

Embedded and ambient systems

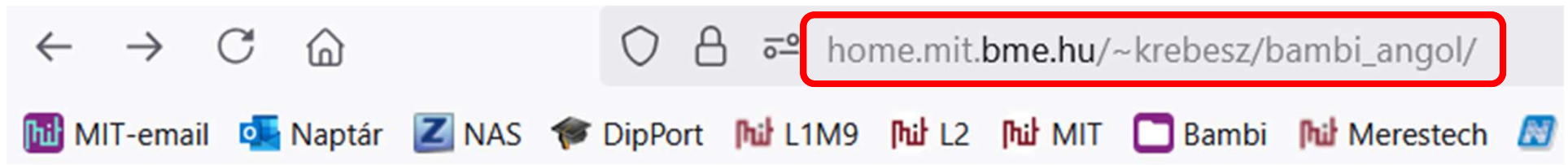
2025.09.10.

Practice 1



Department of
Artificial Intelligence and
Systems Engineering

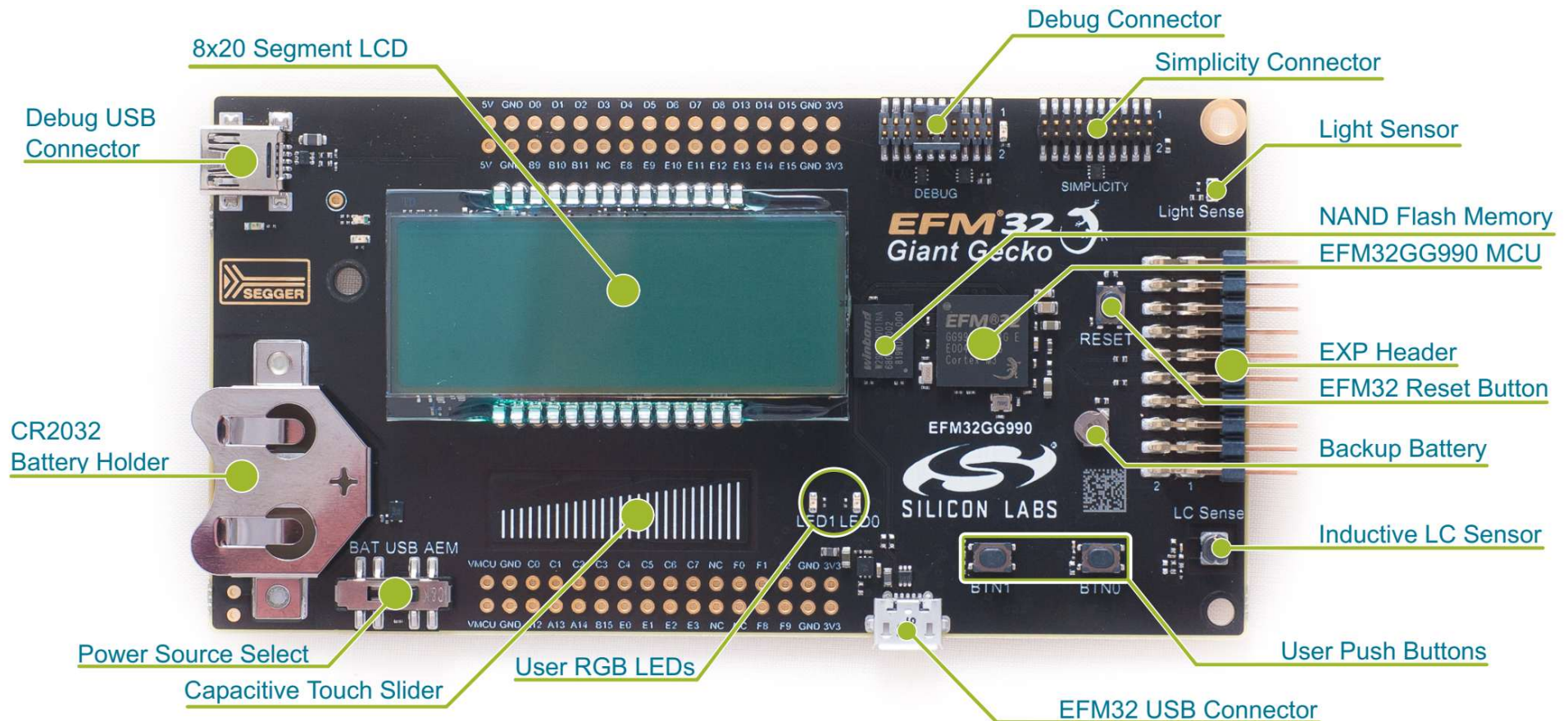
Preliminary



Index of /~krebesz/bambi_angol

- [Parent Directory](#)
 - [2025/](#)
 - [exams/](#)
 - [midterm/](#)
 - [references/](#)
- [Parent Directory](#)
 - [Lectures/](#)
 - [Practices/](#)
- An arrow points from the [2025/](#) link in the first list to the [Practices/](#) link in the second list.

