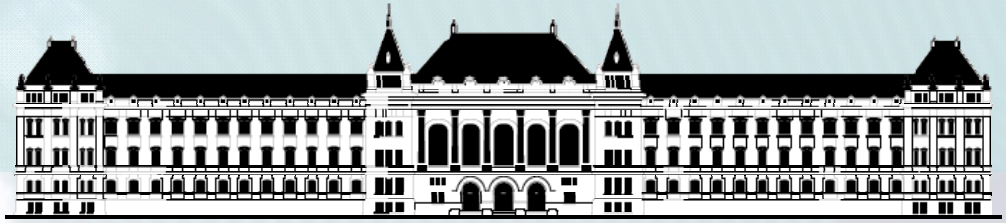


BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Rendszerarchitektúrák labor
Xilinx EDK

Raikovich Tamás

BME MIT

Labor tematika (Xilinx EDK)

- **1. labor:**
  - A Xilinx EDK fejlesztői környezet ismertetése
- **2. labor:**
  - Egyszerű processzoros rendszer összeállítása
  - Egyszerű szoftver alkalmazások készítése
- **3. labor:**
  - Saját periféria illesztése
  - Megszakításkezelés
  - HW/SW együttes fejlesztés (debugger, ChipScope)

Témakörök

- Beágyazott rendszerek
- MicroBlaze processzor
- EDK alapok
- Gyári és saját IP-k hozzáadása
- ***Szoftverfejlesztés***
- HW és SW együttes fejlesztése

Asztali vs. beágyazott SW fejlesztés

Asztali rendszerek:

- **Fejlesztés, hibakeresés és futtatás ugyanazon a gépen**
- **Az OS akkor tölti be a programot a memóriába, ha a felhasználó ezt kéri**
- **Címek feloldása**
  - Az alkalmazás betöltésekor
  - A betöltő az OS része

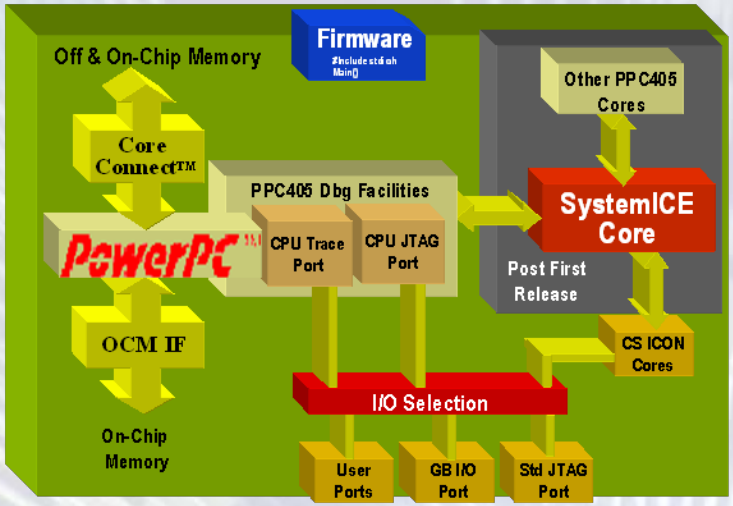
Beágyazott rendszerek:

- **Fejlesztés: a host gépen**
- **Futtatás: a célrendszeren**
- **Egy futtatható állomány**
  - ELF fájl
  - Bootloader, alkalmazás, ISR, operációs rendszer
  - Címek feloldása linkeléskor
- **Futtatható kód letöltése a célrendszerre**
  - JTAG, Ethernet, soros port
  - FLASH programozó

Asztali vs. beágyazott SW fejlesztés

- A fejlesztés külön számítógépen (host) történik, az alkalmazást le kell tölteni a célrendszerre
- A keresztfordító a host gépen fut

Célrendszer



Host számítógép

Asztali vs. Beágyazott SW fejlesztés

Különféle problémák:

- **Minden terv esetén egyedi a hardver**
- **Megbízhatóság**
- **Valósidejű válasz megkövetelése**
  - RTOS ↔ OS
- **Kis méretű, kompakt kód**
- **Magasszintű nyelvek (C/C++) ↔ assembly**

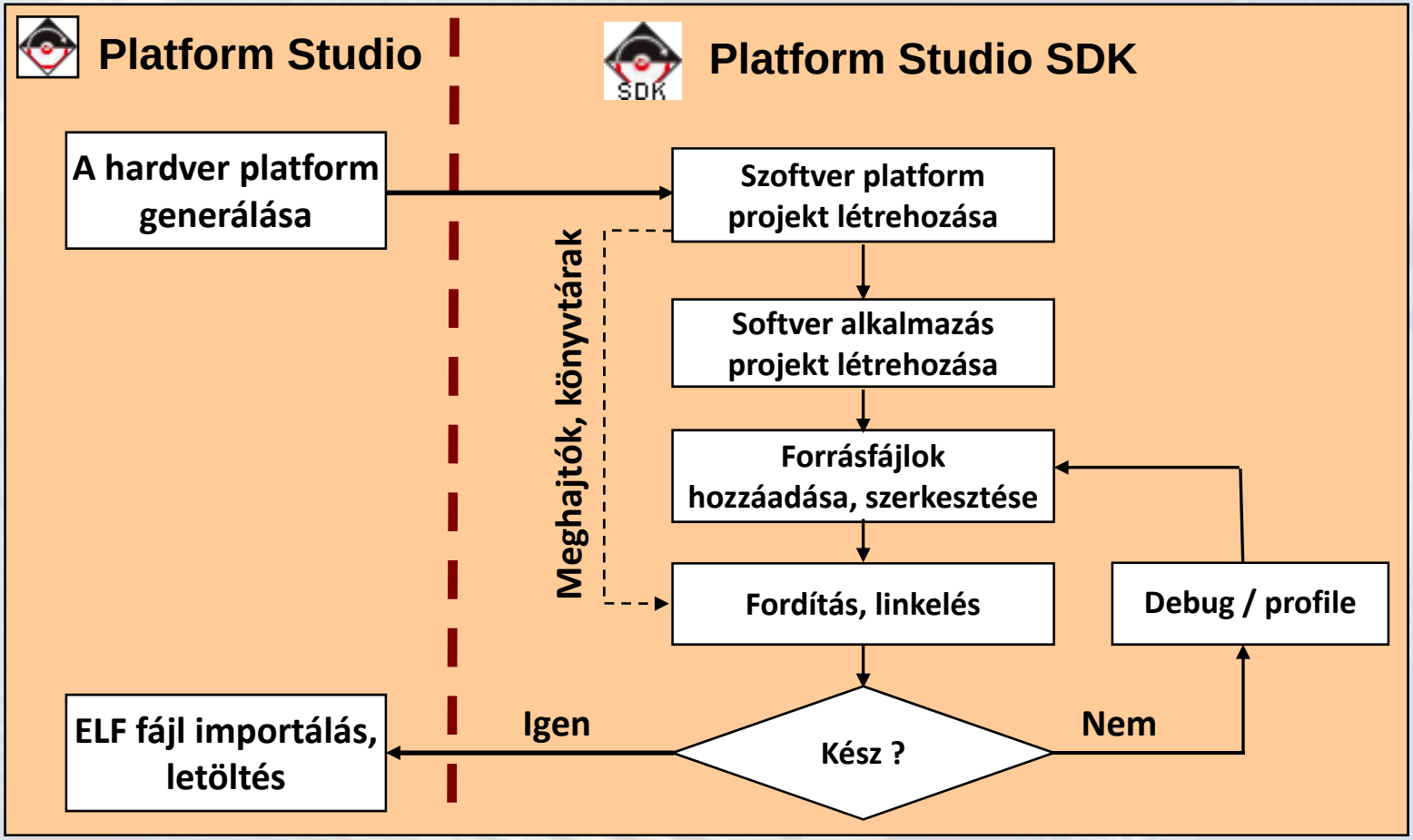
Software Development Kit (SDK)

Xilinx Software Development Kit (SDK):

