



January 9, 2012.

Application Note: 0031

Version: v1.00

## **1** Introduction

The purpose of this application note is to summarize and explain tasks required to use software device drivers for Xylon's logicBRICKS<sup>TM</sup> IP cores in standalone (no operating system) Xilinx® FPGA based system configurations. The described workflow works with the Xilinx Software Development Kit (SDK) embedded software development environment for Xilinx embedded processors.

logicBRICKS is Xylon's library of IP cores designed for Xilinx FPGA and EPP products. All logicBRICKS IP cores come with detailed documentation, software support and reference designs. The IP cores are fully supported by the Xilinx Platform Studio (XPS) and the EDK integrated software solution, which enables FPGA designers that are familiar with Xilinx tools to start working immediately. The logicBRICKS allow for configuration through a graphical user interface (GUI) and the implementation of targeted SoCs without hand coding.

logicBRICKS software device drivers fully support the functionality and configurability of related logicBRICKS IP cores.

## 2 Typical logicBRICKS IP Core Deliverables

logicBRICKS IP core deliverables are compatible with the Xilinx Platform Studio and the EDK integrated software solution. The deliverables include: encrypted VHDL IP core's sources, device driver and xyl\_oslib library.

The folder structure of the logicBRICKS IP core:

- pcores

ip\_core HW description

- data (XPS related files)
- doc (IP documentation)
- hdl (encrypted IP HDL sources)
- simmodel (simulation model)
- drivers

 $\circ$  ip\_core\_generic\_driver

- src (driver sources and makefile)
- data (the .mdd and .tcl driver files)
- examples (driver examples)
- doc (driver documentation (please see doc/html/index.html))
- sw\_services
  - xyl\_oslib (Xylon OS abstraction library for Xilinx Xilkernel embedded kernel and BSP; support for other OS on request)





January 9, 2012.

Application Note: 0031

Version: v1.00

## 3 What Are xyl\_oslib and logicBRICKS Generic Device Drivers?

Xylon's generic device drivers for logicBRICKS IP cores are similar to software device drivers for Xilinx's IP cores delivered with the Xilinx ISE® Design Suite: Embedded Edition. The logicBRICKS device drivers are generic because they are fully independent from the operating system.

In addition to the logicBRICKS software device drivers, Xylon provides an operating system abstraction library called xyl\_oslib.

Xyl\_oslib provides the following interfaces to generic device drivers:

- Bus access to memory/registers
- Data types
- Memory allocation
- Timers
- DMA
- Debug
- SpinLock

A quick look at logicBRICKS device drivers' source code shows that they use the xyl\_oslib system abstraction library extensively. It makes the maintenance easier, and reduces design efforts during IP core porting to different operating systems and hardware platforms.

Please check the Chapter 7 for more details on xyl\_oslib library.

# 4 Compiling the logicBRICKS Device Driver in the Xilinx SDK

Xylon logicBRICKS device drivers are designed to work with the Xilinx SDK Software Development Kit, and are very similar to Xilinx device drivers. However, there are small differences when working with the Xylon and Xilinx device drivers.

### 4.1 Including logicBRICKS Drivers in the SDK Search Path

Xylon logicBRICKS drivers are not delivered with the Xilinx ISE/EDK installation. They have to be obtained from Xylon and added in the SDK search path:

- select **SDK**->**Xilinx Tools** -> **Repositories**, the Preferences dialog box opens (Fig. 4.1)
- select New and browse for directory containing the Xylon drivers and SW\_services folders
- click on Rescan Repositories



4.1 Setting up the Search Path to Xylon Drivers/SW Services

### 4.2 Including logicBRICKS Drivers in the SDK "Xilinx BSP project"

Properly setup search path enables the SDK to locate device drivers for Xylon logicBRICKS IP cores. The following steps describe the procedure for a new BSP project creation:

- 1. **SDK-> New -> Xilinx Board Support Project** (select HW platform and standalone/Xilkernel) and click Finish (Fig. 4.2)
- 2. In the Board Support Package Settings window (Fig. 4.3):
  - a. Select the xyl\_oslib checkbox under Overview -> Supported libraries
  - b. Select the Xylon **drivers** for Xylon IP cores under Overview->drivers
  - c. Add "-DNDEBUG" in the Overview->Drivers->CPU->extra\_compiler\_flags (Fig. 4.4). This recommended (not mandatory) compiler flag reduces the size of the xyl\_oslib library when it is used by drivers.

