



## Application Note

### BRT\_AN\_073

# ESD 4.10 Exported Project Porting Guide for STM32L4 Discovery Board and FreeRTOS

Version 1.0

Issue Date: 23-06-2021

This application note is intended as a guide for porting an **EVE Screen Designer (ESD) 4.10** exported project to a non-FT9xx based MCU platform. Users are expected to have knowledge of ESD 4.10 as well as **BT81x** and **STM32L4XX** MCU

Use of Bridgetek Pte 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 Pte harmless from any and all damages, claims, suits or expense resulting from such use.

#### **Bridgetek Pte Limited (BRTChip)**

178 Paya Lebar Road, #07-03 Singapore 409030

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

Web Site: <http://brtchip.com>

Copyright © Bridgetek Pte Limited

## **Table of Contents**

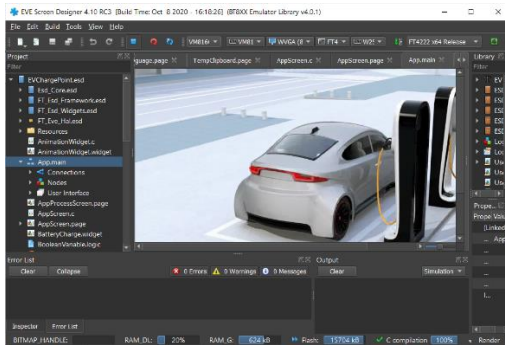
<b>1</b>	<b>Introduction .....</b>	<b>4</b>
1.1	Overview .....	4
1.2	Scope .....	4
<b>2</b>	<b>ESD 4.10 Exported Project - Introduction .....</b>	<b>5</b>
<b>3</b>	<b>Porting principles .....</b>	<b>7</b>
3.1	Hardware .....	7
3.2	Software.....	7
<b>4</b>	<b>Example.....</b>	<b>8</b>
4.1	Hardware Connection .....	9
4.2	Software Setup.....	9
4.2.1	Toolchain and Utility.....	9
<b>4.3</b>	<b>Project porting procedure .....</b>	<b>10</b>
4.3.1	Create project on ESD and generate source code for Eclipse IDE .....	11
4.3.2	Generate project for STM32L4 Discovery board .....	12
4.3.3	Port ESD exported project.....	16
4.3.4	Build and run .....	21
4.3.5	Storage Media Configuration and Access .....	22
4.3.6	APIs Re-Implementation .....	23
<b>5</b>	<b>Interrupt handling example.....</b>	<b>24</b>
5.1	STM32CubeMX configuration .....	24
5.2	Create ESD project .....	25
5.3	Modify source code.....	26
5.4	Build and run.....	28
<b>6</b>	<b>Contact Information .....</b>	<b>29</b>
	<b>Appendix A– References .....</b>	<b>30</b>
	Document References .....	30
	Acronyms and Abbreviations.....	30
	<b>Appendix B – List of Tables &amp; Figures .....</b>	<b>31</b>
	List of Figures .....	31

---

<b>List of Tables.....</b>	<b>32</b>
<b>Appendix C– Revision History .....</b>	<b>33</b>

## 1 Introduction

This application note is intended as a guide for porting an **EVE Screen Designer (ESD)** 4.10 exported project to an application on **FreeRTOS**, running on an ARM Cortex-M4 based MCU platform, i.e., STM32L476 Discovery board. In this document, an ESD 4.10 exported example project “EvChargePoint” and a **STM32L476 Discovery** board are used to showcase the porting procedure. Readers are expected to have the knowledge of ESD 4.10 as well as the **STM32** platform.



**Figure 1 - EVEChargePoint on ESD 4.10**



**Figure 2 - EVEChargePoint on EVE 4**

### 1.1 Overview

This guide covers the following topics:

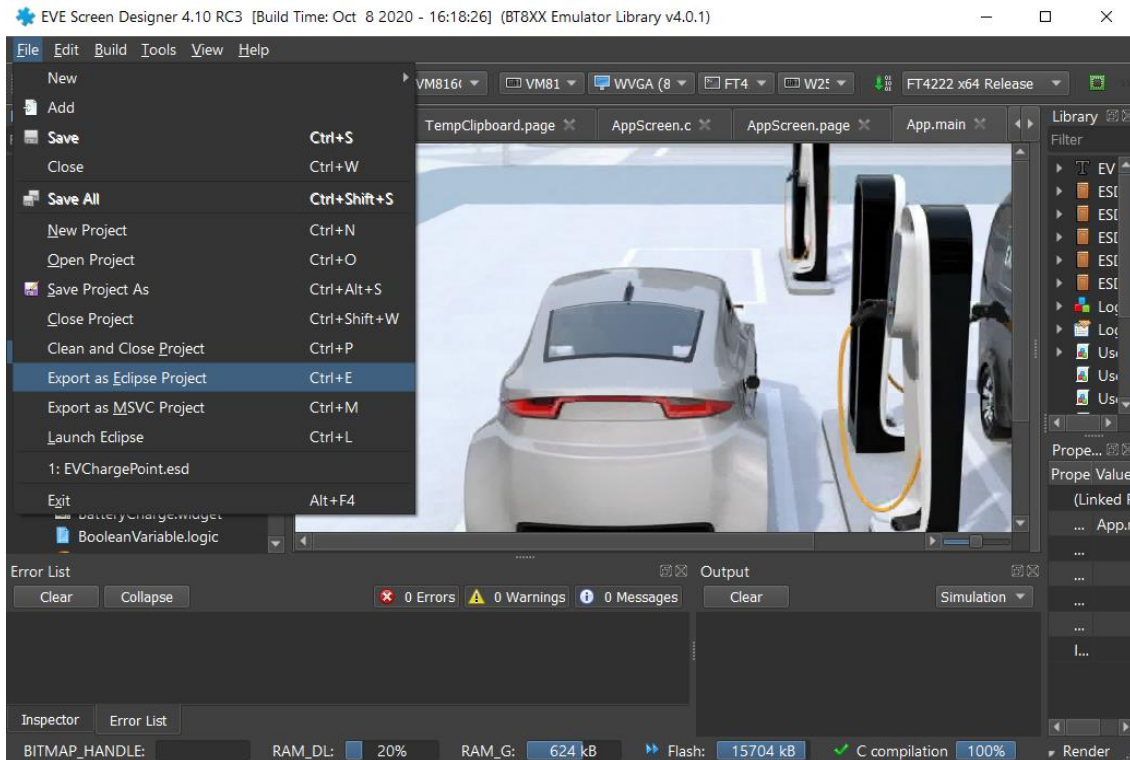
- ESD 4.10 exported project introduction
- Principles of porting
- Export and port **ESD** example project “EvChargePoint” to **STM32** platform
- Handle interrupt in an **ESD** project and its exported code

### 1.2 Scope

This document covers hardware setup and software modifications required as well as some debugging tips while porting the exported project. It also provides some basic principles to successfully port a project.

## 2 ESD 4.10 Exported Project - Introduction

ESD 4.10 enables users to design an **EVE** based GUI application with minimum effort. Upon completing the design and successfully simulating it on a PC, users can choose to export the currently opened project by selecting "**File → Export Project**", as shown below:



**Figure 3 - Export EvChargePoint Project in ESD 4.10**

The "EvChargePoint" project is located within the "\$ (ESD Installation Folder)\Examples\Advanced" folder.

Users are prompted to select an empty folder as the destination folder for the exported project. Once exporting has completed, the destination folder will contain the structure listed as below:

D:\...\evchargepoint_exported			
Name	Size Auto	Modified	Type
..		11/13/2020 3...	File folder
ThirdPartyLib		11/10/2020 1...	File folder
FT_Eve_Hal		11/10/2020 1...	File folder
FT_Esd_Widgets		11/10/2020 1...	File folder
FT_Esd_Framework		11/10/2020 1...	File folder
EVChargePoint		11/10/2020 1...	File folder
Esd_Core		11/10/2020 1...	File folder
Data		11/10/2020 1...	File folder
.settings		11/12/2020 4...	File folder
.project	82 kB	11/10/2020 1...	PROJECT File
.cproject	26 kB	11/12/2020 4...	CPROJECT File

**Figure 4 - Folder Structure of EvChargePoint exported Project**

Folder / file Name	Description	Remarks
ThirdPartyLib	Third party libraries	Currently only <b>FatFs</b> Library source code inside
FT_Eve_Hal	<b>EVE</b> hardware abstraction layer	The major folder to be modified when porting the project
FT_Esd_Widgets	Widgets-related source code	Reusable and common module
FT_Esd_Framework	Application framework source code	Reusable and common module
EVChargePoint	The screen logic and user design	The folder name should be the same as project name
ESD_Core	<b>ESD</b> specific library	Reusable and common module
Data	<b>Eve</b> specific assets	Read only data
.cproject	<b>Eclipse</b> CDT project file	Build configurations, tool chains, individual tools etc.
.project	<b>Eclipse</b> CDT project file	Build specification and build commands

**Table 1 - Folder Contents**

By default, the exported ESD 4.10 project supports FT90X series platform only. Therefore, the default project file works only in "Eclipse for FT90X" IDE which is part of the FT90x Toolchain.

## 3 Porting principles

### 3.1 Hardware

An ESD 4.10 exported project usually needs access to the following hardware resources of the MCU:

- **SPI interface:** Read/Write EVE Module
- **Clock:** Provide delay and timing control
- **Storage media:** Store Eve assets

Different **MCU** platform have different hardware configurations. Therefore, users need to ensure that the hardware components above work well. Users are assumed to be familiar with Eve series IC as well as its modules before starting the porting work.

### 3.2 Software

The following software modules are required to modify or add:

- **Project files:** Add configuration files of **MCU** toolchain to build the project.
- **Linker script:** Instruct the linker software to generate **MCU** specific executable.
- **Application code:** Rename the entry function `main()` of ESD exported project if the `main()` function has been defined in **FreeRTOS** and invoke it properly.
- **APIs:** Re-implement the **MCU** specific **APIs** of ESD exported project. They are all located in the files of folder **FT\_Eve\_Hal**, which Initialize the target **MCU** platform and implement the transportation layer. It must be modified manually by comparing to the reference project.



