Using The Software Code Library

By Hakimi Wanyusof

OVERVIEW

This application note provides an overview of our C / C++ software code library, which is available for all of our optical digital sensors except for VCNL4010, VCNL4020, and VCNL3020. There are dedicated software code and application note for these optical sensors, which can be found here. This application note explains the contents, the structure, and how to use the software code library. The software code library can be found under the "Design Tools" tab for each of the product page of the digital sensors in the website. The proximity sensor VCNL4035X01 will be used as an example for this application note. The software code library for this proximity sensor can be found here under the "Design Tools" tab.

THE CONTENTS OF THE SOFTWARE CODE LIBRARY

The software code file contains **"C"**, **"C++"**, and **"Project Examples"** folders. The **"C"** and **"C++"** folders contain the raw software code, which can be exported into the IDE software application of your selected microcontroller.

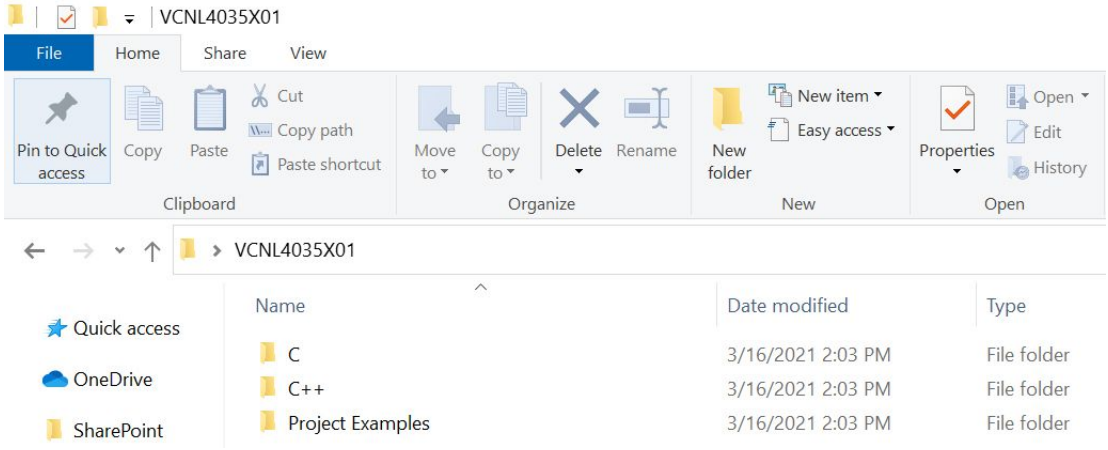

Fig. 1 - The Contents of the Software Code

APPLICATION NOTE

Using The Software Code Library

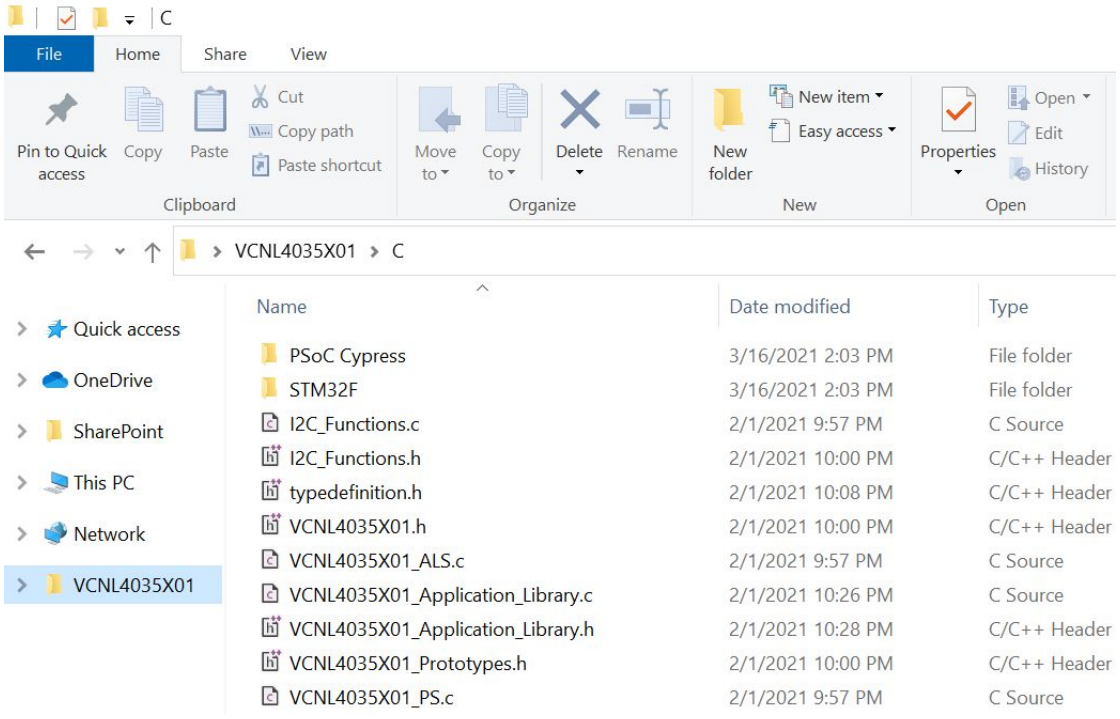

Fig. 2 - The Contents of the **"C"** Folder

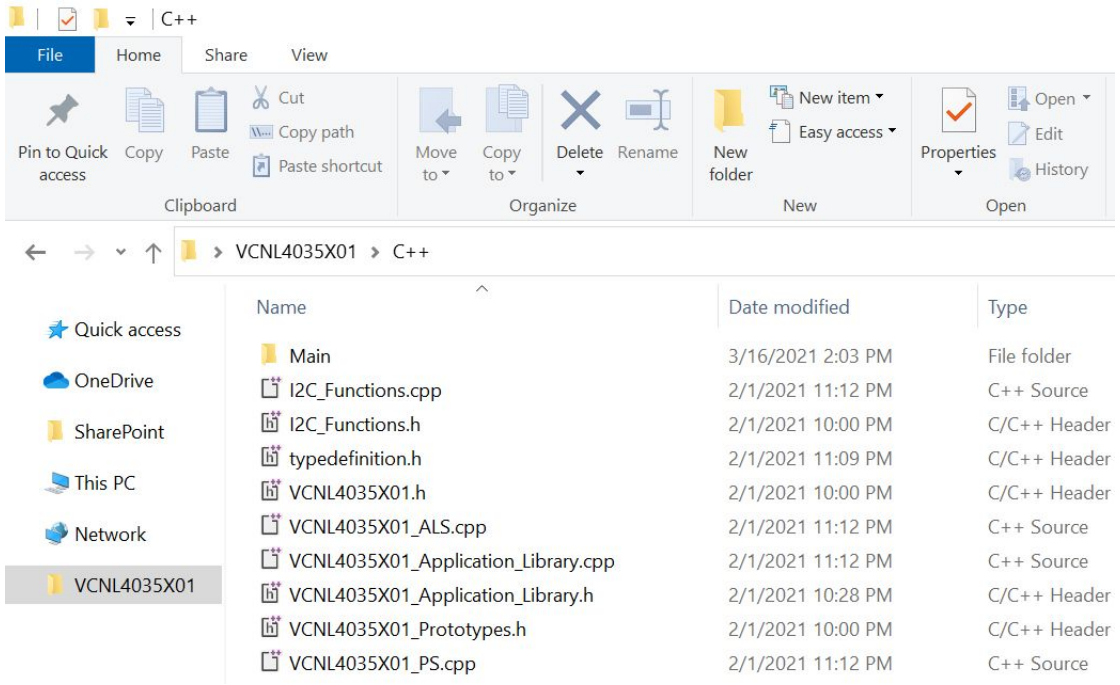

Fig. 3 - The Contents of the **"C++"** Folder

APPLICATION NOTE

Using The Software Code Library

On the other hand, the **"Project Examples"** folder contains the project files, which can be directly run by clicking the project / workspace files. The project files are the fastest way to start testing with our sensors, where the software code could be directly executed, given that the programmer has set up the same pinouts as the one, which has been set up in the project files. Currently, only STM32F, PSoC® Cypress, and Arduino microcontroller families will be supported for this project examples.

