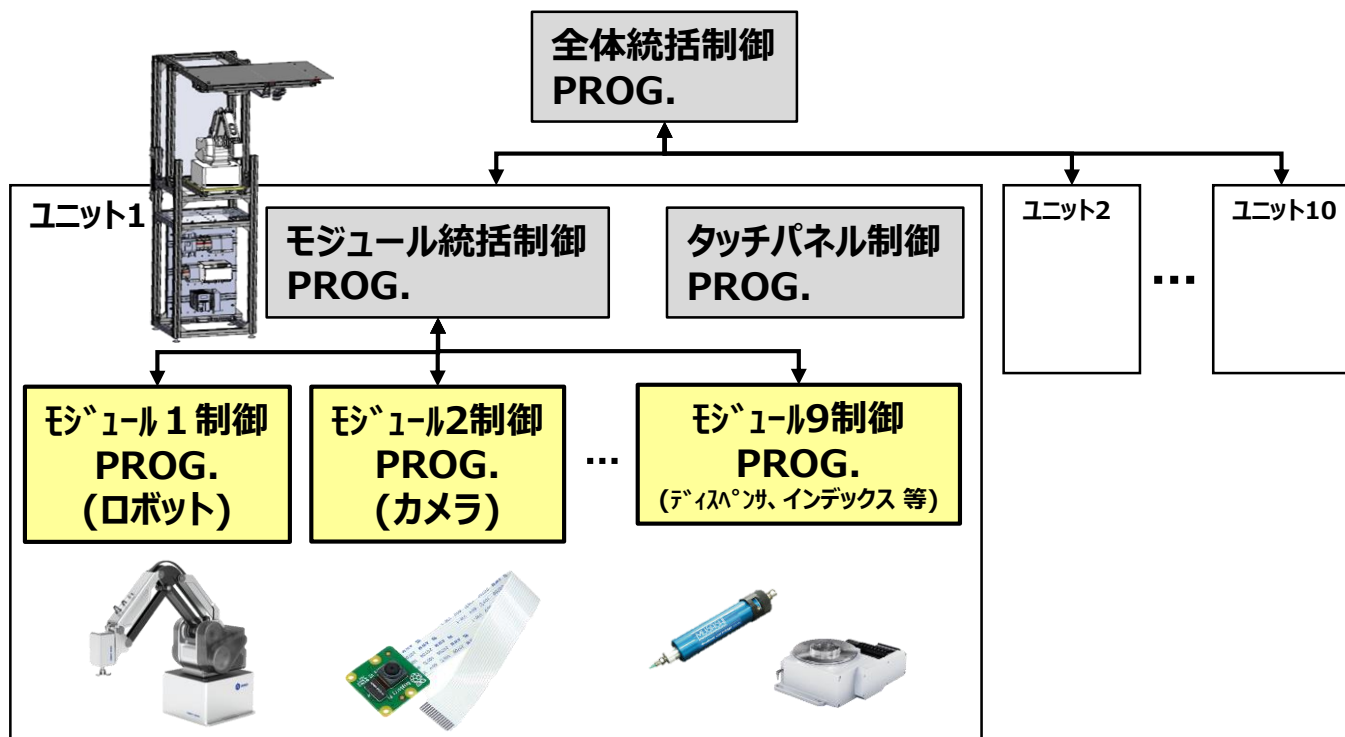


- 全体構成
  - ・ 制御構成
  - ・ デバイス使用領域
- プログラム説明
  - ・ プログラム詳細構成
  - ・ シーケンス記載ルール
- プログラム編集方法
  - ・ ユニット、モジュール追加
  - ・ IO制御
  - ・ ロボット動作
  - ・ カメラ動作
  - ・ サイクルタイム測定
  - ・ ワーク在荷
  - ・ エラー
  - ・ 干渉領域
  - ・ プログラム間通信
  - ・ サイクル停止

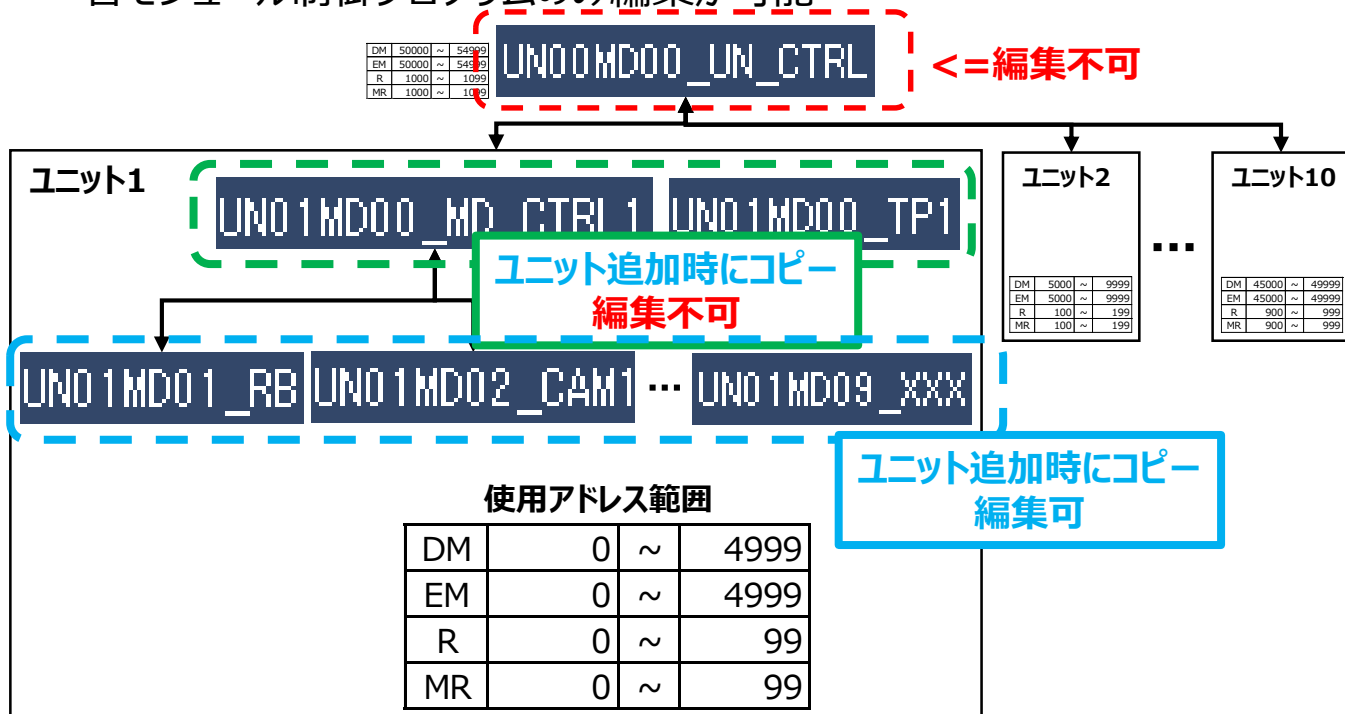
### ・制御構成

- 最大、ユニット10台 + タッチパネル10台を同時に制御可能
- 1ユニット当たり最大1台のロボット、9台のモジュールを制御可能



### ・プログラム構成

- 各モジュール制御プログラムのみ編集が可能





デバイス使用領域

・下記に示す領域のアドレスは既に使用されている  
(※アドレスの詳細は、“アドレスマップ.xlsx”を参照)

	UN1			UN2			UN3		
DM	0	～	4999	5000	～	9999	10000	～	14999
EM	0	～	4999	5000	～	9999	10000	～	14999
MR	0	～	99	100	～	199	200	～	299
R	0	～	99	100	～	199	200	～	299
@R	0	～	29	0	～	29	0	～	29
	UN4			UN5			UN6		
DM	15000	～	19999	20000	～	24999	25000	～	29999
EM	15000	～	19999	20000	～	24999	25000	～	29999
MR	300	～	399	400	～	499	500	～	599
R	300	～	399	400	～	499	500	～	599
@R	0	～	29	0	～	29	0	～	29
	UN7			UN8			UN9		
DM	30000	～	34999	35000	～	39999	40000	～	44999
EM	30000	～	34999	35000	～	39999	40000	～	44999
MR	600	～	699	700	～	799	800	～	899
R	600	～	699	700	～	799	800	～	899
@R	0	～	29	0	～	29	0	～	29
	UN10			UNコントローラ			共通		
DM	45000	～	49999	50000	～	54999	55000	～	60000
EM	45000	～	49999	50000	～	54999	55000	～	60000
MR	900	～	999	1000	～	1099	1100	～	1400
R	900	～	999	1000	～	1099	1100	～	1400
@R	0	～	29		～			～	

