Programmable Multi-Axis Controller

Application Guide
Laser Application

CK3M-CPU1□1
CK3W-AX2323□
CK3W-GC2200

Application
Starting
Guide

NOTE

1. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.
2. No patent liability is assumed with respect to the use of the information contained herein.
   Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice.
3. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions.
   Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

Trademarks

- Microsoft, Windows, Excel, and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.
- EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- ODVA, CIP, CompoNet, DeviceNet, and EtherNet/IP are trademarks of ODVA.

Other company names and product names in this document are the trademarks or registered trademarks of their respective companies.

Copyrights

- Microsoft product screen shots used with permission from Microsoft.
- This product incorporates certain third party software. The license and copyright information associated with this software is available at http://www.fa.omron.co.jp/nj_info_e/.

Sections in this Manual

CONTENTS

Section 1 Introduction

Section 2 Device Configuration

Section 3 Connection Procedures

Section 4 How to Customize Various Settings

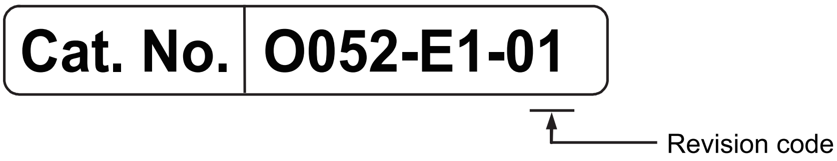
Related Manuals

To safely use the system, obtain manuals or user's guides for devices and equipment that make up the system, and confirm and understand the precautions related to safety such as "Safety Precautions" and "Precautions for Safe Use", and other contents of the manuals or user's guides, including "Precautions for Correct Use", before use.
The manuals provided by OMRON Corporation (hereinafter, "OMRON") and Delta Tau Data Systems Inc. (hereinafter "DT") are as shown below.

Manufacturer	Cat. No.	Model	Manual name
OMRON	O036	CK3M-CPU□□□ CK3W-AX2323□ CK3W-GC2200	Programmable Multi-Axis Controller Hardware User's Manual
DT	O014	---	Power PMAC User's Manual
DT	O015	---	Power PMAC Software Reference Manual
DT	O016	---	Power PMAC IDE User's Manual (V4)

Revision History

A manual revision code appears as a suffix to the catalog number on the front and back covers of the manual.



Revision code	Date	Revised content
01	May 2021	Original production

Terms and Definitions

Term	Description and definition
PMAC	An acronym for Programmable Multi-Axis Controller.
Power PMAC IDE	Computer software that is used to configure the Controller, create user programs, and perform monitoring.
AC servo motor	A servo motor that is driven by AC signal input. A rotating magnetic field is formed in the motor by the input of an AC signal in order to rotate the rotor of the motor.
Servo amplifier	A control device that controls a servo motor. Upon receipt of a command from the Controller, it switches the power by means of a power device in order to output a power signal to drive the motor.
Direct PWM	OMRON’s unique communications method for communications between the Power PMAC and the servo amplifier. The Power PMAC directly sends an ON/OFF signal from the power device of the servo amplifier, while the servo amplifier sends a current feedback value to the Power PMAC. This allows the Power PMAC to form a current loop.
Electrical angle	The angle of the rotating magnetic field used to rotate an AC servo motor.
Commutation control	Control for creating a rotating magnetic field with the required electrical angle in order to rotate the motor.
Phase reference	The home of the electrical angle. To perform commutation control, it is necessary to establish a phase reference at first (i.e., phase reference search).
Sinusoidal Encoder	A type of encoder that outputs SIN/COS waveforms at 1 Vpp.
Galvo Scanner	A control device for controlling the angle of rotation of the mirror that reflects laser light for scanning the position where the laser light is irradiated. It utilizes the high response performance of the mirror to ensure high-speed trajectory control.
SL2-100	A type of protocol for communications with digital Galvo Scanners.
Control command	A command that sets the type of data to be sent from the Galvo Scanner to the Controller.
XY stage	A positioning stage that moves in the X-axis and Y-axis directions. Although the response speed is slower than that of a Galvo Scanner, it can scan a larger area due to its wide range of movement.
MOTF	An acronym for Motion On The Fly. It is an algorithm that divides a position command into high frequency components and low frequency components, and distributes them to actuators with different response speeds. In this manual, the high frequency components are commanded to the Galvo Scanner and the low frequency components are commanded to the XY stage.
Pulsed laser	A laser that blinks repeatedly at a specific time interval. In contrast, a device that constantly outputs laser light is called a continuous oscillation laser.
Q-switching	An oscillation method for pulsed lasers. It uses external oscillation pulse input to obtain pulsed laser output.
TCR	An acronym for Trigger Output by Commanded distance for Rapid Processing. It provides the function to trigger a laser output according to the distance set in the CK3W-GC□2□□.

Precautions

- For actual system construction, check the specifications for devices and equipment that make up the system, use a method with sufficient margin for ratings and performance, and adopt safety circuits and other safety measures to minimize risks even if a breakdown occurs.
- To safely use the system, obtain manuals or user’s guides for devices and equipment that make up the system, and confirm and understand the precautions related to safety such as “Safety Precautions” and “Precautions for Safe Use”, and other contents of the manuals or user’s guides, including “Precautions for Correct Use”, before use.
- The customer themselves must check all regulations, laws, and rules that are applicable to the system.
- No part or the whole of this document may be copied, duplicated, or redistributed without the permission of OMRON Corporation.
- The contents of this document are current as of May 2021.
  The contents of this document may be subject to change without notice for the purpose of improvement.

Special information in this document is classified as follows:

Precautions for Correct Use

Precautions on what to do and what not to do to ensure correct operation and performance.

Additional Information

Additional information to read as required.
This information is provided to increase understanding and make operation easier.

Symbols

❗	The filled circle symbol indicates operations that you must do. The specific operation is shown in the circle and explained in text. This example shows a general precaution for something that you must do.

1

Introduction

This section provides an introduction of this document.

1-1 Introduction

This document describes the procedures for building a laser processing system with a OMRON Programmable Multi-Axis Controller CK3M-□□□□ (hereinafter called “Controller”).

⚠ CAUTION

The scope of this document is to confirm the connection of devices listed as the connection targets. To configure a system using instructions that are not described in this document, obtain manuals or user’s guides for devices and equipment that make up the system, and confirm their contents including the precautions related to safety such as “Safety Precautions” and “Precautions for Safe Use” before use. ❗

2

Device Configuration

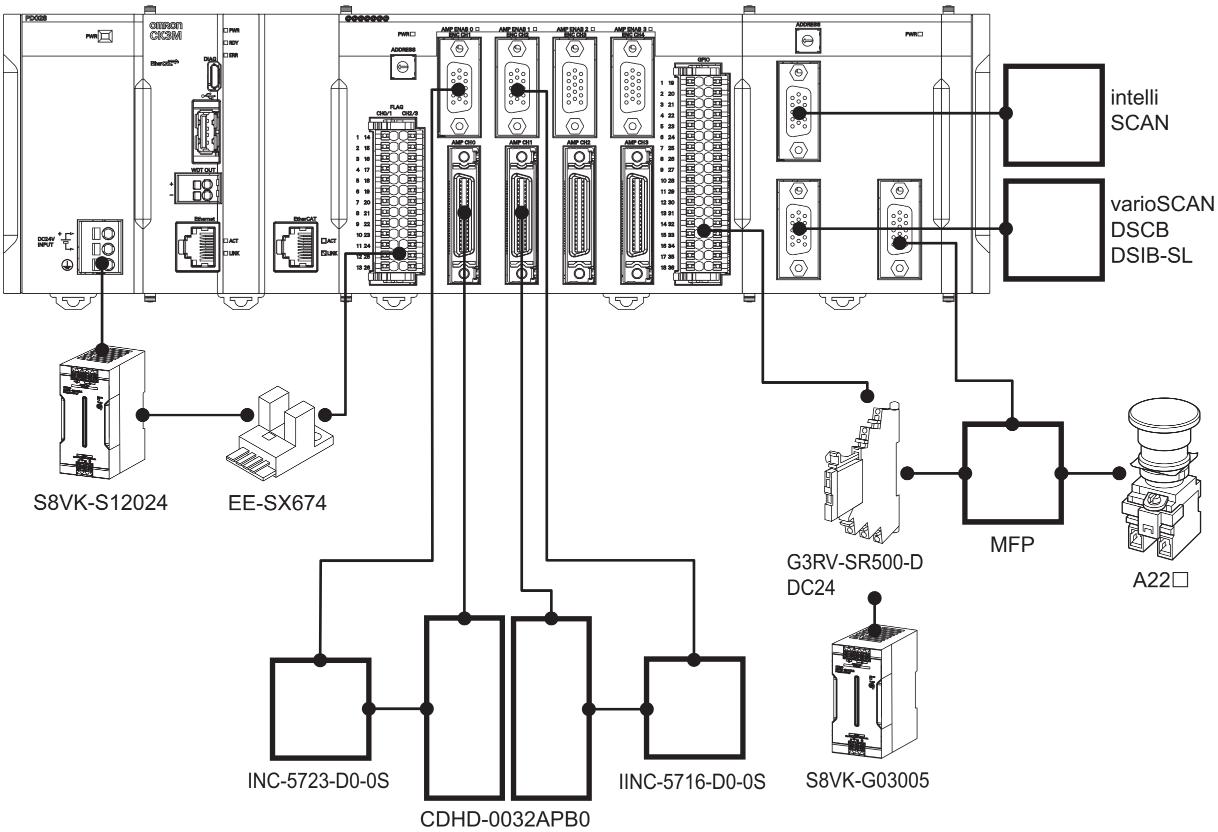
This section describes the configuration of devices.

2-1 Device Configuration

The configuration devices for reproducing the connection procedures in this document are shown below.

Manufacturer	Name	Model	Version
OMRON	Programmable Multi-Axis Controller CPU Unit	CK3M-CPU1□1	Ver. 2.6.1 or later
OMRON	Programmable Multi-Axis Controller Axis Interface Unit	CK3W-AX2323N[*1]	---
OMRON	Programmable Multi-Axis Controller Galvano Interface Unit	CK3W-GC2200	---
OMRON	Programmable Multi-Axis Controller Power Supply Unit	CK3W-PD048	---
OMRON	Programmable Multi-Axis Controller End Cover	CK3W-TER11	---
OMRON	Programmable Multi-Axis Controller Direct PWM Cable	CK3W-CAAD036A	---
OMRON	Programmable Multi-Axis Controller Serial Encoder Cable	CK3W-CAES03A	---
OMRON	Programmable Multi-Axis Controller Laser Interface Unit Cable	CK3W-CAG03A	---
OMRON	Switch Mode Power Supply (24 VDC)	S8VK-S12024	---
OMRON	Switch Mode Power Supply (5 VDC)	S8VK-G03005	---
OMRON	Switch Mode Power Supply (15 VDC)	S8FS-G10015CD	---
OMRON	Solid State Relay	G3RV-SR500-D DC24	---
OMRON	Photomicro Sensor	EE-SX674	---
Servotronix	Direct PWM Amplifier	CDHD-0032APB0[*1]	---
Maxphotonics	Laser Oscillator	MFP	---
INNO6	Linear Motor (Y Axis)	INC-5716-D0-0S	---
INNO6	Linear Motor (X Axis)	INC-5723-D0-0S	---
HEIDENHEIN	Linear Encoder	LIDA48[*1]	---
SCANLAB	Galvo Scanner	intelliSCAN	---
SCANLAB	fθ Lens	---	---
SCANLAB	Dynamic Focusing Unit	varioSCAN	---
---	Windows PC	---	---
DT	Power PMAC Setting Tool	Power PMAC IDE	Ver. 4.5 or later

[*1] This document assumes that the XY stage is controlled by the Direct PWM or Sinusoidal Encoder method. To use other control methods, change the settings in *3-4 Setup of the Motors and Encoder* on page 3-21 according to the corresponding Startup Guide.


intelli
SCAN
varioSCAN
DSCB
DSIB-SL
S8VK-S12024
EE-SX674
G3RV-SR500-D
DC24
MFP
A22□
INC-5723-D0-0S
IINC-5716-D0-0S
S8VK-G03005
CDHD-0032APB0

3

Connection Procedures

This section describes the procedures to connect the Controller and the devices that make up the laser application. The description assumes that the Controller is set to factory default.

3-1 Work Flow

The procedures for building a laser application are shown below.

3-2 Controller Setting Preparations **on page 3-4**	Perform the Controller setting preparations.
▼	
3-2-1 Creation of a New Project on page 3-4	
▼	
3-2-2 Controller Initial Setting on page 3-5	
▽	
3-3 Setup of the MOTF **on page 3-7**	Set up the MOTF.
▼	
3-3-1 Programming on page 3-7	
▼	
3-3-2 Determination of Filter's Coefficients on page 3-13	
▽	
3-4 Setup of the Motors and Encoder **on page 3-21**	Set up the motors and encoder.
▼	
3-4-1 Wiring of Devices on page 3-21	
▼	
3-4-2 Programming on page 3-22	
▼	
3-4-3 Tuning of the Current Loop on page 3-26	
▼	
3-4-4 Establishment of the Phase Reference on page 3-28	
▼	
3-4-5 Open-loop Testing on page 3-29	
▼	
3-4-6 Position Loop Auto-tuning on page 3-30	
▼	
3-4-7 Position Loop Interactive Tuning on page 3-31	
▼	
3-4-8 Tuning of the Y Axis on page 3-35	
▼	
3-4-9 Confirmation of the Tuning Results on page 3-36	
▽	
3-5 Setup of Overtravel Limit Switches and Home Positions **on page 3-39**	Set up various sensors.
▼	
3-5-1 Wiring of Devices on page 3-39	
▼	
3-5-2 Programming on page 3-39	
▼	
3-5-3 Confirmation of the Settings (Overtravel Limit Switches) on page 3-41	
▼	

3-5-4 Setup of Home Positions on page 3-42	
▽	
***3-6 Setup of the Galvo Scanner* on page 3-44**	Set up the Galvo Scanner.
▼	
3-6-1 Wiring of Devices on page 3-44	
▼	
3-6-2 Programming on page 3-45	
▼	
3-6-3 Setup of Control Commands on page 3-47	
▼	
3-6-4 Confirmation of the Settings on page 3-49	
▼	
3-6-5 Verification of the MOTF on the Actual Machine on page 3-50	
▽	
***3-7 Setup of the Laser Oscillator* on page 3-51**	Set up the laser oscillator.
▼	
3-7-1 Wiring of Devices on page 3-51	
▼	
3-7-2 Guide Laser Output on page 3-51	
▼	
3-7-3 Programming on page 3-53	
▼	
3-7-4 Setup of the Laser Oscillator on page 3-56	
▼	
3-7-5 Confirmation of the Settings on page 3-58	

3-2 Controller Setting Preparations

Perform the Controller setting preparations.
Install the Power PMAC IDE on the PC beforehand.

3-2-1 Creation of a New Project

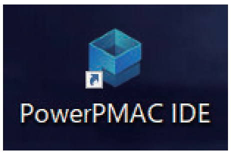
Follow the procedure below to create a new project.

1 Connect the Controller and computer with an Ethernet cable.

2 Turn ON the power supply to the Controller.

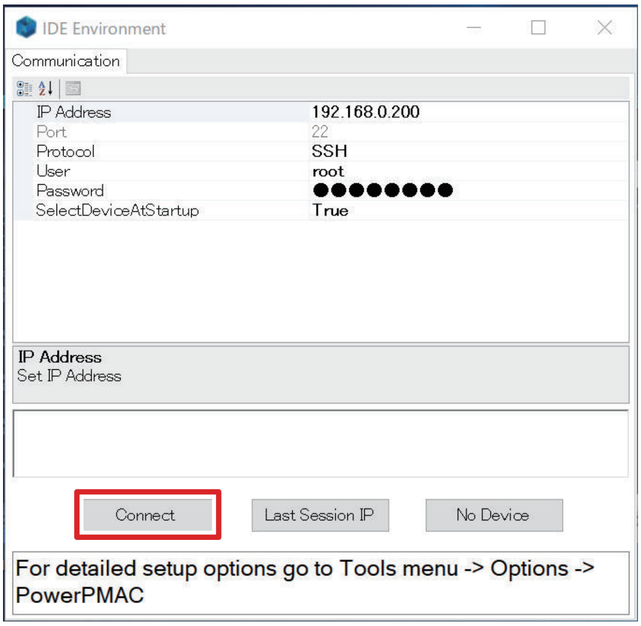
3 Start up Power PMAC IDE.

- If a dialog for checking access rights is displayed at the time of startup, select the option for starting up.



4 The Communication screen is displayed, so specify the IP address of the Controller to be connected to, and click the **Connect** button.

- The default IP address for the Controller is "192.168.0.200".
- If necessary, change the Windows IP address to "192.168.0.X".

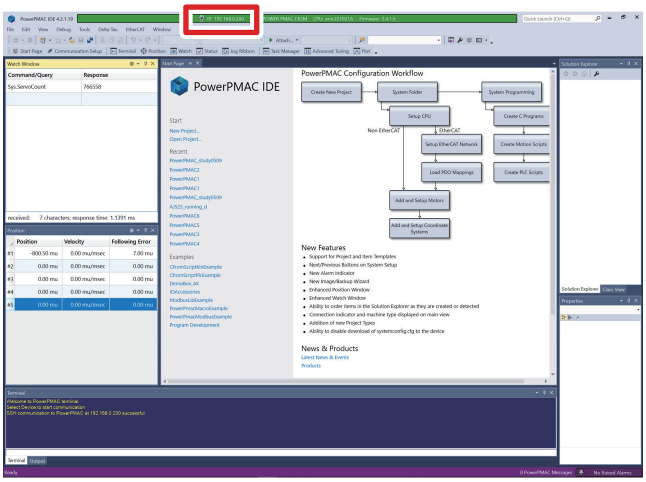


IP Address	192.168.0.200
Port	22
Protocol	SSH
User	root
Password	●●●●●●●●
SelectDeviceAtStartup	True

IP Address
Set IP Address

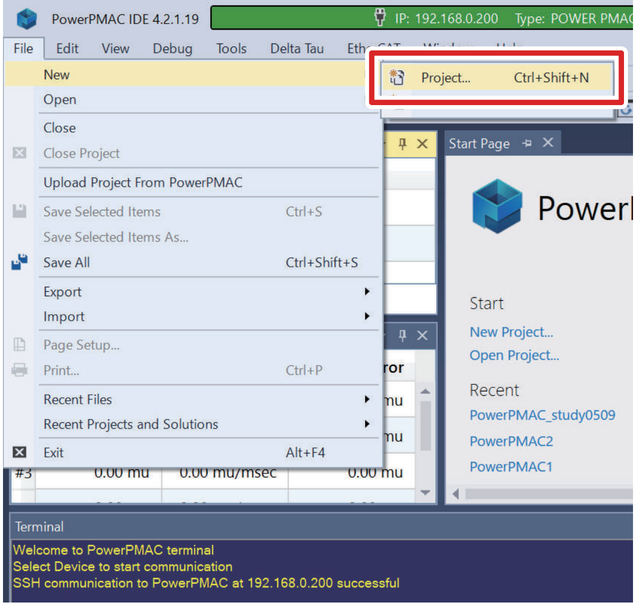
For detailed setup options go to Tools menu -> Options -> PowerPMAC

5 Power PMAC IDE starts up, and the Controller will come online.



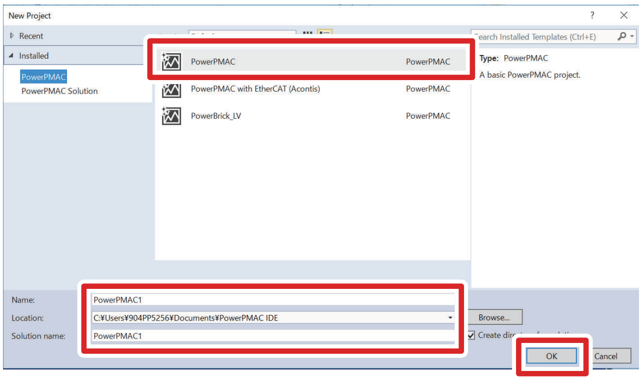
```
Welcome to PowerPMAC terminal
Select Device to start communication
SSH communication to PowerPMAC at 192.168.0.200 successful
```

6 From the **File** menu, select **New** – **Project**.



```
Welcome to PowerPMAC terminal
Select Device to start communication
SSH communication to PowerPMAC at 192.168.0.200 successful
```

7 As shown in the right figure, after you select **Power PMAC**, input a project name and save destination, and select the **OK** button.



Name: PowerPMAC1

Location: C:¥Users¥904PP5256¥Documents¥PowerPMAC IDE

Solution name: PowerPMAC1

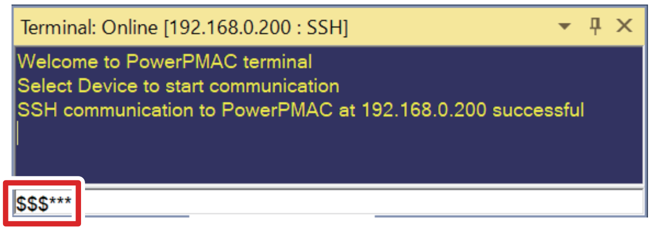
3-2-2 Controller Initial Setting

Follow the procedure below to perform the initial settings for the Controller.

Precautions for Correct Use

Since all memory is cleared by the initial settings, be sure to save any data remaining in the Controller that you may need.

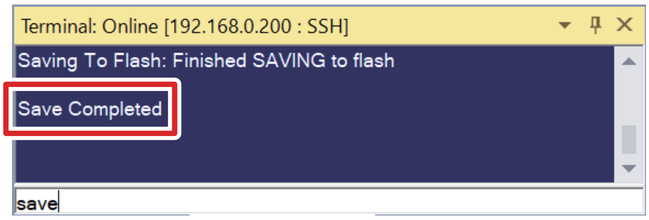
1 Type the **$$$***** command from the Terminal, and set the Controller to the factory default state.




```
Welcome to PowerPMAC terminal
Select Device to start communication
SSH communication to PowerPMAC at 192.168.0.200 successful

$$$***
```

2 Type the **save** command in the Power PMAC IDE Terminal.

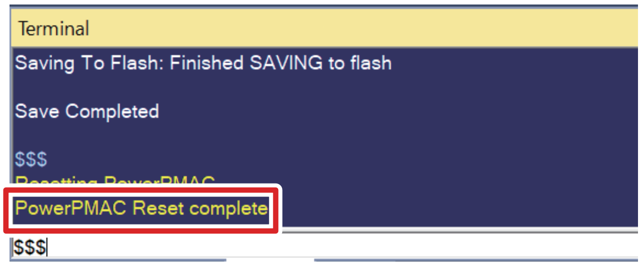
- When the save is completed, "Save Completed" is displayed in the Terminal.



```
Terminal: Online [192.168.0.200 : SSH]
Saving To Flash: Finished SAVING to flash

Save Completed

save
```

3 Type the **$$$** command in the Power PMAC IDE Terminal.

- When the reset is completed, "PowerPMAC Reset complete" is displayed in the Terminal.



```
Terminal
Saving To Flash: Finished SAVING to flash

Save Completed

$$$
Resetting PowerPMAC
PowerPMAC Reset complete
$$$
```

3-3 Setup of the MOTF

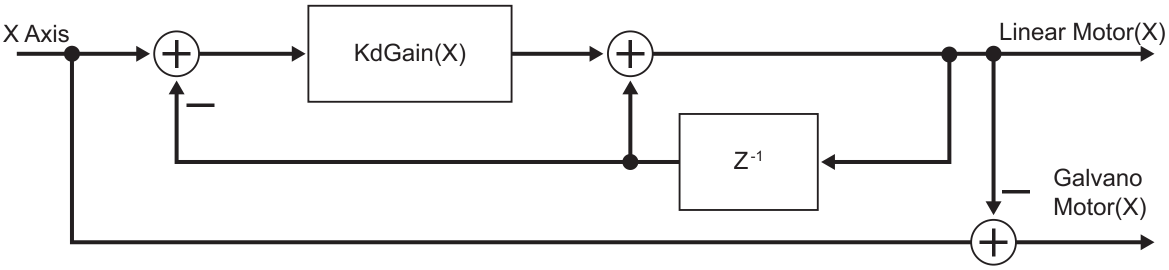
3-3-1 Programming

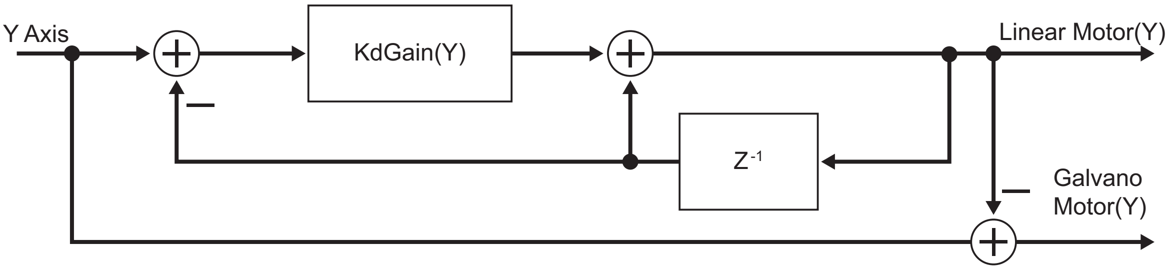
Set up the MOTF. This document assumes that you use a low-pass filter with EMA (Exponential Moving Average) to sort a positioning command into low frequency components and high frequency components.

