

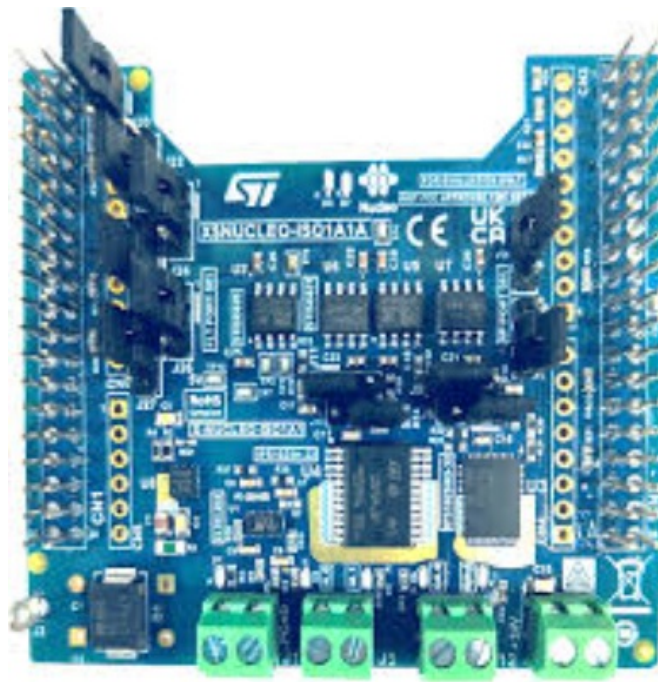


## Contents [ [hide](#) ]

- [1 STMicroelectronics UM3469 X-CUBE-ISO1 Software Expansion](#)
- [2 Introduction](#)
- [3 Acronyms and abbreviations](#)
- [4 What is STM32Cube?](#)
- [5 X-CUBE-ISO1 software expansion for STM32Cube](#)
- [6 System setup guide](#)
- [7 Documents / Resources](#)
  - [7.1 References](#)



## STMicroelectronics UM3469 X-CUBE-ISO1 Software Expansion



## Introduction

The X-CUBE-ISO1 expansion software package for STM32Cube runs on the STM32 and includes firmware for the X-NUCLEO-ISO1A1. The software provides an easy-to-use solution for the development of a basic PLC device provided by the X-NUCLEO. The expansion is built on STM32Cube software technology to ease portability across different STM32 microcontrollers.

The software comes with an implementation running on the X-NUCLEO-ISO1A1 expansion board connected to a NUCLEO-G071RB development board (or either a NUCLEO-G0B1RE or a NUCLEO-G070RB). From now on, in the document only the NUCLEO-G071RB will be mentioned for simplicity.

The X-NUCLEO-ISO1A1 board is designed to support the stacking of two boards with appropriate jumper settings to extend the input and output capabilities.

## Acronyms and abbreviations

**Table 1. List of acronyms**

Acronym	Description
PLC	Programmable logic controller

API	Application programming interface
PWM	Pulse width modulation
GPIO	General-purpose input/output.
HAL	Hardware abstraction layer
PC	Personal computer
FW	Firmware

## What is STM32Cube?

STM32Cube™ represents the STMicroelectronics initiative to make developers' lives easier by reducing development effort, time, and cost. STM32Cube covers the STM32 portfolio.

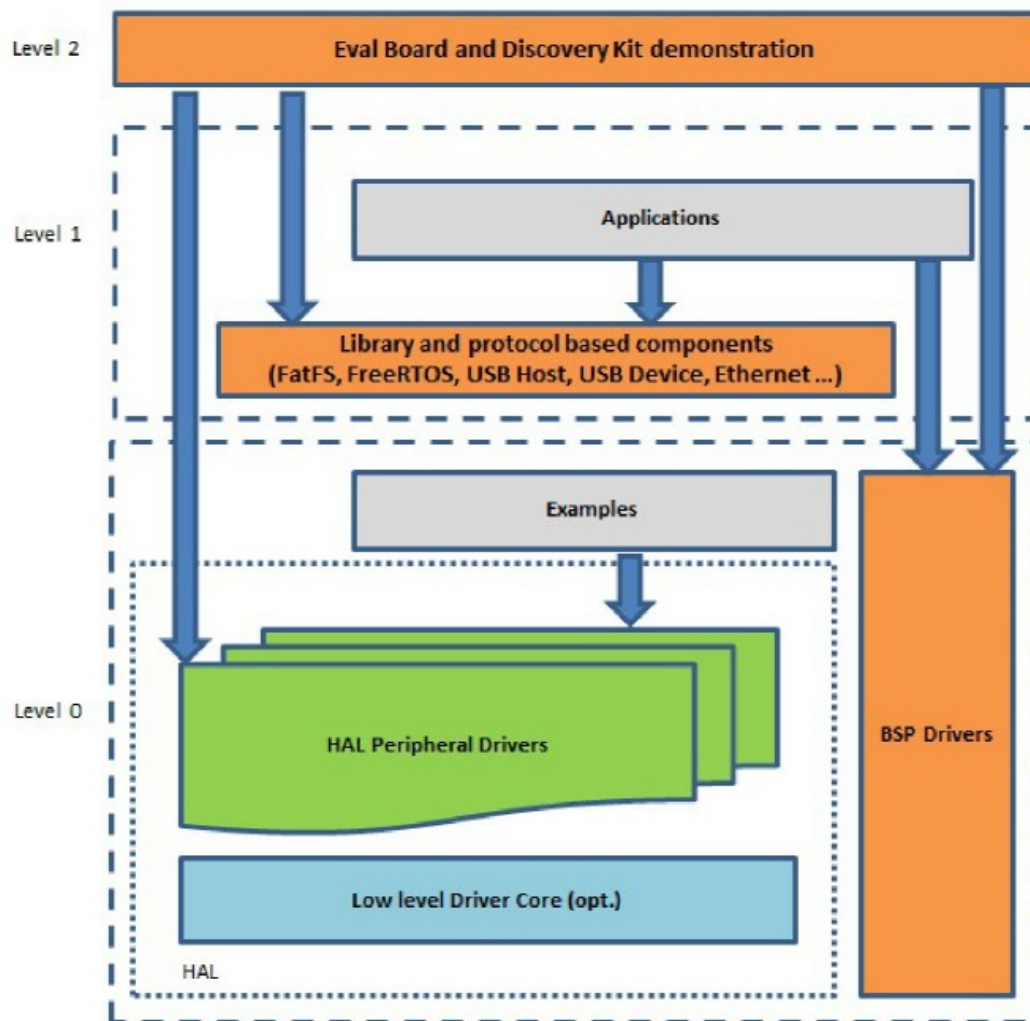
STM32Cube version 1.x includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform specific to each series (such as the STM32CubeG0 for the STM32G0 series), which includes:
  - the STM32Cube HAL embedded abstraction-layer software, ensuring maximized portability across the STM32 portfolio
  - a consistent set of middleware components such as RTOS, USB, TCP/IP, and graphics
  - all embedded software utilities with a full set of examples.

## STM32Cube architecture

The STM32Cube firmware solution is built around three independent levels that can easily interact with one another, as described in the diagram below.

**Figure 1. Firmware architecture**



## X-CUBE-ISO1 software expansion for STM32Cube

### Overview

The firmware for the X-NUCLEO-ISO1A1, industrial isolated input/output expansion board, developed around STM32 environments and libraries, leverages the high-performance MCU of STM32 Nucleo boards to manage digital inputs, outputs with integrated diagnostics along with dynamic current limit, and PWM signal generation. It features comprehensive board configuration and control, including frameworks for default and alternate conditions, macros for setting pre-scaler values, and definitions for GPIO ports and pins.

It supports various sample application use cases such as digital input to output mirroring, UART communication through the Nucleo board, fault detection, test cases, and PWM generation which can be directly used and can be easily customized and expanded.

The API provides a robust set of functions for digital input/output control, fault detection, and board status updates, with configuration settings for running two boards simultaneously in different modes. Specific API functions are available for initializing, starting, stopping, and configuring PWM signals for digital output channels.

