

Application Note

BRT\_AN\_XXX

ESD 4.10 Exported Project Porting Guide for STM32L4 Discovery Board And FreeRTOS

**Version 0.2**

**Issue Date:** 2022-07-25

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 FT81x and STM32L4XX MCU

Table of Contents

[1 Introduction 4](#_Toc109815940)

[1.1 Overview 4](#_Toc109815942)

[1.2 Scope 4](#_Toc109815943)

[2 ESD 4.15 Exported Project – Introduction 5](#_Toc109815944)

[2.1 Opening the Example Project 5](#_Toc109815946)

[2.2 Exporting the Example Project 5](#_Toc109815947)

[3 Porting principles 7](#_Toc109815948)

[3.1 Hardware 7](#_Toc109815949)

[3.2 Software 7](#_Toc109815950)

[3.2.1 HAL 7](#_Toc109815952)

[3.2.2 Project Files 7](#_Toc109815953)

[3.2.3 Linker Script 7](#_Toc109815954)

[3.2.4 Exported Code 7](#_Toc109815955)

[4 Example 8](#_Toc109815957)

[4.1 Hardware Connection 10](#_Toc109815958)

[4.1.1 Power and Ground Connection 10](#_Toc109815959)

[4.1.2 Signal lines Connection 10](#_Toc109815960)

[4.2 Software Setup 10](#_Toc109815961)

[4.2.1 Toolchain and Utilities 10](#_Toc109815962)

[4.3 Project porting procedure 11](#_Toc109815963)

[4.3.1 Export ESD Generated Source Code 12](#_Toc109815964)

[4.3.2 Generate project for STM32L4 Discovery board 13](#_Toc109815965)

[4.3.3 Port ESD generated source code to STM32L4 Discovery’s project 18](#_Toc109815966)

[4.3.4 Build and run on STM32L4 Discovery board 24](#_Toc109815967)

[4.3.5 Storage Media Configuration and Access 27](#_Toc109815968)

[4.3.6 API Re-Implementation 28](#_Toc109815970)

[5 ESD Interrupt handling example 30](#_Toc109815971)

[5.1 STM32CubeMX configuration 30](#_Toc109815972)

[5.2 Create ESD project 31](#_Toc109815973)

[5.3 Modify source code 33](#_Toc109815974)

[5.4 Build and run 34](#_Toc109815975)

[6 Contact Information 36](#_Toc109815976)

[Appendix A– References 37](#_Toc109815977)

[Document References 37](#_Toc109815978)

[Acronyms and Abbreviations 37](#_Toc109815979)

[Appendix B – List of Tables & Figures 38](#_Toc109815980)

[List of Figures 38](#_Toc109815981)

[List of Tables 39](#_Toc109815982)

[Appendix C– Revision History 40](#_Toc109815983)

# Introduction

This application note is intended as a guide for porting an **E**VE **S**creen **D**esigner (**ESD**) 4.10 project to an ARM Cortex-M4 based MCU platform running FreeRTOS. In the example, an STM32L476 Discovery board is used a target module.

ESD 4.15 contains a large number of example projects. In this note the example “EvChargePoint” is exported and modified to work on an STM32L476 Discovery board.

The procedure for porting to a new platform is described and explained. For this note, readers are expected to have the knowledge of ESD 4.15 as well as the STM32 development tools.

A screenshot of a computer

Description automatically generated with medium confidence 

EvechargePoint on ESD 4.15 EvechargePoint on EVE4

## Overview

This guide covers the following topics:

* ESD 4.15 exported project introduction.
* Principles of porting.
* Example project.

## Scope

This document covers hardware changes and software modification as well as some debugging tips while porting the exported project. It also provides some basic principles to successfully port other ESD projects. Although the example is targeted at an STM32L476 the procedure and changes required apply to a large number of embedded MCUs.

# ESD 4.15 Exported Project – Introduction

ESD 4.15 enables users to design an EVE based GUI application with minimum effort. Upon completing the design and successfully simulating it on a PC, source code for an MCU can be exported.

## Opening the Example Project

The “EvChargePoint” project is located in the ESD installation directory (typically “C:\Users\Public\Documents\EVE Screen Designer”) in the “Examples\Advanced” folder. We are making no modifications in ESD to the example source code so for this Application Note the user can open the example in ESD4.15 directly.

## Exporting the Example Project

Exporting takes a snapshot of the current ESD project and is performed by selecting **“File → Export as Eclipse Project”**, as shown Figure 1.

There are multiple targets available for exporting, selected by the “Host Platform” box. In ESD 4.15 these comprise of FT9xx modules; RP2040 based Raspberry Pi pico; the MPSSE interface on FTDI bridge chips; the FTDI FT4222H device; and a software Emulator.

For porting to embedded systems the most flexible Host Platform is the FT9xx based MM900EVxxx range. This setting exports an Eclipse project with a small amount of FT900 code which can be easily replaced or modified.

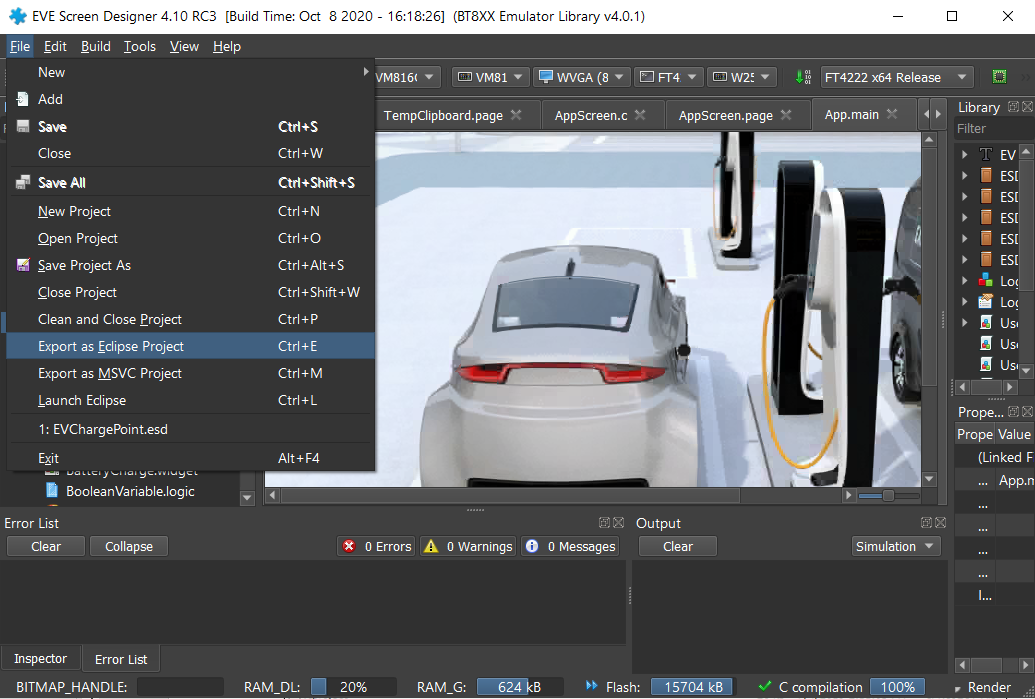


Figure 1 Export EvChargePoint Project in ESD 4.15

When exporting projects, users are prompted to select a new folder as the destination folder. The exported project is a complete set of source code and assets which can be compiled separately from the original example project within the new destination folder.

Once exporting is done, the destination folder will follow the structure shown in Figure 2.

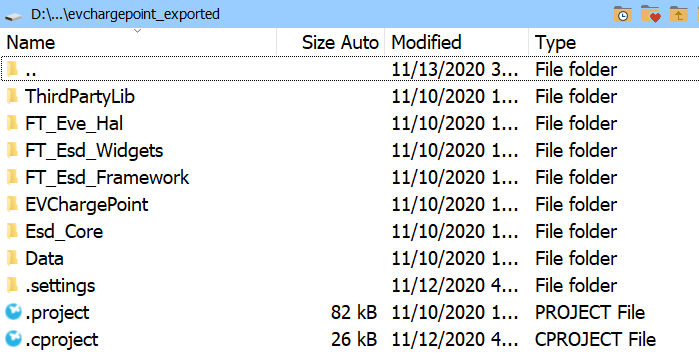


Figure 2 Folder Structure of EvChargePoint exported Project

Table 1 has an explanation of each subdirectory and file within the top level of the destination folder. The source code is divided into folders by function layers within the ESD framework.

| Folder / file Name | Description | Remarks |
| --- | --- | --- |
| ThirdPartyLib | Contains third part libraries. | Currently contains only the FatFs Library source code. |
| FT\_Eve\_Hal | Contains a hardware abstraction layer. | The major folder to be changed. |
| FT\_Esd\_Widgets | Contains source code for widgets. | Reusable common module. |
| FT\_Esd\_Framework | Contains the application framework source code. | Reusable common module. |
| EVChargePoint | Source code for the application, screen logic and user design elements. | The folder name shall be same as project name. |
| ESD\_Core | Contains the core files of ESD framework. | Reusable common module. |
| Data | Contains the converted bitmap assets. | Do not need to change. |
| .cproject | Eclipse CDT project file. | Build configurations, tool chains, individual tools etc. |
| .project | Eclipse CDT project file. | Build specification and build commands. |

Table 1 Folder Contents

By default, the exported ESD 4.15 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.

# Porting principles

## Hardware

An ESD 4.15 exported project will need access to the following hardware resources of the chosen target MCU:

* **SPI interface**: This is required to perform read and write operations to the EVE Module.
* **Clock**: Provide delay and timing control functionality.
* **Storage media**: To store bitmap assets; this can be in flash, SD/MMC card, USB disk etc.

Different MCUs have different hardware configurations, features and limitations. Therefore, users need to ensure that the hardware components above are available on the target MCU. Users are assumed to be familiar with BT81X series ICs as well as EVE 4 series modules before starting the porting work. Refer to [brtchip.com](http://brtchip.com/eve-2-development-modules-landing/) for more details.

