



Universidad
Carlos III de Madrid

Instrumentación Electrónica con Microprocesador II: Procesadores Avanzados

**Microprocesadores empotrados en FPGAs
MicroBlaze™. Programación en C**

Marta Portela García



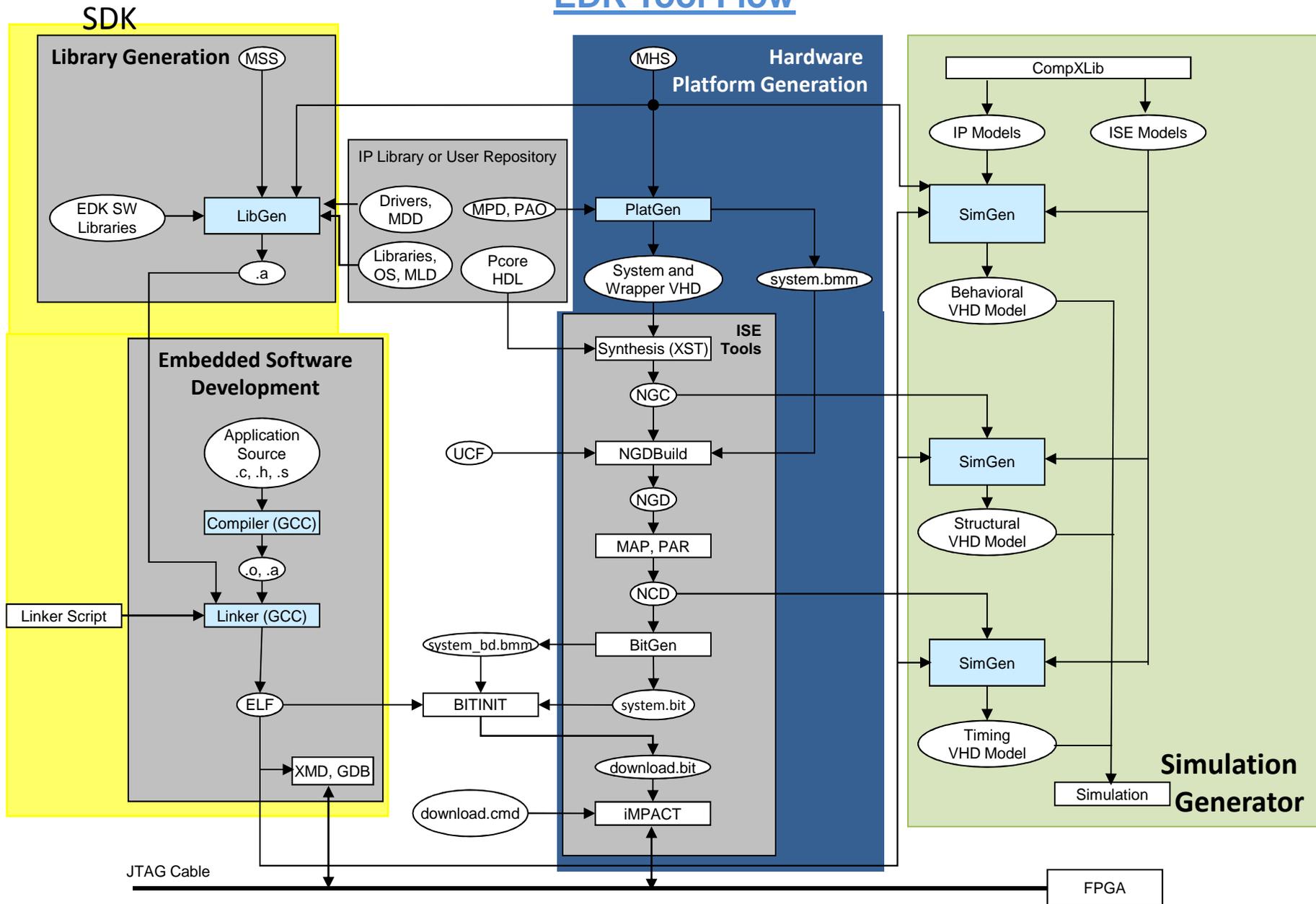


SDK (SOFTWARE DEVELOPMENT ENVIRONMENT)

- SDK facilita el desarrollo de las aplicaciones software para los sistemas empotrados con FPGAs de Xilinx.
 - Para desarrollar el software necesario para programar el sistema hardware desarrollado con XPS (los periféricos y el procesador)
- Basado en Eclipse: entorno de desarrollo de código abierto.
- Incluye las siguientes características:
 - Editor de código enriquecido para C/C++ y un entorno para su compilación.
 - Gestión de proyectos.
 - Generación automática de ficheros *makefile* y configuración para la construcción de aplicaciones.
 - Entorno para la depuración y el análisis de código.
 - Control de versiones.



EDK Tool Flow



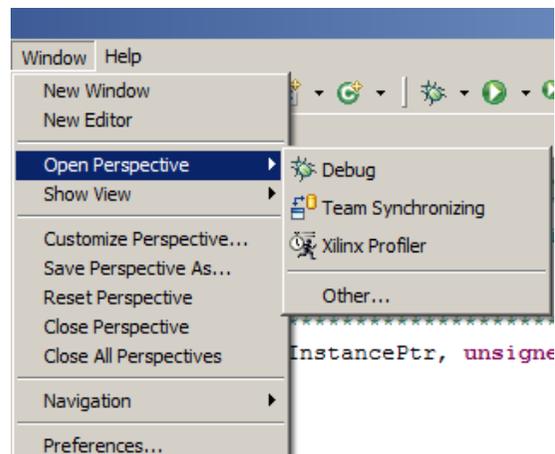
SDK: ELEMENTOS

- Plataforma Software
 - Colección de librerías y drivers que forman la capa más baja del software
 - 2 tipos:
 - Standalone
 - Xilkernel.- Nucleo ligero que proporciona servicios como gestión de hilos, planificación de tareas,...
 - Un mismo proyecto SDK puede contener varias plataformas SW
- Proyecto Software
 - Aplicación de usuario. Una plataforma SW puede contener varios proyectos SW. Contiene el código fuente (ficheros .c, .h)

