PLCopenの最新技術動向

-Function Blocks for Motion Controlの現状とこれから-

PLCopen Japan
Motion Control WG
2009. 11. 27

発表内容

1. Function Blocks for Motion Controlのコンセプト
2. 技術仕様の種類と状況
3. 開発環境
4. 適用可能なアプリケーションと動作例
5. 今後の取り組み

1．Function Blocks for Motion Controlのコンセプト(1)

プログラミング言語の標準化　＜IEC61131－3の環境を利用＞
[シーケンスを制御するPLCと、同一のプログラミング言語(環境)を採用]

Function Blocks for Motion Control を提唱

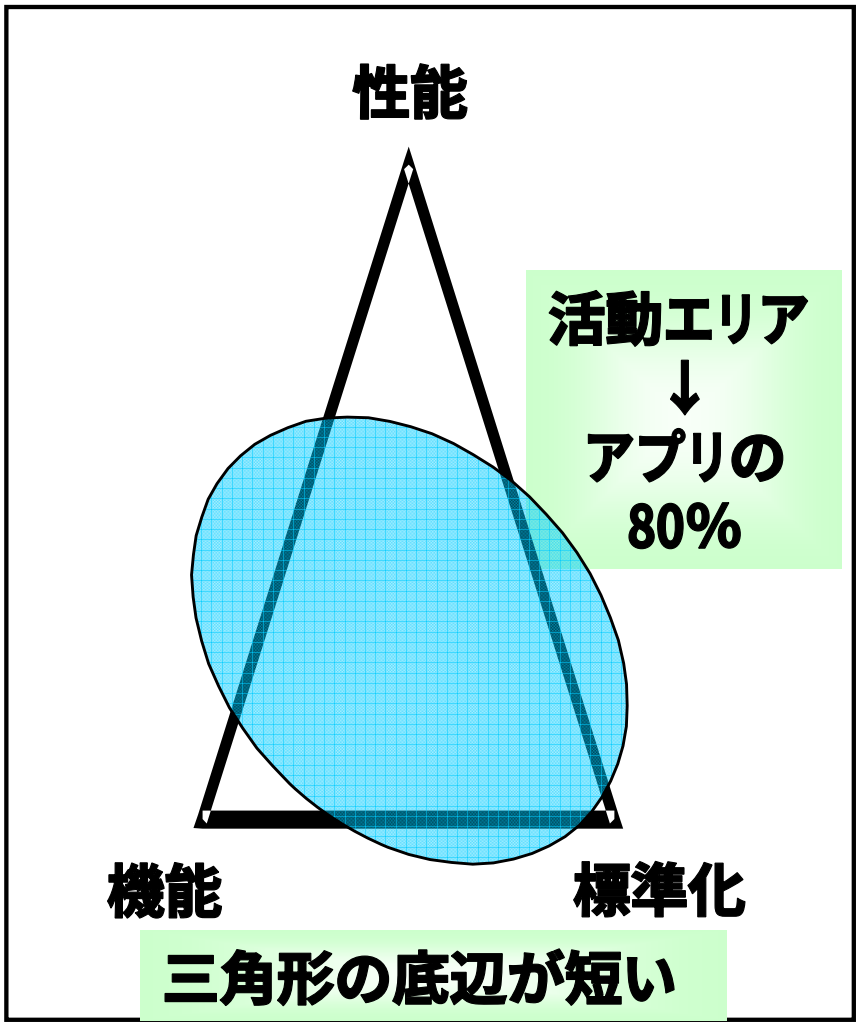
仕様だけで無く、プログラムのインターフェースまで定義(標準化)

ハードウェアへの依存性を低減

⇒ アプリケーションソフトウェアの再利用性を向上
⇒ トレーニングコストを低減
⇒ 様々のアプリケーションに対応
＜5つのPartで仕様を構成し、単軸～多軸～補間機能を実現＞

1. Function Blocks for Motion Controlのコンセプト(2)

ユーザの3つの選択肢



- 性能の追求:
  ハードウェアに密接なプログラム
- 機能の拡充:
  ユーザには非常に有用(広範な適用)
- 標準化:
  トレーニングコストを最少化

本仕様の位置付け

＜最高性能は求めず、豊富な
機能および、標準化を狙う＞

⇒ アプリの80%カバー

⇒ 三角形の底辺が短い

(標準化を基本に機能性重視)

2．技術仕様の種類と状況（1）

5つのPartで規定している仕様の内容と各々の関係 ⇒ 6つのPartに？

- Part 1 - Function Blocks for Motion Control
  ＜基本仕様＞
- Part 2 - Extensions
  ＜Part1からの拡張仕様＞
- Part 3 - User Guidelines
  ＜ユーザガイドライン＞
- Part 4 – Coordinated Motion
  ＜多軸間の協調動作仕様（補間機能）＞
- Part 5 – Homing
  ＜原点サーチ関連の追加仕様＞
- Part 6 –Fluid Power
  ＜ Fluid Power関連の仕様を追加･統合化予定＞

2．技術仕様の種類と状況（2）

各Partのリリース状況

- Part 1 - Function Blocks for Motion Control
  ＜Ver1．1　：　2005/04/09　ﾘﾘｰｽ＞⇒翻訳公開
- Part 2 - Extensions
  ＜Ver1．0　：　2005/09/16　ﾘﾘｰｽ＞⇒翻訳公開
- Part 3 - User Guidelines
  ＜Ver0．4　：　2008/04/18　ﾘﾘｰｽ＞
- Part 4 –Coordinated　Motion
  ＜Ver1．0　：　2008/12/03　ﾘﾘｰｽ＞⇒翻訳中
- Part 5 - Homing
  ＜Ver0．99：　2005/11/10　ﾘﾘｰｽ＞
  ［ 2010年初めに、Ver1.0をリリース目標 ］

