RZ/T2H

Vector Control for Permanent Magnetic Synchronous Motor with Encoder (9-axes) - Absolute Encoder

Introduction

This document describes encoder-based vector control of permanent magnet motors (9 axes) using an evaluation board equipped with the RZ/T2H MPU from Renesas Electronics.

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

Operating device

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

・RZ/T2H (R9A09G077M44GBG)

Contents

1. Introduction

1.1 Summary

This firmware performs vector control of permanent magnet motors using encoders.

1.2 Function

The firmware implements the following main functions:

- Encoder-based vector control of permanent magnet motors (9 axes)
- Communication with RZ/T2H Motion Control Utility

2. Software configuration

2.1 Basic specifications

Table 2-1 shows basic specifications.

Table 2-1 Basic specifications

Item	Description
Control method	Vector control
Rotar position detection	Absolute encoder
Input voltage	DC 24 [V]
PWM frequency	e^2 studio：20 [kHz]
	EWARM：10 [kHz]
Dead time	1 [us]
Current control period	e^2 studio：50 [us]
	EWARM：100 [us]
Speed control period	100[us]
Position control period	100[us]
Speed command range	CW：0 [rpm] ～ 3000 [rpm]
	CCW：0 [rpm] ～ 3000 [rpm]
Position command range	-2147483647 [ec] ～ 2147483647 [ec]
Position resolution	0.0055 [degree] (2^{16})
Rotate direction	CW direction: Position [ec] decreases CCW direction: Position [ec] increases
Compiler optimization	e^2 studio：Optimize (-O1)
	EWARM：Low
Protection function	- POEG - Interlock process

2.2 Operating environment

Table 2-2 shows operating environment. Table 2-3 shows the development tool. Table 2-4 shows hardware configuration.

Table 2-2 Operating environment

Integrated Development Environment (IDE)	e^2 studio	IAR Embedded Workbench for Arm
IDE version	2024-10	9.60.2 + patch (EWARM patch for RZ/T2H Rev.1.0)
FSP version	2.2.0	FSP Smart Configurator 2024-10
Toolchain version	GNU Arm Embedded 12.2.1.arm-12-24	-
In Circuit Emulator (ICE)	J-Link OB	IAR I-jet

Table 2-3 Development tool

Tool name	Tool Version
RZ/T2H Motion Control Utility	1.0.0.0

Table 2-4 Hardware configuration

Equipment		Model name
RZ/T2H Evaluation Board		RTK9RZT2H0CW1000BJ
	MPU	R9A09G077M44GBG 729-pin FCBGA, RAM 2[MB]
	On-board memory	OctaFlash: 64[MB]
	Operating frequency	Cortex-R52 CPU0: 1000[MHz] Cortex-R52 CPU1 and Cortex-A55 are unused.
	Operating voltage	DC 15[V]/3[A], 24[V]/3[A]
	Operating mode	xSPI0 boot mode (x1 boot serial flash)
Bus Board		RTK0EM0000Z03000BJ
Inverter Board		RTK0EM0000B15010BJ
	Operating voltage	DC 24[V]
Motor /Encoder (manufactured by TAMAGAWA SEIKI)		TSM3101N2001E020 /TS5669N124 (FA-CODER®)

2.3 File configuration

Table 2-5 shows the file configuration.

Table 2-5 File configuration

Item	Description
rzt_cfg	Configuration files for FSP
rzt_gen	Generated files
rzt\arm	Files related to CMSIS
rzt\board	Files related to evaluation boards
rzt\fsp	Files related to FSP
cg_src\r_cg_scifa.c	SCI driver
cg_src\r_cg_systeminit.c	Initialization process
inc\apl	Header files for source files under src/apl
src\apl\m_commands.c	Command process
src\apl\m_commutation.c	Vector control
src\apl\m_control.c	Motor control
src\apl\m_interlocks.c	Interlock process
src\apl\m_interpreter.c	Command determination process
src\apl\m_phasing.c	Phasing process
src\apl\m_pid_calc.c	PID calculation process
src\apl\m_pos_read.c	Position acquisition process
src\apl\m_recorder.c	Data recording process
src\drv\dsm	DSMIF driver
src\drv\m_rzt.c	Driver-related process
src\hal_entry.c	Main process
src\encoder\FACoder	FACoder driver

3. Firmware details

3.1 Initialization process

The initialization process is completed after the device reset operation. Table 3-1 shows functions related to the initialization process.

