

TD1208 AN0003 ECLIPSE GCC FLASH AND DEBUG CONFIGURATIONS



Disclaimer: The information in this document is provided in connection with Telecom Design products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Telecom Design products.

TELECOM DESIGN ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL TELECOM DESIGN BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF TELECOM DESIGN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Telecom Design makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Telecom Design does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Telecom Design products are not suitable for, and shall not be used in, automotive applications. Telecom Design products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2013 Telecom Design S.A. All rights reserved. Telecom Design®, logo and combinations thereof, are registered trademarks of Telecom Design S.A. SIGFOX™ is a trademark of SigFox S.A.. Other terms and product names may be trademarks of others.

1 Overview

This document provides instructions for creating Eclipse/GCC Flash and Debug configurations for the Telecom Design TD1208 module.

1.1 Scope

The Eclipse IDE used to develop TD1208 module firmwares does not provide built-in function to Flash or debug a firmware.

This guide focuses on the description of the creation of Flash and Debug configuration for the Eclipse/GCC toolchain, providing step-by-step instructions.

1.2 Organization

Each section in this document covers a separate topic, organized as follow:

- Section 1 is this overview
- Section 2 contains the instructions allowing to debug a TD1208 firmware directly into Eclipse
- Section 3 contains the instructions to flash a TD1208 device directly from Eclipse

1.3 Relevant Documents

Additional information on the TD1208 module and on its dedicated evaluation board can be found in:

- *TD1208 Datasheet*
- *TD1202 EVB User's Guide*
- *TD1208 Reference Manual*
- *TD1208 AN0002 Eclipse GCC Toolchain*

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1 Overview.....	3
1.1 Scope	3
1.2 Organization	3
1.3 Relevant Documents	3
2 Debug a TD1208 firmware into Eclipse.....	5
2.1 GNU C/C++ GDB Hardware Debugging Eclipse Plugin	5
2.2 Debug Configuration Creation	9
2.3 GDB Server Launch Configuration Creation	14
2.4 TD1208 Debug Session	16
3 Flash a TD1208 Firmware from Eclipse	19

2 Debug a TD1208 firmware into Eclipse

It is possible to debug a TD1208 in source mode directly into the Eclipse IDE, thus allowing a seamless flow of development and minimizing the development time.

In order to be able to debug a TD1208 device, a dedicated J-Link SWD/JTAG adapter or an Energy Micro EFM32 Tiny Gecko EFM32TG-STK3300 Starter Kit containing an embedded J-Link SWD adapter with the corresponding USB driver is required. This physical device will access the actual TD1208EVb or customer board that you want to debug.

The communication with this J-Link SWD Adapter is handled by a separate “J-Link GDB Server” program running on the PC that should be installed along with the corresponding J-Link USB driver. This software provides a local-only or network-wide access to the J-Link SWD adapter, listening on its TCP/IP port 2331.

The last part is the “GNU C/C++ GDB Hardware Debugging” plugin that must be installed into Eclipse to provide all the standard actions used during debugging, such as breakpoints, step-by-step execution, variable and memory examination and modification:

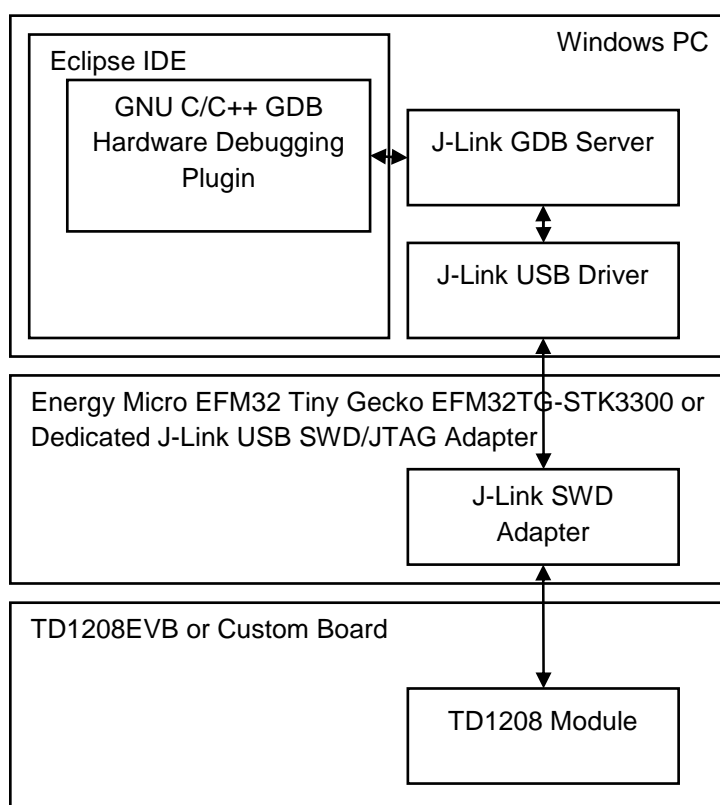


Figure 1 - TD1208 Debug Flow

2.1 GNU C/C++ GDB Hardware Debugging Eclipse Plugin

The installation of the required Eclipse C/C++ GDB Hardware Debugging Eclipse plugin is better performed from within Eclipse itself.

