

Integration for CodeBlocks

Release 02.2025



TRACE32 Online Help	
TRACE32 Directory	
TRACE32 Index	
TRACE32 Documents	Þ
3rd-Party Tool Integrations	
Integration for CodeBlocks	1
Overview	3
Supported Code::Blocks versions	4
Plug-in Installation	5
Plug-in and TRACE32 Configuration	6
Plug-in Configuration	6
TRACE32 Configuration	7
Plug-in Menu and Windows	8
Plug-in Menu	8
Memory window	11
Watches window	12
Registers window	12
Breakpoints List	13
Debugging Example Application	14

Version 13-Feb-2025

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.

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

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:

Code completion	0.7	Yes	C:\Program Files\CodeBlocks\share\codeblocks\plugins\codecomp	
Compiler	0.99	Yes	C:\Program Files\CodeBlocks\share\codeblocks\plugins\compiler.d	Install new
DebuggerT32	1.0	Yes	C:\Program Files\CodeBlocks\share\codeblocks\plugins\Debugger	
Files extension handler	1.0	Yes	C:\Program Files\CodeBlocks\share\codeblocks\plugins\defaultmin	
Foreign projects importer	1.0	Yes	C:\Program Files\CodeBlocks\share\codeblocks\plugins\projectsim	

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.

Compiler and debugger setting	ngs	
	Debugger Trace32 settings	
Global compiler settings	Initialization CMM script Initialization CMM script Info: Selected CMM script is run in Trace32 at debug session start. Script runs asynchronously - there is no notification in Codeblocks who	Browse
Batch builds	Trace32 API Port number: 20000 Packet Length: 1024 Info: Trace32 API connection parameters. Should be the same as in T Please refer to plugin documentation for more informations.	race32 configuration.
Debugger Trace32 settings	Trace32 pooling interval 300 ms 200 ms Slower computer Info: Trace32 is periodically pooled by plugin to refresh debug window Adjust pooling interval beetween CPU usage and windows refresh.	100 ms Faster computer vs in Code::Blocks.
	ОК Сапсе	

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.

C:\T32_SIM_ARM\config.t32	- Notepad2				x
File Edit View Settings ?					
🌡 🔊 رم 🔜 🖾 🚰 🗋	la 🚨	🏦 🏭 🗐 💽	Q		<u>4</u> Þ
PBI=SIM ; Printer settings PRINTER=WINDOWS SCREEN= VFULL FONT=SMALL					
RCL=NETASSIST PACKLEN=1024 PORT=20000					Þ
Ln 14 : 14 Col 1 Sel 0 124	Bytes	ANSI	CR+LF	INS	XML C

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	F8
🐼 Stop Debugging	Alt-F8
Download application	Ctrl-F8
🔹 Set next statement	
🔄 Show current statement	
📰 Show In Trace32	
Memory	•
Watches	+
Registers	
🔯 New Breakpoint	F5
Breakpoints list	
N Step	Shift-F7
🖌 Step over	F7
🕹 Go Next	
🖋 Go Return	
🛃 Go Up	Shift-Ctrl-F7
🛨 Go Till	F4
▶ Go	Ctrl-F7
II Break	Alt-F7

Debug	Start debug session by connecting to TRACE32 and executing startup script (if specified in plug-in settings. See "Plug-in Configuration", page 6 for details)
Stop Debugging	Stop debug session.
Download application	Download application to target. See " Debugging Example Application ", page 14 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", page 11 for details.
Watches	Open watch window. See "Watches window", page 12 for details.

Registers	Open register window. See " Registers window ", page 12 for details.
New Breakpoint	Set new breakpoint at carret position.
Breakpoints list	Open breakpoints list. See "Breakpoints List", page 13 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 <i><address></address></i>		
Show current statement	Data.List		
Show in TRACE32	Data.List <i><address></address></i>		
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 <i><address></address></i>		
Go	Go		
Break	Break		

Memory window

Memory 1						X
Address: 0x0					Width: word (2 bytes)	💌 🔲 Big endian
00000000:	0006	EACO	FFFE	EAFF	êþÿÿê	~
00000010:	FFFE	EAFF	FFFE	EAFF	þýýéþýýé þýýéþýýé byvébyvé	
00000020:	1602 1602	ESAO E321	1901 D001	E281	pyyepyye ãâ Òalã Đ á	
00000030:	1060 D001	E241 E1A0	FOD1 1004	E321 E241	`.AâÑð!ã .ĐáAâ	
00000040:	F0D7 1004	E321 E241	DOO1 FODB	E1A0 E321	×ð!ã.Ð á AâÛð!ã	
00000050:	DOO1 FODF	E1A0 E321	1004 D001	E241 E1A0	.Ð áAâ Bð!ã.Ð á	
00000060:	1B01 D001	E241	F053 1P02	E321 8241	Aâsð!ã Þá Mô	+

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

Watch 1	×
Remove selected Remo	ve all 🛛 🗸 Hexadecimal Display
Name	Yalue
i	0×795E
8i	0×203B78
function	0×F8
i+16	0×796E

Registers window

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

Registers						
CPU: ARM7	TDMI					
cpsr	0x00000073	r13 fiq	0x00203FA0	r5	0x00200000	
pp	0x00000138	r13 irq	0x00204000	r6	0x00000000	
rO	0x00000001	r13 svc	0x00203B78	r7	0x00203B78	
r1	0x00203B88	r13 und	0x00203 F 98	r8	0x00000000	
r10	0x00000000	r13 usr	0x00203F94	r8 fiq	0x00000000	
r10 fiq	0x00000000	r14	0x00000113	r8 usr	0x00000000	
r10 usr	0x00000000	r14 abt	0x00000000	r9	0x00000000	
r11	0x00000000	r14 fiq	0x00000000	r9 fiq	0x00000000	
r11 fiq	0x00000000	r14 irq	0x00000000	r9 usr	0x00000000	
r11 usr	0x00000000	r14 svc	0x00000113	spsr	0x00000010	
r12	0x00000000	r14 und	0x00000000	spsr abt	0x00000010	
r12 fiq	0x00000000	r14 usr	0x00000000	spsr fiq	0x00000010	
r12 usr	0x00000000	r2	0x00203B78	spsr irq	0x00000010	
r13	0x00203B78	r 3	0x00061A7F	spsr svc	0x00000010	
r13 abt	0x00203 F 9C	r4	0x00200000	spsr und	0x00000010	
Name: r13 Bin: 0000	Hex: 0x002038	78 Dec: 21 .00111011.0	12376 Float: 2)1111000	.96007e-039		

This window contains all breakpoints set in current project.

Breakpoints						
🙆 🕲 🗙						
Address	State	Implement	Line	File		
0×00000102	Enabled	Auto	18	main.c		
0x0000010E	Disabled	Auto	20	main.c		
0x00000124	Enabled	Auto	22	main.c		
					<	Enable
						Go to source
					•	Auto
						Software
						OnChin
						Hardware
						Delete

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:

🖶 main.c [Example] - Code::Blocks 8.02	
Eile Edit <u>Vi</u> ew Sea <u>r</u> ch <u>P</u> roject <u>B</u> uild Irace32 <u>w</u> x5mith Iools Plugins <u>S</u> ettings <u>H</u> elp	
▲ 😒 梁 響 圖 🏙 🚳 🔳 🕲 🕲 片 🖌 🤆 子 🕨	
Management X main.c X	,
Projects Symbols 4 1	-
□- 🕢 Workspace 2 int main(void)	
🗄 🏪 Example 3 🔤 🤇	
E Sources 5	
main.c 6 do	
🗄 🔁 ASM Sources 7 🗇 🧹	
startup.s 8 For (i=0;i<5;i++)	
12 Function();	
	-
14 - > WNILE(1);	
16 return 0;	
17	
18	
$\frac{19}{28} \square ($	
21 int i	-
∢ ►	
D:\LAUTERBACH\Code WINDOWS-1250 Line 1, Column 1 Insert Read/Write default	

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):

Project/targets o	options					×			
Project settings	Build targets	Build scripts	Notes	C/C++ parser options	Debugger Trace32				
Application dov	vnload scripts –								
Select target: CMM script path:									
Debug	example.c	mm				Browse			
Reidase	CMM script (arguments:							
			ОК	Cancel					

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).

