Getting started with the STSW-IOD003 software package for L6362A IO-Link communication transceiver device IC based on STM32Cube

Introduction

The STSW-IOD003 is an evaluation software for the STEVAL-IOD003V1 evaluation board which integrates the L6362A IO-Link transceiver device.

The software runs on the STM32 and provides basic management of the L6362A device.

It is built on top of STM32Cube software technology that eases portability across different STM32 microcontrollers.

The software comes with a sample implementation to show its main functionalities. It is compatible with NUCLEO-F401RE or NUCLEO-L073RZ when connected to a STEVAL-IOD003V1 evaluation board.

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
IC	Integrated circuit
IDE	Integrated development environment
LED	Light emitting diode
MCU	Microcontroller unit

2 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.

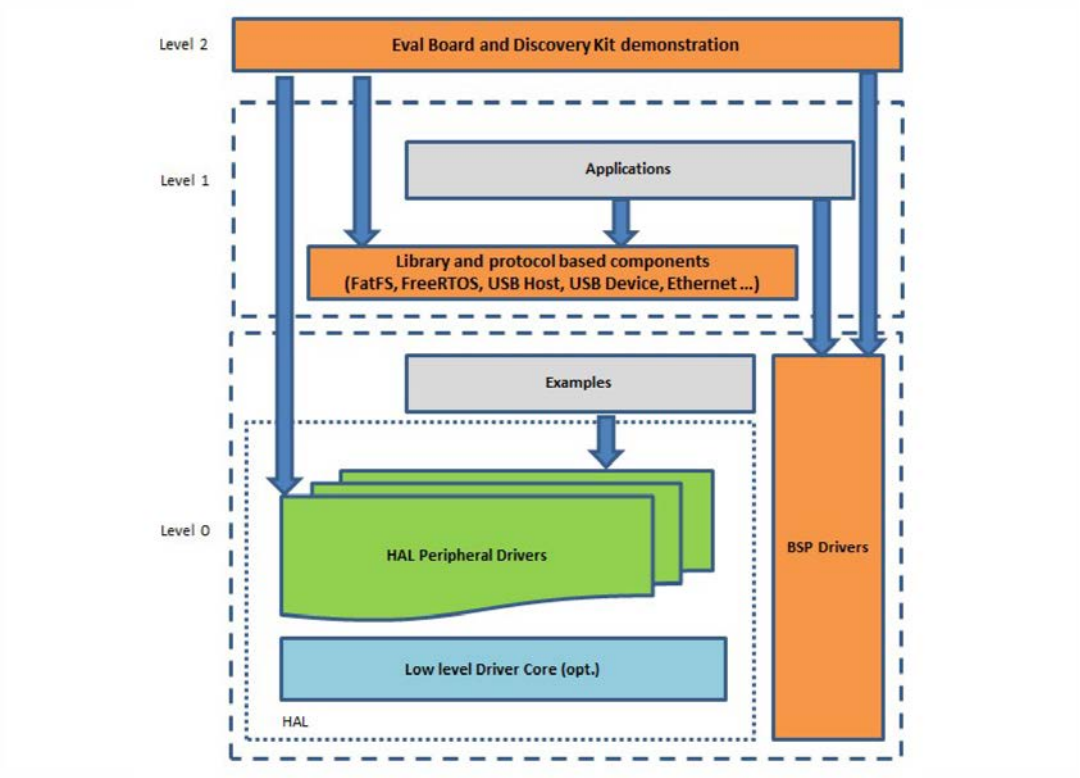
STM32Cube version 1.x includes:

- 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 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 STSW-IOD003 software package based on STM32Cube

3.1 Overview

The STSW-IOD003 software package is based on STM32Cube functionality and features:

- Driver layer for the management of the L6362A IO-Link communication transceiver device IC integrated in the STEVAL-IOD003V1 evaluation board
- GPIOs and IRQs configuration
- I/Q channel control for reception and transmission
- Fault interrupt handling
- Sample application for controlling an L6362A device
- Easy portability across different MCU families, thanks to STM32Cube
- Free, user-friendly license terms

To use the IO-Link device library, first call its initialization function to:

- set up different GPIOs to handle:
  - the L6362A output enable pin *EN*
  - the L6362A diagnostics pin *DIAG* (if the enable and diagnostics pins are the same on the L6362A, they are handled through two different GPIOs on the STM32)
  - the L6362A overload pin *OL*
- configure the L6362A *OUTi/Q* and *IN2* pins which are respectively used for reception and transmission by associating them to one of the STM32 UART peripheral.

You can also write callback functions and attach them to:

- the fault interrupt handler depending on the actions you want to perform when an alarm is reported via the DIAG pin (BSP_IOLinkDevice_AttachFaultInterrupt)
- the overload interrupt handler depending on the actions you want to perform when an alarm is reported via the OL pin (BSP_IOLinkDevice_AttachOLInterrupt)
- the error handler which is called by the library when it reports an error (BSP_IOLinkDevice_AttachErrorHandler)
- the *OUTi/Q* data handler which is called by the library each time a data is received from the OUTi/Q pin (BSP_IOLinkDevice_AttachOutIqDataHandler).

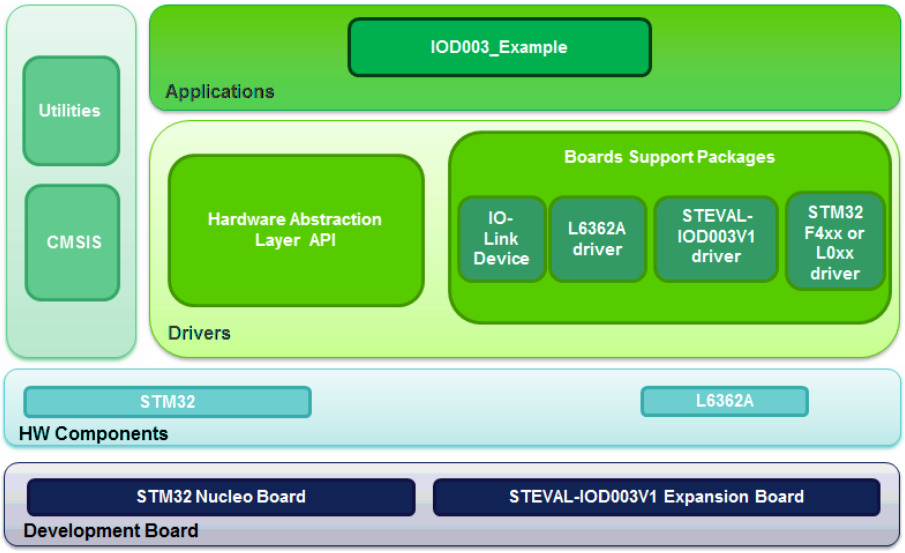
After that, you can request specific actions to:

- enable or disable the I/Q output (BSP_IOLinkDevice_SetEnable)
- send data to the I/Q channel (BSP_IOLinkDevice_SendInData) by enabling the I/Q output (BSP_IOLinkDevice_SetEnable) and disabling it when data are transmitted from the I/Q channel
- set the baud rate of the I/Q channel by selecting COM1 , COM2 or COM3 (BSP_IOLinkDevice_SelectComMode)

3.2 Architecture

This fully compliant STM32Cube software expansion enables development of applications using IO-Link device IC.

Figure 2. **STSW-IOD003 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 IO-Link device (STEVAL-IOD003V1) and a BSP component driver for the L6362A IO-Link communication transceiver device.

The software layers used by the application software to access and use the transceiver device 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 IO-Link devices, the board BSP provides a programming interface for various IO-Link master or device components. In the STSW-IOD003 software, it is associated with the BSP component for the L6362A device transceiver.

3.3 Folder structure

The software is packaged in the following main folders:

- **Drivers**:
  - STM32Cube HAL driver files located in the STM32L0xx_HAL_Driver or STM32F4xx_HAL_Driver subdirectories. These files are not described here as they are not specific for the STSW-IOD003 software but derive directly from the STM32Cube framework. Only the HAL files which are required to run the L6362A sample are present.
  - 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 the STEVAL-IOD003V1 configuration, the L6362A driver and the IO-Link device API (see Section 3.3.1 BSP folder).
- **Projects**: contains a sample L6362A application for different STM32 Nucleo platforms.

3.3.1 BSP folder

3.3.1.1 *STM32L0XX-Nucleo and STM32F4XX-Nucleo BSPs*

Depending on the STM32 Nucleo development board, these BSPs provide an interface to configure and use the development board peripherals with the STEVAL-IOD003V1 evaluation board.

Each subfolder (STM32L0XX-Nucleo/STM32F4XX-Nucleo) contains a couple of stm32XXxx_nucleo.c/h files which derive from the STM32Cube framework (with no modification) and provide the functions to handle the related STM32 Nucleo board user button and LEDs.

3.3.1.2 *STEVAL-IOD003V1 BSP*

This BSP is dedicated to the configuration of the the GPIOs, the UARTs, the interrupt enabling/disabling which are required for the STEVAL-IOD003V1 evalutation board. It includes two files steval_iod003v1.c/h.

3.3.1.3 *IO-Link device BSP*

This BSP provides a common interface to access the driver functions of various IO-Link device drivers like L6362A via a couple of c/h files, IOLink/iolink_device.c/h.

The files define all the functions to configure and control the IO-Link device driver and the functions are then mapped to the functions of the expansion board driver component with the structure: iolinkDeviceDrv_t (defined in Components\Common\iolink.h).

This structure defines a list of function pointers which are filled in the corresponding IO-Link device driver component. For the STSW-IOD003, the instance of the structure is called l6362Drv (see file: BSP\Components\l6362a\l6362a.c).

The IO-Link control BSP is common for all IO-Link device driver boards but not all the functions are available for a given board. In this case, during the instantiation of the iolinkDeviceDrv_t structure in the driver component, the unavailable functions are replaced by null pointers.

3.3.1.4 *L6362A BSP component*

The L6362A BSP component provides the L6362A IO-Link device driver functions in the stm32_cube\Drivers\BSP\Components\L6362a folder, which contains:

- **l6362a.c** for the L6362A driver core functions
- **l6362ah** for the L6362A driver function declaration and the associated definitions
- **l6362a_target_config.h** to predefine values for the L6362A parameters (initial COM mode)

3.3.2 Project folder

For each STM32 Nucleo platform, one sample project is available in the folder stm32_cube\Projects\Multi\Examples\IOLink\:

- **IOD003_Example**: implementation sample of the different IO-Link device BSP functions available for L6362A.

Each sample has a folder dedicated to the targeted IDE:

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

Each sample 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 sample based on the L6362A library)
- **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

The MCU controls the L6362A and communicate with it through GPIOs and UART using five pins (DIAG, EN, IN2, OL, OUTIQ).

Table 2. Required resources for the STSW-IOD003 software

Features (board)	Resources NUCLEO-L073RZ/ NUCLEO-F401RE	Board connector pin
Output enable (EN)	Port C GPIO 7	CN5-2
Diagnostics (DIAG)	Port A GPIO 4	CN8-3
Overload (OL)	Port A GPIO 6	CN5-5
I/Q input (IN2)	Port B GPIO 6 UART1 TX	CN5-3
I/Q output (OUTIQ)	Port A GPIO 10 UART1 TX	CN9-3

3.5 APIs

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

The STSW-IOD003 software API is defined in the IO-Link BSP (functions predefined through BSP_IOLinkDevice_).

3.6 Sample application description

A sample application using the STEVAL-IOD003V1 evalutation board with the NUCLEO-F401RE or NUCLEO-L073RZ board is provided in the “Projects” directory. Ready-to-build projects are available for multiple IDEs (see Section 3.3.2 Project folder).