Table 3-1 Initialization process

Function	Description
R_Systeminit	R_Systeminit function initializes peripherals. The peripherals to be initialized are as follows. ・GPT ・ELC ・DSMIF ・POEG ・SCI ・XSPI Input: None Output: None
m_startup	m_startup function performs the following initialization: - Initialization of t_motor type variables - Lighting of LEDs on the board - Initialization of absolute encoder Input: None Output: None
m_Restore	m_Restore function reads the motor parameters from the flash memory. The read parameter is set in a member variable of the t_motor type structure. Input: (t_console *) pc, (t_motor *) pm Output: None

3.2 Main process

The main process is defined as m_background function. The m_background function is called in while loop in hal_entry.c. Table 3-2 shows functions related to the main process.

Table 3-2 Main process

Function	Description
m_interpreter	The m_interpreter function executes commands received from the RZ/T2H Motion Control Utility. Input: (t_console *) pc Output: None
m_rec_begin	The m_rec_begin function records data for the Motion Scope of the RZ/T2H Motion Control Utility. Input: None Output: None
interlocks	The interlocks function stops the PWM output when the motor's position deviation, velocity, and current values exceed the limit values. Input: (t_motor *) pm Output: None

3.3 Periodic interruption process

The periodic interruption process does encoder-based vector control. The process is implemented in m_background function in m_control.c. Table 3-3 shows functions related to the periodic interruption process.

Table 3-3 Periodic interruption process

Function	Description
bootstrap_charge	The bootstrap_charge function waits and gets offset value for DSMIF. Input: (t_motor *) pm Output: None
pos_read	The pos_read function gets the position from the absolute encoder. The pos_read function is implemented in the file m_pos_read.c. Input: (t_motor *) pm Output: None
pos_loop	The pos_loop function does position control. The pos_loop function is implemented in the file m_control.c. Input: (t_motor *) pm Output: None
vel_loop	The vel_loop function does speed control. The vel_loop function is implemented in the file m_control.c. Input: (t_motor *) pm Output: None
crnt_read	The crnt_read function takes the value of the DSMIF and converts it to a current value. The crnt_read function is implemented in the file m_rzt.c. Input: (t_motor *) pm Output: None
crnt_loop	The crnt_loop function performs vector control from position and current values. The crnt_loop function is implemented in the file m_control.c. Input: (t_motor *) pm Output: None
m_recorder	The m_recorder function acquires data for the Motion Scope function of the RZ/T2H Motion Control Utility. The m_recorder function is implemented in the file m_recorder.c. Input: None Output: None

3.4 Communication process

The communication process receives commands from Motion Control Utility and transmits the result of command execution. Table 3-4 shows functions related to the communication process.

Table 3-4 Communication process

Function	Description
m_rx_interrupt	The m_rx_interrupt function is called when a receive buffer full interrupt occurs. It reads data from the received data register, manipulates the member variables of the t_console type variable, and decodes the received command. The m_rx_interrupt function is implemented in m_rzt.c file. Input: (t_console *) pc Ouput: None
m_tx_interrupt	The m_tx_interrupt function is called when a transmitted data empty interrupt occurs. It writes the data of the t_console variable to the transmit data register. The m_tx_interrupt function is implemented in m_rzt.c file. Input: (t_console *) pc Output: None

3.5 Data Types

Table 3-5 shows main data types.

Table 3-5 Data types

Data type	Description
t_motor	This structure contains the motor position, speed, current value, gain, limit value, address of the compare match register, etc.
t_motor_pars	This structure contains parameters that are stored in flash memory.
t_console	This structure contains variables used for communication with the RZ/T2H Motion Control Utility, the address of the SCI register, etc.
t_command	This structure contains a table for associating variables and functions with ASCII commands sent from the RZ/T2H Motion Control Utility.
t_trace	This structure contains variables related to the Motion Scope function of the RZ/T2H Motion Control Utility.

3.6 Global Variables

Table 3-6 shows the main global variables.

