Application Note

BRT_AN_035

EVE Emulator Library User Guide

Version 0.1

Issue Date: 2018-Apr-12

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

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

Table of Contents

1 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.

1.1 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.

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.

1.2 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.

1.3 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.

1.4 Limitation

EVE emulator does NOT have following functionality enabled till now:

1. Power management and host commands,
2. Coprocessor command "cmd_snapshot" has no effect
3. Coprocessor engine reset,
4. Interrupt,
5. Registers which are reflecting hardware properties, e.g., the pressure value of touch and ADC related touch registers,
6. Performance or hardware related limitation due to EVE emulator is not clock level emulator.

2 EVE Emulator Library Introduction

2.1 EVE Emulator Library Interface

All the EVE Emulator library interface is written in C++ and resides in the "`BT8XXEMU::`" name space only. Within the "`BT8XXEMU::`" name space, there are two modules "Flash" and "Emulator". Here is the structure:

Table 1 EVE Emulator library interface structure

Name Space	Module Name	API Name	Parameter	Return	Description
BT8XXEMU::	Emulator	stop	None	None	Stop the running Emulator
BT8XXEMU::	Emulator	cs	Boolean	None	Make Chip selection to read / write one SPI transfer
BT8XXEMU::	Emulator	isRunning	None	Boolean	Check whether the Emulator is still running. True if running and false if not
BT8XXEMU::	Emulator	destroy	None	None	To destroy the emulator when it exits.
BT8XXEMU::	Emulator	transfer	One byte	One byte	Make one byte SPI transaction on SPI bus.
BT8XXEMU::	Emulator	getRam	None	One byte	Retrieve one byte of Ram data.
BT8XXEMU::	Flash	vTable	None	See below	Retrieve the vTable from the Flash.

2.2 Emulator Module

1. 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.

2. 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

3. 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_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.

4. Typical setting for EVE emulator

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

The callback function "mcu()" **should be defined** by user project and they will be called by emulator. The mcu() implementation should call 2 more functions "setup()" and "loop()". Function "setup()" is assumed to run once by emulator for initialization purpose. Function "loop()" will be called periodically by emulator. These two functions make sure the user project is in the context of EVE emulator. The failure of calling "setup()" and "loop()" in callback function mcu() will cause no input to EVE emulator.

```
a#include <bt8xxemu.h>
#include <stdio.h>

void setup();
void loop();

extern BT8XXEMU_Emulator *g_Emulator;

void mcu(BT8XXEMU_Emulator *sender, void *context)
{
        setup();
        while (BT8XXEMU_isRunning(g_Emulator))
                loop();
}

int main(int, char* [])
{
        BT8XXEMU_EmulatorParameters params;
        BT8XXEMU_defaults(BT8XXEMU_VERSION_API, &params, BT8XXEMU_EmulatorBT815);
        params.Main = mcu;
        BT8XXEMU_run(BT8XXEMU_VERSION_API, &g_Emulator, &params);
        return 0;
}
```

Figure 4 Setup and Run BT815 emulator

5. 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 3 Setup and Run BT815 emulator.

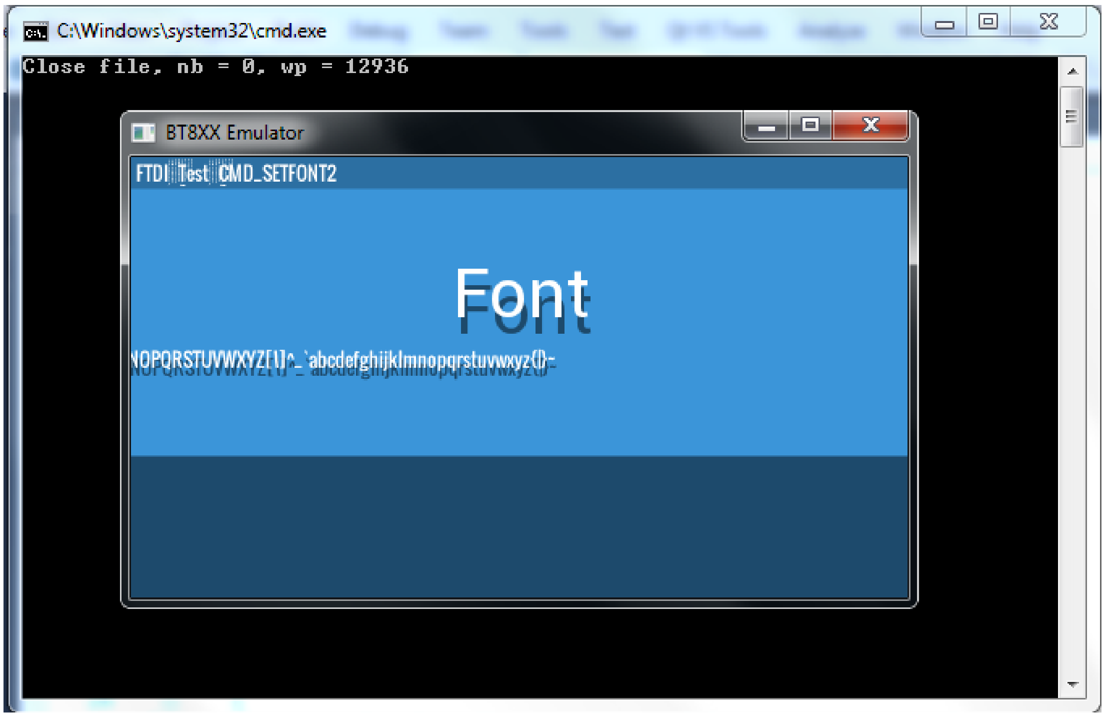
- ***Build and Run***

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.


