



# *Getting Started with the TASKING VX-toolset for ARM*

## ***Getting Started with the TASKING VX-toolset for ARM***

Copyright © 2011 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment. Altium, TASKING, and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.

# Table of Contents

1. Preparing for First Use .....	1
1.1. Installing the Software .....	1
1.1.1. Installation for Windows .....	1
1.1.2. Licensing .....	2
1.2. Starting / Closing Eclipse .....	5
1.3. How to Use the Documentation .....	7
1.4. Related Publications .....	8
2. Setting up a Project .....	11
2.1. Create a Project .....	11
2.2. Delete a Project .....	13
2.3. Manually Add a File to Your Project .....	13
2.4. Editing Files: C/C++ Editor .....	16
2.5. Closing, Opening and Activating a Project .....	17
2.6. Clone a Project .....	18
2.7. Configuring the Target .....	18
2.8. Setting Project Options .....	21
2.9. Build a Project .....	24
2.10. Using the Sample Projects .....	24
2.11. Import/Export Project Properties .....	25
3. Debugging your Application .....	27
3.1. Setting up a Project for Debugging .....	27
3.1.1. Create a Sample Project .....	27
3.1.2. Create a Debug Configuration .....	28
3.2. Start a Debug Session .....	28
3.3. Stepping through the Application .....	29
3.4. Setting and Removing Breakpoints .....	30
3.5. Reload Current Application .....	31
3.6. End a Debug Session .....	32
3.7. Multiple Debug Sessions .....	32



# Chapter 1. Preparing for First Use

This chapter guides you through the installation process of the TASKING VX-toolset for ARM<sup>®</sup>. It also describes which documentation is available and how you best can use it.

In this manual, **TASKING VX-toolset for ARM** and **ARM toolset** are used as synonyms.

## 1.1. Installing the Software

This section describes the installation of the embedded software for Windows. It also describes how to license the software.

### 1.1.1. Installation for Windows

#### System Requirements

Before installing, make sure the following minimum system requirements are met:

- Windows 7, Windows Vista, Windows XP or Windows 2000
- 2 GHz Pentium class processor
- 1 GB memory
- 3 GB free hard disk space
- Screen resolution: 1024 x 768 or higher

#### Installation

1. Insert the TASKING VX-toolset CD-ROM into the CD-ROM drive.
2. If the installation program does not start automatically, browse to your CD-ROM drive and run the program **setup.exe**.

*The TASKING Setup dialog box appears.*

3. Select a product and click on the **Install** button.
4. Follow the instructions that appear on your screen.

You can find the serial number on the invoice, delivery note, or picking slip delivered with the product.

### 1.1.2. Licensing

TASKING products are protected with license management software (FLEXlm). To use a TASKING product, you must install the license key provided by Altium for the type of license purchased.

You can run TASKING products with a node-locked license, with a floating license or in trial mode. When you order a TASKING product determine which type of license you need.

#### Node-locked license (Windows only)

This license type locks the software to one specific PC so you can use the product on that particular PC only.

#### Floating license

This license type manages the use of TASKING product licenses among users at one site. This license type does not lock the software to one specific PC or workstation but it requires a network. The software can then be used on any computer in the network. The license specifies the number of users who can use the software simultaneously. A system allocating floating licenses is called a *license server*. A license manager running on the license server keeps track of the number of users.

#### Trial mode

When you use the product without a valid license, the tools will run in *trial mode*. This means you can use the toolset 15 days with full functionality. When you run the toolset in trial mode, each tool will report this. When you use a license that does not cover the full toolset, the tools that are not covered by the license will run in trial mode.

When after you installed the license file, the tools that are covered by the license still report that they are running in trial mode, this means that there is a license error. If you want to force the termination of the trial mode to get the FLEXlm error message you can set the environment variable `FORCE_NO_TRIAL` to "yes".

#### 1.1.2.1. Obtaining License Information

Before you can install a software license you must have a "License Key" containing the license information for your software product. If you have not received such a license key follow the steps below to obtain one. Otherwise, you can install the license.

1. Run the License Administrator during installation and follow the steps to **Request a license key from Altium by E-mail**.
2. E-mail the license request to your local TASKING sales representative. The license key will be sent to you by E-mail.

#### 1.1.2.2. Installing Node-Locked Licenses

If you do not have received your license key, read [Section 1.1.2.1, Obtaining License Information](#), before continuing.

1. Install the TASKING software product following the installation procedure described in [Section 1.1.1, \*Installation for Windows\*](#), if you have not done this already.
2. Create a license file by importing a license key or create one manually:

### Import a license key

During installation you will be asked to run the License Administrator. Otherwise, start the License Administrator (**licadmin.exe**) manually.

In the License Administrator follow the steps to **Import a license key received from Altium by E-mail**. The License Administrator creates a license file for you.

### Create a license file manually

If you prefer to create a license file manually, create a file called "license.dat" in the `c:\flexlm` directory, using an ASCII editor and insert the license key information received by E-mail in this file. This file is called the "license file". If the directory `c:\flexlm` does not exist, create the directory.

If you wish to install the license file in a different directory, see [Section 1.1.2.4, \*Modifying the License File Location\*](#).

If you already have a license file, add the license key information to the existing license file. If the license file already contains any SERVER lines, you must use another license file. See [Section 1.1.2.4, \*Modifying the License File Location\*](#), for additional information.

The software product and license file are now properly installed.

## 1.1.2.3. Installing Floating Licenses

If you do not have received your license key, read [Section 1.1.2.1, \*Obtaining License Information\*](#), before continuing.

1. Install the TASKING software product following the installation procedure described earlier in this chapter on each computer or workstation where you will use the software product.
2. On each PC or workstation where you will use the TASKING software product the location of a license file must be known, containing the information of all licenses. Either create a local license file or point to a license file on a server:

### Add a licence key to a local license file

A local license file can reduce network traffic.

You can follow the same steps to import a license key or create a license file manually, as explained in the previous section with the installation of a node-locked license.

## Getting Started with the TASKING VX-toolset for ARM

If you wish to install the license file in a different directory, see [Section 1.1.2.4, Modifying the License File Location](#).

If you already have a license file, add the license key information to the existing license file. If the license file already contains any SERVER lines, you must use another license file. See [Section 1.1.2.4, Modifying the License File Location](#), for additional information.

### Point to a license file on the server

Set the environment variable `LM_LICENSE_FILE` to "*port@host*", where *host* and *port* come from the SERVER line in the license file. For Windows, you can use the License Administrator to do this for you. In the License Administrator follow the steps to **Point to a FLEXlm License Server to get your licenses**.

