How to use STLINK-V3PWR to monitor power consumption on STM32 MCUs

Introduction

STLINK-V3PWR is a two-in-one standalone debugger probe and source measurement unit (SMU) that supplies and acquires the current consumption of any microcontroller-centric application. Like any ST-LINK device, the STLINK-V3PWR probe has a debug interface and a bridge connector, with several GPIOs and I^{2}C, SPI, and CAN protocol capabilities.

It is a key element of any electronic application to correctly manage its power consumption, and even more so for low-power applications, where energy consumption is optimized. This application note presents various usages of STLINK-V3PWR during the development of a new application.

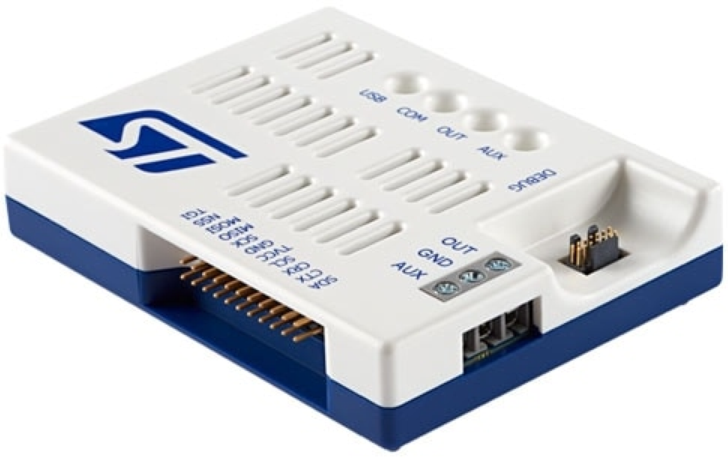
- The first chapter describes the architecture and major features of STLINK-V3PWR.
- The second chapter presents the power consumption considerations related to decoupling capacitor filtering.
- The third chapter covers synchronized trace debugging with the power supply current acquisition waveform feature.
- The following chapters describe the available STLINK-V3PWR software tools.
- The final chapter presents several usages with a basic application.

1 STLINK-V3PWR overview

STLINK-V3PWR is a very useful tool to supply, debug, and control an application. It has three connectors dedicated to those usages:

- Supplying and acquiring the current consumption waveform with the POWER connector (24 AWG cables are included).
- Debugging the microcontroller target with the DEBUG connector.
- Controlling a target application with the BRIDGE connector.

Figure 1. **The STLINK-V3PWR debug probe**



Note: Refer to UM3097: Source measurement unit (SMU) and debugger/programmer for STM32 microcontrollers for more information.

The main feature of the debug probe is the possibility to monitor the current consumption waveform of an application, and acquire helpful data to improve firmware performance and quality by:

- Monitoring and verifying the average power consumption during firmware execution (for example, compared with the product datasheet).
- Verifying the current waveform transient during application execution to ensure any wrong or timing constraint configurations.
- Reducing the number of external device or peripheral configurations to smooth the current call waveform.
- Ensuring there is no problem related to code execution, like unexpected current shapes during concurrent interrupts. (STLINK-V3PWR allows the synchronization of the current consumption waveform with the debug trace, which is very helpful during application debugging.)

2 Current consumption measurements

Power consumption measurement is based on two key elements: determining the current value, and regularly sampling this electrical parameter.

2.1 Current consumption value

STLINK-V3PWR has a unique analog circuitry, composed of:

- Amplifiers, comparators, and regulation loops, to provide a programmable and stabilized supply.
- Shunt resistors, to perform high dynamic range and accurate current-to-voltage conversions to acquire the current consumption waveform.

The main element for current-to-voltage conversion is a shunt resistor, which is a precise low resistor dedicated to establishing a correspondence between the current passing through it and the voltage developed between its terminals. By measuring the voltage at the terminals of the shunt resistor and knowing its resistor value, the current value is easy to calculate.

STLINK-V3PWR has four parallel analog paths, each occupying a different current range to cover a large dynamic range of the power consumption, from tens of nanoamps to a hundred milliamps. The range selection is managed automatically by an embedded algorithm in the probe.

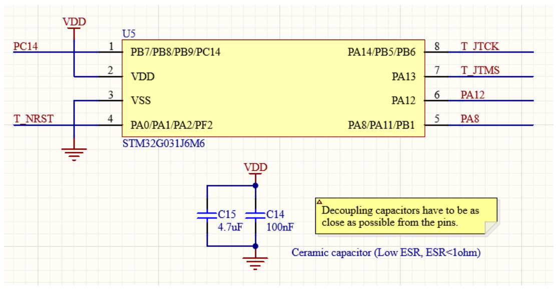
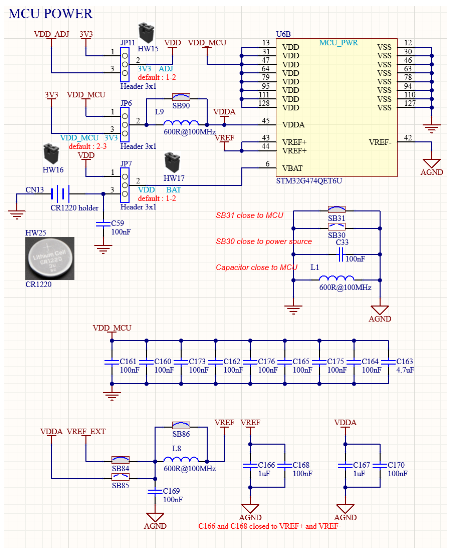
2.2 Current consumption acquisition

The acquisition of the current consumption waveform is performed with an analog-to-digital converter at programmable sample rate, which allows the building of the current consumption waveform. STLINK-V3PWR has a fixed bandwidth of 50 kHz, and a programmable sampling rate from 1 sample per second (sample/s) to 100 ksample/s. Use a 100-kilosample-per-second sampling rate for the initial configuration to acquire the current waveforms. At the maximum sampling rate (100 ksample/s), the minimum current pulse duration with a correct observation of five samples is 50 µs. The minimum number of samples per pulse duration based on the Shannon sampling theorem is two, but this is not sufficient to ensure the correct observation of the current pulse waveform.

The wires used to power the application should have the lowest resistive and inductive impedance possible to limit the voltage drop and ensure a stable voltage supply. However, the total number of decoupling capacitors embedded in the application, which have a filtering effect on the current waveform, needs to be taken into account too.

Decoupling capacitors are required for all electronic components, to reduce the power coupling between the power supply source and the supplied devices. The purpose of a decoupling capacitor is to maintain a low impedance locally, to minimize the voltage drop when the application running in the device requires a fast current response. All STM32 series products require decoupling capacitors (the total value of decoupling capacitors depends on the number of supply pins of the product), which are specified in the product datasheet or hardware guide. These are mandatory to achieve maximum performance either for digital or analog peripherals, which requires a stable power supply.

Table 1. **Total decoupling capacitor comparison between two STM32 series and two packages**

Product power supply schematic	
STM32G031J6M6 8 pins; SO8N package	STM32G484QE 128 pins; LPQFP128 package
STM32G0316-DISCO board Number of supply pins = 1 (VDD) Total decoupling capacitance: 4.8 µF 1x4.7 <µF + 1x100 nF	STM32G484E-EVAL board Number of supply pins = 12 (VDD, VDDA, VREF+) Total decoupling capacitance: 7.7 µF 1x4.7 µF + 2x1 µF + 10x100 nF
 VDD U5 PC14 T_NRST PB7/PB8/PB9/PC14 VDD VSS PA0/PA1/PA2/PF2 PA14/PB5/PB6 PA13 PA12 PA8/PA11/PB1 T_JTCK T_JTMS PA12 PA8 STM32G031J6M6 C15 4.7uF C14 100nF Decoupling capacitors have to be as close as possible from the pins. Ceramic capacitor (Low ESR, ESR<1ohm)	 MCU POWER U6B MCU_PWR VDD_ADJ 3V3 VDD_MCU VDDA VREF VBAT VREF+ VREF- AGND STM32G474QET6U SB31 close to MCU SB30 close to power source Capacitor close to MCU CR1220 holder C166 and C168 closed to VREF+ and VREF-

The next simulation figures show the filtering effect of the decoupling capacitor and acquisition sampling when a current pulse is applied with the same quality wire used.

- Real 50 µs to 10 µs pulse current waveform at STM32 supply pins (blue color).
- Real 50 µs to 10 µs pulse current waveform at STLINK-V3PWR VOUT connector (green color).
- 100-ksample/s acquired pulse current waveform by STLINK-V3PWR (red color).

Table 2. **Current acquisition sampling rate effect on current pulse observation with an STM32G0316-DISCO board**

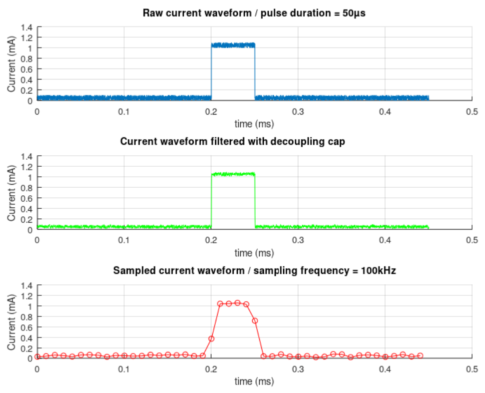
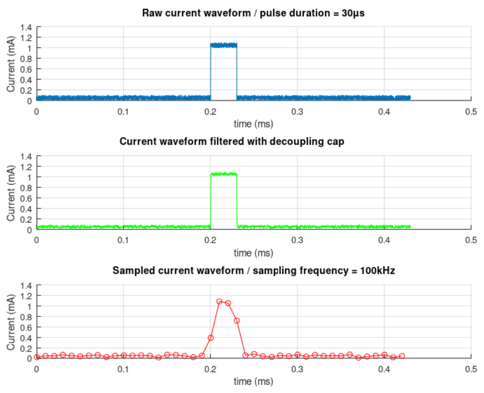
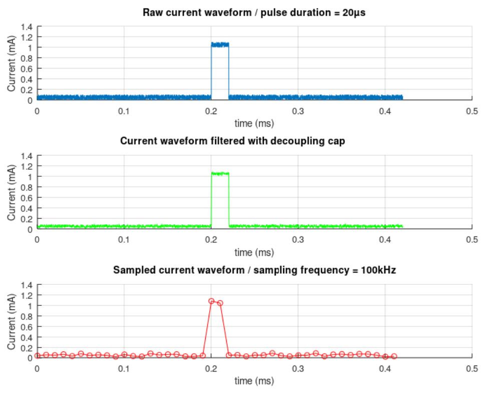
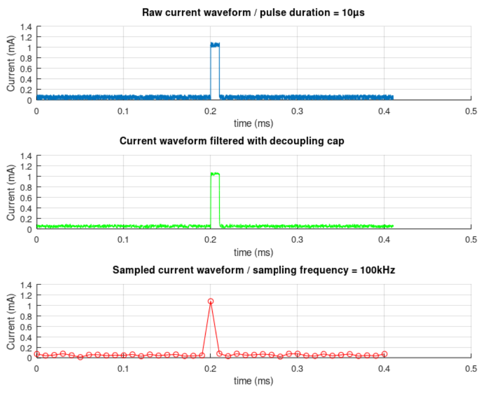
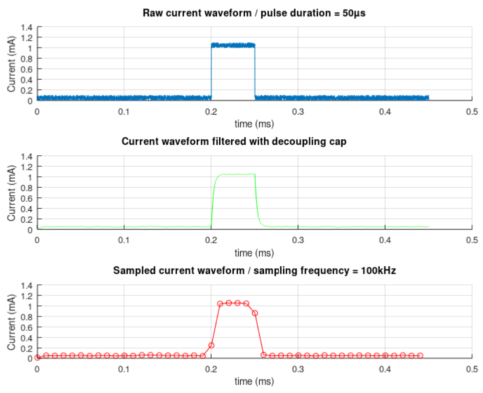
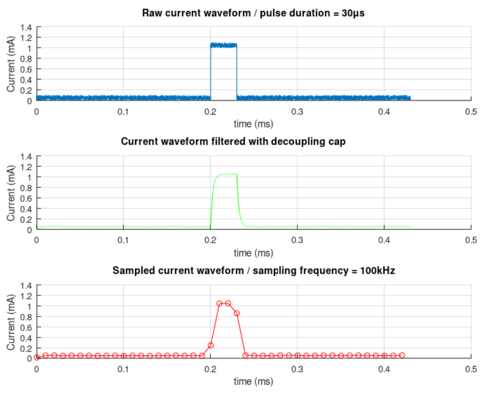
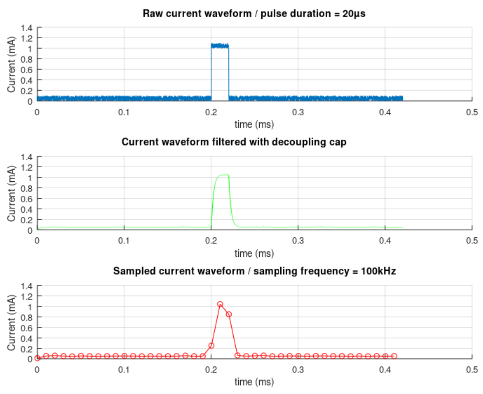
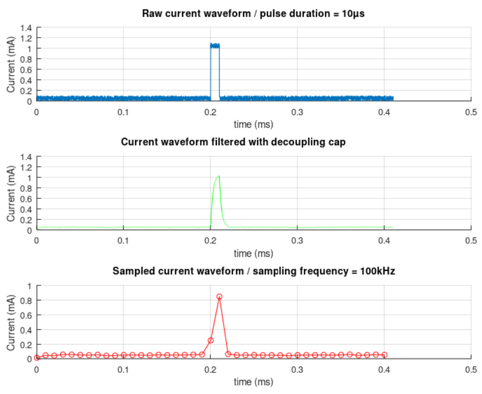
STM32G0316-DISCO board pulse current waveform observation example	
Observation is correct. Current pulse = 50 µs, 5 samples per pulse 	Observation is difficult. Current pulse = 30 µs, 3 samples per pulse 
Observation is difficult. Current pulse = 20 µs, 2 samples per pulse 	Observation is incorrect. Current pulse = 10 µs, 1 sample per pulse 

Table 3. **Current acquisition sampling rate effect on current pulse observation with an STM32G484E-EVAL board**

STM32G484E-EVAL board pulse current waveform observation example	
Observation is correct. Current pulse = 50 µs, 5 samples per pulse 	Observation is difficult. Current pulse = 30 µs, 3 samples per pulse 
Observation is difficult. Current pulse = 20 µs, 2 samples per pulse 	Observation is incorrect. Current pulse = 10 µs, 1 sample per pulse 

The figures above show that the shorter the current pulse, the higher the effects of filtering and sampling. The current waveform observation moves from correct to difficult to incorrect for the same acquisition rate. When the current waveform transient is shorter than 20 µs, only one acquisition current sample point is available when the pulse is high, so the measurement accuracy of the current waveform value is low. When the current waveform pulse duration is greater than 20 µs, a good measurement accuracy of the current waveform value is possible with two samples during the high level of the pulse.

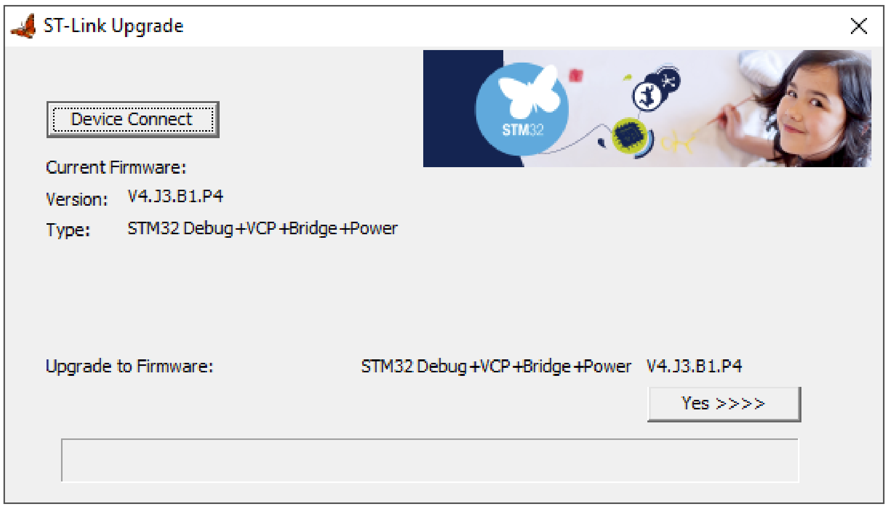
3 STLINK-V3PWR interfaces

3.1 Debug probe driver installation

The power functions of STLINK-V3PWR are controlled by a dedicated Virtual COM port. This Virtual COM port is easily identifiable by its name in the device list if the dedicated USB driver is installed. For easier identification, install the dedicated USB signed driver available on st.com.

In addition, download and execute STSW-LINK007, the firmware upgrade for ST-LINK, ST-LINK/V2, ST-LINK/V2-1, and STLINK-V3 boards. This software checks the current version of the debug probe and upgrades the firmware if needed.

Figure 2. **STSW-LINK007 - ST-LinkUpgrade interface**


ST-Link Upgrade

Current Firmware:
Version: V4.J3.B1.P4
Type: STM32 Debug+VCP+Bridge+Power

Upgrade to Firmware: STM32 Debug+VCP+Bridge+Power V4.J3.B1.P4

3.2 STLINK-V3PWR power interface

3.2.1 Power connector

STLINK-V3PWR can supply power within a 1.6 V to 3.6 V voltage range up to 500 mA on OUT, and 2 A on AUX sources. Only the OUT supply current waveform is acquired, while the AUX supply is dedicated as an auxiliary power supply.

It is recommended not to use wires longer than ~30 cm, and to twist the OUT and GND wires to reduce parasitic inductance. It is also recommended to use a cable diameter adapted to the current value. The 24AWG cable with a 0.72-mm conductor diameter is adapted to the current source capabilities of STLINK-V3PWR. Four cables (two male/male and two male/female) are included in the product package.

The power interface also includes two trigger input (TGI) and output (TGO) signals on the bridge connector, which can be used to start an acquisition from an external event and generate an output event when the current acquisition waveform is above a configured threshold.

Refer to the STLINK-V3PWR connection figure in *UM3097: Source measurement unit (SMU) and debugger/programmer for STM32 microcontrollers* (chapter *Typical application*) for more information on how to connect to the target application.

Attention: *Always make sure the OUT and AUX output supplies are powered off when connecting cables between STLINK-V3PWR and the application board.*

3.2.2 USB communication

After the driver installation, the STLINK-V3PWR Virtual COM port PWR(power interface) is displayed, like in the figure below. The power consumption acquisition feature is available when the probe is connected to a 1.5-A USB port (the probe USB led is green: refer to the chapter *USB connection with a host PC* in *UM3097: Source measurement unit (SMU) and debugger/programmer for STM32 microcontrollers*)

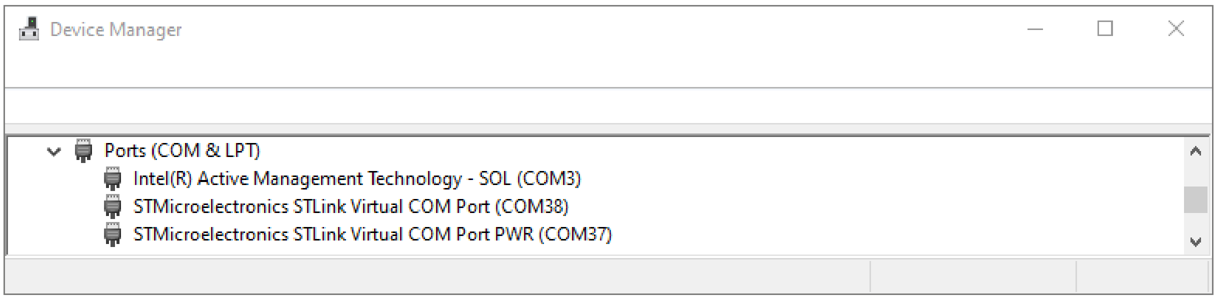

Device Manager
Ports (COM & LPT)
Intel(R) Active Management Technology - SOL (COM3)
STMicroelectronics STLink Virtual COM Port (COM38)
STMicroelectronics STLink Virtual COM Port PWR (COM37)

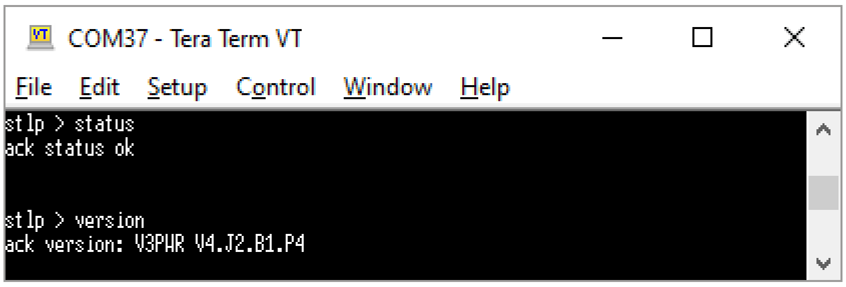
Figure 3. **STLINK-V3PWR Virtual COM port PWR is COM37 in the Windows 10® environment**

The communication between the computer and the STLINK-V3PWR power interface is based on a specific syntax, described in *UM2269: Getting started with PowerShield firmware*. The default configuration of the Virtual COM port is:

- Baud rate: 3.6 Mbit/sec (3686400 bauds)
- Data: 8 bits
- Stop: 1 bit
- Parity: none
- Flow control: none

Enable local echo, new-line receive/transmit CR+LF.

The figure below illustrates a simple example of communication with STLINK-V3PWR, using terminal emulator software like Tera Term with status and version commands.



```
stlp > status
ack status ok


stlp > version
ack version: V3PWR V4.J2.B1.P4
```

Figure 4. **Example of STLINK-V3PWR communication using Tera Term software**

The Virtual COM port interface can be used easily to communicate and interact with script languages or automation software.

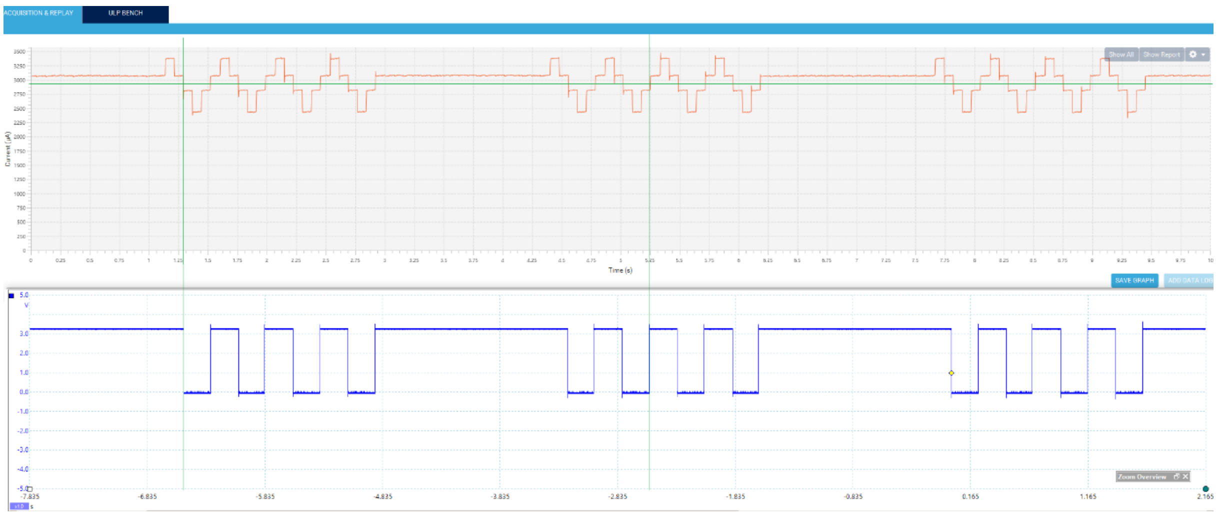
To simplify the user experience, it is recommended to use STM32CubeMonitor-Power, which is a dedicated software tool for interfacing with STLINK-V3PWR.

With STM32CubeMonitor-Power, the user can control STLINK-V3PWR through a graphical interface, to manipulate and save current waveform acquisitions. For more information related toSTM32CubeMonitor-Power, refer to *UM2202: STM32CubeMonitor-Power software tool for power andultra-low-power measurements*. The next chapter presents several usages of STM32CubeMonitor-Power

3.2.3 Triggers input and output

STLINK-V3PWR includes a side connector with dedicated pins for the bridge interface and input/output trigger signals. The trigger input (TGI) and output (TGO) signals can be used to start an acquisition from an external digital event and generate an output digital event while the acquisition is running, when the current acquisition waveform is above or below a configured threshold (output is low when it is below the threshold and high when it is above the threshold).

Figure 5. Example of STLINK-V3PWR trigger output with 2900 µA current threshold (current waveform in red with STM32CubeMonitor-Power, TGO signal in blue with PicoScope® software)



Refer to the STLINK-V3PWR connection figure in *UM3097: Source measurement unit (SMU) and debugger/ programmer for STM32 microcontrollers* (chapter *Typical application*) for more information on how to connect the bridge connector pins.

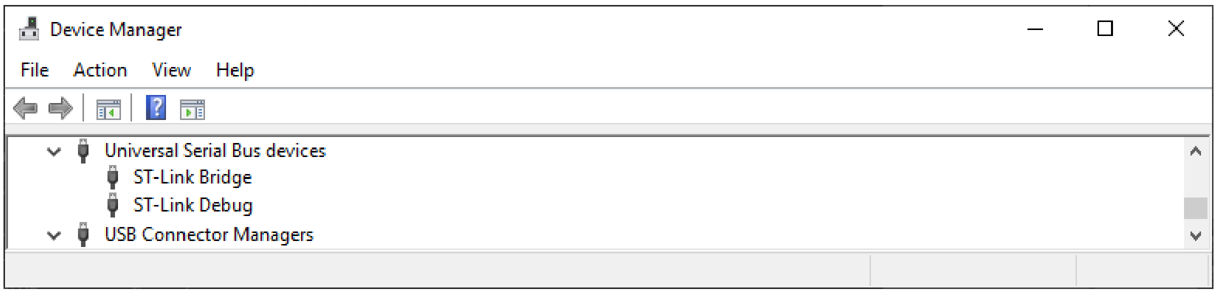
3.3 STLINK-V3PWR debug interface

Being a standard ST-LINK debug probe, STLINK-V3PWR has a debug connector with the STDC14 form factor to connect the SWD or JTAG interfaces of any STM32 microcontroller (STDC14 adaptors should be purchased separately if the application debug connector is Arm® JTAG20 or Tag-Connect™, for example).

