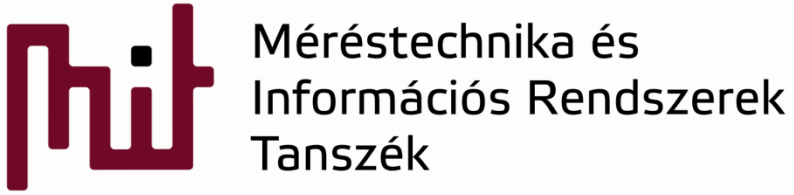
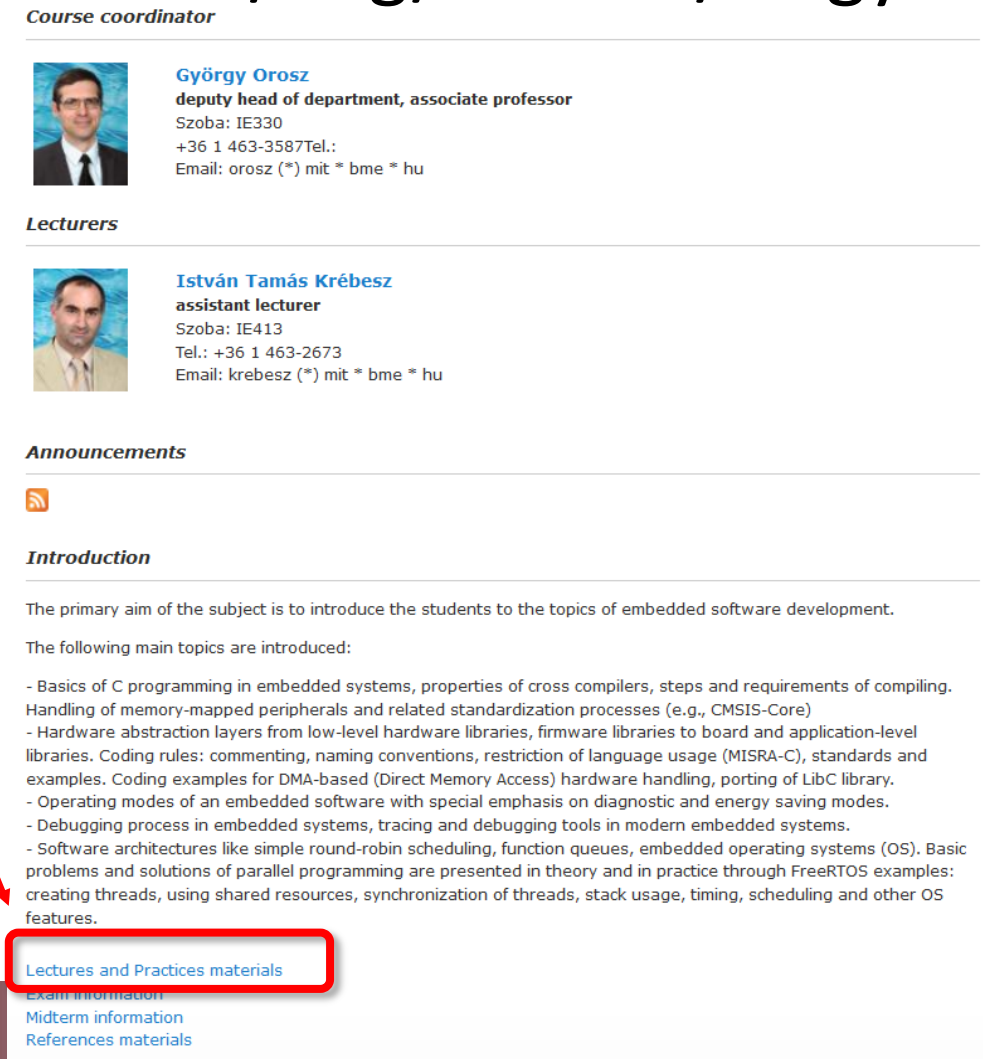
Embedded and ambient systems 2024.09.04.

Practice 1



Preliminary

- Check the web site of the course: www.mit.bme.hu/eng/oktatas/targyak/vimiac17
- See menu at the bottom



Course coordinator

György Orosz
deputy head of department, associate professor
Szoba: IE330
+36 1 463-3587Tel.:
Email: orosz (*) mit * bme * hu

Lecturers

István Tamás Krébesz
assistant lecturer
Szoba: IE413
Tel.: +36 1 463-2673
Email: krebesz (*) mit * bme * hu

Announcements

Introduction

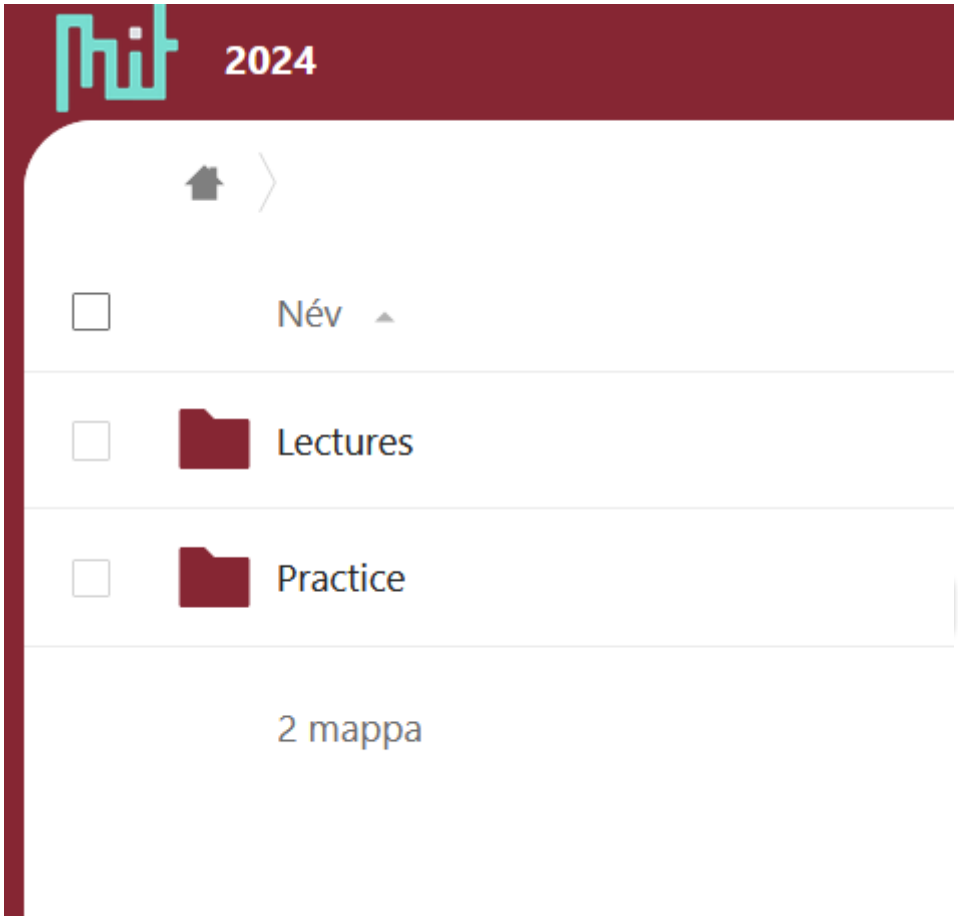
The primary aim of the subject is to introduce the students to the topics of embedded software development.

The following main topics are introduced:

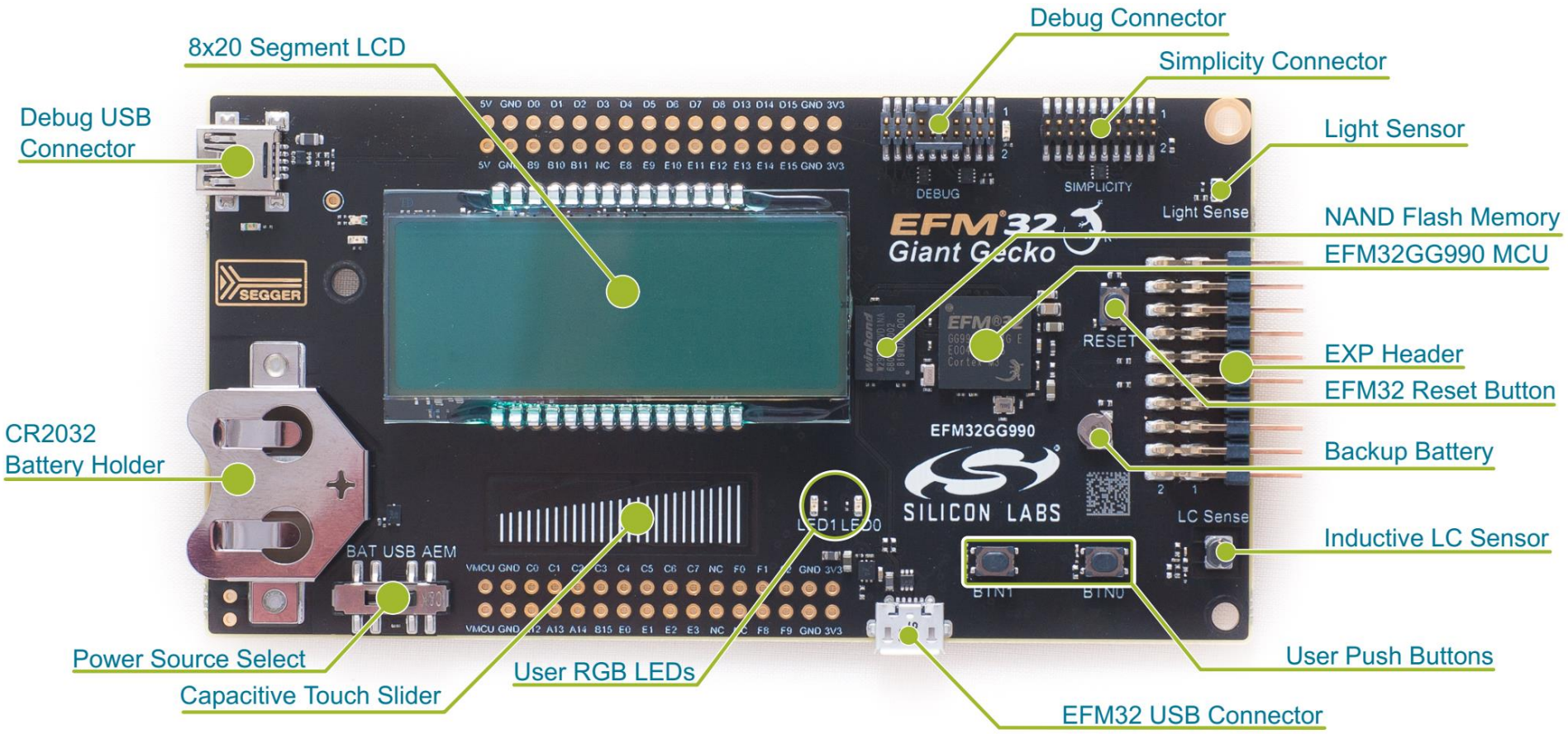
- Basics of C programming in embedded systems, properties of cross compilers, steps and requirements of compiling. Handling of memory-mapped peripherals and related standardization processes (e.g., CMSIS-Core)
- Hardware abstraction layers from low-level hardware libraries, firmware libraries to board and application-level libraries. Coding rules: commenting, naming conventions, restriction of language usage (MISRA-C), standards and examples. Coding examples for DMA-based (Direct Memory Access) hardware handling, porting of LibC library.
- Operating modes of an embedded software with special emphasis on diagnostic and energy saving modes.
- Debugging process in embedded systems, tracing and debugging tools in modern embedded systems.
- Software architectures like simple round-robin scheduling, function queues, embedded operating systems (OS). Basic problems and solutions of parallel programming are presented in theory and in practice through FreeRTOS examples: creating threads, using shared resources, synchronization of threads, stack usage, timing, scheduling and other OS features.

