
User's Guide

(1.0.49.0)

iSYSTEM Debug Plug-in for Eclipse**Table of Contents**

Introduction	2
Debug Plug-In Functionality	2
System Requirements	3
Supported platforms.....	4
Installation	5
Eclipse 3.4 (Ganymede).....	5
Eclipse 3.5 (Galileo).....	8
Eclipse 3.6 (Helios).....	9
Eclipse 3.7 (Indigo).....	10
CodeWarrior.....	10
Usage	10
Create a project in Eclipse / CDT.....	10
Write some code and compile it.....	10
Create a winIDEA workspace.....	10
Using an existing winIDEA workspace (recommended).....	10
Creating a debug configuration.....	12
Special Function Registers View.....	16
Real-time Expressions View.....	17
The Debug View and Stack Depth.....	18
Start debugging session without downloading the code.....	18
Launching without debugging.....	18
Making copies of project.....	18

Introduction

This document describes the installation and use of iSYSTEM Debug Plug-In for Eclipse/CDT.

Debug Plug-In Functionality

The iSYSTEM Debug Plug-In (in further text plug-in) can communicate to iSystem development tools and execute debug commands. All standard debug functionalities available in CDT are also implemented in the plug-in. Additionally, *SFR view* and *Real-time expressions view* are also available. Comparing to winIDEA, the plug-in does not implement trace, execution coverage and profiler functionality. The plug-in therefore provides more than CDT debugger, but less than winIDEA. It currently still requires winIDEA to run in the background, but during the debugging, no switching to winIDEA is necessary.

System Requirements

Before starting with the installation the system must contain the following:

1. **winIDEA 9.9.46** or newer
To download the latest winIDEA, go to <http://www.isystem.si/isystem/winIDEALinks/>
2. **Java 1.6, 32-bit**. The plug-in will install, but will not be available if older version of Java is installed!
3. **Eclipse 3.4.0 or later, 32-bit**
To download the latest Eclipse go to <http://www.eclipse.org/downloads/>. If we want to use Eclipse for C/C++ development only, there is already a package with CDT available (*Eclipse IDE for C/C++ Developers*). In this case we should skip CDT installation described in the next step.
4. **CDT (C++ Development Toolkit)**
To obtain CDT go to <http://www.eclipse.org/cdt/downloads.php>
CDT should be installed as Eclipse plug-in.

Supported platforms

The plug-in has been tested with the following debug platforms:

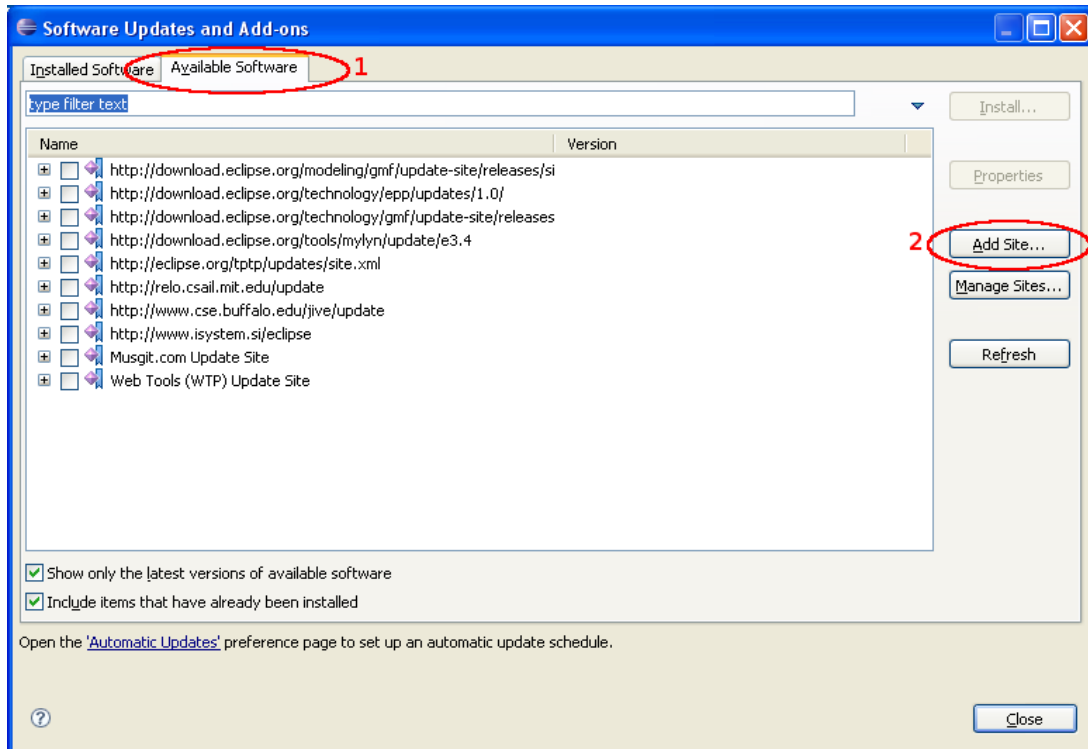
- Freescale MC9S12XDP512 ActivePRO POD
- ARM ARM9 (AT91RM9200)
- Infineon XC166/XC2000 (XC2287)
- NEC NEC78K ActivePRO POD
- Freescale MPC55xx (MPC5554)
- NEC V850ES/Fx3 ActivePRO POD
- Freescale ColdFire (CF5213)
- ARM Cortex-M3 (STM32)

The plug-in should work also with other architectures supported by iSYSTEM tools since the interface to the iSYSTEM debugger is the same for all architectures.

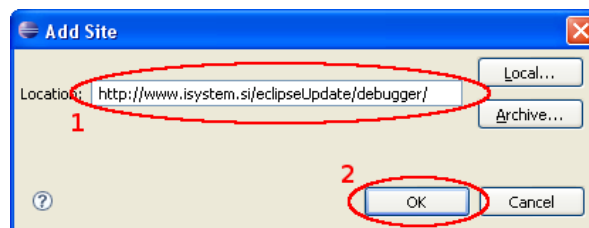
Installation

Eclipse 3.4 (Ganymede)

The plug-in uses the Eclipse's standard installation procedure*. Start Eclipse and select the main menu option 'Help | Software updates'. The 'Software Updates and Add-ons' dialog opens. Select the tab 'Available Software' and then click 'Add site'.

