

# EFM<sup>®</sup>32

... the world's most energy friendly microcontrollers

## A free EFM32 development IDE based on Eclipse for Windows

AN0023 - Application Note

0 1 2 3 4

### *Introduction*

This application note will show you how to set up a simple project in Eclipse, how to generate the code and finally debugging the code on an EFM32 microcontroller.

This application note includes:

- This PDF document
- Eclipse support files (zip)
  - Zylind Embedded CDT plugin
  - Eclipse Embedded Systems Register View (EmbSys) plugin
  - Eclipse Cortex cpu register view patch

# 1 Prerequisites

You will need a "Gecko Starter Kit" (STK) or "Gecko Development Kit" (DK) and the following tools:

- Eclipse IDE for C/C++ Developers.
- CodeSourcery Sourcery G++ Lite, the GNU toolsuite containing *gcc* compiler/linker and *gdb* debugger.
- Zylind Embedded CDT plugin, makes debugging of embedded targets with Eclipse easier.
- Eclipse Embedded Systems Register View plugin (EmbSys). Eclipse view of peripheral registers of all Energy Micro EFM32 parts.
- SEGGER *J-Link software and documentation pack* which contains SEGGER's J-Link GDB Server. The GDB server interfaces the *gdb* debugger to the target debug probe for EFM32 parts.
- Energy Micro's *EFM32 energyAware Software*. It contains the SW examples and *energyAware Commander* which is used to program the flash memory of EFM32 parts.

## 1.1 A note on versions

To succeed in making your complete setup work, it is important to notice that since there are many different tools from different sources involved, you should use the versions shown in Table 1.1 (p. 2). Once you have a stable setup, you can experiment using other tool versions. Microsoft Windows 7 Professional (64bit) was used to prepare this application note.

**Table 1.1. Tool version information**

Tool name	Version
Eclipse IDE for C/C++ Developers	Version: 3.4.2 Build id: M20090211-1700 (Ganymede)
CodeSourcery Sourcery G++ Lite	2010q1-188 gcc version 4.4.1 gdb version 7.0.50.20100218-cvs
Zylin Embedded CDT	4.15.1
Eclipse Embedded Systems Register View	0.1.6
SEGGER J-Link GDB Server	V4.20h
Energy Micro energyAware Commander	2.0

## 1.2 Installing the tools

### 1.2.1 J-Link software and documentation pack

Install the *J-Link software and documentation pack*. It is available from:

<http://www.segger.com/cms/jlink-software.html>

Unzip the file and run the installer executable. You might get prompted if you have an older version of JLinkARM.dll installed on your machine. You should select to overwrite older versions.

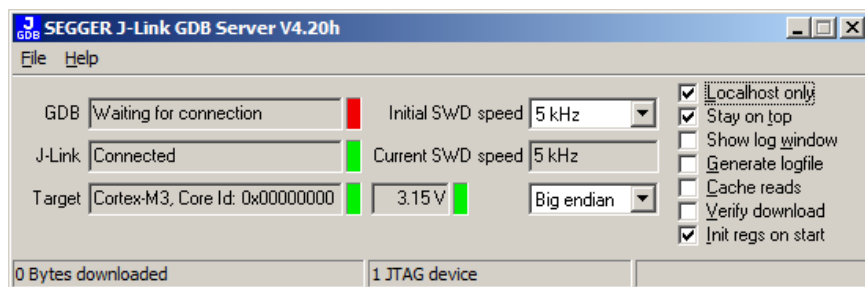
It is now time to connect the STK to your machine via the supplied USB cable and check that the J-Link device driver was properly installed.

Create a shortcut on your Desktop to SEGGER's *J-Link GDB server*. You will find this program in Windows' Start menu under *SEGGER->J-Link ARM V4.20h*. Modify the shortcut properties to:

"C:\Program Files (x86)\SEGGER\JLinkARM\_V420h\JLinkGDBServer.exe" -if SWD

The extra parameters direct the GDB server to use SWD instead of JTAG when communicating with the microcontroller. If everything is OK, the server GUI should look like:

**Figure 1.1. J-Link GDB Server GUI**



Do not exit this program before continuing.

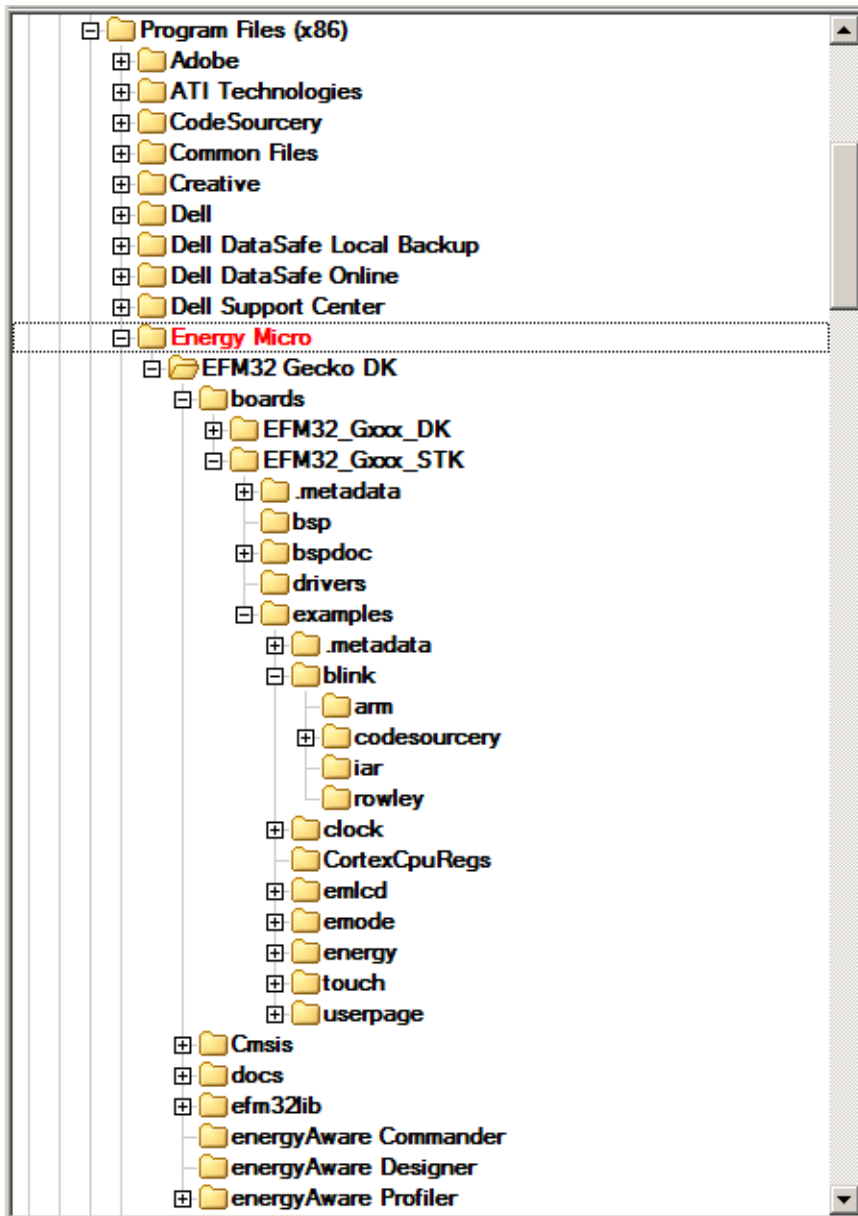
## 1.2.2 Energy Micro example SW and tools

Install the package *energyAware EFM32 Gecko DK Installer*. A complete software and documentation bundle for DK and STK. It is available from:

