

Dr. Kincses Zoltán, Dr. Vörösházi Zsolt:

FPGA-alapú beágyazott rendszerek tervezése



**A felsőfokú informatikai oktatás
minőségének fejlesztése,
modernizációja**

TÁMOP-4.1.2.A/1-11/1-2011-0104



Főkedvezményezett:
Pannon Egyetem
8200 Veszprém
Egyetem u. 10.

Kedvezményezett:
Szegedi Tudományegyetem
6720 Szeged
Dugonics tér 13.



2014

FPGA-alapú beágyazott rendszerek tervezése

Dr. Vörösházi Zsolt

10. Szoftver alkalmazások fejlesztése, hibakeresése (debug) Xilinx SDK használatával (Software Development Kit) - Timer

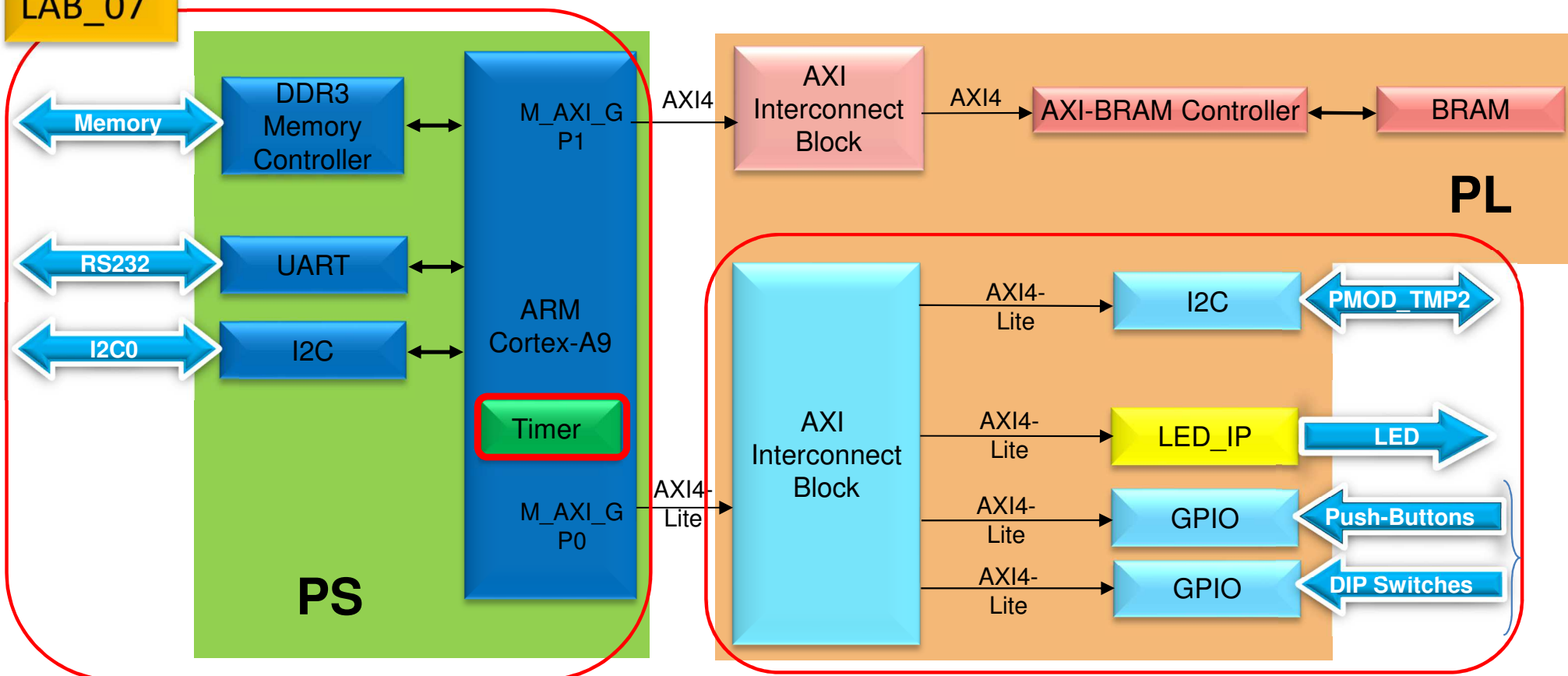


1. Bevezetés – Beágyazott rendszerek.
2. FPGA-k, Digilent ZYBO fejlesztő kártyák és eszközök.
3. Beágyazott Rendszer fejlesztő szoftverkörnyezet (Xilinx Vivado Embedded Development) áttekintése.
4. Beágyazott alap tesztrendszer (BSB - Base System Builder és Board Bring-Up) összeállítása Vivado-ban.
5. Perifériák hozzáadása (IP adatbázisból) az összeállított beágyazott alapszisztemhez.
6. Saját periféria hozzáadása az összeállított beágyazott alapszisztemhez
7. Szoftver alkalmazások fejlesztése, tesztelése, hibakeresése (debug) Xilinx Vivado SDK (Software Development Kit) használatával
8. HW-SW rendszerek együttes tesztelése (Xilinx Vivado ChipScope)
9. Egyedi hardver szellemi termékek fejlesztése és tesztelése (ZYBO VGA vezérlő).
10. Szoftver alkalmazások fejlesztése II. - Timer

- Kezdő-szintű szoftver alkalmazás fejlesztése
 - Kezdő szintű szoftver alkalmazás készítése az SDK-ban található IP periféria (pl. GPIO, I2C, stb.) eléréséhez
 - Linker script létrehozása (.ld)
 - A futtatható részek partícionálása különböző OnChip/External memória területekre
 - Bitstream (.BIT) létrehozása
 - Bitstream (.BIT) letöltése és kipróbálása az ZyBo kártyán
- **Haladó-szintű szoftver alkalmazás fejlesztése**
 - Időzítő (SCU **timer**) használata:
 - SCU = Snoop Control Unit / „private” Timer (ARM időzítője)
 - SDK Debugger használata

A bővíthető tesztrendszer

LAB_07



PS oldal:

- ARM hard-procессzor mag
- belső OnChip-RAM vezérlő
- RS232 soros interfész
- külső DDR3 memória vezérlő
- I2C vezérlő
- **SCU Timer (ARM)**

