Integration for CodeBlocks

Integration for CodeBlocks

Version 18-Nov-2020

This document describes installation and usage of the TRACE32 Code::Blocks integration.

Overview

The TRACE32 Code::Blocks integration is a plug-in allowing the Code::Blocks user to debug his applications on a real target using the TRACE32 debugger or a simulated target using the TRACE32 Instruction Set Simulator.

NOTE:	This integration uses internally the **TRACE32 Remote API**. The Remote API has **restrictions** if TRACE32 runs in demo mode. Please see there for further details.

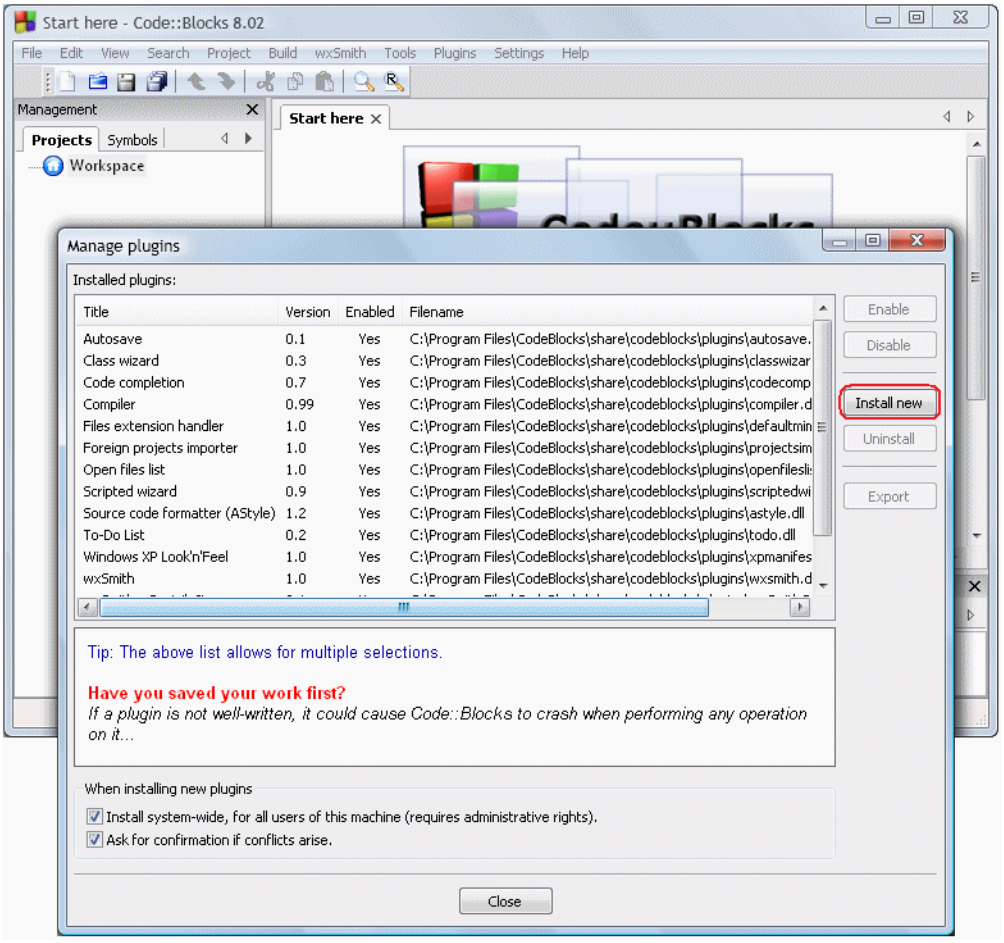
Supported Code::Blocks versions

The integration plug-in supports following Code::Blocks versions:

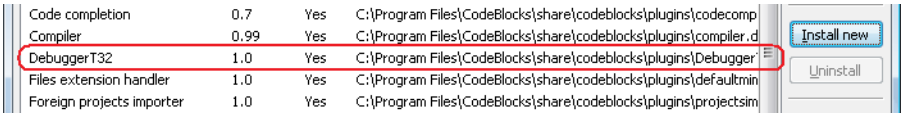
Version	Date
8.02	28 Feb 2008
10.05	30 May 2010
12.11	04 Jul 2013
13.12	06 May 2014

Plug-in Installation

To install the plug-in, select “Plugins->Manage Plugins” from Code::Blocks menu. Make sure that there is no other debugger plug-in installed in Code::Blocks. To uninstall a debugger plug-in select it from the list and press “Uninstall”. To install TRACE32 integration plug-in select “Install new” and navigate to the `debuggert32.cbplugin` file.



