

Keeping Safe at C

IAR Systems, Shanghai
ryan.sheng@iar.com



- Introduction of IAR Systems
- Functional Safety Certificate
- C: Safe or Not
- C-STAT: Static Code Analysis
- C-RUN: Runtime Code Analysis

Introduction of IAR Systems



- Established in 1983
- Headquarter: Uppsala, Sweden
- 170+ employees
- Support for 10,000+ devices
 - 3000+ ARM devices
- A world-leading embedded development tools vendor
- Main products
 - IAR Embedded Workbench: C/C++ Compiler & Debugger Tools
 - IAR visualSTATE: State-Machine Modeling & Software Design Tools
 - IAR I-jet / I-scope / JTAGjet: Debugging & Trace Probes
- China office
 - Shanghai, 021-63758658

Strategic collaboration with Freescale

- Long partnership with Freescale
- Initiated close cooperation around HC12 & S12
- EWCF: released on 2007
- EWS08: released on 2008
- EWARM is the most widely used commercial tool chain for ARM-based processors
- Expand the Freescale ecosystem
- IAR Embedded Workbench
 - EWHCS12: HC12 & S12 MCU
 - EWCF: ColdFire & ColdFire+ MCU/MPU
 - EWS08: S08 MCU
 - EWARM: Kinetis, i.MX, Vybrid, MC1322x, ...



K24
K30
K40
K50
K60
K70
KE02
KE04
KE06
KE14
KE15
KEAZ
KL02
KL03
KL04
KL05
KL14
KL15
KL16
KL17
KL24
KL25
KL26
KL27
KL33
KL34
KL36
KL43
KL46
KM13
KM14
KM32
KM33
KM34
KM38
KV10
KV30
KV31
KV40
KV43
KV44
KV45
KV46
KW01

IAR Embedded Workbench for ARM



IDE

Editors

Source code control systems

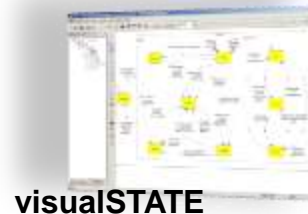
PLUGINS

IAR Embedded Workbench IDE


IDE tools	Build tools	IAR C-SPY Debugger
Editor	IAR C/C++ Compiler	Simulator driver
Project manager	Assembler	Hardware system drivers
Library builder	Linker	Power debugging
Librarian		RTOS plug-ins



Configuration tools



EWARM: Product variants

- 
- EWARM -- *Standard edition*
 - EWARM-BL -- *Baseline edition*
 - 256KB code size limitation
 - EWARM-CM -- *Cortex-M edition*
 - Only for Cortex-M0/M0+/M1/M3/M4/M7
 - EWARM-CM0 -- *CM0/CM1 only*
 - Only for Cortex-M0/M0+/M1
 - EWARMFS -- *Functional Safety*
 - IEC-61508, ISO-26262 and EN-50128

Functional Safety Certificate

What is Functional Safety?

- Simply put, it means that the overall safety of an embedded system depends on the equipment operating correctly in response to its inputs.
 - This includes erroneous inputs
 - Also includes hardware failures
- Traditionally, only safety-critical industries have been interested in functional safety.
 - Automotive
 - Avionics
 - Medical
- However, other industries are also seeing the benefit.

Functional Safety Standards



**Table A.3 – Software design and development –
support tools and programming language**

(See 7.4.4)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Suitable programming language	C.4.5	HR	HR	HR	HR
2	Strongly typed programming language	C.4.1	HR	HR	HR	HR
3	Language subset	C.4.2	---	---	HR	HR
4a	Certified tools and certified translators	C.4.3	R	HR	HR	HR
4b	Tools and translators: increased confidence from use	C.4.4	HR	HR	HR	HR
NOTE 1 See Table C.3.						
NOTE 2 The references (which are informative, not normative) "B.x.x.x", "C.x.x.x" in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.						
* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.						

Certified Tools from IAR Systems

Functional safety standards

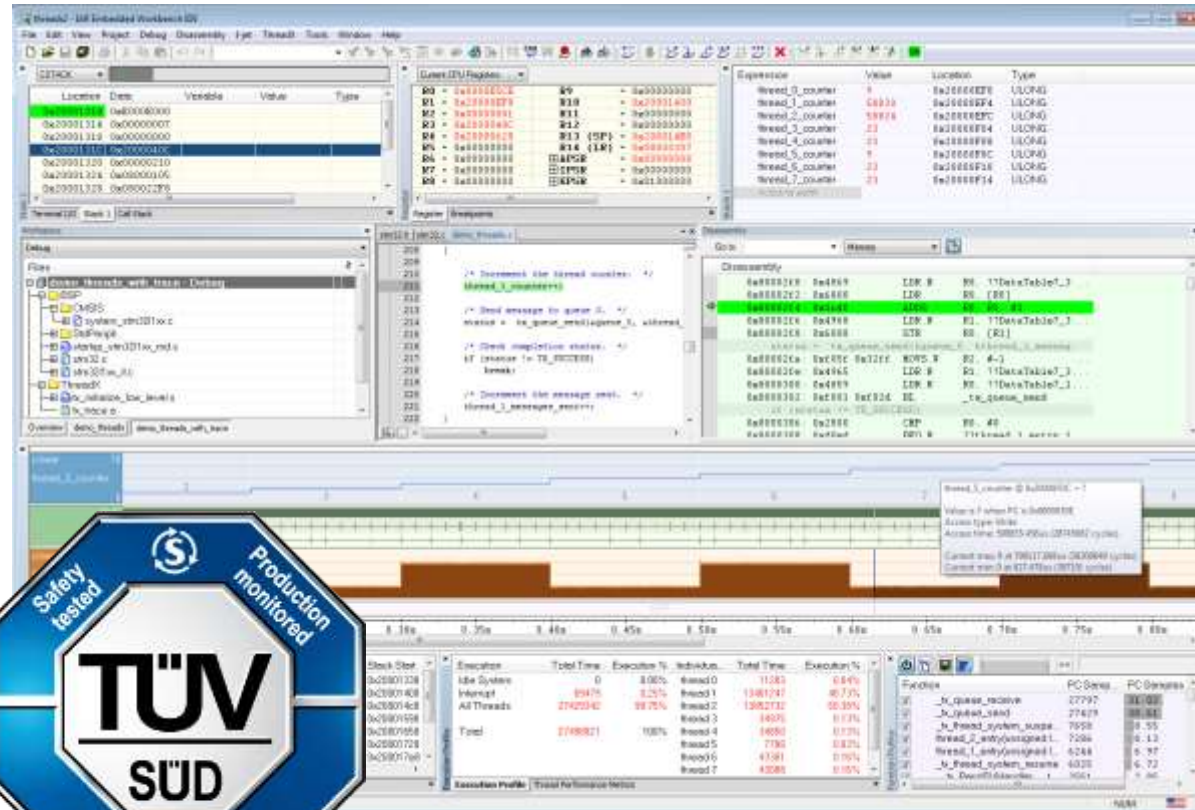
- IEC 61508
- ISO 26262
- EN 50128

Simplified validation

- Functional safety certificate
- Report to the certificate
- Safety Guide

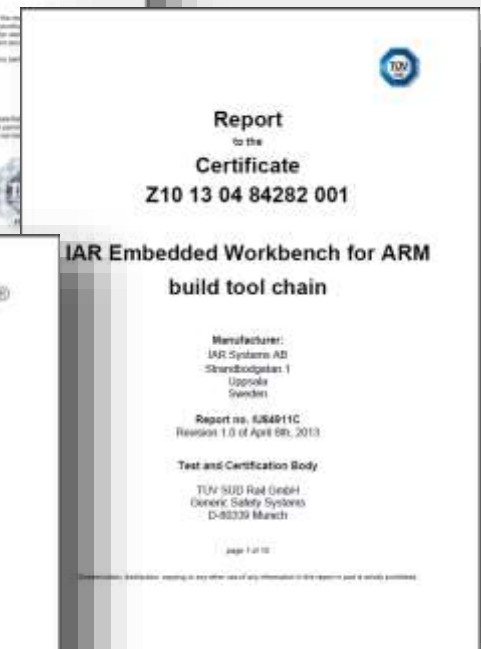
Guaranteed support and upgrade

- Cover the product life cycle
- Prioritized technical support
- Validated service packs
- Report of known problems



