

STSW-SILKT01 Firmware for STEVAL-SILKT01 User Manual

[Home](#) » [ST](#) » STSW-SILKT01 Firmware for STEVAL-SILKT01 User Manual 



UM3282

**Getting started with the STSW-SILKT01 firmware for the STEVAL-SILKT01
User manual**

Contents

- [1 Introduction](#)
- [2 Overview](#)
- [3 Firmware architecture](#)
- [4 How functional safety works at system level](#)
- [5 Documents / Resources](#)
 - [5.1 References](#)

Introduction

The STEVAL-SILKT01 is a kit composed by three boards, one main board STEVAL-SILKTA01 and two daughter boards STEVAL-SILKTB01.

The STSW-SILKT01 firmware package implements an application use case to manage diagnostic functions and on board devices for digital I/O to enable the system for a safety integrity level (SIL3) application use case.

The firmware architecture and the implemented API enable the firmware to support the output driving in operating mode the diagnostic functions and the system safe state condition in case a fault event, as required by the IEC61508 to meet the SIL3 requirements.

The items related to the diagnostic in charge to the user are:

- The self-test library certified by TÜV Rheinland to test the CPU, the RAM, and flash memories of the microcontroller during operation; the user has to enable this diagnostic by using the X-CUBE-STL dedicated

package.

Note: Pay particular attention to the requirements stated for the application software included in UM2331 for the STM32H7 and UM3167 for the STM32G4

- Diagnostic features, such as:
 - Supply voltage monitoring, to detect the undervoltage and overvoltage events
 - Voltage supervisor with watchdog functionality
 - Board temperature monitoring
 - Overcurrent and overvoltage detections through the embedded IPS protections

Regarding the digital I/O handling, the firmware supports the following features that are managed in both the STEVAL-SILKTA01 and the STEVAL-SILKTB01:

- Digital input signal check, using a dedicated timer in input capture mode, to verify the validity of the input signal and the correct functionality of the two CLT03-1SC3
- Digital output management, which implements the output driving for load actuation and board safe state condition in case of fault
- Output feedback detection on the output to verify the correct output driving during normal operation

The communication level with integration of dedicated APIs integration, can manage real time communication for Ethernet/IP,

Modbus, Canopen. In the package is available a free demo stack for Modbus RTU communication.

Overview

The STSW-SILKT01 firmware is developed using IAR Workbench 8.50.9. It is compliant with the STM32Cube framework and offers an application example (not assessed by the TÜV SUD Group), which implements the basic functions to meet the standard requirements for a SIL3 application.

It integrates the library certified by TÜV Rheinland for the STM32H743ZG and STM32G431RB microcontrollers.

The supported features are:

- Digital output supply voltage interruption mechanism
- Diagnostic coverage supported by dedicated API
- Safety function management
- X-CUBE-STL self-test library certified by TÜV Rheinland according to IEC61508 (this is the only firmware component with the official certification)
- Real Time Communication using Ethernet technology and serial communication
- BSP libraries for the on-board IC management
- STM32 framework compliance The STSW-SILKT01 firmware architecture has been implemented to enable the hardware solution as valid reference evaluation board for functional safety application use cases, where the SIL3 level is required. This hosts all the APIs needed for system actuation to activate and put in safe state the system itself, supporting also a set of diagnostic functions to protect the system against failure event put in place safe action.

The STEVAL-SILKTB01 hardware architecture host two digital inputs with the CLT03-1SC3 and one digital output based on high side and low side control using the IPS1025H and IPS4260L.

As required by the IEC61508, the output path is monitored by each microcontroller (the STM32H743ZG and the STM32G431RB), to verify if the output state is in accordance with the IPSs driving signals. The monitoring is managed by the firmware through dedicated GPIOs in input, reading the voltage values on them. Both for digital input as well as for digital output and data processing, the complete system is managed guaranteeing the

redundancy of the signal check or driving signal using dedicated logic circuit at hardware level that are transduced in dual signal verification at microcontroller level.

Supported at application level and not included in the firmware package there are the safety library available in the X-CUBE-STL self-test library, implemented to perform during operating condition of the complete system, a check of dedicated memory location of the MCU (Flash, Ram)

The kit STEVAL-SILKT01 offers three different connectivity interface, that can be supported at firmware level with integration of dedicated APIs.

