

Embedded and ambient systems

2023.09.13.

Practice 1



Méréstechnika és
Információs Rendszerek
Tanszék

Preliminary

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

The screenshot shows the course page for 'Embedded and ambient systems'. On the left, a navigation menu is visible with the following items: 'Official page on the faculty Web', 'Final exam', 'Homework', 'Midterm exam', 'Practice' (highlighted with a red box and a red arrow pointing to it), 'Requirements', and 'Textbooks and resources'. The main content area has the title 'Embedded and ambient systems' and buttons for 'View', 'Edit', 'Revisions', 'Track', and 'Translate'. Below this, the course code 'VIMIAC06' is shown, along with the program path: 'B.Sc. Program > BSc in Electrical Engineering (current) > Embedded and control systems specialization - MIT, IIT, AUT (current)'. There is also a link for the 'Official page on the faculty Web'. The 'Course coordinator' section features a profile for György Orosz, an associate professor, with contact information: Szoba: IE330, +36 1 463-3587, and Email: orosz (*) mit * bme * hu. The 'Lecturers' section features a profile for István Tamás Krébesz, an assistant lecturer, with contact information: Szoba: IE413, Tel.: +36 1 463-2673, and 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

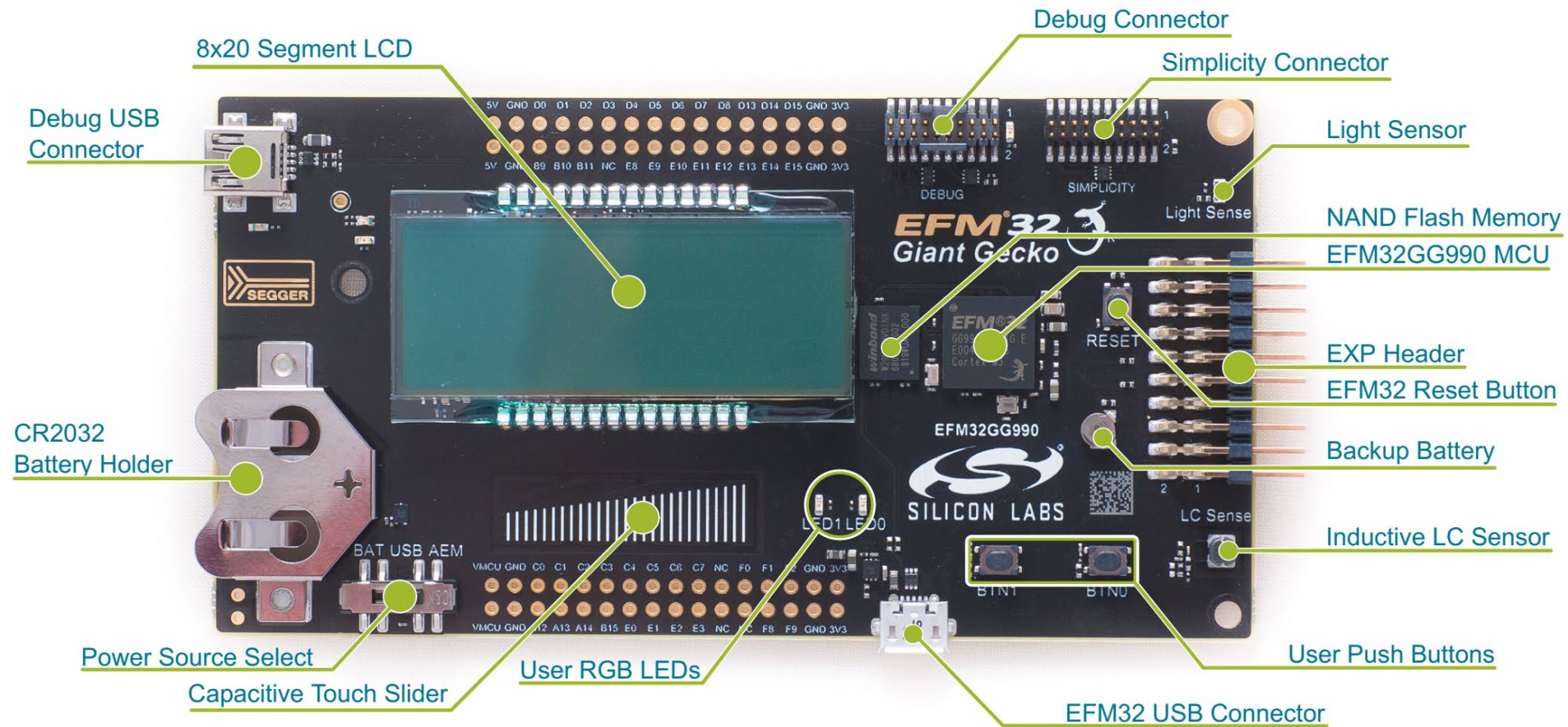
Practice materials will appear during the semester!

