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

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Frissítve: 2019. május 10.

FPGA-alapú beágyazott rendszerek tervezése

Dr. Vörösházi Zsolt

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

Tárgyalt ismeretkörök

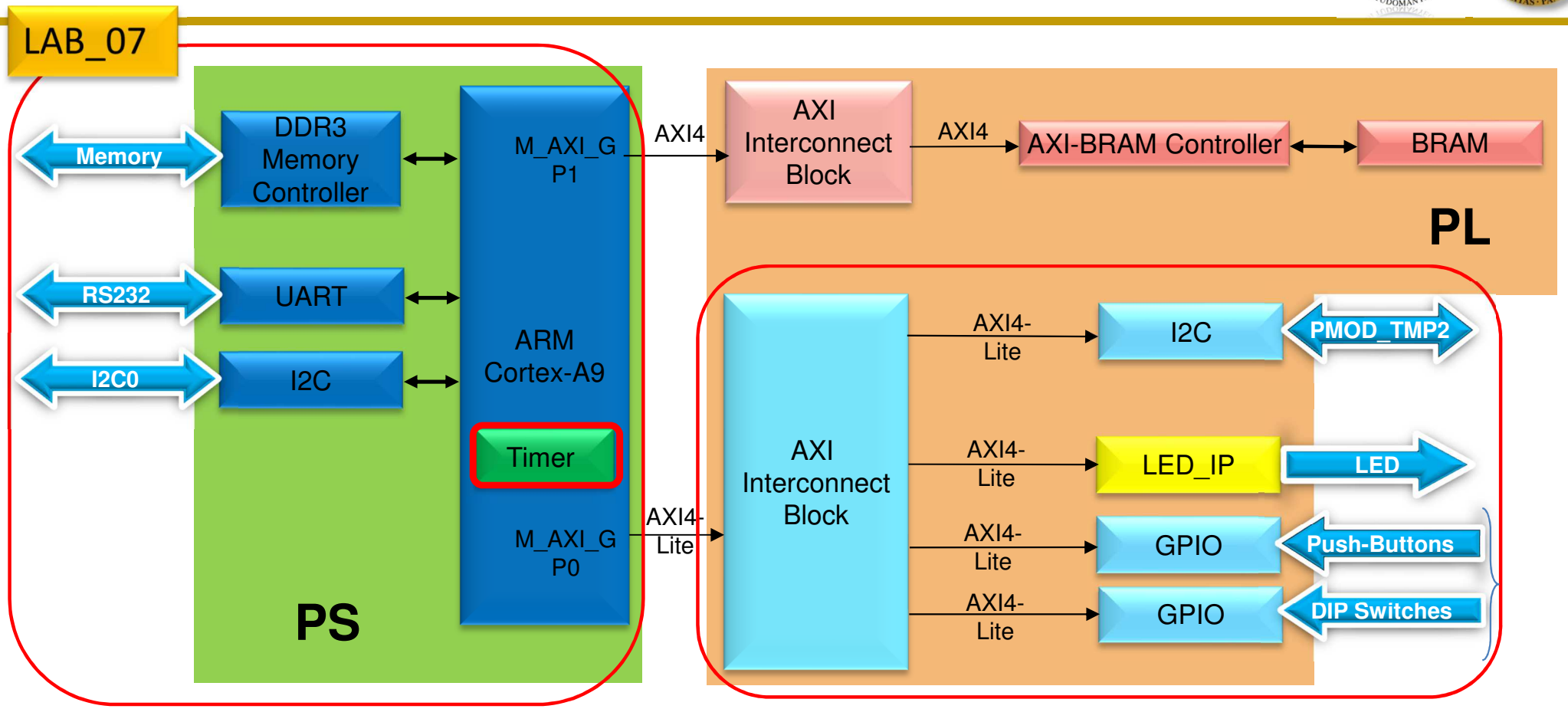
1. Bevezetés – Beágyazott rendszerek.
2. FPGA-k, Digilent ZYBO fejlesztő kártyák és eszközök.
3. Beágyazott Rendszer fejlesztő szoftverkörnyezet (Xilinx Vivado Embedded Development) áttekintése.
4. Beágyazott alap tesztrendszer (BSB - Base System Builder és Board Bring-Up) összeállítása Vivado-ban.
5. Perifériák hozzáadása (IP adatbázisból) az összeállított beágyazott alaprendszerhez.
6. Saját periféria hozzáadása az összeállított beágyazott alaprendszerhez
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

Feladatok

- 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 bővítendő tesztrendszer



PS oldal:

- ARM hard-processzor 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

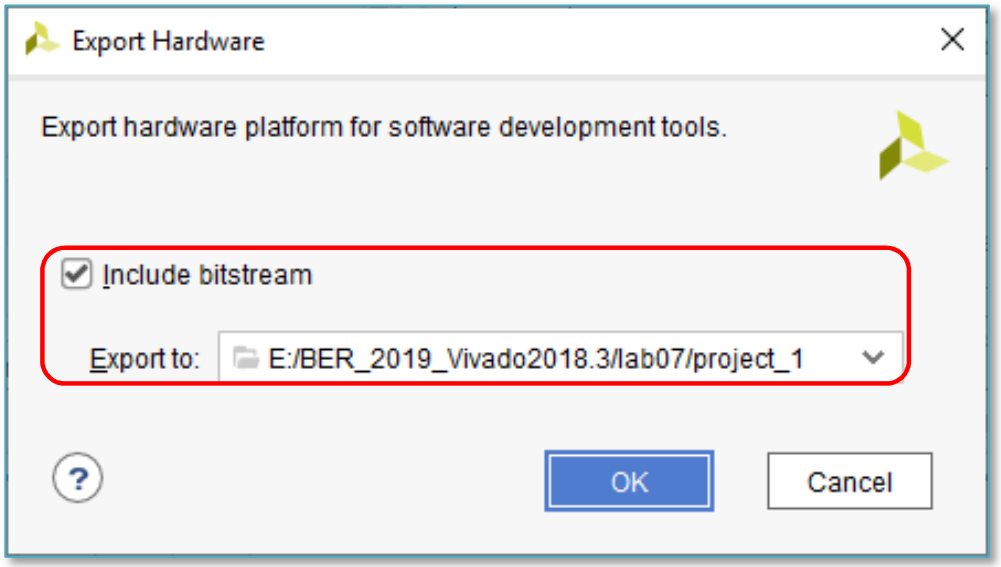
Időzítő és megszakítás vezérlő illesztése I.

