Table 3 subType of Bitmap.....	26
Table 4 subType of Audio	26

Appendix C – Revision History

Document Title : BRT_AN_054 EVE Asset Builder 2.10 User Guide
 Document Reference No. : BRT_000263
 Clearance No. : BRT#161
 Product Page : <https://brtchip.com/toolchains/>
 Document Feedback : [Send Feedback](#)

Revision	Changes	Date
Version 1.5	User Guide for EVE Asset Builder 2.0.0 release	12-08-2020
Version 1.6	Updated User Guide for EVE Asset Builder 2.1.0	17-12-2020
Version 1.7	Updated User Guide for EVE Asset Builder 2.4.0	11-11-2021
Version 1.8	Updated User Guide for EVE Asset Builder 2.5.0	19-01-2022
Version 1.9	Updated User Guide for EVE Asset Builder 2.6.1	20-09-2022
Version 2.0	Updated User Guide for EVE Asset Builder 2.7.1	22-11-2022
Version 2.1	Updated User guide for EVE Asset Builder 2.8.0	10-03-2023
Version 2.2	Updated User Guide for EVE Asset Builder 2.9.0	30-06-2023
Version 2.3	Updated User Guide for EVE Asset Builder 2.10.0	11-10-2023