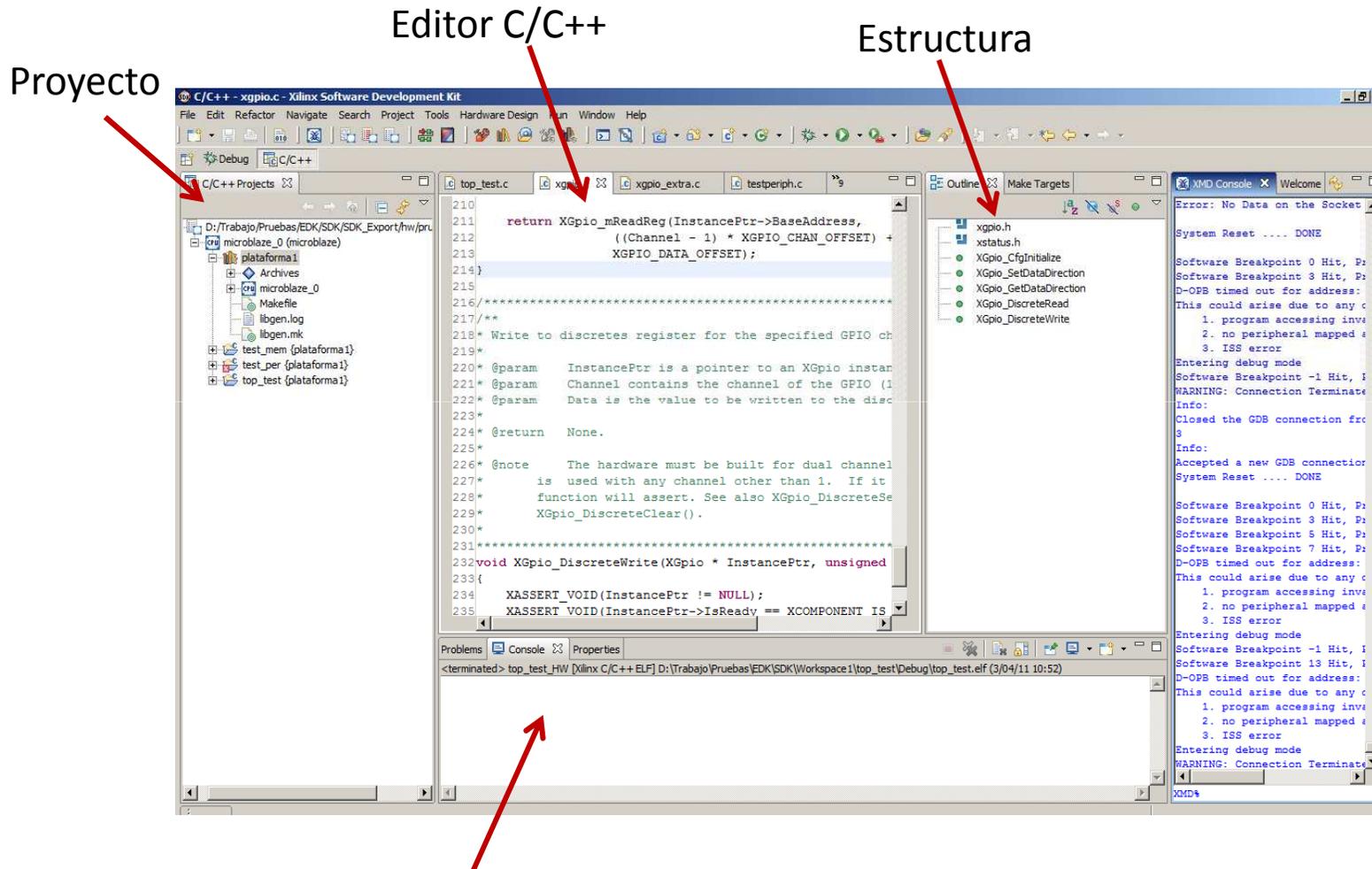


SDK: ENTORNO

- Espacio de trabajo (workspace)
 - Directorio utilizado para guardar toda la información relacionada con un proyecto en SDK
 - Es necesario un workspace por cada diseño hardware
 - Para copiar o mover workspaces de un directorio a otro hay que usar las funciones de importar/esportar de SDK.
- Perspectivas
 - Conjunto de herramientas de desarrollo relacionada con una tarea concreta dentro del desarrollo del SW.
 - Cada perspectiva está formada por un conjunto distinto de ventanas. Cada ventana en una perspectiva se denomina vista.



SDK: ENTORNO. PERSPECTIVA C/C++



Salida de la ejecución, propiedades y problemas encontrados

SDK: ENTORNO. PERSPECTIVA DE DEPURACIÓN

Módulos a
depurar

Información del uP:
variables, registros,...

Consola
XMD

Contenido de la
memoria

The screenshot shows the Xilinx IDE interface during a debug session. The main window displays the source code for `testperiph.c`. The console window at the bottom shows the output of the program. The memory viewer on the right displays the memory content at address `0x00000000`.

