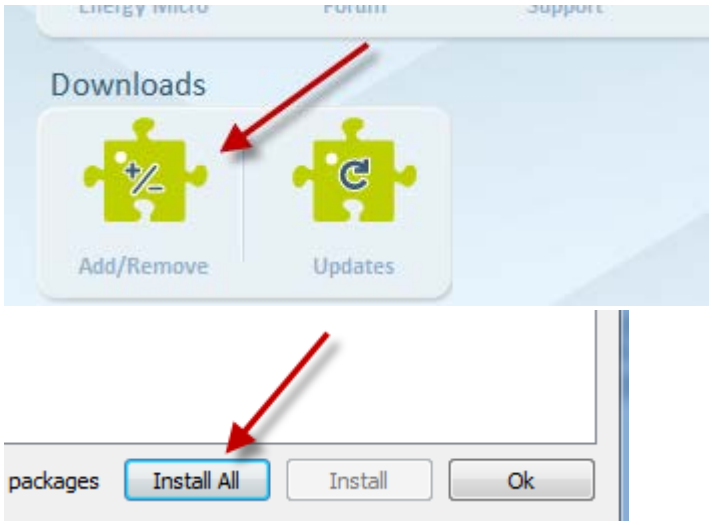
Introduction: This document will guide you through the installation process for a stand-alone installation of the SEGGER J-LINK GDB Server/Debugger, Eclipse IDE, and GNU/GCC Codesourcery compiler for the EFM32 MCU families. A CD with all of the items needed for installation is needed is included. Step-by-Step instruction provide the user with a guided installation of these tools.

NOTE: This installation guide makes use of the following versions of the various tools and software. Other versions may work together though it is recommended to use the versions included with the CD to ensure that everything works together initially before experimenting with different versions.

For this Install Guide Use the following:

Tool Name	Version
Eclipse ID for C/C++ Developers	Version: 3.7.2 Build id: 20120216-1857(Indigo)
Sourcery CodeBench Lite Edition for ARM GNU/Linux	Sourcery CodeBench Lite 2012.03-57
Segger J-Link GDB Server	V4.54a

1) **Install Simplicity Studio:** This can be more easily done if you have an internet connection but I have provided an option for doing this offline as shown below.
   1. Install Simplicity Studio by clicking on 'Simplicity_Studio_Setup.exe'. This is found on the install disk.
   2. If you do not have an internet connection then jump to 3. If you have internet connectivity then simply click on the 'Add/Remove' folder and once open click the 'Install All' button. This will take some time depending on your internet speed. This will ensure that everything is completely installed including Appnotes, Firmware Examples, etc.



   3. If you do not have an internet connection then follow what is copied below from our Lizard Lounge Forum.

      Note: The ***all_ss_packages.zip*** is on the install disk.

Posted 18 October 2012 - 04:18 PM

Normally Simplicity Studio connects to Energy Micro's web servers to check if there is new packages/material available to download. In some cases it can be useful to be able to update a Simplicity Studio installation without Simplicity Studio connecting to the internet itself. By following the steps below you will be able to download the latest packages for Simplicity Studio to a local storage and have Simplicity Studio check for updates from this location.

1. Install Simplicity Studio (Can be downloaded from here).
2. Download zip file of latest Simplicity Studio packages from here (**all_ss_packages.zip**). This file is updated as new packages are released for Simplicity Studio. Note that this zip-file is rather large.
3. Extract **all_ss_packages.zip**. (e.g. to **d:\packages**)
4. Open Simplicity Studio
5. Go to **File->Network Settings....**
6. Check **Use Alternate Download location** check box and fill in the path to the folder you extracted your all_ss_packages.zip into (e.g. **file:///d:/packages**). Note the "file:///" prefix in front of the path and make sure to use forward slash (/)
7. Press Yes to install any recommended packages if you are prompted
8. Go to **Add/Remove** and install any packages you want.
9. Go to **Updates** to check if there are any new versions of packages you already have installed.
10. Smile and enjoy an up-to-date Simplicity Studio installation

Simplicity Studio installs itself into the following locations for PC systems.

WIN7: C:\Users\'YOUR USER NAME'\AppData\Roaming\energymicro
WIN XP: C:\Documents and Settings\'YOUR USER NAME'\Application Data\energymicro

From here on out the directory preceding the energymicro folder will be referred to as **%APPDATA%.** So the directories above would look like what is shown below.

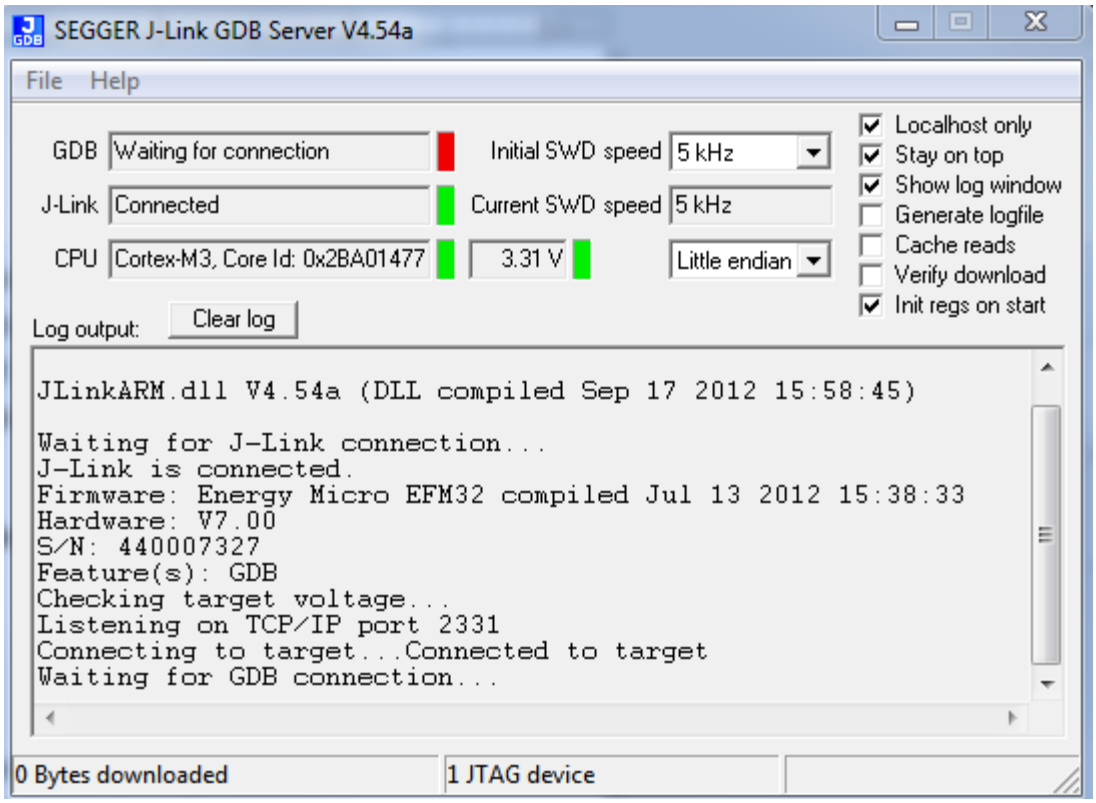
C:\%APPDATA%\energymicro

2) **Install Segger J-LINK GDB Server:** You will find '***Setup_JLinkARM_V454a.exe***' in the Setup_JLinkARM_454a folder. Double click on the executable to install the GDB Server.

After installing the J-LINK GDB Server we can now connect the STK or DK to our PC. We will verify the GDB Server is functioning.

1. Connect the STK/DK to your PC
2. From the PC 'Start Menu' browse to the SEGGER folder in 'Programs' and find the J-LINK ARM V4.54a and open the program called ***J-LINK GDB Server via SWD.***

After opening this if everything installed correctly you should see a GUI that looks similar to this…



LEAVE THIS PROGRAM RUNNING!!

3) **Install Sourcery CodeBench Lite(GCC/GNU Compiler):** Double click the ***'arm-2012.03-56-arm-none-eabi.exe'*** found on the install disk.

When prompted whether to add CODEBENCH to the PATH environment variable, you should accept!!

This is a FREE GNU/GCC compiler from Mentor Graphics. You can also download this from Mentor's site at the link below…
http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/editions/lite-edition/

The LITE version is Command Line only!! The Personal Edition shown below along with other fully supported Sourcery Codebench versions contain a much more integrated Eclipse IDE with project management and customization. The LITE version is FREE but provides no support.

For a fully supported version you can upgrade the LITE version to a version that you can get support for. Typical pricing for the Personal Edition is shown below.

4) Install Java: Eclipse is a Java application and for it to work your PC must have Java Runtime Environment(JRE) installed.

