life.augmented

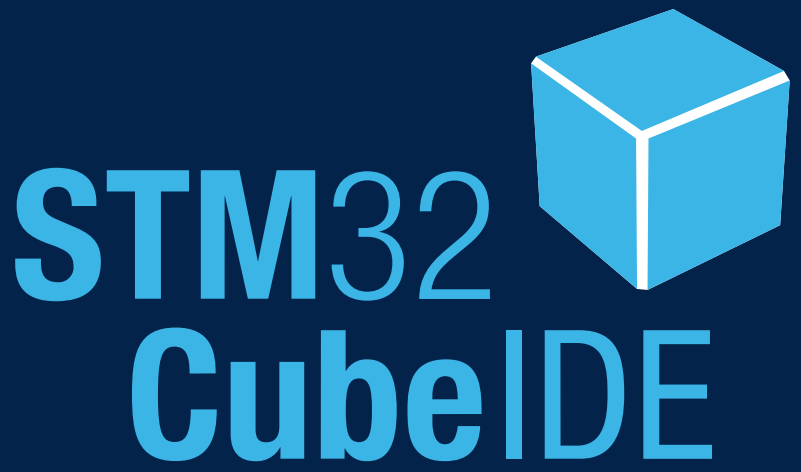
Innovations in debugging tools

Krishan Solanki

GPM Staff Engineer

3rd Aug 2023

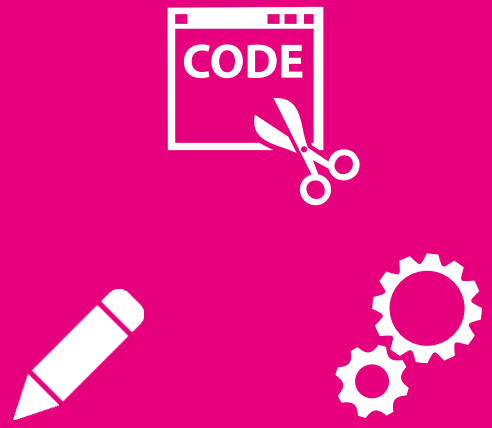
STM32CubeIDE



One tool for all your STM32 development

Chipset / Board Configuration

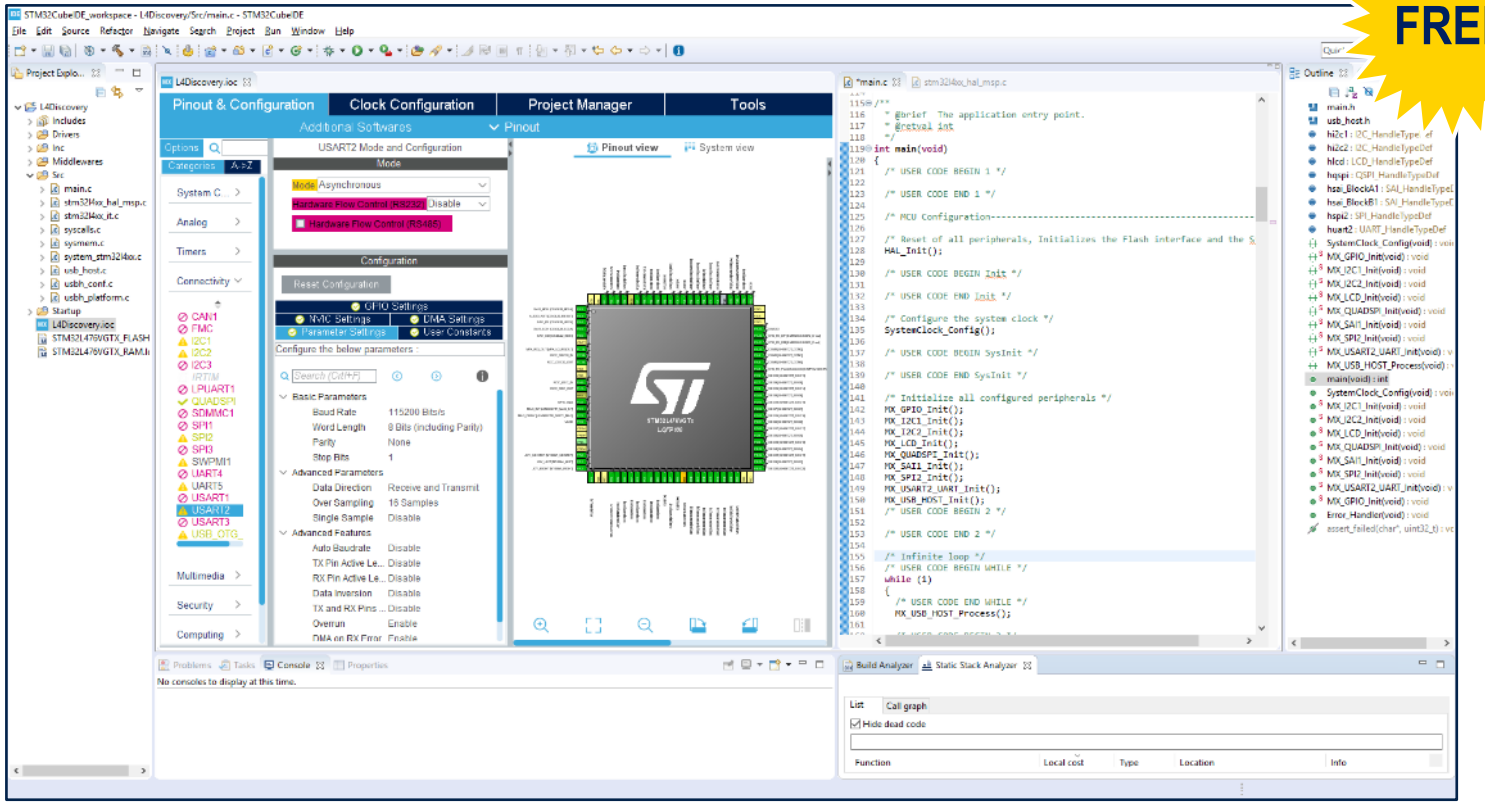
Code Development



Validation Debug

life.augmented

Free multi-platform development tool



eclipse

GCC

Windows

macOS

Eclipse/GCC Based

Free for Commercial Development

Multi-OS Support

life.augmented

TO-DO

Create a STM32CubeIDE project for GPIO toggle with Project Name : FAE_name

OR

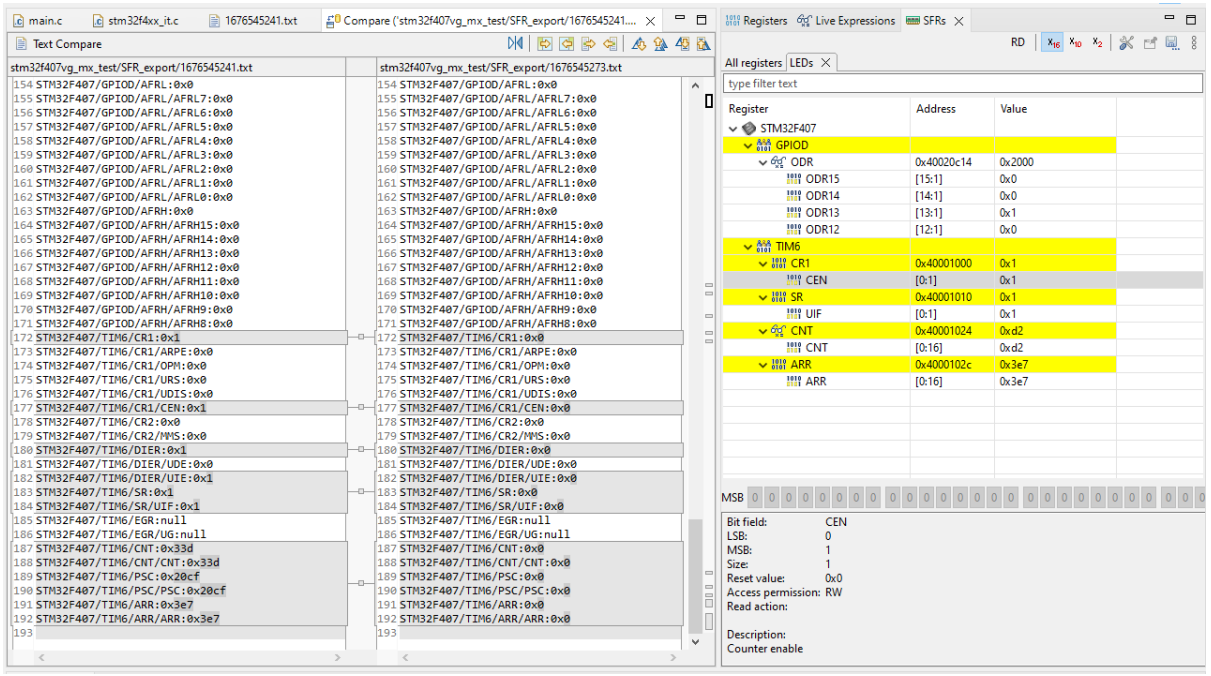
Open below project

Repository\STM32Cube_FW_U5_V1.2.0\Projects\NUCLEO-U575ZI-Q\Examples\GPIO\GPIO_EXTI

- Add the GPIOC->ODR and IDR register in the favorite list
- Give the list name as “CompanyName_ FaeName”

New

SFR-view – 3 use cases



- Favorite lists
  - Focus on a subset of peripherals and registers!
- Live update
  - Live channel provides ~5Hz live update
- Log export
  - Troubleshoot your peripheral configuration by comparing log-files

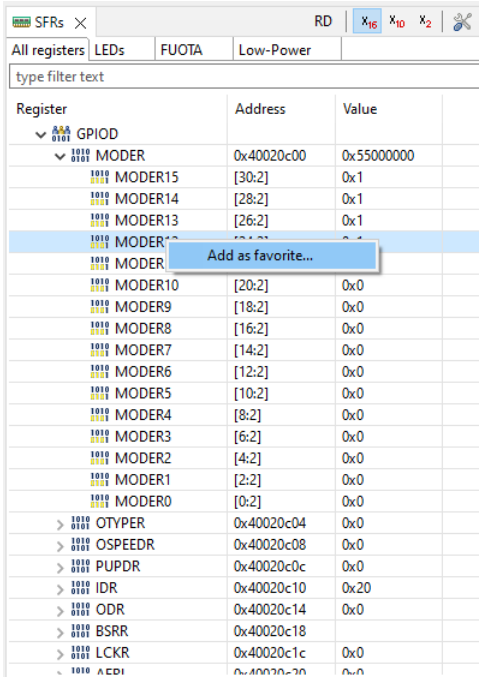
Available in STM32CubeIDE 1.11.x or later

life.augmented

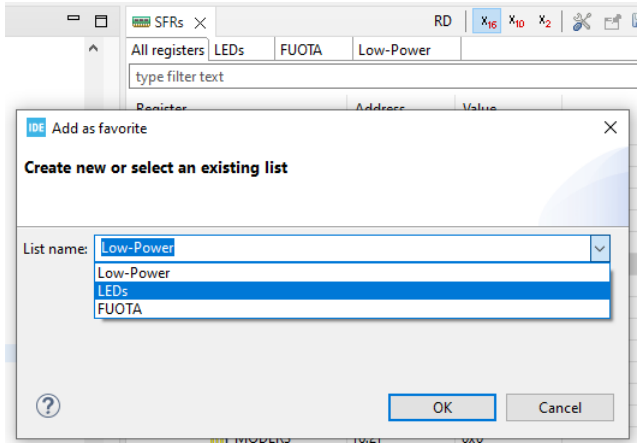
New

Favorite list(s)

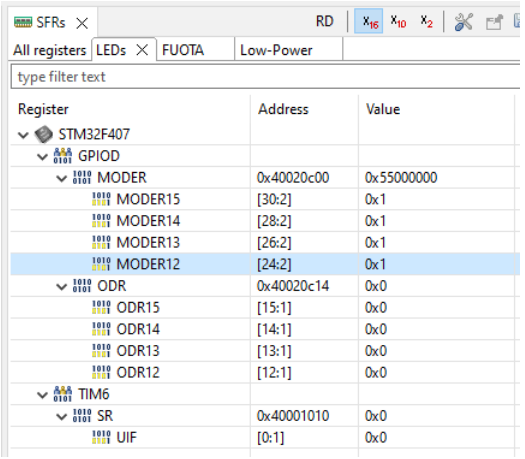
1. Right-click on a node and click *Add as favorite…*
   - Peripheral
   - Register
   - Single bit.



2. Select a *List name* or type in the name of a new *List*



3. Display only your favorites

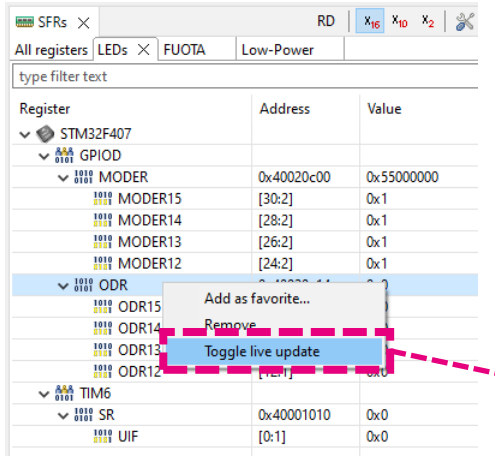


life.augmented

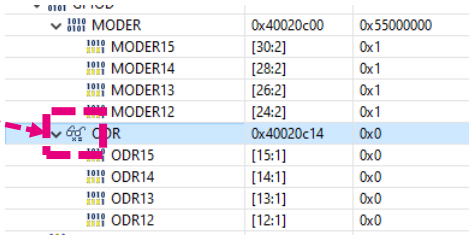
New

Live updates

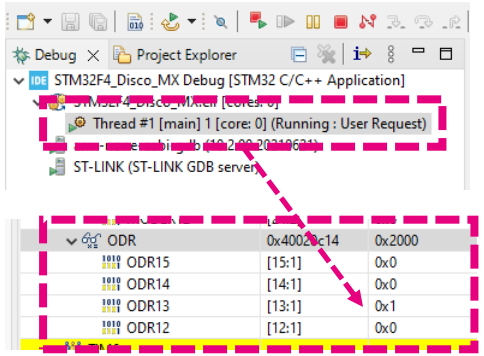
1. Right-click on register-level → *Toggle live update*
   - *Only applicable at register-level since debugger always reads 32-bits.*



2. The “google” icon indicates that live-update is enabled



3. Register values are updated “live” while target is running



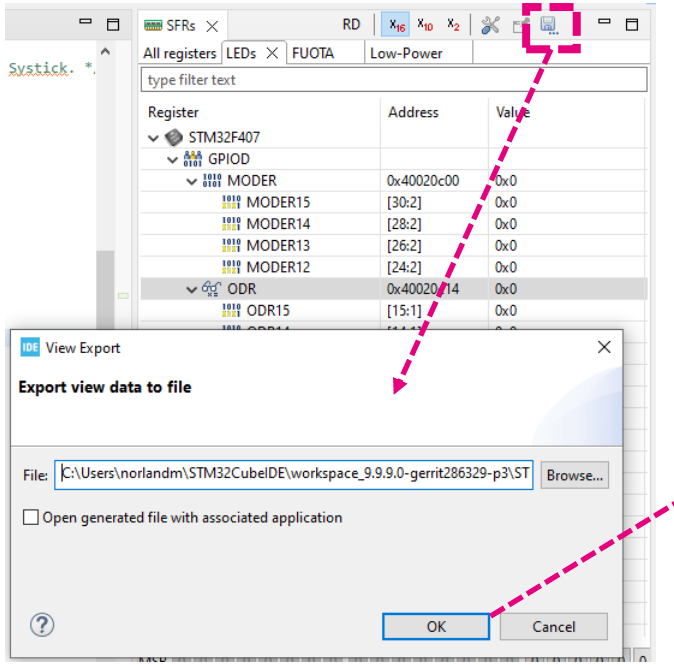
life.augmented

New

Log export and compare

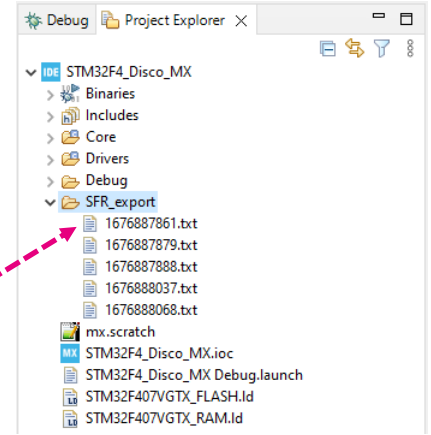
1. Select the Export scope

- All registers?
- Or just a favorite list?



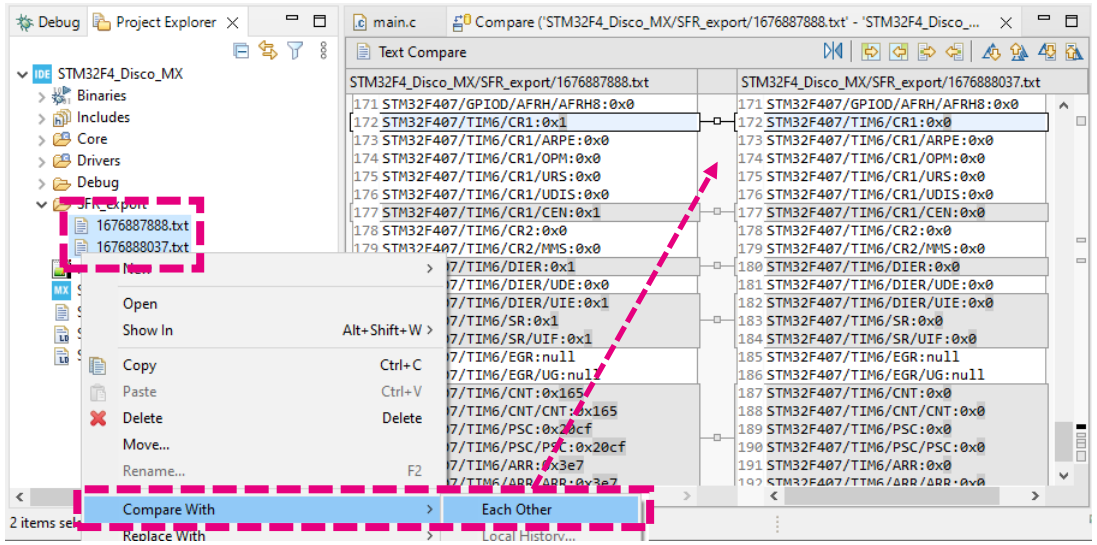
2. Logs are saved in *SFR_export*

- Named by incremental number



3. Compare two log-files

- Right-click → Compare with → each other

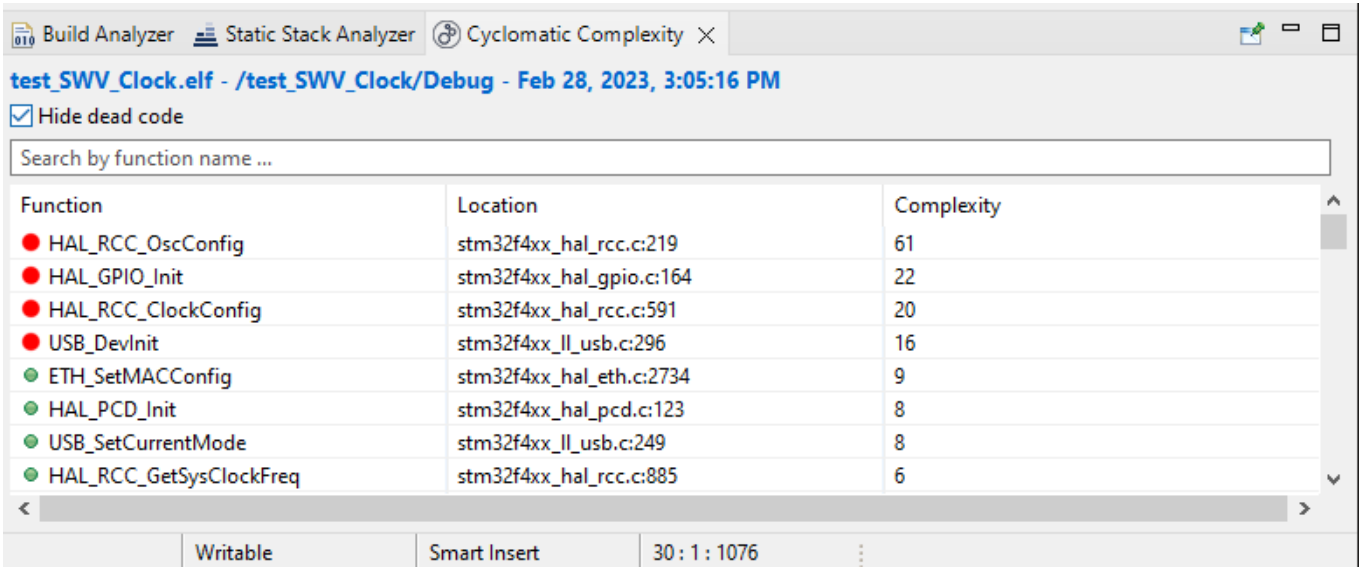


life.augmented

New

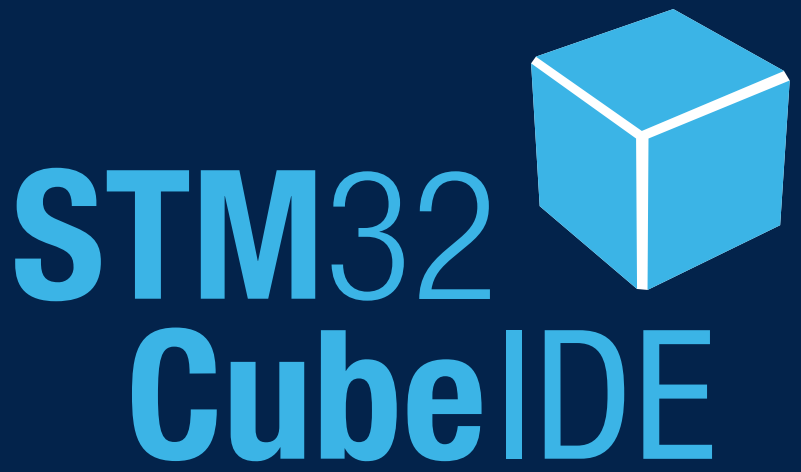
Add support for cyclomatic complexity

- Cyclomatic complexity is a measurement which give the possibility to determine the stability and level of confidence in a program.
- The new view show the level of complexity for each function.



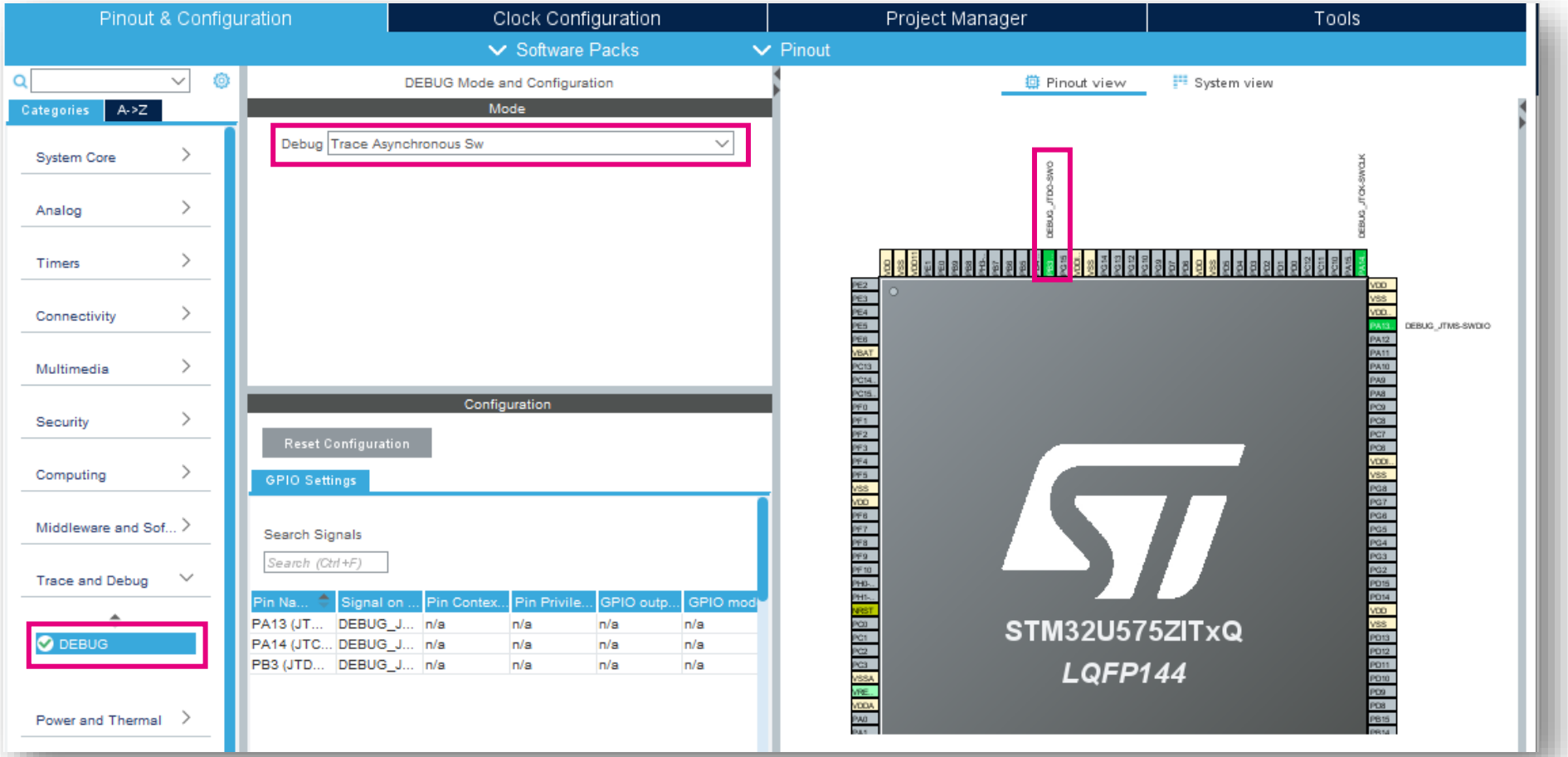
life.augmented

STM32CubeIDE



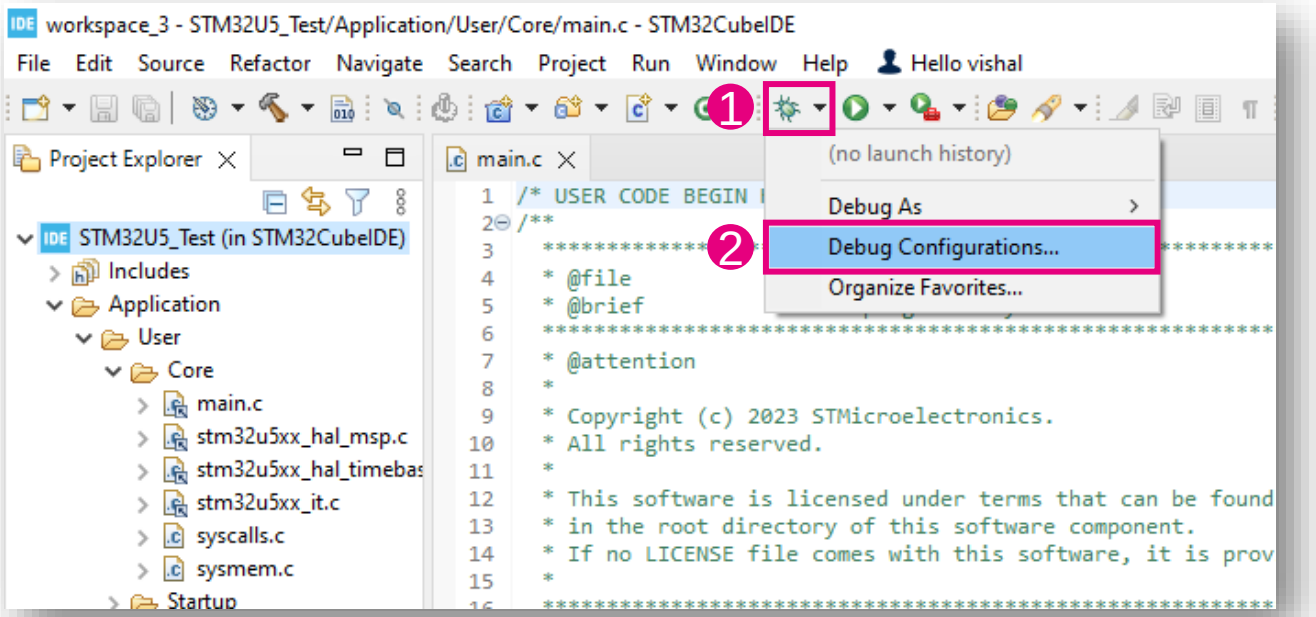
Adding SWO pin

- Enable SWO via the STM32CubeMC perspective and re-generate the code.



Enable SWV in the debugger settings(1/2)

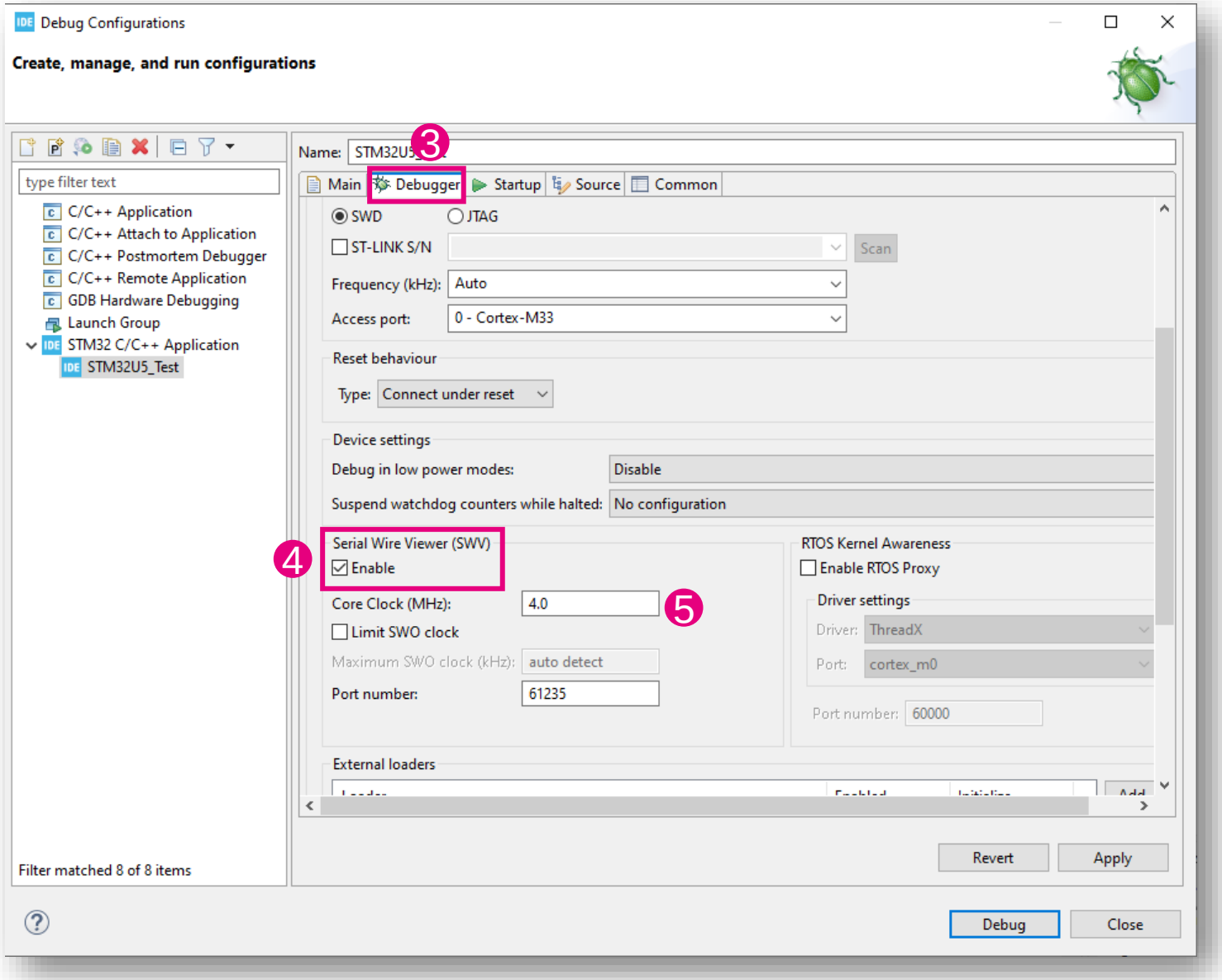
- Steps:
- Enable SWV Ftr:
  - Step 1: Go to Debug
  - Step 2: “Debug Configurations”



Enable SWV in the debugger settings (2/2)

Steps contd:

- Step 3: Go to “Debugger” tab
- Step 4: Enable SWV
- Step 5: SWV tracing requires manually setting the **Core Clock** to the correct value, **4 MHz** in our case
- Click “Apply” -> “Close”



SWV: features explored

- Using Data Trace & Live Watch
- Using Data Trace Timeline Graph
- printf() redirection
- Timing measurement
- Exception Trace Log & Timeline Graph

STM32CubeIDE

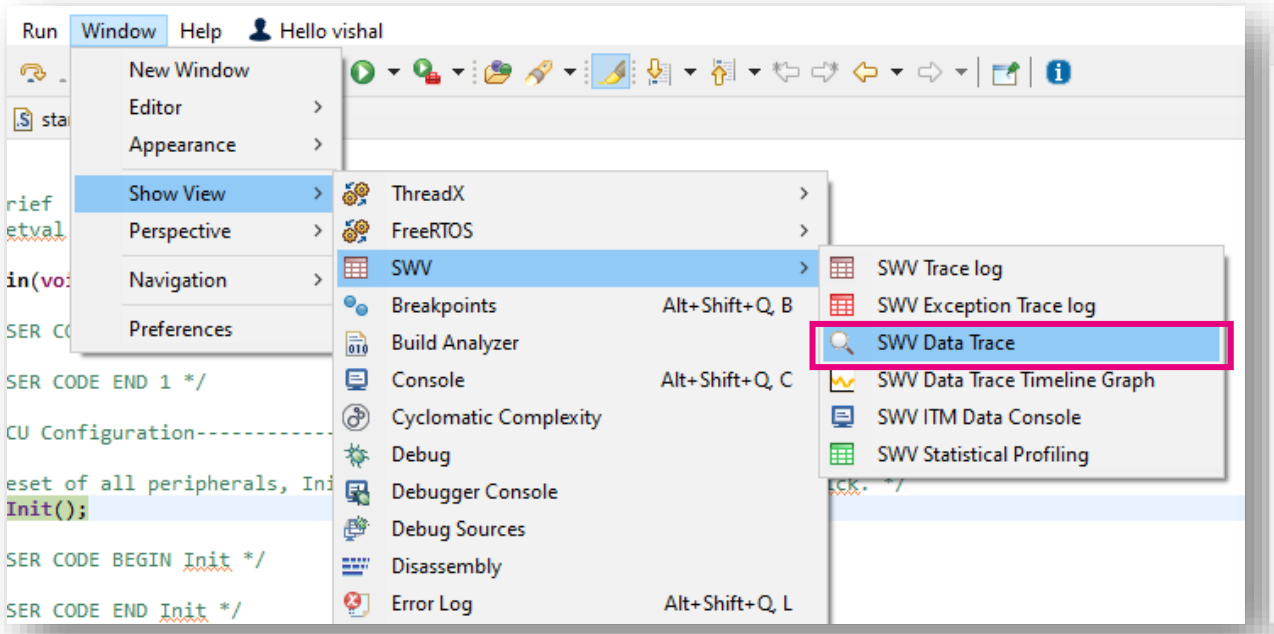
SWV: Data Trace & Live Watch

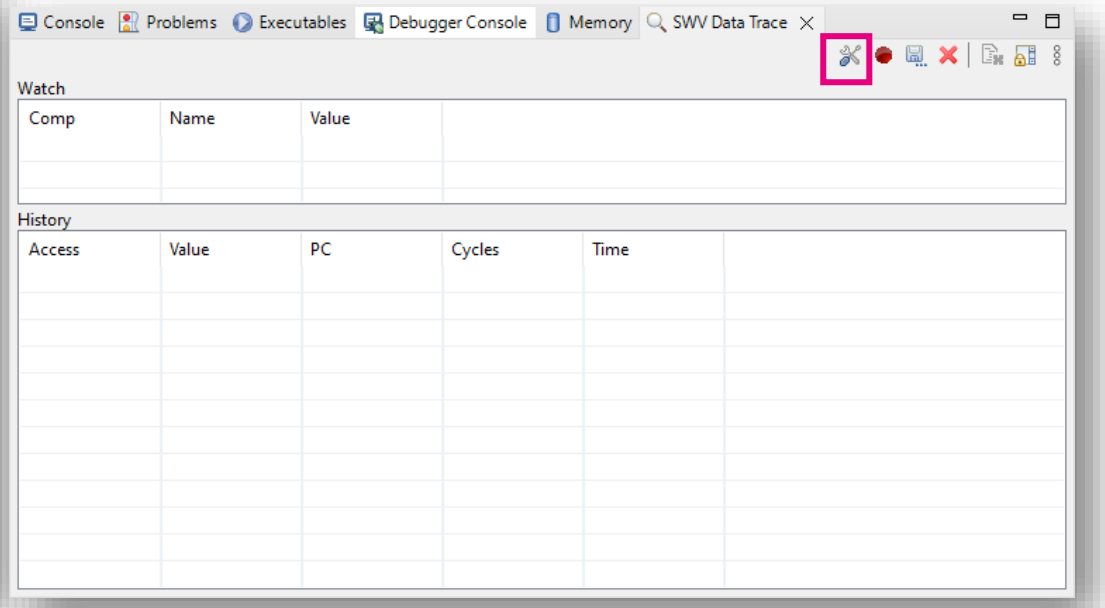
SWV: using Data Trace

- Data Trace:
  - To watch variables live
  - No need to stop on a breakpoint OR Halt execution
  - Monitor the memory reads and writes of particular variables
  - Can be used to detect unexpected RAM access.

Enable SWV Data Trace

- Start Debug session
- Go to **Window -> Show View -> SWV** and click **SWV Data Trace**
- Click on “Configure Trace” button to open the Serial Wire Viewer settings

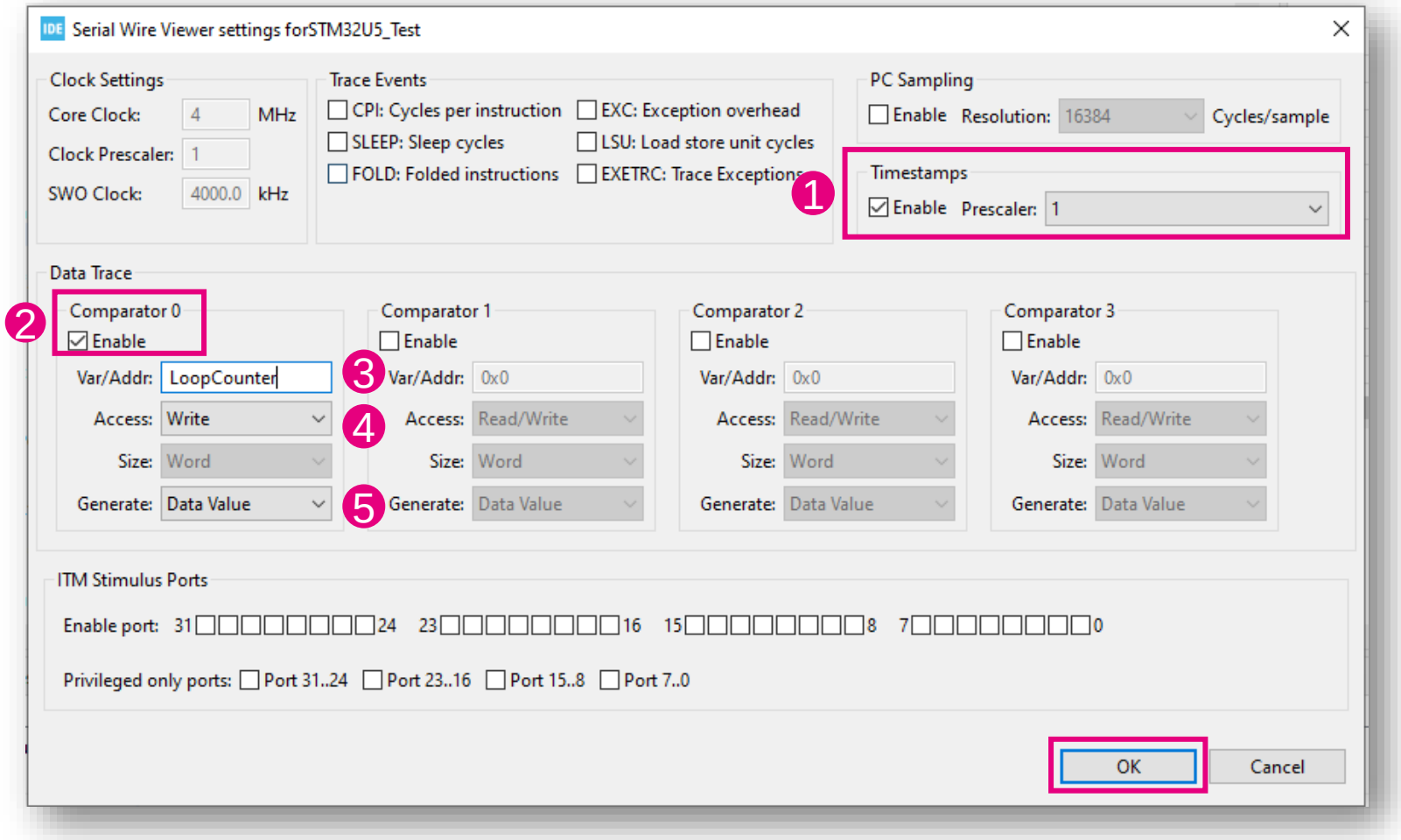




life.augmented

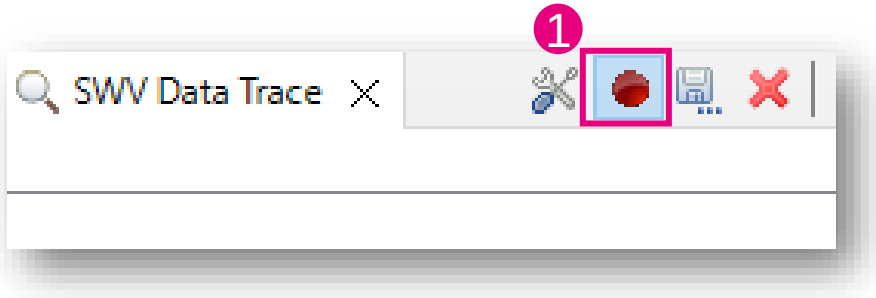
Serial Wire Viewer settings

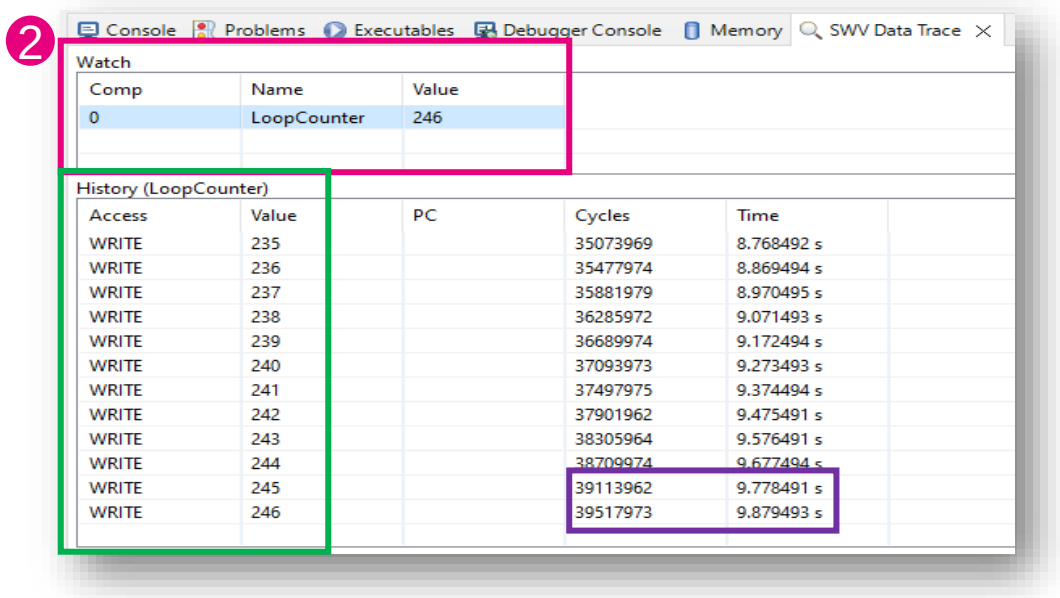
- Step 1: Enable “Timestamps”
- Step 2: Enable **Trace Comparator** 0
- Step 3: Add the variable name/address to be traced (LoopCounter)
- Step 4: Select “Write” in the Access list (only monitor Write access)
- Step 5: Select “Data Value” to be logged, click OK
- Note: Keep other trace events disabled to avoid chances of overflow in the SWO line



SWV: Using Data Trace

- Step 1: Click on “Start Trace Button”
- Start Execution
- Step 2: Observe the Data live on the “SWV Data Trace Console”
  - Live Watch
  - History
  - Timestamps(~101 ms delay between each **Write** in our Case)





