UM2300
User manual

Getting started with X-CUBE-SPN14 stepper motor driver software expansion for STM32Cube

Introduction

The X-CUBE-SPN14 expansion package for STM32Cube gives you full control of stepper motor operations.

When combined with one or more X-NUCLEO-IHM14A1 expansion boards, this software allows a compatible STM32 Nucleo board to control one or more stepper motors.

It is built on top of STM32Cube software technology for easy portability across different STM32 microcontrollers.

The software comes with a sample implementation for one stepper motor. It is compatible with STM32 NUCLEO-F401RE, NUCLEO-F334R8, NUCLEO-F030R8 or NUCLEO-L053R8 boards with an X-NUCLEO-IHM14A1 expansion board mounted on top.

Contents

1 Acronyms and abbreviations 5
2 What is STM32Cube? 6
2.1 STM32Cube architecture 6
3 X-CUBE-SPN14 software expansion for STM32Cube 8
3.1 Overview 8
3.2 Architecture 9
3.3 Folder structure 10
3.3.1 BSP folder 11
3.3.2 Projects folder 12
3.4 Software required resources 12
3.5 APIs 13
3.6 Sample application description 13
4 System setup guide 14
4.1 Hardware description 14
4.1.1 STM32 Nucleo platform 14
4.1.2 X-NUCLEO-IHM14A1 stepper motor driver expansion board 15
4.1.3 Miscellaneous hardware components 15
4.2 Software description 15
4.3 Hardware and software setup 16
4.3.1 Setup to drive a single motor 16
5 Revision history 18

List of tables

Table 1: List of acronyms....5
Table 2: Required resources for the X-CUBE-SPN14 software....12
Table 3: Document revision history....18

List of figures

Figure 1: Firmware architecture 6
Figure 2: X-CUBE-SPN14 software architecture 10
Figure 3: Folder structure 10
Figure 4: STM32 Nucleo board 14
Figure 5: X-NUCLEO-IHM14A1 stepper motor driver expansion board 15
Figure 6: Board connections 16

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
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.

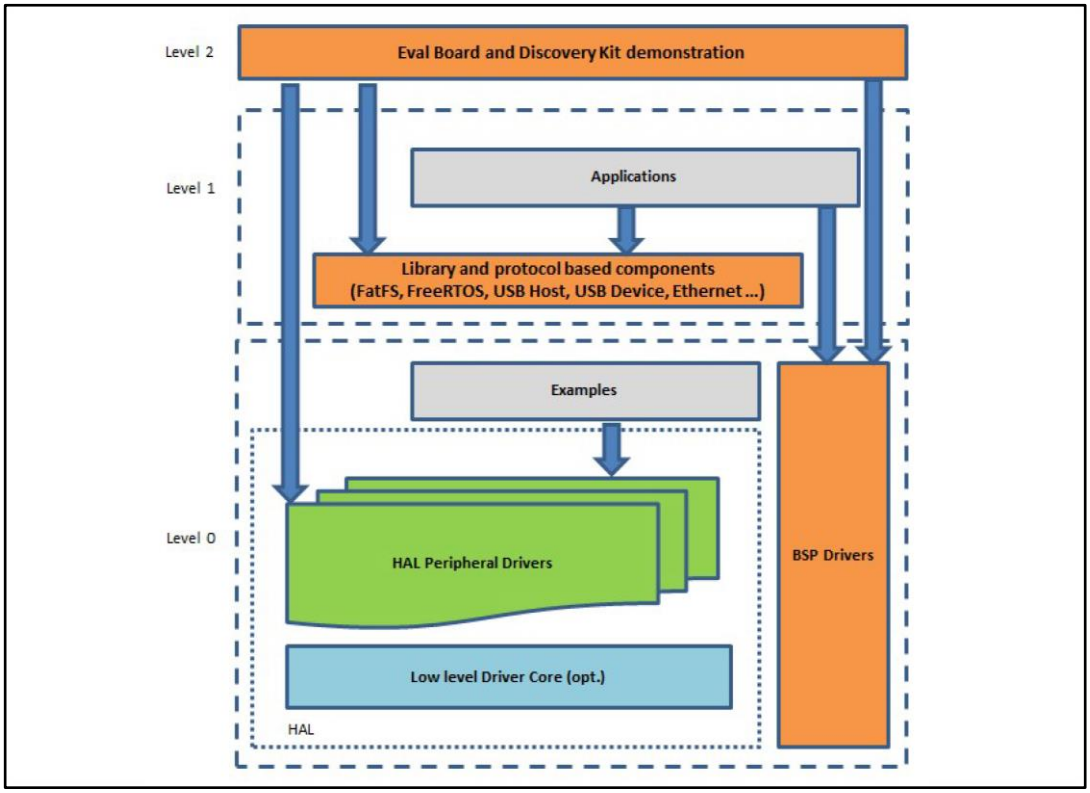
STM32Cube version 1.x includes:

- 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-SPN14 software expansion for STM32Cube

3.1 Overview

The X-CUBE-SPN14 software package expands the functionality of STM32Cube. Its key features include:

- A driver layer for complete management of the STSPIN820 (low power stepper motor driver) device integrated in the X-NUCLEO-IHM14A1 expansion board
- Read and write of the device parameters, GPIO, PWM and IRQ configuration, micro-stepping, direction position, speed, acceleration, deceleration and torque controls, automatic full-step switch management, high impedance or hold stop mode selection, enable and stand-by management
- Fault interrupt handling
- Single stepper motor control sample application
- Easy portability across different MCU families, thanks to STM32Cube
- Free, user-friendly license terms

The software implements pseudo registers and motion commands by:

- configuring timers used to generate step clock and voltage reference
- managing device parameters like acceleration, deceleration, min. and max. speed, positions at speed profile boundaries, mark position, micro-stepping mode, direction, motion state, etc.

The software handles one STSPIN820 device.

At each tick timer pulse end, a callback is executed to call the step clock handler which controls the motor motion by managing:

- motion status (e.g., stop motor at target destination)
- motor direction via GPIO level
- relative and absolute motor position in microsteps
- the speed through zero, positive and negative acceleration

The speed is set by varying the step clock frequency and, optionally, the step mode when the automatic full step switch feature is enabled. The timer used for the step clock is configured in output compare mode. A new capture compare register value is calculated at each step clock handler call to achieve frequency control.

The speed is a linear function of the step clock frequency for a given micro-stepping mode, which can be varied by the software from full to 1/256th step.

To use the STSPIN820 driver library, you must run the initialization function which:

- sets up the required GPIOs to enable the bridges and manage fault pin EN\FAULT, dedicated MODE1, MODE2 and MODE3 step selection pins, the DIR pin for motor direction, the DECAY pin for decay mode selection and the standby reset pin STBY\RESET;
- sets up the timer in output compare mode for the STCK pin and the timer reference voltage generation in PWM mode for REF pin;
- loads the driver parameters with values from stspin820_target_config.h or defined in the main function using a dedicated initialization structure.

Driver parameters can be modified after initialization by calling specific functions. You can also write callback functions and attach them to:

- the flag interrupt handler to perform certain actions when an overcurrent or a thermal alarm is reported
- the error handler which is called by the library when it reports an error

Subsequent motion commands include:

- `BSP_MotorControl_Move` to move a given number of steps in a specific direction
- `BSP_MotorControl_GoTo, BSP_MotorControl_GoHome, BSP_MotorControl_GoMark` to go to a specific position using the shortest path
- `BSP_MotorControl_CmdGoToDir` to go in a specific direction to a specific position
- `BSP_MotorControl_Run` to run indefinitely

The speed profile is completely handled by the microcontroller. The motor starts moving at the `BSP_MotorControl_SetMinSpeed` minimum speed setting, which is then altered at each step by the `BSP_MotorControl_SetAcceleration` acceleration value.

