

Embedded and ambient systems

2024.09.04.

Practice 1



Méréstechnika és
Információs Rendszerek
Tanszék

Preliminary

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

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.

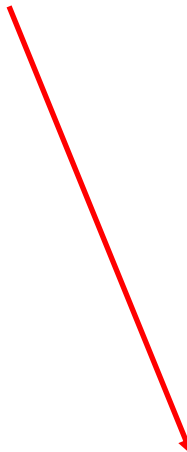
The following main topics are introduced:

- 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.

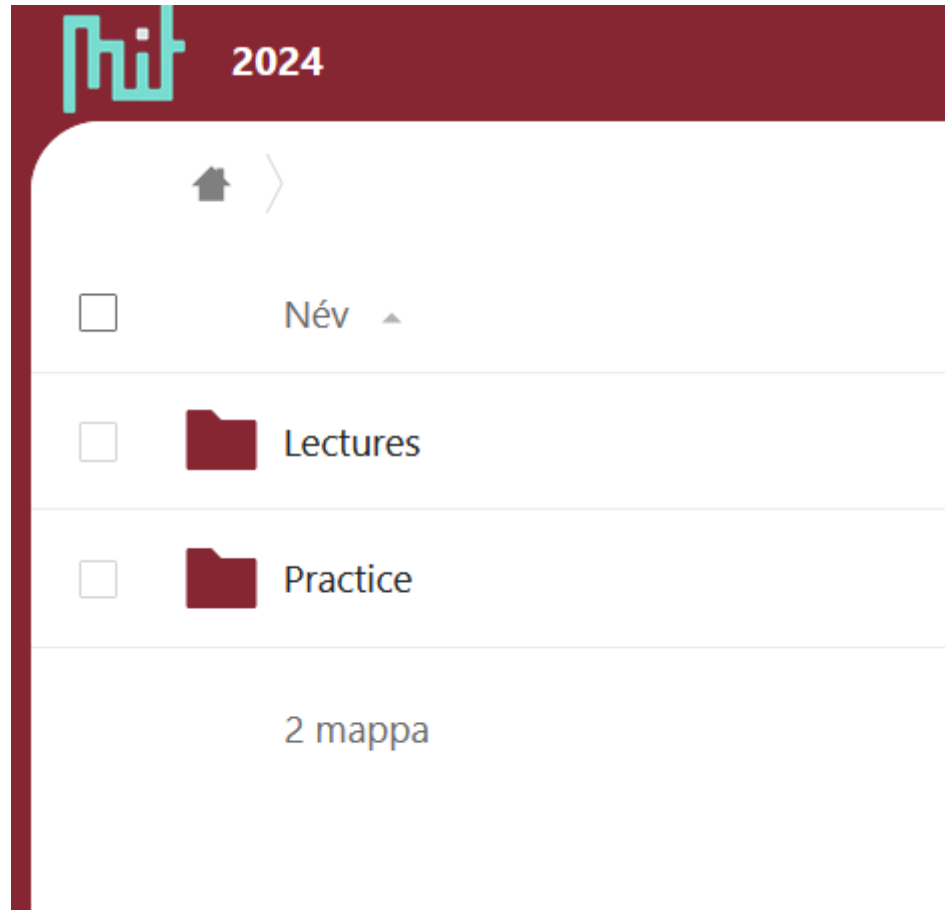
[Lectures and Practices materials](#)

[Exam information](#)
[Midterm information](#)
[References materials](#)

