

**Application Note BRT\_AN\_035**

**EVE Emulator Library User Guide**

**Version** 1.1

**Issue Date: 2018-Aug-01**

This document describes the interface and usage of EVE emulator library.

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 Bridgetek harmless from any and all damages,

claims, suits or expense resulting from such use.

**Bridgetek Pte Ltd (BRTChip)**

178 Paya Lebar Road, #07-03 Singapore 409030

Tel : +65 6547 4827 Fax : +65 6841 6071

Web Site: [http://brtchip.com](http://brtchip.com/)

Copyright © Bridgetek Pte Ltd

**Table of Contents**

[1 Introduction 2](#_Toc513471169)

[1.1 Overview 2](#_Toc513471170)

[1.2 Scope 2](#_Toc513471171)

[1.3 Requirement 2](#_Toc513471172)

[1.4 Limitation 2](#_Toc513471173)

[2 EVE Emulator Library Introduction 3](#_Toc513471174)

[2.1 EVE Emulator Library Interface 3](#_Toc513471175)

[2.2 Emulator Module 4](#_Toc513471176)

[1. API to invoke and perform operations on Emulator 4](#_Toc513471177)

[2. Flags to configure the Emulator 4](#_Toc513471178)

[3. API description 5](#_Toc513471179)

[4. Typical setting for EVE emulator 8](#_Toc513471180)

[5. Use of EVE Emulator library 9](#_Toc513471181)

[2.3 Flash Module 10](#_Toc513471182)

[2. API to invoke and perform operations on Flash Emulator 10](#_Toc513471183)

[3. API description 10](#_Toc513471184)

[4. Typical setting for Flash module using EVE emulator 13](#_Toc513471185)

[5. Use of Flash Emulator library 14](#_Toc513471186)

[3 Contact Information 16](#_Toc513471187)

[Appendix A – References 17](#_Toc513471188)

[Acronyms and Abbreviations 17](#_Toc513471189)

[Appendix B – List of Tables & Figures 18](#_Toc513471190)

[List of Tables 18](#_Toc513471191)

[List of Figures 18](#_Toc513471192)

[Appendix C – Revision History 19](#_Toc513471193)

[Revision Record Sheet 20](#_Toc513471194)

# Introduction

EVE Emulator is the behavior modeling software for any FT8XX/BT8XX running on PC. It is designed as high level (behavior level) emulator other than low level (clock wise accuracy) emulator. It enables the user to evaluate FT8XX/BT8XX features on PC without hardware. EVE emulator library is the implementation and distribution package of EVE emulator. This document describes the interface of EVE Emulator library and shows one example on how to use it in user’s project.

## Overview

EVE emulator has the exact SPI interface as well as memory map with **EVE** Series silicon, i.e., FT80X, FT81X and BT81X. Therefore, user application does not need to change the way to communicate with EVE emulator. In terms of behavior, EVE emulator has the maximum similarity, although there are a few limitations, which are mentioned at [1.4](#_bookmark4).

For touch functionality, EVE emulator requires the mouse of PC to simulate the input of touch.

For visual effect, EVE emulator employs the OS specific graphics driver to display on its monitor. It supports full set of display list commands and most of coprocessor commands.

Therefore, EVE emulator is one ideal tool for customers who would like to prototype or debug their EVE project on PC quickly and easily.

## Scope

This document has covered the interface EVE emulator library provides and introduces how to use it in application. However, this document does not include the implementation details about EVE emulator. The FT8XX/BT8XX relative information, such as registers, memory map, commands, etc. is also not covered.

Although EVE emulator library has the respective version supporting Mac, Linux, Windows, this document only covers the Windows library usage and example.

## Requirement

Currently, the EVE emulator library is built by **MSVC** 2015 Community Edition. So it may require the same tool chain to link it with users’ project successfully.

In addition, the runtime “vcruntime140.dll” and “msvcp140.dll” is required on Windows to run the emulator project successfully.

## Limitation

EVE emulator does NOT have following functionality enabled till now:

* + 1. Power management and host commands,
    2. Coprocessor engine reset,
    3. Interrupt,
    4. Registers which are reflecting hardware properties, e.g., the pressure value of touch and ADC related touch registers,
    5. Performance or hardware related limitation due to EVE emulator is not clock level emulator.

# EVE Emulator Library Introduction

## EVE Emulator Library Interface

The EVE Emulator library interface is written in C++ and it consists of two modules which are “Flash Emulator” and “EVE Emulator” used to emulate the EVE chip and the Flash device. Basically the Flash device is a part of EVE chip in reality. Below are the functions for both emulators mentioned and a brief description. All of the below mentioned functions are wrapped by C++ APIs. Detailed description is available in the following sections.

