PLCopen®

for efficiency in automation

技術資料

PLCopen 普及促進委員会 教育分科会

As part of the

ソフトウェア構築ガイドライン構想

下位委員会

コーディング・ガイドライン

PLCopen Document, Version 1.0, Official Release

Date: April 20, 2016

Total number of pages: 106

これらは PLCopen の公式資料である。

Coding Guidelines

これは PLCopen ソフトウェア構築ガイドライン作成ワーキング・グループの成果を要約したもので
全メンバーの貢献によるものである。

PLCopen		*PLCopen Japan*	
Person	***Company***	***Person***	***Company***
Andreas Weichelt	Phoenix Contact	Ryoko Iwashita	Hivertec
Barry Butcher	Omron	Kenji Hirukawa	Fuji Electric
Bernhard Jany	Siemens	Toru Mizuya	KISTEC
Bernhard Werner	3S / Codesys	Hiroshi Yoshida	Omron
Bert van der Linden	ATS International	Hidenori Sawahara	Yokogawa Electric
Boris Waldeck	Phoenix Contact	Kaoru Komatsu	Zuiko
Carina Schlicker	HS Augsburg	Hideki Kuribayashi	Schneider Electric Japan Holdings
Christoph Berger	HS Augsburg		
Denis Chalon	Itris		
Edward Nicolson	Yaskawa		
Eric Pierrel	Itris		
Geert Vanstraelen	Macq		
Hans-Peter Otto	privat		
Hendrik Simon	RWTH Aachen		
Hiroshi Yoshida	Omron		
Kevin Hull	Yaskawa		
Matthias Kremberg	Phoenix Contact		
Peter Erning	ABB		
René Heijma	Omron		
Rolf Hänisch	Fraunhofer FOKUS		
Sebastian Biallas	RWTH Aachen		
Wolfgang Zeller	HS Augsburg		
Eelco van der Wal	PLCopen		

Change Status List:

Version	***Date***	***Change/Comment***
V 0.1	March 24, 2015	Initial Document as result of work in the original wiki site
V 0.2	June 5, 2015	As result of the webmeeting on June 1 based on the feedback
V 0.3	July 1, 2015	As a result of the webmeeting and last feedback in
V 0.99	July 23, 2015	Version to be published to the open community as Release for Comments till Oct. 23, 2015
V 0.99A	Jan. 17, 2015	As a result of the Face to face meeting in Frankfurt, including decisions on all feedback items
V 0.99B	Jan. 20, 2016	As result of the final feedback and editorial issues
V 1.0	April 20, 2016	Official published version
V 1.0Jp	Nov 20, 2017	Translated to Japanese from V1.0 to V1.0Jp

Table of Contents

1. はじめに

多くのプログラミング言語に対してコーディング・ガイドラインが存在するが、IEC61131-3 やその PLCopen 拡張仕様などの産業用制御における重要な領域にはコーディング規約はほとんど見当たりません。それにもかかわらず、産業用制御ソフトウェアはますます重要性を増しており、ソフトウェア規模の拡大に伴ってエラーコストも増大しています。今日のソフトウェアは初期プロジェクトコストの半分近くを占め、またメンテナンスを含めたソフトウェアの総ライフサイクルコストでも 40～80%に上っています。

大規模プログラムの複雑性を取り扱うには、構造的な手法をとる近代的ソフトウェア開発プロセスが必要とされます。また、定義済み機能の再利用によるコーディングの効率化や、ライフサイクル全体を通したプログラムの理解容易性向上も必要です。

PLCopen は上記のようなメッセージとともに、ソフトウェア構築ガイドライン作成ワーキング・グループを立ち上げるべく関心のあるメンバーを招集しました。キックオフミーティングを開催した結果、異なる関心領域をもった下記の作業グループが構成されることとなりました。

- コーディング・ガイドライン集
- ソフトウェアの品質問題と一貫性
- PLCopen 準拠ファンクションブロック
- SFC による構造化と分解 (すべし、すべからず集)
- ソフトウェア・ドキュメント化ガイドライン
- ライブラリ活用
- ソフトウェア開発プロセス

この新しい PLCopen ソフトウェア構築ガイドライン作成ワーキング・グループにおける主要な議題は、規約、コーディング・パターン、およびガイドラインの定義と、それらを産業用オートメーション領域でどのように活用するかにあります。このワーキング・グループの成果物は、IEC 61131-3 第 2 版までの仕様に基づいて記述されていますが、2013 年 2 月に発行された第 3 版へも容易に適用できるようになっています。

当コーディング・ガイドライン作成サブ・ワーキング・グループの目的は、一連の PLC 実装規約を定義し、これらの規約の活用方法を提案することにあります。近年では大規模なオートメーション企業ではすでに自らの実装規約を策定済みであることが多いですが、中小企業や IEC 61131-3 ビギナーの多くは PLCopen の IEC 61131-3 コーディング・ガイドラインを使用することに大きな関心を寄せています。このようなガイドラインは、IEC 61131-3 を全世界に普及するうえで大きく寄与すると考えられ、またコーディング規約は IEC 61131-3 ユーザのトレーニングや、大学における IEC 61131-3 プログラミングの効果的な教育に大いに役立つことが期待できます。

2. 本文書の利用方法

IEC 61131-3 はプログラミング言語を標準化したもので、PLC テクノロジーにおける大きな前進であるといえる。この標準プログラミング言語を利用することによって制御アプリケーションの品質向上を図ることができるが、IEC 61131-3 にはプログラミング言語をどう利用すればよりアプリケーションの品質をあげることができるかについては記述されていない。IEC 61131-8 としていくつかの利用指針が提供されているものの、ソフトウェア品質上の必要性まではカバーされていない。本文書は、そのギャップを埋める位置づけのものである。

本文書は、アプリケーション・プログラムの製品品質を向上させたいと願う PLC プログラマーに向けて作成されたものである。工程品質については別の技術文書で取り扱うものとし、本文書では取り扱わない。本文書は、コーディングおよびコードレビュー中にプログラマーが参照すべき一連のコーディング規約で構成され、アプリケーションの設計ルールは含まない。このコーディング規約集は、あらゆるものが網羅されているというわけではないが、製品品質に対して十分効果が期待できる内容となっている。各々のプログラマーや開発組織に、本コーディング規約集に対して必要な規約を追加したり、不要なものを削除して活用することを推奨する。

1.

本書の目的はソフトウェア品質の向上である。それには一定の能力あるいは習熟度をもった開発組織が必要である。ソフトウェア開発能力習熟度モデル等を参照すること。

2.

IEC 61131-3 は複数の言語を規定しているが、使用しない言語に関するルールは無視しても構わない。

3.

本文書以外にも、企業による実装規約や一般的なコーディング規約など、品質規約を記載したものが存在すると思われる。多くの大企業ではすでに経験に基づいた標準化が行われている。まずは他に利用可能な規約があるか調べること。また McCall などのような、包括的なソフトウェア品質標準も存在するが、それらは一般的に PLC プログラムには直接適用できないことが多いため、本文書を作成することにした。

4.

プログラマーが全ての規約を一旦まとめ、アプリケーションに適用するものだけを規約のサブセットとして選択しても構わない。もちろんそれが規約のフルセットになることもあるだろう。

2.1. 本書を構成した方法

本書に列挙された規約は、コンピュータサイエンスとして一般的なもの、当ワーキング・グループの参加企業が作成・運用していたもの、PLC プログラム開発の経験に基づき当ワーキング・グループの活動で作成されたものから構成されています。参照した文書は下記の通りです。

- IEC61131-3
- Misra-C
- JSF++
- Codesys on-line help

詳細は 2.4 参考資料を参照してください。

最初の規約リストはスプレッドシートに収集され、行番号が本文書で用いられている各規約の識別子となっています。次に、スプレッドシートの行ごとに作業 Wiki ページを作成し、ワーキング・グループで 1 年半をかけて規約を詳細記述しました。

本書はこうした共同作業の議論の結果である個別の規約の集合として構成されています。

これらの規約は IEC 61131-3 第 3 版を念頭に置いて記述されています。規約の多くは第 3 版以外にも、時には IEC 61131-3 以外にも等しく適用可能です。

本ワーキング・グループは、ユーザからのフィードバックを望んでいます。我々は本書の内容に対し、規約の変更やより有意なサンプルの記述、規約の追加を行うことを検討しています。本書へのご意見を、info@plcopen-japan.jp まで気軽にご連絡いただきますようお願いいたします。

2.2. 本書の構成

本書に記載される規約は、下記に分類されています。

- 命名規約：PLC プログラミング要素に対しどのような名前を与えるかに関する制約事項や命名形式についての規約。
- コメント規約：コードに対して、読みやすく理解しやすく保守しやすいコメント付与方法に関する規約。
- コーディング規約：コーディングに関する様々な規約。
- 言語固有規約：IEC 61131-3 の各振る舞い記述言語に固有の規約。
- ベンダー固有拡張仕様規約：すべての PLC ベンダーで利用できるわけではない IEC 61131-3 言語仕様への拡張仕様に関する規約。

各章は、優先度の高いものから順に規約を記載する。

各規約の識別子には参照が容易なように接頭語が付与されており、CP がコーディング規約、N が命名規約、C がコメント規約、L が言語、E が拡張を表現する。識別子は各分類ごとに CP1 のようの通し番号 1 から始まっており、将来追加がある場合は末尾に追加される。

2.3. 規約記述フォーマット

本書の規約は同じテンプレートにのっとって記載されています。本節では各項目に記述される内容について説明します。規約詳細は以下の例のように記述されます。

識別子: rule XX

重要度: 高

適用言語: すべて

参考文献:

- Misra-C_2004 rule 3.5

内容: ユーザ定義構造体への参照時には…

ガイドライン: メンバーにはオフセット指定により …

理由: オフセット指定による構造体メンバアクセスは、コードが理解しづらく…

例外事項: なし

サンプル、もしくはユースケース:

悪い例:

```
STRUCT example
  X : DINT;
…
```

良い例:

```
instance.Z[1] := 'E';
```

備考: なし

- 識別子：必須
  規約を参照するためのユニークな識別子。接頭語に続く番号に特に意味はない。

- 重要度：必須
  高、中、低　のいずれか。アプリケーションソフトウェアの品質に与える影響度の指標。

- 適用言語：必須
  規約によって適用言語の種類が異なるため、適用する言語

- 参考文献：必須
  参考となる文書・文献

- 内容：必須
  規約の説明。

- ガイドライン：必須
  規約違反に対処するための説明。

- 理由：必須
  規約の妥当性の説明。

- 例外事項：任意
  特定のケースにおいて規約に適用しない場合がありその妥当性の説明。

- サンプル、もしくはユースケース：任意
  規約の理解を促進するための例。

- 悪い例
  コーディング例を赤字で記載。

- 良い例
  緑字で記載

- 備考：任意
  規約の説明を補完するために本書の範囲外の他の規約や他の題材を参照することもある。

2.4. 参考資料

以下の規約や委員会を参照する。

IEC 61131-3
NAME: IEC 6-1131-3 edition 2 and 3

5. コーディング作法

5.1. メンバには名前でアクセスすること

識別子：Rule CP1

重要度：高い

適用言語: 全て

参考文献：

- Misra-C_2004 rule 3.5

内容： 構造体のようなユーザ定義データ型をコード中で参照する際には、構造体の先頭とメンバーの位置の間のメモリ上のオフセット値を使うのではなく、メンバー名を使うべきである。

ガイドライン： オフセット値によるメンバへのアクセスは避けること

理由： オフセット値による参照は読みにくく、分かりにくく、メンテナンスも難しい。もし何らかの理由で他のメンバが挿入された場合、それ以降のメンバのオフセット値は全て計算し直さなくてはならない。さらには（ビッグエンディアンやリトルエンディアンのような）メモリの実装に依存する。特に PLC のメモリ配置の規則は定められていないため、構造体のオフセット値は PLC の機種に依存する。

例外事項： なし

サンプル：

悪い例：

```
STRUCT EXAMPLE_STRUCT
  X : DINT
  Y : BOOL;
  Z : STRING[40];
END_STRUCT;

VAR
instance : EXAMPLE_STRUCT AT %MW500;
END_VAR
// Write the first character of Z:  Z の最初の文字を書き込む:
%MW504 := 'E';
```

良い例：

6. 言語固有の作法

6.1. インデントについて決めておくこと

識別子：: Rule L1

重要度：: 低

適用言語: ST, IL, テキスト形式の宣言

参考文献：

- JSF++ 44

内容： インデントの使い方を決めることができる。その場合，プロジェクト内では一貫性のある使い方をすること。

ガイドライン： インデントには 4 つのスペースを使うこと。

理由： コード中のインデントは、特に条件分岐やループの記述について、可読性の助けとなる。小さなインデント（例えば，スペース 1～2 個）は，必ずしも分かりやすいものではない。一方、大きなインデント（例えばスペース 8 個）では、入れ子になったコードが画面に対して幅が大きくなりすぎる可能性がある。

例外事項： なし

サンプル、ユースケース:

*悪い例：*入れ子の内側の IF 文は Sort の呼び出しと同じインデントにすべきである。

```
IF sizeListToSort < 0 THEN
   Sort (numberElements := sizeListToSort,
      direction := 2,
      idEse := idEse,
      Ese := Ese,
      status => statusTemp);

      IF NOT statusTemp THEN  // このインデントは適切でない
            status := statusTemp;
      END_IF;
END_IF;
```

良い例：

```
IF Dem_froid OR Rep_chaud OR Prem_cycle THEN
    Cmd_vanne_remplissage.Temps_ma := 15;
    Cmd_vanne_vidange.Temps_ma := 15;

    Local := true;
ELSE
```

```
    init_graph := false;
END_IF;
```

備考：6.5.8「タブの使用の定義」も参照。

6.2. ファンクションブロック図（FBD）

6.2.1. ネットワーク内部で中間結果の代入は行わないこと

識別子：Rule L2

重要度：中

適用言語: FBD

参考文献：なし

内容：ネットワーク内部で中間結果の代入は行わないこと

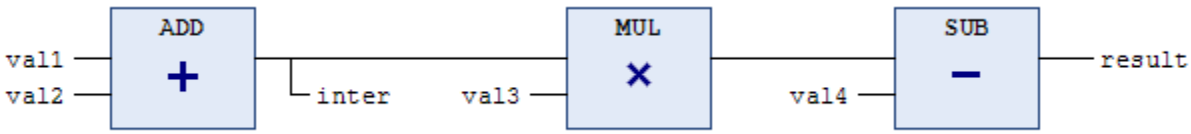
ガイドライン： FBD のネットワークにおいては、ブロック間の接続途中で変数への代入を行わないようにすること。

理由： そのような代入がある場合、ネットワーク内での副作用を理解することが困難になる。

例外事項： なし

サンプル：

悪い例：



良い例：