1) Development board: EFM32GG-STK3700

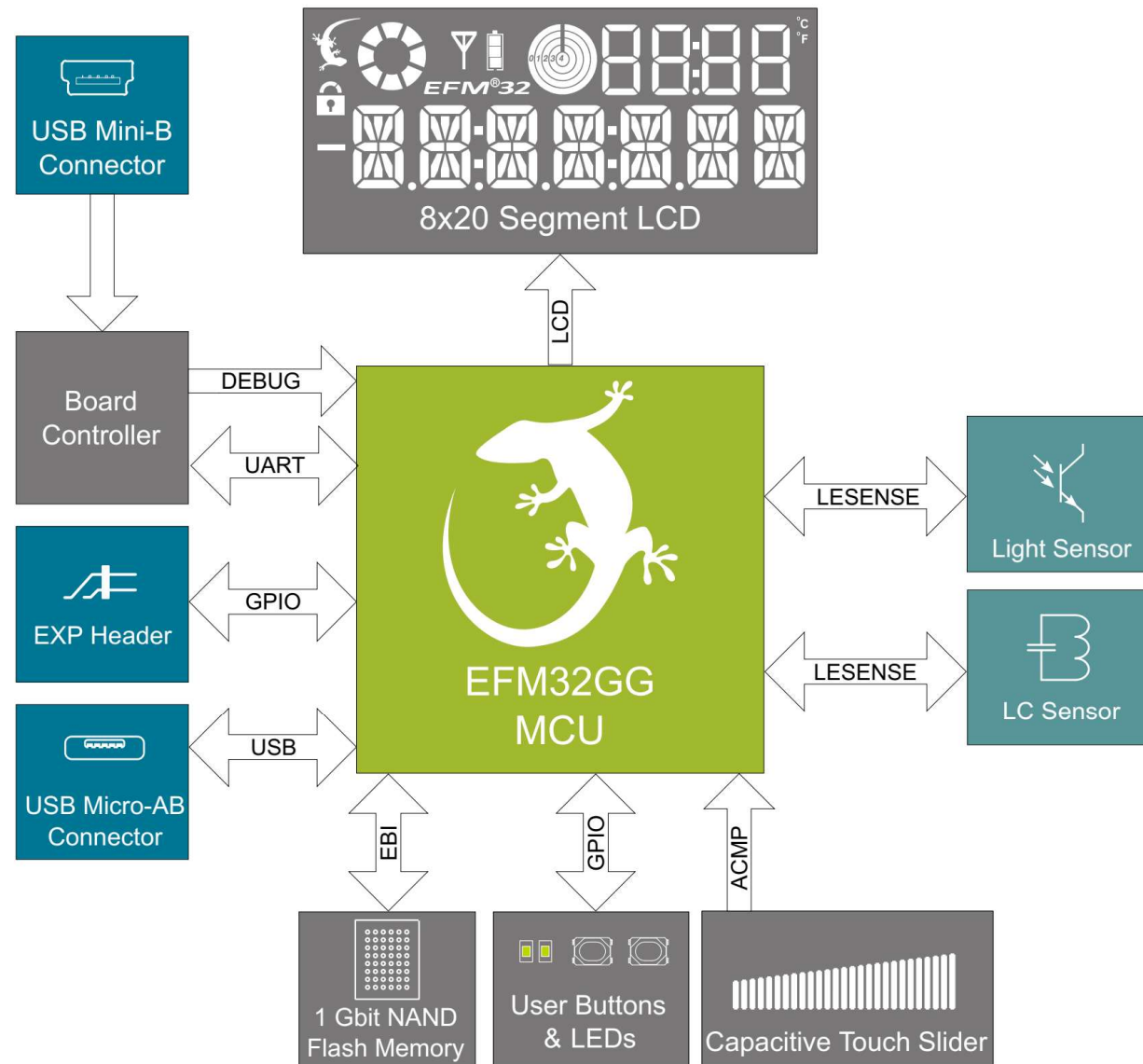


- <https://www.silabs.com/development-tools/mcu/32-bit/efm32gg-starter-kit>

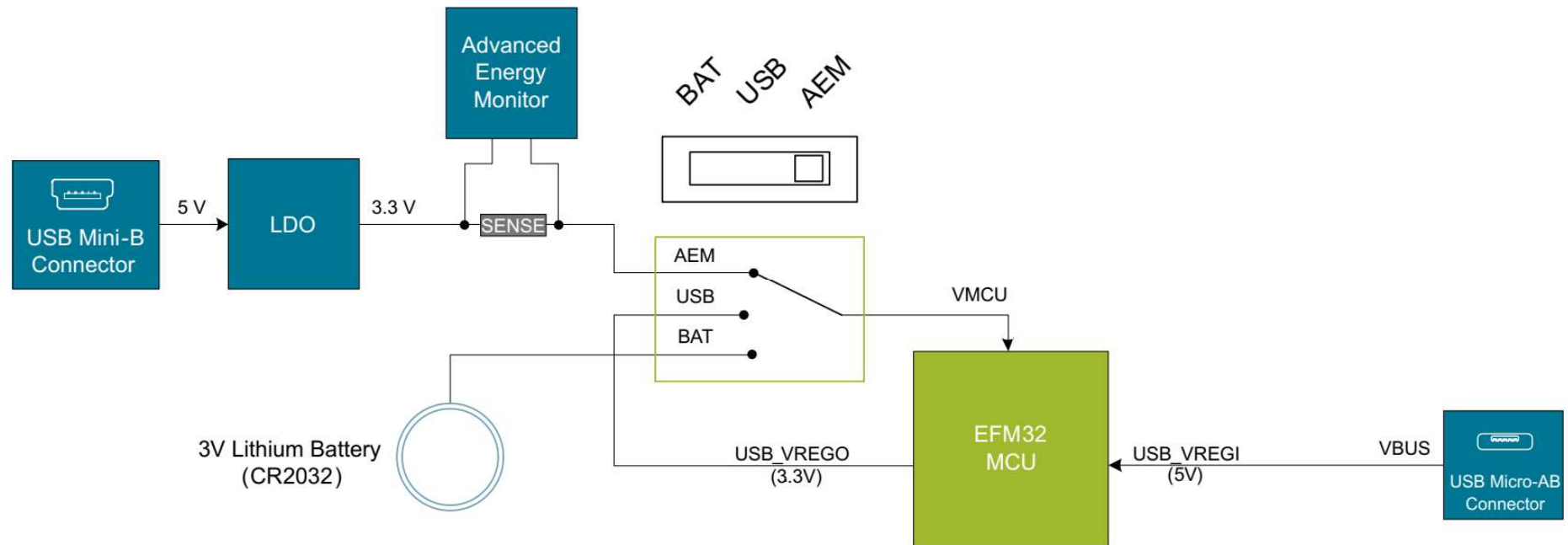
1.1) Main features

- 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

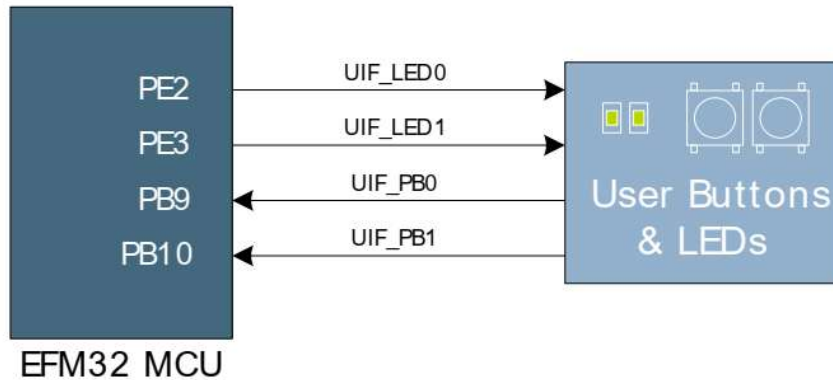


- DBG: via on-board debugger energy monitor can be used (use this)
- BAT: use CR2032 battery
- USB: MCU integrated voltage regulator is used

1.3) Power supply and proper connection

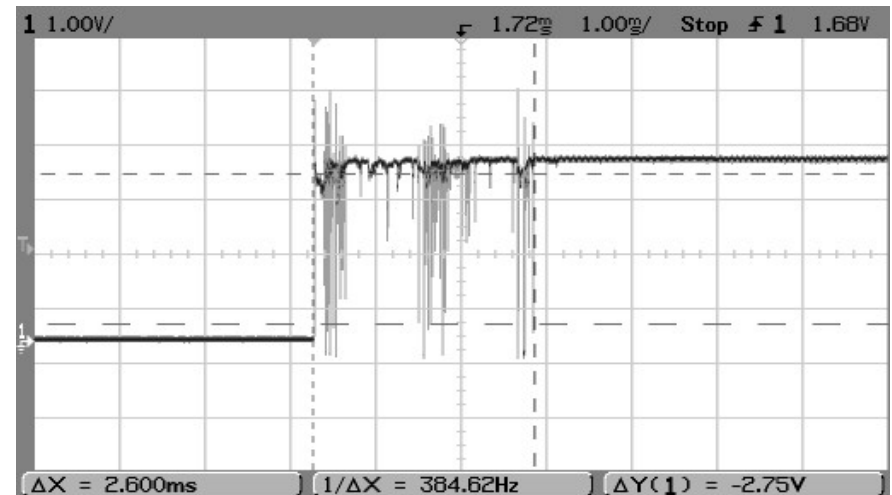
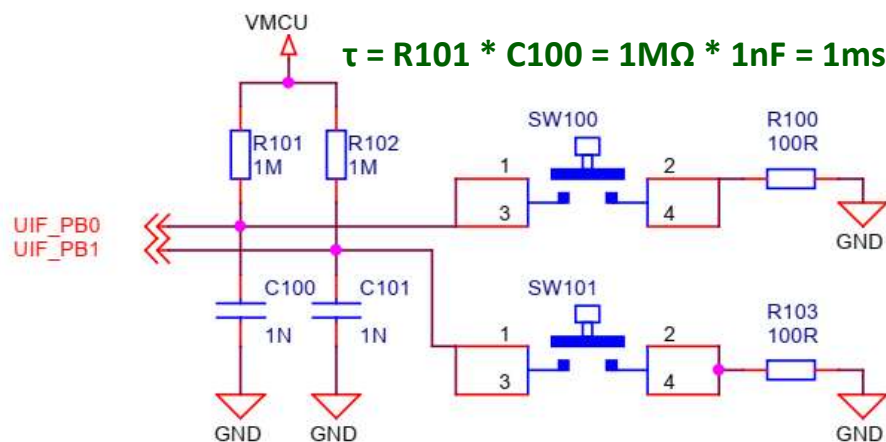


