RL78/F24

Three-Shunt Sensorless Vector Control for PMSM by MCU

R01AN6309EJ0110
Rev.1.10
2023.06.30

Introduction

This application note intends to describe a sample program for operating the sensorless vector control of PMSM (Permanent Magnet Synchronous Motor) by using the features on the RL78/F24 product.

The sample program is only for your reference and Renesas Electronics Corporation never guarantee the operations.

Please use this sample program after carrying out a thorough evaluation in a suitable environment.

Target Device

Operations of the sample program are checked by using the following device.

- RL78/F24 (R7F124FGJ)

Contents

1. Overview

This document describes a sample program that uses the features of RL78/F24 product to drive a PMSM motor (Permanent Magnet Synchronous Motor) with sensorless vector control.

1.1 Usage of the System

This system (Sample program) realizes a sensorless vector control by using the following hardware.

- RTK7F124FGS00000BJ : Inverter Board with RL78/F24 Microcontroller
- Brushless DC motor : TG-55N-KA

Remark

- TG-55N-KA : Brushless DC motor (Made by Tsukasa Electric CO., LTD.) <https://www.tsukasa-d.co.jp/>

1.2 Development Environment

Table 1-1 and Table 1-2 show the software and hardware development environment which were used for the sample program on this application note.

Table 1-1. Software Development Environment

Renesas CS+		
	IDE Version	CS+ for CC E8.07.00 [01 Dec 2021]
	Compiler	CC-RL E1.11.00
IAR		
	IDE Version	IAR Embedded Workbench IDE 8.5.2.7561 (8.5.2.7561)
	Compiler	IAR C/C++ Compiler for Renesas RL78 4.21.3.2447 (4.21.3.2447)

Table 1-2. Hardware Development Environment

On-chip Debugging Emulator	E2 emulator Lite
	E2 emulator
MCU Part Name	RL78/F24 (R7F124FGJ)
Inverter Board	RTK7F124FGS00000BJ
BLDC Motor	PMSM (TG-55N-KA)

2. System Overview

2.1 Hardware Configuration

The hardware configuration is shown in Figure 2-1.

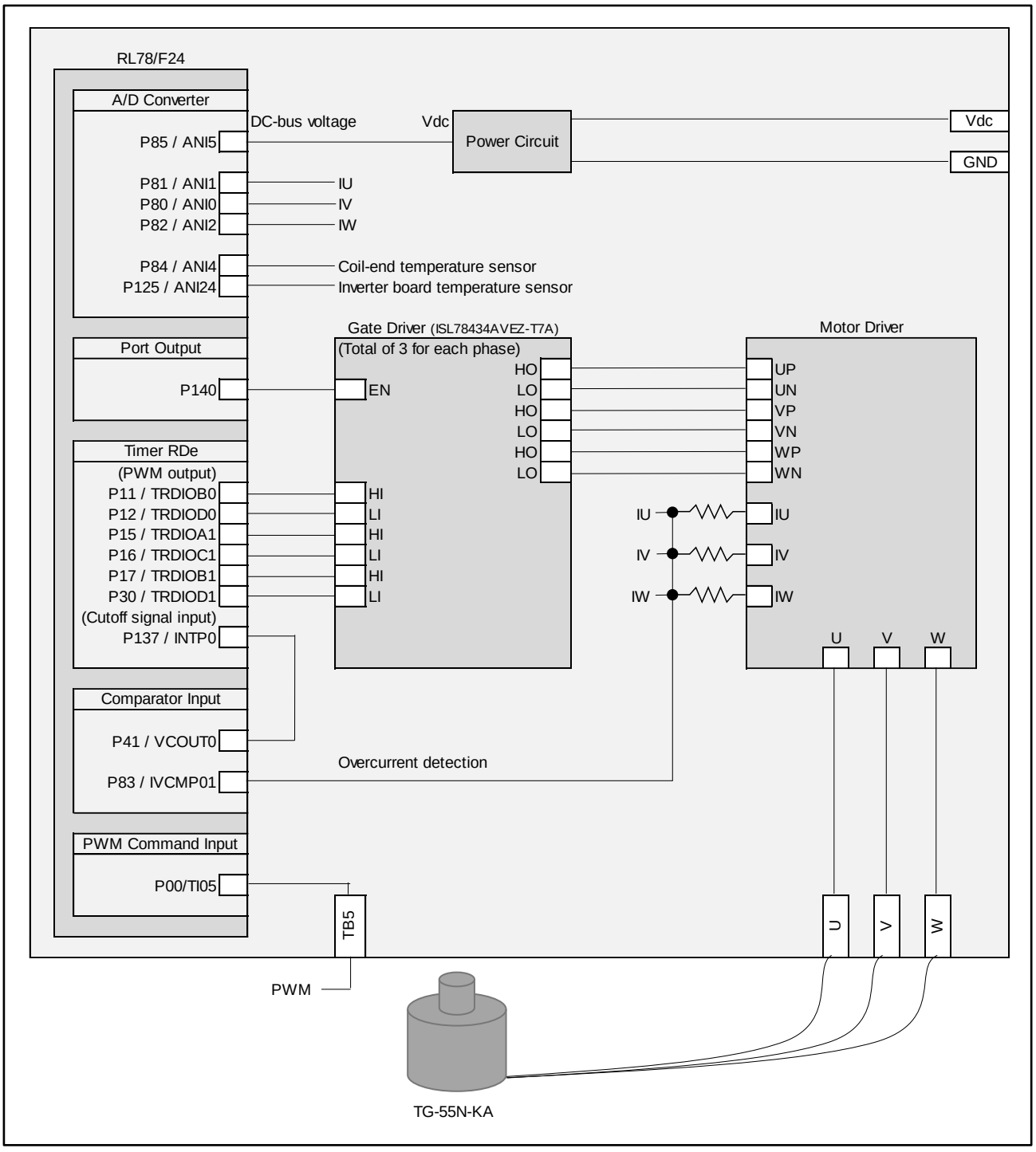


Figure 2-1. Hardware Configuration Diagram

2.2 Hardware Specifications

2.2.1 Hardware Interface

Table 2-1 shows the RL78/F24 pin interface list on this system.

Table 2-1. Pin Interface List

Pin name	Function
P85 / ANI5	Voltage measurement for V_{dc}
P81 / ANI1	Current measurement for U-phase
P80 / ANI0	Current measurement for V-phase
P82 / ANI2	Current measurement for W-phase
P84 / ANI4	Temperature measurement on the motor coil-end
P125 / ANI24	Temperature measurement on the PCB
P140	Control signal output (Enable gate driver)
P11 / TRDIOB0	Motor control output for U-phase (Normal phase)
P12 / TRDIOD0	Motor control output for U-phase (Counter phase)
P15 / TRDIOA1	Motor control output for V-phase (Normal phase)
P16 / TRDIOC1	Motor control output for V-phase (Counter phase)
P17 / TRDIOB1	Motor control output for W-phase (Normal phase)
P30 / TRDIOD1	Motor control output for W-phase (Counter phase)
P137 / INTP0	External interrupt input for overcurrent detection
P41 / VCOUT0	Comparator output for overcurrent detection
P83 / IVCMP01	Comparator input for overcurrent detection
P00 / TI05	Input of PWM duty command value

2.2.2 Peripheral Functions

Table 2-2 shows a list of peripheral functions used in this system.

Table 2-2. Peripheral Function List

Peripheral function	Purpose
A/D converter	• Voltage measurement (Monitoring V_{dc} Voltage) • Motor current measurement (Acquisition of the current values flow into each 3-shunt resistors) • Temperature measurement on the motor coil-end • Temperature measurement on the PCB
Timer Array Unit	• 1 ms interval timer • PWM duty command value input
Timer RDe	• PWM output using Extended Complementary PWM mode (Normal phase: 3 ch, Counter phase: 3 ch) • PWM output forced cutoff (using PWMOPA)
D/A converter	Generating threshold voltage for internal comparator
Comparator	For overcurrent detection
Application Accelerator Unit	• Clarke / Park transformation and PI control for motor operation • Inverse Park (I-Park) and Inverse Clarke (I-Clarke) transformation

2.2.2.1 A/D Converter

The A/D converter converts the analog input voltage to the digital value. Target MCU (RL78/F24: 48 pins product) has 1 unit of 12-bit A/D converter. Up to 19 channels of analog input voltage can be converted to digital values by controlling the analog input channels.

Table 2-3 and Table 2-4 show how to use the A/D converter in this system.

Table 2-3. A/D Converter Usage List (A/D start trigger: INTTM01)

Channel	Measurement items	Physical quantity per bit of A/D converted value
ANI4	Motor coil-end temperature	Refer to the Table 2-6.
ANI24	Temperature on the PCB	Refer to the Table 2-7.

Remark INTTM01: Timer Array Unit 01 interrupt request

Table 2-4. A/D Converter Usage List (A/D start trigger: INTTRD1)

Channel	Measurement items	Physical quantity per bit of A/D converted value
ANI5	V_{dc} Power supply voltage	65.0 [V] / 4095 = 0.0159 [V]
ANI1	U-phase current	50.0 [A] / 4095 = 0.0122 [A]
ANI0	V-phase current	
ANI2	W-phase current	

Remark INTTRD1: Timer RDe carrier interrupt request (Interrupt request signal has been decimated.)

Table 2-5. List of A/D Conversion Targets

Measurement target	Variable name	Channel
V_{dc} Power supply voltage	–	ANI5
U-phase current	focCurrentIu	ANI1
V-phase current	focCurrentIv	ANI0
W-phase current	focCurrentIw	ANI2
Temperature on the motor coil-end	userThCoilEnd	ANI4
Temperature on the PCB	userThOnBoard	ANI24

In case of a temperature measurement, the temperature is converted based on the table calculated in advance from the thermistor and circuit resistance. The voltage conversion table for the motor coil-end and the PCB is shown below.

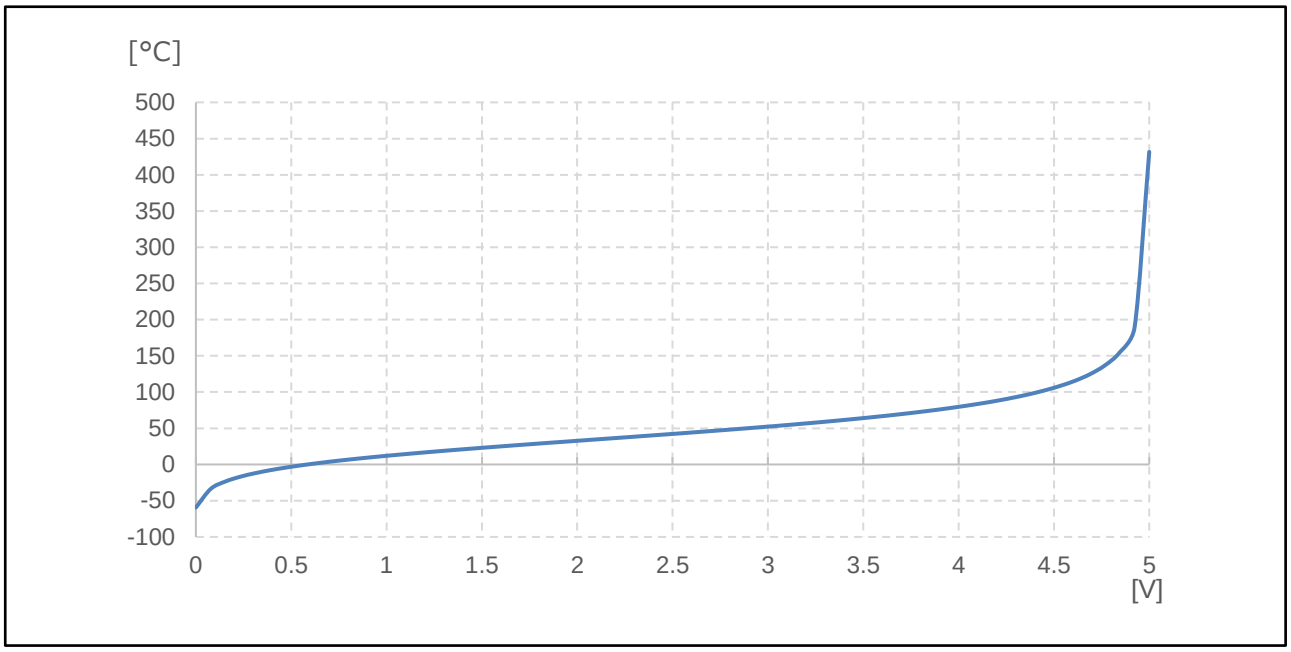


Figure 2-2. A/D Terminal Voltage vs. Temperature Graph by Coil-End Temperature Measurement

Table 2-6. A/D Terminal Voltage vs. Temperature Table by Coil-End Temperature Measurement

Voltage	Temp.	Voltage	Temp.	Voltage	Temp.	Voltage	Temp.
0.000	-59.393	1.328	19.469	2.657	45.264	3.985	79.031
0.078	-33.636	1.407	21.111	2.735	46.818	4.063	82.049
0.156	-23.353	1.485	22.716	2.813	48.396	4.142	85.326
0.234	-16.808	1.563	24.289	2.891	50.002	4.220	88.917
0.313	-11.870	1.641	25.836	2.969	51.641	4.298	92.896
0.391	-7.838	1.719	27.362	3.048	53.317	4.376	97.367
0.469	-4.391	1.797	28.870	3.126	55.035	4.454	102.478
0.547	-1.354	1.875	30.364	3.204	56.801	4.532	108.450
0.625	1.381	1.954	31.849	3.282	58.620	4.611	115.640
0.703	3.884	2.032	33.326	3.360	60.501	4.689	124.664
0.781	6.205	2.110	34.799	3.438	62.450	4.767	136.738
0.860	8.378	2.188	36.272	3.516	64.477	4.845	154.778
0.938	10.430	2.266	37.748	3.595	66.592	4.923	189.159
1.016	12.382	2.344	39.228	3.673	68.808	5.000	431.619
1.094	14.250	2.422	40.717	3.751	71.141		
1.172	16.047	2.501	42.217	3.829	73.607		
1.250	17.783	2.579	43.732	3.907	76.228		

Remarks The units of the table are as follows.
Voltage: [V], Temperature: [°C]

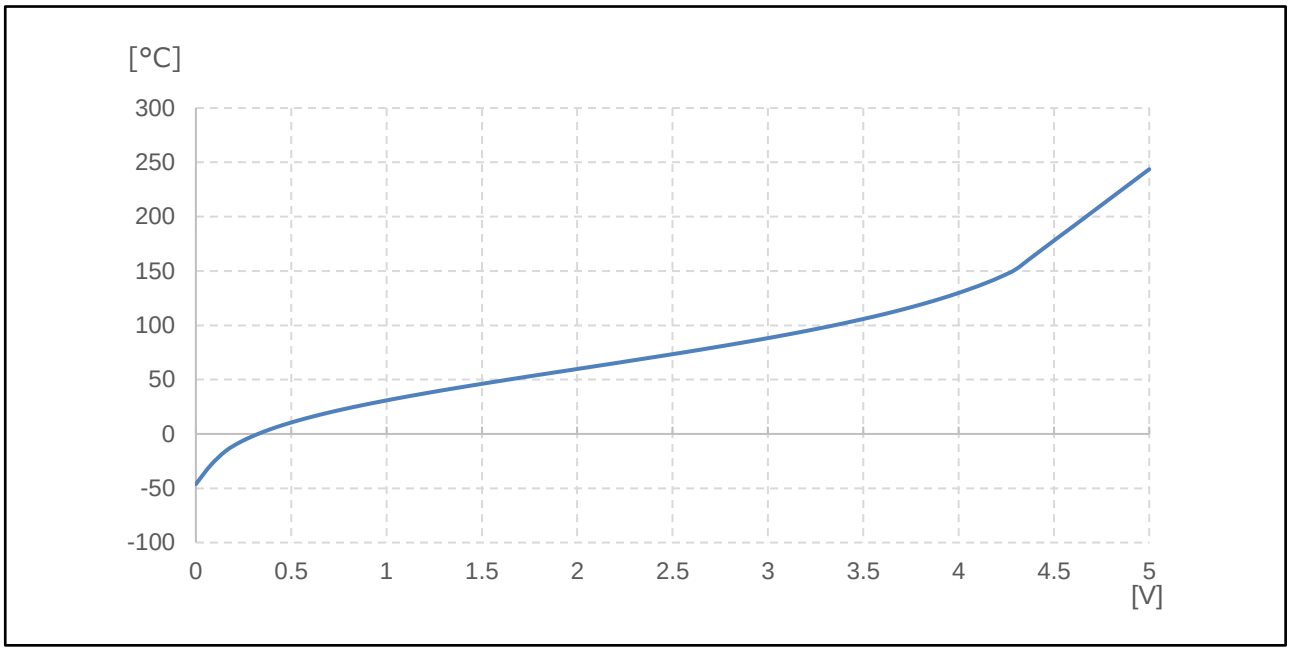


Figure 2-3. A/D Terminal Voltage vs. Temperature Graph by PCB Temperature Measurement

Table 2-7. A/D Input Voltage vs. Temperature Table by PCB Temperature Measurement

Voltage	Temp.	Voltage	Temp.	Voltage	Temp.	Voltage	Temp.
0.000	-46.154	1.328	41.174	2.657	77.864	3.985	128.909
0.078	-28.744	1.407	43.453	2.735	80.135	4.063	133.649
0.156	-15.757	1.485	45.688	2.813	82.45	4.142	138.83
0.234	-7.357	1.563	47.887	2.891	84.814	4.220	144.548
0.313	-0.95	1.641	50.055	2.969	87.233	4.298	151.285
0.391	4.326	1.719	52.2	3.048	89.716	4.376	161.566
0.469	8.869	1.797	54.326	3.126	92.27	4.454	171.848
0.547	12.898	1.875	56.44	3.204	94.905	4.532	182.129
0.625	16.546	1.954	58.545	3.282	97.629	4.611	192.41
0.703	19.903	2.032	60.647	3.360	100.456	4.689	202.691
0.781	23.029	2.110	62.75	3.438	103.397	4.767	212.973
0.860	25.969	2.188	64.858	3.516	106.468	4.845	223.254
0.938	28.758	2.266	66.975	3.595	109.688	4.923	233.535
1.016	31.42	2.344	69.107	3.673	113.076	5.000	243.656
1.094	33.978	2.422	71.257	3.751	116.659		
1.172	36.447	2.501	73.43	3.829	120.465		
1.250	38.842	2.579	75.631	3.907	124.533		

Remarks The units of the table are as follows.
Voltage: [V], Temperature: [°C]

2.2.2.2 Timer Array Unit

Timer Array Unit (TAU) consists of eight 16-bit timers. Each 16-bit timer called ‘Channel’ and can be used as an independent timer as well as an advanced timer function by combining multiple channels. The target microcontroller (RL78/F24) incorporates two units (16 channels in total).

This system uses TAU as shown on Table 2-8.

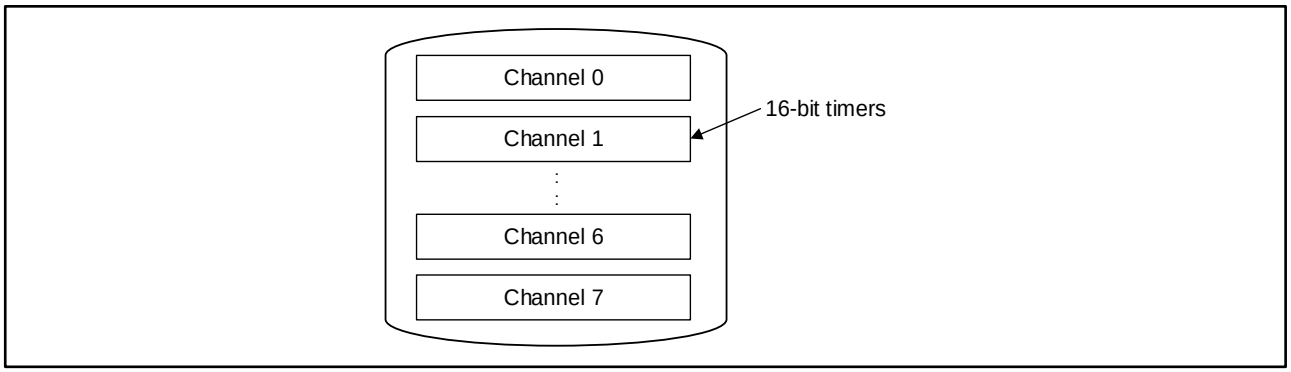


Figure 2-4. Timer Array Unit

Table 2-8. Usage of Timer Array Unit

Unit	CH	Item	Content	Purpose
0	1	Timer operation mode	Interval timer	1 [ms] interval time generation.
		Source clock	CK00	
		Count clock frequency	40 [MHz]	
		Interrupt cycle	1 [ms]	
		Set value of timer data register 1 (TDR01)	39999 (1 [ms] / (1/40 [MHz]) - 1)	
	5	Timer operation mode	Measurement of high-/low-level width of input signal	PWM cycle measurement
		Source clock	CK01	
		Count clock frequency	625 [kHz]	
		Interrupt cycle	At the detection of either falling or rising edges	
		Set value of timer data register 5 (TDR05)	–	
		Remark	Refer to 4.1.1.2 (1) for detail.	
	6	Timer operation mode	One count function	PWM cycle measurement timeout generation
		Source clock	CK01	
		Count clock frequency	625 [kHz]	
		Interrupt cycle	100 [ms]	
		Set value of timer data register 6 (TDR06)	62499 (100 [ms] / (1/625 [kHz]) - 1)	
		Remark	Refer to 4.1.1.2 (2) for detail.	

2.2.2.3 Timer RDe

Timer RDe contains two 16-bit timer units (timer RD0 and timer RD1). And it supports 6 operation modes as below.

- Timer mode
- Reset synchronous PWM mode
- Complementary PWM mode
- PWM3 mode
- Extended PWM mode
- Extended complementary PWM mode

This system uses Timer RDe as shown on Table 2-9.