Lectures and Practices materials
Midterm information
References materials

Preliminary



1) Development board: EFM32GG-STK3700

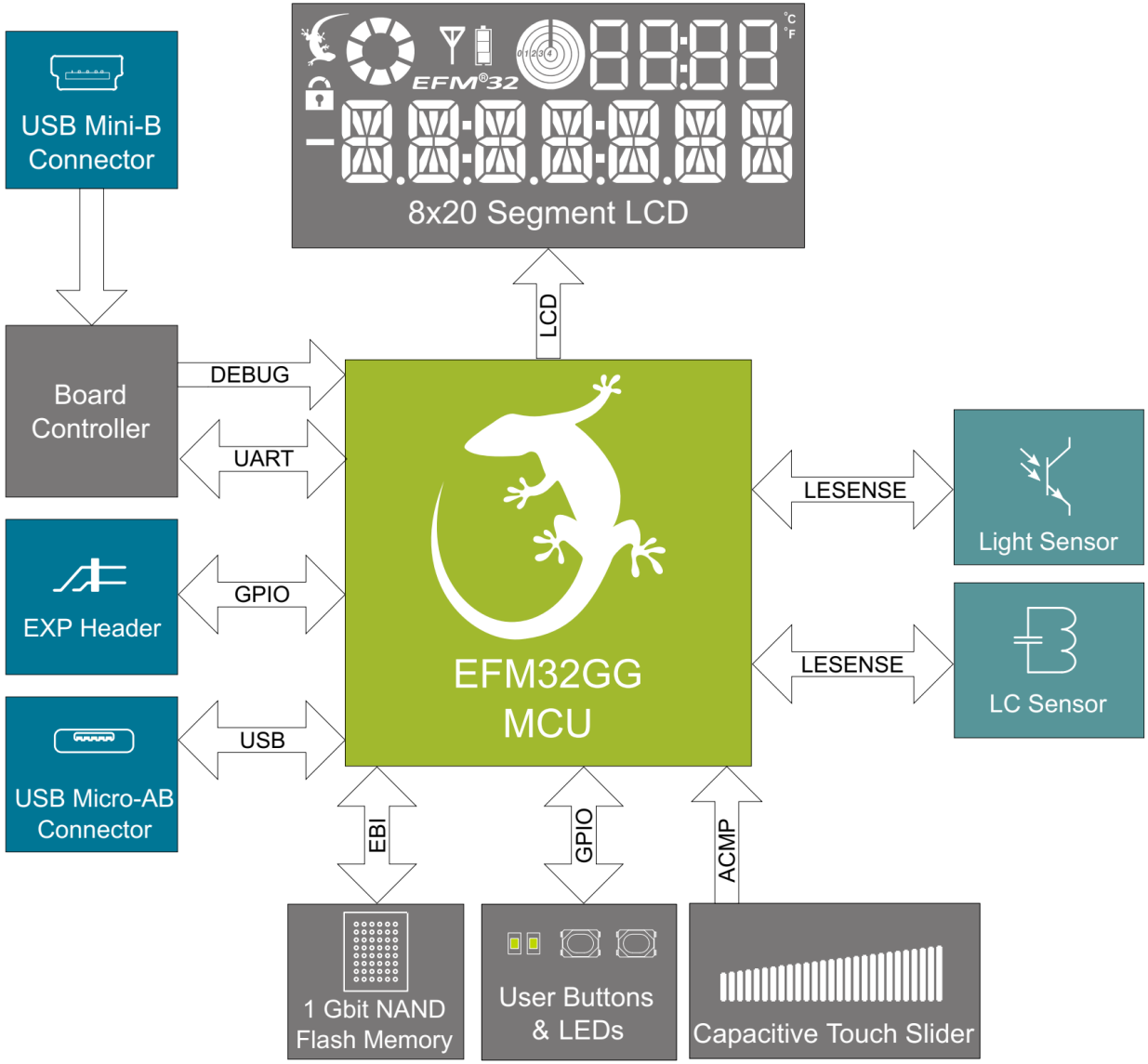


- https://www.silabs.com/development-tools/mcu/32-bit/efm32gg-starter-kit

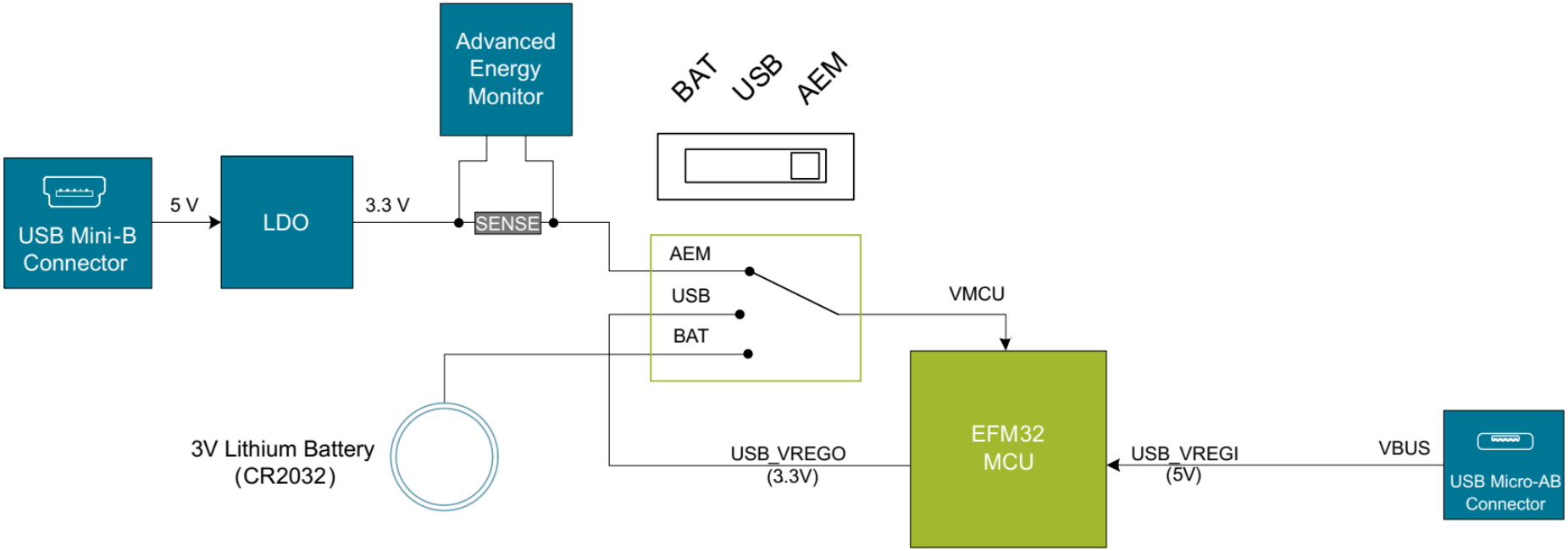
1.1) Main features

- EFM32GG990F1024 MCU with 1 MB Flash and 128 KB RAM.
- Advanced Energy Monitoring system for precise current tracking.
- Integrated Segger J-Link USB debugger/emulator with debug out functionality.
- 160 segment Energy Micro LCD.
- 20 pin expansion header.
- Breakout pads for easy access to I/O pins.
- Power sources include USB and CR2032 battery.
- 2 user buttons, 2 user LEDs and a touch slider.
- Ambient Light Sensor and Inductive-capacitive metal sensor.
- EFM32 OPAMP footprint.
- 32 MB NAND Flash.
- USB Micro-AB (OTG) connector.
- 0.03F Super Capacitor for backup power domain.
- Crystals for LFXO and HFXO: 32.768kHz and 48.000MHz.