The debug interface also provides UART signals for the debug Virtual COM port (COM38 in the following example). The related pinout for the debug connector is described in *UM3097: Source measurement unit (SMU) and debugger/programmer for STM32 microcontrollers* (chapter *Typical application*: connector pinout table).

Moreover, for more connection flexibility, there are three STDC14 to MIPI10/STDC14/MIPI20 flat cables with 1.27-mm pitch connectors included in the product package.

Figure 6. STLINK-V3PWR debug USB device in a Windows 10 environment

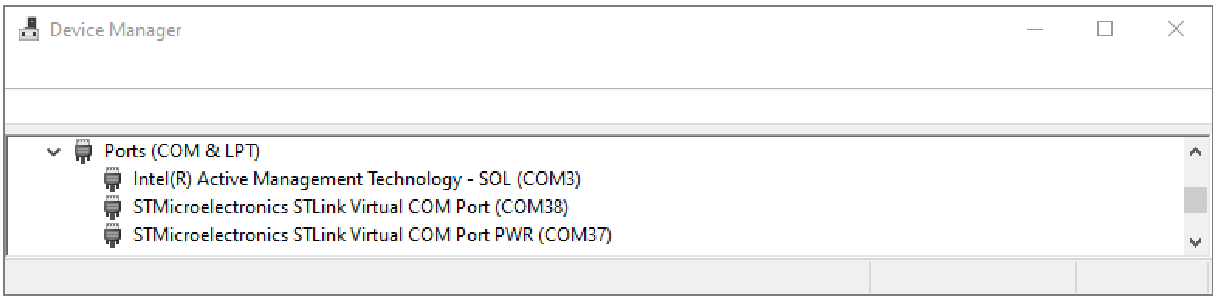


Device Manager

File Action View Help

- Universal Serial Bus devices
  - ST-Link Bridge
  - ST-Link Debug
- USB Connector Managers

Figure 7. STLINK-V3PWR COM PORT is COM38 in a Windows 10 environment



Device Manager

- Ports (COM & LPT)
  - Intel(R) Active Management Technology - SOL (COM3)
  - STMicroelectronics STLink Virtual COM Port (COM38)
  - STMicroelectronics STLink Virtual COM Port PWR (COM37)

3.3.1 SWD protocol

The Serial Wire Debug (SWD) allows access to the STM32 debug interface (Arm® debug interface). This enables the debug probe to become another bus controller to access the embedded memories, peripherals, and debug registers, to configure breakpoints, for example. SWD is a debug interface that uses five signals:

- GND: target application STM32 VSS ground.
- T_VCC: target application V_{DDIO} STM32 supply.
- SWDCLK: SWD clock signal.
- SWDIO: SWD bidirectional data signal.
- T_NRST: target application reset.

The typical communication frequency for this interface with V_{DD} = 3.3 V is 10 MHz.

Refer to *UM3097: Source measurement unit (SMU) and debugger/programmer for STM32 microcontrollers* (chapter *Debug and bridge performance*) and *AN4989: STM32 microcontroller debug toolbox* for more information.

3.3.2 SWD+SWV protocol

In addition to the SWD protocol, the Serial Wire Viewer (SWV) adds a serial output signal dedicated to firmware execution observation. It is configured via SWD: users can configure data comparators, ITM messages, interrupt calls, and program counters, and show those data inside their integrated development environment. SWD+SWV is a debug and output serial interface that uses six signals:

- GND: target application STM32 VSS ground.
- T_VCC: target application V_{DDIO} STM32 supply.
- SWDCLK: SWD clock signal.
- SWDIO: SWD bidirectional data signal.
- SWO: serial output data signal.
- T_NRST: target application reset.

The typical communication frequency for this interface with V_{DD} = 3.3 V is 12 MHz.

Refer to *UM3097: Source measurement unit (SMU) and debugger/programmer for STM32 microcontrollers* (chapter *Debug and bridge performance*) and *AN4989: STM32 microcontroller debug toolbox* for more information.

3.3.3 JTAG protocol

The joint test action group (JTAG) connection was originally developed for continuity testing of electronic boards, but is now also used in debugging and programming. It is an extremely popular in-circuit debugger (ICD) technique in all modern microcontrollers. JTAG is a debug interface that uses seven signals:

- GND: target application STM32 VSS ground.
- T_VCC: target application V_{DDIO} STM32 supply.
- T_JCLK: JTAG clock signal.
- T_JTMS: JTAG mode select signal.
- T_JTDI: JTAG input data signal.
- T_JTDO: JTAG output signal.
- T_NRST: target application reset.

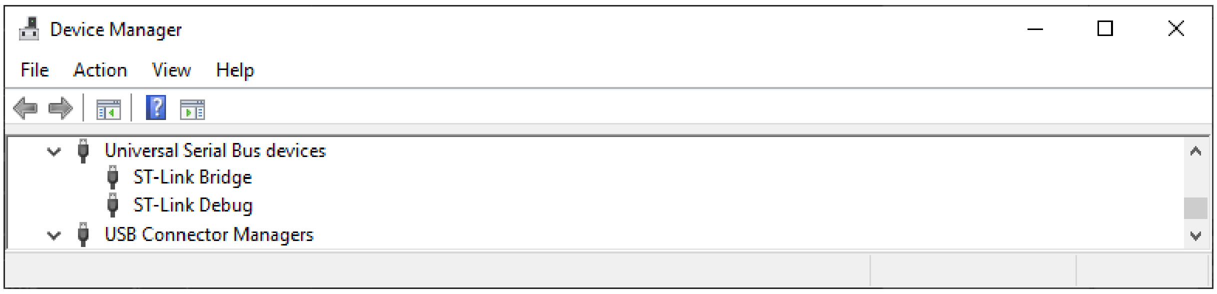
The typical communication frequency for this interface with V_{DD} = 3.3 V is 20 MHz.

Refer to *UM3097: Source measurement unit (SMU) and debugger/programmer for STM32 microcontrollers* (chapter *Debug and bridge performance*) and *AN4989: STM32 microcontroller debug toolbox* for more information.

3.4 STLINK-V3PWR bridge interface

STLINK-V3PWR includes a multipath bridge connector on its side with dedicated pins for SPI, UART, I^2C, and CAN communication, and GPIO control.

Figure 8. **STLINK-V3PWR bridge USB device in a Windows 10 environment**



Refer to the STLINK-V3PWR connection figure in *UM3097: Source measurement unit (SMU) and debugger/ programmer for STM32 microcontrollers* (chapter *Typical application*) for more information on how to connect the bridge connector pins.

4 Connecting STLINK-V3PWR

4.1 Supply a generic application board

Connect the STLINK-V3PWR OUT and GND supplies to your application power connector. The additional AUX supply output can be used for auxiliary usage. Only the current waveform from the OUT supply is acquired by the probe. Use the provided wires, which have good electrical performances (the 24 AWG cable with a 0.72-mm conductor diameter is adapted to STLINK-V3PWR usage). The board schematic and layout should be analyzed to connect the STLINK-V3PWR VOUT supply source and GND ground reference correctly.

4.2 Supply an STM32 Nucleo, Discovery, or Evaluation board

STM32 boards are mostly supplied by the USB ST-LINK connector with a 5-V source by default. Those boards embed one or several voltage regulators to supply the V_{DD} of the STM32 microcontroller. The board schematic and layout should be analyzed to connect the STLINK-V3PWR VOUT supply source and GND ground reference correctly.

Important: *The IDD jumper is dedicated to power consumption measurement with an ammeter. STLINK-V3PWR is a supply measurement unit and not an ammeter, so the IDD jumper should be closed.*

In addition to providing the power supply, STLINK-V3PWR allows for target debugging through its STDC14 debug connector. STLINK-V3PWR should simply be connected to the target application debug connector STDC14 receiver with the cable available in the STLINK-V3PWR kit.

The following examples describe how to configure the board and where to connect the STLINK-V3PWR VOUT and GND signals.

Example 1: NUCLEO-G474RE Nucleo board (MB1367).

Connector, jumper, and solder bridge configurations should be modified with the following setup (refer to board reference documents via https://www.st.com/en/evaluation-tools/nucleo-g474re.html):

- USB CN1 not connected (no USB cable).
- JP1 jumper closed (MB1367 ST-LINK NRST pin connected to GND).
- JP3 jumper closed (NRST signal).
- JP5 jumper closed (5V_USB_STLK position).
- JP6 jumper closed (3V3 connected to VDD).
- JP8 jumper closed (3V3 connected to VREF).
- SB5 solder bridge open (MB1367 U12 VOUT output disconnected from 3V3).

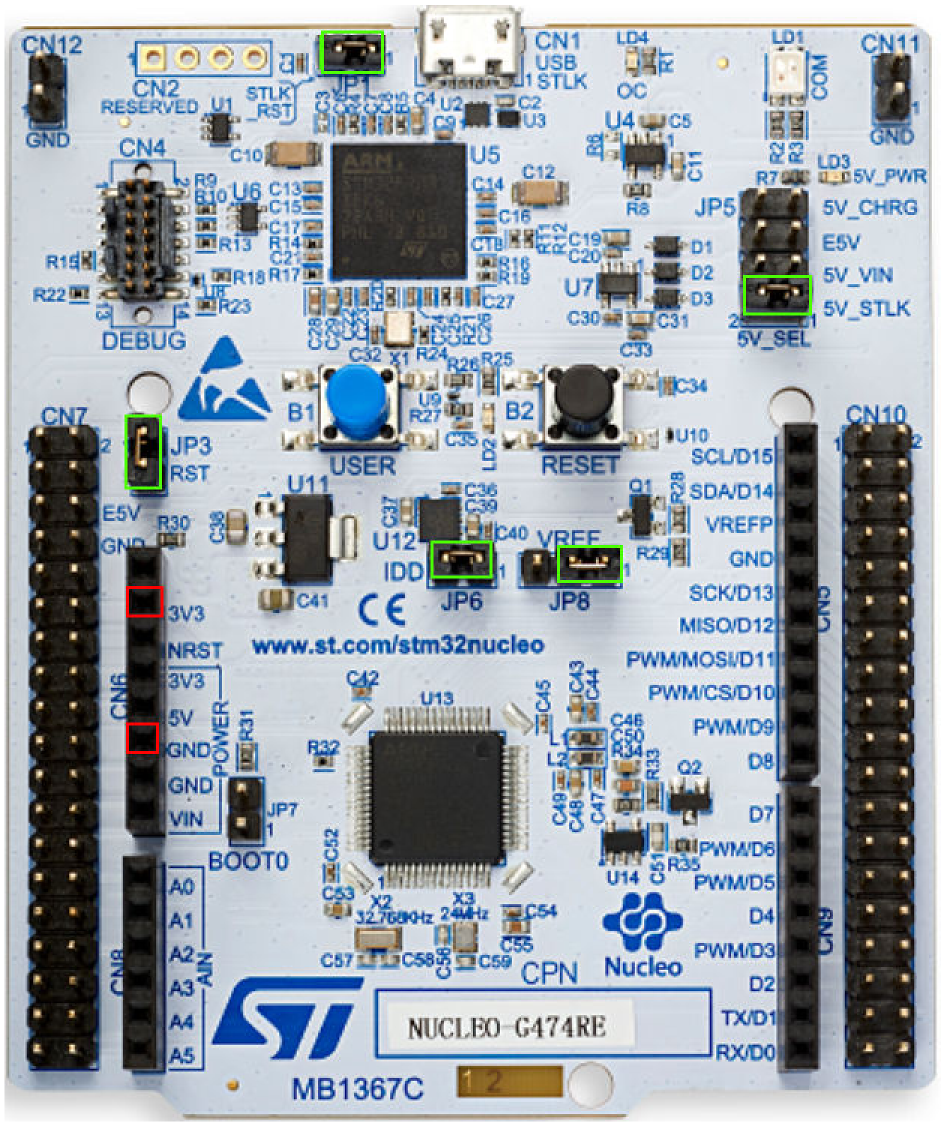
Connect STLINK-V3PWR with the following setup:

- STLINK-V3PWR VOUT to IOREF/VDD (CN6 connector pin #2).
- STLINK-V3PWR GND to GND (CN6 connector pin #6).

Connect the STLINK-V3PWR debug interface:

- STLINK-V3PWR debug connector with an STDC14 cable to MB1367 connector CN4.

Figure 9. **MB1367 configuration (jumpers in green, VOUT/GND connection pins in red)**



Example 2: NUCLEO-U575ZI-Q Nucleo board (MB1549)

Connector, jumper, and solder bridge configurations should be modified with the following setup (refer to board reference documents via https://www.st.com/en/evaluation-tools/nucleo-u575zi-q.html):

- USB CN1 not connected (no USB cable).
- JP1 jumper closed (MB1549 STLINK NRST pin connected to GND).
- JP2 jumper closed (T_NRST signal).
- JP4 jumper open (VDD disconnected from 3V3 and 1V8).
- JP5 jumper closed (IDD).
- JP6 jumper closed (5V_STLK position).

Connect STLINK-V3PWR with the following setup:

- STLINK-V3PWR VOUT to IOREF/VDD (CN8 connector pin #3).
- STLINK-V3PWR GND to GND (CN8 connector pin #11).

Connect the STLINK-V3PWR debug interface:

- STLINK-V3PWR debug connector with an STDC14 cable to MB1549 connector CN5.

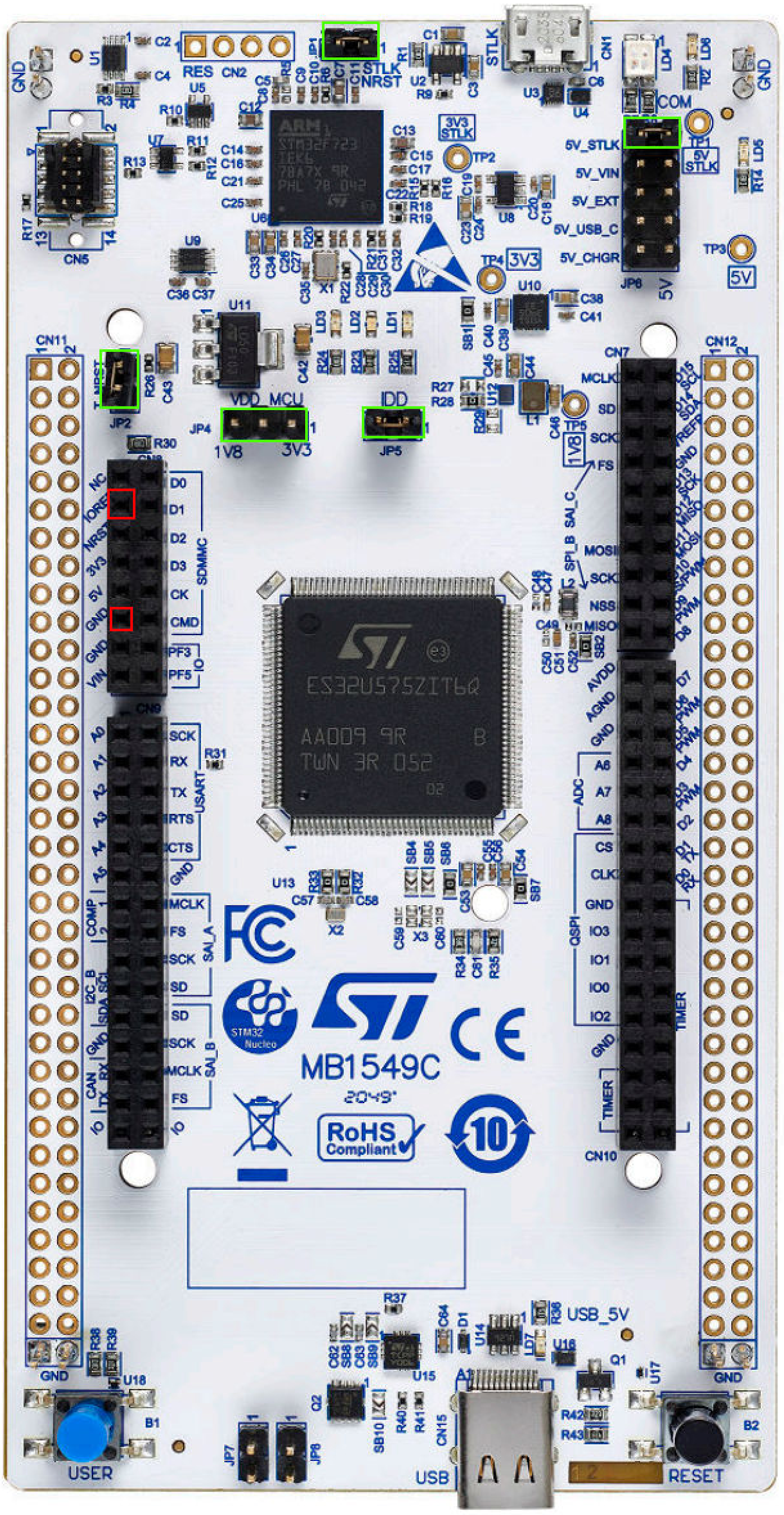


Figure 10. MB1549 configuration (jumpers in green, VOUT/GND connection pins in red)

5 STLINK-V3PWR software tools

STM32CubeMonitor-Power software manages the power interface of the STLINK-V3PWR probe. It uses the dedicated PWR Virtual COM port to communicate with the probe and retrieve the current waveform acquisition samples. This chapter describes the possible usage for STM32CubeMonitor or GNU Octave to create your own graphical user interface.

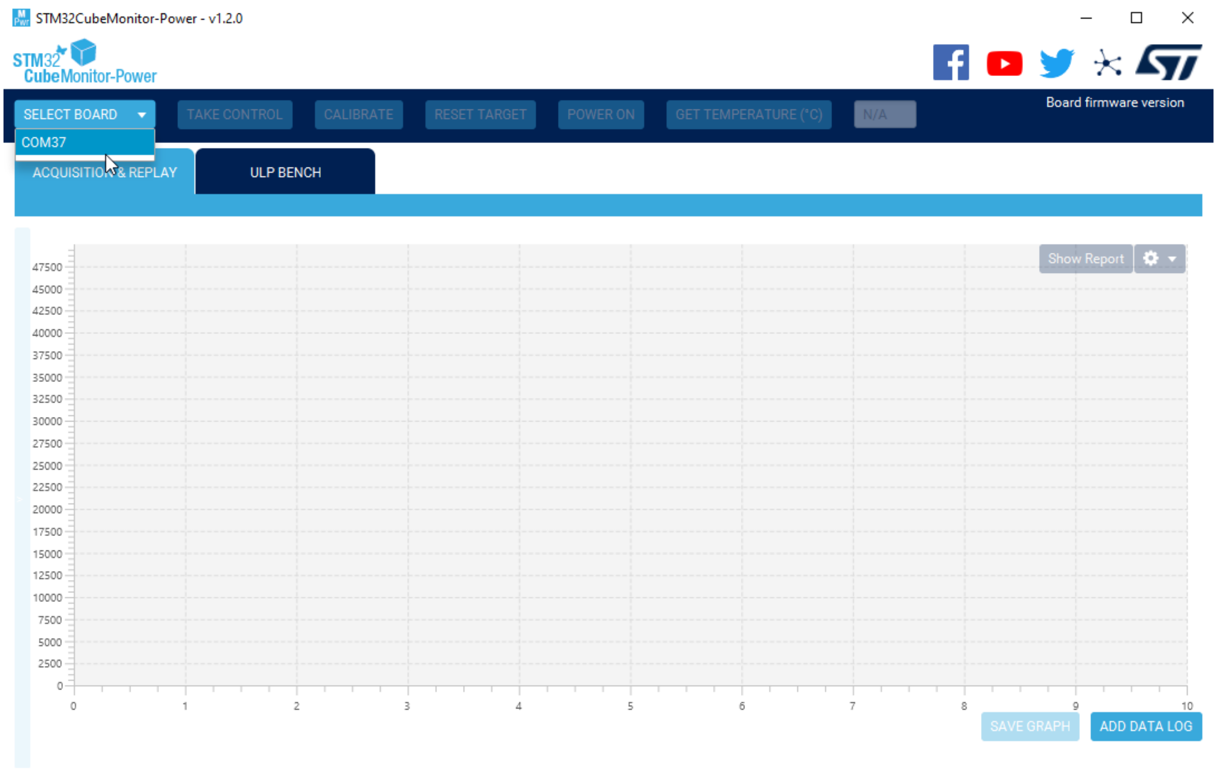
5.1 STM32CubeMonitor-Power

This chapter provides a step-by-step description of the actions to acquire and save a current waveform.

5.1.1 First steps with the software tool

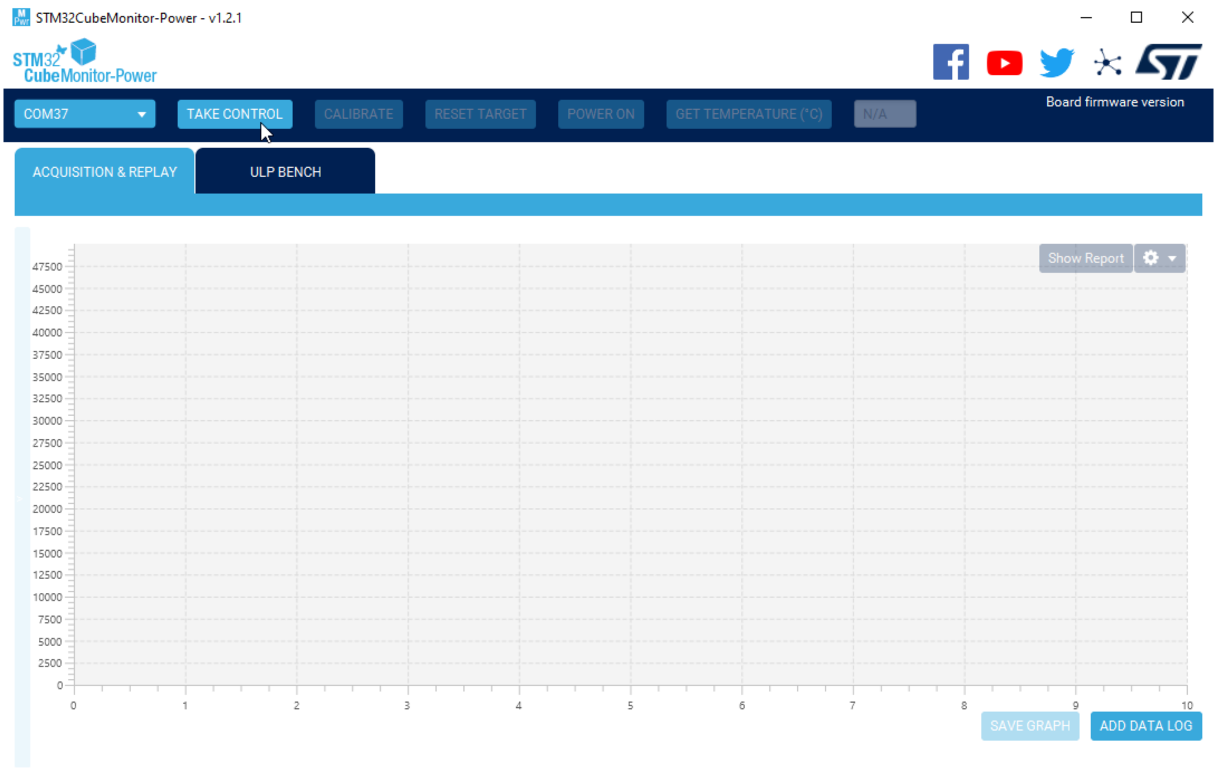
1. Launch STM32CubeMonitor-Power and select an available Virtual COM port (only STLINK-V3PWRVirtual COM ports are displayed).

Figure 11. STLINK-V3PWR configuration with STM32CubeMonitor-Power



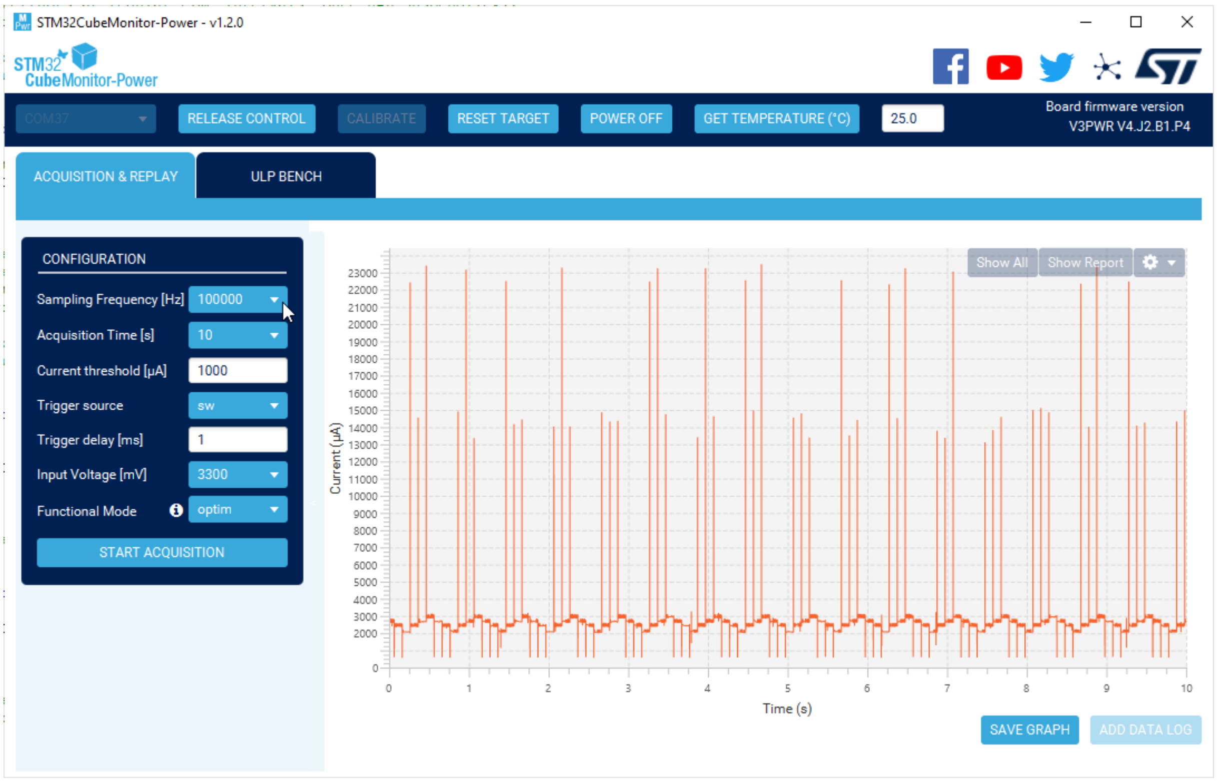
2. Click on [**TAKE CONTROL**] to connect STLINK-V3PWR and start its configuration with the ACQUISITION & REPLAY panel on the left side.