- Java/Eclipse alapú alkalmazás fejlesztői környezet
- Szolgáltatásokban gazdag C/C++ kód szerkesztő és fordító környezet
- Projekt menedzsment
- Automatikus konfigurációs fájl és makefile generálás
- Hiba navigálás
- Jól használható környezet a beágyazott rendszerek hibakereséséhez (debug)
- Verziókövetés

Software Development Kit (SDK)

A fejlesztés folyamata:



Software Development Kit (SDK)

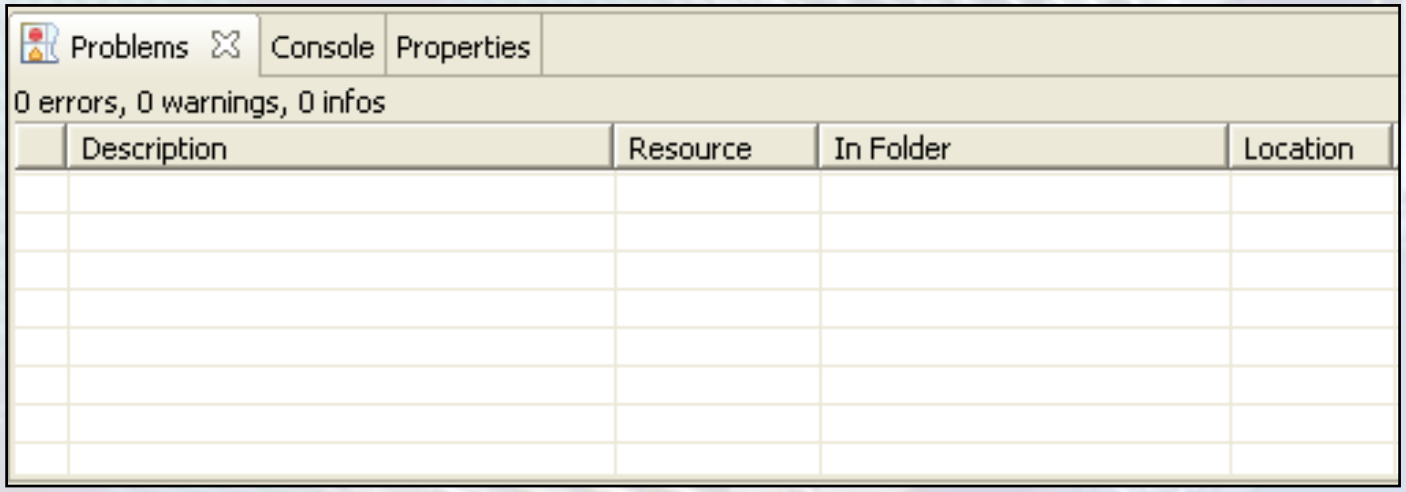
Munkaterületek és perspektívák:

- **Munkaterüket (workspace)**
  - Itt tárolódnak
    - A projektekkel kapcsolatos beállítások
    - Egyéb belső adatok
  - A felhasználók számára transzparens
  - A projekt fájlokat tartalmazó könyvtárak is itt vannak
- **Nézetek (views) és szerkesztők (editors)**
  - Alapelemek a felhasználói felületen
- **Perspektívák (perspectives)**
  - Azonos funkcióhoz kapcsolódó nézetek gyűjteménye
  - A nézetek szabadon elrendezhetők a perspektíván belül

Software Development Kit (SDK)

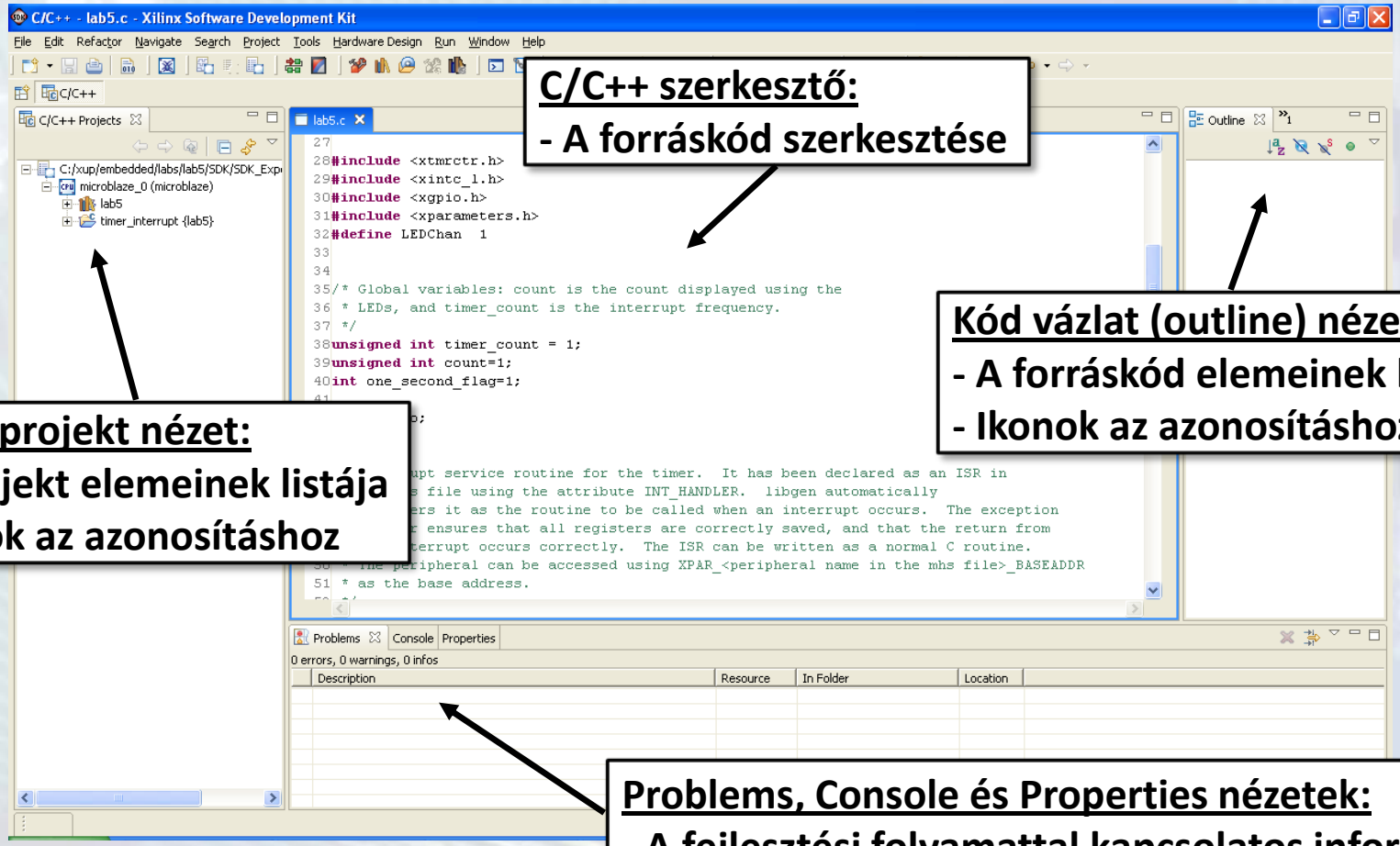
Nézetek (views):

- **Eclipse Platform nézetek:**
  - Navigátor, feladatok (tasks), hibaüzenetek (problems)
- **Debug nézetek:** verem, változók, regiszterek
- **C/C++ nézetek:** projekt, vázlat (outline)



Software Development Kit (SDK)

C/C++ perspektíva:



C/C++ szerkesztő:
- A forráskód szerkesztése

C/C++ projekt nézet:
- A projekt elemeinek listája
- Ikonok az azonosításhoz

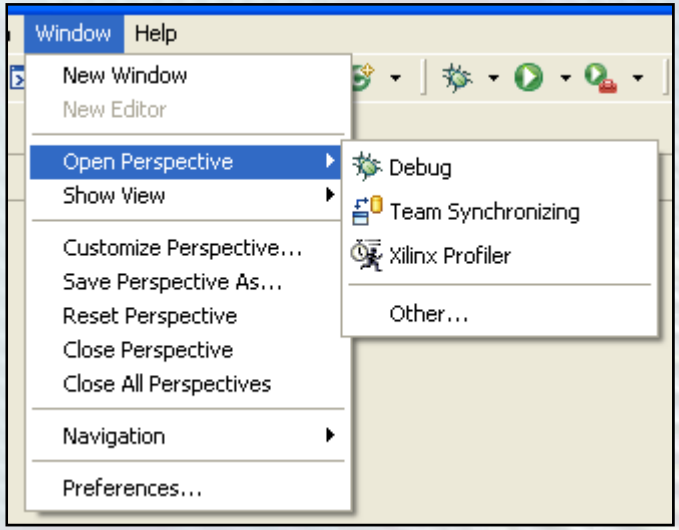
Kód vázlat (outline) nézet:
- A forráskód elemeinek listája
- Ikonok az azonosításhoz

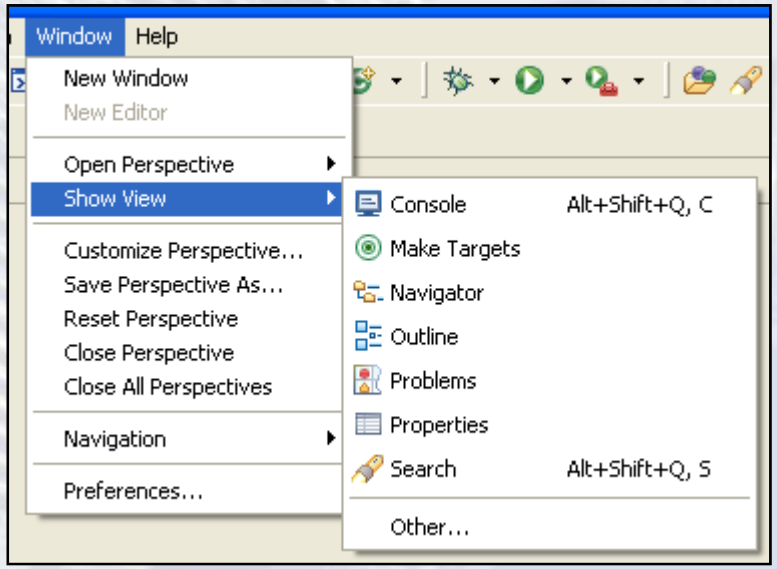
Problems, Console és Properties nézetek:
- A fejlesztési folyamattal kapcsolatos információk

Software Development Kit (SDK)

Perspektívák és nézetek:

- **Perspektívák megnyitása**
  - ***Window*** menü → ***Open Perspective***
- **Nézetek megnyitása**
  - ***Window*** menü → ***Show View***
  - Ha a kiválasztott nézet már látható, akkor előtérbe kerül

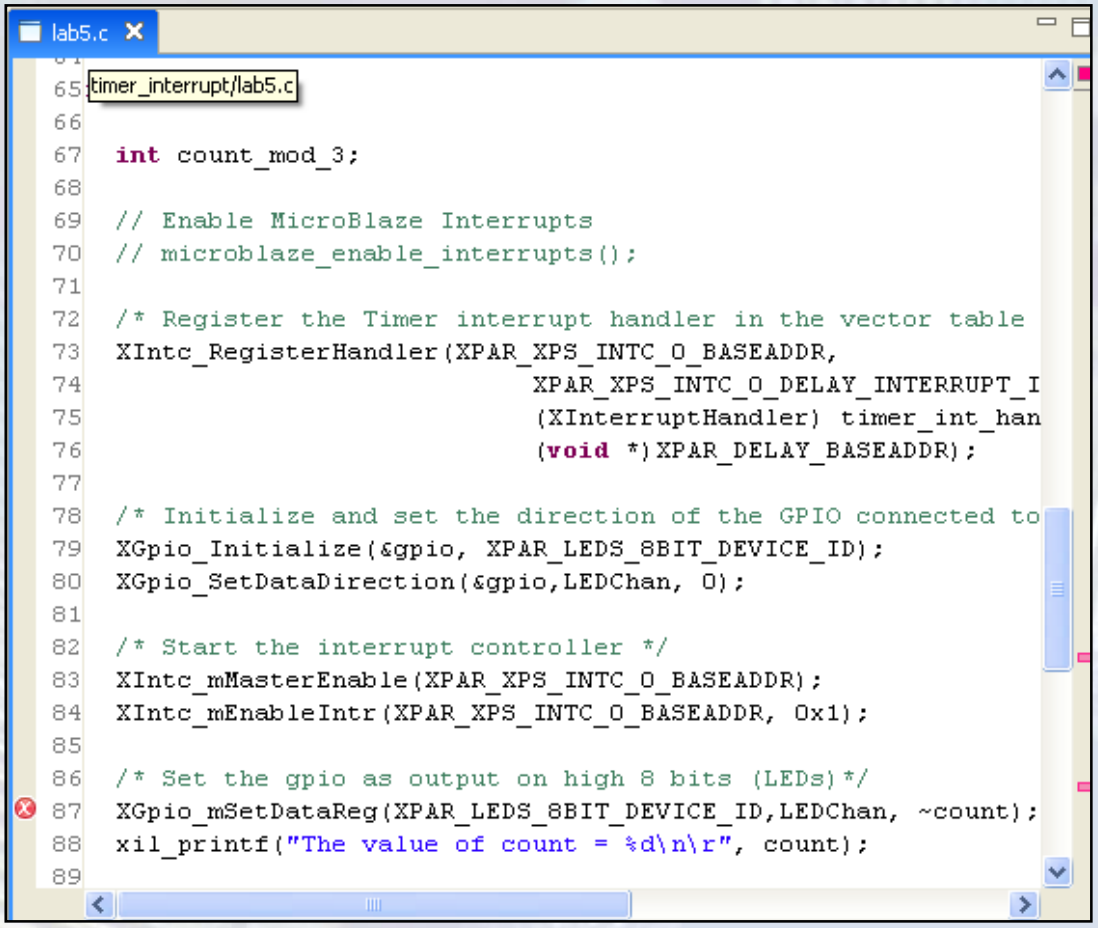




Software Development Kit (SDK)

Forráskód szerkesztő:

- Szintaxis kiemelés
- Zárójelek illesztése
- Content assist
- Refactoring
- Billentyűparancsok



SDK projekt létrehozása

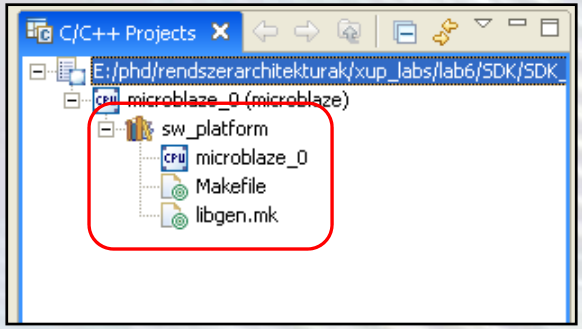
A hardver rendszer exportálása az SDK-ba:

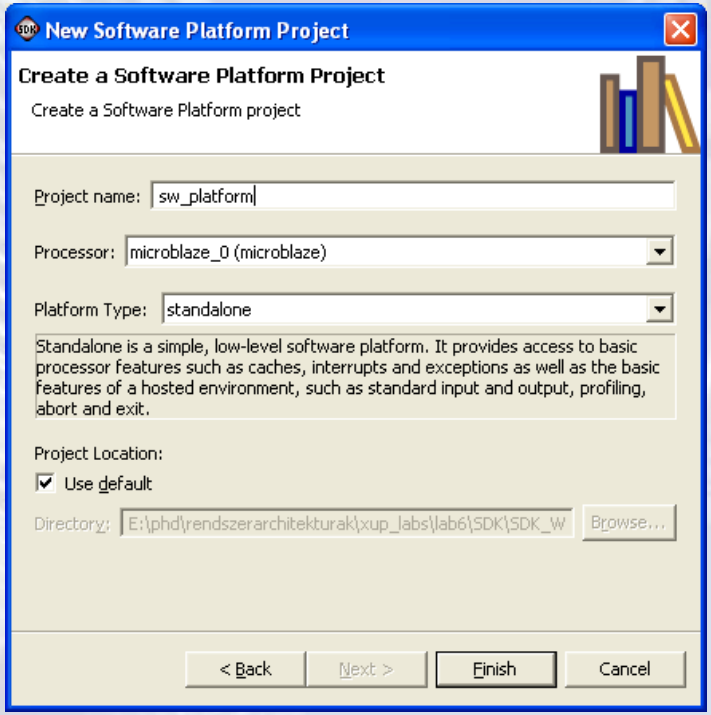
- **A Xilinx Platform Studio-ban:**
  - ***Project*** menü → ***Export Hardware Design to SDK*** vagy
  - A  gomb a toolbar-on
- **A megjelenő ablakban**
  - Az ***Include bitstream and BMM file*** legyen bejelölve
  - Kattintsunk az ***Export & Launch SDK*** gombra

SDK projekt létrehozása

A szoftver platform projekt létrehozása (ha még nem volt):

- **A szoftver platform projekt: meghajtók és könyvtárak**
- **Létrehozása**
  - ***File*** menü → ***New*** → ***Software Platform…*** vagy
  - Projekt nézet: jobb klikk → ***New*** → ***Project…*** → ***Software Platform***
- **A megjelenő ablakban**
  - Projekt elnevezése (pl. ***sw_platform***)
  - Processzor: ***microblaze_0***
  - Platform Type: ***standalone***
  - Projekt helye: alapértelmezett hely
- **Lefordítás: jobb klikk → *Build Project***





Szoftver platform projekt

Eszközmeghajtó programok:

- **Rétegzett architektúra**
- **2. réteg: RTOS alkalmazási réteg**
- **1. réteg: Magas szintű eszközmeghajtók**
  - Teljes funkcionalitás
  - Többféle processzor és operációs rendszer támogatott
- **0. réteg: Alacsony szintű eszközmeghajtók**

2. réteg: RTOS
1. réteg: magas szintű meghajtók
0. réteg: alacsony szintű meghajtók

Szoftver platform projekt

0. réteg: alacsony szintű eszközmeghajtó programok

- **Makrók és függvények, melyek lehetővé teszik a kisméretű rendszerek megvalósítását**
- **Tulajdonságok:**
  - Kis memóriaigény
  - Egyáltalán nincs vagy csak kevés hibaellenőrzés
  - Csak az eszköz alapfunkcióit támogatják
  - Nem támogatják az eszköz konfigurációs paramétereket
  - Több eszközpéldány támogatása: báziscímek megadásával
  - Csak lekérdezéses I/O
  - Blokkoló hívások
  - Header fájlok végződése “_l” (például xuartlite_l.h)

Szoftver platform projekt

1. réteg: magas szintű eszközmeghajtó programok

- **Makrók és függvények, melyek lehetővé teszik az eszközök minden tulajdonságainak kihasználását**
- **Tulajdonságok:**
  - Absztrakt API, amely leválasztja a programozói interfészt a hardver rendszer változásairól
  - Támogatják az eszköz konfigurációs paramétereket
  - Több eszközpéldány támogatása
  - Lekérdezéses és megszakításos I/O
  - Nem blokkoló hívások a komplex alkalmazások támogatásához
  - Nagy memóriaigény lehetséges
  - Tipikusan pufferelt adatátvitel a bájtos adatátvitel helyett
  - Header fájlok végződése nem “_l” (például xuartlite.h)

Szoftver platform projekt

- **Uartlite magas szintű meghajtó**
  - XStatus XUartLite_Initialize(XUartLite *InstancePtr, Xuint16 DeviceId)
  - void XUartLite_ResetFifos(XUartLite *InstancePtr)
  - unsigned int XUartLite_Send(XUartLite *InstancePtr, Xuint8 *DataBufferPtr, unsigned int NumBytes)
  - unsigned int XUartLite_Recv(XUartLite *InstancePtr, Xuint8 *DataBufferPtr, unsigned int NumBytes)
  - Xboolean XUartLite_IsSending(XUartLite *InstancePtr)
  - void XUartLite_GetStats(XUartLite *InstancePtr, XUartLite_Stats *StatsPtr)
  - void XUartLite_ClearStats(XUartLite *InstancePtr)
  - XStatus XUartLite_SelfTest(XUartLite *InstancePtr)
  - void XUartLite_EnableInterrupt(XUartLite *InstancePtr)
  - void XUartLite_DisableInterrupt(XUartLite *InstancePtr)
  - void XUartLite_SetRecvHandler(XUartLite *InstancePtr, XUartLite_Handler FuncPtr, void *CallBackRef)
  - void XUartLite_SetSendHandler(XUartLite *InstancePtr, XUartLite_Handler FuncPtr, void *CallBackRef)
  - void XUartLite_InterruptHandler(XUartLite *InstancePtr)
- **Uartlite alacsony szintű meghajtó**
  - void XUartLite_SendByte(Xuint32 BaseAddress, Xuint8 Data)
  - Xuint8 XUartLite_RecvByte(Xuint32 BaseAddress)

Szoftver platform projekt

Szoftver könyvtárak:

- **Matematikai könyvtár (*libm*)**
- **Standard C könyvtár (*libc*)**
  - A könyvtár függvényei automatikusan rendelkezésre állnak
- **Xilinx C nyelvű meghajtók és könyvtárak (*libxil*)**
  - Xilinx FAT fájlrendszer: ***Fatfs***
  - Xilinx memória fájlrendszer: ***Mfs***
  - Xilinx hálózati támogatás: ***lwIp***
  - Xilinx FLASH memória támogatás: ***Flash***
  - Xilinx In-system és soros FLASH rendszer: ***isf***

Szoftver platform projekt

xparameters.h header fájl:

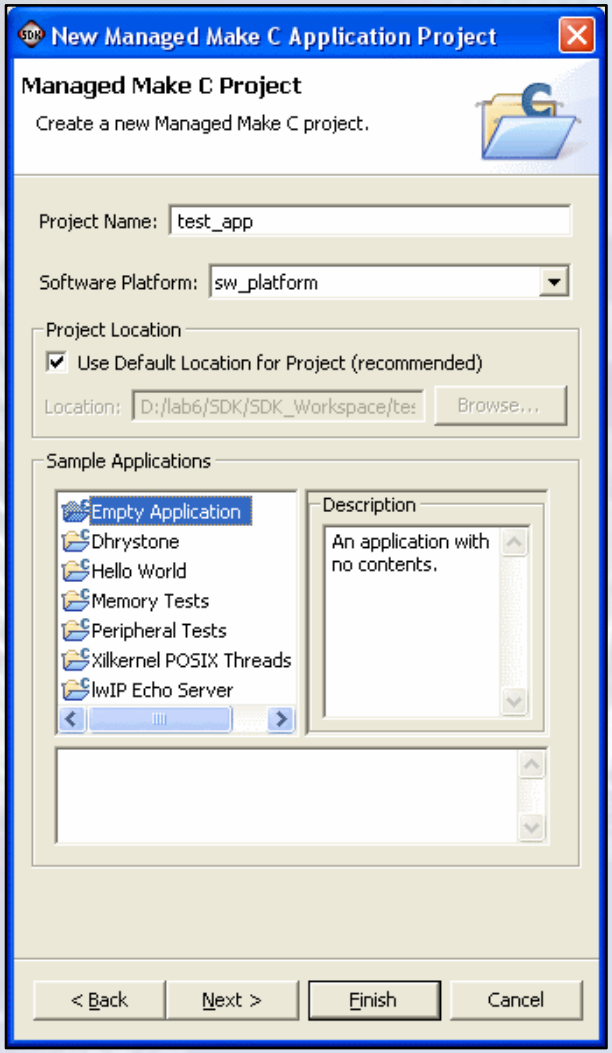
- A rendszerben lévő hardver egységek paramétereit tárolja
- Elnevezési konvenció: ***XPAR_[periféria_név]_[paraméter_név]***

```
/* Definitions for driver GPIO */
#define XPAR_XGPIO_NUM_INSTANCES 3

/* Definitions for peripheral LEDS_8BIT */
#define XPAR_LEDS_8BIT_BASEADDR 0x81400000
#define XPAR_LEDS_8BIT_HIGHADDR 0x8140FFFF
#define XPAR_LEDS_8BIT_DEVICE_ID 0
#define XPAR_LEDS_8BIT_INTERRUPT_PRESENT 0
#define XPAR_LEDS_8BIT_IS_DUAL 0

/* Definitions for peripheral DIP */
#define XPAR_DIP_BASEADDR 0x81420000
#define XPAR_DIP_HIGHADDR 0x8142FFFF
#define XPAR_DIP_DEVICE_ID 1
#define XPAR_DIP_INTERRUPT_PRESENT 0
#define XPAR_DIP_IS_DUAL 0
```

