LAUTERBACH
DEVELOPMENT TOOLS

# TRACE32 as TCF Agent

Release 09.2023

MANUAL

# TRACE32 as TCF Agent

**TRACE32 Online Help**

**TRACE32 Directory**

**TRACE32 Index**

**TRACE32 Documents** ..... 🗁

**TRACE32 as TCF Agent** ..... **1**

**Introduction** ..... **4**

Restrictions 5

Documentation Updates 6

Related Documents and Tutorials 6

**Initial Setup and Requirements** ..... **7**

TRACE32 7

Eclipse 8

Wind River Workbench 10

Synopsys MetaWare IDE 12

**TRACE32 Setup** ..... **15**

Installing the TRACE32 TCF Eclipse Plug-In 15

Option A: Manual Configuration 17

Option B: Select Executable and Configuration File 22

Configuration File 24

T32Start 24

**Establish a Debug Session** ..... **25**

Start TRACE32 25

TCF Discovery 27

Manual Debug Target Setup 29

Open Debug Perspective Automatically 31

TRACE32 View 32

**Troubleshooting** ..... **33**

TRACE32 33

TCF=(illegal command) 33

Eclipse 33

No TRACE32 PowerView instance under “Available Targets” 33

Cannot locate peer TCP:<ip>:<port> 33

Help Us to Help You 34

Export the TRACE32 System Information 34

Export the Eclipse Error Log 34

Export the Eclipse Configuration 35

**TCF Commands** ..... **36**
SYStem.TCFconfig TCF-specific setups 36
SYStem.TCFconfig.TASKCONTEXT Enable/disable task contexts 36

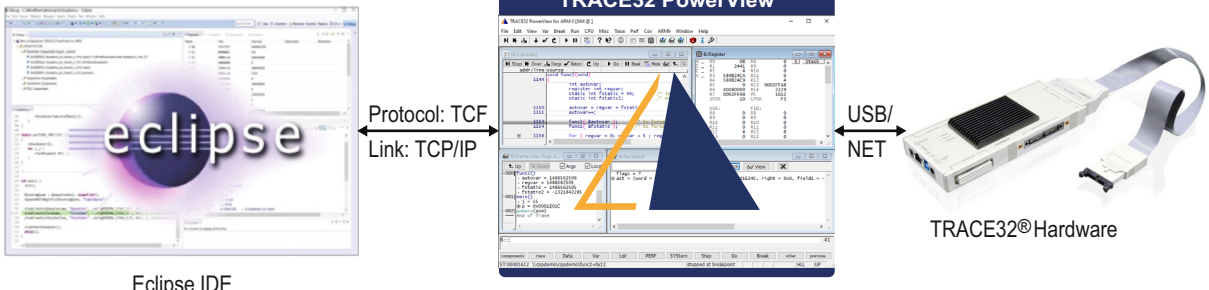
# TRACE32 as TCF Agent

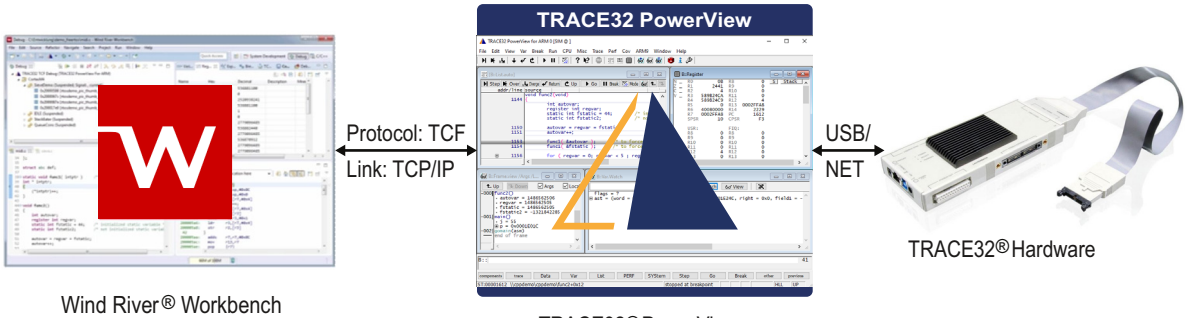
**Version 10-Oct-2023**

## Introduction

The Target Communication Framework (TCF) is a vendor-neutral lightweight network protocol mainly for communication with embedded systems (http://wiki.eclipse.org/TCF).

TRACE32 PowerView can be configured as a TCF agent. This TCF interface is useful to access the debug functionality of TRACE32 from within an Eclipse-based interface. Simultaneous usage of TRACE32 PowerView and Eclipse is also possible.



The TRACE32 TCF integration supports the following debugging features:

- Run control (**Go**, **Break**, **Step**,...)
- Software and on-chip breakpoints
- Register view
- Expressions view
- Memory view
- Display of the source code and the disassembly (mixed mode) for the selected context
- Stack trace
- Display of the operating system tasks in the Eclipse Debug view.
- Function sample-based profiling
- Debug Symbol Browser for the Wind River Workbench
- Multi-core debug in AMP and SMP

The TRACE32 TCF integration also provides a synchronization between TRACE32 PowerView and Eclipse. For example, setting a breakpoint or executing a single step at the TRACE32 side will be reported to Eclipse and vice versa.

| | |
|---|---|
| **NOTE:** | The TRACE32 TCF integration is available for all architectures and compilers supported by TRACE32. |

| | |
|---|---|
| **NOTE:** | The debug symbols have to be loaded in TRACE32 PowerView and do not need to be loaded in Eclipse. |

## Restrictions

This solution is not a full integration of TRACE32 in the described IDEs. Not all features of TRACE32 PowerView are available in Eclipse or Wind River® Workbench. This solution should be only be used for debugging bare-metal applications or simple operating systems as FreeRTOS.

The following features are for instance not supported by this solution:

- Peripheral views, MMU and Cache views
- Memory access classes. The memory view in Eclipse only shows the current context.
- Trace
- MMU and Cache views
- OS Awareness
- Hypervisor debugging
- FLASH programming

Please contact support@lauterbach.com for more information.

## Documentation Updates

The latest version of this document is available for download from:
**www.lauterbach.com/pdf/app_tcf_setup.pdf**

## Related Documents and Tutorials

- For information about how to install TRACE32, see **"TRACE32 Installation Guide"** (installation.pdf).
- For a video tutorial about TRACE32 as a TCF agent for Wind River Workbench, visit: **support.lauterbach.com/kb/articles/demo-of-the-trace32-integration-to-the-wind-river-workbench**
- For a video tutorial about TRACE32 as a TCF agent for Eclipse, visit: **support.lauterbach.com/kb/articles/demo-of-the-trace32-integration-to-eclipse**

# Initial Setup and Requirements

**In this chapter:**


- **TRACE32**
- **Eclipse**: If you want to integrate TRACE32 with Eclipse, then skip the *Wind River Workbench* section in this document.
- **Wind River Workbench**:
  - If you want to integrate TRACE32 with Wind River Workbench, then you have to take the additional steps described in section "**Wind River Workbench**".
  - After that, continue with section **"TRACE32 Setup"**, page 15
- **Synopsys MetaWare IDE**: If you want to integrate TRACE32 with the MetaWare IDE (a special version of Eclipse)


## TRACE32

For information about how to install TRACE32 under MS Windows, see **"MS Windows"** in TRACE32 Installation Guide, page 21 (installation.pdf). We recommend that you install TRACE32 on the system path suggested by the installer: C:\T32.

For information about how to install TRACE32 under Linux, see **"PC_LINUX"** in TRACE32 Installation Guide, page 23 (installation.pdf).

A TRACE32 version from **February 2016 or later** is required. If the TRACE32 version is too old, then you will get an error box with the message **"TCF=(illegal command")"** when trying to start TRACE32 PowerView as a TCF agent.

- To check the TRACE32 version, choose **Help** menu > **About TRACE32**.

For a description of how to configure and start TRACE32 as a TCF agent, see **"TRACE32 Setup"**, page 15.

# Eclipse

According to the Eclipse TCF documentation, the following components are required:

- JDK 1.8.0 or later
- Eclipse SDK 3.8 or later
- CDT (C/C++ Development Tools) SDK 8.1 or later

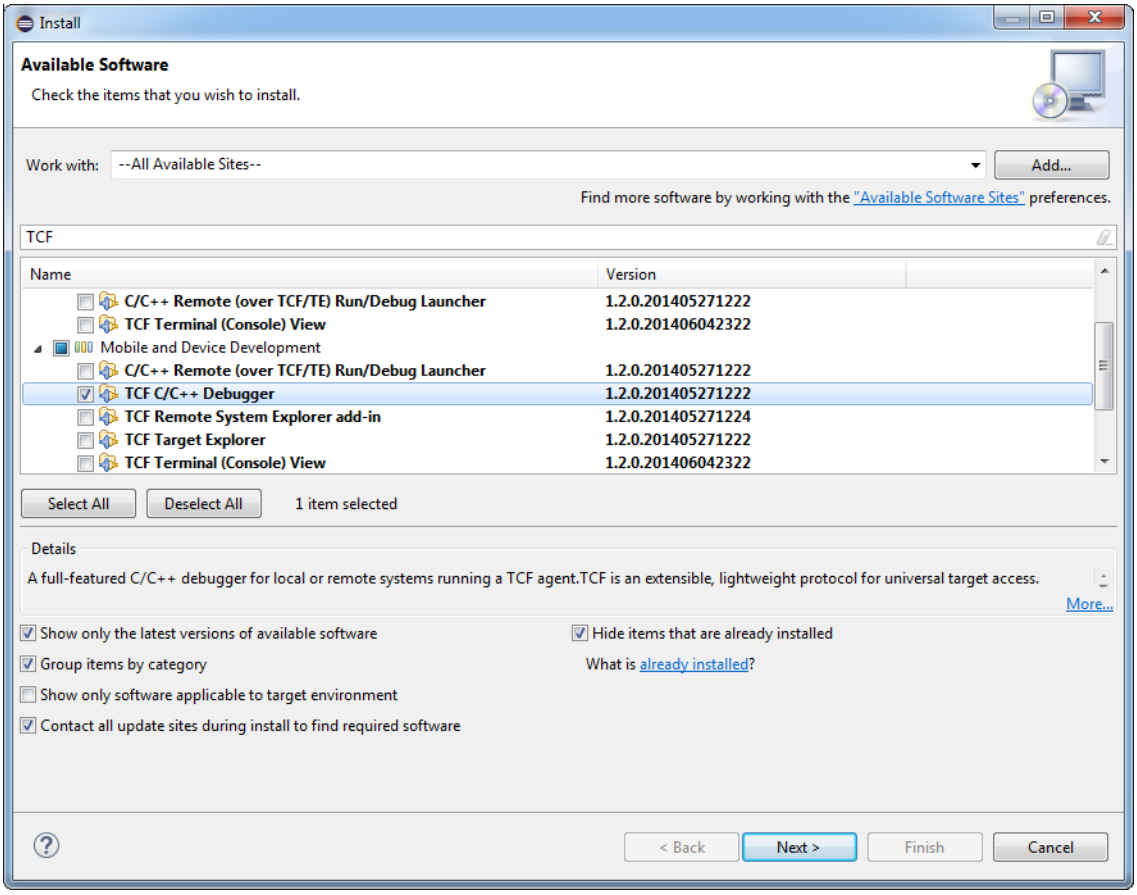
For more information about the minimal required setup, please refer to the TCF Wiki page (http://wiki.eclipse.org/TCF).

| | |
|---|---|
| **NOTE:** | For Eclipse 3.7 (Indigo) and 3.6 (Helios) you need to install the “Target Communication Framework (Incubation)” plug-in from http://download.eclipse.org/tools/cdt/releases/indigo. |

**To install the TCF C/C++ Debugger in Eclipse (3.8 or newer):**

1. Choose **Help** menu > **Install New Software**.
2. From the **Work with** list, select **-- All Available Sites --** or select an update site for your Eclipse version under http://www.eclipse.org/tcf/downloads.php.
3. Type “TCF” in the search field.

