

# Embedded and ambient systems

2022.09.28.

## Practice 1



Méréstechnika és  
Információs Rendszerek  
Tanszék

# Preliminary

- Check the web site of the course:  
[www.mit.bme.hu/eng/oktatas/targyak/VIMIAC06](http://www.mit.bme.hu/eng/oktatas/targyak/VIMIAC06)
- See menu on the left

**Embedded and ambient systems**


- > Official page on the faculty Web
- > Final exam
- > Homework
- > Midterm exam
- > **Practice**
- > Requirements
- > Textbooks and resources

**Embedded and ambient systems**


View Edit Revisions Track Translate

**VIMIAC06**  
B.Sc. Program > BSc in Electrical Engineering (current) > Embedded and control systems specialization - MIT, IIT, AUT (current)  
Official page on the faculty Web

**Course coordinator**

 **György Orosz**  
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

# Preliminary

## Related pages

- › Back to the course page
- › Official page on the faculty Web
- › Final exam
- › Homework
- › Midterm exam
- › Practice
- › Requirements
- › Textbooks and resources

## Practice

View

Edit

Revisions

Track

Translate

Unpublish

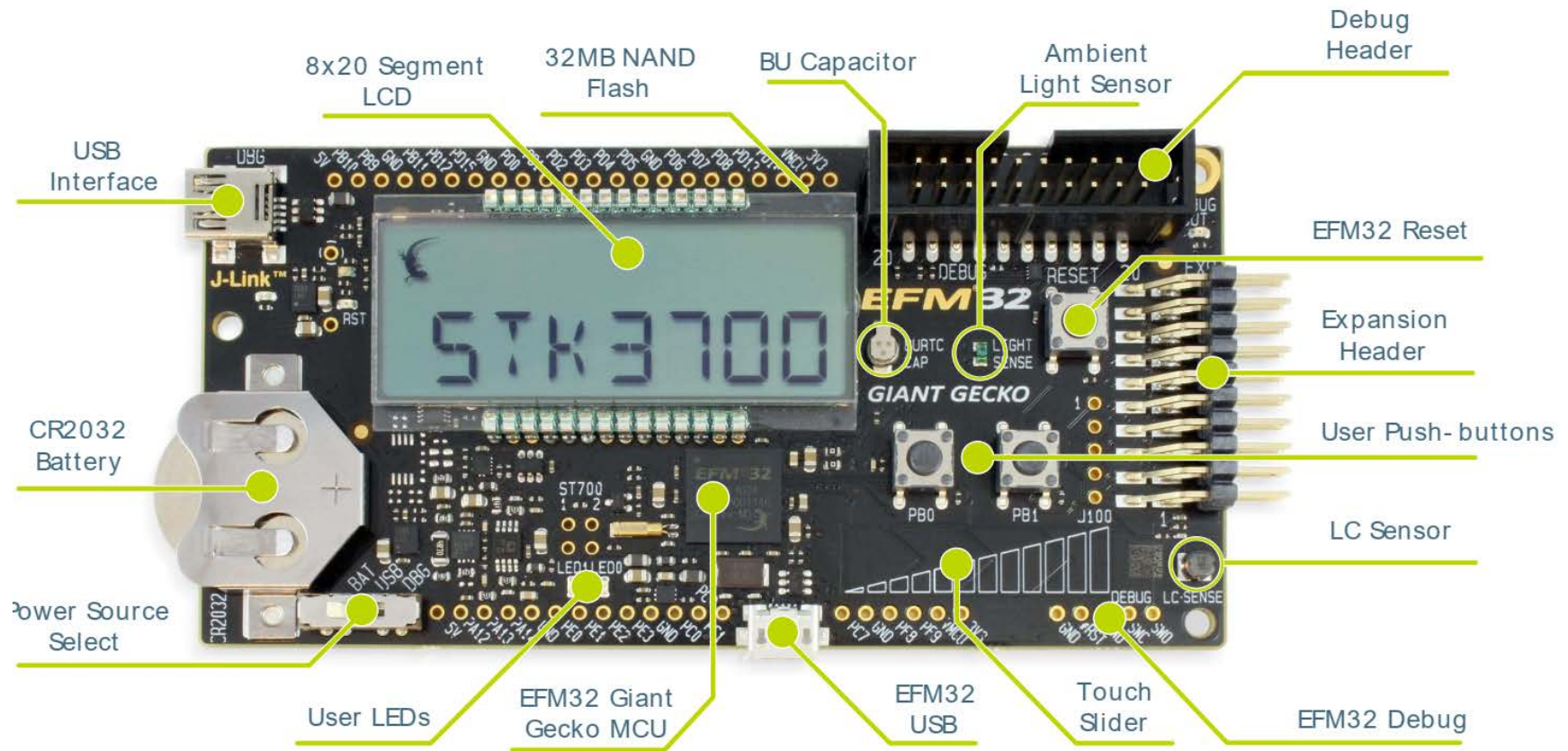
Practice materials will appear during the semester!

References available here:

[http://home.mit.bme.hu/~krebesz/bambi\\_angol/references/](http://home.mit.bme.hu/~krebesz/bambi_angol/references/)

Submitted by Krébesz Tamás István on 2020. September 7. 18:57 | Last updated: 2020. September 9. 09:57

