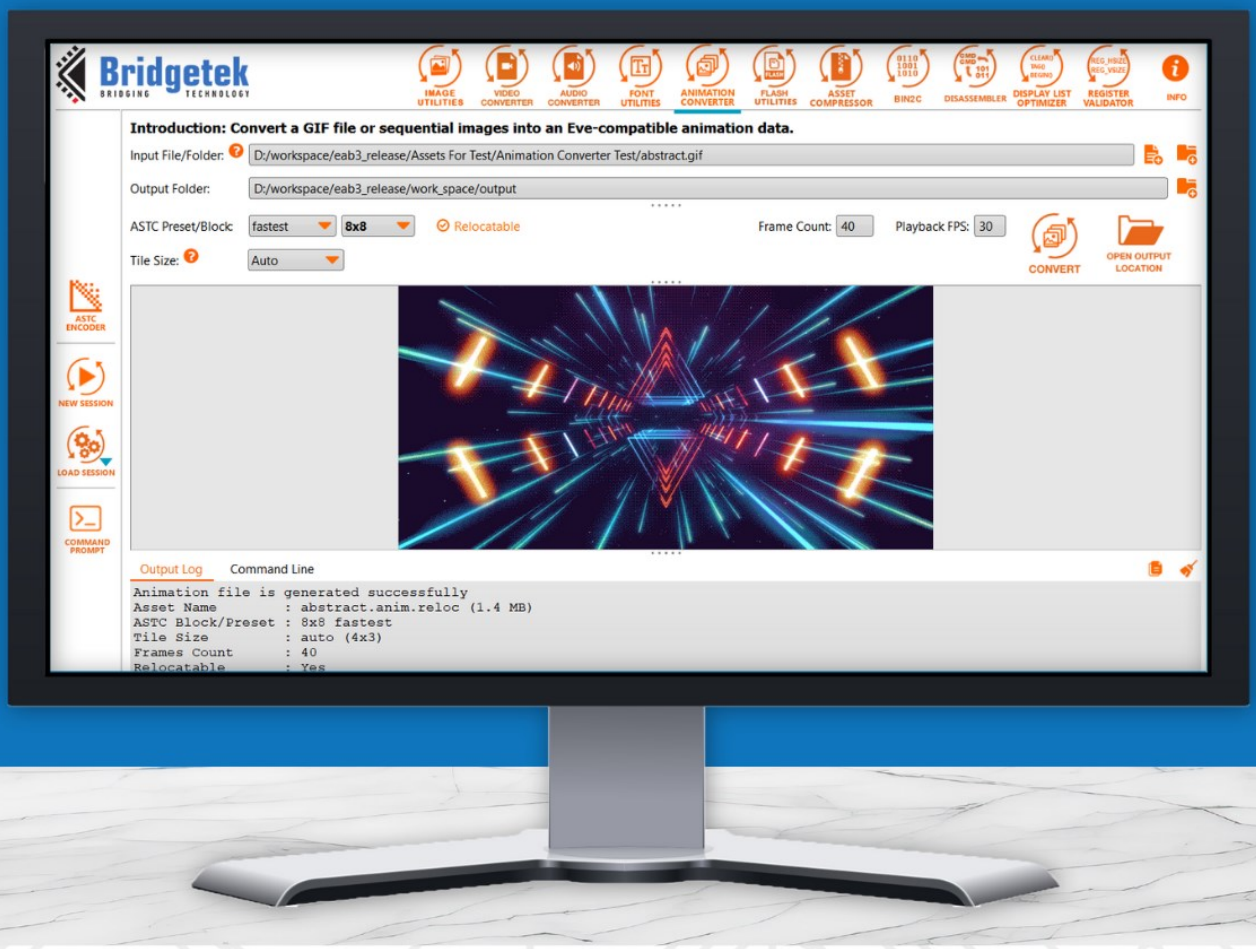
BT82X EVE ASSET BUILDER 3.0

USER GUIDE



DOC. VER. 1,0
ISSUE DATE: 21 MARCH 2025

Contents

I. Preface

A. Purpose

This document describes the features and procedures involved in using the **E**VE **A**sset **B**uilder (**EAB**).

B. Related Documents

- BT82X Series Programming Guide
- BT82X Datasheet

C. Acronyms and Abbreviations

Terms	Description
ASCII	American Standard Code for Information Interchange
ASTC	Adaptive Scalable Texture Compression
BLOB	Binary Large Object
BMP	Bitmap Image File
EAB	EVE Asset Builder
EDF	EVE Flash Description File
ESD	EVE Screen Designer
ESE	EVE Screen Editor
EVE	Embedded Video Engine
HAL	Hardware Abstraction Layer
MPSSE	Multi-Protocol Synchronous Serial Engine
MSVC	Microsoft Visual Studio C++
PNG	Portable Network Graphics
RDID	Read Device Identification
SPI	Serial Peripheral Interface
UI	User Interface
UTF	Unicode Transformation Format
WAV	Waveform Audio File Format

II. Overview

A. Introduction

The EVE Asset Builder (EAB) is a software application specifically designed for Windows operating systems. It provides a range of utilities to assist in generating resources for EVE devices. Starting from version 3.0, EAB supports BT82X and newer devices but does not maintain backward compatibility with BT81X/FT8XX. A separate version, 2.x, is available for these other devices. This document aims to provide a comprehensive guide on effectively using the tools within EAB.

B. Key Features

The following are the key features of EVE Asset Builder:

- Image Utilities: Convert bitmaps, validate PNG/JPEG.
- Video Converter: Convert a video to MJPEG format.
- Audio Converter: Convert an audio to EVE-compatible format.
- Font Utilities: Convert a font to Legacy/Extended1/Extended2 format.
- Animation Converter: Convert a gif file or a sequence of images to EVE-compatible format.
- Flash Utilities: Generate/Detect/Program/Update/Write/Erase/Read.
- Asset Compressor: Compress/Validate an asset to EVE-compatible format.
- Bin2C: Transform a binary file into a C array.
- Disassembler: Translate EVE instructions into human-readable text.
- Register Validator: Validate BT82X display configuration registers.

C. What's New?

Feature Updates:

Image Utility

- Add support for new bitmap formats for BT82X: LA1/2/4/8, ARGB8/6, RGB8/6, PALETTEDARGB8, YCBCR.
- Introduce option to specify Alpha channel when converting image to LA format.
- Add new feature "Raw Bitmap Viewer".
- Enhance the speed of rendering LA/L formats.
- Add a scroll bar to the preview pane to zoom in or navigate the input image.

Font Converter

- Enable a new Extended Font 2 format with added kerning functionality for BT82X.
- Implement a feature to toggle the new BIN/CJK bits on or off.

Register Validator

- Validate BT82X display configuration register values.

Disassembler

- Support BT82X specific display list and coprocessor commands.

Flash Utility

- Update the Blob structure.
- Modify flash operations to support NAND flash chips.
- Add functionality to select NAND or NOR Flash.
- Include a list of supported Flash part numbers for BT820.
- Introduce starting address and length options for flash operations.

Video Converter

- Support new audio stream formats: S16_SAMPLES (mono) and S16S_SAMPLES (stereo)
- Set default output resolution to 1920x1200.
- Set default pixel format to yuvj420p (full color range YUV 4:2:0).

Audio Converter

- Support BT82X specific audio formats: S16_SAMPLES (mono) and S16S_SAMPLES (stereo).

General

- Introduce the relocatable feature for the Animation and Font Converter.
- Integrate functionality allowing users to choose ASTC encoder.
- Ensure that all assets are 4-byte aligned.
- Update "Ask for overwrite" dialog to no longer prompt for the same file again.
- Improve Load and Save session.

D. Known Issues & Limitations

None

E. Credits

Open-Source Software

- Qt: *https://www.qt.io/*
- Python: *https://www.python.org/*
- ASTC Encoder: *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*
- FriBidi: *https://github.com/fribidi/fribidi*

Icons Copyright

Icons made by *Freepik*, *Smashicons*, *Pixel perfect*, *Dave Gandy*, *Royyan Wijaya*, *Icons8*

Audio Copyright

Audio for testing by *Lesfm* from *Pixabay*, and *mixkit*

Video Copyright

Video for testing by *vecteezy*

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
- ✓ 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 1300 x 900 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/toolchains/#EVEAssetBuilder*.

ii. Extract the zip file contents. Double click on the file **EVE-Asset-Builder-setup.exe**.

iii. The EVE Asset Builder Setup Wizard is displayed along with a Welcome message.

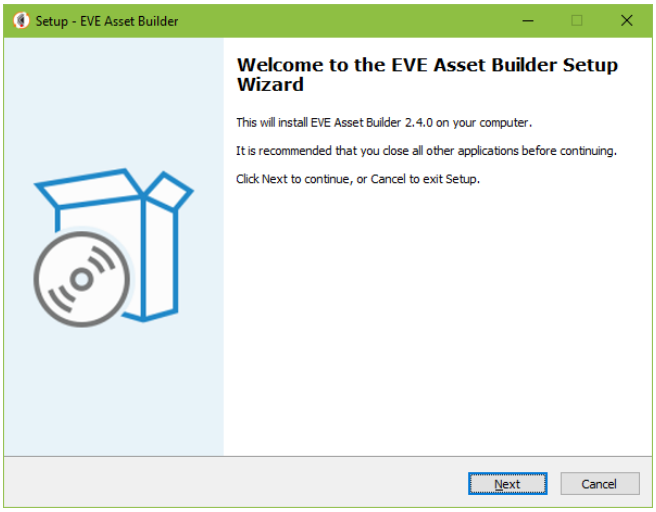


Figure 1 EVE Asset Builder Setup Wizard

iv. 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.

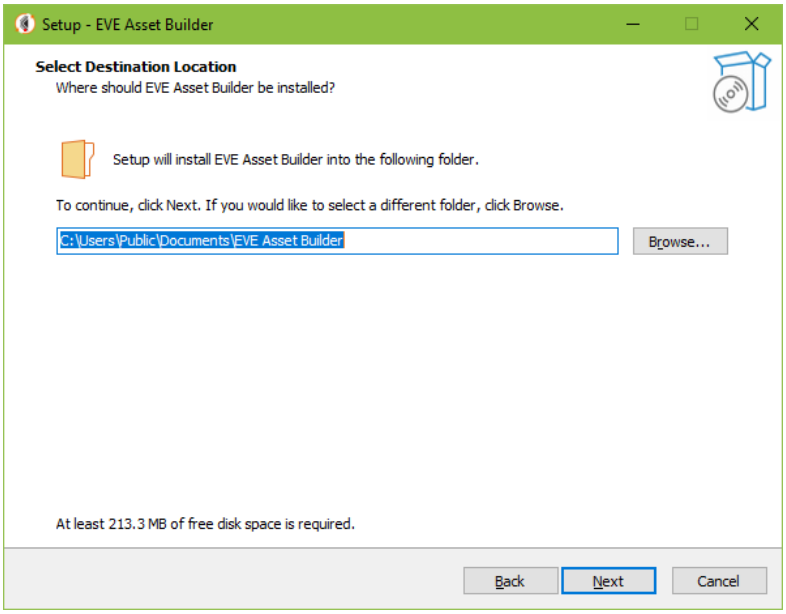


Figure 2 EVE Asset Builder Setup – Select the destination folder

v. In the **Select Additional Tasks** window, check the **"Create a desktop shortcut"** box, to have the EVE Asset Builder icon displayed on the desktop if required. Click **Next** to prepare for the installation.

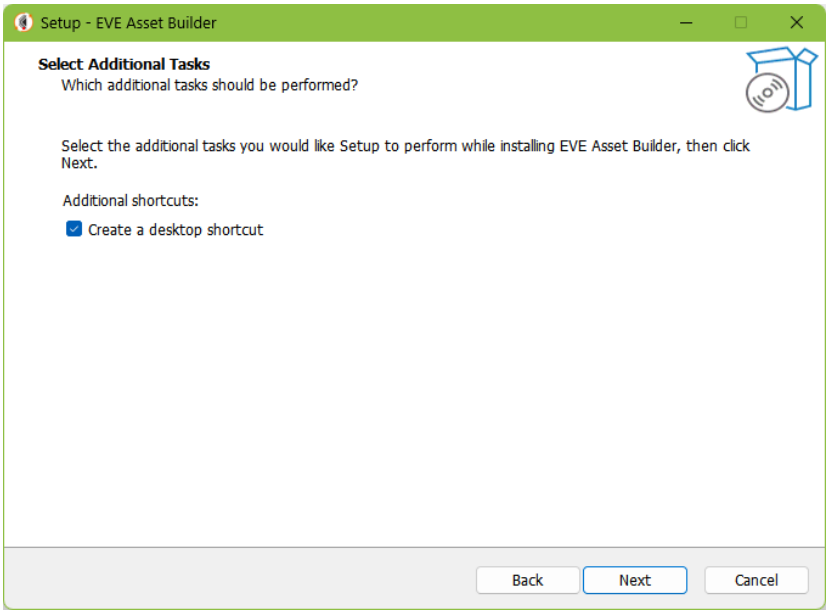


Figure 3 EVE Asset Builder Setup – Ready for Installation

vi. Click **Install** to start the installation.

