

**TILERA**<sup>TM</sup>  
EMBEDDING MULTICORE

# **Multicore Development Environment**

## **Getting Started Guide**

REL. 1.1  
DOC. No. UG204  
EZChip Semiconductor

Copyright © 2020 EZChip Semiconductor Corporation. All rights reserved. Printed in the United States of America.

The following are trademarks of EZChip Semiconductor Corporation: EZChip, the Tiler Logo, Tile Processor Architecture, Tile Processor, and iMesh Multicore. All other trademarks and/or registered trademarks are the property of their respective owners.

This document contains advance information on products that are in development, sampling or initial production phases. This information and specifications contained herein are subject to change without notice at the discretion of EZChip Corporation.

The following third-party trademarks or registered trademarks may appear in this text:

No license, express or implied by estoppels or otherwise, to any intellectual property is granted by this document. EZChip disclaims any express or implied warranty relating to the sale and/or use of EZChip products, including liability or warranties relating to fitness for a particular purpose, merchantability or infringement of any patent, copyright or other intellectual property right.

Products described in this document are NOT intended for use in medical, life support, or other hazardous uses where malfunction could result in death or bodily injury.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. EZChip assumes no liability for damages arising directly or indirectly from any use of the information contained in this document.

**Publishing Information:**

Document number:	UG204
Release	1.1

Contact Information:

---

**EZChip Semiconductor  
Corporation**  
Information [info@ezchip.com](mailto:info@ezchip.com) Web Site:  
<http://www.tilera.com>

**Headquarters**

---

4677 Old Ironsides Dr., Suite 310  
Santa Clara, CA 95054 USA  
+1 408-654-7630 Phone  
+1 408-654-7636 Fax

# Contents

## **INTRODUCTION 1**

Related Documents .....	1
System Requirements .....	2
Unpacking and Handling.....	2

## **INSTALLING THE TILEXPRESS-64 CARD AND SOFTWARE 3**

Installing the TILExpress-64 Card .....	3
Caution! Air Flow .....	4
Jumpers and DIP Switches .....	4
Connecting Cables .....	5
Installing the Software.....	5
Verifying the Installation .....	6

## **OPENING THE MDE WORKBENCH 8**

MDE Perspectives .....	9
------------------------	---

## **BUILDING SAMPLE PROJECTS 10**

Building Sample Projects Using the IDE .....	10
IDE Project Types .....	10
Launch Configuration .....	10
Creating and Building a Project.....	10

## **RUNNING AND DEBUGGING SAMPLE PROJECTS USING IDE 15**

Creating Breakpoints .....	15
Creating a Launch Configuration .....	15
Debugging the Sample Project .....	17
Viewing Output.....	19
Building and Running Sample Projects Using the Command Line Interface .....	21
Setting the Path .....	21
Running the Sample Programs on the Simulator .....	21
Running the Sample Programs on the TILExpress-64 Card .....	22

## **GETTING HELP 24**

Frequently Asked Questions .....	24
----------------------------------	----



# INTRODUCTION

The Multicore Development Environment (MDE) is a collection of standard software development tools that are optimized for use in programming the TILE64 processor. The MDE provides a familiar programming environment while enabling you to harness the compute power of the TILE64 multicore processor.

This Getting Started Guide is for software developers who need to install and configure the TILE Processor hardware or the MDE toolkit. It tells you how to

- Install the TILExpress-64 card board containing the TILE64 processor
- Install the MDE software
- Use either the IDE (GUI) or the command-line tools to build and run sample programs either on the built-in simulator or, optionally, on hardware

## Related Documents

The Tilera IDE is based on the Eclipse development environment. If you have not used the Eclipse development environment before, it would be a good idea to review an introductory user's guide, such as:

- *Eclipse Distilled*, by David Carlson
- The Workbench User Guide at the Eclipse Website
- The Workbench User Guide in Eclipse's online help. (Note: you may need to click the **Show in Contents** button on the help viewer toolbar to open the Workbench User Guide table of contents.)

Note that Eclipse, by itself, is a Java-centric development environment, so the examples in books on Eclipse will necessarily be Java-centric. You should read such introductory material with the goal of becoming comfortable with Eclipse as a GUI development environment, rather than as a guide for C programming in Eclipse.

In the top-level contents of the on-line help, aside from the Tilera MDE User Guide itself, there is also a top-level section on the C Development Toolkit (CDT) plug-in extensions, which covers C/C++ development using the CDT plug-in's features.

# System Requirements

System Requirements:

- Red Hat Enterprise Linux 4; x86\_64 version 4.4 is preferred
- Pre-built kernel drivers are provided for x86\_64 RHEL 4.4 Driver source is provided, other kernels will need to compile their own drivers
- GUI tools require version 2.2.1 or better of the GTK+ widget toolkit and associated libraries (GLib, Pango)

# Unpacking and Handling

**Caution!** When you unpack and handle the TILExpress-64 card, use standard electrostatic discharge (ESD) precautions.

# INSTALLING THE TILEExpress-64 CARD AND SOFTWARE

## Installing the TILEExpress-64 Card

### Board Requirements

The TILEExpress-64 card has a x4 PCIe interface and requires a x4 PCIe connector. Higher-bandwidth connectors are backward compatible; therefore, either a x8 or x16 connector will work. Tilerá has verified that these motherboards have the correct slot length and clearance for the TILEExpress-64 card:

- Intel® Desktop Board D975XBX2
- Dell Precision 490 Mini-Tower Dual Core Xeon Proc 5120 1.86GHz, 4MB L2 Cache, 1066MHz (222-3802)

### Board Installation

To install the TILEExpress-64 card:

1. Power down the PC and unplug the AC line cord.
2. Verify that you have a minimum of two adjacent empty board slots in your PC. The TILEExpress-64 card requires a PCI x 4 slot.

**Note:** Some motherboards have restricted slot usage. Refer to your motherboard documentation for this information.

3. Ensure that both the DIP switch and jumpers are set correctly. Refer to [Table 1](#) and [Table 2](#).
4. Install the TILEExpress-64 card so that the processor side and heat sink face an empty slot.



*Figure 1. TILEExpress-64 card Power Connector*

5. Plug one of the PC's available power connectors into the TILEExpress-64 card ([Figure 1](#)).

6. Close the PC.
7. Reconnect the AC power cord and power up the PC.

## Caution! Air Flow

The board **must** have proper air flow at all times. Ensure that the PC provides adequate airflow across the TILEExpress-64 card. **If you remove the cover of the PC, position an external fan so that it blows on the board.**

## Jumpers and DIP Switches

This section lists the TILEExpress-64 card jumpers and DIP switch and their default settings.

### Jumpers

[Table 1](#) lists the jumper plug settings.

*Table 1 Jumper Plug Settings*

Jumper ID	Reference Designator	Description	Default Setting
External 3.3V	J3053	Connect to a 3.3V supply in the event of standalone operation. Pin 1 = 3.3V, Pin 2 = GND.	
Fan	J3054	Connector for fan head for TILE64. Pin 1 = 12V, Pin 2 = GND	
RJC Jumper	J3044	Allows connecting the TILE64 RJC test pin to either HRESET_L or GND. Pin 1 = HRESET_L, Pin 2 = RJC, Pin 3 = GND	Jumper pins 1 and 2
JTAG TILE64 TDO	J3050	Used in conjunction J3049 to include TILE64 in JTAG chain, or isolate it. Install jumper to include it in the chain. See section 5.1.11.2 for details of JTAG isolation.	Jumper pin 1 to pin 1 of J3049
JTAG Chain TDO	J3049	Used in conjunction J3050 to include TILE64 in JTAG chain, or isolate it. Install jumper to include it in the chain. See section 5.1.11.2 for details of JTAG isolation.	Jumper pin 1 to pin 1 of J3050
TILE64 TCK clock	J3051	Connects 25 MHz clock for test purposes	
UART Header	J3052	Connector for UART cable. Pin 1 = TX, Pin 2 = GND, Pin 3 = RX.	



## DIP Switch

The TILExpress-64 card has one DIP switch. [Table 2](#) lists its settings.

**Table 2 DIP Switch SW7001 Settings**

Switch position	Description	Default Setting
1	Open – TILExpress-64 card is in normal, add-in card mode Closed – TILExpress-64 card is operating in Root Complex mode and drives the PERST# signal to the edge connector	Open - (TILExpress-64 card is not root complex)
2	Open – BOOT_PCIE1_EN is HIGH (boot from PCIE1) Closed – BOOT_PCIE1_EN is LOW	Open - Boot via PCIe
3	Open – BOOT_ROM_EN is HIGH (boot from ROM) Closed – BOOT_ROM_EN is LOW	Closed - Do NOT boot via ROM
4	Open – BOOT_ROM_SEL is HIGH (boot from I2C ROM) Closed – BOOT_ROM_SEL is LOW (boot from SPI ROM)	N/A: Only applies if booting via ROM

## Connecting Cables

Tilera provides a UART debug cable. You must supply your own Ethernet cables.

## Installing the Software

### Installing Linux

The Linux Red Hat installation guide is located at

<http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/x8664-multi-install-guide/>

### Installing the MDE Software

To install the MDE software:

1. Install the Evaluation Board as described in the previous section.
2. Open an X-term or terminal window.
3. Log in as `root`.
4. Insert the CD containing the MDE files into a CD drive.
5. Copy the tar file to `/opt`.
6. Go to the `/opt` directory. This directory must be cross-mounted so it is accessible from the PC where the TILExpress-64 card is installed, as well as from any development host on which the command-line or IDE tools will need to be run. (The two may or may not be the same, depending on your site.)
7. Unpack the archive file using the command

```
tar xzf TileraMDE-<version>.tar.gz
```

This installs a file hierarchy at

```
./TileraMDE-<version>/...
```

All file and directory names given below are relative to this directory.

**Note:** in the steps that follow, the root directory of your Tiler software installation is denoted as `${TILER_HOME}`

#### 8. Set the path:

```
export.PATH/opt/TilerMDE-1.1.0/bin:$PATH
#which tile-cc
[returns the path]
```

9. To run the examples in this Guide, add the MDE `${TILER_HOME}/bin` directory to the `PATH` environment variable.

## Installing the TILExpress-64 card Drivers

To install the drivers:

1. Log in as root and run the device driver installation script:

```
% ${TILER_HOME}/lib/modules/tile-module-install
```

**Note:** You do not need to reboot the PC after running the driver installation script. Installing the TILExpress-64 card enables the driver automatically.

2. Verify that the driver is installed and the board is available:

```
% grep tilepci /proc/devices
```

The file `/proc/devices` lists the currently available devices. If the board is installed and available the above command returns an entry for the board. To verify the path for the board, type:

```
% ls /dev/tilepci0/0
```

## Verifying the Installation

To verify that the installation has succeeded, use three versions of make.

First, type

```
${TILER_HOME}/bin cd examples/getting_started/hello_world
make
```

Doing this:

1. Compiles and links a sample program.
2. Runs a program on the Tiler simulator.
3. Compares the program's output against expected results.
4. Outputs `Hello World!` to verify success.

Next, type:

```
${TILER_HOME}/bin cd examples/getting_started/hello_world
make
```

This:

1. Compiles and links a sample program.
2. Runs a program on the Tiler simulator.
3. Outputs `Hello World!` to verify success or an error message for failure.

Next, type:

```
`${TILERA_HOME}/bin cd examples/getting_started/hello_world  
make test
```

Doing this:

1. Compiles and links a sample program.
2. Runs a program on the Tiler simulator.
3. Compares the program's output against expected results.
4. Provides either no output to verify success or an error message to denote failure.

Finally, type:

```
`${TILERA_HOME}/bin cd examples/getting_started/hello_world  
make test_pci
```

Doing this:

1. Compiles and links a sample program.
2. Runs a program on the hardware.
3. Compares the program's output against expected results.
4. Provides either no output to verify success or an error message to denote failure.

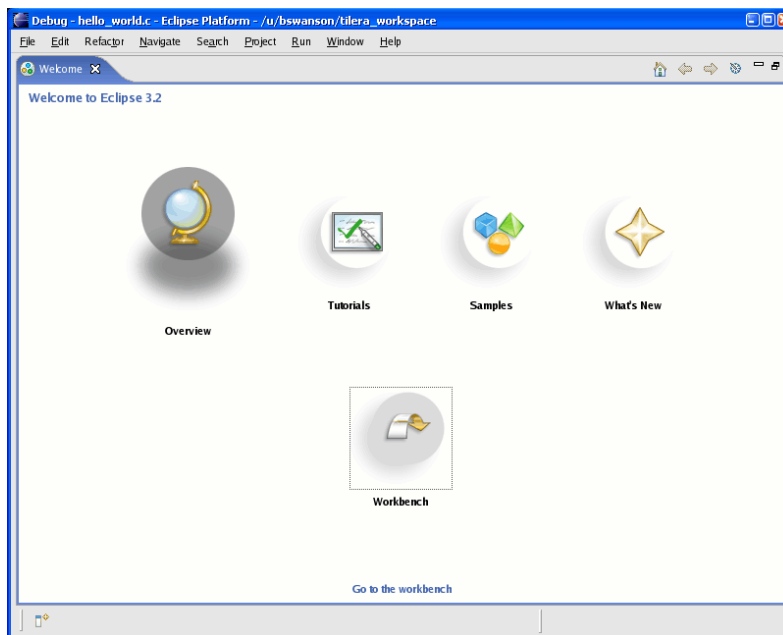
The MDE hardware, software, and board drivers are now installed and ready for use. For complete information on the MDE application, see the *MDE User's Guide*.

# OPENING THE MDE WORKBENCH

The Tiler MDE includes both command-line and IDE development tools. Tiler recommends that new users start with the Tiler IDE. This section describes how to start the IDE for the first time. To open the Workbench and start the Tiler IDE:

1. Open an X-term or terminal window.
2. Type `tile-eclipse`.

When you start the IDE for the first time you will briefly see the standard Eclipse welcome window (Figure 2). Background color and icon alignment may be different for your window.



*Figure 2. MDE Welcome Screen*

3. If the Welcome screen appears you'll see prompts telling you that the IDE is ready for use. Click **OK**.

The Tiler plug-in detects that you're running the IDE for the first time and automatically loads a number of perspectives, or named sets of GUI views, that will be useful to you in developing Eclipse applications (Figure 3).

