

Getting started with the STSW-GLA001V1 software package for the STEVAL-GLA001V1 evaluation board based on STM32Cube

Introduction

The STSW-GLA001V1 firmware is designed to be used with the STEVAL-GLA001V1 insulated AC switch control evaluation board to evaluate its embedded TRIACs and AC switches, implementing different control modes.

The firmware is developed from the STM32Cube embedded software libraries for STM32F0 series using the IAR Embedded Workbench for ARM.

The firmware also lets you interact with the STEVAL-GLA001V1 hardware user interface and allows the configuration of some parameters using a PC user interface.

Thanks to the dynamic storage of the configuration parameters in a dedicated section of the Flash memory embedded in the STM32 microcontroller, you can change any parameter any time without the need of compiling and programming the MCU via third-party tools.

Contents

1	STSW-GLA001V1 firmware structure.....	5
1.1	Overview	5
1.2	Architecture	5
1.3	Parameters.....	6
1.4	Variables	6
1.5	Parameters and variables stored in the MCU Flash memory	6
2	System setup guide.....	9
2.1	Hardware configuration and board setup	9
2.1.1	Programming the NUCLEO-F030R8 board using a binary file	9
2.1.2	Programming the NUCLEO-F030R8 board using a preconfigured source project.....	12
2.2	STEVAL-GLA001V1 operating modes	13
2.2.1	STSW-GLA001V1 control mode detection.....	16
2.2.2	ZVS signal synchronization with Vmains zero voltage crossing	16
2.2.3	ON/OFF pulse basic/timer control mode	18
2.2.4	Phase control mode.....	22
2.2.5	ZVS signal delay measurement	24
2.2.6	User interface	25
3	Revision history	32

List of tables

Table 1: Stored parameters and variables.....	7
Table 2: Factory values for td_buffer	8
Table 3: State machine fault codes	15
Table 4: STEVAL-GLA001V1 operating modes: analog input and digital level.....	16
Table 5: STEVAL–GLA001V1 evaluation board: user interface command list	30
Table 6: Document revision history	32

List of figures

Figure 1: System architecture	5
Figure 2: STEVAL-GLA001V1 evaluation board connected to NUCLEO-F030R8 board via X-NUCLEO-IHM09M1 expansion board.....	9
Figure 3: NUCLEO device on Windows OS	10
Figure 4: ST-LINK utility tool window after bin file opening	11
Figure 5: ST-LINK utility tool Target menu and Program command	11
Figure 6: ST-LINK utility tool download window	12
Figure 7: STSW-GLA001V1 firmware – Project workspace on IAR™	12
Figure 8: STSW-GLA001V1 firmware – IAR workspace configuration selection	13
Figure 9: STSW-GLA001V1 firmware – IAR workspace Make icon.....	13
Figure 10: STSW-GLA001V1 firmware – IAR workspace Download and Debug icon.....	13
Figure 11: STSW-GLA001V1 state machine	14
Figure 12: ZVS delay principle.....	17
Figure 13: ON/OFF control in basic/timer mode with pulse gate command timing diagram	18
Figure 14: ON/OFF control in basic/timer mode with pulse gate command timing diagram with $td_{zvs} > 0$	19
Figure 15: ON/OFF control in basic/timer mode with pulse gate command timing diagram with $td_{zvs} < 0$	20
Figure 16: ON/OFF basic mode control with DC gate command timing diagram	21
Figure 17: ON/OFF timer mode control with DC gate command timing diagram	21
Figure 18: Phase control timing diagram	22
Figure 19: Phase control timing diagram with $td_{zvs} > 0$	23
Figure 20: Phase control timing diagram with $td_{zvs} < 0$	24
Figure 21: ZVS delay measurement - oscilloscope capture	25
Figure 22: HyperTerminal: new connection window	26
Figure 23: HyperTerminal: port settings	26
Figure 24: Tera Term: new connection menu.....	27
Figure 25: Tera Term: setup menu	28
Figure 26: Tera Term: setup menu parameters.....	28
Figure 27: Tera Term: command list.....	29
Figure 28: Tera Term: command list sample view (white background and black font)	29

1 STSW-GLA001V1 firmware structure

1.1 Overview

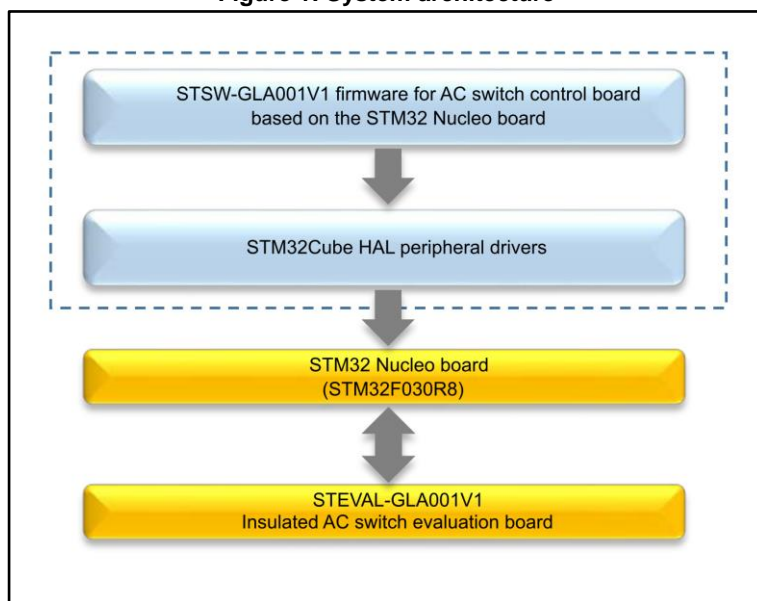
The STSW-GLA001V1 firmware features:

- Based on the STM32Cube embedded software libraries
- Support for the NUCLEO-F030R8 board
- Configuration and independent control of three TRIACs and AC switches embedded in the STEVAL-GLA001V1 evaluation board:
 - On/Off control (DC and pulse modes)
 - Phase control mode
 - Timer mode
- Hardware user interface via push-buttons and switches
- Automatic mains frequency detection at switch-on
- PC user interface through a serial terminal as Windows HyperTerminal or Tera Term to:
 - configure the firmware parameters
 - monitor the main firmware variables
 - control command lines (i.e., start and stop commands)

1.2 Architecture

The STSW-GLA001V1 firmware^a for the STEVAL-GLA001V1 evaluation board is based on the STM32Cube HAL peripheral drivers and supports different operating modes for AC switch control.

Figure 1: System architecture



^a Contact your local STMicroelectronics sales office to obtain the source code

The STSW-GLA001V1 firmware consists of a list of functions that allow the user to interface with the code variables and functions which manage and implement the different evaluation board operating modes^a:

- **AC_Switch_Control_Lib_Interface_Methods.c** file: contains the interfacing functions used by the firmware
- **User_Methods.c** file: contains the methods the user has to refer to for writing his own program
- **AC_Switch_Control_Lib.c** file: contains all the functions that implement the operating modes
- **AC_Switch_Control_Tasks.c** file: contains the `MediumFrequencyTask()` and the `LowFrequencyTask()` functions which calls the operating mode functions

The `MediumFrequencyTask()` function is executed at the SYSTICK frequency (i.e. 1 kHz).

The `LowFrequencyTask()` function is executed at the mains frequency (50/60 Hz), if the ZVS detection circuit is used; otherwise, the `LowFrequencyTask()` function is not executed.

1.3 Parameters

The STSW-GLA001V1 firmware contains two types of parameters defined in the **AC_Switch_Control_param.h** file:

- **Default parameters:** used for the firmware variable initialization/reset.
- **Firmware configuration parameters:** used to set the firmware configuration parameters (i.e., the minimum and maximum values for the system variables).

When the user wants to change a parameter value, the firmware has to be compiled and downloaded in the MCU Flash memory using the IAR Embedded Workbench IDE.

1.4 Variables

The AC switch control firmware variables are defined in the **AC_Switch_Control_Vars_t** structure contained in the **AC_Switch_Control_Lib.h** file.

You may not directly access this structure, but you can use the interfacing functions included in the user project under the **AC_Switch_Control_Lib_Interface_Tasks.h** header file.

1.5 Parameters and variables stored in the MCU Flash memory

The firmware stores the set of variables and parameters listed in [Table 1: "Stored parameters and variables"](#) in the MCU Flash memory whenever one of them is changed and the current machine state is **GATE_CONTROL_AVAILABLE_IDLE**.

Then, the following two cases should be considered:

- The data is changed during the **GATE_CONTROL_AVAILABLE_IDLE** state: it causes the data set to be saved in the Flash memory. During this state, data modification is taken into account in a very short time. Data storage process starts approximately 1 ms after the end of data modification and ends after about 25 ms

^a Refer to [Section 2.2: "STEWAL-GLA001V1 operating modes"](#) for further details.



During this process, do not switch the system off to avoid data loss.

- The data is changed in a state different from **GATE_CONTROL_AVAILABLE_IDLE**: in this case, the store request is activated and the storage process is only performed, when the state is **GATE_CONTROL_AVAILABLE_IDLE**.



To avoid data loss, do not switch the system off before reaching the **GATE_CONTROL_AVAILABLE_IDLE** state.

The stored values are reloaded at board restart.

If errors occur during the erasing/writing process, the variables stored in the Flash memory are corrupted and the firmware will restore them to the factory values on the next board restart.

The user can also manually store data by typing the command `store(data)` in the serial terminal (refer to [Section 2.2.6: "User interface"](#) for further details).



To avoid data loss, call this function before switching the board off.

Table 1: Stored parameters and variables

Description	Name	Unit	Range	Factory value
ZVS available	zvs_hw	N/A	0 or 1	1
Push button pressure delay	PBTlow	ms	100 to 500	100
ZVS_delay	td_ZVS	µs	–MainsPeriod/4 to MainsPeriod/4	0
Pulse time	tp	µs	10 to MainsPeriod/2	5000
Step for pulse time changing after a tp+ or tp- push button pressure	tp_step	µs	10 to 1000	1000
Number of steps for soft-start and soft-stop	soft_start_stop_n_step	N/A	5 to 100	10
On-time (Timer mode)	OnTime	s	0 to 10000	5
Step for on-time setting	OnTime_step	s	1 to 10	1
Margin before next cycle	ZCS_margin	µs	100 to 1000	100
Number of elements for td buffer	N_td	N/A	5 to 100	20
Minimum td delay	td_min	µs	0 to (td_max-1)	0

Description	Name	Unit	Range	Factory value
Maximum td delay	td_max	μs	(td_min+1) to MainsPeriod/2-ZCS_margin	MainsPeriod/2-ZCS_margin
Turn-on delay buffer	td_buffer[N_td]	μs	td_min to td_max	Refer to Table 2: "Factory values for td_buffer"

Table 2: Factory values for td_buffer

1	2	3	4	...	20
MainsPeriod/40	MainsPeriod/20	3·MainsPeriod/40	MainsPeriod/10	...	MainsPeriod/2

2 System setup guide

2.1 Hardware configuration and board setup

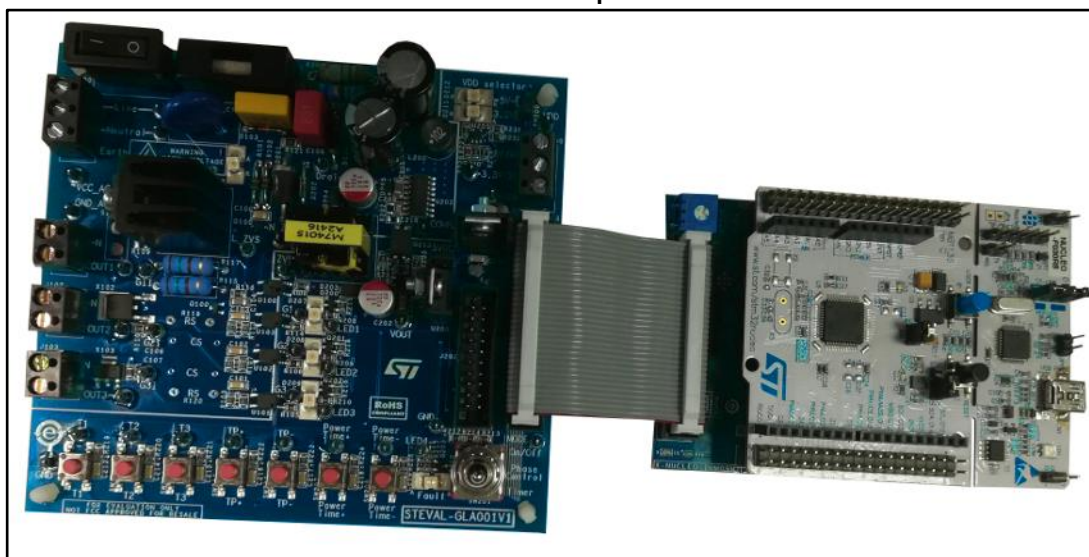
To use the STSW-GLA001V1 firmware, you need a NUCLEO-F030R8 board and a STEVAL-GLA001V1 evaluation board.