Xylon drivers and xyl\_oslib library compile like Xilinx drivers, but the resulting static library is called "libdrivers.a".





January 9, 2012.

#### Application Note: 0031

Version: v1.00

ard Support P	ackage Settings			
ntrol various setti	ngs of your Board Support	Package.		
Overview standalone	standalone_bsp_0_tes	t		
drivers cpu	OS Type: standato OS Version: 3.01.a	<b>▼</b>	Standalone is a simple, low-level software layer. It provides access to bas interrupts and exceptions as well as the basic features of a hosted enviro profiling, abort and exit.	ic processor features such as caches, nment, such as standard input and output
	Hardware Specificatio	on:	:t\software\SDKworkspace\EDK_CC_hw	platform\system.xml
		minuclear 0		
	Check the box next t	o the libraries you	u want included in your Board Support Package.You can configure the li	brary in the navigator on the left.
	Check the box next t	the libraries you	u want included in your Board Support Package.You can configure the li Description	brary in the navigator on the left.
	Check the box next f	Version	u want included in your Board Support Package.You can configure the li Description FAT Generic File System Module July TCTQP Stack library, July 3.0, Xiliny adapter, y	brary in the navigator on the left.
	Check the box next t	Version 1.08.t 3.01.a 1.02.t	u want included in your Board Support Package.You can configure the li Description FAT Generic File System Module IwiP TCP/IP Stack library: IwiP v1.3.0, Xilinx adapter v Xylon Platform Library	brary in the navigator on the left.
	Check the box next the loss ne	Version 1.08.t 3.01.a 1.02.t 1.00.a	u want included in your Board Support Package.You can configure the li Description FAT Generic File System Module wiP TCP/IP Stack library: IwIP v1.3.0, Xilinx adapter v Xylon Platform Library Provides read/write routines to access files stored on	brary in the navigator on the left.
	Check the box next t Name fat_fs buip130 platform xilfatfs xilflash	Version 1.08.t 3.01.a 1.02.t 1.00.a 3.00.a	u want included in your Board Support Package.You can configure the li Description FAT Generic File System Module IwiP TCP/IP Stack library: IwIP v1.3.0, Xilinx adapter v Xylon Platform Library Provides read/write routines to access files stored on Xilinx Flash library for Intel/AMD CFI compliant paral	brary in the navigator on the left.
	Processor: Check the box next t fat_fs invip130 platform xilfatfs xilfash xillisf	Version 1.08.t 3.01.a 1.02.t 1.00.a 3.00.a 2.03.a	u want included in your Board Support Package.You can configure the li Description FAT Generic File System Module MvIP TCP/IP Stack library: NvIP v1.3.0, Xilinx adapter v Xylon Platform Library Provides read/write routines to access files stored on Xilinx Hash library for Intel/AMD CFI compliant paral Xilinx In-system and Serial Flash Library	brary in the navigator on the left.
	Processor: Check the box next t Name fat_fs ivvip130 platform xilfatfs xilfatfs xillisf xillisf xillisf	Version 1.08.t 3.01.a 1.02.t 1.00.a 3.00.a 2.03.a 1.00.a	u want included in your Board Support Package.You can configure the li Description FAT Generic File System Module WiP TCP/IP Stack library: IwIP v1.3.0, Xilinx adapter v Xylon Platform Library Provides read/write routines to access files stored on Xilinx Hash library for Intel/AMD CFI compliant paral Xilinx In-system and Serial Flash Library Xilinx In-system and Serial Flash Library Xilinx Memory File System Yoken O Librare	brary in the navigator on the left.
	Processor: Check the box next t Name fat_fs ivip130 platform xilfatfs xilfash xilfash xilisf xilmfs y xyl_oslib	Version 1.08.t 3.01.a 1.02.t 1.00.a 3.00.a 2.03.a 1.00.a 1.03.t	u want included in your Board Support Package.You can configure the li Description FAT Generic File System Module Infl TCP/IP Stack library: Infl P v1.3.0, Xilinx adapter v Xylon Platform Library Provides read/write routines to access files stored on Xilinx Fahs library for Intel/AMD CFI compliant paral Xilinx In-system and Serial Flash Library Xilinx Memory File System Xylon OS Library	brary in the navigator on the left.
	Check the box next t Name fat_fs ivip130 platform xilfatfs xillfash xillsf xillsf xillsf xillsf	Version 1.08.t 3.01.a 1.02.t 1.00.a 3.00.a 2.03.a 1.00.a 1.03.t	u want included in your Board Support Package.You can configure the li Description FAT Generic File System Module W/P TCP/IP Stack library: W/P v1.3.0, Xilinx adapter v Xylon Platform Library Provides read/write routines to access files stored on Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx Flash library for Intel/AMD CFI compliant paral Xilinx Memory File System Xylon OS Library	brary in the navigator on the left.

#### 4.2 Selecting xyl\_oslib Library

and randas secting	gs of your Board Suppor	t Package.				
Overview						
standalone	Diam					
drivers	Drivers					
cpu	assigned for each o	omnonent. If you do n	ot want to assign:	a driver to a componer	at or peripheral please choose	'none'
	ussigned for each e	omponentin you do m	or mane to assign		ie or penpriera, prease enoos	
	Component	Component Type	Driver	Driver Versi		
	logibayer_2	logibayer	logibayer	2.04.a		
	logibayer_3	logibayer	logibayer	2.04.a		
	logibayer_4	logibayer	logibayer	2.04.a		
	logibitblt_0	logibitblt	logiBITBLT	2.02.t		
	logics_0	logics	generic	1.00.a		-
	logicvc_0	logicvc	logiCVC	2.05.t		
	logimem arb 0	logimem_arb	generic	1.00.a		
	logineni_arb_o	1.	logiSDHC	1.00.c		
	logisdhc_0	logisanc				

4.3 Selecting Xylon Drivers





January 9, 2012.

Application Note: 0031

Version: v1.00

Board Support Package Settings				
Board Support Pa Control various setting	<b>ckage Settings</b> gs of your Board Support Package.			
Overview standalone	Configuration for driver: cpu			
drivers cpu	Name	Value	Default	
	compiler	mb-gcc	mb-gcc	
	archiver	mb-ar	mb-ar	
	compiler_flags	-02 -c	-02 -c	
	extra_compiler_flags	-g -DNDEBUG	-g	
	xmdstub_peripheral	none	none	
	< III		Þ	
?		ок	Cancel	

4.4 Compiler Flags (-DNDEBUG)

# **5 Using logicBRICKS Drivers from the SDK SW Applications**

There are two main differences between the typical Xilinx SDK driver and Xylon logicBRICKS generic driver:

- 1. logicBRICKS drivers and xyl\_oslib binaries are linked to a static library called "libdrivers.a", which has to be setup in the application project linker settings
- 2. logicBRICKS drivers depend on the xyl\_oslib library. Xyl\_oslib for Xilinx Xilkernel or Xilinx BSP requires additional functions to be provided from the application. For this purpose Xylon provides **SwConfig\_X.c** configuration templates, described in 5.2.

### 5.1 Linking Xylon logicBRICKS Drivers from the Application

The "drivers" entry should be added in SW application project's property box C/C++ Build settings->gcc linker->Libraries (Fig. 5.1).

NOTE: application settings for Debug and Release build configurations have to be set separately.



5.1 Add Option to Link libdrivers.a in the Application

#### 5.2 The SwConfig configuration files for the xyl\_oslib library

SwConfig files were introduced in the xyl\_oslib library because there is an often need for additional configurations on a per-application basis. Some software applications use the Xilinx Xilkernel and link with it in RAM memory, while others with a very small memory footprint, use Xilinx BSP without the Xilkernel and link in on-chip Block RAM (BRAM) memory. Instead of providing several specialized instances of the xyl\_oslib library, Xylon provides this configuration at the software application level.