## Software

To complete the port to the target MCU it will be necessary to modify or add to the following software modules within the exported project:

* **FT\_Eve\_Hal**: Hardware Abstraction Layer which handles all the platform-specific calls from the application.
* **Project files**: Configuration and instructions for building the project. MCU tool chain specific.
* **Linker script**: (optional) Instruct the linker software to generate a suitable MCU platform specific binary.
* **Exported code**: Rename the main() function because there is already a main() function in FreeRTOS.

### HAL

The modifications to the HAL are needed to change the target MCU platform and re-implement the transport layer API for the MCU to EVE chip interface. This folder must be modified manually by adding in a modified HAL file to support the STM32L476 device.

### Project Files

The development environment for the target MCU has to be modified to include both the required files for the target MCU and the ESD source code. Certain macros will be defined in the project files to enable features in the ESD source code. The project files can be configuration options in an Integrated Development Environment (IDE) or a traditional makefile.

### Linker Script

The linker script is an optional component of the target MCU development environment. Depending on the MCU platform, it may be necessary to include or alter this script to produce working code.

### Exported Code

There is only one change to the ESD source code is needed. The change is to accommodate the FreeRTOS operating system within the exported code. The main() function in the exported code is renamed and the new function called from a function in the FreeRTOS system. There are several smaller configuration changes required to successfully build the code.

# Example

This example illustrates how to successfully port an exported project to a new MCU. The selected target MCU platform is an [STM32L476 Discovery board](http://www.st.com/en/evaluation-tools/32l476gdiscovery.html) (Figure 3). It is connected to the development PC via a USB cable for downloading, debugging and power supply.

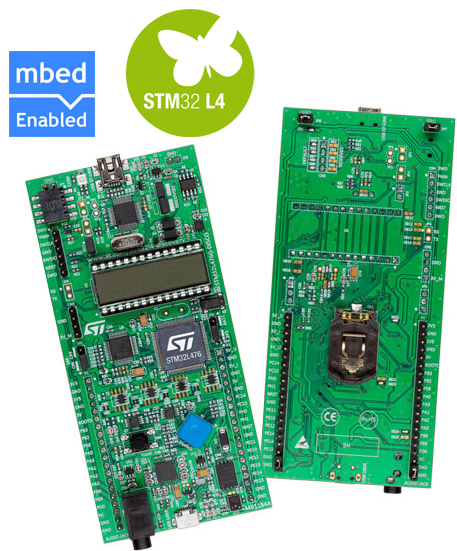


Figure 3 STM32L4 Discovery Board

In Figure 4 is the selected EVE Module. A ME817EV (with BT817) is used in this example.

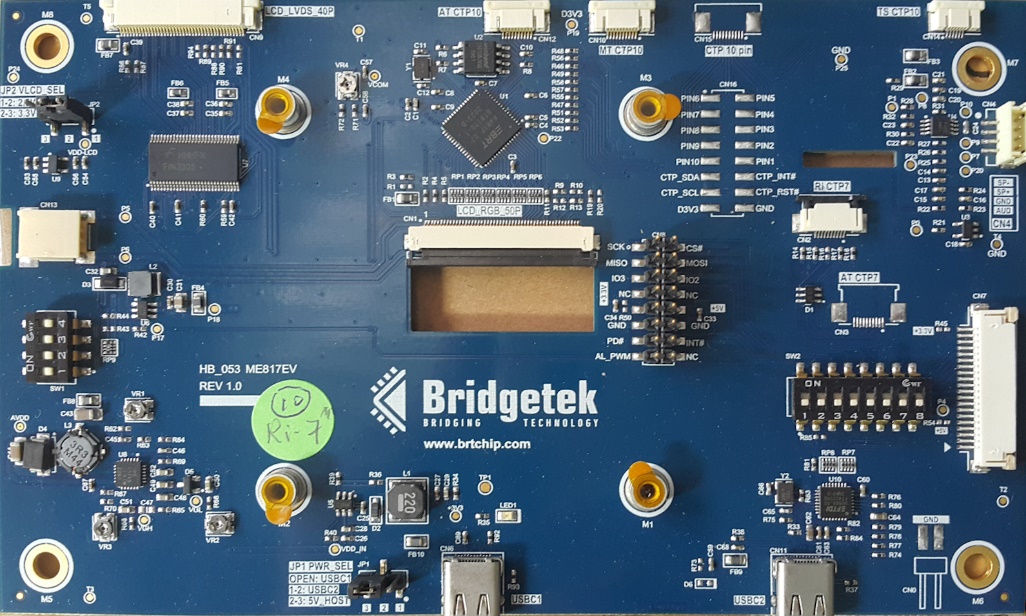


Figure 4 EVE 4 Module

Table 2 illustrates the connection between two boards.

|  |  |  |
| --- | --- | --- |
| 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 Connection

The “EvChargePoint” example project from the ESD 4.15 examples folder is used to demonstrate how the porting will be done. The EvChargePoint example is located in the “Examples\Advanced” folder of the ESD installation directory. Figure 5 is a sample screenshot after it is opened in ESD 4.15.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5 ESD 4.15 EvChargePoint Project Screenshot

## Hardware Connection

### Power and Ground Connection

The EVE Module may be powered by a 5V supply from the STM32L4 Discovery board extension connector. Both the 5V pin and one of the GND pins must be connected.

### Signal lines Connection

In this example, STM32L4 MCU’s SPI 1 interface is selected and the following connection is set up.

Care must be taken if connecting the two boards with simple jumper wires. 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.

## Software Setup

### Toolchain and Utilities

For this example, the [STM32CubeIDE](https://www.st.com/en/development-tools/stm32cubeide.html) was selected as the compiler and linker for the STM32L4 MCU. As shown in Figure 6 version 1.4.0 of STM32CubeIDE is used.

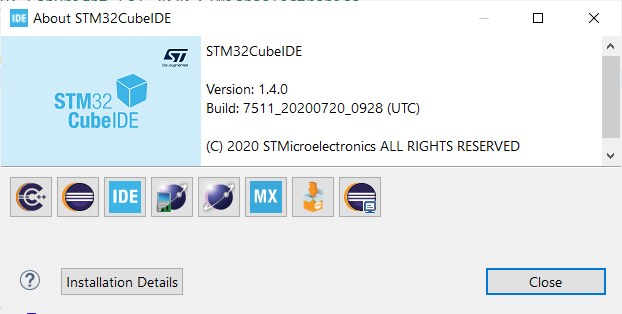


Figure 6 STM32CubeIDE version

Another very helpful tool is called [STM32CubeMX](http://www.st.com/en/development-tools/stm32cubemx.html). This tool can help users configure pin functionality. In addition, it automatically generates suitable source code to configure hardware resources.

The following embedded file is the project file of the STMCubeMX tool which is used by the example project specified in this document. It is also included in the source code for this application note.



Figure 7 screenshot shows how the pin configuration looks like once the project is opened using the STMCubeMX tool.

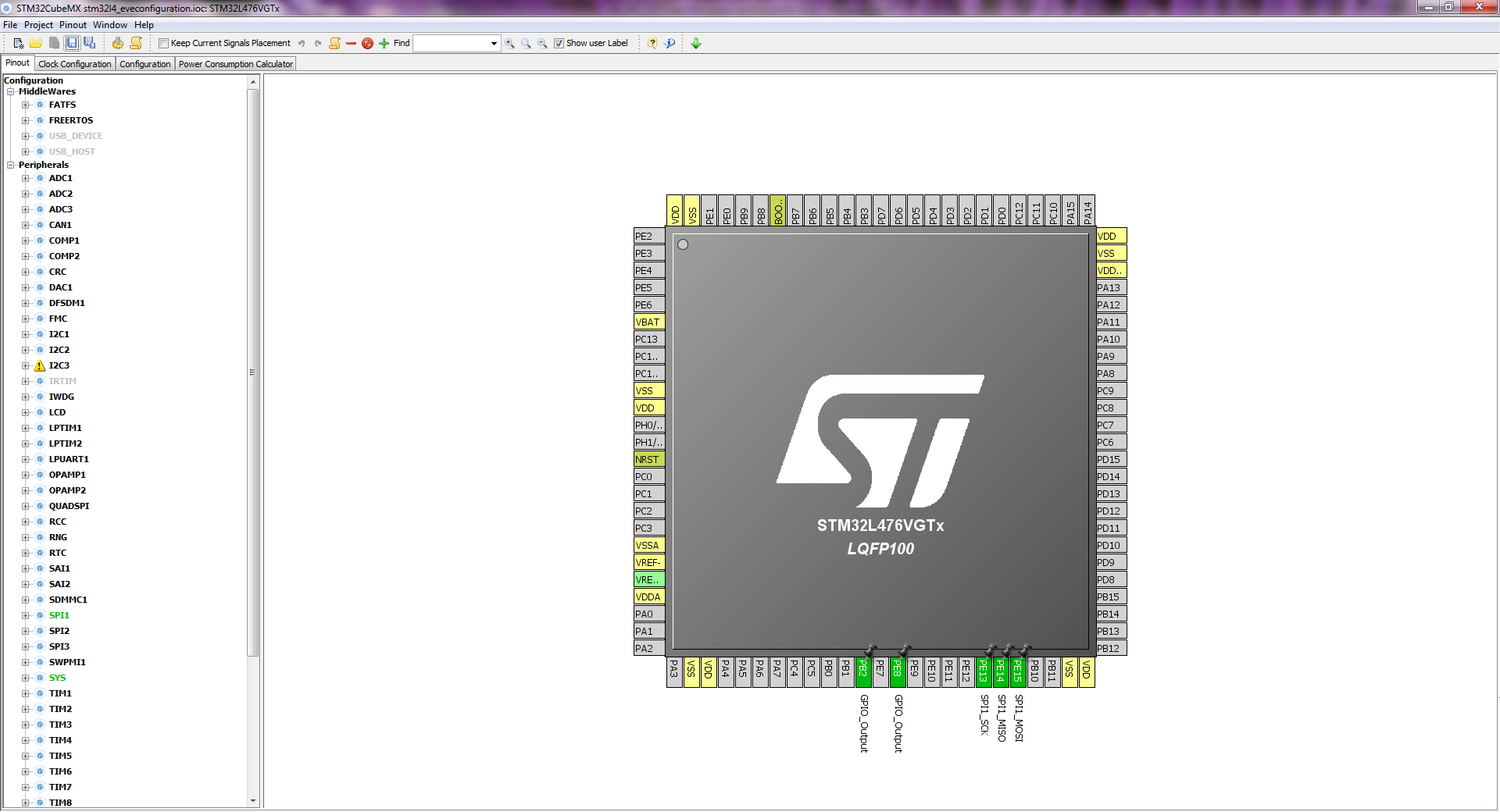


Figure 7 STMCubeMX Snapshot

## Project porting procedure

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

To summarise the procedure, the STMCubeMX utility is used to generate an empty project for STM32L4 Discovery board; then the ESD generated source code is added to the project; the combined code is configured and compiled by the STM32CubeIDE.