Table 3-6 Global variables

Data structure instance	Description
t_motor g_st_m[MOTOR_NUM]	t_motor array (number of elements: MOTOR_NUM)
t_console con2	These variables are dedicated for each of the communication interfaces providing serial connection to a host computer.
long g_tick	This variable is incremented every time slice. It is used to coordinate the operation between real-time tasks and the main loop.
short g_suspend	This flag is intended for temporarily preventing the real-time functions from executing. Its purpose is to enable time-sensitive operations such as writing flash memory from being affected by the real-time functions.
t_command Commands[]	This is an array of command data structures where all host commands are defined. They consist of ASCII name, type and pointer to either function that executes the command or variable that holds the referenced parameter.

3.7 Enumerations

Table 3-7 shows the main enumerations.

Table 3-7 Enumerations

Enumeration	Description
ETYPE	Defines the different encoder types supported
CommutationModes	Defines the different operations of the current loop algorithm
PacketCode	Defines the type of the packet received
PacketError	Defines the possible errors reported by the packet protocol interpreter

4. Motor control process

4.1 9-axes motor control

The m_background function executes the bootstrap_charge, pos_read, pos_loop, vel_loop, crnt_read, and crnt_loop functions for the number of times specified by MOTOR_NUM. MOTOR_NUM is a #define directive that determines the number of motor to be controlled. MOTOR_NUM is defined in m_common.h. Figure 4-1 shows a flowchart of 9-axes motor control.

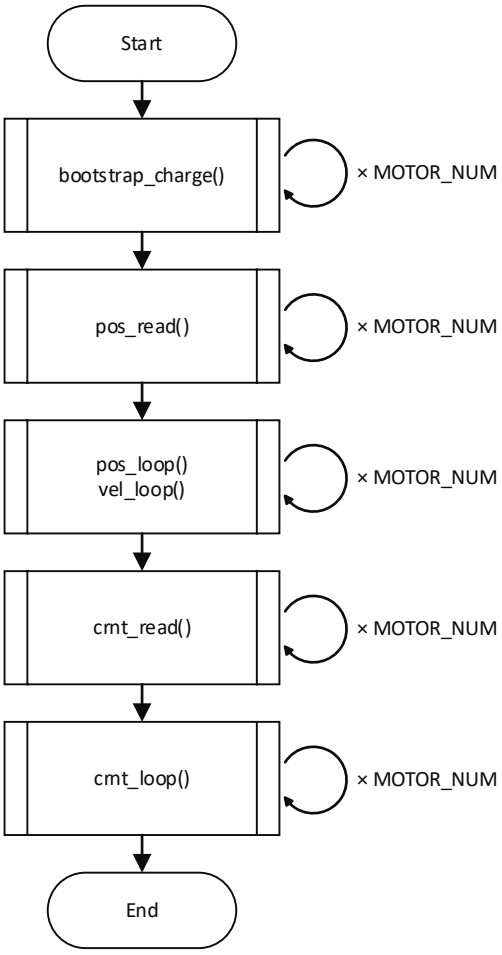

Figure 4-1 9-axes motor control

4.2 m_background function

4.2.1 bootstrap_charge function

The bootstrap_charge function waits and gets offset value of DSMIF.

4.2.2 pos_read function

The pos_read function gets the position of the absolute encoder.

Table 4-1 shows t_motor member variables related to pos_read function.

Table 4-1 t_motor member variables related to pos_read function

Variable	Description
encoder_type	This variable specifies the type of encoder. Since this firmware uses FACODER as an absolute encoder, the encoder_type is initialized to ETYPE_APE_FACODER.

4.2.3 pos_loop function

The pos_loop function gets position error from reference position(cmd_pos64) and current position(crnt_pos64), calls pid_calc_pos64 function and gets speed reference(in_vel64) to input into speed controller.

Table 4-2 shows t_motor member variables related to pos_loop function.

Table 4-2 t_motor member variables related to pos_loop function

Variable	Description
crnt_kp	Proportional Gain (Position Control)
crnt_ki	Integral Gain (Position Control)
crnt_kd	Differential Gain (Position Control)
integral_limit	Integral limit value (Position Control)
crnt_kvff	Speed feedforward gain (Position Control)
crnt_kaff	Acceleration feedforward gain (Position Control)
crnt_bias	Output bias value (Position Control)
cmd_vel	Reference speed
cmd_acc	Reference acceleration
pos_loop_limit	Output limit value (Position Control)
pos_error2	Position error
derivative_err2	Position error derivative
integral_err2	Position error integral

The KP, KI, KD, VFF, and AFF commands of the ASCII communication protocol commands in the Motion Control Utility change the buff_kp, buff_ki, buff_kd, buff_kvff, and buff_kaff. When update_ctrl function is executed, the values of these variables are assigned to crnt_kp, crnt_ki, crnt_kd, crnt_kvff, and crnt_kaff.