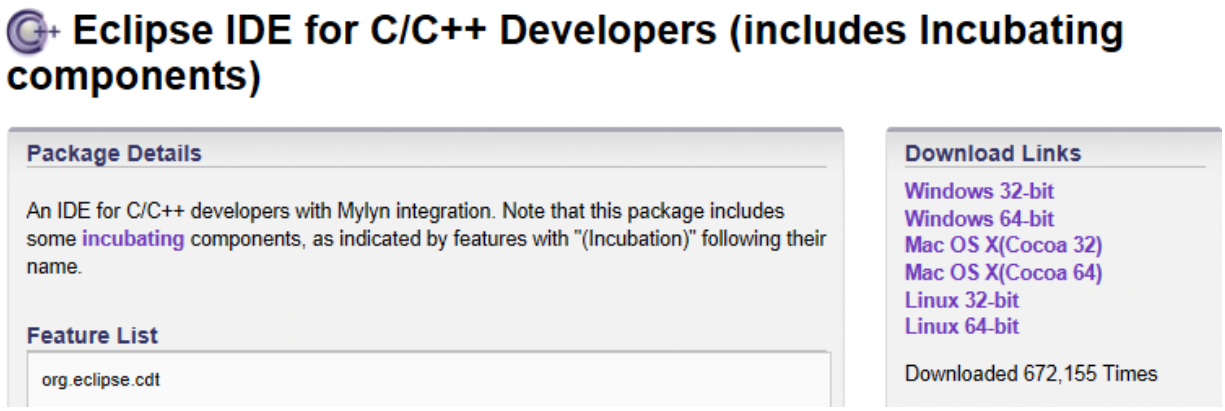
   To verify if your PC already has Jave installed you can open a DOS Prompt and type the following command:

   java –version

   If you don't see a version show up then you will need to install Java on your machine. Depending on whether your machine is a 32bit or 64bit PC you will install either the ***jre-7u9-windows-i586 x32.exe*** or the ***jre-7u9-windows-x64.exe*** respectively. These are both found on the Install Disk.

5) Install Eclipse Indigo IDE: Depending on whether you are using a 32bit or 64bit machine you will unpack either the eclipe-cpp-indigo-SR2-incubation-win32.zip or the eclipe-cpp-indigo-SR2-incubation-win32-x86_64.zip respectively. Unzip the ENTIRE Zip file onto C:\. You now have a full Eclipse intstallation at C:\eclipse.

   You can also download Indigo directly from the following link.
   http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers-includes-incubating-components/indigosr2



6) Install Plug-Ins: Now Start Eclipse by double-clicking the ***eclipse.exe*** found in C:\eclipse\eclipse.exe. You may want to right click on the executable and create a Shortcut or if using WIN7 choose *'Pin to Start Menu'* .

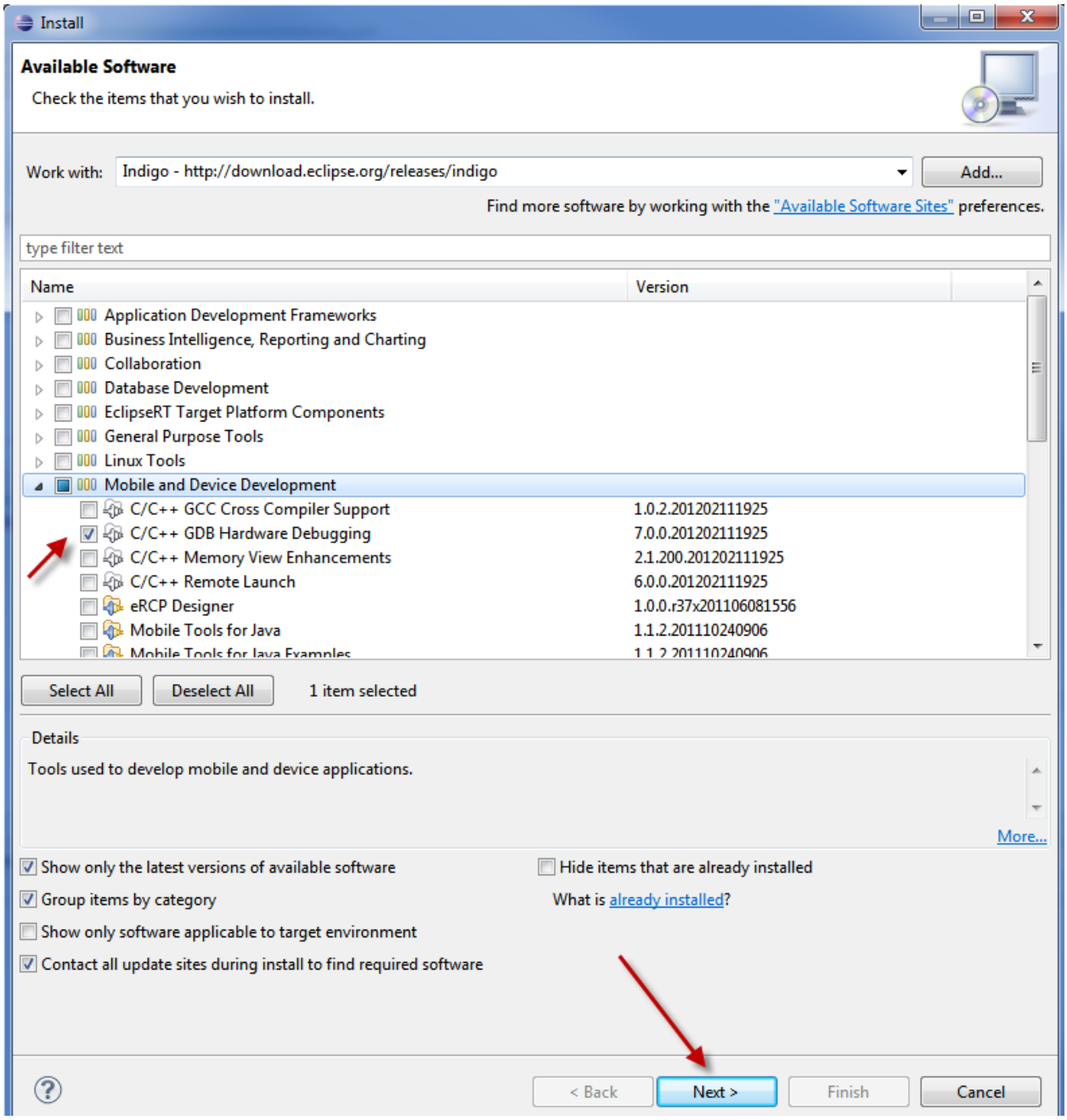
   1. When prompted for a workspace you will navigate to one of the following directories depending on which STK you have.
      GIANT GECKO/STK3700: C:\%APPDATA%\energymicro\kits\EFM32GG_STK3700\examples
      LEOPARD GECKO/STK3600: C:\%APPDATA%\energymicro\kits\EFM32LG_STK3600\examples
      TINY GECKO/STK3300: C:\%APPDATA%\energymicro\kits\EFM32TG_STK3300\examples

Other Kits such as the original Gecko series or possibly Development Kits will follow the same directory structure.

2. You will now see the Eclipse's welcome screen.
3. Select Install New Software… from the Help pulldown menu and enter

   http://download.eclipse.org/releases/indigo

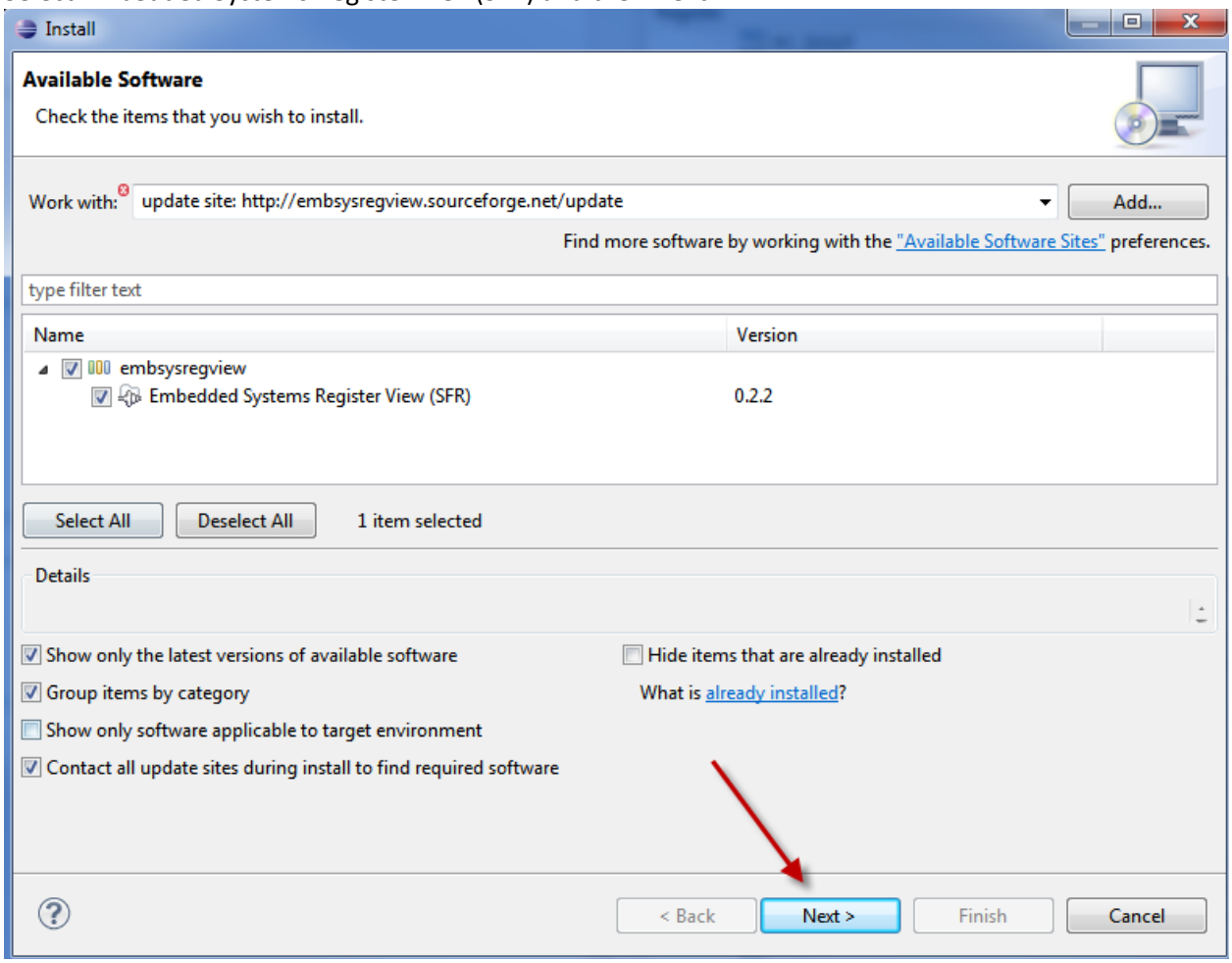
   in the Work with: field as shown below. You might need to wait a while for the plugins to show up. Browse to Mobile and Device Development and select C/C++ GDB Hardware Debugging. Press the Next > button and follow the instructions.



When asked to restart Eclipse, please do so.

4. Install Eclipse Embedded Systems Register View plugin: Proceed as was done in step 6.3 above but use http://embsysregview.sourceforge.net/update as the location.

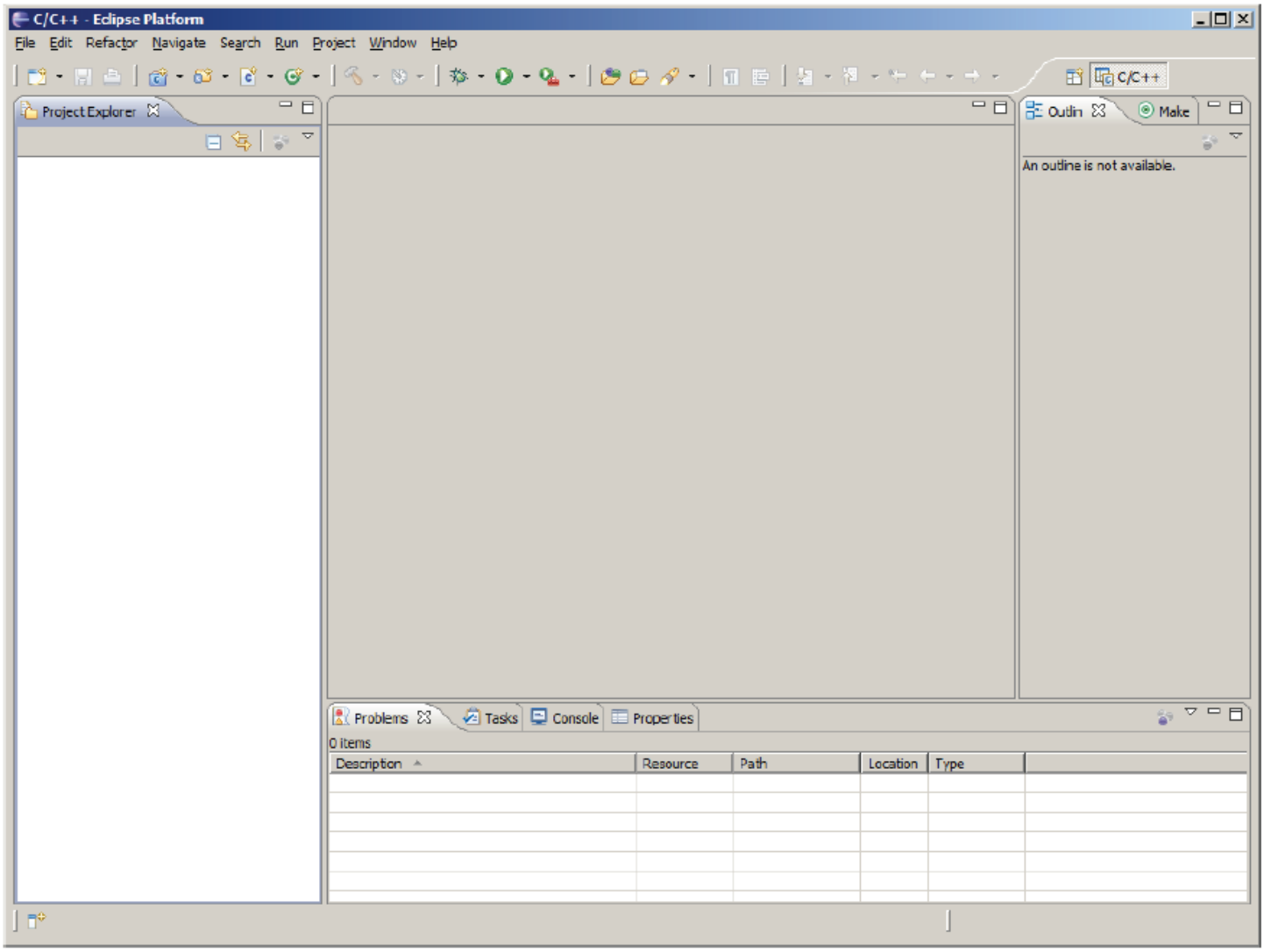
   Select Embedded Systems Register View(SFR) and then Next>.



   When asked to restart Eclipse, please do so.

7) Now we can continue on past the Welcome screen. Click the 3D Arrow to the right of the welcome screen to enter the Workbench view.

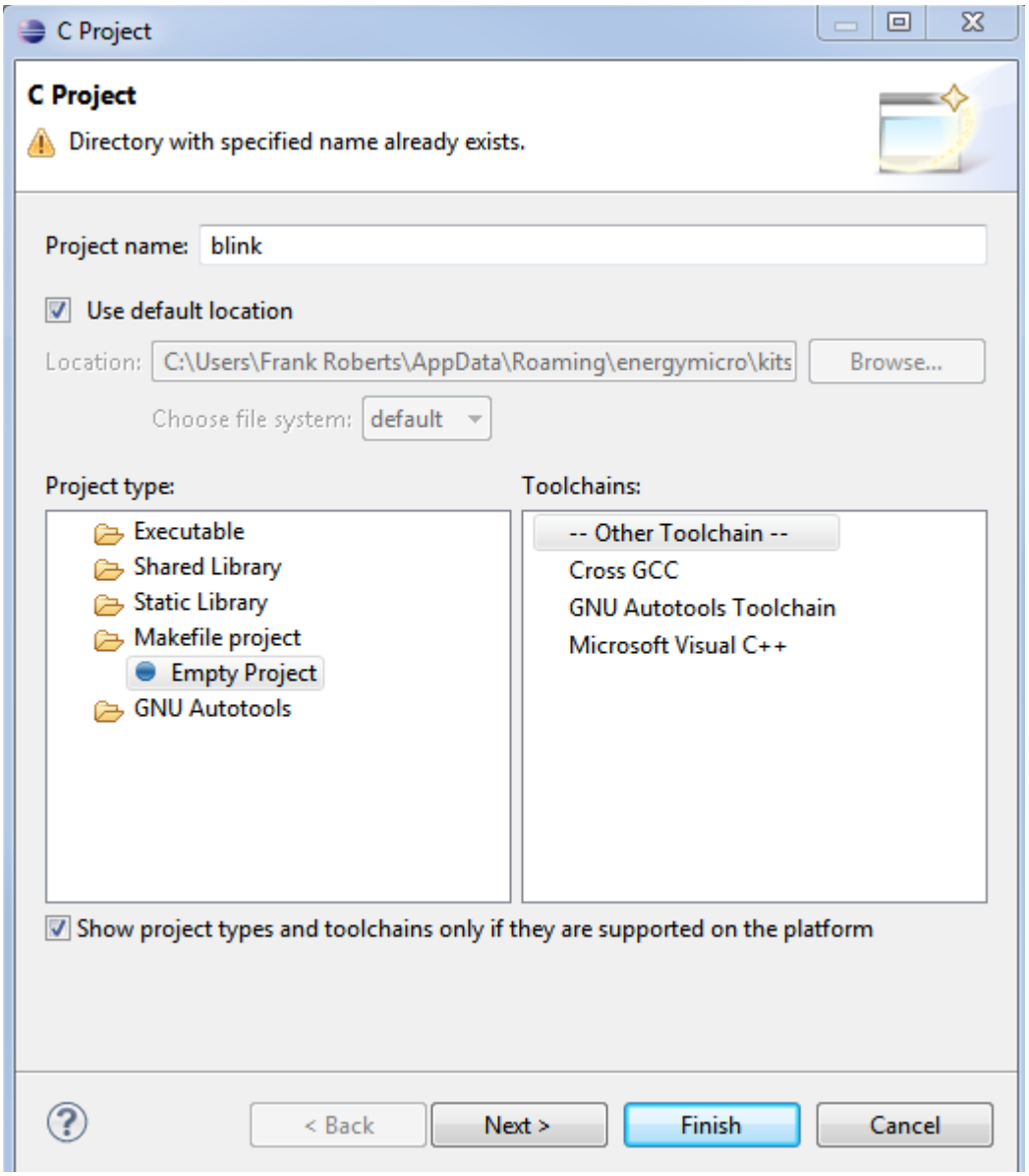
You will see the following now…



8) **Create a Project:** The new project will be based on the 'blink' example provided for the particular STK/DK that you have in your possession. Again you need to make sure you have chosen the Workspace directory to be that of the STK/DK that you have.