Firmware architecture

The firmware package is composed by two firmware inside, one related to the main board STEVAL-SILKTA01 managing communication versus the remote unit and then operators, and the other, related to the STEVALSILKTB01, related to the daughter boards that must be connected to the main.
Both firmware from a safety point of view, manages the functional safety features implementing redundancy on:

- Data processing using two microcontrollers
- Digital input/output management
- Diagnostic functions not only related to voltage monitoring but also for feedback signalization and overcurrent/overtemperature protections

According to the firmware implementation, the complete system has the capability to manage different communication channels to guarantee a remote connection, to implement an internal communication between the STEVAL-SILKTA01 and the two STEVAL-SILKTB01, control system status by monitoring supply parameters, thermal parameters and output line state, and in regards of each board, implements independently one to the other execution of safety functions to put the whole system in safe state in case of fault. Looking forward to the implementation of a system firmware package that it is not strongly linked to the hardware architecture, the architecture has been thought following the guidelines of the STM32CUBE framework highlighting for each board a structure like this:

Figure 1. STSW-SILKTA01 architecture

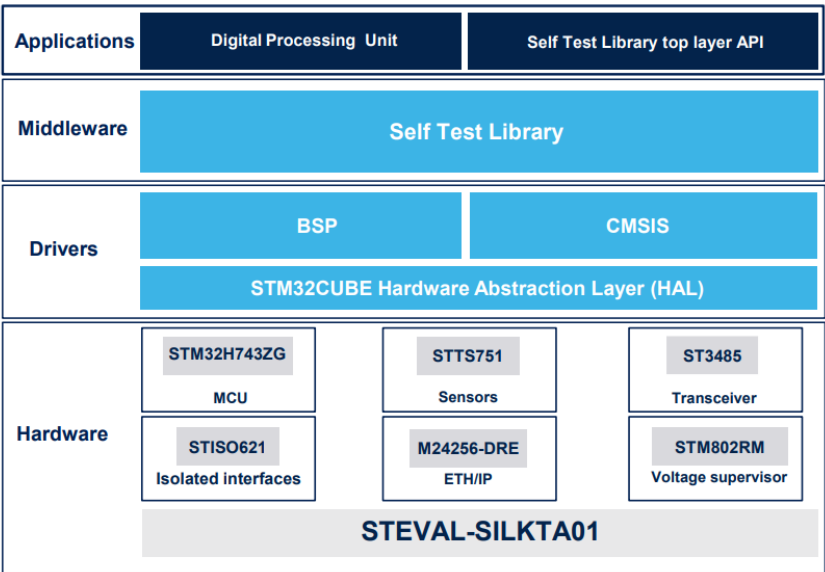
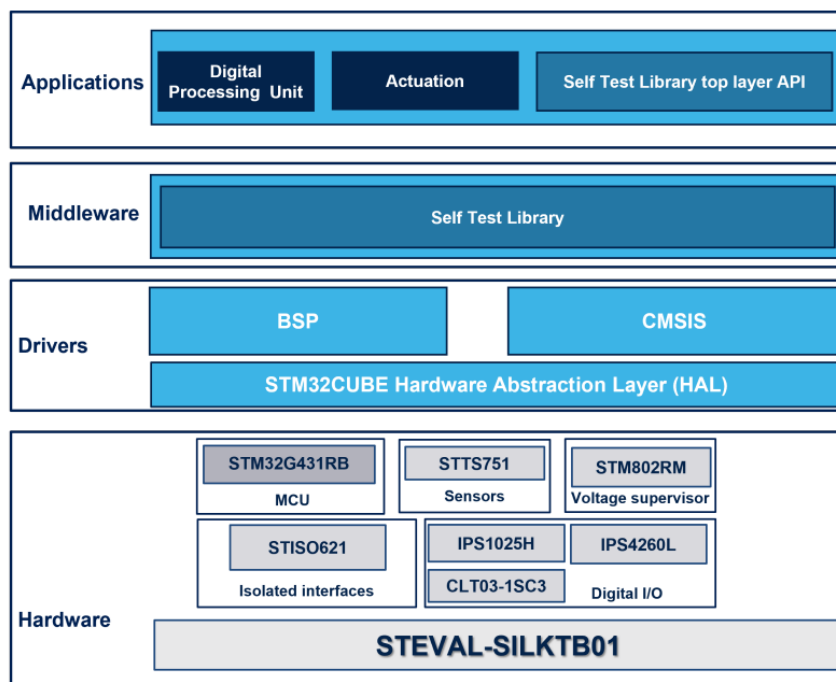


Figure 2. STSW-SILKTB01 architecture



2.1 STEVAL-SILKTA01

This firmware package can support with dedicated APIs integration, real time communication protocol like Canopen and Ethernet/IP; Modbus-RTU protocol is available in demo version .