The input/output equation for the EMA is shown below. Here, u(n) is the input signal at time n and y(n) is the output signal.

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

The following is a block diagram of the MOTF. The low frequency components of a command to an axis that passed through the low-pass filter are input to the motor, whereas the remaining travel distance is input to the Galvo Scanner. This enables high stroke and high responsivity to be compatible.



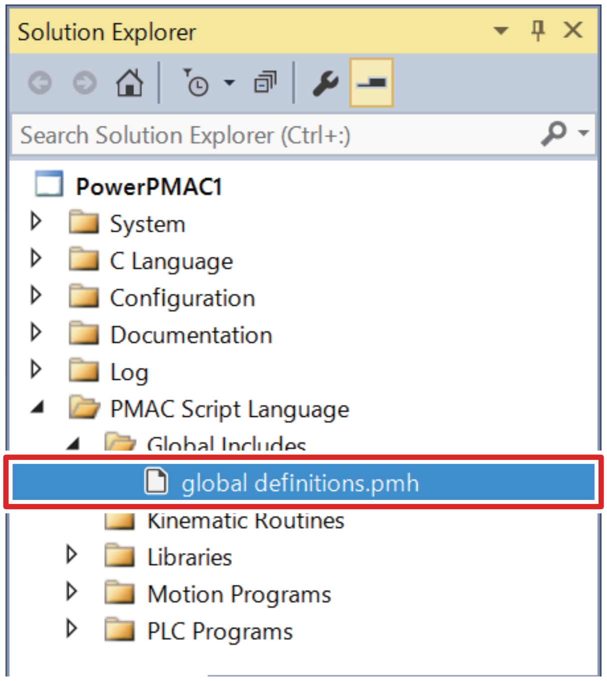


In the above diagram, KdGain is set to 0 or greater, but less than 1. To increase the cutoff frequency of the low-pass filter, set KdGain to a larger value.

The table below shows the relationship between axes on devices and motor numbers in this document.

Axes on devices		Motor number
Machine	Direction	
XY stage	X axis	#1
	Y axis	#2
Galvo Scanner	X axis	#3
	Y axis	#4
Focus unit	Z axis	#5
Virtual axis for TCR calculation		#6

1 Open the **global definitions.pmh** file under **PMAC Script Language** – **Global Includes** in the Solution Explorer.

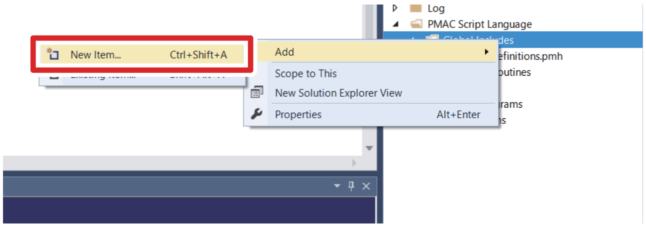


2 Write the text shown on the right to the global definitions.pmh file.

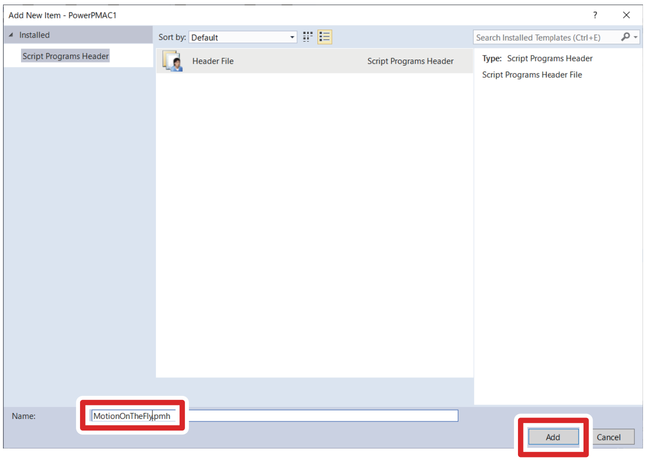
- For details on the annotated settings, refer to *Section 4 How to Customize Various Settings* on page 4-1.

```
Sys.WpKey=$AAAAAAAA
//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 In the Solution Explorer, right-click **PMAC Script Language** and select **Add** – **New Item...**.



4 In the **Name** box, type *MotionOnTheFly.pmh* and click the **Add** button.


Add New Item - PowerPMAC1
?
Installed
Script Programs Header
Sort by: Default
Search Installed Templates (Ctrl+E)
Header File
Script Programs Header
Type: Script Programs Header
Script Programs Header File
Add
Cancel

Name: MotionOnTheFly.pmh

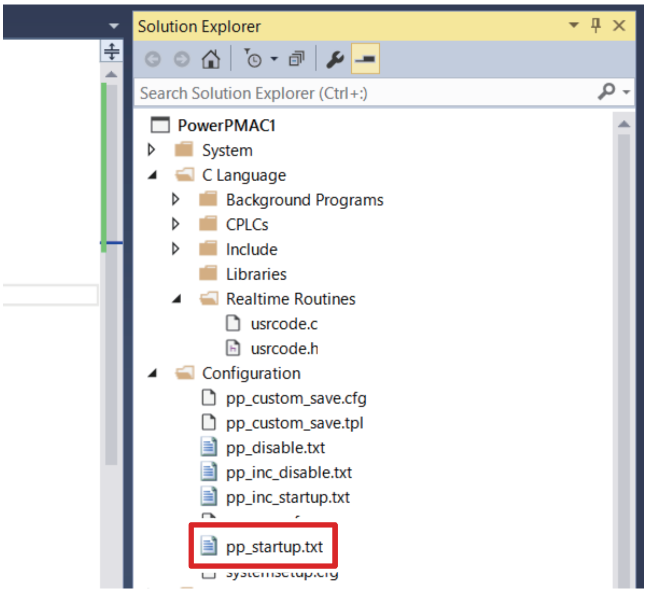
5 Open the **MotionOnTheFly.pmh** file under **PMAC Script Language** – **Global Includes** in the Solution Explorer.

6 Write the text shown on the right to the MotionOnTheFly.pmh file.

- For details on the annotated settings, refer to *Section 4 How to Customize Various Settings* on page 4-1.

```
//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->I;//*5
#2->I;//*5
#3->I;//*5
#4->I;//*5
```

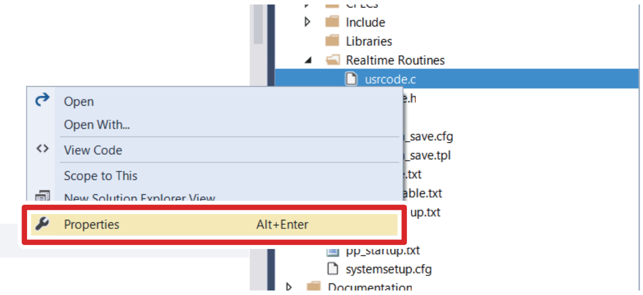
7 Open the **pp_startup.txt** file under **Configuration** in the Solution Explorer.



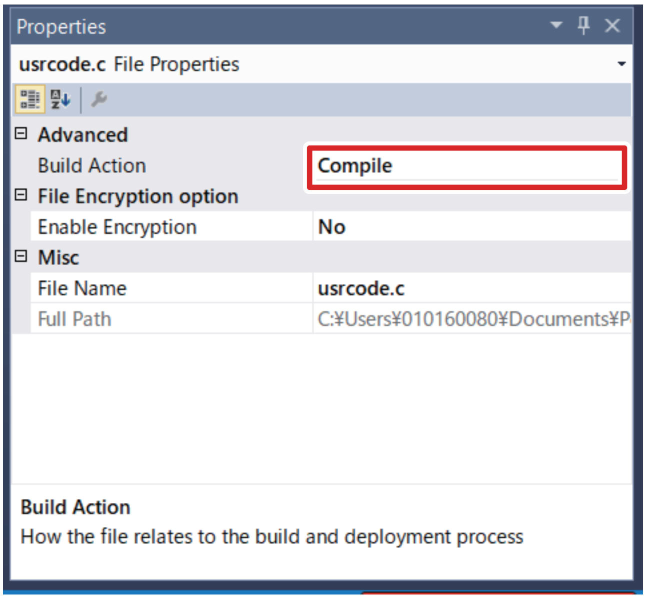
8 Write the text shown on the right to the pp_startup.txt file.

```
//Enable CfromScript
UserAlgo.CFunc=1;
```

9 In the Solution Explorer, right-click the **usercode.c** file under **C Language** – **Realtime Routines** and select **Properties**.



10 In **Build Action** in the **Properties** window, select **Compile**.



usrcode.c File Properties

Advanced

Build Action: Compile

File Encryption option

Enable Encryption: No

Misc

File Name: usrcode.c

Full Path: C:¥Users¥010160080¥Documents¥P

Build Action

How the file relates to the build and deployment process

11 Open the **usercode.c** file under **C Language** – **Realtime Routines**.

12 Delete the CfromScript function definitions shown on the right.

```
double CfromScript(double cfrom_type,double arg2,
double arg3,double arg4,double arg5,double arg6,d
ouble arg7,struct LocalData *Ldata)
{
    int icfrom_type=(int)cfrom_type;
    double*C,*D,*L,*R,rtn;//C,D,R-only needed if
doing Kinematics

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

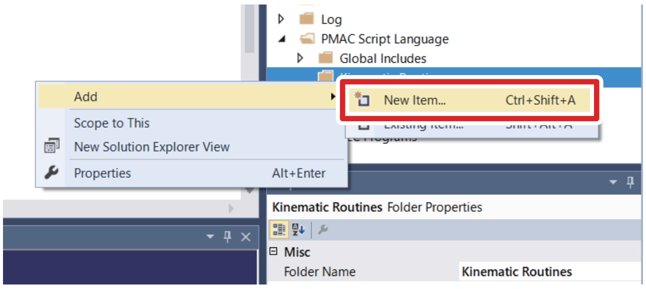
13 Write the text shown on the right to the usercode.c file.

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

double CfromScript(double kinTypeDouble,double ar
g2,double arg3,double arg4,double arg5,double arg
6,double arg7,LocalData *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/Galv
oSfX);
   _KinPosAxisY=_KinPosMotor2+(_KinPosMotor4/Galv
oSfY);
   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)*Gal
voSfX;
   _KinPosMotor4=(_KinPosAxisY-_KinPosMotor2)*Gal
voSfY;
   prevLpfXPos=_KinPosMotor1;
   prevLpfYPos=_KinPosMotor2;
   break;
  }
 }
 return 0;
}
```

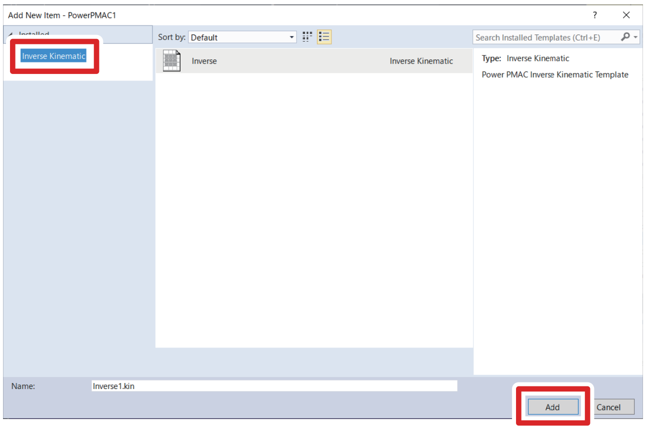
14 Add the text shown on the right to the top of the usercode.c file.

```
#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 In the Solution Explorer, right-click **Kinematics Routines** under **PMAC Script Language** and select **Add** – **New Item...**.



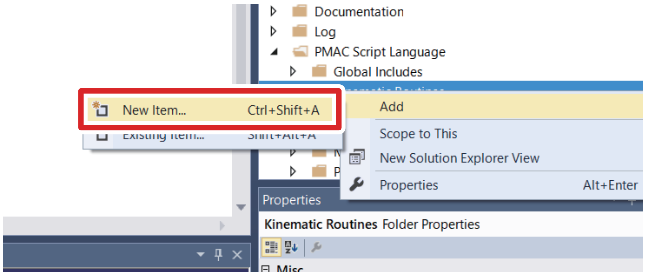
16 Select **Inverse Kinematic** and click the **Add** button.



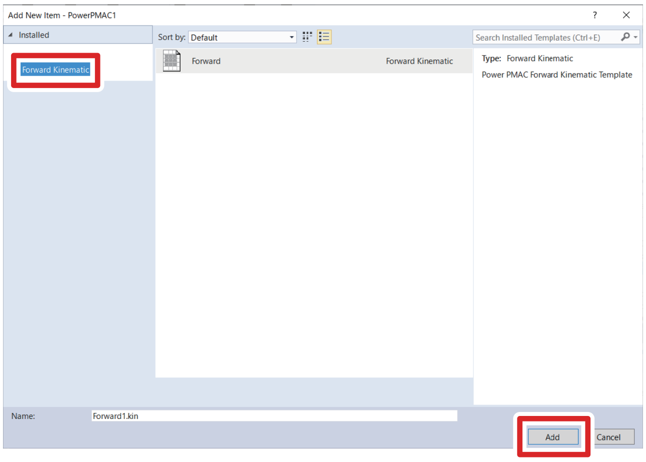
17 Write the text shown on the right to the Inverse1.kin file.

```
open inverse(1)

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

close
```

18 In the Solution Explorer, right-click **Kinematics Routines** under **PMAC Script Language** and select **Add** – **New Item...**.



19 Select **Forward Kinematic** and click the **Add** button.



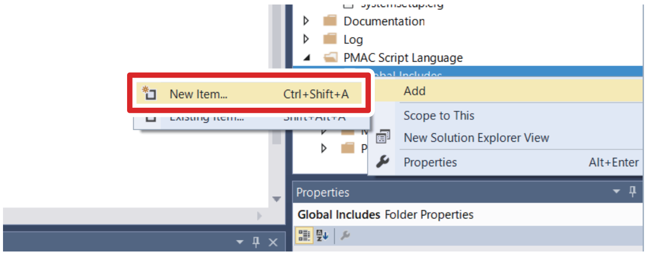
20 Write the text shown on the right to the Forward1.kin file.

```
open forward(1)

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

close
```

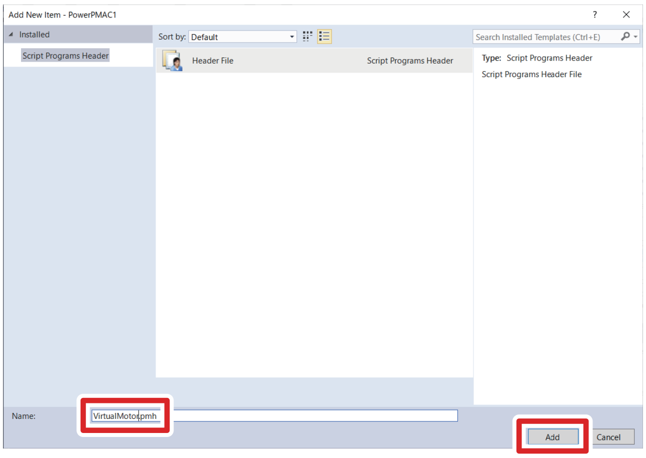
3-3-2 Determination of Filter's Coefficients

Adjust the KdGain parameter of the filter used for the MOTF.

1 In the Solution Explorer, right-click **Global Includes** under **PMAC Script Language** and select **Add** – **New Item...**.



2 In the **Name** box, type *VirtualMotor.pmh* and click the **Add** button.



Type: Script Programs Header

Script Programs Header File

Name: VirtualMotor.pmh

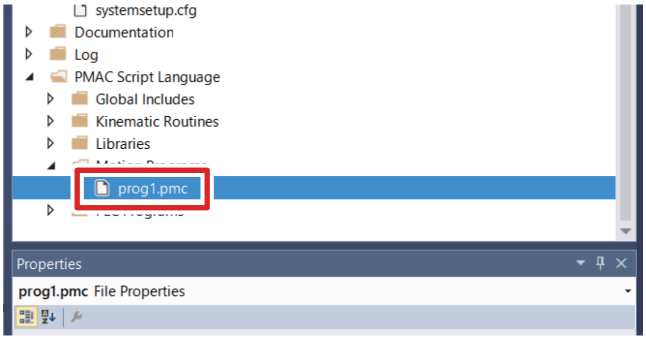
3 Open the **VirtualMotor.pmh** file under **PMAC Script Language** – **Global Includes** in the Solution Explorer.

4 Write the text shown on the right to the VirtualMotor.pmh file.

```
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 Open the **prog1.pmc** file under **PMAC Script Language – Motion Programs** in the Solution Explorer.



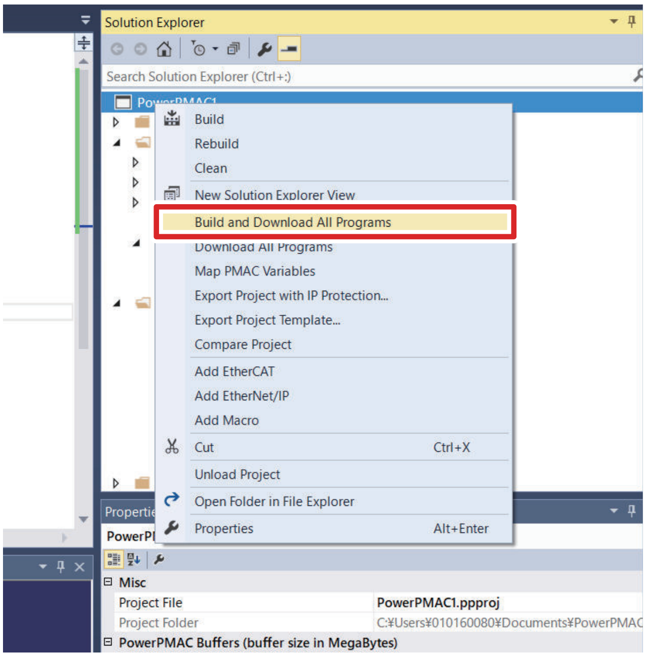
6 Write the program to use for processing to the prog1.pmc file.

The text on the right is an example of the processing program.

```
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
```

7 Download the project.

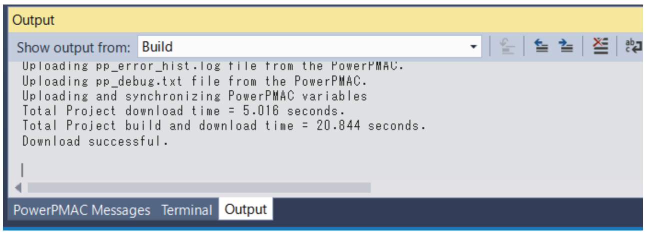
Right click on the Solution Explorer project name at the upper right of the Power PMAC IDE screen, select **Build and Download All Programs**, and execute Build and Download.


Solution Explorer
Search Solution Explorer (Ctrl+;)
PowerPMAC1
Build
Rebuild
Clean
New Solution Explorer View
Build and Download All Programs
Download All Programs
Map PMAC Variables
Export Project with IP Protection...
Export Project Template...
Compare Project
Add EtherCAT
Add EtherNet/IP
Add Macro
Cut Ctrl+X
Unload Project
Open Folder in File Explorer
Properties Alt+Enter
Properties
PowerPl
Misc
PowerPMAC Buffers (buffer size in MegaBytes)

Project File: PowerPMAC1.ppproj
Project Folder: C:¥Users¥010160080¥Documents¥PowerPMAC

8 Make sure that there are no errors in the Output window.

- If the transfer failed, check the content of the error in the Output window.

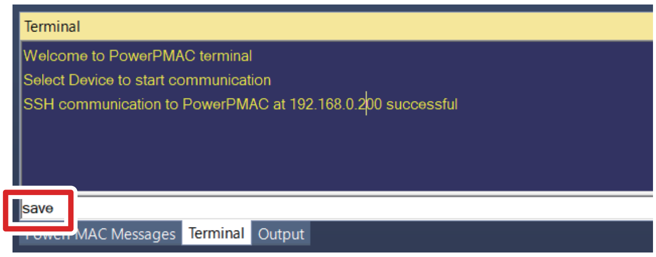


Show output from: Build

```
Uploading pp_error_hist.log file from the PowerPMAC.
Uploading pp_debug.txt file from the PowerPMAC.
Uploading and synchronizing PowerPMAC variables
Total Project download time = 5.016 seconds.
Total Project build and download time = 20.844 seconds.
Download successful.
```

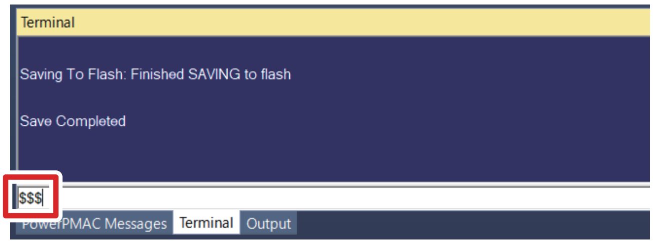
9 Type the **save** command in the Power PMAC IDE Terminal.

- When the save is completed, "Save Completed" is displayed in the Terminal.



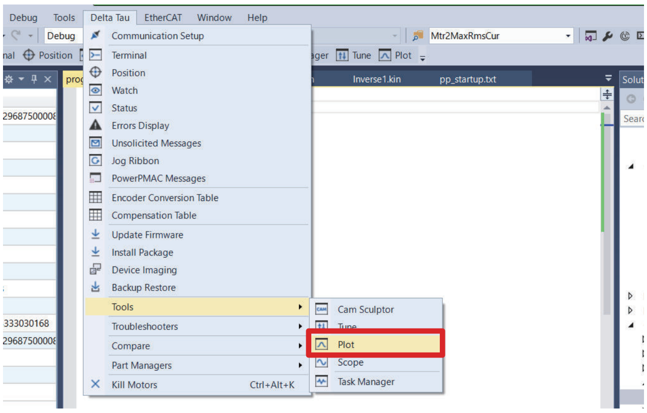
```
Terminal
Welcome to PowerPMAC terminal
Select Device to start communication
SSH communication to PowerPMAC at 192.168.0.200 successful

save
```

10 Type the **$$$** command in the Power PMAC IDE Terminal.



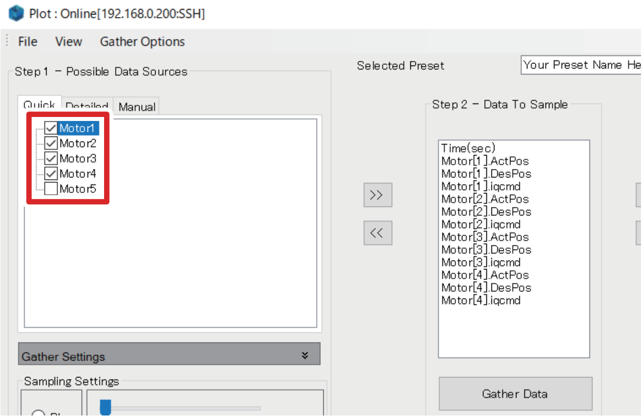
```
Terminal

Saving To Flash: Finished SAVING to flash

Save Completed

$$$
```


11 Select **Tools** – **Plot** from the **Delta Tau** menu.



12 In the **Step1 – Possible Data Sources** area, select the check boxes for the following items.

- **Motor1**
- **Motor2**
- **Motor3**
- **Motor4**



Plot : Online[192.168.0.200:SSH]

Step 1 – Possible Data Sources

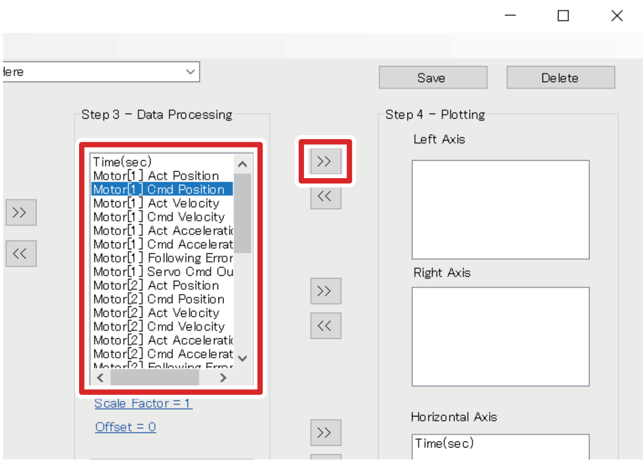
☑ Motor1
☑ Motor2
☑ Motor3
☑ Motor4
□ Motor5

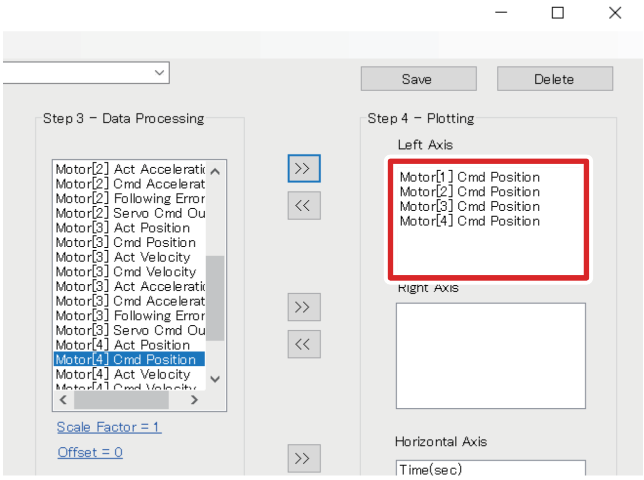
Step 2 – Data To Sample