**Note:** When you start the IDE from now on, the MDE Workbench appears immediately.

The MDE is now ready for use.

Perspective Views are selected by these buttons

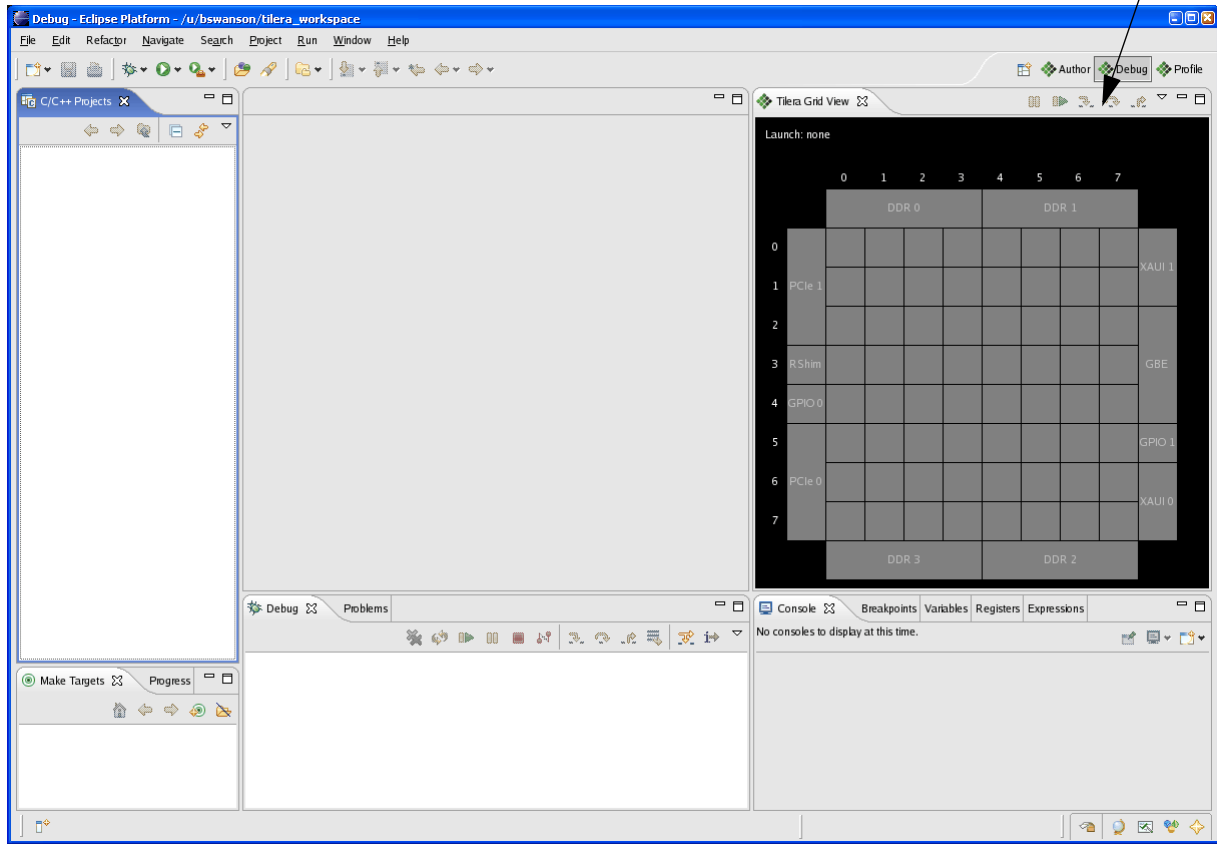


Figure 3. Tiler MDE Perspective

## MDE Perspectives

Perspectives are named sets of GUI views that are useful in developing applications. The Tiler plug-in automatically loads the currently available perspectives.

The available MDE perspectives are:

- **Author** -- views useful for creating projects and authoring code
- **Debug** -- views useful for running or debugging developed code
- **Profile** -- views useful for profiling code and examining the results of profile runs

The example projects for the *Getting Started Guide* already exist, so you will use the Debug profile.

See the *MDE User's Guide* for a complete description of all three perspectives.

# BUILDING SAMPLE PROJECTS

This section tells you how to create two simple MDE sample projects using both the IDE (GUI) interface and the command-line interface (CLI).

## Building Sample Projects Using the IDE

### IDE Project Types

There are two project types:

- A Standard Make project, for which you supply the C sources and your own Makefile
- A Managed Make project, for which you supply the C sources but let the GUI create and update the Makefile structure

The sample programs in this Guide already have their own Makefiles, so you will build Standard Make projects for them.

### Launch Configuration

A Tiler application consists of one or more binary executables that run together as a unit on either the Tiler simulator or on hardware. You create a launch configuration to run or debug an application. A launch configuration is a description of the application you want to invoke, together with any required arguments, settings, or environment information needed to invoke it.

Depending on the target you want to run on (simulator or hardware), the pathnames you use and options you select will be different. The overall process of creating and using a launch configuration is the same, however.

## Creating and Building a Project

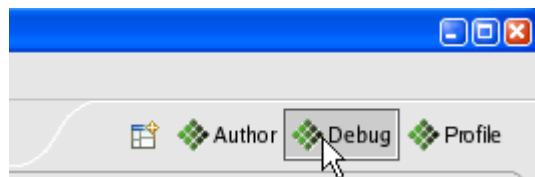
This section tells you how to create a new project, add the Makefile, and build the project.

### Creating the Project

You will create and build the `hello_world_multicore` project, which runs on all 64 tiles. (The `hello_world` project runs on a single tile.)

To create the sample project:

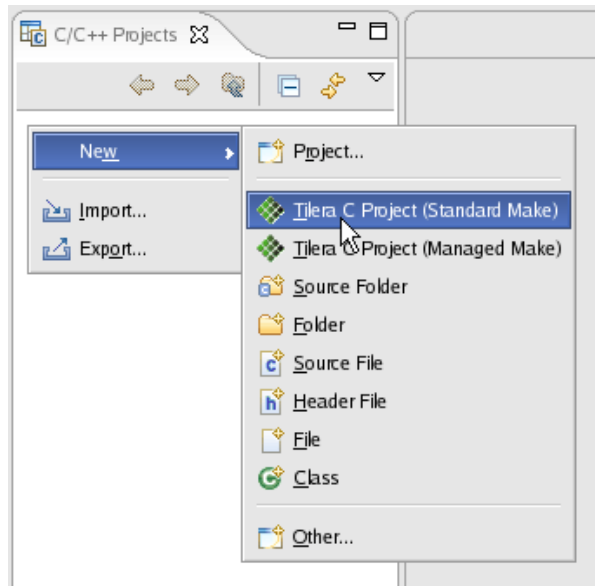
1. Select the Debug perspective by clicking **Debug** on the Perspective toolbar in the upper-right corner of the Eclipse window.