Address	0	1	2	3
00000000	B8080050			
00000010	B8080BE8			
00000020	B8080BE4			
00000030	00000000			
00000040	00000000			
00000050	31A00000			
00000060	80000000			
00000070	E06024C0			
00000080	B8000040			
00000090	E860234C			
000000A0	B0000000			
000000B0	99FC1800			
000000C0	E9E10000			
000000D0	30600000			
000000E0	30C024C4			
000000F0	E86024BC			
00000100	30A024BC			
00000110	E9E10000			
00000120	F9E10000			
00000130	BC720014			
00000140	BC92FFF4			
00000150	BC720014			
00000160	BC92FFF4			
00000170	80000000			
00000180	20A00000			
00000190	B9F40A44			
000001A0	B60F0008			
000001B0	FA61001C			
000001C0	E9E10000			
000001D0	B60F0008			

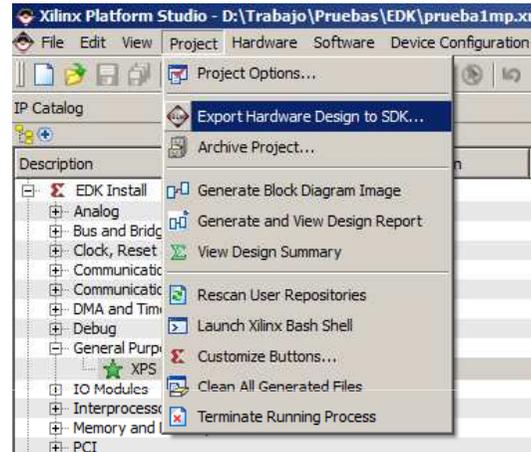


SDK: FLUJO DE DESARROLLO DE UNA APLICACIÓN SW

- Diseño del sistema HW con XPS.
- Exportar el diseño HW a SDK (formato XML)
- Abrir SDK y especificar el fichero con la descripción del sistema HW que se exportó desde XPS.
- Crear la plataforma SW que contenga una biblioteca de funciones a usar por la aplicación.
- Desarrollar la aplicación SW (incluyendo los drivers).
- Se puede modificar el mapa de memoria de la aplicación utilizando las herramientas de generación de scripts para el lincador (*linker script*).
- Desde SDK se puede descargar y ejecutar el software en la FPGA, así como depurar y analizar su rendimiento (profiling).

SDK: CREAR UN PROYECTO

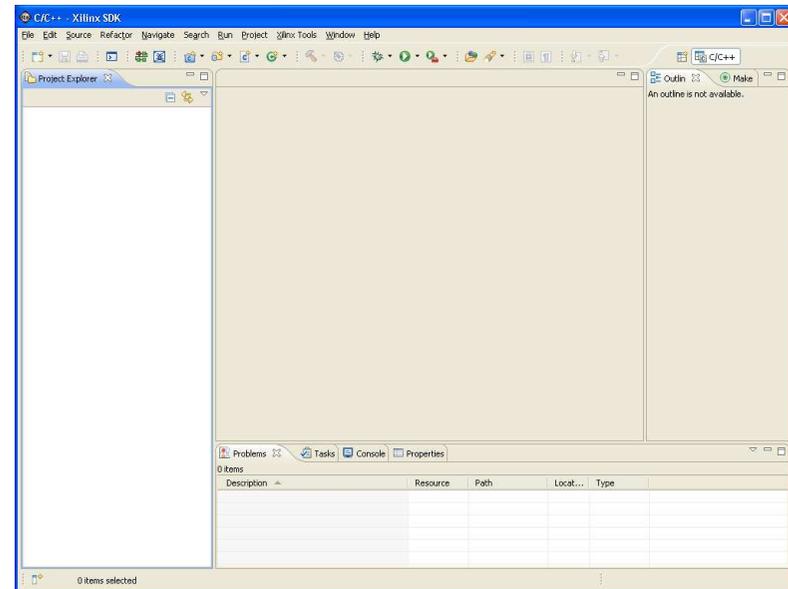
1 Exportar la plataforma HW desde XPS a SDK



2 Seleccionar la plataforma HW

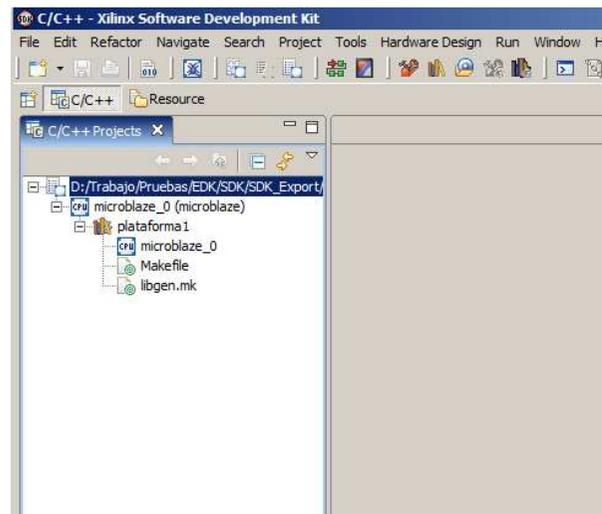
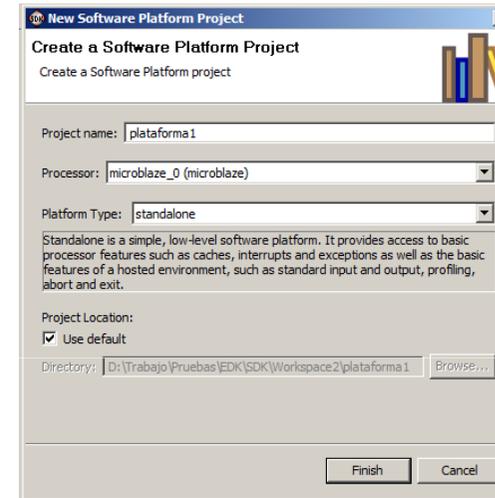
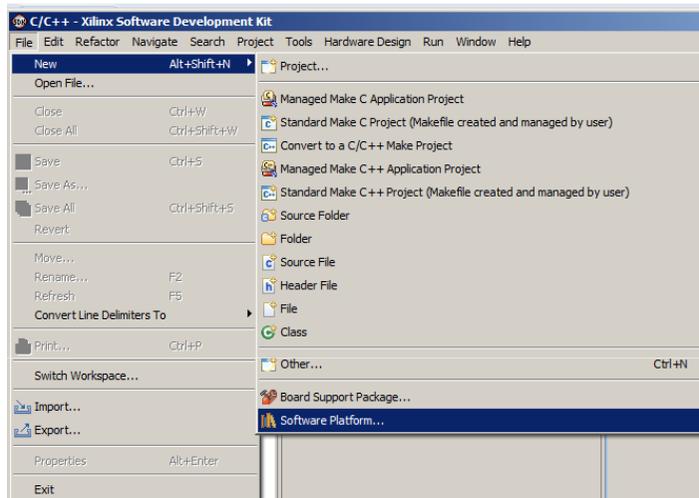
- Al abrir SDK lo primero que hay que hacer es indicar el Workspace
- A continuación es necesario seleccionar la plataforma HW