4 System setup guide

4.1 Hardware description

4.1.1 STM32 Nucleo platform

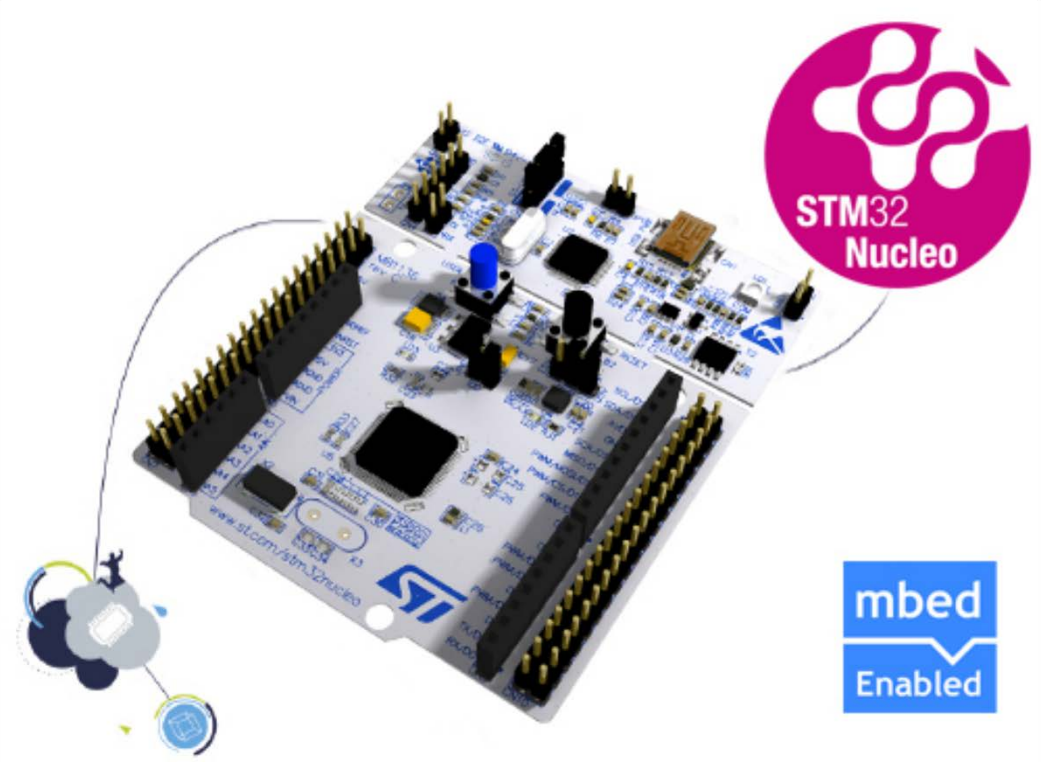
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

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 3. STM32 Nucleo board



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

4.1.2 STEVAL-IOD003V1 evaluation board

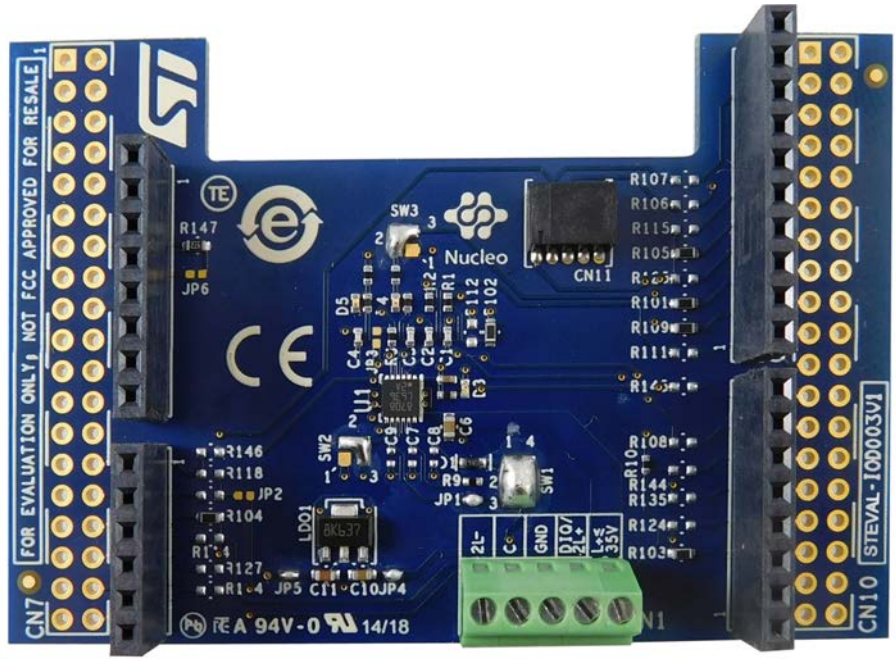
The STEVAL-IOD003V1 evaluation board is based on the L6362A IO-Link PHY device with full integrated EMC protection (according to IO-Link v1.1 specification) and surge protection (according to IEC 60947-5-2). It provides an affordable and easy-to-use solution for the development of IO-Link and SIO applications, letting you easily evaluate the communication features and robustness of the L6362A.

The on-board linear regulators (12 mA-3.3 V from L6362A and 100 mA-12 V from L78L12ABUTR) can be used to supply the micro-controller via the 24 V bus, instead of via USB.