## 4 Example

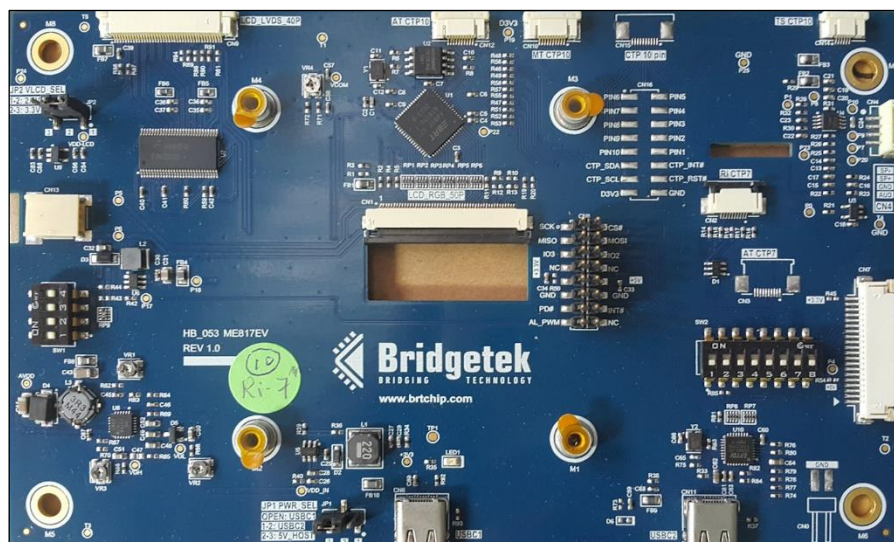
This example illustrates how to achieve porting an **ESD 4.10** project, concurring with the above stated principles.

The selected target **MCU** platform is an [STM32L476 Discovery board](#). It is connected to the development PC via a USB cable for downloading, debugging and power supply.



**Figure 5 - STM32L4 Discovery Board**

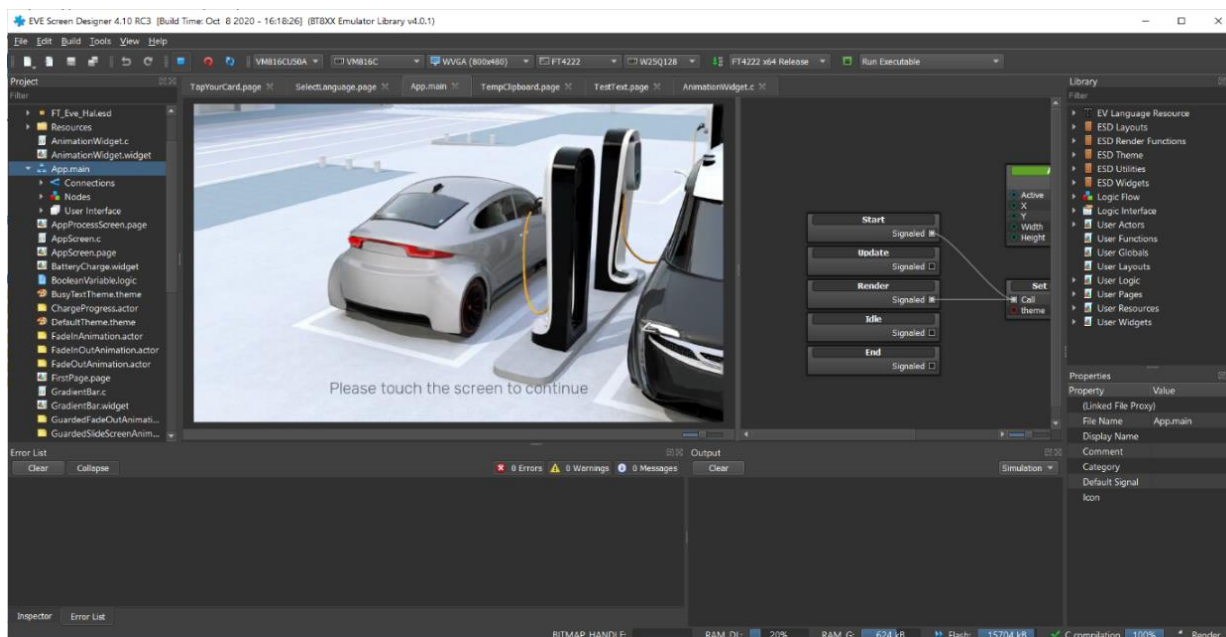
Selected EVE 4 Module is **ME817EV** (with BT817):



**Figure 6 - EVE 4 Module**

The "EvChargePoint" example project from the ESD 4.10 examples folder is used to demonstrate the porting procedure. It is located in the "\$(*ESD Installation Folder*)\Examples\Advanced" folder. Here is a screenshot when it is opened in ESD:





**Figure 7 - EvChargePoint Project Screenshot**

## 4.1 Hardware Connection

The **SPI1** interface of **STM32L4** is in use here and the following connections are assumed:

MCU Pin Name	MCU Function	EVE Pin name
PB2	GPIO	#PD
PE8	GPIO	#CS
PE13	SPI1_SCK	SCK
PE14	SPI1_MISO	MISO
PE15	SPI1_MOSI	MOSI
5v	5v	5v
GND	GND	GND

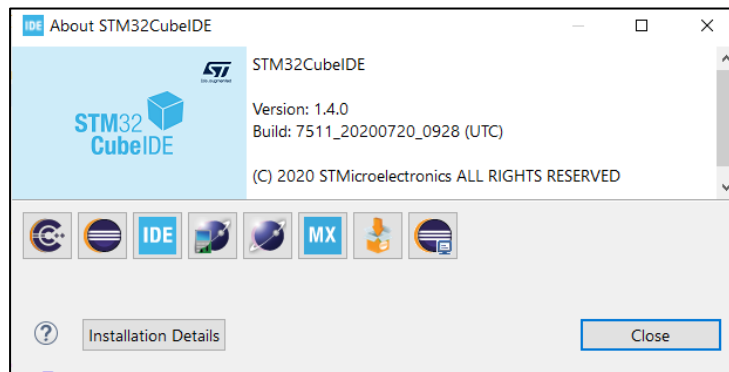
**Table 2 - MCU and EVE Connections**

Care must be taken if jumper wires are in use for connection. In such a case, it may be necessary to lower the frequency of the SPI clock by setting the **BR** bits of the **SPIx\_CR1** register to ensure a stable signal quality. In addition, please also ensure no other SPI devices are sharing the bus which may cause conflicts (such as SPI/QSPI flash memory ICs on the **MCU** board).

## 4.2 Software Setup

### 4.2.1 Toolchain and Utility

For this example, the [STM32CubeIDE](#) v1.4.0 is selected as the compiler and linker for the **STM32L4** MCU.



**Figure 8 - STM32CubeIDE version**

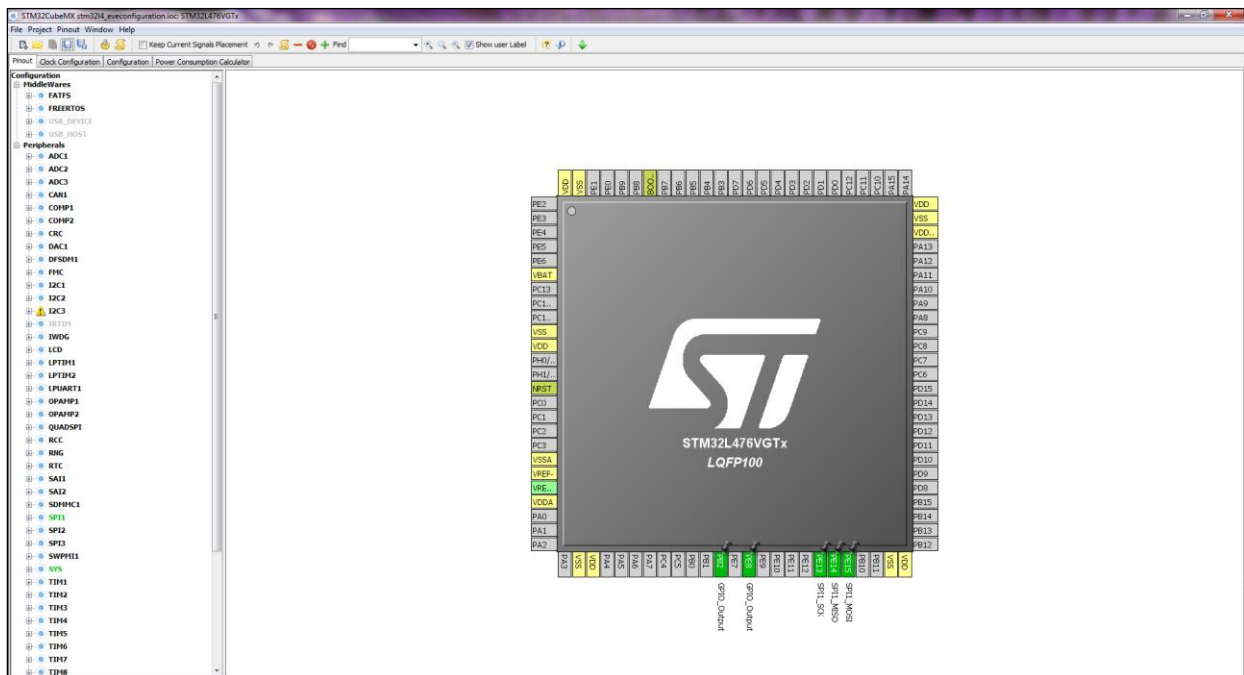
Another very helpful tool is **STM32CubeMX** which can be downloaded from [here](#). This tool allows users to configure pin functionality easily. In addition, it automatically generates the source code to configure hardware resources.

The following file is the project for **STM32CubeMX** tool which is used by the example project:



STM32L476\_Generat  
ed.ioc

Figure 9 shows the pin configuration once the project is opened using the **STM32CubeMX** tool.