(../SDK_Export/HW/nameHW.xml)



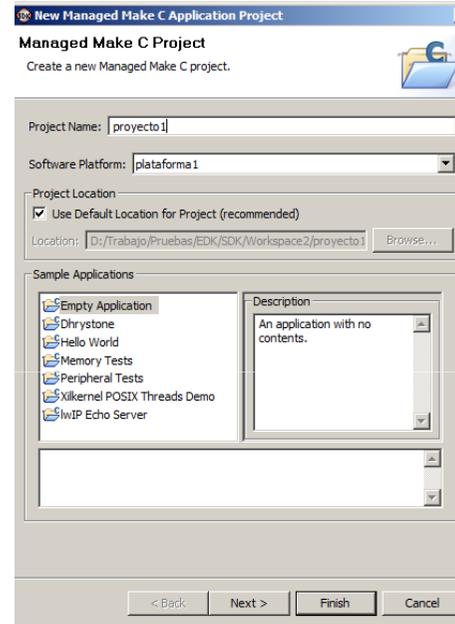
SDK: CREAR UN PROYECTO

3 Crear la plataforma SW



SDK: CREAR UN PROYECTO

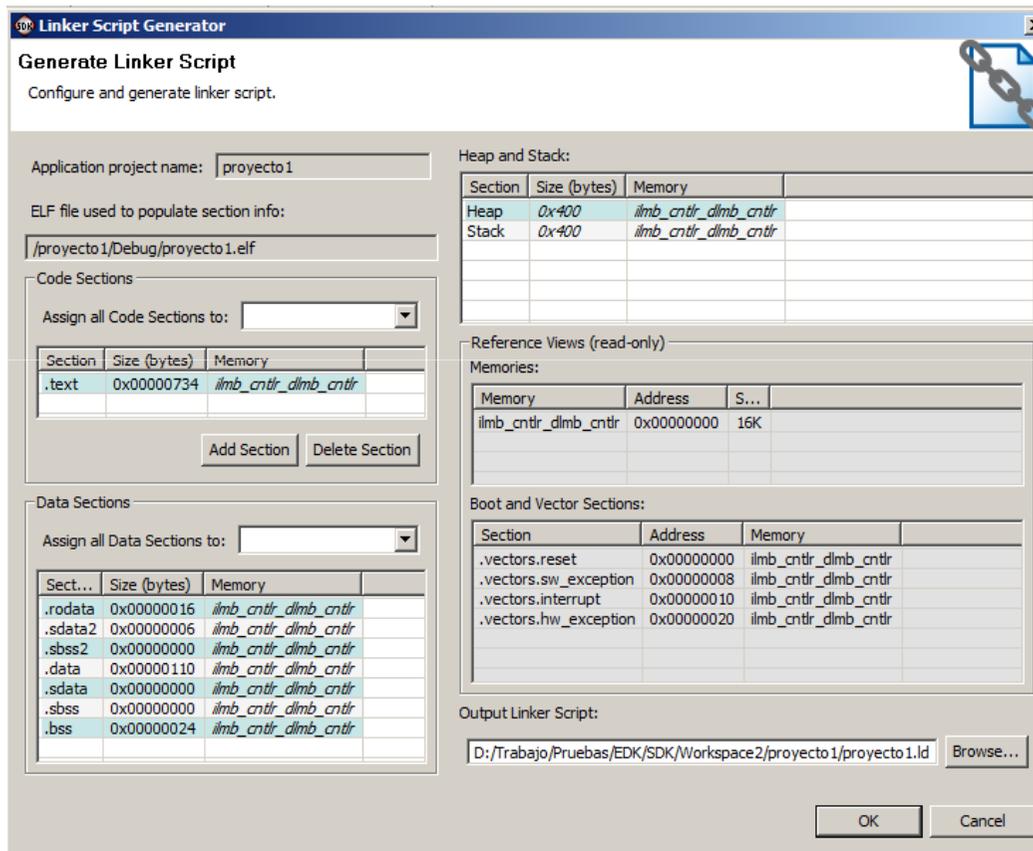
4 Crear el proyecto SW



- Importar un fichero: File → Import
 - Los ficheros importados se copian en el directorio de la aplicación
 - Se añaden automáticamente al fichero makefile
- Cada aplicación genera su propio fichero .elf

SDK: CREAR UN PROYECTO

5 Modificar o configurar el mapa de memoria: *Linker script*



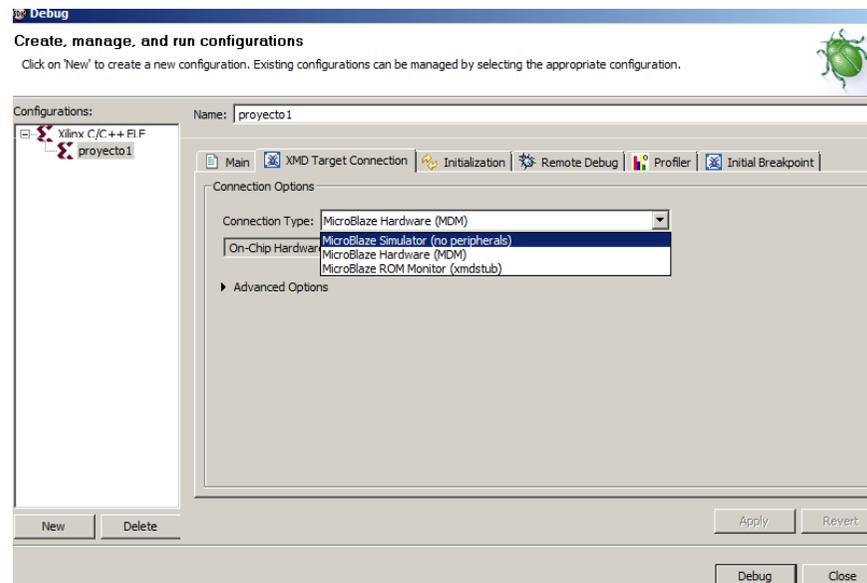
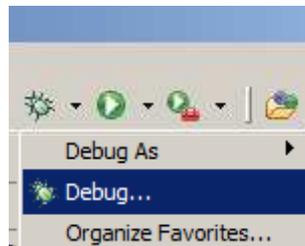
- Tools → Generate Linker script
- Fichero nombre_proyecto.ld
- Especifica dónde se almacenan código y datos dentro del sistema HW especificado.