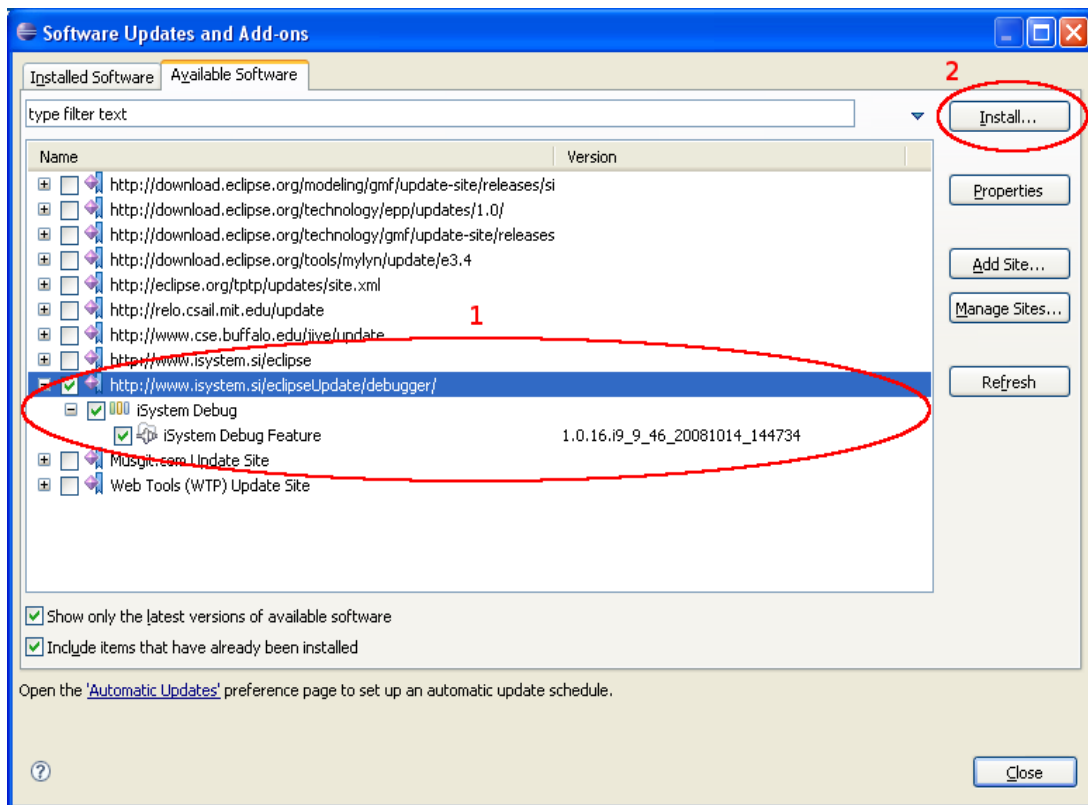


The 'Add site' dialog opens:



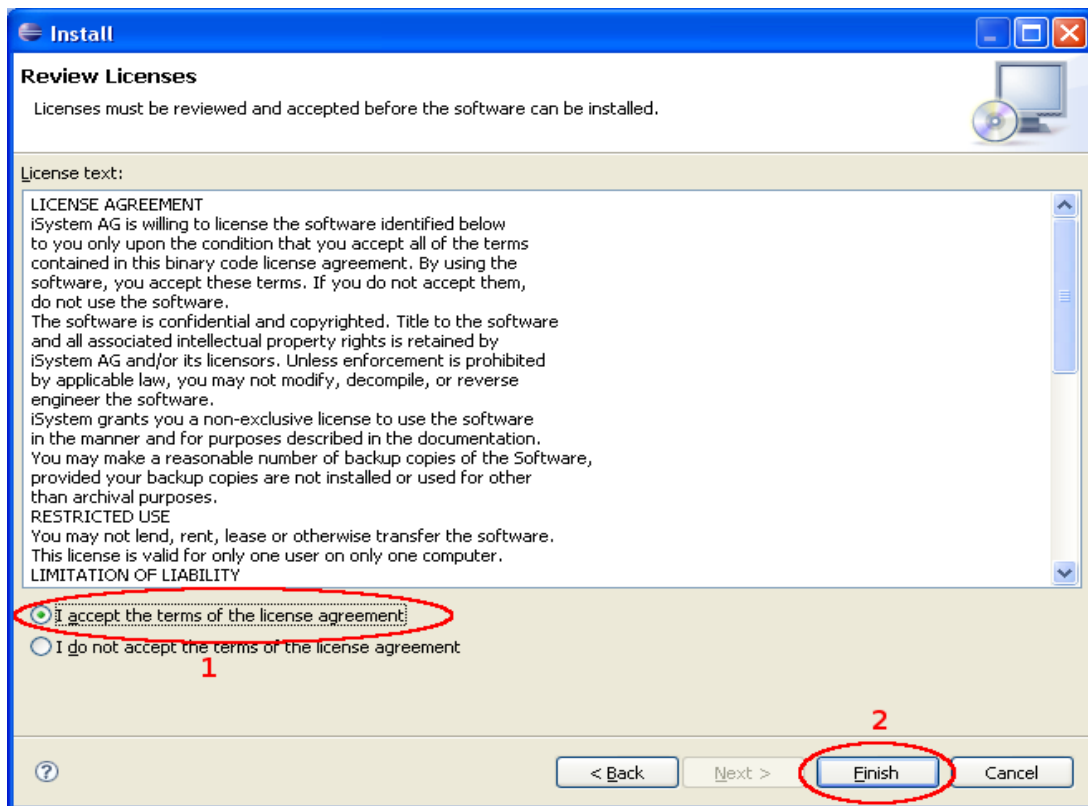
To install the plug-in from the Internet, enter the update site '<http://www.isystem.si/eclipseUpdate/debugger/>', and click the 'OK' button to continue. In the 'Software Updates and Add-ons' dialog we should see a new site and select the iSystem Debug Feature as shown in the image below.