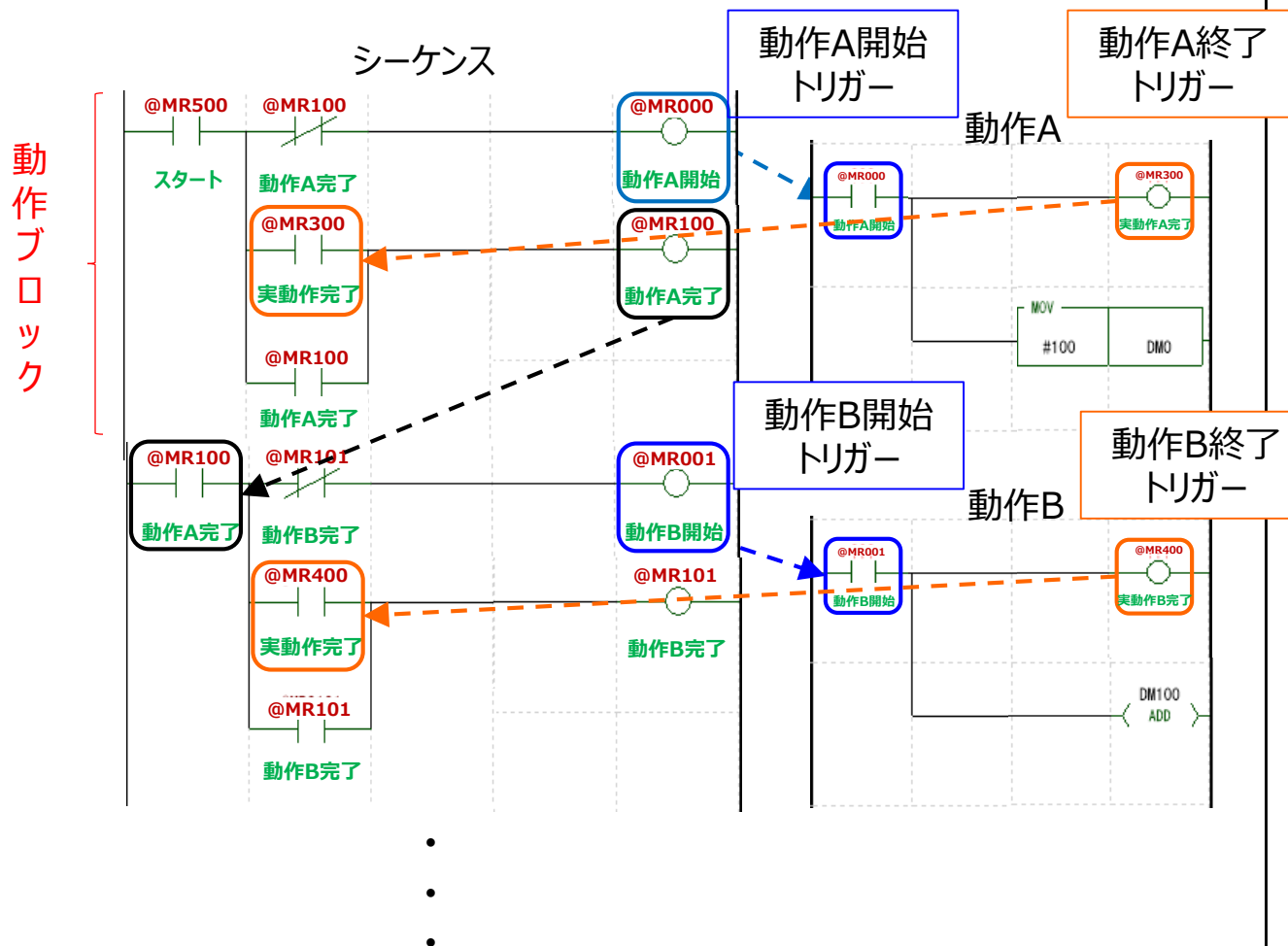
### ・プログラム詳細構成

- プログラムは、処理内容の異なる7パートで構成される

1. Setting : モジュール単位のパラメータを設定
2. Input : グローバル変数をローカル変数に格納
3. Pre Process : サイクル前の処理
4. Cycle : サイクル動作
5. Post Process : サイクル後の処理
6. Function : ロボット・カメラ動作設定
7. Output : ローカル変数をグローバル変数に格納

### ・シーケンス記載方法

- 動作ブロックを組み合わせることでシーケンスを作成する

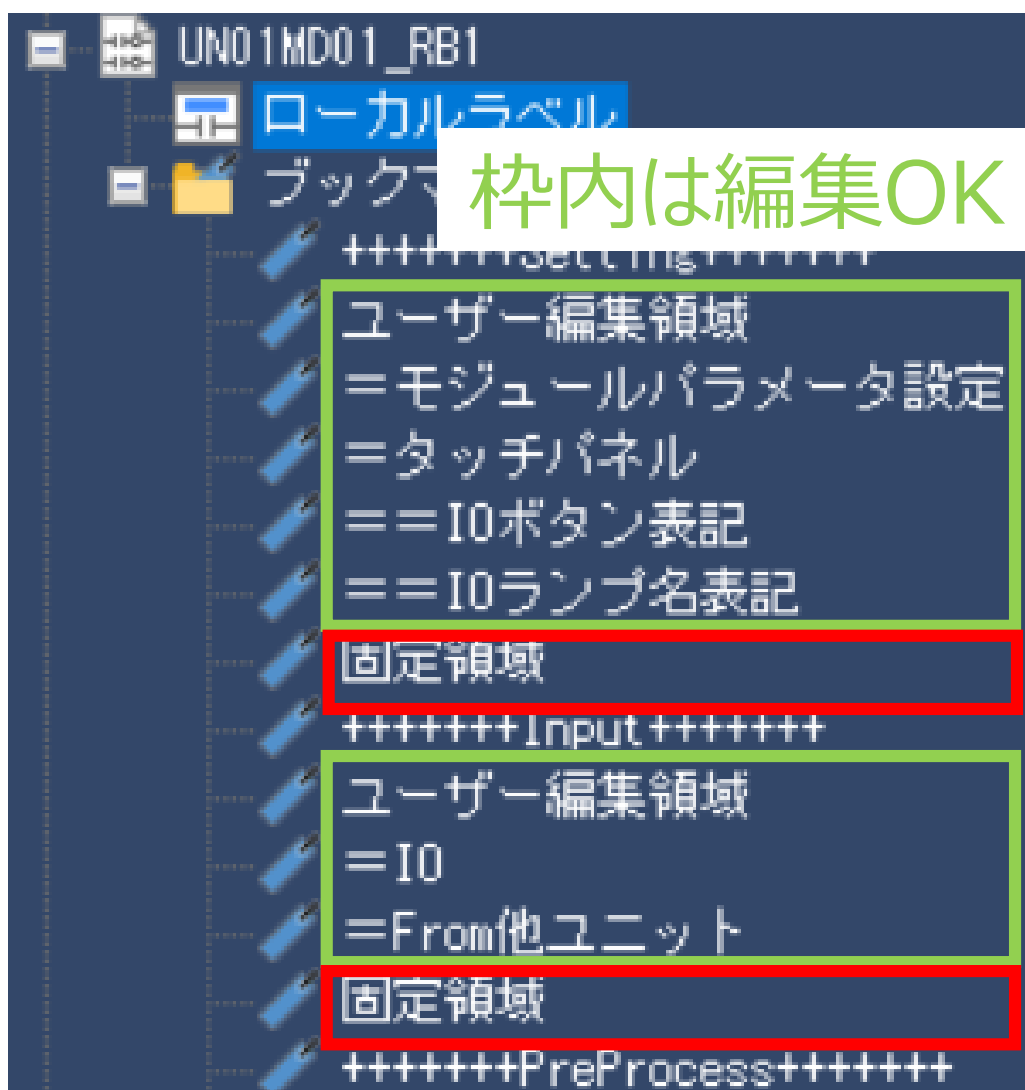


### ・プログラムの編集

- 各パートにて、“ユーザー編集領域”と“固定領域”が設定されている

ユーザー編集領域：以降のラダーは編集可能

固定領域：以降のラダーは編集不可能



- 変数名の追加は自由に可能だが、“D\_”が付与された変数は、編集不可



### ・概要

ユニットとモジュールの台数を変更する。

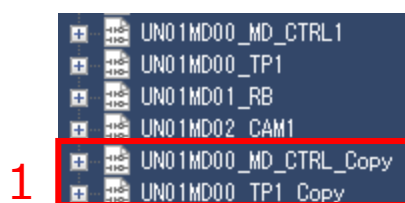
### ・ユニット台数追加方法

1. “CTRL”と“TP”のプログラムをコピー＆ペースト

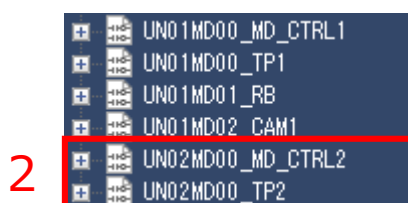
2. プログラム名を変更

推奨：UN番号MD番号\_モジュール名

※番号はプログラムに影響しない、“CTRL”がUN番号を自動で設定



※必ずCTRL、TPの順番



### ・モジュール台数変更方法

1. “RB”や“CAM”モジュールをコピー＆ペースト

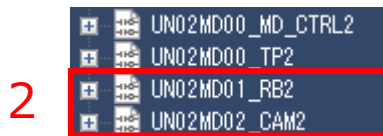
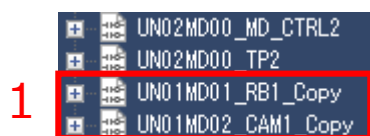
2. プログラム名を変更

