User manual

Getting started with the STSW-IOM001 software package for L6360 IO-Link communication transceiver master IC based on STM32Cube

Introduction

The STSW-IOM001 is an evaluation software for the STEVAL-IOM001V1 evaluation board which integrates the L6360 IO-Link transceiver master.

The software runs on the STM32 and provides basic management of the L6360 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-F446RE when connected to one or more (up to four) STEVAL-IOM001V1 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.

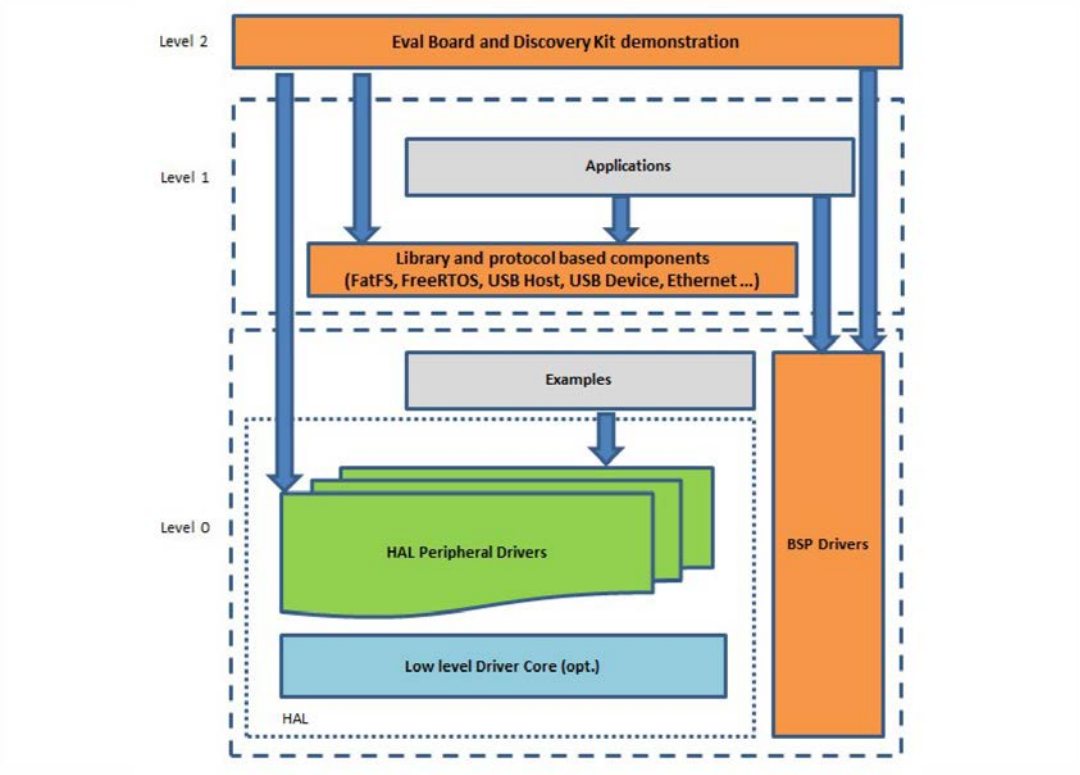
STM32Cube version 1.x includes:

- 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-IOM001 software package based on STM32Cube

3.1 Overview

The STSW-IOM001 software package is based on STM32Cube functionality and features:

- Driver layer for the management of the L6360 IO-Link communication transceiver master IC integrated in the STEVAL-IOM001V1 evaluation board
- Read and write of the L6360 register via I²C
- GPIOs and IRQs configuration
- I/Q channels for reception and transmission
- Fault interrupt handling
- Sample application for controlling up to four L6360 devices
- Easy portability across different MCU families, thanks to STM32Cube
- Free, user-friendly license terms

When starting the IO Link master library, you have to specify the number of L6360 chips connected to the STM32 Nucleo board. Once set, the number of L6360 devices must not be changed.