The two boards are connected through the X-NUCLEO-IHM09M1 expansion board, which acts as an adapter between a standard ST morpho connector and a 34-pin connector usually mounted on ST evaluation boards.^a



The user has to ensure that SB16 and the SB50 jumpers on the NUCLEO-F030R8 board are closed.

Figure 2: STEVAL-GLA001V1 evaluation board connected to NUCLEO-F030R8 board via X-NUCLEO-IHM09M1 expansion board



The NUCLEO-F030R8 board can be programmed using a binary file or a pre-configured project.

2.1.1 Programming the NUCLEO-F030R8 board using a binary file

To correctly program the NUCLEO-F030R8 board using a binary file, the user can follow the drag and drop procedure or use the STM32 ST-LINK utility tool.

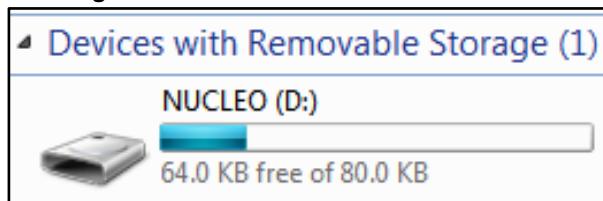
2.1.1.1 Drag and drop procedure

- 1 Install ST-LINK/V2 drivers from www.st.com (usually this is executed automatically if an Internet connection is available).
- 2 On the NUCLEO-F030R8 board put the JP5 in U5V position.

^a For further details, refer to the STEVAL-GLA001V1 evaluation board user manual on www.st.com.

- 3 Plug the NUCLEO-F030R8 board USB connector (CN1) to the PC via a USB type A to mini-B cable.
If the ST-LINK drivers are correctly installed, the PC recognizes it as an external memory device called NUCLEO or similar.
- 4 Download the firmware binary file for the NUCLEO-F030R8 board from www.st.com.
- 5 Drag and drop it into the NUCLEO device listed in the disk driver list^a.

Figure 3: NUCLEO device on Windows OS



- 6 Wait until the flashing operation is complete.

2.1.1.2 STM32 ST-LINK utility tool

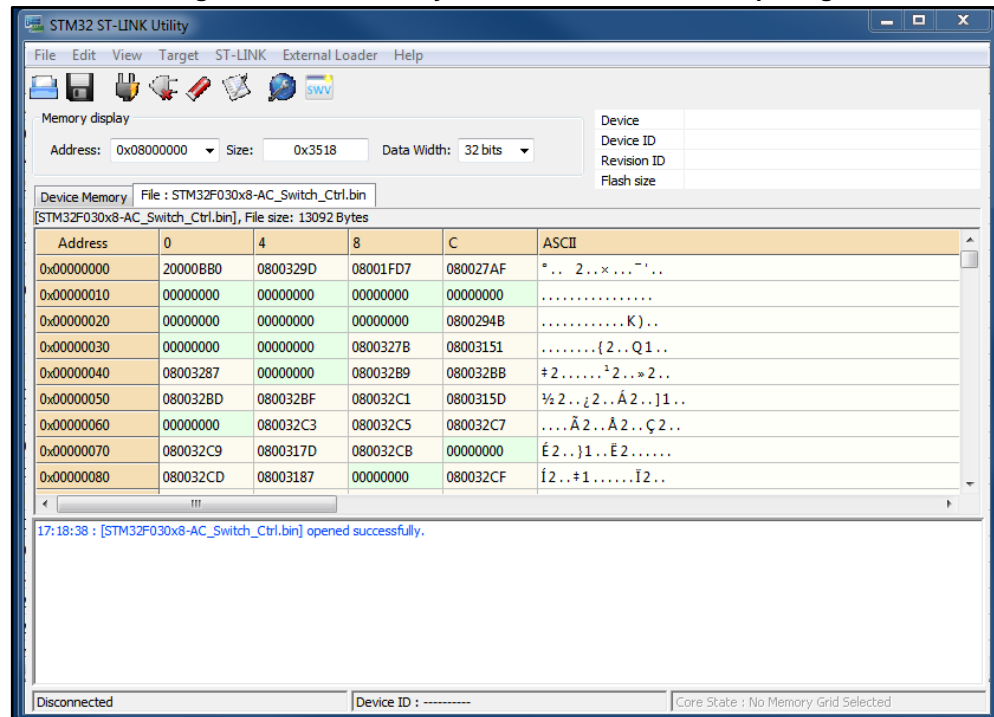
- 1 Download the STM32 ST-LINK Utility tool from www.st.com and install it.
- 2 Open the tool.
- 3 Plug the NUCLEO-F030R8 board to the PC via the board USB connector (CN1) through a USB type A to mini-B cable.
- 4 Ensure the embedded ST-LINK/V2 is configured to program on the NUCLEO-F030R8 board^b.
- 5 Click on **File**→**Open File**
- 6 Select the binary file to use.

^a Located under **Devices with Removable Storage** in the Windows OS computer folder.

^b Both CN2 jumpers must be connected.

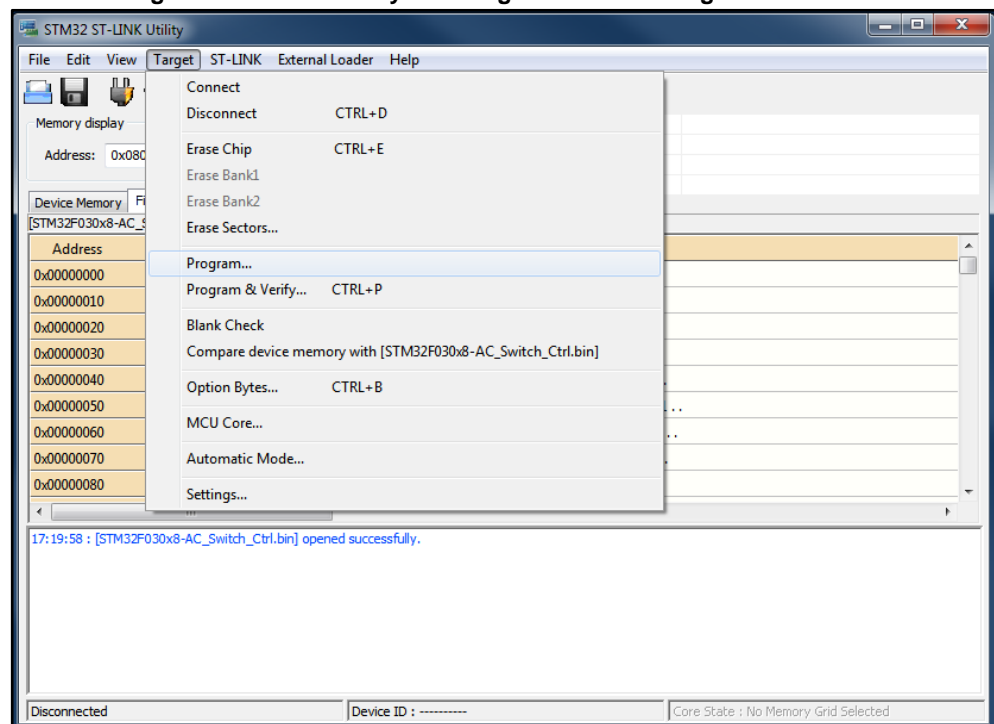
The following window appears.

Figure 4: ST-LINK utility tool window after bin file opening



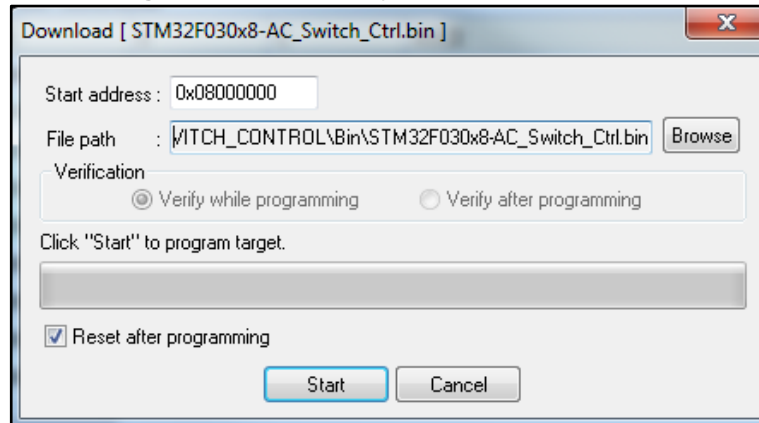
- 7 Click on the **Program** command in the **Target** menu to download the code in the STM32 MCU as shown below.

Figure 5: ST-LINK utility tool Target menu and Program command



- 8 Click on the **Start** button in the window below and wait until the flashing is complete.

Figure 6: ST-LINK utility tool download window



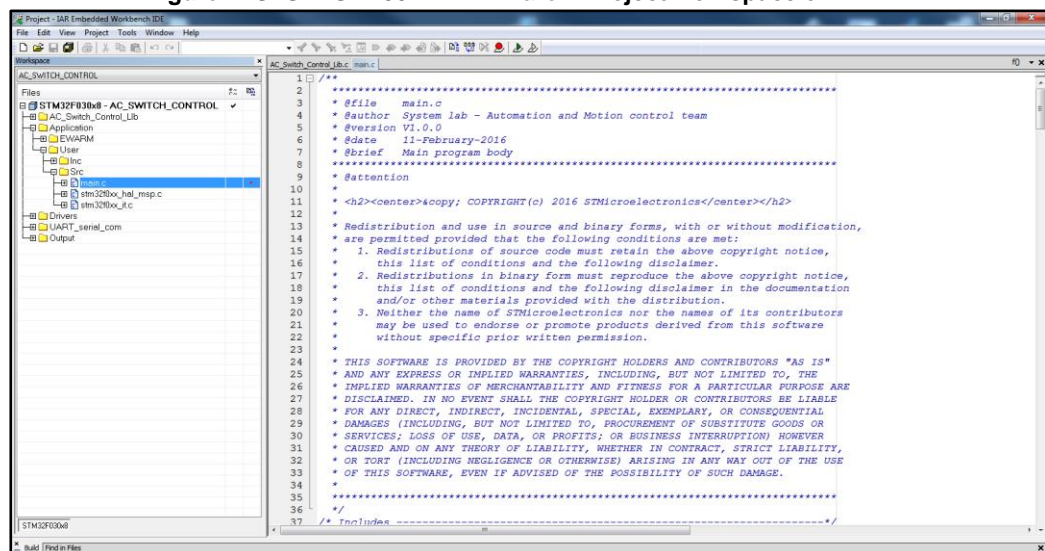
- 9 Close the STM32 ST-LINK utility tool.