If the target position of a motion command is far enough, the motor performs a trapezoidal move by:

- accelerating with the device acceleration parameter
- remaining steady at `BSP_MotorControl_SetMaxSpeed` maximum speed
- decelerating by `BSP_MotorControl_SetDeceleration`
- stopping at the target destination

If the target position is too close for the motor to reach maximum speed, it performs a triangular move involving:

- acceleration
- deceleration
- stopping at the target destination

A motion command can be stopped anytime with `BSP_MotorControl_SoftStop` progressively decreasing the speed using the deceleration parameter or the `BSP_MotorControl_HardStop` command which immediately stops the motor. The power bridge is automatically disabled when the motor stops if the `HIZ_MODE` stop mode was previously set (`BSP_MotorControl_SetStopMode`).

Direction, speed, acceleration and deceleration can be changed either when the motor is stopped or when the motion is requested via `BSP_MotorControl_Run`.

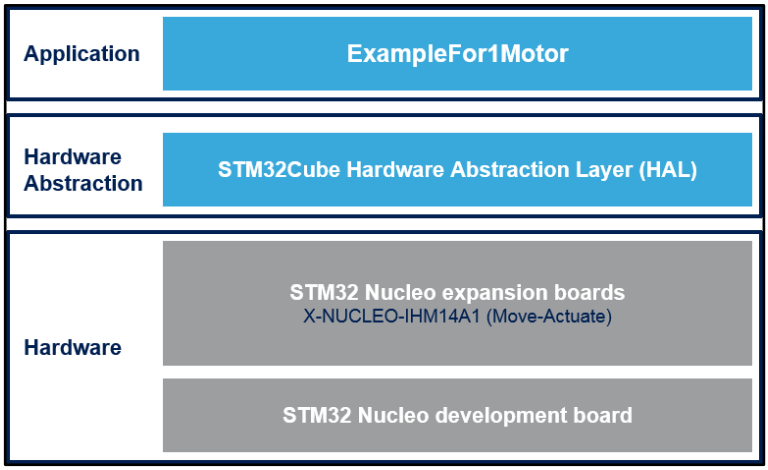
To block new commands before the completion of previous ones, `BSP_MotorControl_WaitWhileActive` locks program execution until the motor stops.

`BSP_MotorControl_SelectStepMode` can change the step mode from full to 1/256th step. When step mode is changed, the device and the current position and speed are reset.

3.2 Architecture

This software expansion for STM32Cube fully complies with STM32Cube architecture and expands it to enable the development of applications using stepper motor drivers.

Figure 2: X-CUBE-SPN14 software architecture



The software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube with a board support package (BSP) for the motor control expansion board and a BSP component driver for the STSPIN820 low voltage stepper motor driver.

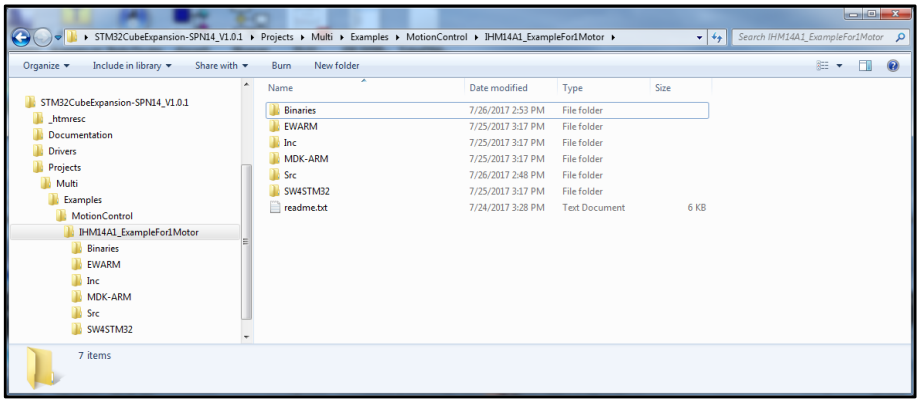
The software layers used by the application software are:

STM32Cube HAL layer: a simple, generic and multi-instance set of APIs (application programming interfaces) to interact with upper application, library and stack layers. It is composed of generic and extension APIs based on a common architecture so that layers built on it, such as the middleware layer, can function without requiring specific microcontroller Unit (MCU) hardware configurations. This structure improves library code reusability and guarantees an easy portability on 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 LED and the user button, and helps in identifying the specific board version. The motor control BSP provides the programming interface for various motor driver components. It is associated with the BSP component for the STSPIN820 motor driver in the X-CUBE-SPN14 software.