STM32CubeIDE

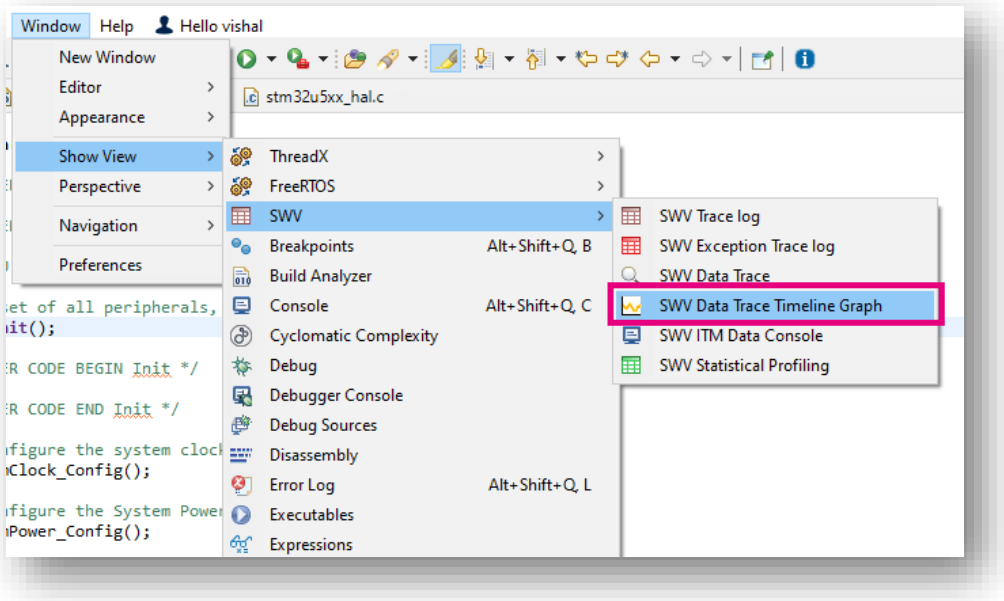
SWV: Data Trace Timeline Graph

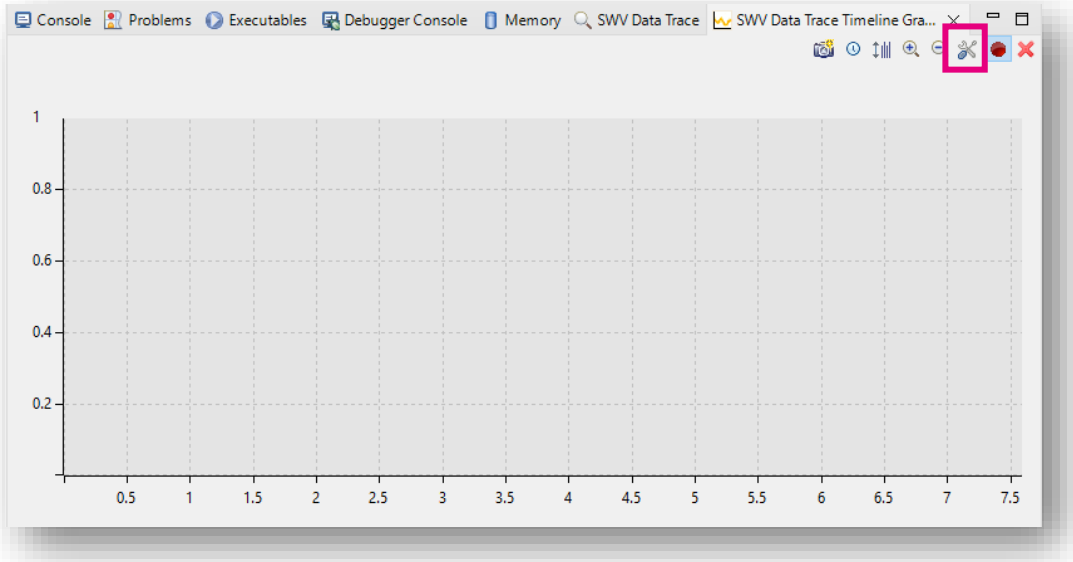
SWV: Using Data Trace Time Graph

- Used for Graphical Data plot of variables in real-time
- An “Oscilloscope style” data plot graph
- Pots the Graph of Data Vs Time OR Data Vs Clock Cycles
- Also allows to save the graphs as images to study later

Enable SWV Data Trace Timeline Graph

- Start Debug session
- Go to **Window -> Show View -> SWV** and click **SWV Data Trace Timeline Graph**
- Click on “Configure Trace” button to open the Serial Wire Viewer settings



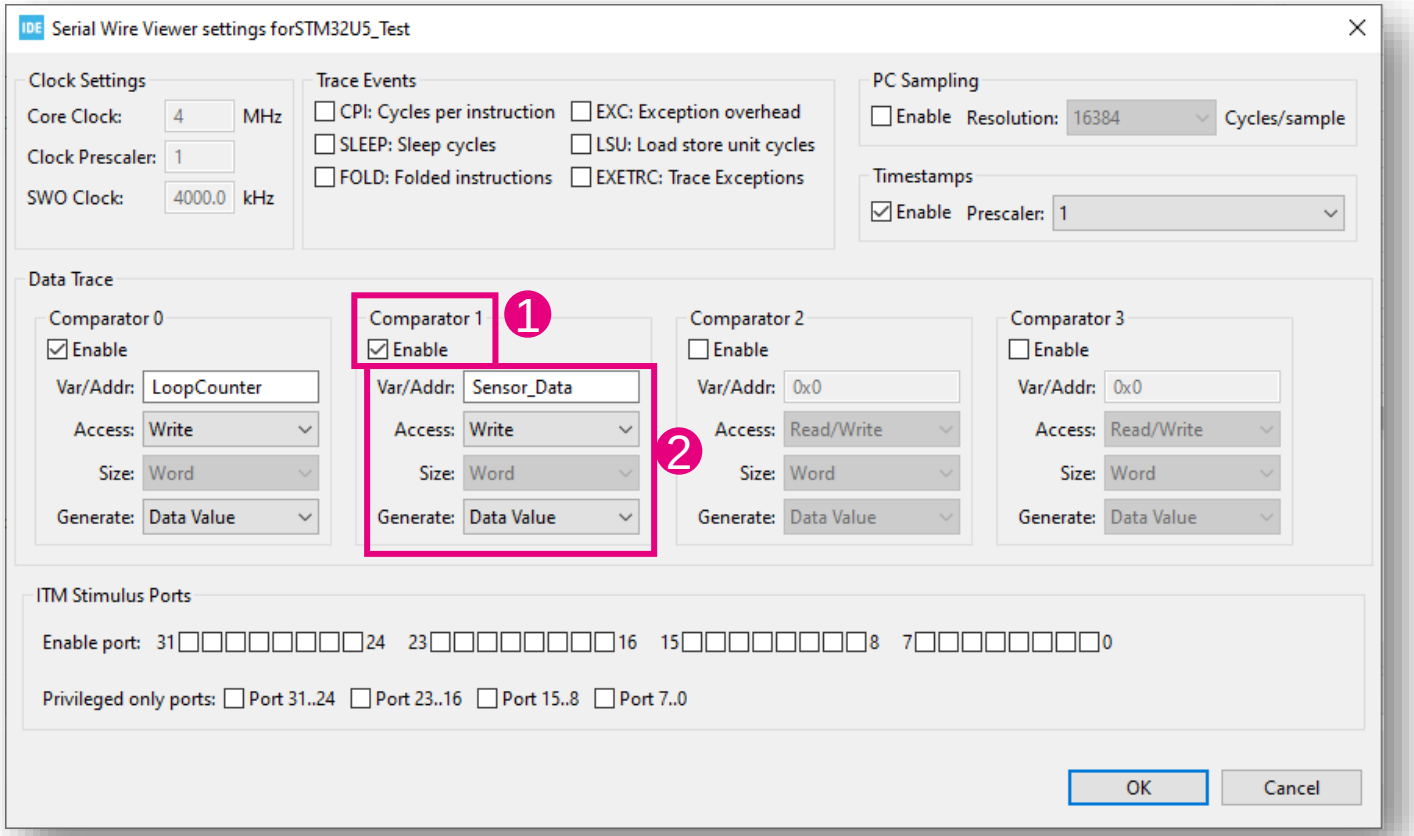


life.augmented

Configure SWV settings

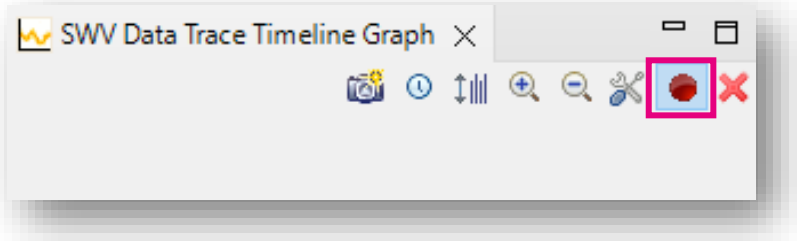
- Step 1: In the “Data Trace” settings, add Comparator 1
- Step 2: add “Sensor_Data” and also set the other fields of Access, Size, Data Value as shown here.

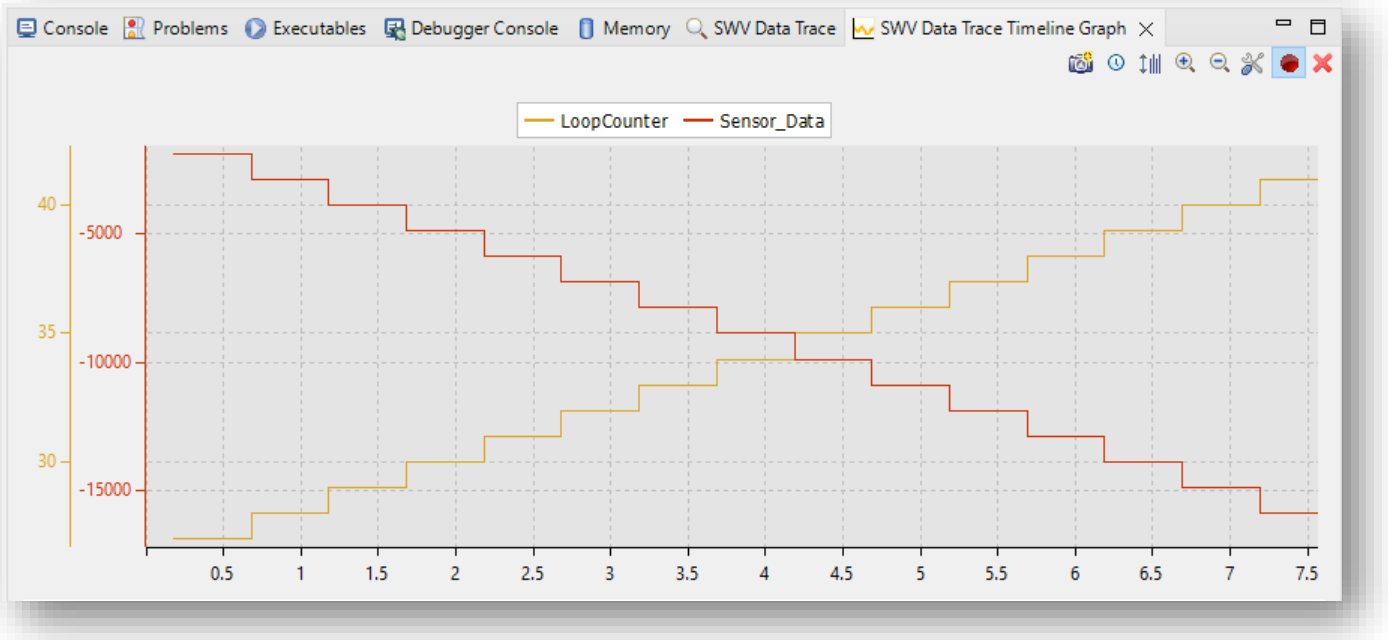
```
/* USER CODE BEGIN WHILE */
while (1)
{

    LoopCounter++;
    HAL_Delay(500);

    if(LoopCounter>100)
        LoopCounter=0;

    //Used to simulate general data pattern, to be observed in Trace Logs.
    if(LoopCounter<25)
    {
        Sensor_Data = Sensor_Data + 5;
    }
    else
    {
        Sensor_Data = Sensor_Data -1000;
    }


  /* USER CODE END WHILE */
```



SWV: Using Data Trace Timeline Graph

- Click on “Start Trace” button
- Start Execution
- You shall now start seeing some graphs being plotted in the console
- Note: Keep other trace events disabled to avoid chances of overflow in the SWO line

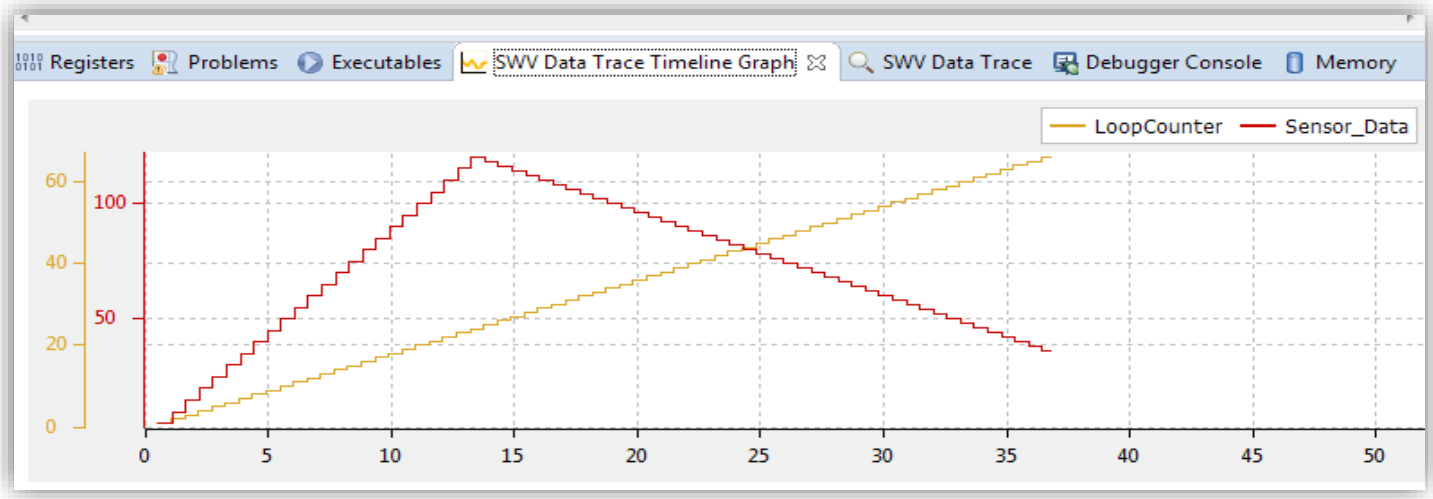




life.augmented

SWV: Using Data Trace Timeline Graph

- Few Sample Graphs generated
- Multiple Data variables can be plotted
- Color coding to identify different variables



STM32CubeIDE

SWV: printf() redirection

SWV: printf() redirection

- Typically, we use printf() messages sent via UART to troubleshoot the application during run time
- SWV can be used for **printf() re-direction**
- Debug messages are sent to a “Console” window in the debugger using the **Serial Wire Output (SWO)** pin
- No need for a USB or UART cable

Redirect printf() to SWO pin

- Include stdio.h library to declare printf functionality
- Add code to print messages
- To redirect printf() to SWO pin
  - Redefine _write() function to send data to SWO pin using ITM_SendChar() function
  - Weak _write() included in syscalls.c file
  - Same technique can be used to print to USART pins

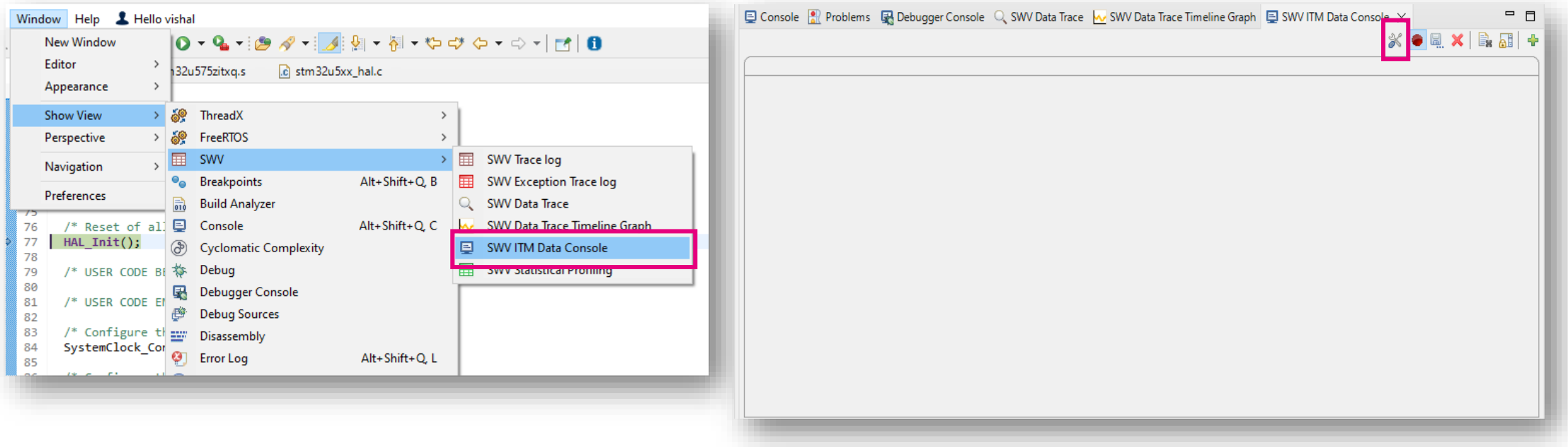
```
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */
```

```
109   /* Initialize all configured peripherals */
110   MX_GPIO_Init();
111   /* USER CODE BEGIN 2 */
112   printf("GPIO Init done\n");
113   /* USER CODE END 2 */
114
115   /* Infinite loop */
116   /* USER CODE BEGIN WHILE */
117   while (1)
118   {
119     /* USER CODE END WHILE */
120
121     /* USER CODE BEGIN 3 */
122     printf("In while loop \n");
123     HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
124     HAL_Delay(500);
125     printf("Counter = %i\n", LoopCounter++);
```

```
/* USER CODE BEGIN 4 */

int _write(int file, char *ptr, int len)
{
 int DataIdx;

  for(DataIdx=0; DataIdx<len; DataIdx++)
  {
    ITM_SendChar(*ptr++);
  }
  return len;
}

/* USER CODE END 4 */
```

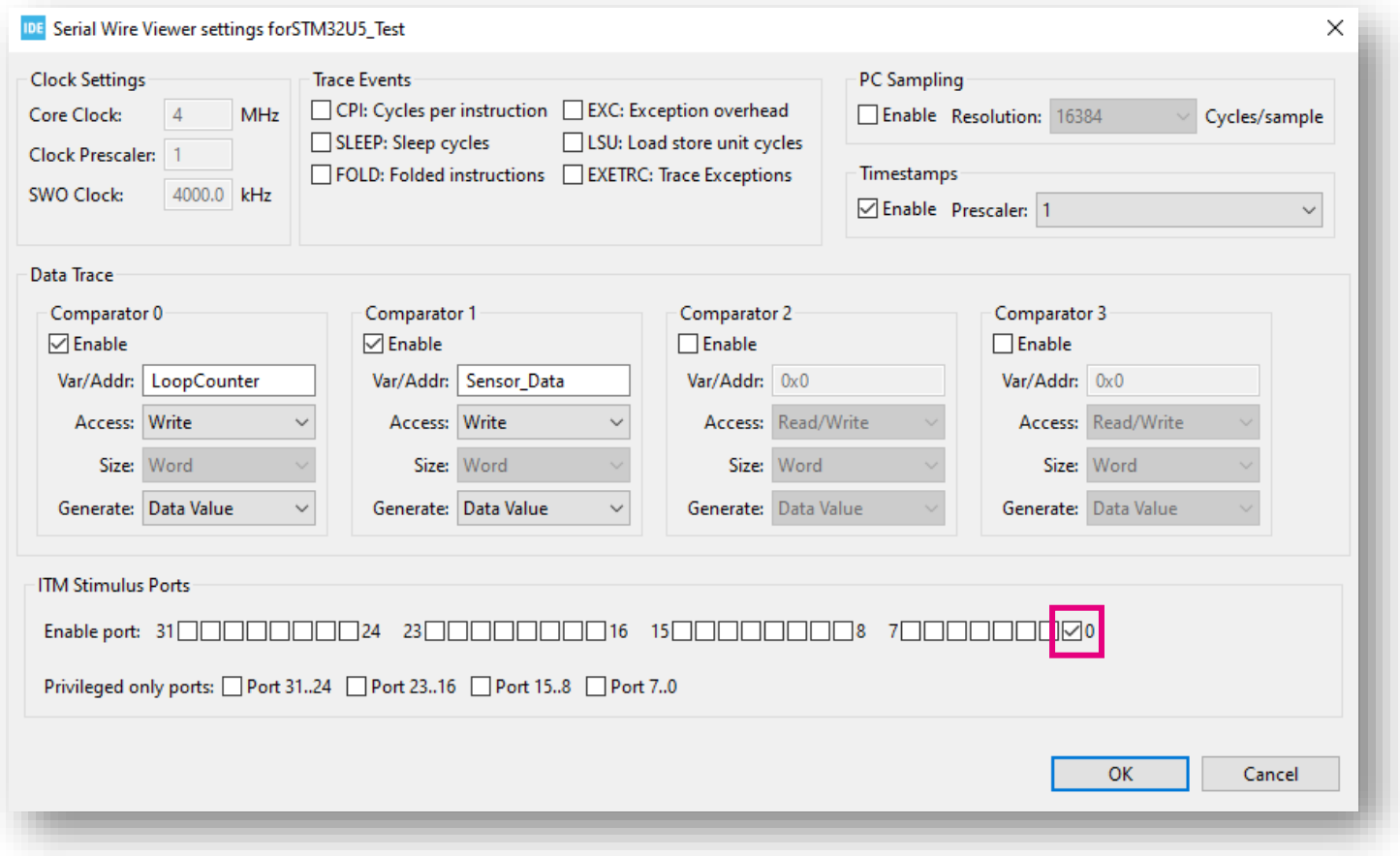
Enable SWV ITM Data Console

- Start Debug session
- Go to **Window -> Show View -> SWV** and click **SWV ITM Data Console**
- Click on “Configure Trace” button to open the Serial Wire Viewer settings



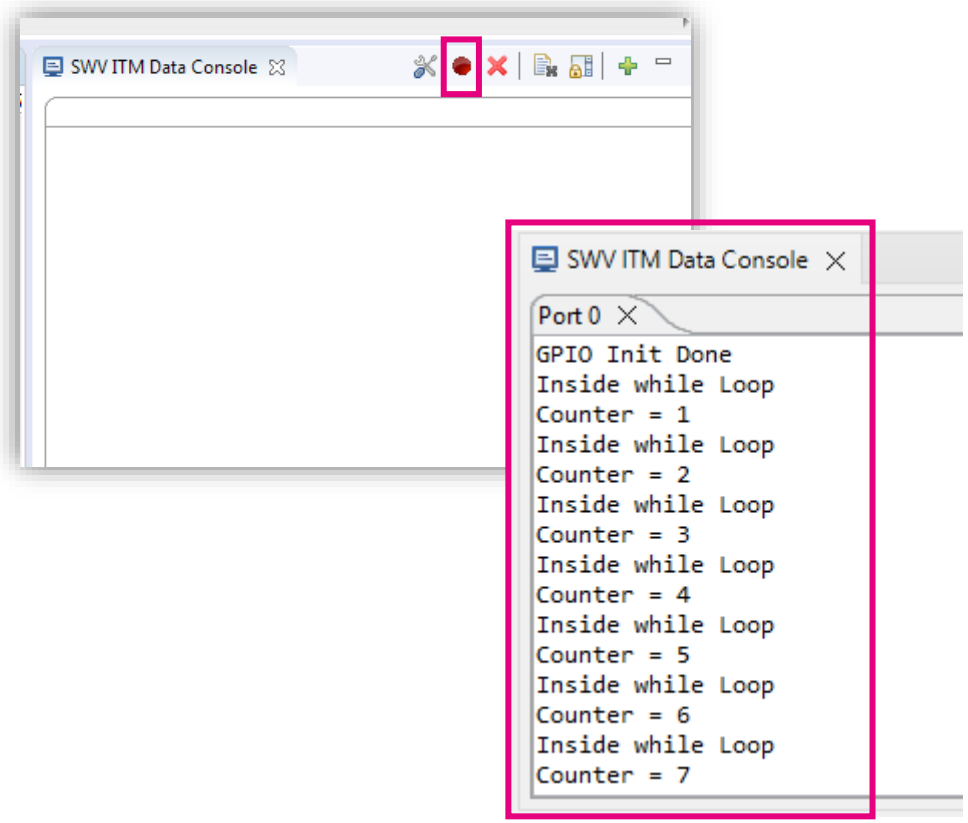
Configure SWV settings

- Enable Channel 0 from the ITM Ports
- Note: Keep other trace events disabled.



SWV printf() redirection

- Click on “Start Trace” button
- Start Execution
- View printf() messages in the “SWV Data Console”



STM32CubeIDE

SWV: Timing Measurement

SWV: Timing Measurement

- Used to **measure execution timings** for a process/function
- **Step 1**: Add ITM Stimulus Port register definition MACRO to your source code (main.c), which allows us to write to an ITM Port
  - `#define ITM_Port32(n)  (*((volatile unsigned long *)(0xE0000000+4*n)))`
- **Step 2**: Write 2 different values to the **ITM Port 31** ***before*** and ***after*** the desired functions
  - `ITM_Port32(31) = 1;`
  - `ITM_Port32(31) = 2;`

1

```
78 /* Private user code ---------------------------------------------------
79 /* USER CODE BEGIN 0 */
80 #define ITM_Port32(n)   (*((volatile unsigned long *)(0xE0000000+4*n)))
81 /* USER CODE END 0 */
82
83 /**
84   * @brief  The application entry point.
85   * @retval int
86   */
87 int main(void)
88 {
89   /* USER CODE BEGIN 1 */
```

2

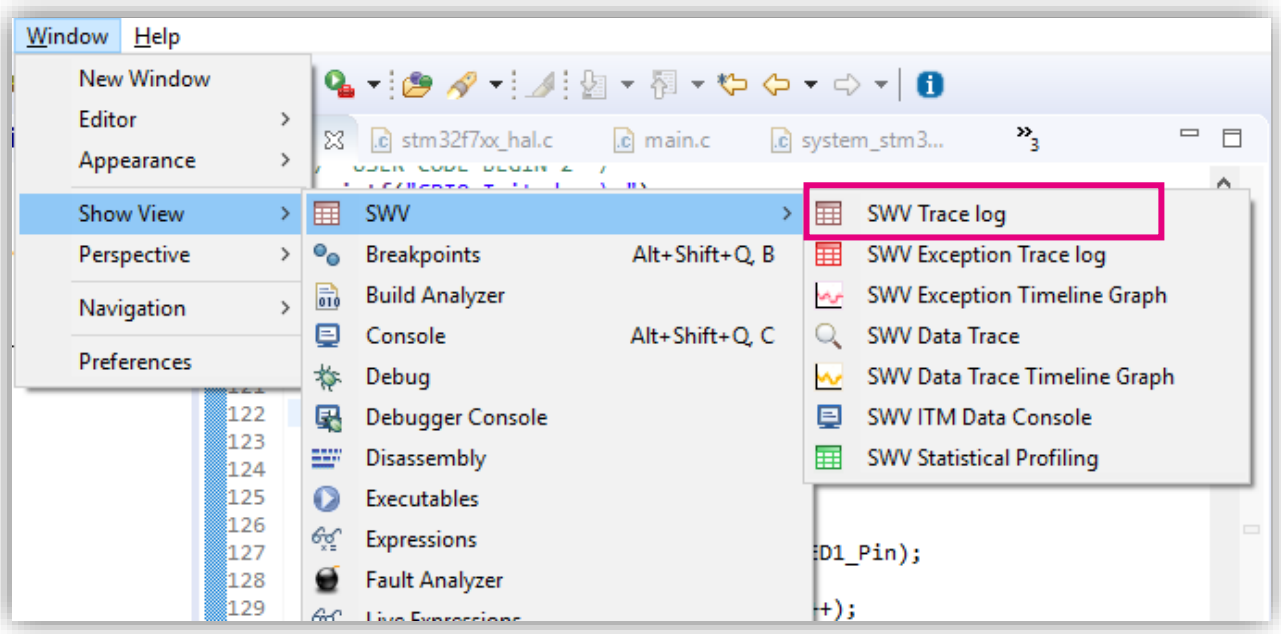
main.c | syscalls.c | stm32f7xx_hal.c | startup_s

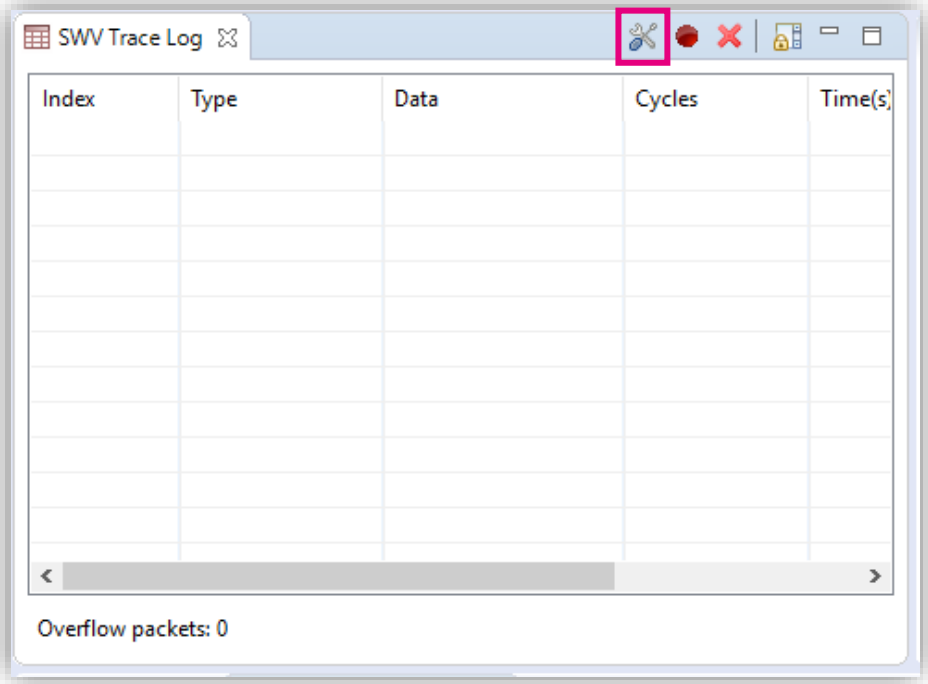
```
106   printf("System Init done\n");
107   /* USER CODE END SysInit */
108
109   /* Initialize all configured peripherals */
110   ITM_Port32(31) = 1;
111
112
113   MX_GPIO_Init();
114   /* USER CODE BEGIN 2 */
115   printf("GPIO Init done\n");
116
117
118   ITM_Port32(31) = 2;
119   /* USER CODE END 2 */
120
```

life.augmented

Enable SWV ITM Data Console

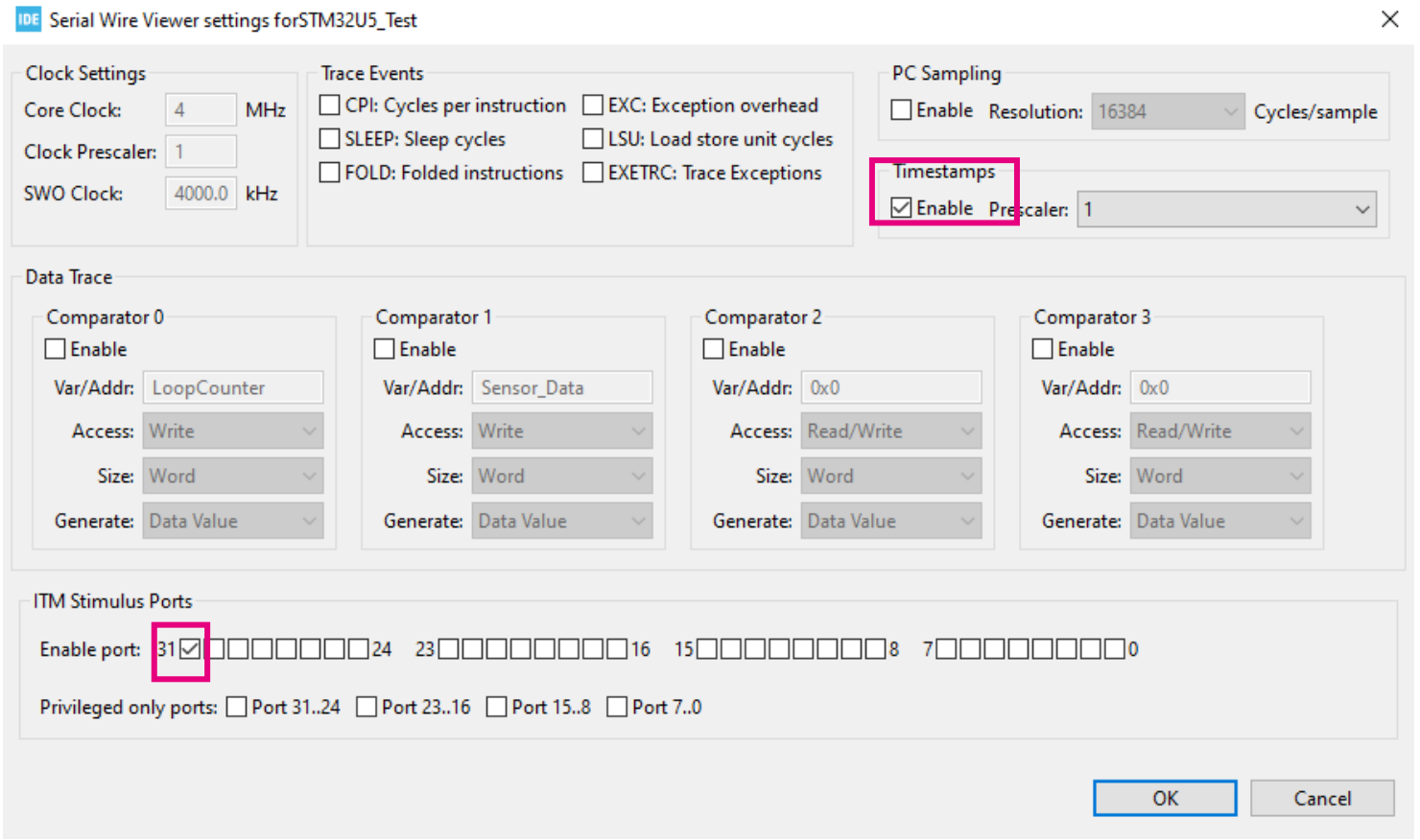
- Start Debug Session
- Go to **Window -> Show View -> SWV** and click **SWV Trace Log**
- Click on “Configure Trace” button to open the Serial Wire Viewer settings





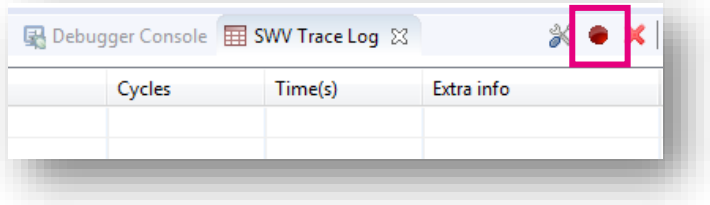
SWV: Timing Measurement

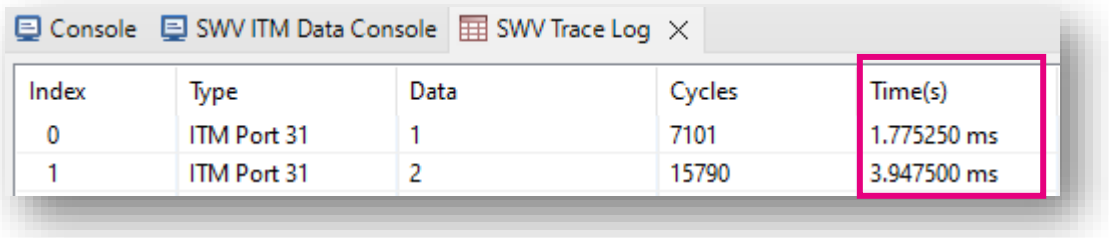
- Enable **Timestamps**
- Enable Channel 31 *only* from the ITM Ports
- Note: Keep other trace events disabled to avoid chances of overflow in the SWO line



SWV: Timing Measurement

- Click on “Start Trace” button
- Start Execution
- Observe the Trace values & timestamps in the “SWV Trace Log Console”





STM32CubeIDE

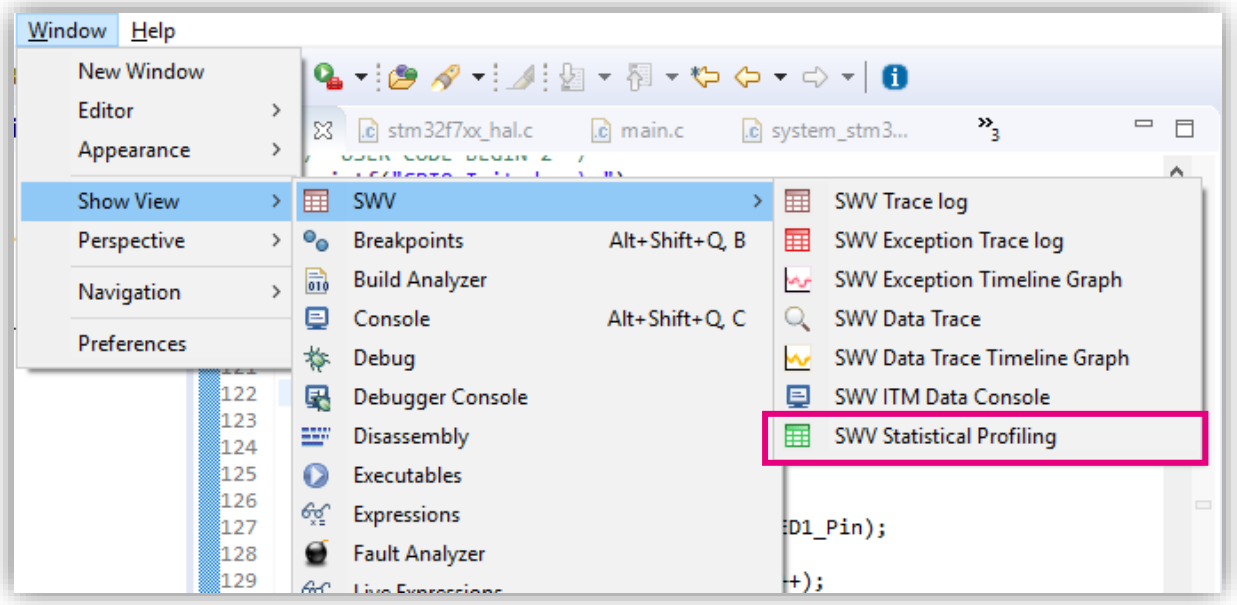
SWV: Statistical Profiling

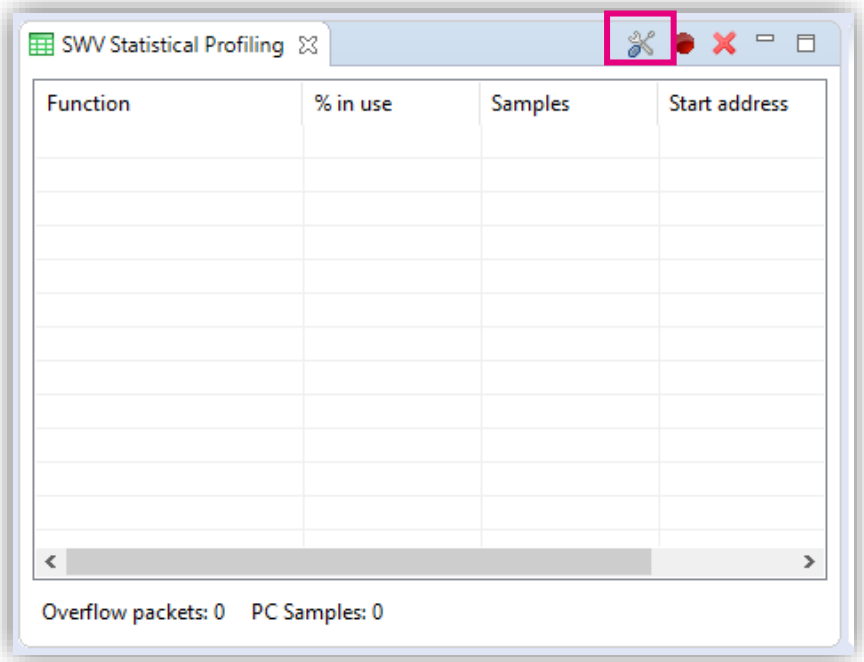
SWV: Statistical Profiling

- Statistical profiling feature of the SWV provides a way to quickly analyze the performance of the application code and optimize speed.

Enable SWV Statistical Profiling

- Start Debug Session
- Go to **Window -> Show View -> SWV** and click **SWV Statistical Profiling**
- Click on “Configure Trace” button to open the Serial Wire Viewer settings

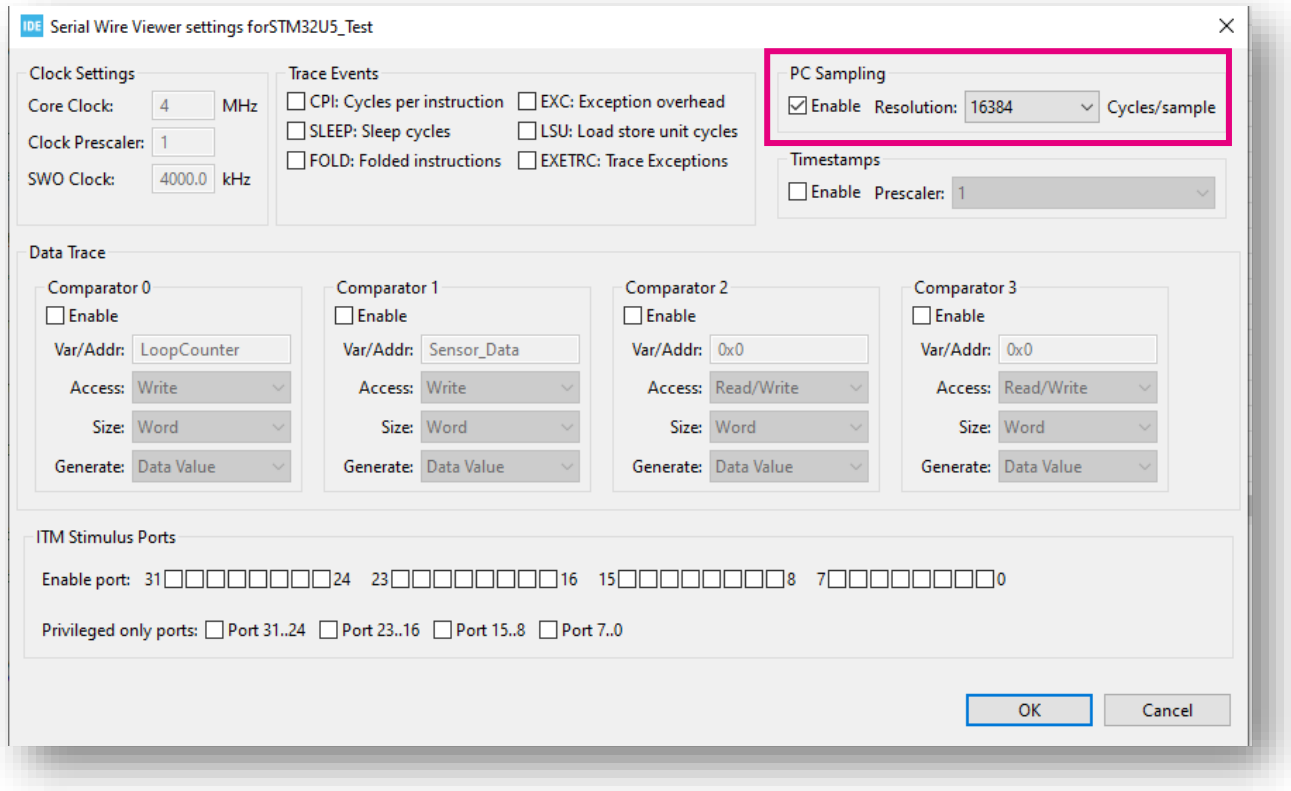




life.augmented

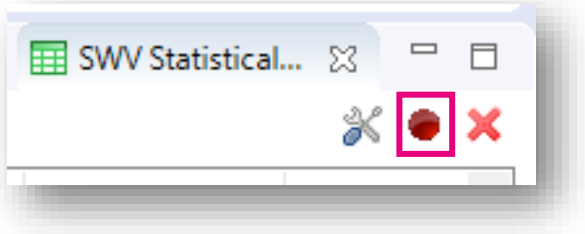
Configure SWV settings

- Enable PC Sampling and click ok
- Note: Keep other trace events disabled to avoid chances of overflow in the SWO line

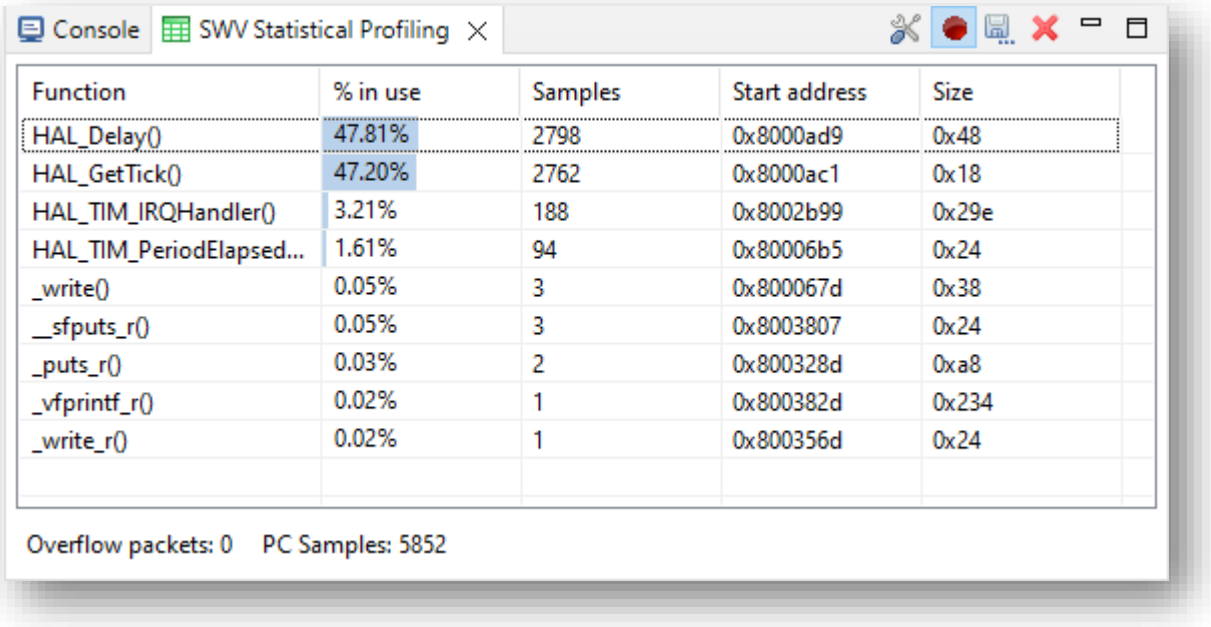


Start Statistical Profiling view

- Click on the Start Trace button.