2.1.2 Programming the NUCLEO-F030R8 board using a preconfigured source project

The STSW-GLA001V1 firmware is based on STM32Cube and is provided for the IAR™ IDE tool. The workspace is located under **UserProjectFolder\Projects\AC Switch Control\STM32F030x8\EWARM**.

- 1 Open the IAR **Project.eww** workspace file for STM32F0x family, located in the above mentioned workspace.^a

Figure 7: STSW-GLA001V1 firmware – Project workspace on IAR™



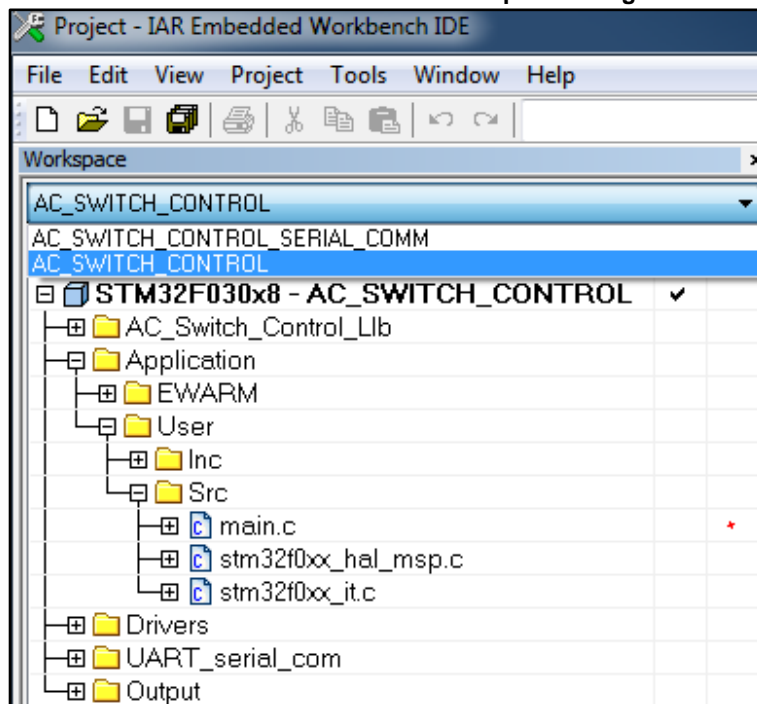
The IAR workspace contains two configurations (see [Figure 8: "STSW-GLA001V1 firmware – IAR workspace configuration selection"](#)):

- **AC_SWITCH_CONTROL_SERIAL_COMM**: to use the serial user interface based on a RS232 serial communication.

^a Contact your local STMicroelectronics sales office to obtain the source code.

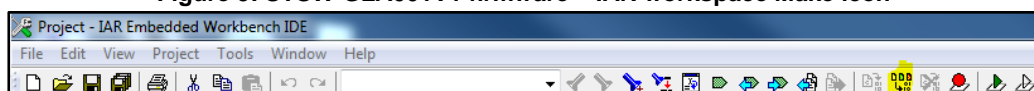
- **AC_SWITCH_CONTROL**: if the user does not want to use the serial communication feature.

Figure 8: STSW-GLA001V1 firmware – IAR workspace configuration selection



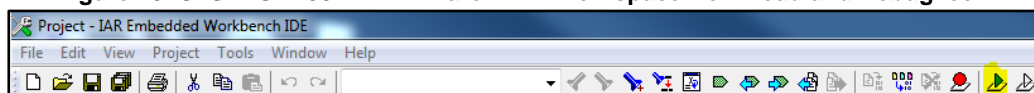
- 2 Compile the active project using the **Make** command from the **Project** menu or pressing F7 on the PC keyboard or clicking on the highlighted icon shown in the following figure.

Figure 9: STSW-GLA001V1 firmware – IAR workspace Make icon



- 3 If the compiling finishes without errors, click on the highlighted icon shown below or go to the **Project** menu to download the project in the microcontroller Flash memory.

Figure 10: STSW-GLA001V1 firmware – IAR workspace Download and Debug icon



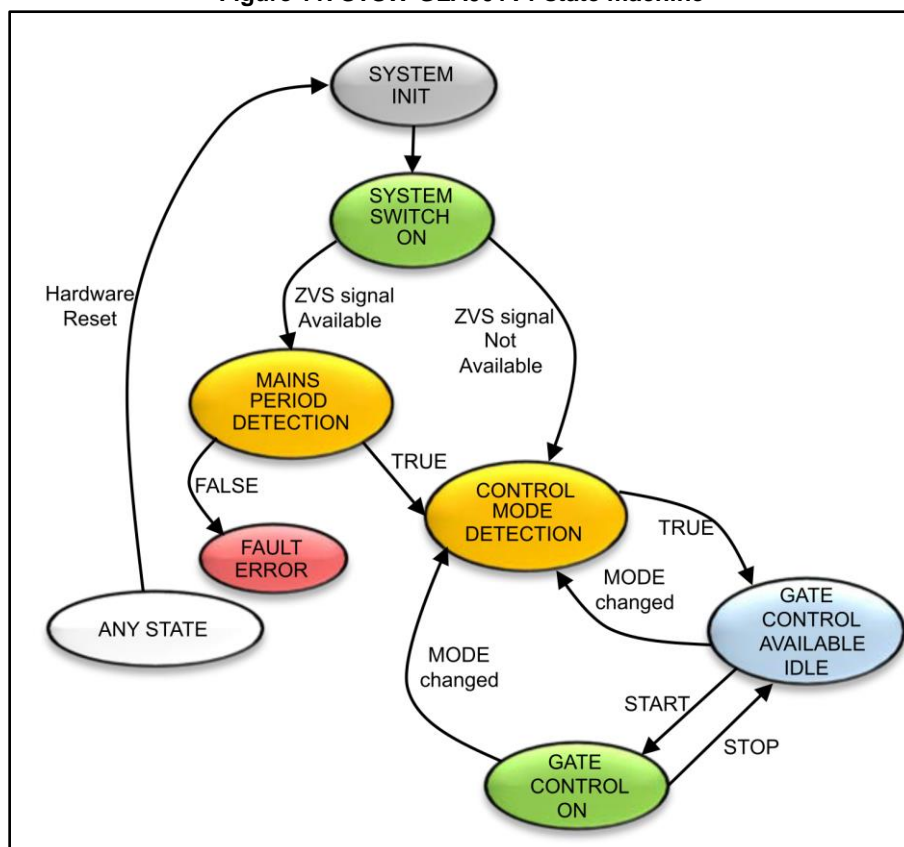
- 4 To download without debugging, open the **Download** sub-menu (in the **Project** menu) and select the **Download active application** command.

2.2 STEVAL-GLA001V1 operating modes

The STEVAL-GLA001V1^a operating modes are managed by the STSW-GLA001V1 firmware via the state machine.

^a Refer to the STEVAL-GLA001V1 evaluation board user manual for details on the board and how to use it.

Figure 11: STSW-GLA001V1 state machine



The different states are:

- **SYSTEM_INIT**: initializes all the firmware variables and parameters used as default values, at board power on or after MCU hardware reset.
- **SYSTEM_SWITCH_ON**: checks if the ZVS detection circuit is available and/or used and sets the next state as **MAINS_PERIOD_DETECTION** if the ZVS signal is available or as **CONTROL_MODE_DETECTION** if the ZVS signal is not available.
- **MAINS_PERIOD_DETECTION**: measures the time elapsed between two ZVS signal rising edges and assigns the **MainsPeriod** value as follows:
 - If **MainsPeriod** is in the 16.50 - 16.70 ms range, **MainsPeriod** = 16.60 ms (**MainsFrequency** = 60 Hz).
 - If **MainsPeriod** is in the 19.90 - 20.10 ms range, **MainsPeriod** = 20.00 ms (**MainsFrequency** = 50 Hz).
 - If the firmware consecutively measures one hundred of **MainsPeriod** values out of the valid range (16.50 and 20.10 ms), the state machine is moved to the **FAULT_ERROR** state. In this case the **Fault_Code** is set to **MAINS_PERIOD_DET_ERROR**, the Fault status flag is set to **FAULT_TRUE** and the **MainsPeriod** value is set to zero. To restart the **MainsPeriod** detection routine, the user has to reset the Nucleo board by clicking on the Reset button (B2).
 - If the firmware measures a sufficient number of **MainsPeriod** values in the valid range, it checks if one of the conditions at points a or b is true. If this does not happen, the **MainsPeriod** detection restarts until case a or b are **TRUE** (and in this case the state machine is moved to **CONTROL_MODE_DETECTION**), or until case c is **TRUE**.

- e. If the ZVS signal input is not present when `zvs_hw = ENABLE`, the MainsPeriod detection algorithm restart as soon as the ZVS signal is available.
- **CONTROL_MODE_DETECTION:** checks the value of the analog selector called MODE and then sets the **GATE_CONTROL_AVAILABLE_IDLE** as next state and the related detected operating mode. If any reading error of MODE value occurs, the **Fault_Code** is set as **CONTROL_MODE_DET_ERROR**, the **Fault_Status** is set as **FAULT_TRUE** and the firmware checks the MODE value until a valid operating mode is detected. During this operation, the active state remains **CONTROL_MODE_DETECTION**.
 - **GATE_CONTROL_AVAILABLE_IDLE:** indicates the system is ready for the TRIAC control. When at least one of the three Tx buttons is pressed, the state machine is moved to the **GATE_CONTROL_ON** state. If, during this idle state, the MODE selector position is changed, the firmware executes a partial reset (only some variables are set to their default values) and the **CONTROL_MODE_DETECTION** state is set as next state.
 - **GATE_CONTROL_ON:** executes the operating mode operations selected by the user: so, at least one of three TRIAC is activated. If during this active state, the MODE selector position is changed, the system executes a partial reset (only some variables are set to their default values) and it goes back to the **CONTROL_MODE_DETECTION** state.
 - **ANY_STATE** is not a real state but it indicates (see [Figure 11: "STSW-GLA001V1 state machine"](#)) any state of the state machine, as listed above. The Nucleo board reset or a falling in the MCU power supply can occur at any time (then during any state) causing a hardware reset and the next state after this reset operation is the **SYSTEM_INIT** state.