References available here:

http://home.mit.bme.hu/~krebesz/bambi_angol/references/

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

1) Development board: EFM32GG-STK3700

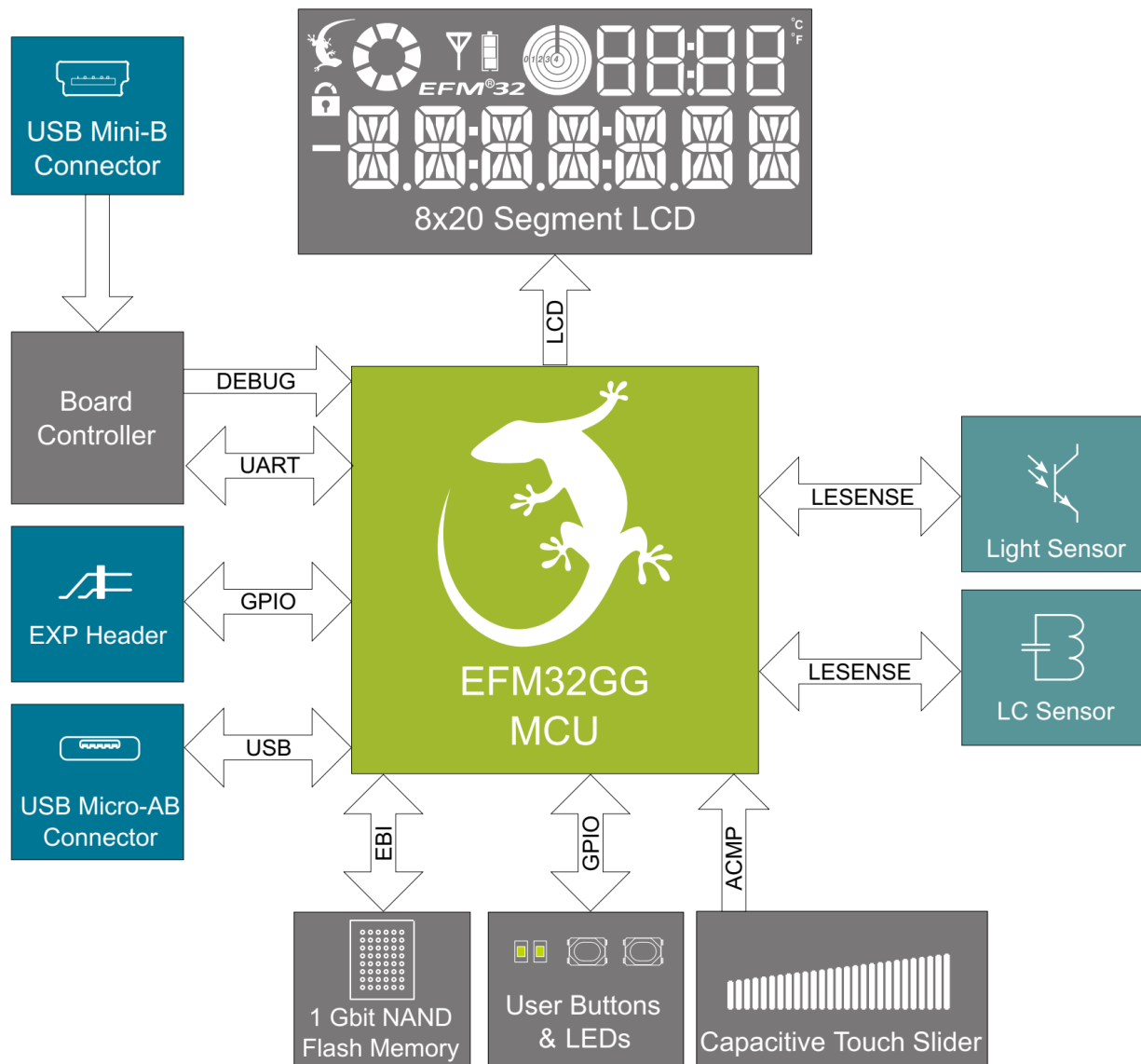


- <https://www.silabs.com/development-tools/mcu/32-bit/efm32gg-starter-kit>

