# xTIMEcomposer Studio Tutorial

IN THIS DOCUMENT



## 1 Introduction

Welcome to xTIMEcomposer. This tutorial provides an introduction to xTIMEcomposer Studio and xSOFTip Explorer. It shows you how to:

- Create an application using xSOFTip Explorer
- Create a project from xSOFTip using xTIMEcomposer
- Edit a xSOFTip component in xTIMEcomposer
- Test your program in the XMOS simulator

We recommend that you follow the tutorial step-by-step. If you need to download the source code for the examples discussed it is available from xmos.com xTIMEcomposer Tutorial Code Examples[1].

## 2 The xSOFTip Explorer Perspective

xSOFTip components use xCORE resources to provide interfacing, DSP, protocols and control functions, allowing you to concentrate on building your application.

1. Select **Window > Open Perspective > XMOS xSOFTip Explorer** to open the xSOFTip Explorer perspective.

The *xSOFTip Explorer* Perspective allows you to browse the XMOS xSOFTip components and select them for use in your system. It provides resource information so you know which XMOS multicore microcontroller is most suitable for your application.

The xSOFTip Explorer Perspective has four windows:

[1] http://www.xmos.com/published/xtimecomposer-studio-tutorial-code?version=latest

Publication Date: 2013/5/29 REV B

- **xSOFTip Browser** – shows the xSOFTip components you can chose for your project
- **My System Configuration**: the xSOFTip components you have selected
- **System Information**: the resources used by the xSOFTip components you have selected, and the XMOS multicore microcontrollers which suit your application
- **Developer Column**: online documentation about the xSOFTip, tools and xCORE multicore microcontrollers

**NOTE**: Additional documentation will be loaded in other tabs in the *Developer Column*. Tabs are located at the bottom of the *Developer Column*. You will need to switch between the *xTutorial* tab and the *Main* tab to see the xSOFTip documentation.

### 2.1 xSOFTip Component Scope

Each xSOFTip component is categorized with a *Scope*, which shows the status of the xSOFTip component:

- **General Use**: The xSOFTip consists of a complete release from XMOS.

  Complete resource information is available.

  NOTE: All attempts have been made to ensure the correct functionality of this block, but the final quality of any product using this block is the responsibility of the user.
- **Early Development**: The xSOFTip is suitable for use in development of products and is fully functional. However, the maturity of the software is such that extra care must be taken in verifying a product using this software block. Resource information is available.
- **Experimental**: The xSOFTip is at an experimental/prototype stage. Code exists but is not feature complete. Resource information may be available.
- **Roadmap**: The xSOFTip is on the XMOS development roadmap. Estimated resource information exists for this xSOFTip, but no code is available.
- **Open Source Community**: The xSOFTip has been developed by the Open Source community. Resource information may not be available.

## 3 Your first application

The *xSOFTip Browser* displays all available xSOFTip components including hardware interfaces, control functions and DSP processing. This section shows how to use xSOFTip to implement a precise PWM driver that uses the real-time capabilities of xCORE.

### 3.1 Add the PWM to your application

You can add xSOFTip components directly to your application using the *xSOFTip Explorer* perspective.

1. Click on the **Tutorial Example LED PWM Driver** xSOFTip component in the *sliceKIT/demos* category.

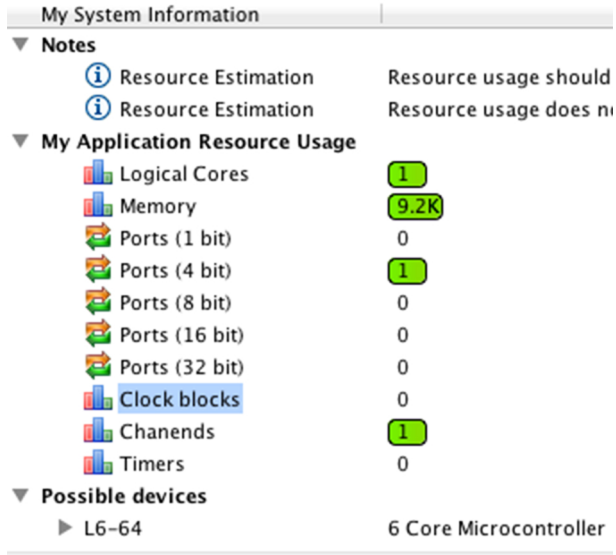
   The *Developer Column* shows information on the component including a description of what it does, its features and which xKIT Development Kits are suitable for use with this xSOFTip.