3. If you already have installed FLEXlm v8.4 or higher (for example as part of another product) you can skip this step and continue with step 4. Otherwise, install SW000098, the Flexible License Manager (FLEXlm), on the license server where you want to use the license manager.
4. If FLEXlm has already been installed as part of a non-TASKING product you have to make sure that the `bin` directory of the FLEXlm product contains a copy of the **Tasking** daemon. This file part of the TASKING product installation and is present in the `flexlm` subdirectory of the toolset. This file is also on every product CD that includes FLEXlm, in directory `licensing`.
5. On the license server also add the license key to the license file. Follow the same instructions as with "Add a license key to a local license file" in step 2.

See the FLEXlm PDF manual delivered with SW000098, which is present on each TASKING product CD, for more information.

### 1.1.2.4. Modifying the License File Location

The default location of the license file for Windows is:

```
c:\flexlm\license.dat
```

If you want to use another name or directory for the license file, each user must define the environment variable `LM_LICENSE_FILE`.

If you have more than one product using the FLEXlm license manager you can specify multiple license files to the `LM_LICENSE_FILE` environment variable by separating each pathname with a ';':

Example Windows:

```
set LM_LICENSE_FILE=c:\flexlm\license.dat;c:\license.dat
```

If the license file is not available on these hosts, you must set `LM_LICENSE_FILE` to *port@host*, where *host* is the host name of the system which runs the FLEXlm license manager and *port* is the TCP/IP port number on which the license manager listens.

To obtain the port number, look in the license file at host for a line starting with "SERVER". The fourth field on this line specifies the TCP/IP port number on which the license server listens. For example:

```
setenv LM_LICENSE_FILE 7594@elliott
```

See the FLEXlm PDF manual delivered with SW000098, which is present on each TASKING product CD, for detailed information.

### 1.1.2.5. How to Determine the Host ID

The host ID depends on the platform of the machine. Please use one of the methods listed below to determine the host ID.

Platform	Tool to retrieve host ID	Example host ID
Windows	<b>licadmin</b> (License Administrator, or use <b>lmhostid</b> )	0060084dfbe9
UNIX	<b>hostid</b>	11ac5702

On Windows, the License Administrator (**licadmin**) helps you in the process of obtaining your license key.

If you do not have the program **licadmin** you can download it from our Web site at: <http://www.tasking.com/support/flexlm/licadmin.zip>. It is also on every product CD that includes FLEXlm, in directory `licensing`.

### 1.1.2.6. How to Determine the Host Name

To retrieve the host name of a machine, use one of the following methods.

Platform	Tool to retrieve host ID
Windows	<b>licadmin</b> or  Go to the Control Panel, open "System". In the "Computer Name" tab look for "Full computer name".
UNIX	<b>hostname</b>

## 1.2. Starting / Closing Eclipse

### Starting Eclipse

To start Eclipse:

1. From the Windows **Start** menu, select **Programs » TASKING VX-toolset for ARM vx.yrz» TASKING VX-toolset for ARM**.

*The Workspace Launcher dialog appears.*

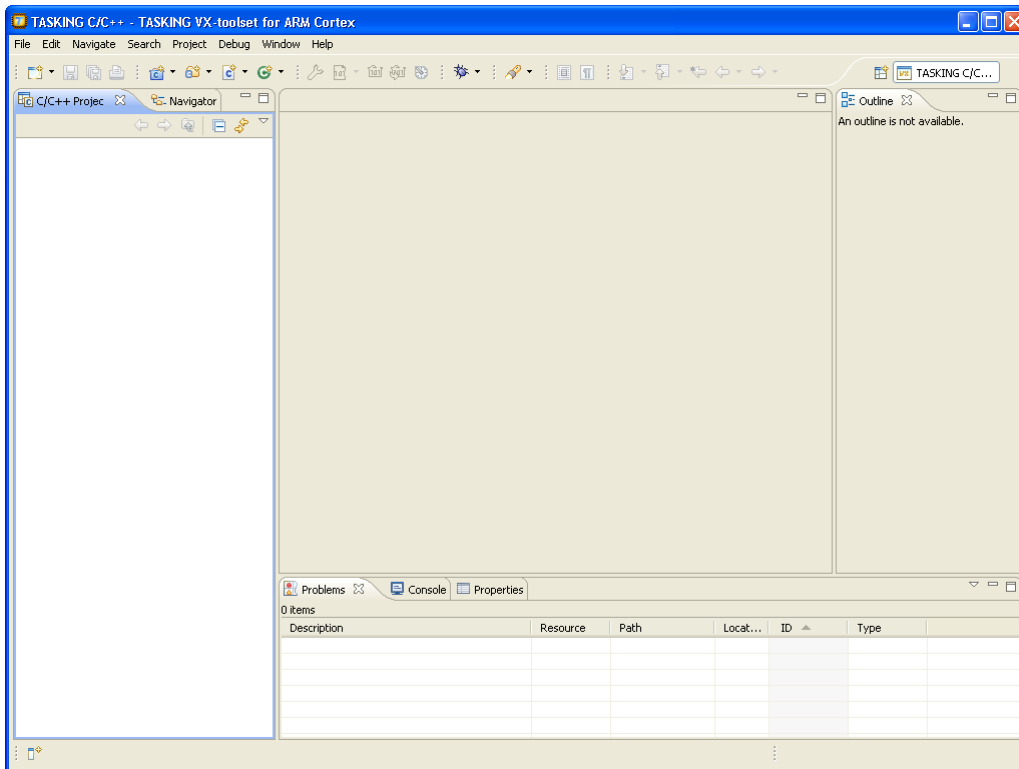
2. Enter the path to the workspace.

*In the remainder of this manual, we assume you use the default.*

## Getting Started with the TASKING VX-toolset for ARM

3. Enable the option **Use this as the default and do not ask again**.
4. Click **OK** to proceed.

Initially, Eclipse opens with a workbench displaying the C/C++ perspective with only the **Welcome** view visible. This view provides some general information and alternative ways to access the online documentation. Click the cross to close the Welcome view, this causes the other views in the perspective to become visible.



At any time you can get the Welcome view back again by selecting **Welcome** from the **Help** menu.

## Closing Eclipse

To close Eclipse:

- From the **File** menu, select **Exit**.

Upon exit, Eclipse saves the current workbench layout. The next time you start Eclipse, the last saved workbench layout is used.

## 1.3. How to Use the Documentation

The documentation for the TASKING VX-toolset for ARM consists of:

- online documentation for Eclipse
- this Getting Started manual
- online TASKING VX-toolset for ARM User Guide

It is strongly recommended to read the documentation in this order.

### Getting acquainted with Eclipse

If you are new to Eclipse, start familiarizing with Eclipse. Eclipse comes with several online documents. One document describes how Eclipse is organized as a Workbench, with Perspectives that contain Views; another document explains how to create a sample C/C++ project, build and debug it (CDT documentation).

To start with this documentation:

1. Start Eclipse.
2. From the **Help** menu, select **Help Contents**.  
*The help screen overlays the Eclipse Workbench.*
3. In the left pane, select **Workbench User Guide** to learn more about working in Eclipse.
4. Continue with **C/C++ Development User Guide** to learn more about creating and developing a C/C++ project.