Figure 12. STLINK-V3PWR configuration with STM32CubeMonitor-Power



3. For the initial configuration, set the sampling frequency parameter to 100 kHz to ensure full bandwidth acquisition. After turning the application power on with the [**POWER ON**] button (both the STLINK-V3PWR OUT and AUX LEDs are now green), set the acquisition time to the desired duration (for example, 10 seconds) and then click on [**START ACQUISITION**] (the OUT LED is now blinking).

Figure 13. STLINK-V3PWR configuration and start acquisition with STM32CubeMonitor-Power

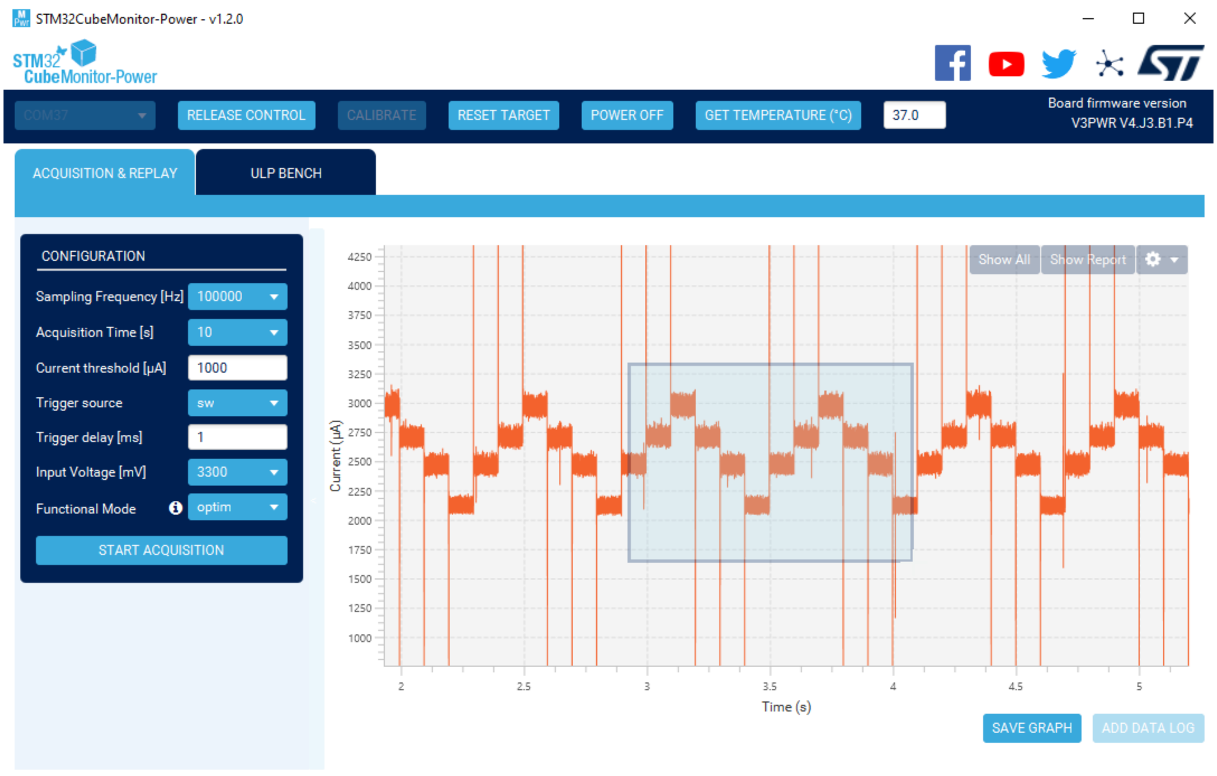


Board firmware version V3PWR V4.J2.B1.P4

Sampling Frequency [Hz]	100000
Acquisition Time [s]	10
Current threshold [µA]	1000
Trigger source	sw
Trigger delay [ms]	1
Input Voltage [mV]	3300
Functional Mode	optim

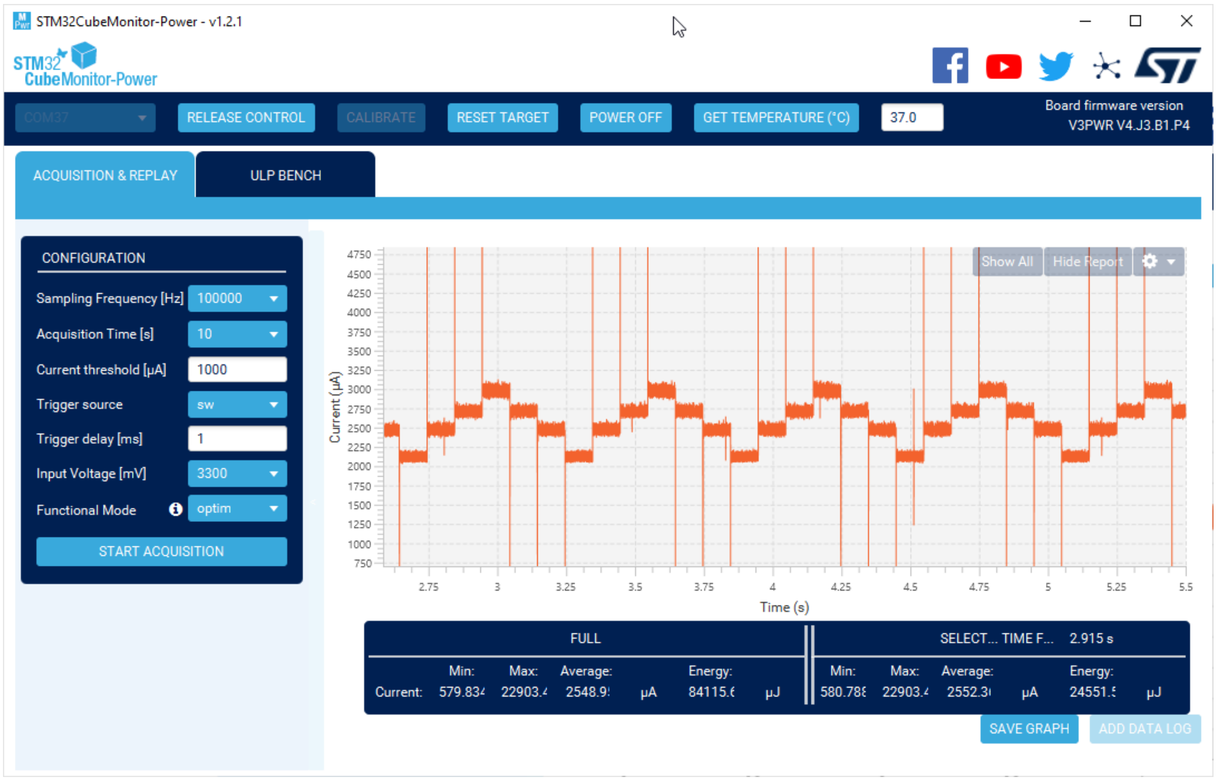
4. The graphical interface allows you to zoom and pan inside the current waveform acquisition.

Figure 14. **STM32CubeMonitor-Power zoom and pan features**



5. Finally, the [**Show Report**] button allows access to statistical data (minimum, maximum, average, etc.). If necessary, the acquisition waveform can be saved or reloaded with the [**SAVE GRAPH**] and [**ADD DATALOG**] buttons.

Figure 15. **STM32CubeMonitor-Power statistical report**



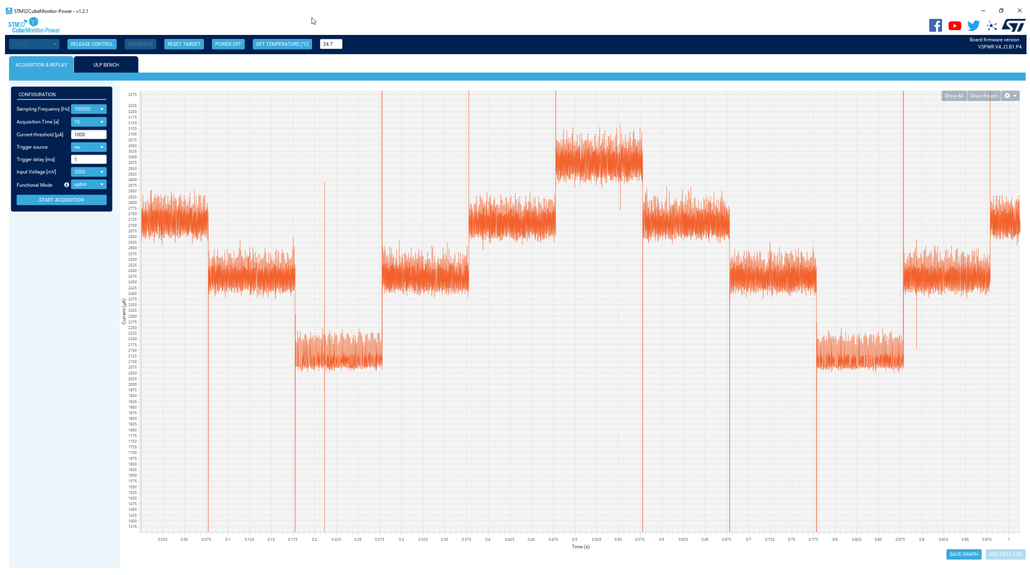
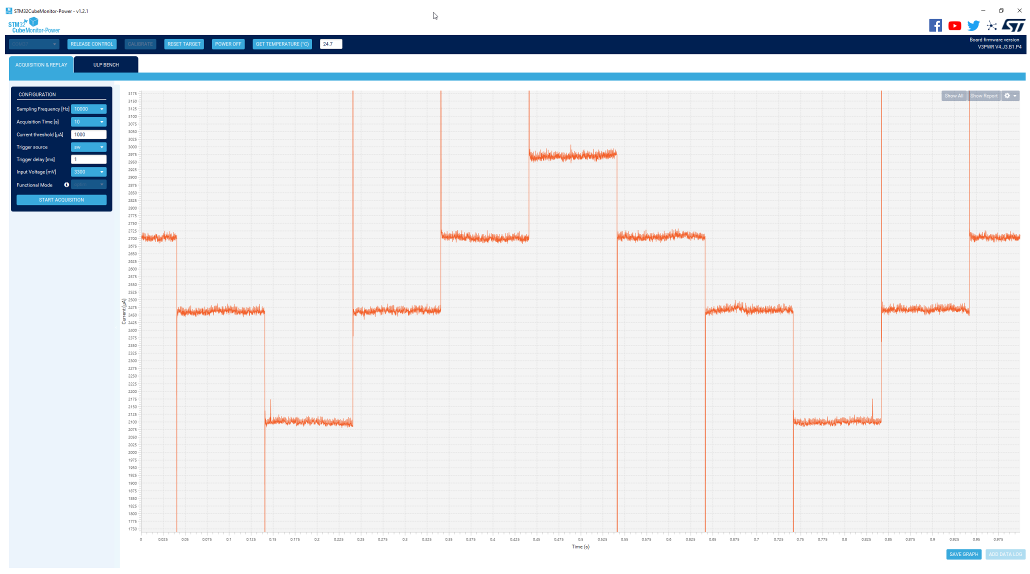
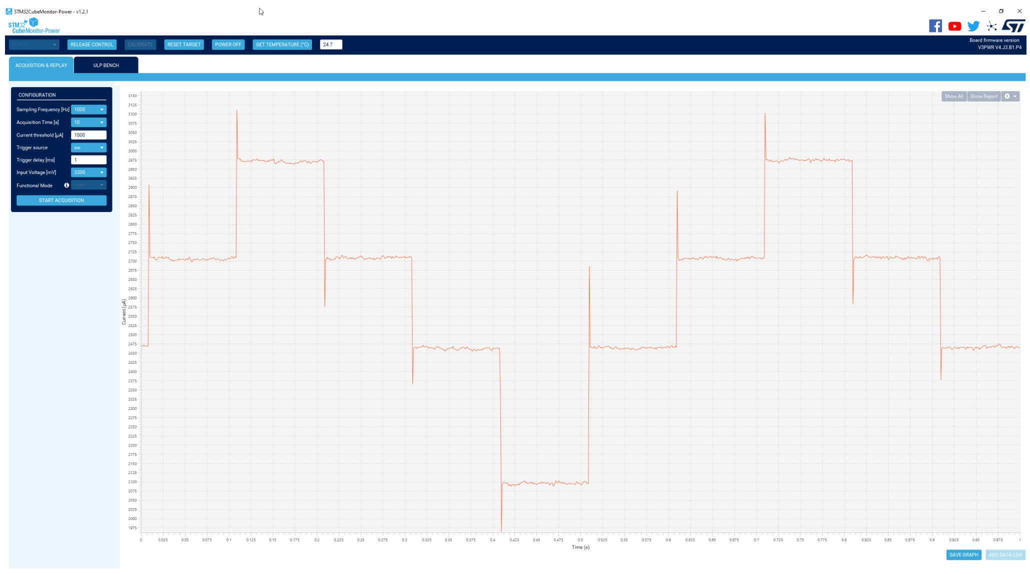
Sampling Frequency [Hz]	100000
Acquisition Time [s]	10
Current threshold [µA]	1000
Trigger source	sw
Trigger delay [ms]	1
Input Voltage [mV]	3300
Functional Mode	optim

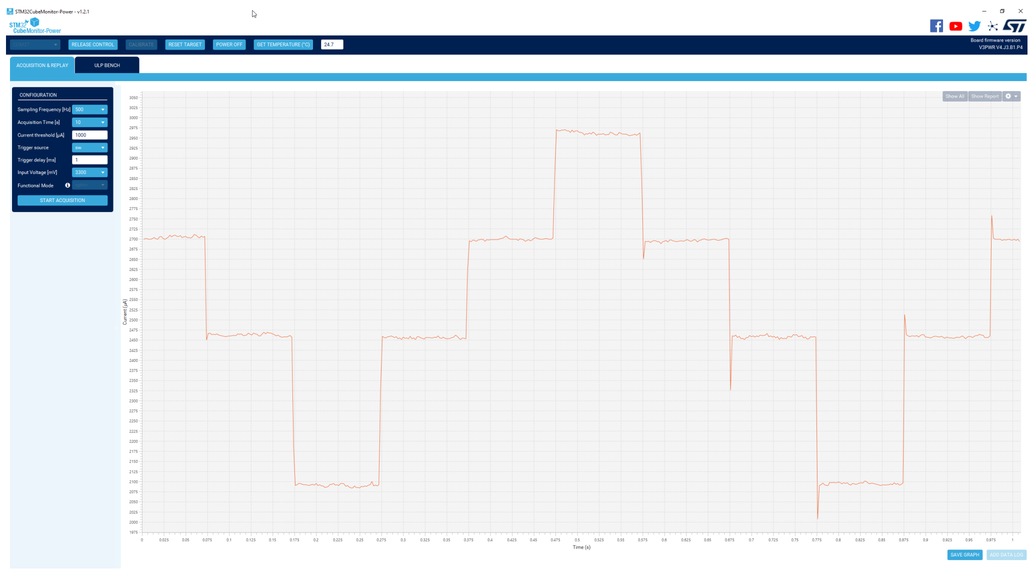
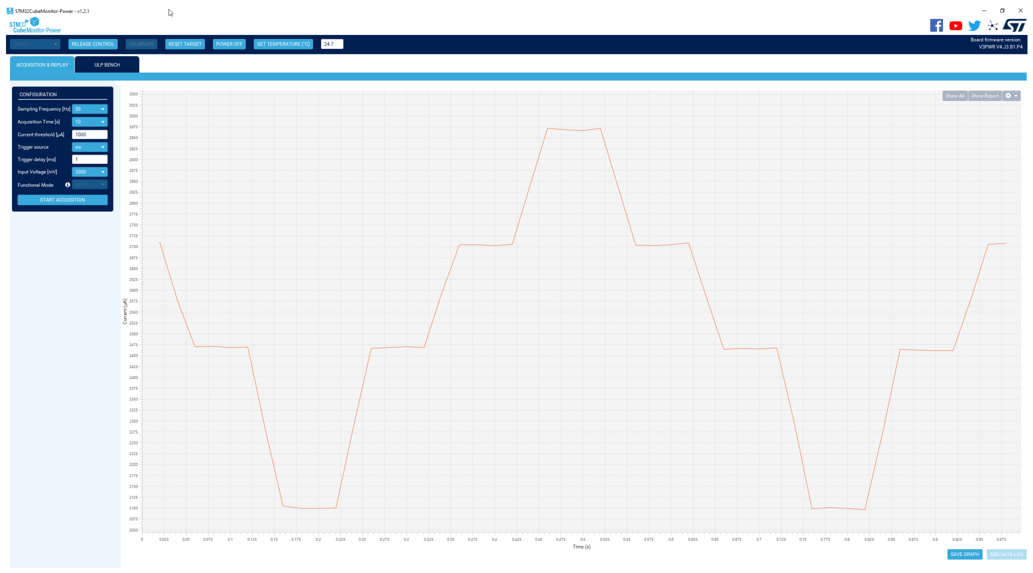
	FULL						SELECT... TIME F... 2.915 s					
	Min:	Max:	Average:		Energy:		Min:	Max:	Average:		Energy:	
Current:	579.834	22903.4	2548.9	µA	84115.6	µJ	580.788	22903.4	2552.3	µA	24551.5	µJ

5.1.2 Sampling frequency selection example

Set the sampling frequency to 100 kHz to ensure the correct acquisition of fast current variation or current transient, but you can lower it to filter the noise and observe small current variations and reduce the acquisition data size. In the following example, 10 kHz or 1 kHz seems a convenient setting for balancing between noise filtering and transient observation. Below the 1-kHz sampling frequency, the acquired signal is undersampled and the current edges are filtered.

Table 4. **Sampling frequency comparison**

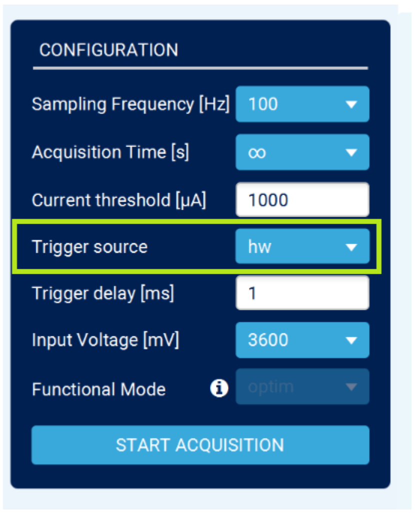
100 kHz	 STM32CubeMonitor-Power - v1.2.1 CONFIGURATION Sampling Frequency [Hz]: 100000 Acquisition Time [s]: 10 Current threshold [µA]: 1000 Trigger source: sw Trigger delay [ms]: 1 Input Voltage [mV]: 3300 Functional Mode: optim GET TEMPERATURE (°C): 24.7
10 kHz	 STM32CubeMonitor-Power - v1.2.1 CONFIGURATION Sampling Frequency [Hz]: 10000 Acquisition Time [s]: 10 Current threshold [µA]: 1000 Trigger source: sw Trigger delay [ms]: 1 Input Voltage [mV]: 3300 Functional Mode: optim GET TEMPERATURE (°C): 24.7
1 kHz	 STM32CubeMonitor-Power - v1.2.1 CONFIGURATION Sampling Frequency [Hz]: 1000 Acquisition Time [s]: 10 Current threshold [µA]: 1000 Trigger source: sw Trigger delay [ms]: 1 Input Voltage [mV]: 3300 Functional Mode: optim GET TEMPERATURE (°C): 24.7

500 Hz	 STM32CubeMonitor-Power - v1.2.1 CONFIGURATION Sampling Frequency [Hz]: 500 Acquisition Time [s]: 10 Current threshold [µA]: 1000 Trigger source: sw Trigger delay [ms]: 1 Input Voltage [mV]: 3300 Functional Mode: optim
50 Hz	 STM32CubeMonitor-Power - v1.2.1 CONFIGURATION Sampling Frequency [Hz]: 50 Acquisition Time [s]: 10 Current threshold [µA]: 1000 Trigger source: sw Trigger delay [ms]: 1 Input Voltage [mV]: 3300 Functional Mode: optim

5.1.3 Input and output triggers TGI and TGO

The trigger input TGI signal from the bridge connector can be used to start an acquisition from an external event. Selecting the hardware ("hw") trigger source from the configuration window in STM32CubeMonitor-Power enables this hardware feature.

Figure 16. **Selecting a hardware trigger source from the STM32CubeMonitor-Power configuration window**



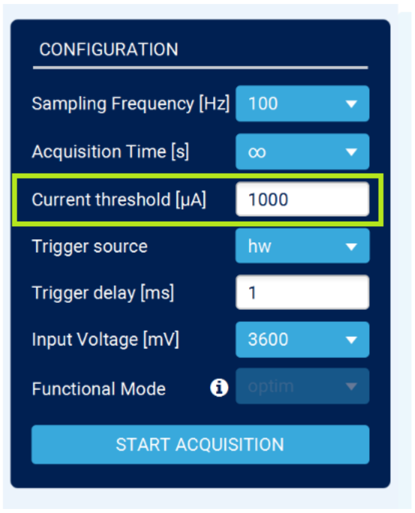
CONFIGURATION

Sampling Frequency [Hz]	100
Acquisition Time [s]	∞
Current threshold [µA]	1000
Trigger source	hw
Trigger delay [ms]	1
Input Voltage [mV]	3600
Functional Mode	optim

The input trigger detection is functional on rising and falling edges. Because it is a level shifter input, it is necessary to apply a reference voltage to the T_VCC pin and to connect a ground reference to the GND pin to ensure correct functionality. (If an STDC14 debug cable is connected between STLINK-V3PWR and the target application, the T_VCC reference and GND are polarized from it.)

The trigger output TGO signals from the bridge connector generate an output event when a sample of the current acquisition waveform is above a current value threshold. The output trigger signal on TGO is set to a high level in real time when the actual acquired current sample value is above the programmed threshold. TGO is set to a low level when the actual acquired current sample value is below the threshold. In addition, TGO is the output signal of a level shifter, so it is necessary to apply a reference voltage to the T_VCC pin and to connect a ground reference to the GND pin to ensure correct functionality. If an STDC14 debug cable is connected between STLINK-V3PWR and the target application, the T_VCC reference and GND are connected through this cable.

Figure 17. **Selecting a current value threshold from the STM32CubeMonitor-Power configuration window**



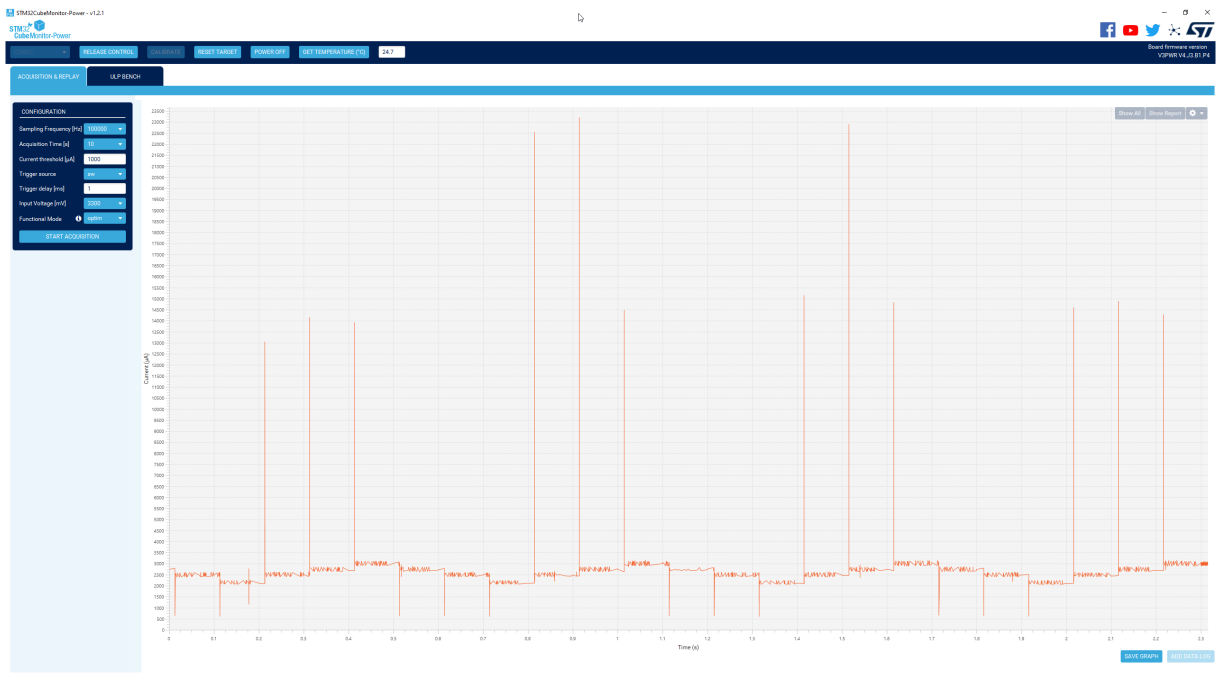
CONFIGURATION

Sampling Frequency [Hz]	100
Acquisition Time [s]	∞
Current threshold [µA]	1000
Trigger source	hw
Trigger delay [ms]	1
Input Voltage [mV]	3600
Functional Mode	optim

5.1.4 Display optimization feature

To optimize the graphical user interface of STM32CubeMonitor-Power, the acquisition waveform is graphically undersampled using maximum and peak values to provide a faster display of the curve. The display looks distorted, but this is normal behavior. If you zoom in on the waveform points, the graphical optimization is reduced, and the number of visible samples increases.

