# UM1929
# User manual

## Getting started with the X-CUBE-SPN4 dual brush DC motor driver software expansion for STM32Cube

## Introduction

The X-CUBE-SPN4 is an expansion software package for STM32Cube. The software runs on the STM32 and includes drivers that recognize the L6206 device to provide complete management of control for brush DC motors. The expansion is built on STM32Cube software technology to ease portability across different STM32 microcontrollers. It is compatible with the X-NUCLEO-IHM04A1 STM32 expansion board connected to a NUCLEO-F401RE, NUCLEO-L053R8 or NUCLEO-F334R8 board. The software package includes sample applications for driving one bidirectional brush DC motor or four unidirectional brush DC motors.

For L6206 device operation, please refer to the L6206 datasheet “DS2188: DMOS dual full bridge driver”, available on www.st.com.

# Contents

# List of tables

# List of figures

# 1 Acronyms and abbreviations

**Table 1: Acronyms and abbreviations**

| Acronym | Description |
|---|---|
| API | Application programming interface |
| BSP | Board support package |
| CMSIS | Cortex® microcontroller software interface standard |
| HAL | Hardware abstraction layer |
| SPI | Serial port interface |
| IDE | Integrated development environment |
| LED | Light emitting diode |

# 2 What is STM32Cube?

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

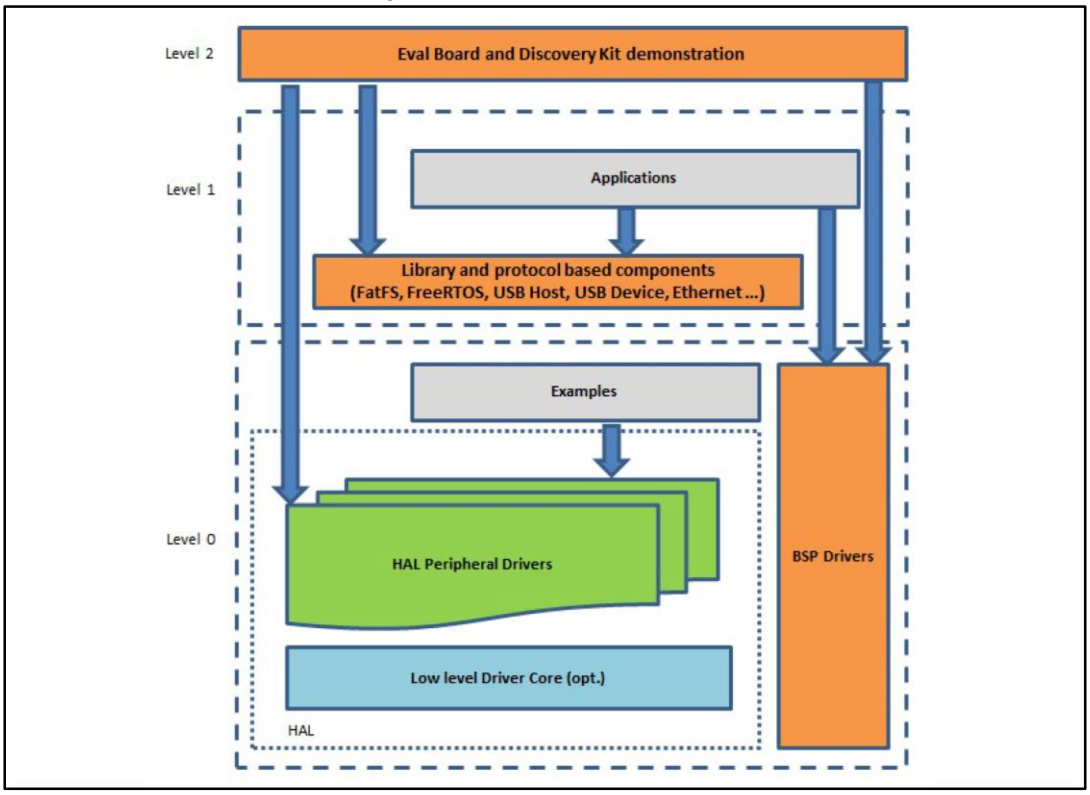
STM32Cube version 1.x includes:

- 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 STM32CubeF4 for the STM32F4 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

## 2.1 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**


Level 2
Eval Board and Discovery Kit demonstration
Level 1
Applications
Library and protocol based components
(FatFS, FreeRTOS, USB Host, USB Device, Ethernet...)
Examples
Level 0
HAL Peripheral Drivers
BSP Drivers
Low level Driver Core (opt.)
HAL