This part of the documentation explains how to create a "hello world" example. Be aware that this example does not use the TASKING tools, it uses the standard GNU compiler in Eclipse instead.

### Getting started with the TASKING VX-toolset for ARM (this manual)

The TASKING Getting Started and User Guide contain specific information for the TASKING VX-toolset for ARM. Its content overrides any information found in the Eclipse and CDT documentation.

The next chapters of this manual explain how to setup and work with an ARM project. It shows some important features of the ARM toolset.

### Online TASKING VX-toolset for ARM User Guide

Once you are introduced to Eclipse and the ARM toolset, you can start creating your own projects. The online documentation for the ARM toolset covers the ARM C/Assembly language, as well as detailed description of the various tools and options. Accessing the documentation for the ARM toolset is similar to accessing the online documentation for Eclipse:

1. Start Eclipse.
2. From the **Help** menu, select **Help Contents**.

## Getting Started with the TASKING VX-toolset for ARM

*The help screen overlays the Eclipse Workbench.*

3. In the left pane, select **TASKING VX-toolset for ARM User Guide** to access the documentation for the ARM toolset.

The TASKING manuals are also available in PDF format via the Windows Start menu:

- Browse to **Start » Programs » TASKING VX-toolset for ARM vx.y.rz » Manuals** and select the manual you need.

## 1.4. Related Publications

### C Standard

- C A Reference Manual (fifth edition) by Samuel P. Harbison and Guy L. Steele Jr. [2002, Prentice Hall]
- ISO/IEC 9899:1999(E), Programming languages - C [ISO/IEC]

More information on the standards can be found at <http://www.ansi.org/>

### C++ Standard

- ISO/IEC 14882:2003 C++ standard [ANSI]

More information on the standards can be found at <http://www.ansi.org/>

- The C++ Programming Language (second edition) by Bjarne Strastrup [1991, Addison Wesley]
- The Annotated C++ Reference Manual by Margaret A. Ellis and Bjarne Strastrup [1990, Addison Wesley]

### CERT C Secure Coding Standard

- The CERT C Secure Coding Standard by Robert C. Seacord [October 2008, Addison Wesley]
- The CERT C Secure Coding Standard web site <http://www.securecoding.cert.org/>

For general information about CERT secure coding, see <http://www.cert.org/secure-coding>

### MISRA-C

- MISRA-C:2004, Guidelines for the Use of the C Language in Critical Systems [MIRA Ltd, 2004]

See also <http://www.misra-c.com/>

- Guidelines for the Use of the C Language in Vehicle Based Software [MIRA Ltd, 1998]

See also <http://www.misra.org.uk/>

### ARM

- ARM Architecture Reference Manual - ARM DDI 0100I [2005, ARM Limited]

- ARMv6-M Architecture Reference Manual - ARM DDI 0419C [2010, ARM Limited]
- ARMv7-M Architecture Reference Manual - ARM DDI 0403D [2010, ARM Limited]
- Cortex-M0/1/3/4 Technical Reference Manuals [ARM Limited]
- Cortex Microcontroller Software Interface Standard (CMSIS)



# Chapter 2. Setting up a Project

This tutorial shows how to create an embedded software project with the ARM toolset. It lets you create your own project with a simple "Hello World" example.

By now you should be familiar with the Eclipse workbench, perspectives and views. If you are not, please read the Eclipse documentation as described in [Section 1.3, How to Use the Documentation](#).

## 2.1. Create a Project

### Set the TASKING C/C++ perspective

Before creating an ARM project, it is necessary to have the TASKING C/C++ perspective on the workbench. By default, this should be the case when you start Eclipse, but if it is not, do the following:

1. Start Eclipse.

*Eclipse starts with the last saved workbench layout.*

2. To open the TASKING C/C++ perspective: from the **Window** menu, select **Open Perspective » Other... » TASKING C/C++**.

*The name of the perspective is displayed in the title bar of the workbench window.*

If you attempt to create an ARM project while the TASKING C/C++ perspective is not active, Eclipse will ask you to activate the TASKING C/C++ perspective after you finish the **New C/C++ Project** wizard.

### Create an ARM project with the New C/C++ Project wizard

1. From the **File** menu, select **New » TASKING VX-toolset for ARM C/C++ Project**

*The New C/C++ Project wizard appears.*

2. Enter a name for your project, for example `myproject`.

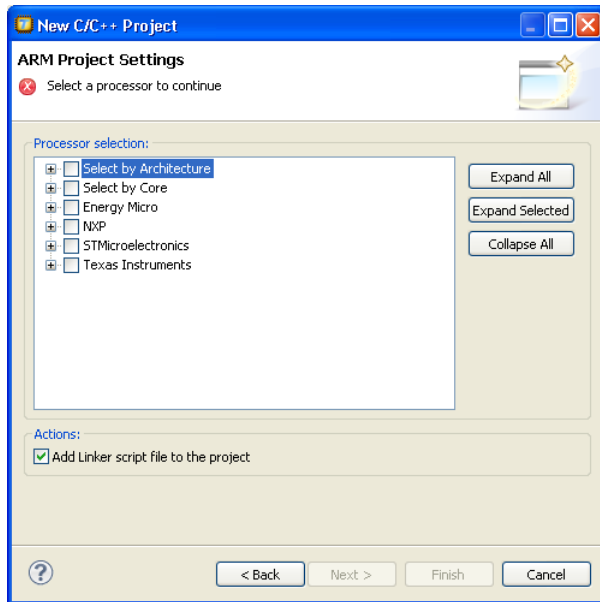
*In the **Location** field you will see the location where the new project will be stored. To change the default location, you can uncheck the **Use default location** check box and browse for an alternative location. However, use the default location for now.*

3. In the **Project type** box you can select whether to create an application or a library.

- Expand **TASKING ARM Application** and select **Hello World C Project**. This creates the file `myproject.c` with a simple main function.
- Click **Next** to continue.

*The ARM Project Settings page appears.*


## Getting Started with the TASKING VX-toolset for ARM



4. Select the target processor core or architecture for which you want to build the application. The default architecture is the ARMv7-M. Afterwards you can always change the processor in the **Project » Properties for** dialog.
5. Enable **Add Linker script file to the project**. This adds the file `myproject.lsl` to your project which can be edited to customize linking and locating. If you do not add the linker script file here, you can always add it later with **File » New » Linker Script File (LSL)**.

For details on changing the linker script file refer to the *TASKING VX-toolset for ARM User Guide*.

6. Click **Finish** to finish the wizard and to create the project.

*The project has now been created and is the active project. If you click on the **Build myproject** button () , the project is built and should give no errors or warnings.*

The left-hand pane of the Workbench window has two views. The **C/C++ Projects** view shows the structure of your projects, complete with all files that are used in the project.