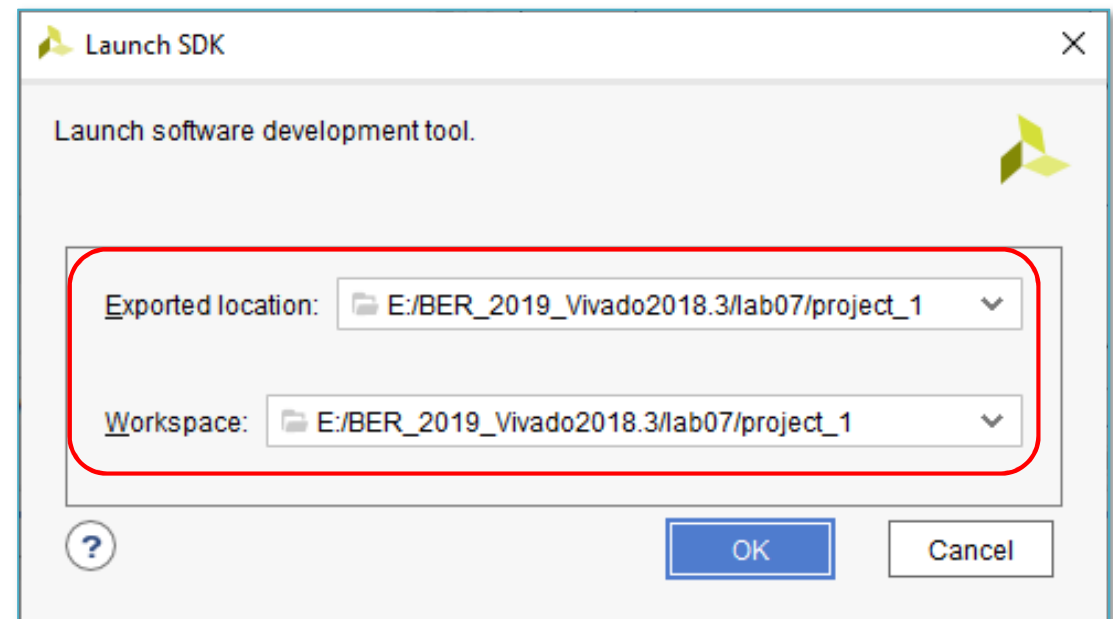
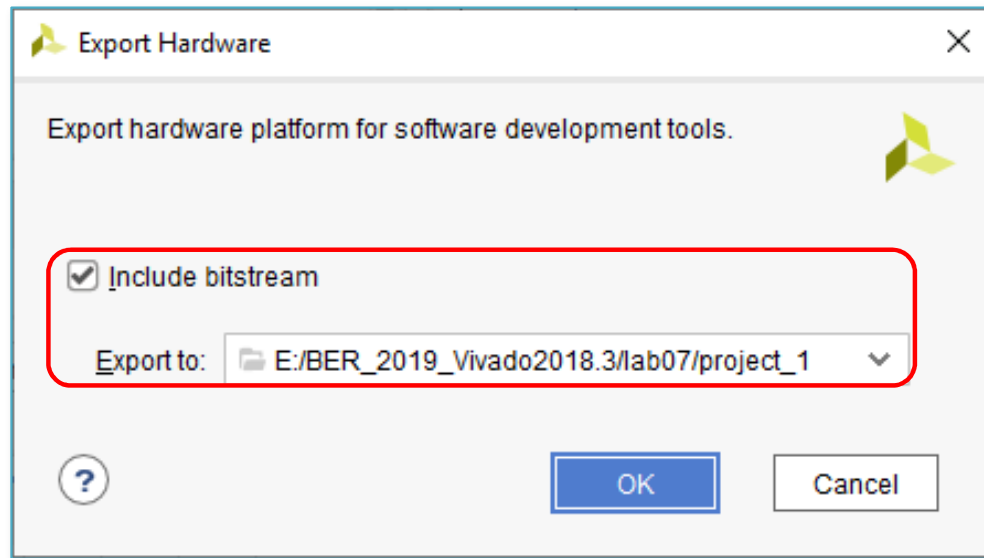
PL oldal:

- **A: GPIO**
 - **PBSs: Push Button** (nyomógomb kezelő)
 - **DIPs: Switches** (kapcsoló kezelő)
- **B: GPIO**
 - **LEDs: LED** kijelző
- **C: I2C** vezérlős hőmérsékletmérő modul (PMOD_TMP2)
 - ARM PS/ vagy PL oldalra is tehető

- Hozzunk létre egy új mappát, legyen a neve `\LAB07`
 - Archiváljuk és másoljuk át az előző ismeretkör (`\LAB02_B`) elsajátításakor létrehozott projektet
 - Project -> Save as ...
- Indítsuk el az Vivado környezetet
- Mivel a CPU (ARM) belső/private Timer-ét használjuk, nem kell újrafordítani a FW-t!
- SDK indítása...

- 1.) SDK elindítása (EDK-ból): **Project → Export Hardware Design to SDK**
- 2.) Kattintsunk az **Export & Launch SDK** gombra
 - Ha még nincs elkészítve az új rendszerhez tartozó netlista és bitstream, akkor ezt le kell generálni mielőtt az SDK elindulna
- 3.) A **Select a workspace** ablakban válasszuk ki a `<project_dir>/LAB07/SDK/SDK_Export` → majd kattintsunk az **OK**-ra
- 4.) Az SDK-ban: **File → New → Xilinx Board Support Package** → kattintsunk a **Finish**-re.
 - Új BSP neve `SCUTimer_bsp_0` lehet

Export&Launch SDK



SDK - system.hdf ellenőrzése



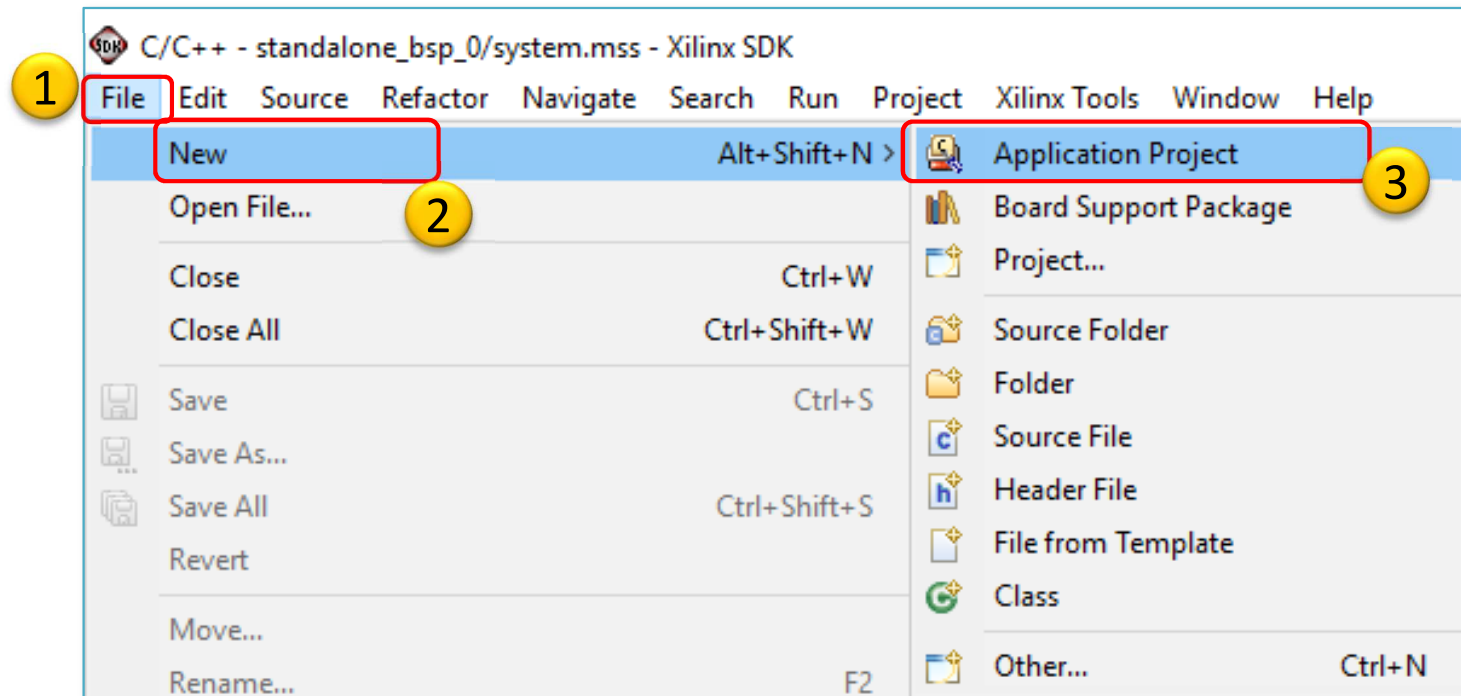
SCU_Timer: PS oldali címtartományának és driverének (verzió szám) ellenőrzése

Address Map for processor ps7_cortexa9_[0-1]

Cell	Base Addr	High Addr	Slave I/f	Mem/Reg
ps7_intc_dist_0	0xf8f01000	0xf8f01fff		REGISTER
ps7_scutimer_0	0xf8f00600	0xf8f0061f		REGISTER
ps7_slcr_0	0xf8000000	0xf8000fff		REGISTER
ps7_scuwdt_0	0xf8f00620	0xf8f006ff		REGISTER
ps7_l2cachec_0	0xf8f02000	0xf8f02fff		REGISTER
ps7_scuc_0	0xf8f00000	0xf8f000fc		REGISTER

system.hdf			
IP blocks present in the design			
ps7_0_axi_periph	axi_interconnect	2.1	
ps7_intc_dist_0	ps7_intc_dist	1.00.a	
leds_4bit	axi_gpio	2.0	Registers
rst_ps7_0_100M	proc_sys_reset	5.0	
ps7_scutimer_0	ps7_scutimer	1.00.a	
ps7_slcr_0	ps7_slcr	1.00.a	
dip	axi_gpio	2.0	Registers
ps7_scuwdt_0	ps7_scuwdt	1.00.a	
ps7_l2cachec_0	ps7_l2cachec	1.00.a	

- Új projekt alkalmazás létrehozása (Application Project)



Tesztalkalmazás (SCUTimer) készítése III.



SDK New Project

Application Project
Create a managed make application project.

1

Project name: SCUTimer

☒ Use default location
Location: E:\BER_2019_Vivado2018.3\lab07\project_1\SCUTimer Browse...

Choose file system: default

OS Platform: standalone

Target Hardware
Hardware Platform: system_wrapper_hw_platform_0 New...
Processor: ps7_cortexa9_0

Target Software
Language: ☒ C ☐ C++
Compiler: 32-bit
Hypervisor Guest: 2 N/A
Board Support Package: ☒ Create New SCUTimer_bsp
☐ Use existing

SCUTimer alkalmazás új BSP-hez rendelése (SCUTimer_bsp)

< Back **Next >** Finish Cancel

Projekt neve: *SCUTimer*

SDK New Project

Templates
Create one of the available templates to generate a fully-functioning application project.

Available Templates:

- Dhrystone
- Empty Application** 3
- Hello World
- lwIP Echo Server
- lwIP TCP Perf Client
- lwIP TCP Perf Server
- lwIP UDP Perf Client
- lwIP UDP Perf Server
- Memory Tests
- OpenAMP echo-test
- OpenAMP matrix multiplication Demo
- OpenAMP RPC Demo
- Peripheral Tests
- RSA Authentication App
- Zynq DRAM tests
- Zynq FSBL

A blank C project.

? < Back Next > **Finish** Cancel

Tesztalkalmazás (TestApp) készítése V.