```
Time(sec)
Motor[1].ActPos
Motor[1].DesPos
Motor[1].iqcmd
Motor[2].ActPos
Motor[2].DesPos
Motor[2].iqcmd
Motor[3].ActPos
Motor[3].DesPos
Motor[3].iqcmd
Motor[4].ActPos
Motor[4].DesPos
Motor[4].iqcmd
```

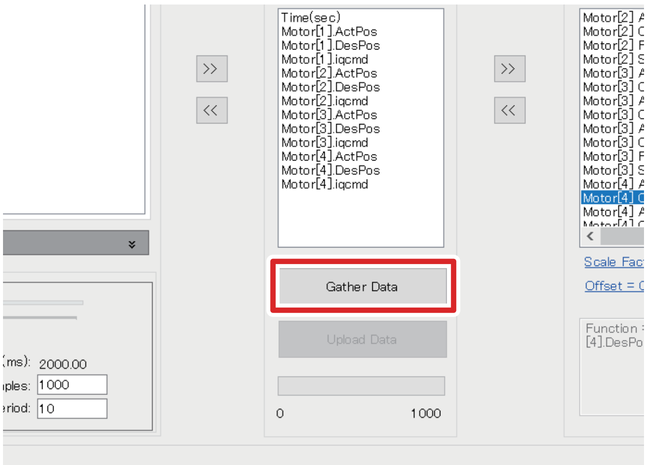
13 In the **Step3 – Data Processing** area, select the following items and click the **>>** button.

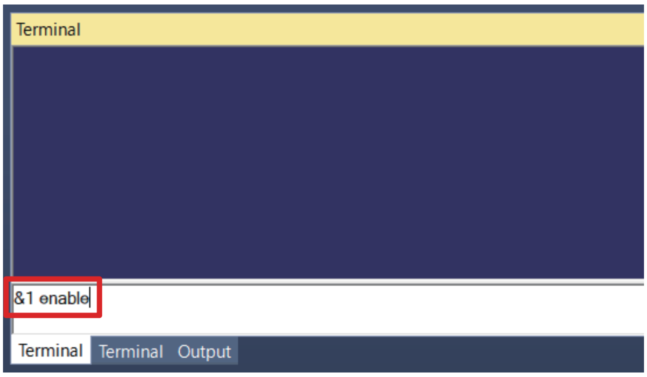
- **Motor[1] Cmd Position**
- **Motor[2] Cmd Position**
- **Motor[3] Cmd Position**
- **Motor[4] Cmd Position**



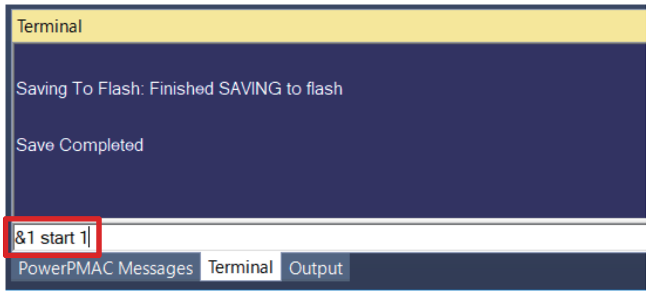


14 Click the **Gather Data** button and type the **&1 enable** and the **&1 start 1** command in the Power PMAC IDE Terminal.



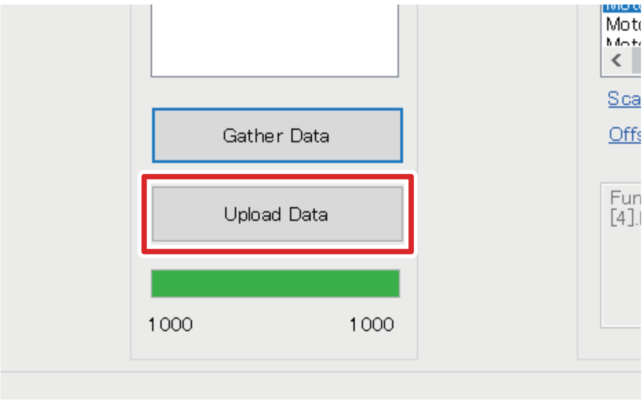


```
&1 enable
```

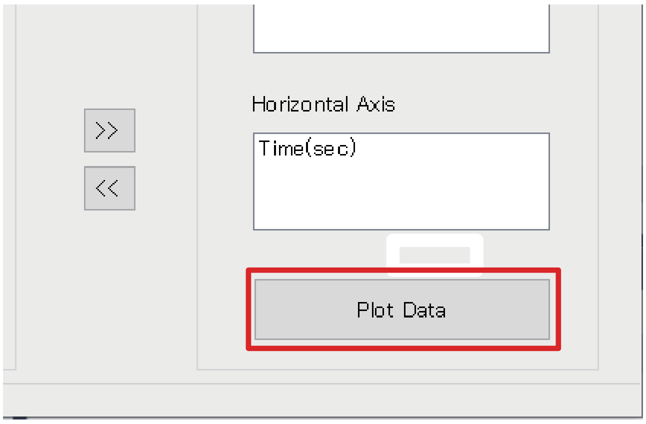


```
Saving To Flash: Finished SAVING to flash

Save Completed

&1 start 1
```

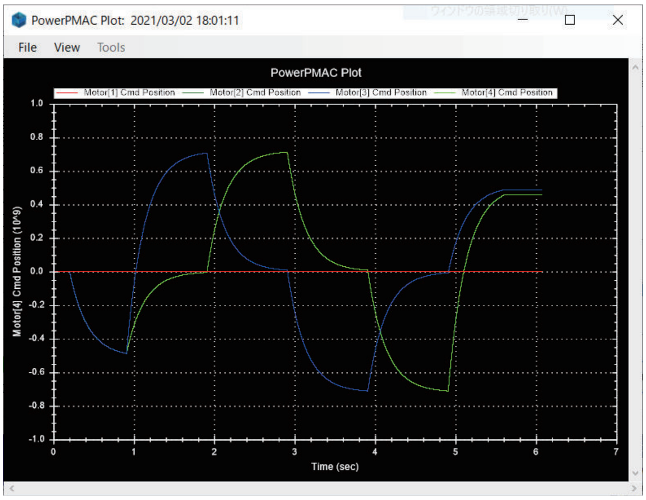
15 Click the **Upload Data** button.



16 Click the **Plot Data** button.



17 Confirm that the position of each axis is displayed as shown on the right.



18 If either Motor[3] Cmd Position or Motor[4] Cmd Position exceeds the maximum or the minimum command value of the Galvo Scanner, open the **global definitions.pmh** file and edit the KdgainX and KdgainY settings. Then, perform steps 7 to 17 again.

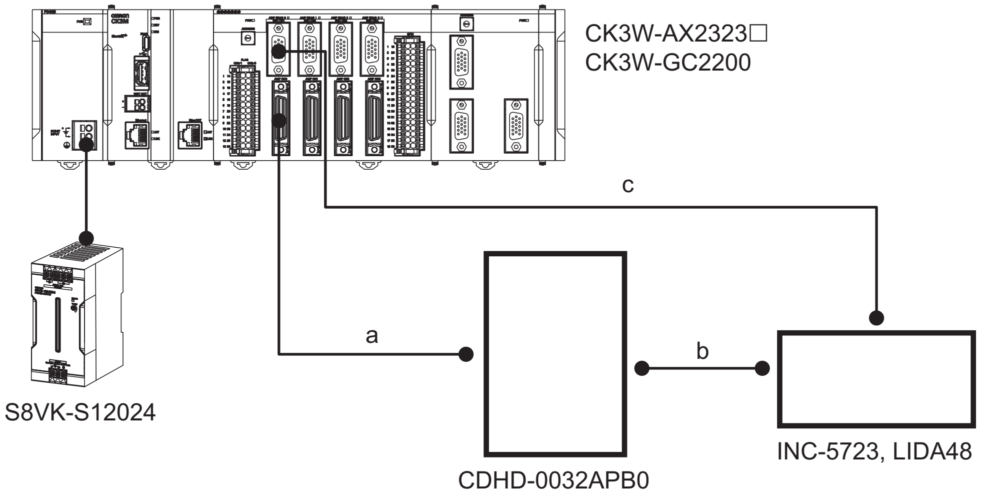
19 If there is no problem with the filter response in step 18, right-click the **VirtualMotor.pmh** file in the Solution Explorer and select **Delete**.

3-4 Setup of the Motors and Encoder

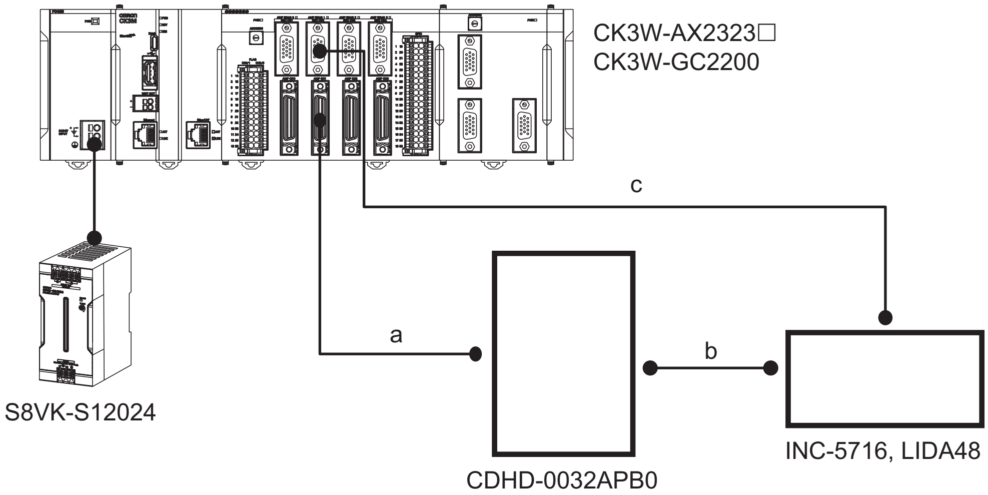
3-4-1 Wiring of Devices

The figures below show how to wire the CK3M to the Servo Drive and linear motors.

- Wiring of the X axis



- Wiring of the Y axis



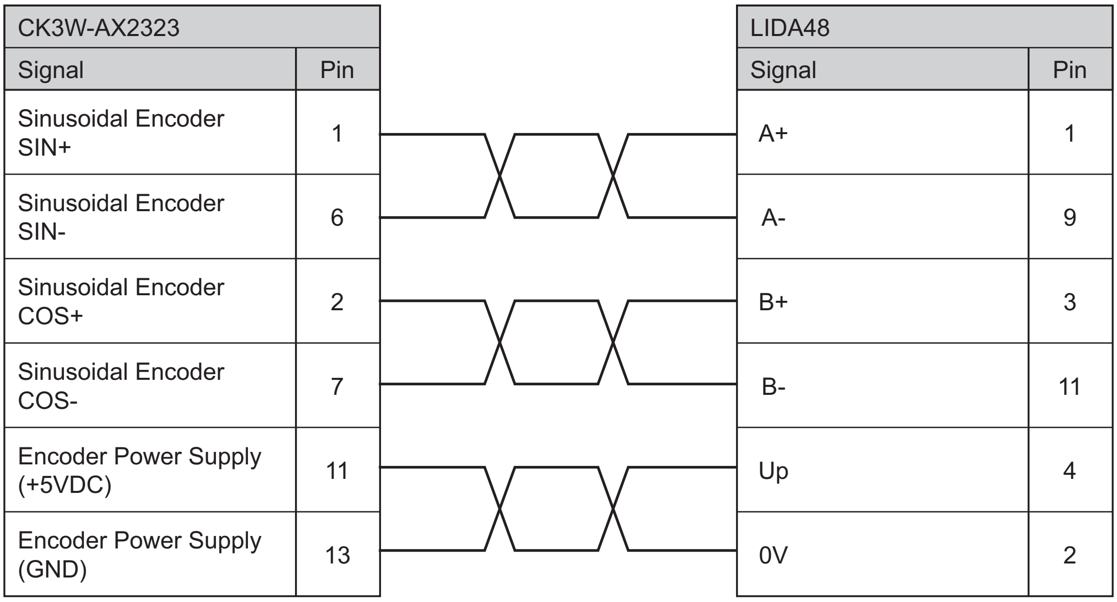
Connect the sections a, b, and c in the above figures as follows.

a. Connection between the Controller and the servo amplifier
   Connect the amplifier connector of the CK3W-AX2323□ to the C2 connector of the servo amplifier with the following dedicated cable.

Manufacturer	Name	Model	Length
OMRON	Direct PWM Cable	CK3W-CAAD009A	0.9 m
		CK3W-CAAD018A	1.8 m
		CK3W-CAAD036A	3.6 m

b. Connection between the servo amplifier and each servo motor
   Connect the motor connector of the linear motor to the P2 connector of the servo amplifier.

c. Connection between the Controller and each servo motor
   Connect the encoder connector of the CK3W-AX2323□ to the connector of the linear encoder with the dedicated cable CK3W-CAEA03A.



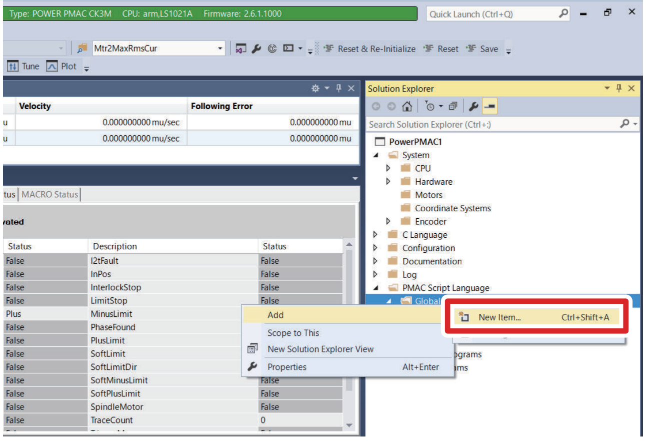
CK3W-AX2323	
Signal	Pin
Sinusoidal Encoder SIN+	1
Sinusoidal Encoder SIN-	6
Sinusoidal Encoder COS+	2
Sinusoidal Encoder COS-	7
Encoder Power Supply (+5VDC)	11
Encoder Power Supply (GND)	13

LIDA48	
Signal	Pin
A+	1
A-	9
B+	3
B-	11
Up	4
0V	2

3-4-2 Programming

Program the settings for controlling the servo amplifier and the linear motor.

1 In the Solution Explorer, right-click **Global Includes** under **PMAC Script Language** and select **Add** – **New Item...**.

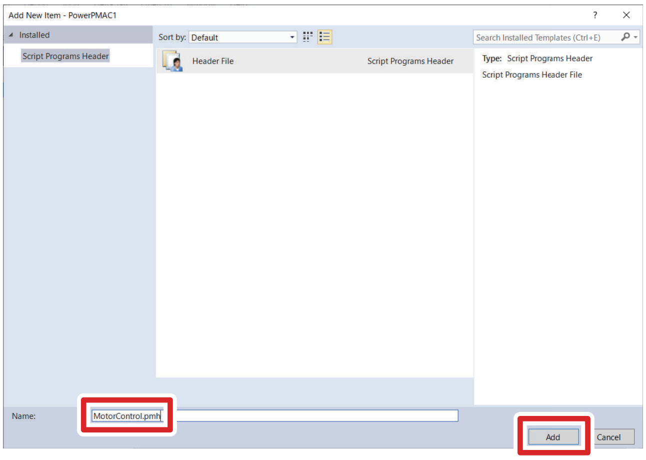


Type: POWER PMAC CK3M   CPU: arm,LS1021A   Firmware: 2.6.1.1000

Velocity	Following Error
0.000000000 mu/sec	0.000000000 mu
0.000000000 mu/sec	0.000000000 mu

Status	Description	Status
False	I2tFault	False
False	InPos	False
False	InterlockStop	False
False	LimitStop	False
Plus	MinusLimit	
False	PhaseFound	
False	PlusLimit	
False	SoftLimit	
False	SoftLimitDir	
False	SoftMinusLimit	False
False	SoftPlusLimit	False
False	SpindleMotor	False
False	TraceCount	0

2 In the **Name** box, type *MotorControl.pmh* and click the **Add** button.

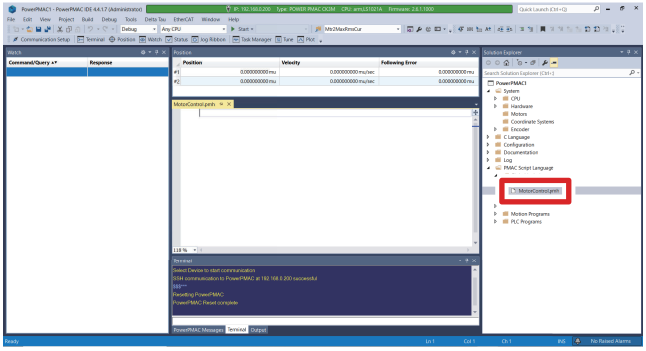


Type: Script Programs Header

Script Programs Header File

Name: MotorControl.pmh

3 Open the **MotorControl.pmh** file under **PMAC Script Language** – **Global Includes** in the Solution Explorer.



	Position	Velocity	Following Error
#1	0.000000000 mu	0.000000000 mu/sec	0.000000000 mu
#2	0.000000000 mu	0.000000000 mu/sec	0.000000000 mu

```
Select Device to start communication
SSH communication to PowerPMAC at 192.168.0.200 successful
$$$***
Resetting PowerPMAC
PowerPMAC Reset complete
```

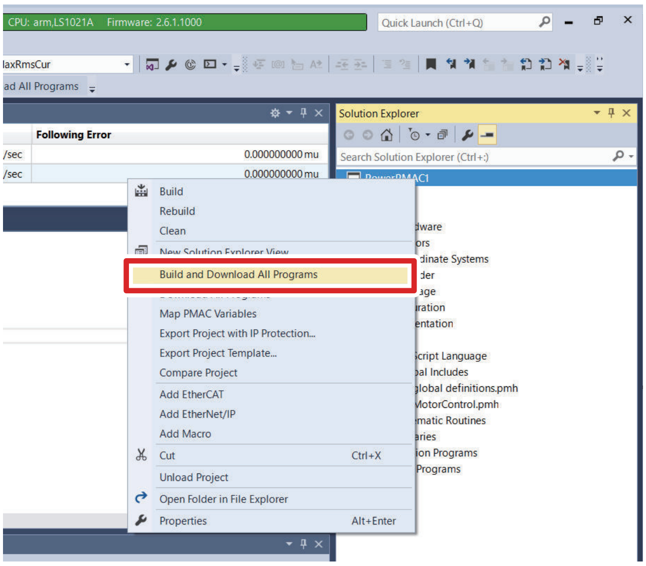
4 Write the following text to the MotorControl.pmh file.

- For details on the annotated settings, refer to *Section 4 How to Customize Various Settings* on page 4-1.

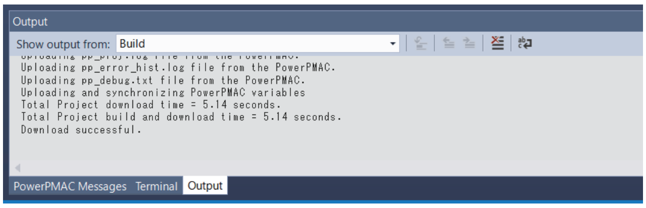
```
Sys.WpKey=$AAAAAAAA

//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=1.0//*9
Motor[2].FatalFeLimit=1.0//*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=9.0*32768*COSD(30)/11.25//*20
Motor[2].MaxDac=9.0*32768*COSD(30)/11.25//*20
Motor[1].I2tSet=3.0*32768*COSD(30)/11.25//*21
Motor[2].I2tSet=3.0*32768*COSD(30)/11.25//*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=15//*23
Gate3[0].Chan[1].PwmDeadTime=15//*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=$fffffc;//*26
Gate3[0].AdcAmpHeaderBits=2;//*27
Gate3[0].AdcAmpClockDiv=5;//*28
Gate3[0].AdcEncClockDiv=5//*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 Right click on the Solution Explorer project name at the upper right of the Power PMAC IDE screen, select **Build and Download All Programs**, and execute Build and Download.



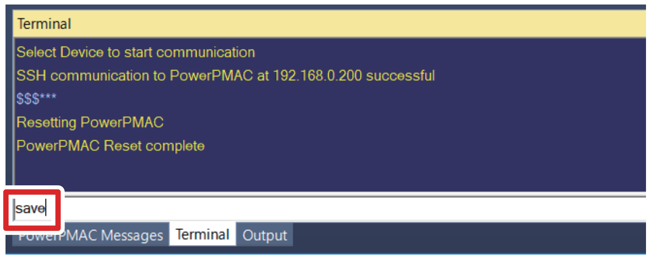
6 Make sure that there are no errors in the Output window.



Show output from: Build

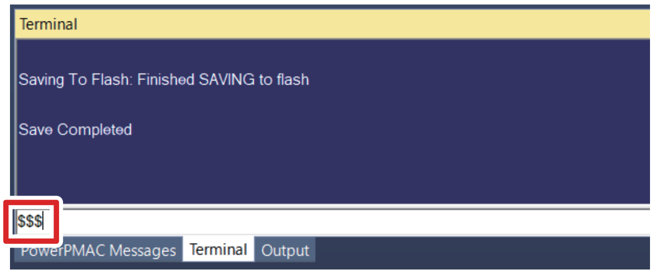
```
Uploading pp_proj.log file from the PowerPMAC.
Uploading pp_error_hist.log file from the PowerPMAC.
Uploading pp_debug.txt file from the PowerPMAC.
Uploading and synchronizing PowerPMAC variables
Total Project download time = 5.14 seconds.
Total Project build and download time = 5.14 seconds.
Download successful.
```

7 Type the **save** command in the Power PMAC IDE Terminal.



```
Select Device to start communication
SSH communication to PowerPMAC at 192.168.0.200 successful
$$$***
Resetting PowerPMAC
PowerPMAC Reset complete
save
```

8 Type the **$$$** command in the Power PMAC IDE Terminal.

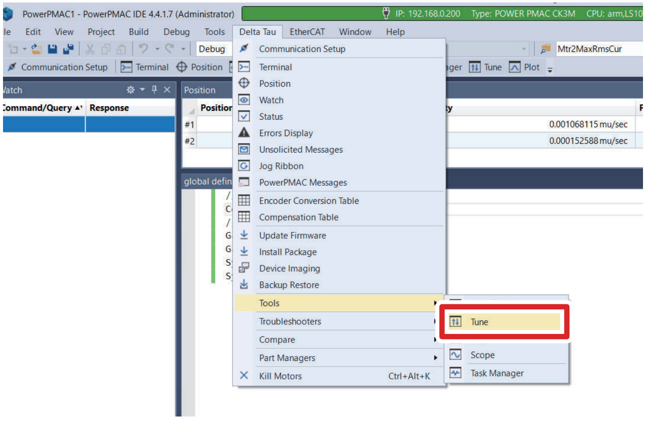



```
Saving To Flash: Finished SAVING to flash

Save Completed
$$$
```

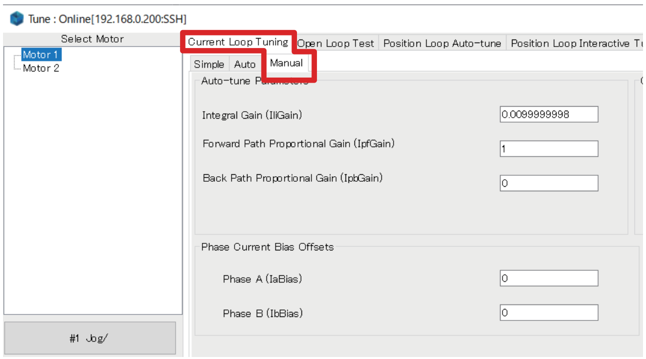
3-4-3 Tuning of the Current Loop

Make the tuning of the current loop of the servo amplifier.

1 Select **Tools** – **Tune** from the **Delta Tau** menu to open the Tune screen.



2 Select **Current Loop Tuning** – **Manual** tab page on the Tune screen.



Integral Gain (IliGain) 0.0099999998

Forward Path Proportional Gain (IpfGain) 1

Back Path Proportional Gain (IpbGain) 0

Phase A (IaBias) 0

Phase B (IbBias) 0

3 Set the following parameters.

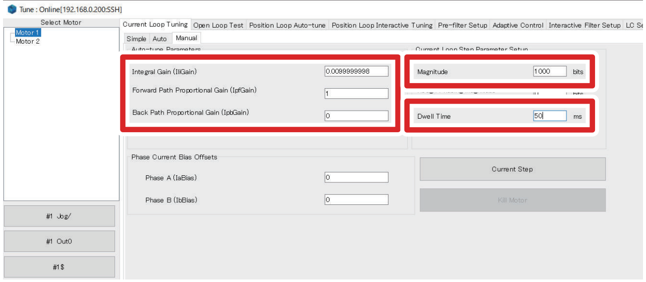
IliGain: 0.0099999998 (Default)

IpfGain: 1 (Default)

IpbGain: 0

Magnitude: 1,000 bits

Dwell Time: 50 ms



Tune : Online[192.168.0.200:SSH]

Integral Gain (IliGain): 0.0099999998

Forward Path Proportional Gain (IpfGain): 1

Back Path Proportional Gain (IpbGain): 0

Magnitude: 1000 bits

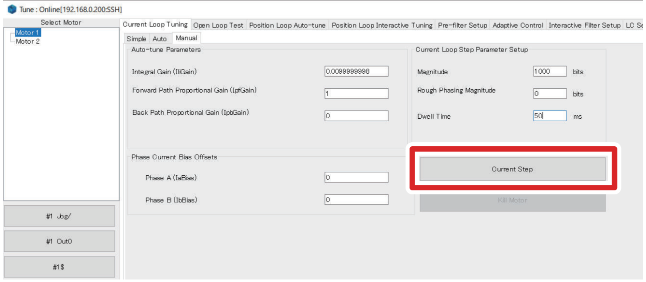
Dwell Time: 50 ms

Phase A (IaBias): 0

Phase B (IbBias): 0

4 Click the **Current Step** button.

- The current step response is displayed.