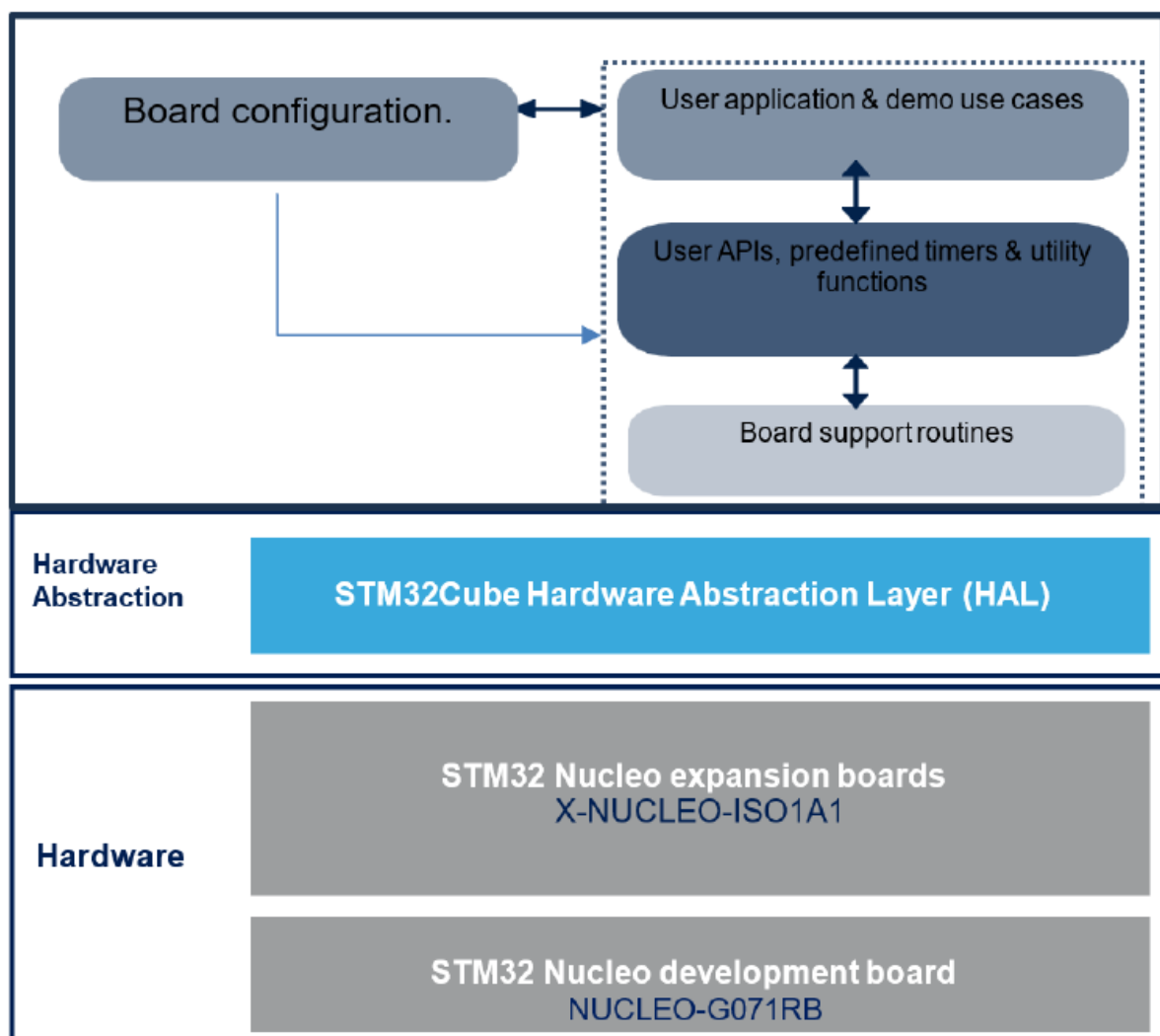
The board support package includes functions to control and monitor the GPIO pins interfaced with IPS1025H-32 and read the state of the GPIO pins interfaced with CLT03-2Q3 via a digital isolator.

Configuration and initialization are based on STM32CubeMX, with development and debugging supported by STM32CubeIDE, IAR Systems, and Keil® tools.

## Architecture

The firmware for the X-NUCLEO-ISO1A1 can be divided into several distinct functional blocks, each responsible for various aspects of the system's operations:

**Figure 2. X-CUBE-ISO1 software architecture**



- **Board Configuration and Control:**

- The `board_config.h` file contains macros to configure the board to run in default or alternate conditions, or both. It also includes definitions for pre-scaler values and GPIO ports and pins.
- This block ensures that the board is set up correctly for the desired operating conditions and that all necessary hardware configurations are in place.

- **Application Use Cases:**

- The `st_iso_app.h` and `st_iso_app.c` files contain application use cases designed to test various functionalities of the board.
- These use cases include digital input to output mirroring, fault detection tests, and PWM signal generation.
- Example configurations are provided for running two boards simultaneously in different modes, demonstrating the versatility and flexibility of the firmware.

- **API Functions:**

- The `iso1a1.h` and `iso1a1.c` files provide a comprehensive set of APIs to support various functionalities.
- These APIs include functions for digital input/output control, fault detection, and board status updates.
- The APIs are designed to be simple and intuitive, making it easy for users to interact with the board and perform necessary operations.

- **PWM Signal Control:**

- The `pwm_api.h` and `pwm_api.c` files contain specific API functions related to PWM signal generation.
- These functions allow for initializing, configuring, starting, and stopping PWM signals for digital output channels.
- The PWM functionality is not default choice. Board configuration has be modified to enable these. Refer to Section 3.5: APIs for more details.

- **Board Support Package:**

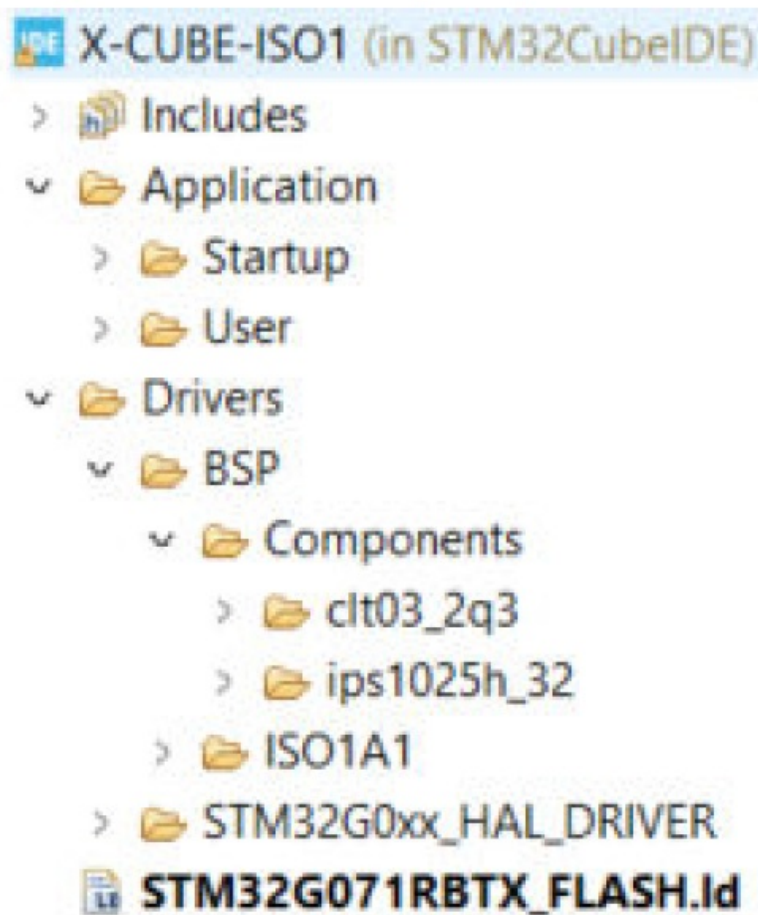
- The board support package includes files for controlling and monitoring the GPIO pins interfaced with IPS1025H-32 and reading the state of the GPIO pins interfaced with CLT03-2Q3.
- The `ips1025h_32.h` and `ips1025h_32.c` files provide functions to set, clear, and detect faults on the GPIO pins interfaced with IPS1025H-32.
- The `clt03_2q3.h` and `clt03_2q3.c` files provide functions to read the state of the

GPIO pins interfaced with CLT03-2Q3.

The demonstration firmware implements several simple use cases to showcase the capabilities of the system. These use cases and user APIs are executed in a coordinated manner to ensure smooth operation and accurate results. The architecture is designed to be easily expandable, allowing users to add new functionalities and use cases as needed. Default configuration is provided for running one board with digital industrial I/Os. The jumper settings are also needed to be in default mode as described in Table 2. Digital input Digital out mirroring (DIDO) is the default firmware application usecase.