1.2) Block diagram

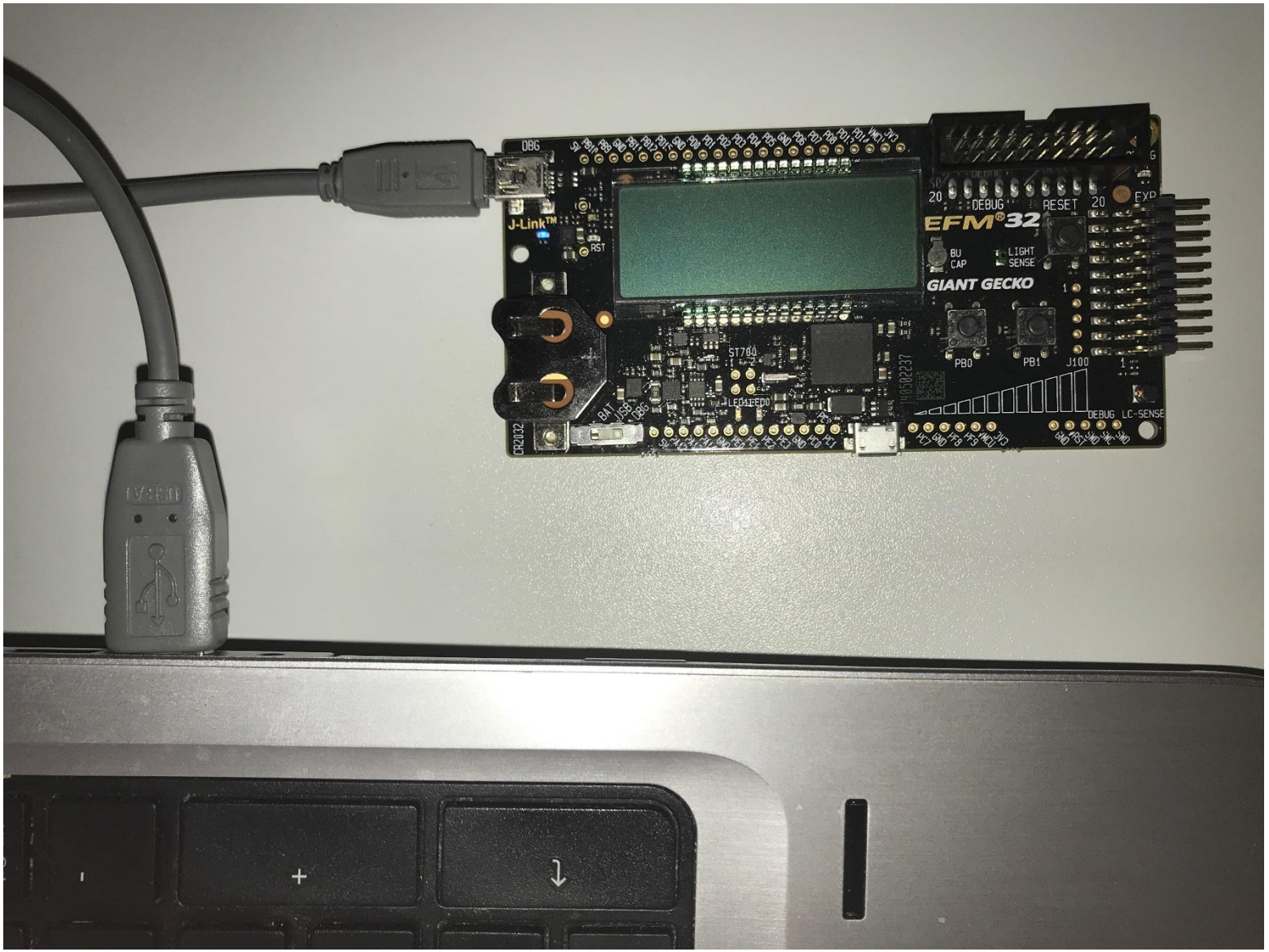


1.3) Power supply

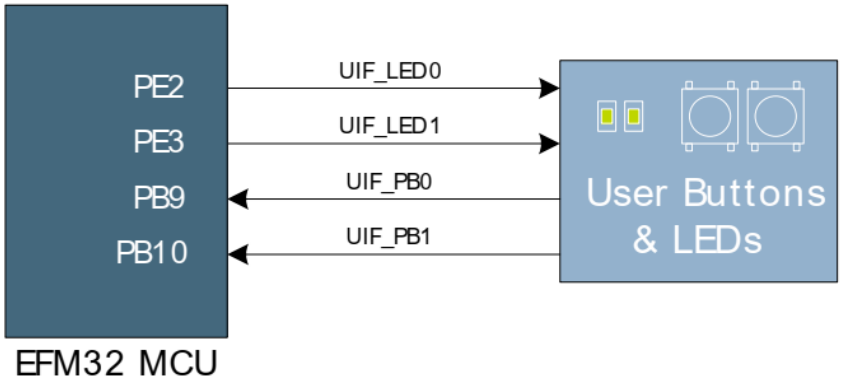


- DBG: via on-board debugger energy monitor can be used (use this)
- BAT: use CR2032 battery
- USB: MCU integrated voltage regulator is used

1.3) Power supply and proper connection

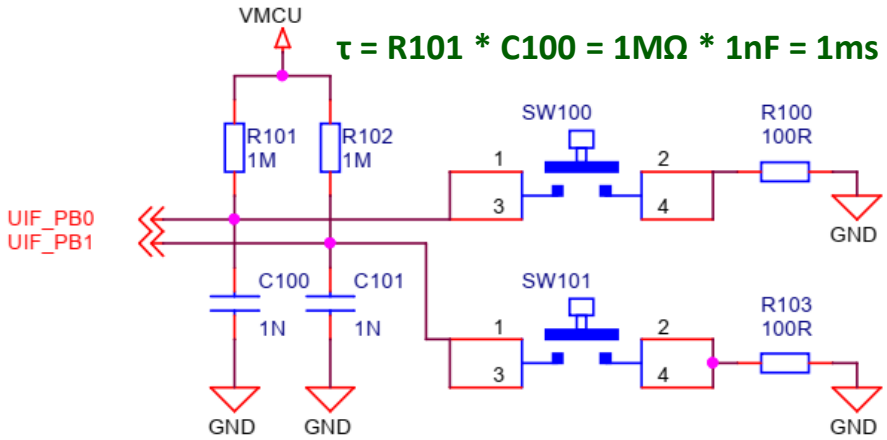


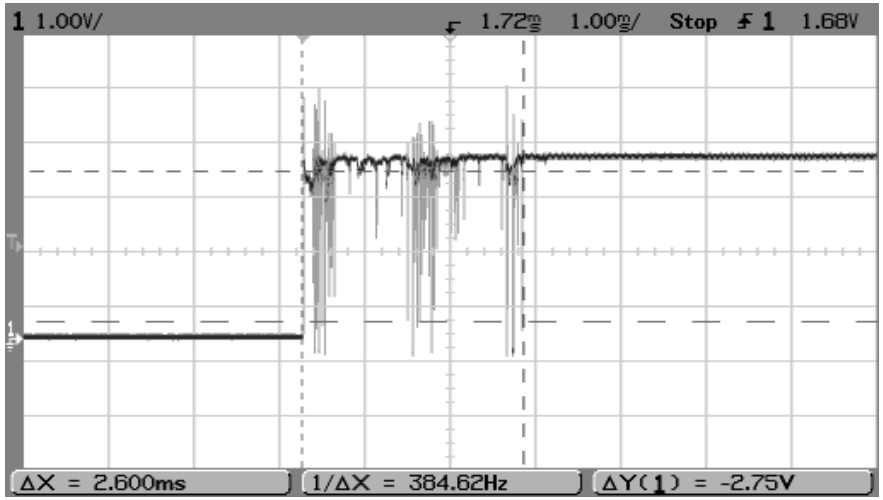
1.4) Peripherals-Buttons/LEDs



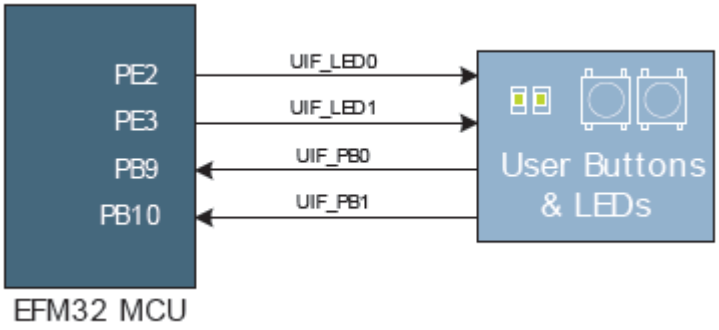
PB0=push button nr. 0
PB9=9th bit of port B
PE3=3rd bit of port E

- Push buttons are debounced by RC filter to avoid bouncing:

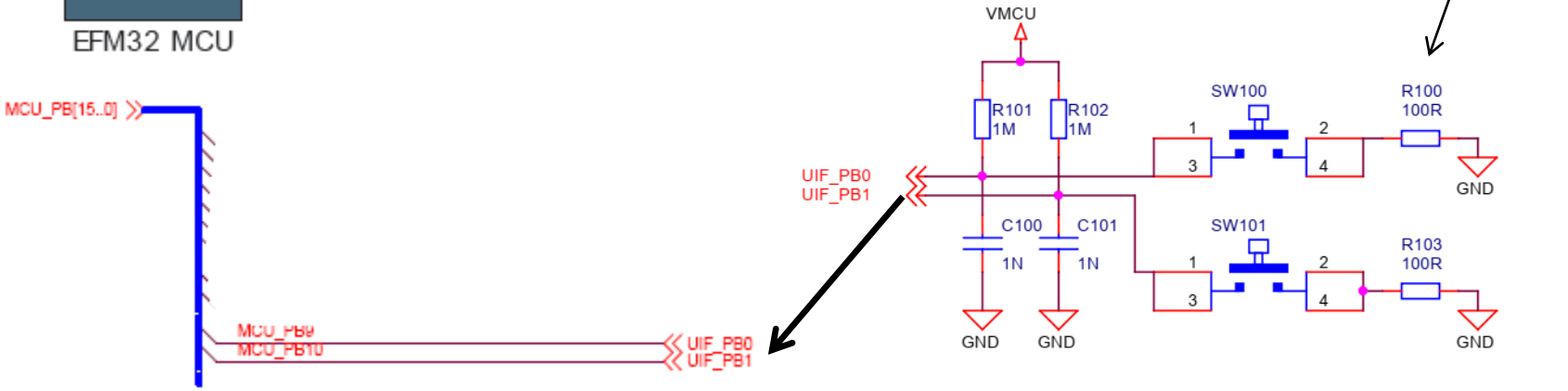




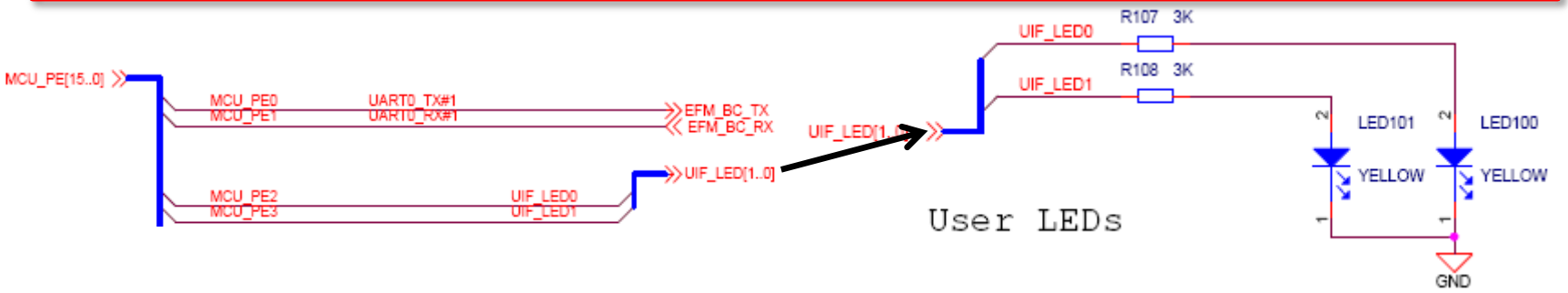
1.4) Peripherals-Buttons/LEDs



Overcurrent protection against the case when PB9 or PB10 pins are set to output by accident



LED shunt resistor: (3.3V-2V)/3kΩ≈0.4mA (note: 2V is the typical forward voltage of the LED)
Approx. 1mA…10mA current and 1.5…2.2V is expected on a LED



1.5) Board Controller

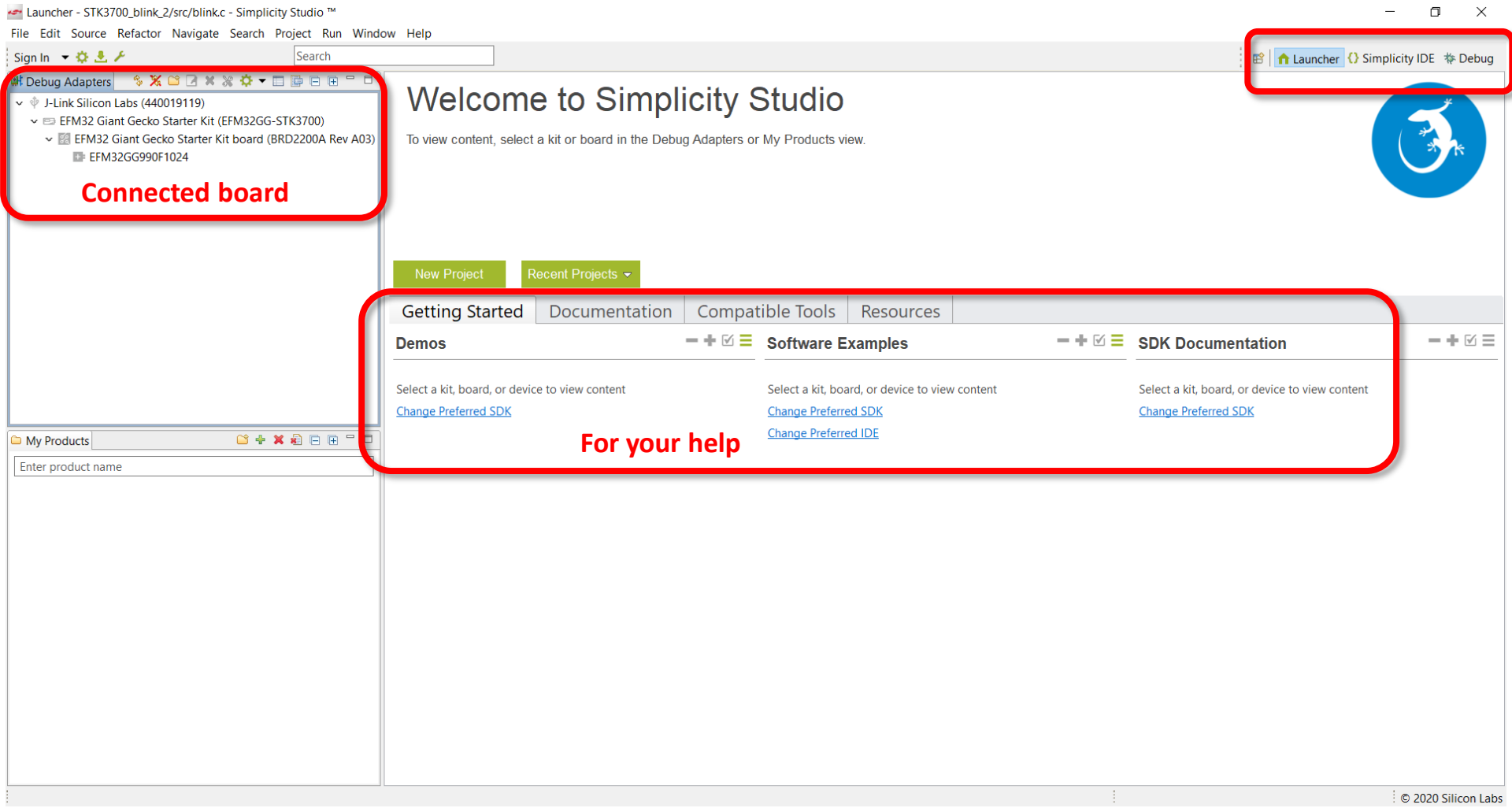
- Responsible for controlling board level tasks like debugger and Advanced Energy Monitor
- Interface is provided between the EFM32 and the board controller in the form of a UART connection
  - Set the EFM_BC_EN (PF7) line high
  - Use the lines
    EFM_BC_TX (PE0)
    and
    EFM_BC_RX (PE1)
- Board Support Package (bsp) is to be installed

2) Integrated Development Environment

- Integrated development environment (IDE): Simplicity Studio 4
- www.silabs.com/products/development-tools/software/simplicity-studio

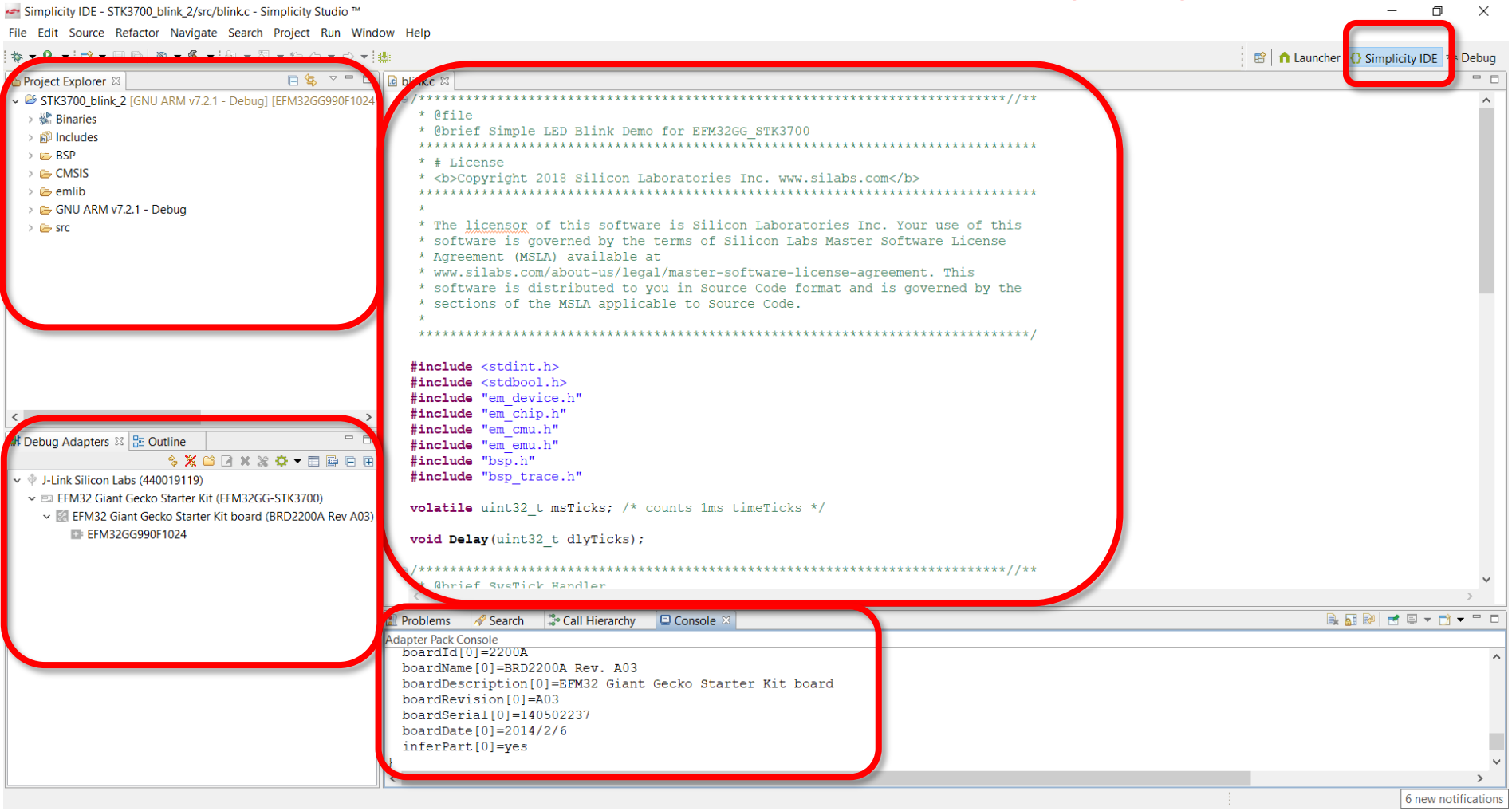
2.1) Getting started with IDE-Launcher

Views (Launcher is now active)