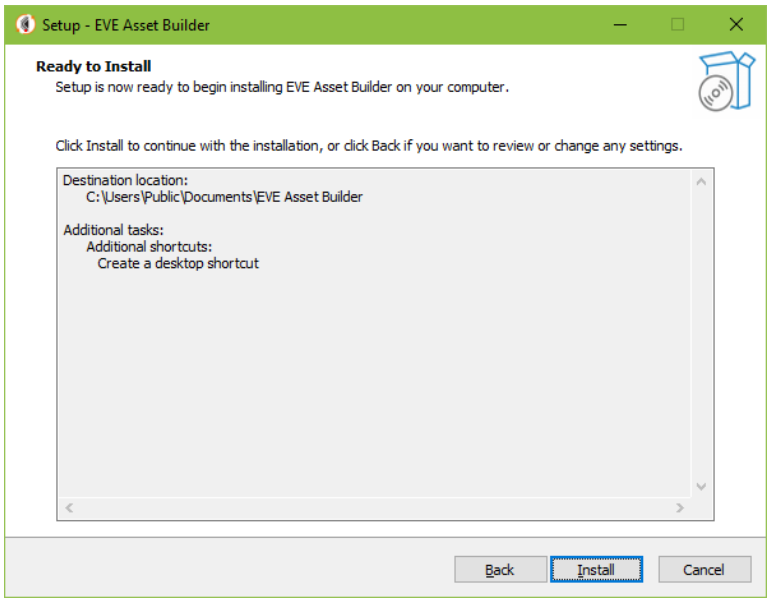


Figure 4 EVE Asset Builder Setup – Start Installation

vii. A progress bar indicates that the installation is in progress.

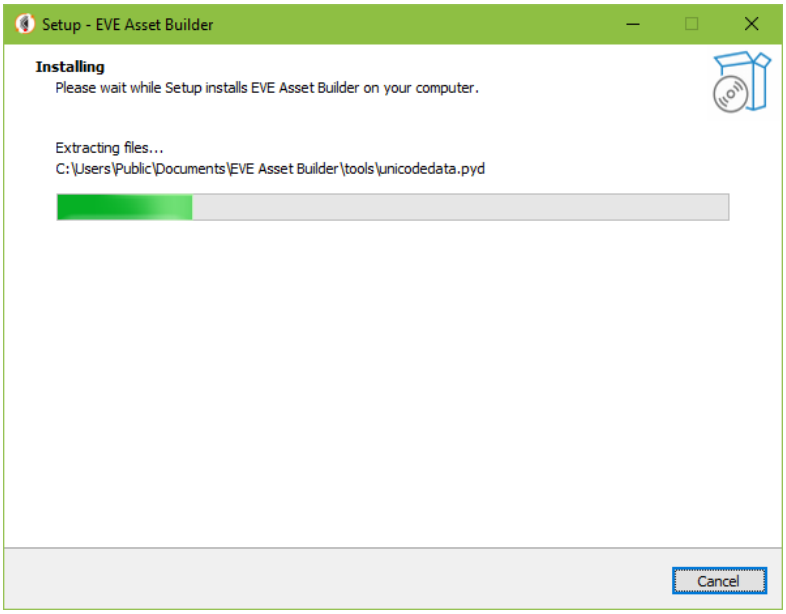


Figure 5 EVE Asset Builder Setup – Installing in Progress

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

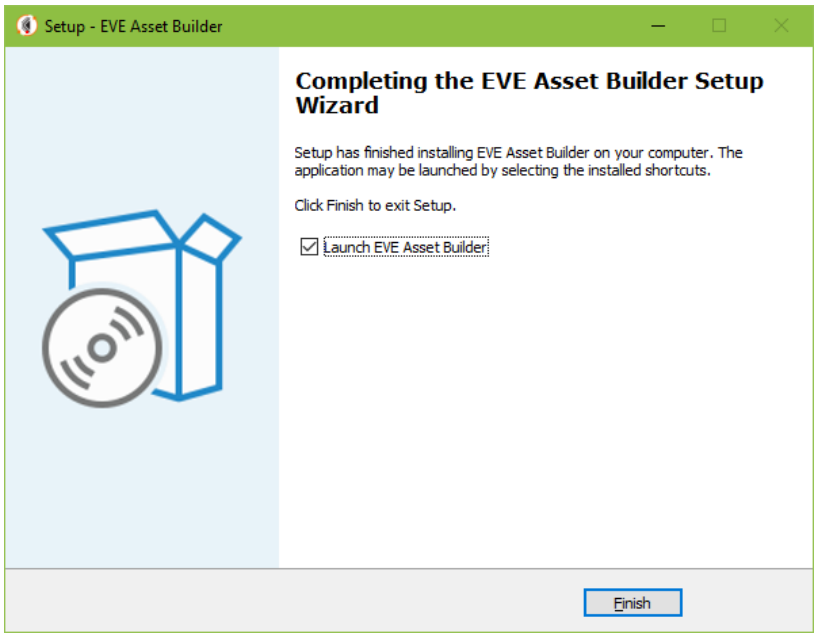


Figure 6 EVE Asset Builder Setup – Finish

IV. Getting Started

A. ASTC Encoder

Users can download various versions of ASTC encoders and choose the most suitable one for use with EAB.

The "Change ASTC Encoder" button is located on the left side panel. When pressed, a Select Folder Dialog will appear, allowing the user to choose a folder containing ASTC encoders with suffixes such as *sse2*, *sse4.1*, and *avx2*. EAB will subsequently read the CPU specifications and automatically select a compatible encoder. The default version employed in EAB is 4.5.

Note:

- ASTC format assets are incompatible with BT81X/FT8XX due to differences in bitmap layout.
- Official link to ASTC encoder releases: https://github.com/ARM-software/astc-encoder/releases

B. Relocatable Asset

The BT82X generation introduces a structure that enables assets with fixed addresses to become relocatable. Using the **cmd_loadasset** command, assets can now be loaded at any position within the RAM_G address space, eliminating the need for users to manage asset addresses manually. Currently, this functionality supports the relocation of Animation and Font assets.

Notes: For Font assets, the relocated address in RAM_G must align to multiples of 16, while Animation assets require alignment to multiples of 64.

C. Session

This functionality allows for the recording of user activities for future reference. On the left panel, two buttons help create or load sessions.

NEW SESSION: Start recording user actions, all data is stored in a session file.

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

Note: When EAB is first started, it creates a default session. On subsequent starts, it will load the most recent session.

1. New 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*

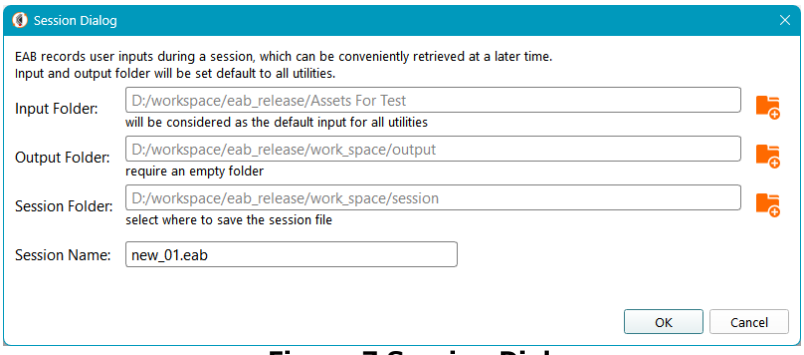


Figure 7 Session Dialog

2. Load Session

Load a previously saved session, restoring all user inputs.

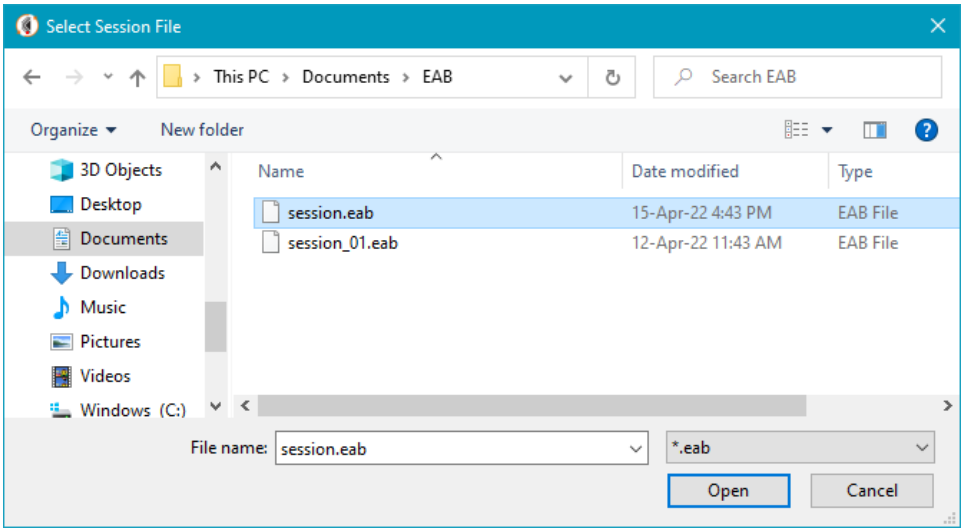


Figure 8 Select session file

D. 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 concern for where they are located.

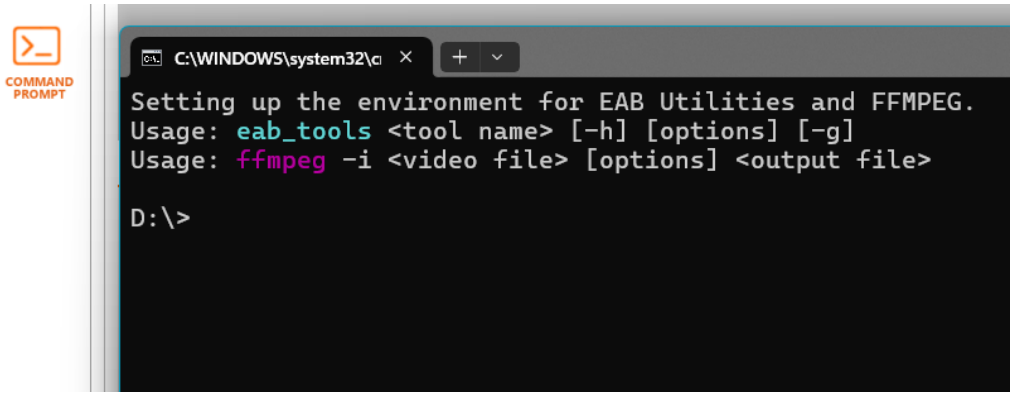


Figure 9 Command Prompt Window

Upon execution, every tool in the EVE Asset Builder 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.

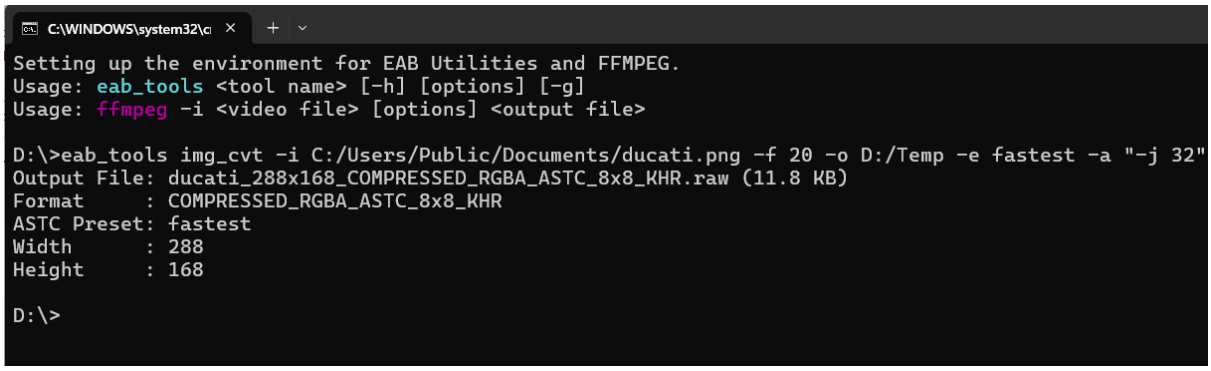


Figure 10 Example of converting an image

E. Log Window

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

Figure 11 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.

