UM2110
User manual

Getting started with the X-CUBE-SPN12, low voltage dual brush DC motor driver software expansion for STM32Cube

Introduction

The X-CUBE-SPN12 is an expansion software package for STM32Cube. The software runs on the STM32 Nucleo providing management of STSPIN240 to control low voltage dual brush DC motors.

The expansion is built on STM32Cube software technology to ease portability across different STM32 microcontrollers.

It is compatible with the NUCLEO-F401RE, NUCLEO-F334R8, NUCLEO-F030R8 or NUCLEO-L053R8 boards connected to an X-NUCLEO-IHM12A1 expansion board.

The software comes with a sample implementation driving two bidirectional low voltage dual brush DC motors.

Contents

List of tables

List of figures

1 Acronyms and abbreviations

Table 1: List of acronyms

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.

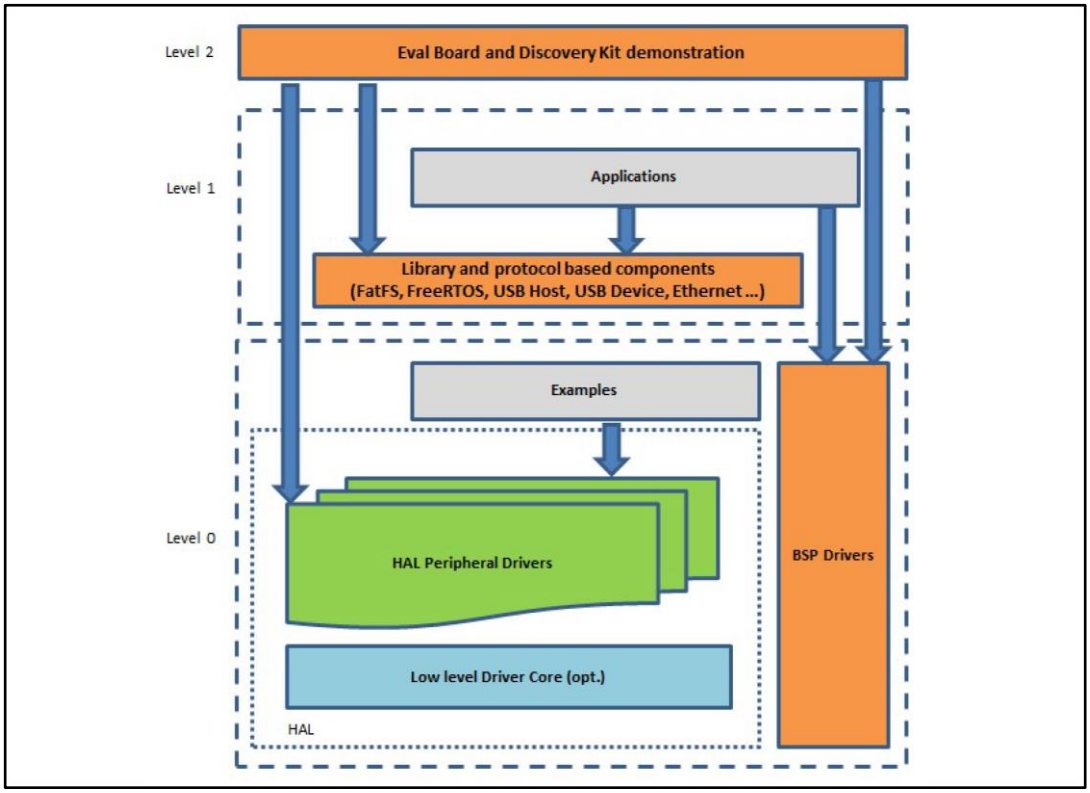
STM32Cube version 1.x includes:

- 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 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 (I²C, 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-SPN12 software expansion for STM32Cube

3.1 Overview

The X-CUBE-SPN12 software package expands STM32Cube functionality, and features:

- STSPIN240 configuration (bridge input and enabling signals)
- flag interrupt handling (overcurrent and thermal alarm reporting)
- handling of up to two bidirectional dual brush DC motors
- STM32 Nucleo and expansion board configuration (GPIOs, PWMs, IRQs, etc.)

