

# Sistemas Críticos

## Tema 2: *FreeRTOS*

Lección 5:  
*Port de FreeRTOS al Zynq Processing System de la Zybo*



# Contenidos

## Tema 2: *FreeRTOS*

Creación de un BSP para nuestra plataforma

*Port de FreeRTOS al Zynq Processing System de la Zybo*

Uso del *port de FreeRTOS*

Generación del *First Stage Boot Loader*

Preparación de la imagen de arranque

Usaremos el mismo diseño que usamos para *Linux*

## Variables de entorno

```
export PRJ_ROOT="${HOME}/zynq-freertos"  
export PLATFORM="my_zynq_platform"  
export PLATFORM_DIR="${PRJ_ROOT}/${PLATFORM}"
```

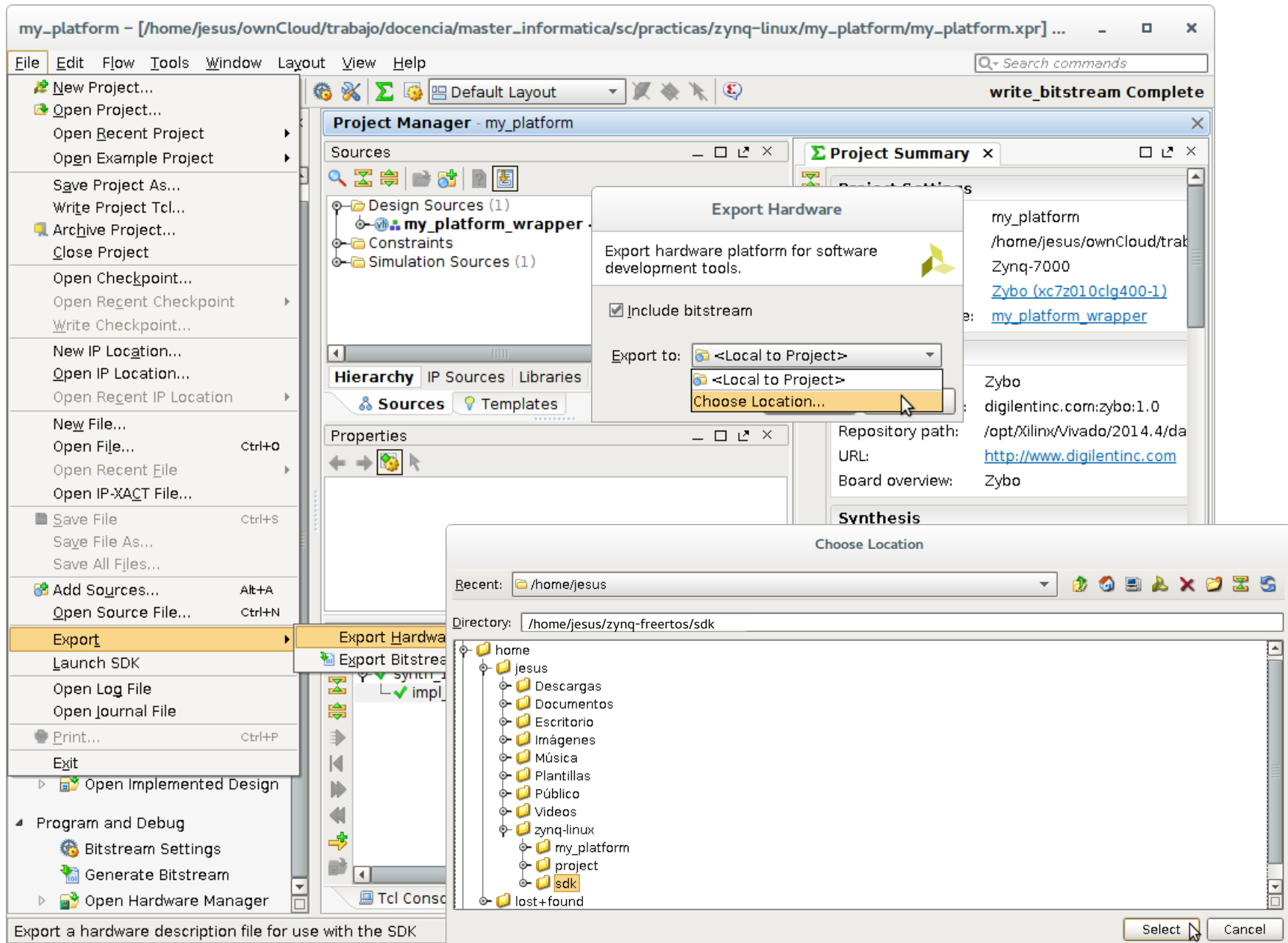
## Generación automática de la plataforma

```
vivado -mode batch -source platform.tcl
```

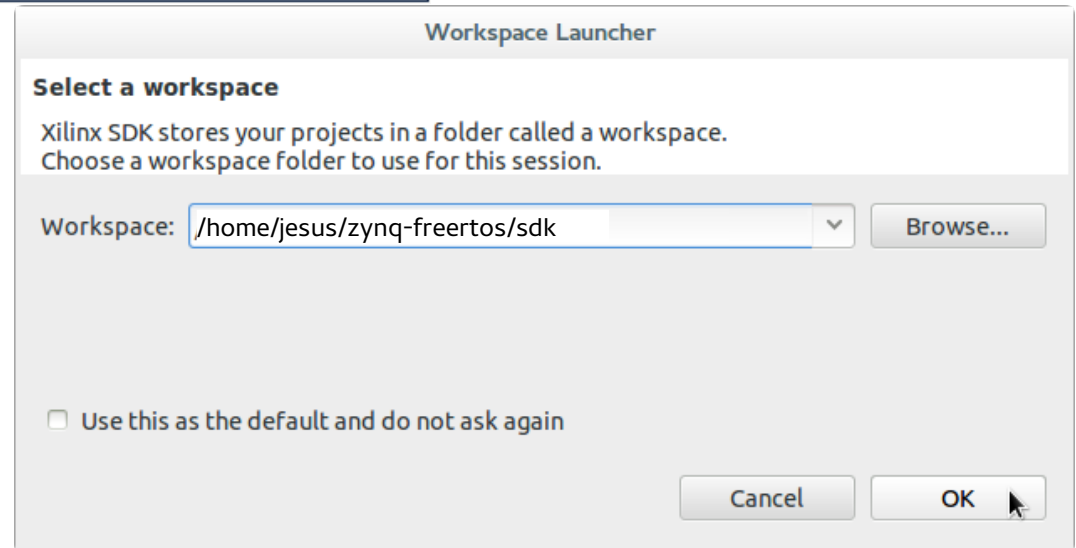
# Exportamos el diseño de la plataforma de Vivado al SDK



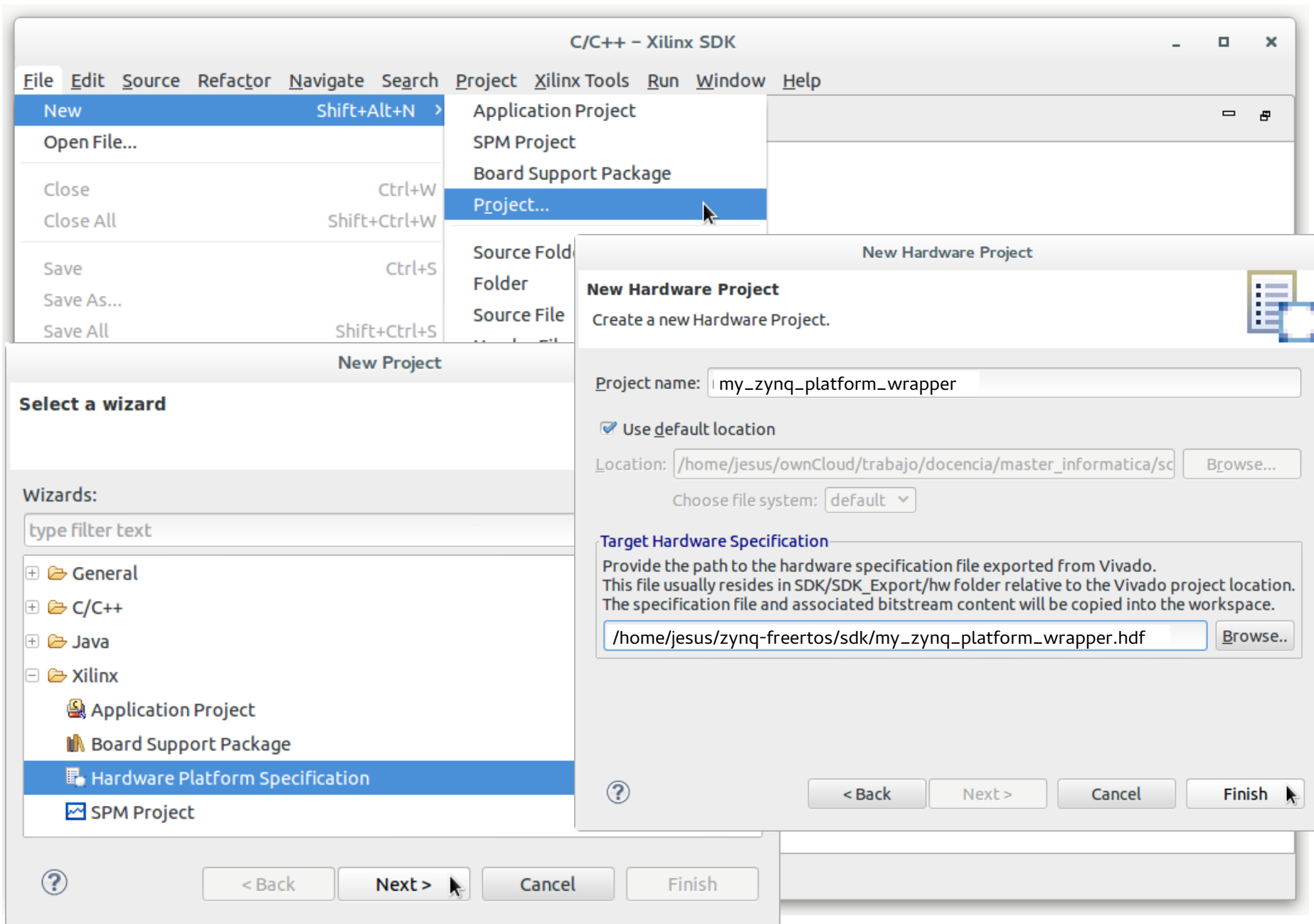
# Exportamos el diseño de la plataforma de Vivado al SDK



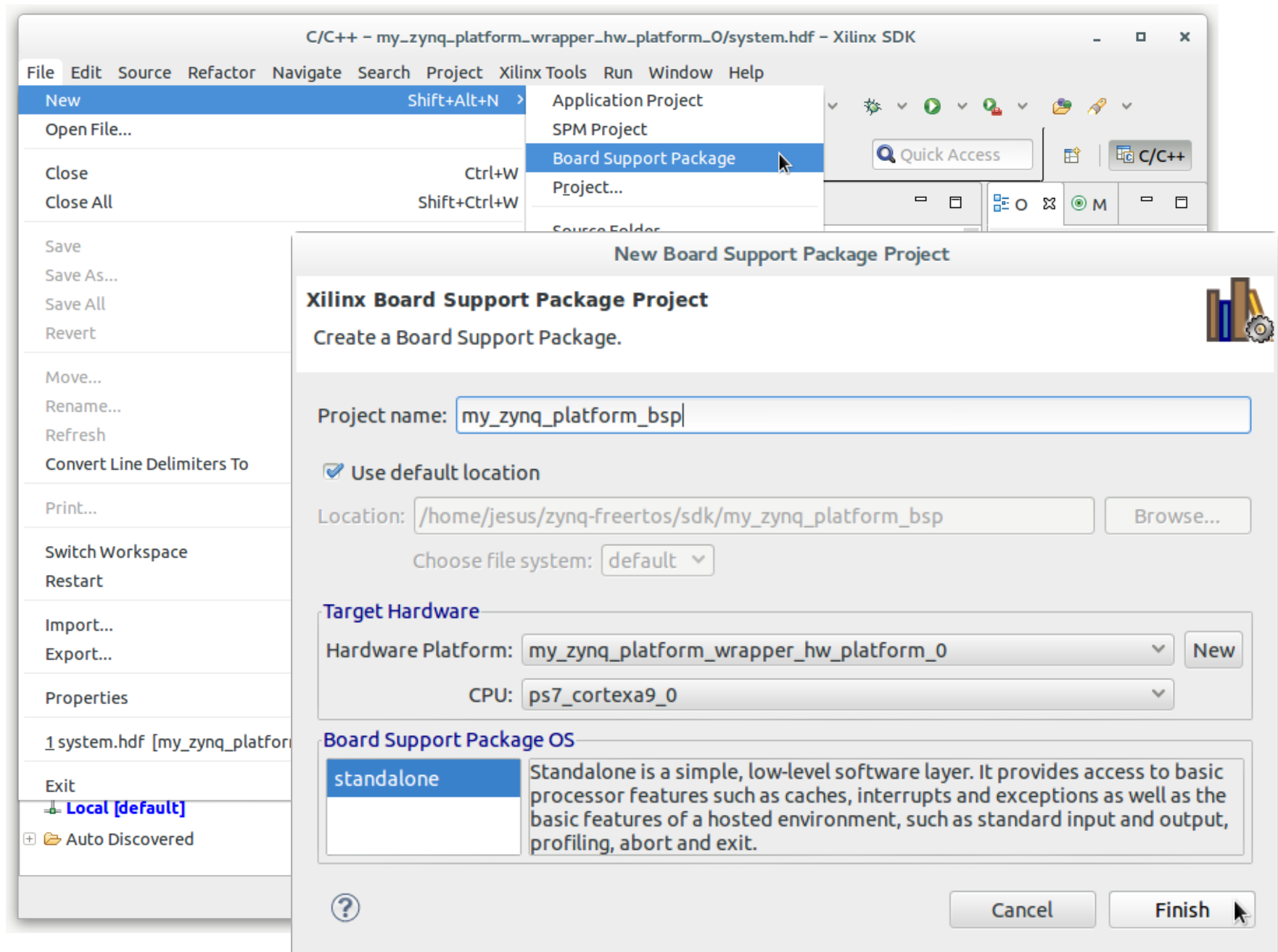
# Creación de un proyecto HW en el *SDK* para nuestra plataforma



# Creación de un proyecto HW en el SDK para nuestra plataforma



# Creamos un BSP en el SDK para nuestra plataforma





# Creamos un BSP en el SDK para nuestra plataforma

Board Support Package Settings

Board Support Package Settings

Control various settings of your Board Support Package.

Overview

standalone

drivers

ps7\_cortexa9\_0

my\_zynq\_platform\_bsp

OS Type: standalone

OS Version: 4.2

Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.

Target Hardware

Hardware Specification: /home/jesus/zynq-freertos/sdk/my\_zynq\_platform\_wrapper\_hw\_platform\_0/system.hdf

Processor: ps7\_cortexa9\_0

Supported Libraries

Check the box next to the libraries you want included in your Board Support Package. You can configure the library in the navigator on the left.

	Name	Version	Description
<input type="checkbox"/>	lwip140	2.3	lwIP TCP/IP Stack library: lwIP v1.4.0, Xilinx adapter v1.05.a
<input type="checkbox"/>	xilffs	2.2	Generic Fat File System Library
<input type="checkbox"/>	xilflash	4.0	Xilinx Flash library for Intel/AMD CFI compliant parallel flash
<input type="checkbox"/>	xilisf	5.0	Xilinx In-system and Serial Flash Library
<input type="checkbox"/>	xilmfs	2.0	Xilinx Memory File System
<input type="checkbox"/>	xilrsa	1.1	Xilinx RSA Library
<input type="checkbox"/>	xilskey	2.0	Xilinx Secure Key Library

?

