Servo control sample program

RX Family

Summary

This application note describes the sample program about inertia estimation and return origin function which are added to vector control using encoder or induction sensor based on Renesas microcontrollers. About the vector control for permanent magnet synchronous motor with encoder or induction sensor, please refer to each application note “RA01AN6187” and “R01AN6317”.

The targeted software for this application note is only to be used as reference purposes and Renesas Electronics Corporation does not guarantee the operations. Please use this after carrying out a thorough evaluation in a suitable environment.

Operation checking device

Operations of the target software of this application note are checked by using the following device.

・ RX72M（R5F572MNDDBD）
・ RX72T（R5F572TKCDFB）
・ RX66T（R5F566TEADFP）
・ RX24T（R5F524TAADFP）

Target software

The following shows the target software for this application note:

・ RXxxx_ESB_SPM_ENCD_FOC_SERVO_CSP_RV100
・ RXxxx_ESB_SPM_ENCD_FOC_SERVO_E2S_RV100
“xxx” is the name of the MCU to be used.

Contents

1. Overview

The purpose of this application note is to explain about inertia estimation and return origin function with a sample program that uses a Renesas RX microcontrollers (MCU) to drive a permanent magnet synchronous motor with position control.

About vector control, the sample program uses the algorithm that is explained in 'Vector control with encoder for permanent magnet synchronous motor (Algorithm)' (R01AN3789), therefore pleaser refer to that for details of the algorithm.

2. Development environment

2.1 Operation check environment

Table 2-1 and Table 2-2 show the development environment of the target software.

Table 2-1 Hardware development environment (H/W)

Classification	Product used
Microcomputer / CPU card P/N	RX72M (R5F572MNDDBD) / RTK0EMXDE0C00000BJ RX72T (R5F572TKCDFB) / RTK0EMX990C00000BJ RX66T (R5F566TEADFP) / RTK0EMX870C00000BJ RX24T (R5F524TAADFP) / RTK0EM0009C03402BJ
Inverter board	Evaluation System for BLDC Motor (RTK0EM0000B10020BJ) included Inverter board for 48V 5A BLDC
motor	BLY171D-24V-4000 (manufactured by Anaheim Automation) and so on
Sensor	Encoder: AMT102-V (manufactured by CUI DEVICES) Magnet sensor : TAS2143, TAD2141 Induction sensor : IPS2200

Table 2-2 Software development environment (S/W)

IDE version	RX smart configurator	Toolchain version
CS+: V8.09.00	Version 2.11.0	CC-RX: V3.05.00
e^2studio: 2023-04	e^2studio plug-in version	

For purchase and technical support, please contact our sales and distributors.

2.2 Hardware specifications

2.2.1 Hardware configuration diagram

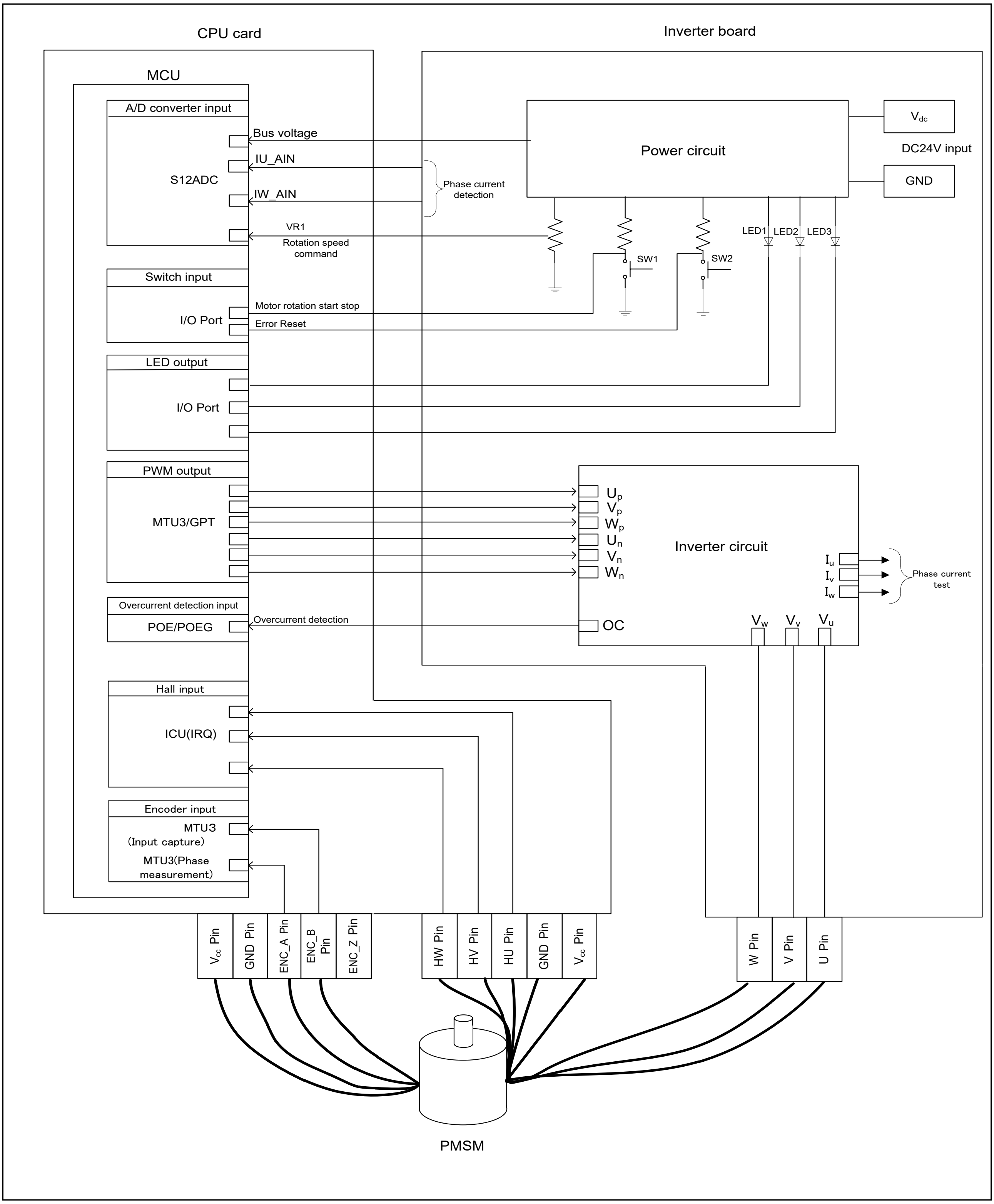

Figure 2-1 Hardware configuration diagram : Encoder

2.2.2 User interface

Table 2-3 shows a list of user interfaces for this system.