<http://www.energymicro.com/downloads/software>

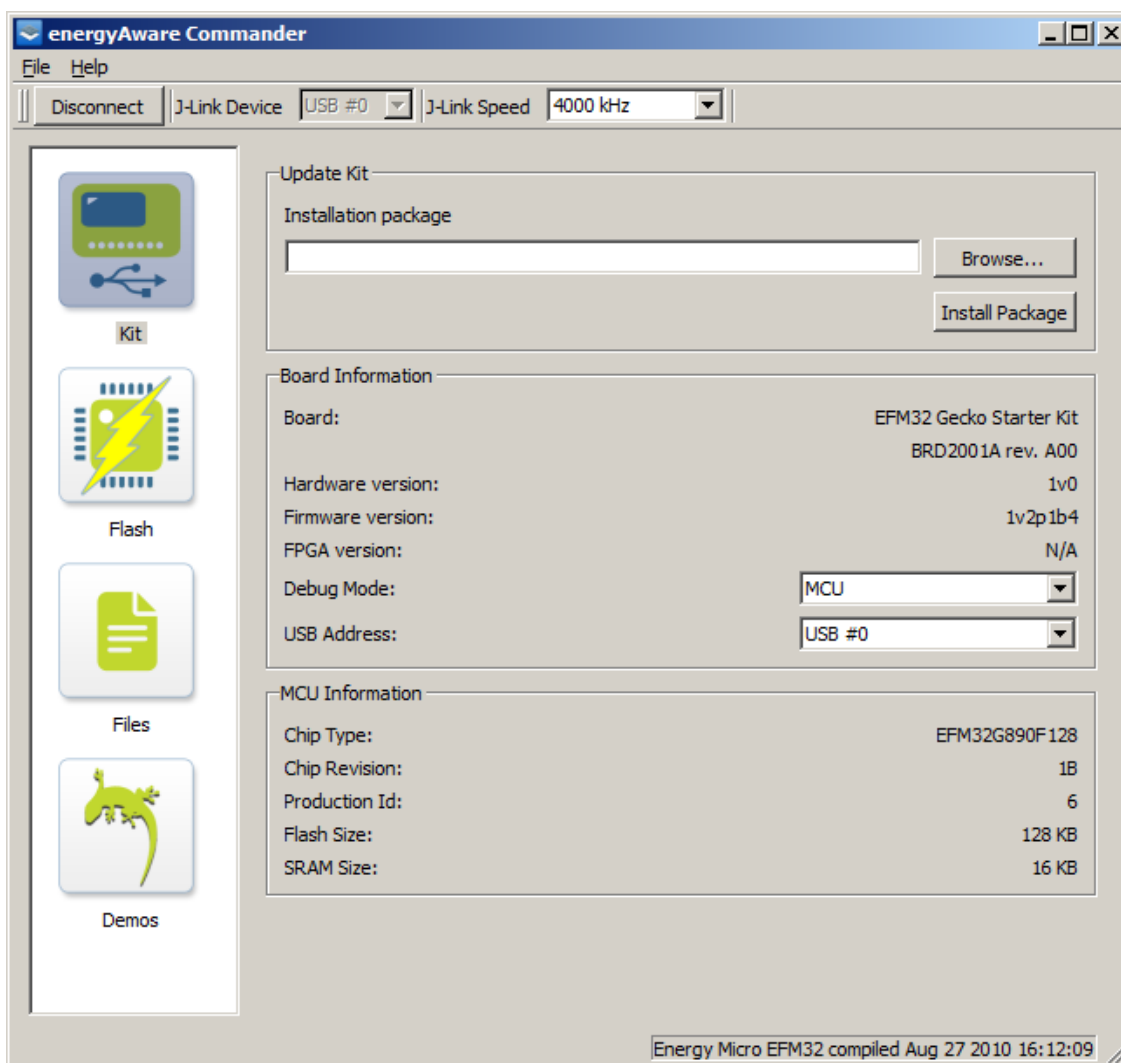
Unzip the file and run the installer executable. Files will be installed as shown in the figure below. Pay special attention to the *blink* example and *energyAware Commander*, as we will use these later in this application note.

Figure 1.2. energyAware EFM32 Gecko DK file structure



To verify target connectivity, start *energyAware Commander*. You will find this program in Windows' Start menu under *Energy Micro->energyAware Tools*. We will be using this tool later to download code to the STK's flash memory. Try the *Connect* button and verify that *Board Information* and *MCU Information* frames are updated with meaningful information.

Figure 1.3. energyAware Commander, Connect status



Exit *energyAware Commander*.

### 1.2.3 CodeSourcery Sourcery G++ Lite

Install the *CodeSourcery Sourcery G++ Lite* software package (make sure you download the Windows version). It is available from:

<http://www.codesourcery.com/sgpp/lite/arm/portal/release1294>

When prompted with a question to add CodeSourcery to the PATH environment variable, you should accept.

### 1.2.4 Eclipse IDE for C/C++ Developers

Eclipse is available from:

<http://eclipse.org/downloads/packages/eclipse-ide-cc-developers/ganymedesr2>

As Eclipse is a Java application you must have Java Runtime Environment (JRE) installed on your computer before installing Eclipse. You can check if Java is installed by issuing the command:

```
java -version
```

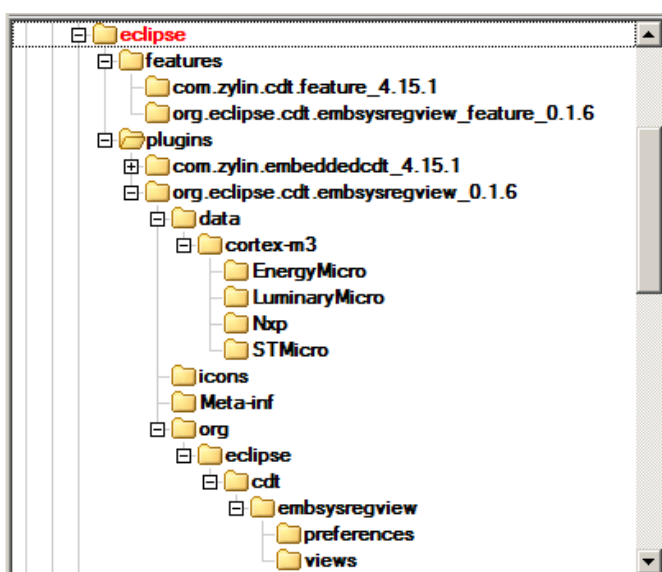
in a Command Prompt window, or you can visit <http://java.com> and click on the link named *Do I have Java?*

When downloading Eclipse make sure you select the Windows version. In this application note we use the Ganymede edition. Other versions tested:

- *Europa*. OK, briefly tested.
- *Ganymede*. OK, recommended.
- *Galileo*. OK, briefly tested.
- *Helios*. Does not work. We did not succeed in getting the new feature *GDB Hardware Debugging* to work.

Unpack the zip file at `c:\` and you have a full Eclipse installation at `c:\eclipse`. The two plugins we will be using can be installed by unzipping `an0023_efm32_eclipse_toolchain.zip` or by following the more official procedure outlined in the next two sections. If you elect to use the plugins in the zip file, copy the relevant files into eclipse's *features* and *plugins* subdirectories (under `c:\eclipse`). This directory structure is replicated inside the zip file.

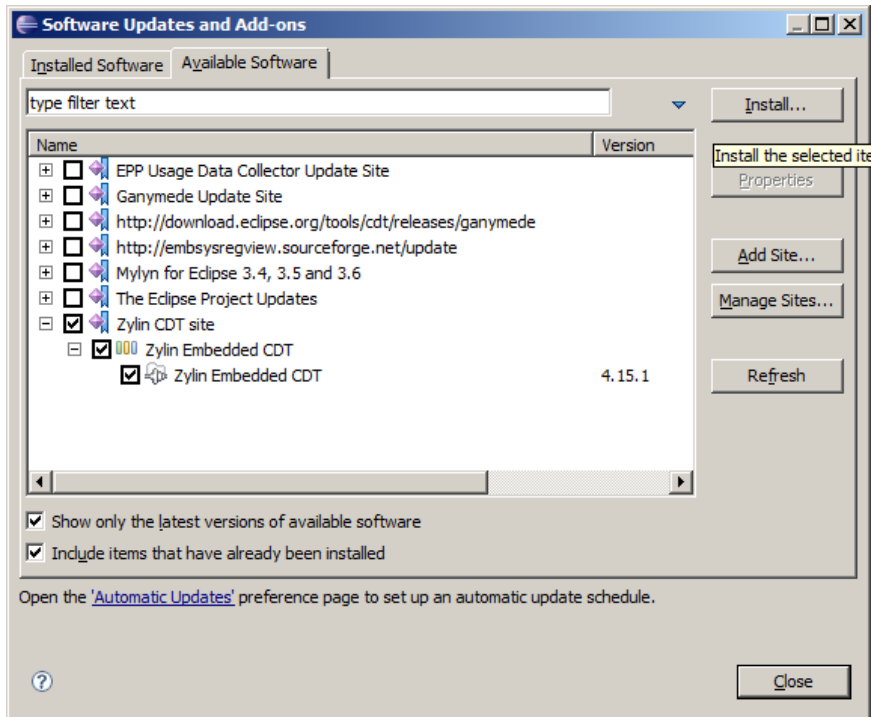
**Figure 1.4. Eclipse plugins file structure**



### 1.2.4.1 Zylin Embedded CDT plugin

Start Eclipse by executing `c:\eclipse\eclipse.exe`. When prompted for *workspace* select `C:\Program Files (x86)\Energy Micro\EFM32 Gecko DK\boards\EFM32_Gxxx_STK\examples` (we will need this later). You will now be greeted by Eclipse's welcome screen.