4.2.4 vel_loop function

The vel_loop function calculates speed error from reference speed(in_vel64) and current speed(crnt_vel64), call pid_calc_vel64 function and gets q-axis reference value(output_q) to input into current controller.

Table 4-3 shows t_motor member variables related to vel_loop function.

Table 4-3 t_motor member variables related to vel_loop function

Variable	Description
crnt_kp_vel	Proportional Gain (Speed Control)
crnt_ki_vel	Integral Gain (Speed Control)
crnt_kd_vel	Differential Gain (Speed Control)
integral_limit_vel	Integral limit value (Speed Control)
PiOut_limit_vel	Output limit value (Speed Control)
vel_error2	Speed error
derivative_err2_vel	Speed error derivative
integral_err2_vel	Speed error integral
output_q	q-axis current [mA]

The VKP, VKI, and VKD commands of the ASCII communication protocol commands in the Motion Control Utility change the buff_kp_vel, buff_ki_vel, and buff_kd_vel. When update_ctrl function is executed, the values of these variables are assigned to the crnt_kp, crnt_ki, and crnt_kd.

4.2.5 crnt_read function

The crnt_read function gets the value of DSMIF and convert it into U, V and W phase current.

Table 4-4 shows t_motor member variables related to crnt_read function.

Table 4-4 t_motor member variables related to crnt_read function

Variable	Description
adc1_raw	DSMIF CH0 conversion value for each unit
adc2_raw	DSMIF CH1 conversion value for each unit
adc3_raw	DSMIF CH2 conversion value for each unit
adc_iu	U phase current [mA]
adc_iv	V phase current [mA]
adc_iw	W phase current [mA]
crnt_volt	Bus board voltage [V]
total_current	Sum of absolute values of U phase current and V phase current [mA]

4.2.6 crnt_loop function

The crnt_loop function executes the processes below.

1. Electrical angle calculation
2. commutate_foc function
3. commutate_svm function
4. update_pwm funciton

In addition to the process, the crnt_loop function executes the phasing process when the “aligning” variable is one.

4.2.6.1 Phasing process

The phasing process passes current from U phase to the V and W phases and aligns the d-axis of the rotor with u-axis.

Table 4-5 shows t_motor member variables related to phasing process.

Table 4-5 t_motor member variables related to phasing process

Variable	Description
aligning	Flags for phasing process
phasing_mode_crnt	Phasing mode variable This value is initialized as PIM_FORCED to call forced_phasing function.
phasing_time	Phasing process time
phasing_power	Phasing process duty ration
phasing_origin	Encoder value after phasing process

4.2.6.2 Electrical angle calculation

This process gets electrical angle(phase_angle) from current position(crnt_pos) and the positon(phase_origin) where d-axis aligns with u-axis and then calculate sin and cos value from angle_rad.

Table 4-6 shows t_motor member variables related to electrical angle calculation.

Table 4-6 t_motor member variables related to electrical angle calculation

Variable	Description
phase_angle	Electrical angel [ec]
angle_rad	Rotor position [rad]
counts2rad	Coefficient value to convert electrical angle [ec] into radians.
angle_sin	Sin function value calculated from angle_rad
angle_cos	Cos function value calculated from angle_rad

4.2.6.3 commutate_foc function

The commutate_foc function converts U and V phase current into alpha and beta axis current. Then, it gets d and q axis current from alpha and beta axis current. Next, the function executes PI calculations to get d and q axis voltage. Finally, it converts d and q axis voltage into alpha and beta axis voltage.

Table 4-7 shows t_motor member variables related to commutate_foc function.

Table 4-7 t_motor member variables related to commutate_foc function

Variable	Description
p_iu	The pointer variable to adc_iu that has U phase current value
p_iv	The pointer variable to adc_iv that has V phase current value
foc_id	d-axis current [mA]
foc_iq	q-axis current [mA]
foc_id_err	d-axis current error [mA]
foc_iq_err	q-axis current error [mA]
foc_id_err_int	d-axis current error integral [mA]
foc_iq_err_int	q-axis current error integral [mA]
output_d	d-axis current reference value [mA]
output_q	q-axis current reference value [mA]
foc_kp	Proportional Gain (Current Control)
foc_ki	Integral Gain (Current Control)
foc_vd	d-axis voltage
foc_vq	q-axis voltage
foc_alpha	α-axis voltage
foc_beta	β-axis voltage

4.2.6.4 commutate_svm function

The commutate_svm function converts alpha and beta axis voltage into three phases voltage and performs space vector modulation.

4.2.6.5 update_pwm function

The update_pwm function updates duty ratios of U, V and W phase PWM signals.

5. Interlock process

The interlock process monitors the variables related to motor control and stops PWM signal output when it detects the abnormal value or state. This process is called by every 1ms as the main process.

Table 5-1 shows the interlock function.

Table 5-1 Interlock process

Function	Description
Inverter Over Current	This function sets the 26 bits of the status (ErrSts) when the total current (total_current) exceeds the overcurrent threshold (tc_limit) for a certain period (tc_limit_time) or more continuously. Initial tc_limit value：TC_LIMIT__VAL_DFLT Initial tc_limit_timer value：TC_LIMIT_TIME_VAL_DFLT
Position Error	This function sets the 17 bits of the status (ErrSts) when the position error (pos_error) exceeds the position error threshold (pos_error_limit) for a certain period (pos_error_timer) or more. Initial pos_error_limit value：POS_ERROR_LIMIT_VAL_DFLT Initial pos_error_time value： POS_ERROR_LIMIT_TIME_VAL_DFLT
Bus board Under Voltage	This function sets 28 bits of the status (ErrSts) when the bus board voltage (crnt_volt) falls below the low voltage threshold (Lvolt_Val). Initial Lvolt_Val value：LVOLT_VAL_DFLT
Bus Board Over Voltage	This function sets 27 bits of the status (ErrSts) when the bus board voltage (crnt_volt) exceeds the overvoltage threshold (Hvolt_Val). Initial Hvolt_Val value：HVOLT_VAL_DFLT
Inverter Fault (POEG)	This function sets the 25 bits of the status (ErrSts) when the PIDF bit of the POEG0GD0 of POEG0 reaches 1.
Overload Pre-detect	This function sets 21 bits of the status (ErrSts) when the total current (total_current) exceeds the threshold value (Ovc_Val). Initial Ovc_Val value：OVC_VAL_DFLT
Over Speed	This function sets 20 bits of the status (ErrSts) when the speed (crnt_vel) exceeds the threshold (Ovs_Val). Initial Ovs_Val value：OVS_VAL_DFLT
Instructed Speed Difference	This function sets 19 bits of the status (ErrSts) when the velocity error (vel_error) exceeds the threshold value (WOvs_Val) for 5 consecutive seconds. Initial WOvs_Val value：WOVS_VAL_DFLT
Maximum Limit Position Minimum Limit Position	This function sets 15 bits of the status (ErrSts) when the position reaches the upper threshold (WPosMax_Val). When the position is less than or equal to the lower threshold of position (WPosMin_Val), set the 13 bits of the status (ErrSts). Initial WPosMax_Val value：WPOSMAX_VAL_DFLT Initial WPosMin_Val value：WPOSMIN_VAL_DFLT This function is disabled by default.

6. Data Recording

The Data Recording functions are intended to enable the analysis of the system behavior in real time. It is an indispensable tool for analyzing the performance of the different control loops, their configuration parameters and their efficiency in application specific contest. The data recorder stores up to four user defined parameters in buffers during system operation. The support functions and configuration parameters enable the selection of rate of recording, which variables are recorded, when the recording is started and when it is supposed to end.

The length of the recording as number of samples is defined by the macro-TRACE_BUFFER_SIZE. By default, it is set to 512. This value can be increased when the application requires longer records and the RAM memory is available. All buffers are combined into a single array named traceData[], The data type of the array is short – 16-bit integer. For this reason, when a 32-bit variable is being recorded, its value is split between the fist and the fourth buffers. When the data is reported, it is combined appropriately.

The host gets the data using the RVAL command. Table 6-1 shows the mapping between the code of the RVAL command and data variables.

Table 6-1 codes and data variables

Codes	Data variables
0	crnt_pos
1	crnt_vel
2	crnt_acc
3	I2t_integral
4 - 7	Reserved
8	pos_error
9	output_q
10	Reserved
11	foc_id
12	foc_iq
13	foc_id_err
14	foc_iq_err
15	adc1_raw
16	adc2_raw
17	pvt_points
18	foc_vd
19	foc_vq
20	g_counter
21	phase_angle
22	adc3_raw
23	captured_pos
24	pos_error2
25	Integral_err2
26	vel_error2
27	Integral_err2_vel
28	foc_id_err_int

29	foc_iq_err_int
30	est_trq
31	angle_rad

The start and stop conditions of the data recorder are configured by the ASCII command TRACE (variable trace.Trigger). The table below describes the possible settings representing different trigger conditions:

Table 6-2 trigger conditions

TRACE codes	Trigger Start Condition	Trigger Stop Condition
0	N/A	Stop data recording
1	Start recording immediately.	Stop when the motion is completed. This trigger is useful for examining the end of a motion.
2	Start recording immediately.	Stop when the buffer is full. This trigger is useful for examining the beginning of a motion.
3	Start recording on start of motion.	Stop at the end of motion.
4	Start recording immediately.	Stop on input change. The input bit mask is defined in the trace.Level variable.
5	Start recording immediately.	Stop on value exceeding the threshold defined in trace.Level variable.
6	Start recording immediately.	Stop on value below the threshold defined in trace.Level variable.
7	Start on PWM output change.	Stop when the buffer is full.
8	Start recording on input change. Input mask is defined in the trace.Level variable.	Stop when the buffer is full.
9	Start on value exceeding the threshold defined in trace.Level variable.	Stop when the buffer is full.
10	Start on value below the threshold defined in trace.Level variable.	Stop when the buffer is full.

The recorder operates synchronously to the real-time task that executes every 50us. The rate of the recorder can be expressed as multiples of this time interval. The multiple factor is set by the ASCII command TRATE and stored in the variable trace.RateMult. For example, if the desired rate of the data recorder is 1ms then the TRATE should be set to 20.
Another variable evaluated during some of the trigger conditions is the ASCII command TLEVEL (variable trace.Level). The value of this variable is the threshold that the recorded variable is compared against. Depending on the trigger code, the condition to start recording can test for value either bigger or smaller than the threshold.
The function invoked periodically to test the start trigger condition is m_rec_begin().
The function invoked periodically to test the stop trigger condition and perform the recording is m_recorder().
The recorder mode of operation is reported by the ASCII command TMODE (variable trace.Mode). The meaning of the codes stored is described in the table below:

Table 6-3 TMODE codes

TMODE codes	Modes of operation / status
0	Idle, stops recording if started
1	Recorder is armed – ready to start on beginning of motion
2	Recording in ongoing. Stop once the buffer is full.
3	Recording in circular buffer. Stop once the motion completes.

Once the data recording is completed, the host can use the command PLAY to get content of the recording buffers. The function implementing this request is **m_Play()**. It also formats binary packet response if the invocation context is packet command handler.

7. ASCII Communication Protocol

The ASCII protocol is based on commands consisting of ASCII characters terminated with Carriage Return (CR, ASCII 13). The controller responds with an optional data string followed by a prompt.

The ASCII protocols uses the following communication parameters:

115,200 bps, 8 data bits, 1 stop bit, no parity

When the command is accepted the prompt consist of CR, Line Feed (LF, ASCII 10) and Greater Than sign (>). When the command is rejected the prompt has Question Mark (?) instead of the Greater Than sign.

Example:

POS ; Host command terminated by CR

120 ; Reply Data String

> ; Reply Prompt

The variable names entered at the command prompt report the value of the referenced variable. If a parameter follows the name it is interpreted as a request to set the variable to a new value. Some variables are read-only. An attempt to set a value to them will be reported as invalid command. Examples:

>POS ; Request value of the variable POS

2100

>POS 2000 ; Set POS to a new value

>POS ; Report the new value

2000

>

The full set of ASCII commands is described in the next table.

Table 7-1 ASCII commands

Command	Size	R/W	Description
STA	2H	R	Status word short act_state
ERR	2	R	Position error [ec] short pos_error
ADC1	2	R	U phase current (A/D conversion value) short adc1_raw
ADC2	2	R	V phase current (A/D conversion value) short adc2_raw
TC	2U	R	Sum of absolute values of U phase current and V phase current [mA] unsigned short total_current
CV	4	R	Current speed [rpm] long crnt_vel
VEL	4	R/W	Reference speed [ec/cycle time] long buffMotion.velocity
ACC	4	R/W	Reference acceleration [ec/cycle time^2] long buffMotion.acceleration
DEC	4	R/W	Reference deceleration [ec/cycle time^2] long buffMotion.deceleration
PRO	2U	R/W	Velocity profile mode (fixed value: 0) short dflt_vgp_mode
KP	2U	R/W	Propotional gain (position control) short buff_kp

Command	Size	R/W	Description
KI	2U	R/W	Integral gain (position control) short buff_ki
KD	2U	R/W	Differential gain (position control) short buff_kd
IL	2U	R/W	Integral limit value (position control) short integral_limit16
VFF	2U	R/W	Speed feedforward gain (position control) short buff_kvff
AFF	2U	R/W	Acceleration feedforward gain (position control) short buff_kaff
MAX	2U	R/W	Position error limit [ec] short buff_err_limit
ETIME	2U	R/W	Position error detection time short pos_error_time
DS	2U	R/W	Data skip interval (fixed value: 0) short crnt_ds
MLIMIT	2U	R/W	Output limit value (position control) long pos_loop_limit
BIAS	2	R/W	Output bias value (position control) short crnt_bias
ASTOP	2U	R/W	Auto stop mode (fixed value: 0) short auto_stop_mode
PIMODE	2U	R/W	Phasing mode (fixed value: 0) short phasing_mode
PITIME	2U	R/W	Phasing time short phasing_time
PIOUT	2U	R/W	Phasing output short phasing_power
PMAP	2U	R/W	PWM output phase selection (fixed value: 0) short phase_config
PORIGIN	4	R	The origin of phase [ec] long phase_origin
PCMODE	2U	R/W	Commutation mode (fixed value: 4) short commutation_mode
PPAIRS	2U	R/W	The number of pole pair (fixed value: 5) short pole_pairs
PCOUNTS	4	R/W	Encoder counts per rotation in electrical angle [ec] long ec_per_ecycle
ECPR	4	R/W	Encoder counts per rotation in mechanical angle [ec] long ec_per_rev
PANGLE	2	R	Electrical angle [ec] short phase_angle
CLIMIT	2U	R/W (Func)	Over current detect threshold [mA] m_CurrentLimit() unsigned short tc_limit
CTIME	2U	R/W	Over current detect time unsigned short tc_limit_time
IDM	2	R	d axis current [mA] short foc_id
IQM	2	R	q axis current[mA] short foc_iq

Command	Size	R/W	Description
IQERR	2	R	q axis current error[mA] short foc_iq_err
QKP	2U	R/W	Propositonal gain (current control) short foc_kp
QKI	2U	R/W	Integral gain (current control) short foc_ki
ECP	4	R	Positon reference value when position error is detected [ec] long mecmd_pos
ECV	4	R	Speed reference value when position error is detected long mecmd_vel
EPO	4	R	Current position when position error is detected [ec] long mecrnt_pos
U	2	R/W	U phase PWM output unsigned short PhaseU
V	2	R/W	V phase PWM output unsigned short PhaseV
W	2	R/W	W phase PWM output unsigned short PhaseW
PHASES	2U	R/W	Motor type (fixed value: 3) short motor_type
TMODE	2U	R/W	Data recording mode short trace.Mode
TRATE	2U	R/W	Data recording rate short trace.RateMult
TLEVEL	4	R/W	Data recording threshold float trace.Level
ABS	4	R/W (Func)	Absolute reference position [ec] m_Abs() long buffMotion.position
REL	4	R/W (Func)	Relative reference position [ec] m_Rel() long buffMotion.position_rel
POS	4	R/W (Func)	Current position [ec] m_Position() volatile long crnt_pos
IND	4	R (Func)	Index position [ec] m_Index() volatile long index_pos
GO	4	W (Func)	m_Go() long buffMotion.position
FWD	0	W (Func)	Rotate forward m_Forward()
REV	0	W (Func)	Rotate backward m_Reverse()
ON	0	W (Func)	Enable servo control m_ServoOn()
OFF	0	W (Func)	Disable servo control m_ServoOff()
ENABLE	0	W (Func)	Enable PWM output m_PowerOn()

Command	Size	R/W	Description
DISABLE	0	W (Func)	Disable PWM output m_PowerOff()
STOP	0	W (Func)	Stop rotation in reference deceleration m_SmoothStop()
ABORT	0	W (Func)	Stop rotation in maximum deceleration m_AbruptStop()
ALIGN	0	W (Func)	Execute the phasing process m_AlignPhase()
VER	string	R (Func)	Firmware version m_Version() const char *s_version
PWM	2	R/W (Func)	Reference q axis current [mA] m_PosLoopCmd() short pos_loop_cmd
IQCMD	2	R/W (Func)	Reference q axis current [mA] m_OutputIQ() short output_q
IDCMD	2	R/W (Func)	Reference d axis current [mA] m_OutputID() short output_d
CH1	2	R/W (Func)	Data channel 1 m_LogChannel0()
CH2	2	R/W (Func)	Data channel 2 m_LogChannel1()
CH3	2	R/W (Func)	Data channel 3 m_LogChannel2()
CH4	2	R/W (Func)	Data channel 4 m_LogChannel3()
TRACE	2	R/W (Func)	Data recording start/stop m_Trace() short trace.Trigger
PLAY	4 x 4	R (Func)	Data acquisition m_Play()
PINVERT	2	R/W (Func)	Invert sign of current position (fixed value: 0) m_PosInvert() short pos_inv_mode
SAVE	0	W (Func)	Save motor parameters into the flash memory (This command is not supported for EWARM.) m_Save()
RESTORE	0	W (Func)	Restore motor parameters from the flash memory (This command is not supported for EWARM.) m_Restore()
ETYPE	2	R/W (Func)	Encoder type (fixed value: 3) m_EncoderType() short encoder_type
EBAUDRATE	4	R/W (Func)	Encoder baudrate (fixed value: 2500 [kHz]) m_EncBaudrate() long enc_baudrate
ESTATUS	2H	R (Func)	Encoder status m_EncStatus() unsigned short enc_status

Command	Size	R/W	Description
TBSIZE	2U	R	Data recording size short trace.buff_size
QKD	2U	R/W	Differential gain (current control) short foc_kd
VKP	2U	R/W	Proposional gain (speed control) short buff_kp_vel
VKI	2U	R/W	Integral gain (speed control) short buff_ki_vel
VKD	2U	R/W	Diffrential gain (speed control) short buff_kd_vel
ELVOLT	4	R/W (Func)	Low voltage detection threshold [V] m_Elvolt() long Lvolt_Val
EHVOLT	4	R/W (Func)	Over voltage detection threshold [V] m_Ehvolt() long Hvolt_Val
EWPOSMIN	4	R/W (Func)	Minimum value of position error detection [ec] m_EwposMin() long WPosMin_Val
EWPOSMAX	4	R/W (Func)	Maximum value of position error detection [ec] m_EwposMax() long WPosMax_Val
EOVS	4	R/W (Func)	Over speed detection threshold [rpm] m_Eovs() long Ovs_Val
EWOVS	4	R/W (Func)	Speed error detection threshold [ec/cycle time] m_Ewovs() long WOvs_Val
ERRMASK	4H	R/W (Func)	Error status mask m_Emask() unsigned long ErrMsk
EVOLT	2	R	Bus board voltage [V] long crnt_volt
EQUERY	4H	R	Error status unsigned long ErrSts
ERESET	0	W (Func)	Clear error status m_Ereset() unsigned long ErrSts
EOVC	4	R/W (Func)	Over current predetect threshold [mA] m_Eovc() long Ovc_Val
CTRLMODE	2U	R/W (Func)	Control mode 0: Position control 1: Speed controls m_CtrlMode() unsigned short ctrl_mode
COMDIR	2U	R/W (Func)	Rotation direction (CW/CCW) m_CommandDirection() unsigned short cmd_dir
COMVEL	2	R/W (Func)	Reference speed [rpm] m_CommandVelocity() short cmd_vel_rpm

Command	Size	R/W	Description
RVAL	2, 4, 8, 4H	R/W (Func)	Read value m_ReadValue()

8. Reference documents

- RZ/T2H Startup Manual (RZ/T2H Motion Control Utility) (R01AN7334)
- RZ/T2H Program Writing Guide (R01AN7335)
- e^2 studio Integrated Development Environment User's Manual: Getting Started (R20UT4535)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	2024.11.26	-	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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

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/.