- Execute the code.
- Halt the execution in order to see a statistical profile of the application when it was running at that particular time frame.



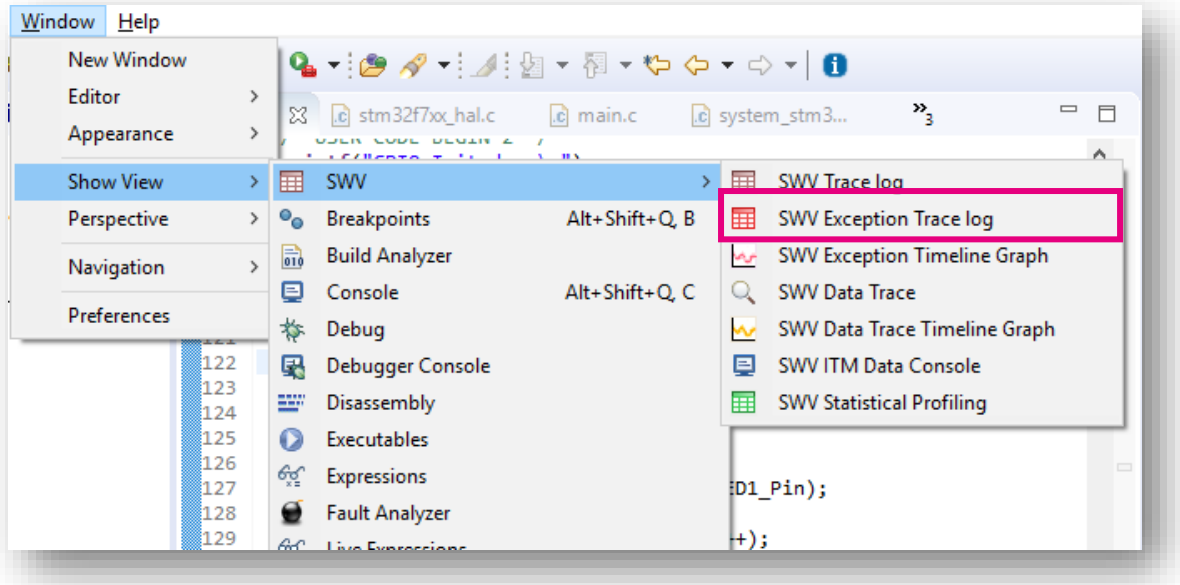
life.augmented

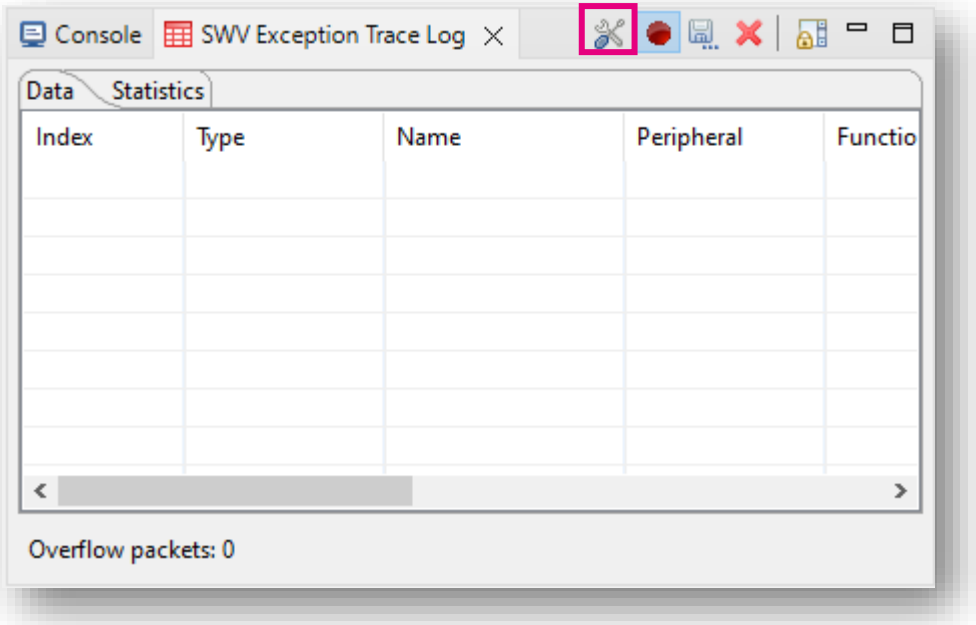
STM32CubeIDE

SWV: Exception Trace Log

Enable SWV Exception Trace Log

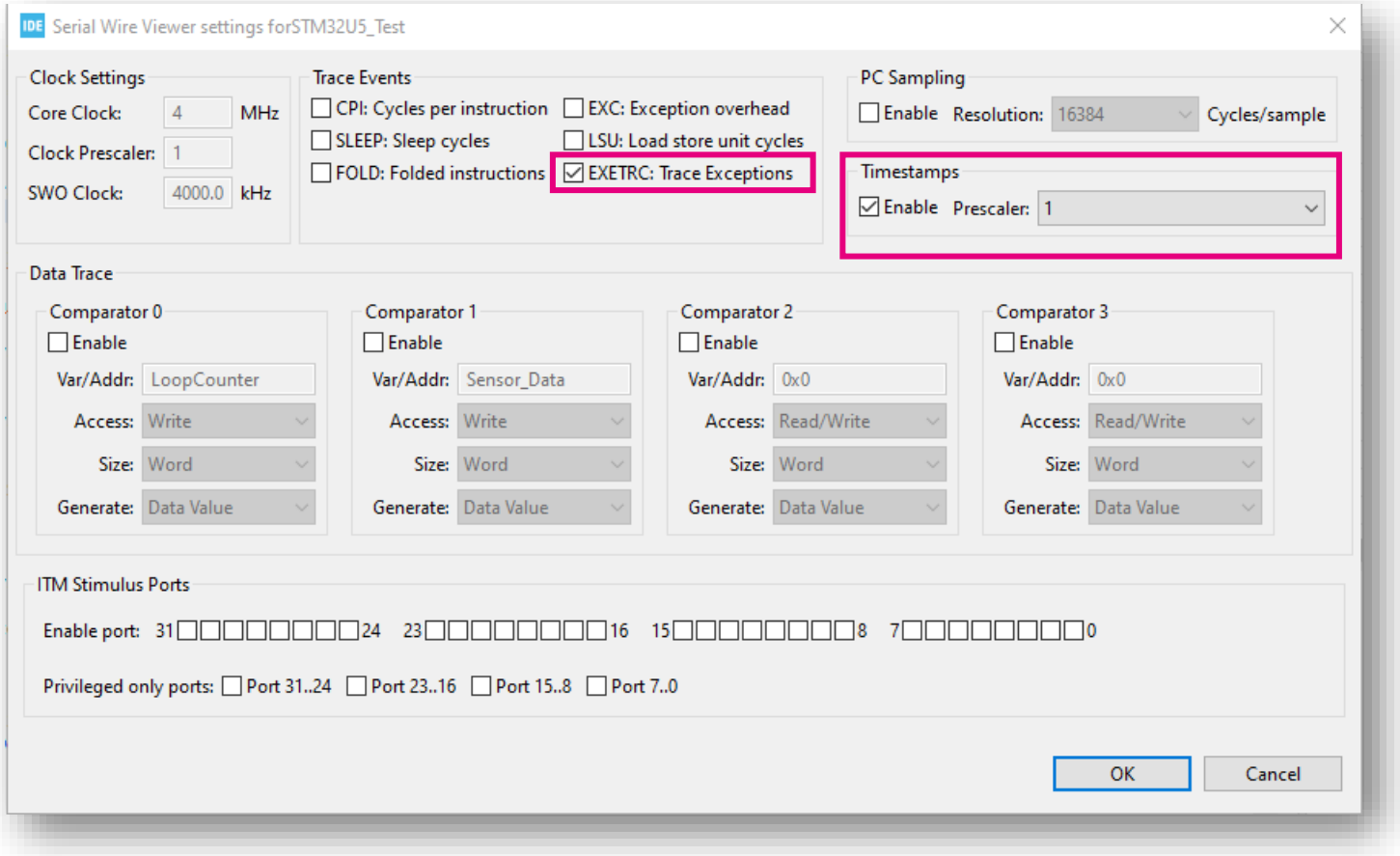
- Start Debug session
- Go to **Window -> Show View -> SWV** and click **SWV Exception Trace Log**
- Click on “Configure Trace” button to open the Serial Wire Viewer settings





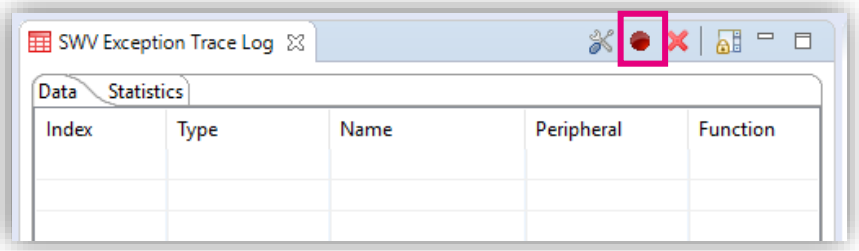
Configure SWV settings

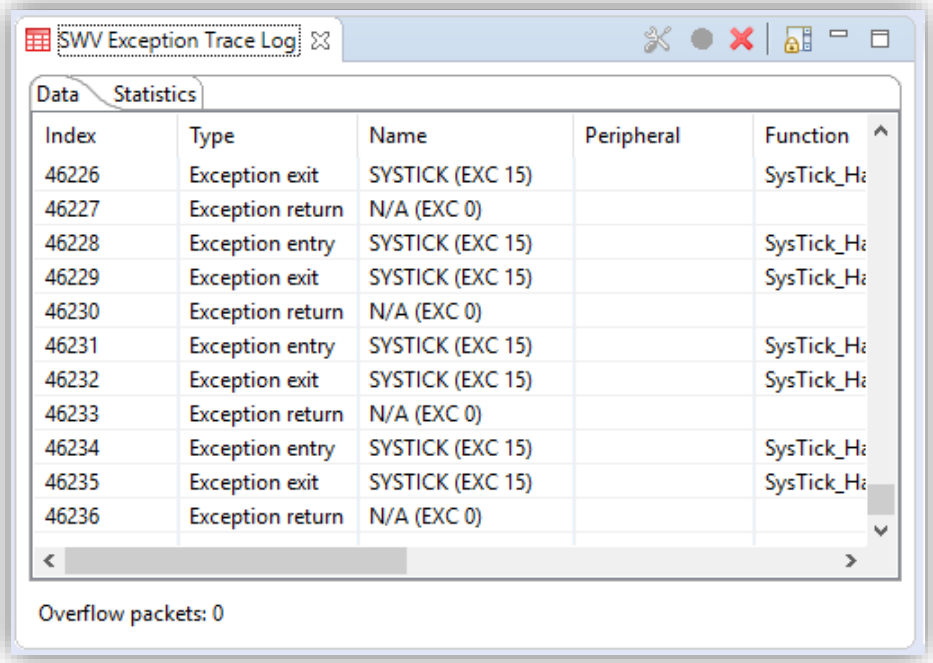
- Enable Timestamps
- Enable EXETRC: Trace Exceptions
- Click OK
- Note: Keep other trace events disabled to avoid chances of overflow in the SWO line

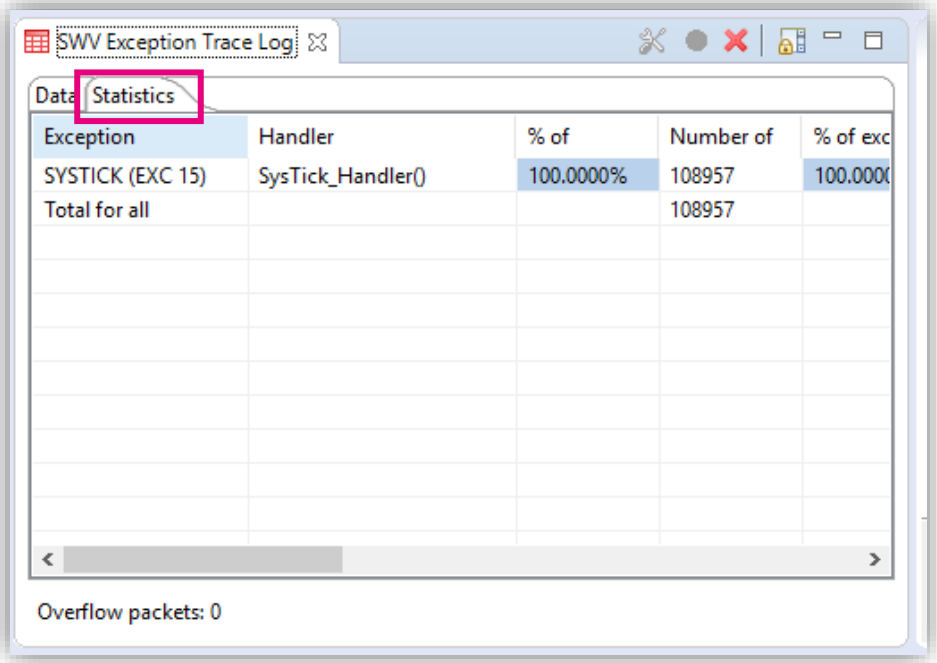


SWV: Exception Trace Log

- Click on Start Trace button
- Execute the code to see exception logs
- Click on the statistics tab to the exception profile.





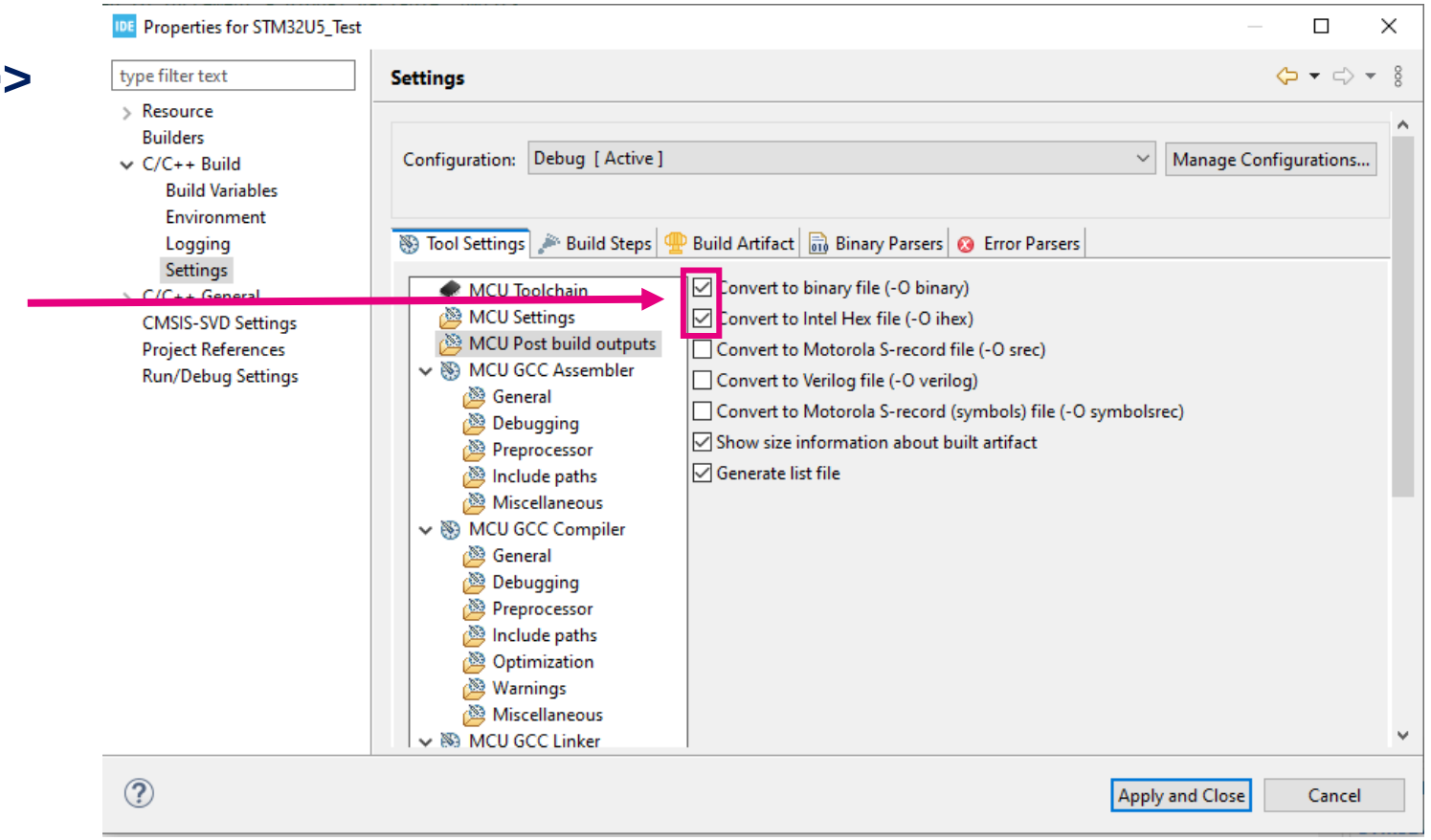


STM32CubeIDE

More IDE Options

IDE options: Good-to-know

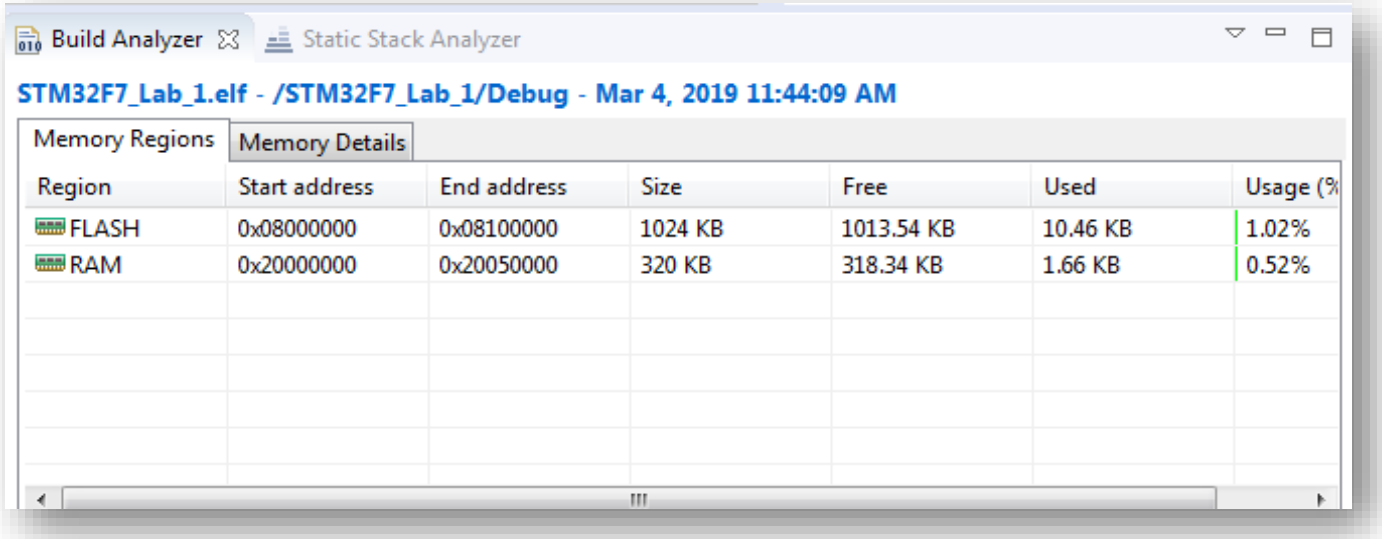
- To generate binary (.bin) and hex (.hex) output files
  - Go to **Project >> Properties >> C/C++ Build >> Settings >> Tool Settings >> MCU Post build outputs**
  - Enable these 2 checkboxes to generate binary (.bin) and hex (.hex) output files



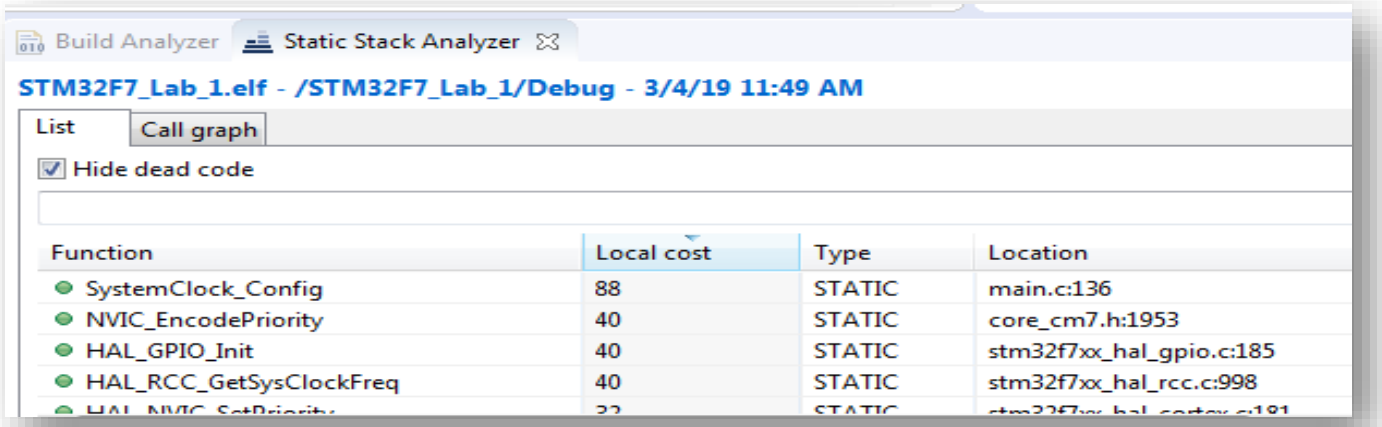
life.augmented

IDE options: Good-to-know

- To check the **Flash and RAM consumption** : **Build Analyzer -> Memory Regions**

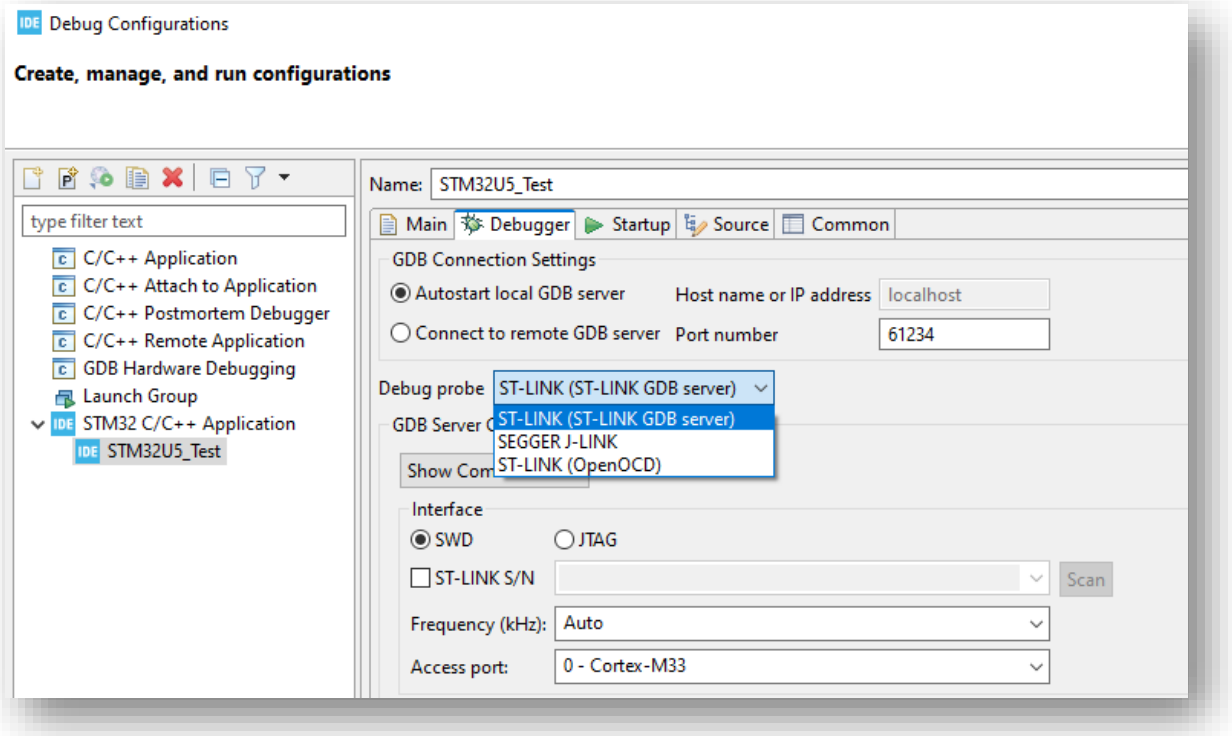


- To check **stack** consumption of individual functions: **Static Stack Analyzer -> List**



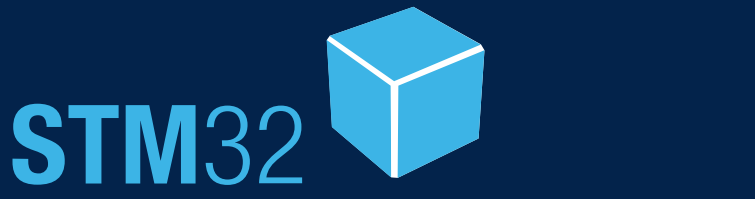
IDE options: Good-to-know

- STM32CubeIDE supports below Debuggers
  - **ST-LINK (GDB server)**
  - **ST-LINK (OpenOCD)**
  - **SEGGER J-LINK**



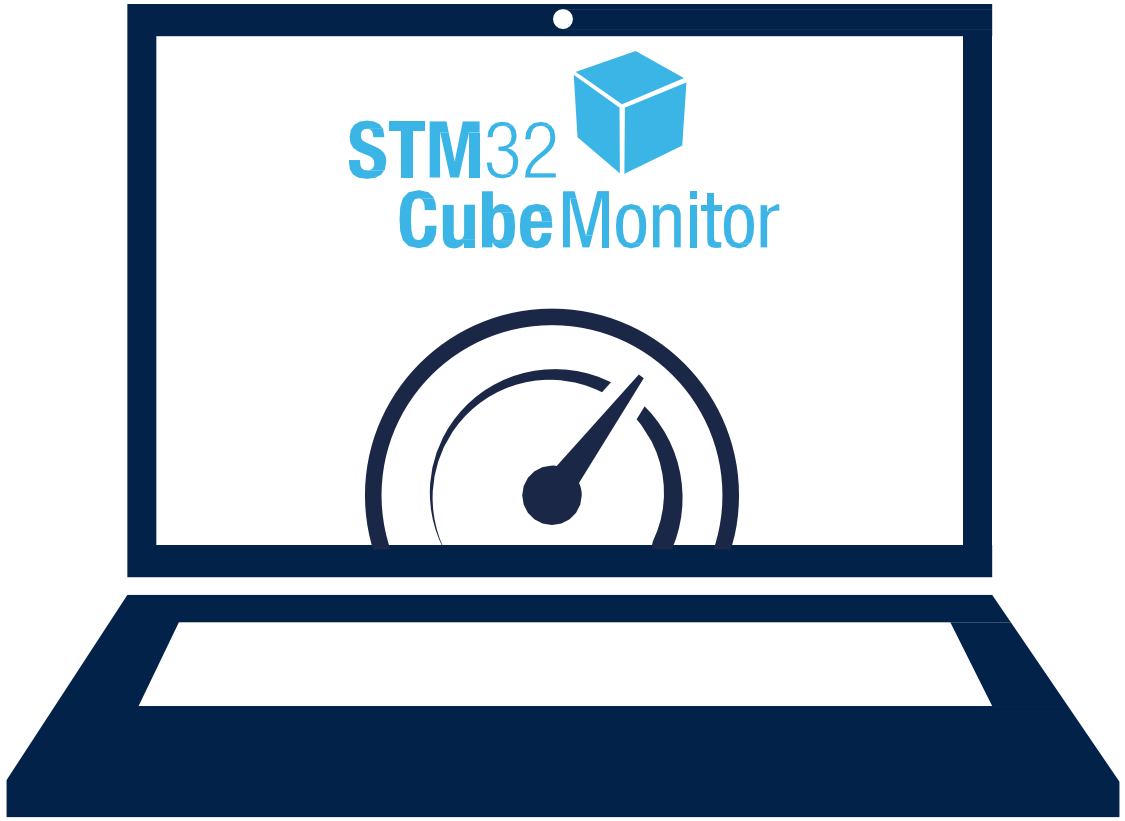
STM32CubeMonitor

Several monitors



life.augmented

Introducing STM32CubeMonitor



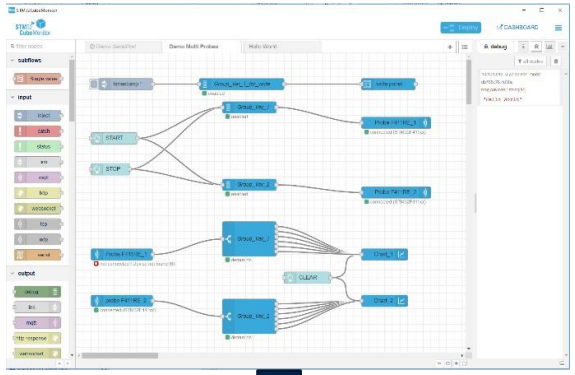
Real-time variable monitoring during application runtime

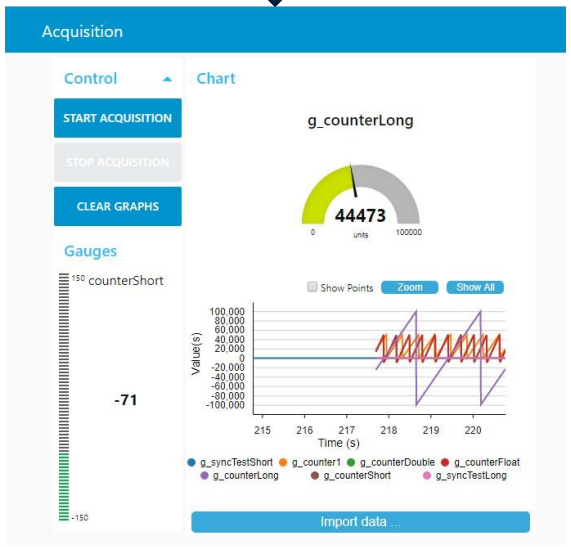
Flexible graphical visualization

Direct support of the Node-RED® open community

STM32 CubeMonitor

Key features





Monitoring application variables during runtime

- Non-intrusive tool to follow application behavior without interruption
- Real-time analysis to finetune application configuration

Drag & drop creation of dashboard UI

- Large choice of graphical components (gauges, bar graphs, plots...)
- Customize settings. No need for programming.

Graphical visualization on any display

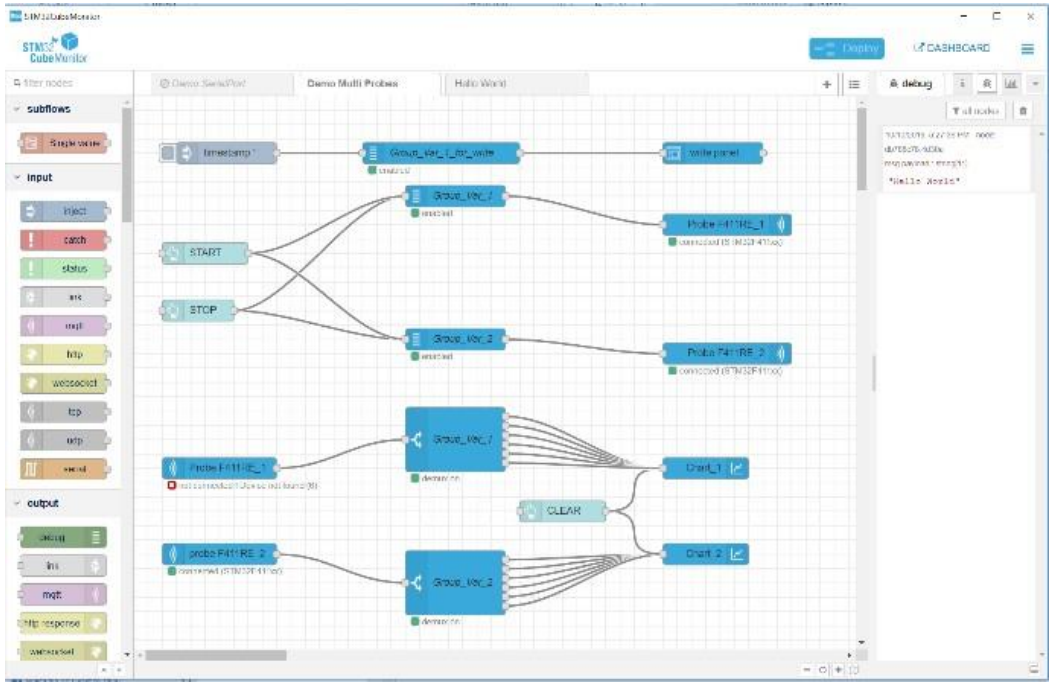
- Multi-OS tool: direct support of PC, tablets and smartphones
- Remote monitoring

life.augmented

Data visualization

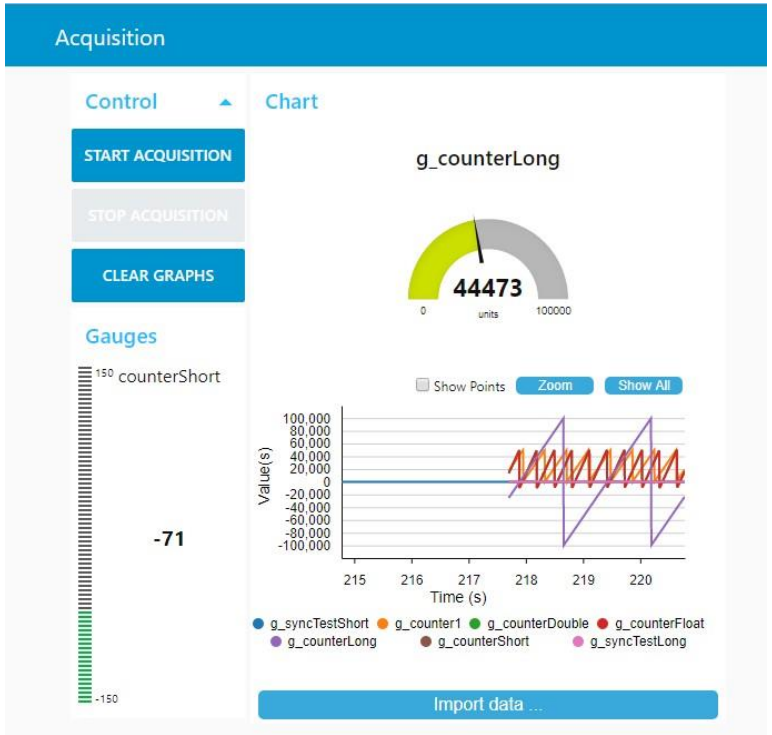
Design mode to create

Build and edit the logical data flow and graphical rendering of the custom monitoring UI.



Dashboard mode to visualize

Use the monitoring UI built previously and visualize locally or remotely.



STM32 CubeMonitor

Open community

Direct support of Node-RED community

Flow-based visual programming for connecting hardware devices, APIs and online services (a flow is composed of connected nodes: input, output, function etc.)

Unlimited possibilities

Large choice of ready-to-use nodes to easily expand STM32CubeMonitor default palette.

life.augmented

Presentation of the tool

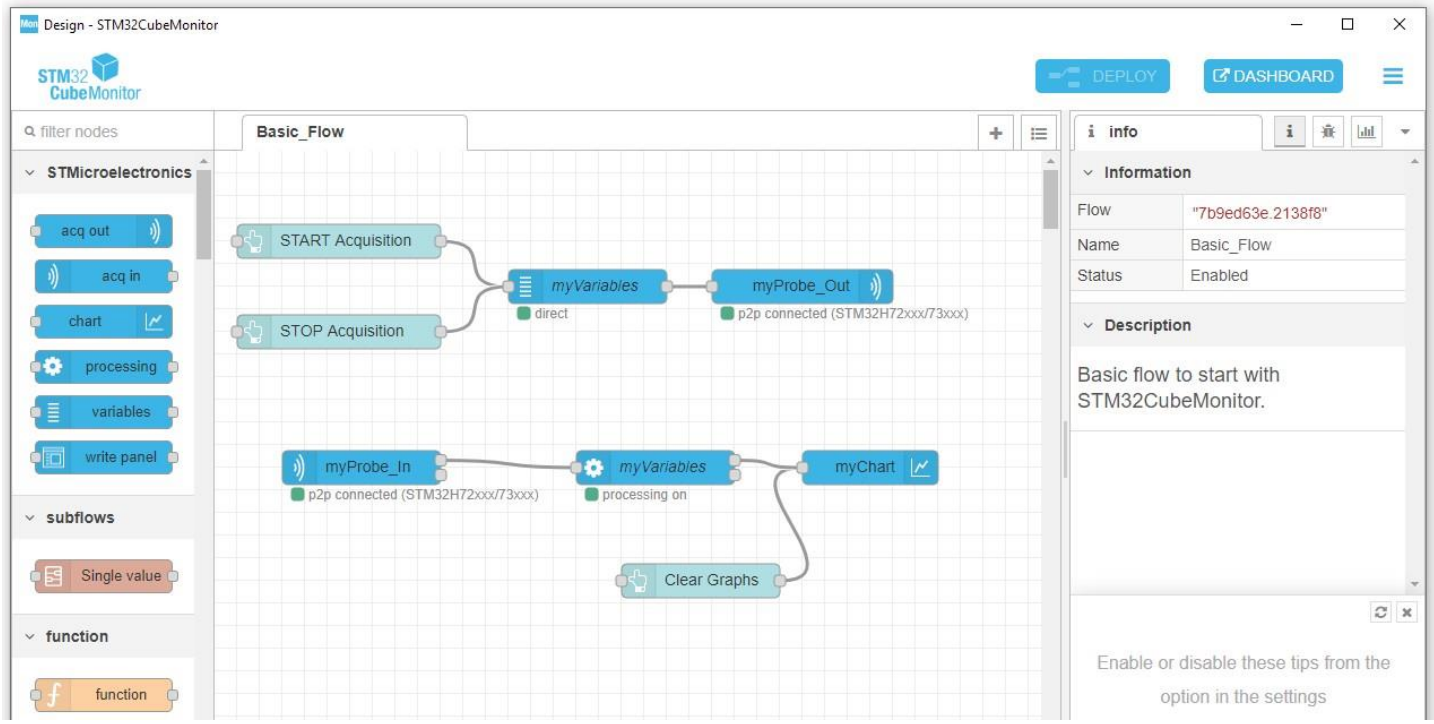
- STM32CubeMonitor performs data acquisition by reading in the target MCU memory though ST-Link and SWD/JTAG connection.
- The elements involved are :
  - STM32 JTAG/SWD access port
    used to connect the MCU debug block to ST-Link.
  - ST-Link device.
  - STM32CubeMonitor software.

  - **All the MCU address** space is accessible.

Let's start with STM32CubeMonitor

The design mode

- Let’s start from this point
  - Please plug in your STM32 board to your PC
  - Launch STM32CubeMonitor
- STM32CubeMonitor starts in design mode
  - The basic flow is displayed

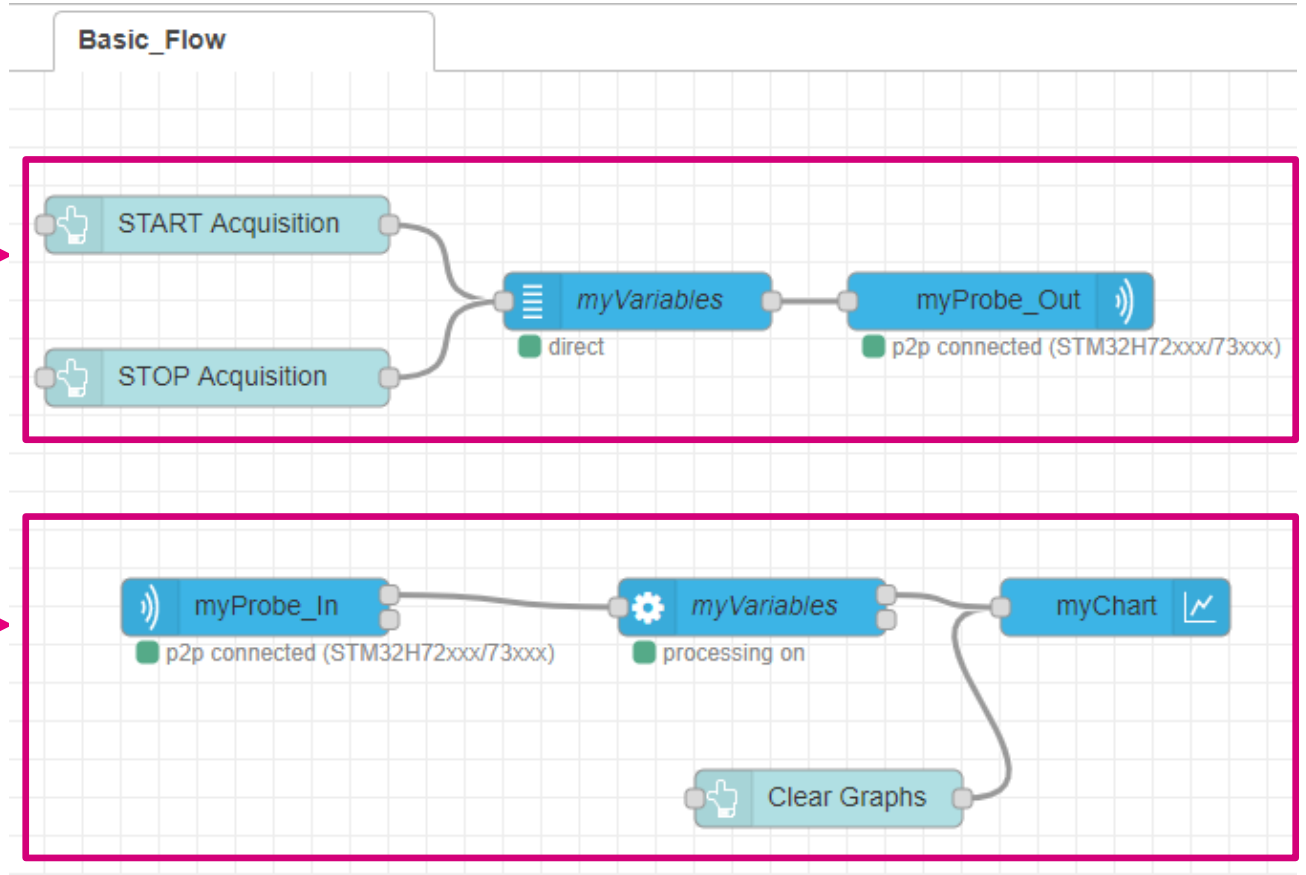


Flow structure

- One TX path from PC to probe
  - START Acquisition

    Messages are sent on the ST-Link:
    - "Read sequencially myVariables“
  - STOP Acquisition

    No more messages are sent on the ST-Link
- One RX path from probe to PC
  - MyVariables are received and displayed on the chart.

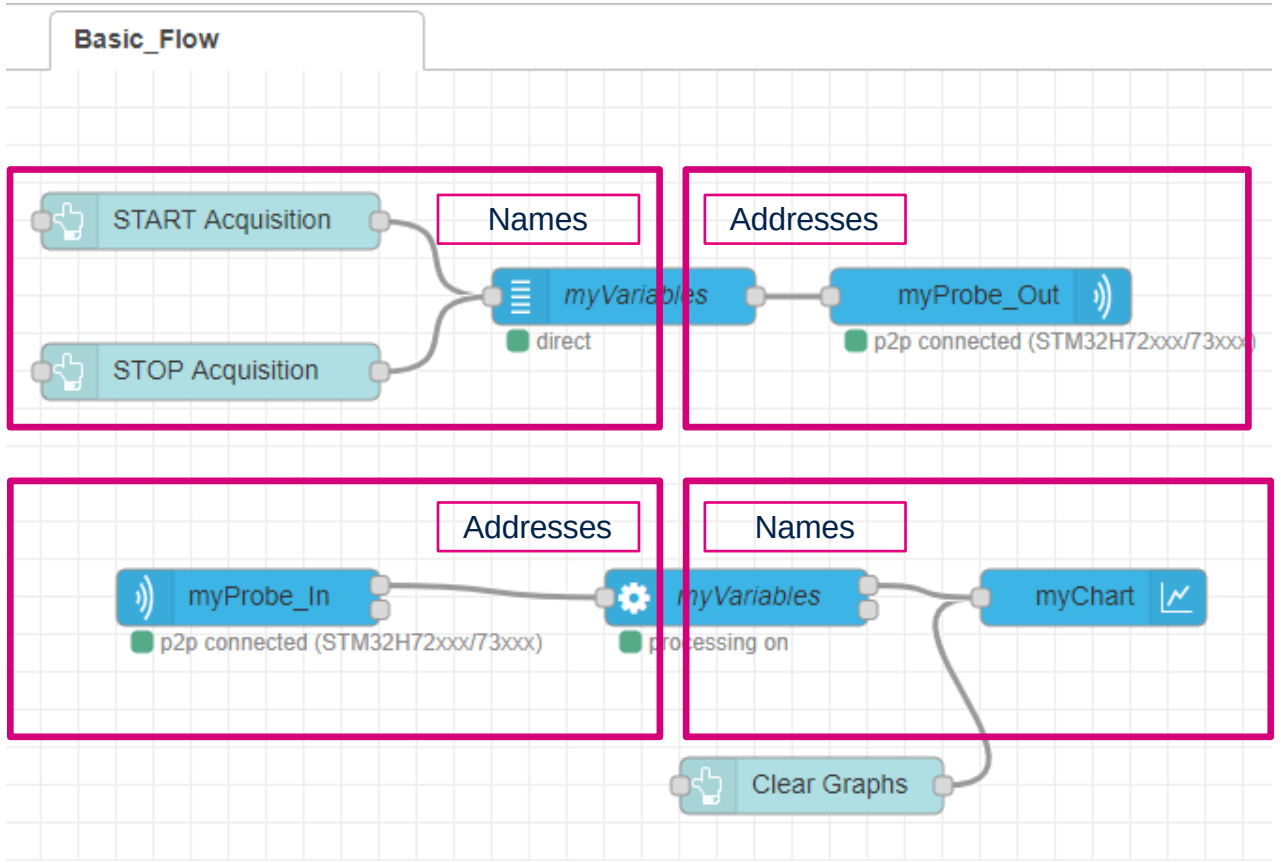


Name and address

- The MCU and its MCU debug block use addresses

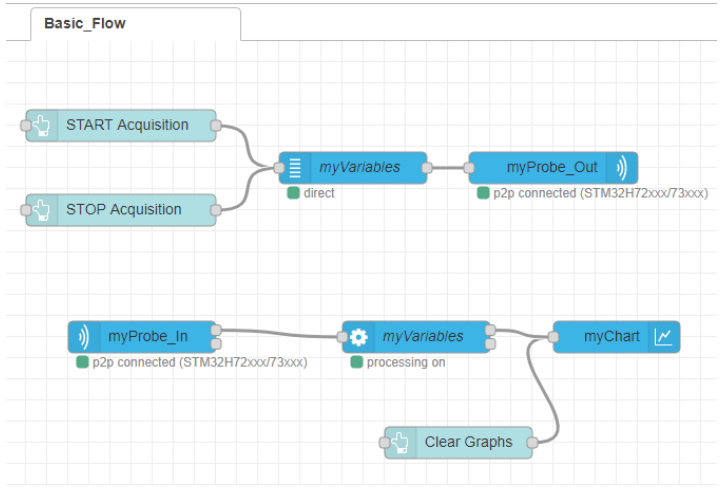
- Humans prefer names
- PC software also!

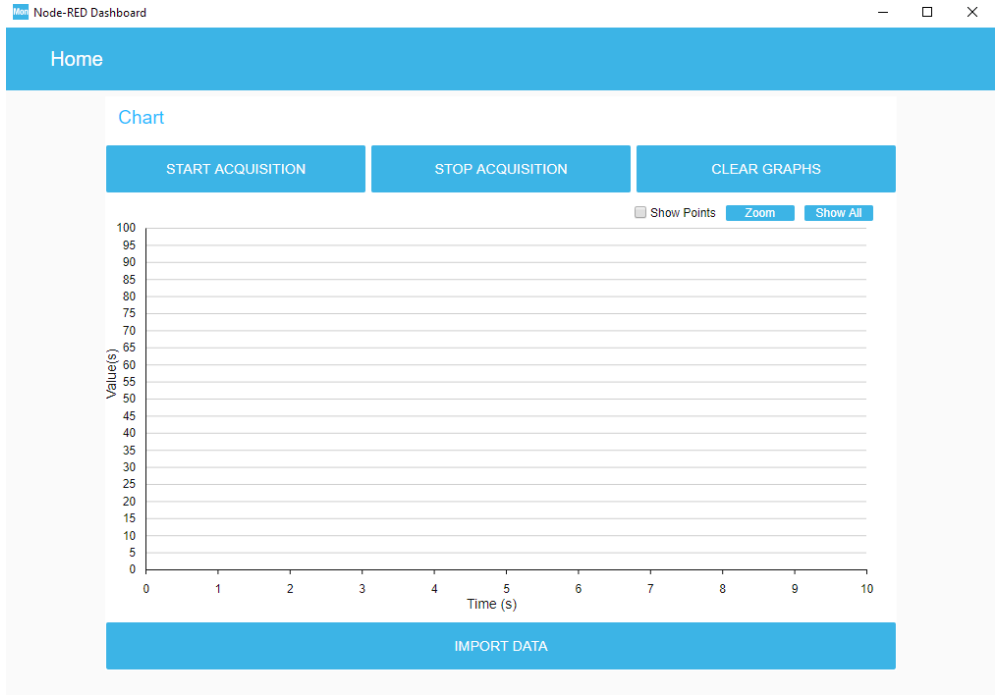
  - The nodes “myVariables” manages the translation



The dashboard mode

- The widgets seen on the flow are displayed on the dashboard.
  - 3 Buttons
    - START, STOP, CLEAR
  - Chart





Read variables from MCU RAM memory

life.augmented

Variable name

- Let’s access to some variable of the firmware
  - The code defines the type of “basicCounter”
    - Unsigned 16-bit

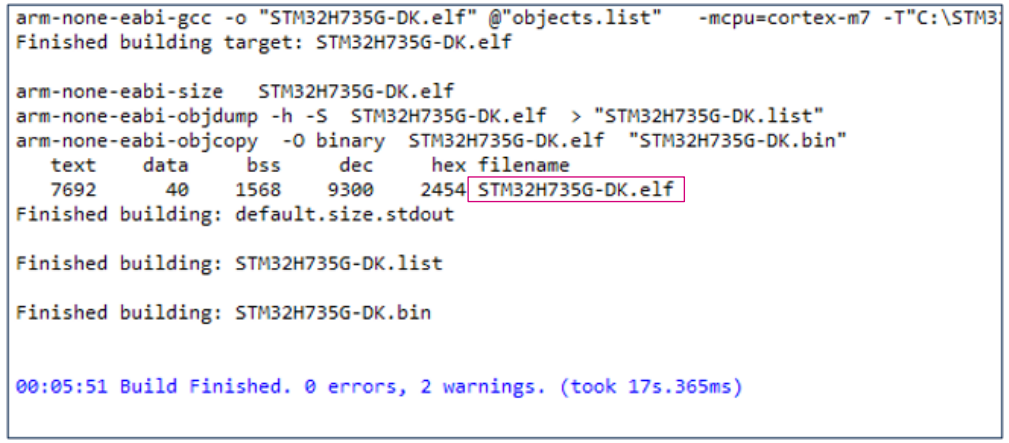
```
70
71  uint16_t basicCounter = 0;
72
```

  - The code increment the “basicCounter” every 500ms

```
134
135   /* Infinite loop */
136   /* USER CODE BEGIN WHILE */
137   while (1)
138   {
139     /* USER CODE END WHILE */
140
141     /* USER CODE BEGIN 3 */
142     HAL_Delay(500);
143     HAL_GPIO_TogglePin(GPIOC, USER_LED1_Pin);
144     basicCounter += 1;
145
```

Get the names from elf

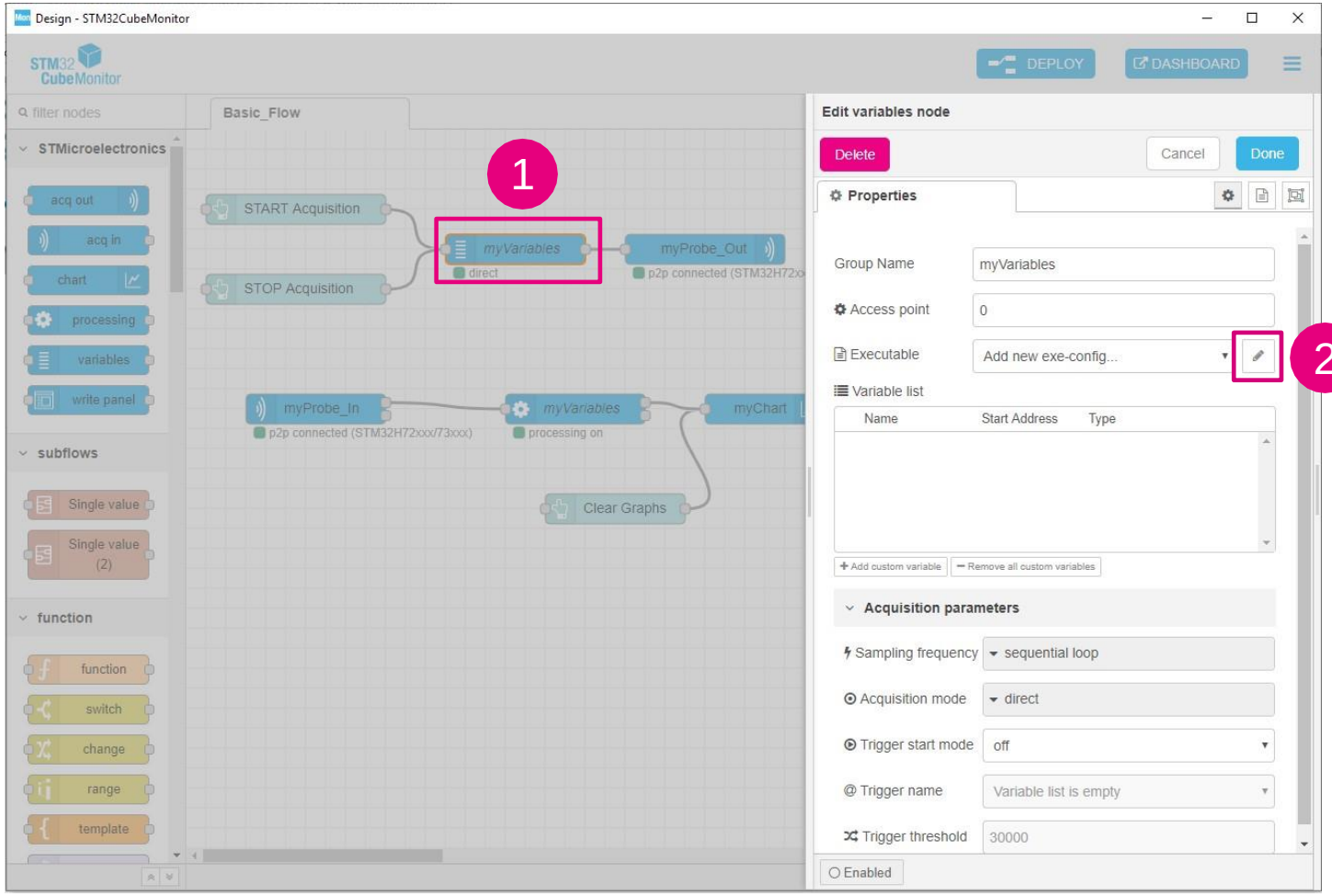
- The variable names are in the elf executable file
  - generated during the project build in STM32CubeIDE
  - STM32H735-DK.elf



  - .out and .axf extensions also supported

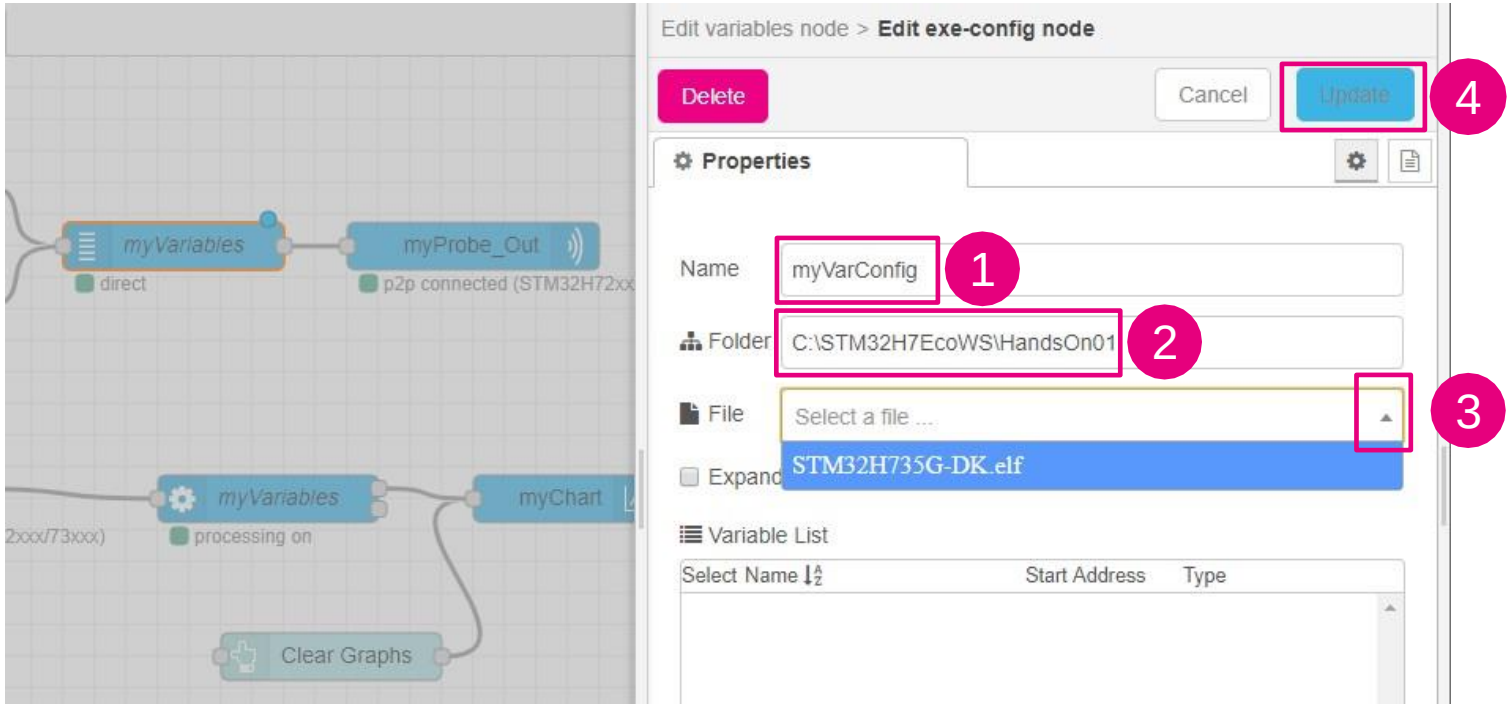
Get the names from elf

- Add new exe-config...
  1. Double-click on “myVariables”
     - The variables node properties are displayed
  2. Click on the pen of “Executable"



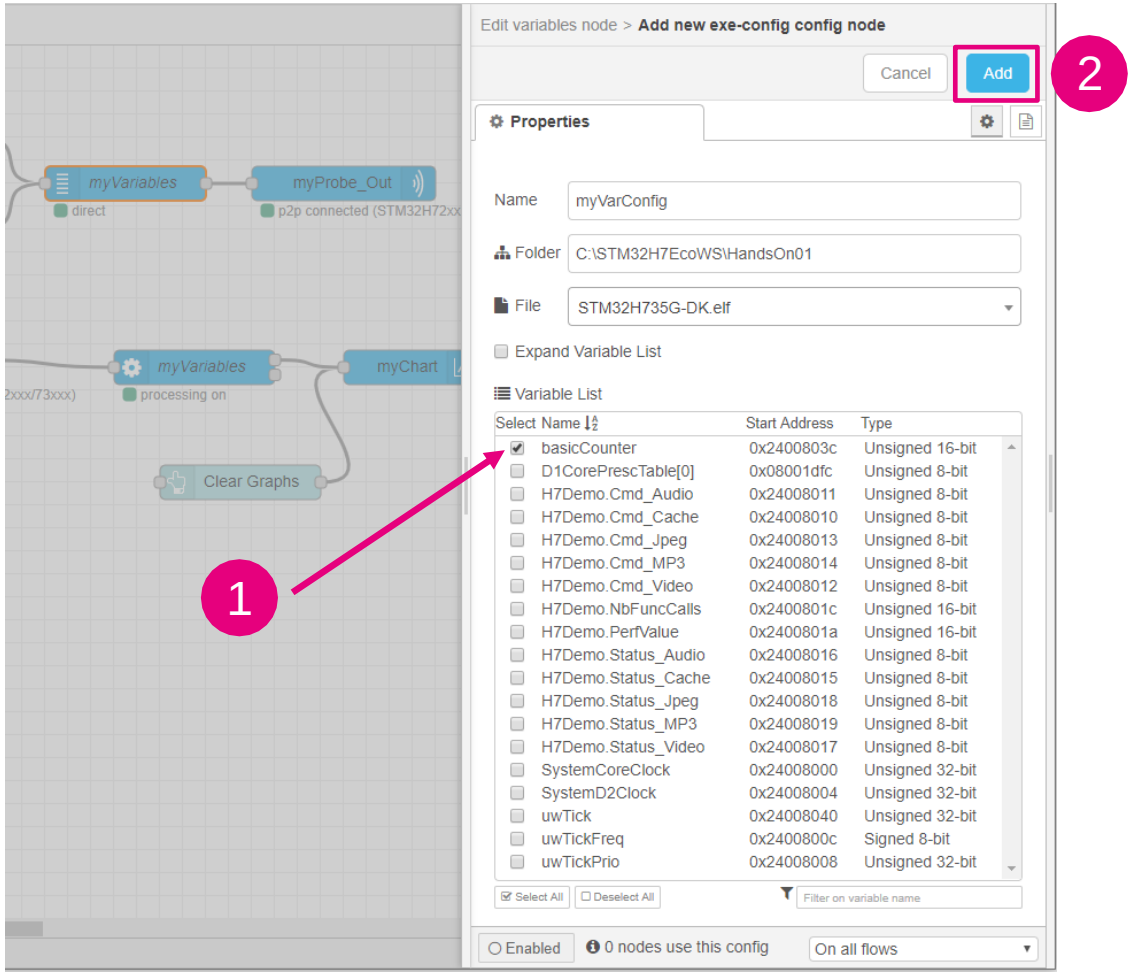
Get the names from elf

- Create the exe-config
  1. Give a name to the configuration
  2. Fill “Folder” with the folder containing the elf file
     - C:\STM32H7EcoWS\HandsOn_01
  3. Select the elf file from the drop-down list
     - STM32H735G-DK.elf
  4. Click in Update



Get the names from elf

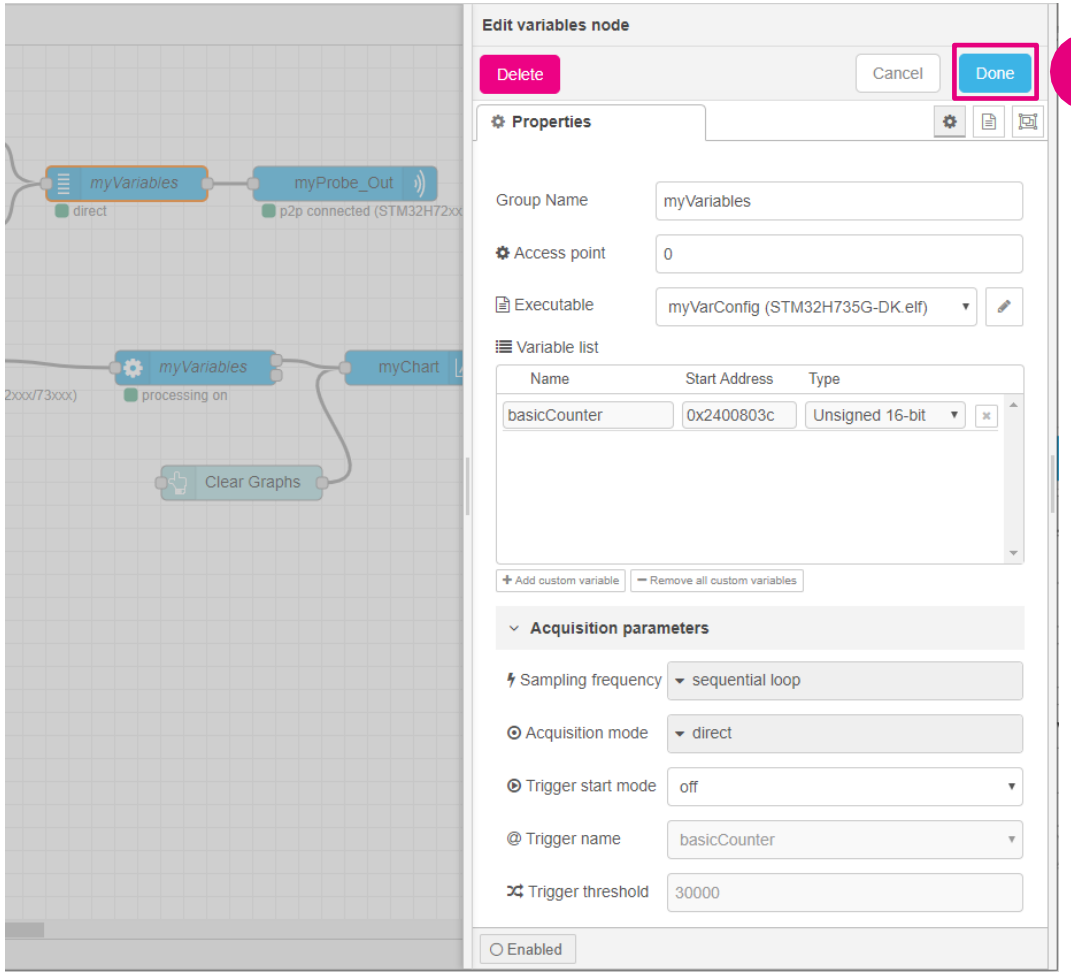
- Select the variables in the “Variable List”
  1. Select “basicCounter”
  2. Click on “Add”
     - Adds the configuration to the “myVariables”



Get the names from elf

- The variable list is configured
  - Name
    - basicCounter
  - Address
    - 0x2400803C
  - Type
    - Unsigned 16-bit

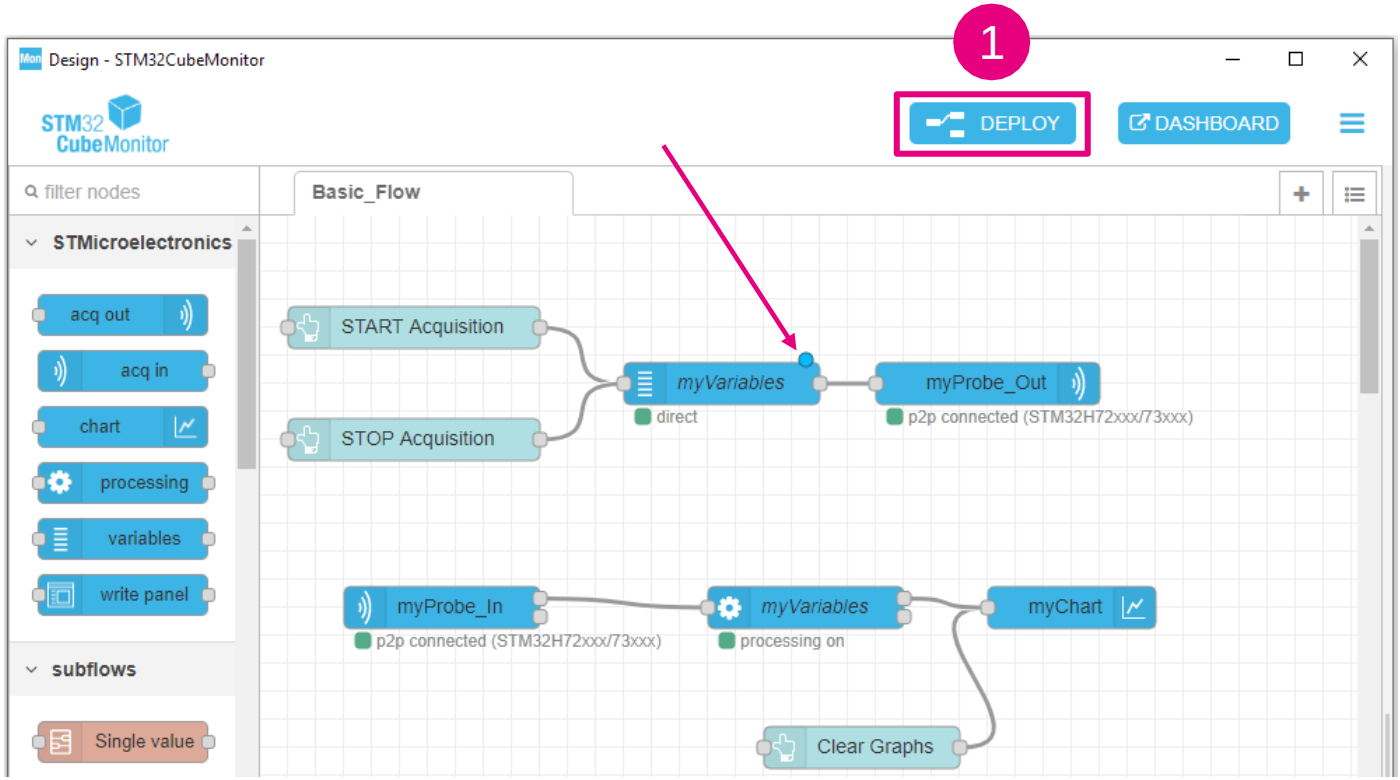
1. Click on “Done”



Get the names from elf

- Save the flow
  - A blue point on “myVariables” shows that the configuration has been modified.

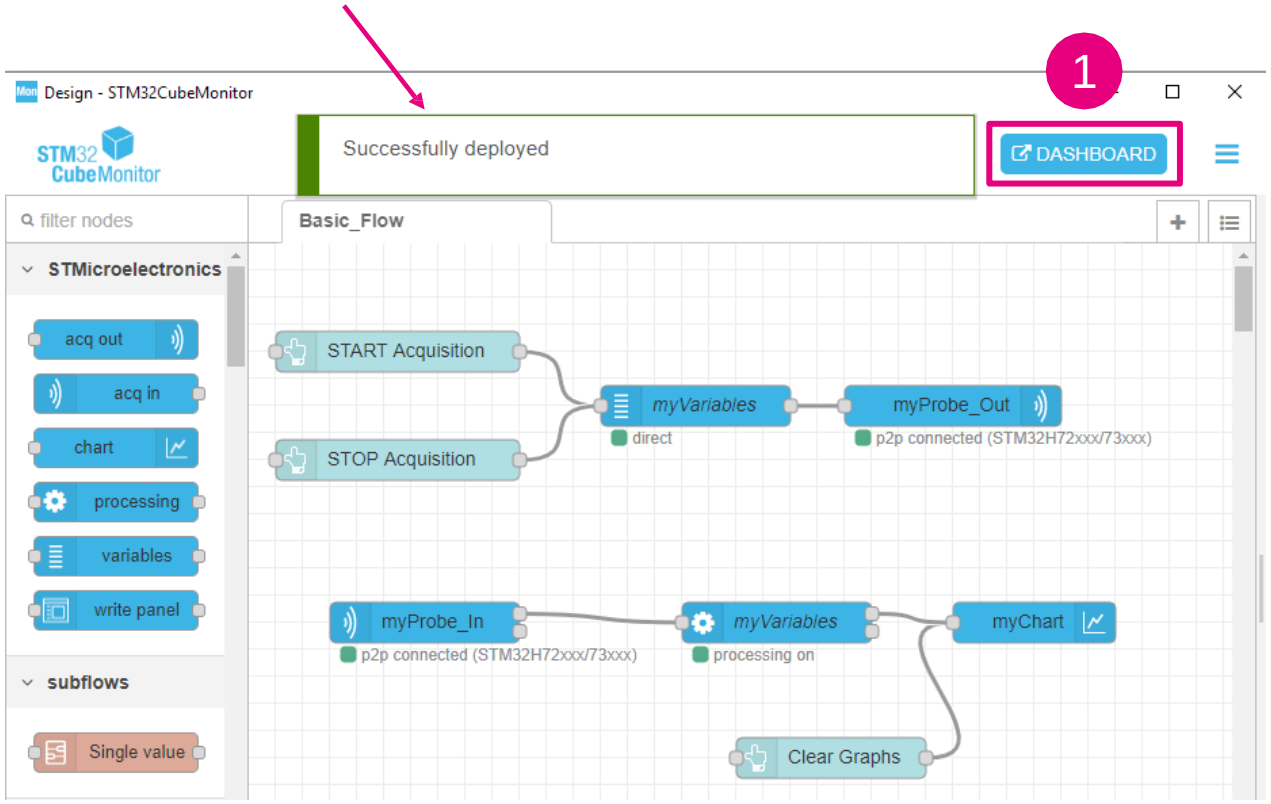
1. Click on “Deploy”



Monitor and visualize read in MCU RAM memory

- Launch the dashboard
  - “Successfully deployed” is displayed

1. Click on “Dashboard”

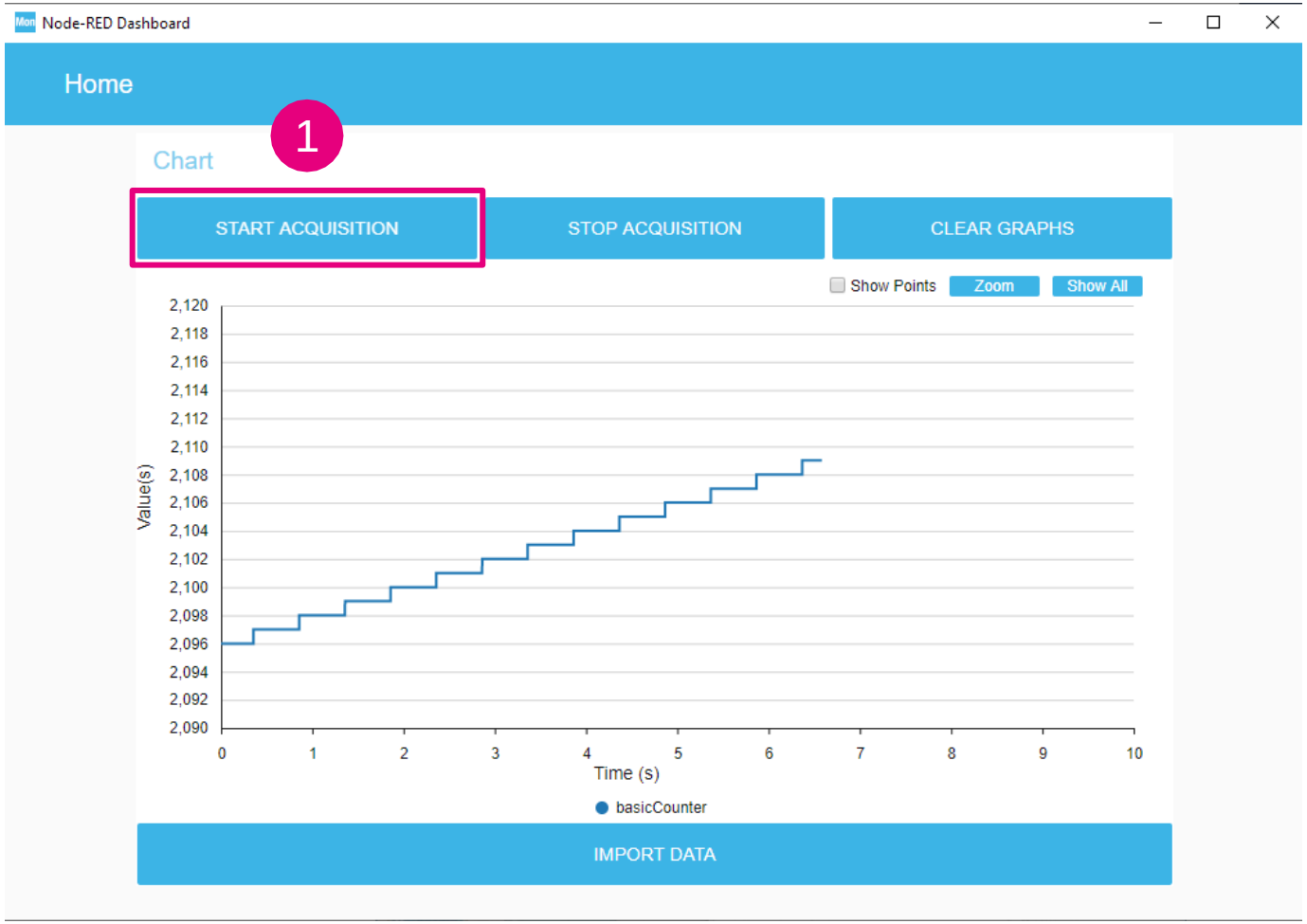


Monitor and visualize read in MCU RAM memory

- Launch the acquisition
  1. Click on “START ACQUISITION”

  ➢ “basicCounter” is sequentially read from the MCU RAM memory

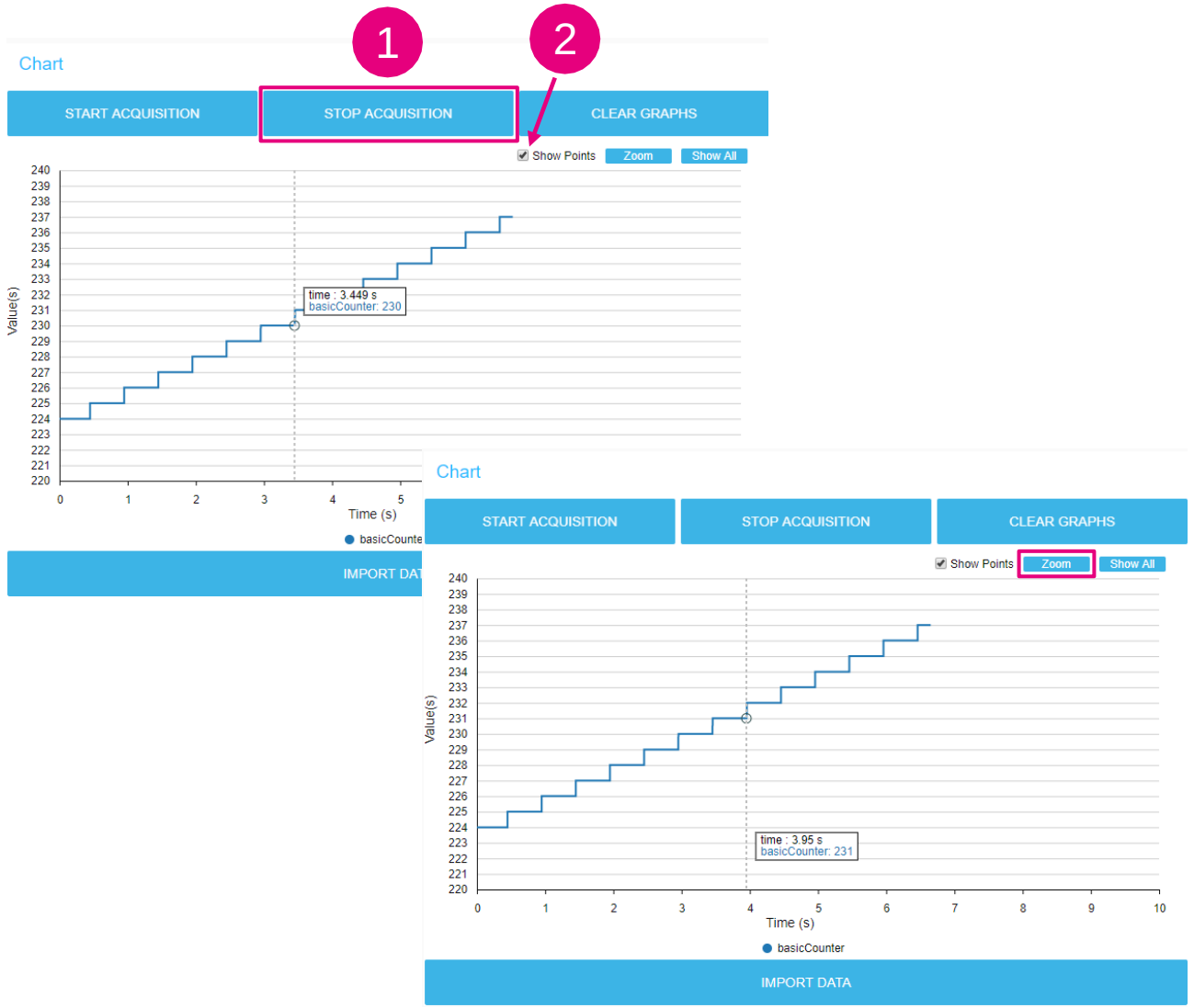
  ➢ “basicCounter” is regularly incremented



Monitor and visualize read in MCU RAM memory

- Stop reading the variable
  1. Click on “STOP ACQUISITION”
- The times and values can be visualized
  2. Click the “Show Points” checkbox
     - ‘basicCounter” is incremented by 1
     - “basicCounter” is incremented every 500ms

- Use the zoom for more precision
  - Zoom or Brush is selectable.
  - In zoom mode, it’s possible to zoom on the curve using the mouse to select a zone.

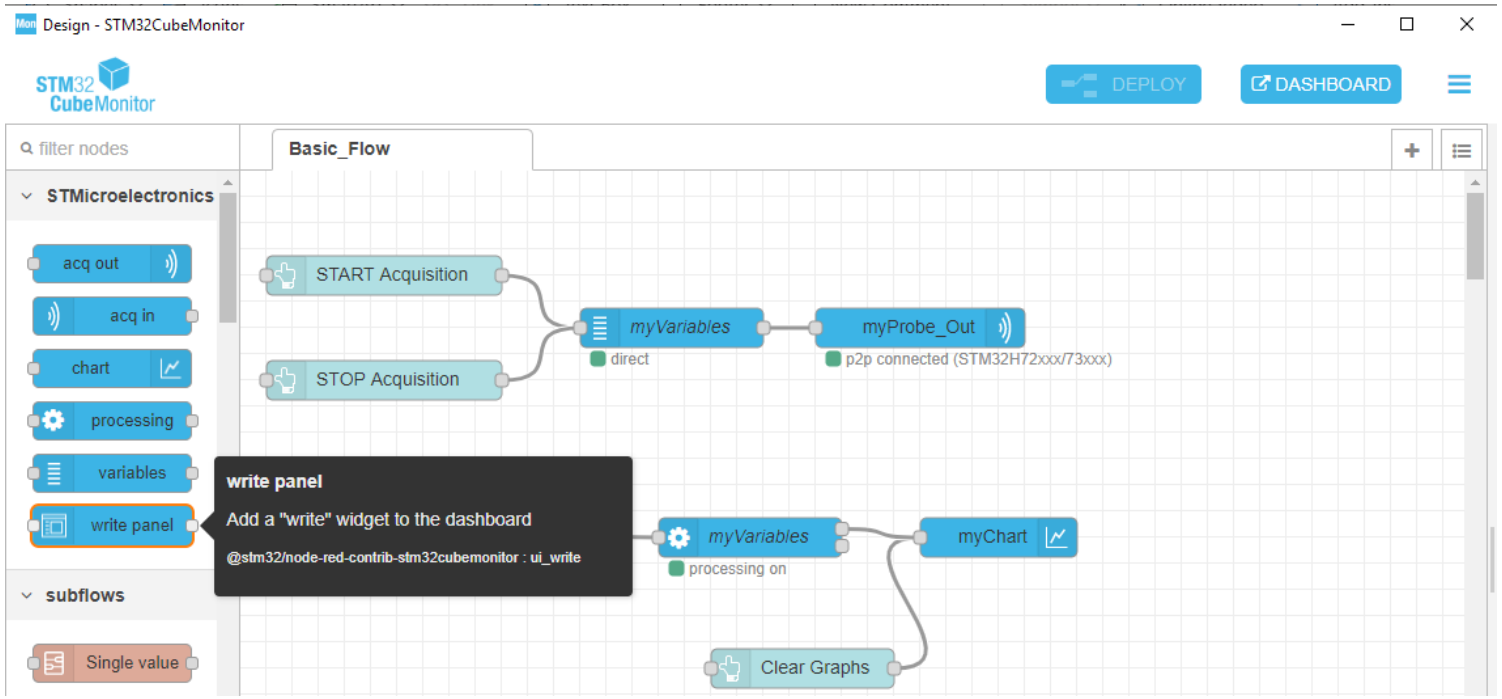


Write variables to MCU RAM memory

life.augmented

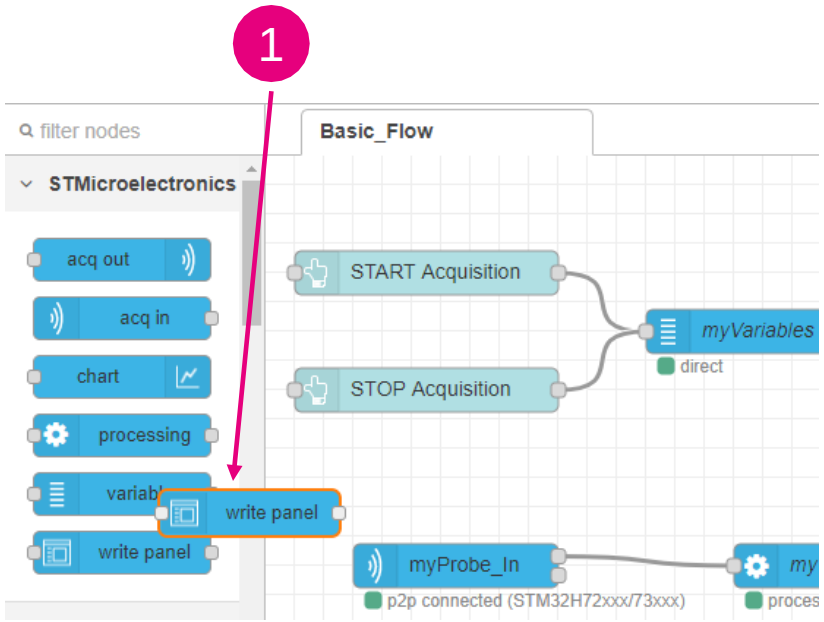
Add a widget

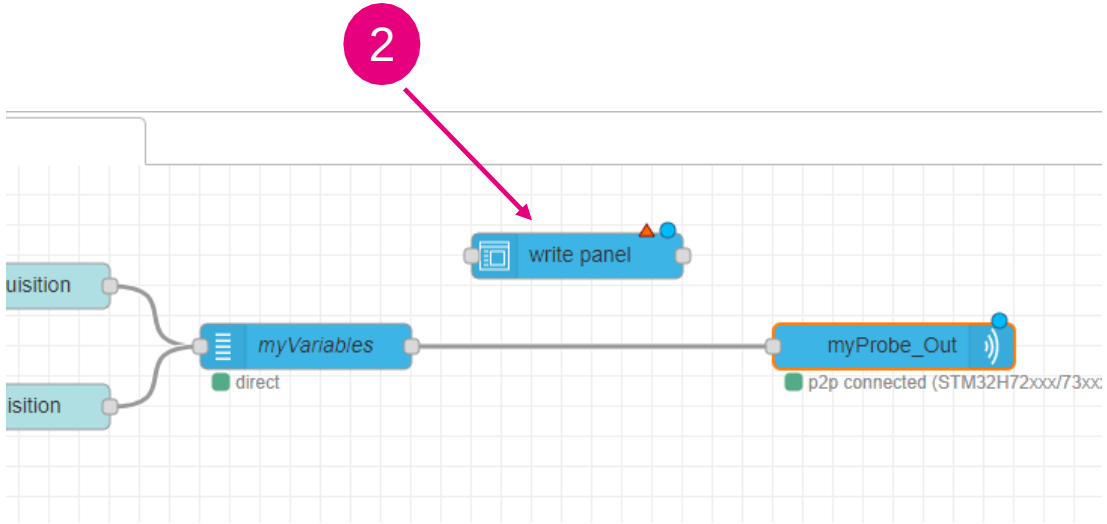
- Currently, no way to write a value to a variable.
  - ➢ The “write panel” widget in the node palette will be used



Add a widget

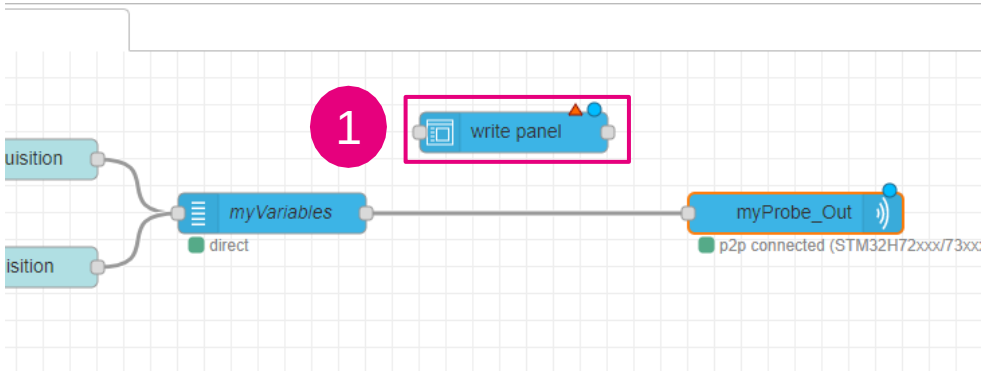
- Drag & drop the “write panel” in the flow editor
  1. Click on the “write panel” node and hold the button down
  2. Drag the “write panel” node to the flow editor
     - “myprobe_Out” can be dragged a bit further



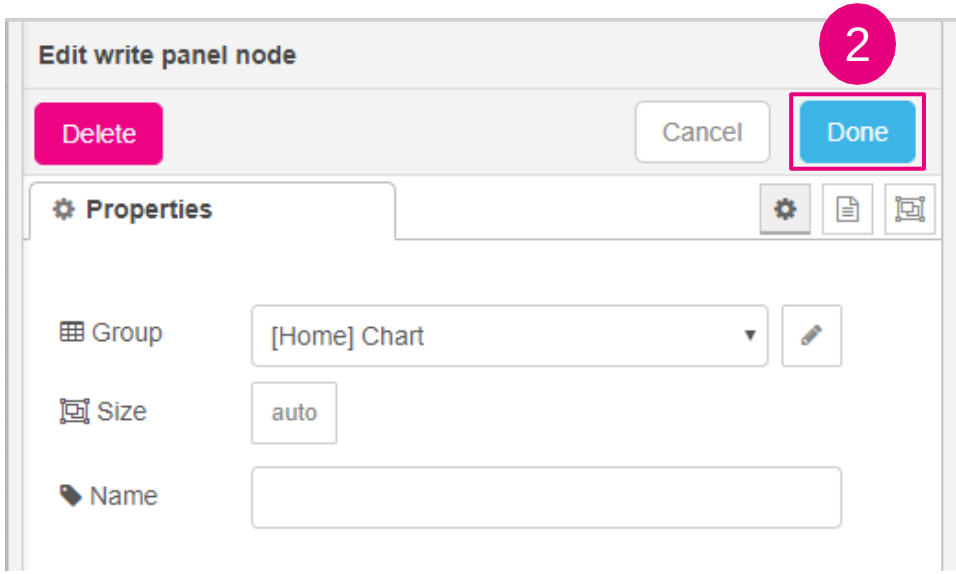


Add a widget

- Set the “write panel” properties
  - A red triangle is visible on the “write panel” node
    - Properties are not configured
    - The”write panel” widget will not be added to the dashboard!