* This section describes the new Update Manager introduced in Ganymede. If you have problems with it, it is still possible to use the old one. More information can be found at http://wiki.eclipse.org/Equinox_p2_Removal. Another option is manual installation by unzipping the zip file into Eclipse installation directory.

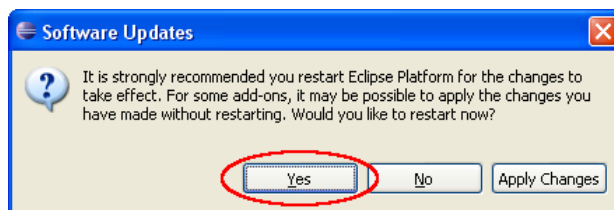


Then we click the 'Install' button.

The dialog shown below opens. Read the license agreement and click 'I accept the terms of the license agreement' if you agree with it.

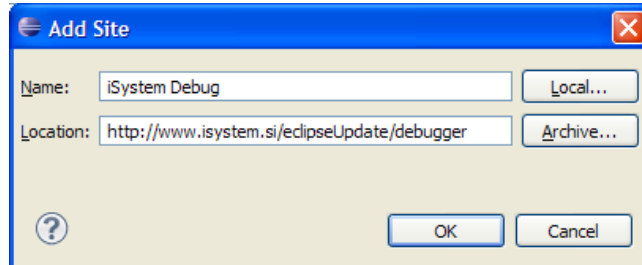


Click 'Finish'. It may take few minutes to install the plug-in. Finally we restart Eclipse and the installation is finished.

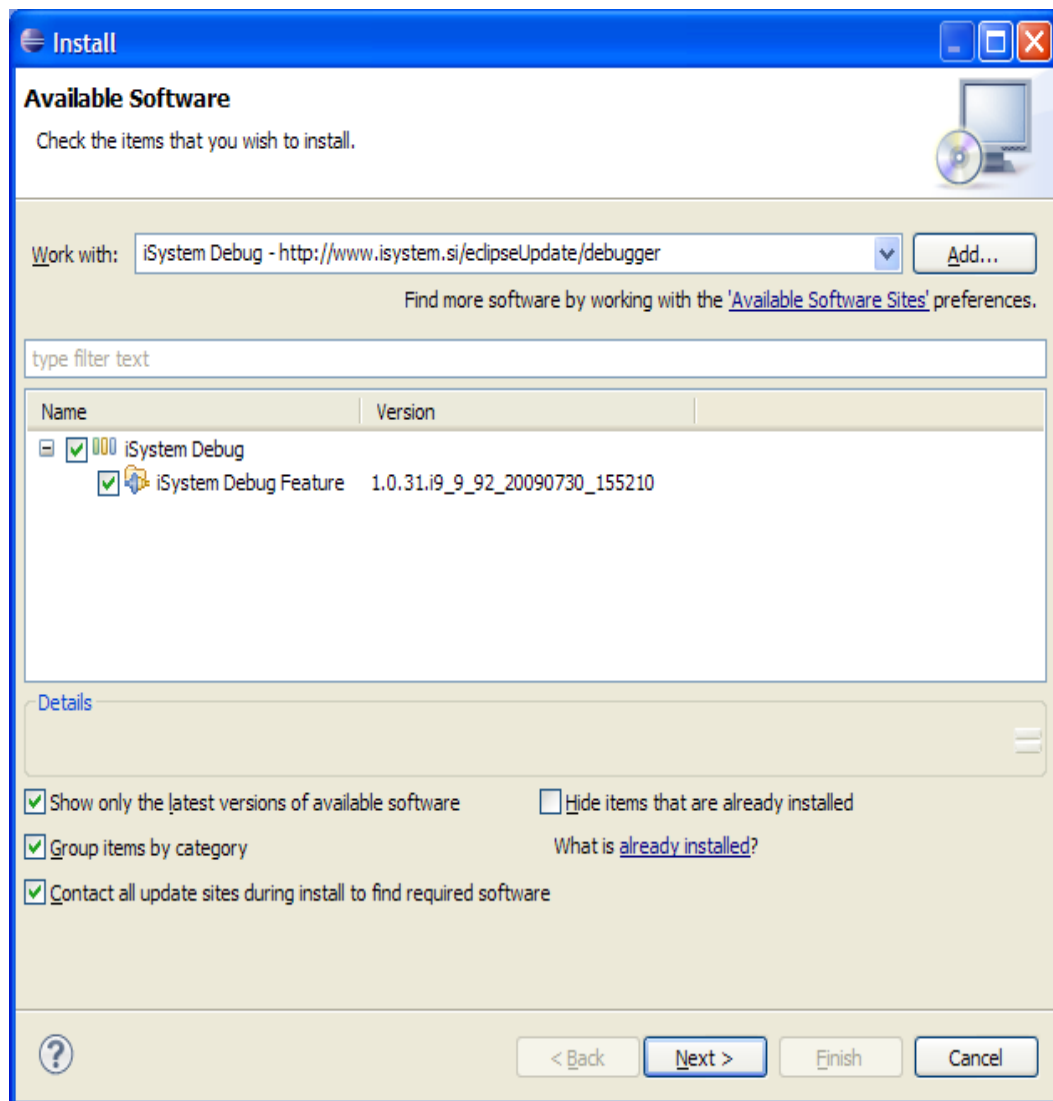


Eclipse 3.5 (Galileo)

The plug-in uses the Eclipse's standard installation procedure. Start Eclipse and select the main menu option 'Help | Install New Software'. The 'Install' dialog opens. Click the 'Add' button and enter the following info to the 'Add site' dialog. The update site is '<http://www.isystem.si/eclipseUpdate/debugger/>':