**Figure 9 - STM32CubeMX Snapshot**

## 4.3 Project porting procedure

This section describes how to use **STM32CubeMX** and **STM32CubeIDE** to port an ESD exported project to **STM32L4** Discovery board.

Basically, we generate project for STM32L4 Discovery board by **STM32CubeMX** at first, then use this project to build the ESD generated source code on **STM32CubeIDE**.

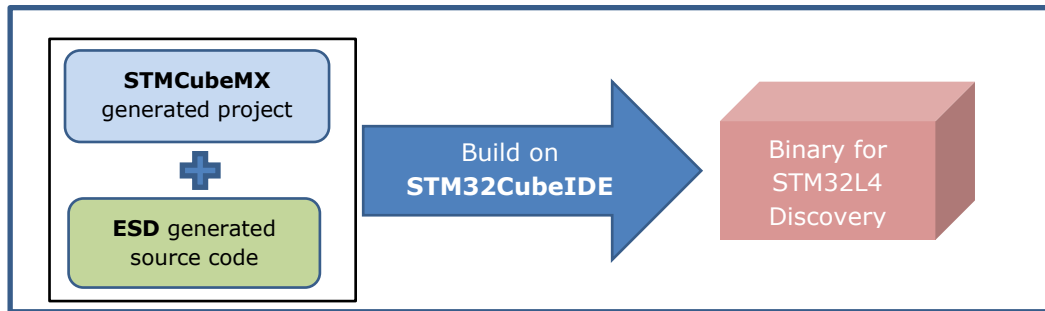


Figure 10 - Project porting procedure

### 4.3.1 Create project on ESD and generate source code for Eclipse IDE

1. In ESD, open the "EvChargePoint" project:  
 It is located at "\$(ESD4.10 Installation Path)\Examples\Advanced" folder.

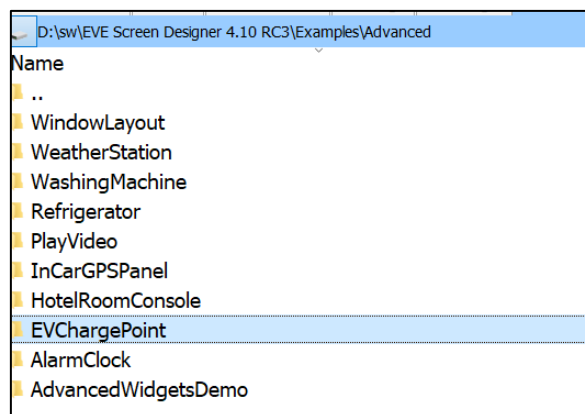


Figure 11 - The EvChargePoint project on ESD

2. Export the project to a local folder:  
 Select **File** → **"Export as Eclipse Project"** and choose an empty folder to store exported files.

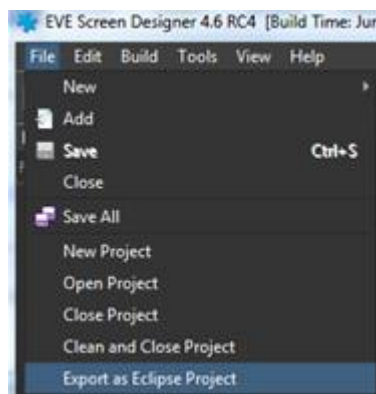
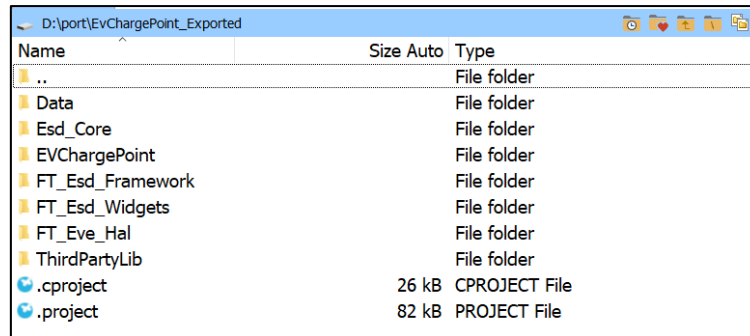


Figure 12 - Export as Eclipse Project

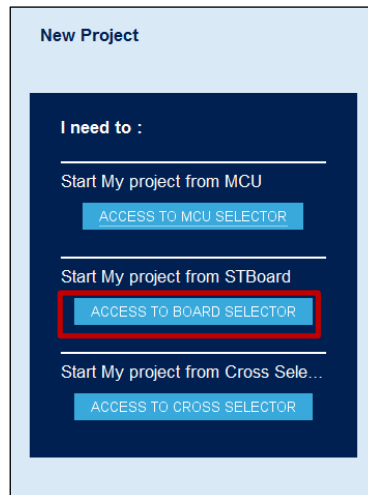
3. The exported project contains below files and folders originally for FT90X platform:



**Figure 13 - ESD exported project files and folders**

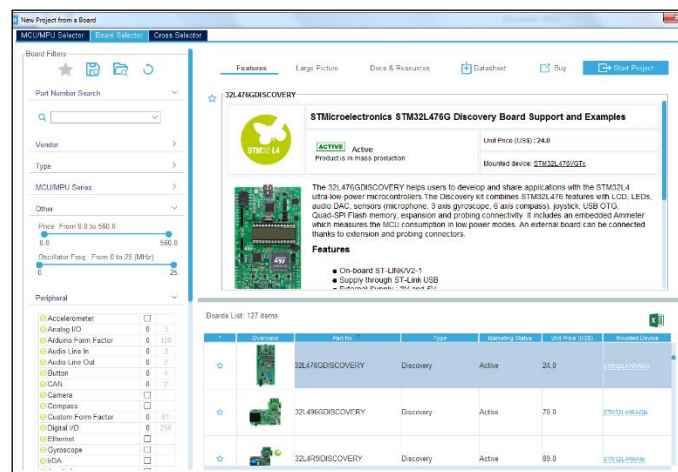
### 4.3.2 Generate project for STM32L4 Discovery board

1. Open **STM32CubeMX** and select "ASSESS TO BOARD SELECTOR":



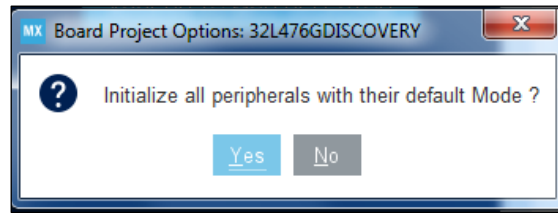
**Figure 14 - ACCESS TO BOARD SELECTOR**

2. Select **32L476GDISCOVERY** board:



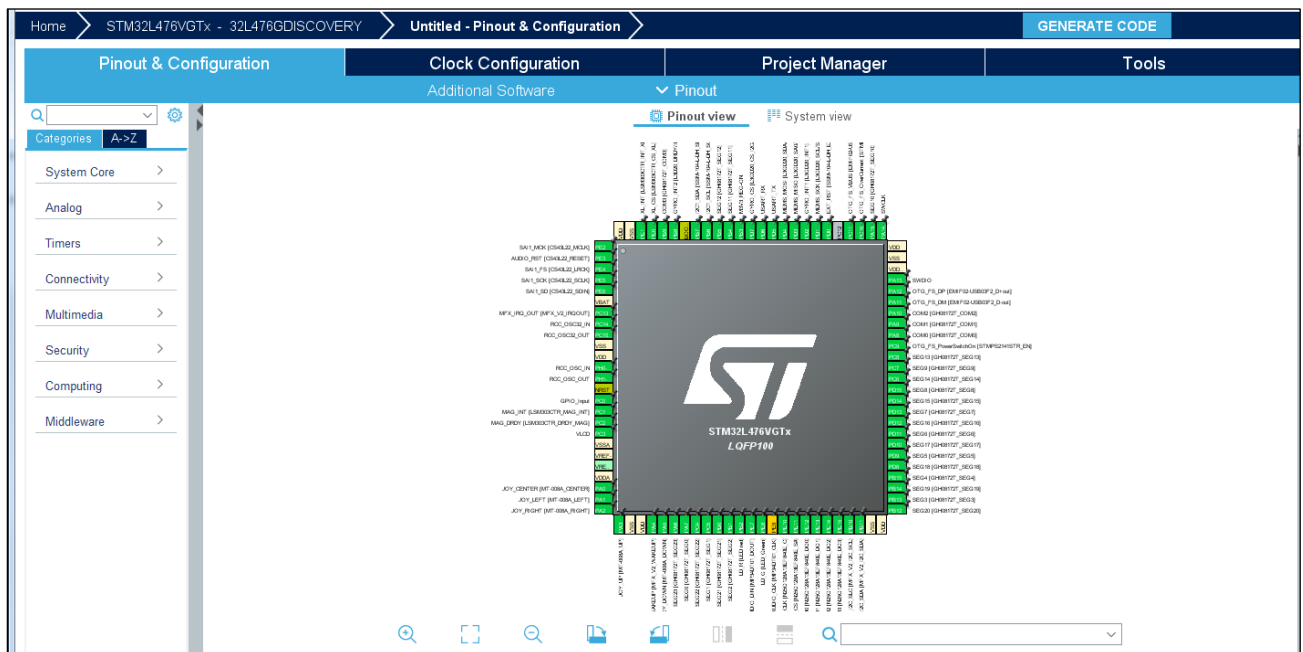
**Figure 15 Select 32L476GDISCOVERY board**

