ＰＬＣの国際標準プログラミング入門

IEC 61131-3 と PLCopen®

IEC 61131-3 の特長① ～5 種類のプログラミング言語～

PLCopen Japan 代表幹事
松隈 隆志

IEC 61131-3 の特長

本稿では IEC 61131-3 の主な特長について、規格に準拠していない従来型のプログラミングツール（以下、従来ツールという）と比較しながら解説いたします。

1. ５種類のプログラミング言語

IEC 61131-3 には、日本で普及しているラダーダイアグラムを含む３種類のグラフィック言語（LD/FBD/SFC）と２種類のテキスト言語（IL/ST）が規定されており、エンジニアのスキルや適用用途に応じて最適な言語を選ぶことが出来ます。※1)

※1）IEC 61131-3 では全ての言語のサポートを PLC ベンダーに強制はしていません。よって、IEC 61131-3 準拠のプログラミングツール選定の際はサポートされている言語に注意して下さい。

① LD（ラダー ダイアグラム：Ladder Diagram）
リレーシーケンス回路の置換えや従来ツールに慣れているエンジニア向けのグラフィック言語で、日本の制御システム開発において最も普及しているプログラミング言語です（図 1)。

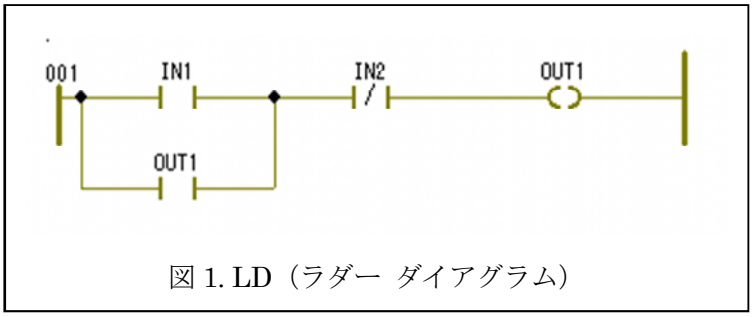

図 1. LD（ラダー ダイアグラム）

LD はリレーシーケンス回路と同等なロジックを使用しており、I/O のインターロック処理など、ビットレベルの処理には向いていますが、システムが大規模且つ複雑になるほど機能単位での区分（モジュール化）が難しく、一本の巻物スタイルになってしまいます。その為、第３者にとっては解読が難しく、他のシステムへの流用や将来発生すると思われる改造要求に対応し難いところが欠点です。

② FBD（ファンクション ブロック ダイアグラム：Function Block Diagram）
DCS(Distributed Control System)に慣れているエンジニア向けのグラフィック言語で、計装分野を中心に使われています。最近では制御適用範囲の拡大や、プログラムの可読性が向上するという理由で PLC でも使われることが多くなりました。
図 2 のように、FBD 言語は電子部品（ファンクションと呼ばれる箱）とそれらを接続する配線により、あたかも電子回路を設計するようにプログラムを記述出来る為、データの流れが一目見てわかるという利点があります。ファンクションの左側にある変数は入力パラメータ、演算結果である右側の変数は出力パラメータと呼びます。

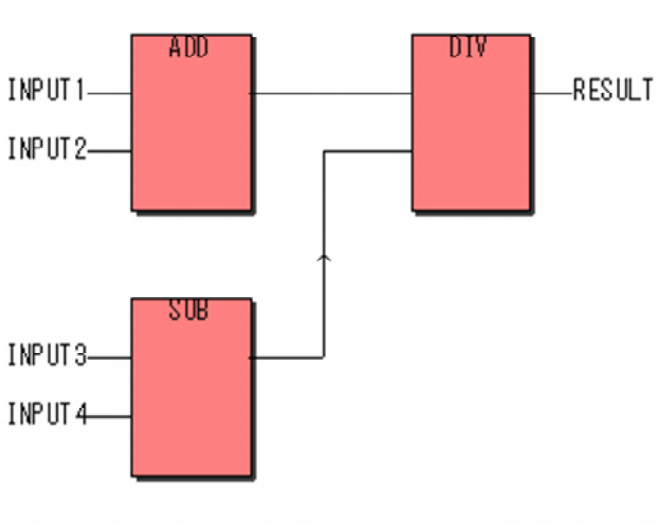

図 2. FBD（ファンクション ブロック ダイアグラム）

③ SFC(シーケンシャル ファンクション チャート：Sequential Function Chart)