## Folder structure

**Figure 3. X-CUBE-ISO1 package folder structure**



**The following folders are included in the software package:**

- Documentation contains a compiled HTML file generated from the source code, detailing the software components and APIs.
- Drivers contains:

- an STM32Cube HAL folder, located in the subfolders STM32G0xx\_HAL\_Driver. These files are not described here as they are not specific to the X-CUBE-ISO1 software but come directly from the STM32Cube framework.
- a CMSIS folder which contains the Cortex® microcontroller software interface standard files from Arm. These files are vendor-independent hardware abstraction layer for the Cortex®-M processor series. This folder also comes unchanged from the STM32Cube framework.
- a BSP folder containing the codes for the components IPS1025H-32 and CLT03-2Q3 and APIs related to X-NUCLEO-ISO1A1.
- Application contains the user folder which contains the main.c file, the application use case file, st\_iso\_app.c and the board\_config.h file, provided for the NUCLEO-G071RB platform.

## **BSP folder**

The X-CUBE-ISO1 software uses two different component files, which are inside BSP/Components:

### **IPS1025**

The ips1025h\_32.h and ips1025h\_32.c files provide a comprehensive driver implementation for the GPIO pins interfaced with IPS1025H-32, including complete functionality for controlling all pins and detecting faults. These files implement functions for initializing the device, setting and clearing channel status, detecting fault conditions, and managing PWM functionality. The driver supports multiple devices and channels, with complete capabilities for both individual channel or as a group.

### **CLT03**

The clt03\_2q3.h and clt03\_2q3.c files implement a full-featured driver for the GPIO pins interfaced with CLT03-2Q3, with complete capabilities for reading all pin states. The driver provides functions to initialize the device, read individual channel status, and obtain status information for all channels simultaneously. It supports multiple device configurations and maintains internal state for effective channel management.

The X-CUBE-ISO1 software APIs are divided into two major source files, which are inside the ISO1A1 subfolder:



## **ISO1A1**

The ISO1A1 files encompass a comprehensive set of API functions designed for board configuration, component interaction, and fault management. These functions facilitate reading and writing operations, fault detection and updates, and include various helper utilities to support the primary API functions. Additionally, the files provide functionality for LED control, GPIO initialization, interrupt handling, and UART communication.

### **PWM API**

The PWM API provides functions for initializing, configuring, starting, and stopping PWM signals. It allows setting the PWM frequency and duty cycle for specified timer pins, ensuring precise control over PWM operations.

### **Application folder**

The Application folder contains the main files required for the firmware, including headers and source files. Below is a detailed description of the files in this folder:

- `board_config.h`: Configuration macros for the board.
- `main.c`: Main program (code of the example which is based on the library for ISO1A1).
- `st_iso_app.c`: Application functions for board testing and configuration.
- `stm32g0xx_hal_msp.c`: HAL initialization routines.
- `stm32g0xx_it.c`: Interrupt handler.
- `syscalls.c`: System call implementations.
- `systemmem.c`: System memory management.
- `system_stm32g0xx.c`: System initialization.

### **Software required resources**

The Nucleo device controls and communicates with the X-NUCLEO-ISO1A1 board via GPIOs. This requires the use of several GPIOs for input, output, and fault detection of the industrial IO devices contained in the X-NUCLEO-ISO1A1 board. Refer to the Hardware user manual UM3483 for more details and the jumper configurations.

### **Board configuration (`board_config.h`)**

The `board_config.h` file defines the resources used and the configurational macros to configure the software according to the board configuration. It handles up to two boards (such as the stacking of two boards).

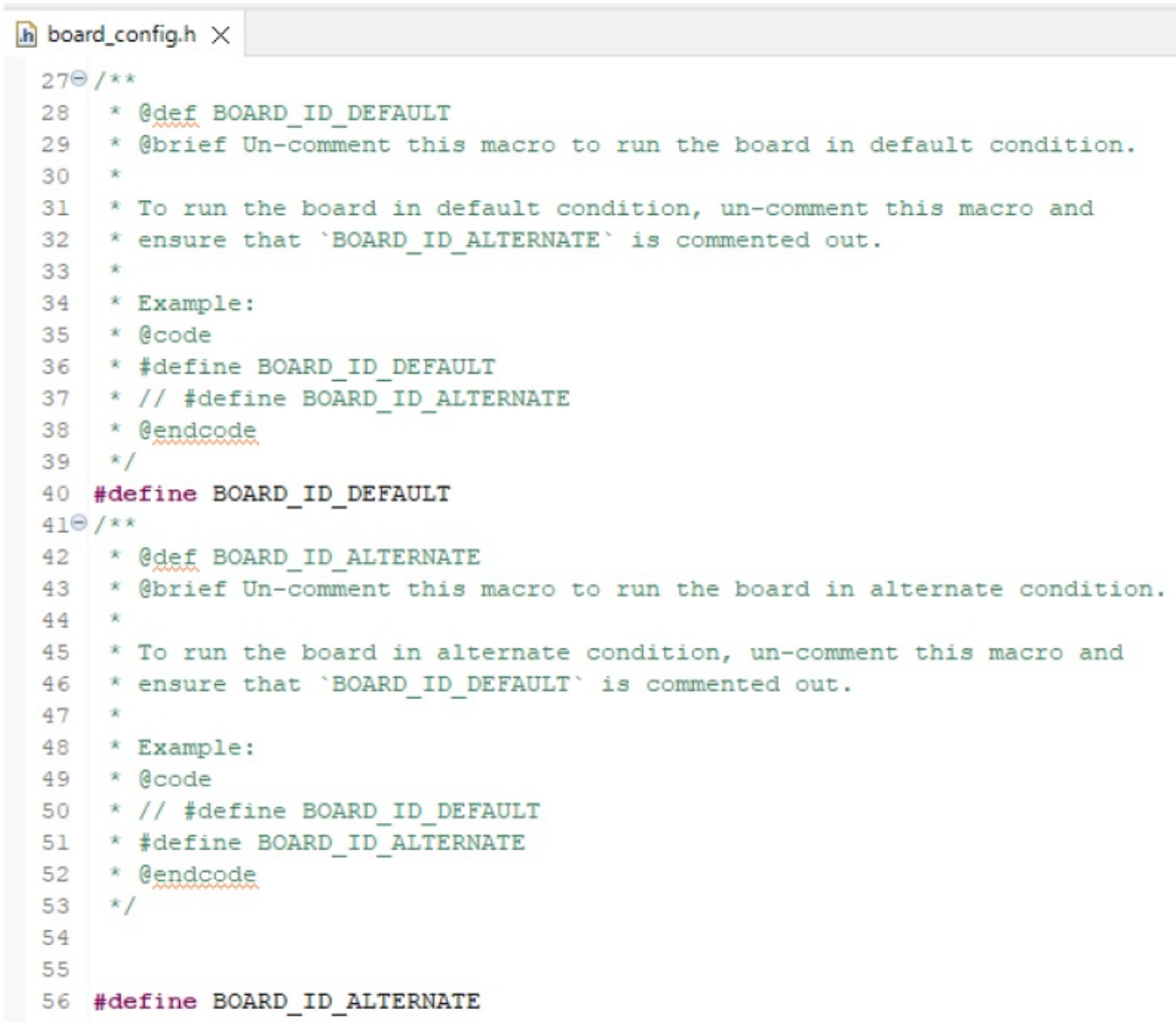
The software DEFAULT configuration is aligned with the X-NUCLEO-ISO1A1 expansion board with its jumpers in the default positions. To configure the software for X-NUCLEO-ISO1A1 in its default setting, uncomment the BOARD\_ID\_DEFAULT macro in the board\_config.h file.

The software ALTERNATE configuration is set by uncommenting the BOARD\_ID\_ALTERNATE macro in the board\_config.h file and changing the jumper positions on the board.

To use two boards simultaneously in a stack-up configuration, uncomment both BOARD\_ID\_DEFAULT and BOARD\_ID\_ALTERNATE macros and ensure one board's jumpers are in the default position and the other in the alternate position. Note that having both boards in the same configuration (either both in default or both in alternate) is not recommended and may result in undesired behavior.