   You may have to wait for the list to be populated.
4. Select the **TCF C/C++ Debugger**.

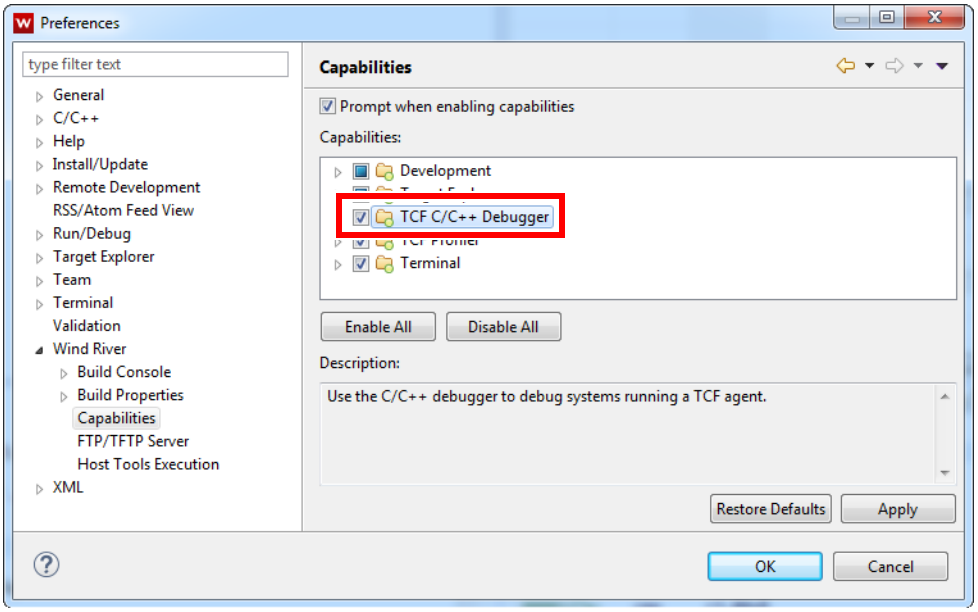



5. Click the various **Next** buttons and follow the instructions of the install wizard to finish the installation.
6. Restart Eclipse.

| NOTE: | For the Wind River Workbench 3.3, you need to install the “Target Communication Framework (Incubation)” plug-in from http://download.eclipse.org/tools/cdt/releases/indigo.<br>The On Chip Debug (OCD) version of the Workbench is not supported by this integration. |
|---|---|

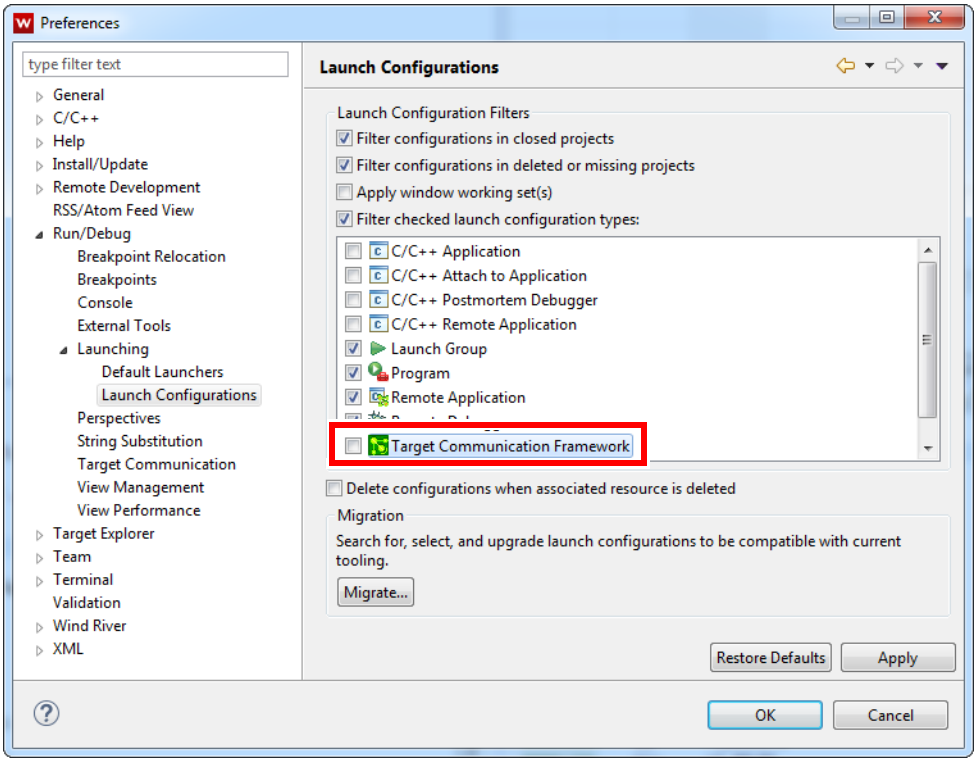
You need to enable the **TCF C/C++ Debugger** in the Wind River Workbench.

1. Choose **Windows** menu > **Preferences**.
2. In the **Preferences** dialog, click **Wind River** > **Capabilities**.
3. Select the check box **TCF C/C++ Debugger**.



Moreover, you need to disable the launch configuration filter for the **Target Communication Framework**.

1. Choose **Windows** menu > **Preferences**
2. In the **Preferences** dialog, click **Run/Debug** > **Launching** > **Launch Configurations**.
3. Clear the check box **Target Communication Framework**.



# Synopsys MetaWare IDE

The MetaWare IDE is based on Eclipse. Thus you can basically you can use TCF the same way than described for native Eclipse before.

Please note that the MetaWare IDE contains proprietary changes to Eclipse. As a result Lauterbach can't ensure that controlling TRACE32 via TCF is fully functional. Thus it is recommended to use native Eclipse.

**To install the TCF C/C++ Debugger in Synopsys MetaWare IDE:**

1. Choose **Help** menu > **About MetaWare IDE** > **Installation Details** > **Features**
2. Check the version number of the Eclipse Platform. This is the Eclipse version on which you copy of MetaWare IDE is based on. You must have version 4.2 or higher.

| MetaWare | IDE | based on Eclipse Platform | Comment |
|---|---|---|---|
| L-2016.06 | 10.3 | Eclipse 4.4 (Luna) | TCF successfully tested |
| L-2016.03 | 10.2 | Eclipse 4.4 (Luna) | |
| K-2015.12 | 10.1 | Eclipse 4.4 (Luna) | |
| K-2015.09 | 10.0 | Eclipse 4.4 (Luna) | |
| J-2015.03 | 9.8 | Eclipse 4.4 (Luna) | |
| J-2014.12 | 9.7 | Eclipse 4.4 (Luna) | |
| J-2014.06-SP1 | 9.6 | Eclipse 4.4 (Luna) | |
| J-2014.06 | 9.5 | Eclipse 4.3 (Kepler | |
| I-2013.12.1 | 9.4 | Eclipse 4.3 (Kepler) | |
| I-2013.12 | 9.3 | Eclipse 4.3 (Kepler) | |

3. Choose **Help** menu > **Install New Software...**

4. In dialog windows “Install” add the in the Work with box the update site according to the version of the Eclipse platform you got before.

| Eclipse Platform | Update Site (p2 Repository) |
|---|---|
| Eclipse 4.14 (2019-12) | http://download.eclipse.org/releases/2019-12 |
| Eclipse 4.13 (2019-09) | http://download.eclipse.org/releases/2019-09 |
| Eclipse 4.12 (2019-06) | http://download.eclipse.org/releases/2019-06 |
| Eclipse 4.11 (2019-03) | http://download.eclipse.org/releases/2019-03 |
| Eclipse 4.10 (2018-12) | http://download.eclipse.org/releases/2018-12 |
| Eclipse 4.9 (2018-09) | http://download.eclipse.org/releases/2018-09 |
| Eclipse 4.8 (Photon) | http://download.eclipse.org/releases/photon |
| Eclipse 4.7 (Oxygen) | http://download.eclipse.org/releases/oxygen |
| Eclipse 4.6 (Neon) | http://download.eclipse.org/releases/neon |
| Eclipse 4.5 (Mars) | http://download.eclipse.org/releases/mars |
| Eclipse 4.4 (Luna) | http://download.eclipse.org/releases/luna |
| Eclipse 4.3 (Kepler) | http://download.eclipse.org/releases/kepler |

You can get a full list of all “p2 Repositories” at **https://wiki.eclipse.org/Simultaneous_Release**

5. Type “TCF” in the search field. (You may have to wait for the list to be populated.)
6. Select the **TCF C/C++ Debugger**.
7. Click on **Next** and follow the dialogs, which guide you through the installation process.

It’s recommended to install the TRACE32 TCF Eclipse Plug-In as described below.

The MetaWare IDE uses several special proprietary views (child windows) when it is connected to a MetaWare (or SeeCode) debugger. When using TCF you have to use the native Eclipse debug views. To get the right views used **Window** menu > **Show View** ( > Other... )

| Data to display | MetaWare view | Native Eclipse View | TRACE32 Command |
|---|---|---|---|
| Core Register | Register (CDI) | Register | **Register.view** |
| Breakpoints | Breakpoints (MetaWare)* | Breakpoints | **Break.List** |
| Watchpoints | Watchpoints | Breakpoints | **Break.List** |
| Disassembled Code | Disassembly (MetaWare) | Disassembly | **List.Mix** |
| Raw Memory | Memory (MetaWare)* | Memory | **Var.DUMP** |
| Local Variables | Locals* | Variables | **Var.Local** |
| Global Variables | Global Variables* | Expressions | **Var.Watch** |
| HLL Expressions | Expressions (MetaWare)* | Expressions | **Var.Watch** |
| Call Stack | Call Stack* | Debug | **Frame.view** |
| ELF Sections | Modules (MetaWare) | Modules | **sYmbol.List.SECtion** |
| Auxiliary Register | Auxiliary register | N/A | **Data.dump AUX:0** |
| Hierarchic Symbol Tree | Executable | | **sYmbol.Browse** |

(* If marked with '*' you can also use the native Eclipse view instead.)