*Figure 4. Perspective Toolbar*

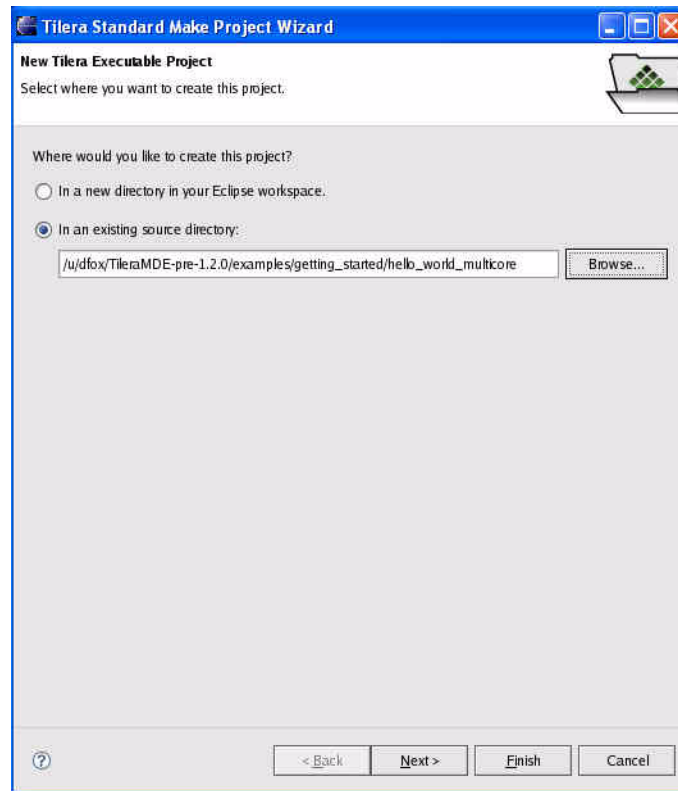
The MDE Perspective window changes to display the Debug tab in the lower pane.

2. Select **File->New->Tiler C Project (Standard Make)** (Figure 5).



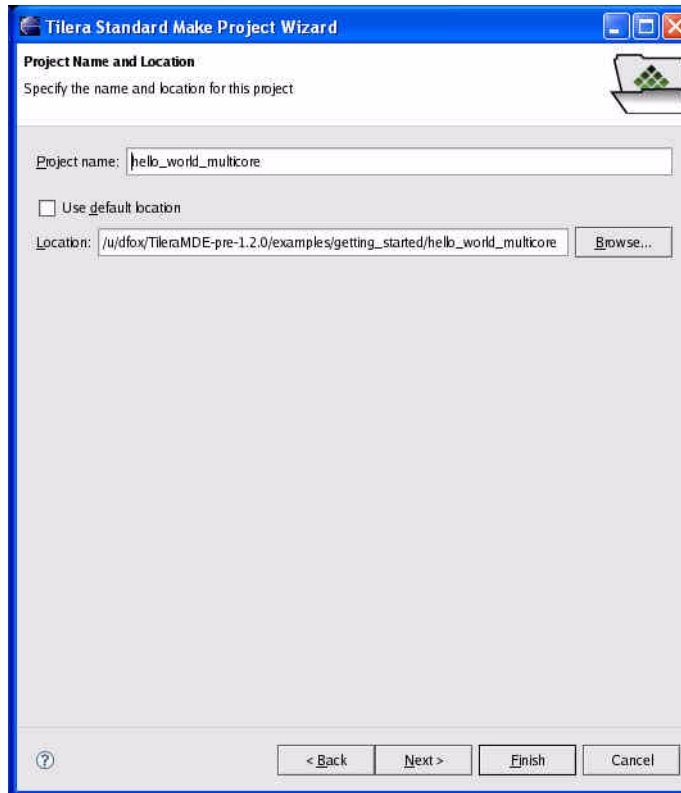
**Figure 5. New Project Menu -- Standard Make Project**

The Standard Make project Wizard appears (Figure 6).



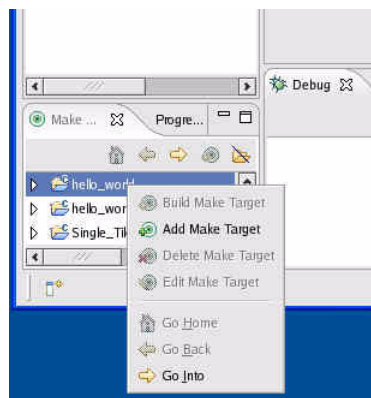
**Figure 6. Standard Make Project Wizard - New Tilera Executable Project**

- Use the **Browse** button to select **In an existing source directory** and specify the `hello_world_multicore` examples directory you copied previously.



**Figure 7. Project Name and Location**

- Click **Finish**.
- If you are asked whether you want to switch to the Tileria Authoring perspective, click **No**.



**Figure 8. Options List with Add Make Target Active**



## Building the Project

1. In the Make Targets view, click the small arrowhead at the left of the `hello_world_multicore` directory.
2. Right click the `hello_world_multicore` project name under the Make Targets view at the lower left portion of the workspace. A list of options appears (Figure 8).
3. Click **Add Make Target**. The Create a new Make target window appears (Figure 9).

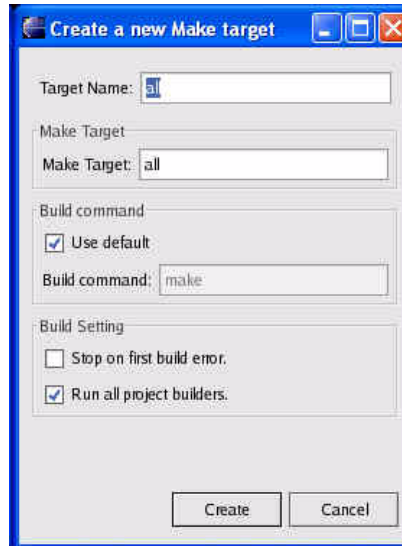


Figure 9. Create a new Make Target Window

4. Change Target Name and Make Target to `build`.
5. Click **Create**. This creates a new make target named `build`, which will run the build make target in the make file.
6. Go through the same process to create a `clean` target.
7. Right click the `build` target name in the Make Targets view and click **Build Make Target** (Figure 10).

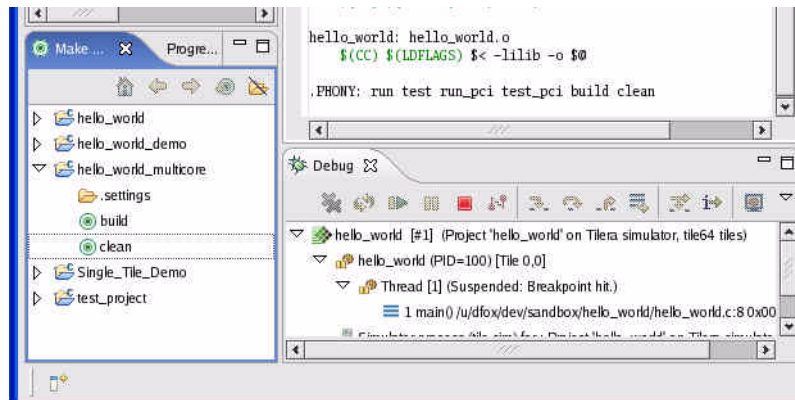
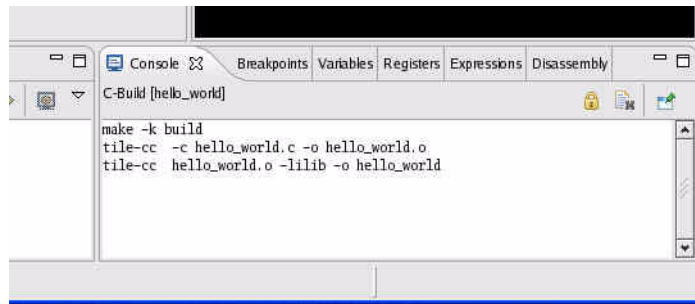