Tune : Online[192.168.0.200:SSH]

Integral Gain (IliGain): 0.0099999998

Forward Path Proportional Gain (IpfGain): 1

Back Path Proportional Gain (IpbGain): 0

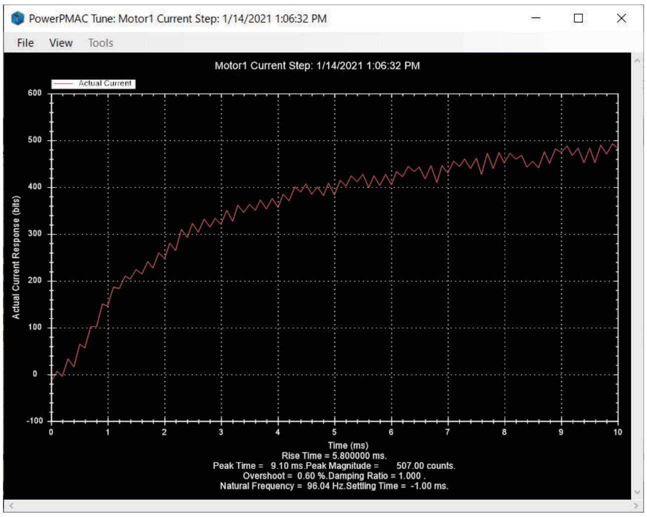
Magnitude: 1000 bits

Rough Phasing Magnitude: 0 bits

Dwell Time: 50 ms

Phase A (IaBias): 0

Phase B (IbBias): 0

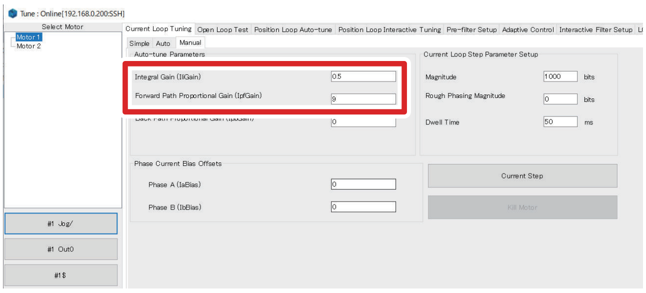


PowerPMAC Tune: Motor1 Current Step: 1/14/2021 1:06:32 PM

```
Rise Time = 5.800000 ms.
Peak Time =   9.10 ms.Peak Magnitude =      507.00 counts.
Overshoot =  0.60 %.Damping Ratio = 1.000 .
Natural Frequency =  96.04 Hz.Settling Time =  -1.00 ms.
```

5 Adjust the **IliGain** and **IpfGain** values to obtain the desired response characteristics.

- If the rising edge response is slow, increase the **IliGain** value.
- If the amount of overshooting or oscillation is large, increase the **IpfGain** value.
- Gradually increase each of the gain values.



Tune : Online[192.168.0.200:SSH]

Select Motor: Motor 1

Current Loop Tuning – Manual

Auto-tune Parameters

Integral Gain (IliGain): 0.5

Forward Path Proportional Gain (IpfGain): 9

Back Path Proportional Gain (IpbGain): 0

Current Loop Step Parameter Setup

Magnitude: 1000 bits

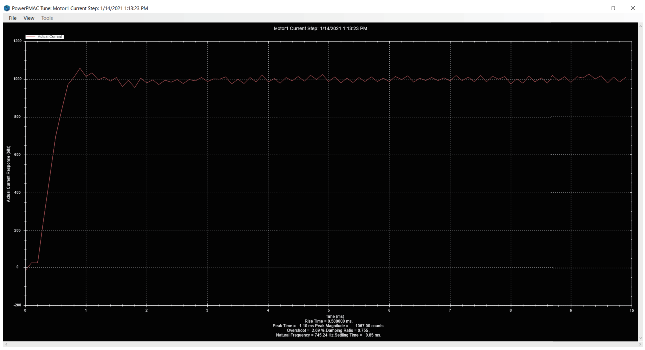
Rough Phasing Magnitude: 0 bits

Dwell Time: 50 ms

Phase Current Bias Offsets

Phase A (IaBias): 0

Phase B (IbBias): 0



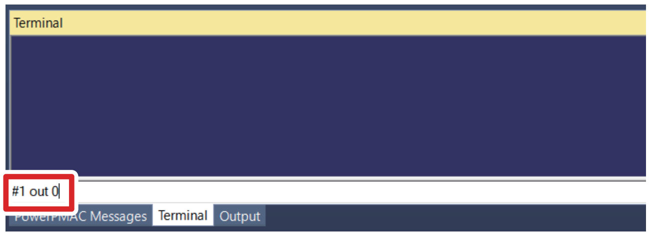
PowerPMAC Tune: Motor1 Current Step: 1/14/2021 1:13:23 PM

Rise Time = 0.500000 ms.
Peak Time = 1.10 ms.Peak Magnitude = 1067.00 counts.
Overshoot = 2.69 %.Damping Ratio = 0.755 .
Natural Frequency = 745.24 Hz.Settling Time = 0.85 ms.

3-4-4 Establishment of the Phase Reference

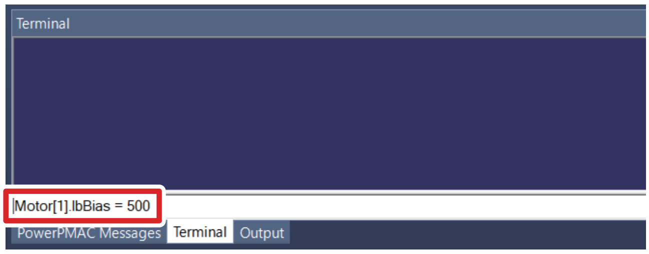
Establish the phase reference of the linear motor.

1 Type the **#1 out 0** command in the Power PMAC IDE Terminal.



```
#1 out 0
```

2 Type the **Motor[1].IbBias=500** command in the Power PMAC IDE Terminal.

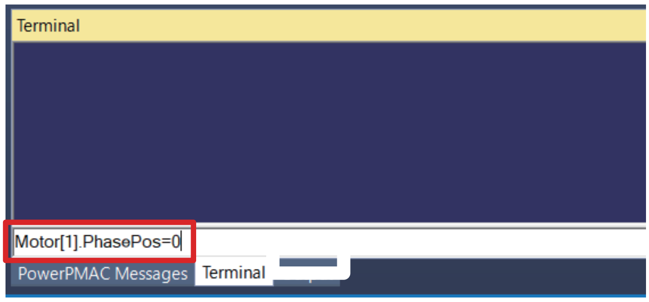


```
Motor[1].IbBias = 500
```

3 Move the X axis manually to confirm that the servo is ON.

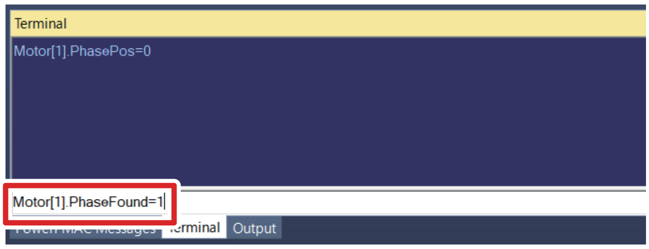
- If the servo is not ON, gradually increase the Motor[1].IbBias value.

4 Type the **Motor[1].PhasePos=0** command in the Power PMAC IDE Terminal.



```
Motor[1].PhasePos=0
```

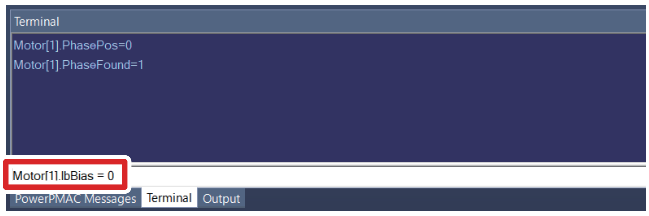
5 Type the **Motor[1].PhaseFound=1** command in the Power PMAC IDE Terminal.



```
Motor[1].PhasePos=0
```

```
Motor[1].PhaseFound=1
```

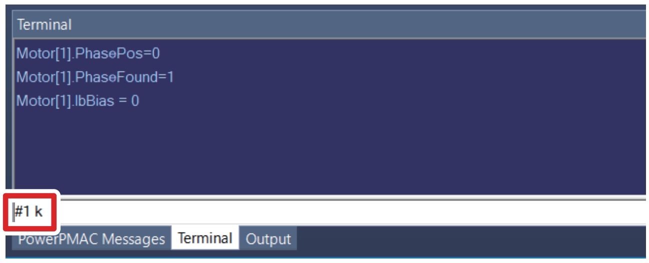
6 Type the **Motor[1].IbBias=0** command in the Power PMAC IDE Terminal.



```
Motor[1].PhasePos=0
Motor[1].PhaseFound=1
```

```
Motor[1].IbBias = 0
```

7 Type the **#1 k** command in the Power PMAC IDE Terminal.



```
Motor[1].PhasePos=0
Motor[1].PhaseFound=1
Motor[1].IbBias = 0
```

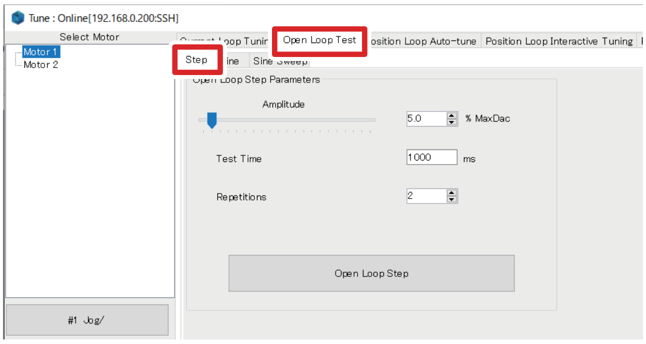
```
#1 k
```

8 Perform steps 1 to 7 for Axis 2 (Y axis) in the same way.

3-4-5 Open-loop Testing

Operate the linear motor on the open loop system to confirm that the program is correct.

1 Select **Tools** – **Tune** from the **Delta Tau** menu to open the Tune screen. Then, select **Open LoopTest** – **Step** tab page.



Tune : Online[192.168.0.200:SSH]

Open Loop Step Parameters

Amplitude 5.0 % MaxDac

Test Time 1000 ms

Repetitions 2

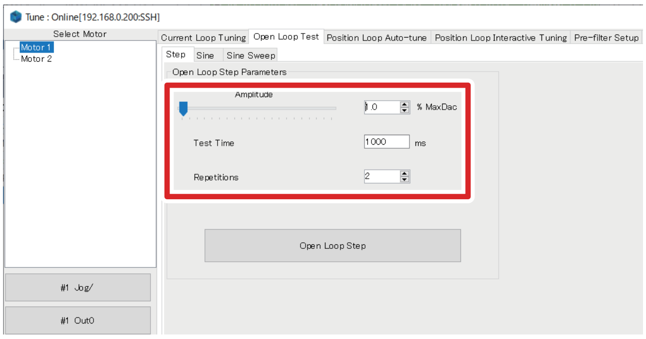
2 Set the following tuning parameters.

Amplitude: 1.0%[*1]

Test Time: 1,000 ms

Repetitions: 2

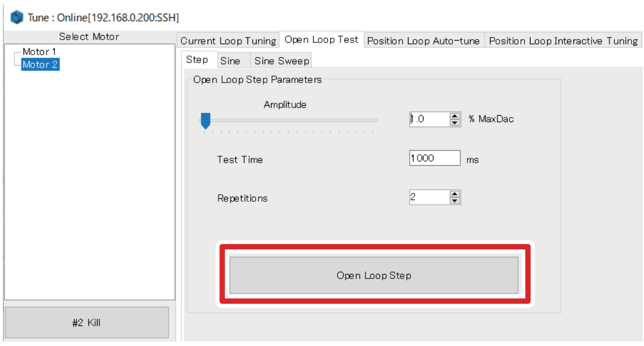
*1. If the motor does not rotate, set a larger value.



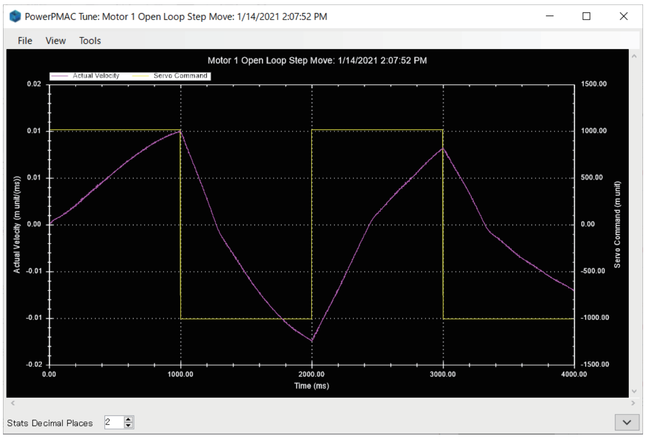
Amplitude: 1.0 % MaxDac
Test Time: 1000 ms
Repetitions: 2

3 Click the **Open Loop Step** button.

- The motor makes a reciprocating motion, and then the test results as shown on the right are displayed.
- If the motor does not rotate, increase the **Test Amplitude** value.
- Here, the test results when **Test Amplitude** is set to 8.0% are shown.



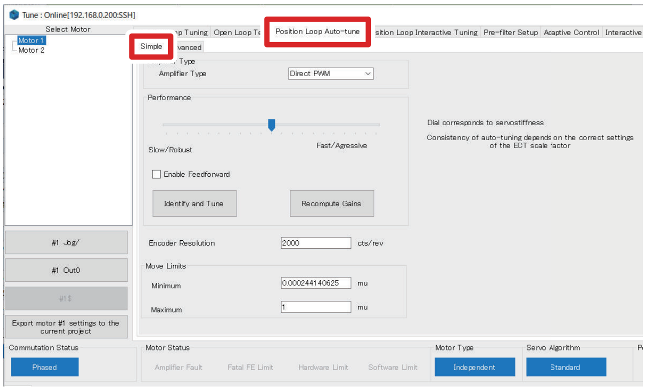
Amplitude: 1.0 % MaxDac
Test Time: 1000 ms
Repetitions: 2



3-4-6 Position Loop Auto-tuning

Perform the position loop auto-tuning in the linear motor.

1 Select **Position Loop Auto-tune** – **Simple** tab page on the Tune screen.



Amplifier Type: Direct PWM

Dial corresponds to servostiffness

Consistency of auto-tuning depends on the correct settings of the ECT scale factor

□ Enable Feedforward

Encoder Resolution: 2000 cts/rev

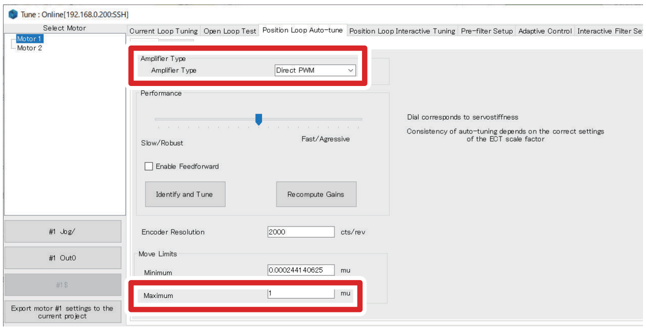
Minimum: 0.000244140625 mu

Maximum: 1 mu

2 Set the following parameters.

Amplifier Type: Direct PWM

Maximum: 1 mu (1 mm)



Amplifier Type: Direct PWM

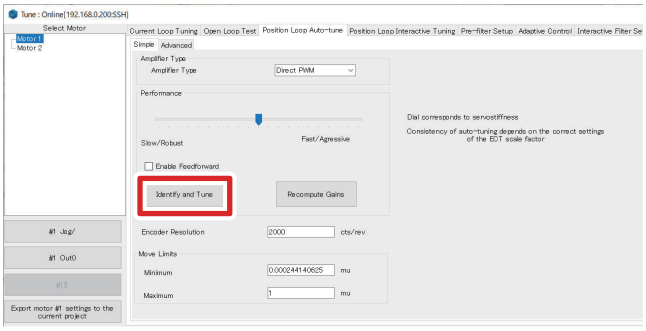
☐ Enable Feedforward

Encoder Resolution: 2000 cts/rev

Minimum: 0.000244140625 mu

Maximum: 1 mu

3 Click the **Identify and Tune** button.



Amplifier Type: Direct PWM

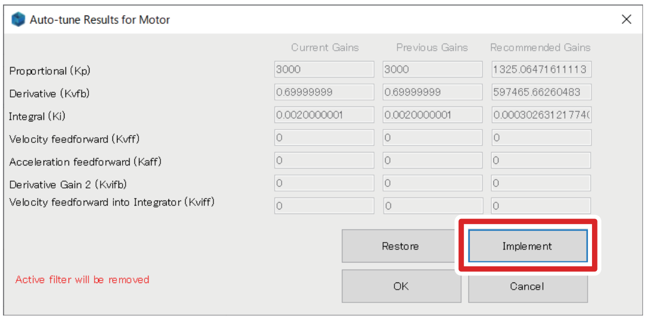
☐ Enable Feedforward

Encoder Resolution: 2000 cts/rev

Minimum: 0.000244140625 mu

Maximum: 1 mu

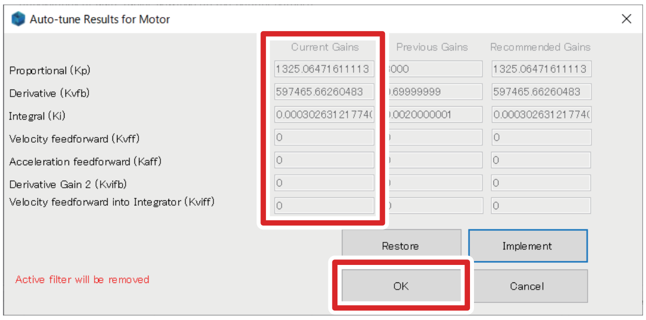
4 When the screen on the right appears, click the **Implement** button.



	Current Gains	Previous Gains	Recommended Gains
Proportional (Kp)	3000	3000	1325.06471611113
Derivative (Kvfb)	0.69999999	0.69999999	597465.66260483
Integral (Ki)	0.0020000001	0.0020000001	0.00030263121774(
Velocity feedforward (Kvff)	0	0	0
Acceleration feedforward (Kaff)	0	0	0
Derivative Gain 2 (Kvifb)	0	0	0
Velocity feedforward into Integrator (Kviff)	0	0	0

Active filter will be removed

5 Confirm that the **Current Gains** settings reflect the **Recommended Gains** values and click the **OK** button.

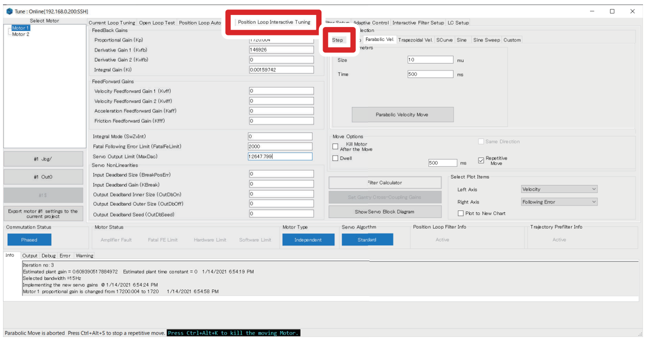
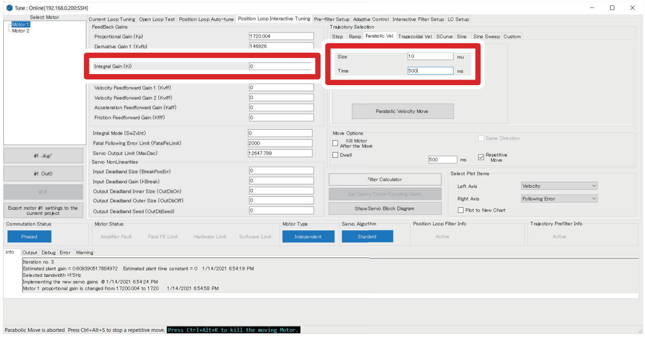
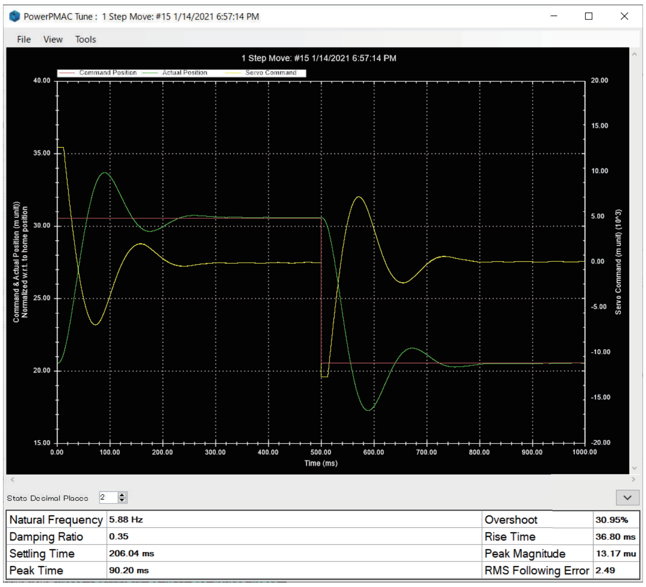


	Current Gains	Previous Gains	Recommended Gains
Proportional (Kp)	1325.06471611113	000	1325.06471611113
Derivative (Kvfb)	597465.66260483	.69999999	597465.66260483
Integral (Ki)	0.00030263121774(	.0020000001	0.00030263121774(
Velocity feedforward (Kvff)	0	)	0
Acceleration feedforward (Kaff)	0	)	0
Derivative Gain 2 (Kvifb)	0	)	0
Velocity feedforward into Integrator (Kviff)	0	)	0

Active filter will be removed

3-4-7 Position Loop Interactive Tuning

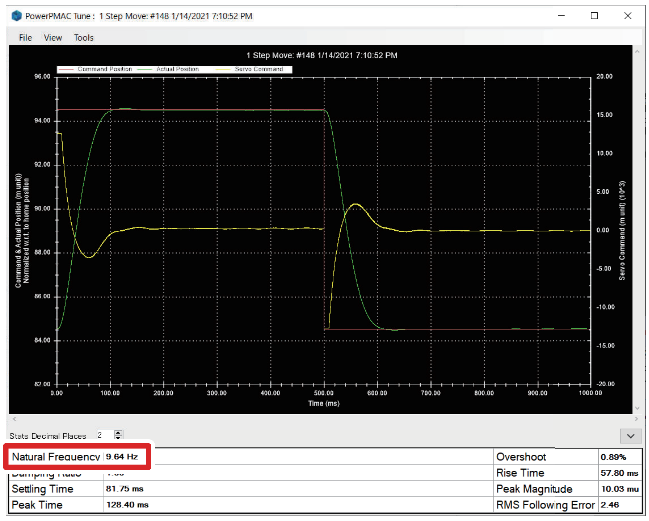
Perform the position loop interactive tuning in the linear motor.

1	Select **Position Loop Interactive Tuning** – **Step** tab page on the Tune screen.	 Iteration no: 3 Estimated plant gain = 0.609390517884972 Estimated plant time constant = 0 1/14/2021 6:54:19 PM Selected bandwidth =15Hz Implementing the new servo gains @ 1/14/2021 6:54:24 PM Motor 1 proportional gain is changed from 17200.004 to 1720 1/14/2021 6:54:58 PM Parabolic Move is aborted Press Ctrl+Alt+S to stop a repetitive move. Press Ctrl+Alt+K to kill the moving Motor.
2	Set the following parameters. **FeedBack Gains** **Integral Gains (Ki)**: 0 **Move Parameters** **Size**: 10 mu **Time**: 1,000 ms	 Iteration no: 3 Estimated plant gain = 0.609390517884972 Estimated plant time constant = 0 1/14/2021 6:54:19 PM Selected bandwidth =15Hz Implementing the new servo gains @ 1/14/2021 6:54:24 PM Motor 1 proportional gain is changed from 17200.004 to 1720 1/14/2021 6:54:58 PM Parabolic Move is aborted Press Ctrl+Alt+S to stop a repetitive move. Press Ctrl+Alt+K to kill the moving Motor.
3	Click the **Step Move** button.	
4	Check the step response characteristics.	

Natural Frequency	5.88 Hz	Overshoot	30.95%
Damping Ratio	0.35	Rise Time	36.80 ms
Settling Time	206.04 ms	Peak Magnitude	13.17 mu
Peak Time	90.20 ms	RMS Following Error	2.49

5 Adjust the **Proportional Gain (Kp)** and **Derivative Gain 1 (Kvfb)** values to obtain the desired response characteristics.

- If the rising edge response is slow, increase the **Proportional Gain (Kp)** value.
- If the amount of overshooting or oscillation is large, increase the **Derivative Gain 1 (Kvfb)** value.
- If the steady-state error is large, increase the **Integral Gain 1 (Ki)** value.
- Gradually increase each of the gain values.
- After you adjust each parameter, finish the tuning when the value of **Natural Frequency** has stopped increasing.



Stats Decimal Places 2

Natural Frequency	9.64 Hz	Overshoot	0.89%
Damping Ratio	1.00	Rise Time	57.80 ms
Settling Time	81.75 ms	Peak Magnitude	10.03 mu
Peak Time	128.40 ms	RMS Following Error	2.46

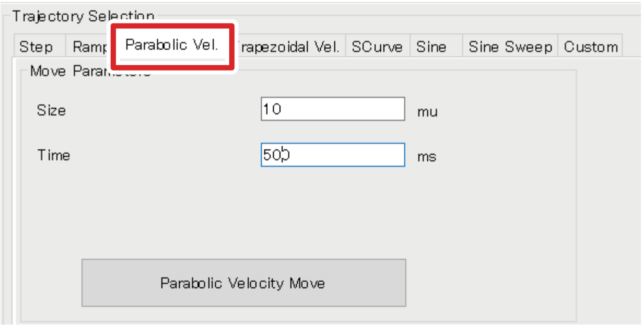
6 Select the **Parabolic Vel.** tab page and set the following parameters.

Size: 10 mu (10 mm)

Time: 500 ms

Left Axis: Velocity

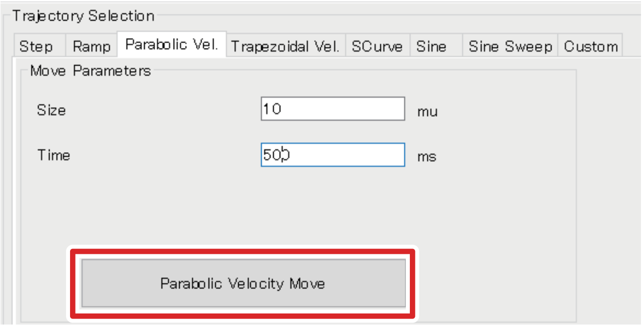
Right Axis: Following Error



Size 10 mu

Time 500 ms

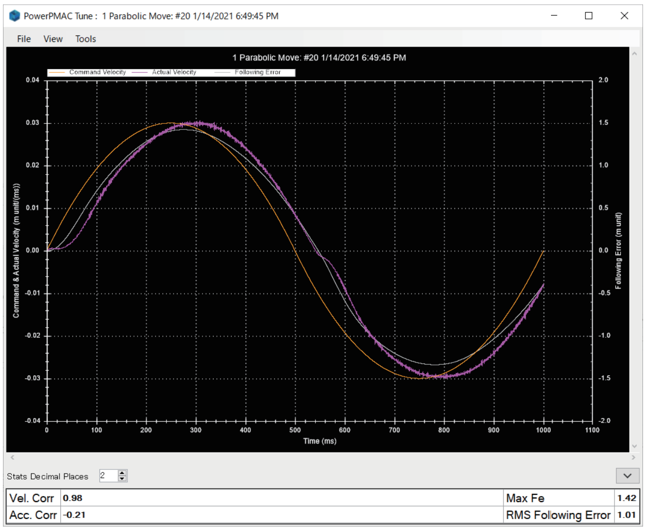
7 Click the **Parabolic Velocity Move** button.



Size 10 mu

Time 500 ms

8 Check the parabolic response characteristics of velocity.

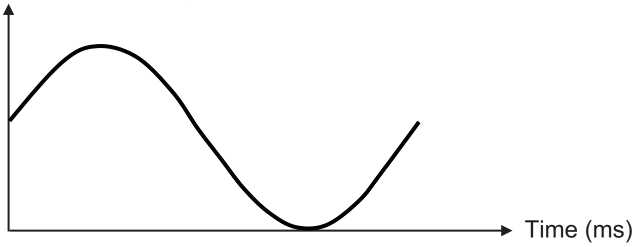


Stats Decimal Places 2

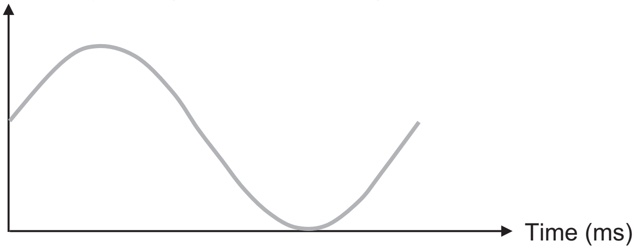
Vel. Corr	0.98	Max Fe	1.42
Acc. Corr	-0.21	RMS Following Error	1.01

9 If the **Following Error** has a positive correlation with the velocity, increase the **Kvff** value. If it has a negative correlation, decrease the **Kvff** value.

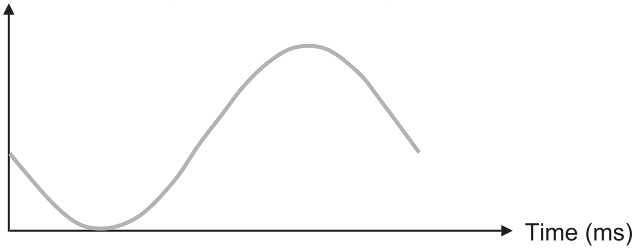
Command Velocity



Following Error (positive correlation)

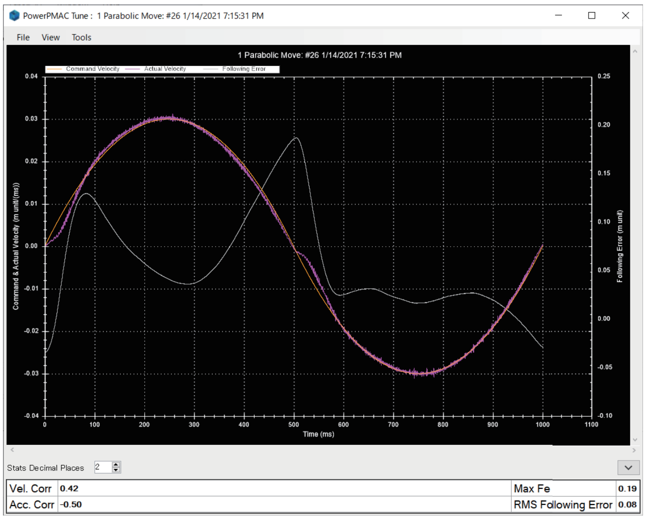


Following Error (negative correlation)



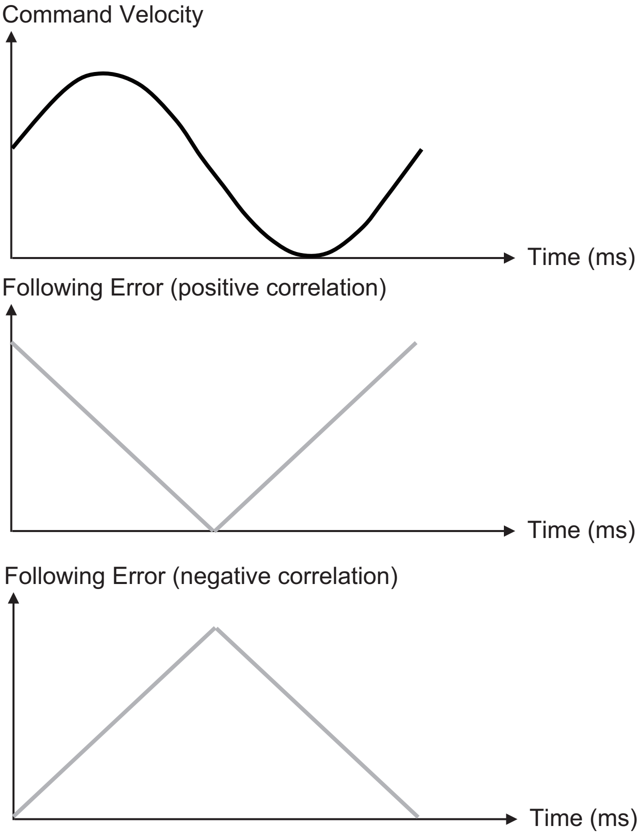
10 Click the **Parabolic Velocity Move** button again.

- Repeat this operation until the **Following Error** is no longer correlated with the velocity.



Vel. Corr	0.42	Max Fe	0.19
Acc. Corr	-0.50	RMS Following Error	0.08

11 Similarly, if the **Following Error** has a correlation with the **Acceleration** or friction, increase or decrease the **Kaff** or **Kfff** value.


Command Velocity

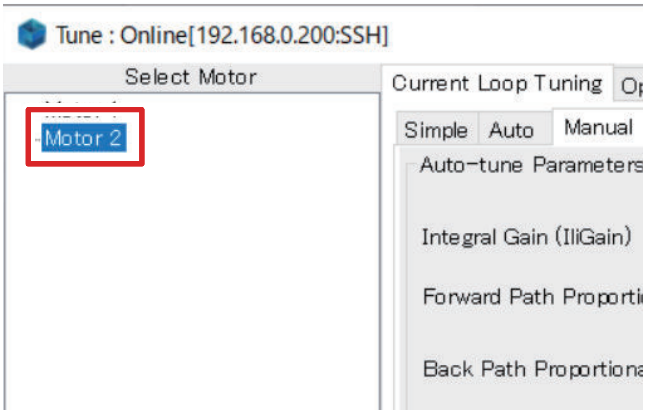
Following Error (positive correlation)

Following Error (negative correlation)

3-4-8 Tuning of the Y Axis

Make the tuning of the Y axis in the same way.

1 Select **Motor2** in the **Select Motor** area of the Tuning screen.

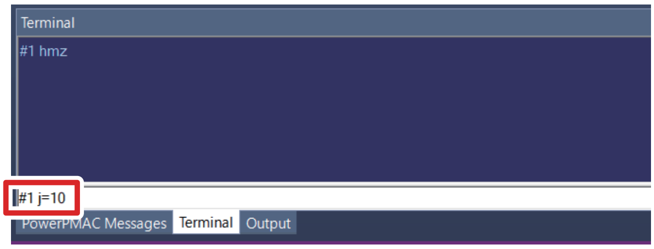


2 Perform the current loop turning, open-loop testing, position loop auto-tuning, and position loop interactive tuning procedures.

3-4-9 Confirmation of the Tuning Results

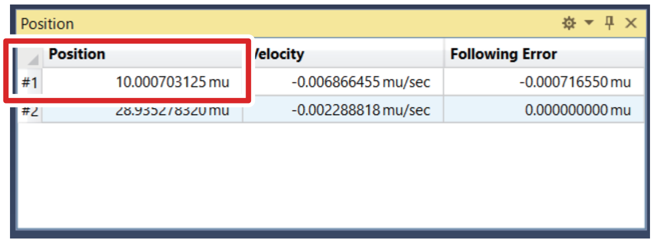
Confirm that the tuning results are correct, and reflect them in the project file.

1 Type the **#1 hmz** command in the Power PMAC IDE Terminal.
After that, type the **#1 j=10** command.



```
Terminal
#1 hmz

#1 j=10
```

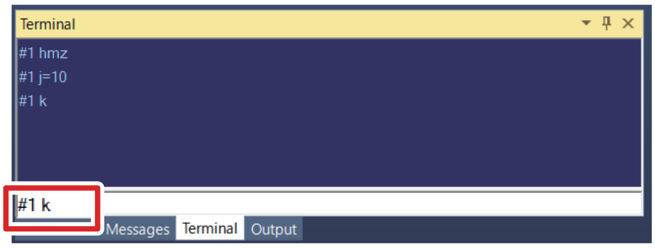
2 Confirm that the X axis is moving. Also, confirm that the value of **Position #1** is around 10.0 in Position window.



Position

	Position	Velocity	Following Error
#1	10.000703125 mu	-0.006866455 mu/sec	-0.000716550 mu
#2	28.935278320 mu	-0.002288818 mu/sec	0.000000000 mu

3 Type the **#1 k** command in the Power PMAC IDE Terminal to stop the motor.



```
Terminal
#1 hmz
#1 j=10
#1 k

#1 k
```

4 Perform steps 1 to 3 for Motor[2] as well.

5 Open the **MotorControl.pmh** file under **PMAC Script Language – Global Includes** in the Solution Explorer.

6 Add the gain settings obtained from the tuning to the MotorControl.pmh file.

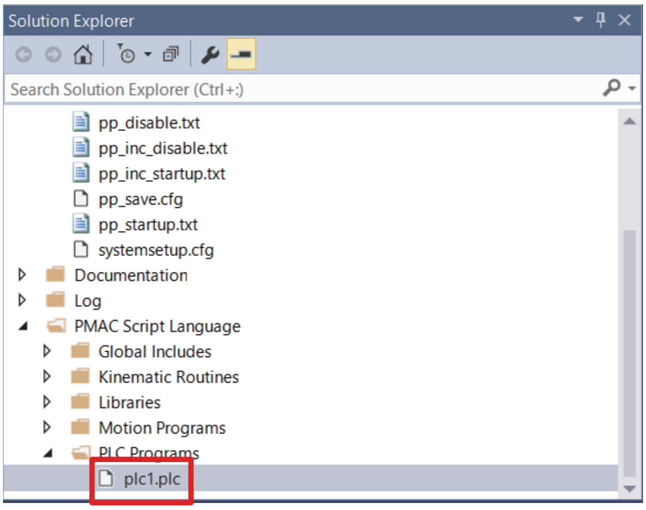
```
Motor[1].IiGain=***
Motor[2].IiGain=***
Motor[1].IpfGain=***
Motor[2].IpfGain=***
Motor[1].Servo.Kp=***
Motor[2].Servo.Kp=***
Motor[1].Servo.Kvfb=***
Motor[2].Servo.Kvfb=***
Motor[1].Servo.Ki=***
Motor[2].Servo.Ki=***
Motor[1].Servo.Kaff=***
Motor[2].Servo.Kaff=***
Motor[1].Servo.Kvff=***
Motor[1].Servo.Kfff=***
```

7 Open the **pp_startup.txt** file under **Configuration** in the Solution Explorer.

8 Write the phase search commands shown on the right.

```
enable plc PhaseSearch
```

9 Open the **plc1.plc** file under **PMAC Script Language** – **PLC Programs** in the Solution Explorer.



10 Add the program shown on the right to the plc1.plc file.

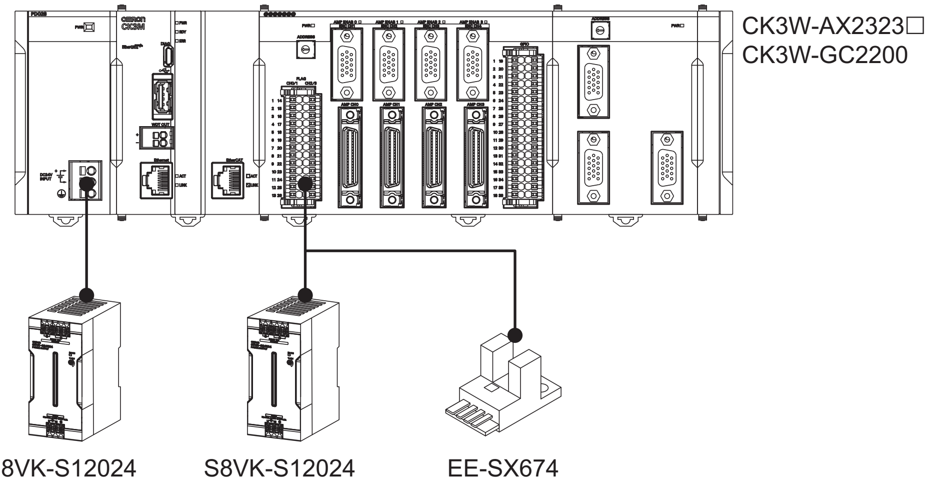
```
open plc PhaseSearch

 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
```

11	Right click on the Solution Explorer project name at the upper right of the Power PMAC IDE screen, select **Build and Download All Programs**, and execute Build and Download. • You can download the gain settings to the PMAC as a program by writing them in the MotorControl.pmh file as shown in steps 5 and 6. • With the phase search commands written in pp_startup.txt as shown in steps 7 and 8, phase search is performed automatically after the power supply is turned ON or the CPU Unit is reset to enable Motor[1] and Motor[2].

3-5 Setup of Overtravel Limit Switches and Home Positions

3-5-1 Wiring of Devices

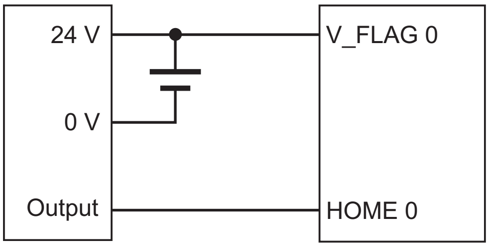
The figures below show how to wire the CK3M to various sensors.



Connect the EE-SX674 to the following FLAG terminals of the CK3W-AX2323N.

- PLIM0: X-axis positive overtravel limit switch
- NLIM0: X-axis negative overtravel limit switch
- PLIM1: Y-axis positive overtravel limit switch
- NLIM1: Y-axis negative overtravel limit switch

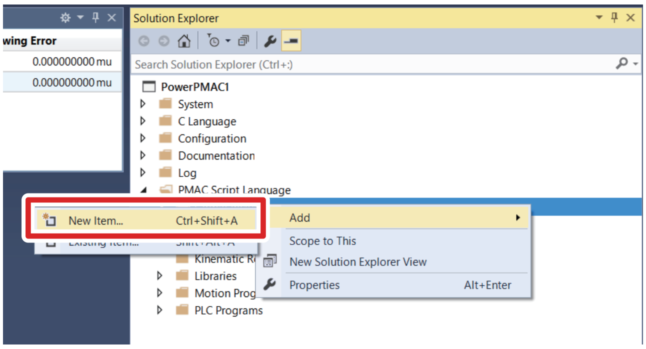
The following shows an example of wiring to the HOME0 terminal.



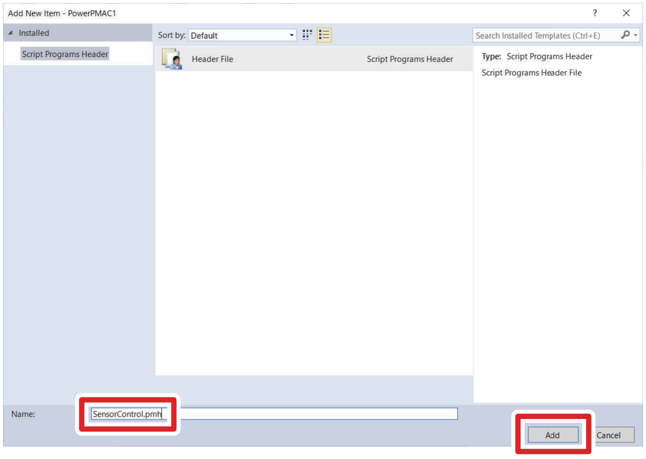
3-5-2 Programming

Set up the overtravel limit switches.

1 In the Solution Explorer, right-click **Global Includes** under **PMAC Script Language** and select **Add** – **New Item...**.



2 In the **Name** box, type *SensorControl.pmh* and click the **Add** button.


Add New Item - PowerPMAC1
Installed
Script Programs Header
Sort by: Default
Header File
Script Programs Header
Search Installed Templates (Ctrl+E)
Type: Script Programs Header
Script Programs Header File
Add
Cancel

Name: SensorControl.pmh

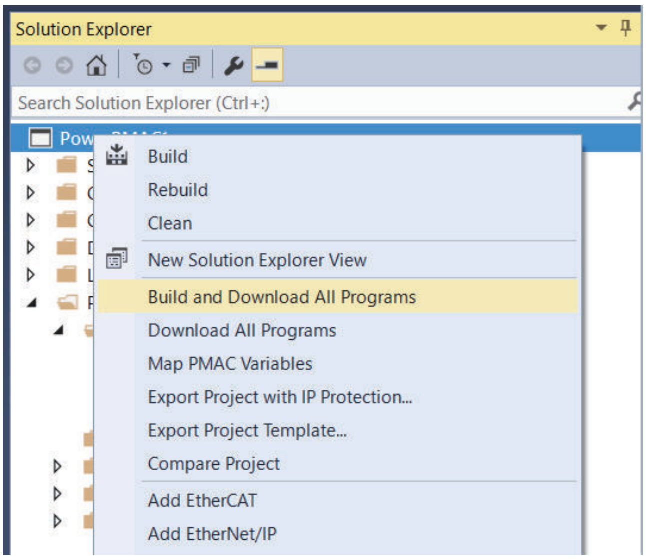
3 Open the **SensorControl.pmh** file under **PMAC Script Language** – **Global Includes** in the Solution Explorer.

4 Write the text shown on the right to the SensorControl.pmh file.

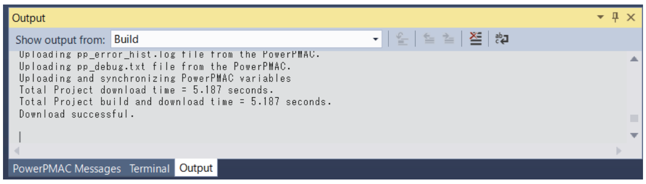
- For details on the annotated settings, refer to *Section 4 How to Customize Various Settings* on page 4-1.

```
Sys.WpKey=$AAAAAAAA

//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
```

5 Right click on the Solution Explorer project name at the upper right of the Power PMAC IDE screen, select **Build and Download All Programs**, and execute Build and Download.


Solution Explorer
Search Solution Explorer (Ctrl+;)
Pow
Build
Rebuild
Clean
New Solution Explorer View
Build and Download All Programs
Download All Programs
Map PMAC Variables
Export Project with IP Protection...
Export Project Template...
Compare Project
Add EtherCAT
Add EtherNet/IP

6 Make sure that there are no errors in the Output window.

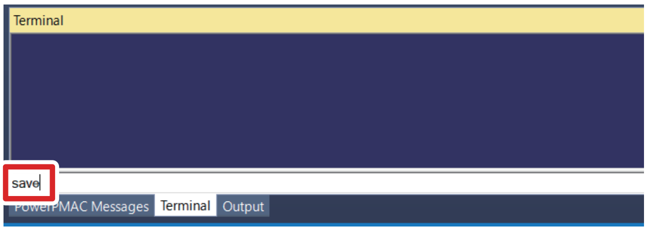


Output

Show output from: Build

```
Uploading pp_error_hist.log file from the PowerPMAC.
Uploading pp_debug.txt file from the PowerPMAC.
Uploading and synchronizing PowerPMAC variables
Total Project download time = 5.187 seconds.
Total Project build and download time = 5.187 seconds.
Download successful.
```

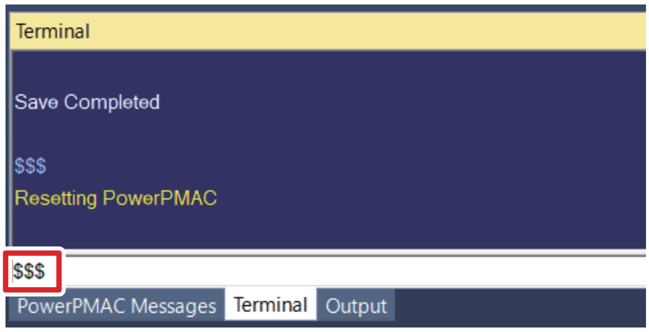
7 Type the **save** command in the Power PMAC IDE Terminal.



Terminal

```
save
```

8 Type the **$$$** command in the Power PMAC IDE Terminal.



Terminal

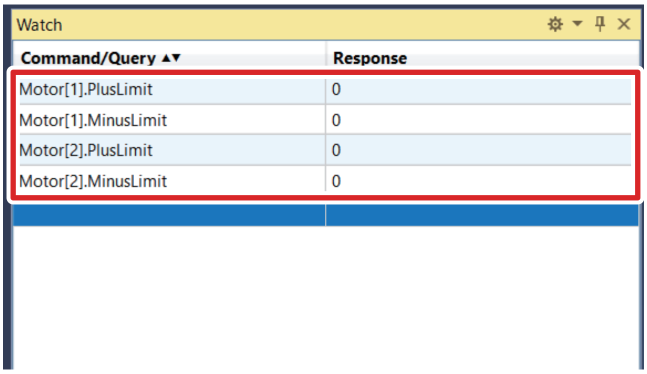
```
Save Completed

$$$
Resetting PowerPMAC

$$$
```

3-5-3 Confirmation of the Settings (Overtravel Limit Switches)

Confirm that the settings of overtravel limit switches operate correctly.

1 Confirm that the following variables are all 0.

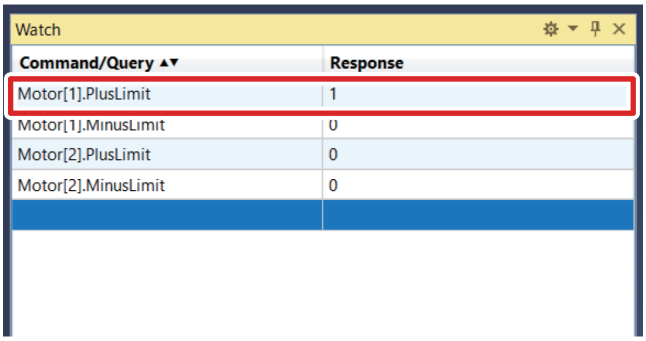
- *Motor[1].PlusLimit*
- *Motor[1].MinusLimit*
- *Motor[2].PlusLimit*
- *Motor[2].MinusLimit*



Watch

Command/Query	Response
Motor[1].PlusLimit	0
Motor[1].MinusLimit	0
Motor[2].PlusLimit	0
Motor[2].MinusLimit	0

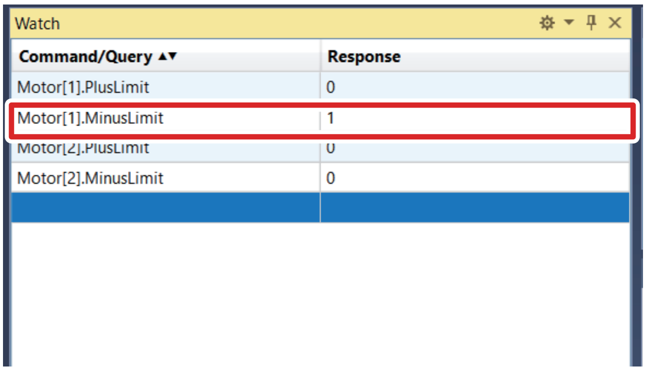
2 Confirm that *Motor[1].PlusLimit* of the X-axis positive overtravel limit switch is 1.



Watch

Command/Query	Response
Motor[1].PlusLimit	1
Motor[1].MinusLimit	0
Motor[2].PlusLimit	0
Motor[2].MinusLimit	0

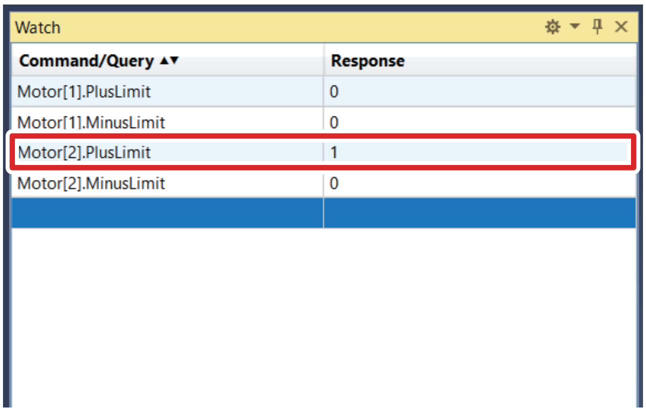
3 Confirm that *Motor[1].MinusLimit* of the X-axis negative overtravel limit switch is 1.



Watch

Command/Query	Response
Motor[1].PlusLimit	0
Motor[1].MinusLimit	1
Motor[2].PlusLimit	0
Motor[2].MinusLimit	0

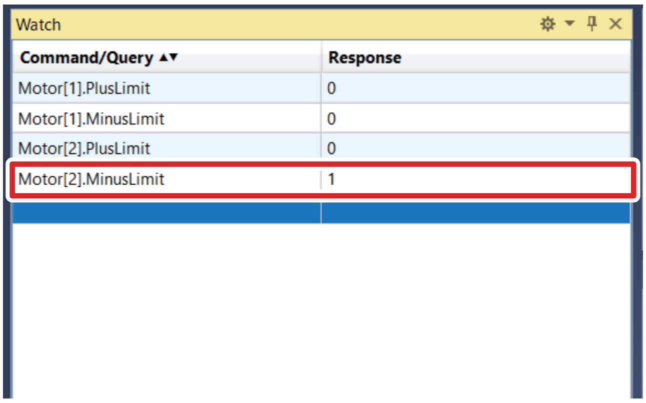
4 Confirm that *Motor[2].PlusLimit* of the Y-axis positive overtravel limit switch is 1.



Watch

Command/Query	Response
Motor[1].PlusLimit	0
Motor[1].MinusLimit	0
Motor[2].PlusLimit	1
Motor[2].MinusLimit	0

5 Confirm that *Motor[2].MinusLimit* of the Y-axis negative overtravel limit switch is 1.



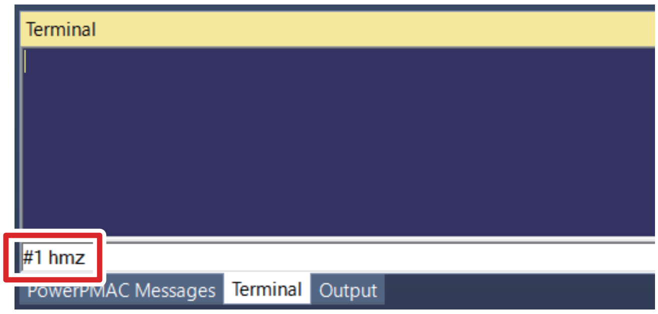
Watch

Command/Query	Response
Motor[1].PlusLimit	0
Motor[1].MinusLimit	0
Motor[2].PlusLimit	0
Motor[2].MinusLimit	1

3-5-4 Setup of Home Positions

Set up the home positions.

1 Move the X axis to the home position and type the **#1 hmz** command in the Power PMAC IDE Terminal.

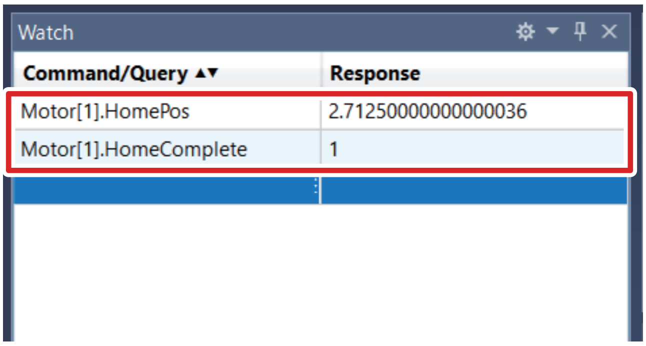


Terminal

```
#1 hmz
```

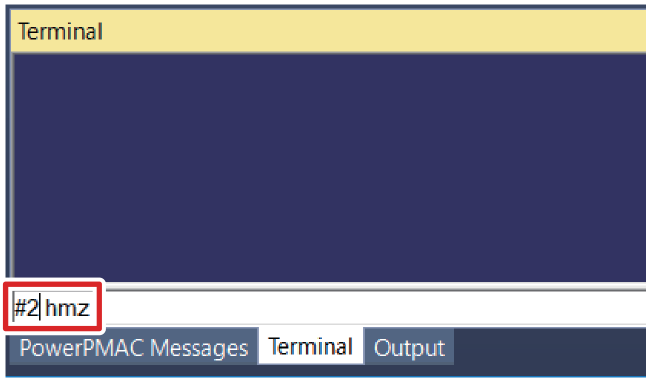
2 Confirm that the following variables are correctly reflected.

- *Motor[1].HomePos*: Home trigger position
- *Motor[1].HomeComplete*: 1



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

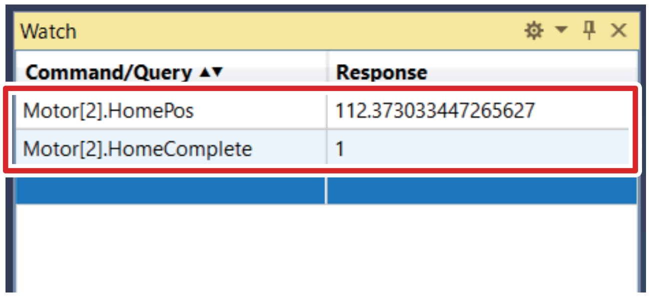
3 Move the Y axis to the home position and type the **#2 hmz** command in the Power PMAC IDE Terminal.



```
#2 hmz
```

4 Confirm that the following variables are correctly reflected.

- *Motor[2].HomePos*: Home trigger position
- *Motor[2].HomeComplete*: 1

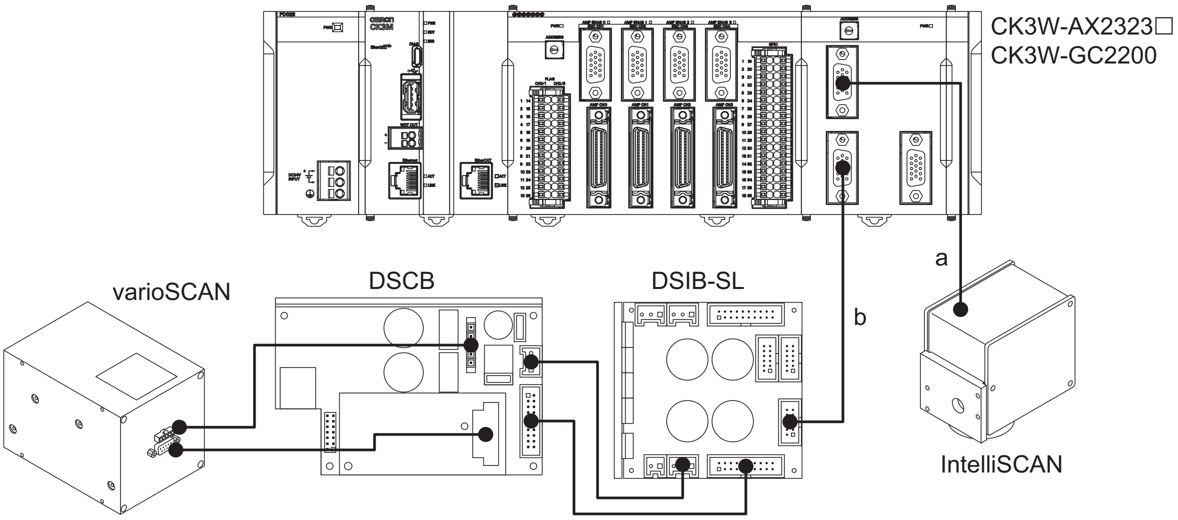


Command/Query ▲▼	Response
Motor[2].HomePos	112.373033447265627
Motor[2].HomeComplete	1

3-6 Setup of the Galvo Scanner

3-6-1 Wiring of Devices

The figures below show how to wire the CK3M to the Galvo Scanner.



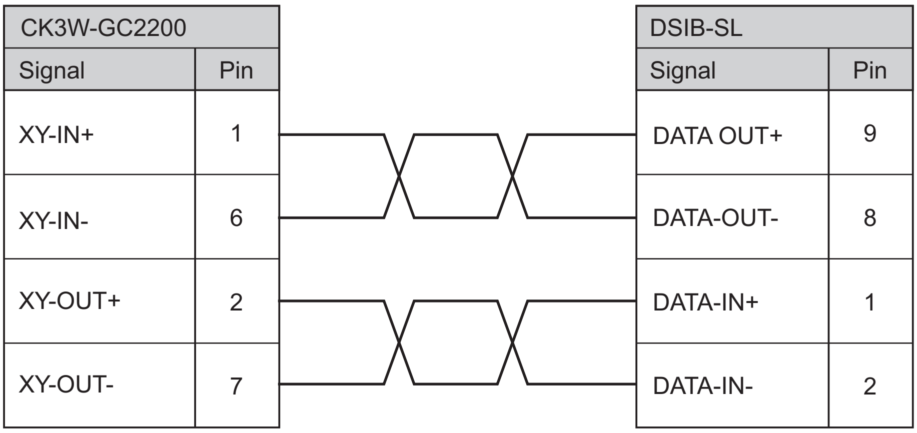
Connect the sections a and b in the above figures as follows.

a. Connection between the Controller and the Galvo Scanner
Connect the XY connector of the CK3W-GC2200 to the SL2-100 connector of the intelliSCAN with the dedicated cable CK3W-CAG03A.

CK3W-GC2200	
Signal	Pin
XY-IN+	1
XY-IN-	6
XY-OUT+	2
XY-OUT-	7

IntelliSCAN	
Signal	Pin
DATA OUT+	5
DATA-OUT-	9
DATA-IN+	1
DATA-IN-	6

b. Connection between the Controller and the dynamic focusing unit
Connect the Z connector of the CK3W-GC2200 to the DIGITAL IN connector of the DSIB-SL with the dedicated cable CK3W-CAG03A.



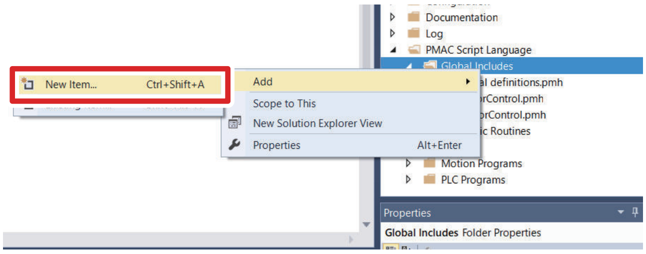
CK3W-GC2200	
Signal	Pin
XY-IN+	1
XY-IN-	6
XY-OUT+	2
XY-OUT-	7

DSIB-SL	
Signal	Pin
DATA OUT+	9
DATA-OUT-	8
DATA-IN+	1
DATA-IN-	2

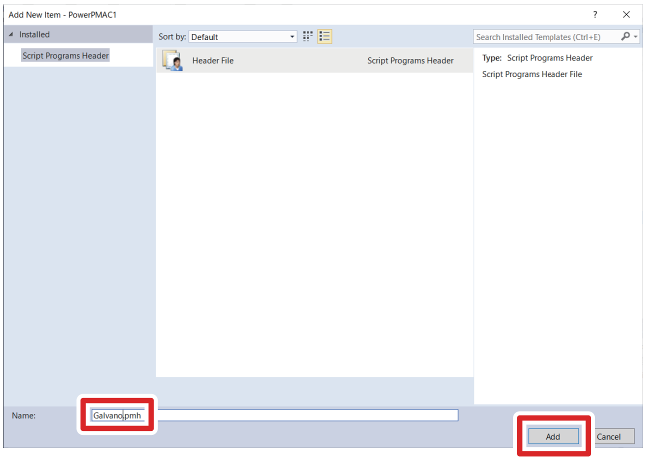
3-6-2 Programming

Program the settings of the Galvo Scanner.

1 In the Solution Explorer, right-click **Global Includes** under **PMAC Script Language** and select **Add** – **New Item...**.



2 In the **Name** box, type *Galvano.pmh* and click the **Add** button.


Add New Item - PowerPMAC1
Installed
Script Programs Header
Sort by: Default
Search Installed Templates (Ctrl+E)
Header File
Script Programs Header
Type: Script Programs Header
Script Programs Header File
Add
Cancel

Name: Galvano.pmh

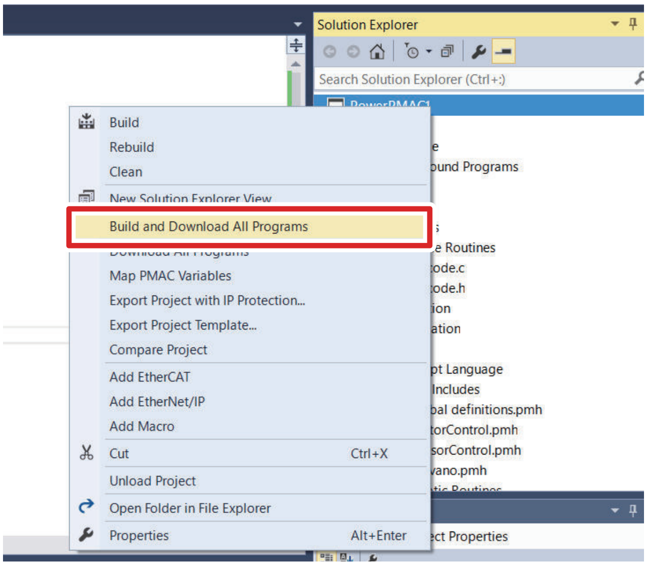
3 Open the **Galvano.pmh** file under **PMAC Script Language** – **Global Includes** in the Solution Explorer.

4 Write the text shown on the right to the Galvano.pmh file.

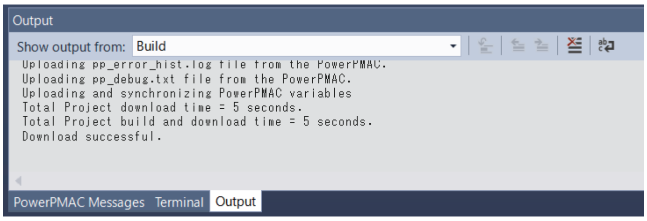
- For details on the annotated settings, refer to *Section 4 How to Customize Various Settings* on page 4-1.

```
Sys.WpKey=$AAAAAAAA

//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 Right click on the Solution Explorer project name at the upper right of the Power PMAC IDE screen, select **Build and Download All Programs**, and execute Build and Download.



6 Make sure that there are no errors in the Output window.

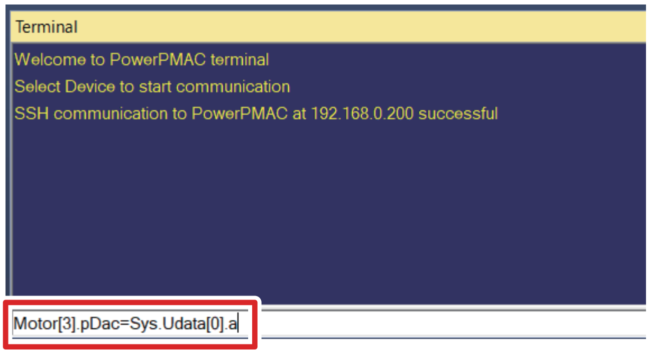


```
Uploading pp_error_hist.log file from the PowerPMAC.
Uploading pp_debug.txt file from the PowerPMAC.
Uploading and synchronizing PowerPMAC variables
Total Project download time = 5 seconds.
Total Project build and download time = 5 seconds.
Download successful.
```

3-6-3 Setup of Control Commands

Set the feedback signal of the Galvo Scanner to the current position.

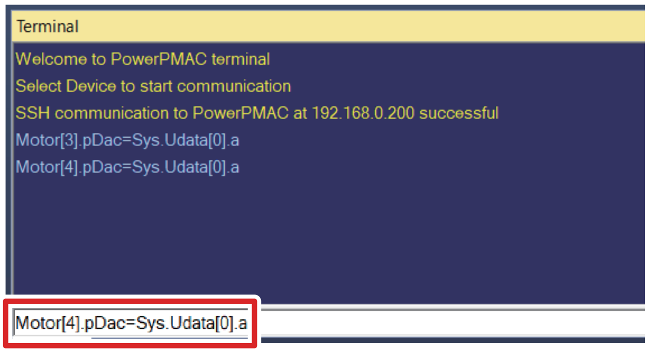
1 Type the **Motor[3].pDac=Sys.Udata[0].a** command in the Power PMAC IDE Terminal.



```
Welcome to PowerPMAC terminal
Select Device to start communication
SSH communication to PowerPMAC at 192.168.0.200 successful
```

```
Motor[3].pDac=Sys.Udata[0].a
```

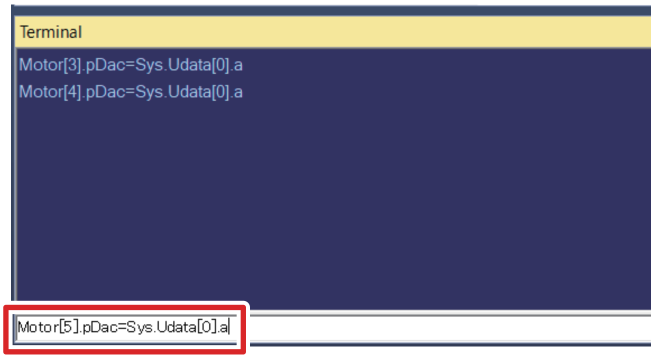
2 Type the **Motor[4].pDac=Sys.Udata[0].a** command in the Power PMAC IDE Terminal.



```
Welcome to PowerPMAC terminal
Select Device to start communication
SSH communication to PowerPMAC at 192.168.0.200 successful
Motor[3].pDac=Sys.Udata[0].a
Motor[4].pDac=Sys.Udata[0].a
```

```
Motor[4].pDac=Sys.Udata[0].a
```

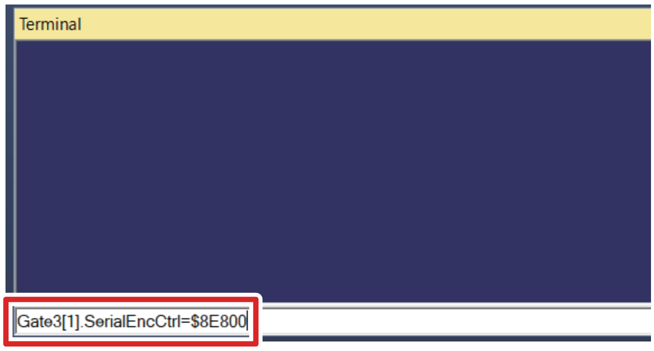
3 Type the **Motor[5].pDac=Sys.Udata[0].a** command in the Power PMAC IDE Terminal.



```
Motor[3].pDac=Sys.Udata[0].a
Motor[4].pDac=Sys.Udata[0].a
```

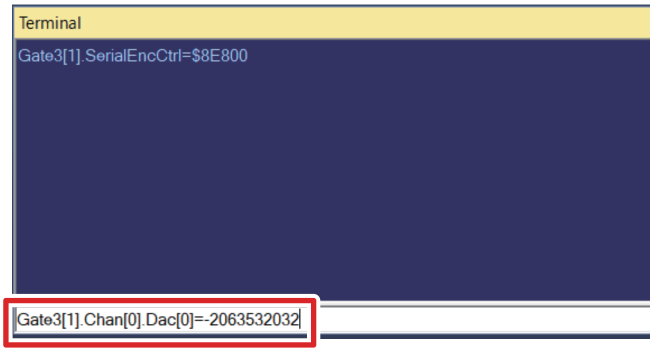
```
Motor[5].pDac=Sys.Udata[0].a
```

4 Type the **Gate3[1].SerialEncCtrl=$8E800** command in the Power PMAC IDE Terminal.



```
Gate3[1].SerialEncCtrl=$8E800
```

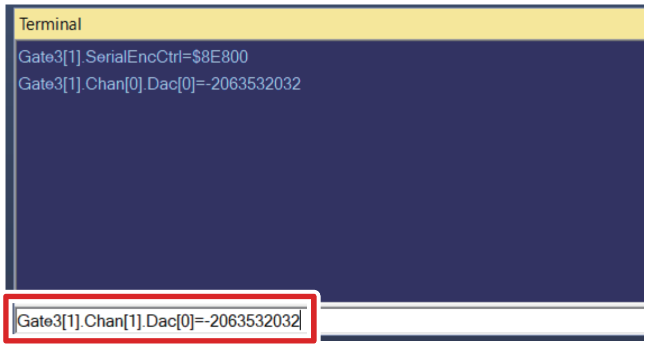
5 Type the **Gate3[1].Chan[0].Dac[0]=-2063532032** command in the Power PMAC IDE Terminal.



```
Gate3[1].SerialEncCtrl=$8E800
```

```
Gate3[1].Chan[0].Dac[0]=-2063532032
```

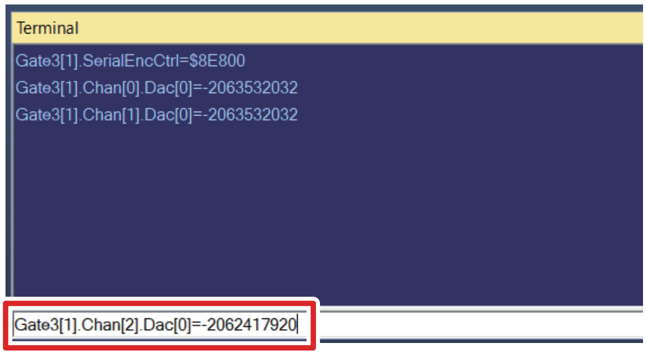
6 Type the **Gate3[1].Chan[1].Dac[0]=-2063532032** command in the Power PMAC IDE Terminal.



```
Gate3[1].SerialEncCtrl=$8E800
Gate3[1].Chan[0].Dac[0]=-2063532032
```

```
Gate3[1].Chan[1].Dac[0]=-2063532032
```

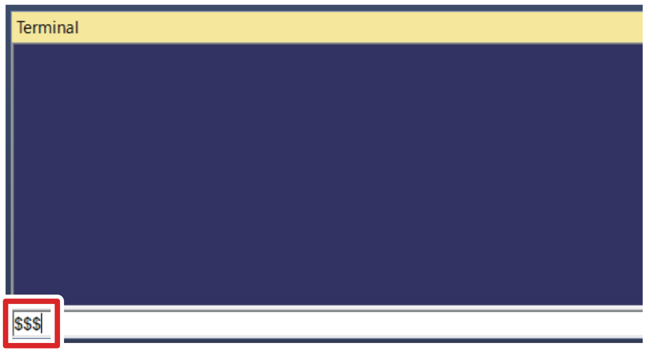
7 Type the **Gate3[1].Chan[2].Dac[0]=-2062417920** command in the Power PMAC IDE Terminal.



```
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
```

8 Type the **$$$** command in the Power PMAC IDE Terminal.

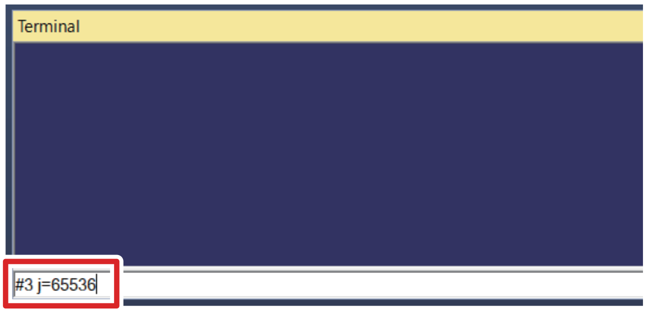



```
$$$
```

3-6-4 Confirmation of the Settings

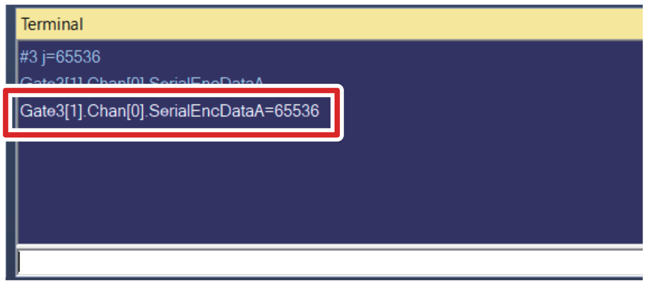
Confirm that the settings of the Galvo Scanner are correct.

1 Type the **#3 j=65536** command in the Power PMAC IDE Terminal.



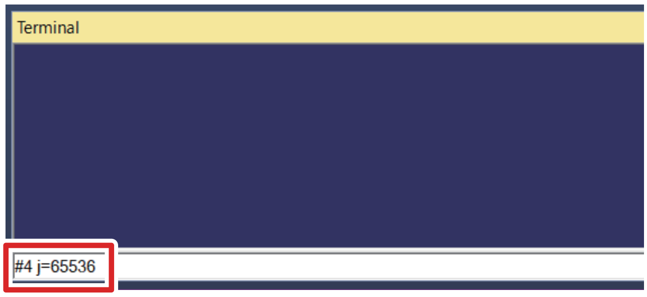

```
#3 j=65536
```

2 Confirm that Gate3[0].Chan[0].SerialEncDataA=65536 or so is displayed.



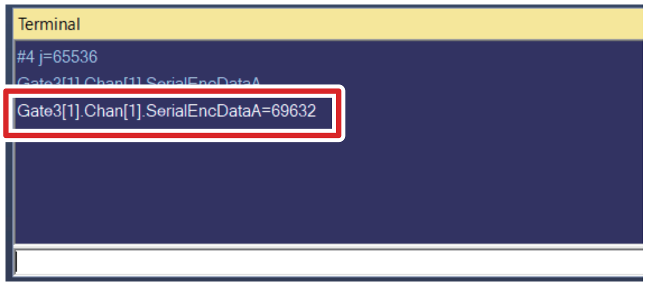

```
#3 j=65536
Gate3[1].Chan[0].SerialEncDataA
Gate3[1].Chan[0].SerialEncDataA=65536
```

3 Type the **#4 j=65536** command in the Power PMAC IDE Terminal.



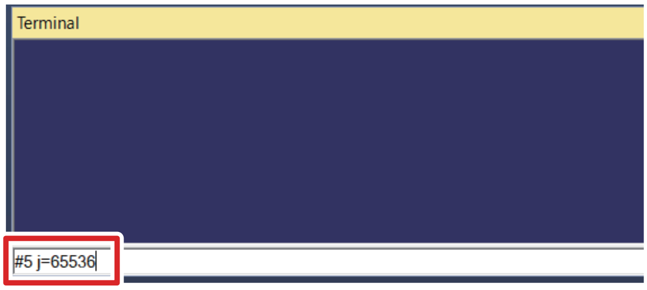
```
#4 j=65536
```

4 Confirm that Gate3[1].Chan[1].SerialEncDataA=65536 or so is displayed.



```
#4 j=65536
Gate3[1].Chan[1].SerialEncDataA
Gate3[1].Chan[1].SerialEncDataA=69632
```

5 Type the **#5 j=65536** command in the Power PMAC IDE Terminal.



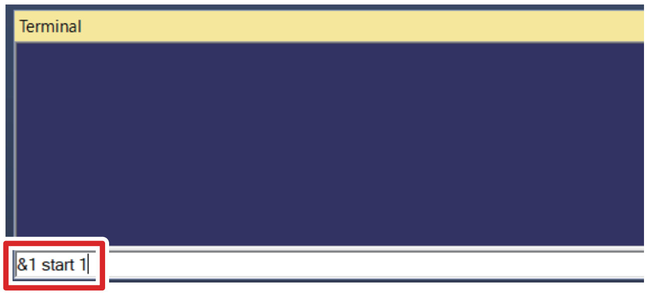
```
#5 j=65536
```

6 Confirm that Gate3[0].Chan[2].SerialEncDataA=65536 or so is displayed.

3-6-5 Verification of the MOTF on the Actual Machine

Confirm that the settings of the MOTF are operating correctly on the actual machine.

1 Type the **&1 start 1** command in the Power PMAC IDE Terminal.



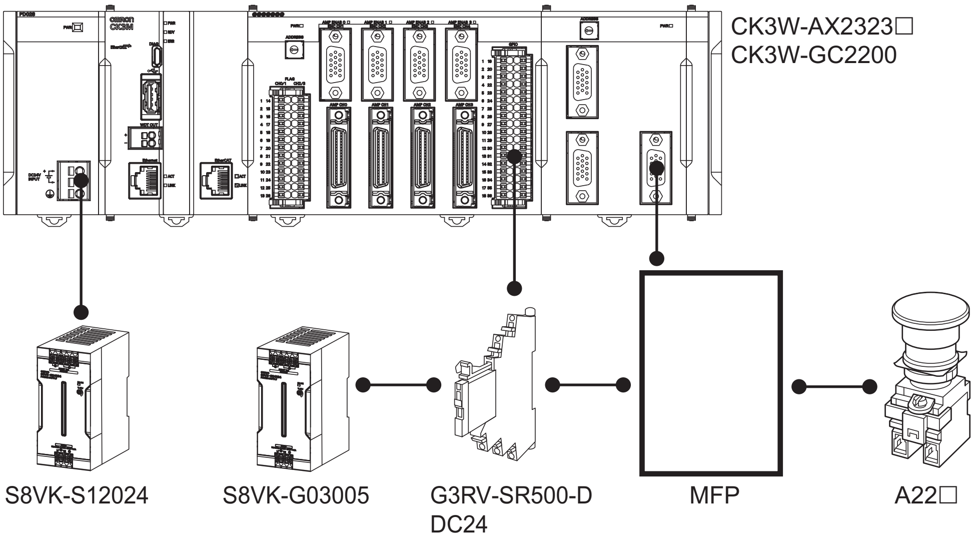
```
&1 start 1
```

2 Confirm that the linear motors and Galvo Scanner for the X and Y axes operate in synchronization.

3-7 Setup of the Laser Oscillator

3-7-1 Wiring of Devices

The figures below show how to wire the CK3M to the laser oscillator.



Connect the GPIO terminal of the CK3W-AX2323N to the MFP as shown below.

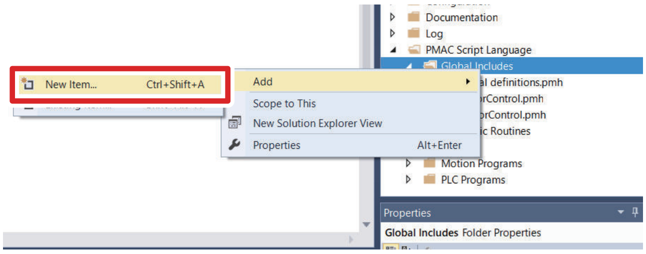
- OUT 0: Pin 5 of the MFP
- OUT 1: Pin 6 of the MFP
- OUT 2: Pin 7 of the MFP
- OUT 3: Pin 8 of the MFP
- OUT 4: Pin 9 of the MFP
- OUT 5: Pin 18 of the MFP
- OUT 6: Pin 22 of the MFP

Connect the LASER terminal of the CK3W-GC2200 to the MFP as shown below.

- OUT 0: Pin 20 of the MFP
- OUT 1: Pin 21 of the MFP

3-7-2 Guide Laser Output

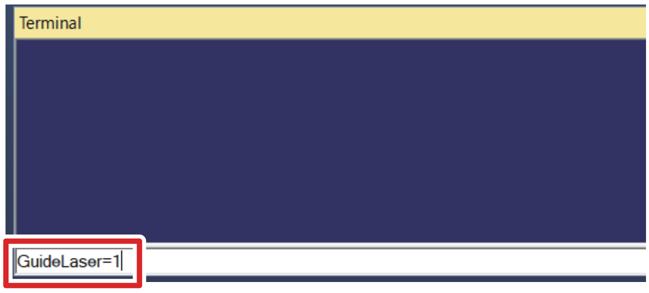
Confirm that the guide laser light is output correctly.

1	In the Solution Explorer, right-click **Global Includes** under **PMAC Script Language** and select **Add** – **New Item...**.  Documentation Log PMAC Script Language Global Includes al definitions.pmh rControl.pmh rControl.pmh ic Routines Motion Programs PLC Programs New Item... Ctrl+Shift+A Add Scope to This New Solution Explorer View Properties Alt+Enter Properties Global Includes Folder Properties
2	In the **Name** box, type *LaserDefinitions.pmh* and click the **Add** button.

3 In the Solution Explorer, open the **LaserControl.pmh** file under **PMAC Script Language** – **Global Includes** and add the text shown on the right.

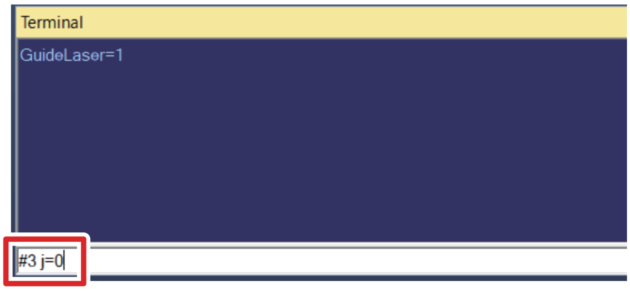
```
ptr LaserPower->Gate3[0].GpioData[0].16.4
ptr LaserLatch->Gate3[0].GpioData[0].20.1
ptr MasterOscillator->Gate3[0].GpioData[0].21
.1
ptr GuideLaser->Gate3[0].GpioData[0].22.1
```

4 Type the **GuideLaser=1** command in the Power PMAC IDE Terminal and confirm that guide laser light is output.

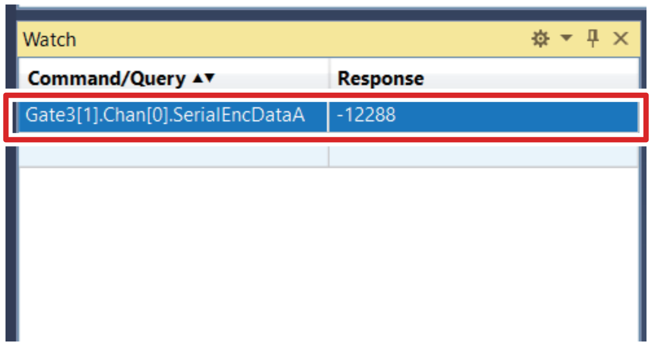


```
GuideLaser=1
```

5 Type the **#3 j=0** command in the Power PMAC IDE Terminal and confirm that Gate3[1].Chan[0].SerialEncDataA=0 or so is displayed.

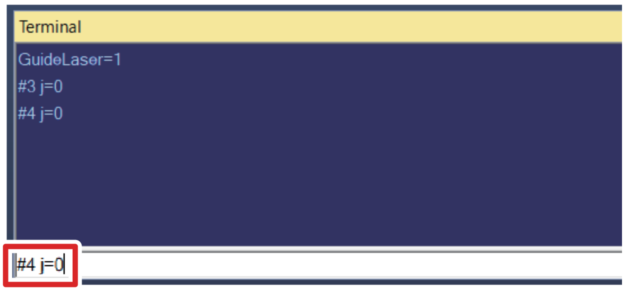


```
GuideLaser=1
#3 j=0
```



Command/Query ▲▼	Response
Gate3[1].Chan[0].SerialEncDataA	-12288

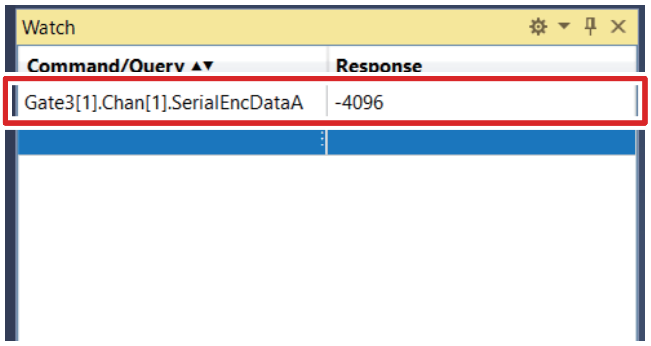
6 Type the **#4 j=0** command in the Power PMAC IDE Terminal and confirm that Gate3[1].Chan[1].SerialEncDataA=0 is displayed.



Terminal

```
GuideLaser=1
#3 j=0
#4 j=0
```

```
#4 j=0
```



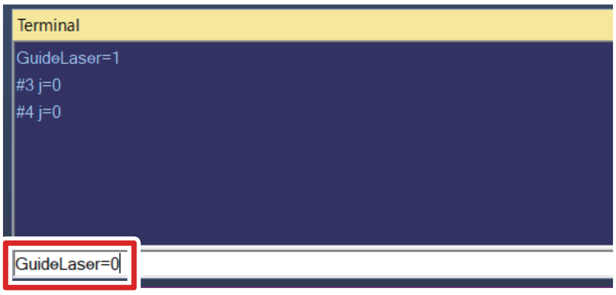
Watch

Command/Query	Response
Gate3[1].Chan[1].SerialEncDataA	-4096

7 Confirm that the guide laser light is output to the center of the XY table.

8 Type the **#5 j={Position}** command in the Power PMAC IDE Terminal and confirm that the spot diameter of guide laser light is appropriate.

9 Type the **GuideLaser=0** command in the Power PMAC IDE Terminal and confirm that guide laser light is not output.



Terminal

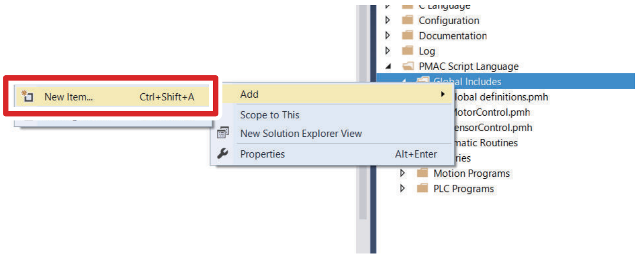
```
GuideLaser=1
#3 j=0
#4 j=0
```

```
GuideLaser=0
```

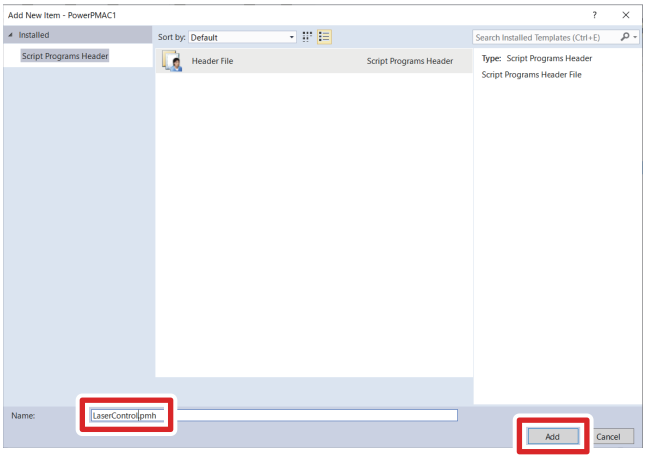
3-7-3 Programming

Program the settings of the laser oscillator.

1 In the Solution Explorer, right-click **Global Includes** under **PMAC Script Language** and select **Add** – **New Item...**.



2 In the **Name** box, type *LaserControl.pmh* and click the **Add** button.



Name: LaserControl.pmh

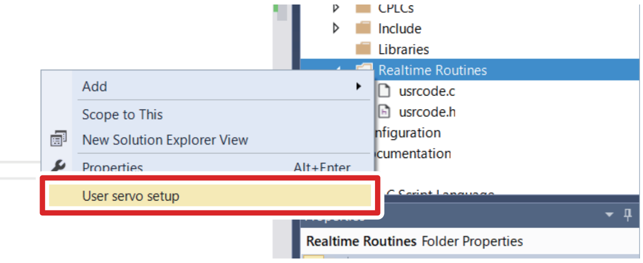
Add New Item - PowerPMAC1
Installed
Script Programs Header
Sort by: Default
Search Installed Templates (Ctrl+E)
Header File
Script Programs Header
Type: Script Programs Header
Script Programs Header File
Add
Cancel

3 In the Solution Explorer, open the **LaserControl.pmh** file under **PMAC Script Language** – **Global Includes** and add the text shown on the right.

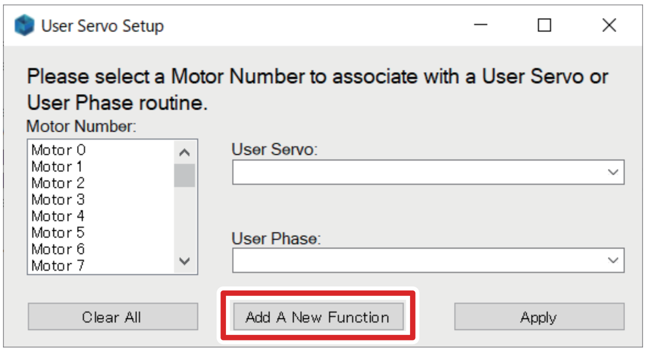
- For details on the annotated settings, refer to *Section 4 How to Customize Various Settings* on page 4-1.

```
Sys.WpKey=$AAAAAAAA

//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
```

4 In the Solution Explorer, right-click the **Realtime Routines** file under **C Language** and select **User servo setup**.

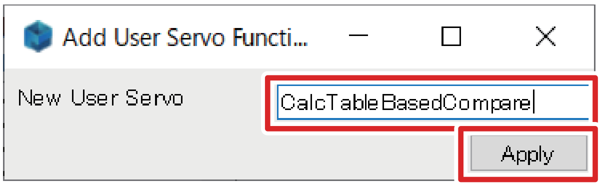


5 Click the **Add a New Function** button.



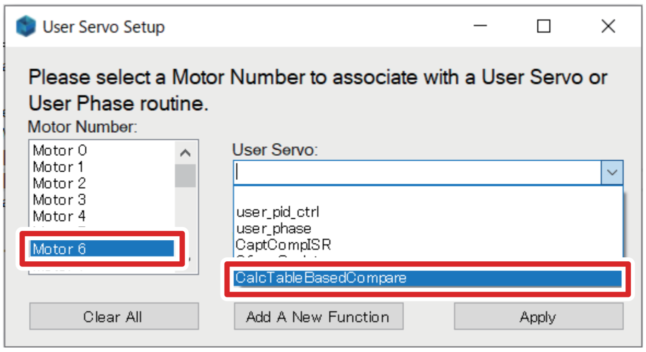
Please select a Motor Number to associate with a User Servo or User Phase routine.

6 Type *CalcTableBasedCompare* and click the **Apply** button.



New User Servo: CalcTableBasedCompare

7 Select **Motor 6**, and then select **CalcTableBasedCompare** from the **User Servo** list.

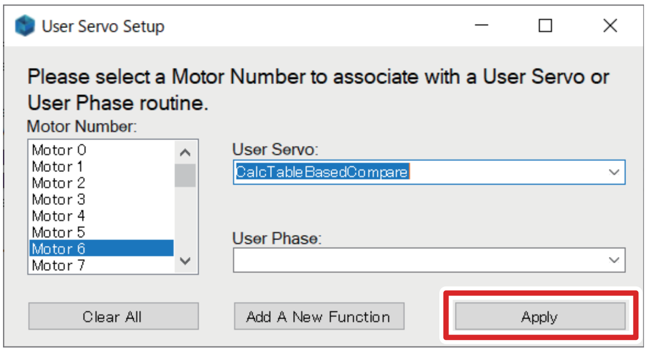


Please select a Motor Number to associate with a User Servo or User Phase routine.

Motor Number: Motor 6

User Servo: CalcTableBasedCompare

8 Click the **Apply** button.



Please select a Motor Number to associate with a User Servo or User Phase routine.

Motor Number: Motor 6

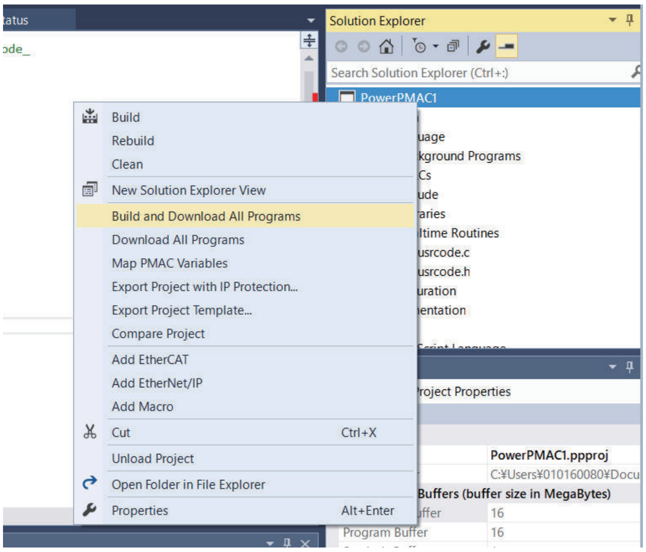
User Servo: CalcTableBasedCompare

User Phase:

9 In the Solution Explorer, click the **usercode.c** file under **C Language – Realtime Routines** and add the functions shown on the right.

```
static double X_old=0;
static double Y_old=0;
static double SumDistance=0;

double CalcTableBasedCompare(struct MotorData*Mpt
r)
{
 volatile GateArray3 *MySecondGate3IC;
 MySecondGate3IC=GetGate3MemPtr(1);

 double X_now;
 X_now=pshm->Motor[1].DesPos+pshm->Motor[3].DesPo
s;
 double Y_now;
 Y_now=pshm->Motor[2].DesPos+pshm->Motor[4].DesPo
s;
 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;
}
```

10 In the Solution Explorer, right-click **Global Includes** under **PMAC Script Language** and select **Add** – **New Item...**. In the **Name** box, type *TCRConfigurations.pmh* and click the **Add** button.

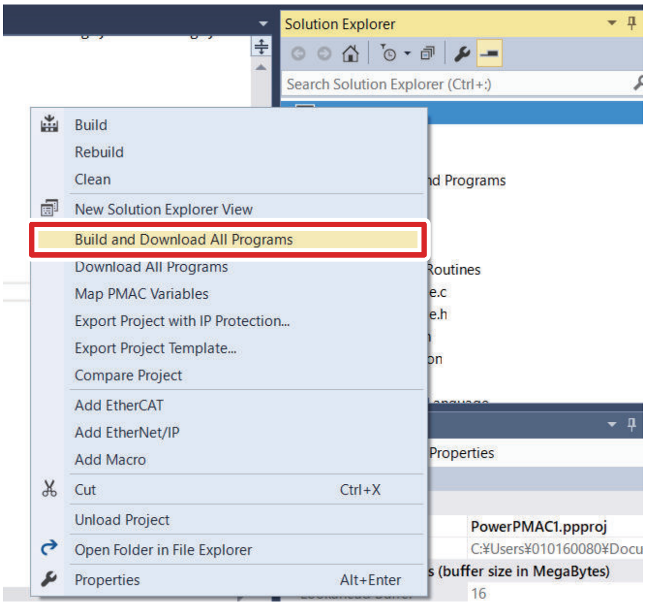


11 Write the text shown on the right to the TCRConfigurations.pmh file.

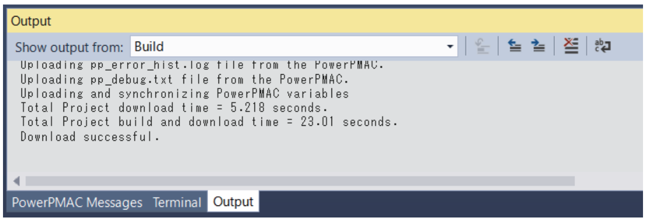
- For details on the annotated settings, refer to *Section 4 How to Customize Various Settings* on page 4-1.

```
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
```

12 Right click on the Solution Explorer project name at the upper right of the Power PMAC IDE screen, select **Build and Download All Programs**, and execute Build and Download.



13 Make sure that there are no errors in the Output window.



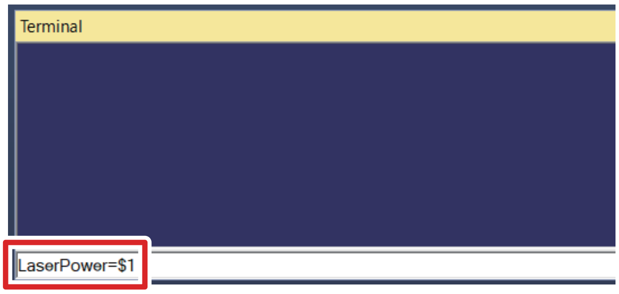
```
Uploading pp_error_hist.log file from the PowerPMAC.
Uploading pp_debug.txt file from the PowerPMAC.
Uploading and synchronizing PowerPMAC variables
Total Project download time = 5.218 seconds.
Total Project build and download time = 23.01 seconds.
Download successful.
```

3-7-4 Setup of the Laser Oscillator

Set up the power of the laser oscillator and make preliminary preparations for laser output.

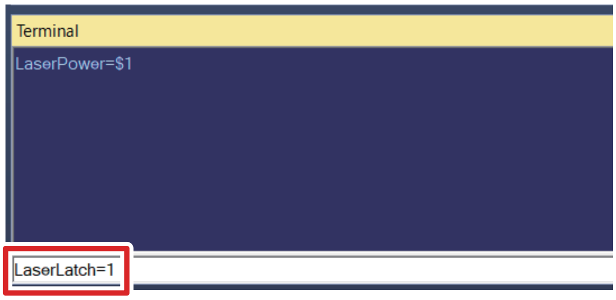
1 Confirm that the emergency stop push-button switch is OFF.

2 Type the **LaserPower=$1** command in the Terminal.



```
LaserPower=$1
```

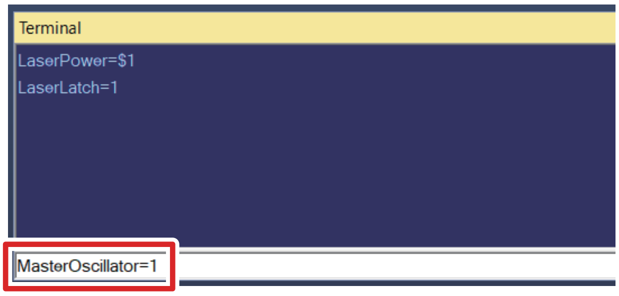
3 Type the **LaserLatch=1** command in the Terminal.



```
LaserPower=$1
```

```
LaserLatch=1
```

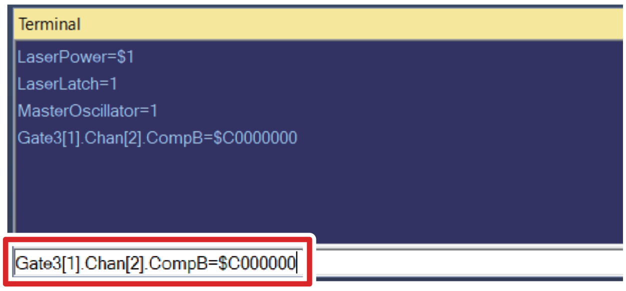
4 Type the **MasterOscillator=1** command in the Terminal.



```
LaserPower=$1
LaserLatch=1
```

```
MasterOscillator=1
```

5 Type the **Gate3[1].Chan[2].CompB= $C000000** command in the Terminal.

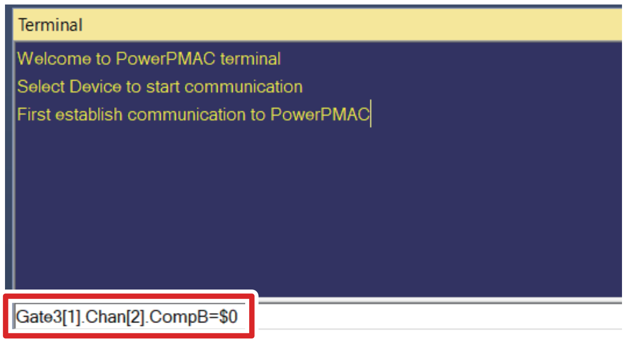


```
LaserPower=$1
LaserLatch=1
MasterOscillator=1
Gate3[1].Chan[2].CompB=$C0000000
```

```
Gate3[1].Chan[2].CompB=$C000000
```

6 Confirm that the laser light is output.

7 Type the **Gate3[1].Chan[2].CompB=$0** command in the Terminal.



```
Welcome to PowerPMAC terminal
Select Device to start communication
First establish communication to PowerPMAC
```

```
Gate3[1].Chan[2].CompB=$0
```

3-7 Setup of the Laser Oscillator

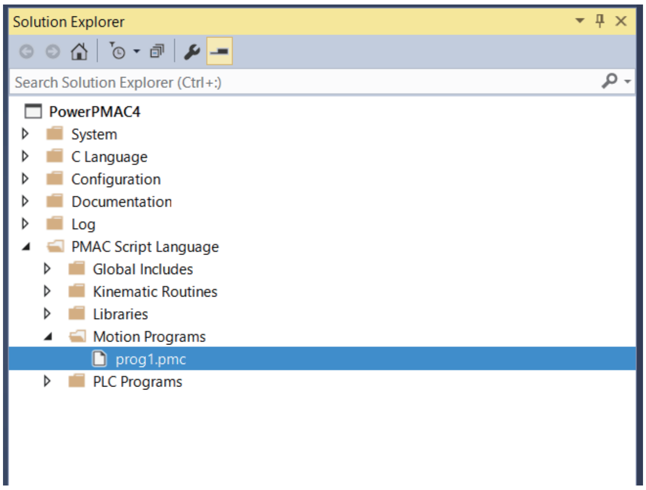
3

3-7-4 Setup of the Laser Oscillator

3-7-5 Confirmation of the Settings

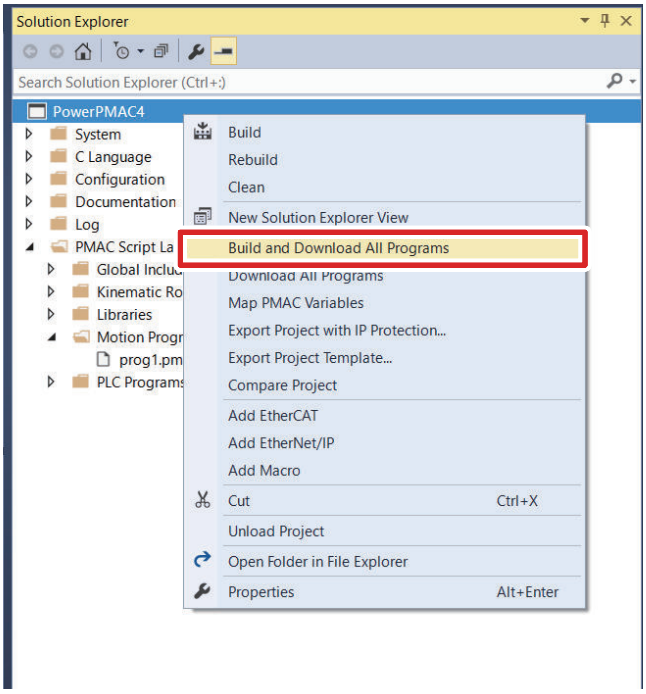
Confirm that the settings in this document are made correctly, by checking whether the processing is actually available.

1 Open the **prog1.pmc** file under **PMAC Script Language** – **Motion Programs** in the Solution Explorer.



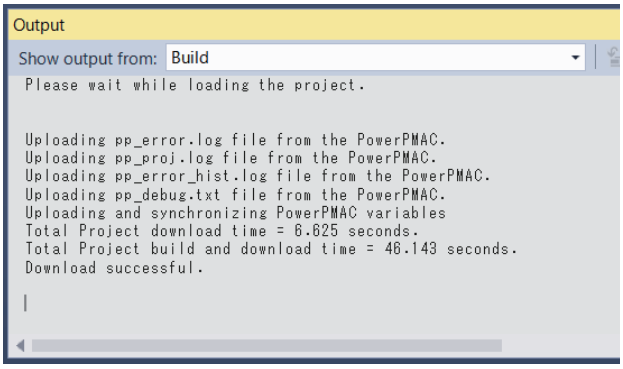
2 Write the program to use for processing to the prog1.pmc file.

3 Right click on the Solution Explorer project name at the upper right of the Power PMAC IDE screen, select **Build and Download All Programs**, and execute Build and Download.

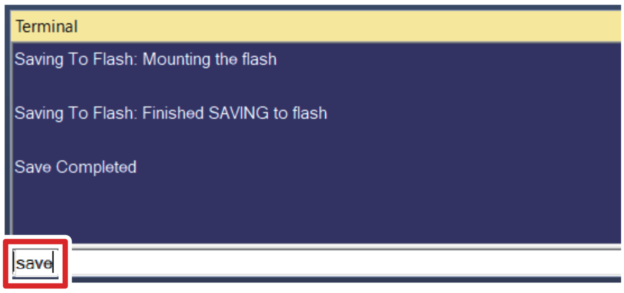
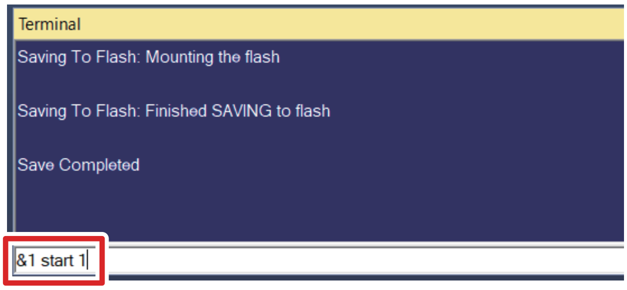


4 Make sure that there are no errors in the Output window.

- If the transfer failed, check the content of the error in the Output window.



```
Please wait while loading the project.


Uploading pp_error.log file from the PowerPMAC.
Uploading pp_proj.log file from the PowerPMAC.
Uploading pp_error_hist.log file from the PowerPMAC.
Uploading pp_debug.txt file from the PowerPMAC.
Uploading and synchronizing PowerPMAC variables
Total Project download time = 6.625 seconds.
Total Project build and download time = 46.143 seconds.
Download successful.
```

5	Type the **save** command in the Power PMAC IDE Terminal. • When the save is completed, "Save Completed" is displayed in the Terminal.	 Terminal Saving To Flash: Mounting the flash Saving To Flash: Finished SAVING to flash Save Completed save
6	Type the **&1 start 1** command in the Power PMAC IDE Terminal.	 Terminal Saving To Flash: Mounting the flash Saving To Flash: Finished SAVING to flash Save Completed &1 start 1
7	Confirm that the processing is completed correctly.	

4

How to Customize Various Settings

This section describes how to customize various settings.

4-1 globaldefinitions.pmh

This file contains various clock and task period settings.

No.	Parameter	Description
*1	Coord[1].SegMoveTime	Specify the interpolation time period in ms for circular interpolation, etc. of the trajectory. The smaller this value, the closer to the exact trajectory, but the larger the computation load on the CPU due to the increase in the amount of computation.
*2	Gate3[0].PhaseFreq	Specify the frequency of the phase clock. The larger this value, the more accurate the processing of current loops, etc., but the larger the computation load on the CPU due to the increase in the amount of computation.
*3	Gate3[0].ServoClockDiv	Specify the division ratio of the servo clock. The smaller this value, the more accurate the processing of position loops, etc., but the larger the computation load on the CPU due to the increase in the amount of computation. The servo clock frequency is determined as follows according to the set value. Servo clock frequency = Phase clock frequency / (Gate3[0].ServoClockDiv + 1)
*4	Sys.PhaseOverServoPeriod	Set the value calculated as follows. Sys.PhaseOverServoPeriod = 1 / (Gate3[0].ServoClockDiv + 1)
*5	Sys.ServoPeriod	Set the value calculated as follows. Sys.ServoPeriod = 1000 * (Gate3[0].ServoClockDiv + 1) / Gate3[0].PhaseFreq
*6	Gate3[x].PhaseServoDir	Specify the unit that supplies the phase clock and the servo clock.

4-2 MotionOnTheFly.pmh

This file contains the settings for the MOTF.

No.	Parameter	Description
*1	KdgainX	Specify the filter's coefficient to use for the MOTF for the X axis. The smaller this value, the larger the operating range of Galvo Scanner. A higher-speed response can be made by setting this value to such a small value as can make the most of the range that the Galvo Scanner can control.
*2	KdgainY	Specify the above filter's coefficient for the Y axis.
*3	GalvoSfX	Specify the galvano motor command position (bit) relative to the travel distance of the laser along the X axis (in the command unit system). (Unit: bit/command unit)
*4	GalvoSfY	Specify the above value for the Y axis.

*5	&1
	#1->I;
	#2->I;
	#3->I;
	#4->I;
		Axes 1, 2, 3, and 4 are assigned to coordinate system 1 as inverse kinematics axes. To change the axis numbers or assign other axes to the coordinate system, add this block.

4-3 MotorControl.pmh

This file contains the settings for linear motor control.

No.	Parameter	Description
*1	Motor[x].ServoCtrl	For axis x that uses servo control, set this to 1. In this manual, six axes are used. However, to use more axes, also set Motor[y].ServoCtrl (y > 6) to 1.
*2	Motor[x].pEnc, Motor[x].pEnc2	Specify the encoder table to be referenced in loop feedback for axis x. In this manual, the encoder table with the same index as the axis is assigned.
*3	Motor[x].EncType	Specify the encoder type for axis x. In this document, 6 is specified to use a sinusoidal encoder.
*4	Motor[x].PosUnit	Specify the command unit for axis x. In this manual, this is set to 3 to use mm as the unit.
*5	EncTable[x].Type	Specify the calculation method for each encoder table. In this manual, this is set to 1 to read 32-bit feedback values.
*6	EncTable[x].pEnc	Specify the register to be referenced by encoder table x. In this manual, feedback values that are calculated by the CK3W-AX2323N Unit are obtained.
*7	EncTable[x].ScaleFactor	Specify $1/2^8$ as the conversion factor to obtain 24-bit data from the CK3W-AX2323N Unit in each 32-bit encoder table.
*8	Motor[x].AmpFaultlevel	Set whether to set the amplifier fault level to High active or Low active. In this manual, this is set to High active.
*9	Motor[x].FatalFeLimit	Specify the amount of following error that stops operation at the axis x. To improve safety, set a smaller value. Contrarily, if any following error prevents operation from starting, set a larger value.
*10	Motor[x].PhaseCtrl	For axis x that use a current loop, set this to 4. In this manual, the Direct PWM control method is used for the XY stage. However, to use analog voltage control or EtherCAT control, set this to 0.
*11	Motor[x].pPhaseEnc	Specify the register from which to obtain the feedback position of each motor.
*12	Motor[x].PhasePosSf	Specify the conversion factor to convert the current position obtained in No. 11 into an electrical angle (0 to 2048). Specifically, this is calculated based on the following formula. $\text{PhasePosSf} = \frac{2048 \times A}{B}$ Here, A represents the length per pulse of Gate3[0].Chan[x].PhaseCapt. It is 0.02/16384 since one cycle of a sine wave is 0.02 mm for the LIDA48 and the number of divisions of interpolation is 16384 for the PMAC. Also, B represents the length per rotation of the motor in electrical angle. It is 36 since the ECL (Electrical Cycle Length) of the linear motor used in this manual is 36 mm.
*13	Motor[x].PhaseOffset	Specify the phase difference in the electrical angle (0 to 2048) of each phase motor. For a three-phase motor, set this to 683 or -683.
*14	Motor[x].pAdc	Specify the register in which to read feedback current values from the servo amplifier. In this manual, the feedback values read from the Direct PWM interface are used.

No.	Parameter	Description
*15	Motor[x].AdcMask	Specify which bits of the register in No. 14 you want to use for feedback values. Set this value according to the specifications of the AD converter for reading current values that is installed in the servo amplifier.
*16	Motor[x].pDac	Specify the start address of the register to which to write the command output of the Controller. Here, the U-phase (Gate3[0].Chan[0].Pwm[0]) at the start of the register is specified since the register of the CK3W-AX2323N for U-, V-, or W-phase output is the output destination.
*17	Motor[x].PhaseFindingDac	Specify the output volume during phase reference search. If phase reference search is not completed, increase the value of this parameter since the output volume may be insufficient. Also, if an I2TFault status error occurs during phase reference search, decrease the value since the output current has exceeded the rated value.
*18	Motor[x].PhaseFindingTime	Specify the duration of time in phase reference search. If phase reference search is not completed, increase the value of this parameter. Also, if an I2TFault status error occurs during phase reference search, decrease the value since the output current has exceeded the rated value.
*19	Motor[x].PwmSf	Specify the coefficient of Direct PWM output. The full range is 16384. This is normally set to less than 95% of the full range so that the duty cycle of the PWM waveform does not reach 100%. Set the parameter value according to the specifications of the servo amplifier in use.
*20	Motor[x].MaxDac	Specify the rated continuous current value of each motor and the servo amplifier. This is calculated based on the following formula. $\text{MaxDac} = \dfrac{\cos 30° \times 32{,}767 \times \text{Rated continuous current}}{\text{Full range of feedback current}}$ Here, the rated continuous current of the servo amplifier is used since it is smaller than that of the motors used for both the X and Y axes. CDHD-0032APB0: 9 A rms INC-5716D0-0S: 9.88 A rms INC-5723D0-0S: 14.82 A rms Also, the full range of feedback current is 11.25 Arms based on the specifications of the CDHD-0032APB0.
*21	Motor[x].I2tSet	Specify the rated short-time current value of each motor and the servo amplifier. This is calculated based on the following formula. $\text{I2tSet} = \dfrac{\cos 30° \times 32{,}767 \times \text{Rated short-time current}}{\text{Full range of feedback current}}$ Here, the short-time rated current of the servo amplifier is used since it is smaller than that of motors used for both the X and Y axes. CDHD-0032APB0: 3 A rms INC-5716D0-0S: 3.08 A rms INC-5723D0-0S: 4.62 A rms

No.	Parameter	Description
*22	Motor[x].I2tTrip	Specify the allowable time for the rated short-time current value of each motor and the servo amplifier. This is calculated based on the following formula. The allowable time is 2 seconds based on the specifications of the CDHD-0032APB0. I2tTrip = (MaxDac^2 + IdCmd^2 - I2tSet^2) × Allowable time (s)
*23	Gate3[0].Chan[x].PwmDeadTime	Specify 800 nanoseconds as the dead time of the PWM signal. The dead time is calculated by the following formula. DeadTime = 0.0533 μs × Gate3[0].Chan[x].PwmDeadTime Set this value according to the specifications of the servo amplifier.
*24	Gate3[0].Chan[x].PwmFreqMult	Set the PWM output frequency to 15 kHz. The PWM output frequency is calculated by the following formula. $f_{PWM} = \frac{\text{PwmFreqMult} + 1}{2} \times \text{Phase frequency}$ Set the PWM frequency to 40 kHz or less according to the maximum input frequency of the servo amplifier.
*25	Gate3[0].Chan[x].PackOutData	Enable the compression format to realize a highly efficient algorithm that completes writing to the output register for the U, V, or W phase in two times. If you use the CK3W-AX2323N, do not change this value.
*26	Gate3[0].AdcAmpStrobe	Specify the ADC strobe word (control command sent from the Controller to the servo amplifier according to the Direct PWM standards). Change this value according to the specifications of the servo amplifier.
*27	Gate3[0].AdcAmpHeaderBits	Specify the number of header bits included in feedback current value data. Change this value according to the specifications of the servo amplifier.
*28	Gate3[0].AdcAmpClockDiv	Specify the clock signal frequency of the AD converter that reads current data in the servo amplifier.
*29	Gate3[0].Chan[x].PackInData	Enable the compression format to realize a highly efficient algorithm that completes reading from the input register for the U, V, or W phase in two times. If you use the CK3W-AX2323N, do not change this value.
*30	Gate3[0].Chan[x].AtanEna	Enable this if you use a sinusoidal encoder. If you use a pulse encoder or serial encoder, set it to 0.
*31	Gate3[0].Chan[x].EncCtrl	Set the decoding method of the encoder. In this manual, quadruple AB-phase CW decoding is set.
*32	Motor[x].PosSf, Motor[x].Pos2Sf	Specify the conversion factor between the feedback value calculated by the Axis Interface Unit and the command position. This is set to 0.020/16384 since the pulse count per cycle is 16834 when a sinusoidal encoder is used and one cycle is 0.020 mm for the sinusoidal encoder.

4-4 SensorControl.pmh

This file contains the settings for overtravel limit switches.

No.	Parameter	Description
*1	Motor[x].pLimits	Specify the register to be referenced by the overtravel limit switch.
*2	Motor[x].LimitBits	Specify which bits of the register (No. 1) are to be referenced by the overtravel limit switch.

4-5 Galvano.pmh

This file contains the settings for controlling the Galvo Scanner.

No.	Parameter	Description
*1	Gate3[1].SerialEncCtrl	Set this according to the figure below.
*2	Motor[x].ServoCtrl	For axis x that uses servo control, set this to 1. In this manual, six axes are used. However, to use more axes, also set Motor[y].ServoCtrl (y > 6) to 1.
*3	Motor[x].Ctrl	For galvano control, set this to Sys.PosCtrl since it performs position control.
*4	Motor[x].MaxSpeed	Specify the maximum velocity of galvano control. This time, specify 0 to disable the maximum velocity monitor.
*5	Motor[x].MaxPos	Specify the maximum value of the command position for galvano control. This time, specify 0 to disable it.
*6	Motor[x].MinPos	Specify the minimum value of the command position for galvano control. This time, specify 0 to disable it.
*7	Motor[x].pDac	Specify the start address of the register to which to write the command output of the Controller. Here, specify Gate3[1].Chan[x].Dac[0] since the register of the CK3W-GC2200 for XY output is the output destination.
*8	Motor[x].pLimits	Set the overtravel limit of galvano control. This time, specify 0 to disable it.
*9	Motor[x].FatalFeLimit	Specify the amount of following error that stops operation at the axis x. Specify 0 because Galvo Scanner disables this setting.
*10	Motor[x].pEnc	Specify the encoder table to be referenced in position loop feedback for axis x. In this manual, the encoder table with the same index as the axis is assigned.
*11	Motor[x].pEnc2	Specify the encoder table to be referenced in velocity loop feedback for axis x. In this manual, the encoder table with the same index as the axis is assigned.
*12	EncTable[x].Type	Specify the calculation method for each encoder table. In this manual, this is set to 1 to read 32-bit feedback values.
*13	EncTable[x].pEnc	Specify the register to be referenced by encoder table x. In this manual, the linear interpolation values calculated by the CK3W-GC2200 Unit are output.
*14	EncTable[x].ScaleFactor	Specify $1/2^{12}$ as the conversion factor to obtain 20-bit SL2-100 data in each 32-bit encoder table.

Reserved												TxEnable	Reserved	ClockSel	Sync	Control_Bits			TriggerEdge	Tx_Valid	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			
–												Enable TX of SL2-100	–	Phase Clock	Asynchronous Mode	Command Position			Rising Edge	Validate Tx data	–										

For Command Position

Reserved												TxEnable	Reserved	ClockSel	Sync	Control_Bits			TriggerEdge	Tx_Valid	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	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
0				0				0				8				E				8				0				0			
–												Enable TX of SL2-100	–	Phase Clock	Asynchronous Mode	Control Commands			Rising Edge	Validate Tx data	–										

For Control Commands

4-6 LaserControl.pmh

This file contains the settings for controlling the laser oscillator.

No.	Parameter	Description
*1	Gate3[1].Chan[0].CompA	Set this according to the figures below.
*2	Gate3[1].Chan[1].CompA	Set the delay time of SL2-100 communications data to the internal clock of the PMAC. In this manual, this is set to 0.
*3	Gate3[1].Chan[2].CompA	Set this according to the figures below.

DutyCycle												PWMPeriod												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
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
8				0				0				0				D				0				0				0			
Duty Ratio = 50%												PWM Frequency = 30 kHz												–							

Settings of Gate3[1].Chan[0].CompA

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																			

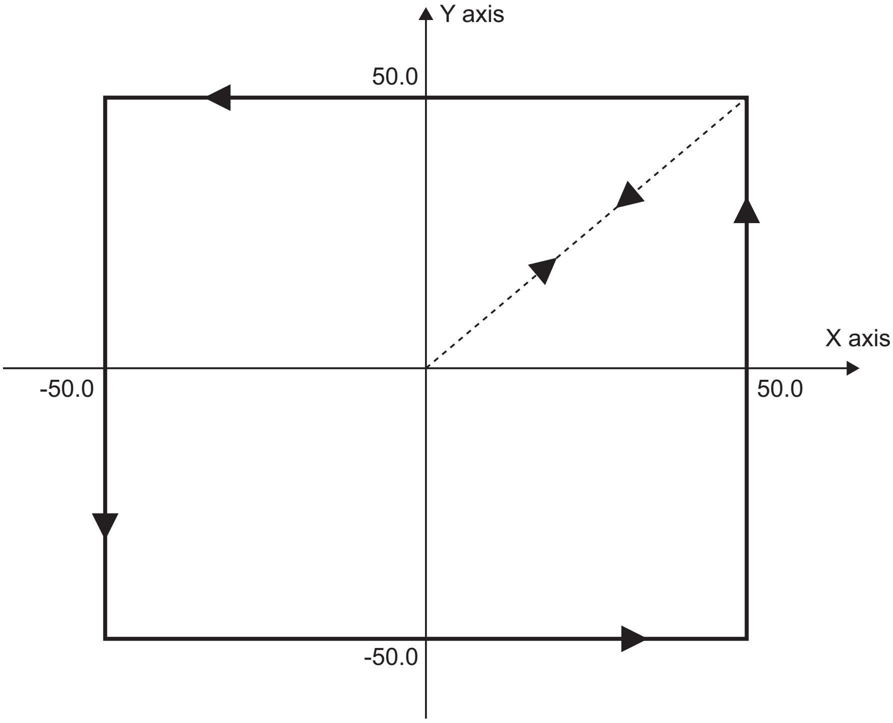
Settings of Gate3[1].Chan[2].CompA

4-7 TCRConfigurations.pmh

This file contains the condition settings for turning ON/OFF the laser.

No.	Parameter	Description
*1	Gate3[1].Chan[1].CompB	Set the travel distance before the laser turns from OFF to ON.
*2	Gate3[1].Chan[1].CompB	Set the travel distance before the laser turns from ON to OFF. After this, add the condition to toggle the laser output.
*3	Gate3[1].Chan[2].CompB	Set this according to the figures below.

The figure bellow illustrates the condition used in this manual, where laser light is output at home, travels around the square, and returns to the home again.



CompareEnable	ClearTable	CompClkSel	Reserved	CompOutWrite		CompOutPol	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
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				0			
Compare Enabled	Normal	Servo Clock	–	Normal		H active	–																								

Settings of Gate3[1].Chan[2].CompB

OMRON Corporation **Industrial Automation Company**
Kyoto, JAPAN

Contact: www.ia.omron.com

Regional Headquarters

OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor: