An Introduction to

MISRA C:2012

Getting familiar with the new

MISRA C language subset

（日本語版）

www.ldra.com

Introduction

MISRA C はCプログラミング言語の言語サブセット（コーディングスタンダードとしてよく言及される）で、Motor Industry Software Reliability Association (MISRA)によって制定されメンテナンスされている。自動車業界のセーフティクリティカルな組込みアプリケーションに安全なC言語の使用を促進することを目的に考案され、初代バージョンはC90をターゲットに1998年にリリースされた（MISRA C:1998)。

そして言語サブセットは航空宇宙、通信、医療、防衛、鉄道など他の産業界のセーフティ/ライフ/ミッション・クリティカルなアプリケーションにも広く採用されるようになった。これを受けて2004バージョン（MISRA-C:2004）では組み込み向けに対象を広げ、初代バージョンを改善し、名前を変更した。

最新のアップデート MISRA C:2012 は2013年3月18日にリリースされ、ISO 9899:1999 (C99)のサポートを追加する（C90のサポートも維持)。新しい言語サブセットでは、プログラマによる準拠への労力を軽減してコーディングにより多くの時間を費やせるようにすることで、セーフティクリティカルなアプリケーションのソフトウエアに関わるリスクを緩和する。
アップデートされた言語サブセットでは、ルールをより正確に定義して、正当な使用や振舞いが阻害されることのないようにした。
加えて開発者はルールの施行に対するより良いガイダンスを得られる。例えば、あるルールはプロジェクトを通して全体的な振舞いを定義しているのか、特定のケースのみであるのかなど。

2012版では、できる限りルールが決定可能（decidable：解析ツールが必ず判断できる）であることを設計目標とした。これに対して、ポインタやデータ値によってコントロールフローが影響されるものは決定不可能（undecidable）- Figure 1　決定不可能なルールでは、解析ツールが十分な情報を得ることができないので、偽陽性（false-positive)、偽陰性（false-negative）といった結果を導くことになる。この改善により、マニュアルによるコードレビューが飛躍的に軽減される。

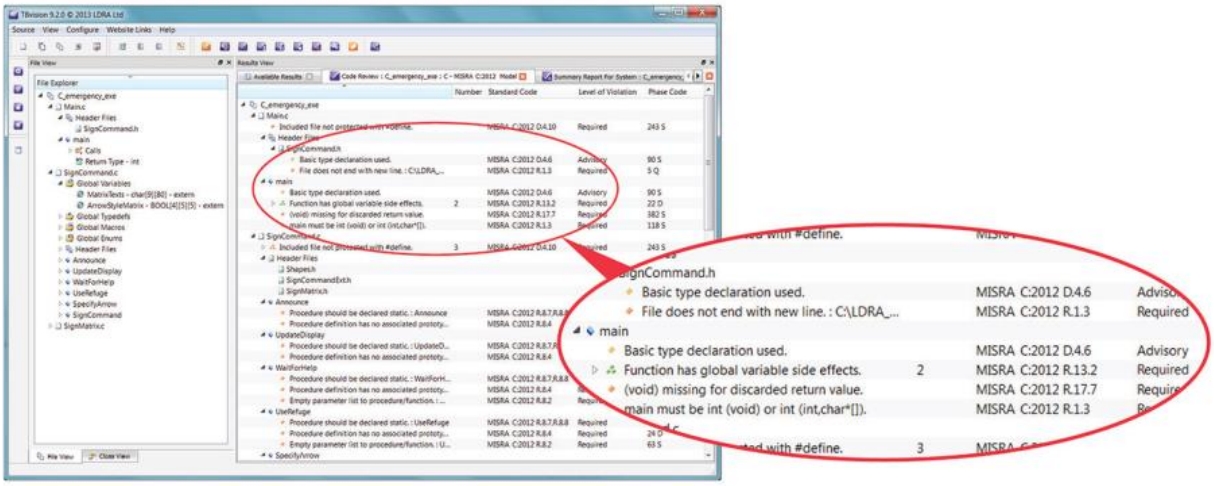

Figure 1: TBvision ツールでMISRA C:2012 の決定可能（decidable）なルールを識別。また様々な決定不可能（undecidable）なルールのチェックも支援。当該コードから違反の詳細記述へのハイパーリンクなど、ユーザフレンドリーになった最新版MISRA スタンダードをサポート

