LDRA
Software Technology

Technical White Paper

MISRA C:2023 と MISRA C:2012 AMD 4 の概要

Get ahead with the MISRA C guidelines

An embedded developer's guide to compliance, including the latest MISRA C:2023 update

www.ldra.com

車載ソフトウェアから医療機器まで、MISRA C は、安全、セキュア、高信頼なコードの実現を推進する有力なガイドラインであり続けています。MISRA C は、ますます複雑化し、つながるという性質も併せ持つ組込み製品の、潜在的な問題を突き止めることで、組込みソフトウェア開発者のニーズを満たします。成熟した組織では、市場での採用率や顧客満足度から MISRA に準拠することの長期的なメリットが認識されており、新規参入者は、電気自動車（EV）や航空宇宙など規制の厳しい業界で競争する上での差別化要因として MISRA を活用しています。

最近リリースされた MISRA C:2012 Amendment 4（AMD 4）と、以前のエディションのガイドラインを統合する MISRA C:2023 には、C11 や C18 をサポートするようマルチスレッドとアトミック型に関する新しいガイダンスが追加され、MISRA はこれまで以上に組込み製品に適するようになっています。

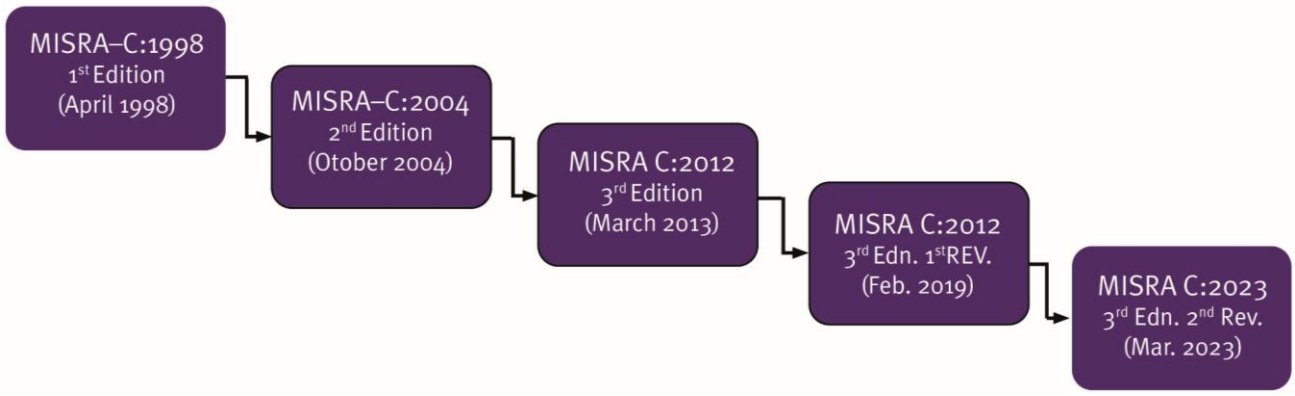

図 1：MISRA C ガイドラインの進化

この資料では、MISRA C ガイドライン、AMD 4 による説明と例、そしてチーム内で規約準拠ツールのフレームワークを推進し配備する戦略について紹介します。

MISRA C の原則を理解する

ソフトウェアの安全性、セキュリティ、信頼性は、多くの組込みシステムにとって必須の重要事項です。C 言語は、もともとアセンブラに代わる軽量で親しみやすい言語として作られましたが、開発者が C 言語を採用する速度が速すぎて、安全性やセキュリティにかかわる重要な懸念事項に対応して構文や意味を更新することが追いつきませんでした。MISRA C ガイドラインは、危険であるとされるコーディング手法を最小限にしたり排除したりするルールとディレクティブによって、C 言語を安全でセキュリティクリティカルなシステムの要件に合致するようにしました。

MISRA C は AUTOSAR や IEC 62304、IEC 61508、ISO 26262、また DO-178C のプロセスに従ったプロジェクトで直接引用されたり、広く実践されたりしており、組込みソフトウェア業界での MISRA C の影響はどれだけ誇張してもしすぎることはありません。また、Joint Strike Fighter C++ Coding Standard や NASA Jet Propulsion Library Coding Standard の基礎ともなっています。