Table 2-3 User Interface

Item	Interface parts	Function
Rotation position/speed	Variable resistor (VR1)	Rotation position/speed command value input (analog value)
START/STOP	Toggle switch (SW1)	Motor rotation start/stop changeover switch
ERROR RESET	Push switch (SW2)	Return command from error status
LED1	Orange LED	・When the motor is rotating: Lights up ・When stopped: Off
LED2	Orange LED	・When an error is detected: Lights up ・During normal operation: Off
LED3	Orange LED	・Positioning completed: Lights up ・Positioning not completed: Off
RESET	Push switch (RESET1)	System reset

2.2.3 Peripheral functions

The I/O function and peripheral functions used in this system are explained below. Since the terminal assignments differ depending on used CPU card, the allocation of MCU peripheral functions also differs. In the sample program, the peripheral functions are set using Smart configurator.

Table 2-4 Input/output functions and peripheral functions

Function	Peripheral functions
Inverter bus voltage measurement	S12AD
Rotation position/speed command value input (analog value)	S12AD
START/STOP toggle switch	I/O Port (Input)
LED1 on/off control	I/O Port(output)
LED2 on/off control	I/O Port(output)
LED3 on/off control	I/O Port(output)
U-phase current measurement	S12AD
W-phase current measurement	S12AD
PWM output (Up) / "Low" active	MTU/GPT *1
PWM output (Vp) / "Low" active	MTU/GPT *1
PWM output (Wp) / "Low" active	MTU/GPT *1
PWM output (Un) / "High" active	MTU/GPT *1
PWM output (Vn) / “High” active	MTU/GPT *1
PWM output (Wn) / “High” active	MTU/GPT *1
Hall U phase input	ICU(IRQ)
Hall V phase input	ICU(IRQ)
Hall W phase input	ICU(IRQ)
Encoder A phase input	MTU
Encoder B phase input	MTU
Digital input from magnet sensor	MTU
SPI interface with magnet sensor	RSPI
Analog input from induction sensor	S12AD
PWM emergency stop input when overcurrent is detected	POE/POEG *2

Note: 1. Peripheral functions that perform PWM output differ depending on the CPU card.
2. Since the input and PWM output stop request are linked, the used peripheral functions are different depending on each CPU card.

2.3 Software

The sample program consists of an application layer, a motor module, and a Smart configurator. The motor module controls by receiving instructions from the application layer operated by the user. The output to the HW layer is done via the Smart configurator.

2.3.1 Overall configuration

Figure 2-2 shows the overall configuration of the software. 'Inertia estimate' and 'Return origin' modules are added.