C:\Windows\system32\cmd.exe
BT8XX Emulator
FTDI Test CMD_SETFONT2
Font
NOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

```
Close file, nb = 0, wp = 12936
```

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

2.3 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.

1. 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

2. 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 destroys the flash emulator instance which was created earlier.

- **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 flash's vtable data.

- **Return value**

uint8_t*	Returns the pointer to the address of the vtable 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 of the flash's vtable data.

- **Return value**

size_t	Returns an unsigned byte of the size of the vtable in flash.

- **Parameter**

flash	Pointer to the flash emulator instance.

3. 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

4. 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 using EVE emulator.

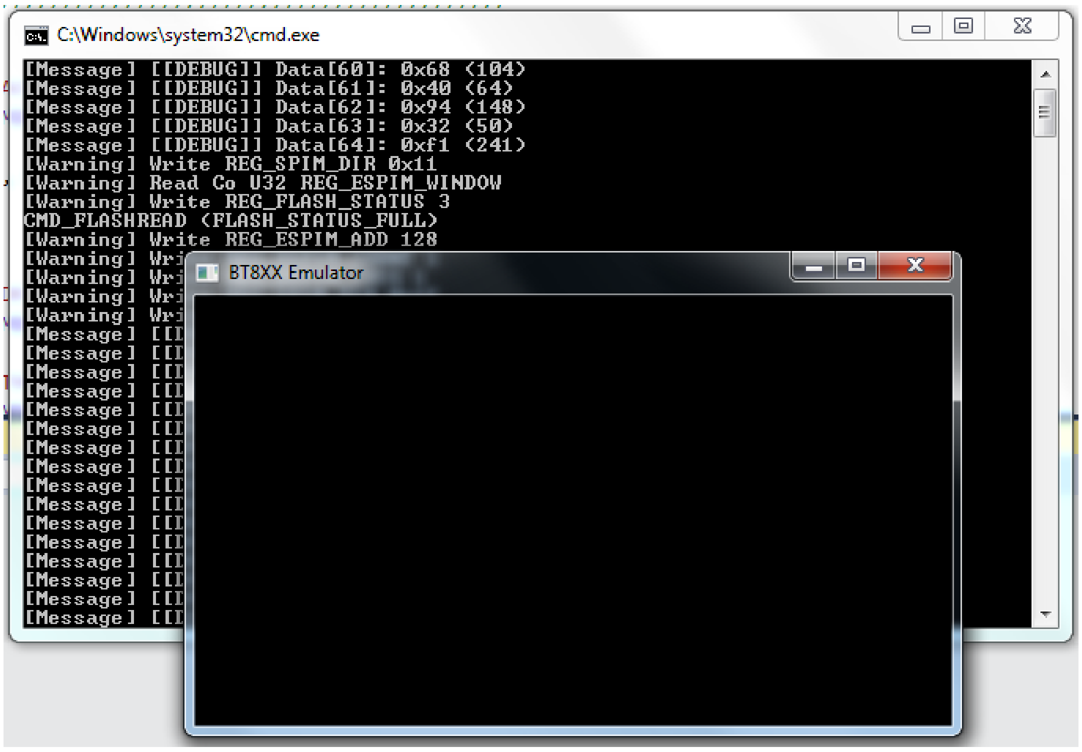
- ***Build and Run***

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.



```
[Message] [[DEBUG]] Data[60]: 0x68 (104)
[Message] [[DEBUG]] Data[61]: 0x40 (64)
[Message] [[DEBUG]] Data[62]: 0x94 (148)
[Message] [[DEBUG]] Data[63]: 0x32 (50)
[Message] [[DEBUG]] Data[64]: 0xf1 (241)
[Warning] Write REG_SPIM_DIR 0x11
[Warning] Read Co U32 REG_ESPIM_WINDOW
[Warning] Write REG_FLASH_STATUS 3
CMD_FLASHREAD (FLASH_STATUS_FULL)
[Warning] Write REG_ESPIM_ADD 128
[Warning] Wri
[Warning] Wri
[Warning] Wri
[Warning] Wri
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
[Message] [[I
```

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

3 Contact Information

Head Quarters – Singapore

Bridgetek Pte Ltd
178 Paya Lebar Road, #07-03
Singapore 409030
Tel: +65 6547 4827
Fax: +65 6841 6071

E-mail (Sales)	sales.apac@brtchip.com
E-mail (Support)	support.apac@brtchip.com

Branch Office – Taipei, Taiwan

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
E-mail (Support)	support.apac@brtchip.com

Branch Office - Glasgow, United Kingdom

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

E-mail (Sales)	sales.emea@brtichip.com
E-mail (Support)	support.emea@brtchip.com

Branch Office – Vietnam

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.apac@brtchip.com
E-mail (Support)	support.apac@brtchip.com

Web Site

http://www.brtchip.com

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

List of Figures

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

Revision	Changes	Date
1.0	Initial Release	2018-04-12

Revision Record Sheet

Authors	Mohamed Fysal
Filename	BRT_AN_035 EVE Emulator Library User Guide.docx

Revision	Date	Details
1.0	YYYY-MM-DD	

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**