The **Navigator** view lets you navigate through the physical project folder and project files on your hard disk. The navigator view does not show the include files needed for your project, because these files are stored in the general `...\include` folder, not in your project folder.

In the standard Eclipse documentation about the workbench is described how you can move and organize views on your workbench.

## 2.2. Delete a Project

The project as you just created, is stored as a subdirectory named `myproject` in the directory `C:\Documents and Settings\user\workspace_arm_version`. To delete a project, it needs to be properly removed from the workbench. To delete the project which you just created:

1. In the C/C++ Projects view, right-click on the name of the project, `myproject`, and select **Delete**.

*A dialog appears which asks for confirmation.*

2. Enable **Delete project contents on disk (cannot be undone)**. This will remove your project from the workbench and also removes the entire `myproject` subdirectory from your hard disk.

If you disable this option, this would have removed your project from the workbench, but leaves it on you hard disk. Files can be used later in other projects, or you can later import the whole project.

3. Click **OK** to confirm.

## 2.3. Manually Add a File to Your Project

We will recreate the project as described in [Section 2.1, Create a Project](#); however, this time without the automatic 'Hello World' example C source file. Instead, the example below illustrates how you can manually add a file to your project.

### Recreate your project without 'Hello World'

1. First repeat steps 1 and 2 of **Create an ARM project with the New C/C++ Project wizard** in [Section 2.1, Create a Project](#).
2. In the **Project type** box you can select whether to create an application or a library.
  - Expand **TASKING ARM Application** and select **Empty Project**. This creates a project without a C source file containing the function `main()`.
  - Click **Next** to continue.

*The ARM Project Settings page appears.*

3. Repeat steps 4 through 6 of **Create an ARM project with the New C/C++ Project wizard** in [Section 2.1, Create a Project](#).
4. Click **Finish** to finish the wizard and to create the project.

### Add a new file to your project

To add a new, empty file:

1. In the C/C++ Projects view, right-click on the name of the project, `myproject`, and select **New » Source File**.

*The New Source File dialog appears.*

## Getting Started with the TASKING VX-toolset for ARM

- Specify a source folder and a name for the new file. By default, the new file will be stored in the project folder (in this case: `myproject`). If your projects contains multiple folders, you can browse for an alternative source folder to store the new file in.
  - In the **Source folder** field, make sure it refers to `myproject`.
  - In the **Source file** field, type the name of the new file, for example `myfile.c`. Note that for C files you must specify the extension `.c`! For C++ files use the extension `.cc` or `.cpp`.
  - In the **Template** field, select a code template for your source file, for example `Default C source template` or select `<None>` if you want to start with an empty file. Note that you can configure your own templates if you click on the **Configure...** button.
- Click **Finish** to continue.

*The new file `myfile.c` is created and ready for editing in the editor view.*

## Add an existing file to your project (import)

There are three ways to add a file to your project:

- Import a file (the original file is copied to the project folder)
- Create a file in the project folder
- Create a link in the project folder to an existing file

### Import a file

Instead of creating a new file, it is also possible to import an existing file into your project or to create a file directly in the `myproject` folder. To demonstrate this, follow the steps below. Do not close Eclipse.

- First create a C source file (for example `existing.c`) with a standard editor outside Eclipse. (As content you can, for example, use a single line containing comments only).
- You can store the file anywhere on your hard disk, but not in your project folder (for example in `C:\TEMP`).

In Eclipse, follow the next steps to import the existing file:

- In the C/C++ Projects view, right-click on the project `myproject` and select **Import...**

*The Import wizard appears.*

- Select **General » File System**. Click **Next** to continue.
- In the **From directory** field, type the path to the directory where you saved `existing.c` (for example `C:\TEMP`) and click in the empty white box below.

*The left box shows the file structure of the directory, the right box shows the files located in that directory, similar to the Windows Explorer.*

4. In the left box, select the folder `TEMP`.
5. In the right box, select the file `existing.c`.
6. Click **Finish** to finish the wizard and import the file into your project.

The file `existing.c` is copied from its location at `C:\TEMP` into your project folder and is added to your project. It is now visible as a C source file in the C/C++ Projects view. Changes you make to this file, will not affect the original file stored in `C:\TEMP`. Also, removing this file from your project will remove the file also from your project folder, but the original file remains untouched.

### Create a file in the project folder

Instead of importing a file, you can create the file `existing.c` with a standard editor outside Eclipse, and store it directly in the `myproject` folder. To add the file to your project:

- In the C/C++ Projects view, right-click on `myproject` and select **Refresh**.

*The file `existing.c` should now be visible as part of your project.*

### Create a link in the project folder to an existing file

The third way to add a file to your project, is to create a link to an existing file which is stored on a different location:

1. In the C/C++ Projects view, right-click on the project `myproject` and select **New » File from Template**.

*The New File wizard appears.*

2. Select the project folder in which to create the link: type the name of your project (`myproject`) or select the project in the box below.
3. In the **File name** field, enter a name for the link, for example `link2existing.c`.
4. Click the **Advanced >>** button.

*Additional options appear on the dialog to let you create a link to an existing file.*

5. Enable the option **Link to file in the file system**.
6. Browse to the location where `existing.c` is located, select this file and click the **Open** button.
7. Click **Finish** to finish the wizard and create the link to the file in your project.

## Remove a file or link from your project

As we do not need this file for the remainder of this tutorial, we can safely remove it again from the project:

- In the C/C++ Projects view, right-click on the file `link2existing.c` and select **Delete**.

The link `link2existing.c` is no longer part of your project and has been removed from your project folder. The original file, however, remains untouched at its original location.

## Getting Started with the TASKING VX-toolset for ARM

- In the C/C++ Projects view, right-click on the file `existing.c` and select **Delete**.

The file `existing.c` is no longer part of your project and has been removed from your project folder.

Be aware that when you remove a file from your project, it always will be removed from its location in the project folder on your hard disk too!

## 2.4. Editing Files: C/C++ Editor

### Editing a file

Enter the following simple C source in your new source document (the code deliberately contains a mistake, which you will correct later on):

```
#include <stdio.h>

int main( void )
{
    printf( "Hello World\n" )    /* <- missing semicolon */
}
```

Note the following:

- The tab label of the editor view shows an asterisk in front of the file name (`*myfile.c`) to indicate that the file has been modified.
- The C/C++ editor view uses syntax coloring.
- The Outline view shows the structure of the file. You can use this view to navigate through (larger) source files easily. Alternatively you can expand the structure of the file in the C/C++ Projects view.
- Right-clicking in the editor view presents you with a list of menu commands.
- To receive more help about the editor view, make sure it is active and press **F1**.

### Saving and closing a file

To save the file:

- From the **File** menu, select **Save** (Ctrl+S).

*The project will be saved.*

To close the file:

- From the **File** menu, select **Close** (Ctrl+F4).