2. Drag the **Tutorial Example LED PWM Driver** xSOFTip into the *My System Configuration* window.

   All peripherals in XMOS are implemented using software, giving you complete freedom to customize the interface to meet your exact requirements.

   xSOFTip is all delivered as C code, so you easily change it to meet your exact requirements. You can also take existing C functions and run them on an xCORE. For interfacing to I/O pins and for communicating between logical cores, XMOS has added a handful of operations to C, called 'XC'.

3. The *My System Information* window is updated to show the resources used by your system configuration.




Resources include:

- **Logical Cores**: 32bit microcontroller cores. XMOS multicore microcontrollers include 4, 6, 8, 10, 12, 16 and 32 core devices.
- **Ports**: I/O pins of XMOS multicore microcontrollers are connected to *ports*, which allow your software to send and receive data to the pins with extremely low latency. Ports are available in different widths: a 1-bit port is connected to 1 I/O pins, a 4-bit port is connected to 4 I/O pins.
- **Clock Blocks**: Clock blocks are used to precisely control timing of I/O pins.
- **Chanends**: Channel Ends are part of the xConnect system, allowing the logical cores to send messages to each other through low latency xConnect channels.

- **Timers**: Timers are used by the software to control the time at which things happen. They run at 100MHz, giving 10ns precision.

A list of *Possible Devices* is displayed at the bottom of the *My System Information* window. This shows the xCORE multicore microcontrollers that most are suitable for this application.

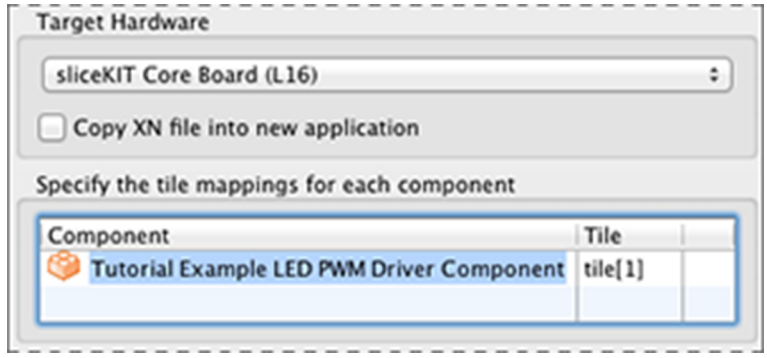
# 4 Creating a project from the xSOFTip

When you create a project from the *xSOFTip Explorer* perspective in xTIMEcomposer Studio, an example instantiation of your selected xSOFTip is added to a new project.

A `main()` function is created, to which you can then easily add your application code.

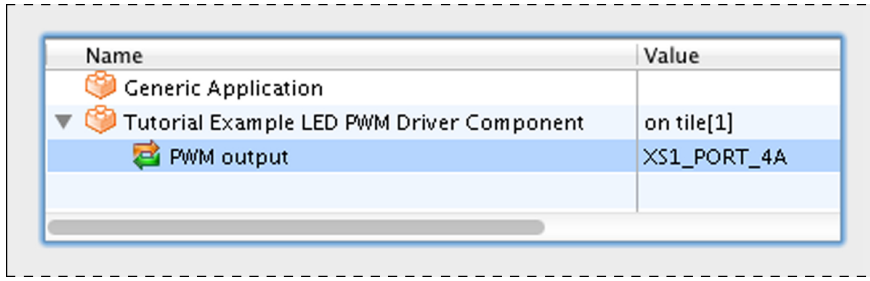
## 4.1 Create a project

1. Click the **Generate Project** button at the top of the *My System Configuration* window.
2. Enter a name for your project in the *Generate Project* window, for example **PWM**.
3. Select **sliceKIT Core Board (L16)** from the *Target Hardware* list for your project.
4. Select **tile[1]** for the *PWM xSOFTip* in the *Specify the tile mappings . . .* control.