3. Select default setting for all peripherals:



**Figure 16 - select default mode**

4. The Pinout and configuration screen will appear:



**Figure 17 - Pinout and configuration screen**

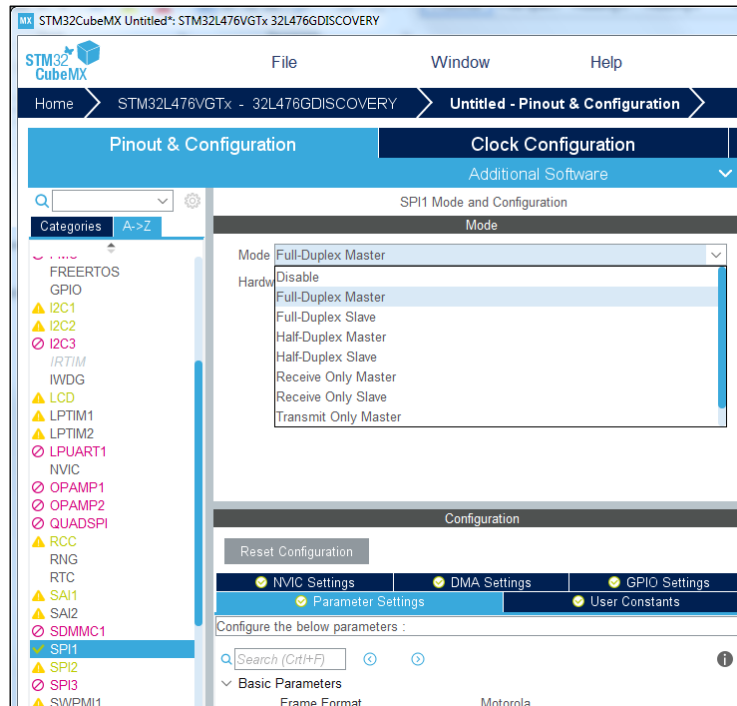
5. Set the PINs for SPI1 on the STM32L4 board:

- PE13 -> SPI1\_SCK
- PE14 -> SPI1\_MISO
- PE15 -> SPI1\_MOSI
- PE8 -> GPIO\_Output
- PB2 -> GPIO\_Output



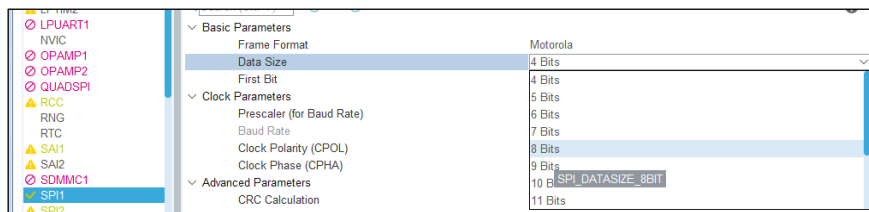
**Figure 18 - Select SPI ports**

- Set **SPI1** to Full-Duplex master mode



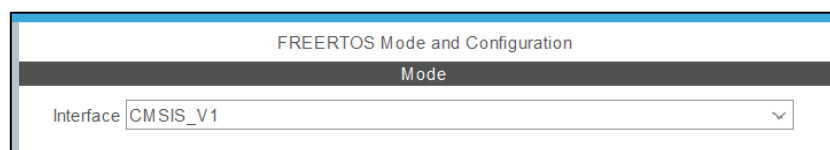
**Figure 19 - Set SPI1 to Full-Duplex master**

- set Data size = 8 bits for SPI1:



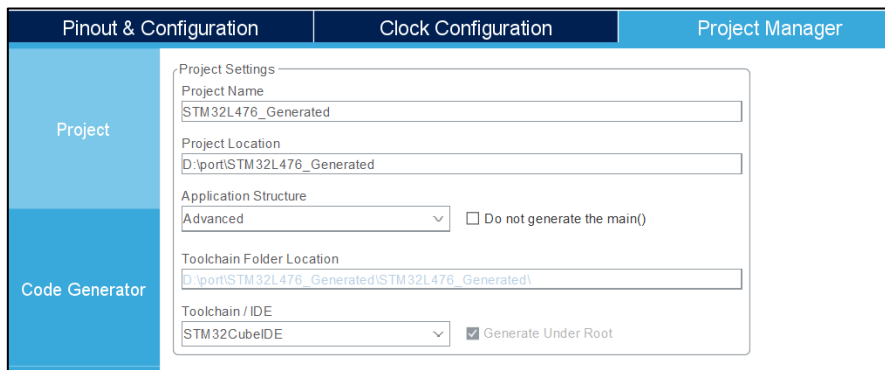
**Figure 20 - SPI1 – Select data size**

7. Enable **FreeRTOS** (group MiddleWare):



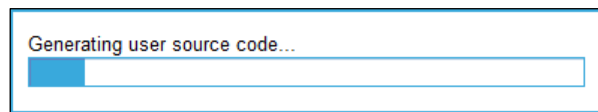
**Figure 21 - Enable FreeRTOS**

8. Generate code for 32L476GDISCOVERY board:
  - Select tab "Project manager"
  - Input the project name
  - Select the project location
  - Select "toolchain/IDE" as **STM32CubeIDE** as we will use it to open the generated project.
  - Select "Application structure" as "Advanced"



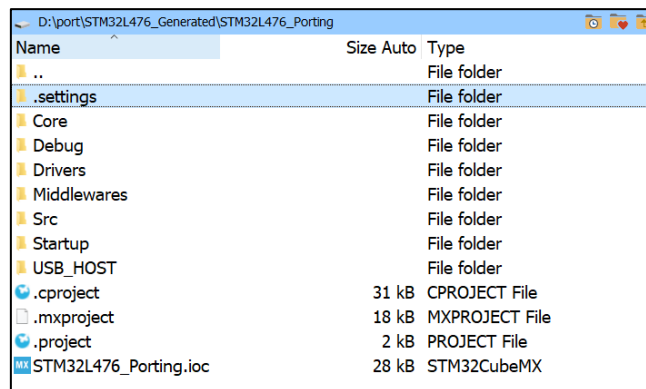
**Figure 22 – Generate code for 32L476GDISCOVERY board**

Click the “GENERATE CODE” button, and wait for completion:



**Figure 23 – Generate Code**

Once complete the project will have the following folder structure:



Name	Size	Type
..		File folder
.settings		File folder
Core		File folder
Debug		File folder
Drivers		File folder
Middlewares		File folder
Src		File folder
Startup		File folder
USB_HOST		File folder
.cproject	31 kB	CPROJECT File
.mxproject	18 kB	MXPROJECT File
.project	2 kB	PROJECT File
STM32L476_Porting.ioc	28 kB	STM32CubeMX

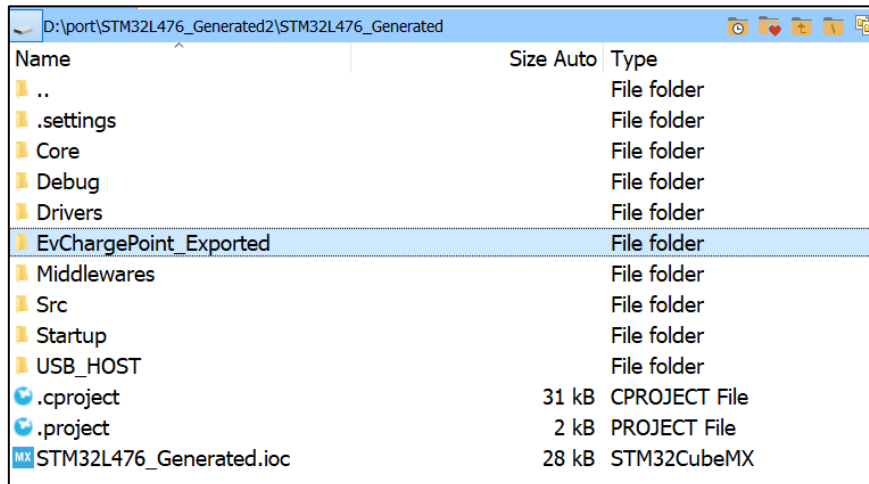
**Figure 24 -The generated project files**



### 4.3.3 Port ESD exported project

#### 4.3.3.1 Load generated STM32L4 Discovery project into STM32CubeIDE

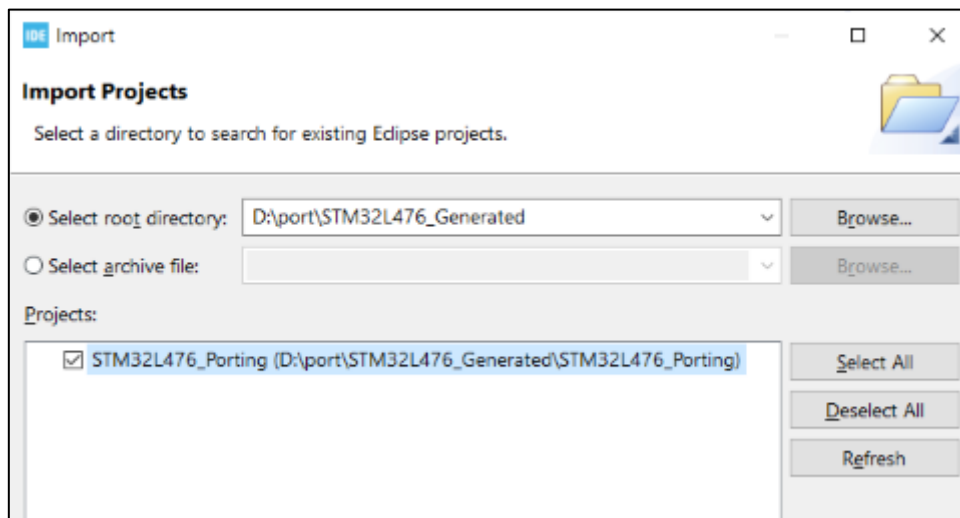
- Copy the "EvChargePoint\_Exported" folder into the "STM32L476\_Generated" folder:



**Figure 25 - Copy ESD exported folder to the generated project**

- Open generated STM32L4 Discovery project with **STM32CubeIDE**:

Select **File** → **"Import"**, choose folder **"STM32L476\_Generated"**



**Figure 26 - STM32CubeIDE - Open Projects from File System**

#### 4.3.3.2 Build configuration in STM32CubeIDE

- Add include path to ESD generated header files:
  - Right click on the project name,
  - Select Properties->C/C++ General -> path and Symbols -> includes tab,
  - Click "Add" button to add below include path:

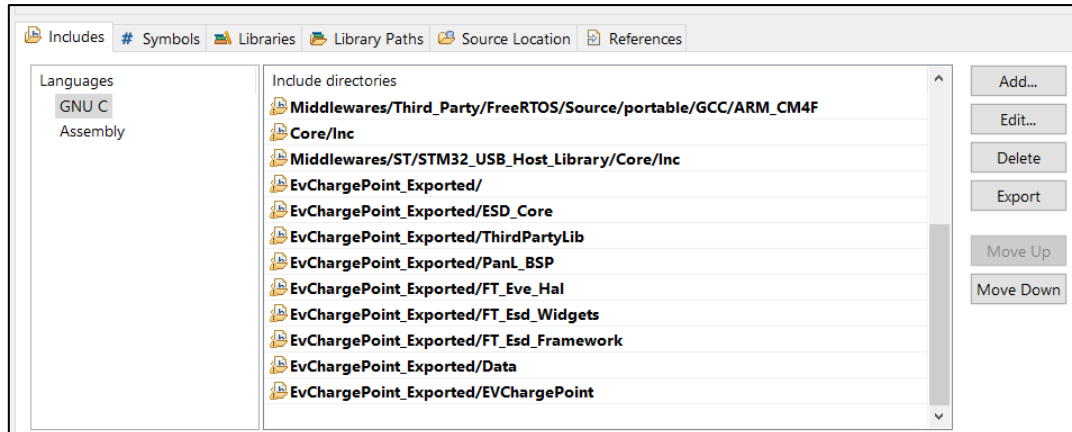
```

-                                     EvChargePoint_Exported/
-                                     EvChargePoint_Exported/ESD_Core
-                                     EvChargePoint_Exported/ThirdPartyLib
-                                     EvChargePoint_Exported/PanL_BSP
-                                     EvChargePoint_Exported/FT_Eve_Hal
-                                     EvChargePoint_Exported/FT_Esd_Widgets
-                                     EvChargePoint_Exported/FT_Esd_Framework

```

-  
 - *EvChargePoint\_Exported/EVChargePoint*

*EvChargePoint\_Exported/Data*

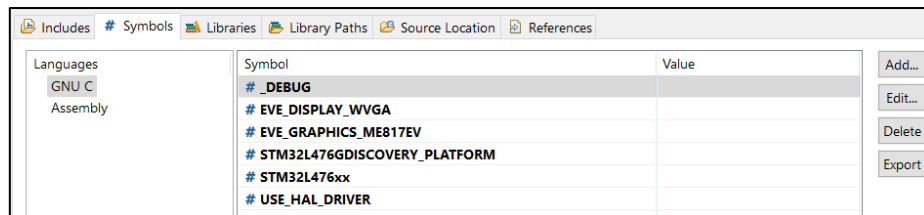


**Figure 27 - Add include path to ESD generated header files**

2. Add platform macro for EVE platform and **STM32L4** platform:

Select tab "Symbols" and add 3 macros:

- *STM32L476GDISCOVERY\_PLATFORM*
- *EVE\_GRAPHICS\_ME817EV*
- *EVE\_DISPLAY\_WVGA*



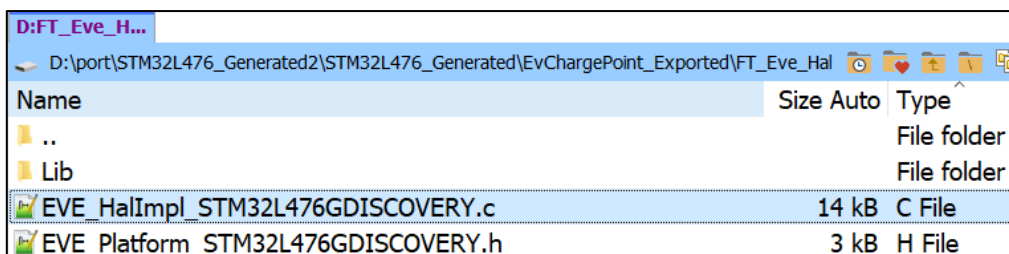
**Figure 28 - Add platform macro**

#### 4.3.3.3 Source code modification

1. Copy two files *EVE\_Platform\_STM32L476GDISCOVERY.h* and *EVE\_HalImpl\_STM32L476GDISCOVERY.c* into folder *FT\_Eve\_Hal*:

  
 EVE\_Platform\_STM3  
 2L476GDISCOVERY.h

  
 EVE\_HalImpl\_STM32  
 L476GDISCOVERY.c



**Figure 29 - Create source files for STM32L4 platform**

2. Include file *EVE\_Platform\_STM32L476GDISCOVERY.h* in *EVE\_Platform.h*:

```
D:\port\STM32L476_Generated\STM32L476_Generated\EvChargePoint_Exported\FT_Eve_Hal\EVE_Platform.h
54 #include "EVE_Platform_FT9XX.h"
55 #endif
56 #if defined(STM32L476GDISCOVERY_PLATFORM)
57 #include "EVE_Platform_STM32L476GDISCOVERY.h"
58 #endif
59 #include "EVE_GpuTypes.h"
60 #include "EVE_HalDefs.h"
```

**Figure 30 -Include *EVE\_Platform\_STM32L476GDISCOVERY.h***

- Rename *main* function to "*SD\_Start*":  
 We use the *main()* from FreeRTOS and rename *main()* function of ESD generated project.

```
D:\_6_Generated\STM32L476_Generated\EvChargePoint_Exported\EVChargePoint\App_Generated.c
132 }
133
134 int ESD_Start()
135 {
136     Esd_Initialize();
137
138 }
```

**Figure 31 - Rename *main* function**

- Disable QUAD-SPI mode for **ME817EV** platform because we use **SPI1** in single mode  
 Disable macro **FT4222\_PLATFORM** too:

```
D:\port\STM32L476_Generated\STM32L476_Generated\EvChargePoint_Exported\FT_Eve_Hal\EVE_Config.h
445
446 #define BT817_ENABLE
447 // #define ENABLE_SPI_QUAD
448
449 #ifndef EVE_DISPLAY_AVAILABLE
450 #define EVE_DISPLAY_AVAILABLE
451 #define DISPLAY_RESOLUTION_WVGA
452 #endif
453
454 #ifndef EVE_PLATFORM_AVAILABLE
455 #define EVE_PLATFORM_AVAILABLE
456 // #define FT4222_PLATFORM
457 #endif
```

**Figure 32 - Disable QUAD-SPI mode for ME817EV platform**

- Define *EVE\_HOST* macro in *EVE\_Config.h*

```
D:\port\STM32L476_Generated\STM32L476_Generated\EvChargePoint_Exported\FT_Eve_Hal\EVE_Config.h
878
879 #endif
880
881 #if defined(STM32L476GDISCOVERY_PLATFORM)
882 #define EVE_HOST EVE_HOST_STM32L476GDISCOVERY
883 #endif
884
885
```

**Figure 33 – Define *EVE\_HOST* macro in *EVE\_Config.h***

Add new host name *EVE\_HOST\_STM32L476GDISCOVERY*:

```
D:\port\STM32L476_Generated\STM32L476_Generated\EvChargePoint_Exported\FT_Eve_Hal\EVE_HalDefs.h
125
126 typedef enum EVE_HOST_T
127 {
128     EVE_HOST_UNKNOWN = 0,
129     EVE_HOST_BT9XXEMU,
130     EVE_HOST_FT4222,
131     EVE_HOST_MPSSE,
132     EVE_HOST_FT9XX,
133     EVE_HOST_STM32L476GDISCOVERY,
134
135     EVE_HOST_NB
136 } EVE_HOST_T;
137
```

**Figure 34 - Add new host platform *EVE\_HOST\_STM32L476GDISCOVERY***

6. Add `M_PI` macro definition for **STM32** platform to avoid compilation error:

```
D:\port\STM32L476_Generated\STM32L476_Generated\EvChargePoint_Exported\Esd_Core\Esd_Math.h
33
34 #include "Esd_Base.h"
35 #define M_PI 3.14159265358979323846
36
```

**Figure 35 - Add M\_PI definition**

7. Enable `LoadFile` functions:

```
D:\port\STM32L476_Generated\STM32L476_Generated\EvChargePoint_Exported\FT_Eve_Hal\EVE_LoadFile_STDI.c
32 #include "EVE_LoadFile.h"
33 #include "EVE_Platform.h"
34 #if !defined(FT9XX_PLATFORM) && !defined(STM32L476GDISCOVERY_PLATFORM)
35
36 #include <stdio.h>
37
```

**Figure 36 - Enable LoadFile functions**

8. Enable external clock for Eve module:

```
D:\port\STM32L476_Generated\STM32L476_Generated\EvChargePoint_Exported\FT_Eve_Hal\EVE_Util.c
250
251 #if !defined(ME810A_HV35R) && !defined(ME812A_WH50R) && !defined(ME813A_W
252 /* Board without external oscillator will not work when ExternalOsc i
253 bootup->ExternalOsc = true;
254 #endif
255 #if defined(STM32L476GDISCOVERY_PLATFORM)
256 bootup->ExternalOsc = true;
257 #endif
258
```

**Figure 37 - Configure EVE platform to use external clock**

9. Include `stddef.h`, `stdio.h` and `stdarg.h` in `EVE_Config.h`:

These header files are required to use `NULL`, `va_list` and `va_arg` in the ESD generated source code.

```
EVE_Config.h
22 * the use of money or anticipated savings; loss of info
23 * opportunity; loss of goodwill or reputation; and/or l
24 * corruption of data.
25 * There is a monetary cap on Bridgetek's liability.
26 * The Software may have subsequently been amended by an
27 * distributed by that other user ("Adapted Software").
28 * have additional licence terms that apply to those am
29 * has no liability in relation to those amendments.
30 */
31
32 #ifndef EVE_CONFIG_H
33 #define EVE_CONFIG_H
34
35 #include "EVE_IntTypes.h"
36 #include <stddef.h> // for NULL
37 #include <stdio.h> // for va_list
38 #include <stdarg.h> // for va_arg
39
40 //
```

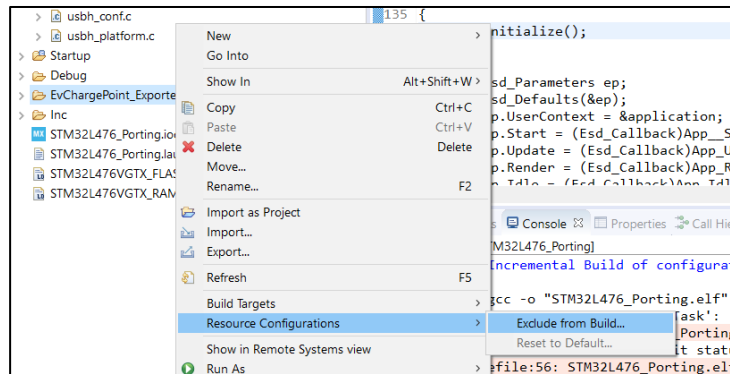
**Figure 38 - Include stddef.h, stdio.h and stdarg.h in EVE\_Config.h**

10. Increase stack size to `512` or any other size the application requires:  
 By default it is `128`, which is not enough to run the example application **EvChargePoint**.