**STMCubeMX** generated project

**ESD** generated source code

Build on **STM32CubeIDE**

Binary for STM32L4 Discovery

Figure 8 Project porting procedure

### Export ESD Generated Source Code

In this section the ESD will generate the source code for the EVE section of the example.

#### Open the ESD Project

From ESD open the “EvChargePoint” project from the “Examples\Advanced” folder of the ESD installation directory, see Figure 9.

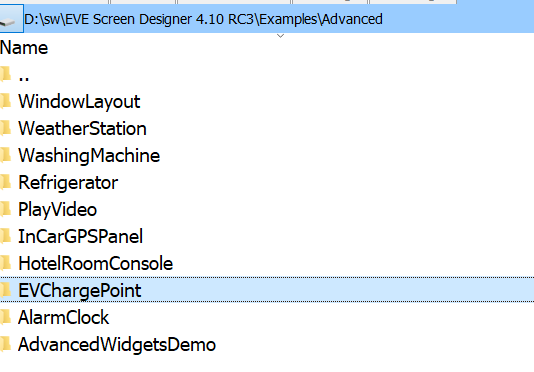


Figure 9 The EvChargePoint project on ESD

#### Export the ESD Project

Export the project to a local folder. From the menu bar, select “File” -> “Export as Eclipse Project” (Figure 10).

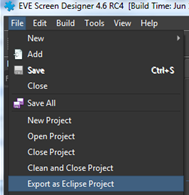


Figure 10 Export as Eclipse Project

Choose a local folder to store the export files. The exported project has the file structure shown in Figure 11.

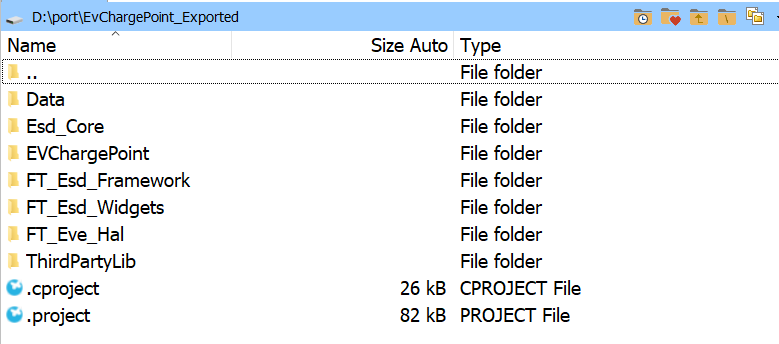


Figure 11 ESD exported project files and folders

### Generate project for STM32L4 Discovery board

In this sections the empty project for the STM32L4 board is generated.

#### Open STMCubeMX

Open the STMCubeMX application and select “ASSESS TO BOARD SELECTOR” highlighted in Figure 12.

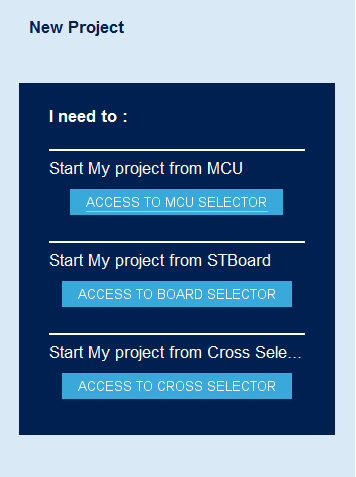


Figure 12 ASSESS TO BOARD SELECTOR

Select the “32L476GDISCOVERY” board as shown in Figure 13.

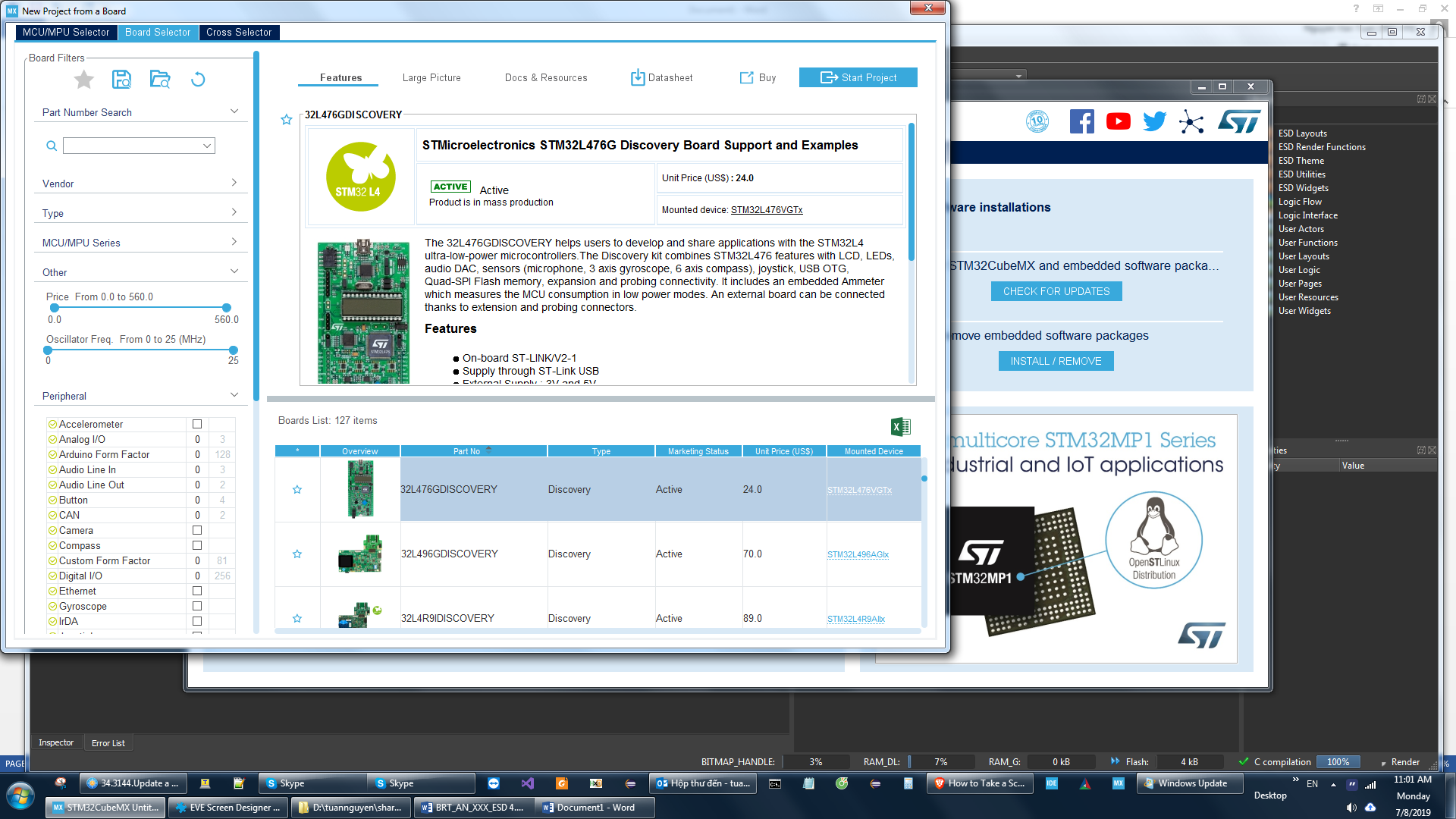


Figure 13 Select 32L476GDISCOVERY board

Select the default setting for all peripherals when the dialog box in Figure 14 is shown.

