Why do I Care if My Development Tools have a Functional Safety Certification?

Shawn A. Prestridge, Senior Field Applications Engineer

IAR SYSTEMS – A GLOBAL LEADING VENDOR

168 Employees with HQ in Uppsala, Sweden

Listed in Stockholm/Nasdaq

R&D investment 32% of revenue

32 years in the industry

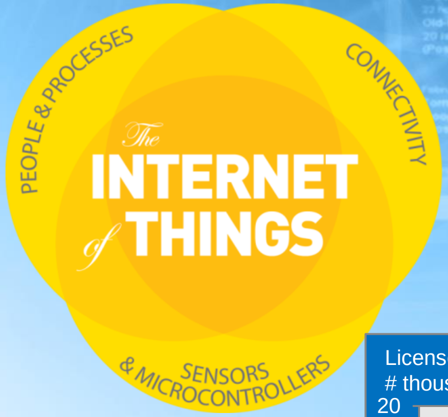
24 hour technical support in

13 languages

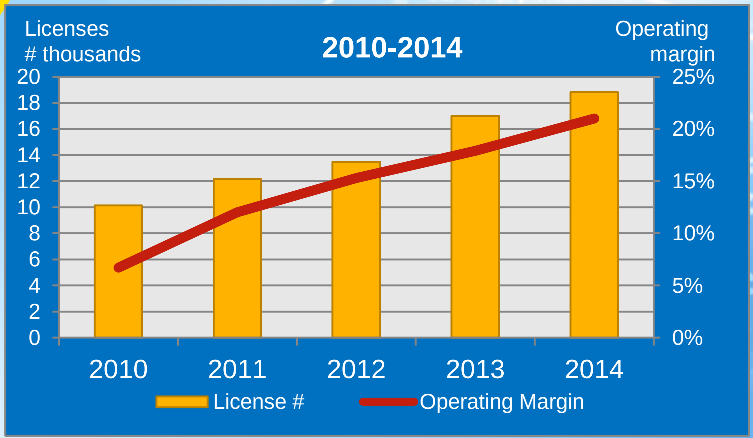
Uppsala	London
Munich	Paris
Sao Paulo	San Francisco
Tokyo	Dallas
Seoul	Boston
Shanghai	Los Angeles