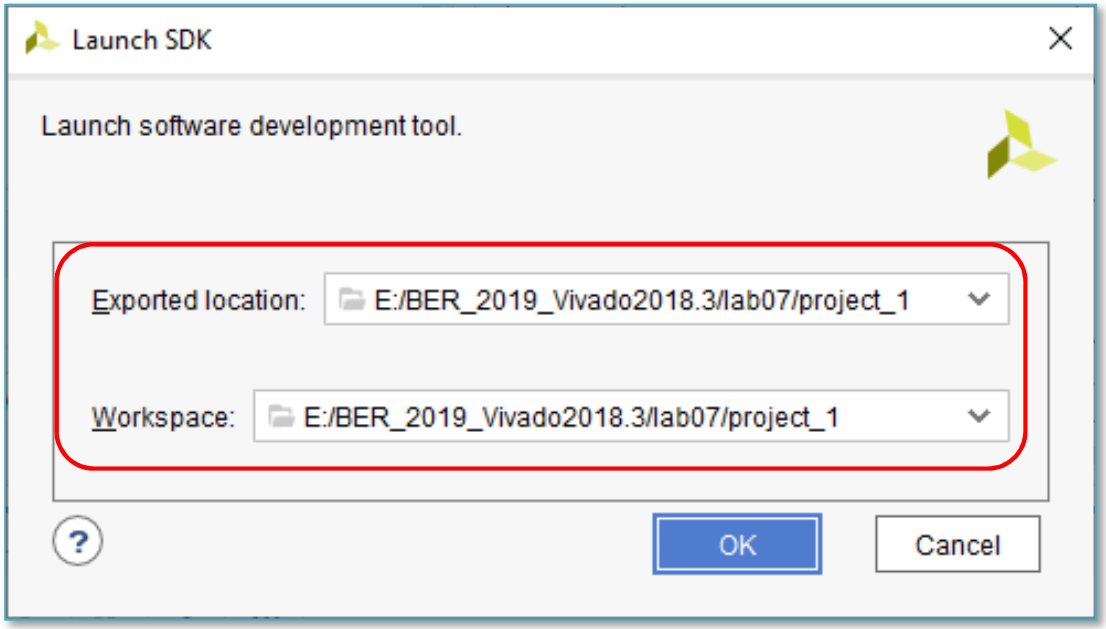
- Hozzunk létre egy új mappát, legyen a neve \LAB07
  - Archiváljuk és másoljuk át az előző ismeretkör (\LAB02_B) elsajátításakor létrehozott projektet
  - Project -> Save as …
- Indítsuk el az Vivado környezetet
- Mivel a CPU (ARM) belső/private Timer-ét használjuk, nem kell újrafordítani a FW-t!
-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

Tesztalkalmazás készítése I.