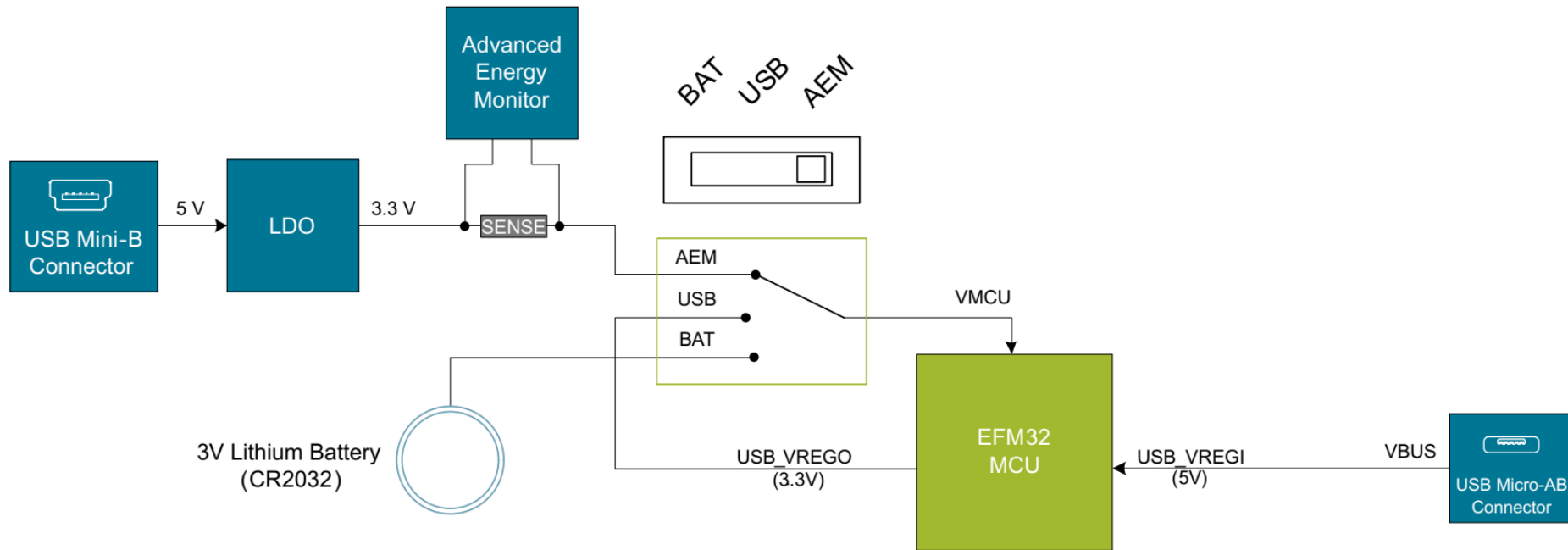
1.1) Main features

- 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

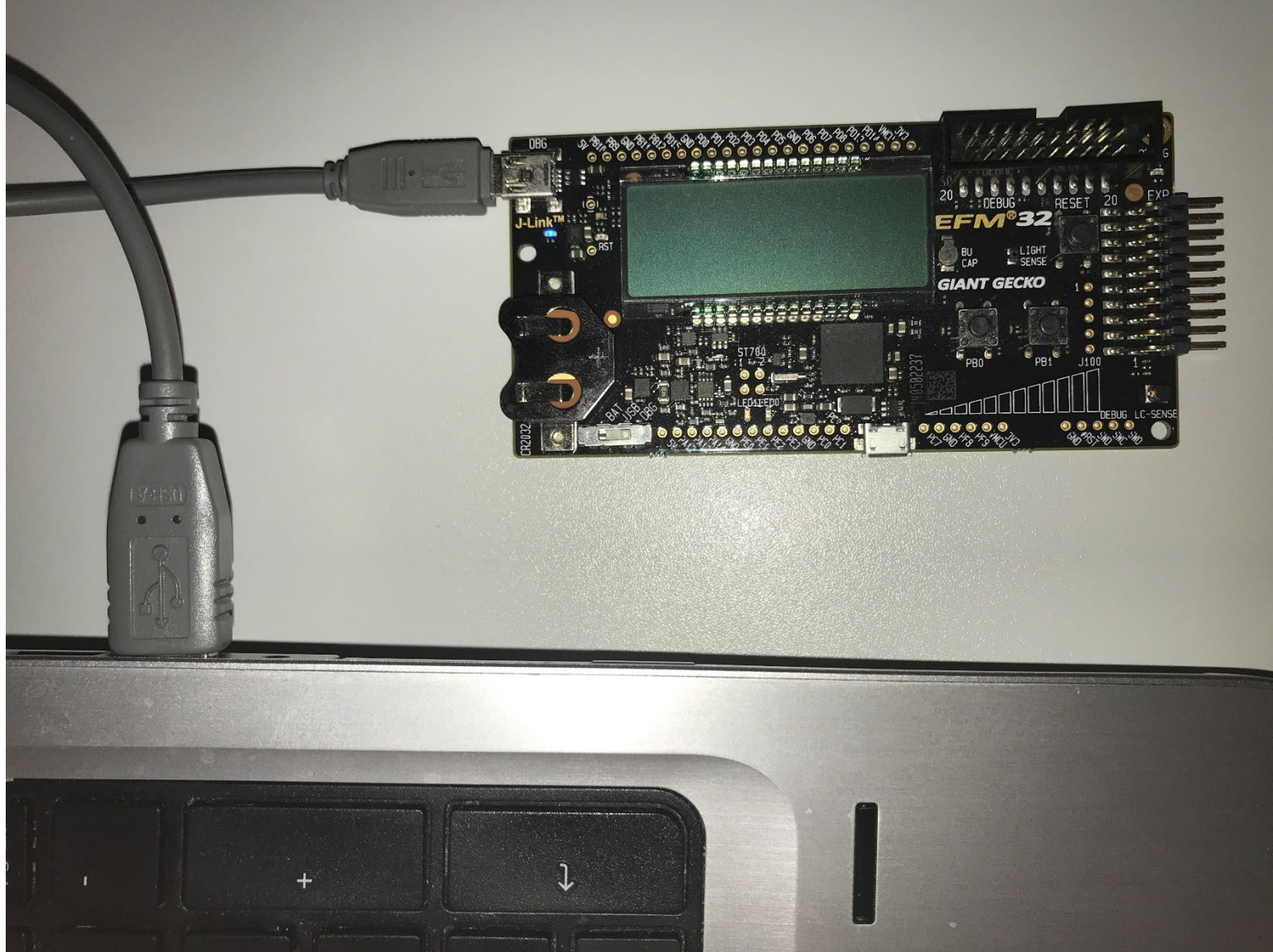


1.3) Power supply

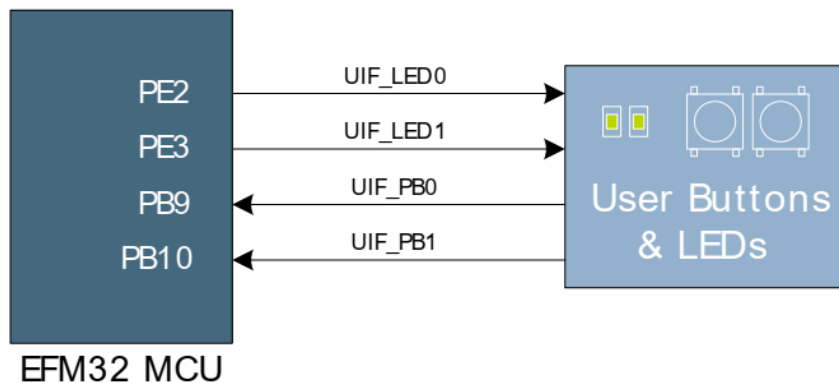