5. Click **Next**.
6. Select XS1_PORT_4A as the port you want to use to drive the LEDs.

   The GPIO Slice Card has 4 LEDs connected to XS1_PORT_4A.




7. Click **Finish**.

   xTIMEcomposer Studio generates a project with your selected xSOFTip.

# 5 Using xSOFTip in the Edit perspective

xTIMEcomposer Studio changes to the *Edit* perspective when it creates a project ready for you to edit the code. You can switch between perspectives at any time using the **Window > Open Perspective** menu.

This section shows you how to edit the xSOFTip project to create a simple application that varies the PWM duty cycle.

## 5.1 Editing the xSOFTip code

1. Open the PWM project in the *Project Explorer* and double-click on `main.xc` to open it in the *Editing* window.

   The `main()` function created by xTIMEcomposer is displayed.

   The `par` statement is used to instantiate a Core. Each function or statement in a `par` statement is run on a different core. In this example, two cores will be specified, one core to run the `pwm_controller()` task and another to run the *xSOFTip PWM driver*.

   `main()` has already instantiated your PWM xSOFTip, so it will run on one logical core on Tile 1. Now add our own function to run on another core also on Tile 1.

2. Add the code below into the `par` statement to instantiate a pwm_contoller logical core.

```
on tile[1]: {
  pwm_controller(c_pwm_duty);
}
```

   The `on tile[1]` statement is used to specify which tile the processing cores are on. Each tile in an xCORE multicore microcontroller has eight logical cores. In this example you will use cores on tile 1.

3. Create a new pwm_controller task above `main()` in main.xc, that will run on your core using the following code:

```
void pwm_controller(chanend c_pwm)
{
}
```

The task needs a chanend (channel end) so that it can communicate with the PWM Driver. Channel ends are part of the xConnect communication system, allowing the logical cores in a multicore microcontroller to communicate with each other with low latency.

xTIMEcomposer has already created a channel for you: `chan c_pwm_duty`. Each channel has two chanends, allowing two logical cores to communicate with each other. The PWM Driver has already been given `c_pwm_duty` as one of its arguments (look in `pwm_tutorial_example.xc` to see the function definition).

From the PWM xSOFTip documentation:

```
The PWM component uses 1 Core, with
a channel interface to the rest of
the application.
The client application sends two
values over the channel to configure
the PWM driver:
1. The PWM period length
2. The PWN duty cycle length
All times are measured with the 100MHz
reference clock. For example, a value
of 100 is 100 x 10ns = 1us.
```

In this case we want to configure the PWM with a low time of 5us and high time of 5us. Therefore you need to send it a value of 1000 for the period length and 500 for the duty cycle length.

Data is sent over the channel `c_pwm` using the the <: XC operator.

4. Add the following code to your `pwm_controller` task:

```
// send the PWM period length
c_pwm <: 1000;

// send the PWM duty cycle length
c_pwm <: 500;
```

The application is now complete and ready to be compiled.

## 5.2 Building your project

1. Select PWM in the *Project Explorer*.

2. Click **Project > Build Project**

   The Console shows the results of the compilation, together with any error messages. Your Console should show that the build completed correctly.

3. Check the *bin* folder in the *Project Explorer*.

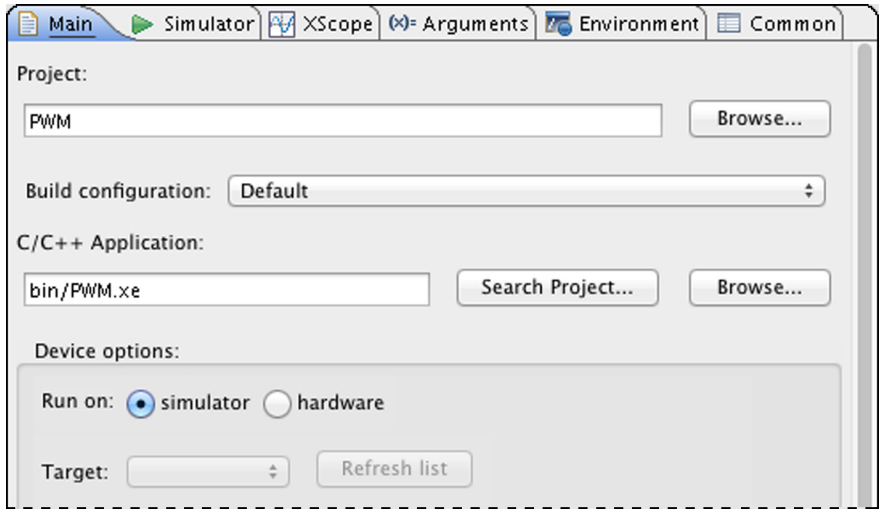
You now have a binary (PWM.xe) that you can execute.

