

プログラマブル多軸モーションコントローラ

アプリケーションガイド レーザアプリケーション編

形CK5M-CPU1□1

形CK3M-CPU1□1

形CK3W-AX2323□

形CK3W-GC2200

Application
Starting
Guide

おことわり

- (1) 本マニュアルの内容の一部または全部を無断で複写、複製、転載することを禁じます。
- (2) 本マニュアルの内容に関しては、改良のため予告なしに仕様などを変更することがあります。
あらかじめご了承ください。
- (3) 本マニュアルの内容に関しては、万全を期しておりますが、万一ご不審な点や誤りなど、お気づきのことがありましたら、お手数ですが、弊社支店または営業所までご連絡ください。
その際、巻末記載の**Man.No.** (マニュアルNo.) も併せてお知らせください。

商標

- Microsoft、Windows、Excel、Visual Basic、Microsoft Edgeは、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。
- EtherCAT®は、ドイツBeckhoff Automation GmbHによりライセンスされた特許取得済み技術であり登録商標です。

その他、本文中に掲載している会社名および製品名は、それぞれ各社の商標または登録商標です。

著作権

- スクリーンショットはマイクロソフトの許諾を得て使用しています。
- 本製品にはサードパーティ製のソフトウェアが組み込まれています。このソフトウェアに関連するライセンスと著作権については、http://www.fa.omron.co.jp/nj_info_j/ をご覧ください。

目次構成

		1
1	本資料の概要	2
2	対象機器とデバイス構成	3
3	接続手順	4
4	各種設定のカスタマイズ方法	

目次

目次構成.....	3
注意事項.....	6
図記号の説明.....	6
関連マニュアル.....	7
用語と定義.....	8
改訂履歴.....	9

第 1 章 本資料の概要

1-1 概要.....	1-2
-------------	-----

第 2 章 対象機器とデバイス構成

2-1 デバイス構成.....	2-2
-----------------	-----

第 3 章 接続手順

3-1 作業の流れ.....	3-2
3-2 コントローラの設定準備.....	3-4
3-2-1 新規プロジェクトの作成.....	3-4
3-2-2 コントローラの初期設定.....	3-5
3-3 MOTF の設定.....	3-7
3-3-1 プログラミング.....	3-7
3-3-2 フィルタ係数の決定.....	3-12
3-4 モータとエンコーダの設定.....	3-21
3-4-1 各機器の配線.....	3-21
3-4-2 プログラミング.....	3-22
3-4-3 電流ループのチューニング.....	3-26
3-4-4 フェーズリファレンスの確立.....	3-28
3-4-5 オープンループテスト.....	3-29
3-4-6 位置ループのオートチューニング.....	3-30
3-4-7 位置ループのインタラクティブチューニング.....	3-31
3-4-8 Y 軸のチューニング.....	3-35
3-4-9 チューニング結果の確認.....	3-35
3-5 オーバトラベルリミットスイッチと原点の設定.....	3-38
3-5-1 各機器の配線.....	3-38
3-5-2 プログラミング.....	3-38
3-5-3 設定の確認（オーバトラベルリミットスイッチ）.....	3-40
3-5-4 原点の設定.....	3-41
3-6 ガルバノスキャナの設定.....	3-43
3-6-1 各機器の配線.....	3-43
3-6-2 プログラミング.....	3-44
3-6-3 コントロールコマンドの設定.....	3-46
3-6-4 設定の確認.....	3-48
3-6-5 MOTF の実機検証.....	3-49
3-7 レーザ発振器の設定.....	3-50

3-7-1	各機器の配線	3-50
3-7-2	ガイドレーザの出力	3-50
3-7-3	プログラミング	3-52
3-7-4	レーザ発振器の設定	3-55
3-7-5	設定の確認	3-56

第 4 章 各種設定のカスタマイズ方法

4-1	globaldefinitions.pmh	4-2
4-2	MotionOnTheFly.pmh	4-3
4-3	MotorControl.pmh	4-4
4-4	SensorControl.pmh	4-7
4-5	Galvano.pmh	4-8
4-6	LaserControl.pmh	4-10
4-7	TCRConfigurations.pmh	4-11

注意事項

- 実際のシステム構築に際しては、システムを構成する各機器・装置の仕様をご確認のうえ、定格・性能に対し余裕を持った使い方をし、万一故障があっても危険を最小にする安全回路などの安全対策を講じてください。
- システムを安全にご使用いただくため、システムを構成する各機器・装置のマニュアルや取扱説明書などを入手し、「安全上のご注意」「安全上の要点」など安全に関する注意事項および「使用上の注意」を含め、マニュアルや取扱説明書に記載の内容を確認し理解したうえで使用してください。
- システムが適合すべき規格・法規または規制に関しては、お客様自身でご確認ください。
- 本資料の一部または全部を、オムロン株式会社の許可なしに複写、複製、再配布することを禁じます。
- 本資料の記載内容は、改良のため予告なく変更されることがあります。

本資料で使われているマークには、次のような意味があります。



使用上の注意

製品の動作不能、誤動作、または性能・機能への悪影響を予防するために実施または回避すべきことを示します。



参考

必要に応じて読んでいただきたい項目です。
知っておくと便利な情報や、使用するうえで参考となる内容について説明しています。

図記号の説明



- 記号は、強制を意味しています。
具体的な内容は、●の中と文章で示します。
左図の場合は、「一般的な強制事項」を表します。

関連マニュアル

システムを安全にご使用いただくため、システムを構成する機器・装置のマニュアルや取扱説明書などを必ず入手し、「安全上の注意」「安全上の要点」など安全に関する注意事項、および「使用上の注意」を含め、マニュアルや取扱説明書に記載の内容を確認し理解したうえ使用してください。
 オムロン株式会社（以下、オムロン）、米国デルタ タウ データ システムズ社（以下、DT 社）のマニュアルは以下のとおりです。

メーカー	Man. No.	形式	マニュアル名称
オムロン	SBCE-435	形 CK5M-CPU□□□ 形 CK3M-CPU□□□ 形 CK3W-AX2323□ 形 CK3W-GC2200	プログラマブル多軸モーションコントローラ ユーザーズマニュアル ハードウェア編
DT 社	SBCE-404	—	Power PMAC ユーザーズマニュアル
DT 社	SBCE-405	—	Power PMAC ソフトウェアリファレンスマニュアル
DT 社	SBCE-439	—	Power PMAC IDE ユーザーズマニュアル(V4)

用語と定義

用語	説明・定義
PMAC	Programmable Multi Axis Controller の略です。
Power PMAC IDE	コントローラの設定、ユーザプログラムの作成、モニタを行うためのパソコン用ソフトウェアです。
AC サーボモータ	交流信号入力で駆動するサーボモータです。交流信号の入力により、モータ内に回転磁界を形成することで、モータの回転子が回転します。
サーボアンプ	サーボモータを制御する制御装置です。コントローラからの指令を受け、電源をパワーデバイスによりスイッチングすることでモータを駆動させる動力信号を出力します。
Direct PWM	Power PMAC とサーボアンプを通信するための当社独自の通信方式です。Power PMAC からはサーボアンプのパワーデバイスの ON/OFF 信号を直接送信し、対してサーボアンプからは、電流フィードバック値を Power PMAC に送信します。これにより、Power PMAC で電流ループを形成することができます。
電気角	AC サーボモータを回転させるための回転磁界の角度を指します。
転流制御	必要な電気角を持つ回転磁界を作り、モータを回転させることを指します。
フェーズリファレンス	電気角の原点のことです。転流制御を行う場合は、まずフェーズリファレンスを確立する（フェーズリファレンスサーチ）必要があります。
正弦波エンコーダ	1VPP の SIN/COS 波形を出力するタイプのエンコーダです。
ガルバノスキャナ	レーザを反射するミラーの回転角を制御することで、レーザの照射位置を走査する制御装置です。ミラーの応答性能の高さを活かして、高速な軌跡制御を行うことができます。
SL2-100	デジタルガルバノスキャナと通信するプロトコルの一種です。
コントロールコマンド	ガルバノスキャナからコントローラに送信するデータの種類を設定するコマンドです。
XY ステージ	X 軸方向、Y 軸方向に動く位置決めステージのことです。ガルバノスキャナに比べて応答速度は遅いですが、可動範囲が広いので、より広いエリアに走査を行うことができます。
MOTF	Motion On The Fly の略です。位置指令を高周波成分と低周波成分に分割し、応答速度の異なるアクチュエータに分配するアルゴリズムです。本書では、高周波成分をガルバノスキャナに、低周波成分を XY ステージに指令します。
パルスレーザ	特定の時間間隔で点滅を繰り返すレーザのことです。対照的に常時出力している機器を連続光発振レーザと呼びます。
Q スイッチ	パルスレーザの発振方式のひとつです。外部から発振用のパルスを入力することでパルスレーザを出力します。
TCR	Trigger Output by Commanded distance for Rapid Processing の略です。形 CK3W-GC□2□□が有する距離により、レーザ出力をトリガする機能です。

改訂履歴

改訂記号は、表表紙・裏表紙に記載されている Man. No.の後尾に付記されます。

Man. No. **SBCE-504B**

↑
改訂記号

改訂記号	改訂年月	改訂理由
A	2021 年 5 月	初版
B	2023 年 1 月	- 対象サーボドライバの変更による改訂 - 誤記訂正

本資料の概要

本章では、本資料の概要を記載します。

1-1	概要	1-2
-----	----------	-----

1-1 概要

本資料は、オムロン製プログラマブル多軸モーションコントローラ 形 CK□M-□□□□（以下、コントローラ）においてレーザ加工システムを組み立てるための手順について記載しています。

注意

本資料は、接続対象として記載した機器の接続確認を利用範囲としています。本資料に記載のない命令の使用やシステム構築に際しては、システムを構築する各機器・装置のマニュアルや取扱説明書などを入手し、「安全上のご注意」「安全上の要点」など安全に関する注意事項を含め、内容を確認のうえ、作業を進めてください。



2

対象機器とデバイス構成

本章では、本資料における接続の対象機器およびシステム構成を記載します。

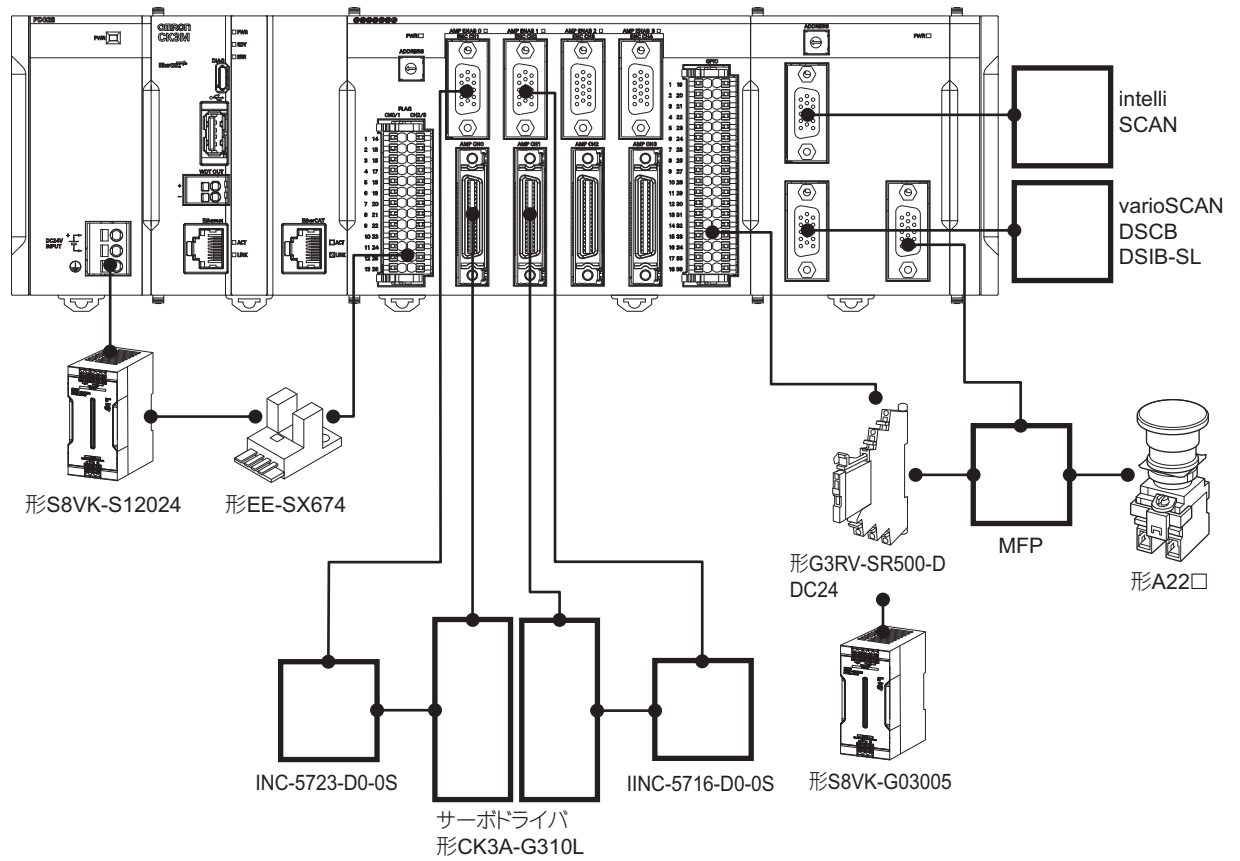
2-1	デバイス構成	2-2
-----	--------------	-----

2-1 デバイス構成

本資料の接続手順を再現するための構成機器は、以下のとおりです。

メーカ	名称	形式	バージョン
オムロン	プログラマブル多軸モーションコントローラ CPU ユニット	形 CK0M-CPU101	Ver.2.6.1 以降
オムロン	プログラマブル多軸モーションコントローラ 軸インタフェースユニット	形 CK3W-AX2323N*1	—
オムロン	プログラマブル多軸モーションコントローラ ガルバノインタフェースユニット	形 CK3W-GC2200	—
オムロン	プログラマブル多軸モーションコントローラ 電源供給ユニット	形 CK3W-PD048	—
オムロン	プログラマブル多軸モーションコントローラ エンドカバー	形 CK3W-TER11	—
オムロン	プログラマブル多軸モーションコントローラ Direct PWM ケーブル	形 CK3W-CAAD036A	—
オムロン	プログラマブル多軸モーションコントローラ シリアルエンコーダケーブル	形 CK3W-CAES03A	—
オムロン	プログラマブル多軸モーションコントローラ レーザインタフェースユニットケーブル	形 CK3W-CAG03A	—
オムロン	スイッチング パワーサプライ(24V DC)	形 S8VK-S12024	—
オムロン	スイッチング パワーサプライ(5V DC)	形 S8VK-G03005	—
オムロン	スイッチング パワーサプライ(15V DC)	形 S8FS-G10015CD	—
オムロン	ソリッドステートリレー	形 G3RV-SR500-D DC24	—
オムロン	フォト マイクロセンサ	形 EE-SX674	—
オムロン	Direct PWM アンプ	形 CK3A-G310L*1	—
Maxphotonics	レーザ発振器	MFP	—
INNO6	リニアモータ (Y 軸)	INC-5716-D0-0S	—
INNO6	リニアモータ (X 軸)	INC-5723-D0-0S	—
HEIDENHEIN	リニアエンコーダ	LIDA48*1	—
SCANLAB	ガルバノスキャナ	intelliSCAN	—
SCANLAB	f θ レンズ	—	—
SCANLAB	ダイナミックフォーカシングユニット	varioSCAN	—
—	Windows パソコン	—	—
DT 社	Power PMAC 設定ツール	Power PMAC IDE	Ver.4.5 以降

*1. 本書では Direct PWM または正弦波エンコーダ方式で XY ステージを制御することを前提としています。他の制御方式を採用する場合は、他のスタートアップガイドを参考にして「3-4 モータとエンコーダの設定 (P.3-21)」の設定内容を変更してください。



接続手順

本章では、レーザアプリケーションを構成する各機器と接続する手順について記載します。説明は、コントローラが工場出荷時の初期設定状態であることを前提とします。

3-1	作業の流れ	3-2
3-2	コントローラの設定準備	3-4
3-2-1	新規プロジェクトの作成	3-4
3-2-2	コントローラの初期設定	3-5
3-3	MOTF の設定	3-7
3-3-1	プログラミング	3-7
3-3-2	フィルタ係数の決定	3-12
3-4	モータとエンコーダの設定	3-21
3-4-1	各機器の配線	3-21
3-4-2	プログラミング	3-22
3-4-3	電流ループのチューニング	3-26
3-4-4	フェーズリファレンスの確立	3-28
3-4-5	オープンループテスト	3-29
3-4-6	位置ループのオートチューニング	3-30
3-4-7	位置ループのインタラクティブチューニング	3-31
3-4-8	Y 軸のチューニング	3-35
3-4-9	チューニング結果の確認	3-35
3-5	オーバトラベルリミットスイッチと原点の設定	3-38
3-5-1	各機器の配線	3-38
3-5-2	プログラミング	3-38
3-5-3	設定の確認（オーバトラベルリミットスイッチ）	3-40
3-5-4	原点の設定	3-41
3-6	ガルバノスキャナの設定	3-43
3-6-1	各機器の配線	3-43
3-6-2	プログラミング	3-44
3-6-3	コントロールコマンドの設定	3-46
3-6-4	設定の確認	3-48
3-6-5	MOTF の実機検証	3-49
3-7	レーザ発振器の設定	3-50
3-7-1	各機器の配線	3-50
3-7-2	ガイドレーザの出力	3-50
3-7-3	プログラミング	3-52
3-7-4	レーザ発振器の設定	3-55
3-7-5	設定の確認	3-56

3-1 作業の流れ

レーザアプリケーションを構築するための手順は以下のとおりです。

「3-2 コントローラの設定準備 (P.3-4)」	コントローラの設定準備を行います。
▼	
「3-2-1 新規プロジェクトの作成 (P.3-4)」	
▼	
「3-2-2 コントローラの初期設定 (P.3-5)」	
▽	
「3-3 MOTF の設定 (P.3-7)」	MOTF の設定を行います。
▼	
「3-3-1 プログラミング (P.3-7)」	
▼	
「3-3-2 フィルタ係数の決定 (P.3-12)」	
▽	
「3-4 モータとエンコーダの設定 (P.3-21)」	モータとエンコーダの設定を行います。
▼	
「3-4-1 各機器の配線 (P.3-21)」	
▼	
「3-4-2 プログラミング (P.3-22)」	
▼	
「3-4-3 電流ループのチューニング (P.3-26)」	
▼	
「3-4-4 フェーズリファレンスの確立 (P.3-28)」	
▼	
「3-4-5 オープンループテスト (P.3-29)」	
▼	
「3-4-6 位置ループのオートチューニング (P.3-30)」	
▼	
「3-4-7 位置ループのインタラクティブチューニング (P.3-31)」	
▼	
「3-4-8 Y 軸のチューニング (P.3-35)」	
▼	
「3-4-9 チューニング結果の確認 (P.3-35)」	
▽	
「3-5 オーバトラベルリミットスイッチと原点の設定 (P.3-38)」	各種センサの設定を行います。
▼	
「3-5-1 各機器の配線 (P.3-38)」	
▼	
「3-5-2 プログラミング (P.3-38)」	
▼	
「3-5-3 設定の確認 (オーバトラベルリミットスイッチ) (P.3-40)」	
▼	
「3-5-4 原点の設定 (P.3-41)」	
▽	
「3-6 ガルバノスキャナの設定 (P.3-43)」	ガルバノスキャナの設定を行います。
▼	
「3-6-1 各機器の配線 (P.3-43)」	

▼	
「3-6-2 プログラミング (P.3-44)」	
▼	
「3-6-3 コントロールコマンドの設定 (P.3-46)」	
▼	
「3-6-4 設定の確認 (P.3-48)」	
▼	
「3-6-5 MOTF の実機検証 (P.3-49)」	
▽	
「3-7 レーザ発振器の設定 (P.3-50)」	レーザ発振器の設定を行います。
▼	
「3-7-1 各機器の配線 (P.3-50)」	
▼	
「3-7-2 ガイドレーザの出力 (P.3-50)」	
▼	
「3-7-3 プログラミング (P.3-52)」	
▼	
「3-7-4 レーザ発振器の設定 (P.3-55)」	
▼	
「3-7-5 設定の確認 (P.3-56)」	

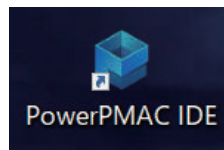
3-2 コントローラの設定準備

コントローラの設定準備を行います。
Power PMAC IDE を、あらかじめパソコンにインストールしてください。

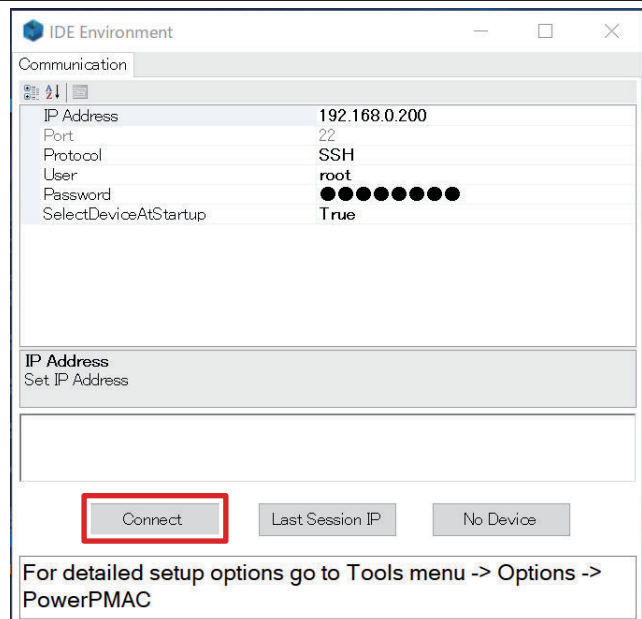
3-2-1 新規プロジェクトの作成

以下の手順に従い、新規プロジェクトの作成を行います。

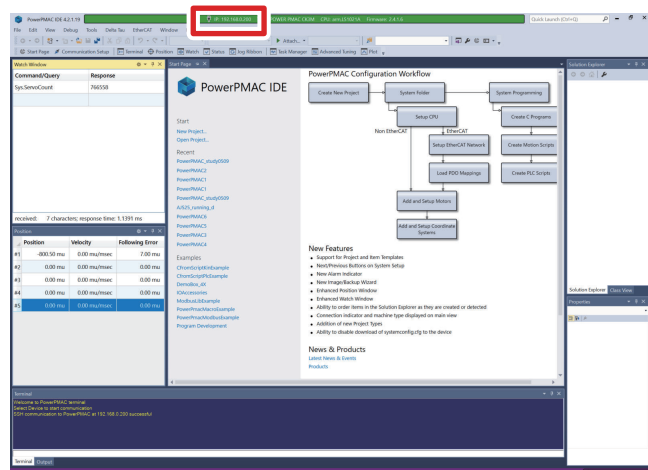
- 1 コントローラとパソコンを Ethernet ケーブルで接続します。
- 2 コントローラの電源を投入します。
- 3 Power PMAC IDE を起動します。
 - 起動時に、アクセス権確認用のダイアログが表示される場合、起動する選択を行ってください。



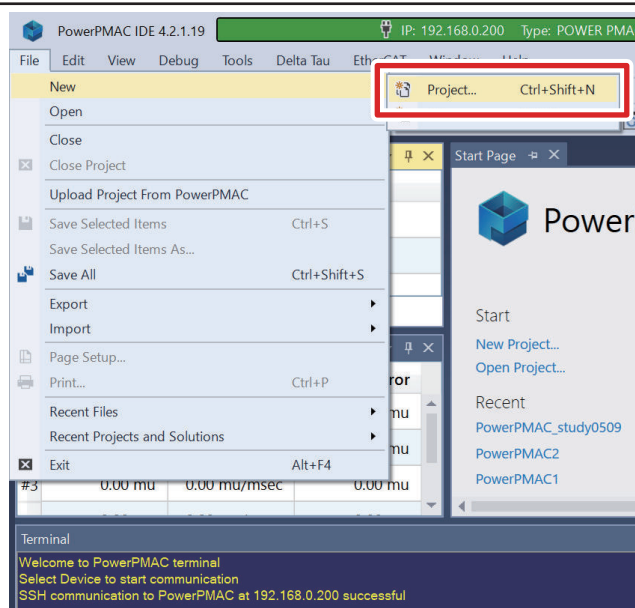
- 4 Communication 画面が表示されますので、接続先のコントローラの IP アドレスを指定し、[Connect] ボタンをクリックします。
 - コントローラのデフォルトの IP アドレスは“192.168.0.200”です。
 - 必要であれば、Windows の IP アドレスを“192.168.0.X”へ変更してください。