SDK: CREAR UN PROYECTO

6 Pruebas: Simulación y en HW

XMD

- Es necesario abrir una consola de la herramienta XMD
- Simulación

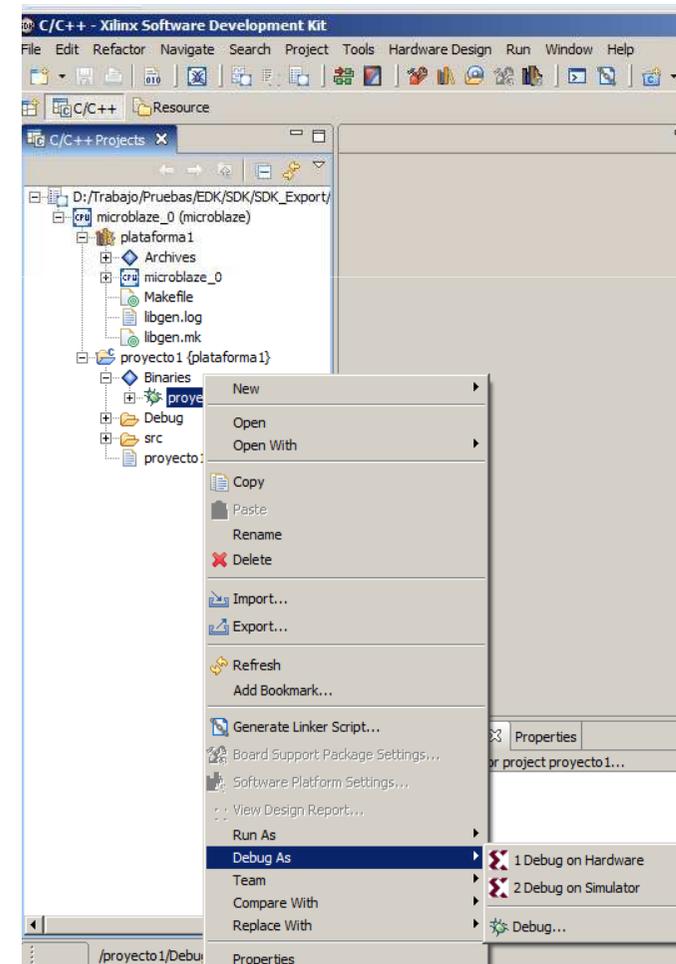
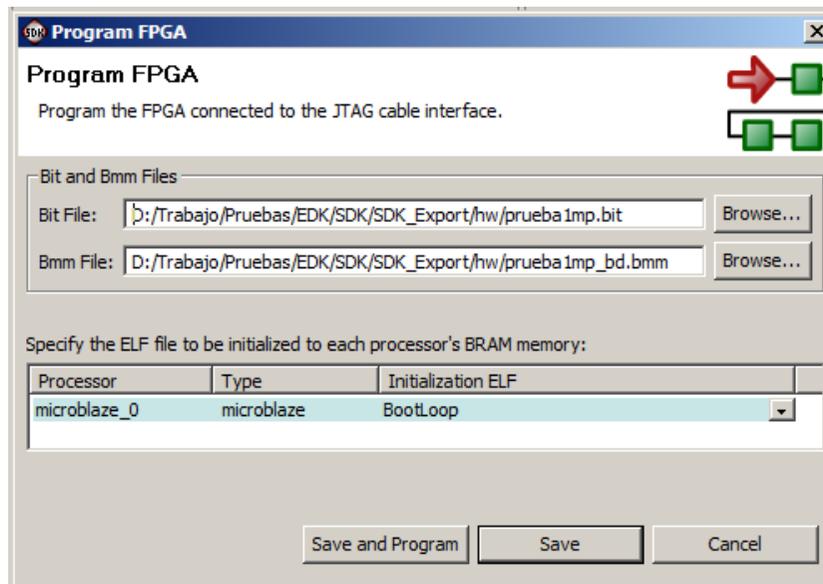


SDK: CREAR UN PROYECTO

6 Pruebas: Simulación y en HW

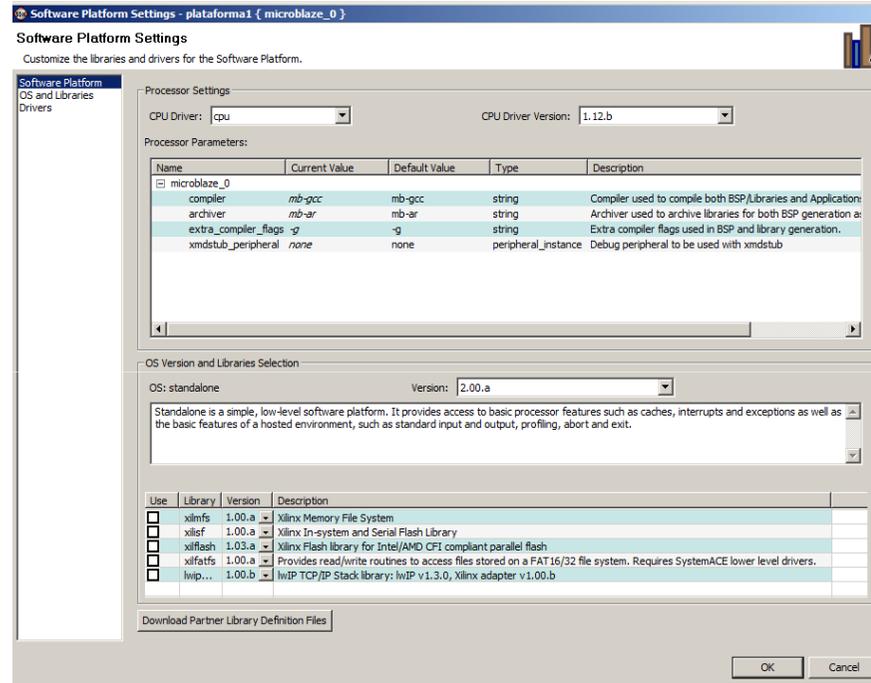
XMD

- Es necesario abrir una consola de la herramienta XMD
- Depuración Hardware
 - Tools → Program FPGA