Cancel

OK

# Creación del HW *wrapper* y el BSP mediante *scripts*

## Variables de entorno

```
export PLATFORM_WRAPPER="${PLATFORM}_wrapper"  
export SDK_DIR="${PRJ_ROOT}/sdk"  
export BSP="${PLATFORM}_bsp"
```

## Exportamos el diseño de la plataforma

```
mkdir -p ${SDK_DIR}  
cp ${PLATFORM_DIR}/${PLATFORM}.runs/impl_1/${PLATFORM_WRAPPER}.sysdef \  
  ${SDK_DIR}/${PLATFORM_WRAPPER}.hdf
```

## Generamos el HW *wrapper* y el BSP

```
xsdk -batch -source bsp.tcl
```

## Fichero *bsp.tcl*

```
set_workspace $env(SDK_DIR)  
create_project -type hw -name $env(PLATFORM_WRAPPER) \  
  -hwspec $env(SDK_DIR)/$env(PLATFORM_WRAPPER).hdf  
create_project -type bsp -name $env(BSP) -hwproject $env(PLATFORM_WRAPPER) \  
  -proc ps7_cortexa9_0 -os standalone  
build -type all  
exit
```

# Contenidos

## Tema 2: *FreeRTOS*

Creación de un BSP para nuestra plataforma

*Port de FreeRTOS al Zynq Processing System de la Zybo*

Uso del *port* de *FreeRTOS*

Generación del *First Stage Boot Loader*

Preparación de la imagen de arranque

# Descargamos el código fuente de *FreeRTOS*

## Variables de entorno

```
FREERTOS_VER="8.2.1"
FREERTOS="FreeRTOSV${FREERTOS_VER}"
ZIPFILE=${FREERTOS}.zip
FREERTOS_DIR="${PRJ_ROOT}/freertos"
FREERTOS_FORGE="http://sourceforge.net/projects/freertos/"
FREERTOS_SRC_DIR="${FREERTOS_DIR}/${FREERTOS}/FreeRTOS"
FREERTOS_KERNEL_DIR="${FREERTOS_SRC_DIR}/Source"
FREERTOS_PORTABLE_DIR="${FREERTOS_KERNEL_DIR}/portable"
FREERTOS_DEMO_DIR="${FREERTOS_SRC_DIR}/Demo/CORTEX_A9_Zynq_ZC702/RTOSDemo"
FREERTOS_PORT_DIR="${SDK_DIR}/${FREERTOS_PORT}"
```

## Descargamos *FreeRTOS*

```
mkdir -p ${FREERTOS_DIR}
cd ${FREERTOS_DIR}
wget ${FREERTOS_FORGE}/files/FreeRTOS/V${FREERTOS_VER}/${ZIPFILE}
unzip ${ZIPFILE}
```

# Copiamos los ficheros necesarios

Creamos un directorio para el *port* para la *Zybo* dentro de *workspace* del *SDK*

```
mkdir -p ${FREERTOS_PORT_DIR}
```

Copiamos el *kernel* de *FreeRTOS* a nuestro directorio

```
cd ${FREERTOS_PORT_DIR}
mkdir ./src ./include
cp ${FREERTOS_KERNEL_DIR}/* ./src
cp ${FREERTOS_KERNEL_DIR}/include/* ./include
```

Copiamos los ficheros del *port* de *FreeRTOS* para *ARM Cortex A9*

```
cp ${FREERTOS_PORTABLE_DIR}/GCC/ARM_CA9/portASM.S ./src
cp ${FREERTOS_PORTABLE_DIR}/GCC/ARM_CA9/port.c ./src
cp ${FREERTOS_PORTABLE_DIR}/GCC/ARM_CA9/portmacro.h ./include
```

Copiamos la implementación de las funciones de gestión de memoria

```
cp ${FREERTOS_PORTABLE_DIR}/MemMang/heap_4.c .
```

Copiamos los ficheros imprescindibles de la demo de *FreeRTOS* para el *Zynq*

```
cp ${FREERTOS_DEMO_DIR}/src/FreeRTOSConfig.h ./include
cp ${FREERTOS_DEMO_DIR}/src/FreeRTOS_asm_vectors.S ./src
cp ${FREERTOS_DEMO_DIR}/src/FreeRTOS_tick_config.c ./src
```

Corregimos erratas (“Task.h” → “task.h” en *FreeRTOS\_tick\_config.c*)

```
sed -i 's/"Task.h"/"task.h"/g' ./src/FreeRTOS_tick_config.c
```

# Extraemos la inicialización del Zynq del fichero main.c de la demo

## Creamos fichero \${FREERTOS\_PORT\_DIR}/src/setup.c

Extraemos el código de inicialización del Zynq del fichero main.c de la demo de *FreeRTOS* (`${FREERTOS_DEMO_DIR}/src/main.c`)

### Includes:

```
#include <limits.h>
#include "xscugic.h"
#include "setup.h"
```

Lo crearemos a continuación

### Declaraciones:

```
extern void vPortInstallFreeRTOSVectorTable( void );
XScuWdt xWatchDogInstance;
XScuGic xInterruptController;
```

Quitamos la *keyword* `static` y la llamada a la función `vParTestInitialise()`

### Funciones:

```
void prvSetupHardware (void)
void vApplicationMallocFailedHook (void)
void vApplicationStackOverflowHook (TaskHandle_t pxTask, char *pcTaskName)
void vApplicationIdleHook (void)
void vAssertCalled (const char * pcFile, unsigned long ulLine)
void vApplicationTickHook (void)
void vInitialiseTimerForRunTimeStats (void)
void * memcpy (void *pvDest, const void *pvSource, size_t ulBytes)
void * memset (void *pvDest, int iValue, size_t ulBytes)
int memcmp (const void *pvMem1, const void *pvMem2, size_t ulBytes)
```

Implementación vacía

# Extraemos la inicialización del *Zynq* del fichero `main.c` de la demo

## Creamos fichero `${FREERTOS_PORT_DIR}/include/setup.h`

Cabeceras de las funciones de inicialización del *Zynq* para incluirlas en nuestras aplicaciones

```
#ifndef __SETUP_H
#define __SETUP_H

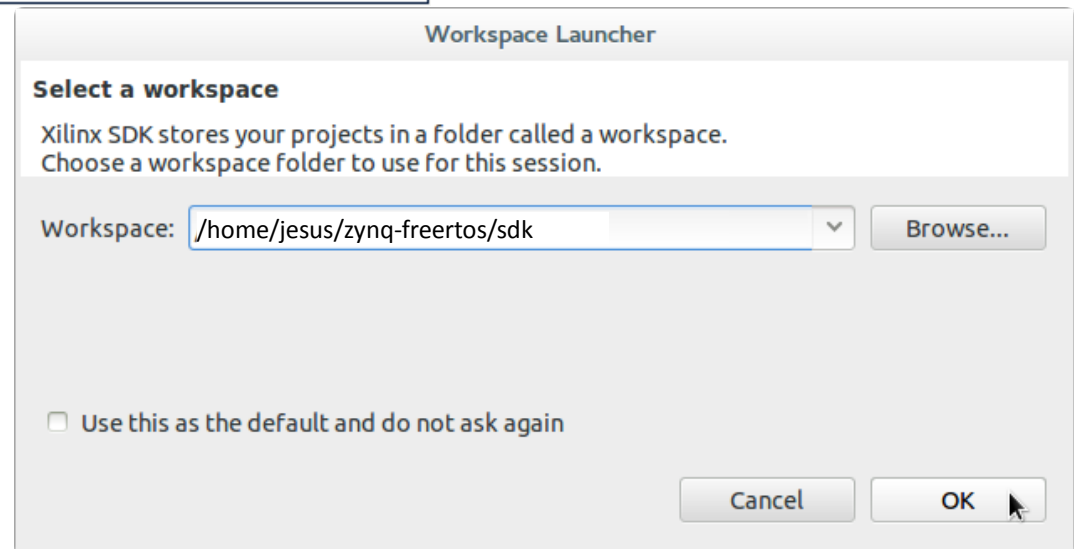
/* Scheduler include files. */
#include "FreeRTOS.h"
#include "task.h"

/*
 * Configure the hardware as necessary to run FreeRTOS on a Zynq SoC
 */
void prvSetupHardware( void );

/* Prototypes for the standard FreeRTOS callback/hook functions */
void vApplicationMallocFailedHook( void );
void vApplicationIdleHook( void );
void vApplicationStackOverflowHook( TaskHandle_t pxTask, char *pcTaskName );
void vApplicationTickHook( void );

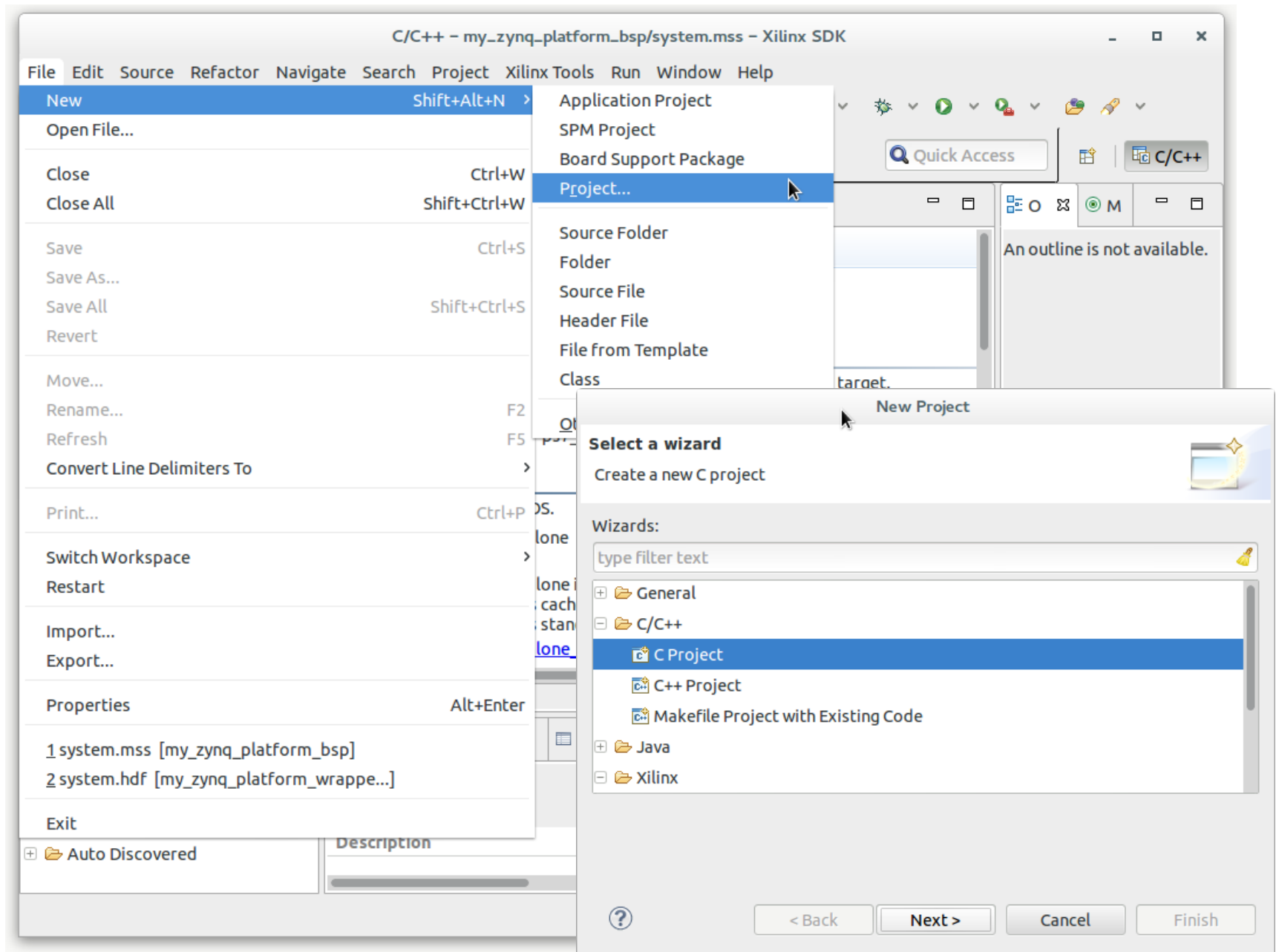
#endif
```

# Creación de un proyecto de biblioteca para el *port* de *FreeRTOS* en *SDK*





# Creación de un proyecto de biblioteca para el *port* de FreeRTOS en SDK



# Creación de un proyecto de biblioteca para el *port* de *FreeRTOS* en *SDK*

C Project

⚠ Directory with specified name already exists.

Project name: my\_zynq\_platform\_freertos\_port

☒ Use default location

Location: /home/jesus/zynq-freertos/sdk/my\_zynq\_platform\_freertos\_port Browse...

Choose file system: default ▾

Project type:

- + Executable
- + Shared Library
- + Static Library
  - ◆ Xilinx ARM Executable
  - ◆ **Xilinx ARM Static Library**
  - ◆ Xilinx ARM Linux Executable
  - ◆ Xilinx ARM Linux Static Library
  - ◆ Xilinx MicroBlaze Executable
  - ◆ Xilinx MicroBlaze Static Library
  - ◆ Xilinx MicroBlaze Linux Executable (Little-Endian)
  - ◆ Xilinx MicroBlaze Linux Static Library (Little-Endian)
  - ◆ Xilinx MicroBlaze Linux Executable (Big-Endian)
  - ◆ Xilinx MicroBlaze Linux Static Library (Big-Endian)

Toolchains:

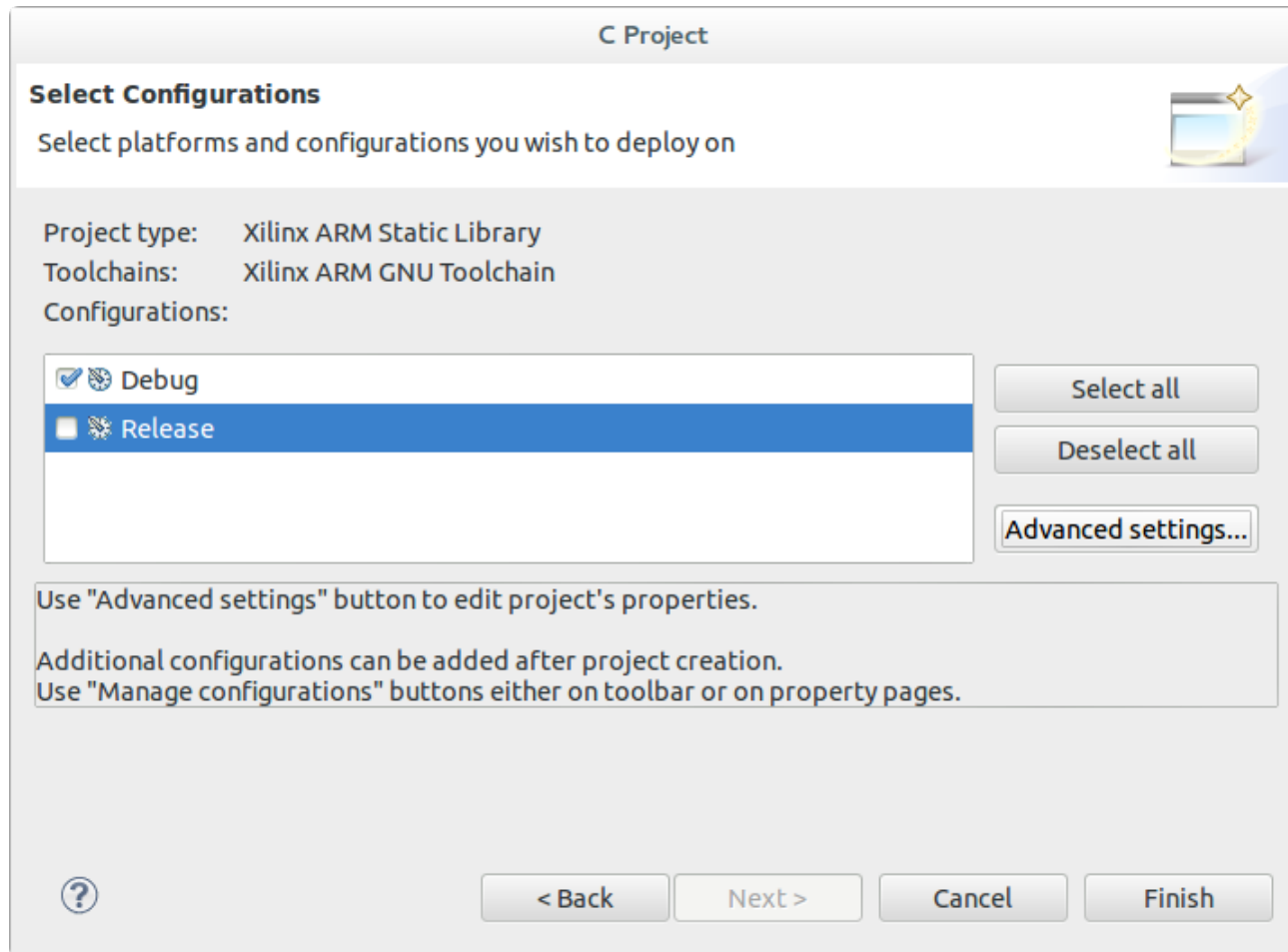
- Xilinx ARM GNU Toolchain**

☒ Show project types and toolchains only if they are supported on the platform

? < Back Next > Cancel Finish

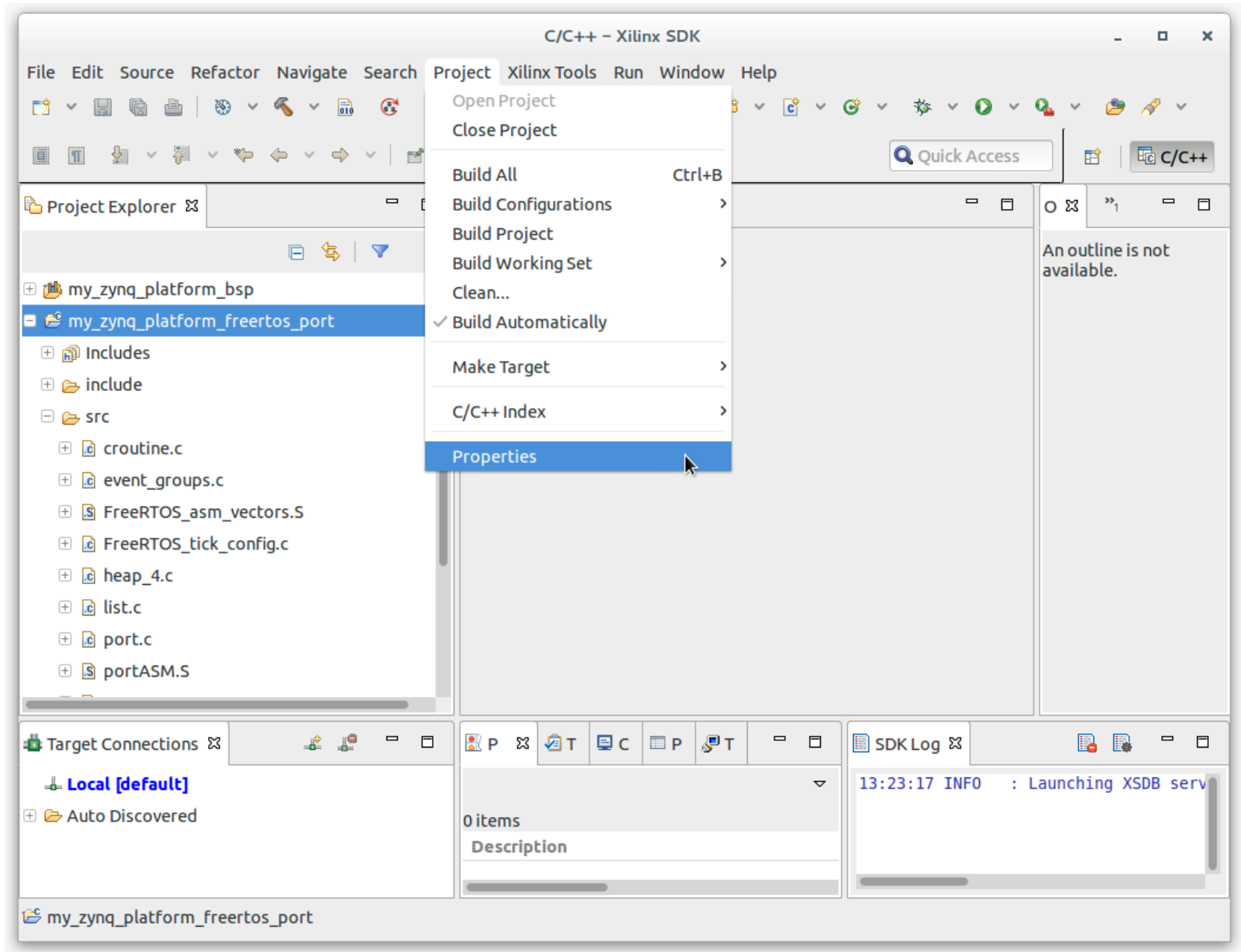
Usamos el mismo directorio al que hemos copiado los ficheros del *port* de *FreeRTOS*

# Creación de un proyecto de biblioteca para el *port* de *FreeRTOS* en *SDK*

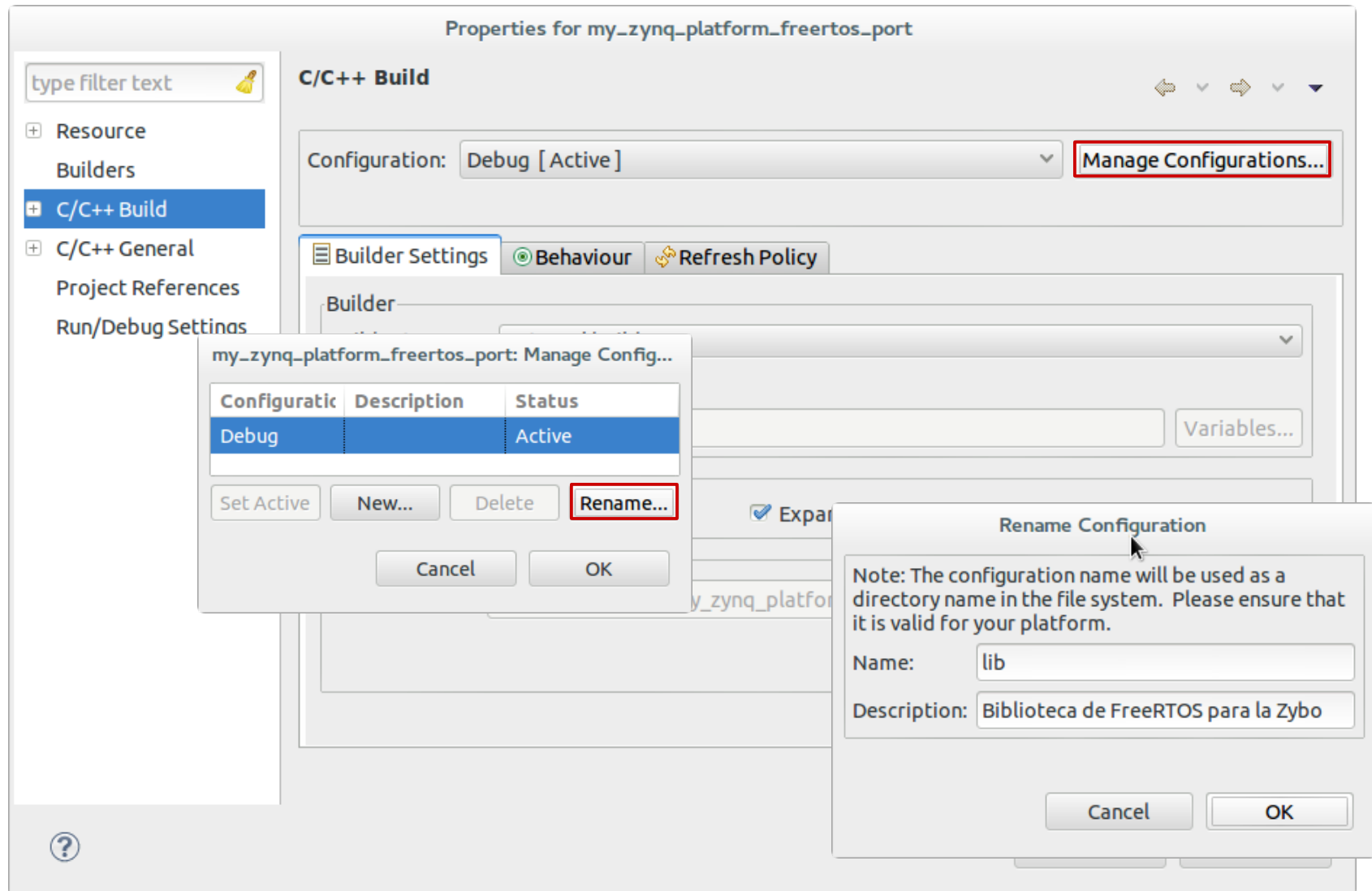


Desactivamos la configuración *Release* y pulsamos el botón *Finish*

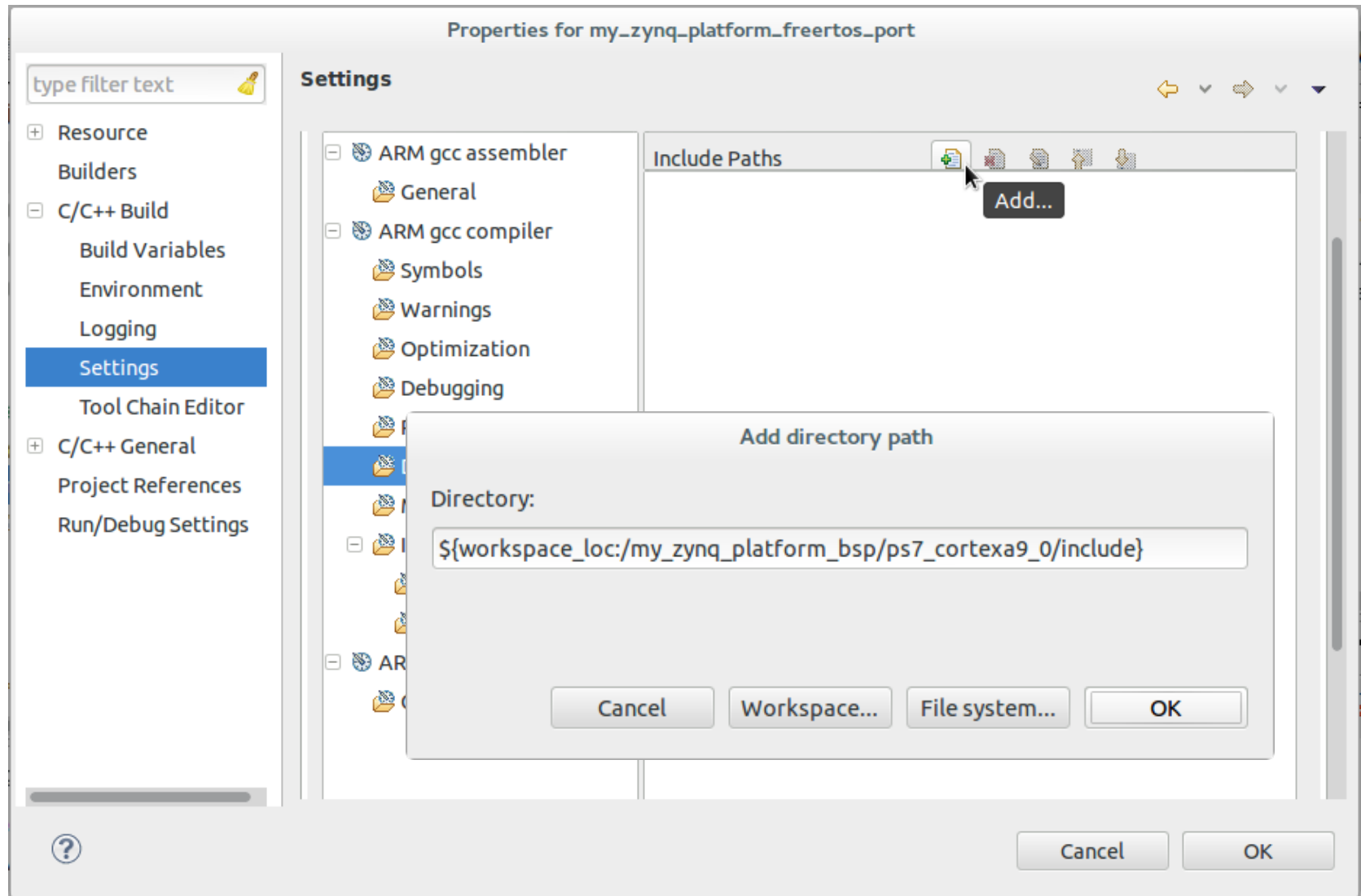
# Configuración del proyecto de biblioteca para el *port* de *FreeRTOS*



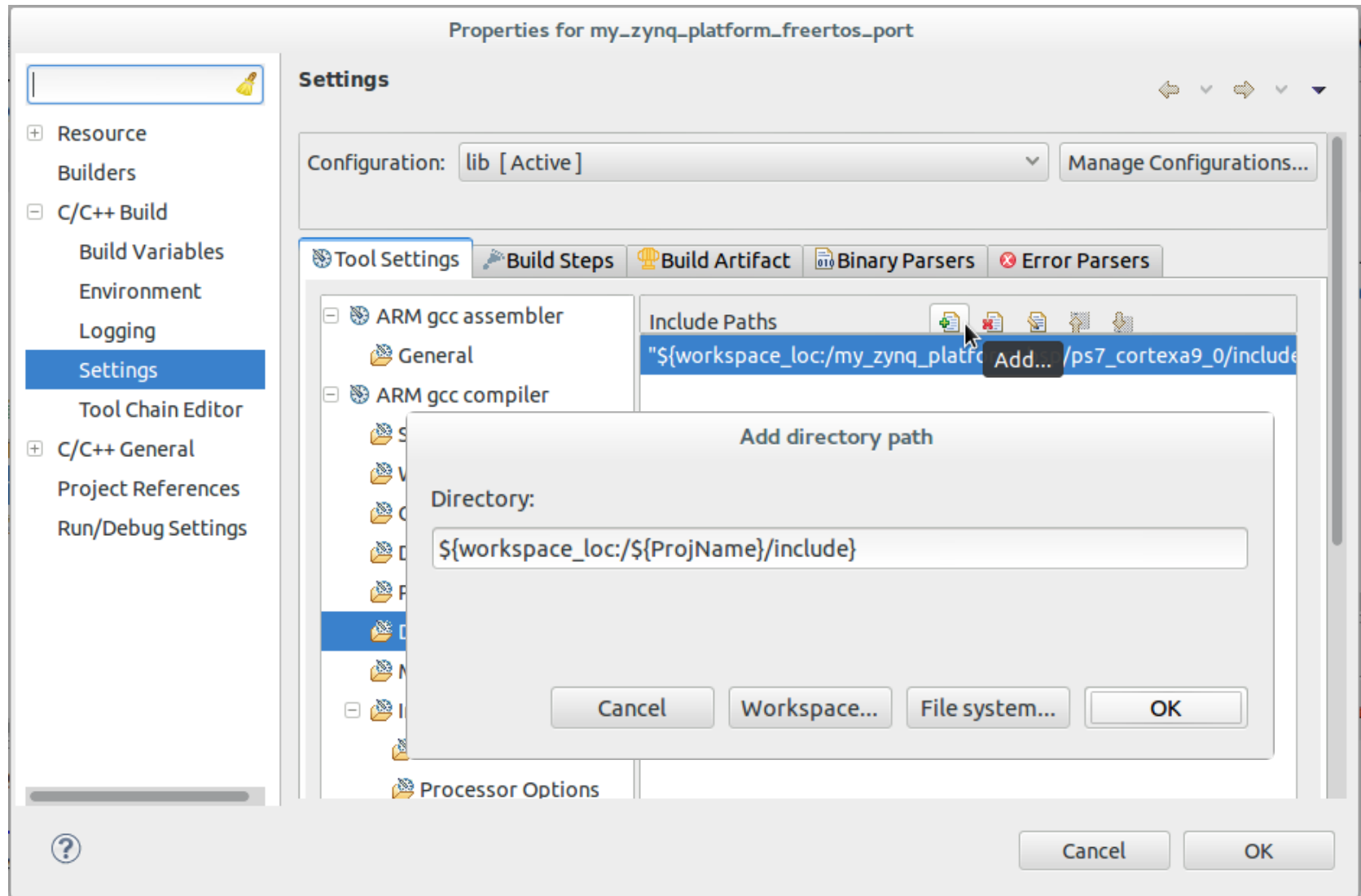
# La biblioteca se generará en el directorio lib



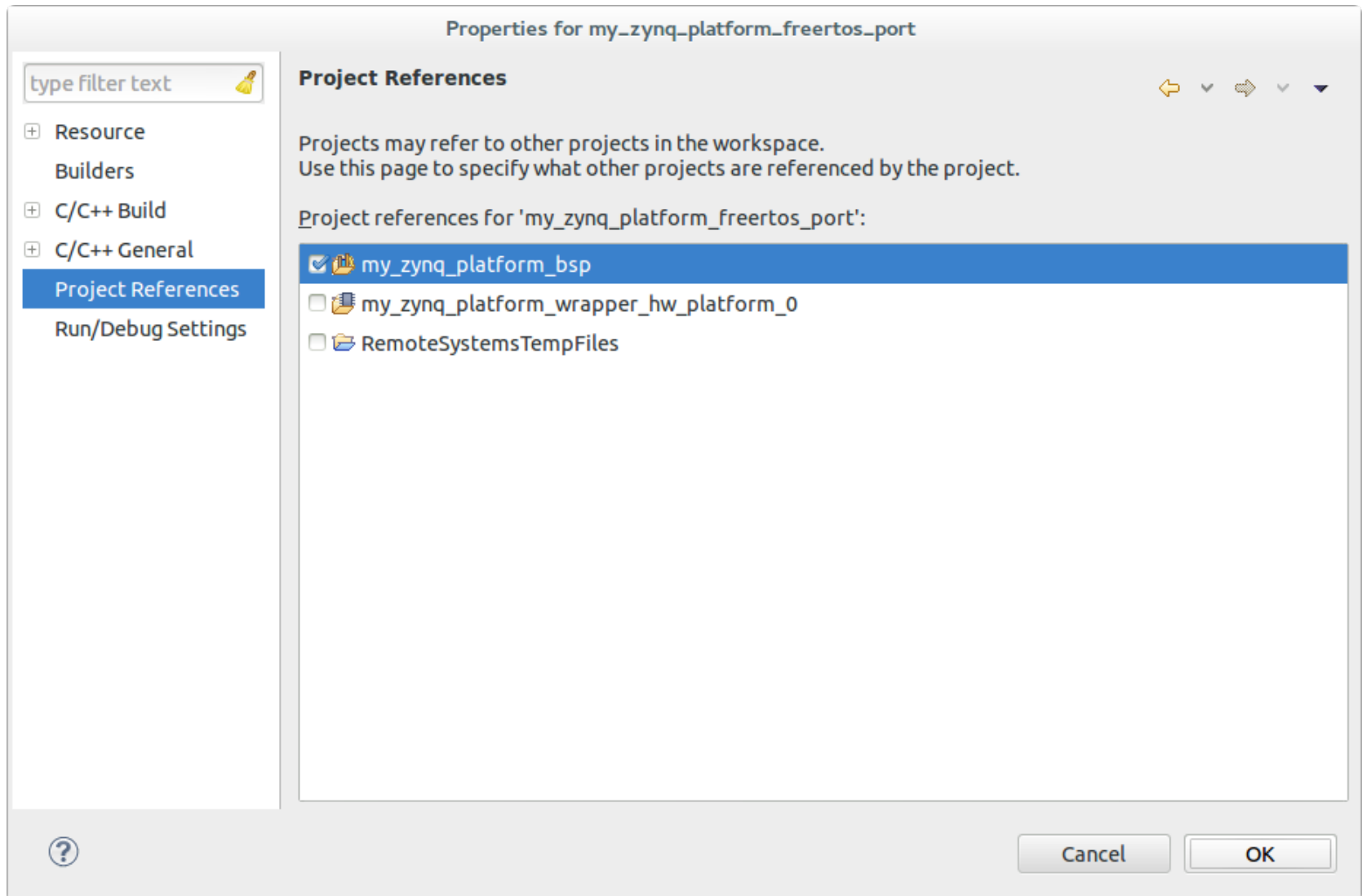
# Añadimos el directorio con las cabeceras de las funciones del BSP



# Añadimos el directorio con las cabeceras de las funciones de *FreeRTOS*

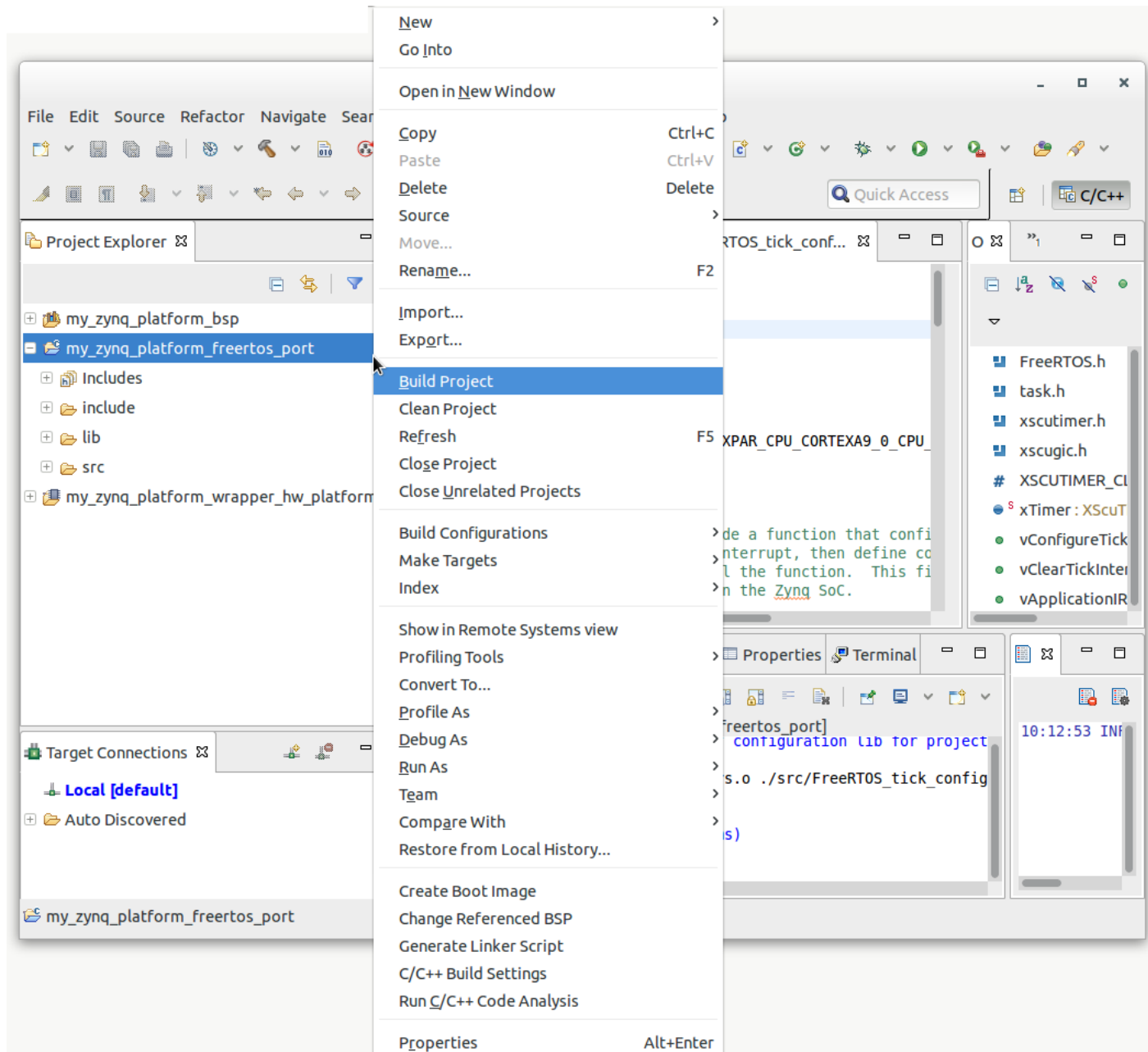


# Indicamos que el *port* de *FreeRTOS* depende del BSP

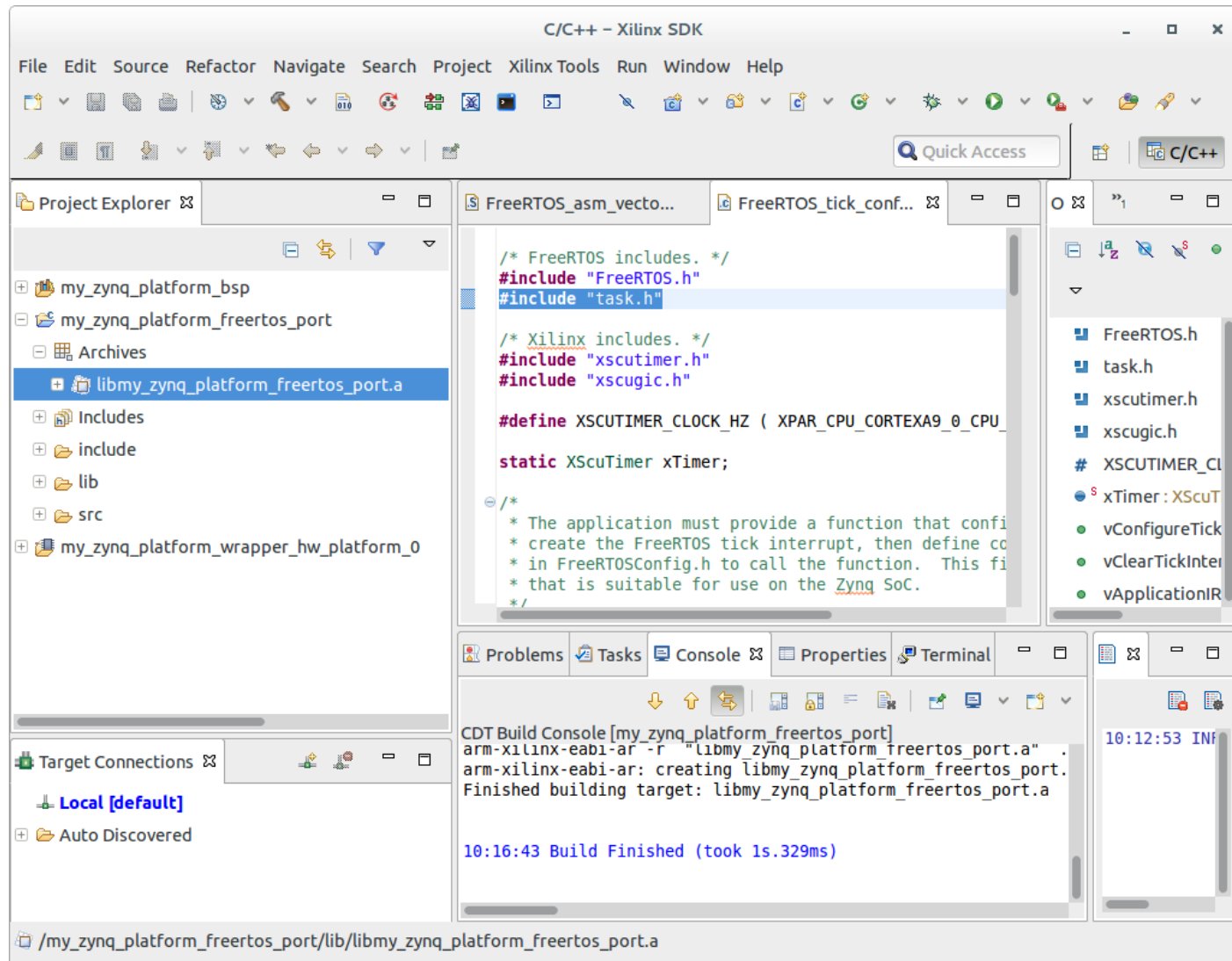




# Construimos el proyecto



# Obtenemos la biblioteca con el *port* de *FreeRTOS*



# Lecturas recomendadas

## Documentación oficial de *FreeRTOS*:

Real Time Engineers Ltd. *FreeRTOS Quick Start Guide*.

<http://www.freertos.org/FreeRTOS-quick-start-guide.html>

Richard Barry. *Using the FreeRTOS Real Time Kernel: A Practical Guide*. Real Time Engineers, 2010. Disponible en la biblioteca

Real Time Engineers Ltd. *Book Companion Source Code*.

<http://www.freertos.org/Documentation/code/>

Real Time Engineers Ltd. *Real Time Application Design Tutorial. Using FreeRTOS in small embedded systems*. <http://www.freertos.org/tutorial/>

## Cursos de FreeRTOS:

Amr Ali. *FreeRTOS Course – Introduction to FreeRTOS*.

<http://embedded-tips.blogspot.com.es/2010/06/free-freertos-course-introduction-to.html>

Amr Ali. *FreeRTOS Course - Task Management*.

<http://es.slideshare.net/amraldo/free-freertos-coursetask-management>

Amr Ali. *FreeRTOS Course - Queue Management*.

<http://es.slideshare.net/amraldo/m3-introduction-to-free-rtos-v605>

Amr Ali. *FreeRTOS Course - Semaphore/Mutex Management*.

<http://es.slideshare.net/amraldo/freertos-course-semaphoremutex-management>

# Lecturas recomendadas

## *FreeRTOS porting:*

Real Time Engineers Ltd. *Official FreeRTOS Ports.*

[http://www.freertos.org/RTOS\\_ports.html](http://www.freertos.org/RTOS_ports.html)

Real Time Engineers Ltd. *Modifying a FreeRTOS Demo.*

<http://www.freertos.org/porting-a-freertos-demo-to-different-hardware.html>

Real Time Engineers Ltd. *Creating a New FreeRTOS Port.*

<http://www.freertos.org/FreeRTOS-porting-guide.html>

## *Port de FreeRTOS para el SoC Zynq:*

Real Time Engineers Ltd. *Xilinx Zynq-7000 (dual core ARM Cortex-A9) SoC Port.*

<http://www.freertos.org/RTOS-Xilinx-Zynq.html>

Circuit Sense. *FreeRTOS on Xilinx Zynq Zybo [Single Core]*, abril 2015.

<http://rishifranklin.blogspot.com.es/2015/04/freertos-on-xilinx-zynq-zybo-single-core.html>