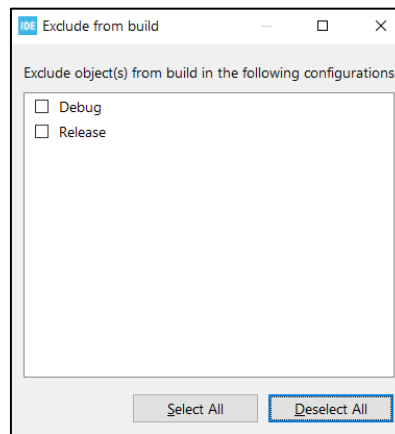
```
STM32L476_Generated2 (in STM32L476_Generated)
> Binaries
> Includes
> Core
> Inc
> Src
> freertos.c
> main.c
> stm32l4xx_hal_msp.c
> stm32l4xx_it.c
> system_stm32l4xx.c
> main.cbak
> Drivers
136 /* USER CODE BEGIN RTOS_QUEUES */
137 /* add queues, ... */
138 /* USER CODE END RTOS_QUEUES */
139
140 /* Create the thread(s) */
141 /* definition and creation of defaultTask */
142 osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 512);
143 defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);
144
145 /* USER CODE BEGIN RTOS_THREADS */
146 /* add threads, ... */
147 /* USER CODE END RTOS_THREADS */
148
149 /* Start scheduler */
150
```

**Figure 39 - Increase stack size**

11. Add folder *EvChargePoint\_Exported* into the compilation process, by default it is excluded:
  - Right click on *EvChargePoint\_Exported*, select **"Resource configuration"** → **"Exclude from build"**

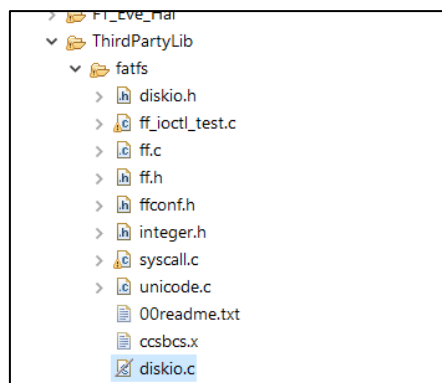


- Deselect all checkbox



**Figure 40 - Add EvChargePoint\_Exported to resource**

12. Exclude *diskio.c* from the compilation to avoid compilation error:



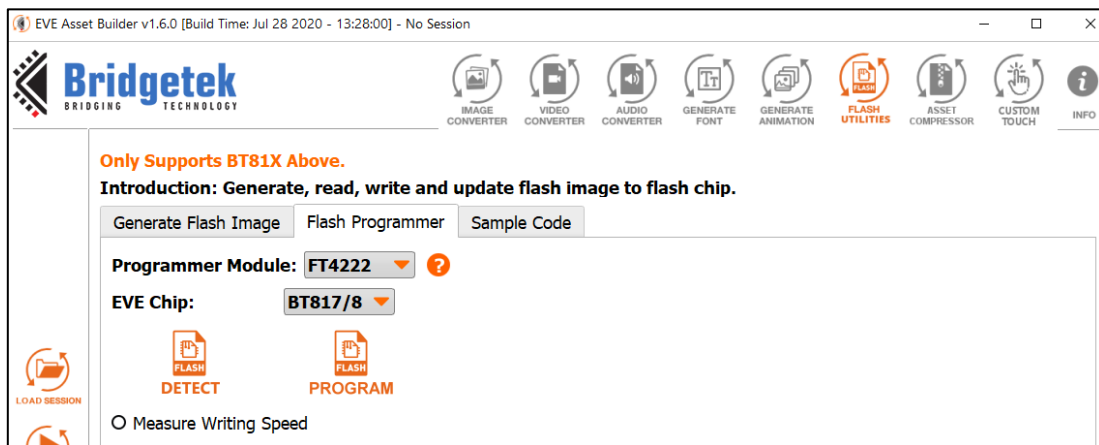
**Figure 41 - Exclude diskio.c**

### 4.3.4 Build and run

Before running, the user must program flash image file "EvChargePoint\_Exported\Data\\_\_Flash.bin" into **Eve** connected flash via the **EVE Asset Builder (EAB)**. User can download **EAB** at <https://brtchip.com/eve-toolchains>

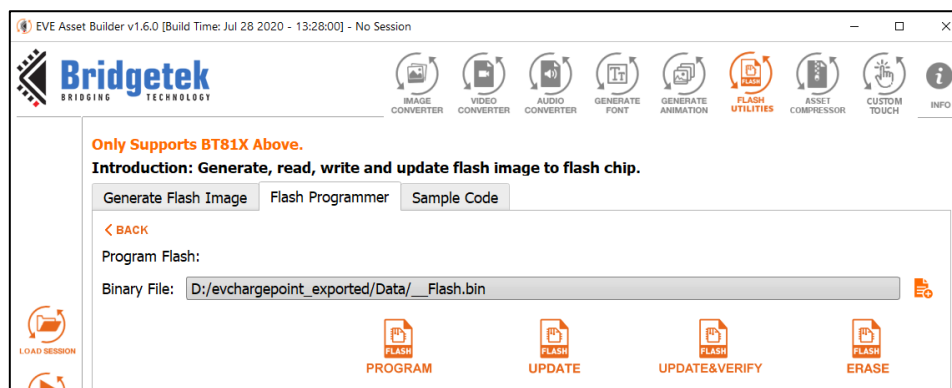
The steps to programm flash image file into **Eve** connected flash:

- Use MPSSE or FT4222 device to connect your PC with EVE 4 module (ME817EV board)
- Open EAB and select an interface: MPSSE or FT4222
- Select EVE chip as BT817
- Click button "**PROGRAM**"



**Figure 42 - Start EAB and select interface**

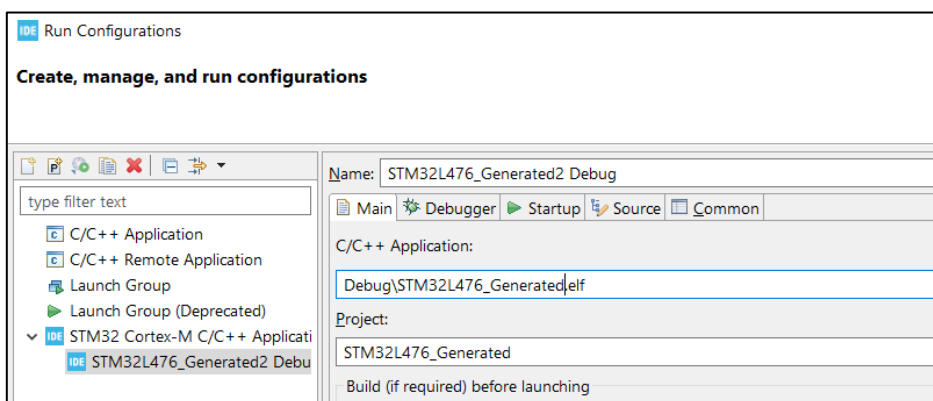
- Program \_\_Flash.bin into Eve 4 module:



**Figure 43 - Select \_\_Flash.bin and click button "Update"**

When the update complete, run the application as below:

- Select Run->"Run Configuration"
- Add new configuration
- Click run button



**Figure 44 - Run with STM32 MCU configuration**

The application should run after that:



**Figure 45 - EvChargePoint screen on LCD**

### 4.3.5 Storage Media Configuration and Access

In this example, the storage media is not enabled to simplify the procedure. To access Eve assets, users need to re-implement the functions below in the file "*FT\_Eve\_Hal\EVE\_LoadFile\_FATFS.c*" :

API name	Remarks	Note
<i>Ft_Hal_LoadImageFile</i>	Loads the image file from storage media with specified "filename" to <b>EVE RAM_G</b> "address" and sends the data through coprocessor command " <b>CMD_LOADIMAGE</b> ".	Keep it as empty or define its own implementation
<i>Ft_Hal_LoadInflateFile</i>	Loads the deflated file from storage media with specified "filename" to <b>EVE RAM_G</b> "address" and sends the data through coprocessor command " <b>CMD_INFLATE</b> ".	Keep it as empty or define its own implementation
<i>Ft_Hal_LoadRawFile</i>	Loads the raw image file from storage media with specified "filename" to <b>EVE RAM_G</b> "address".	Keep it as empty or define its own implementation
<i>Ft_Hal_LoadSDCard</i>	Initialize SD card interface.	Keep it as empty or define its own implementation



These four functions are kept empty here because **STM32L476G** Discovery board does not have internal flash memory or SD card reader.

### 4.3.6 APIs Re-Implementation

To make the **MCU** boot up and communicate with its peripheral, the following **APIs** are required to be re-implemented in the *EVE\_HalImpl\_STM32L476GDISCOVERY.c* file.