Table 2-9. Timer RDe Settings

Timer	Item	Contents	Purpose
Timer RDe	Operation mode	Extended complementary PWM mode (Symmetry three-phase waveforms)	Three-phase complementary PWM output
	PWM cycle	50 [μs]	
	Dead time	1.0 [μs]	
	Count frequency	80 [MHz]	
	Output level	Initial output level: Low level Active output level: High level	
	Buffer operation	Yes	
	Pulse output Forced Cutoff	Enable. (Using PWM option unit A) (Output state during cutoff: High impedance)	

Caution. The extended complementary PWM mode outputs a three-phase complementary PWM waveform by combining the counters of Timer RD0 and Timer RD1 and registers.

Figure 2-5 shows an example of the PWM output waveform in the extended complementary PWM mode.

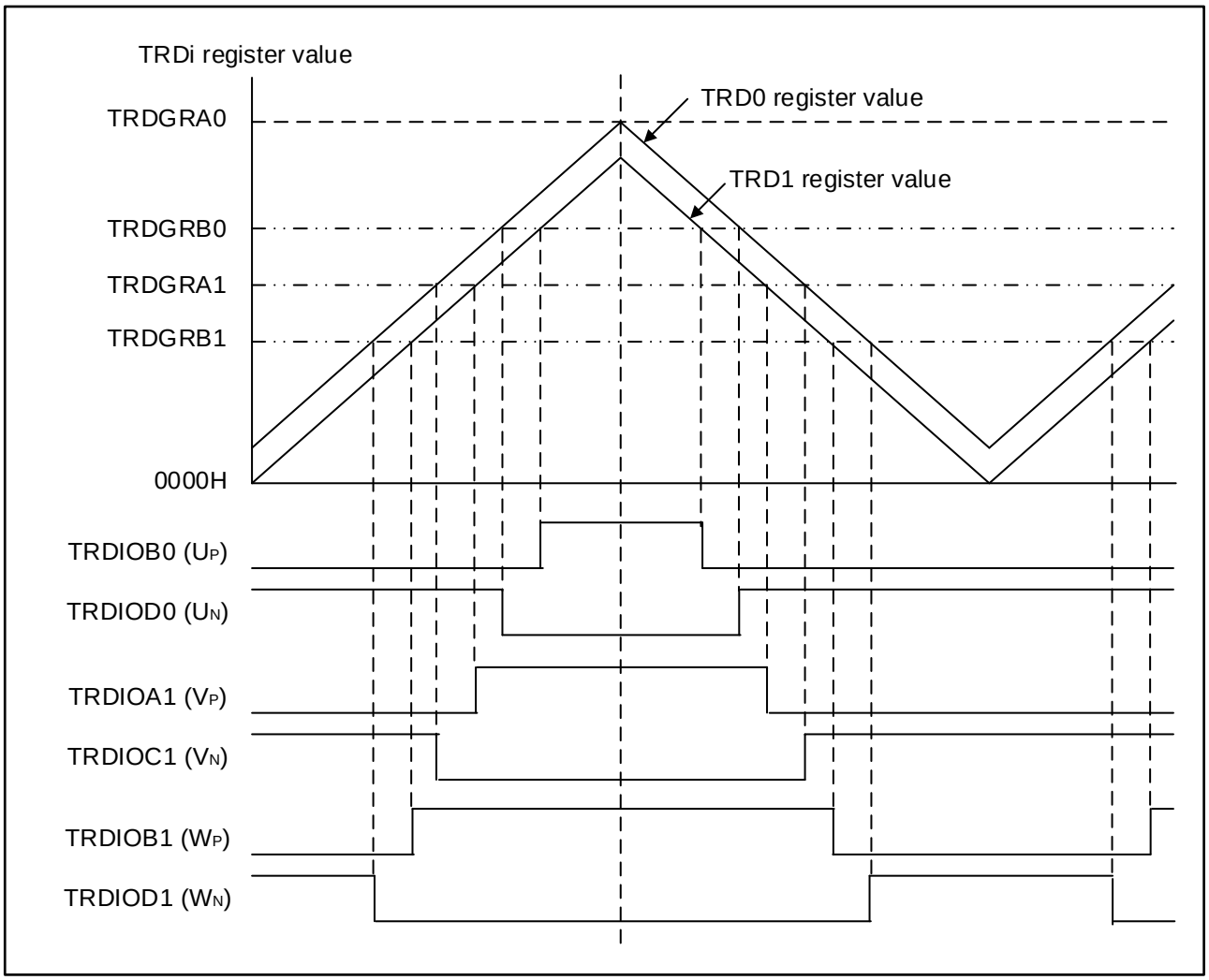


Figure 2-5. Example of the PWM Output Waveform in the Extended Complementary PWM Mode

PWM duty setting from PWM duty command value:

Describes how to set the duty in the extended complementary PWM mode (symmetrical waveform output).

Calculates the normal phase active level width from the normalized PWM duty command value and the PWM cycle count value. And, calculates the setting values for the buffer registers TRDGRD0, TRDGRC1, and TRDGRD1.

Normal phase active level width = PWM duty command value × PWM cycle count value
TRDGRD0, TRDGRC1, TRDGRD1 = PWM cycle count value – Normal phase active level width

2.2.2.4 Application Accelerator Unit (AAU)

Application Accelerator Unit (AAU) is the dedicated arithmetic assist hardware to reduce the software load for FOC (Field Oriented Control) algorithm processing on BLDC motor control and PI (Proportional Integral) algorithm processing on DC/DC converter control.

This sample program uses AAU and executes the following calculation.

Table 2-10. AAU Function Which Is Used in the Sample Program

Algorithm Mode	Operation overview
Clarke and Park transformation and PI control for motor operation	Executes Clarke / Park transformation (3-phase to 2-phase conversion, Rotating coordinate transformation) and PI control calculation.
I-Park and I-Clarke transformation	Inverse Park and Inverse Clarke transformation (Rotating coordinate transformation, 2-phase to 3-phase conversion)

(1) Clarke and Park transformation and PI control for motor operation

In this mode, Clarke / Park transformation and PI control can be performed continuously.

- Clarke transformation : Converts from 3-phase current (I_U, I_V, I_W) to 2-phase current (I_α, I_β).
- Park transformation : Converts from fixed coordinates to rotational coordinates.
- PI control : P control (Proportional control) and I control (Integration control) are performed, and 2-phase output current (I_d, I_q) is calculated.

(2) Inverse Park (I-Park) and Inverse Clarke (I-Clarke) transformation

- I-Park transformation : Converts from rotational coordinate to fixed coordinate.
- I-Clarke transformation : Converts from 2-phase voltage (V_α, V_β) to 3-phase voltage (V_U, V_V, V_W).

2.3 Software Configuration

2.3.1 File Configuration

Table 2-11 shows the file configuration of the sample program.

Table 2-11. File Configuration of the Sample Program (1/2)

RL78F24_SSL_FOC180_V105		
prj	RL78F24_SSL_FOC180.mtpj	CS+ Project File (for CS+)
	cstart.asm	Start-up Routine Definition File (for CC-RL)
	monitor.asm	Section Definition File for Debug Monitor (for CC-RL)
	OPT_BYTE.asm	Device Option Byte Definition File (for CC-RL)
	security_id.asm	Security ID Definition File (for CC-RL)
	stkinit.asm	Stack Area Initialization File (for CC-RL)
	iodefine.h	Register Definition File (for CC-RL)
	RL78F24_SSL_FOC180.dep	IAR workspace files (for IAR)
	RL78F24_SSL_FOC180.ewd	
	RL78F24_SSL_FOC180.ewp	
	RL78F24_SSL_FOC180.ewt	
	RL78F24_SSL_FOC180.eww	
inc	control_math.h	Header file for controlling calculation processing
	control_parameter.h	Header file for motor control parameter definition
	error_management.h	Header file for error management processing
	foc_current_ref_ctrl.h	Header file for speed control and current command value control in vector control
	foc_current_regulator.h	Header file for current control in vector control
	foc_math.h	Header file for vector control calculation
	motor_parameter.h	Header file for motor parameter definition
	mtr_speed_ref_ctrl.h	Header file for speed command value control
	rl78_common.h	Header file for device dependent common definition
	rl78_interrupt.h	Header file for interrupt processing definition
	rssk_inv.h	Header file for inverter dependent processing
	sequence.h	Header file for system state management
	thermistor.h	Header file for thermistor table definition
	user_control.h	Header file for user definition processing
lib	r_fixed_point_math.h	Header file for fixed-point arithmetic library
	r_foc_angle_est.h	Header file for position estimation library
	r_scale_adjust.h	Header file for scaling macro library
lib\CC-RL	r_fixed_point_math.lib	Fixed point arithmetic library
	r_foc_angle_est.lib	Position estimation library

Table 2-11. File Configuration of the Sample Program (2/2)

RL78F24_SSL_FOC180_V105		
src	contrl_math.c	Control arithmetic processing unit
	error_management.c	Error management processing unit
	foc_current_ref_ctrl.c	Speed control and current command value control in vector control
	foc_current_regulator.c	Current control in vector control
	foc_math.c	Vector control arithmetic unit
	mtr_speed_ref_ctrl.c	Speed command value control unit
	rl78_interrupt.c	Interrupt processing unit
	rssk_inv.c	Inverter dependent processing unit
	sequence.c	System state management unit
	user_control.c	User defined processing unit
	mcu_debug_option_for_iar.c	Option Byte setting for IAR complier (for IAR)
src\drv	r_cg_aau.c	AAU driver implementation unit
	r_cg_aau.h	Header file for AAU driver implementation
	r_cg_adc.c	A/D converter driver implementation unit
	r_cg_adc.h	Header file for A/D converter driver implementation
	r_cg_adc_user.c	A/D converter driver user defined implementation unit
	r_cg_cgc.c	Clock generator driver implementation unit
	r_cg_cgc.h	Header file for Clock generator driver implementation
	r_cg_cgc_user.c	Clock generator driver user defined implementation unit
	r_cg_comp.c	Internal comparator driver unit
	r_cg_comp.h	Header file for Internal comparator driver implementation
	r_cg_comp_user.c	Internal comparator driver user defined implementation unit
	r_cg_dac.c	D/A converter driver implementation unit
	r_cg_dac.h	Header file for D/A converter driver implementation
	r_cg_dac_user.c	D/A converter driver user defined implementation unit
	r_cg_macrodriver.h	Header file for CG macro
	r_cg_port.c	Port I/O driver implementation unit
	r_cg_port.h	Header file for Port I/O driver implementation
	r_cg_port_user.c	Port I/O driver user defined implementation unit
	r_cg_timer.c	Timer driver implementation unit
	r_cg_timer.h	Header file for timer driver implementation
	r_cg_timer_user.c	Timer driver user defined implementation unit
	r_cg_timer_rde.c	Timer RDe driver implementation unit
	r_cg_timer_rde.h	Header file for Timer RDe driver implementation
	r_cg_timer_rde_user.c	Timer RDe driver user defined implementation unit
	r_cg_userdefine.h	User defined header file
	r_cg_wdt.c	WDT driver implementation unit
	r_cg_wdt.h	Header file for WDT driver implementation
	r_cg_wdt_user.c	WDT driver user defined implementation unit
	r_main.c	main function implementation unit
	r_systeminit.c	Hardware initialization definition

2.3.2 Module Configuration

Figure 2-6 shows the hierarchical structure of the sample program.

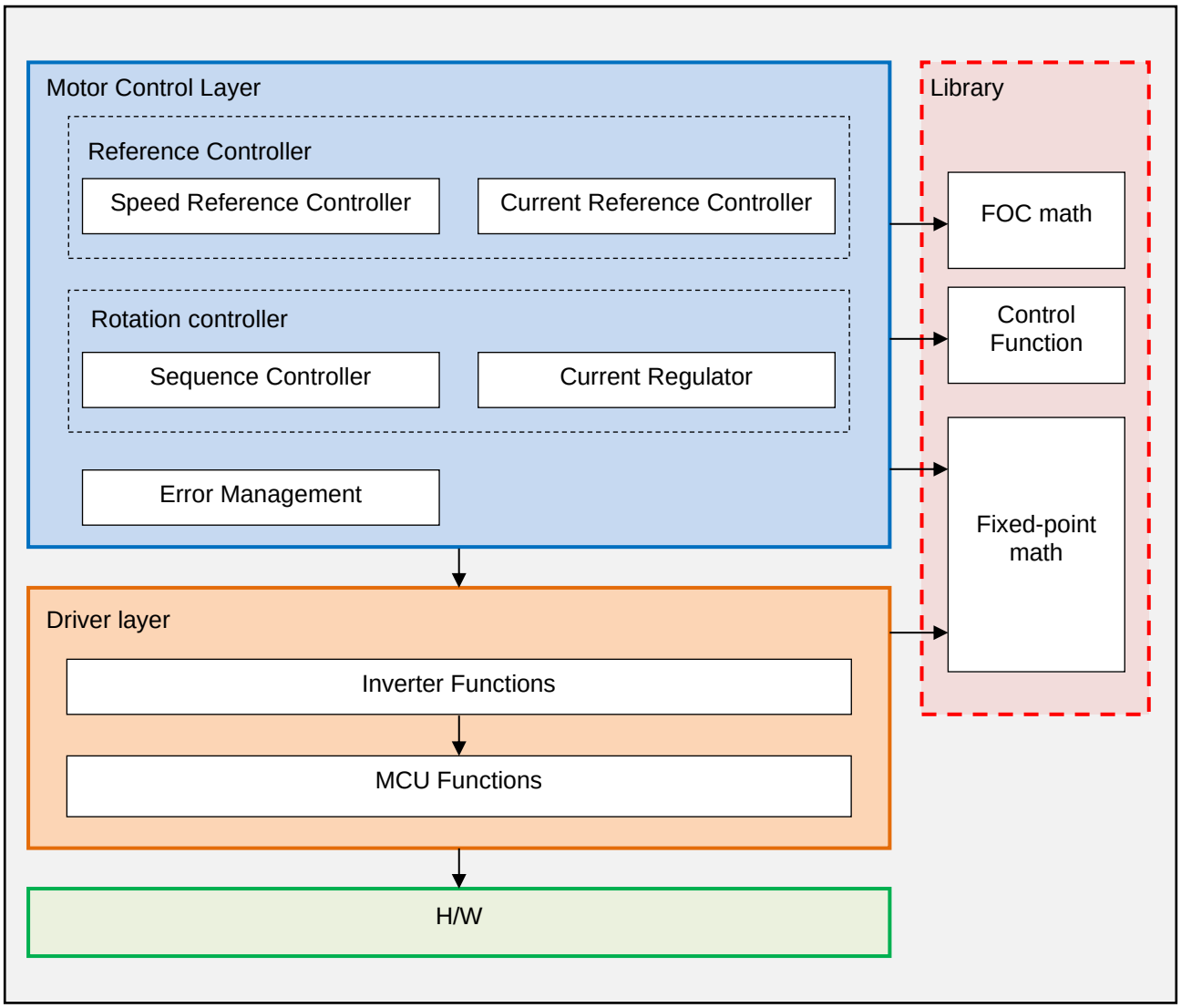


Figure 2-6. Sample Software Structure

2.4 Software Specifications

Table 2-12 shows the basic specifications of the software covered by this application note.

Table 2-12. Basic Specification of the Software

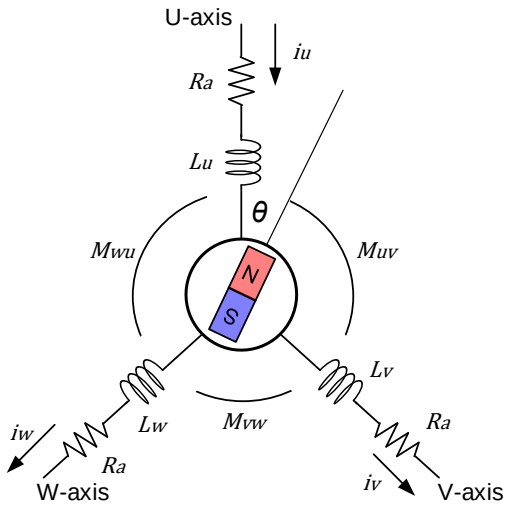
Item	Contents
Control method	Sensorless vector control with 3-shunt method
Motor rotation start/stop	Operates by selecting one of the following two condition. • After MCU reset released, the control rotation state is changed at the following timing. – After 3 [s], starts operation targeted on 1000 [rpm] as speed command value – After 10 [s] from above condition, speed command value changes to 2000 [rpm]. – After 10 [s] from above condition, speed command value changes to 3000 [rpm]. – After 10 [s] from above condition, stops the motor rotation. – After 1 [s] from above condition, reset the counter value. • After MCU reset released, controls rotation speed command value by PWM signal input. – Applicable PWM signal input range is from 10 [Hz] to 1000 [Hz]. – If duty ratio is 100%, rotation speed command value is 3000 [rpm]. – If duty ratio is 0%, rotation speed command value is 0 [rpm].
Rotor pole position detection method	Sensorless control based on current estimation error method
Carrier frequency (PWM)	20 [kHz]
Dead time	1 [µs]
Control cycle	100 [µs] (50 × 2)
Rotation speed control range	Both CW/CCW are supported from 500 [rpm] to 3000 [rpm].
Protection procedure	Deactivates the motor control signal outputs (6 ch), if any of the following conditions are met. • In case of exceeding +/- 16.97 [A] (10Arms + 20%) in any one of the three-phase current values. (Monitored in every 100 [µs]) • In case of the inverter power supply voltage (V_{dc}) exceeds 28.0 [V]. (Monitored in every 100 [µs].) • In case of the inverter power supply voltage (V_{dc}) is lower than 8.0 [V]. (Monitored in every 100 [µs].) • In case of the rotation speed exceed 5000 [rpm]. (Monitored in every 1 [ms]) • When low level signal input to INTP0. (Signal stop operation is real time. Error detection is monitored in every 100 [µs].) • In case of the PCB temperature exceeds 120 [°C]. (Monitored in every 1 [ms].) • In case of the motor coil-end temperature exceeds 180 [°C]. (Monitored in every 1 [ms].)

3. Motor Control Method

This chapter describes the vector control of the PMSM motor used in the sample program.

3.1 Voltage Equation of the Motor Control System

The voltage equation of a permanent magnet synchronous motor with a sinusoidal magnetic flux distribution can be expressed as shown in Figure 3-1.



$$\begin{pmatrix} v_u \\ v_v \\ v_w \end{pmatrix} = R_a \begin{pmatrix} i_u \\ i_v \\ i_w \end{pmatrix} + p \begin{pmatrix} \varphi_u \\ \varphi_v \\ \varphi_w \end{pmatrix}$$

$$\begin{pmatrix} \varphi_u \\ \varphi_v \\ \varphi_w \end{pmatrix} = \begin{pmatrix} L_u & M_{uv} & M_{wu} \\ M_{uv} & L_v & M_{vw} \\ M_{wu} & M_{vw} & L_w \end{pmatrix} \begin{pmatrix} i_u \\ i_v \\ i_w \end{pmatrix} + \Psi \begin{pmatrix} \cos\theta \\ \cos(\theta - 2\pi/3) \\ \cos(\theta + 2\pi/3) \end{pmatrix}$$

v_u, v_v, v_w: Each phase armature voltage
i_u, i_v, i_w: Each phase armature current
$\varphi_u, \varphi_v, \varphi_w$: Each phase armature interlinkage flux
L_u, L_v, L_w: Each phase self inductance
M_{uv}, M_{vw}, M_{wu}: Each phase mutual inductance
R_a: Each phase armature resistance
ψ: Maximum value of armature interlinkage flux depending on permanent magnet
θ: Lead angle of permanent magnet (rotor) from U-phase
p: Differential operator

Figure 3-1. Conceptual Diagram of the 3-phase Permanent Magnet Synchronous Motor

Here, self-inductance and mutual inductance are expressed as shown in the flowing formula.

$$\begin{cases} L_u = l_a + L_a - L_{as}\, cos(2\theta) \\ L_v = l_a + L_a - L_{as}\, cos(2\theta + 2\pi/3) \\ L_w = l_a + L_a - L_{as}\, cos(2\theta - 2\pi/3) \end{cases}$$

$$\begin{cases} M_{uv} = -L_a/2 - L_{as}\, cos(2\theta - 2\pi/3) \\ M_{vw} = -L_a/2 - L_{as} \cos(2\,\theta) \\ M_{wu} = -L_a/2 - L_{as}\, cos(2\,\theta + 2\pi/3) \end{cases}$$

l_a: Leakage inductance for one phase
L_a: Average value of effective inductance for one phase
L_{as}: Amplitude of effective inductance for one phase

3.2 Vector Control

The d-axis is set in the direction of the magnetic flux (N pole) of the permanent magnet and the q-axis is set in the direction which progresses by 90-degree from the d-axis. Then by using the following conversion matrix, coordinate conversion is performed.

$$C = \sqrt{\frac{2}{3}}\begin{pmatrix} \cos\theta & \cos(\theta - 2\pi/3) & \cos(\theta + 2\pi/3) \\ -\sin\theta & -\sin(\theta - 2\pi/3) & -\sin(\theta + 2\pi/3) \end{pmatrix}$$

$$\begin{pmatrix} v_d \\ v_q \end{pmatrix} = C\begin{pmatrix} v_u \\ v_v \\ v_w \end{pmatrix}$$

The voltage equation in the d-q coordinate system is obtained as follow.

$$\begin{pmatrix} v_d \\ v_q \end{pmatrix} = \begin{pmatrix} R_a + pL_d & -\omega L_q \\ \omega L_d & R_a + pL_q \end{pmatrix}\begin{pmatrix} i_d \\ i_q \end{pmatrix} + \begin{pmatrix} 0 \\ \omega\psi_a \end{pmatrix}$$

v_d, v_q: Each phase armature voltage

i_d, i_q: Each phase armature current

ψ_a: Value of armature interlinkage flux depending on permanent magnet

R_a: Each phase armature resistance

L_d, L_q: Each phase self inductance

$L_d = l_a + 3/2(L_a - L_{as}), \quad L_q = l_a + 3/2(L_a + L_{as})$

$\psi_a = \sqrt{3/2}\psi$

Based on this, it can be assumed that 3-phase alternating current is 2-phase direct current.