*Eclipse will ask you to save the files that have been modified since the last save.*

Notice also the menu commands **Save All** and **Close All** which you can use when you are working with multiple files.

## Opening a file in the C/C++ editor

There are several ways to open an existing file. An easy way to open the C source file `myfile.c` directly in the C/C++ editor is:

- In the C/C++ Projects view, double-click on the file name.

*Eclipse recognizes the file as a C source file and opens the file in the C/C++ editor.*

- Correct the file by entering the missing semicolon. Save and close the file.

## Opening a file in a system editor

If you want to open a C source file in an application (editor) outside Eclipse (instead of the built-in C/C++ editor), proceed as follows:

- In the C/C++ Projects view, right-click on the file `myfile.c` and select **Open With » System Editor**.

*The file opens in the application that is associated with the file extension `.c`.*

## 2.5. Closing, Opening and Activating a Project

### Closing a project

Like files, you can close a complete project. To do so:

1. In the C/C++ Projects view, right-click on the project `myproject` and select **Close Project**.

*If there are unsaved files, the Save Resources dialog appears in which you can choose which modified files need to be saved before closing the project.*

2. Select the files you want to be saved and click **OK** to continue.

*Any selected unsaved files are saved first, then the project closes. In the C/C++ Projects view the project `myproject` is now visible as a closed map.*

### Opening a project

To reopen the project again:

- In the C/C++ Projects view, right-click on the project `myproject` and select **Open Project**.

*The project is open for modifications again. You may need to expand the project structure to view its contents.*

### Activating a project

The project related menu items and buttons act on the currently active project. When you create a new project, the new project automatically becomes the active project. If you want to work with another project, you have to make it active first.

To make a project active:

- In the C/C++ Projects view, right-click on the project `myproject` and select **Set Active Project**.

*The project is now active. This is mentioned after the project name. Project properties, build and debug will act on this active project.*

### 2.6. Clone a Project

If you want to use a project as a starting point for a new project, you can make a clone of a project. A "clone" of a project is like a copy of a project, but also the project specific filenames and settings will reflect the new project name.

To clone a project:

1. In the C/C++ Projects view, right-click on the project `myproject` and select **Clone Project...**

*The Clone Project dialog appears.*

2. Enter a name for your new project, for example `mycloneproject`.

*In the **Location** field you will see the location where the copy of your project will be stored. To change the default location, you can uncheck the **Use default location** check box and browse for an alternative location. However, use the default location for now.*

3. Click **OK** to continue.

*Eclipse makes a copy of the project, renames the LSL file and updates the project configuration and launch configurations. In the C/C++ Projects view the project `mycloneproject` is now visible as a closed map.*

### 2.7. Configuring the Target

In order to build your application, run it and debug it, your project needs information about the target execution environment. The target can be a simulator or an evaluation board. If you are using the TASKING simulator you do not need to configure a target, because when you created your project with the New C/C++ Project wizard, Eclipse already made a default simulator debug launch configuration for your project (`myproject.simulator.launch`)

If you are using an evaluation board you have to create a launch configuration for your board in order to debug on a board. For some boards you also need to import a board configuration into your project. Based on your selections your project settings are adjusted, such as memory settings and the linker script file (added LSL statements have the tag "dtc"). For a list of all boards for which this is necessary, refer to the *TASKING VX-toolset for ARM User Guide*.

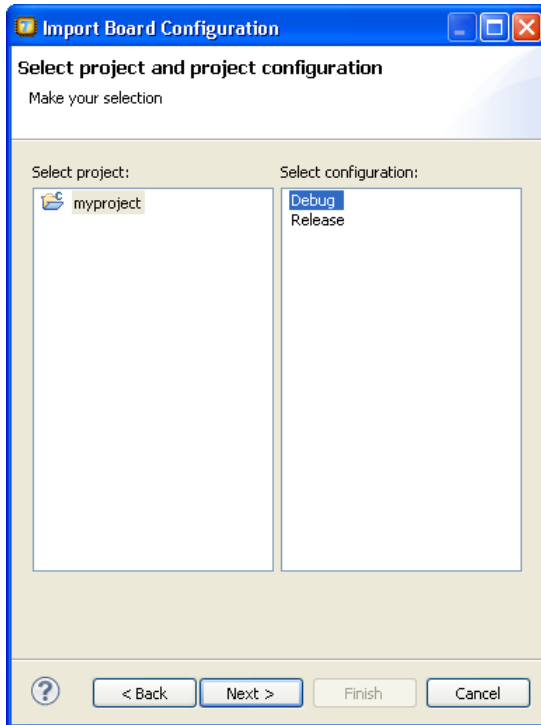
## Import ARM board configuration

1. From the **File** menu in Eclipse, select **Import**.

*The Import wizard appears.*

2. Expand **TASKING C/C++**, select **Board Configuration** and click **Next**.

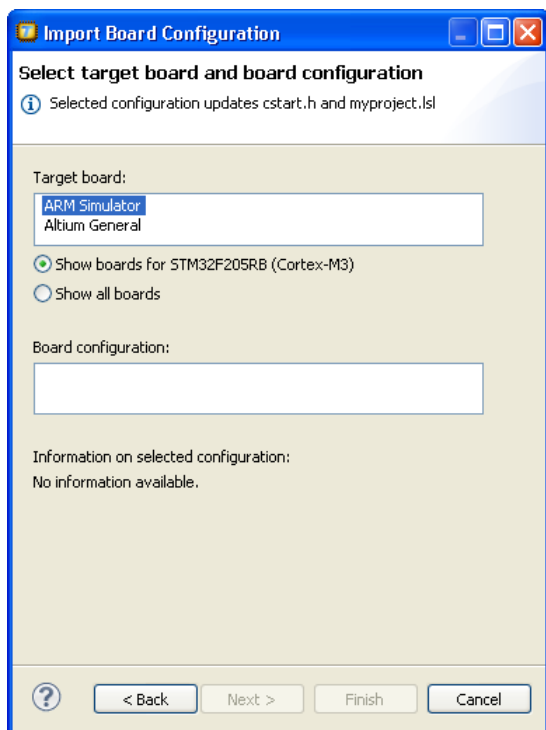
*The Import Board Configuration page appears.*



3. Select your project (`myproject`) and project configuration (`Debug`) and click **Next**.

*The Select target board and board configuration page appears.*

## Getting Started with the TASKING VX-toolset for ARM



4. In the **Target board** field, select the target evaluation board that you use to debug your application. By default only the boards are shown for the selected processor.
5. If your board supports multiple communication methods, select the **Communication** you want to use.
6. In the **Board configuration** field, select the configuration that matches the settings on your board.