When the L78L12ABUTR is enabled (default configuration), you can also perform evaluation of complete industrial sensor modules by connecting the STEVAL-IOD003V1 to a NUCLEO-L073RZ (or NUCLEO-L053R8) board and an X-NUCLEO-IKS01A2 expansion board.

The STEVAL-IOD003V1 interfaces with the STM32 controller via UART and GPIO pins and is compatible with the Arduino UNO R3 (default configuration) and ST morpho (optional, not mounted) connectors.

Figure 4. STEVAL-IOD003V1 evaluation board



4.1.3 Miscellaneous hardware components

To complete the hardware setup, you need:

- an external 24 V DC power supply with two electric cables to connect to the STEVAL-IOD003V1 evaluation board
- 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 needed for a suitable development environment for applications based on the IO-Link device evaluation board:

- STSW-IOD003 evaluation software based on STM32Cube dedicated to L6362A IO-Link communication transceiver device IC application development. The STSW-IOD003 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 V5.22
  - IAR Embedded Workbench for ARM (EWARM) toolchain V7.80
  - OpenSTM32 System Workbench for STM32 (SW4STM32)

4.3 Hardware and software setup

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

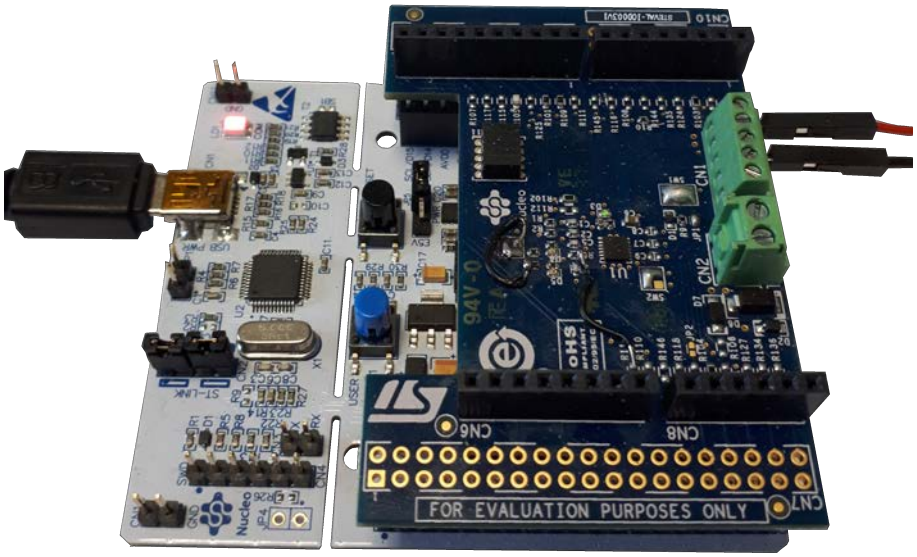
- JP1 off
- JP5 (PWR) on UV5 side
- JP6 (IDD) on

Step 1. Plug the STEVAL-IOD003V1 evaluation board on top of the STM32 Nucleo via the Arduino UNO connectors

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 evaluation board on by connecting its connectors CN1-1 (L+) to “+” and CN1-3 (L-) to the GND of the 24 V DC power supply

Figure 5. STEVAL-IOD003V1 evaluation board connections



Step 4. Open your preferred toolchain (MDK-ARM from Keil, EWARM from IAR, or SW4STM32 from www.openstm32.org)

Step 5. Depending on the STM32 Nucleo board used, open the software project from:

– \stm32_cube\Projects\Multi\Examples\IOLink\IOD003_Example \YourToolChainName\STM32F401RE-Nucleo for NUCLEO-F401RE

– \stm32_cube\Projects\Multi\Examples\IOLink\IOD003_Example \YourToolChainName\STM32L073RZ-Nucleo for NUCLEO-L073RZ

Step 6. To adapt the L6362A default parameters, use:

– the BSP_IOLinkDevice_Init function with the NULL pointer and open the file: stm32_cube\ Drivers \ BSP\ Components\ l6362a\ l6362a_target_config.h

– BSP_IOLinkDevice_Init function with the address of l6362_Init_t structure

Step 7. Rebuild all files and load your image into target memory

Step 8. Run the sample.

The demo sequence starts when you press the user button and performs a new step each time it is pressed (see main.c for the detailed demo sequence).

Revision history

Table 3. **Document revision history**

Date	Version	Changes
13-Jun-2018	1	Initial release.

Contents

List of tables

List of figures