SDK: PROPIEDADES DE UN PROYECTO

- Propiedades de la plataforma SW (Tools → SW Platform Settings)



- Propiedades de cada proyecto SW
 - Compilación, depuración, linkado...



BIBLIOGRAFÍA:

- “EDK Concepts, Tools, and Techniques. *Guide to Effective Embedded System Design*”
Xilinx → Manual

DRIVERS

- Xilinx proporciona drivers para los periféricos de la plataforma HW.
 - Código fuente
 - Documentación HTML
 - Ejemplos de cómo se pueden usar

The screenshot displays the Xilinx ISE environment. On the left, the IP Catalog shows the 'XPS General Purpose IO' selected. The central 'Bus Interfaces' table lists various components like 'fpga_0_RS232_UART_RX_PIN' and 'microblaze_0'. A context menu is open over the 'uartlite_v1_14_a' component, with 'Driver: uartlite_v1_14_a' selected. On the right, a 'Select one or more files to open' dialog shows the file structure of the driver, including folders like 'code', 'include', 'lib', and 'src', and files like 'uartlite.c' and 'uartlite_g.c'. The console at the bottom shows successful programming output.

Name	Net	Direction	Range	Class
fpga_0_RS232_UART_RX_PIN	fpga_0_RS23...	I		NONE
fpga_0_RS232_UART_TX_PIN	fpga_0_RS23...	O		NONE
fpga_0_LEDS_8Bit_GPIO_IO...	fpga_0_LEDS...	O	[0:7]	NONE
fpga_0_DIP_Switches_4Bit_GPI...	fpga_0_DIP...	I	[0:3]	NONE
fpga_0_Buttons_4Bit_GPIO_IO...	fpga_0_Butto...	I	[0:3]	NONE
fpga_0_clk_1_sys_clk_pin	ldm_clk_s	I		CLK
fpga_0_rst_1_sys_rst_pin	sys_rst_s	I		RST



LIBRERIAS

- Las librerías se configuran automáticamente con la herramienta Libgen al crear una plataforma HW.
- Xilinx proporciona tres librerías (`.\microblaze_0\lib\`)
 - **Libm**: para funciones matemáticas
 - **Libc**: para funciones estándar de C (stdio, stdlib, string) compiladas para MicroBlaze.

```
_ansi.h      fastmath.h  machine/    reent.h     stdlib.h    utime.h
_syslist.h   fcntl.h    malloc.h    regdef.h    string.h    utmp.h
ar.h         float.h    math.h      setjmp.h    sys/
assert.h     grp.h      paths.h     signal.h    termios.h
ctype.h      ieee754.h  process.h   stdarg.h    time.h
dirent.h     limits.h   pthread.h   stddef.h    unctrl.h
errno.h      locale.h   pwd.h       stdio.h     unistd.h
```

- **Libxil**: contiene drivers y librerías de Xilinx
 - LibXil Driver.- drivers de los dispositivos de Xilinx
 - LibXil MFS.- sistema de ficheros de memoria de Xilinx
 - LibXil flash/isf.- librería de programación de memoria flash paralela/serie



LIBRERIAS

- Además de las función estándar de C, Xilinx ofrece las siguientes funciones para entrada/salida:
 - void **print** (char *)
 - void **putnum** (int)
 - void **xil_printf** (const *char ctrl1,...)
- Las cabeceras de las funciones de xilinx se guardan en `.\microblaze_0\include\`

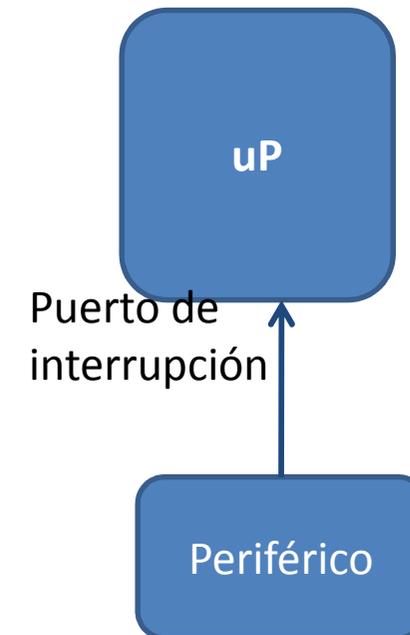
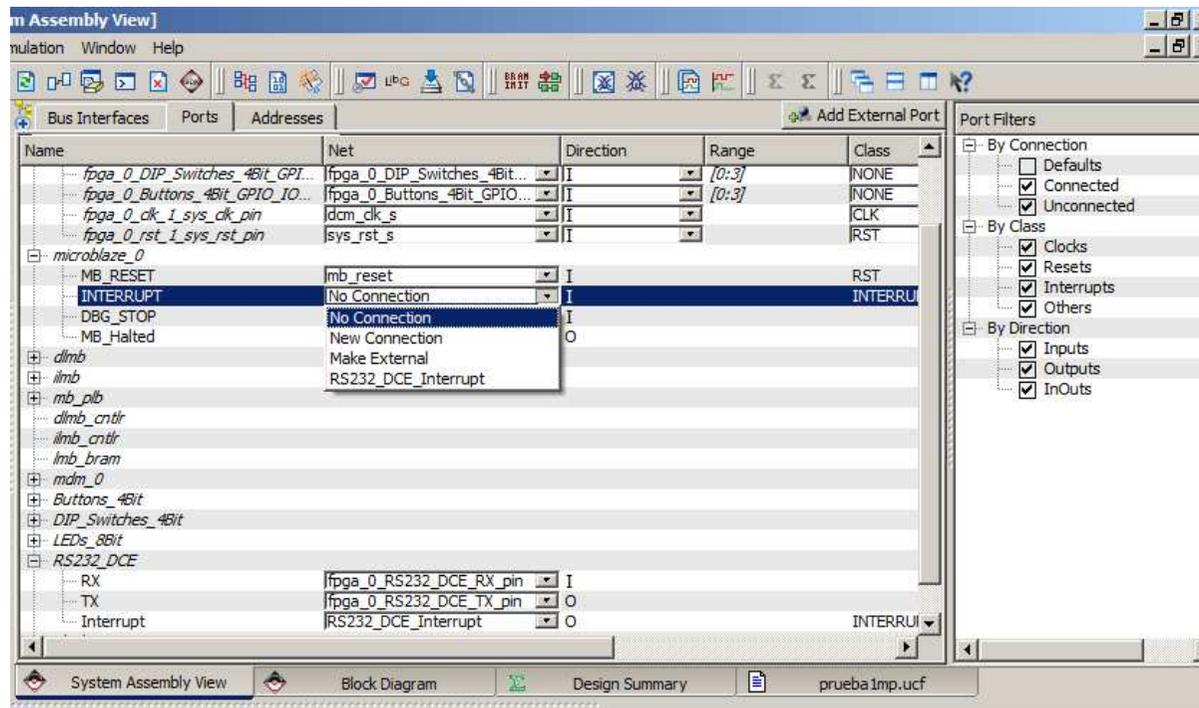