**Table 1 EVE Emulator library interface structure**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | **API Name** | **Parameter** | **Return** | **Description** |
| Emulator | run | Emulator Params | None | Start the Emulator  using preset parameters. |
| Emulator | stop | None | None | Stop the running Emulator |
| Emulator | cs | Boolean | None | Make Chip selection to read / write one SPI transfer |
| Emulator | isRunning | None | Boolean | Check whether the Emulator is still running. True if running and false if not |
| Emulator | destroy | None | None | To destroy the emulator when it exits. |
| Emulator | transfer | One byte | One byte | Make one byte SPI transaction on SPI bus. |
| Emulator | getRam | None | One byte | Retrieve one byte of Ram data. |
| Flash | vTable | None | See below | Retrieve the raw pointer of the Flash emulator. |

## Emulator Module

### API to invoke and perform operations on Emulator

|  |  |
| --- | --- |
| API Name | Function |
| BT8XXEMU\_defaults | Initialize the default emulator parameters |
| BT8XXEMU\_run | Run the emulator on the current thread. Returns when the emulator is fully stopped when a Main function is supplied, returns when the emulator is fully started otherwise. |
| BT8XXEMU\_stop | Stop the emulator. Can be called from any thread. Returns when the emulator has fully stopped. Safe to call multiple times |
| BT8XXEMU\_destroy | Destroy the emulator. Calls BT8XXEMU\_stop implicitly. Emulator must be destroyed before process exits. |
| BT8XXEMU\_isRunning | Poll if the emulator is still running. Returns 0 when the output window has been closed, or when the emulator has been stopped. |
| BT8XXEMU\_transfer | Transfer data over the imaginary SPI bus. Call from the MCU thread (from the setup/loop callbacks). |
| BT8XXEMU\_chipSelect | Set chip select. Must be set to 1 to start data transfer, 0 to end. See FT8XX documentation for CS\_N |
| BT8XXEMU\_hasInterrupt | Returns 1 if there is an interrupt flag set. Depends on mask. |

### Flags to configure the Emulator

The enumerate below defines the behavior of emulator flags to be run. You can “OR” these enumerates and assign these values to “Flags” field in the parameter structure “EmulatorParameters”.

enum BT8XXEMU\_EmulatorFlags

**{**

// enables the keyboard to be used as input BT8XXEMU\_EmulatorEnableKeyboard =0x01**,**

// enables audio BT8XXEMU\_EmulatorEnableAudio= 0x02,

// enables coprocessor BT8XXEMU\_EmulatorEnableCoprocessor=0x04**,**

// enables mouse as touch BT8XXEMU\_EmulatorEnableMouse=0x08**,**

// enable debug shortkeys BT8XXEMU\_EmulatorEnableDebugShortkeys=0x10**,**

// enable graphics processor multithreading BT8XXEMU\_EmulatorEnableGraphicsMultithread=0x20**,**

// enable dynamic graphics quality degrading by interlacing BT8XXEMU\_EmulatorEnableDynamicDegrade =0x40**,**

// enable emulating REG\_PWM\_DUTY by fading the rendered display to black

BT8XXEMU\_EmulatorEnableRegPwmDutyEmulation **=** 0x100**,**

// enable usage of touch transformation matrix BT8XXEMU\_EmulatorEnableTouchTransformation = 0x200,

// enable output to stdout from the emulator BT8XXEMU\_EmulatorEnableStdOut = 0x400,

// enable performance adjustments for running the emulator as a background process without window BT8XXEMU\_EmulatorEnableBackgroundPerformance = 0x800,

// enable performance adjustments for the main MCU thread

BT8XXEMU\_EmulatorEnableMainPerformance = 0x1000,

**};**

**Figure 1 Emulator Flags field definition**

### API description

#### BT8XXEMU\_defaults

* **Prototype**

void BT8XXEMU\_defaults (uint32\_t versionApi, BT8XXEMU\_EmulatorParameters \*params, BT8XXEMU\_EmulatorMode mode);

* **Description**

Call this API in order to initialize the default parameters of the Emulator by setting the parameters in the params argument and setting the mode in mode argument which are passed to this function.