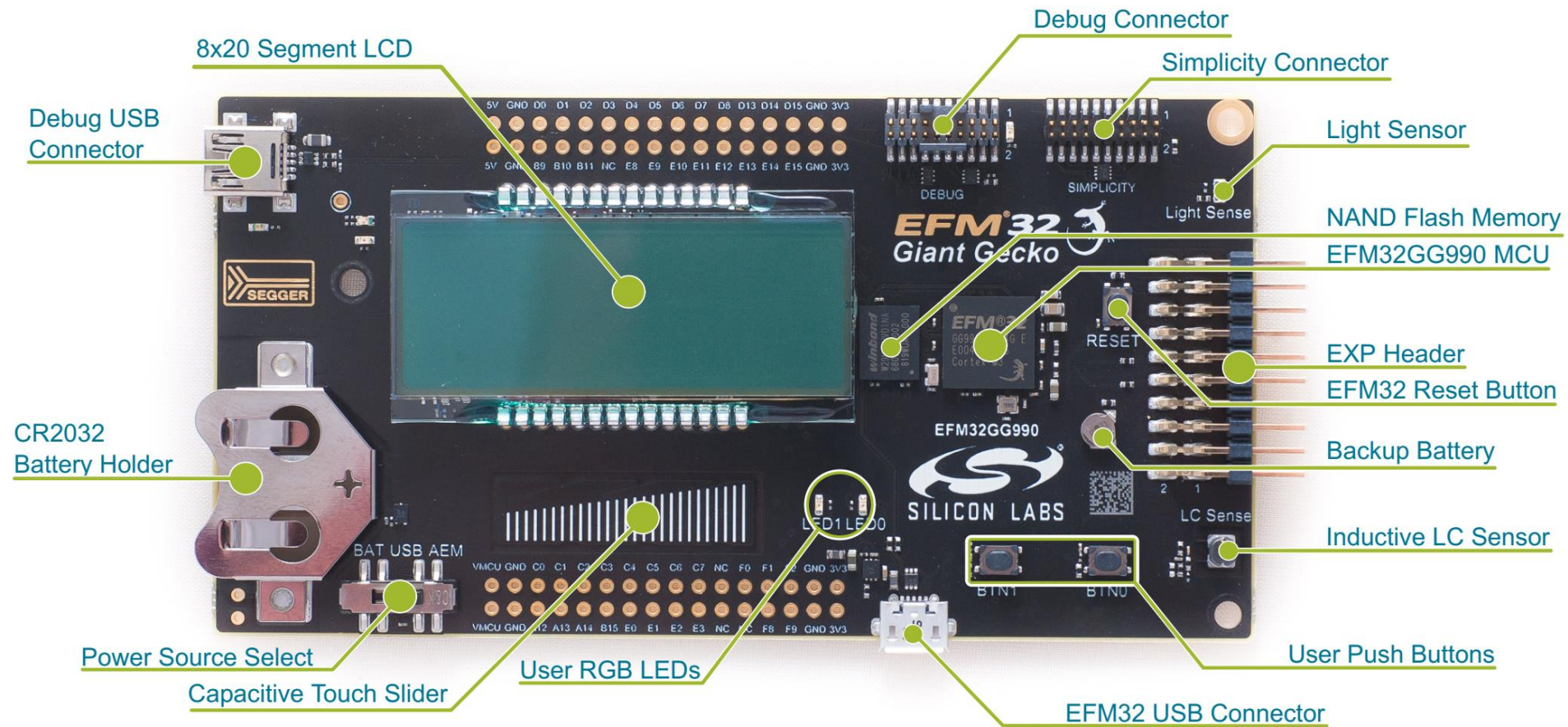
- See menu at the bottom



Preliminary



1) Development board: EFM32GG-STK3700

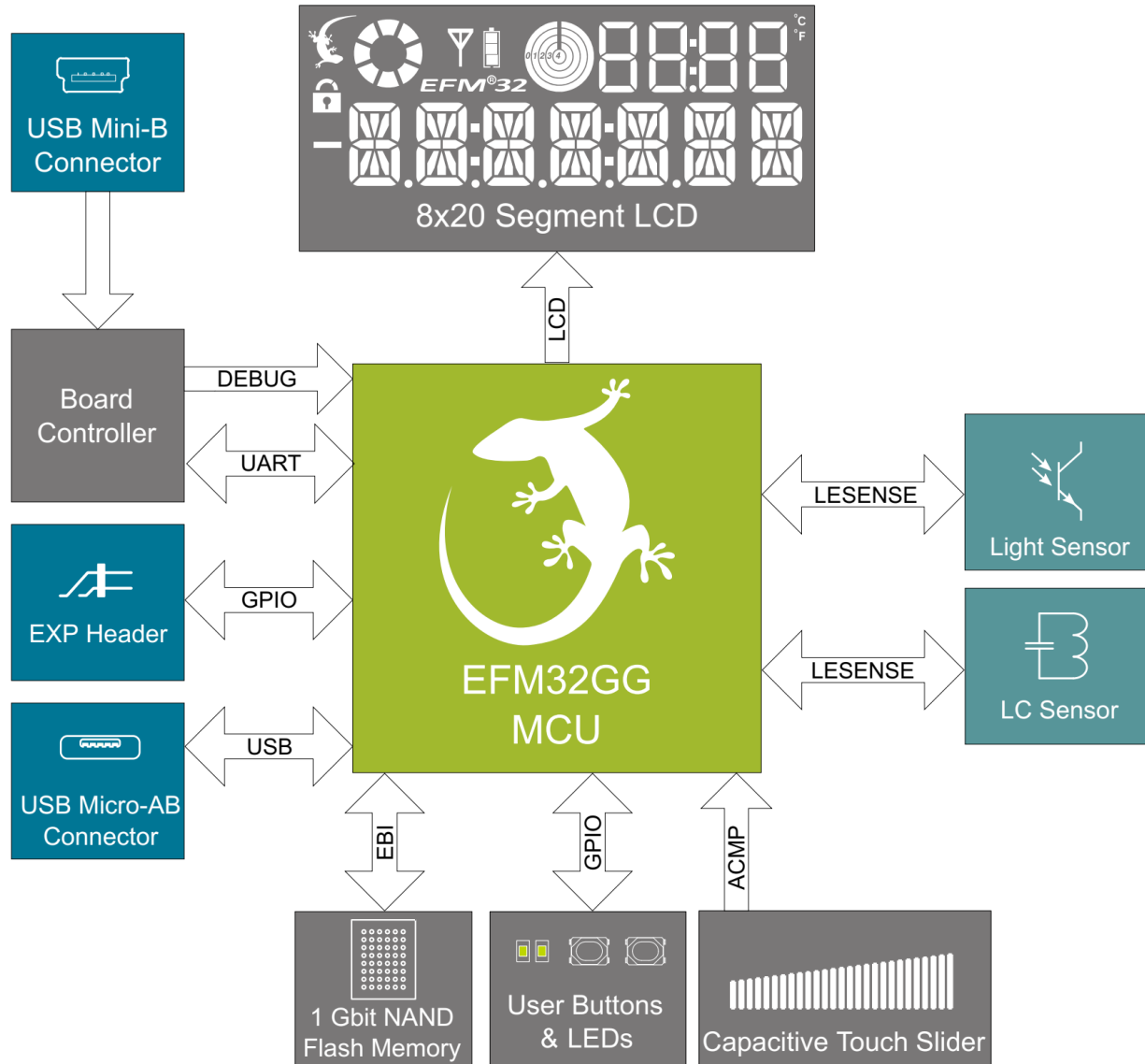


- <https://www.silabs.com/development-tools/mcu/32-bit/efm32gg-starter-kit>

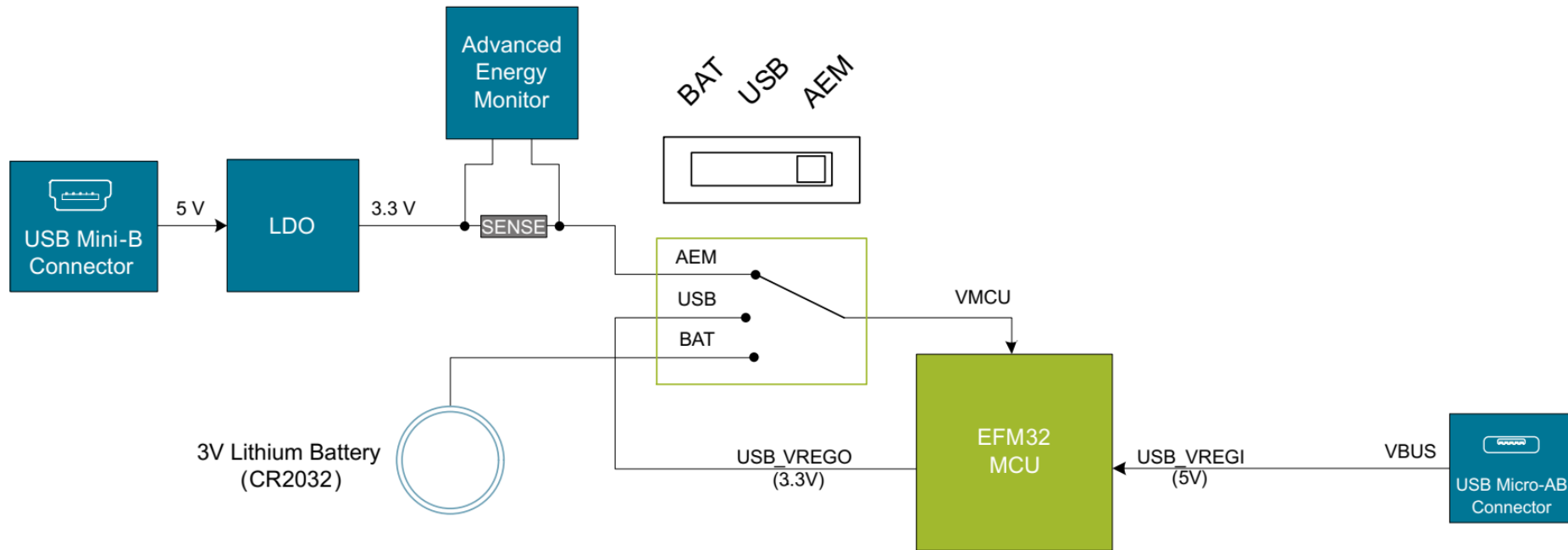
1.1) Main features

- 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

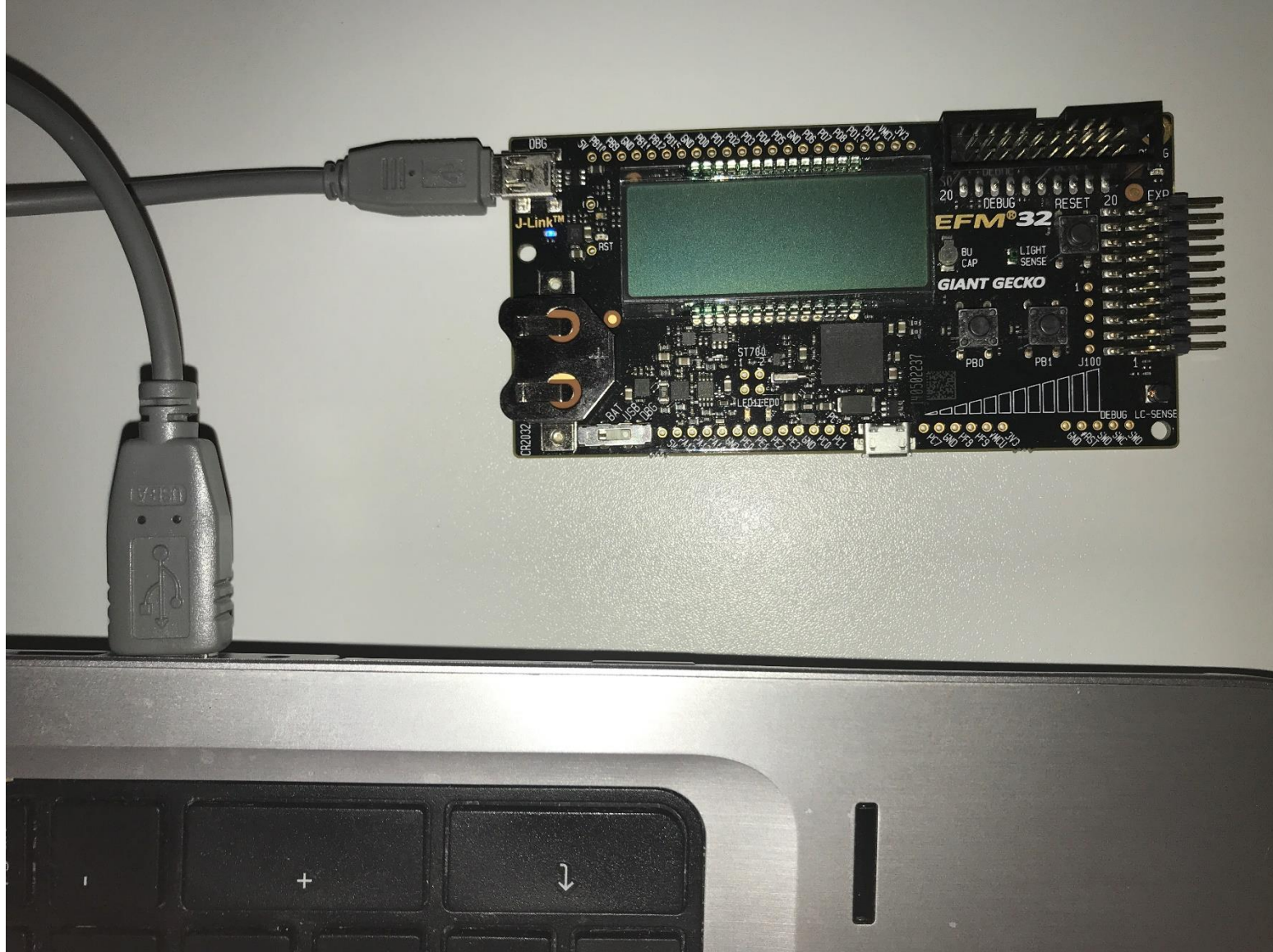


1.3) Power supply

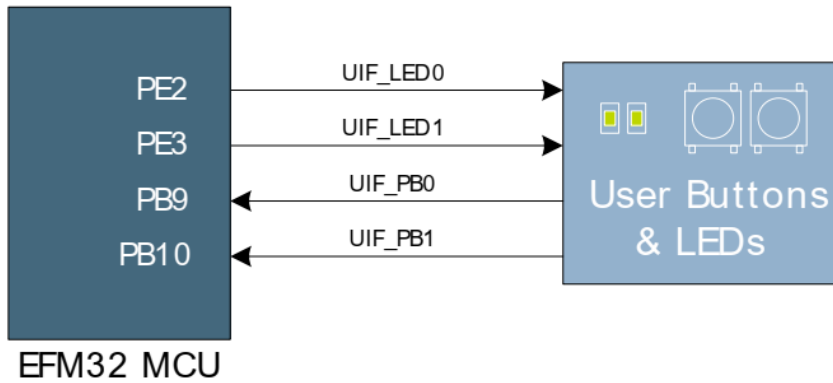


- DBG: via on-board debugger energy monitor can be used (use this)
- BAT: use CR2032 battery
- USB: MCU integrated voltage regulator is used

1.3) Power supply and proper connection

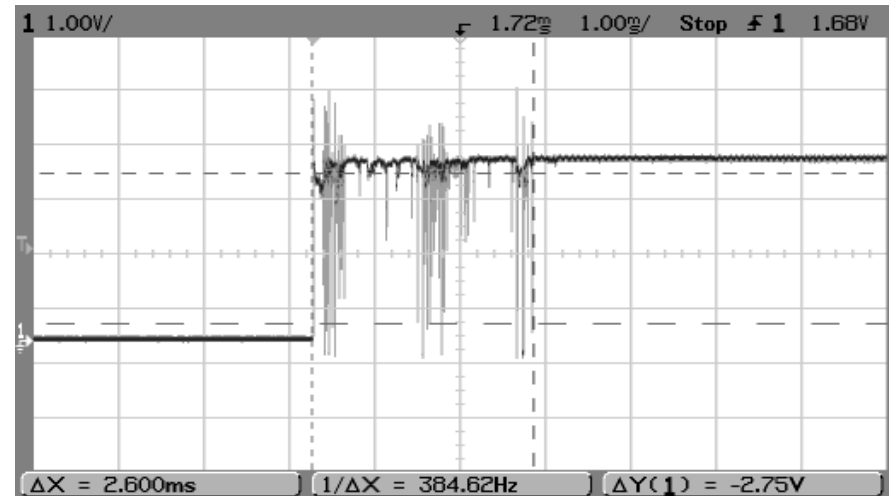
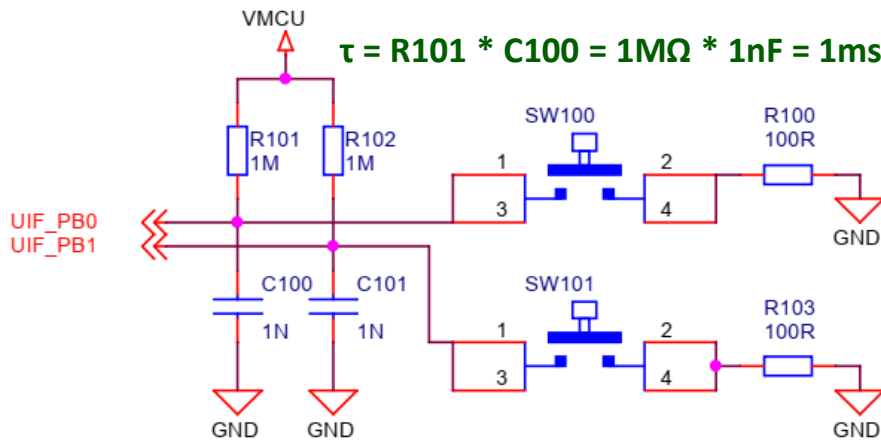


1.4) Peripherals-Buttons/LEDs

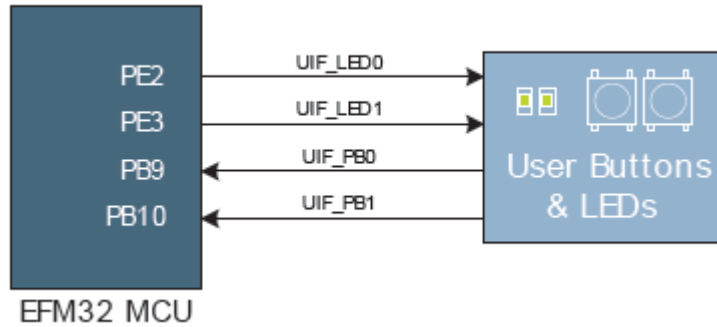


PB0=push button nr. 0
PB9=9th bit of port B
PE3=3rd bit of port E

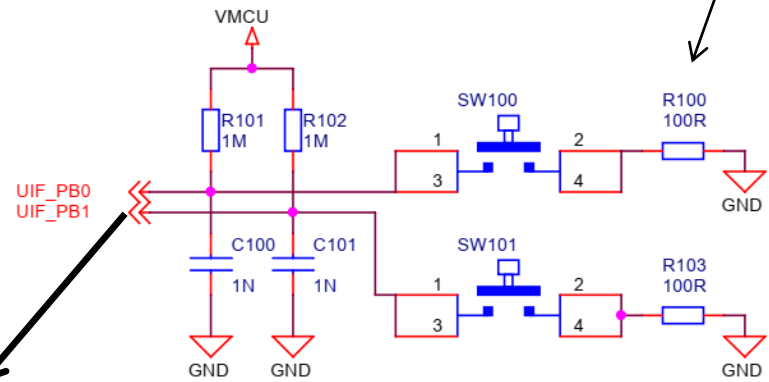
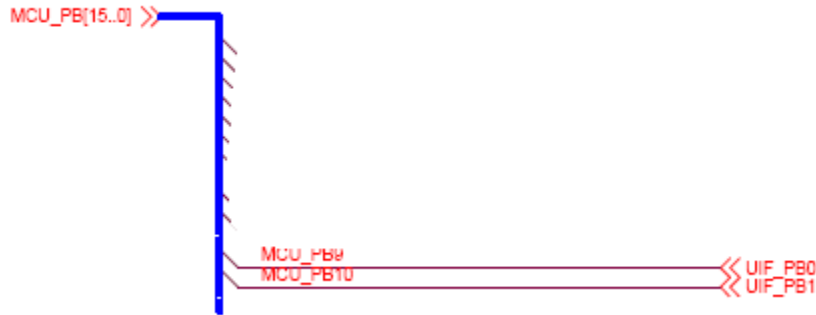
- Push buttons are debounced by RC filter to avoid bouncing:



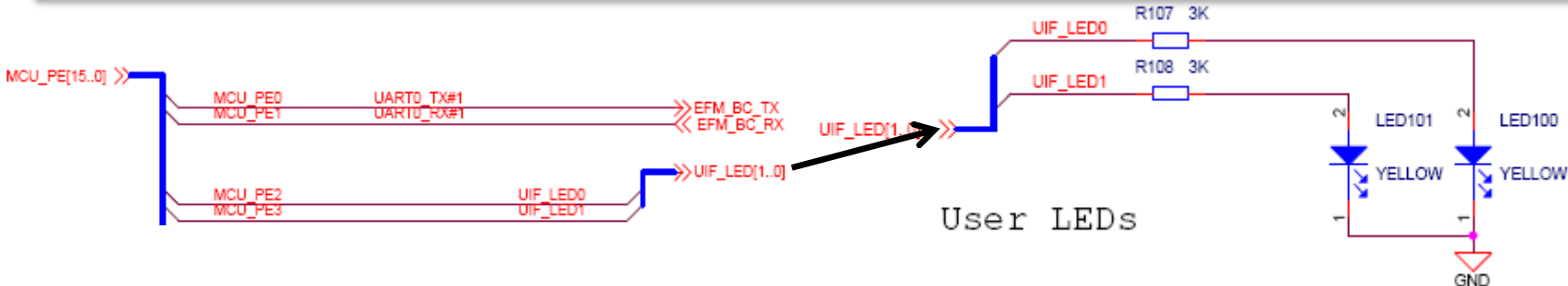
1.4) Peripherals-Buttons/LEDs



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



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

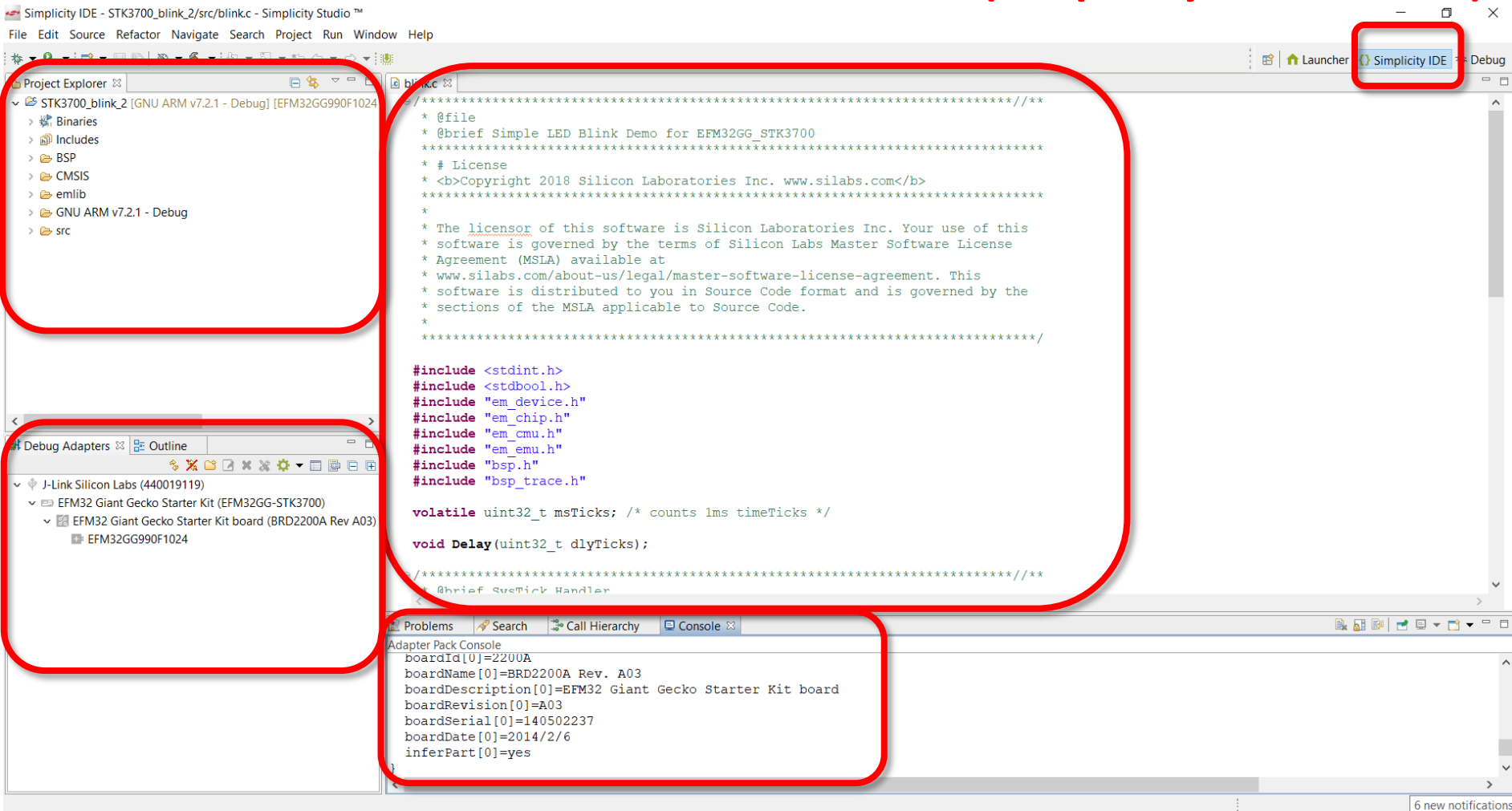
The screenshot displays the Simplicity Studio IDE interface. The title bar reads "Launcher - STK3700_blink_2/src/blink.c - Simplicity Studio™". The menu bar includes "File", "Edit", "Source", "Refactor", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains icons for "Sign In", "Launcher", "Simplicity IDE", and "Debug". The "Launcher" view is active, showing a tree view of "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

The text "Connected board" is overlaid in red on the selected board. The main content area displays "Welcome to Simplicity Studio" and instructions: "To view content, select a kit or board in the Debug Adapters or My Products view." Below this are buttons for "New Project" and "Recent Projects". The "Getting Started" tab is active, showing "Demos", "Software Examples", and "SDK Documentation" sections. Each section contains the text "Select a kit, board, or device to view content" and a link "Change Preferred SDK". The text "For your help" is overlaid in red on the "Demos" section. The "My Products" view is visible at the bottom left, with a search field "Enter product name". The bottom right corner shows the copyright notice "© 2020 Silicon Labs".

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 during a debug session. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons, including a red box around the Run button (a play icon) and the Debug button (a bug icon). The Project Explorer on the left shows the project structure for 'Silicon Labs ARM MCU: EFM32GG990F1024' and 'STK3700_blink_2.axf', with 'main() at blinkc:56 0x1368' selected. The Variables window in the center-right shows a table with columns for Name, Type, Value, and Location. The Source Editor at the bottom-left displays the C code for 'blinkc.c', with a red box around the main function. The Disassembly window at the bottom-right shows the assembly code for the 'main' function, with a red box around the code. The Console window at the bottom shows 'Program Output Console'.

| Name | Type | Value | Location |
|---------|-------------------|-------|------------|
| msTicks | volatile uint32_t | 0 | 0x20000090 |

```
/* @brief Main function
*****
*/
int main(void)
{
    /* Chip errata */
    CHIP_Init();

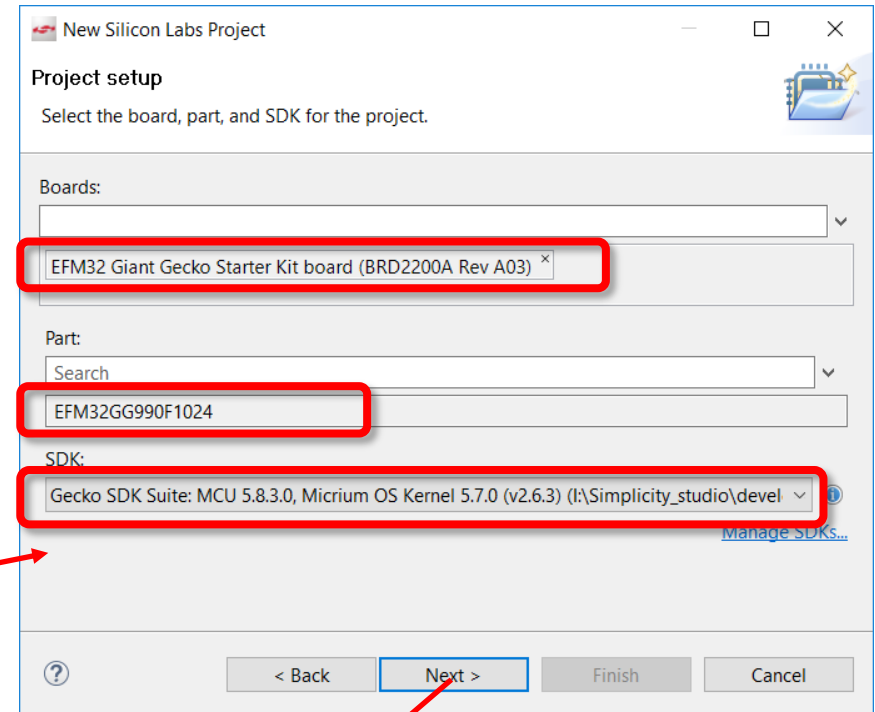
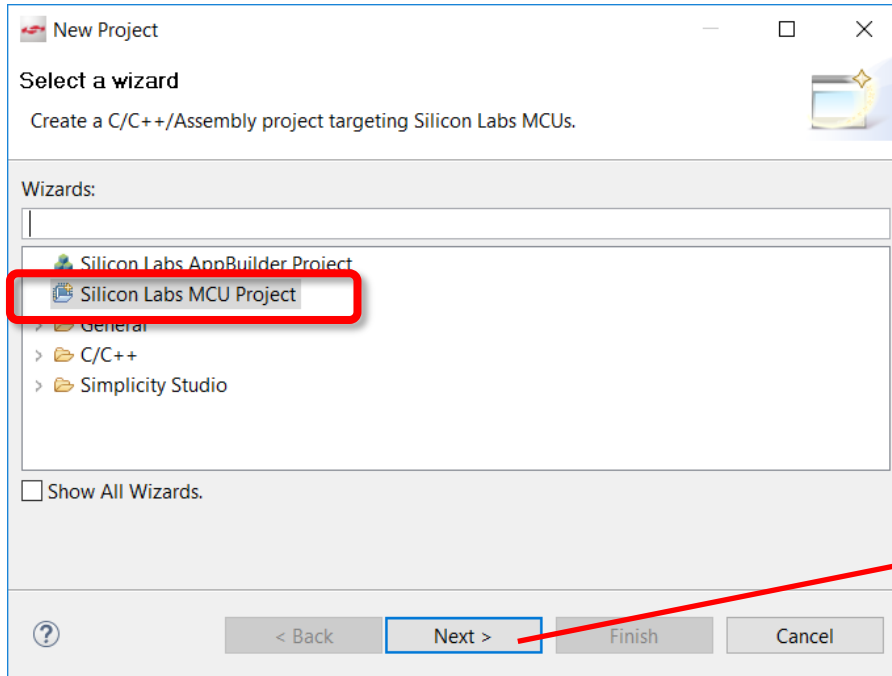
    /* If first word of user data page is non-zero, enable Energy Profiler trace */
    BSP_TraceProfilerSetup();

    /* Setup SysTick Timer for 1 msec interrupts */
    if (SysTick_Config(CMU_ClockFreqGet(cmuClock_CORE) / 1000)) {
        while (1);
    }
}
```

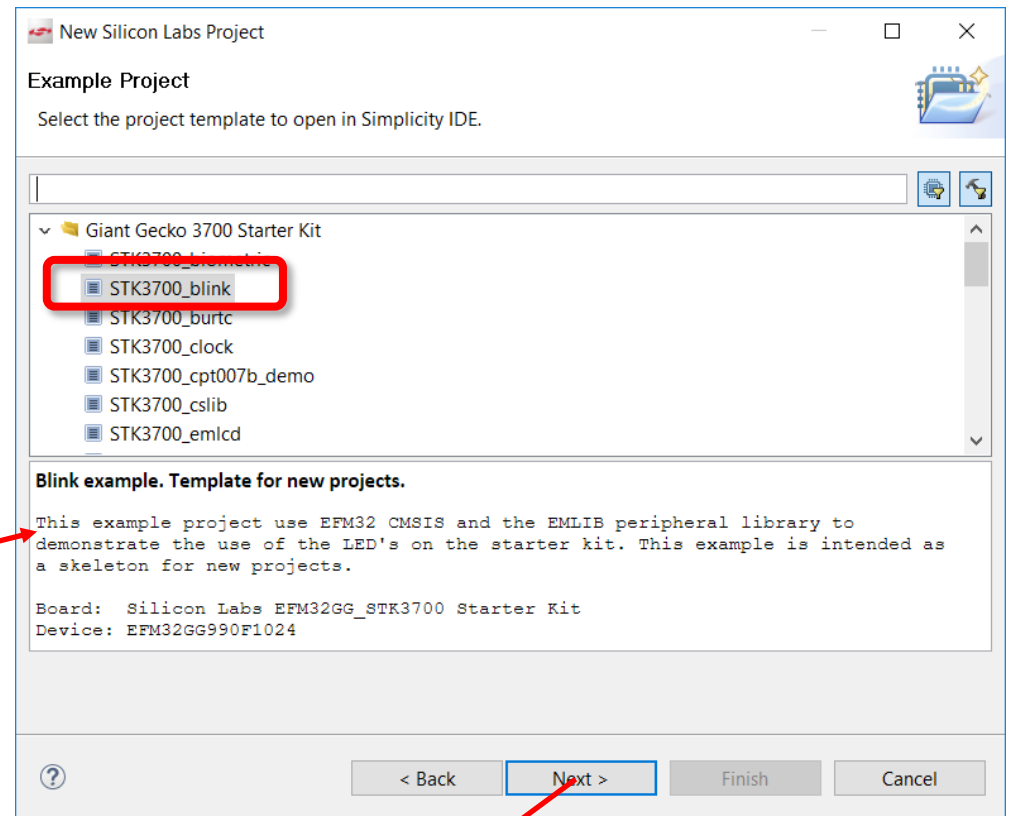
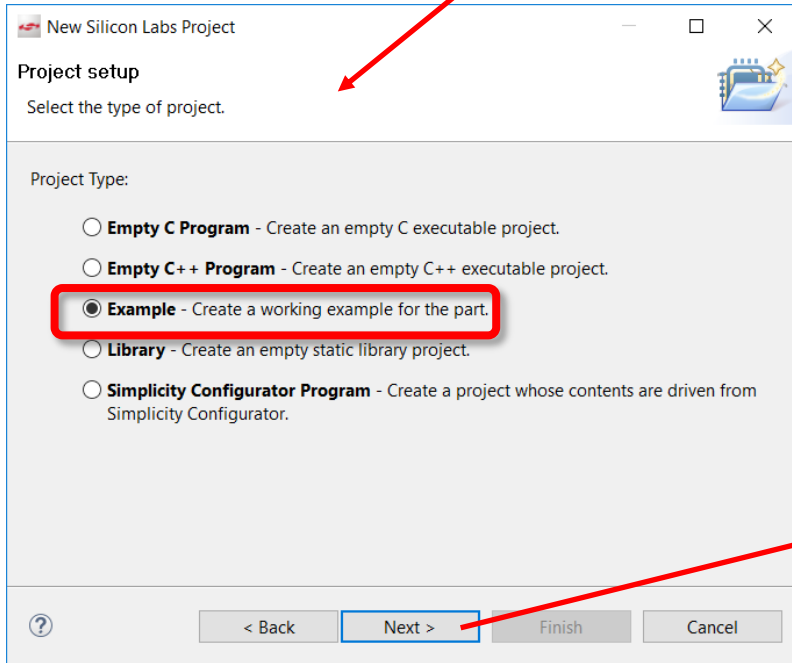
```
main:
00001368: push    {r7,lr}
0000136a: add    r7,sp,#0x0
58      CHIP_Init();
0000136c: bl     0x000012d8
61      BSP_TraceProfilerSetup();
00001370: bl     0x00000398
64      if (SysTick_Config(CMU_ClockFreqGet(cmuClock_CORE) / 1000)) {
00001374: ldr    r0,[pc,#0x3c] ; 0x13b0
00001376: bl     0x00000bc8
0000137a: mov    r2,r0
0000137c: ldr    r3,[pc,#0x38] ; 0x13b4
0000137e: umull r2,r3,r3,r2
00001382: lsr   r3,r3,#6
00001384: mov    r0,r3
```

3) Start a new project

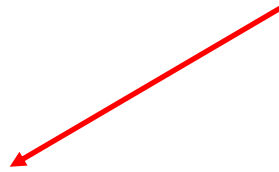
■ File->New->Project:



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 a project named 'STK3700_blink_2' with subfolders 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 several include directives for standard headers and project-specific files. The code also defines a volatile variable 'msTicks' and a 'Delay' function. The bottom status bar 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 */

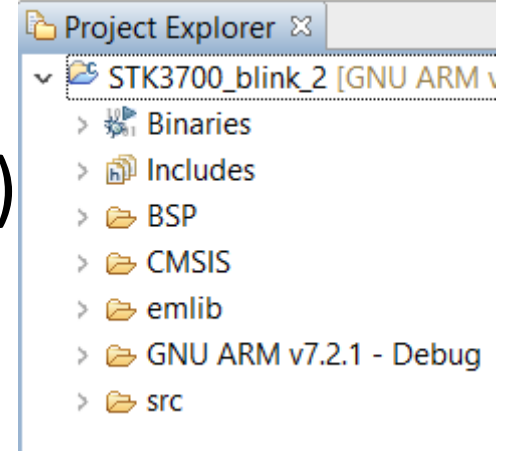
void Delay(uint32_t dlyTicks);

@*****/
* @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 [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

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'.
- Breakpoints:** A pane on the right showing a single breakpoint set at 'blink.c [line: 35]'. This pane is highlighted with a red box.
- Source Code:** The main editor shows the source code for 'blink.c'. Line 35, which contains the function definition 'void SysTick_Handler(void)', is highlighted with a red box. A red arrow points from the breakpoint in the Breakpoints pane to this line.
- Disassembly:** A pane on the right showing the assembly code for the 'main' function, starting at address 00001368.
- Console:** A pane at the bottom showing the program output.

- 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-left pane shows 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 data:

| 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 view for 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, which is currently empty.

- Register content can be manipulated

4.2.2) Expressions

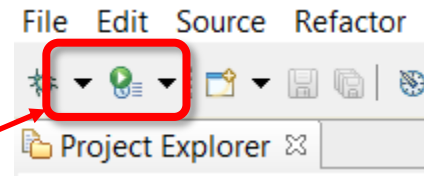
The screenshot displays the Simplicity Studio IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The main workspace is divided into several panes:

- Project Explorer:** Shows the project structure for 'STK3700_blink_2.axf' with the file 'main()' at address 0x1368 selected.
- Expressions Window (highlighted with a red box):** A table with columns for Expression, Type, Value, and Address. It contains a single row with the text 'Add new expression' and a plus icon. Below the table, it states 'No details to display for the current selection.'
- Source Code:** Shows the C code for 'main()' and 'SysTick_Handler()'. The current line of execution is at line 35.
- Disassembly:** Shows the assembly code for the 'main' function, starting at address 00001368. The current instruction is 'push {r7,lr}'.
- Console:** Shows the 'Program Output Console' which is currently empty.

- 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

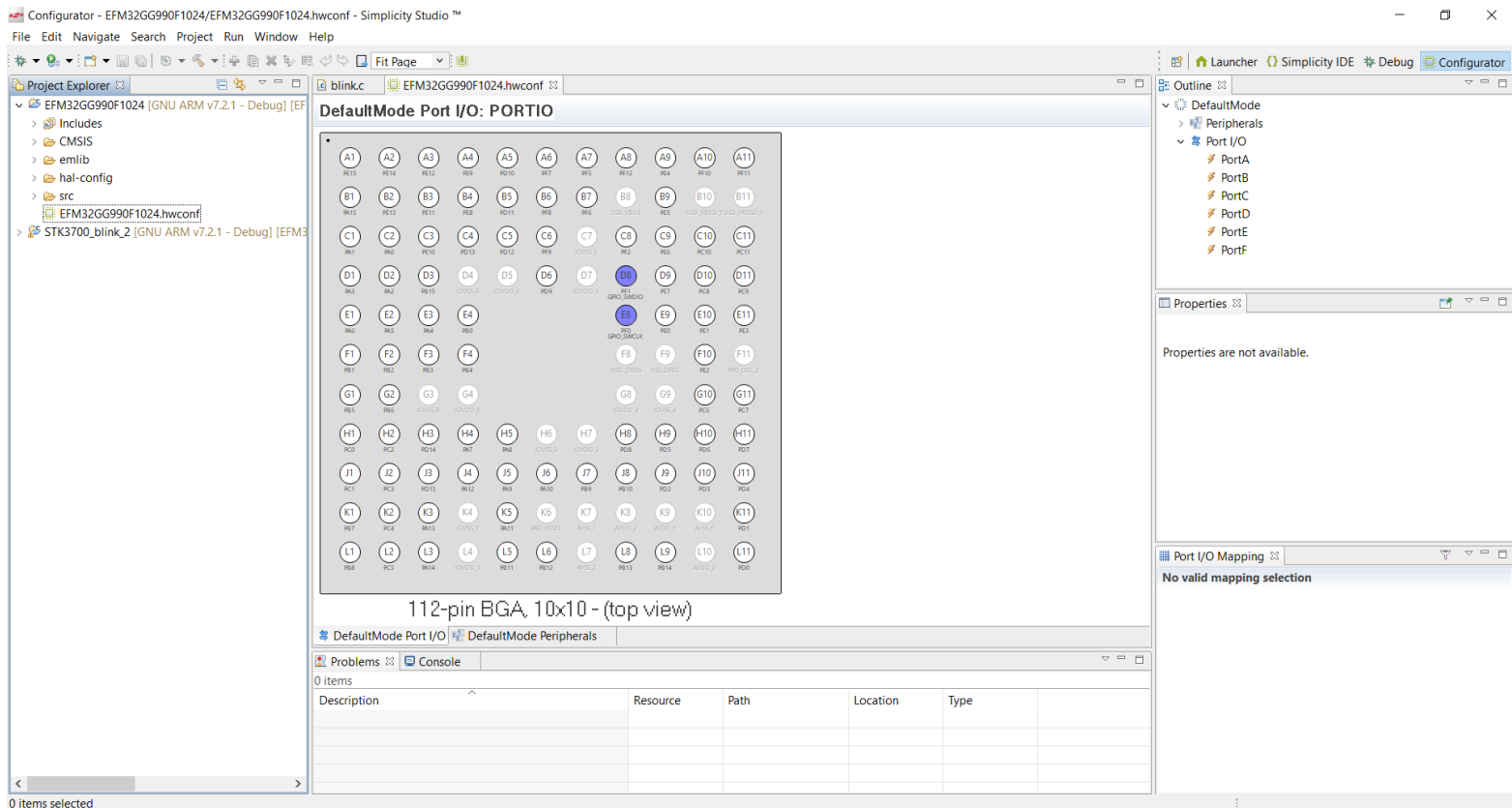


| Function | Energy | Contribution (%) |
|------------------------|------------|------------------|
| 0xC8000000-0xC8000FFF | 24.70 µWh | 33.225% |
| BSP_TraceProfilerSetup | 1.72 µWh | 2.311% |
| 0x013A8000-0x013A8FFF | 159.12 nWh | 0.214% |
| 0x00013000-0x00013FFF | 135.76 nWh | 0.182% |

```
blinck
37 msTicks++; /* incremen
38 }
39
40 /*****
41 * @brief Delays number of msT
42 * @param dlyTicks Number of t
43 *****/
44 void Delay(uint32_t dlyTicks)
45 {
46     uint32_t curTicks;
47
48     curTicks = msTicks;
49     while (msTicks - curTicks)
50 }
51
52 /*****
53 * @brief Main function
54 *****/
55 int main(void)
56 {
57     /* Chip errata */
58     CHIP_Init();
59
60     /* If first word of user dat
61     BSP_TraceProfilerSetup();
62
63     /* Setup SysTick Timer for 1
64     if (SysTick_Config(CMU_Clock
65         while (1) ;
66 }
67
68 /* Initialize LED driver */
69 BSP_LedsInit();
70 BSP_LedSet(0);
71
72 /* Infinite blink loop */
73 while (1) {
74     //BSP_LedToggle(0);
75     BSP_LedToggle(1);
76     Delay(1000);
77 }
78 }
79
```

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