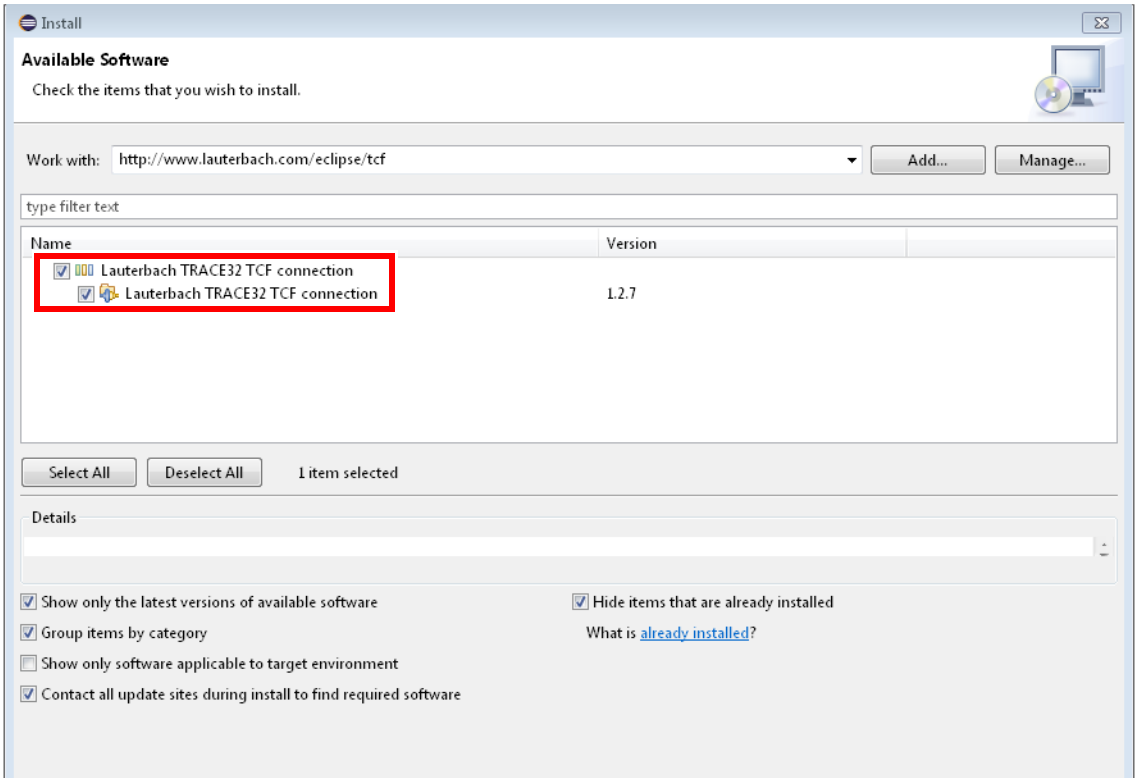
Don't mix up the Target Communication Framework (TCF) with the ARC target configuration files (TCF files). Both have nothing in commen except theire abbreviation.

## Installing the TRACE32 TCF Eclipse Plug-In

Lauterbach offers an Eclipse plug-in with a simplified and adapted launch configuration. Using this plug-in, you can configure and start TRACE32 from within Eclipse or the Wind River Workbench.

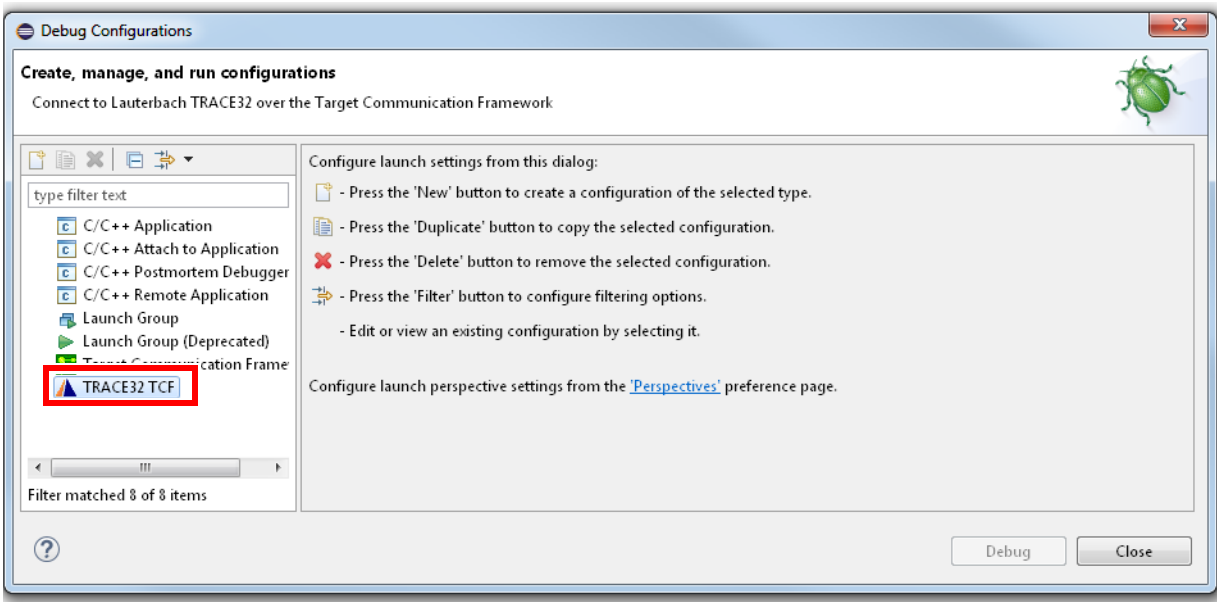
**To install the TRACE32 TCF Eclipse plug-in:**

1. Choose **Help** menu > **Install New Software**.
2. In the **Work with** box, type the update site: **https://www.lauterbach.com/eclipse/tcf**
3. Press **Enter**.
4. Make the selection shown below.




5. Click the various **Next** buttons and follow the instructions of the install wizard to finish the installation.
6. Restart Eclipse.
7. Choose **Run** menu > **Debug Configurations**.

   You should now have the **TRACE32 TCF** configuration in the **Debug Configurations** window, as shown below.




8. Double-click **TRACE32 TCF** to access the new **TRACE32** tab.

9. On the **TRACE32** tab, choose one of the following configuration methods:

**Option A** — If you have not worked with TRACE32 before, we recommend that you configure the different TRACE32 settings in the Eclipse plug-in by selecting the **Manual configuration** option.
See **"Option A: Manual Configuration"**, page 17.

**Option B** — If you already have a working TRACE32 configuration, we recommend that you specify the TRACE32 executable and the configuration file directly by selecting the **Select executable and configuration file** option.
See **"Option B: Select Executable and Configuration File"**, page 22.

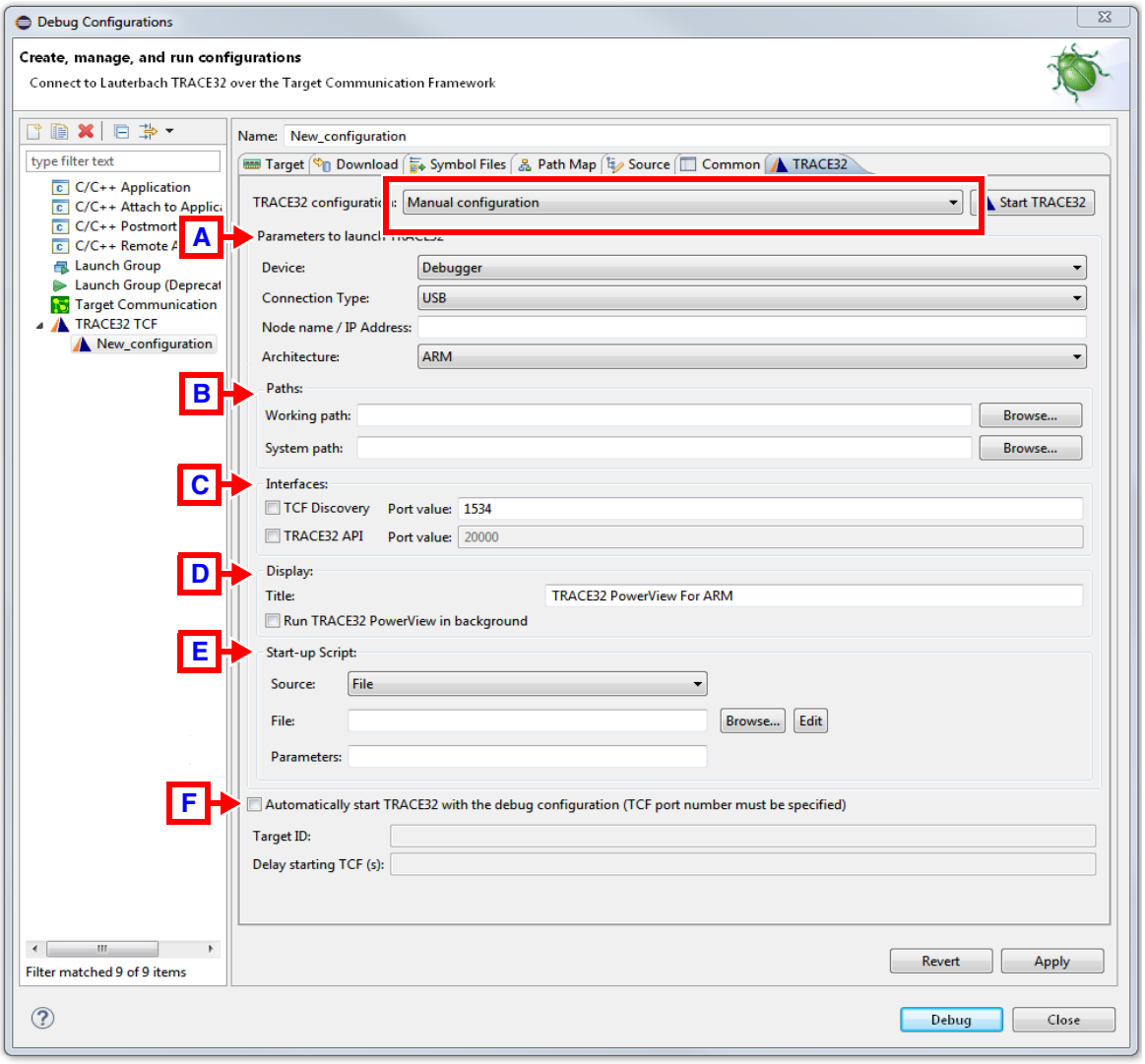
# Option A: Manual Configuration

1. If you want to configure the TRACE32 options manually, select **Manual configuration** from the **TRACE32 configuration** drop-down list in Eclipse.