Depending on the device number, the library:

- sets the required GPIOs to handle
  - the L6360 reset pin *RST*
  - the L6360 C/Q output enable pin $EN_{C/Q}$
  - the L6360 L+ rail enable pin EN_{L+}(1)
  - the L6360 I/Q channel logic output pin $OUT_{I/Q}$
  - the L6360 *IRQ* pin used to report fault
- configures the L6360 $OUT_{C/Q}$ and $IN_{C/Q}$ pins respectively used for reception and transmission by associating them to one of the STM32 UART peripheral
- initializes an STM32 I²C peripheral used to drive the L6360 *SDA* and *SCL* pins
- disables the L6360 reset

1. *In case of high capacitive loads on L+rail, the user can enable the IPS161H (connected in parallel to the L+ switch of L6360) by L+on signal*

Once the initialization is done, you can modify the driver parameters by calling specific functions. Most of the library functions take a port ID (from 0 to 3) as input parameter which gives the possibility of specifying which L6360 device is addressed.

You can also write callback functions and attach them to:

- the flag interrupt handler depending on the actions you want to perform when an overcurrent or a thermal alarm is reported via the IRQ pin (BSP_IOLinkMaster_AttachIrqInterrupt)
- the error handler which is called by the library when it reports an error (BSP_IOLinkMaster_AttachErrorHandler)
- the $OUT_{C/Q}$ data handler called by the library each time it receives data from the $OUT_{C/Q}$ pin (BSP_IOLinkMaster_AttachOutCqDataHandler).

After that, you can request specific actions to:

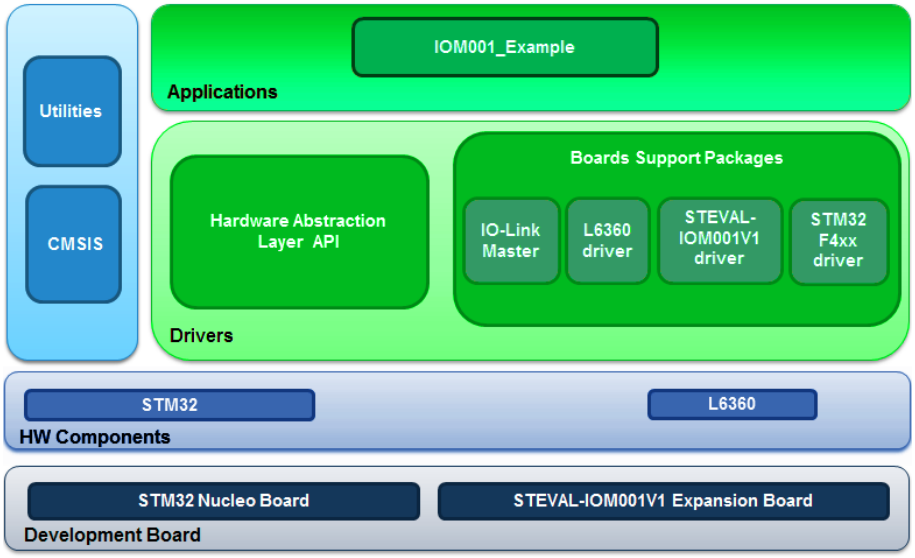
- enable or disable
  - the reset (BSP_IOLinkMaster_HandleReset)
  - the C/Q output (BSP_IOLinkMaster_SetCQOutputEnable)
  - the L+ line (BSP_IOLinkMaster_SetLPlusEnable) or its associated power supply (BSP_IOLinkMaster_SetLPowerSupply)
- set the I²C frequency (BSP_IOLinkMaster_GetI2cFreq)
- read an L6360 register (BSP_IOLinkMaster_ReadRegister) or write it (BSP_IOLinkMaster_WriteRegister)

- send data to the C/Q channel (BSP_IOLinkMaster_SendInCqData) by enabling the C/Q output (BSP_IOLinkMaster_SetCQOutputEnable) and disabling it when data are transmitted from the C/Q channel
- set the baud rate of the C/Q channel by selecting COM1 , COM2 or COM3 (BSP_IOLinkMaster_SelectComMode)

3.2 Architecture

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

Figure 2. STSW-IOM001 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-IOM001V1) and a BSP component driver for the L6360 IO-Link communication master transceiver.

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-IOM001 software, it is associated with the BSP component for the L6360 master transceiver.

3.3 Folder structure

The software is packaged in the following main folders:

- **Drivers**:
  - STM32Cube HAL driver files located in the STM32F4xx_HAL_Driver subdirectories. These files are not described here as they are not specific for the STSW-IOM001 software but derive directly from the STM32Cube framework. Only the HAL files which are required to run the L6360 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-IOM001V1 configuration, the L6360 driver and the IO-Link device API (see BSP folder).
- **Projects**: contains a sample L6360 application for different STM32 Nucleo platforms.

3.3.1 BSP folder

3.3.1.1 *STM32F4XX-Nucleo-BSP*

This BSP provides an interface to configure and use the development board peripherals with the STEVAL-IOM001V1 evaluation board. It is composed of the **stm32F4x_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 *IO-Link master BSP*

This BSP provides a common interface to access the driver functions of various IO-Link master drivers like L6360 via a couple of c/h files, IOLink/iolink_master.c/h.

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

This structure defines a list of function pointers which are filled in the corresponding IO-Link master driver component. For the STSW-IOM001, the instance of the structure is called l6360Drv (see file: BSP\Components\l6360\l6360.c).

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

3.3.1.3 *STEVAL-IOM001V1 BSP*

This BSP is dedicated to the configuration of the the GPIOs, the I²C, the UARTs, the interrupt enabling/disabling which are required for the STEVAL-IOM001V1 evalutation board. It includes two files steval_iom001v1.c/h.

3.3.1.4 *L6360 BSP component*

The L6360 BSP component provides the L6360 IO-Link master driver functions in the stm32_cube\Drivers\BSP\Components\L6360 folder, which contains:

- **l6360.c** for the L6360 driver core functions
- **l6360h** for the L6360 driver function declaration and the associated definitions
- **l6360_target_config.h** to predefine values for the L6360 parameters (I²C and 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\:

- **IOM001_Example**: implementation sample of the different IO-Link master BSP functions available for L6360.

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 L6360 and communicate with it through GPIOs, I²C and UART. For one L6360 device (a STEVAL-IOM001V1 board), this requires the use of ten pins (ENCQ, ENL+,INCQ, I2C_SCL, I2C_SDA, IRQ, L +ON, OUTCQ, OUTIQ, RST).

If several L6360 devices are used (with a NUCLEO-F446RE board, up to four STEVAL-IOM001V1 boards can be used while you can use only two with the NUCLEO-F401RE boards), only I²C pins are shared. In this case, the software allows to address the different L6360:

- by their I²C address for the actions related to registers
- by a port number (from 0 to 3) for the other actions

Table 2. Required resources for the STSW-IOM001 software

Features (board)	Port	Resources NUCLEO-F446RE	Resources NUCLEO-F401RE	Board connector pin
I2C Clock (I2C_SCL)	All	Port B GPIO 8 I2C1 clock		CN5-10
I2C Data (I2C_SDA)	All	Port B GPIO 9 I2C1 data line		CN5-9
C/Q output enable (ENCQ)	0	Port C GPIO 0		CN8-6
C/Q input (INCQ)	0	Port A GPIO 9 UART1 TX		CN5-1
C/Q output (OUTCQ)	0	Port A GPIO 10 UART1 RX		CN9-3
I/Q output (OUTIQ)	0	Port B GPIO 6		CN5-3
L+ Enable (ENL+)	0	Port A GPIO 6		CN5-5
L+ Power supply (L+ON)	0	Port B GPIO 4		CN9-6
IRQ (IRQ)	0	Port A GPIO 4 EXTI4_IRQn		CN8-3
Reset (RST)	0	Port B GPIO 5		CN9-5
C/Q output enable (ENCQ)	1	Port A GPIO 5		CN5-6
C/Q input (INCQ)	1	Port A GPIO 0 UART4 TX	Port C GPIO 6 UART6 TX (add a wire to CN10-4)	CN8-1
C/Q output (OUTCQ)	1	Port A GPIO 1 UART4 RX	Port C GPIO 7 UART6 RX (add a wire to CN5-2)	CN8-2
I/Q output (OUTIQ)	1	Port C GPIO 9		CN10-1