3. “Setting”ブックマーク内の“MD\_No”と“RB\_No”もしくは“CAM\_No”の値を設定

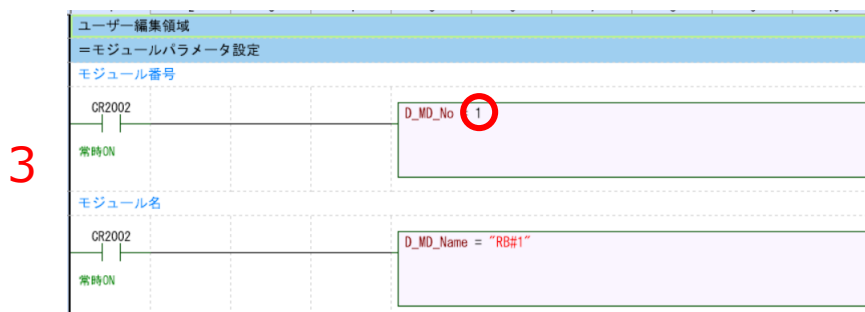
MD\_No: ユニット内でのモジュール番号 ※UN内での重複不可

RB\_No: タッチパネルの“RasPi”画面に表記されるロボット番号

CAM\_No: タッチパネルの“RasPi”画面に表記されるカメラ番号



※必ずTPの後に、モジュールを追加する



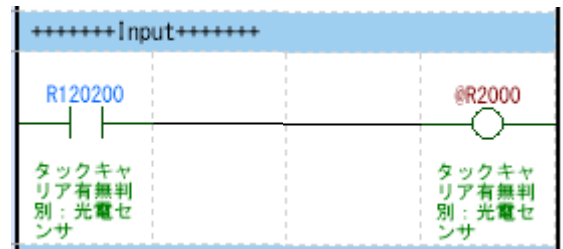
### ・ IO入力設定手順

1. 実際、PLCに接続されているIOユニットをラダーと合わせる(予約を解除)
2. 任意のローカル変数に格納する ※推奨:@R3000、@LR000以降

#### 1. 共通UNとUN1用のIOユニットを使用する場合

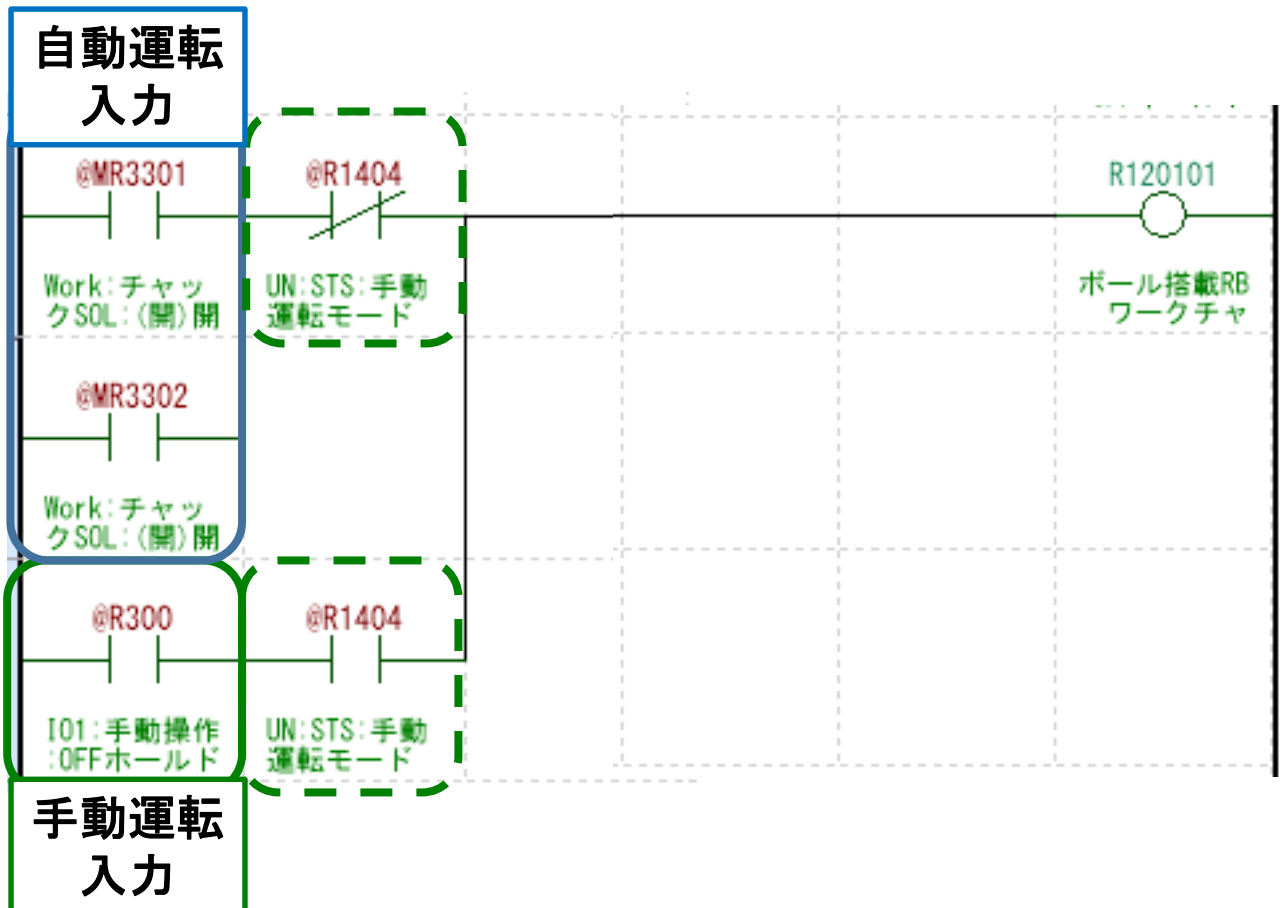
ユニット構成			
[0]	KV-7500		
[1]	KV-SIR32XT	R110000/R110200	----- [予約]
[2]	KV-C16XTD	R111000/R111100	-----
[3]	KV-C16XTD	R120000/R120100	-----
[4]	KV-C16XTD	R120200/R120300	----- [予約]
[5]	KV-C16XTD	R120400/R120500	----- [予約]
[6]	KV-C16XTD	R120600/R120700	----- [予約]
[7]	KV-C16XTD	R120800/R120900	----- [予約]
[8]	KV-C16XTD	R121000/R121100	----- [予約]
[9]	KV-C16XTD	R121200/R121300	----- [予約]
[10]	KV-C16XTD	R121400/R121500	----- [予約]
[11]	KV-C16XTD	R121600/R121700	----- [予約]
[12]	KV-C16XTD	R121800/R121900	----- [予約]

← 共通UN用  
← UN1用  
← UN2用  
← UN3用  
← UN4用  
← UN5用  
← UN6用  
← UN7用  
← UN8用  
← UN9用  
← UN10用