# 1) Development board: EFM32GG-STK3700

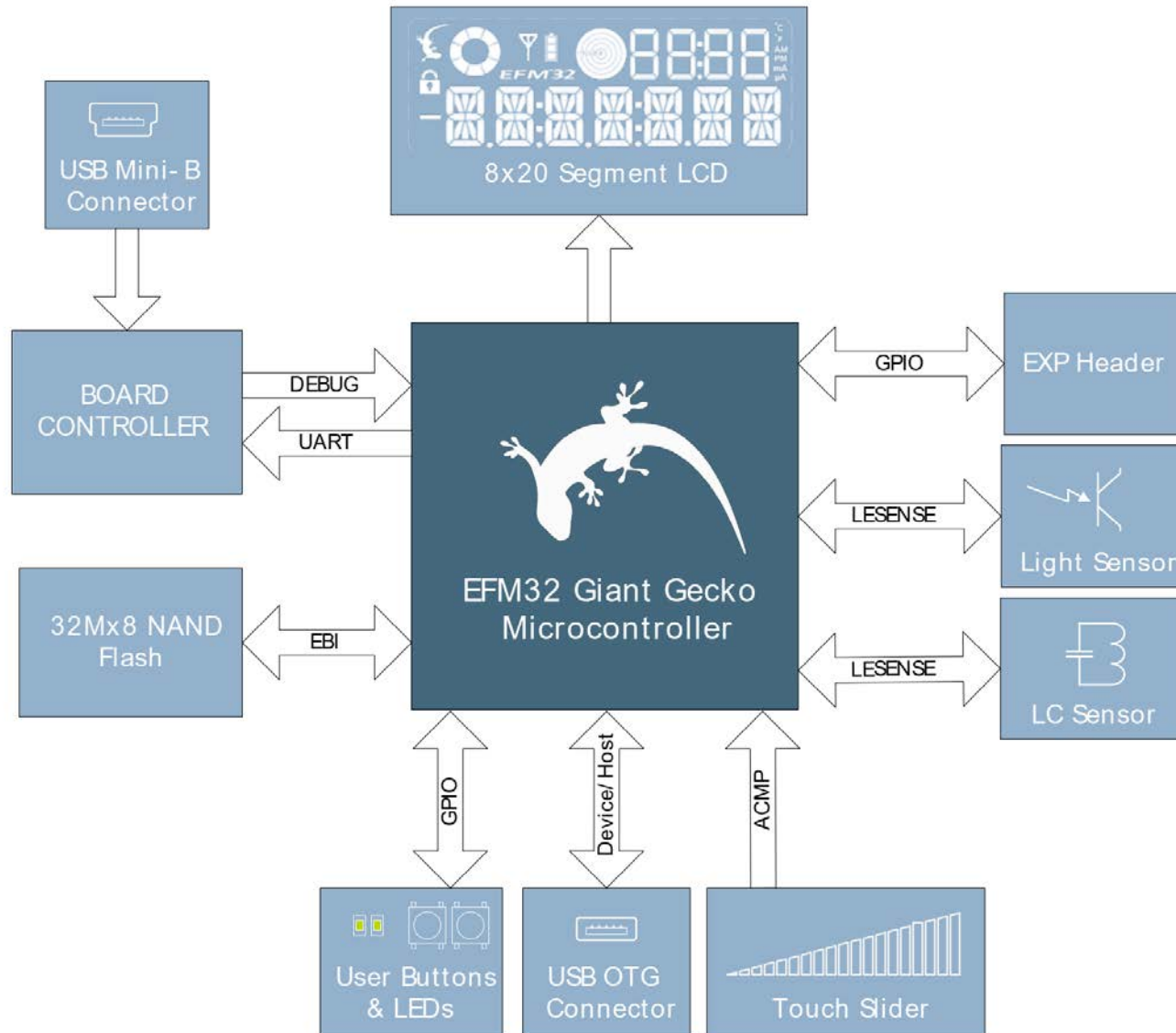


- <https://www.silabs.com/development-tools/mcu/32-bit/efm32gg-starter-kit>

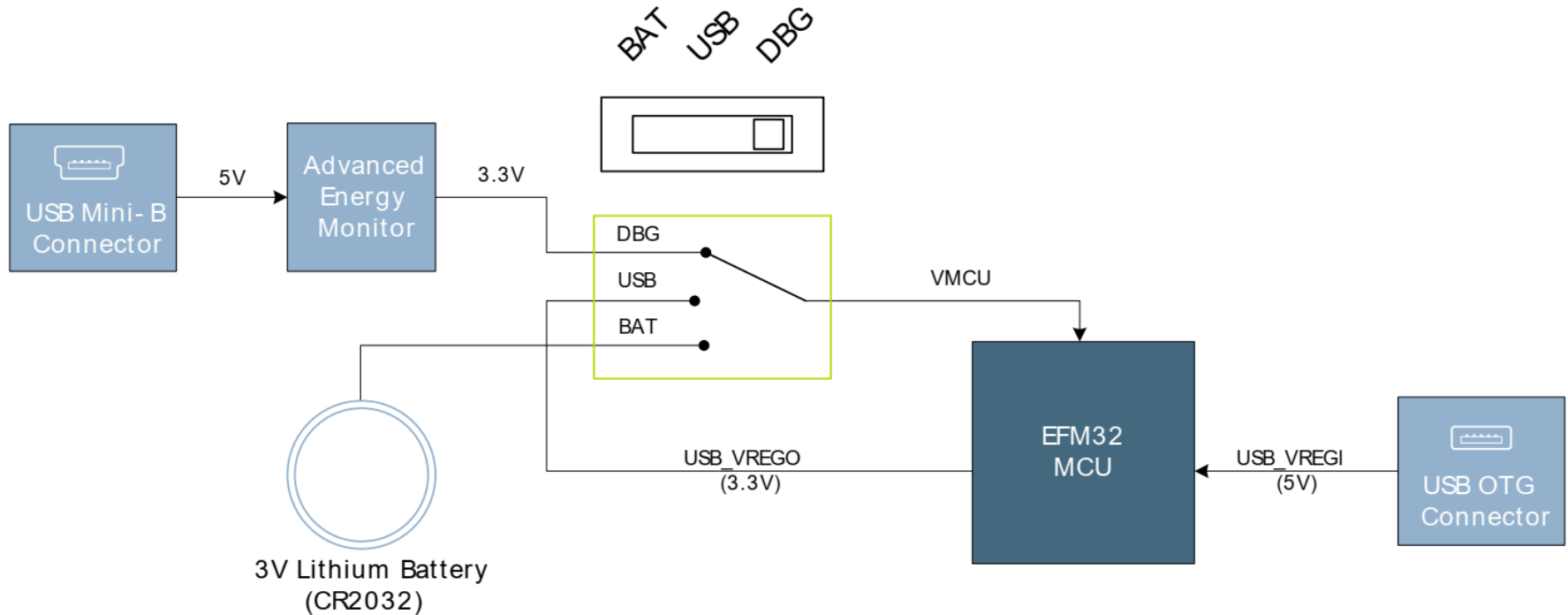
# 1.1) Main features

- 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