INTERRUPCIONES

- Sólo hay una señal de interrupción
- Se pueden gestionar distintas fuentes de interrupción:
 - Or de las distintas señales de interrupción
 - Utilizar un controlador de interrupciones (periférico)
- mb_interface.h
 - void microblaze_enable_interrupts(void)
 - void microblaze_disable_interrupts(void)
 - void microblaze_register_handler(XInterruptHandler *Handler*, void **DataPtr*)
- Para incluir interrupción en un sistema hay que seguir los siguientes pasos:
 - Realizar las conexiones HW necesarias
 - Desarrollar las funciones HW de atención a la interrupción

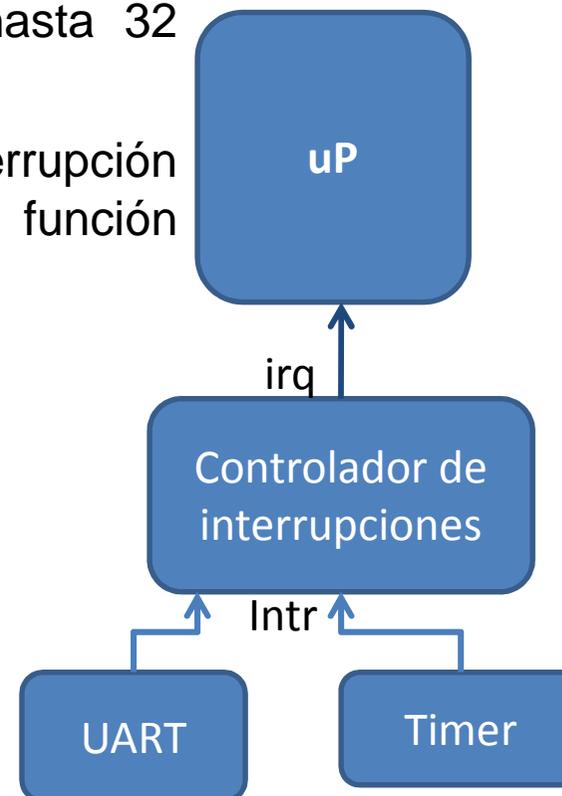
GESTIÓN DE INTERRUPTOS: SIN CONTROLADOR

- El controlador de interrupciones no es necesario cuando sólo hay un dispositivo con interrupción
- La señal de interrupción del periférico debe conectarse a la entrada de interrupción del microprocesador.



GESTIÓN DE INTERRUPTOS: CON CONTROLADOR

- El controlador de interrupciones sí es necesario cuando hay más de un dispositivo capaz de producir una interrupción
- La señal de interrupción del periférico debe conectarse a la entrada del controlador de interrupciones
- El MicroBlaze puede gestionar hasta 32 fuentes de interrupción
- Las funciones de atención a la interrupción se escriben en C igual que otra función cualquiera:
 - `void nombre(void *)`





GESTIÓN DE INTERRUPTIONES: CON CONTROLADOR

The screenshot displays the Xilinx Platform Studio interface. The IP Catalog on the left lists various IP blocks, including the XPS Interrupt Controller (xps_intc). The Bus Interfaces window shows a table of connections:

Name	Net	Direction	Range	Class
INTERRUPT	xps_intc_0_Irq	I		INTERRUI
DBG_STOP	No Connection	I		
MB_Halted	No Connection	O		
... (other bus interfaces) ...				
xps_intc_0				
Intr	L to H: RS232_DCE_Interrupt&...	I		[[C_NUM_INTR_I... INTERRUI
Irq	xps_intc_0_Irq	O		INTERRUI

The Interrupt Connection Dialog is open, showing a list of potential connections on the left and connected interrupts on the right:

- Potential Interrupt Connection(s): mb_plb:Bus_Error_Det, mdm_0:Interrupt
- Connected Interrupt(s): RS232_DCE_Interrupt, xps_timer_0_Interrupt

The Console window at the bottom shows the following output:

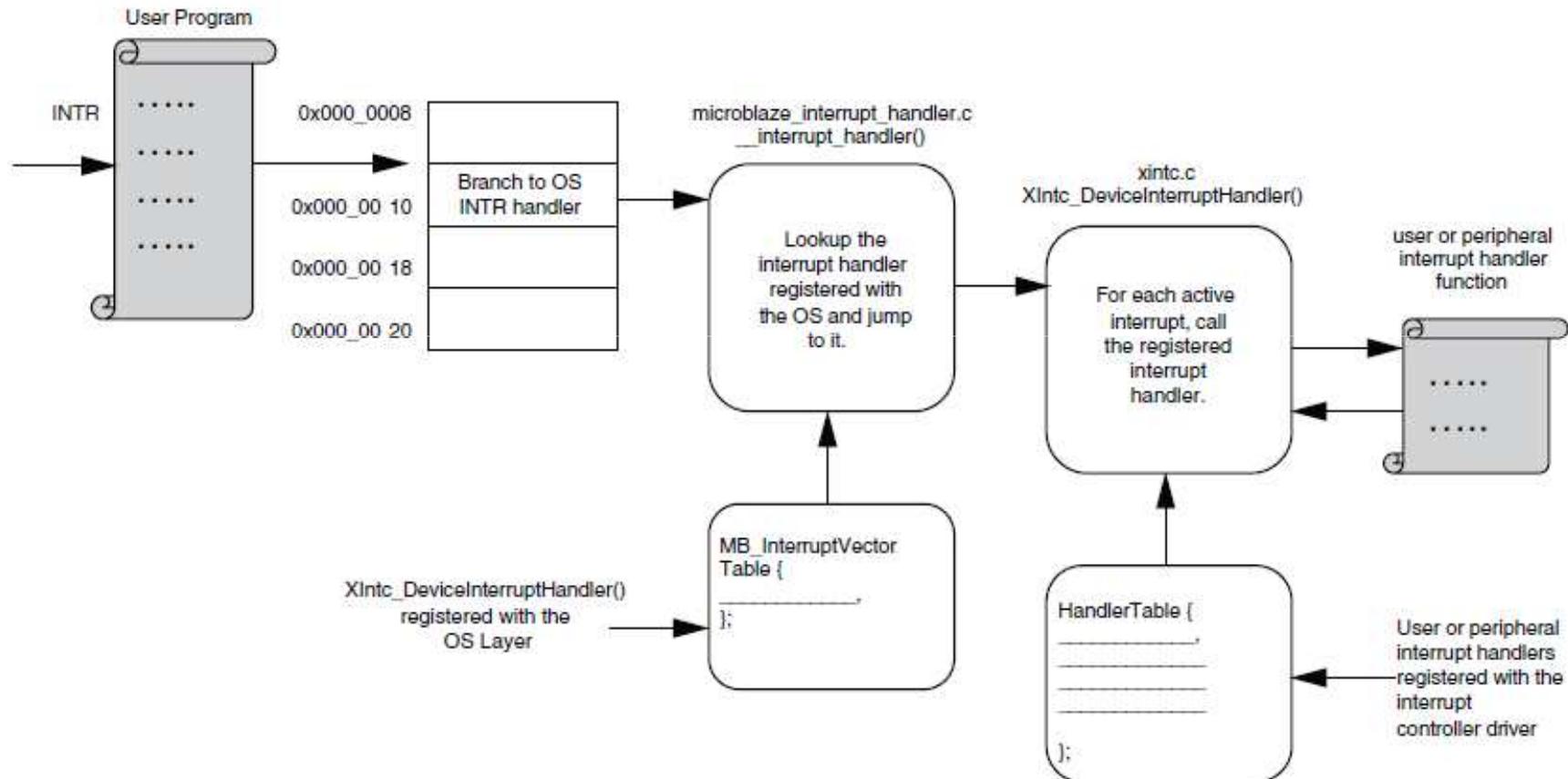
```
Done!  
Assigned Driver tmrctr 1.11.a for instance xps_timer_0  
WARNING:EDK:2137 - Peripheral xps_timer_0 is not accessible from any  
xps_timer_0 has been added to the project  
WARNING:EDK:2137 - Peripheral xps_timer_0 is not accessible from any  
Assigned Driver intc 1.11.a for instance xps_intc_0
```



GESTIÓN DE INTERRUPCIONES: CON CONTROLADOR

- Int Xintc_Initialize (Xintc **InstancePtr*, u16 *DeviceId*)
- Int Xintc_Connect(Xintc **InstancePtr*, u8 *Id*, XInterruptHandler *Handler*, void **CallbackRef*)
- Int Xintc_Disconnect(Xintc **InstancePtr*, u8 *Id*)
- Void Xintc_Enable (Xintc **InstancePtr*, u8 *Id*)
- Void Xintc_Disable (Xintc **InstancePtr*, u8 *Id*)
- Int Xintc_Start (Xintc **InstancePtr*, u8 *Mode*)
- Int Xintc_Stop(Xintc **InstancePtr*)

GESTIÓN DE INTERRUPCIONES : FLUJO DE ATENCIÓN A LA INTERRUPCIÓN (CON CONTROLADOR)



"Embedded System Tools Reference Guide", Appendix B Interrupt Management. Xilinx

GESTIÓN DE INTERRUPTACIONES : CÓDIGO

```
71 int main() {
72
73     int count_mod_3;
74
75     // Enable MicroBlaze Interrupts
76     microblaze_enable_interrupts();
77
78     /* Register the Timer interrupt handler in the vector table */
79     XIntc_RegisterHandler(XPAR_XPS_INTC_O_BASEADDR,
80                          XPAR_XPS_INTC_O_DELAY_INTERRUPT_INTR,
81                          (XInterruptHandler) timer_int_handler,
82                          (void *)XPAR_DELAY_BASEADDR);
83
84     /* Initialize and set the direction of the GPIO connected to LEDs */
85     XGpio_Initialize(&gpio, XPAR_LEDS_8BIT_DEVICE_ID);
86     XGpio_SetDataDirection(&gpio, LEDChan, 0);
87 }
```

"Embedded System Tools Reference Guide", Appendix B Interrupt Management. Xilinx



BIBLIOGRAFÍA:

- “OS and Libraries Document Collection”, Xilinx → Librerías
- “Embedded System Tools Reference Guide”, Appendix B *Interrupt Management*