1. Double-click on the “write panel”
   - “Group” has been set automatically as only one exists
     - UI group on the dashboard = Chart
     - Tab = Home
   - Several groups can be created on the dashboard
   - Several flows can be opened in the flow editor
     - The dashboard will then have several tabs with several UIs



2. Click on “Done”
   - The name is optional

life.augmented

Add a widget – done!

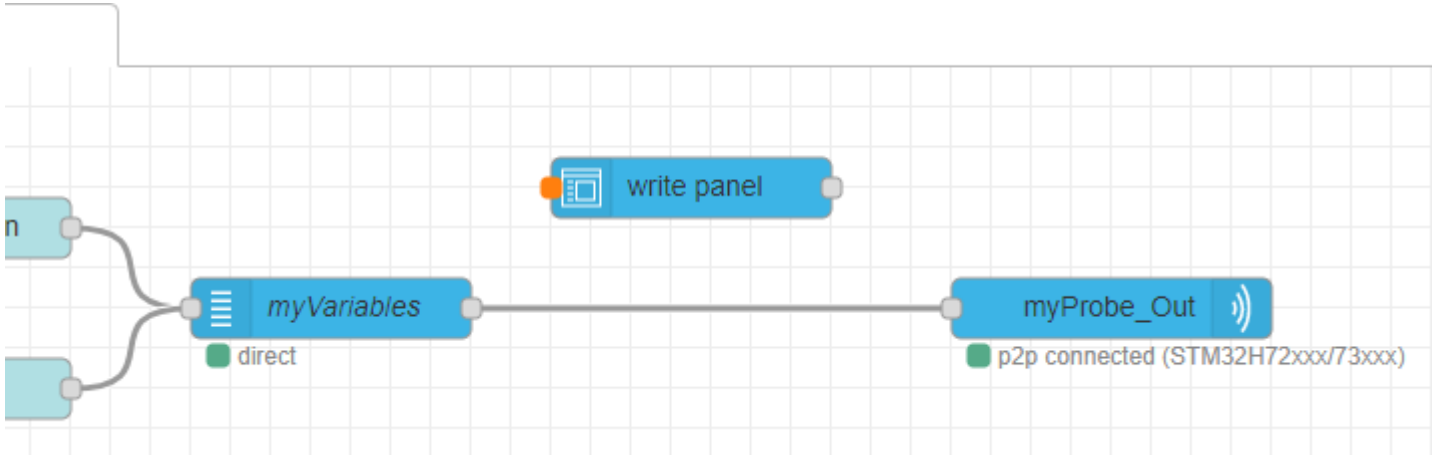
- Launch the dashboard
  1. Click on “DEPLOY”
  2. Click on “Dashboard”

- The write panel widget has been added
  - Visual programming
  - A drag & drop and some clicks

- Let’s add a variable in the panel

Add a variable

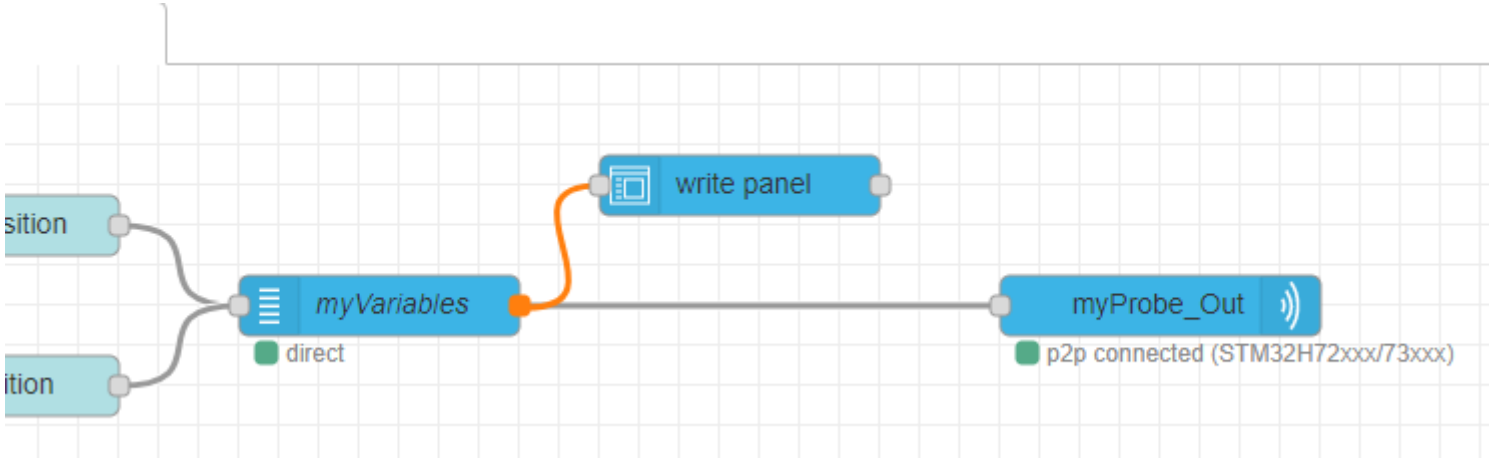
- Add a link with “myVariables” to populate the write panel
  1. Put the mouse on the input of the “write panel” node
     - The input point is displayed in orange



Add a variable

- Add a link with “myVariables” to populate the write panel
  2. Click and drag to the output of “myVariable”
     - Hold the mouse button down until the output point is displayed in orange

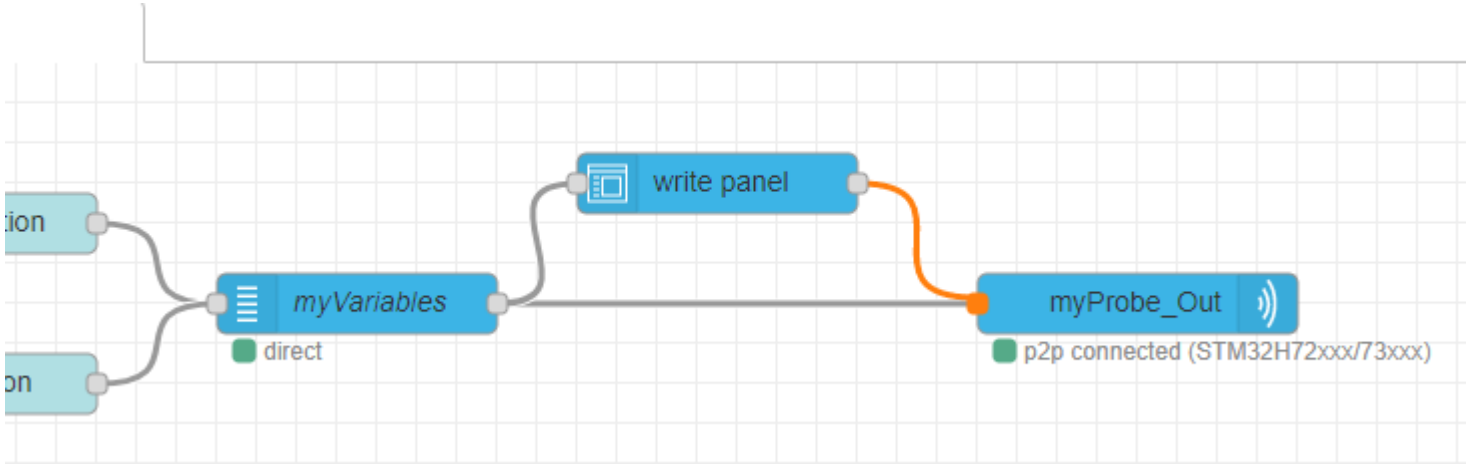
  - The link is done
  - The list of variables will appear in the write panel as soon as the “START ACQUISITION” button is clicked



Activate the “WRITE” button

- Add a link with “myProbe_Out”
  1. Drag a link between the output of the “write panel” and the input of “myProbe_Out”

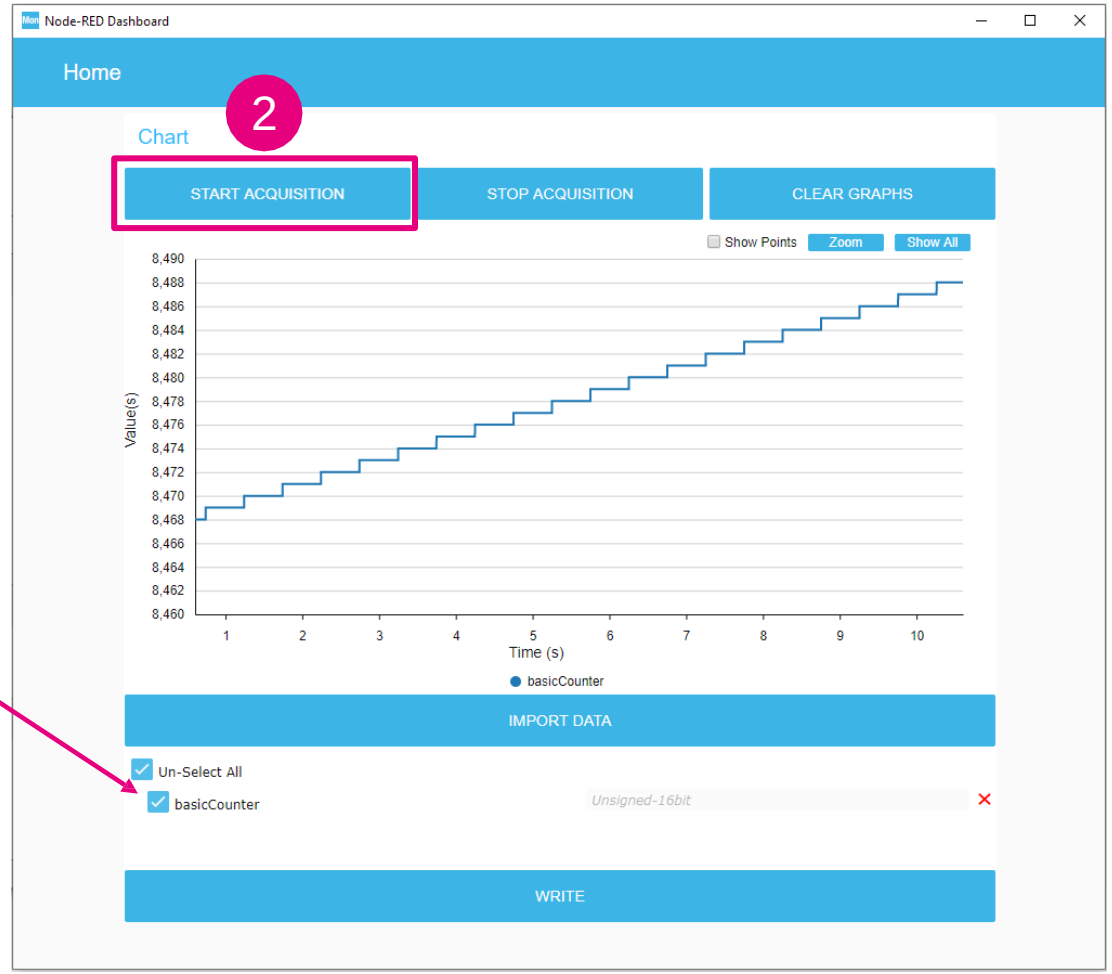
  - The link is done
  - A click on the “WRITE” button will produce a message sent on the ST-LINK:
    - “Write the selected variables“



Monitor and visualize write in MCU RAM memory

- Launch the acquisition
  1. Click on “DEPLOY”, then “DASHBOARD”
  2. Click on “START ACQUISITION”

➢ “basicCounter” is in the write panel list



Monitor and visualize write in MCU RAM memory

- Write to the basicCounter variable
  1. Write a value in the text box beside “basicCounter” name
  2. Click on “WRITE”

➢ “basicCounter” value is changed to the written number.

Read or Write a peripheral register

Read/write to an address

- **All the MCU address** space is accessible.

- But some addresses don’t have a variable name in the elf file.
  - Registers

- How to read/write an address without a name?
  - Let’s take the example of an I/O port

From RM0468

- The GPIOC base address is 0x58020800
- The ODR register has the offset 0x14
  - The lowest byte of ODR register is at 0x58020814

11.4.6 GPIO port output data register (GPIOx_ODR) (x = A to H, J, K)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

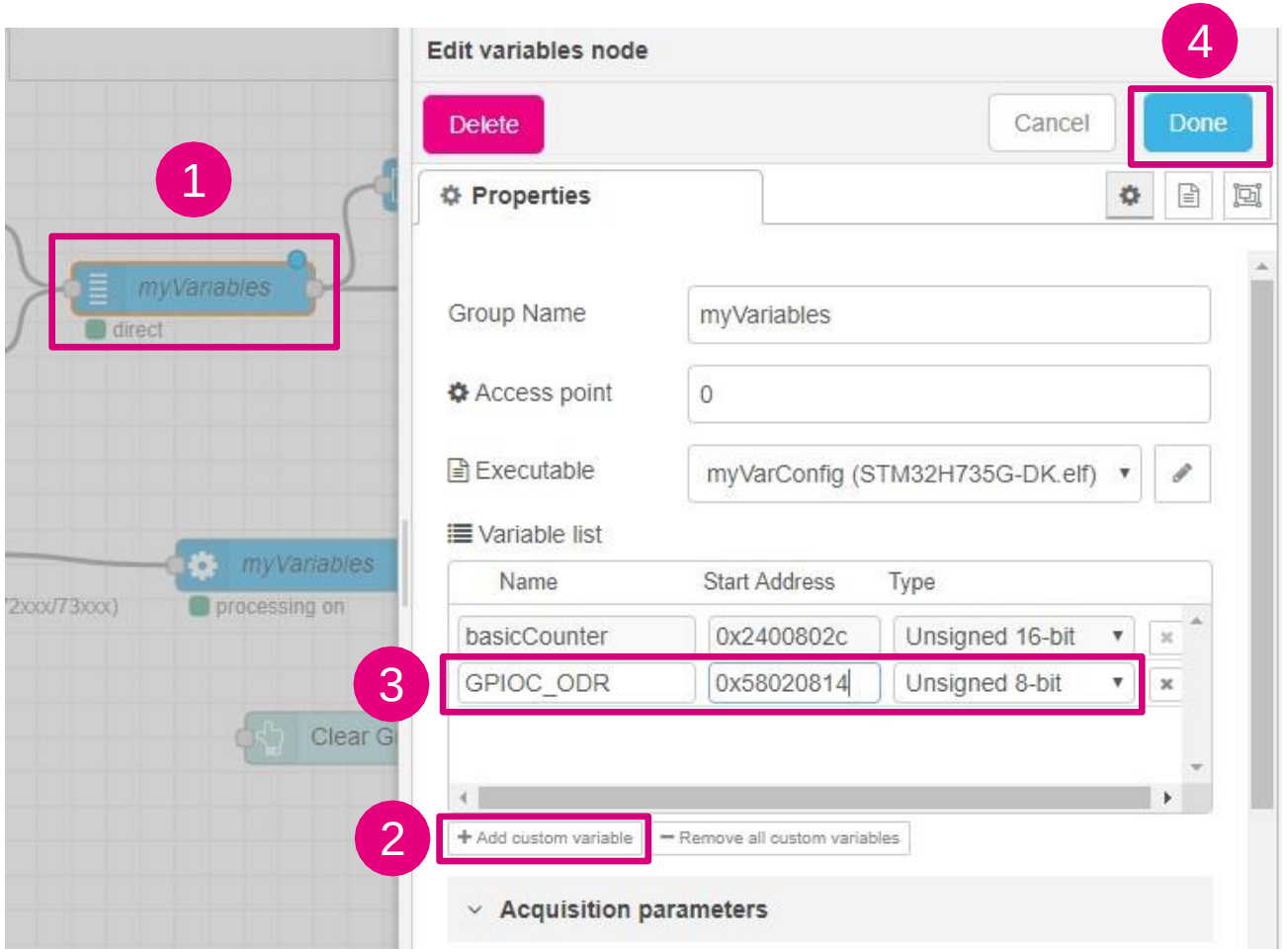
Table 7. Register boundary addresses[1]

Boundary address	Peripheral	Bus	Register map
0x58026400 - 0x580267FF	HSEM	AHB4 (D3)	*Section 10.4: HSEM registers*
0x58026000 - 0x580263FF	ADC3		*Section 28.7: ADC common registers*
0x58025800 - 0x58025BFF	DMAMUX2		*Section 17.6: DMAMUX registers*
0x58025400 - 0x580257FF	BDMA		*Section 16.6: BDMA registers*
0x58024C00 - 0x58024FFF	CRC		*Section 21.4: CRC registers*
0x58024800 - 0x58024BFF	PWR		*Section 6.8: PWR registers*
0x58024400 - 0x580247FF	RCC		*Section 8.7: RCC registers*
0x58022800 - 0x58022BFF	GPIOK		*Section 11.4: GPIO registers*
0x58022400 - 0x580227FF	GPIOJ		*Section 11.4: GPIO registers*
0x58021C00 - 0x58021FFF	GPIOH		*Section 11.4: GPIO registers*
0x58021800 - 0x58021BFF	GPIOG		*Section 11.4: GPIO registers*
0x58021400 - 0x580217FF	GPIOF		*Section 11.4: GPIO registers*
0x58021000 - 0x580213FF	GPIOE		*Section 11.4: GPIO registers*
0x58020C00 - 0x58020FFF	GPIOD		*Section 11.4: GPIO registers*
0x58020800 - 0x58020BFF	GPIOC		*Section 11.4: GPIO registers*
0x58020400 - 0x580207FF	GPIOB		*Section 11.4: GPIO registers*
0x58020000 - 0x580203FF	GPIOA		*Section 11.4: GPIO registers*

Add the address

- Add the address in the variable list
  1. Open “myVariables” properties
  2. Click on “Add custom variable”
  3. Fill the fields
     - Name
       - GPIOC_ODR
     - Address
       - 0x58020814
     - Type
       - Unsigned 8-bit
  4. Click on “Done”

- The name is mandatory
  - For humans
  - For STM32CubeMonitor software

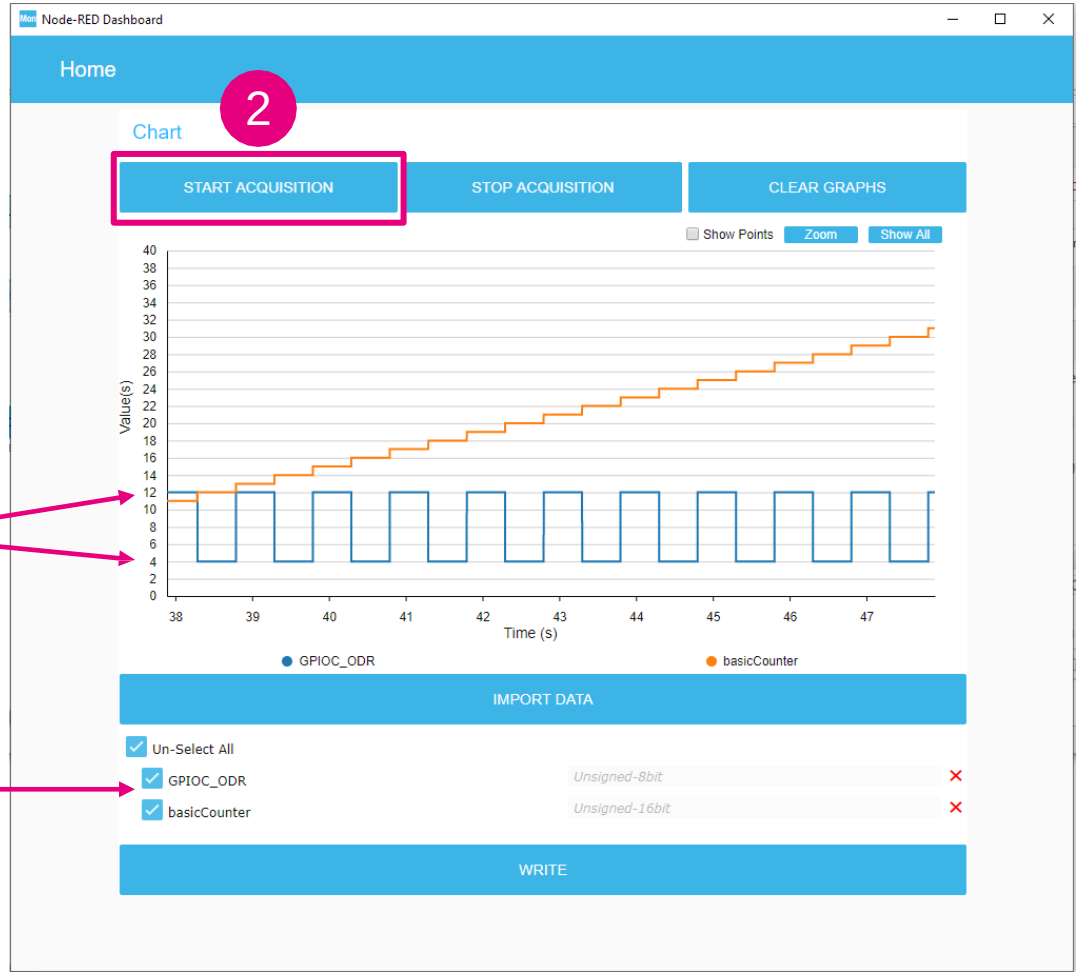


Monitor and visualize
Read in MCU register fixed address

- Launch the acquisition
  1. Click on “DEPLOY”, then “DASHBOARD”
  2. Click on “START ACQUISITION”

  ➢ “basicCounter” is displayed

  ➢ The GPIOC_ODR is also displayed



We have seen...

- How to **configure** the flow
  - How to connect to the STM32 MCU board through ST-Link
  - How to get the variable names from the executable file
- How to **access** to the STM32 memory
  - Read / Write using the variable name
  - Read / Write using the address
    - All the STM32 address space is accessible
- How to add a **widget** to the dashboard
  - Drag & drop on the flow editor
  - Insert in the flow
    - Make the connection with previous and next node

STMH72x/3x Demo flow

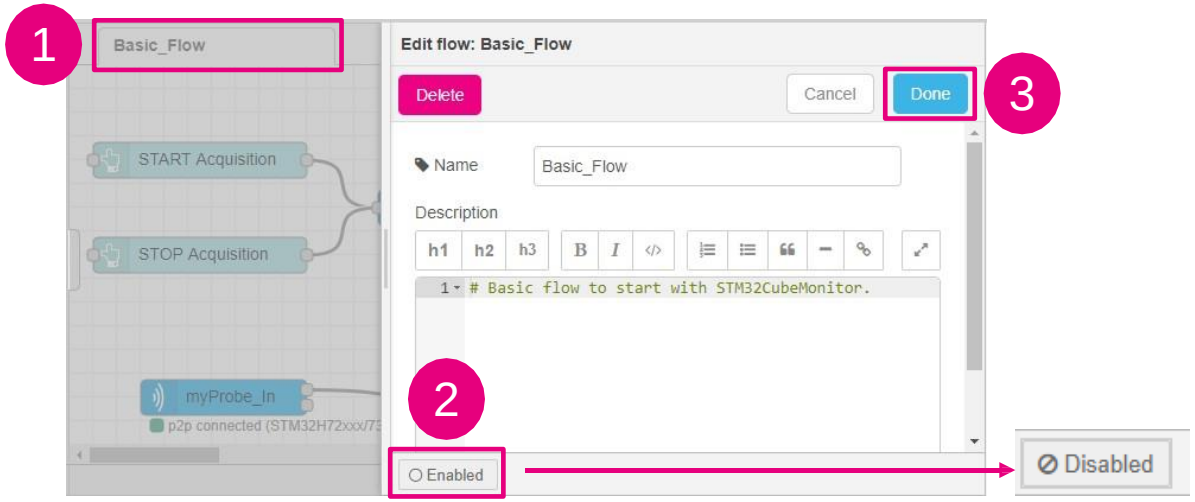
life.augmented

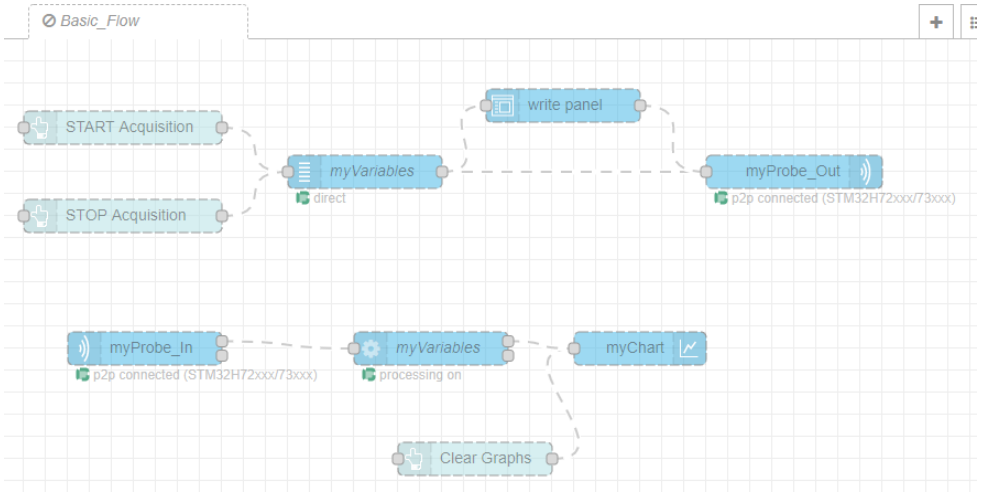
STMH72X/3X Demo flow

- An example of customization
  - More widgets
  - But same concepts
- Let’s import the flow

Disable a flow

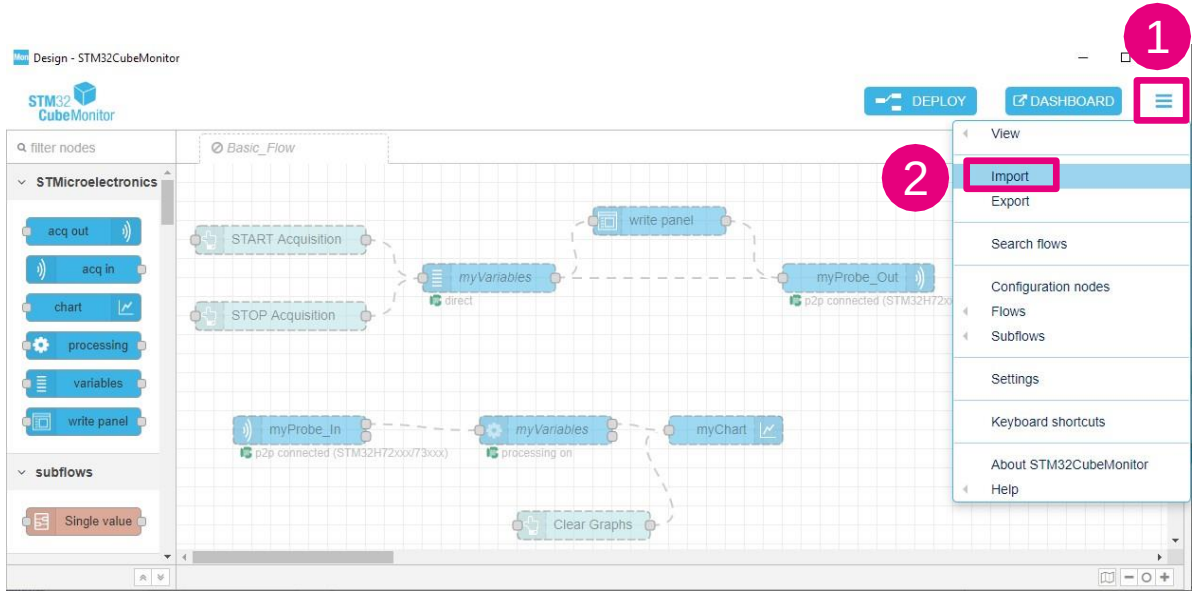
- Disable the “Basic_Flow”
  1. Double-click on the “Basic_Flow” tab
  2. Click in Enabled
     - ➢ It is changed to disabled
  3. Click on Done
     - ➢ The “Basic_Flow” tab remains in the flow editor but no “Home” tab will be accessible in the dashboard.





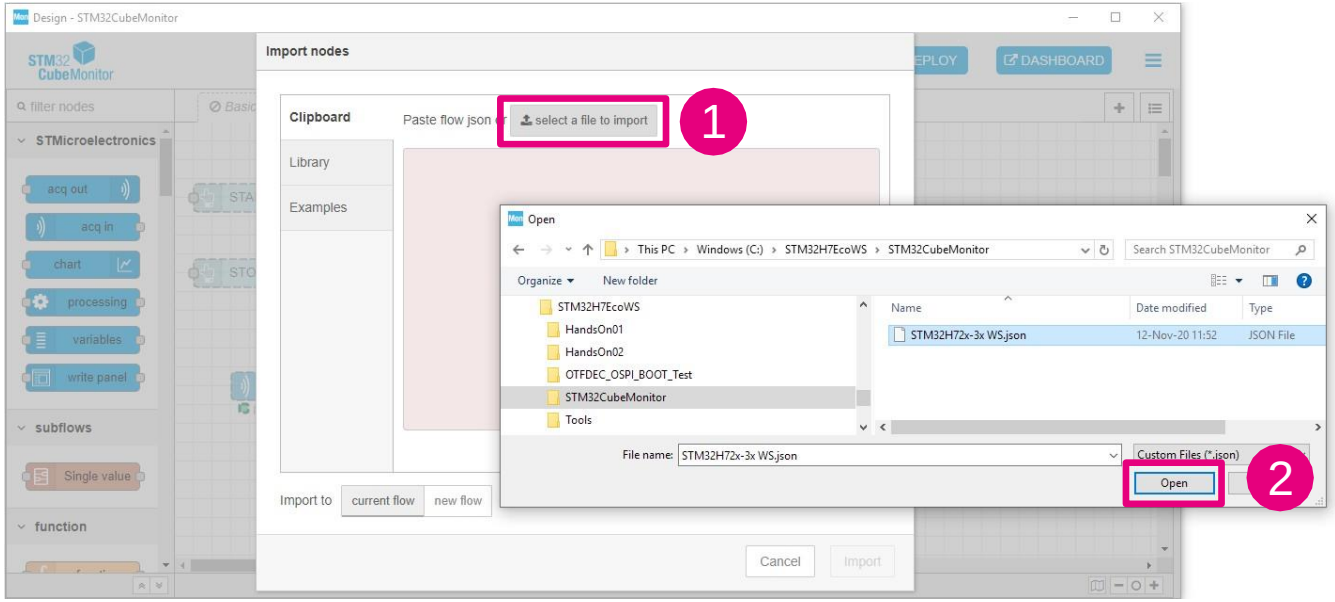
Import a flow

- **Please click on “DEPLOY”**
  - If not closed and acquisition not stopped, the dashboard is disabled and the ST-Link is relaxed.
- Open the menu and select import
  1. Click on the menu
  2. Click on “Import”



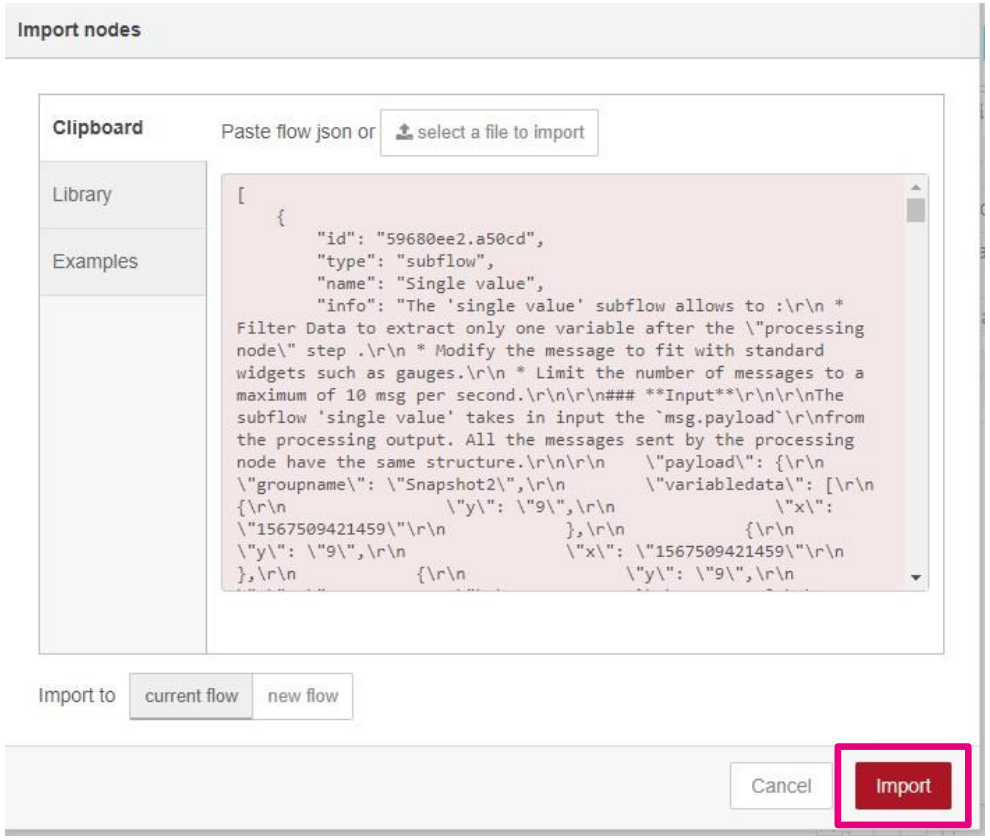
Import a flow

- Import from a file
  1. Click on “select a file to import”
  2. Open the STMH72x/3X Demo flow
     - ➢ In C:\STM32H7EcoWS\STM32CubeMonitor
     - ➢ STM32H72x-3x WS.json



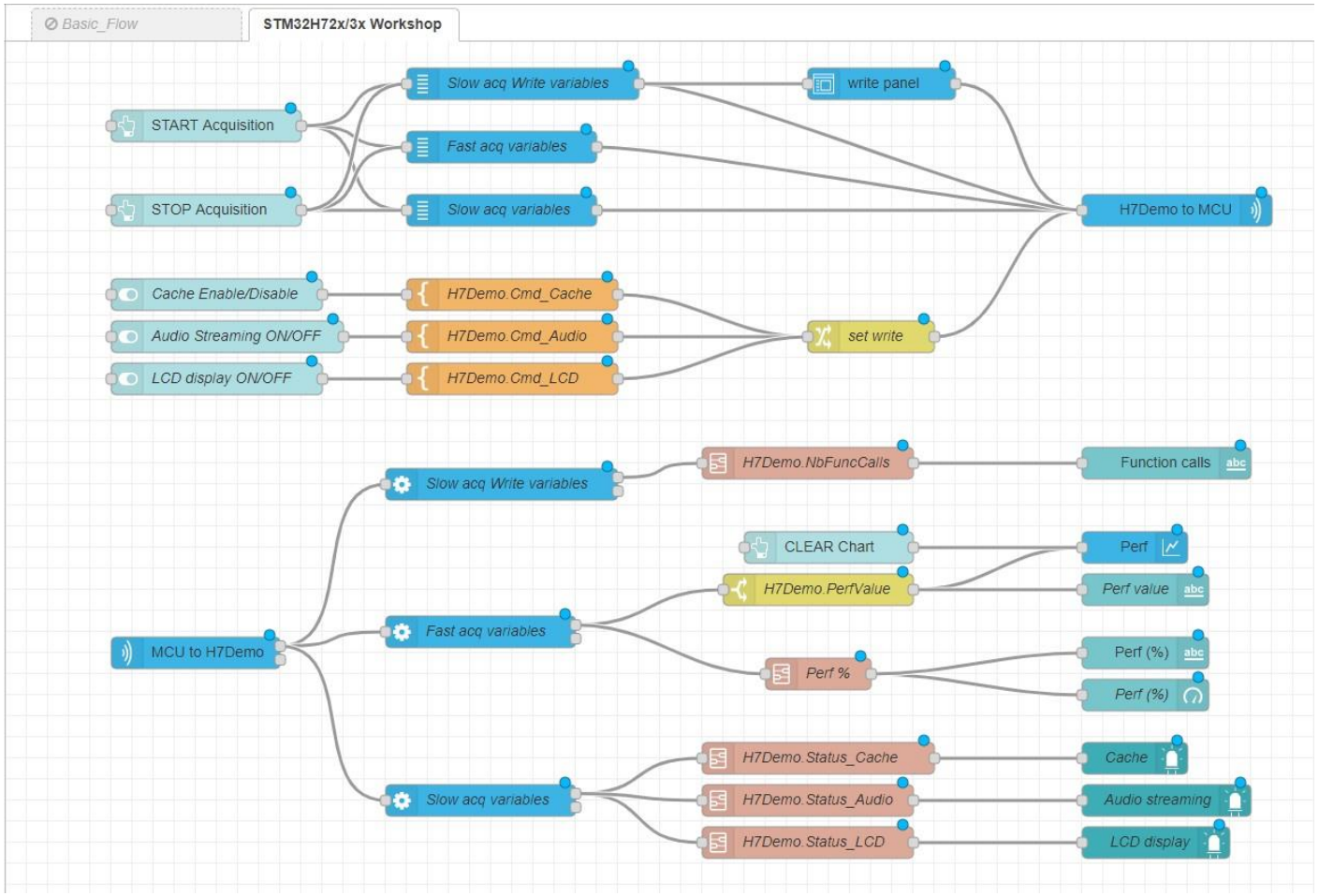
Import a flow

- Import from a file
  1. Click on “Import”



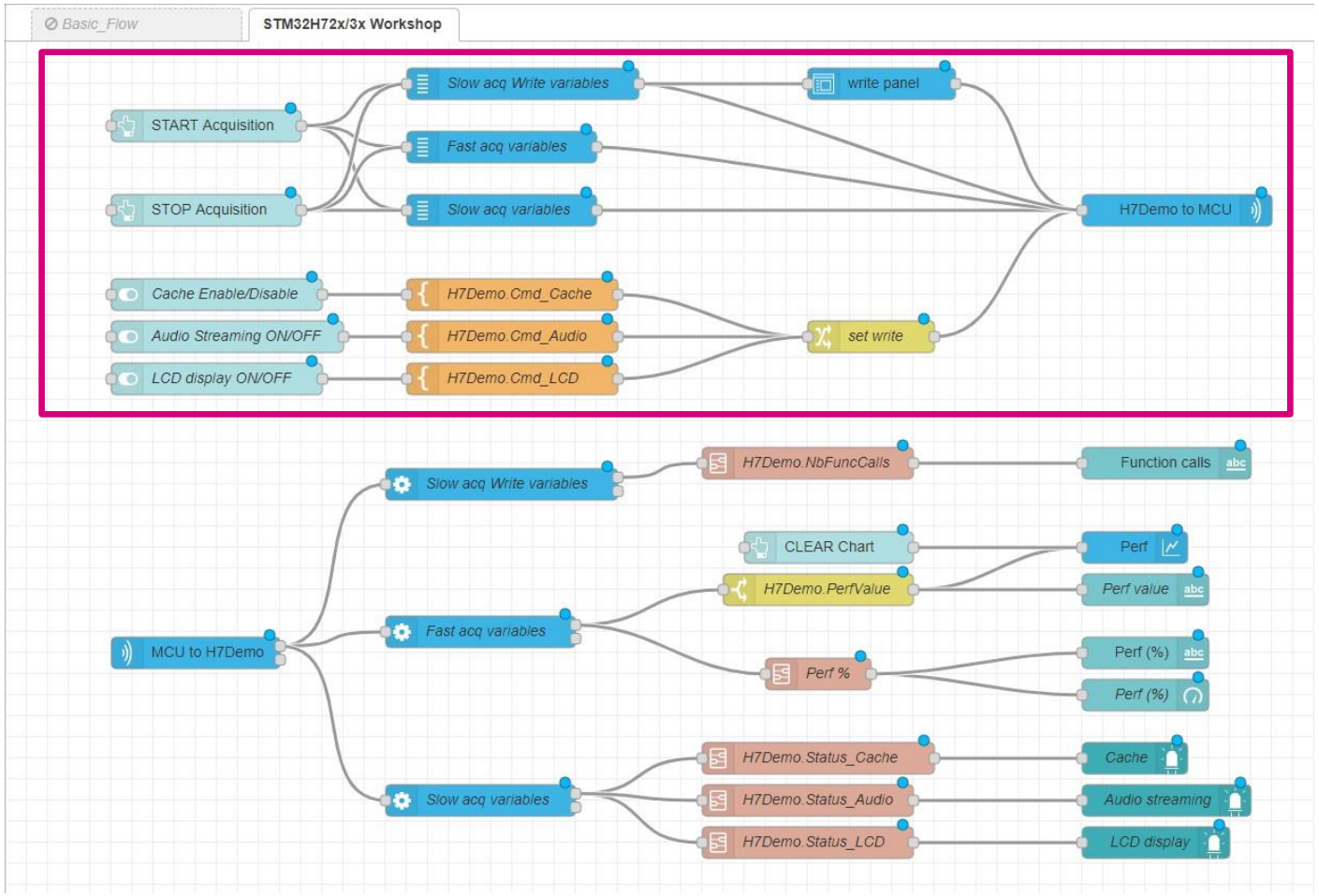
Import a flow

- The STMH72x/3X Demo flow is now visible in the flow editor
- More nodes, more links, but…



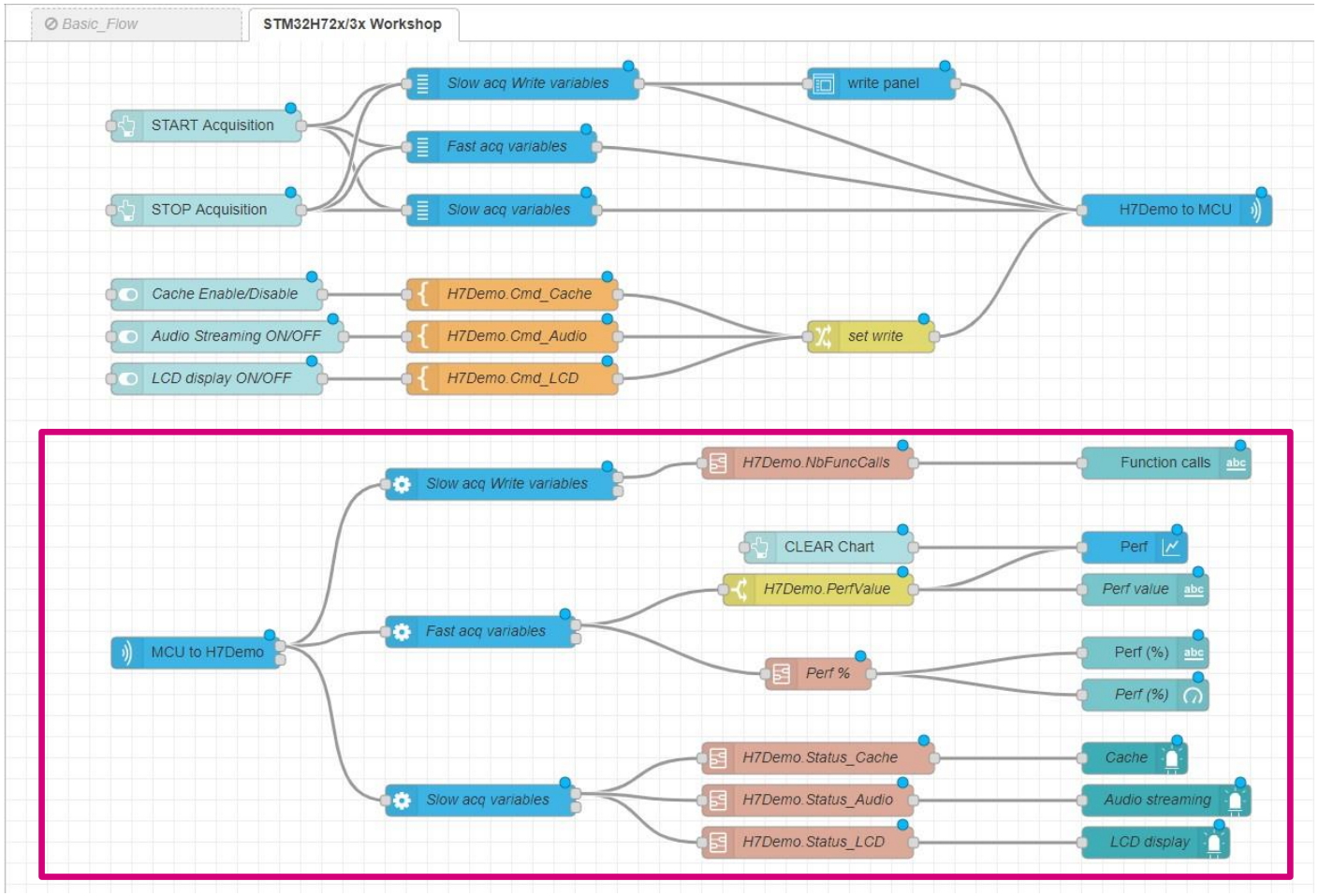
Same flow structure

- One TX path from PC to probe
  - But 3 variable lists
    - Only 1 variable can be written in the write panel
    - The sampling frequency is slow or fast
  - And a new way to write in MCU RAM memory
    - ➢ 3 switch widgets allows to write to 3 variables
    - ➢ But the same message to the ST-Link probe than the write panel
      - ➢ “write this value at that address”
      - ➢ 0 for switch OFF, 1 for switch ON