F. Naming Convention and Usage of Output Files

The output files generated by each tool in the EVE Asset Builder 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	*<input>_ <width>x<height>_ <output format>. <raw|rawh>*	lena.png -> lena_185x150_L8.raw, lena_185x150_L8.rawh	- Load the .raw or .rawh file to RAM_G - Decode it using cmd_setbitmap

	ASTC: Additional .astc file has been created, originating directly from the ASTC Encoder tools.	lena.png -> lena_185x150_ASTC_5x5.astc	- Specify the .astc file in Raw Pixels Viewer, then view it in the Preview pane.
Video Converter	<*input*>.avi	big_buck.avi -> big_buck.avi	- Load the output file to RAM_G - Decode it using cmd_playvideo
Audio Converter	<*input*>.raw	happy_summer.mp3 -> happy_summer.raw	- Load the .raw file to RAM_G - Play it by controlling the appropriate registers
	<*input*>.wav	happy_summer.mp3 -> happy_summer.wav	- Load the .wav file to RAM_G - Use cmd_loadwav or cmd_play wav to playback the audio
Font Converter	Legacy Format: <input>_<font size>_ <bitmap format>. <raw|rawh|reloc>	symbol.ttf -> symbol_25_L8.raw, symbol_25_L8.rawh, symbol_25_L8.reloc	**Relocatable:** - Use cmd_loadasset to load the .reloc file into RAM_G **Raw:** - Write the raw file to a predefined offset in RAM_G **Render:** - Decode it using cmd_setfont
	Extended Format: <*input*>_<font size>_ <bitmap format>. <raw|reloc>	arial.ttf -> arial_39_ASTC.raw, arial_39_ASTC.reloc	
Animation Converter	<*input*>.anim.<raw|reloc> Notes: Assets use **.reloc** as the suffix when relocatable, and **.raw** otherwise.	abstract.gif -> abstract.anim.raw, abstract.anim.reloc	**Relocatable:** - Use cmd_loadasset to load the .reloc file into RAM_G **Raw:** - Write the .anim.raw file to a predefined offset in RAM_G **Render:** - Use cmd_animstart, or cmd_animframe to play the animation
Flash Image Generator	<*output*>.bin <*output*>.map <*output*>.edf	abstract.anim.reloc, lena_ASTC_5x5.raw -> flash_16MB.bin,	- Write the .bin file to flash chip - See the offset and length of each asset in the map or the edf file - Use cmd_flashread to copy an asset to RAM_G, then use the appropriate commands to render it - Or, use cmd_flashsource to specify the asset's offset and render it with commands that support OPT_FLASH
		flash_16MB.map	Each line contains the asset information in the format: *name:offset:length*
		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 - Use cmd_inflate to decompress the assets
	Zopfli: <*output*>.zopfli	arial.xfont, big_buck.avi -> inflate.zopfli	
Bin2C Converter	<*input*>.c	img_argb2.bin -> img_argb2.c	Include the C file to use the converted C-array
Disassembler	<*input*>.da.txt	bt82x_dump.bin -> bt82x_dump.da.txt	Open .da.txt file to see the human -readable EVE commands
All Utilities	<*input*>.json	abstract.gif -> abstract.anim.json	A .json file contains the meta information of converted asset. It can be opened by any text editor. ESE or ESD will use it to load the assets into their projects too.

Table 1 Naming convention and usage of output files

V. Utilities

A. Image Utilities

This utility provides various image-related features. The first tab focuses on converting PNG, JPEG, and BMP images to a format compatible with EVE. The second tab allows you to visualize the resulting raw output.

1. Image Converter

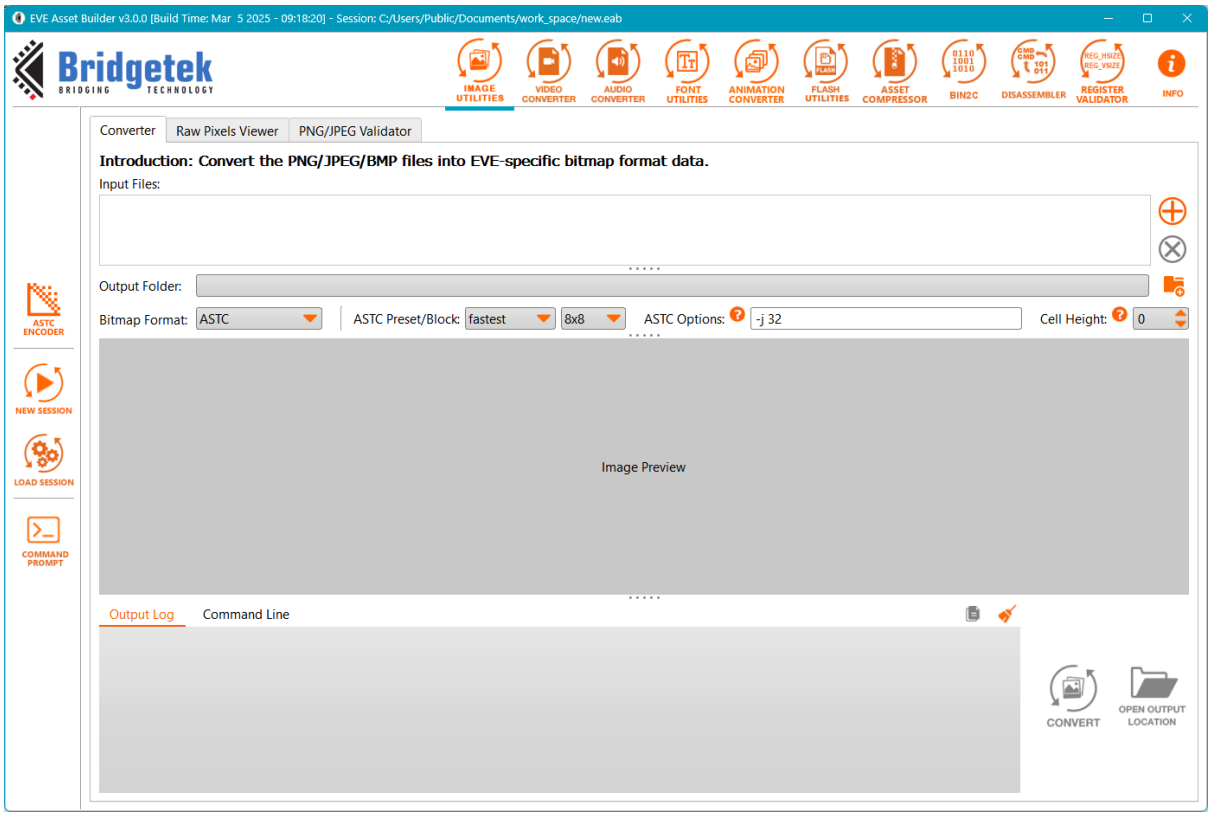


Figure 12 Image Converter

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

Output Folder: The folder contains the converted files.

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

- ARGB1555, ARGB2, ARGB4, ARGB6, ARGB8
- RGB332, RGB565, RGB6, RGB8
- L1, L2, L4, L8
- LA1, LA2, LA4, LA8
- PALETTEDARGB8
- ASTC
- YCBCR

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

ASTC Block: Choose one from the following options: 4x4, 5x4, 5x5, 6x5, 6x6, 8x5, 8x6, 8x8, 10x5, 10x6, 10x8, 10x10, 12x10, 12x12.

ASTC Options: Users can explore various options to determine the optimal balance for their specific needs, considering factors like memory availability, performance demands, and visual quality.

Cell Height: Include padding data to ensure that each bitmap cell aligns to 64 bytes in memory, enabling proper rendering in the ASTC format on the EVE chip. Specifying a value greater than 0 will enable this feature. EAB will adjust the Cell Height value to align with the ASTC block size.

Dithering: Dithering Enabling dithering can help reduce noise levels, but it may negatively impact compression efficiency. The bitmap formats ASTC, YCBCR, and PALETTEDARGB8 do not support dithering.

Alpha (%): Ranges from 0 to 100. A value of 0 means fully transparent, while 100 means fully opaque. Users can draw a rectangle in the Image Preview by holding Ctrl and using the left mouse button to define the area where the alpha effect applies. This option is available in bitmap formats LA1, LA2, LA4, and LA8.

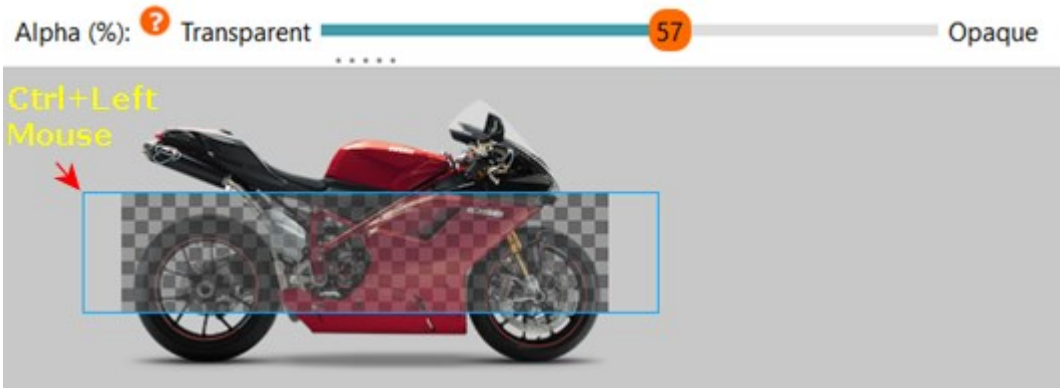


Figure 13 Image Converter - Alpha

2. Raw Pixels Viewer

This tool attempts to visualize the raw bitmap data using the provided bitmap format and image size.

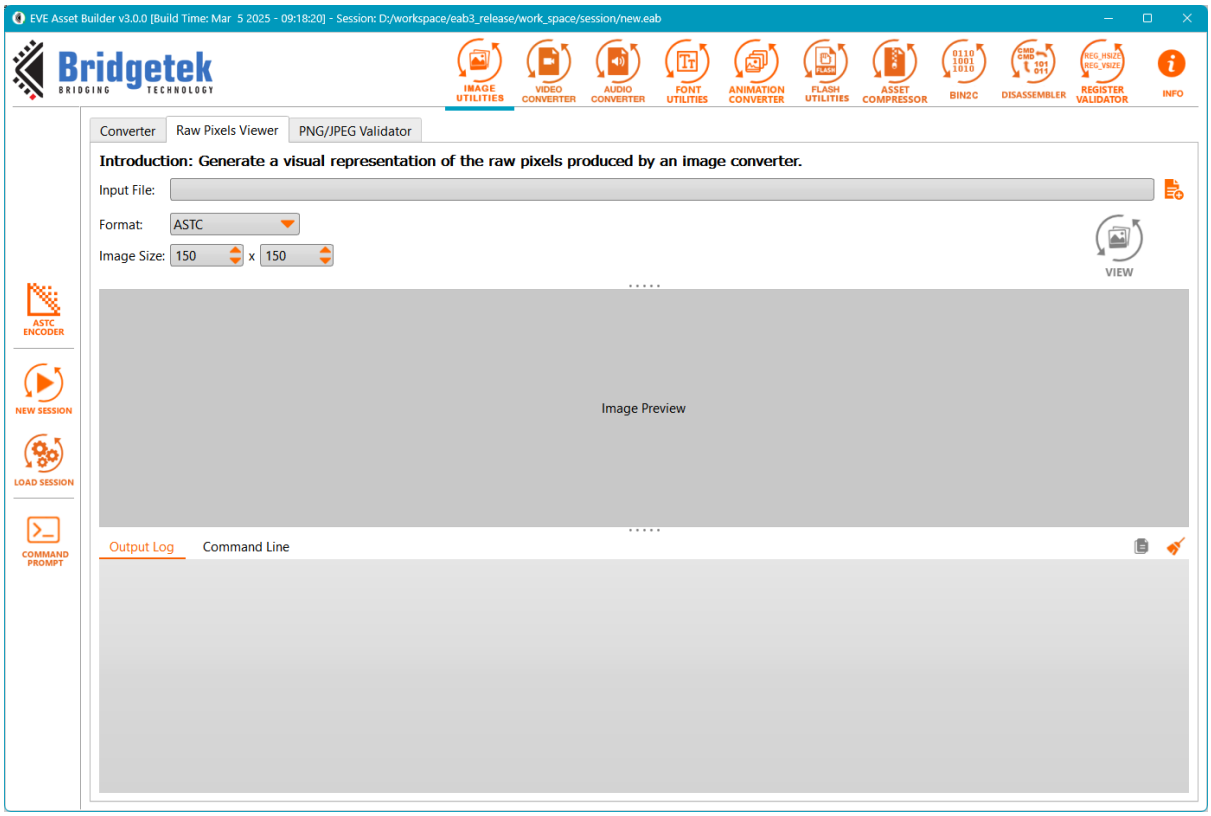


Figure 14 Raw Pixels Viewer

Input File: Pick the .raw file to view. For ASTC format, select the .astc file. For PALETTEDARGB8, select the .index.raw.

Format: See *Bitmap Format*.

Image Size: Set the image width and height.

VIEW: Render the raw bitmap data. The resulting image appears in the Image Preview panel.

Notes: If EAB finds the corresponding json file, it will retrieve the bitmap format and image size, and update the UI accordingly.

3. PNG/JPEG Validator

This tool helps validate PNG or JPEG images to determine if they can be successfully loaded by ***cmd_loadimage***. It also provides an option to optimize incompatible images using third-party tools, *optipng* and *jpegtran*, to generate EVE-compatible images.