Features (board)	Port	Resources NUCLEO-F446RE	Resources NUCLEO-F401RE	Board connector pin
L+ Enable (ENL+)	1	Port A GPIO 8		CN9-8
L+ Power supply (L+ON)	1	Port A GPIO 7 Timer3 Ch2		CN5-4
IRQ (IRQ)	1	Port B GPIO 0 EXTI0_IRQn		CN8-4
Reset (RST)	1	Port C GPIO 1		CN8-5
C/Q output enable (ENCQ)	2	Port A GPIO 12	Not available	CN10-12
C/Q input (INCQ)	2	Port B GPIO 10 UART3 TX	Not available	CN9-7
C/Q output (OUTCQ)	2	Port C GPIO 5 UART3 RX	Not available	CN10-6
I/Q output (OUTIQ)	2	Port B GPIO 13	Not available	CN10-30
L+ Enable (ENL+)	2	Port C GPIO 2	Not available	CN7-35
L+ Power supply (L+ON)	2	Port B GPIO 2	Not available	CN10-22
IRQ (IRQ)	2	Port B GPIO 1 EXTI1_IRQn	Not available	CN10-24
Reset (RST)	2	Port B GPIO 12	Not available	CN10-16
C/Q output enable (ENCQ)	3	Port C GPIO 8	Not available	CN10-2
C/Q input (INCQ)	3	Port C GPIO 6 UART6 TX	Not available	CN10-4
C/Q output (OUTCQ)	3	Port C GPIO 7 UART6 RX	Not available	CN5-2
I/Q output (OUTIQ)	3	Port B GPIO 14	Not available	CN10-28
L+ Enable (ENL+)	3	Port C GPIO 4	Not available	CN10-34
L+ Power supply (L+ON)	3	Port B GPIO 15	Not available	CN10-26
IRQ (IRQ)	3	Port C GPIO 3 EXTI3_IRQn	Not available	CN7-37
Reset (RST)	3	Port A GPIO 11	Not available	CN10-14

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-IOM001 software API is defined in the IO-Link BSP (functions predefined through BSP_IOLinkMaster_).

3.6 Sample application description

A sample application using the STEVAL-IOM001V1 evalutation board with the NUCLEO-F401RE or NUCLEO-F446RE 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

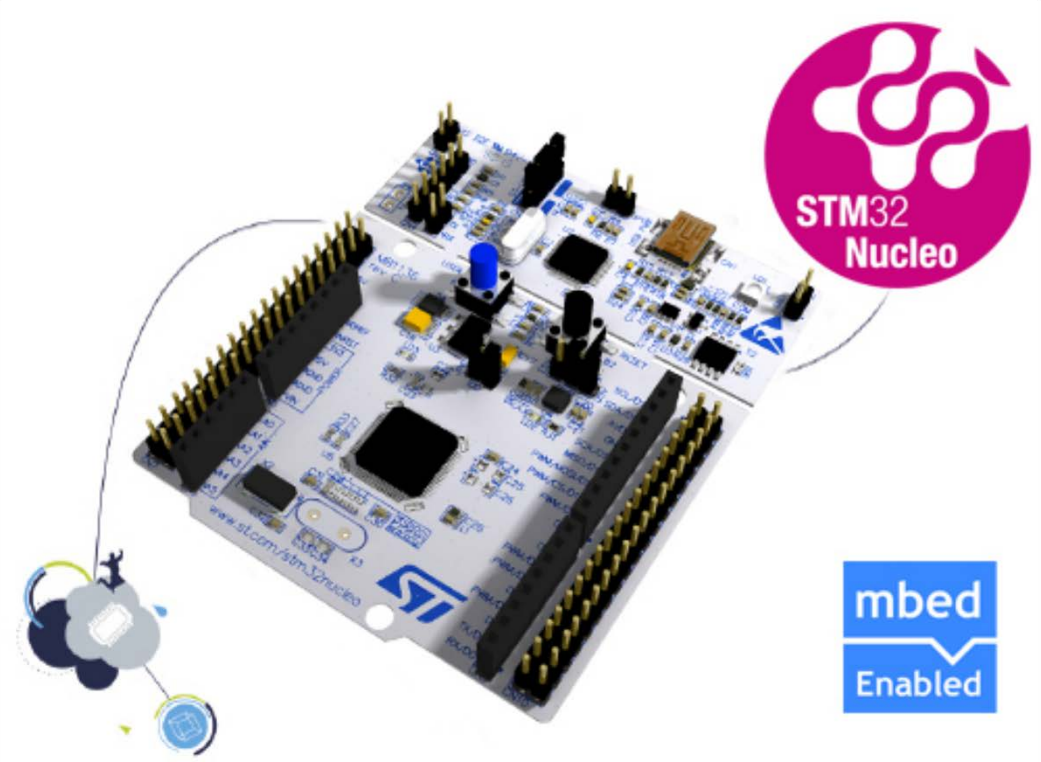
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

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-IOM001V1 evaluation board

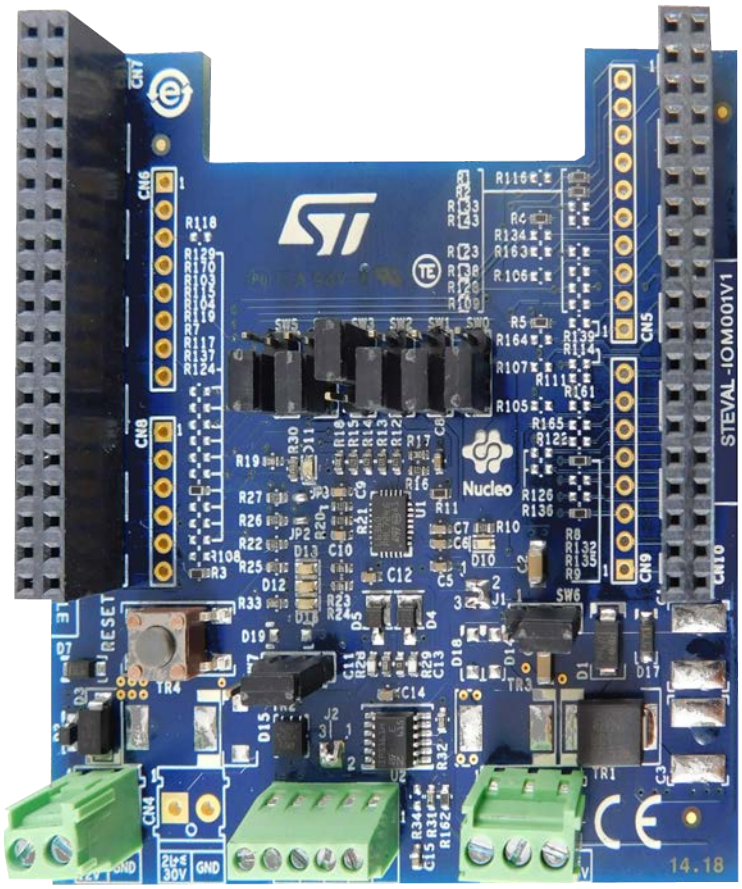
The STEVAL-IOM001V1 evaluation board is based on the L6360 IO-Link master transceiver with physical layer compliant with IO-Link v1.1 specification.

Together with the STSW-IOM001 example code, it provides an affordable and easy-to-use solution for the development of IO-Link applications, letting you easily evaluate the communication features and robustness of the L6360.

You can also perform evaluation of multiple ports industrial IO-Link master modules by connecting up to four STEVAL-IOM001V1 with few solder bridge modifications.

The STEVAL-IOM001V1 interfaces with the STM32 Nucleo microcontroller via UART and GPIO pins and is compatible with the Arduino UNO R3 (optional configuration) and ST morpho (default configuration) connectors.

Figure 4. STEVAL-IOM001V1 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-IOM001V1 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 master evaluation board:

- STSW-IOM001 evaluation software based on STM32Cube dedicated to L6360 IO-Link communication master transceiver IC application development. The STSW-IOM001 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

4.3.1 Setup for driving a single STEVAL-IOM001V1 evaluation board

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

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

The STEVAL-IOM001V1 evaluation board has to be configured as follows:

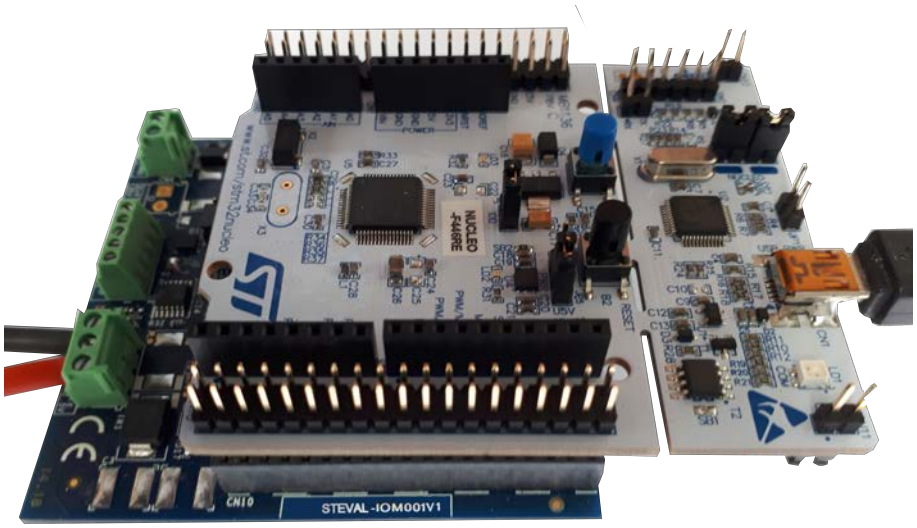
- SW0, SW1, SW2, SW3, SW4, SW5 and SW7 jumpers set to position 1-2
- SW4 and SW6 jumpers set to position 2-3

Step 1. Plug the STEVAL-IOM001V1 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 CN1-3 to “+” and CN1-1 to the GND of the 24 V DC power supply

Figure 5. STEVAL-IOM001V1 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 \STM32F446RE-Nucleo for NUCLEO-F446RE

Step 6. To adapt the L6360 default parameters, use:

- the BSP_IOLinkMaster_Init function with the NULL pointer and open the file: stm32_cube\ Drivers \ BSP\ Components\ L6360\ L6360_target_config.h
- BSP_IOLinkMaster_Init function with the address of l6360_Init_t structure

Step 7. Set to 1 the constant NUMBER_OF_IOM001_EXPANSIONS_BOARDS_USED in the main.c file, as there is only one STEVAL-IOM001V1 evaluation board

Step 8. Rebuild all files and load your image into target memory

Step 9. 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).

4.3.2 Setup for driving several STEVAL-IOM001V1 evaluation boards

To use several boards (up to two NUCLEO-F401RE and up to four NUCLEO-F446RE), the STM32 Nucleo configuration is the same as the single STEVAL-IOM001V1 evaluation board setup (for details, refer to UM2414 on www.st.com).

The setup of the additional STEVAL-IOM001V1 boards requires to solder some supplementary resistors. The detailed setup is provided in the STEVAL-IOM001V1 user manual on www.st.com.

The I²C address must be different for each board. By default, the firmware is set to use the I²C addresses provided by the following configuration:

- for the second board, SW0 must be set to 2-3, SW1 and SW2 to 1-2
- for the third board, SW1 must be set to 2-3, SW0 and SW2 to 1-2
- for the fourth board, SW0 and SW1 must be set to 2-3, SW2 to 1-2

Moreover, if the STM32 Nucleo board is the NUCLEO-F401RE, wires must be added on the second board among:

- CN8-1 and CN10-4
- CN8-2 and CN5-2

Once the board are properly configured, they can be stacked at the bottom of the STM32 Nucleo. A 24 V DC power supply must be provided for each evalution board.

To launch the software, follow the procedure described in Section 4.3.1 Setup for driving a single STEVAL-IOM001V1 evaluation board but you have to set the constant NUMBER_OF_IOM001_EXPANSIONS_BOARDS_USED to the corresponding number of STEVAL-IOM001V1 evaluation boards used.

Revision history

Table 3. **Document revision history**

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

Contents

List of tables

List of figures