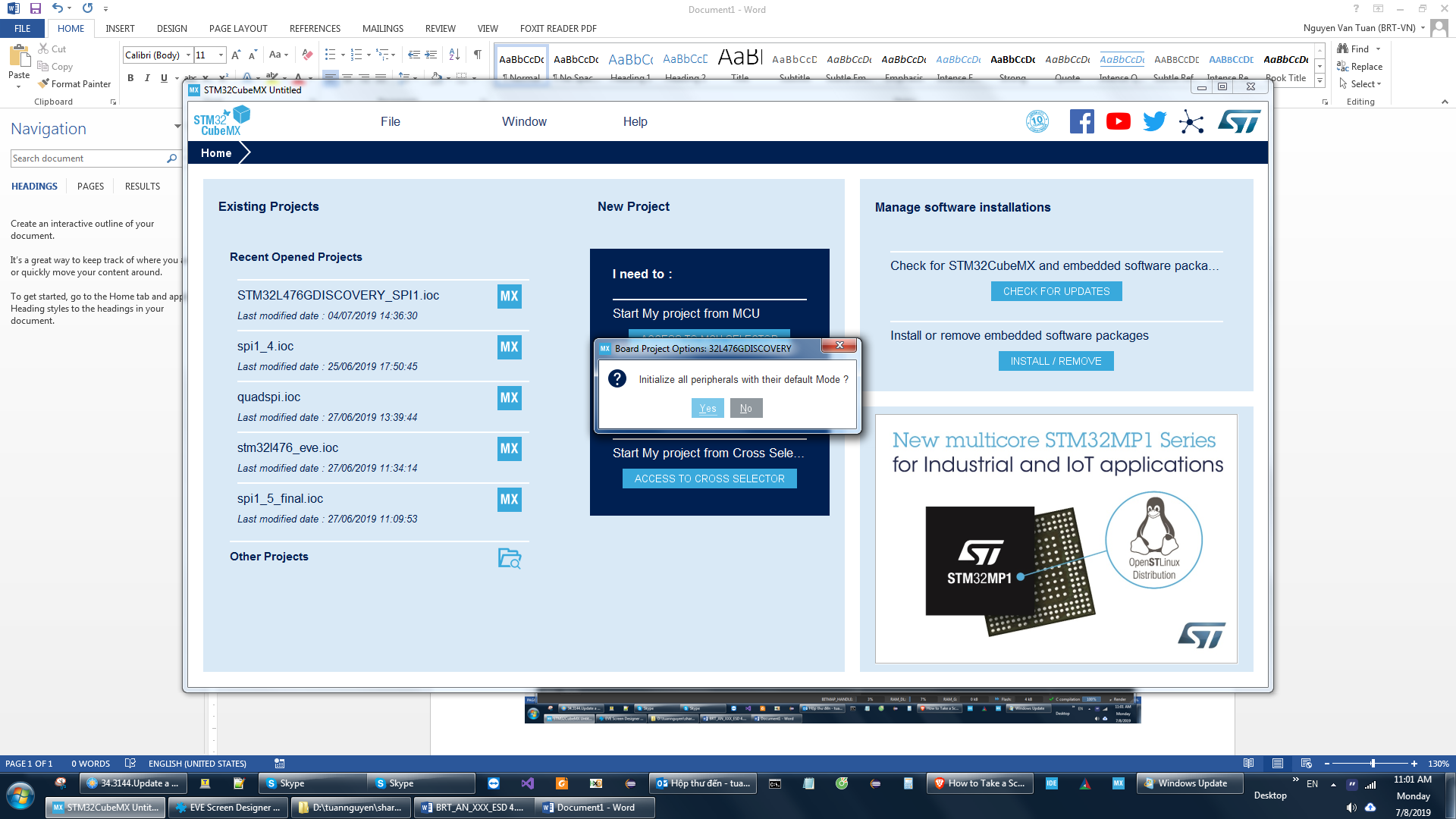


Figure 14 select default mode

#### Select Pin Configuration

Next the Pinout and configuration screen in Figure 15 will appear.

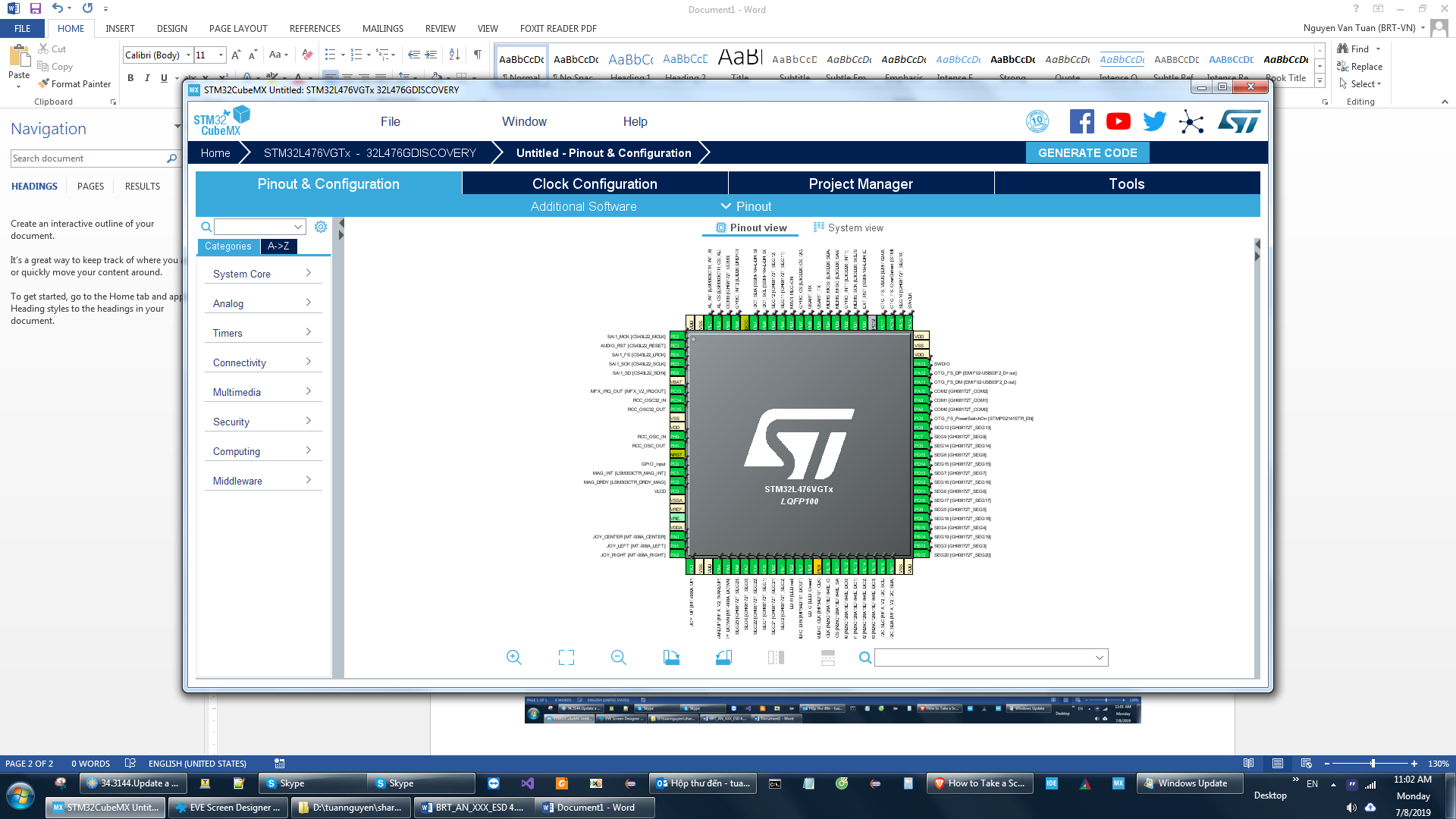


Figure 15 Pinout and configuration screen

Now, select the interface pins to the EVE module used in this example. The pins for SPI1 and GPIO on the STM32L4 board are as follows:

- PE13 -> SPI1\_SCK

- PE14 -> SPI1\_MISO

- PE15 -> SPI1\_MOSI

- PE8 -> GPIO\_Output

- PB2 -> GPIO\_Output

In Figure 16 the detail of the pin selection dialog is shown.

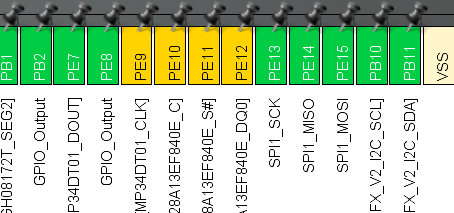


Figure 16 Select SPI ports

#### Enable SPI1

Once the pins for SPI have been selected then the SPI1 peripheral can be enabled. Set SPI1 to “Full-Duplex master” mode. The setting is shown in Figure 17.

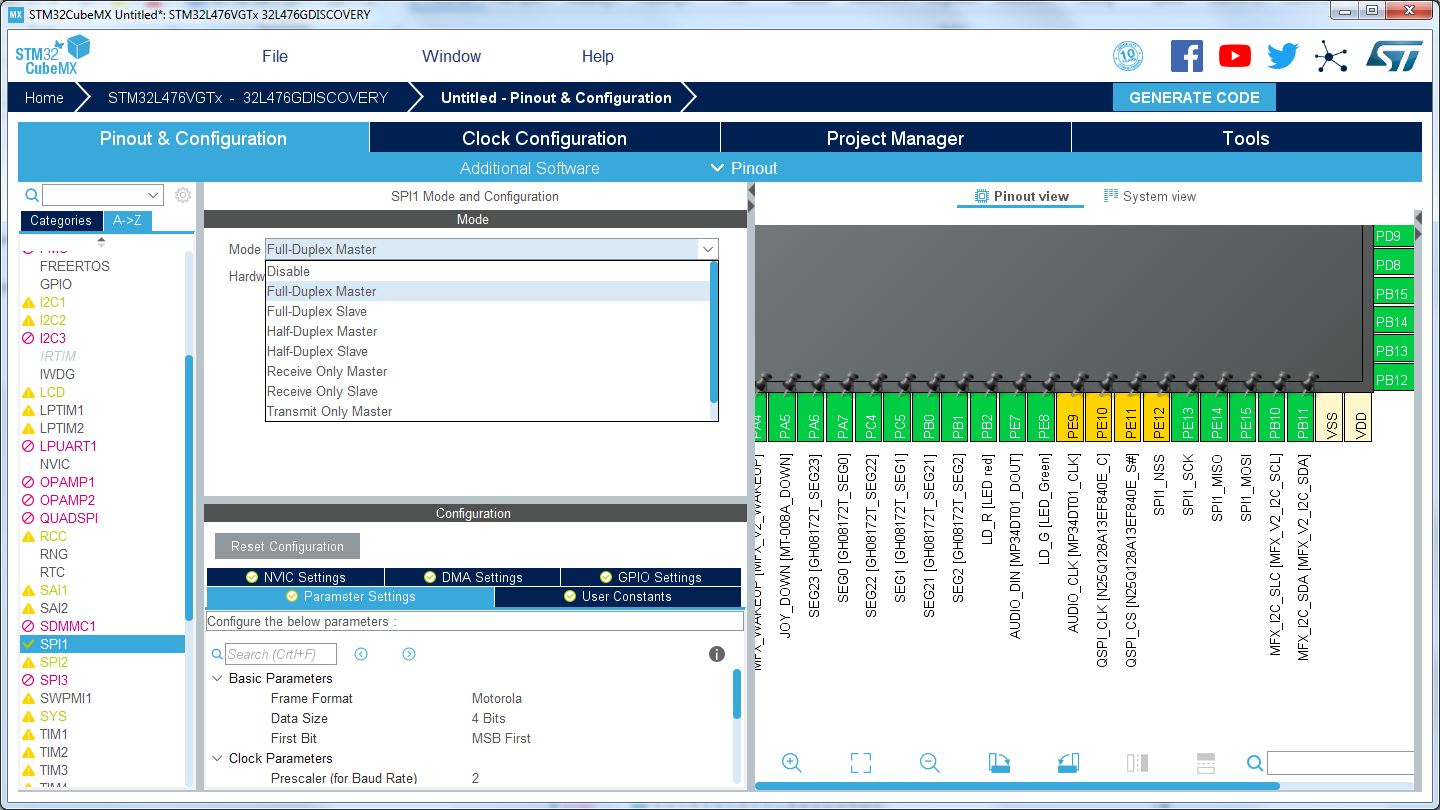


Figure 17 Set SPI1 to Full-Duplex master

The data size (i.e. number of bits transferred on the SPI bus) is set to 8 as in Figure 18.