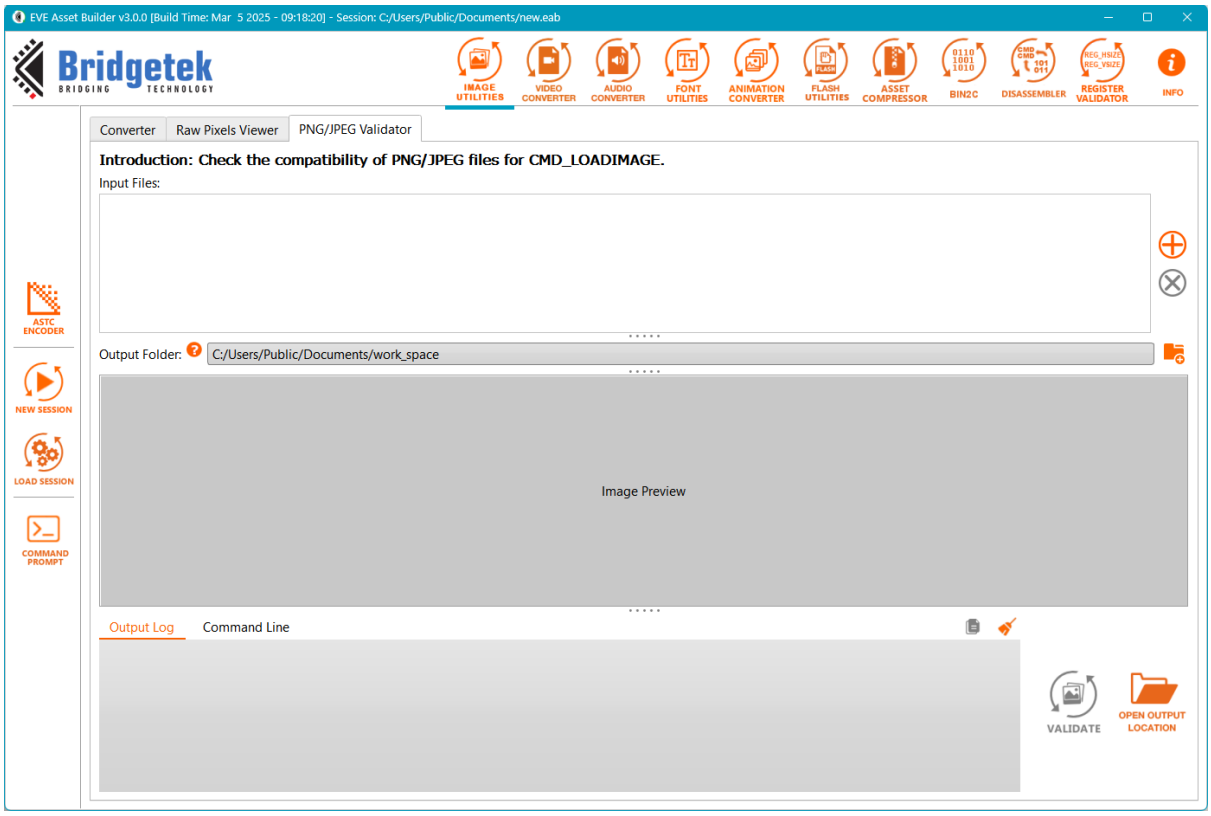


Figure 15 PNG/JPEG Validator

Input File: Select PNG or JPEG images for validation.

Output Folder: The optimized image will be saved here, unless all input images are compatible.

Image Preview: Display the selected image.

VALIDATE: Use the BT82X emulator to load images via the ***cmd_loadimage*** command. Consider the load successful if there is no coprocessor fault or if RAM_CMD is not stuck. This tool will prompt for optimizing incompatible images and then validate them again.

Notes:

- PNG: Adam-7 interlaced images are not supported. Only bit-depth 8 is allowed; bit-depths 1, 2, 4, and 16 are not supported.
- JPEG: Only baseline JPEGs with YUV formats 444, 422, or 420 are supported by cmd_loadimage. Progressive JPEGs or other YUV formats are not compatible.
- This tool relies on the BT82X emulator to validate PNG/JPEG images. Users may see different results on the hardware.

B. Video Converter

This tool converts a video file into the MJPEG format and packages it within an AVI container. The resulting .avi file will be compatible with EVE devices. Users can also play it on personal computers to quickly verify if the converted file contains the desired video and audio. The audio stream format can be either mono or stereo.

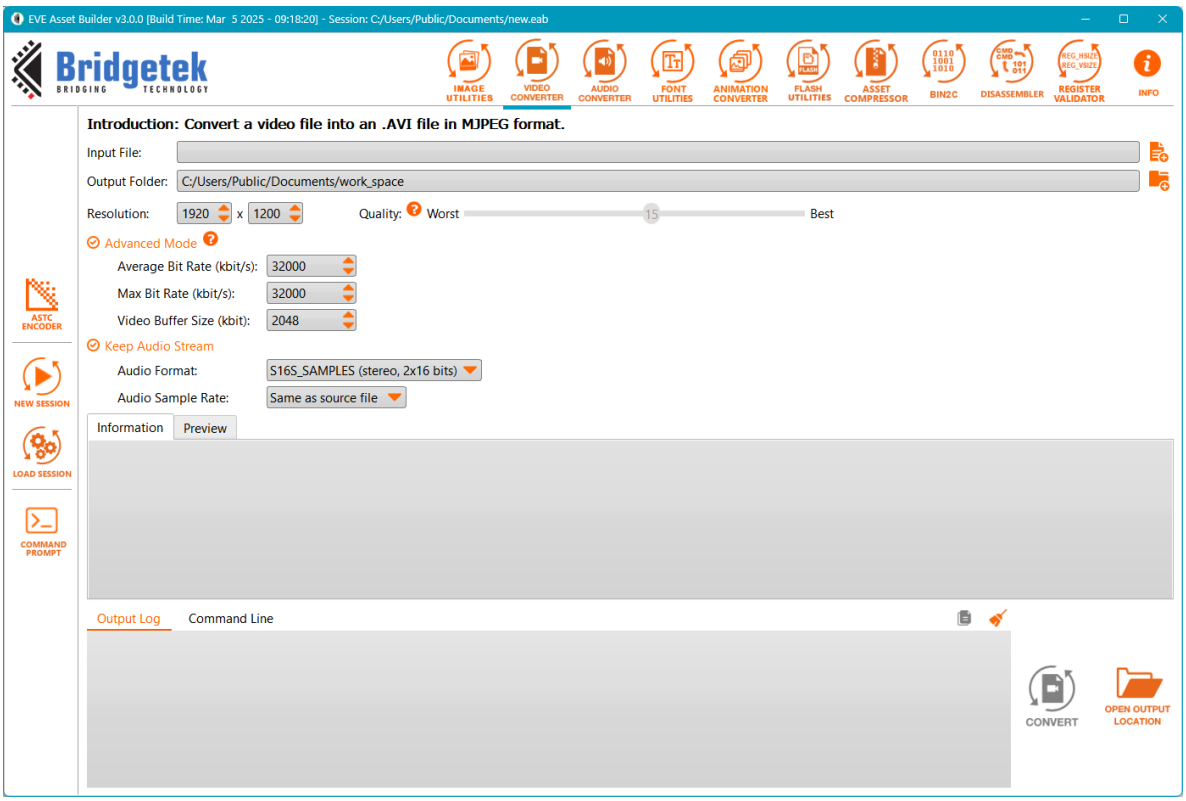


Figure 16 Video Converter

Input: The original video will be converted.

Output Folder: The folder contains converted files.

Resolution: Must be specified, the video will be converted to this resolution.

Quality: From 1 (best) to 31 (worst).

Advanced Mode: Allow users to control the output bitrate. Enabling this mode will override the Quality setting, and vice versa.

- **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 Size (kbit):** Set buffer size for the output video

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

Audio Sample Rate: Choose the sample rate for the output audio stream, either keeping it the same as the input video (default) or selecting any value from the provided list. They are: 8000Hz, 11025Hz, 16000 Hz, 22050 Hz, 32000 Hz, 32780 Hz, 44056 Hz, 44100 Hz, and 48000 Hz.

Audio Format: LINEAR_SAMPLES, ULAW_SAMPLES, ADPCM_SAMPLES, S16_SAMPLES, S16S_SAMPLES. Only S16S_SAMPLES supports stereo mode, while the others are mono.

C. Audio Converter

Convert a WAV or MP3 audio file into a format optimized for EVE devices, ensuring seamless processing and playback. The audio stream format can be either mono or stereo.

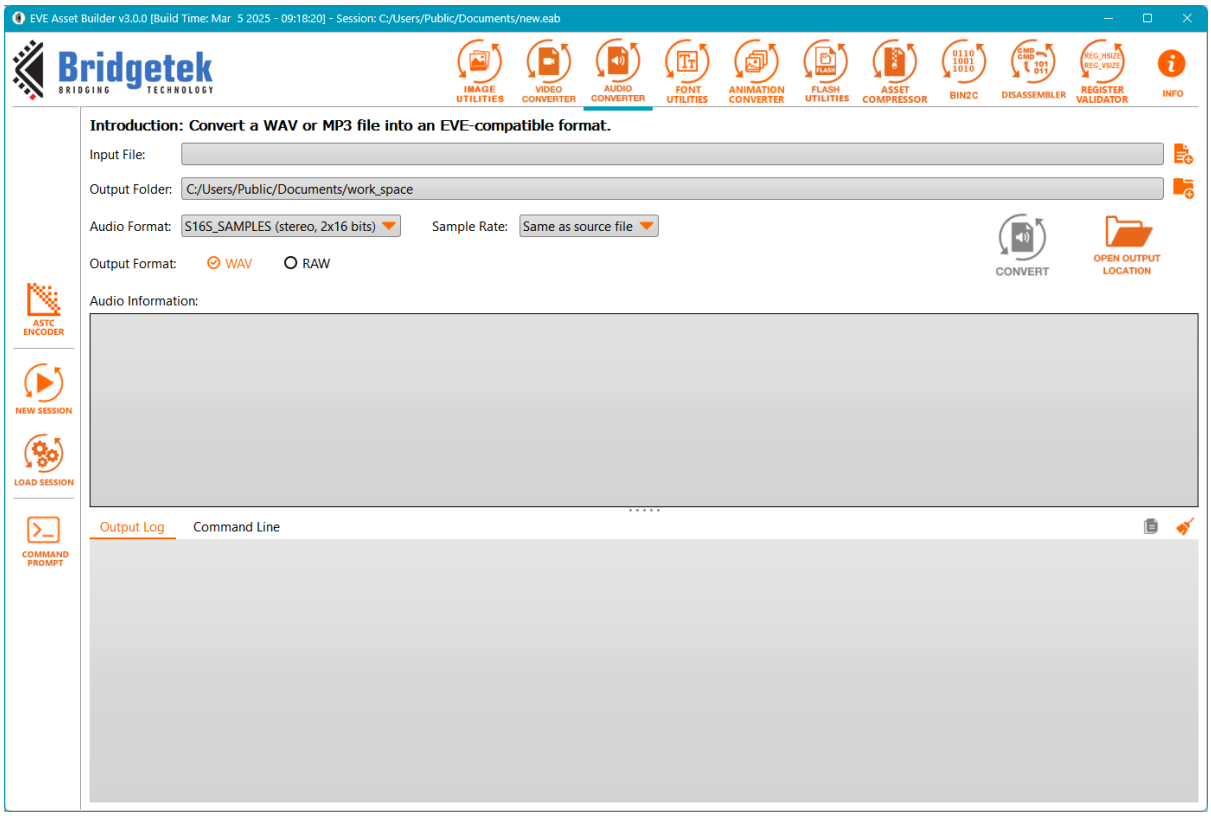


Figure 17 Audio Converter

Input File: The audio file will be converted.

Output folder: The folder contains the converted files.

Audio Format: LINEAR_SAMPLES, ULAW_SAMPLES, ADPCM_SAMPLES, S16_SAMPLES, S16S_SAMPLES. Only S16S_SAMPLES supports stereo mode, while the others are mono.

Audio Sample Rate: Users can retain the input sample rate or select a different value from the drop-down list.

Output Format: WAV or RAW.

D. Font Utilities

1. Font Converter

Extract characters from the OpenType or TrueType font file into a specific font structure. Unprintable characters will be filtered out. Since each glyph is converted to a bitmap individually, combination characters are not supported.

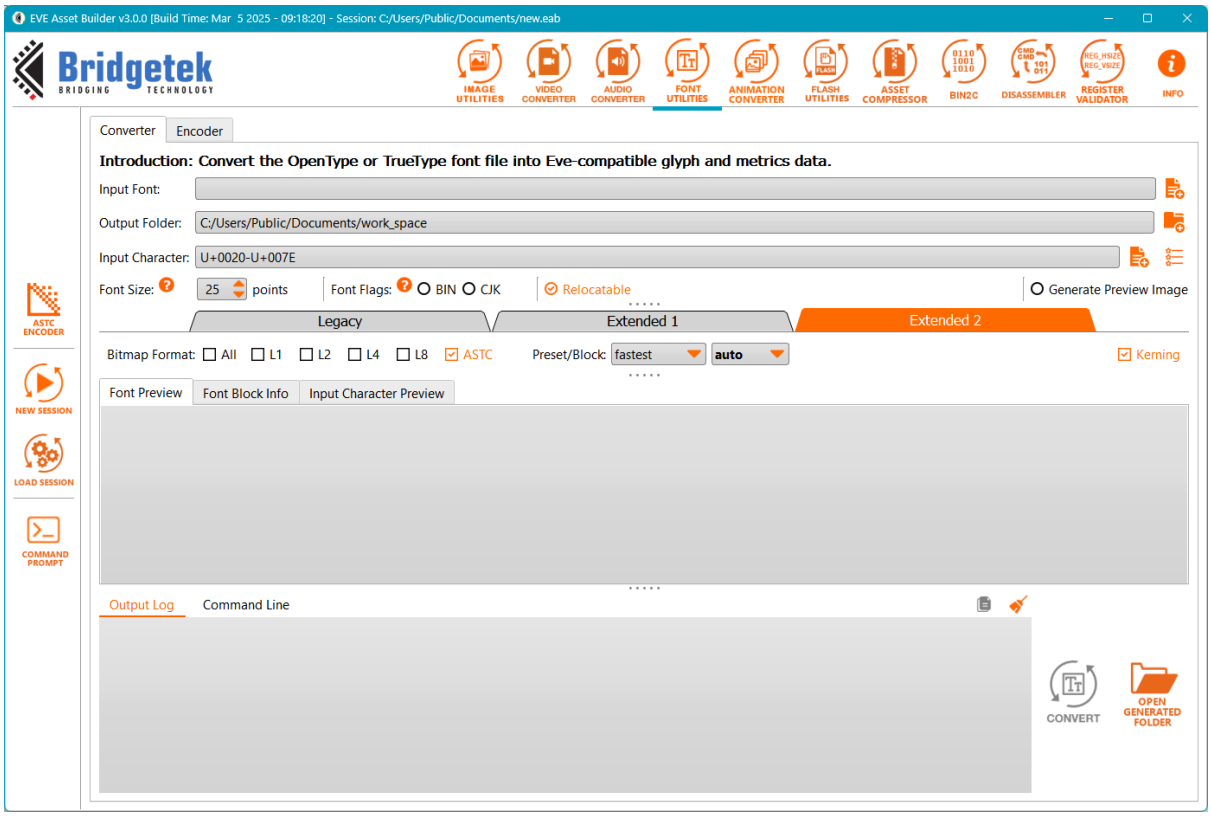


Figure 18 Font Converter tab

Input Font File: Font file to be converted.