※ Logic，Motion，Safetyを融合した仕様を検討中

※ 新規に、Part6：Fluid　Powerの仕様を検討開始

3．開発環境

シーケンスと同じ開発環境で、モーションのアプリケーションを開発

[システム構成]
汎用PLCシステムを利用してモーションを制御するシステムを構築

従来 ： モーション制御用のアプリケーションプログラムを開発する
専用の開発環境が必要

本仕様 ： ＜IEC61131-3のプログラミング言語を採用＞

シーケンス制御用のアプリケーションプログラムを開発するのと同じ
開発環境を使用可能 ＜標準化＞

4. 適用可能なアプリケーションと動作例 (1)

基本仕様だけで、適用可能なアプリケーション

[単軸～独立多軸／マスタ･スレーブの仕様 (パート)]
- ☆ Part1：基本仕様　⇒　単軸～多軸制御，管理用の命令を準備
- ☆ Part2：Part1の拡張仕様
- ☆ Part5：原点サーチに関して、Part1への追加仕様

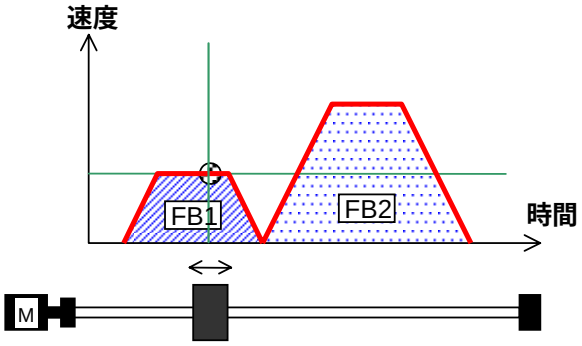
[各種の搬送アプリケーションや、独立多軸の位置決め用途等]
- ☆ 単軸～多軸の単純位置決め動作
- ☆ ***モード指定を利用した、単軸～多軸の連続動作***
- ☆ マスタースレーブ方式の多軸動作
- ☆ ***機械式カムの置き換え***
- ☆ ギア動作 (速度比率動作)