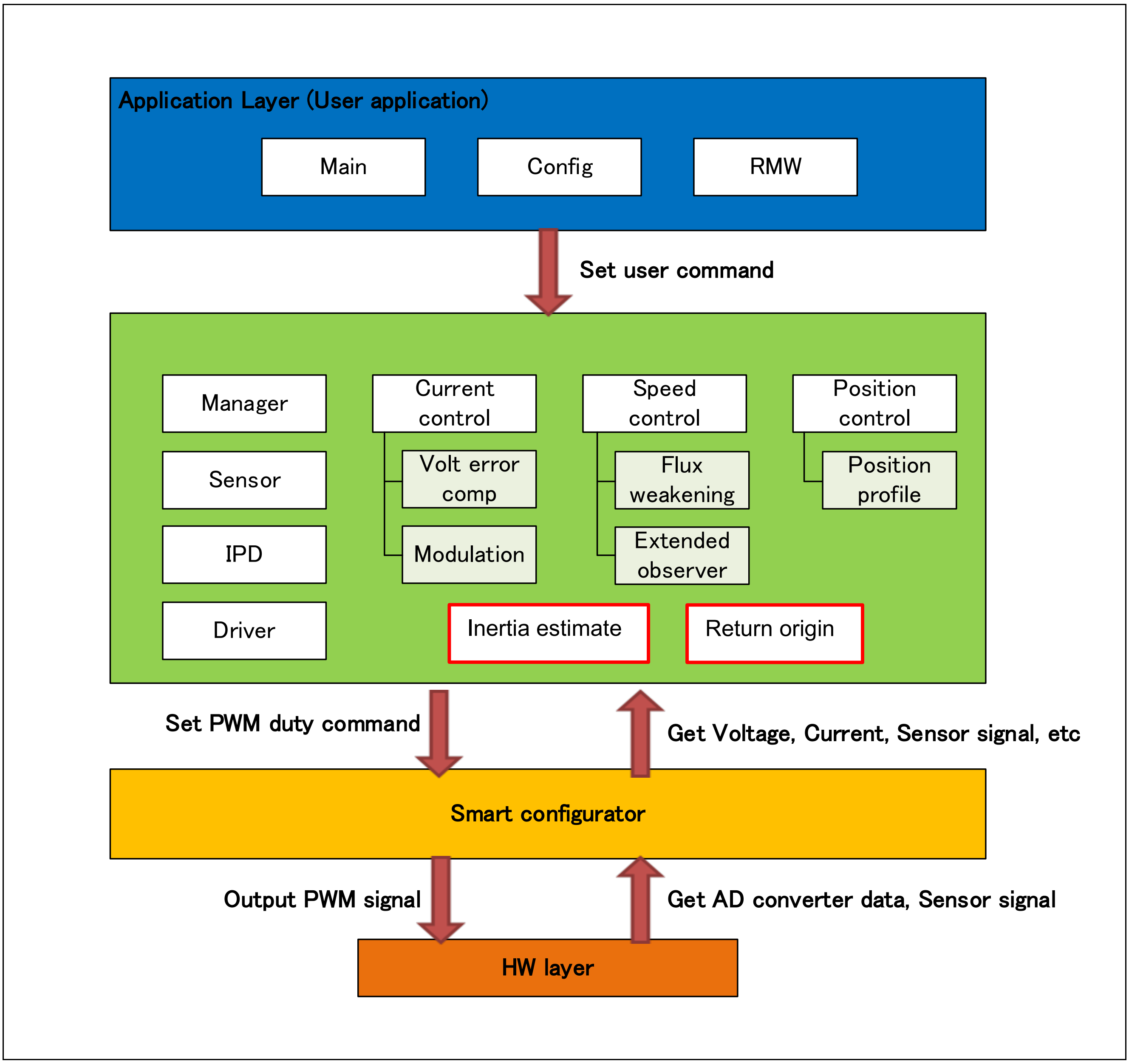

Figure 2-2 Overall configuration of motor control software

2.4 File/folder structure

Table 2-5 shows the folder and file structure of the sample program.

Table 2-5 File/folder structure [1/2]

Folder	Sub-folder	File	Remarks
app	main	r_app_main.c/h	User main function
	rmw	r_app_rmw.c/h	RMW Analyzer UI related function definition
		r_app_rmw_interrupt.c	RMW interrupt function definition
		ICS2_RX"xxx".lib/h, ICS_RX"xxx".obj/h	RMW communication library
	board_ui	r_app_board_ui.c/h	Board UI related function definition
		r_app_board_ui_ctrl.h	MCU-dependent board UI function definition
		r_app_board_ui_ctrl_rx"xxx"_esb.c	MCU-dependent board UI function definition
	cfg	r_app_control_cfg.h	App layer configuration definition
motor_module	encoder_vector_rx	r_motor_encoder_vector_action.c	Action function definition
		r_motor_encoder_vector_api.c/h	Manager module API function definition
		r_motor_encoder_vector_manager.c/h	Manager module local function definition
		r_motor_encoder_vector_protection.c/h	Function definition of protection function
		r_motor_encoder_vector_ statemachine.c/h	Function definition related to state transition
	current_rx	r_motor_current_api.c/h	Current control module API function definition
		r_motor_current.c/h	Current control module local function definition
		r_motor_current_modulation.c/h	Modulation module function definition
		r_motor_current_volt_err_comp.lib/h	Function definition of voltage error compensation module
		r_motor_current_pi_gain_calc.c	Current control module control gain calculation function definition
	speed_rx	r_motor_speed_api.c/h	API function definition of speed control module
		r_motor_speed.c/h	Local function definition of speed control module
		r_motor_speed_fluxwkn.lib/h	Function definition of weak flux module
		r_motor_speed_extobserver.lib/h	Function definition of the disturbance observer module
		r_motor_speed_pi_gain_calc.c	Calculation of control gain of speed control module
	position_rx	r_motor_position_api.c/h	Position control module API function definition
		r_motor_position.c/h	Position control module local function definition
		r_motor_position_profiling.c/h	Function definition for position control command value creation
		r_motor_position_gain_calc.c	Position control module control gain calculation function definition
	ipd_rx	r_motor_ipd_api.lib/h	IPD module API function definition

Table 2-6 File/folder structure [2/2]

Folder	Sub-folder	File	Remarks
motor_module	driver_rx	r_motor_driver.c/h	Driver module function definition
	sensor_rx	r_motor_sensor_api.c/h	Sensor module API function definition
		r_motor_sensor_encoder.c/h	Sensor module encoder processing function definition
		r_motor_sensor_hall.c/h	Hall sensor processing function definition for sensor module
	inertia_est_rx	r_motor_inertia_est_api.c/h	API functions for inertia estimation
		r_motor_inertia_est.c/h	Variables process for inertia estimation
	return_org_rx	r_motor_return_org_api.c/h	API functions for return origin
		r_motor_return_org.c/h	Variables process for return origin
		r_motor_return_org_push.c/h	Process for pushing to the stopper
	general	r_motor_filter.c/h	General-purpose filter function definition
		r_motor_pi_control.c/h	PI control function definition
		r_motor_common.h	Common definition
	cfg	r_motor_inverter_cfg.h	Inverter configuration definition
		r_motor_module_cfg.h	Control module configuration definition
		r_motor_targetmotor_cfg.h	Motor configuration definition
src	smc_gen	−	Drivers and APIs generated by Smart configurator

Note: The name of the MCU is entered in "xxx". ex. RX72T etc.

3. Description of functions

Basic functions are not changed from original software which support position control. Therefore, in this chapter, only "Inertia estimation" and "Return origin" functions are described.

3.1 Contents of control

3.1.1 Method to estimate inertia

The inertia estimation function estimates the rotor and load inertia connected to the motor shaft. Since inertia is an important parameter to drive the motor properly, this function is used to estimate it.