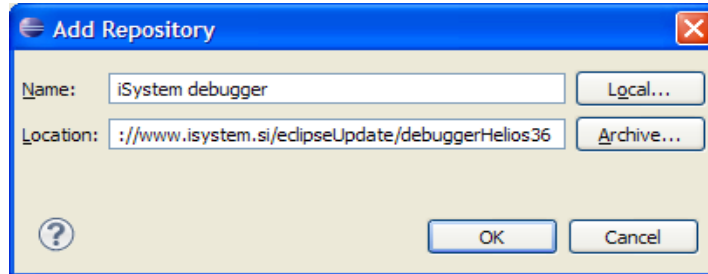
The new site and iSystem Debug plugin should appear in the 'Install' dialog. Select the iSystem Debug plug-in:



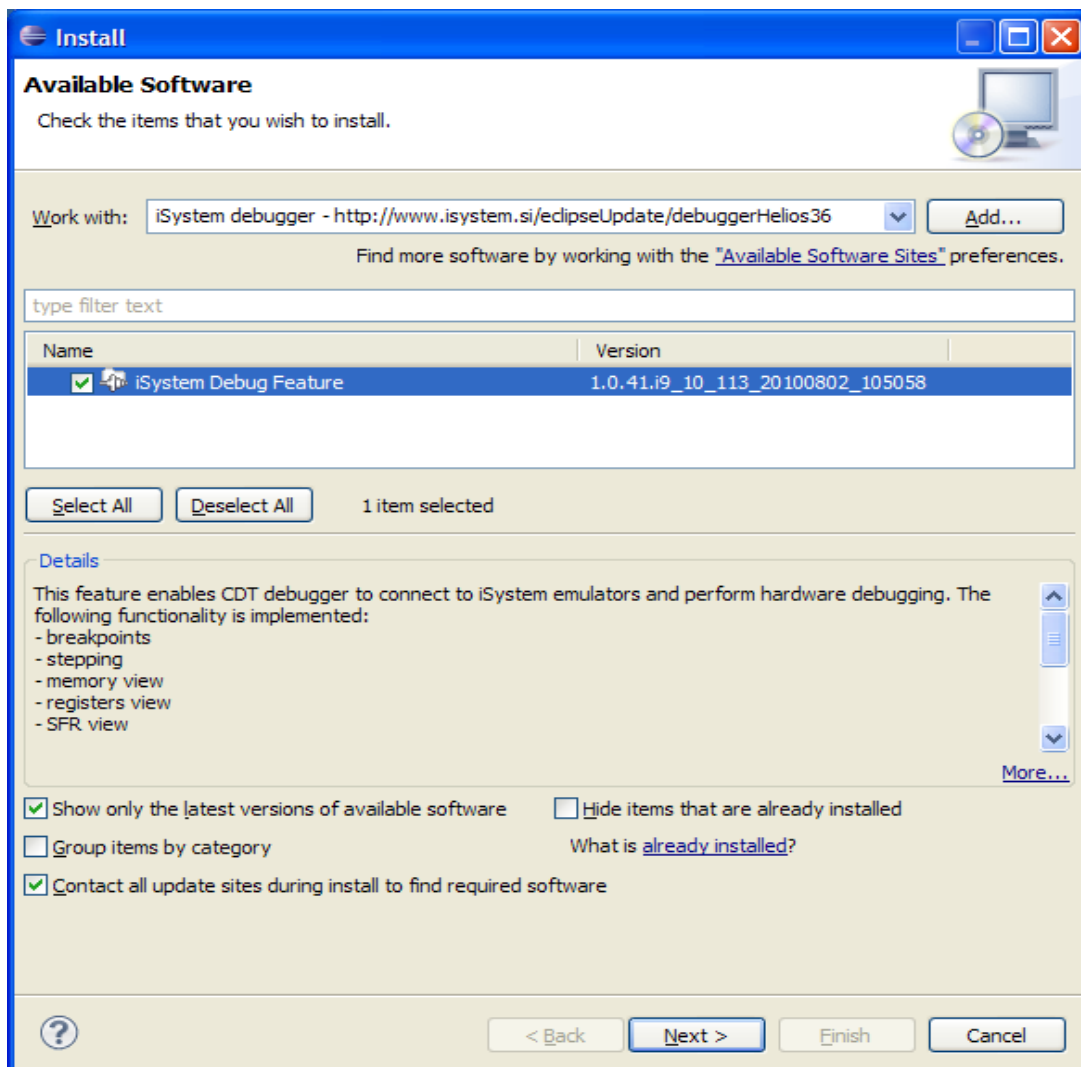
Click 'Next' to view install details, then click 'Finish'. Wait until the installation is finished and confirm restart. The plug-in is installed.

Eclipse 3.6 (Helios)

The plug-in uses the Eclipse's standard installation procedure. Start Eclipse and select the main menu option 'Help | Install New Software'. The 'Install' dialog opens. Click the 'Add' button and enter the following info to the 'Add repository' dialog (Location should be set to: <http://www.isystem.si/eclipseUpdate/debuggerHelios36>)



The new site and iSystem Debug plugin should appear in the 'Install' dialog. Select the iSystem Debug feature:



Click 'Next' to view install details and accept license agreement, then click 'Finish'. Wait until the installation is finished and confirm restart. The plug-in is installed.

Eclipse 3.7 (Indigo)

The installation procedure is the same as for Eclipse 3.6 (Helios) above.

CodeWarrior

The installation procedure is the same as for standard Eclipse, but we have to be careful about the version of the plug-in which we install. To find the version of Eclipse which serves as a base of our CodeWarrior, select `Help | About CodeWarrior Development Studio`. A dialog opens, where we should click the button `Installation details`. Select tab `Features` in the next dialog and check the version of `Eclipse Platform` feature. Depending on this version you install the iSystem's debug plug-in for Eclipse 3.4 and 3.5 or 3.6 and 3.7 as described above. For example, CW 10.2 beta is based on Eclipse 3.6.1.

To enable selection of iSystem's debug plug-in, we must enable generic C/C++ debugger capabilities:

- Select `Window | Preferences` from main menu and click `General | Capabilities` on the Preferences window
- Select `Development` node in the left tree and then hit `Advanced ...` button in the right lower corner
- In the pop-up menu enable `Development | Generic C/C++ GNU Development`

This will enable you to see `C/C++ Local Application in Debug Configuration` window as described for Eclipse below.