* **Return value**

None

* **Parameter**

versionApi This parameter describe the version of the API. The version of API changes whenever there is a change in the number of Emulator parameters or the format and changes in the functions itself.

params Emulator Parameters – (Refer figure 2 below for Emulator Parameters)

mode Defines which emulator needs to be invoked. See note below for the different emulator modes.

typedef struct

{

// Microcontroller main function. This will be run on a new thread managed by the

// emulator. When not provided the calling thread is assumed to be the MCU thread

void(\*Main)(BT8XXEMU\_Emulator \*sender, void \*context);

// See EmulatorFlags.

int Flags;

// Emulator mode

BT8XXEMU\_EmulatorMode Mode;

// The default mouse pressure, default 0 (maximum).

// See REG\_TOUCH\_RZTRESH, etc.

uint32\_t MousePressure;

// External frequency. See CLK, etc.

uint32\_t ExternalFrequency;

// Reduce graphics processor threads by specified number, default 0

// Necessary when doing very heavy work on the MCU or Coprocessor

uint32\_t ReduceGraphicsThreads;

// Sleep function for MCU thread usage throttle. Defaults to generic system sleep

void(\*MCUSleep)(BT8XXEMU\_Emulator \*sender, void \*context, int ms);

// Replaces the default builtin ROM with a custom ROM from a file.

// NOTE: String is copied and may be deallocated after call to run(...)

wchar\_t RomFilePath[260];

// Replaces the default builtin OTP with a custom OTP from a file.

// NOTE: String is copied and may be deallocated after call to run(...)

wchar\_t OtpFilePath[260];

// Replaces the builtin coprocessor ROM.

// NOTE: String is copied and may be deallocated after call to run(...)

wchar\_t CoprocessorRomFilePath[260];

// Graphics driverless mode

// Setting this callback means no window will be created, and all

// rendered graphics will be automatically sent to this function.

// For enabling touch functionality, the functions

// Memory.setTouchScreenXY and Memory.resetTouchScreenXY must be

// called manually from the host application.

// Builtin keyboard functionality is not supported and must be

// implemented manually when using this mode.

// The output parameter is false (0) when the display is turned off.

// The contents of the buffer pointer are undefined after this

// function returns. Create a copy to use it on another thread.

// Return false (0) when the application must exit, otherwise return true (1).

int(\*Graphics)(BT8XXEMU\_Emulator \*sender, void \*context, int output, const argb8888

\*buffer, uint32\_t hsize, uint32\_t vsize, BT8XXEMU\_FrameFlags flags);

// Log callback

void(\*Log)(BT8XXEMU\_Emulator \*sender, void \*context, BT8XXEMU\_LogType type,

const char \*message);

// Safe exit. Called when the emulator window is closed

void(\*Close)(BT8XXEMU\_Emulator \*sender, void \*context);

// User context that will be passed along to callbacks

void \*UserContext;

// Flash device to connect with, default NULL

BT8XXEMU\_Flash \*Flash;

} BT8XXEMU\_EmulatorParameters;

**Figure 2 Emulator parameters**

typedef enum

{

BT8XXEMU\_EmulatorFT800 = 0x0800,

BT8XXEMU\_EmulatorFT801 = 0x0801,

BT8XXEMU\_EmulatorFT810 = 0x0810,

BT8XXEMU\_EmulatorFT811 = 0x0811,

BT8XXEMU\_EmulatorFT812 = 0x0812,

BT8XXEMU\_EmulatorFT813 = 0x0813,

BT8XXEMU\_EmulatorBT815 = 0x0815,

BT8XXEMU\_EmulatorBT816 = 0x0816

} BT8XXEMU\_EmulatorMode;

**Figure 3 Emulator mode**

#### BT8XXEMU\_run

* **Prototype**

void BT8XXEMU\_run (uint32\_t versionApi, BT8XXEMU\_Emulator \*\*emulator, const BT8XXEMU\_EmulatorParameters \*params);

* **Description**

This API runs the emulator on the current thread. The API returns when the emulator is fully stopped when a Main function is supplied or returns when the emulator is fully started otherwise.

* **Return value**

None

* **Parameter**

versionApi This parameter describe the version of the API. The version of API changes whenever there is a change in the number of emulator parameters or the format and changes in the functions itself.

emulator This parameter is the handle to the global emulator instance.

params This parameter is used to set the emulator params. (Refer figure 2 above for emulator params)

#### BT8XXEMU\_chipSelect

* **Prototype**

void BT8XXEMU\_chipSelect (BT8XXEMU\_Emulator \* emulator, int cs);

* **Description**

This API does the chip selection by setting the cs value to 1 to start data transfer and 0 to stop the data transfer.