- 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

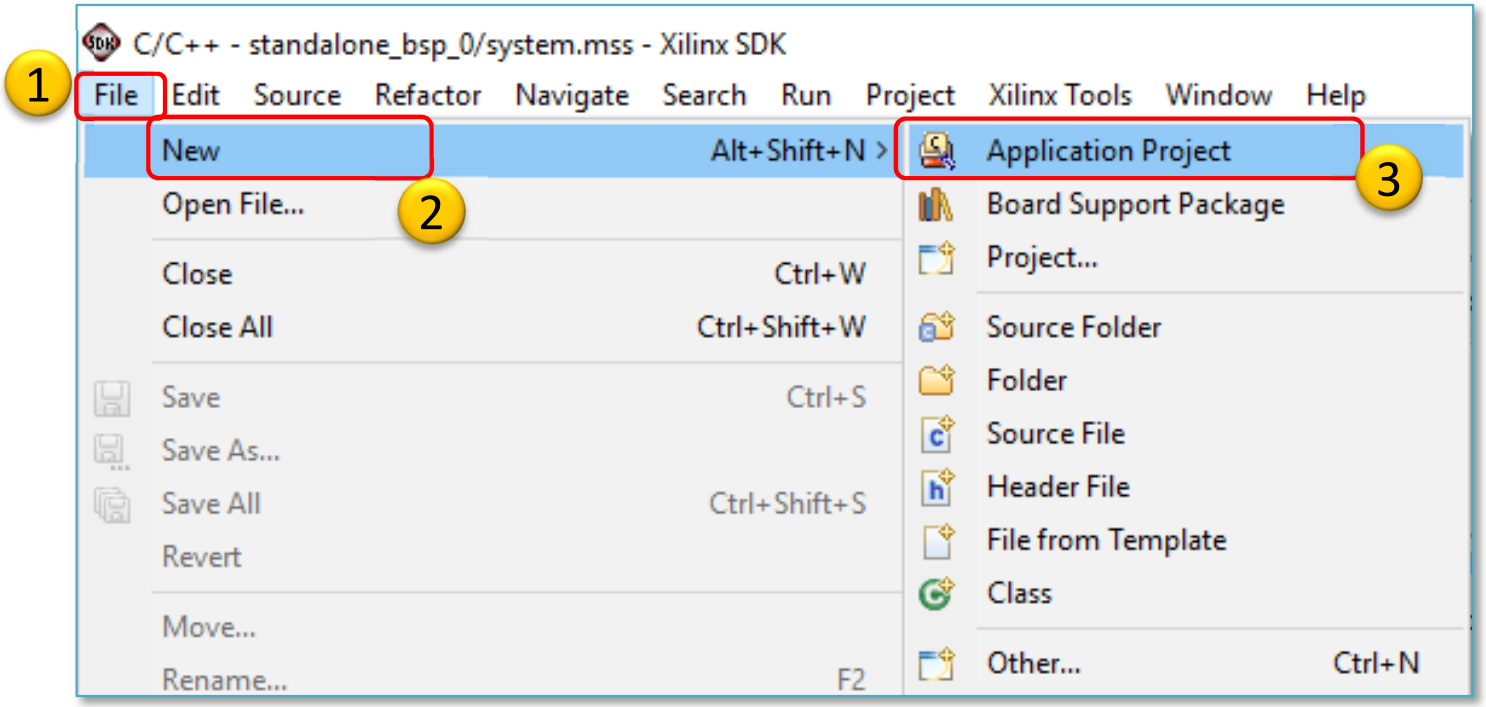
system.hdf

IP blocks present in the design

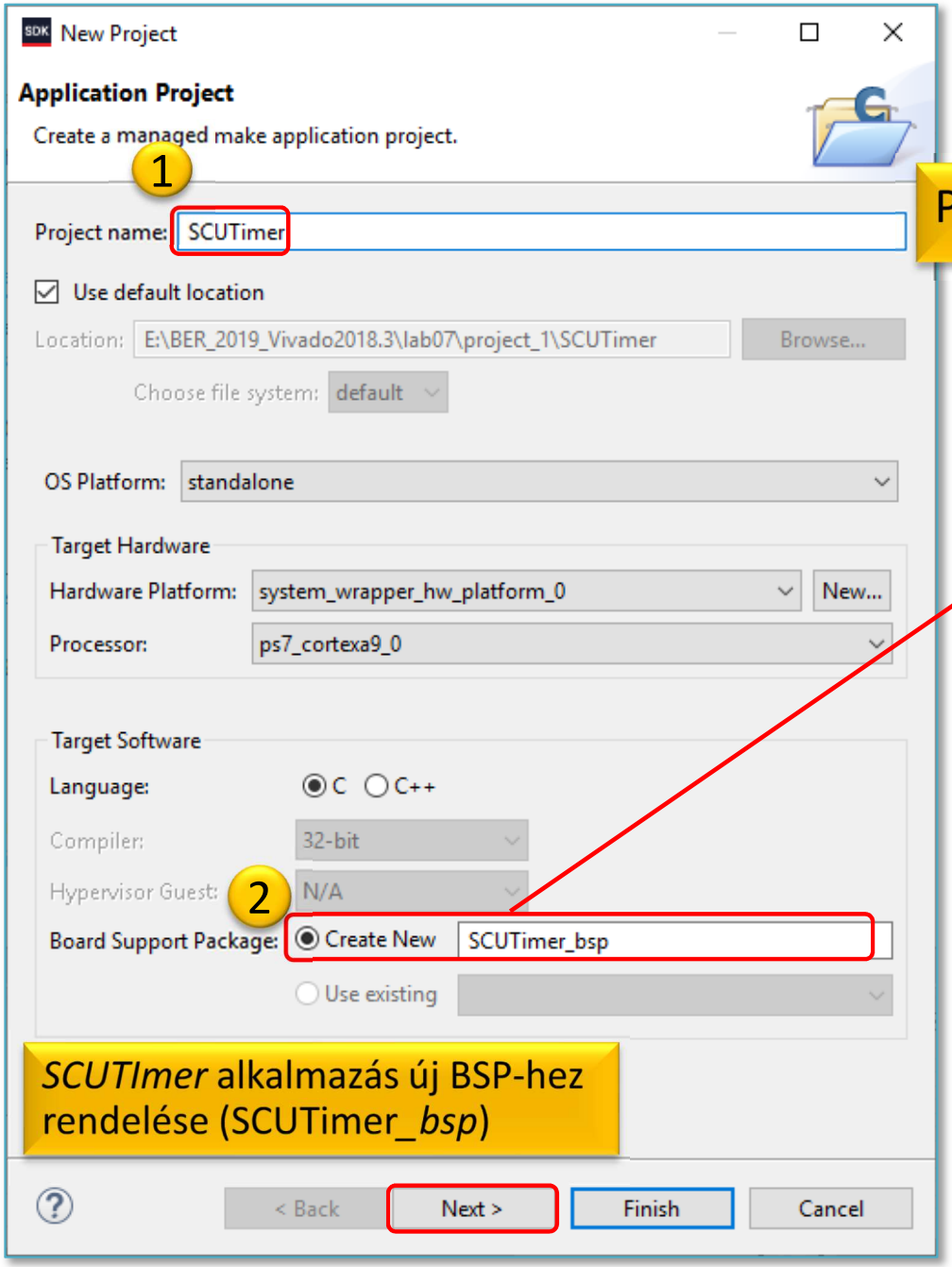
ps7_0_axi_periph	axi_interconnect	2.1	
ps7_intc_dist_0	ps7_intc_dist	1.00.a	
leds_4bit	axi_gpio	2.0	Registers
rst_ps7_0_100M	proc_sys_reset	5.0	
