IAR Systems & Express Logic Annual Developers' Meeting in Israel

Agenda – IAR Systems

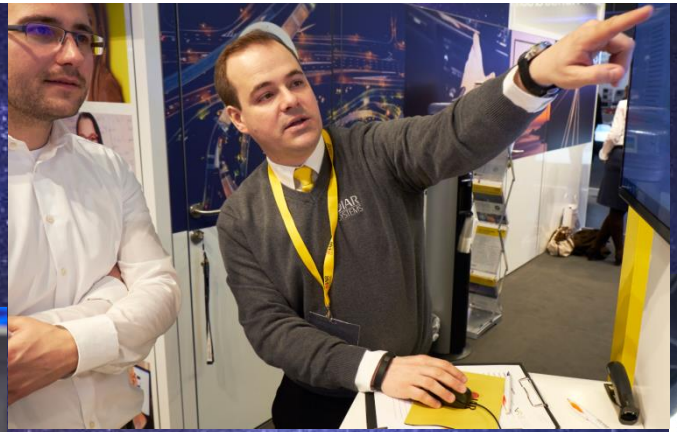
- Introduction
- Embedded Workbench
  - New Features
  - Roadmap
- Code Analysis with C-STAT and C-RUN
- The importance of Functional Safety
- Benchmarks
- IAR Support for Cortex A9 based FPGA SoCs
- Visual State in practice

Introduction

Future-proof software tools and services for embedded development, enabling companies worldwide to create the products of today and the innovations of tomorrow.

IAR SYSTEMS

- **Dedicated team of support, sales and service worldwide**
- **46,000 customers**
- **32% of revenue invested**



- **34 years in the industry**
- **Listed on NASDAQ Stockholm**

Uppsala	**Shanghai**	**+ Distributor representation in 40+ countries**
Munich	**Dallas**	
Paris	Boston	
Tokyo	Los Angeles	
Seoul	San Francisco	

2016

- **Sales SEK 328,4 m**
- **Operating profit 96,5 m**
- **Net cash 96,5 m**

IoT is everywhere!

IAR SYSTEMS

Virtually all industries have devices or products that can be further utilized through improved communication and connectivity.

27 billions
of connected devices in
2025

- Connected cars
- Smart road infrastructure
- Industrial IoT
- Healthcare IoT
- Wearables
- Smart cities

Functional safety and reliability

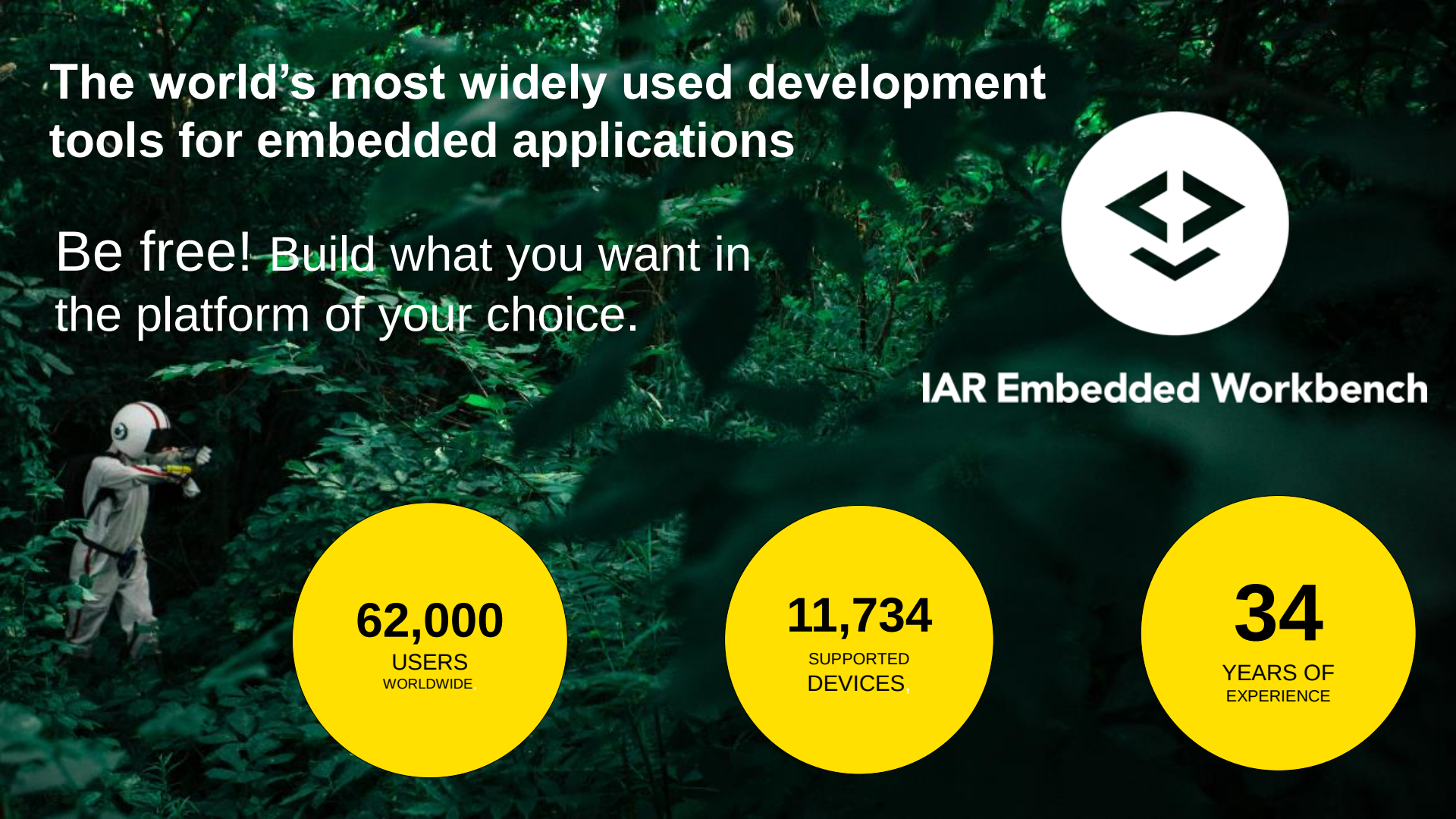
IAR SYSTEMS

One of the most important features, especially within automotive, industrial automation and medical.

- Coding standards
- Safety requirements
- Proof of compliance for tools

The world’s most widely used development tools for embedded applications

Be free! Build what you want in the platform of your choice.