* **Return value**

None

* **Parameter**

cs This parameter is set to 1/0 to start/stop the data transfer.

emulator This parameter is the handle to the global emulator instance.

#### BT8XXEMU\_transfer

* **Prototype**

uint8\_t BT8XXEMU\_transfer (BT8XXEMU\_Emulator \* emulator, uint8\_t data);

* **Description**

Calling this API is equivalent to sending one byte data on the MOSI line of SPI bus, at same time receiving one byte data on the MISO line of SPI bus. The data to be sent is specified as parameter, while the data to be received is given as return value.

* **Return value**

One byte data received from EVE emulator, equivalent to receiving from the MISO line of SPI bus.

* **Parameter**

emulator This parameter is the handle to the global emulator instance.

data One byte data sending to EVE emulator. In the case of SPI read transaction, this byte can be anything.)

#### BT8XXEMU\_stop

* **Prototype**

void BT8XXEMU\_stop (BT8XXEMU\_Emulator \* emulator);

* **Description**

Call to this API will stop the running emulator instance.

* **Return value**

Nothing.

* **Parameter**

emulator This parameter is the handle to the global emulator instance.

#### BT8XXEMU\_destroy

* **Prototype**

void BT8XXEMU\_destroy (BT8XXEMU\_Emulator \* emulator);

* **Description**

Call to this API will destroy the emulator instance.

* **Return value**

Nothing.

* **Parameter**

emulator This parameter is the handle to the global emulator instance.

### Typical setting for EVE emulator

Currently, please note that only the settings as below are strongly recommended to use now.

**For running only one output of Emulator instance**

If the objective is to run only one emulator output, then the most convenient way is to set the callback function pointer Main() in BT8XXEMU\_EmulatorParameters struct by defining a callback function in the user project and assign the pointer to callback function.

The implementation of the callback function Main will consists of 2 user defined functions setup() for initialization purpose and loop() which will be periodically called by emulator. These 2 user defined functions will make sure that the user project is in the context of BT815 emulator.

When using BT8XXEMU\_EmulatorParameters.Main, the function will be blocking, until either BT8XXEMU\_stop is called, or the emulator window is closed. Then it will kill the BT8XXEMU\_EmulatorParameters.Main thread to stop the process.

The drawback of this implementation is only single output display.

**For running multiple output of Emulator instance**

On the other hand, when BT8XXEMU\_EmulatorParameters.Main is not provided, the function will return as soon as the emulator is ready to use.

This should be the preferred method which allows for multiple emulator displays to be configured. In the event of using this method, the user must periodically check BT8XXEMU\_isRunning() on all emulator instances to verify that the emulator window was not closed by the user, and gracefully exit the application in case the emulator has stopped running.

#include <bt8xxemu.h>

#include <stdio.h>

BT8XXEMU\_Emulator \*pEmulator;

void loop();

void setup();

void mcu(BT8XXEMU\_Emulator \*sender, void \*context)

{

setup();

while (BT8XXEMU\_isRunning(pEmulator))

loop();

}

int main(int, char\* [])

{

BT8XXEMU\_EmulatorParameters params;

BT8XXEMU\_defaults(BT8XXEMU\_VERSION\_API, &params, BT8XXEMU\_EmulatorBT815);

params.Main = mcu;

BT8XXEMU\_run(BT8XXEMU\_VERSION\_API, &pEmulator, &params);

return 0;

}

**Figure 4 Setup and Run BT815 emulator**

### Use of EVE Emulator library

This chapter will give one example on how to use EVE emulator in sample application. Users are encouraged to get familiar with BT815 sample application before starting this chapter.

#### Start the BT815 emulator

To make use of BT815 emulator, user project requires to call the API “BT8XXEMU\_defaults()” with the specific parameter to set up the BT815 emulator, then it is required to call the API BT8XXEMU\_run() in order to start the emulator.

Please [see Figure 4 Setup and Run BT815 emulator](#_bookmark16).

#### Build and Run

After porting the application to BT815 emulator according to the instructions above, to build the final executable, user project is needed to specify the path and name of EVE emulator library.

For debug/release build, please specify the BT815 emulator library named “bt8xxemu.lib”.

Please note that Microsoft Visual Studio 2015 Community version is a must to build with your application.

The picture below shows the screenshot when the BT815 sample application run on top of BT815 emulator.