API name	Remarks	Note
<i>EVE_millis</i>	Get the current system tick	
<i>EVE_sleep</i>	Delay the specified amount of time	
<i>EVE_Mcu_release</i>	Release the MCU and its peripheral (SPI, GPIO)	
<i>EVE_Mcu_initialize</i>	Initialize the MCU and its peripheral (SPI, GPIO)	
<i>EVE_Millis_initialize</i>	Init MCU's timer	
<i>EVE_Millis_release</i>	Release MCU's timer	
<i>EVE_HalImpl_initialize</i>	Initialize HAL platform	
<i>EVE_HalImpl_release</i>	Release HAL platform	
<i>EVE_HalImpl_defaults</i>	Set the default configuration parameters	
<i>EVE_HalImpl_close</i>	Close a HAL context	
<i>EVE_HalImpl_idle</i>	Idle callback function. Call regularly to update frequently changing internal state	
<i>EVE_Hal_startTransfer</i>	Initiate address phase by transmitting 3 bytes address code and assert CS	For SPI reading, there are 3 dummy bytes to be returned and discarded.
<i>EVE_Hal_endTransfer</i>	De-assert the CS to end the SPI transferring	
<i>EVE_Hal_flush</i>	Flush data to Coprocessor	
<i>EVE_Hal_transfer8</i>	Send or receive one byte	
<i>EVE_Hal_transfer16</i>	Send or receive 2 bytes	
<i>EVE_Hal_transfer32</i>	Send or receive 4 bytes	
<i>EVE_Hal_hostCommand</i>	Send host commands to EVE	
<i>EVE_Hal_hostCommandExt3</i>	Send 3 bytes host commands to EVE	
<i>EVE_Hal_powerCycle</i>	Toggle PD pin to wake up EVE	The delay after each toggle is mandatory
<i>EVE_Util_closeFile</i>	Close opened file	
<i>EVE_Util_loadSdCard</i>	Mount the SD card	
<i>EVE_Util_sdCardReady</i>	Check if SD card is ready	
<i>EVE_Util_loadImageFile</i>	Load the image file from storage into EVE RAM_G	Keep it as empty or define its own implementation
<i>EVE_Util_loadInflateFile</i>	Load the compressed asset file from storage into EVE RAM_G	Keep it as empty or define its own implementation
<i>EVE_Util_loadRawFile</i>	Load the raw data file from storage into EVE RAM_G	Keep it as empty or define its own implementation
<i>EVE_Util_loadSdCard</i>	Load the file from SD card into EVE RAM_G	Keep it as empty or define its own implementation

**Table 3 - APIs to be re-implemented**

The changes above may not be optimal for the SPI transfer performance because the primary target is to keep the code structure working on different platforms. Users are encouraged to read the code thoroughly and optimize the transfer performance by sending more data for each SPI transaction.

## 5 Interrupt handling example

To handle the interrupt from **MCU**, users shall introduce the interrupt handler and update a global variable to capture the changes. When the **GUI** thread is scheduled, the UI will be rendered according to the updated value in the global variable.

In this example, a timer interrupt from **STM32** is captured into a global counter and displayed on an ESD clock and an ESD label.

### 5.1 STM32CubeMX configuration

Configuration for STM32 is the same as section [4.3.2](#), with an additional configuration for timer. We use **TIM2** in this example.

The timeout is calculated by this formula:

$$\text{Timeout} = \text{Prescaler} * \text{Counter period} / \text{HCLK}.$$

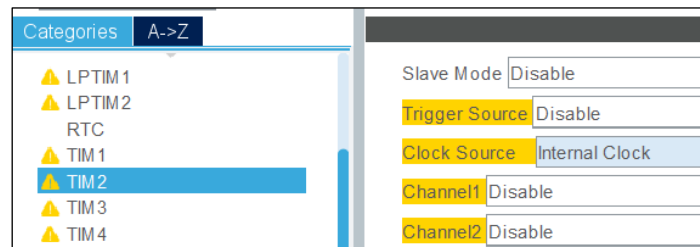
By default, on the Discovery board HCLK is 20 MHz, we want Timeout = 1 second, and choose Prescaler = 1000, so Counter period would be 20000.



**Figure 46 - Default HCLK configuration**

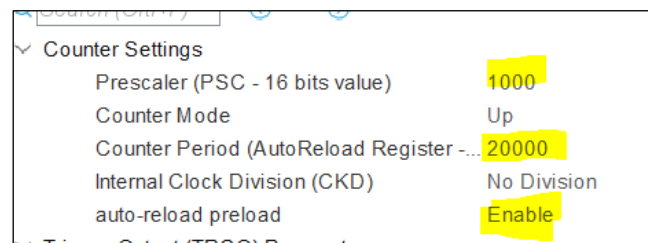
Below are the steps to enable **TIM2**:

- Select clock source as Internal clock:



**Figure 47 - Select TIM2 clock source**

- Select Prescaler and Counter period as 1000 and 20000, enable auto reload:



**Figure 48 - Setup TIM2 parameters**

- Enable **TIM2** global interrupt:

Parameter Settings	User Constants	NVIC Settings
NVIC Interrupt Table		
		Enabled
		Preemption
TIM2 global interrupt		<input checked="" type="checkbox"/>
		5

**Figure 49 - Setup TIM2 interrupt**

- Save and export project:

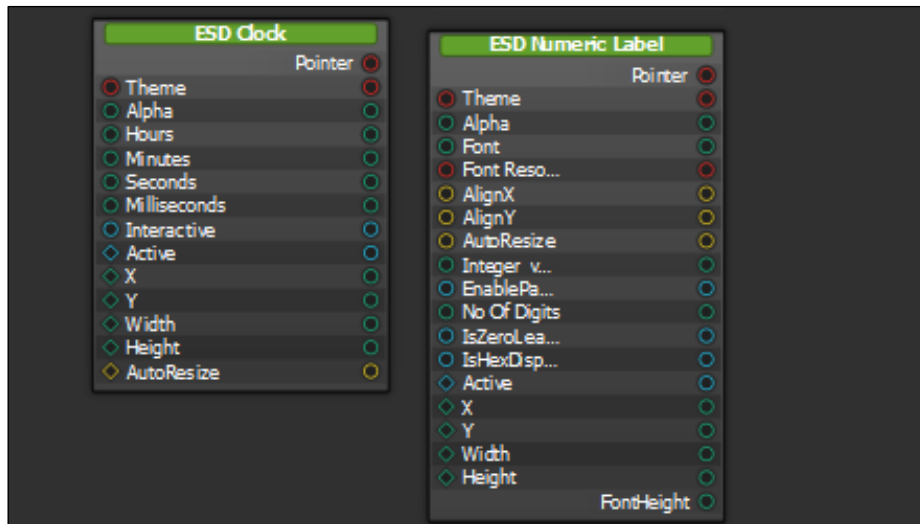
Reference:



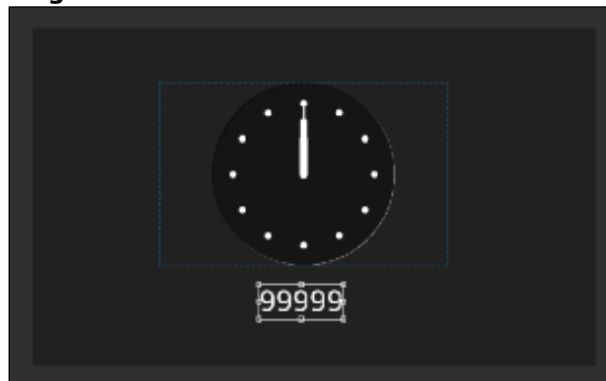
stm32Exported.ioc

## 5.2 Create ESD project

In ESD, create new project and add a clock and a numeric label into it:

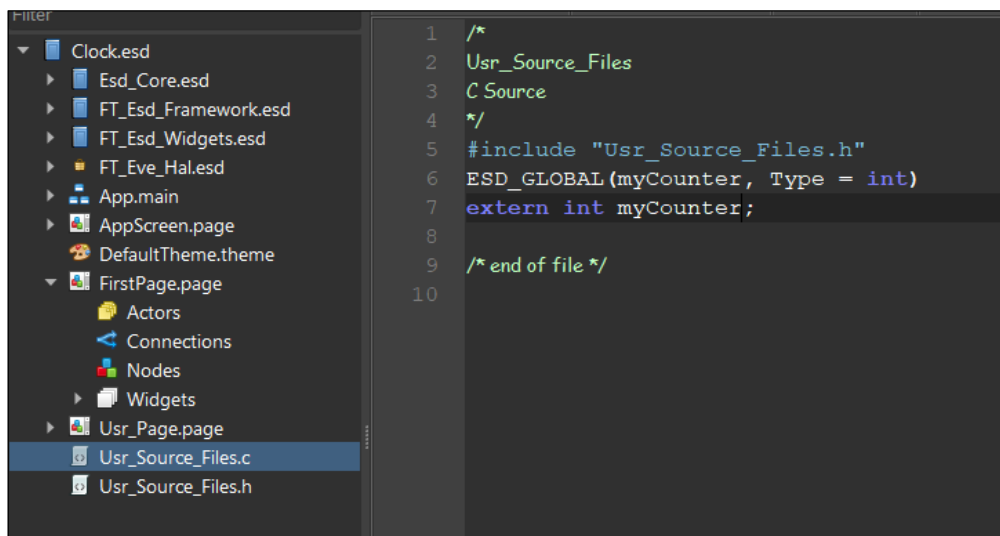


**Figure 50 - Create ESD clock and ESD label**



**Figure 51 - New project in ESD**

Add a new source file and declare a global variable "myCounter" into it:



**Figure 52 - Declare global variable in ESD**

Variable "myCounter" will appear in "User Globals" tree, now connect global variable "myCounter" with **ESD** clock and label widget:



**Figure 53 - Connect global variable and clock/label**

Save and export the project.

Reference:



Usr\_Source\_Files.c

## 5.3 Modify source code

We copy the **ESD** exported project into a **STM32** generated project and complete the porting similar to section [4.3.3](#).