Figure 18 SPI1 – Select data size

#### Enable Middleware

Finally, enable FreeRTOS from the “MiddleWare” group. Figure 19.

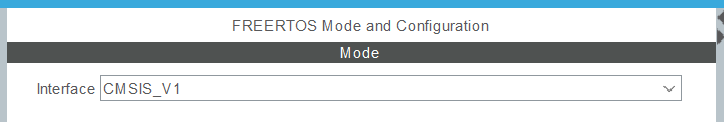


Figure 19 Enable FreeRTOS

#### Export STM32L4 Project

Once all the configuration is complete then the 32L476GDISCOVERY project can be exported. Select the “Project manager” tab to show the options in Figure 20.

* Input the project name. For example, “STM32L476\_Porting”.
* Input the project output path.
* Select “Toolchain/IDE” as STM32CubeIDE. We will use STM32CubeIDE to open the generated project.
* Select “Application structure” as “Advanced”.

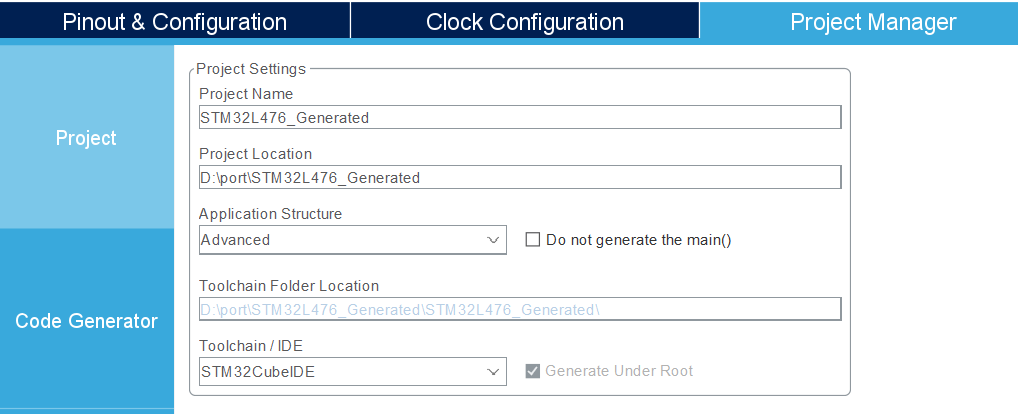


Figure 20 Export project for 32L476GDISCOVERY board

Click button “GENERATE CODE”, wait for STMCubeMX complete this action.

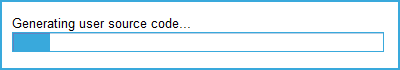


Figure 21 GENERATE CODE

Once complete the file structure shown in Figure 22 will be generated.

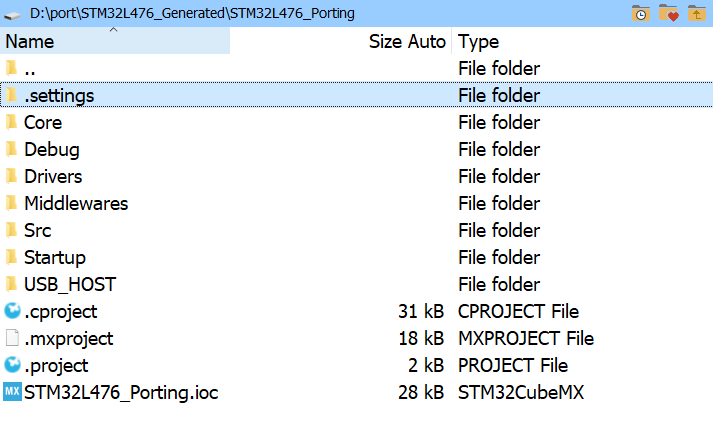


Figure 22 The generated project files for 32L476GDISCOVERY board

### Port ESD generated source code to STM32L4 Discovery’s project

#### Load generated STM32L4 Discovery’s project to STM32CubeIDE

1. Copy folder “EvChargePoint\_Exported” into “STM32L476\_Generated” folder as shown in Figure 23.

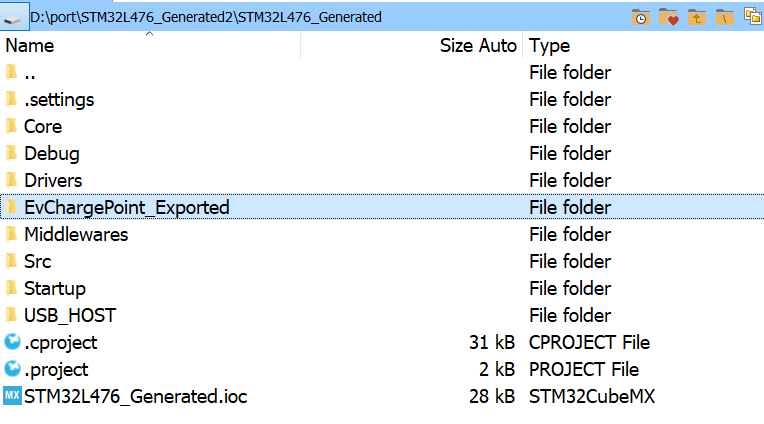


Figure 23 Copy ESD generated files to the generated STM32L4 Discovery’s project

1. Open generated STM32L4 Discovery’s project with STM32CubeIDE

Open STM32CubeIDE, select File -> “Import”, choose local folder of “STM32L476\_Generated”. The dialog box is shown in Figure 24.

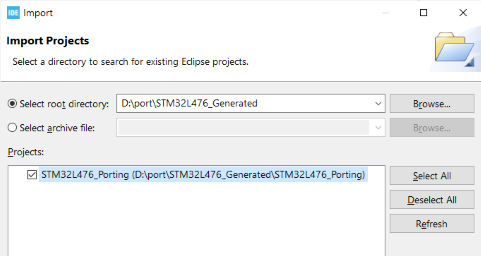


Figure 24 STM32CubeIDE - Open Projects from File System

#### Build configuration on STM32CubeIDE

Add include paths to ESD generated header files.

Right click on project name, select Properties->C/C++ General -> path and Symbols -> “includes” tab. Click “Add” button to add the paths listed:

- 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/Data  
- EvChargePoint\_Exported/EVChargePoint

The dialog box should be as in Figure 25.

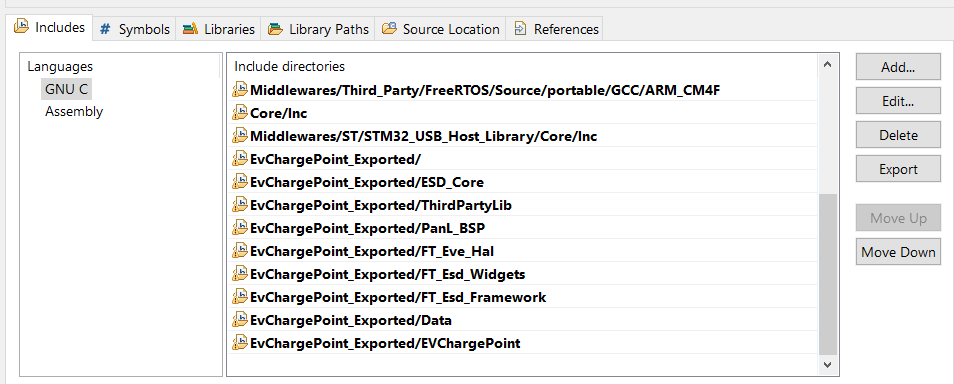


Figure 25 Add include path to ESD generated header files

Add the platform macros for the EVE platform and the STM32L4 platform. The result should look like Figure 26.

Select tab “Symbols” and add 3 macros:

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

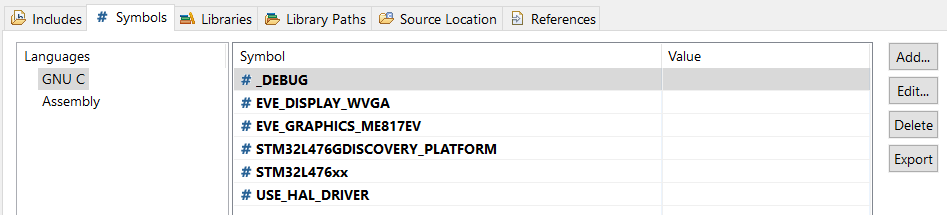


Figure 26 Add platform macro

#### Source code modification

There are 2 files which act as the Hardware Abstraction Layer for the STM32476DISCOVERY board. These are embedded in this document and included in the application note.

Copy the files “EVE\_Platform\_STM32L476GDISCOVERY.h” and “EVE\_HalImpl\_STM32L476GDISCOVERY.c” into the “FT\_Eve\_Hal” folder as shown in Figure 27.

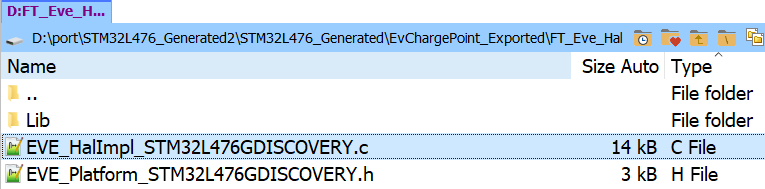


Figure 27 Create source files for STM32L4 platform

**EVE HAL Modifications**

Add a pre-processor include line for “EVE\_Platform\_STM32L476GDISCOVERY.h” in EVE\_Platform.h.

**EvChargePoint\_Exported\FT\_Eve\_Hal\EVE\_Platform.h**

53: #if defined(FT9XX\_PLATFORM)

54: #include "EVE\_Platform\_FT9XX.h"

55: #endif

56: #if defined(STM32L476GDISCOVERY\_PLATFORM)

57: #include "EVE\_Platform\_STM32L476GDISCOVERY.h"

58: #endif

59: #if defined(RP2040\_PLATFORM)

60: #include "EVE\_Platform\_RP2040.h"

61: #endif

62: #include "EVE\_GpuTypes.h"

Figure 28 Include EVE\_Platform\_STM32L476GDISCOVERY.h in EVE\_Platform.h

. .