   The fields for a manual configuration are will be displayed on the **TRACE32** tab.

2. Make your settings.

   - For a description of the fields on the **TRACE32** tab, see tables **below**.
   - A temporary configuration file will be created in the default temporary directory when TRACE32 is started.
   - For examples of the connection types USB and Ethernet, click **here**.




3. Click **Apply** when you are done.

You are now ready to start TRACE32 as described in **"Start TRACE32"**, page 25.

**[A] Parameters to launch TRACE32**

| | |
|---|---|
| Device | Select if you are using a JTAG Debugger or a TRACE32 Instruction Set Simulator. |
| Connection Type | For a JTAG Debugger, select if you have a USB or Ethernet connection to the Lauterbach debugger hardware. |
| Node Name / IP Address | **For USB connection**, optionally set the name of the connected device. This option is useful if multiple Lauterbach debuggers are connected per USB to one PC.<br><br>**For Ethernet connection**, set the IP address or the host name of the used Lauterbach debugger hardware. |
| Architecture | From the drop-down list, select the used target architecture (e.g. Arm, TriCore...). |

**[B] Paths**

| | |
|---|---|
| Working Path | Active directory after starting the TRACE32 instance. |
| System Path | Directory where the executable and system files of TRACE32 are located (e.g. C:\T32).<br><br>If you have observed our recommendation and installed TRACE32 on the system path suggested by the installer, you can ignore the step-by-step procedure below. |

**To determine the system path of a particular TRACE32 installation:**

1. Start TRACE32.
2. Type at the TRACE32 command line:

```
PRINT OS.PresentSystemDirectory()
                ;the message line below the TRACE32 command line
                ;displays the system path of this particular
                ;TRACE32 installation
```

**[C] Interfaces**

| | |
|---|---|
| TCF Discovery | Enable/disable the TCF discovery. If the discovery is disabled, a TCF port number must be specified (default: 1534).<br>The TCF discovery is a mechanism where agents advertise their peers by sending UDP packets to other agents. |
| TRACE32 API | Enable/disable the TRACE32 Remote API |

**[D] Display**

| | |
|---|---|
| Title | Set the window title of the TRACE32 instance. The title will be displayed in the **Name** column under **Available Targets** if the TCF discovery is enabled. |
| Run TRACE32 PowerView in background | Start TRACE32 PowerView without a graphical user interface. TRACE32 can be terminated using the Eclipse **Terminate** button. |

**[E] Start-up Script**

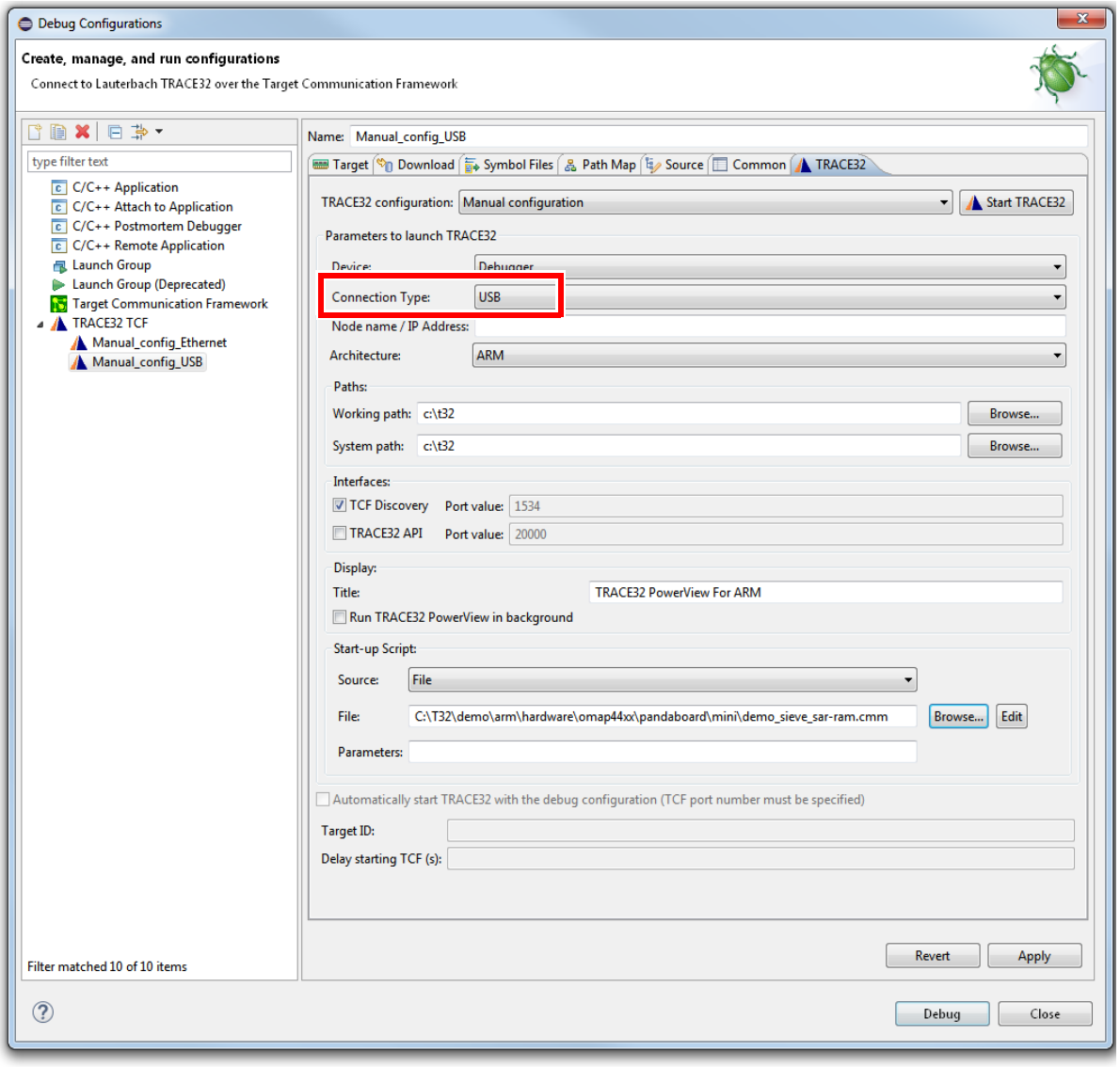
| | |
|---|---|
| Source | When a TRACE32 instance starts, the PRACTICE script **autostart.cmm** is executed, which then calls the following scripts:<br>• **system-settings.cmm** (from the TRACE32 system directory, usually C:\t32)<br>• **user-settings.cmm** (from the user settings directory: on Windows %APPDATA%\TRACE32 or ~/.trace32 otherwise)<br>• **work-settings.cmm** (from the current working directory)<br><br>In the TRACE32 TCF plug-in you can specify an additional PRACTICE script which is automatically started afterwards.<br><br>The TRACE32 TCF plug-in supports two types of start-up scripts:<br>• **File**<br>• **Built-in Script**<br>When **Source** is set to **File**, the script assigned to the **File** item will be executed. |
| File | If **Source** is set to **File**, specify the start-up script (*.cmm) here. |
| Parameters | Set the parameters that are passed to the start-up script (*.cmm) from **File**. |
| Built-in Script | The start-up script can be edited and stored directly. |

**[F] Start TRACE32 with the debug configuration**

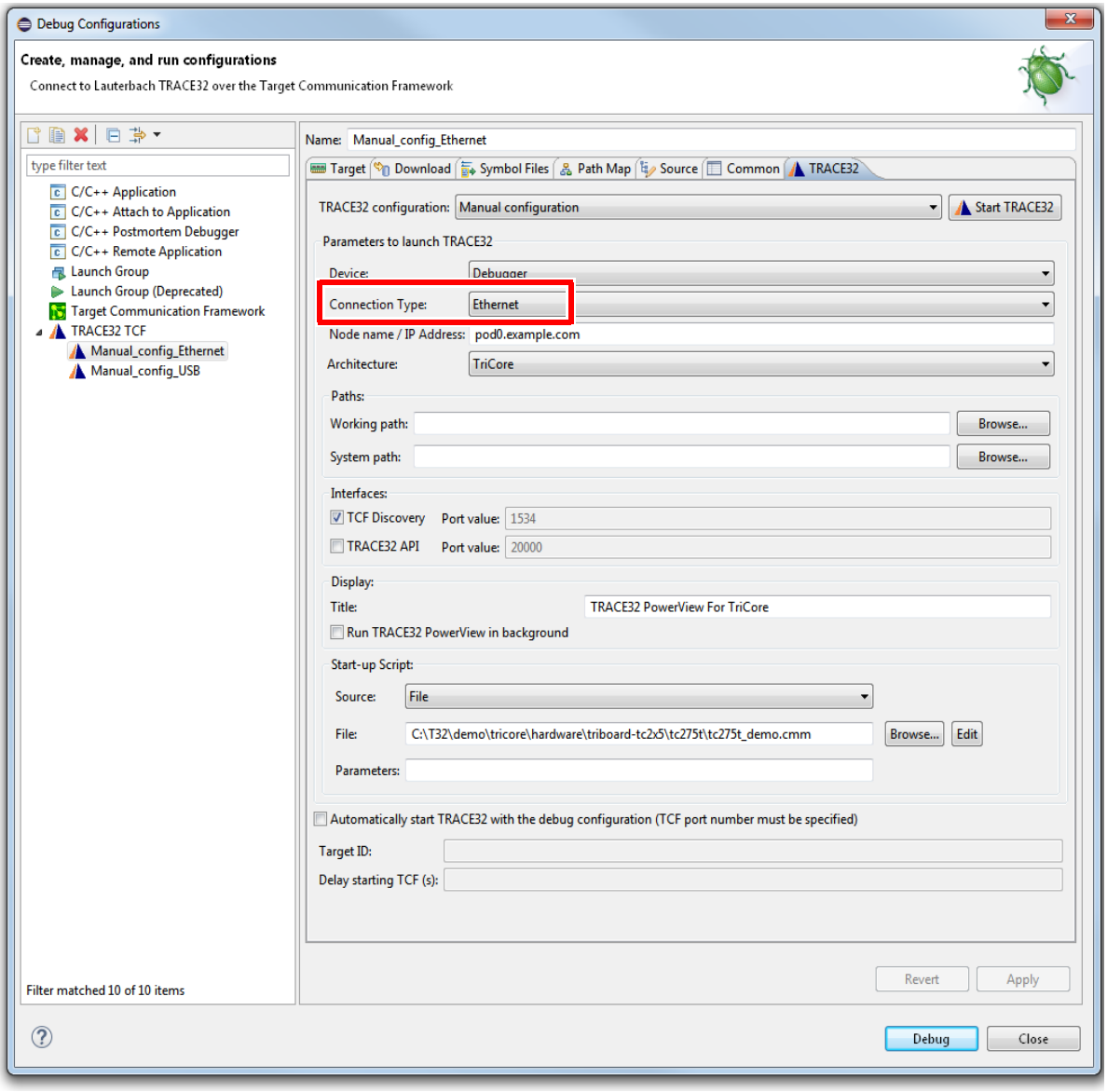
| | |
|---|---|
| Target ID | This option can be used only if TCF discovery is disabled.<br>A new target with a specific port must be declared first (see **Manual Debug Target Setup**). The new “Target ID” should be then written into this box. |
| Delay | An optional delay time until TRACE32 completes its start-up. |

| | |
|---|---|
| **NOTE:** | The TRACE32 TCF plug-in automatically creates a TRACE32 configuration file when you start TRACE32 from within Eclipse. |

## Example of the Connection Type USB



## Example of the Connection Type Ethernet



# Option B: Select Executable and Configuration File

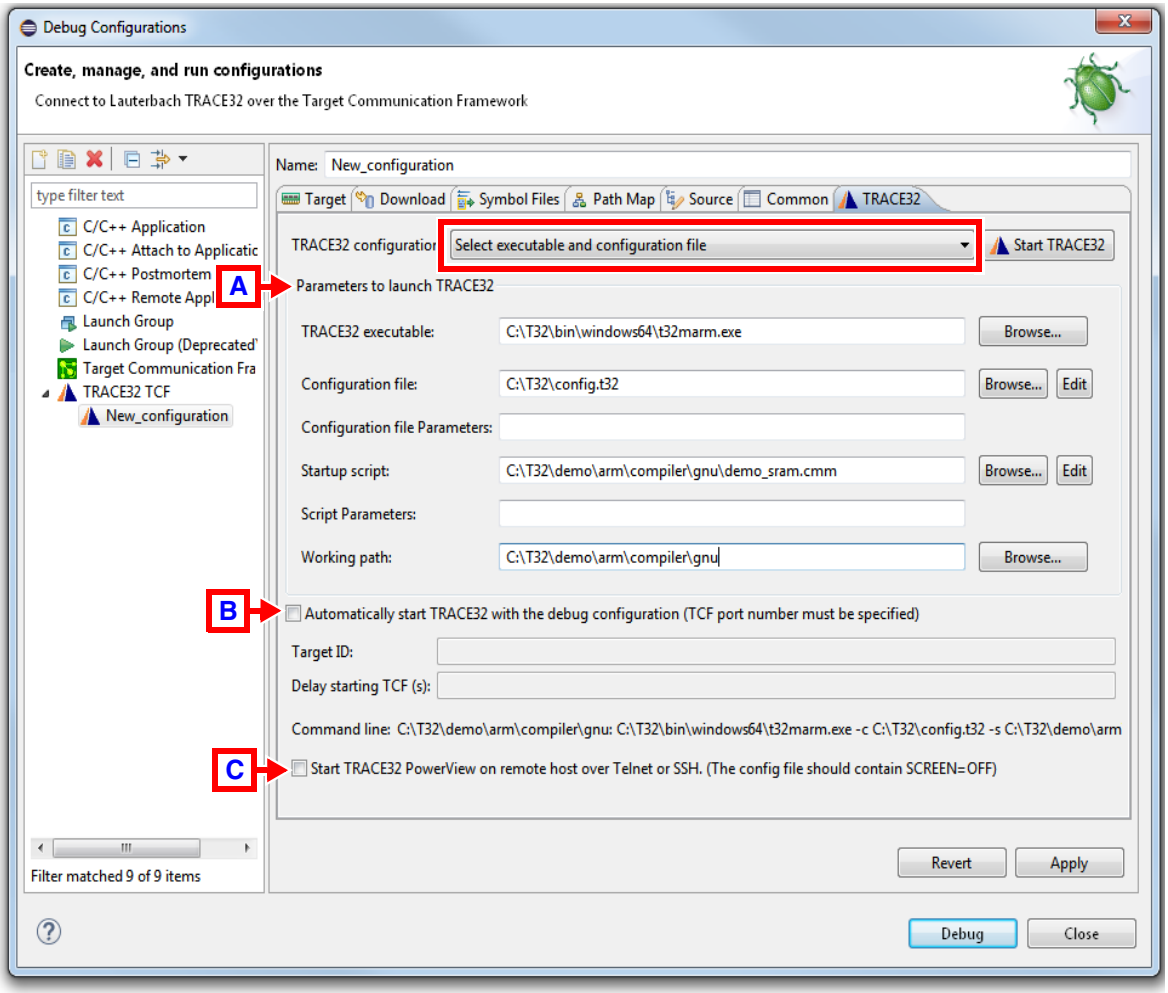
You can, instead of configuring the TRACE32 parameters manually, directly set the path to the TRACE32 executable and configuration file.

1. From the **TRACE32 configuration** drop-down list, **Select Executable and configuration file**.

   The fields for a direct configuration are will be displayed on the **TRACE32** tab.

2. Make your settings.

   For a description of the fields on the **TRACE32** tab, see tables below.




3. Click **Apply** when you are done.

**[A] Parameters to launch TRACE32**

| | |
|---|---|
| TRACE32 executable | Used TRACE32 executable e.g. C:\T32\bin\windows64\t32marm.exe |
| Configuration file | Used TRACE32 configuration file e.g. C:\T32\config.t32. The configuration file must contain the TCF block as described in **“Configuration File”**, page 24. |
| Configuration file parameters | Parameters for the configuration file |
| Start-up script | When a TRACE32 instance starts, the PRACTICE script **autostart.cmm** is executed, which then calls the following scripts:<br>• **system-settings.cmm** (from the TRACE32 system directory, usually C:\t32)<br>• **user-settings.cmm** (from the user settings directory: on Windows %APPDATA%\TRACE32 or ~/.trace32 otherwise)<br>• **work-settings.cmm** (from the current working directory)<br><br>Here you can specify an additional PRACTICE script which is automatically started afterwards. |
| Script parameters | Parameters for the start-up script file |
| Working path | Active directory after starting the TRACE32 instance. |

**[B] Start TRACE32 with the debug configuration**

| | |
|---|---|
| Target ID | This option can be used only if TCF discovery in disabled.<br>A new target with a specific port must be declared first (see **Manual Debug Target Setup**), then its “Target ID” should be written into this box. |
| Delay | An optional delay time until TRACE32 complete its startup. |

**[C] Start TRACE32 on remote host**

When checked, TRACE32 PowerView will be open in a remote machine using Telnet or SSH protocol.

The parameters specified in [**A**] should exists in the remote machine.

## Configuration File

To configure TRACE32 as a TCF agent, you need to add the following lines to your TRACE32 configuration file. The default configuration file is config.t32 and is located in the TRACE32 system directory.

```
                                    <- mandatory empty line
;T32 TCF Access                     <- optional comment line
TCF=
                                    <- mandatory empty line
```

An optional TCF port number can be added to the configuration file. If a port number is specified, then the TCF discovery mechanism is disabled in TRACE32. The TCF front-end (Eclipse) needs then to connect to TRACE32 using the specified port number. This will be explained in details later in this document.

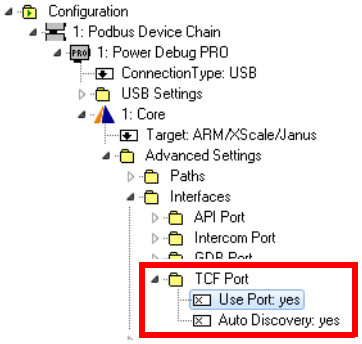
```
                                    <- mandatory empty line
;T32 TCF Access                     <- optional comment line
TCF=
PORT=1534                           <- TCF discovery disabled
                                    <- mandatory empty line
```

**An empty line is mandatory before and after the TCF block in the TRACE32 configuration file, otherwise, a "syntax error" will be reported when starting TRACE32.**

For more information about the TRACE32 configuration, please refer to **"Training Basic Debugging"** (training_debugger.pdf).

## T32Start

In case you are using t32start.exe utility to start TRACE32, you can enable TCF under **Advanced Settings** > **Interfaces** > **TCF Port.** Please note that at least t32start.exe version 2.4.7 is required.

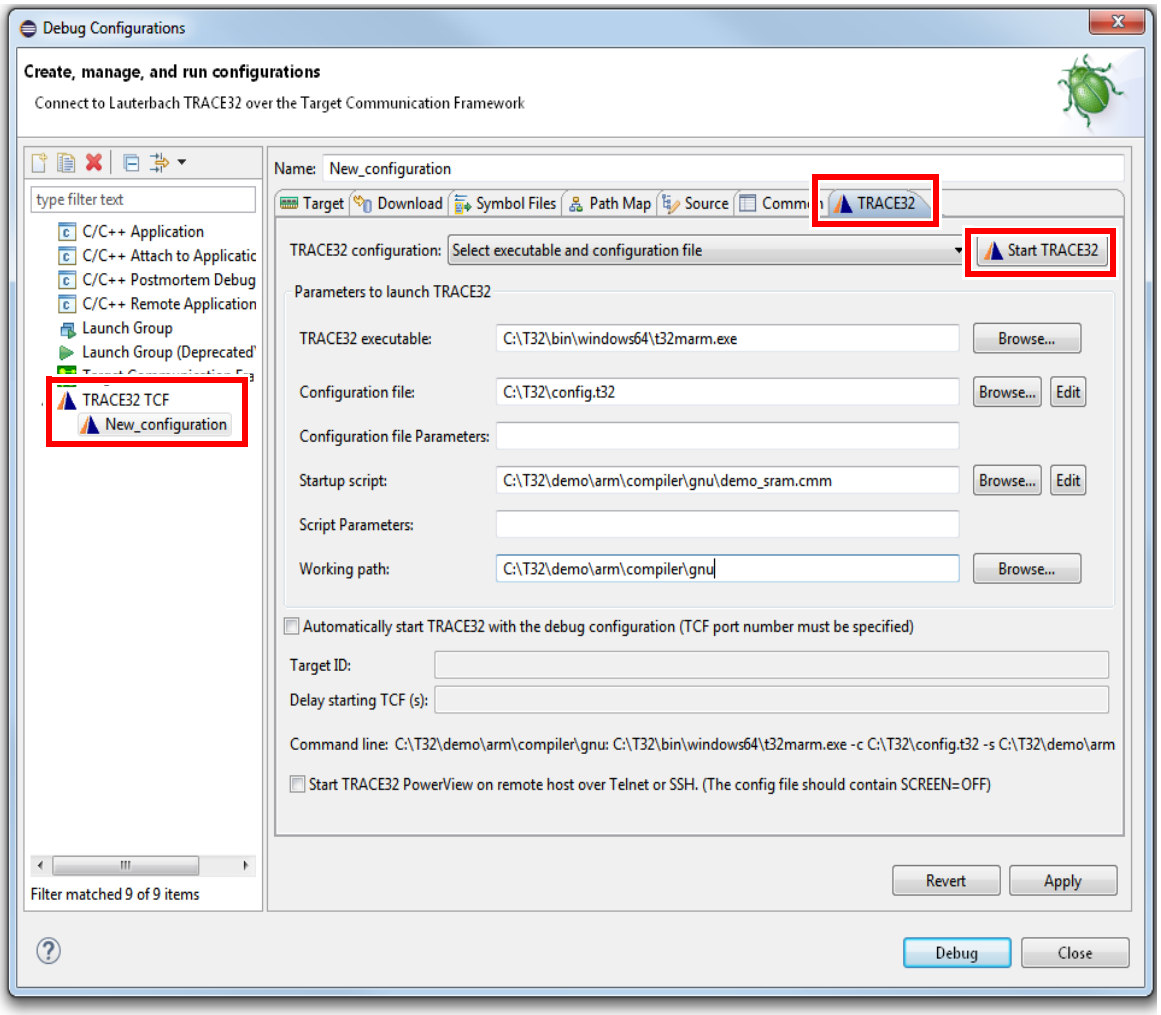



Please refer to the **"T32Start"** (app_t32start.pdf) manual for more information about the T32Start utility.

## Start TRACE32

**To start TRACE32 from within Eclipse:**

1. Choose **Run** menu > **Debug Configurations**.
2. In the left window pane of the **Debug Configurations** window, click a configuration under the entry **TRACE32 TCF**.

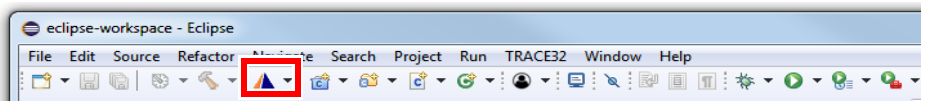



3. Click the **TRACE32** tab.
4. Click the **Start TRACE32** button.

   After starting, TRACE32 executes the PRACTICE start-up script (*.cmm) you have specified.
5. Have you specified a fixed TCF port number in the TRACE32 configuration file?
   - **No**: Please continue with section **TCF Discovery**.

- **Yes**: Please continue with section **Manual Debug Target Setup**.

**NOTE:** The TRACE32 TCF Eclipse plug-in also adds a new button to the tool bar with the Lauterbach logo to start TRACE32.

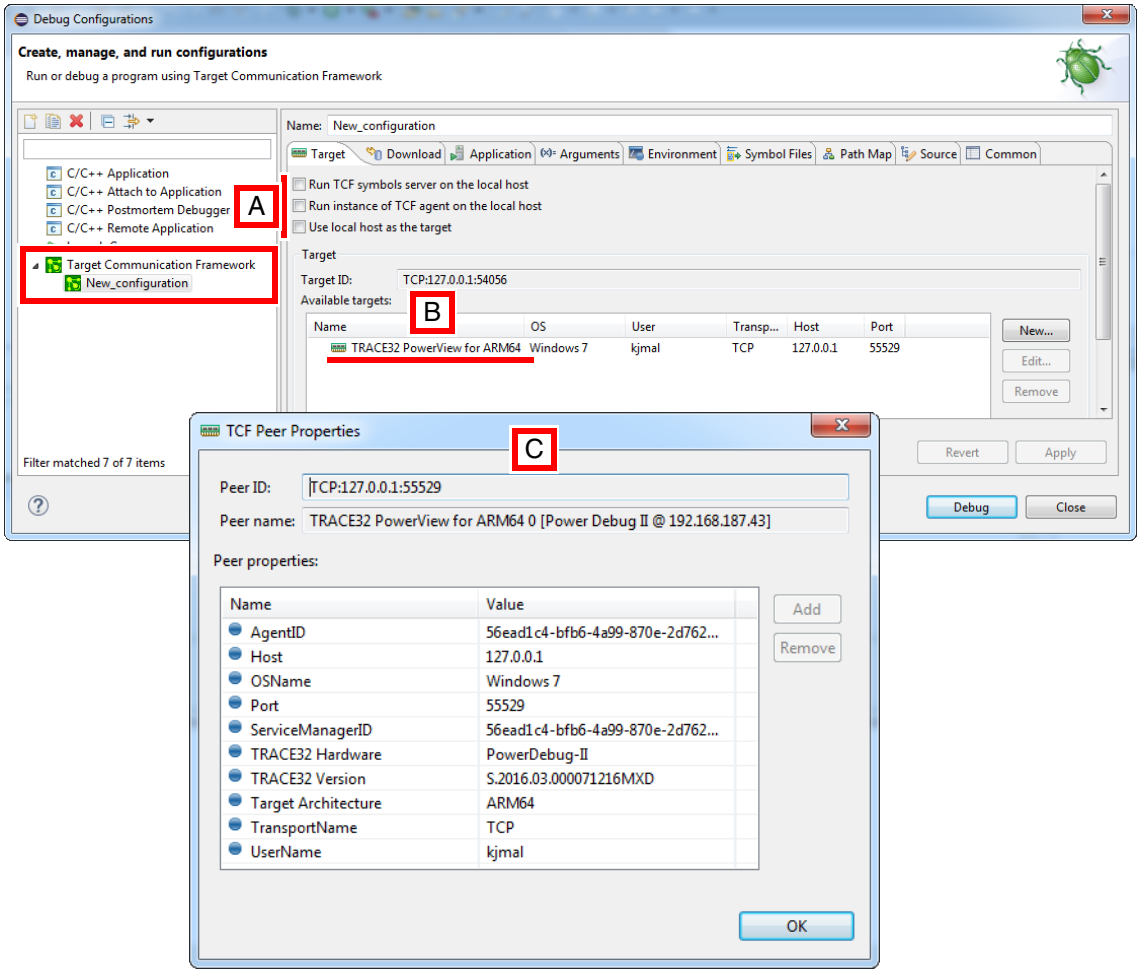



**If this button is not displayed in the tool bar, take these steps:**

- Choose **Windows** menu > **Perspective** > **Customize Perspective**.
- On the **Tool Bar Visibility** tab, select the check box **Lauterbach TRACE32**.

**To establish a debug connection using the TCF discovery:**

1. In the **Debug Configurations** dialog, click **Target Communication Framework**.




**A** See Step 2. below.

**B** If the TCF discovery has been enabled in TRACE32, then you should see the TRACE32 PowerView instance on the **Target** Tab under **Available Targets**.

**C** A double-click on the **Name** will show the properties of the TRACE32 PowerView instance including the target architecture, the TRACE32 software version and the used TRACE32 hardware.

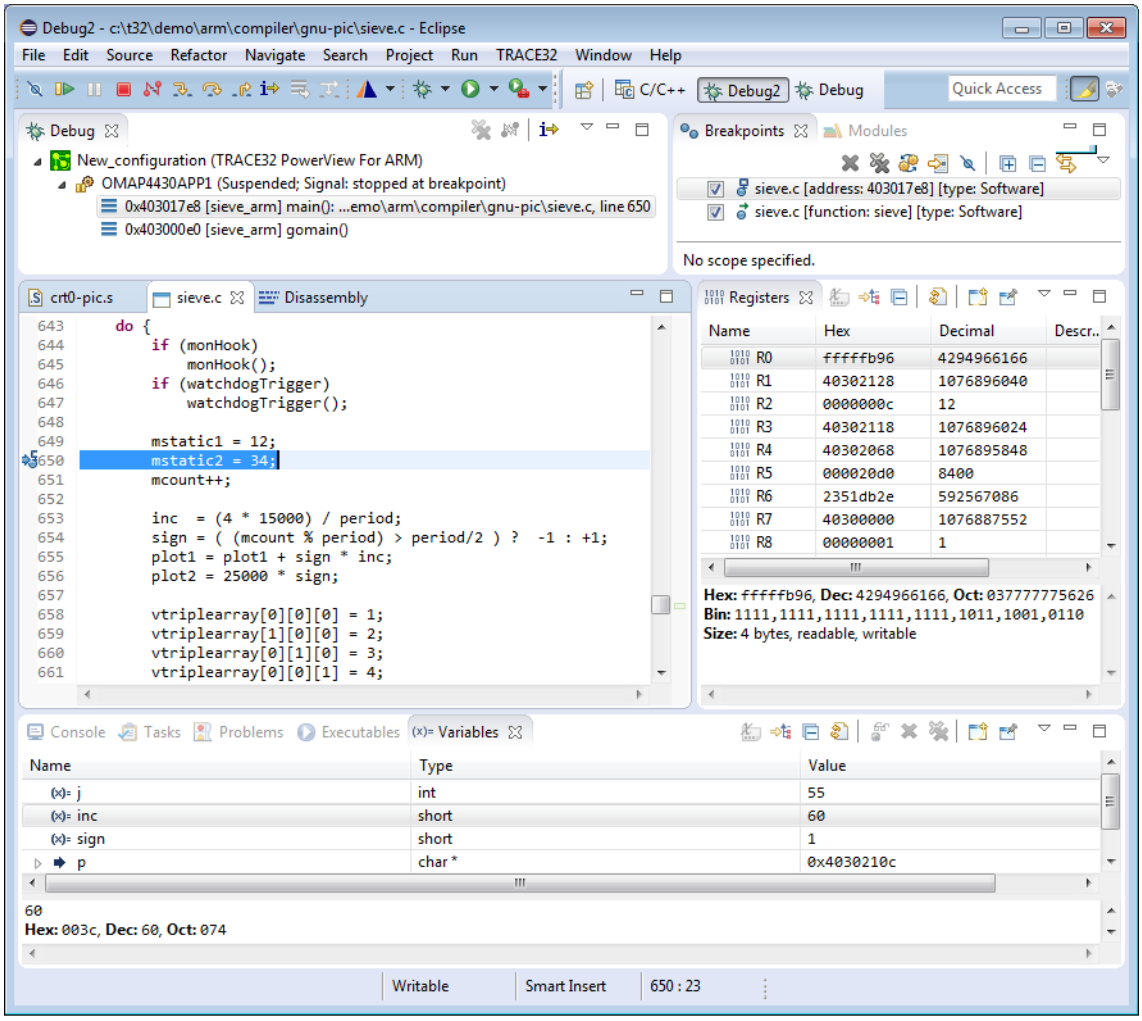
2. In case you are using the standard Target Communication Framework configuration, clear all three check boxes on the **Target** tab:

- **Run TCF symbols server on the local host**
- **Run instance of TCF agent on the local host**
- **Use local host as the target**

3. To establish a debug connection, select the TRACE32 instance under **Available Targets**, and then click the **Debug** button.

| | |
|---|---|
|  | **If the TCF discovery is enabled in TRACE32, the TCF port number will be automatically selected. This means that a new port number could be used each time a new TRACE32 PowerView instance is used.** |

4. Choose **Windows** menu > **Perspective** > **Other** > **Debug**.



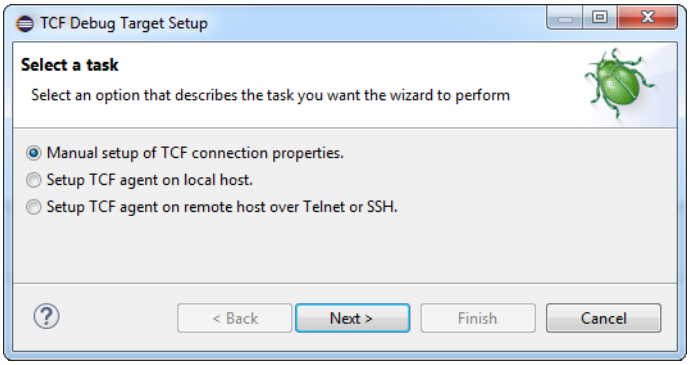
# Manual Debug Target Setup

If the TCF discovery has been disabled in TRACE32 by specifying a fixed TCF port number in the configuration file, then you need to create a new target setup.

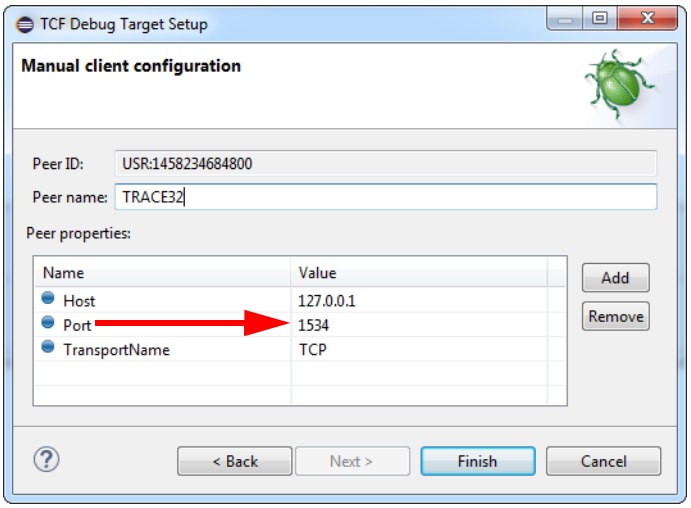
**To create a new target setup:**

1. Choose **Run** menu > **Debug Configurations** dialog > **Target Communication Framework** to open the **Debug Configuration** window.
2. Click the **New** button on the **Target** tab.

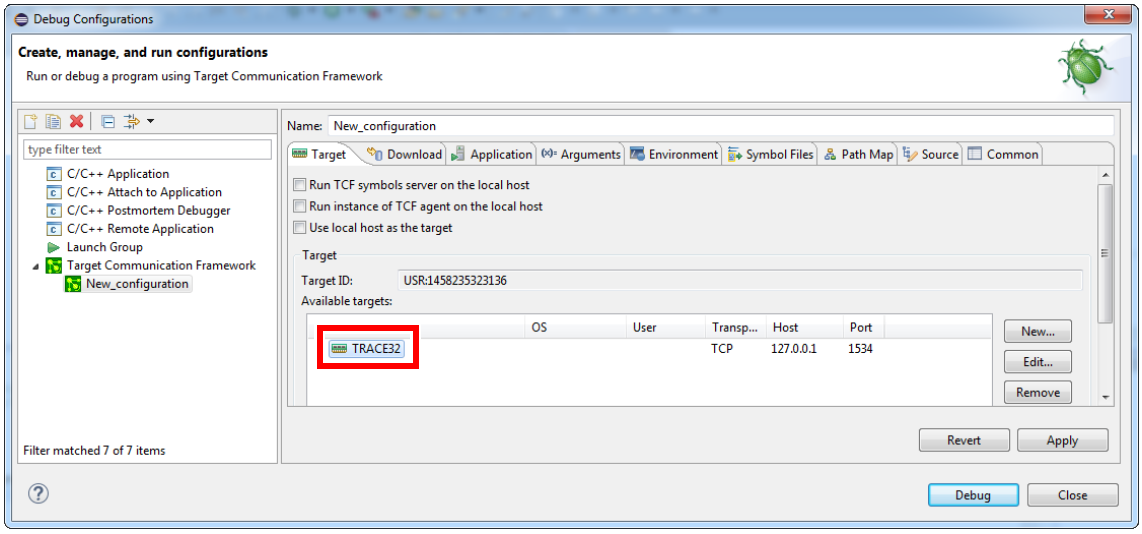
   The **TCF Debug Target Setup** dialog opens.




3. Select **Manual setup of TCF connection properties**, and then click **Next**.
4. In the **Peer name** field, enter any name e.g. “TRACE32”
5. For the **Port** property, set the port number used by TRACE32, e.g. we use here the default port number 1534, then press **Finish**.

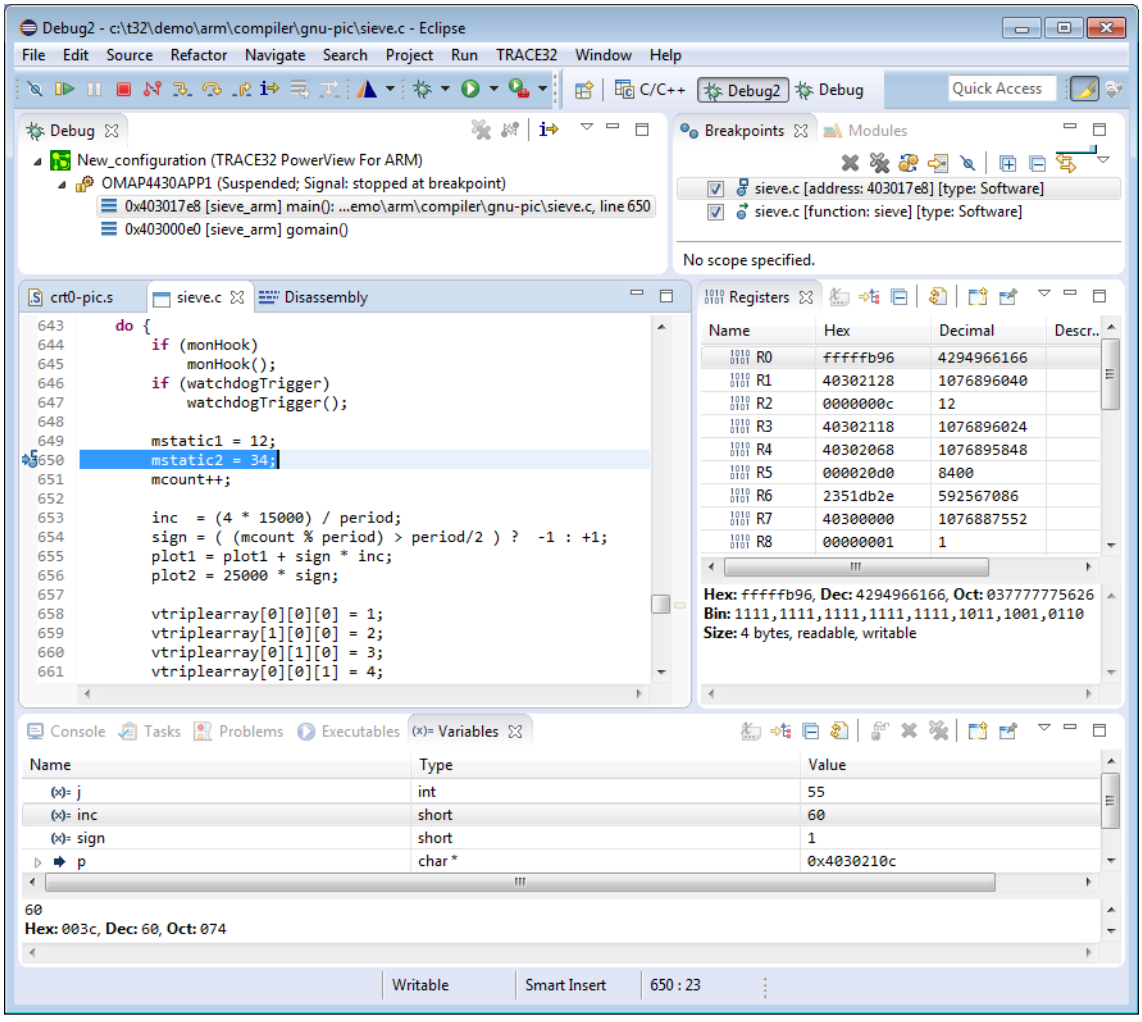



A new entry with the selected name will then appear on the **Target** tab under **Available Targets**, see screenshot below.




6. To establish a debug connection, select the TRACE32 instance under **Available Targets**, and then click the **Debug** button.

7. Choose **Windows** menu > **Perspective** > **Other** > **Debug**.

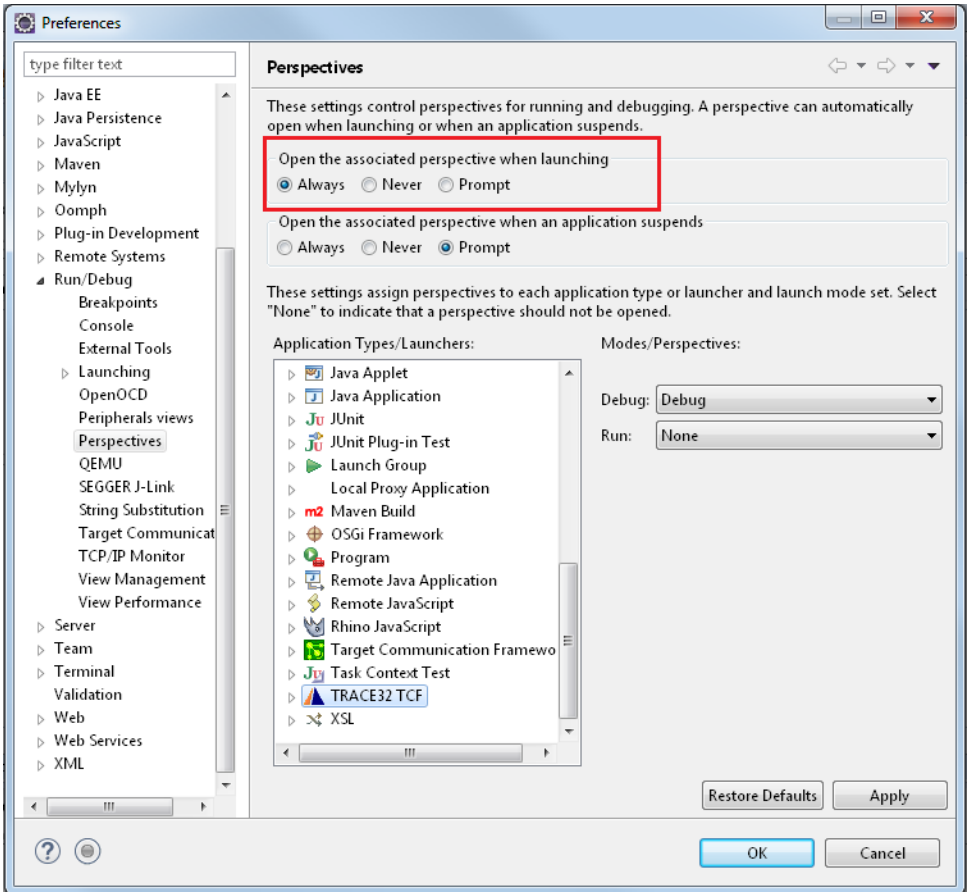


# Open Debug Perspective Automatically

**To activate the Debug perspective when a program is launched, do the following:**

1. Choose **Windows** menu > **Preferences**.
2. In the left window pane, click **Run/Debug** > **Perspectives**.

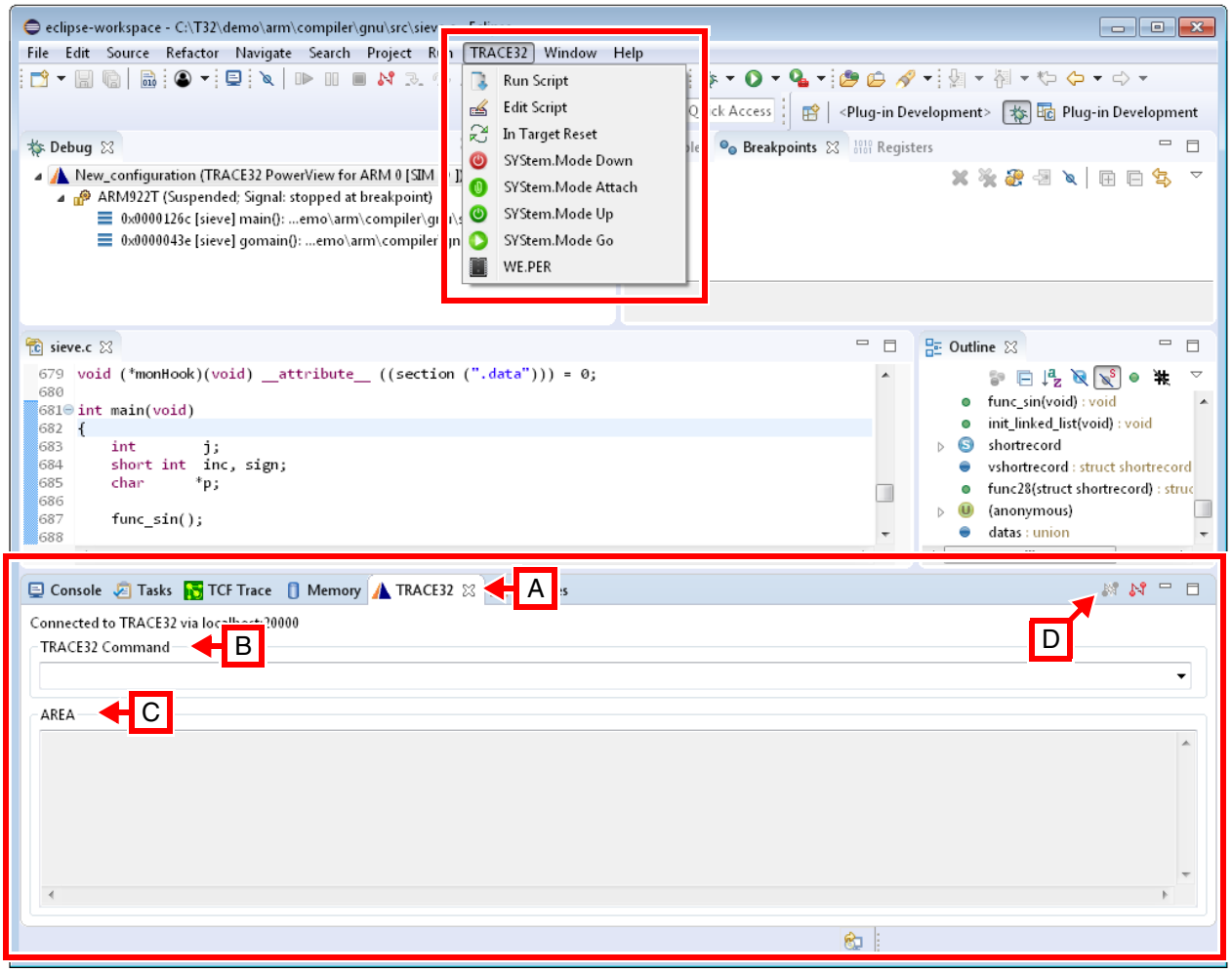
- Under **Open the associated perspective when launching**, select the **Always** option.
  This will cause the perspective associated with a program to become active whenever it is launched.



# TRACE32 View

In addition to the TCF communication, it is possible to control TRACE32 PowerView via a second UDP/IP channel using the Remote API. The TRACE32 View [**A**] can be used for this purpose. To open this view go to **Window** > **Show View** > **Others** > **Debug** > **TRACE32**. The command line [**B**] can be used to execute TRACE32 and PRACTICE commands similar to the TRACE32 command line. TRACE32 PowerView messages are then printed in the AREA field [**C**]. A **TRACE32** menu is additionally added to the Eclipse menu bar and contains shortcuts for special TRACE32 commands.

If TRACE32 is started from the TRACE32 Tab with an API port specified and the TRACE32 View is open, it will connect automatically. You can also connect to TRACE32 using the **Connect** button [**D**].



# Troubleshooting

## TRACE32

### TCF=(illegal command)

If you get this error message when starting TRACE32, then your TRACE32 version is too old and does not support TCF. You should use a TRACE32 version from February 2016 or newer.

- To check your TRACE32 version, choose **Help** menu > **About TRACE32**.

## Eclipse

### No TRACE32 PowerView instance under “Available Targets”

Please make sure that TRACE32 PowerView has been started as TCF agent with enabled TCF discovery.

1. To check this, select the TRACE32 **Help** menu > **About TRACE32**.

   Under **Environment**, you can see the used configuration file.

2. Click **edit** and check if the configuration file contains the TCF block.

If the TCF discovery has been disabled by using a fixed port number in the configuration file, the target setup needs to be done manually. Moreover, you can use the PRACTICE function **TCF.PORT()** in TRACE32 to print the used port number (requires TRACE32 build 71550 or newer):

```
PRINT TCF.PORT()
```

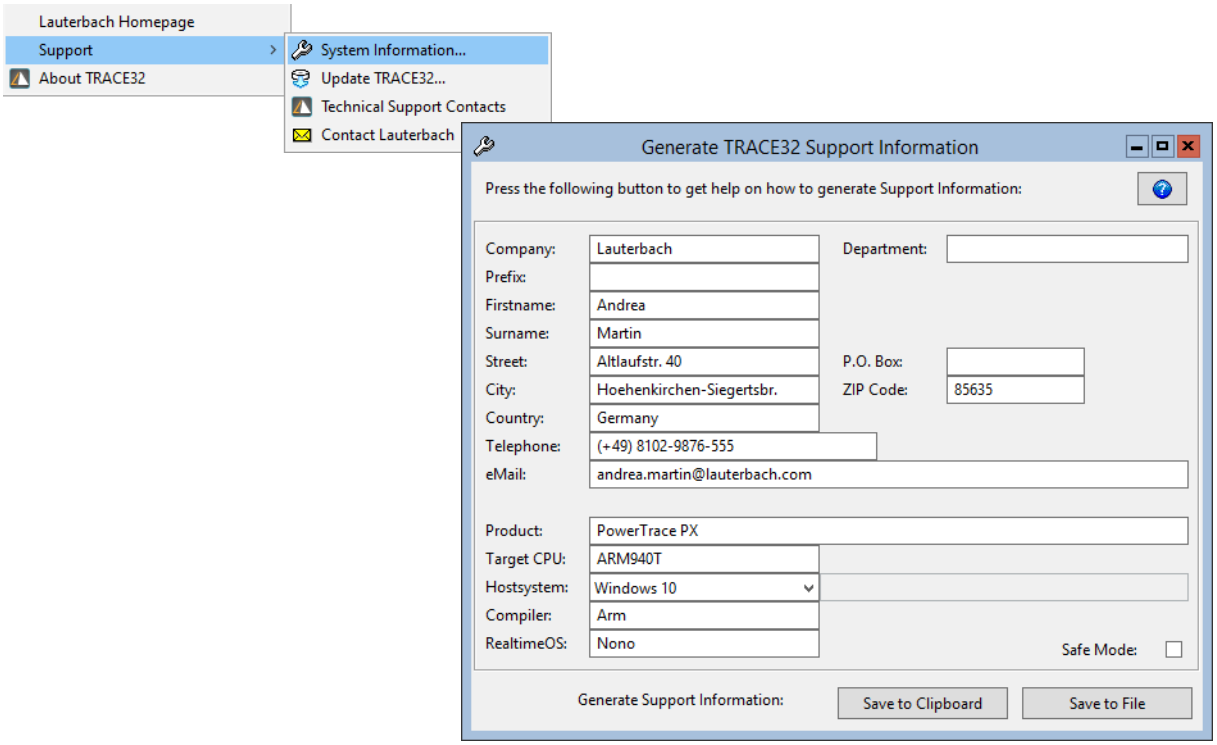
### Cannot locate peer TCP:<ip>:<port>

Please check that TRACE32 PowerView has been started as TCF agent and that you are using the correct port number in Eclipse.

## Export the TRACE32 System Information

Be sure to include detailed system information about your TRACE32 configuration.

1. To generate a system information report, choose **TRACE32** > **Help** > **Support** > **Systeminfo**.




| | |
|---|---|
| **NOTE:** | Please help to speed up processing of your support request. By filling out the system information form completely and with correct data, you minimize the number of additional questions and clarification request e-mails we need to resolve your problem. |

2. Preferred: click **Save to File**, and send the system information as an attachment to your e-mail.
3. Click **Save to Clipboard**, and then paste the system information into your e-mail.

## Export the Eclipse Error Log

Please include the full Eclipse **Error Log** as a file in your support request:

1. Choose **Window** menu > **Show View** > **Error Log** to open the **Error Log** view in Eclipse.
2. On the **Error Log** view tool bar, click the **Export** icon.
3. **Save** the log as a file.
4. Attach this file to your support request.

## Export the Eclipse Configuration

Export the Eclipse configuration settings in text form to the clipboard. With this we can check your Eclipse configuration for any missing or outdated components.

1. Choose **Help** menu > **About Eclipse**.
2. Click the button **Installation Details**.
3. Click the **Configuration** tab.
4. Click **Copy to Clipboard**.
5. Paste the clipboard content into your support mail to Lauterbach.

## SYStem.TCFconfig — TCF-specific setups

The **SYStem.TCFconfig** command group is used to define TCF-specific setups for debugging.

**See also**

- SYStem.state

## SYStem.TCFconfig.TASKCONTEXT — Enable/disable task contexts

Format: **SYStem.TCFconfig.TASKCONTEXT** [**ON** | **OFF**]

Default: ON.

**ON** — Operating system tasks are displayed in the Eclipse debug view as child contexts.

**OFF** — Operating system tasks are not displayed in the Eclipse debug view as contexts. Only the name of the current task is displayed for information.