4. 動作例 ＜単軸動作：Demo#1＞

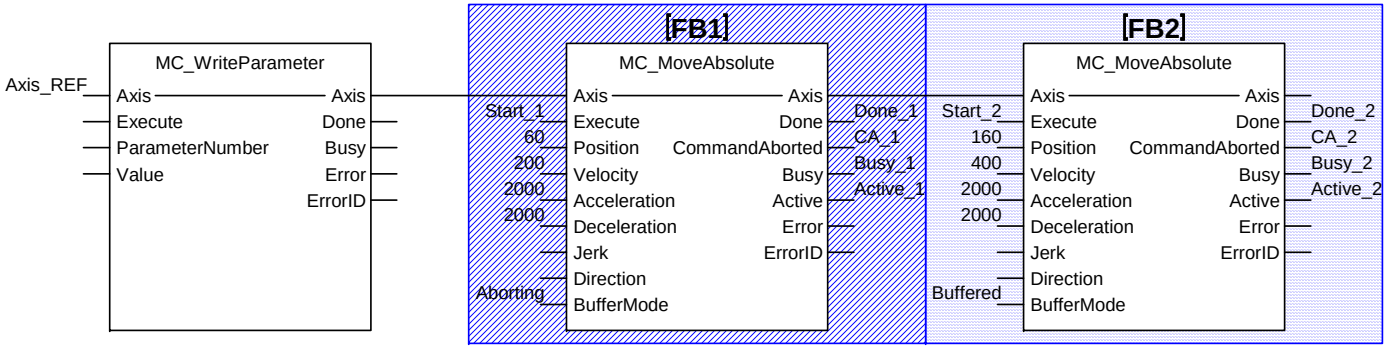
単独軸において、モード指定によりFB間の動作の違いを実現

ビデオ クリップ

Buffered Mode

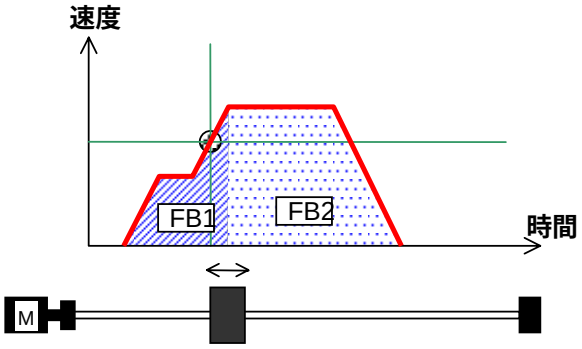


＜Buffered Modeの Program例＞

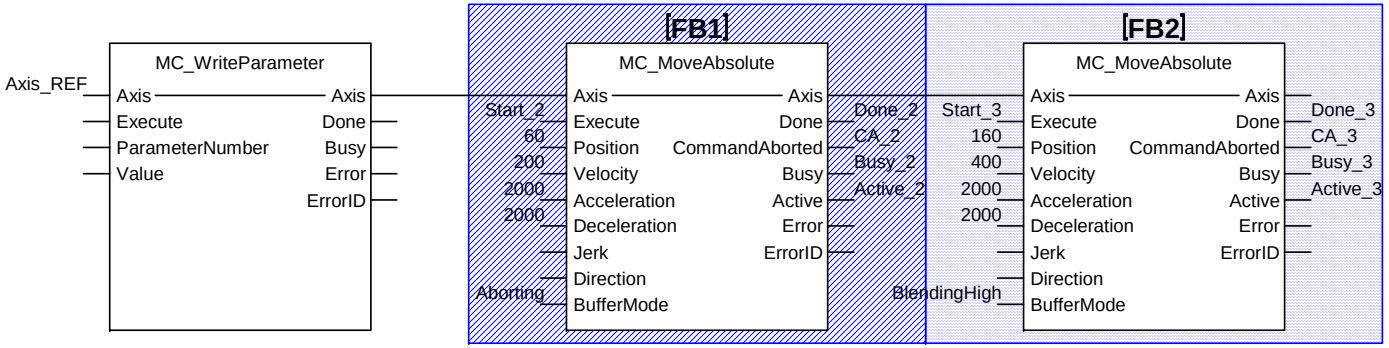


※ [FB1] の動作が完了すると、直ぐに [FB2] が動作

Blending High Mode



＜Blending High Modeの Program例＞



※ [FB1] の最終位置では、[FB2] の速度 (速い方) を使用して連続的に動作

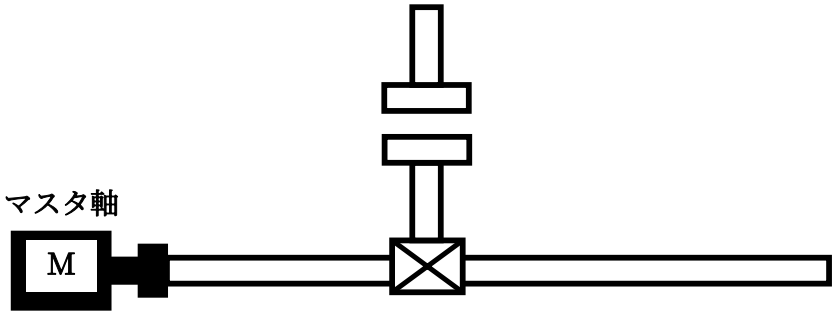
4. 動作例 ＜CAM動作：Demo#2 -1＞

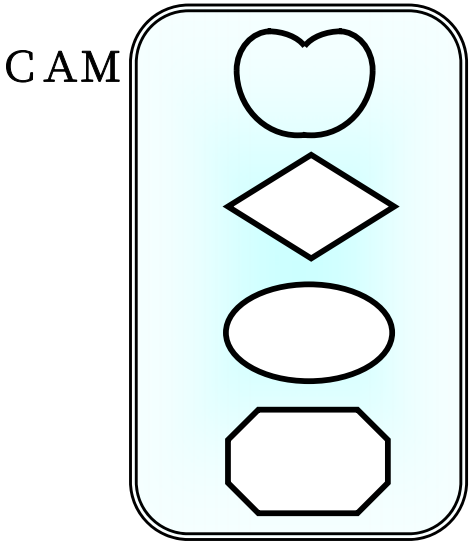
PLCopen
for efficiency in automation

機械式CAMによる運転パターンを、電気式のカム動作で実現

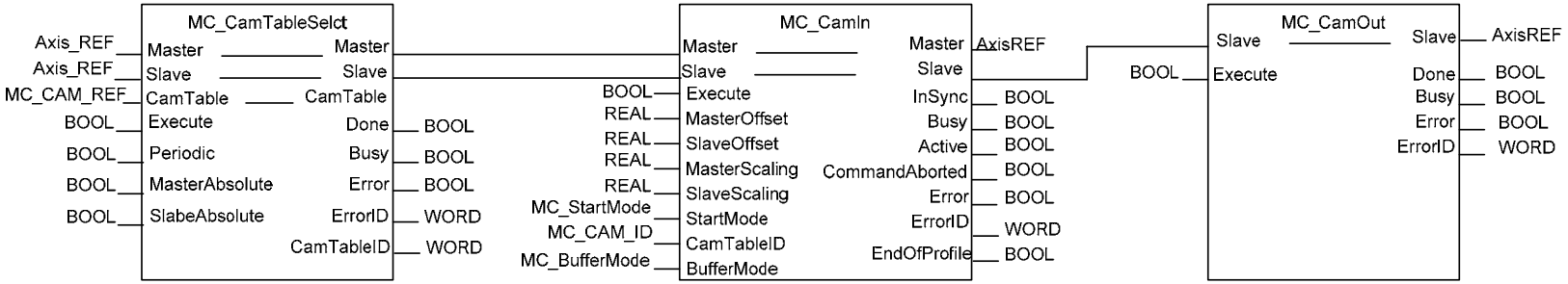
ビデオ クリップ

1）停止時





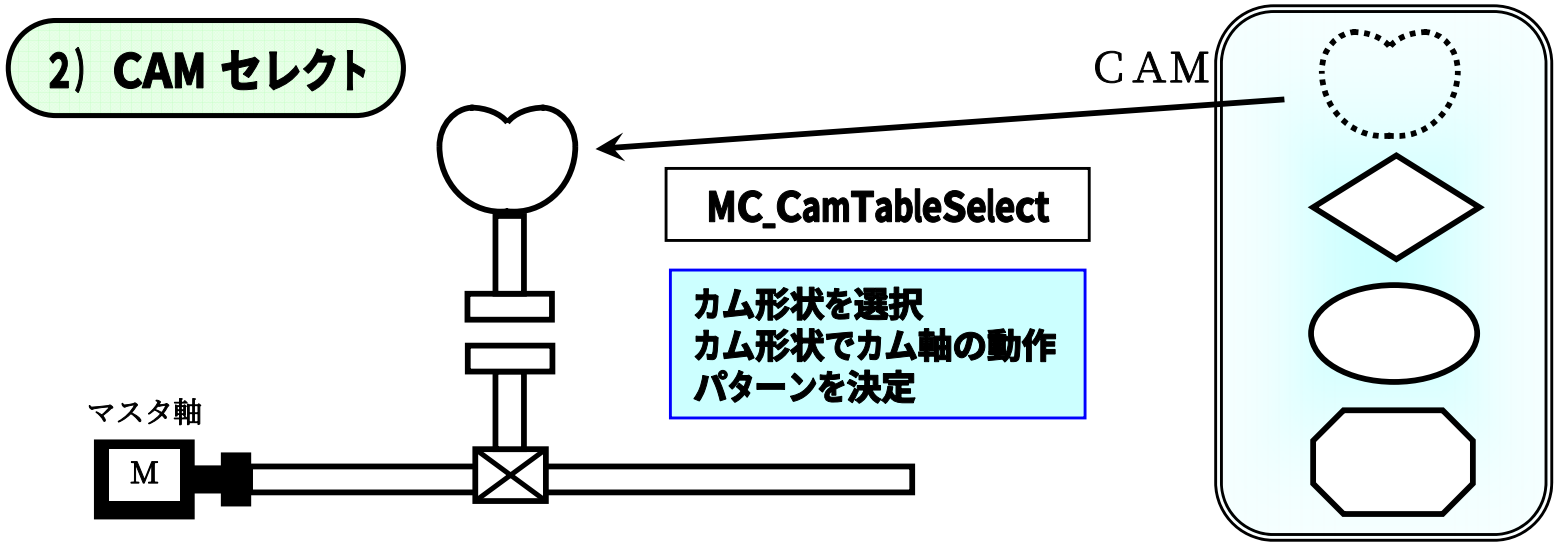
＜CAM動作の Program例＞



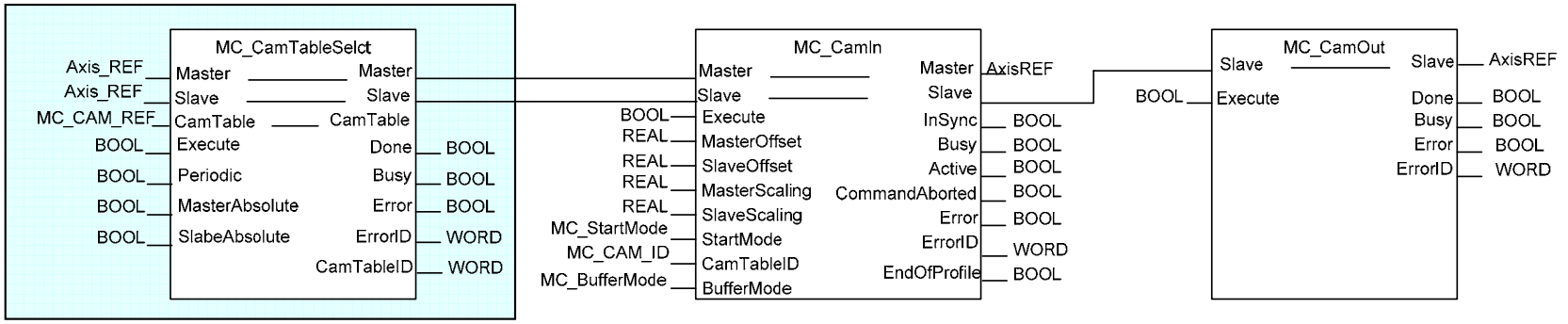
※ WriteParameter ⇒ CAMTableSelect ⇒ CANIN ⇒ CAMOUT

4. 動作例 ＜CAM動作：Demo#2-2＞

機械式CAMによる運転パターンを、電気式のカム動作で実現



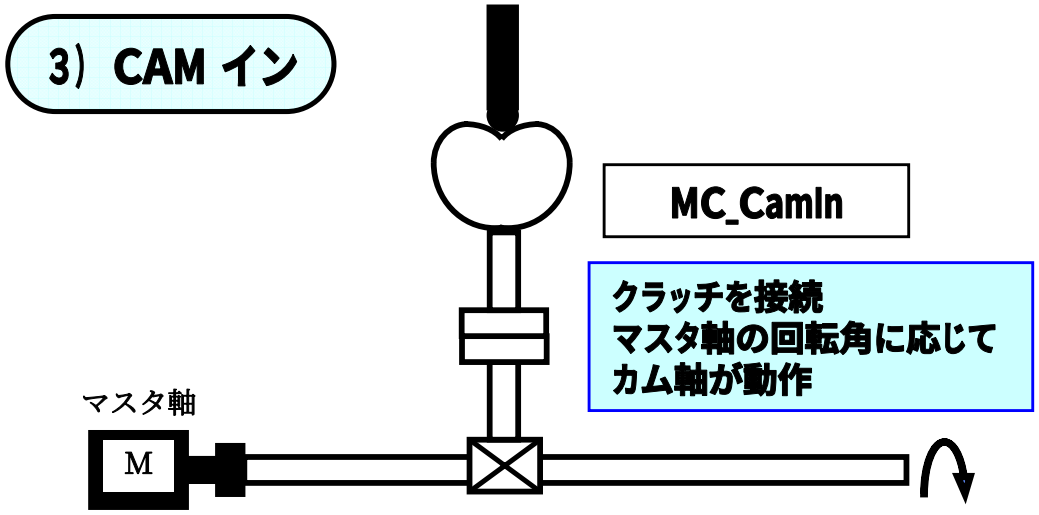
＜CAM動作の Program例＞

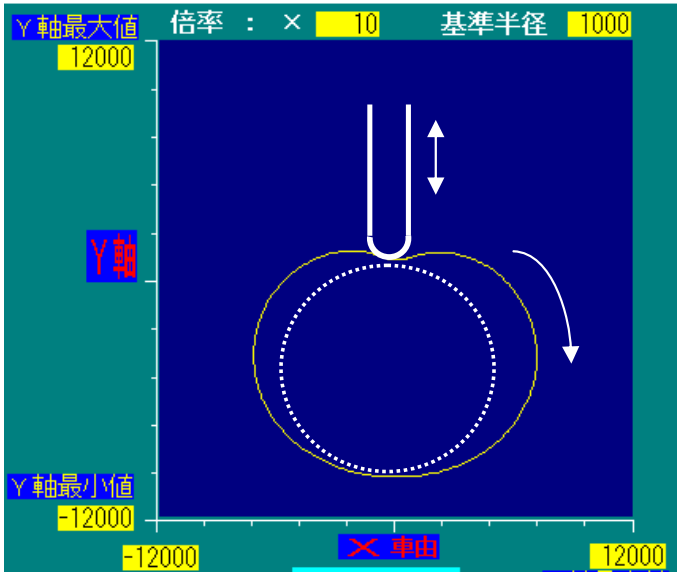


※ WriteParameter ⇒ CAMTableSelect ⇒ CANIN ⇒ CAMOUT

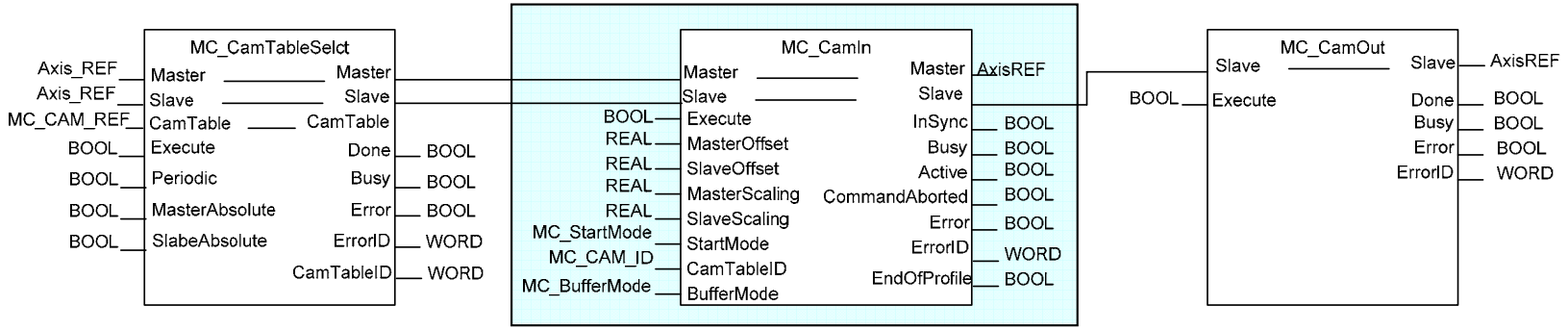
4. 動作例 ＜CAM動作：Demo#2-3＞

機械式CAMによる運転パターンを、電気式のカム動作で実現





＜CAM動作の Program例＞

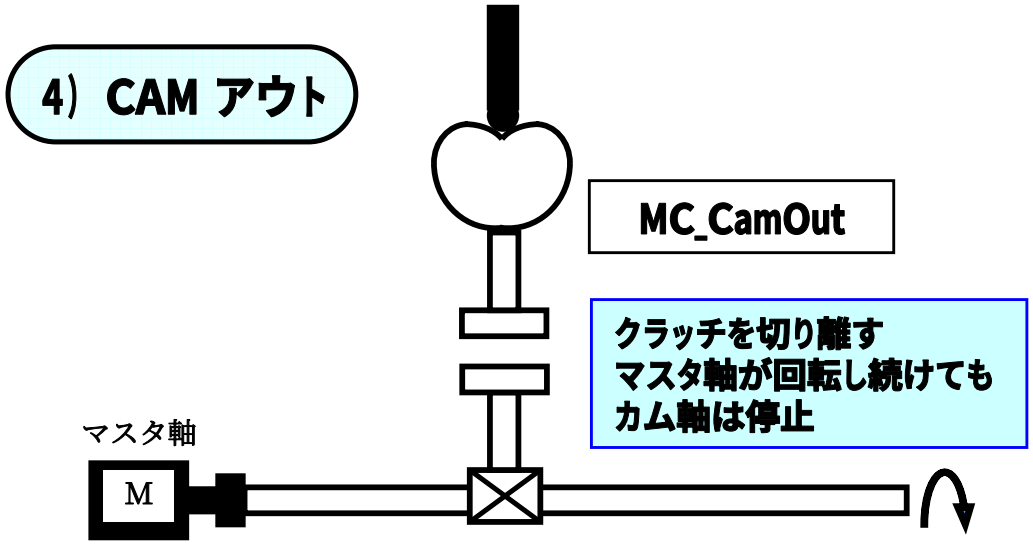


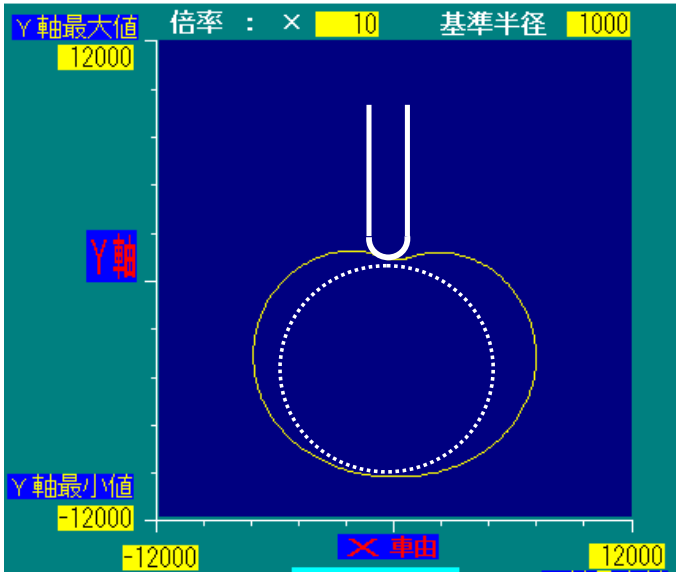
※ WriteParameter ⇒ CAMTableSelect ⇒ CANIN ⇒ CAMOUT

4．動作例 ＜CAM動作：Demo#2 -4＞

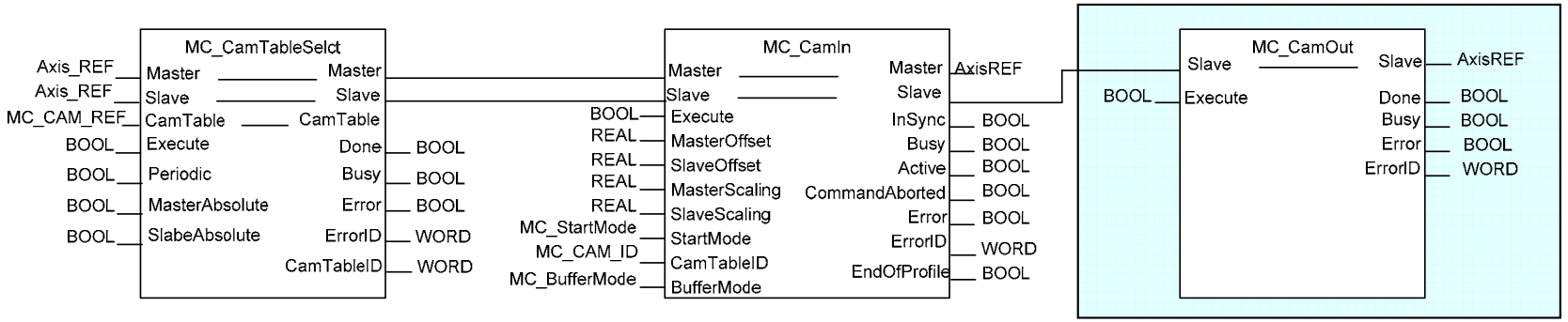
PLCopen
for efficiency in automation

機械式CAMによる運転パターンを、電気式のカム動作で実現





＜CAM動作の Program例＞



※ WriteParameter ⇒ CAMTableSelect ⇒ CANIN ⇒ CAMOUT

4. 適用可能なアプリケーションと動作例 (2)

拡張仕様の追加により、適用可能なアプリケーション

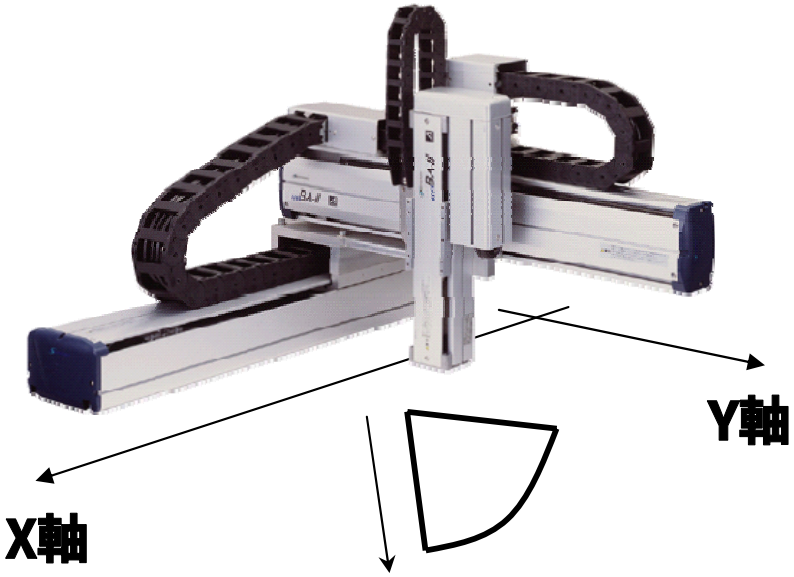
[多軸協調動作 (補間機能) の仕様 (パート)]
☆ Part 4 : 多軸仕様 ⇒ 多軸間で協調する動作 (補間) を規定

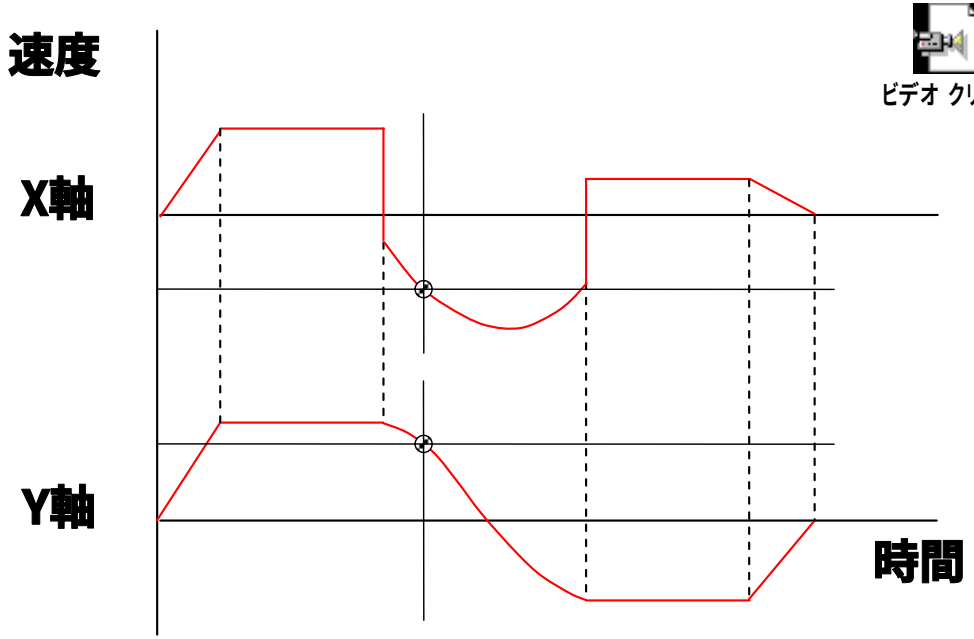
[基本仕様だけでは適用が困難なアプリケーション
(多軸で自在な協調動作など) に適用範囲を拡大]
☆ 2軸以上で、軸間の協調を取りながら任意の軌跡を実現
＜補間動作＞
☆軸座標系～機械座標系～製品座標系と、座標変換を規定
☆マスタ軸を必要としない多軸動作
☆軸グループ間での追従機能であるトラッキング動作

4. 動作例 ＜補間動作：Demo#3＞

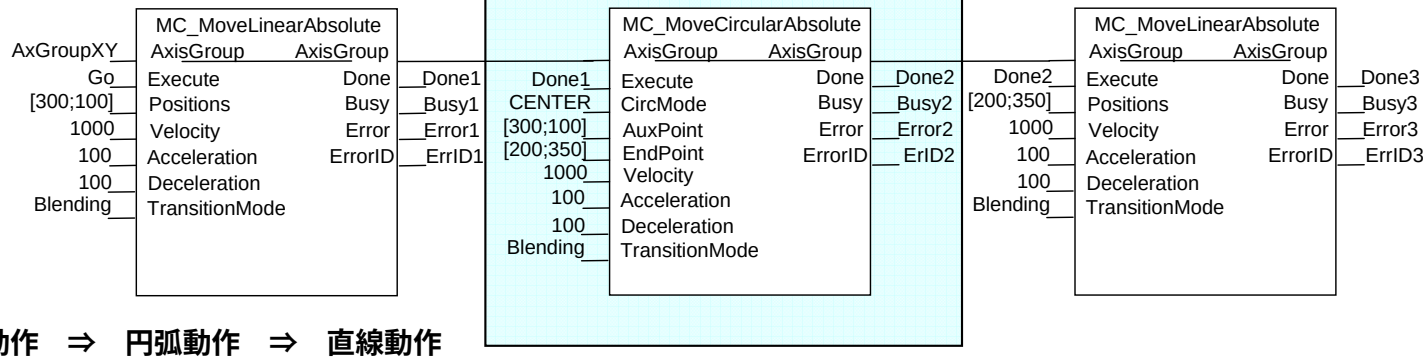
PLCopen
for efficiency in automation

マスタースレーブの関係でなく、複数軸で関連しあった補間動作を実現





＜補間動作の Program例＞



※　直線動作　⇒　円弧動作　⇒　直線動作

5．今後の取り組み

Safety，Logicとの融合と、適用可能なアプリケーションの更なる拡大

[検討中の仕様（パート）]

☆ Part3：ユーザガイトライン
Logic，Motion，Safetyを融合した仕様もガイドラインとして準備

☆ Part6： Fluid Power の仕様
Fluid Power 関係を、Motionと同じ仕様で規定することを検討

[PLCopen Japan としての取り組み]

☆Part4およびPart5の翻訳版を公開

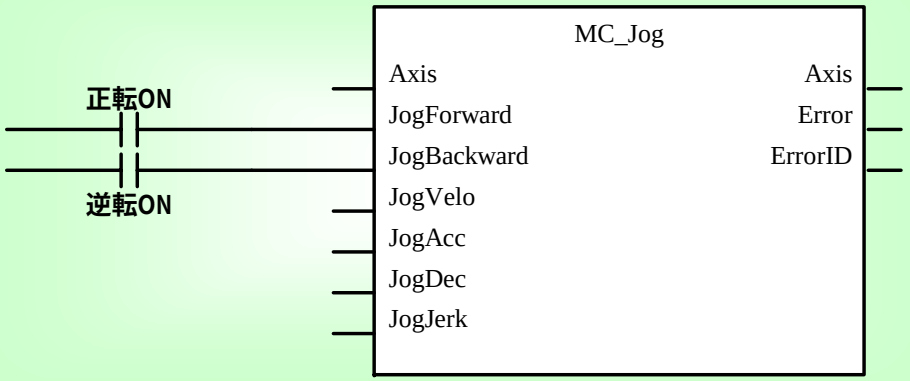
☆Part3および他のSafety FBなどとの融合を図りながら、
ユーザの使用を考慮したガイドラインを準備

☆ユーザの使用する観点からみた、実装仕様の妥当性を検証

☆ 国内での適用拡大を目指した活動を継続

5．実装仕様の検証 ＜JOG動作の例＞

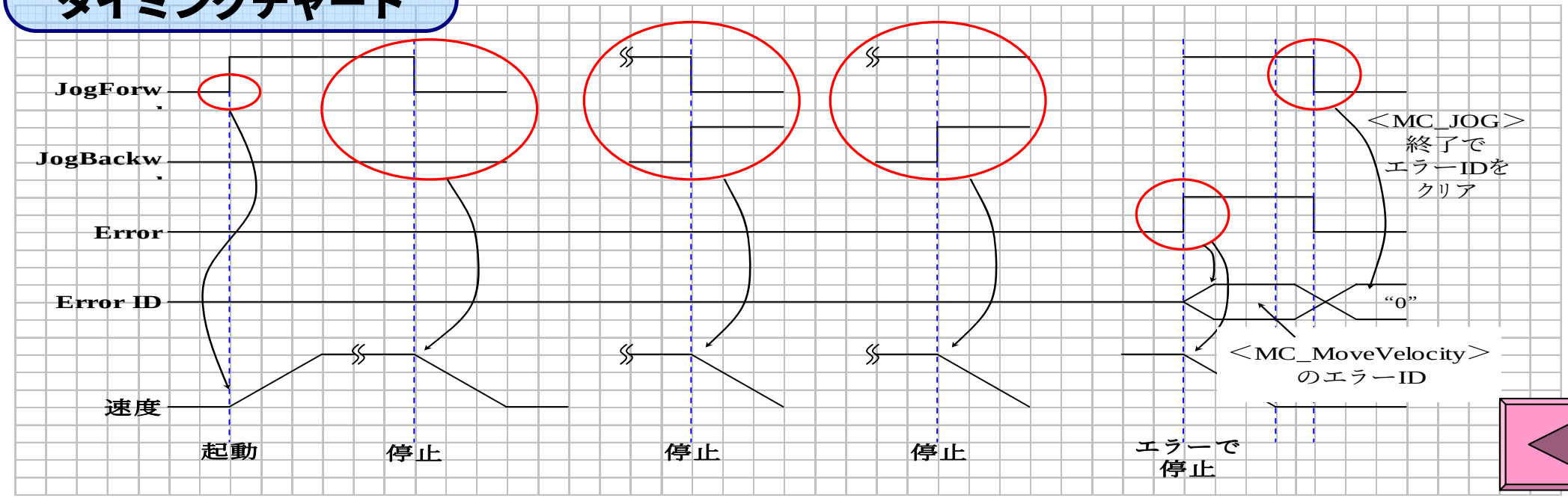
ラダー例（FBを使用した例）



【記述】
『MC_MoveVelocity』と『MC_Stop』の2つの規定されたFBを使用して実現

【条件】
起動条件：JogForwardかJogBackward　何れかの信号がON
停止条件：ONした信号がOFF，
もしくは反対側の信号がON

タイミングチャート



5．実装仕様の検証　＜JOG動作の例＞

プログラム例

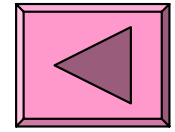
＜MC_JOG＞を規定

```
FUNCTION_BLOCK MC_Jog

        VAR_IN_OUT
                    Axis                        : AXIS_REF;
        END_VAR
        VAR_INPUT
                    JogForward    : BOOL;
                    JogBackward   : BOOL;
                    JogVelo                     : LREAL;
                    JogAcc                      : LREAL;
                    JogDec                      : LREAL;

                JogJerk                     : LREAL;
END_VAR
VAR_OUTPUT
                Error                       : BOOL;
                ErrorId       : UDINT;
END_VAR
VAR
fbStop                      : MC_Stop;
fbMoveVelocity              : MC_MoveVelocity;
state                       : INT;
MC_Positive_Direction       : MC_Direction := 0;
MC_Negative_Direction       : MC_Direction := 1;
END_VAR
```

```
CASE state OF
0: (* wait for start conditions *)
IF JogForward AND NOT JogBackward
THEN
                                                    state := 10;
END_IF
IF JogBackward AND NOT JogForward
THEN
                                                    state := 20;
END_IF
IF JogForward AND JogBackward
THEN
                                                    state := 32;
END_IF
```

正転JOGを起動

```
10: (* move forwards *)
fbMoveVelocity.Execute                  := TRUE;
fbMoveVelocity.Direction   := MC_Positive_Direction;
state := 11;
```



ご静聴ありがとうございました.