1.4) Peripherals-Buttons/LEDs



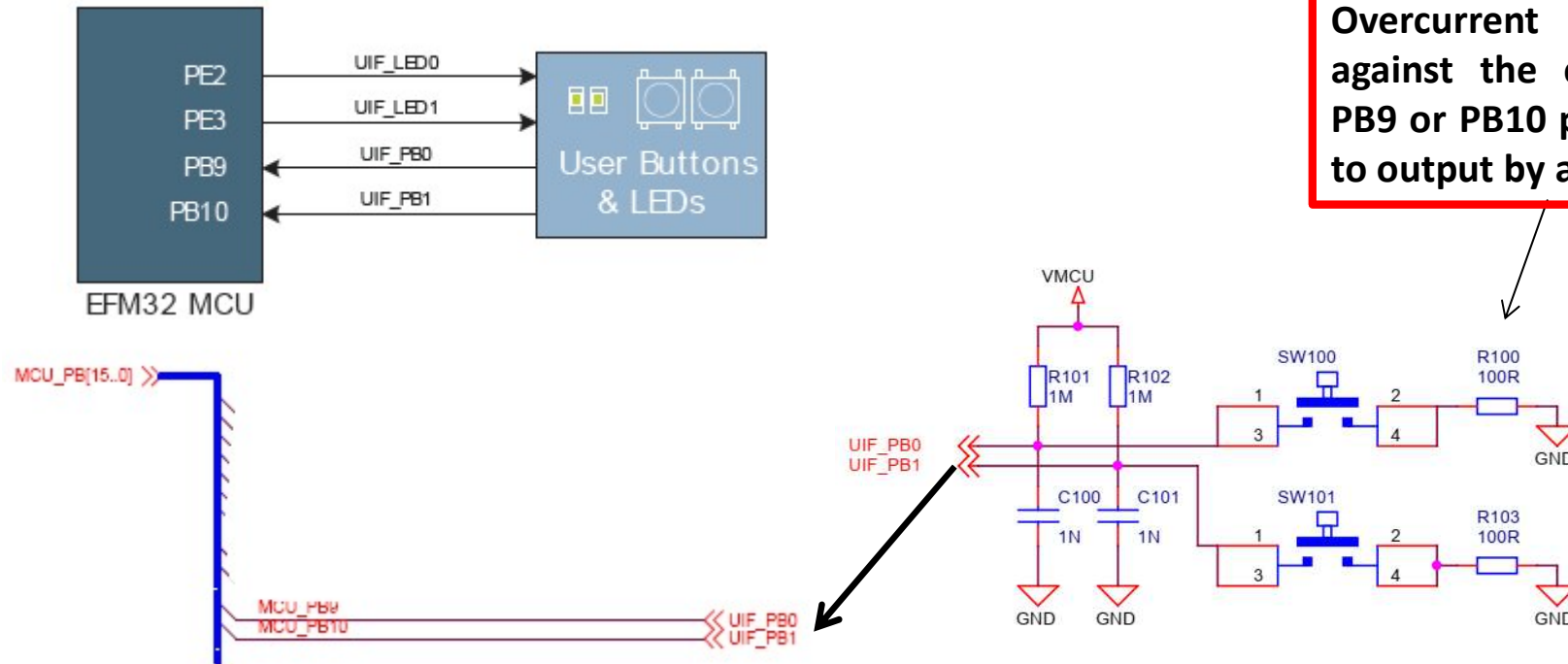
PB0=push button nr. 0
PB9=9th bit of port B
PE3=3rd bit of port E

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

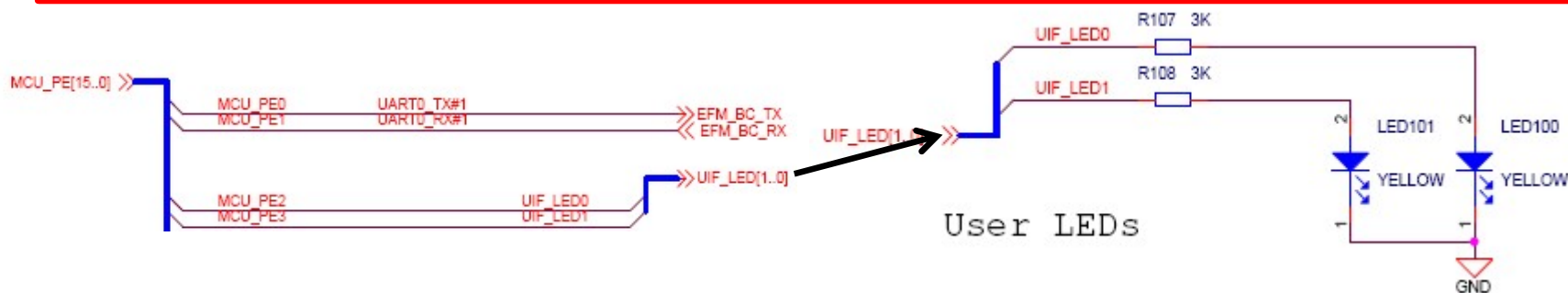


1.4) Peripherals-Buttons/LEDs

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



LED shunt resistor: $(3.3V-2V)/3k\Omega \approx 0.4mA$ (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)