*The text below this field shows information about the selected configuration.*

7. Click the **Finish** button to import the configuration settings.

*Your project settings, such as memory, flash settings and LSL file (myproject.lsl) are adjusted to the selected board configuration for you to build your application. Note that only those registers are changed that are needed for the board to operate.*

The information in the Import Board Configuration wizard is based on Debug Target Configuration (DTC) files. DTC files define all possible configurations for a debug target. The files are located in the `etc` directory of the installed product and use `.dtc` as filename suffix. For more information on DTC files, see the *TASKING VX-toolset for ARM User Guide*.

## 2.8. Setting Project Options

Now you are familiar with opening and editing (files in) your project, and you have selected a target configuration, we will have a look at the options you can set for building your project.

First make sure the project `myproject` is open.

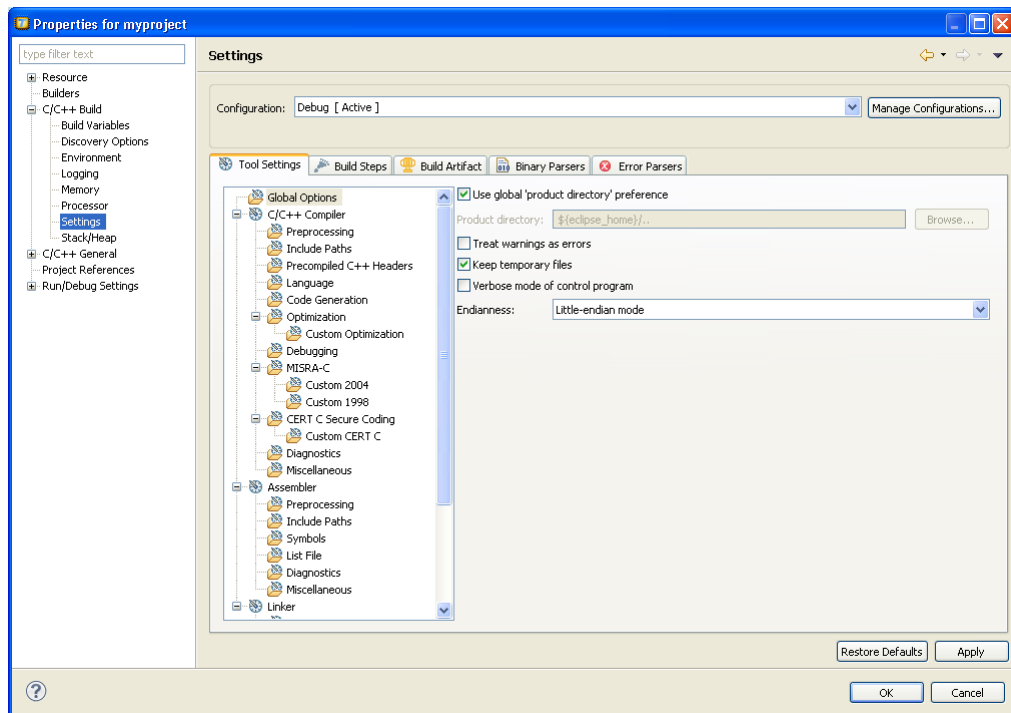
To access the options for your project:

1. From the **Project** menu, select **Properties for**. Alternatively, you can click the  button.

*The Properties for myproject dialog appears.*

2. If not selected, expand **C/C++ Build** and select **Settings** to access the ARM tool settings.

*A screen similar to the following should now appear.*



On the **Tool Settings** tab, the options are grouped in Global Options, C/C++ Compiler, Assembler and Linker or Archiver if you are building a library. Note that the options you enter in the Assembler page are not only used for hand-coded assembly files, but also for the assembly files generated by the compiler. In the **Configuration** field, you can choose a configuration for which you want to make changes. Note that this does not make the configuration active.

For a detailed description of all TASKING VX-toolset for ARM options refer to the *TASKING VX-toolset for ARM User Guide*.

## Selecting a predefined build configuration

A build configuration is a predefined set of options. When you created the sample project `myproject` as described in [Section 2.1, Create a Project](#), you should be able to choose between the **Debug** and the **Release** configuration. Both have their own settings. Check this for your self:

1. Select the **Release** configuration.  
*The option **Generate symbolic debug information** is set to **None**.*
2. Expand the **C/C++ Compiler** entry and select **Debugging**.  
*The option **Generate symbolic debug information** is set to **Default**.*

## Setting options and restoring defaults

You can use one of the available configurations as starting point for setting your options. For now, choose the Debug configuration.

1. Change the option **Default** to **Full**.  
*At this point you can change as many options as you like.*
2. Click **Apply** to apply the new setting(s) to your project.  
*The dialog does not close, but the new options are saved to the Debug configuration.*

To restore to the default Debug configuration options:

1. Click the **Restore Defaults** button.  
*The option settings are changed to the default settings of the chosen configuration.*
2. Click **Apply** to apply the default settings to your project.

If you change options without applying them and you try to change the configuration, you are asked whether to apply the changes first.

## Creating your own build configuration

Because of the amount of possible options, it may be very convenient to create your own build configuration.

1. Click on the **Manage Configurations...** button next to the **Configuration** field.  
*The Manage Configurations dialog appears.*
2. Click on the **New...** button.  
*The Create New Configuration dialog appears.*

3. Type a **Name** (`Myconfig`) and optional a **Description** for your configuration.

*In the **Copy settings from** box, you can choose the initial option settings for your configuration:*

4. Select **Existing configuration** and choose the **Debug** configuration.

*The existing Debug configuration is the same as the default Debug configuration because we applied the default settings in the previous example.*

5. Click **OK**.

*The Manage Configurations dialog shows the new configuration.*

6. Select the new configuration (`Myconfig`) and click **Set Active**.

7. Click **OK**.

Your new configuration has become the active configuration. From now on, a build will use the option settings from the `Myconfig` configuration. Note that when you select a configuration from the **Configuration** field, this only affects the property pages; it does not make the configuration active.

Important: the **Restore Defaults** button is still associated with the default Debug configuration! Because the new configuration `Myconfig` is based on the Debug configuration, the defaults of the Debug configuration also apply to the `Myconfig` configuration.

## Creating your own defaults

The previous example showed how to create your own build configuration to store settings. However, it was impossible to return to your own defaults, only the original **Debug** and/or original **Release** defaults were available. Below it is described how you can create your own defaults. Basically, you create a configuration A in which you set your own defaults; then you create a new configuration B which will be based on configuration A:

If you have found a satisfying combination of option settings, you can create a configuration named `Mydefaults`.

1. First change the option settings to your own needs.
2. Repeat steps 1 through 7 of *Creating your own build configuration* but in step 3, type the name `Mydefaults`.

Normally, any settings you change from here, are saved to `Mydefaults`, thus losing your original defaults. To prevent this:

1. Repeat steps 1 through 3 of *Creating your own build configuration*, to create a second new configuration and name it `Myworkoptions`. The name suggests that this will be the configuration for experimentally changing option settings.
2. Select **Existing configuration** and choose the **Mydefaults** configuration.