SDK projekt létrehozása

Szoftver alkalmazás projekt létrehozása:

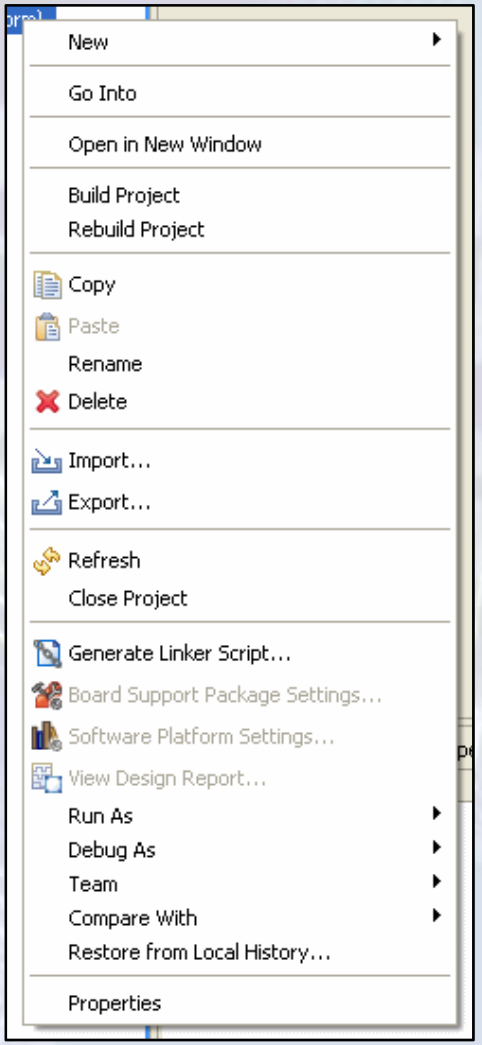
- **Managed Make C Application Project**
  - ***File*** menü → ***New***
  - ***Projekt nézet***: job klikk → ***New***
- **A megjelenő ablakban**
  - Nevezzük el a projektet
  - Válasszuk ki a minta alkalmazást
    - Üres alkalmazás
    - Memória teszt, periféria teszt, stb.
  - Az alapértelmezett beállítások megfelelőek
- **A projekt létrehozása: *Finish* gomb**
- **Egyéb beállítások: *Next* gomb (nem érdekesek)**
  - Konfigurációk kiválasztása
    - Debug, Release és Profile
    - A konfigurációkhoz eltérő beállítások rendelhetők
  - Egyéb projekt beállítások



SDK projekt létrehozása

Projekt nézet: jobb klikk → menü

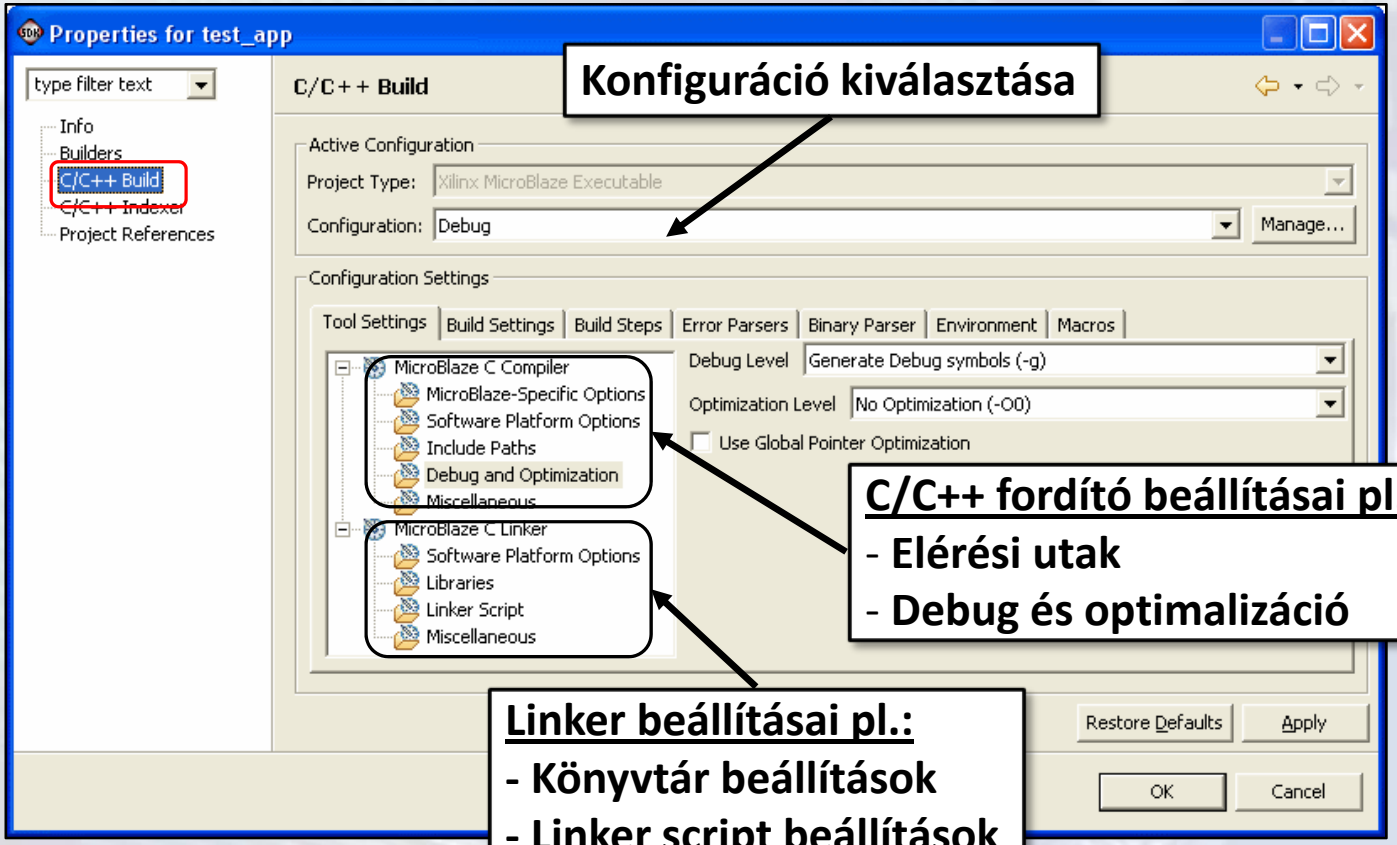
- **Új forrásfájl hozzáadása**
  - New → Source File / Header File
- **A projekt lefordítása**
  - Build Project / Rebuild Project
- **Másolás, beillesztés, törlés, átnevezés**
- **Importálás**
  - Tömörített fájlból, fájlrendszerből, stb.
- **Exportálás**
  - Tömörített fájlba, fájlrendszerbe, stb.
- **Linker script generálása**
- **Alkalmazás futtatása:** hardveren, szimulátoron
- **Debug:** hardveren, szinulátoron
- **A kiválasztott elem tulajdonságai**