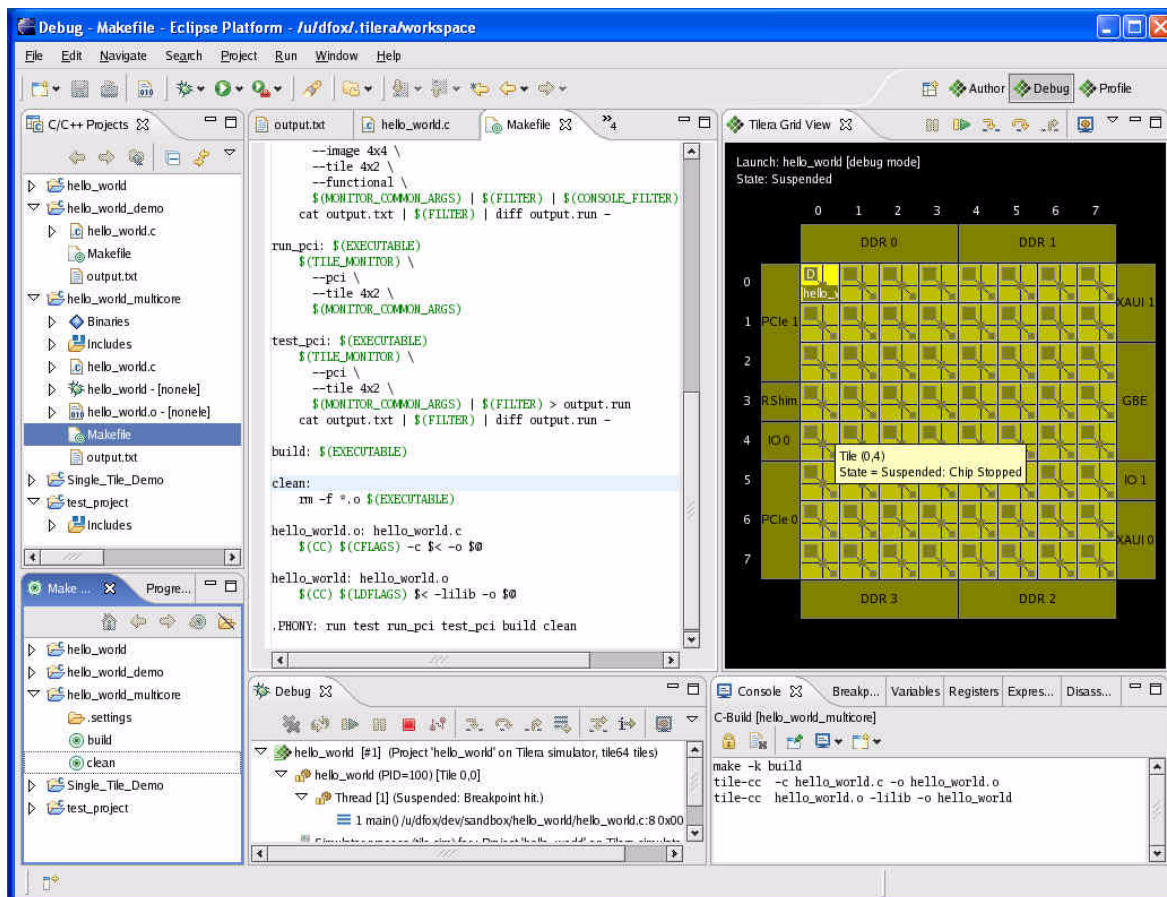
Figure 10. Build Make Target

- If the **Console** tab is not currently selected, click it. If your Makefile's build target completed successfully, you will see build output messages similar to the example in [Figure 11](#).



**Figure 11. Console View -- Standard Make Build Output**

Your project is now populated and built. Open the project and double click on the source file. The IDE views should resemble [Figure 12](#).



**Figure 12. IDE Views**

# RUNNING AND DEBUGGING SAMPLE PROJECTS USING IDE

## Creating Breakpoints

To create breakpoints in your code:

1. Open the source file to view the line where you want to place the first breakpoint.
2. Select the `hello_world.c` tab to view the code ([Figure 15](#)).
3. Set one or more breakpoints by double clicking in the gray line to the left of the instruction you want to use. Tiler suggests setting two breakpoints:

```
if (ilib_proc_go_parallel(remaining + 1, NULL) != ILIB_SUCCESS)
and
```

```
printf("Process %d: Hello, Brave New World!\n", my_rank);
```

This provides breakpoints in both the non-parallel and parallel portions of the code. You will use these breakpoints when you debug the project.

## Creating a Launch Configuration

To run the `hello_world_multicore` project on the hardware in debug mode:

1. Select the project folder of the **C/C++ Projects** view. This allows the Run/Debug dialog to default many options based on the name and contents of the project.
2. Select **Run->Debug** to open the Run/Debug dialog in Debug mode.

The first time you bring up the Run/Debug dialog, it appears as in [Figure 13](#).