Select *Software Updates...* from the *Help* pulldown menu, select the *Available Software* tab and push the *Add Site...* button. Enter `http://opensource.zylin.com/zylincdt` as Location. After pressing the OK button, select Zylin and push the *Install...* button as shown in the figure below.

**Figure 1.5. Zylin Embedded CDT plugin installation**

The installation process may take a while... When asked for restarting Eclipse, do so.

#### 1.2.4.2 Eclipse Embedded Systems Register View plugin

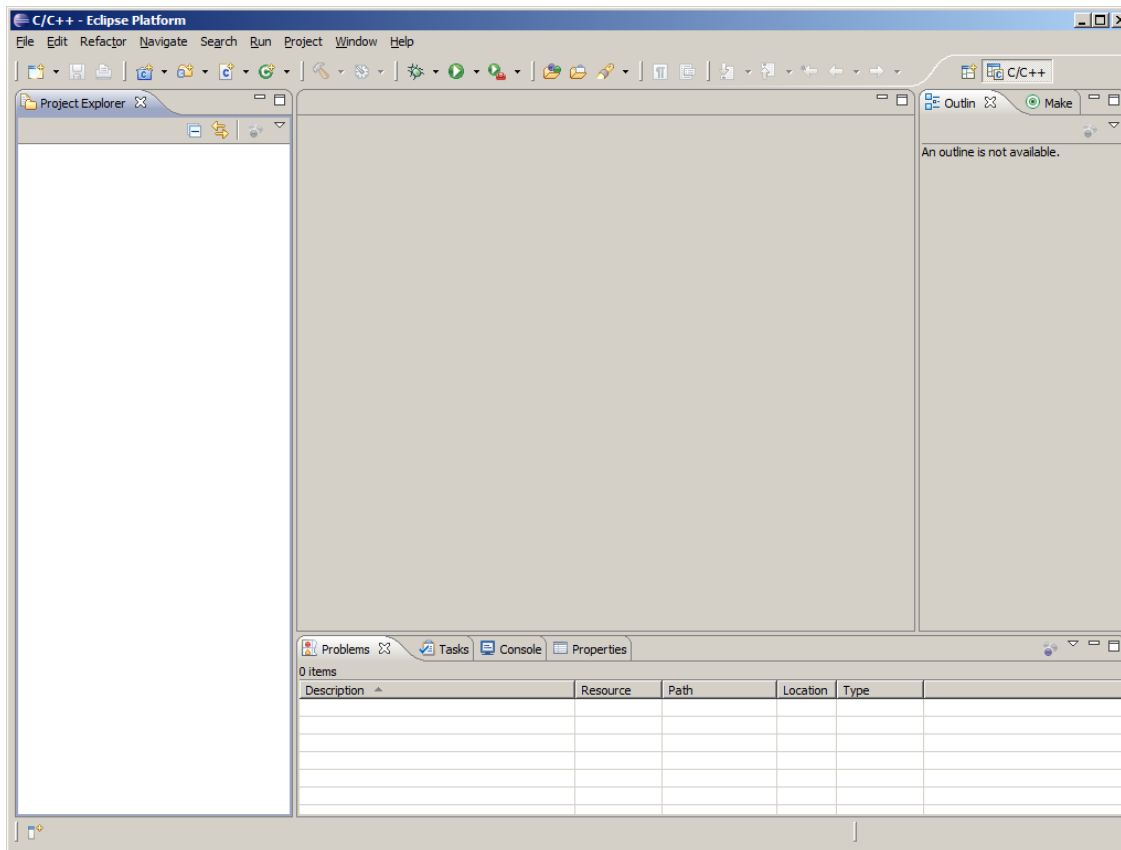
Proceed as described in the previous section, but use <http://embsysregview.sourceforge.net/update> as Location.

Restart Eclipse when asked to do so.

## 2 Working with Eclipse

Now it's time to continue with Eclipse. Close Eclipse's welcome screen tab and Eclipse's Workbench view appears:

**Figure 2.1. The Eclipse Workbench View**



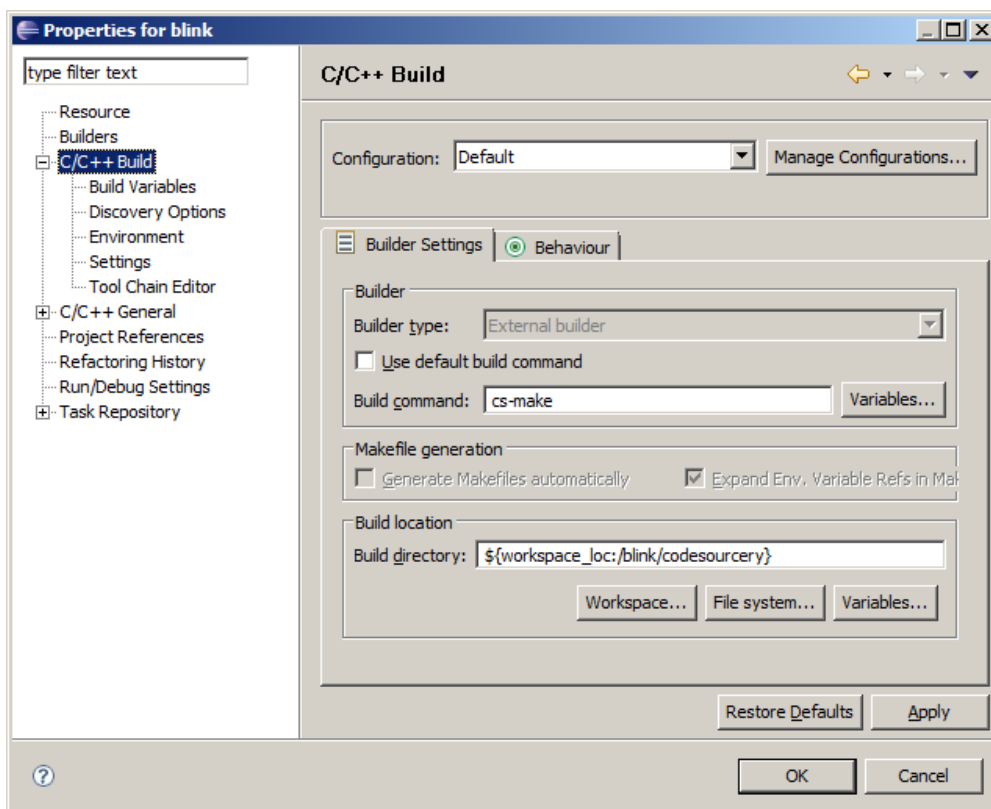
### 2.1 Create a project

We will create a project based on the blink example provided within the energyAware package already installed.

Create the project by selecting *File->New->C Project*. Use *blink* as project name (same name as the project directory). Make sure that your *Project type* is a *Makefile* project with *Toolchains: -- Other Toolchain --*. Hit the *Finish* button to complete the project definition.

Now we will fine-tune some project properties. Select *Project->Properties*:

Figure 2.2. The Eclipse Project Properties window



Do the following modifications:

- *C/C++ Build*: Do not use default build cmd, uncheck the checkbox and modify build command to `cs-make`.
- *C/C++ Build*: Modify build directory to `${workspace_loc:/blink/codesourcery}`.
- *C/C++ Build->Discovery Options*: Uncheck the *Report path detection problems* and *Automatic discovery of paths and symbols* checkboxes.
- *C/C++ Build->Settings*: Check the *GNU Elf Parser* checkbox.

Hit the *OK* button to complete project property fine-tuning.