3.3 Folder structure

Figure 3: Folder structure



The software is located in two main folders:

- **Drivers**, with:
  - the STM32Cube HAL files in the STM32L0xx_HAL_Driver, STM32F0xx_HAL_Driver, STM32F3xx_HAL_Driver or STM32F4xx_HAL_Driver subfolders. These files are taken directly from the STM32Cube framework and only include those required to run the motor driver examples.
  - a CMSIS folder with the CMSIS (Cortex® microcontroller software interface standard), vendor-independent hardware abstraction layer for the Cortex-M processor series from ARM. This folder is also unchanged from the STM32Cube framework.
  - a BSP folder with the code files for X-NUCLEO-IHM14A1 configuration, the STSPIN820 driver and the motor control API.
- **Projects**, which contains several use examples of the STSPIN820 motor driver for different STM32 Nucleo platforms.

3.3.1 BSP folder

The X-CUBE-SPN14 software includes the following BSPs:

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

These BSPs provide an interface for each compatible STM32 Nucleo board to configure and use its peripherals with the X-NUCLEO-IHM14A1 expansion board. Each subfolder has two.c/.h file pairs:

- **stm32XXxx_nucleo.c/h**: these unmodified STM32Cube framework files provide the user button and LED functions for the specific STM32 Nucleo board.
- **stm32XXxx_nucleo_ihm14a1.c/h**: these files are dedicated to the configuration of the PWMs, the GPIOs, and interrupt enabling/disabling required for X-NUCLEO-IHM14A1 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 L6474, powerSTEP01, L6208 and STSPIN820 via MotorControl/motorcontrol.c/h file pair. These files define all the driver configuration and control functions, which are then mapped to the functions of the motor driver component used on the given expansion board via 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. For X-CUBE-SPN14, the structure is called stspin820Drv (see file: BSP\Components\stspin820\stspin820.c).

As the motor control BSP is common for all motor driver expansion boards, all its functions may not be available for a given expansion board. Unavailable functions are replaced by null pointers during the instantiation of the motorDrv_t structure in the driver component.

3.3.1.3 STSPIN280 BSP component

The STSPIN820 BSP component provides the driver functions of the STSPIN820 motor driver in the folder stm32_cube\Drivers\BSP\Components\STSPIN820.

This folder has 3 files:

- **stspin820.c**: core functions of the STSPIN820 driver

- **stspin820.h**: declaration of the STSPIN820 driver functions and their associated definitions
- **stspin820_target_config.h**: predefined values for the STSPIN820 parameters and for the motor devices context

3.3.2 Project folder

For each STM32 Nucleo platform, one example project is available in stm32_cube\Projects\Multi\Examples\MotionControl\:

- **IHM14A1_ExampleFor1Motor** examples of control functions for single-motor configurations

The example has a folder for each compatible IDE:

- **EWARM** for IAR
- **MDK-ARM** for Keil
- **SW4STM32** for OpenSTM32

The following code files are also included:

- **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 based on the motor control library for STSPIN820)
- **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

MCU control of a single STSPIN820 (one X-NUCLEO-IHM14A1 board) and communication between the two is handled through seven GPIOs (STBY\RESET, EN\FAULT, MODE1, MODE2, MODE3, DIR, DECAY pins) and a PWM for REF pin. The GPIO for the STCK pin is configured to be used as a TIMER OUTPUT COMPARE alternate function.

For the handling of overcurrent and the overtemperature alarms, the X-CUBE-SPN14 software uses an external interrupt configured on the GPIO used for the EN\FAULT pin, after it has enabled or disabled the power bridges.

Table 2: Required resources for the X-CUBE-SPN14 software

Resources F4xx	Resources F3xx	Resources F0xx	Resources L0xx	Pin	Features (board)
Port A GPIO 10 EXTI15_10_IRQn	Port A GPIO 10 EXTI15_10_IRQn	Port A GPIO 10 EXTI4_15_IRQn	Port A GPIO 10 EXTI4_15_IRQn	D2	EN/FAULT (EN)
Port B GPIO 3 Timer2 Ch2	Port B GPIO 3 Timer2 Ch2	Port B GPIO 3 Timer15 Ch1	Port B GPIO 3 Timer2 Ch2	D3	STCK (CLK)
Port B GPIO 4				D5	DECAY (DEC)
Port A GPIO 8				D7	DIRECTION (DIR)

Resources F4xx	Resources F3xx	Resources F0xx	Resources L0xx	Pin	Features (board)
Port A GPIO 9				D8	STBY/RESET (STBY)
Port C GPIO 7 Timer3 Ch2	Port C GPIO 7 Timer3 Ch2	Port C GPIO 7 Timer3 Ch2	Port C GPIO 7 Timer22 Ch2	D9	PWM REF (REF)
Port A GPIO 7				D11	MODE3 (M3)
Port A GPIO 6				D12	MODE2 (M2)
Port A GPIO 5				D13	MODE1 (M1)

3.5 APIs

The API of the X-CUBE-SPN14 software is defined in the motor control BSP. Its functions contain the “BSP_MotorControl_” prefix.



Not all the functions of this module are available for the STSPIN820 and hence the X-NUCLEO-IHM14A1 expansion board.

Full user API function and parameter descriptions are compiled in an HTML file in the software Documentation folder.

3.6 Sample application description

An example application using the X-NUCLEO-IHM14A1 expansion board with a compatible STM32 Nucleo board is provided in the Projects directory, with ready-to-build for multiple IDEs (see *Section 3.3.2: "Project folder"*).

4 System setup guide

4.1 Hardware description

This section describes the hardware components needed to develop a stepper motor driver-based application using X-CUBE-SPN14.

The following sub-sections describe the individual components.

4.1.1 STM32 Nucleo platform

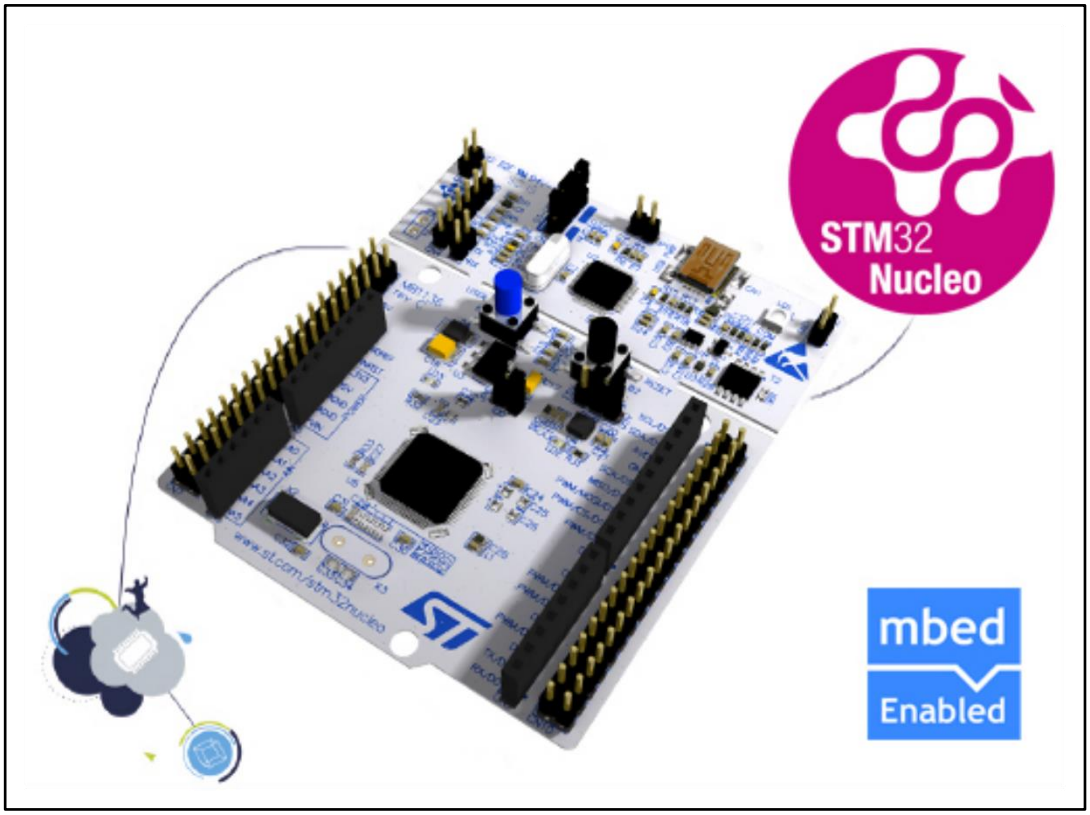
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

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 4: STM32 Nucleo board



Information regarding the STM32 Nucleo board is available at *www.st.com/stm32nucleo*

4.1.2 X-NUCLEO-IHM14A1 stepper motor driver expansion board

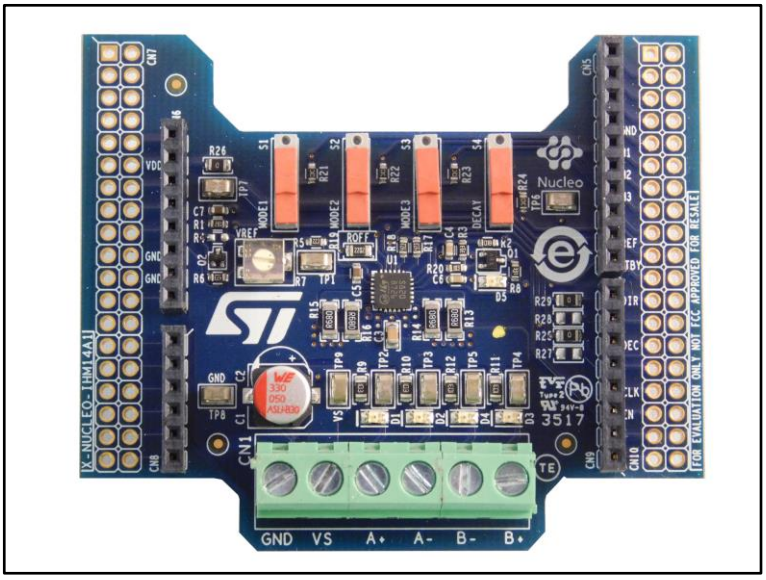
The X-NUCLEO-IHM14A1 motor driver expansion board is based on the STSPIN820 monolithic driver for stepper motors.

It represents an affordable, easy-to-use solution for driving stepper motors in your STM32 Nucleo project, implementing motor driving applications such as 2D/3D printers, robotics and security cameras.

The STSPIN820 implements a PWM current control with constant OFF time adjustable via an external resistor and a microstepping resolution up to the 256th step.

The X-NUCLEO-IHM14A1 expansion board is compatible with the Arduino UNO R3 connector and the ST morpho connector, so it can be plugged to the STM32 Nucleo development board and stacked with additional X-NUCLEO expansion boards.

Figure 5: X-NUCLEO-IHM14A1 stepper motor driver expansion board



Information regarding the X-NUCLEO-IHM14A1 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:

- 1 bipolar (7 to 45 V) stepper motor
- an external DC power supply with two electric cables for the X-NUCLEO-IHM14A1 board
- a USB type A to mini-B USB cable to connect the STM32 Nucleo board to a PC

4.2 Software description

The following software components are needed in order to set up the suitable development environment for creating applications based on the motor driver expansion board:

- X-CUBE-SPN14 STM32Cube expansion for STSPIN820 low voltage stepper motor driver application development. The X-CUBE-SPN14 firmware and related documentation is available on www.st.com.
- One of the following development tool-chain and compilers:
  - Keil RealView Microcontroller Development Kit (MDK-ARM) toolchain V5.17

- IAR Embedded Workbench for ARM (EWARM) toolchain V7.40
- 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 Setup to drive a single motor

Configure the following jumpers on the STM32 Nucleo board:

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

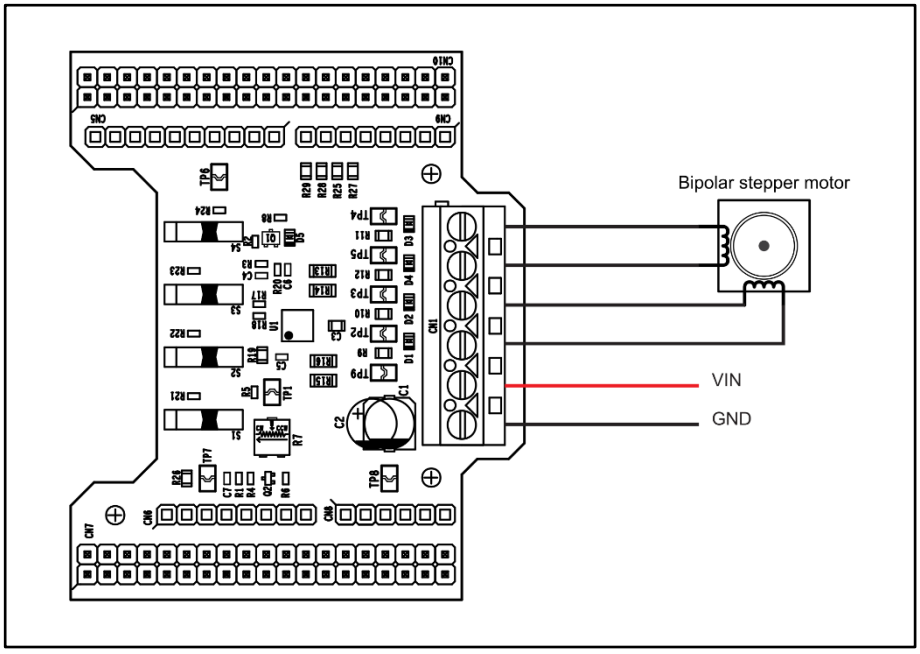
Configure the X-NUCLEO-IHM14A1 expansion board thus:

- Tune R7 potentiometer to 1 kΩ.
- Set S1, S2, S3 and S4 switch to the pull-down side as in *Figure 5: "X-NUCLEO-IHM14A1 stepper motor driver expansion board"*. The micro-stepping mode is selected through the MODE1, MODE2 and MODE3 levels controlled by the STM32 Nucleo board.

Once the board is properly configured:

- Plug the X-NUCLEO-IHM14A1 expansion board on top of the STM32 Nucleo board via the Arduino UNO connectors
- Connect the STM32 Nucleo board to a PC with the USB cable through USB connector CN1 to power the board
- Power on the X-NUCLEO-IHM14A1 expansion board by connecting Vin and Gnd connectors to a DC power supply
- Connect the stepper motor to the X-NUCLEO-IHM14A1 bridge connectors A+/- and B+/-

Figure 6: Board connections



Once the system setup is ready:

- Open your preferred toolchain
- Depending on the STM32 Nucleo board, open the software project from:
  - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM14A1_ExampleFor1Motor\YourToolChainName\STM32F401RE-Nucleo for Nucleo STM32F401
  - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM14A1_ExampleFor1Motor\YourToolChainName\STM32F030R8-Nucleo for Nucleo STM32F334
  - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM14A1_ExampleFor1Motor\YourToolChainName\STM32F030R8-Nucleo for Nucleo STM32F030
  - \stm32_cube\Projects\Multi\Examples\MotionControl\IHM14A1_ExampleFor1Motor\YourToolChainName\STM32L053R8-Nucleo for Nucleo STM32L053
- To adapt the default STSPIN820 parameters to your low voltage stepper motor characteristics, either:
  - use `BSP_MotorControl_Init` with the NULL pointer and open stm32_cube\ Drivers\ BSP\ Components\ STSPIN820\ STSPIN820_target_config.h to modify the parameters according to your needs
  - use `BSP_MotorControl_Init` with the address of the `initDevicesParameters` structure with appropriate values.
- Rebuild all files and load your image into target memory.
- Run the example. The motor automatically starts (See main.c for demo sequence details).

5 Revision history

Table 3: Document revision history

Date	Version	Changes
17-Oct-2017	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.

© 2017 STMicroelectronics – All rights reserved