- C forrás fájl hozzáadása (lab07.c):

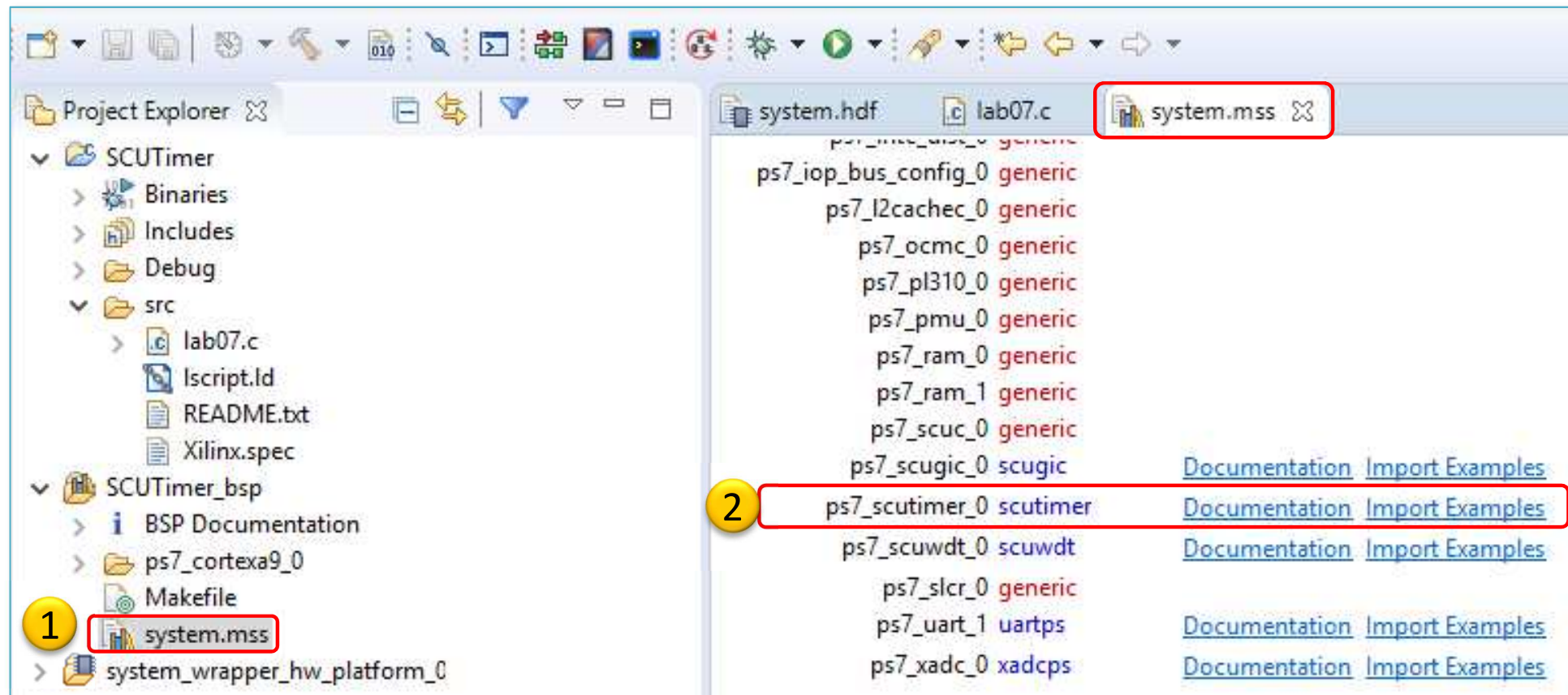
The screenshot illustrates the process of adding a C source file to a project in the Xilinx IDE. The steps are numbered 1 through 4:

1. In the Project Explorer, right-click on the **SCUTimer** project.
2. Select **Import...** from the context menu.
3. In the **Import** dialog, select **General** and then **Archive File**.
4. In the **Archive file** dialog, select the file **lab07.c** from the **From archive file:** list, and click **Next >**.

Below the screenshots, a yellow box contains the following text:

SCUTimer -> Src -> Import... -> General -> Archive file

- SCUTimer_bsp -> **system.mss** vizsgálata



Driver API: dokumentáció (html), és példaprogramok elérhetőek (**polling** vs. interrupt változatban is)

SCU Timer API driver dokumentáció



The screenshot shows a web browser window displaying the Xilinx SCU Timer API driver documentation. The browser's address bar shows the file path: `file:///C:/Xilinx_2018.3/SDK/2018.3/data/embeddedsw/XilinxProcessorIPLib/drivers/scutimer_v2_1/doc/html/ap`. The page header features the Xilinx logo and the title "scutimer_v2_1 Xilinx SDK Drivers API Documentation". A navigation bar at the top includes tabs for "Overview", "Data Structures", "APIs", "File List", and "Examples". On the left side, a sidebar menu is visible, with a red rectangle highlighting the "scutimer_v2_1" section and its sub-items: "Data Structures", "APIs", "File List", and "Examples". The main content area is titled "scutimer_v2_1 Documentation" and contains the following text:

The timer driver supports the Cortex A9 private timer. The timer driver supports the following features:

- Normal mode and Auto reload mode
- Interrupts (Interrupt handler is not provided in this driver. Application has to register it's own handler)

Initialization and Configuration

The device driver enables higher layer software (e.g., an application) to communicate with the Timer.

XScuTimer_CfgInitialize() API is used to initialize the Timer. The user needs to first call the **XScuTimer_LookupConfig()** parameter to the **XScuTimer_CfgInitialize()** API.

SCU Timer API példaprogramok



File Edit View History Bookmarks Tools Help

scutimer_v2_1: Examples

file:///C:/Xilinx_2018.3/SDK/2018.3/data/embeddedsw/XilinxProcessorIPLib/drivers/scutimer_v2_1/doc/html/api/exam

XILINX **scutimer_v2_1**
Xilinx SDK Drivers API Documentation

Overview Data Structures APIs File List Examples

▼ scutimer_v2_1
▶ Data Structures
▶ APIs
▶ File List
▶ Examples

Examples

You can refer to the below stated example applications for more details on how to use scutimer driver.

xscutimer_intr_example.c

Contains an example on how to use the XScutimer driver directly. This example contains the Cortex A9 Scu Private Timer and the driver (**XScuTimer**) using interrupts.
For details, see [xscutimer_intr_example.c](#).

xscutimer_polled_example.c

Contains an example on how to use the XScutimer driver directly. This example shows the usage of the Scu Private Timer driver (**XScuTimer**) and hardware timer device. Reload mode is not enabled.
For details, see [xscutimer_polled_example.c](#).

SCU Timer (**xscutimer.h** referencia)



Data Structures

```
struct XScuTimer_Config
struct XScuTimer
```

Defines

```
#define XScuTimer_IsExpired(InstancePtr)
#define XScuTimer_RestartTimer(InstancePtr)
#define XScuTimer_LoadTimer(InstancePtr, Value)
#define XScuTimer_GetCounterValue(InstancePtr)
#define XScuTimer_EnableAutoReload(InstancePtr)
#define XScuTimer_DisableAutoReload(InstancePtr)
#define XScuTimer_EnableInterrupt(InstancePtr)
#define XScuTimer_DisableInterrupt(InstancePtr)
#define XScuTimer_GetInterruptStatus(InstancePtr)
#define XScuTimer_ClearInterruptStatus(InstancePtr)
```

Functions

```
XScuTimer_Config * XScuTimer_LookupConfig (u16 DeviceId)
int XScuTimer_SelfTest (XScuTimer *InstancePtr)
int XScuTimer_CfgInitialize (XScuTimer *InstancePtr, XScuTimer_Config *ConfigPtr, u32 EffectiveAddress)
void XScuTimer_Start (XScuTimer *InstancePtr)
void XScuTimer_Stop (XScuTimer *InstancePtr)
void XScuTimer_SetPrescaler (XScuTimer *InstancePtr, u8 PrescalerValue)
u8 XScuTimer_GetPrescaler (XScuTimer *InstancePtr)
```

SCU Timer alkalmazás: fordítási hibák javítása



The screenshot shows the Xilinx SDK IDE with the SCUTimer project open. The main editor displays the lab07.c file, which contains the following code:

```
#include "xparameters.h"
#include "xgpio.h"
// Include scutimer header file

//=====
XScuTimer Timer; /* Cortex A9 SCU Private Timer Instance */

#define ONE_SECOND <> // be a half of the CPU clock speed/10

int main (void)
{
    XGpio dip, push, led;
    int psb_check, dip_check, dip_check_prev, count, Status;
}
```

The console window shows the build output with the following error messages:

```
CDT Build Console [SCUTimer]
02:32:35 **** Auto Build of configuration Debug for project SCUTimer ****
make pre-build main-build
a9-linaro-pre-build-step
'Building file: ../src/lab07.c'
'Invoking: ARM v7 gcc compiler'
arm-none-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0 -MT"src/lab07.o" -mcpu=cortex-a9 -mfpv=vfpv3 -mfloat-abi=hard -I../SCUTimer_bsp/ps7_c
../src/lab07.c:6:1: error: unknown type name 'XScuTimer'
XScuTimer Timer; /* Cortex A9 SCU Private Timer Instance */
^~~~~~
../src/lab07.c: In function 'main':
../src/lab07.c:17:4: error: unknown type name 'XScuTimer_Config'; did you mean 'XGpio_Config'?
XScuTimer_Config *ConfigPtr;
^~~~~~
XGpio_Config
../src/lab07.c:18:4: error: unknown type name 'XScuTimer'
XScuTimer *TimerInstancePtr = &Timer;
^~~~~~
../src/lab07.c:22:27: error: 'XPAR_SWITCHES_DEVICE_ID' undeclared (first use in this function); did you mean 'XPAR_SCUTIMER_DEVICE_ID'?
XGpio Initialize(&dip, XPAR SWITCHES_DEVICE_ID);
^~~~~~
```

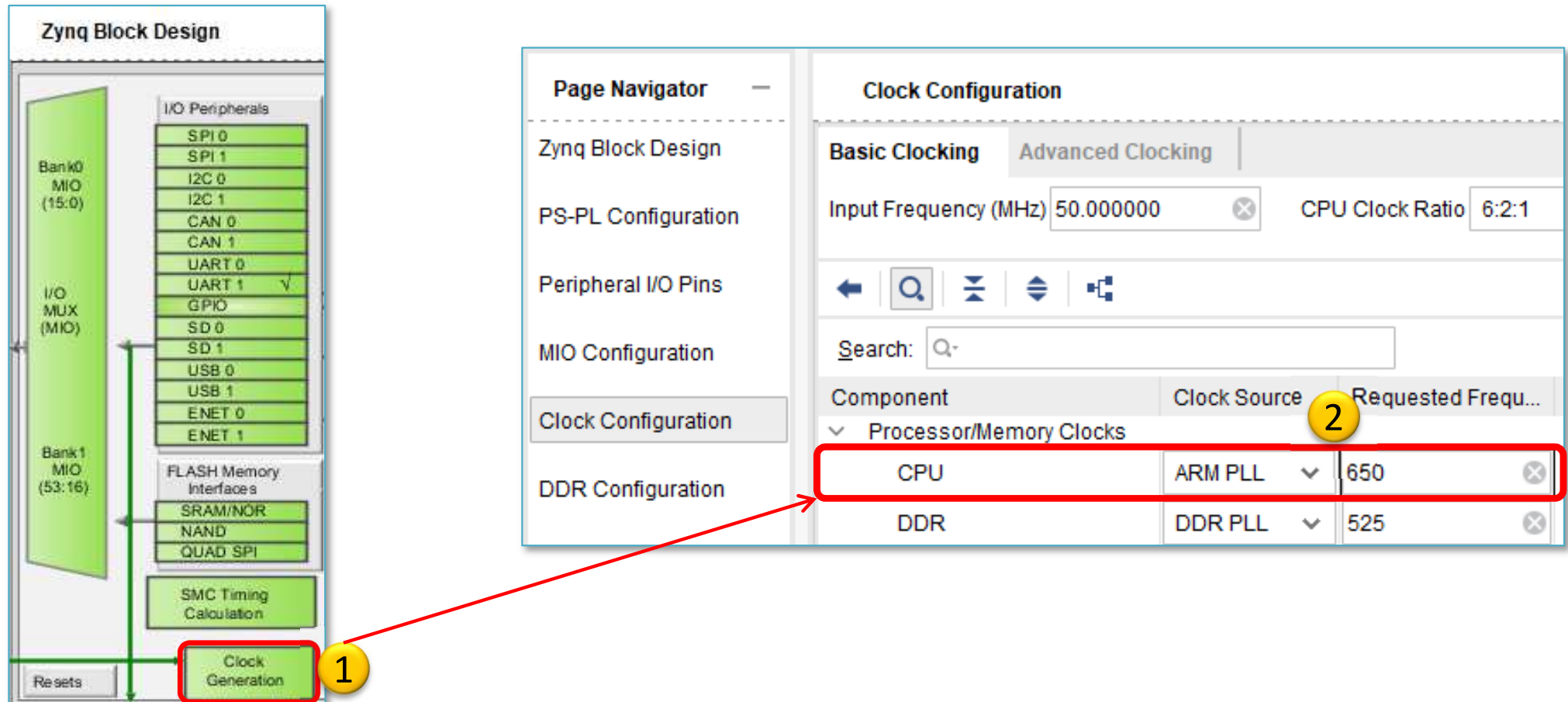
The Project Explorer on the left shows the project structure, including the src directory and the lab07.c file. The Target Connections window at the bottom left shows the hardware server, Linux TCF Agent, and QEMU TcfGdbClient.

SCU Timer alkalmazás: ellenőrzés és bővítés I.

```
XGpio_Initialize(&push, XPAR_PUSH_DEVICE_ID);  
XGpio_SetDataDirection(&push, 1, 0xffffffff);
```

```
#define XPAR_PUSH_DEVICE_ID XPAR_PB_DEVICE_ID
```

Vivado:



The image shows two screenshots from the Vivado IDE. The left screenshot, titled 'Zynq Block Design', displays a block diagram of a Zynq device. It includes I/O Peripherals (Bank0 MIO (15:0), I/O MUX (MIO), Bank1 MIO (53:16)), I/O Peripherals (SPI 0, SPI 1, I2C 0, I2C 1, CAN 0, CAN 1, UART 0, UART 1, GPIO, SD 0, SD 1, USB 0, USB 1, ENET 0, ENET 1), FLASH Memory Interfaces (SRAM/NOR, NAND, QUAD SPI), SMC Timing Calculation, and a 'Clock Generation' block highlighted with a red box and a yellow circle labeled '1'. The right screenshot, titled 'Clock Configuration', shows the 'Basic Clocking' tab. It displays 'Input Frequency (MHz)' as 50.000000 and 'CPU Clock Ratio' as 6:2:1. Below this, a table lists the clock sources for various components. The 'CPU' row is highlighted with a red box and a yellow circle labeled '2'.

Component	Clock Source	Requested Frequency (MHz)
CPU	ARM PLL	650
DDR	DDR PLL	525

```
#define ONE_SECOND <> // be half of the CPU clock speed
```

```
#define ONE_SECOND 32500000 // be half of the CPU clock speed - 650 MHz
```

Fordítási hibák: Xparameters.h



1.) Define-ok hibáinak javítása (átdefiniálás):

```
../src/lab5.c: In function 'main':  
../src/lab5.c:33:28: error: 'XPAR_PUSH_DEVICE_ID' undeclared (first use in this function)  
../src/lab5.c:33:28: note: each undeclared identifier is reported only once for each function it appears in  
../src/lab5.c:36:27: error: 'XPAR_LEDS_DEVICE_ID' undeclared (first use in this function)  
make: *** [src/lab5.o] Error 1
```

Hozzáadás, vagy helyettesítés (22., 25. és 28. sorokban):

```
#define XPAR_PUSH_DEVICE_ID    XPAR_PB_DEVICE_ID  
#define XPAR_DIPSWITCH_DEVICE_ID  XPAR_DIP_DEVICE_ID  
  
#define XPAR_LEDS_DEVICE_ID    XPAR_LEDS_4BITS_DEVICE_ID
```

```
//=====
XScuTimer Timer; /* Cortex A9 SCU Private Timer Instance */
```



2.) Header hozzáadása (4. sor):

```
// Include scutimer header file
#include "xscutimer.h"
```

3.) Timer inicializálásának hozzáadása, status lekérdezéssel (33-39. sorok):

```
// Initialize the timer
ConfigPtr = XScuTimer_LookupConfig(XPAR_PS7_SCUTIMER_0_DEVICE_ID);
Status = XScuTimer_CfgInitialize(TimerInstancePtr, ConfigPtr, ConfigPtr->BaseAddr);
if (Status != XST_SUCCESS) {
    return XST_FAILURE;
}
```

4.) Timer beállításai, és indítása (42. - 47. sorok):

```
// Load timer with delay in multiple of ONE_SECOND
XScuTimer_LoadTimer(TimerInstancePtr, ONE_SECOND*dip_check_prev);

// Set AutoLoad mode
XScuTimer_EnableAutoReload(TimerInstancePtr);

// Start the timer
XScuTimer_Start(TimerInstancePtr);
```

5.) Nyomógomb ellenőrzése – legbaloldalibb megnyomása esetén lépjen ki (52. sor) – folyamatos lekérdezés while(1)-on belül:

```
// Read push buttons and break the loop if leftmost button pressed
psb_check = XGpio_DiscreteRead(&push, 1);
if(psb_check >= 0x8)
```

6.) Timer betöltése a dip kapcsoló új értékével (63. sor) – folyamatos lekérdezés while(1)-on belül:

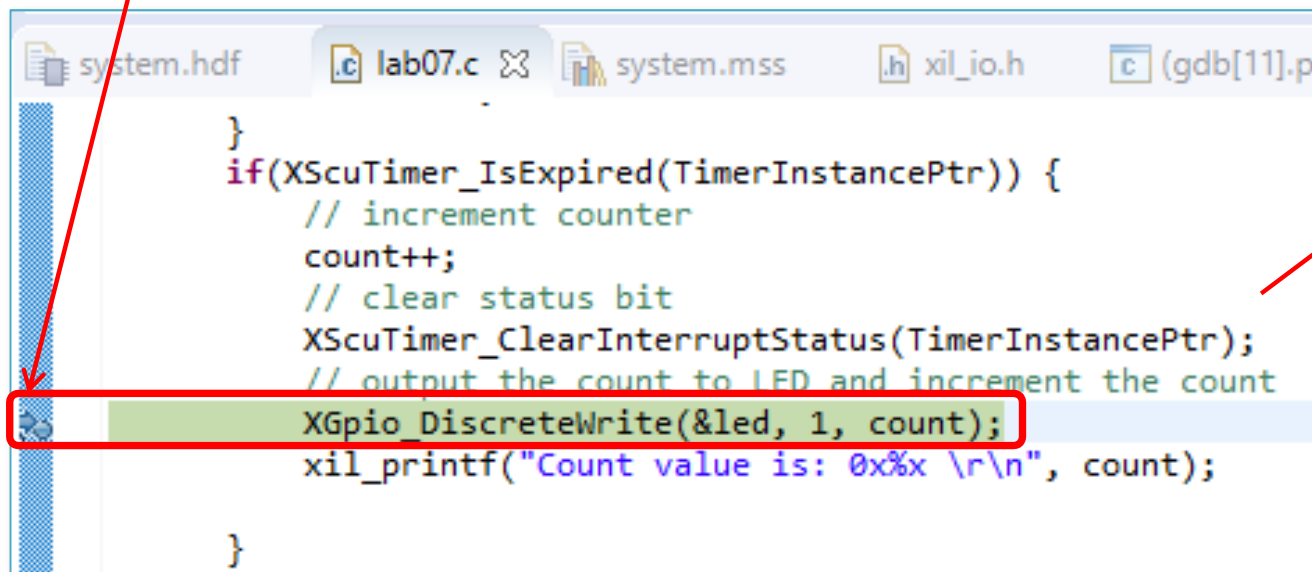
```
// load timer with the new switch settings
XScuTimer_LoadTimer(TimerInstancePtr, ONE_SECOND*dip_check);
```

7.) Timer státusz bitjének törlése, és counter értékének LED-re íratása (67. - 73. sorok):

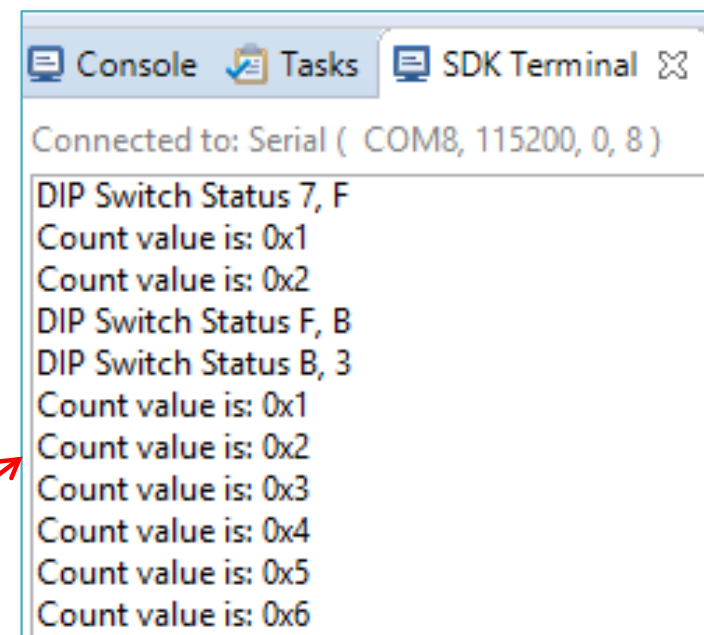
```
if(XScuTimer_IsExpired(TimerInstancePtr)) {
    // increment counter
    count++;
    // clear status bit
    XScuTimer_ClearInterruptStatus(TimerInstancePtr);
    // output the count to LED and increment the count
    XGpio_DiscreteWrite(&led, 1, count);
    xil_printf("Count value is: 0x%x \r\n", count);
}
```


A program tesztelése a fejlesztő kártyán I.

- Debugger elindítása
 - **Project Explorer** → Jobb kattintás a `\SCUtimer` projektre → **Debug As** → **Launch on Hardware** → **Yes** (Debug perspektíva engedélyezése)
 - A `SCUtimer.elf` letöltődik
- Jobb kattintás a **Variables** fül-re → **Add Global Variables** → **count** változó kiválasztása → **OK**
- **BreakPoint** elhelyezése
 - Dupla kattintás a kiválasztott soron



```
system.hdf lab07.c system.mss xil_io.h (gdb[11].p
}
if(XScuTimer_IsExpired(TimerInstancePtr)) {
    // increment counter
    count++;
    // clear status bit
    XScuTimer_ClearInterruptStatus(TimerInstancePtr);
    // output the count to LED and increment the count
    XGpio_DiscreteWrite(&led, 1, count);
    xil_printf("Count value is: 0x%x \r\n", count);
}
```



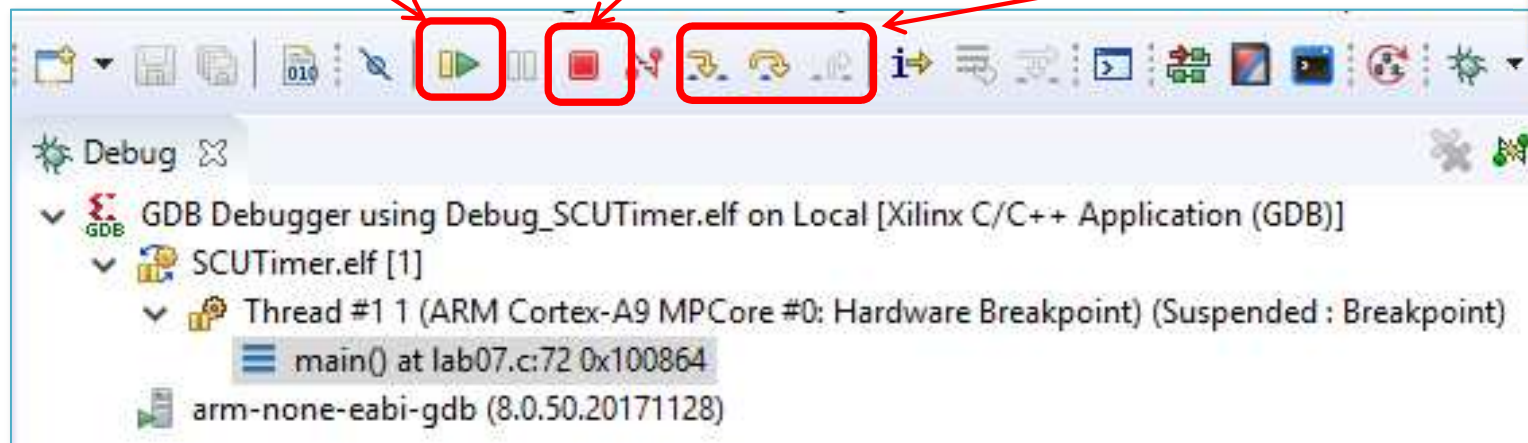
```
Console Tasks SDK Terminal
Connected to: Serial ( COM8, 115200, 0, 8 )
DIP Switch Status 7, F
Count value is: 0x1
Count value is: 0x2
DIP Switch Status F, B
DIP Switch Status B, 3
Count value is: 0x1
Count value is: 0x2
Count value is: 0x3
Count value is: 0x4
Count value is: 0x5
Count value is: 0x6
```

A program tesztelése a fejlesztő kártyán II.