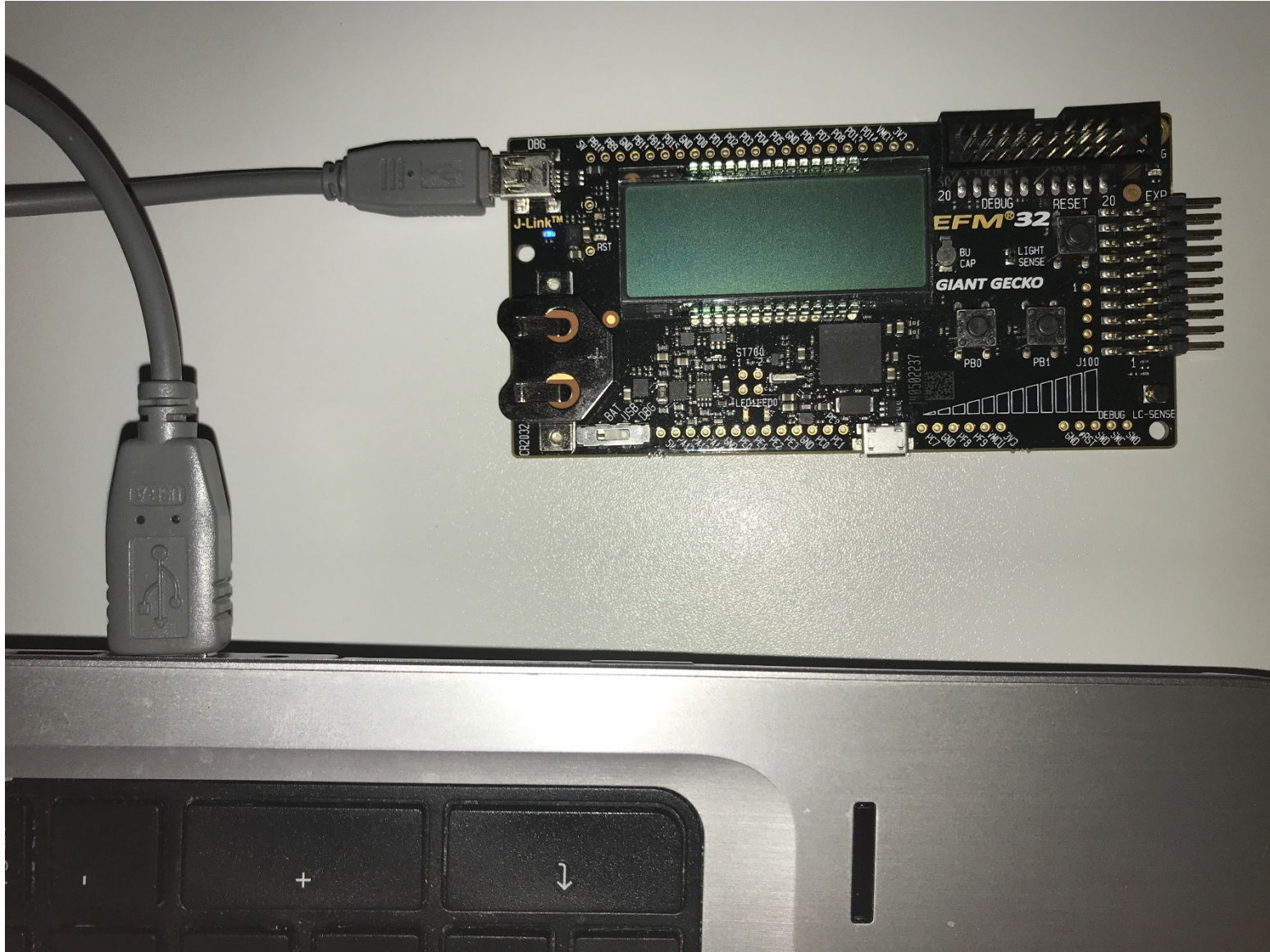


# 1.3) Power supply



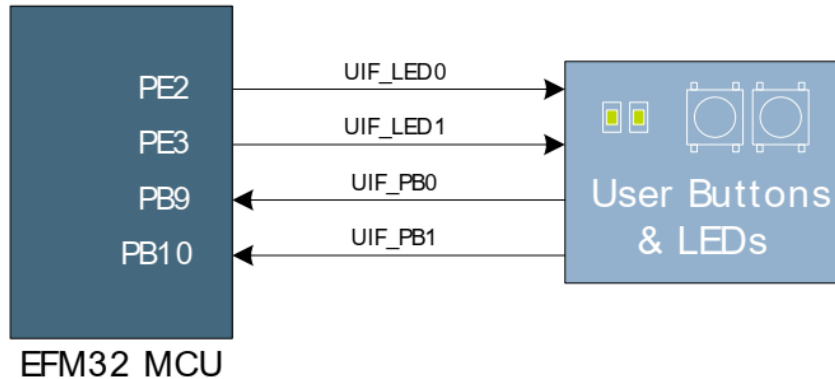
- DBG: via on-board debugger energy monitor can be used (use this)
- BAT: use CR2032 battery
- USB: MCU integrated voltage regulator is used