www.iar.com/safety

- EWARMFS
 - The Functional Safety edition of IAR Embedded Workbench for ARM
- Certified by TÜV SÜD
- Functional safety standards
 - IEC 61508-3:2010 (SIL 3)
For electrical, electronic and programmable systems in all kinds of industry.
 - ISO 26262-8:2011 (ASIL D)
Safety standard for road vehicles, derived from IEC 61508.
 - EN 50128
Safety standard for railway control and protection systems.



C: Safe or Not

**Table A.3 – Software design and development –
support tools and programming language**

(See 7.4.4)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Suitable programming language	C.4.5	HR	HR	HR	HR
2	Strongly typed programming language	C.4.1	HR	HR	HR	HR
3	Language subset	C.4.2	---	---	HR	HR
4a	Certified tools and certified translators	C.4.3	R	HR	HR	HR
4b	Tools and translators: increased confidence from use	C.4.4	HR	HR	HR	HR
NOTE 1 See Table C.3.						
NOTE 2 The references (which are informative, not normative) "B.x.x.x", "C.x.x.x" in column 3 (Ref.) indicate detailed descriptions of techniques/measures given in Annexes B and C of IEC 61508-7.						
* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. It is intended the only one of the alternate or equivalent techniques/measures should be satisfied. The choice of alternative technique should be justified in accordance with the properties, given in Annex C, desirable in the particular application.						

Table C.4.5 of IEC 61508-7 gives a general description of “suitable programming language”:

- The language should be fully and unambiguously defined.

.....

- The language should encourage:

- *The use of small and manageable software modules;*
- *Restriction of access to data in specific software modules;*
- *Definition of variable subranges; and*
- *Any other type of error-limiting constructs.*

Recommendation of languages

With the right C subset, coding standard and the use of static analysis tools, C can be a Highly Recommended programming language for all 4 levels of SIL.

Table C.1 – Recommendations for specific programming languages

9	C	R	–	NR	NR
10	C with subset and coding standard, and use of static analysis tools	HR	HR	HR	HR

NR: Not Recommended
HR: Highly Recommended

Table C.4.2 of IEC 61508-7 gives the aim and description of language subsets:

Aim:

- To reduce the probability of introducing programming faults;
- Increase the probability of detecting any remaining faults.

Description:

- The language is examined to determine programming constructs which are either error-prone or difficult to analyze, for example, using static analysis methods.
- A language subset is then defined which excludes these constructs.

C-STAT: Static Code Analysis

Code analysis tools

- Code analysis tools – Detecting erroneous code in the application
 - Static analysis tools
 - Analyze the source code without executing them
 - Runtime analysis tools
 - Analyze the source code dynamically during execution

Static analysis
applies *before*
compilation and
debug

```
int main(int i, int v)
{
    int a[10];
    while (i < 10)
        return a[i+1];
}

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

void source_error(void)
{
    int *p = (int*) (0);
    *p = 1;
    if (i < 0)
    {
        p[0] = 1;
    }
    else
    {
        p[0] = 0;
    }
}
```

C / C++ code



Build chain
(compiler &
linker)

Runtime analysis applies
after compilation and
during debug/execution



Software debugger

C-STAT: Static code analysis

- C-STAT is a static analysis tool developed by IAR Systems
 - Launched in Feb, 2015
 - Both C and C++ source code are supported
- C-STAT is an add-on product of IAR Embedded Workbench
 - Fully integrated
 - No additional installation
 - No separate license
 - Cannot work with 3rd-party compiler & debugger tools
- Target support
 - IAR Embedded Workbench for ARM, from version 7.40
 - IAR Embedded Workbench for TI MSP430, from version 6.30
 - IAR Embedded Workbench for Atmel AVR32, from version 4.30

C-STAT: What does it check



- Common Weakness Enumeration
- cwe.mitre.org
- An unified and measurable set of software weaknesses.
- Enumerate design and architecture weaknesses, as well as low-level coding errors.



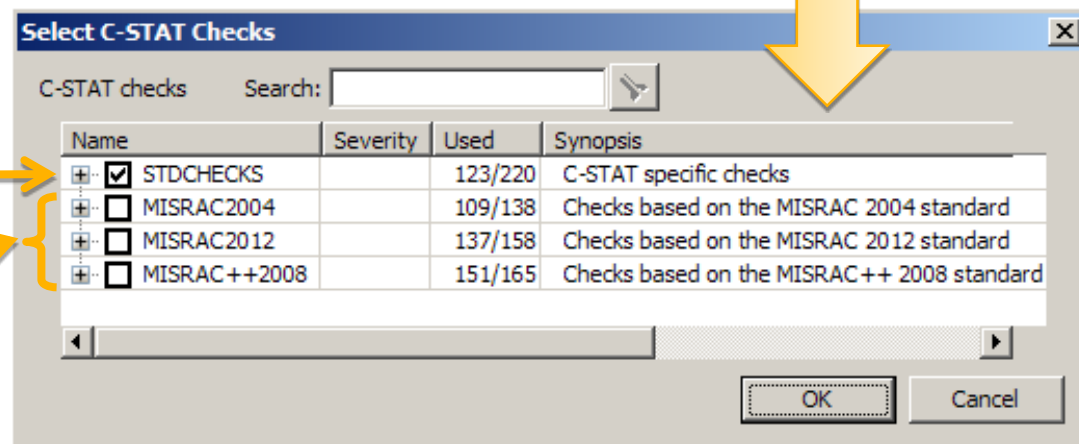
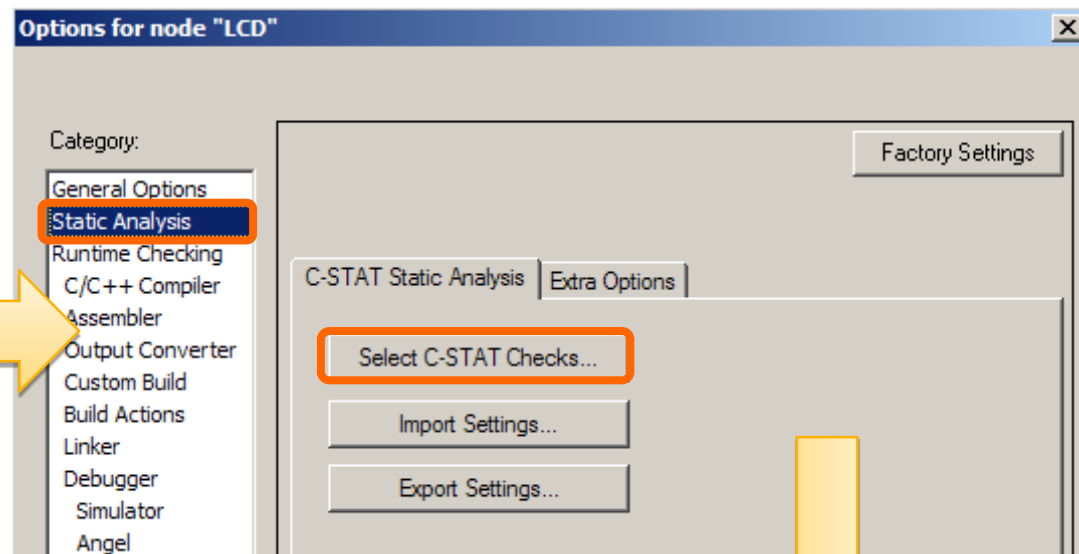
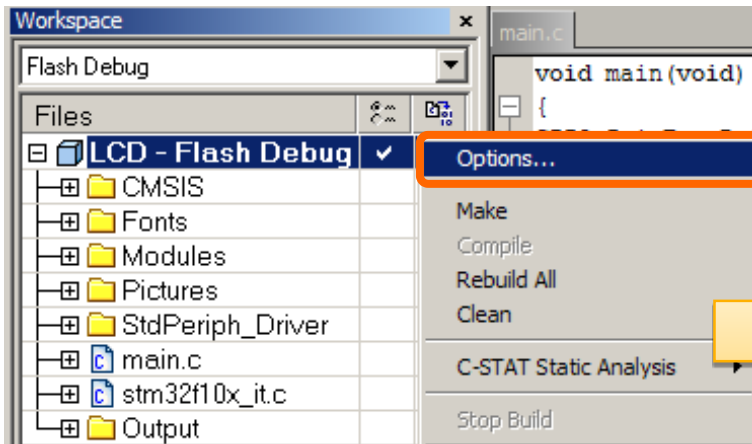
- Computer Emergency Response Team
- www.cert.org
- C/C++ secure coding standards, identifying insecure constructs which could expose a weakness or vulnerability in the software.
- Guidelines to avoid implementation, coding as well as low-level design errors.

