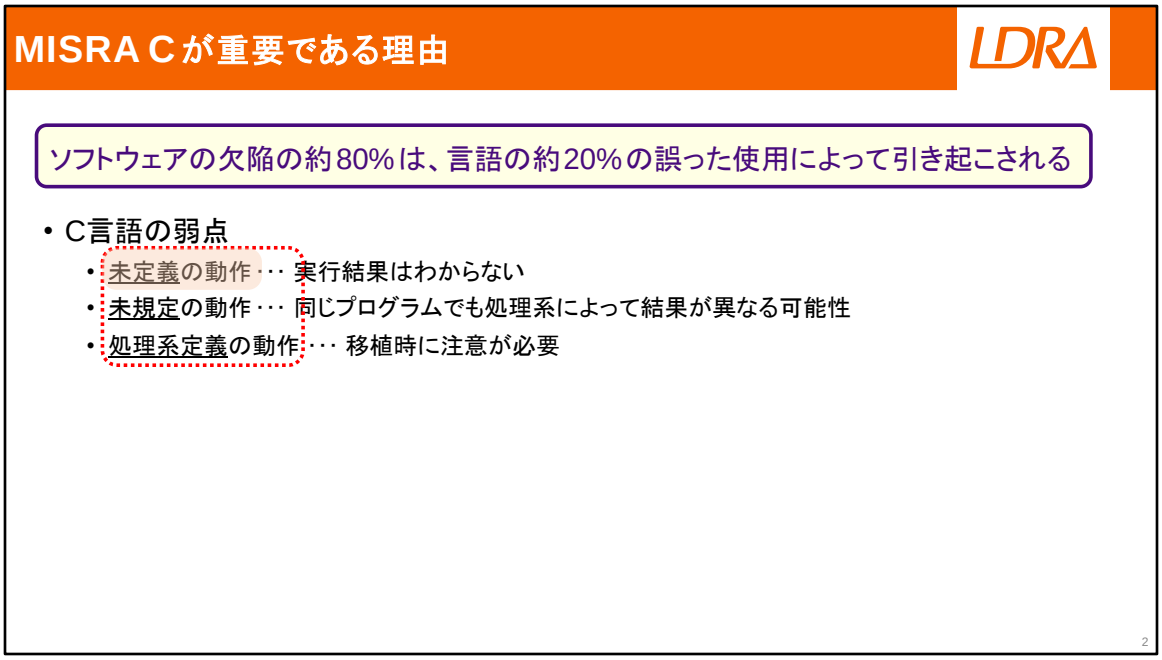
MISRA C Evolution

MISRA C発展の概要

Presented by
Keisuke Toyama
CTO Fujisetsubi

2023.6.12



まず、ソフトウェア開発にとってMISRA Cのようなコーディング規約、すなわち言語のサブセッティングを行って使用を制限することが重要である理由です。ソフトウェアは、正しく動作するものをきちんと作成するべきものですが、正しくないこと、すなわち欠陥の80%は、言語の20% の誤った使用によるものという経験則があります。
C言語は非常にポピュラーで、ソフトウェア開発において多く利用されていますが、誤った使用によって欠陥につながる可能性が高いという「弱点」があります。
中でも大きなものは、未定義、未規定、処理系依存の動作で、言語の仕様として動作が定められていないものです。たとえば、未定義の動作は、文字通り動作が定義されていないもので、実行結果はわかりません。