Figure 18. Current waveform with graphical optimization (show all)


STM32CubeMonitor-Power - v1.2.1

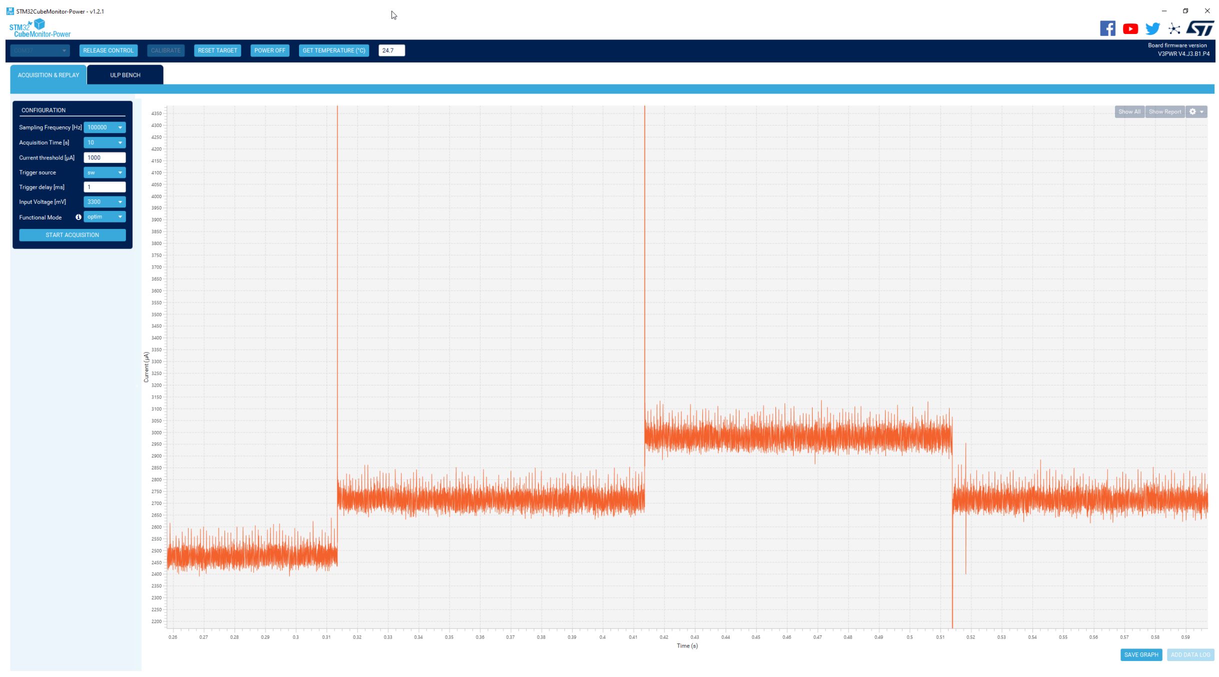
Board firmware version V3PWR V4.J3.B1.P4

CONFIGURATION

Sampling Frequency [Hz] 100000
Acquisition Time [s] 10
Current threshold [µA] 1000
Trigger source sw
Trigger delay [ms] 1
Input Voltage [mV] 3300
Functional Mode optim

GET TEMPERATURE (°C) 24.7

Figure 19. Same current waveform with graphical optimization (zoom in)


STM32CubeMonitor-Power - v1.2.1

Board firmware version V3PWR V4.J3.B1.P4

CONFIGURATION

Sampling Frequency [Hz] 100000
Acquisition Time [s] 10
Current threshold [µA] 1000
Trigger source sw
Trigger delay [ms] 1
Input Voltage [mV] 3300
Functional Mode optim

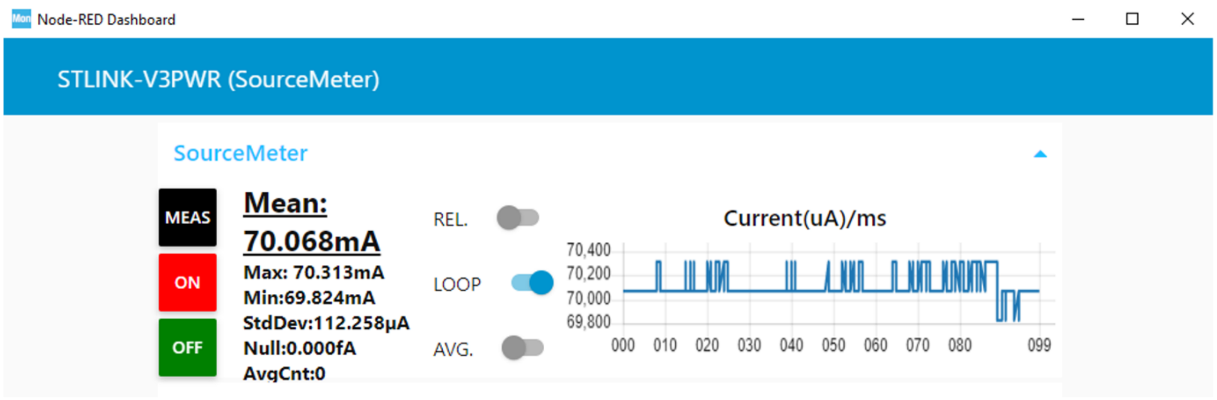
GET TEMPERATURE (°C) 24.7

5.2 STM32CubeMonitor

STM32CubeMonitor software is based on Node-RED® technology, and helps design and create graphical interfaces with elements like buttons, label boxes, and graphs to control and display any type of data. You can easily develop a Node-RED® flow with STM32CubeMonitor to manage a power Virtual COM port and retrieve the acquisition data from the STLINK-V3PWR debug probe (the STLINK-V3PWR power interface is based on a specific syntax described in *UM2269: Getting started with PowerShield firmware*).

These data can be displayed on a graphical dashboard to highlight various types of numerics, gauges, or charts relative to the current consumption waveform. There are buttons to start the current acquisition, and turn the VOUT power on or off. Some functionalities are controlled with switches, like the point-per-point waveform average to filter high frequency noise, the loop mode to perform continuous acquisitions, and the simple constant subtraction calculation to focus on relative measurements. Refer to the power measurement section in the STM32 wiki portal for more information.

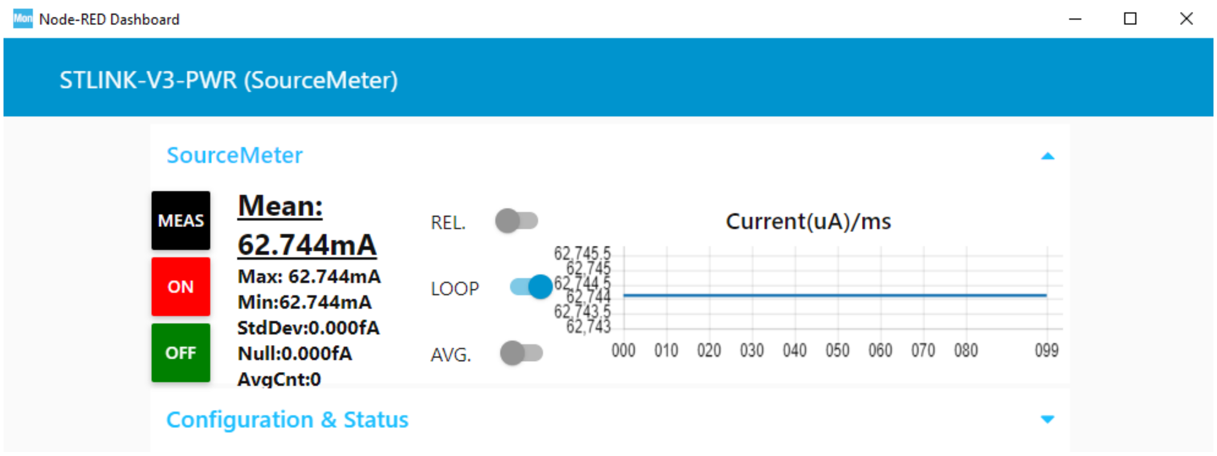
Figure 20. STM32CubeMonitor dashboard with STLINK-V3PWR example



Mean:
70.068mA
Max: 70.313mA
Min:69.824mA
StdDev:112.258µA
Null:0.000fA
AvgCnt:0

The relative measurement feature is very useful to measure the current consumption difference between two application states. The example below shows the power consumption difference when the application is running at 160 MHz or 24 MHz.

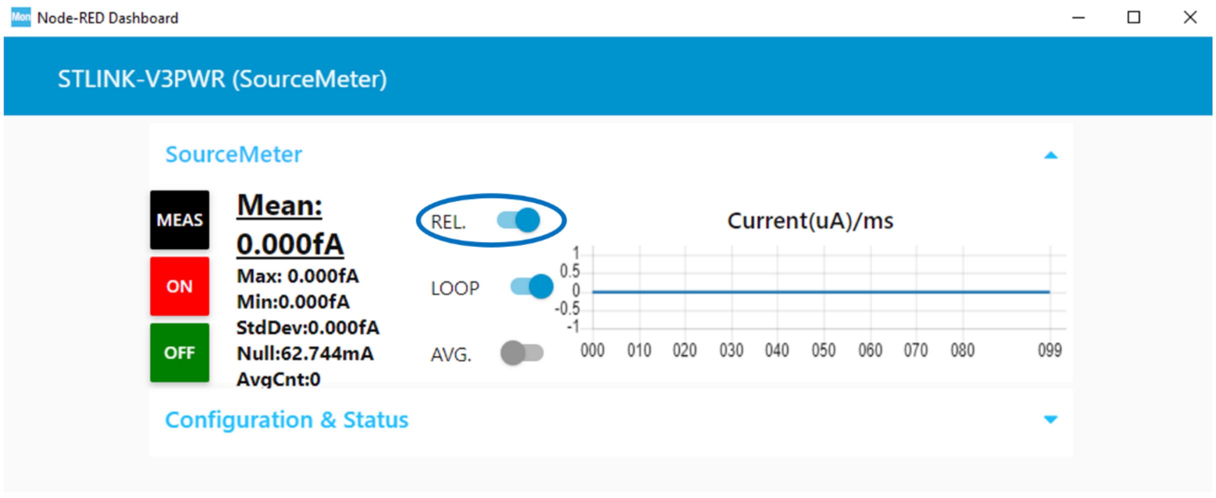
Figure 21. Dashboard STLINK-V3PWR, application running at 160 MHz



Mean:
62.744mA
Max: 62.744mA
Min:62.744mA
StdDev:0.000fA
Null:0.000fA
AvgCnt:0

Activation of the relative mode creates a constant compensation offset and gets a relative measurement. The result is 62.744 mA - 62.744 mA = 0 mA from this measuring point.

Figure 22. **Dashboard STLINK-V3PWR, application running at 160 MHz (relative mode enabled)**


Node-RED Dashboard

STLINK-V3PWR (SourceMeter)

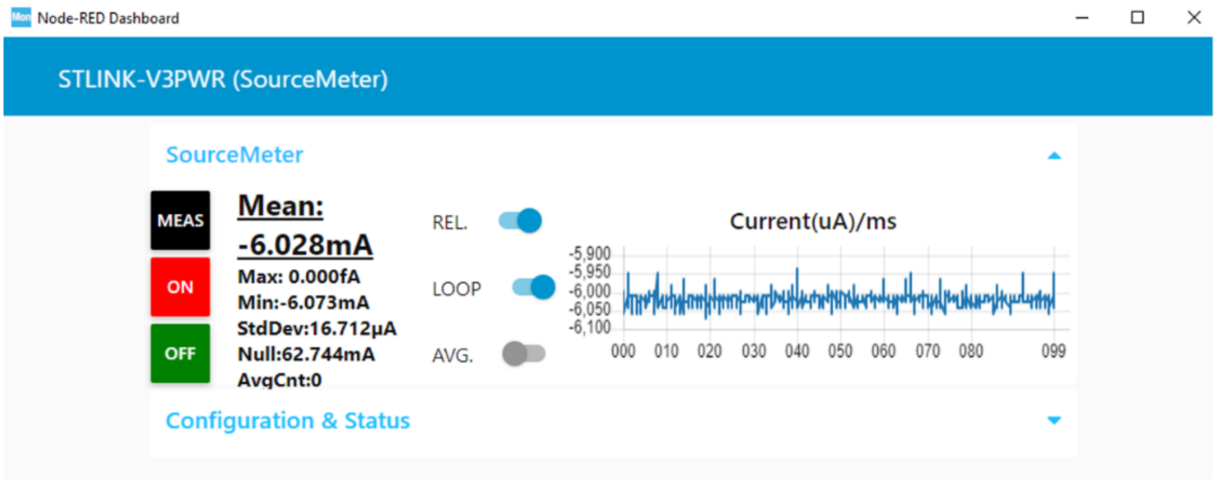
SourceMeter

Mean:
0.000fA
Max: 0.000fA
Min:0.000fA
StdDev:0.000fA
Null:62.744mA
AvgCnt:0

Configuration & Status

When the application frequency is reduced to 24 MHz while the relative mode is enabled, the graphical interface shows an easily visible current consumption reduction of 6.028 mA compared to the 160 MHz configuration.

Figure 23. **Dashboard STLINK-V3PWR, application running at 24 MHz (relative mode enabled)**


Node-RED Dashboard

STLINK-V3PWR (SourceMeter)

SourceMeter

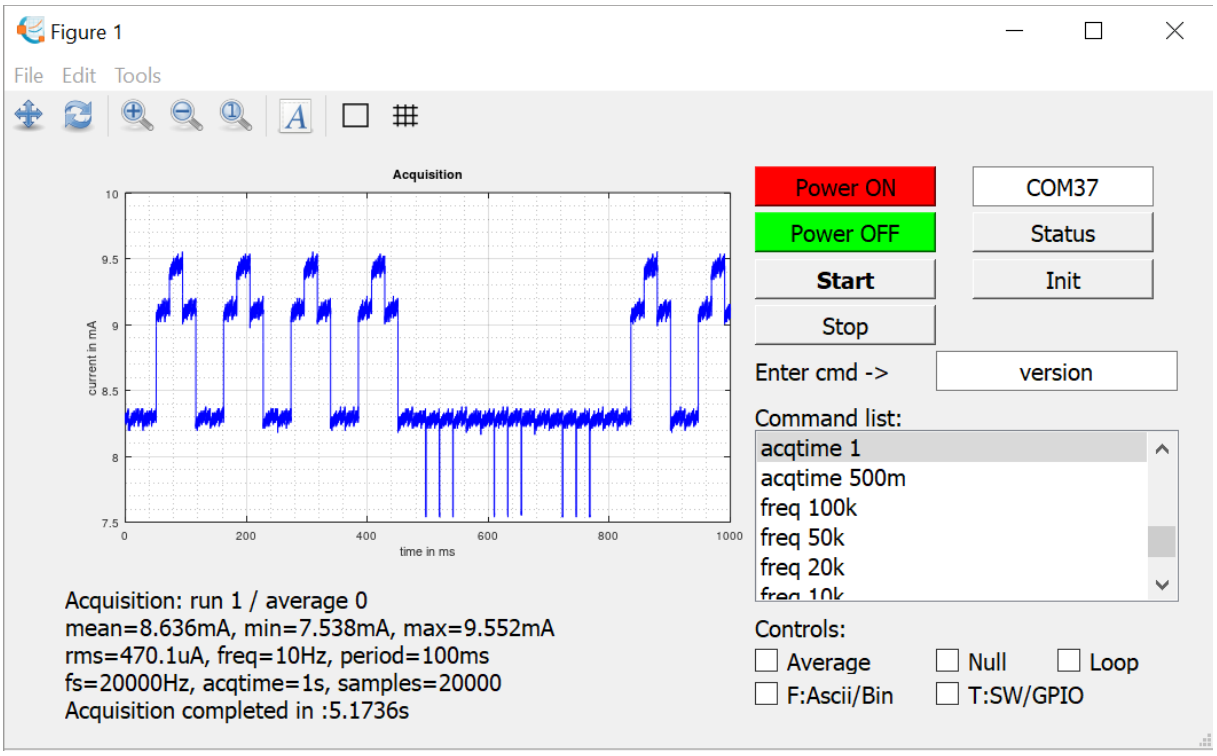
Mean:
-6.028mA
Max: 0.000fA
Min:-6.073mA
StdDev:16.712µA
Null:62.744mA
AvgCnt:0

Configuration & Status

5.3 GNU Octave 7.3

Another example of a graphical interface with STLINK-V3PWR is the GNU Octave software. Using the commands described in *UM2269: Getting started with PowerShield firmware* and the Octave serial port functions, you can design a custom graphical interface for your application power consumption analysis. Refer to the power measurement section in the STM32 wiki portal for more information.

Figure 24. GNU Octave STLINK-V3PWR, power consumption waveform acquisition



```
Acquisition: run 1 / average 0
mean=8.636mA, min=7.538mA, max=9.552mA
rms=470.1uA, freq=10Hz, period=100ms
fs=20000Hz, acqtime=1s, samples=20000
Acquisition completed in :5.1736s
```

6 STLINK-V3PWR debug only IDE software tools

STM32CubeIDE can be used for debugging only with the STLINK-V3PWR probe. The supply source should be managed separately, using STLINK-V3PWR and STM32CubeMonitor-Power, or the embedded application supply.

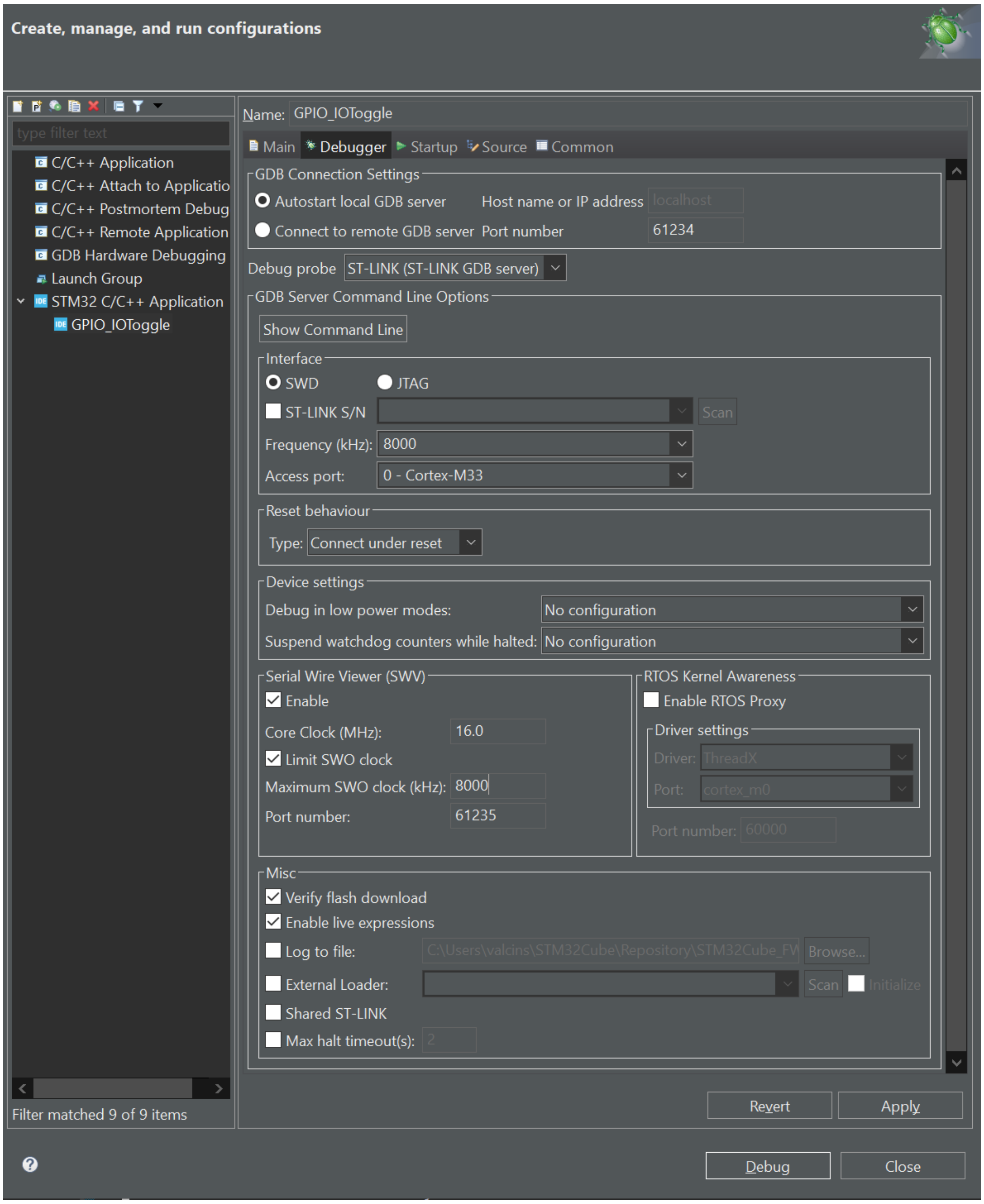
6.1 STM32CubeIDE

After the STM32 target is powered on and the debug cable attached to it, the STM32CubeIDE debug session can be configured and started.

6.1.1 Debug configuration and run

Use the [**Run**] menu and then [**Debug Configurations**] to open the Settings window. Either OpenOCD or GDB serverdrivers can be used for debugging. Use an SWD interface frequency around 8 MHz/10 MHz to ensure compatibility with STLINK-V3PWR. In the following configuration, the Serial Wire Viewer feature is enabled to configure and acquire the debug trace content. It is mandatory that the core clock frequency matches the application firmware configuration.

Figure 25. **STM32CubeIDE Debug Configuration window**


Create, manage, and run configurations

type filter text
- C/C++ Application
- C/C++ Attach to Applicatio
- C/C++ Postmortem Debug
- C/C++ Remote Application
- GDB Hardware Debugging
- Launch Group
- STM32 C/C++ Application
  - GPIO_IOToggle

Filter matched 9 of 9 items

Name: GPIO_IOToggle

Main | Debugger | Startup | Source | Common

GDB Connection Settings
- [✓ Autostart local GDB server] Host name or IP address: localhost
- Connect to remote GDB server Port number: 61234

Debug probe: ST-LINK (ST-LINK GDB server)

GDB Server Command Line Options

Show Command Line

Interface
- [✓ SWD] JTAG
- ST-LINK S/N: Scan
- Frequency (kHz): 8000
- Access port: 0 - Cortex-M33

Reset behaviour
- Type: Connect under reset

Device settings
- Debug in low power modes: No configuration
- Suspend watchdog counters while halted: No configuration

Serial Wire Viewer (SWV)
- ✓ Enable
- Core Clock (MHz): 16.0
- ✓ Limit SWO clock
- Maximum SWO clock (kHz): 8000
- Port number: 61235

RTOS Kernel Awareness
- Enable RTOS Proxy
- Driver settings
  - Driver: ThreadX
  - Port: cortex_m0
- Port number: 60000

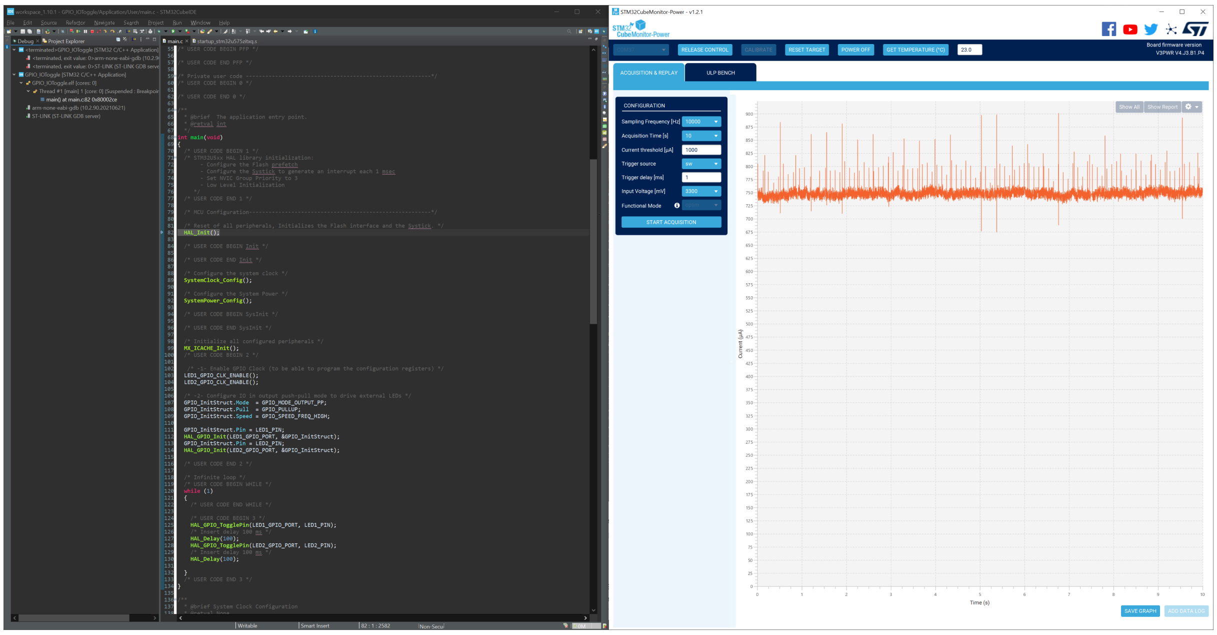
Misc
- ✓ Verify flash download
- ✓ Enable live expressions
- Log to file: C:\Users\valcins\STM32Cube\Repository\STM32Cube_FW Browse...
- External Loader: Scan Initialize
- Shared ST-LINK
- Max halt timeout(s): 2

Revert Apply

Debug Close

Once the debug has been configured, you can launch the same window for the debug session by clicking on the [**Debug**] button. The firmware execution is paused after the `main()` function; select [**Resume**] from the [**Run**] menu to continue the firmware execution. Current consumption can be observed step by step, parallel to the code execution.

Figure 26. STM32CubeIDE Debug window side by side with STM32CubeMonitor-Power



```
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* STM32U5xx HAL library initialization:
       - Configure the Flash prefetch
       - Configure the Systick to generate an interrupt each 1 msec
       - Set NVIC Group Priority to 3
       - Low Level Initialization
     */
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* Configure the System Power */
  SystemPower_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_ICACHE_Init();
  /* USER CODE BEGIN 2 */

   /* -1- Enable GPIO Clock (to be able to program the configuration registers) */
  LED1_GPIO_CLK_ENABLE();
  LED2_GPIO_CLK_ENABLE();

  /* -2- Configure IO in output push-pull mode to drive external LEDs */
  GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull  = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;

  GPIO_InitStruct.Pin = LED1_PIN;
  HAL_GPIO_Init(LED1_GPIO_PORT, &GPIO_InitStruct);
  GPIO_InitStruct.Pin = LED2_PIN;
  HAL_GPIO_Init(LED2_GPIO_PORT, &GPIO_InitStruct);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_GPIO_TogglePin(LED1_GPIO_PORT, LED1_PIN);
    /* Insert delay 100 ms */
    HAL_Delay(100);
    HAL_GPIO_TogglePin(LED2_GPIO_PORT, LED2_PIN);
    /* Insert delay 100 ms */
    HAL_Delay(100);

  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
```

STM32CubeMonitor-Power - v1.2.1

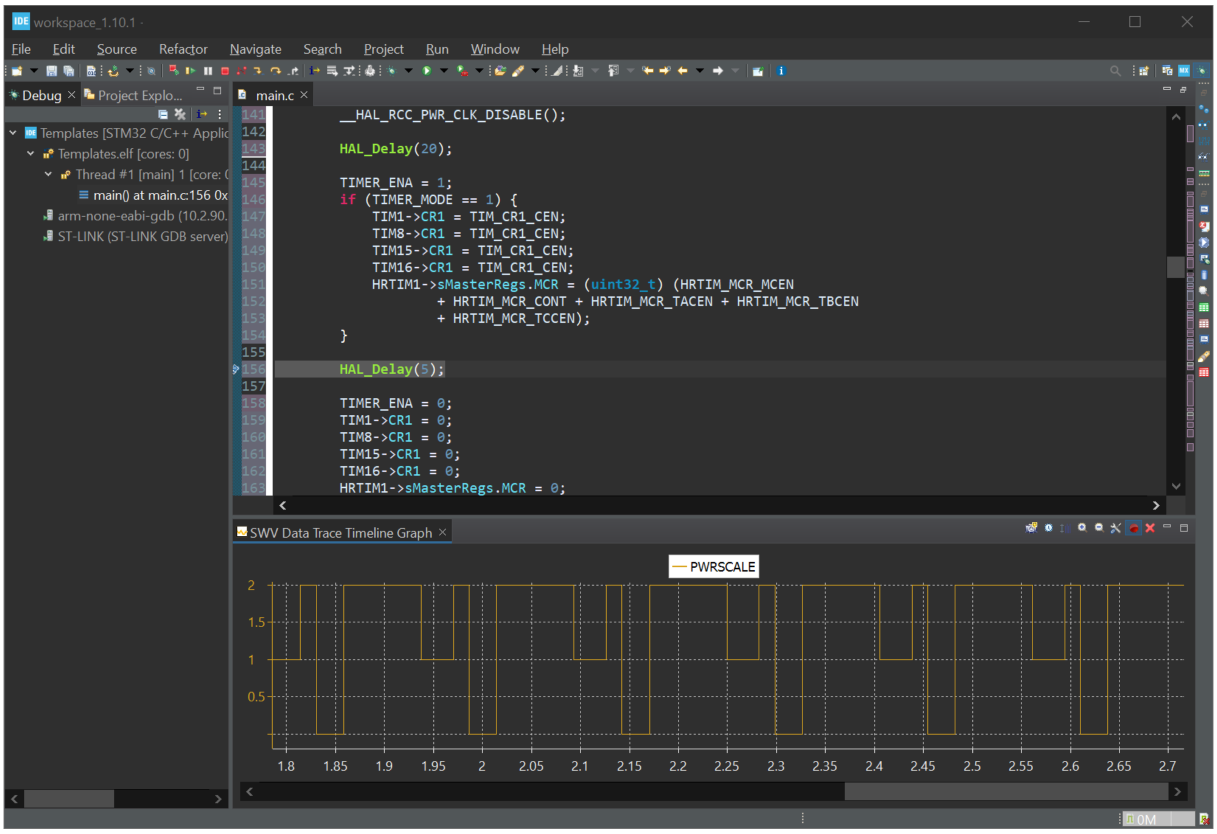
Board firmware version V3PWR V4.J3.B1.P4

CONFIGURATION

Sampling Frequency [Hz]: 10000
Acquisition Time [s]: 10
Current threshold [µA]: 1000
Trigger source: sw
Trigger delay [ms]: 1
Input Voltage [mV]: 3300

Moreover, a global variable can be plotted using the debug interface in the Serial Wire Viewer Data Trace Timeline Graph window.