The SwConfig.c file has to implement the following functions (see xyl\_oslib\_vX\_y\_a\src\possix\common\inc\SwConfig.h):

Copyright © Xylon d.o.o. 2012 All Rights Reserved





#### January 9, 2012.

Application Note: 0031

Version: v1.00

OsSpinLockT \*swConfig\_spinLockConstruct(OsObjectHandleT hParent);

- void swConfig\_spinLockDestruct(OsSpinLockT \*pLock);
- void swConfig\_spinLockAcquire(OsSpinLockT \*pLock);
- void swConfig\_spinLockRelease(OsSpinLockT \*pLock);

Xylon provides several SwConfig.c template files with the library:

- xyl\_oslib\_vX\_y\_a\src\possix\common\src\swconfig\_template\SwConfig\_xilkernel.c
  - $_{\odot}\,$  to be used when the application uses the Xilinx Xilkernel
- xyl\_oslib\_vX\_y\_a\src\possix\common\src\swconfig\_template\**SwConfig\_bsp.c** 
  - $_{\odot}$  to be used when the application uses the Xilinx BSP (no Xilkernel)
- xyl\_oslib\_vX\_y\_a\src\possix\common\src\swconfig\_template\**SwConfig\_no\_heap.c** 
  - to be used when the application uses the Xilinx BSP (no Xilkernel) and the available memory is very small (redefined malloc/free to some simple implementation)

User can modify existing templates to fit system requirements.

### 5.3 Setting up the SwConfig Files Required by the xyl\_oslib Library

Xylon provides template SwConfig.c files in the following folder: xyl\_oslib\_vX\_y\_a\src\possix\common\src\swconfig\_template

The SwConfig files setup:

- 1. Setup the application project build settings to compile one of the template SwConfig.c files located in the xyl\_oslib
- 2. Copy one of the template SwConfig.c files from xyl\_oslib folder to the application project folder. Setup application project build settings to build this local file
- 3. If the provided template SwConfig.c files miss needed functionality, copy the template file and make modifications. Setup the application project build settings to build the modified SwConfig.c file





January 9, 2012.

Application Note: 0031

Version: v1.00

## 6 Making Application Project with Xylon Device Driver Examples

All Xylon logicBRICKS device drivers are delivered with example source files, which are located in the \${DRIVER\_NAME}/examples folder. The examples are typically tested on specific hardware platforms and include hardware platform dependent and independent code.

NOTE: Platform dependent parts of the driver example may not be directly applicable to other targeted platform and driver's source code changes may be necessary.

NOTE: Xylon provides a platform library that supports several hardware platforms manufactured by Xylon and Xilinx. The provided device driver examples can be used on already supported hardware platforms with no modifications.

#### 6.1 Example Project Setting Up

The logiCVC-ML Compact Multilayered Video Controller IP core is an advanced display graphics controller for LCD and CRT displays, which enables an easy video and graphics integration into embedded systems with the Xilinx FPGAs. This IP core is a typical example of logicBRICKS IP cores that come with the software device driver and here presented information applies to other logicBRICKS IP cores as well.

The logiCVC-ML IP core's deliverables include the device driver, which allows for an easier work with the display graphics, with the following files provided in the examples folder:

- 1. logiCVC\_main.c (hardware platform dependant )
- 2. logiCVC\_demo.h (hardware platform independent)
- 3. logiCVC\_demo.c (hardware platform independent)

logiCVC\_main.c file initializes the hardware platform and calls functions from the logiCVC\_demo.c.

To make an application project with the video controller IP core's device driver example:

- 1. Include the platform independent part (logiCVC\_demo.c) of the example code in the project.
- 2. Modify the platform dependent part (logiCVC\_main.c) of the example to match used hardware platforms, and include it to the project. Code modifications are not required if Xylon already supports the selected hardware platform.
- 3. Add the "-Idrivers" in the application project linker properties (see 5.1)
- 4. Add the SwConfig.c file in the application project build settings (see 5.3)





January 9, 2012.

Application Note: 0031

Version: v1.00

# 7 Xyl\_oslib - In More Detail

#### 7.1 General Idea Behind the xyl\_oslib Library

Device driver for specific hardware is usually written multiple times for different operating systems and communication busses which are used to access the hardware.

The idea behind the Xyl\_oslib library creation can be demonstrated by an example of two hardware blocks, the UART and the I2C IP cores, which should work with two different operating systems (Linux and Microsoft® Windows® CE) and two different communication busses (PCI and USB bus).

There are 8 HW/SW combinations that should be supported by separated device drivers:

- 1. UART, Linux OS, PCI bus
- 2. UART, Linux OS, USB bus
- 3. UART, Microsoft WCE OS, PCI bus
- 4. UART, Microsoft WCE OS, USB bus
- 5. I2C, Linux OS, PCI bus
- 6. I2C, Linux OS, USB bus
- 7. I2C, Microsoft WCE OS, PCI bus
- 8. I2C, Microsoft WCE OS, USB bus

Device driver writing is an expensive process, and in order to increase efficiency, implement optimal code and decrease work efforts and costs, Xylon programmers isolate common drivers' elements.

