

Command Line Tools Quick Start Guide

Version 9.9



Publication Date: 2009/09/24

Copyright © 2009 XMOS Ltd. All Rights Reserved.

1 Introduction

This document provides a short introduction to the XMOS Command Line Tools and how to use them to write programs for XMOS development kits.

1.1 System Requirements

The Tools are supported on the platforms listed in the table below.

Windows	XP SP2 (32-bit with 32-bit JRE)
Linux	RHEL 4.6 (32-bit with 32-bit JRE) RHEL 4.6 (64-bit with 64-bit JRE)
Mac	OS X 10.5 (Intel)

You must have a Java Runtime Environment (JRE) version 1.5 or later installed, which can be downloaded from the Sun website.¹

The Tools are known to work on other variants of Linux. Please check the XMOS website² for a list of known compatibility issues.

The Tools in the Linux 64bit distribution require the 32bit compatibility layer libraries to be installed. For further information on installing the required libraries see the Red Hat Support website³ or the documentation provided with your distribution.

¹<http://java.sun.com/javase/downloads>

²<http://www.xmos.com/support>

³<http://www.redhat.com/support>

2 Install the Tools

This section explains how to install the Tools for the supported platforms.

2.1 Windows Users

1. Download the tools as an executable installer from:

<http://www.xmos.com/downloads>

2. Double-click the downloaded installer to execute it, and follow the on-screen prompts to install the tools.

The installer adds a *Desktop Tools Prompt (version)* shortcut to the *Desktop* and *Start* menu.

3. Select Start>Programs>XMOS>XMOS>Desktop>Tools>*version*> Desktop Tools Prompt (or double-click the Desktop Tools Prompt desktop icon).
4. Use the open command line prompt to run the tool you want to use. See [Section 5](#) for further details on the compiler, debugger and simulator.

2.2 Linux Users

1. Download the tools as an archived package from:

<http://www.xmos.com/downloads>

2. Unarchive the **DesktopTools_<version>.tgz** package to your install directory using the following command:

```
tar -xzf DesktopTools_<version>.tgz -C /home/user
```

3. Go to the installation directory and run:

```
source SetEnv
```

4. Use the open Terminal window to run the tool you want to use. See [Section 5](#) for further details on the compiler, debugger and simulator.

2.3 Mac OS X Users

1. Download the tools as a Macintosh DMG installer from:
<http://www.xmos.com/downloads>
2. Double-click the downloaded DMG file to open it, and then drag the XMOS icon into your Applications folder.
3. When the tools have been installed, eject the DMG file (CMD+E) and drag it to the Trash to delete it.
4. Go to the installation directory in Finder and run:
`SetEnv.command`
5. Use the open terminal window to run the tool you want to use. See *Section 5* for further details on the compiler, debugger and simulator.

3 Connect a development kit

The Tools interact with a hardware device over a JTAG interface. XMOS development kits include USB-to-JTAG connectors for interfacing with a host PC. You may have to install the necessary USB drivers to communicate with your development board.

3.1 Install USB drivers for Windows

1. Connect the development kit to your development system using the USB cable provided with your development kit.
2. If prompted, follow the *New Hardware* instructions on your computer screen. When Windows prompts you, select *Yes this time only to Connect to Windows Update*, and *Next* on the following screens. There should be two hardware devices and one serial port device to install if none of the drivers have been installed previously.

NOTE: The Tools installer includes a set of drivers that you can use instead of using Windows Update, but they are not guaranteed to be the latest version of the drivers. They are copied to the *Desktop Tools/Drivers* directory during installation.

3.2 Install USB drivers for Linux

1. Connect the development kit to your development system using the USB cable provided with your development kit.

2. Log into a shell with root permissions.
3. Open the file `/etc/fstab` and add the line:

```
none /proc/bus/usb usbfs defaults,devmode=0666 0 0
```
4. Unmount the USB file system, for example:

```
umount /proc/bus/usb
```
5. Remount the USB file system, for example:

```
mount /proc/bus/usb
```
6. Log out from root access.

3.3 Install USB drivers for Mac OS X

Mac OS X contains the hardware drivers required to use the USB device. No additional configuration is required.

4 A "Hello World" program

This section shows you how to use the command line compiler to compile a simple "Hello World" program and run it on the XMOS simulator or development kit.

Open a text editor, copy the following program to a file and save this file as `helloworld.xc` to the Tools installation folder.

```
#include <stdio.h>

int main(void)
    puts("Hello World!");
    return 0;
```

4.1 Compile the program

To compile your program, type:

```
xcc helloworld.xc -o hw.xe
```

The `-o` option causes the compiled program to be output to the file `hw.xe`. If this option is omitted, the program is output to `a.xe`.

4.2 Run the program on the XMOS simulator

To run the program on the XMOS simulator, type:

```
xsim hw.xe
```

The simulator should print the text "Hello World!" in standard output.