Navigate to the Eclipse installation directory and launch the “**eclipse.exe**” application. This will bring up a dialog window asking for a “workspace” location:

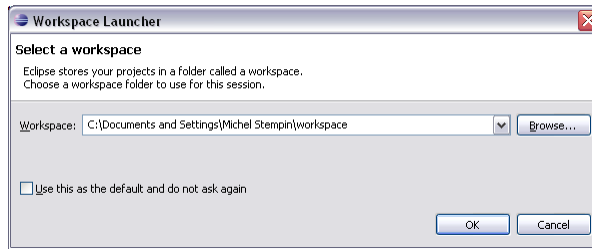


Figure 2- Eclipse Workspace Launcher

The Eclipse “workspace” is the location where Eclipse will store all the files related to your development projects. If you intend to have a single workspace for all your work, you may check the appropriate box to use it as a default and not be asked for it at every launch. Click “**OK**”, the main Eclipse window will appear:

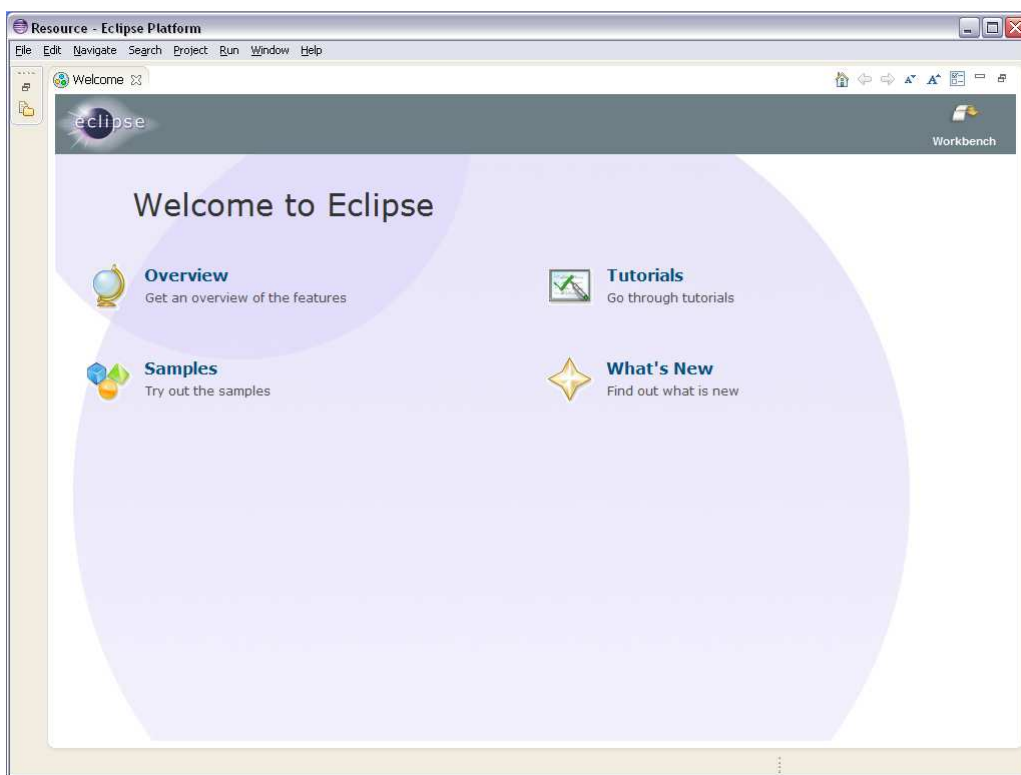


Figure 3- Eclipse Welcome Screen

To install the C/C++ GDB Hardware Debugging plugin for Eclipse, choose “**Help>Install New Software**” from the menus:

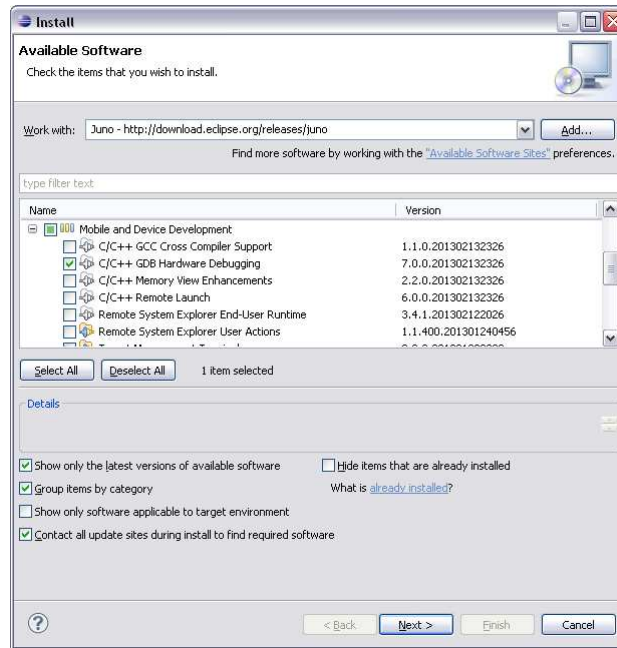


Figure 4- Eclipse Install New Software Dialog

Select the “Juno - <http://download.eclipse.org/releases/juno>” as the update site to work with, then choose the “**Mobile and Device Development > C/C++ GDB Hardware Debugging**” entry into the selection tree, then click on the “**Next**” button.

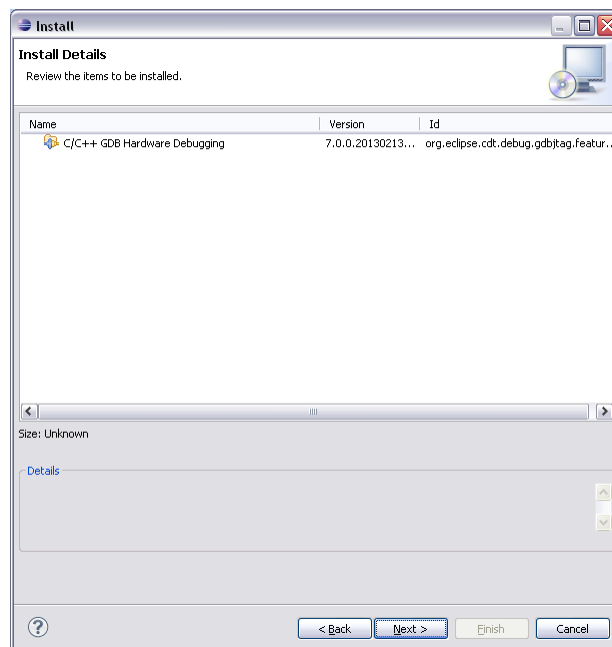


Figure 5- Eclipse Software Install Details

Check the box in front of the “**C/C++ GDB Hardware Debugging**” item in the list, then click “**Next**”.



Figure 6- Eclipse Software License Review

Check the **"I accept the terms of the license agreements"** radio button and click **"Finish"**, the installation will begin.

Eventually, Eclipse will need to be restarted for the changes to take effect:

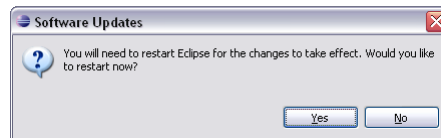


Figure 7- Restart Eclipse

Please accept by pressing **"Yes"**.

Once restarted, please navigate to the **"Window > Preferences"** menu entry:

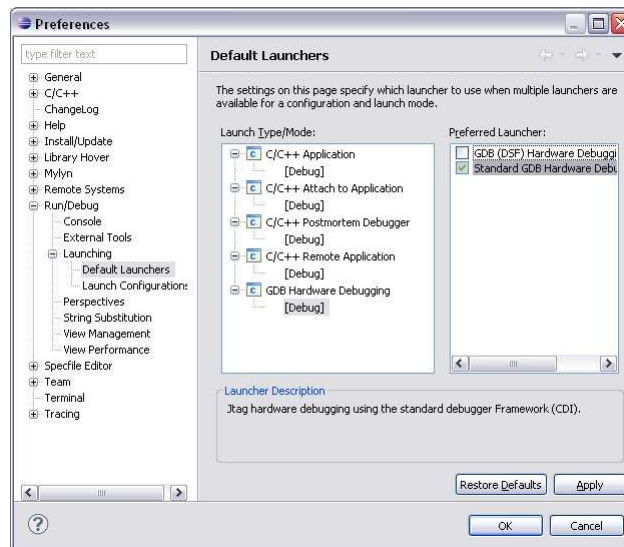



Figure 8 - Default Launchers

Navigate to the **“Run/Debug > Launching > Default Launchers”** entry in the preference tree, then select the **“GDB Hardware Debugging > [Debug]”** entry into the **“Launch Type/Mode”** tree end and choose **“Standard GDB Hardware Debugging Launcher”** as the **“Preferred Launcher”**, then click on **“OK”**.

2.2 Debug Configuration Creation

First select an executable project into the **“Project Explorer”** panel and right-click on it with the mouse. In the contextual menu, please select the **“Build Configurations > Set Active > GCC Debug Gecko”** build configuration.

In the main menu bar, click on the downwards arrow right to the **“bug”** button , then select **“Debug Configurations...”** into the menu:

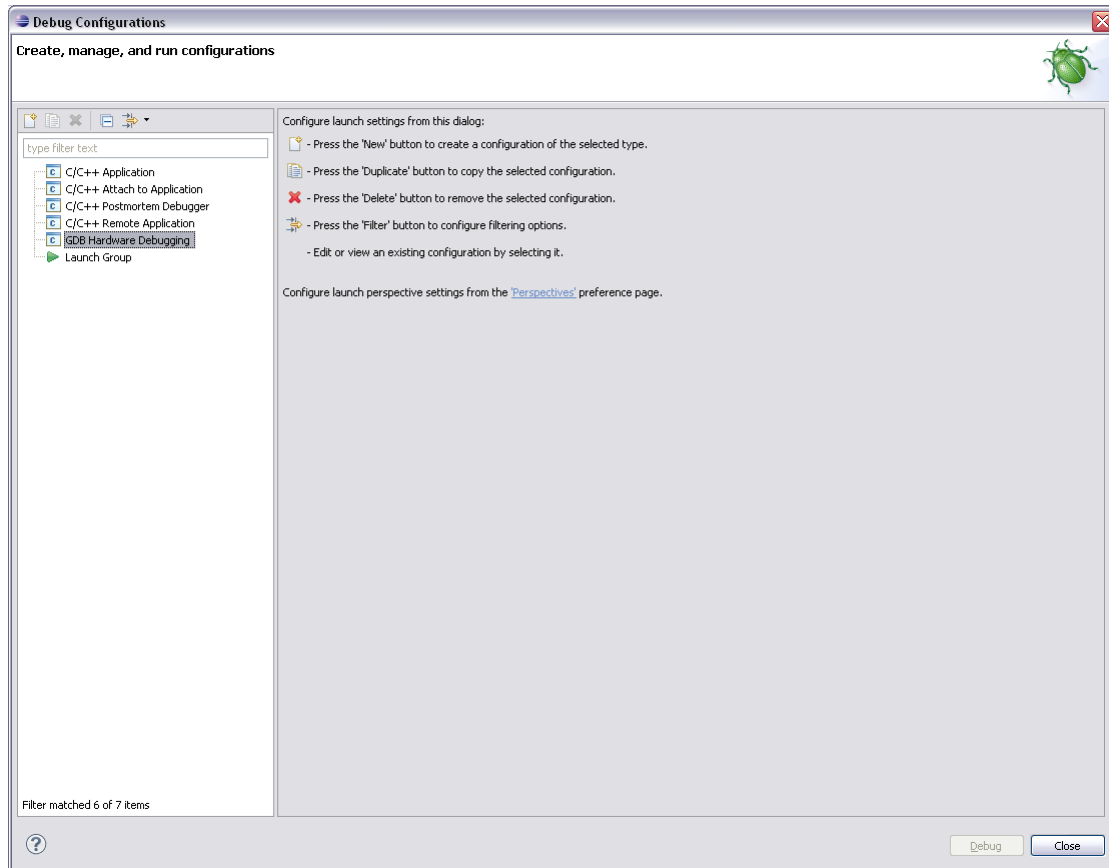


Figure 9 - Debug Configurations

Right-click on the “GDB Hardware Debugging” into the tree and select the “**New**” entry: a new debug configuration holding the concatenated project and active configuration names will be created, with the “**Project:**” field pre-filled with the project name.