When running only one board, ensure that the software is configured for just one configuration and the macro corresponding to the other configuration is commented.

**Figure 4. Software configured for two boards stacked**



```
board_config.h X
27 /**
28  * @def BOARD_ID_DEFAULT
29  * @brief Un-comment this macro to run the board in default condition.
30  *
31  * To run the board in default condition, un-comment this macro and
32  * ensure that `BOARD_ID_ALTERNATE` is commented out.
33  *
34  * Example:
35  * @code
36  * #define BOARD_ID_DEFAULT
37  * // #define BOARD_ID_ALTERNATE
38  * @endcode
39  */
40 #define BOARD_ID_DEFAULT
41 /**
42  * @def BOARD_ID_ALTERNATE
43  * @brief Un-comment this macro to run the board in alternate condition.
44  *
45  * To run the board in alternate condition, un-comment this macro and
46  * ensure that `BOARD_ID_DEFAULT` is commented out.
47  *
48  * Example:
49  * @code
50  * // #define BOARD_ID_DEFAULT
51  * #define BOARD_ID_ALTERNATE
52  * @endcode
53  */
54
55
56 #define BOARD_ID_ALTERNATE
```

## Pre-scalers

We can configure the pre-scaler values in board\_config.h to achieve different frequency ranges for the PWM output by setting the appropriate macros. To use a pre-scalar value, uncomment the corresponding macro and comment others. By default, DEFAULT\_PRESCALAR is used.

- PRESCALER\_1
- PRESCALER\_2
- DEFAULT\_PRESCALAR

The prescaler values are used only when timers are being used, and not required for any basic I/O operation. The values of the pre-scalar macros and their corresponding frequency ranges can be looked in the code documentation or in the code itself.

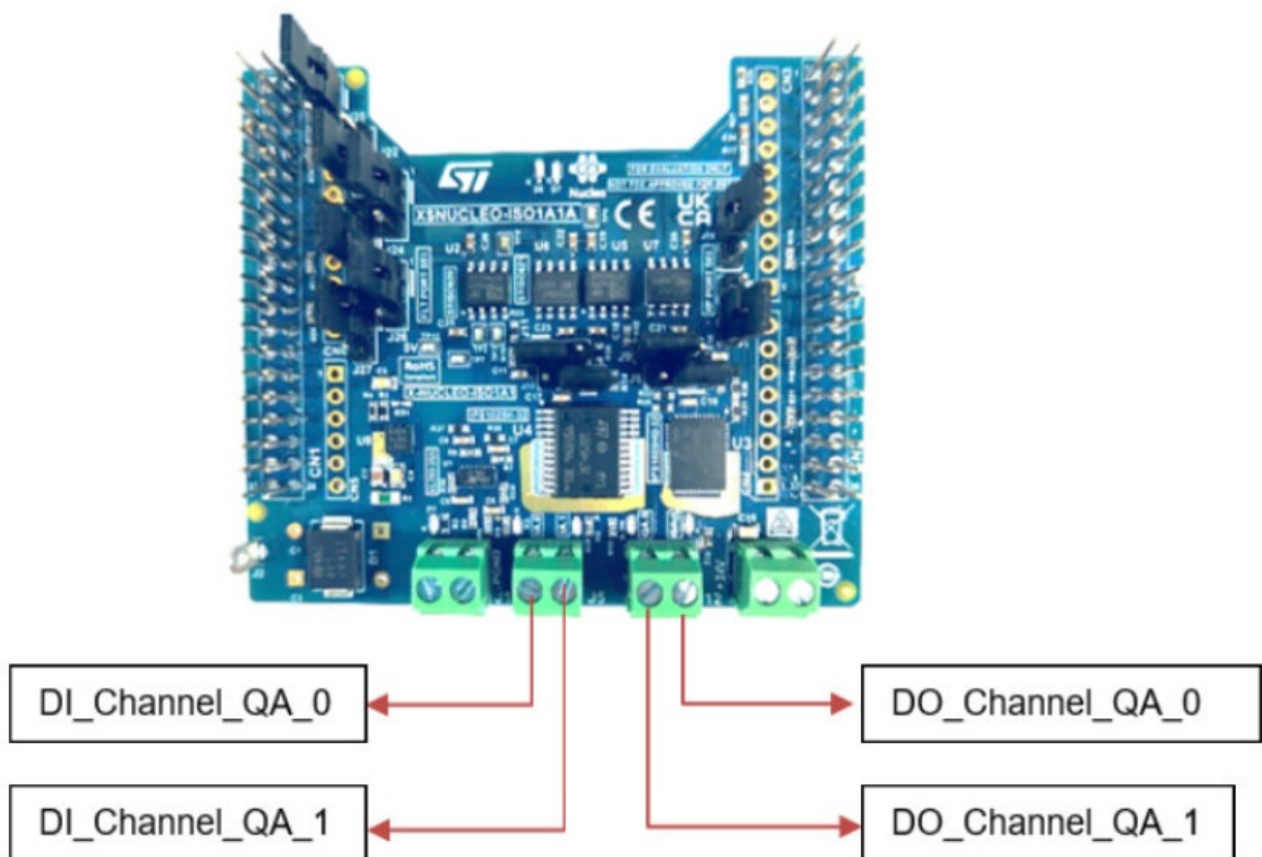
## Heartbeat LED

We can configure the green user LED, D7 to blink in a heartbeat fashion as a test for proper connection to the NUCLEO-G071RB board. The macro, `HEARTBEAT_LED` when uncommented, blinks the green LED on X-NUCLEO-ISO1A1 when it is connected to the NUCLEO. It remains on for 1 second and off for 2 seconds, with the timing taken care of by timers. When it is not used or any function involving LEDs being called, the macro should be uncommented.

## Input and output GPIO configuration

Each X-NUCLEO-ISO1A1 board is equipped with two input ports and two output ports. The capabilities of the board can be expanded by stacking two X-NUCLEO-ISO1A1 boards on top of each other, thereby enabling the use of four digital input ports and four digital output ports. The provided software includes comprehensive APIs that facilitate reading, setting, and clearing the ports. Additionally, the APIs allow for the simultaneous setting, reading, or clearing of all ports. Detailed information about the API functions is available in the code documentation as well as in the API section of this document.

**Figure 5. X-NUCLEO-ISO1A1 expansion board in default configuration**



Here the prefix DI indicates digital input port and DO indicates digital output port. For alternate configuration, the software uses the same naming conventions with \_alt suffix attached.

The following table details the GPIO macros defined in the software corresponding to various IO ports:

**Table 2. GPIOs allocated for Default and alternative software configurations**

Name	Function	Default configuration	Alternate configuration
INPUT PIN	Input pin 1	GPIOC, IA0_IN_1_PIN	GPIOD, IA0_IN_1_PIN
	Input pin 2	GPIOD, IA1_IN_2_PIN	GPIOC, IA1_IN_1_PIN
OUTPUT PIN	Output pin 1	GPIOC, QA0_CNTRL_1_PIN	GPIOD, QA0_CNTRL_1_PIN
	Output pin 2	GPIOC, QA1_CNTRL_2_PIN	GPIOC, QA1_CNTRL_2_PIN
FAULT PIN	Fault pin 1	GPIOC, FLT1_QA0_2_OT_PIN	GPIOD, FLT1_QA0_1_OT_PIN
	Fault pin 2	GPIOC, FLT2_QA0_2_OL_PIN	GPIOD, FLT2_QA0_1_OL_PIN
	Fault pin 3	GPIOC, FLT1_QA1_2_OT_PIN	GPIOC, FLT1_QA1_1_OT_PIN
	Fault pin 4	GPIOC, FLT2_QA1_1_OL_PIN	GPIOD, FLT2_QA1_2_OL_PIN
CONFIGURATION MACRO		BOARD_ID_DEFAULT	BOARD_ID_ALTERNATE

## Timers and PWM

Timers can be used in the X-CUBE-ISO1 firmware to generate PWM signals for specific pins. By default, timers are not initialized except TIM3. The respective timers should be initialized before generating the PWM signals and the respective output ports must be initialized in the PWM mode.

For normal GPIO input/output operations, there is no need to configure any timer or output port, as it is taken care of by default. However, if once the output pins are set in the PWM mode, we need to reconfigure them in GPIO mode to be used as GPIO pins.

**Note:** When the output pins are being used for PWM generation, the GPIO output is disabled, both functionalities cannot be implemented simultaneously. To re-enable GPIO after PWM usage, one can call the API function `ST_ISO_BoardConfigureDefault()` or `ST_ISO_InitGPIO()` to configure all ports as GPIO at once or `ST_ISO_Init_GPIO()` with a particular GPIO port and pin.

As mentioned above, the software also uses one timer by default, TIM3, which is used for user LED timing, clock, and UART timing implementation. It is configured for a period of 1 second by default.

The following table details the timers available for each pin in our code:

**Table 3. Timers available for each pin**

Pin name	Software representation	Timer	Timer channel	Alternate function
QA0_CNTRL_1_PIN	QA_0	TIM2	TIM_CHANNEL_4	GPIO_AF2_TIM2
QA1_CNTRL_2_PIN	QA_1	TIM1	TIM_CHANNEL_3	GPIO_AF2_TIM1
QA0_CNTRL_2_PIN	QA_0_ALT	TIM1	TIM_CHANNEL_4	GPIO_AF2_TIM1
QA1_CNTRL_1_PIN	QA_1_ALT	TIM17	TIM_CHANNEL_1	GPIO_AF2_TIM17

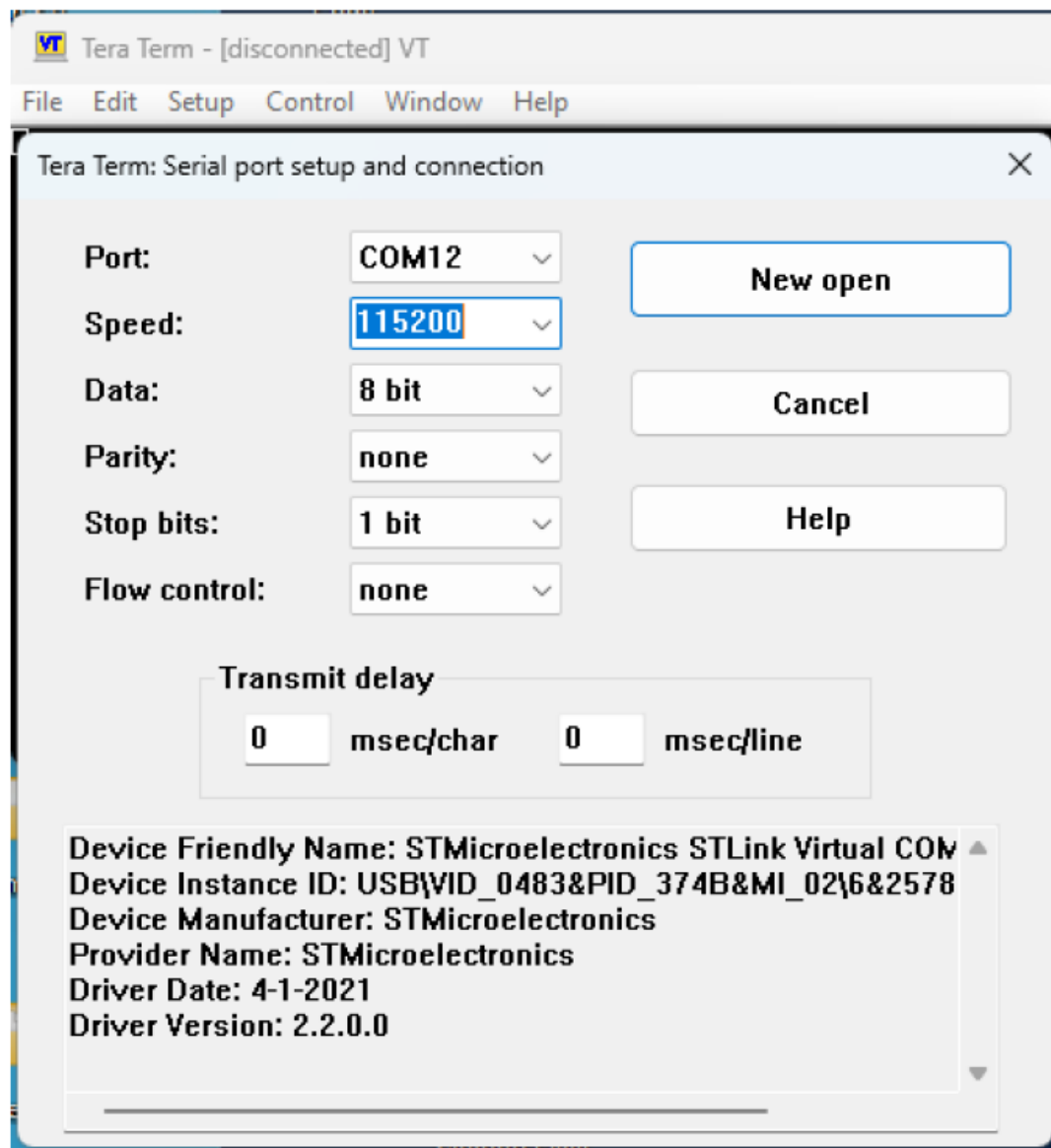
## **Additional utilities of the firmware**

The firmware includes additional utilities to enhance the functionality of the X-NUCLEO-ISO1A1 evaluation board. Some of which are described below.

### **UART**

The UART communication feature allows for real-time monitoring and debugging of the board status through PC utilities such as TeraTerm, PuTTY and other similar applications. The software enables the UART data transmission through the UART present in the NUCLEO-G071RB board. The `ST\_ISO\_UART` function sends detailed board status information over UART, including system uptime, firmware configuration, and fault status. This data can be viewed using any serial port application, such as TeraTerm. The `ST\_ISO\_APP\_DIDOandUART` function combines digital input/output operations with UART communication, transmitting the status of all input and output channels at specified intervals. Below are the configuration settings and a sample of how the data appears in TeraTerm. The port name could vary based on the system and serial port being used.

**Figure 6. TeraTerm serial port configurational settings**





**Figure 7. Board status visible in TeraTerm sent over UART**

```
Time since board is up and working: 00:00:20
Firmware is configured for both default and alternate modes.
Board Status Information (Default):
Board ID: Board_ID_Default
QA0 Mode: Input_output_Mode
QA1 Mode: Input_output_Mode
IA0 Mode: Input_output_Mode
IA1 Mode: Input_output_Mode
Input Status:
Channel 0: No input
Channel 1: No input
Output Status:
Channel 0: No output
Channel 1: No output
Fault Status:
QA0_OL: 0
QA0_OT: 0
QA1_OL: 0
QA1_OT: 0
Board Status Information (Alternate):
Board ID: Board_ID_Alternate
QA0 Mode: Input_output_Mode
QA1 Mode: Input_output_Mode
IA0 Mode: Input_output_Mode
IA1 Mode: Input_output_Mode
Input Status:
Channel 0: Input present
Channel 1: No input
Output Status:
Channel 0: Output present
Channel 1: No output
Fault Status:
QA0_OL: 0
QA0_OT: 1
QA1_OL: 1
QA1_OT: 1
```

## IO pin mode configuration

The IO pin mode configuration utility allows users to set the input and output ports of the board using the `ST_ISO_BoardConfigure()` function. This function supports configuring two output ports (QA0, QA1) and two input ports (IA0, IA1) to either Input/Output mode, PWM output mode, or Interrupt input mode. By adjusting the parameters and calling this function, users can easily customize the board's IO configuration to meet specific needs.

In Input/Output mode, the utility initializes the GPIO pins for general-purpose digital operations. In PWM output mode, it sets up the timers for precise PWM signal control. When in Interrupt input mode, the utility configures the pins to handle interrupts, allowing for responsive event-driven programming.

## Interrupt handling

For handling FAULT signals, the software enables the associated interrupt lines, allowing for responsive event-driven programming. A customized handler can be associated with these interrupts via the

HAL\_GPIO\_EXTI\_Rising\_Callback function defined in the API. The software includes features for initializing GPIO pins in interrupt mode via ST\_ISO\_BoardConfigure function and configuring specific actions in the EXTI IRQ handlers. This allows users to customize how the board responds to external events, ensuring it can effectively manage various fault conditions and triggers.

## **APIs**

The X-CUBE-ISO1 software API provides a comprehensive set of functions to control and monitor the X-NUCLEO-ISO1A1 board, including PWM signal generation and GPIO operations. The API is designed to be easy to use and integrate into various applications, providing flexibility and control over the board's functionality.