After successful installation DebuggerT32 plug-in appears in the list:



Plug-in and TRACE32 Configuration

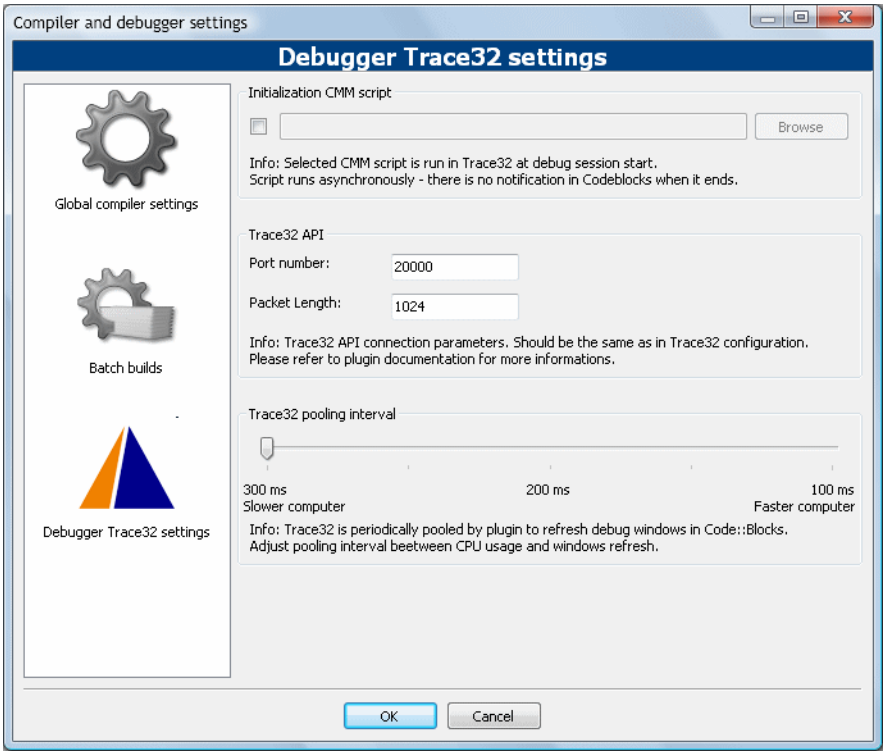
The plug-in and TRACE32 communication is based on TRACE32 API. To enable this communication API UDP port number and packet length need to be configured in both the plug-in and TRACE32.

Plug-in Configuration

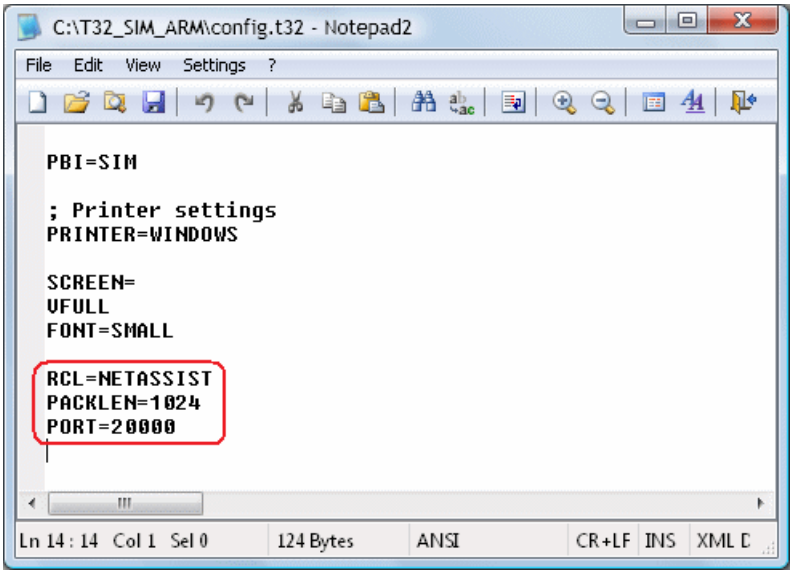
Select “Settings -> Compiler and Debugger -> Debugger TRACE32 Settings” from Code::Blocks menu.

Specify port number and packet length or leave default values.