4.3 Run the program on your XMOS development kit

To run your program on an XMOS development kit, type:

```
xrun --io hw.xe
```

The `--io` option runs an I/O server after loading the program, which communicates with the hardware to enable system calls (as required by the function `puts`). The message "Hello World!" should be printed in standard output.

5 Command line tools

The following section provides a quick reference to some of the more frequently used options to the command-line tools.

A list of all supported options can be obtained for each tool using the `--help` option. For example, the following command provides a list of options for the XMOS compiler:

```
xcc --help
```

Supported options for each tool are also listed in the XMOS *Tools User Guide* [1].

5.1 XCC

XCC is the XMOS compiler. It supports programs written in C, C++, XC and ASM.

Files may be compiled individually and the generated ELF objects subsequently linked:

```
xcc -c file01.xc
xcc -c file02.c
xcc file01.o file02.o file03.xc
```

The `-c` option compiles the source file into a linkable object; the linker is not invoked to generate an executable.

The `-Wall` option instructs the compiler to make additional checks for errors:

```
xcc -Wall file.xc
```

The `-g` option causes the compiler to generate debug information that uses for symbolic debugging:

```
xcc -g file.xc
```

The `-O1`, `-O2`, `-O3` options turn on compiler optimisations that improve performance and/or code size:

```
xcc -O2 file.xc
```

The `-I` option adds a directory to the list of directories to be searched for header files:

```
xcc -I ./headers file.xc
```

5.2 XGDB

is the XMOS debugger. It provides symbolic debug support for programs written in C, C++, XC and ASM.

Programs must be compiled by XCC with the `-g` option to enable all debug features, for example:

```
xcc -g file.xc -o file.xe
```

To start with a target executable, type:

```
xgdb file.xe
```

To connect to the hardware device, type:

```
(gdb) connect
```

Alternatively, to connect to the simulator, type:

```
(gdb) connect -s
```

To load the executable into the device memory, type:

```
(gdb) load
```

All commands can now be used to debug the program. For example, to run until the program reaches `main` type:

```
(gdb) break main  
(gdb) continue
```

5.3 XSIM

XSIM is the XMOS cycle-based simulator, which simulates XE files generated by the compiler XCC.

To simulate a program, type:

```
xsim file.xe
```

To simulate a program with instruction tracing, type:

```
xsim -t file.xe
```

5.3.1 Trace instructions

When instruction tracing is enabled, XSIM outputs a textual instruction level trace on standard input. The format of the trace is as follows:

```
NID@PID@TID FLAGS TSTATUS.ATSR.PC : INST @CYCLE
```

NID	Node number
PID	Processor number
TID	Thread number
FLAGS	<p>D = Instruction caused debug interrupt * = Instruction excepted P = instruction paused d = XCore running in debug mode</p>
TSTATUS	<p>Status of all threads. Two characters per thread in use. First character shows thread status. If the a or i is capitalised (A / I) this thread executes the instruction being traced.</p> <p>- = not in use a = active i = active with ININT bit set m = paused with MSYNC bit set s = paused with SSYNC bit set w = paused with WAITING bit set p = paused for instruction fetch</p> <p>Second character shows the thread has the following attribute:</p> <p>- = Interrupts and events disabled b = Interrupts and events enabled i = Interrupts enabled, events disabled e = Events enabled, interrupts disabled</p>
ATSR	<p>Active thread's Status Register bits</p> <p>MSYNC bit = m set, - not set SSYNC bit = s set, - not set INK bit = k set, - not set INENB bit = n set, - not set</p>
PC	Hexadecimal value of active thread's program counter
INST	Instruction description
CYCLE	Processor cycle count

6 Further Reading

Additional information can be found in the following documents:

- Programming XC on XMOS Devices [2].
- XMOS Tools User Guide [1].
- GDB Manual [3].
- The XS1 Architecture [4].

See also:

- <http://www.xmos.com/documentation>

Bibliography

- [1] Douglas Watt and Huw Geddes. *The XMOS Tools User Guide*. XMOS Limited, 2009. http://www.xmos.com/published/xtools_en.
- [2] Douglas Watt. *Programming XC on XMOS Devices*. XMOS Limited, Sep 2009. http://www.xmos.com/published/xc_en.
- [3] The Free Software Foundation. Debugging with GDB. Website, 2008. <http://www.xmos.com/published/xgdb87>.
- [4] David May. *The XMOS XS1 Architecture*. XMOS Limited, 2009. http://www.xmos.com/published/xs1_en.

Disclaimer

XMOS Ltd. is the owner or licensee of this design, code, or Information (collectively, the “Information”) and is providing it to you “AS IS” with no warranty of any kind, express or implied and shall have no liability in relation to its use. XMOS Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.