The X-CUBE-ISO1 software API is defined in the BSP/ISO1A1 folder. Its functions are prefixed by ST\_ISO. The API visible to the applications through the iso1a1.c and pwm\_api.c files is a combination of constants, data structures, and functions. Sample firmware applications utilizes these APIs to show some of possible usages of these functions.

### **The X-CUBE-ISO1 software package provides two sets of APIs:**

- ISO1A1 API
- PWM API

## **ISO1A1 API**

The ISO1A1 API is defined in the iso1a1.h and iso1a1.c files. It provides functions to configure and control the ISO1A1 board, including GPIO input/output operations and fault detection.

## **Key functions**

- ST\_ISO\_BoardConfigureDefault: Configures the board's IO ports with default GPIO configuration.
- ST\_ISO\_BoardConfigure: Configures the mode of the input and output ports for the board.
- ST\_ISO\_BoardInit: Initializes the board hardware.

- `ST_ISO_BoardMapInit`: Initializes the board functionality based on the channel handles configuration.
- `ST_ISO_GetFWVersion`: Returns the current firmware version.
- `ST_ISO_GetChannelHandle`: Retrieves the channel handle for a specified channel name.
- `ST_ISO_InitGPIO`: Initializes the specified GPIO pin with the given module ID.
- `ST_ISO_InitInterrupt`: Initializes the specified GPIO pin as an interrupt with the given module ID.
- `ST_ISO_EnableFaultInterrupt`: Initializes the fault GPIO pins in interrupt mode.
- `ST_ISO_SetChannelStatus`: Sets the status of a specified channel.
- `ST_ISO_SetOne_DO`: Sets a single digital output channel.
- `ST_ISO_ClearOne_DO`: Clears a single digital output channel.
- `ST_ISO_WriteAllChannels`: Writes data to all digital output channels.
- `ST_ISO_GetOne_DI`: Gets the status of a single digital input channel.
- `ST_ISO_ReadAllChannel`: Reads the status of all input channels.
- `ST_ISO_ReadAllOutputChannel`: Reads the status of all output channels.
- `ST_ISO_ReadFaultStatus`: Reads the fault status from all fault detection ports.
- `ST_ISO_ReadFaultStatusPolling`: Tests the fault detection of the boards in polling mode.
- `ST_ISO_DisableOutputChannel`: Disables the output for that channel.
- `ST_ISO_UpdateBoardStatusInfo`: Updates the board status information.
- `ST_ISO_UpdateFaultStatus`: Updates the fault status for a specific channel.
- `ST_ISO_BlinkLed`: Blinks the specified LED with a given delay and repeat count.
- `ST_ISO_UART`: Sends the board status information over UART.
- `ST_ISO_SwitchInit`: Initializes the switch components.
- `ST_ISO_SwitchDeInit`: De-initializes the switch instance.
- `ST_ISO_DigitalInputInit`: Initializes the digital input components.
- `ST_ISO_DigitalInputDeInit`: De-initializes the digital input instance.

## **PWM API**

The PWM API is defined in the `pwm_api.h` and `pwm_api.c` files. It provides the following functions to initialize and control PWM signals for specific pins.

- `ST_ISO_Init_PWM_Signal`: Initializes the timers and specific pin for the PWM signal.

- `ST_ISO_Set_PWM_Frequency`: Sets the PWM frequency for the specific pin.
- `ST_ISO_Set_PWM_Duty_Cycle`: Sets the PWM duty cycle for the specific pin.
- `ST_ISO_Start_PWM_Signal`: Starts the PWM signal on the specific pin.
- `ST_ISO_Stop_PWM_Signal`: Stops the PWM signal on the specific pin.

To start a PWM signal on a respective channel, firstly call the `ST_ISO_Init_PWM_Signal` function, then set the desired frequency and duty cycle by calling the `ST_ISO_Set_PWM_Frequency` and `ST_ISO_Set_PWM_Duty_Cycle` functions respectively and then you can start the PWM signal by calling the `ST_ISO_Start_PWM_Signal` function and stop by calling `ST_ISO_Stop_PWM_Signal`.

The function needs to be called with the corresponding pin name and the timers available, the details of which have been provided in table 3. Different output channels can be set up with different frequencies and duty cycles; changing frequency or duty cycle does not affect the other, it remains the same.

Detailed technical information about the APIs available to the user can be found in a compiled HTML file located inside the “Documentation” folder of the software package where all the functions and parameters are fully described.

## **Application description**

The demonstration application implements several simple use cases. The `st_iso_app` and `board_config` files play a crucial role in setting up and using the board and its application functions. Before using these functions ensure the board and the software’s configuration are in sync with each other.

Application Functions (`st_iso_app.h` and `st_iso_app.c`)

The application functions are prefixed by `ST_ISO_APP`; they are the top-level functions visible to the user which calls the API functions for their implementation. The application functions can be called in the `main.c` file for their functioning.

- **Use Case Selection**: The user can uncomment the desired use case macro in the `st_iso_app.c` file. The function `ST_ISO_APP_SelectUseCaseMacro()`, called in `main.c`, initializes that use case, and the function `ST_ISO_APP_SelectedFunction()` implements it in `main.c`. This approach allows for easy configuration of the operational

mode by simply modifying the macro definitions, ensuring that the appropriate functionality is executed based on the selected use case. By default, the use case DIDO is selected, and the user does not have to make any changes to the code to implement it.

- Digital Input to Digital Output Mirroring (ST\_ISO\_APP\_UsecaseDIDO): This function reads the status of all input channels and writes the same status to all output channels. It is useful for mirroring digital inputs to digital outputs.
- Digital Input to Digital Output Mirroring with UART (ST\_ISO\_APP\_DIDOandUART): This function mirrors the digital inputs to digital outputs, similar to the ST\_ISO\_APP\_UsecaseDIDO function. Additionally, it transmits the board status through the UART interface on the Nucleo device, allowing the status to be viewed on a serial port using applications like Tera Term.
- Test Case Function (ST\_ISO\_APP\_TestCase): This function performs a series of tests and actions based on the board configuration. It checks the fault status, reads the status of two digital input channels, and performs actions based on their values. This function helps in evaluating the board's performance and functionality quickly and getting visual feedback through different LED patterns. Ensure the HEARTBEAT\_LED macro in the board\_config.h file is commented to observe proper LED patterns.
- PWM Generation (ST\_ISO\_APP\_PWM\_OFFSET): This function starts the PWM signal on both output channels with a frequency of 1 Hz and duty cycle 50%. It initializes the PWM signal, sets the frequency and duty cycle, and starts the PWM signal for the specified board ID. The PWM signal is generated with an offset between both channels and thus they are not in phase.
- Fault Detection Test (ST\_ISO\_APP\_FaultTest): This function evaluates the fault detection by motoring inbuilt diagnostic pins of smart output module IPS1025. in either polling or interrupt mode. It configures the fault detection mode, initializes fault detection, and updates the fault status structure based on the selected mode. This function is crucial for ensuring the reliability and safety of the board by detecting and handling faults effectively. When it is in polling mode, the fault status is updated every second with the help of a timer and is reflected in the structure defaultBoardFaultStatus or alternateBoardFaultStatus. When it is in the interrupt mode, the fault status is updated only when the fault occurs, and it triggers the software to clear the corresponding output port.
- PWM Variation Test (ST\_ISO\_APP\_PwmVariationTest): This function is designed to

test the variation of PWM (Pulse Width Modulation) signals on different output channels based on the board configuration. It initializes the PWM signals for both default and alternate board configurations, setting their frequency to 100 Hz and the initial duty cycle to 0%. The function then varies the duty cycle from 0% to 100% in increments of 5%, and back from 100% to 0% in decrements of 5%, with a 2-second delay between each step. This controlled variation allows for the observation and evaluation of the PWM signal behavior on channels QA\_0 and QA\_1 for the default board, and QA\_0\_ALT and QA\_1\_ALT for the alternate board.

By following these configurations and utilizing the provided application functions, you can effectively set up and use the X-NUCLEO-ISO1A1 board for various demonstration use cases.