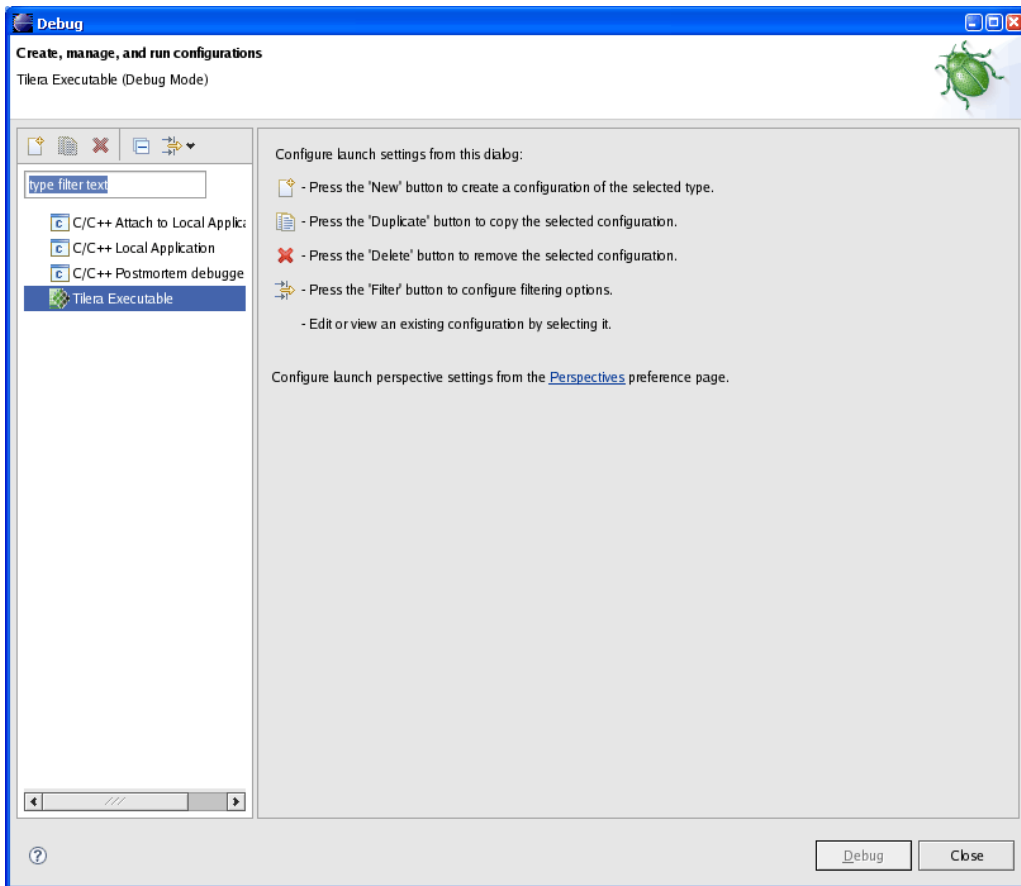
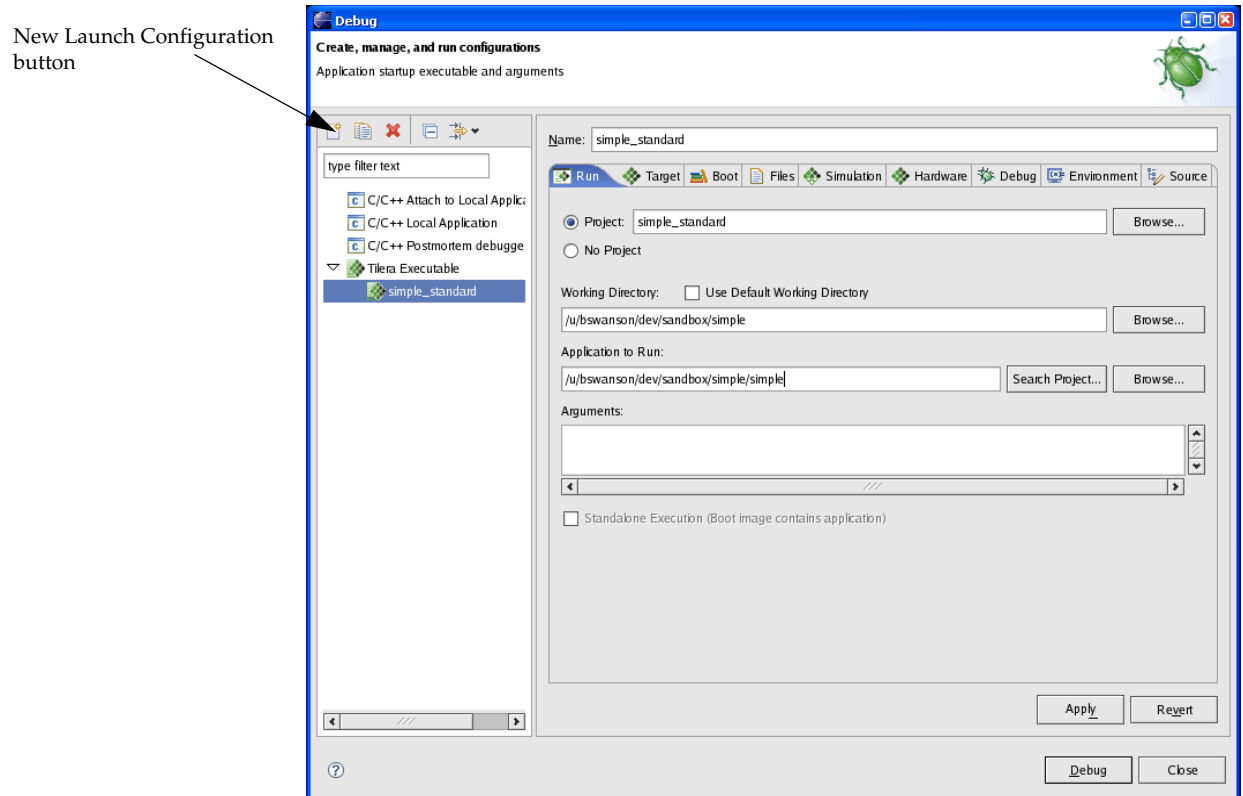


Figure 13. Run/Debug Dialog -- Initial State

3. Select **Tilera Executable** from the list on the left.
4. Click the **New Launch Configuration** button on the toolbar above the list. This adds a new Tilera Executable launch (Figure 14).



**Figure 14. Run/Debug Dialog**

5. The **Name** field at the top of the editor area defines the displayed name for this launch configuration. This is already set to the name of your project.
6. Click the **Target** tab.
7. Select **Hardware** as the target. from the Execution Target drop-down list.
8. Click **Apply** to save the configuration.
9. Click **Close**.

## Debugging the Sample Project

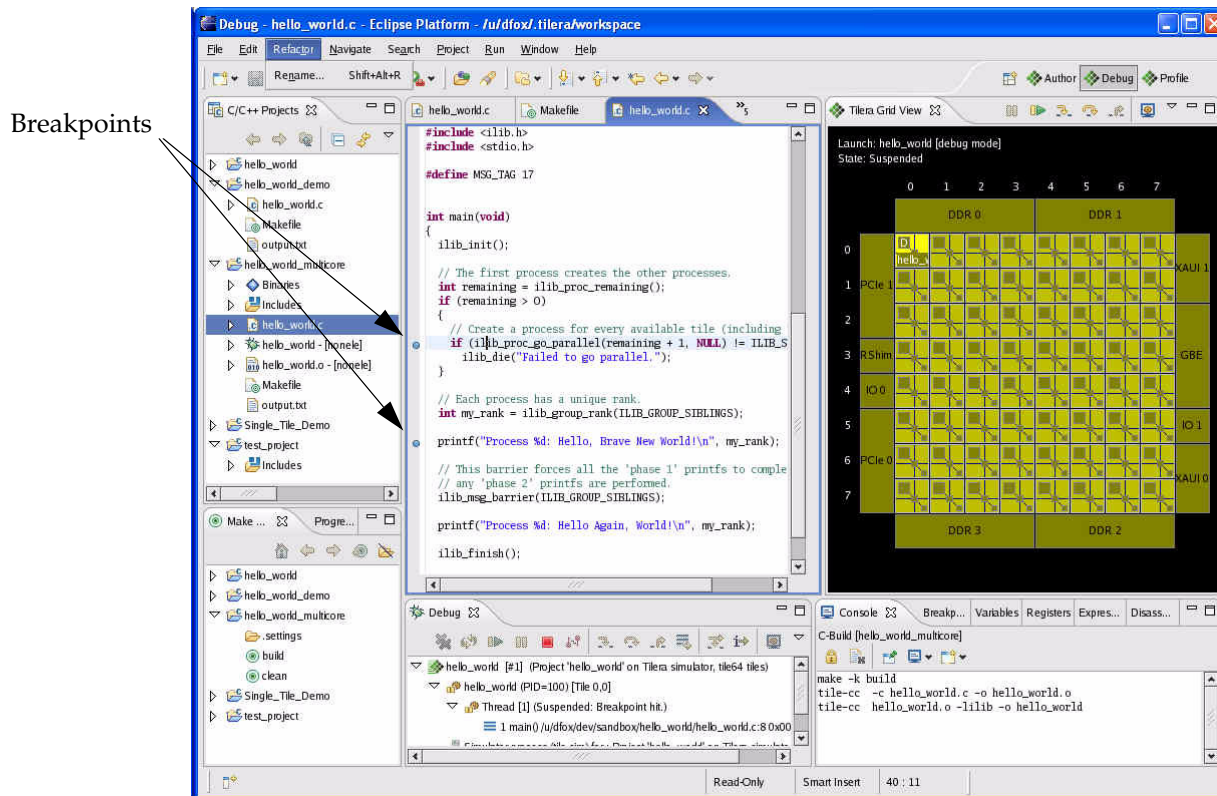
To debug your project:

1. Exit the C++ pane.
2. Click **Run --> Debug** to run the application.

The application executes. Now you can use either the Debug View or the Grid View to view the state of your application and control debugging operations such as stepping, resuming, and terminating the run.