未定義の動作の危険性

- 未定義の動作について、コンパイラは「どのように」実装してもよい

  たとえば、以下の関数は値を返さない

  コンパイラは警告を出すがコードは生成されて実行される

  この関数を呼び出した側で戻り値を参照した場合、どうなるかはわからない

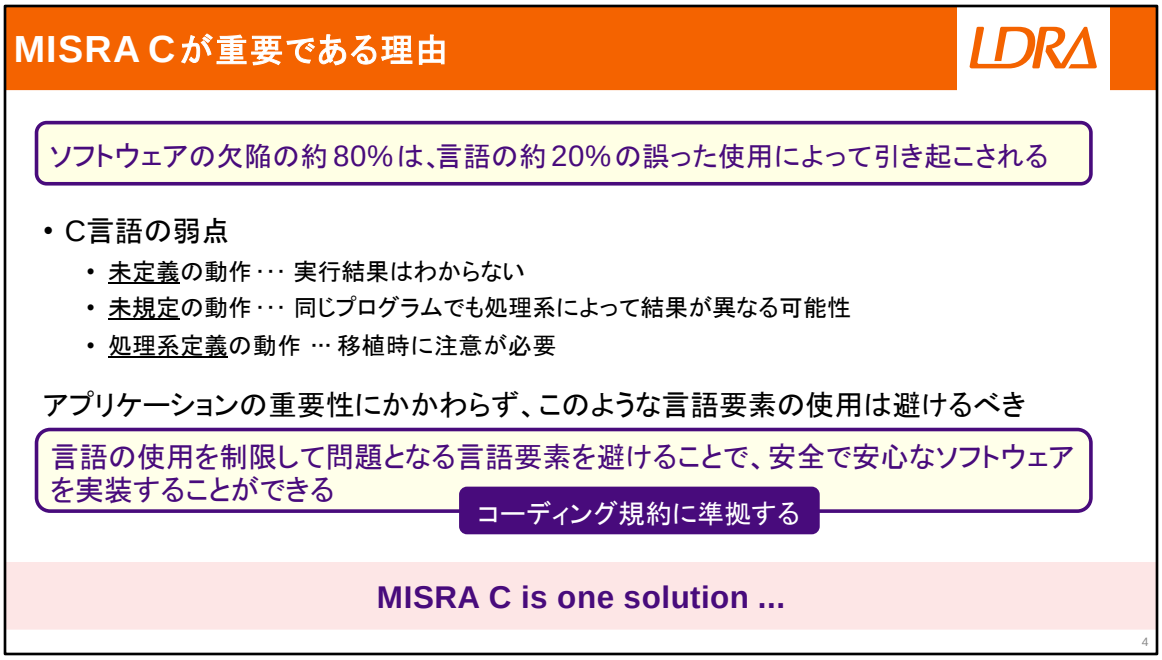
```
 7  int32_t global = 10;
 8
 9  int32_t undef(void)
10  {
16
17  int32_t noret(void)
18  {
19      global += undef();
20      // no return exp.
21  }
22
23  int main(void)
24  {
25      printf("%d %d\n", global, noret());
26  }
27
```

```
/c/Users/toyama-k/test1
$ gcc -Wall unspec.c
unspec.c: In function 'noret':
unspec.c:21:1: warning: control reaches end of non-void function [-Wreturn-type]
   21 | }
      | ^
```

3

「未定義の動作」は、その動作が言語規格に定義されていなく、コンパイラはどのように実装しても許されます。

この例の関数ではint 型 の戻り値があるとして宣言されていますが、return が書かれてないので値を返さず、オプションによってはコンパイラは警告を出しますが、コードは生成されます。この関数を呼び出した側で戻り値を参照した場合、どうなるかはわかりません。



このように、アプリケーションのコードがC言語規格の要件を満たしていても、どのように動作するかわからず、セキュリティ侵害や人命損失につながる可能性があります。どのようなアプリケーションであれ、このようなコードは避けるべきものです。

コーディング規約に準拠する、すなわち、言語の使用を制限することで、問題となる言語要素を避け、安全で安心なソフトウエアを実装できるようになると期待できます。

コンパイラの中には、コーディングでのリスクの高い実装を特定できるものもありますが、コードに問題が入り込む前に防ぐ方が、効率的で費用効果も高くなります。MISRA Cは、セーフティクリティカルなサブセットに制限してC言語を使用することで、潜在的に危険なコードを回避したり排除できます。

ISO 26262 における MISRA C

- ISO 26262-6:2018, Table 1
  - モデリングやコーディングガイドラインでカバーされるトピック
    - 例 1 : MISRA C は C 言語に対するコーディングガイドラインで、自動生成コードのガイダンスも含む
- ISO 26262-6:2018, Table 6
  - ソフトウェアユニットの設計と実装についての設計原則
    - 注 : C 言語に対しては、MISRA C が Table 6 の原則の多くをカバーする

トピック		ASIL			
		A	B	C	D
1a	低複雑性の実施	++	++	++	++
1b	言語サブセットの使用	++	++	++	++
1c	強い型付けの実施	++	++	++	++
1d	防御的実装技法の使用	+	+	++	++
1e	高信頼な設計原則の使用	+	+	++	++
1f	明確な図形表現の使用	+	++	++	++
1g	スタイルガイドの使用	+	++	++	++
1h	命名規約の使用	++	++	++	++
1i	並列性の見地	+	+	+	+