## **System setup guide**

### **Hardware description**

#### **STM32 Nucleo platform**

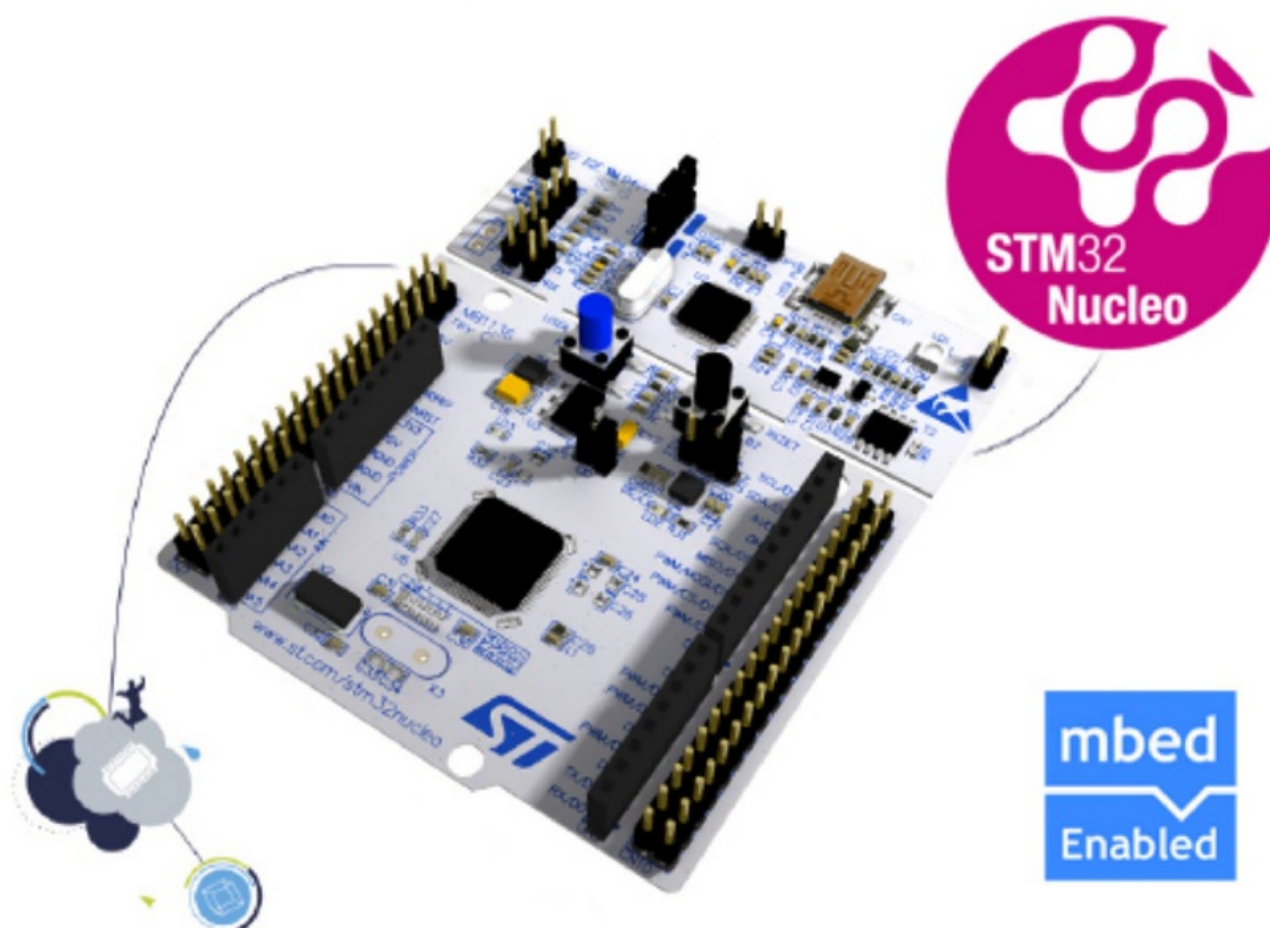
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

The Arduino® connectivity support and ST morpho connectors make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from.

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples.

**Figure 8. STM32 Nucleo board**



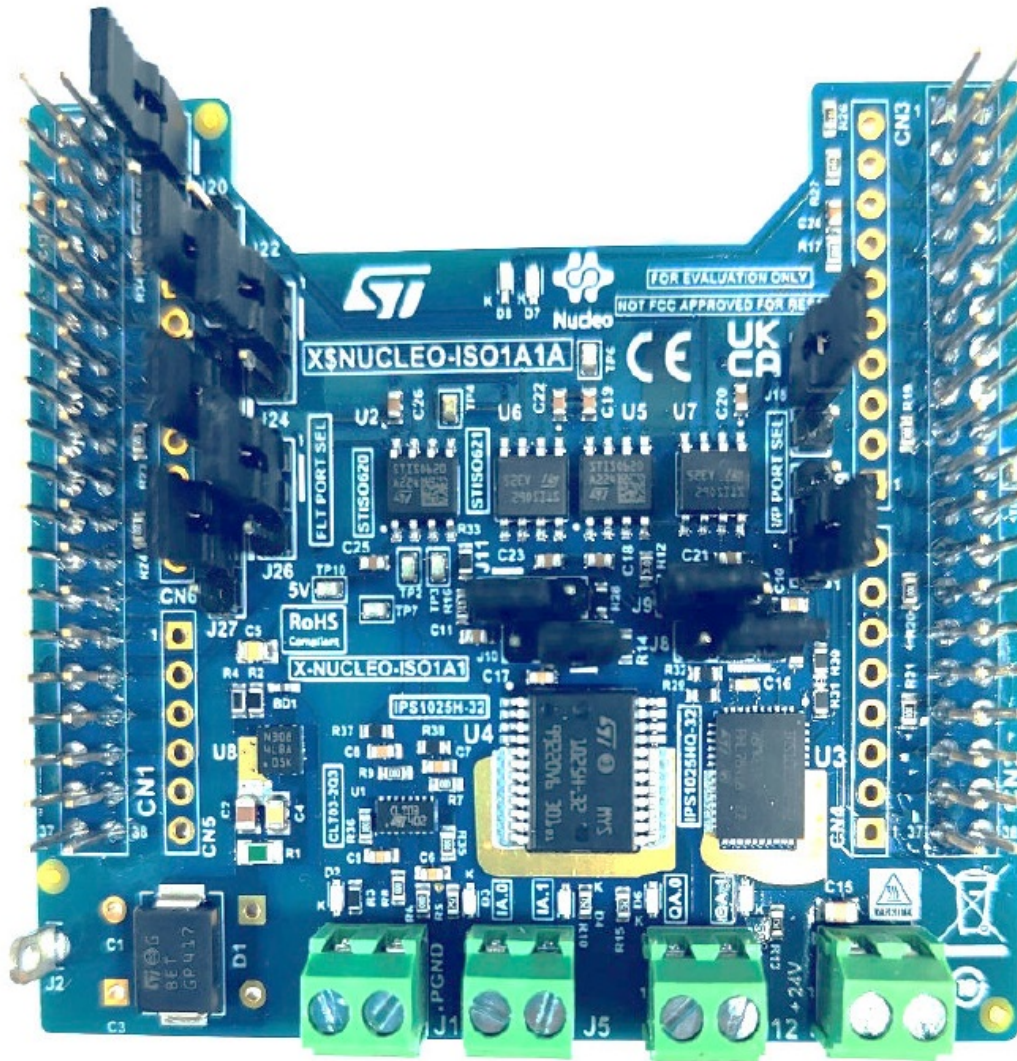
Information regarding the STM32 Nucleo board is available at [www.st.com/stm32nucleo](http://www.st.com/stm32nucleo)

### **X-NUCLEO-ISO1A1 expansion board**

The X-NUCLEO-ISO1A1 is an evaluation board with isolated industrial input/output designed to expand the STM32 Nucleo board and provide micro-PLC functionality. Two of the X-NUCLEO-ISO1A1 boards can be stacked together on top of an STM32 Nucleo board with the appropriate selection of jumpers on the expansion board to avoid conflict in GPIO interfaces. The UL1577 certified digital isolators STISO620 and STISO621 provide isolation between logic and process side components. Two current limited high-side inputs from the process side are realized through CLT03-2Q3. The CLT03-2Q3 provides protection, isolation, and energy-less status indication for industrial conditions, designed to meet standards such as IEC61000-4-2, IEC61000-4-4, and IEC61000-4-5. One each of the high-side switches IPS1025H-32/HQ-32 provides protected output up to 5.6 A with diagnostics and smart driving features. These can drive capacitive, resistive, or inductive loads. The X-NUCLEO-ISO1A1 allows rapid evaluation of the onboard ICs using the X-CUBE-ISO1 software package.