10 separated software modules can be isolated from the above example requiring 8 different HW/SW combinations:

- 1. Linux I2C driver
- 2. Linux UART driver
- 3. Microsoft WCE I2C driver
- 4. Microsoft WCE UART driver
- 5. I2C generic part of driver
- 6. UART generic part of driver
- 7. Microsoft WCE PCI bus interface
- 8. Microsoft WCE USB bus interface
- 9. Linux PCI bus interface
- 10. Linux USB bus interface

The number of isolated modules is greater than a number of the planned HW/SW configurations, but at the end, this way of device driver partitioning adds value through reusability:





January 9, 2012.

Application Note: 0031

Version: v1.00

1 - 4 are OS dependant and require upper device driver's layer to be written for each new OS. These driver's layers are not reusable. They are only wrappers and require relatively small working efforts from designers

5 and 6 are generic hardware drivers that describe the hardware (IP core) functionality. The IP core generic drivers can be reused during driver porting to new OS/BUS combination.

7 and 8 are bus access interface implementations for the Microsoft WCE OS. This part is common for all drivers that will be written on Microsoft WCE and can be reused.

9 and 10 are bus access interface implementations for the Linux OS. This part is common for all drivers that will be written on Linux and can be reused.

#### 7.2 Layered Device Driver's Architecture

Xylon logicBRICKS device drivers implement the layered architecture to leverage reusable portions of the drivers' code. The driver layers are:

1. OS driver (OSDRV)

OS dependant part of the driver, which is specifically made for the targeted OS and the targeted bus interface (i.e. Linux driver for UART working with the USB bus). The OSDRV is usually just a small code wrapper required by OS for a particular type of driver. Main driver functionality should reside in the next layer.

2. Generic driver (GENDRV)

Generic part of the driver that describes the hardware functionality, i.e. the generic part of the UART driver defines UART registers and common functions for usual UART operation. The generic driver is completely independent from the OS and the bus interface.

3. OS library (OSLIB)

A library written for the specific OS system that provides a common interface which generic drivers can use with any OS/bus combination. It provides the interface to the specific OS types, synchronization mechanisms, bus access mechanisms, and other.

NOTE: Driver layers are separated only in the source code format. Compiled and linked driver layers make a single binary file.

The layered device driver's architecture minimizes the amount of required work for driver porting to new OS/bus combinations:





January 9, 2012.

Application Note: 0031

Version: v1.00

- 1. For each OS, the OSLIB has to be written only once
- 2. For specific HW platform, the generic GENDRV has to be written only once
- 3. OSDRV has to be written for each new OS/bus combination

#### 7.3 Xyl\_oslib Interface

Xylon OS library (xyl\_oslib) provides interfaces to all OS specific functions required by the IP core's device driver and enables logicBRICKS generic device drivers (GENDRV) to remain completely OS and bus interface type independent.

The library provides the following interface groups:

- OS types it provides typical types (pointers, unsigned 8, 16, 32, etc.). See Oslibtypes.h
- OS debug provides debug macros for printing, asserting. See OslibDebug.h
- OS HW access provides access functions to hardware register. Hardware access is independent from the bus interface type. Bus independency is obtained by pointer to struct \_OsHwAccessObject, which is an argument to all access functions. See OslibHwAccess.h.
  - oFunctions OsRegRangeInit and OsRegRangeDeinit provide init/deinit interface for the bus access descriptor.
  - oOsHwResourceT represents the hardware resource input structure
  - struct \_OsHwAccessObject represents the OSLIB descriptor for accessing hardware registers
- OS spinlock -provides interface for spinlocks, see OslibSpinLock.h
- OS misc provides interface for miscellaneous OS functions, see OslibMisc.h
- OS Time provides interface to time functions, see OslibTime.h

#### **Revision History**

Version	Date	Note
1.00.a	9.1.2012.	Initial Xylon release



Xylon d.o.o. Fallerovo setaliste 22, 10000 Zagreb, Croatia www.logicbricks.com All other trademarks and registered trademarks are the property of their respective owners.