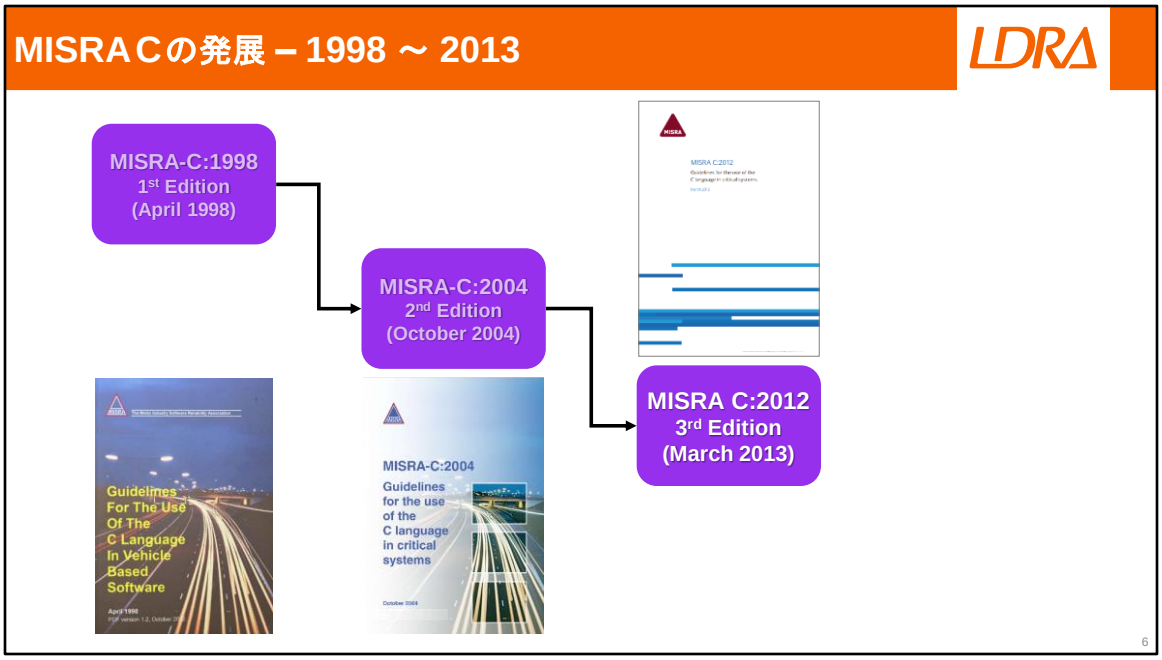
原則		ASIL			
		A	B	C	D
1a	手続き、関数の 1 入口・1 出口	++	++	++	++
1b	動的なオブジェクトや変数なし、でなければそれら生成時のオンラインテスト	+	++	++	++
1c	変数の初期化	++	++	++	++
1d	変数名を多重使用しない	+	+	++	++
1e	グローバル変数は使用しない、もしくは使用の正当性を示す	+	+	++	++
1f	ポインタの使用の制限	o	+	+	++
1g	暗黙の型変換を行わない	+	+	++	++
1h	隠れたデータフローや制御フローなし	+	+	++	++
1i	無条件分岐なし	++	++	++	++
1j	再帰なし	+	+	++	++

5

これらの言語要素を避けるにはMISRA Cのようなコーディング規約を採用すべきであり、これは機能安全規格でも推奨されています。

たとえば、ISO 26262のパート6では、表 1 で「言語サブセットの使用」などコーディングガイドラインに言及があり、MISRA Cが、そのようなガイドラインの例としてあげられています。

また、同表の 6 で、ソフトウェアユニットの設計と実装についての設計原則があげられ、その原則について、ご覧のように、MISRA Cが、多くをカバーするとされています。



本日の話題である最新版2023に至るまで、MISRA Cは何度かの改訂を経ています。1998年発行の第1版から、2004年の第2版、2013年の第3版と発展してきました。

第1版は1998年4月に発表されました。タイトルは「Guidelines for the use of the C language in vehicle based software」で、MISRA-C:1998 と呼ばれています。
2004年10月に第2版が公開されました。タイトルは「Guidelines for the use of the C language in critical systems」です。MISRA-C:2004と呼ばれています。
MISRA Cの第3版は2013年3月に発行されました。タイトルは変わらず、MISRA-C:2012と呼ばれています。

日本語によるMISRA Cドキュメント

Since its launch in 1998, the uptake and usage of MISRA C has far exceeded the authors' original expectations.
・・・
In Japan, a Japanese translation of MISRA C has been published by JSAE, and the MISRA C Study Group have produced a book (in Japanese) giving detailed explanations of the rules and additional code examples.

MISRA Cホームページより

- MISRA-C:1998
  - 解説書：日本規格協会MISRA-C研究会編
    『組込み開発者におくるMISRA-C組込みプログラミングの高信頼化ガイド』
- MISRA-C:2004
  - 翻訳書：公益社団法人自動車技術会（JSAE）
    『自動車用C言語利用のガイドライン（第2版）』
  - 解説書：日本規格協会MISRA-C研究会編
    『組込み開発者におくるMISRA-C 2004 C言語利用の高信頼化ガイド』
- MISRA C:2012
  - 解説書：日本規格協会MISRA-C研究会著
    『MISRA C 2012組み込みプログラミングの高信頼性ガイド』

7

これらの文書は日本語でも発行されており、MISRA Cのホームページで、このように自動車技術会様とMISRA-C研究会様の活動への言及があります。

MISRA-C:1998では、解説書として日本規格協会からMISRA-C研究会による『組込み開発者におくるMISRA-C －組込みプログラミングの高信頼化ガイド』が出版されています。
MISRA-C:2004では、翻訳が公益社団法人自動車技術会(JASAE)から『自動車用C言語利用のガイドライン（第2版）』として出版、解説書が、MISRA-C研究会による『組込み開発者におくるMISRA-C:2004 －C言語利用の高信頼化ガイド』として出版されています。
MISRA-C:2012は、昨年(2022年7月)に、解説書として、MISRA-C研究会による『MISRA C：2012 －組み込みプログラミングの高信頼性ガイド』が出版されました。

MISRA C発展の概要

- MISRA-C:1998
  - Guidelines for the use of the C language in vehicle based software
    自動車用ソフトウェアでC言語を利用するための手引き
  - C90言語規格が対象、127のルール
  - 現在でもレガシーシステムのメンテナンスで使用
- MISRA-C:2004
  - Guidelines for the use of the C language in critical systems
    クリティカルシステムでC言語を利用するための手引き
  - 引き続きC90を対象、141のルール
  - 自動車以外のソフトウェアへも適用できることを認識して文書のタイトルを変更し、
    組込みシステムのさまざまな側面に対応するセクションに新しいルールを追加
- MISRA C:2012

8

1998版は、「自動車用ソフトウェアで・・・」というタイトルになっています。
C90言語規格を対象とし、127のルールがあり、そのうちRequired(必要)ルールは93、Advisory(推奨)ルールは34となっています。

2004版では、1998版の運用や研究の結果が反映され、タイトルを「クリティカルシステムで・・・」とし、対象を自動車用ソフトウェアから全ての組込みソフトウェアへと
拡大したことが特徴です。組込みシステムのさまざまな側面に対応するセクションに新しいルールが追加されました。
これは、引き続いてC90の規格を扱うもので、ルールが141あり、そのうち必要ルールは121、推奨ルールは20です。

そして、現行の2012版になります。

MISRA C:2012 (March 2013)

- 対象範囲をC99までに拡大
- Amplificationや Rationaleの導入、「決定可能」/「決定不能」のルール分類、「ディレクティブ」の追加、など
  - **ルール**: 客観的で曖昧さのないソースコードの要件 ー ツールで準拠を確認できるもの
    - **決定可能**(Decidable): 静的解析によって確定的に検証できるもの
    - **決定不能**(Undecidable): そうでないもの
  - **ディレクティブ**: 開発プロセスや文書に関連するもの、解釈の余地があるもの

	ルール	Mandatory	Required	Advisory	ディレクティブ
MISRA-C:1998	127	-	93	34	
MISRA-C:2004	141	-	121	20	
MISRA C:2012	143	10	101	32	16

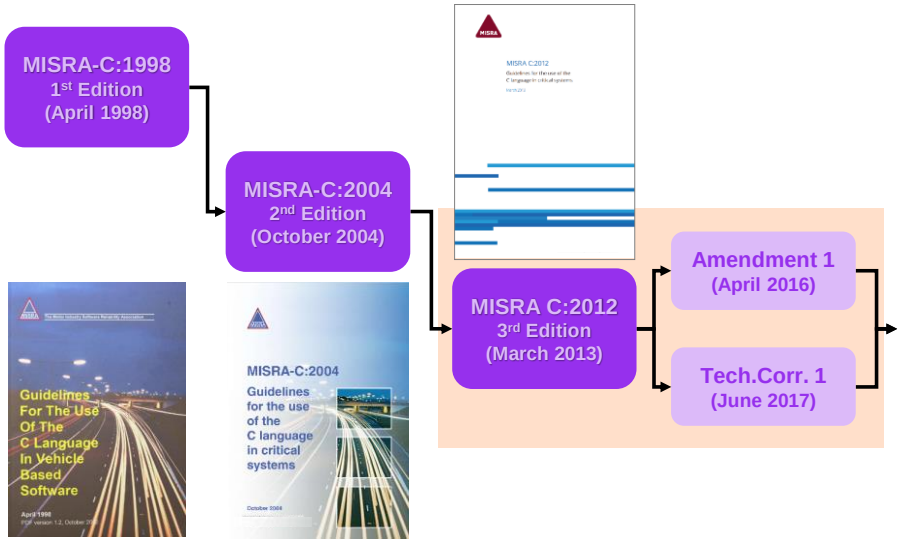
9

第3版となるMISRA C:2012は、C99規格への対応のため改訂です。また、Amplification や Rationale の導入、「決定可能」/「決定不能」のルール分類、「ディレクティブ(指針)」の追加などによって内容の整理や充実化、定義の明確化もなされています。

ルールが解析ツールでチェックできるよう客観的で曖昧さのないソースコードの要件であるのに対して、ディレクティブは、プロセスや文書などによって満たせる可能性のあるガイドラインです。さらに、ルールは、解析によって確定的に検証できる「決定可能」と、検証による保証が不可能な「決定不能」に分類されます。

MISRA C:2013からMandatory(必須)のルール分類が加えられ、必須ルールは違反が認められないものとされました。ディレクティブが16、ルールが143で、ルールのうち必須は10、必要は111、推奨は39です。
これら3つの版でのルール分類の表はこの通りです。

MISRA Cの発展 – ～ 2019



10

さらにMISRA C:2012は改正・訂正、さらに補遺を受け、2019年2月にリビジョン1が発行されました。
次で紹介します。

MISRA C:2012 Revision 1 (February 2019)

LDRA

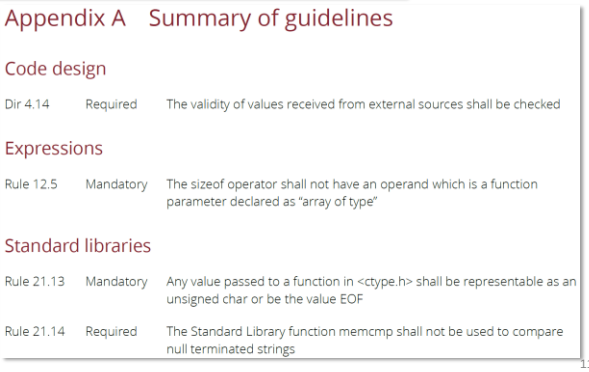
The MISRA C Guidelines define a subset of the C language in which the opportunity to make mistakes is either removed or reduced.
Many standard for the development of *safety-related software* require, or recommend, the use of a language subset, and this can also be used to develop any application with *high integrity or high reliability requirements*.

MISRA C:2012 Amendment 1 Forward より

Unfortunately, many people focus on the *safety-related software* reference, and a perception exists that MISRA C is only *safety-related* and not *security-related*.

以下の追加を実施

- MISRA C:2012 Amendment 1 (2016)
  組込みシステムのサイバーセキュリティ改善のため、14 のセキュリティガイドラインを規定
- MISRA-C:2004 のルールとの対応づけ
- MISRA C:2012 ルールの適用範囲の対照
  - ISO/IEC TS 17961:2013（プログラミング言語、その環境、およびシステムソフトウェアインターフェイス — C の安全なコーディングルール）との対応
  - CERT C 2016 Edition との対応



Appendix A Summary of guidelines

Code design

Dir 4.14	Required	The validity of values received from external sources shall be checked

Expressions

Rule 12.5	Mandatory	The sizeof operator shall not have an operand which is a function parameter declared as "array of type"

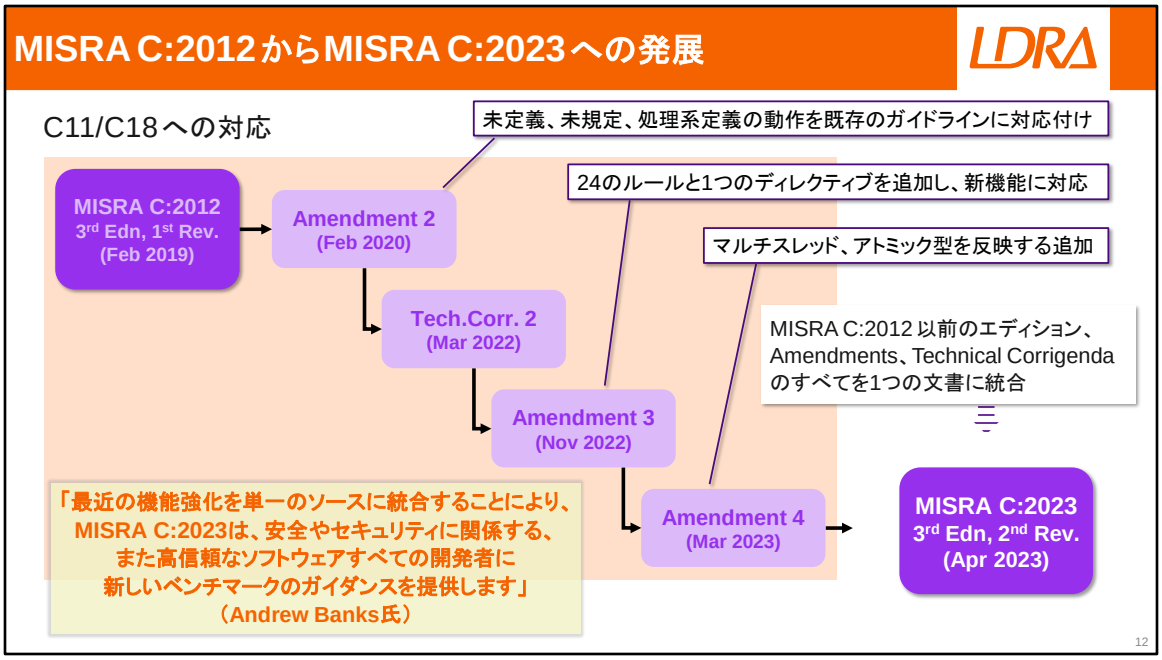
Standard libraries

Rule 21.13	Mandatory	Any value passed to a function in <ctype.h> shall be representable as an unsigned char or be the value EOF
Rule 21.14	Required	The Standard Library function memcmp shall not be used to compare null terminated strings

11

組込みシステムにおいてもセキュリティ対応は必須なものです。現状のガイドラインが安全関連の懸念だけに適用され、セキュリティには適用されないという認識から、MISRA Cワーキンググループは急いでMISRA C:2012の最初の改訂を行いました。コードの安全性に関する多くの要件はコードのセキュリティにも同様に適用されますが、既知の脆弱性に対処するためのガイドラインを追加することで、アプリケーションのセキュリティ方針を改善できるようにしたのです。

すなわち、MISRA C:2012 Amendment 1で、14のセキュリティガイドラインを規定しました。これはAppendix A に示されるこれら追加ガイドラインのサマリーです。
さらに、補遺として、旧版MISRA-C:2004のルールとの対応づけや、ISO/IEC 17961:2013 (C言語規格委員会発行のC言語セキュアコーディング規約) とのルール対応、CERT C 2016 Edition との対応づけが行われました。



今回発行されたMISRA C:2023は、このリビジョン1に対してC11、C18への対応を行ったものになります。未定義、未規定、処理系定義の動作を既存のガイドラインに対応付けた改正・訂正と、C11/18の新機能に対応する改正が実施され、さらに、この3月にマルチスレッドとアトミック型を反映するための追加が行われました。
これらの改正・訂正・補遺をすべて1つの文書に統合したものがMISRA C:2023になります。

次に詳細講演いただきますAndrew Banks氏によりますと、「最近の機能強化を単一のソースに統合することにより、MISRA C:2023は、安全やセキュリティに関係する、また高信頼なソフトウェアすべての開発者に新しいベンチマークのガイダンスを提供します」となります。

MISRA C:2012 Amendment 4 (2023)

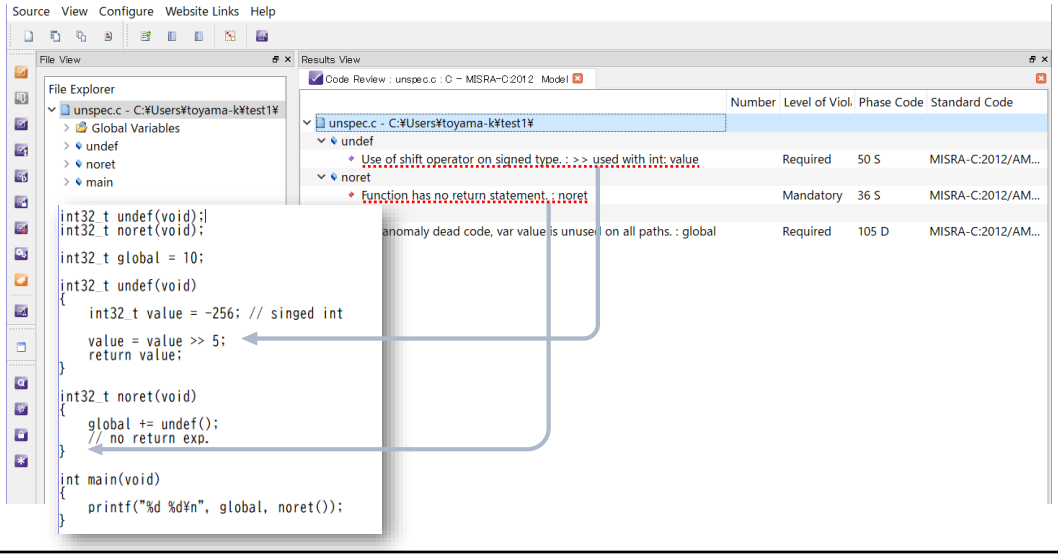
C11/C18 で規定されるマルチスレッドとアトミック型を反映するため、19 のルールと3つのディレクティブを追加

- マルチスレッド機能を安全なサブセットに制限
  - 12のルールと3つのディレクティブを追加
    … 他のスレッドによる意図しないアクセスなど望ましくない動作を最小限に抑える
- アトミック型に関する新しいガイダンス
  - 3つの新ルール、3つの既存ルールの変更 … システム全体でアトミック性を保証
- 追加のガイダンス
  - 4つの新ルール…長年にわたり問題があるとわかっているC言語機能の使用を制限

13

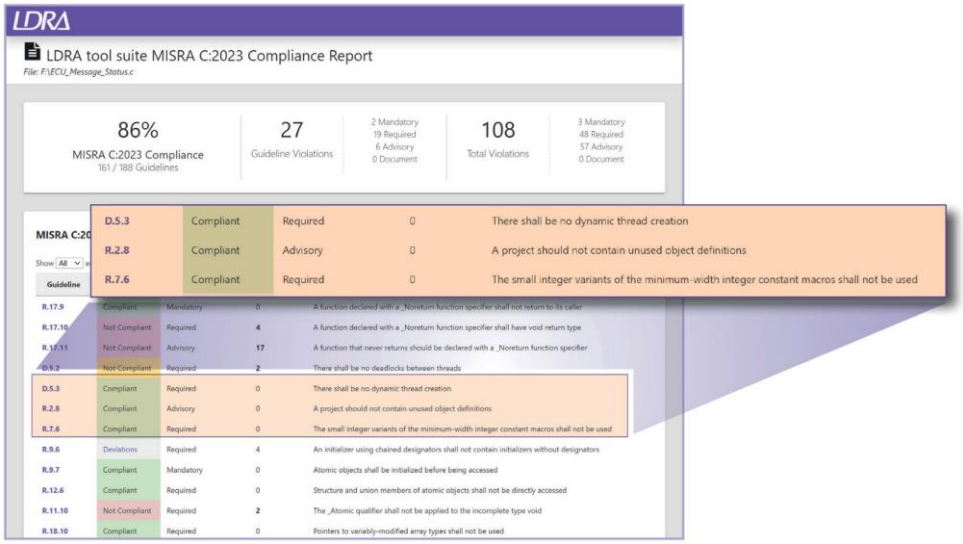
C11では、POSIXのpthreadsなど特定の実装に縛られずに並行性をサポートするため、マルチスレッド処理の標準的な手法が導入されました。この機能追加のために、望ましくない副作用が予期せずシステムにもたらされる可能性があります。組込みソフトウェアでも並行処理が増加していることに注目し、Amendment 4では、マルチスレッドとアトミック型に対するルールとディレクティブを規定し、また、最近のC言語の使い方に合わせて、既存のガイダンスの修正と明確化を行っています。C言語ではスレッド固有のストレージを共通のユーザー空間に割り当てることを妨げませんが、それを制限することによって、他のスレッドによる意図しないアクセスなど望ましくない動作を最小限に抑えます。そのような12のルールと３つのディレクティブが追加されました。また、3つの新ルールと、3つの既存ルール変更で、システム全体でアトミック姓を保証できるようになります。さらに、追加のルールで、これまで問題とされてきたC言語機能の使用を制限しました。

MISRA C準拠のチェック（LDRAツール）



ツールでどのようにMISRA C準拠をチェックするかについてですが、
これは最初にご覧いただきました未定義や処理系依存を含むサンプルコードに対して、LDRA社の静的解析ツールでMISRAの違反を検出して、当該コードにリンクした結果です。

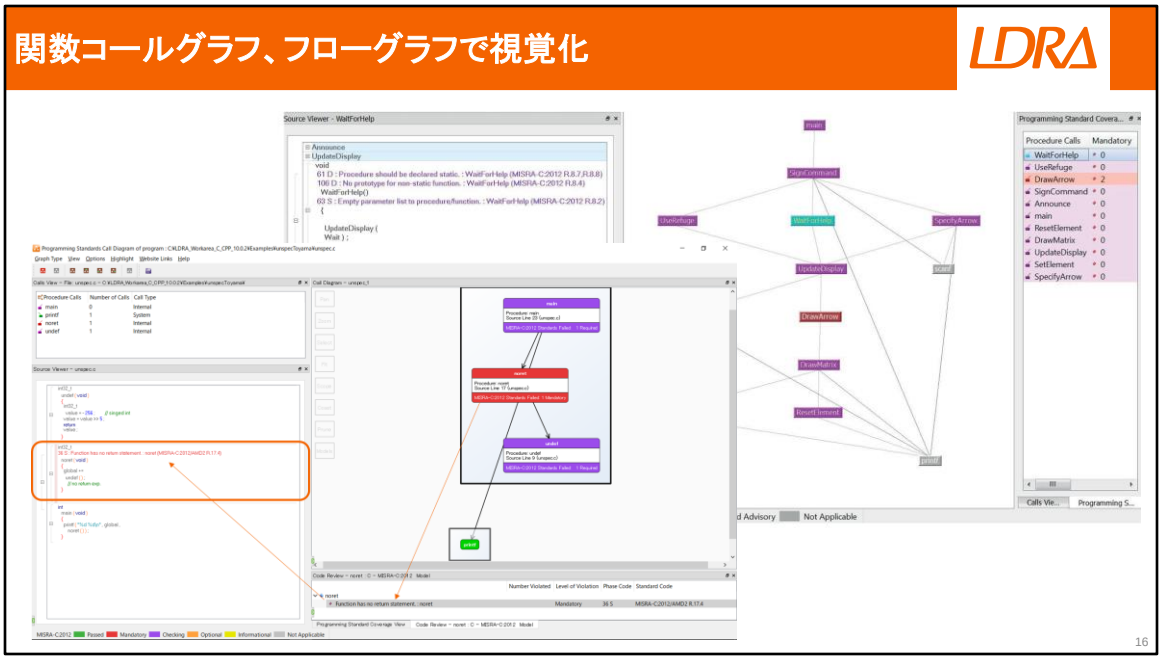
MISRA C準拠レポート（LDRAツール）



LDRAツールスイートは、静的解析を使用して不適合であるコードの領域を特定し、文書化と修正を支援し、MISRA ガイドラインに沿ってソースコードの理解を深めるための広範なレポートやグラフィック表示も行います。

LDRAは静的解析によって得た情報を、単体テストや構造カバレッジ解析にも利用します。これはMISRAガイドラインで推奨される、開発者がテスト済みコードの量を測定し維持することにも活用できるということです。

関数コールグラフ、フローグラフで視覚化



関数コールグラフやフローグラフに解析結果を反映させることで、視覚的に問題個所を特定して、該当するソースコードに辿ることができます。
この視覚化は複雑度解析などのコード品質の尺度や、動的テストのカバレッジ結果にも利用されます。

MISRA Cに準拠することの効果

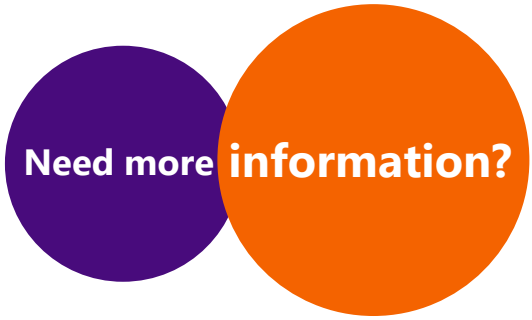
- コンパイラやプラットフォームに固有の言語要素を回避することで移植性を高める
- アプリケーションの予期しない動作を回避する
- 未規定や未定義の動作を回避する
- デッドコード（多くの場合、欠陥や潜在的なセキュリティ脆弱性につながる）を特定する
- 特定の言語要素を禁止することで、安全・セキュアでないコーディングプラクティスを削減する
- プログラムの複雑さが大幅に軽減する
- プログラムのテスト容易性が向上する
- 機能安全やセキュリティ規格への準拠を容易にする

Start your path towards MISRA C now

17

MISRA Cは、あらゆる組込みソフトウェアの認証プロセスにおいて不可欠なコンポーネントであり、静的解析ツールは、規約準拠の目標達成のための貴重な投資です。
これらツールの採用を推奨することにより、組込みソフトウェア開発者は、より高いレベルのビジネス目標へ注力しながら、コードの安全性・セキュリティ・信頼性を向上できます。
静的解析ツールを採用し、MISRA C準拠を目指しましょう。
Andrew Banks氏のご講演につながりましたらと存じます。ありがとうございました。

Contact Us



LDRA.com

LDRA Software Technology

LDRA Limited

@ldra_technology

LDRA Tools

info@ldra.com

18