IAR Embedded Workbench

62,000
USERS
WORLDWIDE

11,734
SUPPORTED
DEVICES

34
YEARS OF
EXPERIENCE

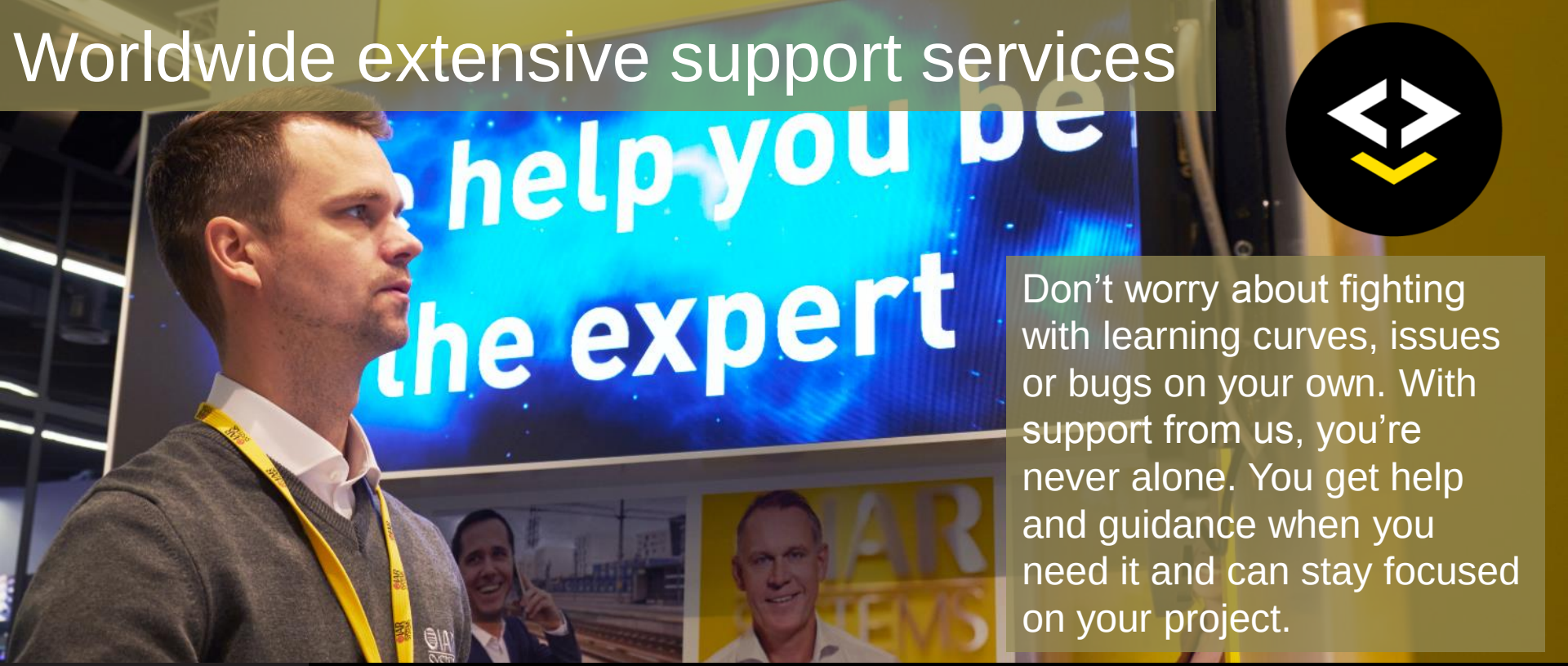
Support for 11,700+ devices

40+ architechtures

All available 8-,16- and 32-bit MCUs

Cortex-M0	Cortex-A5	AVR	R8C
Cortex-M0+	Cortex-A7	AVR32	H8
Cortex-M1	Cortex-A8	RX	STM8
Cortex-M3	Cortex-A9	RL78	ColdFire
Cortex-M4	Cortex-A15	RH850	HCS12
Cortex-M7	ARM11	78K	S08
Cortex-M23	ARM9	SuperH	MAXQ
Cortex-M33	ARM7	V850	CR16C
Cortex-R4	SecurCore	R32C	SAM8
Cortex-R5	8051	M32C	
Cortex-R7	MSP430	M16C	
Cortex-R8			

Worldwide extensive support services



Don’t worry about fighting with learning curves, issues or bugs on your own. With support from us, you’re never alone. You get help and guidance when you need it and can stay focused on your project.

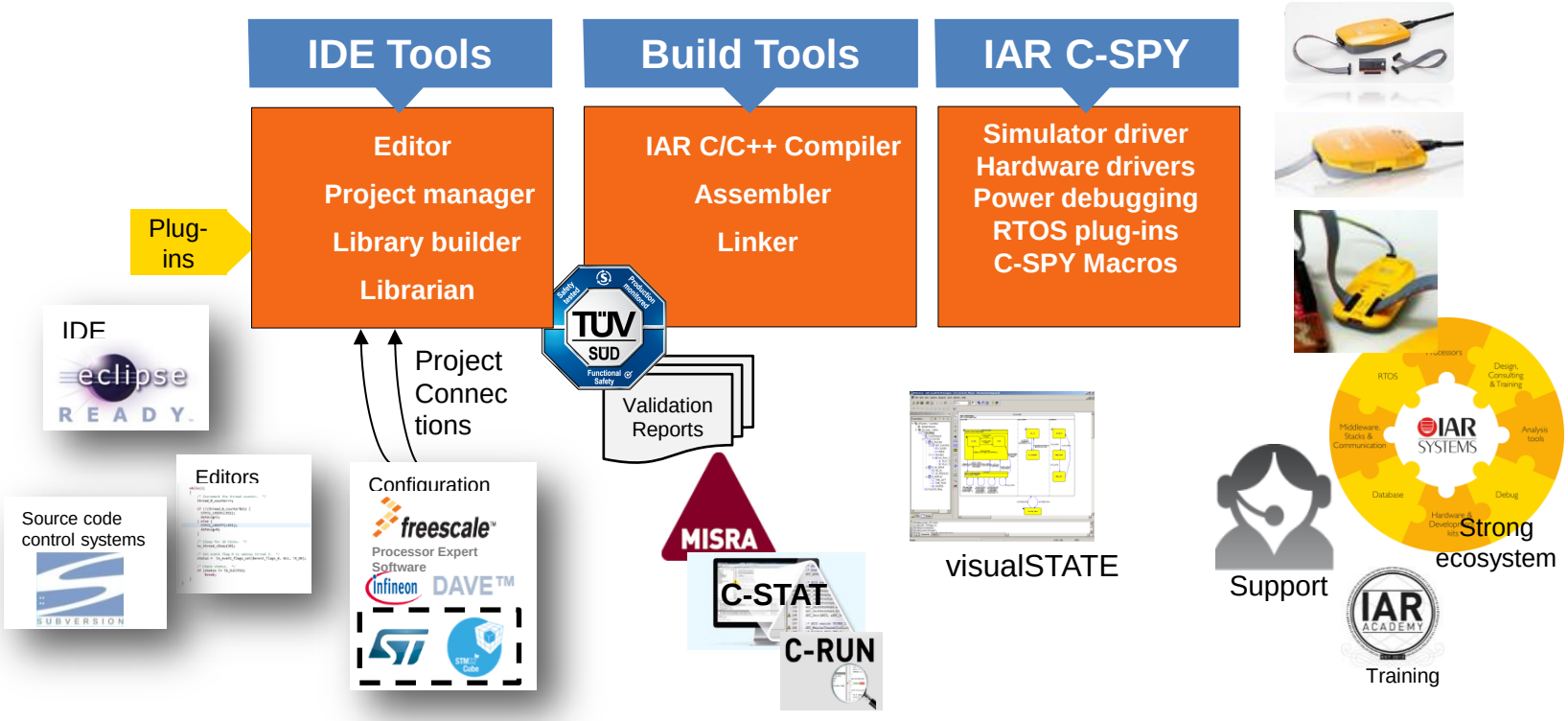
Get help from technical experts in your time zone. Support centers covering 9 languages in the US, Japan, China, Korea, Germany and Sweden.

IAR SYSTEMS

Embedded Workbench

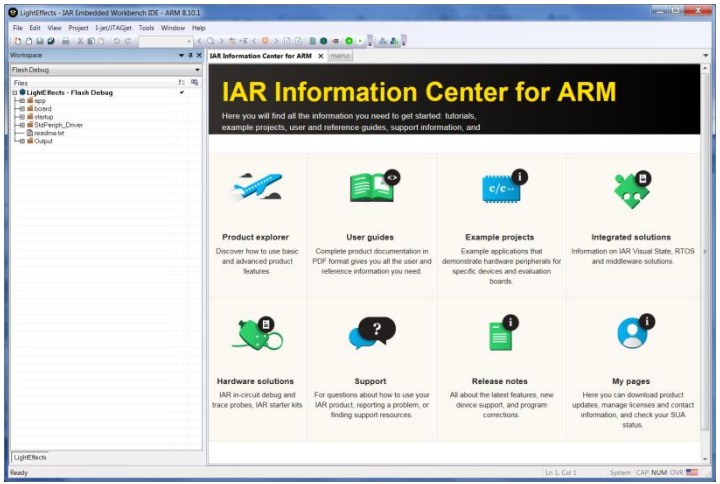
Let’s go into the details…

IAR Embedded Workbench



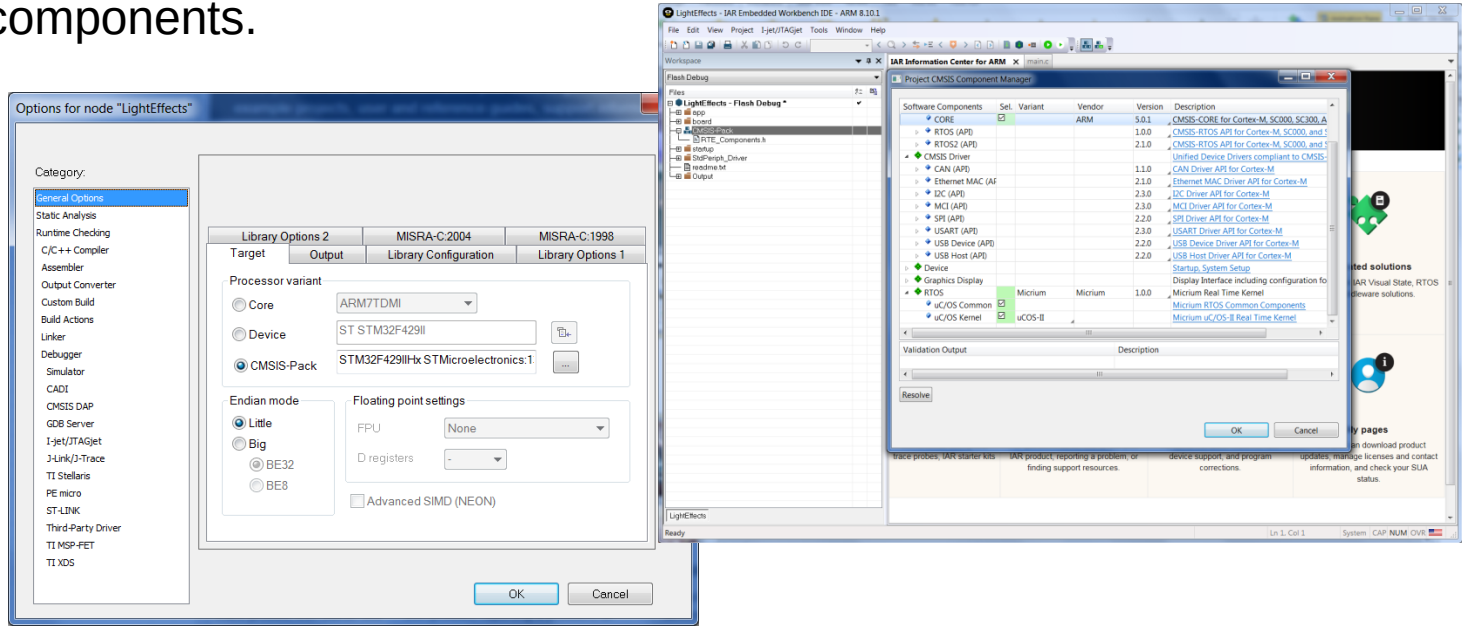
Updated IDE look and feel

- New EW logo
- New artwork
- Better window management and docking
- Revised Information Center
- Updated tutorials



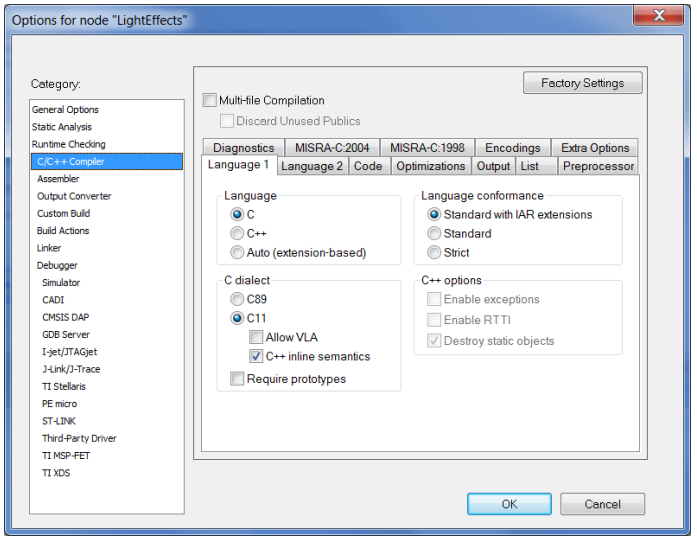
CMSIS-Pack support

- Work with ARM CMSIS-Pack software components.



C11 and C++14 language standard support

- C language standard ISO/IEC 9899:2011
- C++ standard ISO/IEC 14882:2014



Background

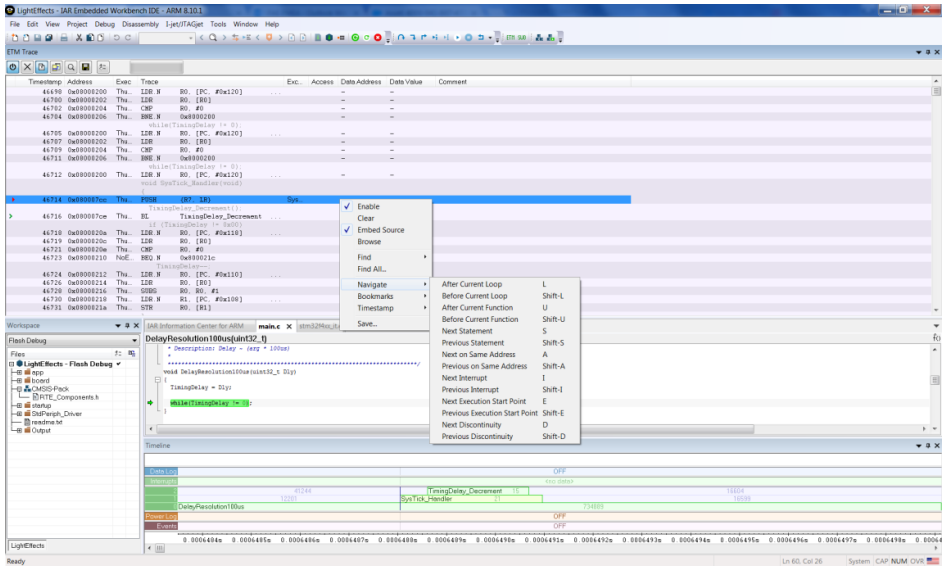
IAR SYSTEMS

- C++14 is a smallish modification to
- C++11 which was a significant upgrade from
- C++03
- C++03 is ”based” on C99
- C++14 is ”based” on C11

Demo

Trace filtering and navigation

- Navigate forward/backward
  - on loop
  - function
  - interrupts
  - statement boundaries

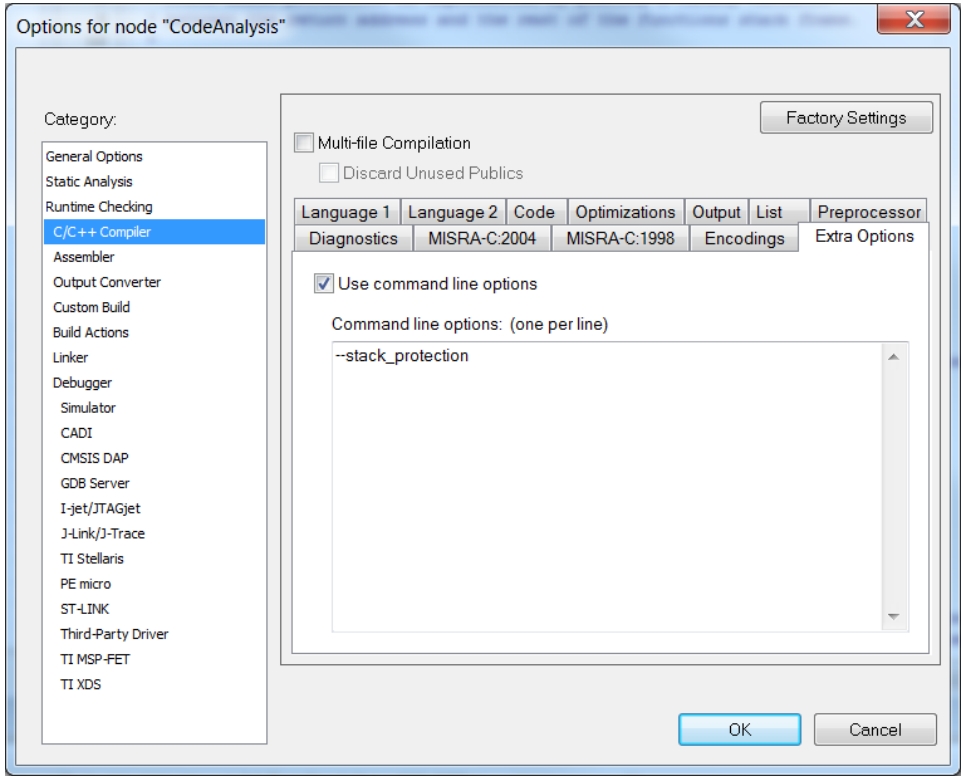


Demo

Stack protection

- Added in EWARM 8.20
- Detects corruption of a function return address
- Canary value placed between stack variables and the return address
- Compiler will determine if a function needs protection or not.

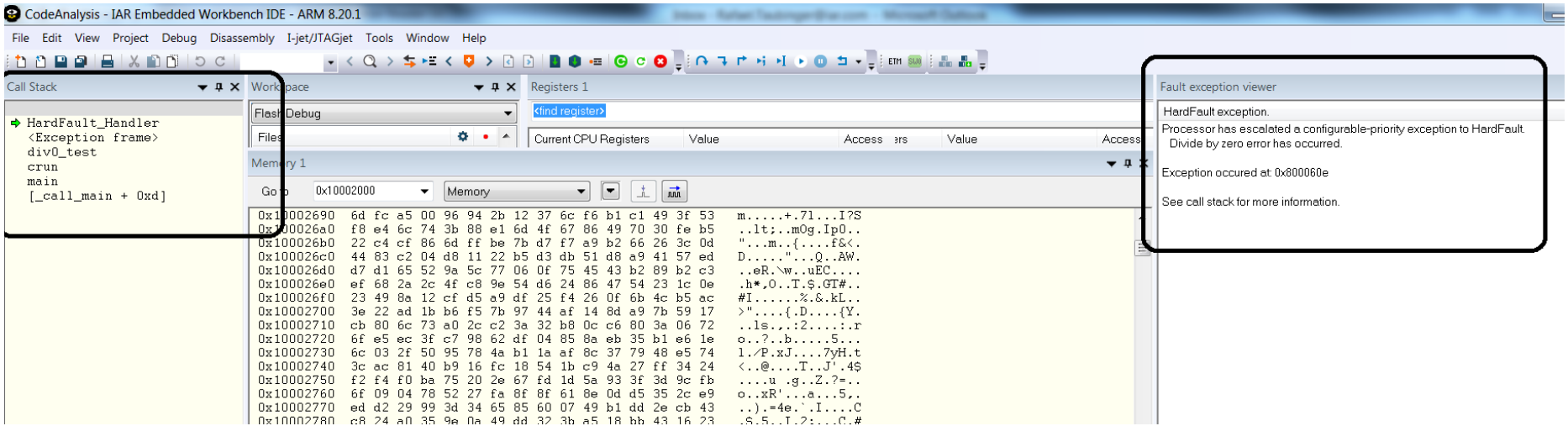
Stack protection



Fault exception viewer

- Debugger window with information about the cause of a fault exception
- Lists peripheral register in SCB
- Available for all Cortex-M devices

Fault exception viewer



New devices in EWARM 8.x

- 463 new devices support in EWARM 8.x
- ST 49
- Cypress 104
- Atmel 60
- NXP 36
- Silicon Labs 148
- Intel FPGA/Altera Cyclone V SoC

New Releases

-EWARM V8.20
Release 27 of October
* Hard fault viewer
* Stack security protection
* Optimized floating point lib
* Multi-core debugging using CTI

-EWRXFS V3.10.5
Release 31 October
* New IDE
* V2 core support
* C-STAT/C-RUN
* New devices
* IEC 62304(Medical)

-EWAVR32 V5.10
Release 20 October
* IDE facelift
* Add Atmel ICE as selectable debugger
* Improved C-STAT
* Power debugging visulization
* New devices

-EW430 V7.11
Release 6 November
* New devices

-EWRL78FS 3.10.2
Release 15 November
* New IDE
* C-STAT
* New devices
* IEC 62304(Medical)

-EWRH850 V2.10
Release 22 November
* New IDE
* C11 and C++14
* Static stack analysis
* Support for new global pointers
* E2 emulator support

Visual State V8.10-AUTO/8.20
Release 30 November
* Support for Req Mgmnt tools
* New InfoCenter
* Examples and tutorial remake
* XMI update
* Doc update

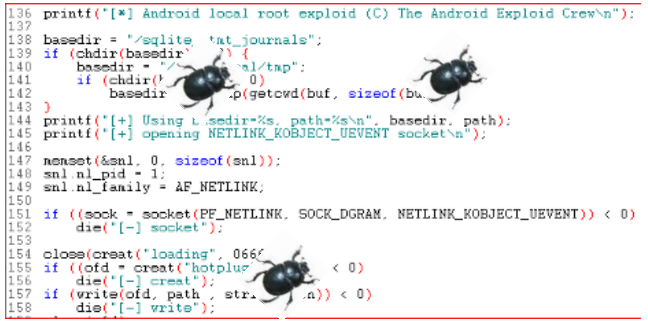
Code Analysis with C-STAT and C-RUN

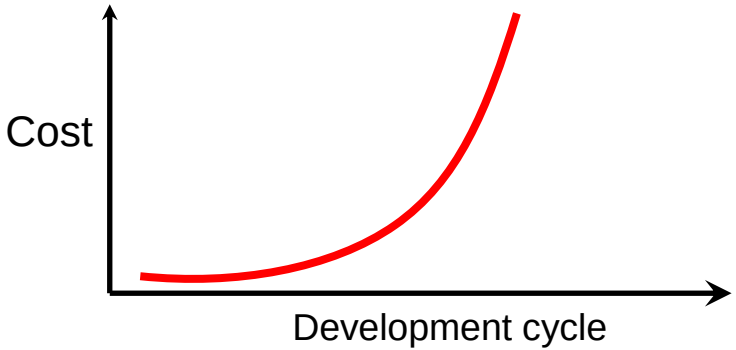
Why do we need code analysis?

C is not safe

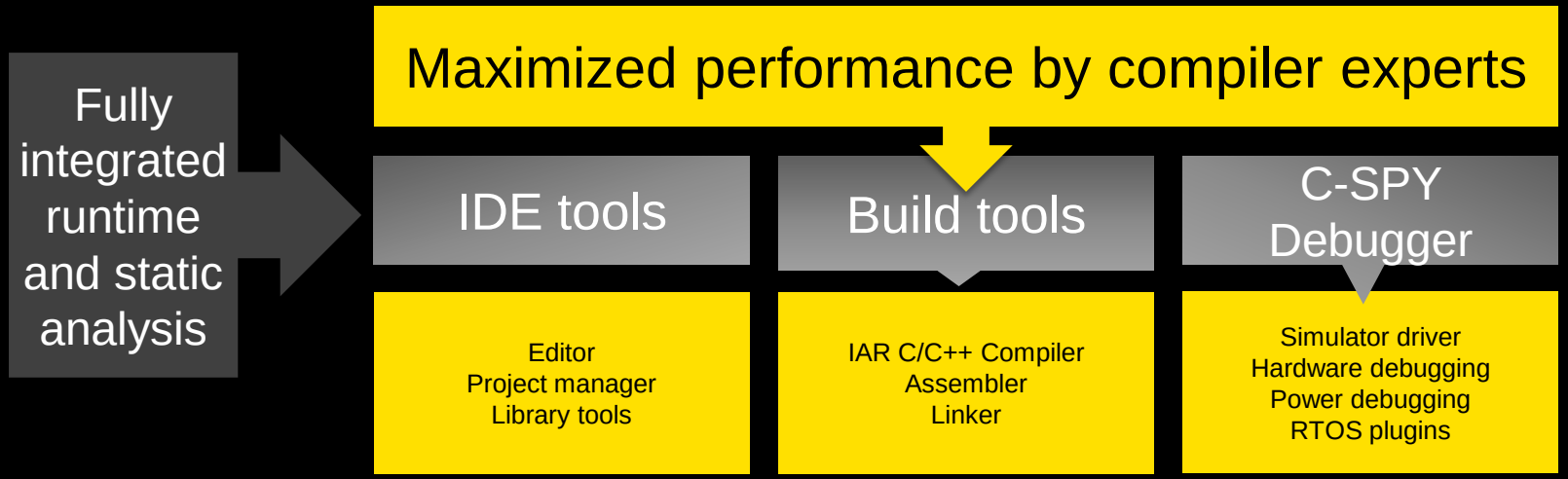
All software contains bugs

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





Integrated analysis tools

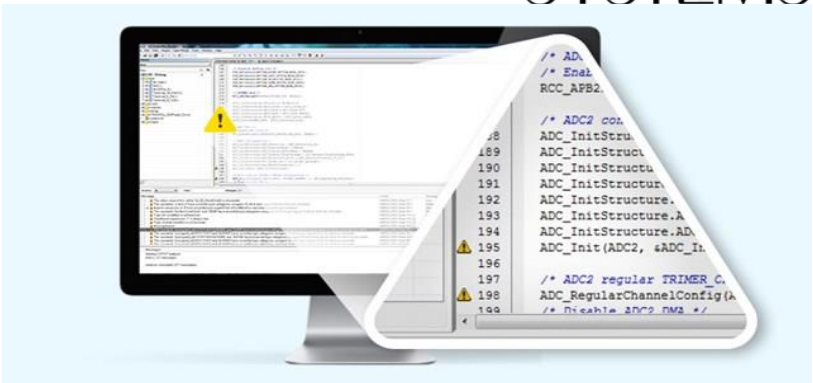


We enable developers to ***take full control of their development*** *and gain efficient, adaptable workflows delivering dependable products.*

C-STAT

C-STAT Overview

- Complete static analysis tool
- Includes:
  - MISRA-C: 2004
  - MISRA-C: 2012
    - Improved version of the 2004 edition
  - MISRA-C++: 2008
    - 165 checks to address C++ specific issues
  - Standard checks
    - More than 200 additional checks for C and C++ to address issues covered by CWE (common weakness enumeration), CERT C coding standard etc.
    - Categories like Array bounds, Arithmetic errors
- Support for export/import selection of checks
- Fully integrated in the IAR Embedded Workbench IDE
- Can also be invoked from command line and through Eclipse plugin
- Generation of html report



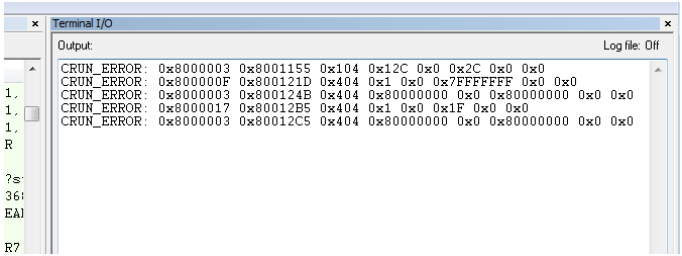
C-RUN

C-RUN

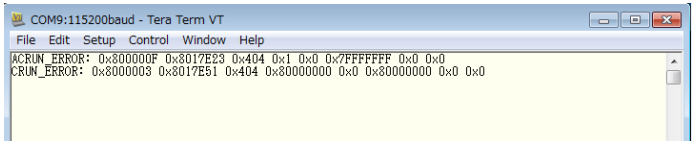
- Detection of data manipulation issues **during runtime**
- Arithmetic, bound and heap checking
- Very efficient instrumentation of compiled code
- Fully integrated in the Embedded Workbench for ARM and Renesas RX

C-RUN in Standalone mode

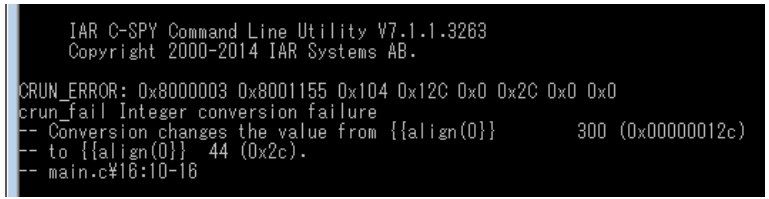
- Redirect the C-RUN messages to Terminal I/O



- Redirect C-RUN messages to UART



- Redirect C-RUN messages to RAM/SRAM
- Parse the C-RUN raw data with cspybat.exe



Demo

Coffee Break

The importance of Functional Safety

What are the benefits of Functional Safety?

IAR SYSTEMS

- Reduce liability risks in your application
- Reduce odds of product recall
- Reduce number of firmware updates
- Ensure compliance with international standards

- Protects your company’s reputation
- Also protects your company’s bottom line

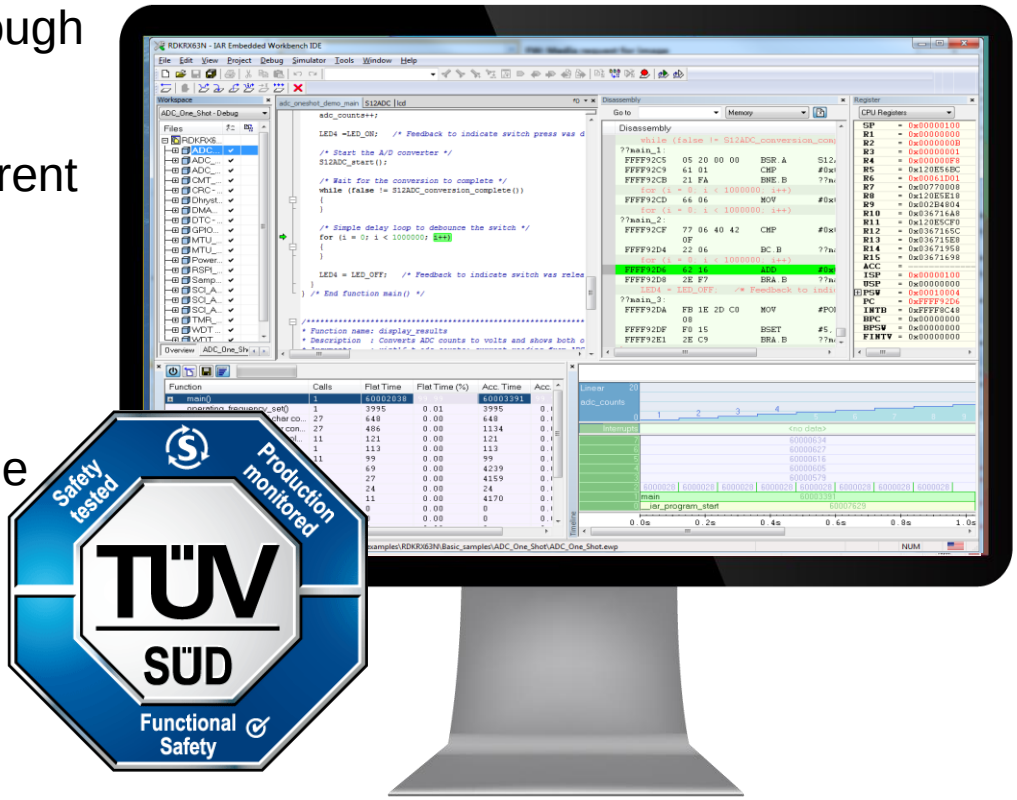
What do these safety standards do?

- Assign safety goals
- Best practice to reduce systematic failures
- Diagnostic measures to detect random failure
- Ongoing procedures after product deployment
- They outline how to identify and deal with risks
- They require FS-certified tools

What does functional safety certification for my tools mean?

The development tool has gone through a rigorous qualification process

- Development processes for different functional safety standards
- Validation of compliance with different language standards
- Handling issues reported from the field and updated to the users



What does functional safety certification for my tools mean?

It also means:

- There are specific processes and metrics in place to handle issues reported from the field and how users are updated about known issues
- A safety manual is provided to show proof-of-compliance with standards and how to operate the development tool to comply with FS standards
- Assessment takes into consideration how many developers are using the toolchain to ensure it has a broad user-base

How does a functional safety-certified tool speed up my certification process?

- Removes the requirement to prove that the toolchain complies with the safety standard.
- Software Development Lifecycle (SDLC) can focus on finding bugs instead of compiler issues
- Certified service packs provide worry-free support

How much time and money can it save me?

- Avoid the back-and-forth with your certifying entity
- Tool certifications can take up to 6-12 months and occupy several employees (usually 2-5)
- Avoid extra testing requirements on each project
- Frees people from also needing to prove the development tools
- Numbers depend on the SIL level, but the cost could be upwards of $100k US

Solutions for safety-critical applications

Certified toolchain

- A special functional safety edition of IAR Embedded Workbench

Simplified validation

- Functional Safety certificate from TÜV SÜD
- Safety report from TÜV SÜD
- Safety guide

Guaranteed support through the product life cycle

- Prioritized support
- Validated service packs
- Regular reports of known problems

Available for:

ARM
Renesas RX
Renesas RL78
Renesas RH850

Validated according to:

IEC 61508
ISO 26262
EN 50128 (ARM, RH850)
IEC 62304 (RX, ARM (2018))

Certifications

IAR Embedded Workbench for ARM V7.40.6

IAR Embedded Workbench for RX V2.42.4

IAR Embedded Workbench for RL78 V1.40.7

IAR Embedded Workbench for RH850 V 1.40.3

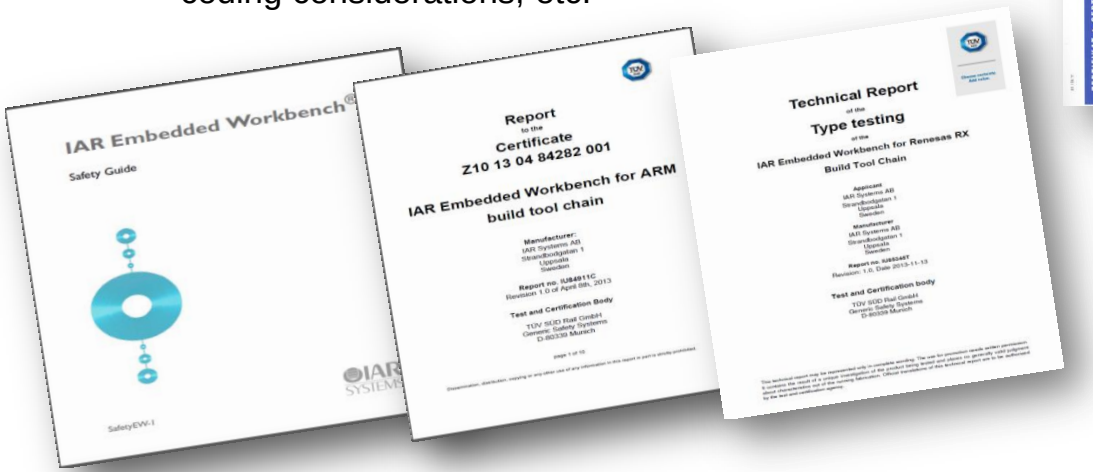
Certified for safety-related software development for each Safety Integrity Level (SIL) according to IEC 61508 and each Automotive Safety Integrity Level (ASIL) of ISO 26262 without further tool qualification

The tools for ARM and RH850 are also certified for EN 50128, a European railway standard derived from IEC 61508.

The certification validates the quality of IAR Systems' entire development processes, as well as the delivered software.

Simplified validation

- Functional safety certificate from TÜV SÜD
- Safety report from TÜV SÜD
- Safety Guide
  - Complement to the IAR Embedded Workbench user guides
  - Highlights issues to be considered when using the build toolchain for projects with functional-safety requirements
  - Includes system considerations, implementation and coding considerations, etc.



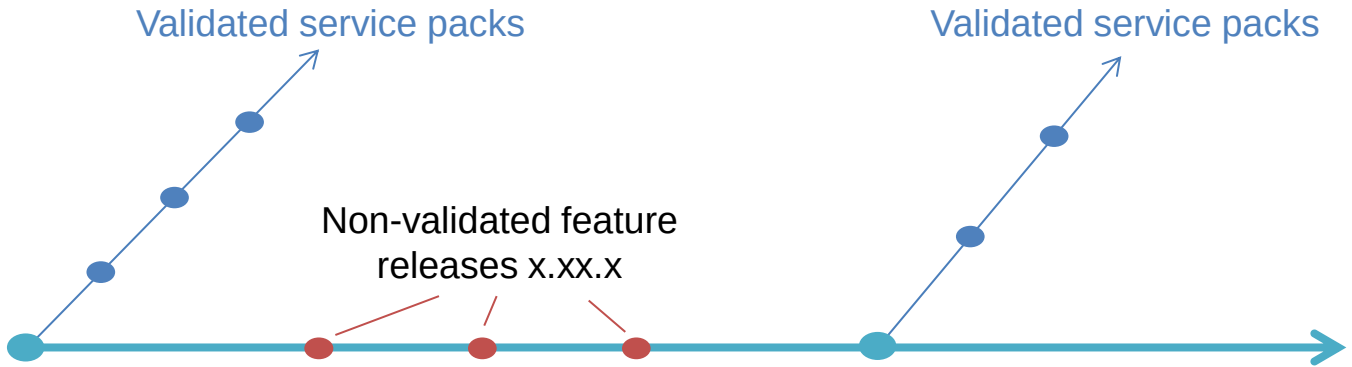
Support and updates

Functional Safety Support and Update Agreement (SUA)

- Guaranteed support for the sold version for the longevity of the contract
- Prioritized technical support
- Validated service packs
- Regular reports of known deviations and problems
- Included for the first year

Extensive technical support
when and where you need it provided by support offices worldwide

Validated product versions



- For a certified product, a new certified version is released approximately every 12-18 months
- A certified version is considered a ”frozen” version, on which bug fixes are applied in terms of validated service packs
- No new product features are added to a certified version or the corresponding service packs

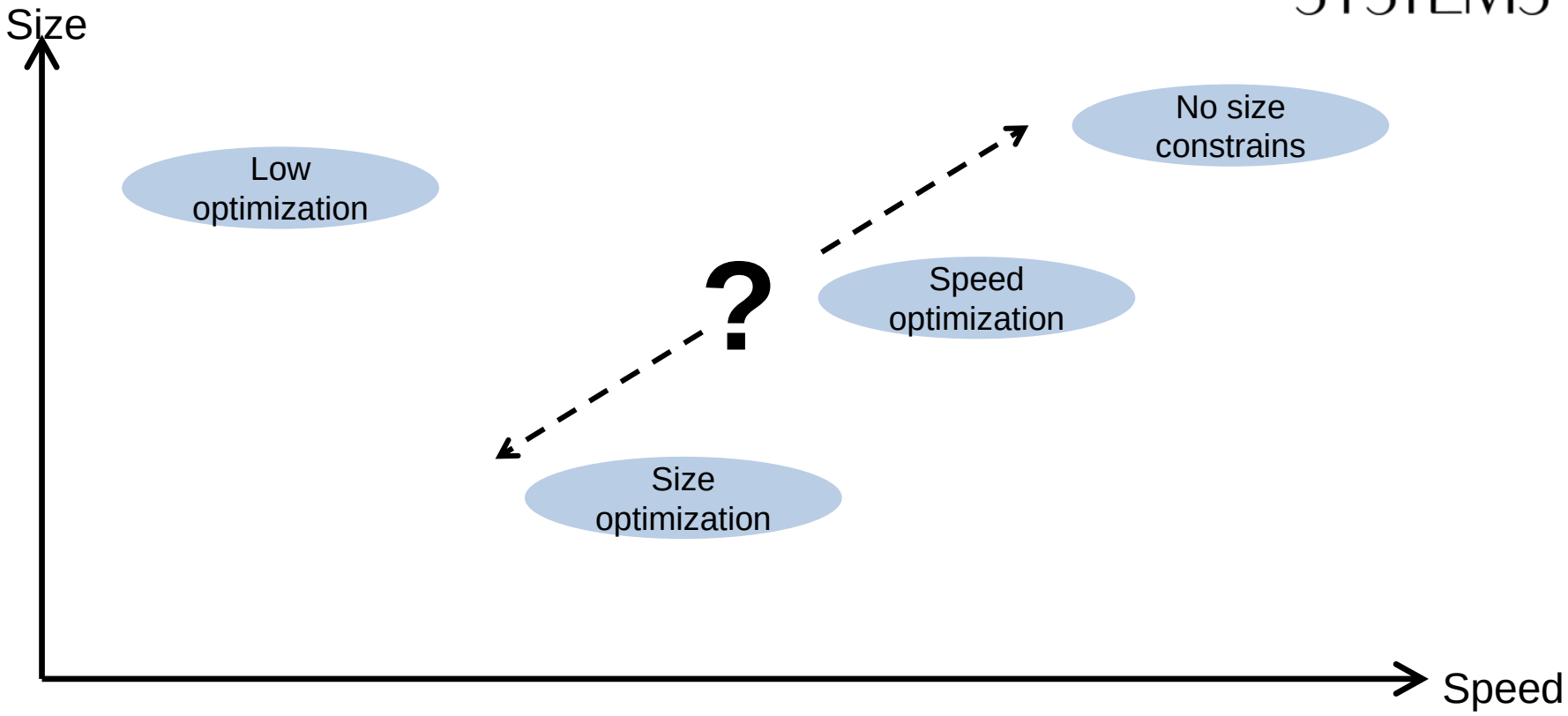
Benchmarks

Benchmarks

Benchmark code

Your code

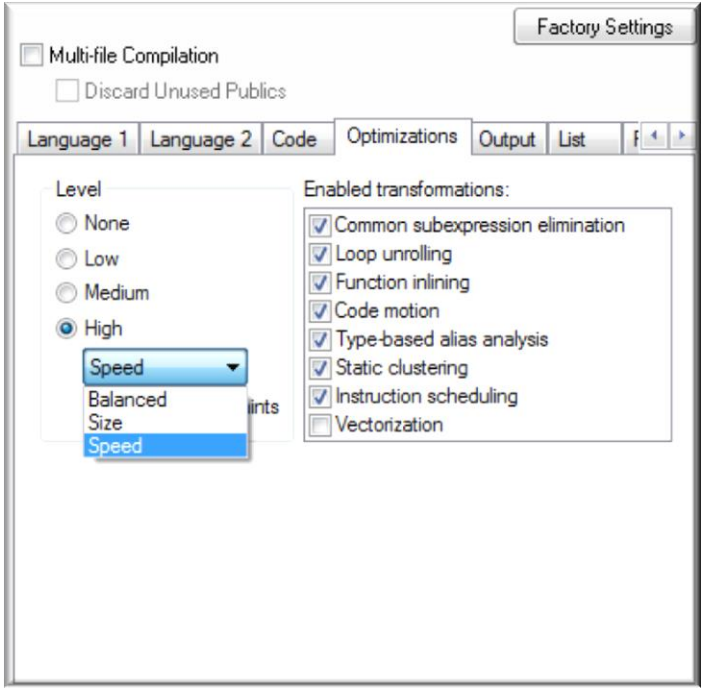
Find the sweetspot



Optimization settings

- Choices:
  - Minimize code size (“size”)
  - Minimize execution time (“speed”)
  - Low, Medium, High
  - Enable/disable individual transformations

- Use highest settings!
- Settings only approximate
  - Best guess by compiler writer
  - Test several settings, check the results
  - Enable the productive transformations



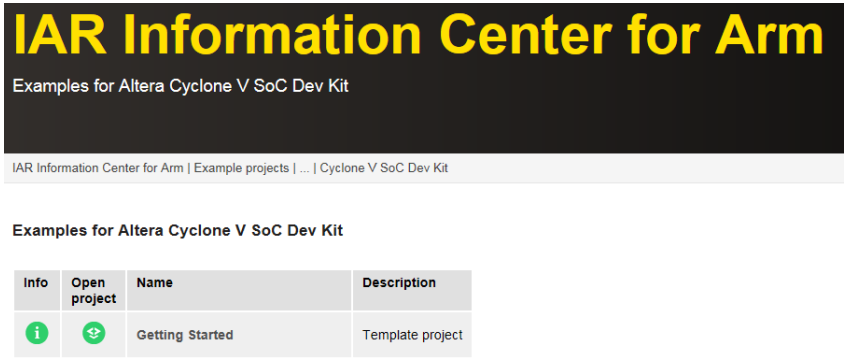
Coremark - STM32

Clear	Processor	Compiler	Operating Speed in Mhz	CoreMark /MHz (*)	CoreMark (*)	EEMBC	Comments	Date Submitted
	STMicroelectronics STM32H743	IAR C/C++ Compiler for ARM 7.60.1.11101 (7.60.1.11101)	400	5.05	2020.55	✓	comment	10/24/16
	STMicroelectronics STM32F746NGH6	IAR C/C++ Compiler 7.60.2.11341	216	5.01	1082.41		comment	07/13/16
	STMicroelectronics STM32F756NGH6	IAR ANSI C/C++ Compiler V7.30.0.7673/W32 for ARM	200	5.01	1001.79	✓	comment	09/24/14
	STMicroelectronics STM32L476	IAR ANSI C/C++ Compiler V6.60.1.5097 for ARM	80	3.42	273.55	✓	1	02/17/15
	STMicroelectronics STM32F427IGT6	IAR 6.60	180	3.38	608.19		comment	07/22/13
	STMicroelectronics STM32F417IGt6	IAR-EWARM-6.50	168	3.37	565.73		comment	11/20/12
	STMicroelectronics STM32F446RE	IAR for ARM, Version 7.70.1	180	3.35	602.44		2	05/02/17
	STMicroelectronics STM32L476	IAR ANSI C/C++ Compiler V6.60.1.5097 for ARM	80	3.35	267.80	✓	comment	02/04/15
	STMicroelectronics STM32L152	IAR ANSI C/C++ Compiler V6.60.1.5097 for ARM	16	3.34	53.36	✓	comment	01/12/15
	STMicroelectronics STM32L476	IAR ANSI C/C++ Compiler V6.60.1.5097 for ARM	80	3.32	265.61	✓	comment	02/04/15
	STMicroelectronics STM32F417IGt6	IAR-EWARM-6.40	168	2.98	501.85	✓	comment	06/05/12
	STMicroelectronics STM32L152ZDT6	IAR-EWARM-6.40.3	16	2.93	46.84		comment	10/10/12
	STMicroelectronics STM32F417IGt6	GreenHills Multi 6.1 Compiler 2012	168	2.91	488.59	✓	comment	11/29/11
	STMIcroelectronics STM32L152	IAR ANSI C/C++ Compiler V6.60.1.5097 for ARM	32	2.89	92.36	✓	comment	01/12/15
	STMicroelectronics STM32F417IGt6	GreenHills 6.0.0	168	2.87	481.61		1	11/29/11
	STMicroelectronics STM32F417IGt6	GreenHills Multi 6.1 Compiler 2012	168	2.81	472.96	✓	comment	11/29/11
	STMicroelectronics STM32F417IGt6	GreenHills 6.0.0	168	2.79	469.17		1	11/29/11
	STMicroelectronics STM32L152ZDT6	IAR-EWARM-6.40.3	32	2.61	83.64		comment	10/12/12
	STMicroelectronics STM32L053	IAR ANSI C/C++ Compiler V7.30.0.7673 for ARM	16	2.49	39.91	✓	comment	01/12/15
	STMicroelectronics STM32L053	IAR ANSI C/C++ Compiler V7.30.0.7673 for ARM	32	2.35	75.18	✓	comment	01/12/15
	STMicroelectronics STM32F051C8	IAR 6.60	24	2.33	56.05		comment	07/24/13
	STMicroelectronics STM32F051C8	IAR 6.60	48	2.20	105.61		comment	07/24/13

IAR Support for Cortex A9 based FPGA SoCs

How to use IAR for FPGA SoCs

- Altera and Xilinx SoCs based on Cortex-A9 is supported by Embedded Workbench
- Example projects provided for single or dual core configurations
- Work with Altera/Xilinx tools to create BSP and code bare metal or RTOS app in Embedded Workbench



Why use IAR for FPGA SoCs

- Bare metal app or RTOS
- Supported tools
- Projects with safety requirements
  - Combine for example with ThreadX
- Highly optimized code needed

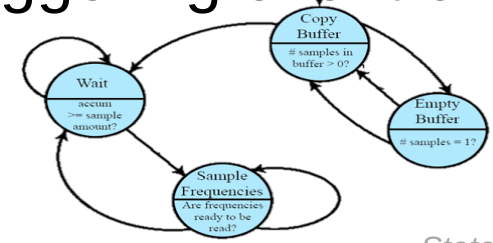
Demo

Visual State in practice

What is state machine

Wikipedia definition:

A state machine, is a mathematical model of computation used to design both computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition.

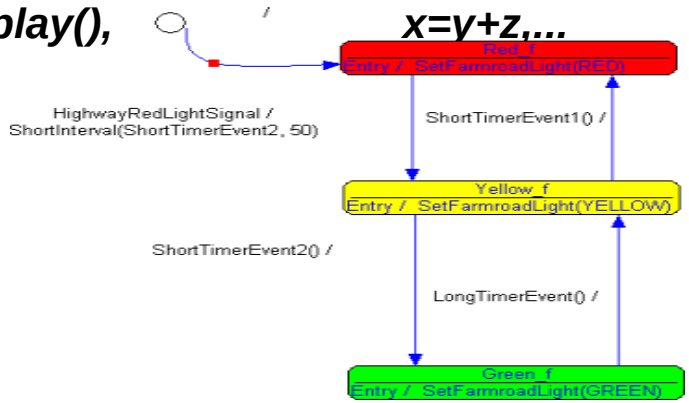


State machine basics

- A few examples
  - ***Washing machine*** *: depending on the choice the user is setting (heat, type of material, ...) the washing machine is, in a specific time and order, washing the clothes (water in, add soap, wash, water out, spin, water in, rinse, ...). You can easily describe the states and events in a state machine*
  - ***Mobile phones***
  - ***Vending machines*** *(coffee, candy, cigarettes, ATM, ...)*
  - ***Operator-panels***
  - ***Sewing machines***
  - ***Controllers*** *(heat, light, humidity, ...)*
  - *etc*

State machine elements

- The central elements in state machines are:
  - *States* ***PowerOn, DoorOpen, SupplyVoltageLow,...***
  - *Events* ***eButton1Press, e100msTimeout,...***
  - *Actions* ***StartTimer(100), ClearDisplay(),***
  - *Transitions* ***”Arcs connecting states”***



State machine elements - events

- ”Things ocurring” that affect the state machine
  - Think button presses, timeouts, interrupts in general etc...
  - Does not in any way have to be interrupts; can be detected by any method, like polling or combined from several sources
- An event occurs momentarily and asynchronously

PowerIsOn, DoorIsOpen, SupplyVoltageLow,...

State machine elements - states

- Describes what the system is ’doing’ between two events. A state is not leaved until being told to move to another state.

PowerIsOn, DoorIsOpen, SupplyVoltageLow,...

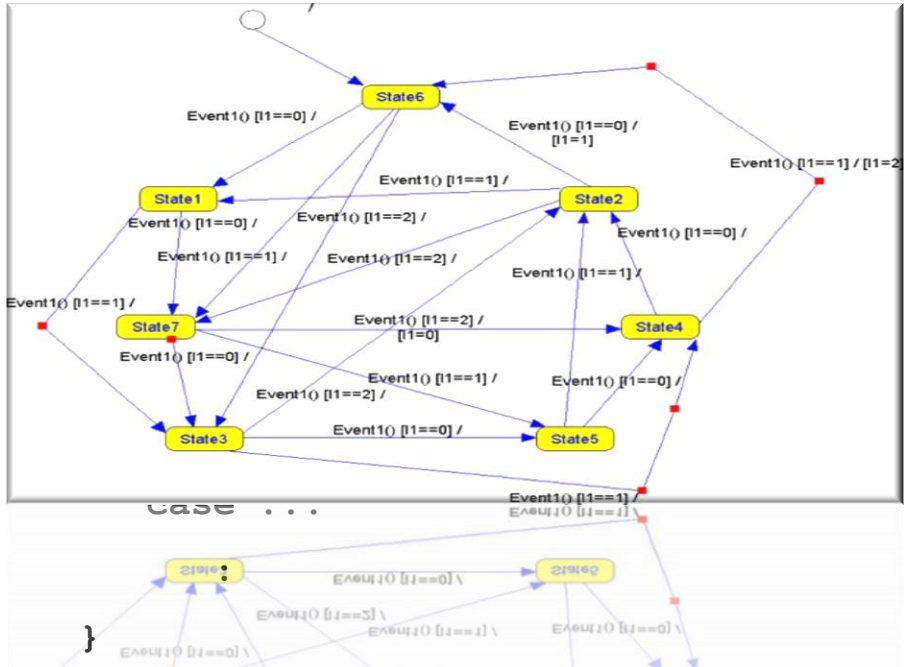
State machine elements- actions

- Describes how the system affects the environment as the reaction to an event
- In theory an action occurs momentarily and without interruption
- Happens on transitions (or when entering/exiting states)

StartTimer(100), ClearDisplay(), x=y+z,...

State machines

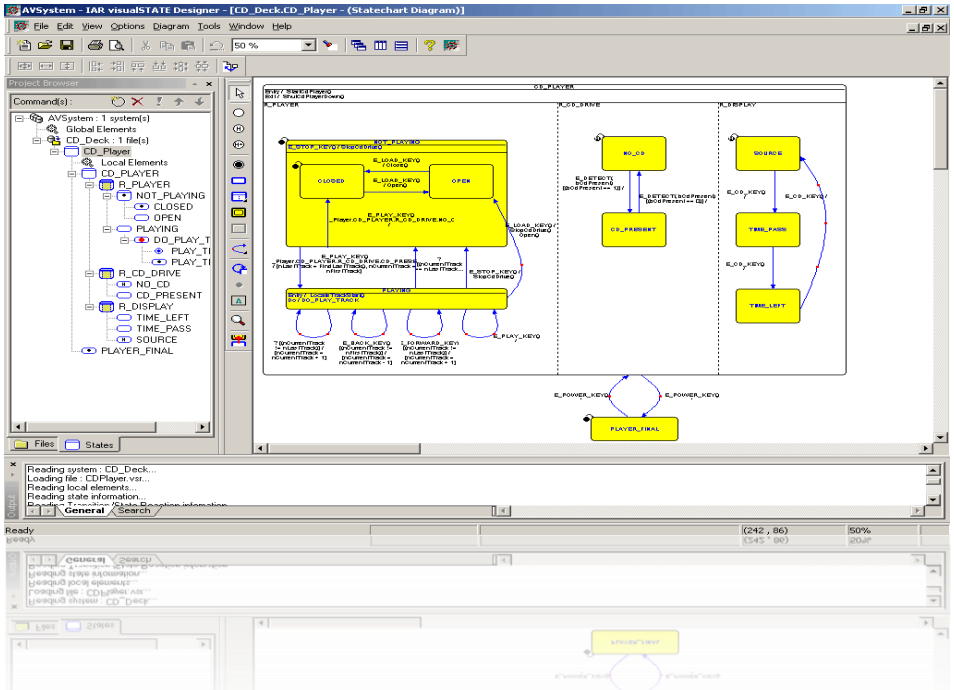
Manual coding using switch-case, if..else, tables etc. works to some point.



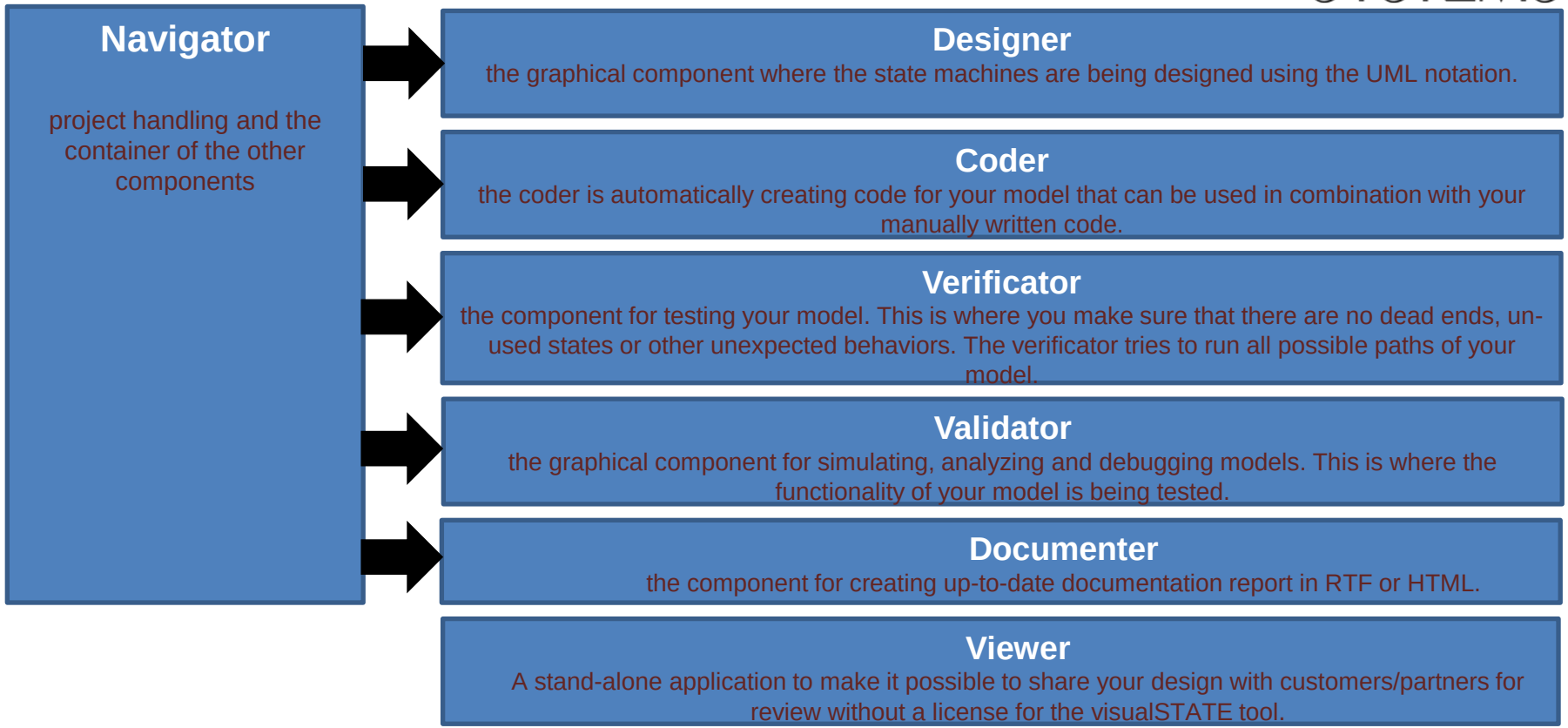
- As complexity increases, how do I ensure that I capture the complete design in code?
- How do I restructure the code if I discover omissions or errors in the designs?
- How can I verify that there is no risk for deadlocks or ambivalence in my design?

IAR visualSTATE

- Graphical toolset for designing, testing and implementing embedded applications
- Based on the UML state machine subset
- Generates C/C++ code 100% consistent with your design
- Advanced verification and validation tools

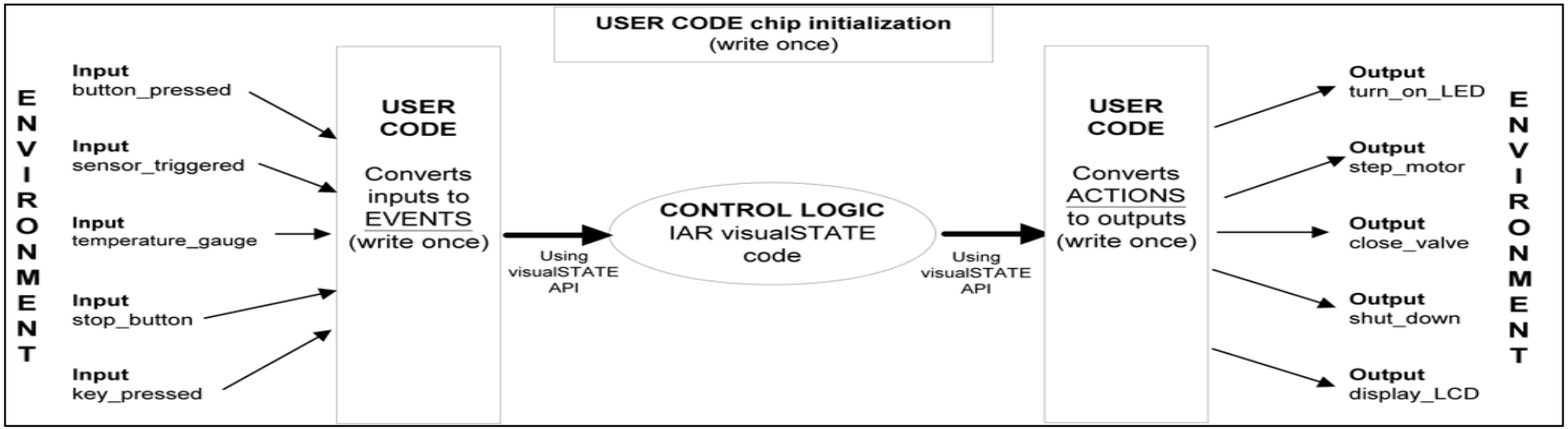


IAR visualSTATE components



Interfacing with your own C project

- Create your control logic with visualSTATE
- Add your board support files
- Write or get your device drivers/FW library



Demo

Thank you for your attention!

www.iar.com