Table 3: State machine fault codes

Fault code	Name	Description
0	NO_FAULT_ERROR	No error occurred
1	MAINS_PERIOD_DET_ERROR	The MainsPeriod detected by the firmware is outside the valid range
2	CONTROL_MODE_DET_ERROR	MODE selector value detected by the firmware is outside the three control mode valid ranges. This error also occurs when <code>zvs_hw = DISABLE</code> and the board is not powered on
3	PHASE_CONTROL_MODE_NOT_AVAILABLE	This error occurs when the user moves the MODE selector to the Phase Control Mode position and <code>zvs_hw = DISABLE</code> (ZVS circuit not used or unavailable)
4	ZVS_SIGN_ERROR	This error indicates the ZVS signal absence when <code>zvs_hw = ENABLE</code> . The possible causes are: ZVS circuit is not working or the board is not correctly powered on
5	FLASH_WRITING_ERROR	Some error occurs during the Flash memory erasing before the writing operation

2.2.1 STSW-GLA001V1 control mode detection

In the `AC_Switch_Control_Tasks.c` file, the `MediumFrequencyTasks()` function calls the `AC_Switch_Ctrl_MODE_Selector_Reading()` method, if the actual state is not `SYSTEM_INIT`, `SYSTEM_SWITCH_ON`, `MAINS_PERIOD_DETECTION` and `FAULT_ERROR`.

The `AC_Switch_Ctrl_MODE_Selector_Reading()` method detects if the MODE selector has been moved from the previous position and so if the actual MODE selector value is changed.

This function calls the `AC_Switch_Ctrl_ADC_Conversion()` function that reads the converted value from ADC and, after software filtering operations, it checks if the filtered value is in the three valid ranges fixed for the MODE selector values and sets the `MODE_value` (see [Table 4: "STEVAL-GLA001V1 operating modes: analog input and digital level"](#)) as follows:

- `MODE_1_VALUE`, `MODE_2_VALUE` or `MODE_3_VALUE` return TRUE.
- If a valid value is not detected, the function returns FALSE.

If the `AC_Switch_Ctrl_ADC_Conversion()` returns TRUE, the value read from ADC is valid and the firmware compares it with the previous one to detect if a change occurred.

In this case, the state machine is moved to **CONTROL_MODE_DETECTION** to stop control of the TRIACs, reset the system and configure the firmware for the new selected control mode.

These operations are executed by the `AC_Switch_Ctrl_Operating_Mode_Detection()` function.

If the `AC_Switch_Ctrl_ADC_Conversion()` returns FALSE, the read value is not valid and the **CONTROL_MODE_DET_ERROR** and fault status are set to `FAULT_TRUE`.

Table 4: STEVAL-GLA001V1 operating modes: analog input and digital level

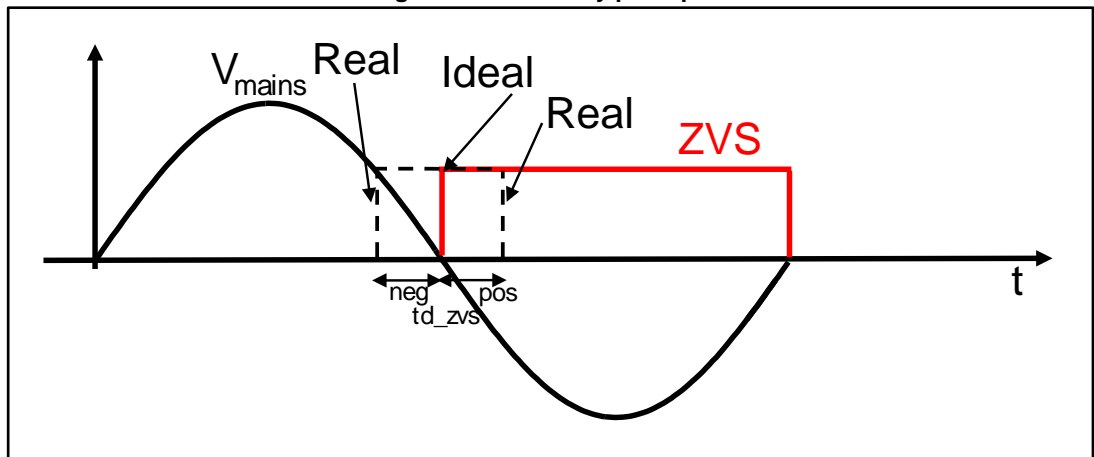
MODE selector position vs. silkscreen	Analog input	Digital value (MODE_X_VALUE)	Routine output (Oper_Mode)
ON/OFF Basic	VDD/1.62	40454	1
ON/OFF Timer	VDD/2	32768	2
Phase control	VDD/2.62	25013	3

2.2.2 ZVS signal synchronization with Vmains zero voltage crossing

The ZVS signal is used to synchronize the gate control with the mains zero voltage in ON/OFF mode and for the phase control mode, but a delay can occur, due to the component tolerance, the opto-coupler delay, etc. In these cases, there is a delay (positive or negative) on the ZVS signal^a.

^a t_{d_zvs} .

Figure 12: ZVS delay principle



The TRIAC control synchronization with ZVS has to be accurate in some applications; so, this delay has to be compensated.

For this purpose, the TIM6 is used. As mentioned, the td_{zvs} delay can be positive or negative: in the first case the pulse gate command must be anticipated and in the second case it must be delayed.

The TIM6 manages the falling edge zero crossing IRQ emulation (counting the $MainsPeriod/2$ after the rising edge ZVS_IRQ) and the td_{zvs} compensation.

The delay management is executed by setting the TIM6_ARR as follows:

- **$td_{zvs} > 0$**
 - **Equation 1**
ZVS_IRQ event

$$TIM6_ARR = \left(\frac{MainsPeriod}{2} - td_{zvs} \right) \cdot f_{clock_TIM6}$$

- **Equation 2**
TIM6_UP_IRQ event

$$TIM6_ARR = \frac{MainsPeriod}{2} \cdot f_{clock_TIM6}$$

- **$td_{zvs} < 0$**
 - **Equation 3**
ZVS_IRQ event

$$TIM6_ARR = |td_{zvs}| \cdot f_{clock_TIM6}$$

- **Equation 4**
TIM6_UP_IRQ event

$$TIM6_ARR = \frac{MainsPeriod}{2} \cdot f_{clock_TIM6}$$

In the two cases above (where $td_{zvs} \neq 0$), the firmware starts the TIM6 counting in the ZVS_IRQ, whereas it starts the TIM1 counting and TIM1 PWM output generation in the TIM6_UP_IRQ^a

^a For further details on the different operating modes, see the following paragraphs.

- **td_zvs = 0**
 - Equation 5
 - ZVS_IRQ event

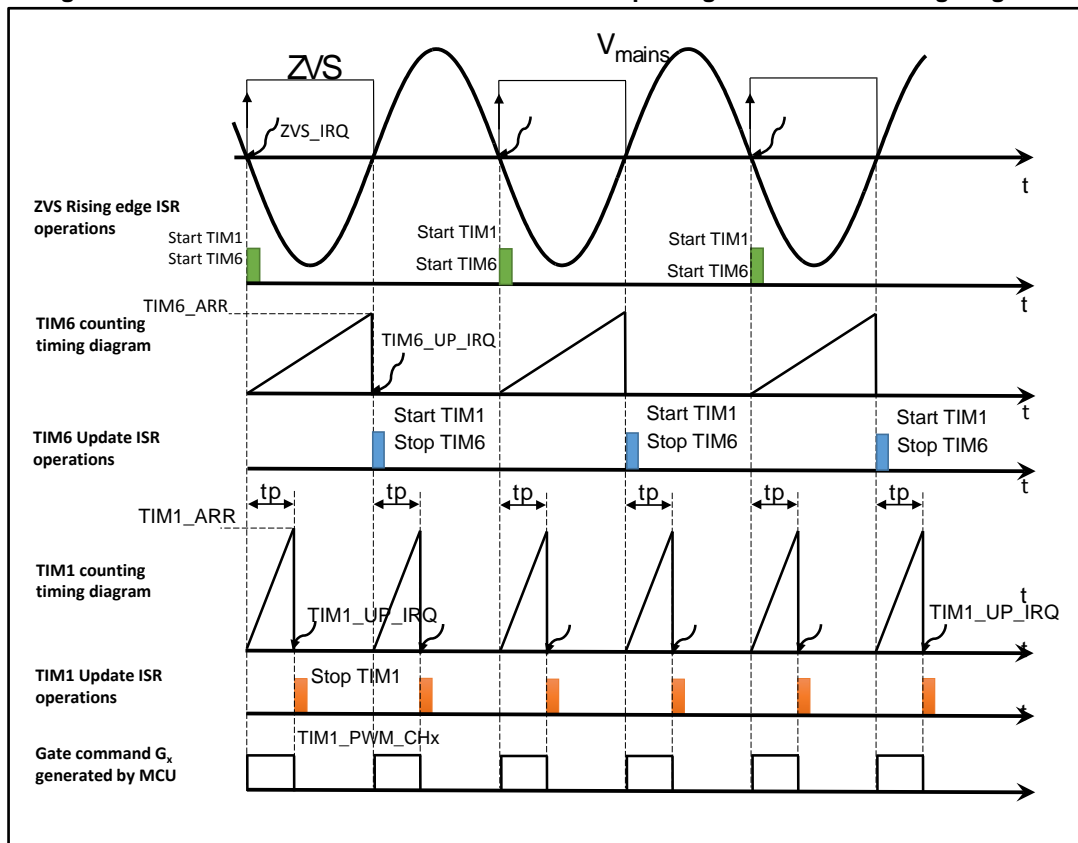
$$TIM6_ARR = MainsPeriod / 2 \cdot f_{clock_TIM6}$$

2.2.3 ON/OFF pulse basic/timer control mode

2.2.3.1 td_zvs = 0

The figure below shows the timing diagram of the ON/OFF control in basic/timer mode related to the generic TRIAC Tx with pulse command.

Figure 13: ON/OFF control in basic/timer mode with pulse gate command timing diagram



The basic mode is started and stopped via the Tx push button which corresponds to the TRIAC to control. For example, in the case of pulse command, the gate command is a pulse sequence with t_p time length.

The timer mode starts via the Tx push button and stops after a time called **OnTime**. For safety reasons, the Tx button can be pressed to stop the control. If any button is pressed, the control is stopped when the OnTime is elapsed.

As you can press the Tx button any time, the basic control starts only at the V_{mains} first rising edge after the Tx pressure (not shown in [Figure 13: "ON/OFF control in basic/timer mode with pulse gate command timing diagram"](#)).

The gate command pulse sequence is synchronized with the zero voltage switching (ZVS) signal: the **TIM1_PWM_CHx** generation starts in the ZVS signal interrupt service routine (ISR) for the rising edge zero crossing and in the **TIM6 Update ISR** for the falling edge zero

crossing of the V_{mains} signal. The TIM6 is used to count the $MainsPeriod/2$ to generate the gate pulse command at the falling edge zero crossing. The gate command signal stops in the **TIM1 Update ISR** (TIM1_UP_ISR). The TIM1_ARR (auto reload register) is calculated with the t_p value.

The TIM6_ARR is calculated with the $MainsPeriod/2$ value. In this case, the TIM1_CCRx is equal to zero to start the PWM output immediately when the counting of TIM1 starts, remaining at high level for the TIM1 period, which corresponds, in terms of counter value, to the ARR value and, in terms of time, to the pulse width t_p .

Equation 6

$$TIM1_ARR = t_p \cdot f_{\text{clock_TIM1}}$$

2.2.3.2 $td_zvs \neq 0$

The following cases are referred to the ZVS signal rising edge where the possible cases are $td_zvs > 0$ and $td_zvs < 0$.^a

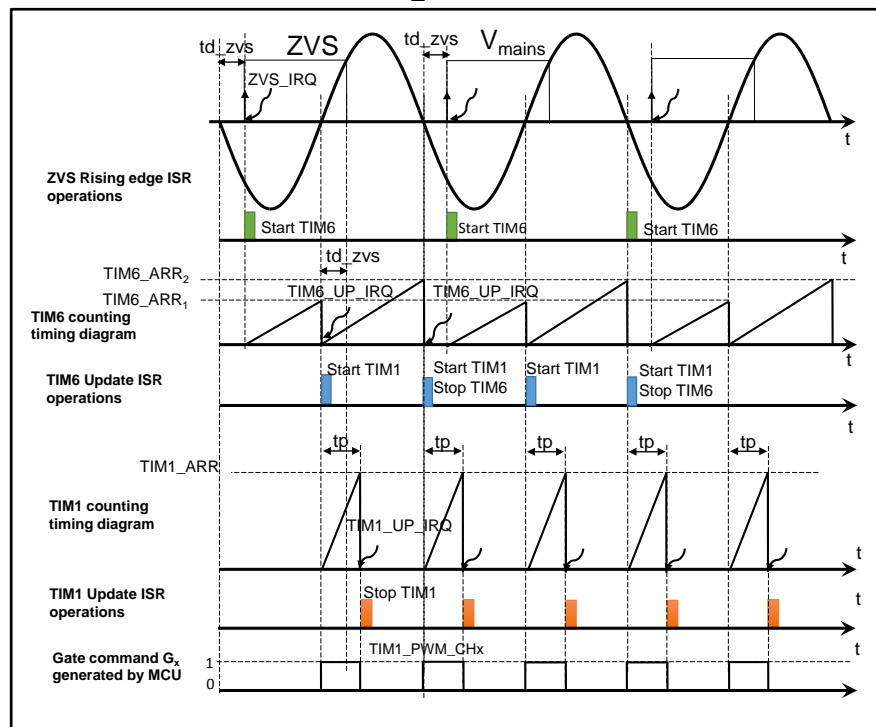


The ZVS signal rising edge corresponds to the V_{mains} signal falling edge and vice versa (refer to [Figure 12: "ZVS delay principle"](#)).

In the following timing diagrams, the pressure of the Tx push button occurs at a random time before the ZVS signal first rising event (ZVS_IRQ in the figures below).

The following figure shows the timing diagram for the generic TRIAC (Tx) gate command generation in case of a positive delay in the ZVS signal.

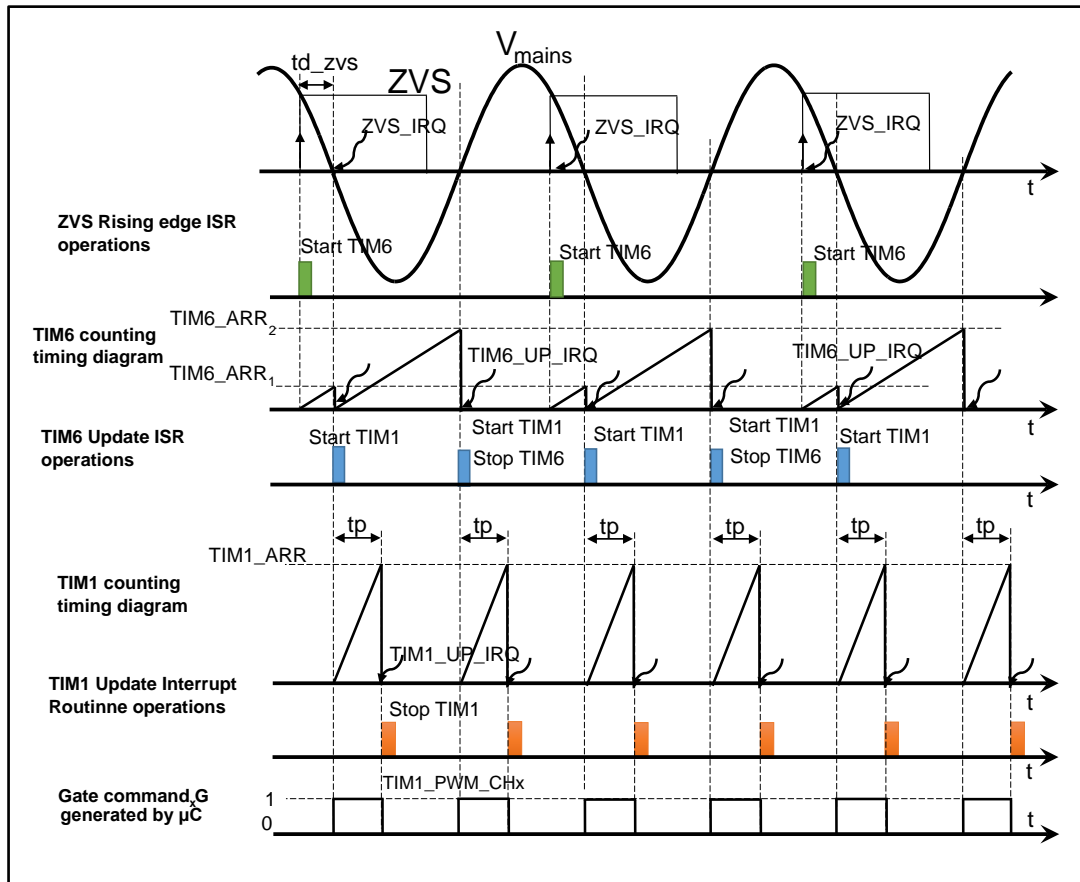
Figure 14: ON/OFF control in basic/timer mode with pulse gate command timing diagram with $td_zvs > 0$



^a Refer to [Section 2.2.5: "ZVS signal delay measurement"](#) for further details on td_zvs measurement.

The following figure shows the timing diagram for the generic TRIAC (Tx) gate command generation in case of a negative delay in the ZVS signal.

Figure 15: ON/OFF control in basic/timer mode with pulse gate command timing diagram with $td_zvs < 0$



2.2.3.3 ON/OFF DC basic/timer control mode with no ZVS signal

This control mode is active when the `zvs_hw` parameter is set as `DISABLE`; that is, when the ZVS signal detection circuit is not available or the user does not want to use it.

As shown in the figures below, if the ZVS signal is not available, the ZVS synchronization is not possible.

The gate control output is activated when a valid pressure of the Tx button is detected.

The gate control output corresponds to the TIM1 PWM_Channel_x output and is deactivated only when the button is pressed again (if the basic mode is selected) or when the **OnTime** has elapsed (if the timer mode is selected) or when the STOP(Tx) command is sent via the user interface, as detailed in [Section 2.2.6: "User interface"](#).

Figure 16: ON/OFF basic mode control with DC gate command timing diagram

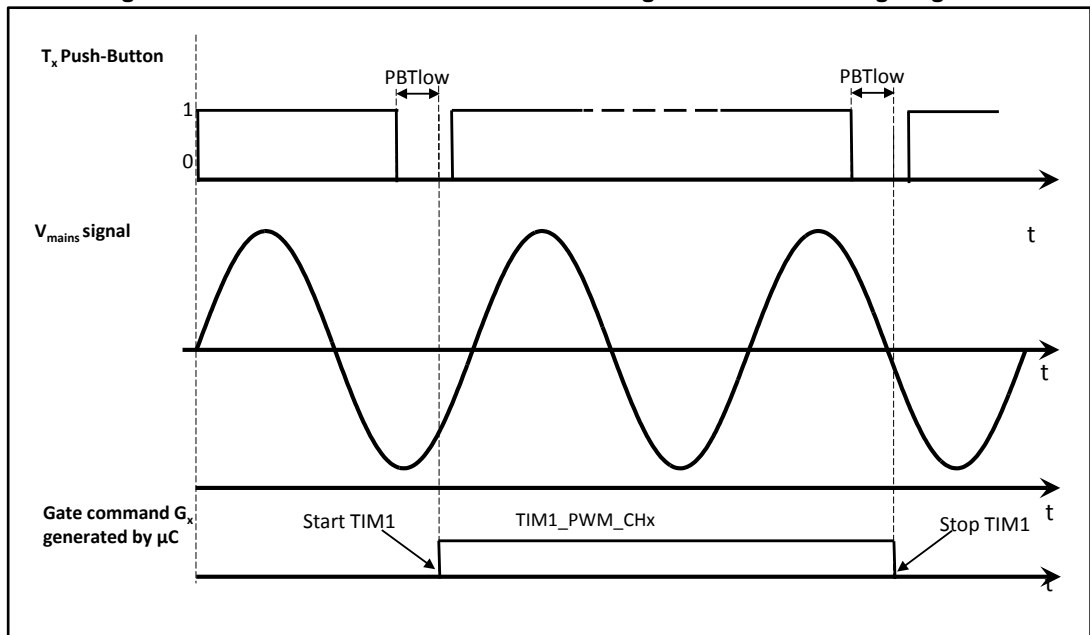
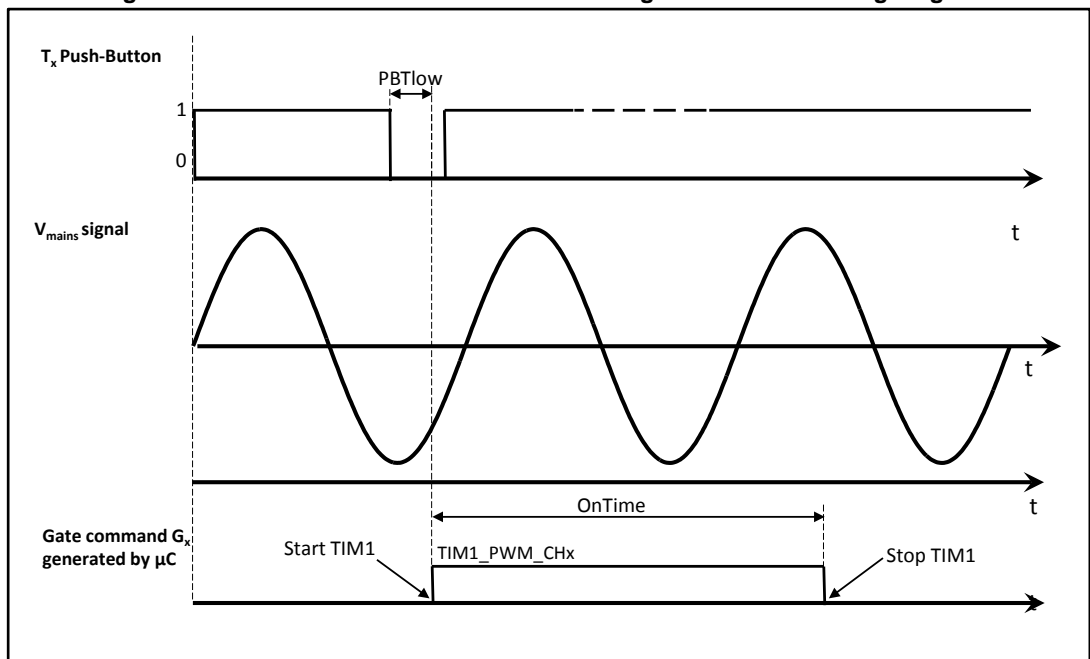


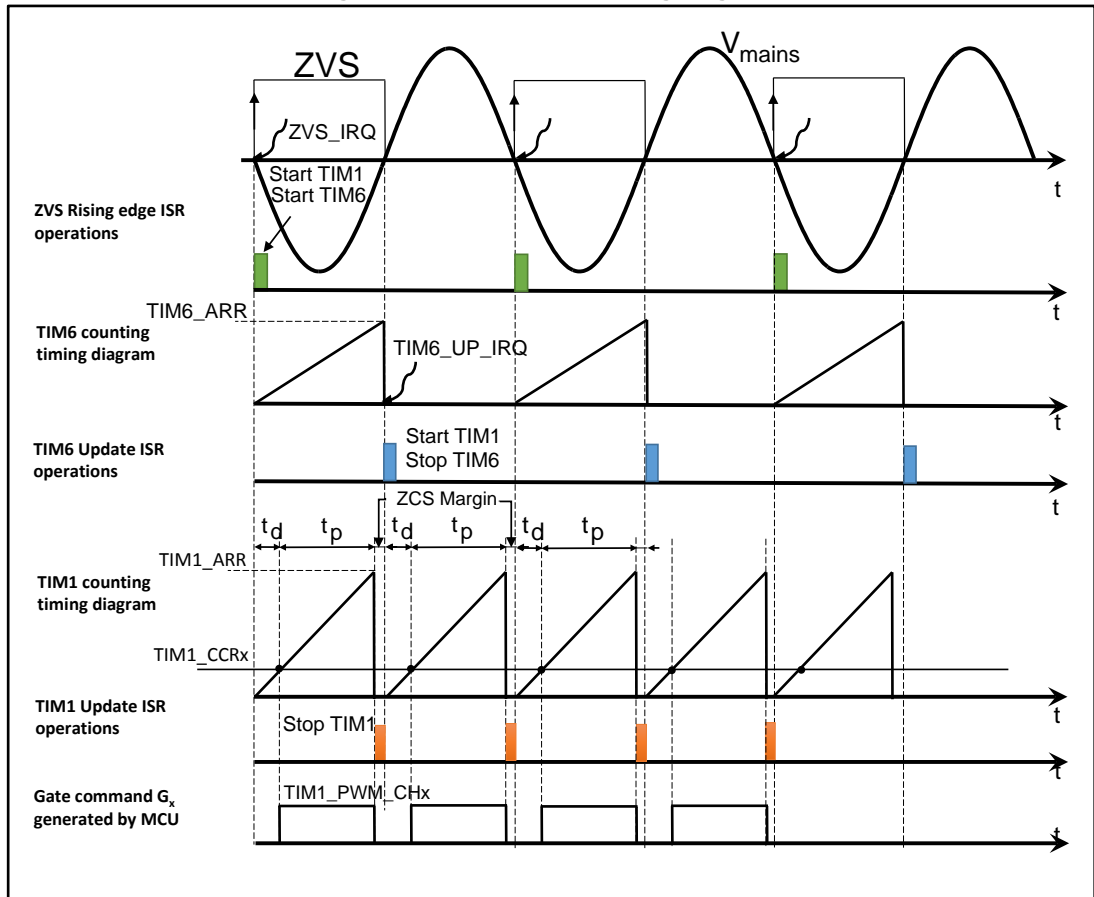
Figure 17: ON/OFF timer mode control with DC gate command timing diagram



2.2.4 Phase control mode

2.2.4.1 $td_zvs = 0$

Figure 18: Phase control timing diagram



The phase control works only with a pulse gate control and so requires a ZVS detection circuit. The control starts and stops via to the button corresponding to the TRIAC to control. In this mode, the TRIAC is turned on with a delay after the V_{mains} waveform zero crossing.

This delay is called t_d and, by changing it, the user can regulate the average current via the +/- power buttons; the t_p time is equal to:

Equation 7

$$t_p = \frac{MainsPeriod}{2} - t_d - ZCS_margin$$

The ZCS_margin time is subtracted to avoid the gate pulse overlaps the next half-cycle.

The value of the TIM1 (TIM1_ARR) auto reload register is calculated as follows:

Equation 8

$$TIM1_ARR = \left(\frac{MainsPeriod}{2} - ZCS_margin \right) \cdot f_{clock_TIM1}$$

The CCRx value of TIM1 (TIM1_CCRx) is calculated with t_d , as follows:

Equation 9

$$TIM1_CCRx = t_d \cdot f_{clock_TIM1}$$

The TIM1_PWM_Chx is activated (1 logic level) when the TIM1 counting (TIM1_CNT) value is greater than the CCRx value and deactivated (0 logic level) when it is lower than the CCRx value.

The TIM1_ARR value is equal to [Equation 8](#).

The length of the pulse command t_p becomes equal to [Equation 7](#).

For this control mode, soft-start and soft-stop operations are implemented to gradually increase and decrease the power at load start and load stop, respectively.

During the soft-start/stop operations, the STSW-GLA001V1 firmware sets different values for t_d and the TIM1_CCRx value gradually decreases/increases according to the soft-start/stop buffer defined by the user.

The TIM6 is used to count the MainsPeriod/2 value to emulate the falling edge zero crossing interrupt and generate the gate pulse command in TIM6_UP_ISR (refer to [Figure 18: "Phase control timing diagram"](#) for details).

The value of the TIM6 (TIM6_ARR) auto reload register is calculated as [Equation 2](#).

2.2.4.2 $td_zvs \neq 0$

In the phase control mode, when $td_zvs \neq 0$, the behavior is the same as the one described in [Section 2.2.3.2: " \$td_zvs = 0\$ "](#)

Figure 19: Phase control timing diagram with $td_zvs > 0$

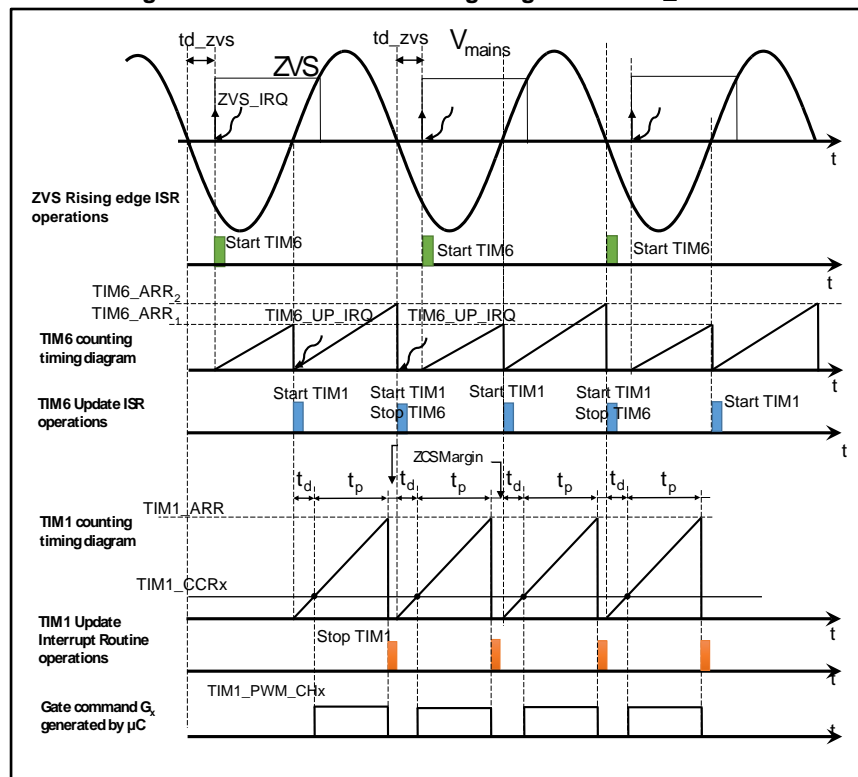
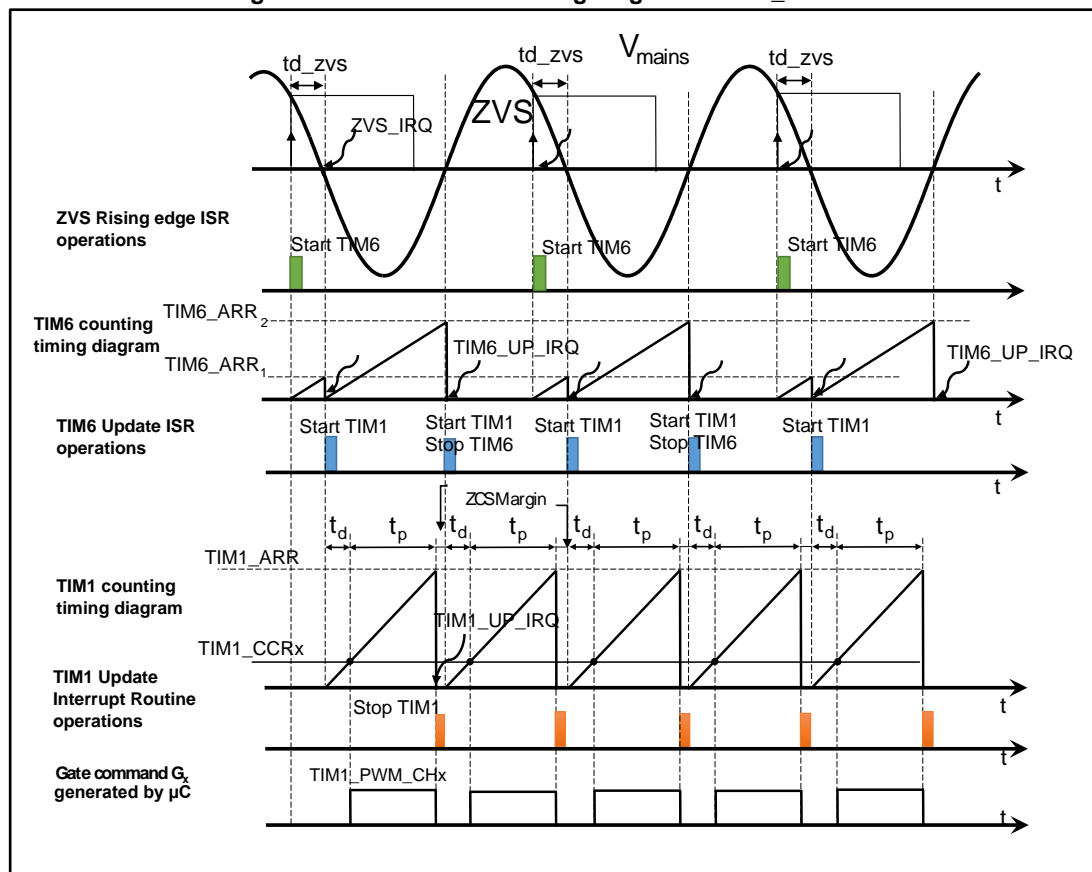


Figure 20: Phase control timing diagram with $td_{zvs} < 0$ 

2.2.5 ZVS signal delay measurement

As explained in [Section 2.2.2: "ZVS signal synchronization with V_{mains} zero voltage crossing"](#), when there is a delay in the ZVS signal, the user can measure it.

To obtain an accurate measure on the basis of the STSW-GLA001V1 firmware operations, consider the time when the ISR related to the ZVS signal as the "real zero crossing time".