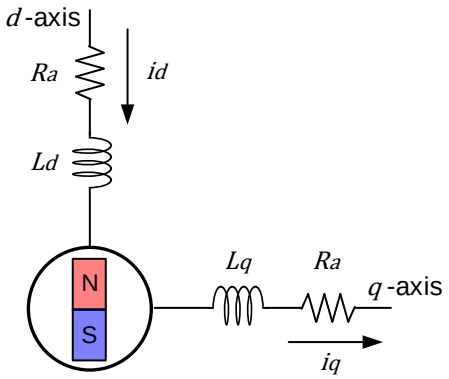


Figure 3-2. Conceptual Diagram of the 2-phase Direct Current Motor

Size of the torque generated in the motor can be obtained as follows from the exterior product of the electric current vector and armature interlinkage magnetic flux. The first term on the right side of this formula is called magnetic torque and the second term on the right side of this formula is called the reluctance torque.

$$T = P_n\{\psi_a i_q + (L_d - L_q) i_d i_q\}$$

T: Motor torque P_n: Number of pole pairs

The motor which has no difference between the rotor's main flux (d-axis) inductance and the main torque-producing (q-axis) inductance is defined as a motor which does not have saliency. In this case, as the reluctance torque is 0, the torque increases proportionally to the q-axis current. For that reason the q-axis current is called torque current. On the other hand, d-axis current is sometimes called excitation current, because it can be assumed that the d-axis current increase is the magnetic flux of permanent magnet decrease for q-axis voltage.

As SPMSM (Surface Permanent Magnet Synchronous Motor) generally does not have saliency, the d-axis current is unnecessary for generating torque. Instead, the q-axis current is controlled to adjust the speed. This is known as id = 0 control. On one hand, the motion equation of the motor in this case is expressed as follows. This equation shows that motor speed is increased by increasing the q-axis current.

$$I\frac{d\omega}{dt} = P_n \psi_a i_q - T_L$$

T_L: Load torque I: Motor moment of inertia

This system uses not motion equation but PI control for speed control. The q-axis current command value is calculated by the following formula.

$$i_q^* = (K_{P\omega} + \frac{K_{I\omega}}{s})(\omega^* - \omega)$$

$K_{P\omega}$: Speed PI proportional gain $K_{I\omega}$: Speed PI integral gain s: Laplace operator

To achieve early stabilization, the PI control is also used for the d-axis and q-axis current values. A command voltage value is acquired by current PI control.

$$v_d^* = (K_{Pi_d} + \frac{K_{Ii_d}}{s})(i_d^* - i_d)$$

K_{Pi_d}: d − axis current PI propotional gain K_{Ii_d}: d − axis current PI integral gain

$$v_q^* = (K_{Pi_q} + \frac{K_{Ii_q}}{s})(i_q^* - i_q)$$

K_{Pi_q}: q − axis current PI propotional gain K_{Ii_q}: q − axis current PI integral gain

Inductive voltage is generated when the motor is rotated. The effect on d-axis voltage due to q-axis current and on q-axis voltage due to d-axis current and magnetic flux of permanent magnet becomes significant along with the increase in speed. This d-axis and q-axis interference may delay the stability of a current value. To avoid it, the voltage of each axis is calculated by performing feed forward so that the interference term of each axis can be canceled beforehand.

$$v_d^* = (K_{Pi_d} + \frac{K_{Ii_d}}{s})(i_d^* - i_d) - \omega L_q i_q$$

$$v_q^* = (K_{Pi_q} + \frac{K_{Ii_q}}{s})(i_q^* - i_q) + \omega(L_d i_d + \psi_a)$$

This method to eliminate the effect of the interference term is known as decoupling control. This enables independent control of the d-axis and q-axis.

Vector control is a method by which the three-phase alternating current motor is converted to the 2-phase direct current motor that can be controlled each phase (d,q) independently while managing the position, speed and torque of the rotor.

Control flow of the vector control is shown below.

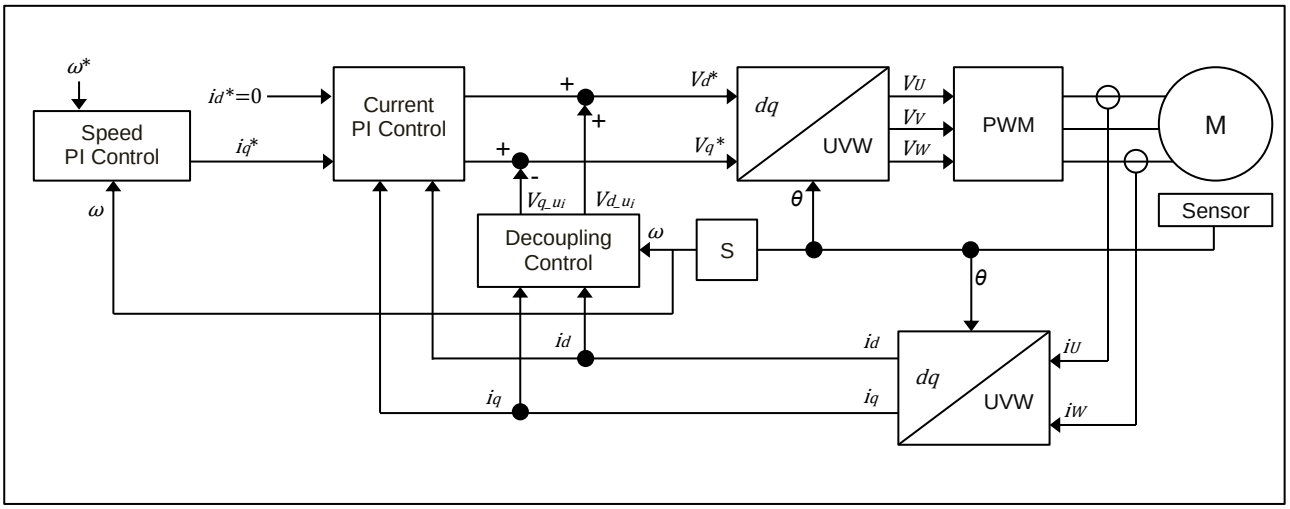


Figure 3-3. Control Flow of the Vector Control

3.3 Sensorless Vector Control Based on the Current Estimation Error

For the vector control, position sensors of the encoder and resolver etc. are required as voltage is set according to the rotor position. When the position sensors are not used, in other words, in the case of the sensorless vector control, it is necessary to estimate the position by some methods. These days, the demand for motor control by sensorless has increased and several methods are provided for estimating the position. This part introduces the sensorless vector control used in this system, which is using current estimation error.

Position of the d-axis is not clear as the position information of the actual motor is not available. As shown in the below figure, when γ-axis is set in the location which lags behind by $\Delta\theta$ from d-axis and δ-axis is set in the location 90-degree ahead of the γ-axis, the conversion formula from d/q-axis to the γ/δ-axis can be illustrated as Figure 3-4.

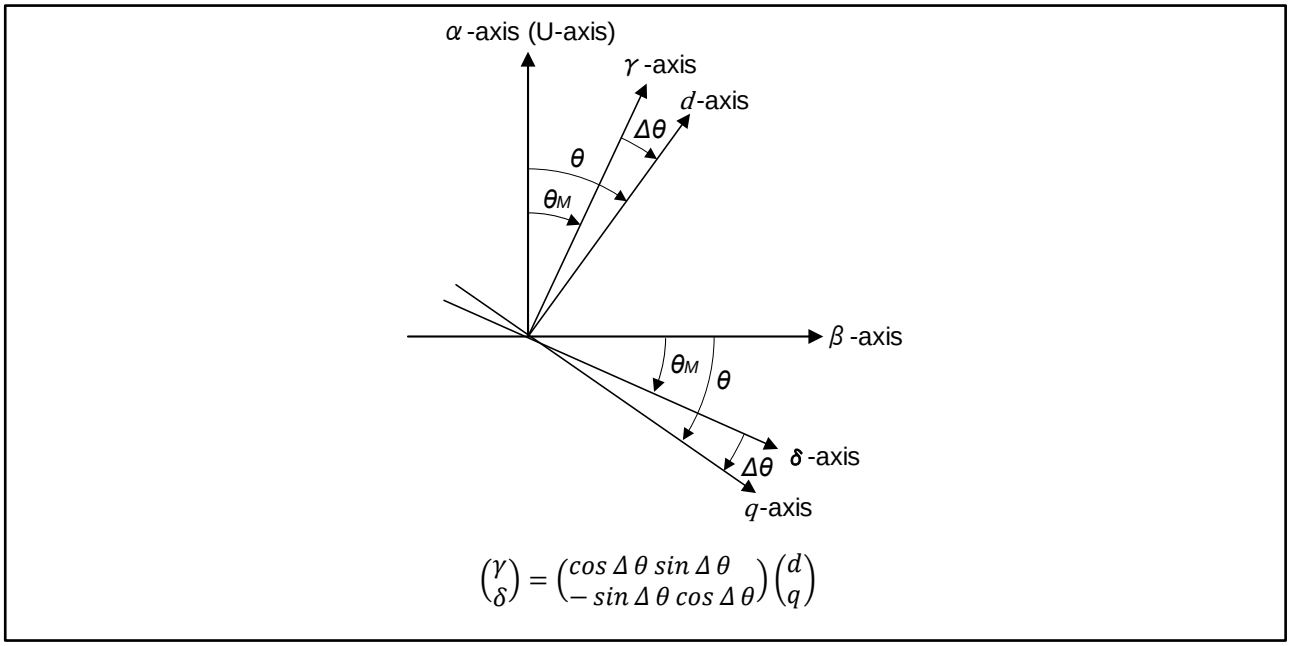


Figure 3-4. Relation Between d/q-axis and γ/δ-axis

The equation in which above is applied to the SPMSM voltage equation and written in the electric current state equation format is as follows.

$$p\begin{pmatrix}i_\gamma\\ i_\delta\end{pmatrix} = -\begin{pmatrix}\frac{R}{L} & -\dot{\theta}_M\\ \dot{\theta}_M & \frac{R}{L}\end{pmatrix}\begin{pmatrix}i_\gamma\\ i_\delta\end{pmatrix} + \frac{1}{L}\begin{pmatrix}v_\gamma\\ v_\delta\end{pmatrix} - \frac{K_E\dot{\theta}}{L}\begin{pmatrix}-\sin\Delta\theta\\ \cos\Delta\theta\end{pmatrix}$$

Discretization of this state equation by using backward differential approximation (Euler's approximation).

$$\begin{pmatrix}i_\gamma(n)\\ i_\delta(n)\end{pmatrix} = \begin{pmatrix}i_\gamma(n-1)\\ i_\delta(n-1)\end{pmatrix} + \frac{T}{L}\left\{\begin{pmatrix}v_\gamma(n-1)\\ v_\delta(n-1)\end{pmatrix} - R\begin{pmatrix}i_\gamma(n-1)\\ i_\delta(n-1)\end{pmatrix} - \dot{\theta}_M(n-1)L\begin{pmatrix}-i_\delta(n-1)\\ i_\gamma(n-1)\end{pmatrix} - e(n-1)\begin{pmatrix}-\sin\Delta\theta(n-1)\\ \cos\Delta\theta(n-1)\end{pmatrix}\right\}$$

$$\because\ e(n-1) = K_E\dot{\theta}(n-1)$$

As a motor model here, given that the motor parameters are written as R_M, L_M and e_M which are sufficiently equal to motor parameters of an actual motor and $\Delta\theta$ is set to 0, the current value at a sample point n can be represented as follows.

$$\begin{pmatrix} i_{\gamma M}(n) \\ i_{\delta M}(n) \end{pmatrix} = \begin{pmatrix} i_{\gamma}(n-1) \\ i_{\delta}(n-1) \end{pmatrix} + \frac{T}{L_M}\left\{ \begin{pmatrix} v_{\gamma}(n-1) \\ v_{\delta}(n-1) \end{pmatrix} - R_M \begin{pmatrix} i_{\gamma}(n-1) \\ i_{\delta}(n-1) \end{pmatrix} - \dot{\theta}_M(n-1) L_M \begin{pmatrix} -i_{\delta}(n-1) \\ i_{\gamma}(n-1) \end{pmatrix} - e_M(n-1) \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

Depending on the difference between actual motor current and motor model current, the current estimation error can be indicated as follows.

$$\begin{pmatrix} \Delta i_{\gamma}(n) \\ \Delta i_{\delta}(n) \end{pmatrix} = \frac{T}{L} \begin{pmatrix} e(n-1)\, sin\, \Delta\theta(n-1) \\ e_M(n-1) - e(n-1)\, cos\, \Delta\theta(n-1) \end{pmatrix}$$

When $\Delta\theta$ is sufficiently small, the current estimation error can be approximated as follows.

$$\begin{pmatrix} \Delta i_{\gamma}(n) \\ \Delta i_{\delta}(n) \end{pmatrix} \approx \frac{T}{L} \begin{pmatrix} e(n-1)\Delta\theta(n-1) \\ -\Delta e(n-1) \end{pmatrix}$$
$$\Delta e(n-1) = e(n-1) - e_M(n-1)$$

If both Δe and $\Delta\theta$ are 0, it can be considered that the actual model is synchronized with the motor model. e_M is estimated by feeding back $\Delta i\delta$ such that Δe becomes 0. Similarly, the θ_M value is estimated by feeding back $\Delta i\gamma$ such that $\Delta\theta$ becomes 0. The motor model is thus matched with the actual model.

The e_M estimation equation can be expressed as follows.

$$e_M(n) = e_M(n-1) - K_e \Delta i_{\delta}(n)$$

Here, Ke is the speed electromotive force gain. Similarly, the θ_M estimation equation can be written as follows.

$$\theta_M(n) = \theta_M(n-1) + \frac{T}{K_{EM}} e_M(n) + K_{\theta}\, sgn\{\dot{\theta}_M(n-1)\}\, \Delta i_{\gamma}(n)$$
$$sgn\{\dot{\theta}_M(n-1)\} = \begin{cases} 1; \dot{\theta}_M(n-1) \geq 0 \\ -1; \dot{\theta}_M(n-1) < 0 \end{cases}$$

Here, K_{EM} is the electromotive force coefficient of the motor model and K_{θ} is the position estimation gain. Also, $p\theta_M$ sign is used instead of the $p\theta$ sign.

$$\dot{\theta}_M = \frac{1}{T}\{\theta_M(n) - \theta_M(n-1)\} = \frac{e_M}{K_{EM}} + \Delta\dot{\theta}_M(n)$$
$$\Delta\dot{\theta}_M(n) = \frac{K_{\theta}}{T} sgn\{\dot{\theta}_M(n-1)\}\, \Delta i_{\gamma}(n)$$

In the control, LPF for the speed correction term is used as follows. Here, 0 < K < 1.

$$\dot{\theta}_{Mo}(n) = \frac{e_M(n)}{K_{EM}} + \Delta\dot{\theta}_{Mo}(n)$$
$$\Delta\dot{\theta}_{Mo}(n) = \Delta\dot{\theta}_{Mo}(n-1) + K\{\Delta\dot{\theta}_M(n) - \Delta\dot{\theta}_{Mo}(n-1)\}$$

Control flow of this control method is shown below.

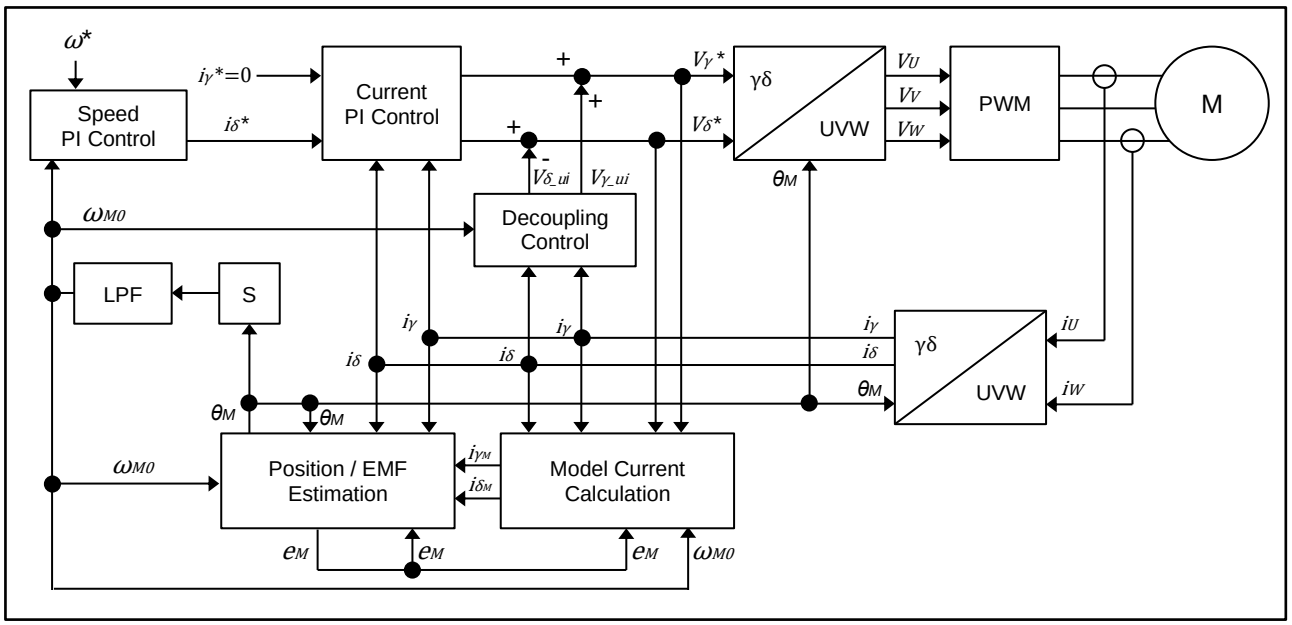


Figure 3-5. Control Flow of the Sensorless Vector Control Based on the Current Estimation Error Method

3.4 Triangular Wave Comparison Method

In order to actually output the voltage command value, the triangular wave comparison method which determines the pulse width of the output voltage by comparing the carrier waveform (triangular wave) and voltage command value waveform is used. By using this PWM formula, output of the voltage command value of the pseudo sinusoidal wave can be performed.

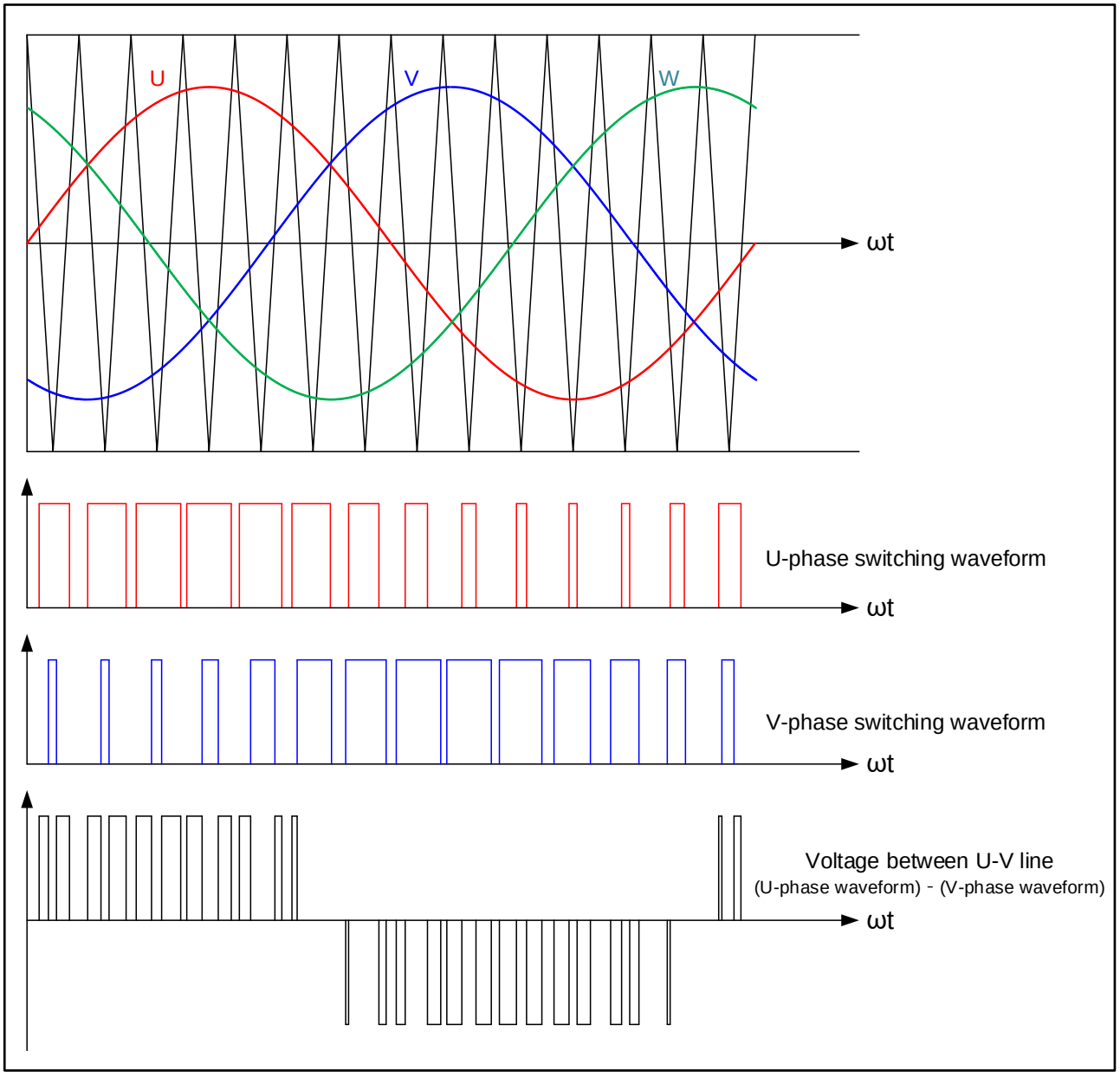


Figure 3-6. Conceptual Diagram of the Triangular Wave Comparison Method

Here, as shown in the Figure 3-7, the ratio of the output voltage pulse to the carrier wave is called the duty ratio, or simply duty.