**Level 0**: This level is divided into three sub-layers:

- Board Support Package (BSP): this layer offers a set of APIs relative to the hardware components in the hardware boards (Audio codec, IO expander, Touchscreen, SRAM driver, LCD drivers. etc…); it is based on modular architecture allowing it to be easily

ported on any hardware by just implementing the low level routines. It is composed of two parts:
  - Component: is the driver relative to the external device on the board and not related to the STM32, the component driver provides specific APIs to the external components of the BSP driver, and can be ported on any other board.
  - BSP driver: links the component driver to a specific board and provides a set of easy to use APIs. The API naming convention is BSP_FUNCT_Action(): e.g., BSP_LED_Init(), BSP_LED_On().
- Hardware Abstraction Layer (HAL): this layer provides the low level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries and stacks). It provides generic, multi-instance and function-oriented APIs to help offload user application development time by providing ready to use processes. For example, for the communication peripherals ($I^2C$, UART, etc.) it provides APIs for peripheral initialization and configuration, data transfer management based on polling, interrupt or DMA processes, and communication error management. The HAL Drivers APIs are split in two categories: generic APIs providing common, generic functions to all the STM32 series and extension APIs which provide special, customized functions for a specific family or a specific part number.
- Basic peripheral usage examples: this layer houses the examples built around the STM32 peripherals using the HAL and BSP resources only.

**Level 1**: This level is divided into two sub-layers:

- Middleware components: set of libraries covering USB Host and Device Libraries, STemWin, FreeRTOS, FatFS, LwIP, and PolarSSL. Horizontal interaction among the components in this layer is performed directly by calling the feature APIs, while vertical interaction with low-level drivers is managed by specific callbacks and static macros implemented in the library system call interface. For example, FatFs implements the disk I/O driver to access a microSD drive or USB Mass Storage Class.
- Examples based on the middleware components: each middleware component comes with one or more examples (or applications) showing how to use it. Integration examples that use several middleware components are provided as well.

**Level 2**: This level is a single layer with a global, real-time and graphical demonstration based on the middleware service layer, the low level abstraction layer and basic peripheral usage applications for board-based functions.

# 3 X-CUBE-SPN4 software expansion for STM32Cube

## 3.1 Overview

The X-CUBE-SPN4 software package expands the functionality provided by STM32Cube. The key features of the package are:

- Driver layer for complete management of the L6206 dual DMOS full bridge driver
- Example implementation to control one bidirectional brush DC motor or 4 unidirectional brush DC motors
- Easy portability across different MCU families, thanks to STM32Cube
- Free user-friendly license terms

Once initialization has been performed, the user can modify the driver parameters by calling specific functions to change the bridge paralleling configuration, types and number of motors or the PWM frequency.

The user can also write callback functions and attach them to:

- the flag interrupt handler, depending on the desired actions triggered by an overcurrent or thermal alarm
- the error handler which is called by the library when it reports an error

The user can then drive the different brush DC motors with direction and speed commands. When a motor is requested to run, the corresponding bridge is automatically enabled.

A motion command can be stopped at any moment either by a hard stop which immediately stops the motor or by a hardHiz command which immediately stops the motor and disables the bridge used by the motor.

The library also provides functions to disable or enable the bridges independently of the run or stop commands.

## 3.2 Architecture

This software expansion for STM32Cube fully complies with the STM32Cube architecture and expands it for the development of brush DC motor driver applications based on the L6206 device.

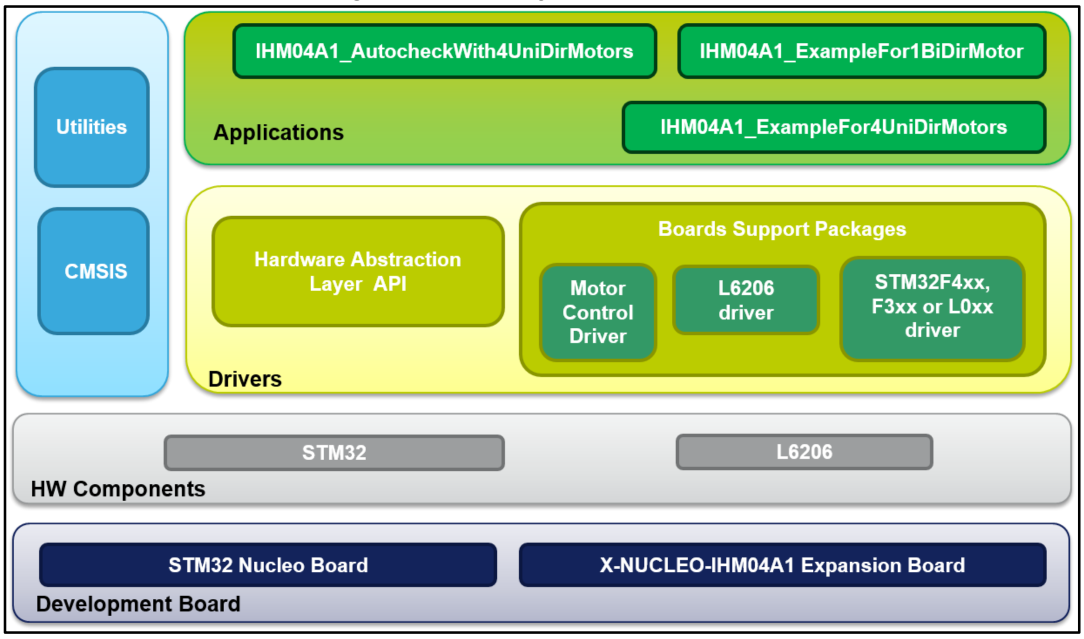
The software is based on the STM32CubeHAL hardware abstraction layer for theSTM32 microcontroller. The package extends STM32Cube by providing a board support package (BSP) for the motor control expansion board and a BSP component driver for the L6206 motor driver.