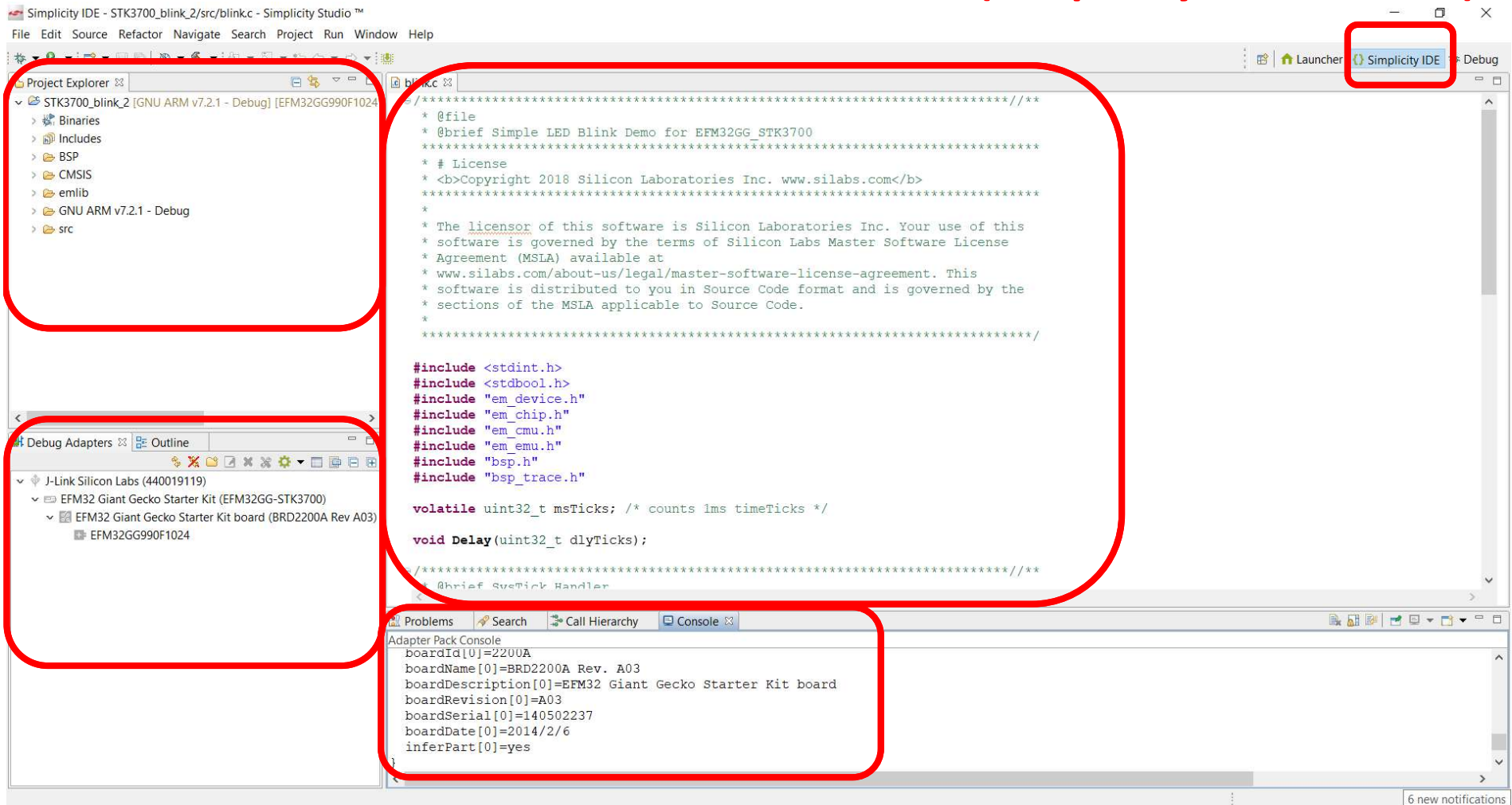
The screenshot shows the Simplicity Studio IDE interface. The title bar reads "Launcher - STK3700_blink_2/src/blink.c - Simplicity Studio™". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. A search bar is located below the menu. The left sidebar contains a "Debug Adapters" view, which is highlighted with a red box and labeled "Connected board". It shows a tree structure: J-Link Silicon Labs (440019119) > EFM32 Giant Gecko Starter Kit (EFM32GG-STK3700) > EFM32 Giant Gecko Starter Kit board (BRD2200A Rev A03) > EFM32GG990F1024. The main area displays a "Welcome to Simplicity Studio" message. Below this, there are tabs for "Getting Started", "Documentation", "Compatible Tools", and "Resources". The "Getting Started" tab is active, showing "Demos", "Software Examples", and "SDK Documentation" sections. Each section has a "Change Preferred SDK" link. A red box labeled "For your help" encompasses the "Getting Started" tab and its content. The bottom status bar shows "© 2020 Silicon Labs".

Connected board

For your help

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

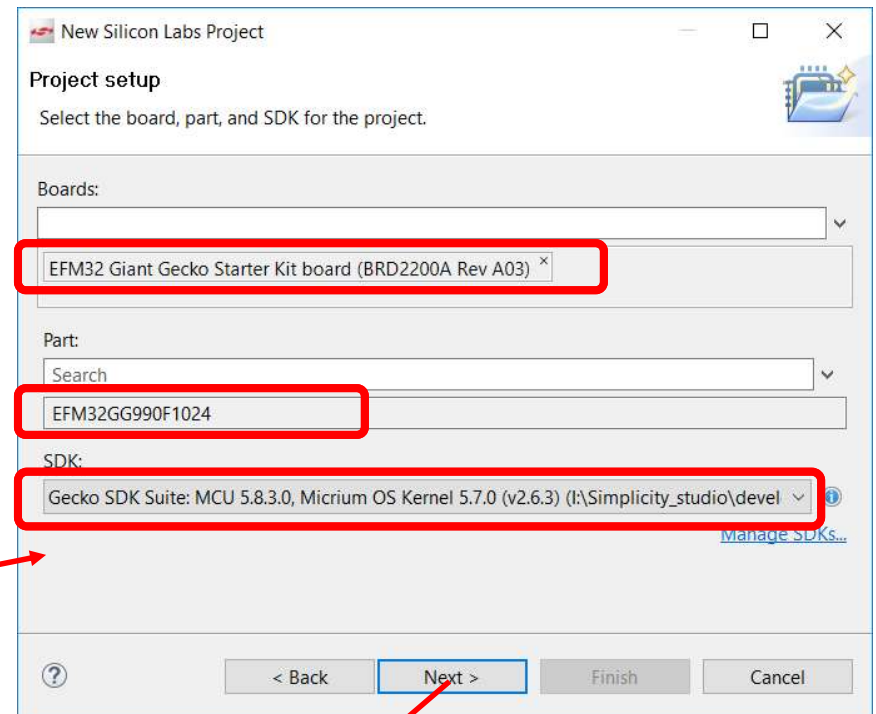
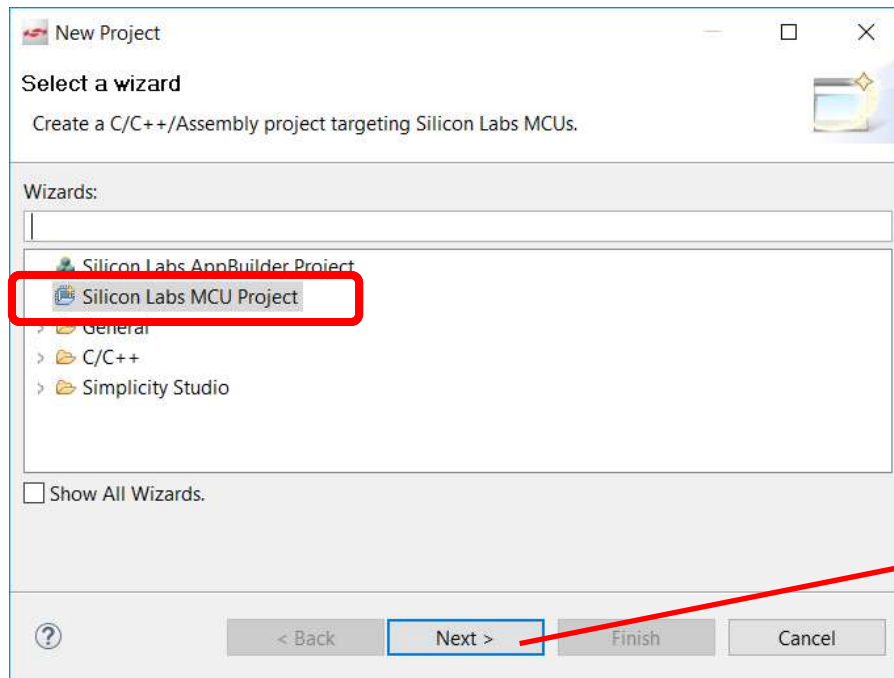
View (Debug is active)