SDK projekt beállítások

A C/C++ fordító és a linker beállításai:

- Projekt tulajdonságai → C/C++ Build



SDK projekt beállítások

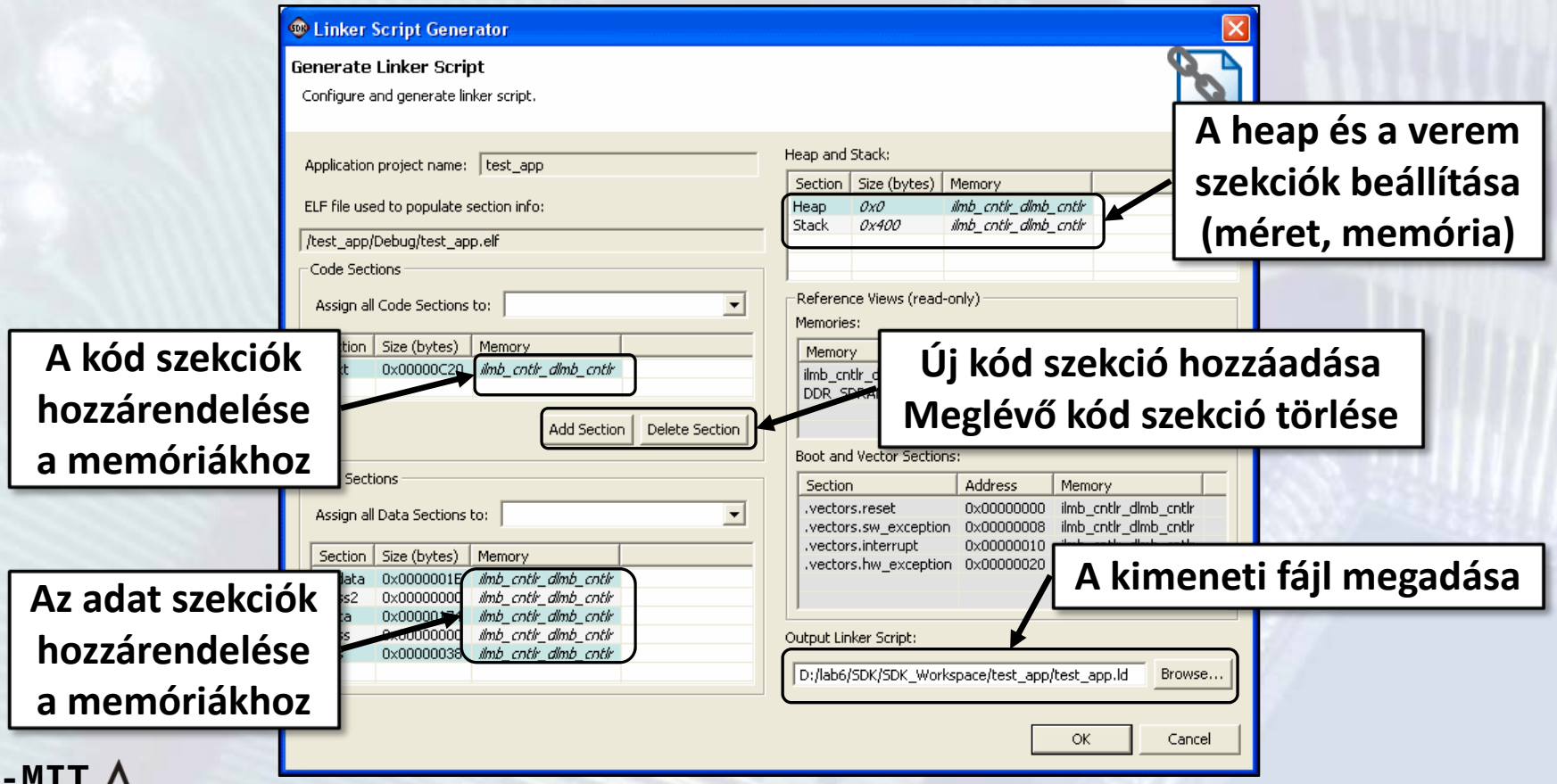
Linker script létrehozása:

- **A tárgykód és a végrehajtható fájl szekciókból áll**
  - ***.text***: végrehajtható kód
  - ***.rodata***: csak olvasható adatok
  - ***.sdata2***: kis méretű (max. 7 byte), csak olvasható adatok
  - ***.sbss2***: kis méretű, nem inicializált, csak olvasható adatok
  - ***.data***: írható/olvasható adatok
  - ***.sdata***: kis méretű, írható/olvasható adatok
  - ***.sbss***: kis méretű, nem inicializált adatok
  - ***.bss***: nem inicializált adatok
  - ***.heap***: szekció a dinamikus memória foglaláshoz
  - ***.stack***: verem szekció
- **A linker script (többek között) azt mondja meg, hogy hol helyezkedjenek el az egyes szekciók a memórián belül**

SDK projekt beállítások

Linker script létrehozása:

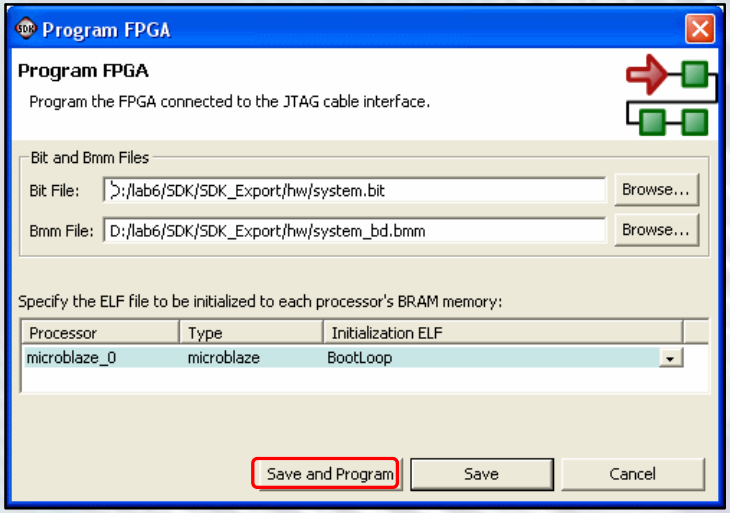
- Projekt tulajdonságai → Generate Linker Script…



Alkalmazás futtatása, debuggolása

Az FPGA konfigurálása, a lefordított alkalmazás futtatása:

- **Az FPGA konfigurálása:**
  - ***Tools*** menü → ***Program FPGA…*** vagy a  gomb
  - Kattintsunk a Sava and Program gombra



- **A lefordított szoftver alkalmazás letöltése és futtatása**
  - ***Run*** menü → ***Run…*** vagy a  gomb

Alkalmazás futtatása, debuggolása

A szoftver alkalmazás debuggolása:

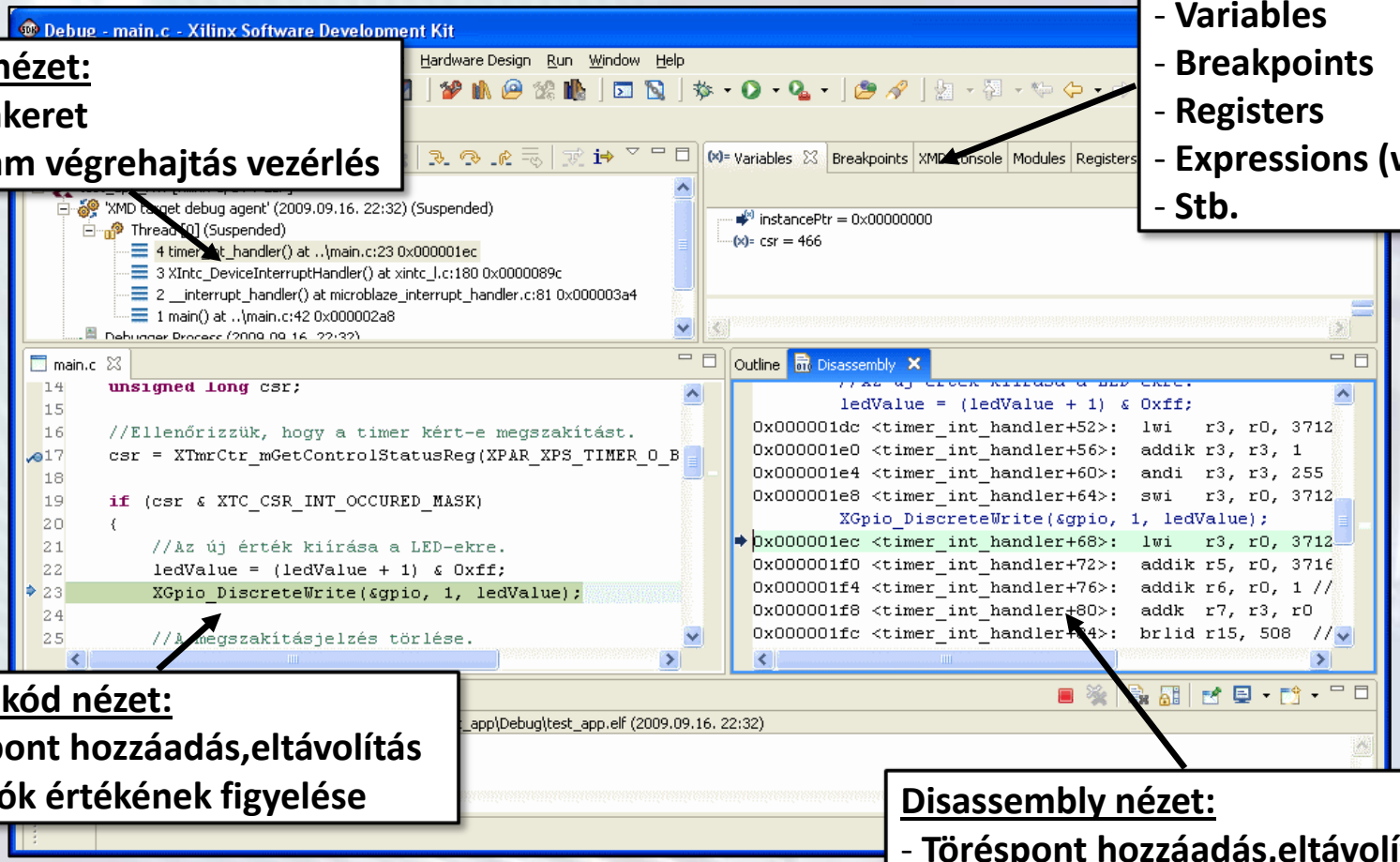
- **Az alkalmazás lefordítása: debug konfigurációval**
  - Az optimalizáció le van tiltva
  - A debug szimbólumok generálása engedélyezett
- **A debugger elindítása**
  - ***Run*** menü → ***Debug…*** vagy a  gomb
- **A debugger leállítása**
  - ***Run*** menü → ***Terminate*** vagy a  gomb

Alkalmazás futtatása, debuggolása

A debug perspektíva:

Debug nézet:
- Veremkeret
- Program végrehajtás vezérlés

Egyéb nézetek pl.:
- Variables
- Breakpoints
- Registers
- Expressions (watch)
- Stb.



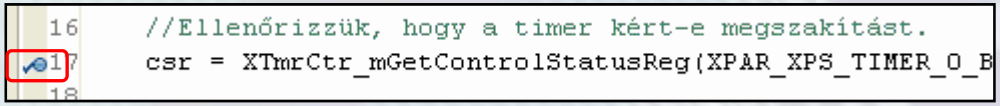
C forráskód nézet:
- Töréspont hozzáadás,eltávolítás
- Változók értékének figyelése

Disassembly nézet:
- Töréspont hozzáadás,eltávolítás

Alkalmazás futtatása, debuggolása

A debugger funkciói:

- **Töréspontok elhelyezése és eltávolítása:**
  - Dupla kattintás a sor száma mellett a szürke területen



- **A program végrehajtás vezérlése**
  - ***Resume***: a program futásának folytatása
  - ***Suspend***: a program futásának megállítása
  - ***Step Into***: az aktuális forráskód sor végrehajtása
    - Függvényhívás esetén belép a függvénybe
  - ***Step Over***: az aktuális forráskód sor végrehajtása
    - Függvényhívás esetén lefut a függvény, nem lép be a függvénybe
  - ***Step Return***: a futás leáll a függvényből való kilépéskor
  - ***Run to Line***: futtatás a kijelölt forráskód sorig

Alkalmazás futtatása, debuggolása

A debugger funkciói: nézetek

- **Forráskód nézet (C, disassembly)**
  - Töréspontok elhelyezése, eltávolítása
  - Változók értékeinek megtekintése
    - Vigyük az egérkurzort a változó fölé
- **Debug nézet**
  - Veremkeret
  - A program végrehajtás vezérlése
- **Variables: a lokális változók listája, értékeik módosítása**
- **Breakpoints: töréspontok engedélyezése, tiltása**
- **Registers: a CPU regiszterek listája, értékeik módosítása**
- **Expressions: kifejezések értékének figyelése (watch)**
- **Memory: memóriatartalom megjelenítése, módosítása**

Példák

Nagyon egyszerű szoftver alkalmazás (1):

- Két 8 bites GPIO periféria: LED-ek, kapcsolók
- A kapcsolók állapotát megjelenítjük a LED-eken
- A perifériákat közvetlenül kezeljük, nem használjuk az eszközmeghajtó programokat
- GPIO regiszterkészlet: a periféria adatlapjában

Register Name	Description	PLB Address	Access
GPIO_DATA	Channel 1 XPS GPIO Data Register	C_BASEADDR + 0x00	Read/Write
GPIO_TRI	Channel 1 XPS GPIO 3-state Register	C_BASEADDR + 0x04	Read/Write
GPIO2_DATA	Channel 2 XPS GPIO Data register	C_BASEADDR + 0x08	Read/Write
GPIO2_TRI	Channel 2 XPS GPIO 3-state Register	C_BASEADDR + 0x0C	Read/Write

Register Name	Description	PLB Address	Access
GIER	Global Interrupt Enable Register	C_BASEADDR + 0x11C	Read/Write
IP IER	IP Interrupt Enable Register	C_BASEADDR + 0x128	Read/Write
IP ISR	IP Interrupt Status Register	C_BASEADDR + 0x120	Read/TOW[1]

Példák

Nagyon egyszerű szoftver alkalmazás (1):

- **I/O makrók: az *xio.h* fájlban vannak definiálva**
- **Memória írás**
  - 8 bit: ***XIo_Out8(OutputPtr, Value)***
  - 16 bit: ***XIo_Out16(OutputPtr, Value)***
  - 32 bit: ***XIo_Out32(OutputPtr, Value)***
- **Memória olvasás**
  - 8 bit: ***XIo_In8(InputPtr)***
  - 16 bit: ***XIo_In16(InputPtr)***
  - 32 bit: ***XIo_In32(InputPtr)***

Példák

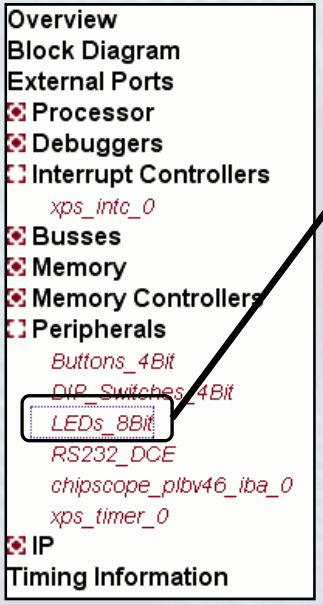
Nagyon egyszerű szoftver alkalmazás (1):

