Take full control of your MCU software development

Ryan Sheng
ryan.sheng@iar.com
IAR Systems

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

Global technical support in
9 languages

Distributor representation in
43 countries

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

Stability and growth

<table>
<thead>
<tr>
<th>Year</th>
<th>License # thousands</th>
<th>Operating margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>5</td>
<td>0%</td>
</tr>
<tr>
<td>2011</td>
<td>6</td>
<td>5%</td>
</tr>
<tr>
<td>2012</td>
<td>7</td>
<td>10%</td>
</tr>
<tr>
<td>2013</td>
<td>8</td>
<td>15%</td>
</tr>
<tr>
<td>2014</td>
<td>9</td>
<td>20%</td>
</tr>
</tbody>
</table>

License #

Operating Margin 25%
IAR Embedded Workbench
*C/C++ compiler and debugger toolchain*

Outstanding optimization for both compact code size and high performance

Comprehensive debugger

User-friendly features and broad ecosystem integration

Global technical support

MISRA-C checker
ARM ABI compliant
IAR Embedded Workbench

Device support for NXP MCU/MPU

- **IAR Embedded Workbench for ARM**
  - LPC, Kinetis
  - i.MX, Vybrid
  - S32K, MAC57D5xx
- **IAR Embedded Workbench for ColdFire**
  - V1, V2, V3, ColdFire+
- **IAR Embedded Workbench for HCS12**
  - 16-bit HC12, S12 devices
- **IAR Embedded Workbench for S08**
  - 8-bit S08 devices
- **IAR Embedded Workbench for 8051**
  - 8-bit LPC7xx, LPC9xx, …
IAR Embedded Workbench for ARM

- Unique independence with support for all available ARM cores, from all major vendors including NXP, ST, TI, Cypress, Microchip, Renesas, SiLabs, Toshiba, etc.
- 5,000+ supported devices
- Close cooperation with SoC vendors

IAR KickStart Kit
IAR Embedded Workbench for ARM
Recent updates and highlights

- Updated IDE look and feel
- Enhanced compiler optimizations (for speed)
- Support Cortex-M23 & Cortex-M33 (ARMv8-M)
- CMSIS-Pack
- I-jet Trace for ARM (Cortex-A/R/M)
- Flash breakpoint
- Multi-core debugging
- Stack usage analysis
- Enhanced static and runtime code analysis
Leading compiler optimizations enable IAR Embedded Workbench to generate the most compact and fast performing code.

- Smaller memory size
- Better real-time reaction
- Lower power consumption

EEMBC proved code performance: Find the leading CoreMark® scores on the website.

www.eembc.org/coremark
IAR Embedded Workbench for ARM

Functional safety certified edition of IAR Embedded Workbench for ARM

- IEC 61508
- ISO 26262
- EN 50128

Simplified validation
- Functional safety certificate
- Report of the certificate
- Safety Guide

Guaranteed support through the product life cycle
- Prioritized technical support
- Validated service packs
- Regular report of known problems
**IAR I-jet & I-jet Trace**

*Hardware debuggers for ARM*

**I-jet for ARM**

I-jet provides an exceptionally **fast** debugging platform.

**I-jet Trace for ARM**

I-jet Trace delivers **large trace memory capacities** and **high-speed communication** via SuperSpeed USB 3.0.

**I-scope for ARM**

I-scope is a small probe that adds **current & voltage measurement capabilities** to I-jet.

**3rd-Party Debuggers**

- CMSIS-DAP
- Segger J-Link / J-Trace
- P&E Micro Multilink
- P&E Micro Cyclone
  ......
Generate IAR example projects using NXP MCUXpresso Tools
MCUXpresso Software and Tools
for Kinetis and LPC microcontrollers

MCUXpresso IDE
Edit, compile, debug and optimize in an intuitive and powerful IDE

MCUXpresso SDK
Runtime software including peripheral drivers, middleware, RTOS, demos and more

MCUXpresso Config Tools
Online and desktop tool suite for system configuration and optimization
MCUXpresso 配置工具
MCUXpresso配置工具提供一套系统配置工具，通过基于Kinetis或LPC的MCU解决方案为各级用户提供帮助。它可引导帮助您完成从初次评估到生产开发的整个过程。
Create a new configuration

- mcuxpresso.nxp.com/zh/builder
- Select from:
  - Boards
  - Processors
  - Kits
- Select configuration
- Specify configuration settings
- Jump start your configuration
Configuration settings