2.2) Getting started with IDE-Simplicity IDE

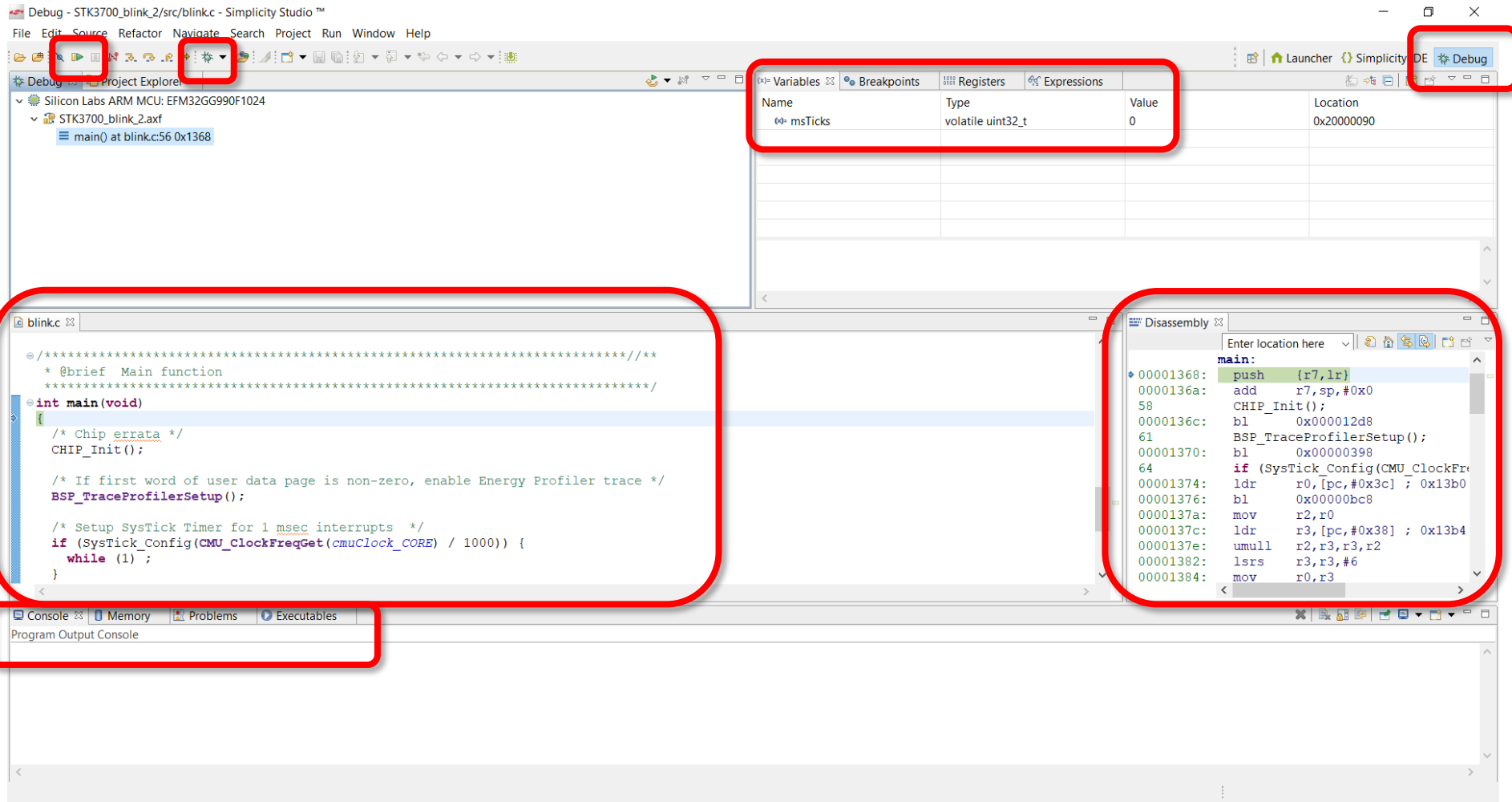
View (Simplicity IDE is active)



2.3) Getting started with IDE-Debug

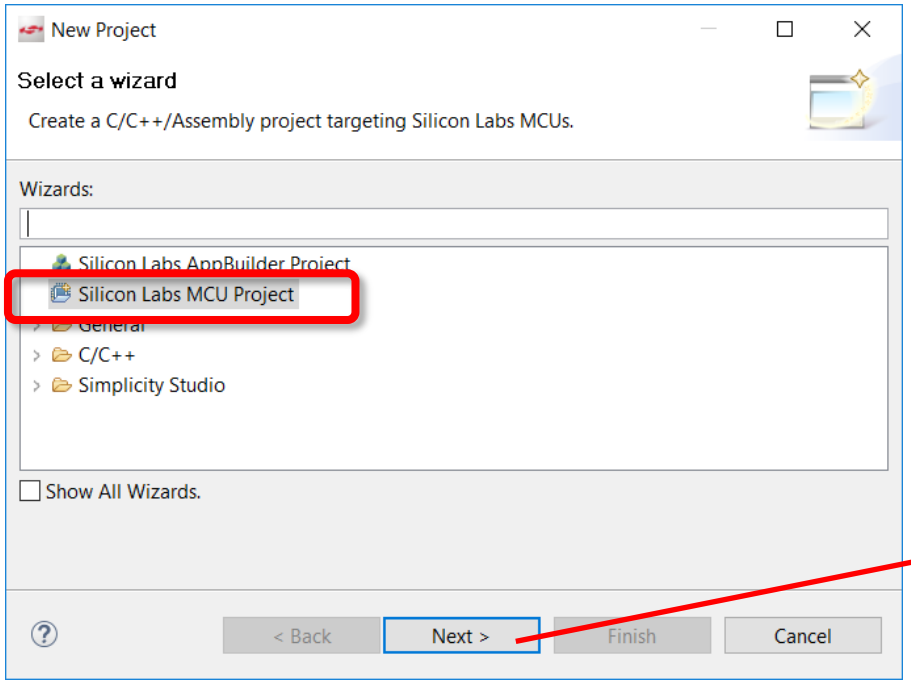
Run **Debug deploy and run**

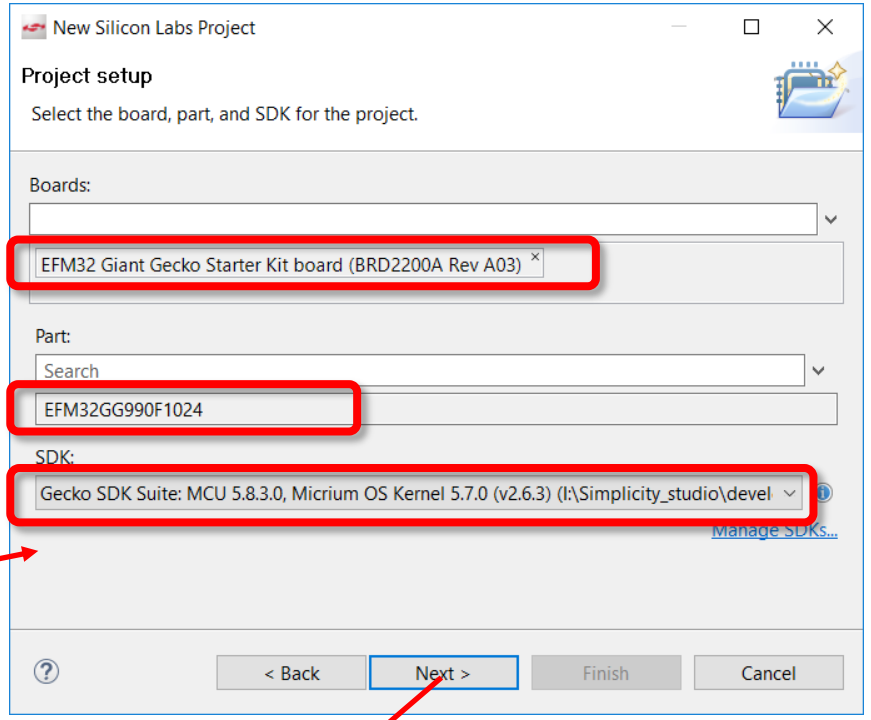
View (Debug is active)



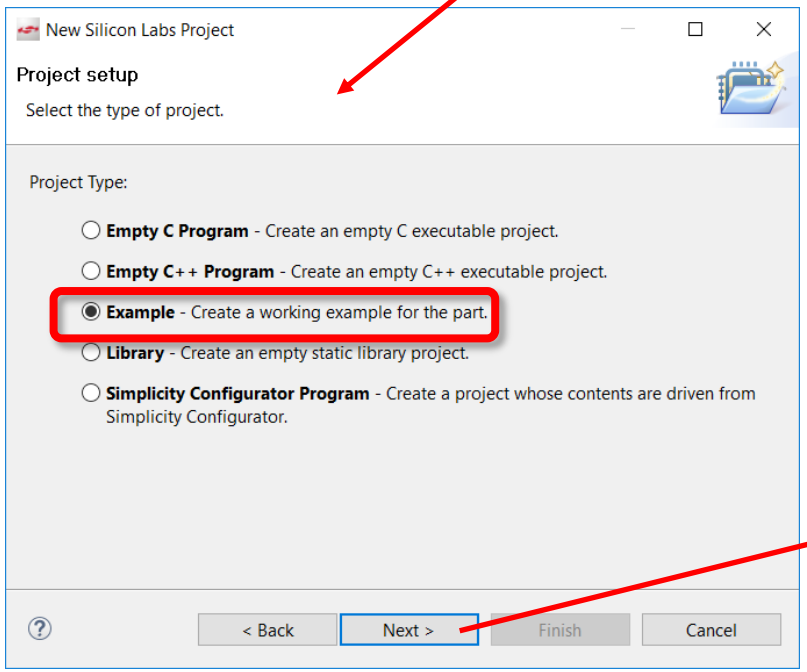
3) Start a new project

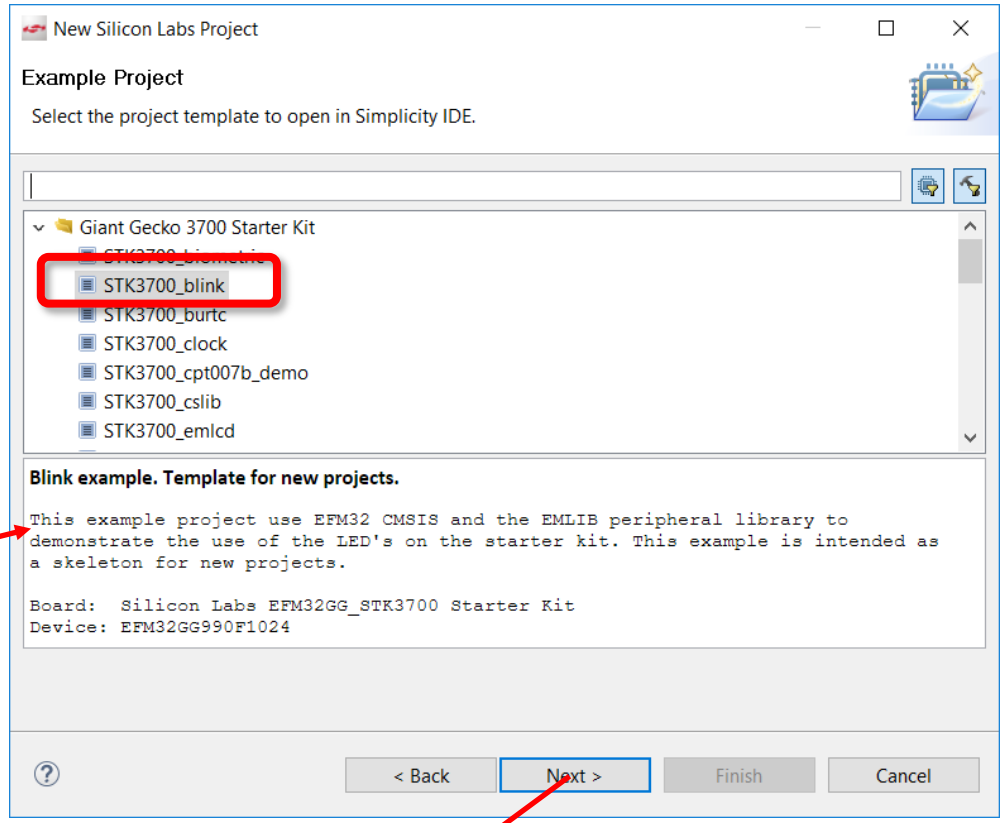
- File->New->Project:



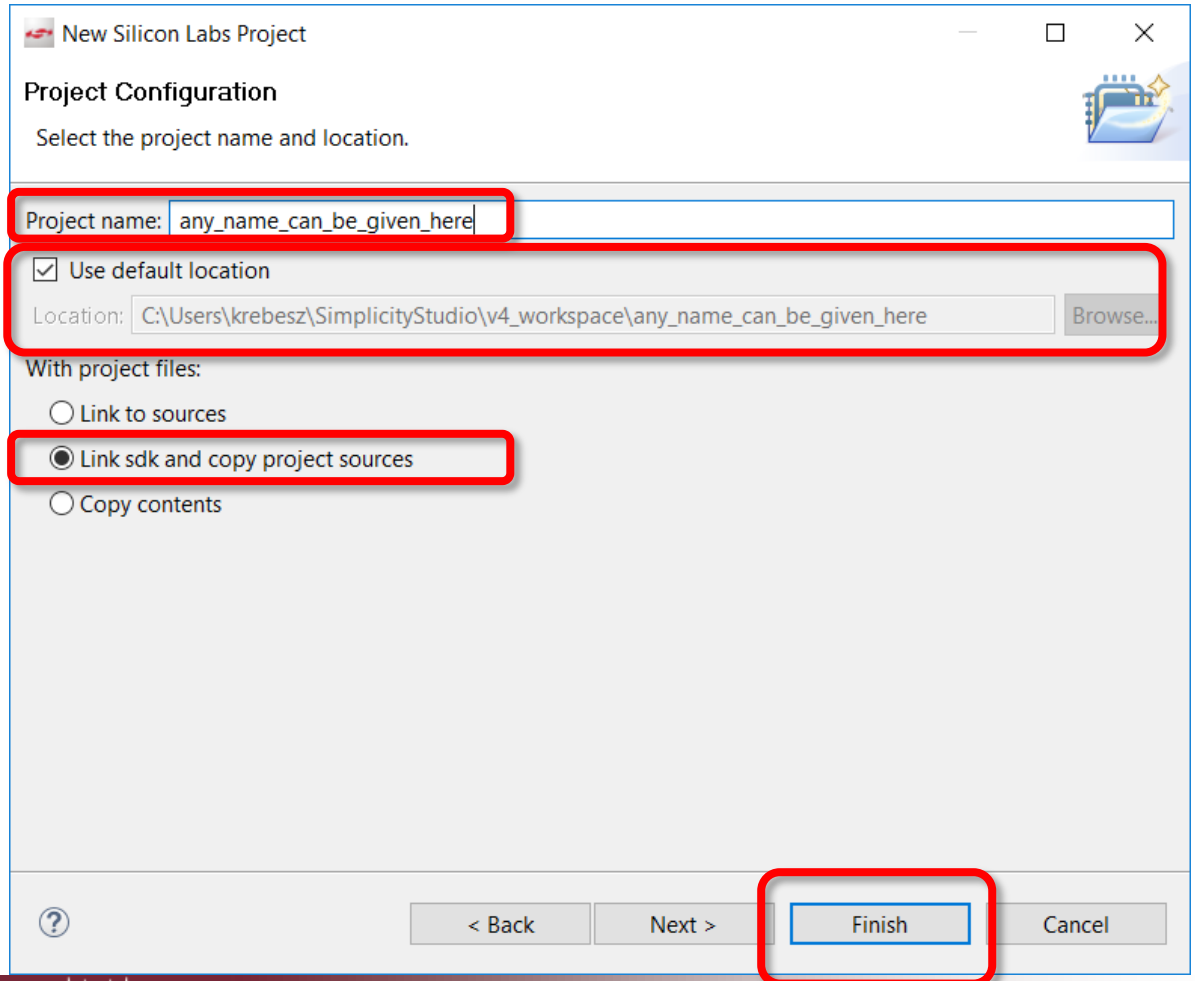


3) Start a new project

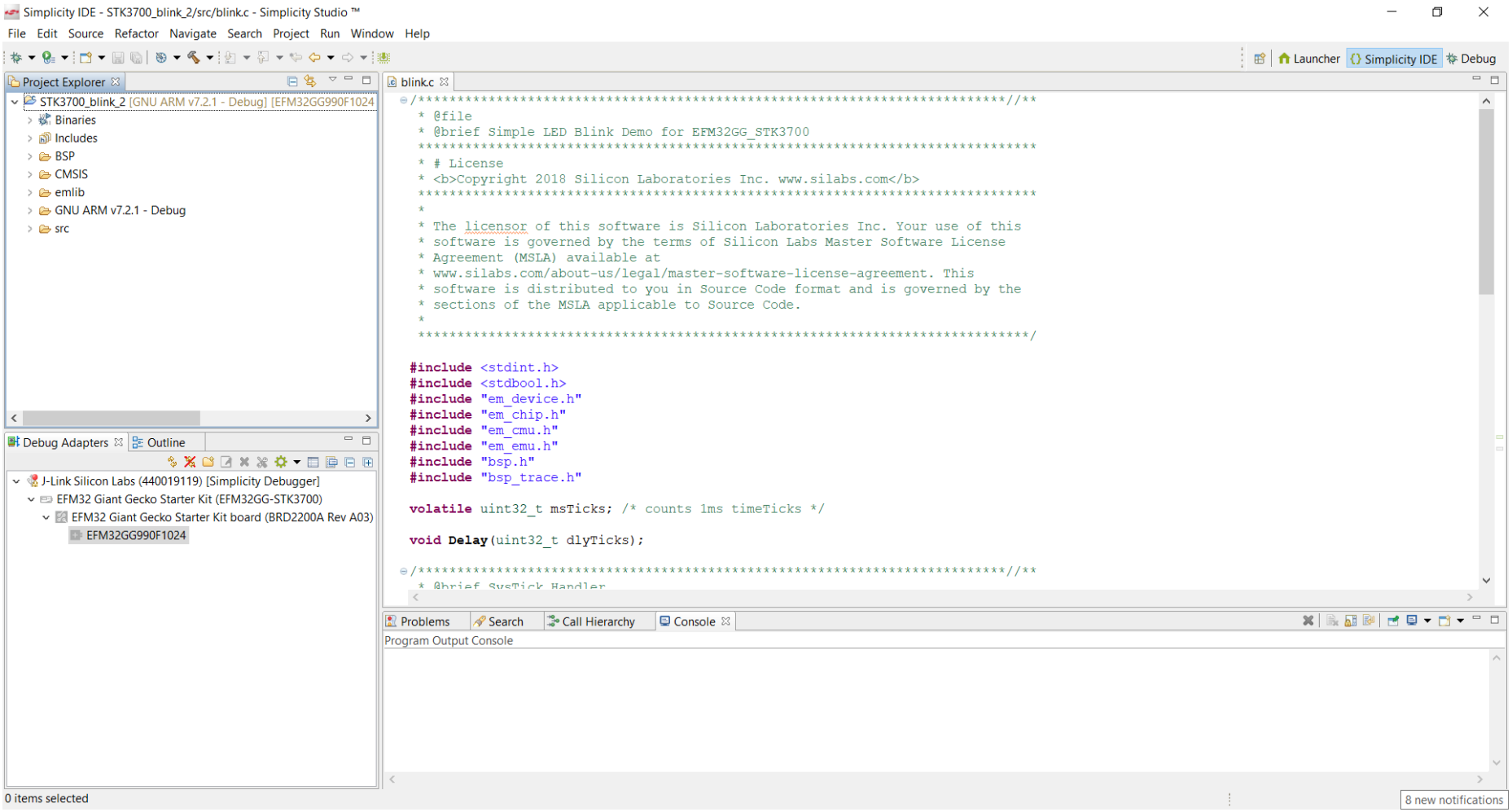




3) Start a new project

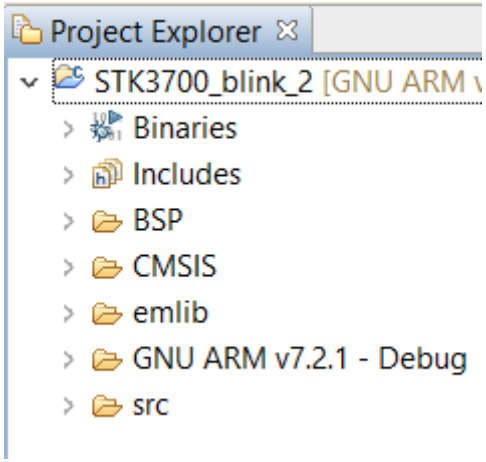


4) Example project created



4.1) Project Explorer

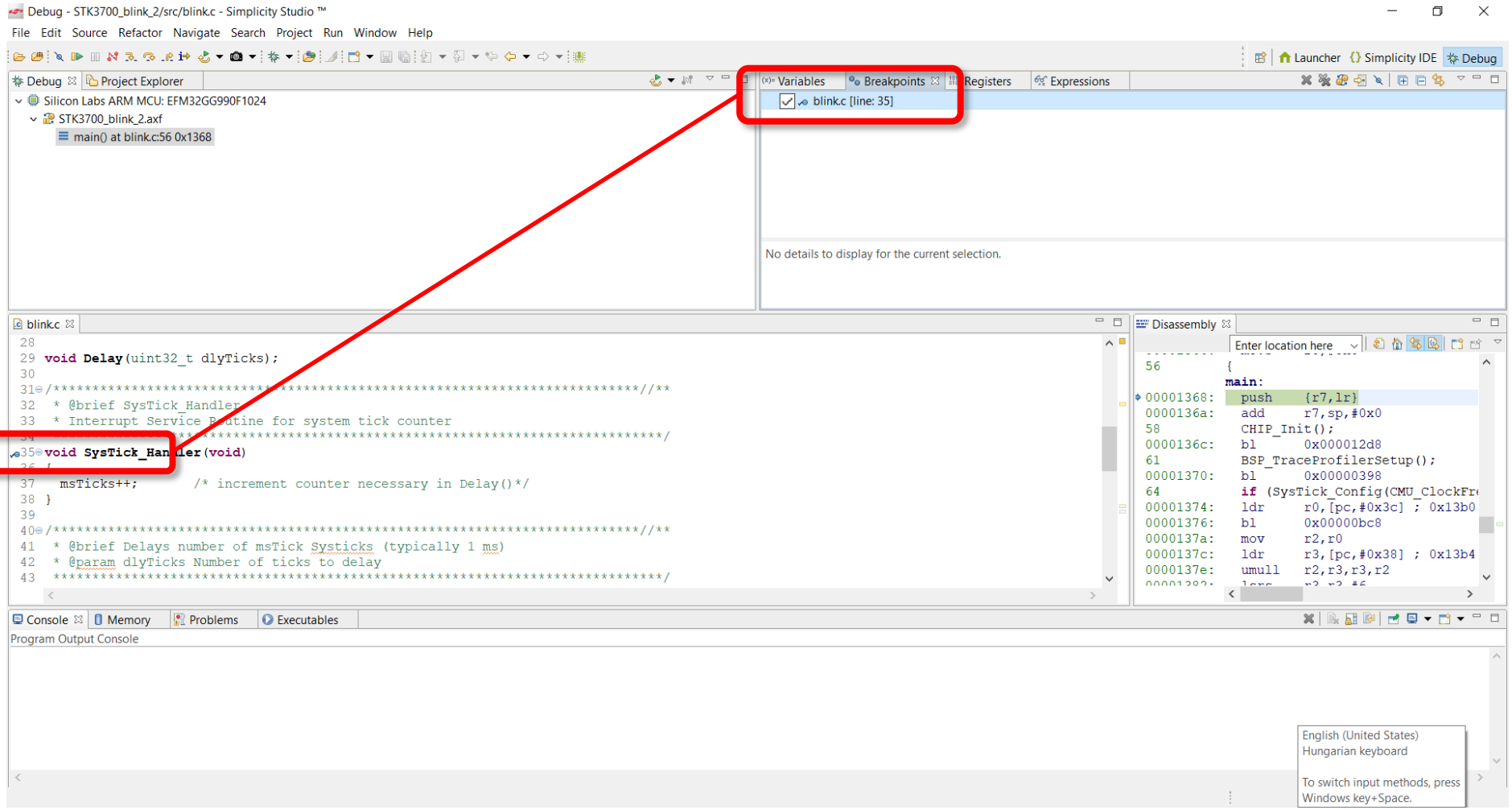
- Binaries: “raw” files (hex, bin)
- Includes: header files (function defs)
- BSP: board support package
- CMSIS: core management
- emlib: manages the whole uC
- GNU… : compiled SW components
- src: source files



4.2) Debug mode

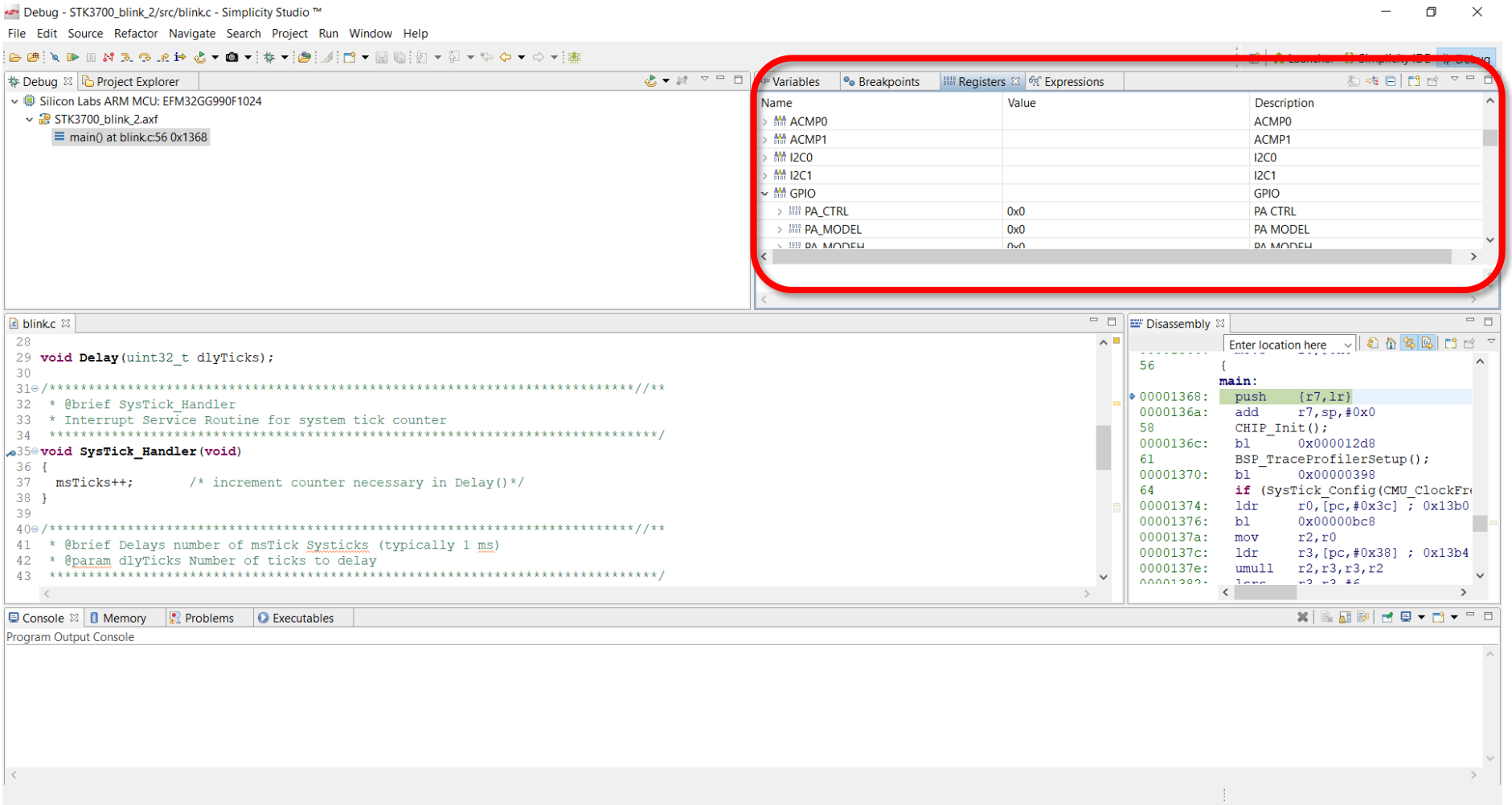
Icon	Command	Description
	Debug	The [**Debug**] button starts a new debug session. An active debug session must be disconnected before starting a new session using the same debug adapter.
	Resume	The [**Resume**] button runs the MCU after reset or after hitting a breakpoint.
	Suspend	The [**Suspend**] button halts the MCU.