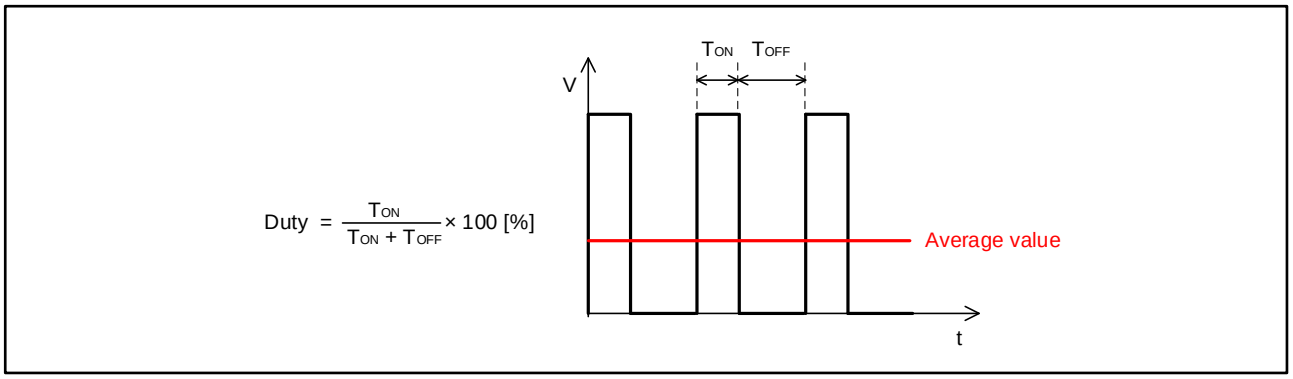


Figure 3-7. Definition of Duty

Modulation factor ‘m’ is defined as follows.

$$m = \frac{V}{E}$$

m: Modulation Factor V: Reference Voltage E: Inverter Voltage

A request to control can be performed by setting this modulation factor in the register which determines PWM duty.

4. Description of the Control Software

4.1 Contents of Control Software

4.1.1 Motor Start / Stop

This sample software supports two control methods as below.

To avoid conflicts of both control method, the controls shown in 4.1.1.1 are enabled as the initial state. And the others are disabled by the compile switch.

4.1.1.1 Motor Start due to Elapsed Time after Reset

After reset release, motor control starts. A counter variable is provided during interrupt processing of the 1 [ms] interval timer. Changes the control according to the number of counter variable. The rotation speed switches by changing the control.

This sample software repeats the following operation.

	Sample Software Processing
(1)	After 3 [s] from reset release, starts operation targeted on 1000 [rpm] as speed command value.
(2)	After 10 [s] from above condition, speed command value changes to 2000 [rpm].
(3)	After 10 [s] from above condition, speed command value changes to 3000 [rpm].
(4)	After 10 [s] from above condition, stops the motor rotation.
(5)	After 1 [s] from above condition, reset the counter variable.
	Resume to (1).

4.1.1.2 Motor Start by PWM Command Input

The motor rotation speed is generated from the duty ratio given to the PWM communication port.

The duty ratio is generated by comparing the time of the high-level period and the low-level period of the voltage level on the PWM communication port. And the motor rotation speed is calculated from the obtained duty ratio.

A rotation command value is generated with 100 [%] duty ratio as 0 [rpm] (energization stop state) and 0 [%] as 3000 [rpm].

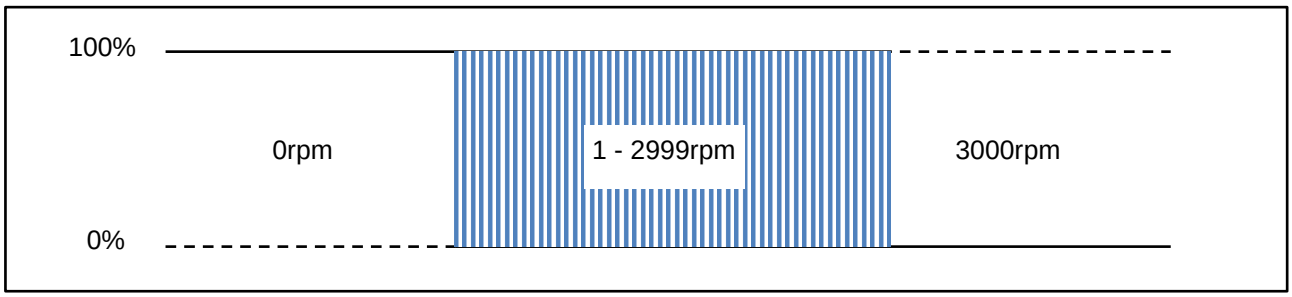


Figure 4-1. Conversion from PWM Ratio to Rotation Command Value

(1) Duty measurement method

TAU05 (TI05) is used for the duty measurement.
The input pulse width measurement function of TAU is possible to measure both high-level and low-level width. The input edge should enable both edges.
After the interrupt by edge detection on TI05 occurred, it determines if rising or falling edge was input by checking port input level. The period from the falling edge to the rising edge is measured as the low-level width period. If the high-level width and low-level width periods have been acquired at the detection timing of the falling edge, the duty ratio is calculated using these two values.

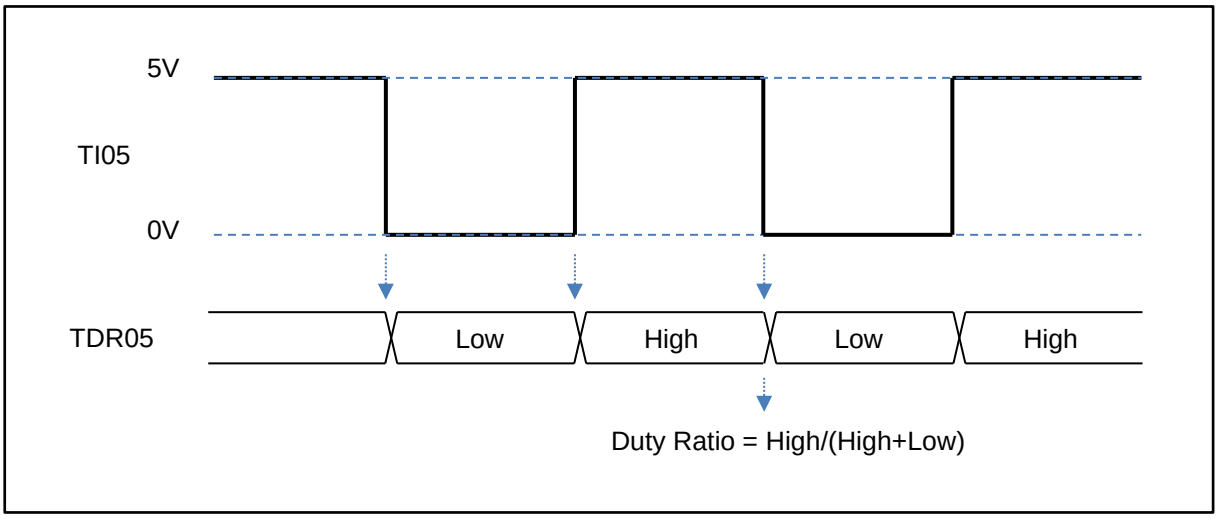


Figure 4-2. PWM Duty Ratio Measurement Method

(2) 0%, 100% duty ratio detection

Input pulse width measurement can't detect 0% and 100% duty ratio as no edge input occur. Therefore, a time-out event by the interval timer function is used instead of the pulse width measurement function to determine that the duty is 0% or 100% if the next edge is not detected within a certain period after the counter start or the last edge detection.
This timer is reset every time at edge detection, and once time out event is generated, it will not be restarted until the edge is detected again.

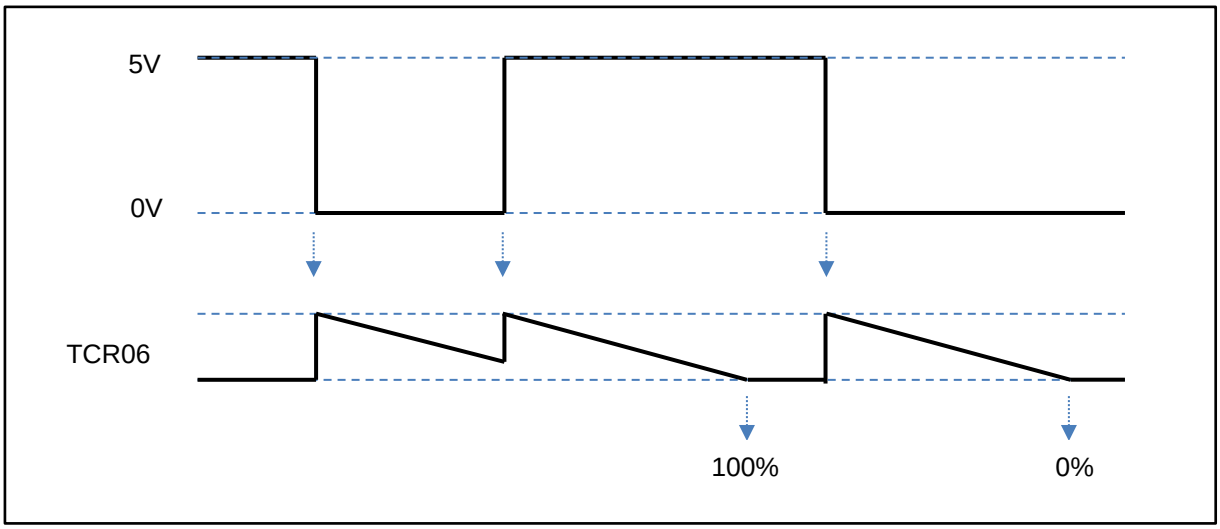


Figure 4-3. Duty Ratio 0% or 100% Detection Method

(3) Setting of timer counter

Applicable input frequency to TI05 is 10 [Hz] to 1000 [Hz].
To satisfy this condition, this sample program uses the following setting to the count source and counter setting values.

Table 4-1. Timer Specification of Motor Operation by PWM Command Input

Timer function	Operation clock	Compare or Capture value	Resolution	Error amount based on 100% = 3000 rpm
Input pulse width measurement	62.5 kHz (40 MHz/2^6)	1 [ms] (625) @ 1 kHz	0.16 %	4.8 rpm
		100 [ms] (62,500) @ 10 Hz	0.0016 %	0.048 rpm
One count timer	62.5 kHz (40 MHz/2^6)	100 [ms] (62,500)	—	—

Caution: System operation can't be expected when input frequency is out of support range. (When the total value of high-level width + low-level width is less than 1 ms or exceeds 100 ms.)

4.1.2 Motor Starting Method

Position detection by sensorless vector control estimates the amount of rotor movement and rotation speed by measuring the three-phase current value and using the current error with the motor model. When the motor is stopped or rotating at low speed, the induced voltage by the rotation of the motor is small. So the amount of error does not generate sufficiently and the position cannot be estimated.

In this sample software, open-loop control (synchronous operation) is performed to rotate the motor by applying a positioning current to the motor at start-up and forcibly commutating it.

When the motor system is changed to the 'SEQ_MODE_RUN' state, the current command value control increases (decreases during CCW) the current command value of the d-axis and q-axis according to the application amount setting during each open-loop control. When the amount of d-axis current applied reaches the d-axis current request value of open-loop, the rotation command value starts to increase or decrease.

The required time for position estimation depends on the d-axis current application speed and the d-axis current command value of open-loop.

The forced commutation time is determined by the interrelationship between the speed control amount of the internal speed command value and the FOC feedback control switching rotation speed.

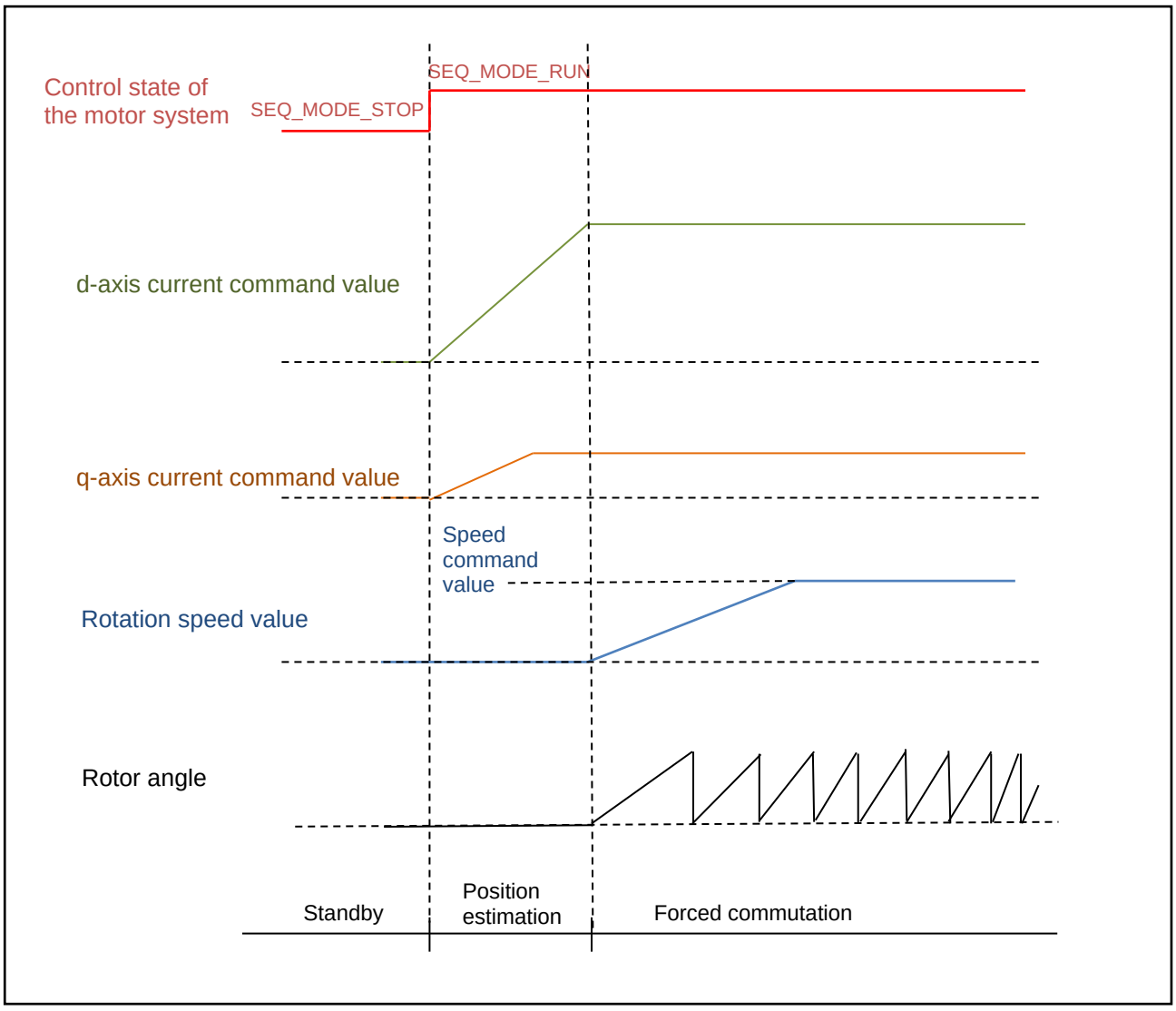


Figure 4-4. Operation Image of Synchronous Operation

4.1.3 Closed-Loop Control

When the rotation speed exceeds a certain level in open-loop control, the control state is switched to closed-loop control. In closed-loop control, the q-axis current command value is increased or decreased by speed feedback to approach the target value. PI control is used for this feedback processing. For low-loss and highly efficient control, the d-axis current is controlled as Id = 0, and is lowered so that the d-axis current command value becomes 0 [A].

In addition, a settling time is provided immediately after switching from open-loop control to closed-loop control. During this period, the speed command value is not increased or decreased.

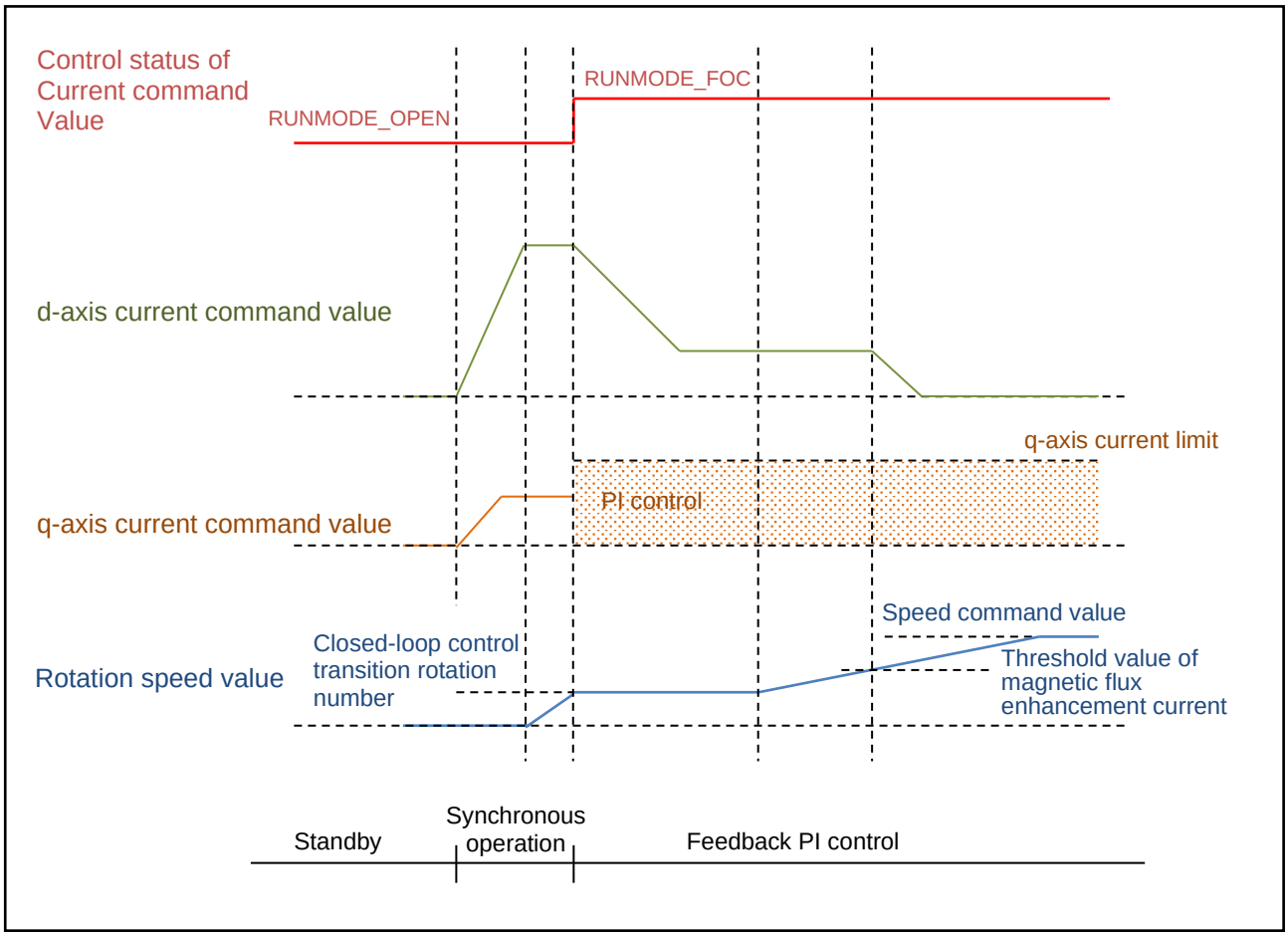


Figure 4-5. Operation Image of the Speed Control

4.1.4 Motor Control State Transition

The state transition diagram of the sample software is explained.

4.1.4.1 Current Reference Controller

It manages current command value control status.

The current command control of this sample software has two states which are ‘RUNMODE_OPEN’ (Open-loop control) and ‘RUNMODE_FOC’ (FOC current feedback control).

After initialization, it starts operation by ‘RUNMODE_OPEN’. If the system is in the ‘SEQ_MODE_RUN’ state and the ‘RUNMODE_OPEN’ state, the motor will start rotation. When the rotation speed reaches the closed-loop control transition rotation number, it transitions to ‘RUNMODE_FOC’.

If the control state of the motor system is ‘SEQ_MODE_STOP’ or ‘SEQ_MODE_ERROR’, it will be in the ‘RUNMODE_OPEN’ state.

The state transition diagram of the current command value control status is shown in Figure 4-6.