The software layers used by the application software to access and use the dual full bridge driver expansion board are:

- **STM32Cube HAL layer**: which provides a simple, generic, multi-instance set of APIs to interact with the upper layers (application, libraries and stacks). It is composed of generic and extension APIs. It is directly built around a generic architecture and allows the layers built on it (e.g., the middleware layer) to implement their functions without requiring specific hardware configurations for a given microcontroller unit (MCU). This structure improves library code reusability and guarantees easy portability to other devices.
- Board support package (BSP) layer: provides support for all the STM32 Nucleo board peripherals, except the MCU. It is a limited set of APIs providing a programming interface for certain board specific peripherals like the LED, user button, etc. This

interface also helps in identifying the specific board version. This specific motor control BSP provides the programming interface for various motor driver components. In the X-CUBE-SPN4 software, it is associated with the BSP component for the L6206 dual full bridge driver.

**Figure 2: Overall system architecture**


IHM04A1_AutocheckWith4UniDirMotors
IHM04A1_ExampleFor1BiDirMotor
Utilities
Applications
IHM04A1_ExampleFor4UniDirMotors
Boards Support Packages
Hardware Abstraction Layer API
CMSIS
Motor Control Driver
L6206 driver
STM32F4xx, F3xx or L0xx driver
Drivers
STM32
L6206
HW Components
STM32 Nucleo Board
X-NUCLEO-IHM04A1 Expansion Board
Development Board

## 3.3 Folder structure

- **Drivers**:
  - the required STM32Cube HAL files located in the STM32L0xx_HAL_Driver, STM32F4xx_HAL_Driver and STM32F3xx_HAL_Driver subfolders. These files are not described here as they are not specific to the X-CUBE-SPN4 software but come directly from the STM32Cube framework. Only the HAL files which are required to run the motor driver examples are present.
  - a CMSIS folder with the CMSIS (Cortex® Microcontroller Software Interface Standard) files from ARM. These files are a vendor-independent hardware abstraction layer for the Cortex-M processor series.
  - a BSP (board support package) folder with the codes required for X-NUCLEO-IHM04A1 configuration, the L6206 driver and the motor control API.
- **Project**: several use examples of the L6206 dual full bridge motor driver for different configurations.

### 3.3.1 BSP folder

The following BSPs are used by the X-CUBE-SPN4 software:

#### 3.3.1.1 STM32L0XX-Nucleo, STM32F3XX-Nucleo and STM32F4XX-Nucleo BSPs

Depending on the Nucleo used, these BSPs provide an interface to configure and use the Nucleo peripherals with the X-NUCLEO-IHM04A1 expansion board. Under each STM32L0XX-Nucleo, STM32F3XX-Nucleo and STM32F4XX-Nucleo subfolder, there are two .c/.h file pairs:

- **stm32XXxx_nucleo.c/h**: these files come unmodified from the STM32Cube framework and provide the functions to handle the Nucleo board user button and LEDs.
- **stm32XXxx_nucleo_ihm04a1.c/h**: these files are dedicated to the configuration of the PWMs, GPIOs and interrupt enabling/disabling required for X-NUCLEO-IHM04A1 expansion board operation.

#### 3.3.1.2 Motor control BSP

This BSP provides a common interface to access the driver functions of various motor drivers like the L6206, L6474, Powerstep01, etc. This is done via the MotorControl/motorcontrol.c/h file pair which defines all the functions which can be used to configure and control the motor driver. These functions are then mapped to the functions of the motor driver component used on the expansion board via the `motorDrv_t` structure file (defined in Components\Common\motor.h). This structure defines a list of function pointers which are filled during its instantiation in the corresponding motor driver component. If using X-CUBE-SPN4, the instance of the structure is called l6206Drv (see file: BSP\Components\l6206\l6206.c).

As the motor control BSP is common for all motor driver expansion boards, not all of its functions are available for a given expansion board. In this case, during the instantiation of the `motorDrv_t` structure in the driver component, the unavailable functions are replaced by a null pointer.

#### 3.3.1.3 L6206 BSP component

The L6206 BSP component provides the driver functions of the L6206 dual full bridge driver in the stm32_cube\Drivers\BSP\Components\l6206 folder. This folder has the following files:

- l6206.c: core functions of the L6206 driver.
- l6206.h: declaration of the L6206 driver functions and their associated definitions.
- l6206_target_config.h: predefines values for the L6206 parameters (bridge paralleling configuration, type and number of brush DC motors, PWM frequency of the bridge inputs).