MISRA ガイドラインの進化と組込みソフトウェアの複雑化によって、規約準拠の検査ツールも数十年の進歩を遂げました（MISRA C 自体も規約準拠を確保するために静的解析ツールを使用することを推奨しています）。

MISRA C が重要な理由

C 言語では、システムの安全性、セキュリティ、信頼性を損ねかねない方法によってアプリケーションの動作やメモリアクセスを制御できてしまいます。アプリケーションのコードは C 言語規格の要件を満たしていても、クリティカルシステムにおいて次の例のような望ましくない影響を与え、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。

- 読み取り専用でオープンされたファイルストリームへ書き込むと、望ましくない動作になる可能性があります
- 自身を呼び出す（再帰）関数を使用すると、スタックオーバーフローの可能性があります
- データ構造の範囲外のメモリにアクセスすることで、ハッカーがよく使用するエクスプロイトにつながる可能性があります

C 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、コードベースに問題が導入される前に防ぐ方が、効率的で費用対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、排除するよう開発者を導くことができます。

また静的解析ツールを使用して MISRA C 準拠を自動化することで、チームはライフサイクルのできるだけ早い段階で問題を特定し修正できます。

C 言語への準拠

以下に示す MISRA C ガイドラインの発展は、C 言語のバージョンの推移と MISRA C ワーキンググループが把握しているユーザー実体験に沿ったものです。

- **MISRA C:1998** – 「Guidelines for the use of the C language in vehicle based software」というタイトルの最初のエディションには C90 に対する 127 のルールがあり、現在でもレガシーシステムのメンテナンスで使用されています
- **MISRA C:2004** – このエディションでは、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「Guidelines for the use of the C language in critical systems」と変更し、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しました
- **MISRA C:2012** – このエディションでは、対象範囲を C99 までに拡大し、ガイドラインの背景にある理論的根拠、チェックツールでどのようにして規約準拠の問題を特定できるかを明確にするための「決定可能」/「決定不能」ルールの分類、そして準拠の解釈がより柔軟にできる「ディレクティブ」の追加など、ユーザーからのフィードバックを取り入れました

MISRA C:2012 は進化を続けて、特にセキュリティに配慮する追加を行い、最近では C 言語規格の新しいバージョン (C11、C18) を取り入れています。

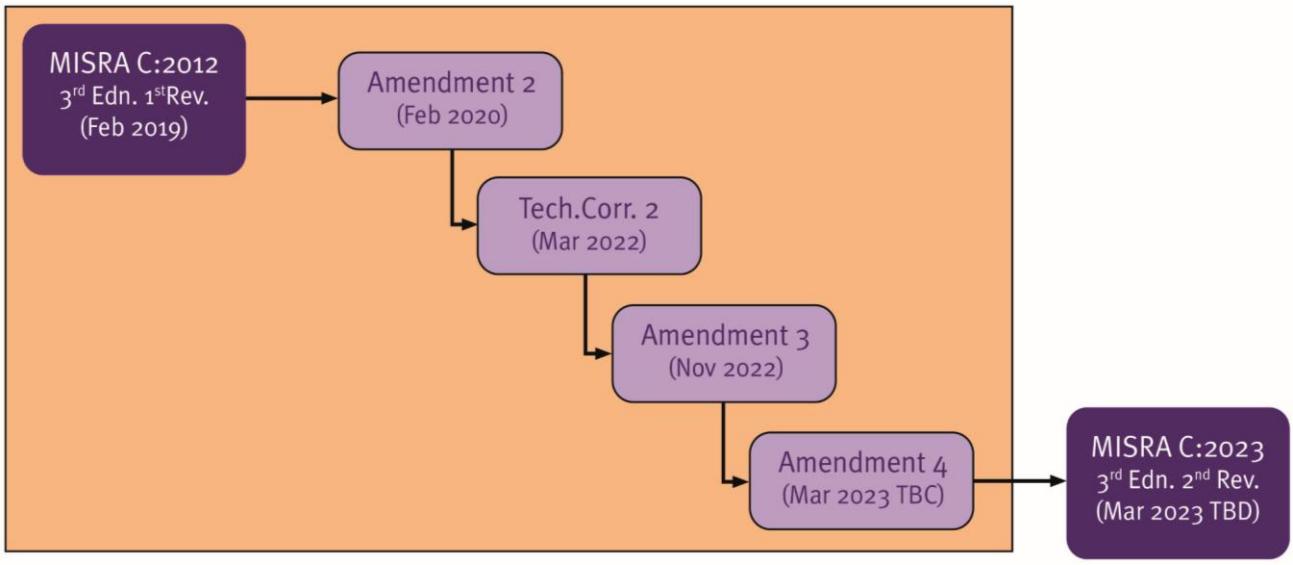

図 2：MISRA C:2012 から MISRA C:2023 への進化

- **MISRA C:2012 Amendment 1 (2016)** – 既存の MISRA C:2012 ガイドラインと 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) の適用範囲の比較結果として公開されたもので、組込みシステムのサイバーセキュリティ方針の改善への適用性を認識し、14 のセキュリティガイドラインを規定しました
- **MISRA C:2012 Amendment 2 (2020)** – 未定義および未規定の動作を既存の MISRA C ガイダンスに対応付けることで、ISO/IEC 9899:2011 規格 (C11) をカバーするようにしました
- **MISRA C:2012 Amendment 3 (2022)** – ISO/IEC 9899:2011/2018 規格で概説された新しい C11 および C18 の言語機能を反映するため、24 のルールと 1 つのディレクティブを追加しました
- **MISRA C:2012 Amendment 4 (2023)** – ISO/IEC 9899:2011/2018 規格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しました

この一連の文書をサポートするために、MISRA は次に示すようにガイドラインと関連規格との対応付けを作成しました。

- MISRA C:2012 Addendum 2 – MISRA C:2012 と ISO/IEC 17961 "C Secure" との対応付け
- MISRA C:2012 Addendum 3 – MISRA C:2012 と CERT C コーディング規約との対応付け

そして、MISRA C:2023 は、MISRA C:2012 以前のエディション、Amendments、および Technical Corrigenda のすべてを 1 つの文書に統合したものです。

「MISRA C:2023 は最近の機能強化を単一のソースに統合することにより、
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」
Andrew Banks（LDRA テクニカルスペシャリスト、MISRA C ワーキンググループ議長）

さらに読む：素晴らしい MISRA C:2023 でのアップデート

MISRA ガイドラインの構成

MISRA C:2012 では、次のようにガイドラインをルールまたはディレクティブとして分類しました。

ルール：完全で客観的、かつ曖昧さのないソースコードの要件です。開発者は解析ツールを使用してルールに準拠することを確認できます。さらに、解析によって確定的に検証できるものを決定可能、検証による保証が不可能なものを決定不可能として分類しています。

ルールの例：「関数へのポインタと他の型の間で変換を行ってはならない」（MISRA C:2012 ルール 11.1）

ディレクティブ：コードやプロセス、文書、または機能要件を通じて満たされる可能性のあるガイドラインです。ディレクティブは解釈の対象になりますので、解析ツールによって準拠の検査を支援できる場合があります。

ディレクティブの例：「プログラムの出力が依存する処理系定義の動作は、文書化され、理解されなければならない」（MISRA C:2012 ディレクティブ 1.1）

さらに読む：LDRA はどのように MISRA C をサポートするか

MISRA C とサイバーセキュリティ

現状のガイドラインが安全関連の懸念だけに適用され、サイバーセキュリティには適用されていないという認識から、MISRA C ワーキンググループは MISRA C:2012 の最初の改訂をリリースしました。コードの安全性に関する多くの要件は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明示的なガイドラインを確立することで、開発者と静的解析ツールにアプリケーションのセキュリティ方針を改善するためのフレームワークを提供できました。

MISRA C:2012 のルール 12.5 の「`sizeof` 演算子は、'`array of type`' として宣言された関数パラメータであるオペランドを持ってはならない」を例に挙げます。配列のサイズを計算するためによく使用される `sizeof()` 関数は、その引数が関数パラメータとして渡された配列である場合、潜在的な脅威ベクトルにつながる可能性があります（図 2 参照）。図 2 の 6 行目で `sizeof()` に `A[]` を渡すと、関数に「型へのポインタ」として渡されるため、不正な値になる可能性があります。その結果、配列の境界を超える可能性があり、ハッカーによくつけこまれることになります。

```c
#include <stddef.h>
#include <stdint.h>

static int32_t calculate_threshold (int32_t A[ 2 ])
{
   size_t size = sizeof ( A ) / sizeof ( A[ 0 ] );
   ...
   return A[ size - 2 ];
}

int main (void)
{
   int32_t A[] = { 1, 2 };
   ...
   return calculate_threshold ( A );
}
```

図 3：sizeof() 関数の潜在的に危険な使用を示すコード例

MISRA C:2012 Amendment 4 の紹介

MISRA C:2012 Amendment 4 では、2022 年後半にリリースされた Amendment 3 に倣って、C11（ISO/IEC 9899:2011）および C18（ISO/IEC 9899:2018）規格で導入された多くの新機能をサポートするよう拡張されています。組込みソフトウェアでの並行処理の増加に焦点を当てて、AMD 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、開発者の最近の 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。

マルチスレッドに関する新しいガイダンス

C11 では、開発者が特定の実装（POSIX pthreads など）に縛られないで組込みシステムでの並行性のニーズをサポートするために、マルチスレッド処理の標準的な手法が導入されました。スレッドや条件変数、ミューテックスなどの機能が追加されて、開発者が予期しない望ましくない副作用をシステムにもたらしてしまう可能性がありました。

AMD 4 では、マルチスレッド機能を安全なサブセットに制限するための 12 のルールと 3 つのディレクティブを追加することで、これらの問題の多くに対処します。そのガイドラインでは、スレッドの使用に関する以下の重要な要素をカバーしています。

- 動的なスレッド生成を制限して、並行性に対してより決定論的なアプローチを促進する
- ミューテックスがリンクされる前にスレッドが生成されることを保証する
- システム内のデッドロックとデータ競合のリスクを最小限に抑える
- スレッドオブジェクトとスレッド ID を安全に使用するよう管理する

たとえば、AMD 4 には、1 つのスレッド内だけでアクセスされることになっているメモリストレージは、そのスレッドによってのみアクセスされなければならない、という新しいルールがあります。C 言語では開発者が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ないため、このルールは、他のスレッドによる意図しないアクセスや望ましくないアクセスを最小限に抑えるのに役立ちます。

C 言語のアトミック型に関する新しいガイダンス

C11 では、アトミック型やアトミック操作についてのセマンティクスとメモリモデルも導入され、開発者はデータオブジェクトを不可分に、すなわち他のスレッドによる干渉のリスクなしに操作できるようになりました。これは、データ競合の可能性を減らすことが目的でしたが、C 言語には開発者がシステム全体でアトミック性を保証することを困難にする機能もあります。

AMD 4 には、3 つの新ルールと、3 つの既存ルールの変更があり、次のように開発者が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。

- アトミック型の正しい構成を保証する
- ポインタを介してアトミック型を参照する際、意図せずにアトミック性が除去されることを防止する
- 同一文内での複数のアトミック型の使用を制限する

追加のガイダンス

AMD 4 アップデートには、長年にわたり問題があるとわかっている C 言語機能に対する 4 つの新ルールが含まれており、以下を制限するものになります。

1. コンパイラがマクロを展開するときに、予期しない、あるいは予測できない動作が発生するケースから保護するために短整数マクロを使用すること
2. 集成型 (配列・構造体・共用体) について、同一要素を多重に初期化することを防ぐため、同一型に対する指示付き初期化子 (コンマ区切りリストの変数) と、指定なし初期化子 (明示的な要素名) を組み合わせて使用すること
3. 可変長配列 (VLA) の使用制限を維持しながら、可変に変更される配列の制限を (ガイダンス付きで) 緩和すること
4. アプリケーションで決して参照されない変数を使用すること (これは、未使用のマクロや構造体、その他のデータ型に関する既存のガイダンスに追加されます)

さらに、MISRA C:2012 のルール 3.1 (「コメント内で文字シーケンス /* および // を使用してはならない」)を更新して、Web サイトの URL をサポートするために「//」の使用を認める例外を設けるなど、既存のガイダンスを最新のものにするためのマイナーな改良がいくつかあります。

MISRA C:2023 の紹介

最初の MISRA C ガイドラインの発行から 25 周年を記念して、MISRA C ワーキンググループは MISRA C:2023 を発行しました。このエディションは、以前のエディションと最近の AMD 4 の機能拡張を統合したもので、既存の MISRA C ユーザーには単一の包括的なベースラインを提供し、新規プロジェクトを開始する企業には参入を容易にします。

MISRA C への準拠を開発プロセスに組み込む方法

MISRA C への準拠は、そのガイドラインに照らしたテストとレポートが、トレーニングや文書化、ツール、そしてメトリクスに組み込まれるようにソフトウェア開発プロセスを更新することから始まります。組込みソフトウェアの性質とできる範囲を考えると、これらのプロセスは通常、静的解析ツールと文書、そして手作業の組み合わせによって MISRA C の決定可能/決定不能なルールへの準拠を検証しなければなりません。

開発プロセスを更新する場合、チームは次の項目を考慮する必要があります。

MISRA 準拠のプロセスを文書化する

MISRA は、プロジェクトのソフトウェア開発プロセスに MISRA C ガイドラインを含めることを推奨しています。これらのプロセスは形式化されて、完全で曖昧さのない正しいソフトウェア要件を保証する厳格なアプローチが明確にされ、その実装はすべて、開発ライフサイクルの各フェーズの出力だけによってなされなければなりません。

このアプローチの一部として、MISRA 準拠（すなわち、人・プロセス・ツールをどのように連携してそれを達成するか、また違反の報告を含めて MISRA C ガイドラインへの準拠を主張するための要件を詳述すること）が必要です。

自動化が役立つ場所を理解する

一般に、MISRA の決定可能ルールへの準拠は、静的解析など自動化された手法によって証明できますが、決定不能ルールについては追加の裏付け証拠が必要になる場合があります。決定不能ルールの検証を単一のツールで行おうとすると、正確な評価を保証するのに十分な情報がないため、誤検知や検知漏れとなる可能性があります。

MISRA のディレクティブは通常、主観的なものであり、単一の自動化ツールで検証することは困難です。たとえば、MISRA C:2012 のディレクティブ 1.1 では、「プログラムの出力が依存する処理系定義の動作は、すべて文書化され理解されなければならない」と要求しており、これは、静的解析では検証できない成果物です。

逸脱を文書化する方法を知る

「ガイドライン違反の管理は重要な問題です。違反は時として避けられないものであり、
明確に定義されたプロセスを通じて正当と認可され、
逸脱記録によって裏付けられた場合にのみ、準拠を主張することが意味を持ちます」

- MISRA Compliance:2020 Guide

MISRA は、ガイドラインに準拠することを証明することが現実的でないか、または従うことが不可能である場合、それから逸脱することを認めています。それらの逸脱は、図 4 に示す例のように、対象のガイドライン、逸脱の場所、逸脱の根拠、リスク分析、を含めて文書化されて、承認されなければなりません。

Project	F10_BCM		
Deviation ID	D_00102	Status	Approved
Permit	Permit / Example / C:2012 / R.10.6.A.1		
Rule 10.6	The value of a composite expression shall not be assigned to an object with wider essential type		
Use case	The value of a composite expression is assigned to an object of wider essential type to avoid sub-optimal compiler code generation		
Reason	Code Quality (Time behaviour)	Scope	Project
Tracing tags	D_00102_1 to D_00102_10		

Raised by	E C Unwin	Approved by	D B Stevens
	Signature		*Signature*
Position	Software Team Leader	Position	Engineering Director
Date	14-Mar-2015	Date	12-Apr-2015

図 4：MISRA 逸脱記録の例（出典：MISRA Compliance:2020 guide）

自分用の MISRA 準拠のツールスタックを定義する

新しい準拠ツールを選択することが難しいという場合がありますが、MISRA 自体がガイダンスを提供しています。MISRA では、静的解析ツールを使用してルールとディレクティブに対してコードを検証することを推奨しています。MISRA Compliance:2020 guide では、以下の基本原則に従って、静的解析ツールを選択し設定する方法についての情報を提供しています。

1. ツールは、プロジェクトに関連する MISRA のエディションをサポートし、最新の AMD 4/MISRA C:2023 を含むできるだけ多くのガイドラインを適用できなければなりません
2. チームは、ツールの出力に確信が得られるようツールを検証しなければなりません（IEC 61508、ISO 26262、DO-178 など一部の規格では、特定の状況で静的解析ツールの認定が必要です）
3. ツールは、必要な言語と環境の特性（コンパイラの設定や言語の拡張機能など）をサポートしなければなりません
4. ツールは、コードカバレッジと追加のレビューが必要となるコードを特定するのに助けとなるメトリクスを生成できなければなりません
5. ツールは、オープンソースソフトウェアやベンダーライブラリ、デバイスドライバなど、レガシーコードやサードパーティコードの解析をできる限りサポートしなければなりません

最近の組込みソフトウェアの開発速度向上をサポートし、アジャイルプロセスや継続的インテグレーション/継続的デリバリー（CI/CD）のパイプラインに適合するための静的解析ツールの必要性については、明示的に言及されていません。

LDRA はどのように MISRA 準拠をサポートするか

LDRA は、LDRA ツールスイートによる自動コードチェックやレポート作成、専門家によるトレーニングとコンサルティングサービスを通じ、LDRA は MISRA 準拠の合理化と有効性を改善しています。航空宇宙や産業・エネルギー、医療機器、そして自動車分野などの組込みソフトウェアチームは、LDRA MISRA チェッカーを使用して、コードの安全性・セキュリティ・信頼性を確保しています。

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。また LDRA 静的解析ツールは、MISRA ガイドラインで推奨されているように、開発者がテスト済みコードの量を測定し維持できるよう、構造カバレッジ解析に必要な情報も生成します。

LDRA は長年にわたって MISRA をサポートしており、規格準拠、ソフトウェア検証の自動化、静的コード解析、そしてテストツールに関する専門知識を活かして、最新の C 言語と開発手法に合わせて規格を進化させてきました。LDRA の技術スペシャリストである Andrew Banks と Chris Tapp は、それぞれ MISRA C ワーキンググループ議長、MISRA C++ワーキンググループ議長を務めています。

図 5：LDRA ツールスイートでの MISRA 準拠レポートの例

LDRA の MISRA 準拠ツールのフルセットには、以下の機能が含まれます。

- 最新の AMD 4/MISRA C:2023 を含む、すべてのエディションの MISRA 言語サブセットの準拠に関する自動ソースコードチェック
- コードの理解を簡素化し、問題を迅速に修正するための広範なレポートと視覚化
- MISRA ガイドラインで推奨されている、テストカバレッジの信頼性を高めて経時変化を追跡するための構造的カバレッジ解析
- 正当なルール逸脱の効率的な管理および追跡
- 機能安全規格 IEC 61508、ISO 26262、IEC 62304、EN 50128 に対する TÜV SUD 認証の取得

MISRA 準拠を成功させるために静的解析をどのように推進するか

MISRA 準拠のために不可欠な要素として静的解析ツールの採用を提唱するには、品質やセキュリティ、効率の向上によるメリットと、採用しない場合の潜在的なリスクとコストを明確に伝えることが重要です。また、静的解析の導入に成功した他の組織の事例を提供し、それが規約準拠の作業にプラスの影響があることを強調することも役立つ場合があります。

「MISRA C:2012 の遡及適用は、想像していたよりも難しくないと判明しました。
LDRA のルールは高レベルの MISRA ガイドラインを
より詳細で完全な定義に分解しているので、大きな助けとなりました」
- 事例：Renishaw

静的解析ツールを導入するようソフトウェア開発リーダーを説得するには、次の 4 つの重要な議論があります。

1. 準拠への取り組みを合理化し、コストを削減する

静的解析ツールは、リスクを特定し、疑わしいコードをピンポイントで特定して開発者がさらに調査できるようにすることで、MISRA 準拠の効率を向上させます。最近のコードベースは多くの場合、複数のソフトウェアユニットにわたり、複数の場所に保存され、異なる製品ラインにまたがっており、その規模と複雑さを考えると、これらのツールは手動での検査や従来のテスト方法よりもはるかに包括的で正確です。

静的解析によって次のように、開発期間を短縮し MISRA 準拠のコストを削減することにつながります。

- 開発者のデスクトップ上で、オリジナルのソースコードのコンテキストで MISRA 違反を明らかにすることにより、ソフトウェア単体テストの前に問題を特定します（「シフトレフト」）
- 従来の方法では規模を拡大することが難しいか、コスト的にできない包括的なコードカバレッジを実現します
- 開発者がコードを変更するたびに、継続的なコーディングと準拠チェックの規範を適用することにより、不具合を事前に防止し、技術的負債を最小限に抑えます
- 静的解析手法では、コーディング規約への準拠だけでなく、複雑さや明確さ、保守性、テスト容易性など、コードの全体的な品質を測定することもできます

2. MISRA C に関する開発者トレーニングのサポート

静的解析ツールはルール駆動型である傾向があり、その出力は、開発者が自身のコードに MISRA C がどのように適用されるかをよりよく理解するための理想的なトレーニングの場を提供します。経験豊富な開発者であろうと新人であろうと、解析結果は一種のマイクロラーニングであり、規約準拠の問題や関連情報がコードのコンテキストで理解しやすく、より定着しやすい一口サイズのチャンクに分割されます。

この「瞬時の」トレーニングは、MISRA C に慣れていない人は迅速に習得できるようにし、経験豊富な開発者では準拠することよりも機能に集中できるようにガードレールを提供します。

動画：Software engeneering: Being compliant with MISRA C and MISRAC++

3. MISRA のエディションを 1 つのソースに統合する

MISRA のエディションやガイダンス文書に気を配ることは、特に組織が複数の開発ストリームをサポートしている場合には面倒なことがあります。多くの MISRA の成果物が公開されていくにつれて、ドキュメントがファイル共有、デスクトップ間、さらには本棚にまで広がるため、信頼できる唯一の情報源を維持することはほぼ不可能になります。

規約準拠活動の一貫性と正確性は、MISRA 文書のリポジトリを 1 つ確立することによって得られます。静的解析ツールは、すべての開発ストリームで MISRA のエディションを追跡し、現状を最新のアップデートに保つので、理想的な選択肢です。手動ツールや開発者のハードドライブに依存するのではなく、静的解析ツールを使用することで、すべての人にとって準拠に関する真実のソースを 1 つにできます。

4. 新規開発におけるリスクを低減する

MISRA を初めて使用する場合、チームは長期にわたる認証機関と顧客の期待に応える認証成果物を得るために必要な、計画、テスト、追跡、報告など、困難な課題を抱えることになります。EV の新興企業から新市場に進出する成熟したデバイスメーカーまで、静的解析ツールを用いると、MISRA C に基づく堅牢な安全性とセキュリティの準拠チェックに向けて有利なスタートを切れます。

今すぐ MISRA C への道を歩み始めましょう

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、静的解析ツールを採用することは、規約準拠の目標を達成するための貴重な投資です。これらの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の注力も示しながら、コードの安全性・セキュリティ・信頼性を向上させることができます

LDRA は設立以来、MISRA C の作成と継続的な開発に貢献しており、その発展を担う MISRA C ワーキンググループにおいて強い代表性をもっています。LDRA 静的解析ツールは、あらゆる組織の準拠活動に対する包括的かつ最新のサポートを通じて、このコラボレーションを反映しています。

LDRA が MISRA C 準拠への道をどのように支援できるかについての詳細は、
https://www.fuji-setsu.co.jp/products/LDRA/MISRA.html をご覧いただくか、
https://www.fuji-setsu.co.jp/#cont3 からお問い合わせください。

www.ldra.com
LDRA UK & Worldwide
Portside, Monks Ferry,
Wirral, CH41 5LH
Tel: +44 (0)151 649 9300
e-mail: info@ldra.com

LDRA Technology Inc.
2540 King Arthur Blvd, 3rd Floor, 12th Main Lewisville Texas 75056
Tel: +1 (855) 855 5372
e-mail: info@ldra.com

LDRA Technology Pvt. Ltd.
Unit B-3, Third floor Tower B, Golden Enclave
HAL Airport Road Bengaluru 560017
Tel: +91 80 4080 8707
e-mail: india@ldra.com

MISRA C: 23 White Paper v2.1 03/23

富士設備工業株式会社 電子機器事業部
https://www.fuji-setsu.co.jp