Output Folder: The folder contains converted files.

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

Font Size: Specify the intended font size. Any character exceeding 255 pixels in width will be eliminated.

Font Flags:

BIN: If set, the font is binary, disabling newline interpretation of the font string
CJK: If set, then OPT_FILL breaks lines on any character, instead of at word breaks.

Relocatable: Allow the converted asset to be loaded at any address within RAM_G with full flexibility. The address should be a multiple of 16.

Address: The converted font will be loaded to the offset address in RAM_G, which must be a multiple of 4. This option is available when **Relocatable** is unchecked.

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.

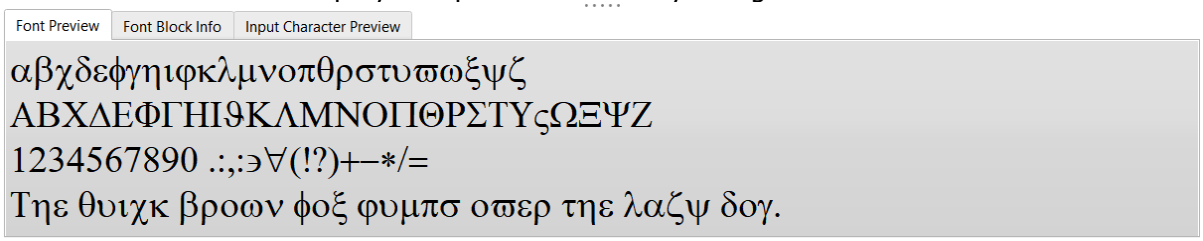


Figure 19 Font Preview Area

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.

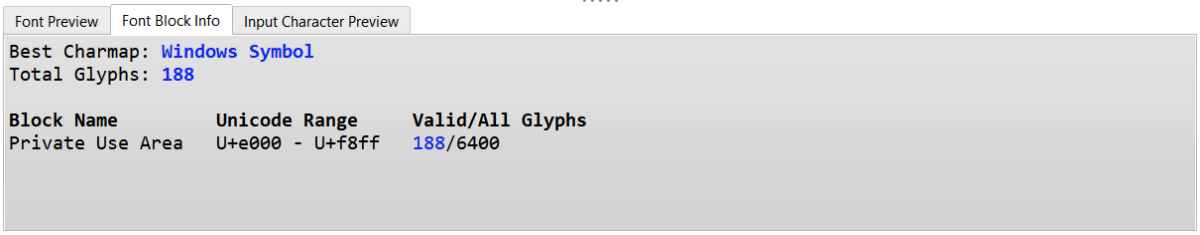


Figure 20 Font Block Info

Input Character Preview:

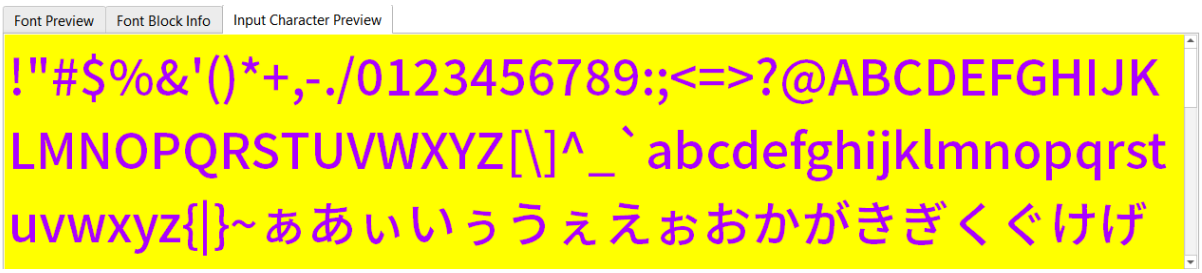


Figure 21 Input Character Preview

Legacy Format

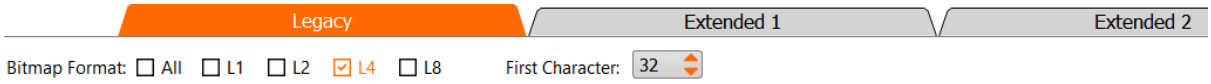


Figure 22 Legacy Format

Bitmap Format: Supported bitmap formats are L1, L2, L4, and L8. Each selected format will be converted separately in a single operation.

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

Output:

For each bitmap format, the tool generates:

- A .raw file includes the font metrics block and graphic data, or
- A .reloc file contains the converted font in a relocatable format.

Extended Format 1

Handle fonts with a full range of Unicode code points.

Figure 23 Extended Format 1

Bitmap Format: Supported bitmap formats are ASTC, L1, L2, L4, and L8. Each selected format will be converted separately in a single operation.

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

ASTC Block: auto, 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

Output:

For each bitmap format, the tool generates:

- A .raw file includes the font metrics block and graphic data, or
- A .reloc file contains the converted font in a relocatable format.
- Sample C code is provided to demonstrate its usage.

Extended Format 2

This format supports all Unicode code points and enables the kerning feature. Users can freely turn the kerning feature on or off. The font structure is entirely different from Extend Format 1.

Figure 24 Extended Format 2

Kerning: Enable kerning support for fonts with a GPOS table.

Refer to *Extended Format 1* for other settings.

2. Text Encoder

The Text Encoder serves as a companion utility to the font converter. It encodes user input text using either UTF-8 or ordinal code points, as determined by the .json file generated by the font converter. It aids programmers in seamlessly working with EVE coprocessor commands.

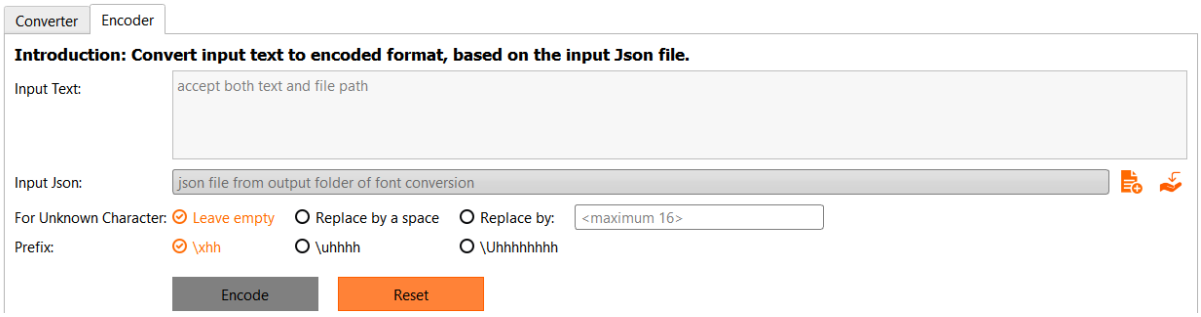


Figure 25 Encoder tab

Input Text: The character set to encode.

Input Json: The file which determines what code points the input text maps to.

For Unknown Character: Options on how to process with unknown characters.

Prefix: The prefix of the code point.

One example is as below:

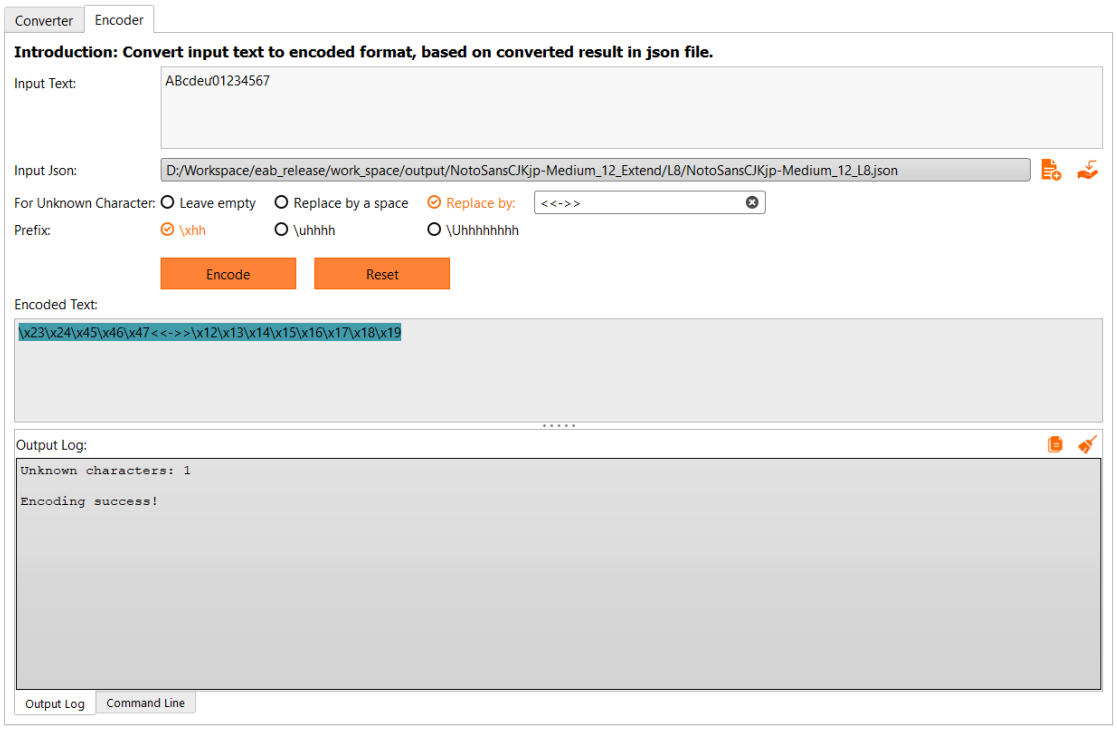


Figure 26 Text Encoder Example

E. Animation Converter

Convert a **GIF** file or a series of **PNG/JPEG/BMP** files into EVE-compatible animation files.

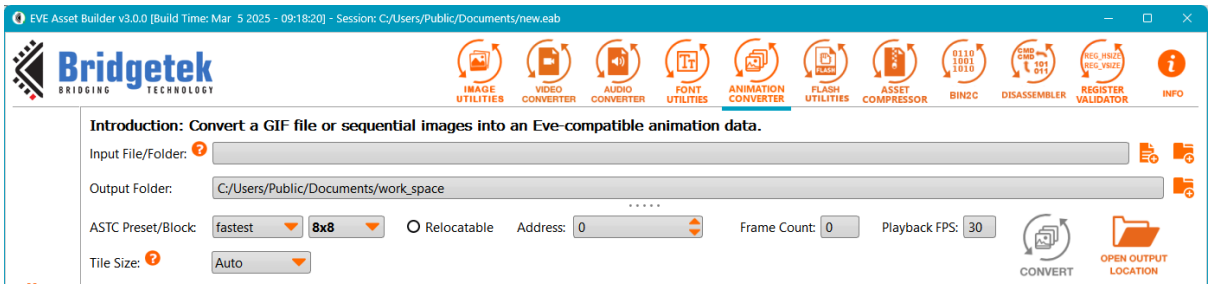


Figure 27 Animation Converter

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 regular expression *".*[0-9]+\.(Json)"* defined. For example: "001.png," "002.png."

Output Folder: The folder contains the converted files.

Relocatable: Allow the converted asset to be loaded at any address within RAM_G with full flexibility. The address should be a multiple of 16.

Address: The converted animation will be loaded to the offset address in RAM_G, which must be a multiple of 64. This option is available when **Relocatable** is unchecked.

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

ASTC Block: Choose one from the following options: 4x4, 5x4, 5x5, 6x5, 6x6, 8x5, 8x6, 8x8, 10x5, 10x6, 10x8, 10x10, 12x10, 12x12.

Frame Count: Display the number of available frames.

Playback FPS: For previewing the speed of animation.

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

- Auto: The optimal tile size will be automatically chosen by EAB.
- Manual: The user can manually choose both the Tile Width and Tile Height.
  Note: The product of *Tile Width* and *Tile Height* must be a multiple of 4, in compliance with the rendering constraints of ASTC images.
- Disabled: Exclude the application of tiles in the conversion process.