# 1.3) Power supply and proper connection



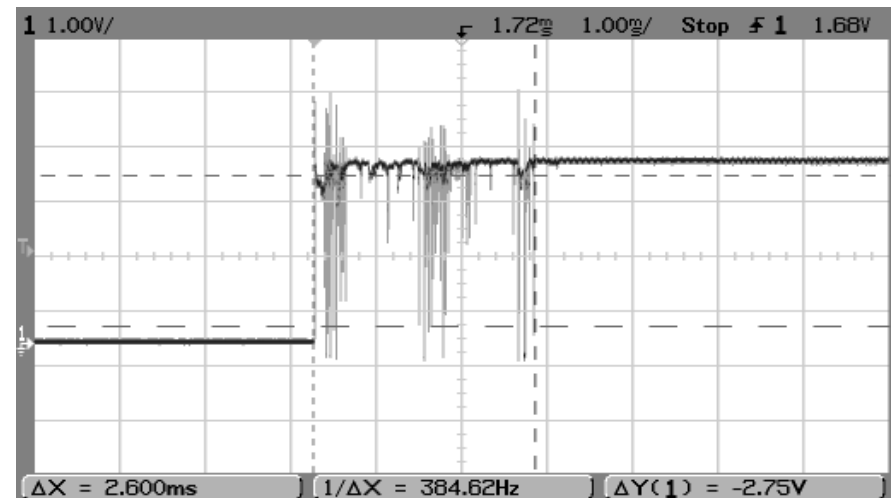
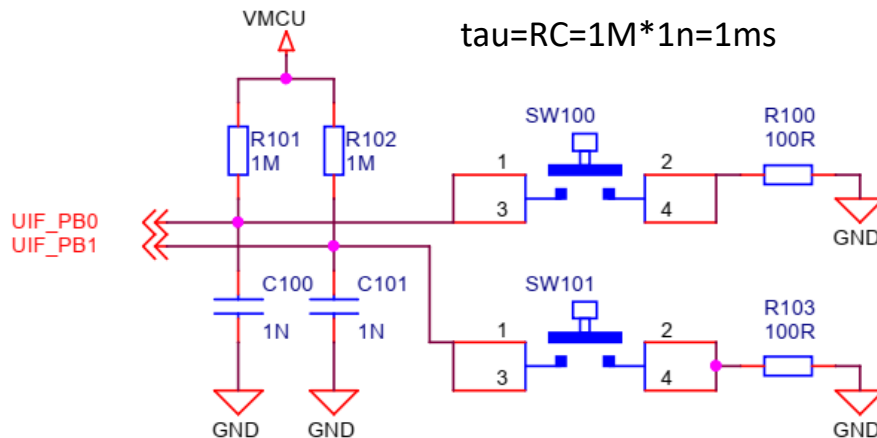


# 1.4) Peripherals-Buttons/LEDs

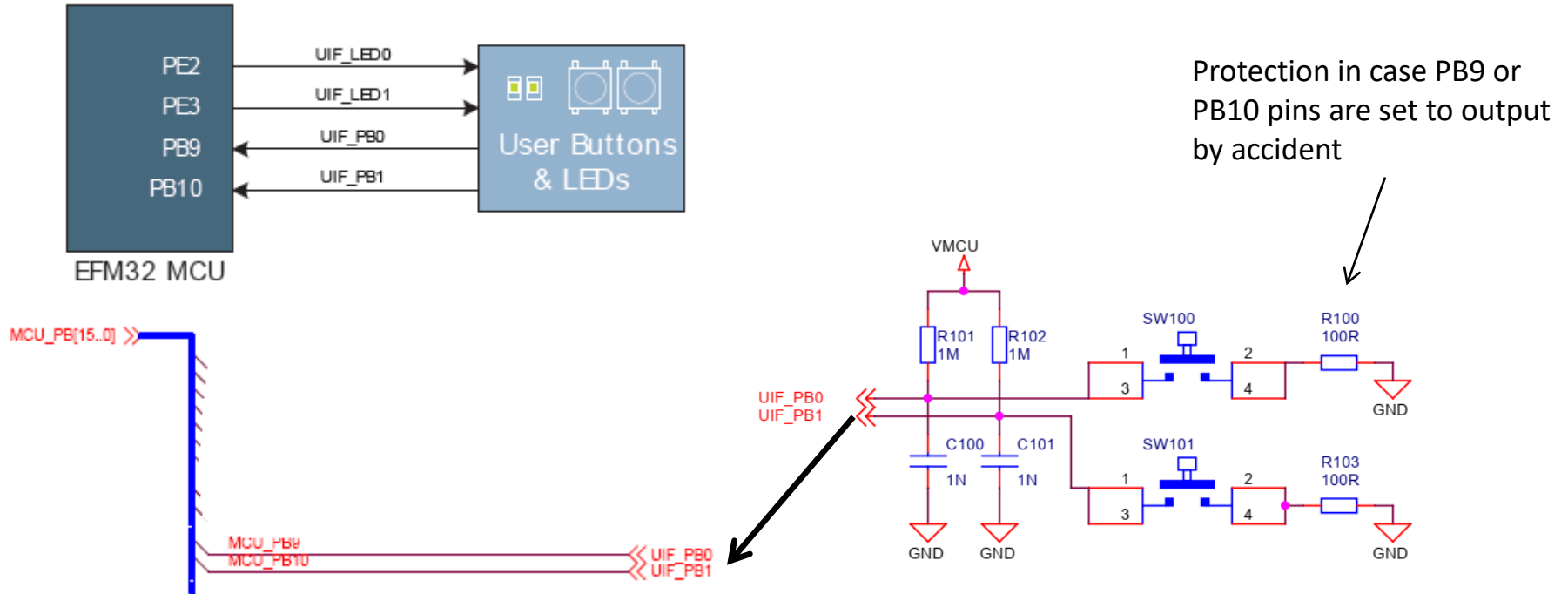


PB0=push button nr. 0  
PB9=9<sup>th</sup> bit of port B  
PE3=3<sup>rd</sup> bit of port E

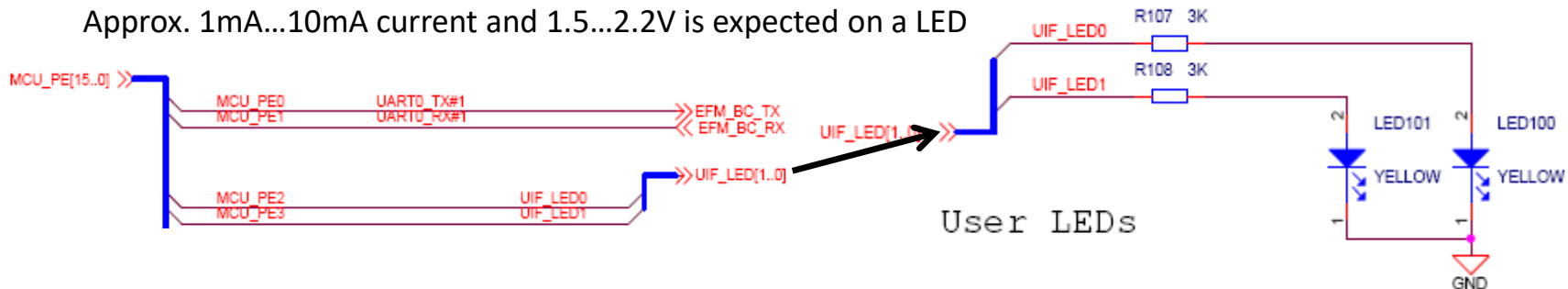
- Push buttons are debounced by RC filter to avoid:



# 1.4) Peripherals-Buttons/LEDs



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](http://www.silabs.com/products/development-tools/software/simplicity-studio)



# 2.1) Getting started with IDE-Launcher

The screenshot displays 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". The toolbar contains icons for "Sign In", "Search", and "Debug Adapters". The left sidebar shows a tree view under "Debug Adapters" with the following structure:

- J-Link Silicon Labs (440019119)
  - EFM32 Giant Gecko Starter Kit (EFM32GG-STK3700)
    - EFM32 Giant Gecko Starter Kit board (BRD2200A Rev A03)
      - EFM32GG990F1024

A red box highlights this tree view with the text "Connected board". The main workspace area displays "Welcome to Simplicity Studio" and a message: "To view content, select a kit or board in the Debug Adapters or My Products view." Below this, there are buttons for "New Project" and "Recent Projects". The bottom section of the workspace is divided into three tabs: "Getting Started", "Documentation", and "Compatible Tools". The "Getting Started" tab is active and contains three columns: "Demos", "Software Examples", and "SDK Documentation". Each column has a sub-header and a "Change Preferred SDK" link. A red box highlights this entire bottom section with the text "For your help". The top right corner of the IDE shows a window titled "Views (Launcher is now active)" with buttons for "Launcher", "Simplicity IDE", and "Debug". A red box highlights this area. The bottom right corner of the IDE shows the copyright notice "© 2020 Silicon Labs".

# 2.2) Getting started with IDE-Simplicity IDE

View (Simplicity IDE active)

The screenshot displays the Simplicity IDE interface with several key components highlighted by red rounded rectangles:

- Project Explorer:** Shows the project structure for 'STK3700\_blink\_2' under the 'GNU ARM v7.2.1 - Debug' configuration. The 'src' folder is expanded.
- Source Editor:** Displays the source code for 'blink.c', which includes license information and code for LED blink control using a timer.
- Debug Adapters:** Shows the 'J-Link Silicon Labs' adapter and the selected hardware target: 'EFM32 Giant Gecko Starter Kit board (BRD2200A Rev A03)'. The specific board ID 'EFM32GG990F1024' is also visible.
- Console:** Shows the output from the 'Adapter Pack Console', providing board details such as board ID, name, description, revision, serial number, date, and part number.

```
*****  
* @file  
* @brief Simple LED Blink Demo for EFM32GG_STK3700  
*****  
* # License  
* <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>  
*****  
*  
* The licensor of this software is Silicon Laboratories Inc. Your use of this  
* software is governed by the terms of Silicon Labs Master Software License  
* Agreement (MSLA) available at  
* www.silabs.com/about-us/legal/master-software-license-agreement. This  
* software is distributed to you in Source Code format and is governed by the  
* sections of the MSLA applicable to Source Code.  
*  
*****/  
  
#include <stdint.h>  
#include <stdbool.h>  
#include "em_device.h"  
#include "em_chip.h"  
#include "em_cm_u.h"  
#include "em_emu.h"  
#include "bsp.h"  
#include "bsp_trace.h"  
  
volatile uint32_t msTicks; /* counts 1ms timeTicks */  
  
void Delay(uint32_t dlyTicks);  
  
*****  
* @brief SysTick Handler  
*****
```

Adapter Pack Console  
boardId[0]=2200A  
boardName[0]=BRD2200A Rev. A03  
boardDescription[0]=EFM32 Giant Gecko Starter Kit board  
boardRevision[0]=A03  
boardSerial[0]=140502237  
boardDate[0]=2014/2/6  
inferPart[0]=yes

# 2.3) Getting started with IDE-Debug

Run      Debug deploy and run

View (Debug active)

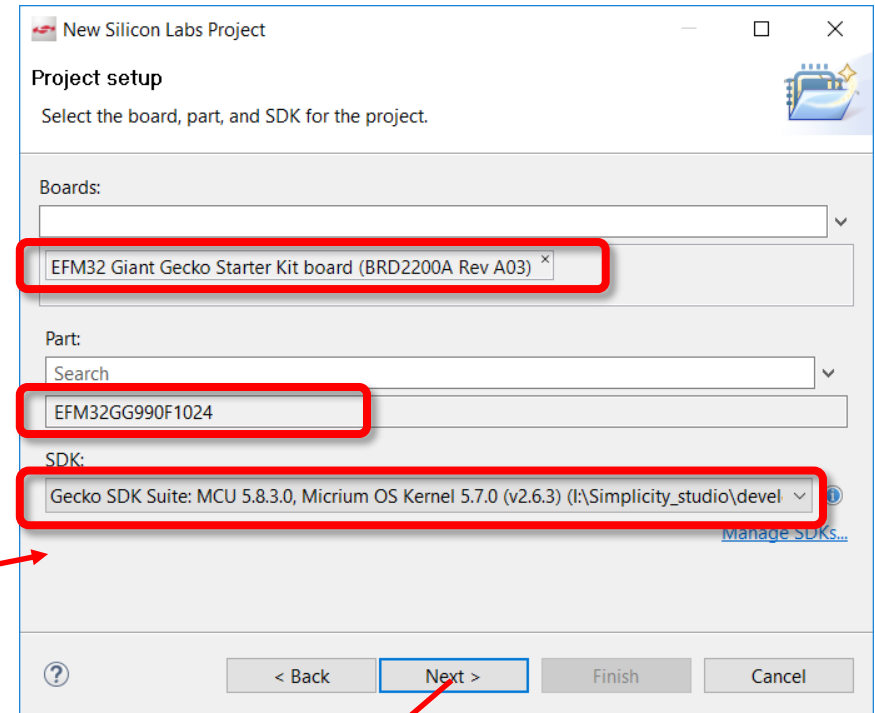
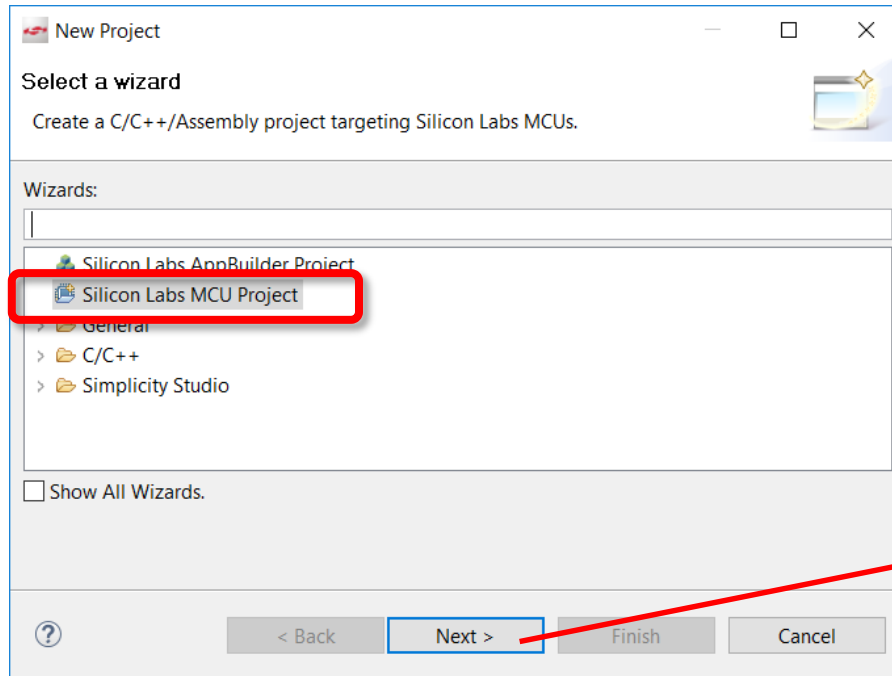
The screenshot shows the SImplicity Studio IDE interface during a debug session. Several key components are highlighted with red boxes:

- Run and Debug Buttons:** The 'Run' (play) and 'Debug' (bug) buttons in the top toolbar.
- Project Explorer:** Shows the project structure for 'Silicon Labs ARM MCU: EFM32GG990F1024' and the current file 'main() at blinkc:56 0x1368'.
- Variables Panel:** A table showing the current state of variables.

Name	Type	Value	Location
msTicks	volatile uint32_t	0	0x20000090
- Source Code Editor:** Displays the C code for 'blinkc.c', with the 'main' function visible. The code includes comments and function calls like 'CHIP\_Init()' and 'BSP\_TraceProfilerSetup()'. A red box highlights the entire code editor area.
- Disassembly Window:** Shows the assembly code for the 'main' function, starting at address 00001368. The first instruction is 'push {r7,lr}'. A red box highlights the disassembly window.
- Console:** The 'Program Output Console' is visible at the bottom, currently empty.
- Debug Active Indicator:** The 'Debug' button in the top right corner is highlighted, indicating that the debugger is active.

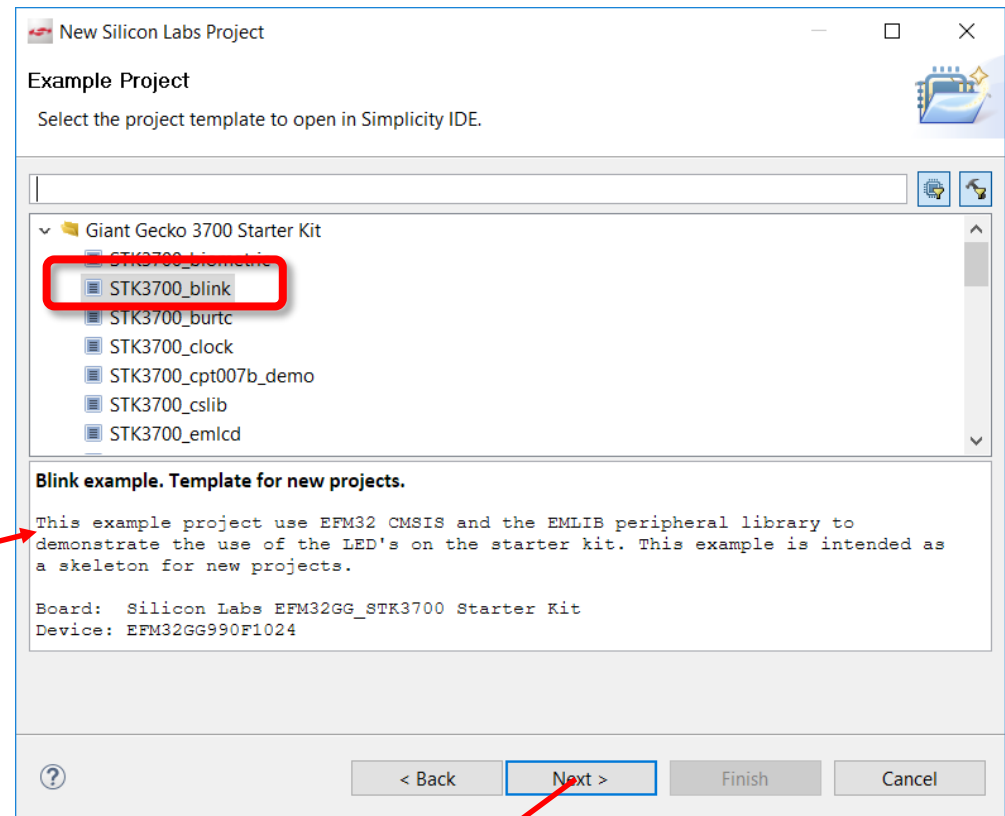
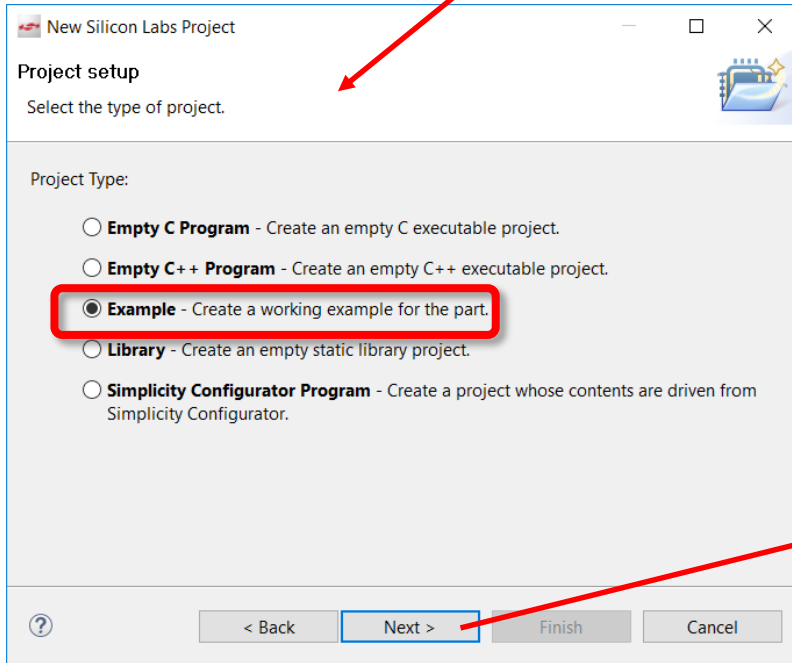
# 3) Start a new project

## ■ File->New->Project:

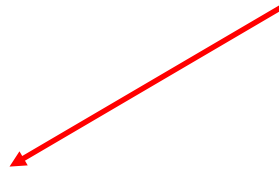




# 3) Start a new project



# 3) Start a new project



New Silicon Labs Project

Project Configuration

Select the project name and location.

Project name:

Use default location

Location:  Browse...

With project files:

Link to sources

Link sdk and copy project sources

Copy contents

# 4) Example project created

The screenshot displays the SImplicity 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 [GNU ARM v7.2.1 - Debug] [EFM32GG990F1024]', with folders for Binaries, Includes, BSP, CMSIS, emlib, GNU ARM v7.2.1 - Debug, and src. The main editor window shows the source code for 'blink.c', which includes a license notice from Silicon Laboratories Inc. and C code for a blink demo. The code includes headers like <stdint.h>, <stdbool.h>, and various project-specific headers. It defines a 'msTicks' variable and a 'Delay' function. The bottom of the IDE shows the Debug Adapters pane with 'J-Link Silicon Labs (440019119) [Simplicity Debugger]' and 'EFM32 Giant Gecko Starter Kit board (BRD2200A Rev A03)'. The Console pane is empty, and the status bar at the bottom indicates '0 items selected' and '8 new notifications'.

```

/*****
 * @file
 * @brief Simple LED Blink Demo for EFM32GG_STK3700
 *****/
# License
 * <b>Copyright 2018 Silicon Laboratories Inc. www.silabs.com</b>
 *****/
 *
 * The licensor of this software is Silicon Laboratories Inc. Your use of this
 * software is governed by the terms of Silicon Labs Master Software License
 * Agreement (MSLA) available at
 * www.silabs.com/about-us/legal/master-software-license-agreement. This
 * software is distributed to you in Source Code format and is governed by the
 * sections of the MSLA applicable to Source Code.
 *****/

#include <stdint.h>
#include <stdbool.h>
#include "em_device.h"
#include "em_chip.h"
#include "em_cm0.h"
#include "em_emu.h"
#include "bsp.h"
#include "bsp_trace.h"

volatile uint32_t msTicks; /* counts lms timeTicks */

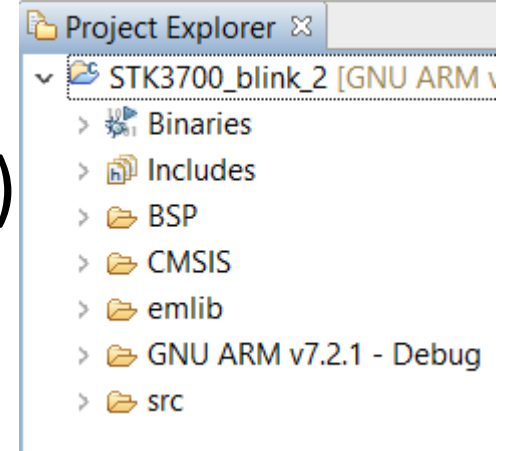
void Delay(uint32_t dlyTicks);

/*****
 * @brief SysTick Handler
 *****/










```

# 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 <b>[Debug]</b> 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 <b>[Resume]</b> button runs the MCU after reset or after hitting a breakpoint.
	Suspend	The <b>[Suspend]</b> button halts the MCU.
	Disconnect	The <b>[Disconnect]</b> 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 <b>[Reset the Device]</b> button performs a hardware reset on the MCU.
	Step Into	The <b>[Step Into]</b> button single steps into the first line of a function.
	Step Over	The <b>[Step Over]</b> button single steps over a function, executing the entire function.
	Step Return	The <b>[Step Return]</b> button steps out of a function, executing the rest of the function.
	Instruction Stepping Mode	The <b>[Instruction Stepping Mode]</b> button toggles assembly single stepping. When enabled, single steps will execute a single assembly instruction at a time. See the <b>[Disassembly]</b> view for the assembly code corresponding to the source code at the current line of execution.

# 4.2.1) Breakpoints

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 toolbar contains various icons for file operations and debugging. The main workspace is divided into several panes:

- Project Explorer:** Shows the project structure for 'Silicon Labs ARM MCU: EFM32GG990F1024' and the file 'STK3700\_blink\_2.axf' with 'main() at blinkc:56 0x1368'.
- Breakpoints:** A window showing a single breakpoint set on 'blink.c [line: 35]'. This window is highlighted with a red box.
- Source Code:** The 'blink.c' file is open, showing C code. Line 35, 'void SysTick\_Handler(void)', is highlighted with a red box. A red arrow points from the breakpoint in the Breakpoints window to this line.
- Disassembly:** A window showing assembly code for the 'main' function, starting at address 00001368.
- Console:** A 'Program Output Console' is visible at the bottom.

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

# 4.2.2) Register values

The screenshot shows the Simplicity Studio IDE interface. The top-left pane displays the Project Explorer with the file structure for 'STK3700\_blink\_2.axf'. The top-right pane, titled 'Registers', is highlighted with a red circle and contains the following table:

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', including functions like 'Delay' and 'SysTick\_Handler'. The bottom-right pane shows the disassembly of the 'main' function, with instructions such as 'push {r7,lr}', 'add r7,sp,#0x0', and 'CHIP\_Init()'. The bottom-most pane is the 'Program Output Console'.

- 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 main workspace is divided into several panes:

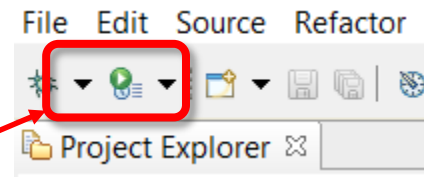
- Project Explorer:** Shows the project structure for 'Silicon Labs ARM MCU: EFM32GG990F1024' and the file 'STK3700\_blink\_2.axf'.
- Expressions Window:** A table with columns 'Expression', 'Type', 'Value', and 'Address'. It contains a single row with a plus icon and the text 'Add new expression'. This window is highlighted with a red rounded rectangle.
- Code Editor:** Displays the source code for 'blink.c', showing a `Delay` function and a `SysTick_Handler` function.
- Disassembly Window:** Shows the assembly code for the `main` function, starting at address 00001368.
- Console:** Shows the 'Program Output Console' at the bottom.

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



# 5) Energy profiler

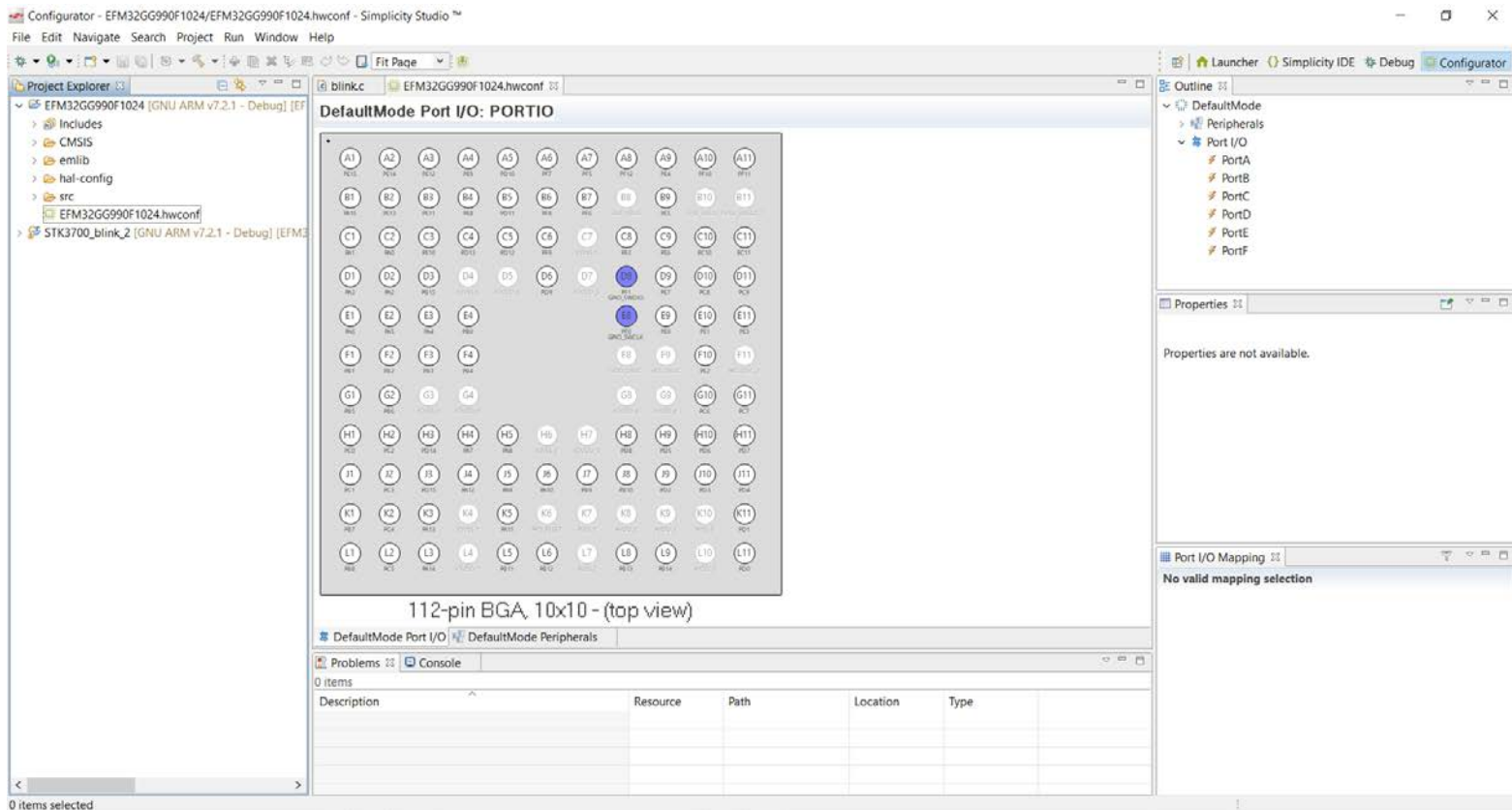
- Disable one LED (use e.g. comment `//`)
- Switch IDE mode and choose this icon



C	Function	Energy	Contribution (%)
JW	0xC8000000-0xC8000FFF	24.70 µWh	33.225%
JW	BSP_TraceProfilerSetup	1.72 µWh	2.311%
JW	0x013A8000-0x013A8FFF	159.12 nWh	0.214%
JW	0x00013000-0x00013FFF	135.76 nWh	0.182%

# 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