The screenshot shows the SImplicity Studio IDE interface with the following components highlighted by red boxes:

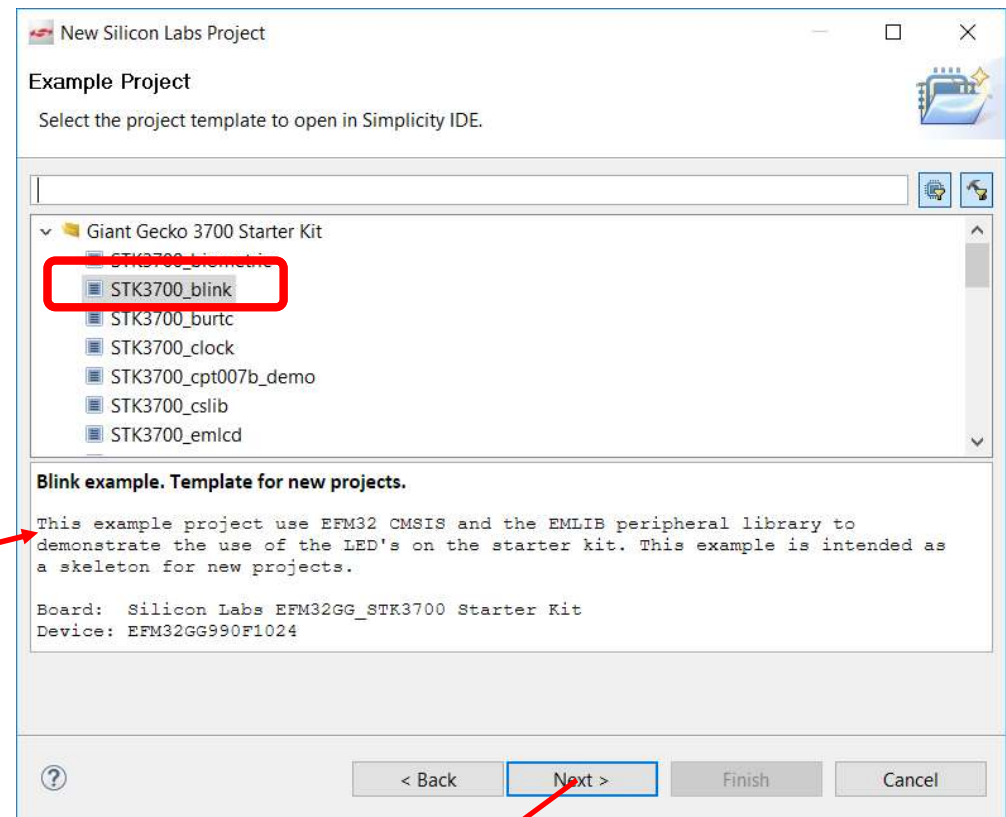
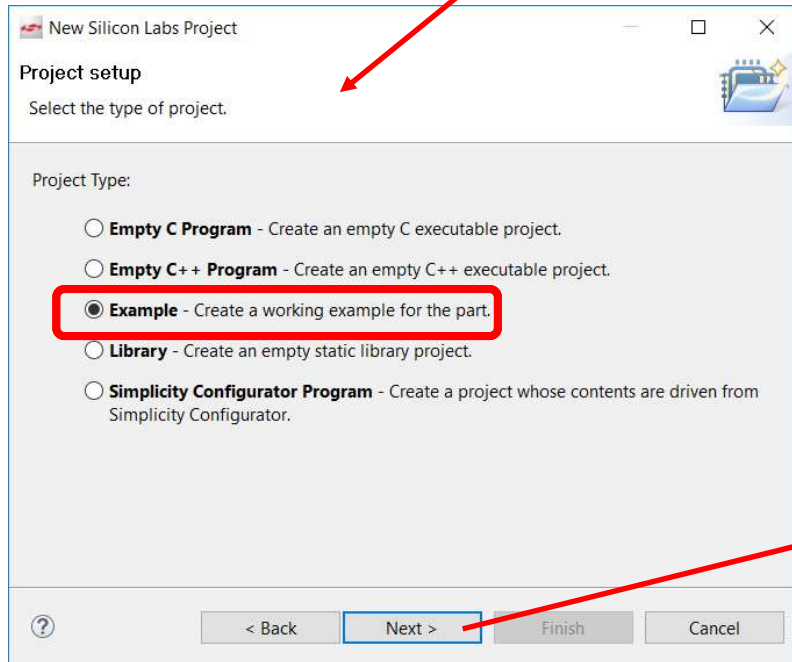
- Run and Debug Buttons:** The 'Run' (green play) and 'Debug' (blue bug) buttons in the top toolbar.
- Project Explorer:** The left sidebar showing the project structure, including 'Silicon Labs ARM MCU: EFM32GG990F1024' and 'STK3700_blink_2.axf'.
- Variables Panel:** A panel on the right showing the 'msTicks' variable with a value of 0 and type 'volatile uint32_t'.
- Source Code Editor:** The main editor window showing the C code for 'blink.c', including the 'main' function and comments.
- Disassembly Panel:** A panel on the right showing the assembly code for the 'main' function, starting with 'push {r7,lr}'.
- Console Panel:** The bottom panel showing the 'Program Output Console'.

3) Start a new project

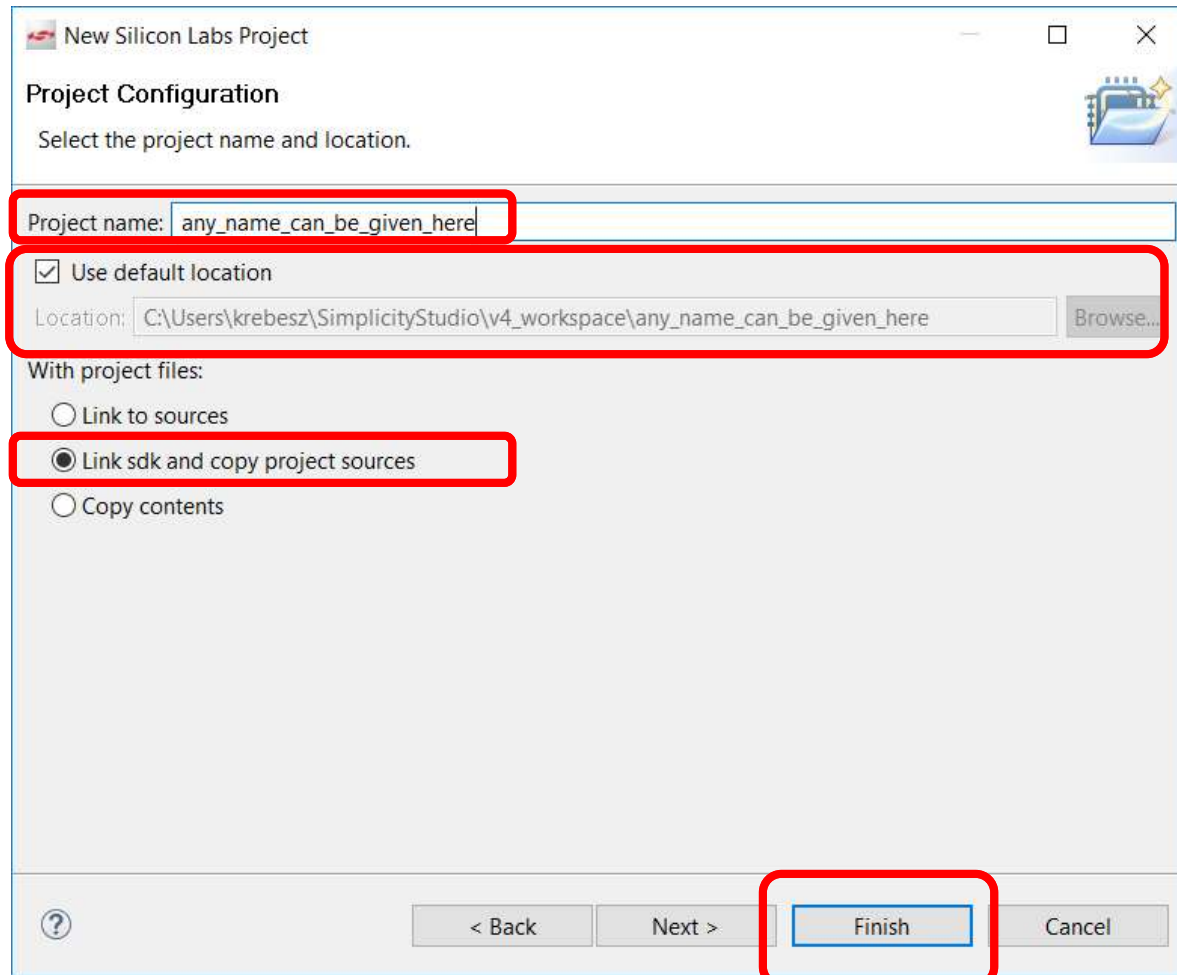
■ File->New->Project:



3) Start a new project



3) Start a new project



The screenshot shows the 'New Silicon Labs Project' dialog box. A red arrow points from the title '3) Start a new project' to the dialog. Red boxes highlight the 'Project name' field (containing 'any_name_can_be_given_here'), the 'Use default location' checkbox (checked), the 'Location' field (containing 'C:\Users\krebesz\SimplicityStudio\v4_workspace\any_name_can_be_given_here'), the 'Link sdk and copy project sources' radio button (selected), and the 'Finish' button at the bottom right.

New Silicon Labs Project

Project Configuration

Select the project name and location.

Project name: any_name_can_be_given_here

☒ Use default location

Location: C:\Users\krebesz\SimplicityStudio\v4_workspace\any_name_can_be_given_here Browse...

With project files:

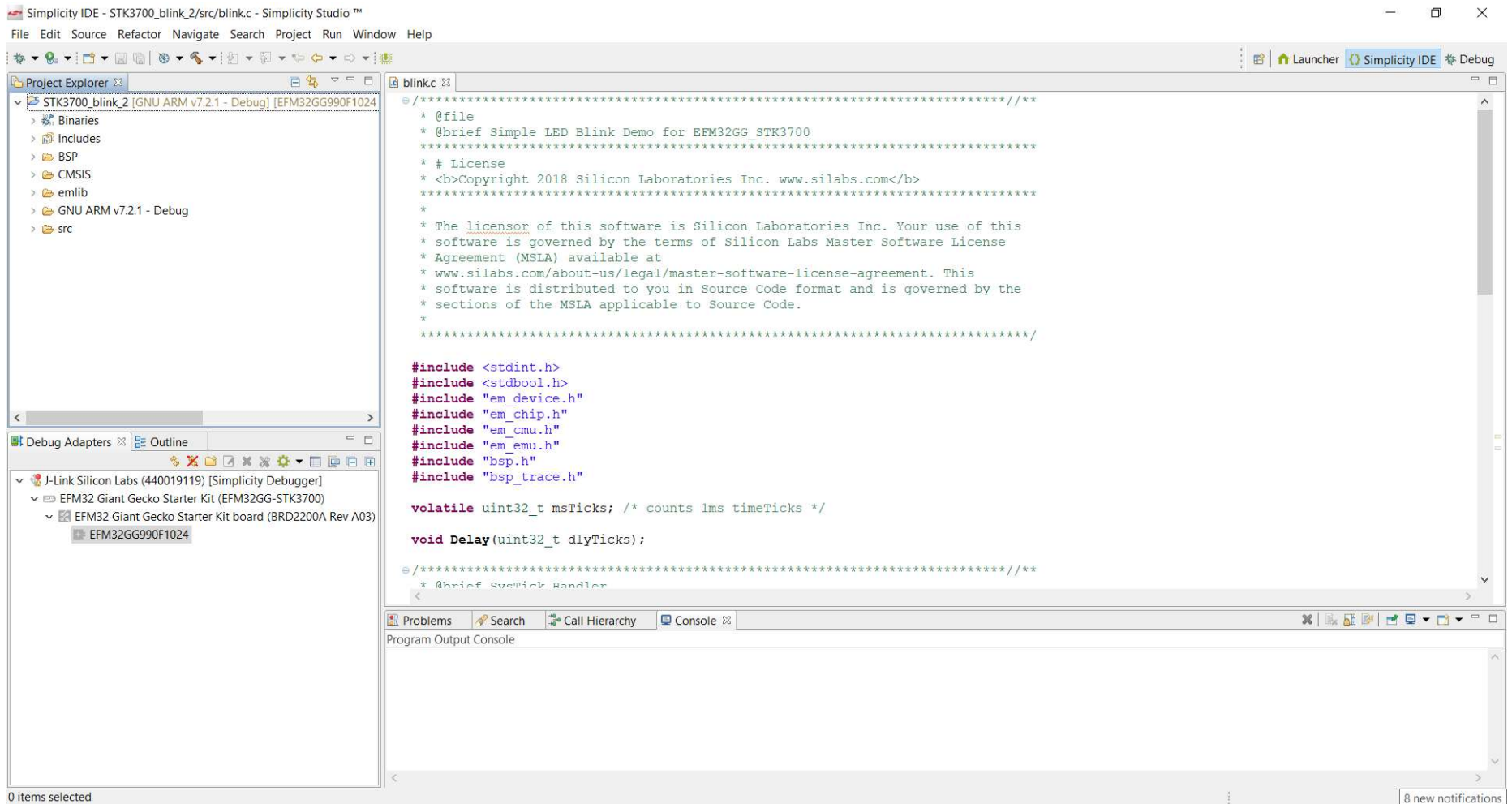
☐ Link to sources

☒ Link sdk and copy project sources

☐ Copy contents

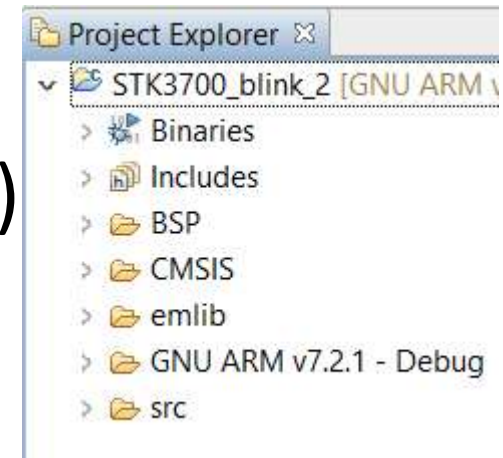
< Back Next > Finish Cancel

4) Example project created












4.1) Project Explorer

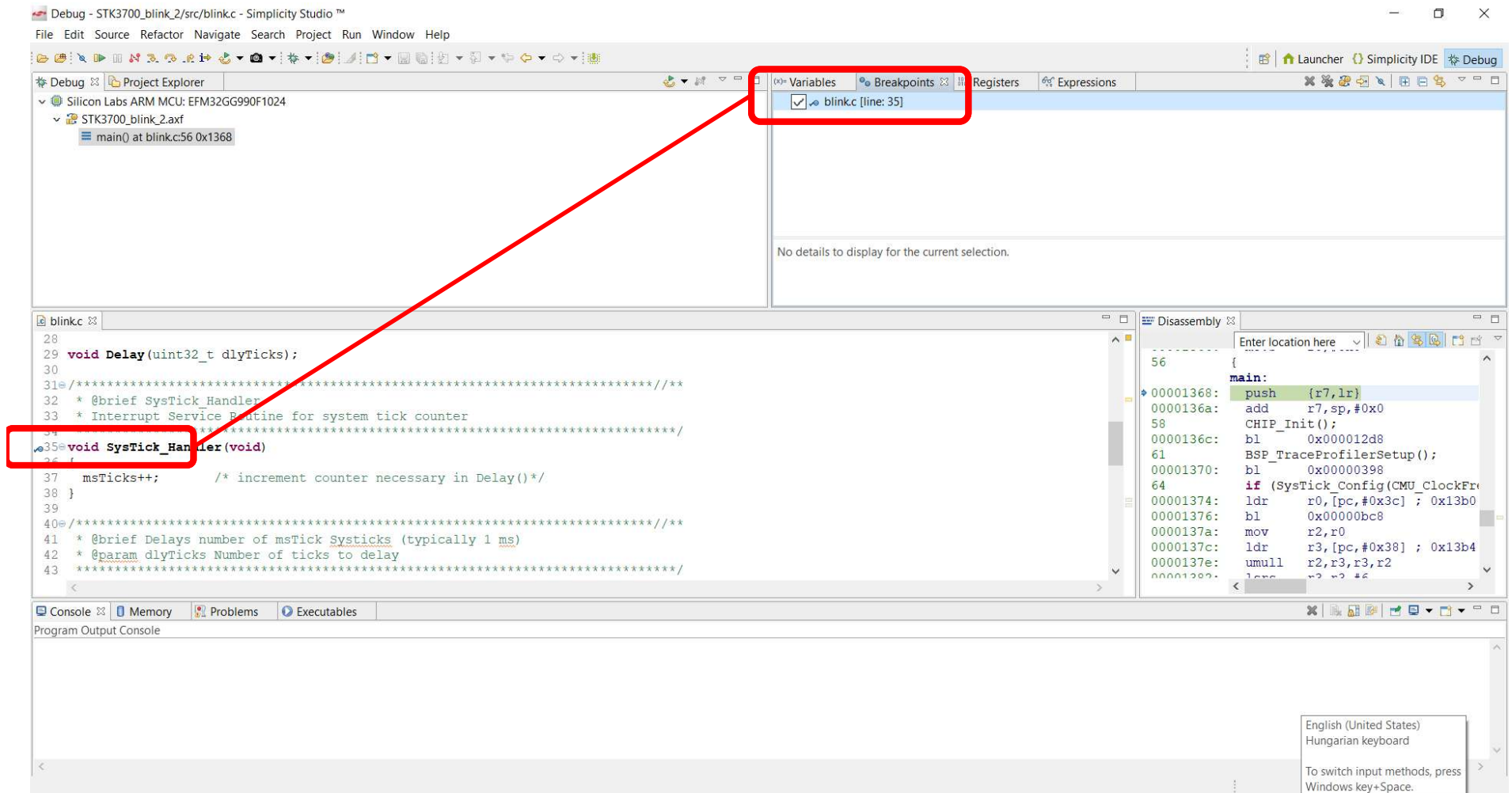
- Binaries: “raw” files (hex, bin)
- Includes: header files (function defs)
- BSP: board support package
- CMSIS: core management
- emlib: manages the whole uC
- GNU... : compiled SW components
- src: source files



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

The screenshot displays the Simplicity Studio IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Project Explorer on the left shows the project structure for 'STK3700_blink_2.axf', with the current file being 'main() at blink.c:56 0x1368'. The main window is divided into three panes. The top pane, titled 'Registers', is highlighted with a red rectangle and contains a table of hardware registers:

Name	Value	Description
ACMP0		ACMP0
ACMP1		ACMP1
I2C0		I2C0
I2C1		I2C1
GPIO		GPIO
PA_CTRL	0x0	PA CTRL
PA_MODEL	0x0	PA MODEL
PA_MODELH	0x0	PA MODELH

The bottom-left pane shows the C source code for 'blink.c', with the 'SysTick_Handler' function visible. The bottom-right pane, titled 'Disassembly', shows the assembly code for the 'main' function, starting at address 00001368. The assembly code includes instructions like 'push {r7,lr}', 'add r7,sp,#0x0', 'CHIP_Init()', 'BSP_TraceProfilerSetup()', and various load/store operations. The bottom status bar indicates the 'Program Output Console' is active.

- Register content can be manipulated

4.2.2) Expressions

The screenshot shows the Simplicity Studio IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The left sidebar shows the Project Explorer with the following structure:

- Debug - STK3700_blink_2/src/blink.c - Simplicity Studio™
- Silicon Labs ARM MCU: EFM32GG990F1024
 - STK3700_blink_2.axf
 - main() at blink.c:56 0x1368

The main window displays the source code for `blink.c`. The code includes a `Delay` function and a `SysTick_Handler` function. The `SysTick_Handler` function is currently selected, and the cursor is at line 35.

The right sidebar shows the Expressions window, which is highlighted with a red rectangle. It contains a table with the following columns: Expression, Type, Value, and Address. The table is currently empty, and a red box highlights the "Add new expression" button. Below the table, it says "No details to display for the current selection."