- DBG: via on-board debugger energy monitor can be used (use this)
- BAT: use CR2032 battery
- USB: MCU integrated voltage regulator is used

1.3) Power supply and proper connection

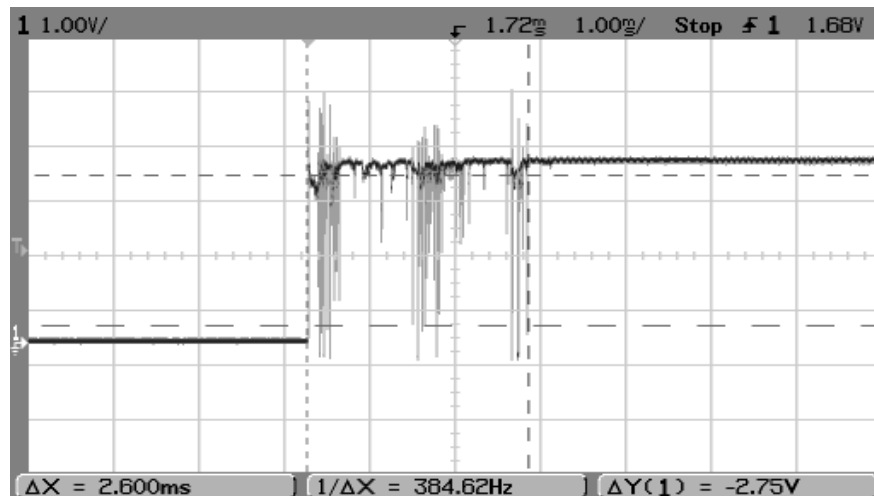
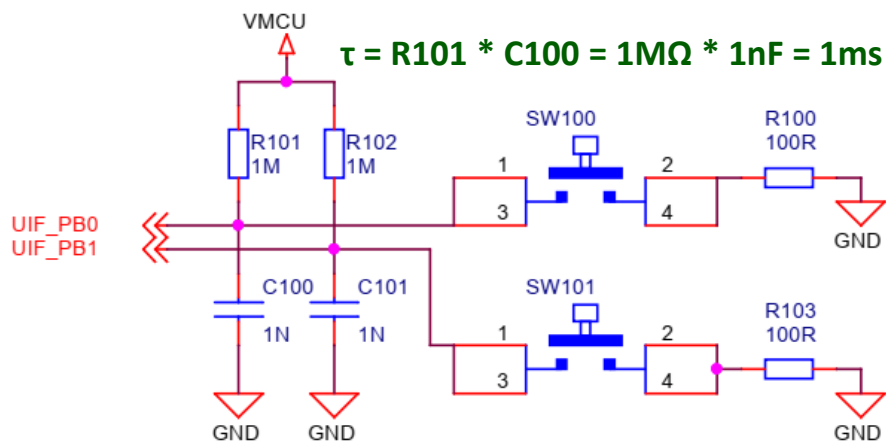