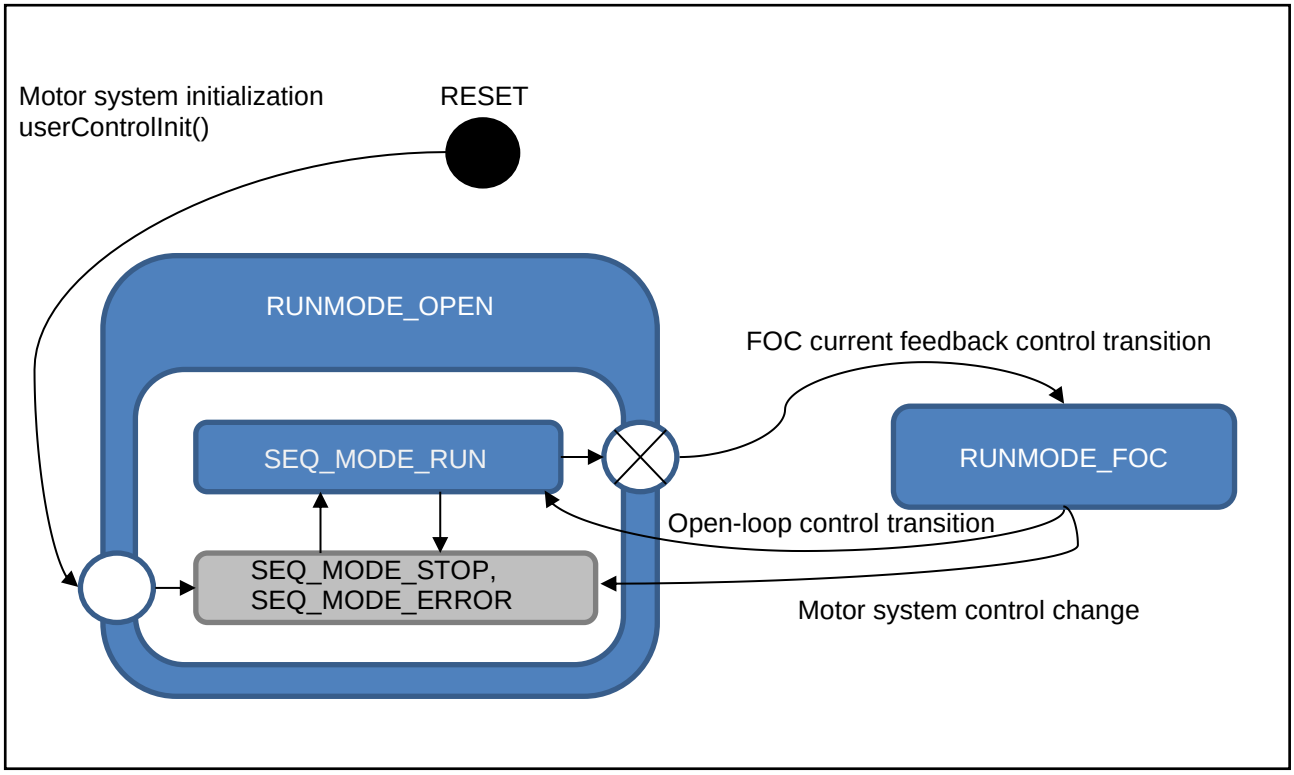


Figure 4-6. State Transition Diagram of the Current Command Control

Remark: Regarding the state transition of the motor system, refer to “4.1.4.3 Sequence Controller”.

4.1.4.2 Current Regulator

It manages rotation control status of vector control.

The rotation control status transitions to the current sensor adjustment state (FOC_STATE_ADJUST) by initialization. Then, after adjusting the offset of the current sensor, it transitions to the motor operable state (FOC_STATE_ACTIVE).

If an error occurred in each state, it transitions to the error state (FOC_STATE_ERROR). It is necessary to initialize again to recover from the error state.

Figure 4-7 shows the state transition diagram of rotation control status.

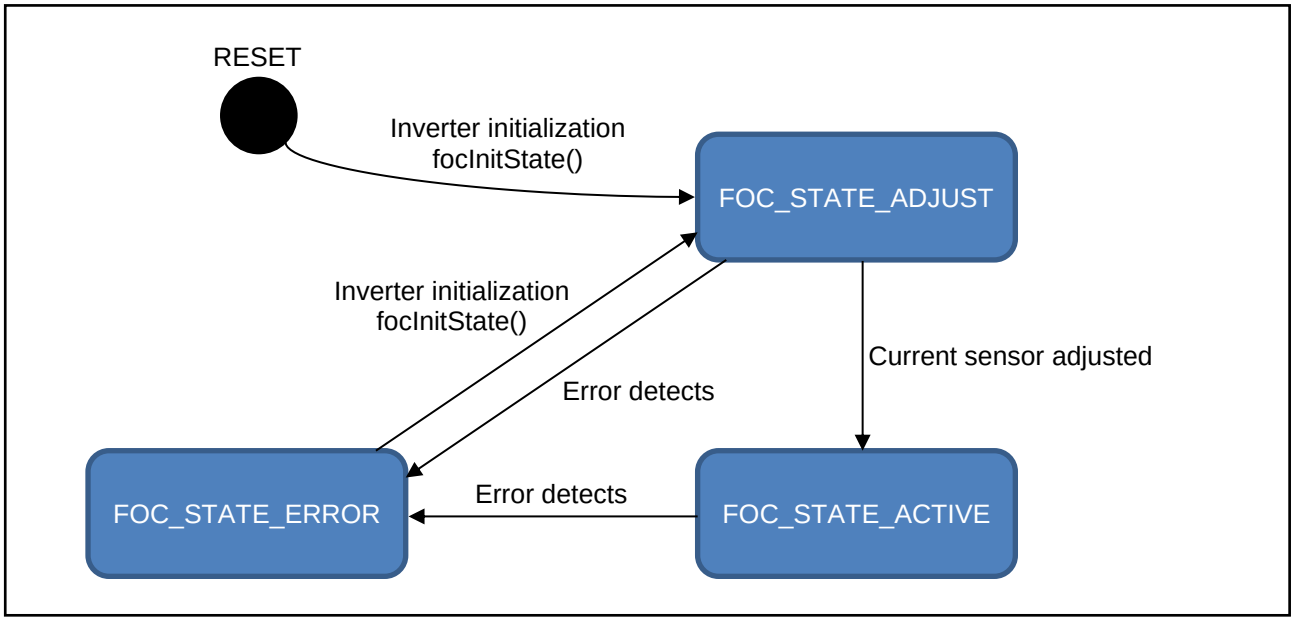


Figure 4-7. State Transition Diagram of the Rotation Control

4.1.4.3 Sequence Controller

It manages control state of the motor system.

The motor system transitions to 'SEQ_MODE_STOP' after initialization. Control state is changed according to the request of the motor system every 1 ms. Transitioning to 'SEQ_MODE_RUN', rotation control status should be 'FOC_STATE_ACTIVE' (motor operable state) as it involves the output of the inverter.

Figure 4-7 shows the state transition diagram of the motor system.

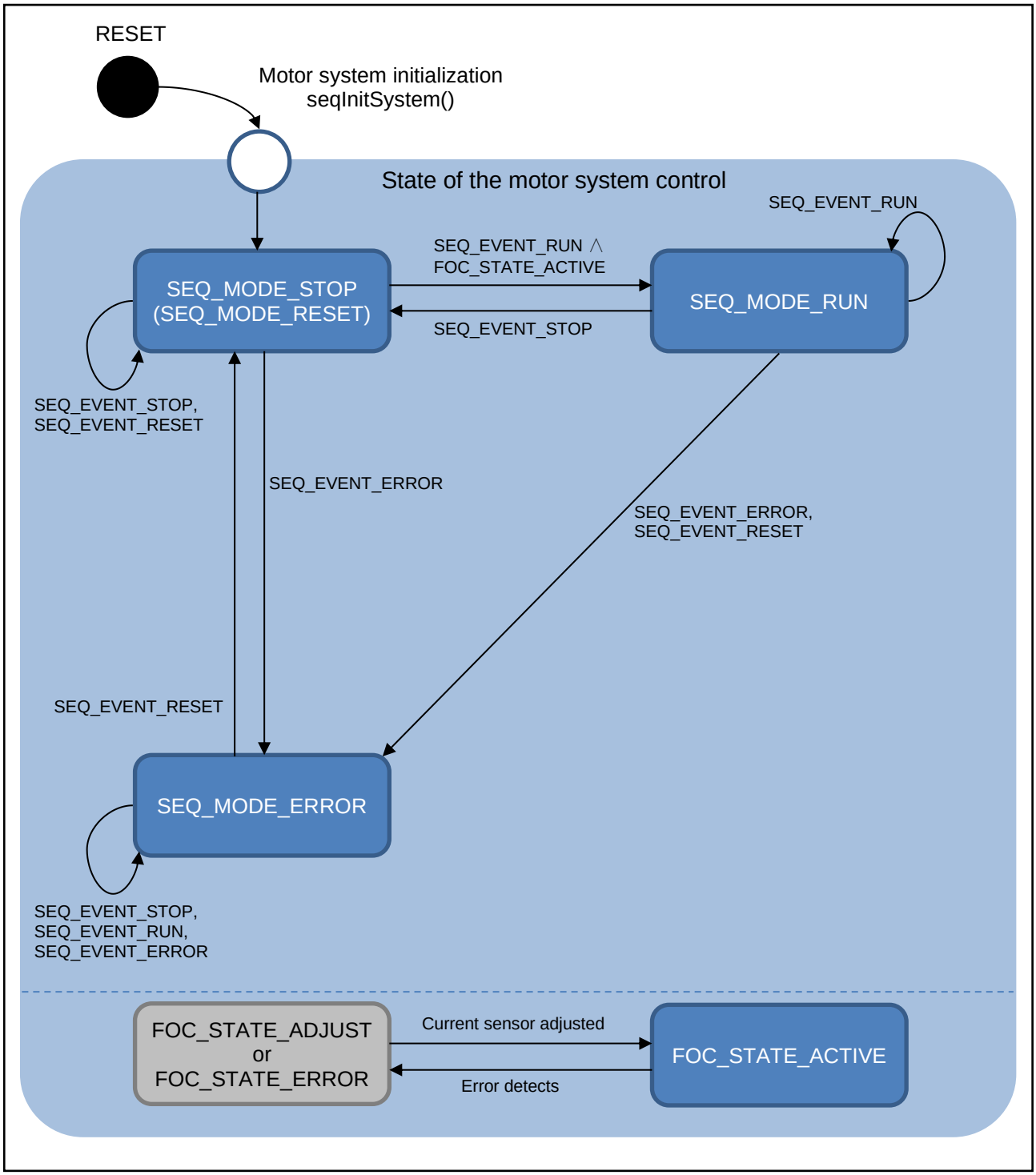


Figure 4-8. State Transition Diagram of the Motor System

Remark: Regarding the state transition diagram for rotation control status of vector control, refer to "4.1.4.2 Current Regulator".

4.1.5 System Protection Functions

This sample program supports the following error state. Each error state executes stop processing.

(1) Error identification

- Overcurrent detection (H/W)
  The pulse output is forcibly cut off by the emergency stop signal (overcurrent detection) from the hardware without intervention of CPU. Overcurrent is detected using the internal comparator on the MCU.
- Overcurrent detection (S/W)
  The current value is monitored for every PWM carrier interrupt. If the current on any one of the three-phase circuit exceeds the positive or negative overcurrent detection threshold value, it is judged as overcurrent. If an overcurrent is detected, the output is stopped, and the error code is generated.
  In this sample program, the current detection threshold is set to 16.97 [A] (10Arms + 20%).

Table 4-2. Parameter of the Overcurrent Detection

Parameter name	Value	Unit	Description
invOverCurLevel	16.97	A	Motor overcurrent detection threshold

- Overvoltage detection
  The V_{dc} power supply voltage is monitored overvoltage in every each PWM carrier interrupt by using the bus voltage detection circuit. If an overvoltage is detected, the output is stopped, and the error code is generated.

Table 4-3. Parameter of the Overvoltage Detection

Parameter name	Value	Unit	Description
invOverVolLevel	28.0	V	V_{dc} overvoltage detection threshold

- Undervoltage detection
  The V_{dc} power supply voltage is monitored undervoltage in every each PWM carrier interrupt by using the bus voltage detection circuit. If an undervoltage is detected, the output is stopped, and the error code is generated.

Table 4-4. Parameter of the Undervoltage Detection

Parameter name	Value	Unit	Description
invUnderVolLevel	8.0	V	V_{dc} undervoltage detection threshold

- PCB overtemperature detection
  The PCB temperature is monitored for every interval timer interrupt. If the PCB overtemperature is detected, the output is stopped, and the error code is generated. In addition, it issues a warning at 10 °C lower than the overtemperature detection threshold. The warning is cleared when the temperature drops by 5 °C below the overtemperature warning threshold.

Table 4-5. Parameter of the PCB Overtemperature Detection

Parameter name	Value	Unit	Description
errTempBoardOHD	120	°C	PCB overtemperature detection temperature
errTempBoardWD	110	°C	PCB overtemperature warning detection temperature
errTempBoardWC	105	°C	PCB overtemperature warning release temperature

- Motor coil-end overtemperature detection
  The motor coil-end temperature is monitored for every interval timer interrupt. If the overtemperature of the motor coil-end is detected, the output is stopped, and the error code is generated. In addition, it issues a warning at 10 °C lower than the overtemperature detection threshold. The warning is cleared when the temperature drops by 5 °C below the overtemperature warning threshold.

Table 4-6. Parameter of the Motor Coil-end Overtemperature Detection

Parameter name	Value	Unit	Description
errTempCoilOHD	180	°C	Motor coil-end overtemperature error detection temperature
errTempCoilWD	170	°C	Motor coil-end overtemperature warning detection temperature
errTempCoilWC	165	°C	Motor coil-end overtemperature warning release temperature

- Detection of acceleration and motor lock detection
  It monitors step-out state using estimated speed. When abnormal rotation exceeds the control range, it is judged to be step-out state and executes stop processing. At high rpm, stop processing is performed when the speed exceeds the overspeed detection rpm.
  At low rpm, if the motor lock detection rotation number continues intermittently for 1 second or longer, it is judged as an error and stop processing is performed.

Table 4-7. Parameter of the Motor Lock Detection

Parameter name	Value	Unit	Description
errRpmOsd	5000	rpm	Overspeed detection threshold rotation number
errRpmUsd	150	rpm	Motor lock detection threshold rotation number

(2) Error number

If system error occurred, error code is stored in errErrorStatus. errErrorStatus is a 16bit variable. Meaning of each bit is shown in Figure 4-9 and Table 4-8.

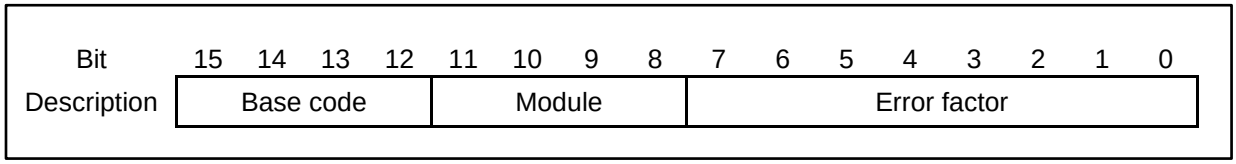


Figure 4-9. Bit Assignment on errErrorStatus

Table 4-8. Error Number

Item	Value	Content
Base code	0x8	Warning code
	0xC	Error code
Module	0x0	MCU
	0x1	INV
	0x8	FOC
	0x9	APP
Error factor	0x00	Overcurrent detection * Module is only MCU. Initialization error.
	0x10	Overvoltage detection
	0x11	Undervoltage detection
	0x20	Overtemperature detection
	0x30	Overspeed detection
	0x31	Motor lock detection
	0x80	State transition error
	0xFF	UNKNOWN

4.2 System Resources

4.2.1 Interrupt Function

Table 4-9 is shown interrupt function which is used for this sample program.

Table 4-9. Interrupt Function List

Interrupt source	Interrupt handler	Interrupt condition	Main function
Carrier cycle interrupt (INTTRD1)	intCarrierInterrupt()	100.0 [µs] (20 [kHz]) *Decimated once	• User processing • Current control loop • Error monitor
Interval timer interrupt (INTTM01)	intIntervalTimerInterrupt()	1 [ms] (1 [kHz])	• User processing • Speed control • Current control • Error monitor
Capture timer interrupt (INTTM05)	r_tau0_ channel5_interrupt()	TI05 input signal edge detection	PWM signal input control
One count timer interrupt (INTTM06)	r_tau0_ channel6_interrupt()	100 [ms] (10 [Hz])	Occurs 100 ms after the last INTTM05

(1) Interrupt priority level

Target MCU has 4 interrupt priority level. And maximum 5 interrupt priority level is available by combination with multiple interrupt servicing. Table 4-10 shows the priority levels of interrupt tasks used in this system.

Table 4-10. Interrupt Priority Level Setting

Interrupt priority level	Interrupt task	Remark
0 (Disable Interrupt)	Not applicable	Multiple interrupt servicing is prohibited.
0 (Enable Interrupt)	Not applicable	Multiple interrupts servicing is possible if it is Lv0 interrupt.
1 (Enable Interrupt)	INTTRD1	Multiple interrupts servicing is possible if it is Lv0 interrupt.
2 (Enable Interrupt)	INTTM01	Multiple interrupts servicing is possible if it is Lv0 and Lv1 interrupt.
3 (Enable Interrupt)	INTTM05, INTTM06	Multiple interrupts servicing is possible if it is Lv0 to 2 interrupt.

(2) Setting of register bank

Using register bank is possible to shorten Using a register bank can shorten the register save time when an interrupt occurs and improve the processing speed. The register bank settings for each interrupt task are as follows.

Table 4-11. Register Bank Setting

Interrupt task	Bank setting	Description
INTTRD1	RB1	Don’t use RB1 (register bank 1) for other.
INTTM01	Not specified (RB0)	Default operation.
INTTM05	Not specified (RB0)	Default operation.
INTTM06	Not specified (RB0)	Default operation.

4.2.2 PWM Output Function

Table 4-12 shows the list of PWM output used in this control program.

Table 4-12. PWM Output Function List

I/O	Pin	Function	Remark
Output	TRDIOB0 / P11	U-phase upper arm motor control PWM output (U_P)	Logic setting is High level active
	TRDIOD0 / P12	U-phase lower arm motor control PWM output (U_N)	
	TRDIOA1 / P15	V-phase upper arm motor control PWM output (V_P)	
	TRDIOC1 / P16	V-phase lower arm motor control PWM output (V_N)	
	TRDIOB1 / P17	W-phase upper arm motor control PWM output (W_P)	
	TRDIOD1 / P30	W-phase lower arm motor control PWM output (W_N)	

4.3 Function Specifications of the Sample Program

Table 4-13 shows the function list of this sample program.

Table 4-13. Function List (1/9)

Function specification	Process overview
File name: control_math.c	
Function name : ctrlPiControl() Input : pi_prm_t * obj Output : int16_t ret_value	PI control obj: PI control object ret_value: PI control result
Function name : ctrlLimInt16() Input : int16_t src int16_t limit Output : int16_t ret_value	Limit control src: Control value limit: Limit value ret_value: Limit control result
File name: error_management.c	
Function name : errClearStatus() Input : None Output : None	Clearing the error status
Function name : errCheckError() Input : None Output : None	• Error check • Warning release check • Warning occurrence check • Process when the error occurs • Motor system update
File name: foc_current_ref_ctrl.c	
Function name : crefFocControlTimer() Input : None Output : None	Current command value control Identifies control mode of open-loop or closed-loop and calls the function.
Function name : crefForcedCommutation() Input : None Output : None	Forced commutation control
Function name : crefFluxCurrentRefControl() Input : None Output : None	Magnetic flux current value control • Calls the user defined process of d-axis current • Addition/ subtraction of d-axis current command value
Function name : crefSpeedControl() Input : None Output : None	Generates d-axis current by speed feedback control.

Table 4-13. Function List (2/9)

Function specification	Process overview
File name: foc_current_regulator.c	
Function name : focInitState() Input : None Output : None	Initialization of the inverter • Reset the rotation control status and the error status • Initialization of the inverter • Clearing control parameter • Initialization and re-setting of the status on the position estimation library
Function name : focFocCarrierLoop() Input : None Output : None	Current control loop process • Feedback calculation by AAU function • Execution of the position estimation calculation • Calculation of non-interference term • Output voltage setting • Execution of error check
Function name : focErrorCheck() Input : None Output : None	Execution of error check
Function name : focStartDrive() Input : None Output : None	• Starts the gate driver output if the rotation control status is the standby state (FOC_STATE_ACTIVE). • Updates the parameters of the position estimation library to the operation state.
Function name : focStopDrive() Input : None Output : None	• Updates the parameters of the position estimation library to the stopping state. • Stops the gate driver output.
Function name : focGetStatus() Input : None Output : uint8_t focStatus	Acquisition of the rotation control status focStatus: Rotation control status
Function name : focGetError() Input : None Output : uint8_t focError	Acquisition of FOC error code focError: FOC error code
Function name : focCompensateAngle() Input : int16_t theta int16_t angular_velocity int16_t comp_time Output : int16_t ret_value	Calculation of the corrected angle value from estimation angle value, estimation speed value and corrected time. theta: Estimation angle value [rad] angular_velocity: Estimation speed value [rad/s] comp_time: Corrented time [s] ret_value: Corrected angle value [rad]
File name: foc_math.c	
Function name : focCalcVqLim() Input : int16_t vd_ref int16_t vdq_limit Output : int16_t ret_value	q-axis voltage limit value calculation vd_ref: d-axis voltage command value vdq_limit: dq-axis voltage limit value ret_value: q-axis voltage limit value
File name: mtr_speed_ref_ctrl.c	
Function name : spdSlopeControl() Input : None Output : None	Speed command value control process • Checking the upper/lower limit of speed commend value • Adding slope value to speed command value • Judgment of switching to open-loop or closed-loop
File name: rl78_interrupt.c	
Function name : intIntervalTimerInterrupt() Input : None Output : None	Interval timer interrupt process • Calls the user cyclic interrupt process function • Calls the speed slope control function • Calls the cyclic interrupt process function • Calls the error detection process function
Function name : intCarrierInterrupt() Input : None Output : None	Carrier cycle interrupt process • Calls the current control loop function • Calls the user carrier cyclic interrupt process function

Table 4-13. Function List (3/9)

Function specification		Process overview
File name: rssk_inv.c		
Function name : Input : Output :	invInitBoard() None uint8_t error_type	• Initialization of the inverter board • Monitoring overcurrent state error_type: Inverter error code
Function name : Input : Output :	invEnableGateDriver() None None	Enables the gate driver output
Function name : Input : Output :	invDisableGateDriver() None None	Disables the gate driver output
Function name : Input : Output :	invSetUVW() int16_t ref_vu int16_t ref_vv int16_t ref_vw None	Update process of the three-phase PWM output value • Calculation of duty ratio • Execution of the modulation process • Updates three-phase PWM output value ref_vu: U-phase voltage command value ref_vv: V-phase voltage command value ref_vw: W-phase voltage command value
Function name : Input : Output :	invModulation() int16_t * mu int16_t * mv int16_t * mw uint16_t mode None	PWM modulation process mu: U-phase voltage command value mv: V-phase voltage command value mw: W-phase voltage command value mode: Modulation mode (Normal or 1/2 voltage addition method)
Function name : Input : Output :	invGetError() None uint8_t error_type	Execution of the inverter error check error_type: Inverter error code
Function name : Input : Output :	invGetCurrent() None int16_t * p_iu int16_t * p_iv int16_t * p_iw	• Acquisition of the three-phase current values • Removal of the offset value • Updating the current value of each phase p_iu: U-phase current value p_iv: V-phase current value p_iw: W-phase current value
Function name : Input : Output :	invAdjustCurrent() None int16_t * p_iu int16_t * p_iv int16_t * p_iw	• Acquisition of the three-phase current values • Calculation of the offset value • Removal of the offset value • Updating the current value of each phase p_iu: U-phase current value p_iv: V-phase current value p_iw: W-phase current value
Function name : Input : Output :	invAdGetIx() uint8_t channel int16_t calc_buf	Acquisition of the three-phase current values channel: A/D conversion channel calc_buf: A/D conversion result
Function name : Input : Output :	invGetVdc() None int16_t calc_buf	Acquisition of the V_{dc} voltage calc_buf: Power supply voltage value
Function name : Input : Output :	invGetBoardTemp() None int16_t temp1	Acquisition of the PCB temperature temp1: PCB temperature
Function name : Input : Output :	invGetCoilendTemp() None int16_t temp1	Acquisition of the motor coil-end temperature temp1: Motor coil-end temperature

Table 4-13. Function List (4/9)

Function specification	Process overview
File name: sequence.c	
Function name : seqInitSystem() Input : None Output : None	Motor system initialization • Clearing sequence error status • Initialization of the rotor state
Function name : seqExecEvent Input : uint8_t sys_mode_req Output : None	Executes sub task from combination of current motor system and requirement motor system. sys_mode_req: Requirement motor system
Function name : seqActRun() Input : uint8_t req_state Output : uint8_t ret	Starts motor operation req_state: Requirement motor system ret: Requirement motor system
Function name : seqActStop() Input : uint8_t req_state Output : uint8_t req_state	Stops motor operation req_state: Requirement motor system req_state: Requirement motor system
Function name : seqActNone() Input : uint8_t req_state Output : uint8_t req_state	No process req_state: Requirement motor system req_state: Requirement motor system
Function name : seqActReset() Input : uint8_t req_state Output : uint8_t req_state	Initialization of the parameter for rotation control req_state: Requirement motor system req_state: Requirement motor system
Function name : seqActError() Input : uint8_t req_state Output : uint8_t req_state	Stops motor operation req_state: Requirement motor system req_state: Requirement motor system
Function name : seqGetSystemMode() Input : None Output : uint8_t seqSystemMode	Acquisition of the motor system seqSystemMode: Motor system
File name: user_control.c	
Function name : userControlInit() Input : None Output : None	Initialization of the user parameter
Function name : userControlTimer() Input : None Output : None	Execution of the user cyclic interrupt process
Function name : userControlCarrier() Input : None Output : None	Execution of the user carrier cyclic interrupt process
Function name : userIdReferenceControl() Input : None Output : None	Selection of d-axis current control method
Function name : userErrorDetected() Input : uint8_t * const sys_mode uint16_t const err_status Output : None	Execution of the user error detection process sys_mode: Motor system err_status: Error status
Function name : userWarningDetected() Input : uint8_t * const sys_mode uint16_t const err_status Output : None	Execution of the user warning detection process sys_mode: Motor system err_status: Error status

Table 4-13. Function List (5/9)

Function specification		Process overview
File name: r_cg_aau.c		
Function name : Input : Output :	R_AAU_Create() None None	Initialization of AAU function
Function name : Input : Output :	R_AAU_MotorPI_Set_Reference() int16_t id_ref int16_t iq_ref None	Sets PI control target value for AAU function id_ref: d-axis target value setting iq_ref: q-axis target value setting
Function name : Input : Output :	R_AAU_MotorPI_Set_Params() float kp_d float ki_d float kp_q float ki_q int16_t i_limit int16_t pi_limit None	Sets PI control parameter for AAU function kp_d: d-axis proportionality coefficient ki_d: d-axis integral coefficient kp_q: q-axis proportionality coefficient ki_q: q-axis component coefficient i_limit: Integral term limit pi_limit: PI control output limit
Function name : Input : Output :	R_AAU_MotorPI_Set_Integral() int16_t id_buf int16_t iq_buf None	Updates PI control integral term for AAU function id_buf: d-axis integral value iq_buf: q-axis integral value
Function name : Input : Output : 	R_AAU_UW2DQ_withPI() int16_t src_iu int16_t src_iw int16_t theta int16_t * p_dst_id int16_t * p_dst_iq int16_t * p_dst_vd int16_t * p_dst_vq	• Execution of Clarke & Park transformation • Current PI control calculation src_iu: U-phase current value [A] src_iw: W-phase current value [A] theta: angle value [rad] p_dst_id: d-axis current value storage address p_dst_iq: q-axis current value storage address p_dst_vd: d-axis voltage value storage address p_dst_vq: q-axis voltage value storage address
Function name : Input : Output : 	R_AAU_DQ2UVW() int16_t src_vd int16_t src_vq int16_t theta int16_t * p_dst_vu int16_t * p_dst_vv int16_t * p_dst_vw	Execution of I-Park & I-Clarke transformation src_vd: d-axis voltage value [V] src_vq: q-axis voltage value [V] theta: angle value [rad] p_dst_vu: U-phase voltage value storage address p_dst_vv: V-phase voltage value storage address p_dst_vw: W-phase voltage value storage address
File name: r_cg_adc.c		
Function name : Input : Output :	R_ADC_Create() None None	Initialization of the A/D converter
Function name : Input : Output :	R_ADC_Start() None None	Permission to start the A/D converter
Function name : Input : Output :	R_ADC_Get_Result() uint8_t channel uint16_t * const buffer	Acquisition of the A/D conversion result channel: Converted channel buffer: Conversion result storage address

Table 4-13. Function List (6/9)

Function specification	Process overview
File name: r_cg_cgc.c	
Function name : R_CGC_Create() Input : None Output : None	Initialization of the clock generator
File name: r_cg_comp.c	
Function name : R_COMP_Create() Input : None Output : None	Initialization of the comparator
Function name : R_COMP0_Start() Input : None Output : None	Starts the comparator operation
File name: r_cg_dac.c	
Function name : R_DAC_Create() Input : None Output : None	Initialization of the D/A converter
Function name : R_DAC0_Start() Input : None Output : None	Starts the D/A converter operation
File name: r_cg_port.c	
Function name : R_PORT_Create() Input : None Output : None	Initialization of I/O port
Function name : R_PORT140_Set() Input : uint8_t level Output : None	P14.0 port setting level: Port output level
File name: r_cg_timer_rde_user.c	
Function name : r_tmr_rd1_interrupt() Input : None Output : None	Timer RDe interrupt handler

Table 4-13. Function List (7/9)

Function specification		Process overview
File name: r_cg_timer_rde.c		
Function name : Input : Output :	R_TMR_RD0_Create() None None	Initialization of Timer RDe
Function name : Input : Output :	R_TMR_RD0_Start() None None	Counter start of Timer RDe
Function name : Input : Output :	R_TMR_RD0_Stop() None None	Counter stop of Timer RDe
Function name : Input : Output :	R_TMR_RD0_Reset() None None	Reset of Timer RDe
Function name : Input : Output :	R_TMR_RD0_Enable_Output() None None	Enables Timer RDe output
Function name : Input : Output :	R_TMR_RD0_Disable_Output() None None	Disables Timer RDe output
Function name : Input : Output :	R_TMR_RD0_Set_Duty() int16_t pwm1 int16_t pwm2 int16_t pwm3 uint8_t ret	Duty ratio setting of three-phase PWM pwm1: PWM1 duty ratio pwm2: PWM2 duty ratio pwm3: PWM3 duty ratio ret: Success or failure of writing (0: Success, 1: Fail)
Function name : Input : Output :	R_TMR_RD0_Get_Cutoff_Status() None uint8_t ret	Acquisition of forced cutoff status by PWMOPA ret: Status of forced cutoff
Function name : Input : Output :	R_TMR_RD0_Release_Cutoff() None uint8_t ret	Releases the forced cutoff state by PWMOPA ret: Success or failure of releasing the forced cutoff
File name: r_cg_timer_user.c		
Function name : Input : Output :	r_tau0_channel1_interrupt() None None	Timer array unit 0 ch.1 interrupt handler
Function name : Input : Output :	r_tau0_channel5_interrupt() None None	Timer array unit 0 ch.5 interrupt handler
Function name : Input : Output :	r_tau0_channel6_interrupt() None None	Timer array unit 0 ch.6 interrupt handler
Function name : Input : Output :	r_tau0_channel5_get_duty() None uint16_t g_tau0_ch5_duty	Acquisition of duty ratio of command value (PWM input) g_tau0_ch5_duty: Duty ratio

Table 4-13. Function List (8/9)

Function specification	Process overview
File name: r_cg_timer.c	
Function name : R_TAU0_Create() Input : None Output : None	Initialization of Timer array unit 0
Function name : R_TAU0_Channel1_Start() Input : None Output : None	Counter start of Timer array unit 0 ch.1
Function name : R_TAU0_Channel1_Stop() Input : None Output : None	Counter stop of Timer array unit 0 ch.1
Function name : R_TAU0_Channel5_Start() Input : None Output : None	Counter start of Timer array unit 0 ch.5
Function name : R_TAU0_Channel6_Start() Input : None Output : None	Counter start of Timer array unit 0 ch.6
File name: r_cg_wdt.c	
Function name : R_WDT_Create() Input : None Output : None	Initialization of WDT
Function name : R_WDT_Restart() Input : None Output : None	Reset WDT counter
File name: r_main.c	
Function name : main() Input : None Output : None	Main function
Function name : R_MAIN_UserInit() Input : None Output : None	Call each initialization function
File name: r_systeminit.c	
Function name : R_Systeminit() Input : None Output : None	Initialization of each macro
Function name : hdwinit() Input : None Output : None	Initialization of hardware setting values Note: This function should be called by the user's initialization software process.

Table 4-13. Function List (9/9)

Function specification		Process overview
File name: r_fixed_point_math.lib		
Function name : Input : Output :	R_SinCos() signed short sc_rad signed short * p_sin signed short * p_cos None	Sin, Cos calculation process sc_rad: Scaled radian angle data p_sin: Sin value address pointer p_cos: Cos value address pointer
Function name : Input : Output :	R_MTR_AngleEstInit() est_configure_t * p_est_conf signed short	Position estimation library initialization process p_est_conf: Initialization parameter address pointer (0: Initialization succeeded, 1: Initialization failed)
Function name : Input : Output :	R_MTR_AngleEstReset() None None	Position estimation library reset process
Function name : Input : Output :	R_MTR_AngleEstExec () signed short * p_speed_rad signed short int * p_angle_rad None	Rotor angle error estimation process p_speed_rad: Rotation speed address pointer p_angle_rad: Rotation angle address pointer

4.4 Variables Specifications of the Sample Program

Table 4-14 shows a list of variables in the sample program. However, local variables are not listed.

Table 4-14. Variables List (1/2)

Variables name	Type	Contents
errErrorStatus	uint16_t	Error status
errRpmOsd	int16_t	Overspeed error threshold [rpm (machine angle)]
errRpmUsd	int16_t	Low speed threshold for motor lock detection [rpm (machine angle)]
errTempBoardOHD	int16_t	PCB overtemperature detection threshold [°C]
errTempBoardWD	int16_t	PCB overtemperature warning threshold [°C]
errTempBoardWC	int16_t	PCB overtemperature warning release threshold [°C]
errTempCoilOHD	int16_t	Motor coil-end overtemperature detection threshold [°C]
errTempCoilWD	int16_t	Motor coil-end overtemperature warning detection threshold [°C]
errTempCoilWC	int16_t	Motor coil-end overtemperature warning release threshold [°C]
errLockDetectFilterSet	uint16_t	Motor lock detection filter value [ms]
errLockDetectCountSet	uint16_t	Motor lock detection threshold [ms]
mathSqrt_3d2	int16_t	Defines $\sqrt{3/2}$ value
mathSqrt_2d3	int16_t	Defines $\sqrt{2/3}$ value
mathSqrt_3_2	int16_t	Defines $\sqrt{3}/2$ value
mathSqrt_2	int16_t	Defines $\sqrt{2}$ value
mathSqrt_2_2	int16_t	Defines $\sqrt{2}/2$ value
crefRunMode	uint8_t	Motor operation mode
crefRpmRef	int16_t	Speed command value [rpm (machine angle)]
crefRpmEst	int16_t	Speed estimation value [rpm (machine angle)]
crefSpeedRadRef	int16_t	Angular velocity command value [rad (electric angle)]
crefIdRefRequest	int16_t	d-axis current request value [A]
crefIdSlopeUpVec32	int32_t	d-axis current application rising slope [A/0.001s]
crefIdSlopeDownVec32	int32_t	d-axis current application falling slope [A/0.001s]
spdfRpmSlopeOl32	int32_t	Acceleration / deceleration amount during open-loop control [rpm/0.001s]
crefIdRefOlRequest	int16_t	Open-loop d-axis current request value [A]
crefIdSlopeOl32	int32_t	Open-loop d-axis current application slope [A/0.001s]
crefIqRefOlRequest	int16_t	Open-loop q-axis current request value [A]
crefIqSlopeOl32	int32_t	Open-loop q-axis current application slope [A/0.001s]
crefPiPrmSpeed	PI_PRM_T	Speed PI control parameter structure
crefIdRefBuffer32	int32_t	Buffer for the d-axis current request valued
crefIqRefBuffer32	int32_t	Buffer for the q-axis current request valued
focTimeSettingOffset	uint16_t	Offset current acquisition time [s/0.00005]
focTimeCountOffset	uint16_t	Counter for the offset current acquisition time
focStatus	uint8_t	Rotation control status
focError	uint8_t	Current control error status
focIdRef	int16_t	d-axis current command value [A]
focIqRef	int16_t	q-axis current command value [A]

Table 4-14. Variables List (2/2)

Variables name	Type	Contents
focCurrentIu	int16_t	U-phase current value [A]
focCurrentIv	int16_t	V-phase current value [A]
focCurrentIw	int16_t	W-phase current value [A]
focCurrentId	int16_t	d-axis current value [A]
focCurrentIq	int16_t	q-axis current value [A]
focVdRef	int16_t	d-axis voltage command value [V]
focVqRef	int16_t	q-axis voltage command value [V]
focVuRef	int16_t	U-phase voltage command value [V]
focVvRef	int16_t	V-phase voltage command value [V]
focVwRef	int16_t	W-phase voltage command value [V]
focVdqLimit	int16_t	dq-axis voltage limit value [V]
focAngleRad	int16_t	Rotor angle [rad (electric angle)]
focSpeedRad	int16_t	Rotor angular velocity [rad (electric angle)/s]
focCompTimeAd	int16_t	Current acquisition angle correction time [s]
focCompTimePWM	int16_t	Voltage output angle correction time [s]
focMtrLd	int16_t	d-axis inductance [H]
focMtrLq	int16_t	q-axis inductance [H]
focMtrKe	int16_t	Back electromotive force coefficient [Vs/rad]
spdRpmRefRequest	int16_t	Speed request value [rpm]
spdRpmLimitMax	int16_t	Speed upper limit value [rpm]
spdRpmLimitMin	int16_t	Speed lower limit value [rpm]
spdRpmSlopeUp32	int32_t	Acceleration slope [rpm/s]
spdRpmSlopeDw32	int32_t	Deceleration slop [rpm/s]
spdRpmOlToFoc	int16_t	Open-loop to closed-loop control transition rotation number [rpm]
spdRpmFocToOl	int16_t	Closed-loop to open-loop control transition rotation number [rpm]
spdCntDelayOlToFoc	uint16_t	Current control settling time [s/0.001]
invOverCurLevel	int16_t	Overcurrent detection threshold [A]
invOverVolLevel	int16_t	Overvoltage detection threshold [V]
invUnderVolLevel	int16_t	Undervoltage detection threshold [V]
invOffsetIu	int16_t	U-phase offset current value [A]
invOffsetIv	int16_t	V-phase offset current value [A]
invOffsetIw	int16_t	W-phase offset current value [A]
userRpmEnhanced	int16_t	Magnetic flux enhancement application threshold [rpm]
userIdRefEnhRequest	int16_t	Magnetic flux enhancement application amount [A]
userThOnBoard	int16_t	PCB temperature [°C]
userThCoilEnd	int16_t	Motor coil-end temperature [°C]
seqSystemMode	uint8_t	Motor system control status
seqErrorStatus	uint8_t	Motor system control error status
g_opsr_clear_flag	uint8_t	PWM output forced cut off clear flag
g_tau0_ch5_width	uint32_t	TAU05 input PWM signal pulse width
g_tau0_ch5_Hwidth	uint32_t	TAU05 input PWM signal High level width
g_tau0_ch5_Lwidth	uint32_t	TAU05 input PWM signal Low level width
g_tau0_ch5_duty	uint16_t	TAU05 input PWM signal duty

4.5 Macro Definitions of the Sample Program

Table 4-15 and Table 4-16 are listed the macro definition in this sample program.

Table 4-15. Macro Definition List (1/7)

Macro name	Defined value	Contents
File name: control_parameter.h (1/2)		
CP_FREQ_SPEED_Hz	(1000)	Speed control cycle [Hz]
CP_RPM_MAX_SPEED	(3000)	Rotation speed command maximum value [rpm]
CP_RPM_MIN_SPEED	(500)	Rotation speed command minimum value [rpm]
CP_RPM_OL_TO_FOC	(300)	Open-loop to closed-loop control switching threshold [rpm]
CP_RPM_FOC_TO_OL	(100)	Closed-loop to open-loop control switching threshold [rpm]
CP_RPM_OSD_SPEED	(5000)	Overspeed error detection threshold [rpm]
CP_RPM_USD_SPEED	(150)	Motor lock detection speed threshold [rpm]
CP_RPM_SLOPE_OL_REQ	(10000)	Acceleration / deceleration value in open-loop [rpm/s]
CP_ID_REF_OL_REQ	(1.02f)	d-axis current command value in open-loop (position estimation) [A]
CP_IQ_REF_OL_REQ	(0.30f)	q-axis current command value in open-loop [A]
CP_ID_SLOPE_OL_REQ	(30.0f)	d-axis current slope in open-loop [A/s]
CP_IQ_SLOPE_OL_REQ	(10.0f)	q-axis current slope in open-loop [A/s]
CP_RPM_SLOPE_UP_REQ	(40000)	Speed command value in closed-loop acceleration slope [rpm/s]
CP_RPM_SLOPE_DW_REQ	(25000)	Speed command value in closed-loop deceleration slope [rpm/s]
CP_DELAY_OL_TO_FOC_s	(0.050f)	Settling time after switching from open-loop to current control [s]
CP_ID_SLOPE_UP_REQ	(8.0f)	d-axis current in closed-loop rising slope [A/s]
CP_ID_SLOPE_DOWN_REQ	(80.0f)	d-axis current in closed-loop falling slope [A/s]
CP_SPEED_PI_KP	(0.1f)	Speed PI control proportional coefficient
CP_SPEED_PI_KI	(0.002f)	Speed PI control integral coefficient
CP_IQ_LIMIT	(2.88f)	q-axis current limit
CP_RPM_ENHANCE	(450)	Magnetic flux enhancement control threshold [rpm]
CP_ID_REF_ENH_REQ	(0.5f)	Magnetic flux enhancement current value [A]

Table 4-15. Macro Definition List (2/7)

Macro name	Defined value	Contents
File name: control_parameter.h (2/2)		
CP_RPMSLOPE_UP_DIV_FREQ	(Note 1)	Speed command value acceleration slope during closed-loop per speed control cycle [rpm/s]
CP_RPMSLOPE_DW_DIV_FREQ	(Note 2)	Speed command value deceleration slope during closed-loop per speed control cycle [rpm/s]
CP_RPMSLOPE_OL_DIV_FREQ	(Note 3)	Acceleration / deceleration during open-loop per speed cycle [rpm/s]
CP_IDSLOPE_OL_DIV_FREQ	(Note 4)	d-axis current slope during open-loop per speed cycle [A/s]
CP_IQSLOPE_OL_DIV_FREQ	(Note 5)	q-axis current slope during open-loop per speed cycle [A/s]
CP_IDSLOPE_UP_DIV_FREQ	(Note 6)	d-axis current rising slope during closed-loop per speed cycle [A/s]
CP_IDSLOPE_DW_DIV_FREQ	(Note 7)	d-axis current falling slope during closed-loop per speed cycle [A/s]
CP_PWM_CARRIER_Hz	(20000)	PWM carrier frequency [Hz]
CP_TIME_OFFSET	(0.1f)	Offset acquisition time [s]
CP_DECIMATION	(1)	Carrier interrupt decimation number
CP_ID_PI_KP	(1.05188f)	d-axis current PI control proportional coefficient
CP_ID_PI_KI	(0.01000f)	d-axis current PI control integral coefficient
CP_IQ_PI_KP	(1.15313f)	q-axis current PI control proportional coefficient
CP_IQ_PI_KI	(0.01100f)	q-axis current PI control integral coefficient
CP_THETA_EST_K	(0.331446f)	Gain for angle estimation
CP_SPEED_LPF_K	(0.070914f)	Gain for rotation speed estimation
CP_EMF_EST_K	(0.356745f)	Gain for induced voltage estimation
CP_CONTROL_FREQ	(Note 8)	Decimated PWM carrier frequency [Hz]
CP_CONTROL_INTERVAL	(Note 9)	Decimated PWM carrier period [s]

Notes 1. ((float)(CP_RPM_SLOPE_UP_REQ) / (float)CP_FREQ_SPEED_Hz)
2. ((float)(CP_RPM_SLOPE_DW_REQ) / (float)CP_FREQ_SPEED_Hz)
3. ((float)(CP_RPM_SLOPE_OL_REQ) / (float)CP_FREQ_SPEED_Hz)
4. ((float)(CP_ID_SLOPE_OL_REQ) / (float)CP_FREQ_SPEED_Hz)
5. ((float)(CP_IQ_SLOPE_OL_REQ) / (float)CP_FREQ_SPEED_Hz)
6. ((float)(CP_ID_SLOPE_UP_REQ) / (float)CP_FREQ_SPEED_Hz)
7. ((float)(CP_ID_SLOPE_DOWN_REQ) / (float)CP_FREQ_SPEED_Hz)
8. ((float)CP_PWM_CARRIER_Hz / (float)(CP_DECIMATION + 1))
9. ((float)(CP_DECIMATION + 1) / (float)CP_PWM_CARRIER_Hz)

Table 4-15. Macro Definition List (3/7)

Macro name	Defined value	Contents
File name: error_management.h		
WAR	(0x8000U)	Warning code definition
ERR	(0xC000U)	Error code definition
MCU	(0x000U)	MCU module definition
INV	(0x100U)	INV module definition
FOC	(0x800U)	FOC module definition
ERROR_NOT_OCCURRED	(0x0000U)	No error detected
E_MCU_INIT_ERR	(ERR+MCU+0x00U)	Initialization error
E_INV_OVER_CURRENT	(ERR+INV+0x00U)	Overcurrent error (H/W detection)
E_INV_OVER_VOLTAGE	(ERR+INV+0x10U)	Overvoltage error
E_INV_UNDER_VOLTAGE	(ERR+INV+0x11U)	Undervoltage error
E_INV_OVER_TEMP	(ERR+INV+0x20U)	PCB overtemperature detection
E_FOC_OVER_CURRENT	(ERR+FOC+0x00U)	Overcurrent error (S/W detection)
E_FOC_OVER_TEMP	(ERR+FOC+0x20U)	Motor overtemperature detection
E_FOC_OVER_SPEED	(ERR+FOC+0x30U)	Overspeed error
E_FOC_MOTOR_LOCKED	(ERR+FOC+0x31U)	Motor lock detection
E_FOC_INVALID_SEQUENCE	(ERR+FOC+0x80U)	State transition error
E_FOC_UNKNOWN	(ERR+FOC+0xFFU)	Unknown current control error
W_INV_OVER_TEMP	(WAR+INV+0x20U)	PCB overtemperature warning
W_FOC_OVER_TEMP	(WAR+FOC+0x20U)	Motor overtemperature warning
IS_WARNING(status) Input : status Output : TRUE/FALSE	(WAR == ((status) & 0xF000U))	Warning code judgement
IS_ERROR(status) Input : status Output : TRUE/FALSE	(ERR == ((status) & 0xF000U))	Error code judgement
File name：foc_current_ref_ctrl.h		
CREF_RUNMODE_OPEN	(1U)	Operation mode: Open-loop
CREF_RUNMODE_CLOSE_SPD	(2U)	Operation mode: Closed-loop

Table 4-15. Macro Definition List (4/7)

Macro name	Defined value	Contents
File name：foc_current_regulator.h		
FOC_STATE_ADJUST	(0x00U)	Current control state: Current adjustment in progress
FOC_STATE_ACTIVE	(0x01U)	Current control state: Current control enabled
FOC_STATE_ERROR	(0xFFU)	Current control state: Current control error
FOC_ERR_BASE	(0xE0U)	Current control error code base
FOC_ERR_NONE	(0x00U)	No error detected
FOC_ERR_OCD_HW	(FOC_ERR_BASE+0x0U)	Overcurrent error by H/W detection
FOC_ERR_OCD	(FOC_ERR_BASE+0x1U)	Overcurrent error
FOC_ERR_OVD	(FOC_ERR_BASE+0x2U)	V_{dc} overvoltage error
FOC_ERR_UVD	(FOC_ERR_BASE+0x3U)	V_{dc} undervoltage error
FOC_ERR_SEQUENCE	(FOC_ERR_BASE+0xDU)	State transition error
FOC_ERR_INIT	(FOC_ERR_BASE+0xEU)	Initialization error
FOC_ERR_UNKNOWN	(FOC_ERR_BASE+0xFU)	Unknown error
FOC_RPM2RADPS	(Note 1)	Conversion coefficient from rotation speed [rpm] to angular velocity [rad/s]
FOC_RADPS2RPM	(Note 2)	Conversion coefficient from angular velocity [rad/s] to rotation speed [rpm]
FOC_SUP2DQLIM	(Note 3)	Conversion coefficient from the DC-bus voltage to dq-axis voltage limit value
FOC_DIV_PI	(1.0f / MATH_PI)	1/π definition
RECOGNIZE_SPD_LIM	(CP_RPM_OSD_SPEED)	Overspeed error detection threshold [rpm]
FOC_FP_RPM	(Note 4)	Overspeed error detection threshold after scaling [rpm]
FOC_FP_SPEED_RAD	(Note 5)	Overspeed error detection threshold after scaling [rad/s]
FOC_FP_ANGLE_RAD	(SC_GEN_FIX_PT (MATH_TWOPI))	2π definition after scaling
FOC_FP_RPM2RADPS	(SC_GEN_FIX_PT (FOC_RPM2RADPS))	Conversion coefficient from rotation speed [rpm] to angular velocity [rad/s] after scaling
FOC_FP_RADPS2RPM	(SC_GEN_FIX_PT (FOC_RADPS2RPM))	Conversion coefficient from angular velocity [rad/s] to rotation speed [rpm] after scaling
FOC_FP_VSUP2VDQLIM	(SC_GEN_FIX_PT (FOC_SUP2DQLIM))	Conversion coefficient from the bus voltage to dq-axis voltage limit value after scaling
FOC_FP_DIV_PI	(SC_GEN_FIX_PT (FOC_DIV_PI))	1/π definition after scaling
FOC_FP_LD	(SC_GEN_FIX_PT (MP_LD))	d-axis inductance after scaling [H]
FOC_FP_LQ	(SC_GEN_FIX_PT (MP_LQ))	q-axis inductance after scaling [H]
FOC_FP_KE	(SC_GEN_FIX_PT (MP_KE))	Back electromotive force coefficient after scaling [Vs/Rad]

Notes 1. ((MATH_TWOPI * (float)MP_PP) / 60.0f)
2. (60.0f / (MATH_TWOPI * (float)MP_PP))
3. (MATH_SQRT_2_2 * INV_MAX_RATIO)
4. (SC_GEN_FIX_PT((float)(RECOGNIZE_SPD_LIM)))
5. (SC_GEN_FIX_PT((FOC_RPM2RADPS * (float)RECOGNIZE_SPD_LIM)))

Table 4-15. Macro Definition List (5/7)

Macro name	Defined value	Contents
File name: foc_math.h		
MATH_PI	(3.14159265f)	π definition
MATH_TWOPI	(2.0f * (MATH_PI))	2π definition
MATH_SQRT_3d2	(1.224745f)	$\sqrt{3/2}$ definition
MATH_SQRT_2d3	(0.816497f)	$\sqrt{2/3}$ definition
MATH_SQRT_3_2	(0.866025f)	$\sqrt{3}/2$ definition
MATH_SQRT_2	(1.414214f)	$\sqrt{2}$ definition
MATH_SQRT_2_2	((MATH_SQRT_2) / 2.0f)	$\sqrt{2}/2$ definition
FP_SQRT3D2	(SC_GEN_FIX_PT (MATH_SQRT_3d2))	$\sqrt{3/2}$ definition after scaling
FP_SQRT2D3	(SC_GEN_FIX_PT (MATH_SQRT_2d3))	$\sqrt{2/3}$ definition after scaling
FP_SQRT3_2	(SC_GEN_FIX_PT (MATH_SQRT_3_2))	$\sqrt{3}/2$ definition after scaling
FP_SQRT2	(SC_GEN_FIX_PT (MATH_SQRT_2))	$\sqrt{2}$ definition after scaling
FP_SQRT2_2	(SC_GEN_FIX_PT (MATH_SQRT_2_2))	$\sqrt{2}/2$ definition after scaling
File name: motor_parameter.h		
MP_PP	(2U)	Pole logarithm
MP_RA	(2.80f)	Winding resistance value per 1 phase [Ω]
MP_LD	(0.00084150f)	d-axis inductance [H]
MP_LQ	(0.00092250f)	q-axis inductance [H]
MP_KE	(0.00853396f)	Back electromotive force coefficient [Vs/Rad]
MP_MAX_TEMP	(180.0f)	Motor maximum temperature [°C]
File name: sequence.h		
SEQ_MODE_STOP	(0U)	Motor system: Stop
SEQ_MODE_RUN	(1U)	Motor system: Run
SEQ_MODE_ERROR	(2U)	Motor system: Error
SEQ_SIZE_STATE	(3U)	Motor system size
SEQ_EVENT_STOP	(0U)	Required motor system: Stop
SEQ_EVENT_RUN	(1U)	Required motor system: Run
SEQ_EVENT_ERROR	(2U)	Required motor system: Error
SEQ_EVENT_RESET	(3U)	Required motor system: Reset
SEQ_SIZE_EVENT	(4U)	Required motor system size
SEQ_ERR_NONE	(0x00U)	No error detected
SEQ_ERR_UNKNOWN	(0xFFU)	Unknown error
File name: sequence.c		
STATE_TABLE	Refer to Table 4-16	State transition definition table
ACTION_TABLE	Refer to Table 4-16	Action definition table

Table 4-15. Macro Definition List (6/7)

Macro name	Defined value	Contents
File name: rssk_inv.h		
INV_PWM_DEADTIME_s	(0.000001f)	Dead time [s]
INV_SHUNT_R	(0.005f)	Shunt resistor [Ω]
INV_AMP_GAIN	(20U)	Amplifier gain of the voltage between the shunt resistors
INV_AD2CUR	(Note 1)	Conversion coefficient from A/D conversion result to current value
INV_AD2VPN	(65.0f/(float)0x0FFFU)	Conversion coefficient from A/D conversion result to voltage value
INV_MAX_RATIO	(Note 2)	Ratio other than dead time in 1 carrier cycle
INV_OV_LEVEL	(28.0f)	Overvoltage detection threshold [V]
INV_UV_LEVEL	(8.0f)	Undervoltage detection threshold [V]
INV_VOLTAGE_MAX	(40.0f)	Maximum detectable voltage [V]
INV_OC_LEVEL	(16.9706f)	Overcurrent detection threshold [A]
INV_CURRENT_MAX	(20.0f)	Maximum detectable current [A]
INV_CURRENT_OFFSET_K	(0.125f)	Current offset filter coefficient
INV_OVERHEAT_DETECT_TEMP	(120.0f)	PCB overtemperature detection threshold [°C]
INV_MODU_MODE_NONE	(1U)	No modulation control
INV_MODU_MODE_CSVPWM	(2U)	Middle voltage 1/2 addition method
INV_ERR_BASE	(0xC0U)	INV error code base
INV_ERR_NONE	(0x00U)	No error detected
INV_ERR_VDC_OVD	(INV_ERR_BASE + 0x0U)	Overvoltage detection
INV_ERR_VDC_UVD	(INV_ERR_BASE + 0x1U)	Undervoltage detection
INV_ERR_OCD	(INV_ERR_BASE + 0x2U)	Overcurrent detection
INV_AD_CHANNEL_IU	(1U)	A/D channel number: Iu
INV_AD_CHANNEL_IV	(0U)	A/D channel number: Iv
INV_AD_CHANNEL_IW	(2U)	A/D channel number: Iw
INV_AD_CHANNEL_VDC	(5U)	A/D channel number: V_{dc}
INV_AD_CHANNEL_T_RT1	(24U)	A/D channel number: RT1
INV_AD_CHANNEL_T_NCT1	(4U)	A/D channel number: NCT1
INV_FP_CURRENT	(SC_GEN_FIX_PT(INV_CURRENT_MAX))	Maximum detectable current after scaling [A]
INV_FP_VOLTAGE	(SC_GEN_FIX_PT(INV_VOLTAGE_MAX))	Maximum detectable voltage after scaling [V]
INV_FP_AD2CUR	(SC_GEN_FIX_PT (INV_AD2CUR))	Conversion coefficient from A/D conversion result to current value after scaling
INV_FP_AD2VPN	(SC_GEN_FIX_PT (INV_AD2VPN))	Conversion coefficient from A/D conversion result to voltage value after scaling
INV_FP_OFFSET_K	(SC_GEN_FIX_PT (INV_CURRENT_ OFFSET_K))	Current offset filter coefficient after scaling

Notes 1. (((5.0f/(float)(INV_AMP_GAIN))/(INV_SHUNT_R))/(float)0x0FFFU)
2. (1.0f - (2.0f * INV_PWM_DEADTIME_s * (float)(CP_PWM_CARRIER_Hz)))

Table 4-15. Macro Definition List (7/7)

Macro name	Defined value	Contents
File name: thermistor.h		
TH_TEMPTABLE_ NTCG163JF103FT1S	Refer to Table 2-7	NTCG163JF103FT1S conversion table
TH_TEMPTABLE_ 103NT_4_R025H41G	Refer to Table 2-6	103NT_4_R025H41G conversion table
File name: user_control.c		
RPM2REF(mRpm) Input : mRpm Output : value	(SC_FIXED_VAL16 ((mRpm), FOC_FP_RPM))	Rotation speed command value after scaling. mRpm: Rotation speed command value value: Rotation command value after scaling

Table 4-16. Multi-line Macro Definition

STATE_TABLE

```
/* state    0:SEQ_MODE_STOP, 1:SEQ_MODE_RUN,  2:SEQ_MODE_ERROR */\
/* request          */                                          \
/* 0:SEQ_EVENT_STOP  */ { SEQ_MODE_STOP,   SEQ_MODE_STOP,   SEQ_MODE_ERROR },\
/* 1:SEQ_EVENT_RUN   */ { SEQ_MODE_RUN,    SEQ_MODE_RUN,    SEQ_MODE_ERROR },\
/* 2:SEQ_EVENT_ERROR */ { SEQ_MODE_ERROR,  SEQ_MODE_ERROR,  SEQ_MODE_ERROR },\
/* 3:SEQ_EVENT_RESET */ { SEQ_MODE_STOP,   SEQ_MODE_ERROR,  SEQ_MODE_STOP }
```

ACTION_TABLE

```
/* state    0:SEQ_MODE_STOP, 1:SEQ_MODE_RUN,  2:SEQ_MODE_ERROR */\
/* request          */                                          \
/* 0:SEQ_EVENT_STOP  */ { seqActNone,      seqActStop,      seqActNone },\
/* 1:SEQ_EVENT_RUN   */ { seqActRun,       seqActNone,      seqActNone },\
/* 2:SEQ_EVENT_ERROR */ { seqActError,     seqActError,     seqActNone },\
/* 3:SEQ_EVENT_RESET */ { seqActReset,     seqActError,     seqActReset }
```

4.6 Flowchart of the Sample Program

Explains the flow chart for this control program.

4.6.1 Main Processing Function

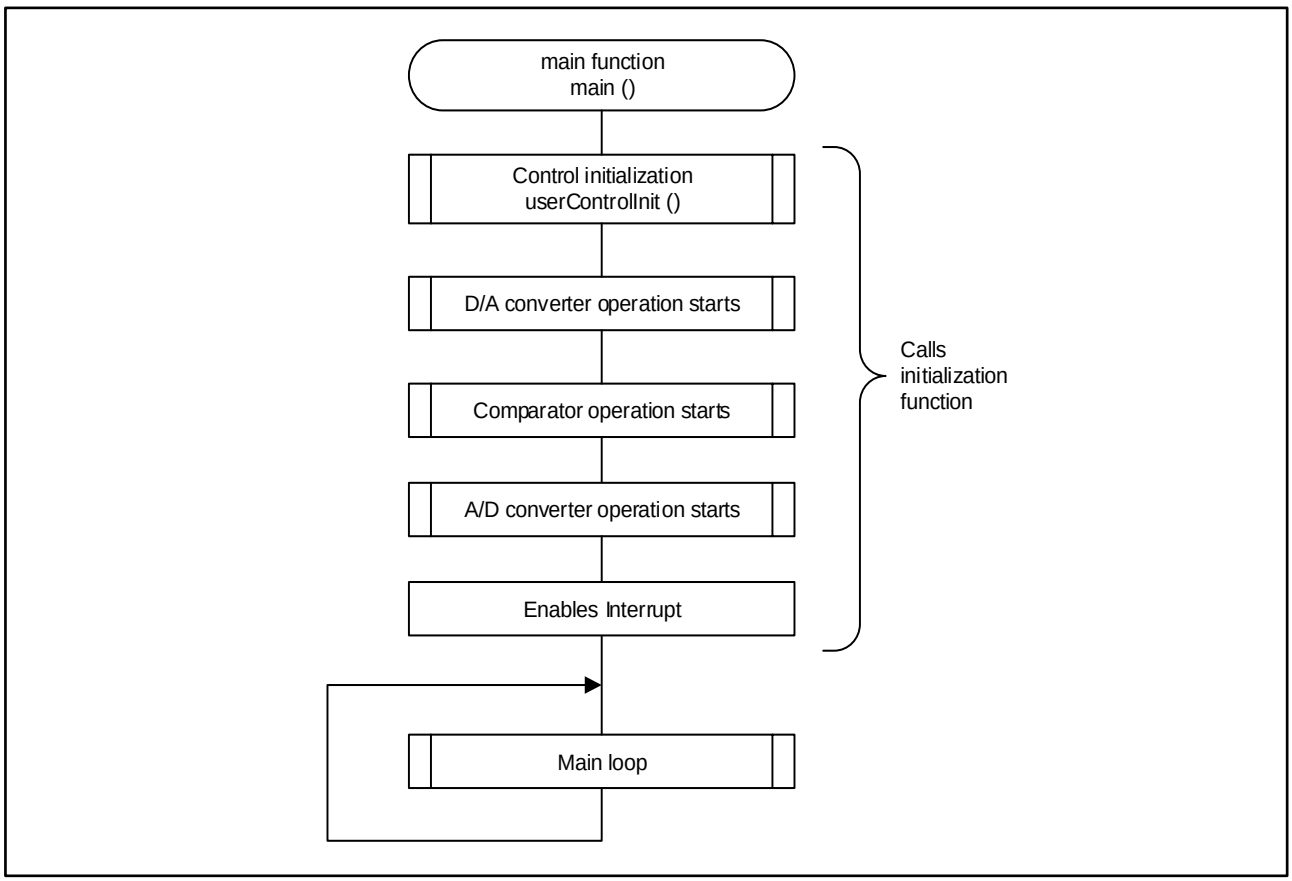


Figure 4-10. Main Function

4.6.2 Controller Initialization Function

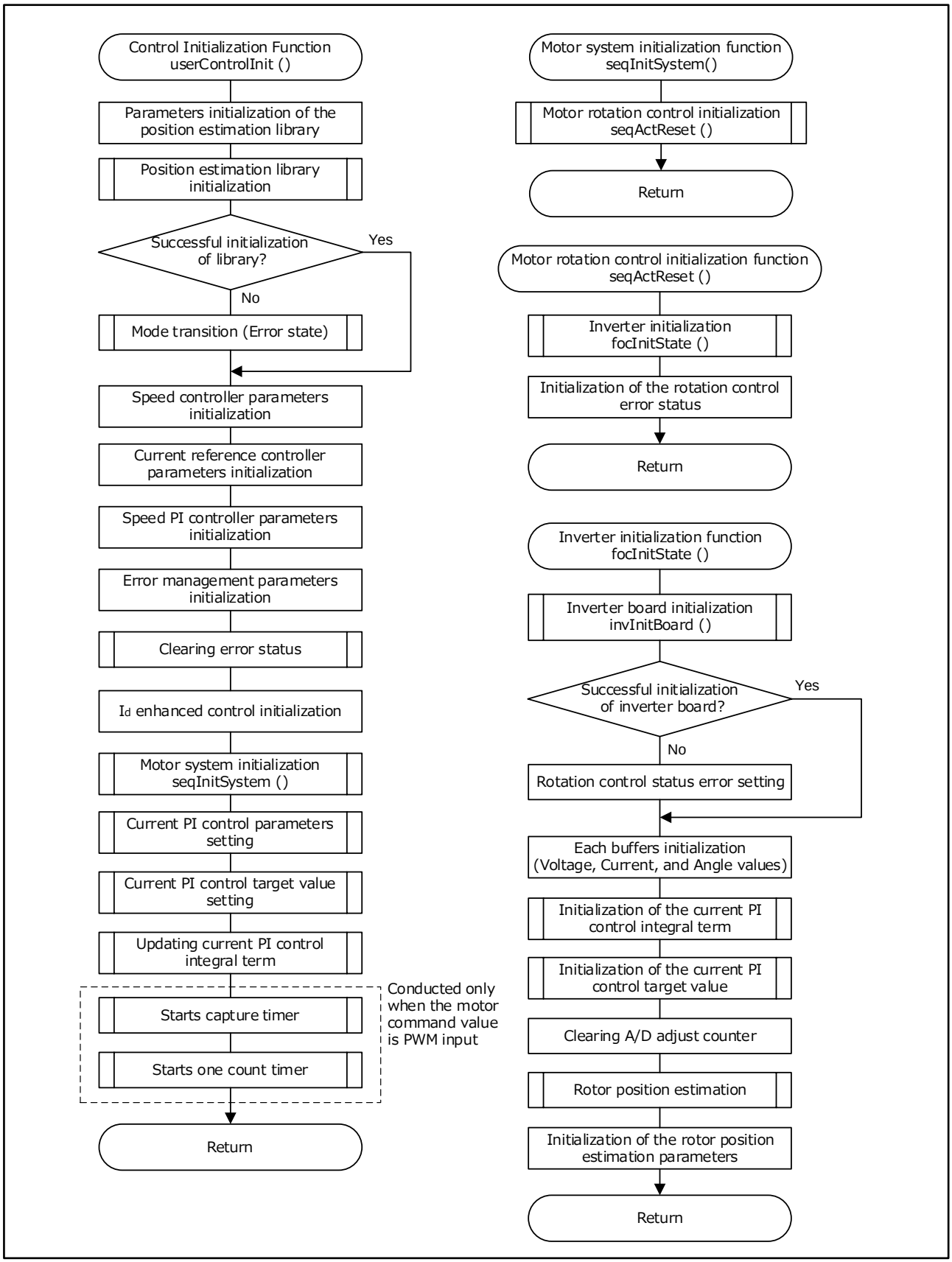


Figure 4-11. Control Initialization Function

4.6.3 Inverter Board Initialization Function

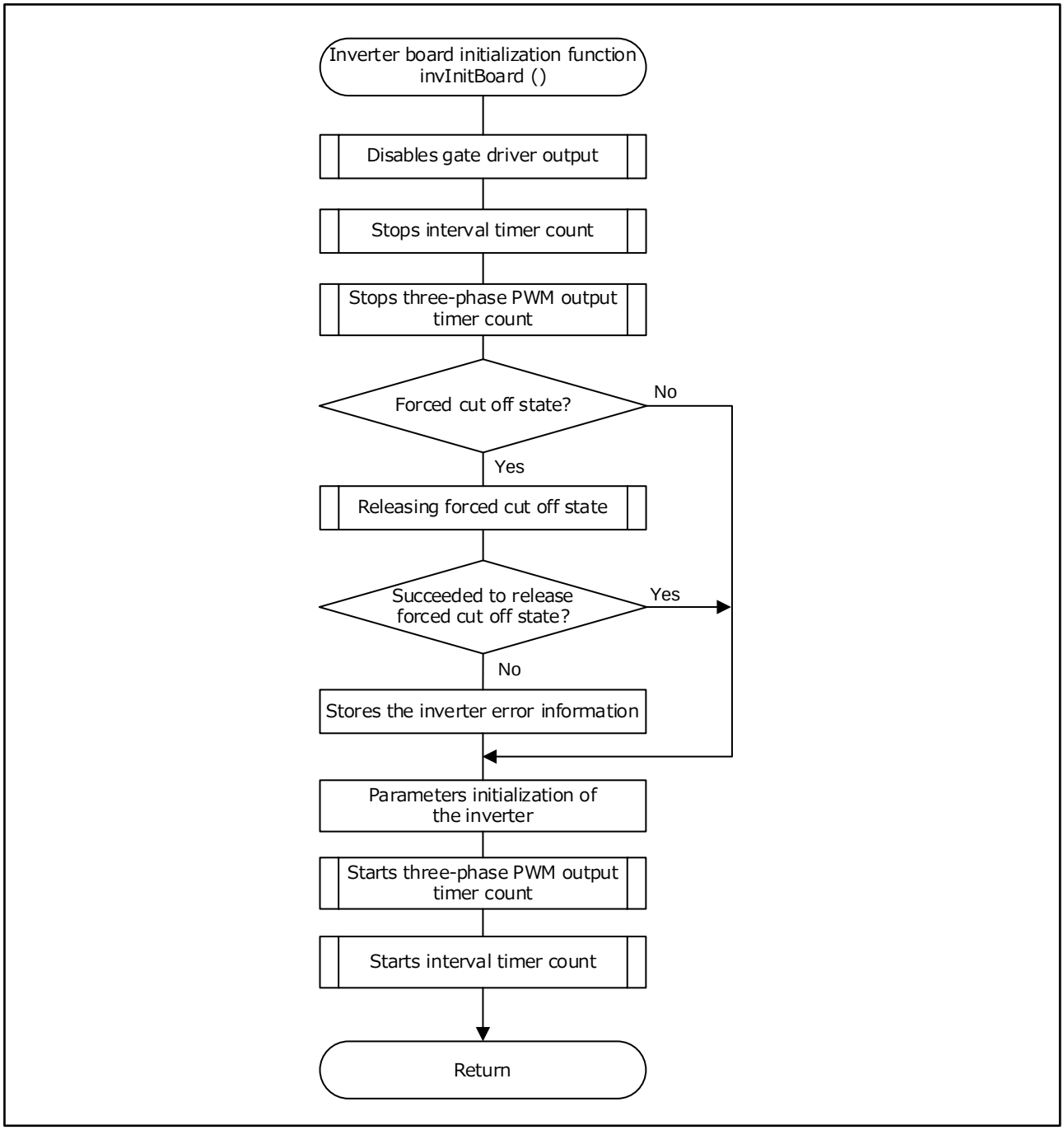


Figure 4-12. Inverter Initialization Function

4.6.4 Interval Timer Interrupt Handler

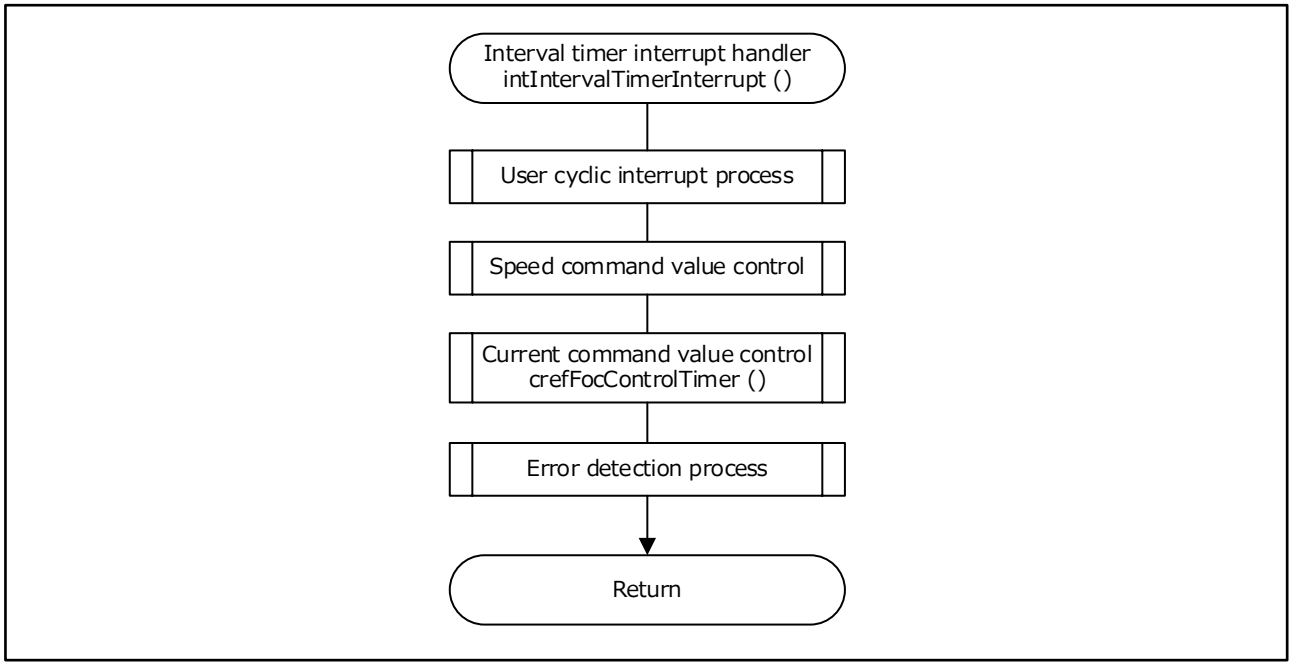


Figure 4-13. Interval Timer Interrupt Handler

4.6.5 Current Reference Control Function

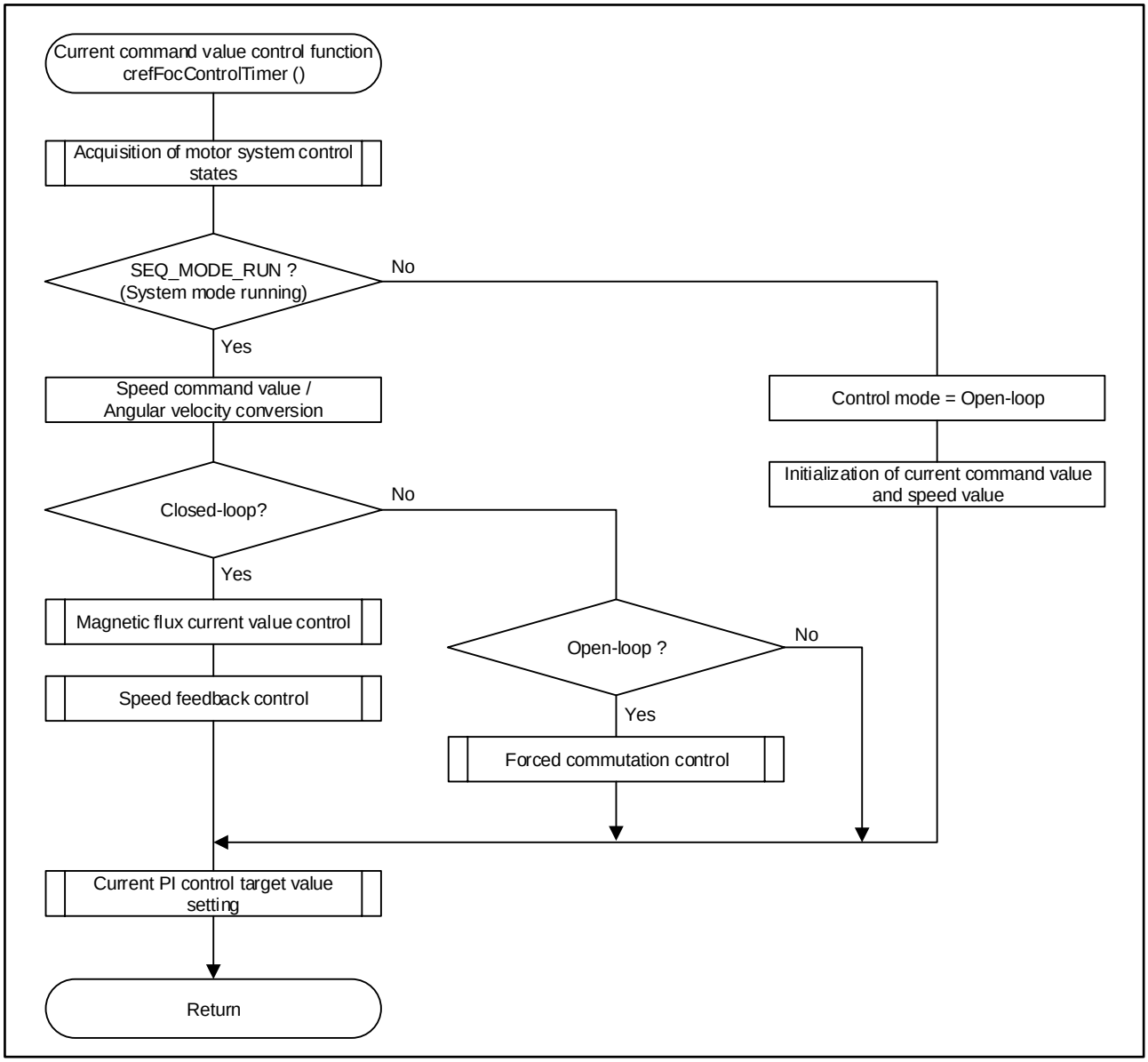


Figure 4-14. Current Command Value Control

4.6.6 Three-phase PWM Carrier Frequency Interrupt Handler

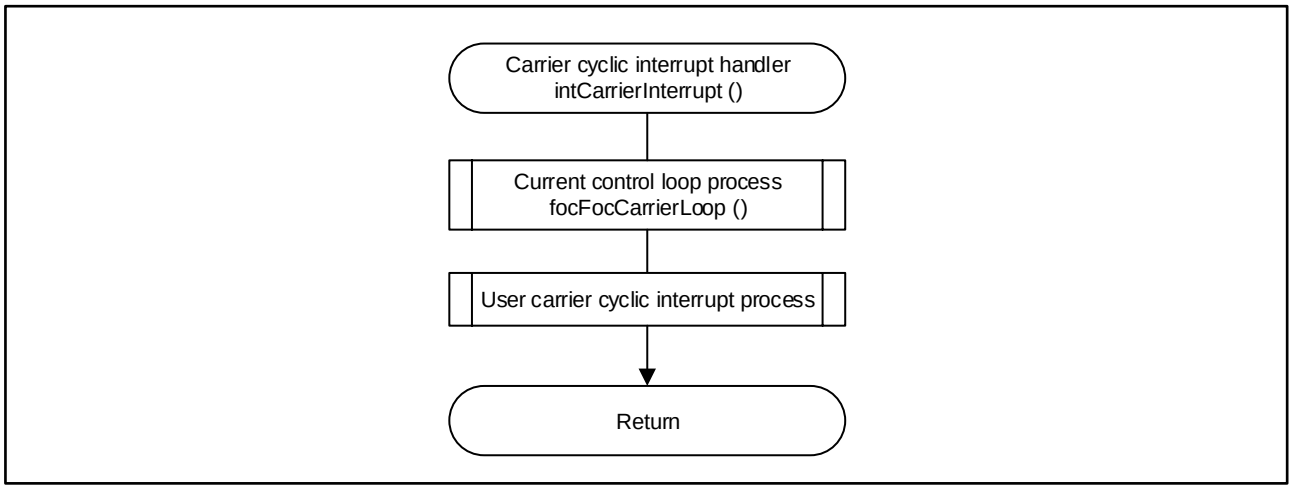


Figure 4-15. Carrier Cyclic Interrupt Handler

4.6.7 Current Control Loop Processing Function

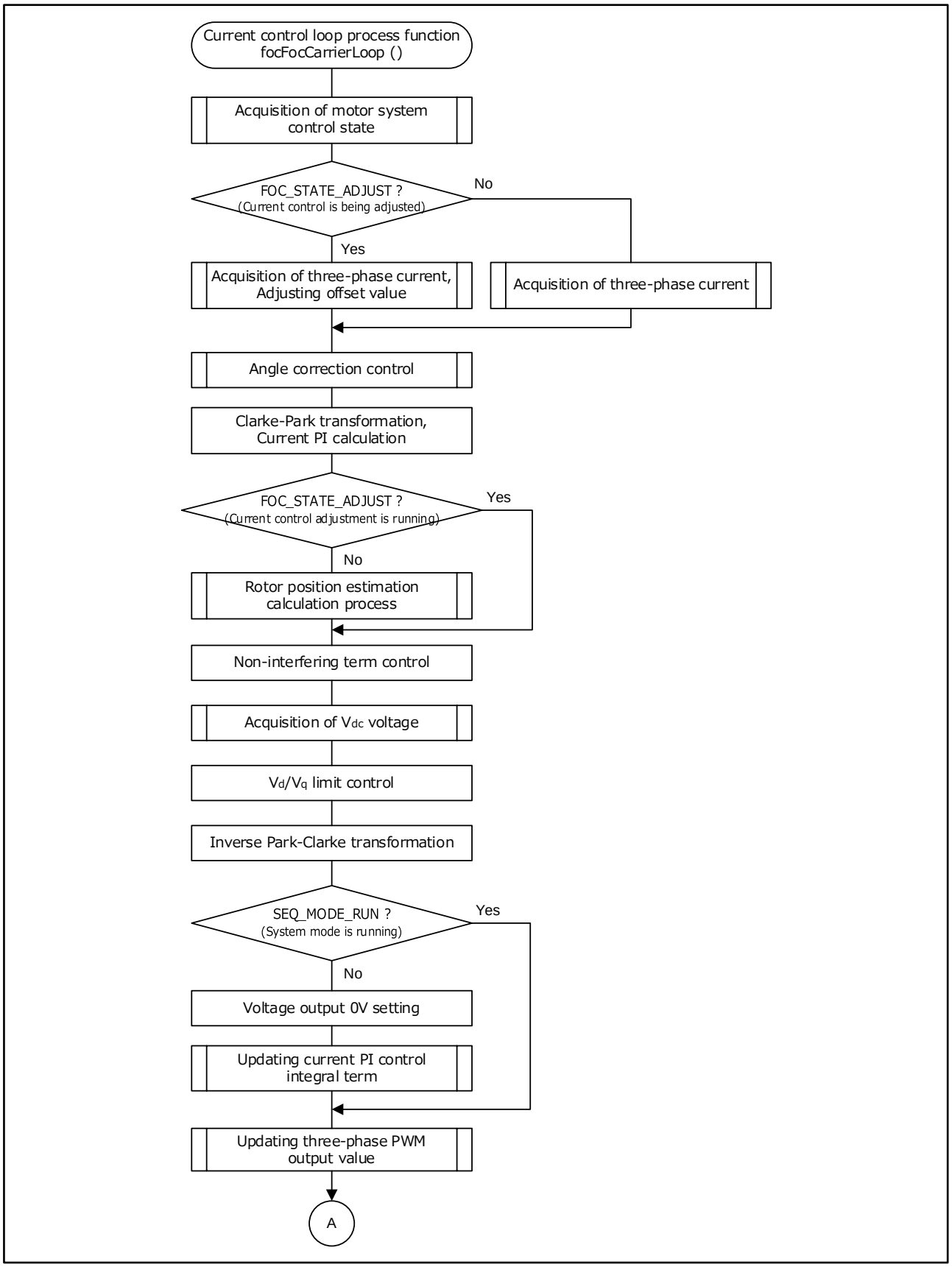


Figure 4-16. Current Control Loop Process (1/2)

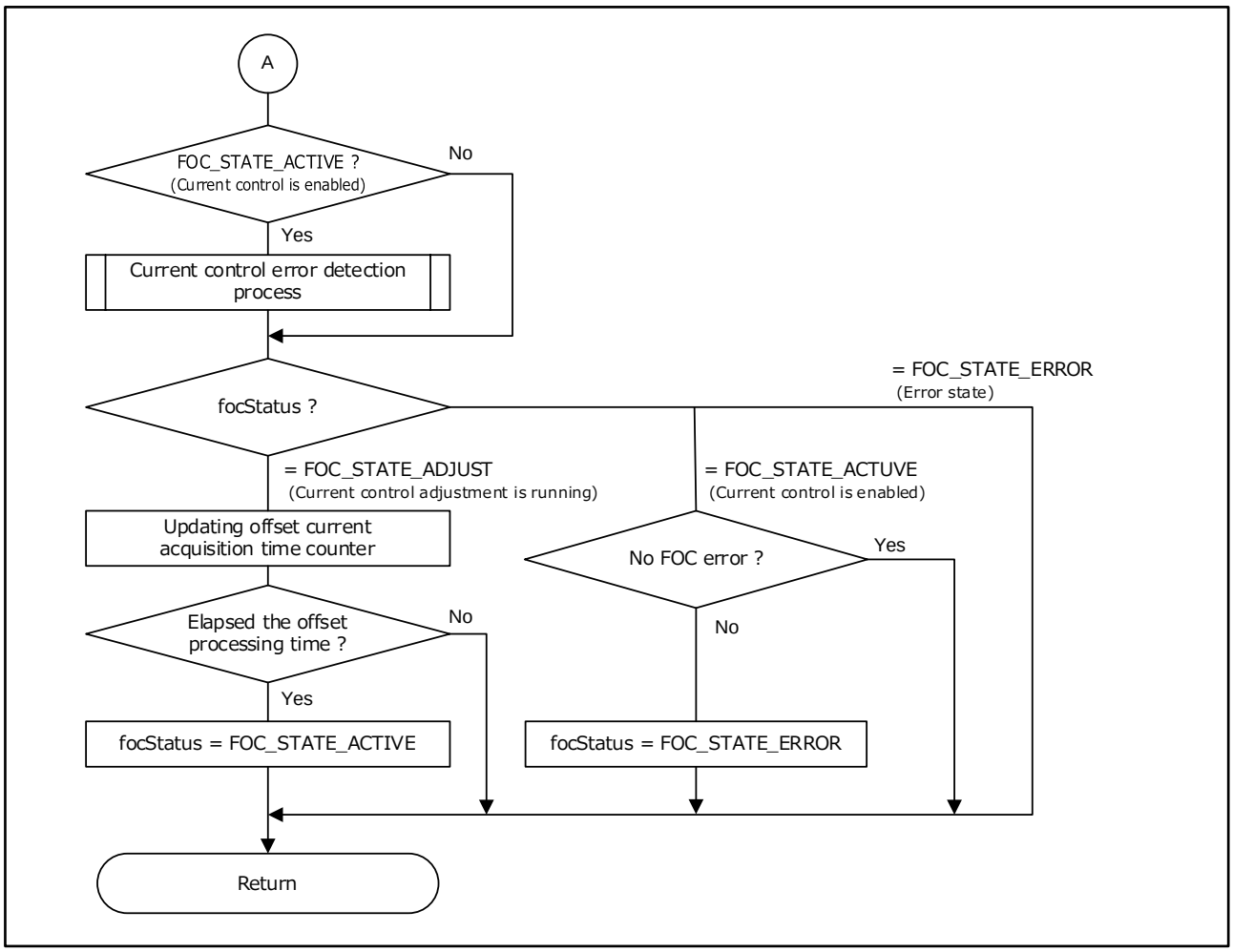


Figure 4-16. Current Control Loop Process (2/2)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	2022. 9.30	–	1st edition issued
1.10	2023. 6.30	P.13, 14	Updated software version from V.1.04 to V.1.05.

		P.39	Corrected calculation in function ctrlPiControl ().
			Before:
			obj->ibuffer = ((int32_t)refi << 16U) + (int32_t)((int16_t) obj->ibuffer);
			After:
			obj->ibuffer = (((int32_t)refi << 16U) & 0xFFFF0000) | (obj->ibuffer&0x0000FFFF);

		P.46	Added a note to hdwinit () function.

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.

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/.