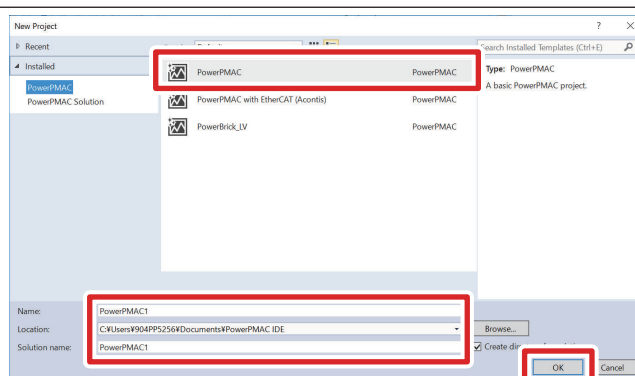
- 5 Power PMAC IDE が起動し、コントローラとオンラインの状態になります。



- 6 [File] メニューの [New] - [Project] を選択します。



- 7 右図の通り [Power PMAC] を選択後、任意のプロジェクト名、保存先を入力し、[OK] ボタンをクリックします。



3-2-2 コントローラの初期設定

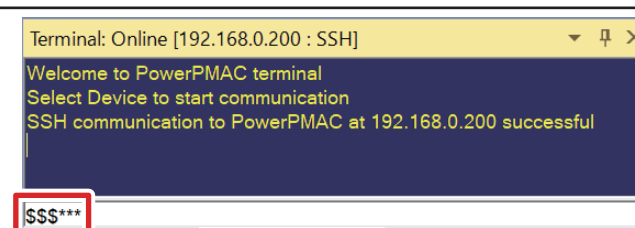
以下の手順に従い、コントローラの初期設定を行います。

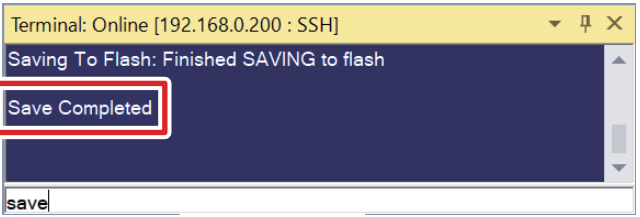
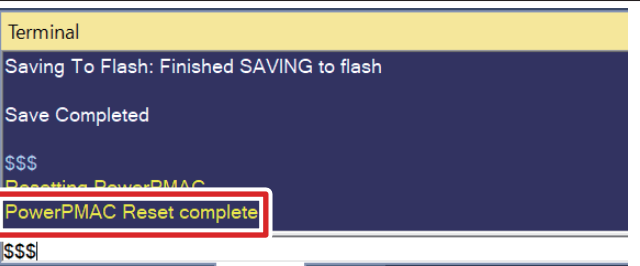


使用上の注意

初期設定により、メモリがすべてクリアされますので、必要なデータがコントローラ内に残っている場合は、データを保管してください。

- 1 Terminal から [\$\$\$***] コマンドをタイプし、コントローラを工場出荷状態にします。



2	Power PMAC IDE の Terminal で [save] コマンドをタイプします。 終了すると、Terminal に “Save Completed” と表示されます。	 Terminal: Online [192.168.0.200 : SSH] Saving To Flash: Finished SAVING to flash Save Completed save
3	Power PMAC IDE の Terminal で [\$\$\$] コマンドをタイプします。 終了すると、Terminal に “PowerPMAC Reset complete” と表示されます。	 Terminal Saving To Flash: Finished SAVING to flash Save Completed \$\$\$ Resetting PowerPMAC PowerPMAC Reset complete \$\$\$

3-3 MOTF の設定

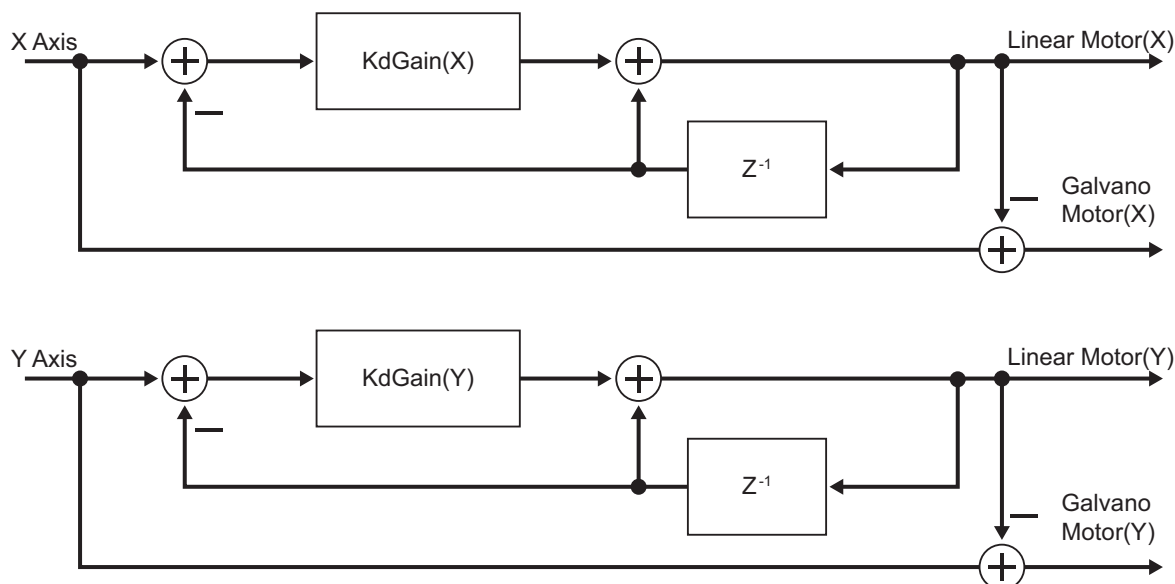
3-3-1 プログラミング

MOTF の設定を行います。本書では、EMA（指数平滑移動平均）によるローパスフィルタによって、低周波成分と高周波成分に振り分けます。

以下に EMA の入出力方程式を示します。ここでは、 $u(n)$ が時刻 n における入力信号、 $y(n)$ が出力信号です。

$$y(n) = KdGain * u(n) + (1 - KdGain) * y(n-1)$$

以下に、MOTF のブロック線図を示します。ある軸に対する指令のうち、ローパスフィルタを通過したあとの低周波成分をモータに、残移動量をガルバノスキャナに入力します。これにより、高スローク、高応答の両立を行うことができます。



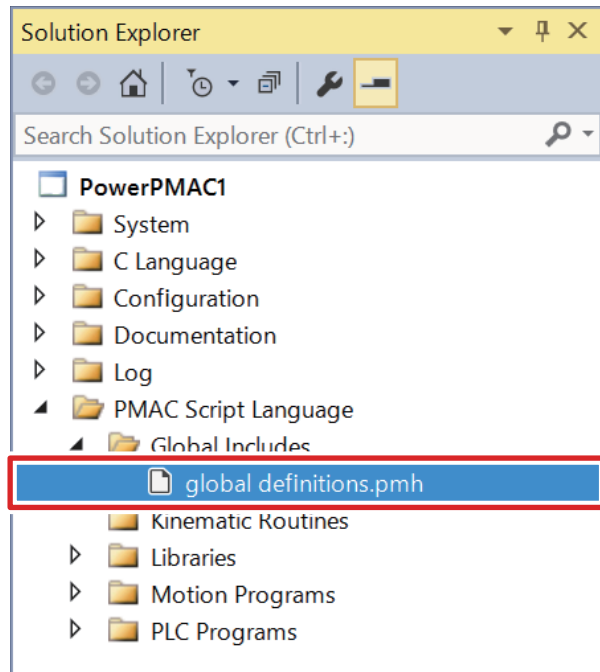
上記において、KdGain は 0 以上 1 未満の間に設定します。ローパスフィルタのカットオフ周波数を大きくしたい場合には、KdGain をより大きい値に設定します。

本書における、装置上の軸とモータ番号の関連を以下の表に示します。

装置上の軸		モータ番号
機構	方向	
XY ステージ	X 軸	#1
	Y 軸	#2
ガルバノスキャナ	X 軸	#3
	Y 軸	#4
フォーカスユニット	Z 軸	#5
TCR 計算用の仮想軸		#6

1

Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [global definitions.pmh] を開きます。



2

右のテキストを global definitions.pmh に書き込みます。

- 注釈付きの設定の内容については、「第 4 章 各種設定のカスタマイズ方法 (P.4-1)」を参照してください。

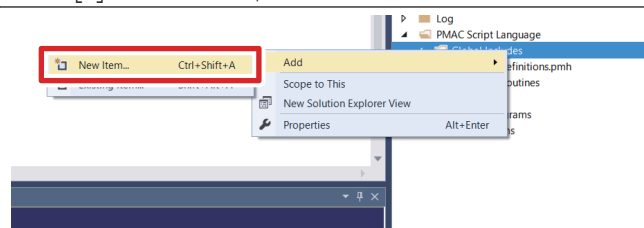
```
Sys. WpKey=$AAAAA
//Segmentation Time
Coord[1]. SegMoveTime=0.1; //*1

//Global Clock Configurations
Gate3[0]. PhaseFreq=10000; //*2
Gate3[0]. ServoClockDiv=0; //*3
Sys. PhaseOverServoPeriod=1; //*4
Sys. ServoPeriod=0.1; //*5
Gate3[0]. PhaseServoDir=0; //*6
Gate3[1]. PhaseServoDir=3; //*6

//ADC settings
Gate3[0]. AdcEncCtrl=$3fffc000
```

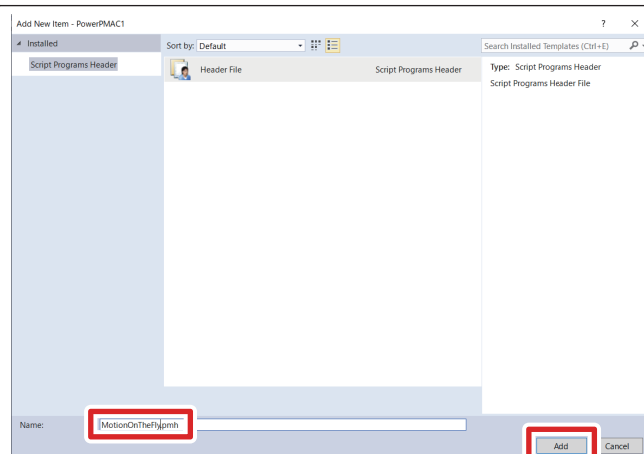
3

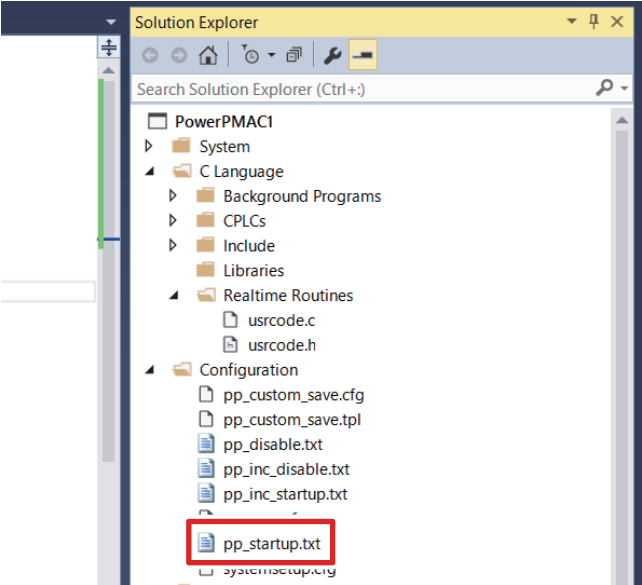
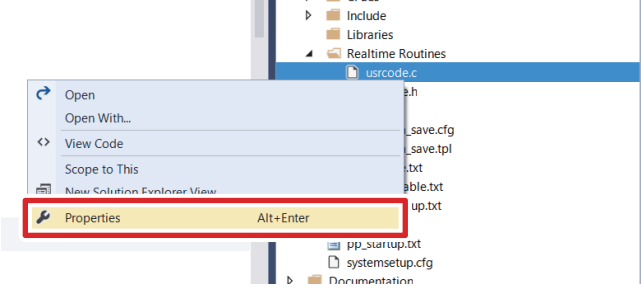
Solution Explorer の [PMAC Script Language] を右クリックし、[Add] から [New Item...] を選択します。



4

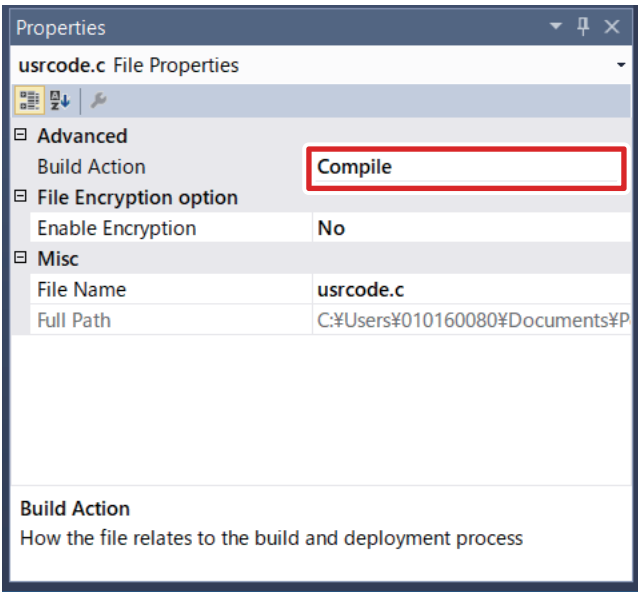
[Name] の欄に「MotionOnTheFly.pmh」とタイプし、[Add] ボタンをクリックします。



5	Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [MotionOnTheFly.pmh] を開きます。
6	右のテキストを MotionOnTheFly.pmh に書き込みます。 注釈付きの設定の内容については、「第 4 章 各種設定のカスタマイズ方法 (P.4-1)」を参照してください。 <pre>//LPF's Gain global KdgainX=0.00045;//*1 global KdgainY=0.00045;//*2 //Scale Factors of Galvo Scanner global GalvoSfX=-503316*4096/(170*0.374);//*3 global GalvoSfY=-503316*4096/(170*0.374);//*4 //Settings of Coordinate System &1;//*5 #1->1;//*5 #2->1;//*5 #3->1;//*5 #4->1;//*5</pre>
7	Solution Explorer の [Configuration] - [pp_startup.txt] を開きます。 
8	右のテキストを pp_startup.txt に書き込みます。 <pre>//Enable CfromScript UserAlgo. CFunc=1;</pre>
9	Solution Explorer の [C Language] - [Realtime Routines] - [usercode.c] で右クリックし、[Properties] を選択します。 

10

[Properties] ウィンドウの [Build Action] で [Compile] を選択します。



11

Solution Explorer の [C Language] - [Realtime Routines] - [usrcode.c] を開きます。

12

右の CfromScript の関数定義を削除します。

```
double CfromScript(double cfrom_type, double arg2, double arg
3, double arg4, double arg5, double arg6, double arg7, struct Lo
calData *Ldata)
{
    int icfrom_type=(int)cfrom_type;
    double*C, *D, *L, *R, rtn;//C, D, R-only needed if doing Kine
matics

    C=GetCVarPtr(Ldata);//Only needed if doing Kinematics
    D=GetDVarPtr(Ldata);//Only needed if doing Kinematics
    L=GetLVarPtr(Ldata);//Only needed if using Ldata or Kin
ematics
    R=GetRVarPtr(Ldata);//Only needed if doing Kinematics
    rtn=-1.0;
    return rtn;
}
```

13

右のテキストを usercode.c に書き込みます。

```
static double prevLpfXPos=0;
static double prevLpfYPos=0;

double CfromScript(double kinTypeDouble, double arg2, double
arg3, double arg4, double arg5, double arg6, double arg7, LocalD
ata *Ldata)
{
    int kinType=(int)kinTypeDouble;
    double deltaXPos, deltaYPos;
    double *L=GetLVarPtr(Ldata);
    double *C=GetCVarPtr(Ldata);

    switch(kinType)
    {
        case Forward_Kinematics_State:
        {
            _KinPosAxisX=_KinPosMotor1+(_KinPosMotor3/GalvoSfX);
            _KinPosAxisY=_KinPosMotor2+(_KinPosMotor4/GalvoSfY);
            if(Ldata->Status & 0x40)
            {
                prevLpfXPos=_KinPosMotor1;
                prevLpfYPos=_KinPosMotor2;
            }
            break;
        }
        case Inverse_Kinematics_State:
        {
            deltaXPos=_KinPosAxisX-prevLpfXPos;
            deltaYPos=_KinPosAxisY-prevLpfYPos;
            _KinPosMotor1=prevLpfXPos+(KdgainX*deltaXPos);
            _KinPosMotor2=prevLpfYPos+(KdgainY*deltaYPos);
            _KinPosMotor3=(_KinPosAxisX-_KinPosMotor1)*GalvoSfX;
            _KinPosMotor4=(_KinPosAxisY-_KinPosMotor2)*GalvoSfY;
            prevLpfXPos=_KinPosMotor1;
            prevLpfYPos=_KinPosMotor2;
            break;
        }
    }
    return 0;
}
```

14

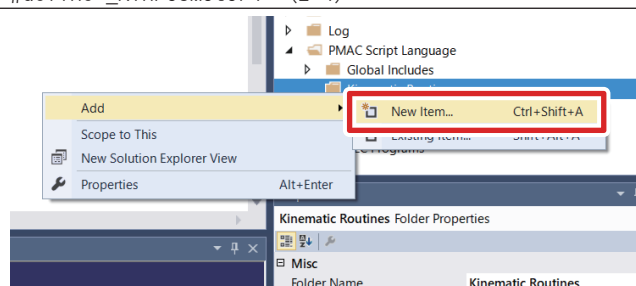
usercode.c のファイルの先頭に、右のテキストを追加します。

```
#define Forward_Kinematics_State 0
#define Inverse_Kinematics_State 1

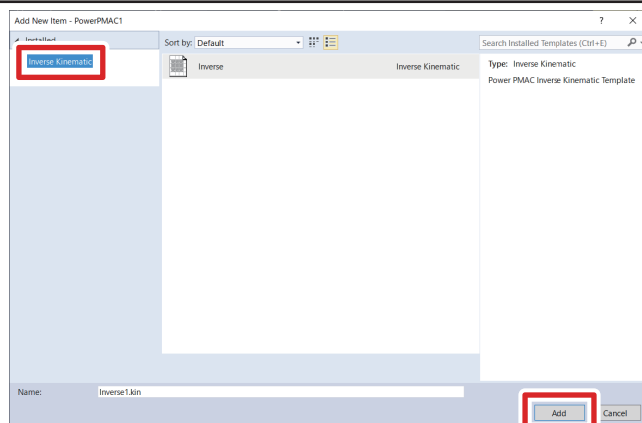
#define _KinPosAxisX *(C+6)
#define _KinPosAxisY *(C+7)
#define _KinPosMotor1 *(L+1)
#define _KinPosMotor2 *(L+2)
#define _KinPosMotor3 *(L+3)
#define _KinPosMotor4 *(L+4)
```

15

Solution Explorer の [PMAC Script Language] - [Kinematics Routines] を右クリックし、[Add] から [New Item...] を選択します。



- 16 [Inverse Kinematic] を選択し、[Add] ボタンをクリックします。



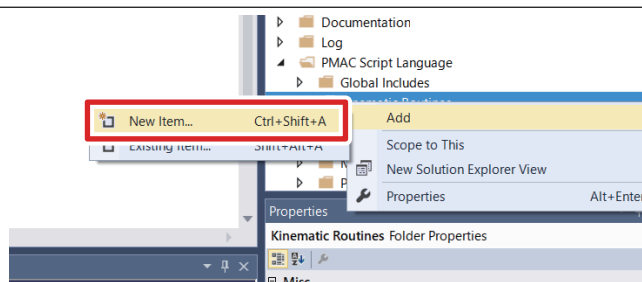
- 17 右のテキストを Inverse1.kin に書き込みます。

```
open inverse(1)

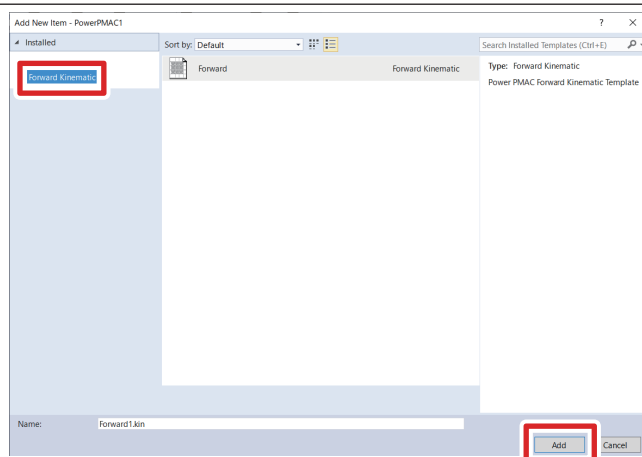
local ret;
ret=CfromScript(1, 0, 0, 0, 0, 0);

close
```

- 18 Solution Explorer の [PMAC Script Language] - [Kinematics Routines] を右クリックし、[Add] から [New Item...] を選択します。



- 19 [Forward Kinematic] を選択し、[Add] ボタンをクリックします。



- 20 右のテキストを Forward1.kin に書き込みます。

```
open forward(1)

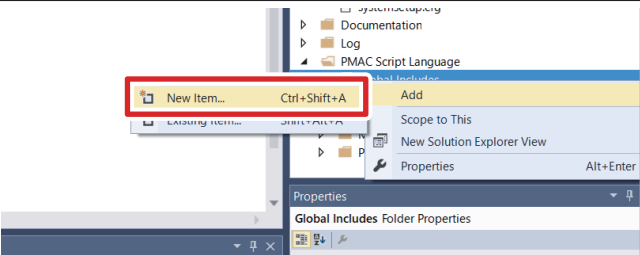
local ret;
ret=CfromScript(0, 0, 0, 0, 0, 0);

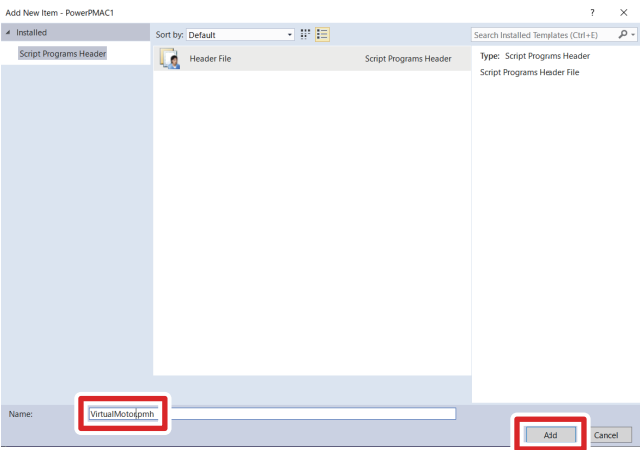
close
```

3-3-2 フィルタ係数の決定

MOTF に使用するフィルタのパラメータ KdGain を調整します。

- 1** Solution Explorer の [PMAC Script Language] - [Global Includes] を右クリックし、[Add] から [New Item...] を選択します。


- 2** [Name] の欄に「VirtualMotor.pmh」とタイプし、[Add] ボタンをクリックします。


- 3** Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [VirtualMotor.pmh] を開きます。

4

右のテキストを VirtualMotor.pmh に書き込みます。

```
EncTable[1].type=1
EncTable[1].index1=0
EncTable[1].index2=0
EncTable[1].index3=0
EncTable[1].index4=0
EncTable[1].pEnc1=Sys. udata[10]. a
EncTable[1].pEnc =Sys. udata[10]. a;
EncTable[1].ScaleFactor=1;
Motor[1].pDac=Sys. udata[10]. a
Motor[1].pEnc =EncTable[1]. a
Motor[1].pEnc2=EncTable[1]. a
Motor[1].pLimits=0
Motor[1].pAmpFault=0
Motor[1].Ctrl=Sys. PosCtrl
Motor[1].ServoCtrl=1
Motor[1].MaxSpeed=0
Motor[1].FatalFeLimit=0

EncTable[2].type=1
EncTable[2].index1=0
EncTable[2].index2=0
EncTable[2].index3=0
EncTable[2].index4=0
EncTable[2].pEnc1=Sys. udata[11]. a
EncTable[2].pEnc=Sys. udata[11]. a;
EncTable[2].ScaleFactor=1;
Motor[2].pDac=Sys. udata[11]. a
Motor[2].pEnc=EncTable[2]. a
Motor[2].pEnc2=EncTable[2]. a
Motor[2].pLimits=0
Motor[2].pAmpFault=0
Motor[2].Ctrl=Sys. PosCtrl
Motor[2].ServoCtrl=1
Motor[2].MaxSpeed=0
Motor[2].FatalFeLimit=0
```

```

EncTable[3]. type=1
EncTable[3]. index1=0
EncTable[3]. index2=0
EncTable[3]. index3=0
EncTable[3]. index4=0
EncTable[3]. pEnc1=Sys. udata[12]. a
EncTable[3]. pEnc=Sys. udata[12]. a
EncTable[3]. ScaleFactor=1
Motor[3]. pDac=Sys. udata[12]. a
Motor[3]. pEnc=EncTable[3]. a
Motor[3]. pEnc2=EncTable[3]. a
Motor[3]. pLimits=0
Motor[3]. pAmpFault=0
Motor[3]. Ctrl=Sys. PosCtrl
Motor[3]. ServoCtrl=1
Motor[3]. MaxSpeed=0
Motor[3]. FatalFeLimit=0

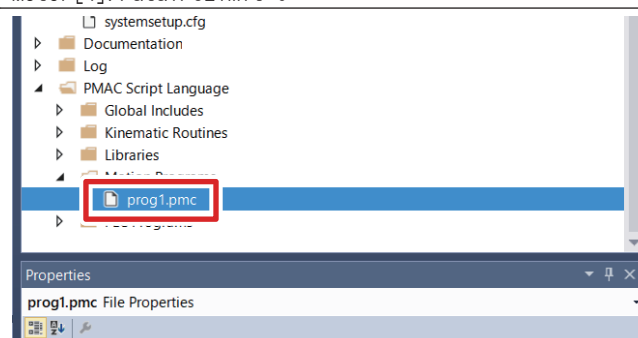
```

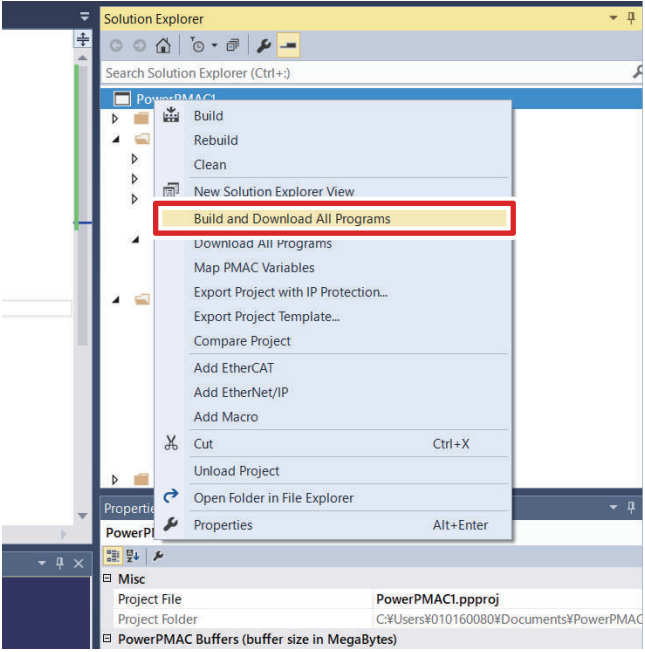
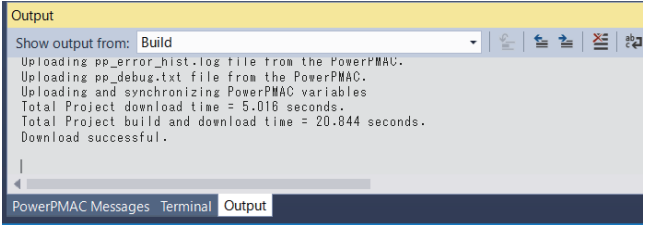
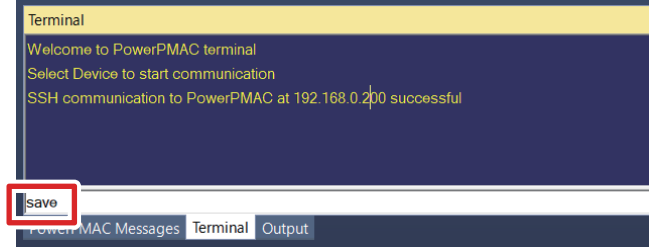
```

EncTable[4]. type=1
EncTable[4]. index1=0
EncTable[4]. index2=0
EncTable[4]. index3=0
EncTable[4]. index4=0
EncTable[4]. pEnc1=Sys. udata[13]. a
EncTable[4]. pEnc=Sys. udata[13]. a
EncTable[4]. ScaleFactor=1
Motor[4]. pDac=Sys. udata[13]. a
Motor[4]. pEnc=EncTable[4]. a
Motor[4]. pEnc2=EncTable[4]. a
Motor[4]. pLimits=0
Motor[4]. pAmpFault=0
Motor[4]. Ctrl=Sys. PosCtrl
Motor[4]. ServoCtrl=1
Motor[4]. MaxSpeed=0
Motor[4]. FatalFeLimit=0

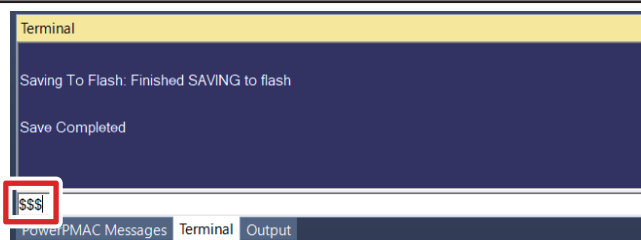
```

- 5 Solution Explorer の [PMAC Script Language] - [Motion Programs] の下にある [prog1.pmc] を開きます。

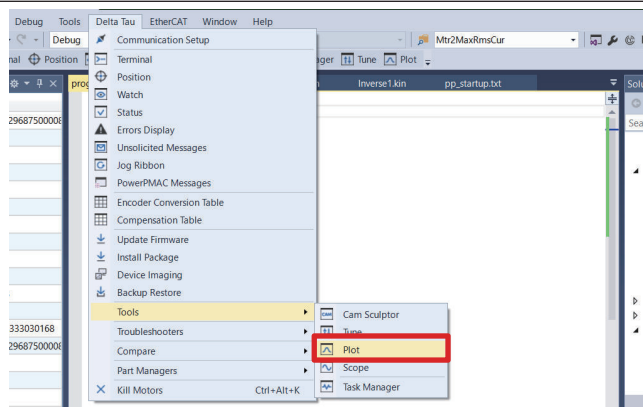


6	加工で使用するプログラムを prog1.pmc に書き込みます。右のテキストは、加工プログラムの一例です。	<pre>open prog 1 coord[1].FeedTime=1000; Coord[1].MaxFeedRate=180; ta(0.1); td(0.1); ts(0); F(100); abs; linear; X(50) Y(50); X(-50) Y(50); X(-50) Y(-50); X(50) Y(-50); X(50) Y(50); X(0) Y(0); close</pre>
7	プロジェクトのダウンロード Power PMAC IDE の画面右上の Solution Explorer のプロジェクト名を右クリックし、[Build and Download All Programs] を選択し、ビルド&ダウンロードを実行します。	
8	Output で異常がないことを確認します。 転送が失敗した場合は、Output でエラーの内容を確認してください。	
9	Power PMAC IDE の Terminal で [save] コマンドをタイプします。 終了すると、Terminal に “Save Complete” と表示されます。	

- 10 Power PMAC IDE の Terminal で [\$\$ \$] コマンドをタイプします。

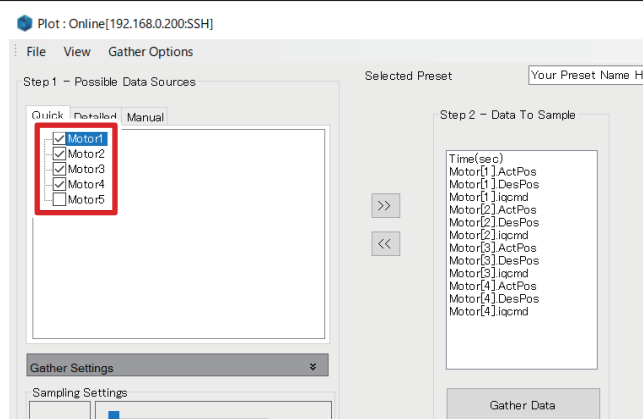


- 11 [Delta Tau] メニューの [Tools] から [Plot] を選択します。



- 12 [Step1 – Possible Data Sources] 内の、以下のアイテムのチェックボックスにチェックを入れます。

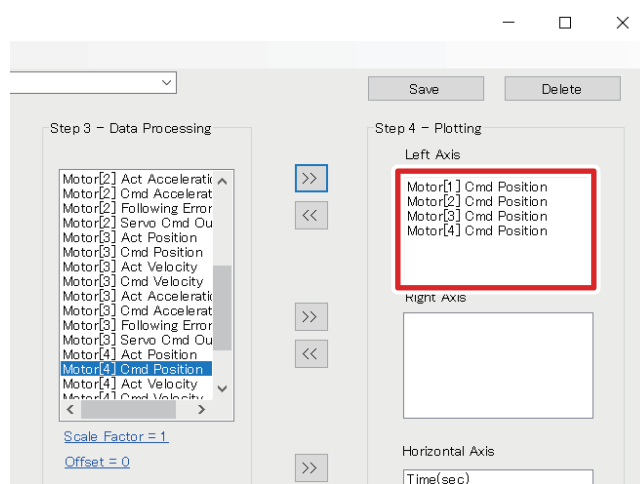
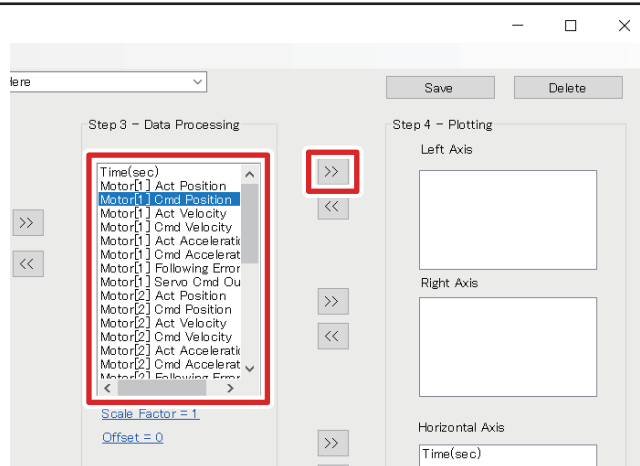
- [Motor1]
- [Motor2]
- [Motor3]
- [Motor4]



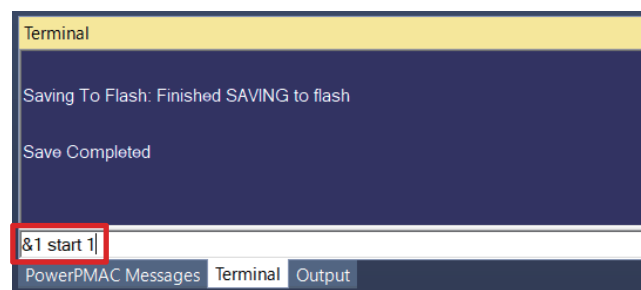
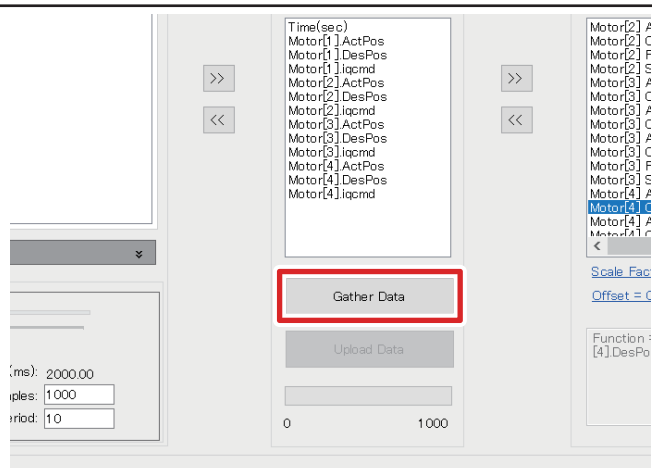
13

[Step3 - Data Processing] 内の以下のアイテムを選択し、[>>] ボタンをクリックします。

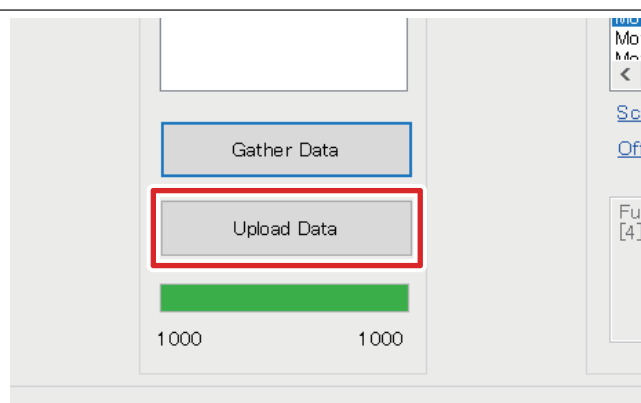
- [Motor[1] Cmd Position]
- [Motor[2] Cmd Position]
- [Motor[3] Cmd Position]
- [Motor[4] Cmd Position]



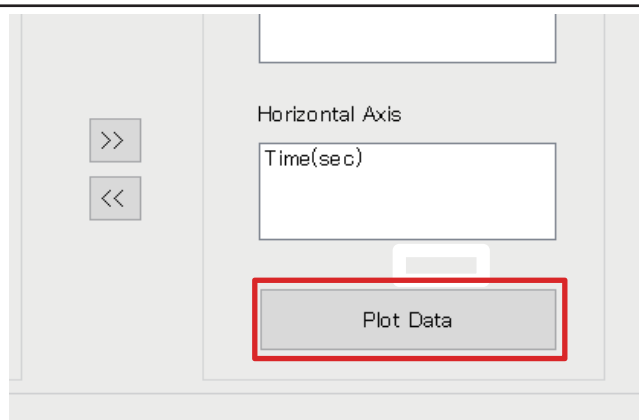
- 14** [Gather Data] ボタンをクリックし、Power PMAC IDE の Terminal で [&1 enable] と [&1 start 1] コマンドをタイプします。



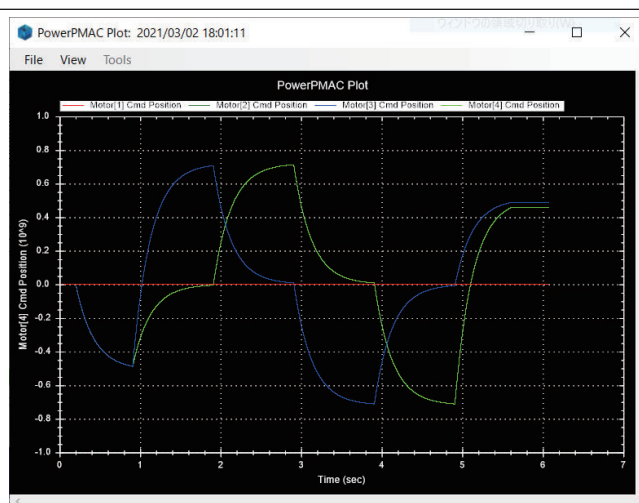
- 15** [Upload Data] ボタンをクリックします。



16 [Plot Data] ボタンをクリックします。



17 右のように、各軸位置が表示されることを確認します。



18 Motor[3] Cmd Position、Motor[4] Cmd Position のいずれかがガルバノスキャナの指令最大値／最小値を超える場合は、[global definitions.pmh]を開き、KdgainX、KdgainYを編集します。そのあと、手順 7～17 を再度実行します。

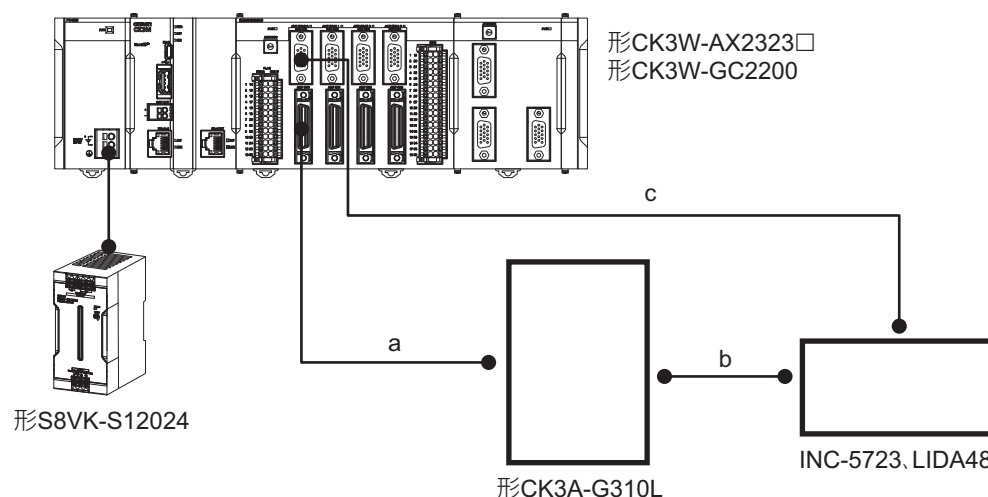
19 手順 18 でフィルタの応答に問題がなければ、Solution Explorer 上の [VirtualMotor.pmh] を右クリックし、[Delete] を選択します。

3-4 モータとエンコーダの設定

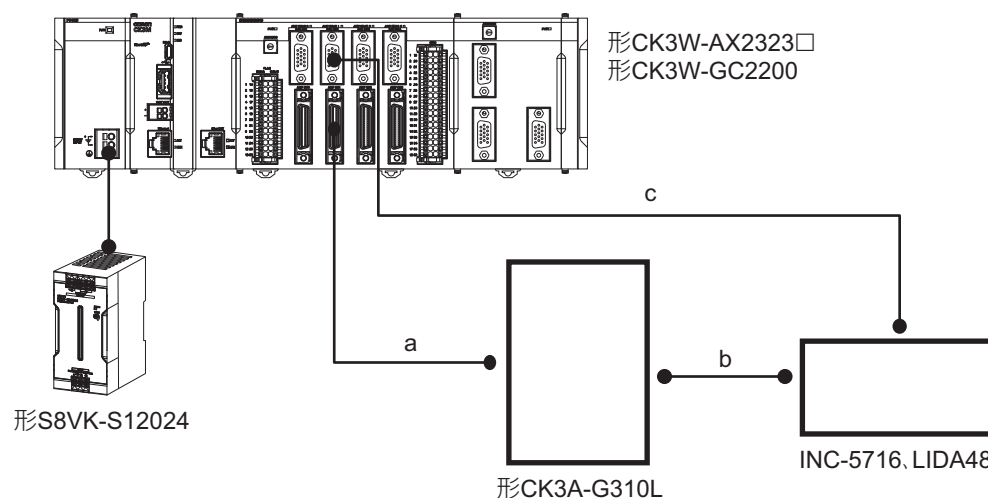
3-4-1 各機器の配線

CK3M とサーボドライバ、リニアモータとの配線方法について記載します。

・ X 軸の配線



・ Y 軸の配線



上図 a、b、c の箇所を以下の要領に従って接続します。

a. コントローラとサーボアンプ間の接続

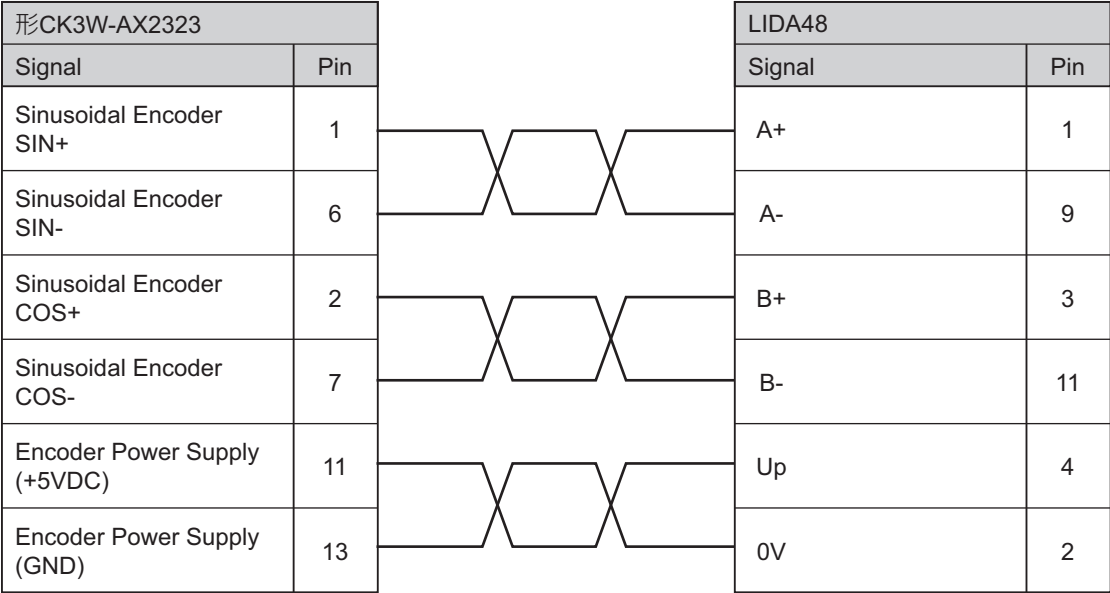
形 CK3W-AX2323□ のアンプ接続コネクタとサーボアンプの C2 コネクタを、以下の専用ケーブルで接続します。

メーカー	名称	形式	長さ
オムロン	Direct PWM ケーブル	形 CK3W-CAAD009A	0.9m
		形 CK3W-CAAD018A	1.8m
		形 CK3W-CAAD036A	3.6m

b. サーボアンプとサーボモータ間の接続

サーボアンプの P2 コネクタにリニアモータの Motor Connector を接続します。

- c. コントローラとサーボモータ間の接続
形 CK3W-AX2323口のエンコーダ接続コネクタとリニアエンコーダのコネクタを、専用ケーブル
形 CK3W-CAEA03A で接続します。

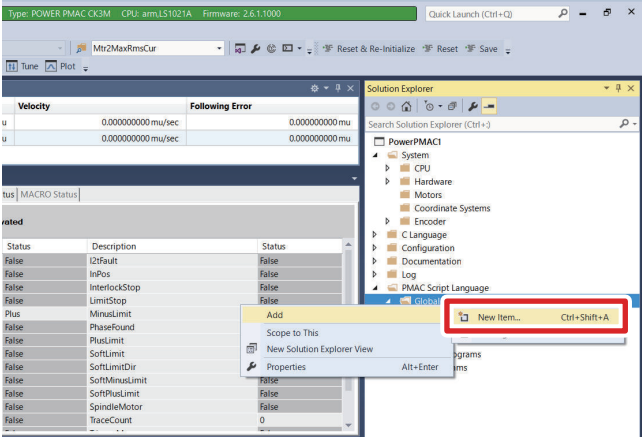


3-4-2 プログラミング

サーボアンプ、リニアモータを制御するための設定をプログラミングします。

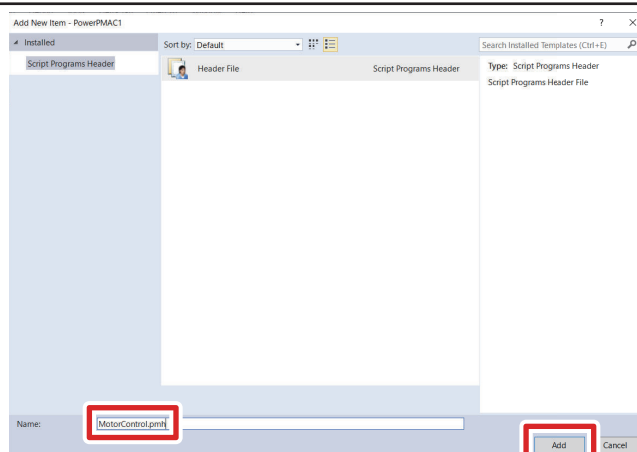
1

Solution Explorer の [PMAC Script Language] - [Global Includes] を右クリックし、[Add] から [New Item...] を選択します。



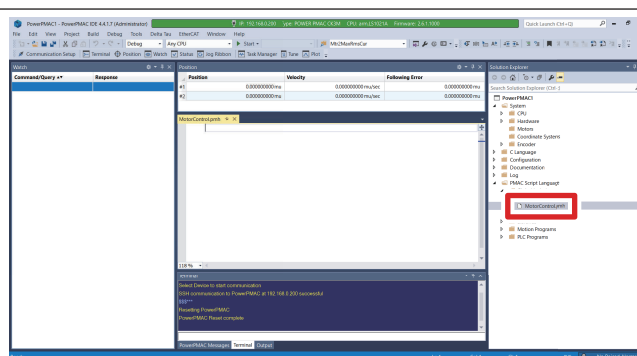
2

[Name] の欄に「MotorControl.pmh」とタイプし、[Add] ボタンをクリックします。



3

Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [MotorControl.pmh] を開きます。



4

下のテキストを MotorControl.pmh に書き込みます。

- 注釈付きの設定の内容については、「第 4 章 各種設定のカスタマイズ方法 (P.4-1)」を参照してください。

```
Sys. WpKey=$AAAAAAA

//Servo Task Configurations
Motor[1]. ServoCtrl=1 // *1
Motor[2]. ServoCtrl=1 // *1
Motor[1]. pEnc=EncTable[1]. a // *2
Motor[2]. pEnc=EncTable[2]. a // *2
Motor[1]. pEnc2=EncTable[1]. a // *2
Motor[2]. pEnc2=EncTable[2]. a // *2
Motor[1]. EncType=6 // *3
Motor[2]. EncType=6 // *3
Motor[1]. PosUnit=3 // *4
Motor[2]. PosUnit=3 // *4
EncTable[1]. Type=1 // *5
EncTable[2]. Type=1 // *5
EncTable[1]. pEnc=Gate3[0]. Chan[0]. ServoCapt. a // *6
EncTable[2]. pEnc=Gate3[0]. Chan[1]. ServoCapt. a // *6
EncTable[1]. ScaleFactor=1.0 // *7
EncTable[2]. ScaleFactor=1.0 // *7
Motor[1]. AmpFaultLevel=1 // *8
Motor[2]. AmpFaultLevel=1 // *8
Motor[1]. FatalFeLimit=5.0 // *9
Motor[2]. FatalFeLimit=2.0 // *9

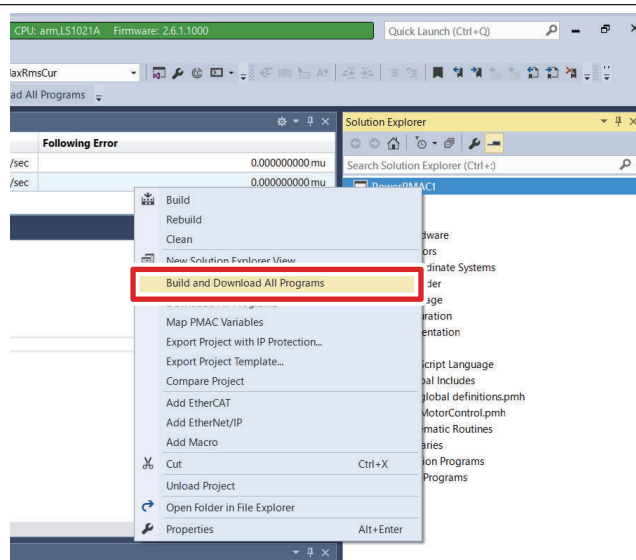
//Phase Task Configurations
Motor[1]. PhaseCtrl=4; // *10
Motor[2]. PhaseCtrl=4; // *10
Motor[1]. pPhaseEnc=Gate3[0]. Chan[0]. PhaseCapt. a // *11
Motor[2]. pPhaseEnc=Gate3[0]. Chan[1]. PhaseCapt. a // *11
Motor[1]. PhasePosSf=2048*(0.020/16384)/36 // *12
Motor[2]. PhasePosSf=2048*(0.020/16384)/36 // *12
Motor[1]. PhaseOffset=683 // *13
Motor[2]. PhaseOffset=683 // *13
Motor[1]. pAdc=Gate3[0]. Chan[0]. AdcAmp[0]. a // *14
Motor[2]. pAdc=Gate3[0]. Chan[1]. AdcAmp[0]. a // *14
Motor[1]. AdcMask=$FFFF0000 // *15
Motor[2]. AdcMask=$FFFF0000 // *15
Motor[1]. pDac=Gate3[0]. Chan[0]. Pwm[0]. a // *16
Motor[2]. pDac=Gate3[0]. Chan[1]. Pwm[0]. a // *16
Motor[1]. PhaseFindingDac=Motor[1]. I2tSet/2 // *17
Motor[2]. PhaseFindingDac=Motor[2]. I2tSet/2 // *17
Motor[1]. PhaseFindingTime=3000/(2*Sys. ServoPeriod*(Sys. RtIntPeriod+1)) // *18
Motor[2]. PhaseFindingTime=3000/(2*Sys. ServoPeriod*(Sys. RtIntPeriod+1)) // *18
```

```
//PWM Output Configurations
Motor[1].PwmSf=0.95*16384 // *19
Motor[2].PwmSf=0.95*16384 // *19
Motor[1].MaxDac=9.88*32768*COSD(30)/31.35 // *20
Motor[2].MaxDac=9.88*32768*COSD(30)/31.35 // *20
Motor[1].I2tSet=3.08*32768*COSD(30)/31.35 // *21
Motor[2].I2tSet=3.08*32768*COSD(30)/31.35 // *21
Motor[1].I2tTrip=(Motor[1].MaxDac*Motor[1].MaxDac-Motor[1].I2tSet*Motor[1].I2tSet)*2 // *22
Motor[2].I2tTrip=(Motor[2].MaxDac*Motor[2].MaxDac-Motor[2].I2tSet*Motor[2].I2tSet)*2 // *22
Gate3[0].Chan[0].PwmDeadTime=3 / 0.0533 // *23
Gate3[0].Chan[1].PwmDeadTime=3 / 0.0533 // *23
Gate3[0].Chan[0].PwmFreqMult=2 // *24
Gate3[0].Chan[1].PwmFreqMult=2 // *24
Gate3[0].Chan[0].PackOutData=0 // *25
Gate3[0].Chan[1].PackOutData=0 // *25

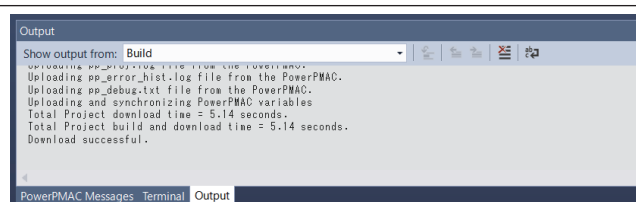
//Current Loop Configurations
Gate3[0].AdcAmpStrobe = $901001; // *26
Gate3[0].AdcAmpHeaderBits=4; // *27
Gate3[0].AdcAmpClockDiv=5; // *28
Gate3[0].AdcEncClockDiv=3 // *28
Gate3[0].Chan[0].PackInData=0 // *29
Gate3[0].Chan[1].PackInData=0 // *29

//Sinusoidal Encoder Settings
Gate3[0].Chan[0].AtanEna=1 // *30
Gate3[0].Chan[1].AtanEna=1 // *30
Gate3[0].Chan[0].EncCtrl=3 // *31
Gate3[0].Chan[1].EncCtrl=3 // *31
Motor[1].PosSf=0.020/16384 // *32
Motor[2].PosSf=0.020/16384 // *32
Motor[1].Pos2Sf=Motor[1].PosSf // *32
Motor[2].Pos2Sf=Motor[2].PosSf // *32
```

- 5 Power PMAC IDE の画面右上の Solution Explorer のプロジェクト名を右クリックし、[Build and Download All Programs] を選択し、ビルド&ダウンロードを実行します。

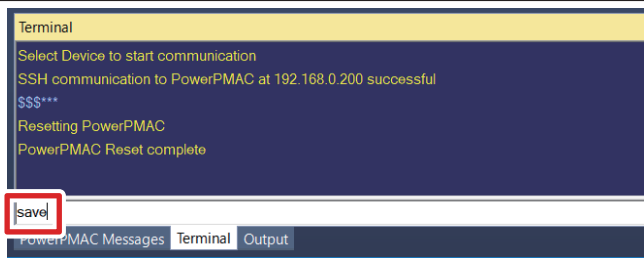


- 6 Output で異常がないことを確認します。



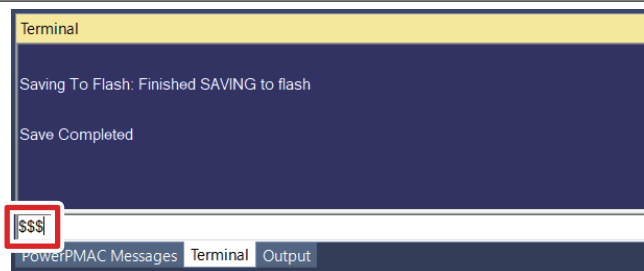
7

Power PMAC IDE の Terminal で [save] コマンドをタイプします。



8

Power PMAC IDE の Terminal で [\$\$\$] コマンドをタイプします。

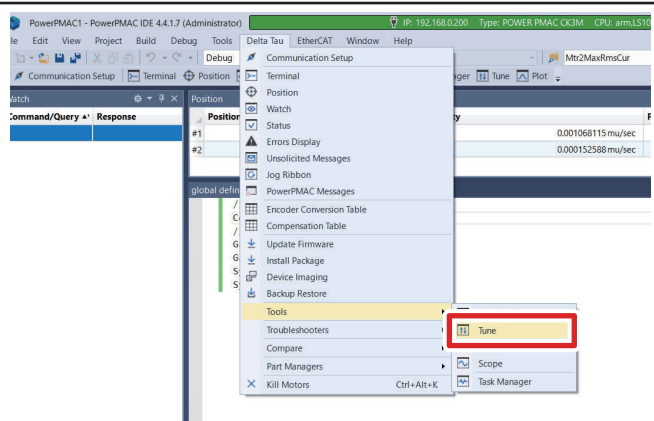


3-4-3 電流ループのチューニング

サーボアンプの電流ループをチューニングします。

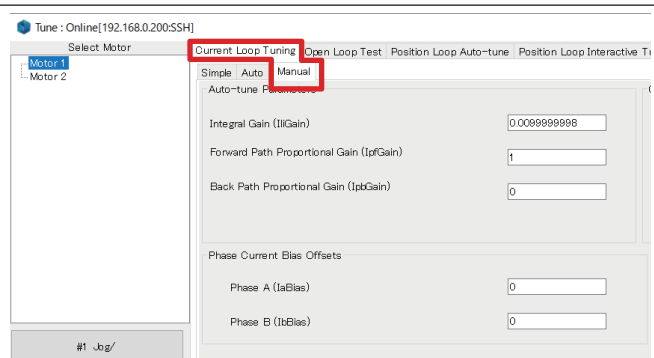
1

[Delta Tau] メニューの [Tools] から [Tune] を選択し、Tune 画面を開きます。

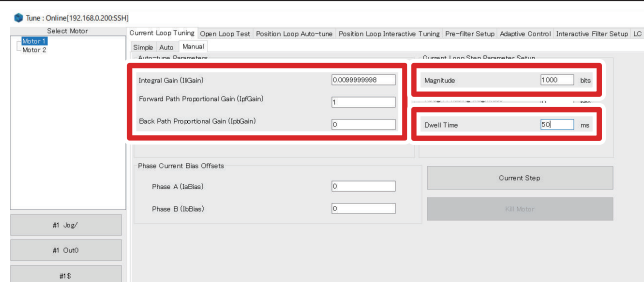


2

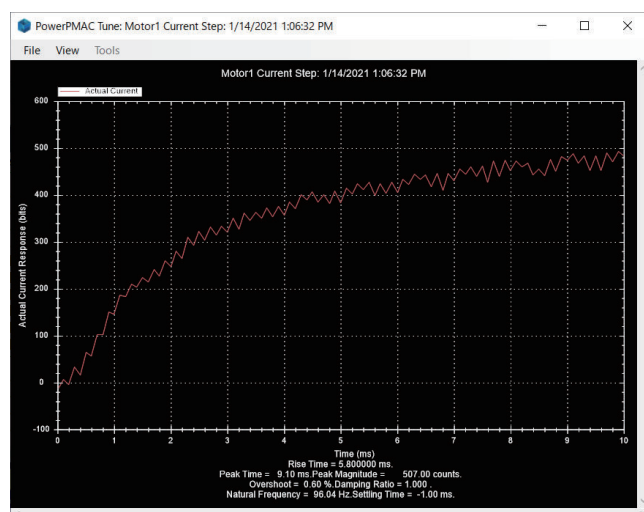
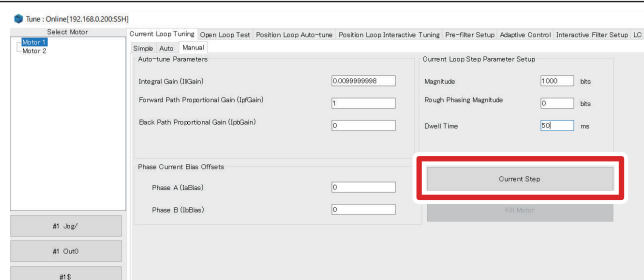
Tune 画面で [Current Loop Tuning] の [Manual] を選択します。



- 3 次のパラメータを設定します。
 [IliGain] : 0.0099999998 (デフォルト)
 [IpfGain] : 1 (デフォルト)
 [IpbGain] : 0
 [Magnitude] : 1000bits
 [Dwell Time] : 50ms



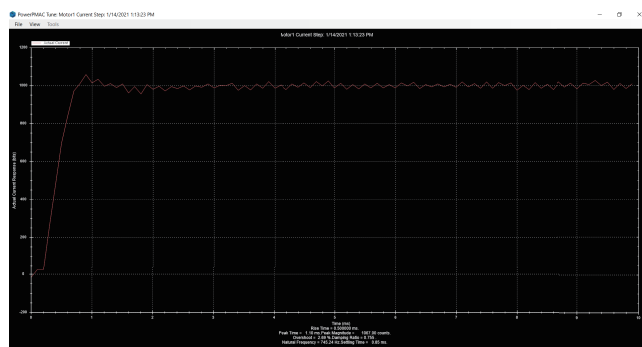
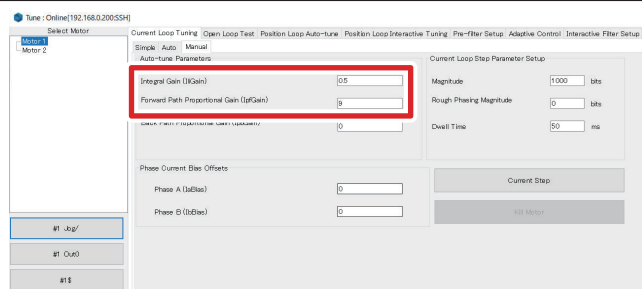
- 4 [Current Step] ボタンをクリックします。
 • 電流のステップ応答が表示されます。



5

目的の応答特性が得られるよう、
[IliGain] と [IpfGain] を調整します。

- 立ち上がりの応答が遅い場合は
[IliGain] を大きくします。
- オーバーシュートや振動が大きい場合は
[IpfGain] を大きくします。
- 各ゲインは小さい値から徐々に大きくしてください。

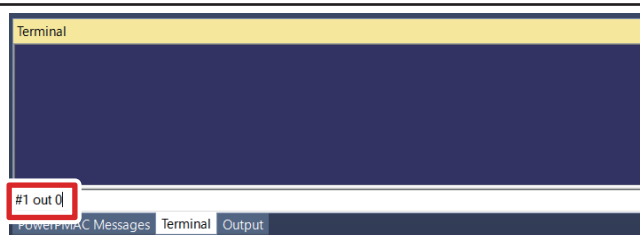


3-4-4 フェーズリファレンスの確立

リニアモータのフェーズリファレンスを確立します。

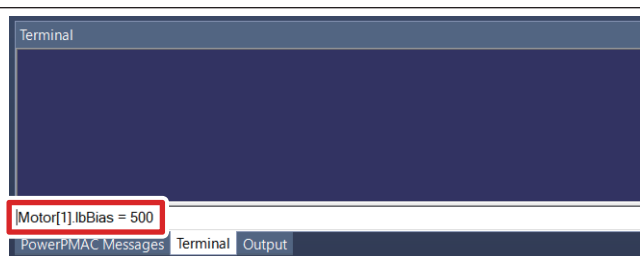
1

Power PMAC IDE の Terminal で [#1 out 0] コマンドをタイプします。



2

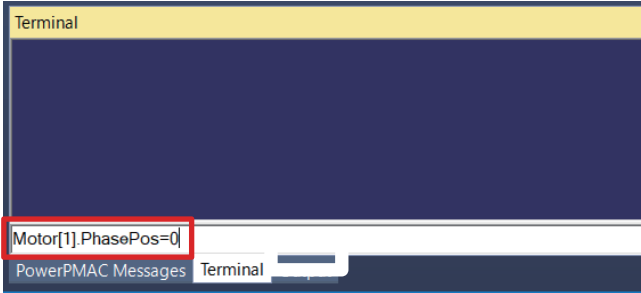
Power PMAC IDE の Terminal で
[Motor[1].lbBias=500] コマンドをタイプします。



3

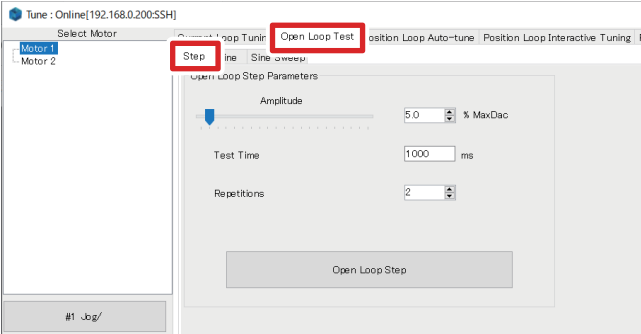
X 軸を手動で動作させ、サーボロックしていることを確認します。

- サーボロックしていない場合は、
Motor[1].lbBias の値を徐々に大きく
設定してください。

4	Power PMAC IDE の Terminal で [Motor[1].PhasePos=0] コマンドを タイプします。	
5	Power PMAC IDE の Terminal で [Motor[1].PhaseFound=1] コマンド をタイプします。	
6	Power PMAC IDE の Terminal で [Motor[1].IbBias=0] コマンドをタイ プします。	
7	Power PMAC IDE の Terminal で [#1 k] コマンドをタイプします。	
8	軸 2 (Y 軸) についても手順 1~7 を同 様に実行します。	

3-4-5 オープンループテスト

リニアモータをオープンループで動作させ、プログラムが正しいことを確認します。

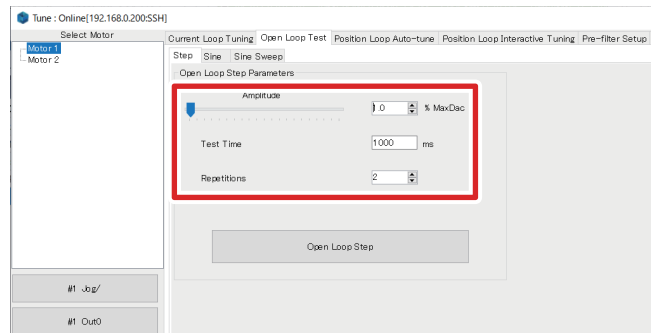
1	[Delta Tau] メニューの [Tools] から [Tune] を選択し、Tune 画面を開き、 [Open Loop Test] の [Step] を選択 します。	
---	--	--

2

次のチューニングパラメータを設定します。

[Amplitude] : 1.0%*1
[Test Time] : 1,000ms
[Repetitions] : 2

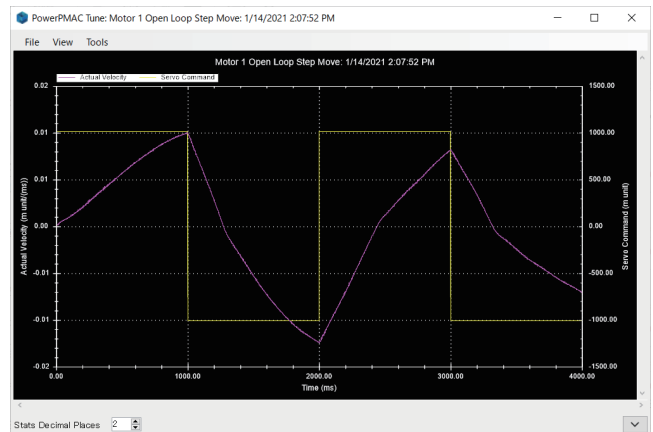
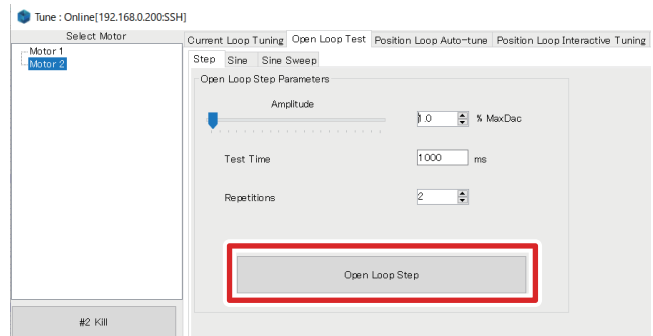
*1. モータが回転しない場合は大きな値を設定してください。



3

[Open Loop Step] ボタンをクリックします。

- モータが往復動作し、右のようなテスト結果が表示されます。
- モータが回転しない場合は、[Test Amplitude] を大きな値に変更してください。
- 今回のテスト結果は、[Test Amplitude] が 8.0% の場合になります。

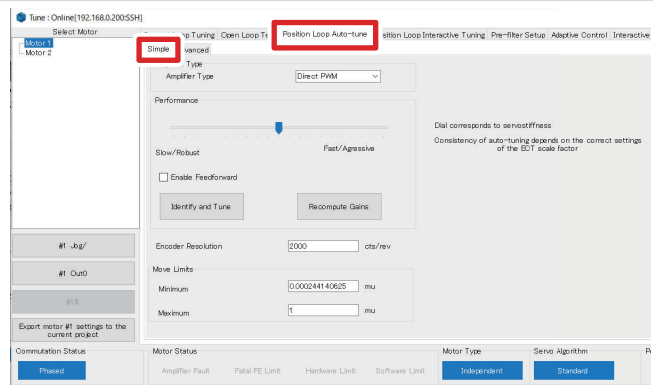


3-4-6 位置ループのオートチューニング

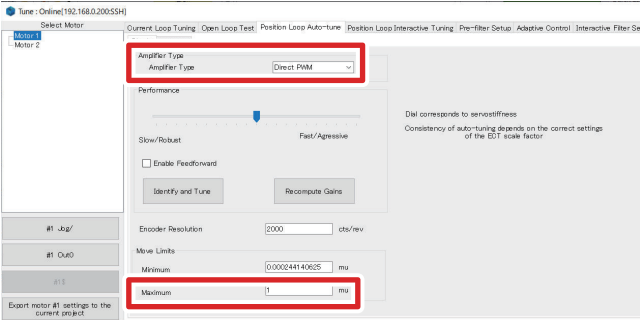
リニアモータにおける位置ループのオートチューニングを行います。

1

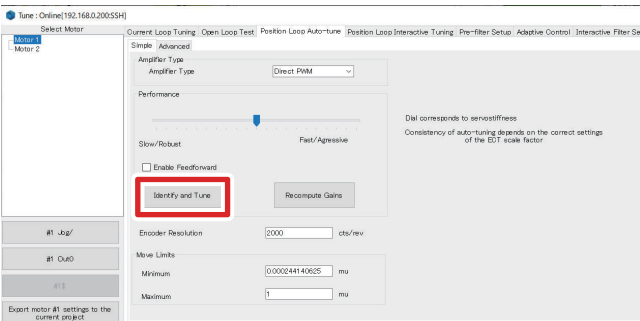
Tune 画面で [Position Loop Auto-tune] の [Simple] を選択します。



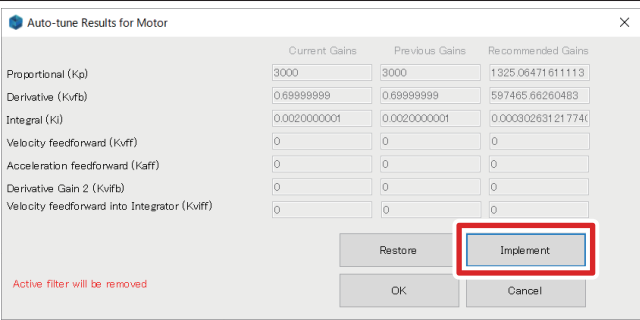
2 次のパラメータを設定します。
[Amplifier Type] : Direct PWM
[Maximum] : 1mu(1mm)



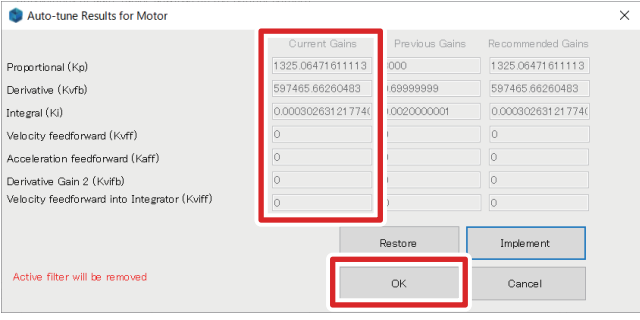
3 [Identify and Tune] ボタンをクリックします。



4 右の画面が表示されたら、[Implement] ボタンをクリックします。



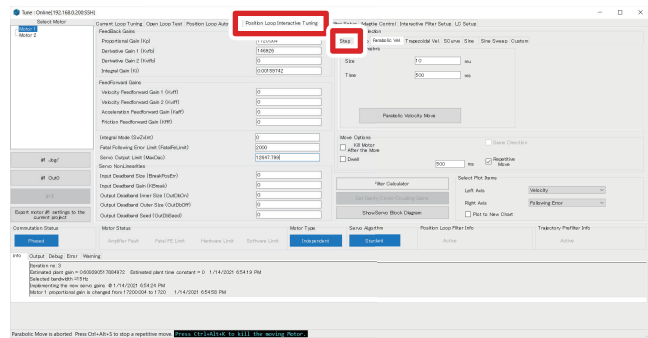
5 [Current Gains] に [Recommended Gains] の値が反映されていることを確認し、[OK] ボタンをクリックします。



3-4-7 位置ループのインタラクティブチューニング

リニアモータにおける位置ループのインタラクティブチューニングを行います。

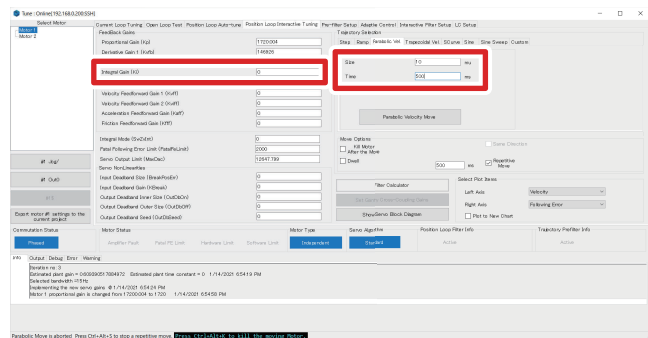
- 1 Tune 画面で [Position Loop Interactive Tuning] の [Step] を選択します。



- 2 次のパラメータを設定します。

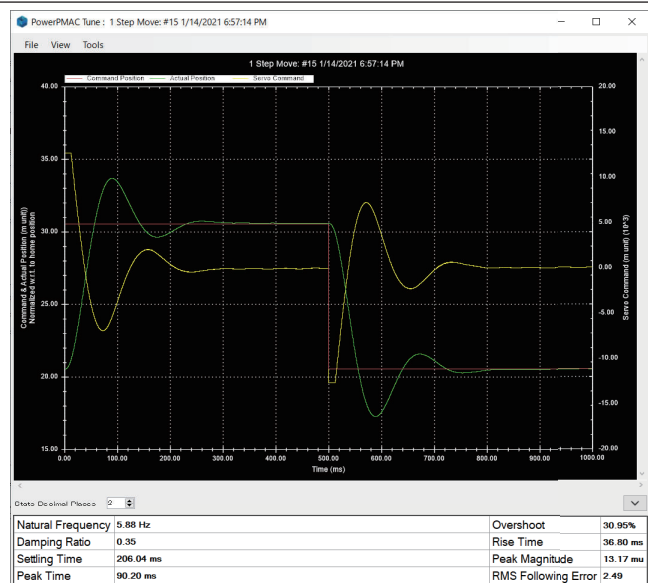
[FeedBack Gains]
[Integral Gains(Ki)] : 0

[Move Parameters]
[Size] : 10mu
[Time] : 1,000ms



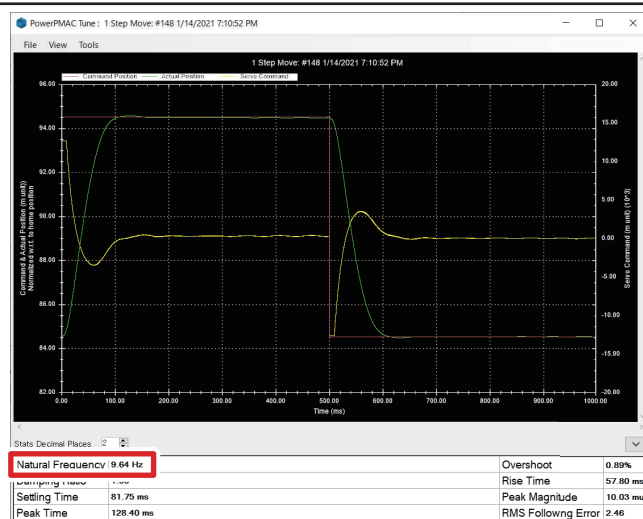
- 3 [Step Move] ボタンをクリックします。

- 4 ステップ応答特性を確認します。

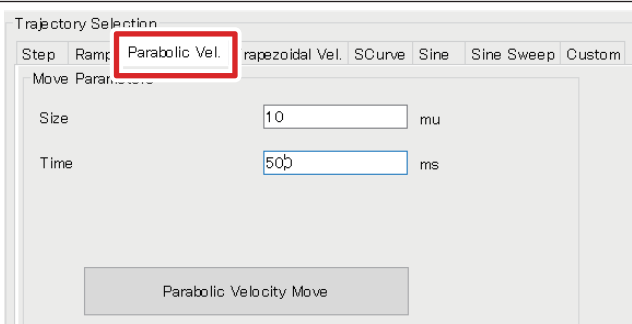


5 目的の応答特性が得られるよう、
[Proportional Gain(Kp)]、
[Derivative Gain 1(Kvfb)]を調整しま
す。

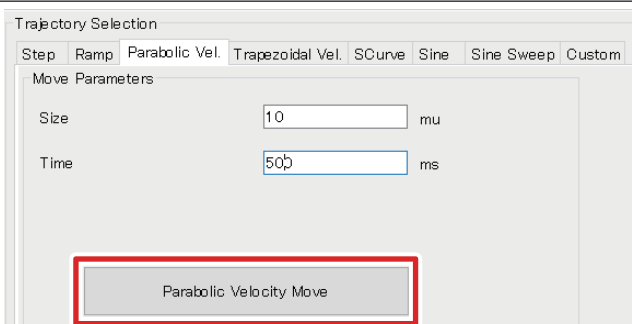
- ・立ち上がりの応答が遅い場合は
[Proportional Gain(Kp)] を大きく
します。
- ・オーバーシュートや振動が大きい場
合は [Derivative Gain 1(Kvfb)] を
大きくします。
- ・定常偏差が大きい場合は [Integral
Gain 1(Ki)] を大きくします。
- ・各ゲインは小さい値から徐々に大き
くしてください。
- ・各パラメータを調整後、[Natural
Frequency]の値が大きくならなくな
ったところで、チューニングを終了し
ます。



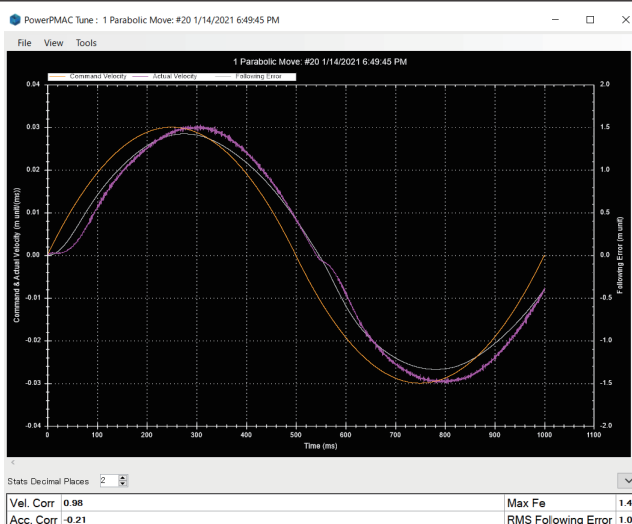
6 [Parabolic Vel.] を選択し、次のパラ
メータを設定します。
[Size] : 10mu(10mm)
[Time] : 500ms
[Left Axis] : Velocity
[Right Axis] : Following Error



7 [Parabolic Velocity Move] ボタンを
クリックします。



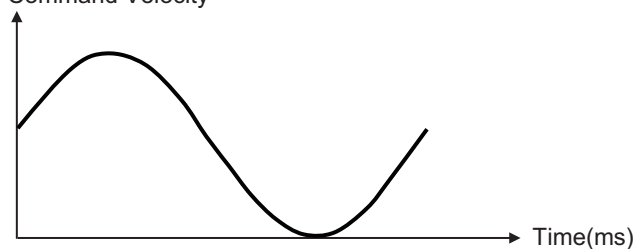
8 速度の放物線応答特性を確認します。



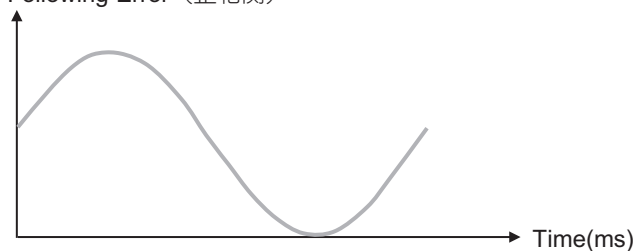
9

[Following Error] が速度に対して正相関を有する場合は [Kvff] を大きくします。逆相関を有する場合は [Kvff] を小さくします。

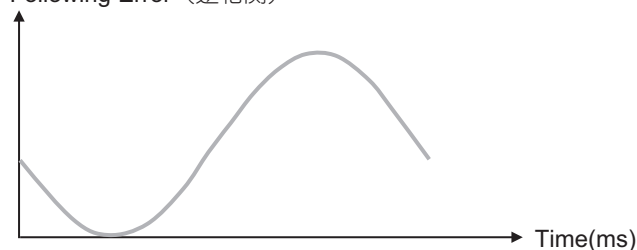
Command Velocity



Following Error (正相関)



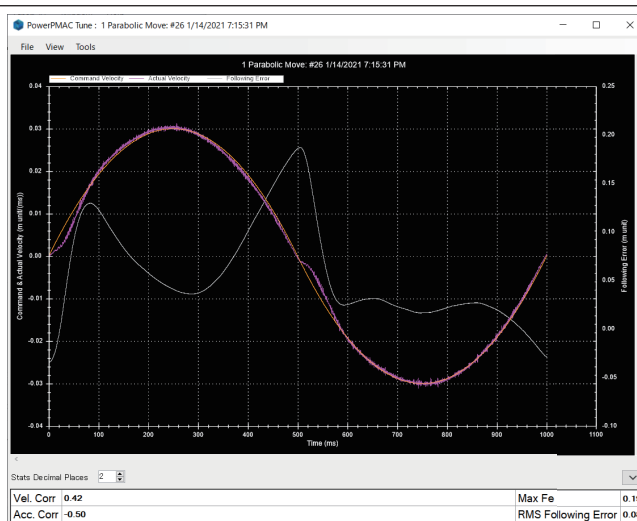
Following Error (逆相関)



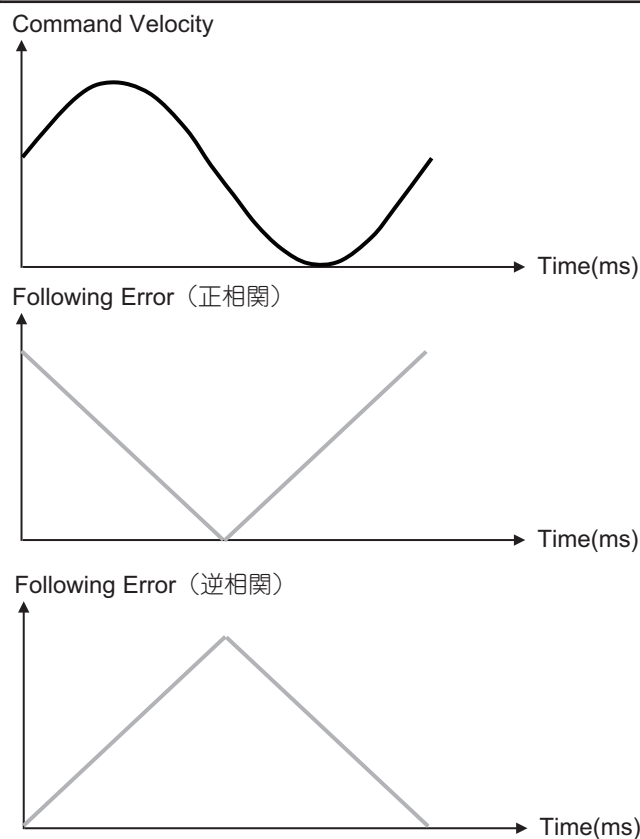
10

再び [Parabolic Velocity Move] ボタンをクリックします。

- これを [Following Error] が速度に対して相関がなくなるまで繰り返します。



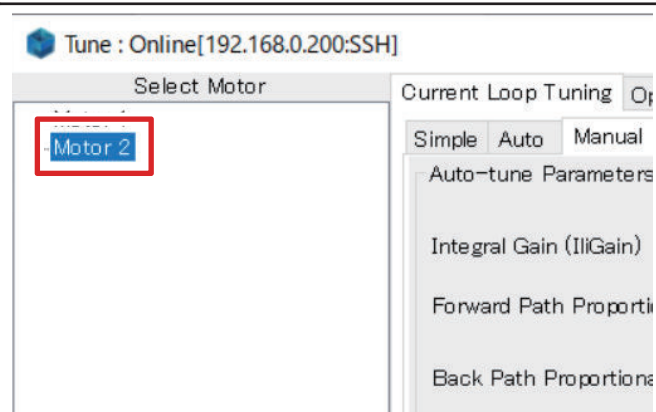
- 11 同様に、[Following Error] が [Acceleration] (加速度) や摩擦に相関を持つ場合は、[Kaff]、[Kfff] の値を増減させます。



3-4-8 Y 軸のチューニング

Y 軸についても同様にチューニングを行います。

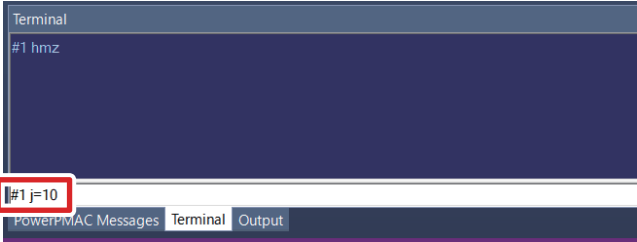
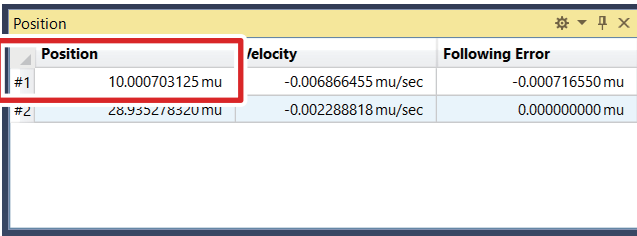
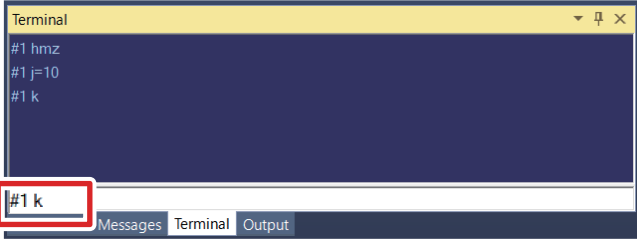
- 1 Tune 画面で [Select Motor] から [Motor2] を選択します。

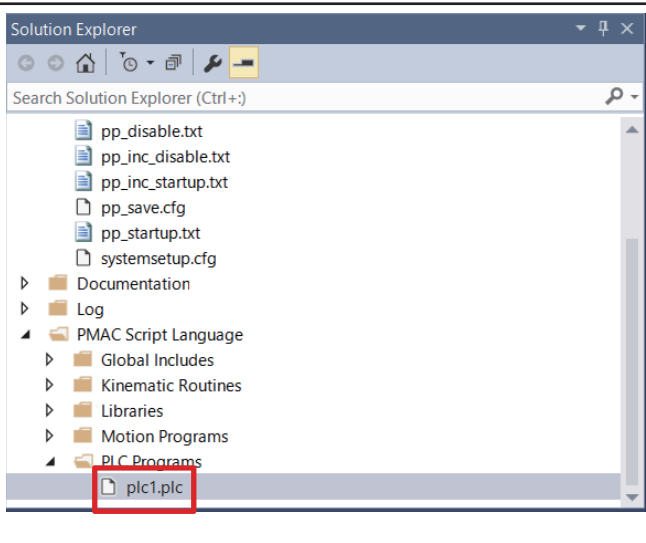


- 2 電流ループのチューニング、オープンループテスト、位置ループのオートチューニング、位置ループのインタラクティブチューニングに記載の内容をそれぞれ実施します。

3-4-9 チューニング結果の確認

チューニング結果が正しいことを確認し、プロジェクトファイルに反映します。

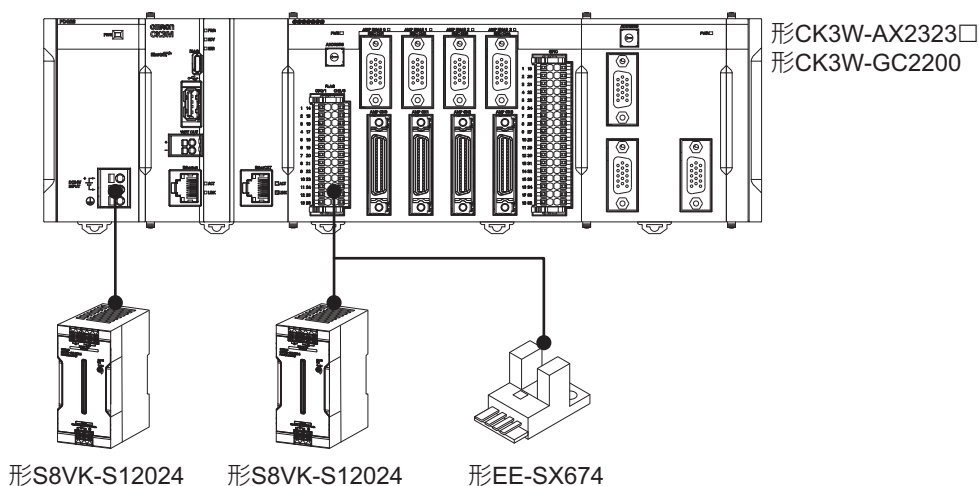
1	Power PMAC IDE の Terminal で [#1 hmz] コマンドをタイプします。 そのあとに、[#1 j=10] コマンドをタイプします。													
2	X 軸が動作していることを確認します。 また、Position で [#1] の [Position] の値が 10.0 付近であることを確認します。	 <table><thead><tr><th></th><th>Position</th><th>Velocity</th><th>Following Error</th></tr></thead><tbody><tr><td>#1</td><td>10.000703125 mu</td><td>-0.006866455 mu/sec</td><td>-0.000716550 mu</td></tr><tr><td>#2</td><td>28.955278320 mu</td><td>-0.002288818 mu/sec</td><td>0.000000000 mu</td></tr></tbody></table>		Position	Velocity	Following Error	#1	10.000703125 mu	-0.006866455 mu/sec	-0.000716550 mu	#2	28.955278320 mu	-0.002288818 mu/sec	0.000000000 mu
	Position	Velocity	Following Error											
#1	10.000703125 mu	-0.006866455 mu/sec	-0.000716550 mu											
#2	28.955278320 mu	-0.002288818 mu/sec	0.000000000 mu											
3	Power PMAC IDE の Terminal で [#1 k] コマンドをタイプし、モータを停止させます。													
4	手順 1～3 を Motor[2]に対しても実施します。													
5	Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [MotorControl.pmh] を開きます。													
6	チューニングにより得たゲインを MotorControl.pmh に追記します。	Motor [1]. liGain=*** Motor [2]. liGain=*** Motor [1]. lpfGain=*** Motor [2]. lpfGain=*** Motor [1]. Servo. Kp=*** Motor [2]. Servo. Kp=*** Motor [1]. Servo. Kvfb=*** Motor [2]. Servo. Kvfb=*** Motor [1]. Servo. Ki=*** Motor [2]. Servo. Ki=*** Motor [1]. Servo. Kaff=*** Motor [2]. Servo. Kaff=*** Motor [1]. Servo. Kvff=*** Motor [1]. Servo. Kfff=***												
7	Solution Explorer の [Configuration] の下にある [pp_startup.txt] を開きます。													
8	右に示すフェーズサーチ実行コマンドを記載します。	enable plc PhaseSearch												

9	Solution Explorer の [PMAC Script Language] - [PLC Programs] の下にある [plc1.plc] を開きます。	
10	右記のプログラムを plc1.plc に追記します。	<pre> open plc PhaseSearch P0=Sys. Time+1. 0; while (P0>Sys. Time) {}; Motor [1]. PhaseFindingStep=1; while (Motor [1]. PhaseFindingStep!=0) {}; Motor [2]. PhaseFindingStep=1; while (Motor [2]. PhaseFindingStep!=0) {}; disable plc PhaseSearch close </pre>
11	Power PMAC IDE の画面右上の Solution Explorer のプロジェクト名を右クリックし、[Build and Download All Programs] を選択し、ビルド&ダウンロードを実行します。 - 手順 5、6 に示すように MotorControl.pmh にゲインを記述することで、ゲインをプログラムとして PMAC にダウンロードできます。 - 手順 7、8 に示すように pp_startup.txt にフェーズサーチ実行コマンドを記述することで、電源投入後またはリセット後に自動的にフェーズサーチを実行し、Motor[1]、Motor[2]を有効にします。	

3-5 オーバトラベルリミットスイッチと原点の設定

3-5-1 各機器の配線

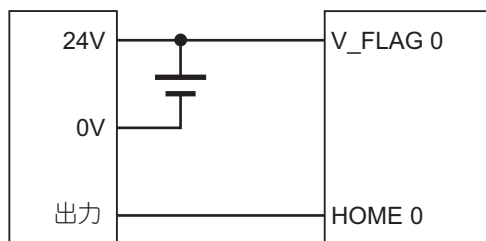
CK3M と各種センサとの配線方法について記載します。



形 EE-SX674 は、形 CK3W-AX2323N の FLAG 端子の以下にそれぞれ接続します。

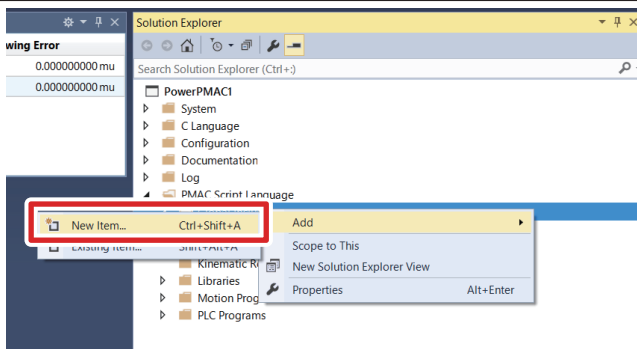
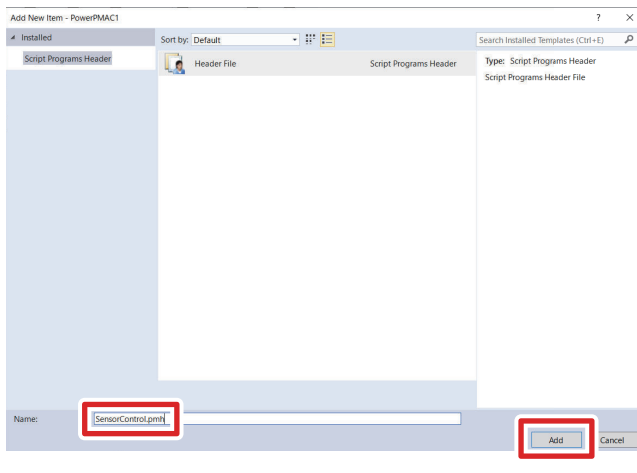
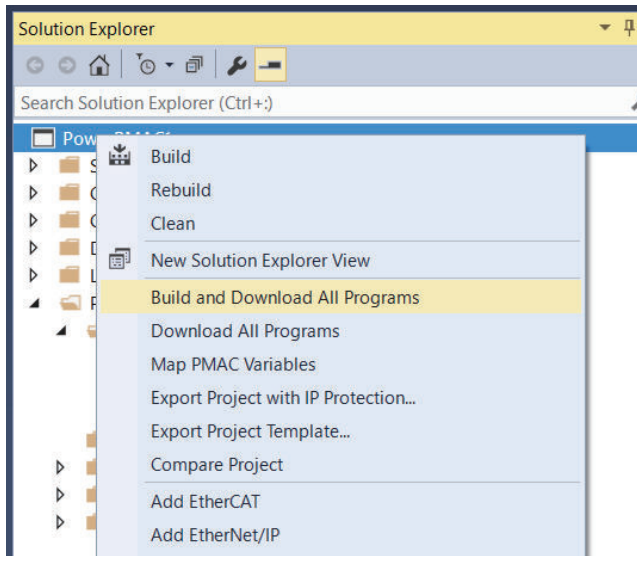
- PLIM0: X 軸の正方向オーバトラベルリミットスイッチ
- NLIM0: X 軸の負方向オーバトラベルリミットスイッチ
- PLIM1: Y 軸の正方向オーバトラベルリミットスイッチ
- NLIM1: Y 軸の負方向オーバトラベルリミットスイッチ

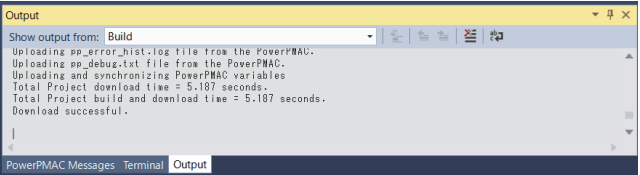
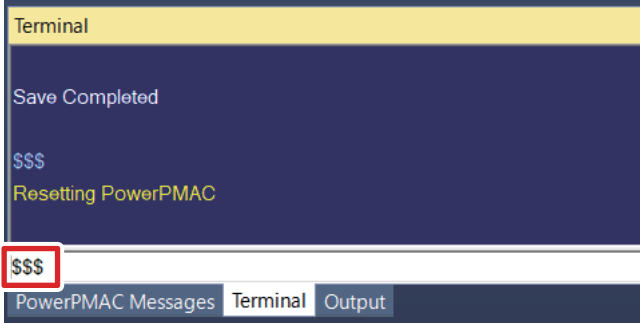
以下は一例として、HOME0 を配線した例を示します。



3-5-2 プログラミング

オーバトラベルリミットスイッチの設定を行います。

1	Solution Explorer の [PMAC Script Language] - [Global Includes] を右クリックし、[Add] から [New Item...] を選択します。	
2	[Name] の欄に「SensorControl.pmh」とタイプし、[Add] ボタンをクリックします。	
3	Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [SensorControl.pmh] を開きます。	
4	右のテキストを SensorControl.pmh に書き込みます。 注釈付きの設定の内容については、「第 4 章 各種設定のカスタマイズ方法 (P.4-1)」を参照してください。	<pre> Sys. WpKey=\$AAAAA //Overtravel Limit Switch Configurations Motor[1].pLimits=Gate3[0].Chan[0].Status.a;/**1 Motor[2].pLimits=Gate3[0].Chan[1].Status.a;/**1 Motor[1].LimitBits=9;/**2 Motor[2].LimitBits=9;/**2 </pre>
5	Power PMAC IDE の画面右上の Solution Explorer のプロジェクト名を右クリックし、[Build and Download All Programs] を選択し、ビルド＆ダウンロードを実行します。	

6	Output で異常がないことを確認します。	
7	Power PMAC IDE の Terminal で [save] コマンドをタイプします。	
8	Power PMAC IDE の Terminal で [\$\$\$] コマンドをタイプします。	

3-5-3 設定の確認（オーバトラベルリミットスイッチ）

オーバトラベルリミットスイッチの設定が正しく動作することを確認します。

1 以下の変数がすべて 0 であることを確認します。

- 「Motor[1].PlusLimit」
- 「Motor[1].MinusLimit」
- 「Motor[2].PlusLimit」
- 「Motor[2].MinusLimit」

Watch 🔍 📄 ✕	
Command/Query ▲▼	Response
Motor[1].PlusLimit	0
Motor[1].MinusLimit	0
Motor[2].PlusLimit	0
Motor[2].MinusLimit	0

2 X 軸の正方向オーバトラベルリミットスイッチの「Motor[1].PlusLimit」が 1 となることを確認します。

Watch 🔍 📄 ✕	
Command/Query ▲▼	Response
Motor[1].PlusLimit	1
Motor[1].MinusLimit	0
Motor[2].PlusLimit	0
Motor[2].MinusLimit	0

3 X 軸の負方向オーバトラベルリミットスイッチの「Motor[1].MinusLimit」が 1 となることを確認します。

Watch ⚙️ ▾ 🔍 ✕	
Command/Query ▲▼	Response
Motor[1].PlusLimit	0
Motor[1].MinusLimit	1
Motor[2].PlusLimit	0
Motor[2].MinusLimit	0

4 Y 軸の正方向オーバトラベルリミットスイッチの「Motor[2].PlusLimit」が 1 となることを確認します。

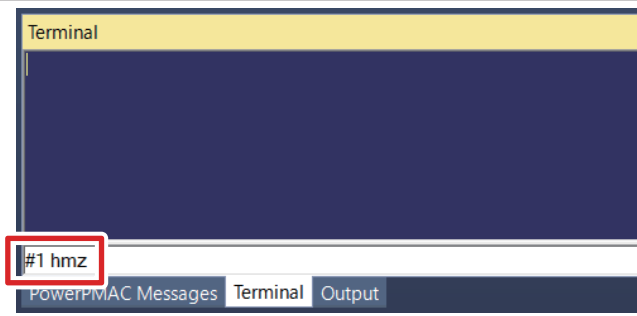
Watch ⚙️ ▾ 🔍 ✕	
Command/Query ▲▼	Response
Motor[1].PlusLimit	0
Motor[1].MinusLimit	0
Motor[2].PlusLimit	1
Motor[2].MinusLimit	0

5 Y 軸の負方向オーバトラベルリミットスイッチの「Motor[2].MinusLimit」が 1 となることを確認します。

Watch ⚙️ ▾ 🔍 ✕	
Command/Query ▲▼	Response
Motor[1].PlusLimit	0
Motor[1].MinusLimit	0
Motor[2].PlusLimit	0
Motor[2].MinusLimit	1

3-5-4 原点の設定

原点の設定を行います。

1	原点位置まで X 軸を移動し、Power PMAC IDE の Terminal で [#1 hmz] コマンドをタイプします。	 <pre>#1 hmz</pre>
---	--	--

2

以下の変数が正しく反映されていることを確認します。

- 「Motor[1].HomePos」：原点トリガ位置
- 「Motor[1].HomeComplete」：1

Watch	
Command/Query ▲▼	Response
Motor[1].HomePos	2.71250000000000036
Motor[1].HomeComplete	1

3

原点位置まで Y 軸を移動し、Power PMAC IDE の Terminal で [#2 hmz] コマンドをタイプします。

Terminal	
#2 hmz	
PowerPMAC Messages Terminal Output	

4

以下の変数が正しく反映されていることを確認します。

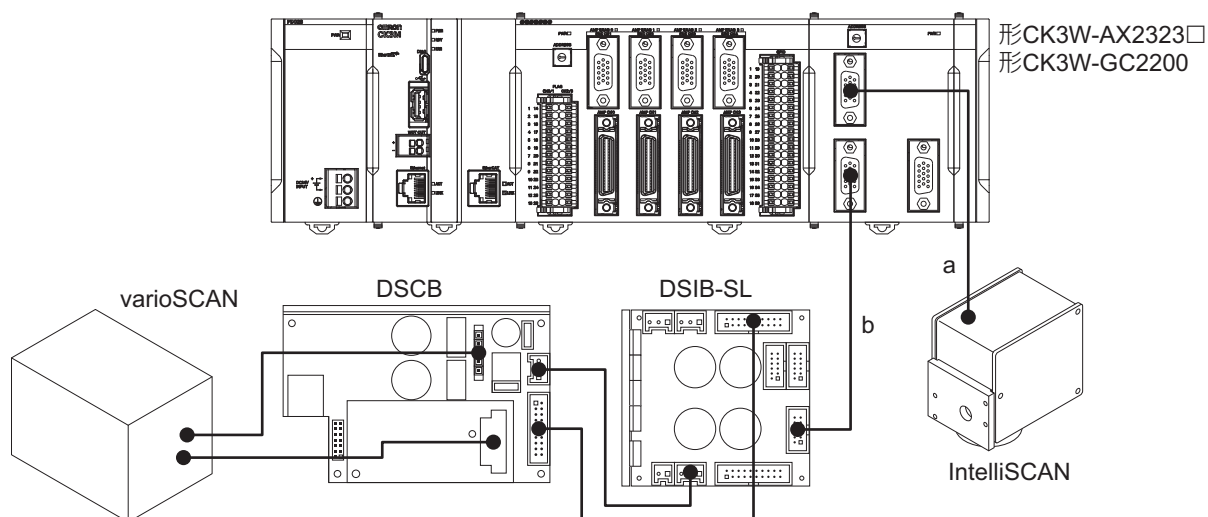
- 「Motor[2].HomePos」：原点トリガ位置
- 「Motor[2].HomeComplete」：1

Watch	
Command/Query ▲▼	Response
Motor[2].HomePos	112.373033447265627
Motor[2].HomeComplete	1

3-6 ガルバノスキャナの設定

3-6-1 各機器の配線

CK3M とガルバノスキャナとの配線方法について記載します。



上図 a、b の箇所を以下の要領に従って接続します。

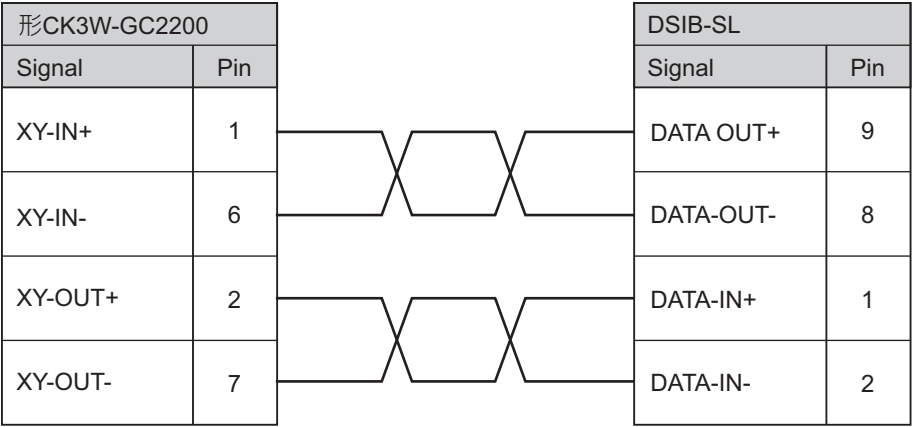
a. コントローラとガルバノスキャナとの接続

形 CK3W-GC2200 の XY コネクタと intelliSCAN の SL2-100 コネクタを専用ケーブル形 CK3W-CAG03A で接続します。

形CK3W-GC2200				IntelliSCAN	
Signal	Pin			Signal	Pin
XY-IN+	1			DATA OUT+	5
XY-IN-	6			DATA-OUT-	9
XY-OUT+	2			DATA-IN+	1
XY-OUT-	7			DATA-IN-	6

b. コントローラとダイナミックフォーカシングユニットとの接続

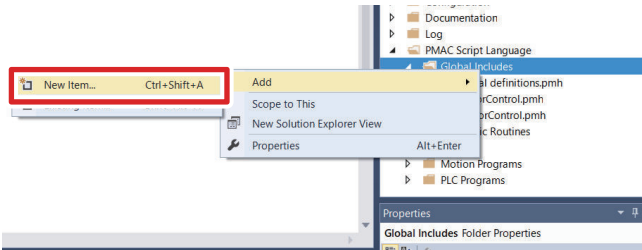
形 CK3W-GC2200 の Z コネクタと DSIB-SL の DIGITAL IN コネクタを専用ケーブル形 CK3W-CAG03A で接続します。



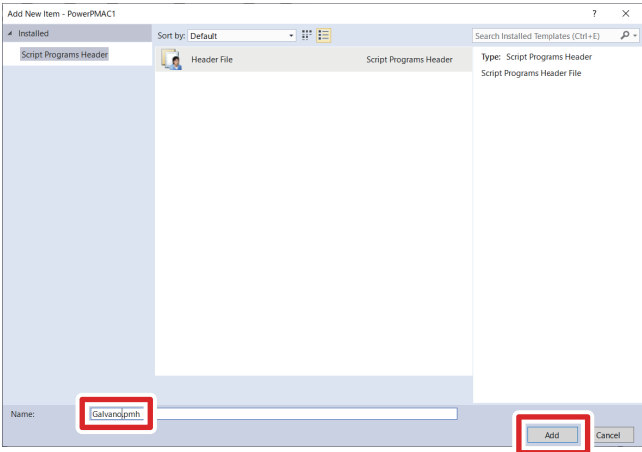
3-6-2 プログラミング

ガルバノスキャナの設定をプログラミングします。

- 1** Solution Explorer の [PMAC Script Language] - [Global Includes] を右クリックし、[Add] から [New Item...] を選択します。



- 2** [Name] の欄に「Galvano.pmh」とタイプし、[Add] ボタンをクリックします。



- 3** Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [Galvano.pmh] を開きます。

4

右のテキストを Galvano.pmh に書き込みます。

- 注釈付きの設定の内容については、「第 4 章 各種設定のカスタマイズ方法 (P.4-1)」を参照してください。

```
Sys. WpKey=$AAAAA

//SL2-100 Configurations
Gate3[1]. SerialEncCtrl=$82800//*1

//Servo Task Configurations
Motor[3]. ServoCtrl=1//*2
Motor[4]. ServoCtrl=1//*2
Motor[5]. ServoCtrl=1//*2
Motor[3]. Ctrl=Sys. PosCtrl//*3
Motor[4]. Ctrl=Sys. PosCtrl//*3
Motor[5]. Ctrl=Sys. PosCtrl//*3
Motor[3]. MaxSpeed=0//*4
Motor[4]. MaxSpeed=0//*4
Motor[5]. MaxSpeed=0//*4

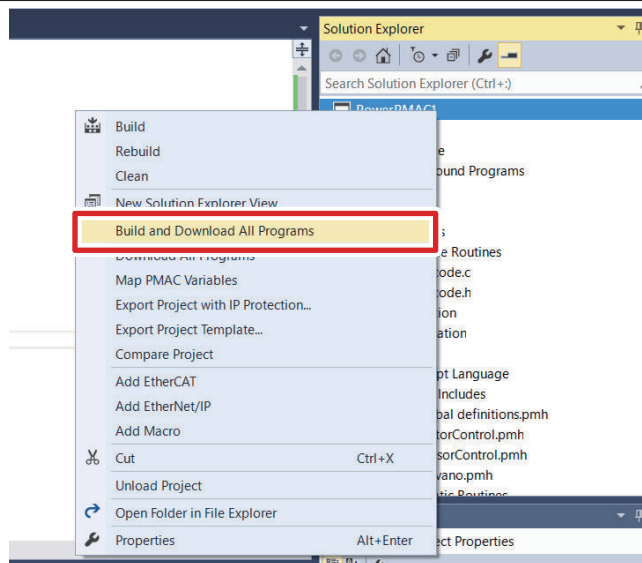
//Motor Output Configurations
Motor[3]. MaxPos=0//*5
Motor[4]. MaxPos=0//*5
Motor[5]. MaxPos=0//*5
Motor[3]. MinPos=0//*6
Motor[4]. MinPos=0//*6
Motor[5]. MinPos=0//*6
Motor[3]. pDac=Gate3[1]. Chan[0]. Dac[0]. a//*7
Motor[4]. pDac=Gate3[1]. Chan[1]. Dac[0]. a//*7
Motor[5]. pDac=Gate3[1]. Chan[2]. Dac[0]. a//*7
Motor[3]. pLimits=0//*8
Motor[4]. pLimits=0//*8
Motor[5]. pLimits=0//*8
Motor[3]. FatalFeLimit=0//*9
Motor[4]. FatalFeLimit=0//*9
Motor[5]. FatalFeLimit=0//*9

//Encoder Input Configurations
Motor[3]. pEnc=EncTable[3]. a//*10
Motor[4]. pEnc=EncTable[4]. a//*10
Motor[5]. pEnc=EncTable[5]. a//*10
Motor[3]. pEnc2=EncTable[3]. a//*11
Motor[4]. pEnc2=EncTable[4]. a//*11
Motor[5]. pEnc2=EncTable[5]. a//*11
EncTable[3]. Type=1//*12
EncTable[4]. Type=1//*12
EncTable[5]. Type=1//*12
EncTable[3]. pEnc=Gate3[1]. Chan[0]. SerialEncDataA. a//*13
EncTable[4]. pEnc=Gate3[1]. Chan[1]. SerialEncDataA. a//*13
EncTable[5]. pEnc=Gate3[1]. Chan[2]. SerialEncDataA. a//*13
EncTable[3]. ScaleFactor=1/4096//*14
EncTable[4]. ScaleFactor=1/4096//*14
EncTable[5]. ScaleFactor=1/4096//*14

//Enable Motor 6
Motor[6]. ServoCtrl=1//*2
Motor[6]. pDac=Sys. Udata[0]. a
```

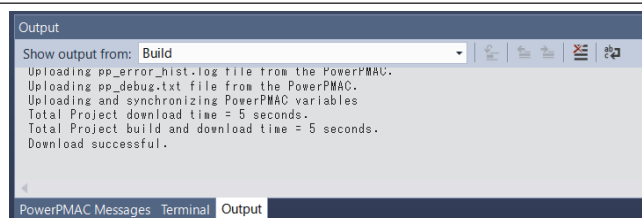
5

Power PMAC IDE の画面右上の Solution Explorer のプロジェクト名を右クリックし、[Build and Download All Programs] を選択し、ビルド&ダウンロードを実行します。



6

Output で異常がないことを確認します。

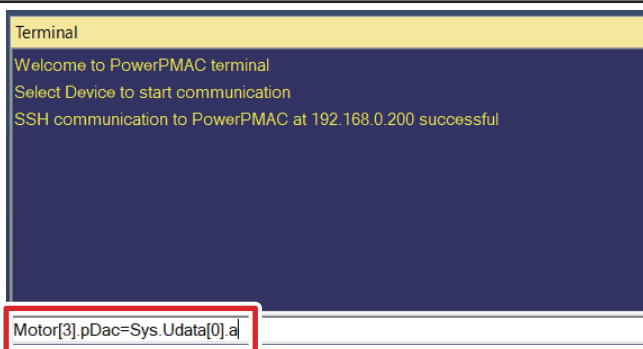


3-6-3 コントロールコマンドの設定

ガルバノスキャナのフィードバック信号を現在位置に設定します。

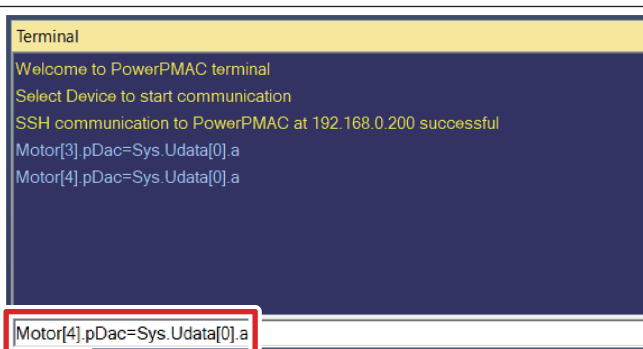
1

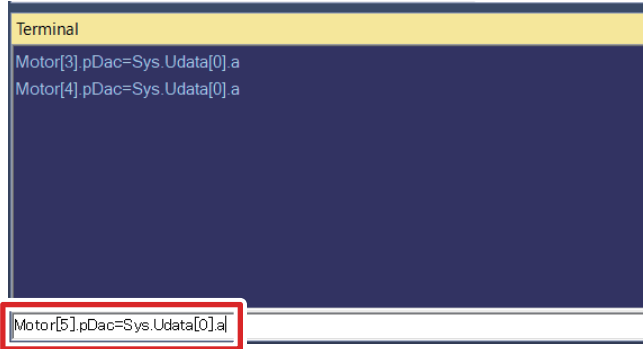
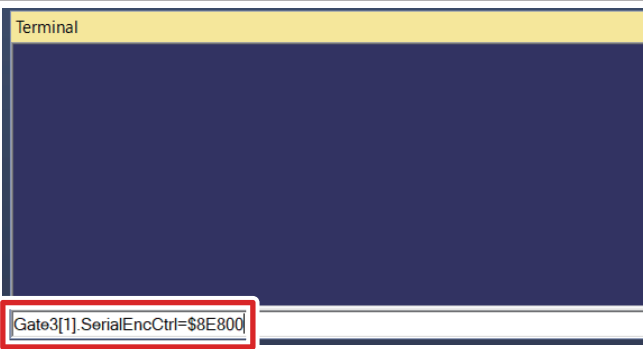
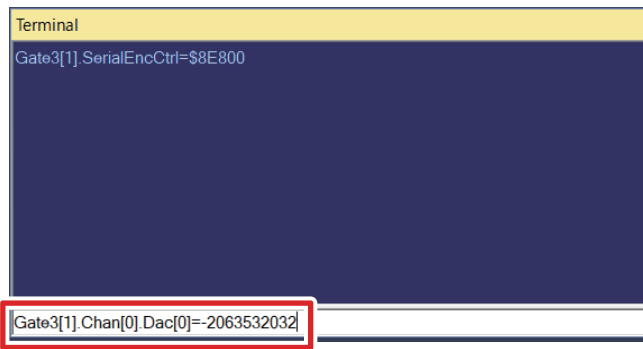
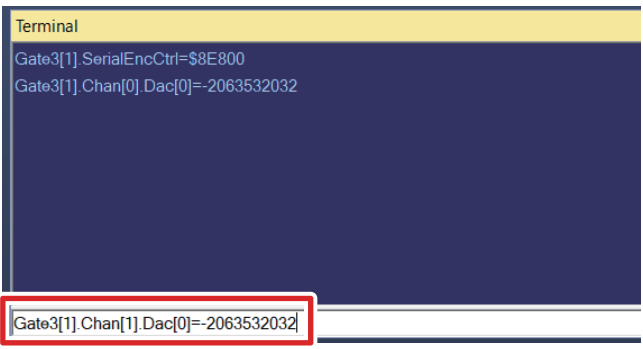
Power PMAC IDE の Terminal で [Motor[3].pDac=Sys.Udata[0].a] コマンドをタイプします。

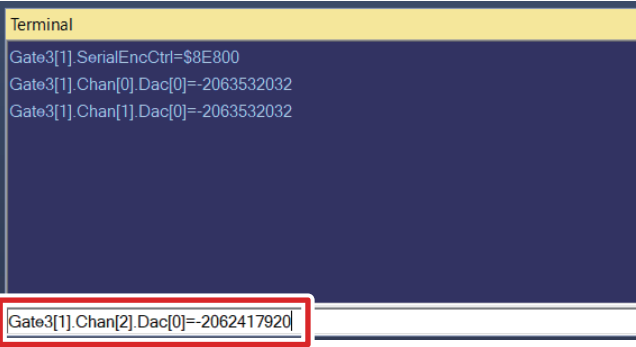


2

Power PMAC IDE の Terminal で [Motor[4].pDac=Sys.Udata[0].a] コマンドをタイプします。

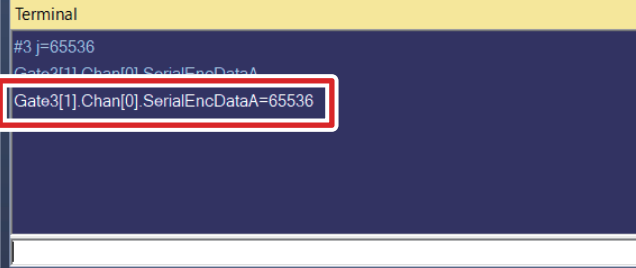


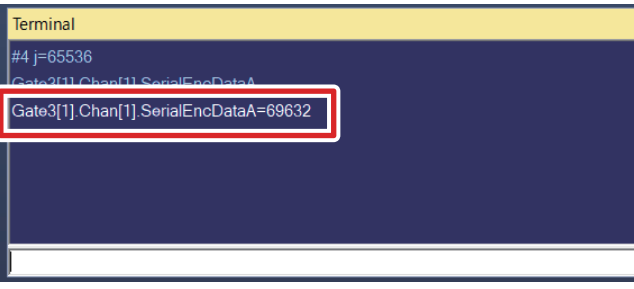
3	Power PMAC IDE の Terminal で [Motor[5].pDac=Sys.Udata[0].a] コ マンドをタイプします。	
4	Power PMAC IDE の Terminal で [Gate3[1].SerialEncCtrl=\$8E800] コ マンドをタイプします。	
5	Power PMAC IDE の Terminal で [Gate3[1].Chan[0].Dac[0]=-20635 32032] コマンドをタイプします。	
6	Power PMAC IDE の Terminal で [Gate3[1].Chan[1].Dac[0]=-20635 32032] コマンドをタイプします。	

7	Power PMAC IDE の Terminal で [Gate3[1].Chan[2].Dac[0]=-2062417920] コマンドをタイプします。	 Terminal <pre>Gate3[1].SerialEncCtrl=\$8E800 Gate3[1].Chan[0].Dac[0]=-2063532032 Gate3[1].Chan[1].Dac[0]=-2063532032 Gate3[1].Chan[2].Dac[0]=-2062417920</pre>
8	Power PMAC IDE の Terminal で [\$ \$] コマンドをタイプします。	 Terminal <pre>\$ \$</pre>

3-6-4 設定の確認

ガルバノスキャナの設定が正しいことを確認します。

1	Power PMAC IDE の Terminal で [#3 j=65536] コマンドをタイプします。	 Terminal <pre>#3 j=65536</pre>
2	Gate3[0].Chan[0].SerialEncDataA=65536 付近となることを確認します。	 Terminal <pre>#3 j=65536 Gate2[1].Chan[0].SerialEncDataA= Gate3[1].Chan[0].SerialEncDataA=65536</pre>

3	Power PMAC IDE の Terminal で [#4 j=65536] コマンドをタイプします。	
4	Gate3[1].Chan[1].SerialEncDataA=65536 付近となることを確認します。	
5	Power PMAC IDE の Terminal で [#5 j=65536] コマンドをタイプします。	
6	Gate3[0].Chan[2].SerialEncDataA=65536 付近となることを確認します。	

3-6-5 MOTF の実機検証

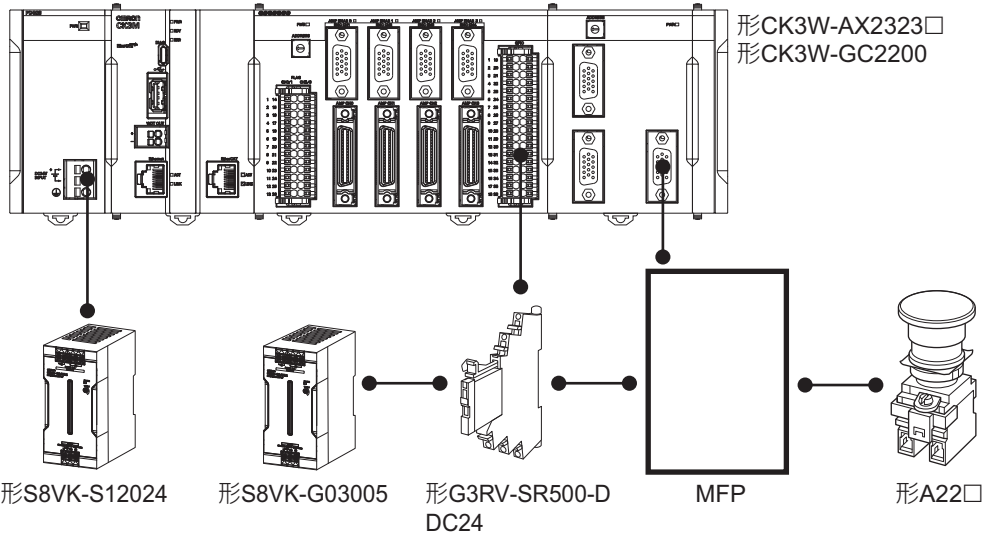
MOTF の設定が実機で正しく動作していることを確認します。

1	Power PMAC IDE の Terminal で [&1 start 1] コマンドをタイプします。	
2	X、Y 軸のリニアモータとガルバノスキャナが同期して動作することを確認します。	

3-7 レーザ発振器の設定

3-7-1 各機器の配線

CK3M とレーザ発振器との配線方法について記載します。



形 CK3W-AX2323N の GPIO 端子と MFP は、それぞれ以下のように接続します。

- OUT 0: MFP の PIN5
- OUT 1: MFP の PIN6
- OUT 2: MFP の PIN7
- OUT 3: MFP の PIN8
- OUT 4: MFP の PIN9
- OUT 5: MFP の PIN18
- OUT 6: MFP の PIN22

形 CK3W-GC2200 の LASER 端子と MFP は、それぞれ以下のように接続します。

- OUT 0: MFP の PIN20
- OUT 1: MFP の PIN21

3-7-2 ガイドレーザの出力

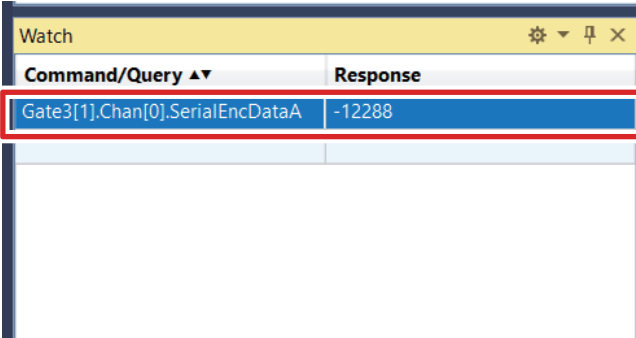
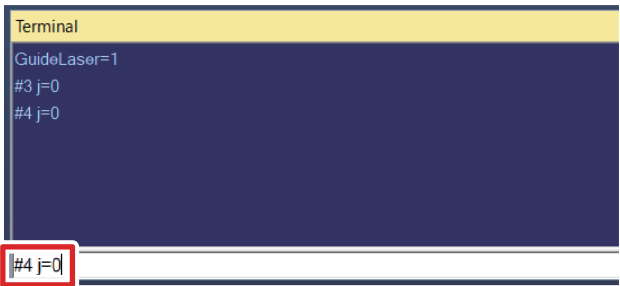
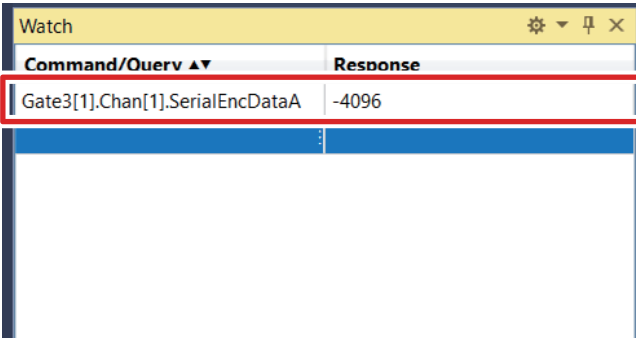
ガイドレーザが正常に出力されることを確認します。

1

Solution Explorer の [PMAC Script Language] - [Global Includes] を右クリックし、[Add] から [New Item...] を選択します。

2

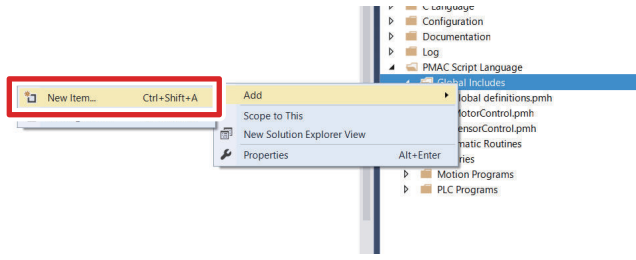
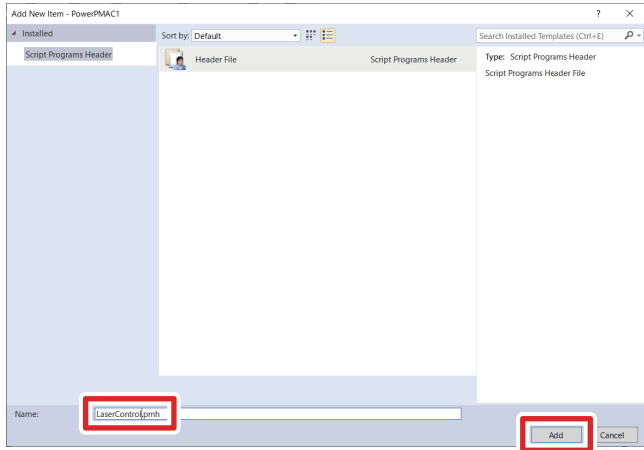
[Name] の欄に「LaserDefinitions.pmh」とタイプし、[Add] ボタンをクリックします。

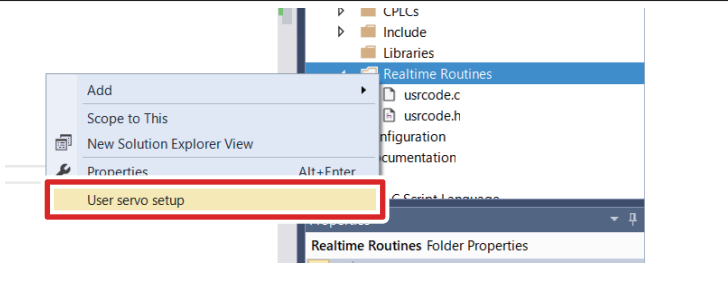
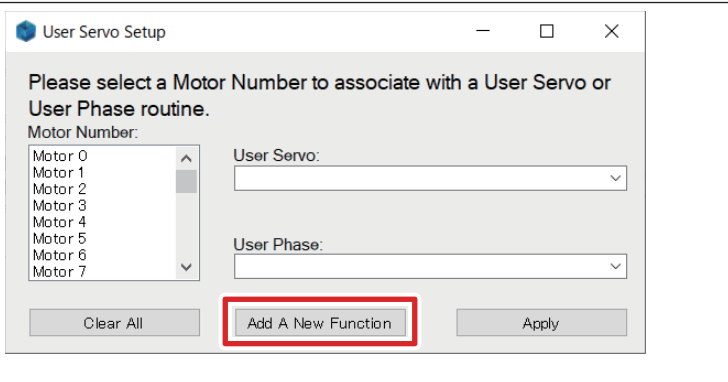
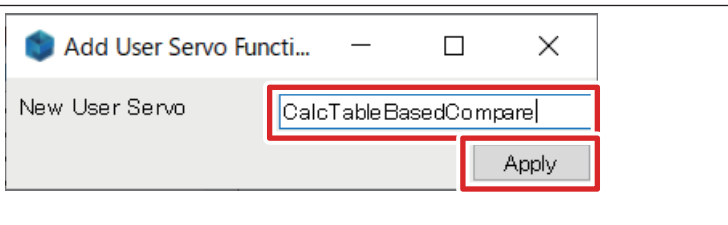
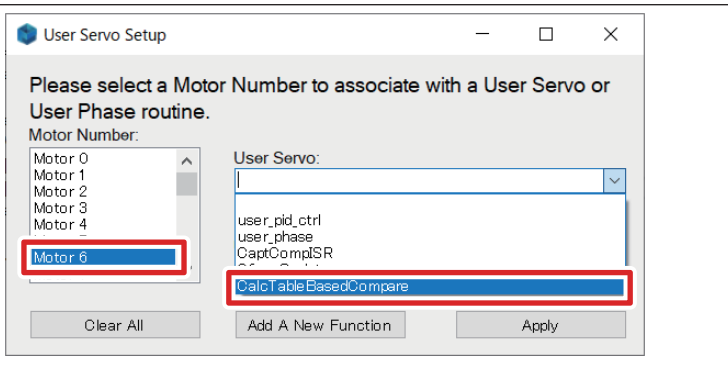
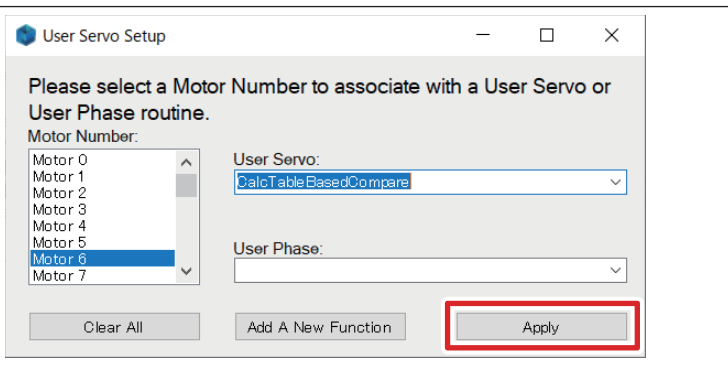
3	Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [LaserControl.pmh] を開き、右のテキストを追記します。	<pre>ptr LaserPower->Gate3[0].GpioData[0]. 16. 4 ptr LaserLatch->Gate3[0].GpioData[0]. 20. 1 ptr MasterOscillator->Gate3[0].GpioData[0]. 21. 1 ptr GuideLaser->Gate3[0].GpioData[0]. 22. 1</pre>
4	Power PMAC IDE の Terminal で [GuideLaser=1] コマンドをタイプし、ガイドレーザが出力されていることを確認します。	
5	Power PMAC IDE の Terminal で [#3 j=0] コマンドをタイプし、Gate3[1].Chan[0].SerialEncDataA=0 付近となることを確認します。	 
6	Power PMAC IDE の Terminal で [#4 j=0] コマンドをタイプし、Gate3[1].Chan[1].SerialEncDataA=0 となることを確認します。	 

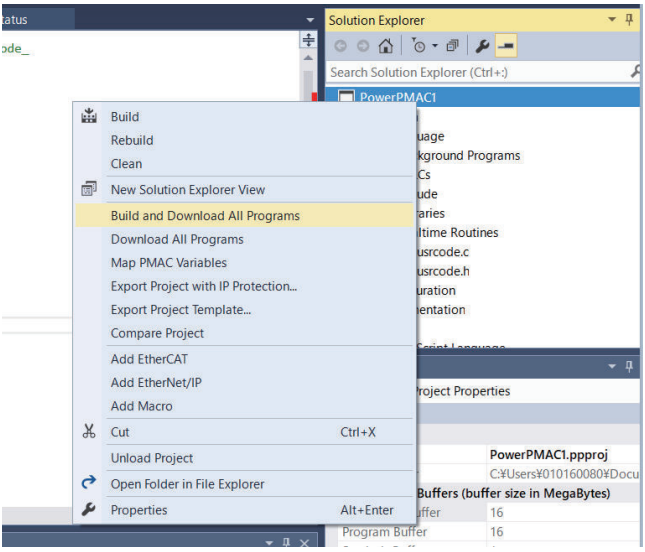
7	XY テーブルの中央にガイドレーザが出力されていることを確認します。
8	Power PMAC IDE の Terminal で [#5 j={位置}] コマンドをタイプし、ガイドレーザのスポット径が適切であることを確認します。
9	Power PMAC IDE の Terminal で [GuideLaser=0] コマンドをタイプし、ガイドレーザが出力されていないことを確認します。 

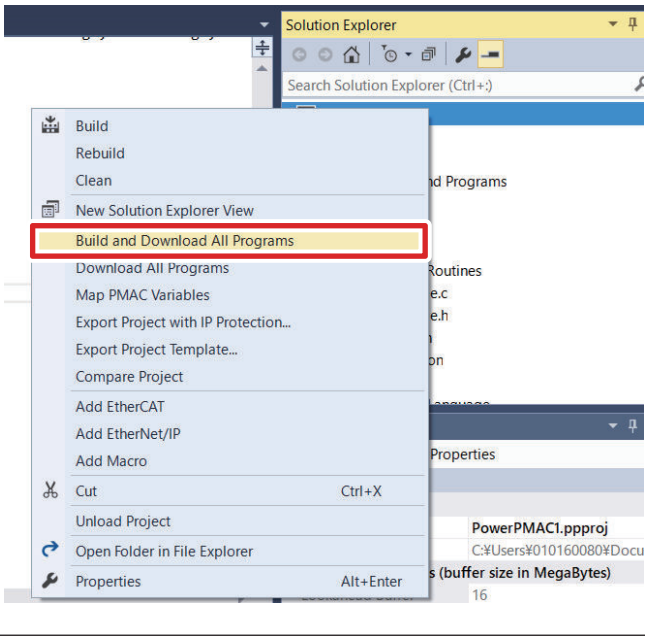
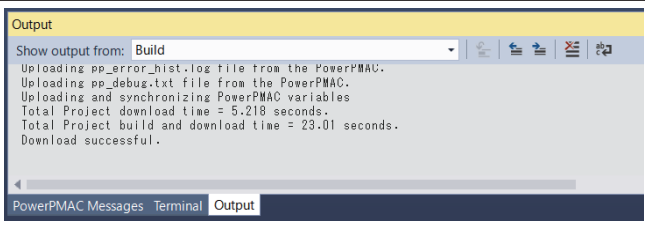
3-7-3 プログラミング

レーザ発振器の設定をプログラミングします。

1	Solution Explorer の [PMAC Script Language] - [Global Includes] を右クリックし、[Add] から [New Item...] を選択します。 
2	[Name] の欄に「LaserControl.pmh」とタイプし、[Add] ボタンをクリックします。 
3	Solution Explorer の [PMAC Script Language] - [Global Includes] の下にある [LaserControl.pmh] を開き、右のテキストを追記します。 注釈付きの設定の内容については、「第 4 章 各種設定のカスタマイズ方法 (P.4-1)」を参照してください。 <pre>Sys. WpKey=\$AAAAAAA //Q-Switch Configurations Gate3[1]. Chan[0]. CompA=\$8000D000//*1 Gate3[1]. Chan[1]. CompA=\$0//*2 //Laser Output Configurations Gate3[1]. Chan[2]. CompA=\$FFF00//*3</pre>

4	Solution Explorer の [C Language] - [Realtime Routines] で右クリックし、[User servo setup] を選択します。	
5	[Add a New Function] ボタンをクリックします。	
6	[CalcTableBasedCompare] とタイプし、[Apply] ボタンをクリックします。	
7	[Motor 6] を選択し、[User Servo] から、[CalcTableBasedCompare] を選択します。	
8	[Apply] ボタンをクリックします。	

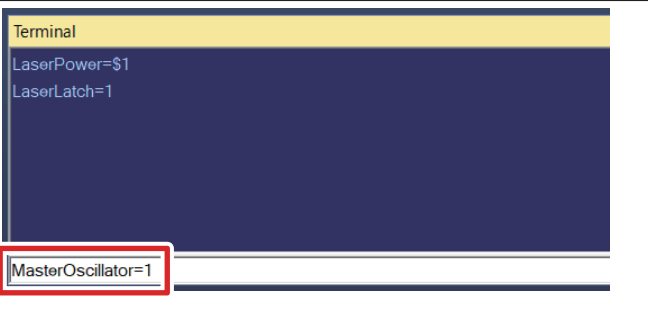
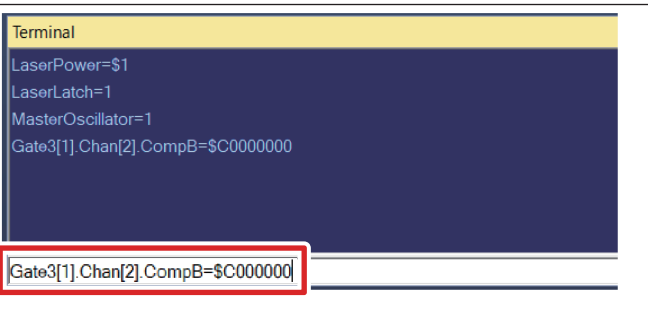
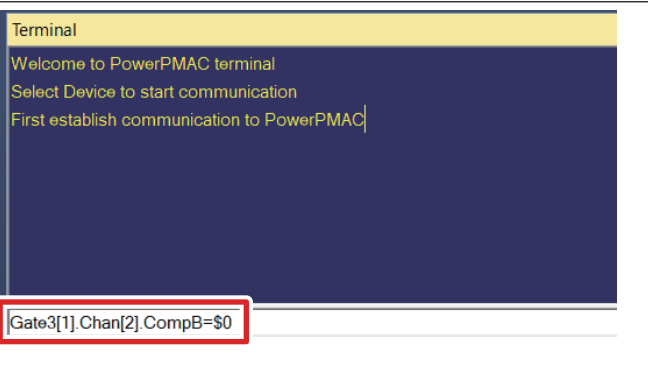
9	Solution Explorer の [C Language] - [Realtime Routines] - [usercode.c] をクリックし、右の関数を追記します。	<pre>static double X_old=0; static double Y_old=0; static double SumDistance=0; double CalcTableBasedCompare(struct MotorData*Mptr) { volatile GateArray3 *MySecondGate3IC; MySecondGate3IC=GetGate3MemPtr(1); double X_now; X_now=pshm->Motor[1].DesPos+pshm->Motor[3].DesPos; double Y_now; Y_now=pshm->Motor[2].DesPos+pshm->Motor[4].DesPos; double distance; distance=sqrt((X_old-X_now)*(X_old-X_now)+(Y_old-Y_now)*(Y_old-Y_now)); if(DistanceCountOn)SumDistance+=distance; X_old=X_now; Y_old=Y_now; MySecondGate3IC->Chan[0].CompB=SumDistance; return 0; }</pre>
10	Solution Explorer の [PMAC Script Language] - [Global Includes] を右クリックし、[Add] から [New Item...] を選択します。そのあとに、[Name] の欄に「TCRConfigurations.pmh」とタイプし、[Add] ボタンをクリックします。	
11	右のテキストを TCRConfigurations.pmh に書き込みます。 注釈付きの設定の内容については、「第 4 章 各種設定のカスタマイズ方法 (P.4-1)」を参照してください。	<pre>global DistanceCountOn Gate3[1]. Chan[1]. CompB=50. 0*1. 414//*1 Gate3[1]. Chan[1]. CompB=50. 0*1. 414+400//*2 //Gate3[1]. Chan[1]. CompB=**** Gate3[1]. Chan[2]. CompB=\$80000000//*3</pre>

12	Power PMAC IDE の画面右上の Solution Explorer のプロジェクト名を右クリックし、[Build and Download All Programs] を選択し、ビルド&ダウンロードを実行します。	
13	Output で異常がないことを確認します。	

3-7-4 レーザ発振器の設定

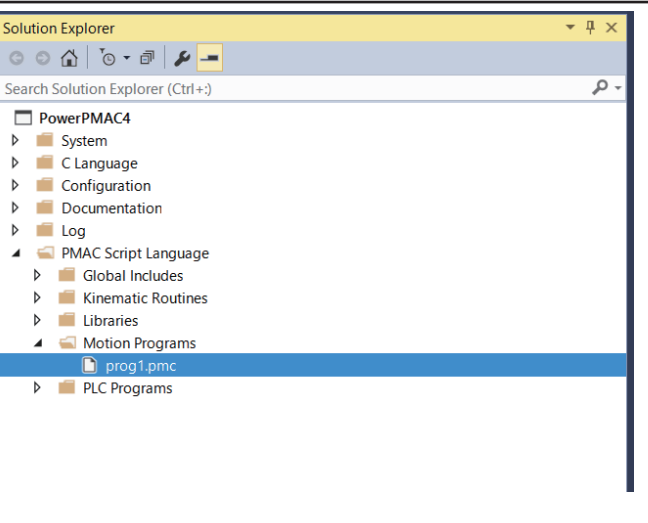
レーザ発振器のパワー設定やレーザ出力のための事前準備を行います。

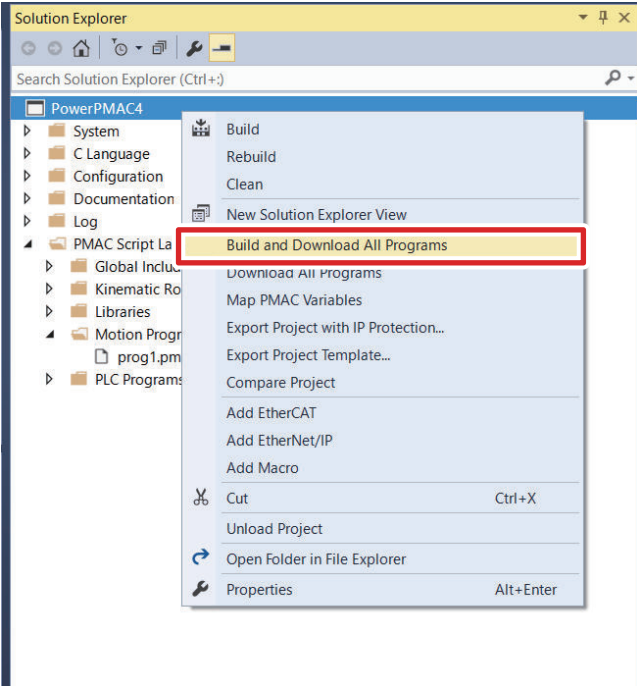
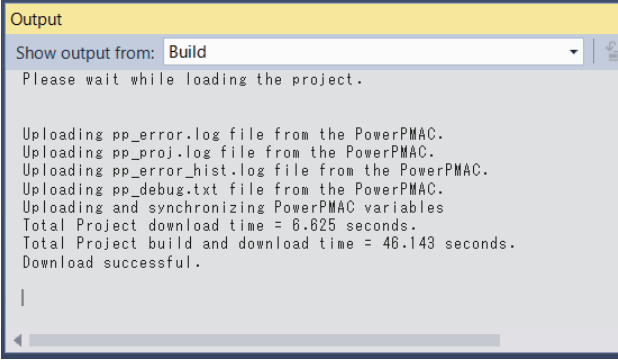
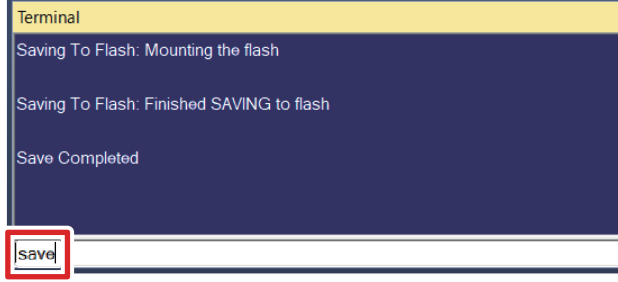
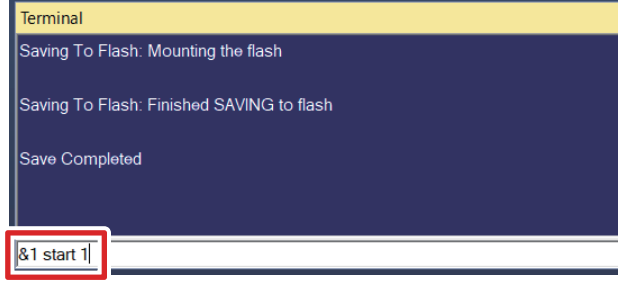
1	非常停止用押しボタンスイッチが OFF となっていることを確認します。	
2	Terminal で [LaserPower=\$1] コマンドをタイプします。	
3	Terminal で [LaserLatch=1] コマンドをタイプします。	

4	Terminal で [MasterOscillator=1] コマンドをタイプします。	
5	Terminal で [Gate3[1].Chan[2].CompB=\$C000000] コマンドをタイプします。	
6	レーザが出力されていることを確認します。	
7	Terminal で [Gate3[1].Chan[2].CompB=\$0] コマンドをタイプします。	

3-7-5 設定の確認

本紙の設定が正しく行われたかを、実際に加工が可能かどうかによって確認します。

1	Solution Explorer の [PMAC Script Language] - [Motion Programs] の下にある [prog1.pmc] を開きます。	
2	加工で使用するプログラムを prog1.pmc に書き込みます。	

3 Power PMAC IDE の画面右上の Solution Explorer のプロジェクト名を右クリックし、[Build and Download All Programs] を選択し、ビルド&ダウンロードを実行します。	
4 Output で異常がないことを確認します。 転送が失敗した場合は、Output でエラーの内容を確認してください。	
5 Power PMAC IDE の Terminal で [save] コマンドをタイプします。 終了すると、Terminal に “Save Completed” と表示されます。	
6 Power PMAC IDE の Terminal で [&1 start 1] コマンドをタイプします。	
7 正しく加工ができていることを確認します。	

4

各種設定のカスタマイズ方法

本章では、各種設定のカスタマイズ方法について記載します。

4

4-1	globaldefinitions.pmh.....	4-2
4-2	MotionOnTheFly.pmh.....	4-3
4-3	MotorControl.pmh	4-4
4-4	SensorControl.pmh.....	4-7
4-5	Galvano.pmh	4-8
4-6	LaserControl.pmh	4-10
4-7	TCRConfigurations.pmh	4-11

4-1 globaldefinitions.pmh

各種クロックやタスク周期の設定を行うファイルです。

No.	パラメータ	説明
*1	Coord[1].SegMoveTime	円弧補間などの軌跡の補間時間を ms 単位で指定します。本値が小さいほど、正確な軌跡に近づきますが、計算量増大により CPU の計算負荷が大きくなります。
*2	Gate3[0].PhaseFreq	フェーズクロックの周波数を指定します。本値が大きいほど、電流ループなどの処理を高精度に行えますが、計算量増大により CPU の計算負荷が大きくなります。
*3	Gate3[0].ServoClockDiv	サーボクロックの分周比を指定します。本値が小さいほど、位置ループなどの処理を高精度に行えますが、計算量増大により CPU の計算負荷が大きくなります。 サーボクロック周波数は本設定値により以下の値になります。 サーボクロック周波数 = フェーズクロック周波数 / (Gate3[0].ServoClockDiv + 1)
*4	Sys.PhaseOverServoPeriod	以下に示す値を設定します。 $\text{Sys.PhaseOverServoPeriod} = 1 / (\text{Gate3[0].ServoClockDiv} + 1)$
*5	Sys.ServoPeriod	以下に示す値を設定します。 $\text{Sys.ServoPeriod} = 1000 * (\text{Gate3[0].ServoClockDiv} + 1) / \text{Gate3[0].PhaseFreq}$
*6	Gate3[x].PhaseServoDir	フェーズクロックとサーボクロックをどのユニットから供給するかを指定します。

4-2 MotionOnTheFly.pmh

MOTF 機能に関する設定を行うファイルです。

No.	パラメータ	説明
*1	KdgainX	X 軸において MOTF に使用するフィルタの係数を指定します。本値が小さいほど、ガルバノスキャナの動作範囲が大きくなります。本値はガルバノスキャナが制御できる範囲を最大限使用できるぐらい小さい値に設定することで、より高速な応答をすることができます。
*2	KdgainY	上記の Y 軸のフィルタ係数を指定します。
*3	GalvoSfX	X 軸におけるレーザの移動距離（指令単位系）に対するガルバノモータ指令位置（ビット）を指定します。（単位: ビット／指令単位）
*4	GalvoSfY	上記の Y 軸の値を指定します。
*5	&1 #1-> ; #2-> ; #3-> ; #4-> ;	座標系 1 にインバースキネマティクス軸として軸 1、2、3、4 を割り付けています。軸番号を変更する場合、また他の軸を座標系に割り付けたい場合には本箇所を追記してください。

4-3 MotorControl.pmh

リニアモータ制御に関する設定を行うファイルです。

No.	パラメータ	説明
*1	Motor[x].ServoCtrl	サーボ制御を行う軸 x については 1 を設定します。本書では、6 軸を使用していますが、より多くの軸数を使用する場合は、Motor[y].ServoCtrl(y>6)についても 1 を設定してください。
*2	Motor[x].pEnc, Motor[x].pEnc2	軸 x のループフィードバックとして参照するエンコーダテーブルを指定します。本書では軸と同一インデックスのエンコーダテーブルを割り付けています。
*3	Motor[x].EncType	軸 x のエンコーダ種別を指定します。本書では正弦波エンコーダを対象としているため、6 を指定します。
*4	Motor[x].PosUnit	軸 x の指令単位を指定します。本書では mm 単位を使用するため、3 を設定しています。
*5	EncTable[x].Type	エンコーダテーブルの計算方式を指定します。本書ではフィードバック値を 32 ビット読み出しするため、1 を設定しています。
*6	EncTable[x].pEnc	エンコーダテーブル x が参照するレジスタを指定します。本書では形 CK3W-AX2323N ユニットで計算されたフィードバック値を取得しています。
*7	EncTable[x].ScaleFactor	形 CK3W-AX2323N ユニット (24 ビット) のデータをエンコーダテーブル (32 ビット) に取得する上での変換係数 $1/2^8$ を指定します。
*8	Motor[x].AmpFaultlevel	アンプフォールトをハイアクティブとするか、ローアクティブとするかを設定します。本書ではハイアクティブとします。
*9	Motor[x].FatalFeLimit	軸 x で運転を停止する偏差量の大きさを指定します。安全性を向上させるためには、より小さい値を設定してください。対して、偏差異常が発生し運転が開始されない場合は、より大きい値を設定してください。
*10	Motor[x].PhaseCtrl	電流ループを行う軸 x については 4 を設定します。本書では XY ステージで Direct PWM 制御方式を使用していますが、アナログ電圧制御や EtherCAT 制御などを使用する場合は、0 に設定してください。
*11	Motor[x].pPhaseEnc	モータのフィードバック位置を取得するレジスタを指定します。
*12	Motor[x].PhasePosSf	No.11 で取得した現在位置から電気角 (0~2048) への変換係数を指定します。具体的には以下の式に基づいて計算します。 $\text{PhasePosSf} = \frac{2048 \times A}{B}$ ここで、A は Gate3[0].Chan[x].PhaseCapt の 1 パルスあたりの長さを指定します。LIDA48 は正弦波 1 周期あたり 0.02mm、PMAC の内挿分割数が 16384 であるため、0.02/16384 を指定します。 また、B はモータの電気角が 1 回転する長さを指定します。本書で使用するリニアモータは ECL(Electrical Cycle Length)が 36mm であるため、36 を設定します。
*13	Motor[x].PhaseOffset	各相モータの電気角 (0~2048) の位相差を指定します。三相モータの場合は、683 または -683 となります。
*14	Motor[x].pAdc	サーボアンプからのフィードバック電流値を読み込むレジスタを指定します。本書では、Direct PWM インタフェースから読み取った値を使用しています。
*15	Motor[x].AdcMask	No.14 のレジスタのうち、どのビットをフィードバック値として使用するかを指定します。この値はサーボアンプに搭載された電流値を読み取る AD コンバータの仕様に応じて設定します。

No.	パラメータ	説明
*16	Motor[x].pDac	コントローラの指令出力を書き込むレジスタの先頭アドレスを指定します。今回は形 CK3W-AX2323N の U 相/V 相/W 相出力用のレジスタが出力先であるため、その先頭である U 相 (Gate3[0].Chan[0].Pwm[0])を指定します。
*17	Motor[x].PhaseFindingDac	フェーズリファレンスサーチ時に使用する出力の大きさを指定します。フェーズリファレンスサーチが完了しない場合は、出力が不足している可能性があるため、本値を大きくしてください。また、フェーズリファレンスサーチ中に I2TFault のステータス異常が発生する場合は、出力電流が定格値を超えているため、本値を小さくしてください。
*18	Motor[x].PhaseFindingTime	フェーズリファレンスサーチの継続時間の大きさを指定します。フェーズリファレンスサーチが完了しない場合は、本値を大きな値に設定してください。また、フェーズリファレンスサーチ中に I2TFault のステータス異常が発生する場合は、出力電流が定格値を超えているため、本値を小さくしてください。
*19	Motor[x].PwmSf	Direct PWM 出力の係数を指定します。フルレンジは 16384 です。PWM 波形のデューティサイクルが 100%とならないように、通常、フルレンジの 95%未満に設定します。 本パラメータは使用しているサーボアンプの仕様に応じて設定してください。
*20	Motor[x].MaxDac	モータおよびサーボアンプの連続定格電流値を指定します。以下の式に基づいて計算してください。 $\text{MaxDac} = \frac{\cos 30^\circ \times 32767 \times \text{連続定格電流}}{\text{フィードバック電流のフルレンジ電流}}$ 今回使用する機器は X 軸、Y 軸ともにサーボアンプの連続定格電流の方が小さいため、サーボアンプの連続定格電流を採用します。 形 CK3A-G310L: 20A rms INC-5716D0-0S: 9.88A rms INC-5723D0-0S: 14.82A rms また、フィードバック電流のフルレンジは形 CK3A-G310L の仕様に基づき 31.35Arms を採用します。
*21	Motor[x].I2tSet	モータおよびサーボアンプの短時間定格電流値を指定します。以下の式に基づいて計算してください。 $\text{I2tSet} = \frac{\cos 30^\circ \times 32767 \times \text{短時間定格電流}}{\text{フィードバック電流のフルレンジ電流}}$ 今回使用する機器は X 軸、Y 軸ともにサーボアンプの連続定格電流の方が小さいため、サーボアンプの短時間定格電流値を採用します。 形 CK3A-G310L: 10A rms INC-5716D0-0S: 3.08A rms INC-5723D0-0S: 4.62A rms
*22	Motor[x].I2tTrip	モータおよびサーボアンプの短時間定格電流値の許容時間を指定します。以下の式に基づいて計算してください。許容時間は形 CK3A-G310L の仕様から 2 秒とします。 $\text{I2tTrip} = (\text{MaxDac}^2 + \text{IdCmd}^2 - \text{I2tSet}^2) \times \text{許容時間 (秒)}$
*23	Gate3[0].Chan[x].PwmDeadTime	PWM 信号のデットタイムを 3 マイクロ秒に指定します。デットタイムは以下の式で計算します。 $\text{DeadTime} = 0.0533 \mu\text{s} \times \text{Gate3[0].Chan[x].PwmDeadTime}$ 本値はサーボアンプの仕様に応じて設定してください。

No.	パラメータ	説明
*24	Gate3[0].Chan[x].PwmFreqMult	PWM 出力周波数を 15kHz に設定します。PWM 周波数は以下の式で計算します。 $f_{\text{PWM}} = \frac{\text{PwmFreqMult}+1}{2} \times \text{フェーズ周波数}$ PWM 周波数は、40kHz 以下でかつサーボアンプの最高入力周波数に応じて設定してください。
*25	Gate3[0].Chan[x].PackOutData	U/V/W 相の出力レジスタへの書き出しを 2 回で完了し、高効率なアルゴリズムを実現するために圧縮フォーマットを有効に設定します。 形 CK3W-AX2323N を使用している場合はこの値は変更しないでください。
*26	Gate3[0].AdcAmpStrobe	ADC Strobe Word (Direct PWM 規格においてコントローラからサーボアンプに送信する制御指令)を指定します。本値はサーボアンプの仕様に応じて変更してください。
*27	Gate3[0].AdcAmpHeaderBits	フィードバック電流値データに含まれるヘッダのビット数を指定します。本値はサーボアンプの仕様に応じて変更してください。
*28	Gate3[0].AdcAmpClockDiv	サーボアンプで電流データを読み込む AD コンバータのクロック信号周波数を指定します。
*29	Gate3[0].Chan[x].PackInData	U/V/W 相の入力レジスタからの読み出しを 2 回で完了し、高効率なアルゴリズムを実現するために圧縮フォーマットを有効に設定します。 形 CK3W-AX2323N を使用している場合はこの値は変更しないでください。
*30	Gate3[0].Chan[x].AtanEna	正弦波エンコーダを利用する場合に有効とします。パルスエンコーダやシリアルエンコーダを使用する場合は 0 を設定してください。
*31	Gate3[0].Chan[x].EncCtrl	エンコーダのデコード方式を設定します。本書では、4 通倍 AB 相 CW デコードを設定します。
*32	Motor[x].PosSf, Motor[x].Pos2Sf	軸インタフェースユニットで計算したフィードバック値と指令位置との変換係数を指定します。 正弦波エンコーダ使用時は 1 周期あたり 16834 パルスカウントとなること、また使用している正弦波エンコーダが 1 周期あたり 0.020mm であるため、0.020/16384 とします。

4-4 SensorControl.pmh

オーバトラベルリミットスイッチに関する設定を行うファイルです。

No.	パラメータ	説明
*1	Motor[x].pLimits	オーバトラベルリミットスイッチの参照先レジスタを指定します。
*2	Motor[x].LimitBits	オーバトラベルリミットスイッチの参照先レジスタ(No.1)のうち、どのビットを参照するかを指定します。

4-5 Galvano.pmh

ガルバノスキャナの制御に関する設定を行うファイルです。

No.	パラメータ	説明
*1	Gate3[1].SerialEncCtrl	下図に記載のとおりに設定します。
*2	Motor[x].ServoCtrl	サーボ制御を行う軸 x については 1 を設定します。本書では、6 軸を使用していますが、より多くの軸数を使用する場合は、Motor[y].ServoCtrl(y>6)についても 1 を設定してください。
*3	Motor[x].Ctrl	ガルバノ制御では位置制御を行うため、Sys.PosCtrl を設定します。
*4	Motor[x].MaxSpeed	ガルバノ制御の最大速度を指定します。今回は最大速度監視を無効とするため 0 を指定します。
*5	Motor[x].MaxPos	ガルバノ制御の指令位置の最大値を指定します。今回は無効とするため、0 を指定します。
*6	Motor[x].MinPos	ガルバノ制御の指令位置の最小値を指定します。今回は無効とするため、0 を指定します。
*7	Motor[x].pDac	コントローラの指令出力を書き込むレジスタの先頭アドレスを指定します。今回は形 CK3W-GC2200 の XY 出力用のレジスタが出力先であるため、Gate3[1].Chan[x].Dac[0]を指定します。
*8	Motor[x].pLimits	ガルバノ制御のオーバトラベルリミットを設定します。今回は無効とするため、0 を指定します。
*9	Motor[x].FatalFeLimit	軸 x で運転を停止する偏差量の大きさを指定します。ガルバノスキャナはこの設定を無効にするために 0 を指定します。
*10	Motor[x].pEnc	軸 x の位置ループフィードバックとして参照するエンコーダテーブルを指定します。本書では軸と同一インデックスのエンコーダテーブルを割り付けています。
*11	Motor[x].pEnc2	軸 x の速度ループフィードバックとして参照するエンコーダテーブルを指定します。本書では軸と同一インデックスのエンコーダテーブルを割り付けています。
*12	EncTable[x].Type	エンコーダテーブルの計算方式を指定します。本書ではフィードバック値を 32 ビット読み出しするため、1 を設定しています。
*13	EncTable[x].pEnc	エンコーダテーブル x が参照するレジスタを指定します。本書では形 CK3W-GC2200 ユニットで計算した線形補間値を出力します。
*14	EncTable[x].ScaleFactor	SL2-100 データ (20 ビット) をエンコーダテーブル (32 ビット) に取得するうえでの変換係数 $1/2^{12}$ を指定します。

Reserved												Reserved												Tx_Valid		TriggerEdge												Control_Bits		Sync		ClockSel		Reserved		TxEnable		Reserved											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																												
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0																												
0				0				0				8				2				8				0				0				—																											
—												Asynchronous Mode		Command Position																																													
												Phase Clock																																															
												Enable TX of SL2-100																																															

Command Positionの場合

4-6 LaserControl.pmh

レーザ発振器の制御に関する設定を行うファイルです。

No.	パラメータ	説明
*1	Gate3[1].Chan[0].CompA	下記のとおり設定します。
*2	Gate3[1].Chan[1].CompA	SL2-100 通信データの PMAC 内部クロックに対するディレイ時間を設定します。本書では 0 を設定しています。
*3	Gate3[1].Chan[2].CompA	下記のとおり設定します。

Reserved																PWMPeriod																DutyCycle															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1																
0				0				0				D				0				0				0				8																			
—																PWM Frequency=30KHz																Duty Ratio=50%															

Gate3[1].Chan[0].CompAの設定

Reserved																PulseCount																Reserved							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0								
0				0				0				F				F				F				0				0											
												Continuous Pulse Outputs																											

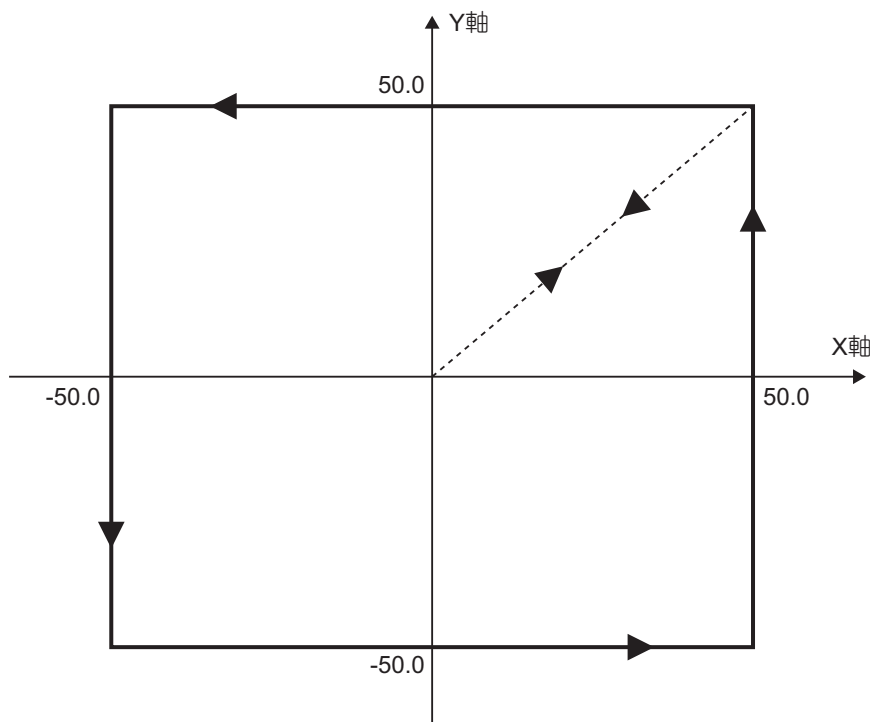
Gate3[1].Chan[2].CompAの設定

4-7 TCRConfigurations.pmh

レーザの ON/OFF に関する条件を設定するファイルです。

No.	パラメータ	説明
*1	Gate3[1].Chan[1].CompB	レーザを OFF から ON にするまでの走行距離を設定します。
*2	Gate3[1].Chan[1].CompB	レーザを ON から OFF にするまでの走行距離を設定します。 以後、レーザ出力がトグルする条件を追記します。
*3	Gate3[1].Chan[2].CompB	下記のとおりに設定します。

本書では、以下のように原点から正方形上にレーザ出力して、再び原点に戻ってくるケースを条件として記載しています。



Reserved																																CompOutPol		CompOutWrite		Reserved		CompCkSel		ClearTable		CompareEnable	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
8				0				0				0				0				0				0																			
—																																											
—																																											
H active																																											
Normal																																											
—																																											
Servo Clock																																											
Normal																																											
Compare Enabled																																											

Gate3[1].Chan[2].CompBの設定

オムロン株式会社 インダストリアルオートメーションビジネスカンパニー

製品に関するお問い合わせ先



0120-919-066

携帯電話・IP電話などではご利用いただけませんので、右記の電話番号へおかけください。

055-982-5015
(通話料がかかります)

受付時間：9:00～19:00 (12/31～1/3を除く)

クイック オムロン



オムロンFAクイックチャット

www.fa.omron.co.jp/contact/tech/chat/

技術相談員にチャットでお問い合わせいただけます。(I-Webメンバーズ限定)

受付時間：平日9:00～12:00 / 13:00～17:00 (土日祝日・年末年始・当社休業日を除く)

※受付時間、営業日は変更の可能性がございます。最新情報はリンク先をご確認ください。



その他のお問い合わせ：納期・価格・サンプル・仕様書は貴社のお取引先、または貴社担当オムロン販売員にご相談ください。オムロン制御機器販売店やオムロン販売拠点は、Webページでご案内しています。



オムロン制御機器の最新情報をご覧ください。緊急時のご購入にもご利用ください。 www.fa.omron.co.jp

本誌には主に機種のご選定に必要な内容を掲載しており、ご使用上の注意事項等を掲載していない製品も含まれています。

本誌に注意事項等の掲載のない製品につきましては、ユーザーズマニュアル掲載のご使用上の注意事項等、ご使用の際に必要な内容を必ずお読みください。

- 本誌に記載の標準価格はあくまで参考であり、確定されたユーザ購入価格を表示したものではありません。本誌に記載の標準価格には消費税が含まれておりません。
- 本誌にオープン価格の記載がある商品については、標準価格を決めていません。
- 本誌に記載されているアプリケーション事例は参考用ですので、ご採用に際しては機器・装置の機能や安全性をご確認の上、ご使用ください。
- 本誌に記載のない条件や環境での使用、および原子力制御・鉄道・航空・車両・燃焼装置・医療機器・娯楽機械・安全機器、その他人命や財産に大きな影響が予測されるなど、特に安全性が要求される用途に使用される際には、当社の意図した特別な商品用途の場合や特別の合意がある場合を除き、当社は当社商品に対して一切保証をいたしません。
- 本製品の内、外国為替及び外国貿易法に定める輸出許可、承認対象貨物（又は技術）に該当するものを輸出（又は非居住者に提供）する場合は同法に基づく輸出許可、承認（又は役務取引許可）が必要です。
- 規格認証/適合対象機種などの最新情報につきましては、当社Webサイト (www.fa.omron.co.jp) の「規格認証/適合」をご覧ください。

オムロン商品のご用命は

カタログ番号 SBCE-504B

2023年1月現在

©OMRON Corporation 2021 - 2023 All Rights Reserved.

お断りなく仕様などを変更することがありますのでご了承ください