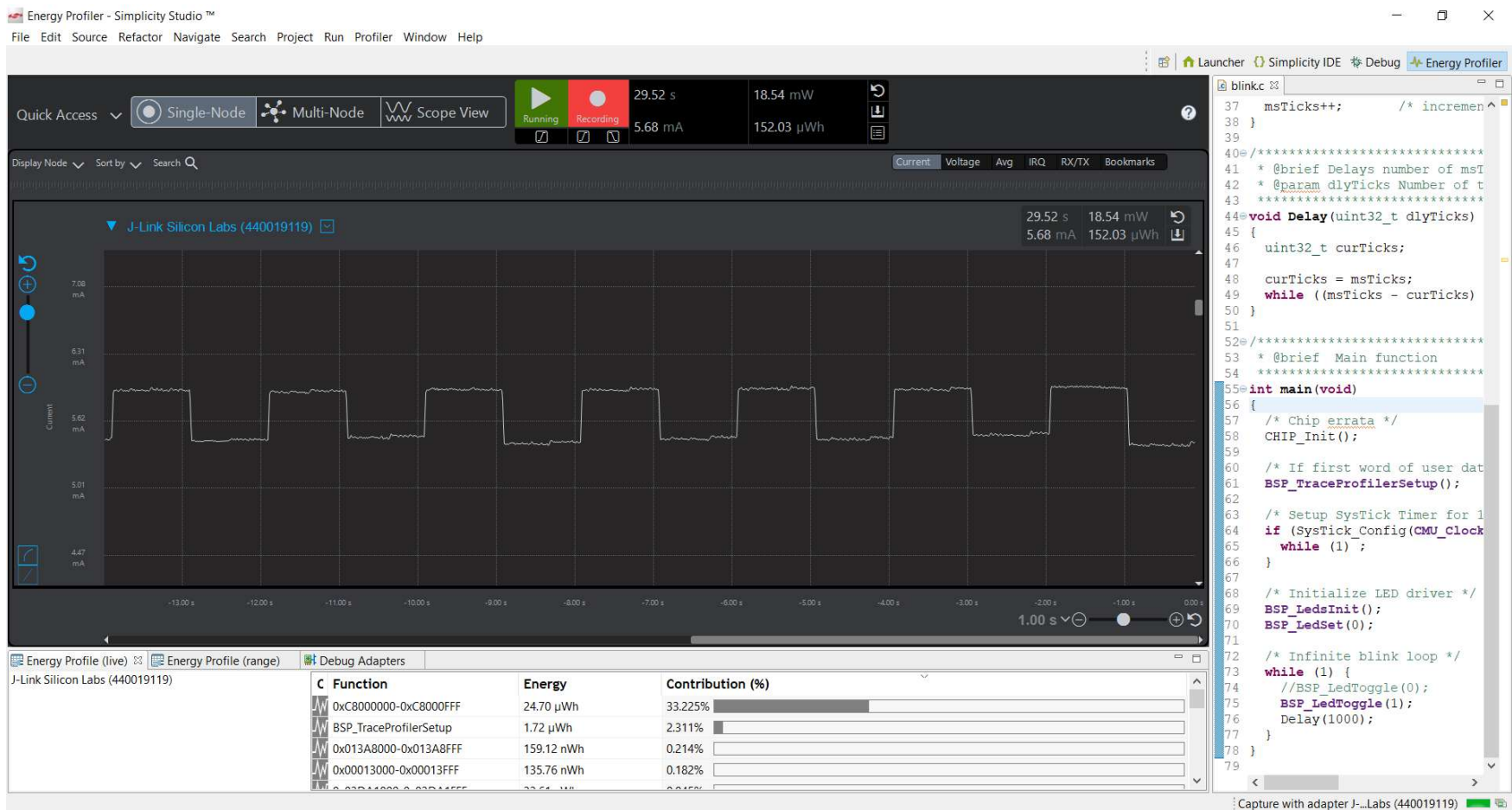
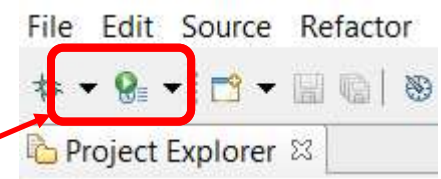
The bottom of the IDE shows the Disassembly window, which displays the assembly code for the `main` function. The assembly code is as follows:

```
56      {
00001368:  push    {r7,lr}
0000136a:  add     r7,sp,#0x0
58      CHIP_Init();
0000136c:  bl      0x000012d8
61      BSP_TraceProfilerSetup();
00001370:  bl      0x00000398
64      if (SysTick_Config(CMU_ClockFr
00001374:  ldr     r0,[pc,#0x3c] ; 0x13b0
00001376:  bl      0x00000bc8
0000137a:  mov     r2,r0
0000137c:  ldr     r3,[pc,#0x38] ; 0x13b4
0000137e:  umull   r2,r3,r3,r2
00001380:  ldr     r2,[pc,#0x38] ; 0x13b4
```

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

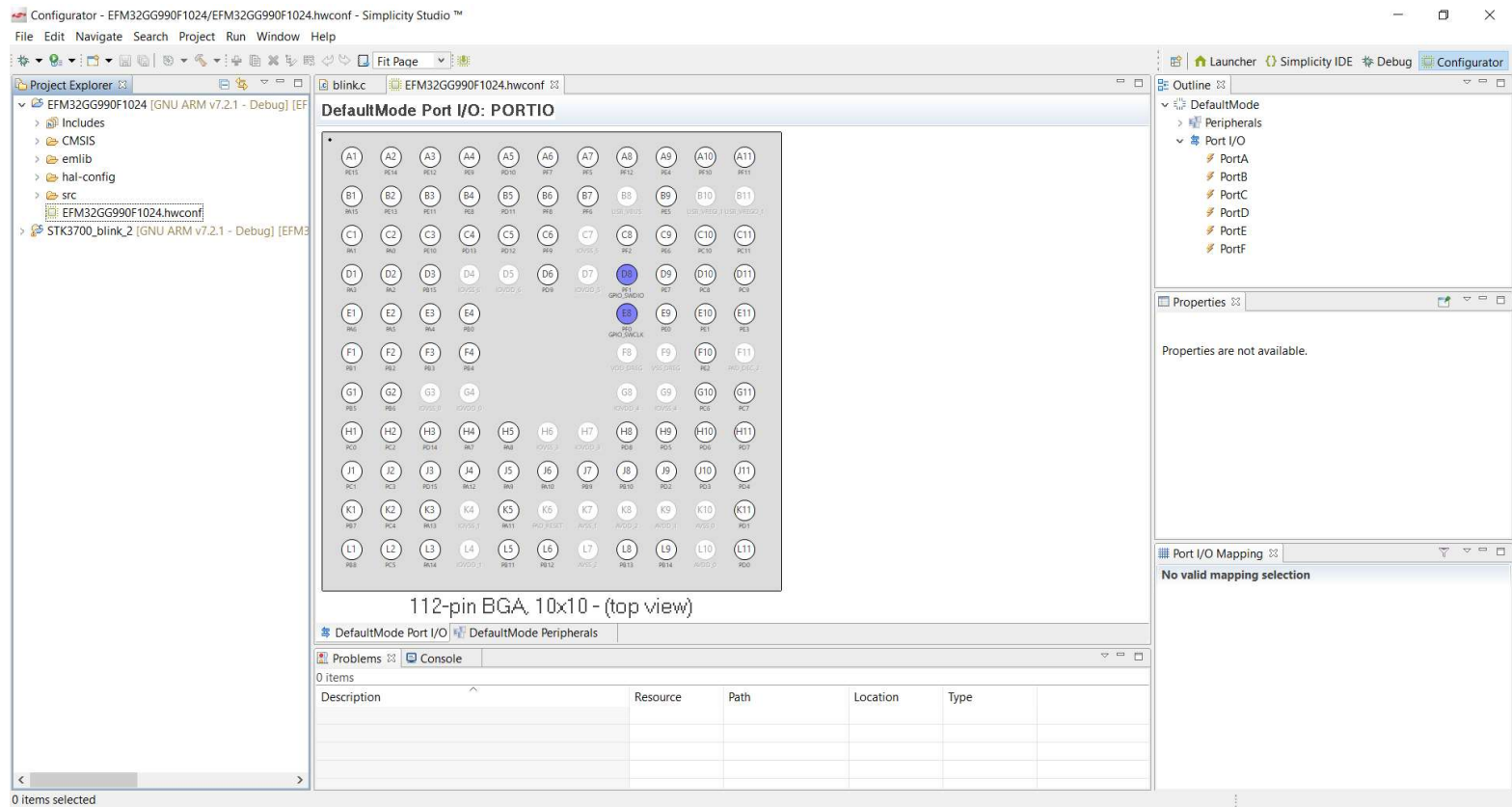
5) Energy profiler

- 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