### 3.3.2 Projects folder

For each Nucleo platform, two sample projects are available in the stm32_cube\Projects\Multi\Examples\MotionControl\ folder:

- **IHM04A1_ExampleFor1BiDirMotor**: examples of control function use for single bidirectional brush DC motor driving with paralleling of bridge input 1A with input 2A, and bridge input 1B with input 2B.
- **IHM04A1_ExampleFor4UniDirMotors**: examples of control function use for four unidirectional brush DC motor driving with no paralleling of the bridge inputs.

Each example has a folder dedicated to the targeted IDE:

- EWARM, where the project files for IAR are located.
- MDK-ARM, where the project files for Keil are located.
- SW4STM32, where the project files for AC6 OpenSTM32 are located.

Each example also has the following code files:

- inc\main.h: Main header file.
- inc\ stm32xxxx_hal_conf.h: HAL configuration file.
- inc\stm32xxxx_it.h: header for the interrupt handler.

- src\main.c: main program (code of the example which is based on the motor control library for l6206).
- src\stm32xxxx_hal_msp.c: HAL initialization routines.
- src\stm32xxxx_it.c: interrupt handler.
- src\system_stm32xxxx.c: system initialization.
- src\clock_xx.c: clock initialization.

## 3.4 Required software resources

Communication between the L6206 and the MCU is accomplished through GPIOs. It requires the use of:

- one common GPIO for the flag interrupt (overcurrent detection and thermal protection) and the enable pin for bridge A.
- one common GPIO for the flag interrupt (overcurrent detection and thermal protection) and the enable pin for bridge B.
- 4 GPIOs used to generate PWM for each bridge input (1A, 2A, 1B, 2B).

**Table 2: Required resources for the X-CUBE-SPN4 software**

| Resources F3xx | Resources F4xx | Resources L0xx | Pin | Features |
|---|---|---|---|---|
| Ext Line 10<br>GPIO PA10 | | | D2 | Flag interrupt and enable pin of bridge A |
| Ext Line 1<br>GPIO PC1 | | | A4 | Flag interrupt and enable pin of bridge B |
| Timer 3 Ch1<br>GPIO PB4 | | Timer 22 Ch1<br>GPIO PB4 | D5 | PWM for IN1A |
| Timer 3 Ch2<br>GPIO PB5 | | Timer 22 Ch2<br>GPIO PB5 | D4 | PWM for IN2A |
| Timer 2 Ch1<br>GPIO PA0 | | | A0 | PWM for IN1B |
| Timer 2 Ch2<br>GPIO PA1 | | | A1 | PWM for IN2B |

## 3.5 APIs

The API of the X-CUBE-SPN4 software is defined in the Motor Control BSP. Its functions are prefixed by `BSP_MotorControl_`.

Not all the functions of this module are available for the L6206 and hence for the expansion board X-NUCLEO-IHM04A1.

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

## 3.6 Sample application description

Two sample applications using the X-NUCLEO-IHM04A1 STM32 expansion board with either the NUCLEO-F401RE, NUCLEO-F334R8 or NUCLEO-L053R8 board is provided in the "Projects" directory.

Ready-to-build projects are available for multiple IDEs (see *Section 5.3.2: "Projects folder"*).

# 4 System setup guide

## 4.1 Hardware description

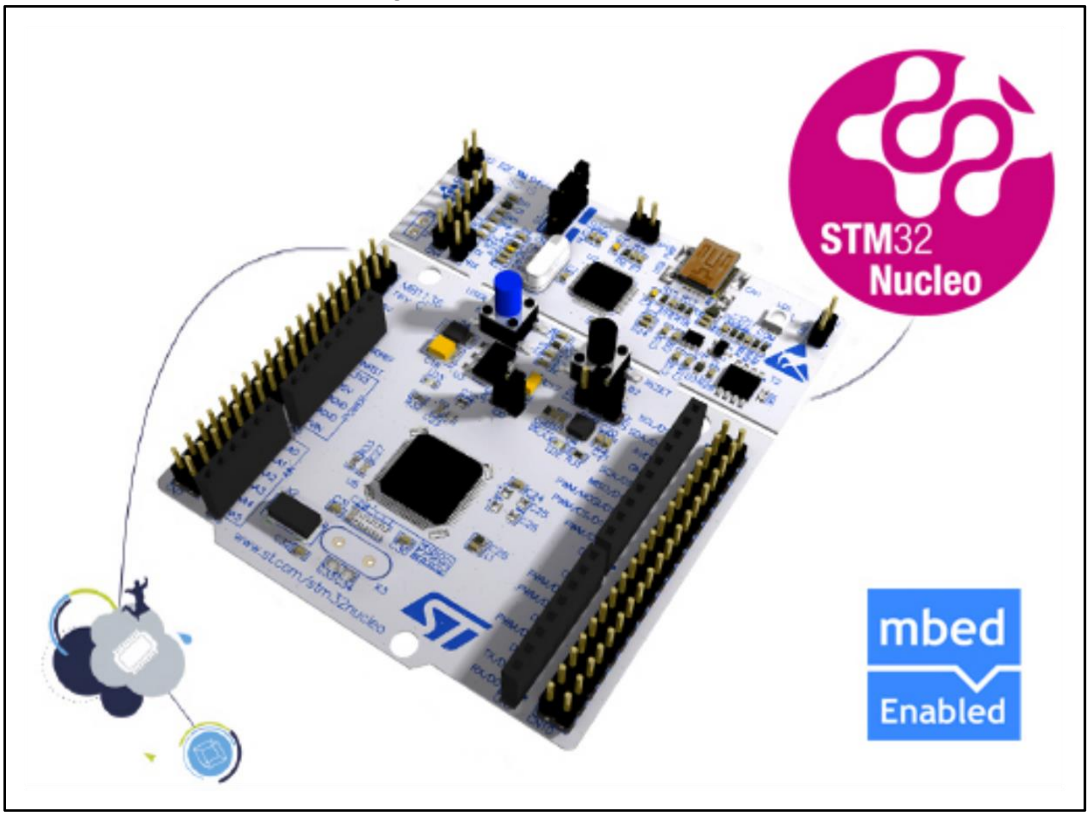
This section describes the hardware required to run the XCUBE-SPN4 software and successfully drive one or several brush DC motors.