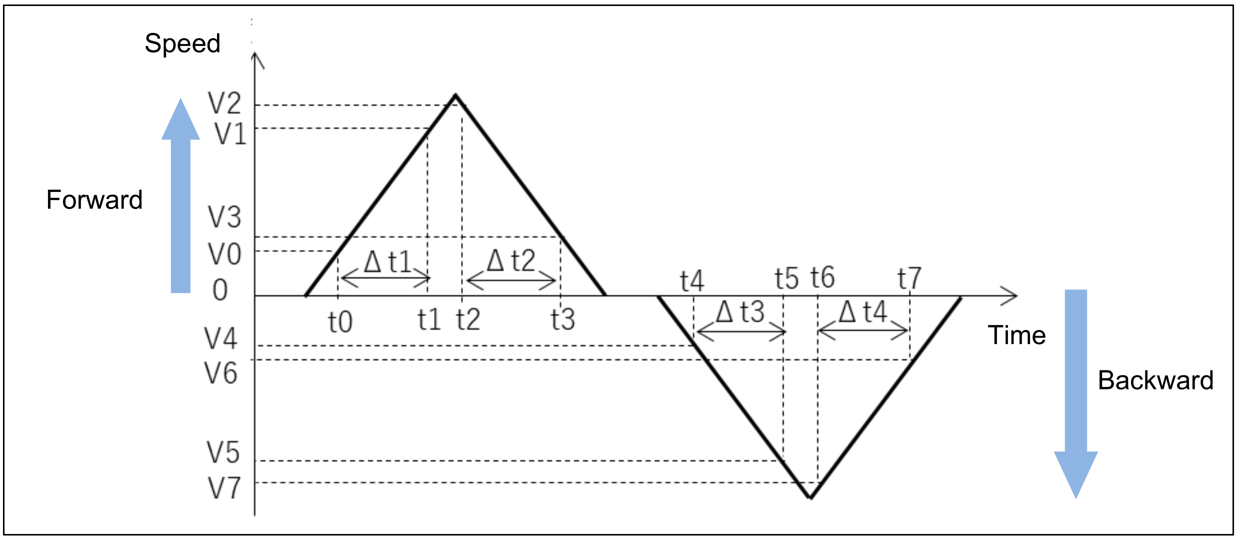

Figure 3-1 Action at inertia estimation

Inertia estimation is performed automatically after the function is started. The motor is rotated forward and backward in same angle with position control like as above figure. Inertia is estimated with the acceleration period at forward(Δt1), deceleration period at forward(Δt2), acceleration period at backward(Δt3), and deceleration at backward(Δt4), the average of q-axis current and acceleration in each period.

3.1.2 Return origin

Return to origin is an action that positions the motor to the mechanical reference position. It is used to return to the reference position when the reference position is not known at startup.

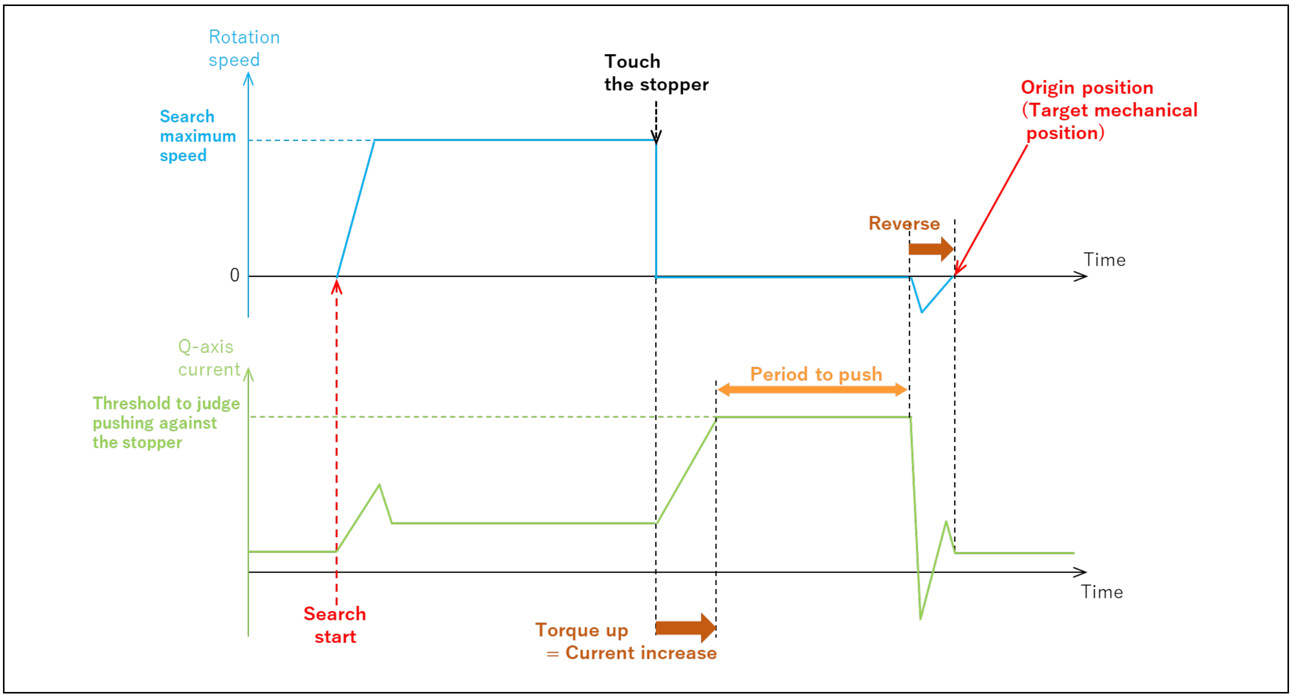

Figure 3-2 Example of return origin movement with pushing method

After the function is started, the motor starts rotation according to set search speed and direction. When the motor hits the stopper, torque is increased. At that time, q-axis current is also increased according to torque. When the q-axis current reaches set threshold value, the motor is judged to hit the stopper. After waiting for set period for pushing, the motor rotate reverse according to set reverse angle, then the function finish.

3.2 Function specifications

3.2.1 API functions for inertia estimation

The list of API functions for inertia estimation is shown below.

Table 3-1 API functions for inertia estimation

API function	Arguments	Contents
R_MOTOR_INERTIA_EST_Open	-	Open inertia estimate module
R_MOTOR_INERTIA_EST_Close	-	Close inertia estimate module
R_MOTOR_INERTIA_EST_Reset	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables	Reset inertia estimate module
R_MOTOR_INERTIA_EST_ParameterSet	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables st_inertia_est_input_t * p_st_inertia_est_input / Pointer of structure for input data to inertia estimate module	Set necessary data to inertia estimate module
R_MOTOR_INERTIA_EST_ParameterUpdate	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables st_inertia_est_cfg_t * p_st_inertia_est_cfg / Pointer of structure for configuration of inertia estimate	Update the configuration about inertia estimate
R_MOTOR_INERTIA_EST_GetStatus	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables st_inertia_est_sts_t * p_st_inertia_est_sts / Pointer of structure to get status from inertia estimate module	Get the status of inertia estimation
R_MOTOR_INERTIA_EST_UpdateGetStatus	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables st_inertia_est_sts_t * p_st_inertia_est_sts / Pointer of structure to get status from inertia estimate module	Get updated status of inertia estimation
R_MOTOR_INERTIA_EST_CheckErrorStatus	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables uint8_t u1_motor_status / status of motor rotation uint8_t u1_ctrl_loop_mode / control method	Check error about inertia estimation
R_MOTOR_INERTIA_EST_Run	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables	Start inertia estimation
R_MOTOR_INERTIA_EST_UpdateRun	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables	Update the ratio of inertia
R_MOTOR_INERTIA_EST_Abort	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables	Abort inertia estimation
R_MOTOR_INERTIA_EST_InertiaEstCyclic	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables	Cyclic process of inertia estimation in speed control cyclic
R_MOTOR_INERTIA_EST_InertiaEstCurrentCyclic	st_inertia_est_ctrl_t * p_st_iec / Pointer of structure for inertia estimate module variables	Cyclic process of inertia estimation in current control cyclic

3.2.2 API functions for return origin

The list of API functions for return origin is shown below.

Table 3-2 API functions for return origin

API function	Arguments	Contents
R_MOTOR_RETURN_ORG_Open	-	Open return origin module
R_MOTOR_RETURN_ORG_Close	-	Close return origin module
R_MOTOR_RETURN_ORG_Reset	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables	Reset return origin module
R_MOTOR_RETURN_ORG_ParameterSet	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables st_return_org_input_t * p_st_return_org_input / Pointer of structure for input to return origin module	Set necessary data to return origin module
R_MOTOR_RETURN_ORG_ParameterUpdate	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables st_return_org_cfg_t * p_st_return_org_cfg / Pointer of structure for configuration of return origin	Update configuration data about return origin
R_MOTOR_RETURN_ORG_ParameterGet	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables st_return_org_output_t * p_st_return_org_output / Pointer of structure for output data of return origin module	Get information from return origin module
R_MOTOR_RETURN_ORG_GetStatus	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables st_return_org_sts_t * p_st_return_org_sts / Pointer of structure for status of return origin module	Get status of return origin module
R_MOTOR_RETURN_ORG_CheckErrorStatus	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables uint8_t u1_motor_status / status of motor rotation uint8_t u1_ctrl_loop_mode / control method	Check error of return origin module
R_MOTOR_RETURN_ORG_Run	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables	Start return origin function
R_MOTOR_RETURN_ORG_Abort	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables	Abort return origin function
R_MOTOR_RETURN_ORG_ReturnOrgCyclic	st_return_org_ctrl_t * p_st_roc / Pointer of structure for return origin module variables	Return origin process which should be performed in speed control cyclic

4. Operation with Renesas Motor Workbench (RMW)

Servo tuning functions can be performed easily with the tool “Renesas Motor Workbench” (from here “RMW”). In this chapter, the operation how to use is described.

4.1 Operation at the beginning

When RMW application is started, the below window is displayed.

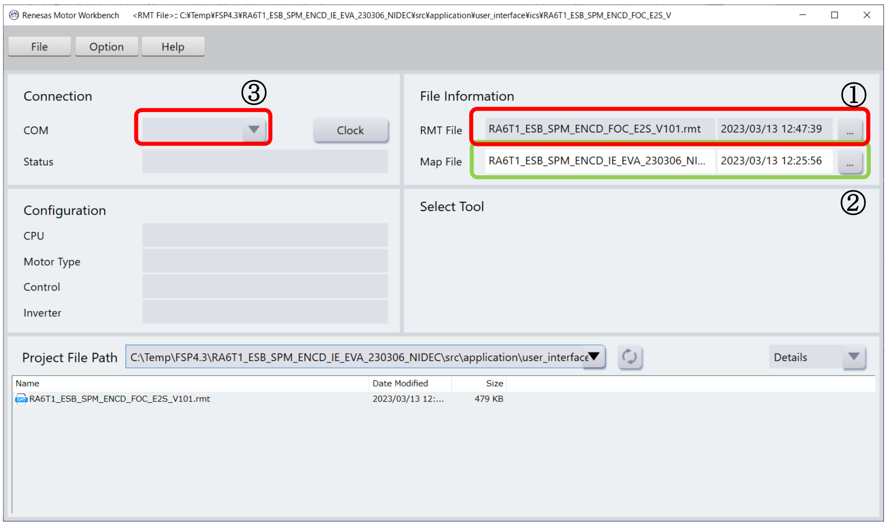

Figure 4-1 First window of RMW

Please perform below procedures.

i. Click the icon “…” of right side of “File information” / “RMT File”. And then select RMT file (***.rmt) which is included below the folder “app/rmw”.

ii. Click the icon “…” of right side of “Map File”. And then select a map file (***.map) which is included below same folder.

After that, a window to reflect variable map is popped up, please click “Set”.

iii. After above procedure finished, connect the application to the target. Click the icon “▼” of right side of “Connection”/”COM”. And then select correct COM port from the pull down menu.

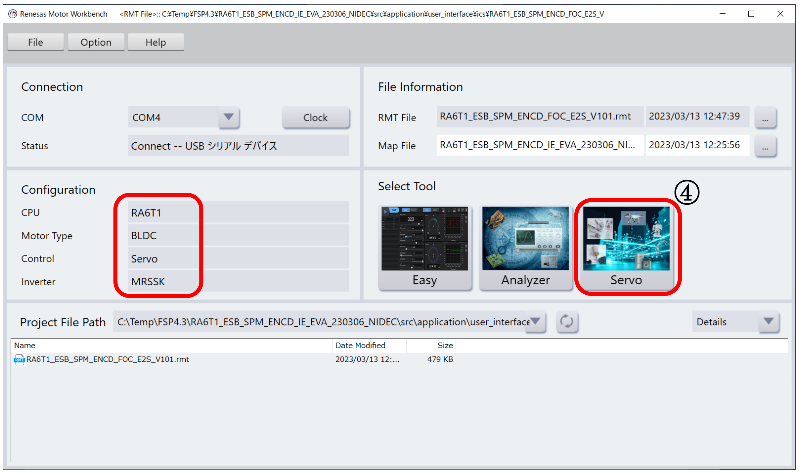

Figure 4-2 Main window of RMW

If the connection is valid, then the window changes like above.

Please confirm the information in "Configuration" like above. (Figure 4-2 is an example with RA6T1.)

If the information is displayed correctly,

iv. Click "Servo" icon in "Select Tool".

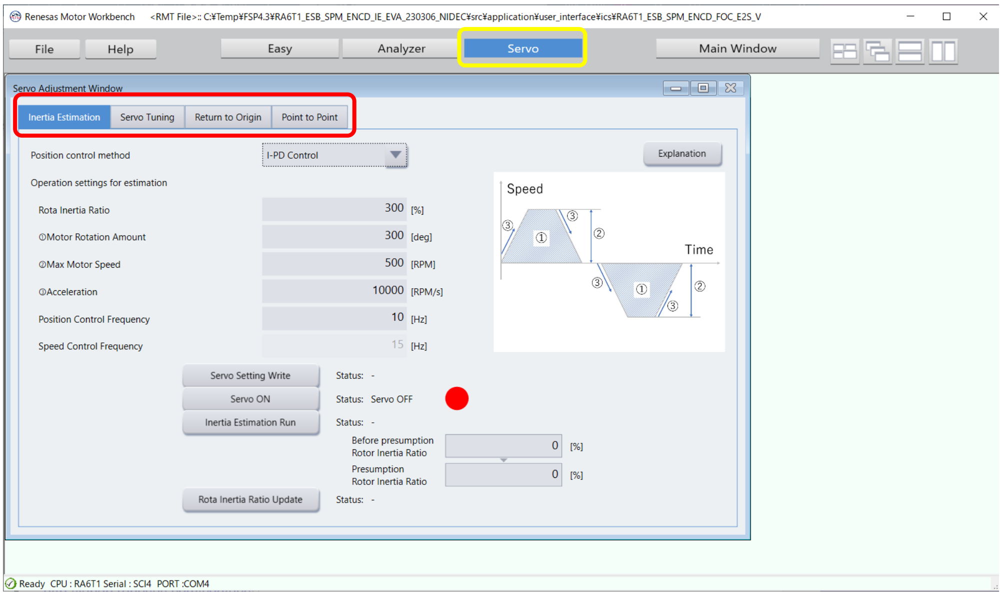

Figure 4-3 Servo function window (selected "Inertia Estimation")

In the case that the application and CPU work correctly, above window is displayed.

This window is "Servo function" window that is displayed upper side (yellow flamed).

In this window, user can use several functions with selection of function TAB (red flamed).

- Inertia Estimation
- Servo Tuning
- Return to Origin
- Point to Point

How to use these functions are explained below.

4.2 Operation of "Inertia Estimation"

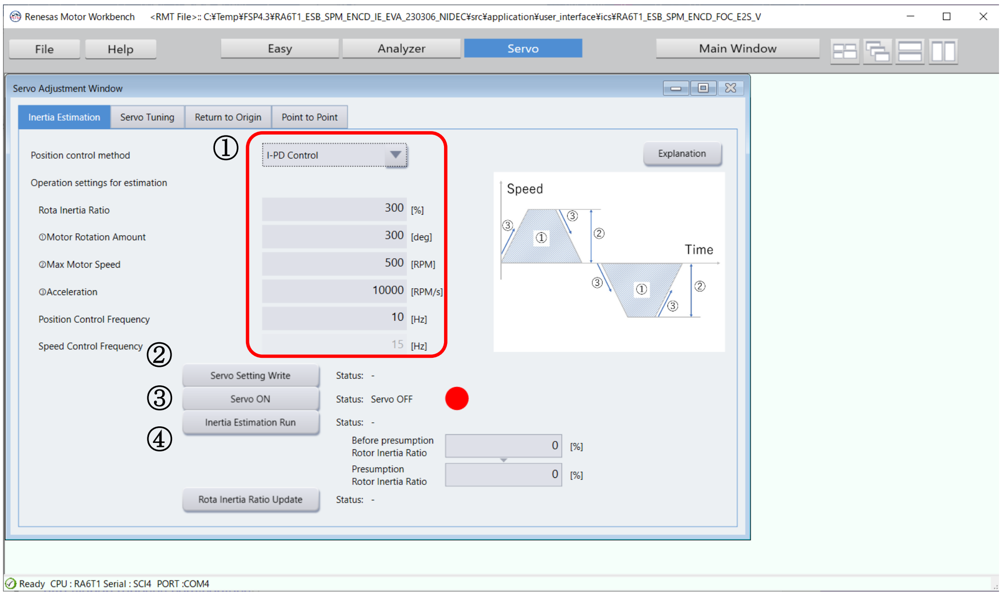

Figure 4-4 Operation window of "Inertia Estimation"

User can perform inertia estimation easily with below procedure in this window.

i. Set parameters for inertia estimation.

Table 4-1 Parameters for inertia estimation

Item	Description
Position control method	Specify the method for position control.
Rota Inertia Ratio	Set the rotor inertia ratio.
Motor Rotation Amount	Set the motor rotation amount.
Max Motor Speed	Set the maximum motor speed.
Acceleration	Set the acceleration.
Position Control Frequency	Set the position control natural frequency.
Speed Control Frequency	Set the speed control natural frequency. You can enter it when "Position control method" is "PID Control".

ii. Click "Servo Setting Write" button to reflect the parameters. Even if you don't change parameters, please click the button to reflect initial values.

iii. Click "Servo ON" button. Then the status changes to "Servo ON" and red indicator changes to green. The motor is turned on "position control". And clicked button name also changes to "Servo OFF".

iv. Click "Inertia Estimation Run" button. And then, inertia estimation starts. Clicked button name changes to "Inertia Estimation Stop". After the process finished correctly, the button returns to "Inertia Estimation Run" automatically. When an error happened, error window is popped up.

When you change to other window, click "Servo OFF" button to cancel "position control".

4.3 Operation of return origin

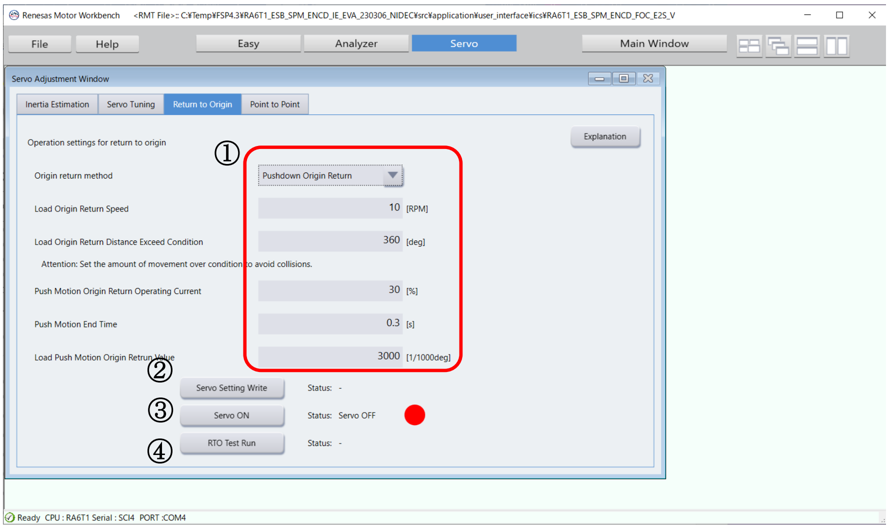

Figure 4-5 Operation window of "Return Origin"

User can perform return origin easily with below procedure in this window.

i. Set parameters for return origin.

Table 4-2 Parameters for return origin

Item	Description
Load Origin Return Speed	Set the load speed for return to origin.
Load Origin Return Distance Exceed Condition	Set the angle to determine impossibility of search during origin search.
Push Motion Origin Return Operating Current	Set the push motion origin return operating current.
Push Motion End Time	Set the push motion ending time.
Load Push Motion Origin Return Value	Set the return value of the load push motion origin return.

ii. Click "Servo Setting Write" button to reflect the parameters. Even if you don't change parameters, please click the button to reflect initial values.

iii. Click "Servo ON" button. Then the status changes to "Servo ON" and red indicator changes to green. The motor is turned on "position control". And clicked button name also changes to "Servo OFF".

iv. Click "RTO Test Run" button. And then, return origin function starts. Clicked button name changes to "RTO Test Stop". After the process finished correctly, the button returns to "RTO Test Run" automatically. When an error happened, error window is popped up.

When you change to other window, click "Servo OFF" button to cancel "position control".

4.4 Operation of servo tuning

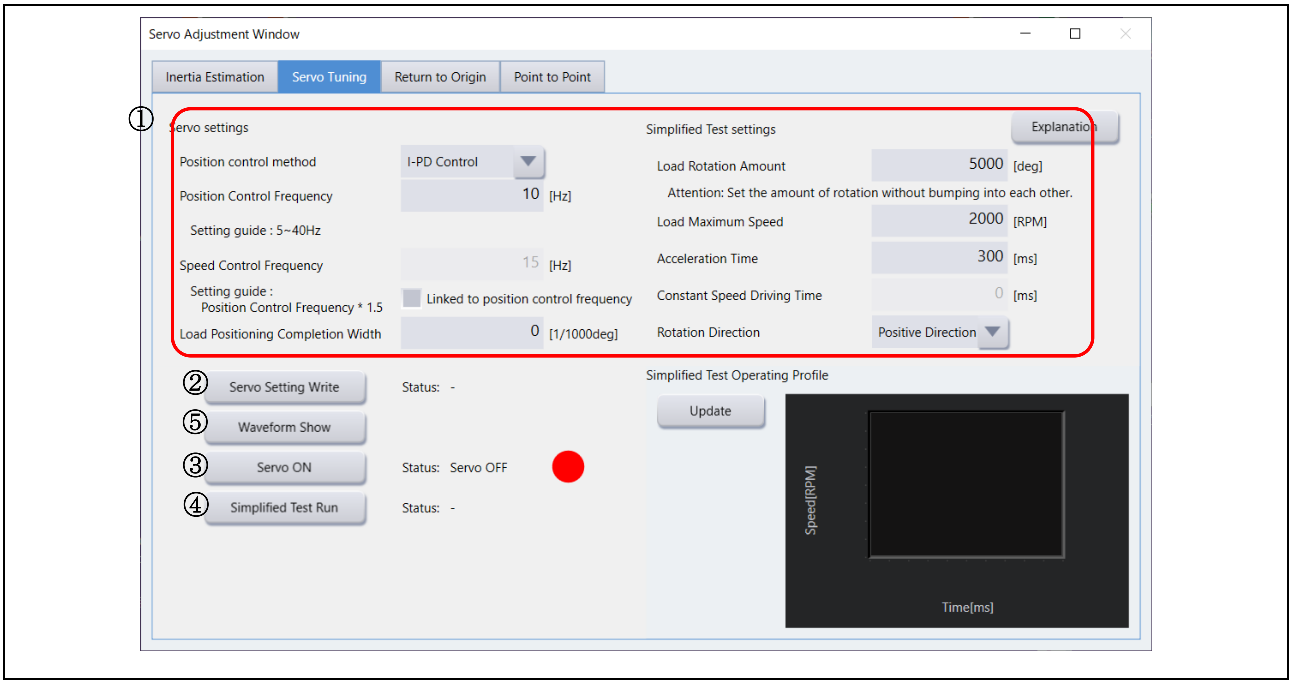

Figure 4-6 Operation window of "Servo Tuning"

User can perform servo tuning easily with below procedure in this window.

i. Set parameters for servo tuning.

Table 4-3 Parameters for servo tuning

Item	Description
Position control method	Specify the method for position control
Position Control Frequency	Set the position control natural frequency.
Speed Control Frequency	Set the speed control natural frequency. You can enter it when "Position control method" is "PID Control".
Linked to position control frequency	When checked, the position control natural frequency is multiplied by 1.5, and the value is set to the speed control natural frequency automatically. The value set in No.3 is overwritten.
Load Positioning Completion	Set the range of the load positioning completion.
Load Rotation Amount	Set the load rotation amount.
Load Maximum Speed	Set the load maximum speed.
Acceleration Time	Set the acceleration time.
Rotation Direction	Set the rotation direction.

ii. Click "Servo Setting Write" button to reflect the parameters. Even if you don't change parameters, please click the button to reflect initial values.

iii. Click "Servo ON" button. Then the status changes to "Servo ON" and red indicator changes to green. The motor is turned on "position control". And clicked button name also changes to "Servo OFF".

iv. Click "Simplified Test Run" button. And then, servo tuning starts. Clicked button name changes to "Simplified Test Stop". After the process finished correctly, the button returns to "Simplified Test Run" automatically. When an error happened, error window is popped up.

When you change to other window, click "Servo OFF" button to cancel "position control".

4.5 Operation of “Point to Point”

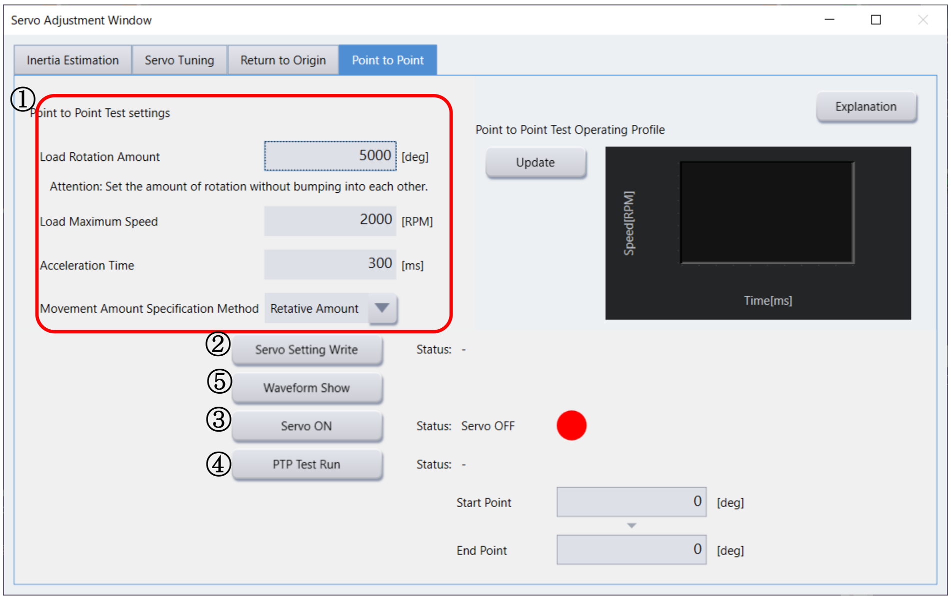

Figure 4-7 Operation window or “Point to point”

User can perform “Point to point” easily with below procedure in this window.

i. Set parameters for “Point to point”.

Table 4-4 Parameters for “Point to point”

Item	Description
Load Rotation Amount	Set the load rotation amount.
Load Maximum Speed	Set the load maximum speed.
Acceleration Time	Set the acceleration time.
Movement Amount Specification Method	Set the method for specifying movement amount.

ii. Click “Servo Setting Write” button to reflect the parameters. Even if you don’t change parameters, please click the button to reflect initial values.

iii. Click “Servo ON” button. Then the status changes to “Servo ON” and red indicator changes to green. The motor is turned on “position control”. And clicked button name also changes to “Servo OFF”.

iv. Click “PTP Test Run” button. And then, servo tuning starts. Clicked button name changes to “PTP Test Stop”. After the process finished correctly, the button returns to “PTP Test Run” automatically. When an error happened, error window is popped up.

When you change to other window, click “Servo OFF” button to cancel “position control”.

5. Reference materials

- Renesas Motor Workbench V.3.00 User's Manual (R21UZ0004)
- Evaluation System for BLDC Motor User's Manual (R12UZ0062)
- Smart Configurator User's Manual RX API Reference (R20UT4360)
- RX Smart Configurator User Guide: CS + Edition (R20AN0470)
- RX Smart Configurator User Guide: e^2 studio edition (R20AN0451)
- RX72M CPU Card Instruction Manual (R12UZ0098)
- RX72M Group User's Manual Hardware Edition (R01UH0804)
- RX24T CPU Card User’s Manual (R20UT3696)
- RX24T Group User’s Manual: Hardware (R01UH0576)
- RX66T CPU Card User’s Manual (R12UZ0028)
- RX66T Group User’s Manual: Hardware (R01UH0749)
- RX72T CPU Card User’s Manual (R12UZ0031)
- RX72T Group User’s Manual: Hardware (R01UH0803)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May 30, 2023	-	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.