Output

- An *.anim.raw* file that can be directly written to flash and then used to render animations, or
- An *.anim.reloc* file in relocatable format, generated if the **Relocatable** option is checked.
- A *SampleApp* folder illustrates how to display an animation.

F. Flash Utilities

1. Flash Image Generator

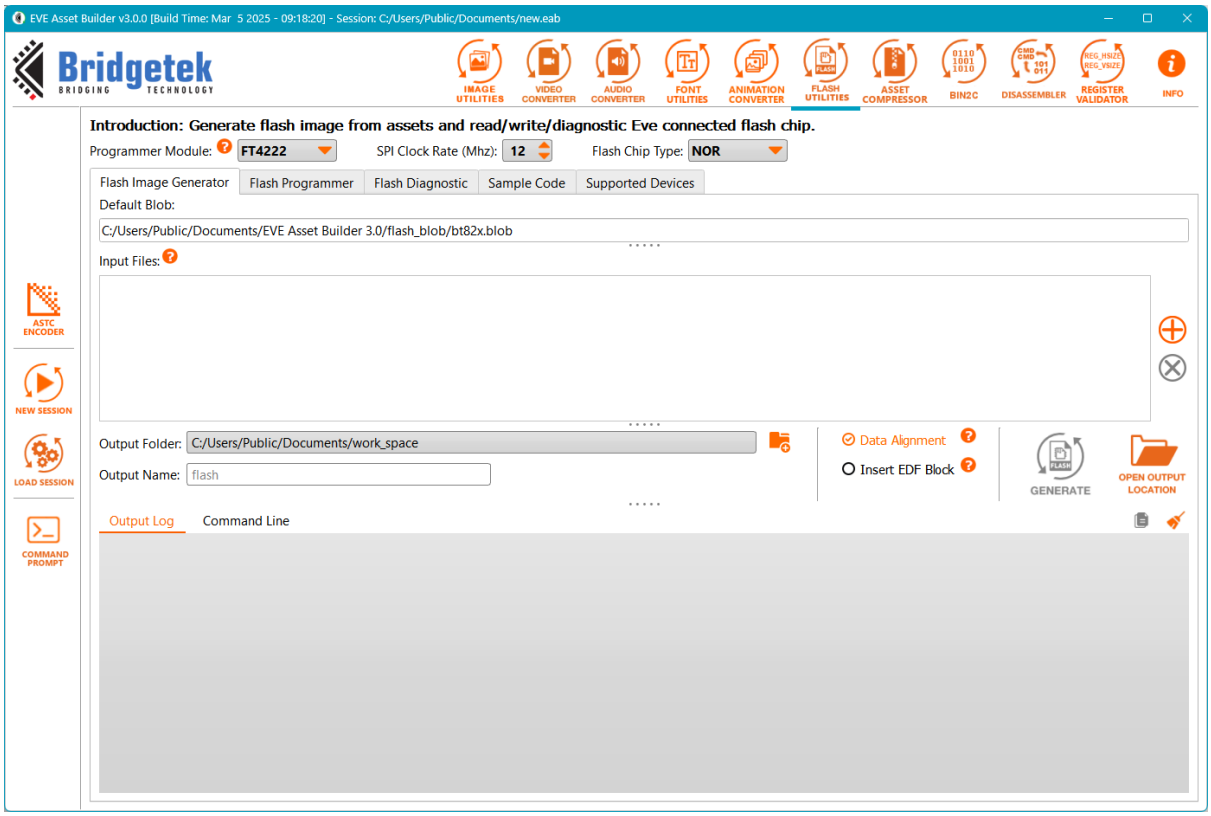


Figure 28 Generate Flash Image

Default Blob: The 4096-byte flash driver used by the firmware, always located in the first block.

Input Files: Select the assets for flash content.

Data Alignment: By default, each asset is aligned to 64 bytes, and the entire flash file is aligned to 4096 bytes. If left unchecked, alignment is not applied, and users should have a strong reason for doing so.

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

Note:

- To modify the properties of an asset, users must place the corresponding JSON file in the same folder.

Output Folder: Folder containing the generated files.

Output Name: Set the generated flash image name.

Output:

- A .bin file to write to the flash chip,
- A .map file keeps the offset and size of each asset as the format: ***<asset name> : <offset> : <size>***,
- An .edf file has the details of each assets, as described in *EDF Block*.

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 {
    uint16_t numAsset;      // total number of assets
    uint16_t edfSize;       // size of EVE_Flash_Asset_Info
    EVE_Flash_Asset_Info assets[0... numAsset -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 asset size, in byte
    uint8_t  compression;               // 0 = raw, 1 = compress, 2 = relocatable
    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	*Table 3*	Bitmap
0x03	-	Bitmap PALETTED
0x05	-	PNG/JPEG File
0x10	-	Legacy Font
0x11	-	Extended Font 1
0x12	-	Extended Font 2
0x20	-	Animation
0x30	*Table 4*	Audio
0x50		Video
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
17	L2
19	RGB8
20	ARGB8
21	PALETTEDARGB8
22	RGB6
23	ARGB6
24	LA1
25	LA2
26	LA4
27	LA8
28	YCBCR
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
3	S16 Sample Format
4	S16S Sample Format (stereo)

Table 4 subType of Audio

2. Flash Programmer

Detect Flash

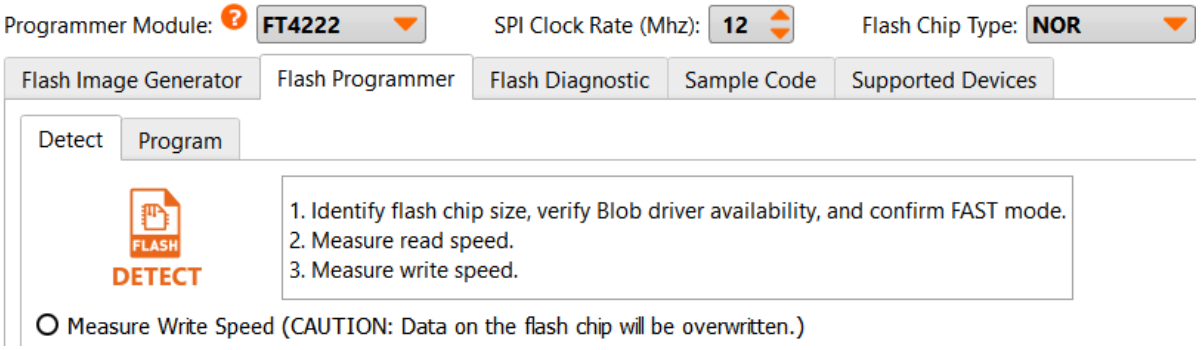


Figure 29 Detect Flash

Programmer Module: 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. At any given time, only one Programmer Module can be utilized to execute a flash operation.

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

Flash Chip Type: NOR or NAND.

Measure Write 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.

DETECT: Detect flash information.

Program Flash

Figure 30 Program Flash – NOR

Figure 31 Program Flash – NAND

Flash Image: This file will be utilized to program, update, write data to the flash chip, or verify the flash content.

Destination: Must be aligned as described below and within the range of flash chip size.

- PROGRAM: Must be 4096-byte aligned.
- UPDATE: Must be 4096-byte aligned.
- WRITE: Must be 256-byte aligned.

WRITE NAND: Erase the NAND flash chip, then write the flash image.

PROGRAM: Write data to the flash chip in its factory state (all bits set to 1). This operation is two or three times faster, as it does not compare the data before writing.

UPDATE: Updates the flash chip, erasing if necessary. The flash is not completely cleared; only the required partitions are updated.

UPDATE & VERIFY: Performs the update and then compares the updated data with the source data.

VERIFY: Compare the input binary file with flash content.

ERASE: Clears all data from the flash chip.

Read Flash

Read the content of flash chip into .bin file.

Figure 32 Read Flash

Output Folder: Select a folder to save the output file.

Output Name: Select the name for the output file.

Start: The offset indicating where to begin reading.

Length: The number of bytes to be read.

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

3. Flash Diagnostic

The purpose of this feature is to troubleshoot the flash chip.

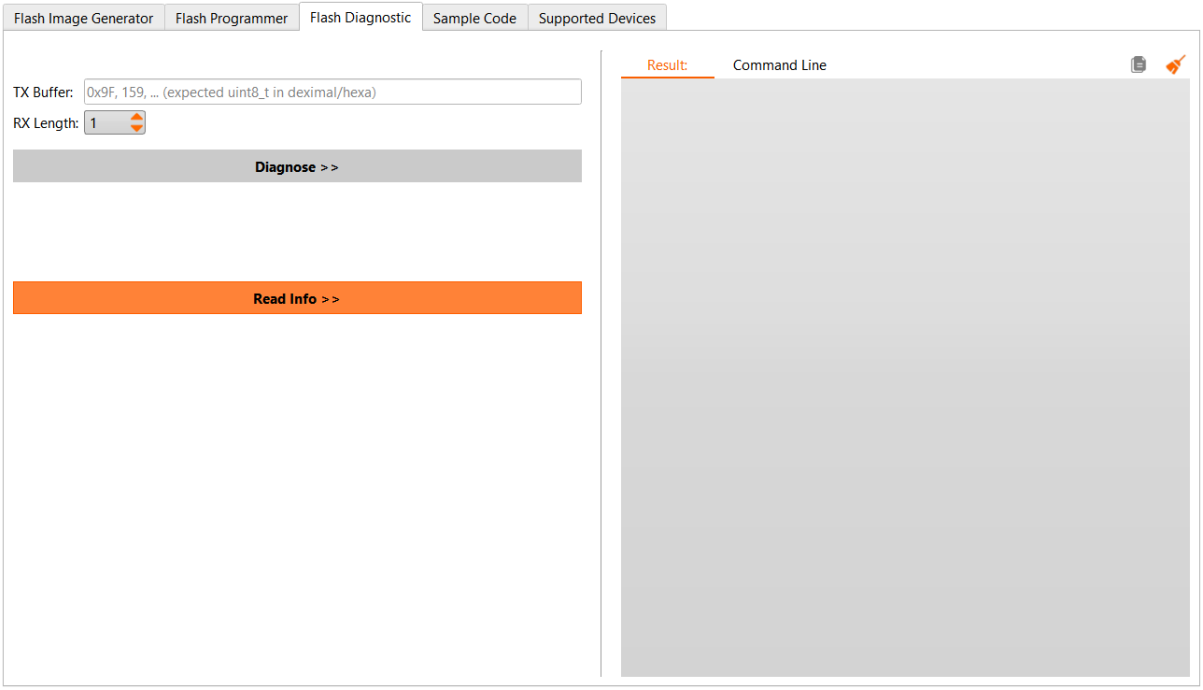


Figure 33 Flash Diagnosis

Diagnose

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: Refer to the flash chip's manual for details on supported commands and expected data lengths.

Read Info

This will report:

1. Flash RDID
2. Flash status
3. Flash size (in Mbytes and Mbits)

4. Sample Code

The pseudo codes are provided here to showcase the procedure of interacting the EVE connected flash.