We also need to make some changes in the makefile. Make a copy of the original *Makefile.blink* in the *blink/codesourcery* directory and save it with filename *Makefile*. These operations are easily performed by right-clicking on the files in the *Project Explorer* pane and selecting *Copy and Paste*. Open *Makefile* by double-clicking on it in the *Project Explorer*, then change:

```
TOOLDIR = 'C:/Program Files (x86)/CodeSourcery/Sourcery G++'
```

to

```
TOOLDIR = 'C:/Program Files (x86)/CodeSourcery/Sourcery G++ Lite'
```

Also check that the *CFLAGS* macro contains option *-O0*. The *-O<n>* option select code optimization level. Using *-O0* makes it easier to use the debugger. Also make sure that the project name defined with the *PROJECTNAME* macro has the same name as the project root directory:

```
PROJECTNAME = blink
```

## 2.2 Build the code

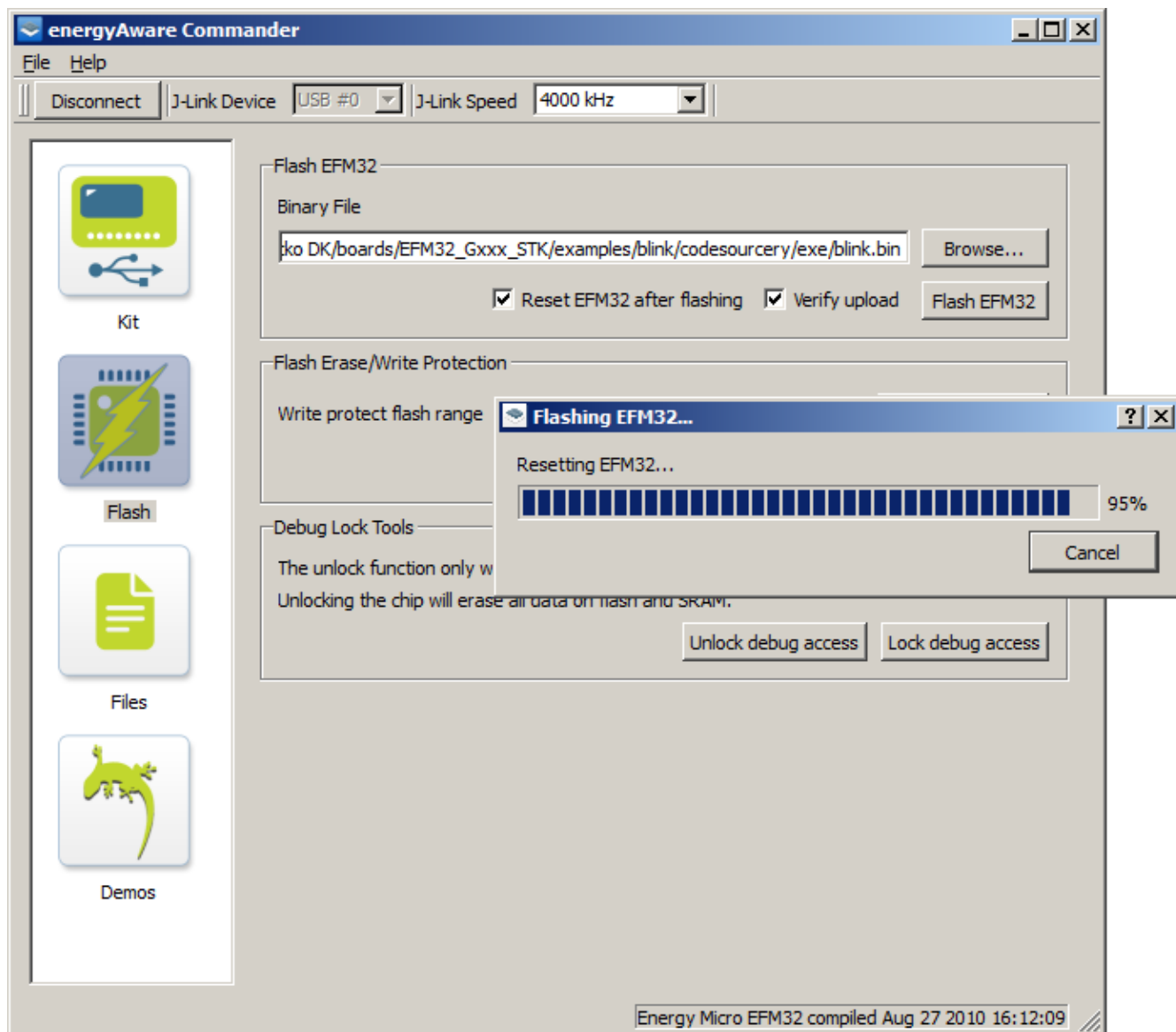
Now you are ready to build the code. Select *Build Project* or *Clean...* from the *Project* menu, or simply use *Ctrl-B* from the keyboard. The output from the build process is visible in the *Console* tab.

## 2.3 Download application code to target

To execute the application code on the target cpu, we must program the flash memory in the EFM32 microcontroller.

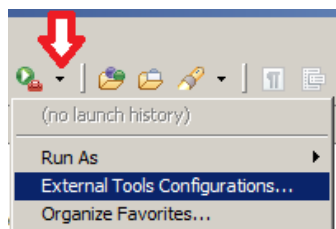
You can do this using *energyAware Commander*. Hit the flash symbol and browse to the correct location of your blink.bin file (it will reside in the *codesourcery/exe* subdirectory of your project. Hit the *Flash EFM32* button to download the application.

Figure 2.3. *energyAware Commander, flash command*



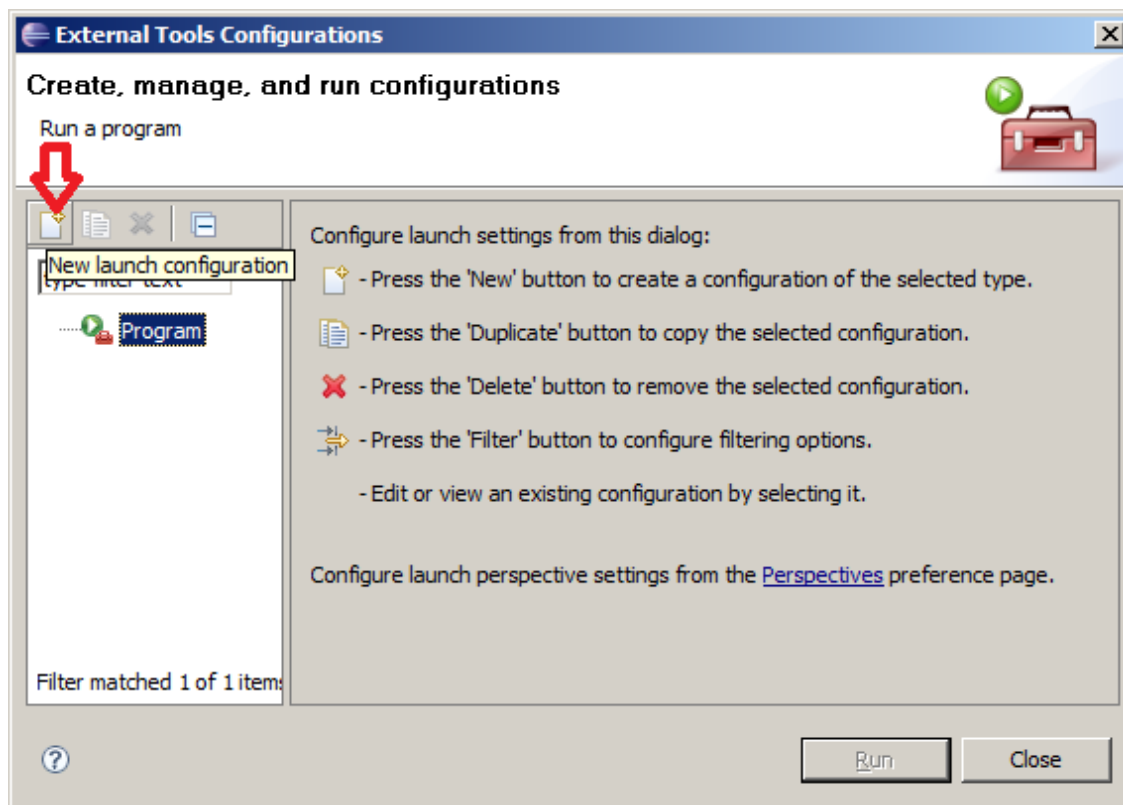
A more elegant approach is to define an *External Tools* configuration. Open the pulldown menu of the *External Tools* button:

Figure 2.4. *External Tools button*



and select *External Tools Configurations...*. Highligh *Program* and hit the *New launch configuration* button as shown below:

**Figure 2.5. Creating a new external tools launch configuration**

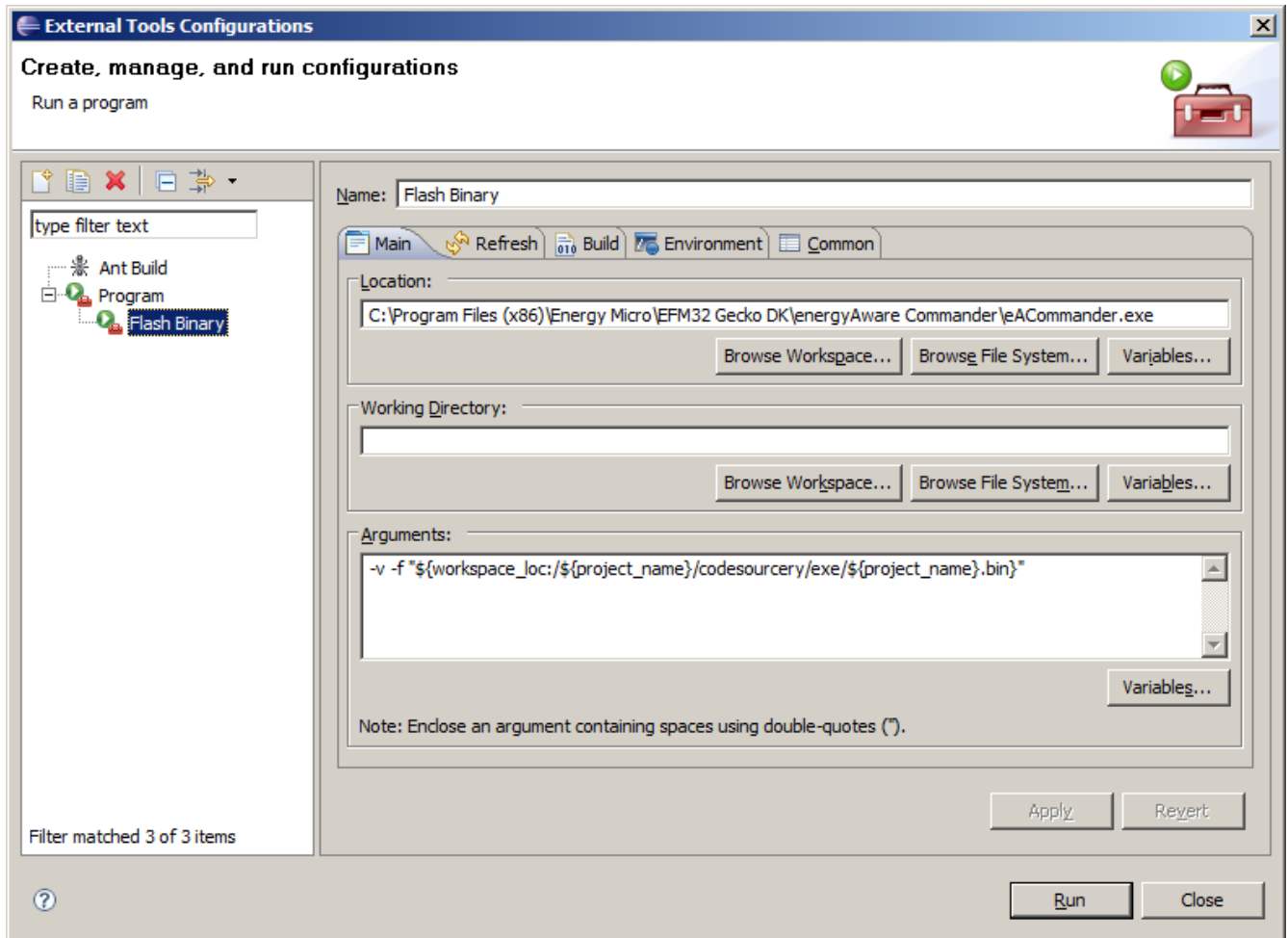


Fill in as shown in the figure below, cut and paste from here :

```
C:\Program Files (x86)\Energy Micro\EFM32 Gecko DK\energyAware Commander\eaCommander.exe
-v -f "${workspace_loc}/${project_name}/codesourcery/exe/${project_name}.bin"
```

- -v turns on verify
- -f specifies the file with the target binary.

Figure 2.6. External tools configuration details



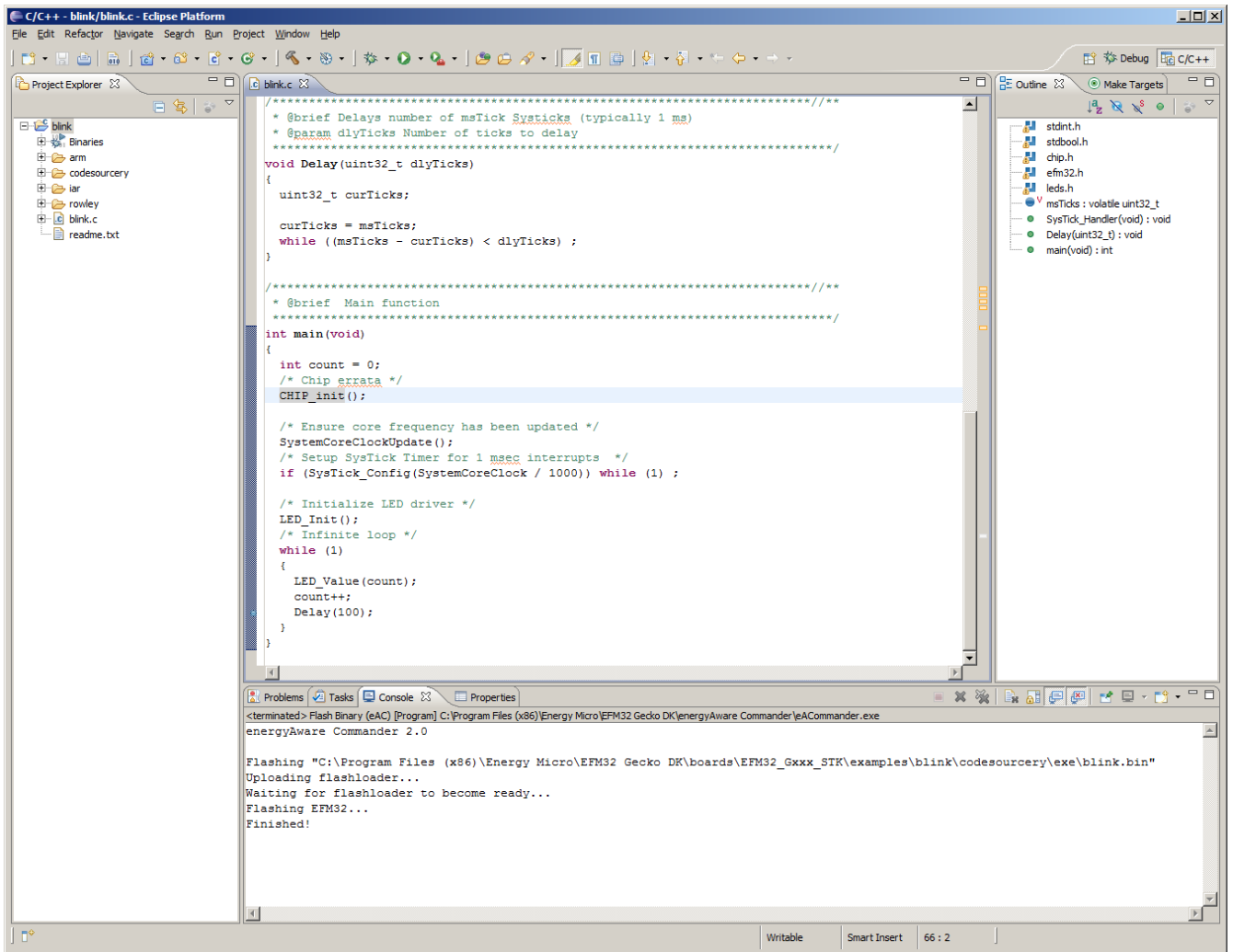
Do the following modifications in the other tabs:

- *Build* tab: Uncheck *Build before launch* checkbox.
- *Common* tab: Check *External Tools* checkbox in the *Display in favorites menu* frame.

Now you can download code by selecting *Flash Binary* from the *External Tools* button dropdown menu. Note that Eclipse will expand the argument macros correctly only if your project is highlighted in the *Project Explorer* tab. (Hint: You can open a *Project Explorer* tab in Eclipse *Debug* view too. More on debug later).

Check the output in the *Console* tab in the figure below. It shows the result of a successful flash operation.

Figure 2.7. Downloading code

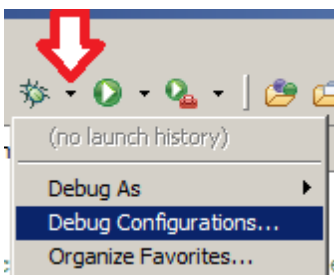


## 2.4 Debug application code

### 2.4.1 Create a debug launch configuration

To be able to debug the application, you must create a *Debug* configuration. Open the pulldown menu of the *Debug* button:

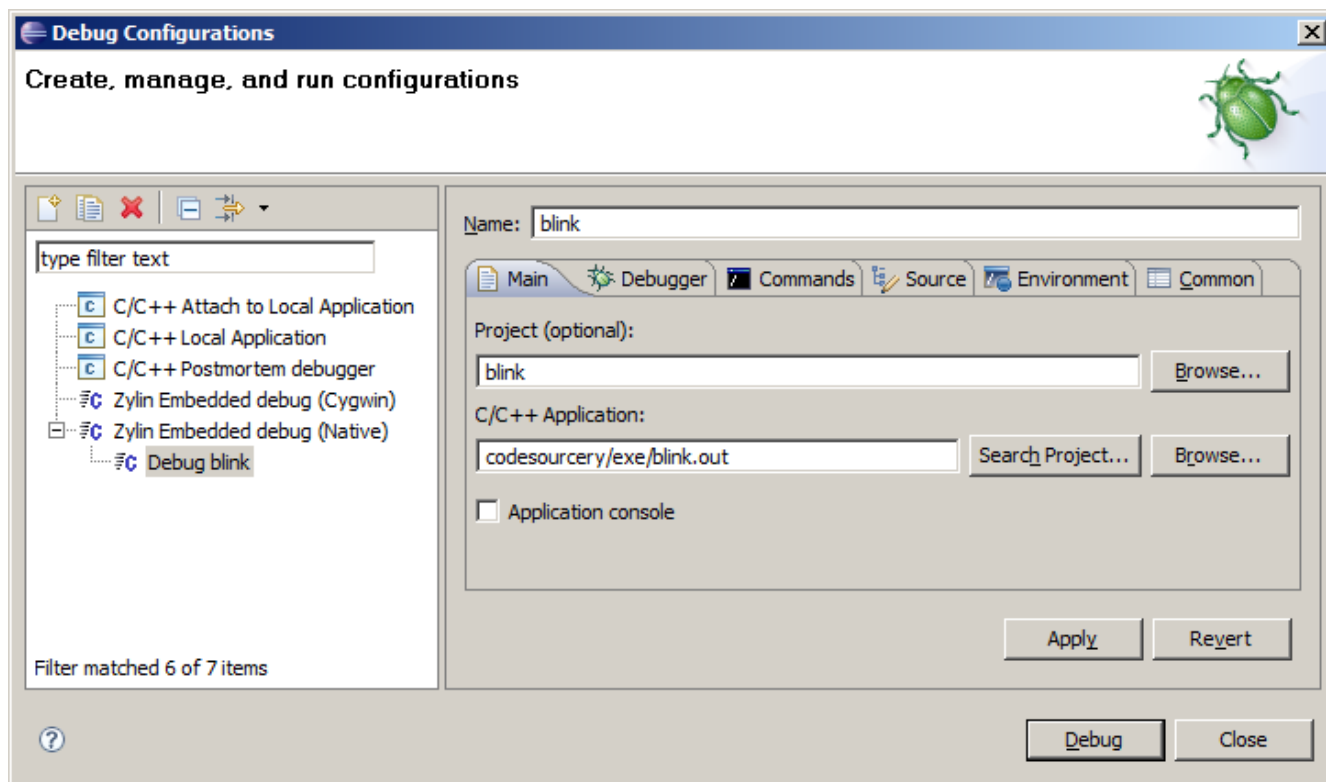
Figure 2.8. Debug button



and select *Debug Configurations...*. Highlight *Zylin Embedded Debug (Native)* and hit the *New launch configuration* similarly as you did when creating an *External Tools* configuration.

Fill in as shown below:

Figure 2.9. Debug launch configuration details



Do the following modifications in the other tabs:

- *Debugger* tab: Enter *arm-none-eabi-gdb* in the *GDB debugger* field.
- *Debugger* tab: Clear *GDB command file* field.
- *Commands* tab: Enter

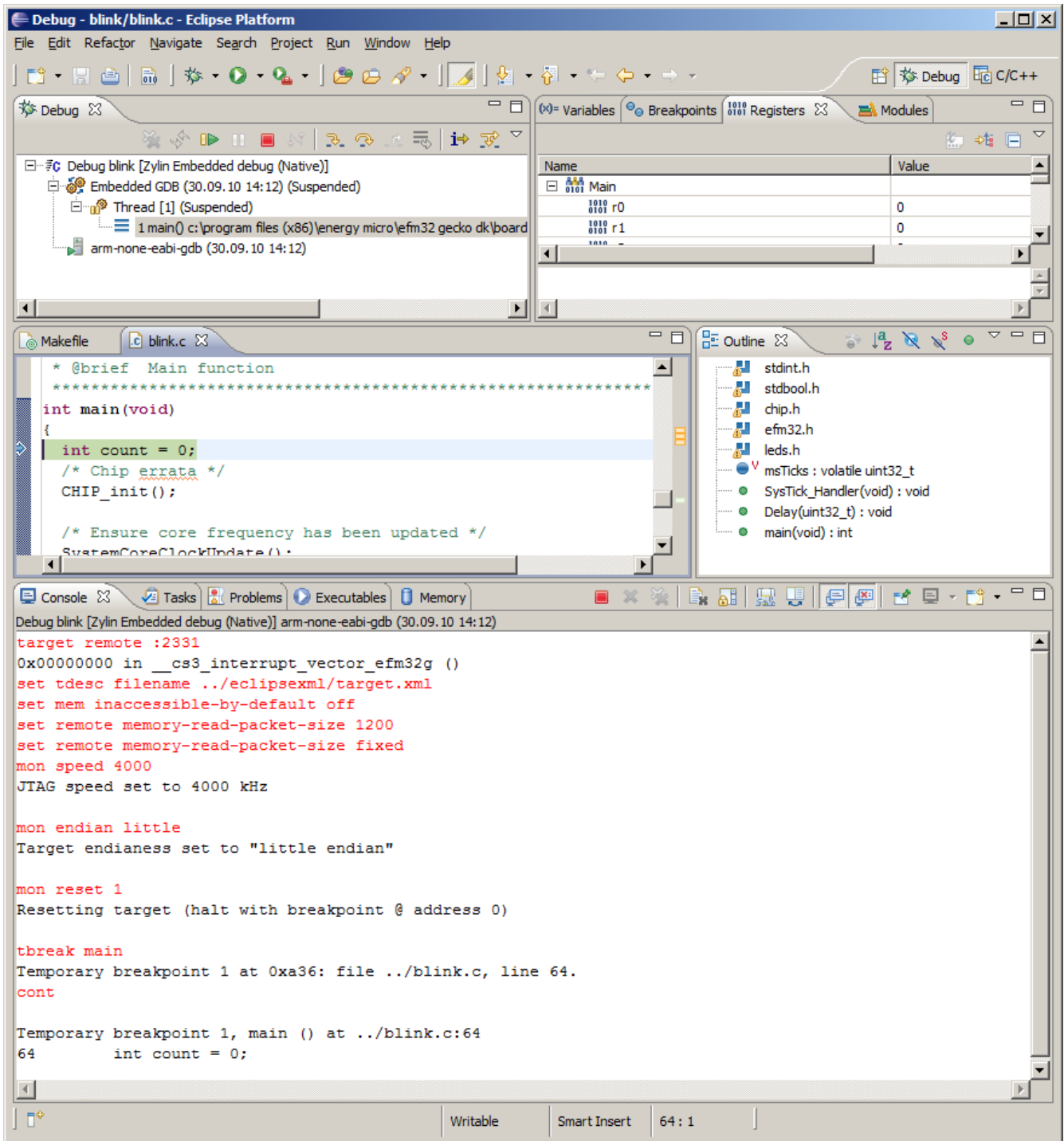
```
target remote :2331
set tdesc filename ../CortexCpuRegs/target.xml
set mem inaccessible-by-default off
set remote memory-read-packet-size 1200
set remote memory-read-packet-size fixed
mon speed 4000
mon endian little
mon reset 1
tbreak main
cont
```

in the *'Initialize' commands* field. Consult *gdb's* manual for an explanation of these commands, *mon* is shorthand for *monitor*. *Gdb* will pass monitor commands directly to the debug server. Consult SEGGER's *J-Link GDB server* manual for these commands. The *set tdesc filename* command makes Eclipse's *cpu register view* look nicer. The *xml-files* needed for this is included in *an0023\_efm32\_eclipse\_toolchain.zip*. Put them in a directory named *CortexCpuRegs* at the same directory level as your *blink* project directory.

- *Common* tab: Check *Debug* checkbox in the *Display in favorites menu* frame.

Now you can start debugging your code by selecting *blink* from the *Debug* button dropdown menu.

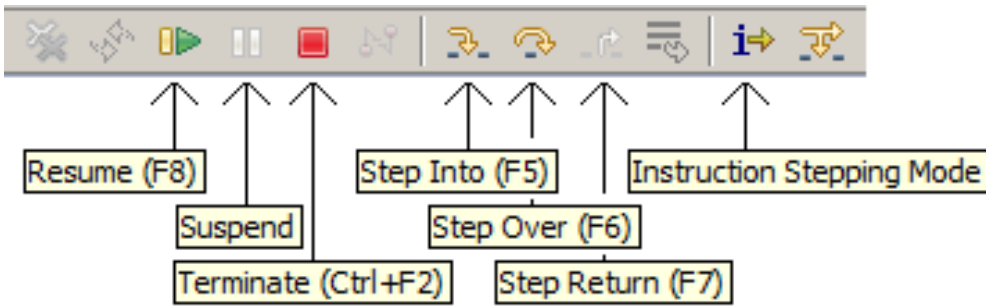
Figure 2.10. A typical debug session launch



### 2.4.2 Run, Stop, Single-Step, Breakpoints

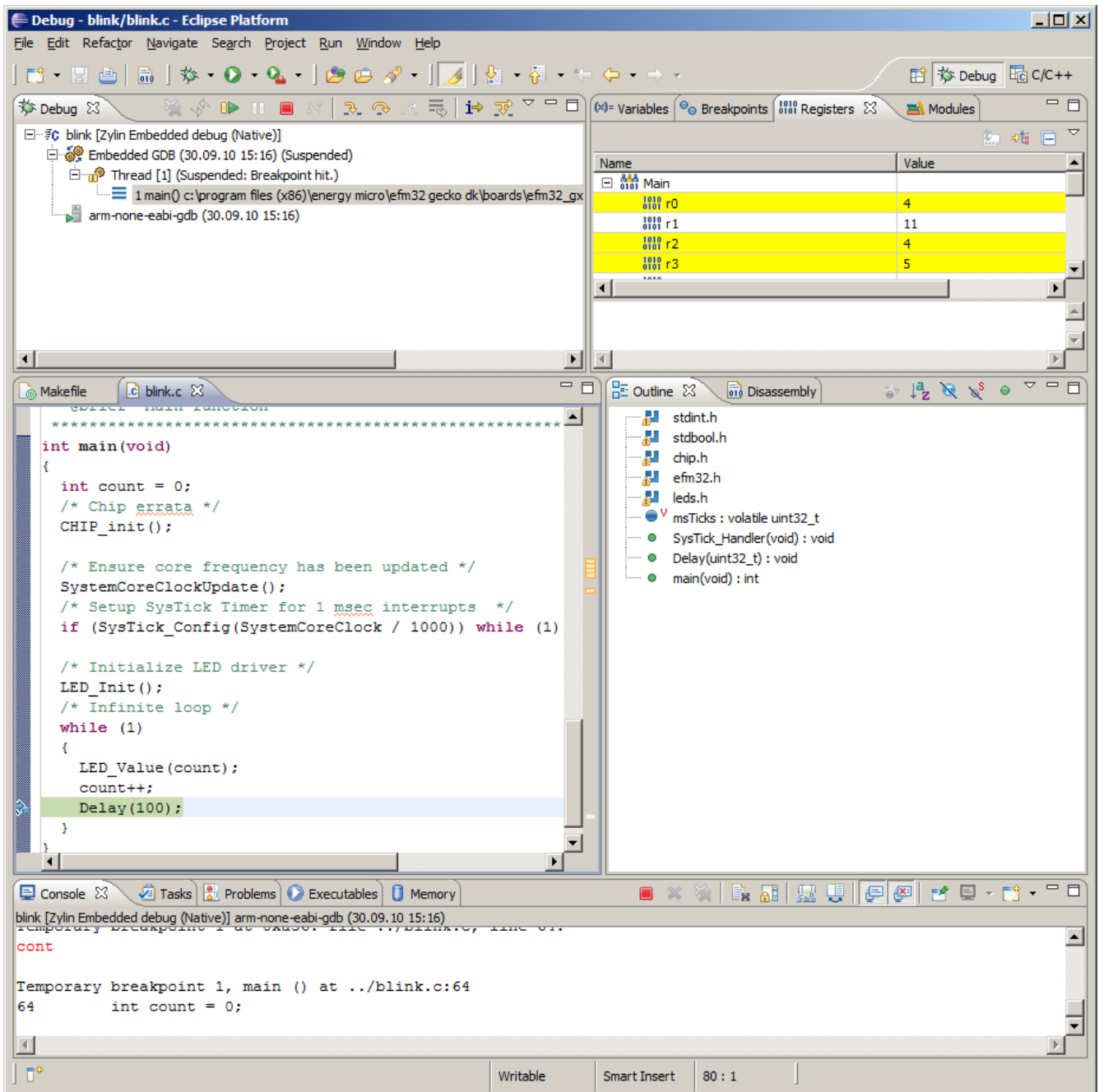
Look for the buttons shown in the figure below in the debug tab.

Figure 2.11. Debug button



These are all you need to do simple debugging. Breakpoints are set by doubleclicking in the left gutter in the source code tab. Set a breakpoint on the `Delay(100)` function call in the end of `main()` in `blink.c`, hit F8 several times and observe how the LED's on the STK blink. (Hint: You must *Terminate* before reflashing and starting a new debug session. Exit and restart the GDB server if you get stuck).

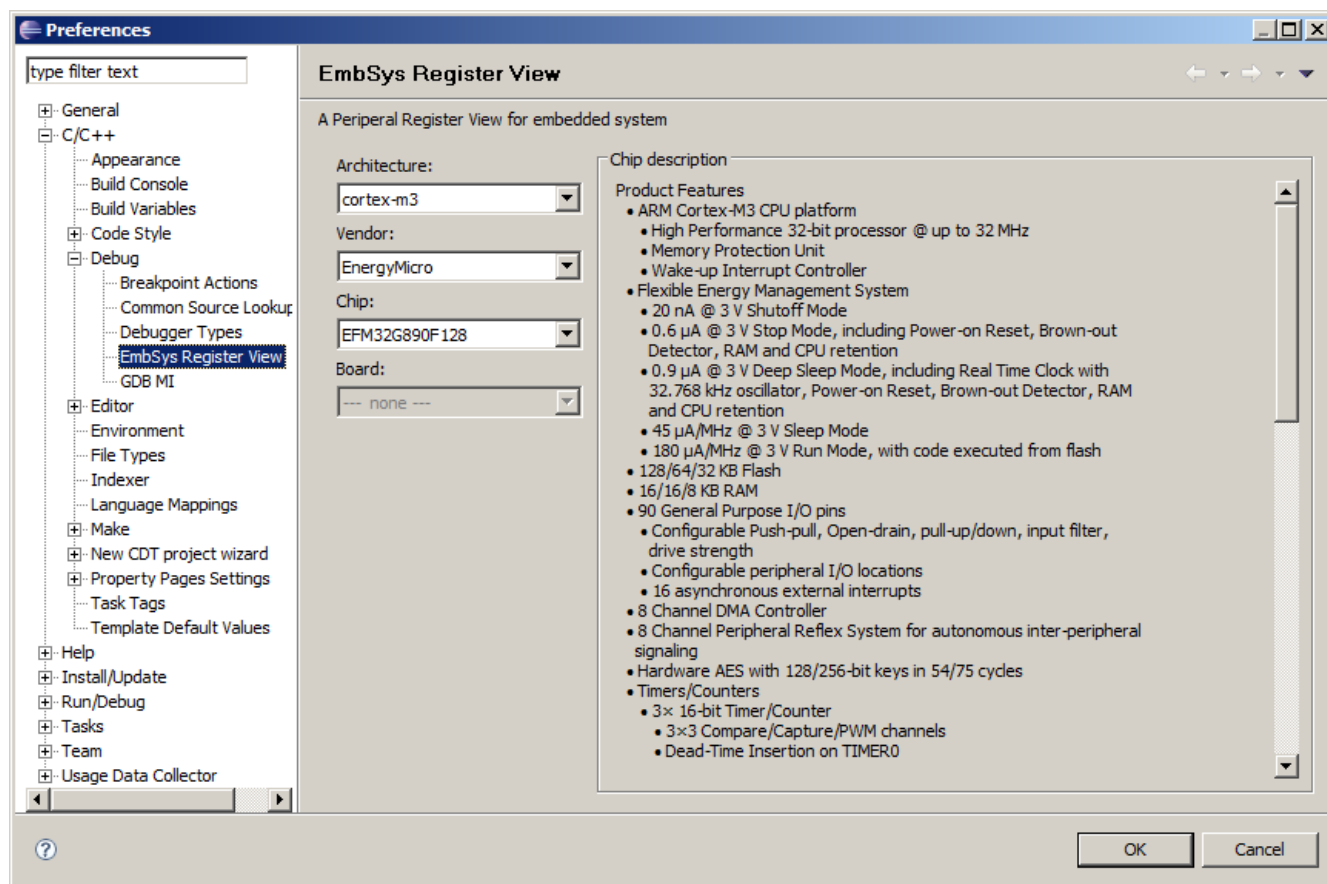
Figure 2.12. Debug with breakpoint



## 2.4.3 Using EmbSys Register viewer

The register viewer is a practical tool when you are debugging code for the peripherals of the microcontroller. The register viewer also contain documentation on peripheral registers and their bitfields (as tooltips). Open *Preferences* in Eclipse's *Window* pulldown menu and select the correct device as shown in the figure below.

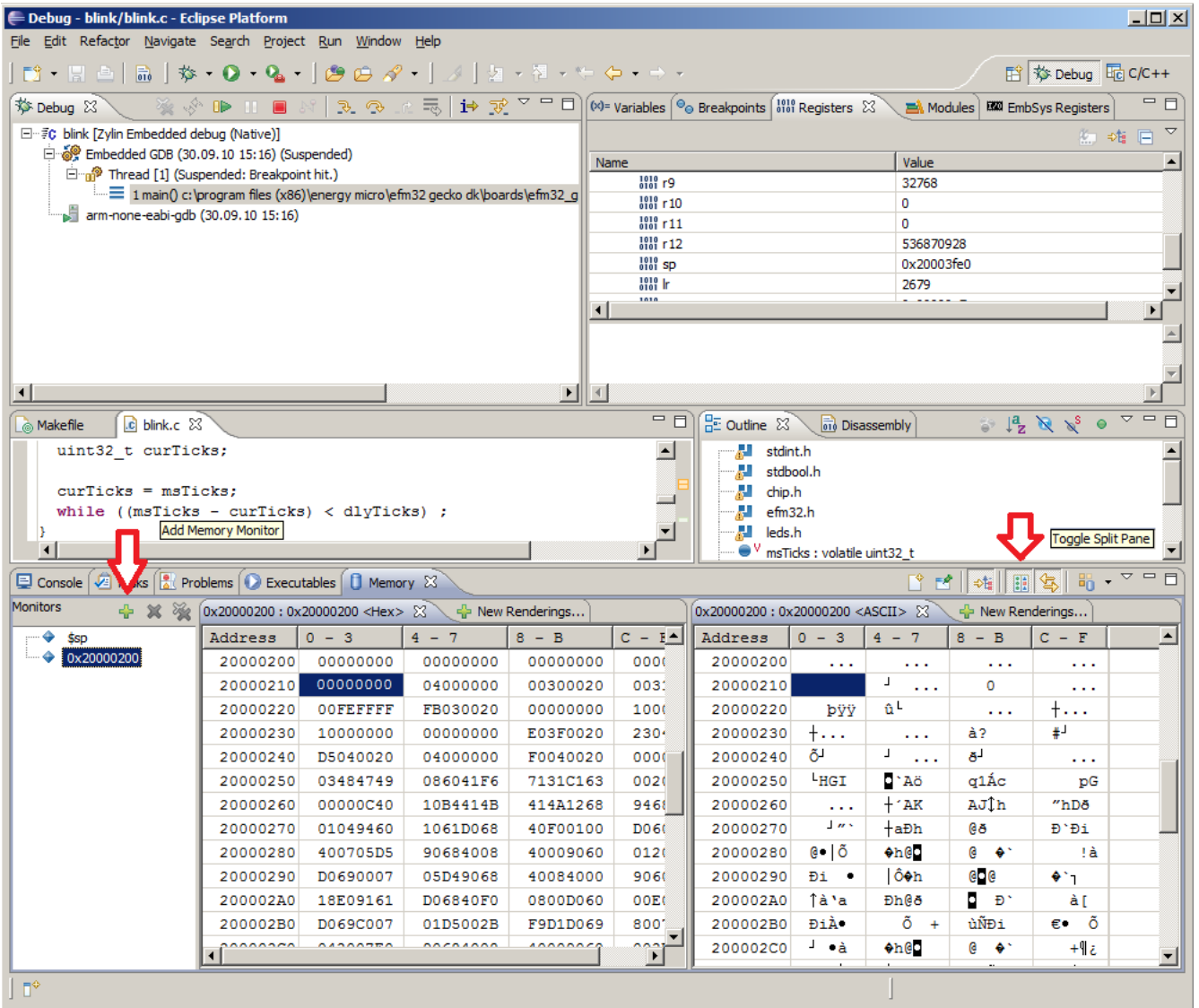
**Figure 2.13. EmbSys register viewer device selection**



To open a register viewer in the debug view, select *Show View -> Other...* in Eclipse's *Window* pulldown menu. Then expand the *Debug* node and select *EmbSys Registers*.



Figure 2.15. Eclipse Memory Monitor



## 3 Eclipse summary

This application note has gone through the basic steps of installing, configuring and using Eclipse as a code development environment for writing programs for EFM32 microcontrollers. Keep in mind that Eclipse is an advanced tool with virtually hundreds of menus with configuration options. The reader is encouraged to experiment further to finetune Eclipse. Use a search engine on the internet and you will find lots of articles, tutorials and books on using Eclipse.

## 4 Revision History

### 4.1 Revision 1.04

August 30th, 2011.

Fixed Zylind update link.

### 4.2 Revision 1.03

June 28th, 2011.

Fixed typo regarding EMEclipseSupport.zip to an0023\_efm32\_eclipse\_toolchain.zip renaming.

### 4.3 Revision 1.02

March 24th, 2011.

Changed the name of the zip file EMEclipseSupport.zip to an0023\_efm32\_eclipse\_toolchain.zip on section 1.2.4

### 4.4 Revision 1.01

November 22nd, 2010.

Added comment about the necessity of having Java installed before installing Eclipse.

### 4.5 Revision 1.00

November 10th, 2010.

Initial revision.

# A Disclaimer and Trademarks

## A.1 Disclaimer

*Energy Micro AS intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Energy Micro products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Energy Micro reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Energy Micro shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Energy Micro. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Energy Micro products are generally not intended for military applications. Energy Micro products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.*

## A.2 Trademark Information

Energy Micro, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Energy Micro AS. ARM, CORTEX, THUMB are the registered trademarks of ARM Limited. Other terms and product names may be trademarks of others.

## B Contact Information

### B.1 Energy Micro Corporate Headquarters

Postal Address	Visitor Address	Technical Support
Energy Micro AS P.O. Box 4633 Nydalen N-0405 Oslo NORWAY	Energy Micro AS Sandakerveien 118 N-0484 Oslo NORWAY	support.energymicro.com Phone: +47 40 10 03 01

**www.energymicro.com**

Phone: +47 23 00 98 00

Fax: + 47 23 00 98 01

### B.2 Global Contacts

Visit **www.energymicro.com** for information on global distributors and representatives or contact **sales@energymicro.com** for additional information.

Americas	Europe, Middle East and Africa	Asia and Pacific
<a href="http://www.energymicro.com/americas">www.energymicro.com/americas</a>	<a href="http://www.energymicro.com/emea">www.energymicro.com/emea</a>	<a href="http://www.energymicro.com/asia">www.energymicro.com/asia</a>

## Table of Contents

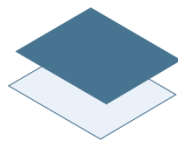
1. Prerequisites .....	2
1.1. A note on versions .....	2
1.2. Installing the tools .....	2
2. Working with Eclipse .....	8
2.1. Create a project .....	8
2.2. Build the code .....	9
2.3. Download application code to target .....	10
2.4. Debug application code .....	13
3. Eclipse summary .....	20
4. Revision History .....	21
4.1. Revision 1.04 .....	21
4.2. Revision 1.03 .....	21
4.3. Revision 1.02 .....	21
4.4. Revision 1.01 .....	21
4.5. Revision 1.00 .....	21
A. Disclaimer and Trademarks .....	22
A.1. Disclaimer .....	22
A.2. Trademark Information .....	22
B. Contact Information .....	23
B.1. Energy Micro Corporate Headquarters .....	23
B.2. Global Contacts .....	23

## List of Figures

1.1. J-Link GDB Server GUI .....	3
1.2. energyAware EFM32 Gecko DK file structure .....	4
1.3. energyAware Commander, Connect status .....	5
1.4. Eclipse plugins file structure .....	6
1.5. Zylind Embedded CDT plugin installation .....	7
2.1. The Eclipse Workbench View .....	8
2.2. The Eclipse Project Properties window .....	9
2.3. energyAware Commander, flash command .....	10
2.4. External Tools button .....	10
2.5. Creating a new external tools launch configuration .....	11
2.6. External tools configuration details .....	12
2.7. Downloading code .....	13
2.8. Debug button .....	13
2.9. Debug launch configuration details .....	14
2.10. A typical debug session launch .....	15
2.11. Debug button .....	16
2.12. Debug with breakpoint .....	16
2.13. EmbSys register viewer device selection .....	17
2.14. EmbSys register viewer in action .....	18
2.15. Eclipse Memory Monitor .....	19

# List of Tables

1.1. Tool version information ..... 2



**ENERGY**<sup>®</sup>  
*micro*

*Energy Micro AS  
Sandakerveien 118  
P.O. Box 4633 Nydalen  
N-0405 Oslo  
Norway*

*[www.energymicro.com](http://www.energymicro.com)*