*Your new 'working' configuration is now the same as the configuration named `Mydefaults`.*

## Getting Started with the TASKING VX-toolset for ARM

3. Click **OK**.

*The Manage Configurations dialog shows the new configuration.*

4. Select the new configuration (`Myworkoptions`) and click **Set Active**.
5. Click **OK**.

Now you can work with the `Myworkoptions` configuration. If you want to return to your defaults, you can either make the `Mydefaults` configuration active, or create a new configuration using the `Mydefaults` configuration to copy the settings from.

## 2.9. Build a Project

When you build a TASKING VX-toolset for ARM C/C++ project in Eclipse, the TASKING VX-toolset for ARM compiler, assembler and linker are used to compile and link all the source code and the libraries associated with the project.

To build a project:

- From the **Project** menu, select **Build myproject**.

From the **Project** menu, the following "Build" commands are available:

<b>Build Project</b>	Builds the selected project.
<b>Build Working Set »</b>	Opens a wizard in which you can create a customized set of files that will be built.
<b>Clean...</b>	Removes all intermediate files that are created during a build. As a consequence, the next build cannot rely on existing results from previous builds (thus simulating a rebuild).
<b>Build Automatically</b>	If you set this option, the selected project will be built automatically after each applied change in the project properties and after each saved change in the source files. This way of building is not recommended for C/C++ development. In order for this option to work, you must also enable option <b>Build on resource save (Auto build)</b> on the <b>Behaviour</b> tab of the <b>C/C++ Build</b> page of the <b>Project » Properties for</b> dialog.
<b>Build project</b>	Builds the active project.
<b>Rebuild project</b>	Rebuilds the active project. This builds every file in the project whether or not a file has been modified since the last build. A rebuild is a clean followed by a build.

## 2.10. Using the Sample Projects

The TASKING VX-toolset for ARM comes with a number of examples (delivered in the directory `<ARM installation path>\examples`). Each directory contains a file `readme.txt` with information about the example.

You can import the ARM examples via the Welcome page. This is an alternative for importing existing projects via the **File » Import » TASKING C/C++ » TASKING ARM Example Projects** wizard.

## Import an existing project from the Welcome page

1. From the **Help** menu, select **Welcome**.

*The Welcome page appears.*

2. Click the following button:



*The Welcome Samples page appears.*

3. Click **ARM examples**.

*The Import ARM examples dialog appears.*

4. Select the ARM examples you want to import into the current workspace.

5. Enable the option **Goto workbench after import** and click **Finish**

*The original examples are copied into the current workspace.*

The project(s) should now be visible in the C/C++ Projects view.

You can set additional project options and build the sample projects as explained in the previous sections.

Note that you can also use the Import Board Configuration wizard to configure the examples for an evaluation board.

## 2.11. Import/Export Project Properties

You can export project properties into a file (.prop), so that you can import a specific configuration into a project whenever you want (for example in another workspace).

### Export project properties

1. From the **File** menu, select **Export**.

*The Export dialog appears.*

2. Select **TASKING C/C++ » TASKING C/C++ Project Properties** and click **Next**.

*The Export TASKING C/C++ Project Properties dialog appears.*

3. Select the project and configuration from which you want to export the project properties.

## **Getting Started with the TASKING VX-toolset for ARM**

4. Specify the destination properties file (extension `.prop`) and click **Finish**.

*The properties will be saved in the specified file.*

## **Import project properties**

1. From the **File** menu, select **Import**.

*The Import dialog appears.*

2. Select **TASKING C/C++ » TASKING C/C++ Project Properties** and click **Next**.

*The Import TASKING C/C++ Project Properties dialog appears.*

3. Specify the properties file (extension `.prop`) you want to import.
4. Select the destination project and configuration into which you want to import the project properties and click **Finish**.

*The properties of the selected project will be replaced by the properties from the selected file.*

# Chapter 3. Debugging your Application

This tutorial shows how to debug your application using the internal debugger.

Before you start with this chapter, it is recommended to read the Eclipse documentation first. It provides general information about the debugging process. This chapter guides you through a number of examples using the TASKING debugger with simulation as target.

You can find the Eclipse documentation as follows:

1. Start Eclipse.
2. From the **Help** menu, select **Help Contents**.  
*The help screen overlays the Eclipse Workbench.*
3. In the left pane, select **C/C++ Development User Guide**.
4. Open the **Getting Started** entry and select **Debugging projects**.

This Eclipse tutorial provides an overview of the debugging process. Be aware that the Eclipse example does not use the TASKING tools and TASKING debugger.

## 3.1. Setting up a Project for Debugging

### 3.1.1. Create a Sample Project

1. Create or reopen the project `myproject` as created in [Section 2.1, Create a Project](#). Use the default values and make sure that you:
  - select **Hello World C project** in the New C/C++ Project wizard.
  - enable at least the **Debug** configuration.
2. Edit the file `myproject.c` as follows:

```
#include <stdio.h>

int main( void )
{
    int i;
    for (i=1; i<=3; i++)
    {
        printf( "%d\n",i );
    }
    printf( "Hello world, " );
    printf( "this is \n" );
    printf( "a small %dst\n",i-3 );
    printf( "debugging example.\n" );
}
```

## Getting Started with the TASKING VX-toolset for ARM

3. Save the file.
4. Build your project.

To be able to debug, it is essential that your project has been built properly!

All steps required above are demonstrated in [Chapter 2, Setting up a Project](#).

### 3.1.2. Create a Debug Configuration

Before you can debug a project, you need a Debug launch configuration. Such a configuration, identified by a name, contains all information about the debug project: which debugger is used, which project is used, which binary debug file is used, which perspective is used, ... and so forth.

When you created your project, you already have a default launch configuration for the TASKING simulator. At any time you can change this configuration. If you want to debug on a target board, you have to create a custom debug configuration for your target board.

For details on creating or changing a debug configuration, refer to section *Creating a Customized Debug Configuration* in Chapter *Using the Debugger* of the *TASKING VX-toolset for ARM User Guide*.