Usage

The typical development procedure consists of the following steps:

Create a project in Eclipse / CDT

This step is described in the Eclipse/CDT help.

Write some code and compile it

See the Eclipse/CDT help for details. Make sure you know the name and location of the output file, which should be downloaded to the target.

Create a winIDEA workspace

There are two options: we can use an existing winIDEA workspace configured for our target, or create a new one from scratch.

Using an existing winIDEA workspace (recommended)

Minor modifications are necessary to migrate an existing winIDEA project to Eclipse.

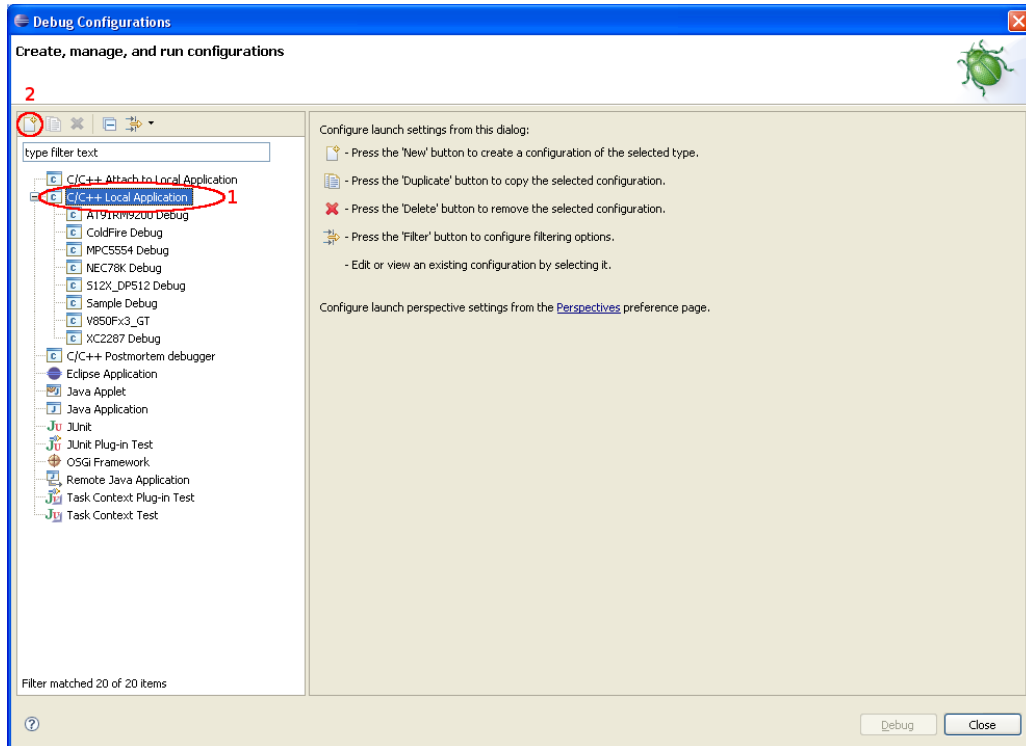
Typically, files which are downloaded by the debugger contain the code and the debug information. Source files can be removed from winIDEA project manager, if the debug information contains source files' absolute paths. If not, the user must either specify Eclipse project source files in the winIDEA project manager or specify the alternate project files search directories in winIDEA (Debug/Debug Options/Directories tab).

Creating a new winIDEA workspace

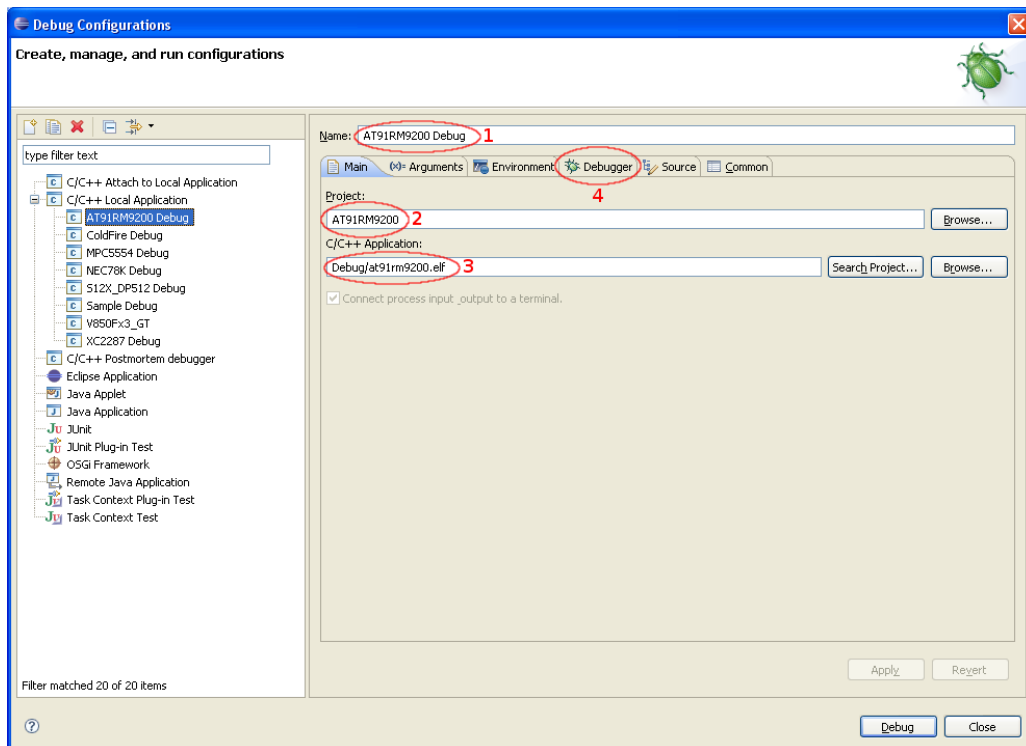
First we create a new workspace, then configure the emulator (Hardware | Hardware), and the target options (Hardware | Emulation Options). Consult the winIDEA manual for details.