製造ラインなどの状態遷移を記述するのに適したグラフィック言語です。SFC は演算機能や入出力機能を持たない為、厳密には言語ではなく「要素」と規定されていますが、言語要素を併せ持つため、ここでは言語として扱います。図 3 に図 1 の自己保持回路と等価な SFC プログラム例を示します。

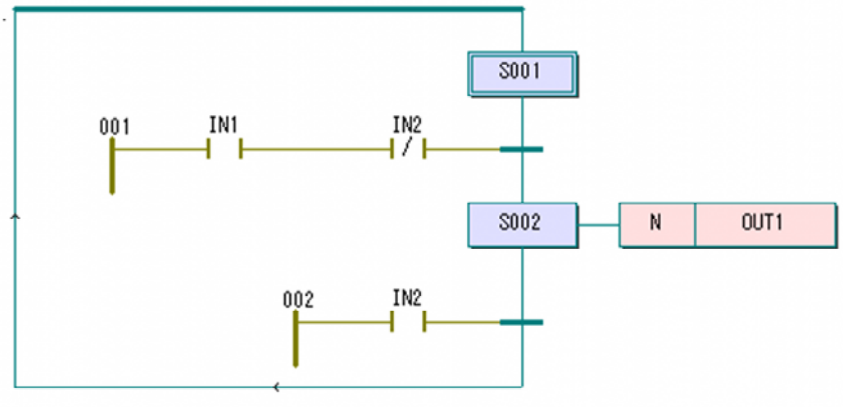

図 3. SFC（シーケンシャル ファンクション チャート）

四角の箱の S001、S002 は“ステップ”と呼ばれ、工程の状態を示します。特に 2 重四角の S001 のことは“初期ステップ”と呼んでいます。SFC 内では同時に 1 つだけのステップが実行され、実行状態にあることを“活性化”と呼びます。ステップの下に位置する横棒は“トランジション”と呼ばれ、次のステップへ移行する為の遷移条件となっています。S001 から S002 への遷移条件は“IN1”が TRUE 且つ“IN2”が FALSE となることです。この条件が成立したときに S001 が非活性化されると同時に S002 が活性化されます。S002 の右横の四角は“アクション”と呼ばれ、S002 で行う処理を記述します。S002 が活性化状態の時、“OUT1”に TRUE が出力されます。“OUT1”への TRUE 出力は、S002 の下に記述されたトランジション“IN2”が TRUE になるまで継続します。“IN2”が TRUE になると、S002 は非活性化状態になり、最初の S001 が活性化状態となります。

このように、SFC では工程間の遷移条件や工程内の処理を分けて記載出来るところが利点です。

④ IL（インストラクション リスト：Instruction List）

従来ツールにあるニーモニックに相当する言語で、マイコンで言えばアセンブラのようなテキスト言語です。アプリケーションの小型化や高速化に有効ですが、プログラミングの生産性やメンテナンス性に劣る為、使用する機会は減ってきていると思われます。図 1 の自己保持回路を IL で記述すると図 4 のようになります。

```
LD    IN1
OR    OUT1
ANDN  IN2
ST    OUT1
```

図 4. IL（インストラクション リスト）

⑤ ST（ストラクチャード テキスト：Structured Text）

PASCAL をベースに設計された構造化テキスト言語で、パソコンの高級言語に慣れ親しんだマイコンボードの開発者や C、C++などの教育を受けてきた新卒のエンジニアに向いています。特に数値演算式やデータ処理など、LD が苦手としている用途で効果を発揮します。仮に LD で多項式あるいは括弧付きの計算式を処理しようとした場合は、初めに二つの項式に分解し、次に分解された其々の演算結果を一時的にメモリに格納してから、最後にそれらを繋げるといった処理が必要になってしまいます。このことは処理内容の可読性を著しく阻害してしまいます。

図 5 は図 2 の FBD と同じ処理です。“INPUT1” と “INPUT2” を加算した値を “INPUT3” から “INPUT4” を引いた値で除算し、結果を “RESULT” に代入しています。

```
RESULT:= (INPUT1 + INPUT2 ) / (INPUT3 - INPUT4);
```

図 5. ST（ストラクチャード テキスト）

以上のようにプログラミング言語にはそれぞれ得手・不得手があり、適材適所で使い分けることによりコーディング時間の短縮や可読性の向上を図ることができるのです。

JEMA（社団法人日本電機工業会）によって作成された 5 言語の適正表（表 1）を目安にして下さい。

主な使用状況	LD言語	IL言語	ST言語	FBD言語	SFC言語
単純なリレーシーケンス処理	◎	×	△	△	×
数式演算処理	△	×	◎	○	×
状態せん移に基づく順序制御 （ステップシーケンス処理）	△	×	○	×	◎
連続的なアナログ信号処理	△	×	○	◎	×
複雑な情報処理	△	×	◎	△	×
プログラムメモリ制約の厳しい場合	○	◎	△	△	×
最も高速に性能を求められる場合	○	◎	○	○	×
運転方案と対応がとりやすい表現	×	×	○	○	◎
動作を視覚的に確認したい場合	○	×	×	◎	○
注記　記号の意味は，次による。 ◎：最も適している，○：適している，△：困難な場合もある，×：適さない					

表 1. プログラミング言語の得意・不得意

（出典：JEMA「PLC アプリケーションの開発効率化指針」）

2. 変数によるプログラミング

変数とは、データを格納する入れ物（メモリ）につける名前のことです。従来ツールでは、入れ物の名前はアドレスであり、アドレスに付属する情報として名前（コメント）を付けていました。IEC 61131-3 では、最初に変数（信号名）があり、属性としてアドレスの定義が可能となっています。表２のように、従来ツールではベンダー毎に固有のアドレスが指定されている為、メーカー変更は勿論のこと、機種変更の際にもアドレスの管理に注意と労力が必要でした。その結果、多くのユーザーは特定ベンダーの PLC を使い続けることになったのです。

一方 IEC 61131-3 では、外部機器に接続される入出力信号以外はアドレスを割り付ける必要がありません。これにより、実績のあるプログラムを流用する際でも変数の変更は不要です。例えメーカーを変更する場合であっても、IEC 準拠の PLC 間では 外部入出力のアドレス定義だけに注意をすればよいのです。また、分かり易い変数名を付けることでプログラムの「可読性」も向上します。社内における変数の命名ルールを決めておけば、次回に説明するコンカレント開発（分業）も容易になります。

	従来ツール		IEC 61131-3		
	A社	B社	①変数(信号名)	②変数の型	③アドレス
一般メモリ	V1.8	M100	運転準備	BOOL	自動割付
	VD1	D0	風量	DWORD	自動割付
	VD2	D10	運転日	DATE	自動割付
リテイン(保持)メモリ	MD3	D100	累積運転時間	TIME	自動割付
デジタル入力	I0.0	X00	ファン始動	BOOL	%IX1.0.0
デジタル出力	Q1.7	Y01	ファンモータ	BOOL	%QX2.0.0
アナログ出力	AQ4	D1000	風量	INT	%QW3.0

①IECでは変数(信号名)でのプログラミングが基本。
②変数のデータ型も厳格に定義 → 誤り防止。
③外部入出力など、絶対アドレスが必要なもののみ、アドレスを指定。

表 2. 変数とアドレスの定義

参考文献；
図 1～図 5 は「はじめてのＳＴ言語（PLCopen Japan 監修）」より