The user can choose between the first V_{mains} rising edge or the first falling edge zero crossing, after the Tx button is pressed, which equates to choosing between the following two defines, leaving one of them uncommented:

```
#define WAIT_FIRST_VMAINS_RISING
#define WAIT_FIRST_VMAINS_FALLING
```

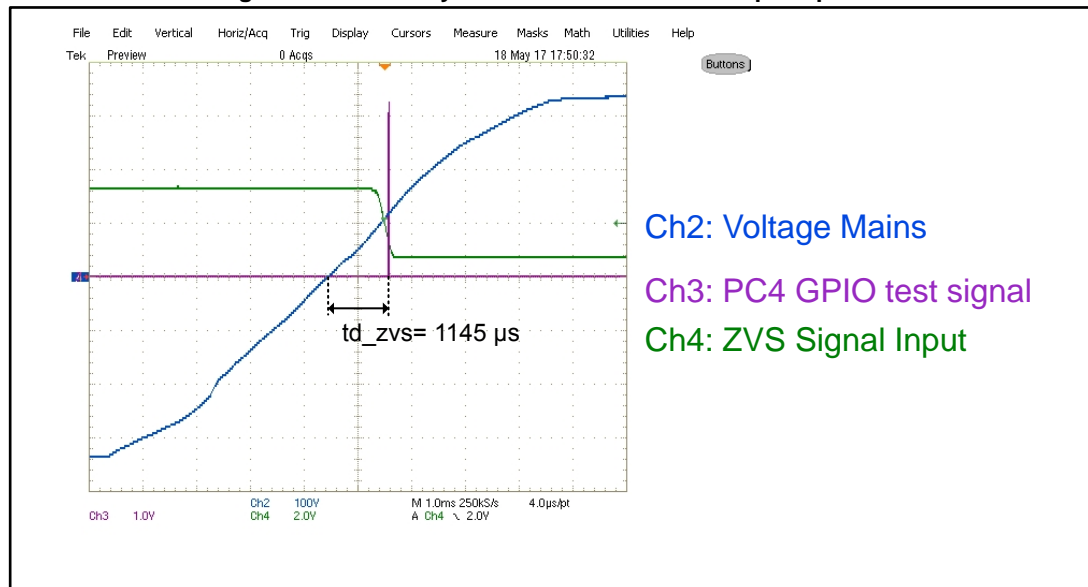
The default setting is the first define. The user has to connect two voltage probes on the NUCLEO-F030R8 board CN10 connector (pins PC4 and PA6^a), to show the PC4 GPIO test signal and the ZVS signal input, respectively. An high-voltage differential probe also has to be connected on the STEVAL-GLA001V1 evaluation board test points, **Line** and **Neutral**, ensuring to connect the high voltage probe positive terminal to Neutral and the negative one to the Line.

^a The user can alternatively use the STEVAL-GLA001V1 test point, as better described in the related documentation on www.st.com.

The PC4 GPIO signal output is switched on/off in the `void EXTI4_15_IRQHandler(void)` interrupt function to capture the time the function is called.

The following figure shows the oscilloscope capture with the delay measure considering the ZVS signal zero crossing interrupt event falling edge (ZVS signal is inverted with respect to the V_{mains} signal).

Figure 21: ZVS delay measurement - oscilloscope capture



At this point, the user has to write the measured value in the **TD_ZVS_FALLING** constant definition in the `AC_Switch_Control_param.h` header file (or in the **TD_ZVS_RISING** if the **WAIT_FIRST_VMMAIN_FALLING** constant has been defined). The value is considered positive if the PC4 pulse comes after the mains voltage zero crossing and negative if it comes before.

```
#define TD_ZVS_FALLING 1145/*< ZVS delay (-1000 to 1000 µs)
                           at the ZVS signal falling edge zero
crossing */
```

In the user interface, the measured value has to be written via the **set(td_zvs)** command, which sets the value in the `td_zvs` variable used by the STSW-GLA001V1 firmware to compensate the delay.

The value is stored in the MCU Flash memory dedicated section to restore it at any MCU reset.

2.2.6 User interface

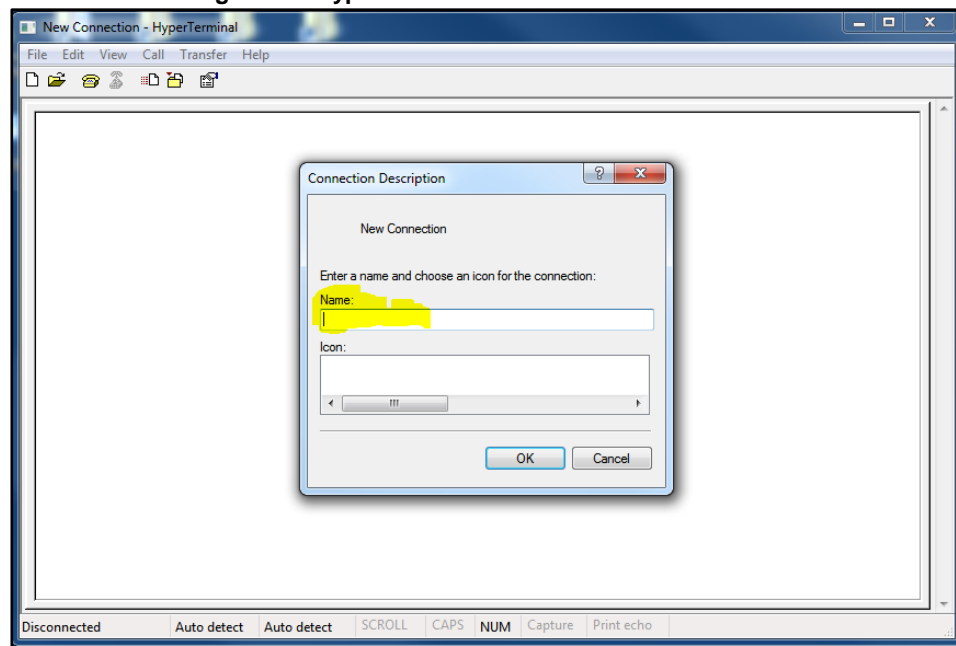
The user interface is available by selecting the correct configuration in the IAR project as explained in [Section 2.1.2: "Programming the NUCLEO-F030R8 board using a preconfigured source project"](#).

The interface can be used with any serial terminal (e.g., HyperTerminal or Tera Term).

2.2.6.1 HyperTerminal configuration

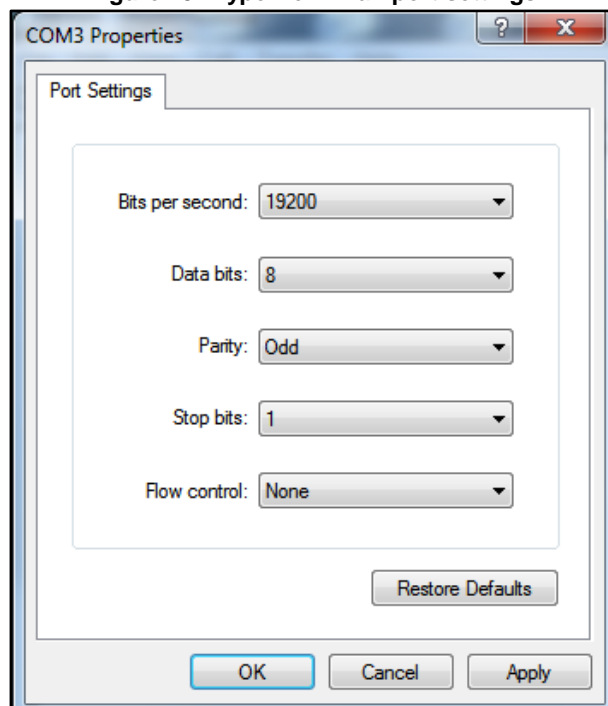
- 1 Open the HyperTerminal interface, enter a name for the connection in the related field and click OK.

Figure 22: HyperTerminal: new connection window



- 2 In the new window, select the virtual COM port in the **Connect using** field and click OK.
- 3 Set the **Port Settings** on the basis of the USART2 peripheral configured by the CubeMX program.



Figure 23: HyperTerminal: port settings



- 4 Click **Apply** and OK to confirm and close the window.
- 5 Open the project **Properties** from the **File** menu.

- 6 Select the **Settings** window and choose **VT100** in the **Emulation** field, then click OK.
- 7 Open the project **Properties** from the **File** menu and select **Terminal Setup**.
- 8 Check the **132 column mode** option and click OK to close the window.
- 9 Click on the **ASCII Setup** button.
- 10 Check the **Echo typed characters locally** field and click OK to close the window.
- 11 Click OK once again to close the Properties window.
- 12 Click on the **Call** command from the **Call menu** to start the communication if it does not start automatically.

To compile and download the program in the MCU, select **AC_SWITCH_CONTROL_SERIAL_COMM** configuration in the IAR environment and click on the **Download and Debug** button as explained in [Section 2.1.2: "Programming the NUCLEO-F030R8 board using a preconfigured source project"](#).

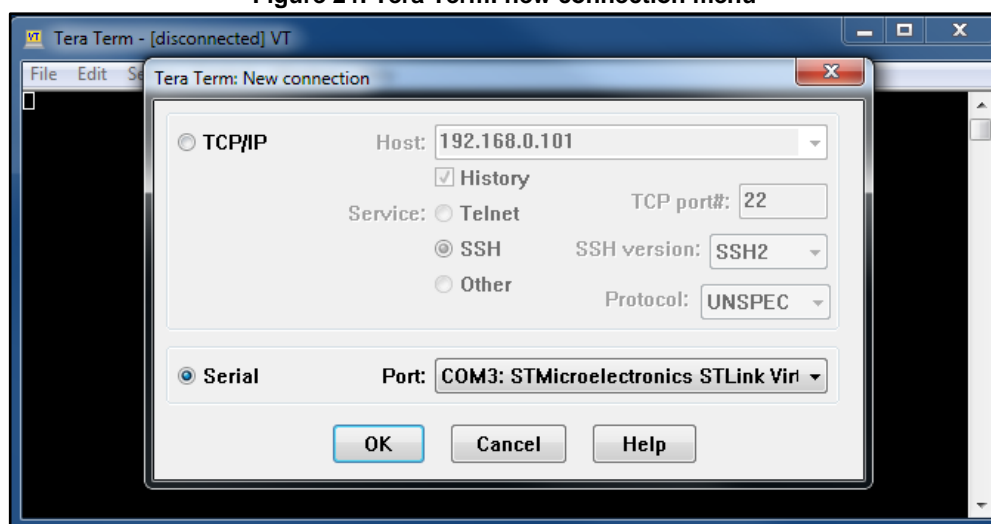
When the flashing operation finishes, in the IAR environment, click on the **GO** icon () to start the debugging, or click on the **Stop Debugging** icon () to stop it.

Another possibility for the user is to program the MCU downloading a compiled file as explained in [Section 2.1.1: "Programming the NUCLEO-F030R8 board using a binary file"](#). If the firmware has already been downloaded in the MCU, the NUCLEO-F030R8 board has to be reset by clicking on the reset button (B2) to initialize the system and display the list of commands on the HyperTerminal window.

2.2.6.2 Tera Term configuration

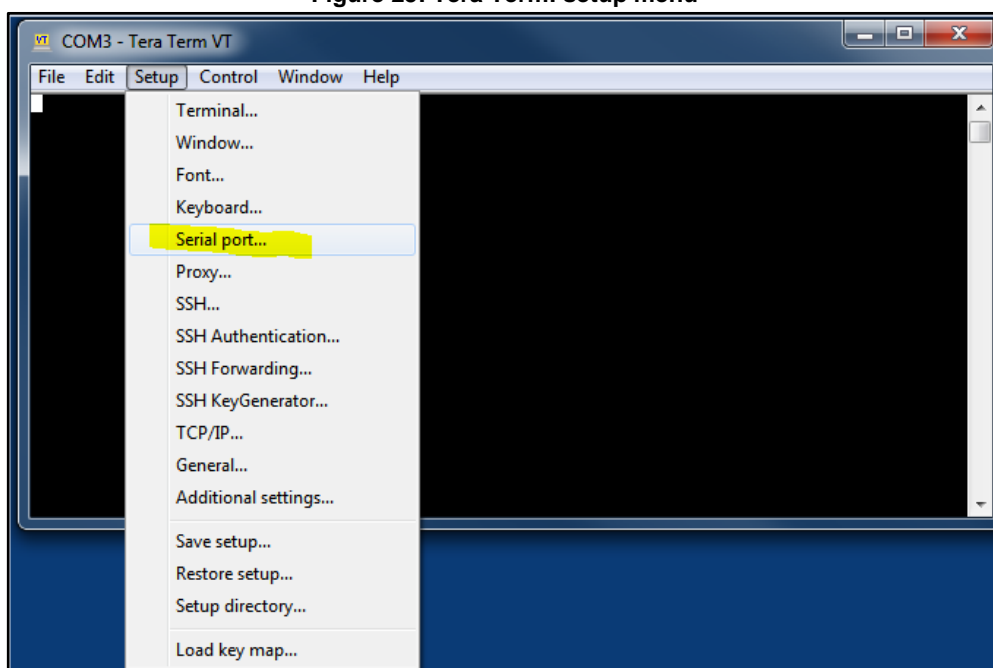
- 1 Open the Tera Term interface.
The following window pops up.

Figure 24: Tera Term: new connection menu



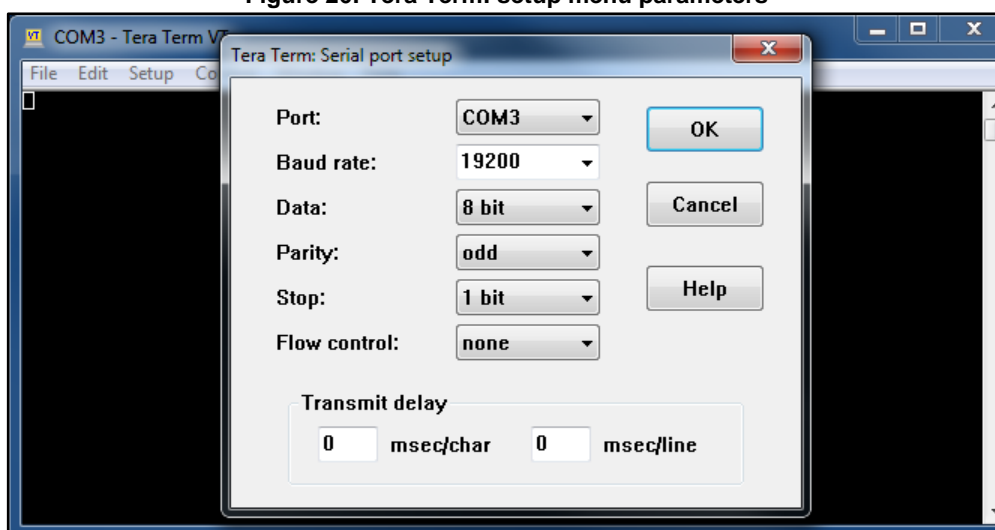
- 2 Select the **Serial** field, choose the COM port in the drop down list and click OK.
- 3 Open the **Serial Port** menu from the Setup menu.

Figure 25: Tera Term: setup menu



- 4 Set the parameters using the values shown below and click OK.

Figure 26: Tera Term: setup menu parameters



The serial terminal is now configured.^a

^a For further details about the use of this program, refer to the program Help menu.

- 5 Click on the Nucleo board reset button or reset the MCU via software or hardware. The following window appears.

Figure 27: Tera Term: command list

```

COM3 - Tera Term VT
File Edit Setup Control Window Help
=====
* STEVAL-GLA001V1 TRIACs-CONTROL BOARD
=====
List of commands:
<set(tp)>          -- Sets tp pulse value in us (10 to Mains Period/2)
<set(tp_step)>    -- Sets the variation step for TP+/TP- push buttons pressure (10 to 1 000 us)
<set(on_time)>    -- Sets the On Time duration for the Timer mode (0 to 10 000 s)
<set(on_time_step)> -- Sets the variation step for the Power Line+/Power Line- push buttons pressure (1 to 10 s)
<set(zcs_margin)> -- Sets the ZCS Margin value (100 to 1 000 us)
<set(pbt_low)>    -- Sets the Push Button pressure delay (100 to 500 ns)
<set(zus_hv)>     -- Sets if the ZUS signal is available or not (0 = unavailable or 1 = available)
<set(td_zus)>     -- Sets the delay on ZUS signal (1- Mains Period/41 to [Mains Period/41])
<set(td_min)>     -- Sets the minimum value for the time delay in us (0 to [td_max-1])
<set(td_max)>     -- Sets the maximum value for the time delay in us (<[td_min+1] to [Mains Period/2 - zcs_margin])
<set(td_index)>   -- Selects the delay buffer index to set a desired td value (1 to N_td)
<set(n_step)>     -- Sets the number of steps for Soft Start/Soft Stop (5 to 100)
<create(td_buffer)> -- Creates the time delay buffer (min 5 elements, max 100 elements). Type esc to cancel.
<rst(td_buffer)>   -- Restores the time delay buffer to factory values
<set(td_element)> -- Sets a td value into the delay buffer index (td_min to td_max)
<get(info)>       -- Monitors the System State, Operating Mode and Fault Code information
<get(vars_params)> -- Displays mode parameters. Replace 'vars_params' with: 'basic_vars' for ON/OFF mode, 'timer_vars' for Timer mode, 'phase_vars' for phase control mode, 'params' for miscellaneous parameters
<start(tx)>       -- Starts TRIAC Tx control. x must be 1, 2 or 3
<stop(tx)>        -- Stops TRIAC Tx control. x must be 1, 2 or 3
<store(data)>     -- Stores data on the Flash Memory
<rst(data)>       -- Restores the system to factory settings, erasing the stored data and resetting the MCU.
<list>           -- Type list to view command list
>_

```

- 6 To change the window background and/or text color, select the **Window** menu from the **Setup** menu.

Figure 28: Tera Term: command list sample view (white background and black font)

```

COM3 - Tera Term VT
File Edit Setup Control Window Help
=====
* STEVAL-GLA001V1 TRIACs-CONTROL BOARD
=====
List of commands:
<set(tp)>          -- Sets tp pulse value in us (10 to Mains Period/2)
<set(tp_step)>    -- Sets the variation step for TP+/TP- push buttons pressure (10 to 1 000 us)
<set(on_time)>    -- Sets the On Time duration for the Timer mode (0 to 10 000 s)
<set(on_time_step)> -- Sets the variation step for the Power Line+/Power Line- push buttons pressure (1 to 10 s)
<set(zcs_margin)> -- Sets the ZCS Margin value (100 to 1 000 us)
<set(pbt_low)>    -- Sets the Push Button pressure delay (100 to 500 ns)
<set(zus_hv)>     -- Sets if the ZUS signal is available or not (0 = unavailable or 1 = available)
<set(td_zus)>     -- Sets the delay on ZUS signal (1- Mains Period/41 to [Mains Period/41])
<set(td_min)>     -- Sets the minimum value for the time delay in us (0 to [td_max-1])
<set(td_max)>     -- Sets the maximum value for the time delay in us (<[td_min+1] to [Mains Period/2 - zcs_margin])
<set(td_index)>   -- Selects the delay buffer index to set a desired td value (1 to N_td)
<set(n_step)>     -- Sets the number of steps for Soft Start/Soft Stop (5 to 100)
<create(td_buffer)> -- Creates the time delay buffer (min 5 elements, max 100 elements). Type esc to cancel.
<rst(td_buffer)>   -- Restores the time delay buffer to factory values
<set(td_element)> -- Sets a td value into the delay buffer index (td_min to td_max)
<get(info)>       -- Monitors the System State, Operating Mode and Fault Code information
<get(vars_params)> -- Displays mode parameters. Replace 'vars_params' with: 'basic_vars' for ON/OFF mode, 'timer_vars' for Timer mode, 'phase_vars' for phase control mode, 'params' for miscellaneous parameters
<start(tx)>       -- Starts TRIAC Tx control. x must be 1, 2 or 3
<stop(tx)>        -- Stops TRIAC Tx control. x must be 1, 2 or 3
<store(data)>     -- Stores data on the Flash Memory
<rst(data)>       -- Restores the system to factory settings, erasing the stored data and resetting the MCU.
<list>           -- Type list to view command list
>_

```



Using the Tera Term program, the user must not use the numerical pad to insert the numerical inputs as this might cause false errors in typing a command due to a wrong decoding of numbers and signs.

2.2.6.3 Command list

The user interface available commands allow directly changing the parameters via the STEVAL-GLA001V1 evaluation board push buttons.

Some parameters^a cannot be modified via the push buttons, but can be modified by inserting the valid values listed below.

Table 5: STEVAL–GLA001V1 evaluation board: user interface command list

Command	Description
set(tp)	Sets the tp pulse value [μs]
set(tp_step)	Sets the variation step associated to tp +/- push buttons [μs]
set(on_time)	Sets the ON time duration for the ON/OFF timer control mode [s]
set(on_time_step)	Sets the variation step associated with the PowerTime +/- push buttons [s]
set(zcs_margin)	Sets the ZCS margin before the next cycle [μs]
set(pbtlow)	Sets the push button pressure delay [ms]
set(zvs_hw)	Sets if the ZVS signal circuit is available/used or not [0 or 1]. ⁽¹⁾
set(td_zvs)	Sets the time delay of ZVS signal [μs]. ⁽¹⁾
set(td_min)	Sets the minimum value for the turn-on delay [μs].
set(td_max)	Sets the maximum value for the turn-on delay [μs].
set(td_index)	Sets the turn on delay, creating the td buffer index [1 to N_td]
set(n_step)	Sets the number of steps for soft start/stop (5 to 100)
create(td_buffer)	Creates a new td buffer setting the number of elements, N_td, and inserting all the td buffer elements [μs] in ascending order. The user can abort this process typing esc instead of a numeric value. In this case, all the values entered are restored to default settings.
rst(td_buffer)	Restores the td buffer to default settings by checking the actual td_min and td_max values.
set(td_element)	Sets a single element in a desired index position.
get(info)	Shows the system information as: System State, Operating Mode, Fault code information, Mains Period value, Tx Order and LED states.
get(vars_params) where vars_params can be:	vars_params has to be one of following fields.
• get(basic_vars)	Shows the tp and tp_step values.
• get(timer_vars)	Shows the tp, tp_step, OnTime and OnTime_step values.
• get(phase_vars)	Shows the applied td, soft start/stop n_steps, index related to the applied td, number of elements in case of td buffer, ZCS margin, td min and td max and the td_buffer.
• get(params)	Shows the common parameters: PBTlow, td_zvs, zvs_hw.
start(tx) where tx can be:	This command can be independently applied to the three Triacs to start the triac control.
• t1	
• t2	
• t3	

^a zcs_margin, tp_step, on_time_step, tp values.

Command	Description
stop(tx) where tx can be:	This command can be independently applied to three Triacs to stop the triac control.
• t1	
• t2	
• t3	
store(data)	Stores a set of data in a dedicated section of the Flash memory of the microcontroller.
rst(data)	Restores the system to factory settings, erasing the stored data and resetting the MCU.
list	Shows the list of available commands.

Notes:

⁽¹⁾The user can change it only in the IDLE state.



To avoid data loss, launch the **store(data)** command before switching the board off or before MCU reset.

3 Revision history

Table 6: Document revision history

Date	Version	Changes
23-Nov-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