In the same “**Main**” tab, please enter into the “**C/C++ Application:**” field:

```
${config_name:${project_name}}/${project_name}.elf
```

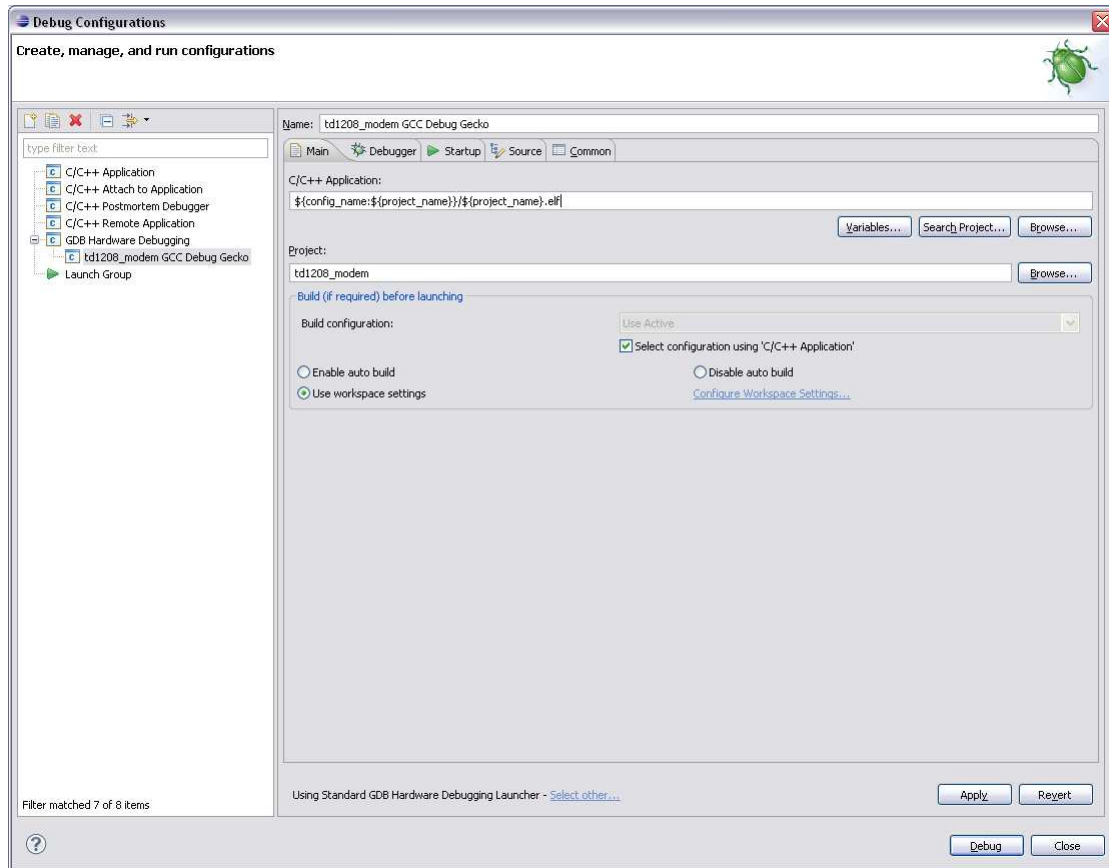


Figure 10 - Main Debug Configuration

Click on the “**Debugger**” tab, and replace the “**gdb**” value in the “**GDB Command:**” field by “**arm-none-eabi-gdb**”, and the default “**10000**” value in the “**Port number:**” field by “**2331**”:

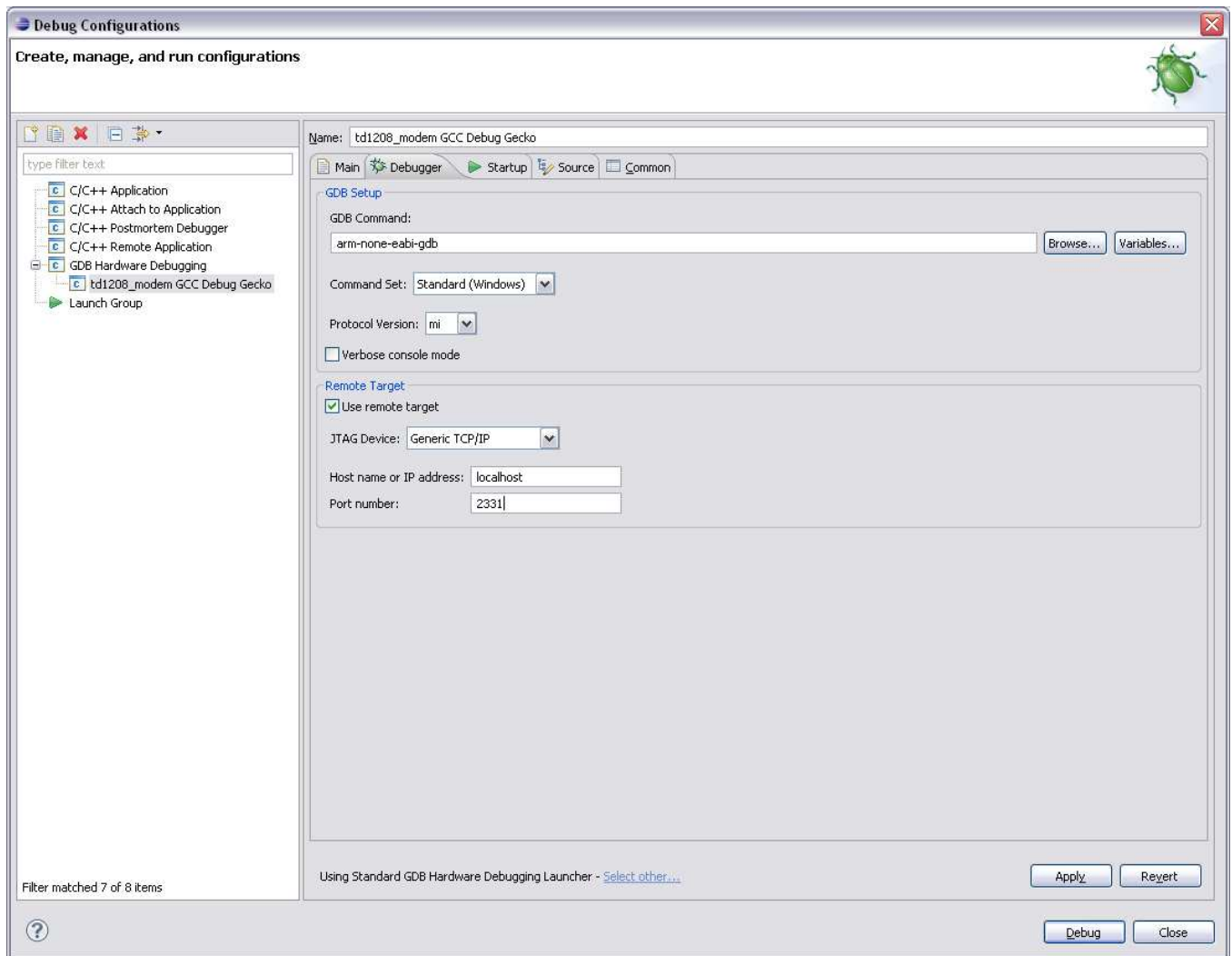


Figure 11 - Debugger Configuration

Click on the **"Startup"** tab and fill in the following text into the **"Initialization Commands"** text area:

```
set tdesc filename target-m3.xml Initialization Commands
mon speed 4000
mon endian little
mon flash download = 1
mon flash device = EFM32G210F128
mon reset 1
```

Check the **"Set breakpoint at:"** checkbox and enter **"TD_USER_Setup"** into the following text area.

Check also the **"Resume"** checkbox:

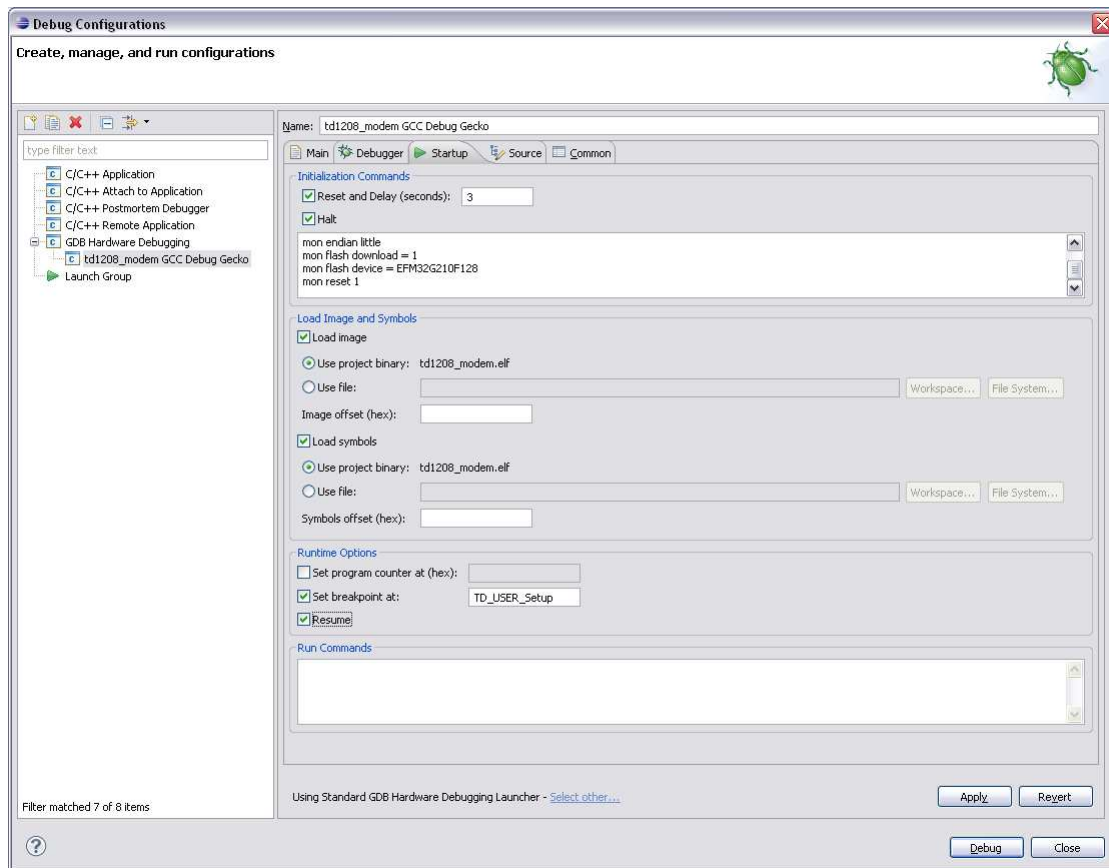


Figure 12 - Startup Configuration

Click on the **“Common”** tab and check the **“Debug”** entry in the favorite’s menu tree, then click on the **“Apply”** and **“Close”** buttons:

Note: a debug configuration can be cloned by right-clicking on it and selecting **“Duplicate”** in the contextual menu.

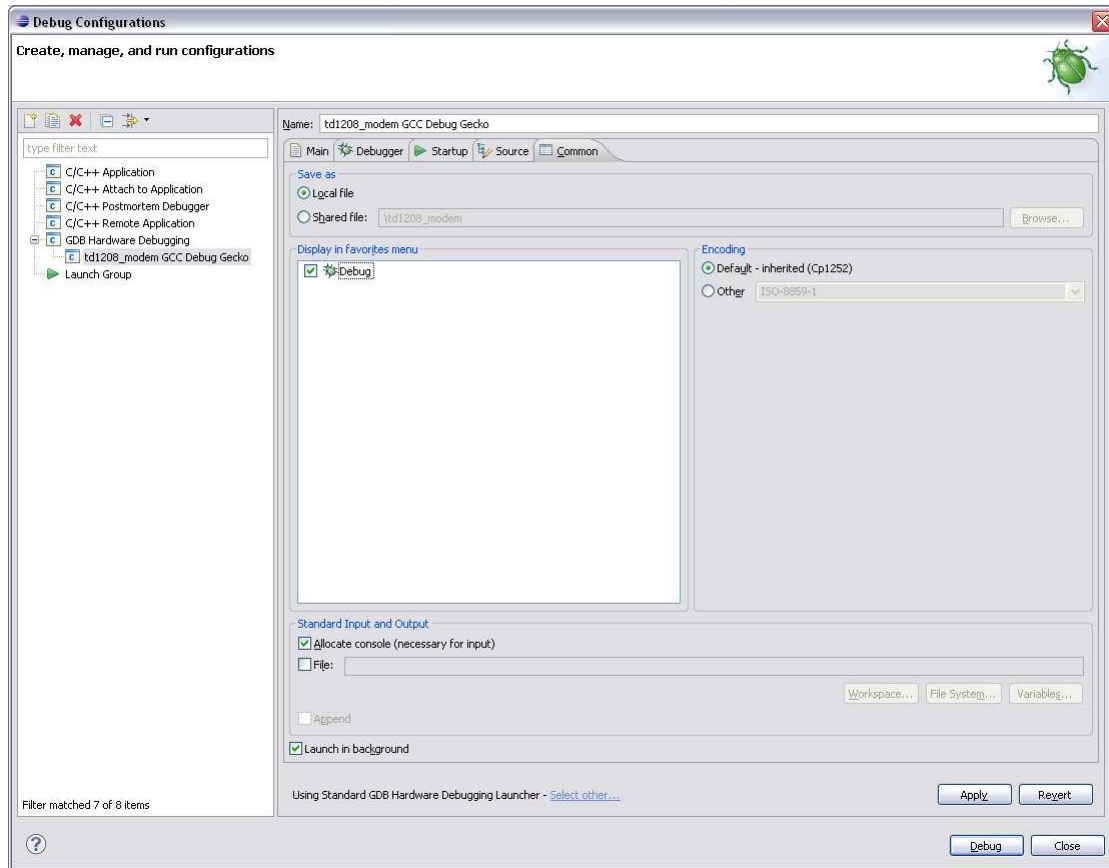



Figure 13 - Common Debug Configuration

2.3 GDB Server Launch Configuration Creation

In the main menu bar, click on the downwards arrow right to the **“External Tools”** button , then select **“External Tools Configurations...”** into the menu:

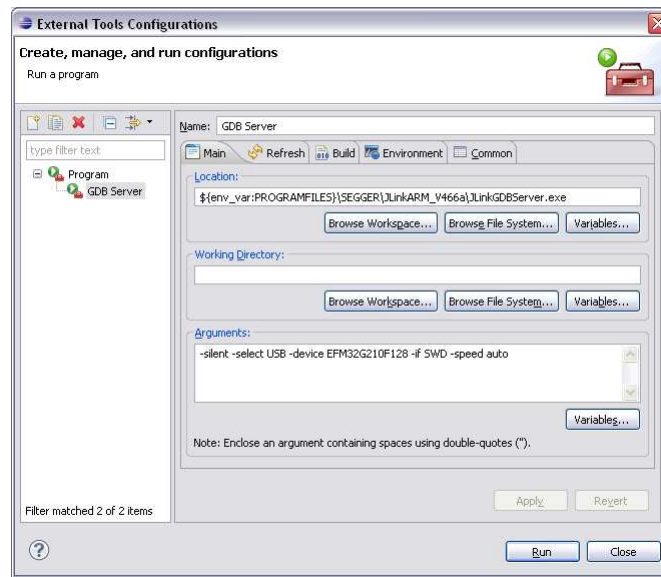


Figure 14 - External Tools Configuration

In the **"Name:"** field, enter **"GDB Server"**.

In the **"Location:"** field, enter:

```
${env_var:PROGRAMFILES}\SEGGER\JLinkARM_V466a\JLinkGDBServer.exe
```

In the **"Arguments:"** field, enter:

```
-silent -select USB -device EFM32G210F128 -if SWD -speed auto
```

Click on the **"Common"** tab and check the **"External Tools"** entry into the favorite's menu tree:

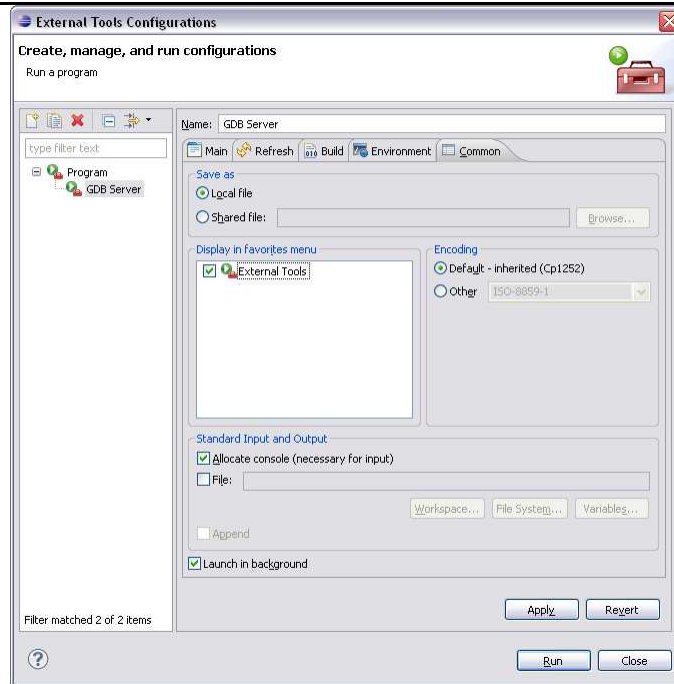



Figure 15 - Common External Tools Configuration

Click on the “**Apply**” and “**Close**” buttons to save the new configuration.

2.4 TD1208 Debug Session

First, make sure that all hardware devices are properly connected:

- The TD1208 on either the TD1208EVB or the customer's board should be connected to the Energy Micro EFM32 Tiny Gecko EFM32TG-STK3300 Starter Kit or the dedicated J-Link SWD/JTAG USB Adapter using a flat ribbon cable and a small adapter PCB
- The Energy Micro EFM32 Tiny Gecko EFM32TG-STK3300 Starter Kit or the dedicated J-Link SWD/JTAG USB Adapter must be connected to the PC using a standard Mini-USB cable

Launch the GDB server from Eclipse by pressing on the “**External Tools**” button , then select “**GDB Server**” into the menu, a new windows will pop up:

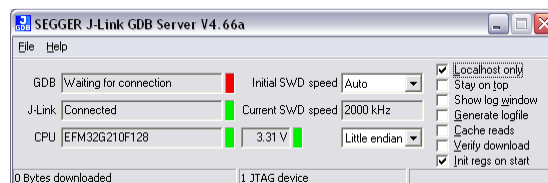



Figure 16 - J-Link GDB Server

The 3 green rectangles show that the J-Link is properly connected, as well as the TD1208 CPU and that the power supply voltage is within range. If one of these indicators is read, please check the corresponding hardware.

Launch the debug session by clicking on the downwards arrow right to the “bug” button , then select the desired entry into the menu: the corresponding firmware will first be built if not already done, then flashed into the TD1208 flash memory, then launched automatically.

The first time a debug session is launched, Eclipse prompts you if you want to open a “Debug Perspective”: this is a panel arrangement specifically aimed at debugging, with dedicated breakpoints, memory, source panes. Check the “Remember my decision” checkbox and click on “Yes”:

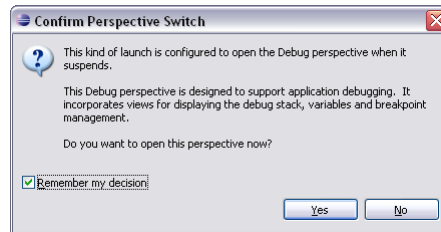


Figure 17 - Confirm Perspective Switch

The Eclipse panels will get reorganized in a new way, making it easy to debug:

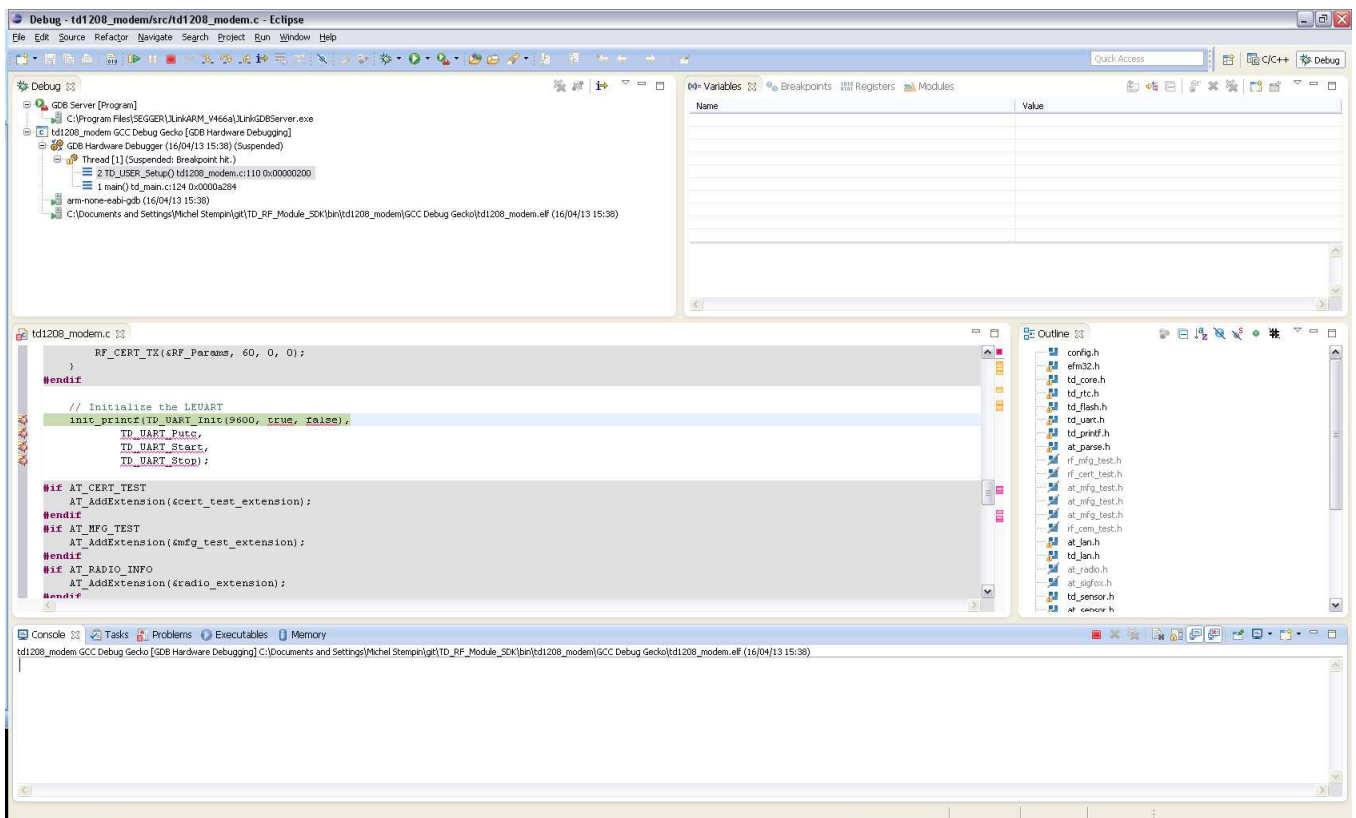


Figure 18 - Debug Perspective

The menu bar provides most of the common commands used for debugging:

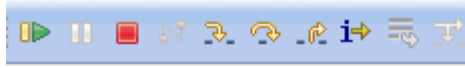


Figure 19 - Debug toolbar

From left to right:

- **Resume (F8)** : to resume program execution
- **Suspend** : to suspend program execution
- **Terminate (F2)**: to terminate program execution
- **Disconnect**: this command is disabled
- **Step Into (F5)**: to step the program execution, entering into called functions
- **Step Over (F6)**: to step the program execution, without entering into called functions
- **Step Return (F7)**: to step the program execution until the exit of the current function
- **Instruction Stepping Mode**: to toggle between C and assembly language instruction stepping


Other common actions consist in performing breakpoints, registers or memory operations; please use the corresponding windows panels. If one desired panel is not available, navigate to the “**Window > Show View**” submenu to display them.

To terminate the debugging session, select the “**GDB Hardware Debugger**” entry into the tree and press on the “**Terminate**” menu icon.

To switch back to the standard C/C++ perspective, click on the corresponding top-right icon.

3 Flash a TD1208 Firmware from Eclipse

It is possible to flash a firmware directly from Eclipse. For this, an external tool launcher configuration must be created first.

In the main menu bar, click on the downwards arrow right to the **“External Tools”** button , then select **“External Tools Configurations...”** into the menu:

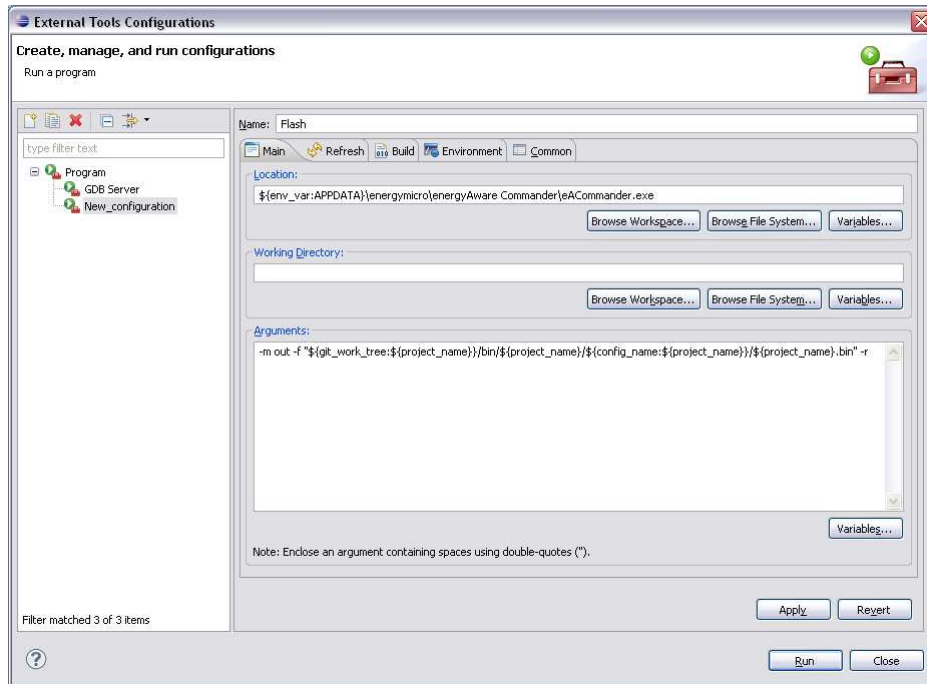


Figure 20 - Flash Launch Configuration

In the **“Name:”** field, enter **“Flash”**.


In the **“Location:”** field, enter:

```
${env_var:APPDATA}\energymicro\energyAware Commander\EACommander.exe
```

In the **“Arguments:”** field, enter:

```
-m out -f
"${git_work_tree:${project_name}}\bin/${project_name}/${config_name:${project_name}}
/${project_name}.bin" -r
```

Click on the **“Apply”** and **“Close”** buttons to save the Flash launch configuration.

It is now possible to Flash a TD1208 firmware by selecting the corresponding project into the **“Project Explorer”** tree, then clicking on the **“External Tools”** button , then selecting **“Flash”** into the menu.

DOCUMENT CHANGE LIST

Revision 1.0

- First Release

Revision 1.1

- Changed contact information

NOTES:

CONTACT INFORMATION

Telecom Design S.A.

Europarc — 22 Avenue Léonard de Vinci
33600 PESSAC, France
Tel: +33 5 57 35 63 70
Fax: +33 5 57 35 63 71

Please visit the Telecom Design web page:

<http://www.telecomdesign.fr/>

The information in this document is believed to be accurate in all respects at the time of publication but is subject to change without notice. Telecom Design assumes no responsibility for errors and omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Telecom Design assumes no responsibility for the functioning of undescribed features or parameters. Telecom Design reserves the right to make changes without further notice. Telecom Design makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Telecom Design assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Telecom Design products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Telecom Design product could create a situation where personal injury or death may occur. Should Buyer purchase or use Telecom Design products for any such unintended or unauthorized application, Buyer shall indemnify and hold Telecom Design harmless against all claims and damages.

Telecom Design is a trademark of Telecom Design S.A.

SIGFOX™ is a trademark of SigFox S.A.

Other products or brand names mentioned herein are trademarks or registered trademarks of their respective holders.