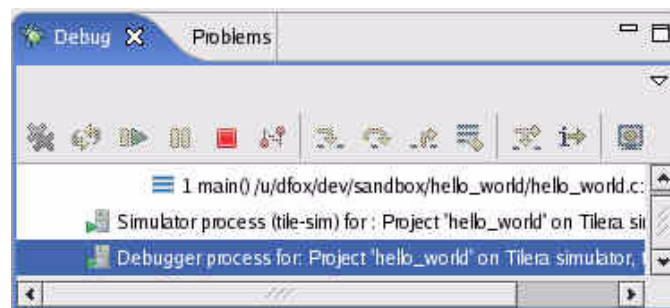
**Note 1:** You are debugging tile 0,0, so the breakpoints only stop tile 0.0.

**Note 2:** Remove the breakpoints set for the `hello_world` project by double-clicking them before setting breakpoints and debugging the `hello_world_multicore` example. You only need to do this in cases where the source files being debugged have the same name.



**Figure 15. Debug View Showing `hello_world_multicore` Execution**

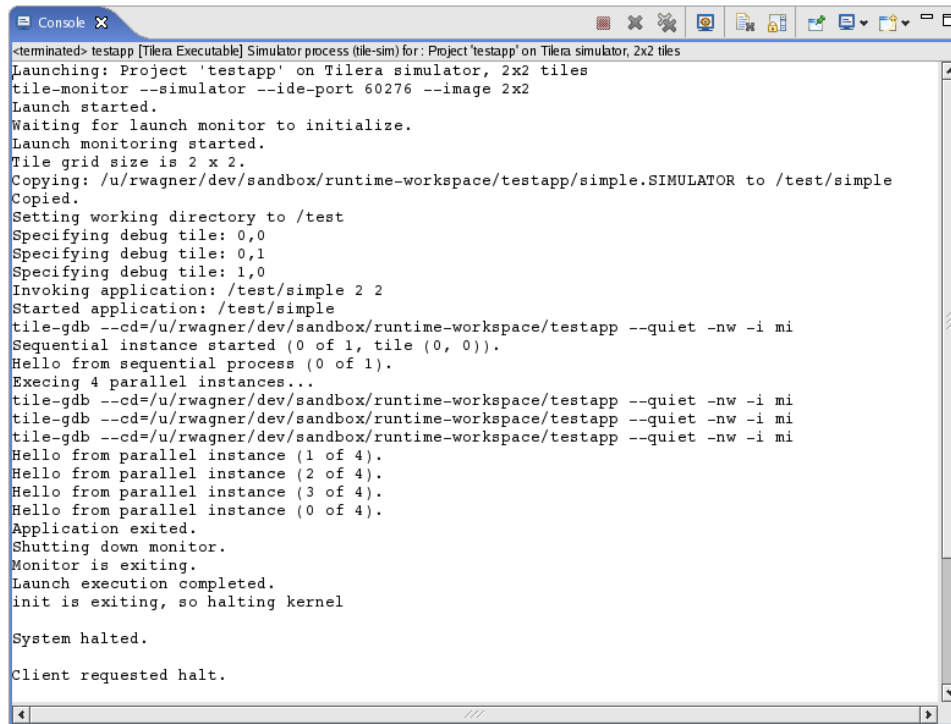
The Debug view (Figure 16) is the main point of control for debugging your application. It displays the current execution state of all processes associated with the running application and provides controls for resuming, halting, and single-stepping processes.



**Figure 16. Debug View**

## Viewing Output

If the **Console** tab is not currently displayed, select it. The output of your application (and any execution messages) displays (Figure 17).



```
<terminated> testapp [Tilera Executable] Simulator process (tile-sim) for: Project 'testapp' on Tilera simulator, 2x2 tiles
Launching: Project 'testapp' on Tilera simulator, 2x2 tiles
tile-monitor --simulator --ide-port 60276 --image 2x2
Launch started.
Waiting for launch monitor to initialize.
Launch monitoring started.
Tile grid size is 2 x 2.
Copying: /u/rwagner/dev/sandbox/runtime-workspace/testapp/simple.SIMULATOR to /test/simple
Copied.
Setting working directory to /test
Specifying debug tile: 0,0
Specifying debug tile: 0,1
Specifying debug tile: 1,0
Invoking application: /test/simple 2 2
Started application: /test/simple
tile-gdb --cd=/u/rwagner/dev/sandbox/runtime-workspace/testapp --quiet -nw -i mi
Sequential instance started (0 of 1, tile (0, 0)).
Hello from sequential process (0 of 1).
Execing 4 parallel instances...
tile-gdb --cd=/u/rwagner/dev/sandbox/runtime-workspace/testapp --quiet -nw -i mi
tile-gdb --cd=/u/rwagner/dev/sandbox/runtime-workspace/testapp --quiet -nw -i mi
tile-gdb --cd=/u/rwagner/dev/sandbox/runtime-workspace/testapp --quiet -nw -i mi
Hello from parallel instance (1 of 4).
Hello from parallel instance (2 of 4).
Hello from parallel instance (3 of 4).
Hello from parallel instance (0 of 4).
Application exited.
Shutting down monitor.
Monitor is exiting.
Launch execution completed.
init is exiting, so halting kernel

System halted.

Client requested halt.
```

Figure 17. Console View -- Run Output

Since each of your application's processes can produce its own output by writing to `stdout` or `stderr`, you can isolate the output from one or more of your application's processes. To do so, click the **Open Console for Process...** button in the toolbar of the Console view (Figure 18).