To use the STSPIN240 driver library, first call its initialization function to:

- set up the required GPIOs to handle the bridge enabling pins and the FLAG interupt which reports overcurrent detection or thermal protection
- set up the drivers
- load the driver parameters with the values in "stspin240_250_target_config.h", in order to program the PWM frequency of the bridge inputs and the number of brush DC motors .

You can easily modify the driver parameters through specific functions able to change the bridge configuration, the number of motors or the PWM frequency.

You can also use callback functions with:

- the flag interrupt handler, when an overcurrent or a thermal alarm occur;
- the error handler, called by the library when it reports an error.

Then, you can drive different brush DC motors by setting a specified running direction and by changing the maximum speed. When a motor is requested to run, the related bridge is automatically enabled.

A motion command can be stopped at any moment:

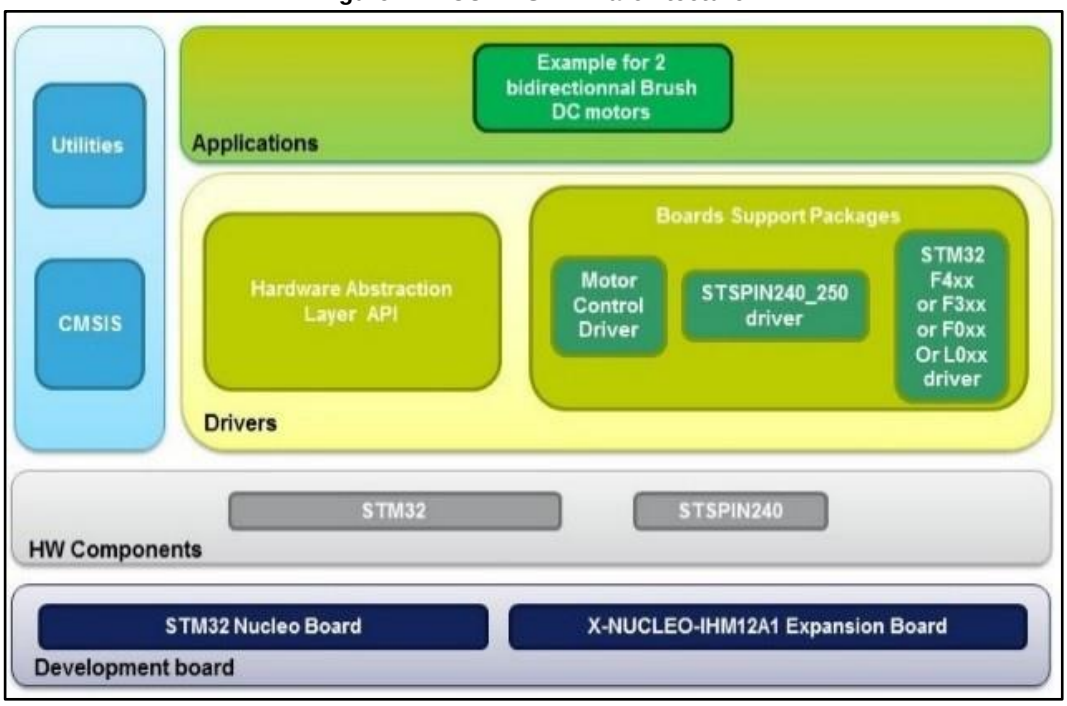
- by a hard stop which immediately stops the motor
- or by a hardHiz command which immediately stops the motor and disables the bridge it uses.

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

3.2 Architecture

This fully compliant STM32Cube software expansion enables development of applications using low voltage dual brush DC motor drivers.

Figure 2: X-CUBE-SPN12 architecture



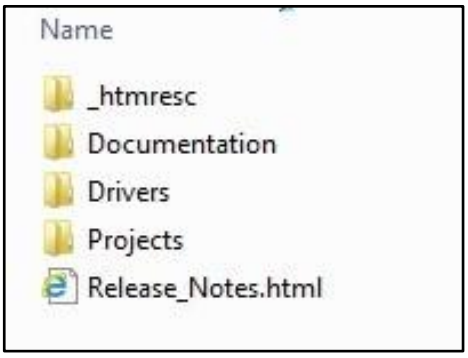
The software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller and extends STM32Cube with a board support package (BSP) for the low voltage dual brush DC motor driver expansion board (X-NUCLEO-IHM12A1) and a BSP component driver for STSPIN240 motor driver.

The software layers used by the application software to access and use the dual brush DC motor driver expansion board are:

- STM32Cube HAL layer: provides a generic, multi-instance set of simple APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). These generic and extension APIs are based on a common architecture and the layers above them like the middleware layer can function without requiring specific hardware configuration data for a given microcontroller unit (MCU). This structure improves the library code reusability and guarantees easy portability across other devices.
- Board support package (BSP) layer: supports the peripherals on the STM32 Nucleo board, except for the MCU. This limited set of APIs provides a programming interface for certain board specific peripherals like the user button, the LED, etc, and helps in identifying the specific board version. In case of motor control expansion boards, the motor control BSP provides a programming interface for various motor driver components. The BSP in the X-CUBE-SPN12 software provides the drivers to manage the STSPIN240 motor driver.

3.3 Folder structure

Figure 3: X-CUBE-SPN112 package folder structure



The software is packaged in the following main folders:

- **Drivers**:
  - STM32Cube HAL driver files located in the STM32F4xx_HAL_Driver, STM32F3xx_HAL_Driver, STM32F0xx_HAL_Driver or STM32L0xx_HAL_Driver subdirectories. These files are not described here as they are not specific for the X-CUBE-SPN12 software but derive directly from the STM32Cube framework.
  - CMSIS folder with the CMSIS (Cortex® microcontroller software interface standard) files from ARM. These files form a vendor-independent hardware abstraction layer for the Cortex-M processor series. This folder is also unchanged from the STM32Cube framework.
  - BSP folder with code files required for X-NUCLEO-IHM12A1 configuration, the STSPIN240 driver and the motor control API. (see *Section 3.3.1: "BSP folder"*)
- **Projects**: contains a sample STSPIN240 application using two bidirectional brush DC motors

3.3.1 BSP folder

3.3.1.1 STM32F4XX-Nucleo/STM32F3XX-Nucleo/STM32F0XX-Nucleo/STM32L0XX-Nucleo BSPs

Depending on the STM32 Nucleo development board, these BSPs provide an interface to configure and use the development board peripherals with the X-NUCLEO-IHM12A1 expansion board.

Each subfolder (STM32F4XX-Nucleo/STM32F3XX-Nucleo/STM32F0XX-Nucleo/STM32L0XX-Nucleo) contains two couples of .c/.h files:

- **stm32XXxx_nucleo.c/h**: these files derive from the STM32Cube framework (with no modification) and provide the functions to handle the related STM32 Nucleo board user button and LEDs.
- **stm32XXxx_nucleo_IHM12a1.c/h**: these files are dedicated to the configuration of the PWMs, the GPIOs and the interrupt enabling/disabling.

3.3.1.2 Motor control BSP

This BSP provides a common interface to access the driver functions of various motor drivers like L6206, L6474, powerSTEP01, STSPIN240, etc. This is done via a couple of c/h files: MotorControl/motorcontrol.c/h, which defines all the functions to configure and control the motor driver. These functions are then mapped to the functions of the motor driver component used on the given expansion board via the structure file: motorDrv_t (defined in Components\Common\motor.h).

This structure defines a list of function pointers filled during its instantiation in the corresponding motor driver component.

For X-CUBE-SPN12, the structure instance is called stspin240_250Drv (see file BSP\Components\stspin240_250\stspin240_250.c)

As the motor control BSP is common for all motor driver expansion boards, some functions are not available for all expansion boards. 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 Stspin240_250 BSP component

The stspin240_250 BSP component provides the driver functions of the STSPIN240 low voltage dual brush DC motor driver in the folder stm32_cube\Drivers\BSP\Components\stspin240_250, which contains:

- **stspin240_250.c:** Stspin240_250 driver core functions
- **stspin240_250.h:** declaration of the Stspin240_250 driver functions and their associated definitions
- **stspin240_250_target_config.h:** parameter value setup for the STSPIN240 (bridge configuration, number of brush DC motors, bridge input PWMs frequency)

When used with an STSPIN240 driver as in the case of the X-NUCLEO-IHM12A1 expansion board, this component requires the compilation flag declaration: STSPIN_240.

3.3.2 Project folder

For each STM32 Nucleo board, the example projects are in the folder stm32_cube\Projects\Multi\Examples\MotionControl\: IHM12A1_ExampleFor2BiDirMotors (examples of control functions for two bidirectional brush DC motor driving).

There is a dedicated folder for the target IDE:

- **EWARM** containing the project files for IAR
- **MDK-ARM** containing the project files for Keil
- **SW4STM32** containing the project files for OpenSTM32

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 Stspin240)
- **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 Software required resources

STSPIN240 and the MCU communicate through GPIOs:

- 1 common GPIO for the flag interrupt (overcurrent detection or overtemperature protection) and the enable pin
- 2 GPIOs to generate a PWM for each bridge input (PWMA and PWMB)
- 2 GPIOs for the bridge input phase and to set the motor direction (PHA/DIR_A and PHB/DIR_B)

- 1 GPIO to generate a PWM for REF level setup
- 1 GPIO to set/reset the reset pin

Table 2: Required resources for the X-CUBE-SPN12 software

Resources for F4xx/F3xx	Resources for L0xx	Resources for F0xx	Pin	Features
ext. line 10 GPIO PA10			D2	flag interrupt and enable pin
GPIO PB5 TIM3 CH2		GPIO PB5 TIM22 CH2	D4	PWM for B PWMB brige
GPIO PB4 TIM3 CH1		GPIO PB4 TIM22 CH1	D5	PWM for A PWMA brigde
GPIO PB10			D6	direction for A PHA/DIR_A bridge
GPIO PA8			D7	Direction for B PHB/DIR_B bridge
GPIO PA0 TIM2 CH1	GPIO PA9 TIM1 CH2	GPIO PA0 TIM2 CH1	A0 (or D8 for F0)	REF
GPIO PC7			D9	reset

3.5 APIs

Detailed function and parameter descriptions for the user-APIs are compiled in an HTML file in the software package **Documentation** folder.

X-CUBE-SPN12 software API is defined in the BSP motor control (functions predefined through BSP_MotorControl_).

Not all the functions of this module are available for the STSPIN240 and, consequently, for the X-NUCLEO-IHM12A1 expansion board.

4 System setup guide

4.1 Hardware description

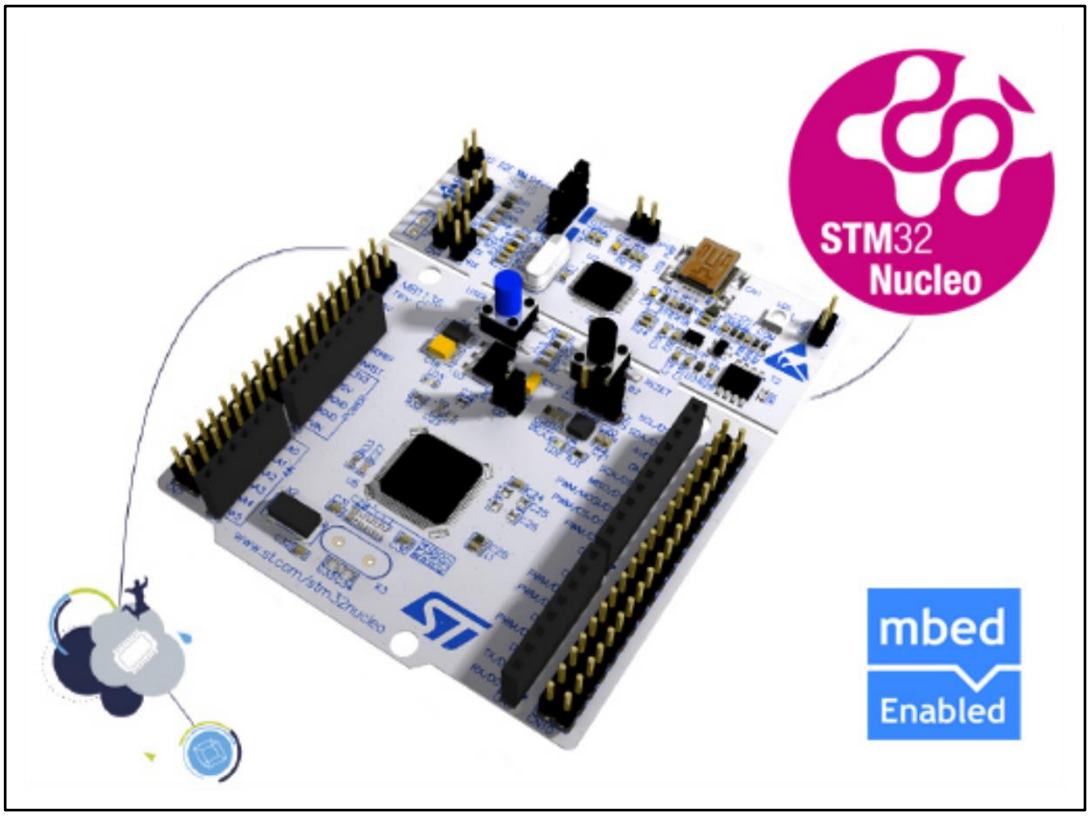
This section describes the hardware components which are required to execute the X-CUBE-SPN12 software and successfully drive one or two low voltage dual brush DC motors.

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 4: STM32 Nucleo board



4.1.2 X-NUCLEO-IHM12A1 low voltage dual brush DC motor driver expansion board

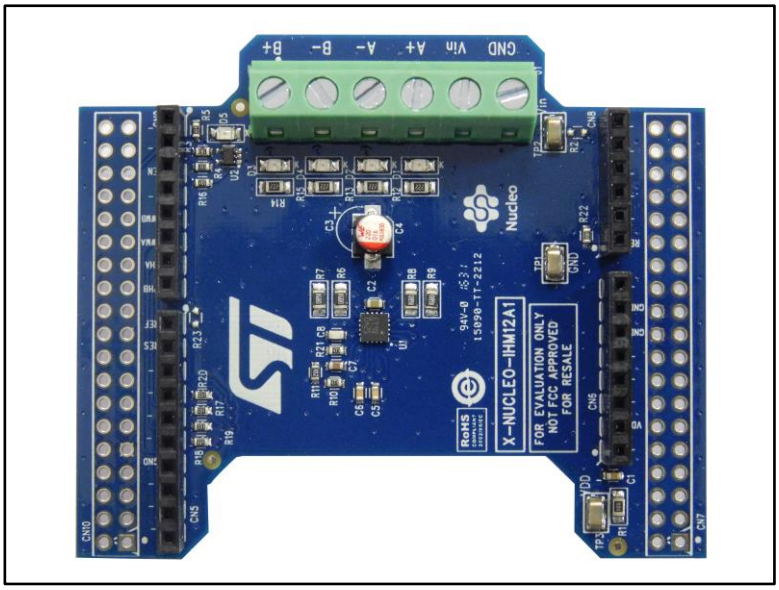
The X-NUCLEO-IHM12A1 is a low voltage dual brush DC motor driver expansion board based on the STSPIN240 for STM32 Nucleo.

It provides an affordable and easy-to-use solution for the implementation of portable motor driving applications such as thermal printers, robotics and toys.

Thanks to its programmable current limiter and complete set of protection features, it offers high levels of performance and robustness.

The X-NUCLEO-IHM12A1 is compatible with the Arduino UNO R3 connector and most STM32 Nucleo boards.

Figure 5: X-NUCLEO-IHM12A1 expansion board



Information about the X-NUCLEO-IHM12A1 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 need:

- one or two low voltage dual brush DC motor
- an external DC power supply with two electric cables (up to 10 V)
- a USB type A to mini-B USB cable to connect the STM32 Nucleo to a PC
- a Windows™ (v. 7 and above) – laptop or PC

4.2 Software description

The following software components are needed for a suitable development environment for applications based on the motor driver expansion board:

- ST-LINK/V2-1 USB driver
- ST-LINK/V2-1 firmware upgrade
- X-CUBE-SPN12 expansion for STM32Cube dedicated to STSPIN240 low voltage dual brush motor driver application development. The X-CUBE-SPN12 firmware and related documentation are available on www.st.com.
- One of the following development tool-chain and compilers:
  - Keil RealView Microcontroller Development Kit (MDK-ARM) toolchain V5.12
  - IAR Embedded Workbench for ARM (EWARM) toolchain V7.20
  - OpenSTM32 System Workbench for STM32 (SW4STM32)

4.3 Hardware and software setup

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

4.3.1 Common setup for all configurations

The STM32 Nucleo board has to be configured with the following jumpers position:

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

4.3.2 REF pin setup on X-NUCLEO-IHM12A1 expansion board

Depending on the nucleo board, the REF pin setup has to be adapted to the X-NUCLEO-IHM12A1 expansion board.

With NUCLEO-F401RE, NUCLEO-F334R8 or NUCLEO-L053R8, the default configuration of the board is: R22 (200 kΩ) mounted and R23 not mounted.

With NUCLEO-F030R8, R23 (200 kΩ) has to be mounted and R22 not mounted. If you want to keep the default board configuration, you can simply put a wire between the Arduino UNO R3 CN5-1 and CN8-1 connector pins.

4.3.3 Setup to drive two bidirectional brush DC motors

1. Plug the X-NUCLEO-IHM12A1 expansion board on top of the STM32 Nucleo board via the Arduino UNO R3 connectors.
2. Connect the STM32 Nucleo board to a PC with the USB cable through USB connector CN1 to power the board
3. Connect the leads of the first dual brush DC motor to the X-NUCLEO-IHM12A1 bridge output connector A+/A- and the second to connectors B+/B-.
4. Power on the X-NUCLEO-IHM12A1 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 to the three-phase brushless motor.
5. Open your preferred toolchain (MDK-ARM from Keil, EWARM from IAR, or SW4STM32 from OpenSTM32)

6. Depending on the used STM32 Nucleo board, open the software project from:
   - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM12A1_ExampleFor2BiDirMotors*YourToolChainName*\STM32F401RE-Nucleo for NUCLEO-F401RE
   - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM12A1_ExampleFor2BiDirMotors*YourToolChainName*\STM32F334R8-Nucleo for NUCLEO-F334R8
   - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM12A1_ExampleFor2BiDirMotors*YourToolChainName*\STM32F030R8-Nucleo for NUCLEO-F030R8
   - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM12A1_ExampleFor2BiDirMotors*YourToolChainName*\STM32L053R8-Nucleo for NUCLEO-L053R8
7. Adapt the default parameters used by the STSPIN240 to your motor characteristics by modifying the parameters in stm32_cube\Drivers\BSP\Components\stspin240_250\stspin240_250_target_config. h.
8. Rebuild all files and load your image into target memory.
9. Run the sample application.
10. Push the user button to start the motor.
11. Open main.c to watch the detailed demo sequence. Each time you press the user button, a different demo sequence step appears.

5 Revision history

Table 3: Document revision history

Date	Version	Changes
04-Oct-2016	1	Initial release.

IMPORTANT NOTICE – PLEASE 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 acknowledgement.

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. 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.