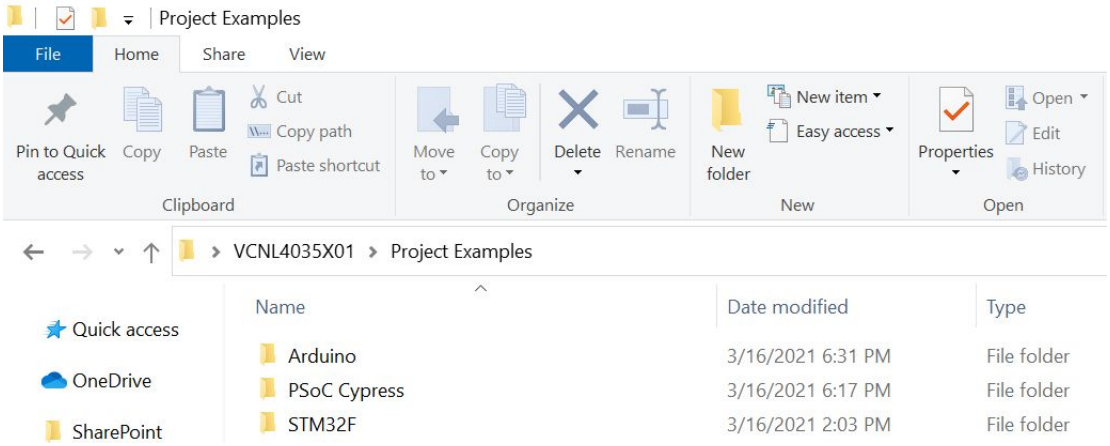

Fig. 4 - The contents of the **"Project Examples"** Folder

THE STRUCTURE OF THE SOFTWARE CODE LIBRARY

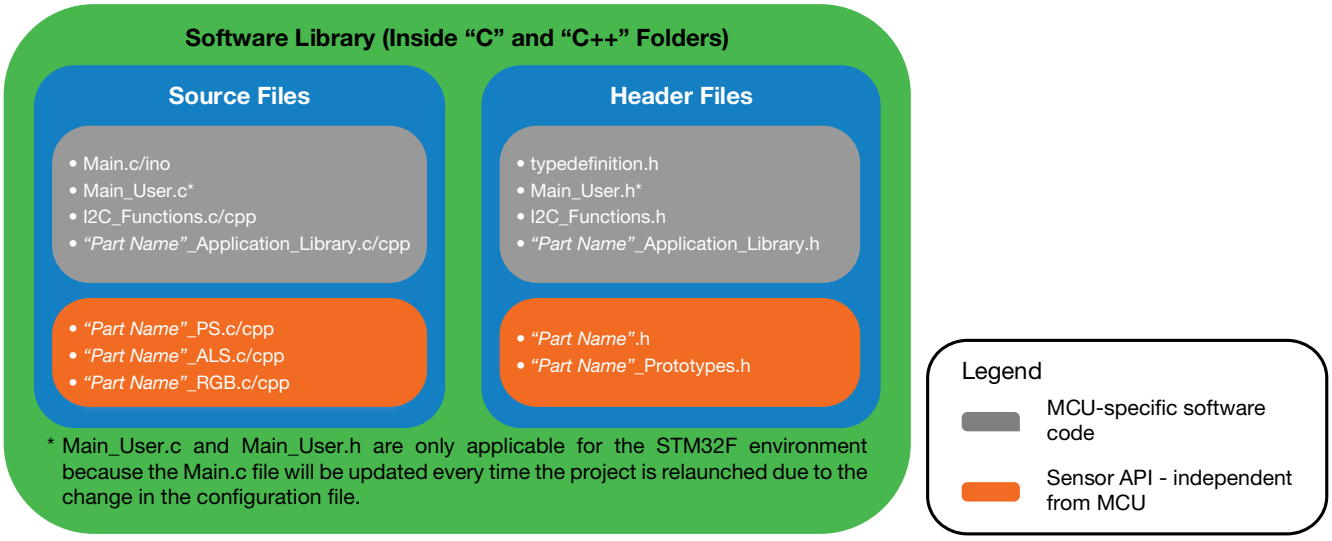

Fig. 5 - The Structure of the Software Code Library

The provided software code, which consists of MCU-specific software code and sensor API as shown in Fig. 5, is designed to be modular. It means that the programmer can use the sensor API regardless of microcontroller platforms. Only MCU specific software code has to be added / ported or just use the provided code for the supported platforms.

Using The Software Code Library

This modular approach will reduce the time during the application development regardless of the platform used because only the MCU-specific software code has to be added / ported for the whole code to work. For example, the programmer can use/port the examples code in **Main.c/ino** (For PSoC® Cypress or Arduino platform), **Main_User.c** (For STM32F platform), and **"Part Name"_Application_Library.c/cpp** to quickly initialize different sensor modes, printing sensor register values via COM Port for debugging purpose as well as some useful application functions like calculating Lux for the ALS sensor and calculating CCT value using the empirical approach for the color sensor.

The following platforms are supported by the MCU-specific software code:

- PSoC® Cypress
- Arduino and Arduino-supported MCU
- STM32F

Notes

- Vishay does not own the MCU-specific code. The MCU-specific codes, which have been modified, are owned either from Cypress (now Infineon AG) and STMicroelectronics under the condition that the codes have to be used with their respective microcontrollers. The MCU-specific code from Arduino can be used, distributed and / or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation
- For Arduino and Arduino-supported MCU - The codes have been tested with Arduino Uno and Teensy3.5. Other Arduino and Arduino-supported MCU platforms could still work or some small code adjustment might be needed especially with regards to **Wire.h/i2c_t3.h** library compatibility

Any other unsupported platforms would require the addition of API library code in **I2C_Functions.h** and **I2C_Functions.c/cpp**.