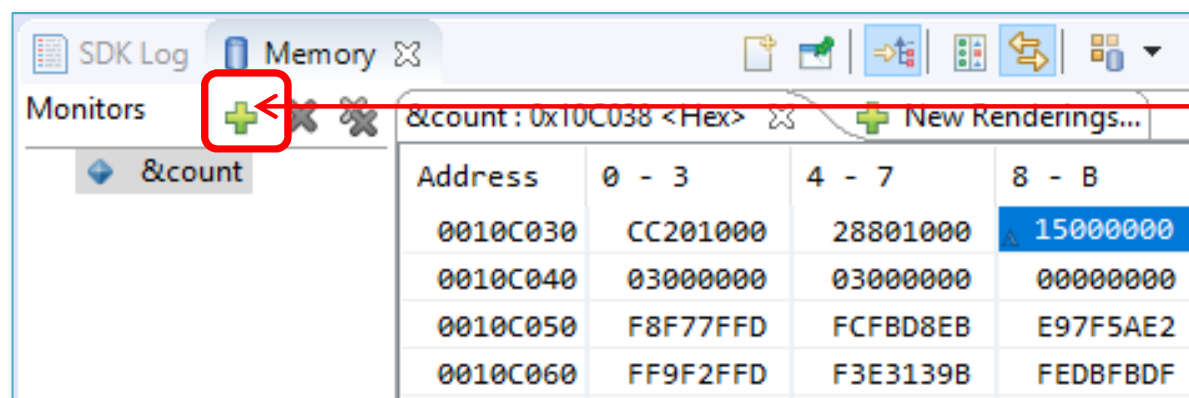
- Kattintsunk a **Resume** gombra
 - A program lefut egészen az első/következő beállított breakpoint-ig
- Kattintsunk a **Memory** fülre → Adjunk hozzá egy **Memory Monitor**-t

Resume (F8)

Terminate (F9)



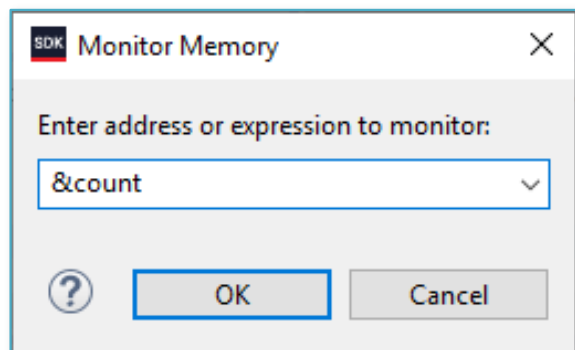
- Step Into (F5)
- Step Over (F6)
- Step Return (F7)



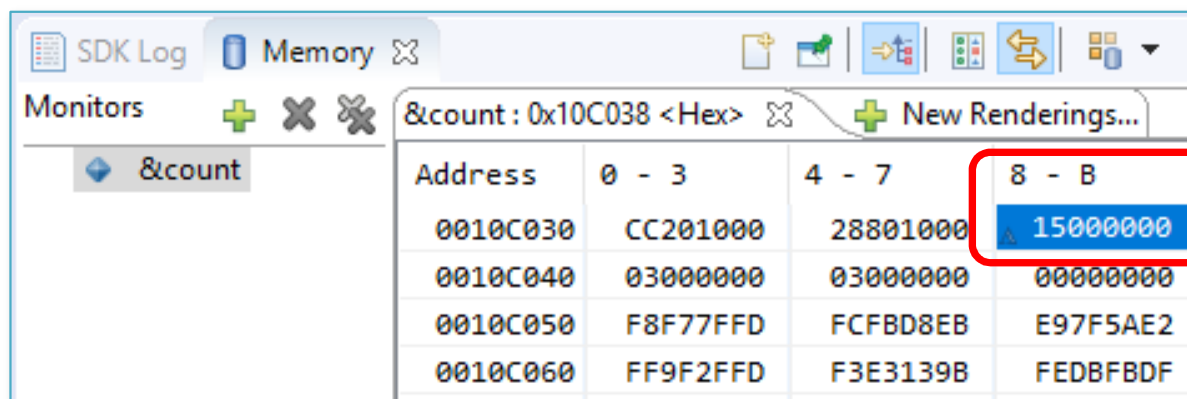
Memory Monitor hozzáadása: csak akkor láthatjuk a memória tartalmát, ha felfüggesztettük (pause) az alkalmazás debuggolását!

A program tesztelése a fejlesztő kártyán III.

- Vizsgáljuk meg a **count** változó értékét → **OK**



- Kattintsunk a **Resume** gombra
 - Minden **resume**-ra kattintással nő a **count** értéke
- Állítsuk le a program futását a **Terminate** gombbal



Address	0 - 3	4 - 7	8 - B
0010C030	CC201000	28801000	15000000
0010C040	03000000	03000000	00000000
0010C050	F8F77FFD	FCFBD8EB	E97F5AE2
0010C060	FF9F2FFD	F3E3139B	FEDBFBD

Resume-ra kattintva
változik a **count**
értéke

- Megtanultuk hogyan lehet *Vivado SDK-ban az ARM PS oldali Private Timer nevű időzítő-vezérlőt* kezelni
- Timer hívások, konfiguráció
- A belső Timer-t teszteltük a hardveren is (Digilent **ZyBo** kártya)
- Megnéztük az SDK Debugger használatát, láthattuk, hogyan módosul egy változó értéke, memória térkép, stb.