Creating a debug configuration

Before we can run debugger from Eclipse, we need to create a debug configuration. The main menu option 'Run | Debug configurations ...' opens the Debug configurations dialog.



First we select C/C++ Local Application, then we create a new configuration. Next we enter important information into the new configuration as shown below.



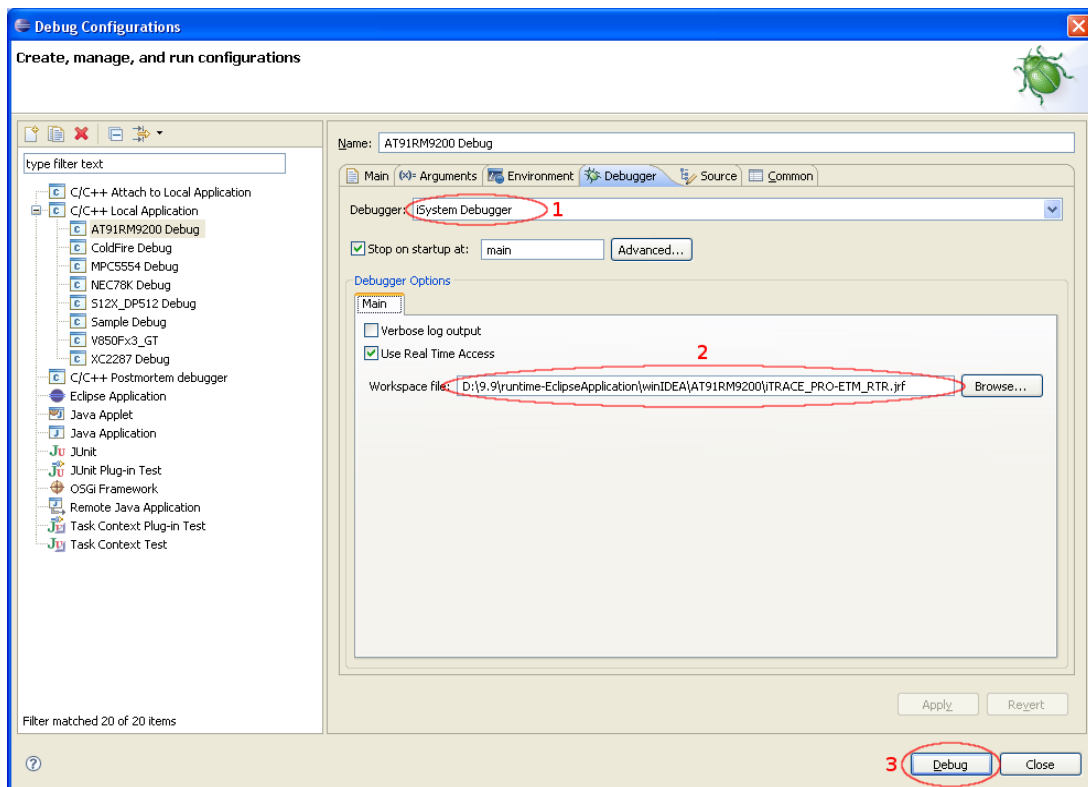
Fields have the following meaning:

Name – configuration name, which is shown in the list of configurations on the left

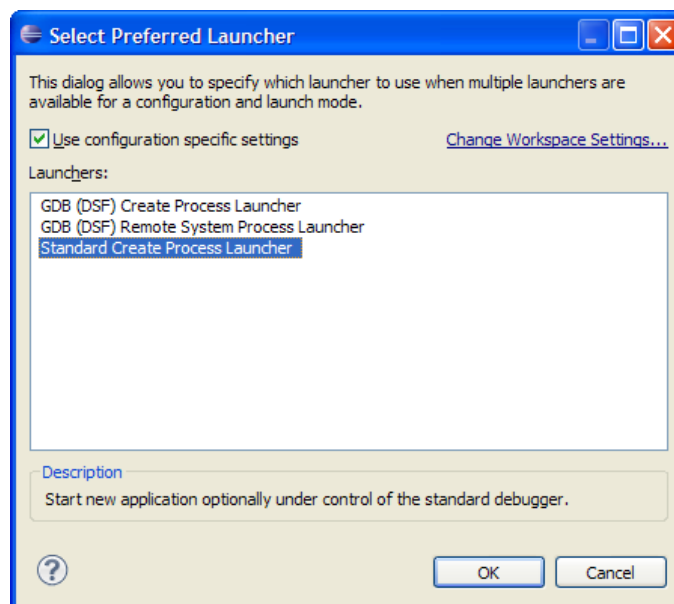
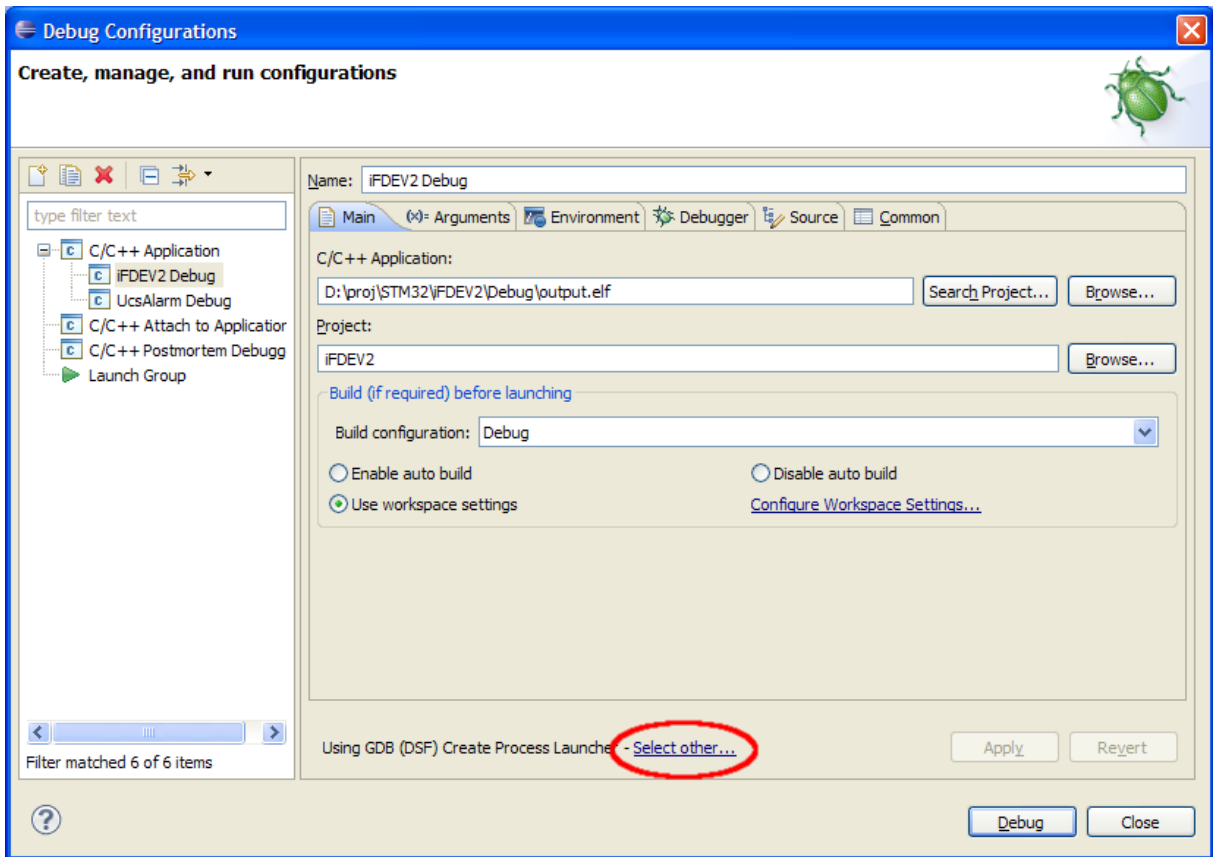
Project – the project which the configuration belongs to