Create the project by selecting ***File->New->C Project***. Use blink as the project name.
For Project type: select Makefile project
For Toolchains: select - - Other Toolchain - -

You window should look like the following…



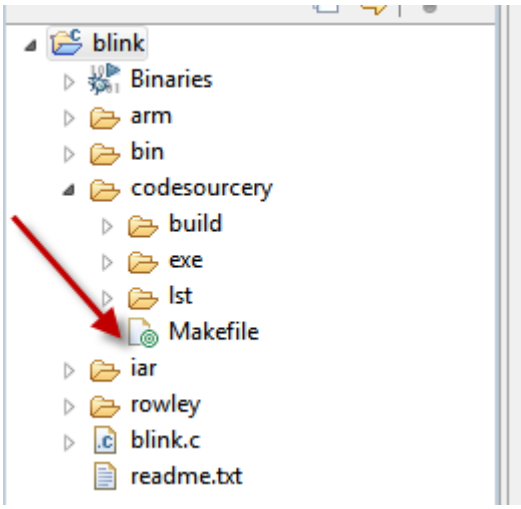
Click ***Finish***

9) **Add Project Properties:** ***Select Project->Properties*** from the Project dropdown menu.
In the Project Properties window make the following modifications which are also shown in the updated window below.

1. C/C++ Build: UNCHECK the checkbox for 'Use default build command' and type in cs-make.
2. C/C++ Build: Modify build directory to ${workspace_loc:/blink/codesourcery}.
3. C/C++ Build->Discovery Options: Uncheck thet Automatic discovery of paths and symbols checkbox.
4. C/C++Build->Settings: Check the GNU Elf Parser checkbox.

Click the ***OK*** button to save project properties.

10) Modifying the Makefile: Find the Makefile.blink file in the blink\codesourcery directory for your particular STK/DK. Copy and rename this file to Makefile without a file type suffix. We will want to inspect this Makefile and understand how it is setup and ensure it is setup correctly for our project. You will know that you have renamed the Makefile.blink to simply Makefile because in Eclipse you will now see a Green Bullseye next to the Makefile file as shown below.



Double click Makefile in Project Explorer pane to open inside of Eclipse.

Ensure Correct Path is set for your Codesourcery Compiler Install Directory!!

```
##################################################################
# Definitions of toolchain.                                      #
# You might need to do changes to match your system setup        #
##################################################################

# Change path to CodeSourcery tools according to your system configuration
WINDOWSCS ?= CodeSourcery\Sourcery_CodeBench_Lite_for_ARM_EABI  #This may change if you did not
install CodeSourcery to its Default location!!
```

Ensure correct EFM32 and Project name are selected!!

```
##################################################################
# Definitions                                                    #
##################################################################

DEVICE = EFM32GG990F1024
PROJECTNAME = blink
```

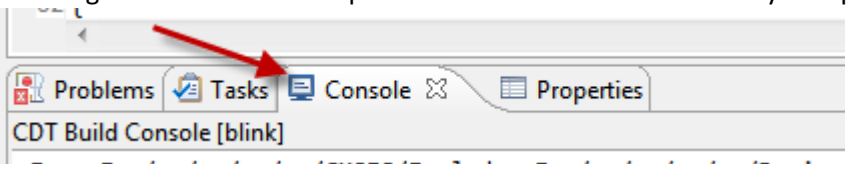
Check CFLAGS macro contains –O0. This selects the Compiler optimization. Use –O0 is best for debugging

```
debug:   CFLAGS += -DDEBUG -O0 -g3
```

Inspect the Makefile further and you will find the Include paths listed for the header files as shown below along with the C Source files that are being compiled in the project. Keep in mind the Makefile has to be used since the LITE version of Codesourcery only provides command line interface.

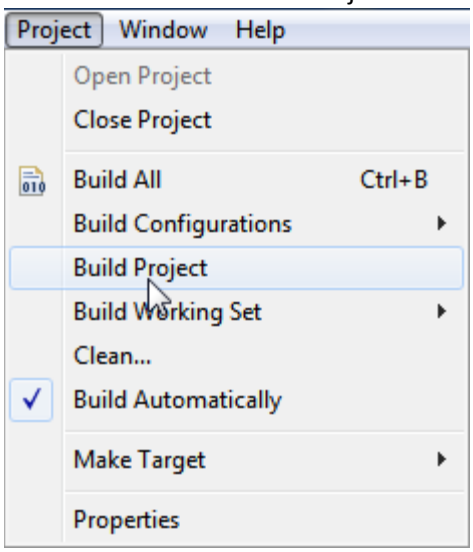
```
INCLUDEPATHS += \
-I.. \
-I../../../../../CMSIS/Include \
-I../../../../../Device/EnergyMicro/EFM32GG/Include \
-I../../../../../emlib/inc \
-I../../../../common/bsp \
-I../../../config

##################################################################
# Files                                                          #
##################################################################

C_SRC +=  \
../../../../../Device/EnergyMicro/EFM32GG/Source/system_efm32gg.c \
../../../../common/bsp/bsp_stk.c \
../../../../common/bsp/bsp_stk_leds.c \
../../../../common/bsp/bsp_trace.c \
../../../../../emlib/src/em_assert.c \
../../../../../emlib/src/em_cmu.c \
../../../../../emlib/src/em_ebi.c \
../../../../../emlib/src/em_emu.c \
../../../../../emlib/src/em_gpio.c \
../../../../../emlib/src/em_system.c \
../../../../../emlib/src/em_usart.c \
../blink.c

s_SRC +=  \
../../../../../Device/EnergyMicro/EFM32GG/Source/G++/startup_efm32gg.s
```

11) Compile the project: We are now ready to compile the project. Before we do so make sure we have the Console Tab showing in the bottom of Eclipse so we can see the Codesourcery Compiler output.

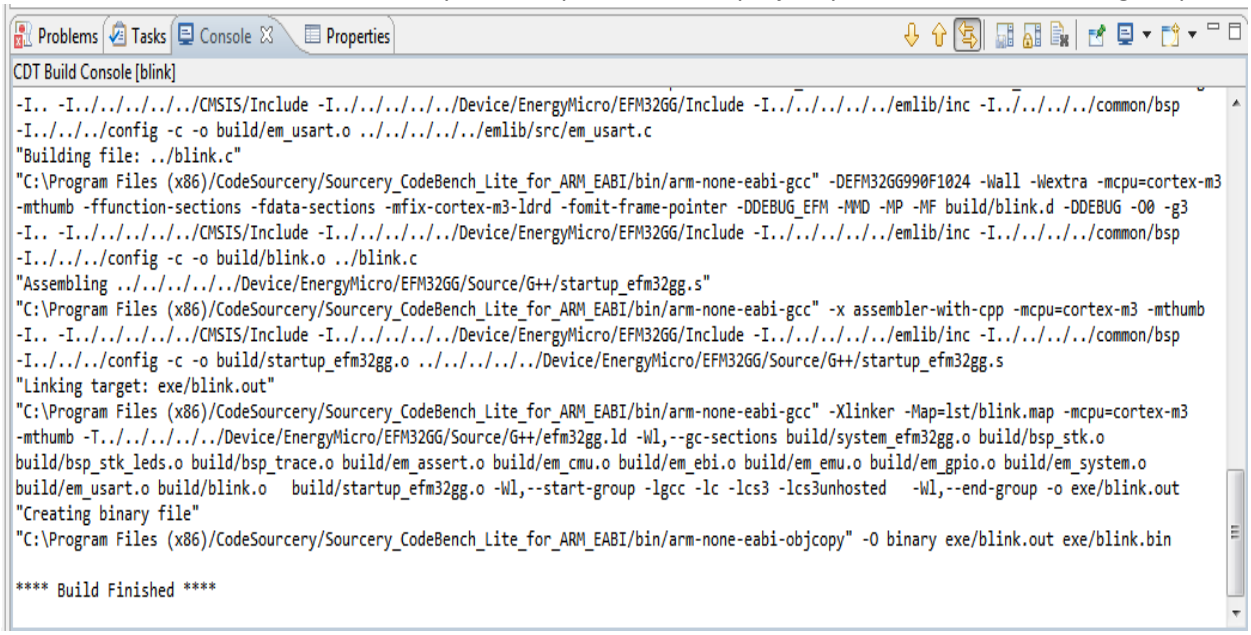


There are multiple ways to Build the project. Some are listed below…

1. From the main menu bar Project->Build Project



2. From the Project Explorer pane, right click on the top level 'blink' project and select Build Project.
3. Use Ctrl-B from the keyboard.
4. You will now see a Progress Window appear after you select Build Project and information scrolling down in the Console Window. After Codesourcery has compiled the blink project you will see the following output in Console.