配置设置

指定包含的中间件、RTOS选择和开发首选项。

当前配置

开发者环境设置

此处的选择将影响SDK下载和生成工程中包含的文件和示例工程

主机 OS

Windows

工具链/IDE

IAR Embedded Workbench for ARM

选择可选中间件

此处的选择将包含在您的SDK下载、生成工程中，并将影响外设工具设置

3 items selected

选定的中间件

FatFS, USB stack, FreeRTOS

www.iar.com
下载SDK生成存档项

如果10秒后未开始下载，请单击以下链接，开始下载软件包：
SDK_2.2_LPCXpresso54114.zip

查看SDK详情

您的SDK下载将包含侧边面板列出的项目。
可使用“工具”→“配置设置”页编辑这些设置的项。

此MCUXpresso SDK配置可直接下载

软件包名称
SDK_2.2_LPCXpresso54114
Download an existing example project

下载SDK生成存档项

如果10秒后未开始下载，请单击以下链接，开始下载软件包:
hello_world_cm4.zip

选择一个示例工程

下载示例  应用工程到配置工具
Advanced debugging & trace on LPC & Kinetis Microcontrollers
Cortex-M3/M4 trace system

- **Cortex-M3/M4 Core** connects to **ETM** through **CPU I/F**.
- **ETM** has **DWT** for **ETM Trigger**, **4 Watchpoints**, **PC Sampler**, and **Interrupt Trace**.
- **ITM** includes **Software Trace** and **Timestamp**.
- **TPIU** manages **Trace Port Interface Unit**.
- **Bus Matrix** connects **AHB-AP** and **DP** via **DAP Bus**.
- **AHB-AP** has **IAR I-jet Trace**.
- **IAR I-jet** communicates with **MIPI-10 or MIPI-20 Cable**.
- **MIPI-20 Cable** connects through **SWO**, **JTAG**, and **SWD**.

---

This diagram illustrates the integration of the Cortex-M3/M4 core with various trace systems, including the ETM, ITM, and TPIU, along with the communication through bus matrices and interface units.
### Trace pins in different connector

#### Standard 20-pin

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTRref</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>(nTRST)</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>(TDI)</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWDIO</td>
<td>(TMS)</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWCLK</td>
<td>(TCK)</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>(RTCK)</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWO</td>
<td>(TDO)</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nSRST</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+5V</td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### JTAG/SWD

#### MIPI-10 0.05”

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTref</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNDDetect</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### JTAG/SWD

#### MIPI-20 0.05”

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTref</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNDDetect</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td></td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SWO trace

- **SWO (Serial Wire Output)**
  - A serial high speed signal that transmits ITM packets
  - Events and sampling based
  - Supported by Cortex-M3/M4 architectures
  - Supported by IAR I-jet and other debuggers

- **Trace information going through SWO**
  - **DWT (Data Watchpoint and Trace)**
    - Watchpoint: 4 independent comparators for address and data
    - PC sampler: Sampling the PC register at regular intervals
    - Interrupt trace: Logging the enter and exit of each interrupt
  - **ITM (Instrumentation Trace Macrocell)**
    - Generate ITM events on 32 independent ports
    - Packetize and timestamp the DWT events
• DWT generates trace events when one of the watchpoint finds a match on specified address/data.
• Up to four different static variables can be monitored together.
• C-SPY displays the collected trace information in the Data Log window and the graphical Timeline window.

Static variables monitoring

![Graph showing static variable monitoring]
Using data log breakpoints

Right-click on the name of a variable to be monitored:

The breakpoint is triggered when the variable at 0x20002DE0 is read or written as a word (4 bytes). DWT will generate an event but the execution will not be stopped.

Check the status and edit the properties of the breakpoint in View → Breakpoints:
Interrupt logging

- DWT generates trace events when entering or leaving any interrupt.
- C-SPY displays the collected trace information in the Interrupt Log window and the graphical Timeline window.
- Useful to find interrupts which can be fine-tuned to execute faster and analyze problems with nested interrupts.
Function profiling

- Code profiling information of each C function is retrieved by counting the number of PC samples generated by the PC sampler of DWT.
- Useful to find where the CPU is spending its time.
- Functions where the most time is spent should be carefully optimized or moved to more efficient memory to increase the performance.
The target application can send data directly to the host debugger through ITM stimulus ports.

Each of the 32 ITM ports has its own address (based at 0xE0000000).

C-SPY displays the data sent from ITM port #1~#4 in the Event Log window and the graphical Timeline window.

```c
#include <arm_itm.h>

......

for (x=0; x<10; x++)
{
    for (y=1; y<5; y++)
    {
        ITM_EVENT8(y, x);
    }
}
```
Execution time measurement

- Use the functionality of ITM events to measure the time consumption of a piece of code.
- Send two ITM packets before and after the code to be measured and the actual execution time is the interval between them.
- Easy, accurate and no additional equipments are required!

```
#include <arm_itm.h>
……
ITM_EVENT8(1, 0x55);
CodeToBeMeasured();
ITM_EVENT8(1, 0xAA);
```

- Time consumption of CodeToBeMeasured() on a 100MHz CPU:
  - \((476367457 – 382817996) / 100000000 = 0.935 \text{ (s)}\)
Instruction trace

• Collect a sequence of every executed instruction continuously for a selected portion of the program.
• Developers can inspect the program flow up to a specific state and locate the origin of the problem.
• Very useful for locating errors that have irregular symptoms and occur sporadically.
  • Illegal instructions and data aborts
  • Runaway programs
  • Interrupt/exception problems
  • Context switch problems
• ......
• Also helpful for analyzing dynamic system behaviors
  • Code profiling
  • Code coverage
Instruction trace techniques

- ETM (Embedded Trace Macrocell)
  - Off-chip trace buffer (in the probe, 2~32 MB)
  - 4~16-bit data bus at CPUCLK or CPUCLK/2
  - High requirement on the board design, e.g. for acceptable signal quality
  - Expensive trace probes required (e.g. JTAGjet-Trace)

- ETB (Embedded Trace Buffer)
  - On-chip dedicated trace buffer (small – a few Kbytes)
  - No extra pins, no requirement for trace probes
  - Cortex-M3, Cortex-M4

- MTB (Micro Trace Buffer)
  - On-chip configurable trace buffer (small – a few Kbytes)
  - No extra pins, no requirement for trace probes
  - Cortex-M0+
Collect executed instructions

- Go to “ETM Trace” window to check the recorded instructions together with mixed C source code.
View the trace data at function-level

- Go to “ETM Function Trace” window to view function-level information.
- Useful to find the internal process of complex functions, or the actual calling sequence of interrupt / task switches.
Using trace start/stop breakpoints

Right-click in the source or disassembly window:

- Toggle Breakpoint (Trace Start)
- Toggle Breakpoint (Trace Stop)
- Toggle Breakpoint (Trace Filter)
- Enable/Disable Breakpoint
- Edit Breakpoint
- Set Next Statement

Check the status and edit the properties of the breakpoint in View → Breakpoints:

- Location in the source
- Address of the instruction

The breakpoint is triggered when the instruction at the specified address is fetched.
Graphical call stack

Posting a semaphore

Pending on a semaphore

Implementation of printf( )
Code coverage

- **Red**: 0% of the module or function has been executed.
- **Green**: 100% of the module or function has been executed.
- **Red & Green**: Some part of the module or function has been executed.
- **Yellow**: The statement in C/C++ source code that has not been executed.
EWARM debugging environment
Multi-core debugging

• Symmetric multicore debugging (SMP)
  • Debugging two or more identical cores;
  • One single debug probe;
  • One single instance of IAR EWARM IDE.

• Asymmetric multicore debugging (AMP)
  • Debugging two different cores;
  • One single debug probe;
  • Two cooperating instances of IAR EWARM IDE.
Multi-core debugging: Configuration

SMP: Number of cores

AMP: Set the direction to the slave project in the master project.
Multi-core debugging: LPC54114

Execution State:
- in focus, not executing
- in focus, executing
- not in focus, executing
- in focus, in sleep mode
- not in focus, in sleep mode

Execute command for each core
Start all cores
Stop all cores
Keeping Safe at C: Static & Runtime Code Analysis
Integrated code analysis add-on tools

- Static code analysis: C-STAT
  - Analyze the C/C++ source code without executing the program.
  - Fully integrated in IAR Embedded Workbench for ARM.
- Runtime code analysis: C-RUN
  - Find C programming errors at runtime.
  - Fully integrated in IAR Embedded Workbench for ARM.

Maximized performance by compiler experts

- Fully integrated runtime and static analysis
- IDE tools: Editor, Project manager, Library tools
- Build tools: C/C++ Compiler, Assembler, Linker
- C-SPY Debugger: Simulator, Hardware debugging, Power debugging, RTOS plugins
C-STAT and C-RUN

C-STAT Static analysis
C-STAT performs advanced analysis of your C/C++ code and finds potential issues. It helps you improve your code quality as well as prove alignment with standards such as MISRA C:2012.

C-RUN Runtime analysis
C-RUN helps you find errors at an early stage. It is completely integrated with IAR Embedded Workbench for ARM, and provides detailed runtime error information.
Most bugs are found until Test phases.

Find bugs later = Higher cost

Find more bugs during coding and debugging!
C-STAT: What does it check

- Common Weakness Enumeration  
  - [cwe.mitre.org](http://cwe.mitre.org)  
  - An unified and measurable set of software weaknesses.
  - Enumerate design and architecture weaknesses, as well as low-level coding errors.

- Computer Emergency Response Team  
  - [www.cert.org](http://www.cert.org)  
  - C/C++ secure coding standards, identifying insecure constructs which could expose a weakness or vulnerability in the software.
  - Guidelines to avoid implementation, coding as well as low-level design errors.

- Motor Industry Software Reliability Association  
  - [www.misra.org.uk](http://www.misra.org.uk)  
  - MISRA C:2004 (MISRA C2): Identify unsafe code constructs in the C89 standard.
  - MISRA C:2012 (MISRA C3): Extend the support to C99 version of the programming language whilst maintaining the guidelines for C89 standard.
C-STAT: Options in IAR EWARM

- CWE/CERT rules
- MISRA C/C++ rules
Enable or disable a set of rules or any individual rule.

Highlight a rule and press F1 to show the detailed description.
Filter the C-STAT messages by selecting a level of severity: All, Low, Medium or High.

Double click the C-STAT message to direct to the line of source code.

Highlight the C-STAT message and press F1 to show the related rules information.
C-RUN: What does it check

- Heap checking
- Arithmetic checking
- Bounds checking
C-RUN: How does it work

- Traditional runtime analysis tools:
  - Independent with compiler and debugger;
  - Different applications and license models;
  - Less knowledge about the target and optimization;
  - Insert test code at the source code level;
  - Large overhead in memory size and execution speed.

- C-RUN:
  - Created by compiler and debugger experts;
  - Fully integrated within IAR Embedded Workbench;
  - Insert target optimized test code directly during compilation;
  - Replace the C/C++ standard library with a dedicated library which contains special functionality for runtime error checking;
  - Result in minimized ROM/RAM overhead and speed penalty.
Detecting integer overflow

```c
void main (void)
{
    int v1 = 0x7fffffff;
    unsigned int v2 = 0xffffffff;

    v1++; /* signed integer overflow */
    v2++; /* unsigned integer overflow */
}
```

The image shows a code snippet with comments for detecting signed and unsigned integer overflow. The code initializes `v1` with the value `0x7fffffff`, which is the maximum value for a signed integer, and `v2` with `0xffffffff`, which is the maximum value for an unsigned integer. The code then increments both `v1` and `v2`, resulting in integer overflow for the signed integer and detection of the overflow by the IDE.
#include <stdlib.h>
#include <iar_dlmalloc.h>

void main (void)
{
    char *c = malloc(10);
    c = malloc(20);    /* memory leak */

    free(c);

    /* check for memory leaks, manually called */
    __iar_checkLeaks();
}

Detecting heap errors

- Memory leak detected:
  - There were a total of 1 heap blocks with no references.
  - Heap block 0 at 0x00102450 has no references.
  - The block was allocated at line 6 of main.c.

Call Stack:
- main
- [main + 0x4]
int main (void)
{
    int i, j;
    int a[3] = {1, 2, 3};

    for (i=0; a[i]!=0; i++) /* out of bounds */
    {
        /* when i==3 */
        j = a[i];
    }

    return j;
}
Take full control of your development

- Implement your design in code
  - Let C-STAT analyze your code
    - C-STAT Static Analysis
    - Analyze Project
    - Analyze File(s)
    - Stop Build
    - Clear Analysis Results

- Build and debug the application
  - Let C-RUN analyze your project
    - C-RUN Runtime Checking
      - Enable
      - Use checked heap
      - Enable bounds checking
      - Insert checks for
      - Integer overflow

- Review potential issues
- Investigate runtime errors

Release the application

Requirements Design Implementation Verification Maintenance