## 3.2. Start a Debug Session

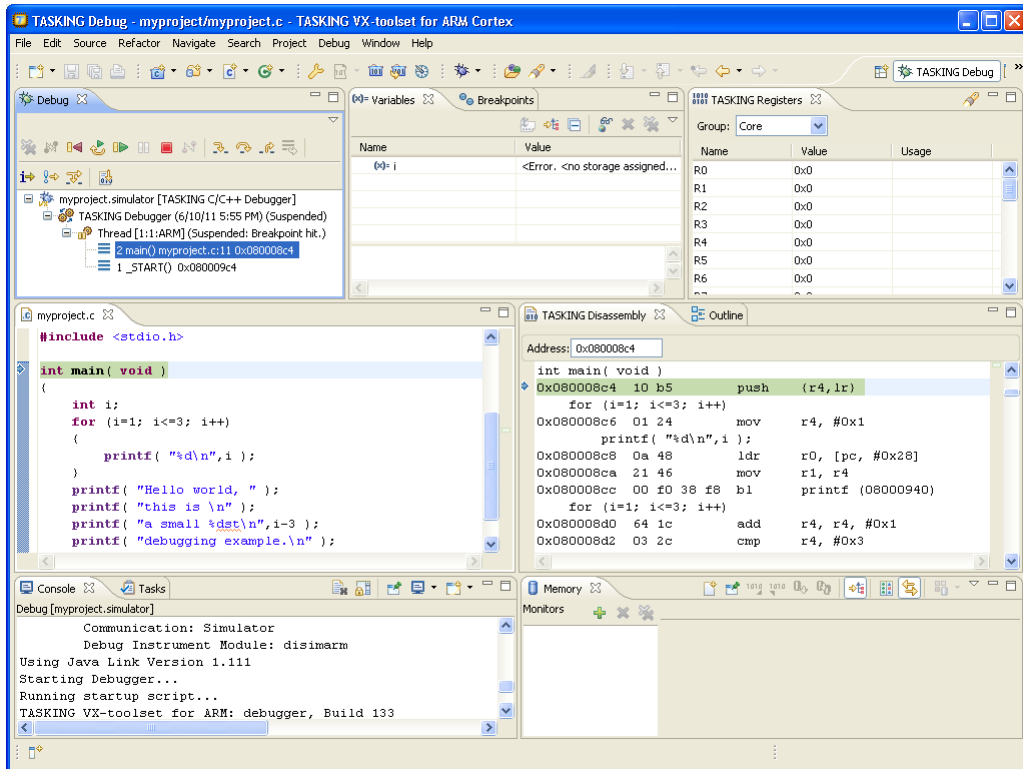
1. From the **Debug** menu select **Debug project**.

Alternatively you can click the  button in the main toolbar.

*The TASKING Debug perspective is associated with the TASKING C/C++ Debugger. Because the TASKING C/C++ perspective is still active, Eclipse asks to open the TASKING Debug perspective.*

2. Optionally, enable the option **Remember my decision** and click **Yes**.

*The debug session is launched. This may take a few seconds.*



- The Debug view shows your running application. Because of the settings in the debug configuration, execution has suspended at the first instruction in the function `main()`.
- The Editor view shows the C source files of your application and shows the line where the execution has suspended.
- The Variables view shows the variables in your application; in this case `int i`.

### 3.3. Stepping through the Application

At this moment your application is executing but suspended on the function `main()`. This means the C startup code has been executed already. From this point, you can step through your application while inspecting what happens.

1. From the **Debug** menu, select **Step Into**, or press **F5**, or click on the **Step Into** button (↵) in the Debug view.

*The highlight in the Edit view moves to the next statement.*

2. Press **F5** again.

*The highlight in the Edit view moves to the next statement.*

## Getting Started with the TASKING VX-toolset for ARM

*In the Variables view, you can inspect the value of the variable `i`. It is now set to 1.*

3. Press **F5** again.

*The `printf` statement has been executed now. The bottom area of your workbench now shows a new view: FSS # 1 - myproject.simulator.*

FSS stands for *File System Simulation*. The FSS view simulates the input and output to and from the target board or simulator when you are debugging. The value of `int i` is printed and sent to the FSS view for output.

To clear the FSS view, right-click in the view and select **Clear**.

To restart your application, in the Debug view, right-click on **TASKING Debugger** and select **Restart**.

4. Step further through your application.

*Watch the value of `int i` in the Variables view and observe the output in the FSS view. The output is only flushed after a newline (`\n`)!*

## 3.4. Setting and Removing Breakpoints



Instead of stepping, you can set breakpoints to suspended the application at a certain point.

A breakpoint is set on an executable line of a program. If a breakpoint is enabled during debugging, the execution suspends *before* that line of code executes.

### Add breakpoints

To add a breakpoint:

- Double-click the marker bar located in the left margin of the C/C++ Editor next to the line of code where you want to add a breakpoint.

*A dot  is displayed in the marker bar and in the Breakpoints view, along with the name of the associated file. When the breakpoint is actually set, a check mark  appears in front of the dot.*

### Disable breakpoints

You can disable a breakpoint or completely remove it. To disable a breakpoint, do one of the following:

- In the Breakpoints view, disable a breakpoint by clearing the check box.
- In the Editor view, right-click on a breakpoint dot in the margin and select **Disable Breakpoint**.

*The blue breakpoint dot turns white.*

## Remove breakpoints

To completely remove the breakpoint, do one of the following:

- In the Breakpoints view, right-click on a breakpoint and select **Remove**.
- In the Editor view, right-click on a breakpoint dot in the margin and select **Toggle Breakpoint**.
- In the Editor view, double-click on a breakpoint.


*The blue breakpoint dot disappears.*

## Example

With the techniques described above:

1. Set a line breakpoint on the code line `printf( "a small %dst\n",i-3 );`.
2. Clear the FSS view.
3. Restart your application.

*The application suspends when entering the `main()` function because this was defined in the Debug configuration.*

4. To resume execution, from the **Debug** menu, select **Resume**, or press **F8**, or click on the **Resume** button () in the Debug view.

*The application suspends execution, before this line is executed. The FSS view now shows:*



```
1
2
3
Hello world, this is
```

5. Resume execution again to finish execution.

*Note that though the application has finished execution, it has not been terminated yet. Your debug session is still active.*

## 3.5. Reload Current Application

When your application had changed, for example because you solved a bug, you can reload the application in the debugger without restarting it.



1. Make the necessary changes in your source.
2. Rebuild your application ()
3. Click on the **Reload current application** button () in the Debug view.

## Getting Started with the TASKING VX-toolset for ARM

*The new application is loaded in the debugger.*

### 3.6. End a Debug Session

To end the debug session:

1. From the **Debug** menu select **Terminate** or click on the **Terminate** button (  ) in the Debug view.
2. To remove the debug session from the Debug view, right-click on the debug session and select **Remove All Terminated** or click on the **Remove All Terminated Launches** button (  ) in the Debug view.

### 3.7. Multiple Debug Sessions

It is possible to run multiple debug sessions. To do so, just repeat the steps for starting a debug session. First make sure that you have terminated all debug sessions.

1. From the **Debug** menu, select **Debug Configurations...**

*The Debug Configurations dialog appears.*

2. Select the debug configuration `myproject.simulator` and click on the **Debug** button.

*The debug session launches.*

3. Repeat steps 1 and 2, but in step 2 choose `myproject.board`.

There are now two debug sessions for the same application. In case you have multiple projects, you can make dedicated debug configurations for them. You can use these debug configurations to run multiple debug sessions at the same time.

Each session uses its own FSS view for output. In the Debug view you can select the debug session (or file in the debug session) for which you want to inspect, for example, the value of its variables in the Variables view.