1.4) Peripherals-Buttons/LEDs

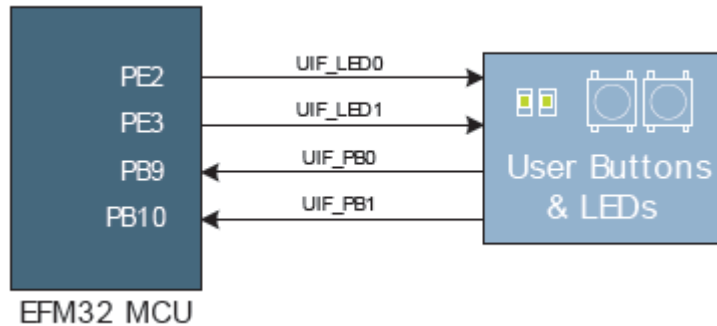


PB0=push button nr. 0
PB9=9th bit of port B
PE3=3rd bit of port E

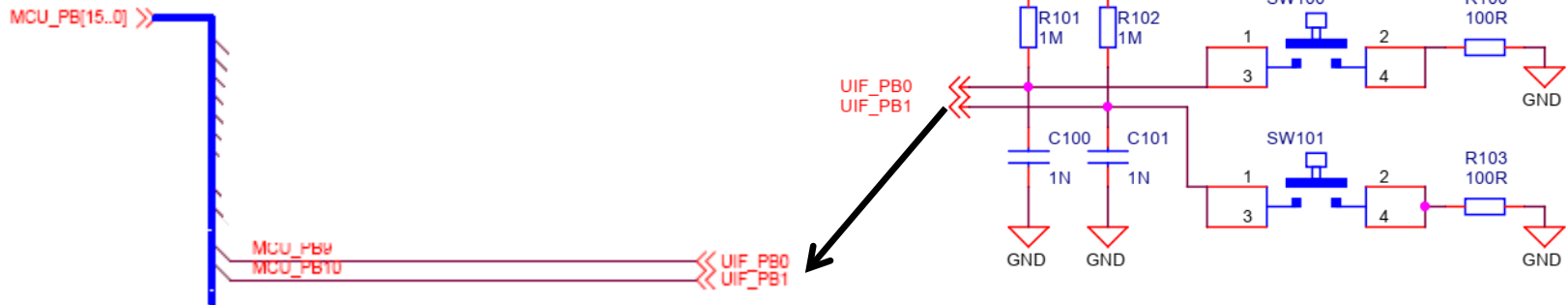
- Push buttons are debounced by RC filter to avoid bouncing:



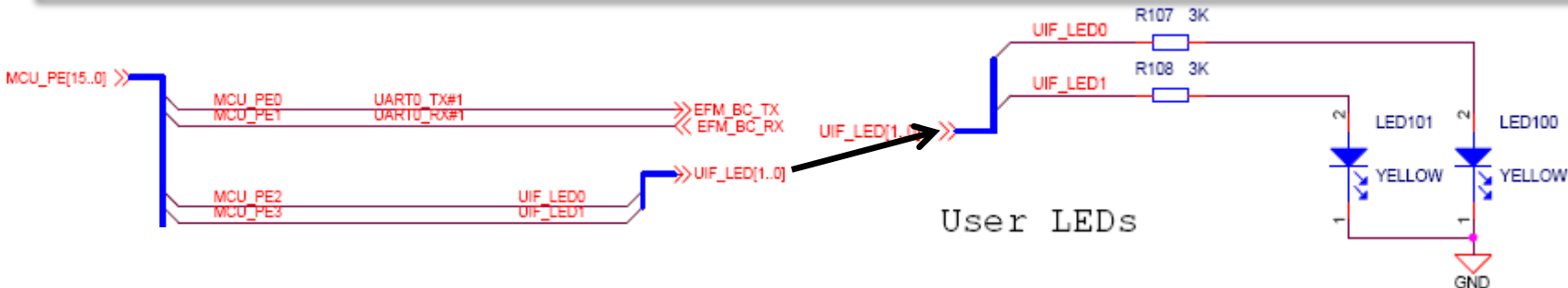
1.4) Peripherals-Buttons/LEDs



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 top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The main window title is "Launcher - STK3700_blink_2/src/blink.c - Simplicity Studio™". The left sidebar shows the "Debug Adapters" tree with a red box around the "EFM32 Giant Gecko Starter Kit board (BRD2200A Rev A03)" entry, labeled "Connected board". The main content area displays "Welcome to Simplicity Studio" with instructions to select a kit or board. Below this are buttons for "New Project" and "Recent Projects". The bottom section features three tabs: "Getting Started", "Documentation", and "Compatible Tools". The "Getting Started" tab is active, showing "Demos", "Software Examples", and "SDK Documentation" sections, each with a "Change Preferred SDK" link. A red box highlights this section, labeled "For your help". The bottom right corner shows the "My Products" section with an input field for "Enter product name".

2.2) Getting started with IDE-Simplicity IDE

View (Simplicity IDE is 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 'EFM32 Giant Gecko Starter Kit board (BRD2200A Rev A03)'.
- Console:** Shows the output of the 'Adapter Pack Console', displaying board information such as 'boardId[0]=2200A', 'boardName[0]=BRD2200A Rev. A03', and 'boardDescription[0]=EFM32 Giant Gecko Starter Kit board'.

```
*****  
* @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  
*****
```

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 is active)

The screenshot shows the SImplicity Studio IDE interface during a debug session. The main window displays the source code of `blink.c` with the `main` function highlighted. The `main` function includes comments and code for initializing the chip, setting up the Energy Profiler, and configuring the SysTick timer. The `main` function is currently active, and the cursor is positioned at the start of the function.

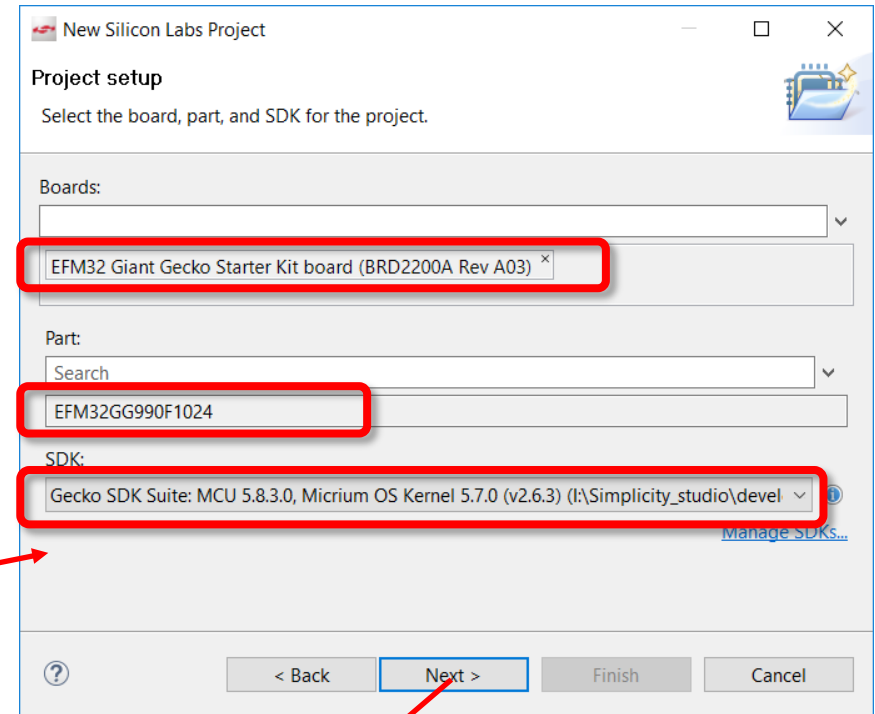
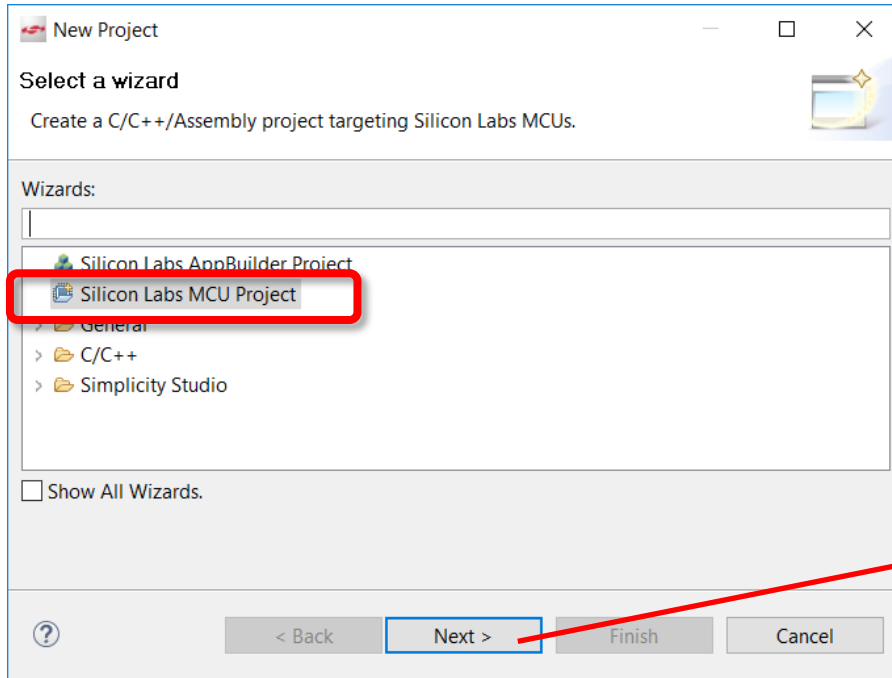
The `Variables` window shows the `msTicks` variable, which is of type `volatile uint32_t` and has a value of `0`. The `Disassembly` window shows the assembly code for the `main` function, starting with `push {r7,lr}` at address `00001368`. The `Console` window is empty, showing the `Program Output Console`.

Name	Type	Value	Location
msTicks	volatile uint32_t	0	0x20000090

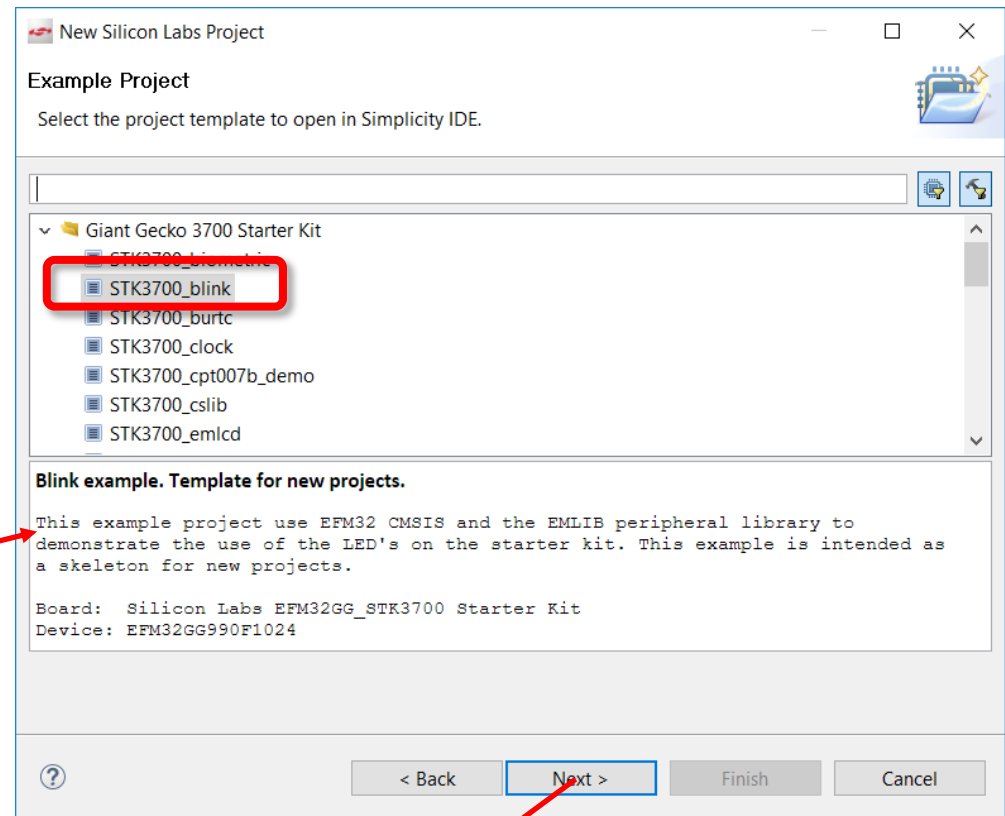
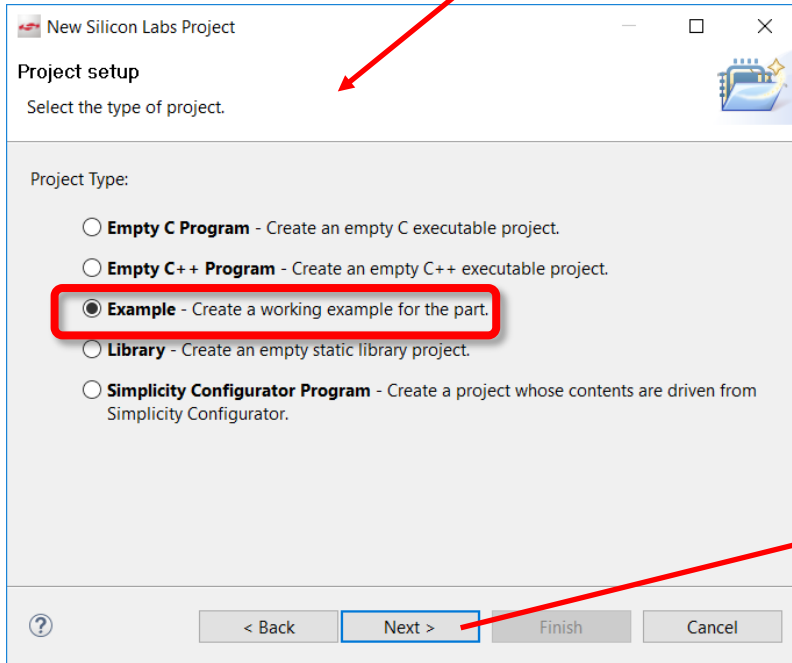
```
main:  
00001368: push {r7,lr}  
0000136a: add r7,sp,#0x0  
58 CHIP_Init();  
0000136c: bl 0x000012d8  
61 BSP_TraceProfilerSetup();  
00001370: bl 0x00000398  
64 if (SysTick_Config(CMU_ClockFreqGet(CMU_ClockCore) / 1000)) {  
00001374: ldr r0,[pc,#0x3c] ; 0x13b0  
00001376: bl 0x00000bc8  
0000137a: 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

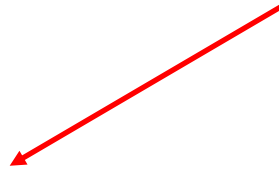
■ File->New->Project:



3) Start a new project



3) Start a new project



New Silicon Labs Project

Project Configuration

Select the project name and location.

Project name: 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

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]', including 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 headers like <stdint.h>, <stdbool.h>, and various project-specific headers. It defines a volatile variable 'msTicks' and a 'Delay' function. The bottom panel shows the Debug Adapters section with 'J-Link Silicon Labs (440019119) [Simplicity Debugger]' and the hardware configuration 'EFM32 Giant Gecko Starter Kit board (BRD2200A Rev A03)'. The Console window is currently empty.

```

/*****
 * @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 */

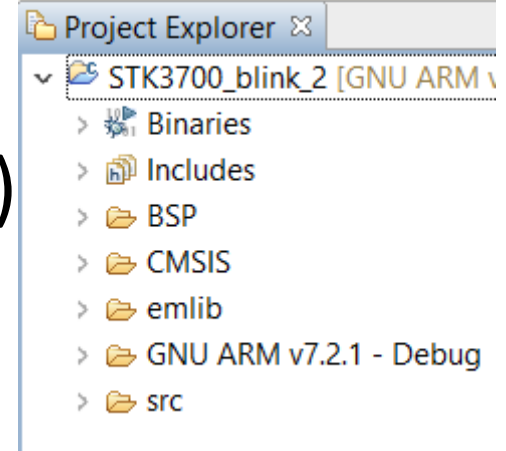
void Delay(uint32_t dlyTicks);

/*****
 * @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 toolbar contains various icons for debugging and development. 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 C 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 for program output.