MISRA C:2012 を採用することで、C99 へ移行して最新の言語機能の優位性を取る、あるいはC90 ベースで継続、いずれの場合でも様々なセーフティクリティカルなアプリケーションの規格認証を支援できる。

Developing a new version

MISRA C の三代目を検討するうえで、いくつかの不都合があった。開発者が新しいバージョンを学ぶことや、ツールベンダーの対応が必要なことなど。

結局のところ、これらの不都合は以前の改訂版のフィードバックに対する修正や経験を基に最小限にできた。一方C99サポートの要件がMISRA Cスタンダードの3代目への大きな動機付けになったものの、単純にC99をカバーする新しいルールの追加だけでは済まなかった。

MISRA-C:2004 を MISRA C:2012に進化させるうえで、ルールに対するレビューを適切な経験に基づいて実施した

- 各ルールはどちらのC標準（C90か、C99） 、あるいは両方を適用するかを明確に
- 未定義の動作（undefined behaviour)、未規定の動作（unspecified behaviour）の全ての事例をルールでカバー
- 全てのルールを決定可能（Decidable)、決定不可能（Undecidable）で分類する。決定不可能なルールは少しの整復によって決定可能にできるかどうかによって判断された（残る大抵の決定不可能なルールはコントロールフローに影響するポインタやデータ値）
- ルールの施行については見出し（Headline text）の単一行に記載。詳細をルールを規定するテキスト（normative text）で補足
- ルールの精度を改善して、望ましくない使用のみが制限され、妥当なものは制限されないようにされた

総体的に既存のもの（よりprescriptive(規定的）で柔軟性に欠ける）に比較して、よりわかりやすく参考となるドキュメントになった。

Examples of specific beneficial changes

ユーザフレンドリーになったMISRA C ドキュメントへの進化は、様々な角度から見ることができる。

mandatory（必須）ルールの採用

既存のRequired（必要)、Advisory（推奨）ルールに、逸脱してはならない Mandatory（必須）ルールの分類が新しく加わった。(Figure 2)　この分類以外のルールは、それぞれに該当した判断手順で逸脱することができる。Advisory（推奨）はプログラマの裁量、Required（必要）は管理者の承認が必要など。

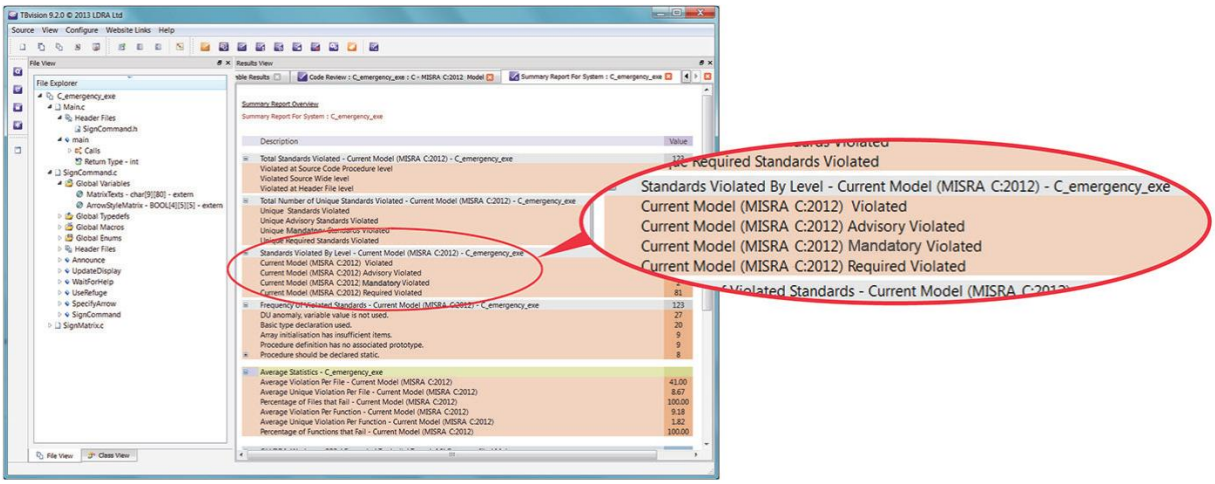

Figure 2: MISRA C:2012 では逸脱してはならない Mandatory（必須）の分類がルールに導入され、既存のRequired（必要）、Advisory（推奨）の分類を補完する。TBvisionでは、検出された違反ごとに分類して要約できる。

例：

1. *Rule 22.2 A block of memory shall only be freed if it was allocated by means of a Standard Library function.*

Rule 22.2　標準ライブラリ関数でアロケートされたメモリブロックなら解放してよい

変数を使用するためのメモリアロケーションを自動で開放する様々な手法がある。これは正当なCの構文ではあるが、危険であり、不必要。このルールは開発者が巧妙に過ぎることを未然に防ぐことを目的にする。

```
void fn ( void )
        {
        int32_t a;
        free ( &a ); /* Non-compliant - a does not point to allocated storage */
        }
```

2. Rule 22.4 There shall be no attempt to write to a stream which has been opened as read-only.

Rule 22.4　リードオンリーとしてオープンされるストリームへの書き込みは存在しない

ISO/IEC 9899（C 言語標準）では、リードオンリーストリームへの書き込みが意図される場合の振舞いについて仕様として定めていない。その理由からリードオンリーのストリームへの書き込みは安全ではないと考慮された。

このルールは誤りを防ぐことを考慮。そうすることに何の利益もないし、ありえそうにもないので。

既存ルールを部分的に変更

既存のルールは改正され、洗練され、調整された。　MISRA C:2012では、確立されたコンセプトである理論的根拠（rationale）の解説（何故各ルールが有効であるか）を強化した。　このアプローチは、MISRA C:2012の完全な導入を目指す場合でも、インハウスのスタンダードのベースにすることを目的にしている場合でも有益。

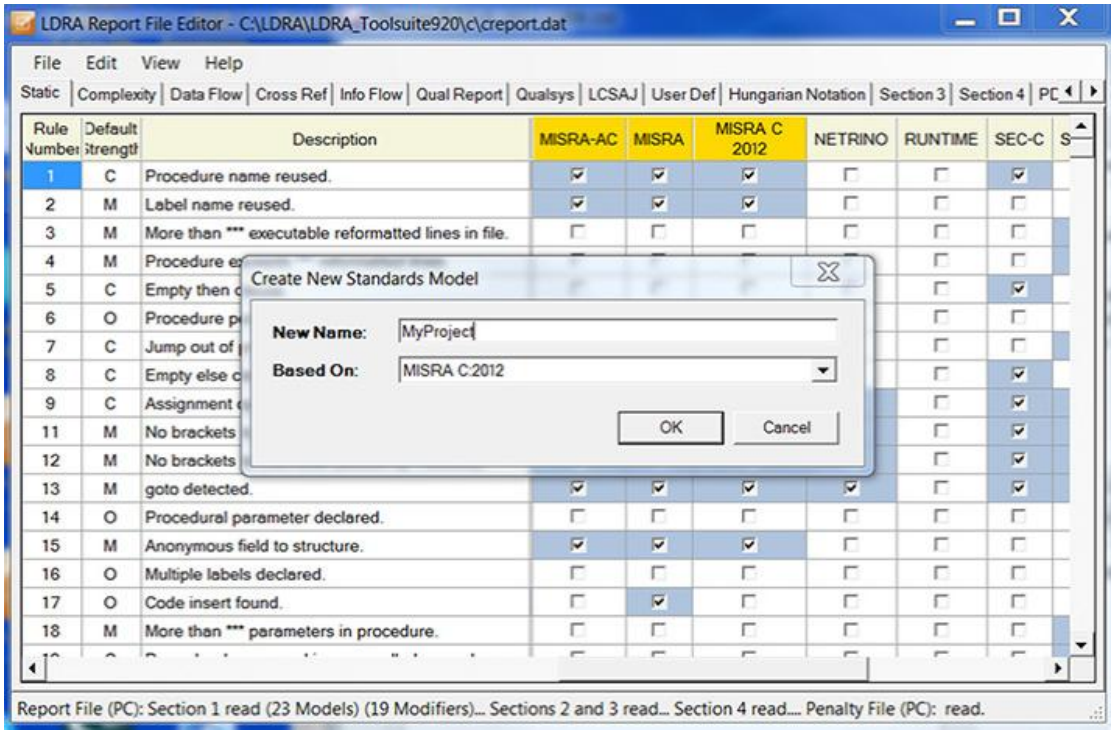

Figure 3: TBvision や TBrules ではMISRA のスタンダードをベースに企業や開発プロジェクトごとで固有のルール集を編集できる。

例：
１．“goto”文の使用が認められる箇所があるが、大抵は熟考されていないか、あいまいなアルゴリズムに継ぎ接ぎして使用されている。

プロセス制御のアプリケーションに例外など非常時の状態がある場合、アルゴリズム内でフラグをセットして後でチェックするほうが、goto で直行させることより本当に優れているといえるか？

Rule 15.1 The goto statement should not be used
Rule 15.1 goto文を用いてはならない

は Advisory（推奨）とされ、Required（必要）ではなくなった。そして２つのRequired（必要）ルールによって、容認できる範囲を限定する。

Rule 15.2 The goto statement shall jump to a label declared later in the same function.
Rule 15.2　goto文は同一関数内で後に宣言されるラベルにジャンプする

と

Rule 15.3 Any label referenced by a goto statement shall be declared in the same block, or in any block enclosing the goto statement.
Rule 15.3 goto文によって参照されるラベルは同じブロック内で宣言されるか、goto文を含むブロック内で宣言される

２．Rule 12.1 式中における演算子の優先順位は明示的にすべき

は、MISRA C 2004の言語サブセットのルール（*式中におけるC言語の演算子優先順位規則への依存は、極力制限すべき*）から差し替えられた。この表現とデシジョンテーブルの精度の改善により、ユーザとツール開発者が一貫性のある解釈をとれるようになる。

改良と洗練：″処理系定義″　個々のコンパイラに依存した動作について

ハードウエア/ソフトウエアインターフェイスに関わる個々の観点で、新しい言語サブセットをより良く定義する様々な改訂がある

例：
・typedef 使用に関わるDirective（指示・通達）は、“Required”から“Advisory” に改訂された。

Dir 4.6 typedefs that indicate size and signedness should be used in place of the basic numerical types
Dir 4.6　基本型の代わりにサイズ及び符号属性(signedness)を示すtypedefs　を用いなければならない

例えば、32ビットのC90の実装では以下の定義は適切：

```
typedef signed char int8_t;
typedef signed short int16_t;
typedef signed int int32_t;
typedef signed long int64_t;
```

理論的根拠(rationale)の項目では、移植性の観点から、一つの整数型表現がint型のサイズ次第で汎整数昇格(または汎整数拡張)の対象になるかどうかが決定されるため、この移植性を保証するガイドラインの厳守が誤認識を招きかねないことを明らかにしている。例えば、型int16_t　の式では、int　が16ビットで実装されると推奨されないが、32ビットでなら起用できる。

システムワイドと単一コンパイルユニットの区別

全ての決定可能（”Decidable”）なルールは理論上決定可能だけれども、全てのコンプライアンスチェッカーが全ての決定可能なルールをチェックできるわけでは無い。例えば、より洗練されたツールスイートならシステムワイドな解析が可能（リンカの作業と同様)であるが、低レベルのツールでは単一コンパイルユニットの解析に過ぎない(コンパイラ同様)。

ルールは解析対象としてシステムワイドと単一コンパイルユニットに分類され、それらを識別することで、使用するツールが後者のみ対応する場合は、代替え手段を取る必要がある。

例：

“Rule 22.3 The same file shall not be open for read and write access at the same time on different streams.”

“Rule 22.3　同じファイルは異なるストリームから同時に読み書きのアクセスのために開かれてはならない

これはシステムワイドな解析を必要とするルールの良い例。上述 22.4 のルールのように、わざとこれをする者はいないだろうが、過ちを回避することができる。システムワイドな解析を介してシステム内の全てのソースコードを参照できるツールなら、この違反の有無を判断できる。

Summary

MISRA C:2012 は既存のMISRA C スタンダードの進化版。内容は新規に採用するユーザにとって有益であるとともに、MISRA を既に採用しているユーザにも親しみやすいようにした。その恩恵は、C99 に適応するためのルール追加だけでなく、ルールの精度の改善や、より良いルールの分類、そしてより包括的な解釈で高い理解を得られるようになったこと。

適切なチェキングツールの支援により、C言語の意図的あるいは不注意による誤用に起因する問題を防ぐことができるようになる。

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, Suite #228 Lewisville Texas 75056
Tel: +1 (855) 855 5372
e-mail: info@ldra.com

LDRA Technology Pvt. Ltd
#2989/1B, 3rd Floor, 12th Main, 80 Feet Road,
HAL II Stage, Bangalore- 560008. Near BSNL Building
Tel: +91 80 4080 8707
e-mail: india@ldra.com

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