**EvChargePoint\_Exported\FT\_Eve\_Hal\EVE\_Config.h**

28: \* have additional licence terms that apply to those amendments. However, Bridgetek

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 <stdio.h>

37: #include <stddef.h>

38: #include <stdarg.h>

39:

40: /\*

.

.

**EvChargePoint\_Exported\FT\_Eve\_Hal\EVE\_Config.h**

727: #elif defined(EVE\_GRAPHICS\_ME817EV)

728:

729: #define BT817\_ENABLE

730: //#define ENABLE\_SPI\_QUAD

731:

732: #ifndef EVE\_DISPLAY\_AVAILABLE

733: #define EVE\_DISPLAY\_AVAILABLE

734: #define DISPLAY\_RESOLUTION\_WVGA

735: #endif

736:

737: #ifndef EVE\_MULTI\_PLATFORM\_TARGET

738: #define EVE\_PLATFORM\_AVAILABLE

739: //#ifndef EVE\_PLATFORM\_RP2040

740: #if !defined(EVE\_PLATFORM\_RP2040) || !defined(MM2040EV) || !defined(IDM2040EV)

741: //#define FT4222\_PLATFORM

742: #endif

743: #endif

type . The is used to select the STM32L476DISCOVERY board.

EvChargePoint\_Exported\FT\_Eve\_Hal\EVE\_Config.h

1313: #if defined(FT900\_PLATFORM) || defined(FT93X\_PLATFORM)

1314: #define FT9XX\_PLATFORM

1315: #define EMBEDDED\_PLATFORM

1316: #define EVE\_HOST EVE\_HOST\_EMBEDDED

1317: #endif

1318:

1319: #if defined(STM32L476GDISCOVERY\_PLATFORM)

1320: #define EVE\_HOST EVE\_HOST\_STM32L476GDISCOVERY

1321: #define EMBEDDED\_PLATFORM

1322: #define EVE\_HOST EVE\_HOST\_EMBEDDED

1323: #endif

1324:

1325: #if defined(RP2040\_PLATFORM)

1326: #define EMBEDDED\_PLATFORM

1327: #define EVE\_HOST EVE\_HOST\_EMBEDDED

1328: #endif

1329:

1330: #define EVE\_CONFIG\_\_STR(x) #x

1331: #define EVE\_CONFIG\_STR(x) EVE\_CONFIG\_\_STR(x)

.

EvChargePoint\_Exported\FT\_Eve\_Hal\EVE\_HAL\_Defs.h

126: typedef enum EVE\_HOST\_T

126: {

126: EVE\_HOST\_UNKNOWN = 0,

126: EVE\_HOST\_BT8XXEMU,

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;

**Generated Project Code Modification**

Rename main() function of ESD to “ESD\_Start()”.

There is a main() function in the ESD generated project which will not be used. The main() function from FreeRTOS will be used. This will call the renamed ESD\_Start() function from the ESD code.

EvChargePoint\_Exported\EVChargePoint\App\_Generated.c

133:

134: int ESD\_Start()

135: {

136: Esd\_Initialize();

137:

138: {void StartDefaultTask(void const \* argument)

Figure 29 Rename main function

**FreeRTOS Modifications**

We now need to modify the FreeRTOS default task to call the ESD\_Start() function.

A prototype for the ESD\_Start() function needs to be added to the FreeRTOS code. The main() function from FreeRTOS is found in the “Core\Src\main.c” file. The call to the EST\_Start() function is added to the “StartDefaultTask()” function. The code changes for the FreeRTOS main.c file are shown in Figure 30.

Core\Src\main.c

757: /\* USER CODE BEGIN 4 \*/

758: Extern void ESD\_Start();

759:

760: /\* USER CODE END 4 \*/

761:

762: /\* USER CODE BEGIN Header\_StartDefaultTask \*/

763: /\*\*

764: \* @brief Function implementing the defaultTask thread.

765: \* @param argument: Not used

766: \* @retval None

767: \*/

768: /\* USER CODE END Header\_StartDefaultTask \*/

769: void StartDefaultTask(void const \* argument)

770: {

771: /\* init code for USB\_HOST \*/

772: MX\_USB\_HOST\_Init();

773:

774: /\* USER CODE BEGIN 5 \*/

775: ESD\_Start();

776:

777: /\* Infinite loop \*/

778: for(;;)

779: {

780: osDelay(1);

781: }

782: /\* USER CODE END 5 \*/

783: }

Figure 30 Add call to ESD\_Start to FreeRTOS default task

.

Core\Src\main.c

152: /\* USER CODE END RTOS\_QUEUES \*/

153:

154: /\* Create the thread(s) \*/

155: /\* definition and creation of defaultTask \*/

156: osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 512);

157: defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);

158:

159: /\* USER CODE BEGIN RTOS\_THREADS \*/

**ESD Core Modifications**

Add PI number definition. STM32 platform does not have this definition by default, this causes compile error in the ESD code.

EvChargePoint\_Exported\Esd\_Core\Esd\_Math.h

34: #include “Esd\_Base.h”

35: #define M\_PI 3.14159265358979323846

36:

Figure 33 Add M\_PI definition

**EVE Storage Modification**

Enable the LoadFile functions.

EvChargePoint\_Exported\FT\_Eve\_Hal\EVE\_LoadFile\_STDIO.c

32: #include “EVE\_LoadFile.h”

33: #include “EVE\_Platform.h”

34: #if !defined(FT9XX\_PLATFORM) && !defined(STM32L476GDISCOVERY\_PLATFORM)

35:

36: #include <stdio.h>

Figure 34 Enable LoadFile functions

**EVE Utilities Modification**

Enable the external clock.

EvChargePoint\_Exported\FT\_Eve\_Hal\EVE\_Util.c

251: #if !defined(ME810A\_HV35R) && !defined(ME812A\_WH50R) && !defined(ME813A\_WH50C)

252: /\* Board without external oscillator will not work when ExternalOsc is enabled \*/

253: bootup->ExternalOsc = true;

254: #endif

255: #if defined(STM32L476GDISCOVERY\_PLATFORM)

256: bootup->ExternalOsc = true;

257: #endif

258:

259: #ifdef EVE\_SYSTEM\_CLOCK

Figure 35 Configure EVE platform to use external clock

#### Project Modification

#### 

#### 

Include EvChargePoint\_Exported to the compilation, by default it is excluded. See Figure 38.

* Right click on EvChargePoint\_Exported, select “Resource configuration” -> “Exclude from build”.
* Deselect all checkbox.

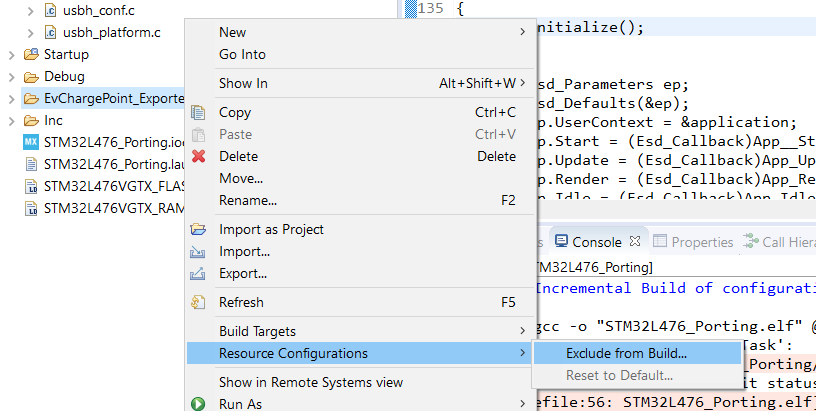
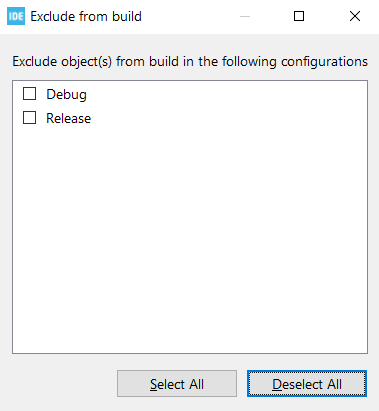


Figure 38 Add EvChargePoint\_Exported to resource

Also exclude the file diskio.c from the compilation using the same method. We do not build this file on either STM32 or FT9XX platform. The Project Explorer section should have the diskio.c file highlighted as in Figure 39.

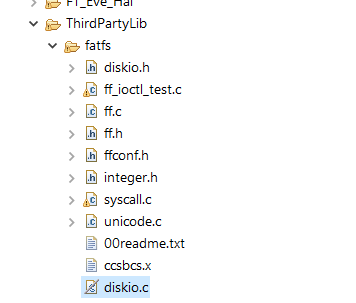


Figure 39 Exclude diskio.c

### Build and run on STM32L4 Discovery board

Before running, user must flash file “EvChargePoint\_Exported\Data\\_\_Flash.bin” into EVE4 by EVE Asset Builder (EAB) tool.

The EVE Asset Builder (EAB) is a GUI based tool, it is used to flash binary file into EVE4.

User can get free software EAB at <https://brtchip.com/eve-toolchains>

Here are steps to flash the file “\\_\_Flash.bin” into EVE4.

#### Connect PC and EVE4

Use an MPSSE interface or an FT4222 device to directly connect your PC and EVE4.

#### Flash EVE4

Open EAB and select the “FLASH UTILITIES” icon at the top of the Window. Then select the “Flash Programmer” tab and the “Program” tab within the top tab, as shown in Figure 40.

Next select the interface to EVE4 (this will be FT4222 or MPSSE) and select the EVE chip as BT817.

Graphical user interface, text, application, email

Description automatically generated

Figure 40 Start EAB and select interface

Select the file path to “\_\_Flash.bin” file and click button “UPDATE” or “UPDATE&VERIFY”.

#### Connect STM32 and EVE4

When the update is completed, connect the STM32 board to the EVE4 (see Table 2).

#### Run the Ported Application

Reopen the STM32CubeIDE application with the ported application project.

* Build project. The keyboard shortcut for Build is Ctrl+B.
* Setup a Run Configuration to run the ported application on the STM32L4 Discovery board.
* Select Run->”Run Configuration”.
* Add “New Configuration”. The “C/C++ Application” and “Project” boxes should be filled out correctly if the project is selected in the “Project Explorer" as per Figure 42.
* Click on the “Run” button.

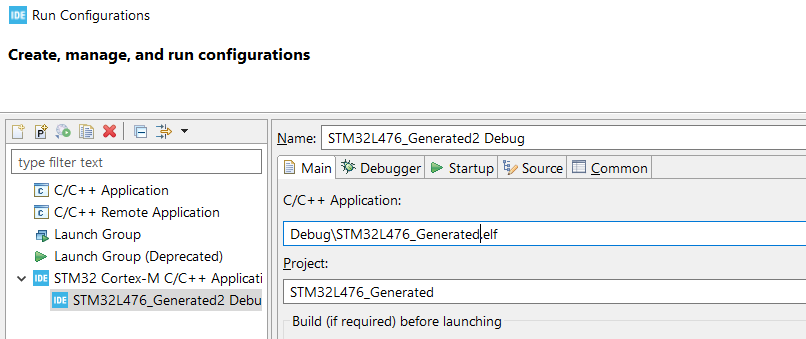


Figure 42 Run with STM32 MCU configuration

The application should run on the STM32L4 Discovery board. Figure 43 shows a photograph of the ported application running on the board.



Figure 43 EvChargePoint screen on LCD

### Storage Media Configuration and Access

In this example, the storage media is not enabled to simplify the procedure. Implementing storage media would negate the need to program assets into the EVE4 flash in sections 4.3.4.1 and 4.3.4.2. The assets could be programmed onto an SD card prior to running the ported application.

To access bitmap assets, users need to implement the functions in Figure 43 in the EVE Framework file “FT\_Eve\_Hal\EVE\_LoadFile\_FATFS.c” for the STM32 platform.

FT\_Eve\_Hal\EVE\_LoadFile\_FATFS.c

bool EVE\_Util\_loadImageFile(EVE\_HalContext \*phost, uint32\_t address, const char \*filename, uint32\_t \*format)

bool EVE\_Util\_loadInflateFile(EVE\_HalContext \*phost, uint32\_t address, const char \*filename)

bool EVE\_Util\_loadRawFile(EVE\_HalContext \*phost, uint32\_t address, const char \*filename)

bool EVE\_Util\_loadSdCard(EVE\_HalContext \*phost)

Figure 43 Storage Media Functions

| API name | Remarks |
| --- | --- |
| Ft\_Hal\_LoadImageFile | It loads the image file from storage media with specified “filename” to EVE RAM\_G “address” and sends the data through coprocessor command “CMD\_LOADIMAGE”. |
| Ft\_Hal\_LoadInflateFile | It loads the deflated file from storage media with specified “filename” to EVE RAM\_G “address” and sends the data through coprocessor command “CMD\_INFLATE”. |
| Ft\_Hal\_LoadRawFile | It loads the raw image file from storage media with specified “filename” to EVE RAM\_G “address”. |
| Ft\_Hal\_LoadSDCard | Prepare SD card if it is used in FATFS. Otherwise, just make it as empty. |

Table 3 Storage Function

In our example, these four functions were empty.

The STM32L476G Discovery does not have internal flash memory or SD card reader port. However, it can communicate with an external SD card reader to read or write files. An externally connected SD card reader is shown connected in Figure 44.

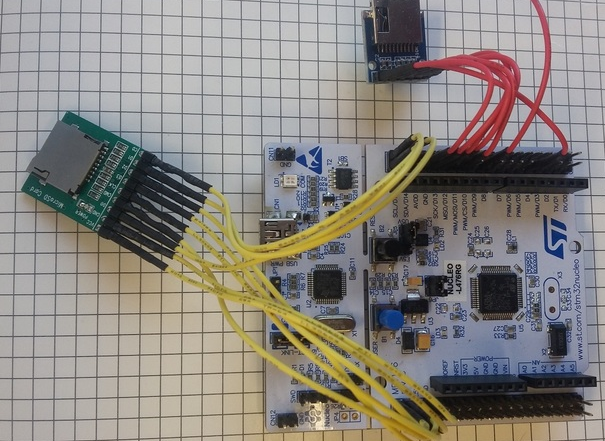


Figure 44 External SD card reader

### API Re-Implementation

Provided by the example code is a platform specific implementation file for the EVE for the STM32L4 platform. This is in the file “EVE\_HalImpl\_STM32L476GDISCOVERY.c”.

The EVE API calls have been rewritten to use STM32L4 methods for timing and communication with the EVE.

|  |  |  |
| --- | --- | --- |
| **API name** | **Remarks** | **Note** |
| EVE\_millis | Get the current system tick |  |
| EVE\_sleep | Delay the specified amount of time |  |
| EVE\_Mcu\_release | Release the MCU and its peripheral (SPI, GPIO) |  |
| EVE\_Mcu\_initialize | Initialize the MCU and its peripheral (SPI, GPIO) |  |
| EVE\_Millis\_initialize | Init MCUs timer |  |
| EVE\_Millis\_release | Release MCUs timer |  |
| EVE\_HalImpl\_initialize | Initialize HAL platform |  |
| EVE\_HalImpl\_release | Release HAL platform |  |
| EVE\_HalImpl\_defaults | Set the default configuration parameters |  |
| EVE\_HalImpl\_close | Close a HAL context |  |
| EVE\_HalImpl\_idle | Idle callback function. Call regularly to update frequently changing internal state |  |
| EVE\_Hal\_startTransfer | 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. |
| EVE\_Hal\_endTransfer | De-assert the CS to end the SPI transferring |  |
| EVE\_Hal\_flush | Flush data to Coprocessor |  |
| EVE\_Hal\_transfer8 | Send or receive one byte |  |
| EVE\_Hal\_transfer16 | Send or receive 2 bytes |  |
| EVE\_Hal\_transfer32 | Send or receive 4 bytes |  |
| EVE\_Hal\_hostCommand | Send host commands to EVE |  |
| EVE\_Hal\_hostCommandExt3 | Send 3 bytes host commands to EVE |  |
| EVE\_Hal\_powerCycle | Toggle PD pin to wake up EVE | A delay after each toggle is mandatory |
| EVE\_Util\_closeFile | Close opened file |  |
| EVE\_Util\_loadSdCard | Mount the SD card |  |
| EVE\_Util\_sdCardReady | Check if SD card ready or not |  |
| EVE\_Util\_loadImageFile | Load the image file from storage into EVE RAM\_G | Empty for users to define implementation |
| EVE\_Util\_loadInflateFile | Load the image file from storage into EVE RAM\_G | Empty for users to define implementation |
| EVE\_Util\_loadRawFile | Load the image file from storage into EVE RAM\_G | Empty for users to define implementation |
| EVE\_Util\_loadSdCard | Load the image file from storage into EVE RAM\_G | Empty for users to define implementation |

Table 3 APIs to be changed

The changes in Table 3 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.

# ESD Interrupt handling example

To handle the interrupt from MPU/MCU, users shall introduce the interrupt handler and update a global variable to capture the changes. When 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.

## STM32CubeMX configuration

Configuration for STM32 is the same to [4.3.2](#_Port_ESD_generated), with an addition configuration for timer. We use TIM2 in this example.

The timeout is calculated by this formular:

**Timeout = Prescaler \* Counter period / HCLK.**

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



Figure 45 Default HCLK configuration

Below step to enable TIM2:

* Select clock source as Internal clock:

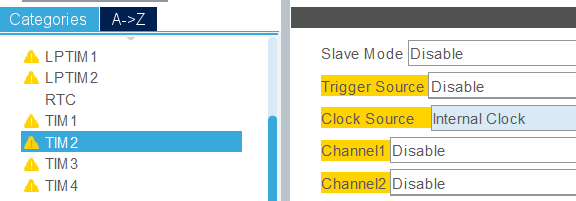


Figure 46 Select TIM2 clock source

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

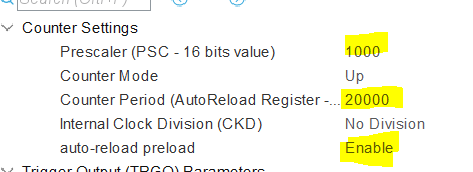


Figure 47 Setup TIM2 parameters

* Enable TIM2 global interrupt:

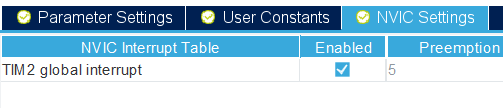


Figure 48 Setup TIM2 interrupt

* Save and export project

Reference:



## Create ESD project

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



Figure 49 Create ESD clock and ESD label

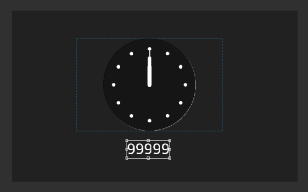


Figure 50 New project in ESD

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

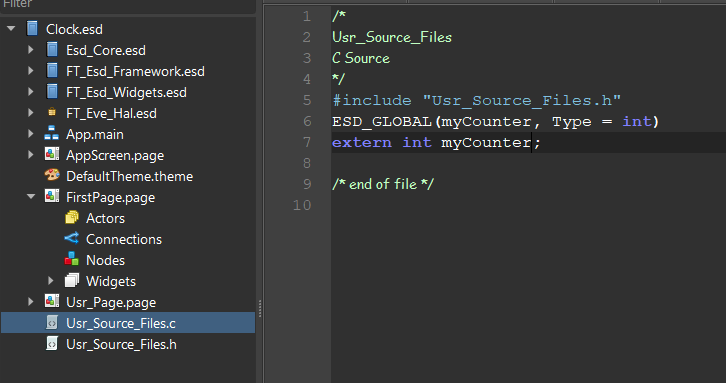


Figure 51 Declare global variable in ESD

Variable “myCounter” will appear in “User Globals” tree, now connect global variable “myCounter” with clock and label:

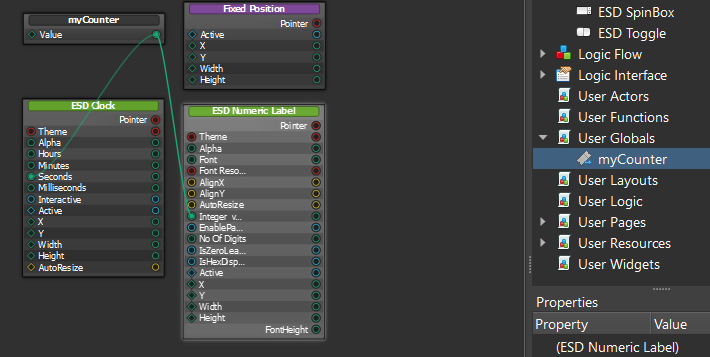


Figure 52 Connect global variable and clock/label

Save and export project.