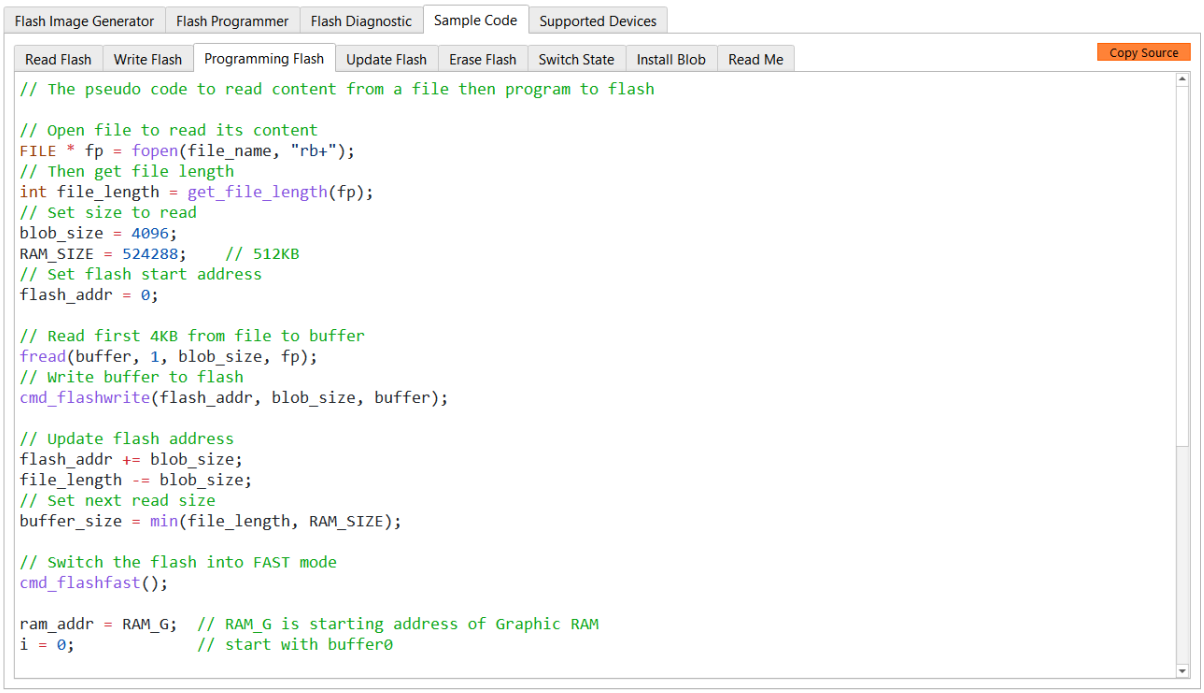
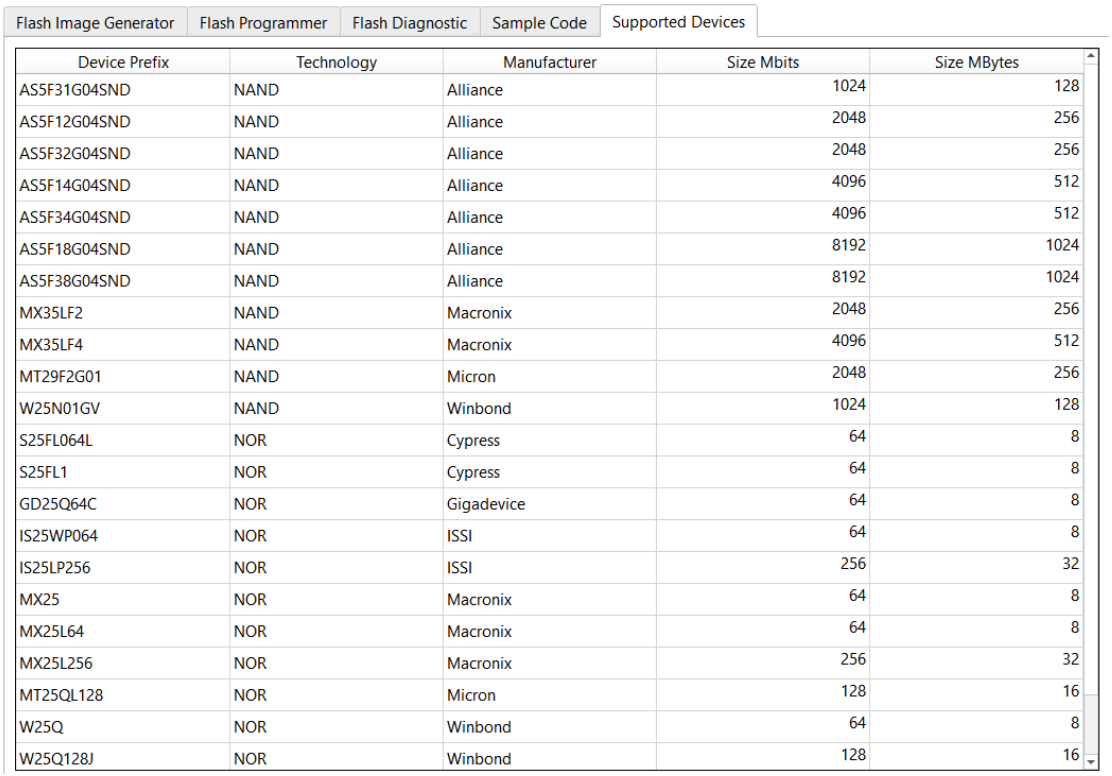


Figure 34 Flash Sample Code

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

5. Supported Devices

The table below contains all the flash chips supported by BT82X, offering users the ability to check essential information such as the NAND/NOR type and flash chip size.



Flash Image Generator | Flash Programmer | Flash Diagnostic | Sample Code | Supported Devices

Device Prefix	Technology	Manufacturer	Size Mbits	Size MBytes
AS5F31G04SND	NAND	Alliance	1024	128
AS5F12G04SND	NAND	Alliance	2048	256
AS5F32G04SND	NAND	Alliance	2048	256
AS5F14G04SND	NAND	Alliance	4096	512
AS5F34G04SND	NAND	Alliance	4096	512
AS5F18G04SND	NAND	Alliance	8192	1024
AS5F38G04SND	NAND	Alliance	8192	1024
MX35LF2	NAND	Macronix	2048	256
MX35LF4	NAND	Macronix	4096	512
MT29F2G01	NAND	Micron	2048	256
W25N01GV	NAND	Winbond	1024	128
S25FL064L	NOR	Cypress	64	8
S25FL1	NOR	Cypress	64	8
GD25Q64C	NOR	Gigadevice	64	8
IS25WP064	NOR	ISSI	64	8
IS25LP256	NOR	ISSI	256	32
MX25	NOR	Macronix	64	8
MX25L64	NOR	Macronix	64	8
MX25L256	NOR	Macronix	256	32
MT25QL128	NOR	Micron	128	16
W25Q	NOR	Winbond	64	8
W25Q128J	NOR	Winbond	128	16

Figure 35 Supported Flash Chip

G. Asset Compressor

1. Compressor

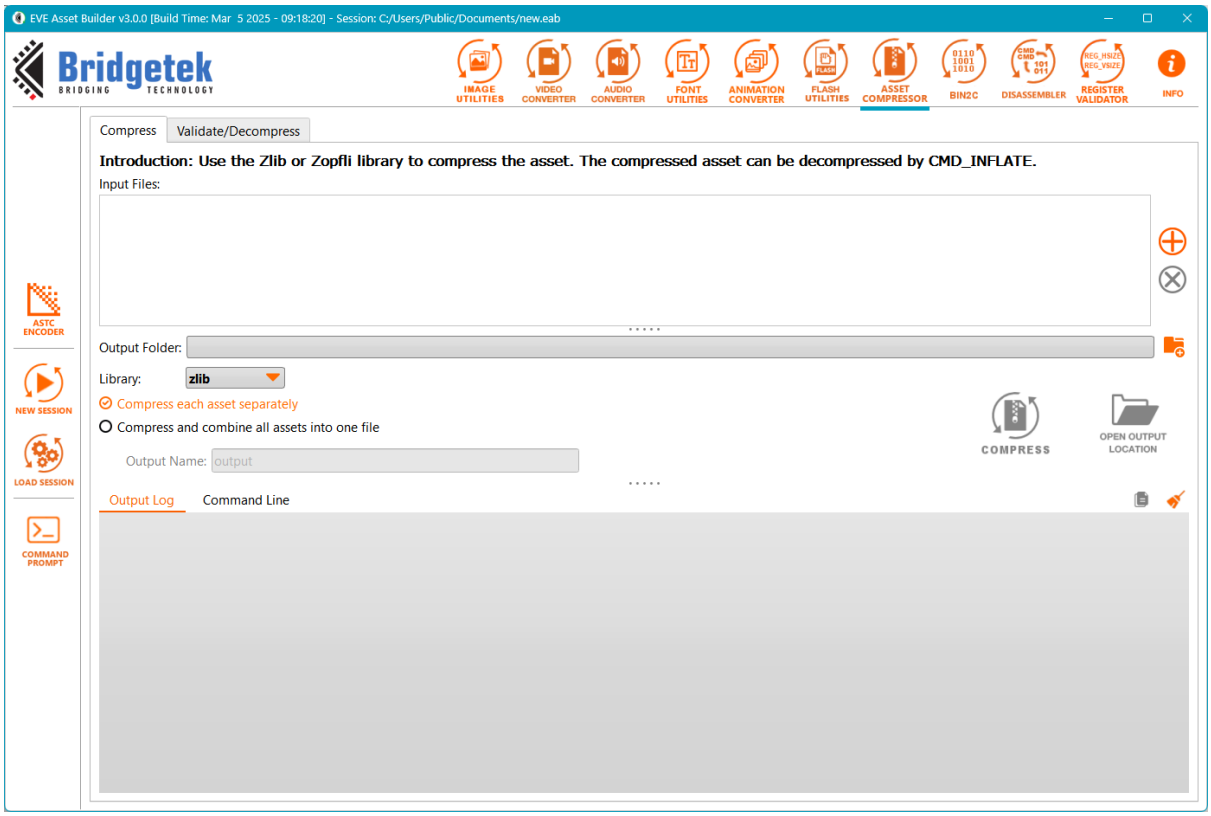


Figure 36 Asset Compressor

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 Zlib but takes much longer to perform the compression. The decompressing speed of Zopfli's output is practically unaffected compared to Zlib's.

2. Validator/Decompressor

Check if the input assets are valid for **cmd_inflate**. This tool supports decompressing the assets if required.

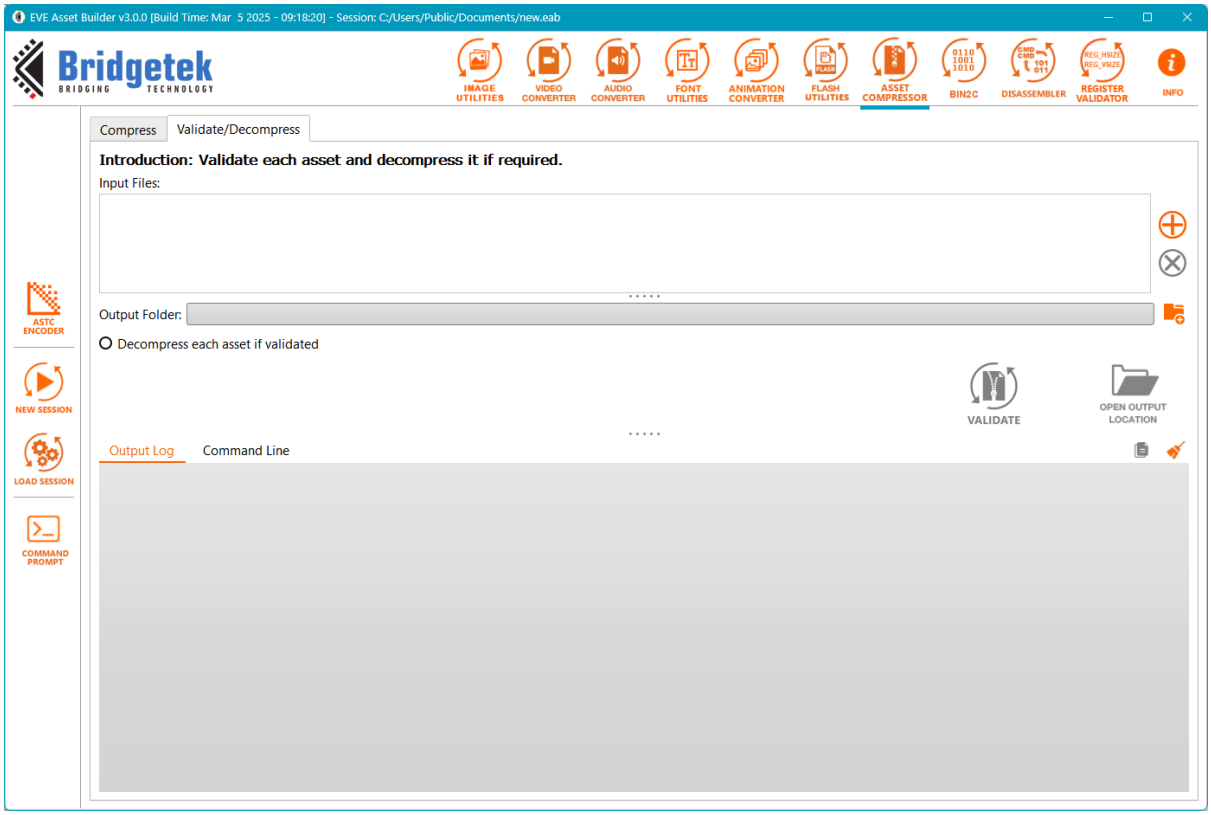


Figure 37 Compressed Asset Validator

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 proceed with decompressing them if required.

H. Binary To C-Array Converter

Bin2C is a utility designed to transform a binary file into a C array. The generated output is a compilable C source file. It supports 1, 2, and 4-byte data types, allowing users to specify the endianness (little or big) for *word* (2 bytes) and *long* (4 bytes) data types.