- 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 menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Project Explorer on the left shows the project structure: Silicon Labs ARM MCU: EFM32GG990F1024 > STK3700_blink_2.axf > main() at blinkc:56 0x1368. The Registers window is open and highlighted with a red circle, displaying a table of 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 main editor shows the source code for blinkc.c, with the SysTick_Handler function visible. The Disassembly window on the right shows the assembly code for the main function, starting with a push instruction for registers r7 and lr.

```
56 {
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_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: lsr  r2,r2,#6
```

- Register content can be manipulated

4.2.2) Expressions

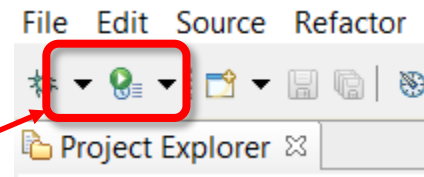
The screenshot displays the Simplicity Studio IDE interface. The top toolbar includes icons for File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The main workspace is divided into several panes:

- Project Explorer:** Shows the project structure for 'Silicon Labs ARM MCU: EFM32GG990F1024', with 'STK3700_blink_2.axf' and 'main() at blink.c:56 0x1368' selected.
- 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'. Below the table, it states 'No details to display for the current selection.' This window is highlighted with a red rounded rectangle.
- Source Code:** Shows the C code for 'blink.c'. The function 'void SysTick_Handler(void)' is visible, with line 35 highlighted. Comments describe the SysTick interrupt service routine and the delay function.
- Disassembly:** Shows the assembly code for the selected location (0x1368). The instructions include 'push {r7,lr}', 'add r7,sp,#0x0', 'CHIP_Init()', 'BSP_TraceProfilerSetup()', and conditional branch instructions.
- Console:** Labeled 'Program Output Console', it 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

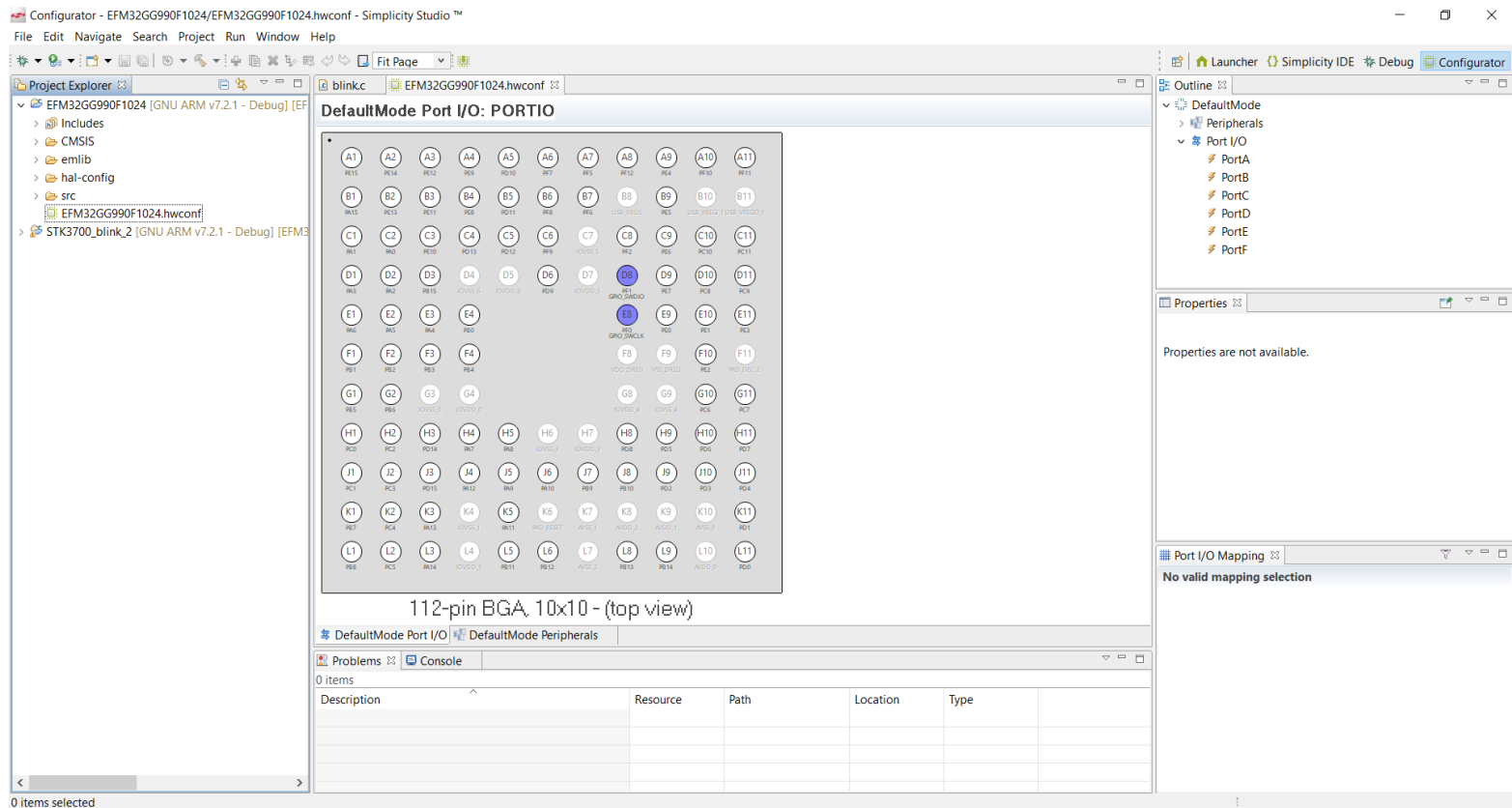


C	Function	Energy	Contribution (%)
JW	0xC8000000-0xC8000FFF	24.70 µWh	33.225%
JW	BSP_TraceProfilerSetup	1.72 µWh	2.311%
JW	0x013A8000-0x013A8FFF	159.12 nWh	0.214%
JW	0x00013000-0x00013FFF	135.76 nWh	0.182%

```
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