Now modify the "main.c" file to define global variable "myCounter" and implement function to capture interrupt:

```

182 int myCounter;
183 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef* htim)
184 {
185     myCounter++;
186     char str[10];
187
188     sprintf(str, "%d", myCounter);
189     BSP_LCD_GLASS_DisplayString(str);
190
191 }

```

**Figure 54 - Implement interrupt handling function**

In the main function, enable the **TIM2** interrupt, by calling `HAL_TIM_Base_Start_IT()`:

```

156 osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 512);
157 defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);
158
159 /* USER CODE BEGIN RTOS_THREADS */
160 HAL_TIM_Base_Start_IT(&htim2);
161 /* add threads, ... */
162 /* USER CODE END RTOS_THREADS */
163
164 /* Start scheduler */
165 osKernelStart();
166

```

**Figure 55 - Enable interrupt timer**

This step is optional, we can display the interrupt counter on a glass display of *STM32L476* discovery board, using library in *STM32L476* Discovery **BSP**, to do so we need to initialize glass display in main function:

```

129 MX_USART2_UART_Init();
130 /* USER CODE BEGIN 2 */
131
132 // Initialize glass display
133 BSP_LED_Init(LED_GREEN);
134 BSP_LCD_GLASS_Init();
135 BSP_LCD_GLASS_Contrast(LCD_CONTRASTLEVEL_5);
136
137 /* USER CODE END 2 */

```

**Figure 56 - Initialize glass display**

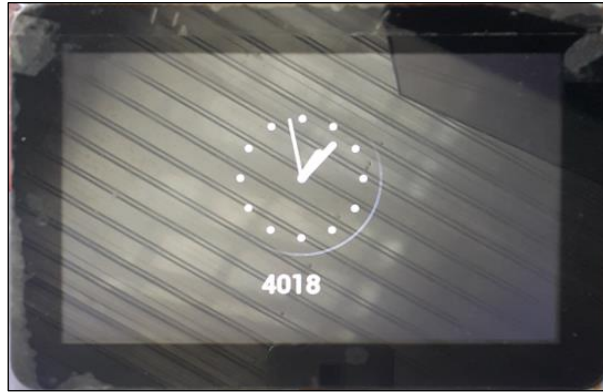
Please refer to main.c here:



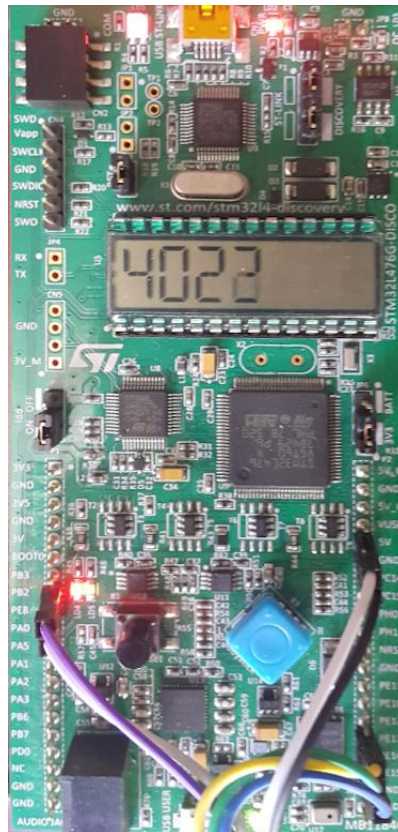
main.c

## 5.4 Build and run

Follow section [4.3.4](#) to run the project, the result is as below:



**Figure 57 - ESD clock screen**



**Figure 58 – Timer interrupt from STM32L4 board**

## 6 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](mailto:sales.apac@brtchip.com)  
E-mail (Support) [support.apac@brtchip.com](mailto:support.apac@brtchip.com)

### Branch Office – Taipei, Taiwan

Bridgetek Pte Ltd, Taiwan Branch  
2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District  
Taipei 114  
Taiwan, R.O.C.  
Tel: +886 (2) 8797 5691  
Fax: +886 (2) 8751 9737

E-mail (Sales) [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com)  
E-mail (Support) [support.apac@brtchip.com](mailto: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@brtchip.com](mailto:sales.emea@brtchip.com)  
E-mail (Support) [support.emea@brtchip.com](mailto:support.emea@brtchip.com)

### Branch Office – Vietnam

Bridgetek VietNam Company Limited  
Lutaco Tower Building, 5th Floor, 173A Nguyen Van  
Troï,  
Ward 11, Phu Nhuan District,  
Ho Chi Minh City, Vietnam  
Tel : 08 38453222  
Fax : 08 38455222

E-mail (Sales) [sales.apac@brtchip.com](mailto:sales.apac@brtchip.com)  
E-mail (Support) [support.apac@brtchip.com](mailto:support.apac@brtchip.com)

### Web Site

<http://brtchip.com/>

### Distributor and Sales Representatives

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

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Bridgetek Pte 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

### Document References

[STM32L4 Reference Manual](#)

[STM32L476xx datasheet](#)

[User Manual of STM32L4 Discovery board](#)

[BT81x Programmers Guide](#)

[BT81x Datasheet](#)

### Acronyms and Abbreviations

Terms	Description
BSP	Board Support Package
Eclipse	An integrated development environment (IDE) used in computer programming. Please refer to: <a href="https://www.eclipse.org/ide/">https://www.eclipse.org/ide/</a>
EVE	Embedded Video Engine
EVE Module	Eve based display module
EVE 4 Module	BT817/8 based display module
FT900	FT900 Microcontroller from FTDI
FreeRTOS	A real-time operating system kernel for embedded devices. Please refer to: <a href="https://www.freertos.org">https://www.freertos.org</a>
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
ESD 4.10	EVE Screen Designer 4.10
STM32	A family of 32-bit microcontroller integrated circuits by STMicroelectronics. Please refers to: <a href="https://www.st.com/content/st_com/en.html">https://www.st.com/content/st_com/en.html</a>

## Appendix B – List of Tables & Figures

### List of Figures

Figure 1 - EVEChargePoint on ESD 4.10	Figure 2 - EVEChargePoint on EVE 4	4
Figure 3 - Export EvChargePoint Project in ESD 4.10		5
Figure 4 - Folder Structure of EvChargePoint exported Project		5
Figure 5 - STM32L4 Discovery Board		8
Figure 6 - EVE 4 Module		8
Figure 7 - EvChargePoint Project Screenshot		9
Figure 8 - STM32CubeIDE version		10
Figure 9 - STM32CubeMX Snapshot		10
Figure 10 - Project porting procedure		11
Figure 11 - The EvChargePoint project on ESD		11
Figure 12 - Export as Eclipse Project		11
Figure 13 - ESD exported project files and folders		12
Figure 14 - ACCESS TO BOARD SELECTOR		12
Figure 15 Select 32L476GDISCOVERY board		12
Figure 16 - select default mode		13
Figure 17 - Pinout and configuration screen		13
Figure 18 - Select SPI ports		13
Figure 19 - Set SPI1 to Full-Duplex master		14
Figure 20 - SPI1 – Select data size		14
Figure 21 - Enable FreeRTOS		14
Figure 22 – Generate code for 32L476GDISCOVERY board		15
Figure 23 – Generate Code		15
Figure 24 -The generated project files		15
Figure 25 - Copy ESD exported folder to the generated project		16
Figure 26 - STM32CubeIDE - Open Projects from File System		16
Figure 27 - Add include path to ESD generated header files		17
Figure 28 - Add platform macro		17
Figure 29 - Create source files for STM32L4 platform		17
Figure 30 -Include <code>EVE_Platform_STM32L476GDISCOVERY.h</code>		18
Figure 31 - Rename <code>main</code> function		18
Figure 32 - Disable QUAD-SPI mode for ME817EV platform		18
Figure 33 – Define <code>EVE_HOST</code> macro in <code>EVE_Config.h</code>		18
Figure 34 - Add new host platform <code>EVE_HOST_STM32L476GDISCOVERY</code>		18
Figure 35 - Add <code>M_PI</code> definition		19
Figure 36 - Enable LoadFile functions		19
Figure 37 - Configure EVE platform to use external clock		19
Figure 38 - Include <code>stddef.h</code> , <code>stdio.h</code> and <code>stdarg.h</code> in <code>EVE_Config.h</code>		19

Figure 39 - Increase stack size .....	19
Figure 40 - Add EvChargePoint_Exported to resource .....	20
Figure 41 - Exclude diskio.c.....	20
Figure 42 - Start EAB and select interface .....	21
Figure 43 - Select __Flash.bin and click button "Update" .....	21
Figure 44 - Run with STM32 MCU configuration.....	22
Figure 45 - EvChargePoint screen on LCD .....	22
Figure 47 - Default HCLK configuration .....	24
Figure 48 - Select TIM2 clock source .....	24
Figure 49 - Setup TIM2 parameters.....	24
Figure 50 - Setup TIM2 interrupt.....	25
Figure 51 - Create ESD clock and ESD label .....	25
Figure 52 - New project in ESD .....	25
Figure 53 - Declare global variable in ESD.....	26
Figure 54 - Connect global variable and clock/label .....	26
Figure 55 - Implement interrupt handling function .....	27
Figure 56 - Enable interrupt timer .....	27
Figure 57 - Initialize glass display .....	27
Figure 58 - ESD clock screen .....	28
Figure 59 - Timer interrupt from STM32L4 board .....	28

## List of Tables

Table 1 - Folder Contents.....	6
Table 2 - MCU and EVE Connections .....	9
Table 3 - APIs to be re-implemented .....	23

## Appendix C– Revision History

Document Title: BRT\_AN\_073 ESD 4.10 Exported Project Porting Guide for STM32L4 Discovery Board and FreeRTOS

Document Reference No.: BRT\_000335

Clearance No.: BRT#169

Product Page: <http://brtchip.com/utilities/#ESD4>

Document Feedback: [Send Feedback](#)

Revision	Changes	Date DD-MM-YYYY
1.0	Initial Release	23-06-2021