Figure 27. STM32CubeIDE SWV Data Trace Timeline Graph window



```
        __HAL_RCC_PWR_CLK_DISABLE();

        HAL_Delay(20);

        TIMER_ENA = 1;
        if (TIMER_MODE == 1) {
            TIM1->CR1 = TIM_CR1_CEN;
            TIM8->CR1 = TIM_CR1_CEN;
            TIM15->CR1 = TIM_CR1_CEN;
            TIM16->CR1 = TIM_CR1_CEN;
            HRTIM1->sMasterRegs.MCR = (uint32_t) (HRTIM_MCR_MCEN
                    + HRTIM_MCR_CONT + HRTIM_MCR_TACEN + HRTIM_MCR_TBCEN
                    + HRTIM_MCR_TCCEN);
        }

        HAL_Delay(5);

        TIMER_ENA = 0;
        TIM1->CR1 = 0;
        TIM8->CR1 = 0;
        TIM15->CR1 = 0;
        TIM16->CR1 = 0;
        HRTIM1->sMasterRegs.MCR = 0;
```

7 STLINK-V3PWR debug and power software tools

STLINK-V3PWR has a specific feature that allows the synchronization of the current waveform acquisition with the debug trace content based on the STM32 target timestamps during the debugging session. This feature requires the availability of a Serial Wire Output pin to retrieve the debug data trace, which also includes the power synchronization data via ITM.

The IDE software tools Arm® µVision® (from version V5.39) and IAR Embedded Workbench® for Arm® (from version V9.32) are compatible with the STLINK-V3PWR debug and power features, including synchronized acquisition.

7.1 IAR Embedded Workbench IDE (from version V9.32.2)

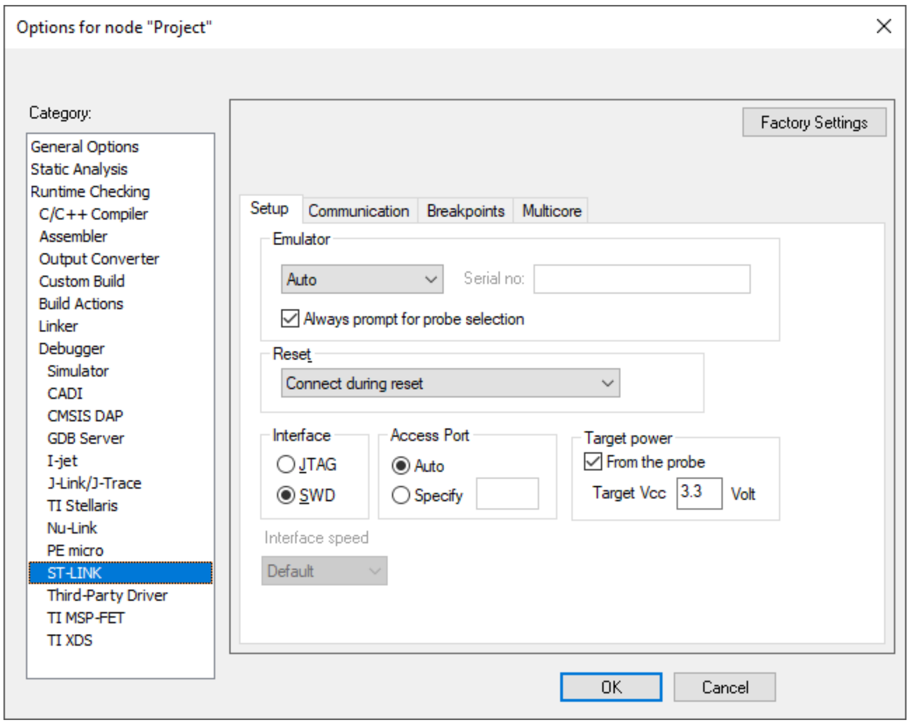
7.1.1 Power configuration

The power configuration is done using two separate windows: the Debugger options window and the Power log setup window from the ST-LINK menu. The ST-LINK menu is fully accessible once the debug session has started.

Warning: *When the debug session starts, the STLINK-V3PWR VOUT and VAUX supplies are automatically turned on. When the debug session ends, the STLINK-V3PWR VOUT and VAUX supplies are automatically turned off.*

IAR Embedded Workbench® (from version V9.32.2) includes the STLINK-V3PWR debugger drivers and allows the user to configure the power settings in the Project Debugger Options window. It allows you to configure the debug interface, the supply voltage, and the CPU and SWO clock frequencies. Select [**Auto Emulator**] and checking [**Always prompt for probe selection**]. Target power sets the target application voltage on the VOUT and VAUX supplies

Figure 28. IAR Embedded Workbench IDE STLINK-V3PWR main options window

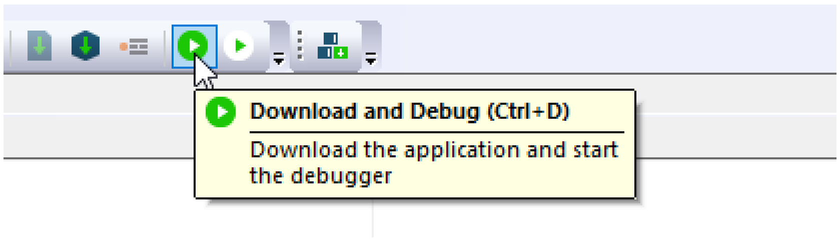


Options for node "Project"

Category: General Options, Static Analysis, Runtime Checking, C/C++ Compiler, Assembler, Output Converter, Custom Build, Build Actions, Linker, Debugger, Simulator, CADI, CMSIS DAP, GDB Server, I-jet, J-Link/J-Trace, TI Stellaris, Nu-Link, PE micro, ST-LINK (selected), Third-Party Driver, TI MSP-FET, TI XDS

Setup tab:
- Emulator: Auto; Serial no: (empty)
- ☑ Always prompt for probe selection
- Reset: Connect during reset
- Interface: ○ JTAG ● SWD
- Access Port: ● Auto ○ Specify
- Target power: ☑ From the probe; Target Vcc 3.3 Volt
- Interface speed: Default

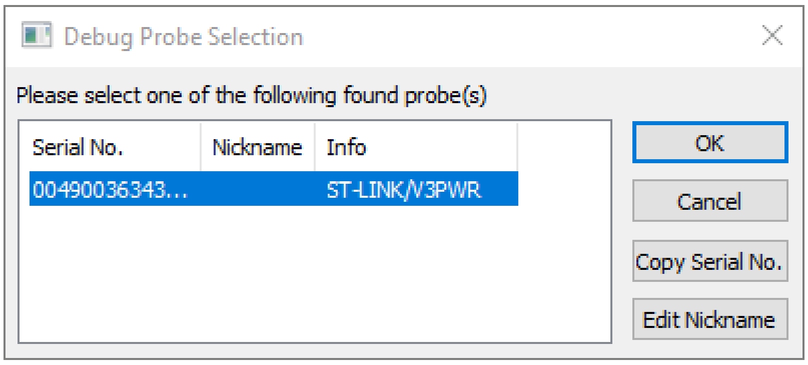
Figure 29. Starting the IAR Embedded Workbench IDE debug session



Download and Debug (Ctrl+D)

Download the application and start the debugger

Figure 30. IAR Embedded Workbench IDE STLINK-V3PWR debug probe selection window

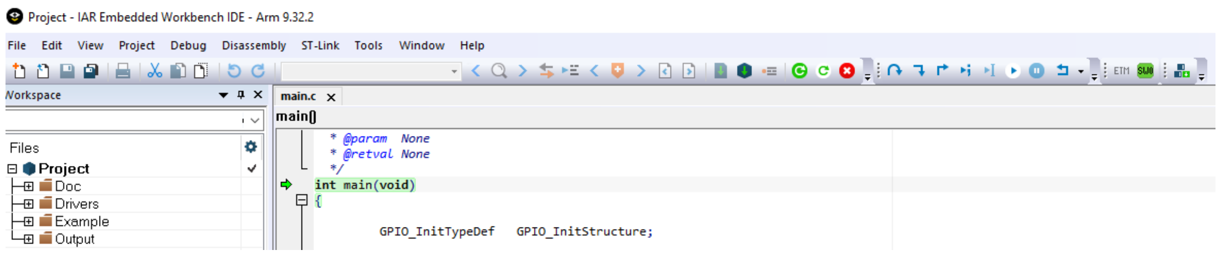


Debug Probe Selection

Please select one of the following found probe(s)

Serial No.	Nickname	Info
00490036343...		ST-LINK/V3PWR

Figure 31. IAR Embedded Workbench IDE debug session break at the main() function

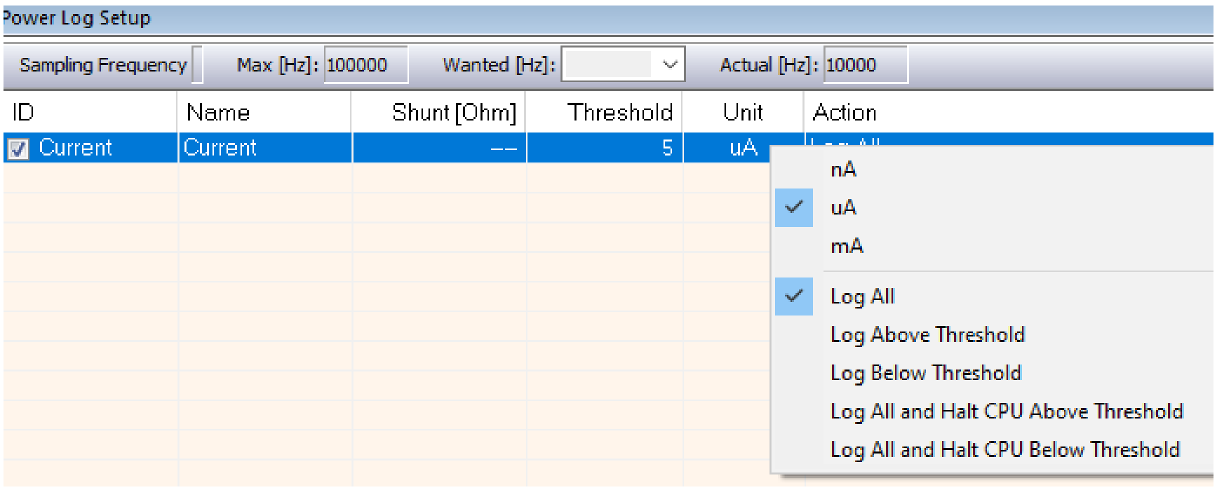


```
  * @param  None
  * @retval None
  */
int main(void)
{

        GPIO_InitTypeDef    GPIO_InitStructure;
```

From the ST-LINK menu, open the Power Log Setup window to configure:

- The sampling frequency (Hz): set the current waveform sampling frequency.
- The current value threshold: set the current threshold and action associated (check [**Log All**], etc.). Select [**Log All**] and [**Halt CPU Above Threshold**] to halt the code execution automatically above a given threshold.

Figure 32. IAR Embedded Workbench IDE Power Log Setup window


Power Log Setup

Sampling Frequency | Max [Hz]: 100000 | Wanted [Hz]: | Actual [Hz]: 10000

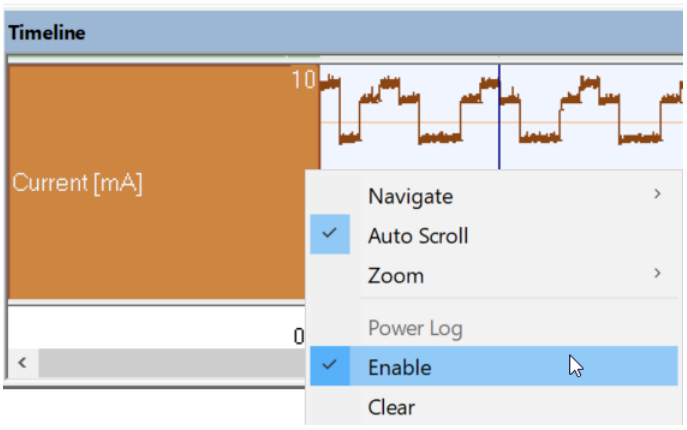
ID	Name	Shunt [Ohm]	Threshold	Unit	Action
✓ Current	Current	--	5	uA	

nA
✓ uA
mA
✓ Log All
Log Above Threshold
Log Below Threshold
Log All and Halt CPU Above Threshold
Log All and Halt CPU Below Threshold

7.1.2 Visualizing the power current consumption waveform

Open the Timeline window from the ST-LINK menu and enable the [**Power Log**] feature.

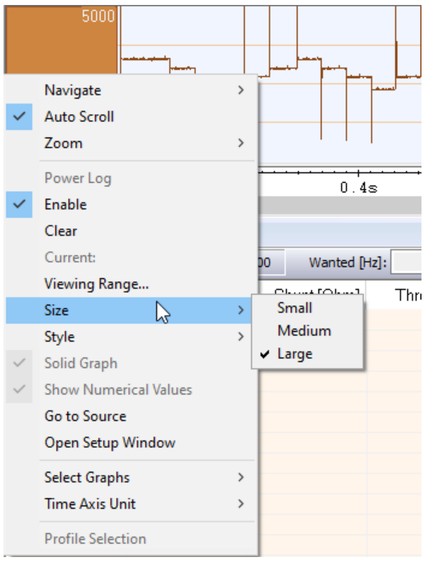
Figure 33. IAR Embedded Workbench IDE Timeline window



Apply the following configuration for better waveform visualization:

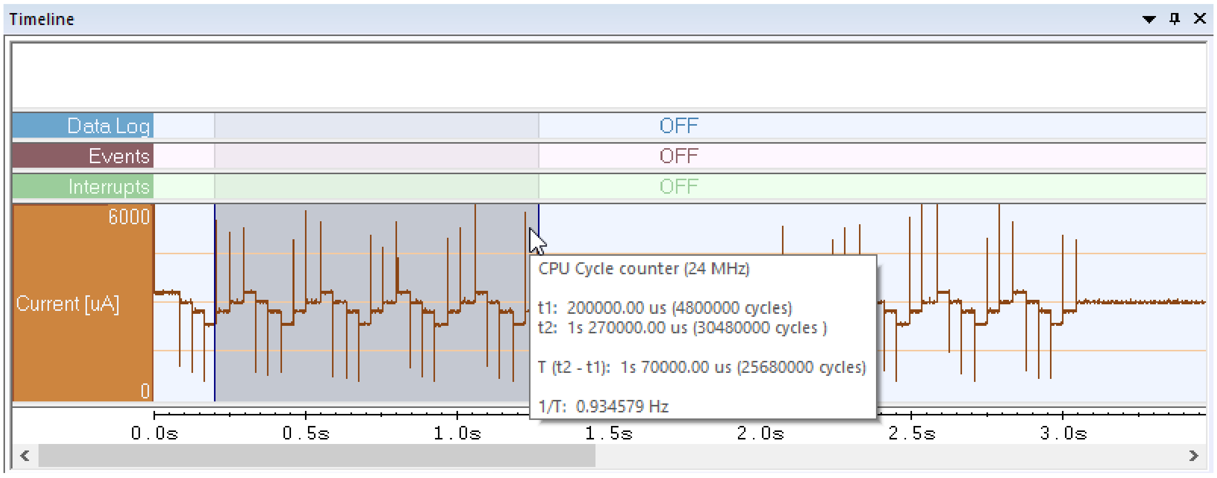
- Size: large.
- Style: linear.
- Viewing range: custom, to fit the current waveform amplitude.

Figure 34. **IAR Embedded Workbench IDE timeline configuration**



The Power Timeline configuration is complete and ready for waveform analysis.

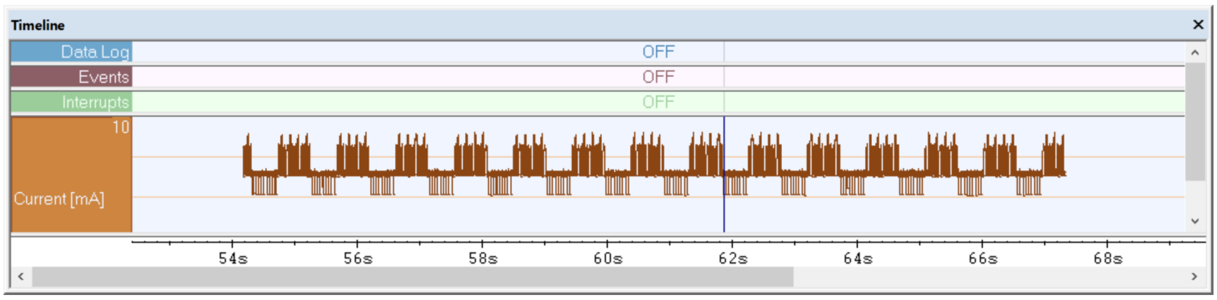
Figure 35. **IAR Embedded Workbench IDE timeline power observation**



```
CPU Cycle counter (24 MHz)

t1:  200000.00 us (4800000 cycles)
t2:  1s 270000.00 us (30480000 cycles )

T (t2 - t1):  1s 70000.00 us (25680000 cycles)

1/T:  0.934579 Hz
```

Note: When you use the zoom feature or scrolling in the IAR Embedded Workbench® IDE timeline, the displayed current waveform is limited to a 15-second duration.

Figure 36. **IAR Embedded Workbench IDE timeline power observation duration after zoom or scrolling**

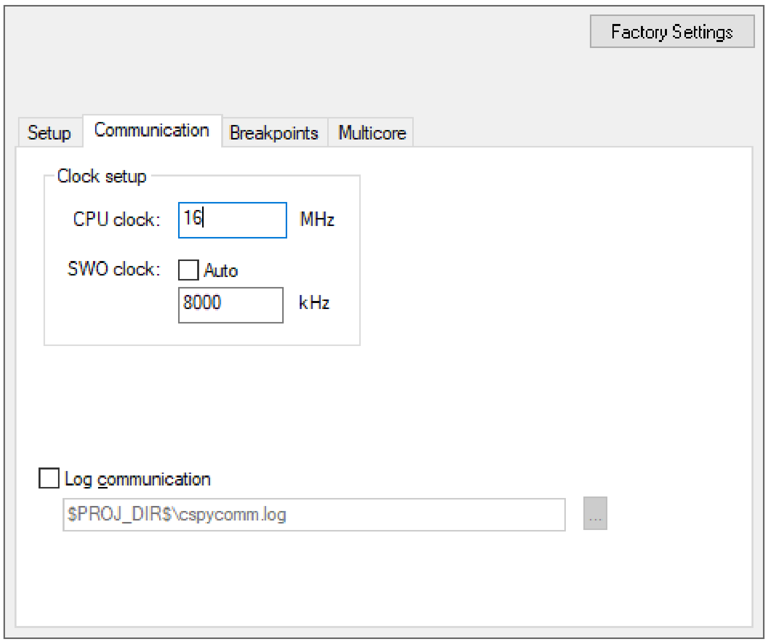


7.1.3 Trace configuration (project options)

The correct trace configuration is mandatory to ensure the correct decoding during the debug session. The following parameters contain the minimum requirements to perform a current waveform acquisition synchronized with the debug trace. The Communication tab in the Project options window should be set in line with the firmware clock configuration:

- CPU clock: exact target CPU frequency value (refer to application firmware configuration).
- SWO clock: value to configure the SWO clock. The recommended frequency for theSTLINK-V3PWR SWO clock is 8~12 MHz range (~10 MHz to be configured).

Figure 37. IAR Embedded Workbench IDE Communication tab in the project options



Factory Settings

Setup | Communication | Breakpoints | Multicore

Clock setup

CPU clock: 16 MHz

SWO clock: ☐ Auto

8000 kHz

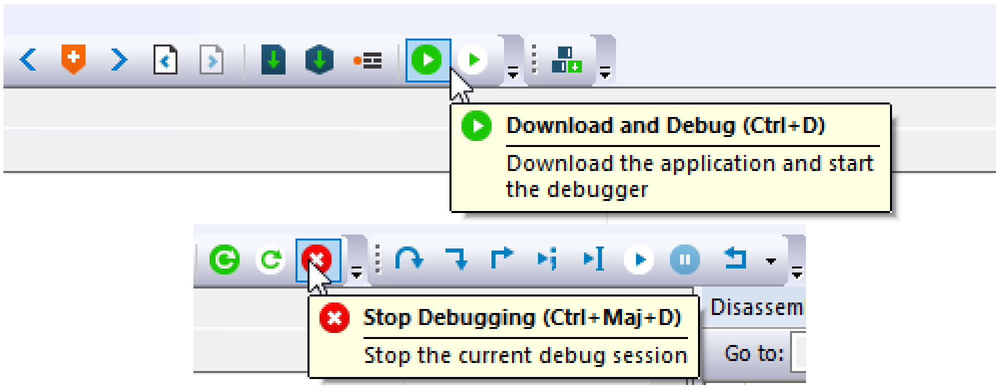
☐ Log communication

$PROJ_DIR$\cspycomm.log

7.1.4 Start/stop debug session

To start or stop a debug session, simply use the dedicated icon shortcut:

Figure 38. IAR Embedded Workbench IDE start/stop debug session



Download and Debug (Ctrl+D)

Download the application and start the debugger

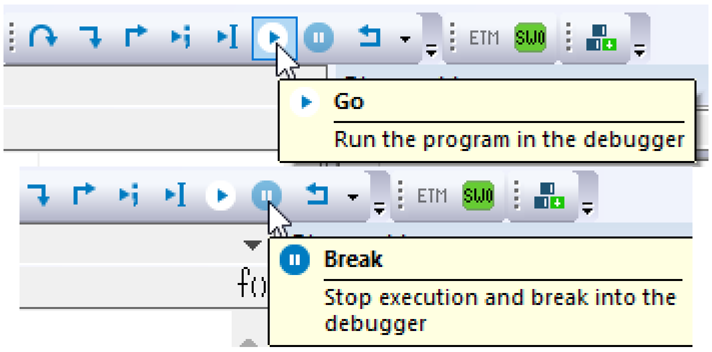
Stop Debugging (Ctrl+Maj+D)

Stop the current debug session

7.1.5 Run/pause code execution

To run or stop a code execution, simply use the dedicated icon shortcut:

Figure 39. **IAR Embedded Workbench IDE run/stop code execution**

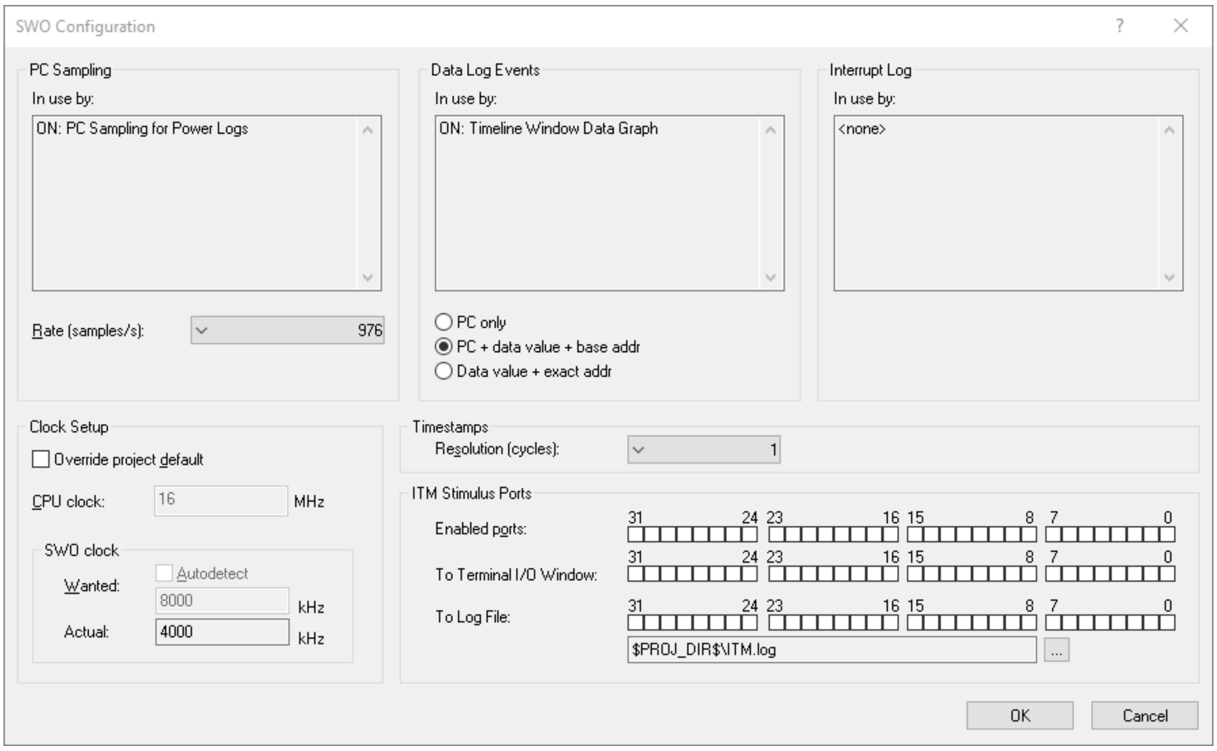

ETM
SWO
Go
Run the program in the debugger
Break
Stop execution and break into the debugger

7.1.6 Trace configuration (during debug session)

From the ST-LINK menu, open the SWO Configuration (only available when a debug session is running) to change the following settings:

- PC Sampling: program counter sampling rate: approximately 1000 SPS is a balanced value between accuracy and trace bandwidth usage.
- Data Log Events: select [**PC + data value + base addr**] to sample the PC value on data comparator read or write access.
- Timestamps/Resolution: select the trace timestamping resolution (1 should be selected).
- ITM Stimulus Ports: depending on the debug session, ITM can be checked.

Figure 40. **IAR Embedded Workbench IDE trace configuration**



SWO Configuration

PC Sampling
In use by:
ON: PC Sampling for Power Logs
Rate (samples/s): 976

Data Log Events
In use by:
ON: Timeline Window Data Graph
PC only
[✓ PC + data value + base addr]
Data value + exact addr

Interrupt Log
In use by:
<none>

Clock Setup
Override project default
CPU clock: 16 MHz
SWO clock
Autodetect
Wanted: 8000 kHz
Actual: 4000 kHz

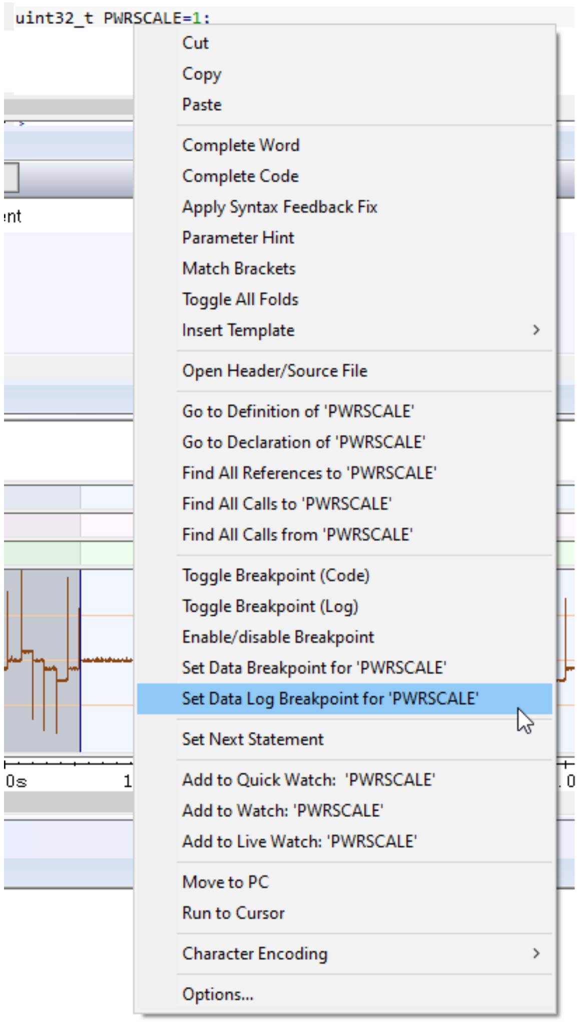
Timestamps
Resolution (cycles): 1

ITM Stimulus Ports
Enabled ports: 31 24 23 16 15 8 7 0
To Terminal I/O Window: 31 24 23 16 15 8 7 0
To Log File: 31 24 23 16 15 8 7 0
$PROJ_DIR$\ITM.log

7.1.7 Power waveforms synchronized with debug trace data

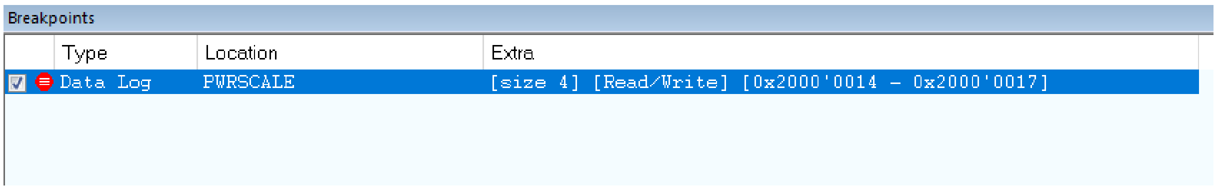
Only global variables (at fixed addresses) can be selected for synchronized observation with the current waveform in real time. In the code source, right-click on the global variable and select [**Set Data Log Breakpoint for [VARIABLE]**].

Figure 41. IAR Embedded Workbench IDE - add variable to trace


uint32_t PWRSCALE=1;
Cut
Copy
Paste
Complete Word
Complete Code
Apply Syntax Feedback Fix
Parameter Hint
Match Brackets
Toggle All Folds
Insert Template
Open Header/Source File
Go to Definition of 'PWRSCALE'
Go to Declaration of 'PWRSCALE'
Find All References to 'PWRSCALE'
Find All Calls to 'PWRSCALE'
Find All Calls from 'PWRSCALE'
Toggle Breakpoint (Code)
Toggle Breakpoint (Log)
Enable/disable Breakpoint
Set Data Breakpoint for 'PWRSCALE'
Set Data Log Breakpoint for 'PWRSCALE'
Set Next Statement
Add to Quick Watch: 'PWRSCALE'
Add to Watch: 'PWRSCALE'
Add to Live Watch: 'PWRSCALE'
Move to PC
Run to Cursor
Character Encoding
Options...

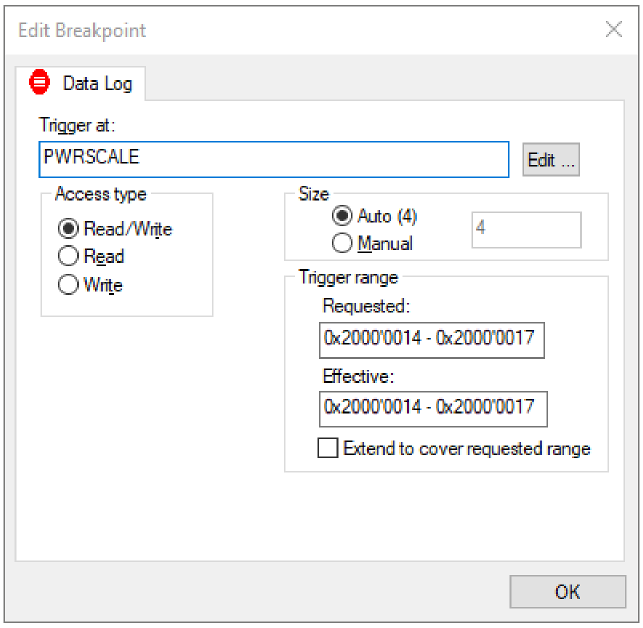
View the selected variable with the [**View/Breakpoints**] menu.

Figure 42. IAR Embedded Workbench IDE - view trace variable (data log breakpoints)


Breakpoints

	Type	Location	Extra
✓	Data Log	PWRSCALE	[size 4] [Read/Write] [0x2000'0014 - 0x2000'0017]

Figure 43. IAR Embedded Workbench IDE - edit trace variable (right-click option)



Edit Breakpoint

Data Log

Trigger at: PWRSCALE

Access type: [✓ Read/Write] Read Write

Size: [✓ Auto (4)] Manual 4

Trigger range
Requested: 0x2000'0014 - 0x2000'0017
Effective: 0x2000'0014 - 0x2000'0017
☐ Extend to cover requested range

Once the data log has been configured, the [**Data Log**] feature must be enabled in the Timeline window using the right-click option:

Figure 44. IAR Embedded Workbench IDE timeline data log

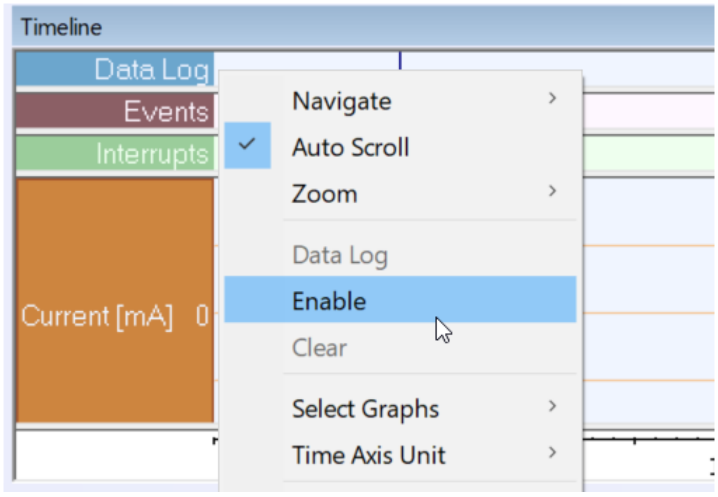
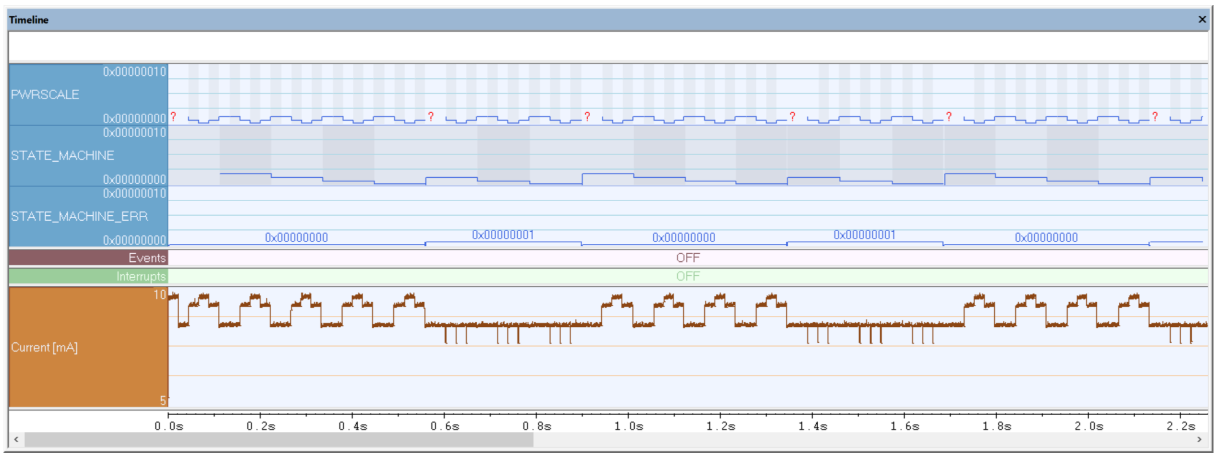


Figure 45. **IAR Embedded Workbench IDE timeline synchronized power and data log**

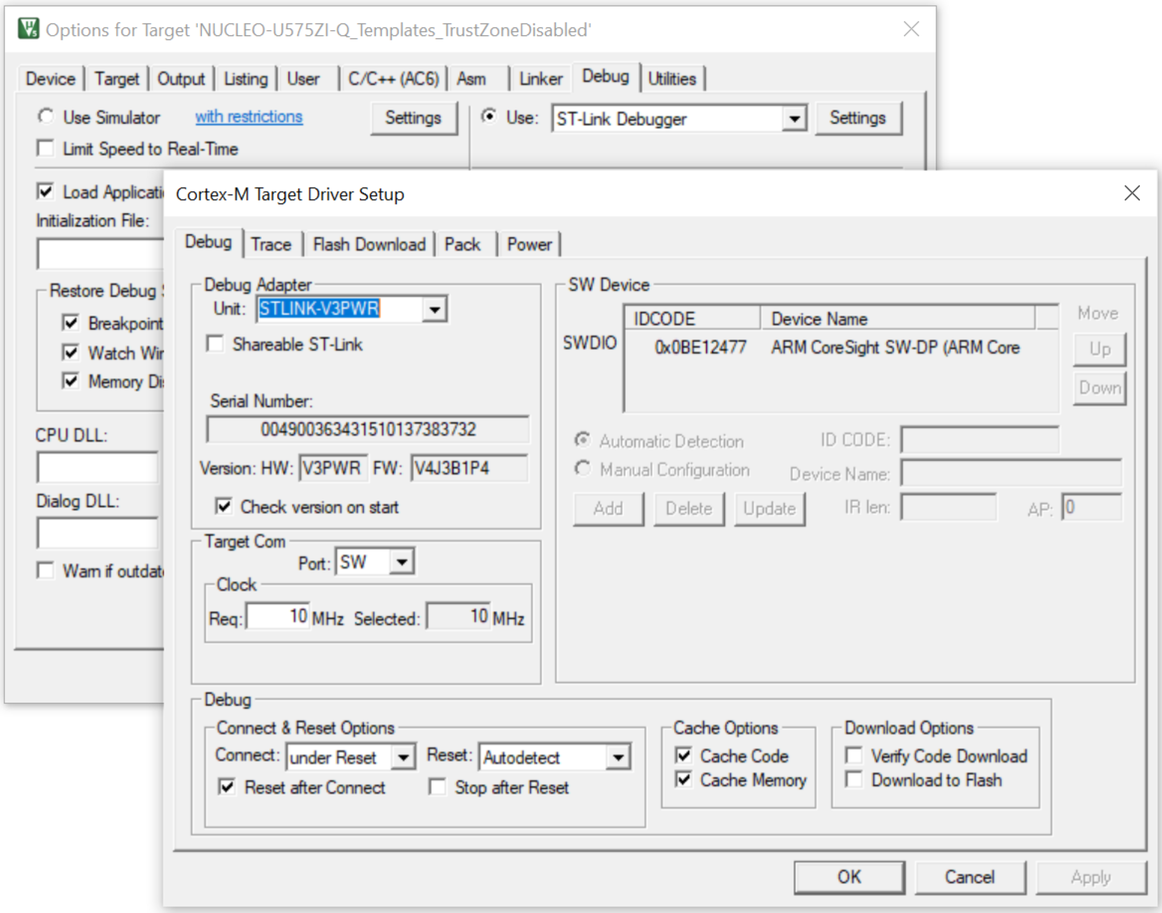


For more information, refer to the *Watch and graph variables over time with sampled graphs* and *Power debugging* articles on the IAR™ knowledge website.

7.2 Arm® µVision® (from version V5.39)

Arm® µVision® (from version V5.39) includes an additional ST-Link Debugger tab, dedicated to the power configuration of the probe when STLINK-V3PWR is present. It allows the configuration of the supply voltage, the sampling frequency acquisition, and the synchronization method. The configuration window is accessible through the Debug tab in the [**Project/Options**] menu.

Figure 46. Arm® µVision® STLINK-V3PWR settings window, from the "Project/Options" menu



Use Simulator with restrictions
Use: ST-Link Debugger
Limit Speed to Real-Time
Load Applicati
Initialization File:
Restore Debug
Breakpoint
Watch Wir
Memory Di
CPU DLL:
Dialog DLL:
Warn if outdat

Debug Adapter
Unit: STLINK-V3PWR
Shareable ST-Link
Serial Number: 004900363431510137383732
Version: HW: V3PWR FW: V4J3B1P4
Check version on start

Target Com
Port: SW
Clock
Req: 10 MHz Selected: 10 MHz

SW Device

	IDCODE	Device Name
SWDIO	0x0BE12477	ARM CoreSight SW-DP (ARM Core

Automatic Detection
Manual Configuration
ID CODE:
Device Name:
IR len:
AP: 0

Debug
Connect & Reset Options
Connect: under Reset
Reset: Autodetect
Reset after Connect
Stop after Reset

Cache Options
Cache Code
Cache Memory

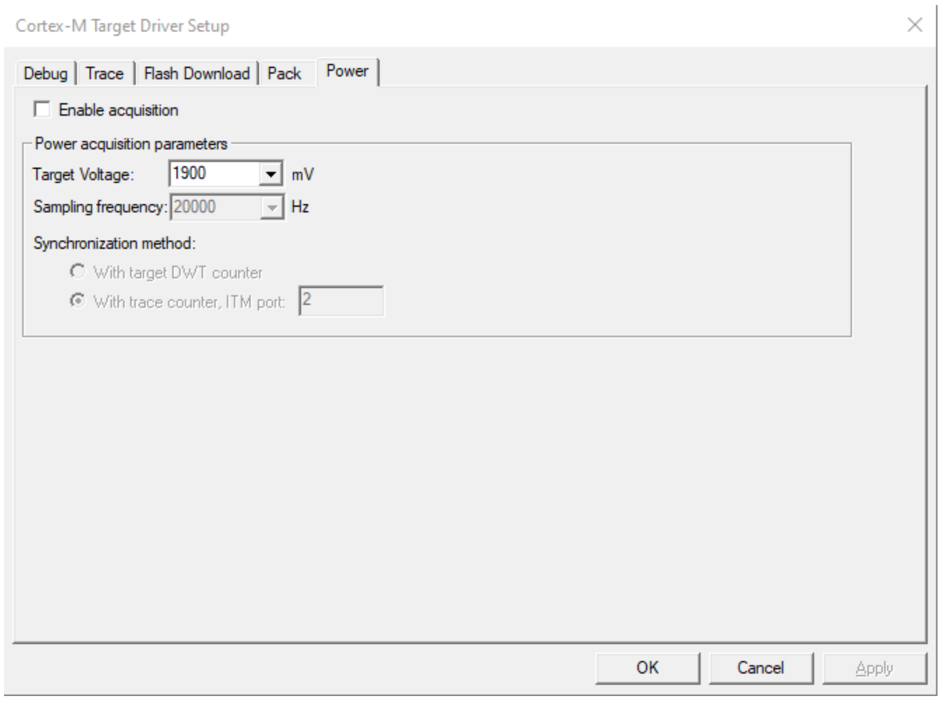
Download Options
Verify Code Download
Download to Flash

From the Debug probe settings window, you can access the Power tab to configure the power features of STLINK-V3PWR.

7.2.1 Power configuration

- [**Enable acquisition**]: enable or disable the current acquisition.
- [**Target voltage**] (mV): set the target application voltage of the VOUT and VAUX supplies.
- [**Sampling frequency**] (Hz): set the current waveform sampling frequency.
- [**Synchronization method**]: target DWT counter through the SWD interface (less accurate) or trace counter through SWO and selected ITM port.

Figure 47. Arm® µVision® STLINK-V3PWR power configuration window (acquisition disabled)



Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack | Power

☐ Enable acquisition

Power acquisition parameters

Target Voltage: 1900 mV

Sampling frequency: 20000 Hz

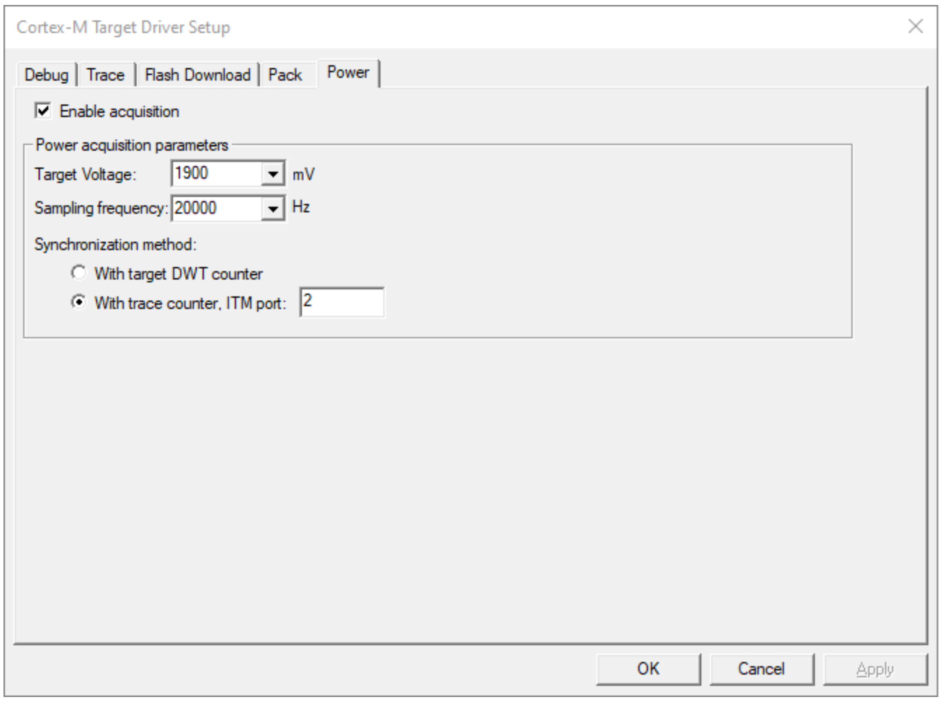
Synchronization method:

○ With target DWT counter

● With trace counter, ITM port: 2

OK | Cancel | Apply

Figure 48. Arm® µVision® STLINK-V3PWR power configuration window (acquisition enabled with trace counter)



Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack | Power

☑ Enable acquisition

Power acquisition parameters

Target Voltage: 1900 mV

Sampling frequency: 20000 Hz

Synchronization method:

○ With target DWT counter

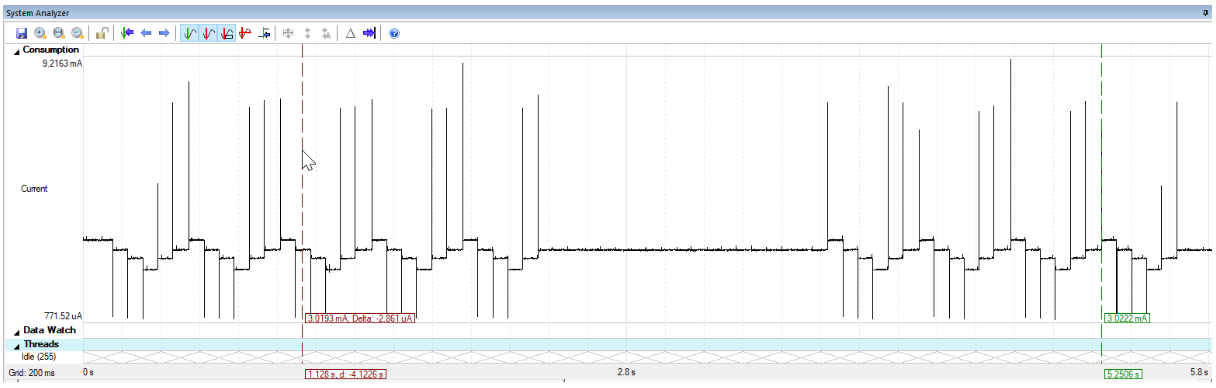
● With trace counter, ITM port: 2

OK | Cancel | Apply

7.2.2 Visualizing the power current consumption waveform

Open the dedicated window for current waveform observations via [**View**]>[**Analysis Window**]>[**System Analyzer**]. Ensure that the consumption figure is expanded to view the current waveform updated in real time. Refer to the *System Analyzer Window* section in the µVision® user guide for more information on using the system analyzer.

Figure 49. Arm® µVision® System Analyzer window



Important: *The current waveform displayed in the µVision® system analyzer is limited to a maximum number of points (approximately 1.2 million samples). Consequently, with a sampling rate of 100 kHz, the current waveform duration is limited to approximately 12 seconds.*

Figure 50. Arm® µVision® System Analyzer with 100-kHz acquisition (~12 s maximum duration)

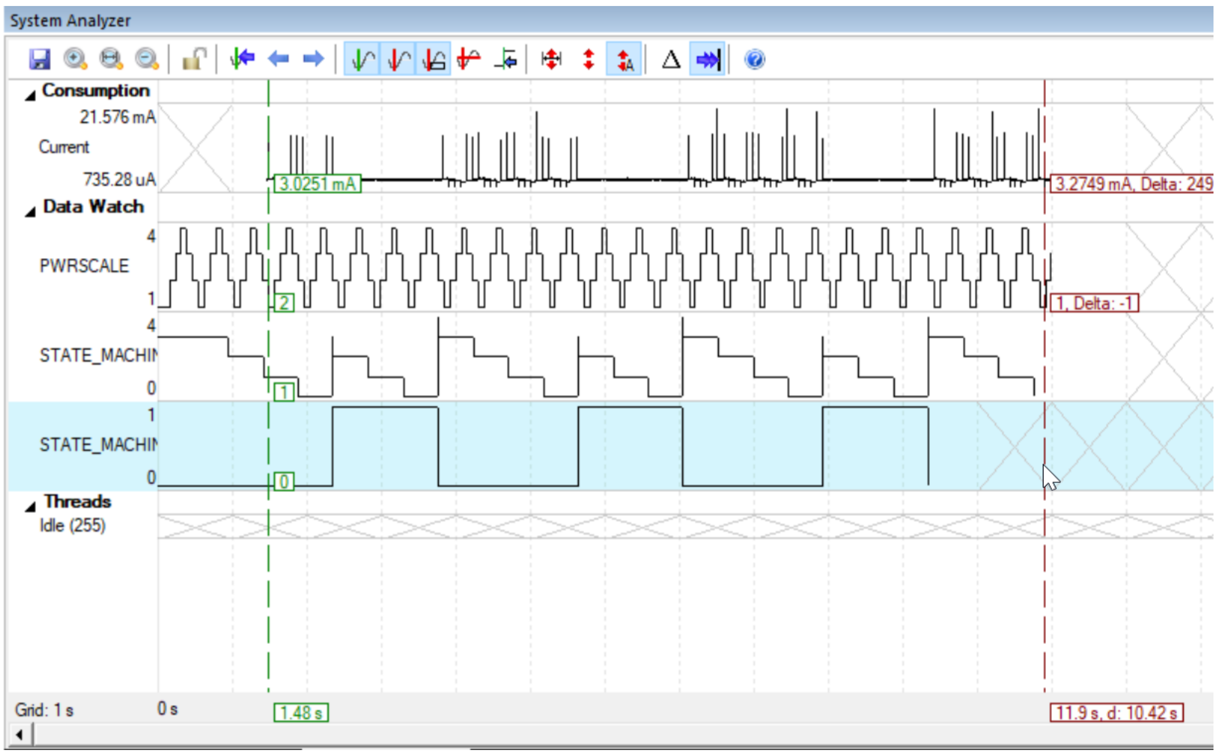
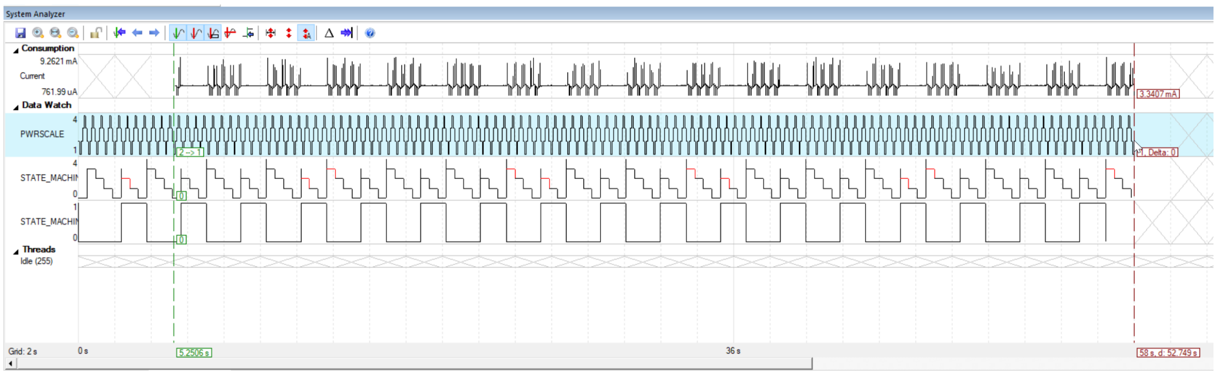


Figure 51. Arm® µVision® System Analyzer with 20 kHz acquisition (~60 s maximum duration)

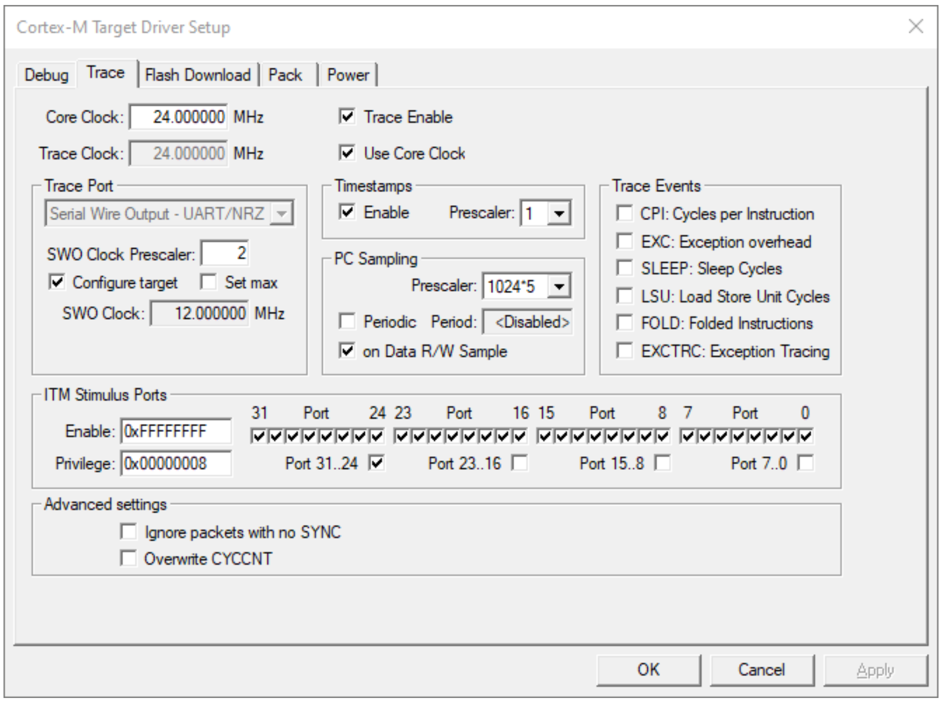


7.2.3 Trace configuration

The correct trace configuration is mandatory to ensure the correct decoding during the debug session. The following parameters contain the minimum requirements to perform a current waveform acquisition through ITM with the debug trace.

- [**Trace enable**]: enable or disable the debug trace (checkbox to be checked).
- [**Core Clock**]: exact target CPU frequency value (refer to the application firmware configuration).
- [**Use Core Clock**]: use (y/n) the core clock as trace clock (checkbox to be checked).
- [**Trace Port**]>[**Configure target**]: automatically configure the target (checkbox to be checked).
- [**Trace Port**]>[**SWO Clock Prescaler**]: value to configure the SWO clock. The recommended frequency for theSTLINK-V3PWR SWO clock is 8~12 MHz range (~10 MHz to be configured).
- [**Timestamps**]>[**Enable/Prescaler**]: enable trace timestamping and set the prescaler value (1 should be selected).
- [**PC Sampling**]>[**Prescaler**]: the prescaler value should be configured in line with the trace bandwidth usage. The trace log window shows if data are missing.
- [**PC Sampling**]>[**Periodic**]: periodic program counter sampling (it is recommended you disable this option to optimize trace bandwidth).
- [**PC Sampling**]>[**on Data RW Sample**]: sample the PC value on data comparator read or write access (checkbox to be checked).
- [**Trace events**]: depending on the debug session, events may be checked (no events checked).
- [**ITM Stimulus Ports**]: depending on the debug session, ITM may checked. At the least, ITM used for current waveform synchronization should be checked ([**ITM pwr sync**] checked).

Figure 52. Arm® µVision® STLINK-V3PWR trace setting window



Core Clock: 24.000000 MHz
Trace Clock: 24.000000 MHz
[✓] Trace Enable
[✓] Use Core Clock

Trace Port
Serial Wire Output - UART/NRZ
SWO Clock Prescaler: 2
[✓] Configure target
[ ] Set max
SWO Clock: 12.000000 MHz

Timestamps
[✓] Enable
Prescaler: 1

PC Sampling
Prescaler: 1024*5
[ ] Periodic
Period: <Disabled>
[✓] on Data R/W Sample

Trace Events
[ ] CPI: Cycles per Instruction
[ ] EXC: Exception overhead
[ ] SLEEP: Sleep Cycles
[ ] LSU: Load Store Unit Cycles
[ ] FOLD: Folded Instructions
[ ] EXCTRC: Exception Tracing

ITM Stimulus Ports
Enable: 0xFFFFFFFF
Privilege: 0x00000008
31 Port 24 23 Port 16 15 Port 8 7 Port 0
Port 31..24 [✓]
Port 23..16 [ ]
Port 15..8 [ ]
Port 7..0 [ ]

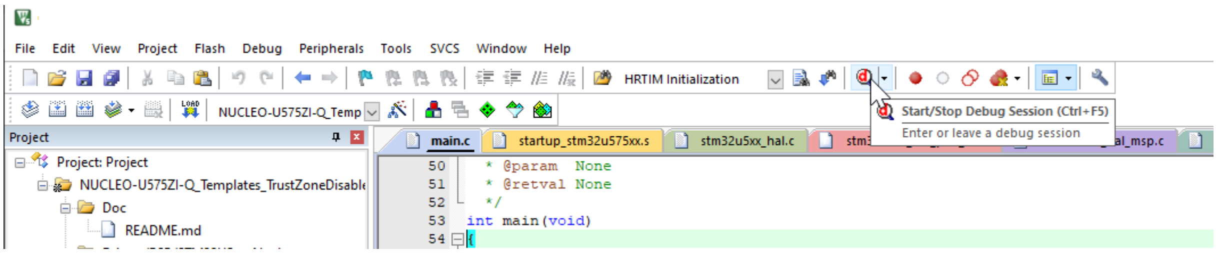
Advanced settings
[ ] Ignore packets with no SYNC
[ ] Overwrite CYCCNT

7.2.4 Start/stop debug session

To start or stop a debug session, simply use the dedicated icon shortcut, as illustrated in the figure below.

Warning: *When the debug session starts, the STLINK-V3PWR VOUT and VAUX supplies are automatically turned on. When the debug session ends, the STLINK-V3PWR VOUT and VAUX supplies are automatically turned off.*

Figure 53. Arm® µVision® - start/stop debug session



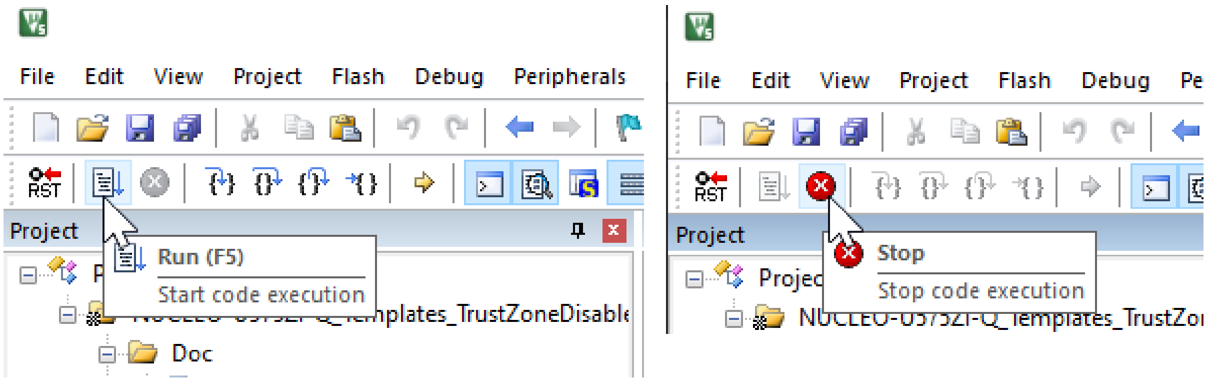
Start/Stop Debug Session (Ctrl+F5)
Enter or leave a debug session

```
  * @param  None
  * @retval None
  */
int main(void)
{
```

7.2.5 Run/pause code execution

To run or stop a code execution, simply use the dedicated icon shortcut:

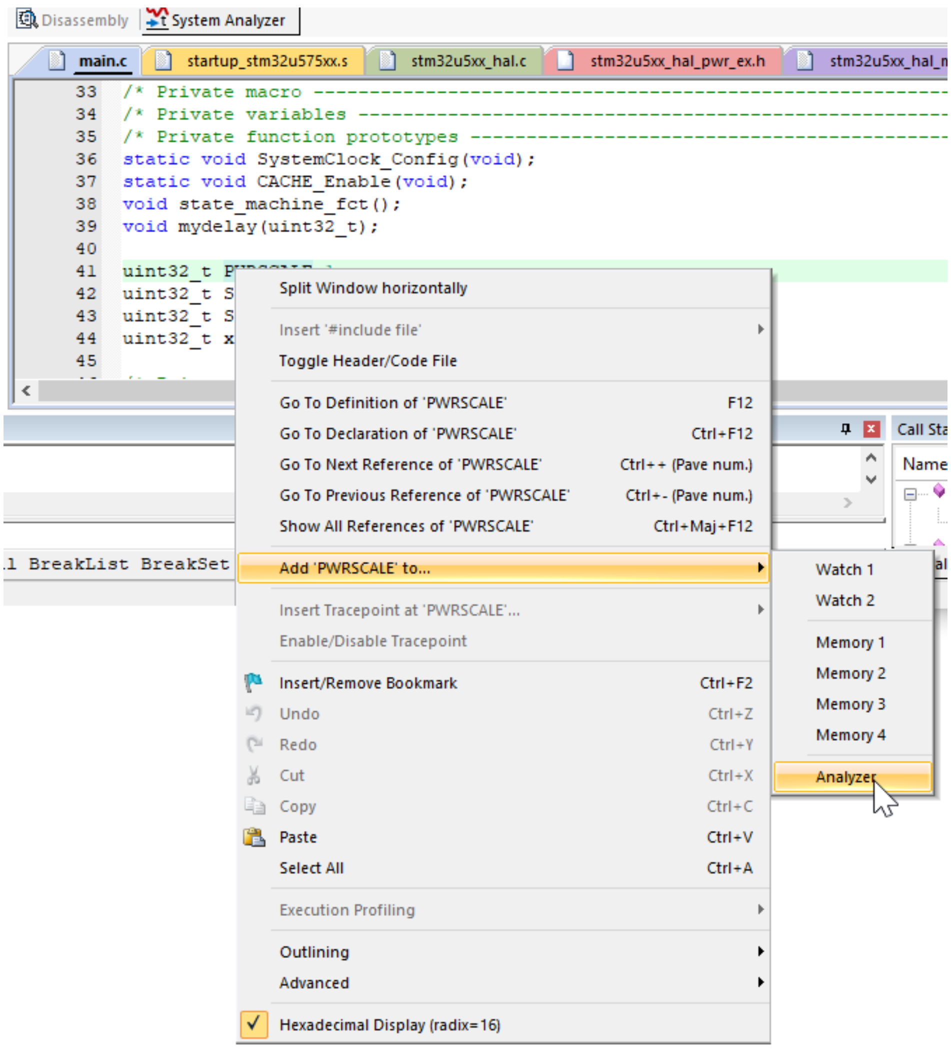
Figure 54. Arm® µVision® - run/stop code execution



7.2.6 Power waveforms synchronized with debug trace data

Only global variables (at fixed addresses) can be selected for synchronized observation with the current waveform. In the Code source window, right-click on the global variable and select [**Add [VARIABLE] to...**]>[**Analyzer**].

Figure 55. Arm® µVision® - add a variable to the analyzer


Disassembly
System Analyzer
main.c
startup_stm32u575xx.s
stm32u5xx_hal.c
stm32u5xx_hal_pwr_ex.h
stm32u5xx_hal_m

```
/* Private macro -------------------------------------------------------
/* Private variables ---------------------------------------------------
/* Private function prototypes -----------------------------------------
static void SystemClock_Config(void);
static void CACHE_Enable(void);
void state_machine_fct();
void mydelay(uint32_t);

uint32_t P
uint32_t S
uint32_t S
uint32_t x
```

Split Window horizontally
Insert '#include file'
Toggle Header/Code File
Go To Definition of 'PWRSCALE' F12
Go To Declaration of 'PWRSCALE' Ctrl+F12
Go To Next Reference of 'PWRSCALE' Ctrl++ (Pave num.)
Go To Previous Reference of 'PWRSCALE' Ctrl+- (Pave num.)
Show All References of 'PWRSCALE' Ctrl+Maj+F12
Add 'PWRSCALE' to...
Insert Tracepoint at 'PWRSCALE'...
Enable/Disable Tracepoint
Insert/Remove Bookmark Ctrl+F2
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A
Execution Profiling
Outlining
Advanced
Hexadecimal Display (radix=16)
Watch 1
Watch 2
Memory 1
Memory 2
Memory 3
Memory 4
Analyzer
Call Sta
Name
l BreakList BreakSet

Figure 56. Arm® µVision® - variable view in the Logic Analyzer

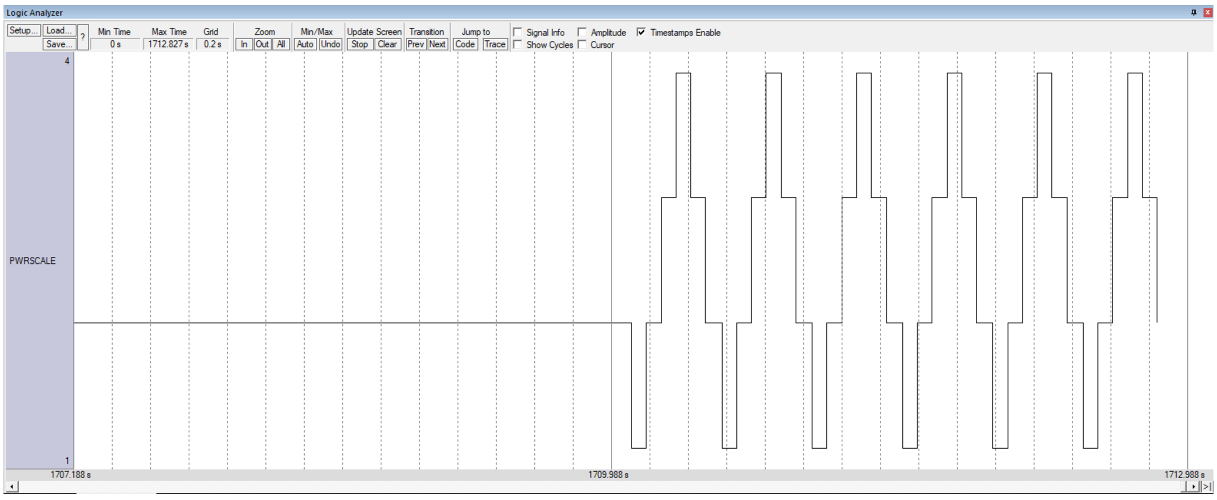
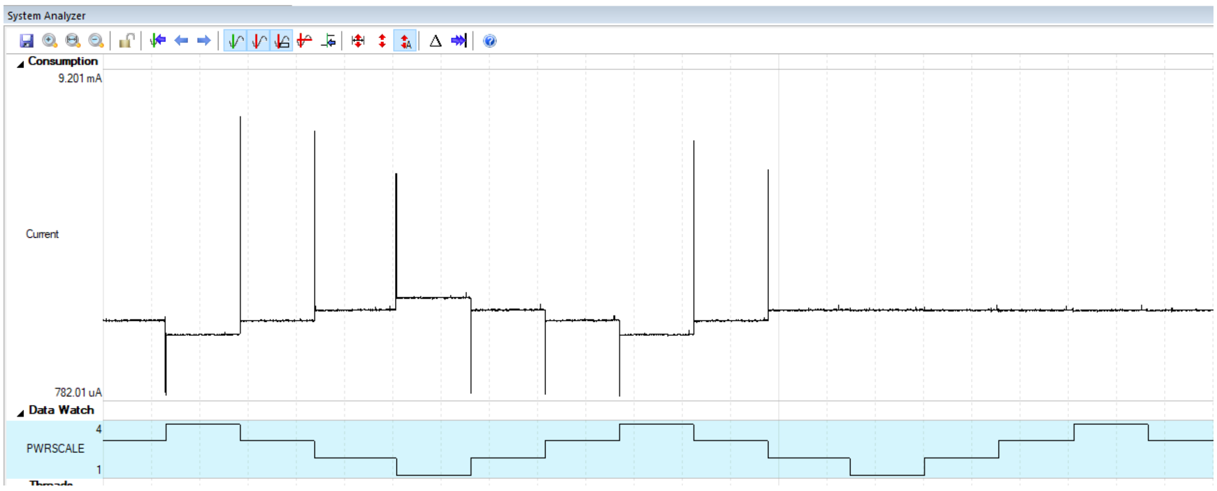


Figure 57. Arm® µVision® - variable view in the System Analyzer, synchronized with the current waveform



8 STLINK-V3PWR bridge control, debug, and power profiling

This chapter describes how to interleave STLINK-V3PWR bridge communication, debug, and current waveform acquisition with a software application developed in the STM32CubeIDE environment. This application example controls the bridge, debug, and power interfaces using the ST-LINK bridge API, STM32CubeProgrammer, and serial port communication.

8.1 Application example

This example emulates a low-power application with an STM32U5 B-U585I-IOT02A Discovery board. The board has several sensors: temperature, humidity, magnetometer, accelerometer, gyroscope, pressure, Time-of-Flight, and gesture-detection sensors, which is activated with the user button B3 or through the CN13 connector and the SPI1 interface. STLINK-V3PWR emulates the battery and acquires the current consumption waveform of the application board.

Figure 58. Low-power application example (connected watch)

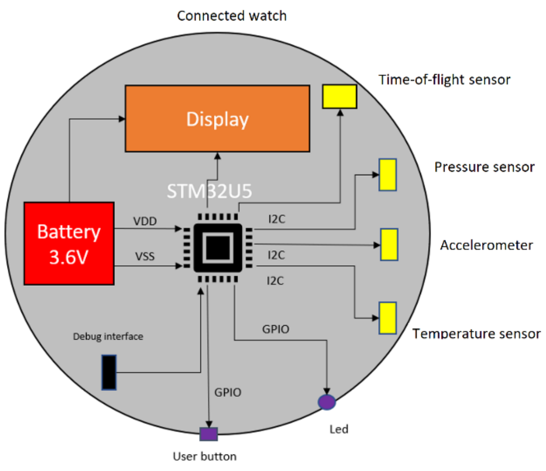
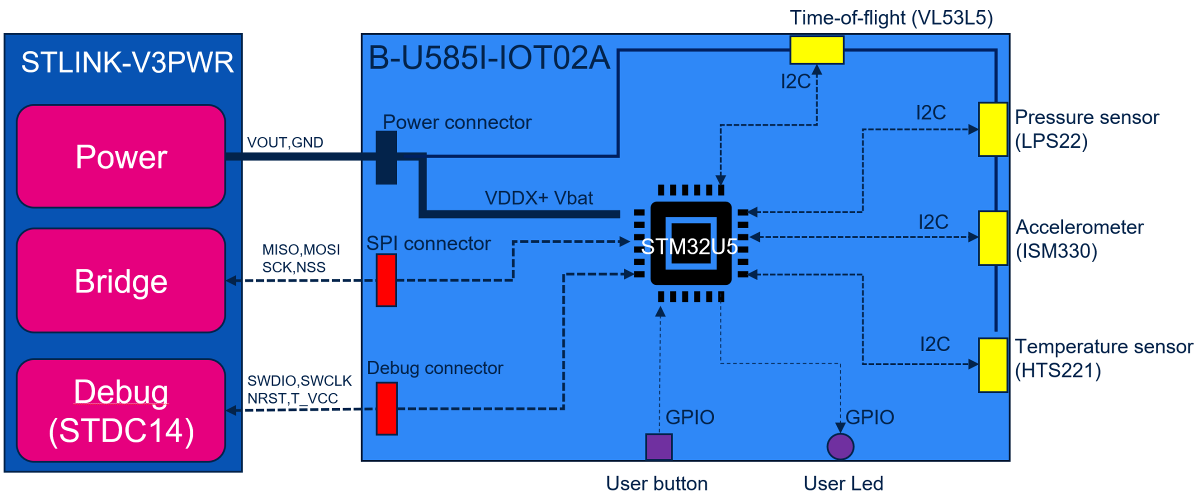


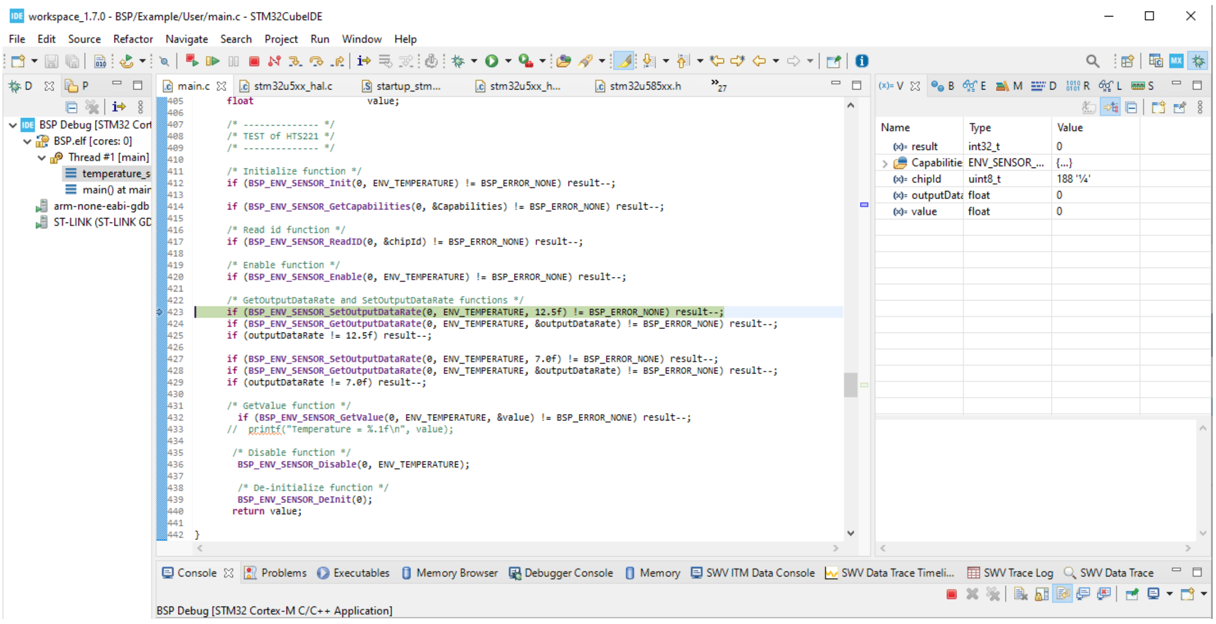
Figure 59. STLINK-V3PWR and B-U585I-IOT02A connections



8.2 Step-by-step debugging and current waveform acquisition

The firmware debug session has been started with STM32CubeIDE and the user can observe the current consumption waveform with STM32CubeMonitor-Power, while executing a step-by-step breakpoint trough the temperature sensor board support package (BSP) functions. Every five seconds, the user executes the firmware code to the next breakpoint to monitor the sensor current consumption (external sensor component). Then, they annotate the full current waveform with the function call at the given time reference.

Figure 60. Step-by-step debugging of the external component temperature sensor



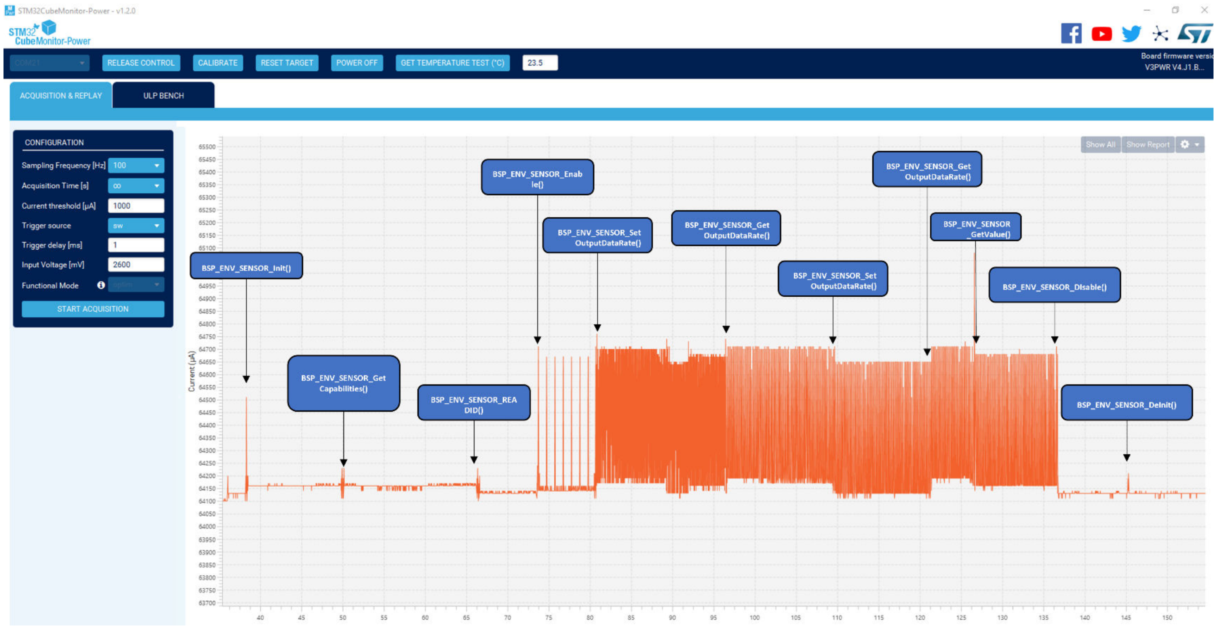
workspace_1.7.0 - BSP/Example/User/main.c - STM32CubeIDE

```c
    float                   value;

    /* -------------- */
    /* TEST of HTS221 */
    /* -------------- */

    /* Initialize function */
    if (BSP_ENV_SENSOR_Init(0, ENV_TEMPERATURE) != BSP_ERROR_NONE) result--;

    if (BSP_ENV_SENSOR_GetCapabilities(0, &Capabilities) != BSP_ERROR_NONE) result--;

    /* Read id function */
    if (BSP_ENV_SENSOR_ReadID(0, &chipId) != BSP_ERROR_NONE) result--;

    /* Enable function */
    if (BSP_ENV_SENSOR_Enable(0, ENV_TEMPERATURE) != BSP_ERROR_NONE) result--;

    /* GetOutputDataRate and SetOutputDataRate functions */
    if (BSP_ENV_SENSOR_SetOutputDataRate(0, ENV_TEMPERATURE, 12.5f) != BSP_ERROR_NONE) result--;
    if (BSP_ENV_SENSOR_GetOutputDataRate(0, ENV_TEMPERATURE, &outputDataRate) != BSP_ERROR_NONE) result--;
    if (outputDataRate != 12.5f) result--;

    if (BSP_ENV_SENSOR_SetOutputDataRate(0, ENV_TEMPERATURE, 7.0f) != BSP_ERROR_NONE) result--;
    if (BSP_ENV_SENSOR_GetOutputDataRate(0, ENV_TEMPERATURE, &outputDataRate) != BSP_ERROR_NONE) result--;
    if (outputDataRate != 7.0f) result--;

    /* GetValue function */
      if (BSP_ENV_SENSOR_GetValue(0, ENV_TEMPERATURE, &value) != BSP_ERROR_NONE) result--;
    //  printf("Temperature = %.1f\n", value);

     /* Disable function */
      BSP_ENV_SENSOR_Disable(0, ENV_TEMPERATURE);

      /* De-initialize function */
      BSP_ENV_SENSOR_DeInit(0);
     return value;

}
```

Name	Type	Value
result	int32_t	0
Capabilitie	ENV_SENSOR_...	{...}
chipId	uint8_t	188 '¼'
outputDat	float	0
value	float	0

BSP Debug [STM32 Cortex-M C/C++ Application]

Figure 61. Annotate acquisition waveform with BSP function calls



The user can observe the current consumption waveform (pulse rate) when the sensor is enabled and its configuration is changed (4th and 5th steps: `BSP_ENV_SENSOR_Enable()` and `BSP_ENV_SENSOR_Set_OutputDataRate()`). The average current value is increased after the output rate is configured; this configuration may be changed to reduce power consumption.

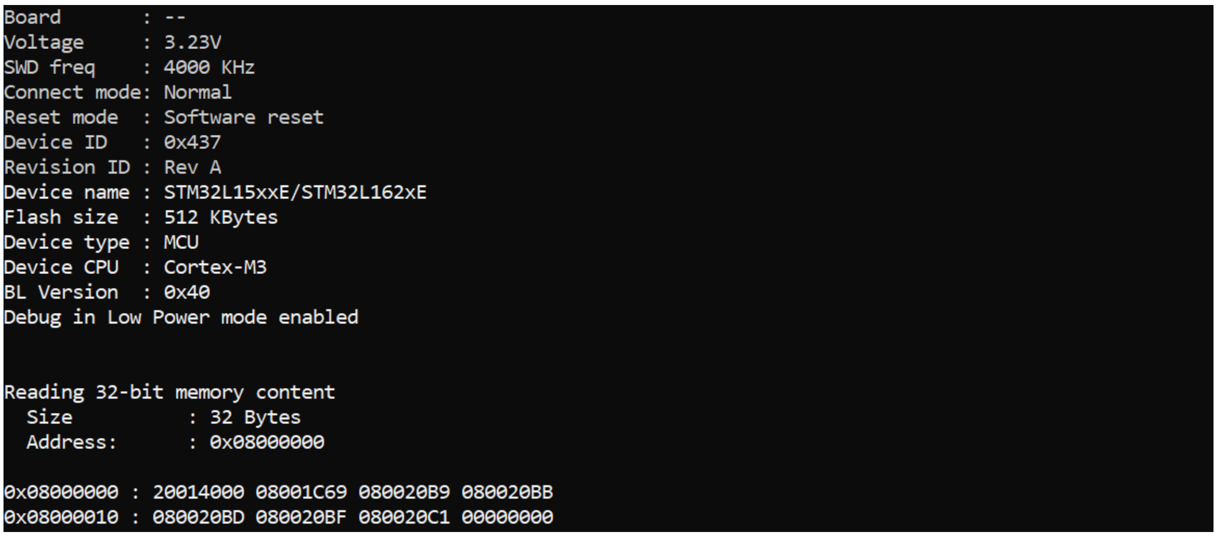
8.3 Program, control, and current waveform acquisition

This section demonstrates how to use STM32CubeProgrammer, the STLINK-V3-BRIDGE API, and serial COM communication to acquire a current waveform.

First, build an application on the host computer to program, verify, control, and acquire the average current consumption.

The application programs and reads the microcontroller memory content with STM32CubeProgrammer.

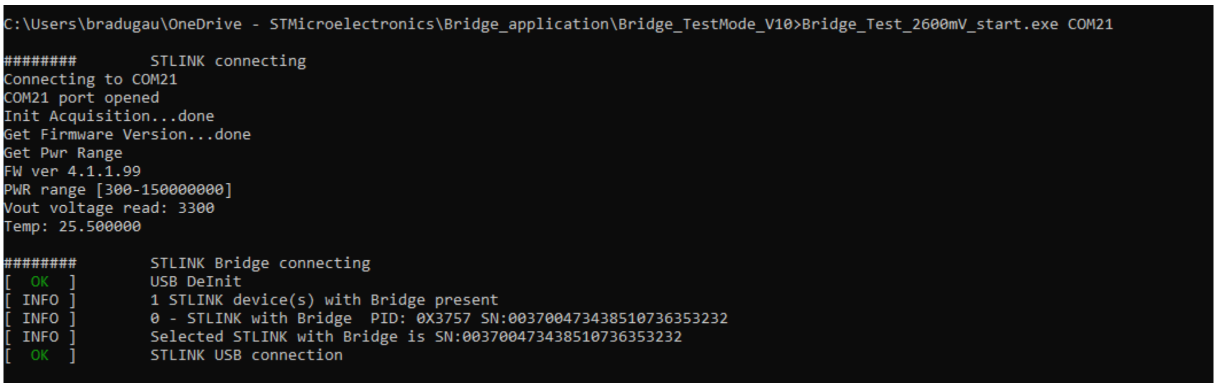
Figure 62. **Memory reading through the debug interface with STM32_Programmer_CLI**



```
Board       : --
Voltage     : 3.23V
SWD freq    : 4000 KHz
Connect mode: Normal
Reset mode  : Software reset
Device ID   : 0x437
Revision ID : Rev A
Device name : STM32L15xxE/STM32L162xE
Flash size  : 512 KBytes
Device type : MCU
Device CPU  : Cortex-M3
BL Version  : 0x40
Debug in Low Power mode enabled


Reading 32-bit memory content
  Size          : 32 Bytes
  Address:      : 0x08000000

0x08000000 : 20014000 08001C69 080020B9 080020BB
0x08000010 : 080020BD 080020BF 080020C1 00000000
```

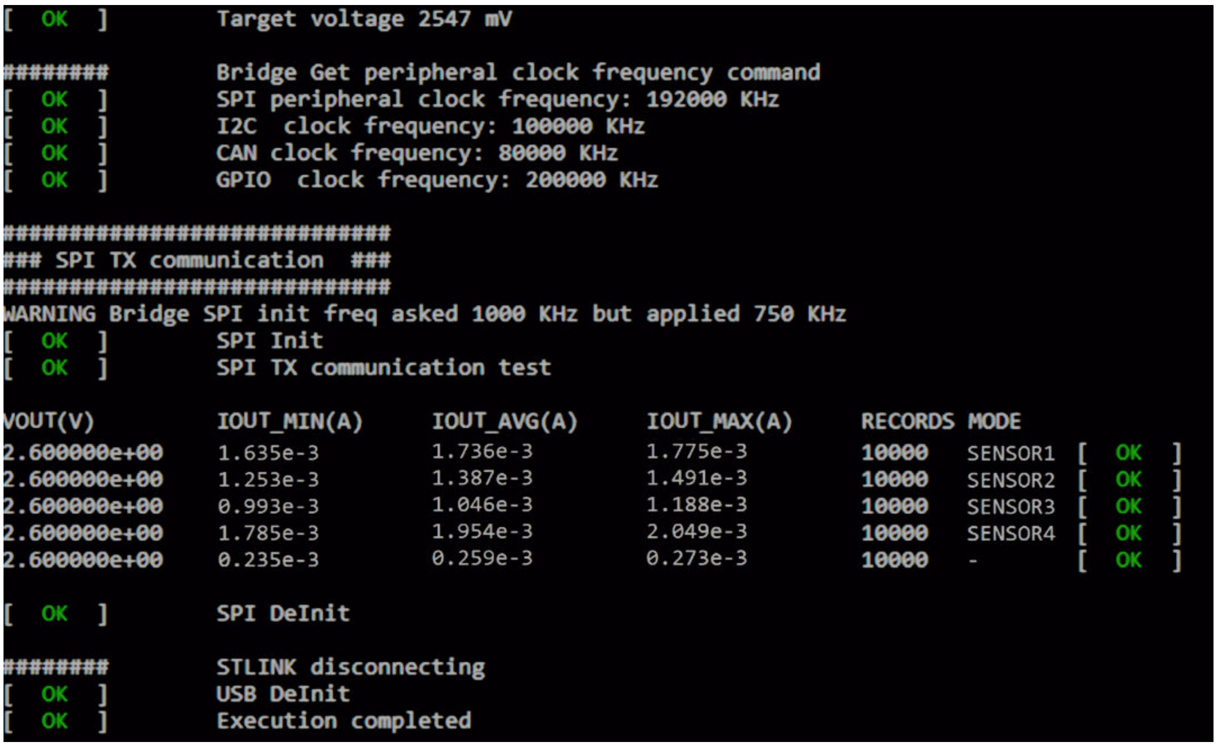
Then, control the target application with SPI communication through the bridge interface, and measure the average current when each sensor is enabled. The following figures show the application interface.

Figure 63. **STLINK-V3PWR bridge connection and initialization**



```
C:\Users\bradugau\OneDrive - STMicroelectronics\Bridge_application\Bridge_TestMode_V10>Bridge_Test_2600mV_start.exe COM21

########        STLINK connecting
Connecting to COM21
COM21 port opened
Init Acquisition...done
Get Firmware Version...done
Get Pwr Range
FW ver 4.1.1.99
PWR range [300-150000000]
Vout voltage read: 3300
Temp: 25.500000

########        STLINK Bridge connecting
[  OK  ]        USB DeInit
[ INFO ]        1 STLINK device(s) with Bridge present
[ INFO ]        0 - STLINK with Bridge  PID: 0X3757 SN:003700473438510736353232
[ INFO ]        Selected STLINK with Bridge is SN:003700473438510736353232
[  OK  ]        STLINK USB connection
```

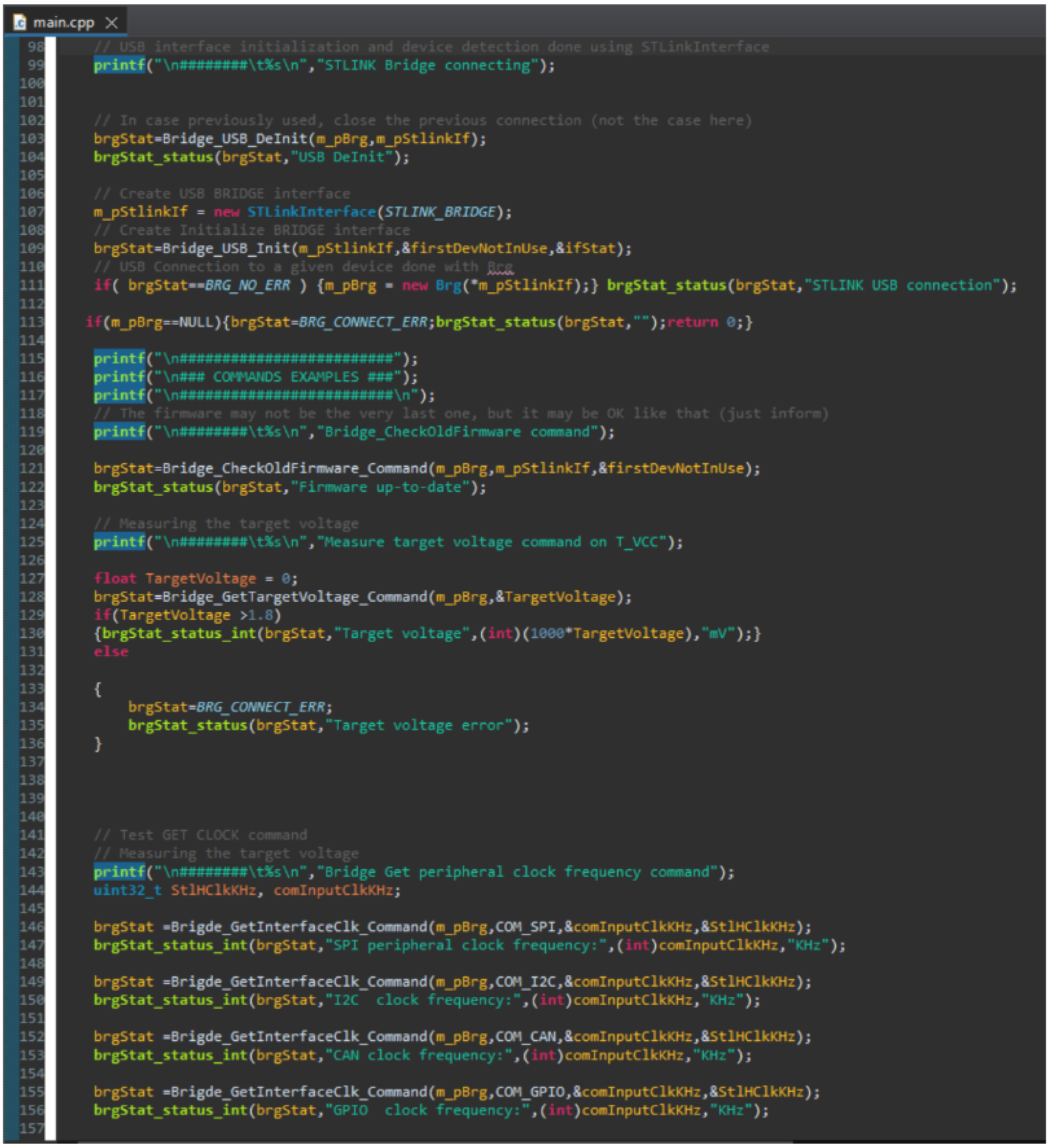
Figure 64. STLINK-V3PWR bridge SPI communication and power consumption measurements



```
[   OK   ]         Target voltage 2547 mV

########           Bridge Get peripheral clock frequency command
[   OK   ]         SPI peripheral clock frequency: 192000 KHz
[   OK   ]         I2C  clock frequency: 100000 KHz
[   OK   ]         CAN clock frequency: 80000 KHz
[   OK   ]         GPIO  clock frequency: 200000 KHz

#############################
### SPI TX communication  ###
#############################
WARNING Bridge SPI init freq asked 1000 KHz but applied 750 KHz
[   OK   ]         SPI Init
[   OK   ]         SPI TX communication test

VOUT(V)            IOUT_MIN(A)      IOUT_AVG(A)      IOUT_MAX(A)      RECORDS  MODE
2.600000e+00       1.635e-3         1.736e-3         1.775e-3         10000    SENSOR1  [   OK   ]
2.600000e+00       1.253e-3         1.387e-3         1.491e-3         10000    SENSOR2  [   OK   ]
2.600000e+00       0.993e-3         1.046e-3         1.188e-3         10000    SENSOR3  [   OK   ]
2.600000e+00       1.785e-3         1.954e-3         2.049e-3         10000    SENSOR4  [   OK   ]
2.600000e+00       0.235e-3         0.259e-3         0.273e-3         10000    -        [   OK   ]

[   OK   ]         SPI DeInit

########           STLINK disconnecting
[   OK   ]         USB DeInit
[   OK   ]         Execution completed
```

The following figures show the application source code in the STM32CubeIDE environment.

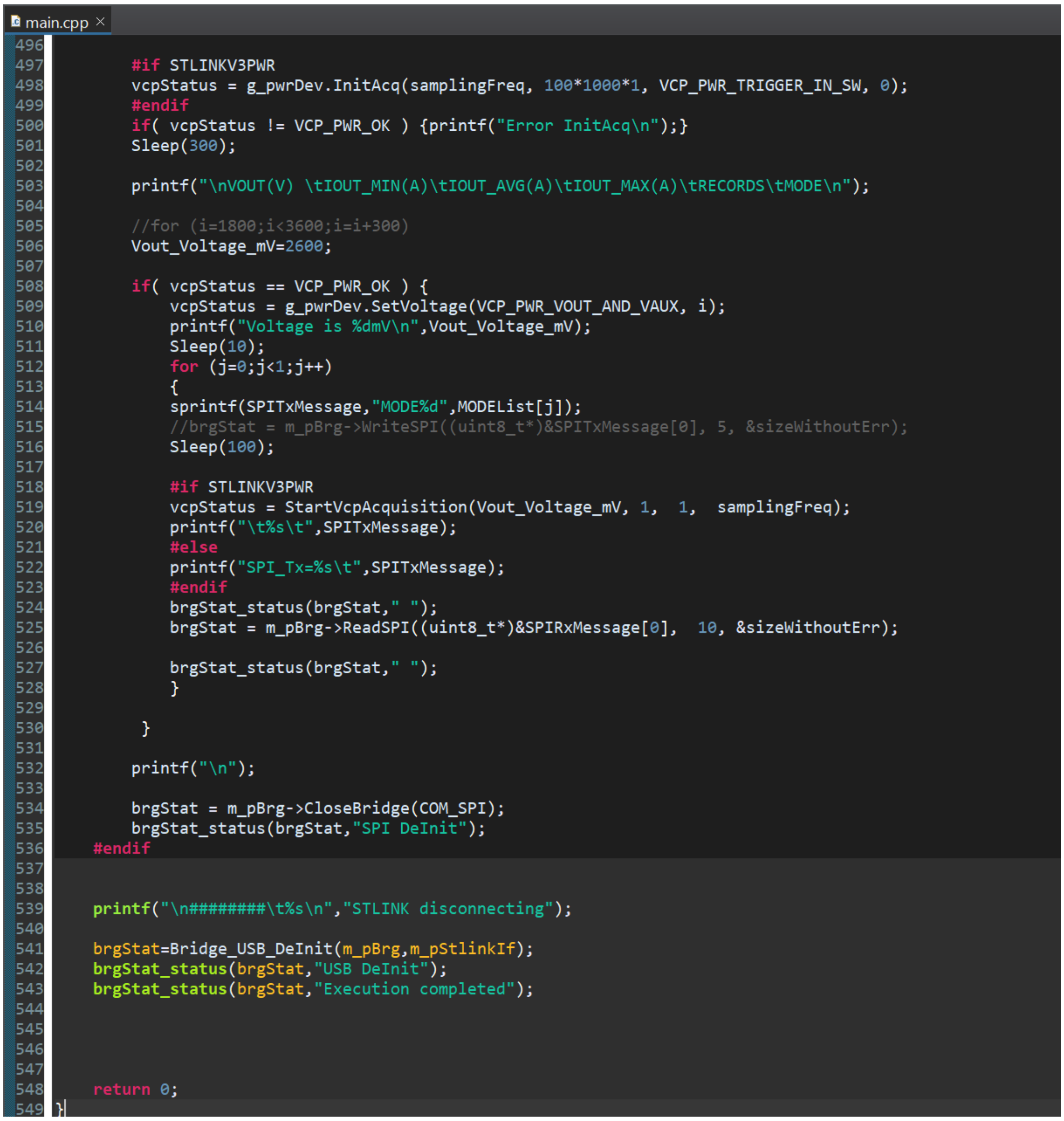
Figure 65. STLINK-V3 bridge API functions (initialization and status)



main.cpp

```cpp
    // USB interface initialization and device detection done using STLinkInterface
    printf("\n########\t%s\n","STLINK Bridge connecting");


    // In case previously used, close the previous connection (not the case here)
    brgStat=Bridge_USB_DeInit(m_pBrg,m_pStlinkIf);
    brgStat_status(brgStat,"USB DeInit");

    // Create USB BRIDGE interface
    m_pStlinkIf = new STLinkInterface(STLINK_BRIDGE);
    // Create Initialize BRIDGE interface
    brgStat=Bridge_USB_Init(m_pStlinkIf,&firstDevNotInUse,&ifStat);
    // USB Connection to a given device done with Brg
    if( brgStat==BRG_NO_ERR ) {m_pBrg = new Brg(*m_pStlinkIf);} brgStat_status(brgStat,"STLINK USB connection");

   if(m_pBrg==NULL){brgStat=BRG_CONNECT_ERR;brgStat_status(brgStat,"");return 0;}

    printf("\n#########################");
    printf("\n### COMMANDS EXAMPLES ###");
    printf("\n#########################\n");
    // The firmware may not be the very last one, but it may be OK like that (just inform)
    printf("\n########\t%s\n","Bridge_CheckOldFirmware command");

    brgStat=Bridge_CheckOldFirmware_Command(m_pBrg,m_pStlinkIf,&firstDevNotInUse);
    brgStat_status(brgStat,"Firmware up-to-date");

    // Measuring the target voltage
    printf("\n########\t%s\n","Measure target voltage command on T_VCC");

    float TargetVoltage = 0;
    brgStat=Bridge_GetTargetVoltage_Command(m_pBrg,&TargetVoltage);
    if(TargetVoltage >1.8)
    {brgStat_status_int(brgStat,"Target voltage",(int)(1000*TargetVoltage),"mV");}
    else

    {
        brgStat=BRG_CONNECT_ERR;
        brgStat_status(brgStat,"Target voltage error");
    }




    // Test GET CLOCK command
    // Measuring the target voltage
    printf("\n########\t%s\n","Bridge Get peripheral clock frequency command");
    uint32_t StlHClkKHz, comInputClkKHz;

    brgStat =Brigde_GetInterfaceClk_Command(m_pBrg,COM_SPI,&comInputClkKHz,&StlHClkKHz);
    brgStat_status_int(brgStat,"SPI peripheral clock frequency:",(int)comInputClkKHz,"KHz");

    brgStat =Brigde_GetInterfaceClk_Command(m_pBrg,COM_I2C,&comInputClkKHz,&StlHClkKHz);
    brgStat_status_int(brgStat,"I2C  clock frequency:",(int)comInputClkKHz,"KHz");

    brgStat =Brigde_GetInterfaceClk_Command(m_pBrg,COM_CAN,&comInputClkKHz,&StlHClkKHz);
    brgStat_status_int(brgStat,"CAN clock frequency:",(int)comInputClkKHz,"KHz");

    brgStat =Brigde_GetInterfaceClk_Command(m_pBrg,COM_GPIO,&comInputClkKHz,&StlHClkKHz);
    brgStat_status_int(brgStat,"GPIO  clock frequency:",(int)comInputClkKHz,"KHz");
```

Figure 66. **STLINK-V3 bridge API and power consumption functions via serial COM port communication**



main.cpp

```

        #if STLINKV3PWR
        vcpStatus = g_pwrDev.InitAcq(samplingFreq, 100*1000*1, VCP_PWR_TRIGGER_IN_SW, 0);
        #endif
        if( vcpStatus != VCP_PWR_OK ) {printf("Error InitAcq\n");}
        Sleep(300);

        printf("\nVOUT(V) \tIOUT_MIN(A)\tIOUT_AVG(A)\tIOUT_MAX(A)\tRECORDS\tMODE\n");

        //for (i=1800;i<3600;i=i+300)
        Vout_Voltage_mV=2600;

        if( vcpStatus == VCP_PWR_OK ) {
            vcpStatus = g_pwrDev.SetVoltage(VCP_PWR_VOUT_AND_VAUX, i);
            printf("Voltage is %dmV\n",Vout_Voltage_mV);
            Sleep(10);
            for (j=0;j<1;j++)
            {
            sprintf(SPITxMessage,"MODE%d",MODEList[j]);
            //brgStat = m_pBrg->WriteSPI((uint8_t*)&SPITxMessage[0], 5, &sizeWithoutErr);
            Sleep(100);

            #if STLINKV3PWR
            vcpStatus = StartVcpAcquisition(Vout_Voltage_mV, 1,  1,  samplingFreq);
            printf("\t%s\t",SPITxMessage);
            #else
            printf("SPI_Tx=%s\t",SPITxMessage);
            #endif
            brgStat_status(brgStat," ");
            brgStat = m_pBrg->ReadSPI((uint8_t*)&SPIRxMessage[0],  10, &sizeWithoutErr);

            brgStat_status(brgStat," ");
            }

         }

        printf("\n");

        brgStat = m_pBrg->CloseBridge(COM_SPI);
        brgStat_status(brgStat,"SPI DeInit");
    #endif


    printf("\n########\t%s\n","STLINK disconnecting");

    brgStat=Bridge_USB_DeInit(m_pBrg,m_pStlinkIf);
    brgStat_status(brgStat,"USB DeInit");
    brgStat_status(brgStat,"Execution completed");




    return 0;
}
```

This chapter has demonstrated how to use the STLINK-V3PWR features to program or access the STM32 memory, control a complex target application with the bridge interface, and perform current consumption measurements with a unique debug probe. This approach can be very useful when verifying the correct application behavior and performances.

9 Conclusion

The STLINK-V3PWR debug probe has many useful features:

- It supplies and acquires the current consumption of a target application with trigger functionalities with the bridge interface.
- It can control any electronic application.
- With the debug interface, it can program, observe the firmware, and synchronize a data trace with the current waveform.

Many tools can interface with the probe:

- For power configuration and monitoring, STM32CubeMonitor-Power or any other software or device with serial communication can control the power functionalities.
- For the debug interface, STM32CubeProgrammer, STM32CubeIDE, and third-partyIDEs are compatible and can control the debug and power interfaces.
- The bridge interface can be used through its software API to generate various protocols and GPIOs

Revision history

Table 5. Document revision history

Date	Version	Changes
24-Aug-2023	1	Initial release.
30-Aug-2023	2	Document title updated (reference to MPUs removed).

Contents

List of tables

List of figures