**Figure 5 Sample applications running on top of BT815 emulator** (Setting Font)

## Flash Module

The latest BT8XX chips do support an external Flash memory using QSPI interface. The flash module could be simulated by sending and receiving data using the EVE emulator. Separate set of APIs are used exclusively to invoke flash emulator using the EVE emulator. Below are the APIs and its description of Flash emulator.

### API to invoke and perform operations on Flash Emulator

|  |  |
| --- | --- |
| API Name | Function |
| BT8XXEMU\_Flash\_defaults | Initialize the default flash emulator parameters |
| BT8XXEMU\_Flash\_create | Create the flash emulator instance |
| BT8XXEMU\_Flash\_destroy | Destroy the flash emulator instance |
| BT8XXEMU\_Flash\_transferSpi4 | Transfer data using SPI or Quad SPI protocol. Bit 0:3 are data, bit 4 is cable select (0 active), SCK is clock. In single mode bit 0 is MOSI and bit 1 is MISO |
| BT8XXEMU\_Flash\_data | Vtable data in flash |
| BT8XXEMU\_Flash\_size | Size of Vtable data in flash |

### API description

#### BT8XXEMU\_Flash\_defaults

* **Prototype**

void BT8XXEMU\_Flash\_defaults(uint32\_t versionApi, BT8XXEMU\_FlashParameters \*params);

* **Description**

Call this API in order to initialize the default parameters of the flash emulator by setting the parameters in the params argument and setting the mode in mode argument which are passed to this function.

* **Return value**

None

* **Parameter**

versionApi This parameter describe the version of the API. The version of API changes whenever there is a change in the number of emulator parameters or the format and changes in the functions itself.

params This parameter is used to set the flash emulator params (Refer figure 4 below for flash emulator parameters)

typedef struct

{

// Device type, by library name, default "mx25lemu"

wchar\_t DeviceType[26];

// Size of the flash memory in bytes, default 8MB

size\_t SizeBytes;

// Data file to load onto the flash, default NULL

wchar\_t DataFilePath[260];

// Internal flash status file, device specific, default NULL

wchar\_t StatusFilePath[260];

// Write actions to the flash are persisted to the used file.

// This is accomplished by memory mapping the file, instead of

// the file being copied to memory. Default false

bool Persistent;

// Print log to standard output. Default false

bool StdOut;

// Data buffer that is written to the flash initially,

// overriding any existing contents that may have been

// loaded from a flash file already, default NULL and 0

void \*Data;

size\_t DataSizeBytes;

// Log callback

void(\*Log)(BT8XXEMU\_Flash \*sender, void \*context, BT8XXEMU\_LogType type,

const char \*message);

// User context that will be passed along to callbacks

void \*UserContext;

} BT8XXEMU\_FlashParameters;

**Figure 6 Flash Emulator Parameters**

#### BT8XXEMU\_Flash\_create

* **Prototype**

BT8XXEMU\_Flash \*BT8XXEMU\_Flash\_create(uint32\_t versionApi, const BT8XXEMU\_FlashParameters \*params);

* **Description**

This API creates a flash emulator instance and assigns the handle to the flash pointer.

* **Return value**

BT8XXEMU\_Flash\* - Returns the pointer to the flash instance created.

* **Parameter**

versionApi This parameter describe the version of the API. The version of API changes whenever there is a change in the number of flash emulator parameters or the format and changes in the functions itself.

params (Refer figure 4 above for flash emulator parameters)

#### BT8XXEMU\_Flash\_destroy

* **Prototype**

void BT8XXEMU\_Flash\_destroy(BT8XXEMU\_Flash \*flash);

* **Description**

This API destroys the flash emulator instance which was created earlier.

* **Return value**

Nothing.

* **Parameter**

flash Pointer to the flash emulator instance to be destroyed.

#### BT8XXEMU\_Flash\_transferSpi4

* **Prototype**

uint8\_t BT8XXEMU\_Flash\_transferSpi4(BT8XXEMU\_Flash \*flash, uint8\_t signal);

**Description**

This API transfers data using SPI or Quad SPI protocol.

* **Return value**

uint8\_t Returns one byte of data while reading.

* **Parameter**

flash Pointer to the flash emulator instance to which the data to be transferred.

signal sends one byte of data while writing.

#### BT8XXEMU\_Flash\_data

* **Prototype**

uint8\_t\* BT8XXEMU\_Flash\_data(BT8XXEMU\_Flash \*flash);

**Description**

This API retrieves the raw data pointer of the flash emulator.

* **Return value**

uint8\_t\* Returns the pointer to the address of the flash emulator data.