ps7_scutimer_0	ps7_scutimer	1.00.a	
ps7_slcr_0	ps7_slcr	1.00.a	
dip	axi_gpio	2.0	Registers
ps7_scuwdt_0	ps7_scuwdt	1.00.a	
ps7_l2cachec_0	ps7_l2cachec	1.00.a	

Tesztalkalmazás készítése II.

- Új projekt alkalmazás létrehozása (Application Project)

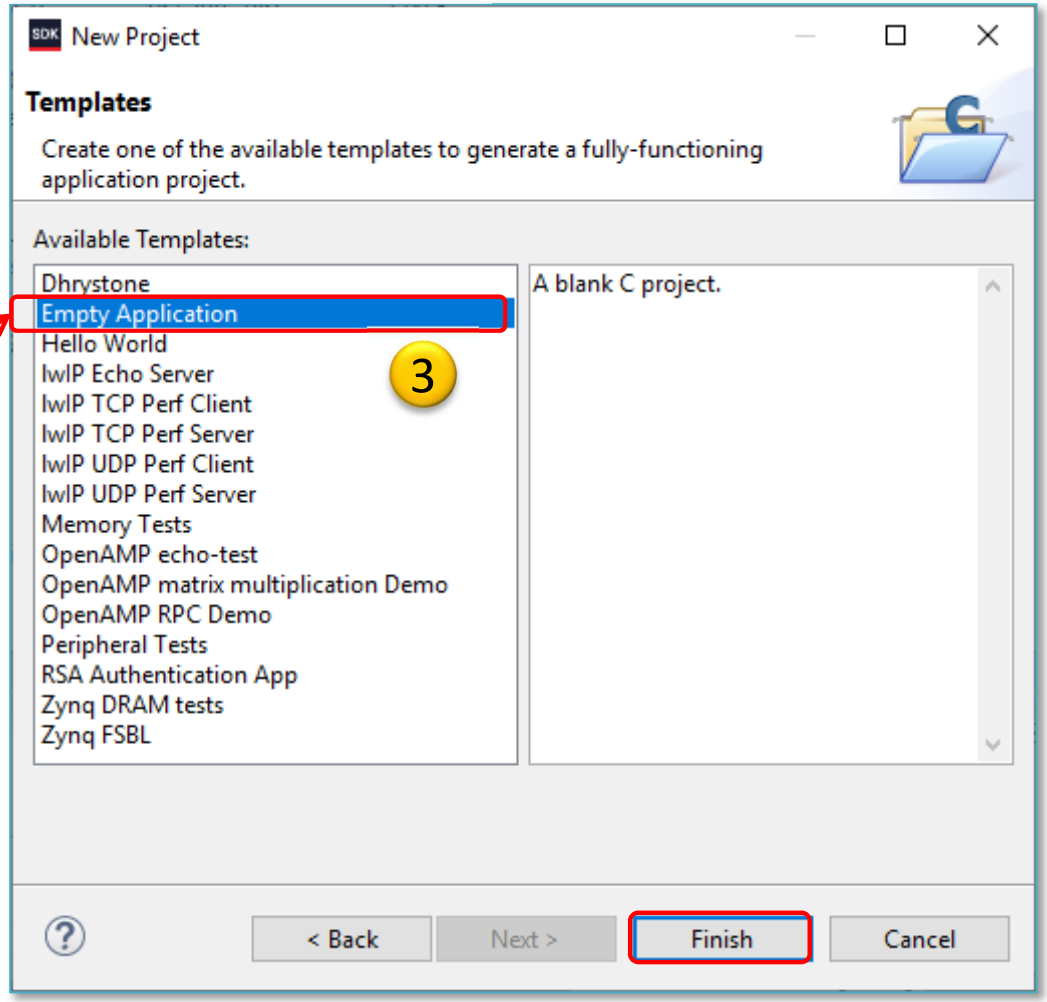


Tesztalkalmazás (SCUTimer) készítése III.



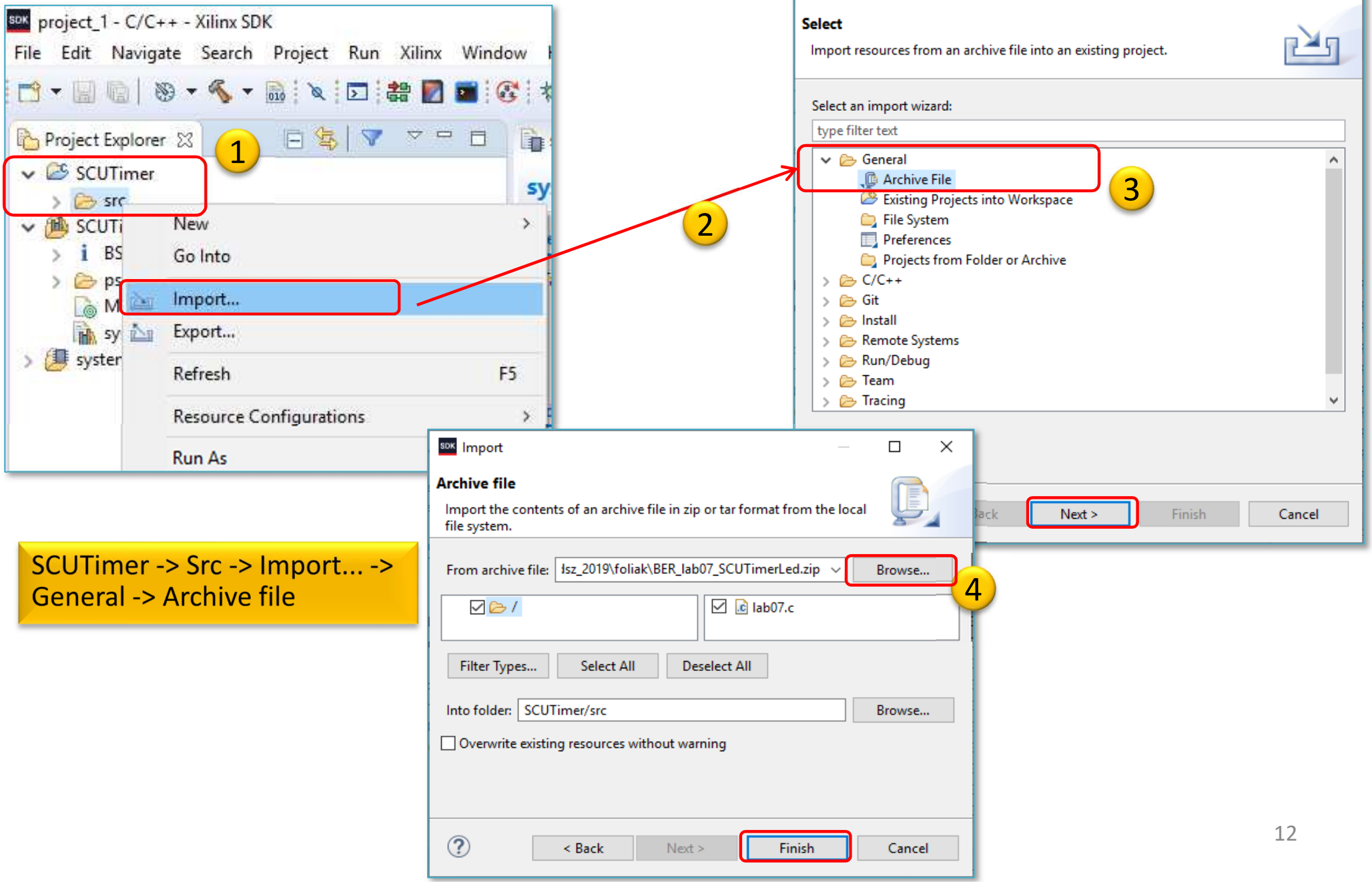
Projekt neve: *SCUTimer*

SCUTImer alkalmazás új BSP-hez rendelése (SCUTimer_*bsp*)



Tesztalkalmazás (TestApp) készítése V.

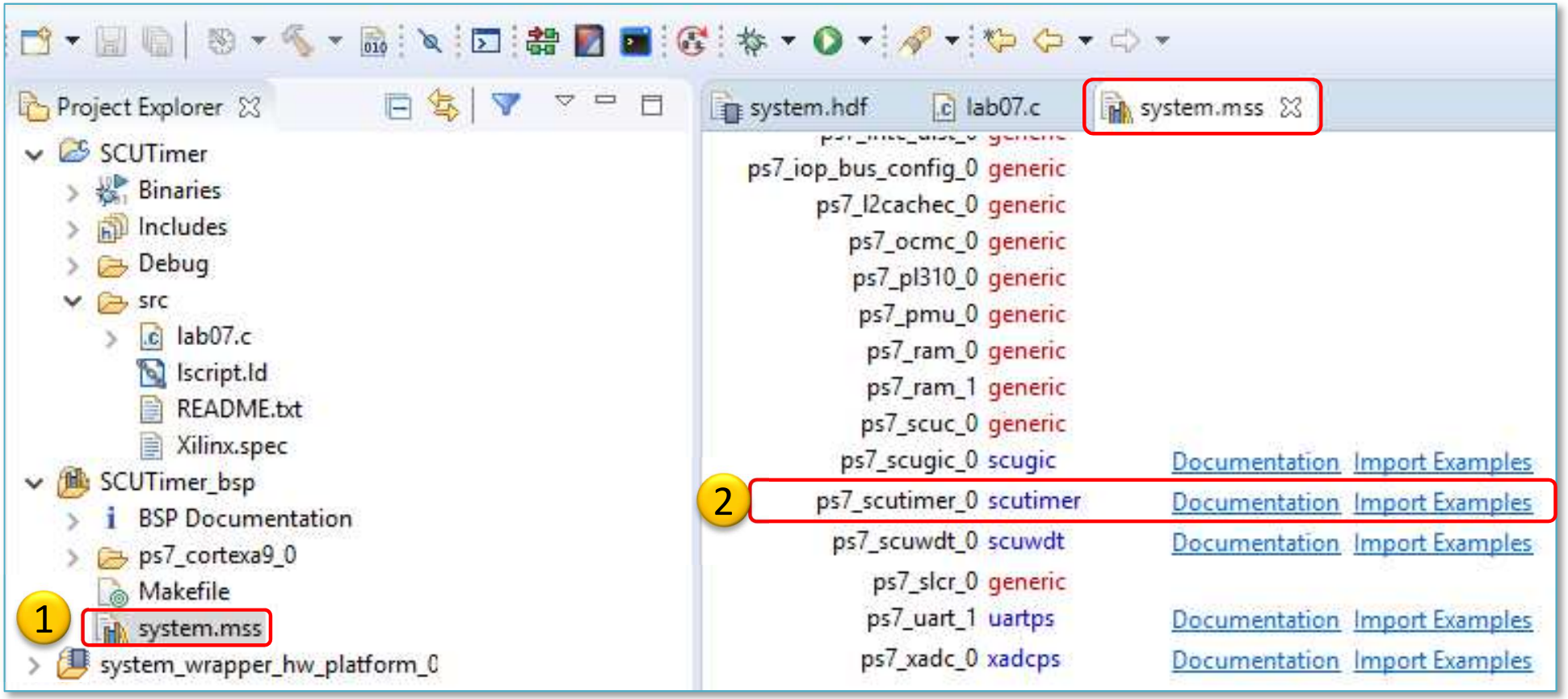
- C forrás fájl hozzáadása (`lab07.c`):



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

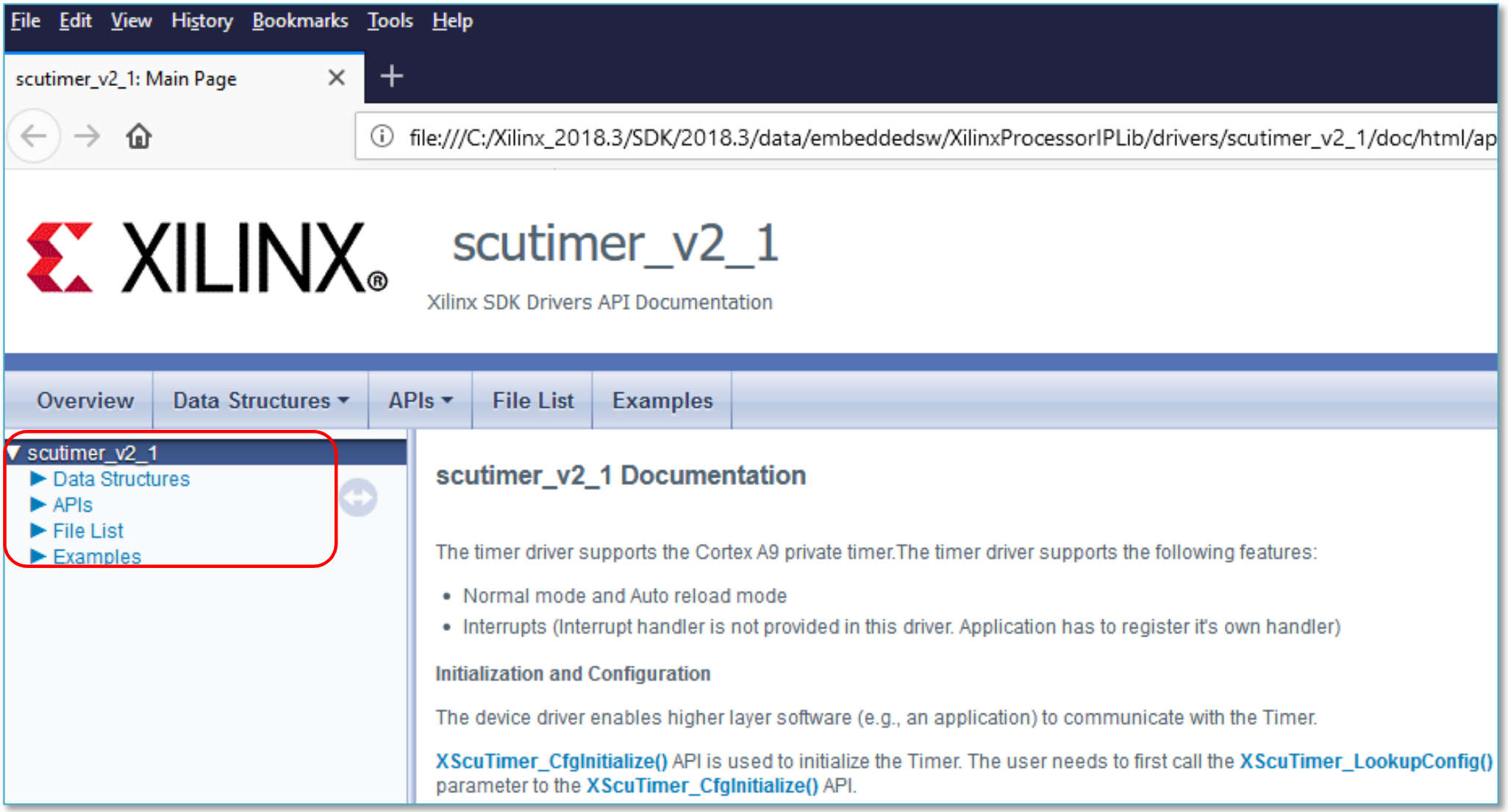
SCU Timer API driver

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

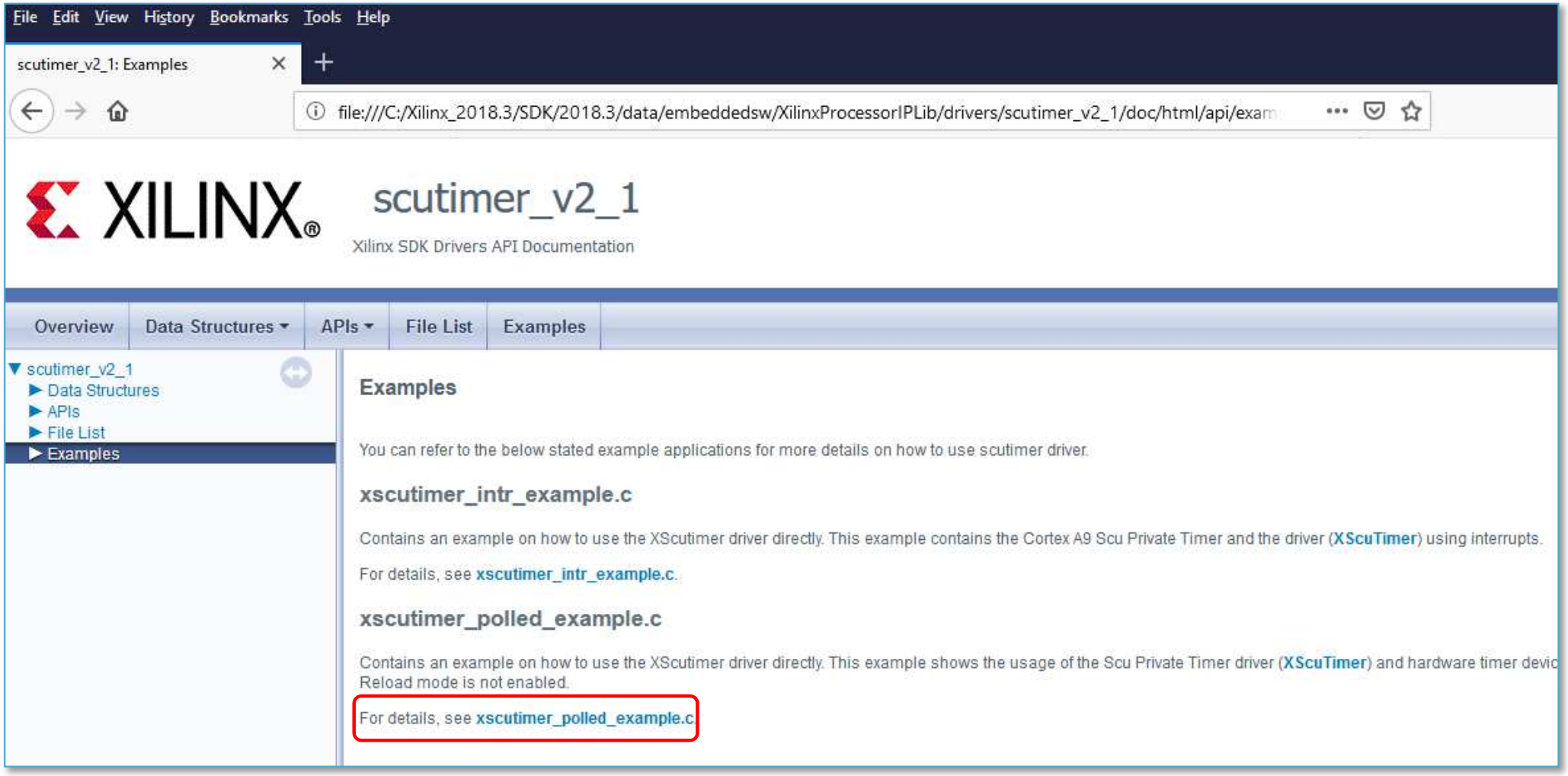


Driver API: dokumentáció (html), és példaprogramok elérhetőek (**polling** vs. interrupt változatban is)

SCU Timer API driver dokumentáció



SCU Timer API példaprogramok



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

Data Structures

struct	**XScuTimer_Config**
struct	**XScuTimer**

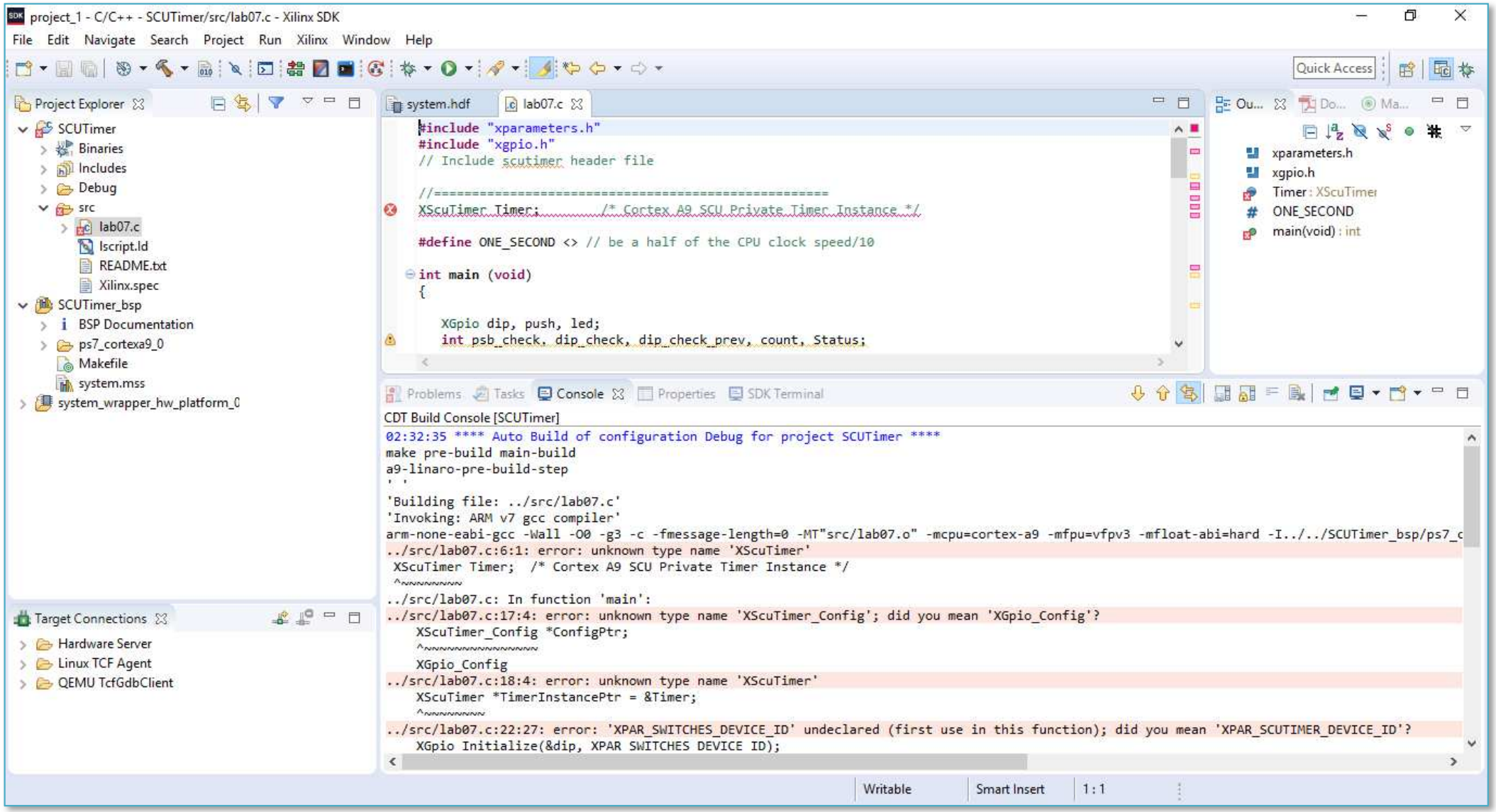
Defines

#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

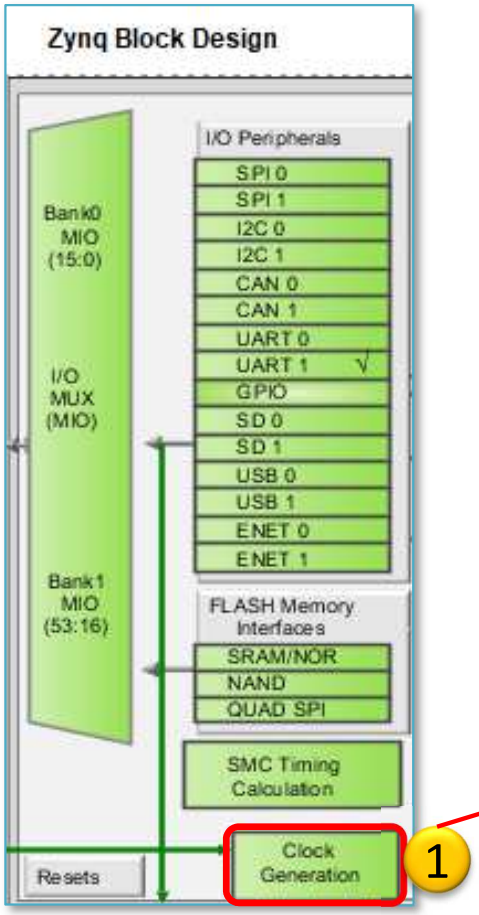


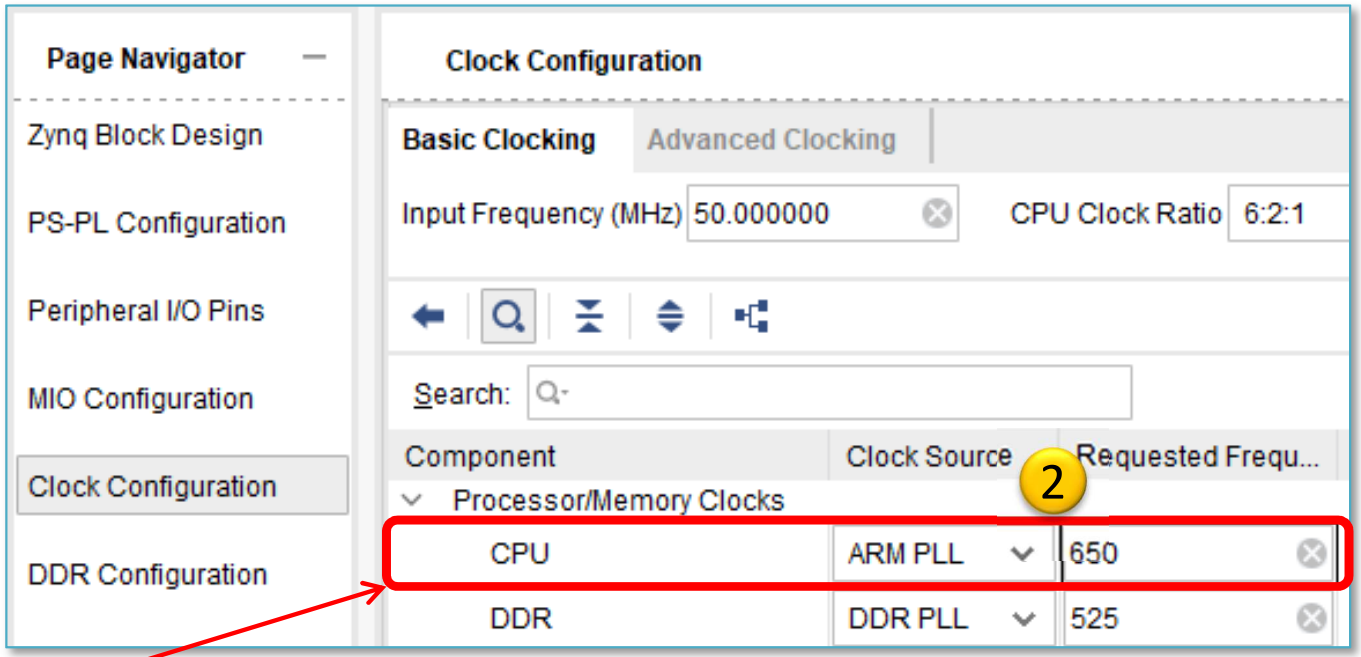
SCU Timer alkalmazás: ellenőrzés és bővítés I.

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

```
#define XPAR_PUSH_DEVICE_ID XPAR_PB_DEVICE_ID
```

Vivado:






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

```
#define ONE_SECOND 325000000 // 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
```

SCU Timer alkalmazás: ellenőrzés és bővítés II.

```
//=====================================================
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);
```

SCU Timer alkalmazás: ellenőrzés és bővítés III.

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:

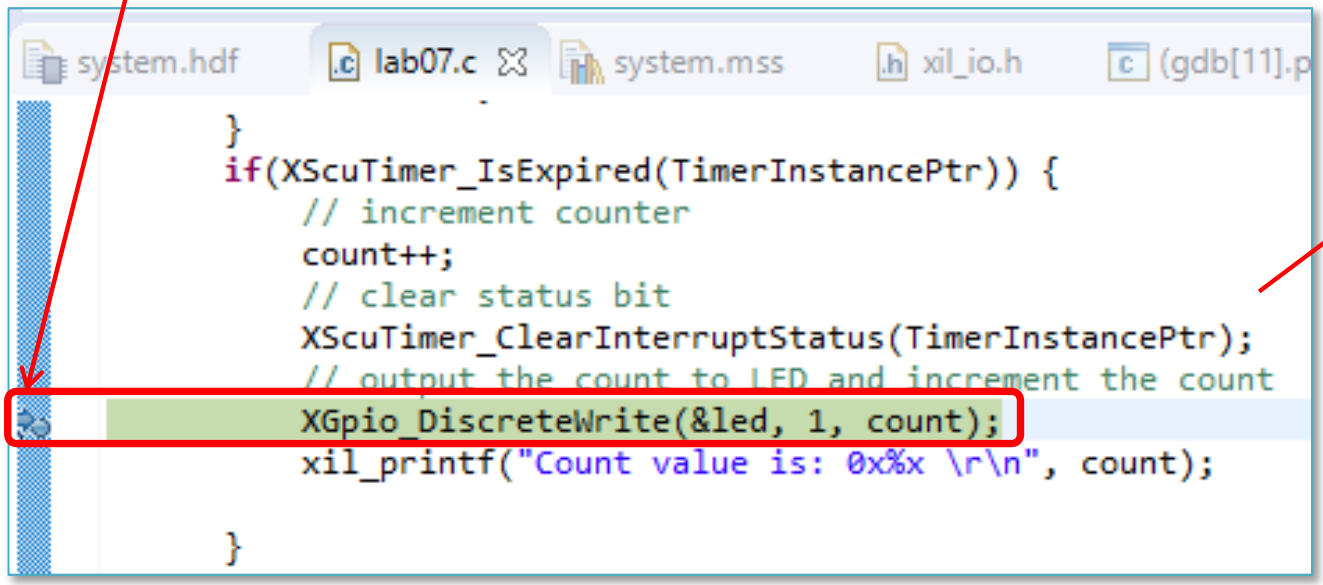
```
// load timer with the new switch settings
 XScuTimer_LoadTimer(TimerInstancePtr, ONE_SECOND*dip_check);
```

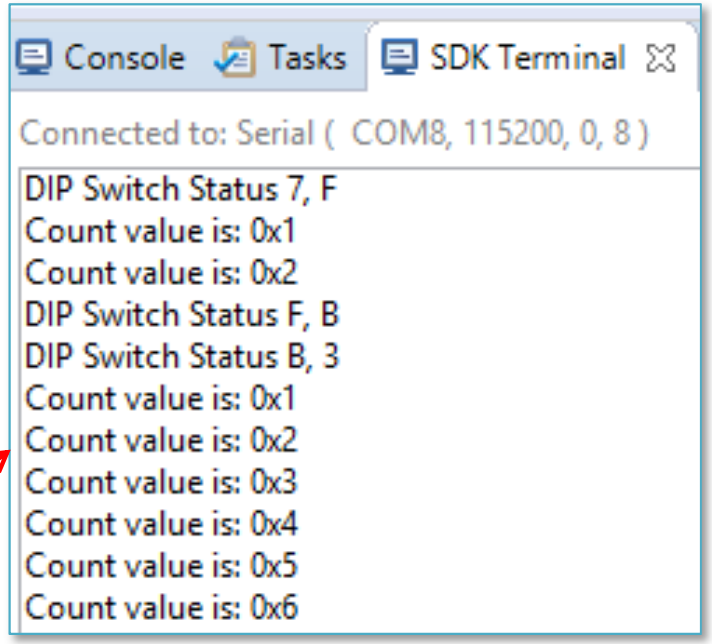
7.) Timer státusz bitjének törlése, és counter értékének LED-re íratása (67. - 73. sorok):

```
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.

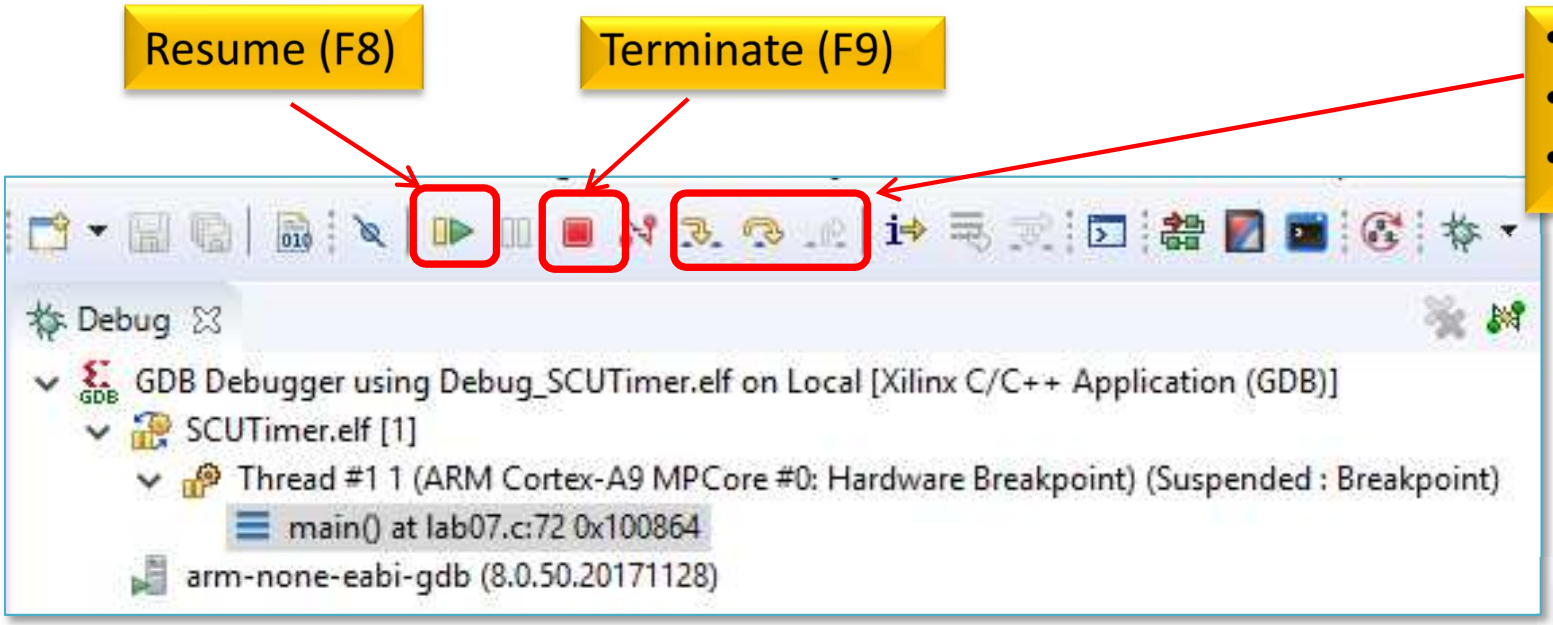
- Debugger elindítása
  - **Project Explorer →** Jobb kattintás a `\SCUTimer` projektre **→ Debug As → Launch on Hardware → Yes (Debug perspektíva engedélyezése)**
    - A `SCUTimer.elf` letöltődik
- Jobb kattintás a **Variables** fül-re **→ Add Global Variables →** **`count`** változó kiválasztása **→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



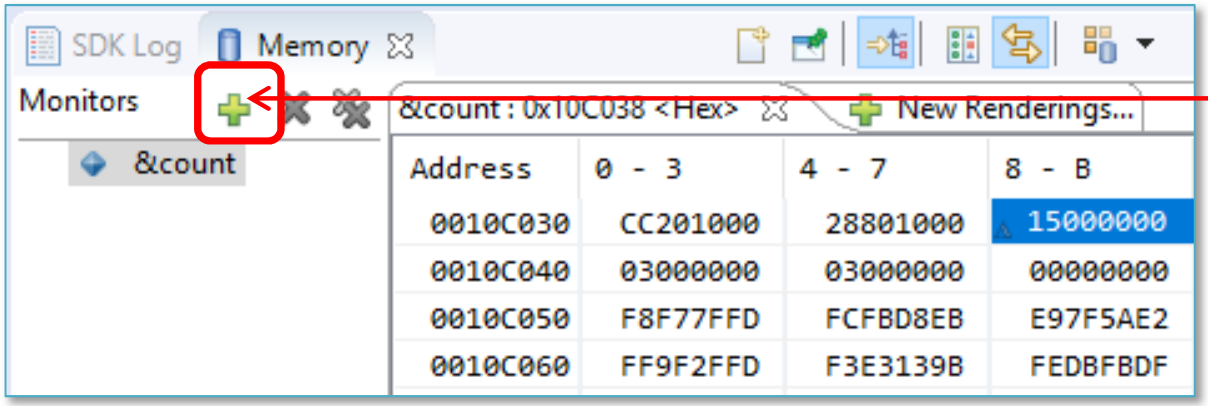


A program tesztelése a fejlesztő kártyán II.

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



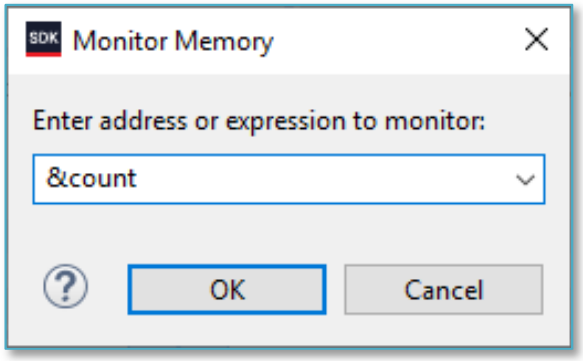
- Step Into (F5)
- Step Over (F6)
- Step Return (F7)



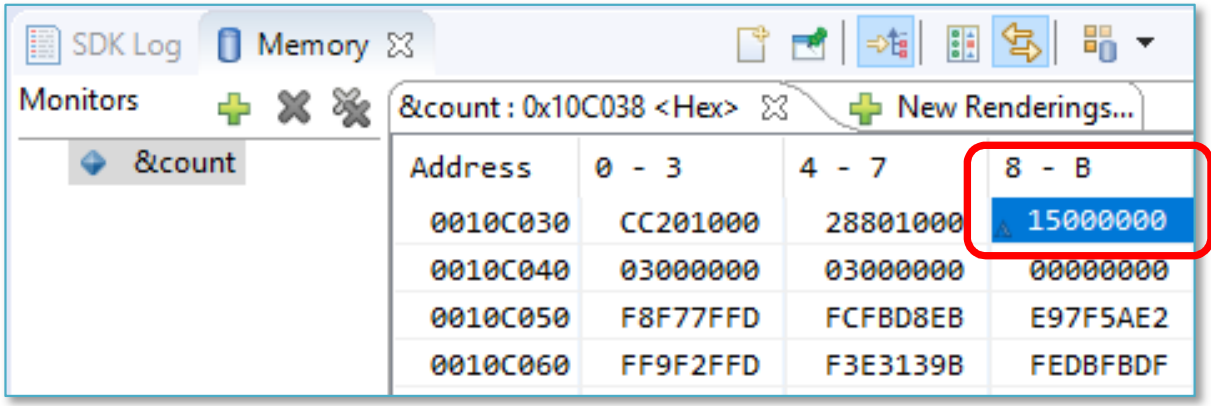
Memory Monitor hozzáadása: csak akkor láthatjuk a memória tartalmat, ha felfüggesztettük (pause) az alkalmazás debuggolását!

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



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

Összefoglalás

- 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.