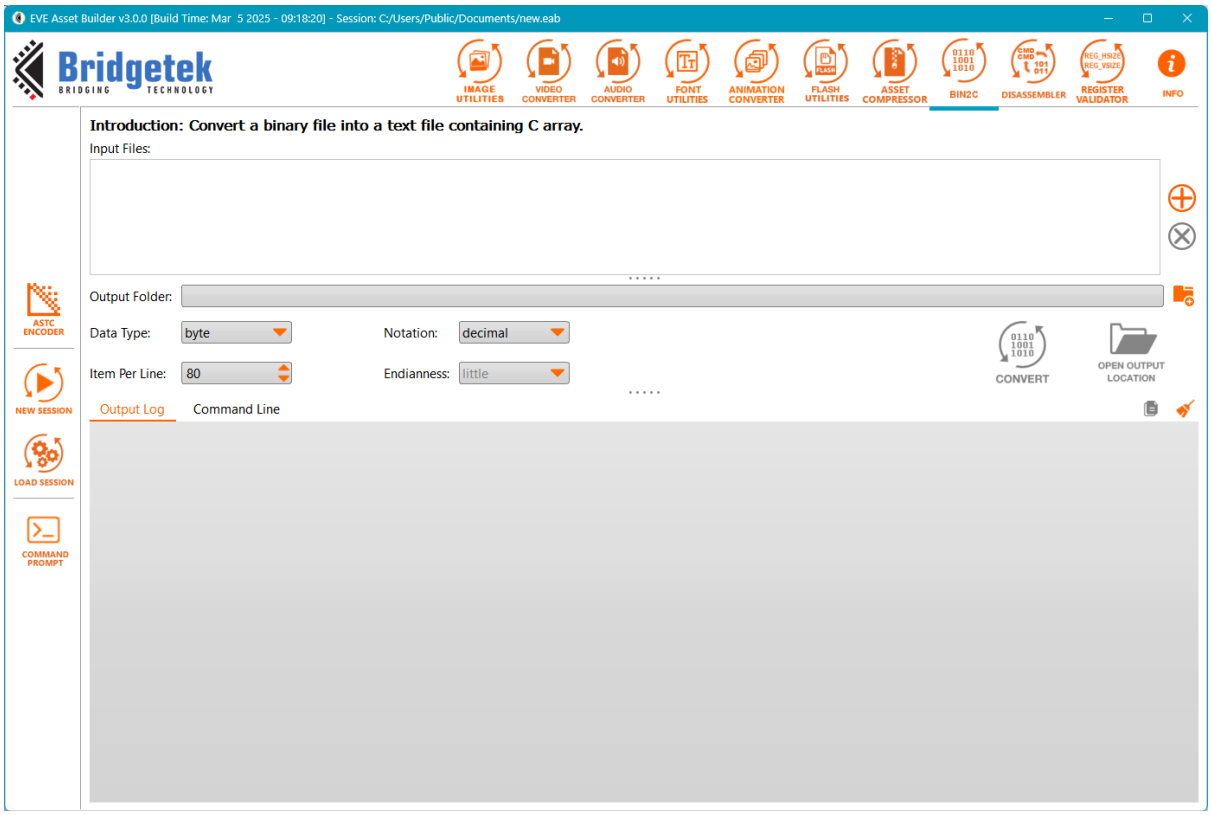


Figure 38 Bin2C Converter

Input Files: A list of binary files to be converted.

Output Folder: A folder where the C array files will be saved.

Data Type: The data type of the C array, which can be byte, word (2 bytes), or long (4 bytes).

Notation: Set the number notation as either decimal (base 10) or hexadecimal (base 16).

Item Per Line: The number of array items to display on each line.

Endianness: Select little-endian or big-endian when reading the binary file.

I. 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.

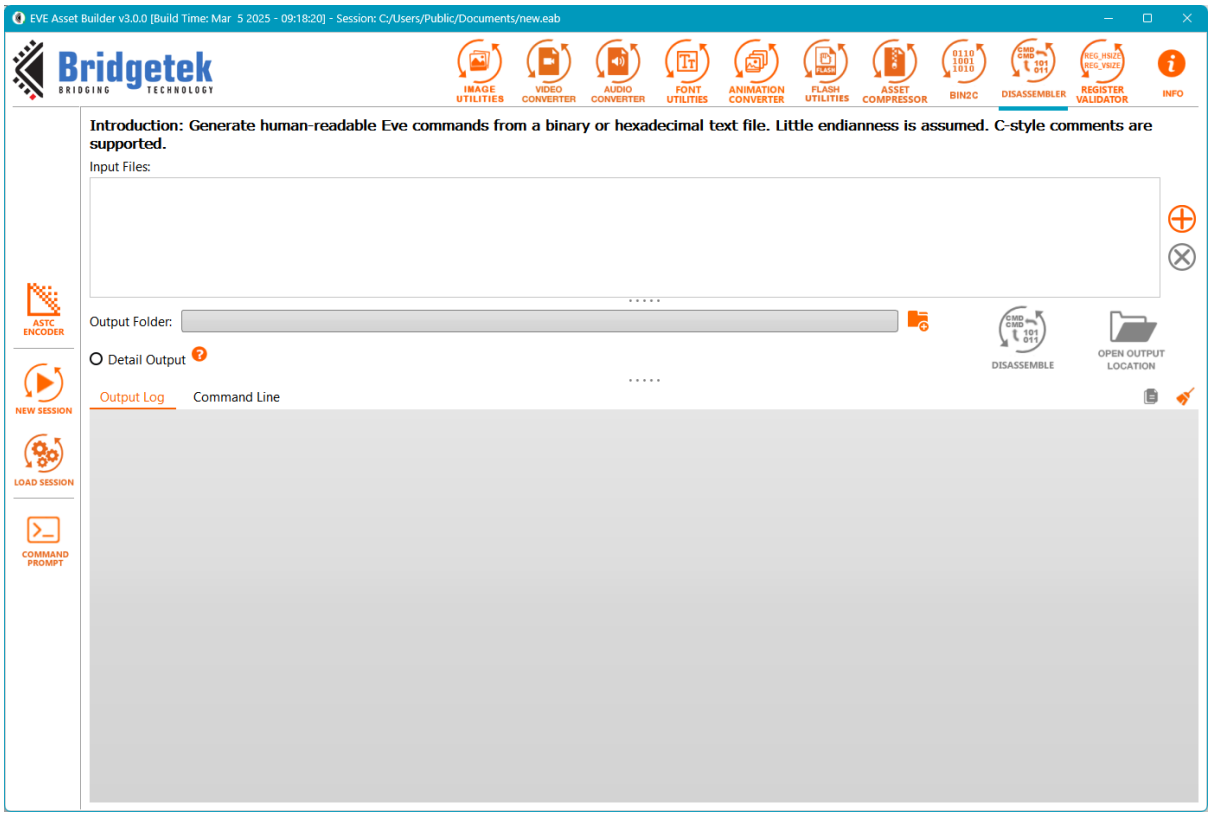


Figure 39 EVE Commands Disassembler

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

Output Folder: A folder to save disassembled files

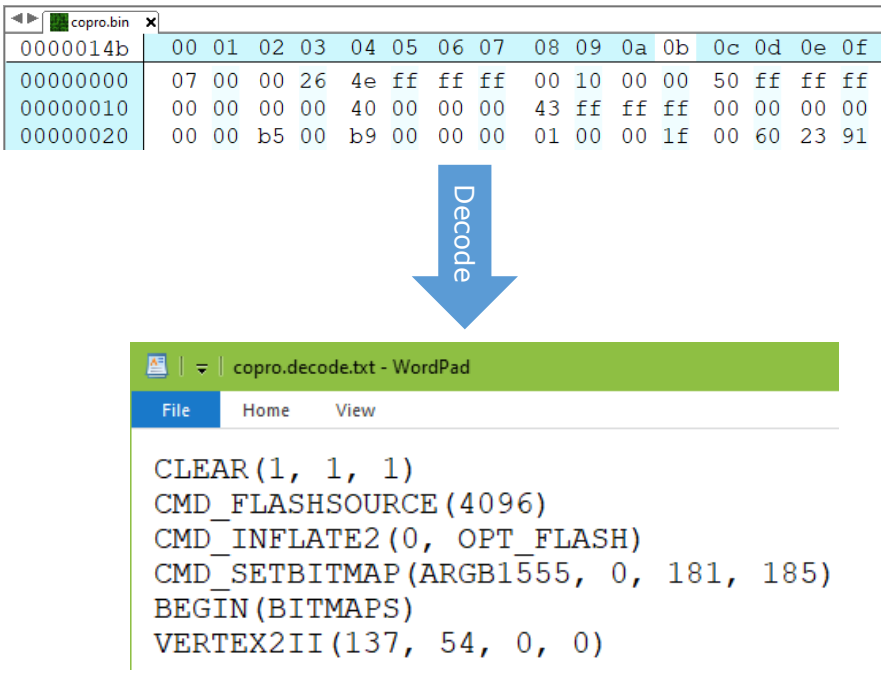
Detail Output: When checked, the output format will be the same as screenshot below.

```
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()
```

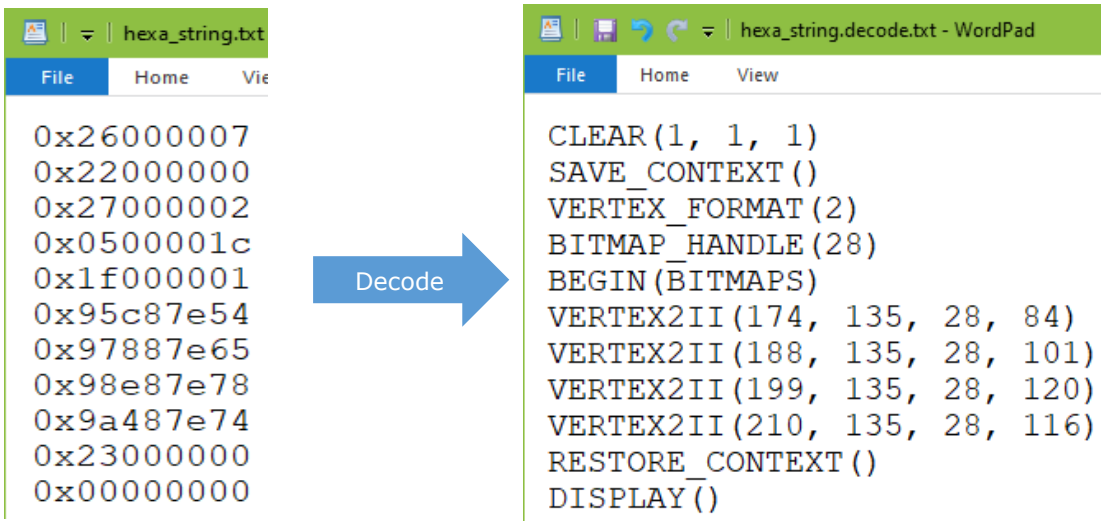
Figure 40 Detail output of Disassembler tool

Sample inputs

- **Binary:**



- **Hexadecimal string:**



J. Register Validator

The tool is designed to validate sets of register values, review them, and highlight the mandatory requirements for these display configuration registers.

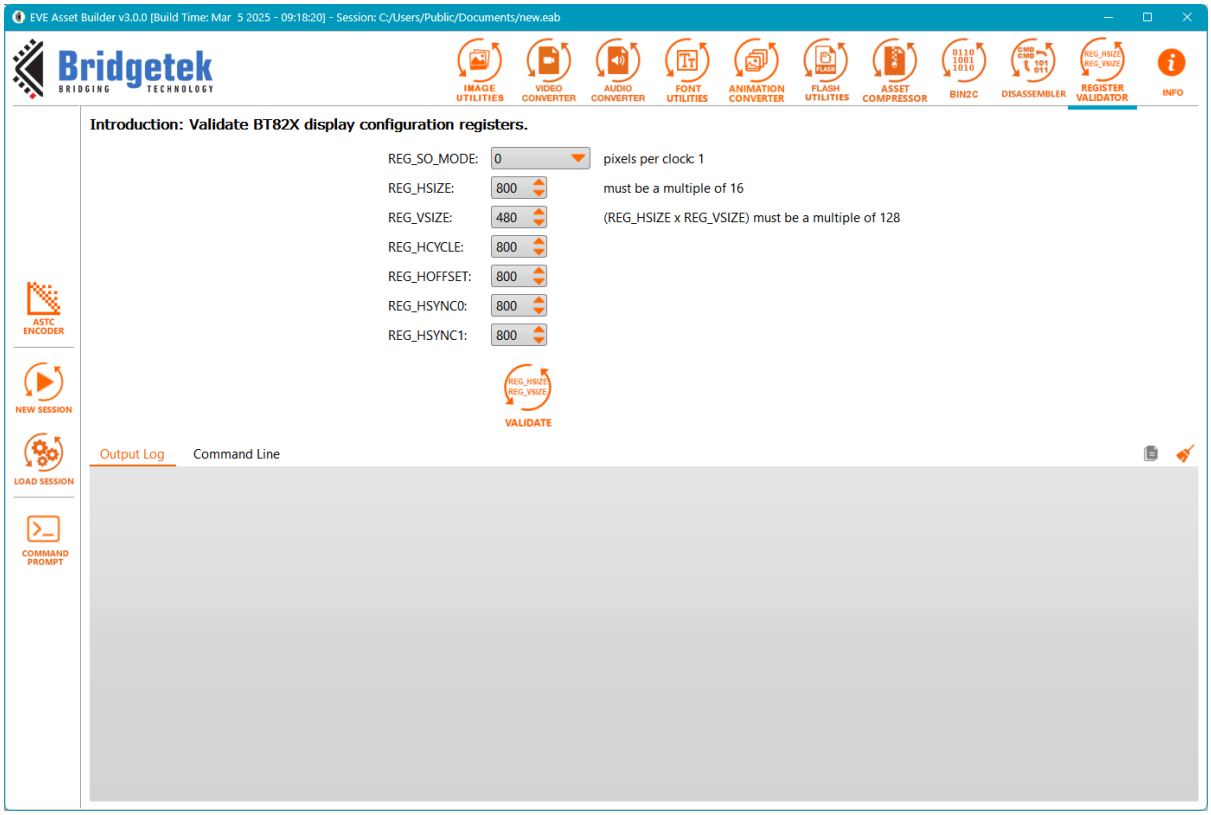


Figure 41 Register Validator

VALIDATE: check the register values and report errors.

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.

Appendix A – List of Figures

Appendix B – List of Tables

Appendix C – Revision History

Document Title	:	BRT_AN_094_BT82X EVE Asset Builder 3.0 User Guide
Document Reference No.	:	BRT_000446
Clearance No.	:	BRT#227
Product Page	:	https://brtchip.com/toolchains/
Document Feedback	:	Send Feedback

Revision	Changes	Date
Version 1.0	User Guide for BT82X EVE Asset Builder 3.0.0	21-03-2025