# 6 Test your project in the simulator

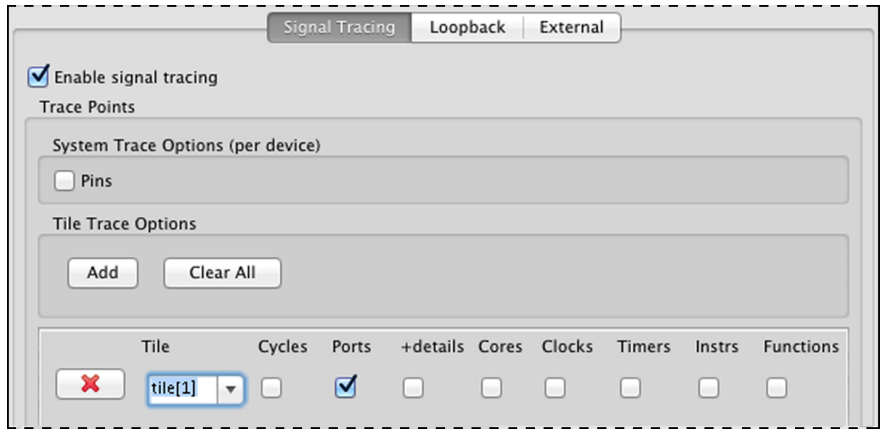
xTIMEcomposer Studio includes a simulator which allows you to simulate your application without hardware. The simulator includes a waveform analyzer, that you can use to view the I/O pin driven by your PWM xSOFTip.

## 6.1 Run your application in the Simulator

1. Select **Run > Run Configurations**.
2. Double-click on the **xCORE Application**. This creates a new Run Configuration, automatically filling in the required options.
3. Select **Run on: Simulator**




4. Click the *Simulator* tab.
   You need to enable tracing of the ports so that you can see the I/O pin behavior.
5. Select **Enable Signal Tracing**.
6. Click **Add** under *Tile Trace Options*.
7. Select **tile[1]** and tick the **Ports** option.

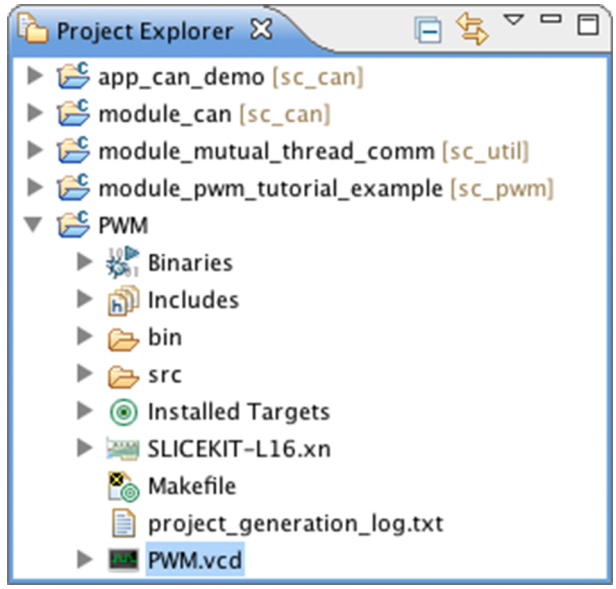



8. Click **Run**.

   The application runs on the Simulator. A red **Stop** button appears in the *Console* toolbar.

9. Let the program run for about 10 seconds, then click the red **Stop** button.

   A PWM.vcd file is added to the *Project Explorer*.




The next section shows how to look at the waveform of your PWM driver.

