

# #変数vsアドレス

Variable  
Programming

×

Address  
Programming

第21回FA設備技術勉強会  
2025/02/01

yuyu (@yuyuTds)

# 自己紹介

- 名前 : yuyu (@yuyuTds)
- 仕事 : PLCメーカーのFAE (SE)
- FA歴 : 3年 (25歳)
- ひとこと : 3回目の発表になります  
#FA\_Study、#変数vsアドレス  
でコメントお待ちしております！



← 過去資料はこちら

# 今回は・・・

- 「PLCのプログラミング方法」について発表させていただきます。
- 「変数」と「アドレス」について、比較をしていきますが、どちらかをオススメするという発表ではないです。
  - 「変数」と「アドレス」については、次のスライドで詳しく説明します。
  - 自分が「変数」ばかり使ってるので、「変数」をオススメしているように聞こえたらすみません。。
- 「#変数vsアドレス」というタグにてポストいただいた内容を引用させていただきます。

みなさんのご意見を教えてください！



# 本編スタート！

# PLCのプログラミング方法

## ■ アドレスプログラミング

□ メモリアドレス（デバイス番号）を直接指定してプログラミングする方式。

- 例えば、入力X000、出力Y010、内部リレーM100など

□ 三菱（FX、IQ-R）、キーエンス（KV）、オムロン（CJ、CS）が採用

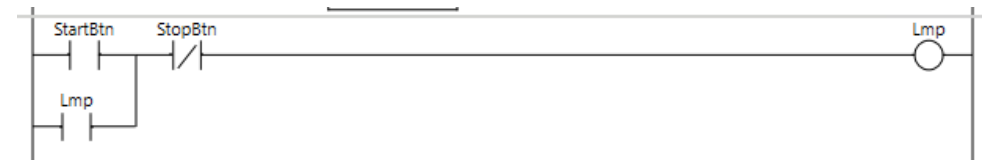


## ■ 変数プログラミング

□ シンボル（変数名）を使ってプログラムを書く方式。

- データ型（整数、文字列など）を指定して、変数を作成する（メモリには自動割り付け）
- 例えば、「startButton」、「servoON」など

□ オムロン（NJ/NX）、Beckhoff（TwinCAT）、CODESYSが採用



## ■ （タグプログラミング）

□ 今回の発表では対象外

# PLCのプログラミング方法


## ■（参考）IEC61131-3 ≡ 変数プログラミング

□PLCのプログラミング言語についての国際規格として「IEC61131-3」があります。

- IEC61131-3では、プログラミング言語（5種類）、FUN・FB、データ型、タスクとスキャンサイクルを定義している。
- ただ、IEC61131-3準拠といっても、準拠具合にバラツキがある。  
（タグプログラミングでもIEC61131-3準拠になる）

## □IEC61131-3のエディション

- IEC61131-3 Ed.2：変数プログラミング、タスクという考えを取り入れた。
- IEC61131-3 Ed.3：オブジェクト指向を追加。
- IEC61131-3 Ed.4：最近出た。詳細不明。TwinCATのPLC++がこれ？



ここから  
変数とアドレスの  
いいところを  
ひたすら紹介します

# 変数のいいところ

## ■名前に意味を持たせることができる

□アドレスを覚えなくてよい。

- プログラム作成時、デバック時に「あれってMのいくつだったっけ」ってことがなくなる。

□コメントがなくても、ラダーだけでもある程度、何をやってるか分かる。



# アドレスのいいところ

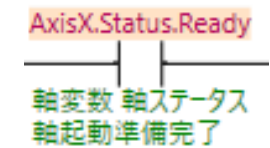
## ■変数名を考えなくて良い、短い

### □変数名を考えなくて良いので、スラスラとプログラムが作れる

- 変数は毎回変数を定義する必要がある。
- （ただ、アドレスマップは必要だけど。。）

### □デバイス名が短いのでラダープログラムが見やすい

- 変数プログラミングだと、20文字以上になってしまうことも。。



### □変数名の管理が不要（ルール作りが不要）

- 変数名のルールを作らないと、無法地帯になり、可読性が格段に落ちる。
- アドレスだと、すでにデバイス（変数）が用意されているので、強制的にルールに準拠したプログラムになる。

# 変数のいいところ

## ■追加、変更に強い

### □機能を追加してもプログラムを変更する必要がない

- アドレスをずらす必要がない。
- アドレスの「綺麗にアドレス割り振らなきゃ」「やってみたらアドレス足りなくて一部だけ違う」ということがなくなる。

### □ユニットの追加、削除しても問題ない

- ハード構成の変更が、プログラムに影響を与えない
- IDE上で、実IOと変数をマッピングする

# アドレスのいいところ

## ■プログラムの自由度が高い

### □連続処理がしやすい

- 例えば、HMIで表を表示する場合やIOの状態を表示する場合など
- 間接参照が強力！  
(変数プログラミングでもポインタに対応している場合は、同様のことができます)

### □データ型にとらわれない処理ができる

- （すみません、どんなときに使うのかが分かってないです。。）

# 変数のいいところ

## ■アドレス被りが発生しない

### □アドレス重複によるデバック時間の短縮

- インデックスレジスタより、どこで重複してバグっているのかがクロスリファレンスしてもわからないってことがある。

### □プログラムの再利用が安易（ライブラリ、OSS向け）

- ハードとの結びつきが小さいので、プログラムの再利用が安易

### □巨大なプロジェクトでも対応可能

- PLCの仮想化が進み、ラインを1つのプロジェクトで管理するとなった場合、アドレス管理が不可能になることが想定される。

# 変数のいいところ

## ■プログラムを階層構造にできる

### □可読性があがる

- 特に他人のプログラムを読むときは、アドレスによる巻物プログラムはどこから手をつけてよいかわからない。
- 変数プログラミングでは、他の人のプログラムでも、比較的、改造しやすい（気がする。。）

### □構造体を使用することで、変数をグループ分けすることができる

- 階層化されていることで、他の人が作ったプログラムでも目的の変数に、比較的簡単に辿り着ける

### □全ての変数にコメントをつけなくても良い

- L、Hといったコメントは必要なくなる。意味も変数名から推測できる。
- （変数名とコメントって同じようなことを指しているなので、二度手間ではというのもある。。なので、コメントは注釈的な使い方をしていることも）

名称	モニタ値	変更	コメント	データ型
▼ EIC01_In				EIC01_EIC01_In
▼ SlaveSts			スレーブターミナルステータス	EIC01_SlaveSts
▶ Reserved1[0-3]				
Obsr			監視情報	BOOL
MinorFault			軽度フォールト	BOOL
PartialFault			部分停止フォールト	BOOL
MajorFault			全停止フォールト	BOOL
▶ Reserved2[0-5]				
ErrorDetected			エラー検出フラグ	BOOL
IORefresh			I/Oリフレッシュフラグ	BOOL
▼ NXUnitSts			NXユニットステータス	EIC01_NXUnitSts
▶ RegistrationSts[0-15]				
▶ MessageEnablesSts[0-15]				
▶ IODataActiveSts[0-15]				
▶ ErrorSts[0-15]				
▼ ID			入力ユニット	EIC01_EIC01_ID
ResetPB			リセットボタン	BOOL
StartPB			スタートボタン	BOOL
StopPB			ストップボタン	BOOL
ClearPB			クリアボタン	BOOL
▶ Reserved[0-11]				
▼ AD			アナログ入力ユニット	EIC01_EIC01_AD
RoomTemperature			室温	UINT
SupplyAirPressure			供給エア圧	UINT
▼ EIC01_Out				EIC01_EIC01_Out
▼ OD			出力ユニット	EIC01_EIC01_OD
PatliteR			パトライト(赤)	BOOL
PatliteY			パトライト(黄)	BOOL
PatliteG			パトライト(緑)	BOOL
NotUse1			未使用	BOOL
▶ Reserved[0-11]				

# アドレスのいいところ

## ■既存資産がある

### □参考になるプログラムが転がっている

- 社内だけでなく、ネット上にもいろいろある。
- 社内の既存プログラムが流用できる

### □保全の方も使うことができる

### □フィールドネットワークは、アドレスを対象としたものが多い

- ModbusやEtherNet/IPなど、アドレスプログラミングと親和性が高いものが多い
- （一方でOPC-UAといったタグ通信に対応したものもでてきている。）

# 変数のいいところ

## ■ITとの親和性

### □上位アプリケーションとデータを意味が分かる形で通信できる

- 変数名によって、上位アプリケーションからデータの意味が分かる
- IT系のシステムとデータ型がほとんど同じであるため楽

### □若手との相性

- ITのプログラミングを学んだ学生には、変数のほうがとっつきやすい。
- 早期戦力化、人材の確保につながる

# ほかにも

- アドレス：電気図面から直感的に(ラダーとかを)追える。
- アドレス：エディタが使いやすい（変数のエディタが成熟していない。。）
- 変数：アドレスでできたことが、変数でまだできるようになっていない。（IDEの完成度が低い）
- 変数：マニュアルを見なくても、変数名だけでサーボのパラメータが設定できる。



# 最後に

## ■（個人的な）まとめ

□変数：面倒、長期的にはメリットがある。

□アドレス：楽。

## ■発表資料

□GitHubにアップロードします（間に合わなかった）



## ■お願い

□「#変数vsアドレス」というタグでポストいただけると嬉しいです！

ありがとうございました！ Thank you !