C/C++ Application – the executable file, which will be downloaded to the target. If we want to change this name later, after it has been downloaded, we have to remove the old file name in winIDEA 'Debug | Files for download' dialog.

Finally we open a 'Debugger' tab and select the 'iSystem Debugger' in the 'Debugger' field. **Eclipse 3.4** and **3.5** show the possibility to select iSystem Debugger directly as shown below:



In **Eclipse 3.6 (Helios)** we have to select the preferred launcher first. GDB(DSF) is used by default, but we need a Standard Create Process Launcher. Click *Select Other...* link in the dialog as shown below, then select the *Standard Create Process Launcher*.



Click OK, and now we can select iSystem debugger also in Eclipse 3.6 (Helios).

Dialog entries for iSystem Debugger have the following meaning:

Stop on startup at – defines if the execution should stop after startup, and where it should stop. Note, that the winIDEA option Debug | Files for download | Options | After download must be set to 'nothing (Stop)', for this option to work properly.

Verbose log output – if checked, a detailed log of debugger execution is stored into log file. Since it takes some space and time, we usually leave this unchecked. In case of problems we can check this option and send log file to support team.

Use Real-time Access – if checked, real time access is enabled. It is important only for Real-time Expressions view.

Workspace file – this field contains the path to the workspace file that will be opened by winIDEA. It's extension must be either '.jrf' or '.xjrf'. Use of later one is recommended. If the file does not exist, winIDEA will ask us to create one.

After we click the 'Debug' button, the plug-in performs the following:

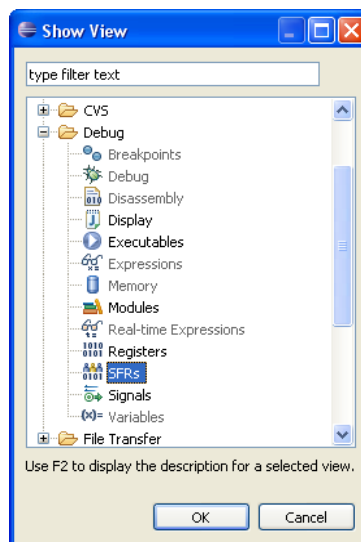
- It checks if winIDEA is already running. If not, one is started and winIDEA workspace open.
- If the output file is specified, and it's not in winIDEA's download list (see option 'Debug | Files for Download'), it is added to the list.
- The plug-in finally executes download on winIDEA, which then downloads all files from its download list to the target. During the download, Eclipse does not open a pop-up dialog with progress bar, but we can see progress in the bottom right corner of the Eclipse window. See also 'Window | Show view | Other | General | Progress' to open the Progress view.

After download the debug session is running.

The plug-in does not contain any other setting for winIDEA, for example hardware related settings. If there are any download related error messages, try to solve the problem in winIDEA. First check the download list in winIDEA (option 'Debug | Files for Download') and then try to perform download directly in winIDEA. When this is successful, check the output file name and location in Eclipse.

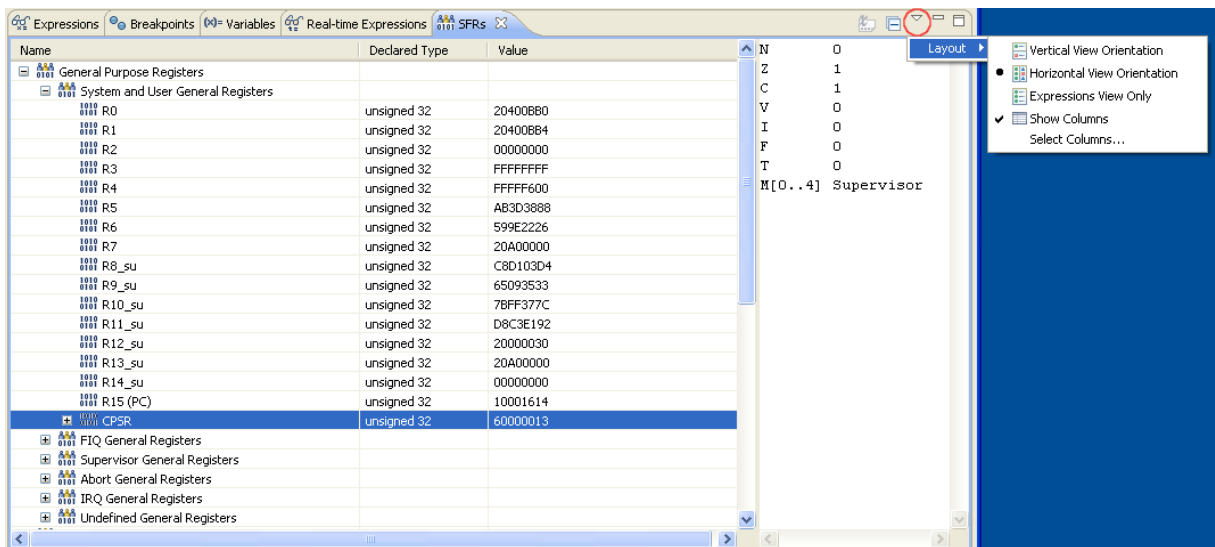
Special Function Registers View

In addition to CDT debugging view, iSystem debugger also provides access to special function registers. To open the SFR view, select the main menu option 'Window | Show view | Other | Debug | SFRs' as shown below.



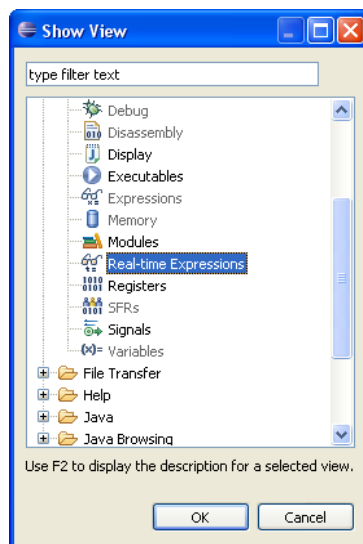
The SFR view is added to Eclipse workbench. SFRs are grouped in several groups. If a group which contains SFRs, or SFR which is composed of groups of bits, is selected, the detailed information is displayed in the details pane. For example, in the next image we can see bits of the CPSR register in the details pane on the right.

There is also a layout menu, which gives options for different view layouts.

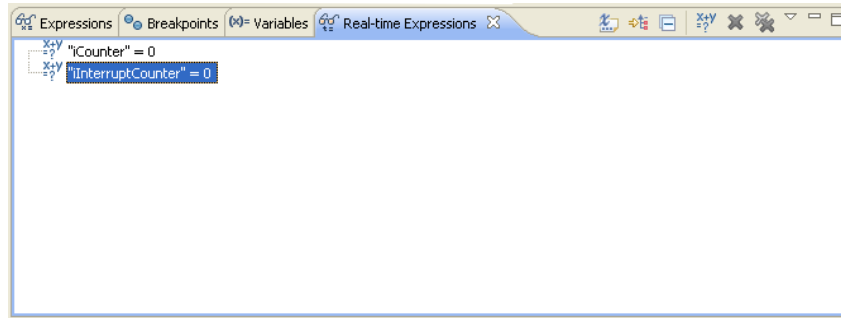


Real-time Expressions View

When the target is not in suspended state, Eclipse does not display information about target state, because it is normally not accessible. However, iSystem hardware may enable us to observe global variables also during the running state (real-time access). If real-time access is available, check the check box in the debug configuration (see section *Creating a Debug Configuration*). Then select the main menu option 'Window | Show view | Other | Debug | Real-time expressions' as shown below.

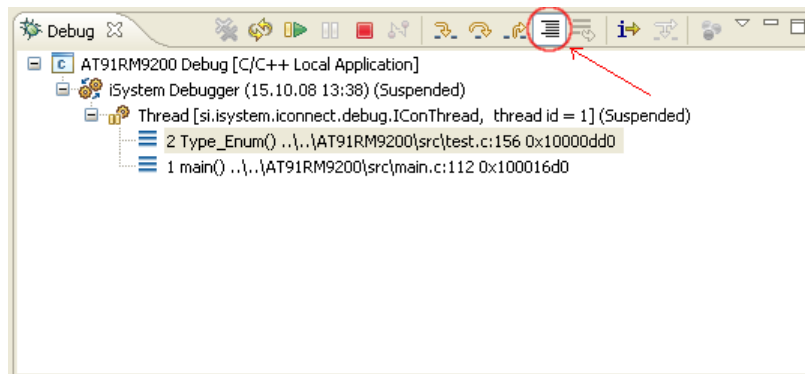


The Real-time expressions view is added to the Eclipse workbench.




The Debug View and Stack Depth

The *Debug view* displays stack frames. The stack must be reconstructed after each debug step command, which may take more than one or two seconds if there are many stack frames. Such delays are not convenient during stepping through the program code, and most of the time only the current stack frame is needed anyway. For this reason a toggle button was added, to turn full stack display on and off. The button is shown in the image below.



When the button is highlighted, full stack is shown.

Start debugging session without downloading the code

If downloading is slow, for example due to slow flash programming, and we know that the code did not change since the last debugging session, we can switch off the code downloading. If the toolbar button  is pressed, then only symbols are downloaded when debugging session starts. The button has toggle state – each click reverses its state.

Launching without debugging

If we want to run the application without the debugger, we can do it by executing one of Run commands. In this case the plug-in tries to get the output file name from Eclipse. The output file name is available only if we use *managed build* in Eclipse. The output file name is specified in project properties: 'C/C++ Build | Settings | Build Artifact', fields 'Artifact Name', and 'Artifact extension'.

If there is no output file found in Eclipse project, we have to add output files manually to winIDEA.

Making copies of project

If we want to copy project files to other locations, we have to move all source files and project files. In case of Eclipse with CDT and iSystem debugger, the following files have to be copied:

- `.project` and `.cproject` – Eclipse and CDT project files, contain information on how to build the project

- `<winIDEAWorkspaceFileName>.xjrf` – winIDEA workspace file contains hardware configuration

Disclaimer: iSYSTEM assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information herein.

© iSYSTEM. All rights reserved.