	Disconnect	The [**Disconnect**] button terminates the current debug session and disconnects the debug adapter. The IDE will automatically switch back to the Development perspective.
	Reset the Device	The [**Reset the Device**] button performs a hardware reset on the MCU.
	Step Into	The [**Step Into**] button single steps into the first line of a function.
	Step Over	The [**Step Over**] button single steps over a function, executing the entire function.
	Step Return	The [**Step Return**] button steps out of a function, executing the rest of the function.
	Instruction Stepping Mode	The [**Instruction Stepping Mode**] button toggles assembly single stepping. When enabled, single steps will execute a single assembly instruction at a time. See the [**Disassembly**] view for the assembly code corresponding to the source code at the current line of execution.

4.2.1) Breakpoints



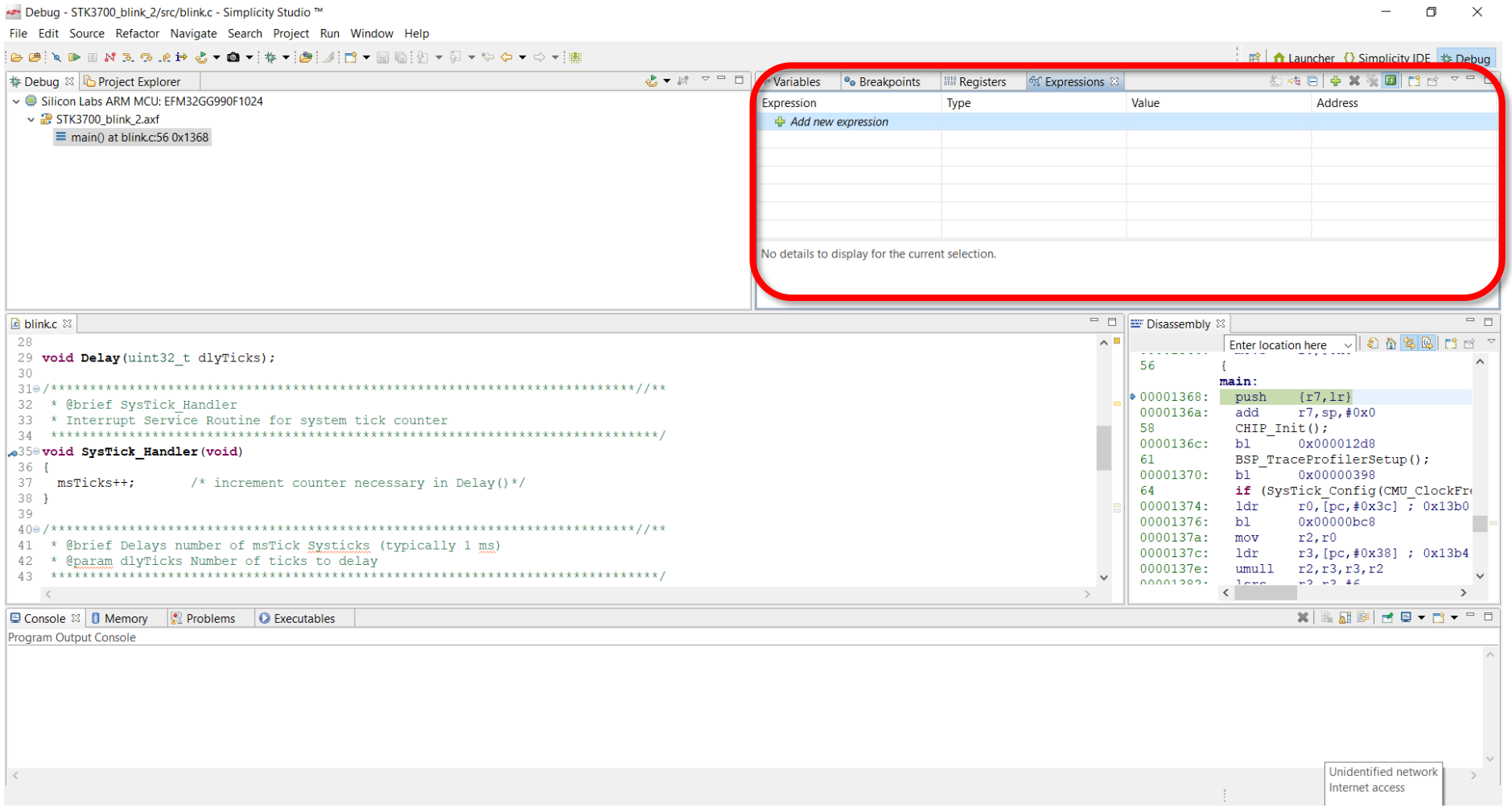
- Right click on the line to be able to add Breakpoint

4.2.2) Register values



- Register content can be manipulated

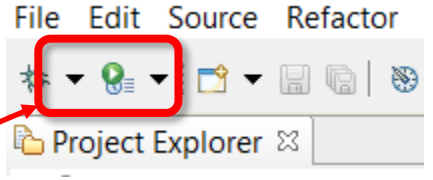
4.2.2) Expressions

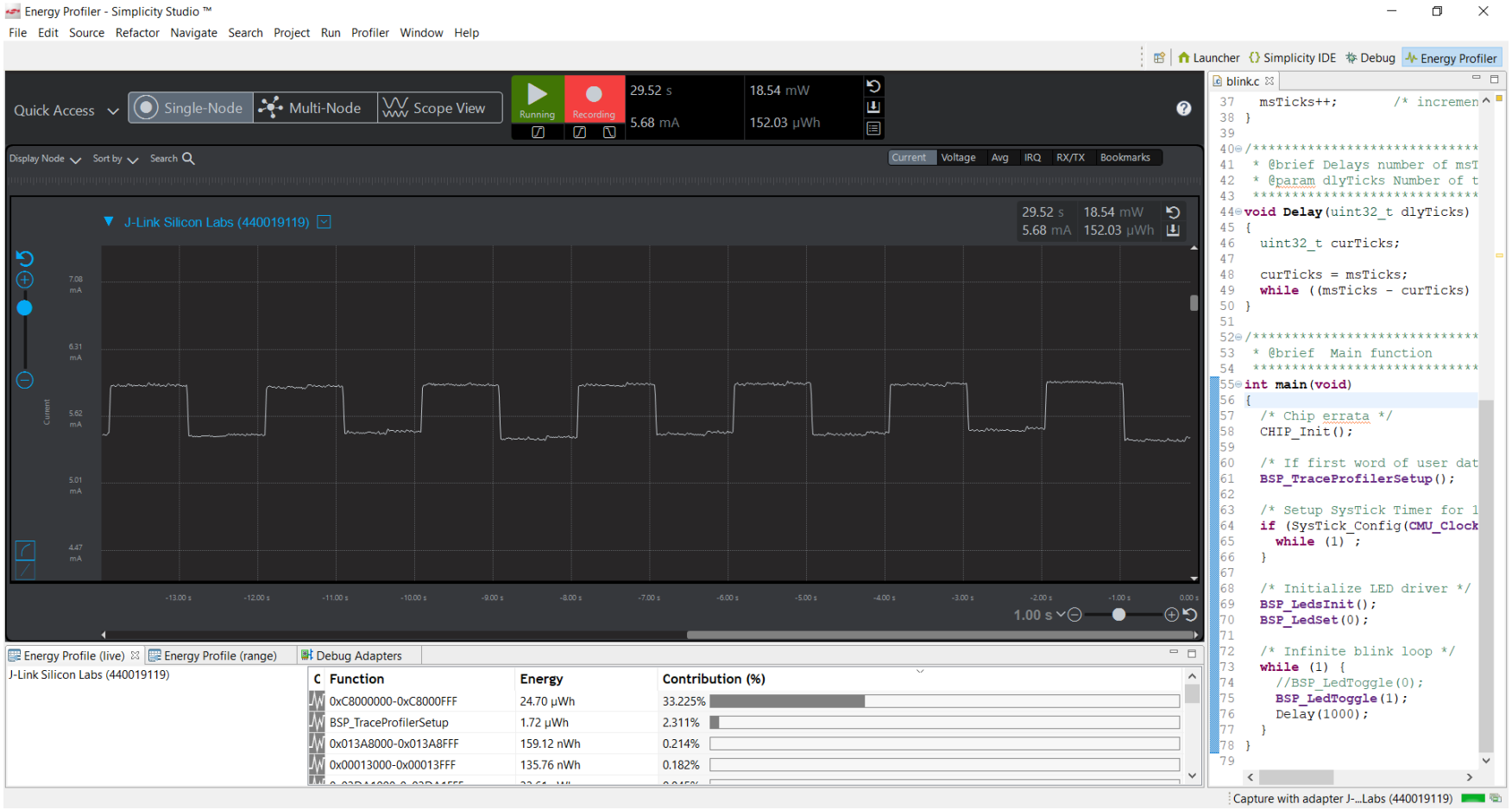


- Expressions can be entered, e.g.: variable1+variable2

5) Energy profiler

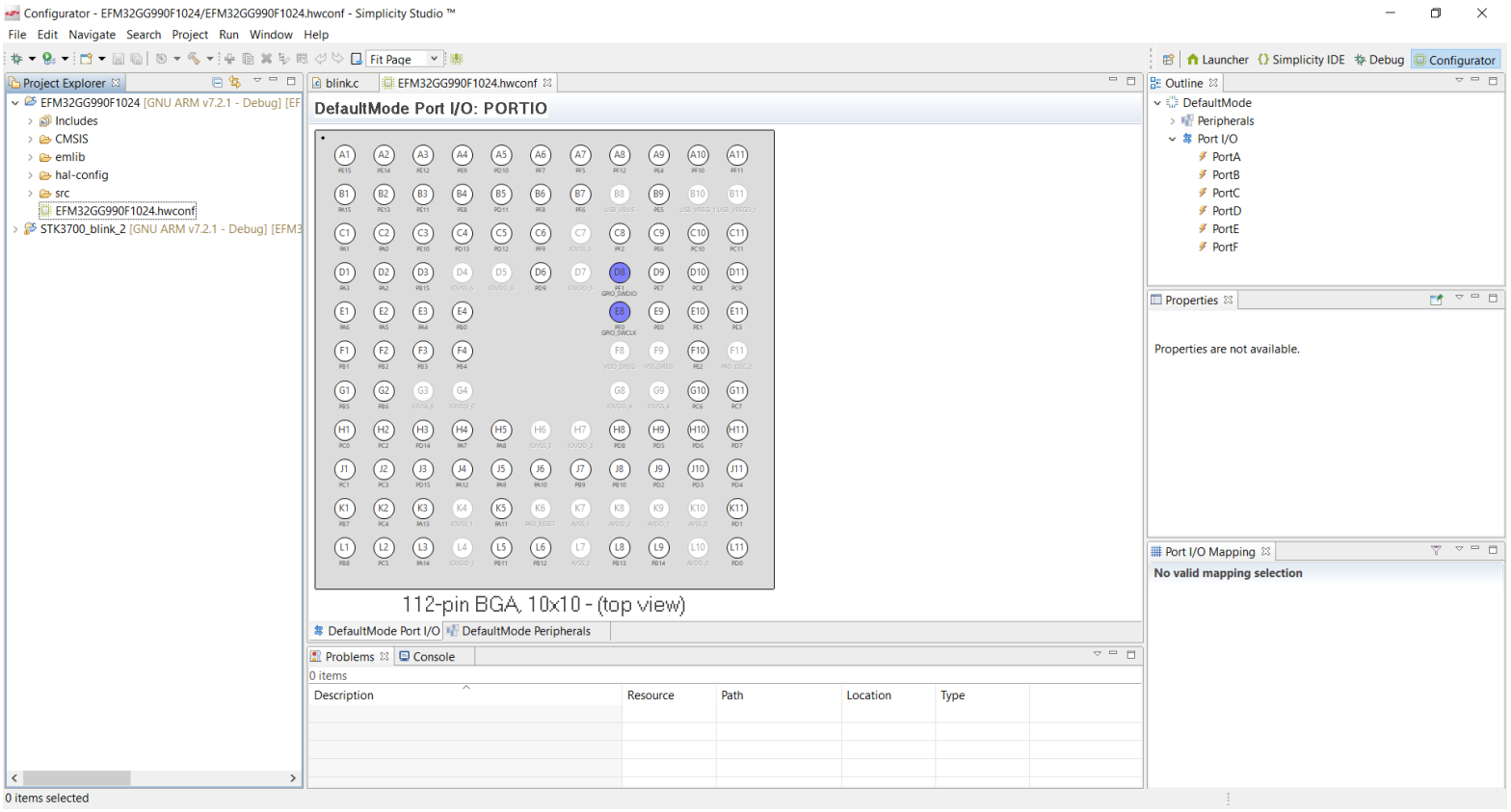
- Disable one LED (use e.g. comment //)
- Switch IDE mode and choose this icon





6) HW configurator

- Project is created by selecting configurator mode
- Simplifies peripheral initialization by presenting peripherals in a graphical user interface



7) Code development and manipulation

- Some useful hints
  - Code completion by Content Assist
    - type the first few letters of a function and press [**Ctrl+Space**]
      - display a list of functions that match
      - works for include files as well
  - Symbol expansion
    - stay over a function and information will pop-up
  - Open declaration
    - stay over a variable and press [**F3**]
      - Redirects where it was declared

7.1) Code development - #include

- Use a header file in your program by including it with the C preprocessing directive **#include**
- Two forms exist:
  - #include <file>
    Used for system header files. It searches for a file named 'file' in a standard list of system directories.
  - #include "file"
    Used for header files of your own program. It searches for a file named 'file' in the directory containing the current file.

7.2) Code explanation

- void
  - represents the absence of type
  - specifies that no value is available
- volatile
  - indicates that a value can change and the compiler should be prevented to perform optimization on it (which may lead to change the value into a constant)
- CHIP_Init();
  - HW errors are corrected in SW