+Distributor representation in

43 countries



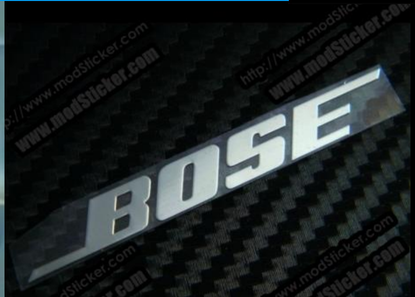
Stability and growth



LARGE AND LOYAL CUSTOMER BASE WORLDWIDE

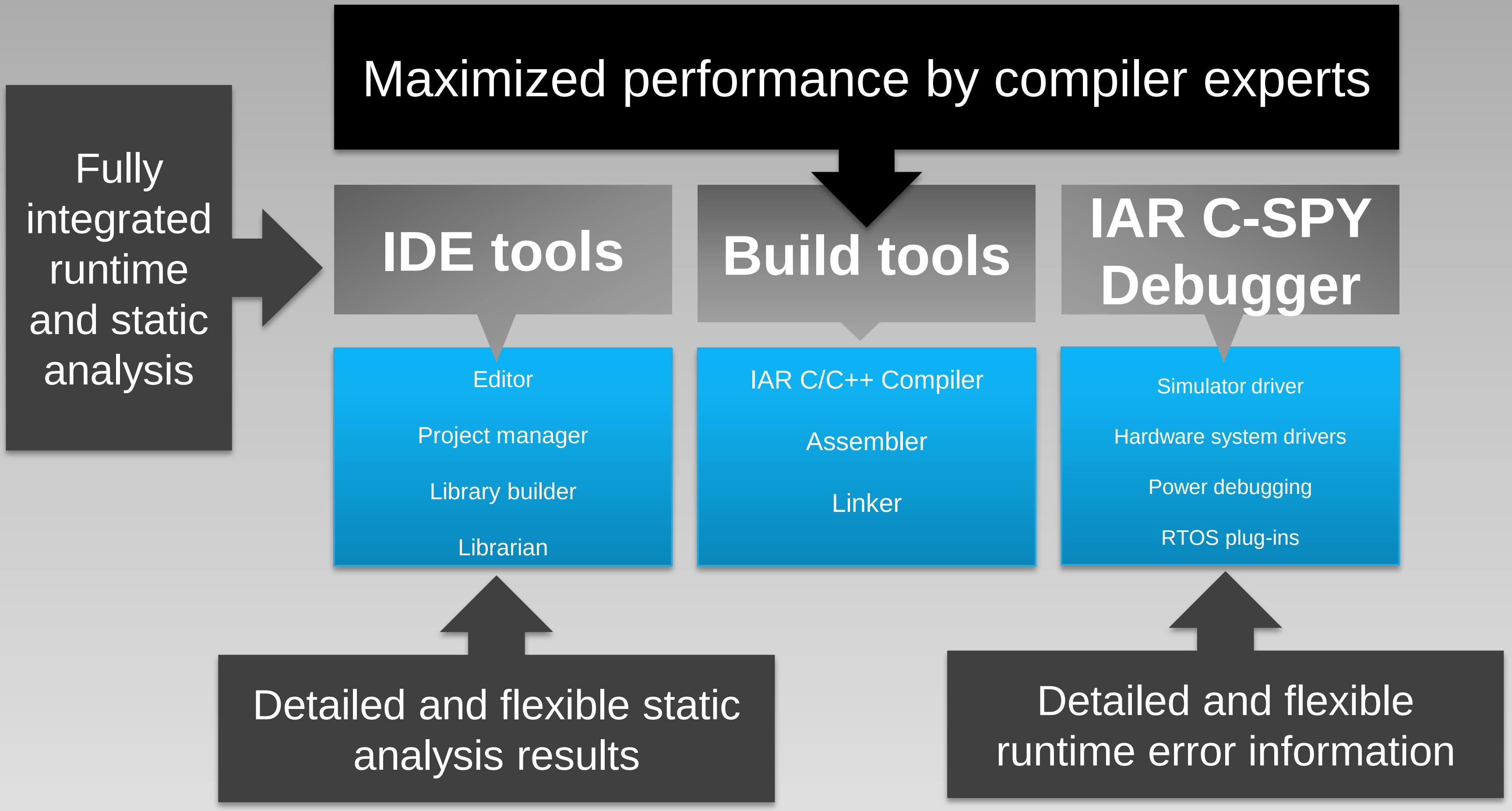
46,000+ customers

95% recurring customers
65% customers with more than one product
22,000+ support agreements
385+ enterprise agreements
200,000+ newsletter subscribers
100,000+ web visitors per month
10,000+ downloads per month
450+ partners in ecosystem



IAR EMBEDDED WORKBENCH WITH INTEGRATED ANALYSIS TOOLS

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



What are the benefits of Functional Safety?

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

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

What safety certifications are there?

The short answer is that there are hundreds

- Each one caters to a specific industry or product category
- You have to find which one is right for your product

The two most broad-reaching certifications are ISO 26262 and IEC 61508

- They cover road vehicles and electronic safety-related systems, respectively
- Most functional safety development tools aim for these two certifications because they cover almost all other certifications and industries
- Specific certifications may go above and beyond these two standards, but these two are considered the basis for many other certifications, e.g. EN 50128 for railway systems is similar to IEC 61508

What does a functional safety certification for my development tools mean?

It means that your development tool has gone through a rigorous qualification process to ensure that it produces reliable and repeatable results when compiling your code. Additionally, it means:

- Development processes are in place to manage how the tool works with specific requirements put forth by different functional safety standards
- There are test and quality measures of the tool show validation of compliance with different language standards
- 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

What do I have to do to certify my current toolchain?