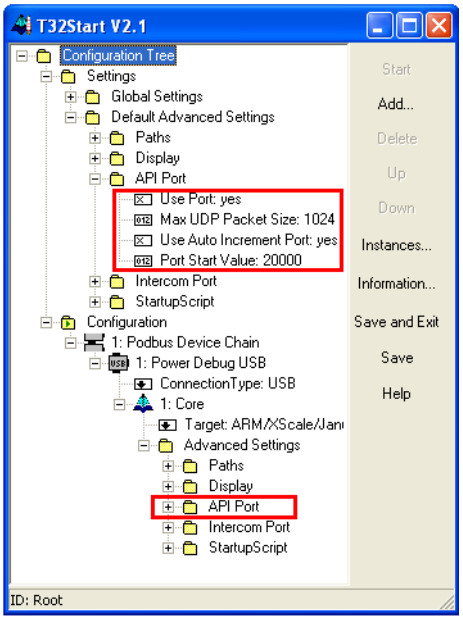
Additionally a start-up CMM script can be specified. The startup script is executed in TRACE32 each time debug session is started.



To enable the TRACE32 API communication with the plug-in, some changes need to be made in `config.t32`. This file can be find in your TRACE32 installation directory. Take care to keep one empty line before section with RCL, PORT and PACKLEN.



Changes described above can be also made by using T32Start utility provided with TRACE32:



Plug-in Menu

After the plug-in is installed, a new Code::Blocks menu with name “TRACE32“ appears.

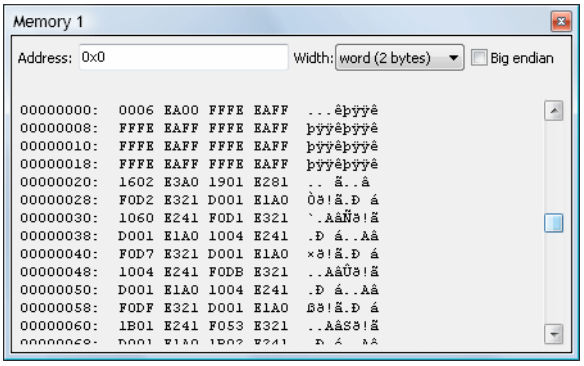
Debug	Start debug session by connecting to TRACE32 and executing startup script (if specified in plug-in settings. See **“Plug-in Configuration”** (int_codeblock.pdf) for details)
Stop Debugging	Stop debug session.
Download application	Download application to target. See **“Debugging Example Application”** (int_codeblock.pdf) for details.
Set next statement	Set program counter to current position in editor.
Show current statement	Open source code for current program counter position.
Show in TRACE32	Open `Data.LIST` window in TRACE32 with source code at editor´s current position.
Memory	Open memory window. See **“Memory window”** (int_codeblock.pdf) for details.
Watches	Open watch window. See **“Watches window”** (int_codeblock.pdf) for details.

Registers	Open register window. See **“Registers window”** (int_codeblock.pdf) for details.
New Breakpoint	Set new breakpoint at carret position.
Breakpoints list	Open breakpoints list. See **“Breakpoints List”** (int_codeblock.pdf) for details.
Step	Step into function call.
Step over	Step over function call.
Go Next	Step to next line. This command can be used to leave loops.
Go Return	Run application to the last instruction in the function.
Go Up	Return to caller function.
Go Till	Run application untill editor´s current position is reached.
Go	Run application.
Break	Break application.

The table below lists features supported in plug-in and their TRACE32 equivalents.

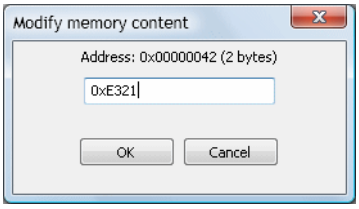
Set next statement	Register.Set PC *<address>*
Show current statement	Data.List
Show in TRACE32	Data.List *<address>*
Memory	Data.dump
Watches	Var.Watch
Registers	Register.view
New breakpoint	Break.Set
Breakpoints list	Break.List
Step	Step
Step over	Step.Over
Go Next	Go.Next
Go Return	Go.Return
Go Up	Go.Up
Go Till	Go *<address>*
Go	Go
Break	Break

Memory window



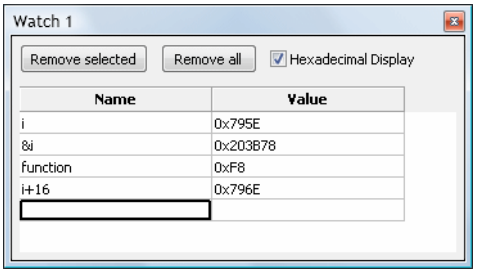
Address	Address expression.
Width	Width of displayed values. Can be either byte, word, long or quad.
Big endian	If activated, values are displayed in big-endian mode.

To modify memory content double click on appropriate value to open “Modify memory content” dialog:



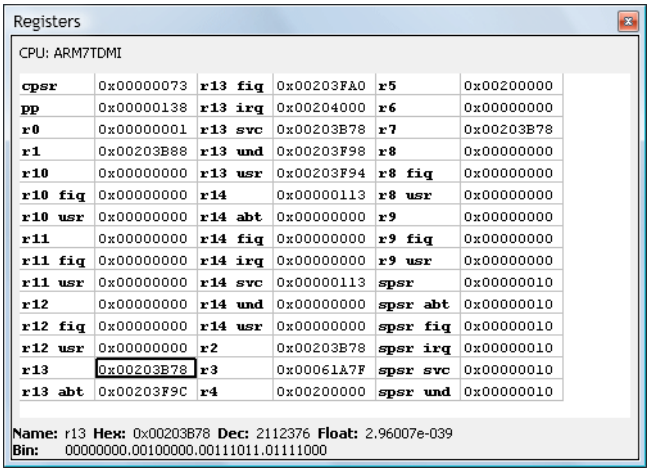
Provided value can be either in hexa-decimal, decimal or floating point format. If floating point value is provided, it is converted to either IEEE754 single or double precision format, depending on width of modified memory content. If width is less than 4 bytes, floating point value cannot be specified.

Watches window



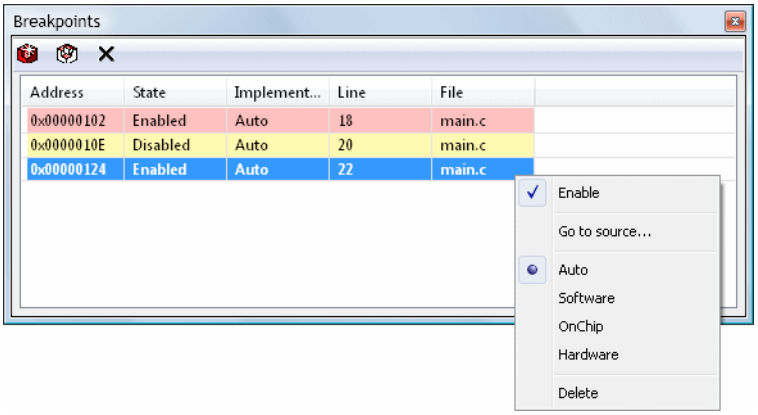
Registers window

To change register content, double-click on appropriate value and provide it in hexa-decimal, decimal or floating point value.



Breakpoints List

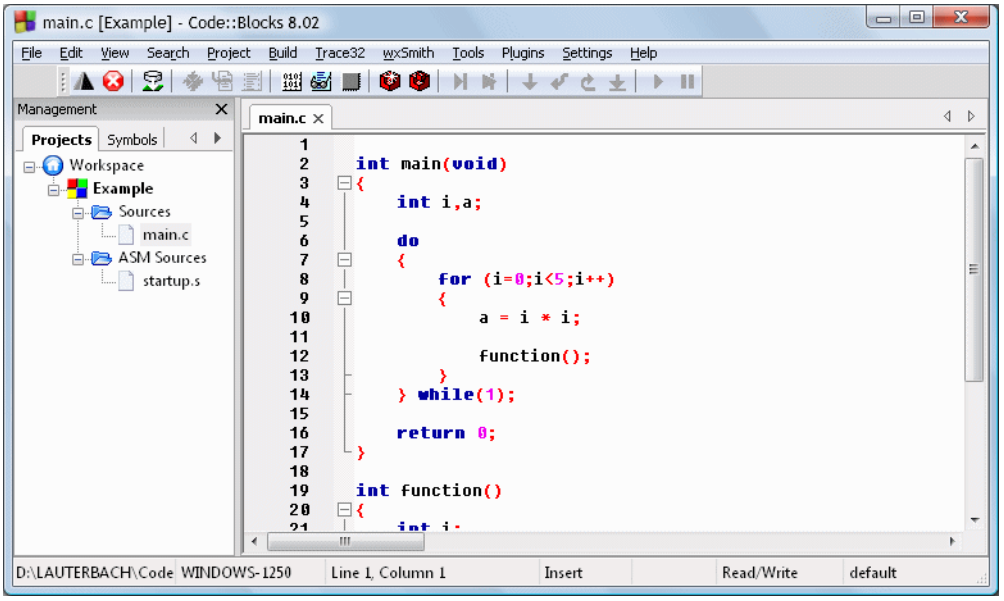
This window contains all breakpoints set in current project.



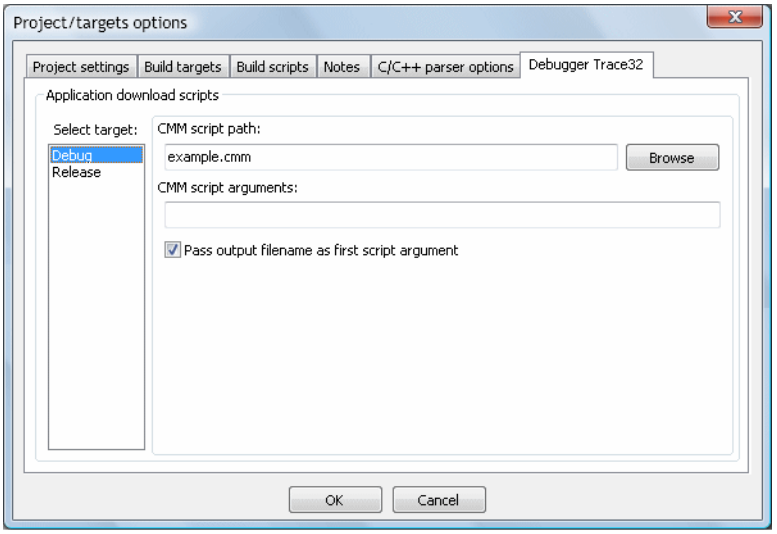
Debugging Example Application

Start Code::Blocks and the TRACE32 Instruction Set Simulator for ARM.

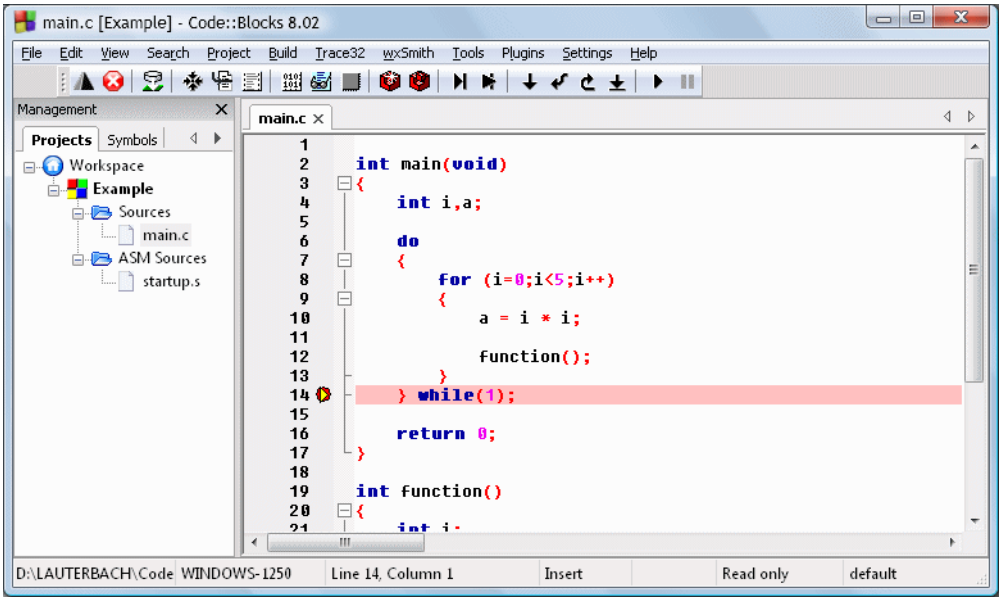
Open project “Example” that is provided with this plug-in and select Debug from plug-in menu to start debug session:



Select “Project -> Properties” from Code::Blocks menu. In tab “Debugger TRACE32” select target “Debug”. These settings specify application download script (in this case example.cmm, that can be found in project directory):



Select “Download application” in the plug-in menu. Download script is executed in TRACE32 and Code::Blocks displays current application state. From this point application can be debugged:



Application source code is also visible in TRACE32. If not, make sure that argument of y.spath command in example.cmm download script points to correct project directory. “Edit source” from context menu can be used to open source code location in Code::Blocks (SETUP.EDITEXT command with parameter ON need to be specified in download CMM script).