



机器自动化控制器

Sysmac Studio的使用方法

欧姆龙自动化（中国）有限公司
ATC中心 商品应用技术部
SysMac应用课

内容介绍:

- 无需梯形图, 电机就能马上运行
- 伺服只要2个命令就能运行
- 电子凸轮·差补动作, 能够这么简单地就实现
- IO的追加也很简单·且无错误
- 程序, 变得更加易懂, 更加不容易出错
- 通过仿真功能, 用3D来确认伺服的动作

一、样机：无需梯形图，电机就能马上运行！

与CJ+NCF相比，启动作业会变得多么轻松。

故障排除，可以如何地实行。

CJ+NCF: 启动调试时最低限度需要的手册

目前为止

至少需要连上电机, 设定正确的参数, 进行试运行, 要读取7册的手册、并且要学会用软件工具。

マニュアル名称	Man. No.	形 式	用 途	内 容
SYSMAC CJ シリーズ 位置制御ユニット ユーザーズマニュアル (本マニュアル)	SBCE-359	形 CJ1W-NC281 形 CJ1W-NC481 形 CJ1W-NC881 形 CJ1W-NC81 形 CJ1W-NC82 形 CJ1W-NC82 形 CJ1W-NC82	NC ユニット (形 CJ1W-NC281/ 481/ 881/ F81/ 482/ 882/ F82) について知り たいとき	NC ユニットの設定方法、操作方 法について説明しています。
CJ シリーズ CJ2 CPU ユニット ユーザーズマニュアル ソフトウェア編	SBCE-350	形 CJ2H-CPU6 □ -EIP 形 CJ2H-CPU6 □ 形 CJ2M-CPU □ □	CJ2 CPU ユニットの ソフトウェア的な仕様 について知りたいとき	CJ2 CPU ユニットに関して、以 下の内容を説明しています。 ・ CPU ユニットの動作について 知りたい ・ 内部メモリについて知りたい ・ プログラムについて知りたい ・ 各種設定について知りたい ・ CPU 内蔵機能について知りたい ユーザーズマニュアル ハード ウェア編 (SBCE-349) と併せて 使用してください。
CJ シリーズ CJ2 CPU ユニット ユーザーズマニュアル ハードウェア編	SBCE-349	形 CJ2H-CPU6 □ -EIP 形 CJ2H-CPU6 □ 形 CJ2M-CPU □ □	CJ2 CPU ユニットの ハードウェア的な仕様 について知りたいとき	CJ2 CPU ユニットに関して、以 下の内容を説明しています。 ・ 概要・特長について知りたい ・ 基本システムの構成を知りたい ・ 各部の名称と機能を知りたい ・ 取り付けと設定方法を知りたい ・ トラブル時の対処方法を知り たい ユーザーズマニュアル ソフト ウェア編 (SBCE-350) と併せて 使用してください。
CJ/NSJ シリーズ ユーザーズマニュアル セットアップ編	SBCE-312	形 CJ1H-CPU □ □ H-R 形 CJ1G/H-CPU □ □ H 形 CJ1G-CPU □ □ P 形 CJ1M-CPU □ □ 形 CJ1G-CPU □ □ 形 NSJ □ - □ □ □ □ (B)-G5D 形 NSJ □ - □ □ □ □ (B)-M3D	CJ/NSJ シリーズの概 要/設計/取り付け/ 保守などの基本的な仕 様について知りたいと き	CJ1 CPU の PLC 本体に関して、 以下の内容を説明しています。 ・ 概要・特長を知りたい ・ システム構成を設計したい ・ 取り付け/配線を知りたい ・ I/O メモリの割り付けを知り たい ・ トラブル時の対処方法を知り たい ユーザーズマニュアル プログラ ミング編 (SBCE-313) と併せて 使用してください。

マニュアル名称	Man. No.	形 式	用 途	内 容
OMNUC G5 シリーズ AC サーボモータ/ド ライバ (EtherCAT 選 信内蔵タイプ) ユーザーズマニュアル	SBCE-385	形 R88M-K □ 形 R88D-KN □ -ECT	OMNUC G5 シリーズ サーボドライバの設定 について知りたいとき	OMNUC G5 シリーズサーボドラ イバの設定方法、調整方法につ いて説明しています。
CJ/NSJ シリーズ ユーザーズマニュアル プログラミング編	SBCE-313	形 CS1G/H-CPU □ □ H CS1G/H-CPU □ □ -V1 CS1D-CPU □ □ H CS1D-CPU □ □ S 形 CJ1H-CPU □ □ H-R 形 CJ1G/H-CPU □ □ H 形 CJ1G-CPU □ □ P 形 CJ1M-CPU □ □ 形 CJ1G-CPU □ □ 形 NSJ □ - □ □ □ □ (B)-G5D 形 NSJ □ - □ □ □ □ (B)-M3D	CS/CJ/NSJ シリーズ の各種機能について知 りたいとき	CS/CJ シリーズの PLC 本体に関 して、以下の内容を説明していま す。 ・ プログラミングをしたい ・ タスク機能を知りたい ・ ファイルメモリ機能を知りたい ・ 各種の機能を知りたい ユーザーズマニュアル セット アップ編 (SBCE-312) と併せて 使用してください。
CX-Programmer オペレーション マニュアル	SBCE-337	形 WS02-CX □ □ -V □	Windows パソコン用 プログラミングツール CX-Programmer の操 作方法について知り たいとき	CX-Programmer の操作方法につ いて説明をしています。

即使是在启动调试手册(技术向导)里, 也有170page。。。Automation Systems

CJ+NCF:连接上、进行轴设定、MC试运行

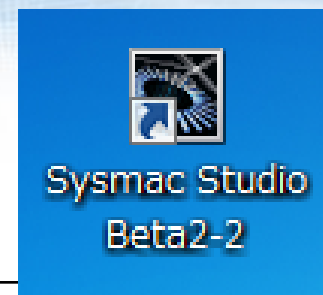
从今以后

把当前的构成配置在线读取出来，登录了使用的轴后，无需梯形图就能进行试运行了。

Sysmac Studio的起动和终止

说明一下起动Sysmac Studio的步骤。

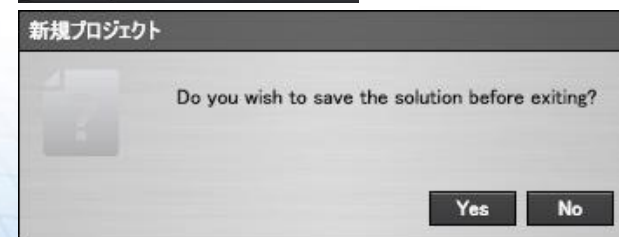
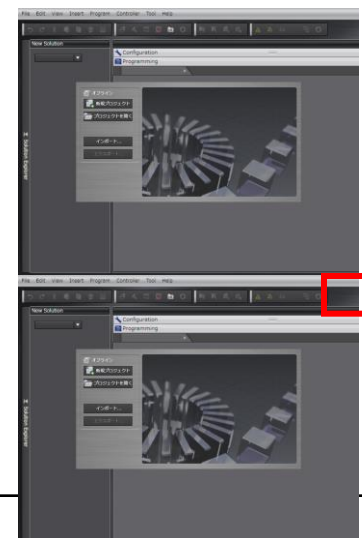
- 1 双击桌面上的Sysmac Studio图标。
(或者从所有程序里选择Sysmac Studio。)
- 2 Sysmac Studio起动。



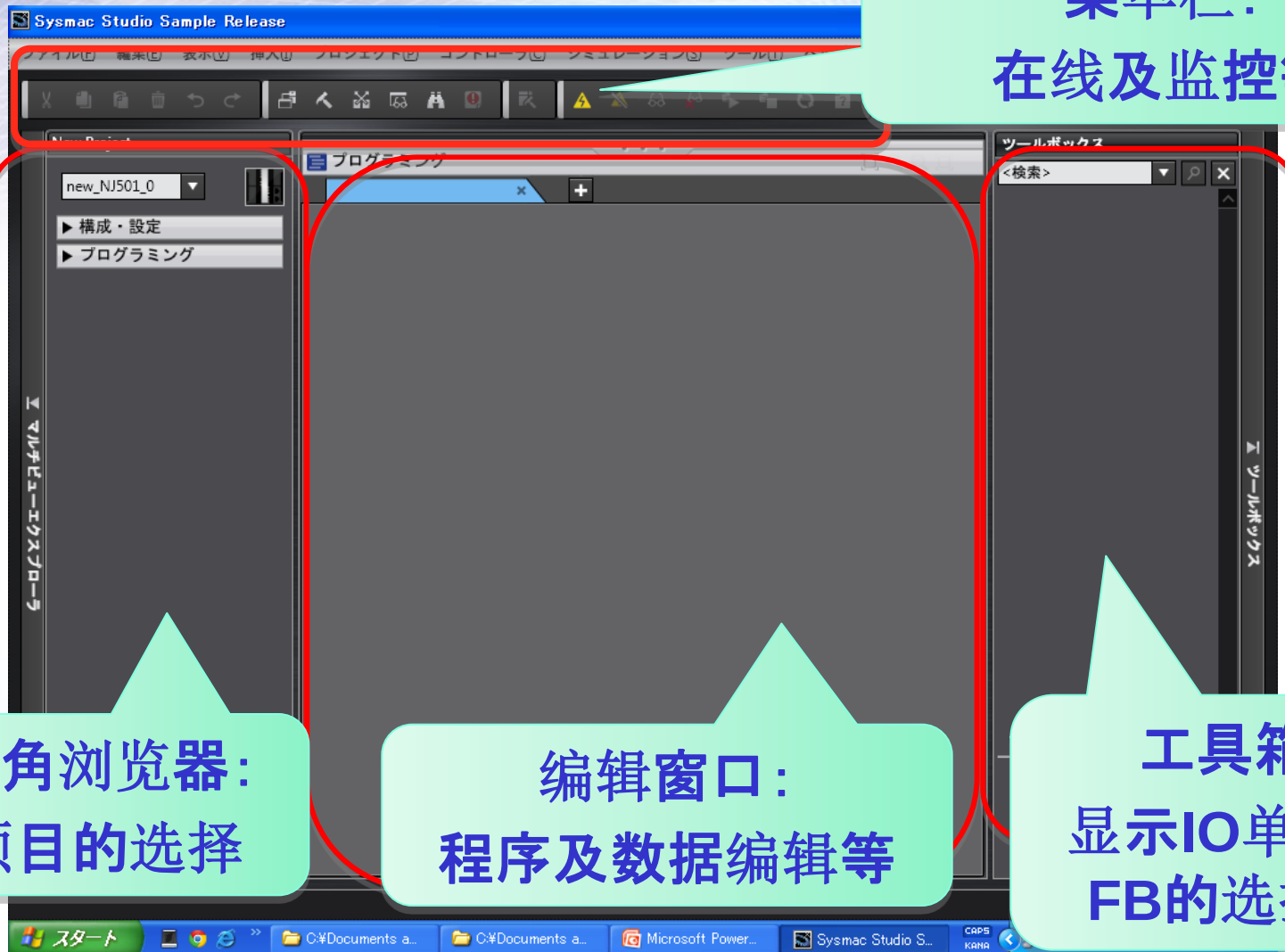
说明一下终止Sysmac Studio的步骤。

- 1 点击标题栏右端的 [×]。
(或者从菜单里选择 [文件] - [终止]。)
- 2 会显示右图的对话框。
选择 [Yes] 或者 [No] 。

Sysmac Studio终止。



SysmacStudio的主画面



菜单栏：
在线及监控等

多视角浏览器：
大项目的选择

编辑窗口：
程序及数据编辑等

工具箱：
显示IO单元及
FB的选择项

EtherCATの設定

NJ系列CPU单元的内置EtherCAT端口上连接的EtherCAT从站的构成配置，在Sysmac Studio上进行制作。另外，还要进行EtherCAT的主站以及从站的设定。

在CJ里
做不了的



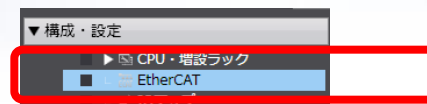
首先，与NJ进行
在线(同步)。



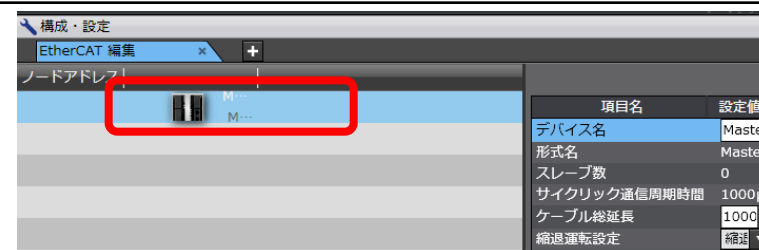
EtherCATの設定: 构成配置の上传

对EtherCAT构成的制作步骤进行说明。

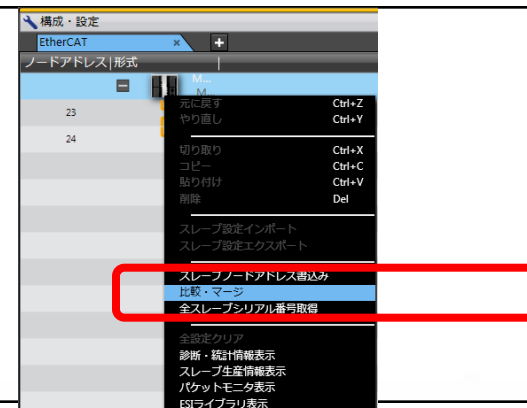
- 1 右击多视角浏览器内的[构成・设定] — [EtherCAT]、点击[编辑]。



- 2 [构成・设定] 编辑内会显示EtherCAT的构成编辑画面。
会看到主站从站的登录。



- 3 从NJ里读取上来构成配置信息。
右击NJ、从菜单里选择「比较・合并」。



- 4 右击连接端的NJ、从菜单里选择「使用实机构成」后、实机的构成配置就会上传到 SysmacStudio 里。



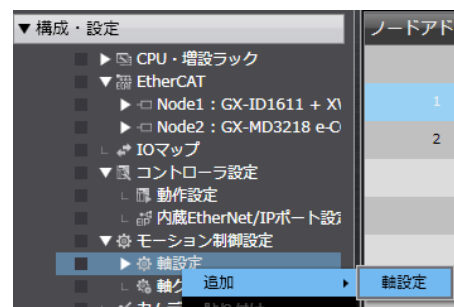
运动控制设定

用运动命令使用的轴的登录、对轴所利用的伺服驱动器、编码器轴的关联操作、轴的参数设定等的一连串设定。

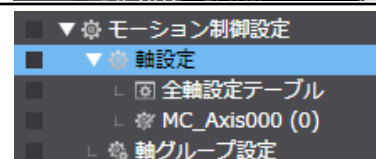
<轴的登录>

对用运动命令使用的轴，进行登录。

- 1 右击多视角浏览器内的「构成・设定」 - 「运动控制设定」 - 「轴设定」、选择「追加」 - 「轴设定」。

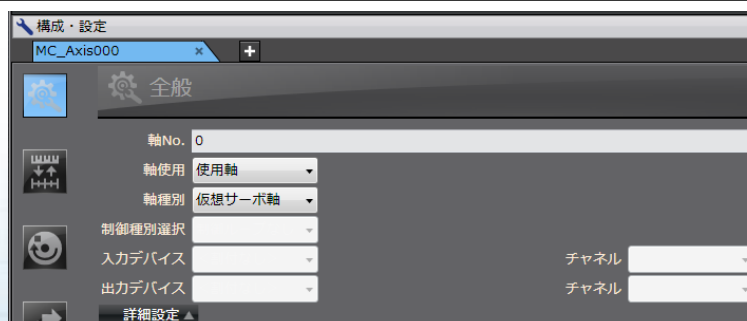


- 2 在[轴设定]的下面追加轴「MC_Axis000」。
(多视角浏览器内)



- 3 双击MC_Axis000。

- 4 在编辑窗口内的配置层里显示轴参数设定画面。
在初始画面里，会显示「全体」。



运动控制设定

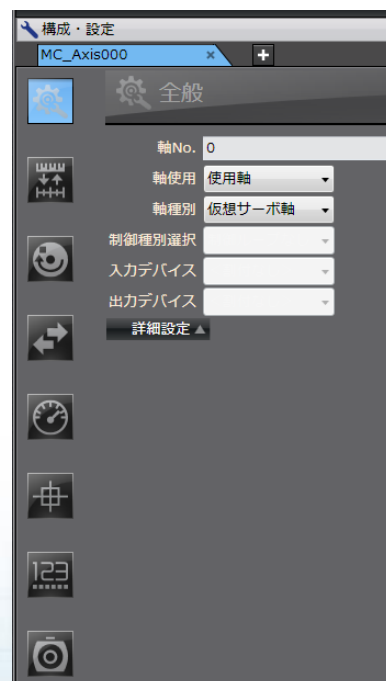
5 利用伺服轴时，选择以下内容。

- 轴使用 : 使用轴
- 轴类别 : 伺服轴



6 在输入设备里、选择已登录的伺服轴。

7 通过轴参数设定视图的左端所显示的按钮，来切换设定画面。

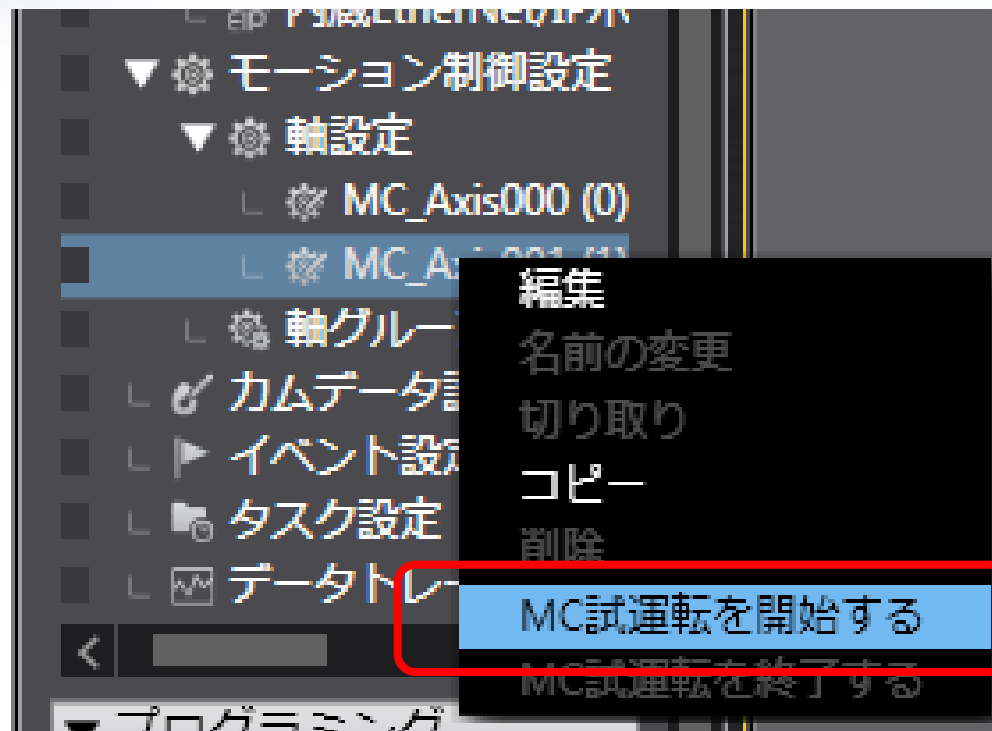


用同样的操作，再制作
1轴。

MC试运行:无需梯形图可进行电机是否运行的确认。

- 1 选择多视角浏览器内的[运动控制设定]
— [轴设定]、右击作为对象的轴。

从右击菜单里、选择「开始MC试运行」。



MC试运行的画面

OMRON

The screenshot shows the OMRON MC Test Run interface. At the top, the title bar reads '構成・設定' (Configuration/Setting) and the window title is 'MC試運転' (MC Test Run). Below this, the '軸選択' (Axis Selection) dropdown is set to 'MC_Axis001(1)(Node 24)'. The 'ステータス' (Status) section contains several buttons: '軸起動準備完了' (Axis Start Preparation Complete), '軸無効' (Axis Invalid), '停止中' (Stopping), '位置決め動作中' (Positioning Operation), '連続動作中' (Continuous Operation), '原点復帰中' (Origin Return), '減速停止中' (Deceleration Stop), 'エラー減速停止中' (Error Deceleration Stop), '原点確定' (Origin Confirmation), and '原点停止' (Origin Stop). The 'モーション異常一覧' (Motion Abnormality List) section has a table with columns 'イベントレベル' (Event Level), '発生源' (Occurrence Source), and 'メッセージ略称' (Message Abbreviation). The '試運転' (Test Run) section includes a '減速停止' (Deceleration Stop) button and a 'サーボOFF' (Servo OFF) button. The 'サーボドライバ状態' (Servo Driver Status) section displays a table of current values and a set of input/output status buttons. The 'ジョグ送り' (Jog Feed) section is highlighted with a red box and contains a table for '目標速度' (Target Speed), '加速度' (Acceleration), and '減速度' (Deceleration). Below this table is an '適用' (Apply) button. At the bottom, there are two large buttons: a yellow 'Jog Feed' button and a green 'Start' button. Three callouts provide additional information: one points to the '原点復帰' (Origin Return) button, another points to the '目標速度' (Target Speed) input field, and a third points to the '適用' (Apply) button.

構成・設定

MC試運転

軸選択 MC_Axis001(1)(Node 24)

ステータス

軸起動準備完了 軸無効 停止中 位置決め動作中 連続動作中 原点復帰中 減速停止中 エラー減速停止中 原点確定 原点停止

モーション異常一覧

イベントレベル | 発生源 | メッセージ略称 |

メッセージ詳細

対策・処置

異常解除

試運転

減速停止 サーボOFF

サーボドライバ状態

項目	値	単位
指令現在位置	-73000	pulse
フィードバック現在位置	-73000	pulse
指令現在速度	0	pulse/s
フィードバック現在速度	0	pulse/s

項目	状態
正方向限界入力	OFF
負方向限界入力	OFF
即停止入力	OFF
原点近傍入力	OFF
ラッチ入力1	OFF
ラッチ入力2	OFF
原点入力	OFF
サーボレディ	ON
主回路電源	ON
ドライバ電源	ON
ラッチ入力	OFF

ジョグ送り 絶対値位置決め 相対値位置決め 原点復帰

項目	値	単位
目標速度	1000000	pulse/s
加速度	0	pulse/s ²
減速度	0	pulse/s ²

適用

ボタンを押しての間のみ動作します

起動按钮

不仅是JOG、还可以进行绝对·相对移动以及原点返回

输入目标速度及加速度等数据

启动按钮

二、伺服只需2个命令就能运行！

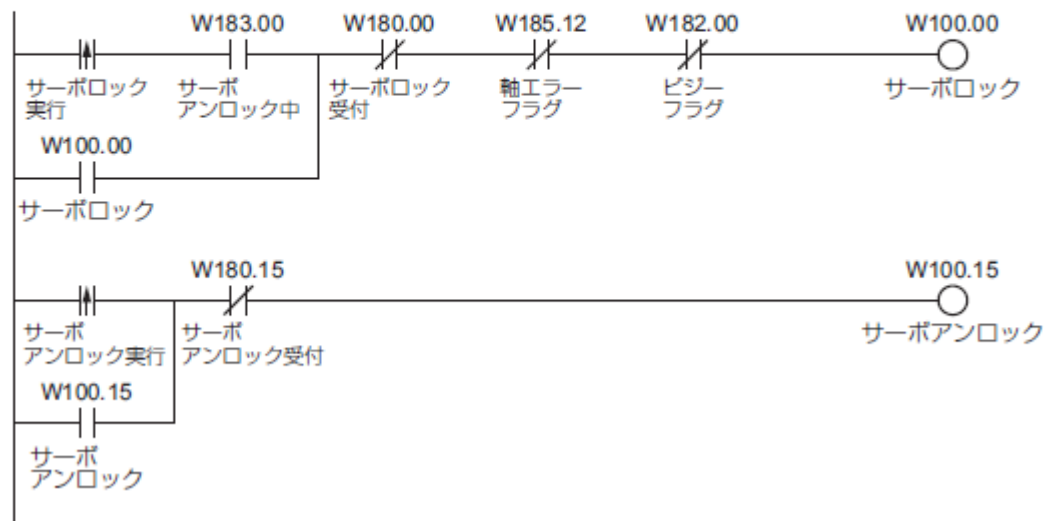
目前为止:为了能进行伺服锁定...

目前为止

在各个轴里进行正确的IO分配后, 还需要以下这样的程序。如果没有加入注释、**仅仅只是IO的罗列**, 在进行怎样的操作, **其他的人谁也不知道**。

サーボロック／サーボアンロック

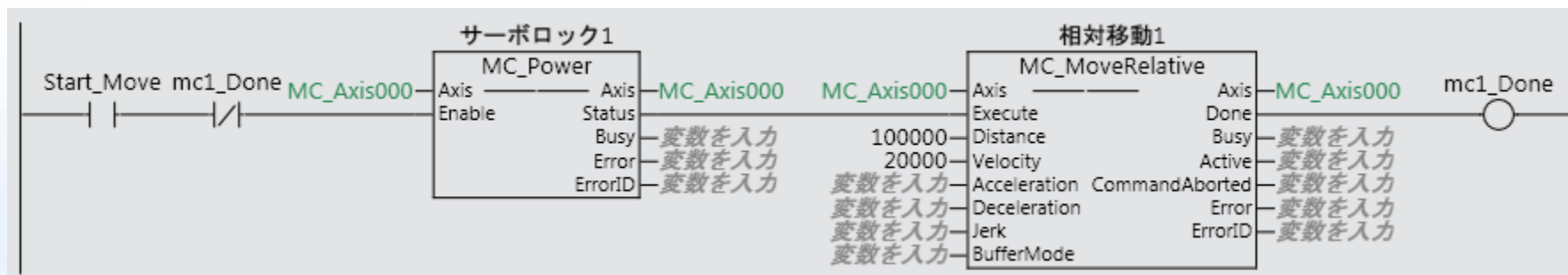
本プログラム例では、軸 1 を対象として、サーボロック、サーボアンロックを実行します。
なお、手動運転指令リレーエリア選択ならびに軸ステータスリレーエリア選択は共通パラメータの設定に基づくものとします。



用这个回路伺服就能运行了！

制作下列梯形图程序。

★什么、仅仅只有2个命令、电机就运行了！！！！



向指定位置的连续运行的方法，有变化吗？

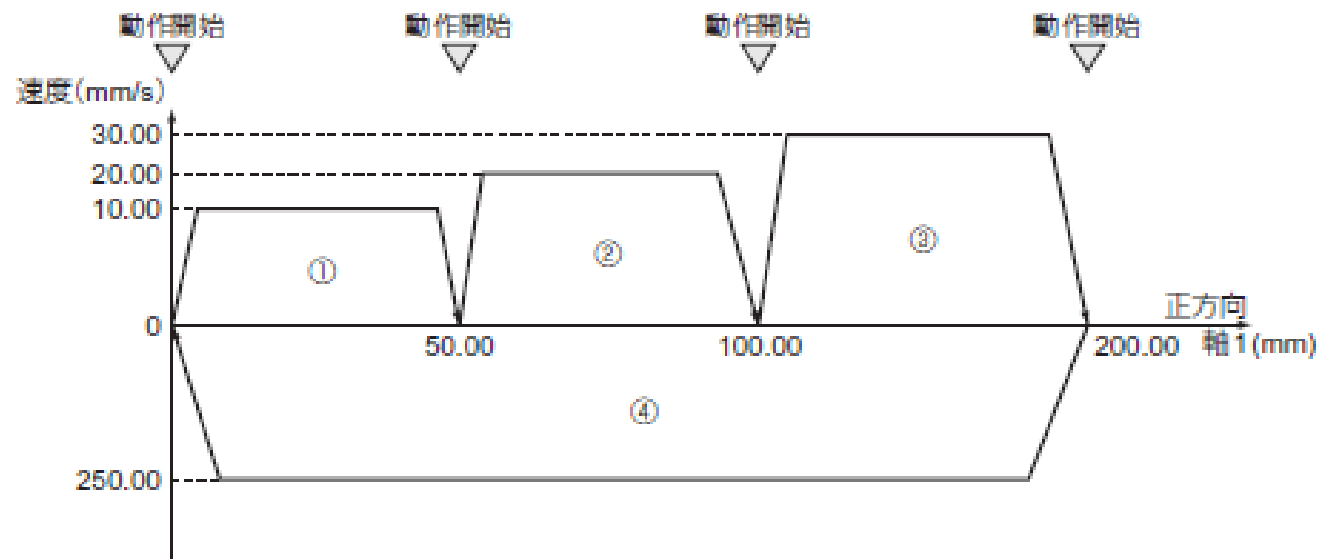
目前为止:为了移动到指定位置...

目前为止

在各个轴里进行正确的IO分配后,
向DM正确地传送参数、
需要以下这样的程序

動作例

データメモリ上の位置データを使用し、絶対移動による位置決めの動作例を以下に示します。
原点はあらかじめ、原点復帰または、現在位置プリセットで設定してください。



目前为止:为了移动到指定位置...

目前为止

需要向指定的地方,将必要的数据正确地用MOVE适当地传送。

データメモリの設定

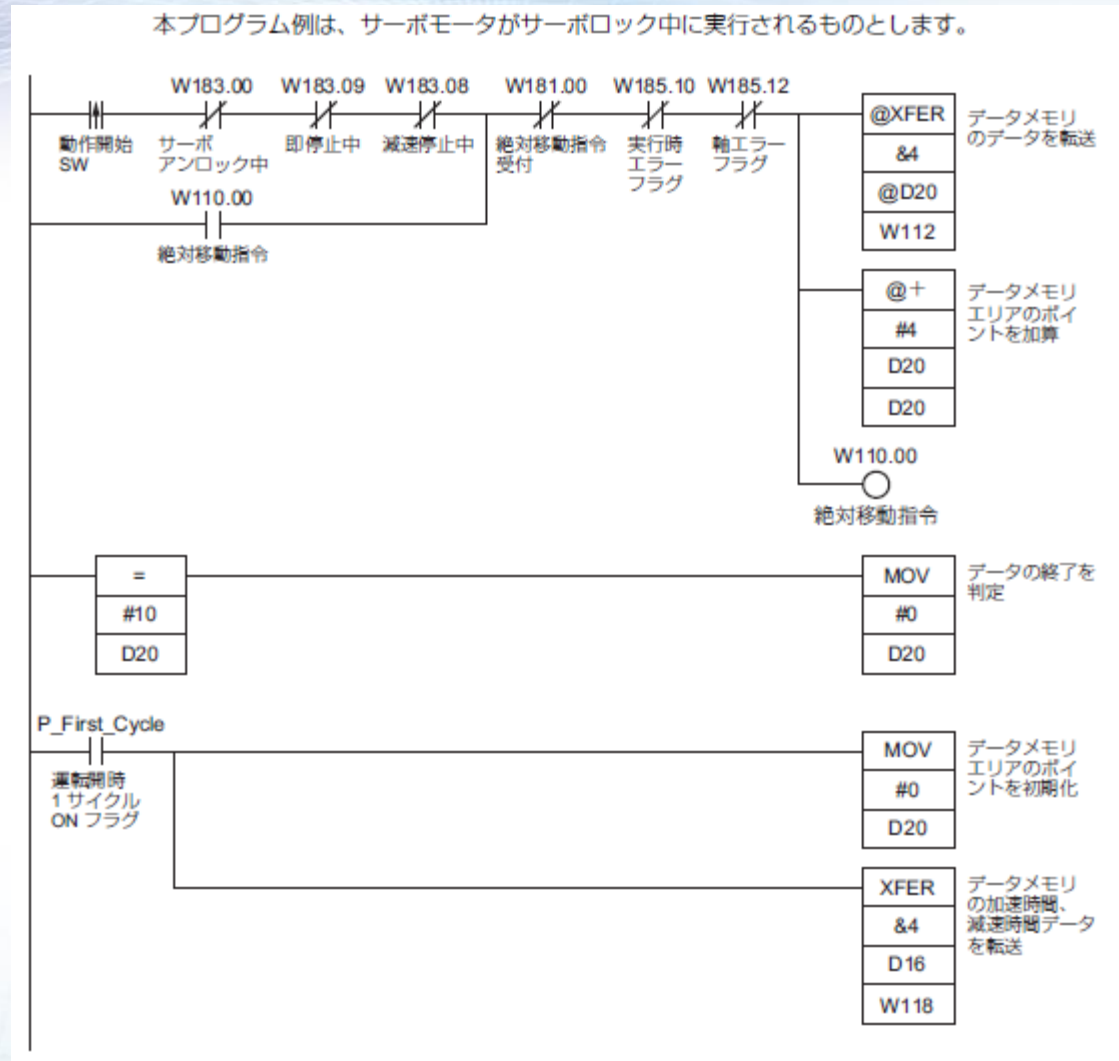
本プログラム例ではデータメモリエリア D00000 ~ D00019 を使用し、運転出力リレーエリアに以下の位置データを設定します。

データメモリ	設定値	設定内容
D00000	5000 (1388Hex)	位置指令: 50.00mm (モータ 10 回転の位置)
D00001	0000Hex	
D00002	1000 (03E8Hex)	指令速度: 10.00mm/s
D00003	0000Hex	
D00004	10000 (2710Hex)	位置指令: 100.00mm (モータ 20 回転の位置)
D00005	0000Hex	
D00006	2000 (07D0Hex)	指令速度: 20.00mm/s
D00007	0000Hex	
D00008	20000 (4E20Hex)	位置指令: 200.00mm (モータ 40 回転の位置)
D00009	0000Hex	
D00010	3000 (0BB8Hex)	指令速度: 30.00mm/s
D00011	0000Hex	
D00012	0000Hex	位置指令: 0.00mm
D00013	0000Hex	
D00014	25000 (61A8Hex)	指令速度: 250.00mm/s
D00015	0000Hex	
D00016	0064Hex	加速時間: 100ms
D00017	0000Hex	
D00018	0064Hex	減速時間: 100ms
D00019	0000Hex	

目前为止:为了移动到指定位置...

目前为止

在事先将数据
以MOVE传送后、
做一个适当的互锁
建立起动BIT。
动作的终止也必须
个别地用梯形图进行
管理才可以。



从今以后：连续地、移动到指定地方

★只需这样。。

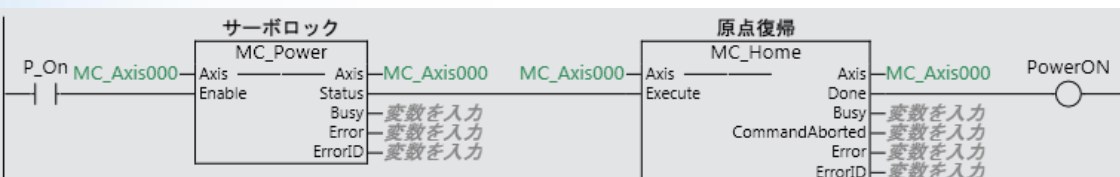
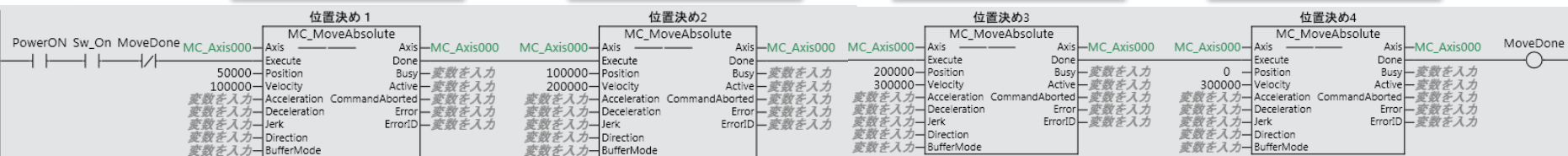
活用复制&粘贴、就更加地轻松！！

第1个

第2个

第3个

第4个



← 伺服锁定和原点返回

POU的登录

POU (Program Organization Unit)

所谓POU、是指构成程序的实行处理的部品。
根据不同的构成单位，有程序、功能、功能块三种部品。

从多视角浏览器内的[编程] - [POU]里，登录程序、功能、功能块、记述实行算法。

显示POU。

- 1 点击多视角浏览器内的[编程] - [POU]。
- 2 POU的下面，会显示「程序」、「功能」、「功能块」的项目。

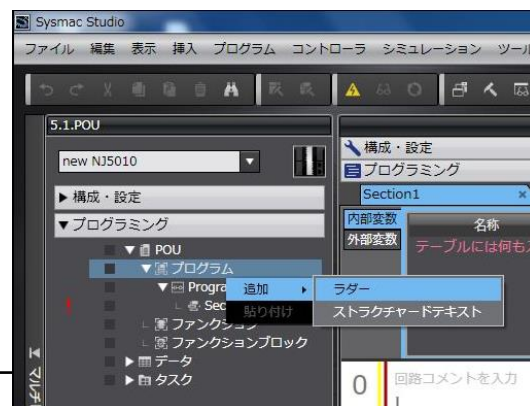


注. 项目新建时POU（梯形图）会做登录。

POUの登録

＜参考＞ 説明一下POUの追加登録方法。

- 1 右击多视角浏览器内的[编程] - [POU] - [程序]。
- 2 选择[追加] - [梯形图]。



- 3 在[程序]项目的下方、追加了[Program1] - [Section1]。



＜参考＞ 快捷键(程序编辑篇)

功能	按键/快捷键	菜单
输入a接点	[C]	无
输入b 接点	[I]	无
输入a接点OR	[W]	无
输入b 接点OR	[X]	无
输入输出线圈	[O]	无
输入输出线圈(否定)	[Q]	无
呼出功能块	[F]	无
呼出功能	[Shift]+[F]	无
给所有程序加上程序check	[F7]	[程序] - [全部程序check]
给所选择的程序加上程序check	[Shift] + [F7]	[程序] [部分程序check]
生成项目文件	[F8]	[程序] [生成]
重建项目文件	[Shift] + [F8]	无

注. 蓝色字体部分为与CX-Programmer相同操作的部分。

三、即使是电子凸轮·差补、、、也简单。

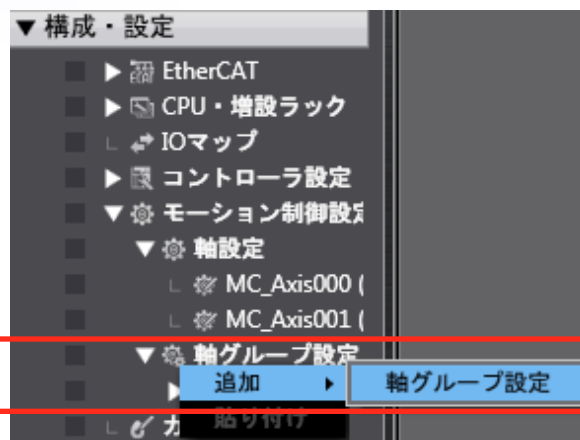
轨迹动作也能很易明白地表现出来。

- 将想使之动作的地方、用命令指定、连接上。

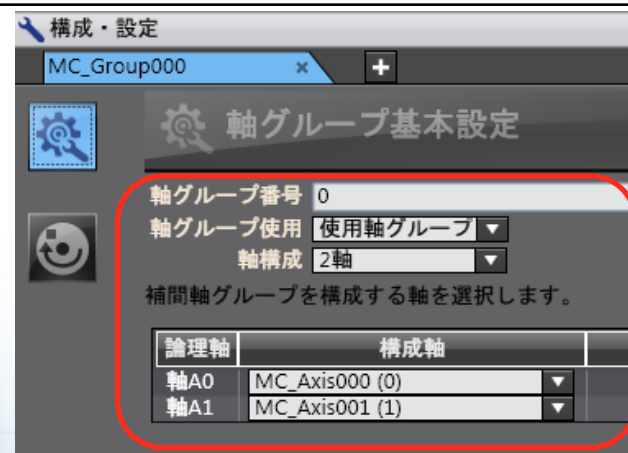
从今以后 : 想要同步的轴进行Group登录

将想要使之同步的轴、作为轴Group进行登录。

- 1 右击多视角浏览器内的 [构成・设定] - [运动控制设定] [轴Group设定]、选择 [追加] - [轴Group设定]。



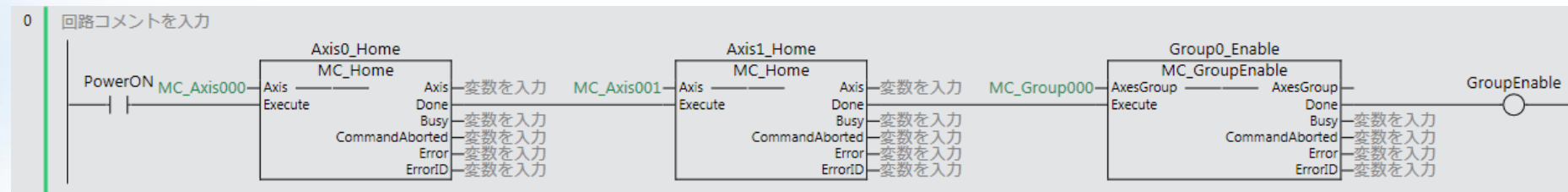
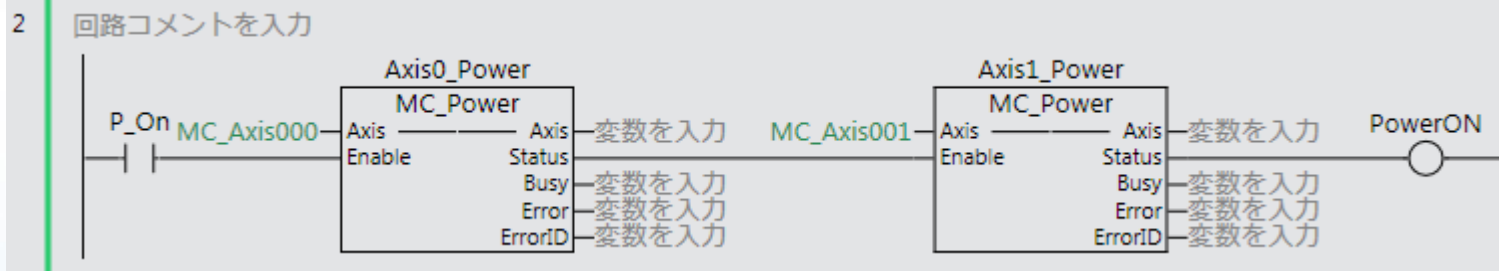
- 2 输入想要同步的轴数、以及分配好的轴。



从今以后：伺服锁定→原点返回→同步ON！

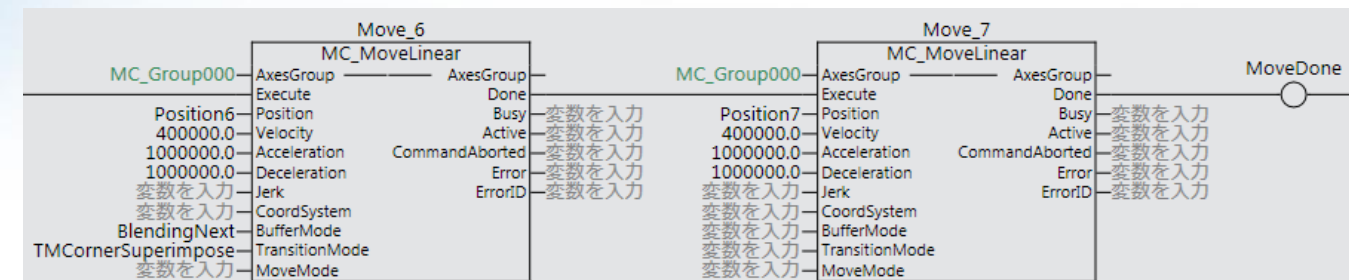
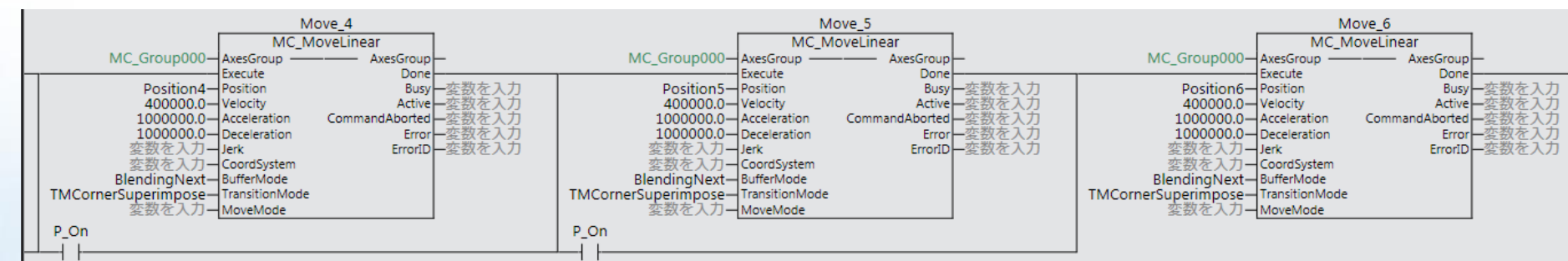
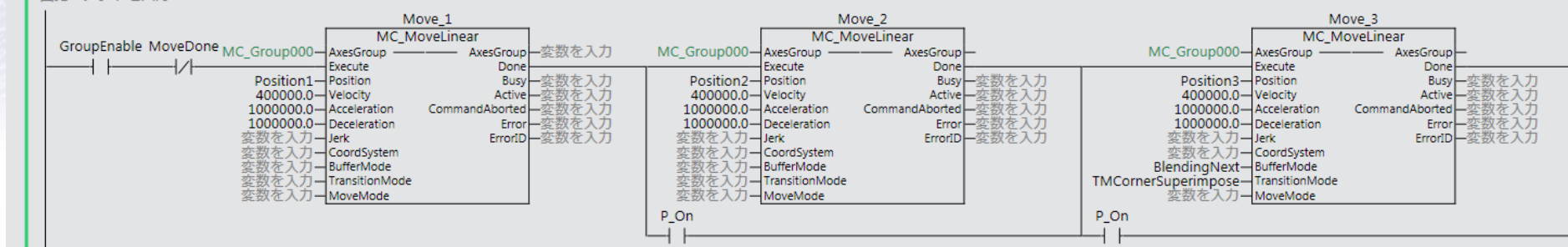
制作下述的梯形图程序。

沿着流程、把动作连接上



从今以后：按照想要描绘的轨迹的顺序连接上。 沿着流程、把动作连接上

1 回路コメントを入力



轨迹数据的登录

记载轨迹命令、进行数据模型的登录。

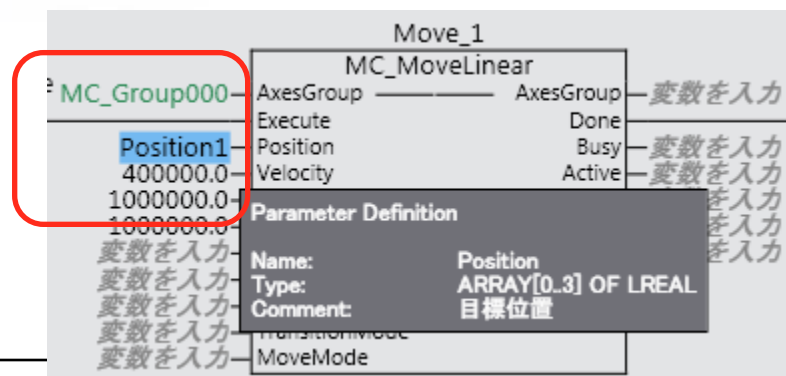
1 使用进行轨迹动作的命令

[MC_MoveLinear]。

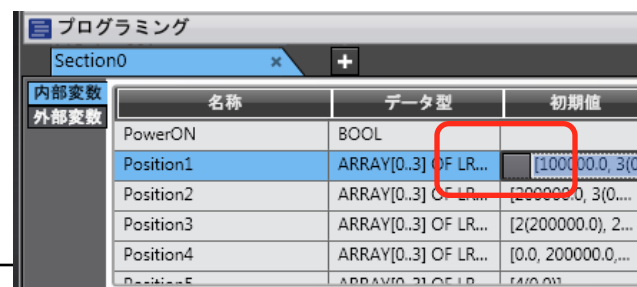
在AxisGroup里登录轴Group名、

在Position里登录用于轨迹的数据变量。

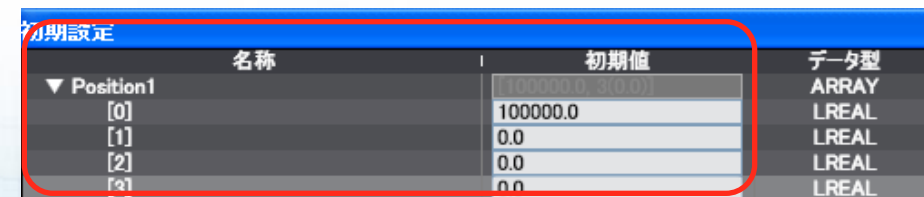
在Velocity里登录速度、除此以外的参数根据需要进行登录。



2 在登录于变量表的用于轨迹的数据变量里双击变量的初始值。



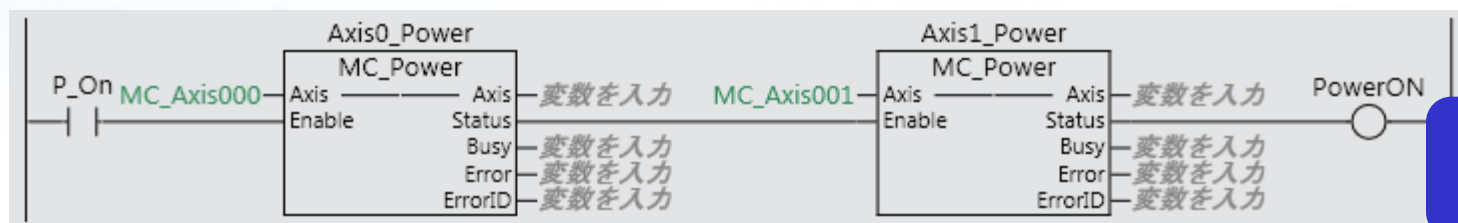
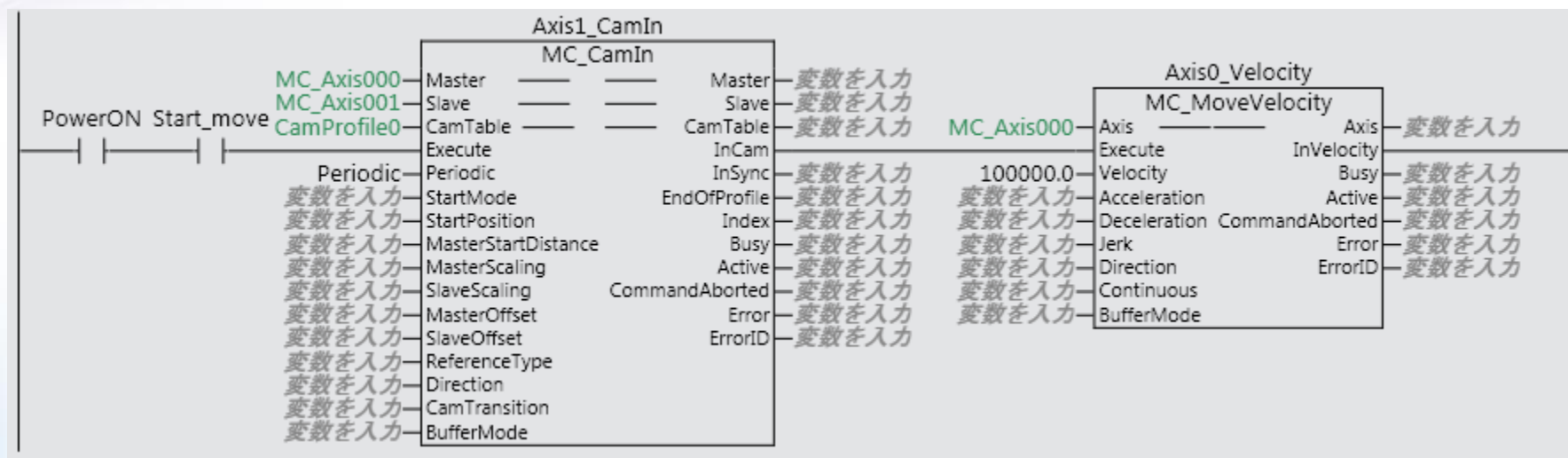
3 按下表格左边出来的黑色按钮后、就能编辑数据了。



从今以后：凸轮动作、更加易读、更加简单。

制作下述的梯形图程序。

电子凸轮、也只要2个命令！



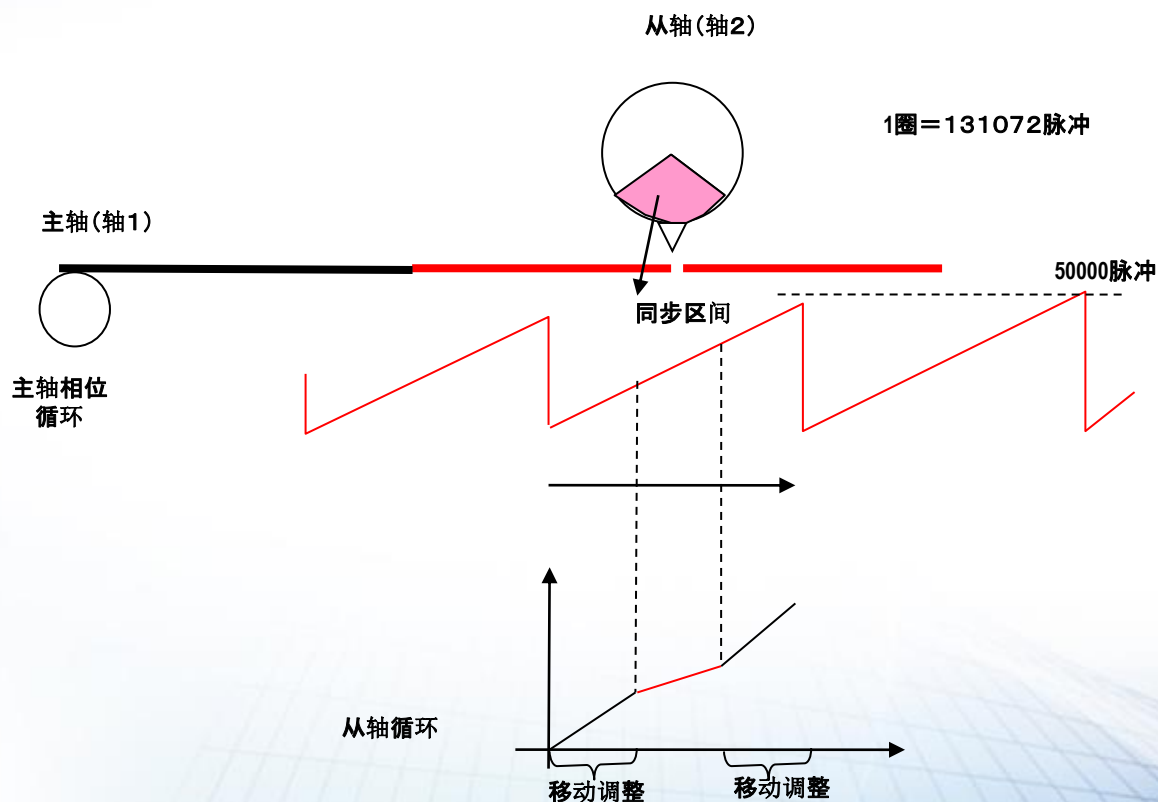
这个是用伺
服锁定的。

凸轮的动作

使用电子凸轮让速度在同步状态下，试着做一下能完成切割／插入等动作的凸轮表和程序。

使用样机，制作能让轴1和轴2能完成下述电子凸轮动作的程序。

动作示意：旋转切刀



轴参数的登录

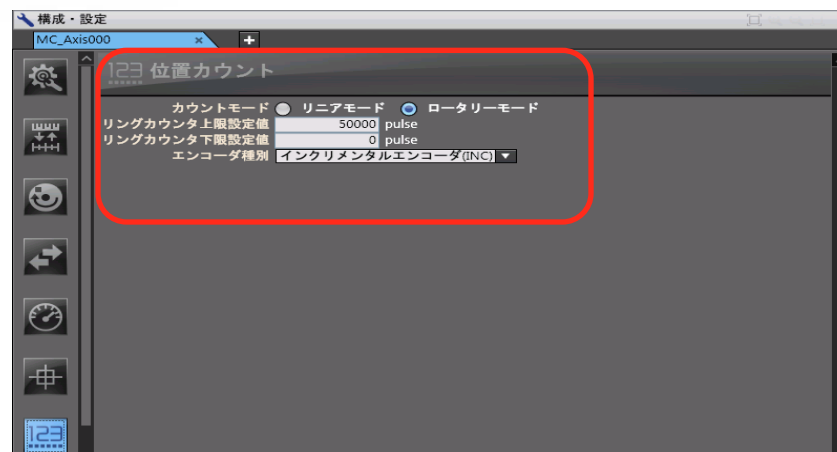
登录电子凸轮的主轴、从轴参数。

- 1 右击多视角浏览器内的[构成・设定] - [轴设定]、选择编辑窗口的[位置计数]。

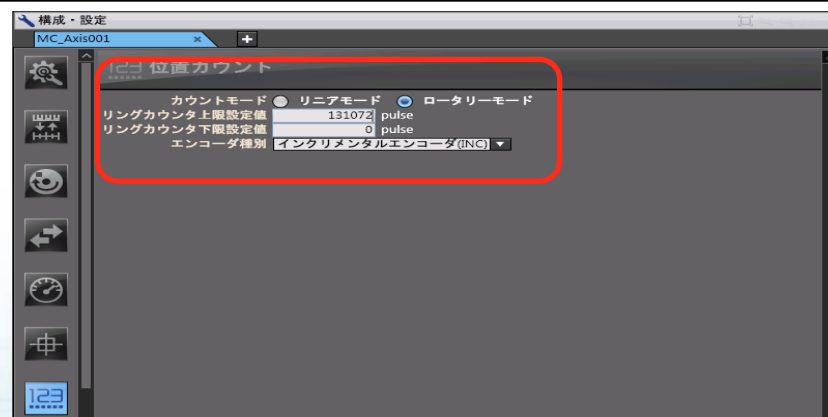
将计数模式设定为「旋转」

环形计数器的上限值设为[50000]

下限值设为[0]。



- 2 同样地、从轴的[位置计数]设定为计数模式为[旋转]
环形计数器的上限值设为[1301072]
=G5的ABS编码器1圈的量
下限值设为[0]。



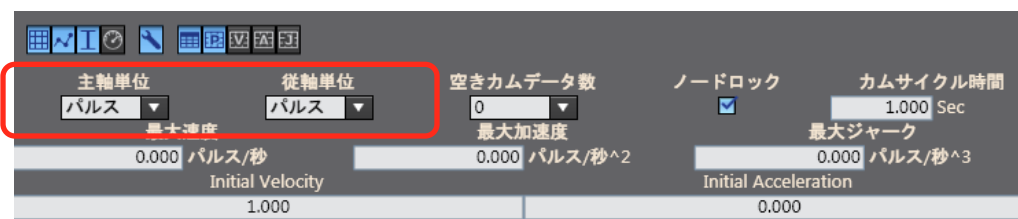
凸轮数据的登录

登录电子凸轮的凸轮模型。

- 1 右击多视角浏览器内的[构成・设定] - [凸轮数据设定]、选择[追加] - [CamProfile(NJ系列)]。



- 2 输入主轴的单位以及速度等。
主轴・从轴、请都将单位设定为[脉冲]。



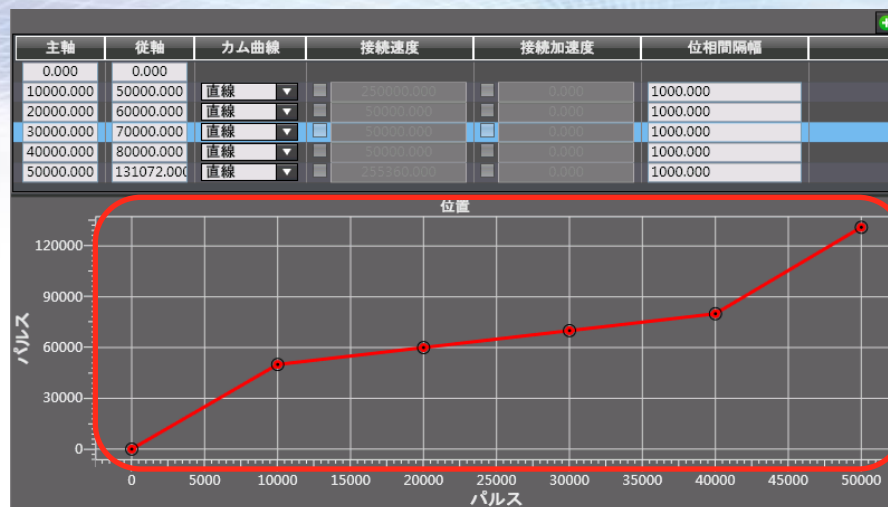
- 3 登录主轴和从轴的凸轮数据。

主軸	従軸	カム曲線	接続速度	接続加速度	位相間隔幅
0.000	0.000	直線	250000.000	0.000	1000.000
10000.000	50000.000	直線	30000.000	0.000	1000.000
20000.000	60000.000	直線	30000.000	0.000	1000.000
30000.000	70000.000	直線	30000.000	0.000	1000.000
40000.000	80000.000	直線	30000.000	0.000	1000.000
50000.000	131072.000	直線	255360.000	0.000	1000.000

运动控制设定

- 4 根据所登录的凸轮数据、在下面会显示图表。

可以马上对凸轮的模型进行确认。

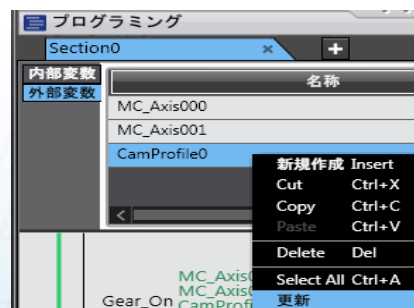


- 5 完成的凸轮、成为数组数据。

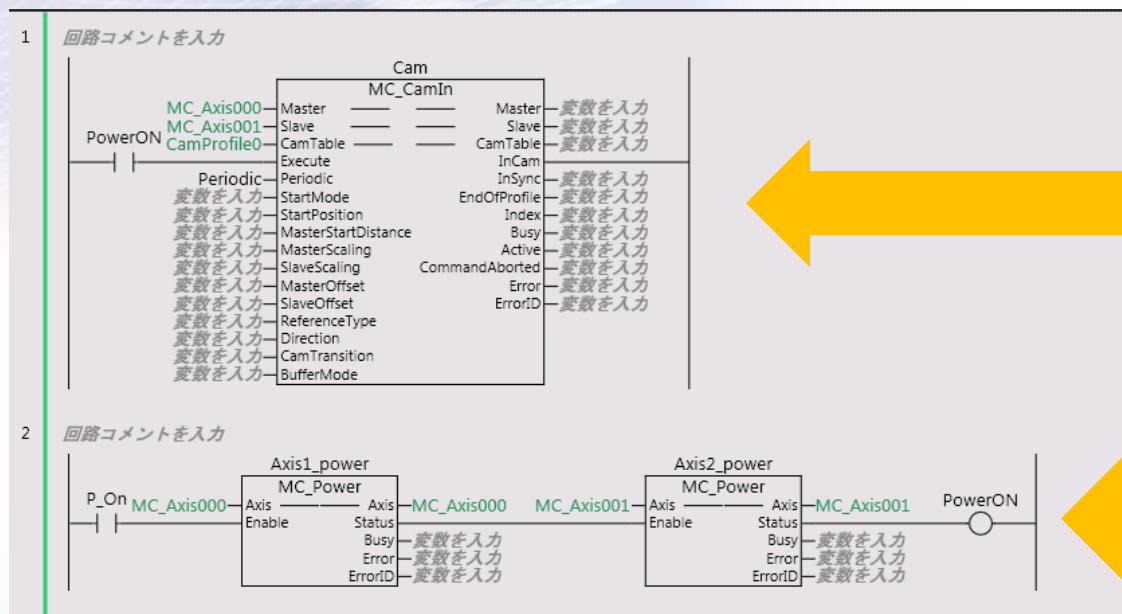
为以防万一、先编译一下。



- 6 编辑完凸轮数据后、在编程窗口的上方有变量表、右击登录在外部变量里的凸轮的上方、将会做[更新]。

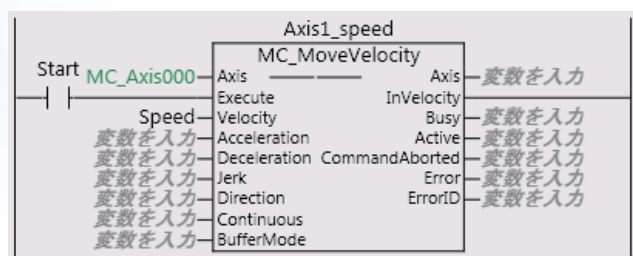


凸轮动作的程序、只要这些



电子凸轮

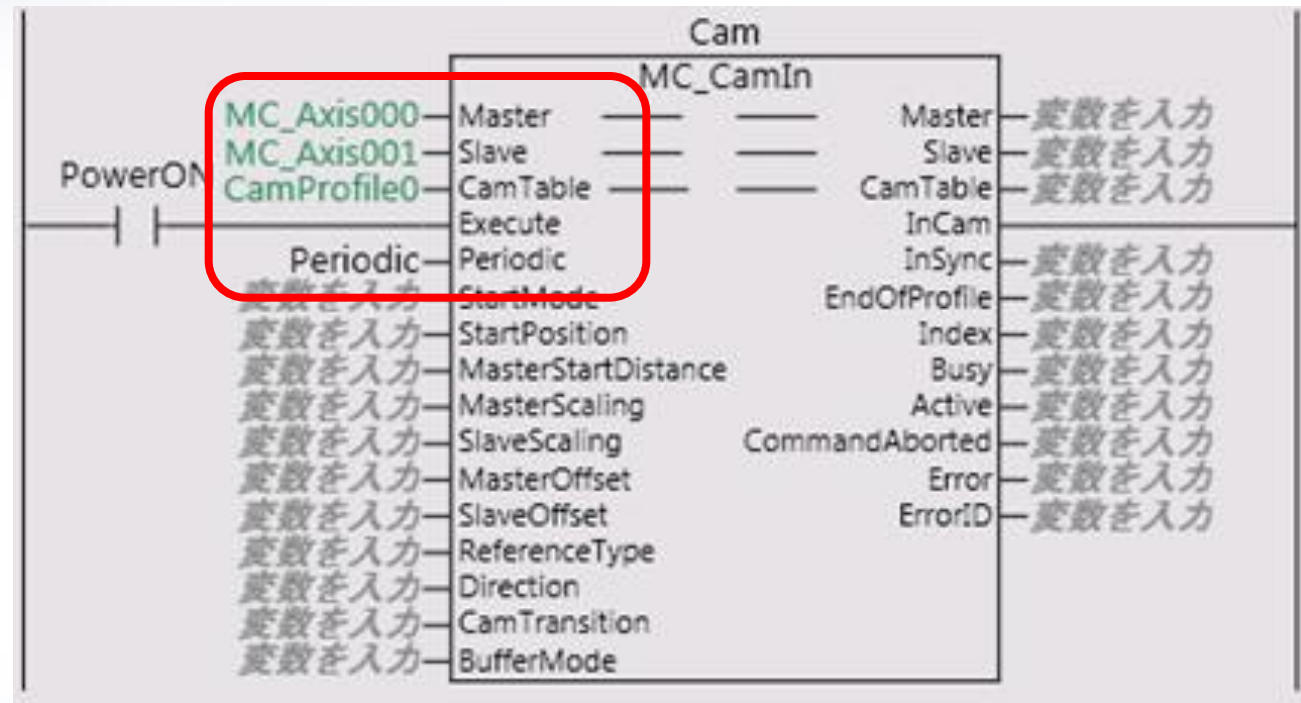
伺服锁定



用速度控制让主轴运行

凸轮动作的程序、只要这些

- 主轴
- 从轴
- 凸轮模型
- 连续动作的设定
(Periodic = 用True做变量登录)



四、IO的追加也很简单·且无错误！

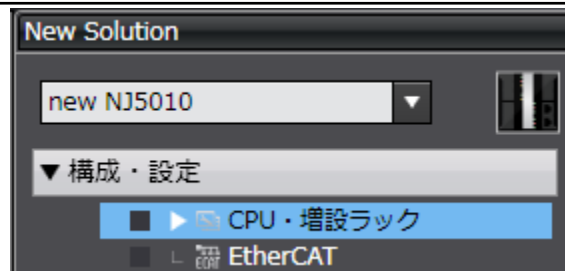
CPU机架的构成・设定

将NJ系列CPU・扩展机架上安装的单元构成在Sysmac Studio上进行制作、并进行高性能单元的设定。
Sysmac Studio、可以用组建实际的机器那样的感觉来构筑单元构成。

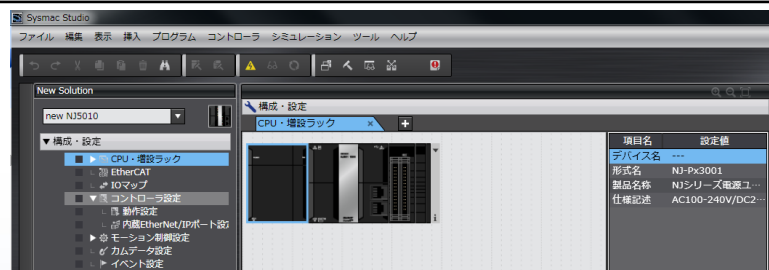
对CPU・扩展机架的设定方法进行说明。

1 点击多视角浏览器的[构成・设定]、打开菜单。

2 双击[CPU・扩展机架]（或者、右击[CPU・扩展机架]选择[编辑]。）。



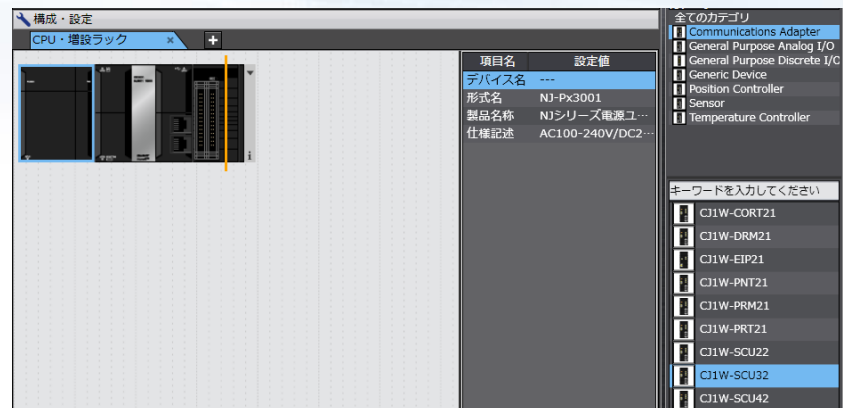
3 在编辑窗口内的[构成・设定]层上会显示「CPU・扩展机架」。



从今以后 : 直观的、不易出错的。

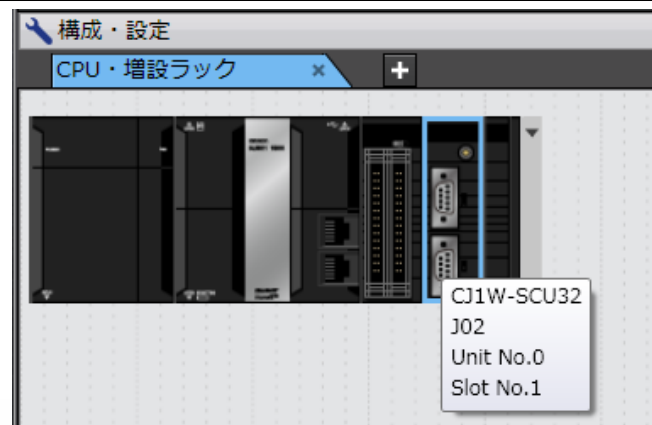
对单元的插入操作的步骤进行说明。

- 1 在工具箱内的机种选择窗口里选择单元 (SCUユニット) 并拖动、在 [CPU・扩展机架] 层上进行下拉。



- 2 单元被插入。

CJ里是做
不了的



从今以后 : 变更也变得简单了。

对单元的机种变更的步骤进行说明。

- 1 机种有了变更的单元，右击（[CPU・扩展机架] 层上）选择[型号变更]。



- 2 会显示 [单元型号变更窗口] 。



- 3 选择单元、点击[OK]。所选择的机种就会变更了。

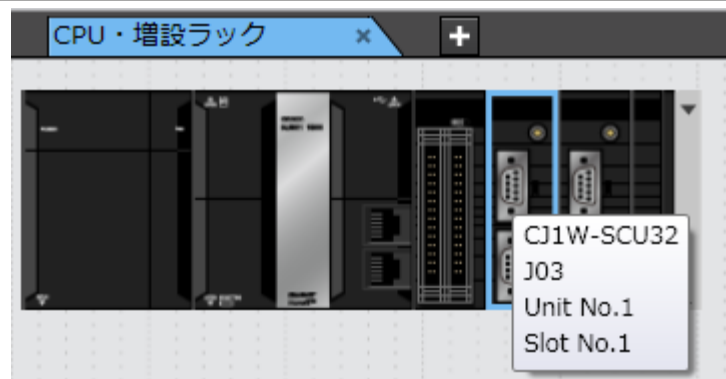
从今以后：单元追加也只需复制和粘贴，简单！

<参考>单元的复制／粘贴操作步骤

1 右击要复制的单元，选择[复制]。



2 右击要插入位置的单元选择[粘贴]。单元就会被粘贴了。



在CJ里是
做不了的

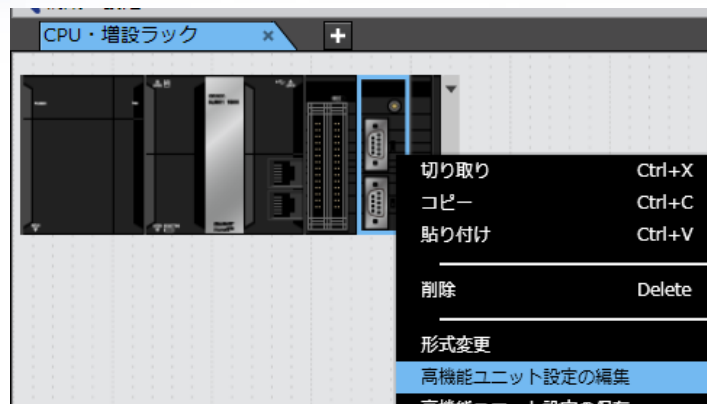


从今以后：参数编辑也能当场马上完成。

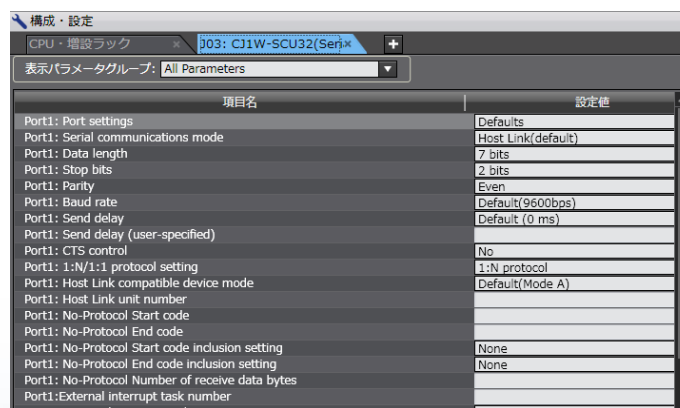
对高性能单元进行设定。

- 1 在 [CPU・扩展机架] 层上双击对象单元。

（右击，选择[高性能单元设定的编辑]，也能完成同样的事情）



- 2 会显示所选择单元的[参数的编辑]对话框。



- 3 进行各项目的设定、点击 [OK] 。

从今以后：变量也好、IO分配也好、都更为简单。

＜自动生成设备变量时＞

生成设备变量。

设备变量自动生成时的变量名为、「设备名」+「I/O端口名」。

- 1 在多视角浏览器中双击 [IO表] 。
- 2 选择 [IO表] 层上的单元的1个或者多个 I/O端口、右击，选择 [设备变量的生成] 。
- 3 对单元的I/O端口、会自动分配设备变量。
会登录在[范围]中指定的变量表里。

位置	ポート	説明	R/W	データ型	変数	変数コメント
▼ CPU・増設ラック						
▼ CPUラック 0						
▼ CJ1W-MD263 (24V DC Input)						
▼ Ch1_Out	出力CH1		RW	WORD		
	Ch1_Out00	出力CH1接点00	RW	BOOL		
	Ch1_Out01	出力CH1接点01	RW	BOOL		
	Ch1_Out02	出力CH1接点02	RW	BOOL		
	Ch1_Out03	出力CH1接点03	RW	BOOL		
	Ch1_Out04	出力CH1接点04	RW	BOOL		
	Ch1_Out05	出力CH1接点05	RW	BOOL		
	Ch1_Out06	出力CH1接点06	RW	BOOL		
	Ch1_Out07	出力CH1接点07	RW	BOOL		

位置	ポート	説明	R/W	データ型	変数	変数コメント	スコープ
▼ CPU・増設ラック							
▼ CPUラック 0							
▼ CJ1W-MD263 (24V DC Input)							
▼ Ch1_Out	出力CH1		RW	WORD	MD_00_Ch1_0		グローバル変数
	Ch1_Out00	出力CH1接点00	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out01	出力CH1接点01	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out02	出力CH1接点02	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out03	出力CH1接点03	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out04	出力CH1接点04	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out05	出力CH1接点05	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out06	出力CH1接点06	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out07	出力CH1接点07	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out08	出力CH1接点08	RW	BOOL	MD_00_Ch1_0		グローバル変数
	Ch1_Out09	出力CH1接点09	RW	BOOL	MD_00_Ch1_0		グローバル変数

「设备名」の默认如下。

- ・ 从站的情况、「E」+「从01开始的连号」
- ・ 单元的情况、「J」+「从01开始的连号」

在CJ里是做不了的

五、程序，变得更加易懂，更加不容易出错

- 条件分支在梯形图里比较难懂、还会变长
- 构造体？不能用的吧？

制作更加易懂的程序！

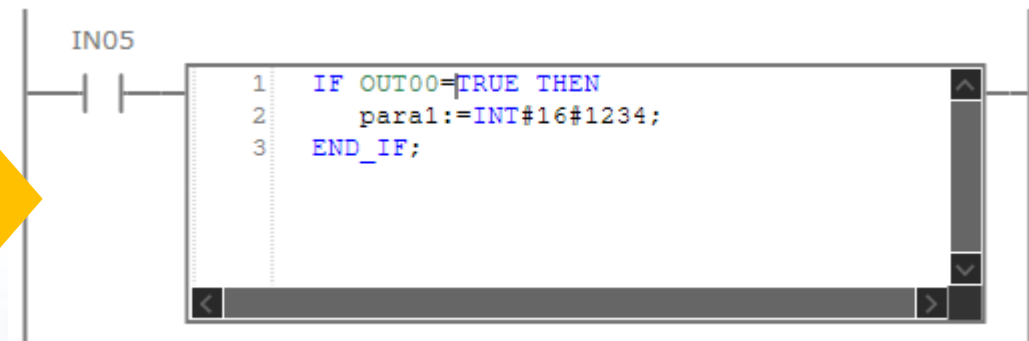
目前为止

想根据通道及DM的值改变处理时、只能将比较命令多个排列起来。**会增加行数、不易看懂。**

从今以后

通过内联ST、变得易懂了、可以记述易懂的条件。

在CJ里是做不了的



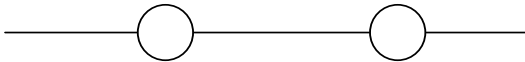
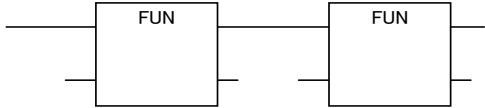
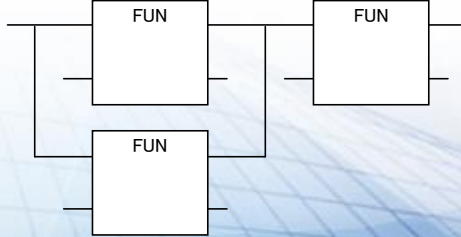
目前为止 以往梯形图的问题点：不易读

输出无法连续。因此、只能把回路分开将动作连接起来才行、
越是复杂的动作、越是不易读、还会成为行数很多的程序。

从今以后

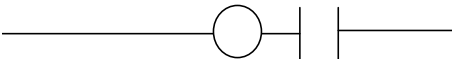
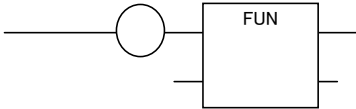
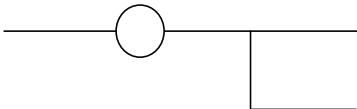
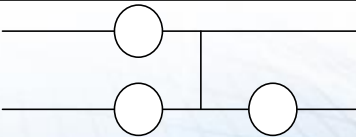
动作连接得起来、易读、成为容易明白的程序。

在CJ里是
做不了的

内容	示意
○ OUT 命令	
○ FUN/FB	
○ FUN/FB	

这些是不行的。

OUT命令的连接

内容	示意
✗ OUT命令里连接LD命令	
✗ OUT命令里连接FUN/FB	
✗ OUT命令里做后分支	
✗ OUT命令后面进行合流	

通过数组、构造体、程序变得易读清晰

目前为止 处理配方及DB用的数组，是1次元。实质不能使用。。

一次元

温度	寸法	温度	寸法	...
100	48	100	48	...

从今以后 通过活用3次元数组及构造体、提高了程序的易读性・生产性！配方管理也方便了

二次元

商品コード	温度	寸法
11	100	48
12	75	48
15	75	48
18	75	48
...	...	...

三次元

商品コード	温度	寸法
11	100	48
12	75	48
15	75	48
18	75	48
...	...	...

在CJ里是做不了的



构造体

構造体と配列＝レシピ、データベースなど

データ番号(数値)	商品コード(文字列)	形名(文字列)	温度(実数)	寸法
1	145A	E5CS-J	100	48
2	148B	E5CS-E	75	48
3	149B	E6CS-E	75	48
4	150B	E7CS-E	75	48
...	...	...	...	...

通过数组、构造体、程序变得易读清晰

目前为止 处理配方及DB用的数组，是1次元。实质不能使用。。

从今以后 通过活用3次元数组及构造体、提高了程序的易读性・生产性！配方管理也方便了。

派生数据型	CJ2系列	NJ系列
<u>数组</u>	用内部变量或者输入输出变量 仅能使用一次元数组 最大尺寸为2048。 (REAL/DINT/UDINT/DWORD:1024) (LREAL/LINT/ULINT/LWORD:512)	通过输入变量、输出变量、输入输出变量、内部变量、外部变量可以使用到三次元数组 最大尺寸为65535。
<u>构造体</u>	可以用内部变量或者输入输出变量来使用	可以用输入变量、输出变量、输入输出变量、内部变量、外部变量来使用
<u>枚举型</u>	在CJ里做 不了的	可以用输入变量、输出变量、输入输出变量、内部变量、外部变量来使用
<u>共用体</u>		[单元Ver.1.0] 可以用内部变量、外部变量来使用 [单元Ver.1.1] 可以用输入变量、输出变量、输入输出变量、内部变量、外部变量来使用

ST、终于能够使用了！

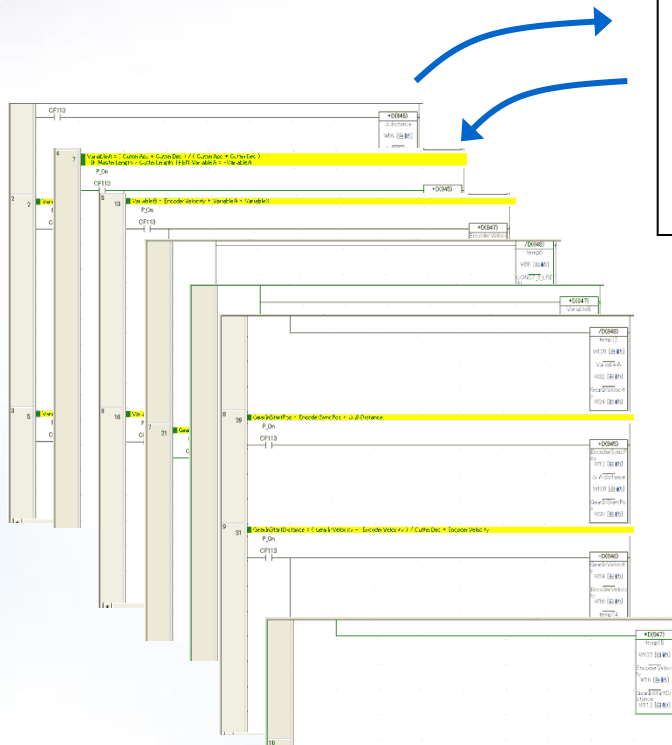
目前为止 ST会比梯形图慢、还会增加内存消耗，所以不好用…

从今以后 梯形图、ST的差别不再有了。通过易读、易用的表述来提高生产性！

	CJ2系列	NJ系列
<u>内存使用量</u>	与梯形图相比、会增加。	ST、梯形图基本没有差异。
<u>实行速度</u>	与梯形图相比、会变慢。	ST、梯形图基本没有差异。
<u>梯形图程序中的ST</u>	无法使用。	通过输入变量、输出变量、输入输出变量、内部变量、外部变量可以使用

使用ST语言时的效果试算

在旋转切刀控制里使用的移动量计算处理，作为基准对象，进行了效果的测定。



LD:6.25页

```

αβDistance := αDistance + βDistance;
VariableX := ( MasterLength - αβDistance ) / EncoderVelocity;
VariableY := CutterLength - αβDistance;
VariableA := ( CutterAcc + CutterDec ) / ( CutterAcc * CutterDec );
IF MasterLength > CutterLength THEN
    VariableA := -VariableA;
END_IF;
VariableB := EncoderVelocity * VariableA + VariableX;
VariableC := EncoderVelocity * EncoderVelocity * VariableA / 2.0 + VariableY;
GearInVelocity := ( VariableB - SQRT ( VariableB * VariableB - ( 2.0 * VariableA * VariableC ) ) ) /
    VariableA;
GearInStartPos := EncoderSyncPos + αβDistance;
GearInStartDistance := ( GearInVelocity - EncoderVelocity ) / CutterDec * EncoderVelocity;
    
```

原先的运算公式

```

P.24
Q.1
Q.2
Q.3
Q.4
Q.5
Q.6
Q.7
Q.8
Q.9
Q.10
Q.11
Q.12
Q.13
Q.14
Q.15
Q.16
Q.17
Q.18
Q.19
Q.20
Q.21
Q.22
Q.23
Q.24
Q.25
Q.26
Q.27
Q.28
Q.29
Q.30
Q.31
Q.32
Q.33
Q.34
Q.35
Q.36
Q.37
Q.38
Q.39
Q.40
Q.41
Q.42
Q.43
Q.44
Q.45
Q.46
Q.47
Q.48
Q.49
Q.50
Q.51
Q.52
Q.53
Q.54
Q.55
Q.56
Q.57
Q.58
Q.59
Q.60
Q.61
Q.62
Q.63
Q.64
Q.65
Q.66
Q.67
Q.68
Q.69
Q.70
Q.71
Q.72
Q.73
Q.74
Q.75
Q.76
Q.77
Q.78
Q.79
Q.80
Q.81
Q.82
Q.83
Q.84
Q.85
Q.86
Q.87
Q.88
Q.89
Q.90
Q.91
Q.92
Q.93
Q.94
Q.95
Q.96
Q.97
Q.98
Q.99
Q.100
    
```

ST:0.25页

六、通过仿真、事前确认动作

时序和运动的连动仿真

一次点击就能开始时序和运动的仿真。
仿真结果可以用各种视图来显示。

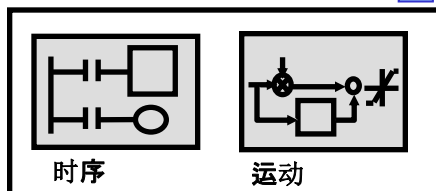
编程失误、齿轮比·加速度的位数·单位等设定错误的发现、
可以减少因调试时的单纯错误而引起的机械破损风险。

通过运动仿真(2D/3D/3D模式)
可以进行视觉的运作确认

一次点击

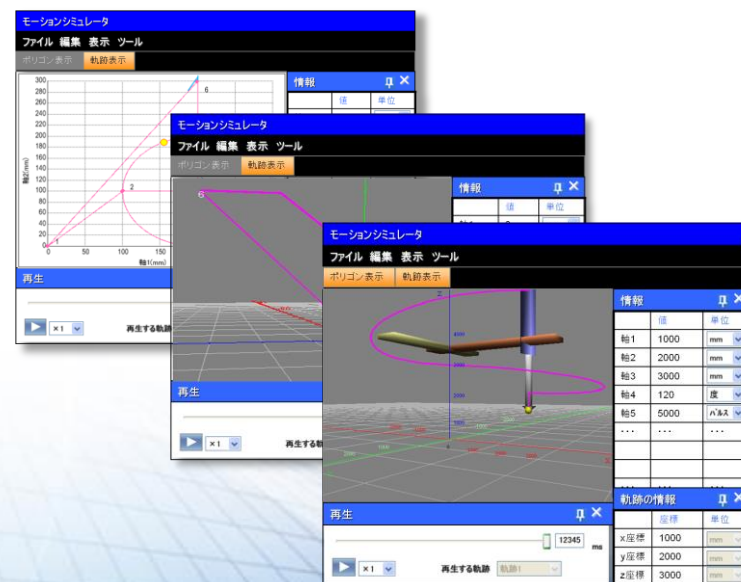


NJ仿真



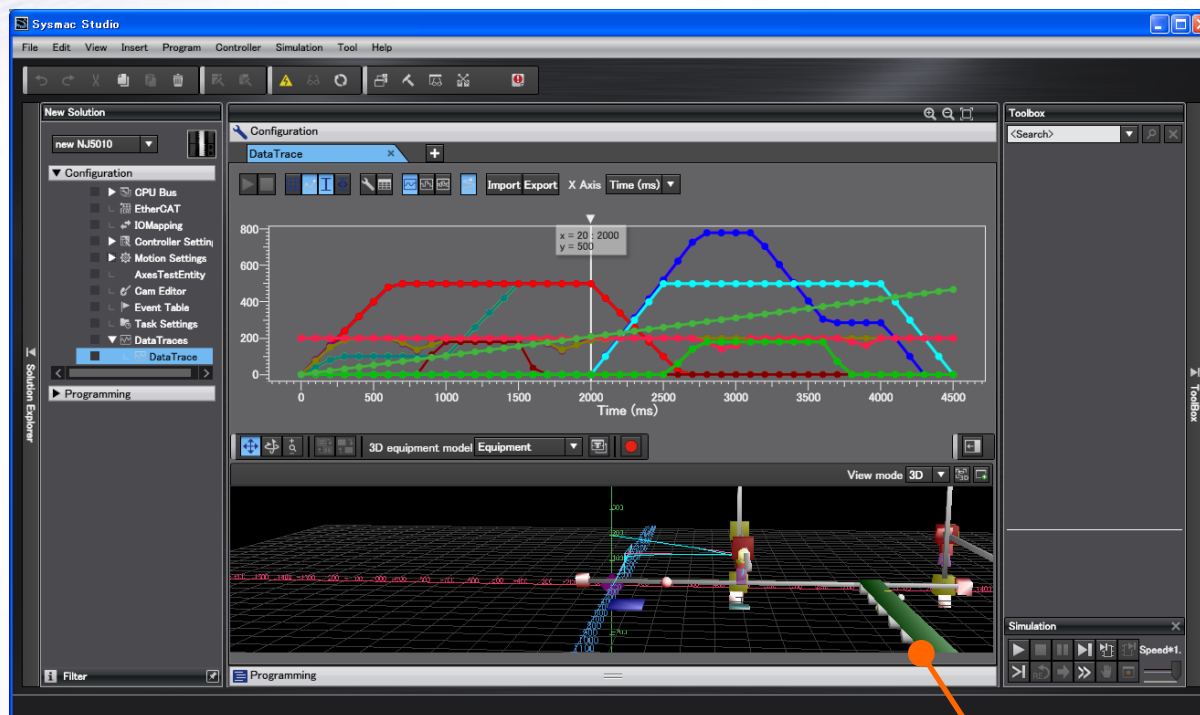
录制·播放
(+同时播放)

数据追踪



运动指令值和实贯值的比较

通过运动指令值和实测值的重叠书写和时间图的同步显示，可以让调试更为有效率。(逻辑仿真和实机动作的补偿)



EtherCAT上的I/O、驱动器的动作实绩也能在数据追踪里确认。
·可以在NJ5的循环时间里采样

数据追踪结果和虚构机械的动作做同时确认

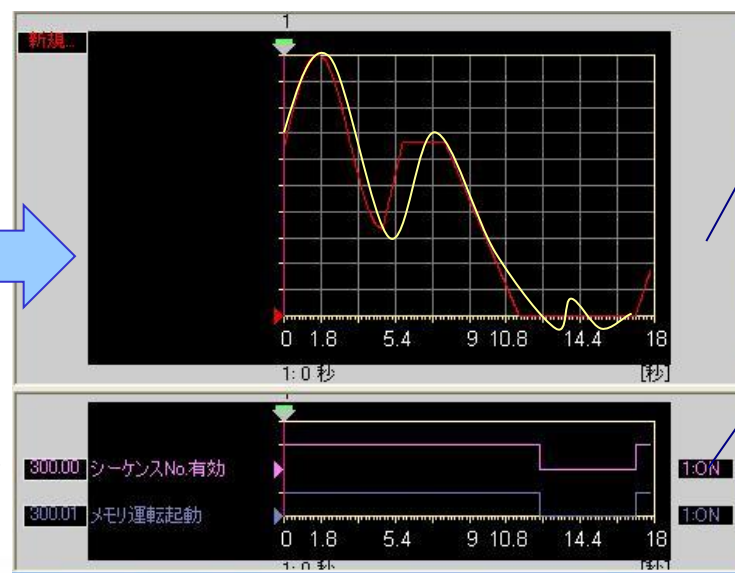
运动指令值和实绩值的比较

通过运动指令值和实测值的重叠书写和时间图的同步显示，可以让调试更为有效率。（逻辑仿真和实机动作的补偿）

表示/非表示		名前	IXオフセット	IYオフセット
☒	黄色	MC_Axis000.Cmd.Pos	0	0
☒	青色	MC_Axis000.Act.Pos	0	0
☒	蓝色	MC_Axis001.Cmd.Pos	0	0
☒	红色	MC_Axis001.Act.Pos	0	0

包含从伺服过来的反馈值的运动相关的状态，可作为追踪对象从清单里简单地进行选择。

インポジ信号
外部信号



运动控制器的指示
和驱动器的实测值
的比较

从时序过来的触发
信号

可以一边确认指令值和实际值的偏差、以及与外部信号的时间一边进行调整。
在仿真结果上、可以将实机动作的伺服位置信号、近信号、甚至接点的ON/OFF等外部信号在同一个图表里显示，所以调试时候所需花费的时间会有很大程序的缩短。（根据客户的调研结果）

调试功能

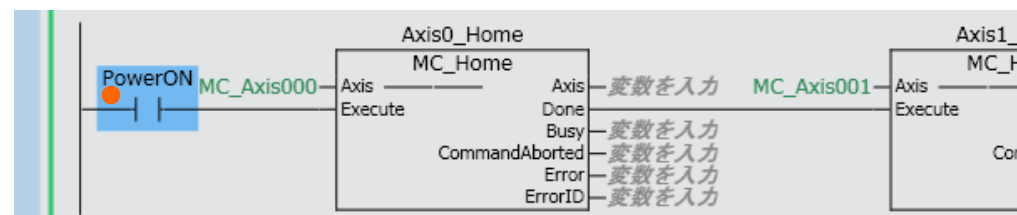
让时序、运动控制同步可以实行停止、步进。
在想要看高速控制的区域停止，通过数据追踪及2D・3D查看、可以确认当时的状态。

【步进实行】…可以一边将程序一步步地实行一边确认结果。

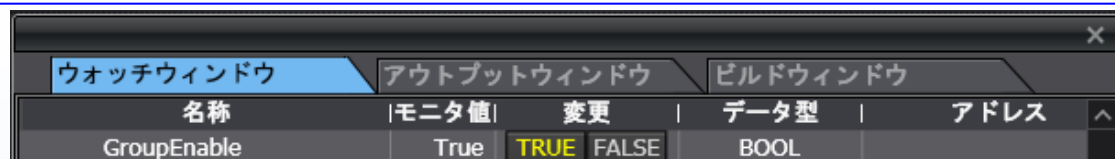
```

1  varA:=13;
2  varB:=49;
3  IF varA < DINT#1 THEN
4    varA:=DINT#100;
5  END_IF;
6  IF varA > DINT#12 THEN
7    varA:=DINT#200;
8  END_IF;
9  FOR varB:= 0 TO 2 BY 1 DO
10   varA:=varB;
11 END_FOR;
12

```



【监视窗口】…提供将外部输入以手动进行On/Off的功能。



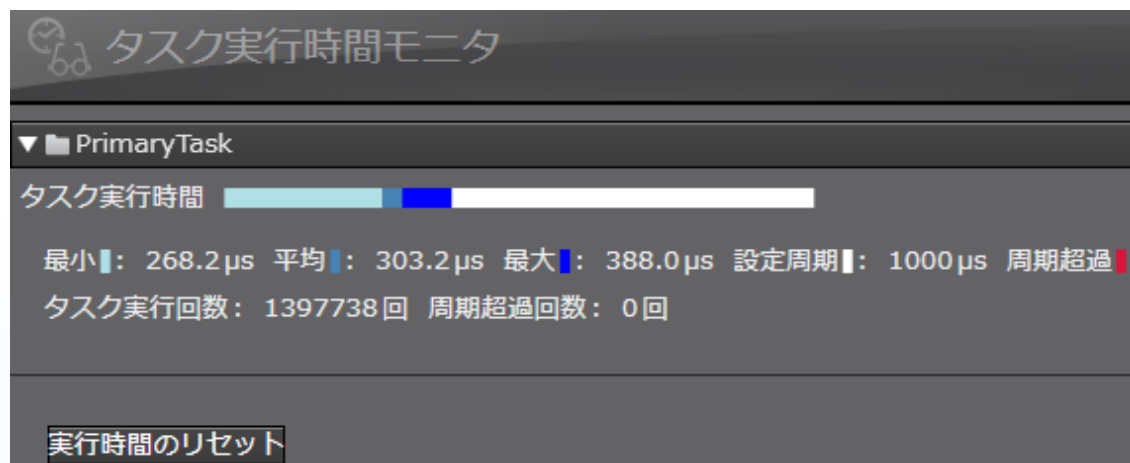
【调试程序任务】…可以制作调试・程序用的特殊任务。

- 调试任务是在仿真里实行的，可以设定为在实机里不执行。
- 仿真的循环时间计算功能里，除去了调试任务的实行时间。

实行时间的仿真

每个任务、从任务的实行开始到实行结束所花费的时间、可以在仿真里进行近似预测。

⇒ 首要定周期任务，在时序和运动的同步实行之外，还与EtherCAT的循环时间同步。因此、通过在仿真里进行循环时间的近似计算、可以推测系统的性能。



- 也加上其他任务以及系统服务的中断时间、显示预测值。
- 可以确认程序中的条件分支等引起的实行时间变动。
 - ⇒计测最小值、平均值、最大值。
 - ⇒可以确认哪个任务发生了超周期。

仿真的开始

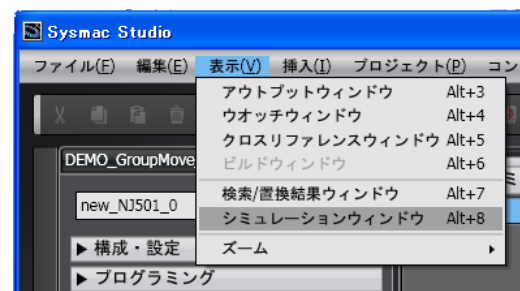
综合仿真

同时开始统合了时序和运动的综合仿真。

对开始仿真的方法进行说明。

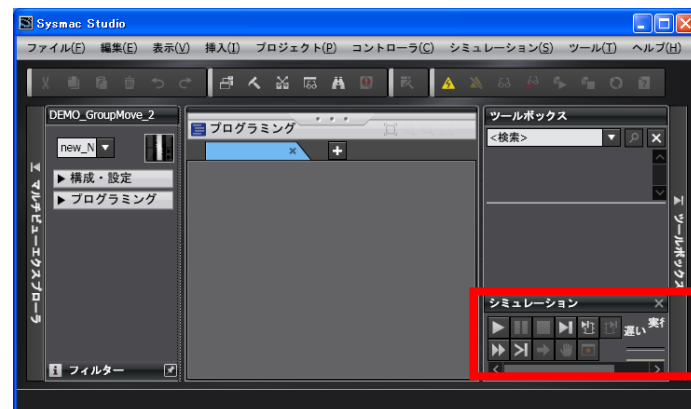
- 1 在菜单栏里、选择 [显示] - [仿真窗口]。

※已经显示「仿真窗口」时，不需要这个操作。



- 2 [仿真窗口] 会追加在右下方。

※ 在显示 [仿真窗口] 后、请先把程序进行「生成」。



- 3 点击 [仿真窗口] 的播放按钮。

1次点击



数据追踪

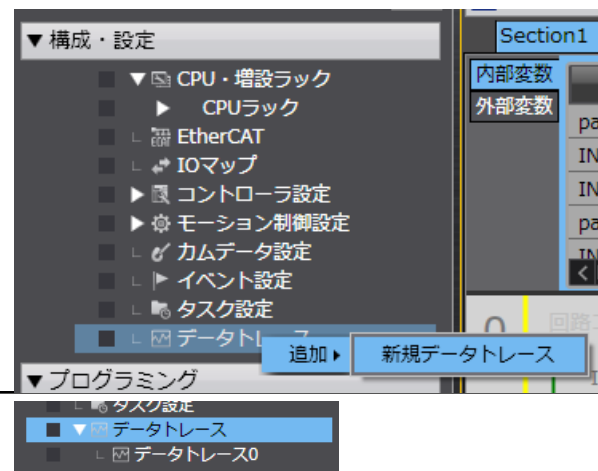
数据追踪

对对象变量进行采样、在追踪内存里无程序地进行保存的功能。

NJ5(实机)、在仿真的同时也能使用同等的功能。

对起动数据追踪的方法进行说明。

- 1 右击多视角浏览器的[构成・设定] - [数据追踪]、选择[追加] - [新建数据追踪]。



- 2 [数据追踪0] 会被追加。

- 3 双击多视角浏览器的[构成・设定] - [数据追踪] - [数据追踪0]。

- 4 在[构成・设定]层内会显示数据追踪的设定画面。
(参照下页)

数据追踪

数据追踪的开始

数据追踪手动触发

视图的追加删除

追踪数据的播放

实时的数据显示

追加追踪对象变量

表示/非表示	名前	Xオフセット	Yオフセット	軸の最小値	軸の最大値	最小	最大	平均	カー
☒	MC_Axis000.Cmd.Pos	0	0	0	0	0	200000	86418.4949579214	0
☒	MC_Axis000.Act.Pos	0	0	0	0	0	200000	86418.0869	0
☒	MC_Axis001.Cmd.Pos	0	0	0	0	0	200000	59326.8676589193	0
☒	MC_Axis001.Act.Pos	0	0	0	0	0	200000	59326.6206	0

時間 (ms)

数据追踪

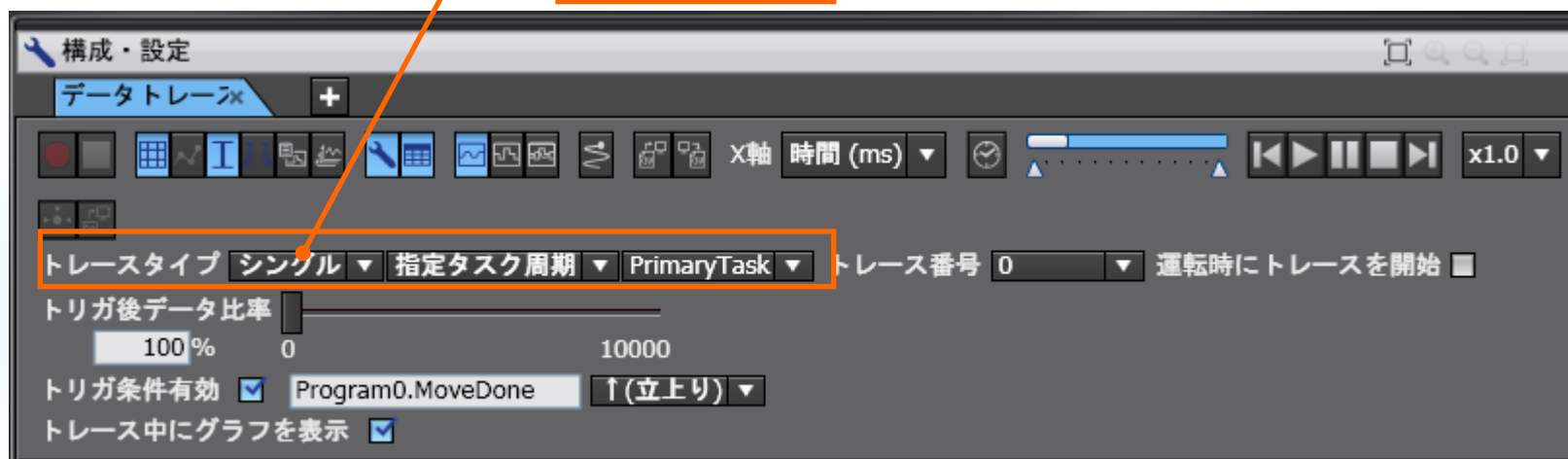
触发追踪和连续追踪

对对象变量进行采样、在追踪内存里无程序地进行保存的功能。

设定触发条件、启示条件成立前后的数据为触发追踪。无触发连续进行采样、将结果按顺序记录在电脑上的文件里的是连续追踪。可以选择这2种追踪方式。

触发追踪时、可以在Sysmac Studio上进行数据的确认，以及保存文件。

追踪类型的
变更



机械机构的选定

3D装置模型的编辑

在选择的3D装置模型里分配轴变量。

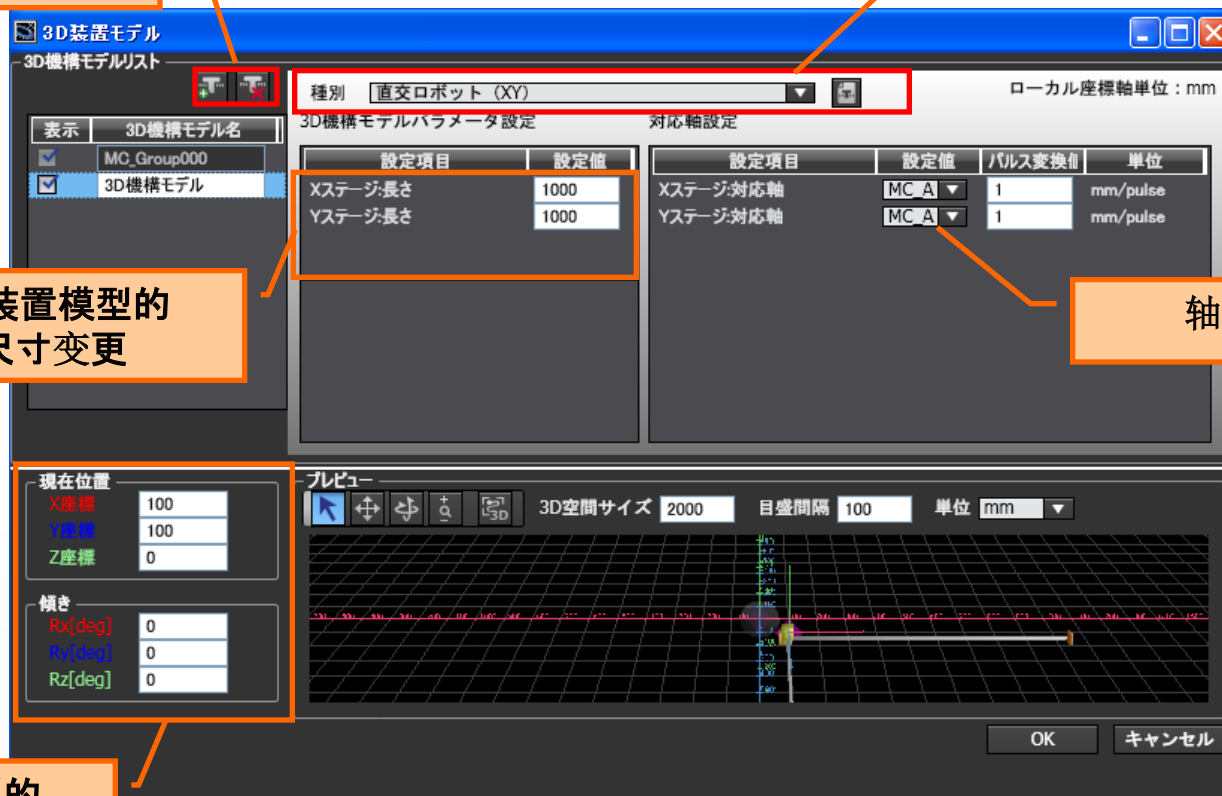
3D装置模型的
追加・删除

3D装置模型的变化

3D装置模型的
尺寸变化

轴变量的分配

3D装置模型的
配置变化



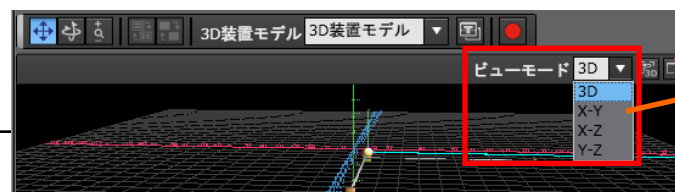
2D/3D视图和实时描绘

3D运动监控

3D运动监控、可以追加视图、切换2D/3D显示，将仿真的结果通过各种不同的观点进行验证。

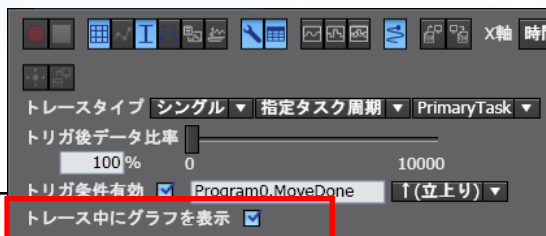
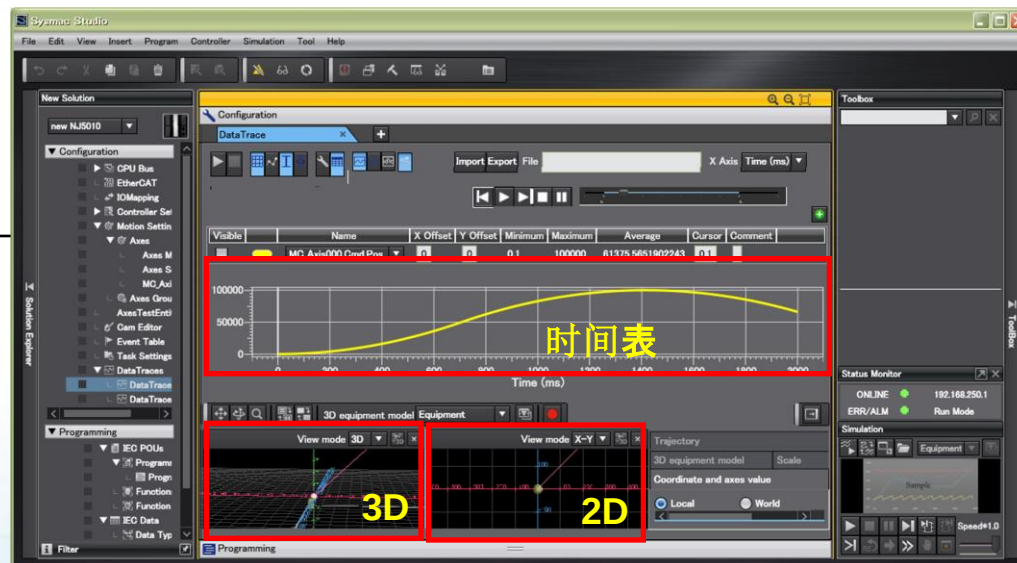
对实时显示仿真结果的方法进行说明。

- 1 可以在3D运动监控里追加视图、切换2D、3D显示。
- 2 这个动作时，数据追踪的时间表形式的图表也会连动。
- 3 要与仿真同步，实时描绘时、在〔追踪中显示图表〕里打上勾。



视图的追加

显示形式的选择



运动指令值和实绩的比较

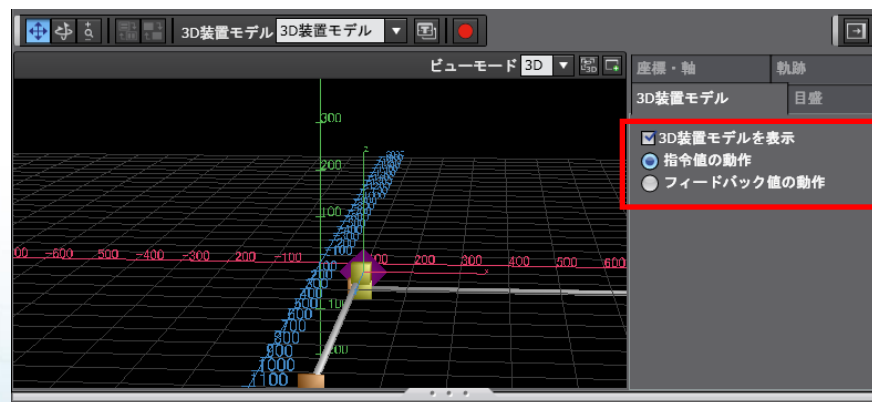
比较运动指令值和实绩值。

运动指令值和实绩值、可以通过2D/3D上的重叠书写来进行比较。

对开始仿真的方法进行说明。

- 1 选定3D装置模型。
- 2 在3D装置模型里分配轴变量。
- 3 3D装置模型里分配好的轴变量会自动地作为追踪对象变量登录。
 - [MC_Axis***.Cmd.Pos]是指令值
 - [MC_Axis***.Act.Pos]是实绩值
- 4 在3D描绘里、可以指定3D装置模型是由指令值、还是实绩值的哪一个来驱动运行。

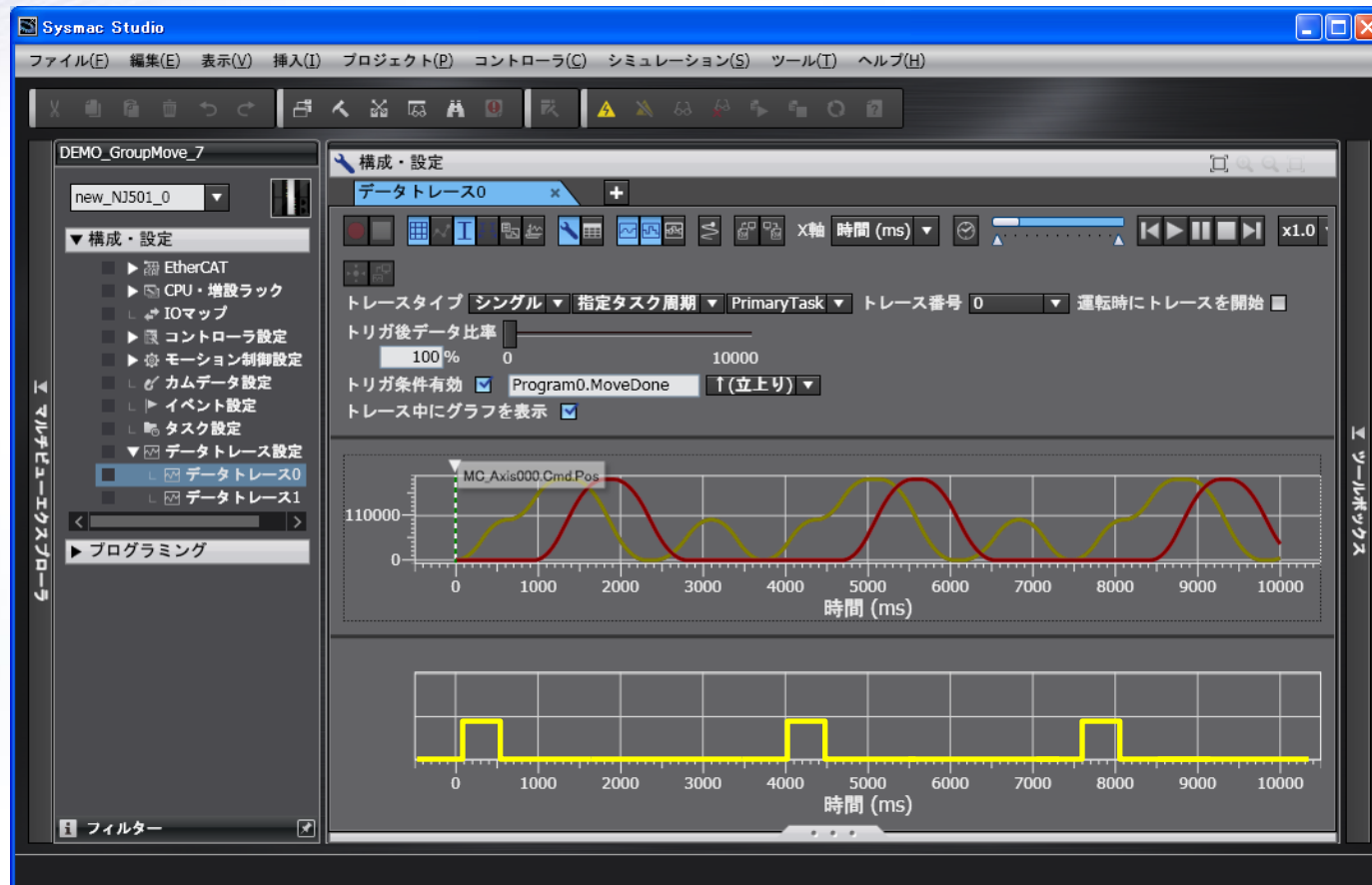
表示/非表示		名前	IXオフセット	IYオフセット
☒	黄色	MC_Axis000.Cmd.Pos	0	0
☒	青色	MC_Axis000.Act.Pos	0	0
☒	蓝色	MC_Axis001.Cmd.Pos	0	0
☒	红色	MC_Axis001.Act.Pos	0	0



运动指令值和实绩的比较

运动指令值和实绩在时间表上进行比较。

在时间表上也能进行波形的重叠、将加速度的图表和时序过来的触发（时间表）作比较后来调整运动的动作及间隔时间。



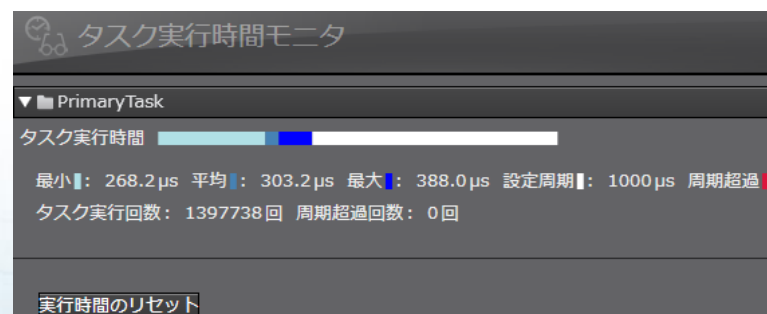
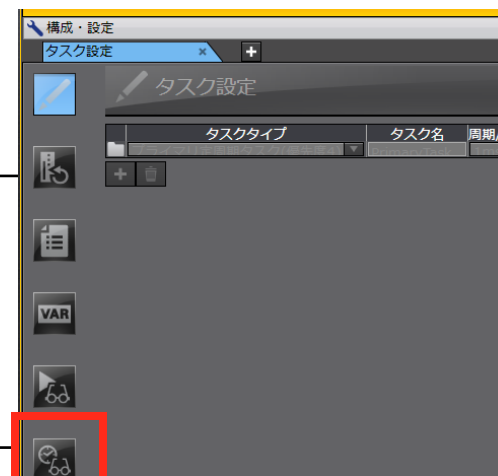
程序实行时间的预测

任务实行时间监控

NJ系列控制器以及仿真里实行程序时、可以对不同的程序实行单位的任务、进行实行时间的监控。也能监控仿真连接时任务实际的处理时间。通过使用这个功能、可以进行控制器的性能验证。

显示任务实行时间监控。

- 1 双击多视角浏览器的[构成・设定] - [任务设定]。
- 2 在编辑窗口里显示的编辑画面上、点击[任务实行时间监控]。
- 3 显示任务实行时间监控。



控制器和仿真功能的区别

调试功能	NJ5	仿真
监控/微分监控、监视窗口下的监控、IO 表视图的监控	○	○
接点的设定/复位、接点的强制值更新复位/取消/解除、数据的当前值变更、定时器/计数器的设定值变更	○	○
交叉参考弹出、交叉参考报告	○	○
在线编辑	○	○
数据追踪、触发的设定、采样变量的设定、追踪的开始·停止、追踪结果的显示、追踪结果的保存	○	○
假想机构的设定	○	○
时间表形式的显示、3D形式的轨迹显示	○	○
任务实行时间的监控	○	○
异常信息监控、异常履历显示、用户事件(事件表)设定、用户内存使用量显示	○	○
信息的监控功能、控制器信息监控、任务实行状态监控、轴状态监控、动作模式的变更、控制器的复位	○	×
控制器时钟设定	○	×
内存全清除、访问权的强制开放、控制器的复位、动作模式的变更	○	×
故障排除、控制器信息监控	○	×
实行处理时间预测、程序的仿真调试、仿真对象的设定、仿真的实行速度的变更、断点的设定、步进实行	×	○

最后、面向客户

NJ+SysmacStudio的诞生

与目前为止不同的方便性、简单性。

欧姆龙可以提供可对客户的设备进行革新的控制器+环境了

这个变化、对客户来说、最重要的是

有了时间，可以来思考设备·机械的动作以及对下一世代设备的革新方法

所以、面向客户。

必定、也在等待着大家的提案。

OMRON

SYSTMAC
always in control

Automation Systems