### 4.1.1 STM32 Nucleo platform

The STM32 Nucleo boards provide an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller lines. The Arduino™ connectivity support and ST morpho headers 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 any separate probe 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.

Information regarding the STM32 Nucleo board is available on www.st.com at http://www.st.com/stm32nucleo

**Figure 3: STM32 Nucleo board**

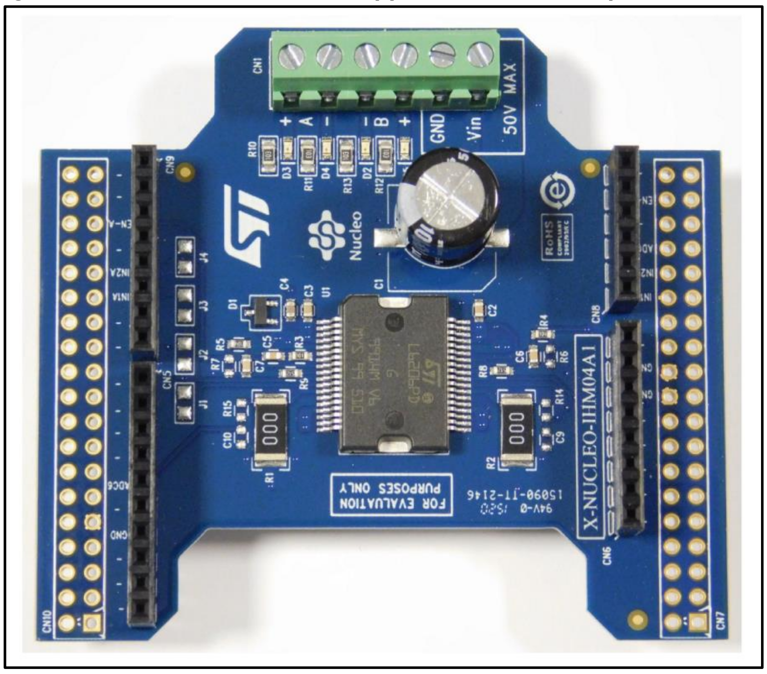


### 4.1.2 X-NUCLEO-IHM04A1 dual brush DC motor driver expansion board

The X-NUCLEO-IHM04A1 is a dual brush DC motor driver expansion board based on the L6206 for STM32 Nucleo. It provides an affordable and easy-to-use solution for driving brush DC motors in your STM32 Nucleo project.

The X-NUCLEO-IHM04A1 is compatible with the Arduino UNO R3 connector, and supports the addition of other expansion boards which can be stacked with a single STM32 Nucleo board.

**Figure 4: X-NUCLEO-IHM01A1 stepper motor driver expansion board**



Information about the X-NUCLEO-IHM04A1 expansion board is available on www.st.com at *http://www.st.com/x-nucleo*.

### 4.1.3 Miscellaneous hardware components

To complete the hardware setup, you will need:

- One to four brush DC motors
- an external DC power supply with two electrical cables
- a USB cable type A to mini-B to connect the STM32 Nucleo to a PC

## 4.2 Software description

The following software components are required to set up a suitable development environment for creating applications based on the dual full bridge driver expansion board:

- The X-CUBE-SPN4 software expansion for STM32Cube dedicated to L6206 dual full bridge driver application development. The X-CUBE-SPN4 firmware and related documentation is available on www.st.com.
- A development toolchain and compiler; the following toolchains are supported:
  - Keil RealView Microcontroller Development Kit (MDK-ARM) toolchain V5.12
  - IAR Embedded Workbench for ARM (EWARM) toolchain V7.20
  - AC6 OpenSTM32 System Workbench for STM32 (SW4STM32)

## 4.3 Hardware and software setup

This section describes the hardware and software setup procedure to run the examples and develop new applications based on the motor driver expansion board.

### 4.3.1 Common setup for all configurations

The STM32 Nucleo must be configured with the following jumper positions:

- JP1 off
- JP5 (PWR) on UV5 side
- JP6 (IDD) on

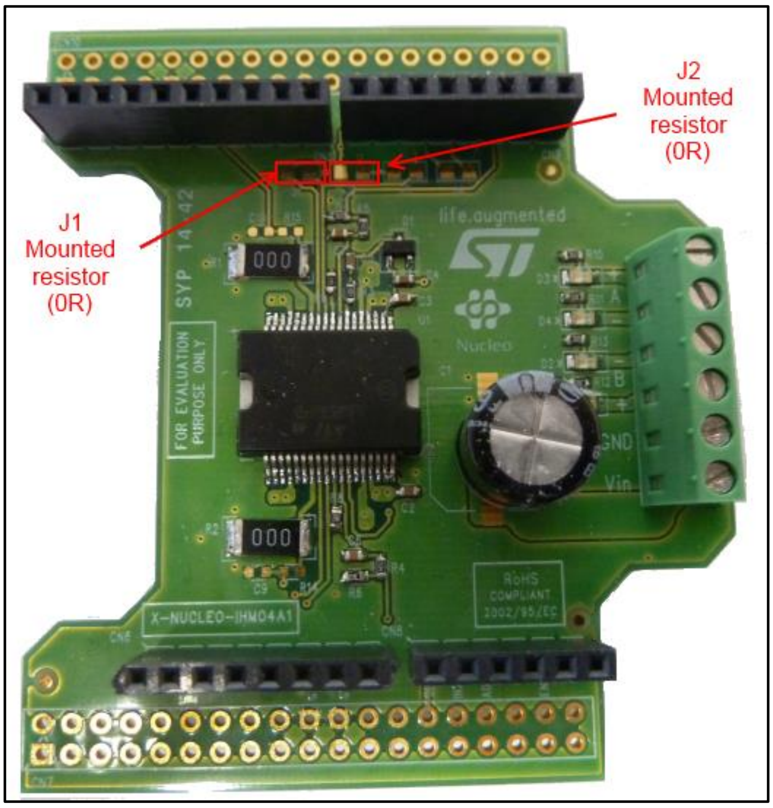
### 4.3.2 Setup to drive 1 bidirectional brush DC motor

The configuration for this particular scenario is bridge input 1A paralleled with 2A and bridge 1B paralleled with 2B, but other bridge configurations are possible.

The X-NUCLEO-IHM04A1 STM32 expansion board must have:

- A mounted resistor (0R) on J1 to put bridge input 1A in parallel with input 2A
- A mounted resistor (0R) on J2 to put bridge input 1B in parallel with input 2B
- No resistors mounted on J3 and J4

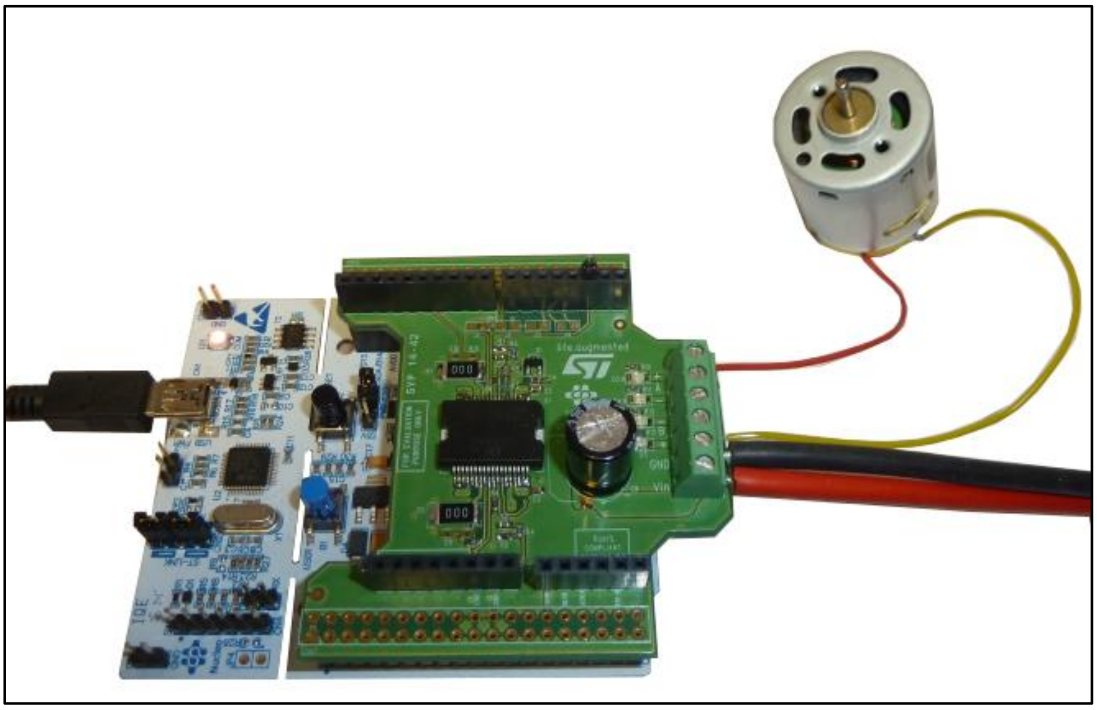
**Figure 5: X-NUCLEO-IHM04A1 STM32 expansion board configuration to drive 1 brush DC motor**



Once the boards are properly configured:

1. Plug the X-NUCLEO-IHM04A1 expansion board on top of the STM32 Nucleo by using the Arduino UNO connectors.
2. Connect the STM32 Nucleo board to a PC with the USB cable through USB connector CN1 to power the board.
3. Connect one lead of the brush DC motor to the X-NUCLEO-IHM04A1 bridge output connector A+ and the other one to B+.
4. Power the X-NUCLEO-IHM04A1 expansion board by connecting its connectors Vin and Gnd to the DC power supply. The DC supply must be set to deliver the voltage required by the brush DC motor.

**Figure 6: Board connections to drive 1 bidirectional brush DC motor**



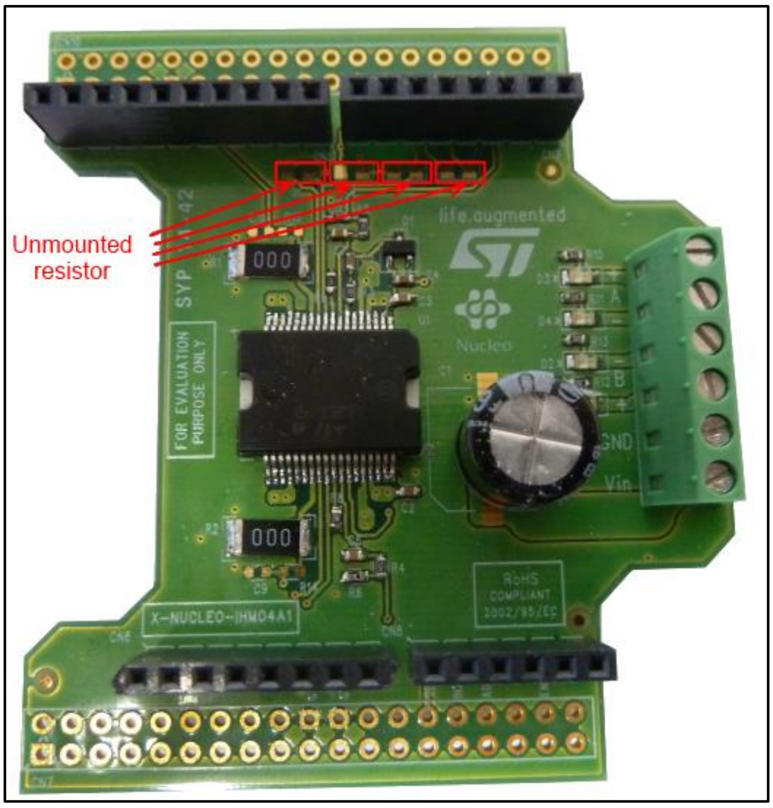
Once the system setup is ready:

1. Open your preferred toolchain (MDK-ARM from Keil, EWARM from IAR, or OpenSTM32).
2. depending on the STM32 Nucleo board used, open the software project from:
   - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM04A1_ExampleFor1BiDirMotor\YourToolChainName\STM32F334R8-Nucleo for Nucleo STM32F334
   - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM04A1_ExampleFor1BiDirMo tor\YourToolChainName\STM32F401RE-Nucleo for Nucleo STM32F401
   - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM04A1_ExampleFor1BiDirMo tor\YourToolChainName\STM32L053R8-Nucleo for Nucleo STM32L053.
3. In order to adapt the default parameters used by the L6206 to your stepper motor characteristics, either:
   - use the `BSP_MotorControl_Init` function with the NULL pointer and open stm32_cube\Drivers\BSP\Components\l6206\l6206_target_config.h to modify the parameters accordingly, or
   - use `BSP_MotorControl_Init` function with the address of the `initDevicesParameters` structure with values modified accordingly.
4. Rebuild all files and load your image into target memory.
5. Run the example. The motor will start when the user button is pressed (see main.c for the detailed demo sequence). Each time the user button is pressed, the step of the demo sequence is changed.

### 4.3.3 Setup to drive 4 unidirectional brush DC motors

The X-NUCLEO-IHM04A1 expansion board for the first motor must not have resistors mounted on J1, J2, J3 and J4 as bridge paralleling is not used.

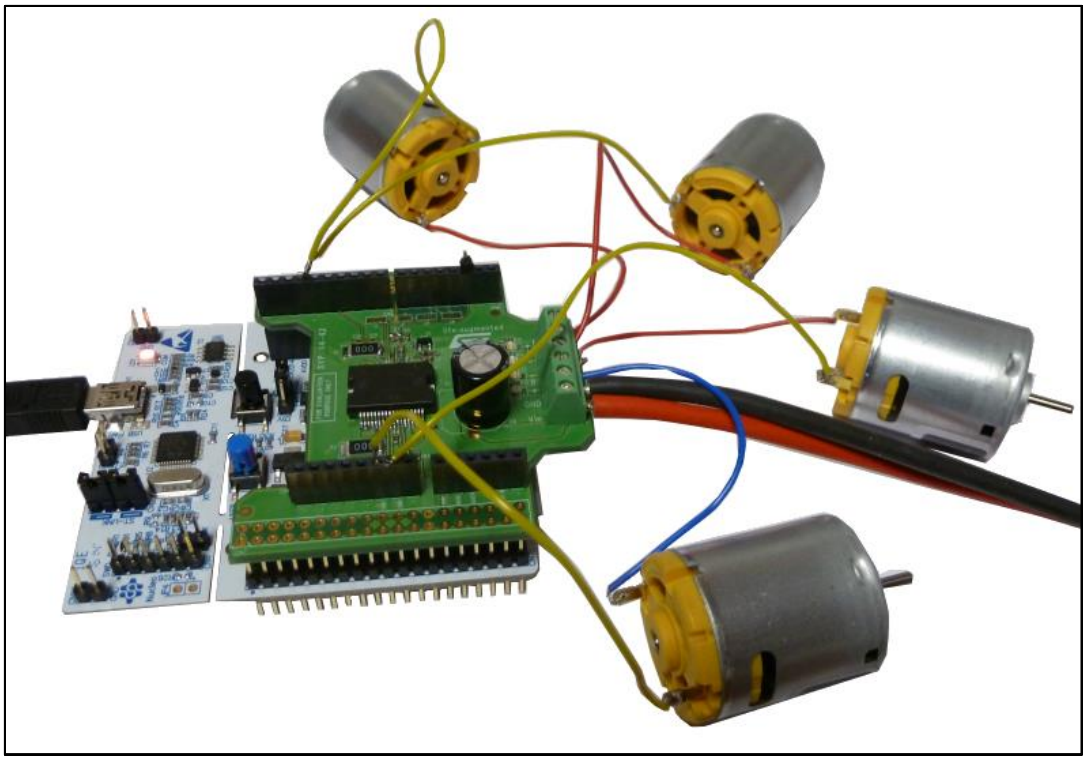
**Figure 7: X-NUCLEO-IHM04A1 STM32 expansion board configuration to drive 4 unidirectional brush DC motors**



Once the boards are properly configured:

1. Plug the X-NUCLEO-IHM04A1 expansion board on top of the STM32 Nucleo by using the Arduino UNO connectors.
2. Connect the STM32 Nucleo board to a PC with the USB cable through USB connector CN1 to power the board.
3. Connect one lead of each brush DC motor to one of the X-NUCLEO-IHM04A1 bridge outputs (A+, A-, B+ or B-) and the other one to Gnd.
4. Power-on the X-NUCLEO-IHM04A1 expansion board by connecting its connectors Vin and Gnd to the DC power supply. The DC supply must be set to deliver the required voltage by the brush DC motors.

**Figure 8: Board connections to drive 4 brush DC motors**



Once the system setup is ready:

1. Open your preferred toolchain (MDK-ARM from Keil, EWARM from IAR, or OpenSTM32).
2. depending on the STM32 Nucleo board used, open the software project from:

- /stm32_cube/Projects/Multi/Examples/MotionControl/IHM04A1_ExampleFor4UniDirMotors/YourToolChainName/STM32F334R8-Nucleo for Nucleo STM32F334.
- \stm32_cube\Projects\Multi\Examples\MotionControl\IHM04A1_ExampleFor4UniDirMotors\YourToolChainName\STM32F401RE-Nucleo for Nucleo STM32F401.
- \stm32_cube\Projects\Multi\Examples\MotionControl\IHM04A1_ExampleFor4UniDirMotors\YourToolChainName\STM32L053R8-Nucleo for Nucleo STM32L053.

1. In order to adapt the default parameters used by the L6206 to your stepper motor characteristics, either:
   – use the `BSP_MotorControl_Init` function with the NULL pointer and open stm32_cube\Drivers\BSP\Components\l6206\l6206_target_config.h to modify the parameters accordingly, or
   – use `BSP_MotorControl_Init` function with the address of the `initDevicesParameters` structure with values modified accordingly.
2. Rebuild all files and load your image into target memory.
3. Run the example. The motors will start when the user button is pressed (see main.c for the detailed demo sequence). Each time the user button is pressed, the step of the demo sequence is changed.

# 5 Revision history

Table 3: Document revision history

| Date | Version | Changes |
|---|---|---|
| 06-Aug-2015 | 1 | Initial release. |
| 06-Jun-2016 | 2 | Text and formatting changes throughout document<br>Added STM32F3xx to list of compatible Nucleo boards<br>Updated *Section 5.5: "APIs"* - API descriptions are now in an HTML file in the software Documentation folder<br>Replaced references to TrueStudio with System Workbench for STM32 (SW4STM32) |