**Figure 9. X-NUCLEO-ISO1A1 expansion board**



### **Hardware setup**

The following hardware components are needed:

1. One STM32 Nucleo development platform (suggested order code: NUCLEO-GO71RB)
2. One industrial digital output expansion board (order code: X-NUCLEO-ISO1A1)
3. One USB type A to Micro USB cable to connect the STM32 Nucleo to the PC
4. An external power supply (24 V) and the associated wires to supply the X-NUCLEO-ISO1A1 expansion board.

### **Software setup**

The following software components are required to set up a suitable development environment for creating applications for the STM32 Nucleo equipped with the X-NUCLEO-ISO1A1 expansion board:



- X-CUBE-ISO1: an expansion for STM32Cube dedicated to application development which requires the use of the X-NUCLEO-ISO1A1 board. The X-CUBE-ISO1 firmware and related documentation is available on [www.st.com](http://www.st.com)
- Development toolchain and Compiler: the STM32Cube expansion software supports the three following environments:
  - IAR Embedded Workbench for ARM® (IAR-EWARM) toolchain
  - RealView Microcontroller Development Kit (MDK-ARM-STM32) toolchain
  - STM32CubeIDE.

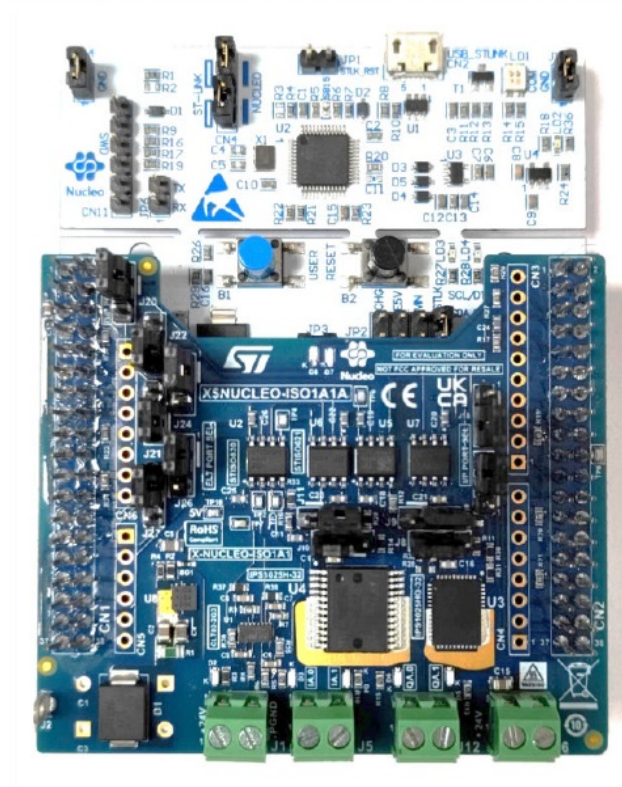
## Board setup

The board must be configured with the appropriate jumper settings as specified in the Hardware user manual (UM3483). Following these guidelines carefully is essential to ensure proper functionality and avoid potential issues.

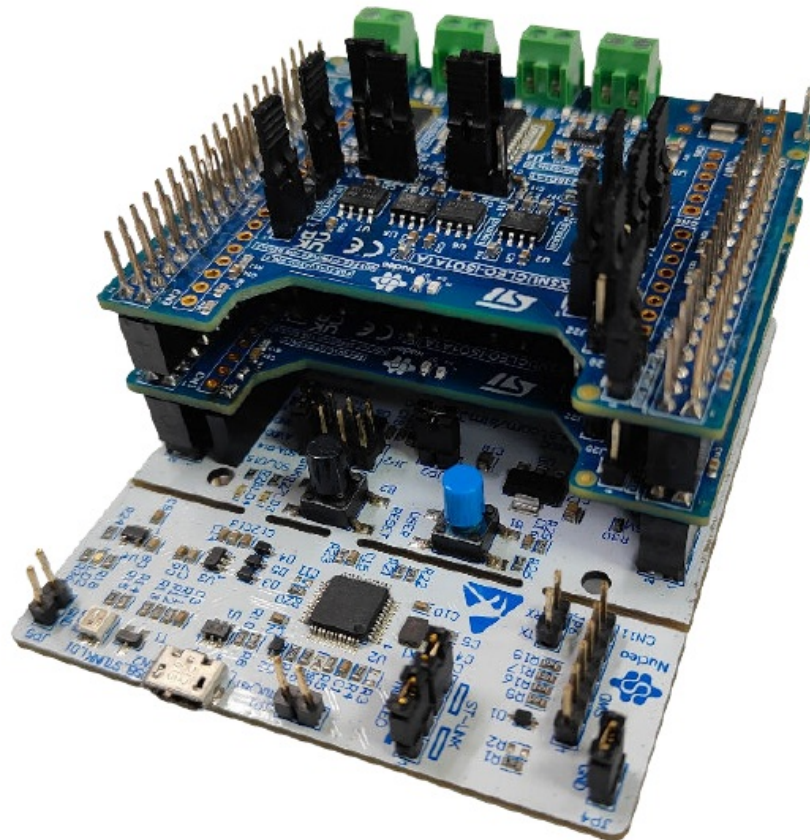
## System setup guide

This section describes how to set up different hardware parts before developing and executing an application on the STM32 Nucleo , NUCLEO-G071RB board with the X-NUCLEO-ISO1A1 expansion board.

**Figure 10. X-NUCLEO-ISO1A1 expansion board connected to an STM32 Nucleo development board**



**Figure 11. Two X-NUCLEO-ISO1A1 expansion boards in stacked configuration**



### Setup for X-CUBE-ISO1 expansion package

The X-NUCLEO-ISO1A1 must be configured with the specific jumper positions based on which configuration you are running the board. The details of which can be further looked in the hardware manual.

- **Step 1.** Plug the X-NUCLEO-ISO1A1 expansion board on top of the STM32 Nucleo via the morphoconnectors.  
If you are using two boards on top of each other, stack them as in Figure 11.
- **Step 2.** Connect the STM32 Nucleo board to a PC with the USB cable through USB connector CN1 to power the board.
- **Step 3.** Power the X-NUCLEO-ISO1A1 expansion board(s) on by connecting J1 to the 24V DC power supply. If using stacked boards, ensure both boards are powered.
- **Step 4.** Open your preferred toolchain (MDK-ARM from Keil, EWARM from IAR, or STM32CubeIDE).
- **Step 5.** Open the software project and make the necessary changes to the board\_config.h file according to the configuration of board(s) being used.
- **Step 6.** Set the appropriate use case macro in st\_iso\_app.c file or call the required

use case using ST\_ISO\_APP\_SelectUseCase function in main.c file along with any other desired function.

- **Step 7.** Build the project to compile all files and load the compiled code into the STM32 Nucleo board's memory.
- **Step 8.** Run the code on the STM32 Nucleo board and verify the expected behavior.

## Revision history

Table 4. Document revision history

Date	Revision	Changes
14-May-2025	1	Initial release.

## IMPORTANT NOTICE – READ CAREFULLY


STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products. No license, express or implied, to any intellectual property right is granted by ST herein. Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

# Documents / Resources

	<p><a href="#">STMicroelectronics UM3469 X-CUBE-ISO1 Software Expansion [pdf]</a> Use r Manual</p> <p>X-NUCLEO-ISO1A1, NUCLEO-G071RB, UM3469 X-CUBE-ISO1 Softwar e Expansion, UM3469, X-CUBE-ISO1 Software Expansion, Software Exp ansion</p>
---	---

## References

- [User Manual](#)

■ STMicroelectronics

◆ NUCLEO-G071RB, Software Expansion, STMicroelectronics, UM3469, UM3469 X-CUBE-ISO1 Software Expansion, X-CUBE-ISO1 Software Expansion, X-NUCLEO-ISO1A1

---

## Leave a comment

Your email address will not be published. Required fields are marked \*

Comment \*

Name

Email

Website

☐ Save my name, email, and website in this browser for the next time I comment.

**Post Comment**

**Search:**

e.g. whirlpool wrf535swhz

**Search**

[Manuals+](#) | [Upload](#) | [Deep Search](#) | [Privacy Policy](#) | [@manuals.plus](#) | [YouTube](#)

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.