The certification process is rather rigorous. The IEC 61508 standard details how support tools should be qualified in Section 7.4.4, but it is rather ambiguous on how a compiler should be qualified. Consider clause 7.4.4.10:

“The selected programming language shall have a translator which has been assessed for fitness for purpose including, where appropriate, assessment against the international or national standards.”

These and other stipulations make it difficult to certify a tool on your own and can result in significant work on your part to prove fitness and even more work to document why you think you have proven fitness! This only gets worse as you try to achieve higher and higher SIL safety levels.

I’ll just use open-source software, thanks...

Not so fast! The common argument here is that if your project uses the same uncertified tool as another project that did eventually achieve certification, then you should be covered…right?

This is definitely not the case! You are still required to prove that:

- Your project is similar enough to the other project that you use the same functionality of the toolchain as the other project (impossible without source code-level access to the other project)
- You use the toolchain in a similar manner as the one that did achieve safety certification

You usually end up having to do the same work to requalify the tool!

If I’m using a Functional Safety-certified tool, how does that speed my certification process?

The simple answer is that it removes the requirement that you have to prove your toolchain complies with the safety standard.

It also means that your test-and-fix phase of the Software Development Lifecycle (SDLC) can focus on finding bugs in your source code instead of wondering if a compiler issue is causing your problems

Certified service packs mean that you don’t have to recertify the tool to get added functionality to your toolchain

What do I get with my FS version?

The benefits of using the functional safety version are:

- A complete build chain that is certified by TŰV SŰD to comply with the requirements for tools selection in IEC 61508 and ISO 26262
- A report that accompanies the certificate that states under what circumstances the certificate is valid
- A test report that shows how the tool was tested to demonstrate compliance
- A compiler that supports C89, C99, and C++ languages (Note that the safety standards do not recommend using exceptions and RTTI in C++)
- Prequalified service packs for your FS tool to maintain certification and support for the life of the FS version (as long as there are paying customers under support contract for that version)
- Regular updates on known issues

How do I quickly certify my application?

Shawn A. Prestridge, Senior Field Applications Engineer

General outline of FS requirements

Identify what the **safety functions** are

Identify risk reduction methods for the safety functions

Verify safety function performs the way it’s supposed to

Verify that system **meets standards** by **rigorous testing**

Conduct Function Safety **audits** to assess testing evidence

General outline of FS requirements

Identify what the **safety functions** are

This is where most of the work in functional safety is done. A **safety function** puts the **machine** in a **safe state** when a **hazard occurs**

- **What** are the **hazards?**
- **What** are **appropriate responses** to the hazards**?**
- **What** is the **industry standard** for **safety functions** for my device**?**

The safety function identifies a hazardous condition and responds appropriately to make the machine safe.

Speeding the path to safety certifications

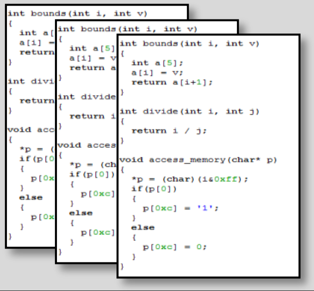
Certifications are **easier to achieve** when you can prove that your code conforms to a **coding standard**.

Testing reports show that the overall **number of defects** in the software **is low**, despite many hours of testing and **proves maturity** of **your development organization**

Code analysis also shows that your **results are repeatable** because you have a **process** in place **to find and fix defects**.

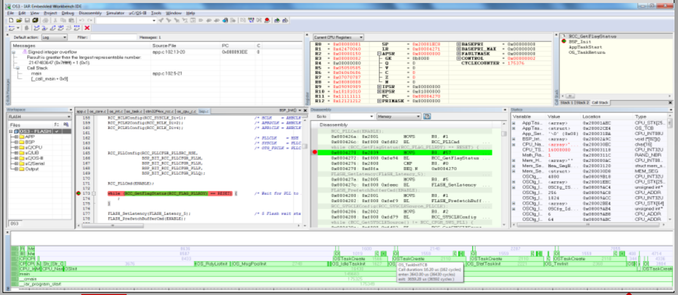
Different types of Code Analysis: Static and Dynamic

Implement your design in code

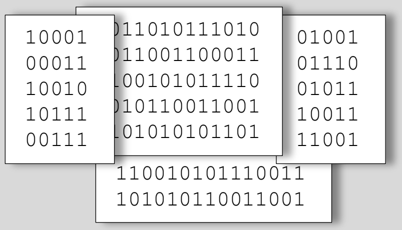


```
int bounds(int i, int v)
{
  int a[5];
  a[i] = v;
  return a[i+1];
}

int divide(int i, int j)
{
  return i / j;
}

void access_memory(char* p)
{
  *p = (char)(i&0xff);
  if(p[0])
  {
    p[0xc] = '1';
  }
  else
  {
    p[0xc] = 0;
  }
}
```

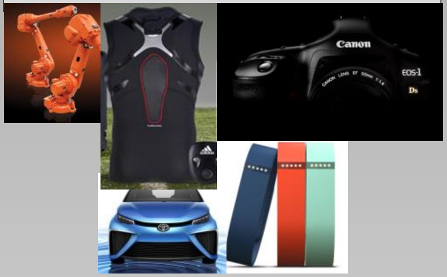
Build and debug the application



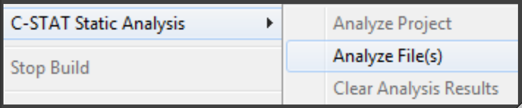
Release the application



```
10001    11010111010    01001
00011    11001100011    01110
10010    00101011110    01011
10111    10110011001    10011
00111    01010101101    11001
       110010101110011
       101010110011001
```

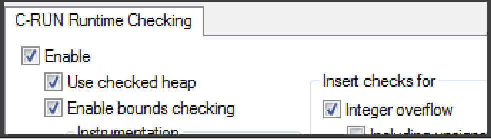


Let C-STAT analyze your code



C-STAT Static Analysis ▸
Stop Build
Analyze Project
Analyze File(s)
Clear Analysis Results

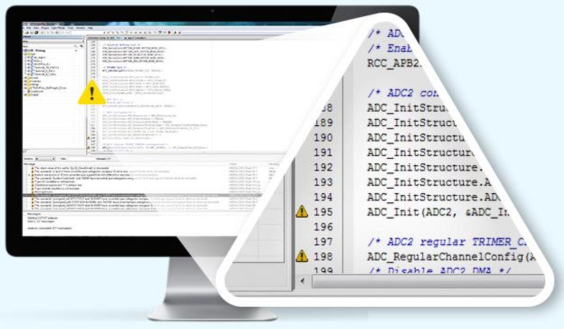
Let C-RUN analyze your project



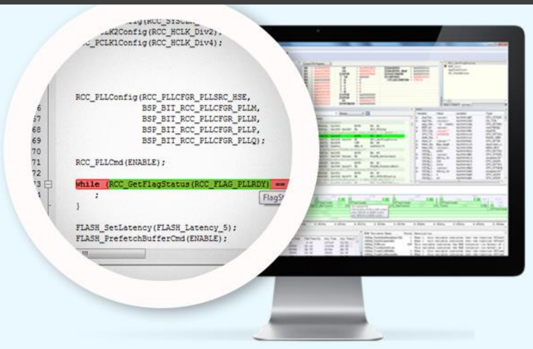
C-RUN Runtime Checking

☑ Enable
☑ Use checked heap
☑ Enable bounds checking
Instrumentation
Insert checks for
☑ Integer overflow
☐ Including unsigned

Review potential issues



Investigate runtime errors



```
LK2Config(RCC_HCLK_Div2);
_PCLK1Config(RCC_HCLK_Div4);

RCC_PLLConfig(RCC_PLLCFGR_PLLSRC_HSE,
              BSP_BIT_RCC_PLLCFGR_PLLM,
              BSP_BIT_RCC_PLLCFGR_PLLN,
              BSP_BIT_RCC_PLLCFGR_PLLP,
              BSP_BIT_RCC_PLLCFGR_PLLQ);

RCC_PLLCmd(ENABLE);

while (RCC_GetFlagStatus(RCC_FLAG_PLLRDY)
  ;
}

FLASH_SetLatency(FLASH_Latency_5);
FLASH_PrefetchBufferCmd(ENABLE);
```

Requirements → Design → Implementation → Verification → Maintenance

C-STAT STATIC ANALYSIS

ADD-ON PRODUCT AVAILABLE FOR ARM

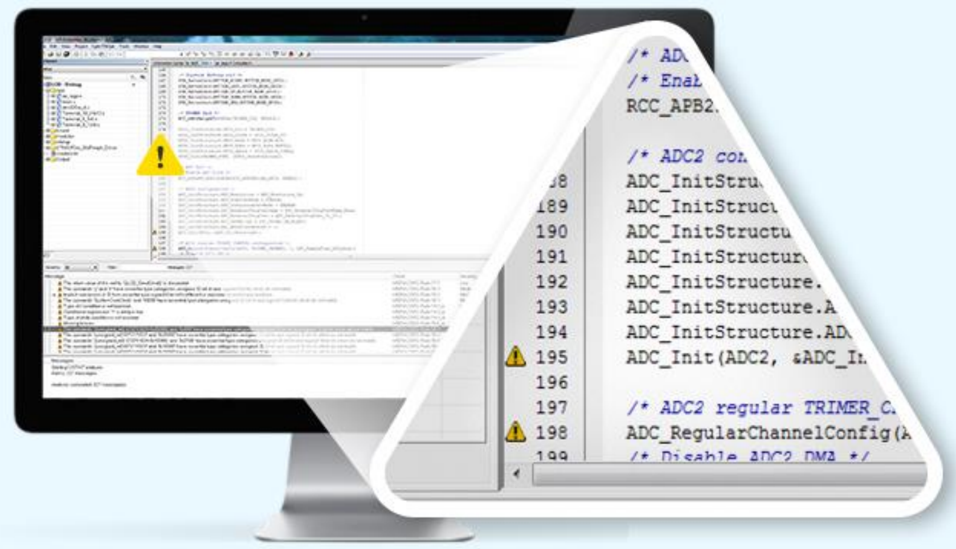
Analysis of **C and C++** code

Intuitive and **easy-to-use settings** with flexible rule selection

Checks compliance with rules as defined by MISRA C:2004, MISRA C++:2008 and **MISRA C:2012**

Includes ~250 checks mapping to hundreds of issues covered by **CWE** and **CERT C/C++**

Over 500 rules!



```
/* AD
/* Enab
RCC_APB2.

/* ADC2 con
ADC_InitStru
ADC_InitStruct
ADC_InitStructu
ADC_InitStructur
ADC_InitStructure.
ADC_InitStructure.A
ADC_InitStructure.AD
ADC_Init(ADC2, &ADC_In

/* ADC2 regular TRIMER_C
ADC_RegularChannelConfig(A
/* Disable ADC2 DMA */
```

C-RUN RUNTIME ANALYSIS

ADD-ON PRODUCT AVAILABLE FOR ARM

Support for **C and C++** code

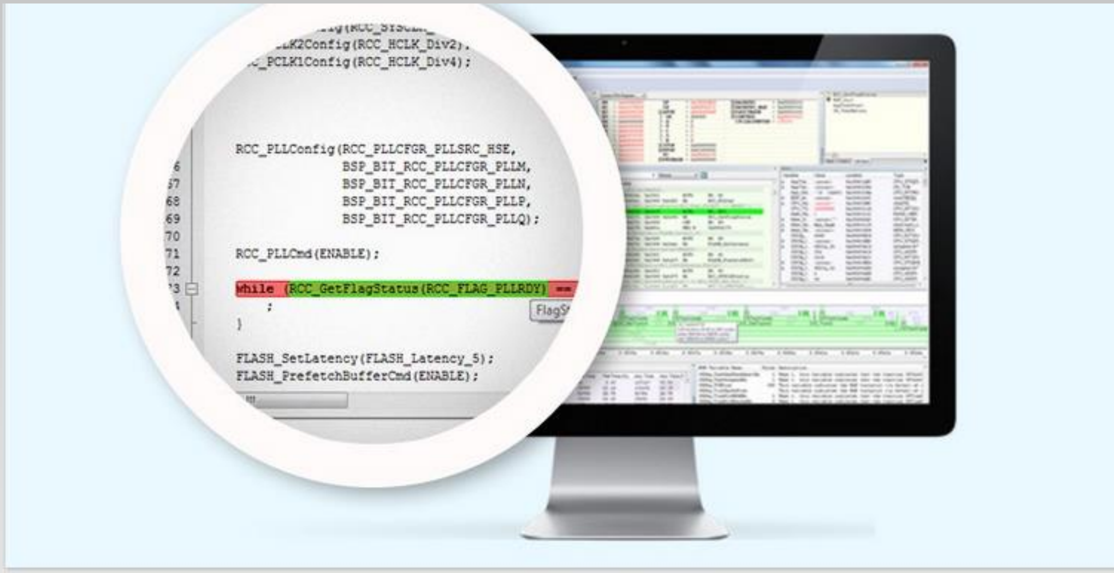
Intuitive and **easy-to-use settings**

Code correlation and graphical feedback in editor

Bounds checking to ensure accesses to arrays and other objects are within boundaries

Heap and memory leaks checking

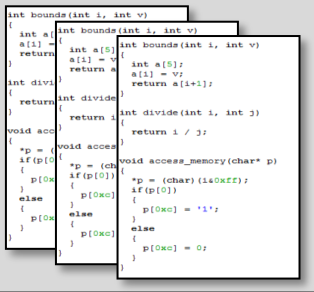
Comprehensive and **detailed** runtime error information



```
LK2Config(RCC_HCLK_Div2);
_PCLK1Config(RCC_HCLK_Div4);

RCC_PLLConfig(RCC_PLLCFGR_PLLSRC_HSE,
              BSP_BIT_RCC_PLLCFGR_PLLM,
              BSP_BIT_RCC_PLLCFGR_PLLN,
              BSP_BIT_RCC_PLLCFGR_PLLP,
              BSP_BIT_RCC_PLLCFGR_PLLQ);

RCC_PLLCmd(ENABLE);

while (RCC_GetFlagStatus(RCC_FLAG_PLLRDY) ==
    ;
}

FLASH_SetLatency(FLASH_Latency_5);
FLASH_PrefetchBufferCmd(ENABLE);
```

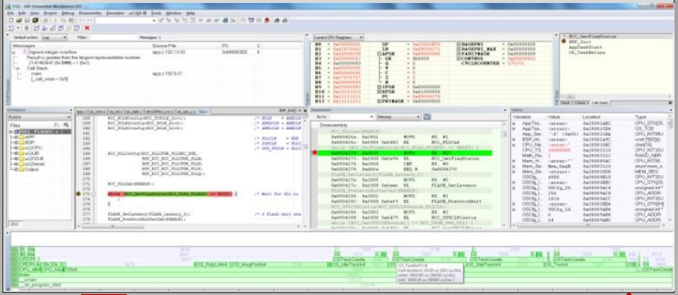
Demonstration of seamless workflow

Implement your design in code



```
int bounds(int i, int v)
{
  int a[5];
  a[i] = v;
  return a[i+1];
}

int divide(int i, int j)
{
  return i / j;
}

void access_memory(char* p)
{
  *p = (char)(i&0xff);
  if(p[0])
  {
    p[0xc] = '1';
  }
  else
  {
    p[0xc] = 0;
  }
}
```

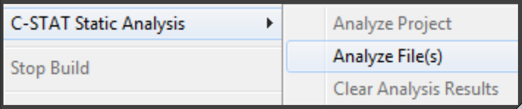
Build and debug the application



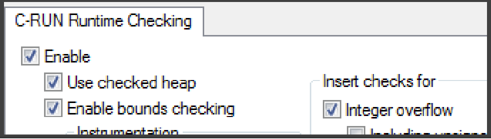
Release the application

```
10001    11010111010    01001
00011    11001100011    01110
10010    00101011110    01011
10111    10110011001    10011
00111    01010101101    11001
         110010101110011
         101010110011001
```

Let C-STAT analyze your code



Let C-RUN analyze your project



C-RUN Runtime Checking

☑ Enable

☑ Use checked heap

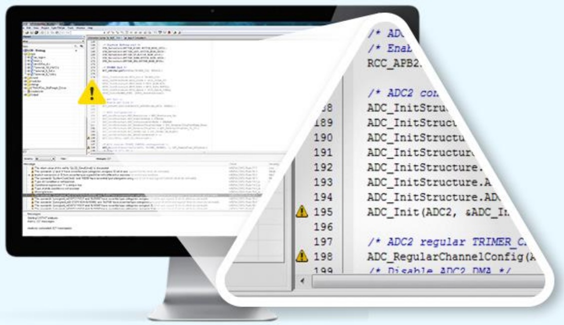
☑ Enable bounds checking

Instrumentation

Insert checks for

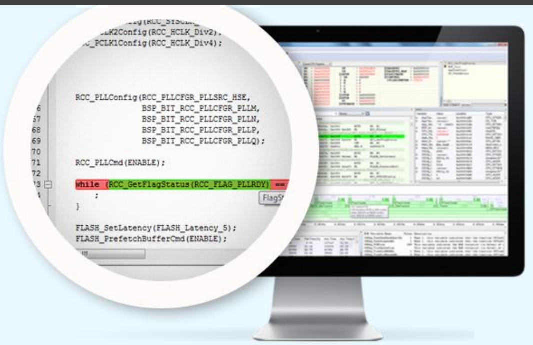
☑ Integer overflow

Review potential issues



```
/* AD
/* Enab
RCC_APB2

/* ADC2 con
ADC_InitStru
ADC_InitStruct
ADC_InitStructu
ADC_InitStructure
ADC_InitStructure.
ADC_InitStructure.A
ADC_InitStructure.AD
ADC_Init(ADC2, &ADC_I

/* ADC2 regular TRIMER_C
ADC_RegularChannelConfig(A
/* Disable ADC2 DMA */
```

Investigate runtime errors



```
LK2Config(RCC_HCLK_Div2);
_PCLK1Config(RCC_HCLK_Div4);

RCC_PLLConfig(RCC_PLLCFGR_PLLSRC_HSE,
              BSP_BIT_RCC_PLLCFGR_PLLM,
              BSP_BIT_RCC_PLLCFGR_PLLN,
              BSP_BIT_RCC_PLLCFGR_PLLP,
              BSP_BIT_RCC_PLLCFGR_PLLQ);

RCC_PLLCmd(ENABLE);

while (RCC_GetFlagStatus(RCC_FLAG_PLLRDY)
;
}

FLASH_SetLatency(FLASH_Latency_5);
FLASH_PrefetchBufferCmd(ENABLE);
```

Requirements → Design → Implementation → Verification → Maintenance

CUSTOMER SERVICES

Support and Update Agreement (SUA)

Normally 12 months SUA period

20% yearly fee (based on listprice of product)

Free software updates via “My Pages”

First class worldwide technical support by telephone, e-mail and fax

Prioritized Technical Support (PTS)

Additional support services for VLP customers

Price and offering on a case by case basis

IAR Academy

Technical training sessions including in-depth lectures and hands-on training

Customized courses covering topics tailored for your needs

SUMMARY

- The benefits of Functional Safety
- What safety certifications exist today
- What you get with a FS toolchain
- Why you don’t want to do your own tool certification
- Speeding the path to safety certification
- Seamless integration into your workflow
- Extensive customer support
- Certification is easier with a FS toolchain!