The comparison of the provided software code files between MCUs and product lines are shown in Fig. 6 and Fig. 7 respectively:

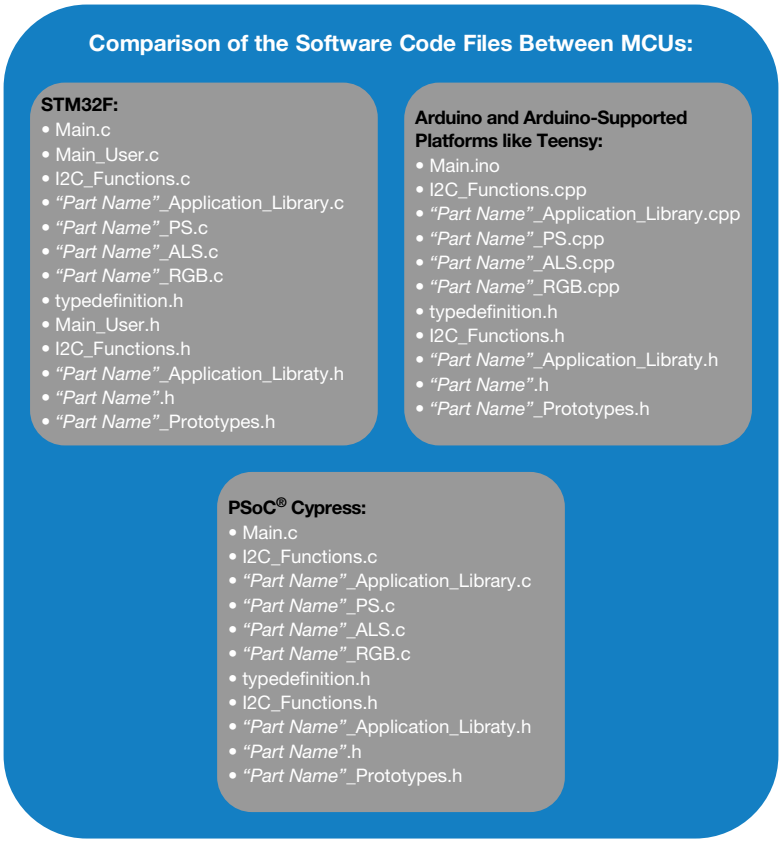

Fig. 6 - The Comparison of the Provided Software Code Files Between MCUs

Using The Software Code Library

Fig. 7 - The Comparison of the Provided Software Code Files Between MCUs

APPLICATION NOTE

Using The Software Code Library

HOW TO USE THE SOFTWARE CODE STEP BY STEP

1. STM32F

Step 1: Create a new project in STM32 Cube IDE

Step 2: Configure the pinouts for **I2C**, **RCC**, **USB_Device**, **USB_OTG_FS**

Step 3: Configure the corresponding USB CDC clock configuration for debugging via COM port purpose

Step 4: Open the **"C"** folder

Step 5: Add the following files into the STM32 Cube IDE or other supported IDE:

- **Main.c** (From the **"STM32F" folder**)

  Note

  - **Main.c** will be updated to a default code by the STM32 Cube IDE. The three lines of codes (**INIT_"Part Name"()**, **INIT_"Part Name"()**, and **PRINT_"Part Name"()**), which call the required header file as well as the initialization and printing functions in **Main_User.c**, have to be written again when **Main.c** is updated to the default code after relaunch. The provided **Main.c** file can be used as an example of where to add this code if a code rewrite happens
- **Main_User.c** (From the **"STM32F"** folder)
- **I2C_Functions.c**
- **"Part Name"_Application_Library.c**
- **"Part Name"_PS.c**
- **"Part Name"_ALS.c**
- **"Part Name"_RGB.c**
- **typedefinition.h**
- **Main_User.h** (From the **"STM32F"** folder)
- **I2C_Functions.h**
- **"Part Name"_Application_Library.h**
- **"Part Name".h**
- **"Part Name"_Prototypes.h**

Step 6: Activate the MCU-specific code in the files **I2C_Functions.c**, **I2C_Functions.h**, and **"Part Name"_Application_Library.c** by defining **#define STM32F** in **typedefinition.h**

Step 7: Set / change the sensor parameters/mode in **Main_User.c**

Step 8: **Run Main.c**

APPLICATION NOTE

Using The Software Code Library

2. PSoC® Cypress

Step 1: Create a new project in PSoC® Creator

Step 2: Configure the I2C Master component in **TopDesign.cysch**

Step 3: Configure the corresponding I2C pins

Step 4: Configure the CDC component as well as the USB CDC clock for debugging via COM port purpose

Step 5: Open the **"C"** folder

Step 6: Add the following files into the PSoC® Creator IDE or other supported IDE:

- **Main.c** (From the **"PSoC Cypress"** folder)
- **I2C_Functions.c**
- **"Part Name"_Application _Library.c**
- **"Part Name"_PS.c**
- **"Part Name"_ALS.c**
- **"Part Name"_RGB.c**
- **typedefinition.h**
- **I2C_Functions.h**
- **"Part Name"_Application_Library.h**
- **"Part Name".h**
- **"Part Name"_Prototypes.h**

Step 7: Activate the MCU-specific code in the files **I2C_Functions.c**, **I2C_Functions.h**, and **"Part Name"_Application_Library.c** by defining **#define Cypress** in **typedefinition.h**

Step 8: Set / change the sensor parameters / mode in **Main.c**

Step 9: Run **Main.c**

Using The Software Code Library

3. Arduino and Arduino-Supported Platforms Like Teensy

Step 1: Open the **"C++"** folder

Step 2: Add the following files into the Arduino **"libraries"** folder (usually the **"Part Name"** folder has to be created):

- **I2C_Functions.cpp**
- **"Part Name"_Application_Library.cpp**
- **"Part Name"_PS.cpp**
- **"Part Name"_ALS.cpp**
- **"Part Name"_RGB.cpp**
- **typedefinition.h**
- **I2C_Functions.h**
- **"Part Name"_Application_Library.h**
- **"Part Name".h**
- **"Part Name"_Prototypes.h**

For example: C:\Arduino\libraries**"Part Name"**

Step 3: Add the **Main.ino** from the **"Main"** folder into your desired **"project"** folder, which should be also named as **"Main"** and it should be separated from the above **"libraries"** folder

Step 4: Activate the MCU-specific code in the files **I2C_Functions.cpp**, **I2_Functions.h**, and **"Part Name"_Application_Library.cpp** by defining **#define Arduino** in **typedefinition.h**

Step 5: Activate the code for Arduino platforms by defining **#define wirelib** in **typedefinition.h** and comment the line **#define i2ct3**. On the other hand, activate the code for Teensy platforms by defining **#define i2ct3** in **typedefinition.h** and comment the line **#define wirelib**

Note

- Arduino platforms use Wire.h library while Teensy platforms use i2c_t3.h library

Step 6: Open the Arduino sketch

Step 7: Click File > Open

Step 8: Select the directory to your already created **"project"** folder, which contains the **Main.ino**

Step 9: Set / change the sensor parameters / mode in **Main.ino**

Step 10: Run **Main.ino**

Using The Software Code Library

4. Other MCUs

Step 1: Create a new project in your selected IDE

Step 2: Configure the required pinouts, the I^2C components as well as the USB CDC components

Step 3: Open the **"C"** folder / **"C++"** folder

Step 4: Add the following files into the IDE:

- **I2C_Functions.c/cpp**
- **"Part Name"_Application _Library.c/cpp**
- **"Part Name"_PS.c/cpp**
- **"Part Name"_ALS.c/cpp**
- **"Part Name"_RGB.c/cpp**
- **typedefinition.h**
- **I2C_Functions.h**
- **"Part Name"_Application_Library.h**
- **"Part Name".h**
- **"Part Name"_Prototypes.h**

Step 5: Add the I2C API code of your MCU into **I2C_Functions.c/cpp** or **I2C_Functions.h** within the **#ifdef #endif** identifier statement of your MCU. **Please ensure that the restart condition is implemented correctly for the I2C read command**

Step 6: Activate the MCU-specific code in the files **I2C_Functions.c/cpp**, **I2C_Functions.h**, and **"Part Name"_Application_Library.c/cpp** by defining **#define (write your MCU name)** in **typedefinition.h**

Step 7: Port the software code from **Main.c/ino** (from the **"PSoC Cypress"** / **"Main"** folder) or **Main_User.c** (from the **"STM32F"** folder) and adapt it in the **Main.c/cpp** based on your selected MCU environment

Step 8: Port the software code from **"Part Name"_Application_Library.c/cpp** as well as **"Part Name"_Application_Library.h** and adapt it based on your selected MCU environment. Here, only Print functions are MCU-specific and the rest are not. Therefore, you can directly use the non-Print functions for your application if needed

Step 9: Set / change the sensor parameters / mode in **Main.c/cpp**

Step 10: Run **Main.c/cpp**

Using The Software Code Library

HOW TO USE THE PROJECT EXAMPLES FILES

1. STM32F

Step 1: Create the following directory: C:\Software\STM Project

Note

- STM project is directory-specific. For the project file to be directly executed, the above directory has to be established, where the **"Software"** and **"STM Project"** folders have to be created in **"C:"**

Step 2: Open the **"Project Examples"** folder

Step 3: Open the **"STM32F"** folder

Step 4: Copy the **"Part Name"** folder and paste it into the directory mentioned in Step 1

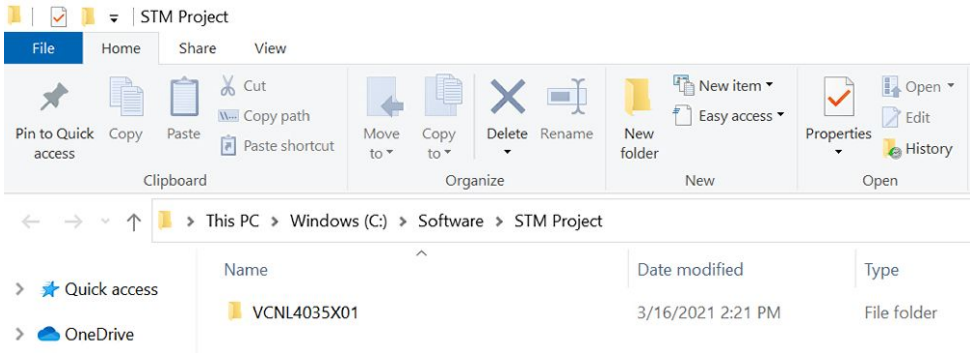

Fig. 8 - Example Project Directory for the STM32F Platform

Step 5: Go into the **"Part Name"** folder and then again into the **"Part Name"** folder

Step 6: Click the **.cproject** file

APPLICATION NOTE

Using The Software Code Library

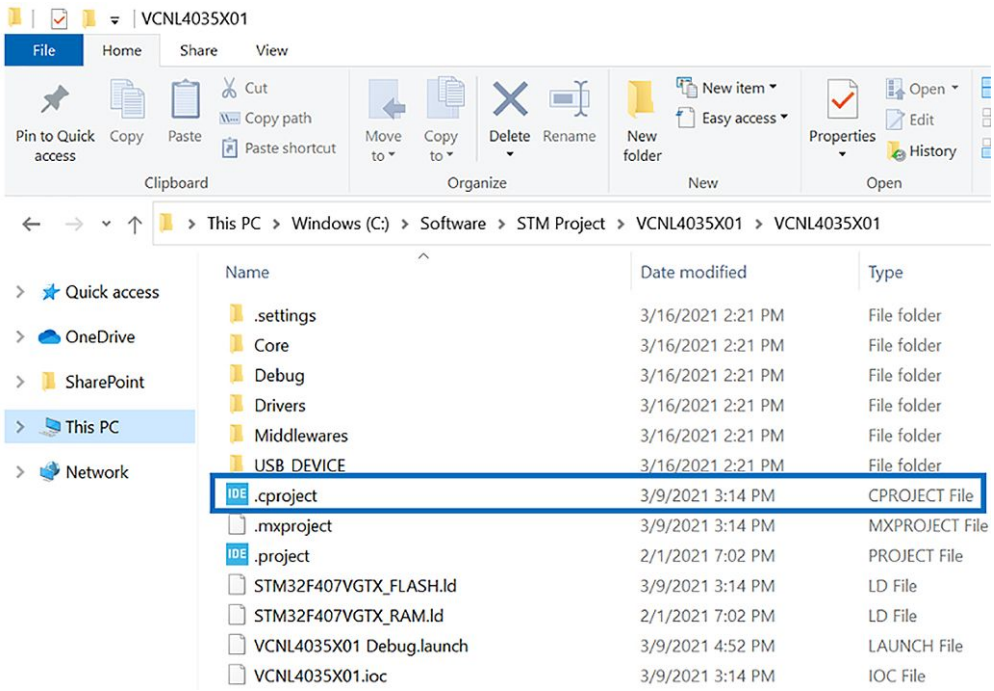

Fig. 9 - The Project File for the STM32F Platform

Step 7: Select the directory to be the same as in Step 1 and click **Launch**

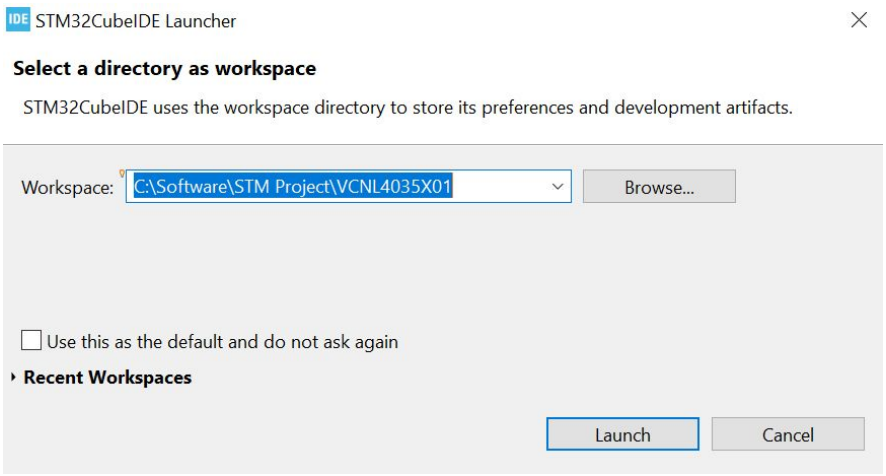

Fig. 10 - Select the Directory

Step 8: Ensure that the following 3 lines exist in **Main.c**:

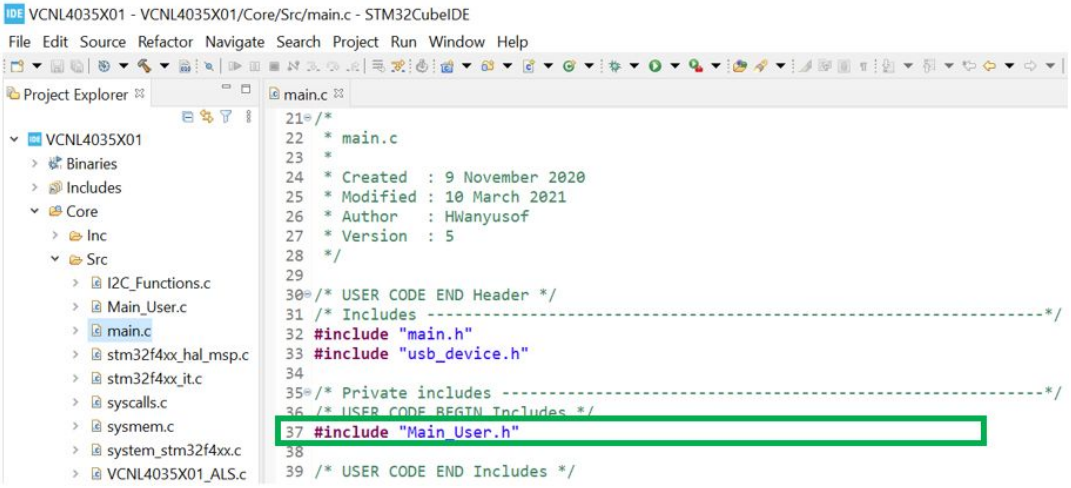

Fig. 11 - Header File **"Main_User.h"** Needs to be Included in **Main.c**

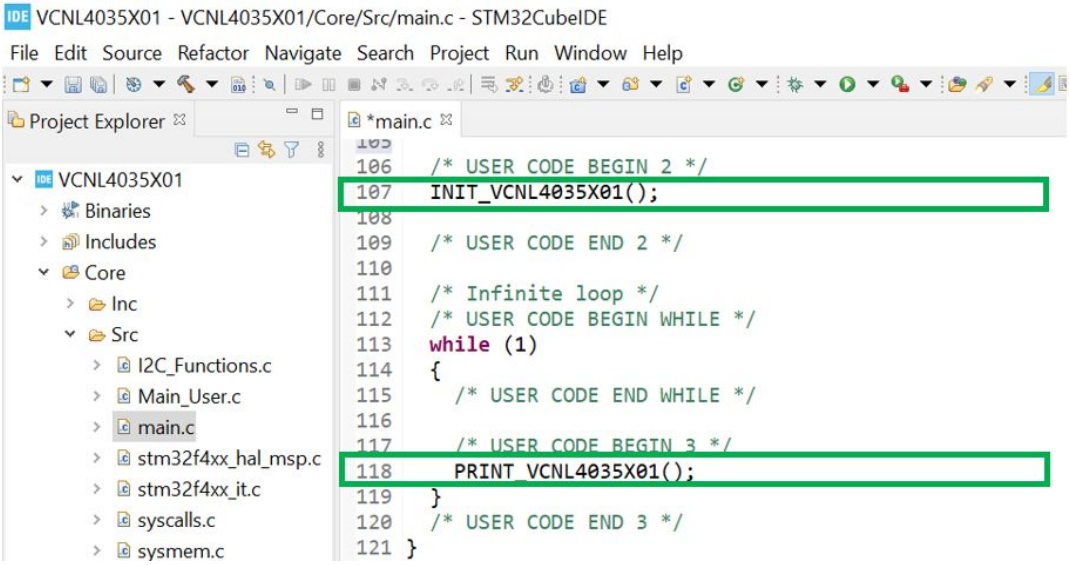

Fig. 12 - The initialization function **INIT_"Part Name"()** and the printing function **PRINT_ "Part Name"()** need to be included in **Main.c**

Note

- **Main.c** will be updated to a default code by the STM32 Cube IDE. The three lines of codes (**INIT_"Part Name"()**, **INIT_"Part Name"()**, and **PRINT_"Part Name"()**), which call the required header file as well as the initialization and printing functions in **Main_User.c**, have to be written again when **Main.c** is updated to the default code after relaunch. The provided **Main.c** file can be used as an example of where to add this code if a code rewrite happens