* **Parameter**

flash Pointer to the flash emulator instance.

#### BT8XXEMU\_Flash\_size

* **Prototype**

size\_t BT8XXEMU\_Flash\_size(BT8XXEMU\_Flash \*flash);

**Description**

This API retrieves the size in bytes of the flash emulator.

* **Return value**

size\_t Returns an unsigned byte of the size of the flash emulator.

* **Parameter**

flash Pointer to the flash emulator instance.

### Typical setting for Flash module using EVE emulator

Currently, please note that only the settings as below are strongly recommended to use now.

Basically the flash module is a part of the EVE emulator itself only in the case of BT8XX chips, and it could be setup by calling the APIs as mentioned below in the code snippet.

#define BTFLASH\_DEVICE\_TYPE L"mx25lemu"

#define BTFLASH\_SIZE (8 \* 1024 \* 1024)

#define BTFLASH\_DATA\_FILE L"C:/Projects/FT8XXEmulator/reference/vc3roms/stdflash.bin"

#define BT8XXEMU\_VERSION\_API 11

#define BTFLASH\_FIRMWARE L"C:/Projects/FT8XXEmulator/fteditor/firmware/mx25l.blob"

int main(int, char\*[])

{

printf("%s\n\n", BT8XXEMU\_version());

BT8XXEMU\_FlashParameters flashParams;

BT8XXEMU\_Flash\_defaults(BT8XXEMU\_VERSION\_API, &flashParams);

wcscpy(flashParams.DeviceType, BTFLASH\_DEVICE\_TYPE);

flashParams.SizeBytes = BTFLASH\_SIZE;

wcscpy(flashParams.DataFilePath, BTFLASH\_DATA\_FILE);

flashParams.StdOut = true;

BT8XXEMU\_Flash \*flash;

/////////////////////////////////////////////////////////////////

//// Test different memory sizes

/////////////////////////////////////////////////////////////////

int sizes[8] = { 2, 4, 8, 16, 32, 64, 128, 256 };

for (int si = 0; si < 8; ++si)

{

int sz = sizes[si];

printf("SIZE %i\n", sz);

/////////////////////////////////////////////////////////////////

//// Emulator

/////////////////////////////////////////////////////////////////

flashParams.SizeBytes = sz \* 1024 \* 1024;

wcscpy(flashParams.DataFilePath, BTFLASH\_FIRMWARE);

flash = BT8XXEMU\_Flash\_create(BT8XXEMU\_VERSION\_API, &flashParams);

data = BT8XXEMU\_Flash\_data(flash);

assert(data[0] == 0x70);

size = BT8XXEMU\_Flash\_size(flash);

assert(size == sz \* 1024 \* 1024);

BT8XXEMU\_EmulatorParameters params;

BT8XXEMU\_defaults(BT8XXEMU\_VERSION\_API, &params, BT8XXEMU\_EmulatorBT815);

params.Flags |= BT8XXEMU\_EmulatorEnableStdOut;

params.Flash = flash;

BT8XXEMU\_Emulator \*emulator = NULL;

BT8XXEMU\_run(BT8XXEMU\_VERSION\_API, &emulator, &params);

uint8\_t \*ram = BT8XXEMU\_getRam(emulator);

wr32(emulator, REG\_HSIZE, 480);

wr32(emulator, REG\_VSIZE, 272);

wr32(emulator, REG\_PCLK, 5);

flush(emulator);

while (!rd32(emulator, REG\_FLASH\_STATUS));

assert(rd32(emulator, REG\_FLASH\_STATUS) == FLASH\_STATUS\_BASIC);

/////////////////////////////////////////////////////////////////

//// Enter full speed mode

/////////////////////////////////////////////////////////////////

; {

printf("CMD\_FLASHFAST\n");

wr32(emulator, REG\_CMDB\_WRITE, CMD\_FLASHFAST);

uint32\_t resAddr = rd32(emulator, REG\_CMD\_WRITE);

wr32(emulator, REG\_CMD\_WRITE, resAddr + 4);

flush(emulator);

assert(rd32(emulator, resAddr) == 0);

assert(rd32(emulator, REG\_FLASH\_STATUS) == FLASH\_STATUS\_FULL);

}

for (int i = 0; i < 4096; ++i)

{

ram[i] = 0x55;

}

; {

printf("CMD\_FLASHREAD (FLASH\_STATUS\_FULL)\n");

wr32(emulator, REG\_CMDB\_WRITE, CMD\_FLASHREAD);

wr32(emulator, REG\_CMDB\_WRITE, 128); // dest

wr32(emulator, REG\_CMDB\_WRITE, 128); // src

wr32(emulator, REG\_CMDB\_WRITE, 512); // num

flush(emulator);

for (int i = 128; i < 128 + 512; ++i)

assert(ram[i] == data[i]);

}

int idx = (sz \* 1024 \* 1024) - 12288;

for (int i = 0; i < 256; ++i)

{

data[idx + i] = i;

}

; {

printf("CMD\_FLASHREAD (%i)\n", idx);

wr32(emulator, REG\_CMDB\_WRITE, CMD\_FLASHREAD);

wr32(emulator, REG\_CMDB\_WRITE, 128); // dest

wr32(emulator, REG\_CMDB\_WRITE, idx); // src

wr32(emulator, REG\_CMDB\_WRITE, 256); // num

flush(emulator);

for (int i = 0; i < 256; ++i)

assert(ram[128 + i] == i);

}

/////////////////////////////////////////////////////////////////

BT8XXEMU\_stop(emulator);

BT8XXEMU\_destroy(emulator);

emulator = NULL;

BT8XXEMU\_Flash\_destroy(flash);

flash = NULL;

data = NULL;

}

}

**Figure 7 Setup and Run flash module using EVE emulator**

### Use of Flash Emulator library

This chapter will give one example on how to use Flash emulator in sample application.

#### Start the Flash module using EVE emulator

To make use of Flash emulator, user project requires to call the API “BT8XXEMU\_Flash\_defaults()” with the specific parameter to set up the Flash module, then it is required to call the API BT8XXEMU\_Flash\_create() in order to create an instance of the Flash module by loading flash module library (**mx25lemu.dll**).

The data transfer to Flash module either in or out is started using the BT8XXEMU\_Flash\_transferSpi4() API. The flash data is retrieved using the vTable pointer to data in Flash module. The API BT8XXEMU\_Flash\_data() will in turn invoke the vTable pointer to give access to the data location.

Please [see Figure 4 Setup and Run Flash module](#_bookmark16) using EVE emulator.

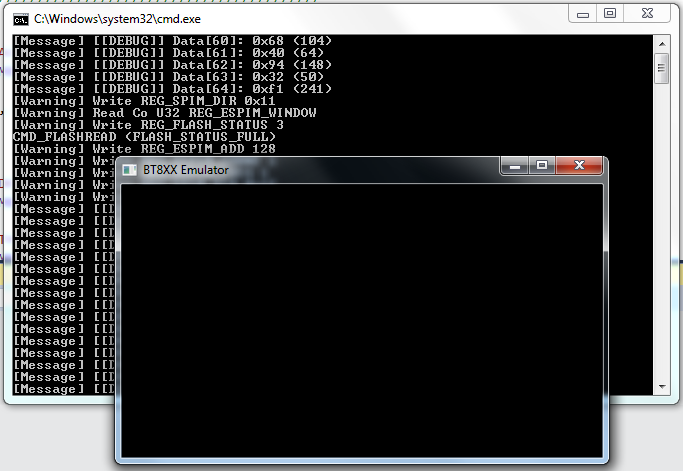
#### Build and Run

After porting the application to run flash module using BT815 emulator according to the instructions above, to build the final executable, user project is needed to specify the path and name of EVE emulator library and the flash module library should be present in the final executable path as well.

For debug/release build, please specify the EVE emulator library named “bt8xxemu.lib”.

Please note that Microsoft Visual Studio 2015 Community version is a must to build with your application.

The picture below shows the screenshot when the flash emulator sample application run on top of EVE emulator using flash module.



**Figure 8 Sample flash module application running on top of BT815 emulator**

# Contact Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Head Quarters – Singapore** | | **Branch Office – Taipei, Taiwan** | |
|  | |  | |
| Bridgetek Pte Ltd  178 Paya Lebar Road, #07-03  Singapore 409030  Tel: +65 6547 4827  Fax: +65 6841 6071 | | 2F, No. 516, Sec. 1, NeiHu Road Taipei 114, Taiwan  Tel: +886 (2) 8797 1330  Fax: +886 (2) 8751 9737 | |
|  | |  | |
| E-mail (Sales) | [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com) | E-mail (Sales) | [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com) |
| E-mail (Support) | [support.apac@brtchip.com](mailto:support.apac@brtchip.com) | E-mail (Support) | [support.apac@brtchip.com](mailto:support.apac@brtchip.com) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Branch Office - Glasgow, United Kingdom** | | **Branch Office – Vietnam** | |
|  | |  | |
| Bridgetek Pte. Ltd.  Unit 1, 2 Seaward Place,  Centurion Business Park  Glasgow G41 1HH  United Kingdom  Tel: +44 (0) 141 429 2777  Fax: +44 (0) 141 429 2758 | | Bridgetek VietNam Company Limited  Lutaco Tower Building, 5th Floor,  173A Nguyen Van Troi, Ward 11, Phu Nhuan District, Ho Chi Minh City, Vietnam  Tel : 08 38453222 Fax : 08 38455222 | |
|  | |  | |
| E-mail (Sales) | [sales.emea@brtichip.com](mailto:sales.emea@brtichip.com) | E-mail (Sales) | [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com) |
| E-mail (Support) | [support.emea@brtchip.com](mailto:support.emea@brtchip.com) | E-mail (Support) | [support.apac@brtchip.com](mailto:support.apac@brtchip.com) |

**Web Site**

<http://www.brtchip.com>

|  |
| --- |
| System and equipment manufacturers and designers are responsible to ensure that their systems, and any Bridgetek Pte Limited (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 for any applications 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 and all damages, claims, suits or expense 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 Limited, 178 Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number: 201542387H. |

# Appendix A – References

## Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Terms** | **Description** |
| USB | Universal Serial Bus |
| USB-IF | USB Implementers Forum |
| PC | Personal Computer |
| Windows | Microsoft Windows Desktop operating system |

# Appendix B – List of Tables & Figures

## List of Tables

[Table 1 EVE Emulator library interface structure 3](#_bookmark7)

## List of Figures

[Figure 1 Flags field definition 4](#_bookmark15)

[Figure 2 Emulator parameters 6](#Figure2)

[Figure 3 Emulator mode 6](#Figure3)

[Figure 4 Setup and Run BT815 emulator 8](#Figure4)

[Figure 5 Sample applications running on top of BT815 emulator 9](#Figure5)

[Figure 6 Flash Emulator Parameters 11](#Figure6)

[Figure 7 Setup and Run flash module using EVE emulator 14](#Figure7)

[Figure 8 Sample flash module application running on top of BT815 emulator 15](#Figure8)

# Appendix C – Revision History

Document Title: BRT\_AN\_035 EVE Emulator Library User Guide

Document Reference No.: BRT\_000229

Clearance No.: BRT# \*\*\*\*

Product Page: <http://www.brtchip.com/FTProducts.htm>

Document Feedback: [Send Feedback](mailto:docufeedback@ftdichip.com)

|  |  |  |
| --- | --- | --- |
| **Revision** | **Changes** | **Date** |
| 1.0 | Initial Release | 2018-04-12 |
| 1.1 | Minor corrections in API descriptions | 2018-08-01 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Revision Record Sheet

|  |  |
| --- | --- |
| **Authors** | Mohamed Fysal |
| **Filename** | BRT\_AN\_035 EVE Emulator Library User Guide.docx |

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Details** |
| 1.0 | 2018-04-12 | First Draft |
| 1.1 | 2018-08-01 | Minor corrections in API descriptions |
|  |  |  |
|  |  |  |

**Sign Off**

|  |  |  |
| --- | --- | --- |
| **Signatory** | **Signature** | **Date** |
| **Managing Director** |  |  |
| **Principal Hardware Engineer** |  |  |
| **Principal Software Engineer** |  |  |
| **Senior Marketing Manager** |  |  |
| **Sales Manager** |  |  |

**Clearance Approval (Please delete unnecessary conditions upon purposes!)**

* **This document contains COMPANY CONFIDENTAIL INFORMATION**
* **This document is for INTERNAL USE ONLY**
* **A Non-Disclosure Agreement (NDA) is required prior to external circulation.**
* **This document is cleared for Future Technology Devices International use and restricted circulation to customers as directed by Sales and Applications.**
* **This Document is cleared for Future Technology Devices International use and unrestricted circulation. An NDA is not required prior to external circulation.**
* **Other specific conditions <describe here>**

**BRT# \*\*\*\* Clearance Number**

**(Where applicable for external communications)**

**Revision History**

Revision history (internal use only, please clearly state all changes here before saving the file)

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date YYYY-MM-DD** | **Changes** | **Editor** |
| 1.0 | 2018-04-12 | First draft. | Mohamed Fysal |
| 1.1 | 2018-09-01 | Minor corrections in API descriptions. | Mohamed Fysal |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |