IAR Embedded Workbench for Infineon XMC

Martin Gisbert
FAE IAR Systems

IAR SYSTEMS

Agenda

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

Agenda

- 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

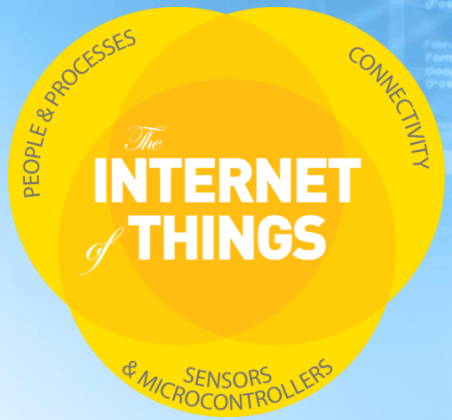
24 hour technical support in

13 languages

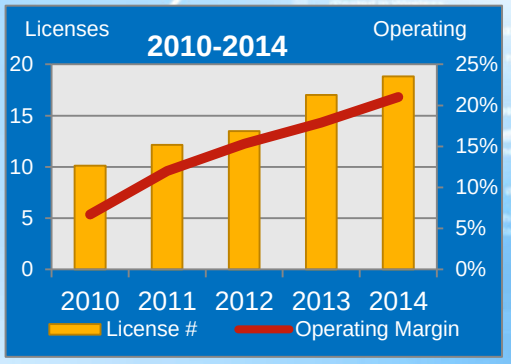
Uppsala
Munich
Sao Paulo
Tokyo
Seoul
Shanghai
London
Paris
San Francisco
Dallas
Boston
Los Angeles

+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

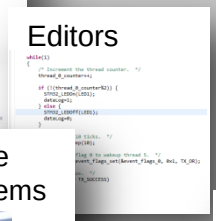
- Simulator driver
- Hardware system drivers
- Power debugging
- RTOS plug-ins
- C-SPY Macros

Plug-ins

IDE

eclipse READY

Editors



Source code control systems

SUBVERSION

Project Connections

Configuration tools

Infineon DAVE™

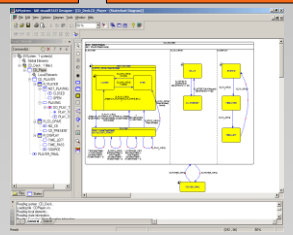
TÜV SUD Safety tested Production monitored Functional Safety

Validation Reports

MISRA

C-STAT

C-RUN



visualSTATE

Support

Strong ecosystem

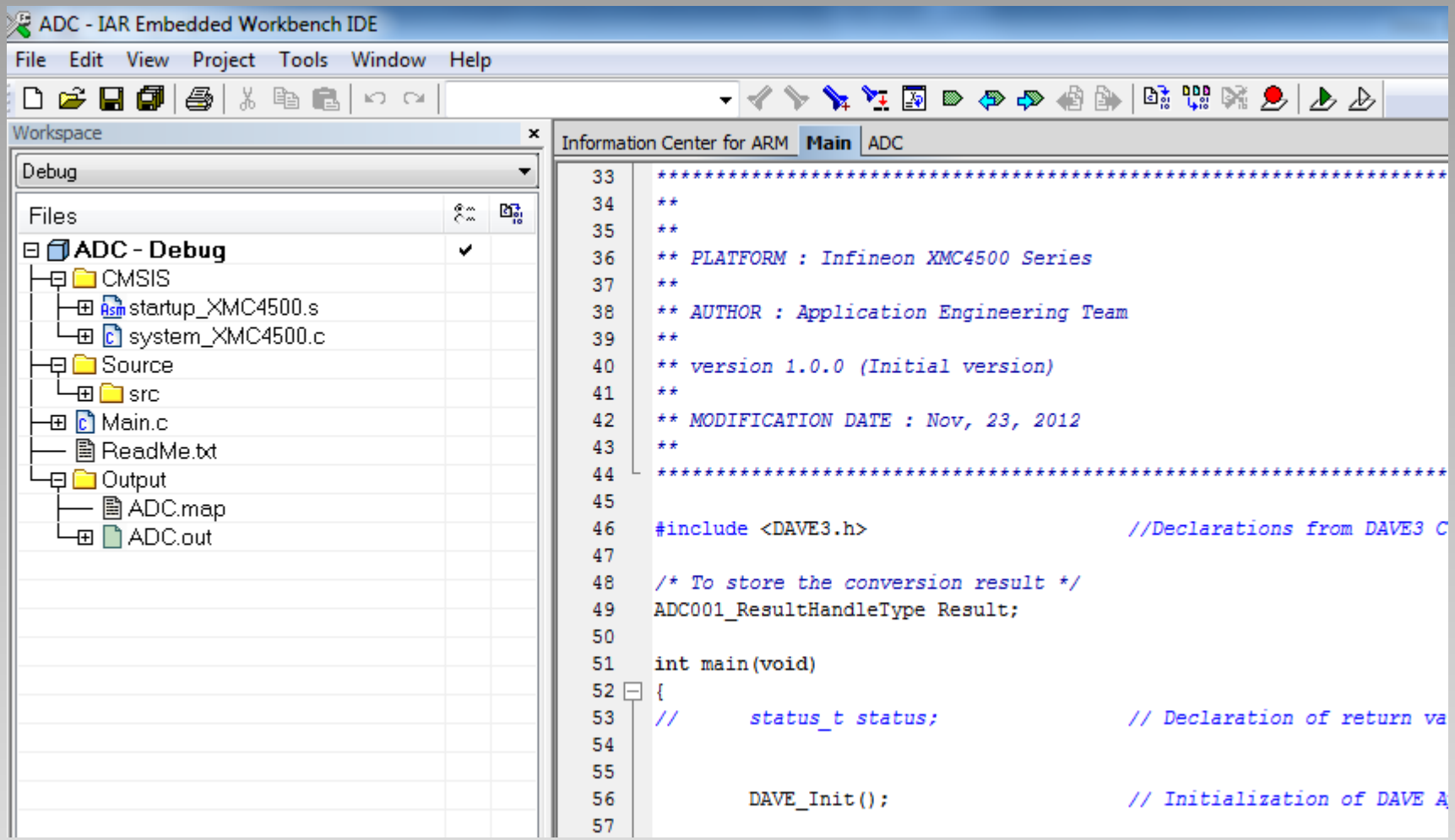
IAR ACADEMY

Training

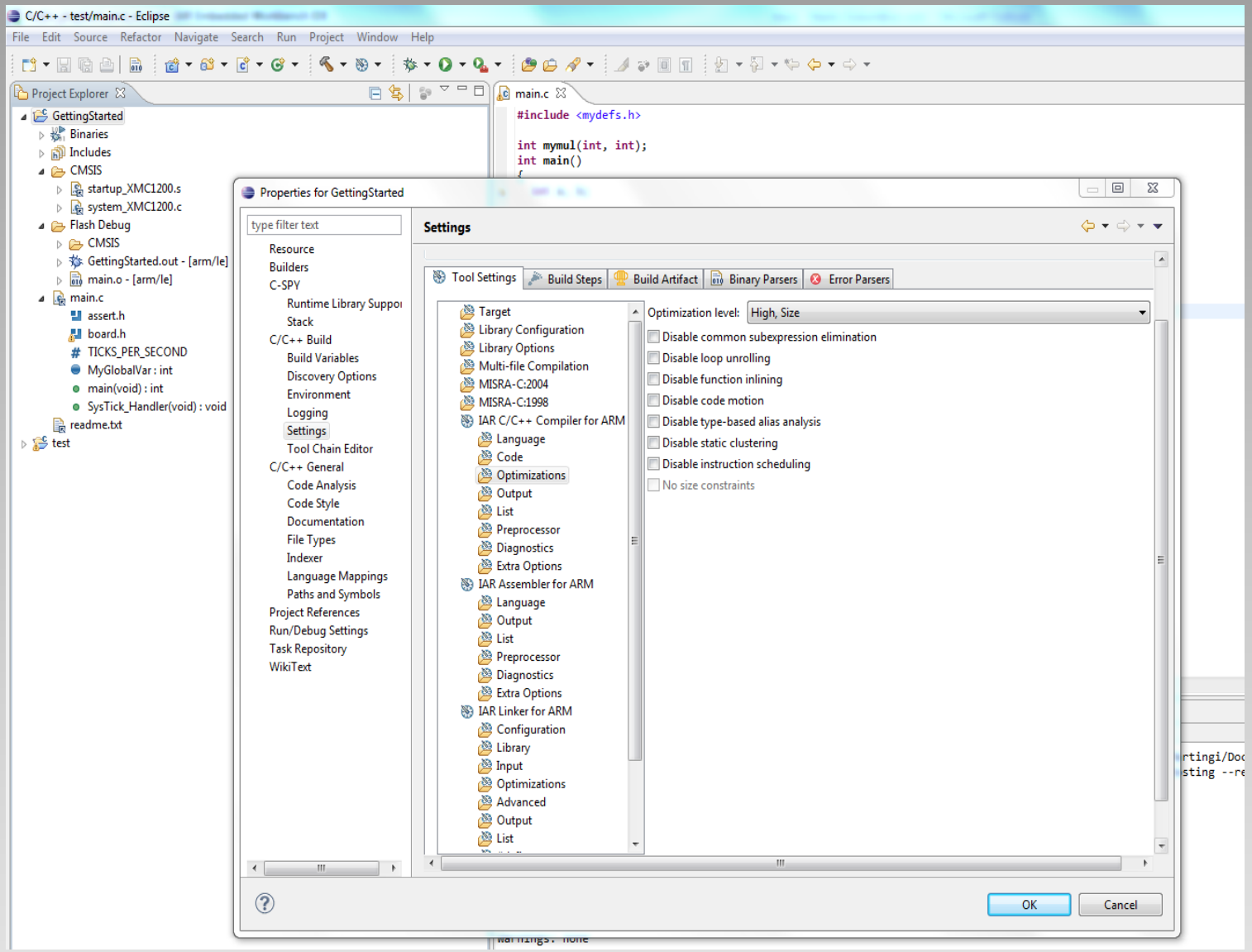
Agenda

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

IAR Embedded Workbench IDE



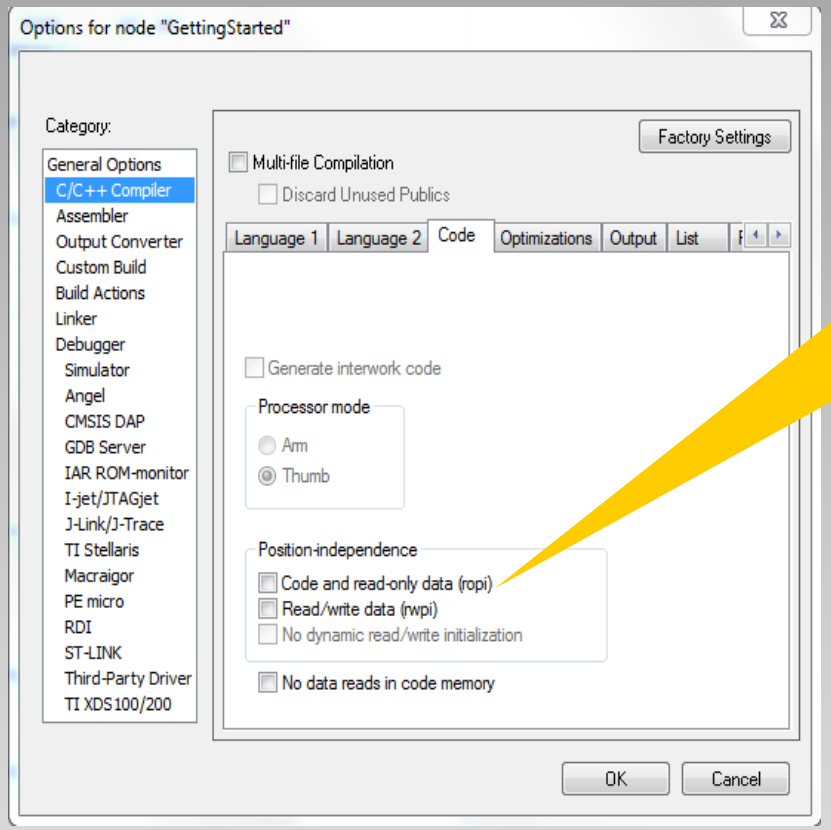
Eclipse Plugin (Code View)



Agenda

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

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



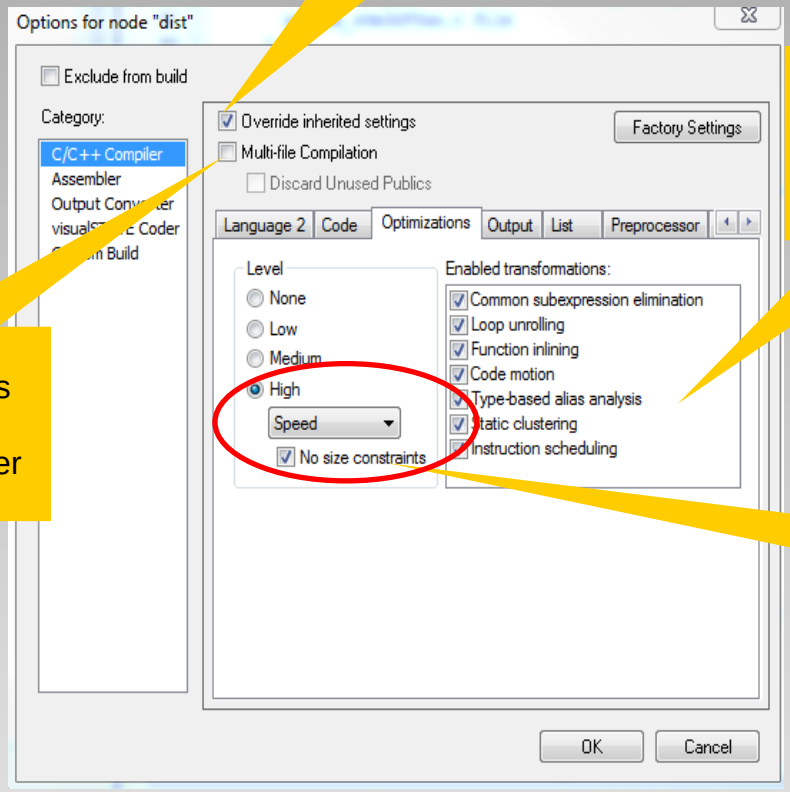
Support for position independent code and data

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



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

Major functions of the optimizer can be controlled individually

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

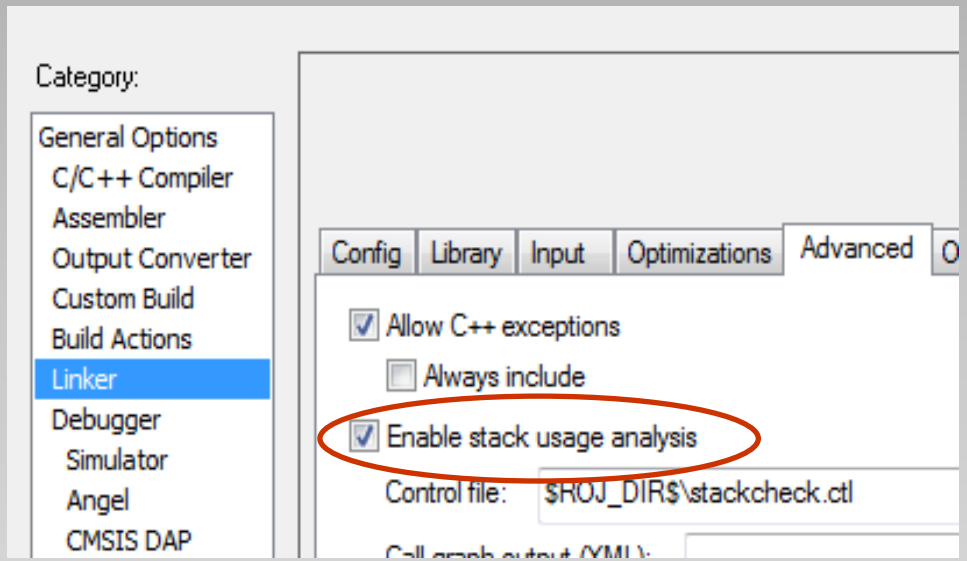
- ISO/IEC 9899:1990 (C94/C90/C89/ANSI C)
- ISO/IEC 9899:1999 (C99/Standard C)
- ISO/IEC 1488:2003 (Standard C++)
- Embedded C++ and Extended Embedded C++ dialects are also supported

Agenda

- 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

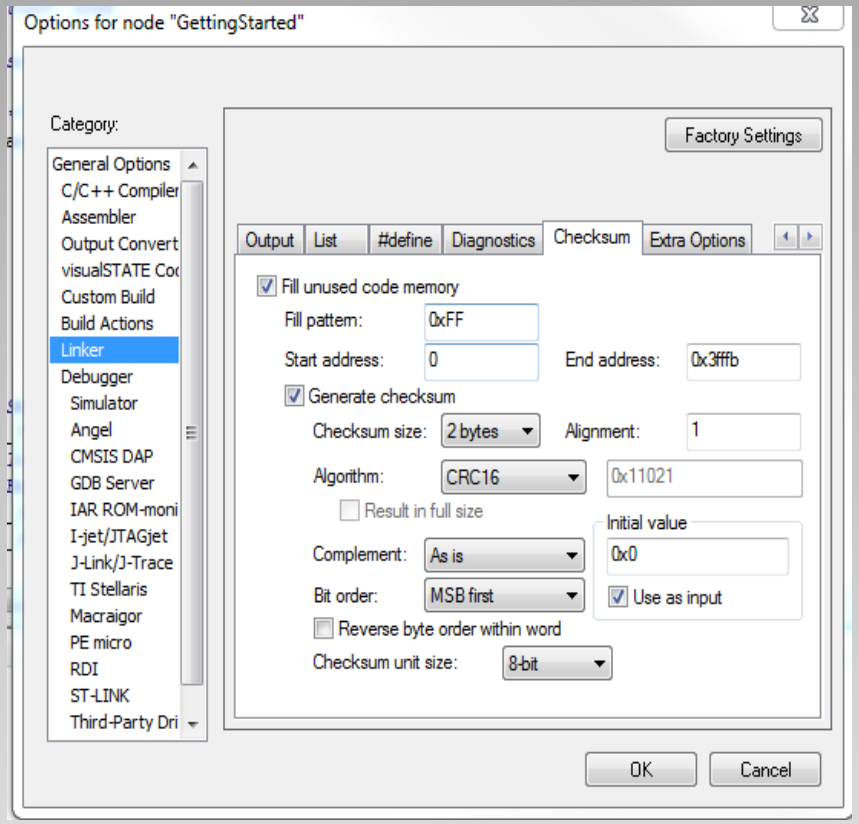
- Stack usage chapter in the linker map file, listing of maximum stack depth for each call graph root.
- Alternative generation of a .xml call graph file
- Use linker extra option `--log call_graph` for better understanding

```
****************
*** 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



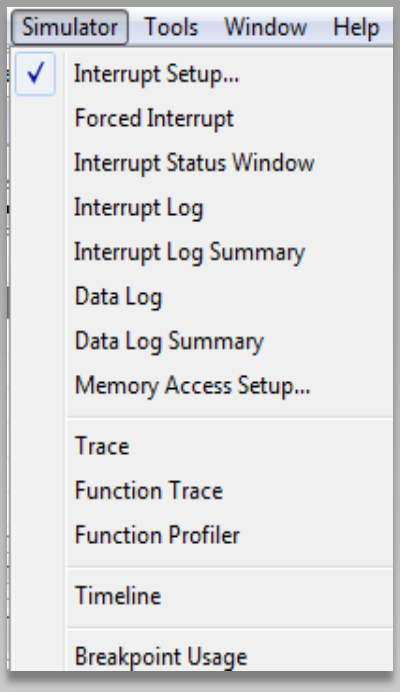
Agenda

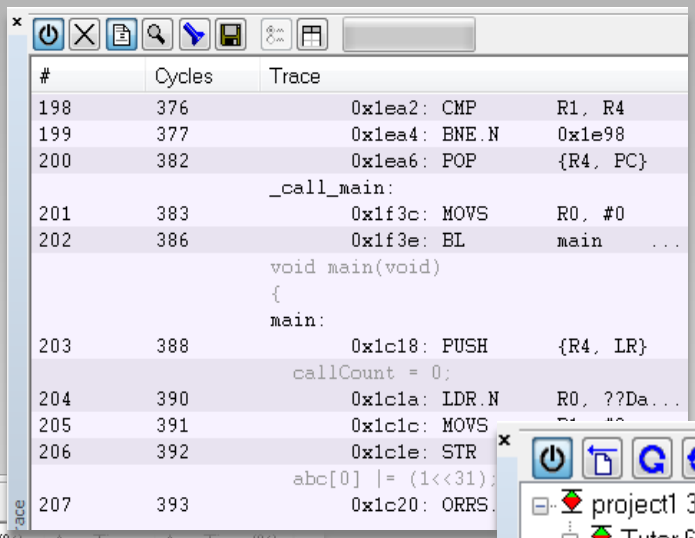
- Overview
- IAR Embedded Workbench IDE vs. Eclipse
- Compiler
- Linker
- Debugger
- Safety
- IAR Academy
- Hands-on & Demo

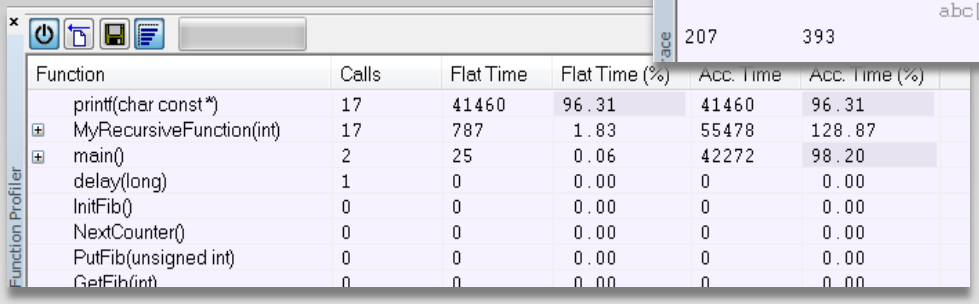
C-SPY Simulator

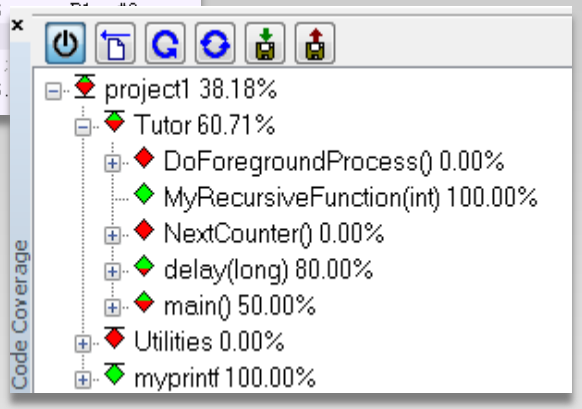
In contrast to the debugger with bandwidth-limited communication interface, the simulator does not miss anything and can show

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





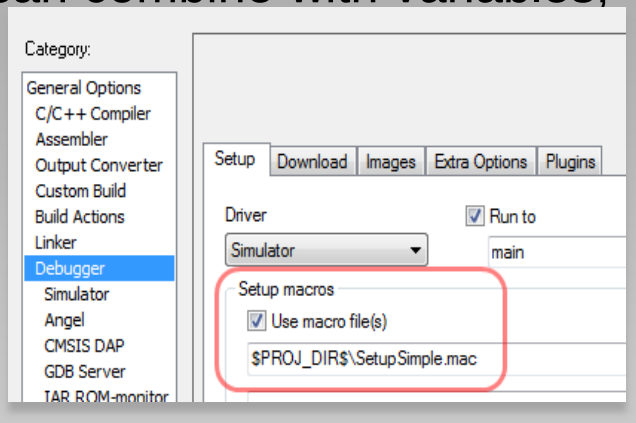




C-SPY Macros

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

There are several predefine 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**

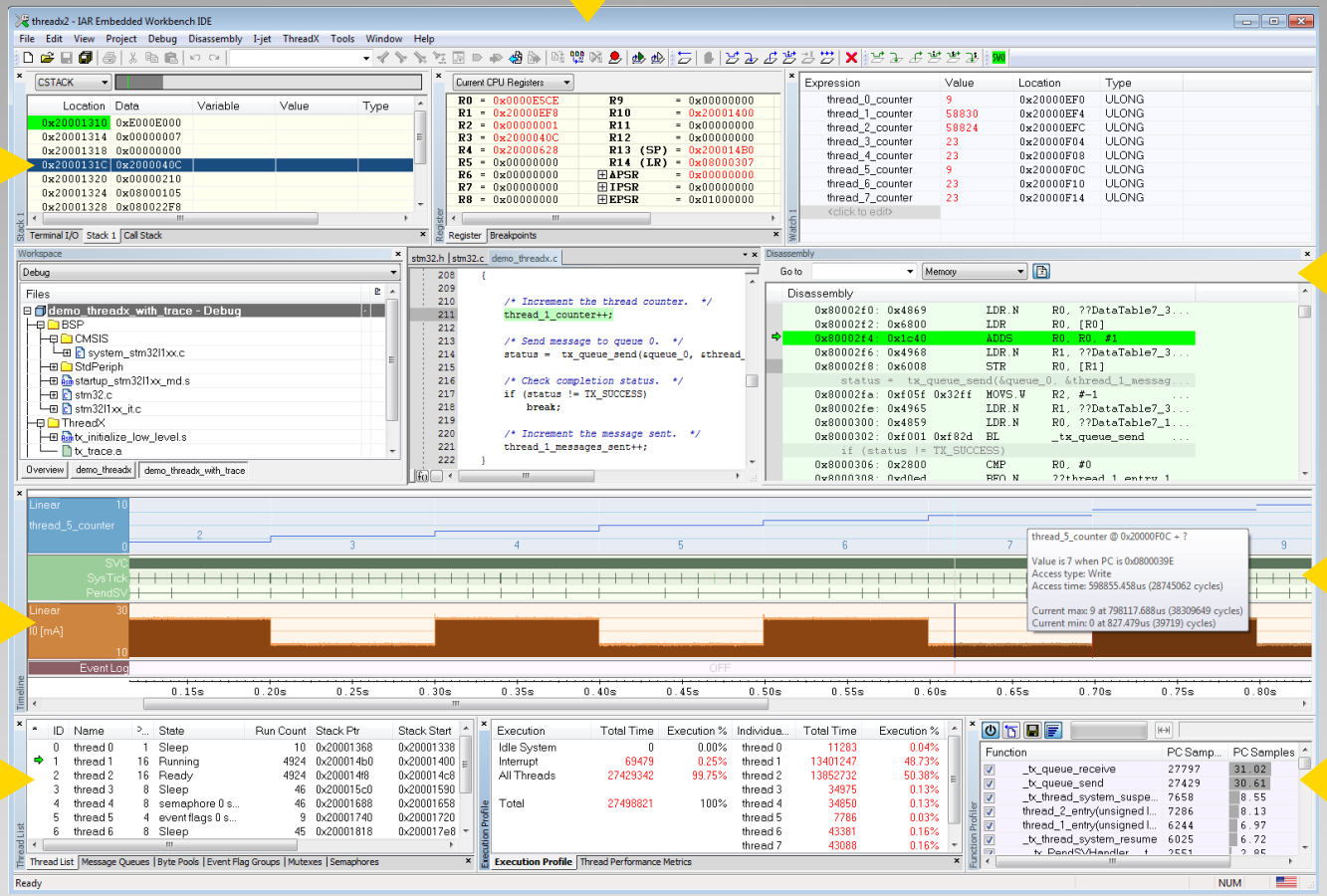
Comprehensive debugger

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

Power vizualization

Timeline window

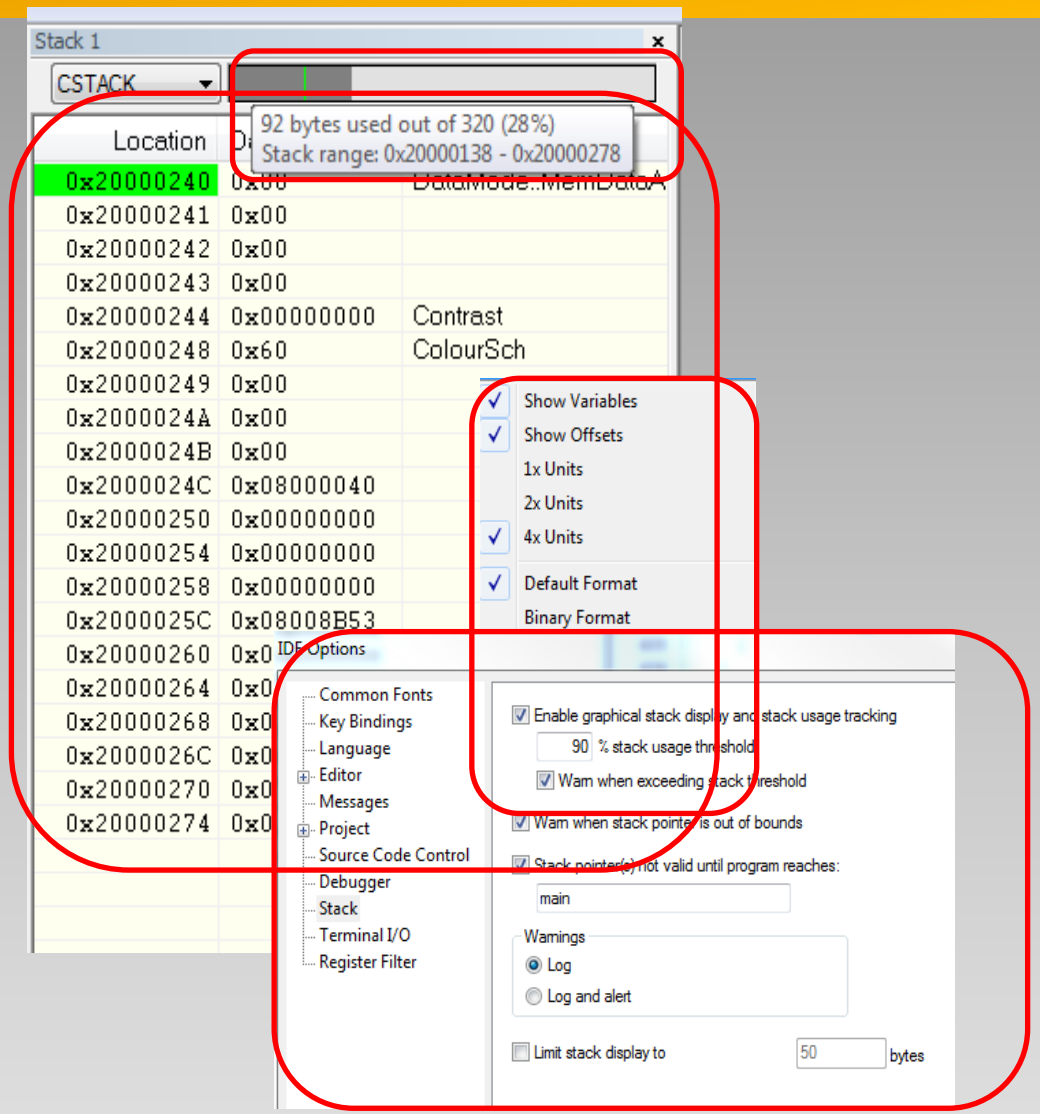
RTOS awareness

Performance analysis

Stack view

Displays of:

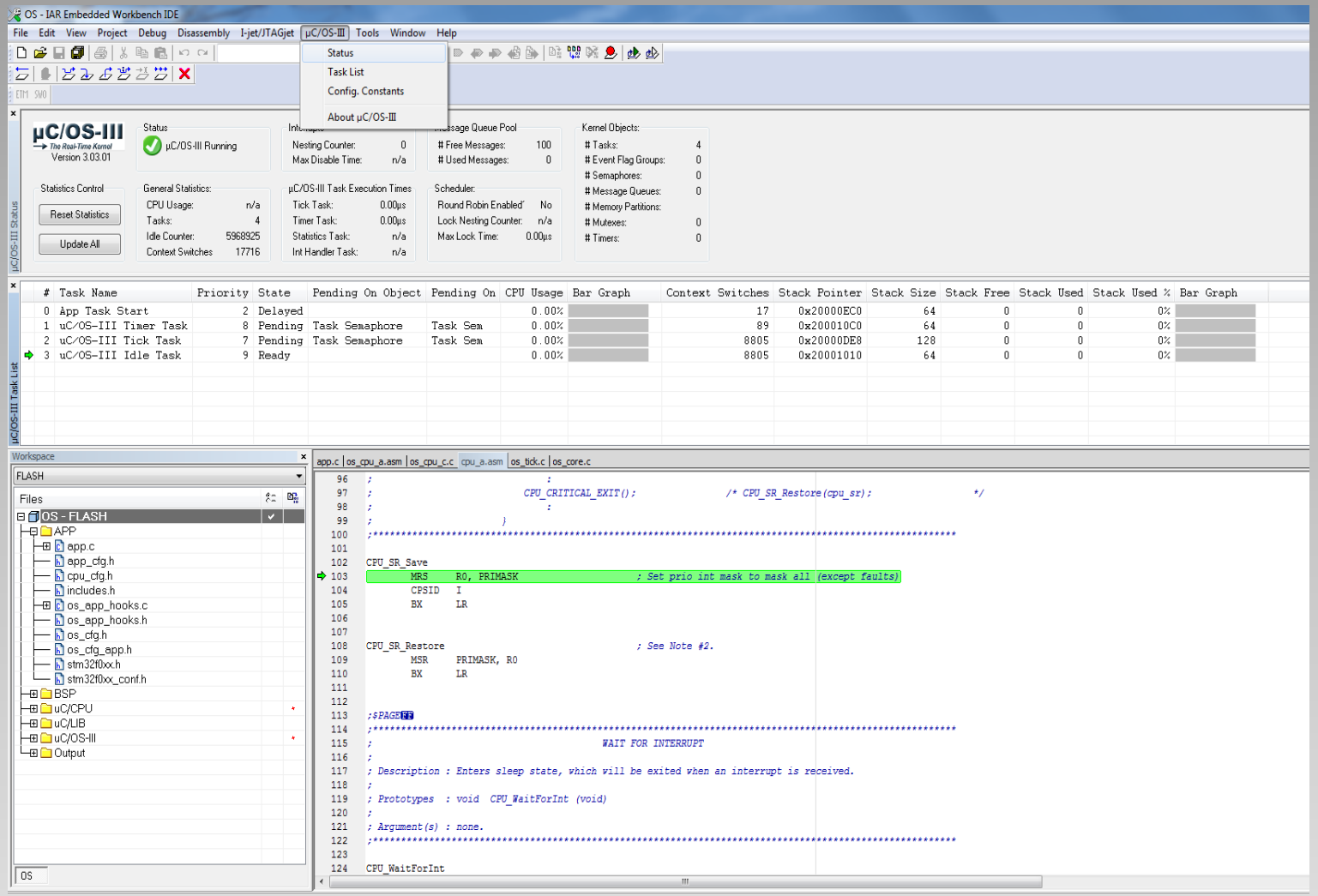
- Current content of stack
- Current depth of stack
- Max. depth of stack while the target was running
- Optional with/without variables or offset
- Optional with threshold warning and limited stack display



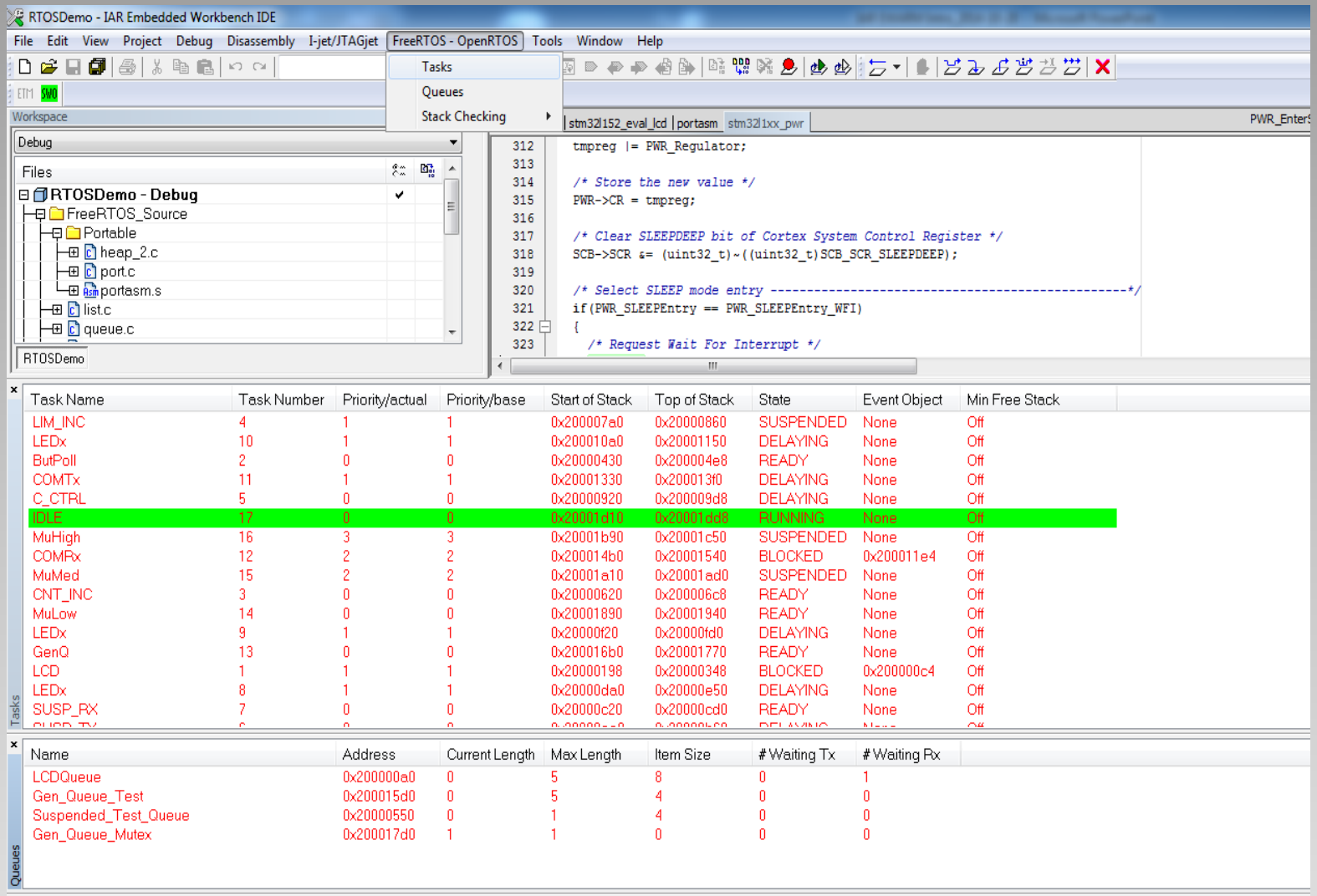
C-SPY Plugin: RTOS

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

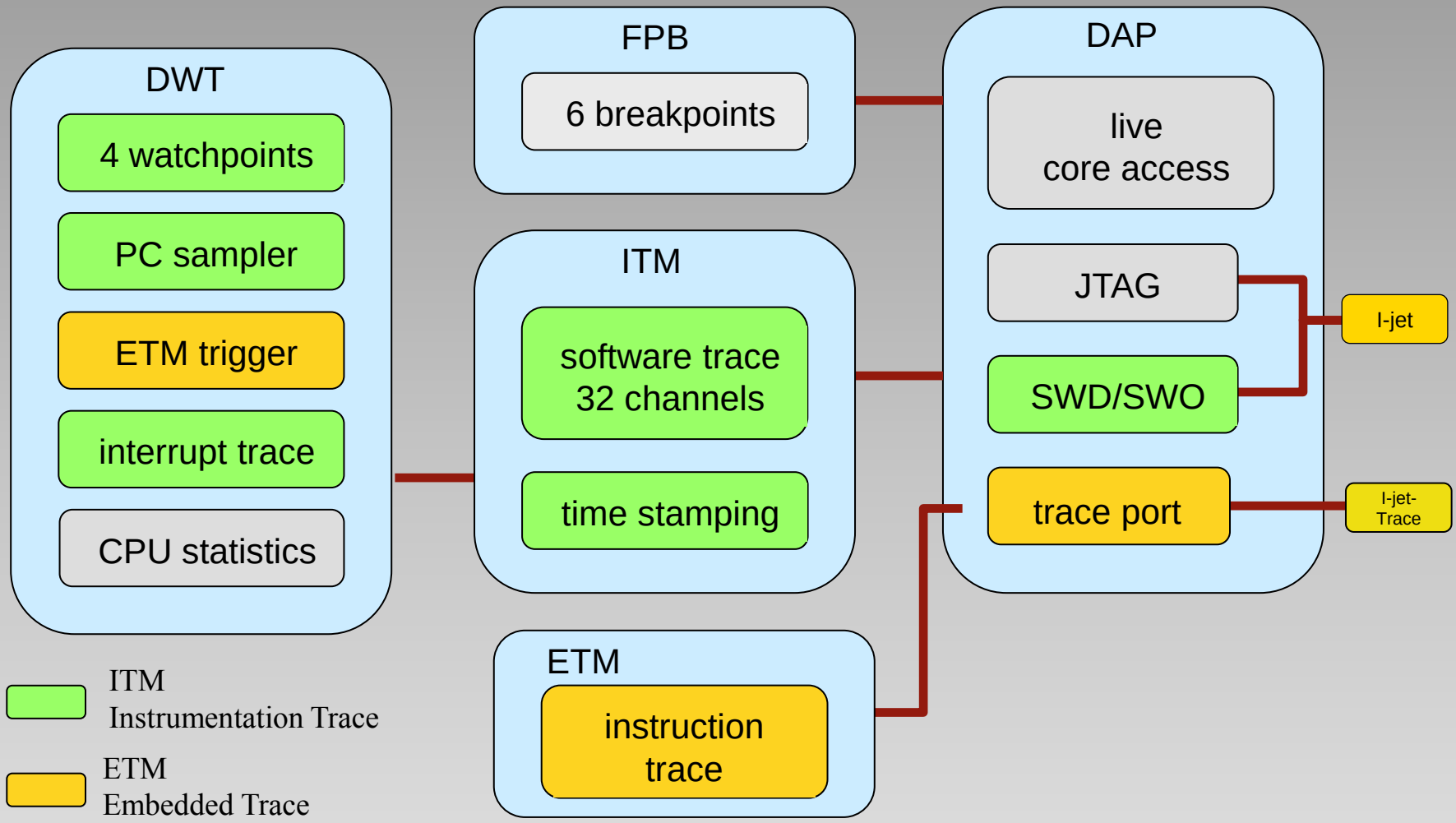
RTOS Plugins (Micrium µC/OS-III)



RTOS Plugins (FreeRTOS)



CoreSight Overview (ARM Cortex-M3/M4)

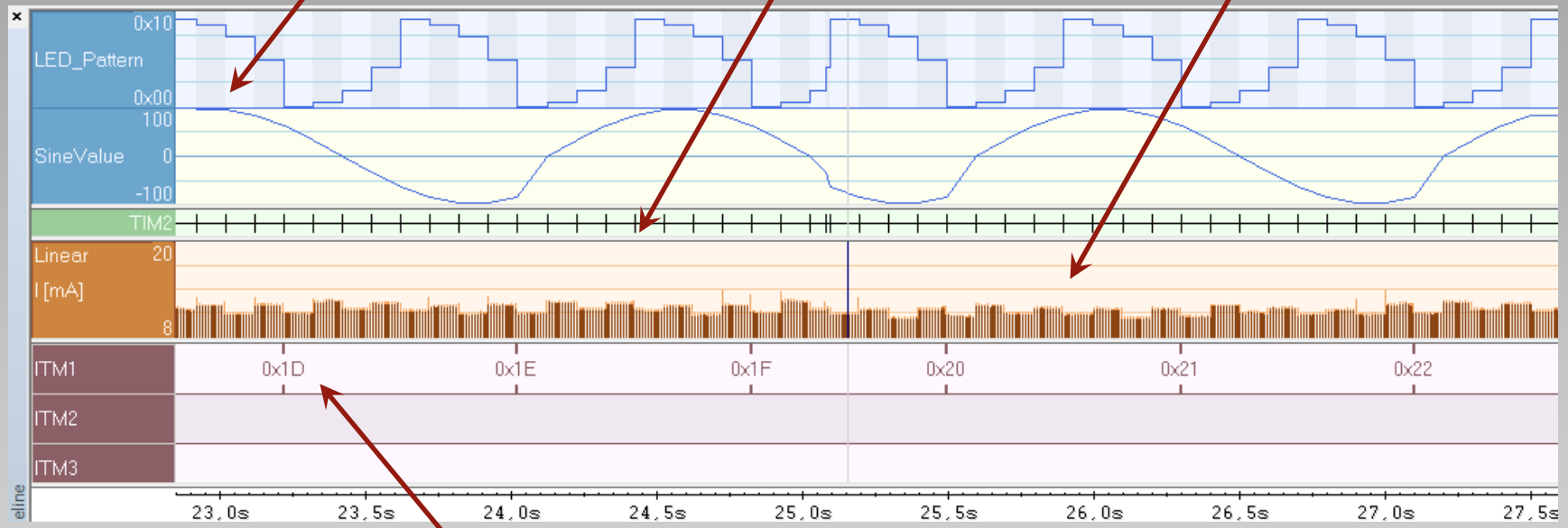


ITM/SWO Usage (example)

Data log window

Interrupt log window

Power debugging



ITM event with line graph

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

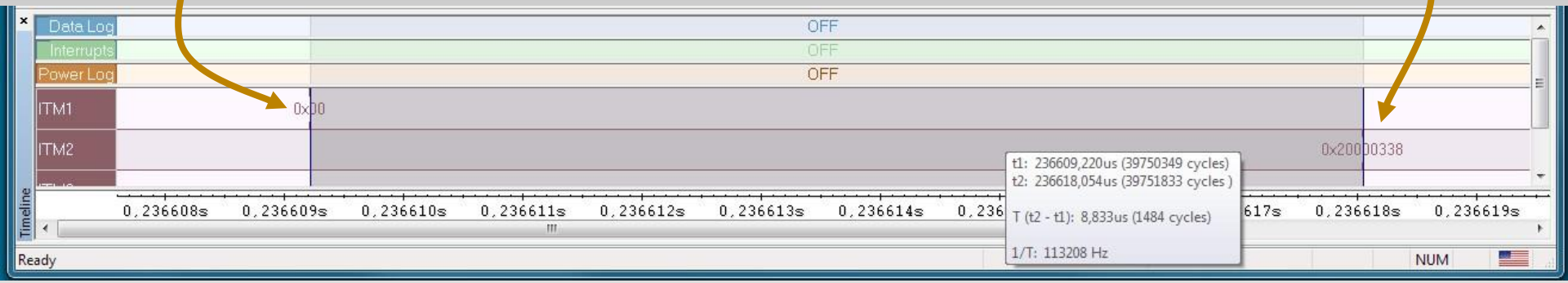
ITM events

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

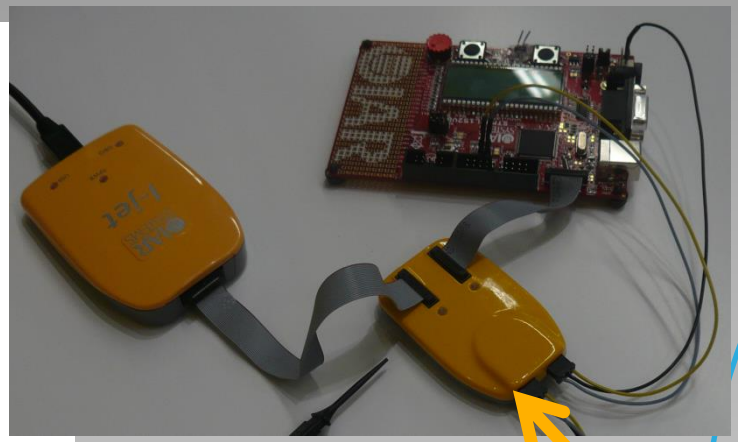
- ITM_EVENT8(channel, value)
- ITM_EVENT16(channel, value)
- ITM_EVENT32(channel, value)
- ITM_EVENT8_WITH_PC(channel, value)
- ITM_EVENT16_WITH_PC(channel, value)
- ITM_EVENT32_WITH_PC(channel, value)

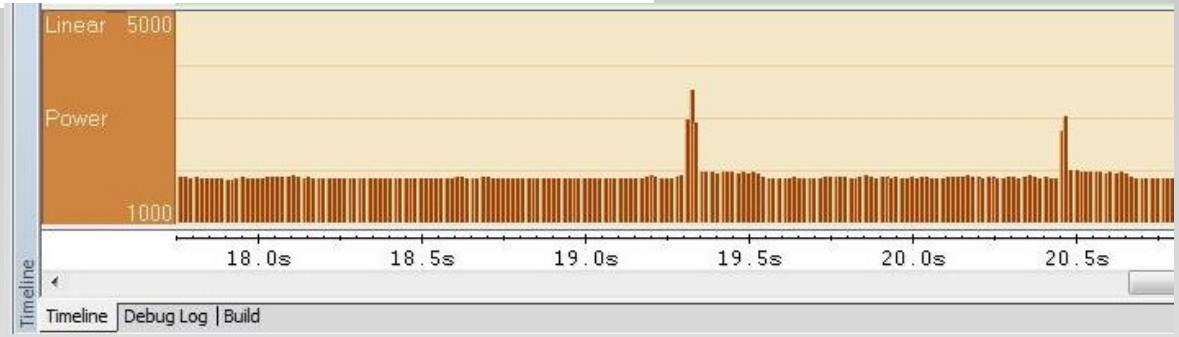
- To use, simply include a macro in your code and decide what channel and what value to send
- During debugging, open the timeline window to display the data

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



Power Debugging

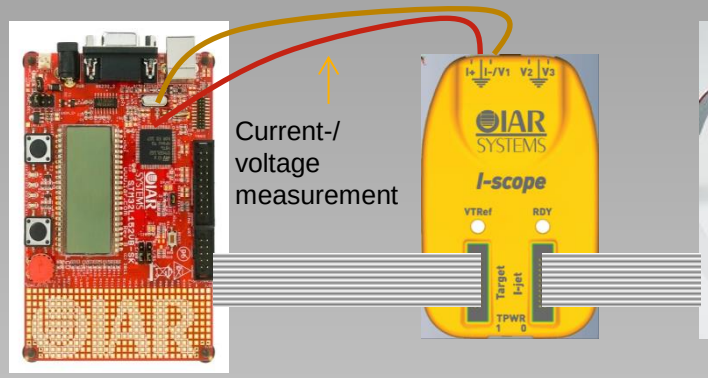


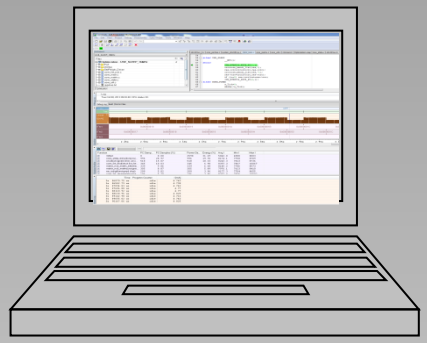


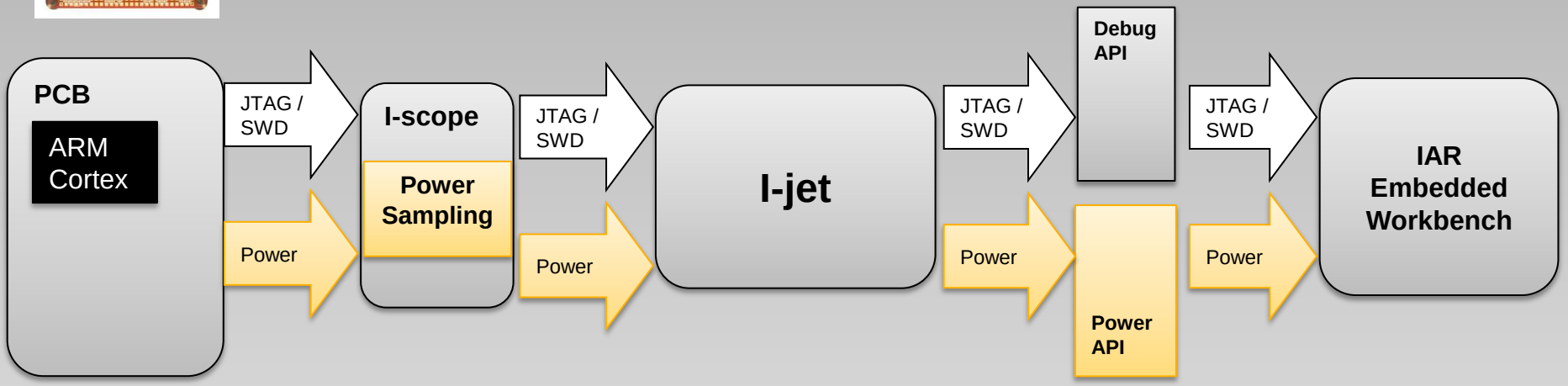
I-scope

- 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 (→ 1.5mV common mode, 27µV diff.)

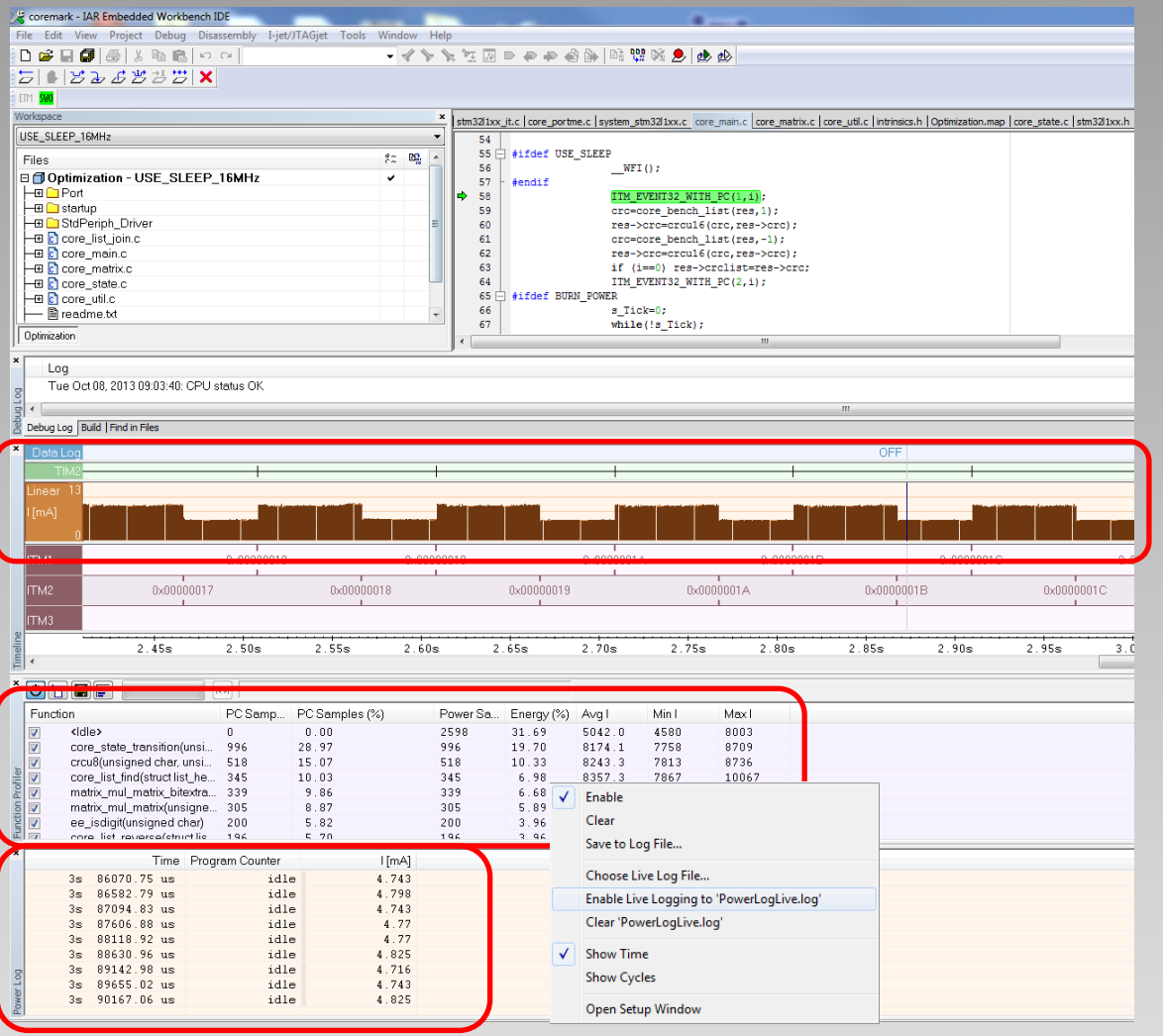
Power Debugging Setup







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

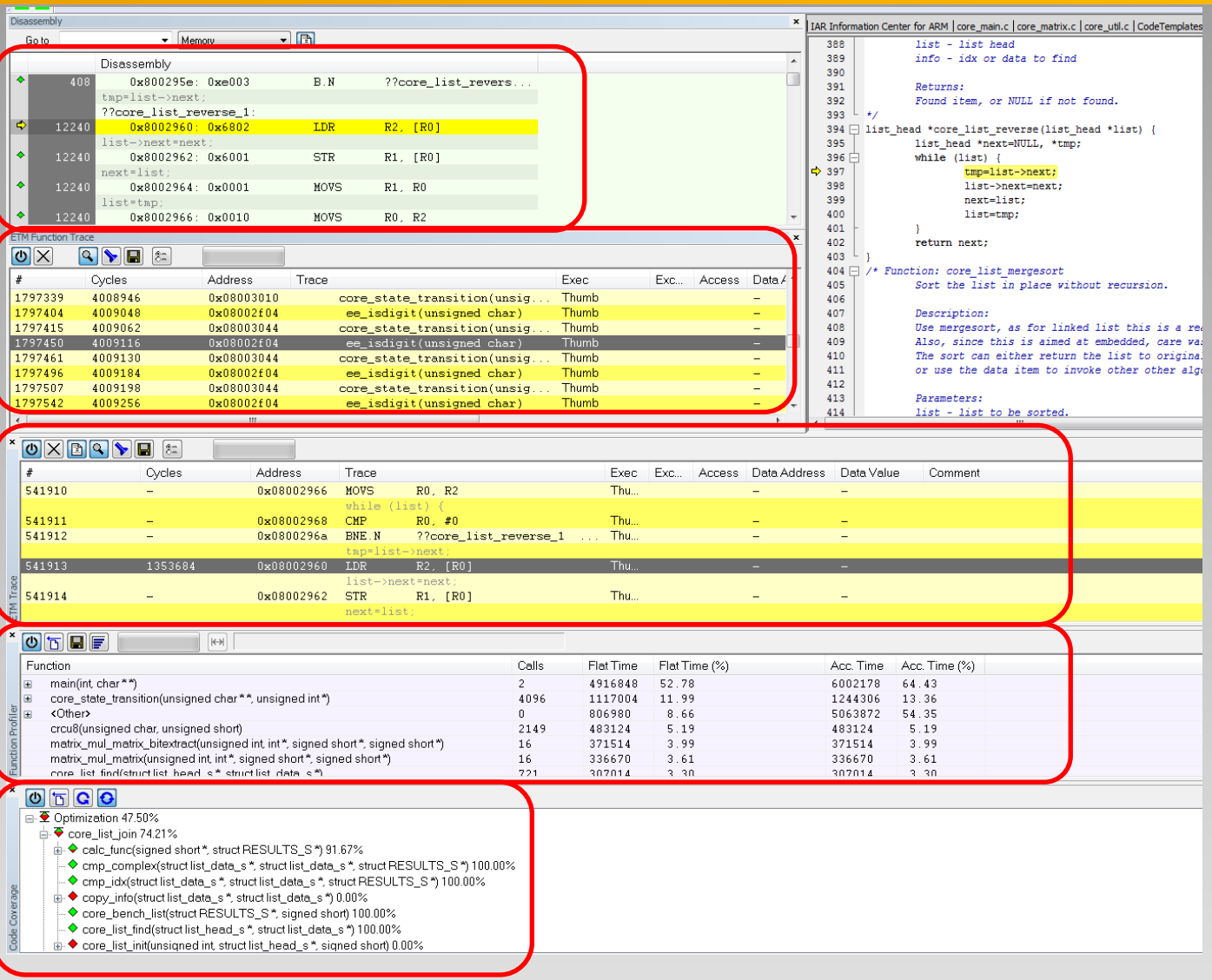
Instruction Profiling →

Function Trace →

ETM Trace →

Function Profiler →

Code Coverage →



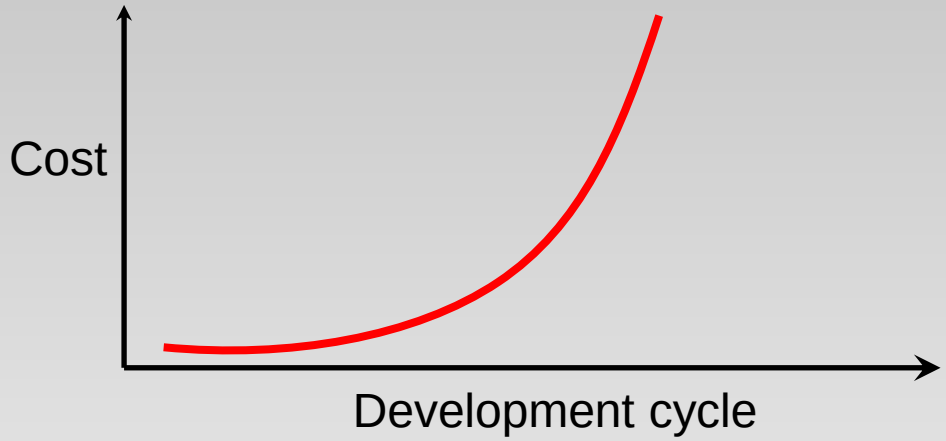
Agenda

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

LET US AGREE ON TWO FACTS

1. All software contains bugs

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



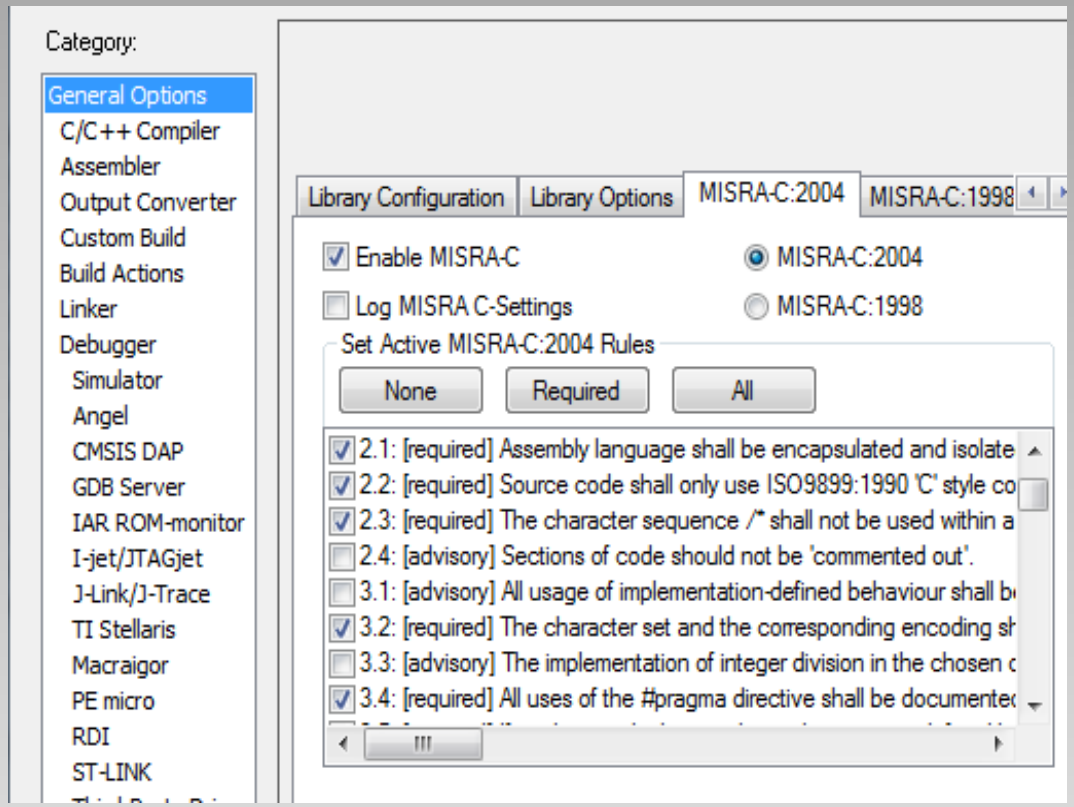
MISRA C definition

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 reliabilty
- 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

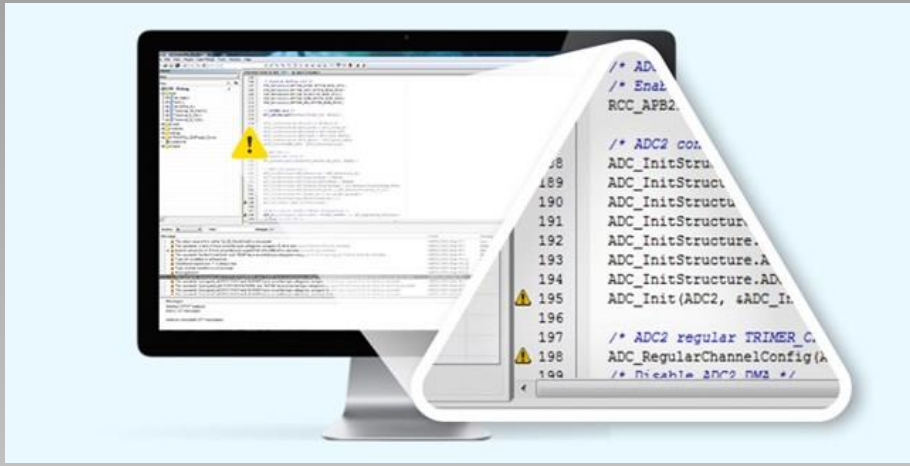
Safety – MISRA-C

- MISRA-C:2004 Checker included in commercial license



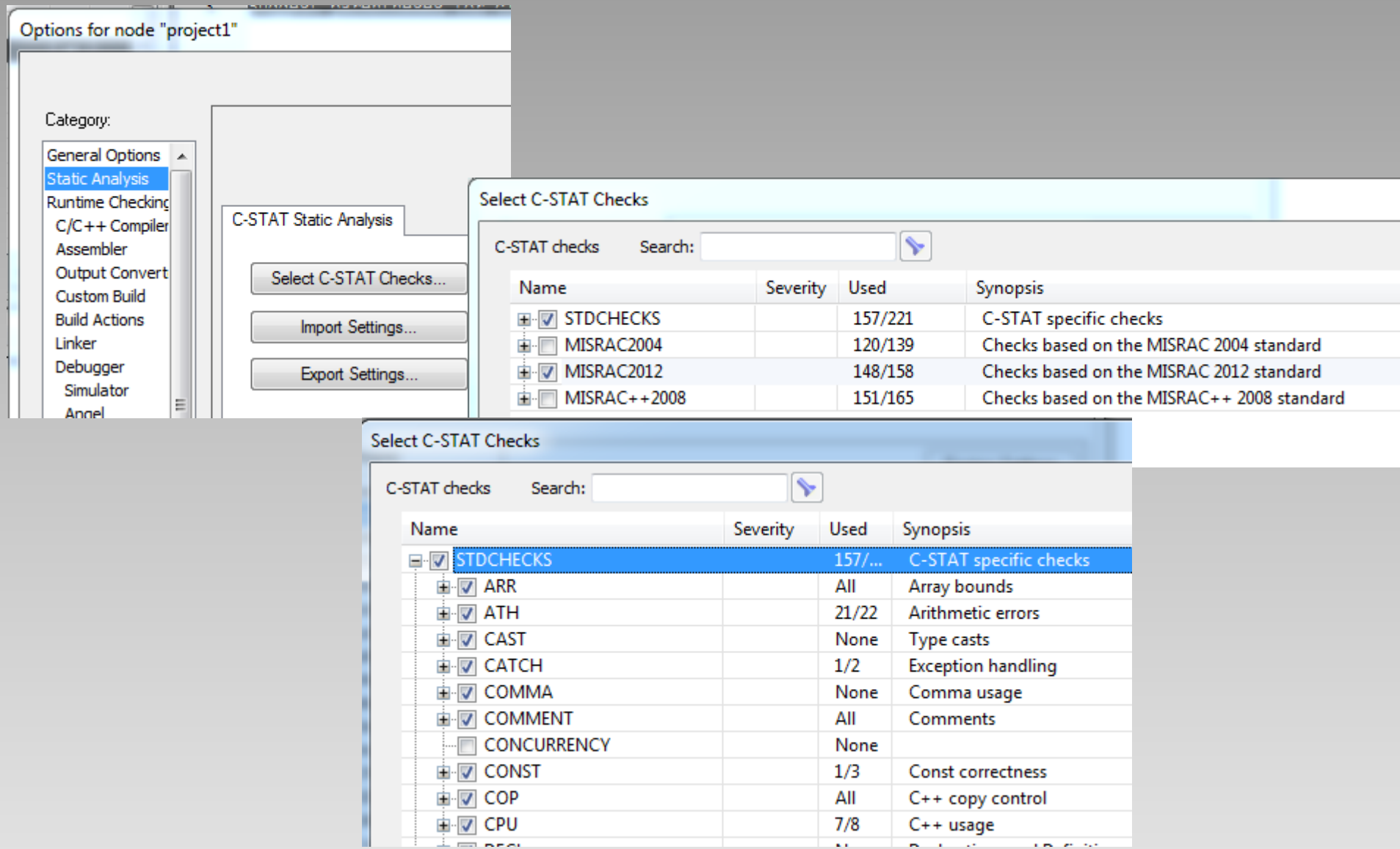
Static Analysis C-STAT

- Coding rules
  - MISRA C: 2004
  - MISRA C: 2012
  - MISRA C++: 2008
  - >200 additional checks based on CERT, CWE etc.

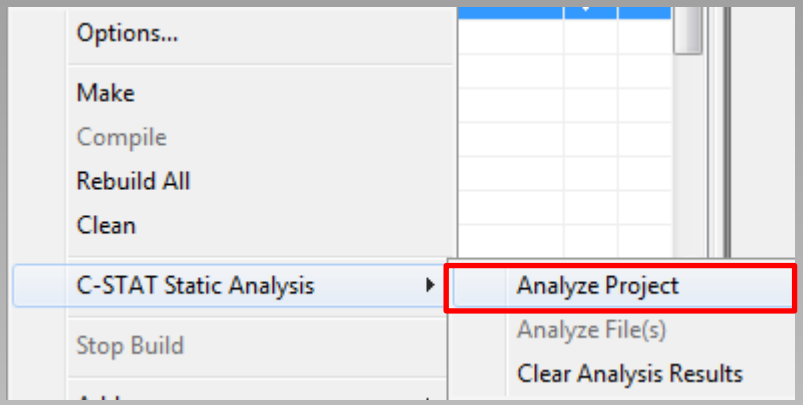


- Export/import of selected checks
- Full integration in IAR Embedded Workbench
- Can also be used through Eclipse plugin or from command line

Static Analysis C-STAT (cont'd)



Static Analysis C-STAT (cont'd)



Severity: Medium/High Filter: Messages: 18 / 52

C-STAT Messages

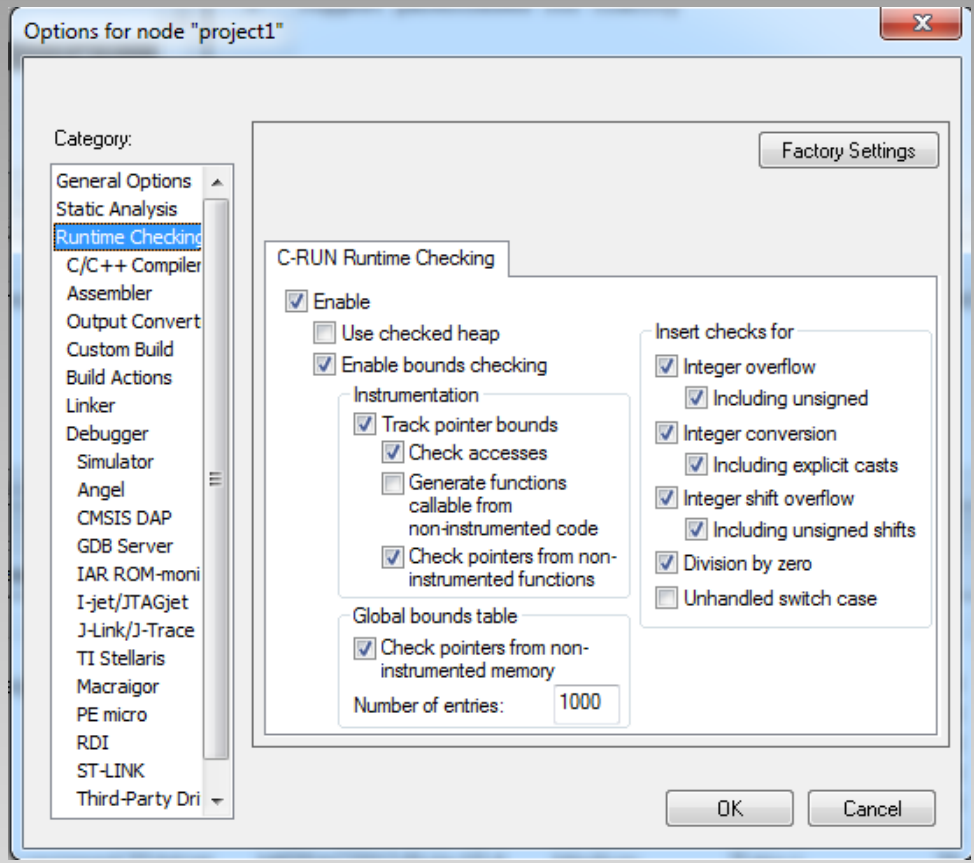
Message	Check	Severity	File	Line
Tutor.c (8 messages)			Tutor.c	
Implicit conversion of `my_uint32' from essential type unsigned 32-bit int to diff...	MISRAC2012-Rule-10.3	Medium	Tutor.c	64
main			Tutor.c	64
narrow_or_different_essential_type			Tutor.c	64
Variable `my_uint32' is uninitialized	SPC-uninit-var-all	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

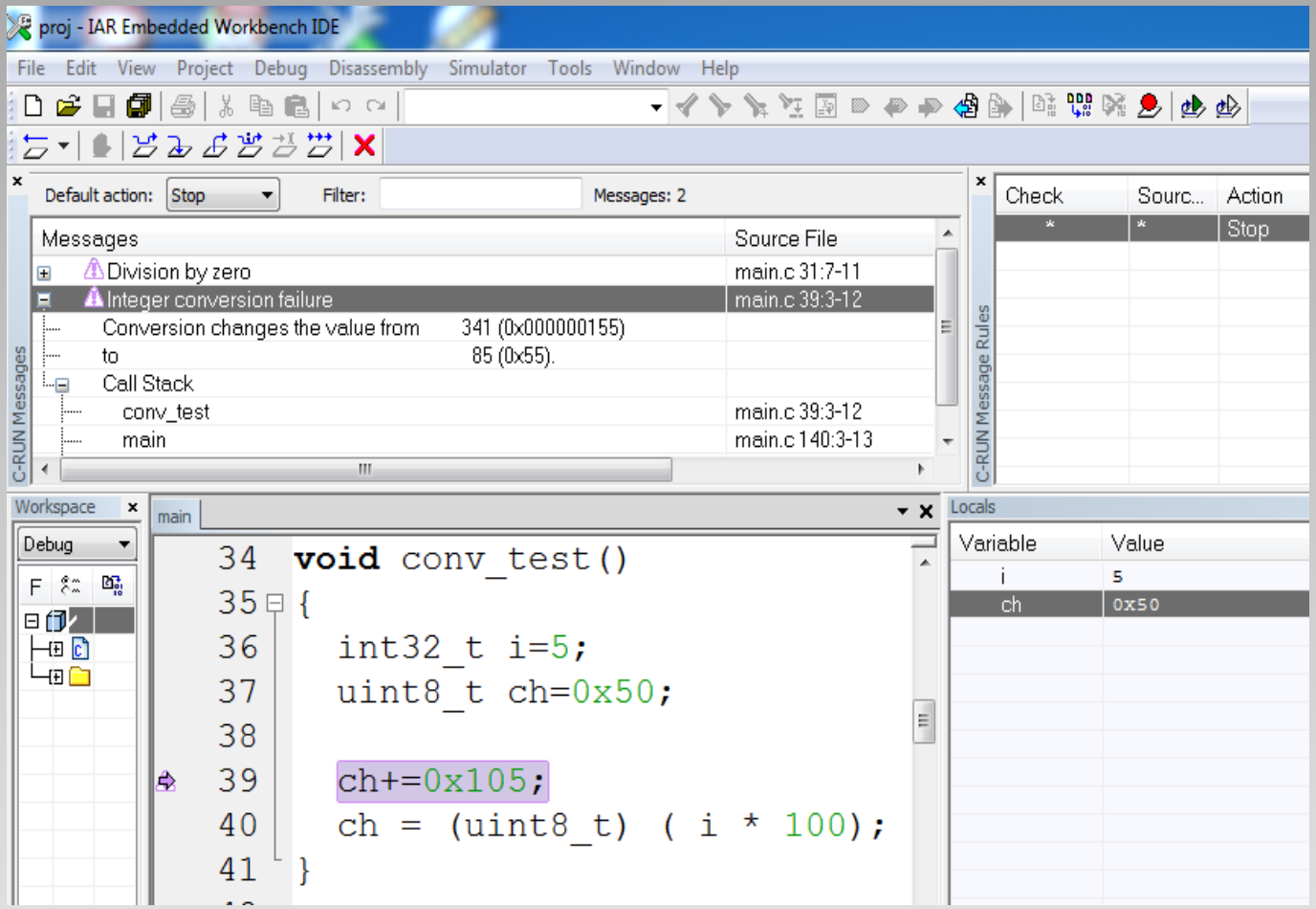
Runtime Analysis C-RUN, cont'd

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



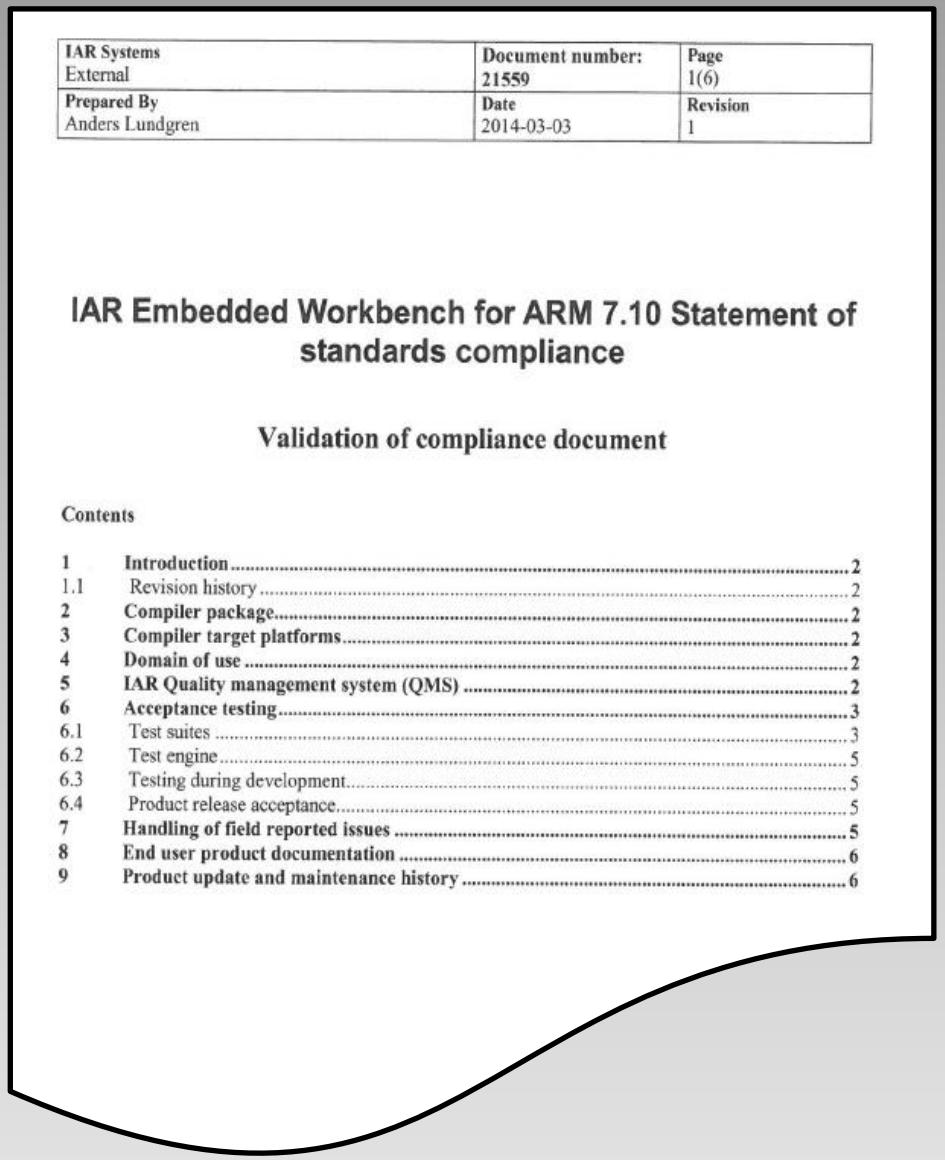
Runtime Analysis C-RUN, cont'd

- Very detailed feedback on what went wrong



Validation of Compliance

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



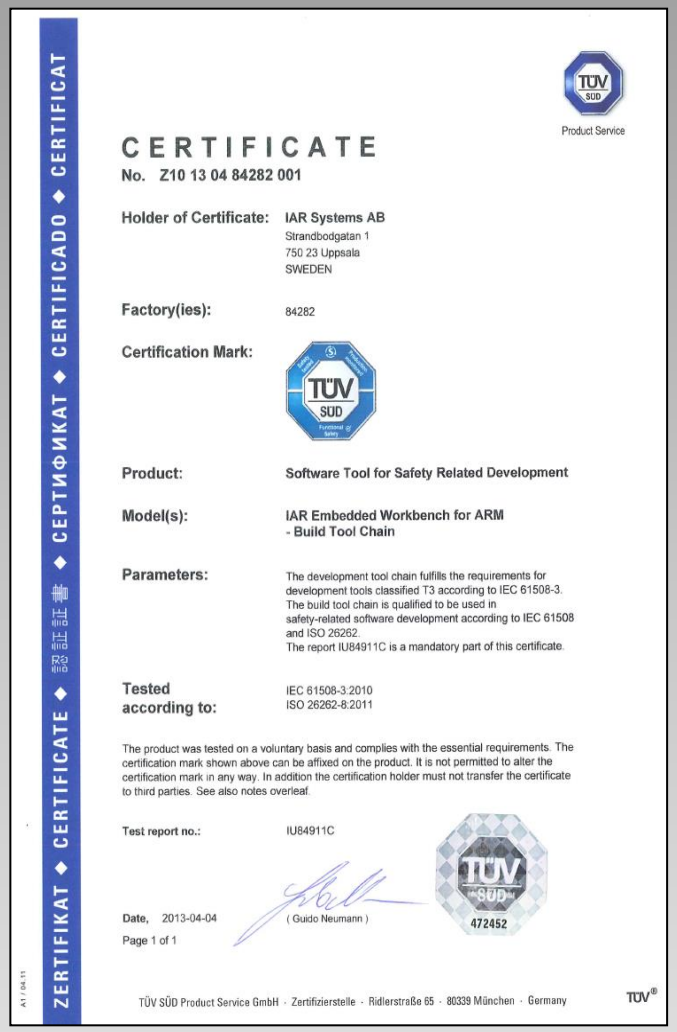
IAR Systems External	Document number: 21559	Page 1(6)
Prepared By Anders Lundgren	Date 2014-03-03	Revision 1

IAR Embedded Workbench for ARM 7.10 Statement of standards compliance

Validation of compliance document

Contents

EWARM – TÜV certified tool chain



ZERTIFIKAT ◆ CERTIFICATE ◆ 認証書 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

Product Service

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

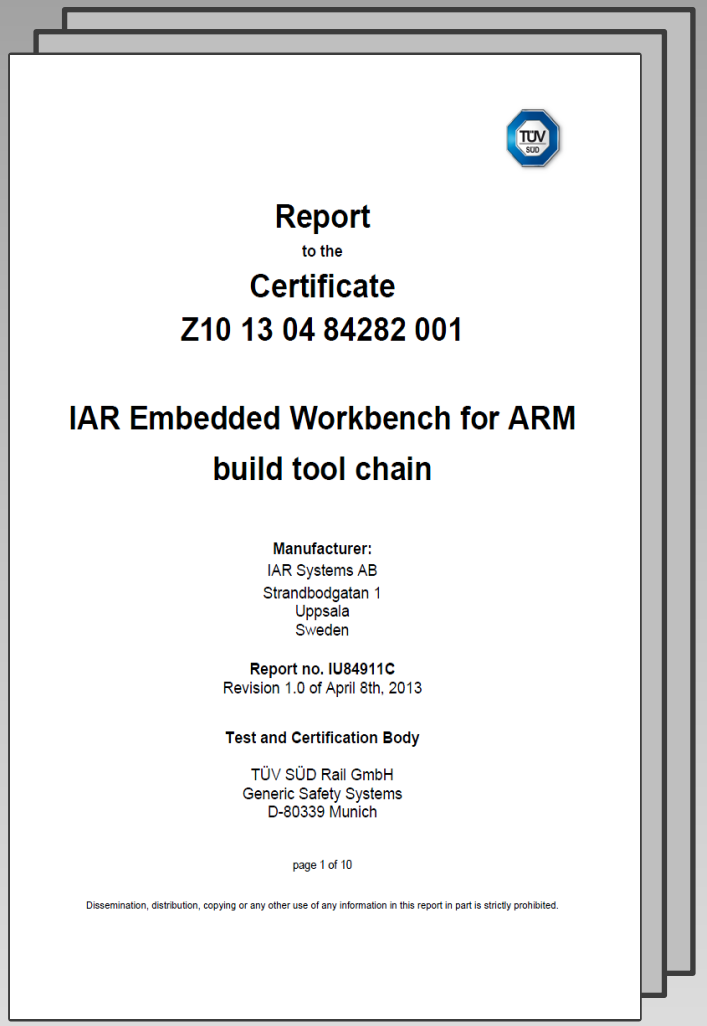
(Guido Neumann)

472452

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

TÜV®

A1 / 04.11



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

Agenda

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

IAR Academy

- 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!

IAR SYSTEMS