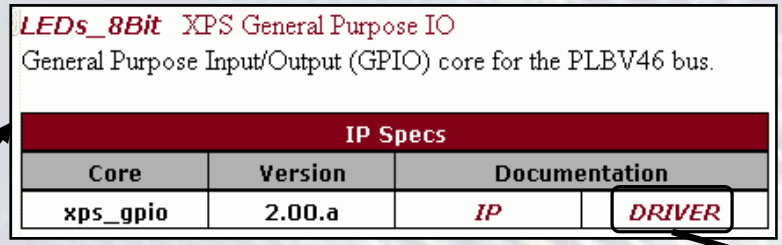
```
#include <xio.h>
#include <xparameters.h>

int main()
{
    unsigned long data;

    //A GPIO portok irányának beállítása.
    XIo_Out32(XPAR_LEDS_BASEADDR + 0x04, 0x00);
    XIo_Out32(XPAR_SWITCHES_BASEADDR + 0x04, 0xff);

    //Végtelen ciklus.
    while (1)
    {
        data = XIo_In32(XPAR_SWITCHES_BASEADDR);
        XIo_Out32(XPAR_LEDS_BASEADDR, data);
    }

    return 0;
}
```

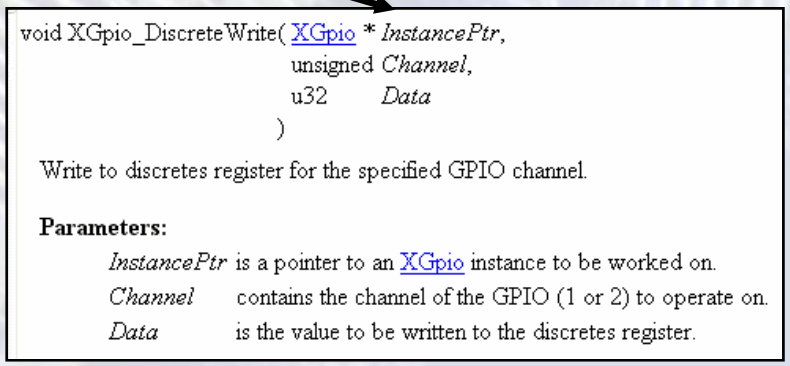
Példák

Nagyon egyszerű szoftver alkalmazás (2):

- Az előző példa alacsony szintű meghajtó használatával
- Eszközpéldány azonosítása: báziscímek alapján
- Eszközmeghajtó dokumentációja:
  - ***Hardware Design*** menü → ***View Design Report…***







Példák

Nagyon egyszerű szoftver alkalmazás (2):

```
#include <xgpio_l.h>
#include <xparameters.h>

int main()
{
    unsigned long data;

    //A GPIO portok irányának beállítása.
    XGpio_mWriteReg(XPAR_LEDS_BASEADDR, XGPIO_TRI_OFFSET, 0x00);
    XGpio_mWriteReg(XPAR_SWITCHES_BASEADDR, XGPIO_TRI_OFFSET, 0xff);

    //Végtelen ciklus.
    while (1)
    {
        data = XGpio_mGetDataReg(XPAR_SWITCHES_BASEADDR, 1);
        XGpio_mSetDataReg(XPAR_LEDS_BASEADDR, 1, data);
    }

    return 0;
}
```

Példák

Nagyon egyszerű szoftver alkalmazás (3):

- Az első példa magas szintű meghajtó használatával
- Eszközpéldány azonosítása: eszköz azonosító alapján
- Hasonlítsuk össze a három példa esetén a kód méreteket

```
#include <xgpio.h>
#include <xparameters.h>

//A GPIO perifériák leírói.
XGpio leds;
XGpio switches;

int main()
{
    unsigned long data;

    //A GPIO leírók inicializálása.
    XGpio_Initialize(&leds, XPAR_LEDS_DEVICE_ID);
    XGpio_Initialize(&switches, XPAR_SWITCHES_DEVICE_ID);
```

Példák

Nagyon egyszerű szoftver alkalmazás (3):

```
    //A GPIO portok irányának beállítása.
    XGpio_SetDataDirection(&leds, 1, 0x00);
    XGpio_SetDataDirection(&switches, 1, 0xff);

    //Végtelen ciklus.
    while (1)
    {
        data = XGpio_DiscreteRead(&switches, 1);
        XGpio_DiscreteWrite(&leds, 1, data);
    }

    return 0;
}
```

Példák

Megszakítások kezelése:

- **Példa: időzítő megszakítások kezelése**
  - Másodperc számláló értékének megjelenítése a LED-eken
- **Lépések**
  - A megszakításkezelő rutin regisztrálása
    - A perifériához tartozik egy megszakítás azonosító érték
  - A megszakítás vezérlő konfigurálása
    - Globális megszakítás engedélyezés
    - A megfelelő megszakítás bemenet engedélyezése
  - Megszakítások engedélyezése a MicroBlaze processzoron
  - Az időzítő konfigurálása
    - A periódusregiszter beállítása
    - A megszakítások engedélyezése
- **A megszakításkezelő rutinban használt globális változók**
  - ***volatile*** → optimalizálás letiltása a változóra

Példák

Megszakítások kezelése:

```
#include <xtmrctr.h>
#include <xintc_l.h>
#include <xgpio.h>
#include <xparameters.h>

//Globális változók.
XGpio leds;
volatile unsigned long ledValue;

//Megszakításkezelő rutin.
void timer_int_handler(void *instancePtr)
{
    unsigned long csr;

    //Az új érték kiírása a LED-ekre.
    ledValue = (ledValue + 1) & 0xff;
    XGpio_DiscreteWrite(&leds, 1, ledValue);

    //A megszakítás jelzés törlése.
    csr = XTmrCtr_mGetControlStatusReg(XPAR_XPS_TIMER_0_BASEADDR, 0);
    XTmrCtr_mSetControlStatusReg(XPAR_XPS_TIMER_0_BASEADDR, 0, csr);
}
```

Példák

Megszakítások kezelése:

```
int main()
{
    //A LED GPIO inicializálása.
    ledValue = 0;
    XGpio_Initialize(&leds, XPAR_LEDS_8BIT_DEVICE_ID);
    XGpio_SetDataDirection(&leds, 1, 0x00);
    XGpio_DiscreteWrite(&leds, 1, ledValue);

    //A megszakításkezelő rutin beállítása.
    XIntc_RegisterHandler(
        XPAR_XPS_INTC_BASEADDR,                  //INTC báziscíme
        XPAR_XPS_INTC_XPS_TIMER_INTERRUPT_INTR,  //Megszakítás azonosító
        (XInterruptHandler) timer_int_handler,   //Megszakításkezelő rutin
        NULL                                     //Megsz. kezelő rutin paramétere
    );

    //A megszakítás vezérlő konfigurálása.
    XIntc_mMasterEnable(XPAR_XPS_INTC_BASEADDR);
    XIntc_mEnableIntr(XPAR_XPS_INTC_BASEADDR, XPAR_XPS_TIMER_INTERRUPT_MASK);

    //A megszakítások engedélyezése a processzoron.
    microblaze_enable_interrupts();
```

Példák

Megszakítások kezelése:

```
    //A timer LOAD regiszterének beállítása (megszakítás másodpercenként).
    XTmrCtr_mSetLoadReg(XPAR_XPS_TIMER_BASEADDR, 0, XPAR_PROC_BUS_0_FREQ_HZ);
    //A timer alapállapotba állítása.
    XTmrCtr_mSetControlStatusReg(
        XPAR_XPS_TIMER_BASEADDR,
        0,
        XTC_CSR_INT_OCCURED_MASK | XTC_CSR_LOAD_MASK
    );
    //A timer elindítása.
    XTmrCtr_mSetControlStatusReg(
        XPAR_XPS_TIMER_BASEADDR,
        0,
        XTC_CSR_ENABLE_TMR_MASK | XTC_CSR_ENABLE_INT_MASK |
        XTC_CSR_AUTO_RELOAD_MASK | XTC_CSR_DOWN_COUNT_MASK
    );

    //Végtelen ciklus.
    while (1);

    return 0;
}
```