Step 9: Run to execute the project

Using The Software Code Library

2. PSoC® Cypress

Step 1: Open the **"Project Examples"** folder

Step 2: Open the **"PSoC Cypress"** folder and then **"Part Name"** folder

Step 3: Click the **"Part Name".cywrk** to execute the project

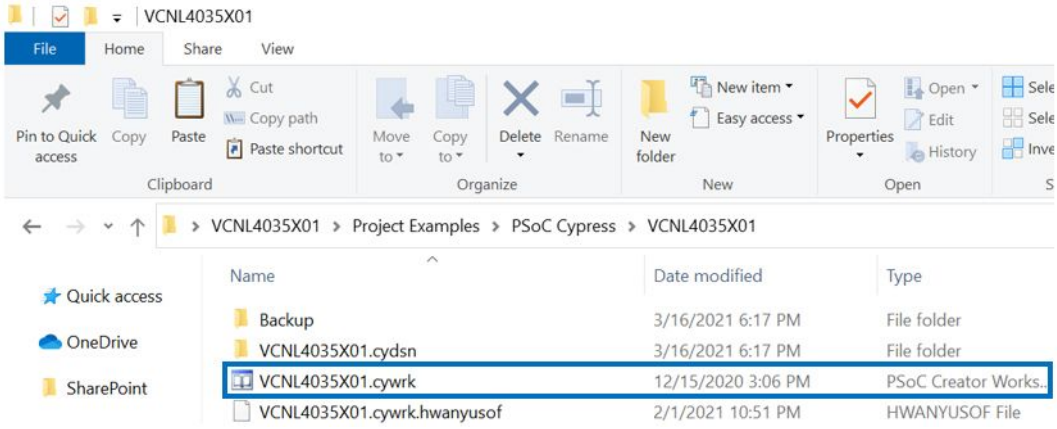

Fig. 13 - Run the PSoC® Cypress Workspace

Step 4: Run to execute the project

3. Arduino and Arduino-Supported Platforms Like Teensy

Step 1: Open the **"Project Examples"** folder

Step 2: Open the **"Arduino"** folder

Step 3: Add the following **"Part Name"** folder into the Arduino **"libraries"** folder

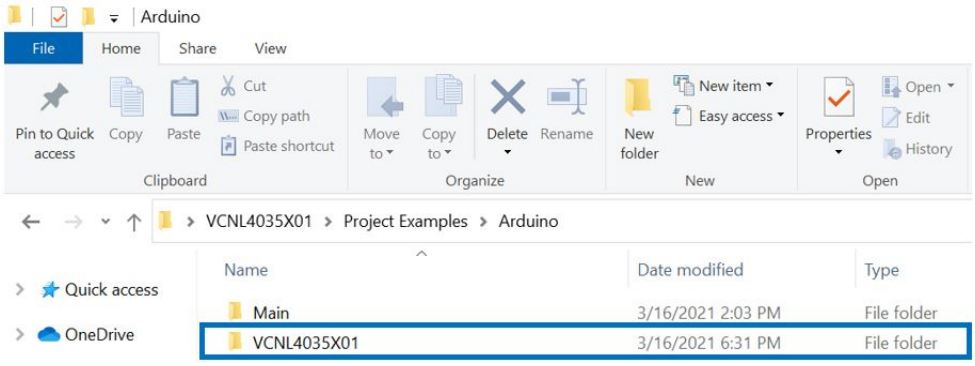

Fig. 14 - Copy the Arduino Library Files

Step 4: Open **"Part Name"** folder and open **typedefinition.h**

Using The Software Code Library

Step 5: Activate the code for Arduino platforms by defining **#define wirelib** in **typedefinition.h** and comment the line **#define i2ct3**. On the other hand, activate the code for Teensy platforms by defining **#define i2ct3** in **typedefinition.h** and comment the line **#define wirelib**

Note

- Arduino platforms use Wire.h library while Teensy platforms use i2c_t3.h library

Step 6: Open the Arduino Sketch

Step 7: Click File > Open

Step 8: Select the directory to the file **Main.ino**, which is located in the above **"Main"** folder

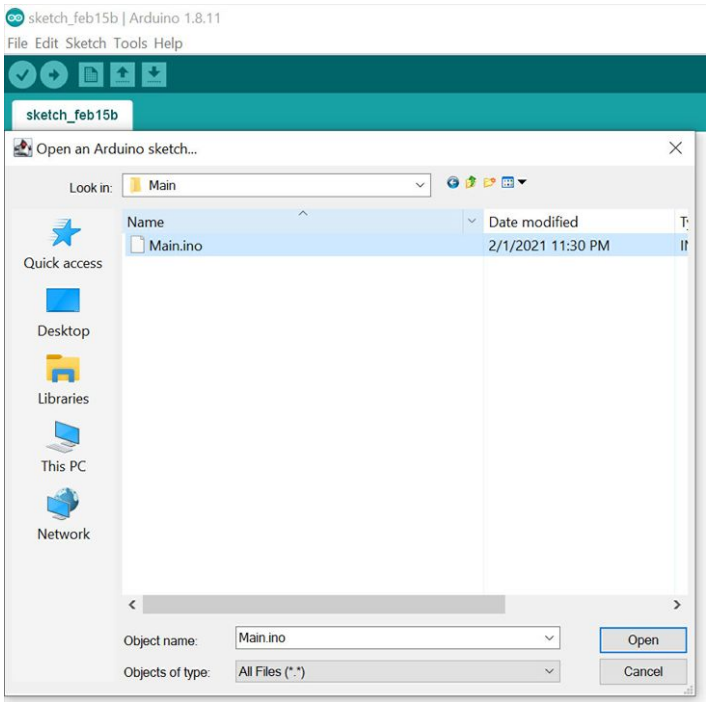

Fig. 15 - Open the Arduino Project

Step 9: Run **Main.ino**

APPLICATION NOTE

If you experience further issues, please contact us via the support E-Mail: sensorstechsupport@vishay.com