

Tutorial

How to use Keil μ Vision with Spansion templates

Warranty and Disclaimer

The use of the deliverables (e.g. software, application examples, target boards, evaluation boards, starter kits, schematics, engineering samples of IC's etc.) is subject to the conditions of Spansion as set out in (i) the terms of the License Agreement and/or the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials.

Please note that the deliverables are intended for and must only be used for reference in an evaluation laboratory environment. The software deliverables are provided on an as-is basis without charge and are subject to alterations. It is the user's obligation to fully test the software in its environment and to ensure proper functionality, qualification and compliance with component specifications.

Regarding hardware deliverables, Spansion warrants that they will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer. Should a hardware deliverable turn out to be defect, Spansion's entire liability and the customer's exclusive remedy shall be, at Spansion's sole discretion, either return of the purchase price and the license fee, or replacement of the hardware deliverable or parts thereof, if the deliverable is returned to Spansion in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Spansion, or abuse or misapplication attributable to the customer or any other third party not relating to Spansion or to unauthorised decompiling and/or reverse engineering and/or disassembling.

Spansion does not warrant that the deliverables do not infringe any third party intellectual property right (IPR). In the event that the deliverables infringe a third party IPR it is the sole responsibility of the customer to obtain necessary licenses to continue the usage of the deliverable.

In the event the software deliverables include the use of open source components, the provisions of the governing open source license agreement shall apply with respect to such software deliverables.

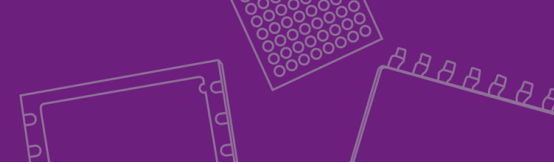
To the maximum extent permitted by applicable law Spansion disclaims all other warranties, whether express or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the deliverables are not designated. To the maximum extent permitted by applicable law, Spansion's liability is restricted to intention and gross negligence. Spansion is not liable for consequential damages.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect. The contents of this document are subject to change without a prior notice, thus contact Spansion about the latest one.

This board and its deliverables must only be used for test applications in an evaluation laboratory environment.

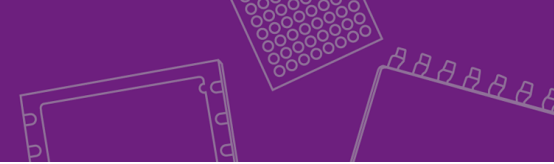


Debugging (Keil μ Vision)



Keil Development Tools (Software)

- MDK-ARM-EVALUATION **Free download**
 - 32K C/C++ Compiler, Assembler, μ Vision IDE/Debugger/Simulator
- MDK-ARM-B (Basic Edition)
 - 256K C/C++ Compiler, Assembler, μ Vision IDE/Debugger/Simulator
- MDK-ARM (Standard Edition)
 - C/C++ Compiler, Assembler, μ Vision IDE/Debugger/Sim., RTX Kernel
- ULINK-ME
 - USB-JTAG adapter (only 3.3 Volt support)
- ULINK2
 - USB-JTAG adapter
- ULINKpro
 - High-Speed debug and trace unit



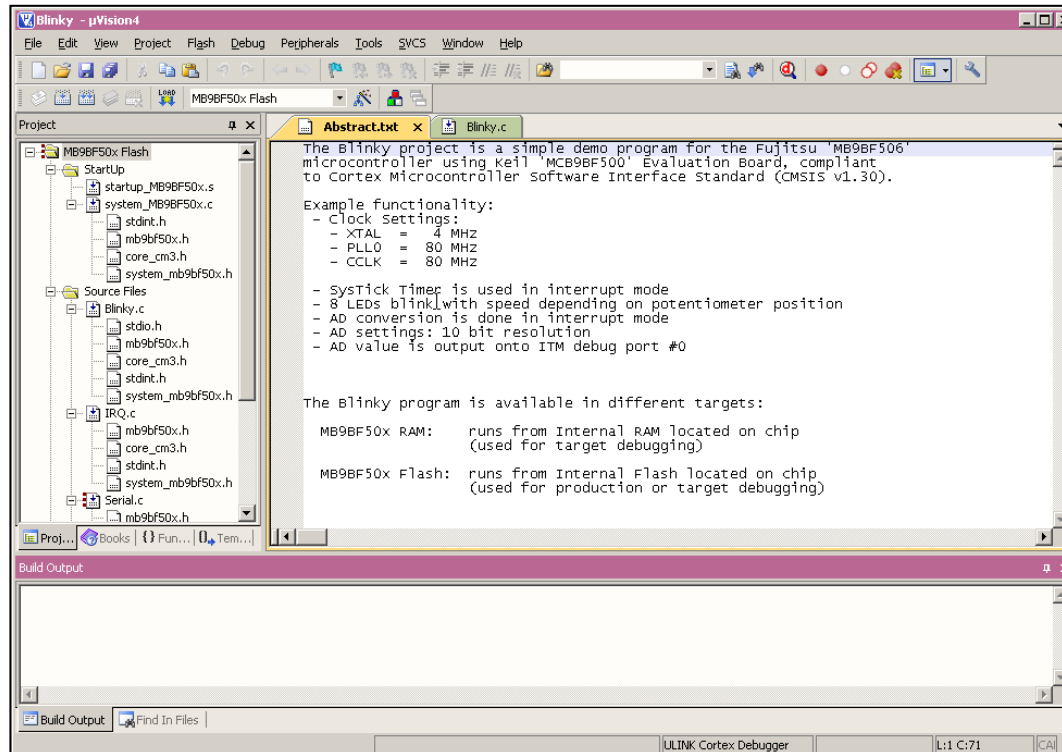
Installation

- Download latest version of μ Vision from KEIL Website
 - Evaluation Version
 - ◆ <https://www.keil.com/demo/eval/arm.htm>
 - ◆ Registration required
- Install μ Vision
- Install ULINK-ME
 - Special installation is not needed, because ULINK-ME acts as a USB Human Interface Device (HID) and thus needs no extra USB driver
- Install ULINK Pro (optional)
 - ULINK Pro needs an own dedicated USB driver located in:
<Installation Path>\Keil\ARM\ULINK
- Start μ Vision

Free download

Getting started

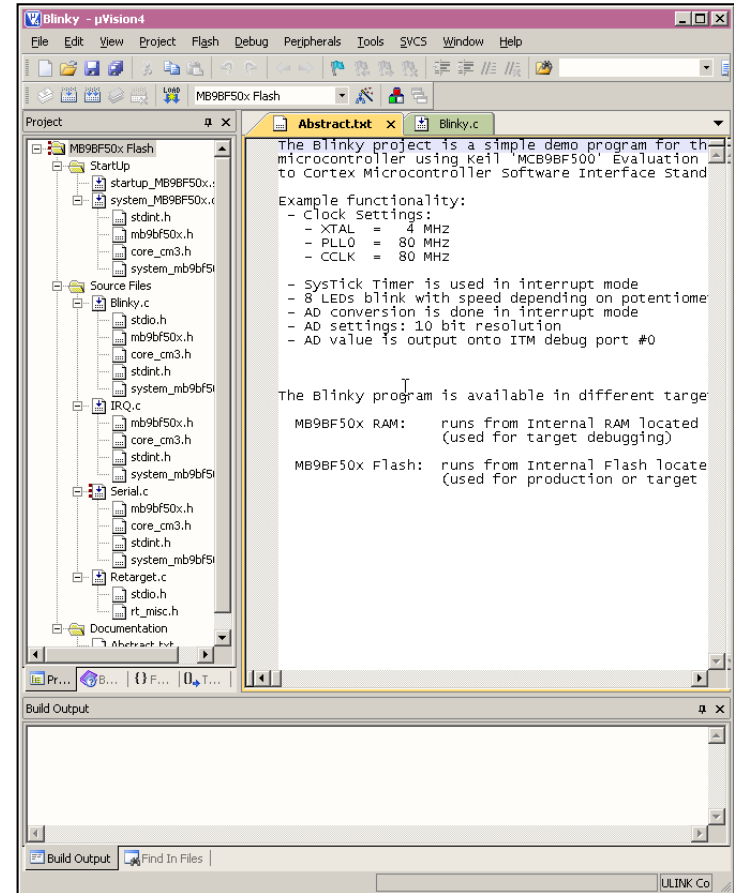
- Choose Menu: *Project* → *Open Project...*
 - Browse to: *<Installation Path>\Keil\ARM\Boards\Keil\MCB9BF500\Blinky*
 - Choose *Blinky.uvproj*



Main Window

■ KEIL μ Vision

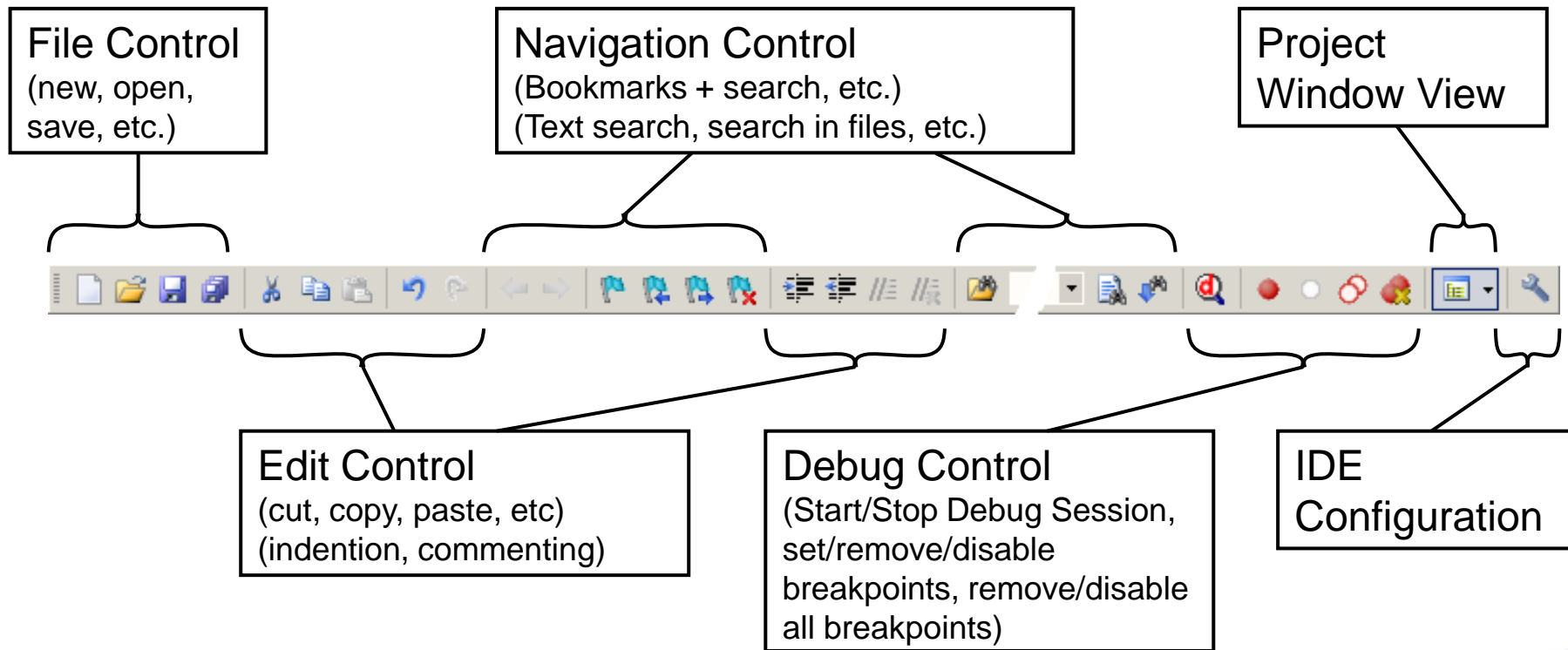
- Project window on left side of IDE window
 - ◆ Choose:
View \rightarrow *Project Window*
if hidden
- Source files on right side of IDE window as tabbed windows
- Output window on bottom side of IDE window



Menu Bars

- Menu Bar 1

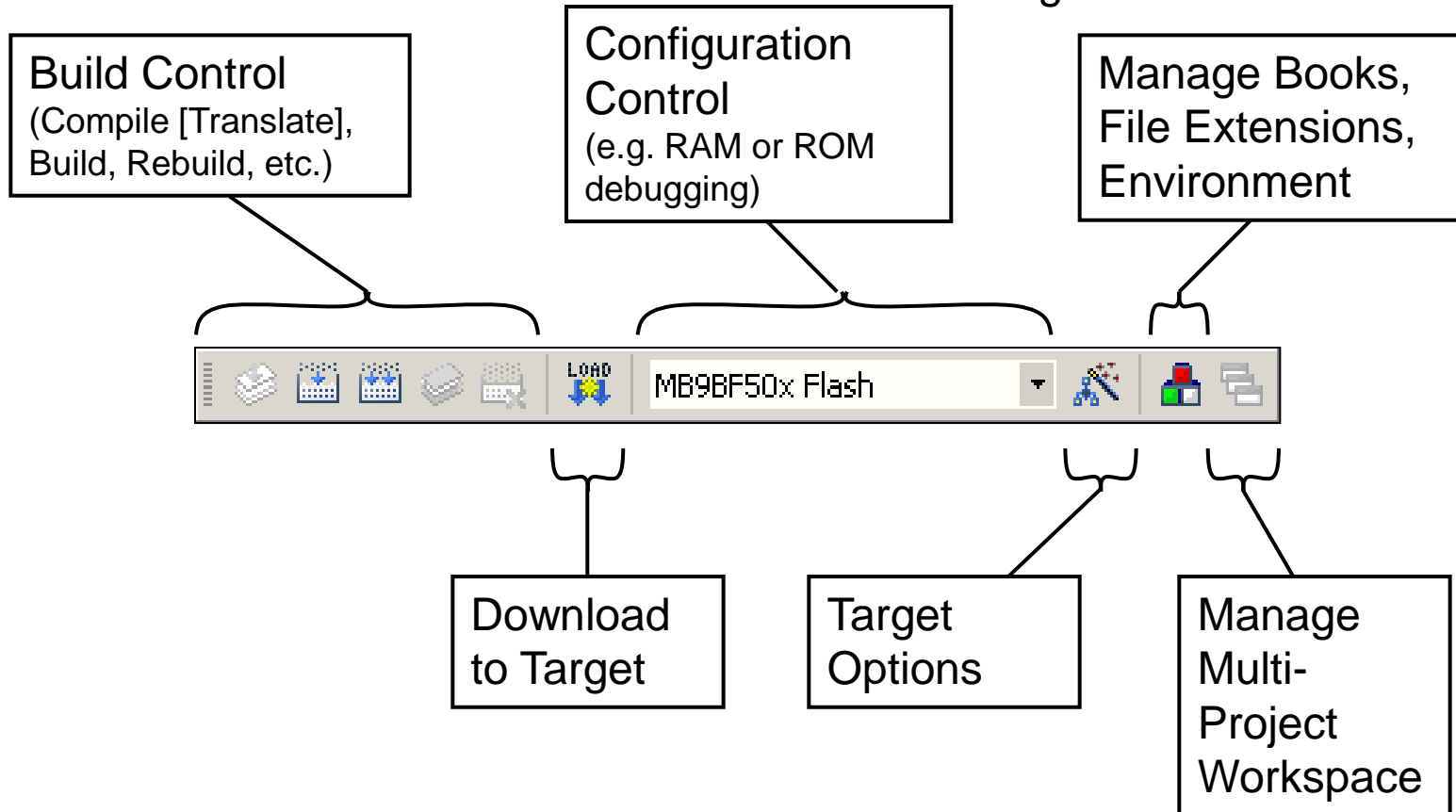
- Can be moved in bar window area or set floating



Menu Bars

- Menu Bar 2

- Can be moved in bar window area or set floating



μ Vision Project Window

Project Name

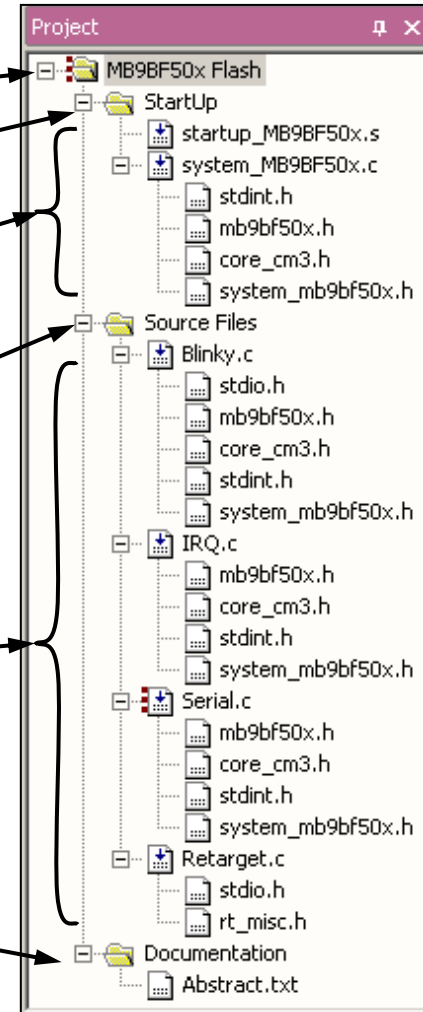
Startup Code Subfolder

Startup Code Source and Header Files

Main Project Code Subfolder


Main Project Code Source and Header Files

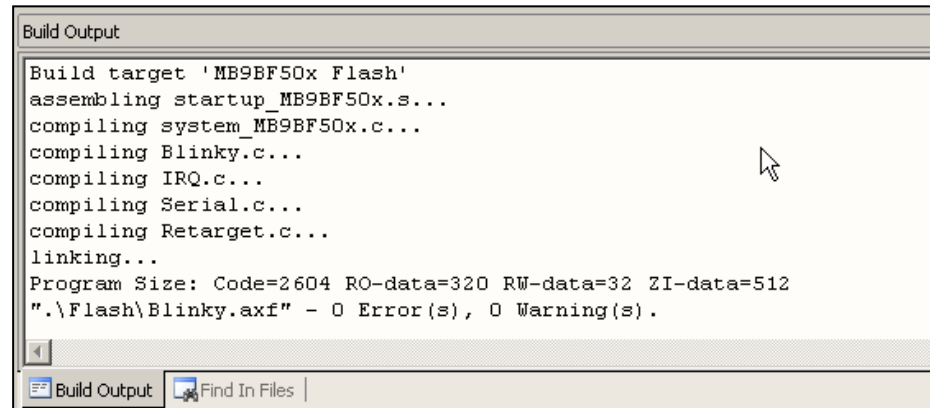
Project Description Subfolder and Abstract File



Making Project

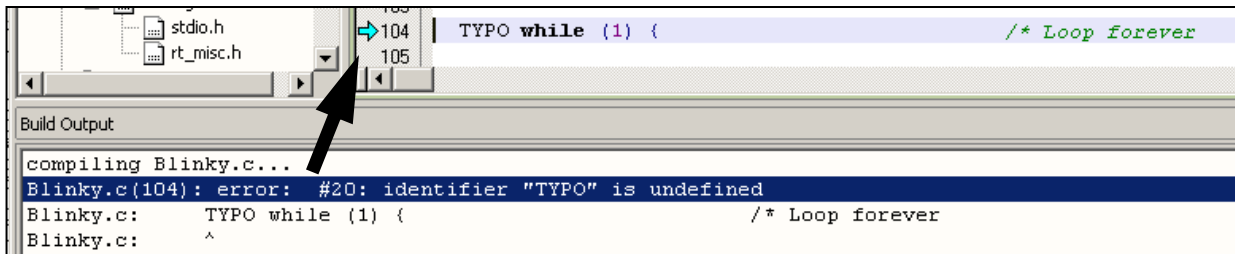
■ Making the Project

- Use Rebuild Icon () or *Project*→*Rebuild all target files*
- Check for no errors in Output window below





```
Build Output
Build target 'MB9BF50x Flash'
assembling startup_MB9BF50x.s...
compiling system_MB9BF50x.c...
compiling Blinky.c...
compiling IRQ.c...
compiling Serial.c...
compiling Retarget.c...
linking...
Program Size: Code=2604 RO-data=320 RW-data=32 ZI-data=512
".\Flash\Blinky.axf" - 0 Error(s), 0 Warning(s).
```

- Build errors are shown in Output window.
 - ◆ Can be double-clicked by showing the source line with a blue arrow



```
104 | TYPO while (1) { /* Loop forever
105 |
Build Output
compiling Blinky.c...
Blinky.c(104): error: #20: identifier "TYPO" is undefined
Blinky.c: TYPO while (1) { /* Loop forever
Blinky.c: ^
```

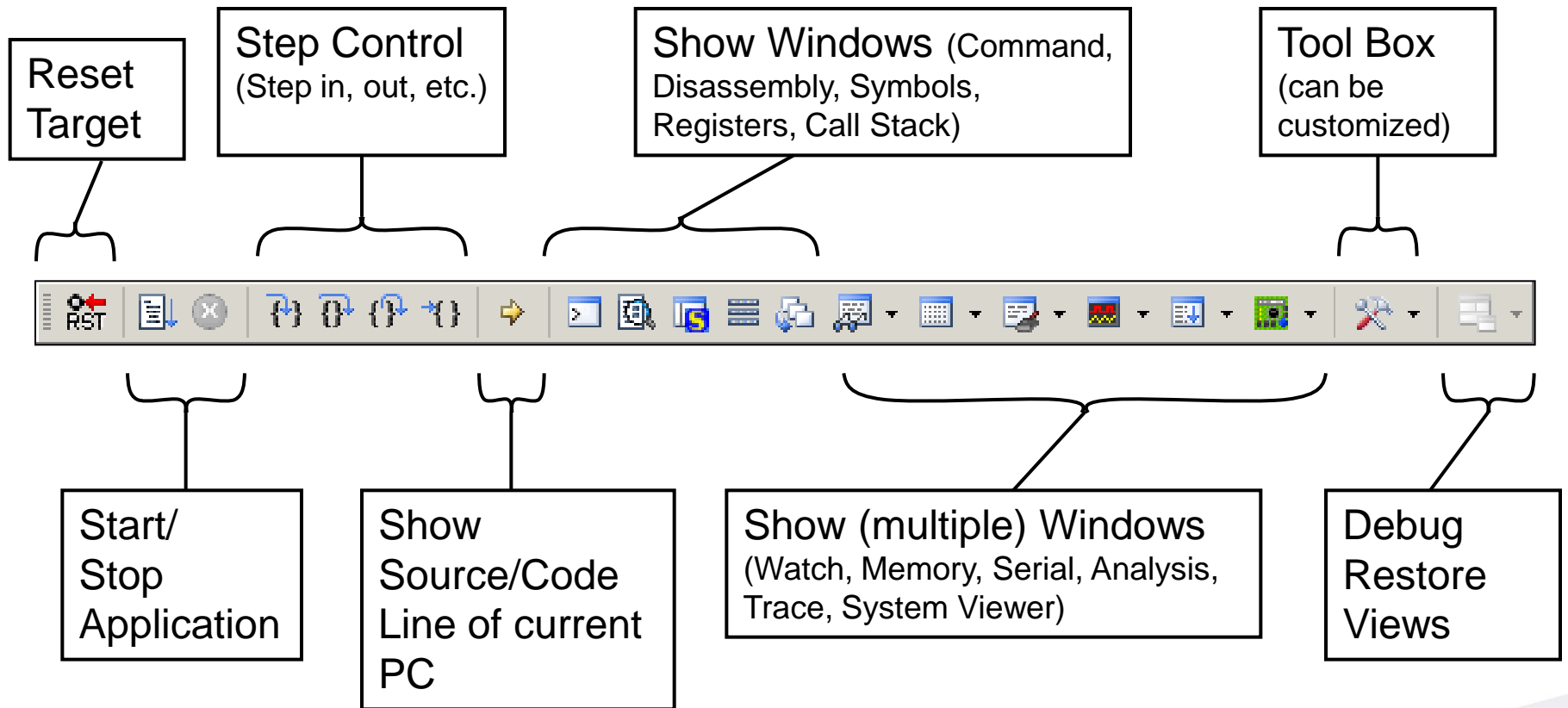
Start Debugging

- Download to target first, when MCU Flash does not contain the current application openend and built in the IDE
 - Use Download Icon () or Menu: *Flash*→*Download*
- Start Debug Session
 - Use Start/Stop Debug Icon () or Menu: *Debug*→*Start/Stop Debug Session*
- Ending Debug Session
 - Use same way as for starting debug session

Debugging (Keil μ Vision)

Debugging Icon Bar

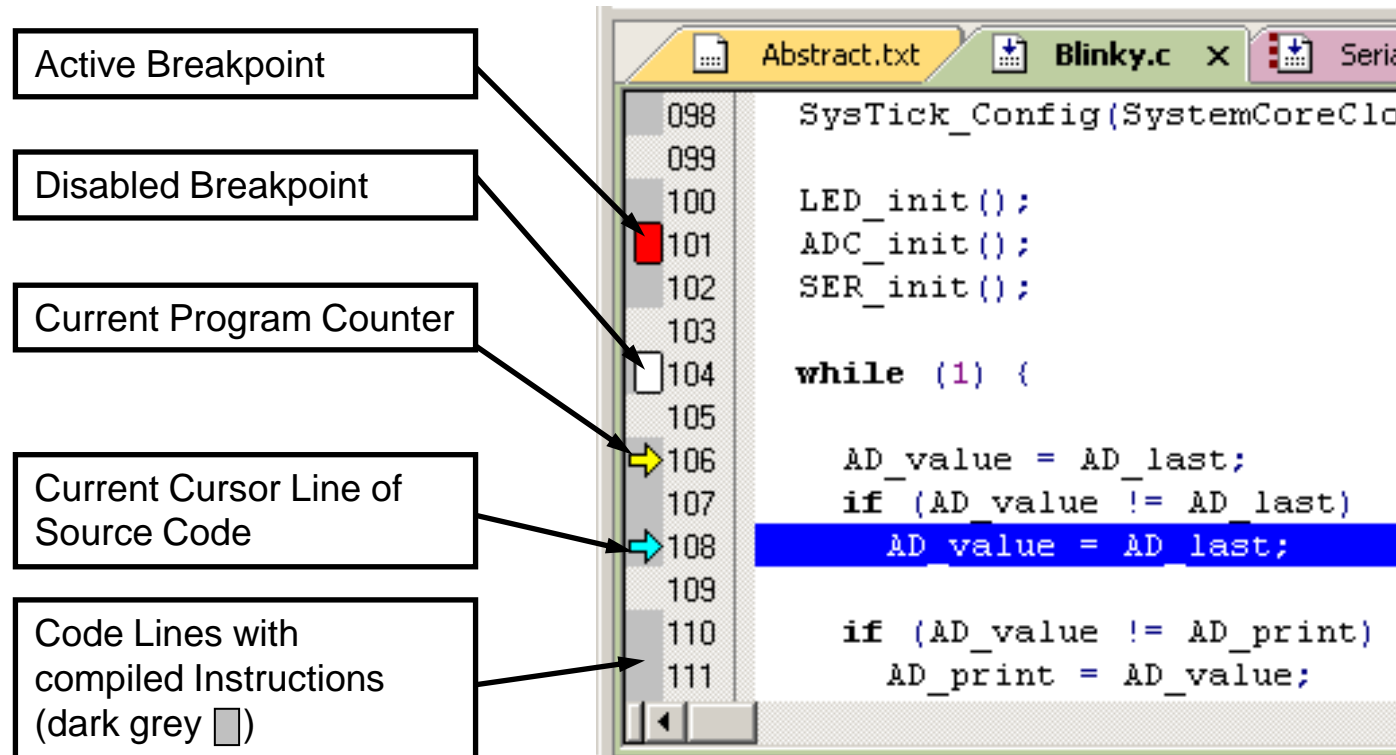
- During a Debug Session there will be visible a new icon bar



Debug Windows

- Source View

- The Source windows do not change contents but get additional information



Debug Windows

- Disassembly View
 - Mixed mode is selectable and deselectable

The screenshot shows the Disassembly window with the following code:

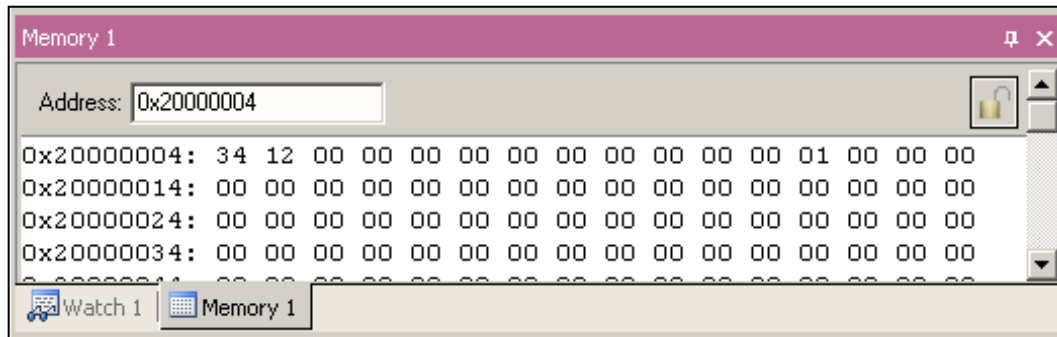
```
0x0000042A F7FFFA3 BL.W LED_i
101: ADC_init();
0x0000042E F7FFF67 BL.W ADC_i
102: SER_init();
103:
0x00000432 F000F8AE BL.W SER_i
104: while (1) {
105:
0x00000436 E015 B 0x0000
106: AD_value = AD_last;
0x00000438 4816 LDR r0, [p
0x0000043A 8804 LDRH r4, [r
107: if (AD_value != AD_last
```

Callouts from the left side of the image:

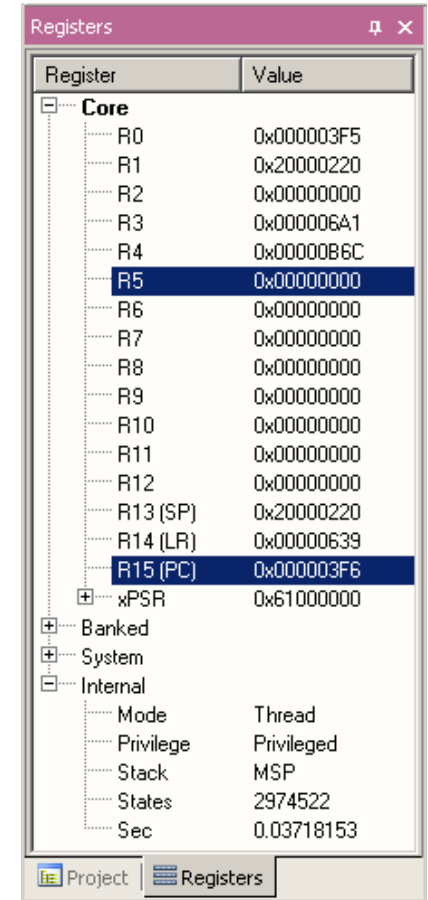
- Active Breakpoint: Points to the red square on the left of line 101.
- Disabled Breakpoint: Points to the white square on the left of line 103.
- Current Program Counter: Points to the yellow arrow on the left of line 106.
- Current Cursor Line of Code highlighted in yellow background (■): Points to the yellow background of line 106.

Debug Windows

- Memory Window
 - Up to 4 Memory windows can be displayed in tabs
 - Memory is updated during runtime
 - Memory window tabs are shared with Watch windows



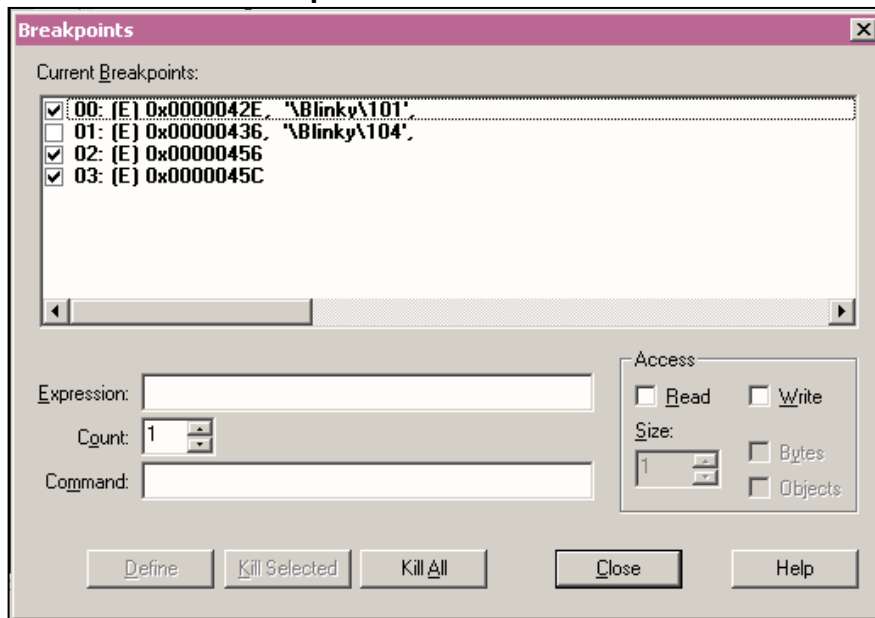
- Register View
 - Register view is a tab of the Project window
 - Changes are highlighted in dark blue text background
 - Register tree knots can be expanded



Debug Windows

■ Breakpoints

- Not a Workspace window, but a Dialog window
- Can be opened by menu: *Debug*→*Breakpoints...* or <Ctrl>-B
- Any changes can only be seen in Workspace after "Close"
- Conditional expressions can be entered here

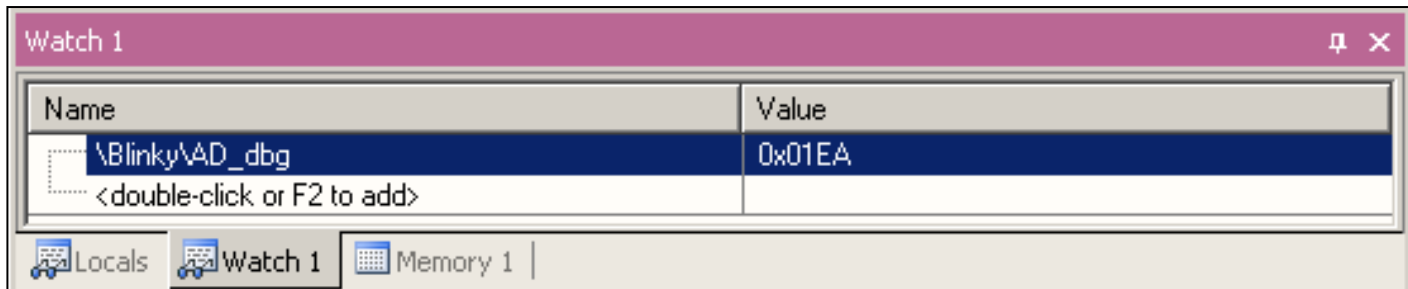


Debug Windows

- Variable Windows (1)

- Watch Windows

- ◆ Up to 2 Watch windows are sharing their tabs with e.g. Memory and Local views
 - ◆ Updated during runtime
 - ◆ Any changes are highlighted in dark blue text background color
 - ◆ Displayed values can be changed by user during break

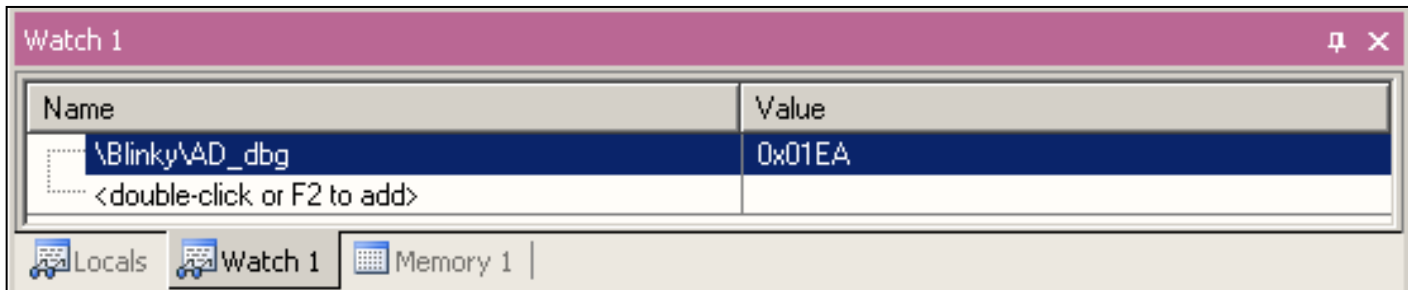


Debug Windows

- Variable Windows (1)

- Watch Windows

- ◆ Up to 2 Watch windows are sharing their tabs with e.g. Memory and Local views
 - ◆ Updated during runtime
 - ◆ Any changes are highlighted in dark blue text background color
 - ◆ Displayed values can be changed by user during break

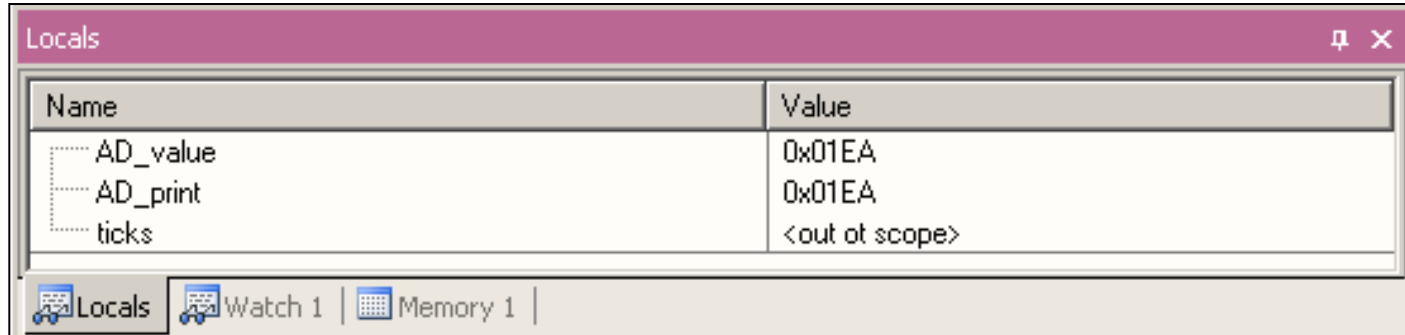


Debug Windows

- Variable Windows (2)

- Local View

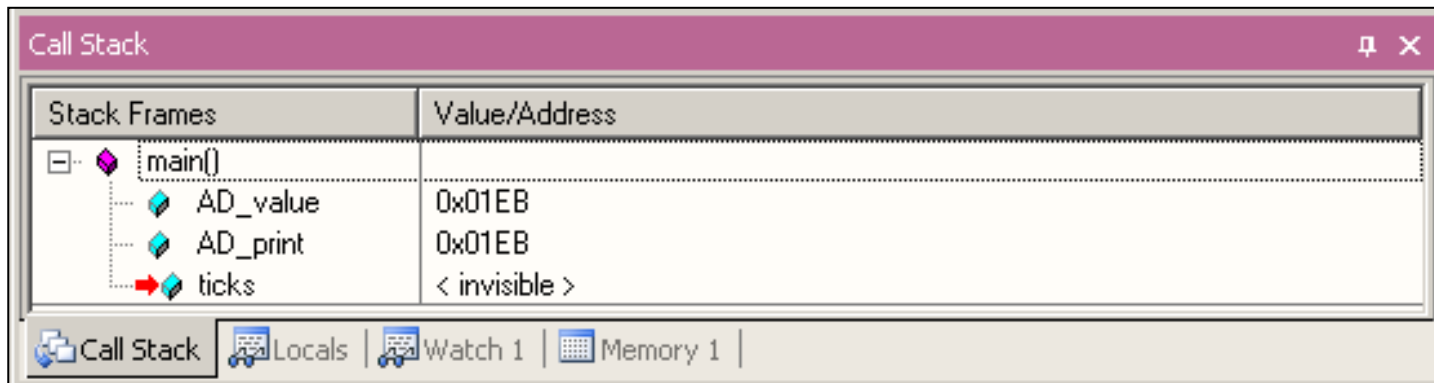
- The local view shares the tab with e.g. Memory and Watch windows
 - Any changes are highlighted in dark blue text background color
 - Displayed values can be changed by user during break



Debug Windows

■ Callstack

- The Call Stack view shares its tab with the Variable and Memory window
- It shows the current called function and their local environment
- All displayed Call Stack values are not modifiable



Debug Windows

■ Symbol Window

- The Symbol window shows all symbols of the user's project
- It is sorted in tree form by dependence
 - ◆ E.g.
Project → Module →
Function → Symbol
- Peripheral registers can also be displayed
- Simulator VTREG can be displayed
- Mouse-over a symbol shows its contents as a tool tip

◆ E.g.: `AD_dbg = 0x01EB`

The screenshot shows the Symbol Window in Keil µVision. The window title is "Symbols" and it has a search mask field containing "*" and a "Case Sensitive" checkbox. The main area displays a tree view of the project's symbol table. The tree is expanded to show the "Blinky" application, which contains a "Runtime Library" folder and a "Blinky" module. The "Blinky" module contains several symbols, including variables like "AD_dbg", "led_mask", "text", "num", "i", "value", and "ticks", and functions like "ADC_init", "IRQn", "LED_init", "LED_Off", "LED_On", "LED_Out", and "main". The table below the tree lists these symbols with their names, addresses, and types.

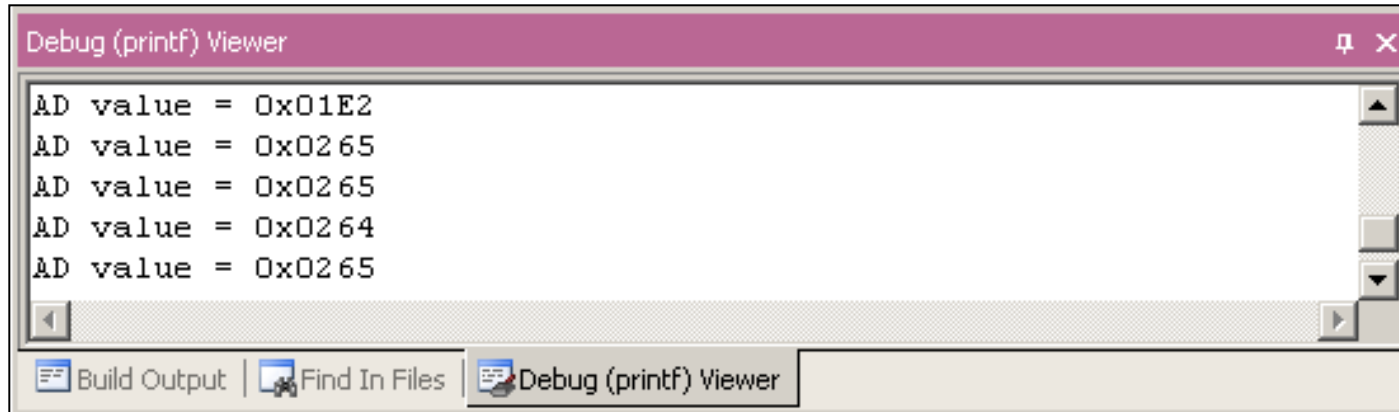
Name	Address	Type
Simulator VTREG		
Peripheral Regist...		
Blinky		Application
Runtime Library		
Blinky		Module
AD_dbg	0x20000004	ushort
led_mask	0x00000B2C	array[8] of uint
text	0x00000000	array[40] of uchar
ADC_init	0x00000300	Function
IRQn	[R13+#0]	uchar
LED_init	0x00000374	Function
LED_Off	0x0000038A	Function
num	R0	uint
LED_On	0x000003A6	Function
num	R0	uint
LED_Out	0x000003CE	Function
i	R3	int
value	R0	uint
main	0x000003F4	Function
AD_...	R5	ushort
AD_...	R4	ushort
ticks	R0	uint
IRQ		Module
Retarget		Module
Serial		Module
startup_MB9...		Module
system_MB9...		Module

Debug Windows

- Terminal Windows

- Up to 3 UART channels can be displayed
- One direct `printf` view is supported (via ITM Viewer)

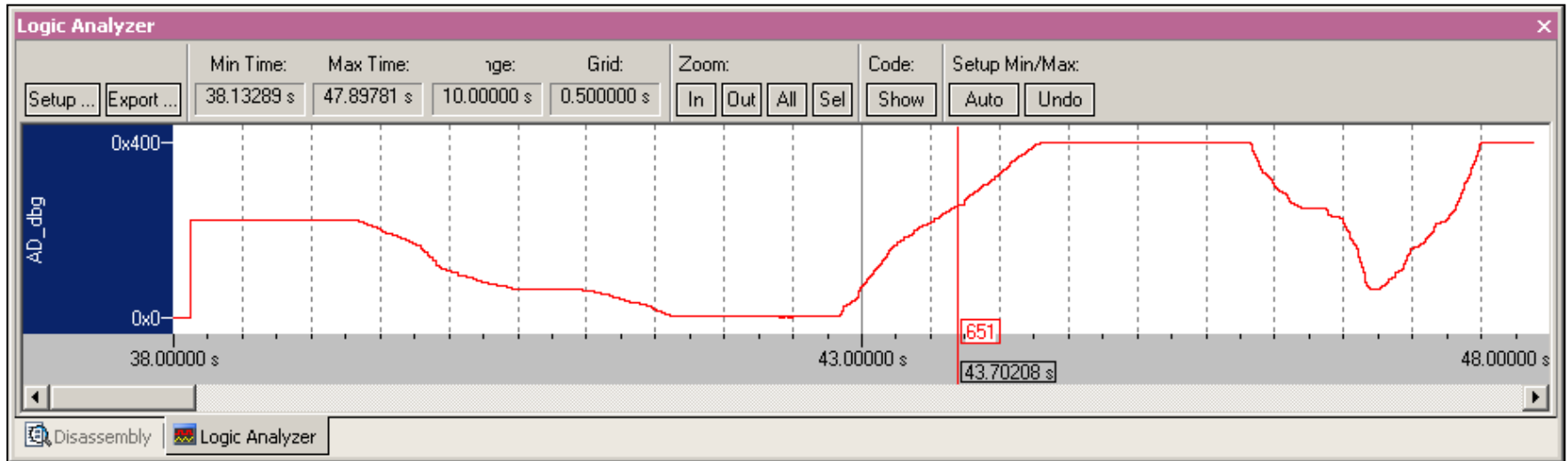
```
116     clock_is = 0;  
117     printf ("AD value = 0x%04X\n\r", AD_print);  
118 }
```



Debug Windows

■ Logic Analyzer

- The IDE allows to display graphically the value of a variable
- Adding a signal is as easy as adding to a Watch window
- The Logic Analyzer is a tabbed window together with the Disassembly window



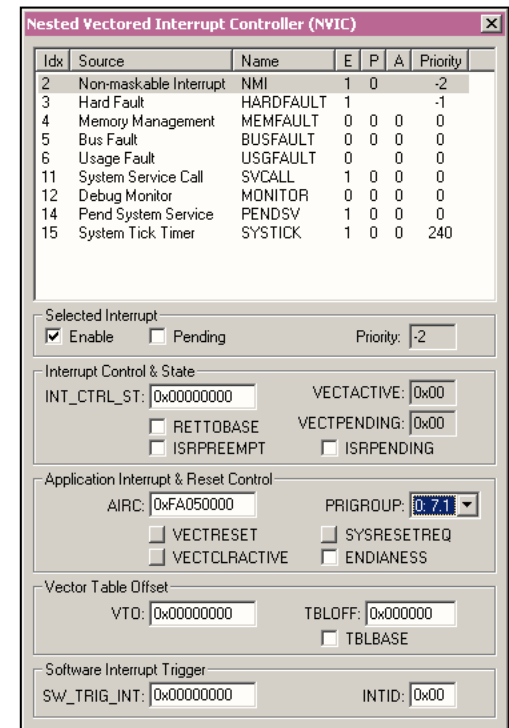
Core Peripherals

■ Nested Vectored Interrupt Controller

– The NVIC settings for the Core Interrupts can be adjusted in a dialog window

– Adjustments for:

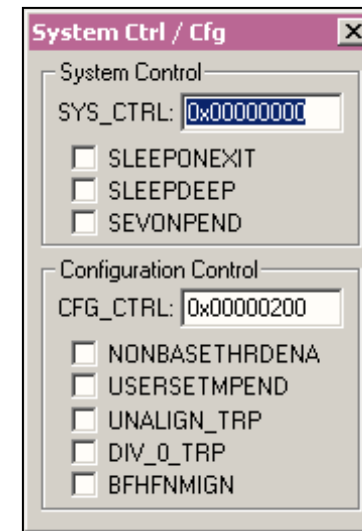
- ◆ Enable/Pending
- ◆ Priority
- ◆ Interrupt Control & State
- ◆ Application Interrupt & Reset Control (PRIGROUP, ENDIANESS, ...)
- ◆ Vector Table Offset
- ◆ Software Interrupt Trigger



Core Peripherals

- System Control Register (0xE000ED10)
 - Dialog window for setting bits for:
 - ◆ SEVONPEND
 - ◆ SLEEPDEEP
 - ◆ SLEEPONEXIT

- Configuration Control Register (0xE000ED14)
 - Dialog window for setting bits for:
 - ◆ NONBASETHRDENA
 - ◆ USERSETMPEND
 - ◆ UNALIGN_TRP
 - ◆ DIV_0_TRP
 - ◆ BFHFNMIGN



Core Peripherals

- System Tick Timer (0xE000E010)
 - Dialog window for setting/watching:
 - ◆ Control & Status
 - ◆ Reload & Current value
 - ◆ Calibration

The screenshot shows the 'System Tick Timer' dialog window with the following settings:

Section	Field	Value	Option	Option
Control & Status	ST_CTRL_STAT:	0x00010007	<input checked="" type="checkbox"/> ENABLE	<input checked="" type="checkbox"/> CLKSOURCE
			<input checked="" type="checkbox"/> TICKINT	<input checked="" type="checkbox"/> COUNTFLAG
	Reload & Current Value			
	ST_RELOAD:	0x000C3500	RELOAD:	0x0C3500
Calibration				
ST_CALIB:	0x400186A0	TENMS:	0x0186A0	
		<input checked="" type="checkbox"/> SKEW	<input type="checkbox"/> NOREF	

Core Peripherals

■ Fault Reports

– Dialog window for setting/watching:

- ◆ Memory Manage Faults
- ◆ Bus Faults
- ◆ Usage Faults
- ◆ Hard Faults
- ◆ Debug Faults

Fault Reports

Memory Manage Faults:
MM_FAULT_ADDR: 0xE000EDF8
MM_FAULT_STAT: 0x00
 ACCVIOL MUNSTKERR
 DACCVIOL MSTKERR
 MMARVALID

Bus Faults:
BUS_FAULT_ADDR: 0xE000EDF8
BUS_FAULT_STAT: 0x00
 IBUSERR UNSTKERR
 PRECISERR STKERR
 IMPRECISERR BFARVALID

Usage Faults:
USG_FAULT_STAT: 0x0000
 UNDEFINSTR NOCP
 INVSTATE UNALIGNED
 INVPC DIVBYZERO

Hard Faults:
HARD_FAULT_STAT: 0x00000000
 VECTBL DEBUGEVT
 FORCED

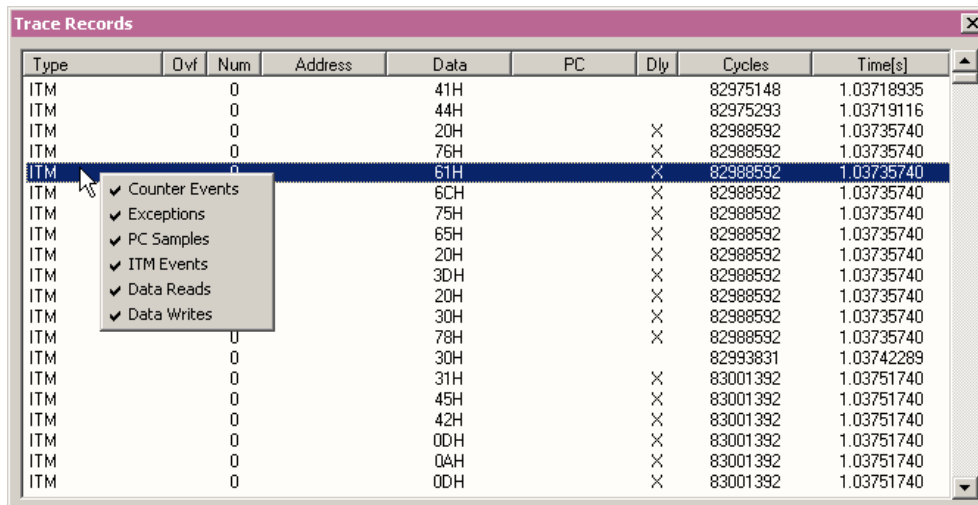
Debug Faults:
DBG_FAULT_STAT: 0x00000002
 HALTED VCATCH
 BKPT EXTERNAL
 DWTTRAP

Trace (ULINK ME)

- Trace via ITM

- Simple Trace views via Instrumentation Trace Macro is supported by μ LINK ME

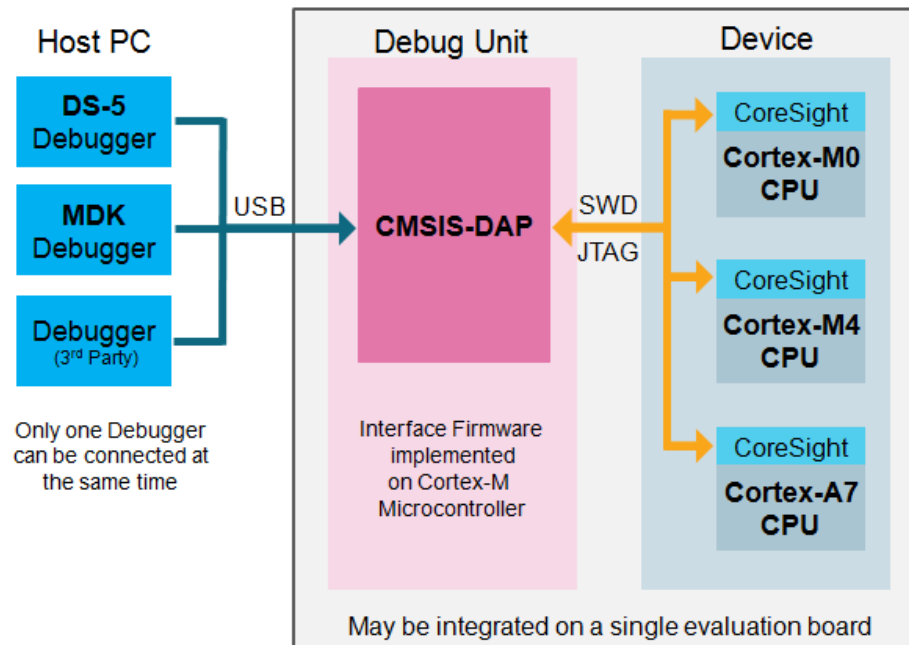
- ◆ Records
- ◆ Exceptions
- ◆ Counters



Type	Ovf	Num	Address	Data	PC	Dly	Cycles	Time[s]
ITM		0	41H				82975148	1.03718935
ITM		0	44H				82975293	1.03719116
ITM		0	20H			X	82988592	1.03735740
ITM		0	76H			X	82988592	1.03735740
ITM		0	61H			X	82988592	1.03735740
ITM		0	6CH			X	82988592	1.03735740
ITM		0	75H			X	82988592	1.03735740
ITM		0	65H			X	82988592	1.03735740
ITM		0	20H			X	82988592	1.03735740
ITM		0	3DH			X	82988592	1.03735740
ITM		0	20H			X	82988592	1.03735740
ITM		0	30H			X	82988592	1.03735740
ITM		0	78H			X	82988592	1.03735740
ITM		0	30H				82993831	1.03742289
ITM		0	31H			X	83001392	1.03751740
ITM		0	45H			X	83001392	1.03751740
ITM		0	42H			X	83001392	1.03751740
ITM		0	0DH			X	83001392	1.03751740
ITM		0	0AH			X	83001392	1.03751740
ITM		0	0DH			X	83001392	1.03751740

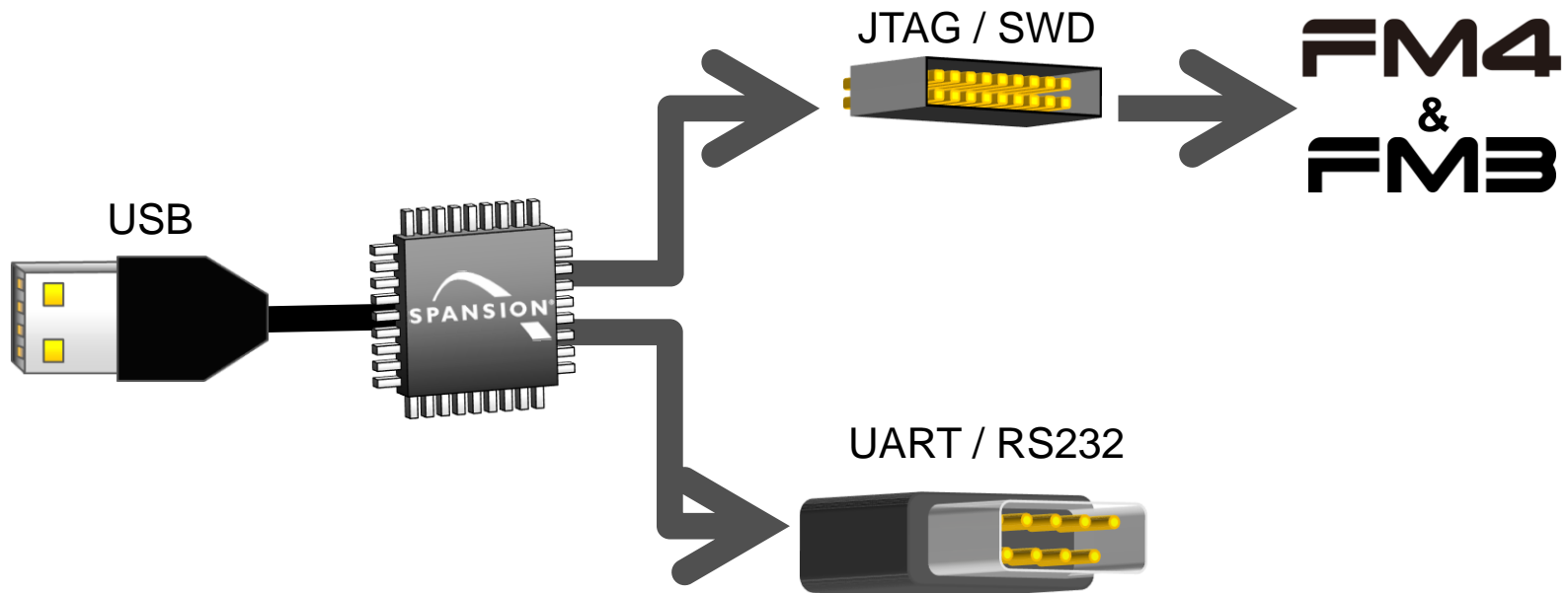
Definition (ARM-Webpage)

- CMSIS-DAP is the interface firmware for a Debug Unit that connects the Debug Port to USB. Debuggers, which execute on a host computer, connect via USB to the Debug Unit and to the Device that runs the application software. The Debug Unit connects via JTAG or SWD to the target Device. ARM Cortex processors provide the [CoreSight Debug and Trace Unit](#). CMSIS-DAP supports target devices that contain one or more Cortex processors.



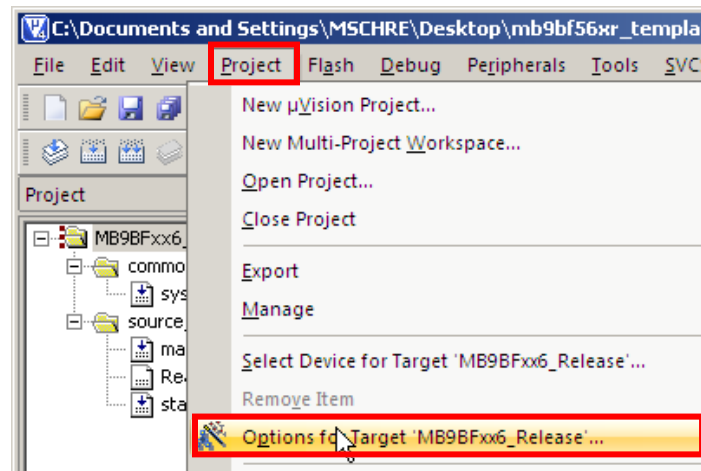
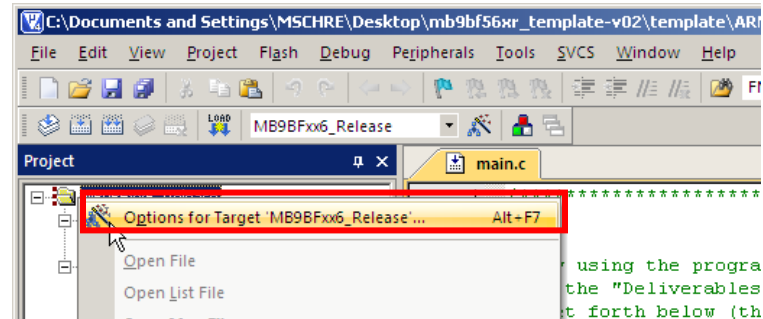
Additional Features

- Spansion CMSIS-DAP implementation offers
 - 1 channel JTAG or SWD (Single Wire Debug)
 - Additional 1 channel UART / RS232



Setup in Keil μ Vision (1)

- Navigate to project options:
 - Via Project
 - ◆ Right-click at the project
 - ◆ Select „Options...“
 - Or via menu „Project“
 - ◆ Select „Options...“



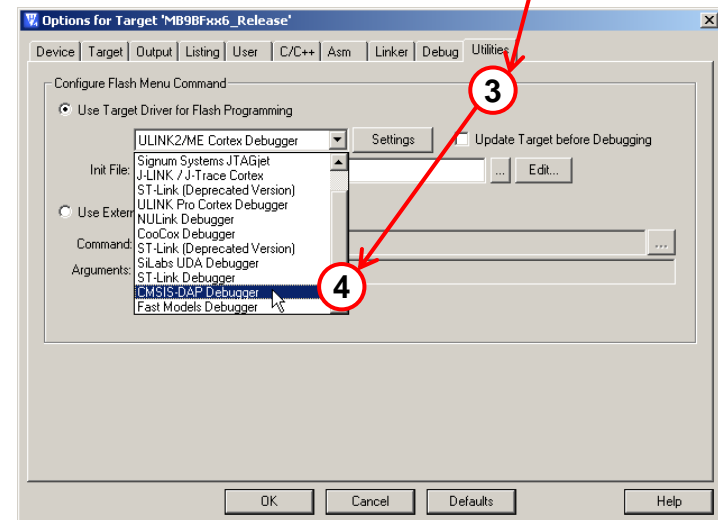
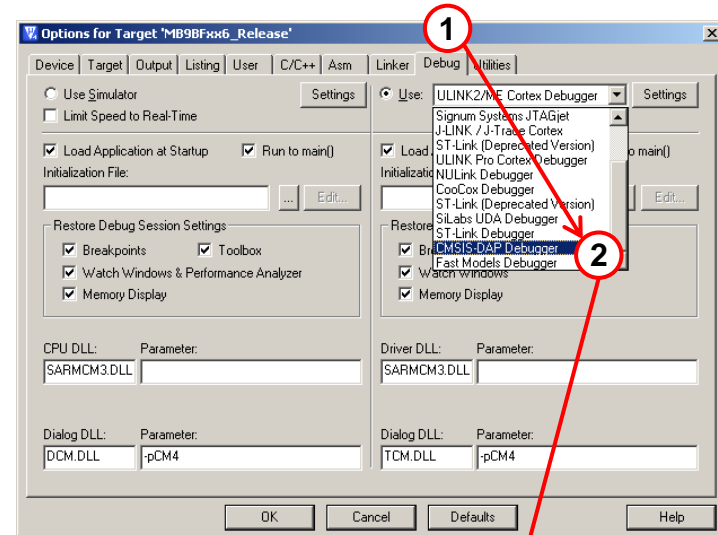
See also Chapter [Debugging \(Keil \$\mu\$ Vision\)](#)

Setup in Keil μ Vision (2)

■ Setup Debug & Utilities

- (1) Select tab „Debug“
- (2) Select „CMSIS-DAP Debugger“

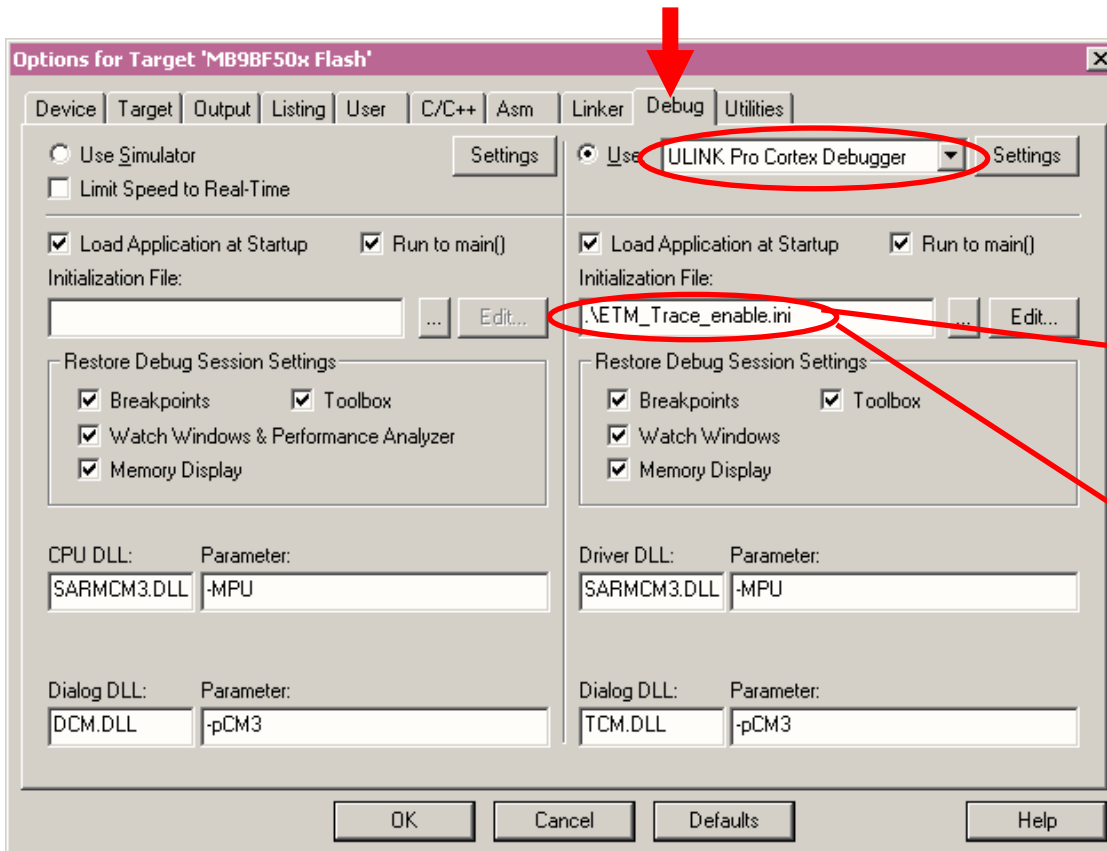
- (3) Select tab „Utilities“
- (4) Select „CMSIS-DAP Debugger“



See also Chapter [Debugging \(Keil \$\mu\$ Vision\)](#)

ULINK Pro

- Trace via ETM
 - Check settings in menu:
Flash → *Configure Flash Tools...* Tab: *Debug*



```
ETM_Trace_enable.ini - Notepad
File Edit Format View Help
_LWdWORD(0x40033000, 0x000003FF);
_WBYTE(0x40033603, 0x03);
```

enables ETM pins

ULINK Pro

■ Instruction Trace

- Real Time Trace recording
- Output can be filtered by several ETM and ITM events
- Trace buffer is held in PC memory and transferred to μ Vision on break

The screenshot displays the 'Instruction Trace' window in Keil μ Vision. The window has a 'Filter' dropdown set to 'All'. Below the filter is a table with the following columns: #, Type, Flag, Num, PC, Opcode, Instruction, and Source Code. The table contains several rows of instruction data, with the row for instruction #1048566 highlighted in blue. Below the table, the source code for 'Blinky.c' is visible, with line 111 highlighted in blue, corresponding to the selected instruction in the trace.

#	Type	Flag	Num	PC	Opcode	Instruction	Source Code
1048564	ETM			0x0000043E	4284	CMP r4,r0	
1048565	ETM			0x00000440	D001	BEQ 0x00000446	
1048566	ETM			0x00000446	42AC	CMP r4,r5	111: if (AD_value != AD_print) { /* Make sure that AD inter
1048567	ETM			0x00000448	D002	BEQ 0x00000450	
1048568	ETM			0x00000450	4814	LDR r0,[pc,#80] ; @0x000004A4	116: if (clock_1s) {
1048569	ETM			0x00000452	7800	LDRB r0,[r0,#0x00]	

```
108     if (AD_value != AD_last)           /* Make sure that AD interrupt did */
109         AD_value = AD_last;           /* not interfere with value reading */
110
111     if (AD_value != AD_print) {        /* Make sure that AD interrupt did */
112         AD_print = AD_value;           /* Get unscaled value for printout */
113         AD_dbg   = AD_value;
```

Trace Exceptions

■ Exception Trace

– Exception Trace has an own window, which can't be docked

– Exception Types:

- ◆ System Exceptions

- ◆ User Exceptions (Interrupts)

– Exception Trace:

- ◆ Count

- ◆ Total Time

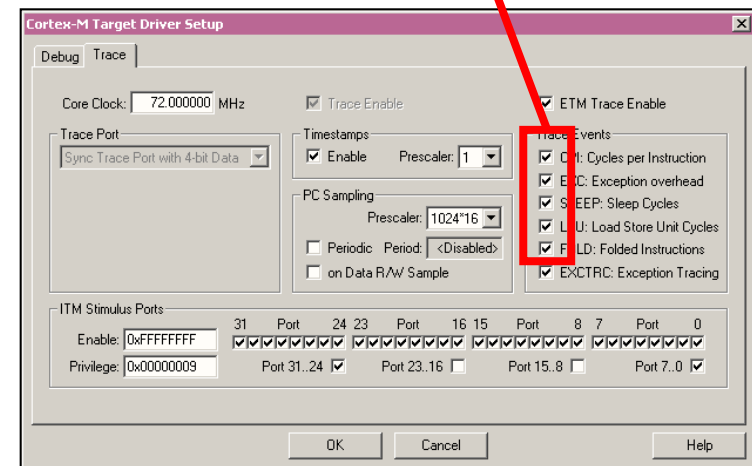
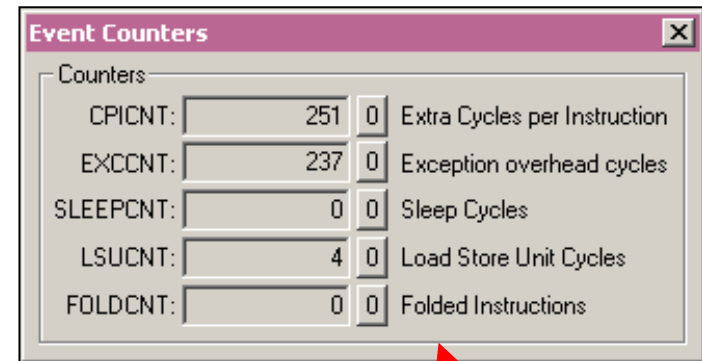
- ◆ Several Points in Time: Min., Max., First, Last, etc.

– Must be configured in: *Debug*→*Debug Settings...* Tab: *Trace/Trace Events*

Num	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
2	NMI	0	0 s						
3	HardFault	0	0 s						
4	MemManage	0	0 s						
5	BusFault	0	0 s						
6	UsageFault	0	0 s						
11	SVCall	0	0 s						
12	DbgMon	0	0 s						
14	PendSV	0	0 s						
15	SysTick	0	0 s						
16	ExtIRQ 0	0	0 s						
17	ExtIRQ 1	0	0 s						
18	ExtIRQ 2	0	0 s						
19	ExtIRQ 3	0	0 s						
20	ExtIRQ 4	0	0 s						
21	ExtIRQ 5	0	0 s						
22	ExtIRQ 6	0	0 s						
23	ExtIRQ 7	0	0 s						

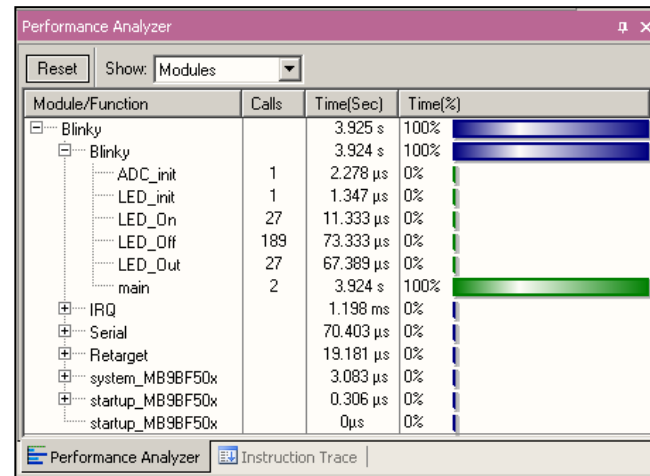
Event Counters

- Following counters are available:
 - Extra cycles per Instruction
 - Exception overhead cycles
 - Sleep cycles
 - Load store unit cycles
 - Folded instructions
- Counters count from 0 to 255
 - Continue counting on overflow starting over from 0
- Must be configured in:
Debug → *Debug Settings...*
Tab: *Trace/Trace Events*
(Checkboxes)



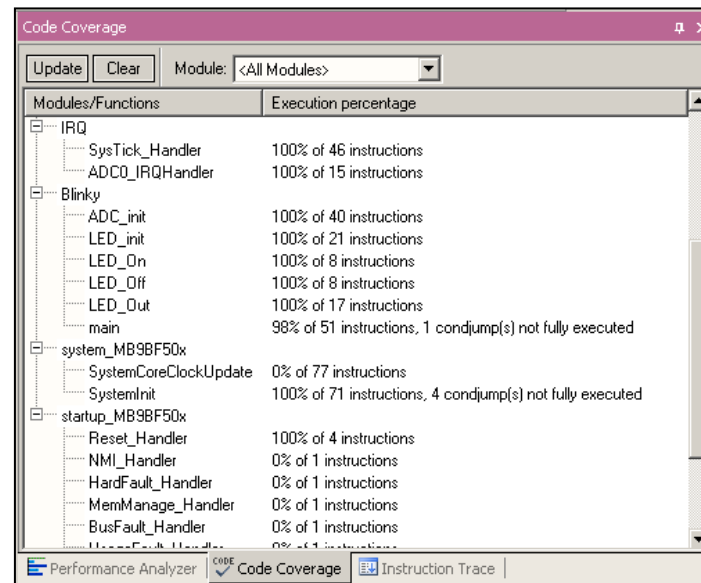
Performance Analyzer (ULINK Pro)

- Shows the execution of any function of an application in:
 - Number of calls
 - Execution time (absolute)
 - Execution time (percentage)
- It shares with a tab the Instruction Trace window and Code Coverage
- Module or function view is supported



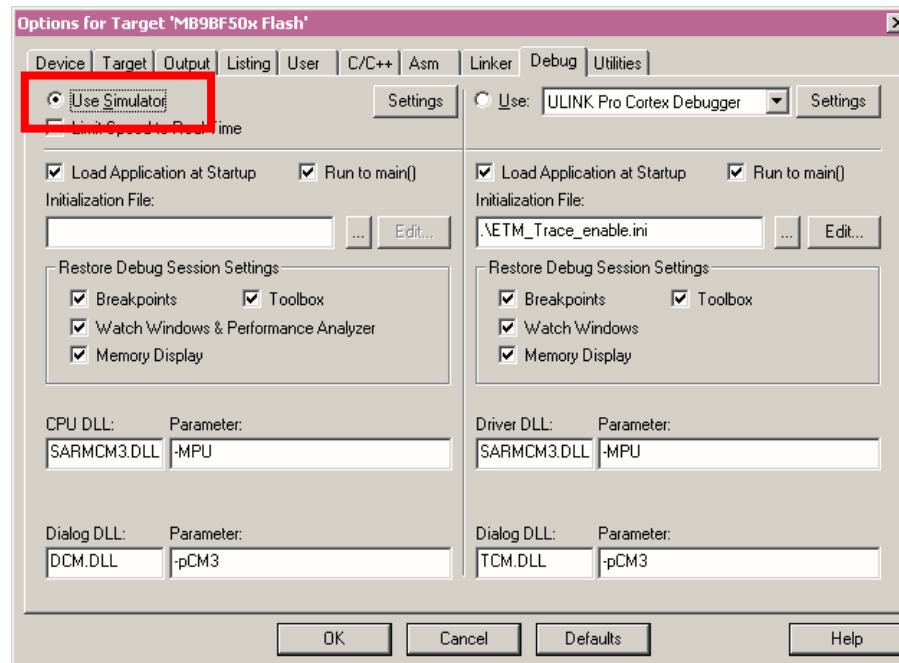
Code Coverage

- The Code Coverage shows the function execution by percentage of the function code
- It shares with a tab the Instruction Trace window and Performance view
- All or only a specific module can be chosen for display



Simulator

- The Core Simulator can be selected by the menu: *Flash*→*Configure Flash Tools...* and then choosing *Use Simulator*
- Look & feel is like using ULINK debugger
- Controlable also with *.ini files





www.spansion.com

SpanSION®, the SpanSION logo, MirrorBit®, MirrorBit® Eclipse™ and combinations thereof are trademarks and registered trademarks of Spansion LLC in the United States and other countries. Other names used are for informational purposes only and may be trademarks of their respective owners.

This document is for informational purposes only and subject to change without notice. Spansion does not represent that it is complete, accurate or up-to-date; it is provided "AS IS." To the maximum extent permitted by law, Spansion disclaims any liability for loss or damages arising from use of or reliance on this document.