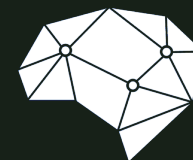




xcalibyte



Horizon
Robotics



SPACEMIT

seL4 SUMMIT

YUNING LIANG - OCT 2022

Agenda

- About Xcalibyte
- ISO 26262 ASIL-D and MISRA
- MISRA and Verification Conflict
- seL4 MISRA Compliance progress



About Xcalibyte - Hardcore deep tech startup

- **Specialised Compiler for RISC-V ONLY**
 - Original MIPS/SGI/HP and Intel Itanium compiler team
 - Open source project open64 code based
 - Compatible with Clang/Gcc/Llvm
- **Specialised for RISC-V Autonomous Driving**
 - seL4 micro-kernel for ISO 26262 ASIL - D
 - Static analyser for ISO 26262 ASIL - D



ISO 26262 ASIL-D and MISRA

- ISO-26262 ASIL-D certified requirements

Table 6 — Design principles for software unit design and implementation

Principle	ASIL			
	A	B	C	D
1a One entry and one exit point in subprograms and functions ^a	++	++	++	++
1b No dynamic objects or variables, or else online test during their creation ^a	+	++	++	++
1c Initialization of variables	++	++	++	++
1d No multiple use of variable names ^a	++	++	++	++
1e Avoid global variables or else justify their usage ^a	+	+	++	++
1f Restricted use of pointers ^a	+	++	++	++
1g No implicit type conversions ^a	+	++	++	++
1h No hidden data flow or control flow	+	++	++	++
1i No unconditional jumps ^a	++	++	++	++
1j No recursions	+	+	++	++

^a Principles 1a, 1b, 1d, 1e, 1f, 1g and 1i may not be applicable for graphical modelling notations used in model-based development.

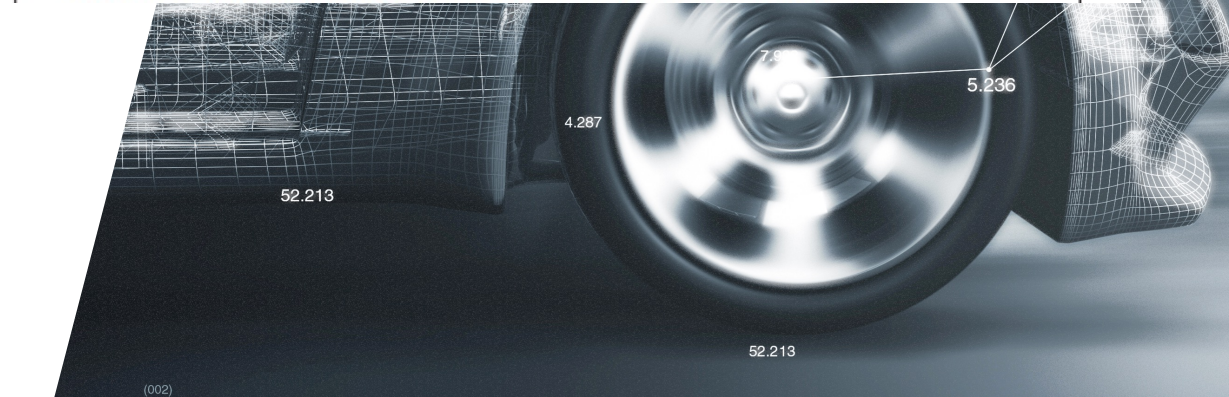
NOTE For the C language, MISRA C (see Reference [3]) covers many of the principles listed in Table 6.

ISO 26262-6:2018(E)






Table 1 — Topics to be covered by modelling and coding guidelines


Topics	ASIL			
	A	B	C	D
1a Enforcement of low complexity ^a	++	++	++	++
1b Use of language subsets ^b	++	++	++	++
1c Enforcement of strong typing ^c	++	++	++	++
1d Use of defensive implementation techniques ^d	+	+	++	++
1e Use of well-trusted design principles ^e	+	+	++	++
1f Use of unambiguous graphical representation	+	++	++	++
1g Use of style guides	+	++	++	++
1h Use of naming conventions	++	++	++	++
1i Concurrency aspects ^f	+	+	+	+


^a An appropriate compromise of this topic with other requirements of this document may be required.



ISO 26262 ASIL-D and MISRA

ISO 26262	MISRA C rule	Coverage
One entry and one exit point in subprograms and functions	15.5 <i>A function should have a single point of exit at the end</i>	
No dynamic objects or variables, or else online test during their creation	Only directive	
Initialization of variables	9.1 The value of an object with automatic storage duration shall not be read before it has been set	
	9.3 Arrays shall not be partially initialized	
	9.4 An element of an object shall not be initialized more than once	
No multiple use of variable names	5.4 Macro identifiers shall be distinct	
	5.5 Identifiers shall be distinct from macro names	
	5.7 A tag name shall be a unique identifier	
Avoid global variables or else justify their usage		

ISO 26262	MISRA C rule	Coverage
Restricted use of pointers	18.1 A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand	
	18.2 Subtraction between pointers shall not be applied to pointers that address elements of the same array	
	18.3 The relational operators >, >=, < and <= shall not be applied to objects of pointer type except where they point into the same object	
	8.14 The restrict type qualifier shall not be used	
	11.2 Conversions shall not be performed between a pointer to an incomplete type and any other type	
	11.3 A cast shall not be performed between a pointer to object type and a pointer to a different object type	
	11.4 A conversion should not be performed between a pointer to object and an integer type	
11.5 A conversion should not be performed from pointer to void into pointer to object		

ISO 26262	MISRA C rule	Coverage
No recursions	17.2 Functions shall not call themselves either directly or indirectly	

ISO 26262 ASIL-D and MISRA

6.2 Guideline categories

Every MISRA C guideline is given a single category of “mandatory”, “required” or “advisory”, whose meanings are described below. Beyond this basic classification the document does not give, nor intend to imply, any grading of importance of each of the guidelines. All required guidelines, whether rules or directives, should be considered to be of equal importance, as should all mandatory and advisory ones.

6.2.1 Mandatory guidelines

C code which is claimed to conform to this document shall comply with every mandatory guideline — deviation from mandatory guidelines is not permitted.

Note: if a checking tool produces a diagnostic message, this does not necessarily mean that a guideline has been violated for the reasons given in Section 6.5.

6.2.2 Required guidelines

C code which is claimed to conform to this document shall comply with every required guideline, with a formal deviation required, as described in Section 5.4, where this is not the case.

An organization or project may choose to treat any required guideline as if it were mandatory.

6.2.3 Advisory guidelines

These are recommendations. However, the status of “advisory” does not mean that these items can be ignored, but rather that they should be followed as far as is reasonably practical. Formal deviation is not necessary for advisory guidelines but, if the formal deviation process is not followed, alternative arrangements should be made for documenting non-compliances.

An organization or project may choose to treat any advisory guideline as if it were mandatory or required.

Directives

7.1	The implementation	21
7.2	Compilation and build	23
7.3	Requirements traceability	23
7.4	Code design	24

Rules

8.1	A standard C environment	37
8.2	Unused code	39
8.3	Comments	45
8.4	Character sets and lexical conventions	46
8.5	Identifiers	48
8.6	Types	58
8.7	Literals and constants	59
8.8	Declarations and definitions	63
8.9	Initialization	75
8.10	The essential type model	81
8.11	Pointer type conversions	93
8.12	Expressions	103
8.13	Side effects	108
8.14	Control statement expressions	115
8.15	Control flow	122
8.16	Switch statements	130
8.17	Functions	136
8.18	Pointers and arrays	143
8.19	Overlapping storage	153
8.20	Preprocessing directives	155
8.21	Standard libraries	165
8.22	Resources	172

Go to page 144

MISRA: seL4 (L4V) Overview

- 2276(1699) Medium&High Risk on Proof Impact [Required+Advisory]
- 1084(952) Medium&High Risk on Proof Impact [Required]

Rule Strength	MISRA Rule/Directive	Error Num (l4v kernel)	l4v Modify Risk (Possibility)	Error Num (Full kernel)	Explanation & Example
Required	MISRA C-2012 Rule 5.7	326	*Medium	360	The identifier is reused [2 cases]
	MISRA C-2012 Rule 16.3	235	*High	300	The unconditional break is missing terminate every switch-clause [4 cases]
	MISRA C-2012 Rule 14.3	137	*High	147	Invalid logical judgment [4 cases]
	MISRA C-2012 Rule 21.2	78	Medium	86	Reserved identifier or macro name have been declared [1 case]
	MISRA C-2012 Rule 16.1	63	High	65	The switch statement is incorrectly formatted [1 case]
	MISRA C-2012 Rule 16.6	46	High	46	The switch statement has no more than two switch-clauses [1 case]
	MISRA C-2012 Rule 21.1	26	Medium	28	#define and #undef has been used on a reserved identifier or reserved n
	MISRA C-2012 Rule 12.2	20	High	22	The right operand of the shift operator is greater than the width of the ba
	MISRA C-2012 Rule 2.2	10	*High	15	Unused code [1 case]
	MISRA C-2012 Rule 17.2	3	High	4	Functions Recursion call themselves [1 case]
	MISRA C-2012 Rule 5.3	3	Medium	3	Identifiers in outer scope are hidden by inner Scope identifiers [1 case]
	MISRA C-2012 Rule 5.5	3	Medium	2	The identifier is the same name as the macro [1 case]
	MISRA C-2012 Rule 5.8	2	Medium	6	The identifier is used by object with external linkage [1 case]
Advisory	MISRA C-2012 Rule 15.5	591	*High	699	Return is not the last statement in a function [1 case]
	MISRA C-2012 Rule 11.4	257	High	277	There are conversions between integers and Pointers [1 case]
	MISRA C-2012 Rule 17.8	100	*High	78	Function parameter have been modified [1 case]
	MISRA C-2012 Rule 2.7	64	High	71	There are unused arguments in the function [1 case]
	MISRA C-2012 Rule 18.4	61	High	67	Pointers perform +, -, +=, -= operations [1 case]

MISRA and Proofs Conflict Examples

MISRA C-2012 Rule 15.5 (500+)

- Statement: "Return is not the last statement"
- Proof Impact: "Control Flow changes"

```
1909
1910 /* ReadRegisters is a special case: replyFromKernel & setMRs are
1911 * unfolded here, in order to avoid passing the large reply message up
1912 * to the top level in a global (and double-copying). We prevent the
1913 * top-level replyFromKernel_success_empty() from running by setting the
1914 * thread state. Retype does this too.
1915 */
1916 exception_t invokeTCB_ReadRegisters(tcb_t *tcb_src, bool_t suspendSource,
1917                                     word_t n, word_t arch, bool_t call)
1918 {
1919     word_t i, j;
1920     exception_t e;
1921     tcb_t *thread;
1922
1923     thread = NODE_STATE(ksCurThread);
1924
1925     if (suspendSource) {
1926         suspend(tcb_src);
1927     }
1928
1929     e = Arch_performTransfer(arch, tcb_src, NODE_STATE(ksCurThread));
1930     if (e != EXCEPTION_NONE) {
```

(1) Event misra_c_2012_rule_15_5_violation: This return statement is not the final statement in the compound statement that forms the body of the function.

```
1931         return e;
1932     }
1933
```


MISRA and Proofs Conflict Examples

MISRA C-2012 Rule 16.1/16.3/16.6 (350+)

- Statement: "An unconditional break statement shall terminate every switch-clause."
- Proof Impact: "Control Flow changes"

【例】 vspace.c line 2645

(1) Event misra_c_2012_rule_16_3_violation: This switch clause does not end with an unconditional break statement.

```
2645 case cap_asid_pool_cap: {
2646     cap_t pdCap;
2647     cte_t *pdCapSlot;
2648     asid_pool_t *pool;
2649     word_t i;
2650     asid_t asid;
2651
2652     if (unlikely(invLabel != ARMASIDPoolAssign)) {
2653         userError("ASIDPool: Illegal operation.");
2654         current_syscall_error.type = sel4_IllegalOperation;
2655
2656         return EXCEPTION_SYSCALL_ERROR;
2657     }
2658
2659     if (unlikely(excaps.excaprefs[0] == NULL)) {
2660         userError("ASIDPoolAssign: Truncated message.");
2661         current_syscall_error.type = sel4_TruncatedMessage;
2662
2663         return EXCEPTION_SYSCALL_ERROR;
2664     }
2665
2666     pdCapSlot = excaps.excaprefs[0];
2667     pdCap = pdCapSlot->cap;
2668
2669     if (unlikely(
2670         cap_get_capType(pdCap) != cap_page_directory_cap ||
2671         cap_page_directory_cap_get_capPDIsMapped(pdCap))) {
2672         userError("ASIDPoolAssign: Invalid page directory cap.");
2673         current_syscall_error.type = sel4_InvalidCapability;
2674         current_syscall_error.invalidCapNumber = 1;
2675
2676         return EXCEPTION_SYSCALL_ERROR;
2677     }
```

【例】 objecttype.c line 23

```
20 bool_t Arch_isFrameType(word_t type)
21 {
22     switch (type) {
23         case sel4_ARM_SmallPageObject:
24             return true;
25         case sel4_ARM_LargePageObject:
26             return true;
27         case sel4_ARM_SectionObject:
28             return true;
29         case sel4_ARM_SuperSectionObject:
30             return true;
31         default:
32             return false;
33     }
34 }
```

(1) Event misra_c_2012_rule_16_3_violation: This switch clause does not end with an unconditional break statement.

【例】 syscall.c line 481

```
472 static void handleReply(void)
473 {
474     cte_t *callerSlot;
475     cap_t callerCap;
476
477     callerSlot = TCB_PTR_CTE_PTR(NODE_STATE(ksCurThread), tcbCaller);
478     callerCap = callerSlot->cap;
479
480     switch (cap_get_capType(callerCap)) {
```

(1) Event misra_c_2012_rule_16_3_violation: This switch clause does not end with an unconditional break statement.

```
481     case cap_reply_cap: {
482         tcb_t *caller;
483
484         if (cap_reply_cap_get_capReplyMaster(callerCap)) {
485             break;
486         }
487         caller = TCB_PTR(cap_reply_cap_get_capTCBPtr(callerCap));
488         /* Haskell error:
489          * "handleReply: caller must not be the current thread" */
490         assert(caller != NODE_STATE(ksCurThread));
491         doReplyTransfer(NODE_STATE(ksCurThread), caller, callerSlot,
492             cap_reply_cap_get_capReplyCanGrant(callerCap));
493         return;
494     }
495
496     case cap_null_cap:
497         userError("Attempted reply operation when no reply cap present.");
498         return;
499
500     default:
501         break;
502 }
503
504 fail("handleReply: invalid caller cap");
505 }
```

Way of Fixing Conflict Example

MISRA-C Rule 5.7 A tag name shall be a unique identifier

(1) Event `misra_c_2012_rule_5_7_violation`: Identifier "tcb" is already used to represent a type.

Also see events: [\[type declaration\]](#)

```
252 void completeSignal(notification_t *ntfnPtr, tcb_t *tcb)
253 {
254     word_t badge;
255
256     if (likely(tcb && notification_ptr_get_state(ntfnPtr) == NtfnState_Active)) {
257         badge = notification_ptr_get_ntfnMsgIdentifier(ntfnPtr);
258         setRegister(tcb, badgeRegister, badge);
259         notification_ptr_set_state(ntfnPtr, NtfnState_Idle);
260 #ifdef CONFIG_KERNEL_MCS
261         maybeDonateSchedContext(tcb, ntfnPtr);
262 #endif
263     } else {
```

(2) Event `type_declaration`: Declaring a type with identifier "tcb".

Also see events: [\[misra c 2012 rule 5 7 violation\]](#)

```
246 struct tcb {
247     /* arch specific tcb state (including context)*/
248     arch_tcb_t tcbArch;
249
250     /* Thread state, 3 words */
251     thread_state_t tcbState;
252
253     /* Notification that this TCB is bound to. If this is set, when this TCB waits on
254      * any sync endpoint, it may receive a signal from a Notification object.
255      * 1 word*/
256     notification_t *tcbBoundNotification;
257 }
```

- Change the structure name with a '_s' suffix. In that, we also need to change
- Change the problematic variable name 'tcb' to some other name. Like 'tcbPtr' for the 'completeSignal' mentioned above.



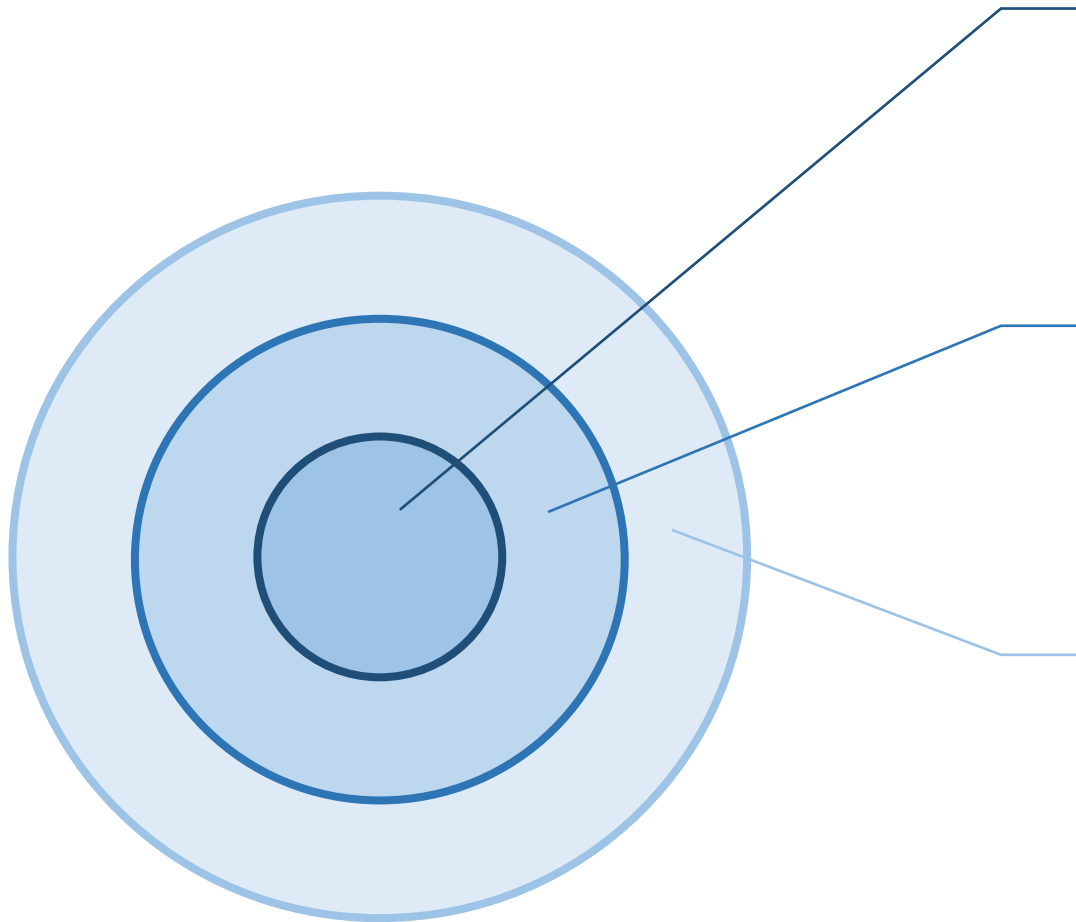
35
36
37
38
39
40
41

MISRA: seL4 (L4V) Compliance Progress

Date	Status: NEW			Total
	aarch32 stream	aarch64 stream	riscv stream	
Origin	7644	10217	6681	24542
19-Apr				4200
20-Apr	1483	2717	1942	3582
21-Apr	931	1839	1387	2264
22-Apr	752	992	735	1280
26-Apr	173	194	168	264
27-Apr	58	78	54	97
28-Apr	16	31	25	43
30-Apr	5	9	3	9



Xcalibyte help you deploying seL4



Faster, Even Safer, and Better seL4 By Xcalibyte

- Xcalscan
- XcalCompile

Faster RISC-V version seL4 By XcalCompile

- Optimise Code for RISC-V SoC
- GCC/Clang Compatible

ISO 26262 ASIL – D/MISRA C Compliance by XcalScan

- Cause Analysis
- Code/Proof fixings

A person in a dark hoodie and white sneakers sits on the edge of a skyscraper, looking out over a city skyline at sunset. The sky is filled with soft, colorful clouds in shades of orange, pink, and teal. The city below is a dense collection of buildings, with the CN Tower visible in the distance. The overall mood is contemplative and serene.

xcali**byte**

THANK YOU