- Motor Industry Software Reliability Association
- www.misra.org.uk



- 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.
- MISRA C++:2008: Identify unsafe code constructs in the 1998 C++ standard.

C-STAT options in IAR EWARM



CWE/CERT rules

MISRA C/C++ rules

C-STAT: Rules configuration

Select C-STAT Checks

C-STAT checks Search:

Name	Severity	Used	Synopsis
<input type="checkbox"/> MISRAC2012		137/158	Checks based on the MISRAC 2012 standard
<input checked="" type="checkbox"/> MISRAC2012-Dir-4		2/6	Code design
<input checked="" type="checkbox"/> MISRAC2012-Dir-4.3	Low		Inline asm statements that are not encapsulated in functions
<input type="checkbox"/> MISRAC2012-Dir-4.4	Low		To allow comments to c
<input type="checkbox"/> MISRAC2012-Dir-4.6_a	Low		Uses of basic types cha
<input type="checkbox"/> MISRAC2012-Dir-4.6_b	Low		Typedefs of basic type
<input type="checkbox"/> MISRAC2012-Dir-4.9	Low		Function-like macros
<input checked="" type="checkbox"/> MISRAC2012-Dir-4.10	Low		Header files without #i
<input checked="" type="checkbox"/> MISRAC2012-Rule-1		All	A standard C environm
<input checked="" type="checkbox"/> MISRAC2012-Rule-2		4/5	Unused code
<input checked="" type="checkbox"/> MISRAC2012-Rule-3		All	Comments
<input checked="" type="checkbox"/> MISRAC2012-Rule-4		None	Character sets and lex
<input checked="" type="checkbox"/> MISRAC2012-Rule-5		All	Identifiers
<input checked="" type="checkbox"/> MISRAC2012-Rule-6		All	Types
<input checked="" type="checkbox"/> MISRAC2012-Rule-7		All	Literals and constants

Highlight a rule and press F1 to show the detailed description.

Enable or disable a set of rules or any individual rule.

IAR Embedded Workbench Help for target

Hide Locate Back Forward Home Print

Contents Index Search

Type in the keyword to find:

MISRAC2012-Dir-4.3

MISRAC2012-Dir-4.3

MISRAC2012-Dir-4.4

MISRAC2012-Dir-4.6_a

MISRAC2012-Dir-4.6_b

MISRAC2012-Dir-4.9

MISRAC2012-Rule-1.3_a

MISRAC2012-Rule-1.3_b

MISRAC2012-Rule-1.3_c

MISRAC2012-Rule-1.3_d

MISRAC2012-Rule-1.3_e

MISRAC2012-Rule-1.3_f

MISRAC2012-Rule-1.3_g

MISRAC2012-Rule-1.3_h

MISRAC2012-Rule-10.1_R2

MISRAC2012-Rule-10.1_R3

MISRAC2012-Rule-10.1_R4

MISRAC2012-Rule-10.1_R5

MISRAC2012-Rule-10.1_R6

MISRAC2012-Rule-10.1_R7

MISRAC2012-Rule-10.1_R8

MISRAC2012-Rule-10.2

MISRAC2012-Rule-10.3

MISRAC2012-Rule-10.4

MISRAC2012-Rule-10.6

MISRAC2012-Rule-10.7

MISRAC2012-Rule-10.8

MISRAC2012-Rule-11.1

MISRAC2012-Rule-11.3

MISRAC2012-Rule-11.4

MISRAC2012-Rule-11.7

MISRAC2012-Rule-11.8

MISRAC2012-Rule-11.9

MISRAC2012-Rule-12.1

Display

IAR SYSTEMS

C-STAT checks : Descriptions of checks : MISRAC2012-Dir-4.3

MISRAC2012-Dir-4.3

Synopsis

Inline asm statements that are not encapsulated in functions

Enabled by default

Yes

Severity/Certainty

Low/Medium

Full description

(Required) Assembly language shall be encapsulated and isolated

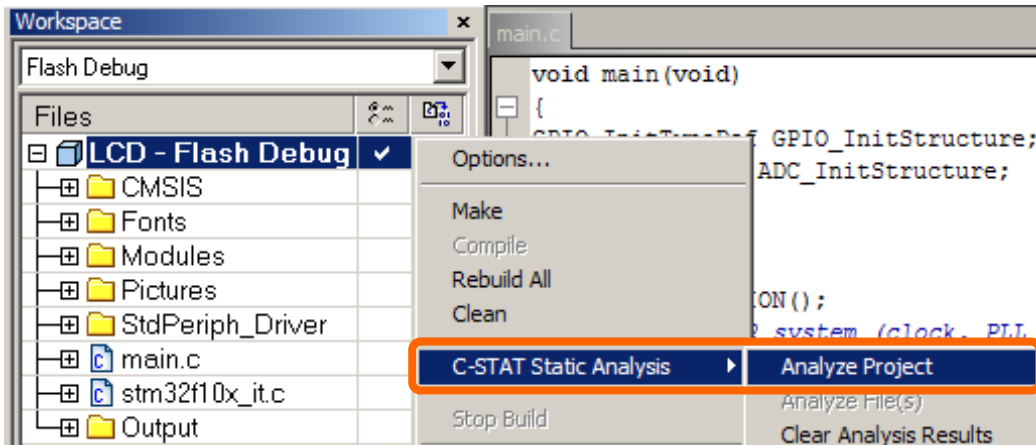
Coding standards

MISRA C:2012 Dir-4.3

(Required) Assembly language shall be encapsulated and isolated

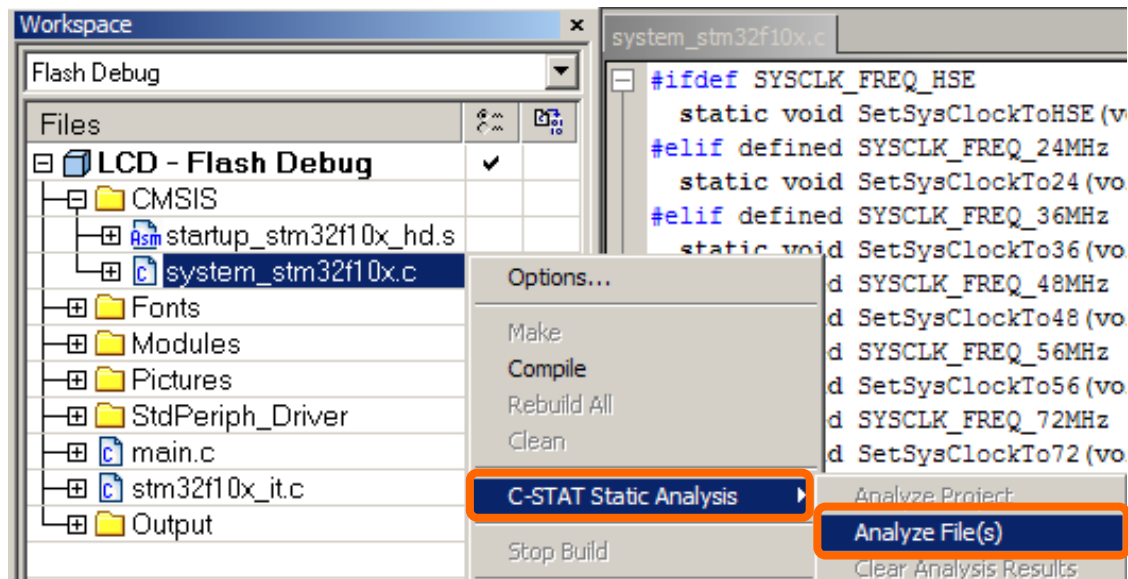
Code examples

C-STAT: Analyze the code



Analyze the whole project

Analyze an individual source file or a group of source files



C-STAT: Result of analysis

Filter the C-STAT messages by selecting a level of severity: All, Low, Medium or High.

The screenshot shows the IAR Embedded Workbench interface. On the left, the 'Flash Debug' window displays a file tree with 'stm32f10x_adc.c' selected. The main editor shows the source code for this file. A red box highlights the line: `tmpreg2 = SQR3_SQ_Set << (5 * (Rank - 1));`. Below the editor, the 'C-STAT Messages' window is open, showing a list of messages. The 'Severity' dropdown is set to 'All'. A red box highlights the first two messages in the list, which correspond to the highlighted line in the source code.

Message	Check	Severity	File
RHS argument is in interval [-5,25] which is out of range of the shift operator	ATH-shift-bounds	Medium	stm32f10x_adc.c
RHS argument is in interval [-5,25] which is out of range of the shift operator	ATH-shift-bounds	Medium	stm32f10x_adc.c
...

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.

The screenshot shows the 'IAR Embedded Workbench Help for target' window. The 'ATH-shift-bounds' check is selected in the left pane. The right pane displays the details for this check, including a synopsis, severity/certainty, and full description.

ATH-shift-bounds

Synopsis
Out of range shifts

Enabled by default
Yes

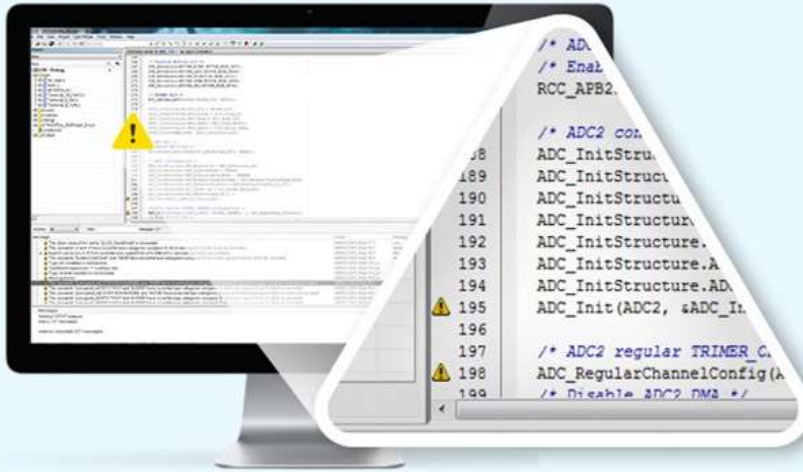
Severity/Certainty
Medium/Medium

Full description
A shift operator on an n-bit argument may only shift between 0 and n-1 bits. In this case, the right-hand operand may be negative, or too large. This check is for all platforms. The behavior in this situation is undefined; the code may work as intended, or data could become erroneous.

Coding standards
CERT INT34-C
Do not shift a negative number of bits or more bits than exist in the operand

C-RUN: Runtime Code Analysis

IAR C-STAT and IAR 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.



C-RUN: Runtime code analysis

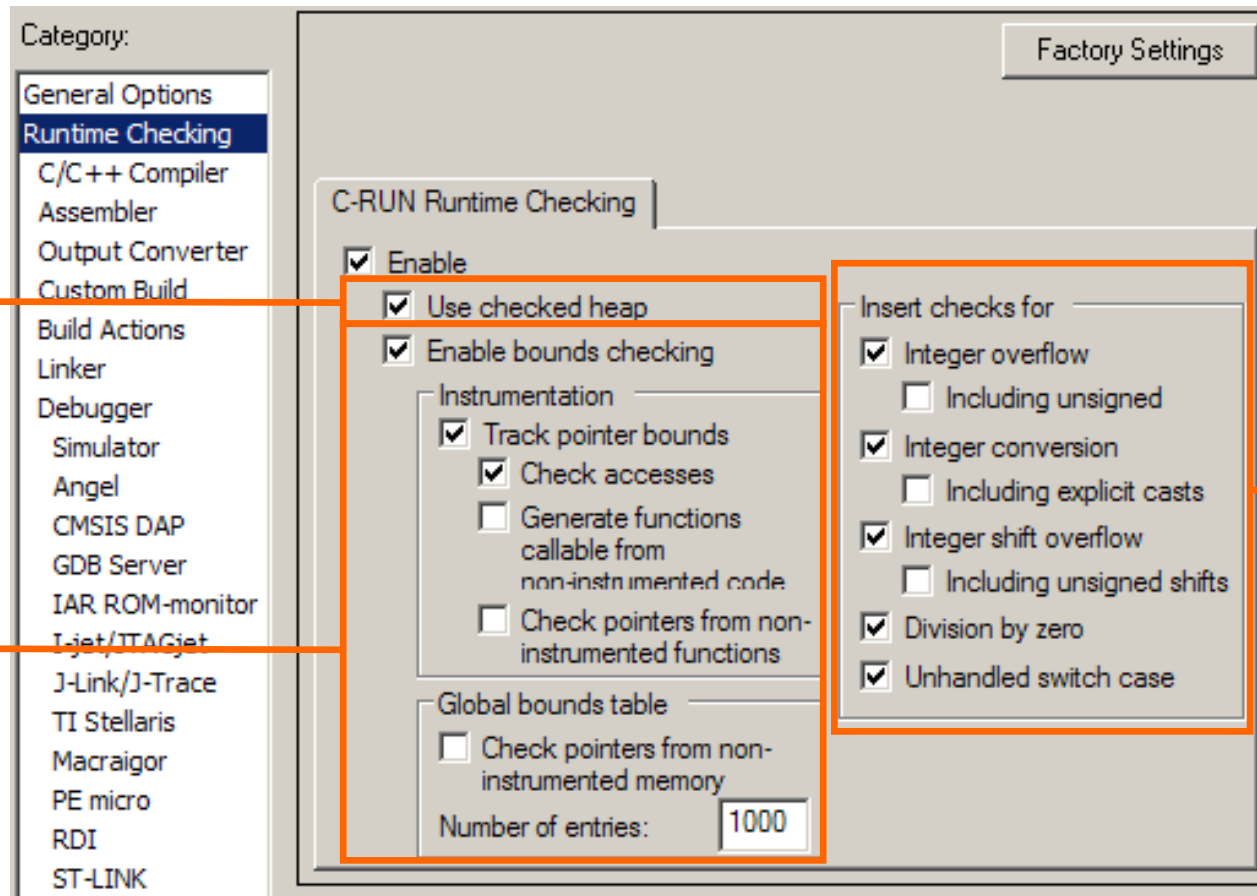
- C-RUN is a runtime analysis tool developed by IAR Systems
 - Launched in May, 2014
 - Both C and C++ source code are supported
- C-RUN is an add-on product of IAR Embedded Workbench
 - Fully integrated
 - No additional installation
 - No separate license
 - Cannot work with 3rd-party compiler & debugger tools
- Target support
 - IAR Embedded Workbench for ARM, from version 7.20
 - All ARM cores are supported

C-RUN options in IAR EWARM

Heap
checking

Bounds
checking

Arithmetic
checking



The screenshot shows the 'C-RUN Runtime Checking' dialog box in IAR EWARM. The 'Runtime Checking' category is selected in the left sidebar. The 'Factory Settings' button is in the top right. The 'C-RUN Runtime Checking' section is expanded, showing the following options:

- Enable
- Use checked heap
- Enable bounds checking
 - Instrumentation
 - Track pointer bounds
 - Check accesses
 - Generate functions callable from non-instrumented code
 - Check pointers from non-instrumented functions
 - Global bounds table
 - Check pointers from non-instrumented memory
 - Number of entries:

- Insert checks for
- Integer overflow
 - Including unsigned
- Integer conversion
 - Including explicit casts
- Integer shift overflow
 - Including unsigned shifts
- Division by zero
- Unhandled switch case

Detecting integer overflow



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

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

Insert checks for

- Integer overflow
 - Including unsigned
- Integer conversion
 - Including explicit casts
- Integer shift overflow
 - Including unsigned shifts
- Division by zero
- Unhandled switch case

Default action: Stop Filter: Messages: 2

Messages	Source File	PC
 Signed integer overflow	main.c 6:3-6	0x000000E8
 Unsigned integer overflow	main.c 7:3-6	0x00000114
Result is greater than the largest representable number: 4294967295 (0xffffffff) + 1 (0x1).		
Call Stack		
main	main.c 7:3-7	
[_call_main + 0x9]		

Detecting integer conversion



```
void main (void)
{
    int v1 = 0x8000;
    short v2;
    char v3;

    v2 = v1; /* 32-bit → 16-bit */
    v3 = v1; /* 32-bit → 8-bit */
}
```

Insert checks for

- Integer overflow
 - Including unsigned
- Integer conversion
 - Including explicit casts
- Integer shift overflow
 - Including unsigned shifts
- Division by zero
- Unhandled switch case

Default action: Stop Filter: Messages: 2

Messages	Source File	PC
 Integer conversion failure	main.c 7:8-9	0x000000D2
 Integer conversion failure	main.c 8:8-9	0x000000E4
Conversion changes the value from 32768 (0x000008000) to 0 (0x00).		
Call Stack		
main	main.c 8:3-10	
[_call_main + 0x9]		

Detecting shift overflow




```
void main (void)
{
    int i;
    short v1 = 1;
    int v2 = 1;

    for (i=0; i<32; i++)
    {
        v1 <<= 1; /* overflow: i>14 */
        v2 <<= 1; /* overflow: i>30 */
    }
}
```

Insert checks for

- Integer overflow
 - Including unsigned
- Integer conversion
 - Including explicit casts
- Integer shift overflow
 - Including unsigned shifts
- Division by zero
- Unhandled switch case

Default action: Stop Filter: Messages: 2

Messages	Source File	PC
 Shift overflow	main.c 9:15-22	0x0000010A
 Shift overflow	main.c 10:5-12	0x0000012A
Result is greater than the largest representable number: signed value 1073741824 (0x40000000) doubled 1 time(s).		
 Call Stack		
main	main.c 10:5-13	
[_call_main + 0x9]		

Detecting unhandled switch-case


```
void main (void)
{
    int i;

    for (i=0; i<2; i++)
    {
        switch (i) /* case 1 is not handled */
        {
            case 0:
                break;
        }
    }
}
```

Insert checks for

- Integer overflow
 - Including unsigned
- Integer conversion
 - Including explicit casts
- Integer shift overflow
 - Including unsigned shifts
- Division by zero
- Unhandled switch case

Default action: Stop Filter: Messages: 1

Messages	Source File	PC
 Unhandled case in switch	main.c 7:5-14	0x0000009C
Switch to undefined case label.		
Call Stack		
main	main.c 5:18-21	
[_call_main + 0x9]		

Detecting heap errors - 1

```
#include <stdlib.h>
```

```
void main (void)
```

```
{
```

```
    char *c1 = (char *)malloc(10);
```

```
    char *c2 = new char[10];
```

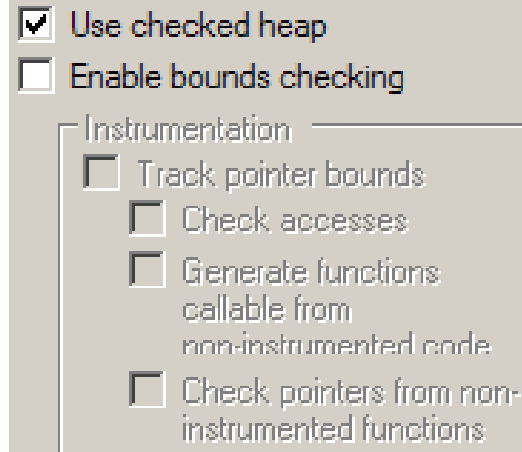
```
    free(c1+2); /* not the start of a block */
```

```
    free(c2); /* non-matched new and free */
```

```
    free(c1);
```

```
    free(c1); /* free a block more than once */
```

```
}
```



Use checked heap

Enable bounds checking

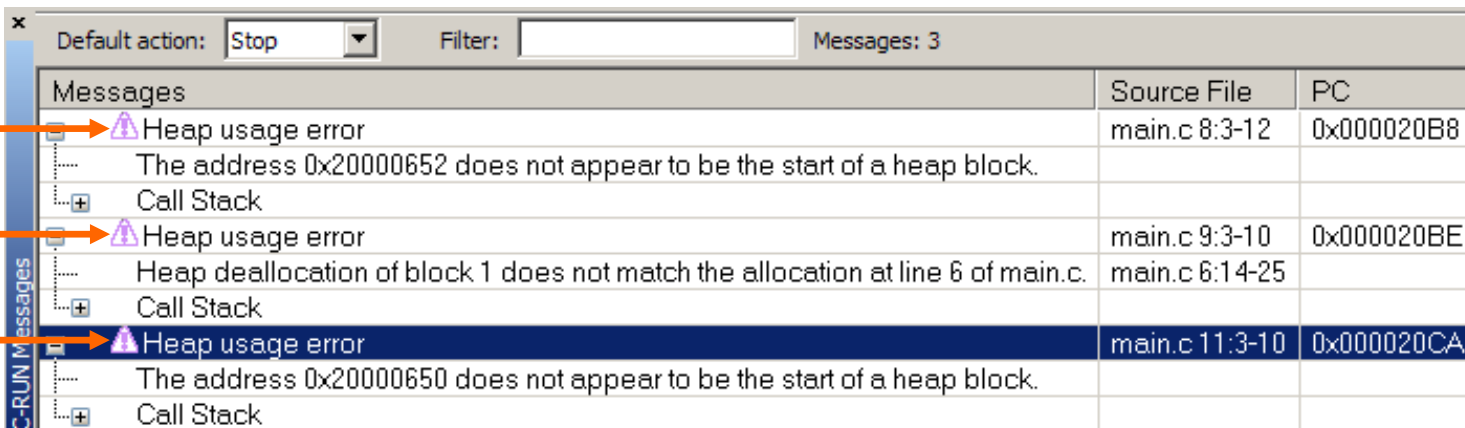
Instrumentation

Track pointer bounds

Check accesses

Generate functions callable from non-instrumented code

Check pointers from non-instrumented functions



Default action: Stop Filter: Messages: 3

Messages	Source File	PC
Heap usage error The address 0x20000652 does not appear to be the start of a heap block. Call Stack	main.c 8:3-12	0x000020B8
Heap usage error Heap deallocation of block 1 does not match the allocation at line 6 of main.c. Call Stack	main.c 9:3-10 main.c 6:14-25	0x000020BE
Heap usage error The address 0x20000650 does not appear to be the start of a heap block. Call Stack	main.c 11:3-10	0x000020CA

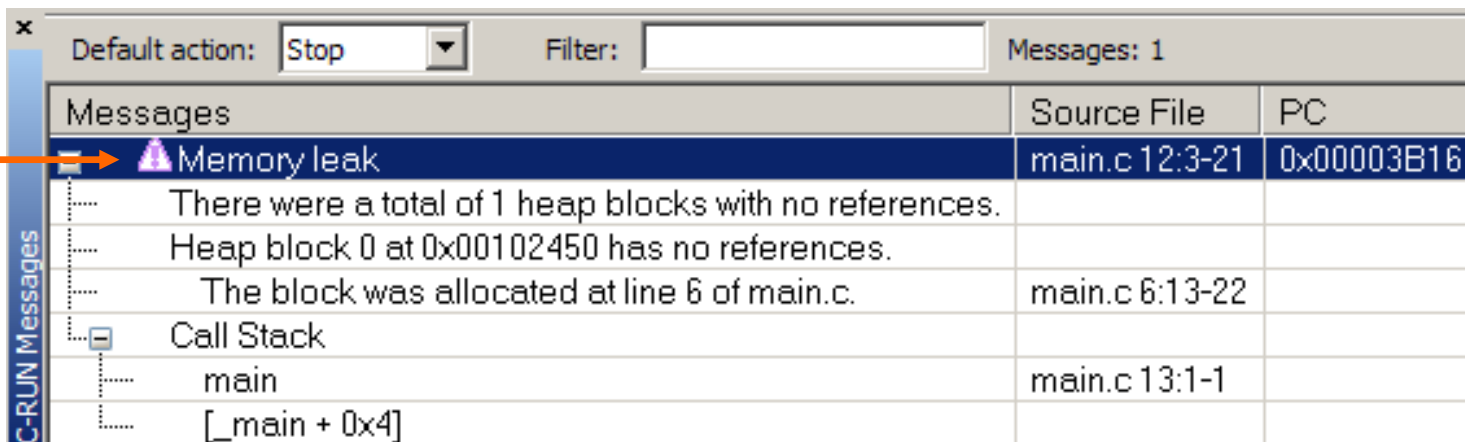
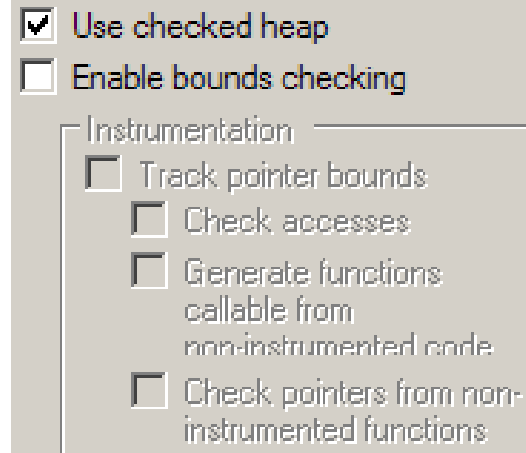
Detecting heap errors - 2

```
#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_check_leaks();
}
```



Default action: Stop Filter: Messages: 1

Messages	Source File	PC
 Memory leak	main.c 12:3-21	0x00003B16
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.		
	main.c 6:13-22	
Call Stack		
	main	main.c 13:1-1
	[_main + 0x4]	

Detecting out-of-bounds

```
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;
}
```

Use checked heap

Enable bounds checking

Instrumentation


Track pointer bounds

Check accesses

Generate functions callable from non-instrumented code

Check pointers from non-instrumented functions

Default action: Stop Filter: Messages: 1

Messages	Source File	PC
 Access out of bounds	main.c 6:13-16	0x000001E8
Access outside pointer bounds:		
Access 0x00101ff0 - 0x00101ff4		
Bounds 0x00101fe4 - 0x00101ff0, int a[3];	main.c 4:7-7	
Call Stack		

Take full control of your development

Implement your design in code

```

1  int main()
2  {
3      int i;
4      int j;
5      int k;
6      int l;
7      int m;
8      int n;
9      int o;
10     int p;
11     int q;
12     int r;
13     int s;
14     int t;
15     int u;
16     int v;
17     int w;
18     int x;
19     int y;
20     int z;
21     int a;
22     int b;
23     int c;
24     int d;
25     int e;
26     int f;
27     int g;
28     int h;
29     int i;
30     int j;
31     int k;
32     int l;
33     int m;
34     int n;
35     int o;
36     int p;
37     int q;
38     int r;
39     int s;
40     int t;
41     int u;
42     int v;
43     int w;
44     int x;
45     int y;
46     int z;
47     int a;
48     int b;
49     int c;
50     int d;
51     int e;
52     int f;
53     int g;
54     int h;
55     int i;
56     int j;
57     int k;
58     int l;
59     int m;
60     int n;
61     int o;
62     int p;
63     int q;
64     int r;
65     int s;
66     int t;
67     int u;
68     int v;
69     int w;
70     int x;
71     int y;
72     int z;
73     int a;
74     int b;
75     int c;
76     int d;
77     int e;
78     int f;
79     int g;
80     int h;
81     int i;
82     int j;
83     int k;
84     int l;
85     int m;
86     int n;
87     int o;
88     int p;
89     int q;
90     int r;
91     int s;
92     int t;
93     int u;
94     int v;
95     int w;
96     int x;
97     int y;
98     int z;
99     int a;
100    int b;
101    int c;
102    int d;
103    int e;
104    int f;
105    int g;
106    int h;
107    int i;
108    int j;
109    int k;
110    int l;
111    int m;
112    int n;
113    int o;
114    int p;
115    int q;
116    int r;
117    int s;
118    int t;
119    int u;
120    int v;
121    int w;
122    int x;
123    int y;
124    int z;
125    int a;
126    int b;
127    int c;
128    int d;
129    int e;
130    int f;
131    int g;
132    int h;
133    int i;
134    int j;
135    int k;
136    int l;
137    int m;
138    int n;
139    int o;
140    int p;
141    int q;
142    int r;
143    int s;
144    int t;
145    int u;
146    int v;
147    int w;
148    int x;
149    int y;
150    int z;
151    int a;
152    int b;
153    int c;
154    int d;
155    int e;
156    int f;
157    int g;
158    int h;
159    int i;
160    int j;
161    int k;
162    int l;
163    int m;
164    int n;
165    int o;
166    int p;
167    int q;
168    int r;
169    int s;
170    int t;
171    int u;
172    int v;
173    int w;
174    int x;
175    int y;
176    int z;
177    int a;
178    int b;
179    int c;
180    int d;
181    int e;
182    int f;
183    int g;
184    int h;
185    int i;
186    int j;
187    int k;
188    int l;
189    int m;
190    int n;
191    int o;
192    int p;
193    int q;
194    int r;
195    int s;
196    int t;
197    int u;
198    int v;
199    int w;
200    int x;
201    int y;
202    int z;
203    int a;
204    int b;
205    int c;
206    int d;
207    int e;
208    int f;
209    int g;
210    int h;
211    int i;
212    int j;
213    int k;
214    int l;
215    int m;
216    int n;
217    int o;
218    int p;
219    int q;
220    int r;
221    int s;
222    int t;
223    int u;
224    int v;
225    int w;
226    int x;
227    int y;
228    int z;
229    int a;
230    int b;
231    int c;
232    int d;
233    int e;
234    int f;
235    int g;
236    int h;
237    int i;
238    int j;
239    int k;
240    int l;
241    int m;
242    int n;
243    int o;
244    int p;
245    int q;
246    int r;
247    int s;
248    int t;
249    int u;
250    int v;
251    int w;
252    int x;
253    int y;
254    int z;
255    int a;
256    int b;
257    int c;
258    int d;
259    int e;
260    int f;
261    int g;
262    int h;
263    int i;
264    int j;
265    int k;
266    int l;
267    int m;
268    int n;
269    int o;
270    int p;
271    int q;
272    int r;
273    int s;
274    int t;
275    int u;
276    int v;
277    int w;
278    int x;
279    int y;
280    int z;
281    int a;
282    int b;
283    int c;
284    int d;
285    int e;
286    int f;
287    int g;
288    int h;
289    int i;
290    int j;
291    int k;
292    int l;
293    int m;
294    int n;
295    int o;
296    int p;
297    int q;
298    int r;
299    int s;
300    int t;
301    int u;
302    int v;
303    int w;
304    int x;
305    int y;
306    int z;
307    int a;
308    int b;
309    int c;
310    int d;
311    int e;
312    int f;
313    int g;
314    int h;
315    int i;
316    int j;
317    int k;
318    int l;
319    int m;
320    int n;
321    int o;
322    int p;
323    int q;
324    int r;
325    int s;
326    int t;
327    int u;
328    int v;
329    int w;
330    int x;
331    int y;
332    int z;
333    int a;
334    int b;
335    int c;
336    int d;
337    int e;
338    int f;
339    int g;
340    int h;
341    int i;
342    int j;
343    int k;
344    int l;
345    int m;
346    int n;
347    int o;
348    int p;
349    int q;
350    int r;
351    int s;
352    int t;
353    int u;
354    int v;
355    int w;
356    int x;
357    int y;
358    int z;
359    int a;
360    int b;
361    int c;
362    int d;
363    int e;
364    int f;
365    int g;
366    int h;
367    int i;
368    int j;
369    int k;
370    int l;
371    int m;
372    int n;
373    int o;
374    int p;
375    int q;
376    int r;
377    int s;
378    int t;
379    int u;
380    int v;
381    int w;
382    int x;
383    int y;
384    int z;
385    int a;
386    int b;
387    int c;
388    int d;
389    int e;
390    int f;
391    int g;
392    int h;
393    int i;
394    int j;
395    int k;
396    int l;
397    int m;
398    int n;
399    int o;
400    int p;
401    int q;
402    int r;
403    int s;
404    int t;
405    int u;
406    int v;
407    int w;
408    int x;
409    int y;
410    int z;
411    int a;
412    int b;
413    int c;
414    int d;
415    int e;
416    int f;
417    int g;
418    int h;
419    int i;
420    int j;
421    int k;
422    int l;
423    int m;
424    int n;
425    int o;
426    int p;
427    int q;
428    int r;
429    int s;
430    int t;
431    int u;
432    int v;
433    int w;
434    int x;
435    int y;
436    int z;
437    int a;
438    int b;
439    int c;
440    int d;
441    int e;
442    int f;
443    int g;
444    int h;
445    int i;
446    int j;
447    int k;
448    int l;
449    int m;
450    int n;
451    int o;
452    int p;
453    int q;
454    int r;
455    int s;
456    int t;
457    int u;
458    int v;
459    int w;
460    int x;
461    int y;
462    int z;
463    int a;
464    int b;
465    int c;
466    int d;
467    int e;
468    int f;
469    int g;
470    int h;
471    int i;
472    int j;
473    int k;
474    int l;
475    int m;
476    int n;
477    int o;
478    int p;
479    int q;
480    int r;
481    int s;
482    int t;
483    int u;
484    int v;
485    int w;
486    int x;
487    int y;
488    int z;
489    int a;
490    int b;
491    int c;
492    int d;
493    int e;
494    int f;
495    int g;
496    int h;
497    int i;
498    int j;
499    int k;
500    int l;
501    int m;
502    int n;
503    int o;
504    int p;
505    int q;
506    int r;
507    int s;
508    int t;
509    int u;
510    int v;
511    int w;
512    int x;
513    int y;
514    int z;
515    int a;
516    int b;
517    int c;
518    int d;
519    int e;
520    int f;
521    int g;
522    int h;
523    int i;
524    int j;
525    int k;
526    int l;
527    int m;
528    int n;
529    int o;
530    int p;
531    int q;
532    int r;
533    int s;
534    int t;
535    int u;
536    int v;
537    int w;
538    int x;
539    int y;
540    int z;
541    int a;
542    int b;
543    int c;
544    int d;
545    int e;
546    int f;
547    int g;
548    int h;
549    int i;
550    int j;
551    int k;
552    int l;
553    int m;
554    int n;
555    int o;
556    int p;
557    int q;
558    int r;
559    int s;
560    int t;
561    int u;
562    int v;
563    int w;
564    int x;
565    int y;
566    int z;
567    int a;
568    int b;
569    int c;
570    int d;
571    int e;
572    int f;
573    int g;
574    int h;
575    int i;
576    int j;
577    int k;
578    int l;
579    int m;
580    int n;
581    int o;
582    int p;
583    int q;
584    int r;
585    int s;
586    int t;
587    int u;
588    int v;
589    int w;
590    int x;
591    int y;
592    int z;
593    int a;
594    int b;
595    int c;
596    int d;
597    int e;
598    int f;
599    int g;
600    int h;
601    int i;
602    int j;
603    int k;
604    int l;
605    int m;
606    int n;
607    int o;
608    int p;
609    int q;
610    int r;
611    int s;
612    int t;
613    int u;
614    int v;
615    int w;
616    int x;
617    int y;
618    int z;
619    int a;
620    int b;
621    int c;
622    int d;
623    int e;
624    int f;
625    int g;
626    int h;
627    int i;
628    int j;
629    int k;
630    int l;
631    int m;
632    int n;
633    int o;
634    int p;
635    int q;
636    int r;
637    int s;
638    int t;
639    int u;
640    int v;
641    int w;
642    int x;
643    int y;
644    int z;
645    int a;
646    int b;
647    int c;
648    int d;
649    int e;
650    int f;
651    int g;
652    int h;
653    int i;
654    int j;
655    int k;
656    int l;
657    int m;
658    int n;
659    int o;
660    int p;
661    int q;
662    int r;
663    int s;
664    int t;
665    int u;
666    int v;
667    int w;
668    int x;
669    int y;
670    int z;
671    int a;
672    int b;
673    int c;
674    int d;
675    int e;
676    int f;
677    int g;
678    int h;
679    int i;
680    int j;
681    int k;
682    int l;
683    int m;
684    int n;
685    int o;
686    int p;
687    int q;
688    int r;
689    int s;
690    int t;
691    int u;
692    int v;
693    int w;
694    int x;
695    int y;
696    int z;
697    int a;
698    int b;
699    int c;
700    int d;
701    int e;
702    int f;
703    int g;
704    int h;
705    int i;
706    int j;
707    int k;
708    int l;
709    int m;
710    int n;
711    int o;
712    int p;
713    int q;
714    int r;
715    int s;
716    int t;
717    int u;
718    int v;
719    int w;
720    int x;
721    int y;
722    int z;
723    int a;
724    int b;
725    int c;
726    int d;
727    int e;
728    int f;
729    int g;
730    int h;
731    int i;
732    int j;
733    int k;
734    int l;
735    int m;
736    int n;
737    int o;
738    int p;
739    int q;
740    int r;
741    int s;
742    int t;
743    int u;
744    int v;
745    int w;
746    int x;
747    int y;
748    int z;
749    int a;
750    int b;
751    int c;
752    int d;
753    int e;
754    int f;
755    int g;
756    int h;
757    int i;
758    int j;
759    int k;
760    int l;
761    int m;
762    int n;
763    int o;
764    int p;
765    int q;
766    int r;
767    int s;
768    int t;
769    int u;
770    int v;
771    int w;
772    int x;
773    int y;
774    int z;
775    int a;
776    int b;
777    int c;
778    int d;
779    int e;
780    int f;
781    int g;
782    int h;
783    int i;
784    int j;
785    int k;
786    int l;
787    int m;
788    int n;
789    int o;
790    int p;
791    int q;
792    int r;
793    int s;
794    int t;
795    int u;
796    int v;
797    int w;
798    int x;
799    int y;
800    int z;
801    int a;
802    int b;
803    int c;
804    int d;
805    int e;
806    int f;
807    int g;
808    int h;
809    int i;
810    int j;
811    int k;
812    int l;
813    int m;
814    int n;
815    int o;
816    int p;
817    int q;
818    int r;
819    int s;
820    int t;
821    int u;
822    int v;
823    int w;
824    int x;
825    int y;
826    int z;
827    int a;
828    int b;
829    int c;
830    int d;
831    int e;
832    int f;
833    int g;
834    int h;
835    int i;
836    int j;
837    int k;
838    int l;
839    int m;
840    int n;
841    int o;
842    int p;
843    int q;
844    int r;
845    int s;
846    int t;
847    int u;
848    int v;
849    int w;
850    int x;
851    int y;
852    int z;
853    int a;
854    int b;
855    int c;
856    int d;
857    int e;
858    int f;
859    int g;
860    int h;
861    int i;
862    int j;
863    int k;
864    int l;
865    int m;
866    int n;
867    int o;
868    int p;
869    int q;
870    int r;
871    int s;
872    int t;
873    int u;
874    int v;
875    int w;
876    int x;
877    int y;
878    int z;
879    int a;
880    int b;
881    int c;
882    int d;
883    int e;
884    int f;
885    int g;
886    int h;
887    int i;
888    int j;
889    int k;
890    int l;
891    int m;
892    int n;
893    int o;
894    int p;
895    int q;
896    int r;
897    int s;
898    int t;
899    int u;
900    int v;
901    int w;
902    int x;
903    int y;
904    int z;
905    int a;
906    int b;
907    int c;
908    int d;
909    int e;
910    int f;
911    int g;
912    int h;
913    int i;
914    int j;
915    int k;
916    int l;
917    int m;
918    int n;
919    int o;
920    int p;
921    int q;
922    int r;
923    int s;
924    int t;
925    int u;
926    int v;
927    int w;
928    int x;
929    int y;
930    int z;
931    int a;
932    int b;
933    int c;
934    int d;
935    int e;
936    int f;
937    int g;
938    int h;
939    int i;
940    int j;
941    int k;
942    int l;
943    int m;
944    int n;
945    int o;
946    int p;
947    int q;
948    int r;
949    int s;
950    int t;
951    int u;
952    int v;
953    int w;
954    int x;
955    int y;
956    int z;
957    int a;
958    int b;
959    int c;
960    int d;
961    int e;
962    int f;
963    int g;
964    int h;
965    int i;
966    int j;
967    int k;
968    int l;
969    int m;
970    int n;
971    int o;
972    int p;
973    int q;
974    int r;
975    int s;
976    int t;
977    int u;
978    int v;
979    int w;
980    int x;
981    int y;
982    int z;
983    int a;
984    int b;
985    int c;
986    int d;
987    int e;
988    int f;
989    int g;
990    int h;
991    int i;
992    int j;
993    int k;
994    int l;
995    int m;
996    int n;
997    int o;
998    int p;
999    int q;
1000   int r;

```

Build and debug the application



Release the application

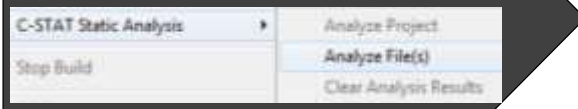
```

10001 11010111010 01001
00011 11001100011 01110
10010 00101011110 01011
10111 10110011001 10011
00111 01010101101 11001
11001010101110011
101010110011001

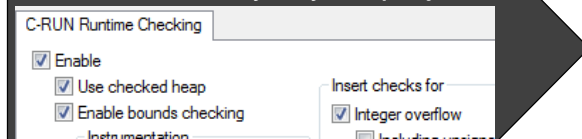
```



Let C-STAT analyze your code



Let C-RUN analyze your project



Review potential issues



Investigate runtime errors



Requirements

Design

Implementation

Verification

Maintenance

IAR Systems: Your strategic partner

- Different architecture, one solution
- Most efficient & high performance code
- Freescale MQX™ RTOS integration
- Freescale Processor Expert integration
- Advanced trace debugging
- Power debugging
- C-STAT static code analysis
- C-RUN runtime code analysis
- Stack usage analysis & tracking
- Functional safety certificate
- Global professional technical support