### ・ IO出力手順

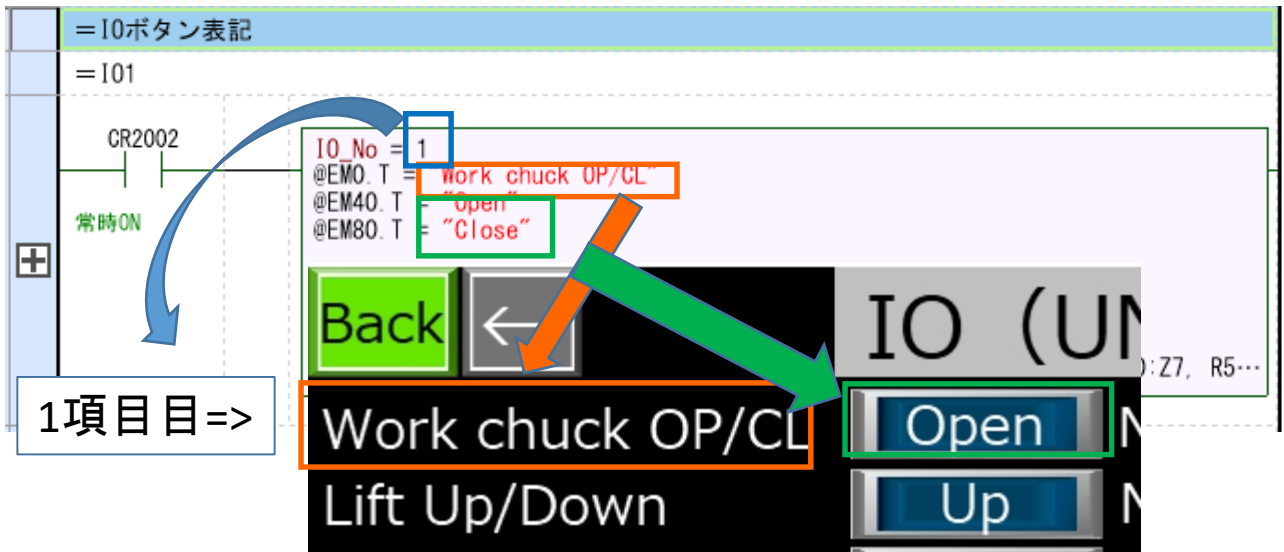
1. 出力操作のローカル接点を出力用グローバル変数に格納  
自動・手動のそれぞれ入力に手動運転モードのb接点・a接点を接続し、個別に出力動作するようにする



### ・タッチパネル表記手順

1. 項目ごとに下記のラダーをコピー。

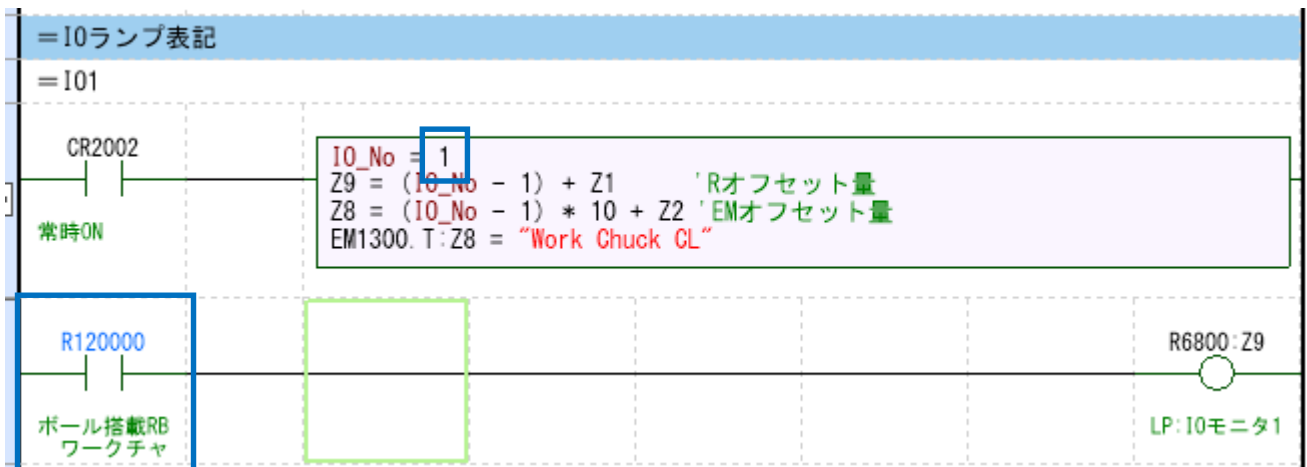
表示項目数 / 表示項目 / 操作時の表記内容を記入



### ・ランプ機能

1. 項目ごとに下記のラダーをコピーし、ランプ表示を実施する項目数を入力

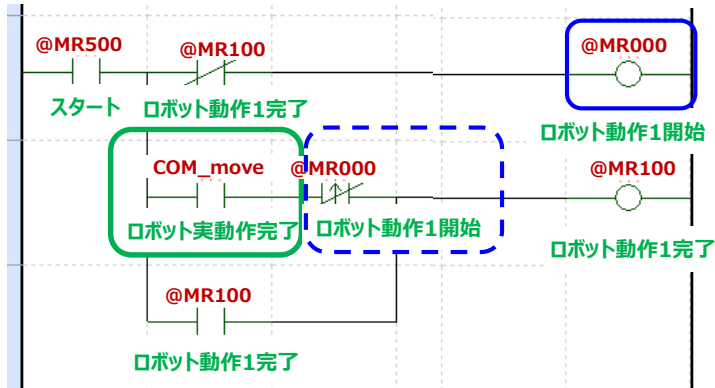
2. ランプ点灯時の動作を検出するセンサ入力を配置





## ロボット動作

- ロボットの1動作フローは動作開始は目標位置/トリガに接続  
実動作完了はローカルデバイスCMP\_moveを配置。  
CMP\_moveの右側にロボット動作開始のB接点を配置  
(ロボット連続動作時の安定のため)



- 目標位置では下記のスクリプトを接続し、ロボット動作に必要な各種パラメータを設定。

目標位置・速度・加速度・位置決め幅・静定時間をティーチング番号を設定することで設定可能。(FCP専用設定)

@MR000

↑

ロボット動作1開始

```
##### FCP専用 #####
' ティーチング番号
teach_No = 1
' ティーチングデータ取得
FUN("getTarget", teach_No, target_pos_x_buf, target_pos_y_buf, target_pos_z...
#####
' Job番号
job_No = 1
' 移動有効軸
use_x = TRUE
use_y = TRUE
use_z = TRUE
use_r = TRUE
' 目標位置
target_pos_x = target_pos_x_buf
target_pos_y = target_pos_y_buf
target_pos_z = target_pos_z_buf
target_pos_r = target_pos_r_buf
' 目標速度・加減速度
target_vel = target_vel_buf
target_acc = target_acc_buf
target_dec = target_dec_buf
' 目標位置決めの完了幅・静定時間
target_static_dist = target_static_dist_buf
target_static_time = target_static_time_buf
' 目標位置オフセット量
offset_x = 0.0
offset_y = 0.0
offset_z = 0.0
offset_r = 0.0
' オーバーライド有無
override_use = TRUE
FUN("setTarget", use_x, current_pos_x, target_pos_x, offset_x, use_y, curre...
```

<=移動軸をTRUE / 非稼働軸 FALSEに設定

<=目標位置を設定

<=速度・加速度を設定

<=位置決め幅・静定時間を設定

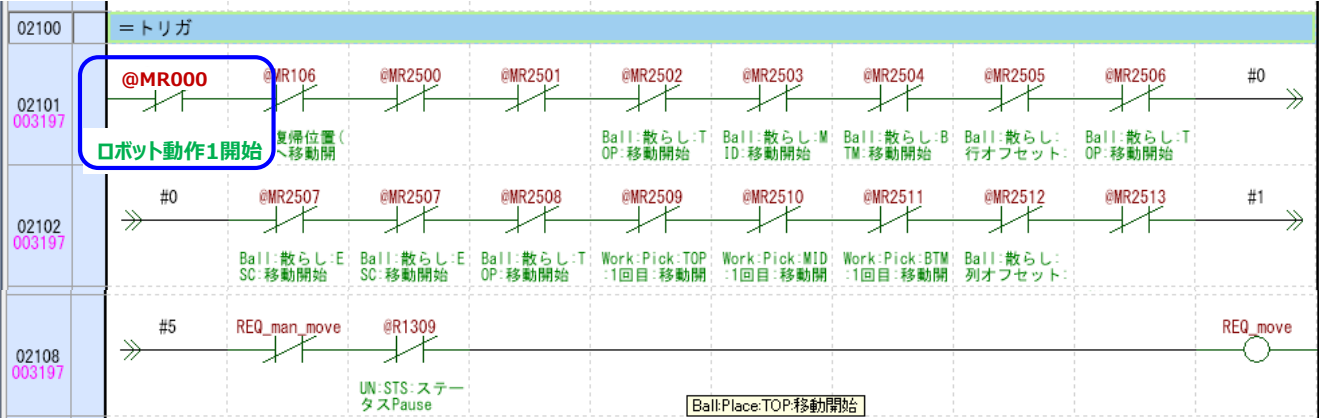
<=各軸のオフセット量を設定

<=オーバーライドの有無を設定



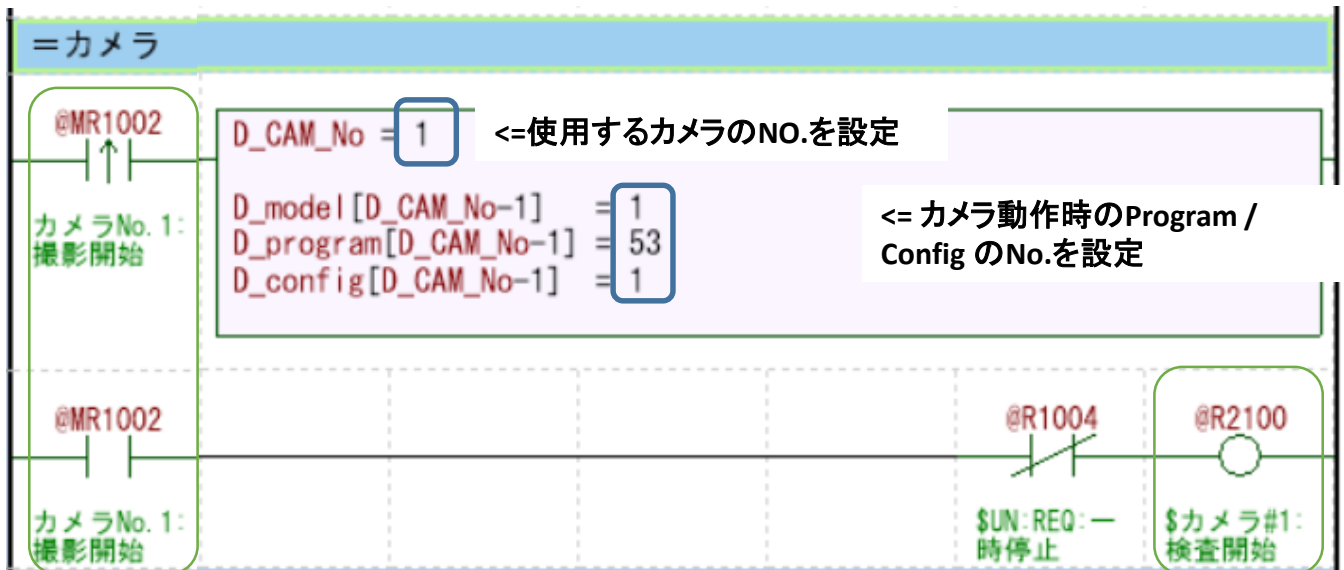
ロボット動作

・トリガに動作開始を追加



### ・カメラ動作トリガー送信

使用するカメラNo. program / Config No. / Model No.を設定。

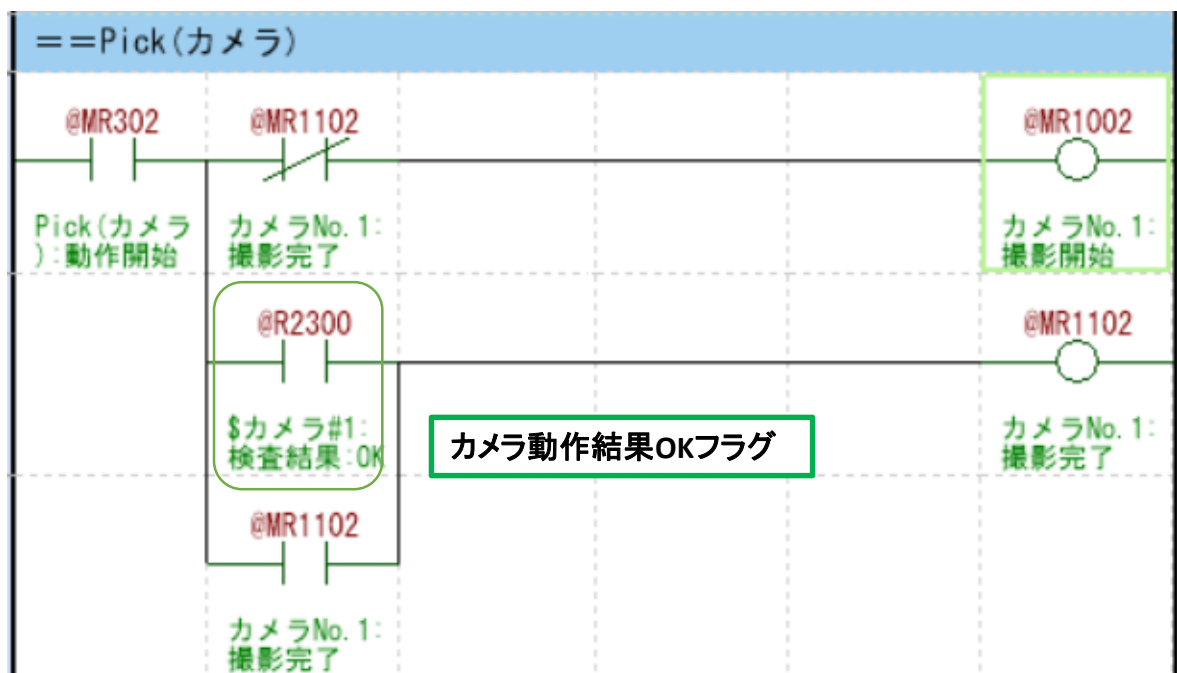


カメラ動作開始信号  
(シーケンスからのトリガー)

動作させる“カメラ  
動作フラグ”を設定

### ・カメラ動作トリガー受信

シーケンスに、カメラ動作結果OKフラグを設定



### ・カメラ動作結果参照

D\_alignment配列に結果が格納されている

※軸×カメラ番号の二次元配列

目標位置オフセット量

```
D_CAM_No = 1
D_offset_x = D_alignment[X_AXIS, D_CAM_No-1]
D_offset_y = D_alignment[Y_AXIS, D_CAM_No-1]
D_offset_z = 0.0
D_offset_rz = D_alignment[R_AXIS, D_CAM_No-1]
D_offset_ry = 0.0 6軸ロボット専用
D_offset_rx = 0.0 6軸ロボット専用
```

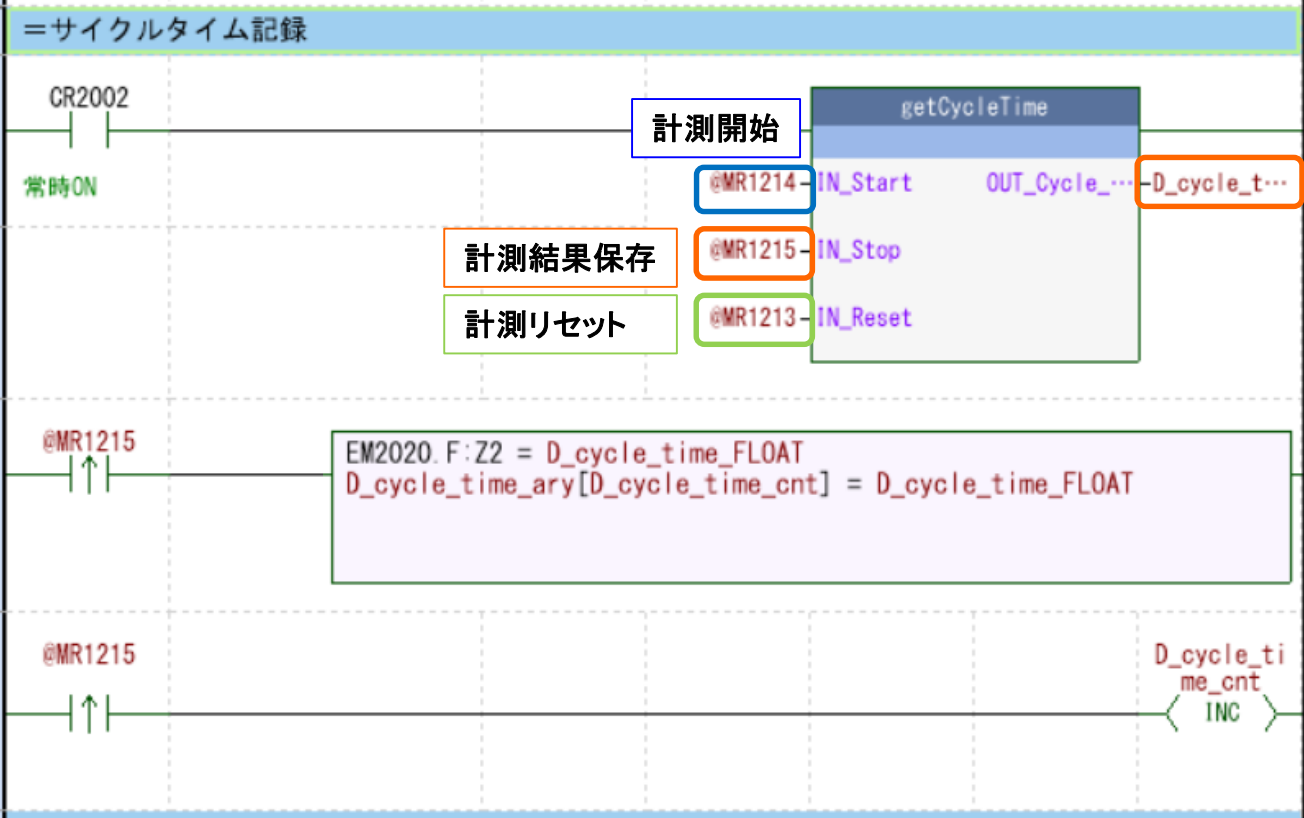
軸番号

カメラ番号  
※0から



# サイクルタイム測定

・サイクルタイム測定を実施するプロセスの開始・完了のデバイスを配置。



- ・作業場所・サンプルの種類・ハンドリング条件を設定することで、ハンドリング可否・ハンドリング時のトレイ位置のオフセット量・トレイ上の位置の確認可。
- 1.パラメータを設定(設定の際のmatrixは下記の様に設定)
- 2.サンプル在庫 or 排出スペースがある際のローカルデバイス設定

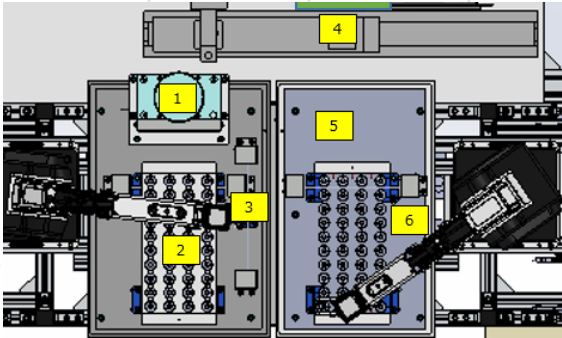
CR2002

常時ON

**1.パラメータ設定**

- ・Counter No. : 作業対応場所
- ・Work No. : サンプル種類
- ・Handling mode (Pick=1, Place=2, Assembly=3)

・UN\_No : 作業対応場所に対する番号 (固定条件)



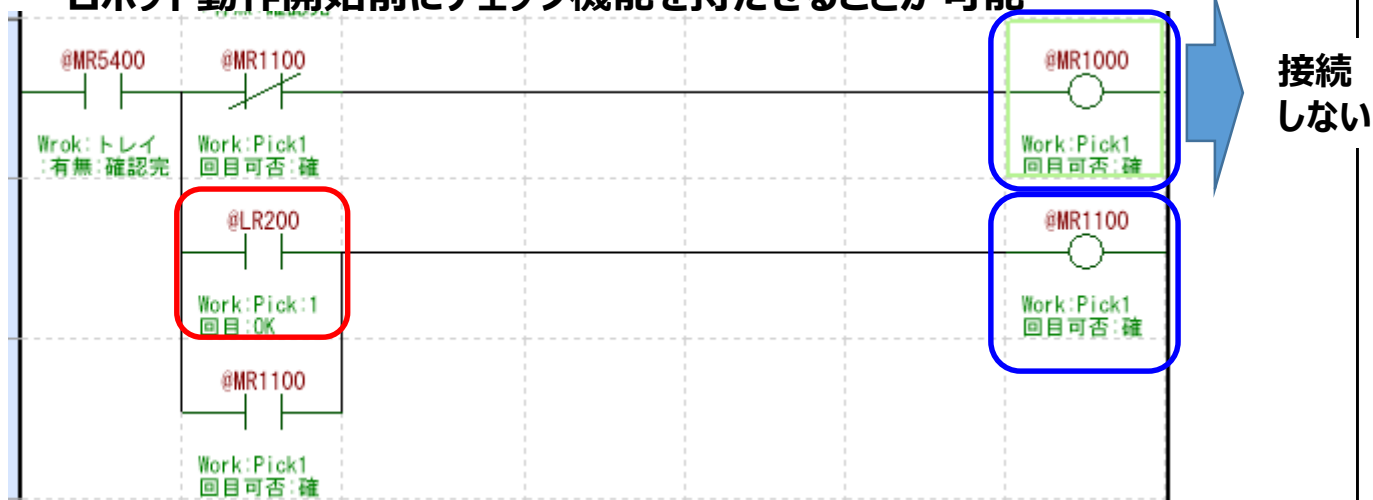
2.ローカルデバイスの設定

checkWork0

#2	IN_Counter...	OUT_Off...	@LR200 Work:Pick:1回...
#2	IN_Work_No	OUT_Tray_0...	tray_2_of...
#1	IN_Handlin...	OUT_Tray_0...	tray_2_of...
	1:Pick, 2:Plac...		
UN_No	IN_Counter...	OUT_Tray_0...	
	ワーク置き場1...		
UN_No	IN_Counter...	OUT_Target...	
	ワーク置き場2...		
UN_No	IN_Counter...		
	ワーク置き場3...		
UN_No	IN_Counter...		
	ワーク置き場4...		
UN_No	IN_Counter...		
	ワーク置き場5...		
UN_No	IN_Counter...		
	ワーク置き場6...		
UN_No	IN_Counter...		
	ワーク置き場7...		
UN_No	IN_Counter...		
	ワーク置き場8...		
UN_No	IN_Counter...		
	ワーク置き場9...		
UN_No	IN_Counter...		
	ワーク置き場10...		

在荷マトリクス			Work No.			
			ボール	ワーク	姿勢制御後 ワーク	ワーク+ ボールAss'y 溶接後
			1	2	3	4
Counter no.	シャーレ	1				
	供給トレイ	2				
	仮置き台1	3				
	溶接機	4				
	仮置き台2	5				
	排出トレイ	6				

### 3. 下記の様に在荷チェックフローに組み込み、 ロボット動作開始前にチェック機能を持たせることが可能



在荷マトリクス		Work No.				
		ボール	ワーク	姿勢制御後 ワーク	ワーク+ ボールAss'y	ワークAss'y 溶接後
		1	2	3	4	5
Counter no.	シャーレ	1				
	供給トレイ	2				
	仮置き台1	3				
	溶接機	4				
	仮置き台2	5				
	排出トレイ	6				



・サンプルのPick / Place時のアームの開閉時にサンプルの増減をカウントする機能。Counter no. / Mode no. Unit modeはワーク在荷と共通設定。供給トレイ2からワークをピックする際の設定は以下の様になる。



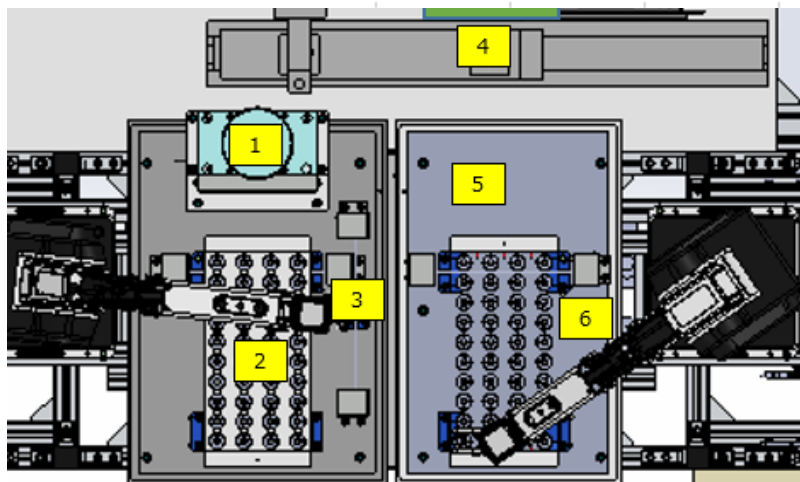
ワークPick時の  
チャック(閉)信号

### パラメータ設定

- ・Counter No. : 2
- ・Work No. : 2
- ・Mode No. : 2  
(加算=1, 減算=2)

・UN\_No : 作業対応場所に対する番号  
(固定条件)

Counter no. (サンプル配直場所) 設定

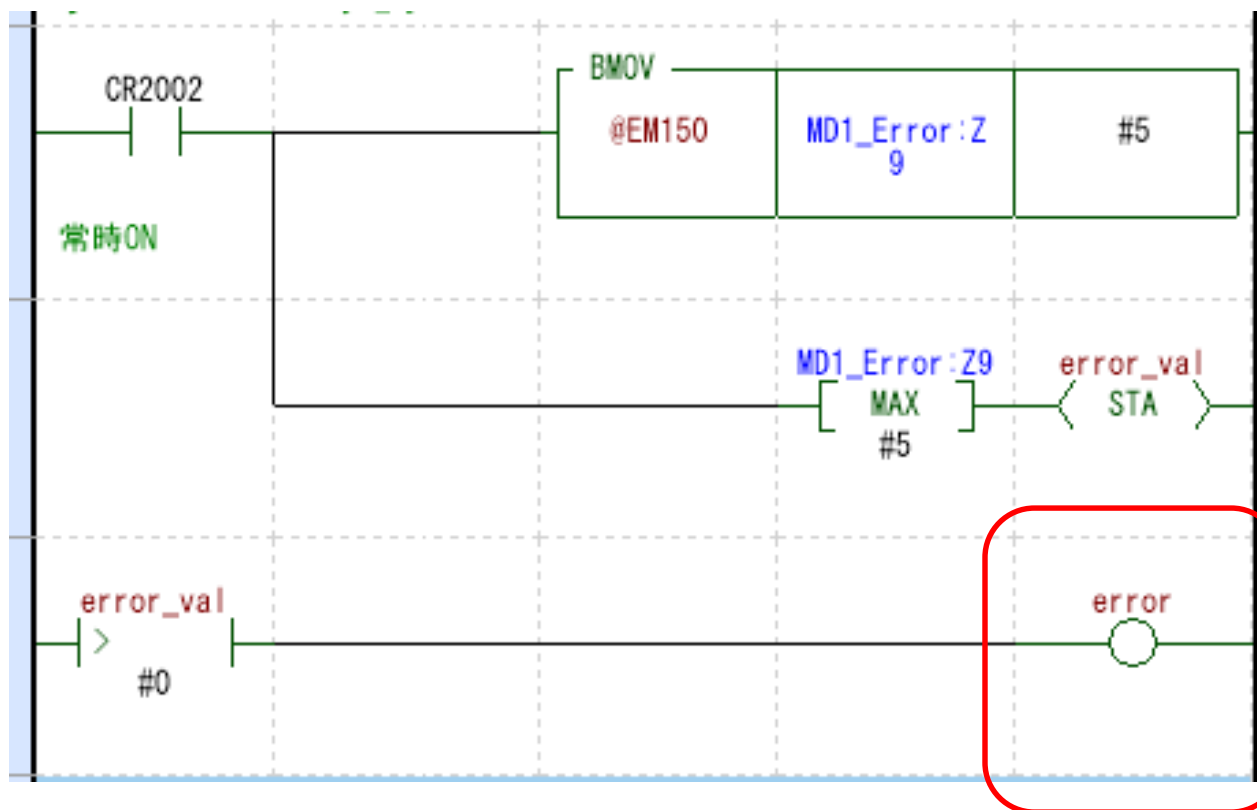
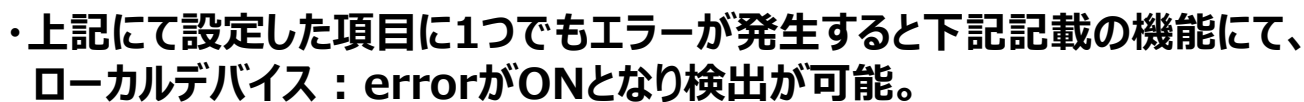


handIeWorkCnt	
#2	IN_Counter... ワーク置き場No.
#2	IN_Work_No ワーク種類No.
#2	IN_Mode_No 1:加算, 2:減算
UN_No	IN_Counter... ワーク置き場1...
UN_No	IN_Counter... ワーク置き場2...
UN_No	IN_Counter... ワーク置き場3...
UN_No	IN_Counter... ワーク置き場4...
UN_No	IN_Counter... ワーク置き場5...
UN_No	IN_Counter... ワーク置き場6...
UN_No	IN_Counter... ワーク置き場7...
UN_No	IN_Counter... ワーク置き場8...
UN_No	IN_Counter... ワーク置き場9...
UN_No	IN_Counter... ワーク置き場1...

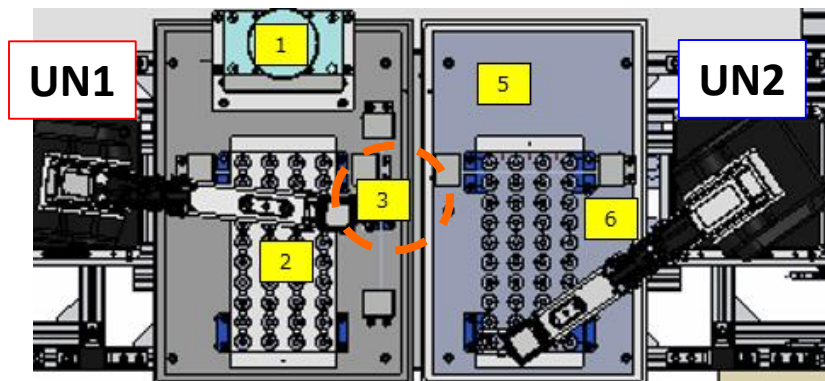
### 在荷Matrix設定

在荷マトリクス		Work No.			
		ボール	ワーク	姿勢制御後 ワーク	ワーク+ ボールAss'y ワークAss'y 溶接後
		1	2	3	4
Counter no.	シャーレ	1			
	供給トレイ	2			
	仮置き台1	3			
	溶接機	4			
	仮置き台2	5			
	排出トレイ	6			

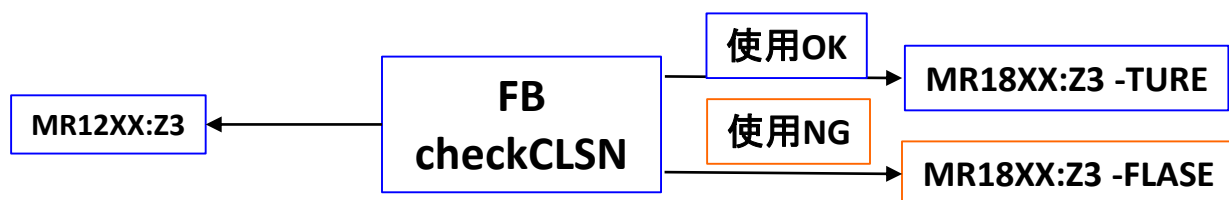
- ## エラー検出 ローカルデバイス



- ・下記の様に共通のワーク置き場所(no.3)をUN1/UN2の2つ以上のロボットが共有する際に、ロボット同士の接触と回避させるために設定。

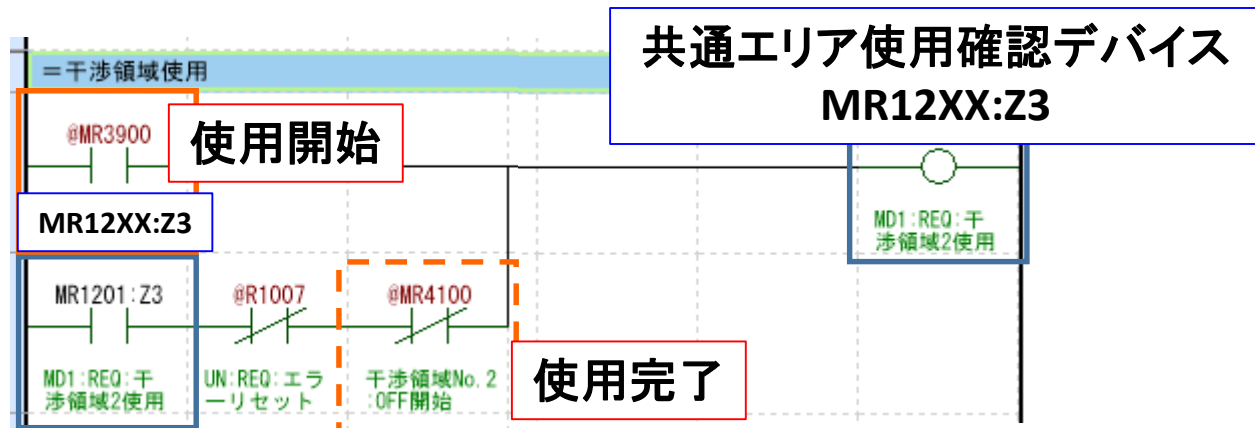


MR12xxをONにすることで、共通エリアの使用可否をMR18XXにて確認可

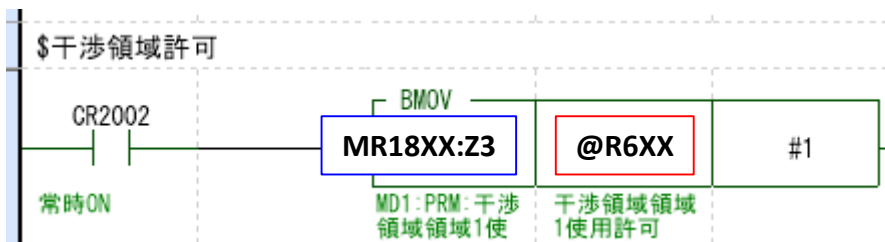


### ・設定

MR12XX:Z3のXX: XXの部分で00～15まで変更することで16か所の共通エリアの設定が可能。(現在は共通エリア4つまで設定可。15まで拡張予定)



プログラム間通信に下記のラダーを配置することで、@R6XXに共通エリア使用可否を反映

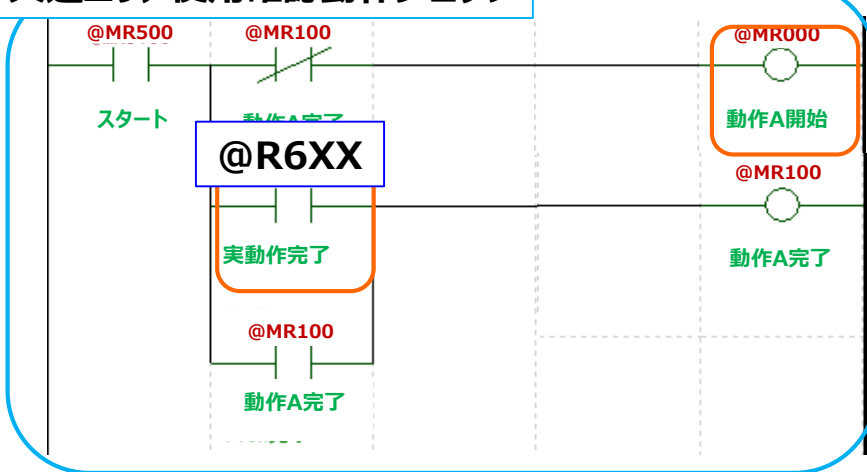


・共通エリアでの動作シーケンス前後に共通エリア使用確認 及び使用OFFの動作ブロックを配置。共通エリア使用確認の実動作完了には@R6XX、使用OFFの実動作完了にて@R6XXのB接点を配置。

他のユニットにて共通エリアを使用している場合@6XXがONにならないので、動作が開始されず、共通エリアにて2つのロボットが動作しない設定が可能。

### 共通エリアでの全動作シーケンス

#### 共通エリア使用確認動作ブロック

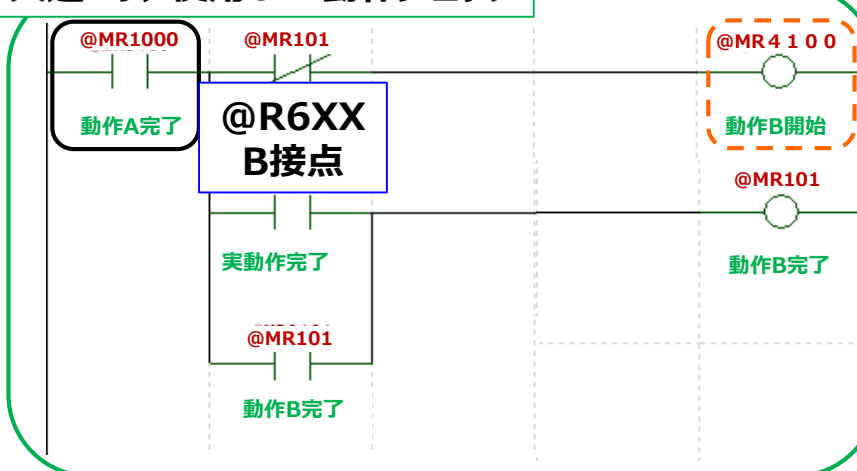


干渉領域使用  
の使用開始へ

#### 動作シーケンス

・  
・  
・

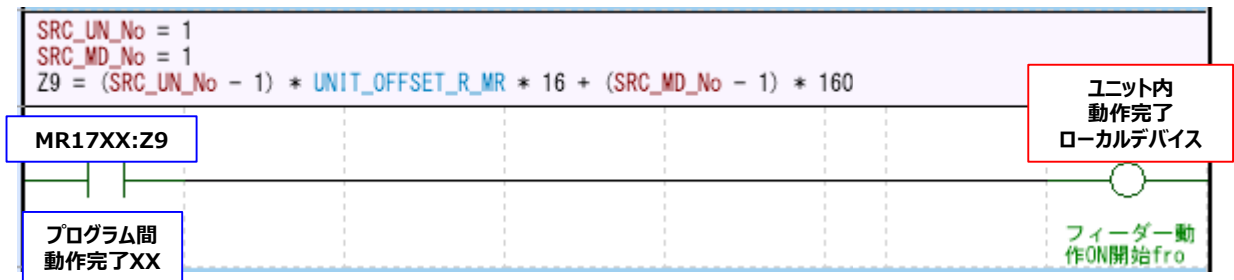
#### 共通エリア使用OFF動作ブロック



干渉領域使用  
の使用完了へ

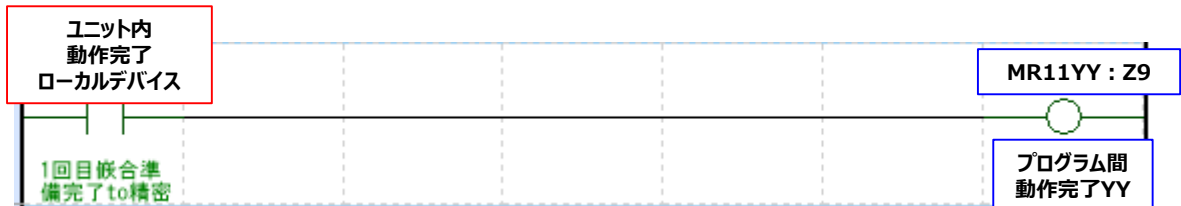
### ・Input側

グローバルデバイスにて管理している信号でユニット制御に必要となるものを任意のローカルデバイスに接続。1モジュール1word、入力 MR17XXに設定。/ 出力MR11XXを設定可。



### ・Output側

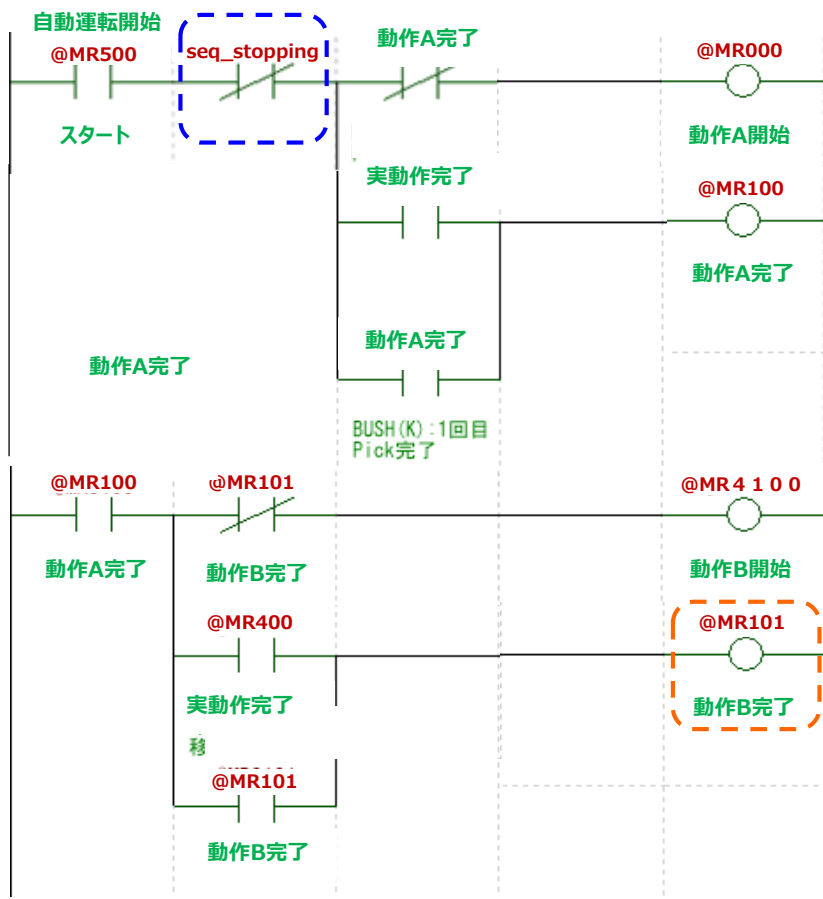
ユニット内のローカルデバイスにて管理している信号にて、他のユニット制御にて用いる信号をグローバルデバイスに接続。1モジュール1word、出力MR11XXを設定可。



## サイクル停止

・ローカルデバイスseq\_stoppingと任意のプロセス終了の信号を下記の様に配置することで、タッチパネルのcycle stopを押すと、自動運転中に任意のプロセス終了時に動作を終了することが可能。

ローカルデバイスseq\_stoppingは自動運転開始の直後に配置



動作を終了させたいプロセスの完了信号をサイクル停止のラダーの下記の位置に配置。

