

IAR Embedded Workbench for Infineon XMC

Martin Gisbert
FAE IAR Systems



- Overview
- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy
- Hands-on & Demo

- Overview

- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy
- Hands-on & Demo

IAR SYSTEMS—THE WORLD’S LEADING PROVIDER OF EMBEDDED DEVELOPMENT TOOLS



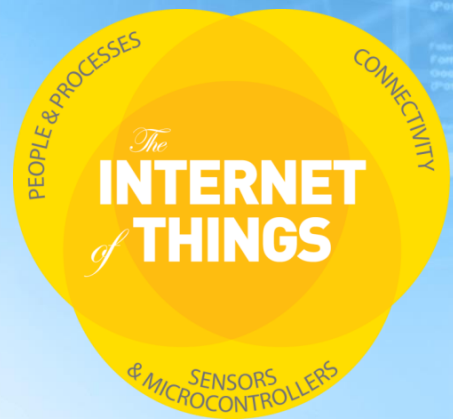
168 Employees with HQ in Uppsala, Sweden
Listed in Stockholm/Nasdaq
R&D investment 32% of revenue
32 years in the industry



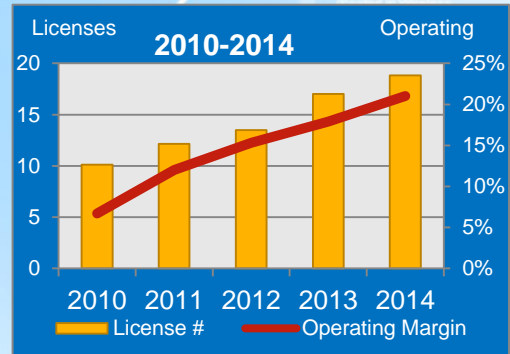
24 hour technical support in
13 languages

- Uppsala
- Munich
- Sao Paulo
- Tokyo
- Seoul
- Shanghai
- London
- Paris
- San Francisco
- Dallas
- Boston
- Los Angeles

+Distributor representation in
43 countries



Stability and growth



What is IAR Embedded Workbench?



IAR Embedded Workbench

IDE Tools

Editor
Project manager
Library builder
Librarian

Build Tools

IAR C/C++ Compiler
Assembler
Linker

IAR C-SPY Debugger

Simulator driver
Hardware system drivers
Power debugging
RTOS plug-ins
C-SPY Macros

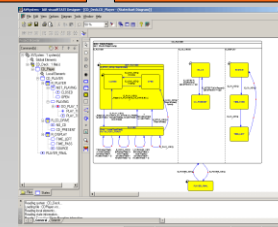


Plug-ins



Validation Reports

MISRA



visualSTATE



Support



Strong ecosystem



Training

C-STAT
C-RUN

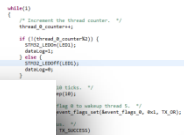


Project Connections

Configuration tools



Editors



Source code control systems

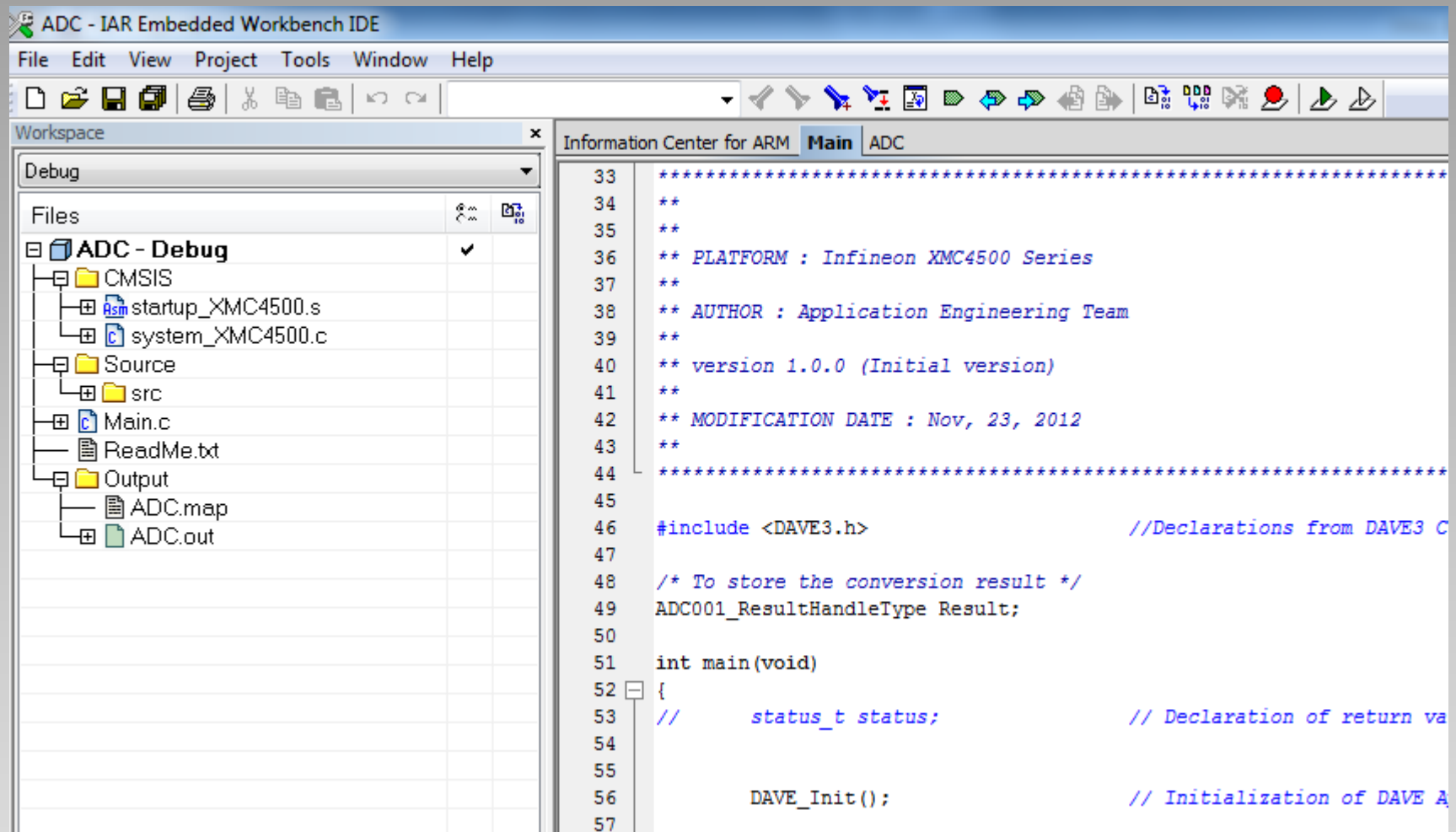


IDE

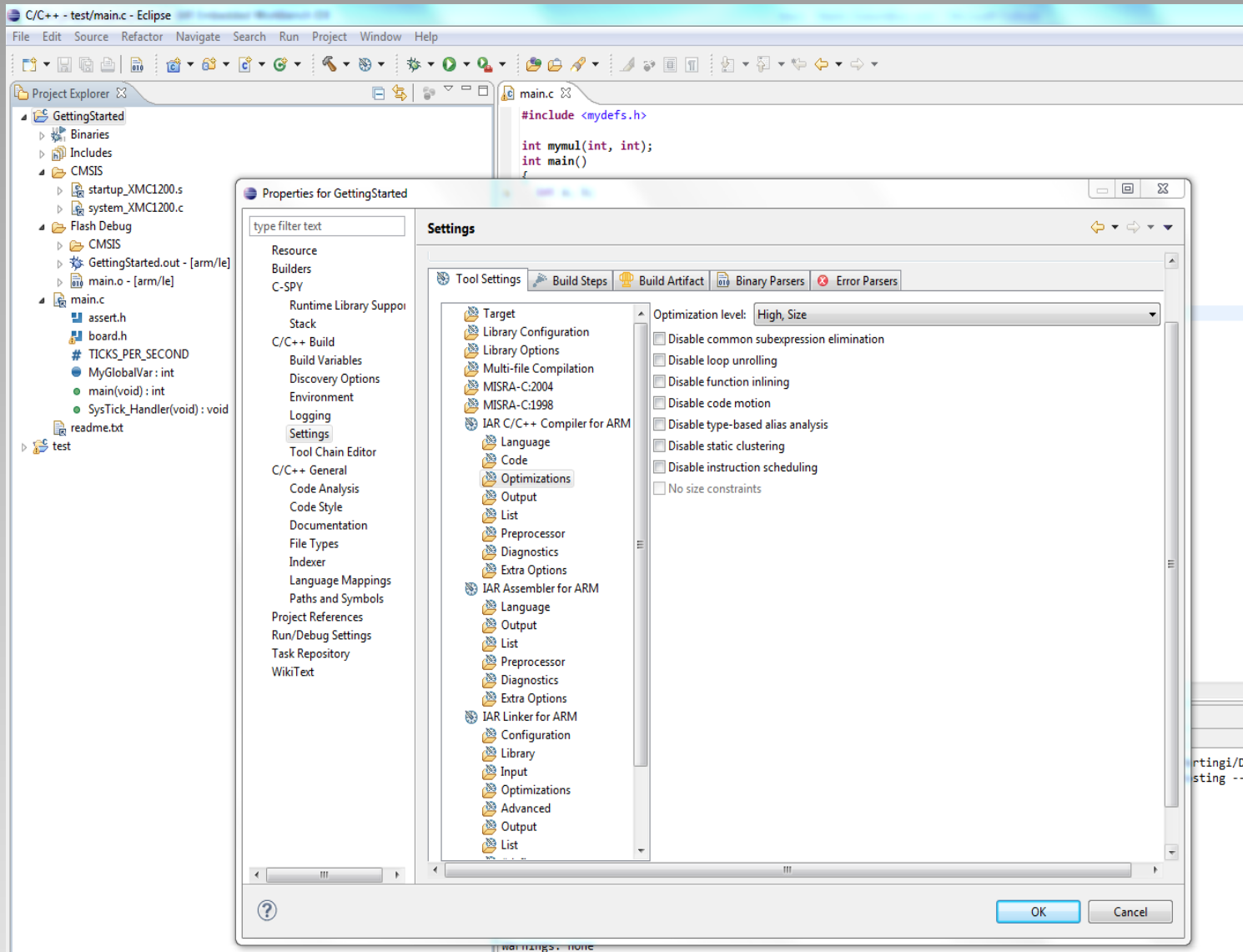


- Overview
- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy
- Hands-on & Demo

IAR Embedded Workbench IDE

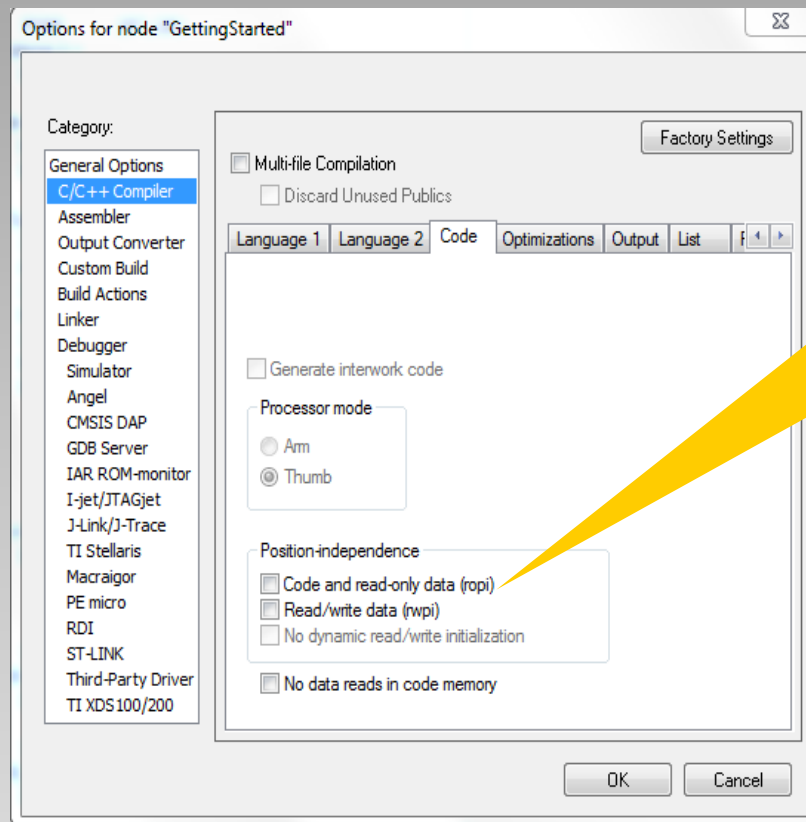


Eclipse Plugin (Code View)



- Overview
- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy
- Hands-on & Demo

Powerful C/C++ Compiler (Cont'd)



Support for position independent code and data

ropi generates code that uses PC-relative references to address code and read-only data

Powerful C/C++ Compiler

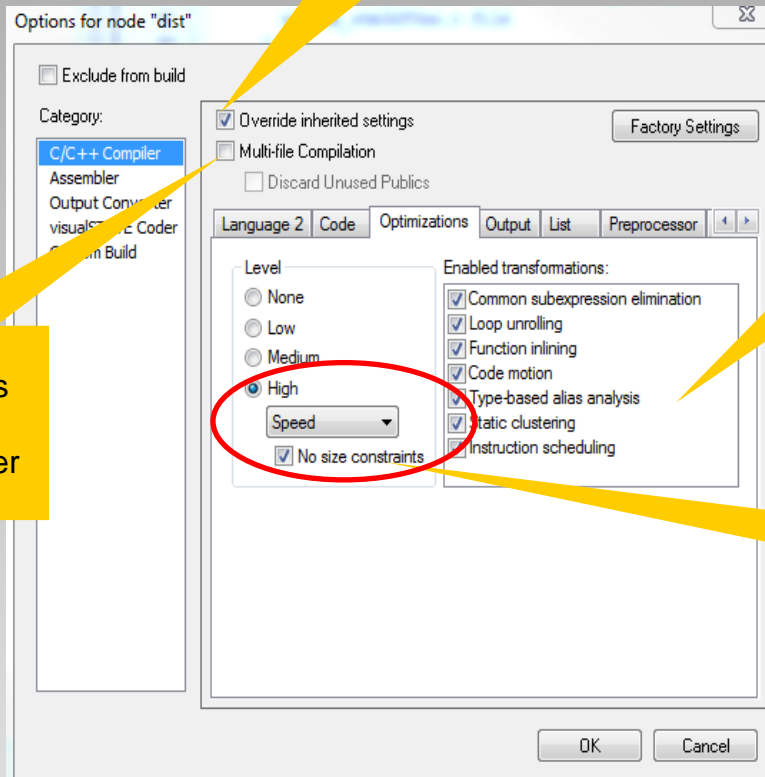


IAR
SYSTEMS

Balance between size and speed can be achieved by setting different optimizations for different parts of the code

Major functions of the optimizer can be controlled individually

Multi-file compilation allows the optimizer to operate on a larger set of code



Speed option "no size constraints"

Well-tested

- Commercial test suites
 - Plum-Hall
 - Perennial
 - Dinkumware library test

- In-house developed test suite >500,000 lines of C/C++ test code run multiple times
 - Processor modes
 - Memory models
 - Optimization levels

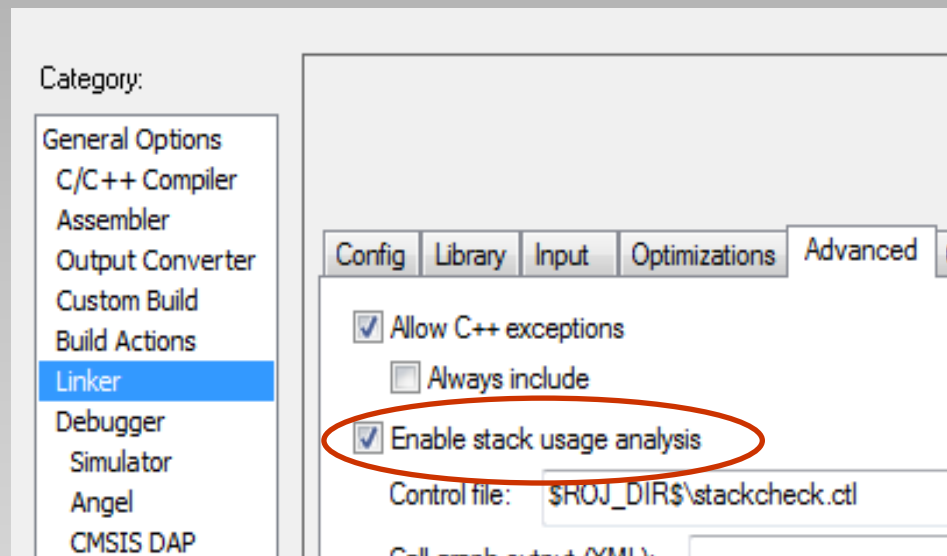
Language standards

- ISO/IEC 9899:1990 (C94/C90/C89/ANSI C)
- ISO/IEC 9899:1999 (C99/Standard C)
- ISO/IEC 1488:2003 (Standard C++)
- Embedded C++ and Extended Embedded C++ dialects are also supported

- Overview
- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy
- Hands-on & Demo

Linker - Stack usage analysis

Under the right circumstances, the ILINK linker can calculate the maximum stack usage for each call graph root.



Linker - Stack usage analysis

- Stack usage chapter in the linker map file, listing of maximum stack depth for each call graph root.
- Alternative generation of a .xml call graph file
- Use linker extra option `--log call_graph` for better understanding

```
*****
***  STACK USAGE
***

Call Graph Root Category  Max Use  Total Use
-----
interrupt                 8          8
main_root                164        164

main_root
  "__iar_program_start": 0x08009dcd

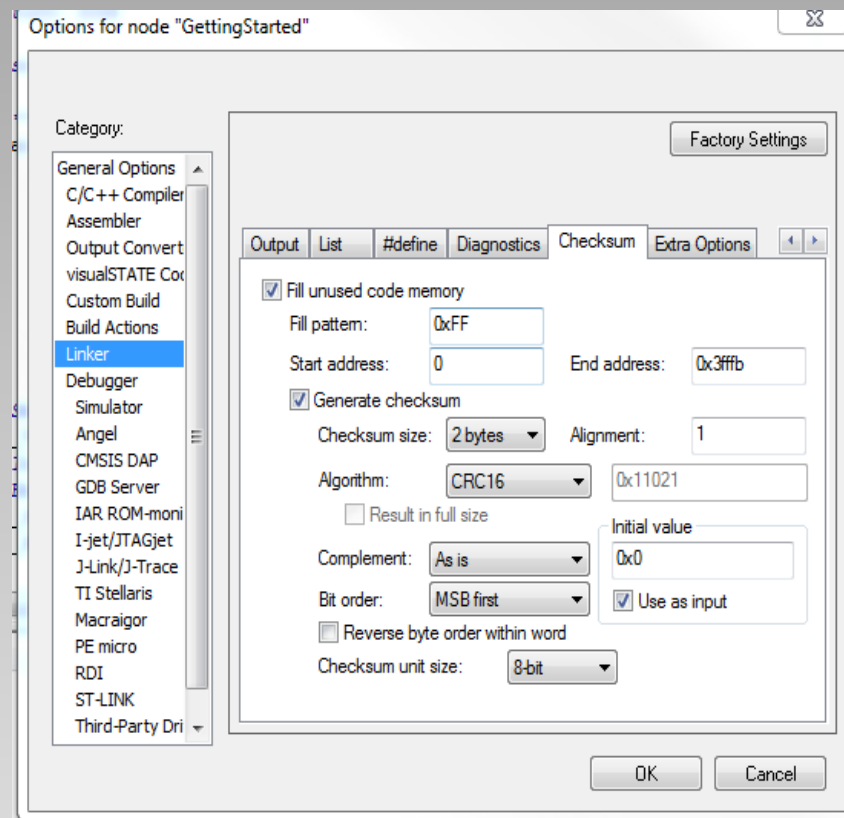
Maximum call chain                      164
bytes

  "__iar_program_start"                  0
  "__cmain"                             0
  "main"                                 8
  "DrawPicture"                          24
  "ResetPicture"                         16
  "GLCD_SendCmd"                         32
  + 1 cycles in nest 0                   32
  "GLCD_SendCmd"                         32
  "GLCD_SPI_ReceiveBlock"                16
  "GLCD_SPI_TranserByte"                 4
  ...
```

.map

Checksum calculation

- The linker can generate a checksum over the complete ROM, alternatively different sections
- After the download, the same checksum calculation is done in source code and the result is compared to verify that the flash is not corrupted
- Relevant Technical Notes:
 - Checksum calculation with IELFTOOL after linking ILINK
<http://supp.iar.com/Support/?note=11927>
 - IELFTOOL Checksum - over several ranges
<http://supp.iar.com/Support/?note=53274>
 - Calculate CRC32 as in K60 hardware
<http://supp.iar.com/Support/?note=85753>

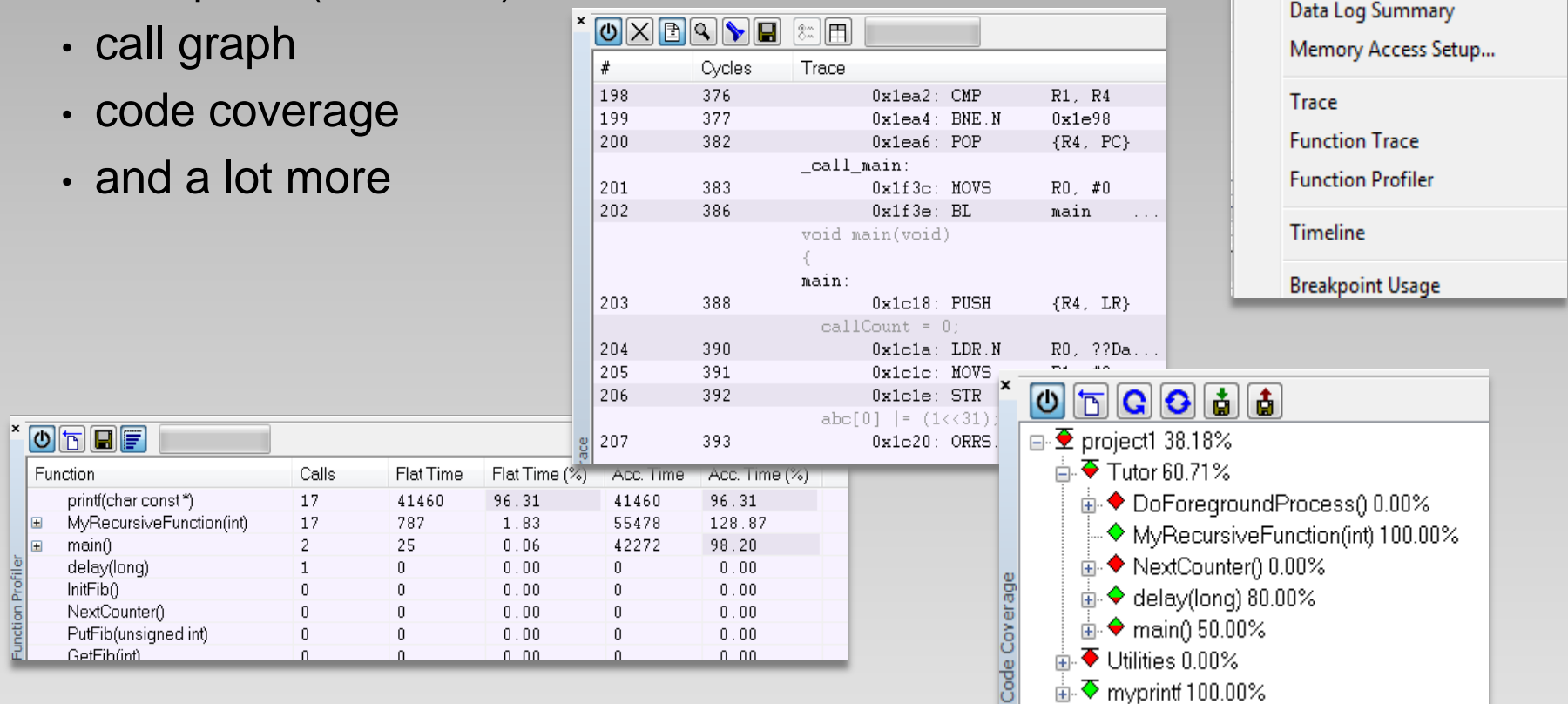


- Overview
- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy
- Hands-on & Demo

C-SPY Simulator

In contrast to the debugger with bandwidth-limited communication interface, the simulator does not miss anything and can show

- Complete (Function) Trace
- call graph
- code coverage
- and a lot more



The screenshot displays the C-SPY Simulator interface with three main windows open:

- Function Profiler:** A table showing the execution time and call counts for various functions.
- Trace:** A window showing the execution trace, including the instruction address, cycles, and the instruction itself.
- Code Coverage:** A window showing the code coverage for the project, including the percentage of code executed for each function.

Function Profiler Table:

Function	Calls	Flat Time	Flat Time (%)	Acc. Time	Acc. Time (%)
printf(char const*)	17	41460	96.31	41460	96.31
MyRecursiveFunction(int)	17	787	1.83	55478	128.87
main()	2	25	0.06	42272	98.20
delay(long)	1	0	0.00	0	0.00
InitFib()	0	0	0.00	0	0.00
NextCounter()	0	0	0.00	0	0.00
PutFib(unsigned int)	0	0	0.00	0	0.00
GetFib(int)	0	0	0.00	0	0.00

Trace Window:

#	Cycles	Trace
198	376	0x1ea2: CMP R1, R4
199	377	0x1ea4: BNE.N 0x1e98
200	382	0x1ea6: POP {R4, PC}
_call_main:		
201	383	0x1f3c: MOVS R0, #0
202	386	0x1f3e: BL main ...
void main(void)		
{		
main:		
203	388	0x1c18: PUSH {R4, LR}
callCount = 0;		
204	390	0x1c1a: LDR.N R0, ??Da...
205	391	0x1c1c: MOVS
206	392	0x1c1e: STR
abc[0] = (1<<31);		
207	393	0x1c20: ORRS

Code Coverage Window:

Function	Coverage (%)
project1	38.18%
Tutor	60.71%
DoForegroundProcess()	0.00%
MyRecursiveFunction(int)	100.00%
NextCounter()	0.00%
delay(long)	80.00%
main()	50.00%
Utilities	0.00%
myprintf	100.00%

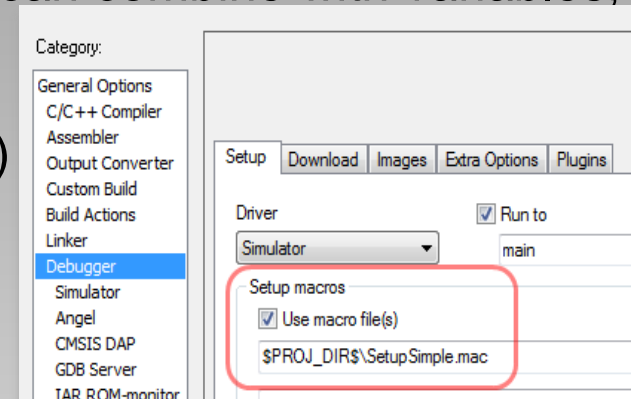
Simulator Menu:

- Interrupt Setup...
- Forced Interrupt
- Interrupt Status Window
- Interrupt Log
- Interrupt Log Summary
- Data Log
- Data Log Summary
- Memory Access Setup...
- Trace
- Function Trace
- Function Profiler
- Timeline
- Breakpoint Usage

C-SPY macros can be written to help you with complex debugging situations.

There are several predefined macros that you can combine with variables, IO and other functions into new macros.

- Interrupt handling (cancel, disable, enable..)
- Reset (issue reset, different types)
- Set/remove breakpoints
- Access files
- Access data



Macros with predefined names will be called at certain times during debugging.

- `execUserReset`
- `execUserExecutionStarted`
- `execUserPreReset`

On-Target debugging: I-jet™

- Supports ARM7/ARM9/ARM11 and Cortex-M/R/A cores
- Hi-speed USB 2.0 interface (480Mbps)
- Target power of up to 400mA can be supplied from I-jet with overload protection
- **Target power consumption** can be measured with ~200µA resolution at 200kHz
- JTAG and Serial Wire Debug (SWD) clocks up to 32MHz (no limit on the MCU clock speed)
- Support for **SWO speeds of up to 60MHz**
- Serial Wire Viewer (SWV) with UART and **Manchester encoding**



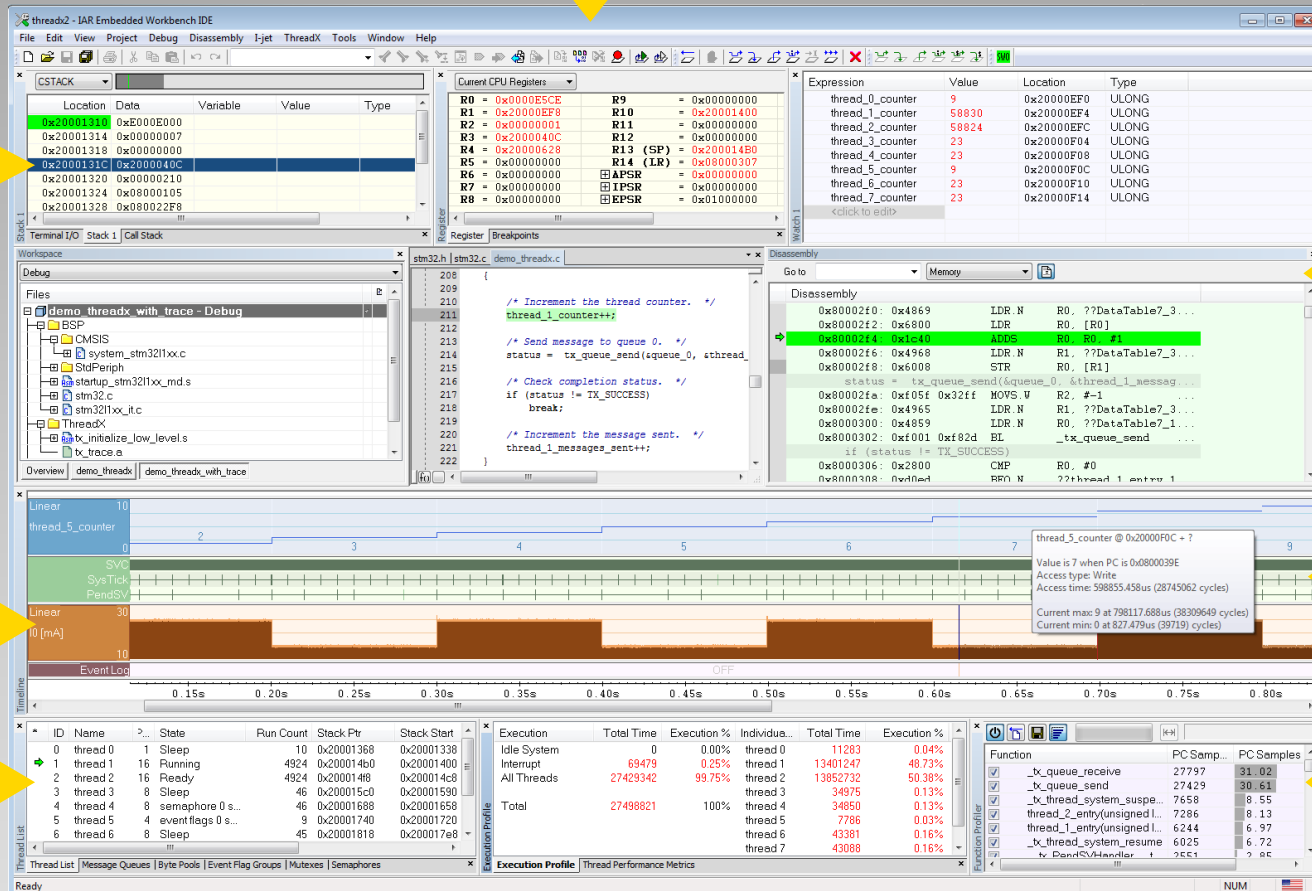
Comprehensive debugger

Integrated debugger for
source and disassembly
debugging

Edit source files
without leaving
the debug
session

- Broad range of in-circuit debugging probes supported
- Trace support
- Direct flash erase and download
- C like macro system extends debugger capabilities
- Built-in simulator driver

Stack usage



Dockable
windows and tab
groups

Timeline window

Power
visualization

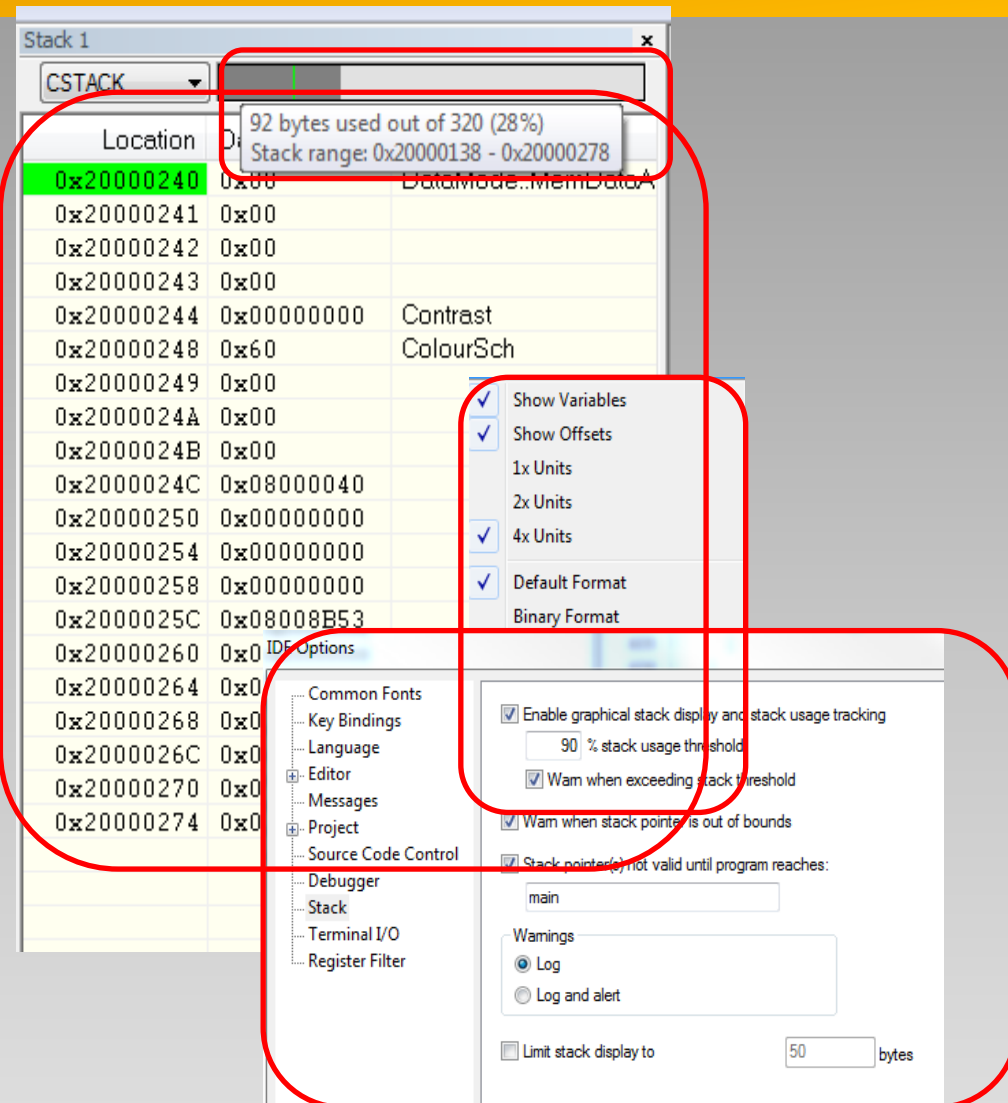
RTOS
awareness

Performance
analysis

Stack view

Displays of:

- Current content of stack
- Current depth of stack
- Max. depth of stack while the target was running
- Optional with/without variables or offset
- Optional with threshold warning and limited stack display



C-SPY Plugin: RTOS

- Ready-made example projects
- Context-sensitive help
- Plugins for RTOS-aware debugging

Micrium

expresslogic



RTOS Plugins (Micrium μ C/OS-III)

OS - IAR Embedded Workbench IDE

File Edit View Project Debug Disassembly I-Jet/ITAGjet Tools Window Help

µC/OS-III

Status: µC/OS-III Running

Statistics Control: Reset Statistics, Update All

General Statistics:

- CPU Usage: n/a
- Tasks: 4
- Idle Counter: 5968925
- Context Switches: 17716

µC/OS-III Task Execution Times:

- Tick Task: 0.00µs
- Timer Task: 0.00µs
- Statistics Task: n/a
- Int Handler Task: n/a

Scheduler:

- Round Robin Enabled: No
- Lock Nesting Counter: n/a
- Max Lock Time: 0.00µs

Message Queue Pool:

- # Free Messages: 100
- # Used Messages: 0

Kernel Objects:

- # Tasks: 4
- # Event Flag Groups: 0
- # Semaphores: 0
- # Message Queues: 0
- # Memory Partitions: 0
- # Mutexes: 0
- # Timers: 0

#	Task Name	Priority	State	Pending On Object	Pending On	CPU Usage	Bar Graph	Context Switches	Stack Pointer	Stack Size	Stack Free	Stack Used	Stack Used %	Bar Graph
0	App Task Start	2	Delayed			0.00%		17	0x20000EC0	64	0	0	0%	
1	µC/OS-III Timer Task	8	Pending	Task Semaphore	Task Sem	0.00%		89	0x200010C0	64	0	0	0%	
2	µC/OS-III Tick Task	7	Pending	Task Semaphore	Task Sem	0.00%		8805	0x20000DE8	128	0	0	0%	
3	µC/OS-III Idle Task	9	Ready			0.00%		8805	0x20001010	64	0	0	0%	

Workspace: app.c | os_cpu_a.asm | os_cpu_c.c | cpu_a.asm | os_tick.c | os_core.c

FLASH

OS - FLASH

- APP
 - app.c
 - app_cfg.h
 - cpu_cfg.h
 - includes.h
 - os_app_hooks.c
 - os_app_hooks.h
 - os_cfg.h
 - os_cfg_app.h
 - stm32f0xx.h
 - stm32f0xx_conf.h
- BSP
 - uC/CPU
 - uC/LIB
 - uC/OS-III
- Output

```
96 /
97 / CPU_CRITICAL_EXIT(); /* CPU_SR_Restore(cpu_sr); */
98 /
99 /
100 /*****
101 /
102 CPU_SR_Save
103 MRS R0, PRIMASK ; Set prio int mask to mask all (except faults)
104 CPSID I
105 BX LR
106
107 CPU_SR_Restore ; See Note #2.
108 MSR PRIMASK, R0
109 BX LR
110
111
112
113 ;PAGE#
114 /*****
115 / WAIT FOR INTERRUPT
116 /
117 ; Description : Enters sleep state, which will be exited when an interrupt is received.
118 /
119 ; Prototypes : void CPU_WaitForInt (void)
120 /
121 ; Argument(s) : none.
122 /*****
123
124 CPU_WaitForInt
```

RTOS Plugins (FreeRTOS)

RTSDemo - IAR Embedded Workbench IDE

File Edit View Project Debug Disassembly I-Jet/JTAGjet FreeRTOS - OpenRTOS Tools Window Help

Workspace

Debug

Files

- RTSDemo - Debug
 - FreeRTOS_Source
 - Portable
 - heap_2.c
 - port.c
 - portasm.s
 - list.c
 - queue.c

RTSDemo

```

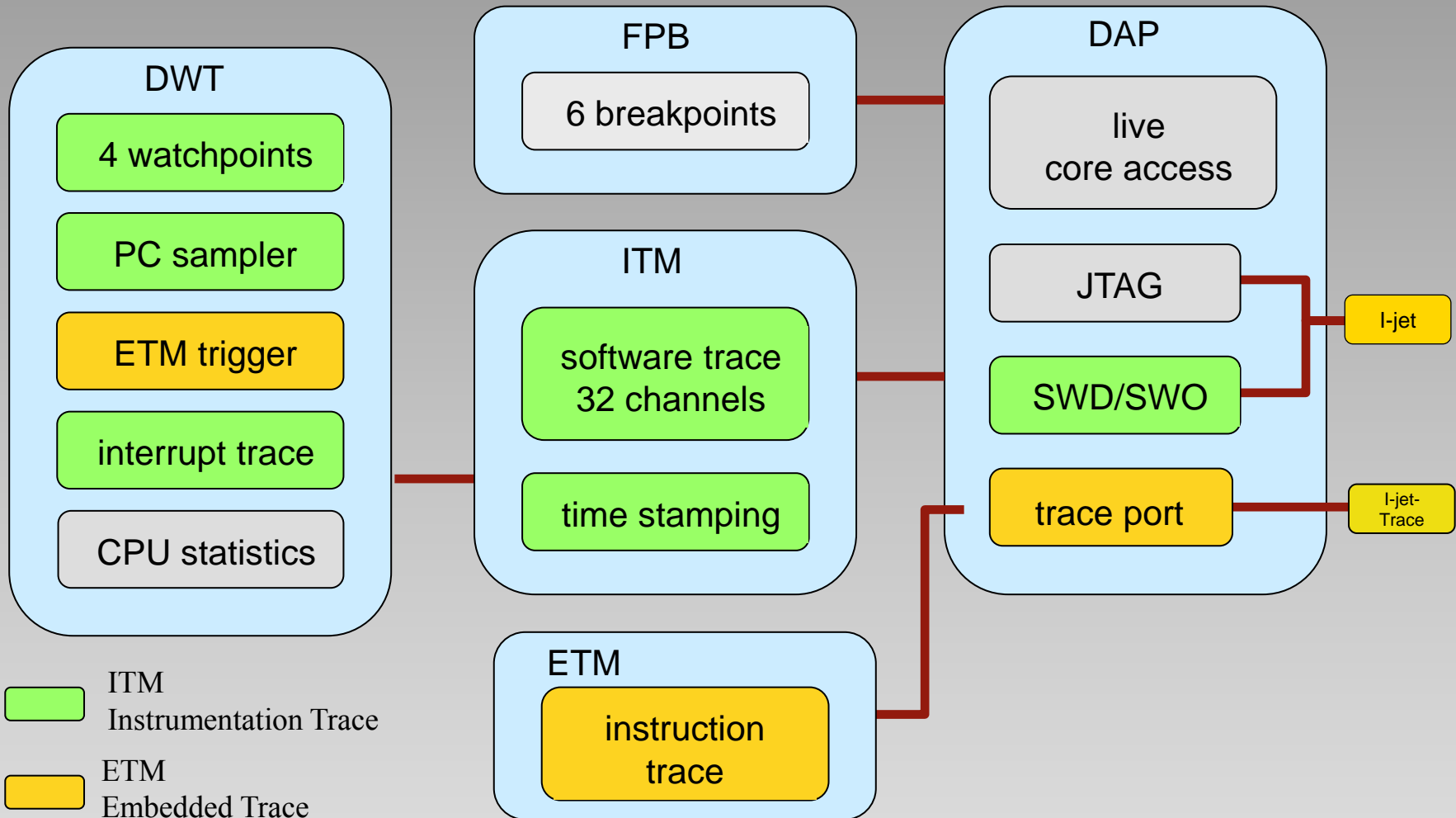
312 tmpreg |= PWR_Regulator;
313
314 /* Store the new value */
315 PWR->CR = tmpreg;
316
317 /* Clear SLEEPDEEP bit of Cortex System Control Register */
318 SCB->SCR &= (uint32_t)~((uint32_t)SCB_SCR_SLEEPDEEP);
319
320 /* Select SLEEP mode entry -----*/
321 if(PWR_SLEEPEntry == PWR_SLEEPEntry_WFI)
322 {
323     /* Request Wait For Interrupt */

```

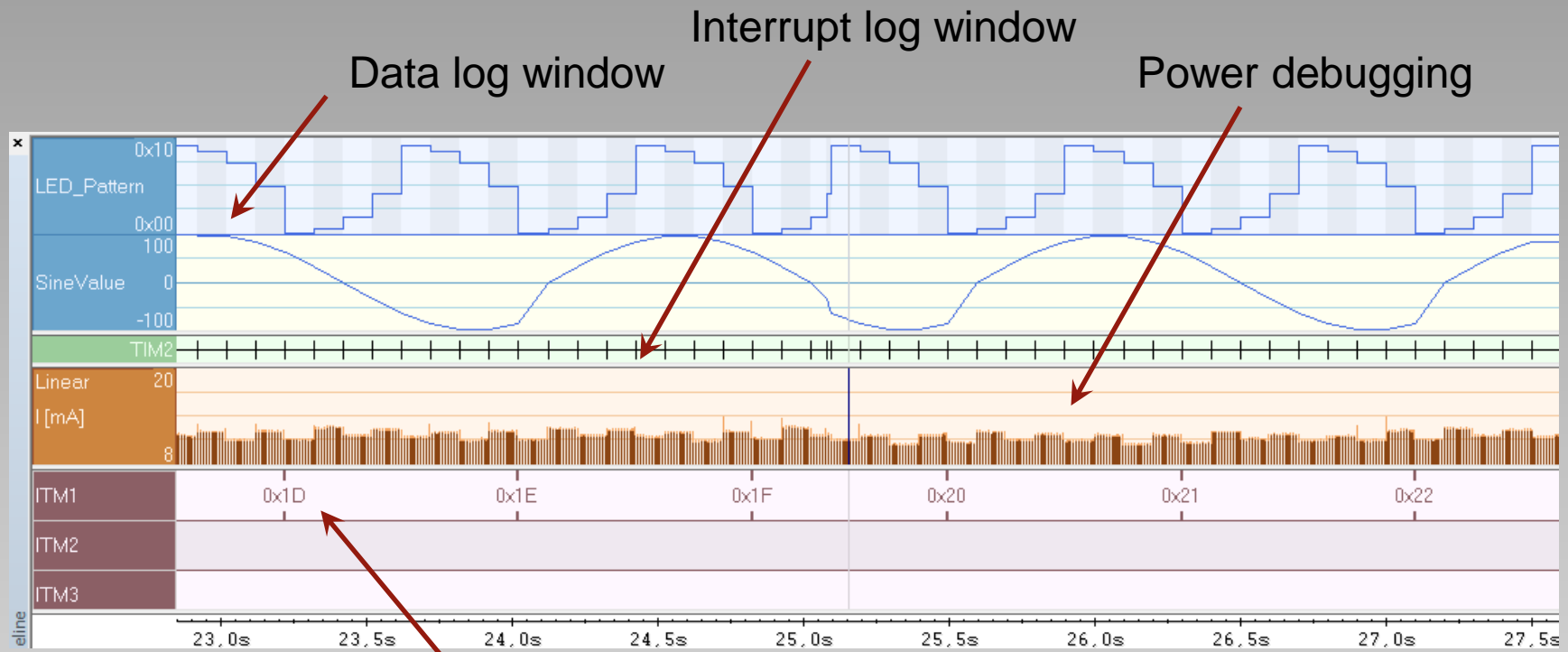
Task Name	Task Number	Priority/actual	Priority/base	Start of Stack	Top of Stack	State	Event Object	Min Free Stack
LIM_INC	4	1	1	0x200007a0	0x20000860	SUSPENDED	None	Off
LEDx	10	1	1	0x200010a0	0x20001150	DELAYING	None	Off
ButPoll	2	0	0	0x20000430	0x200004e8	READY	None	Off
COMTx	11	1	1	0x20001330	0x200013f0	DELAYING	None	Off
C_CTRL	5	0	0	0x20000920	0x200009d8	DELAYING	None	Off
ICL	17	0	0	0x20001d10	0x20001d98	RUNNING	None	Off
MuHigh	16	3	3	0x20001b90	0x20001c50	SUSPENDED	None	Off
COMPr	12	2	2	0x200014b0	0x20001540	BLOCKED	0x20001e4	Off
MuMed	15	2	2	0x20001a10	0x20001ad0	SUSPENDED	None	Off
CNT_INC	3	0	0	0x20000620	0x200006c8	READY	None	Off
MuLow	14	0	0	0x20001890	0x20001940	READY	None	Off
LEDx	9	1	1	0x20000f20	0x20000fd0	DELAYING	None	Off
GenQ	13	0	0	0x200016b0	0x20001770	READY	None	Off
LCD	1	1	1	0x20000198	0x20000348	BLOCKED	0x200000c4	Off
LEDx	8	1	1	0x20000da0	0x20000e50	DELAYING	None	Off
SUSP_RX	7	0	0	0x20000c20	0x20000cd0	READY	None	Off
SUSP_TX	6	0	0	0x20000e00	0x20000eb0	DELAYING	None	Off

Name	Address	Current Length	Max Length	Item Size	# Waiting Tx	# Waiting Rx
LCDQueue	0x200000a0	0	5	8	0	1
Gen_Queue_Test	0x200015d0	0	5	4	0	0
Suspended_Test_Queue	0x20000550	0	1	4	0	0
Gen_Queue_Mutex	0x200017d0	1	1	0	0	0

CoreSight Overview (ARM Cortex-M3/M4)



ITM/SWO Usage (example)



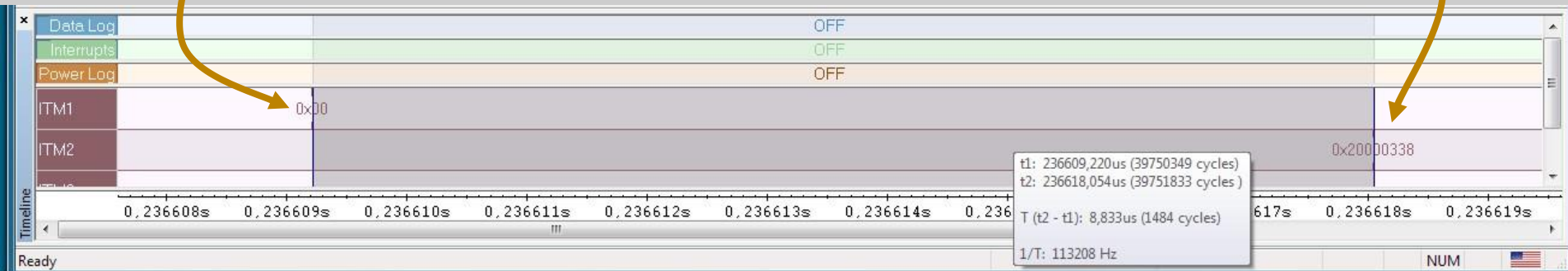
ITM event with line graph

```
ITM_EVENT8(channel_1, my_counter);
```

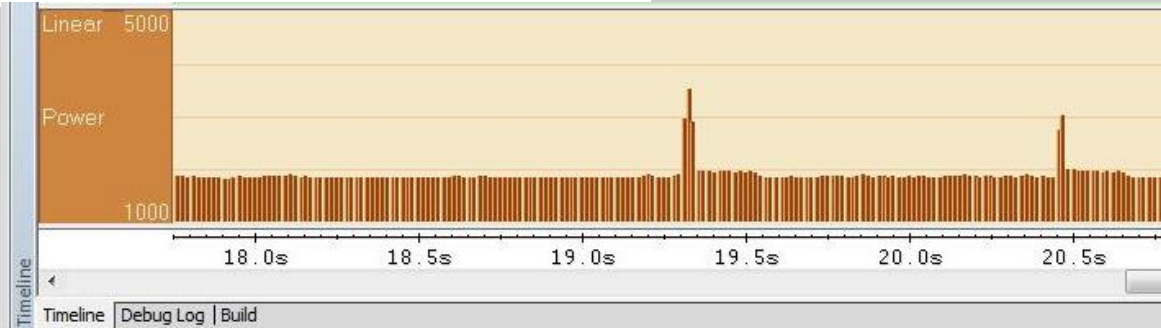
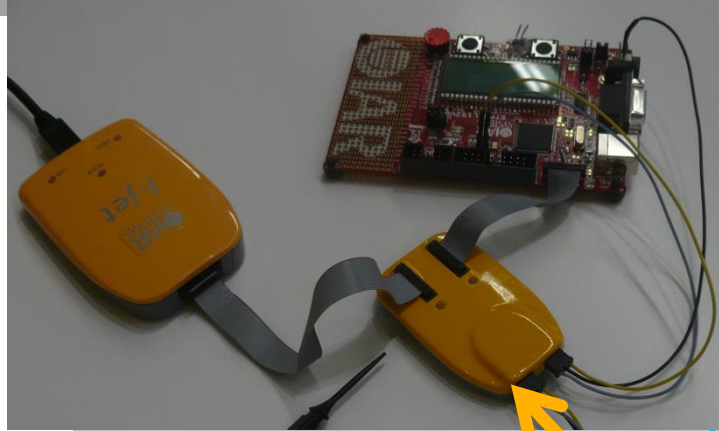
Include the headerfile `arm_itm.h`, here the macros are defined.

- `ITM_EVENT8(channel, value)`
- `ITM_EVENT16(channel, value)`
- `ITM_EVENT32(channel, value)`
- `ITM_EVENT8_WITH_PC(channel, value)`
- `ITM_EVENT16_WITH_PC(channel, value)`
- `ITM_EVENT32_WITH_PC(channel, value)`
- To use, simply include a macro in your code and decide what channel and what value to send
- During debugging, open the timeline window to display the data

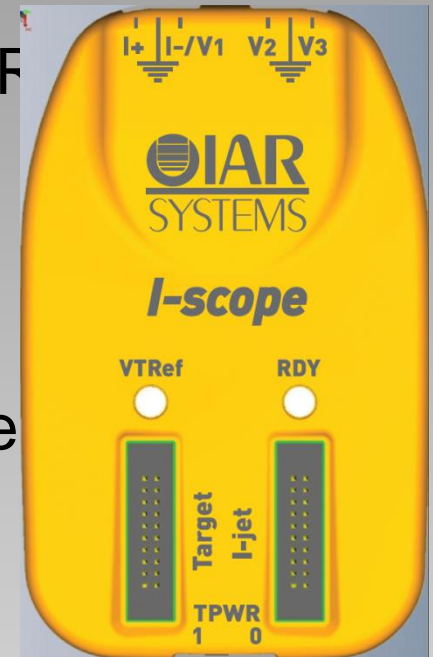
```
ITM_EVENT8(1, 0);  
I2C1_Init();  
ITM_EVENT32(2, __get_SP());
```



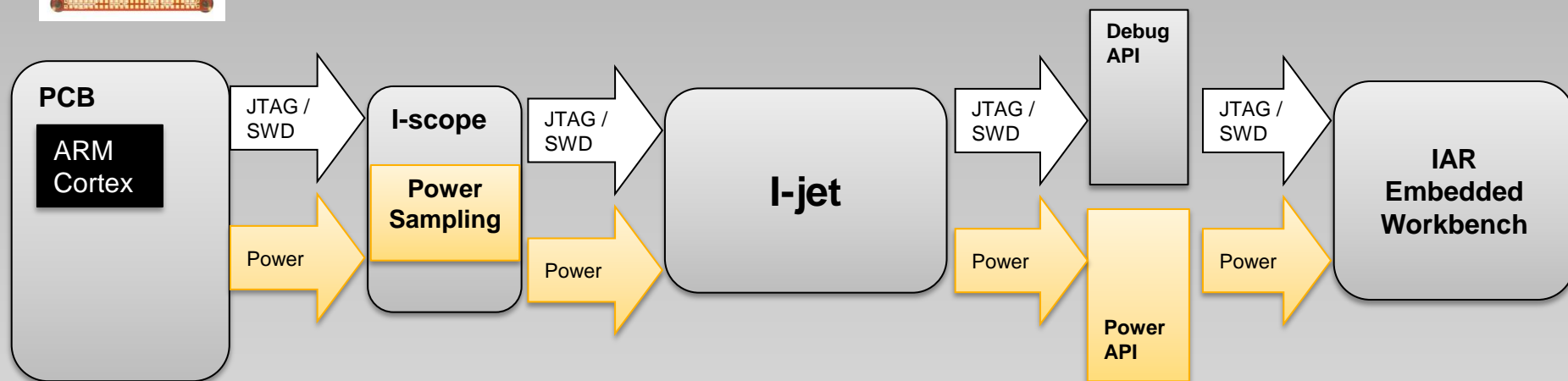
Power Debugging



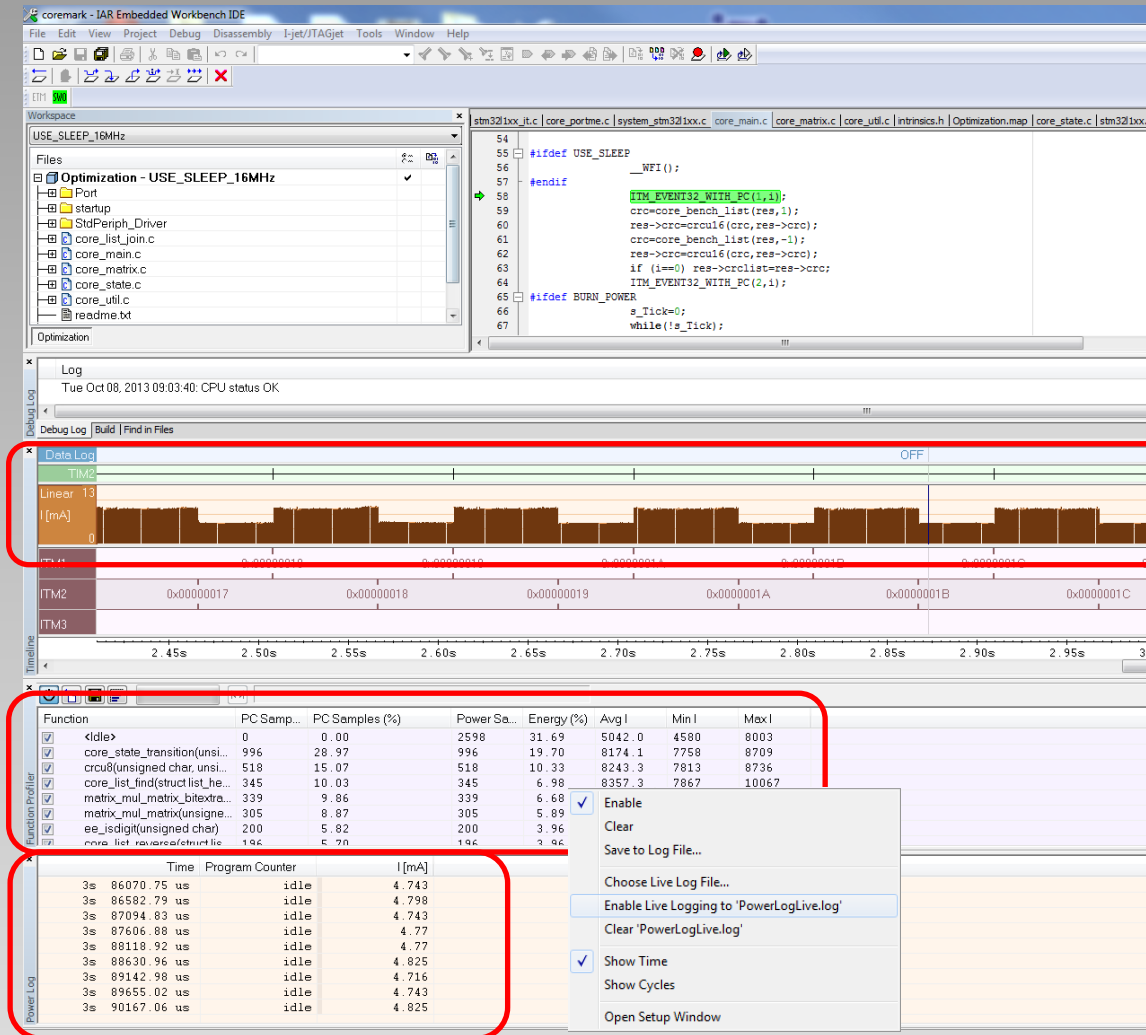
- In-circuit power measurement probe from IAR systems
- Used in combination with I-jet
- V1 – V3 voltage range 0 to 6V
- I+ and I- differential voltage, 110mV full scale
 - Current measurement over shunt:
 - 1.0 Ω (for 110 mA max, ~100uA resolution)
 - 10 Ω (for 11 mA max, ~10uA resolution)
- Sampling rate up to 200 kHz
- 12 bit resolution (\rightarrow 1.5mV common mode, 27 μ V diff.)



Power Debugging Setup



Visualizing current consumption



- In the timeline, together with other parameters
- In the function profiler
- As log optionally live log

I-jet Trace for Cortex-M

- USB powered
 - optional external power supply for higher target power
- USB 3.0 high-speed interface
 - USB 2.0 compatible
- Buffers up to 32,000,000 instructions
- Interface to I-scope



ETM Trace features

Instruction
Profiling



Go to	Memory	
Disassembly		
408	0x800295e: 0xe003	B.N ??core_list_revers...
	tmp=list->next;	
12240	0x8002960: 0x6802	LDR R2, [R0]
	list->next=next;	
12240	0x8002962: 0x6001	STR R1, [R0]
	next=list;	
12240	0x8002964: 0x0001	MOVS R1, R0
	list=tmp;	
12240	0x8002966: 0x0010	MOVS R0, R2

Function
Trace



#	Cycles	Address	Trace	Exec	Exc...	Access	Data
1797339	4008946	0x08003010	core_state_transition(unsig...	Thumb			
1797404	4009048	0x08002f04	ee_isdigit(unsigned char)	Thumb			
1797415	4009062	0x08003044	core_state_transition(unsig...	Thumb			
1797450	4009116	0x08002f04	ee_isdigit(unsigned char)	Thumb			
1797461	4009130	0x08003044	core_state_transition(unsig...	Thumb			
1797496	4009184	0x08002f04	ee_isdigit(unsigned char)	Thumb			
1797507	4009198	0x08003044	core_state_transition(unsig...	Thumb			
1797542	4009256	0x08002f04	ee_isdigit(unsigned char)	Thumb			

ETM Trace



#	Cycles	Address	Trace	Exec	Exc...	Access	Data Address	Data Value	Comment
541910	-	0x08002966	MOVS R0, R2	Thu...					
			while (list) {						
541911	-	0x08002968	CMP R0, #0	Thu...					
541912	-	0x0800296a	BNE.N ??core_list_reverse_1 ...	Thu...					
			tmp=list->next;						
541913	1353684	0x08002960	LDR R2, [R0]	Thu...					
			list->next=next;						
541914	-	0x08002962	STR R1, [R0]	Thu...					
			next=list;						

Function
Profiler



Function	Calls	Flat Time	Flat Time (%)	Acc. Time	Acc. Time (%)
main(int, char **)	2	4916848	52.78	6002178	64.43
core_state_transition(unsigned char *, unsigned int *)	4096	1117004	11.99	1244306	13.36
<Other>	0	806980	8.66	5063872	54.35
crcu8(unsigned char, unsigned short)	2149	483124	5.19	483124	5.19
matrix_mul_matrix_bitextract(unsigned int int *, signed short *, signed short *)	16	371514	3.99	371514	3.99
matrix_mul_matrix(unsigned int int *, signed short *, signed short *)	16	336670	3.61	336670	3.61
core_list_find(struct list_head s *, struct list_data s *)	221	307014	3.30	307014	3.30

Code
Coverage



Optimization 47.50%
core_list_join 74.21%
calc_func(signed short *, struct RESULTS_S *) 91.67%
cmp_complex(struct list_data_s *, struct list_data_s *, struct RESULTS_S *) 100.00%
cmp_idx(struct list_data_s *, struct list_data_s *, struct RESULTS_S *) 100.00%
copy_info(struct list_data_s *, struct list_data_s *) 0.00%
core_bench_list(struct RESULTS_S *, signed short) 100.00%
core_list_find(struct list_head_s *, struct list_data_s *) 100.00%
core_list_init(unsigned int, struct list_head_s *, signed short) 0.00%

- Overview
- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy

LET US AGREE ON TWO FACTS

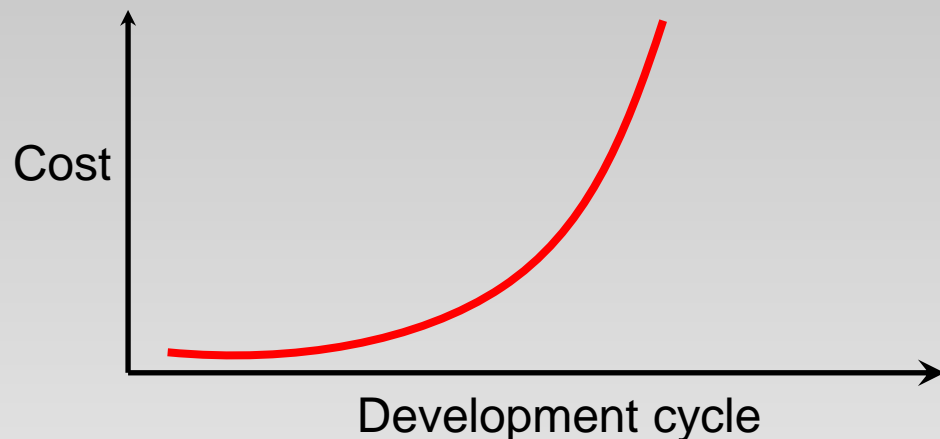
1. All software contains bugs

```

136 printf("[*] Android local root exploit (C) The Android Exploit Crew\n");
137
138 basedir = "/sqlite_stat_journals";
139 if (chdir(basedir) < 0) {
140     basedir = "/data/local/tmp";
141     if (chdir(basedir) < 0)
142         basedir = strdup(getcwd(buf, sizeof(buf)));
143 }
144 printf("basedir=%s, path=%s\n", basedir, path);
145 printf("[+] Opening NETLINK_KOBBEL socket\n");
146
147 memset(&snl, 0, sizeof(snl));
148 snl.nl_pid = 1;
149 snl.nl_family = AF_NETLINK;
150
151 if ((sock = socket(PF_NETLINK, SOCK_DGRAM, NETLINK_KOBBEL)) < 0)
152     die("[+] socket");
153
154 close(creat("load", 0644));
155 if ((ofd = creat("load", 0644)) < 0)
156     die("[+] creat");
157 if (write(ofd, path, strlen(path)) < 0)
158     die("[+] write");

```

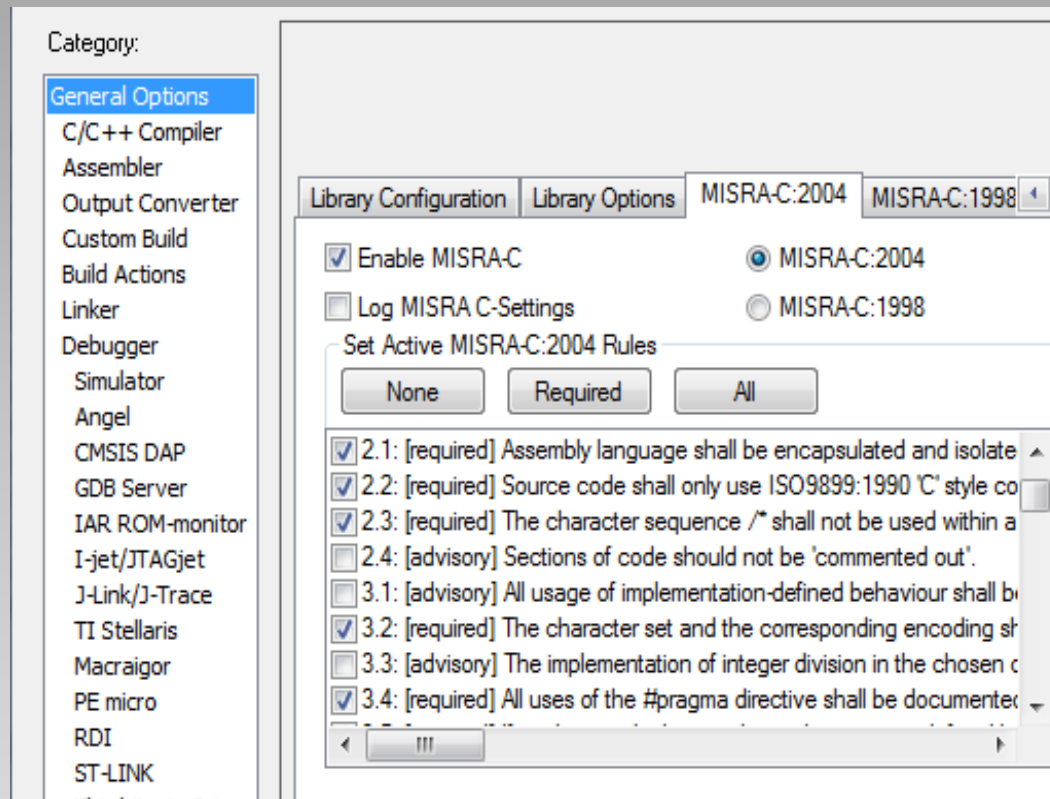
2. The later you find a bug, the more expensive it gets



MISRA C stands for the Motor Industry Software Reliability Association, a consortium based out of Cambridge in the UK that promotes standards to improve the safety and reliability of embedded code. More importantly, MISRA-C:

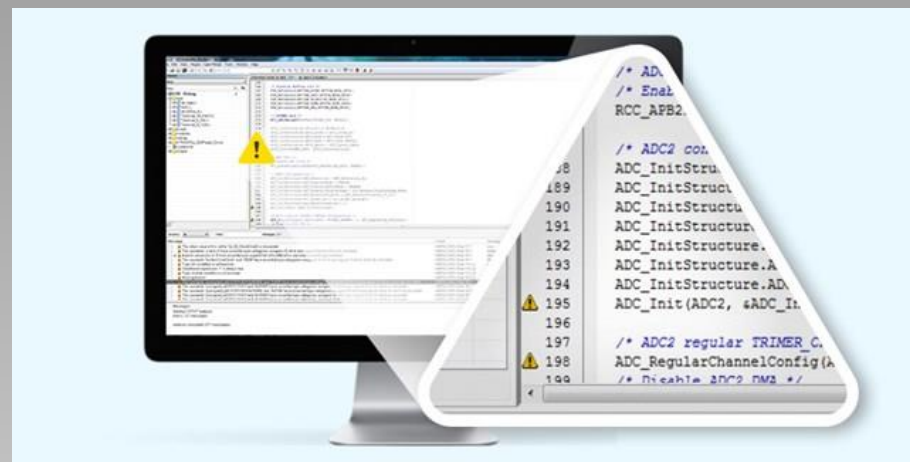
- Set of ~140 coding rules
- Aims to facilitate code safety, portability and reliability
- Must be used in some cases
- Often good to use even when not mandatory
- Recommended in IEC61508
- Has evolved as a widely accepted model for best practices by leading developers in various sectors

- MISRA-C:2004 Checker included in commercial license

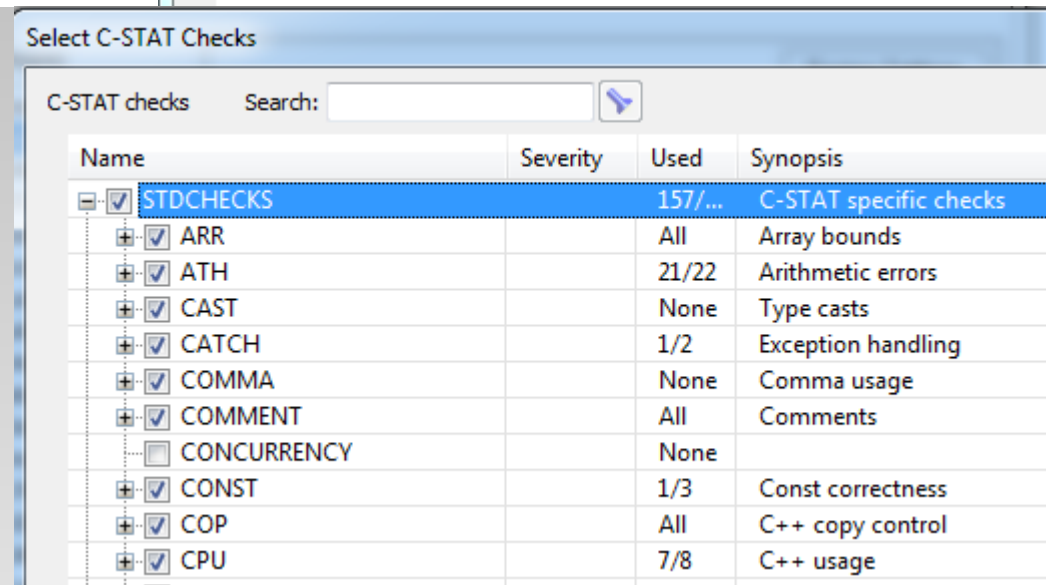
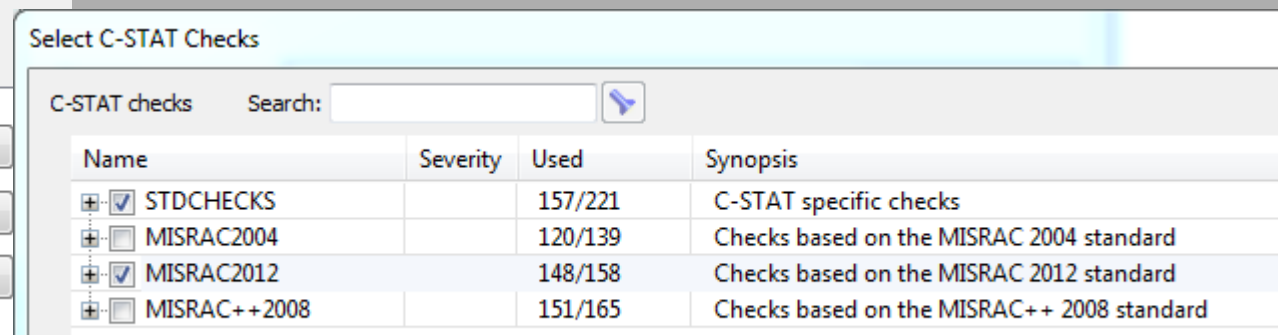
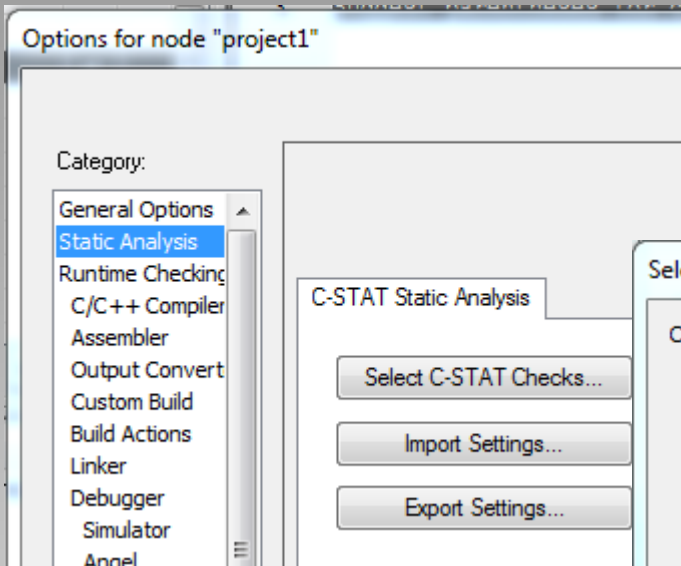


Static Analysis C-STAT

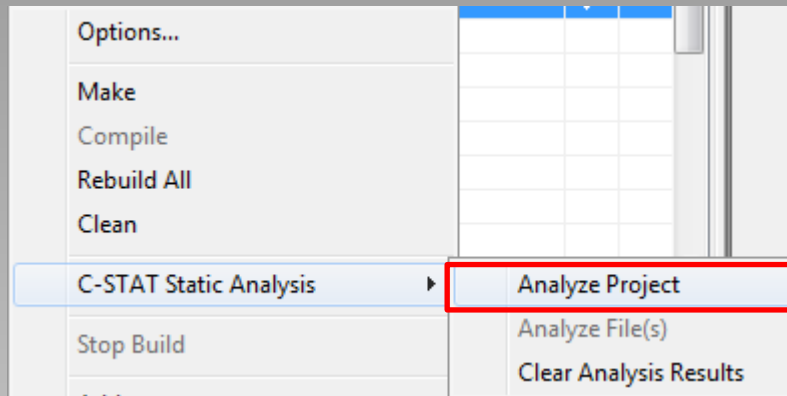
- Coding rules
 - MISRA C: 2004
 - MISRA C: 2012
 - MISRA C++: 2008
 - >200 additional checks based on CERT, CWE etc.
- Export/import of selected checks
- Full integration in IAR Embedded Workbench
- Can also be used through Eclipse plugin or from command line



Static Analysis C-STAT (cont'd)



Static Analysis C-STAT (cont'd)



C-STAT Messages				
Severity: Medium/High Filter: Messages: 18 / 52				
Message	Check	Severity	File	Line
Tutor.c (8 messages)			Tutor.c	
Implicit conversion of 'my_uint32' from essential type unsigned 32-bit int to diff...	MISRAC2012-Rule-10.3	Medium	Tutor.c	64
main			Tutor.c	64
narrow_or_different_essential_type			Tutor.c	64
Variable 'my_uint32' is uninitialized	SPC-uninit-var-all	High	Tutor.c	64
Variable 'my_uint32' is uninitialized	MISRAC2012-Rule-9.1_e	High	Tutor.c	64
Suggest parentheses around '3*2' for clarity	MISRAC2012-Rule-12.1	Medium	Tutor.c	65
The operands 'i' and '10' have essential type categories unsigned 32-bit int ...	MISRAC2012-Rule-10.4	Medium	Tutor.c	70
The operands 'i' and '5' have essential type categories unsigned 32-bit int a...	MISRAC2012-Rule-10.4	Medium	Tutor.c	74
Possible division by 0. Divisor has potential range [0,9]	MISRAC2012-Rule-1.3_f	High	Tutor.c	82
Possible division by 0. Divisor has potential range [0,9]	ATH-div-0-pos	High	Tutor.c	82
Utilities.c (10 messages)			Utilities.c	

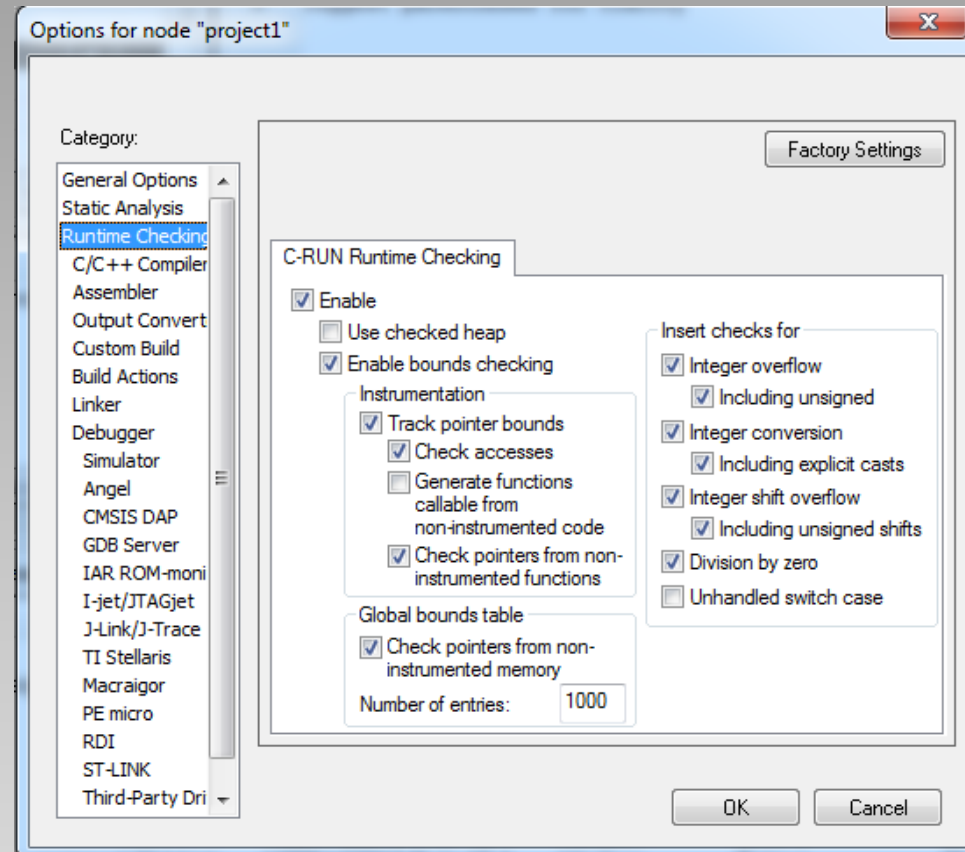
Runtime Analysis C-RUN

- Detection of data manipulation issues during runtime
- Arithmetic, bound and heap checking
- Very efficient instrumentation of compiled code



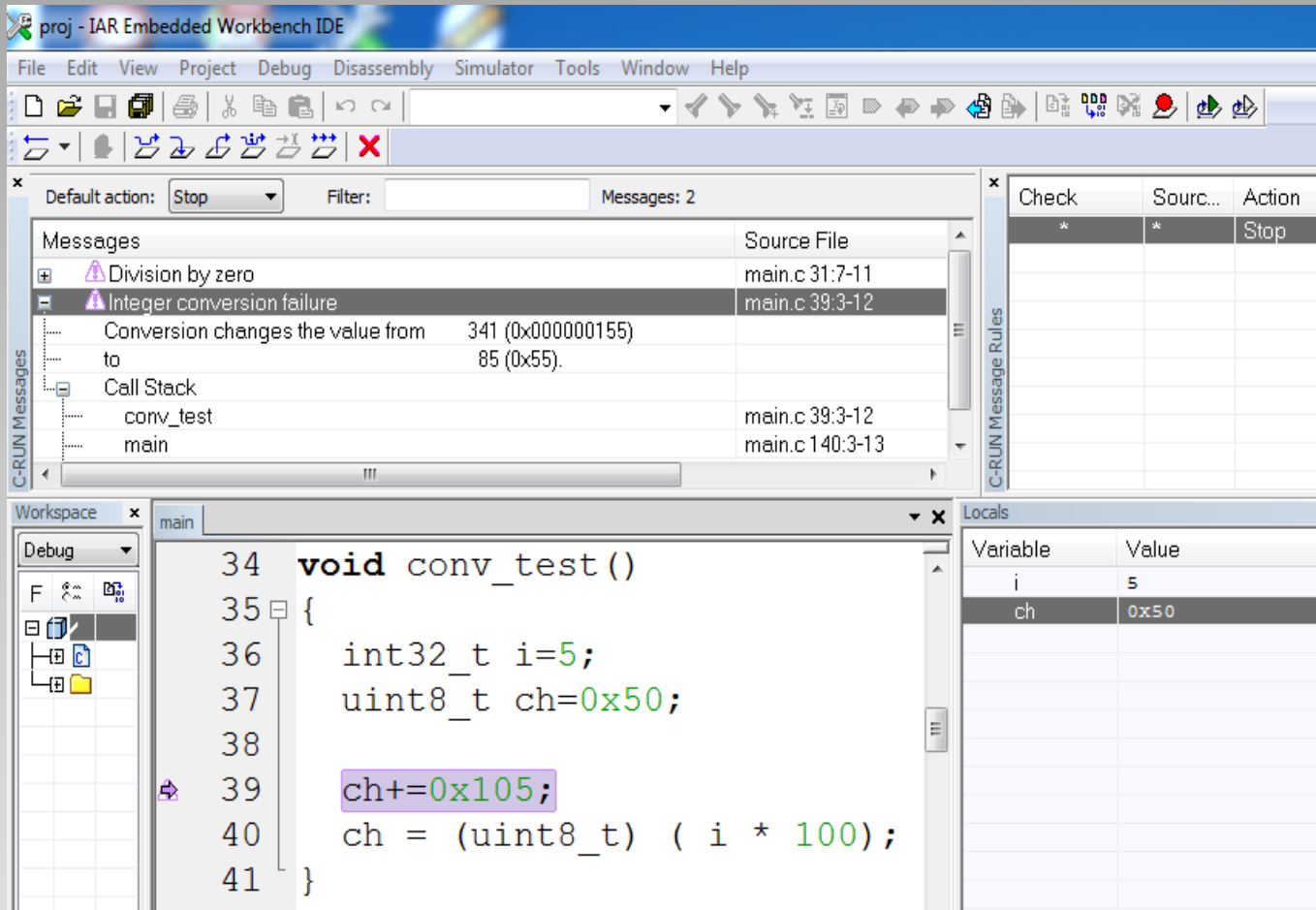
Runtime Analysis C-RUN, cont'd

- Individual checks can be turned on/off
- Overhead depends on
 - Performed checks
 - Application profile
- Possible to configure on module basis



Runtime Analysis C-RUN, cont'd

- Very detailed feedback on what went wrong



The screenshot shows the IAR Embedded Workbench IDE with the C-RUN runtime analysis tool. The Messages window displays two errors: "Division by zero" and "Integer conversion failure". The "Integer conversion failure" message is expanded, showing a conversion from 341 (0x000000155) to 85 (0x55). The Call Stack shows the error occurred in the `conv_test` function, which is currently selected in the Workspace. The Locals window shows the current state of variables `i` (5) and `ch` (0x50).

Messages

Check	Sourc...	Action
*	*	Stop

Call Stack

Function	Source File
conv_test	main.c 39:3-12
main	main.c 140:3-13

Workspace

```

34 void conv_test()
35 {
36     int32_t i=5;
37     uint8_t ch=0x50;
38
39     ch+=0x105;
40     ch = (uint8_t) ( i * 100);
41 }
  
```

Locals

Variable	Value
i	5
ch	0x50

Validation of Compliance

- Document describes test methods at IAR Systems
- provided on request free of charge for required release
- Very useful for certification purposes

IAR Systems External	Document number: 21559	Page 1(6)
Prepared By Anders Lundgren	Date 2014-03-03	Revision 1

IAR Embedded Workbench for ARM 7.10 Statement of standards compliance

Validation of compliance document

Contents

1	Introduction	2
1.1	Revision history	2
2	Compiler package.....	2
3	Compiler target platforms.....	2
4	Domain of use	2
5	IAR Quality management system (QMS)	2
6	Acceptance testing.....	3
6.1	Test suites	3
6.2	Test engine.....	5
6.3	Testing during development.....	5
6.4	Product release acceptance.....	5
7	Handling of field reported issues	5
8	End user product documentation	6
9	Product update and maintenance history	6

EWARM – TÜV certified tool chain

ZERTIFIKAT ♦ CERTIFICATE ♦ CERTIFICADO ♦ CERTIFICAT

CERTIFICATE
No. Z10 13 04 84282 001

Holder of Certificate: IAR Systems AB
Strandbodgatan 1
750 23 Uppsala
SWEDEN

Factory(ies): 84282

Certification Mark: 

Product: Software Tool for Safety Related Development

Model(s): IAR Embedded Workbench for ARM
- Build Tool Chain


Parameters: The development tool chain fulfills the requirements for development tools classified T3 according to IEC 61508-3. The build tool chain is qualified to be used in safety-related software development according to IEC 61508 and ISO 26262. The report IU84911C is a mandatory part of this certificate.


Tested according to: IEC 61508-3:2010
ISO 26262-8:2011

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: IU84911C


Date, 2013-04-04
Page 1 of 1


(Guido Neumann)



TÜV SÜD Product Service GmbH · Zertifizierstelle · Ridlerstraße 65 · 80339 München · Germany

TÜV



Report
to the
Certificate
Z10 13 04 84282 001

IAR Embedded Workbench for ARM
build tool chain

Manufacturer:
IAR Systems AB
Strandbodgatan 1
Uppsala
Sweden

Report no. IU84911C
Revision 1.0 of April 8th, 2013

Test and Certification Body
TÜV SÜD Rail GmbH
Generic Safety Systems
D-80339 Munich

page 1 of 10

Dissemination, distribution, copying or any other use of any information in this report in part is strictly prohibited.

- Overview
- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy
- Hands-on & Demo

- Efficient programming & Advanced debugging (2 days)
 - Compiler technology
 - Coding techniques
 - Best practices
 - Linking applications
 - Safety and standards
 - Advanced debugging
 - Breakpoints
 - SWO/ITM
 - Using trace
- Customized Training
 - On Request



Details and registration:
www.iar.com/academy

Now let's practice!