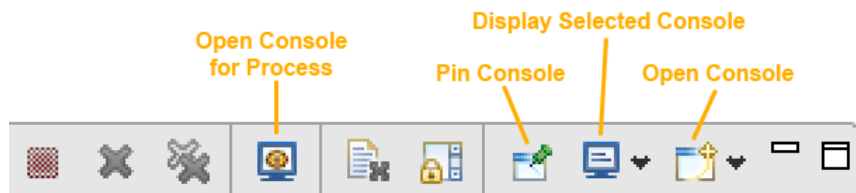
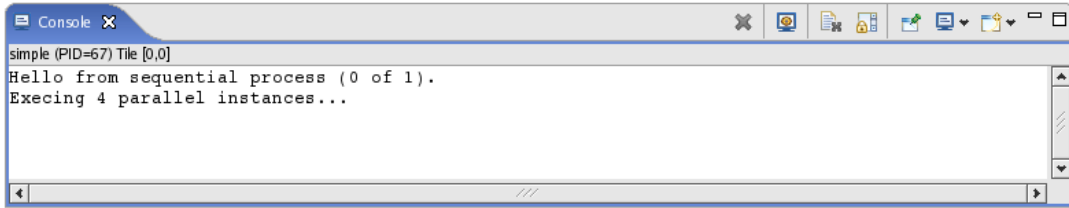


Figure 18. Console view toolbar

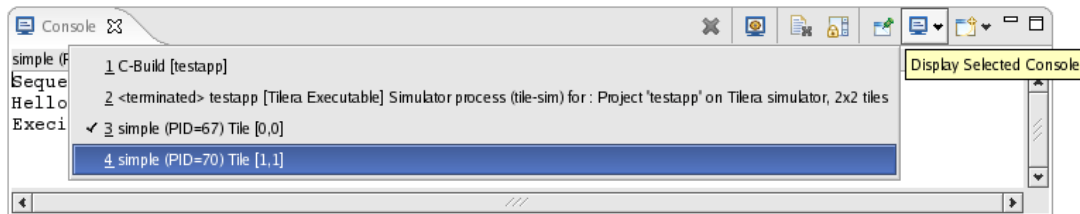
This displays a dialog listing the processes in your application. Select one or more processes (Figure 19).

Each process's output displays on its own page of the console. For example, the console of Figure 19 shows the output from the first process, which spawns four other processes.



**Figure 19. The Open Process for Console dialog**

The dialog snapshot shows one of the two selected processes. Since only one console displays at a time in a Console view, you can use the **Display Selected Console** menu in the toolbar to display another console, such as the second process selected in the dialog (Figure 20).



**Figure 20. The Display Selected Console menu**



# Building and Running Sample Projects Using the Command Line Interface

This section tells you how to run both the single- and multi-tile sample programs using the command line interface (CLI).

## Setting the Path

Before you build and run any of the examples listed here, set the correct path. For the bash shell, this is

```
% export TILERA_HOME=/ . . .
% export PATH=${PATH}:${TILERA_HOME}/bin
```

## Running the Sample Programs on the Simulator

This section tells you how to build and run the single- and multi-tile sample programs on the simulator.

### Building and Running the Single-Tile Sample Program

To run the single-tile sample program using the CLI:

1. Go to the `/${TILERA_HOME}/hello_world` directory:

```
% cd ${TILERA_HOME}/examples/getting_started/hello_world
```

**Note:** You can optionally copy the example files to any convenient working directory and use that directory for these samples.

2. Use the provided Makefile to build and run the example.

```
% cd hello_world
% make run
tile-cc -c hello_world.c -o hello_world.o
tile-cc hello_world.o -lilib -o hello_world
tile-monitor \
--image 2x2 \
--functional \
--batch-mode --mkdir /opt/test --cd /opt/test --upload hello_world hello_world --
hello_world
Hello World!
```

### Running the Multi-Tile Sample Program

The multi-tile example uses iLib library functions to launch multiple instances of itself across multiple tiles. It also uses an iLib barrier routine to ensure that the hello world messages from the various tiles are not intermixed.

To run the example:

1. `cd ../hello_world_multicore`
2. `% make run`

```
tile-cc -c hello_world.c -o hello_world.o
tile-cc hello_world.o -lilib -o hello_world
tile-monitor \
--image 4x4 \
--tile 4x2 \
```

```

--functional \
--batch-mode --mkdir /opt/test --cd /opt/test --upload hello_world hello_world --
hello_world
Process 1: Hello, Brave New World!
Process 2: Hello, Brave New World!
Process 3: Hello, Brave New World!
Process 4: Hello, Brave New World!
Process 5: Hello, Brave New World!
Process 6: Hello, Brave New World!
Process 7: Hello, Brave New World!
Process 0: Hello, Brave New World!
Process 1: Hello Again, World!
Process 2: Hello Again, World!
Process 3: Hello Again, World!
Process 4: Hello Again, World!
Process 5: Hello Again, World!
Process 6: Hello Again, World!
Process 7: Hello Again, World!
Process 0: Hello Again, World!

```

## Running the Sample Programs on the TILExpress-64 Card

This section tells you how to run the single- and multi-tile example programs on the TILExpress-64 card.

**Note:** You must have a TILExpress-64 card installed to run these example programs successfully.

### Building and Running the Single-Tile Program

To run the single-tile sample program on the TILExpress-64 Card:

1. CD to the `hello_world` directory:

```
cd ${TILER_HOME}examples/getting_started/hello_world
```

**Note:** You can optionally copy the example files to any convenient working directory and use that directory for these samples.

2. Use the provided Makefile to build and run the example:

```

% make run_pci
tile-cc -c hello_world.c -o hello_world.o
tile-cc hello_world.o -lilib -o hello_world
tile-monitor --pci hello_world
Hello World!

```

### Building and Running the Multi-Tile Sample Program

1. CD to the `hello_world_multicore` directory:

```
cd ../hello_world_multicore
```

2. Use the provided Makefile to build and run the example:

```

% make run_pci
tile-cc -c hello_world.c -o hello_world.o
tile-cc hello_world.o -lilib -o hello_world
tile-monitor --pci hello_world\
Process 1: Hello, Brave New World!
Process 2: Hello, Brave New World!
Process 3: Hello, Brave New World!
Process 4: Hello, Brave New World!
Process 5: Hello, Brave New World!
Process 6: Hello, Brave New World!

```

```
Process 7: Hello, Brave New World!  
Process 0: Hello, Brave New World!  
Process 1: Hello Again, World!  
Process 2: Hello Again, World!  
Process 3: Hello Again, World!  
Process 4: Hello Again, World!  
Process 5: Hello Again, World!  
Process 6: Hello Again, World!  
Process 7: Hello Again, World!  
Process 0: Hello Again, World!
```

# GETTING HELP

From the IDE: Select Help->Help Contents from the IDE menu bar. In the Help window's Contents pane, select **Tilera MDE User Guide**. This document contains an overview of all the Tilera extensions to the base Eclipse development environment.

From the CLI: the MDE commands generally follow the Linux convention of providing a `--help` option that summarizes the command's usage. For example, Type `tile-sim --help` to see the command-line syntax for a command.

## Frequently Asked Questions

- **When I try to start `tile-eclipse` it says the workspace is already in use.**

The `tile-eclipse` IDE is based on Eclipse, which has a guard mechanism to enforce the rule that at most one Eclipse instance can be running on a given workspace at a time.

First, make sure you don't already have an instance of **tile-eclipse** running on the same host. Remember that if you're running under X Windows the application window might be displayed on a different screen. (Example: you're trying to run `tile-eclipse` from home, and have left an instance running at work.)

Next, check your process listing (e.g., `ps -wx`) and look for any rogue `eclipse` or `java` instances. If Eclipse was not shut down cleanly, a windowless process might have been left behind. Kill any such processes. When the Eclipse process is shut down it should release its lock on the workspace directory.

If the above does not work (or there are no `eclipse/java` processes currently running), manually delete the `.lock` file from your workspace directory. This should be located in:

```
${HOME}/tilera/workspace/.metadata/.lock
```

You should now be able to start `tile-eclipse` normally.

- **How do I kill a process?**

To kill a process, use:

```
kill -KILL <pid>
```