Reference:



## Modify source code

We copy Eclipse exported project into STM32 generated project and do porting similar to [4.3.3](#_Port_ESD_generated).

The old main() function from the ESD exported source code was renamed

Now modify main.c to define global variable “myCounter” and implement function to capture interrupt:

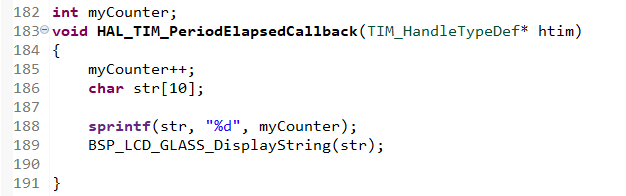


Figure 53 Implement interrupt handling function

In main function, enable the TIM2 interrupt, by calling HAL\_TIM\_Base\_Start\_IT():

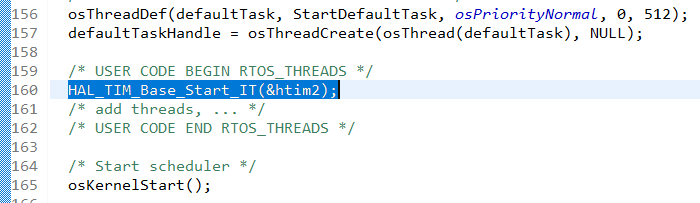


Figure 54 Enable interrupt timer

This step is optional, we display the interrupt counter on a glass display of STM32L476 discovery board, using library in STM32L476G-Discovery BSP, so need to initialize glass display in main function:

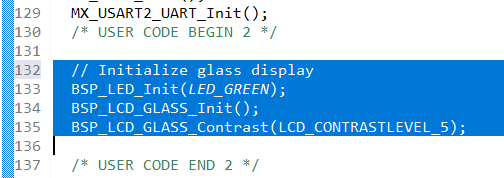


Figure 55 Initialize glass display

Please refer main.c here:



## Build and run

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



Figure 56 ESD clock screen

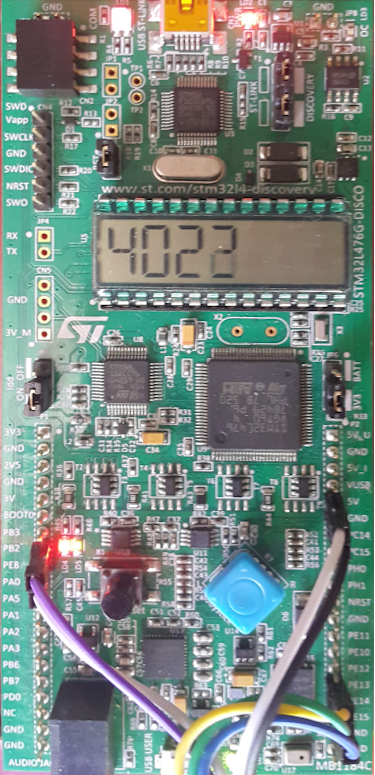


Figure 57 Glass screen

# 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 | | 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 (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://brtchip.com/>

**Distributor and Sales Representatives**

Please visit the Sales Network page of the [Bridgetek Web site](http://brtchip.com/contact-us/) 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](http://www.st.com/resource/en/reference_manual/dm00083560.pdf)

[STM32L476xx datasheet](http://www.st.com/resource/en/datasheet/stm32l476je.pdf)

[User Manual of STM32L4 Discovery board](http://www.st.com/resource/en/user_manual/dm00172179.pdf)

[FT81x Programmer Guide](http://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT81X_Series_Programmer_Guide.pdf)

[FT81x Datasheet](http://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_FT81x.pdf)

## Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Terms** | **Description** |
| EVE | Embedded Video Engine |
| EVE Module | FT81X series based display module |
| FT900 | FT900 Microcontroller from FTDI |
| SPI | Serial Peripheral Interface |
| USB | Universal Serial Bus |
| ESD 4.15 | EVE Screen Designer 4.15 |

# Appendix B – List of Tables & Figures

## List of Figures

[Figure 1 Export EvChargePoint Project in ESD 4.10 5](#_Toc58319200)

[Figure 2 Folder Structure of EvChargePoint exported Project 5](#_Toc58319201)

[Figure 3 STM32L4 Discovery Board 8](#_Toc58319202)

[Figure 4 EVE 4 Module 8](#_Toc58319203)

[Figure 5 ESD 4.10 EvChargePoint Project Screenshot 9](#_Toc58319204)

[Figure 6 STM32CubeIDE version 10](#_Toc58319205)

[Figure 7 STMCubeMX Snapshot 11](#_Toc58319206)

[Figure 8 Project porting procedure 11](#_Toc58319207)

[Figure 9 The EvChargePoint project on ESD 12](#_Toc58319208)

[Figure 10 Export as Eclipse Project 12](#_Toc58319209)

[Figure 11 ESD exported project files and folders 12](#_Toc58319210)

[Figure 12 ASSESS TO BOARD SELECTOR 13](#_Toc58319211)

[Figure 13 Select 32L476GDISCOVERY board 13](#_Toc58319212)

[Figure 14 select default mode 13](#_Toc58319213)

[Figure 15 Pinout and configuration screen 14](#_Toc58319214)

[Figure 16 Select SPI ports 14](#_Toc58319215)

[Figure 17 Set SPI1 to Full-Duplex master 15](#_Toc58319216)

[Figure 18 SPI1 – Select data size 15](#_Toc58319217)

[Figure 19 Enable FreeRTOS 15](#_Toc58319218)

[Figure 20 Export project for 32L476GDISCOVERY board 16](#_Toc58319219)

[Figure 21 GENERATE CODE 16](#_Toc58319220)

[Figure 22 The generated project files for 32L476GDISCOVERY board 16](#_Toc58319221)

[Figure 23 Copy ESD generated files to the generated STM32L4 Discovery’s project 17](#_Toc58319222)

[Figure 24 STM32CubeIDE - Open Projects from File System 17](#_Toc58319223)

[Figure 25 Add include path to ESD generated header files 18](#_Toc58319224)

[Figure 26 Add platform macro 18](#_Toc58319225)

[Figure 27 Create source files for STM32L4 platform 19](#_Toc58319226)

[Figure 28 Include EVE\_Platform\_STM32L476GDISCOVERY.h” in EVE\_Platform.h 19](#_Toc58319227)

[Figure 29 Rename main function 19](#_Toc58319228)

[Figure 30 Disable QUAD-SPI mode for ME817EV platform 20](#_Toc58319229)

[Figure 31 Add new EVE\_HOST 20](#_Toc58319230)

[Figure 32 Add new host platform EVE\_HOST\_STM32L476GDISCOVERY 20](#_Toc58319231)

[Figure 33 Add M\_PI definition 21](#_Toc58319232)

[Figure 34 Enable LoadFile functions 21](#_Toc58319233)

[Figure 35 Configure EVE platform to use external clock 21](#_Toc58319234)

[Figure 36 Include stddef.h, stdio.h and stdarg.h in EVE\_Config.h 21](#_Toc58319235)

[Figure 37 Increase stack size 22](#_Toc58319236)

[Figure 38 Add EvChargePoint\_Exported to resource 22](#_Toc58319237)

[Figure 39 Exclude diskio.c 22](#_Toc58319238)

[Figure 40 Start EAB and select interface 23](#_Toc58319239)

[Figure 41 Select \_\_Flash.bin and click button “Update” 23](#_Toc58319240)

[Figure 42 Run with STM32 MCU configuration 24](#_Toc58319241)

[Figure 43 EvChargePoint screen on LCD 24](#_Toc58319242)

[Figure 44 External SD card reader 25](#_Toc58319243)

[Figure 45 Default HCLK configuration 27](#_Toc58319244)

[Figure 46 Select TIM2 clock source 27](#_Toc58319245)

[Figure 47 Setup TIM2 parameters 27](#_Toc58319246)

[Figure 48 Setup TIM2 interrupt 28](#_Toc58319247)

[Figure 49 Create ESD clock and ESD label 28](#_Toc58319248)

[Figure 50 New project in ESD 28](#_Toc58319249)

[Figure 51 Declare global variable in ESD 29](#_Toc58319250)

[Figure 52 Connect global variable and clock/label 29](#_Toc58319251)

[Figure 53 Implement interrupt handling function 30](#_Toc58319252)

[Figure 54 Enable interrupt timer 30](#_Toc58319253)

[Figure 55 Initialize glass display 30](#_Toc58319254)

[Figure 56 ESD clock screen 31](#_Toc58319255)

[Figure 57 Glass screen 31](#_Toc58319256)

## List of Tables

[Table 1 Folder Contents 6](#_Toc58319257)

[Table 2 MCU and EVE Connection 9](#_Toc58319258)

[Table 3 APIs to be changed 26](#_Toc58319259)

# Appendix C– Revision History

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

Document Reference No.: BRT\_000206

Clearance No.: NA

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

Document Feedback: [Send Feedback](mailto:docufeedback@brtchip.com?subject=Document%20Feedback:%20BRT_AN_001%20Version%201.0)

|  |  |  |
| --- | --- | --- |
| Revision | Changes | Date |
| 0.1 | First Draft | 2020-17-11 |

Revision History

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

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date  YYYY-MM-DD | Changes | Editor |
|  |  |  |  |