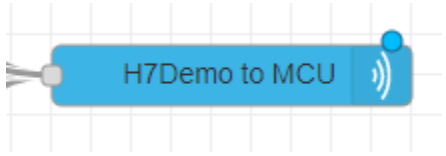
Same flow structure

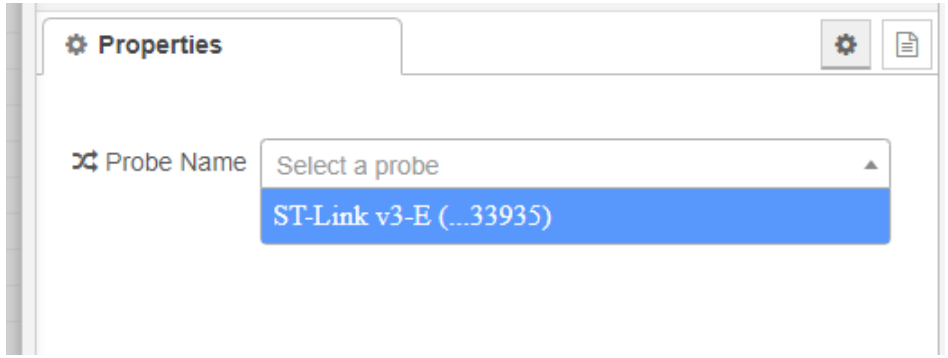
- One RX path from probe to PC
  - 3 variable processing nodes
    - Paired with each variable list
  - Again 1 chart but also other output widgets
    - Text, gauge, LEDs
  - In between, selection nodes
    - Select which variable is displayed on which widget
      - A variable can be displayed on several widgets
      - Several variables can be displayed on a widget
  - **More customization** but...
  - **Nothing more difficult** than "Basic_Flow"



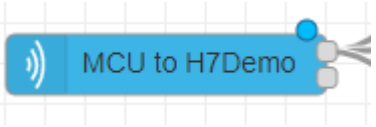
Prepare the flow

- Set ST-Link (Monitor to MCU)
  1. Double-click on “H7Demo to MCU”
  2. Click on the pen of “Probe Config”
  3. Select the ST-Link probe
  4. Click on “Add”
  5. Click on “Done”





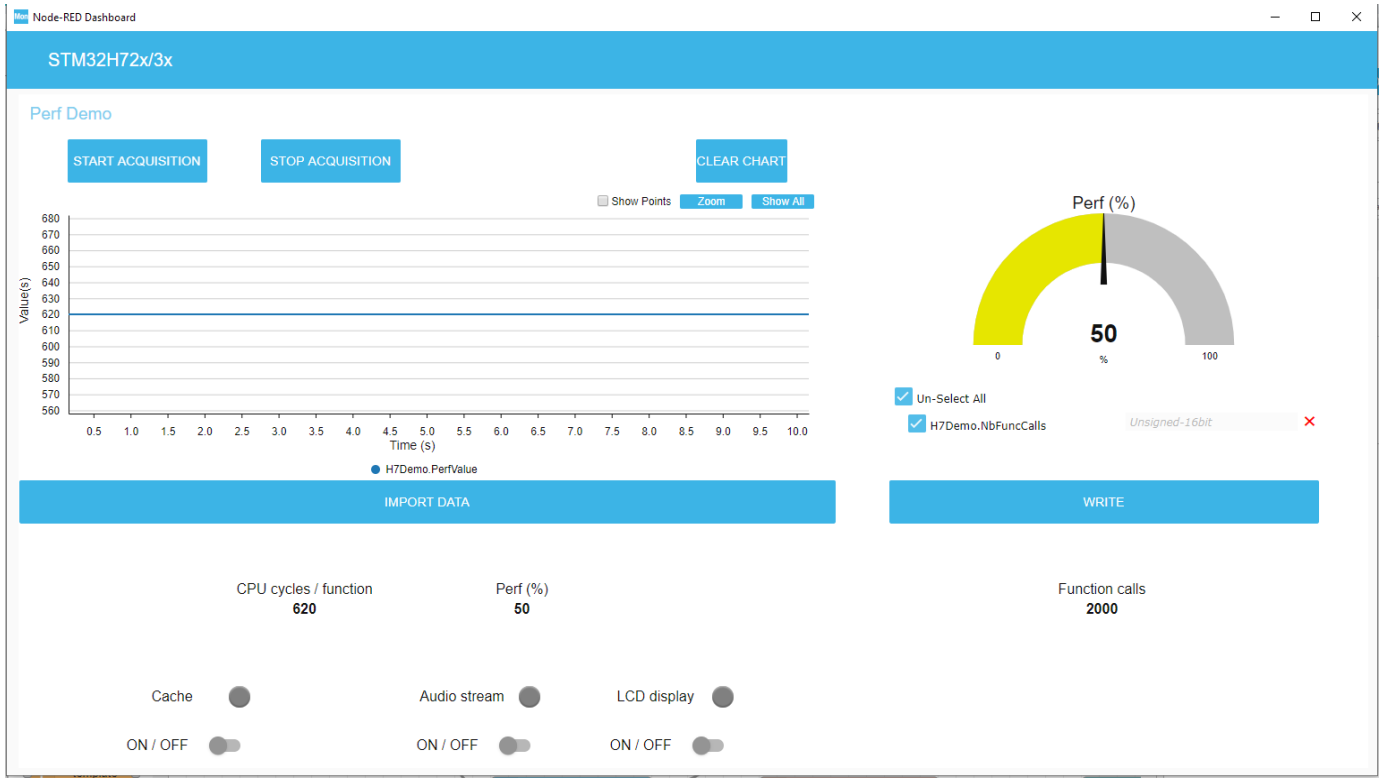
- Set ST-Link (MCU to Monitor)
  - Do the same for “MCU to H7Demo”



life.augmented

Launch the dashboard

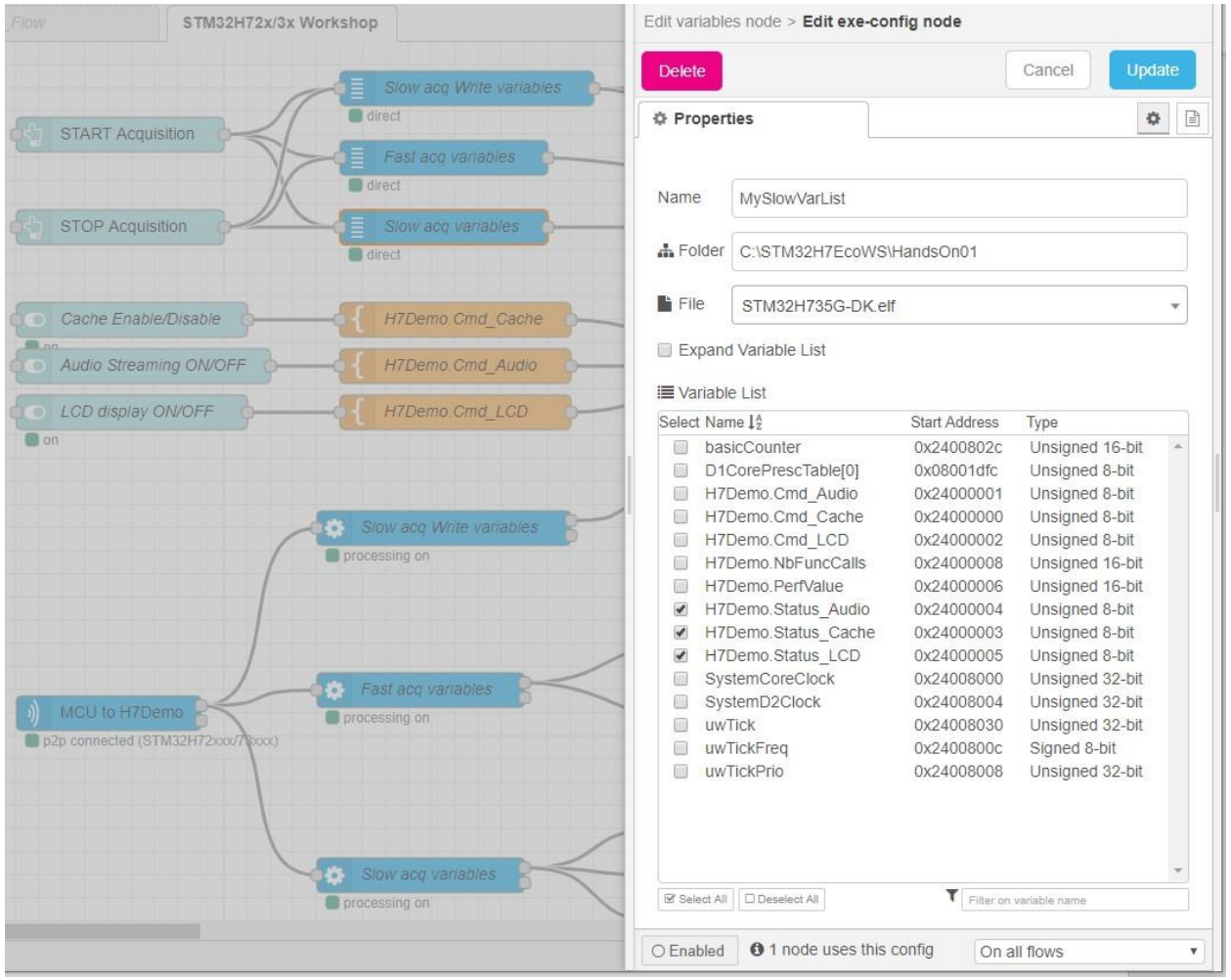
- The dashboard displays
  - Known widgets
    - 3 Buttons
      - START, STOP, CLEAR
    - 1 chart, 1 write panel
  - New widgets
    - 1 gauge
    - 3 text fields
    - 3 switches and LEDs



Variables at fixed address

- Other firmware will be downloaded to the board for HandsOn2 and HandsOn3

- But the same variables will be used.
  - ➢ Variables in a structure at fixed address
    - H7Demo - 0x24000000 to 0x24000008
  - ➢ Reads and writes at the same addresses
  - ➢ **No need to update the elf file**



Launch the dashboard

- The current firmware is a simulation firmware
  - The same logic but only **dummy actions**

- Example of the cache switch
  - When HandsOn2 firmware will do

```
if(H7Demo.Status_Cache != H7Demo.Cmd_Cache){
    if (H7DEMO_ON == H7Demo.Cmd_Cache){
        SCB_EnableICache();
        H7Demo.Status_Cache = H7DEMO_ON;
```

    - ➢ The Perf varies because of cache on/off

  - The current firmware does

```
PerfPercent += 50;
H7Demo.Status_Cache = H7DEMO_ON;
```

    - ➢ The cache is **always off**
    - ➢ The Perf is **dummy**

STM32 CubeMonitor

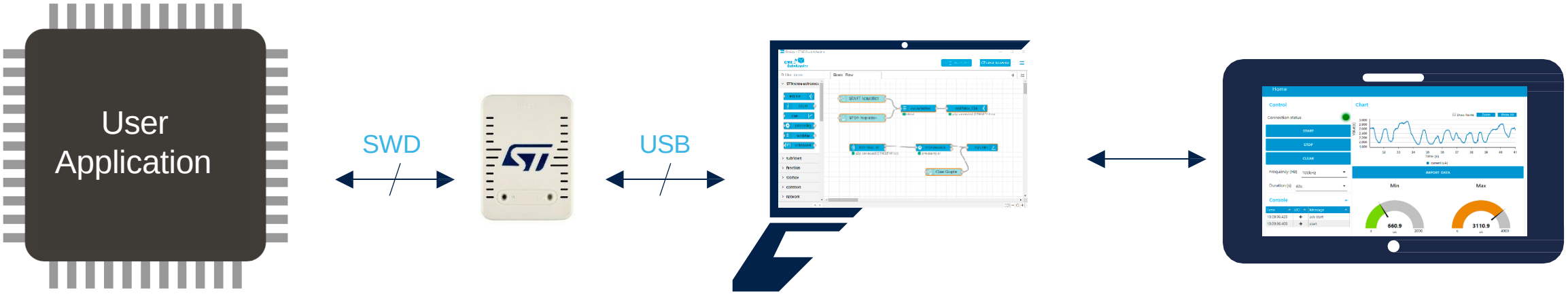
Remote monitoring

Native support of multi-format displays

Dynamic layout of dashboard UI on PCs, tablets, smartphones.

Remote data acquisition with web server technology

Monitor across a network with a web browser



STM32 STLINK probe Gateway Dashboard

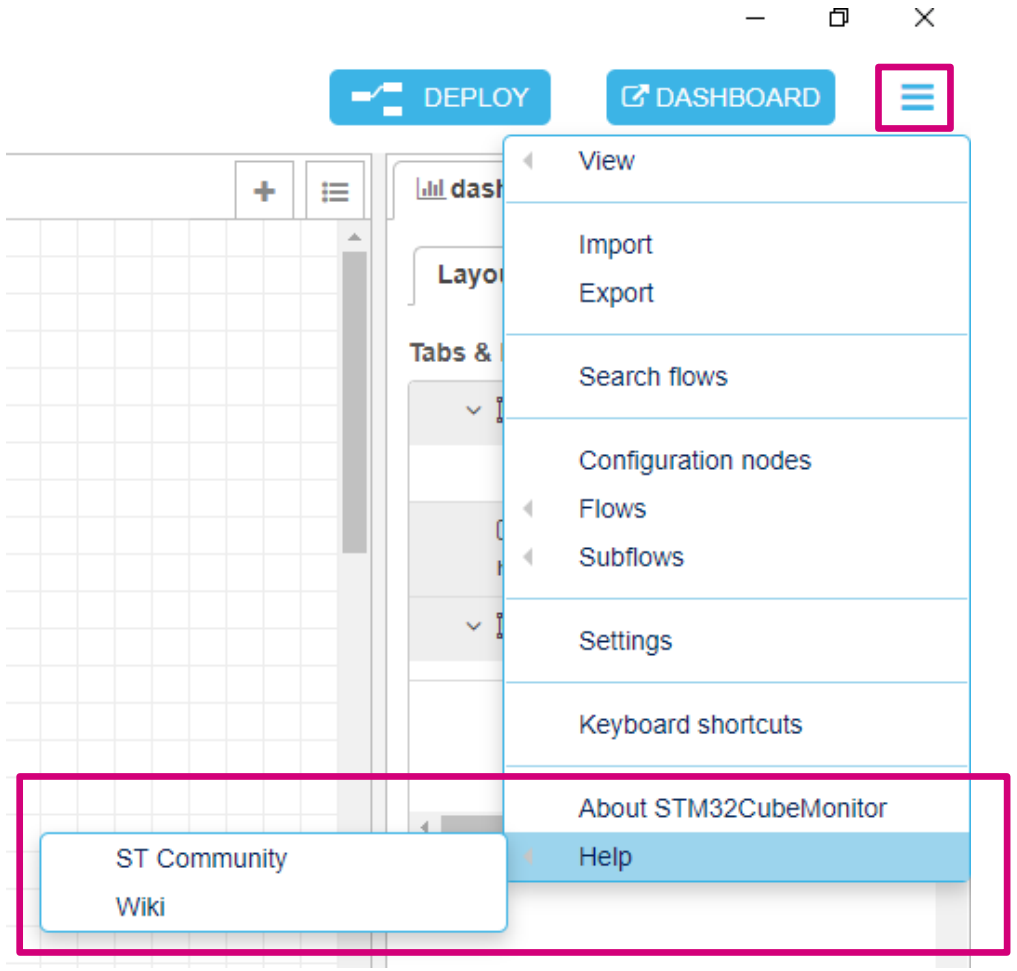
Conclusion

- STM32CubeMonitor is a non-intrusive tool to monitor a running application
  - ➢ Uses the MCU debug bloc and the SWD protocol
- STM32CubeMonitor is easily customizable.
  - We have seen the basics:
    - How to **configure** the flow
    - How to **access** to the STM32 memory
    - How to add **widgets** to the dashboard
  - ➢ You are **ready to design your own** dashboard

Going further - Documentation & Links

To go further

- The help is directly accessible through the menu
  - Allow to find information to guide the user
    - ST Community
    - STM32 MCU wiki



life.augmented

Resources and useful links

Visit our STM32CubeMonitor web page

Read our technical documentation on STM32CubeMonitor

Share use cases with STM32 community

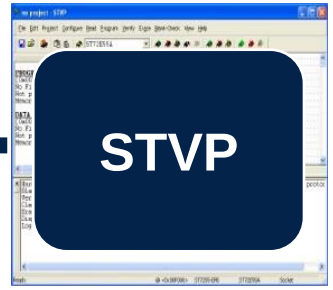
Watch our Getting started webinar (video)

Read our latest blog post

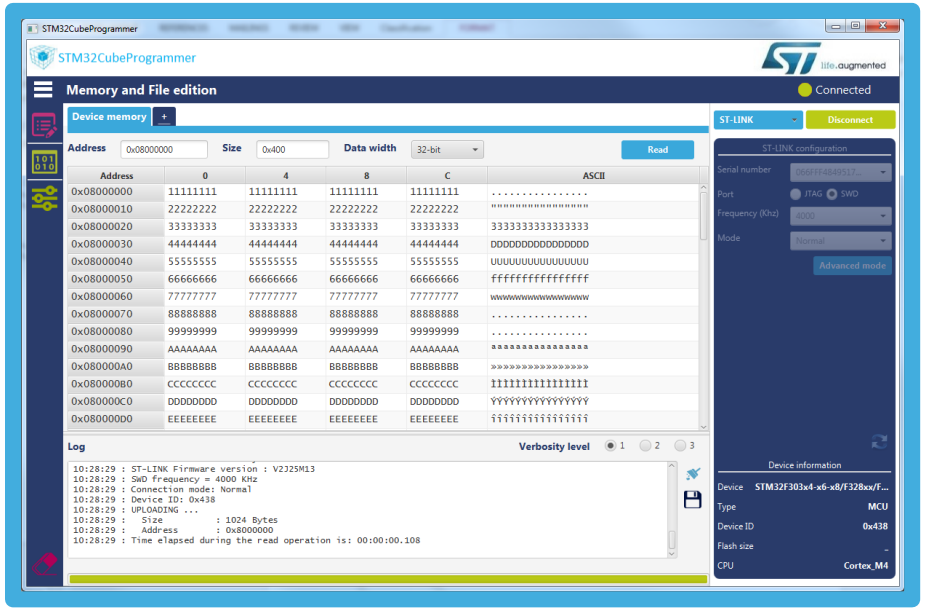
STM32CubeProgrammer

STM32 CubeProgrammer

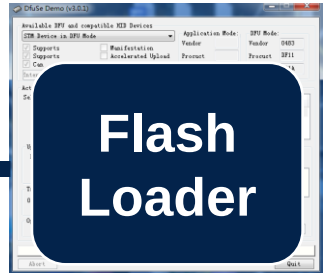
From multiple utilities to STM32CubeProgrammer



– debug interface of JTAG/SWD
internal flash, ext. memory,
option byte



– upgrade STLINK



– bootloader interface of UART
– IAP, ISP

Windows

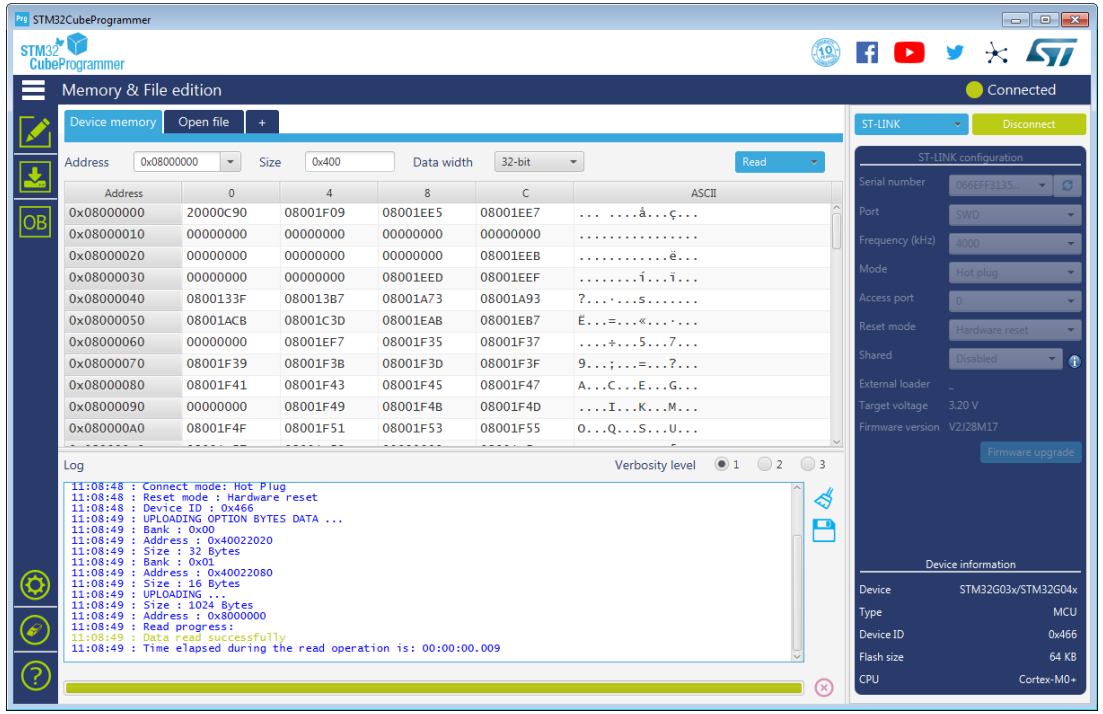
macOS

– bootloader interface of USB
– IAP, ISP

life.augmented

All-in-one programming software tool

STM32 CubeProgrammer



Intuitive GUI

Multi-platform
(Windows, Linux, macOS)

STLink Direct Support
(JTAG, SWD)

Automatic Mode

Option Bytes
Program & Upload

Bootloader Interface Support
(USB, UART, **SPI, I2C, CAN**)

Internal/External Flash Services

API DLL
for Custom Integration

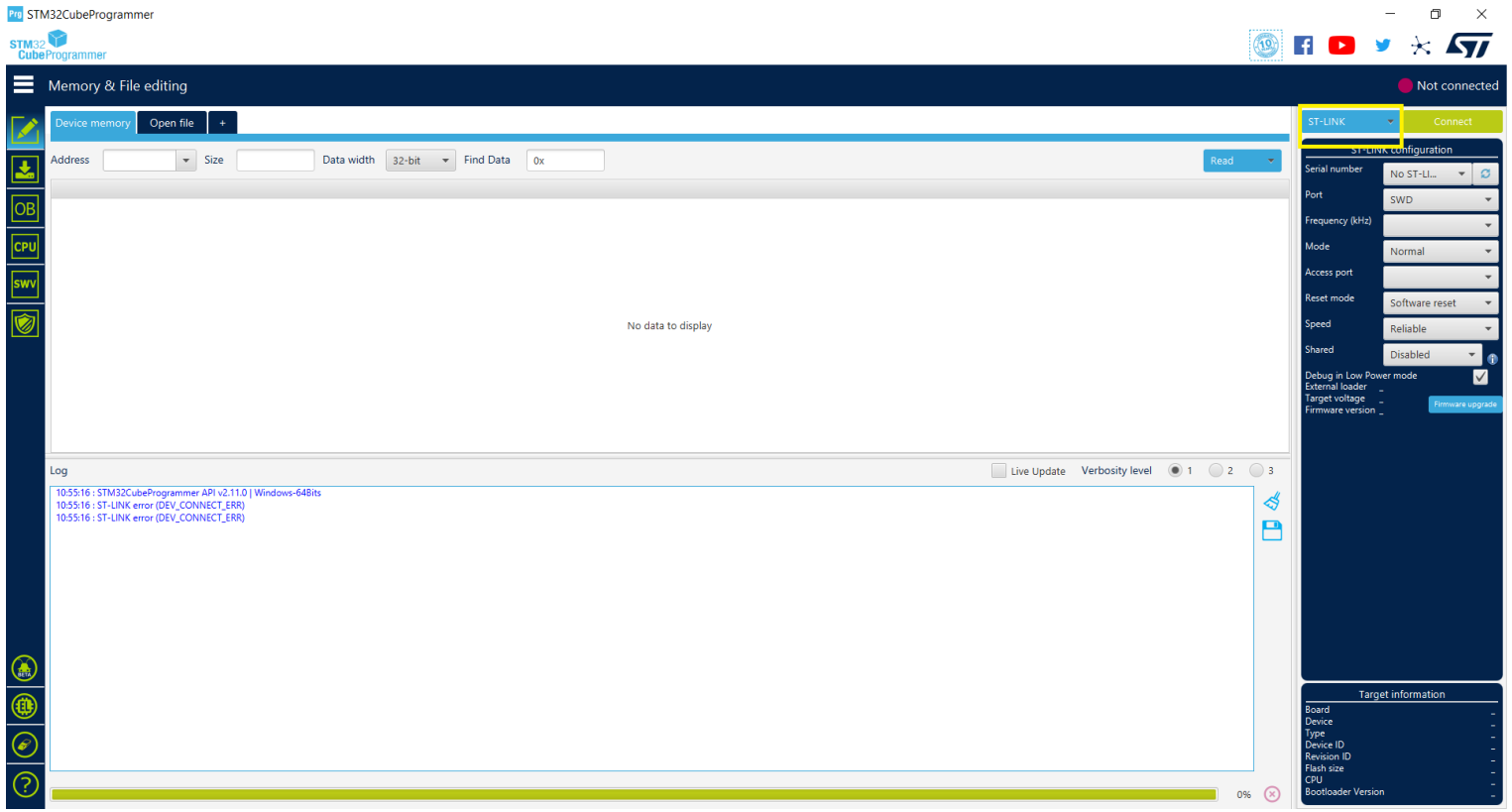
Command Line Interface
for Scripting

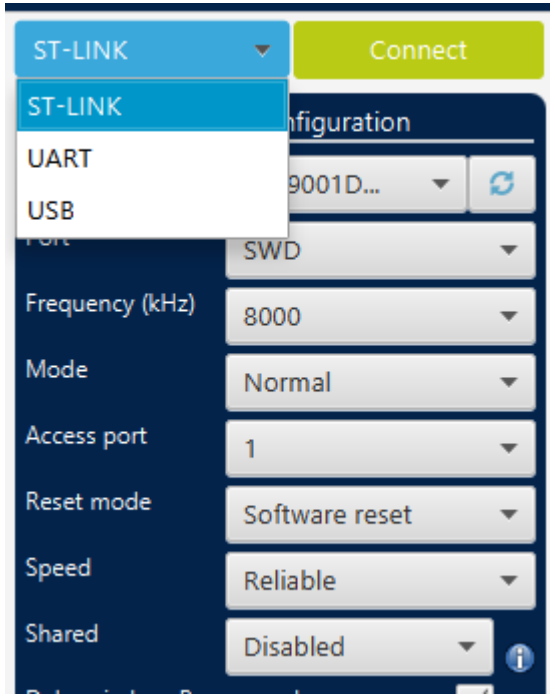
Trusted Package Creator
(secure programming)

life.augmented

STM32 CubeProgrammer

Supported interfaces





ST-LINK

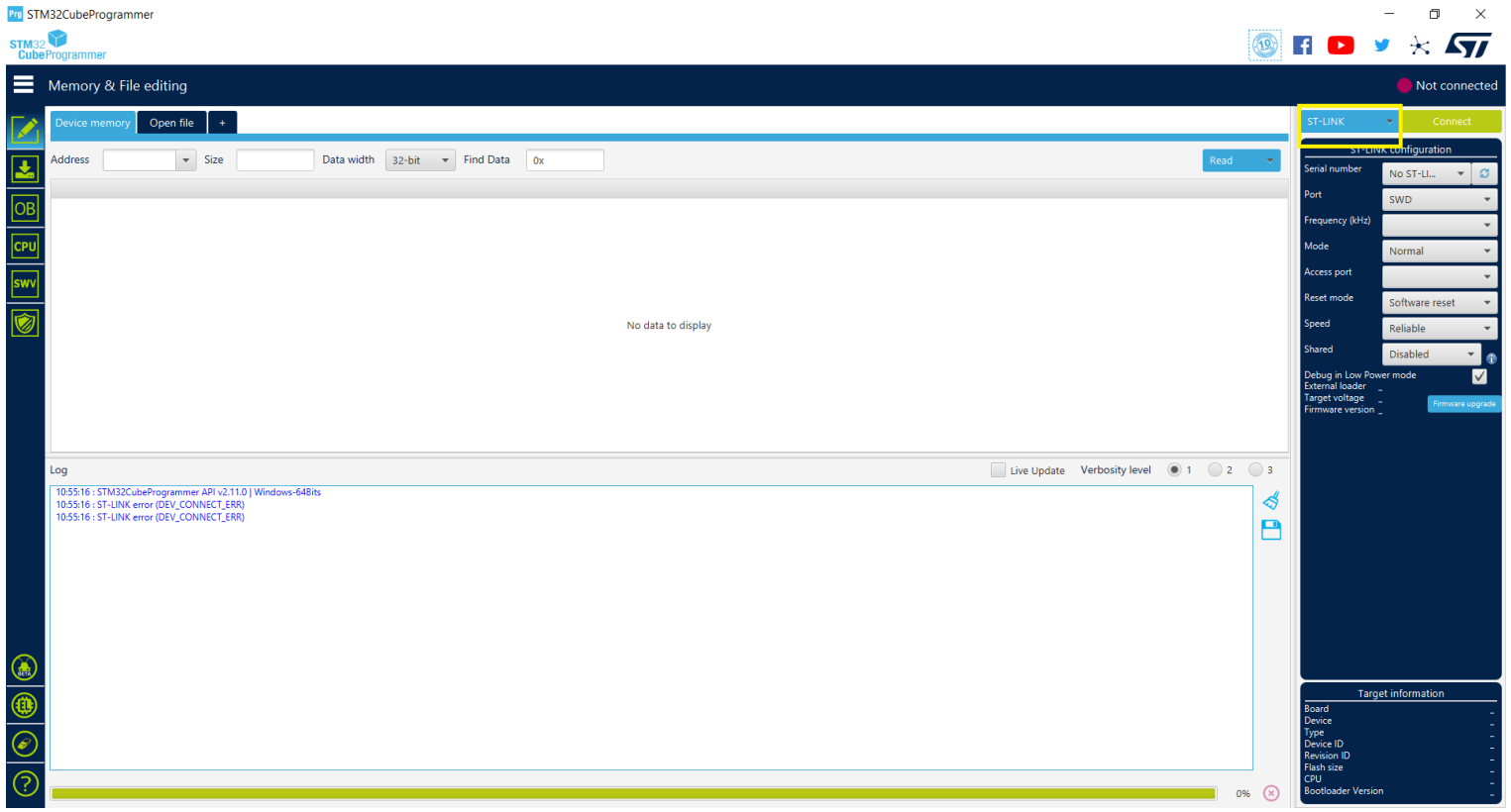
UART

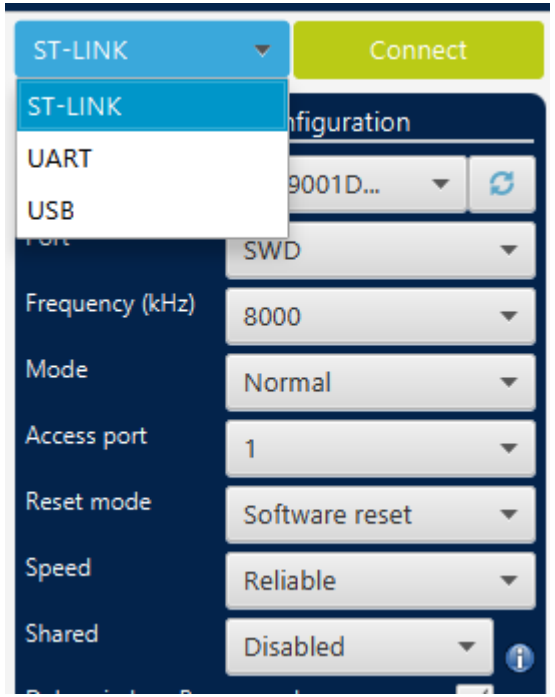
USB

life.augmented

STM32 CubeProgrammer

Supported interfaces





- SPI
- CAN
- I2C

life.augmented

Memory & file editing

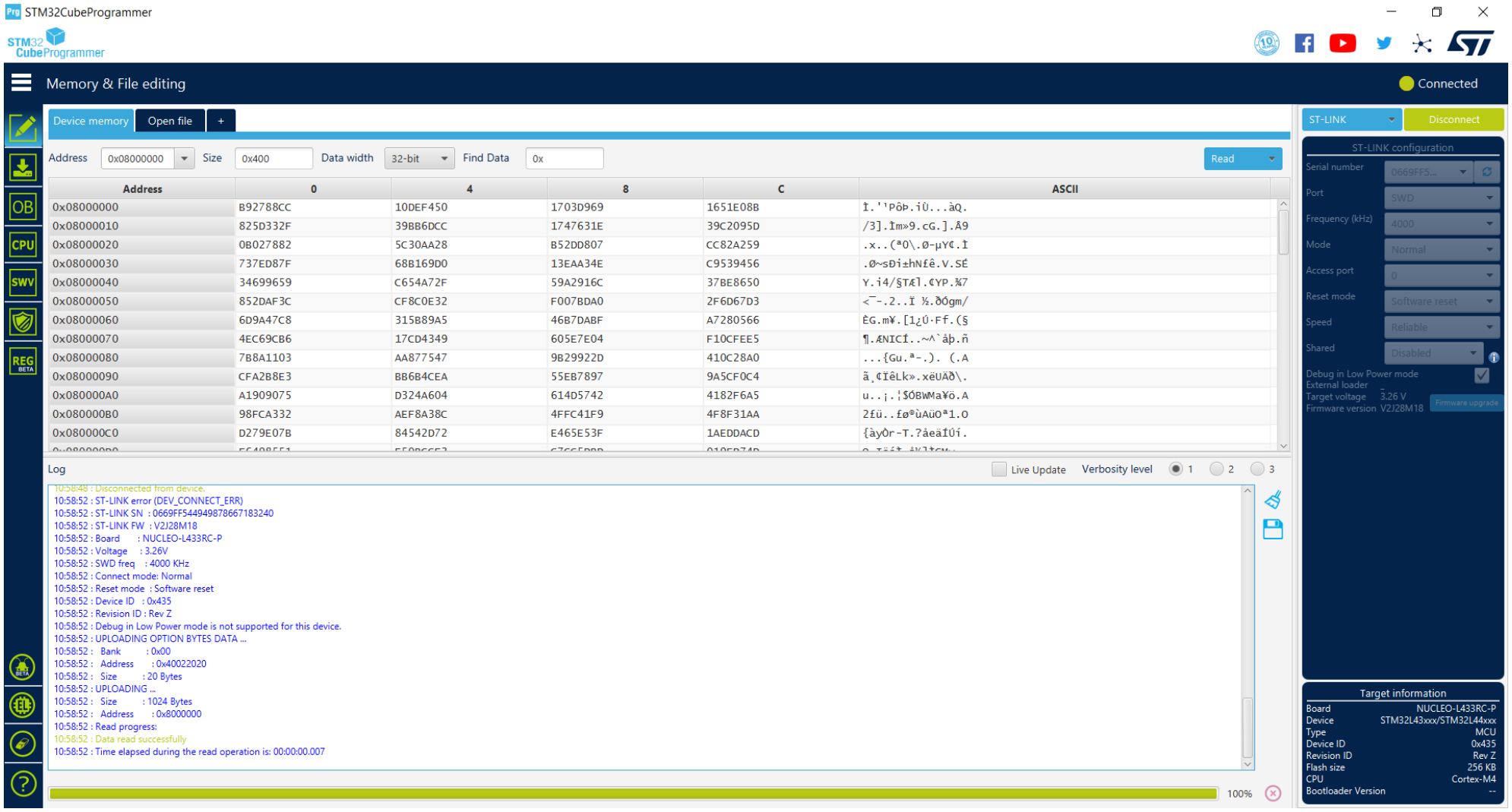
Memory & file editing

Read

Edit

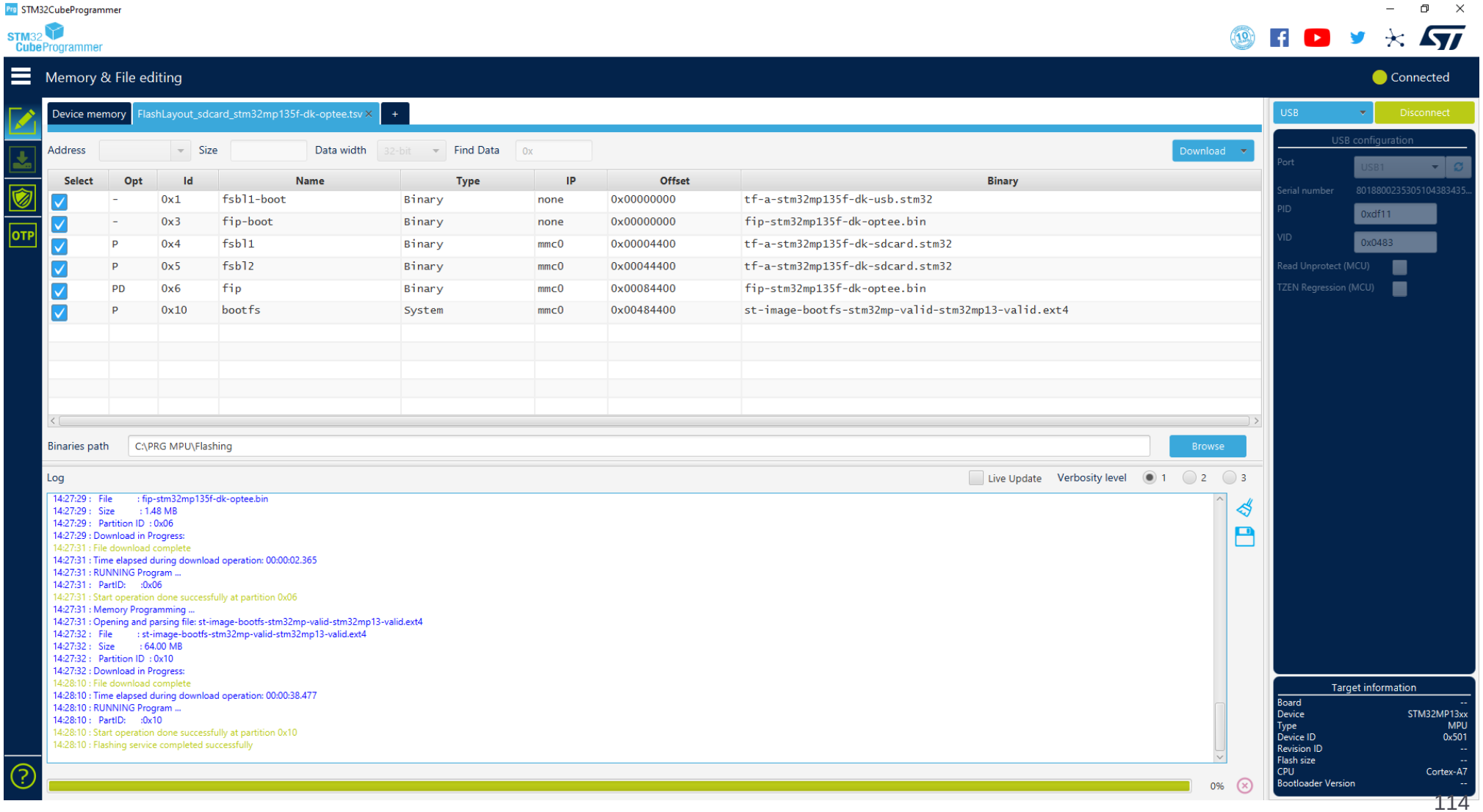
Download

Verify



Memory & file editing

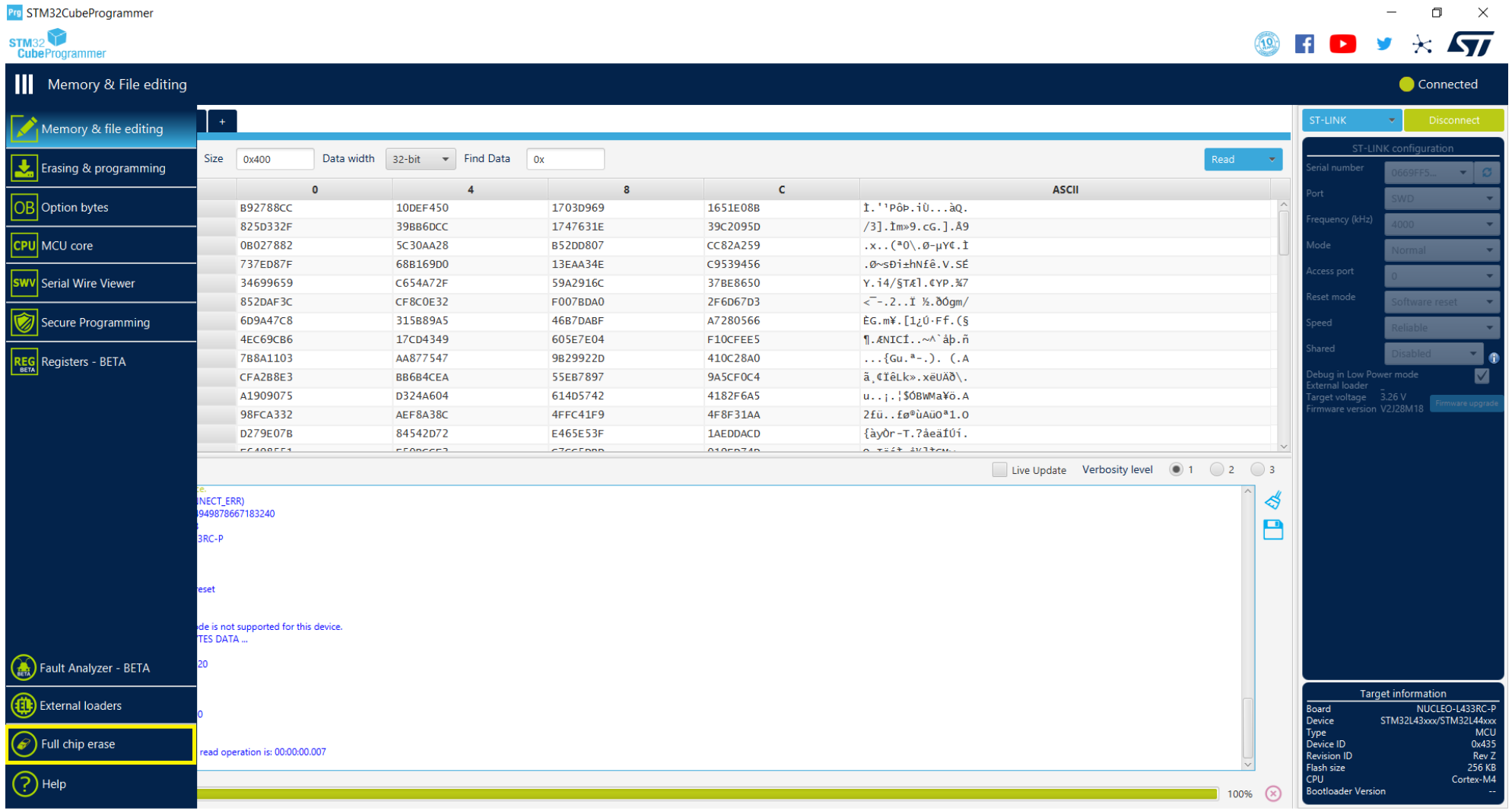
Download



Full chip erase

Full chip erase

Erase entire chip memory



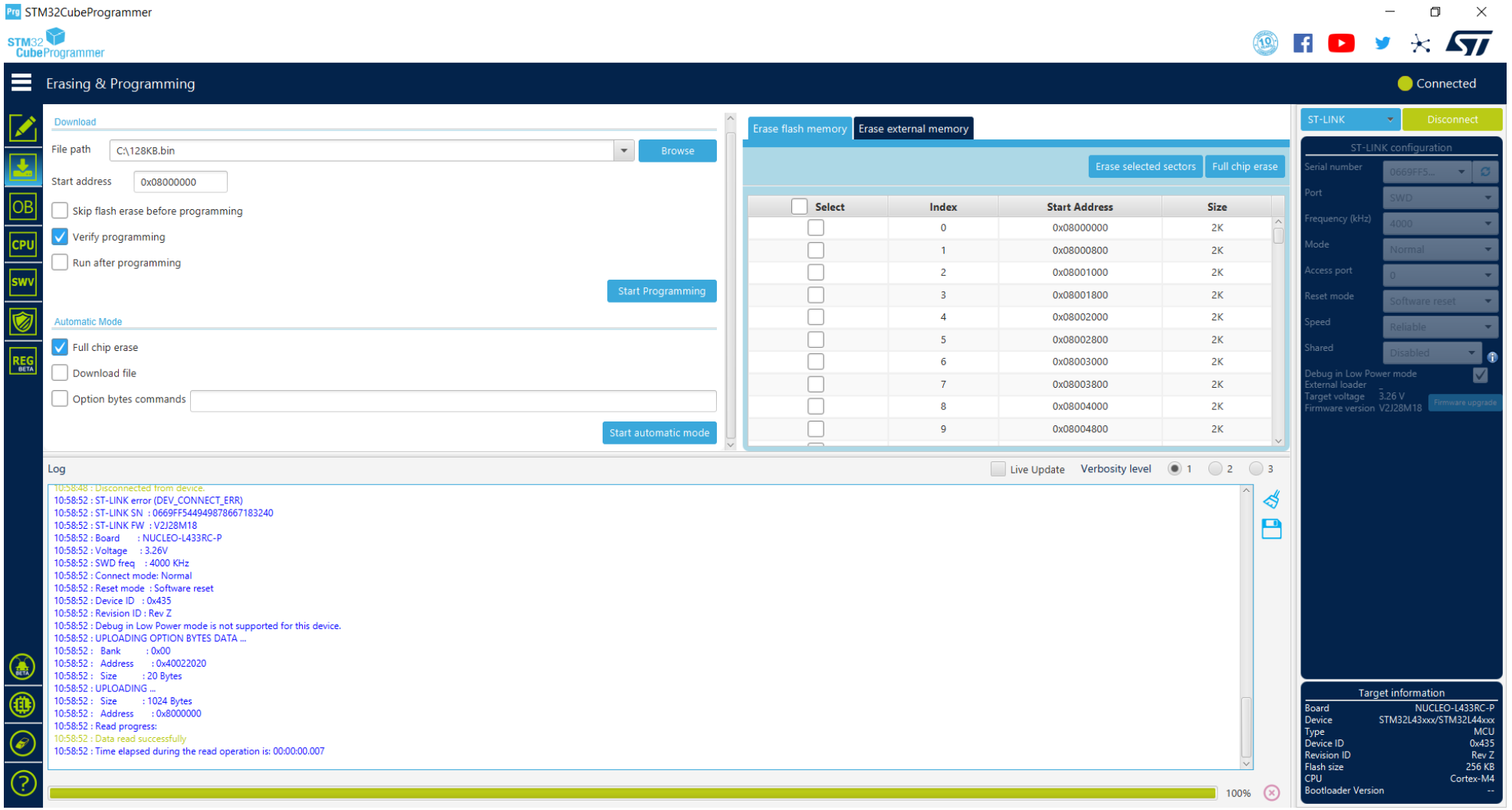
life.augmented

Erasing & programming

File programming (.bin, .hex, .srec, .elf)

Automatic Mode

Sector erase



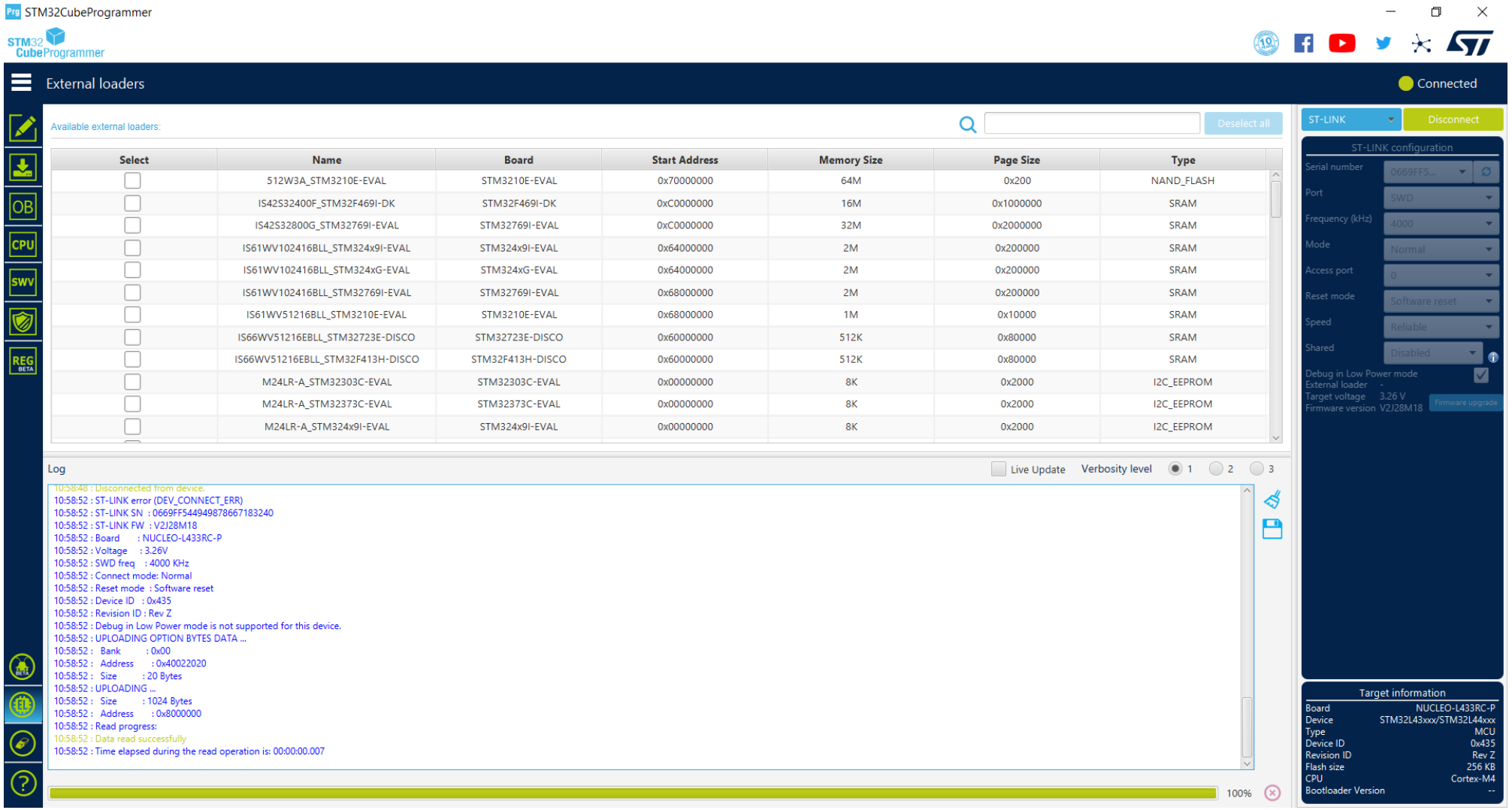
life.augmented

STM32 CubeProgrammer

External loaders

External loaders

Program External memory



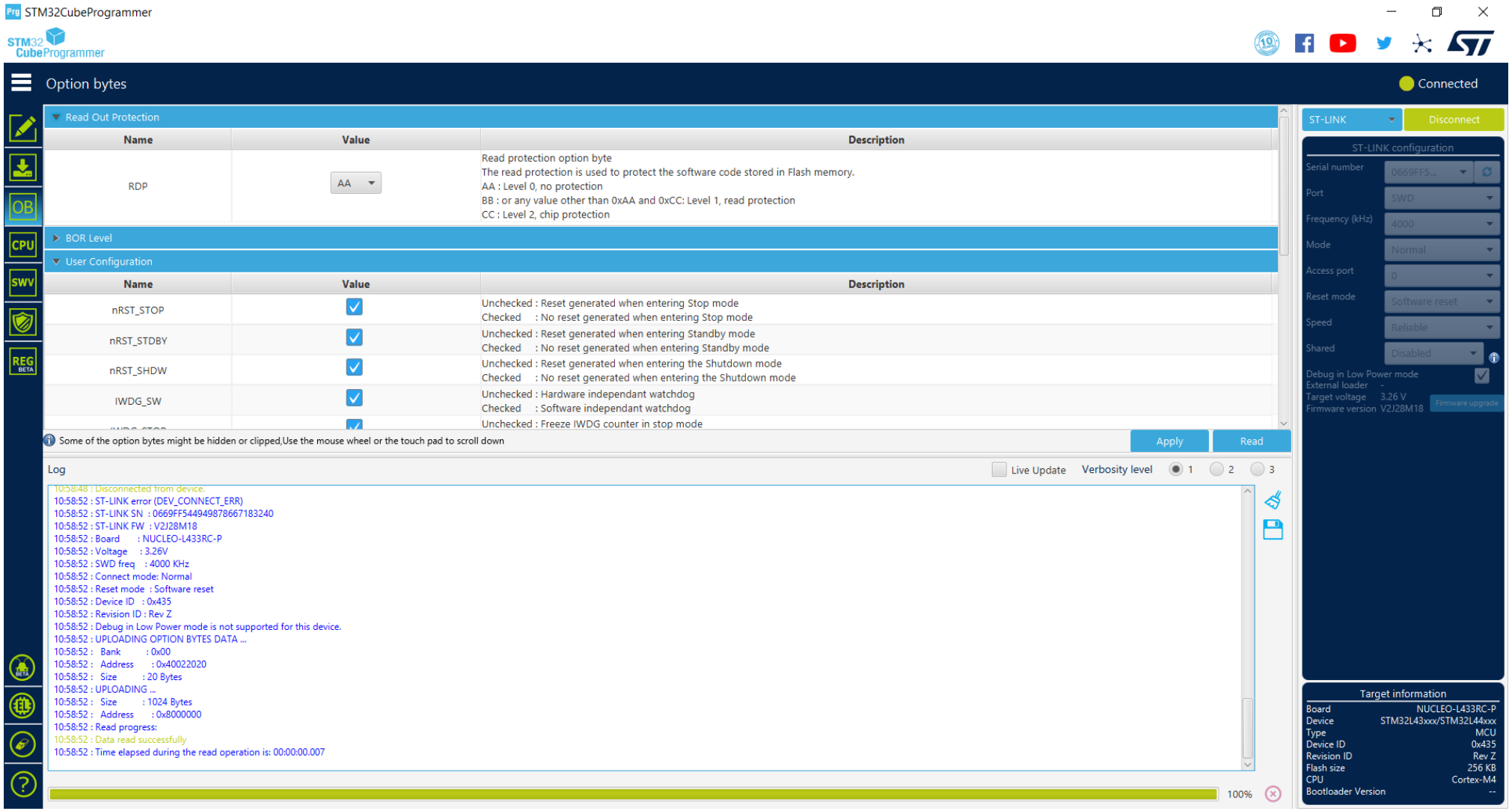
life.augmented

STM32 CubeProgrammer

Option Bytes

OB Option bytes

Read & Program option Bytes

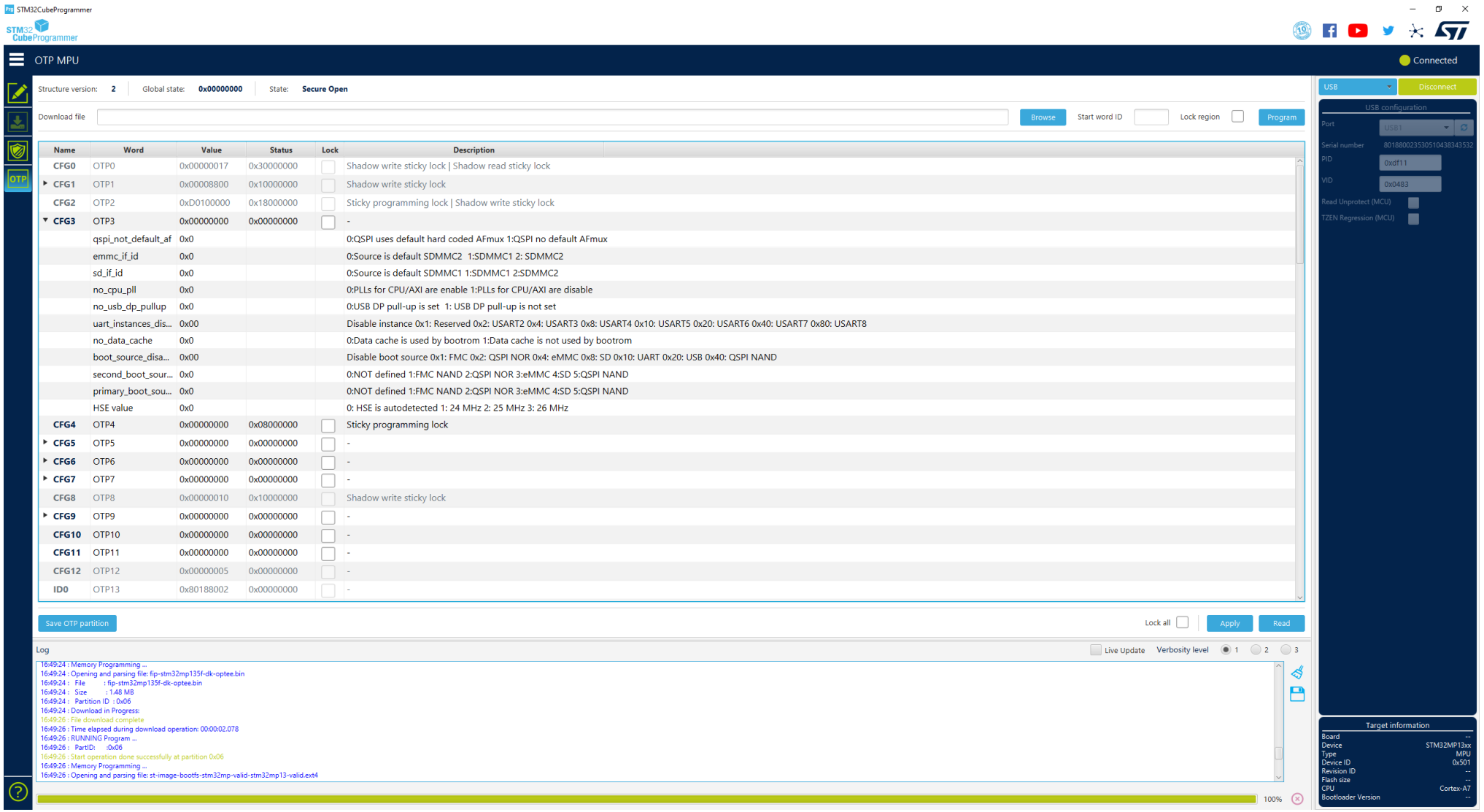


life.augmented

STM32 CubeProgrammer

OTP MPU

OTP programming



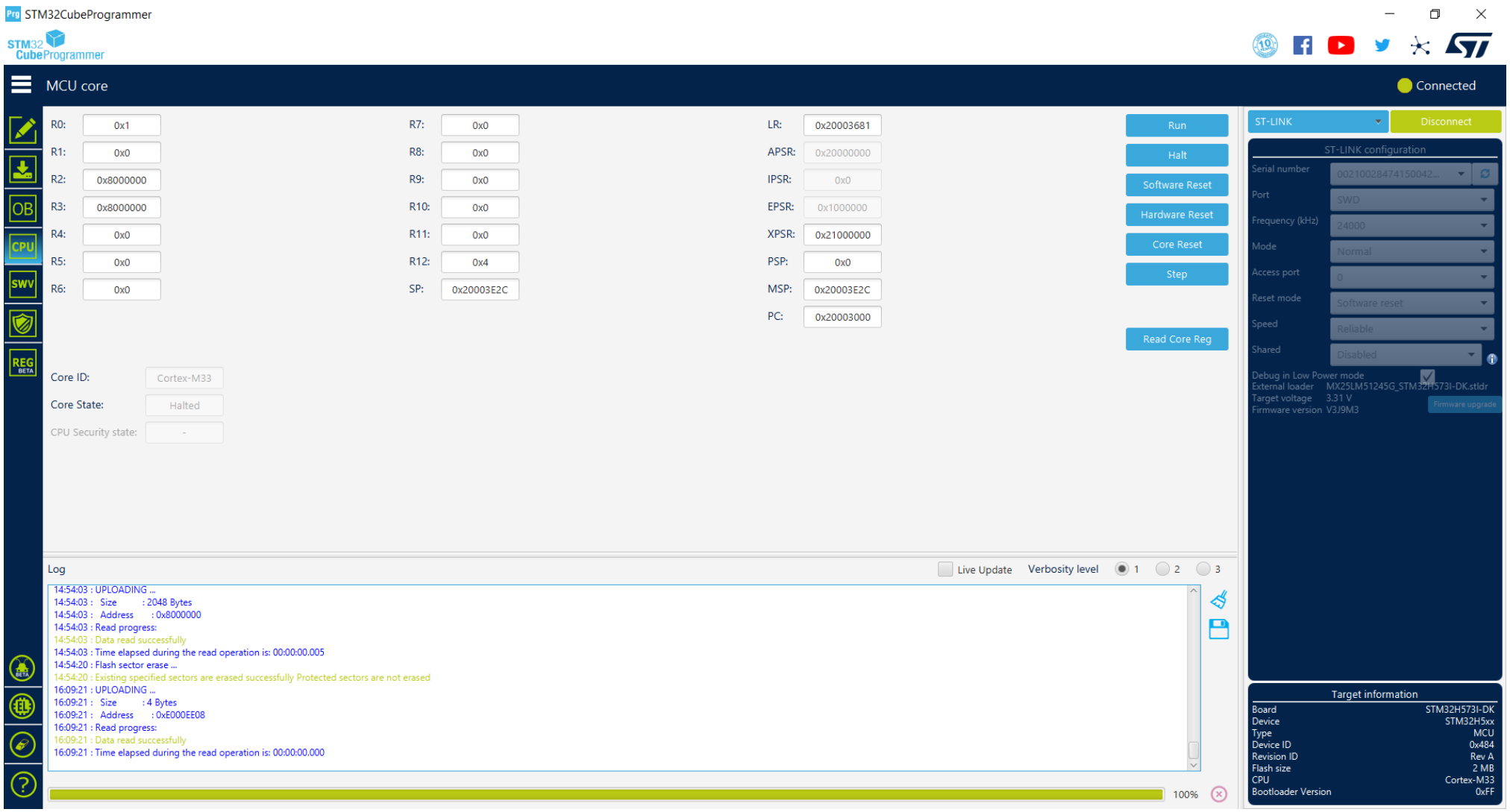
life.augmented

STM32 CubeProgrammer

MCU core

CPU MCU core

Manipulate register



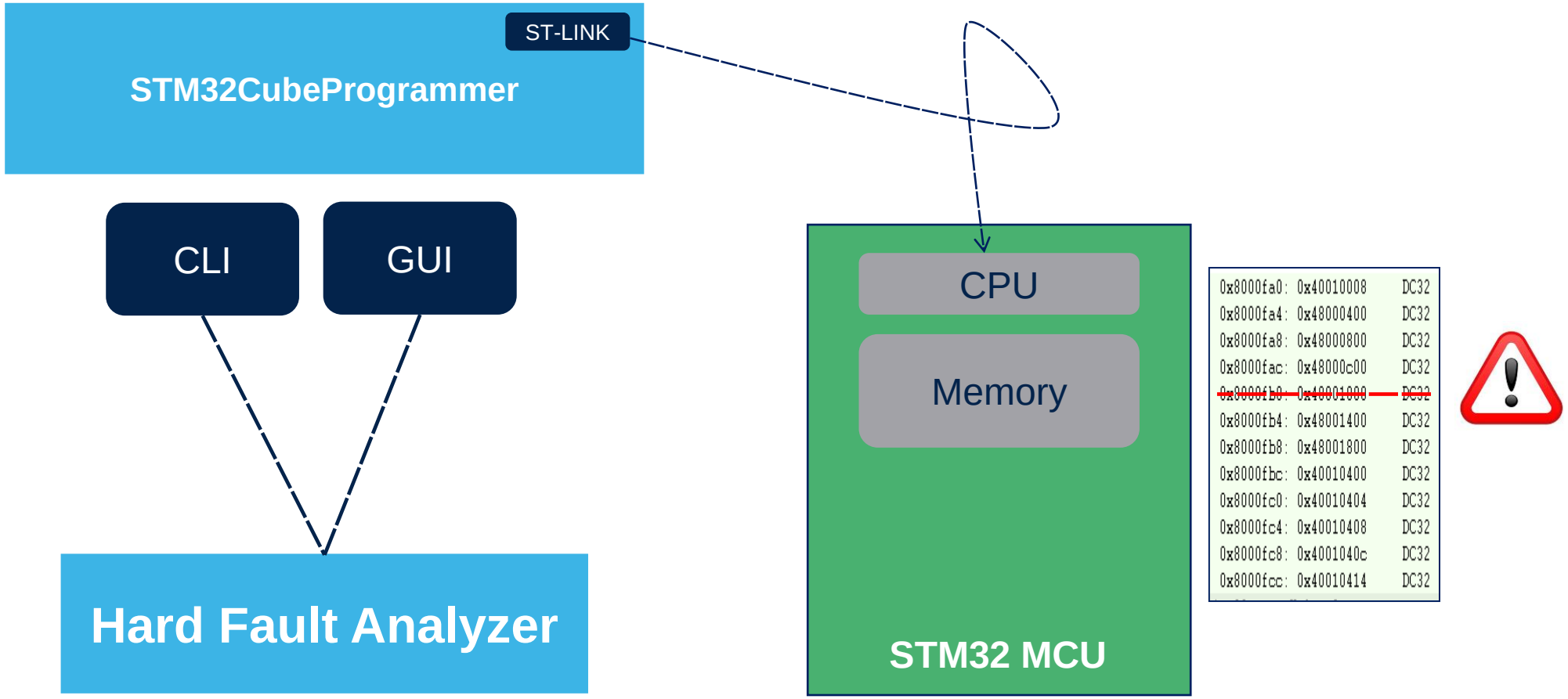
life.augmented

life.augmented

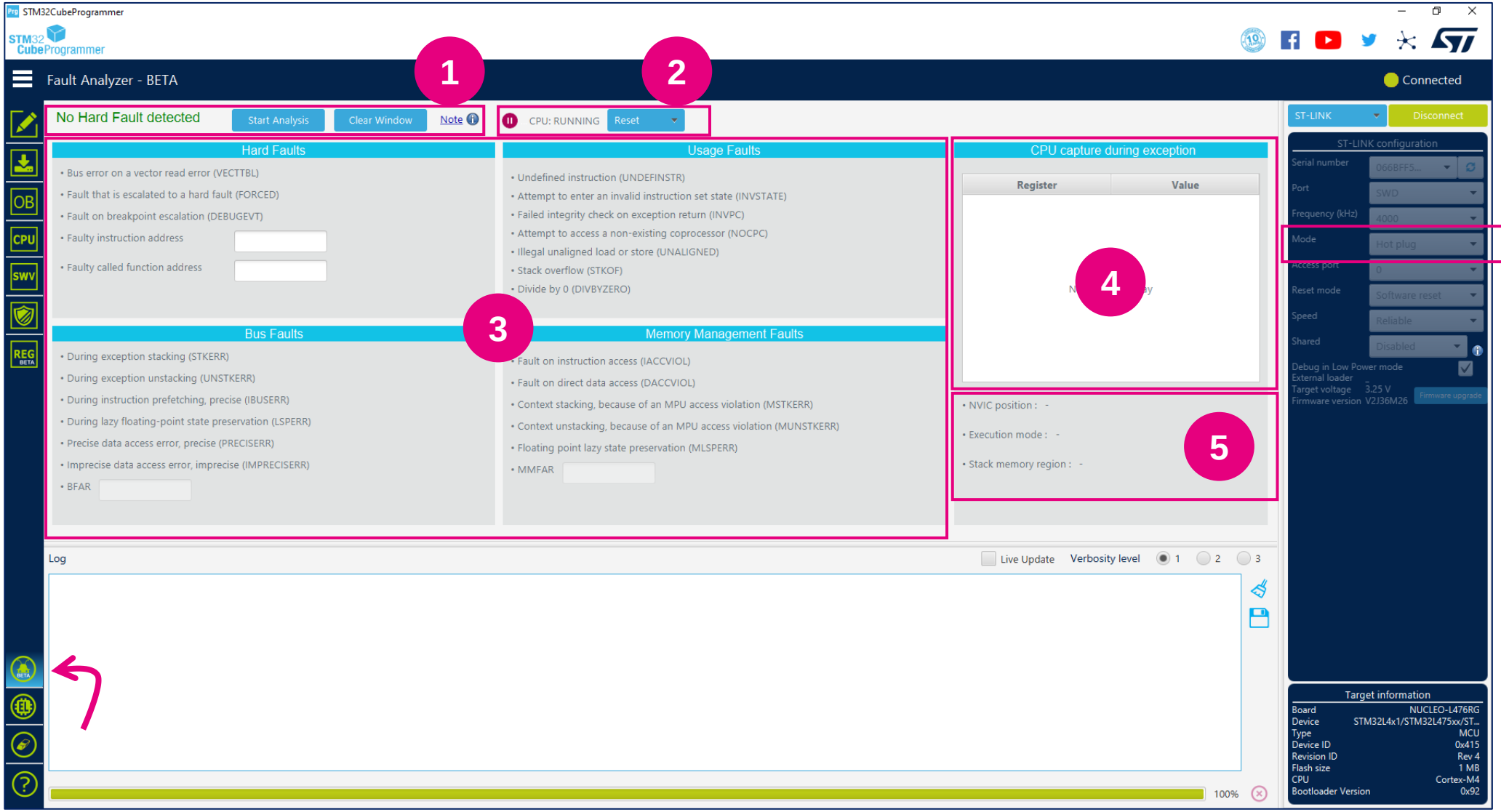
Hard Fault Analyzer

STM32CubeProgrammer

Overview

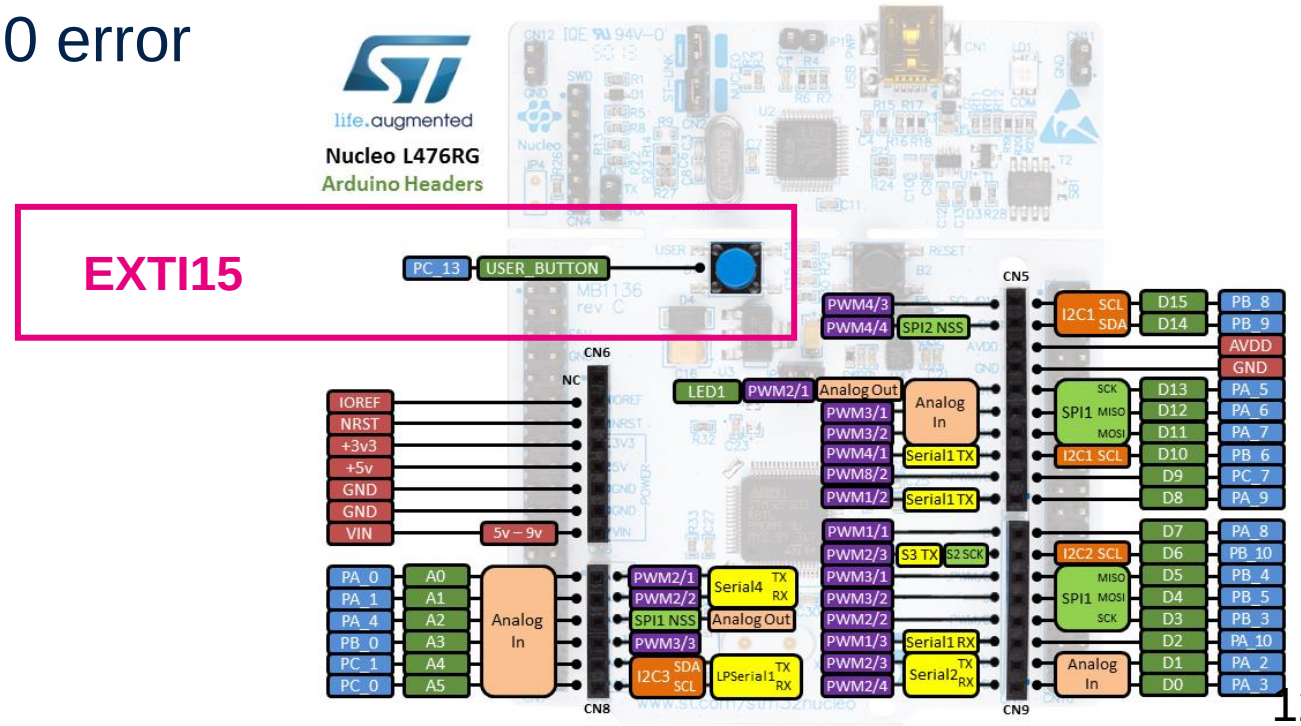

ST-LINK
STM32CubeProgrammer
CLI
GUI
Hard Fault Analyzer
CPU
Memory
STM32 MCU
0x8000fa0: 0x40010008 DC32
0x8000fa4: 0x48000400 DC32
0x8000fa8: 0x48000800 DC32
0x8000fac: 0x48000c00 DC32
0x8000fb0: 0x48001000 DC32
0x8000fb4: 0x48001400 DC32
0x8000fb8: 0x48001800 DC32
0x8000fbc: 0x40010400 DC32
0x8000fc0: 0x40010404 DC32
0x8000fc4: 0x40010408 DC32
0x8000fc8: 0x4001040c DC32
0x8000fcc: 0x40010414 DC32

Hard Fault Analyzer UI



Demo 1: Division By Zero

- **Device** : STM32L476 Nucleo Board
- **Interface** : STLINK via SWD
- **Scenario** : Division By Zero from an EXTI Handler
- **Fault type** : Usage Fault, Devide by 0 error (DIVBYZERO)



Demo 2: illegal READ

- **Device** : STM32U574ZI-Q Nucleo Board
- **Interface** : STLINK via SWD
- **Scenario** : Read a non accessible memory region
- **Fault type** : Bus Fault, Precise data access error (PRECISERR)

Register Viewer

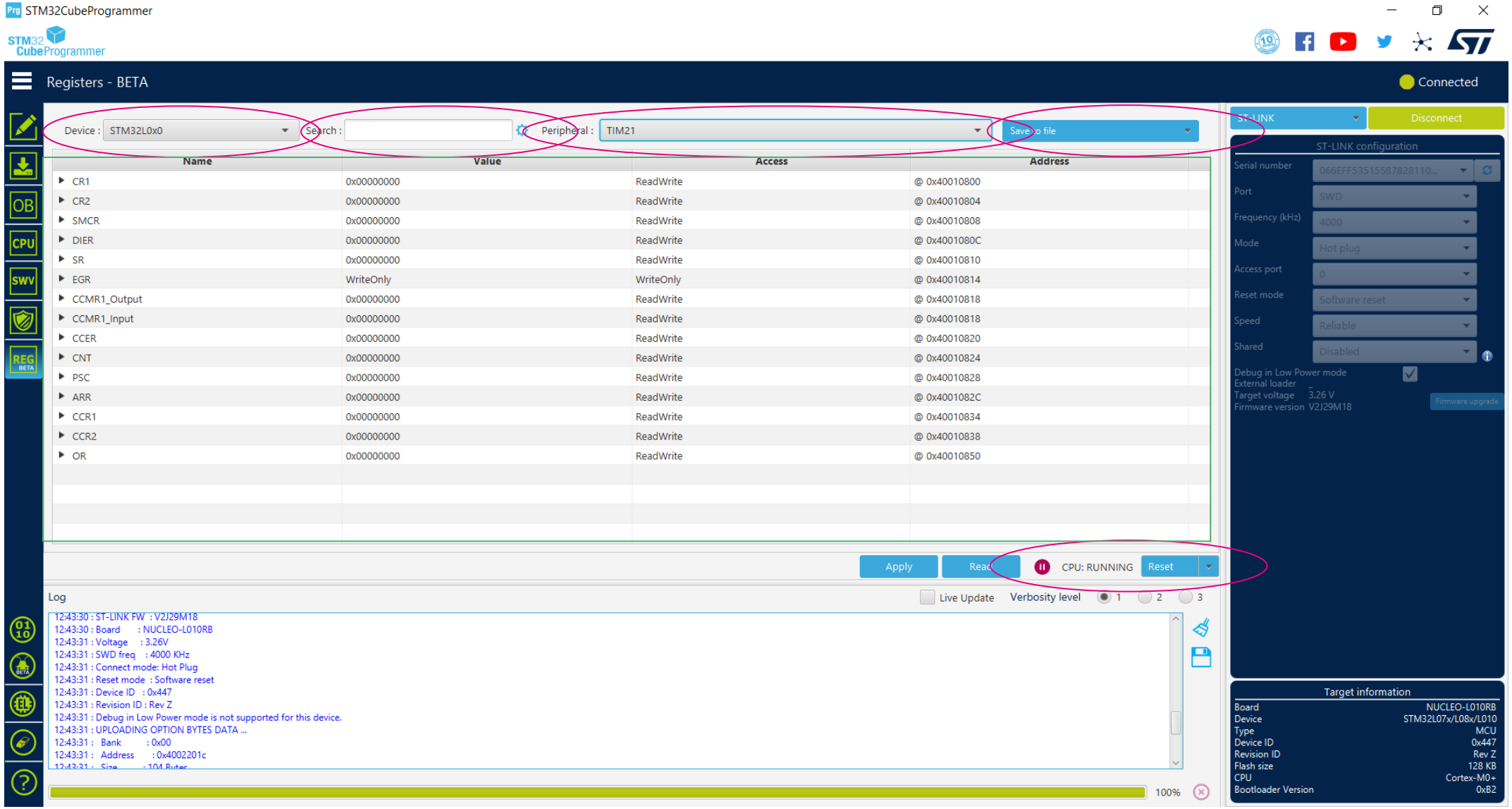
Overview

- Functionalities:
  - ✓Visualize all the MCU and core registers.
  - ✓Modify the MCU registers values.
  - ✓Save MCU registers values into a log file.

CLI Command:

-regdump <file_path.log> [choice=<number>]

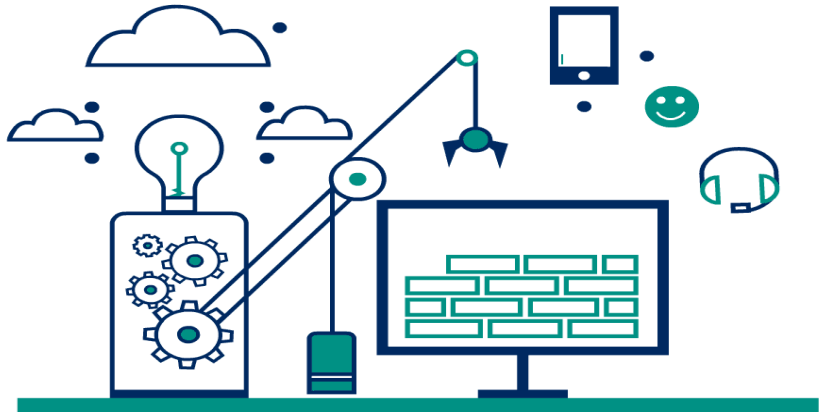
Note: Access using SWD/JTAG.



life.augmented

Register Viewer

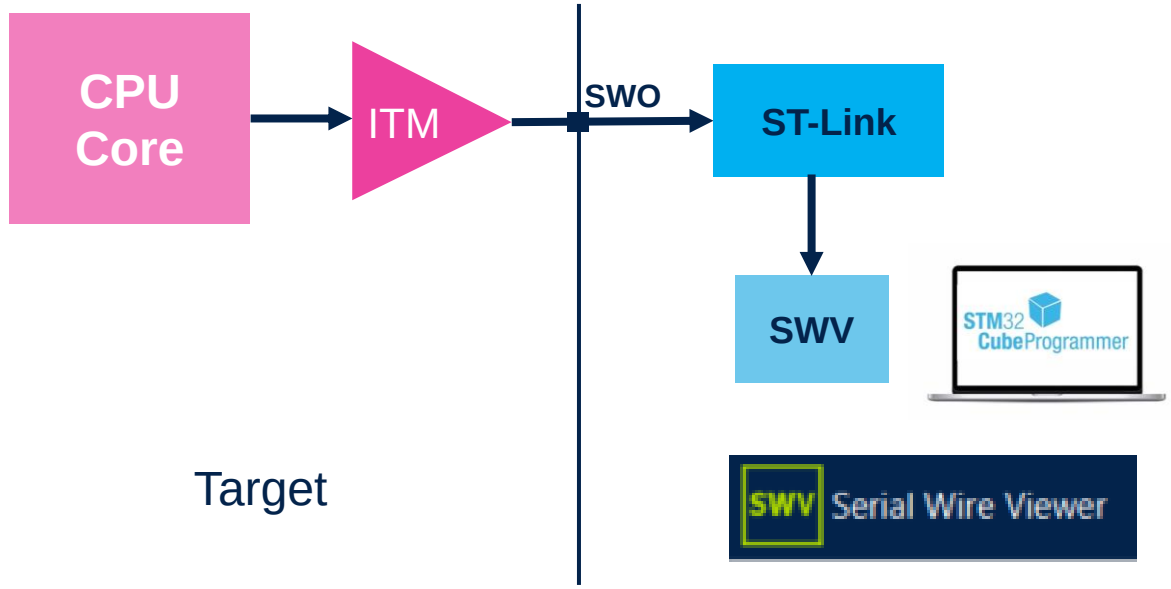
Demo



Serial Wire Viewer

Overview

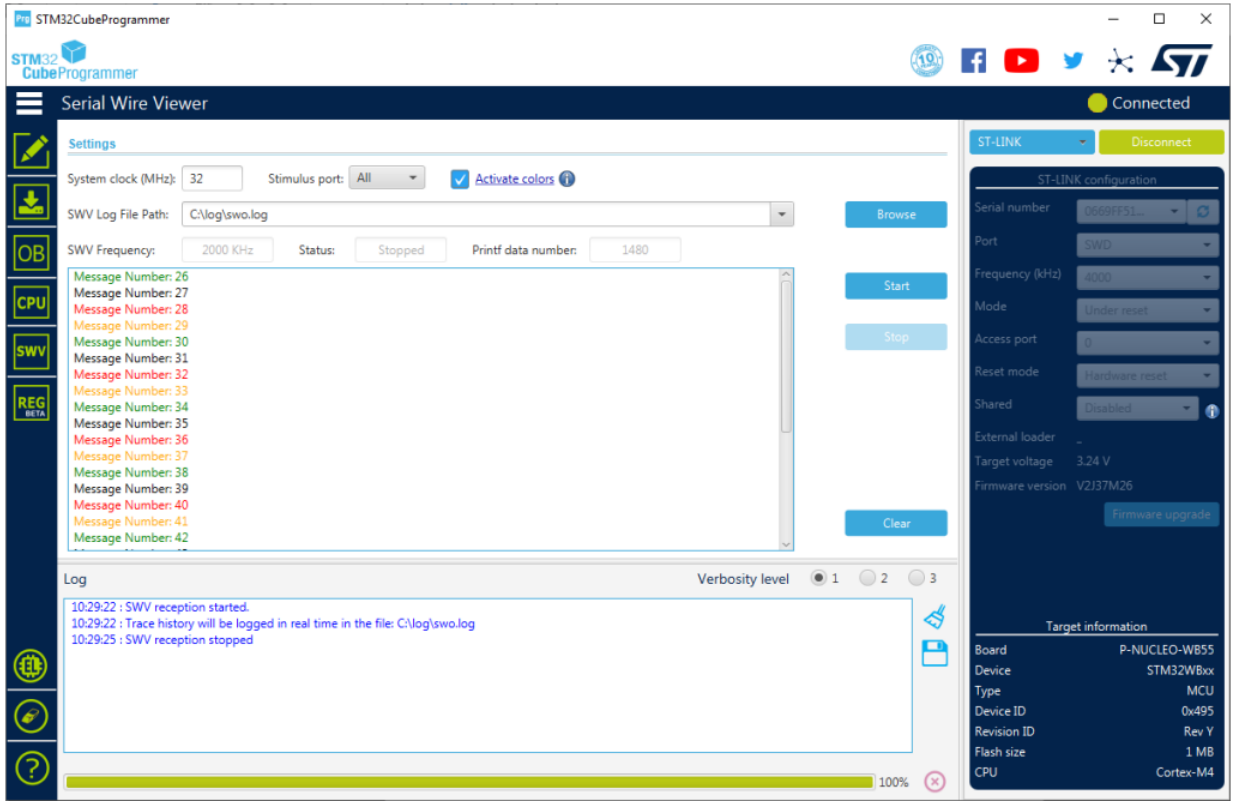
- Display printf data sent from the target
- Available through the SWD interface
- User application should configure the ITM and SWO pin to generate the trace.
- Less intrusive compared to UART/VCP.



Functional description

- Specify target system clock frequency.
- Select the corresponding stimulus port at which the application is sending data.
- Click on the start button.

- Optionally generate a .log file.
- Optionally enable colored traces output.
- Feature available on both CLI and GUI.



life.augmented

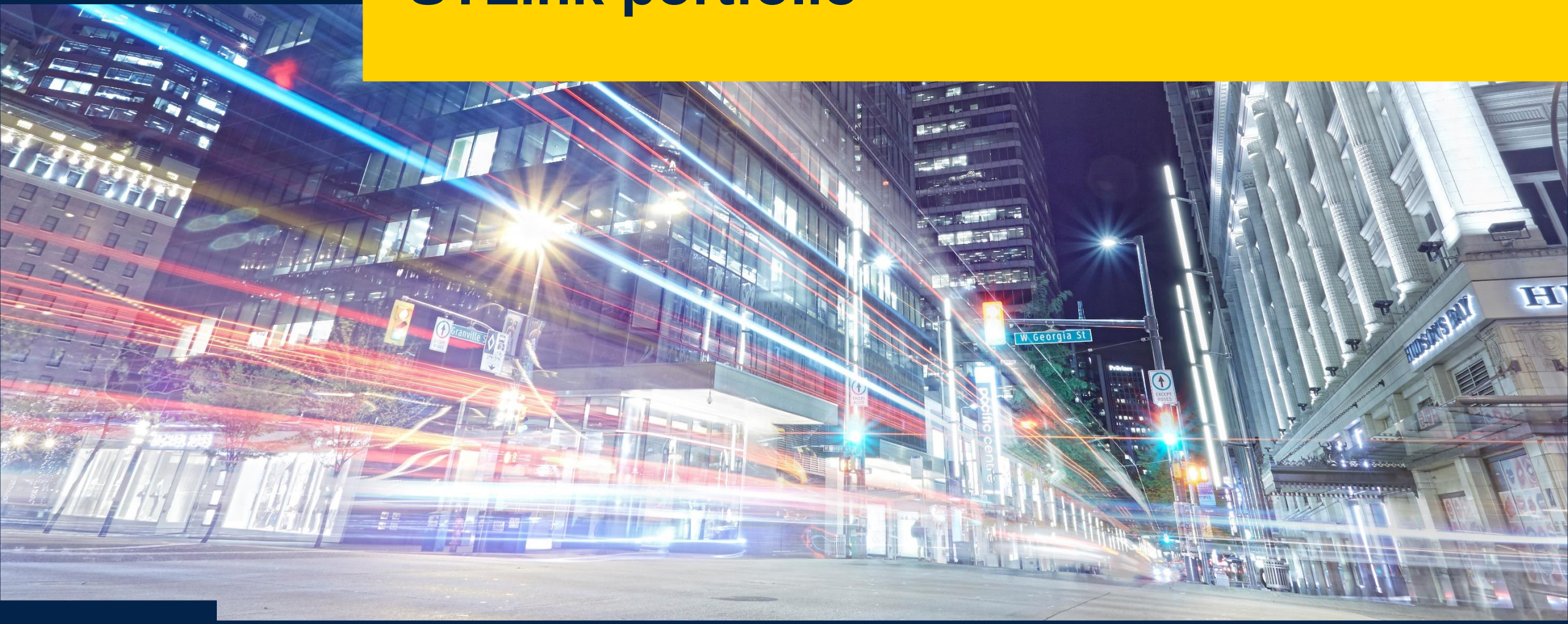
STM32Cubeprogrammer v2.14.0 release information

- Updated the support for the entire STM32H5 series:
  - Debug authentication with password or certificate
  - Authentication key provisioning
  - Key generation
  - Firmware encryption and signing
  - Certificate generation
  - SFI support, secure manager install, and module install/update
- Added support for microcontrollers in the STM32L5 series and STM32WL series:
  - SFI integrity check
- Updated support for the STM32U5 series:
  - SFI support for the STM32U535/545, STM32U575/585, STM32U595/5A5, and STM32U599/5A9 product lines
- Added support for the STM32U5Fxxx and STM32U5Gxxx microcontrollers:
  - SFI support: user interface and command-line interface
  - SFIx support:
    - Via debug: user interface and command-line interface
    - Via bootloader: command-line interface only

Refer to the STM32CubeProgrammer software description user manual (UM2237) for details.

life.augmented

STLink portfolio



life.augmented

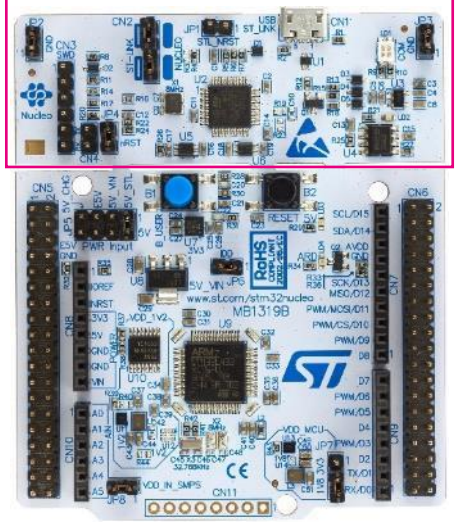
From ST-Link V2 to STLink-V3

ST-Link V2

ST-LINK/V2

ST-LINK/V2-ISOL

ST-LINK/V2-1



STLink-V3

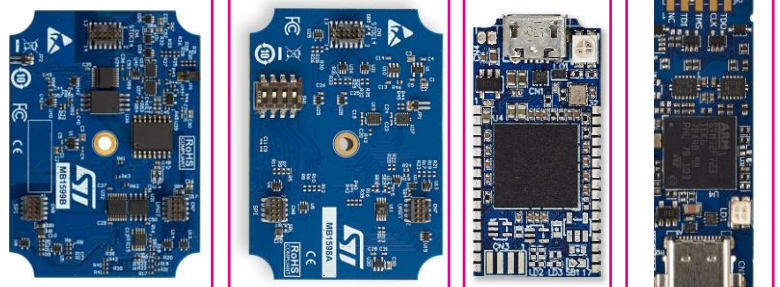
STLINK-V3MINIE

STLINK-V3MODS

B-STLINK-VOLT

B-STLINK-ISOL

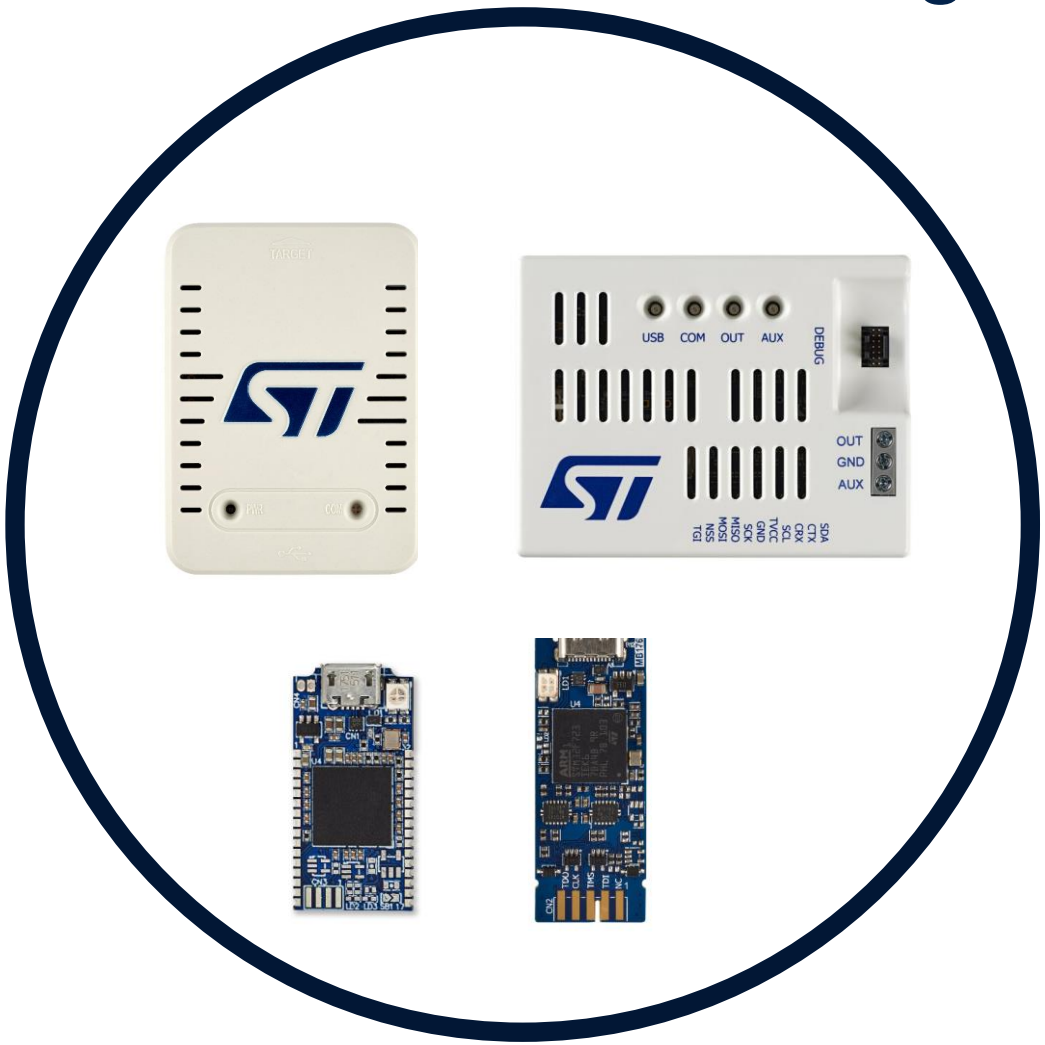
STLINK-V3SET



Why upgrading : STLinkV3 has better performances, more features, and mods.

life.augmented

STLINK-V3 tools enable developers to debug / program with more efficiency



Easier

Faster

More Flexible

life.augmented

STLINK evolution

Energy profiler				****
Voltage adaptation 1.65-3.3 V			***	***
Virtual COM port/ USB bridge		**	**	**
Debug/Program	*			

STLink/V2

STLink-V3SET
STLink-V3MODS

B-STLink-VOLT
STLink-V3MINIE

STLink-V3PWR

life.augmented

…to address a large diversity of needs

STLINK-V3SET

Easy addition of multi-path bridge and debug connector formats

Adapter board

USB to :
UART
SPI
CAN
I2C
:
GPIOs

Connector formats :
STDC-14
JTAG-20

life.augmented

STLINK-V3SET a scalable debugger/programmer

STLINK-V3SET

35$

Easier

- JTAG / SWD / SWV (STM32 debug)
- SWIM (STM8 debug)
- Drag and drop flash programming
- Virtual COM port

Faster

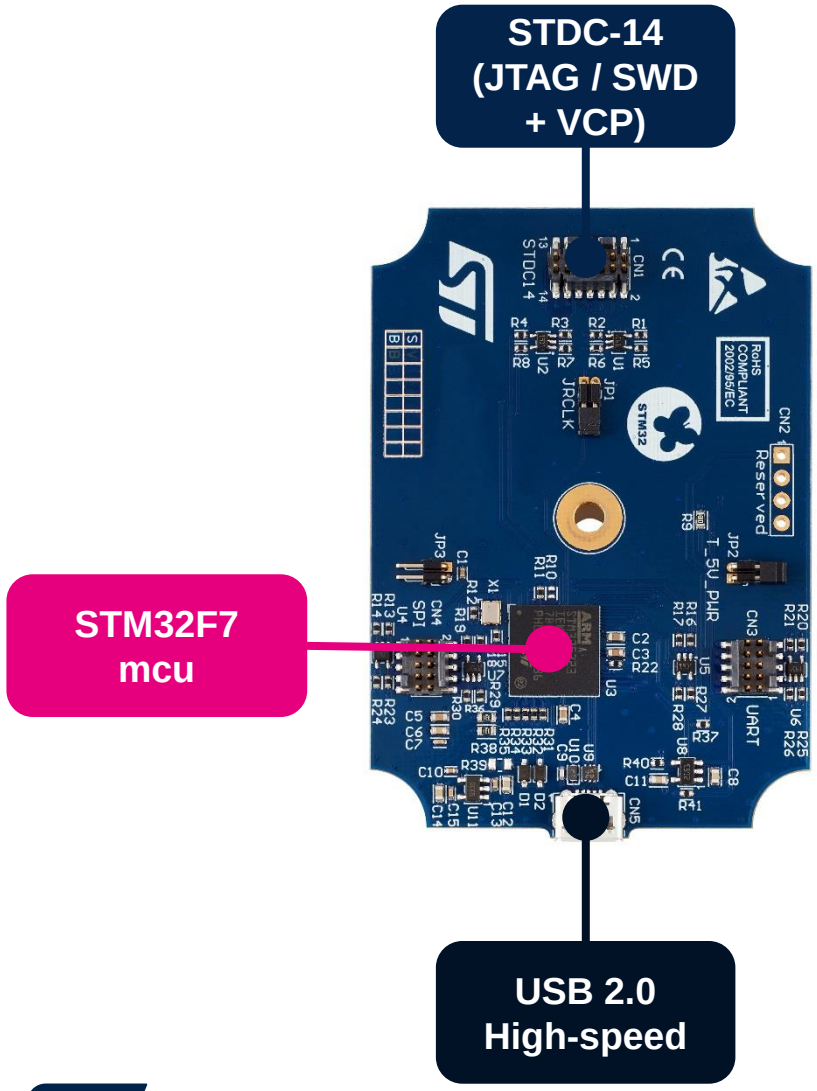
- Performance boost (vs STLink/V2)
- Optimized algorithms
- USB 2.0 High Speed interface

More flexible

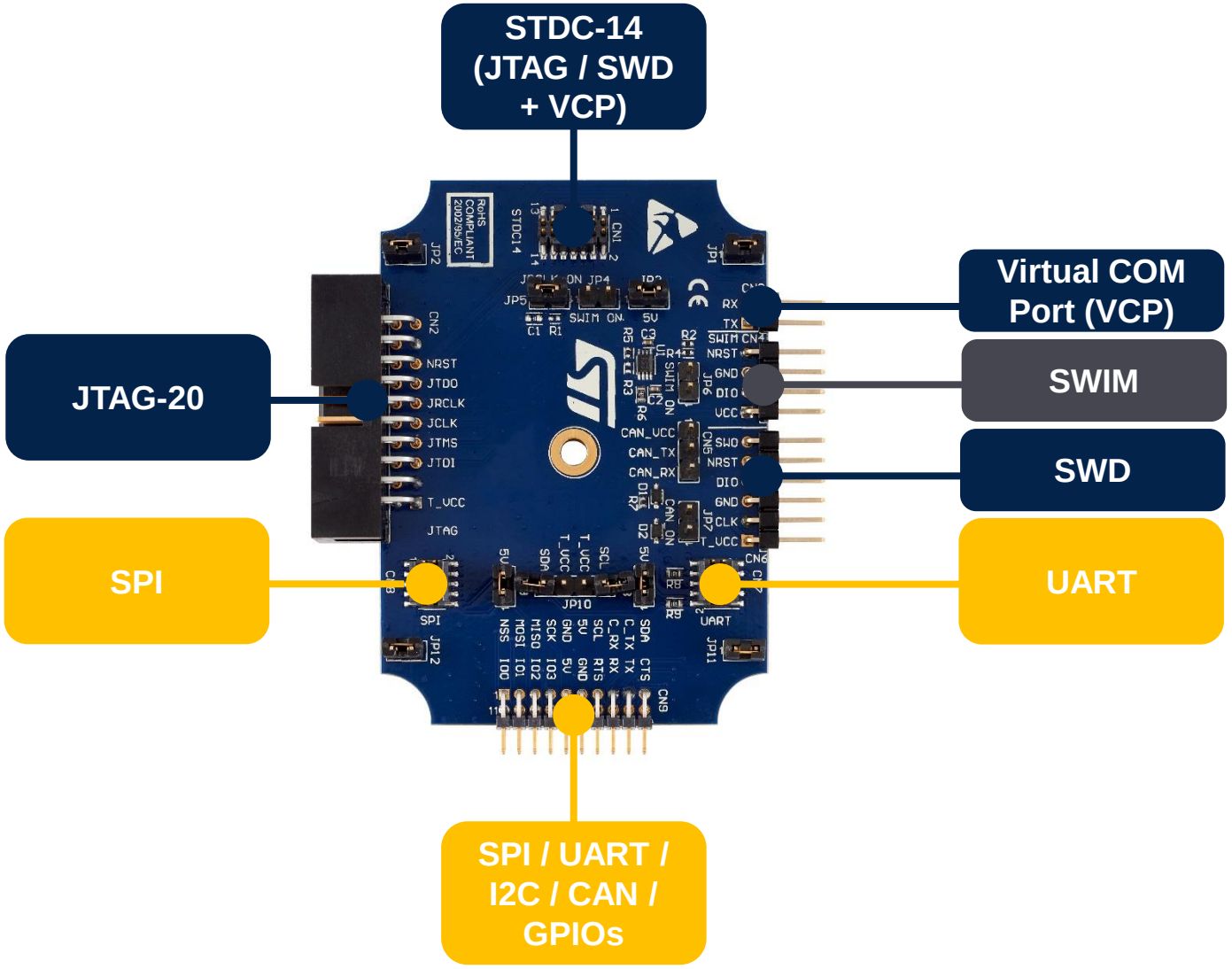
- Extension boards
- Multi-path bridge (through adapter board)

life.augmented

STLINK-V3SET anatomy



Base board



Adapter board

STLINK extension boards

B-STLINK-VOLT

20$

Voltage adaptation

- 1.65 – 3.3 V voltage adaptation for debug / virtual COM port / bridge signals
- Compatible with STLINK-V3SET casing

B-STLINK-ISOL

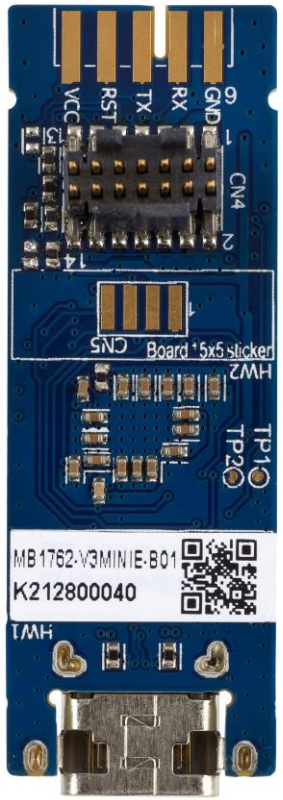
40$

Galvanic isolation & Voltage adaptation

- 1.65 – 3.3 V galvanic isolation and voltage adaptation for debug / virtual COM port / bridge signals
- Compatible with STLINK-V3SET casing

life.augmented

Small STLINK-V3MINIE makes big difference



STLINK-V3MINIE

11$

More performance at lower price (vs STLink/V2)

- JTAG / SWD / SWV debug interfaces
- USB Type-C High Speed interface

Facilitate in-the-field FW update

- Virtual COM port
- Tiny size 15 x 42 mm
- 3D printer files for slim casing

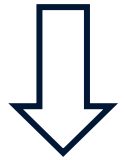
Direct support of power-constrained IoT products

- Voltage adaptation 1.65-3.3V

life.augmented

STLINK-V3MODS, a ready-to-use module for your custom boards

STLINK-V3SET



STLINK-V3MODS

8.25$

Performance boost (vs STLink/V2)

- JTAG / SWD /SWV debug interfaces
- USB 2.0 High Speed interface

Small size, small price

- Footprint 15 x 30 mm
- Price below 9$

Multi-path bridge

- Virtual COM port
- Drag and drop programming
- USB to UART / SPI / I2C / CAN / GPIOs

life.augmented

STLINK comparison

Features	STLink/ V2	STLink-V3MODS	STLink-V3MINIE	STLink-V3SET	B-STLink-ISOL	STLink-V3PWR
Mcus supported	STM32 STM8	STM32	STM32	STM32 STM8	STM32	STM32
Casing	*	(on-board module)	STL file	*	V3SET extension	*
USB	Full Speed (12Mbit/s)	High Speed (480Mbit/s)	High Speed (480Mbit/s)	High Speed (480Mbit/s)	V3SET extension	High Speed (480Mbit/s)
SWD max read data rate	150 kByte/s	800 kByte/s	800 kByte/s	800 kByte/s		475 kByte/s
Virtual COM port	No	16 MHz	16 MHz	16 MHz	10 MHz	12 MHz
Multi-path bridge	No	*	No	*	*	*
Target voltage	3.3V	3.3 V	1.65-3.3 V	3.3 V	1.65-3.3 V	1.6-3.6 V
Energy profiler						*
Price	21$	8.25$	11$	35$	40$	95$

Software support and Product references

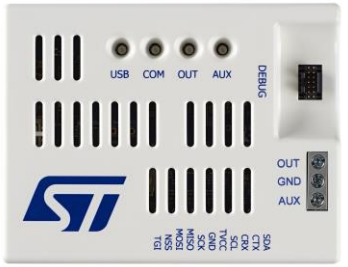
STLINK-V3MINIE / V3MODS
STLINK-V3SET / V3PWR

STM32 CubeMonitor	STM32CubeMonitor STM32CUBEMON
STM32 CubeProgrammer	STM32CubeProgrammer STM32CUBEPROG
STM32 CubeIDE	STM32CubeIDE
arm KEIL	IDE toolset MDK-ARM
iar	IDE toolset EWARM

life.augmented

STLink-V3Power energy profiling tool

…Enables energy profiling of any STM32 MCU



Dvt tools	Power shield	Energy meter	STLINK-V3 Power
Dynamic current ranges	100nA / 50mA	300nA / 150mA	**nA up to 500mA**
Target MCUs	STM32L0, L4 STM32WB	STM32L0, L4, STM32L5, U5, STM32WB, WL	**All STM32 MCUs**
	X-NUCLEO-LPM01A	STM32L562E-DK	STLINK-V3PWR

life.augmented

Optimize the energy efficiency of STM32 mcu applications

- Measure dynamic current consumption with high accuracy
- Visualize energy consumption with STM32CubeMonitor-Power SW tool
- Debug code in sync with energy consumption measurements
- Direct support of Keil and IAR IDEs for power profiling
- Programmer with multi-path bridge

STLINK-V3PWR

95$

life.augmented

STLINK-V3PWR powerful easy-to-use energy profiler

Current measurement with wide dynamic range
nA-500mA

High accuracy (up to +/-0.5%)
Resolution up to 2nA

Programmable output voltage source
1.6 - 3.6V (under up to 2A)

life.augmented

STLINK-V3PWR overview

USB Type-C
high-speed interface
-Simultaneous access of CubeMonitor-Power and IDEs
-Fast programming with CubeProgrammer

Debugger / Programmer
SWD / JTAG (STDC14 connector)
Virtual COM port

Multi-path bridge
USB to
UART / SPI / I2C /
CAN / GPIOs

Debug code in sync with power consumption measurements
Programmable voltage supply for target

Visualize power data with STM32CubeMonitor-Power tool

Display

- Graphical rendering in real-time (up to 100 kSPS)
- Acquisition log over large period of time

Analyze

- Intuitive zoom and navigation into energy consumption data

Benchmark

- Fast computation of EEMBC ULPMark-CP scores

Software support and Product references

	STLINK-V3PWR
STM32 CubeMonitor-Power	STM32CubeMonitor-Power STM32CUBEMONPWR — From release v1.2
STM32 CubeProgrammer	STM32CubeProgrammer STM32CUBEPROG — From release v2.13
STM32 CubeIDE	STM32CubeIDE — From release v1.12
arm KEIL	IDE toolset MDK-ARM — From release v5.38a
iar	IDE toolset EWARM — From release v9.32.2

life.augmented

Summary slide

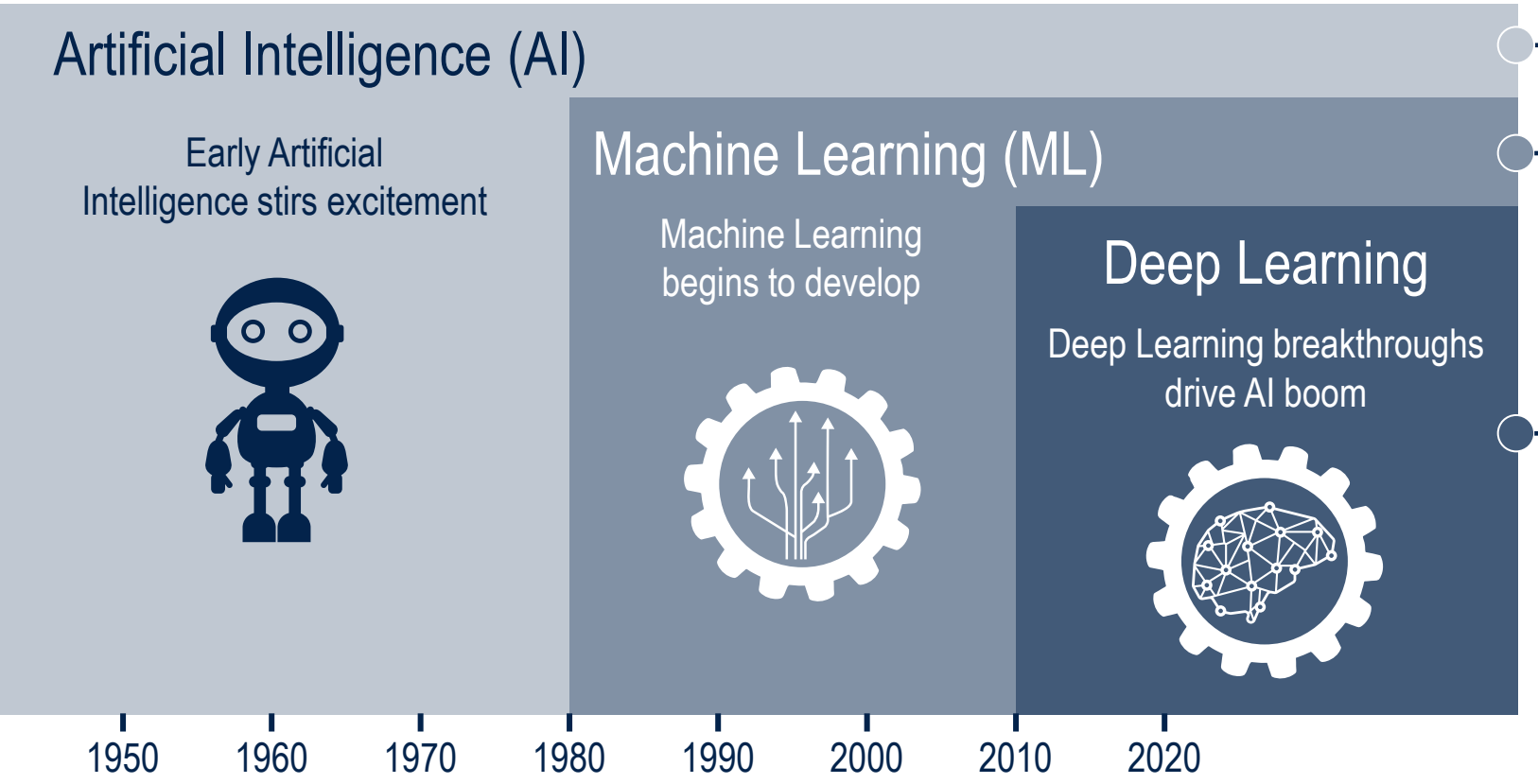
	Configuration	Code generation	Code editor	Debug	Simple Programming	Advanced programming	Monitoring
STM32 CubeMX	Main feature	Main feature					
STM32 CubeIDE	Available thanks to embedded STM32CubeMx Features	Available thanks to embedded STM32CubeMx Features	Main feature	Main feature	Programming of the flash		Limited monitoring via SWD + IDE feature
STM32 CubeProgrammer					Main feature	Main feature Please refer to the slides for details	
STM32 CubeMonitor							Main feature Advanced monitoring

life.augmented

AI solutions

What is AI?

The evolution of AI



Any technique that enables computer to mimic **human behavior**

Subset of AI. Algorithms and methodologies that improve over time through **learning from data**

Subset of ML. Learning algorithms that derive meaning from a **huge amount of data**, by using a hierarchy of multiple layers **that mimic the neural networks of the human brain**

life.augmented

A new way to add environment awareness to your products

Create more robust software using Machine Learning on STM32

Standard programming

Handcrafted rules based on experience

Input Data + Procedural algorithm

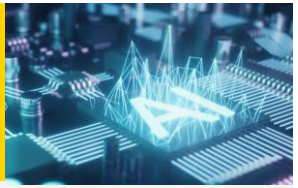
↓

Desired output from the system

- Requires domain expertise to code
- Need to rewrite if environment evolves

Machine Learning

Rules learnt from real-world data



Input Data + Desired output from the system

↓

General ML model trained for the specific problem

- Generate code from real-world observations
- Re-learn from data if environment evolves

life.augmented

AI is used today in almost every market segment

Drones
Flying & landing

Smart Industry
Predictive maintenance

Smart Home
Event detection

Smart Farming
Optimizing conditions

Personal Healthcare
Body measurements

White goods
Smart control

Self-driving Cars
Environment sensing

Smart Buildings
Energy saving

Robots
Collision detection

Personal electronics
Activity recognition

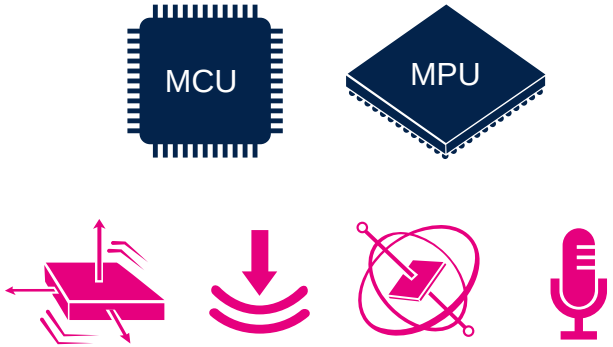
life.augmented

Embedded AI technology trend

“Global Shipments of Deep Edge AI Devices to Reach 2.5 Billion by 2030”

Source: ABI Research

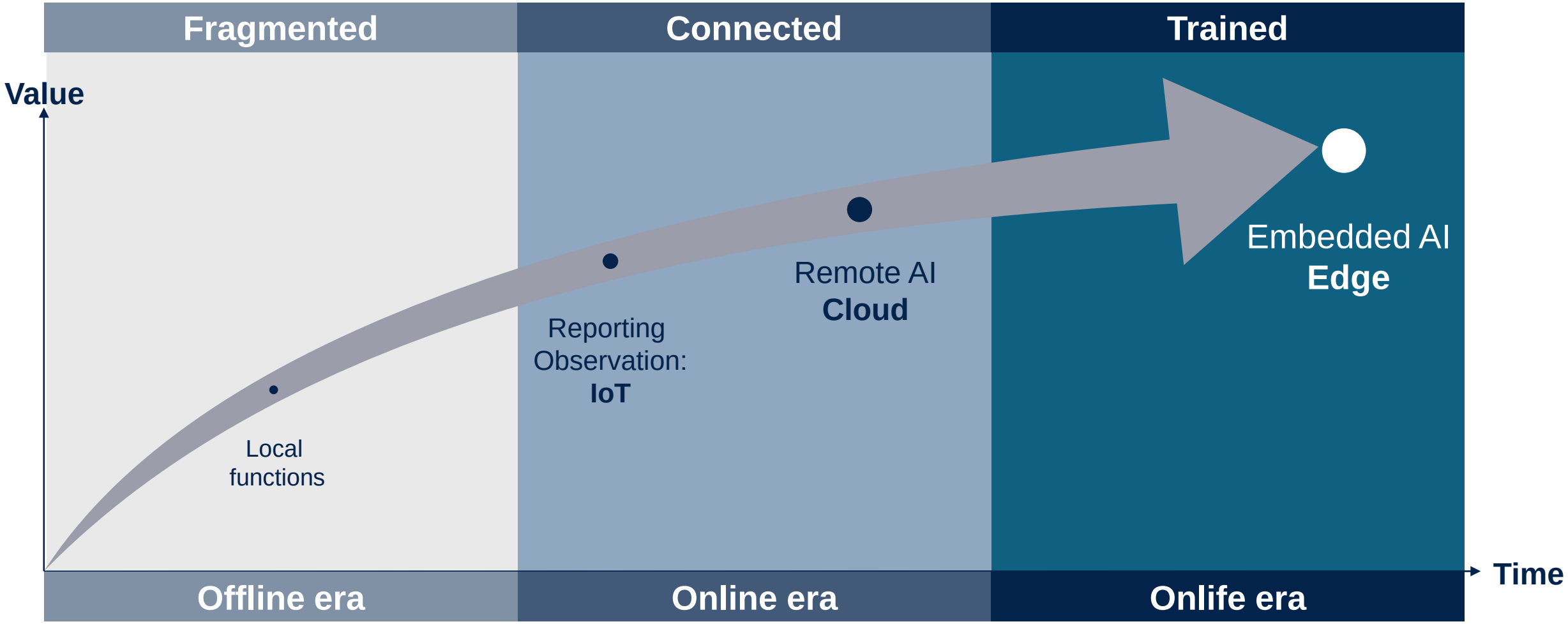
AI technologies are now embedded inside end devices (MPU, MCU and sensors).



Growing community and ecosystem of **Deep Edge AI** technologies focusing on standalone, low-power and cost-efficient embedded solutions.

life.augmented

The quest for an ever-SMARTER infrastructure



life.augmented

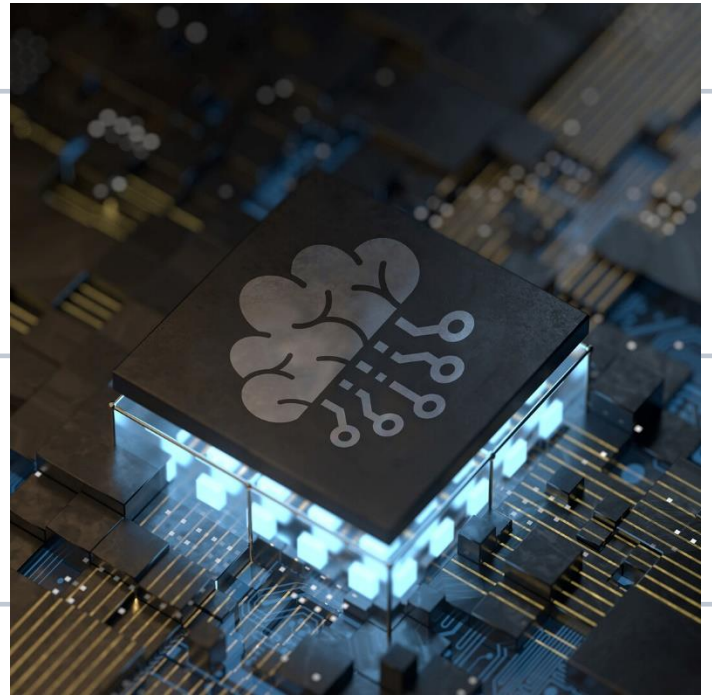
Artificial intelligence at the Edge

Moving part of Artificial Intelligence closer to the data acquisition brings several benefits

Ultra-low latency
Real-time applications

More reliability

Security of data
No sharing in the cloud



Privacy by design
GDPR compliant

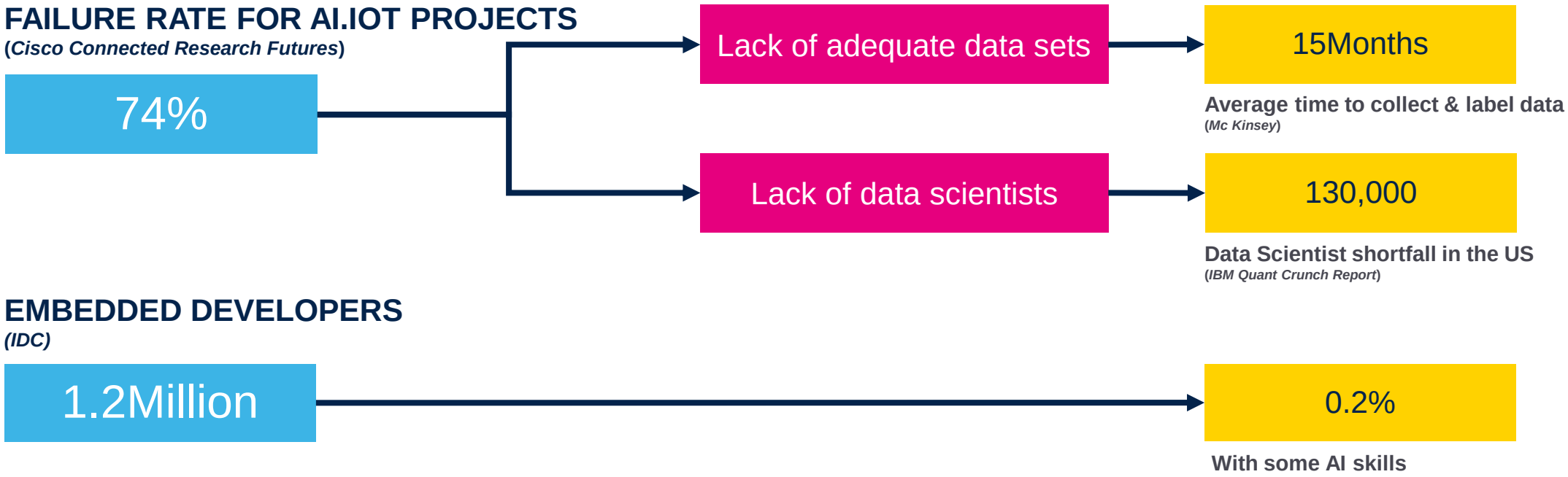
Sustainable on energy
Low-power consumption

Better user experience

life.augmented

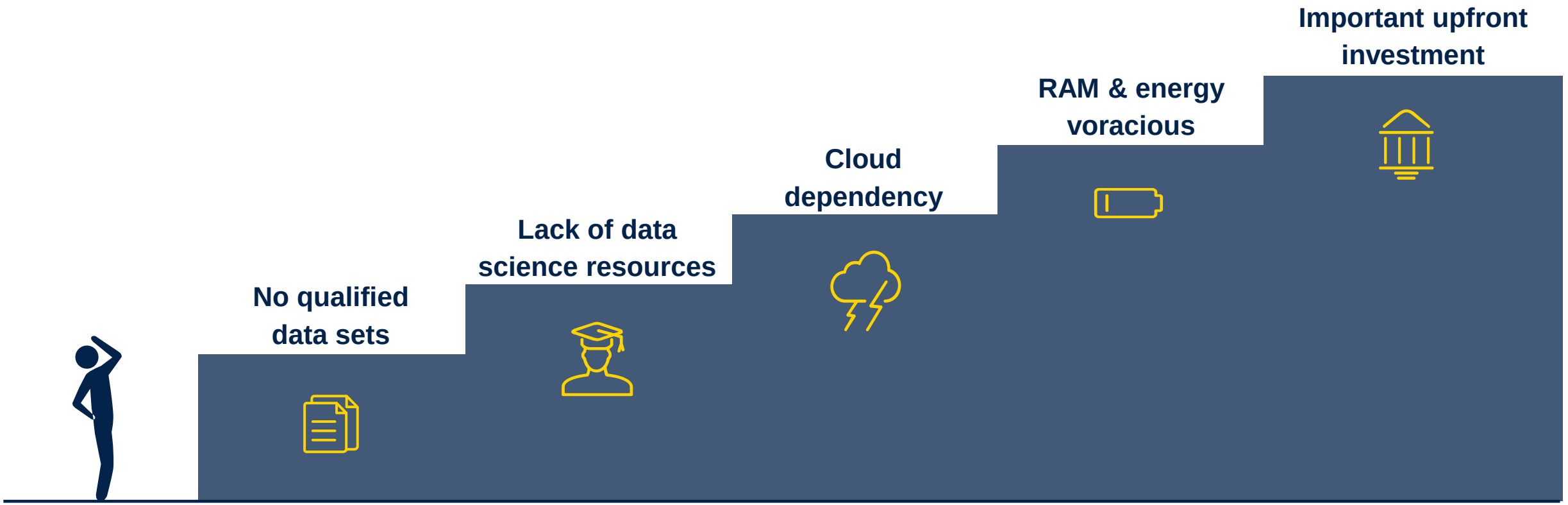
A step too big to climb

Organizations main challenges – AI Skills & datasets



For most companies, creating an edge AI device is a long journey with extraordinary challenges

Investment, complexity and development time are often barriers to AI adoption



life.augmented

Start today with deep edge AI

> If only
> I had solutions to overcome
> AI design challenges

This is where we come in

life.augmented

Whatever your company's profile, you will find an AI solution suited to your needs

COMPANY'S PROFILE	USE CASES		
	Anomaly detection	Classification	Deep Learning
Embedded developers ▪ No dataset available ▪ No dedicated AI Team	NANOEDGE AI STUDIO	NANOEDGE AI STUDIO	Engineering Services STM32 Cube.AI
Team with AI expertise ▪ Dataset available ▪ AI Team		NANOEDGE AI STUDIO STM32 Cube.AI	STM32 Cube.AI

life.augmented

STM32 comprehensive AI ecosystem

Applications

Applicative Examples
(Function Packs)

Pre and post Processing libraries

Frameworks

NANOEDGE AI STUDIO

scikit learn

MATLAB

PyTorch

TensorFlow Lite

ONNX

Keras

STM32 Cube.AI

AI Model convertor

- Graph optimizer
- Quantizer
- Memory optimizer

Edge Hardware

STM32 series

Discovery kit

STM32 Nucleo board

Camera add-on

life.augmented

A tool to seamlessly integrate AI in your projects

Machine Learning

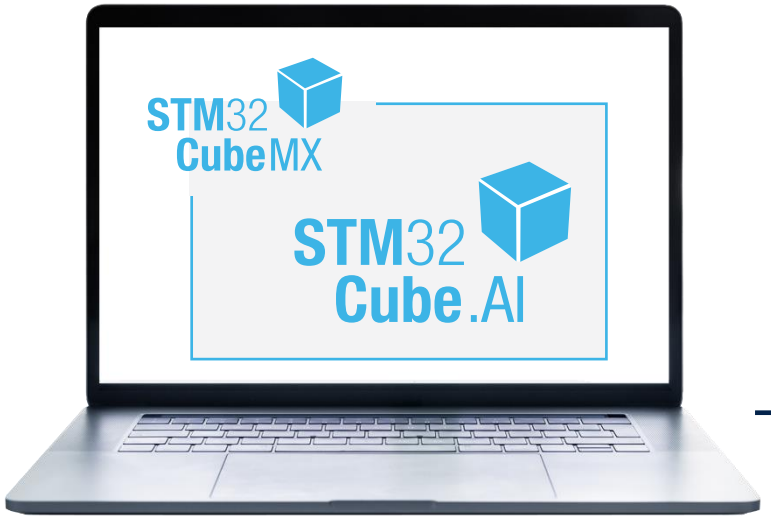
scikit learn ONNX

Deep Learning

MATLAB PyTorch

TensorFlow Lite

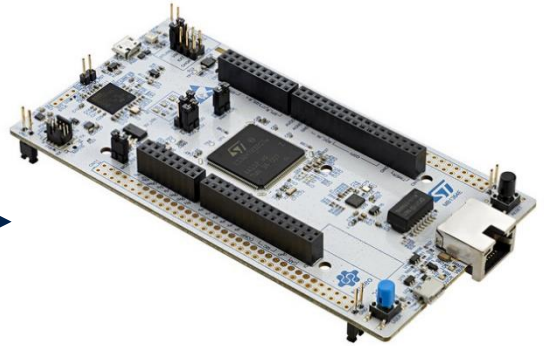
Keras ONNX



Select MCU & upload your model

Optimize and validate

Generate project and deploy



HIGH PERFORMANCE EDGE AI PRODUCT

life.augmented

For embedded developers

NanoEdge™ AI Studio, an automated ML design solution

NANOEDGE AI STUDIO

A unique solution thought from scratch

HOW DID WE DO IT?

We re-wrote, from the algebra, ML and signal processing algorithms so that they can **LEARN** and **INFER** inside STM32

- Patented technology
- Designed for embedded developers
- Ultra memory efficient (Flash and RAM)
- Unsupervised learning in the device
- No dataset need
- Superior security
- Small footprint, runs on any STM32 microcontroller
- Close to 100% accuracy and confidence
- For anomaly detection, classification and extrapolation problems

For teams with AI expertise

STM32Cube.AI helps you accelerate your embedded development

STM32 Cube.AI

Easily evaluate, convert and deploy Machine Learning and Deep Neural Networks on STM32

An AI extension integrated with STM32Cube MCU development environment to **OPTIMIZE** and **TUNE** models, directly on target

- Develop and train your model with major AI frameworks

TensorFlow Lite · K · scikit learn · ONNX · MATLAB · PyTorch

- Best ML performance on STM32 (MLPerf™ Tiny benchmarks)
- Validate performance directly on target
- Small footprint, runs on any STM32
- To address any kind of problem with ML or CNN libraries

life.augmented

Making Edge AI accessible to all STM32 portfolio

STM32Cube.AI & NanoEdge AI are compatible with all STM32 series

MPU						**STM32MP1** 4158 CoreMark Up to 800 MHz Cortex –A7 209 MHz Cortex –M4
High Perf MCUs	STM32**F3** 245 CoreMark 72 MHz Cortex-M4 Optimized for mixed-signal Applications	**STM32G4** 569 CoreMark 170 MHz Cortex-M4	STM32**F2** Up to 398 CoreMark 120 MHz Cortex-M3	STM32**F4** Up to 608 CoreMark 180 MHz Cortex-M4	STM32**F7** 1082 CoreMark 216 MHz Cortex-M7	**STM32H7** Up to 3224 CoreMark Up to 550 MHz Cortex -M7 240 MHz Cortex -M4
Mainstream MCUs	STM32**F0** 106 CoreMark 48 MHz Cortex-M0	**STM32G0** 142 CoreMark 64 MHz Cortex-M0+	STM32**F1** 177 CoreMark 72 MHz Cortex-M3			
Ultra-low Power MCUs	STM32**L0** 75 CoreMark 32 MHz Cortex-M0+	STM32**L1** 93 CoreMark 32 MHz Cortex-M3	STM32**L4** 273 CoreMark 80 MHz Cortex-M4	STM32**L4+** 409 CoreMark 120 MHz Cortex-M4	STM32**L5** 443 CoreMark 110 MHz Cortex-M33	**STM32U5** 651 CoreMark 160 MHz Cortex-M33
Wireless MCUs			**STM32WL** 162 CoreMark 48 MHz Cortex-M4 48 MHz Cortex-M0+	**STM32WB** 216 CoreMark 64 MHz Cortex-M4 32 MHz Cortex-M0+		

Latest product generation

life.augmented

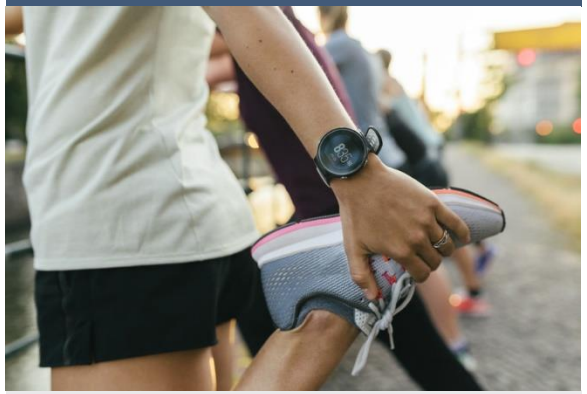
Integrate your ML models more easily with our application-oriented code examples

Time series-based monitoring

FP-AI-MONITOR1

- Predictive maintenance and much more sensor-monitoring apps
- Runs Libraries from NanoEdge™ AI Studio

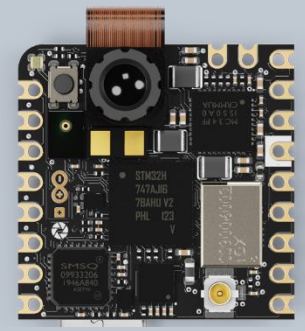
Audio and Sensing



FP-AI-SENSING1

- Human Activity Recognition
- Acoustic Scene Classification
- Data logging, labeling and result on BLE applications

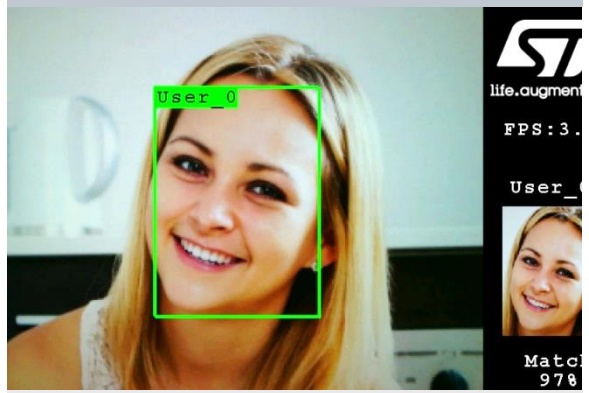
Computer Vision



FP-AI-VISION1

- Food recognition (CNN)
- Person presence detection (CNN)
- People counting (Object detection NN)
- Image processing Library

Face recognition



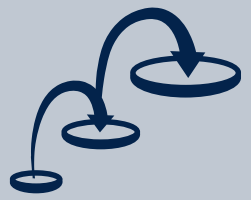
FP-AI-FACEREC1

- Face detection and recognition
- Fully functional without cloud connection

life.augmented

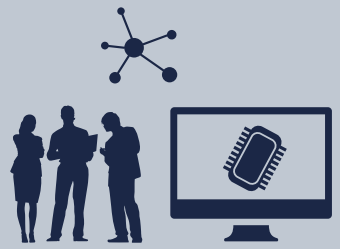
We provide everything to kick off your project

Design documentation



Getting started

Be guided step-by-step to learn STM32 ecosystem

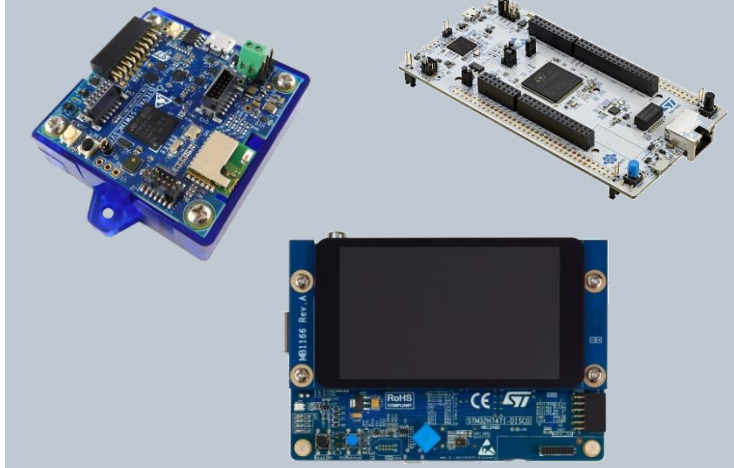


Development zone

Get started on application development and project sharing

- **Wiki by ST** is a great forum to learn and start developing AI on STM32!
- Videos of application examples
- Massive Open Online Course (MOOC)

Hardware and software tools



- Evaluation platforms for STM32 MCU/MPU
- Extra sensor boards
- Full software suite

Support & Updates

- **ST Community**: STM32 ML & AI group
- Competence center
- Distributor certified FAE
- Support center
- Newsletter

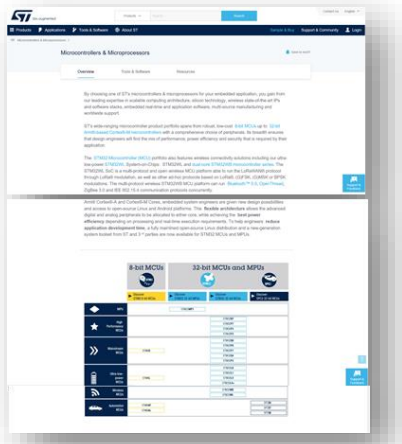
Stay focused on your expertise, we bring you everything else

life.augmented

Information and sharing: ST worldwide

Get connected to STM32 world!

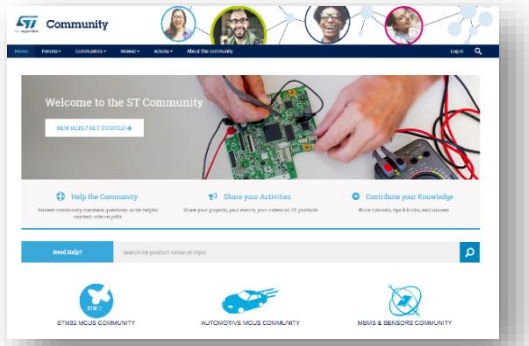
st.com/stm32



ST MCU Finder

community.st.com

github.com/STMicroelectronics



facebook.com/stm32

youtube.com/STonlineMedia

twitter.com/@ST_World

linkedin.com/stmicroelectronics

21IC

YouKu

Information **Product Selector** **Community/GitHub** **Social Media**

►STM32 Education is now available here (MOOC, OLT, Textbooks, Training Courses)

www.st.com/stm32education

Our technology starts with You

Find out more at www.st.com

life.augmented