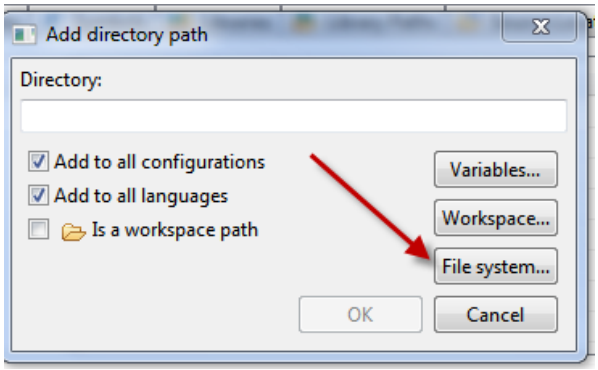
It has successfully built the project with no errors.

5. Remove Eclipse Discrepancies: Even though the project compiled you will notice that if you inspect blink.c you will see that it shows ? marks and what appear to be little red bugs next to indicating that things like uint32_t could not be resolved. This is because Eclipse is rather separated from Codesourcery and does not know where the various Header files are located. Codesourcery knows this because of the Makefile but these details are not inherited by Eclipse via Codesourcery LITE. So we need to tell Eclipse where these files are.

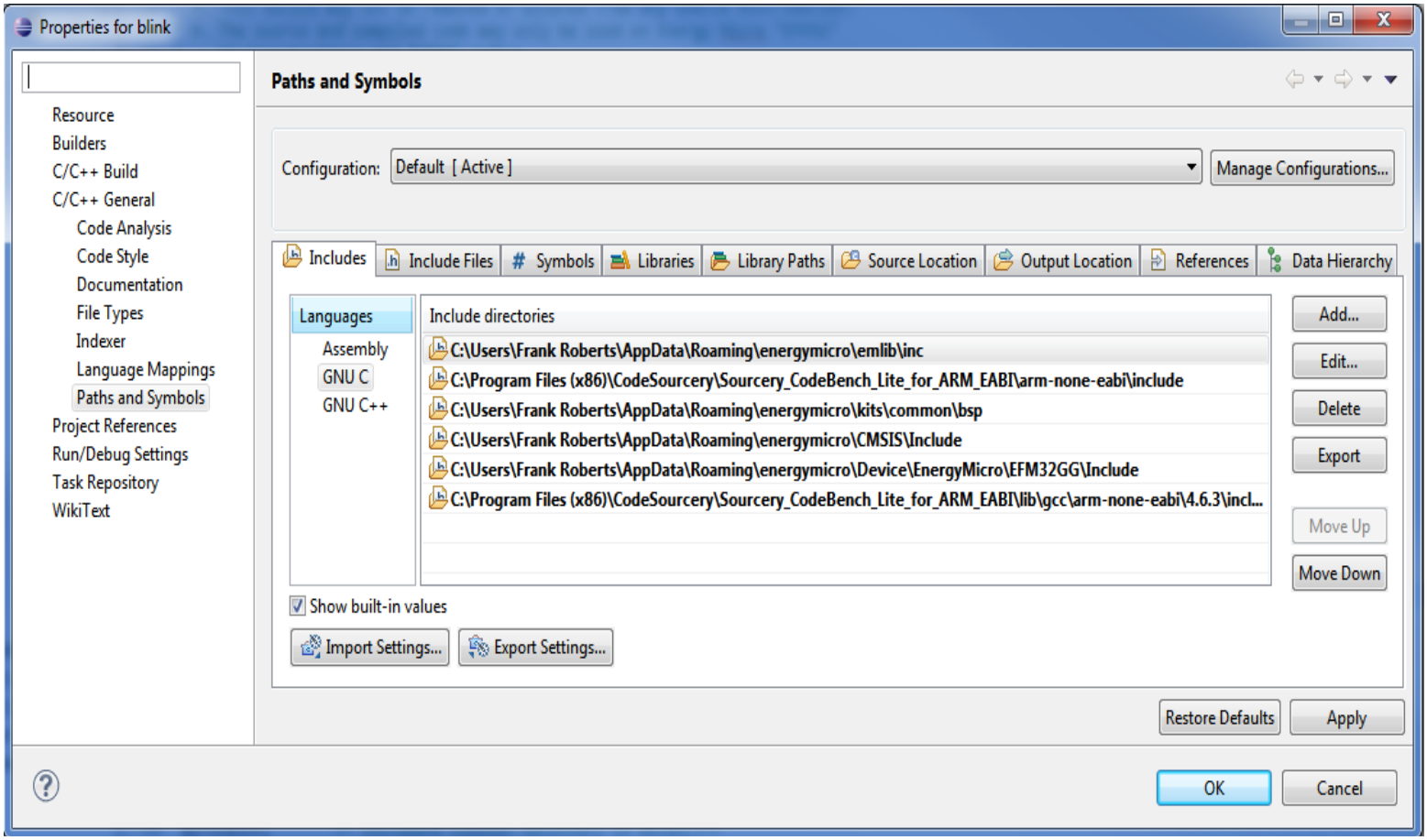
Eclipse doesn't know where the header files are and shows ? marks and red bugs to indicate this…

```
34 ***********************************************************
35 #include <stdint.h>
36 #include <stdbool.h>
37 #include "em_device.h"
38 #include "em_chip.h"
39 #include "em_cmu.h"
40 #include "em_emu.h"
41 #include "bsp.h"
42 #include "bsp_trace.h"
43
44 volatile uint32_t msTicks; /* counts 1ms timeTicks */
45
46 void Delay(uint32_t dlyTicks);
47
48 /***********************************************************
```

Goto ***Project->Properties*** and then into ***C/C++ General->Paths and Symbols***. Now click the Includes Tab and enter the following information below by clicking the Add… button and then File system… button. You should also check the Add to all configurations and Add to all languages boxes. Now populate the directories as shown below.



Enter Include Directory paths as shown below. Keep in mind the C:\Users\Frank Roberts\Appdata\Roaming will differ according to your username, EFM32 STK/DK you are using and whether or not you are on a XP or WIN7 system as described earlier in this document.



After you have entered all the paths click ***Apply*** or ***OK***. It will ask you if you want to rebuild, do so. You may need to still manually rebuild as we have before. If all of the paths have been entered correctly all of the ? marks and bugs will disappear. Eclipse can now reference these files and definitions. **Keep in mind this has NOTHING to do with the Compilation/Makefile that Codesourcery makes use of.**

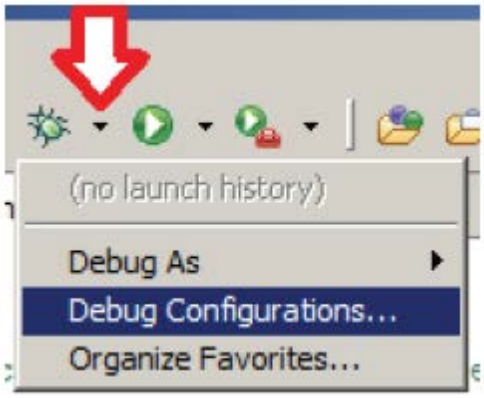
Blink.c after entering directories for Paths and Symbols…

```
 *
 ***********************************************************
#include <stdint.h>
#include <stdbool.h>
#include "em_device.h"
#include "em_chip.h"
#include "em_cmu.h"
#include "em_emu.h"
#include "bsp.h"
#include "bsp_trace.h"

volatile uint32_t msTicks; /* counts 1ms timeTicks */

void Delay(uint32_t dlyTicks);

/***********************************************************
```

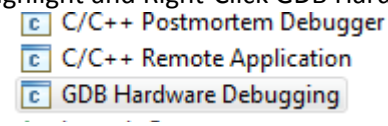
12) Download and Debug Application Code

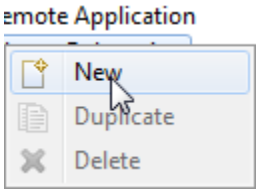
To download and debug we need to create a Debug Application.

1. Create a debug launch configuration: Go to top menu bar and select Debug Configurations…

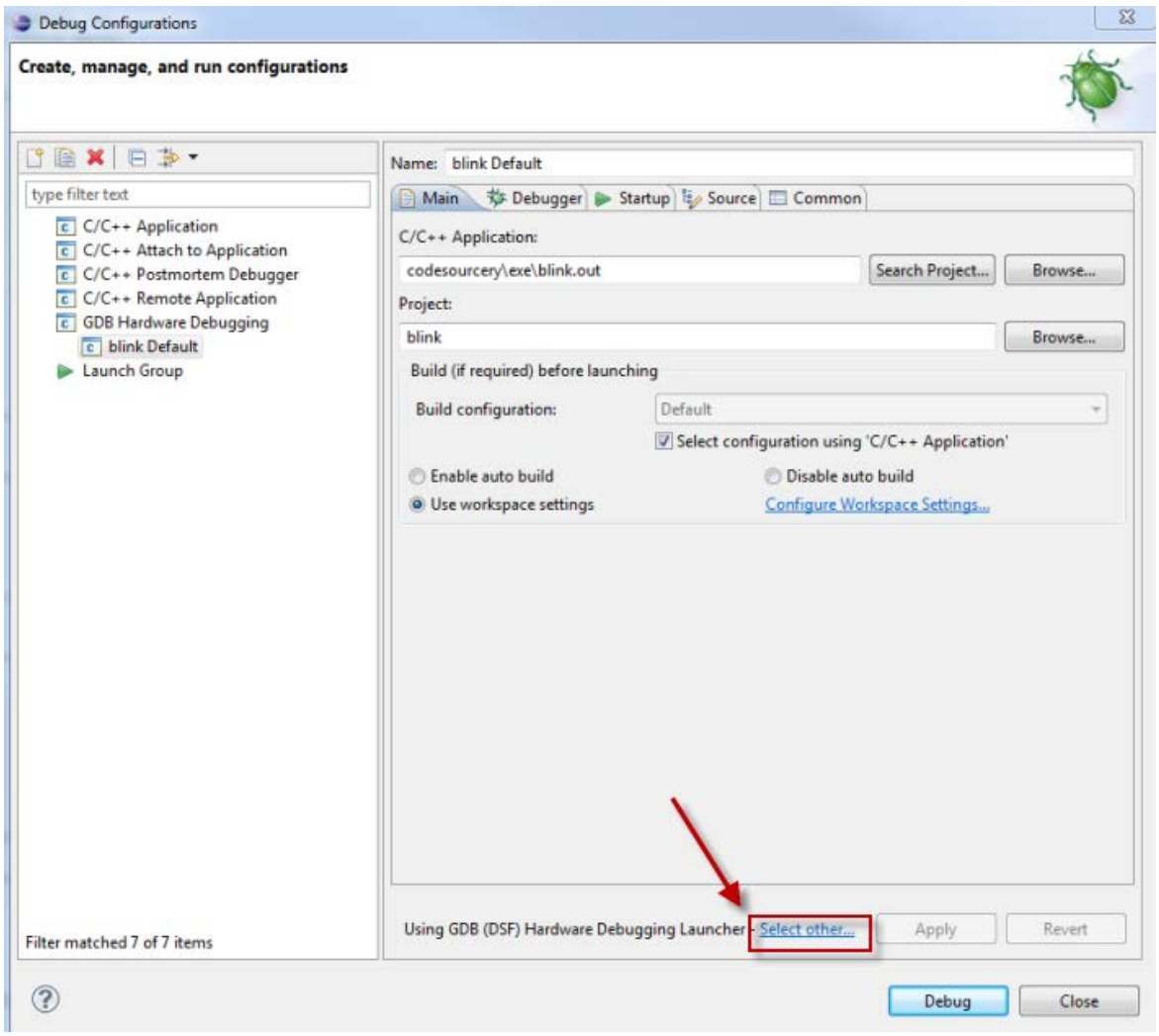


2. Highlight and Right-Click GDB Hardware Debugging and Select New

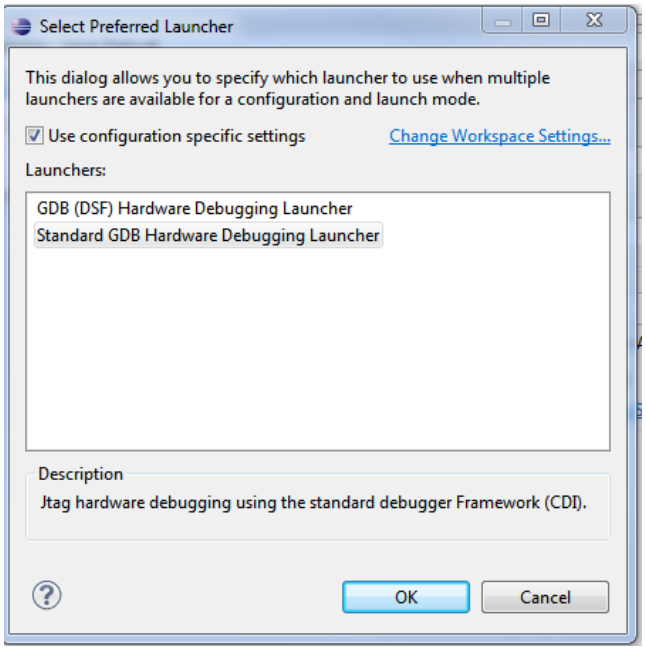




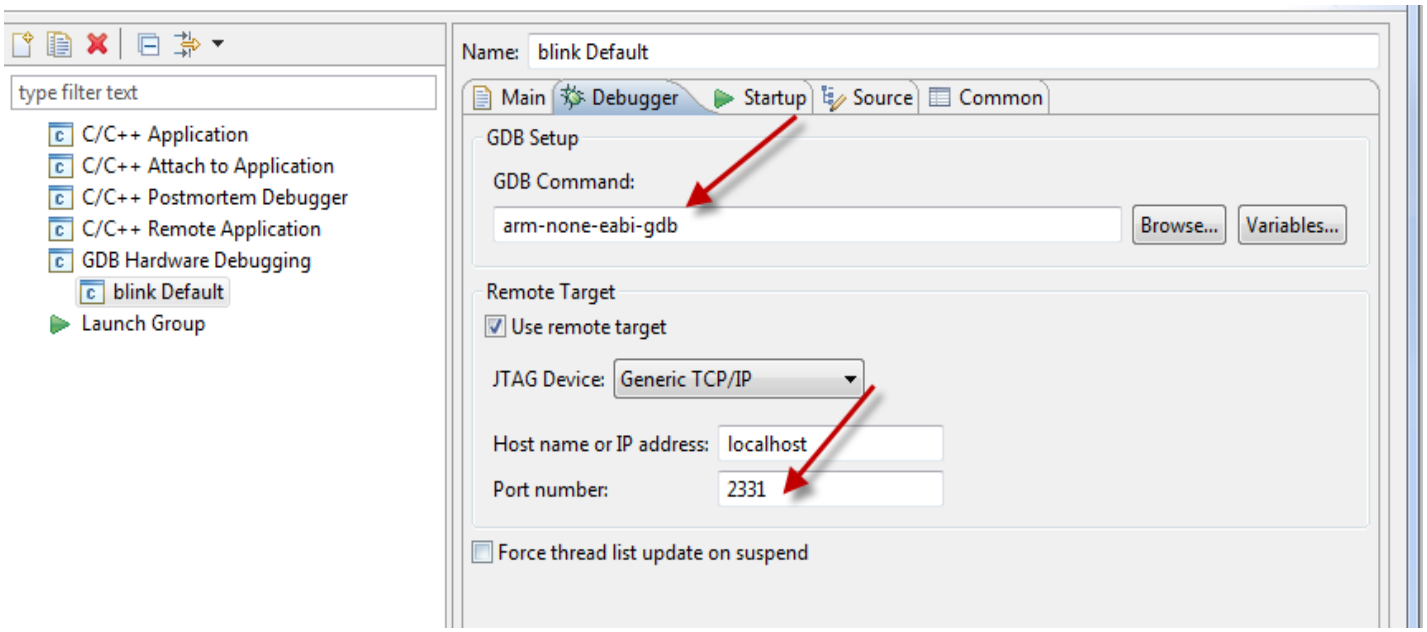
3. You should now see this Main Menu window
   Click on Select Other… as highlighted below



Now Check the ***Use configuration specific settings*** as shown and choose ***Standard GDB Hardware Debugging Launcher*** and click OK.



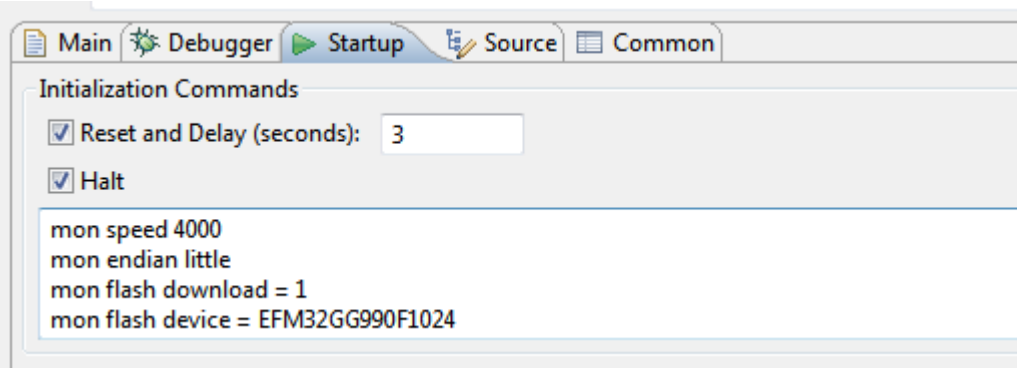
Now with the **Debugger** Tab selected change the GDB Command: and Port Number as shown below. Check other settings to ensure they are the same.



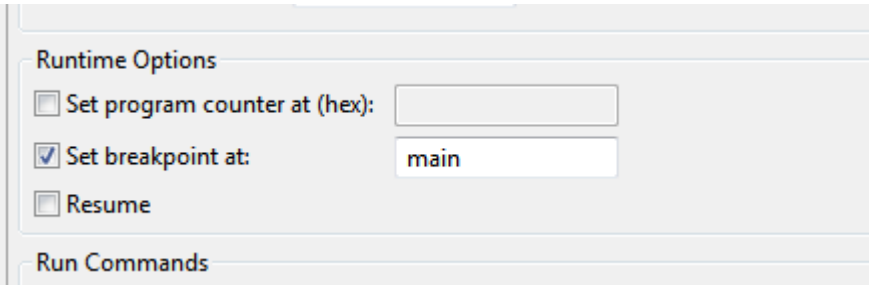
Now with the **Startup** Tab Selected

1. Enter the following in the Initialization Commands section as shown below. Keep in mind the device may differ.

   mon speed 4000
   mon endian little
   mon flash download = 1
   mon flash device = EFM32GG990F1024
   mon reset 1

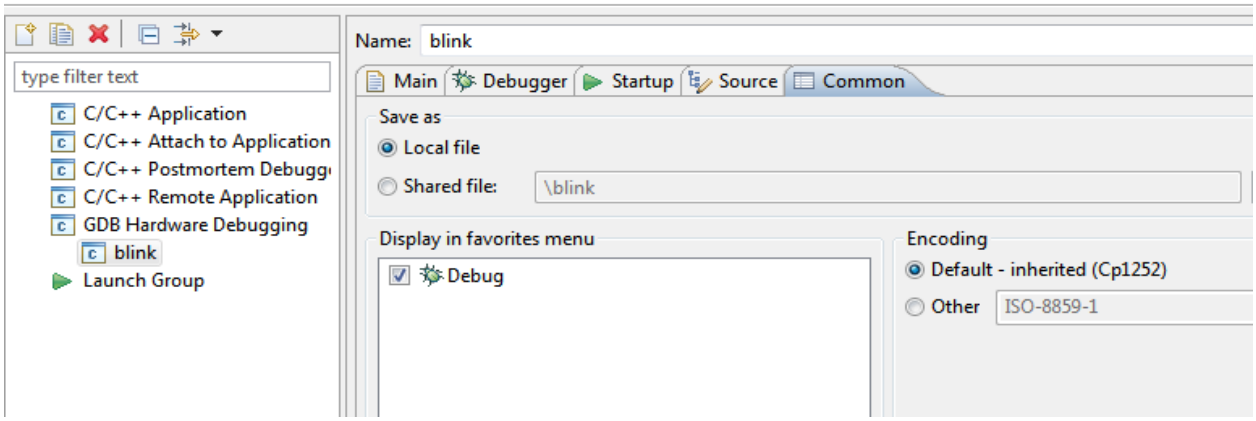


2. Check the Set breakpoint at: box and type main in the window as shown below



Now with the **Common** tab selected

1. Check the Debug box as shown below

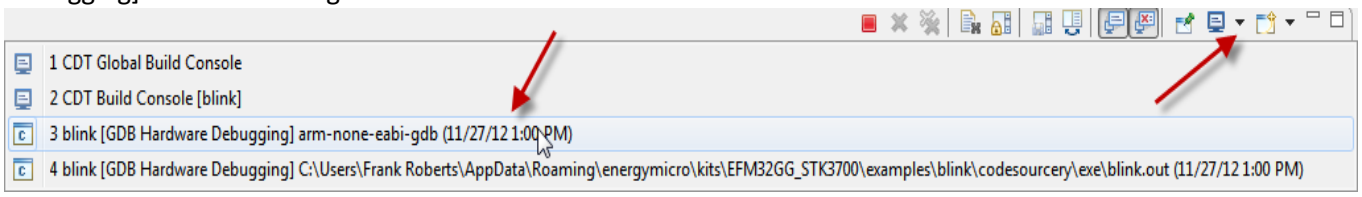


Finally click the Apply button and Close button.

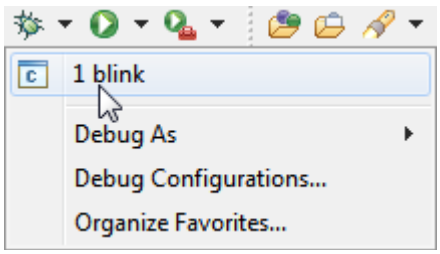
Now we are ready to download the compiled project to the STK and Debug!!

KEEP IN MIND YOU NEED TO HAVE THE SEGGER J-LINK GDB SERVER RUNNING TO FLASH AND DEBUG!!

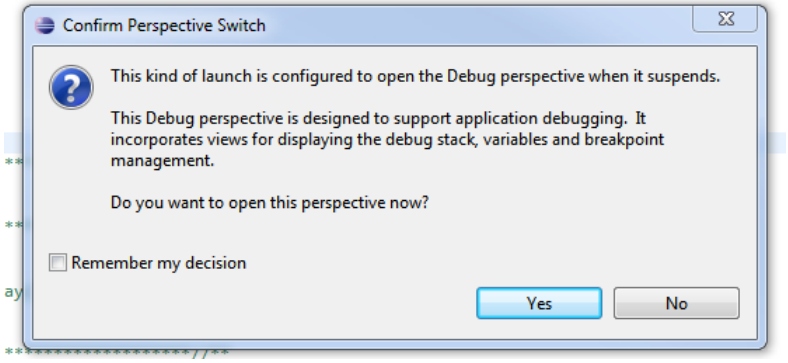
Before we do so let's change the Console Window to output the GDB Debugger information. To do so simply find the Console Window Icon on the bottom right hand side of the Eclipse output pane and select [GDB Hardware Debugging]arm-none-eabi-gdb as shown below.



To Download/Debug click on the Drop-Down menu next to the Green Beetle in the middle of the toolbar and select Blink as shown below.



The first time you do this you will see the window below, click YES to view the Debug Perspective. You also may want to check the box 'Remember my decision' so you don't see this window every time you debug.



You should now see the following in the Console Window…
Notice that it shows it set a Temporary Breakpoint at line 75 in blink.c which is the first instruction in Main() which is exactly what we told the debugger to do in the Debug Configurations.

```
Console  Tasks  Problems  Executables  Memory
blink [GDB Hardware Debugging] arm-none-eabi-gdb (11/27/12 1:00 PM)
mon speed 4000
mon endian little
mon flash download = 1
mon flash device = EFM32GG990F1024
load "C:\\Users\\Frank Roberts\\AppData\\Roaming\\energymicro\\kits\\EFM32GG_STK3700\\examples\\blink\\codesourcery\\exe\\blink.out"
Loading section .text, size 0x1540 lma 0x0
Loading section .eh_frame, size 0x4 lma 0x1540
Loading section .rodata, size 0xdc lma 0x1544
Loading section .data, size 0x10 lma 0x1620
Start address 0xdc, load size 5680
Transfer rate: 2773 KB/sec, 1420 bytes/write.
tbreak main
Temporary breakpoint 1 at 0x1466: file ../blink.c, line 75.
```

You will also see that the Segger J-Link GDB Server GUI has changed to reflect that it has downloaded the project and is connected to the EFM32. Keep in mind specific EFM32 may differ depending on what STK/DK you have.

```
SEGGER J-Link GDB Server V4.54a
File  Help
GDB      Connected to 127.0.0.1      Initial SWD speed  5 kHz
J-Link   Connected                   Current SWD speed  4000 kHz
CPU      EFM32GG990F1024, Halted     3.30 V             Little endian
[x] Localhost only
[x] Stay on top
[x] Show log window
[ ] Generate logfile
[ ] Cache reads
[ ] Verify download
[x] Init regs on start
Log output:  Clear log
Downloading 4 bytes @ address 0x00001540
Downloading 220 bytes @ address 0x00001544
Downloading 16 bytes @ address 0x00001620
Writing register (PC = 0x000000dc)
Read 4 bytes @ address 0x000000DC (Data = 0x47804801)
Read 2 bytes @ address 0x00001466 (Data = 0xF7FF)
GDB closed TCP/IP connection
Closing remote socket
Waiting for GDB connection...Connected to 127.0.0.1
Reading all registers
Read 4 bytes @ address 0x000000DC (Data = 0x47804801)
JTAG speed set to 4000 kHz
Target endianess set to "little endian"
Flash download enabled
Selecting device: EFM32GG990F1024
Resets the core only, not peripherals.
Downloading 5440 bytes @ address 0x00000000
Downloading 4 bytes @ address 0x00001540
Downloading 220 bytes @ address 0x00001544
Downloading 16 bytes @ address 0x00001620
Read 4 bytes @ address 0x000000DC (Data = 0x47804801)
Read 2 bytes @ address 0x00001466 (Data = 0xF7FF)
11 KB downloaded   1 JTAG device
```

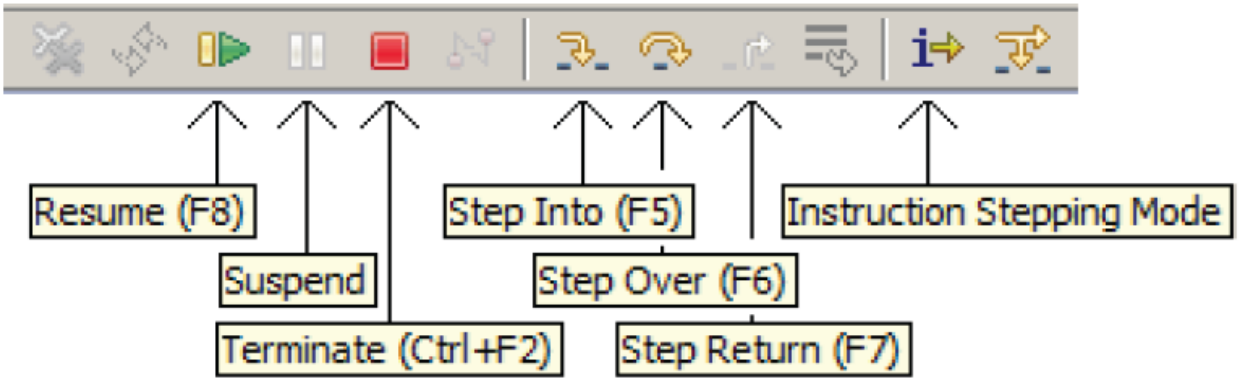
You have successfully installed Eclipse, Codesourcery, Segger J-LINK GDB Server, and can now compile download and Debug the application. This can serve as a starting point for your own custom projects.

Read Further for more Eclipse Tips…

Debugging Functionality:
To set a breakpoint simply double click on a source code line in the Eclipse editor. Other functionality is shown below.

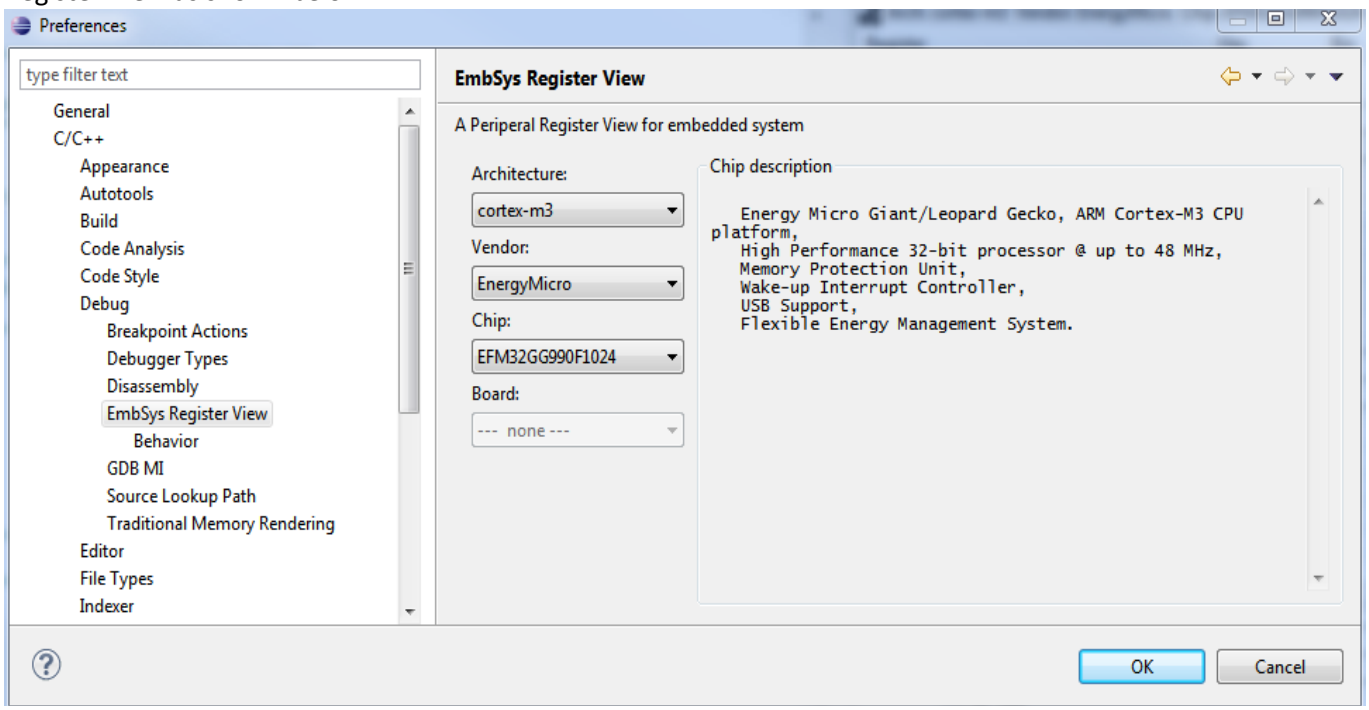
Look for the buttons shown in the figure below in the debug tab.

Figure 2.7. Debug button

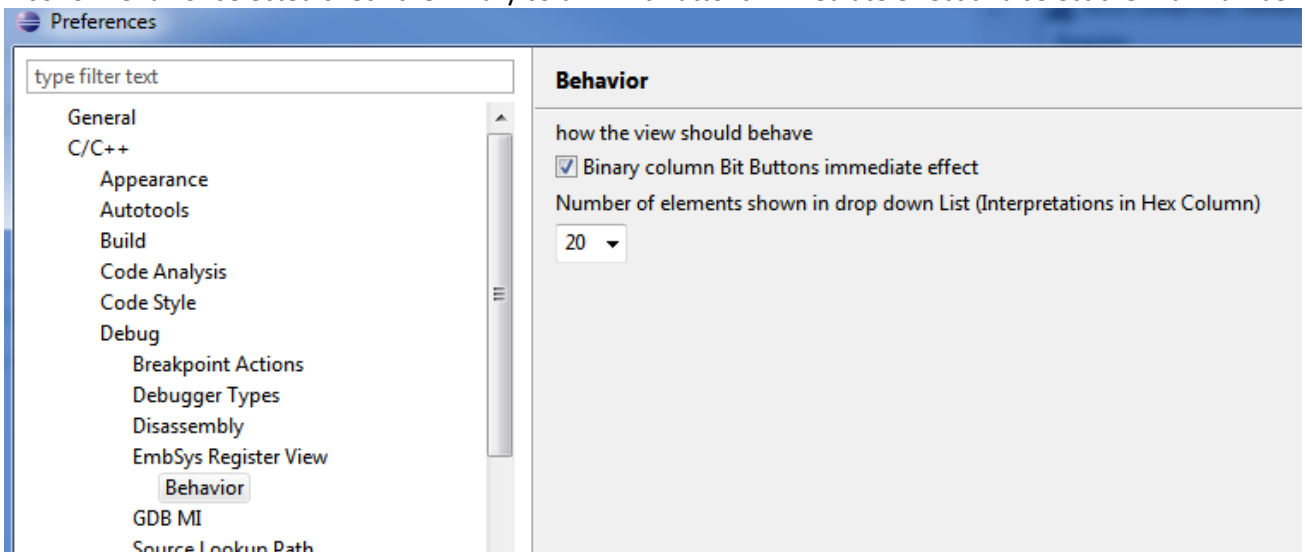


Using EmbSys Register View: The EmbSys register/SFR viewer is very useful when working with the EFM32 peripherals as it allows you to see inside the specific peripheral registers. The register viewer also contains documentation on peripheral registers and their bitfields(as tooltips).

Open Preferences in Eclipse's Window pulldown menu and select the correct device by going to C/C++->Debug->EmbSys Register View as shown below.



Also for Behavior selected check the Binary column Bit Buttons immediate effect and select the Max number of elements.



To open a register viewer in the Debug view, select Show View->Other… and then under the Debug node select EmbSys Registers.

To use the EmbSys view simply double click on a Register to view its contents as shown below.

PD_DOUTCLR	- write only -	---------- write only ----------	0x00000000	WO	0x40006080	Port Data Out Clear Register
PD_DOUTTGL	- write only -	---------- write only ----------	0x00000000	WO	0x40006084	Port Data Out Toggle Register
PD_DIN			0x00000000	RO	0x40006088	Port Data In Register
PD_PINLOCKN			0x0000FFFF	RW	0x4000608C	Port Unlocked Pins Register
PE_CTRL			0x00000000	RW	0x40006090	Port Control Register
PE_MODEL			0x00000000	RW	0x40006094	Port Pin Mode Low Register
PE_MODEH			0x00000000	RW	0x40006098	Port Pin Mode High Register
PE_DOUT	0x0000000C	00000000000000000000000000001100	0x00000000	RW	0x4000609C	Port Data Out Register
DOUT (bits 0-15)	0x000C	0000000000001100				Data Out
PE_DOUTSET	- write only -	---------- write only ----------	0x00000000	WO	0x400060A0	Port Data Out Set Register
PE_DOUTCLR	- write only -	---------- write only ----------	0x00000000	WO	0x400060A4	Port Data Out Clear Register
PE_DOUTTGL	- write only -	---------- write only ----------	0x00000000	WO	0x400060A8	Port Data Out Toggle Register
PE_DIN			0x00000000	RO	0x400060AC	Port Data In Register
PE_PINLOCKN			0x0000FFFF	RW	0x400060B0	Port Unlocked Pins Register
PF_CTRL			0x00000000	RW	0x400060B4	Port Control Register
PF_MODEL			0x00000000	RW	0x400060B8	Port Pin Mode Low Register

Summary: There are numerous other useful features of Eclipse that can be explored