In addition to the remote communication, has been implemented an internal SPI communication with the other two STEVAL-SILKTB01. The firmware is developed to monitor the digital input signals that can be provided by sensors or by an emergency button, and through dedicated logic gates interact with daughter boards to safely protect the complete system.

The firmware architecture respects the following structure:

- Application folder:
 - Containing the top level routines not strictly related to the hardware architecture
 - User: STM32 configuration routines , interrupt service routines, diagnostic functions, digital input detection and peripherals configuration
- Middleware folder:
 - Containing the low level service routines for protocol implementation
 - STL library: embedded part of the functional safety package to implement the safety functions up to the IEC61508 SIL2/SIL3 level
- Drivers:
 - BSP: board support package drivers, including all the low level APIs used to drive properly all the onboard digital input/output ICs, temperature sensor.
 - The functions covered in this section are:
 - Module reset
 - STEVAL-SILKTB01 board connection
 - Digital output drive and supply
 - Diagnostic and feedback check
 - Safety function execution
 - CMSIS: Cortex® microcontroller software interface standard
 - STM32H7xx_HAL_Library: containing the high level routines related to the peripherals management such as timers, communication, voltage conversion, GPIOs

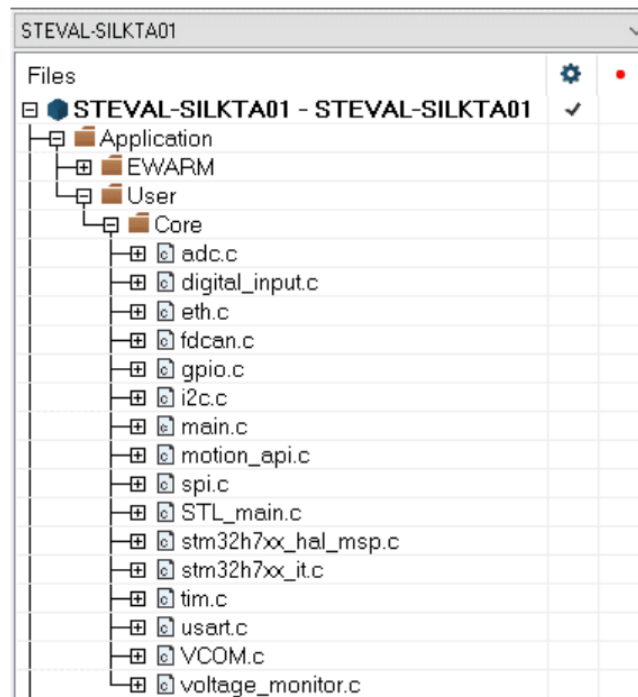
Each layer is split into different groups, which contain all the source files needed for the channel communication management, load driving, diagnostic, and safety function implementation.

2.1.1 Application

This layer consists of a set of routines to manage the user application layers. This set of routines also configures and manages all the peripherals useful for the system management.

As shown in Figure 3. Set of routines for STEVAL-SILKTA01, it embeds the User/Core routines for the peripheral configuration, STL library top level routines, and RS485/VCOM communication management, voltage monitoring for Vbus and LDO.

Figure 3. Set of routines for STEVAL-SILKTA01



2.1.1.1 User

This folder contains a set of routines that provide the user with a basic level of access to interact with the firmware. The user can mainly:

- Configure the STM32 peripherals and GPIOs to run properly the application
- Manage RS485 data transmission to communicate kit status to the upper layer
- Manage the digital input signals to identify the correct functionality of the CLT03-1SC3
- Monitor the several voltage references to detect undervoltage/overvoltage events
- Monitor the board temperature

The main core files are listed below:

- **Adc.c**

This file provides the routines used for ADC configurations; as required by the IEC-61508, to avoid missing information related to voltage monitoring that could cause failure to detect voltage failure, have been used two ADCs, ADC1 and ADC3 that monitor the 24 V and the 3.5 V

- **Digital input.c**

This file includes the routines implemented to detect the presence of the pulsed signals provided by the digital input ICs; the detection is done using two timers TIM1 and TIM2 in input capture mode.

- **Spi.c**

This file provides code for the configuration of the SPI instances. Moreover, there are the functions used to

transmit and receive SPI buffers between the STEVAL-SILKTA01 and the STEVAL-SILKTB01 boards.

- Tim.c

This file provides code for the configuration of the timer used on board. According to the architecture, several timers have been configured as described below:

- TIM1, TIM2 to read the digital input pulsed signals
- TIM6 to start the ADC injected conversion
- TIM16 PWM generation for the voltage supervisor functionality
- Voltage_monitor.c

This source file includes a set of routines used to manage the interrupt service routines related to:

- End of conversion callback routines of ADCx
- Calculation of the real voltage value according to partition factor onboard and converted value. In particular these routines after calculation of the real measured voltage value (this is done for the 24V nominal value) and for the 3.5V, include a control of the converter value by a comparison with a fixed voltages threshold (upper and lower) defined in the voltage monitor. If the measured value is out of the defined range the system is put in safe state.

It is possible to change threshold values within the voltage monitor .file, as shown in Figure 4. Overvoltage and undervoltage.

By default, a range between 22V and 38V is set for the main bus voltage whereas the allowed range for the 3.5V voltage is 3.2V and 3.8V.

Figure 4. Overvoltage and undervoltage

```
#define ADC_RES 65535 //16bit resolution

#define VoltageBUS_PARTITION_FACT 0.052f //24V
#define VoltageBUS_OPL 22 /* undervoltage for main voltage */
#define VoltageBUS_OPH 38 /* overvoltage for main voltage */

#define LDO_MCU_PARTITION_FACT 0.65f // 3.5V
#define LDO_mcu_OPL 3.2 // undervoltage for 3.5V LDO input voltage
#define LDO_mcu_OPH 3.8 // overvoltage for 3.5V LDO input voltage

#define ADC_SKIP 10 // skipped samples for noise filtering
typedef enum
{
    overvoltage = 0x11,
    undervoltage = 0x12,
    voltage_ok = 0x13,
    voltage_unknown = 0x14,
} Voltage_status;

typedef struct
{
    uint32_t conv_value;
    float value;
    float threshold;
    Voltage_status actual_state;
} Data_Detect_TypeDefStruct;
```

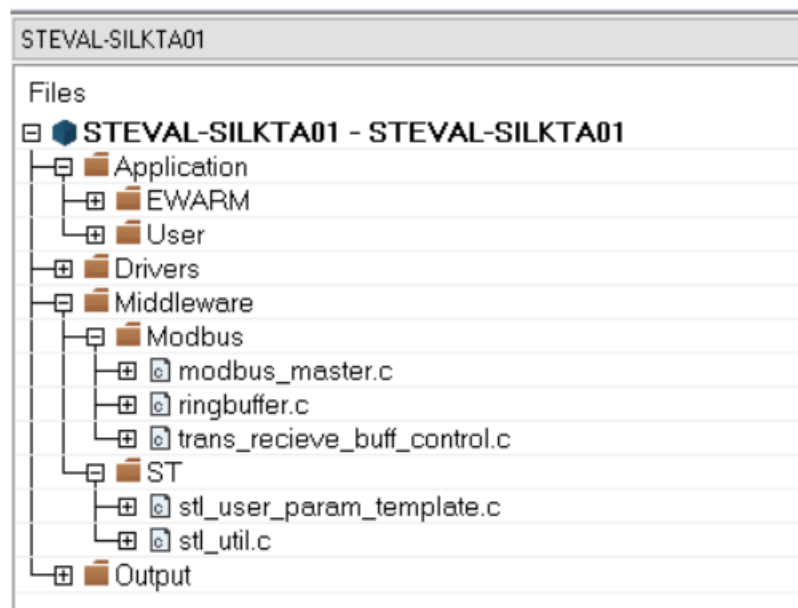
2.1.2 Middleware

This layer contains the safety library APIs for the STM32, in the released package safety library it is not included, it is necessary to sign specific NDA to have it.

Once received library package, integrate it in the Middleware/ST folder.

The middleware layer include free protocol on Modbus to implement a serial communication.

Figure 5. Middleware



2.1.2.1 STL library

This folder contains the API used to address the safety library available in a compiled format.

This API allows running different test modes to check the memory status during the operating condition.

In this application example, the Single_Self_Test has been implemented to check the RAM memory sectors.

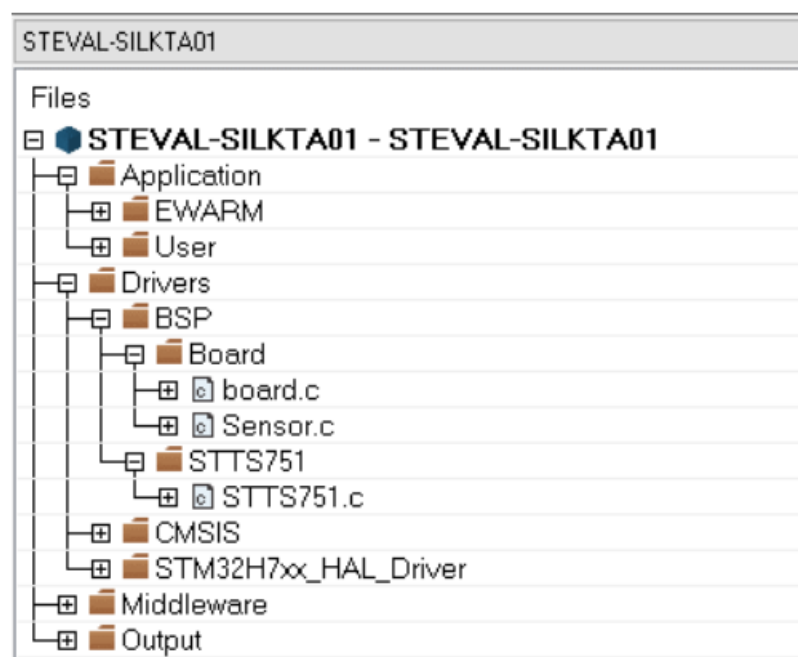
Flash memory and CPU tests are not implemented in this application firmware example. They are supported by the official X-CUBE-STL package.

Note: Safety library is not included in the firmware package, but are supported at application level calling the Single_Self_Test, to include the library in the firmware contact your STMicroelectronics representative to sign a dedicated NDA to have the source files

2.1.3 Drivers

The Drivers layer includes the STM32Cube package libraries (CMSIS and STM32H7xxHAL_Driver) and the BSP routines to handle the digital output I/O ICs, serial communication, voltage supervisor to store in flash the system status in case of fault event on the 3.5 V supply, temperature sensor to monitor microcontroller temperature.

Figure 6. Drivers tree representation



This source file hosts a set of routines used to guarantee the safe output operation in the operating and fault conditions.

Table 1. Board.c routines

| Routine | Parameter | Description |
|---|---|---|
| Void Module_Supply_Drive (Modulex_StructType module, Functional State act_supply_line); | Module: daughter board status parameters; act_supply_line: daughter board supply line status | Enable/disable the supply line for the daughter board |
| Void Module_SafeState (Modulex_StructType module); | Module: daughter board status parameters; | Turn off the digital output and the related supply line |
| Void Module_Output_Drive (Modulex_StructType module, FunctionalState actuation); | ips_output: daughter board status parameters; actuation: daughter board supply line status | Turn off/on the digital output |
| Void Feedback_check (void); | – | Check congruency between inputs and expected outputs for both digital outputs |
| Void Diagnostic_check (void); | – | Check on digital outputs' diag. pins |
| Void Module_Reset (void); | – | Set default condition for digital input and output, state, status |
| Void Module_reset (void); | – | Board variables reset |
| Void Board_Connected (void); | – | Identify which module is connected and assign the module ID (A or B) |
| Void StlSingleTest(void); | – | Routine for safety library test |
| Void SPI_TxRx(void); | – | SPI communication between master and daughter boards |

Sensor.c

Source file containing the main routines used for digital temperature sensor handling, such as sensor:

1. Identification
2. Configuration
3. Temperature reading

Table 2. Sensor.c routines

| Routines | Parameters | Description |
|--------------------------|------------|---------------------------------|
| int8_t Sensor_Init(void) | None | Temperature sensor init routine |

- STTS751.c

Source file containing low level routines for digital sensor handling.

Table 3. STTS751.c routines

| Routine | Parameter | Description |
|--|--|-----------------------------------|
| uint8_t STTS751_ID (STTS751_Param_StructTypeDef* rx_data) | Manufacturer ID code | Temperature sensor identification |
| int8_t STTS751_Configuration(uint8_t IC_Addr,uint8_t* conf_value,uint8_t data_len) | I2C address Configuration register value Data length | Configuration routine |
| int8_t STTS751_Get_Temperature_HighByte(uint8_t IC_Addr) | I2C address | MSB Temperature registers reading |
| int8_t STTS751_Get_Temperature_LowByte(uint8_t IC_Addr) | I2C address | LSB Temperature registers reading |
| int8_t STTS751_Get_Status(uint8_t IC_Addr) | I2C address | Temperature sensor status |
| void Temperature_calc(void) | None | Temperature value calculation |

Figure 7. STTS751 registers

```

/** @defgroup STTS751_Register_Address STTS751 Register Address
 *  @brief STTS751 Register Address
 */

#define STTS751_ADDRESS            0x70
#define STTS751_PRODUCT_ID        0x00
#define STTS751_MANUFACTURER_ID   0x53
#define PRODUCT_VERIFIED           0x88

/** REGISTERS ADDRESS */

#define STATUS_REGISTER             0x01
#define CONFIGURATION              0x03
#define CONVERSION_RATE            0x04
#define SMBUS_TIMEOUT_ENABLE       0x22
#define PRODUCT_REGISTER           0xFD
#define MANUFACTURER_ID            0xFE
#define PRODUCT_REV_ID             0xFF

/* MEASURED TEMPERATURE REGISTER*/
#define TEMPERATURE_HIGH_BYTE      0x00
#define TEMPERATURE_LOW_BYTE       0x02

/* SET TEMPERATURE VALUE FOR EVENT ACTIVATION*/
#define TEMPERATURE_HIGH_LIMIT_HIGH 0x05
#define TEMPERATURE_HIGH_LIMIT_LOW  0x06
#define TEMPERATURE_LOW_LIMIT_HIGH  0x07
#define TEMPERATURE_LOW_LIMIT_LOW   0x08

#define ONE_SHOT_REGISTER          0x0F

/* SET TEMPERATURE VALUE FOR Add/THERM*/
#define THERM_LIMIT                0x20
#define THERM_LIMIT_HYST           0x21

```

2.2 STEVAL-SILKTB01

This firmware package is able to manage at the same time the communication with the STEVAL-SILKTA01; monitor the digital input signals that can be provided by sensors or by an emergency button, and through dedicated logic gates, interact with main boards to safely protect the complete system.

The firmware architecture respect the following structure:

- Application folder:

Containing the top level routines not strictly related to the hardware architecture

- User: STM32 configuration, interrupt service routines, diagnostic functions, digital input detection and

peripherals configuration

- Middleware folder:

- STL library: embedded part of the functional safety package to implement the safety functions up to the IEC61508 SIL2/SIL3 level

- Drivers:

Including:

- BSP: board support package drivers, including all the low level APIs used to drive properly all the onboard digital input/output ICs, temperature sensor.

The functions covered in this section are:

- IPSs driving and diagnostic

- Safety function execution

- Board ID detection

- Digital inputs monitoring

- CMSIS: Cortex® microcontroller software interface standard

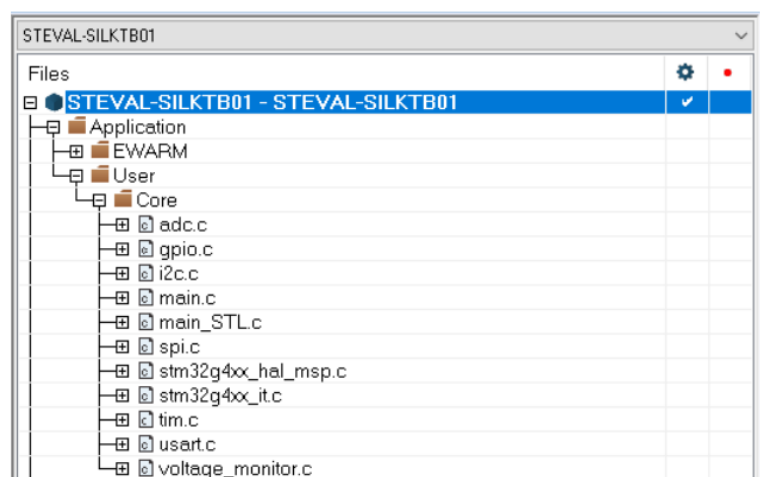
- STM32G4xx_HAL_Library: containing the low level routines related to the peripherals management such as timers, communication, voltage conversion, GPIOs Each layer is split into different groups, which contain all the source files needed for the load driving, diagnostic, and safety function implementation.

2.2.1 Application

This layer consists of a set of routines to manage the user application layers. This set of routines also configures and manages all the peripherals useful for the system management.

As shown in Figure 8. Set of routines for STEVAL-SILKTB01, it embeds the User/Core routines for the peripheral configuration, STL library top level routines and voltage detection for 24 V supply and LDO.

Figure 8. Set of routines for STEVAL-SILKTB01



2.2.1.1 User

This folder contains a set of routines to provide the user with a first access level to interact with the firmware. The user can mainly:

- Configure the STM32 peripherals and GPIOs to run properly the application
- Configure the SPI field to be transmitted
- Monitor the undervoltage/overvoltage events
- Monitor the board temperature

The main core files are listed below:

- **Adc.c**

This file provides the routines used for ADC configurations; as required by the IEC-61508, to avoid missing information related to voltage monitoring that could cause failure to detect voltage failure, have been used two ADCs, ADC1 and ADC3 that monitor the 24 V and the 3.5 V

- **Spi.c**

This file provides code for the configuration of the SPI instances. Moreover, there are the functions used to transmit and receive SPI buffers between the STEVAL-SILKTA01 and the STEVAL-SILKTB01 boards.

- **Tim.c**

This file provides code for the configuration of the timer used on board. According to the architecture, several timers have been configured as described below:

- TIM1, TIM3 to read the digital input pulsed signals
- TIM6 to start the ADC injected conversion
- TIM17 PWM generation for the voltage supervisor functionality

- **Voltage_monitor.c**

This source file includes the routines used to read through interrupt service routine, the ADC data registers, and the routine for the calculation of the real value for 24V and 3.5V monitored by the ADC.

In the calculation routine, the real value is obtained considering the partition factor of each voltage monitoring network, reported in `voltage_monitor.h`; once the real value is calculated, it is compared with the threshold limit fixed in the same file.

It is possible to change threshold values by acting within the “`voltage_monitor.h`” file, as shown in Figure 9. Voltage thresholds settings.

By default, a range between 22V and 38V is set for the main bus voltage whereas the allowed range for the 3.5V voltage is 3.2V and 3.8V

Figure 9. Voltage thresholds settings

```
#define VoltageBUS_PARTITION_FACT 0.052f //24V
#define VoltageBUS_OPL 22 /* undervoltage for main voltage */
#define VoltageBUS_OPH 38 /* overvoltage for main voltage */

#define LDO_MCU_PARTITION_FACT 0.65f // 3.5V

#define LDO_mcu_OPL 3.2 // undervoltage for 3.5V LDO input voltage
#define LDO_mcu_OPH 3.8 // overvoltage for 3.5V LDO input voltage

#define ADC_SKIP 10 // skipped samples for noise filtering
typedef enum
{
    overvoltage = 0x11,
    undervoltage = 0x12,
    voltage_ok = 0x13,
    voltage_unknown = 0x14,
}Voltage_status;

typedef struct
{
    uint32_t conv_value;
    float value;
    float threshold;
    Voltage_status actual_state;
} Data_Detect_TypedefStruct;
```

2.2.2 Middleware

This folder contains the API used to address the safety library available in a compiled format inside the `Middleware\ST\STM32_Safety_STL\Lib` folder.

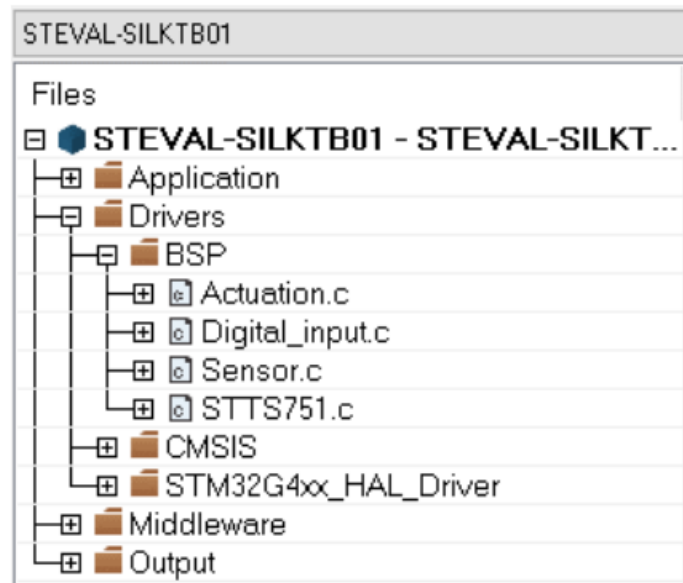
This API (not included in the released package) allows running different test modes to check the memory status during the operating condition.

They are supported by the official X-CUBE-STL package.

2.2.3 Drivers

The Drivers layer includes the STM32Cube package libraries (CMSIS and STM32G4xxHAL_Driver) and the BSP routines to handle the digital I/O ICs, and temperature sensor.

Figure 10. Drivers tree representation



- **Actuation.c**

This source file hosts a set of routines used to guarantee the safe output operation in the operating and fault conditions.

Table 4. Actuation.c routines

| Routines | Parameters | Description |
|--|--------------------------|--|
| Void HS_drv(IPS_State state); | State: ON/OFF | Enable/disable the HS IPS |
| Void LS_drv(IPS_State state); | State: ON/OFF | Enable/disable the LS IPS |
| Void Load_safe_state (void); | – | Turn off the output of both IPSs and the HS IPS input as well |
| Void Load_ON (void); | – | Turn on both the IPSs |
| Void Diag_drv (void); | – | Check on digital outputs' diag. pins |
| Void Supply_drv (FunctionalState var); | var: ENABLE/ DIS ABLE | Turn on/off the supply line for the digital outputs |
| Void IPS_functionality_check (void); | – | Check congruency between inputs and expected outputs for both digital outputs |
| Void Board_start (void); | – | Digital outputs activation |
| Void Board_ID (void); | – | Identify at which slot the daughter board is connected and assign the module ID (A or B) |

- **Digital_input.c**

signals provided by the digital input ICs; the detection is done using two timers TIM1 and TIM3 in input capture

mode.

- **Sensor.c**

Source file containing the main routines used for digital temperature sensor handling, such as sensor:

1. identification

2. configuration

Table 5. Sensor.c routines

| Routines | Parameters | Description |
|--------------------------|------------|---------------------------------|
| int8_t Sensor_Init(void) | None | Temperature sensor init routine |

- **STTS751.c**

Source file containing low level routines for digital sensor handling

Table 6. STTS751.c routines

| Routines | Parameters | Description |
|--|--|-----------------------------------|
| uint8_t STTS751_ID (STTS751_Param_StructTypeDef* rx_data) | Manufacturer ID code | Temperature sensor identification |
| int8_t STTS751_Configuration(uint8_t IC_Addr,uint8_t* conf value,uint8_t data_len) | I2C address Configuration register value Data length | Configuration routine |
| int8_t STTS751_Get_Temperature_HighByte(uint8_t IC_Addr) | I2C address | MSB Temperature registers reading |
| int8_t STTS751_Get_Temperature_LowByte(uint8_t IC_Addr) | I2C address | LSB Temperature registers reading |
| int8_t STTS751_Get_Status(uint8_t IC_Addr) | I2C address | Temperature sensor status |
| void Temperature_calc(void) | None | Temperature value calculation |

How functional safety works at system level

The system evaluation solution, supported by this application firmware, is equipped with a set of routines that are able to manage the hardware architecture defined in consideration with the IEC61508 standard. These routines are designed to protect the solution against failure events by driving the output into a safe state condition. The standard requirements for a SIL3 solution are mainly oriented to:

- Redundancy
- Diagnostic coverage
- Feedback signalization
- System protections

All these features have been covered at hardware level with dedicated hardware circuitry and components that are managed by the STM32 microcontroller onboard. Both boards are programmed to continuously monitor the supply voltage references of 24V and 3.5V, the temperature of the data processing, and to use a PWM signal to keep the

supervisor functionality running in case of failure. If the diagnostic check is successful and the digital input check provides positive results, the output of each STEVAL-SILKTB01 board is automatically activated to actuate the connected system. Since the digital input check is performed not only on the STEVAL-SILKTB01 boards but also on the STEVAL-SILKTA01 board, the output activation is carried out by the microcontrollers STM32H743ZG and STM32G431RB, through an AND connection of the driving signals managed by the MCUs.

During the driving if a fault event is detected (such as voltage monitoring issue, temperature monitoring or overcurrent/overtemperature on output) on at least one board, the system is driven in a safe state if at least one MCU executes the safety functions.

The behavior of the safety function execution, consist of driving the daughter boards de-energizing the load, by a switch-off of both IPSs devices and opening supply voltage path for the IPSs ICs turning off the power MOSFET at least on one board.

Revision history

Table 7. Document revision history

| Date | Revision | Changes |
|-------------|----------|------------------|
| 12-Jun-2024 | 1 | Initial release. |

IMPORTANT NOTICE – 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 acknowledgment.

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.

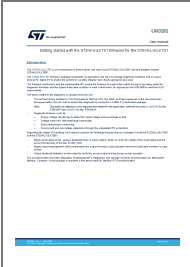
Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. 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.

© 2024 STMicroelectronics – All rights reserved

Documents / Resources

| | |
|---|---|
|  | ST STSW-SILKT01 Firmware for STEVAL-SILKT01 [pdf] User Manual STEVAL-SILKTA01, STEVAL-SILKTB01, STSW-SILKT01 Firmware for STEVAL-SILKT01, STS W-SILKT01, Firmware for STEVAL-SILKT01, STEVAL-SILKT01 |
|---|---|

References

- [User Manual](#)