# 7 The xTIMEcomposer Waveform Viewer

xTIMEcomposer Studio contains a waveform viewer that you can use to look at the waveform of the PWM signal.

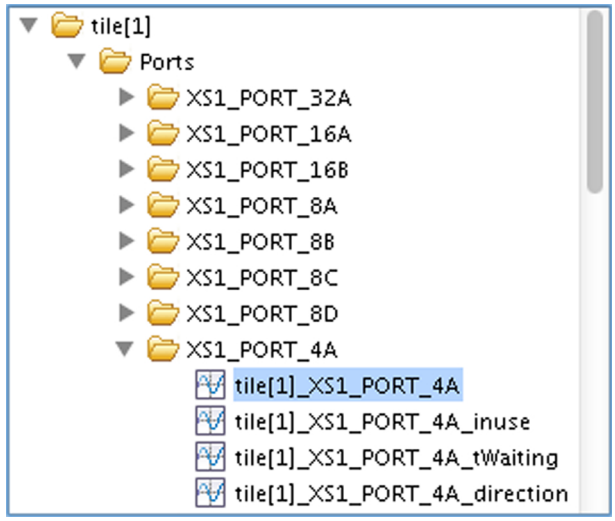
## 7.1 Using the waveform viewer

1. Double-click on **PWM.vcd** in the *Project Explorer*.

   The *Waves* window appears in place of the *Console*.

   The *Signals* window appears next to the *Project Explorer*. You can use this to select which ports to show in the *Waves* window.

2. Browse to the XS1_PORT_4A port (the port you selected for our PWM port), open the folder and double-click on **tile[1]_XS1_PORT_4A**.




3. The *Waves* window shows the value of the pins for our PWM signal. You may need to zoom out to view the PWM transitions.

   The PWM driver uses the deterministic, real-time capabilities of xCORE to generate a precise PWM signal. The waveform viewer shows a timeline so you can measure the PWM output.

4. Verify the timing of your PWM interface by measuring the period length and duty cycle length of your PWM signal.

   You can zoom in and out to see the signals with an appropriate timescale. Click on a signal to place a Marker, then move the cursor to view the time difference between the Marker position and the cursor position.

5. Double-click on a transition in the Waves window when the cursor changes to a pointing finger.

The output statement that caused the transition is highlighted in the *Editor* window, allowing you to link the transition to the source code.

Thanks to its timing deterministic architecture, xCORE multicore microcontrollers provide guaranteed response times up to 100x faster than conventional microcontrollers. xTIMEcomposer Studio includes the XMOS Timing Analyser (XTA), a tool for analyzing your application and telling you precisely how long your code will take to execute.

# 8 Next steps

Congratulations you have now completed this simple xTIMEcomposer tutorial and are ready to run it on an xCORE multicore microcontroller. We aim to make evaluating and development with our xCORE multicore microcontrollers as easy as possible by offering a range of development kits to meet your specific needs.

## 8.1 Buy a development board

Take a look at our xKITS at http://www.xmos.com/discover/products/xkits[2].

If you're not sure which to choose, we recommend our sliceKIT Starter Kit[3].

## 8.2 Try the sliceKIT Development Board tutorial

If you have a sliceKIT starter Kit we recommend that you follow the sliceKIT Development Board Tutorial, which shows you how to run the project you created in this tutorial on a hardware board. It also shows you how to use the real-time features of xCORE multicore microcontrollers to extend the PWM application to create a temperature controlled LED dimmer.

See **Help** > **Tutorials** > **sliceKIT Development Board Tutorial**.

## 8.3 Browse xSOFTip

Take a look at the other xSOFTip components. You can read through the documentation in *Developer Column*, and use them in your project.

XMOS and our partners are working on new xSOFTip components all the time. Some components you'll see are Roadmap components which are in our development plan. If there is a component you require for your system that is not available, please let us know – we'd love to hear from you.

## 8.4 Try some other tutorials

We provide a range of tutorials covering the xTIMEcomposer tools, xSOFTip and xKIT development boards. See **Help** > **Tutorials** for more information.

[2] http://www.xmos.com/products/xkits
[3] http://www.xmos.com/products/xkits/slicekit#slicekit-starter-kit