



# Embedded and ambient systems



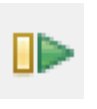

## 2020.10.07.

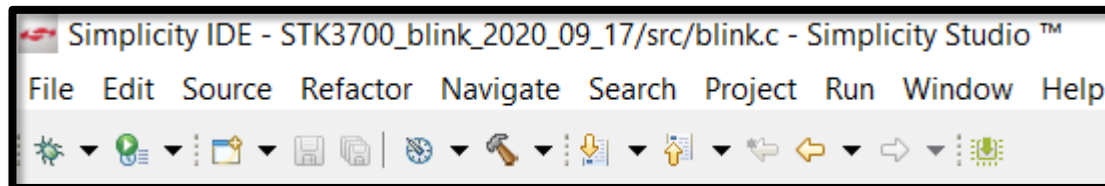
## Practice 2



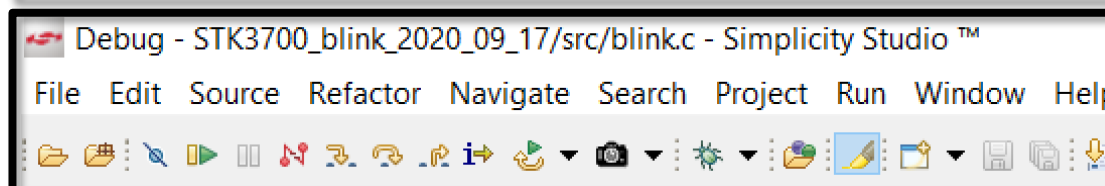
Méréstechnika és  
Információs Rendszerek  
Tanszék

# Example: Blink project

- Review of Blink project at a source code level
- Note icon  : compiles and loads (Debug active)
- Note icon  : only compiles (IDE mode only)
  - Can my code be compiled?
  - Is there any syntactical error?
- Note icon  : starts the downloaded code
- Note icon  : disconnects and switches IDE mode



->IDE mode



->Debug mode

# Example: Blink project

- Written in C programming language
- Entry point is the main function, this function is called and so the program starts here

```
int main(void)
{
    /* Chip errata */
    CHIP_Init(); → HW errors corrected in SW

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

    /* Setup SysTick Timer for 1 msec interrupts */
    if (SysTick_Config(CMU_ClockFreqGet(cmuClock_CORE) / 1000)) {
        while (1); → Initialize SysTick timer peripheral that calls SysTick_Handler interrupt function in every 1ms and increments msTicks variable in every 1ms
    }

    /* Initialize LED driver */
    BSP_LedsInit(); → Initialize the LEDs
    BSP_LedSet(0); → Initialize LED nr.0

    /* Infinite blink loop */
    while (1) { → In the while loop blinking LED algorithm is implemented
        BSP_LedToggle(0); → Change the state of LED nr.0
        BSP_LedToggle(1); → Change the state of LED nr.1
        Delay(1000); → Wait 1000ms
    }
}

void SysTick_Handler(void)
{
    msTicks++; /* increment msTicks variable in every 1ms
}
```

# Delay function

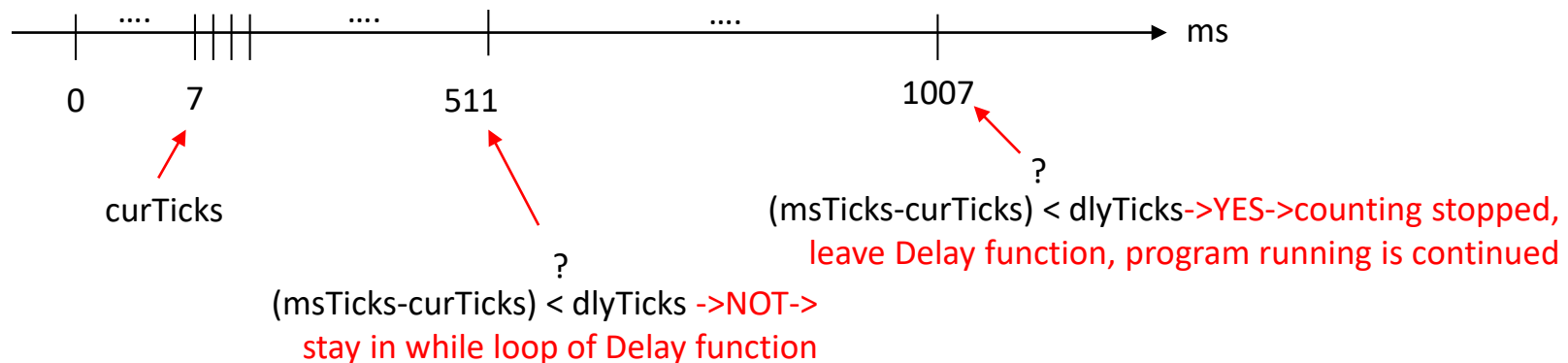
```
void Delay(uint32_t dlyTicks) → One input parameter: how many ticks to wait-> 1tick=1ms in this program
{
    uint32_t curTicks; → Create variable curTicks
    curTicks = msTicks; → Value of msTicks is passed to curTicks
    while ((msTicks - curTicks) < dlyTicks) ; → Wait until condition is met (program sticks in the while loop until condition is met)
}
```

msTicks: the current tick (time, keeps increasing by 1ms)

curTicks: equals msTick value when Delay function was called, constant value during the Delay function runs

dlyTicks: time of delay, now it is 1000ms the time to toggle LEDs

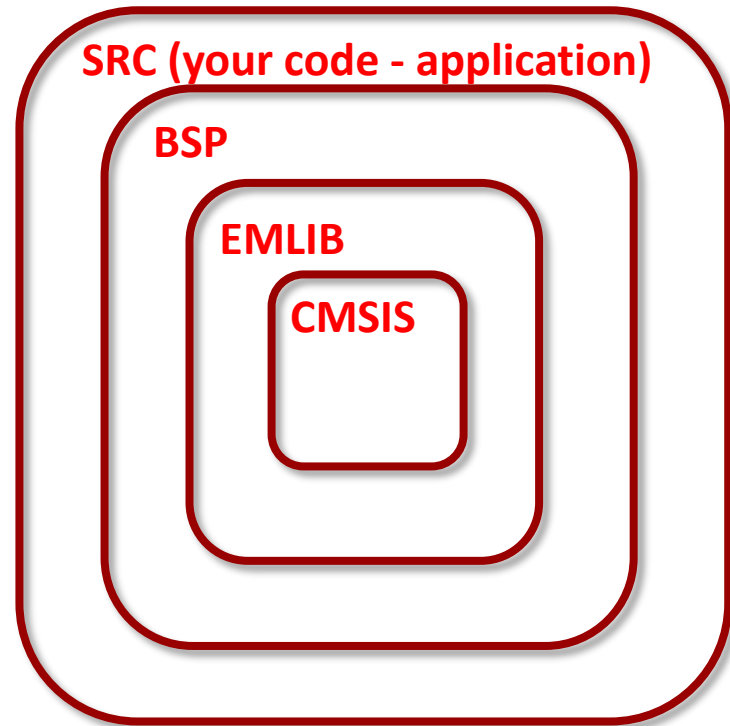
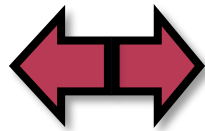
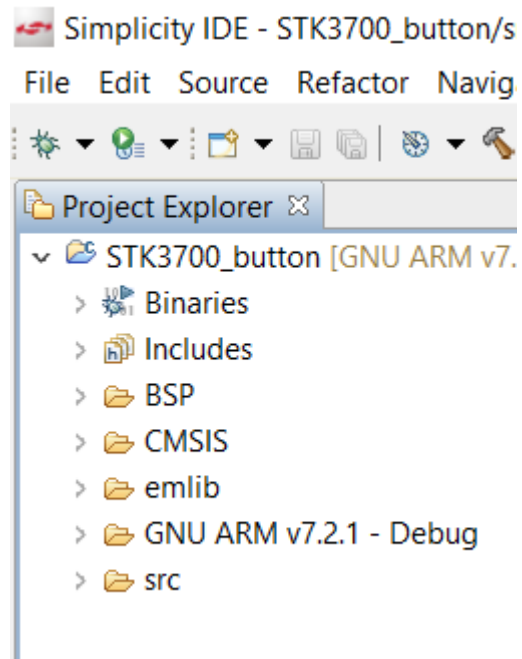
- Operation of Delay function:



- This Delay function is a blocking wait->program cannot run until 1000ms is elapsed

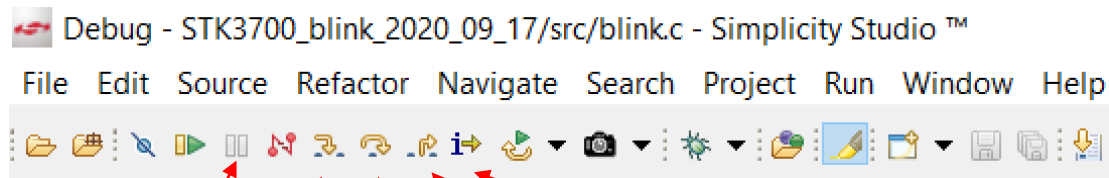
# Hierarchy of functions

- Project Explorer window and the hierarchy of functions gathered in libraries








- All levels can be reached from SRC level directly

# Examination of the Blink program



Suspend Step into Step over Step return Instruction stepping

- When the program is suspended it is most probably in the Delay function since LEDs changes quickly and uC runs the Delay function most of the time

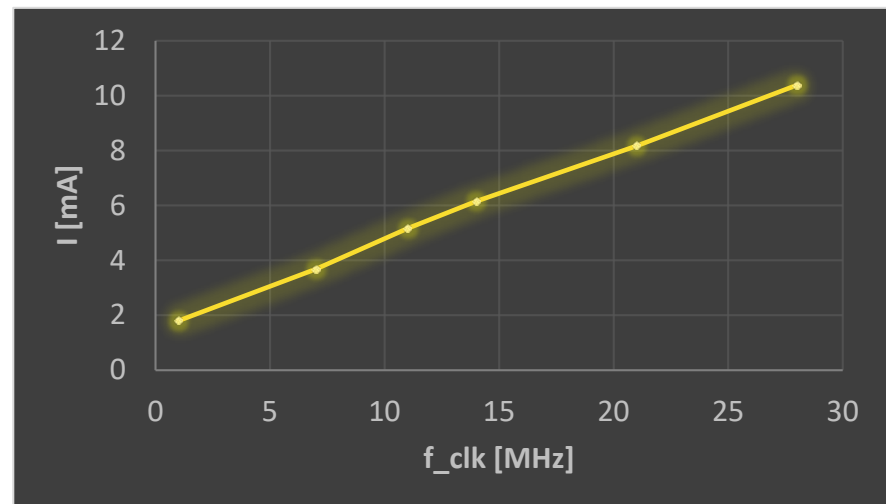
	Suspend	The [ <b>Suspend</b> ] button halts the MCU.
	Step Into	The [ <b>Step Into</b> ] button single steps into the first line of a function.
	Step Over	The [ <b>Step Over</b> ] button single steps over a function, executing the entire function.
	Step Return	The [ <b>Step Return</b> ] button steps out of a function, executing the rest of the function.
	Instruction Step- ping 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.

- Changes in variables and register content can be followed easily in Debug mode

# Current consumption as a function of f\_clk

- Modify the Blink program to measure current consumption as a function of uC clock frequency
- How the clock frequency can be accessed?
  - Insert the appropriate function:
  - `CMU_HFRCOBandSet(cmuHFRCOBand_1MHz);`
- Measure current consumption when clock frequency is set to 1, 7, 11, 14, 21 and 28MHz -> conclusion?

f_clk [MHz]	I [mA]
1	1.8
7	3.68
11	5.16
14	6.15
21	8.17
28	10.38



# Hint for measurement of current consumption as a function of f\_clk

```
int main(void)
{
    /* Chip errata */
    CHIP_Init();

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

    CMU_HFRCOBandSet(cmuHFRCOBand 11MHz);

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

    /* Initialize LED driver */
    BSP_LedsInit();
    BSP_LedSet(0);
    BSP_LedSet(1);

    /* Infinite blink loop */
    while (1) {
        //BSP_LedToggle(0);
        //BSP_LedToggle(1);
        Delay(1000);
    }
}
```



# Buttons

- Certain version of SDK does not contain functions for the buttons in the BSP library, but they are available from the manufacturer (can be downloaded from the course web)
  - `bsp_stk_buttons.c` and `bsp_stk_buttons.h`
  - Copy them into the Includes library (see project explorer window)
    - `i:\Simplicity_studio\developer\sdk\gecko_sdk_suite\v2.6\hardware\kit\common\bsp\`
  - Copy `bsp_stk_buttons.c` file into bsp library
- After opening `bsp_stk_buttons.h` you should use:
  - `BSP_ButtonsInit(void)` -> initialize buttons
  - `BSP_ButtonGet(int btnNo)` -> read button state
  - Do not forget: `#include "bsp_stk_buttons.h"`
- Modify the program to light up LEDx if BTNx is pushed

# Modified code for push buttons

```
/* Initialize LED driver */
BSP_LedsInit();
BSP_LedClear(0);
BSP_LedClear(1);

/* Initialize Buttons */
BSP_ButtonsInit();

/* Infinite button loop */
while (1) {
    if (BSP_ButtonGet(0)) {
        BSP_LedClear(0);
    }else{
        BSP_LedSet(0);
    }
}
```

# The building process (see console)

```
15:28:17 **** Build of configuration GNU ARM v7.2.1 - Debug for project STK3700_button ****
```

```
make -j4 all
```

```
Building file: ../src/button.c
```

```
Invoking: GNU ARM C Compiler
```

```
arm-none-eabi-gcc -g3 -gdwarf-2 -mcpu=cortex-m3 -mthumb -std=c99 '-DDEBUG_EFM=1' '-DEFM32GG990F1024=1' -  
"I:/Simplicity_studio/developer/sdks/gecko_sdk_suite/v2.6//hardware/kit/EFM32GG_STK3700/config" -  
"I:/Simplicity_studio/developer/sdks/gecko_sdk_suite/v2.6//platform/CMSIS/Include" -  
"I:/Simplicity_studio/developer/sdks/gecko_sdk_suite/v2.6//platform/emlib/inc" -  
"I:/Simplicity_studio/developer/sdks/gecko_sdk_suite/v2.6//hardware/kit/common/bsp" -  
"I:/Simplicity_studio/developer/sdks/gecko_sdk_suite/v2.6//platform/Device/SiliconLabs/EFM32GG/Include" -O0 -  
Wall -c -fmessage-length=0 -mno-sched-prolog -fno-builtin -ffunction-sections -fdata-sections -MMD -MP -  
MF"src/button.d" -MT"src/button.o" -o "src/button.o" "../src/button.c"
```

```
Finished building: ../src/button.c
```

```
Building target: STK3700_button.axf
```

```
Invoking: GNU ARM C Linker
```

```
arm-none-eabi-gcc -g3 -gdwarf-2 -mcpu=cortex-m3 -mthumb -T "STK3700_button.ld" -Xlinker --gc-sections -Xlinker -  
Map="STK3700_button.map" --specs=nano.specs -o STK3700_button.axf "./BSP/bsp_bcc.o" "./BSP/bsp_stk.o"  
"./BSP/bsp_stk_buttons.o" "./BSP/bsp_stk_leds.o" "./BSP/bsp_trace.o" "./CMSIS/EFM32GG/startup_gcc_efm32gg.o"  
"./CMSIS/EFM32GG/system_efm32gg.o" "./emlib/em_assert.o" "./emlib/em_cmu.o" "./emlib/em_core.o"  
"./emlib/em_ebi.o" "./emlib/em_emu.o" "./emlib/em_gpio.o" "./emlib/em_system.o" "./emlib/em_usart.o"  
"./src/button.o" -Wl,--start-group -lgcc -lc -lnosys -Wl,--end-group
```

```
Finished building target: STK3700_button.axf
```

# The building process (see console)

Building hex file: STK3700\_button.hex

```
arm-none-eabi-objcopy -O ihex "STK3700_button.axf" "STK3700_button.hex"
```

Building bin file: STK3700\_button.bin

```
arm-none-eabi-objcopy -O binary "STK3700_button.axf" "STK3700_button.bin"
```

Building s37 file: STK3700\_button.s37

```
arm-none-eabi-objcopy -O srec "STK3700_button.axf" "STK3700_button.s37"
```

Running size tool

```
arm-none-eabi-size "STK3700_button.axf" -A
```

```
STK3700_button.axf :
```

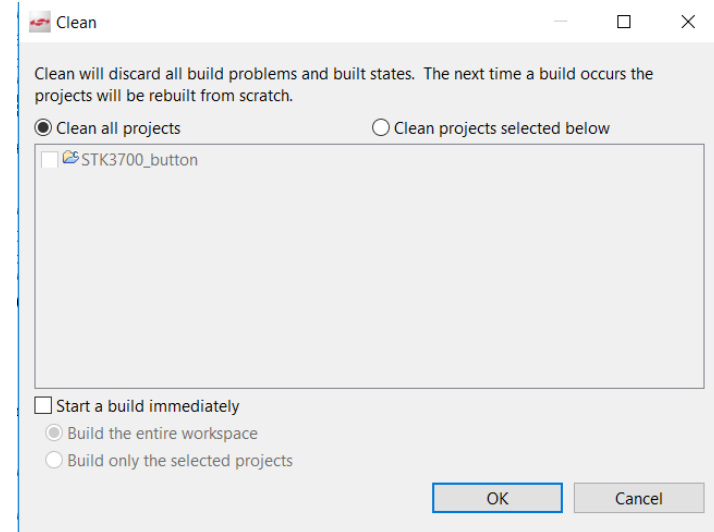
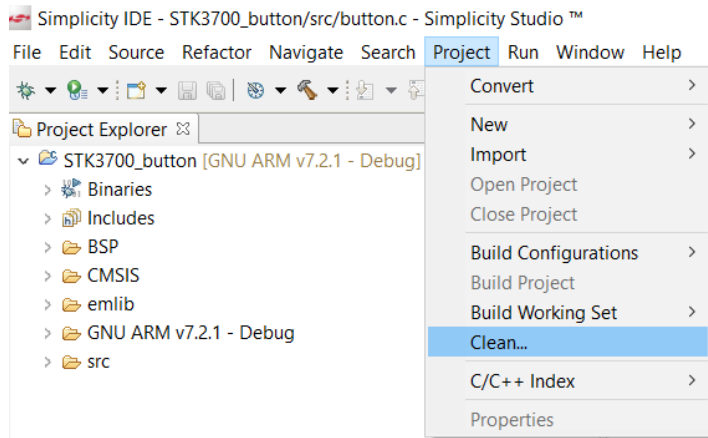
section	size	addr
.text	6612	0
.data	116	536870912
.bss	32	536871028
.heap	3072	536871064
.stack_dummy	1024	536871064
.comment	126	0
.debug_macro	7809	0

..... Some more lines.....

.debug_frame	3764	0
Total	960914	

```
15:28:26 Build Finished (took 8s.456ms)
```

# The clean process (see console)



15:48:05 \*\*\*\* Clean-only build of configuration GNU ARM v7.2.1 - Debug for project STK3700\_button \*\*\*\*

make -j4 clean

```
rm -rf ./src/button.o ./emlib/em_assert.o ./emlib/em_cm_u.o ./emlib/em_core.o ./emlib/em_ebi.o ./emlib/em_emu.o
./emlib/em_gpio.o ./emlib/em_system.o ./emlib/em_usart.o ./CMSIS/EFM32GG/startup_gcc_efm32gg.o
./CMSIS/EFM32GG/system_efm32gg.o ./BSP/bsp_bcc.o ./BSP/bsp_stk.o ./BSP/bsp_stk_buttons.o ./BSP/bsp_stk_leds.o
./BSP/bsp_trace.o ./src/button.d ./emlib/em_assert.d ./emlib/em_cm_u.d ./emlib/em_core.d ./emlib/em_ebi.d
./emlib/em_emu.d ./emlib/em_gpio.d ./emlib/em_system.d ./emlib/em_usart.d
./CMSIS/EFM32GG/system_efm32gg.d ./BSP/bsp_bcc.d ./BSP/bsp_stk.d ./BSP/bsp_stk_buttons.d ./BSP/bsp_stk_leds.d
./BSP/bsp_trace.d STK3700_button.axf
```

15:48:05 Build Finished (took 658ms)