

# **TMPM4KN**

## **Motor Sample Software**

### **Application Note**

Arm and Keil are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. Other company names, product names, and service names may be trademarks of their respective companies.

## Table of Contents

Table of Contents .....	2
1. Overview .....	7
1.1. Introduction .....	7
1.2. Specifications .....	7
1.3. Process Overview.....	9
1.4. Evaluation Board Configuration .....	10
1.4.1. SBK-M4KN Board Configuration.....	10
1.4.2. KES-P2 Board Configuration .....	10
1.4.3. KES-PH Board Configuration.....	10
1.4.4. Connection between the SBK-M4KN and KES-P2 (KES-PH).....	11
1.4.5. DAC Output Connection Configuration .....	12
1.4.6. Board Modification (KES-PH) .....	13
2. System Block Diagram .....	14
3. Source File Configuration .....	15
3.1. DAC Output .....	16
3.2. User Interface .....	17
4. Module Configuration .....	18
5. Microcontroller Hardware Assignment .....	19
5.1. IP.....	19
5.2. Interruption.....	19
6. General Flow .....	20
6.1. Main Routine (main) .....	20
6.2. Main Loop (main_loop).....	21
6.3. Interruption.....	22
6.3.1. VE Vector Processing (INT_VectorControl_byVE) .....	23
7. Motor Operation Status Transition (stage) .....	24
8. User Application.....	25
8.1. User Control.....	25
8.1.1. Get AD Value (soft_adc_getdata).....	26
8.1.2. Key Control (Uikeyscan) .....	26
8.1.3. User Setup (user_interface).....	26
8.1.4. LED Display Control (led_display) .....	26
8.1.5. UART Transmitting Data Setup (send_data_set).....	26
9. Functions.....	27
9.1. Control Commands.....	27
9.1.1. Control Method (usr.com_user) .....	27
9.1.2. Control Target Speed (usr.omega_user).....	27
9.1.3. Starting Current (usr.l <sub>d</sub> _st_user, usr.l <sub>q</sub> _st_user).....	27
9.2. Drive Command.....	28

9.2.1. Driving Method (drv.command) .....	28
9.2.2. Vector Control Command (drv.vector_cmd) .....	28
9.3. Driver Status .....	30
9.3.1. Error Status (drv.state) .....	30
9.4. Motor Control Structure .....	30
9.4.1. List of Variables .....	30
9.5. Function Details .....	34
9.5.1. Encoder Initial Setup (init_ENCen) .....	34
9.5.2. ADC Initial Setup (init_ADCen) .....	34
9.5.3. PMD Initial Setup (init_PMDen) .....	34
9.5.4. VE Initial Setup (init_VEen) .....	34
9.5.5. Motor Control Initial Setup (B_Motor_Init) .....	34
9.5.6. DAC Control Initial Setup (init_Dac) .....	36
9.5.7. Cycle Timer Initial Setup (init_Timer_interval4kHz) .....	36
9.5.8. User Control Initial Setup (init_user_control) .....	37
9.5.9. User Control (user_control) .....	37
9.5.10. User Motor Control (B_User_MotorControl) .....	38
9.6. Motor Control Function .....	38
9.6.1. Status Transfer Processing Function (C_Control_Ref_Model) .....	39
9.6.2. Motor Control Common Processing Function (C_Common) .....	40
9.6.3. Stop Stage Function (C_Stage_Stop) .....	40
9.6.4. Bootstrap Stage Function (C_Stage_Bootstrap) .....	41
9.6.5. Positioning Stage Function (C-Stage_Initposition) .....	42
9.6.6. Forced Commutation Stage Function (C_Stage_Force) .....	42
9.6.7. Forced Steady Changeover Stage Function (C_Stage_Change_up) .....	44
9.6.8. Steady Stage Function (C_Stage_Steady_A) .....	45
9.6.9. Protection Stage Function (C_Stage_Emergency) .....	45
9.6.10. Shift PWM Control (C_ShiftPWM_Control) .....	46
9.7. Motor Drive Function .....	47
9.7.1. Explanation of Terms .....	47
9.7.2. Motor Current, Power Supply Voltage Gain (VE_GetdataFromVEreg) .....	47
9.7.3. Clarke Conversion (E_Clarke) .....	51
9.7.4. Park Conversion (E_Park) .....	52
9.7.5. Position Estimation Function (D_Detect_Rotor_Position) .....	53
9.7.6. Encoder Control Function (H_Encoder) .....	55
9.7.7. Speed Control Function (D_Control_Speed) .....	57
9.7.8. Inverse Park Conversion (E_InvPark) .....	58
9.7.9. Sector Computation .....	59
9.7.10. Spatial Vector Modulation (D_SVM) .....	59
10. Explanation of Definitions for Constants .....	64
10.1. Motor Driver Setting Parameter (D-Para.h) .....	64

10.1.1. Motor Control Channel Selection .....	64
10.1.2. Selecting a DAC Output .....	64
10.1.3. Selection of Unit for Command Speed.....	64
10.1.4. Parameter List.....	64
10.2. Motor Driver Setting Parameter Motor Channel (D_Para_chx.h) x = 0, 1, 2.....	65
10.2.1. Selection of Control.....	65
10.2.2. Parameter List per Motor Channel.....	65
10.2.3. Relationship of Constant Setups and Waveform .....	71
10.3. Constants for User Control.....	72
11. Timings for Control and Data Update .....	74
11.1. Vector Control Using VE.....	74
11.1.1. 3-Shunt Control .....	74
11.1.2. 1-Shunt Control .....	76
12. Peripheral Driver.....	78
12.1. Vector Engine (A-VE+) .....	78
12.1.1. Specifications of Functions .....	78
12.1.2. Data Structure .....	85
12.2. Motor Control Circuit (PMD).....	85
12.2.1. Specifications of Functions .....	85
12.2.2. Data Structure .....	86
12.3. Analogue Digital Converter (ADC) .....	87
12.3.1. Specifications of Functions .....	87
12.3.2. Data Structure .....	87
12.4. Explanation of Constants .....	87
12.4.1. Constants for Microcontroller equipped with A--VE (mcuip_drv.h) .....	87
13. Temperature Measuring.....	93
13.1. Measuring Method.....	93
14. Status Check Monitor .....	94
14.1. Initial Display .....	94
14.2. Display during Rotation .....	94
14.3. DAC Mode Changeover Display .....	94
15. FAQ.....	95
15.1. When the EWARM project setup is deleted. ....	95
16. Appendix .....	101
16.1. Fixed Decimal Point Processing .....	101
16.2. Normalization.....	101
16.3. Data Format.....	101
16.4. Computation with Fixed Decimal Point .....	104
17. Revision History.....	105
<b>RESTRICTIONS ON PRODUCT USE.....</b>	<b>106</b>

## List of Figures

Figure 1.1	Vector Control Software Configuration .....	9
Figure 1.2	Controller Target Speed and Driver Target Speed .....	9
Figure 1.3	Addition of the LPF .....	11
Figure 1.4	Board Modification (KES-PH) .....	13
Figure 2.1	System Block Diagram .....	14
Figure 6.1	Main Routine .....	20
Figure 6.2	Main Loop .....	21
Figure 6.3	Interruption .....	22
Figure 6.4	VE Vector Processing .....	23
Figure 7.1	Motor Operation Status Transition .....	24
Figure 8.1	User Control .....	25
Figure 9.1	Motor Control Status Transition Diagram .....	39
Figure 9.2	Bootstrap .....	41
Figure 9.3	3-phase Modulation .....	47
Figure 9.4	2-phase Modulation .....	47
Figure 9.5	Clarke Conversion .....	51
Figure 9.6	Park Conversion .....	52
Figure 9.7	Position Estimation Block Diagram based on Induced Voltage $E_d$ of the d-Axis .....	54
Figure 9.8	Computation of Position and Speed using Encoder Pulse .....	56
Figure 9.9	Speed Control Block Diagram .....	58
Figure 9.10	Inverter Drive Circuit .....	60
Figure 9.11	Spatial Vector .....	60
Figure 9.12	Voltage Vector at Sector 0 .....	61
Figure 9.13	three-phase Modulation PWM Waveform .....	61
Figure 10.1	Starting Current Waveform (Positioned to electrical angle $0^\circ$ ) .....	71
Figure 10.2	How to Adjust Parameters during Voltage Drive .....	72
Figure 11.1	3-Shunt Control .....	74
Figure 11.2	1-Shunt Control .....	76
Figure 16.1	Example of 16-Bit Data .....	101

## List of Tables

Table 1.1	Motor Control .....	7
Table 1.2	SBK-M4KN Board Configuration .....	10
Table 1.3	KES-P2 Board Configuration .....	10
Table 1.4	DAC Output Connection Configuration .....	12
Table 3.1	DAC Mode Changeover .....	17
Table 5.1	IP .....	19
Table 5.2	Interruption .....	19

Table 9.1	Vector Control Command .....	28
Table 9.2	List of Variables .....	30
Table 9.3	List of Variables .....	35
Table 9.4	List of Variables .....	38
Table 9.5	List of Variables .....	39
Table 9.6	List of Variables .....	40
Table 9.7	List of Variables .....	40
Table 9.8	List of Variables .....	41
Table 9.9	List of Variables .....	42
Table 9.10	List of Variables .....	43
Table 9.11	List of Variables.....	44
Table 9.12	List of Variables .....	45
Table 9.13	List of Variables .....	45
Table 9.14	List of Variables .....	46
Table 9.15	Shift PWM Drive Method Conditions.....	46
Table 9.16	List of Variables .....	48
Table 9.17	Current Conversion Result.....	49
Table 9.18	Current Calculation.....	50
Table 9.19	Conversion Result .....	50
Table 9.20	Current Calculation.....	50
Table 9.21	List of Variables .....	53
Table 9.22	List of Variables .....	55
Table 9.23	List of Variables .....	57
Table 9.24	Sector Computation.....	59
Table 9.25	Inverter Switching Status.....	60
Table 9.26	Formulas for Duty On Duration per Sector .....	63
Table 9.27	Relationship of Sectors and Duties of Phases.....	63
Table 10.1	Parameter List .....	64
Table 10.2	Parameter List per Motor Channel .....	65
Table 11.1	Carrier Waveform.....	74
Table 11.2	Double Buffer Update Timing .....	74
Table 11.3	Interruption.....	75
Table 11.4	Carrier Waveform.....	76
Table 11.5	Double Buffer Update Timing .....	76
Table 11.6	Interruption.....	76
Table 12.1	PMD.....	87
Table 12.2	VE .....	91
Table 16.1	Single Precision (16-bit) Decimal Fraction Format .....	101
Table 16.2	Double Precision (32-bit) Decimal Fraction Format.....	103

## 1. Overview

### 1.1. Introduction

The operation of this motor control sample software has been checked using the SBK-M4KN (TMPM4KN microcontroller evaluation board) by ESP, along with the KES-P2 and KES-PH (inverter board). Please contact ESP (<http://esp.co.jp/>) for details on this evaluation board.

### 1.2. Specifications

- ◆ Applied Products            Brushless DC motor control software (using vector engine "VE")
- ◆ Microcontroller            TMPM4KNFYAFG
- ◆ Clock                        80 MHz
  
- ◆ Development Environment
  - IAR Embedded Workbench for ARM V9.70
  - KEIL μVision MDK V5.43
  - Segger Embedded Studio for ARM V8.24
  - Evaluation Board: ESP
    - SBK-M4KN + KES-P2
    - SBK-M4KN + KES-PH
  - Motor            KES-P2 : Tsukasa Electric Co., Ltd - TG611B  
KES-PH : Mitsubishi Electric FA Industrial Products Corporation - EM-AMF 0.1kW
  - DC link voltage                        KES-P2 : 24 VDC
  - Power Supply Voltage                KES-PH : 100 VAC
  
- ◆ Overview
  - Brushless DC Motor Vector Control
  - Speed Control
  - Evaluation Purpose
    - DAC Output (variable analogue output)
    - Inverter Board Temperature Detection
    - UART Output (initial status checking using Tera Term, etc.)
    - LED Output (Monitor LED)
  
- ◆ Motor Control

**Table 1.1 Motor Control**

Item	Control
Rotating Speed Control Method	Constant Rotating Speed Control (variable resistance operation)
Rotating Direction	CW/CCW (switch changeover)
Modulation	Three-phase modulation or two-phase modulation
Current Detection	Three-shunt or single shunt
Position Detection	Sensor-less
Op Amplifier	Built-In Microcontroller or External

## Precautions

When an EMG occurs, use "Reset" for releasing.

When operating with single shunt, check the setups as follows:

- D\_Para\_CHx.h

```
#define cSHUNT_TYPE_CHx (Note1)
```

```
#define cBOOT_TYPE_CHx (cBoot_v)
```

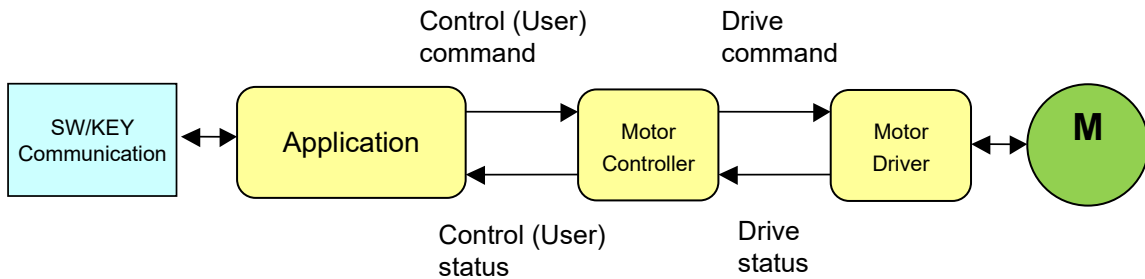
- B\_User.c

```
Motor_chx_usr.com_user.spwm = 1;
```

Note1: Choose a suitable number from "0-2" for "x" according to the channel of the motor used.

## 1.3. Process Overview

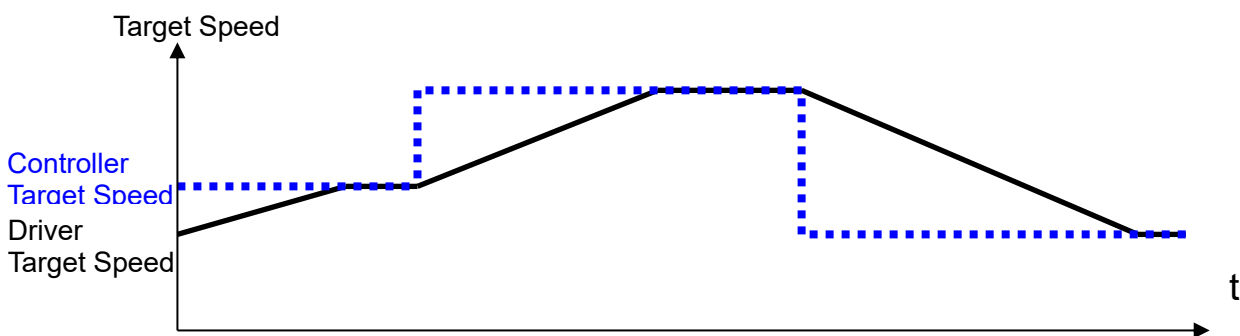
The vector control software consists of three layers: the application that takes care of the user interface processing, the motor controller that controls the motor operation status with the state transition and the motor driver that takes care of the motor drive processing by directly accessing the motor driver circuit.



**Figure 1.1 Vector Control Software Configuration**

1. The application will input the control commands that are set up through the switches, keys, communication, etc. and give it to the motor controller together with other control commands. Furthermore, it acquires the control status from the motor controller and as well as conducting the necessary processing, it outputs to the ports, etc.
2. The motor controller reads the control commands that are given from the application, converts it into the more specific drive commands according to the motor operating status, and gives it to the motor driver. Furthermore, it acquires the driver status from the motor driver and as well as conducting the necessary processing, transfers it to the application.
3. The motor driver reads the drive commands that are given from the motor controller and drives the motor. Furthermore, it monitors the motor operation and, as well as conducting the necessary processing according to the status, transfers the driver status to the motor controller.

For example, when a new control target speed is given from the application during the rotation of motor, it will be temporarily converted to the gradually varying driving target speed in the motor controller and be given to the motor driver because the motor driver cannot respond to an abrupt change in the target speed.



**Figure 1.2 Controller Target Speed and Driver Target Speed**

## 1.4. Evaluation Board Configuration

This software was developed using the following evaluation boards.

### 1.4.1. SBK-M4KN Board Configuration

Configure the jumpers to select the inverter board as the power supply source.

**Table 1.2 SBK-M4KN Board Configuration**

Jumper Name	Terminal name (Silk marking)	function	open	short	Configuration for Evaluation	
JP1	DAPEN	Uses onboard ICE (Short if using the on-board CMSIS-DAP)	Unused	use	short	
JP2	DnBOOT	M366-BOOT mode setting (Normally, leave this open.)	Prohibited	permission	open	
JP3	BOOT	M4KN-BOOT mode setting (Normally, leave this open.)	Prohibited	permission	open	
JP4	LVSFT	Level shift circuit enabled (Short when using CMSIS-DAP)	Prohibited	permission	short	
JP5 JP6 JP7	USB OBJT 5VIN	Microcomputer 5V power supply source selection Be sure to set JP5, JP6, and JP7 as shown in the table. (Only one jumper should be shorted.)	Power supply source JP5 short JP6 open JP7 open	USB2 (USB) short open open	USB1 (DAP) open short open	Inverter Board open open short
JP8	(VDD-AVDD)	Digital/analog 5V power supply separation	separation	common	common	
JP9	(GND-AGND)	Digital/analog GND separation	separation	common	common	

### 1.4.2. KES-P2 Board Configuration

Set the jumpers to either the 1-shunt or 3-shunt configuration to match the software settings.

**Table 1.3 KES-P2 Board Configuration**

Position detection	JPR	3-shunt system(initial)	1 shunt system	
3/1 shunt switching	JPR1	open	short	Be sure to match the settings on the software side.
	JPR2	open	open	
	JPR3	open	open	
	JPR4	open	open	

### 1.4.3. KES-PH Board Configuration

The KES-PH is a dedicated 3-shunt board. No jumper settings are required.

### 1.4.4. Connection between the SBK-M4KN and KES-P2 (KES-PH)

The SBK-M4KN and the KES-P2 are connected using the supplied 26-pin flat cable. Between the SBK-M4KN and the KES-PH, an approximately 8 kHz low-pass filter (LPF) is added to the motor current signals (ADU, ADV, ADW) and the reference voltage VDC, as shown in the figure below.

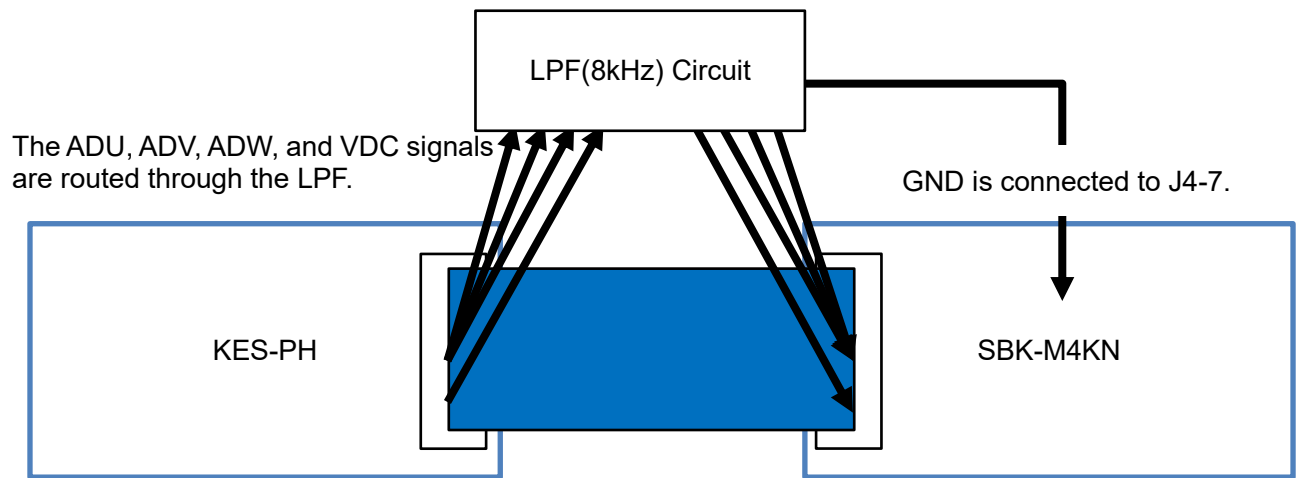


Figure 1.3 Addition of the LPF

## 1.4.5. DAC Output Connection Configuration

To use the DAC output function, it is necessary to connect the DAC board (KES-DAC12), which is supplied with the KES-P2, to the SBK-M4KN board.  
Connect them as shown in the table below.

**Table 1.4 DAC Output Connection Configuration**

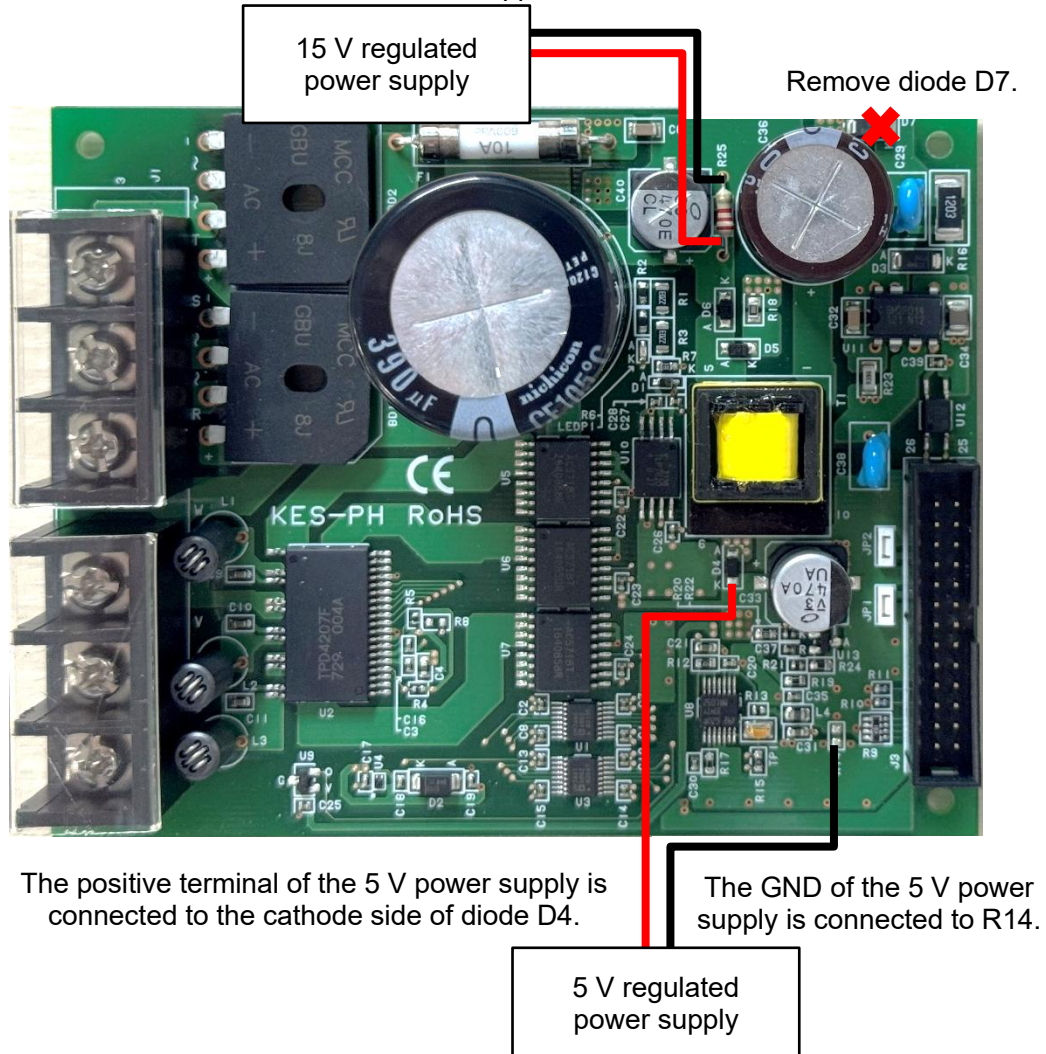
DAC board KES-DAC12 J28			SBK-M4KN board J4	
number	Signal Name		number	Signal Name
1	DVDD	→	5	VDD
2	GND	→	6	GND
			SBK-M4KN board J7	
3	TSP11RXD	→	3	TSP11CS1/PG1
4	TSP11TXD	→	4	TSP11TXD/PG5
5	TSP11CLK	→	6	TSP11SCK/PG6

## 1.4.6. Board Modification (KES-PH)

Disable the DC-DC converter on the KES-PH. (Remove diode D7 between the power supply voltage and the DC-DC converter.)

Use regulated power supplies for the 5 V and 15 V rails.

The GND of the 15 V power supply is connected to the PGND side of R25, and the positive terminal is connected to the opposite side.

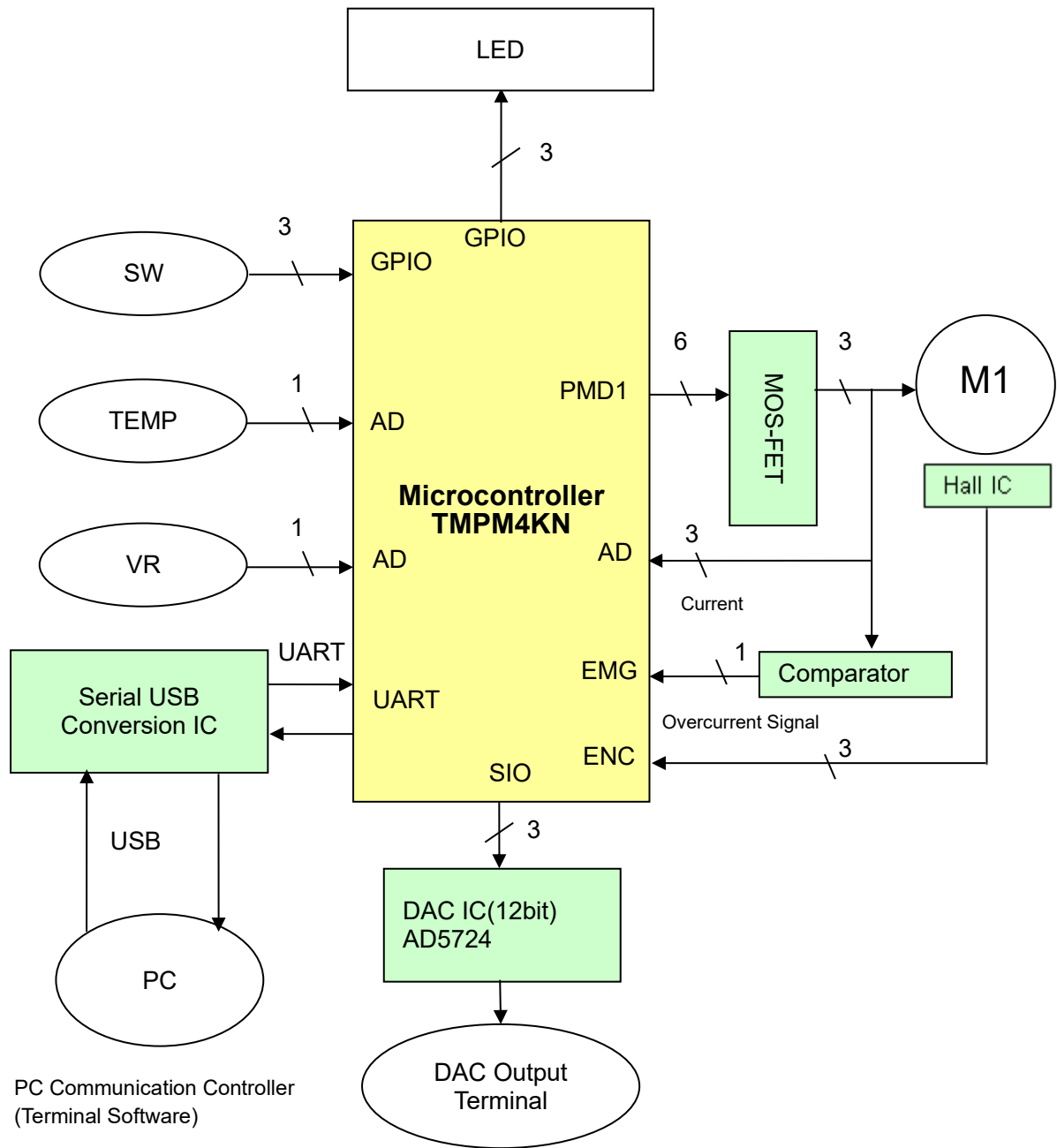


The positive terminal of the 5 V power supply is connected to the cathode side of diode D4.

The GND of the 5 V power supply is connected to R14.

Figure 1.4 Board Modification (KES-PH)

**2. System Block Diagram**



**Figure 2.1 System Block Diagram**

## 3. Source File Configuration

source	
B_User.c	Vector Control User Setup Source File
B_User.h	Vector Control User Setup Header File
C_Control.c	Vector Control Control Source File
C_Control.h	Vector Control Control Header File
D_Driver.c	Vector Control Driver Source File
D_Driver.h	Vector Control Driver Header File
D_Para.h	Vector Control Parameter (common to channels) Header File
D_Para_ch0.h	Vector Control Parameter (Channel 0) Header File (Note1)
D_Para_ch0_KES-P2.h	Vector Control Parameter (Channel 0) Header File (Note1)
D_Para_ch0_KES-PH.h	Vector Control Parameter (Channel 0) Header File (Note1)
D_Para_ch1.h	Vector Control Parameter (Channel 1) Header File
D_Para_ch2.h	Vector Control Parameter (Channel 2) Header File
dac_drv.c	DAC IC Driver Source File
dac_drv.h	DAC IC Driver Header File
E_Sub.c	Computing Function Source File
E_Sub.h	Computing Function Header File
H_Hall.c	Hall Sensor Function Source File
H_Hall.h	Hall Sensor Function Header File
initial.c	Microcontroller Initial Setup Source File
initial.h	Microcontroller Initial Setup Header File
interrupt.c	Interruption Control Source File
interrupt.h	Interruption Control Header File
ipdefine.h	Microcontroller Setup Header File
ipdrv.c	Microcontroller Hardware Reference Source File
ipdrv.h	Microcontroller Hardware Reference Header File
main.c	Main Routine
mcuip_drv.c	Microcontroller Hardware Setup Driver Source File
mcuip_drv.h	Microcontroller Hardware Setup Driver Header File
system_int.c	Interruption Function Source File
system_int.h	Interruption Function Header File
sys_macro.h	Macro Definition Header File
usercon.c	User Control Source File (Note2)
usercon_KES-P2.c	User Control Source File (Note2)
usercon_KES-PH.c	User Control Source File (Note2)
usercon.h	User Control Header File
Libraries	The library for CMSIS Core and individual IPs are stored.
CMSIS	CMSIS Core is stored.
system.c	Microcontroller Initial Setup Source File
system.h	Microcontroller Initial Setup Header File
TMPM4KNA.h	Microcontroller Register Definition Header File
configuration.h	Microcontroller Specific Function Configuration Header File
TSB_CMSIS_CORE.h	Microcontroller Register Definition Selection Header File
startup	
arm	Start Up Assembler Source (arm)
startup_TMPM4KHA.s	
startup_TMPM4KLA.s	
startup_TMPM4KMA.s	
startup_TMPM4KNA.s	
TMPM4KNFYAFG.scad	
TMPM4KNF10ADFG.scad	
iar	Start Up Assembler Source (IAR)
startup_TMPM4KHA.s	
startup_TMPM4KLA.s	
startup_TMPM4KMA.s	
startup_TMPM4KNA.s	
TMPM4KNFYAFG.icf	

```

    |
    |   TMPM4KNF10ADFG.icf
    |   └─segger           Start Up Assembler Source (Segger)
    |       SEGGER_THUMB_Startup.s
    |       TXZ4Aplus_M4K2_Startup.s
    |       TMPM4KHA_Vectors.s
    |       TMPM4KLA_Vectors.s
    |       TMPM4KMA_Vectors.s
    |       TMPM4KNA_Vectors.s
    |       TXZ4Aplus_M4K2_Flash.icf
    |
    └─IP_Driver   Peripheral Driver is stored.
        ipdrv_adc.c
        ipdrv_adc.h
        ipdrv_cg.c
        ipdrv_cg.h
        ipdrv_common.h
        ipdrv_enc.c
        ipdrv_enc.h
        ipdrv_siwdt.c
        ipdrv_siwdt.h
        ipdrv_t32a.c
        ipdrv_t32a.h
        ipdrv_tspi.c
        ipdrv_tspi.h
    
```

Note1: D\_Para\_ch0.h is initially set to the KES P2 configuration. For use with KES PH, rewrite it using the contents of D\_Para\_ch0\_KES PH.h.

Note2: usercon.c is initially set to the KES-P2 configuration. For use with KES-PH, rewrite it using the contents of usercon\_KES-PH.c.

### 3.1. DAC Output

Equipped with the DAC output function for viewing the fluctuation of variables with the Oscilloscope. For enabling the DAC output, enable the following definitions for D\_Para.h

```
#define    __USE_DAC
```

The variables for executing the DAC output are listed in the function UiOutDataStart() in the file usercon.c. Add a variable as necessary if any variable for checking is missing.

<<DAC Output Setup Variables>>

dac.select	Selecting a DAC Output
dac.motch	Set up a Motor CH for DAC Output
dac.datsft0 - 3	Set up the amount of left shifting (x) for the bit data. Formula (x): data x (2^x)

## 3.2. User Interface

The following functions are provided as the user interface:

<<User Interface>>

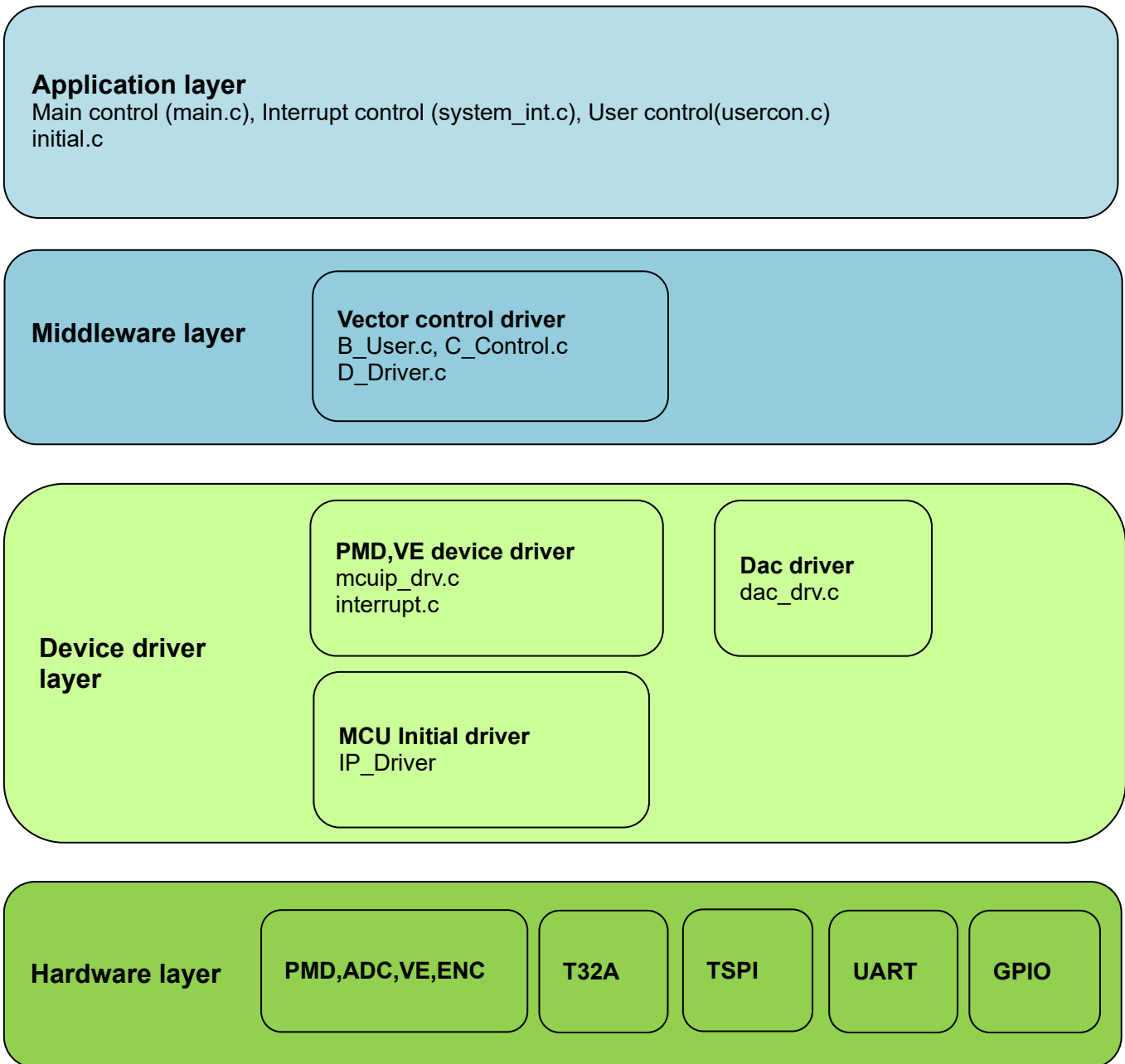
1. Rotating Direction Changeover  
Rotating direction of motor is changed over by switching the SW (S\_SW1).  
High: CW, Low: CCW
2. Phase Changeover  
Rotating direction of motor is changed over by switching the SW (S\_SW2).  
High: two-phase modulation, Low: three-phase modulation
3. Rotating Speed Control  
Rotating speed of motor is designated by operating VR1.  
Speed Range: 12 rps to 200 rps (electrical angle)
4. DAC Mode Changeover  
The DAC modes are changed over according to the table below by pressing the SW (USW1).

**Table 3.1 DAC Mode Changeover**

No.	Present Current	After Pressing a SW
1	Mode 0	Mode 1
2	Mode 1	Mode 2
3	Mode 2	Mode 3
4	Mode 3	Mode 0

5. LED Display  
The motor status is checked on the LED4 thru LED6.
  1. Rotating Status LED (LED4)  
Rotation Stop: Lights OFF  
Rotating (forced commutation): Lights ON  
Rotating (steady): Lights OFF
  2. VE Interruption Indicator LED (LED5)  
VE Interruption Processing: Lights ON  
Other: Lights OFF
  3. EMG Status LED (LED6)  
EMG No Detection: Lights OFF  
EMG Detection: Lights ON
6. Initial Status Display  
Initial status can be checked after releasing the reset.  
See 14 Status Check Monitor for details.

## 4. Module Configuration



## 5. Microcontroller Hardware Assignment

### 5.1. IP

**Table 5.1 IP**

IP	Control	Remarks	
TMRB	channel 0	4 kHz Interruption	
	channel 1	Not Used	
	channel 2	Not Used	
	channel 3	Not Used	
	channel 4	Not Used	
	channel 5	Not Used	
TSPI	channel 0	Not Used	
	channel 1	DAC IC Control	SIO Mode
UART	channel 0	PC Connection	UART0
	channel 1	Not Used	
	channel 2	Not Used	
	channel 3	Not Used	
ADC	Unit A	Get Motor Current and Temperature	
	Unit B	Not Used	
	Unit C	Rotating Speed Control VR	
PMD	channel 0	Motor CH0 Control	
	channel 1	Motor CH1 Control	
	channel 2	Motor CH2 Control	
VE	channel 0	Motor CH0 Control	
EMC	channel 0	Motor CH0 Encoder Signal Control	
	channel 1	Motor CH1 Encoder Signal Control	
	channel 2	Motor CH2 Encoder Signal Control	

### 5.2. Interruption

**Table 5.2 Interruption**

Cause	Process	Function
INTTB00IRQn	4 kHz Interval Timing Generation	INTTB00_IRQHandler
INTVCN0_IRQn	Vector Control Software Processing for CH0 (For vector control using VE)	INTVCN0_IRQHandler
INTADAPDA_IRQn	Vector Control Software Processing for CH0 (For Vector Control Using Software)	INTADAPDA_IRQHandler
INTSC0TX_IRQn	UART Transmission Complete Interruption Processing	INTSC0TX_IRQHandler
INTSC1TX_IRQn	DAC IC Control	INTSC1TX_IRQHandler

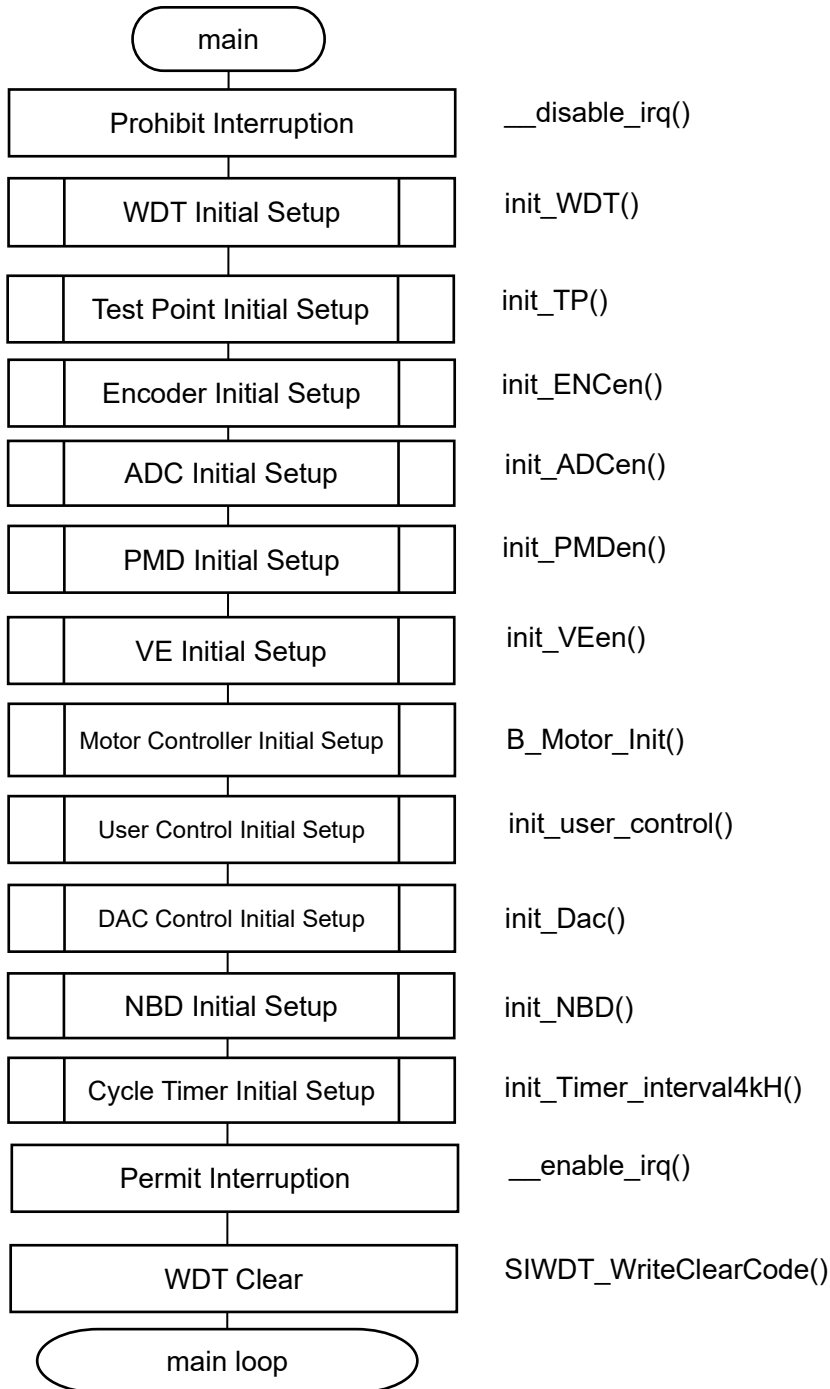
The priority of interruption may be changed using the following constants for ipdefine.h.

```
/* High Low */
/* 0 ---- 7 */
```

```
#define INT4KH_LEVEL      5      /* 4kH interval timer interrupt */
#define INT_VC_LEVEL     3      /* VE interrupt */
#define INT_ADC_LEVEL    3      /* ADC interrupt */
#define INT_DAC_LEVEL    6      /* SIO interrupt for Dac */
```

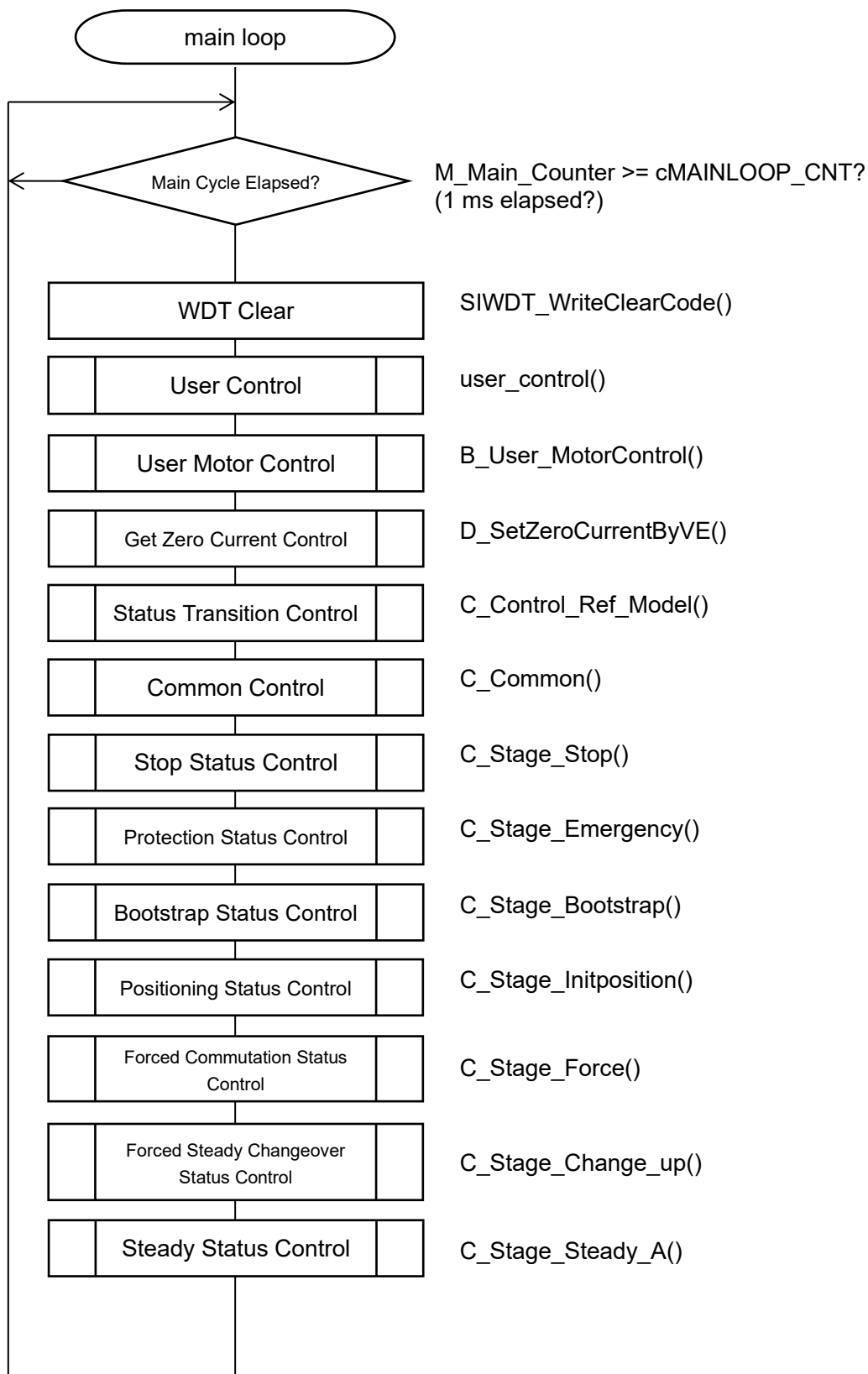
## 6. General Flow

### 6.1. Main Routine (main)



**Figure 6.1 Main Routine**

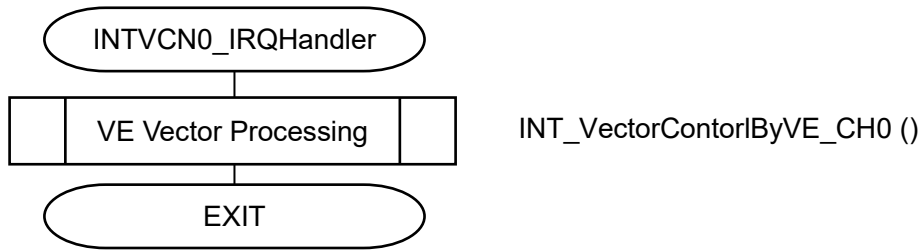
**6.2. Main Loop (main\_loop)**



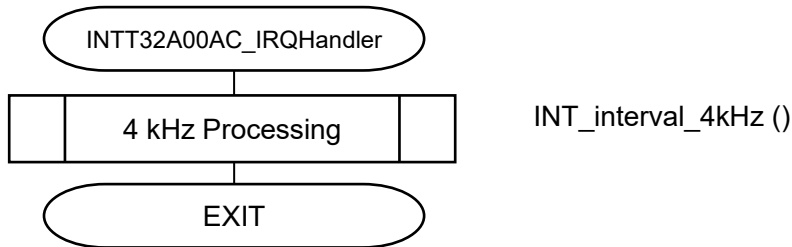
**Figure 6.2 Main Loop**

**6.3. Interruption**

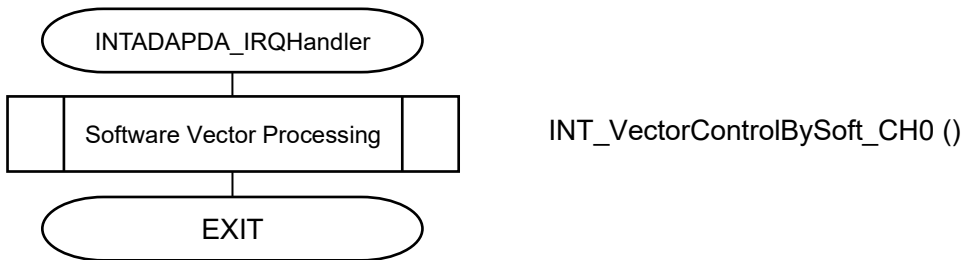
VE Schedule Complete



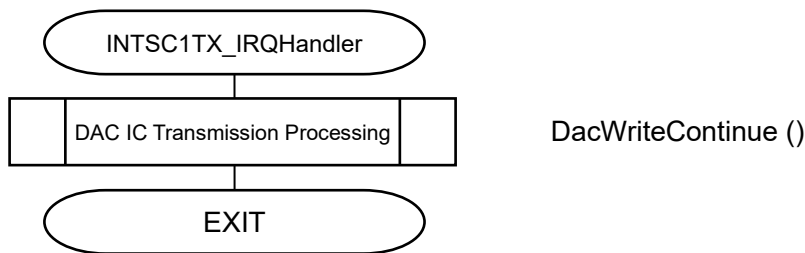
Every 4 kHz Interval



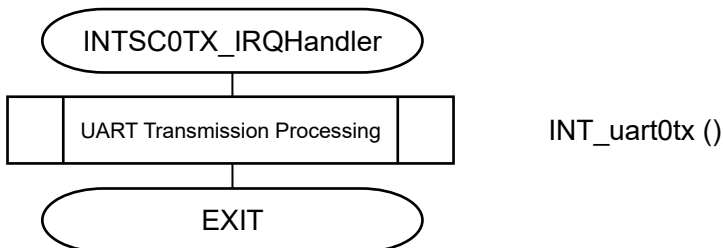
AD Conversion Complete Interrupt



DAC Communication 8-bit Transmission Complete



UART 8-bit Transmission Complete



**Figure 6.3 Interruption**

## 6.3.1. VE Vector Processing (INT\_VectorControl\_byVE)

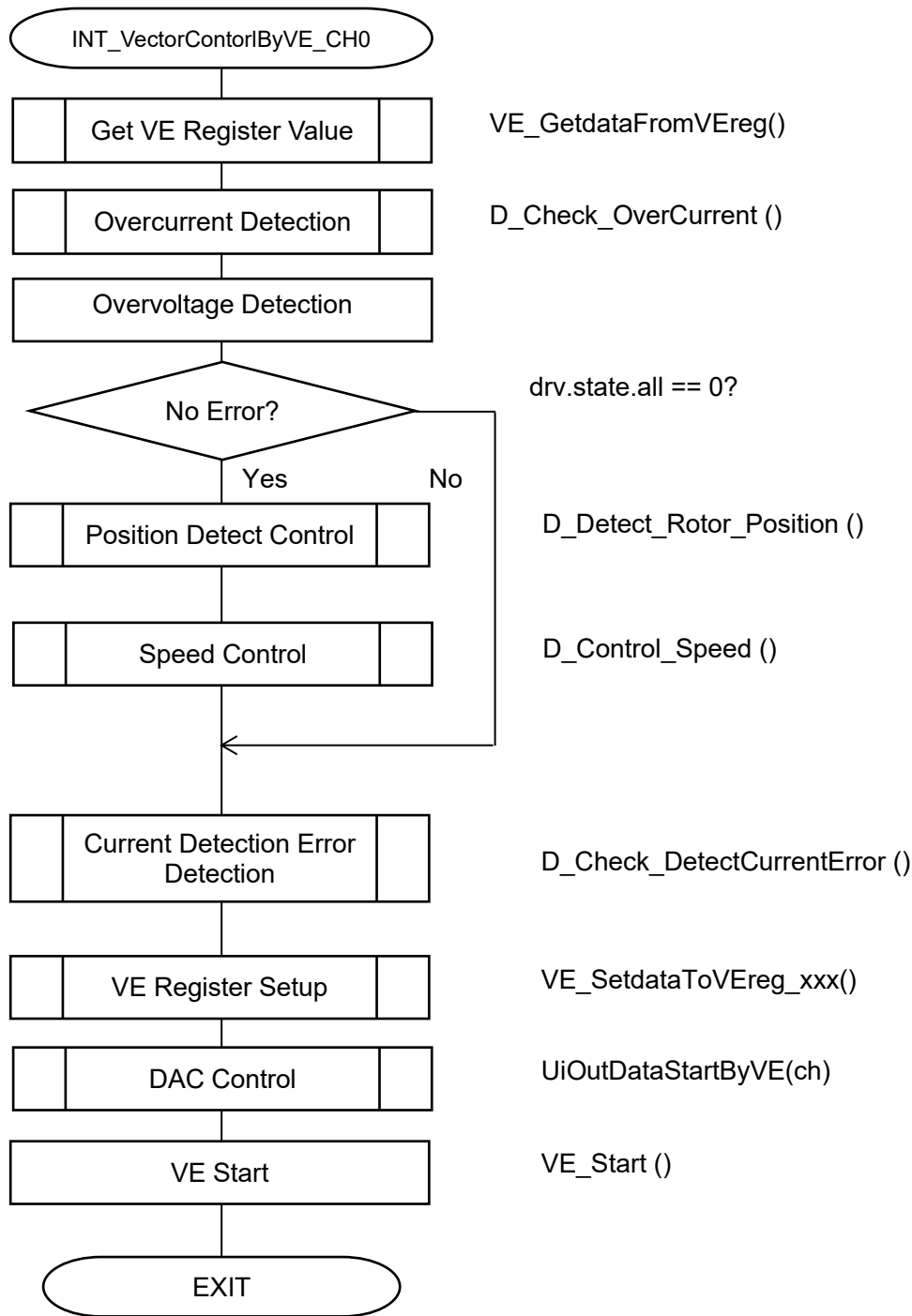
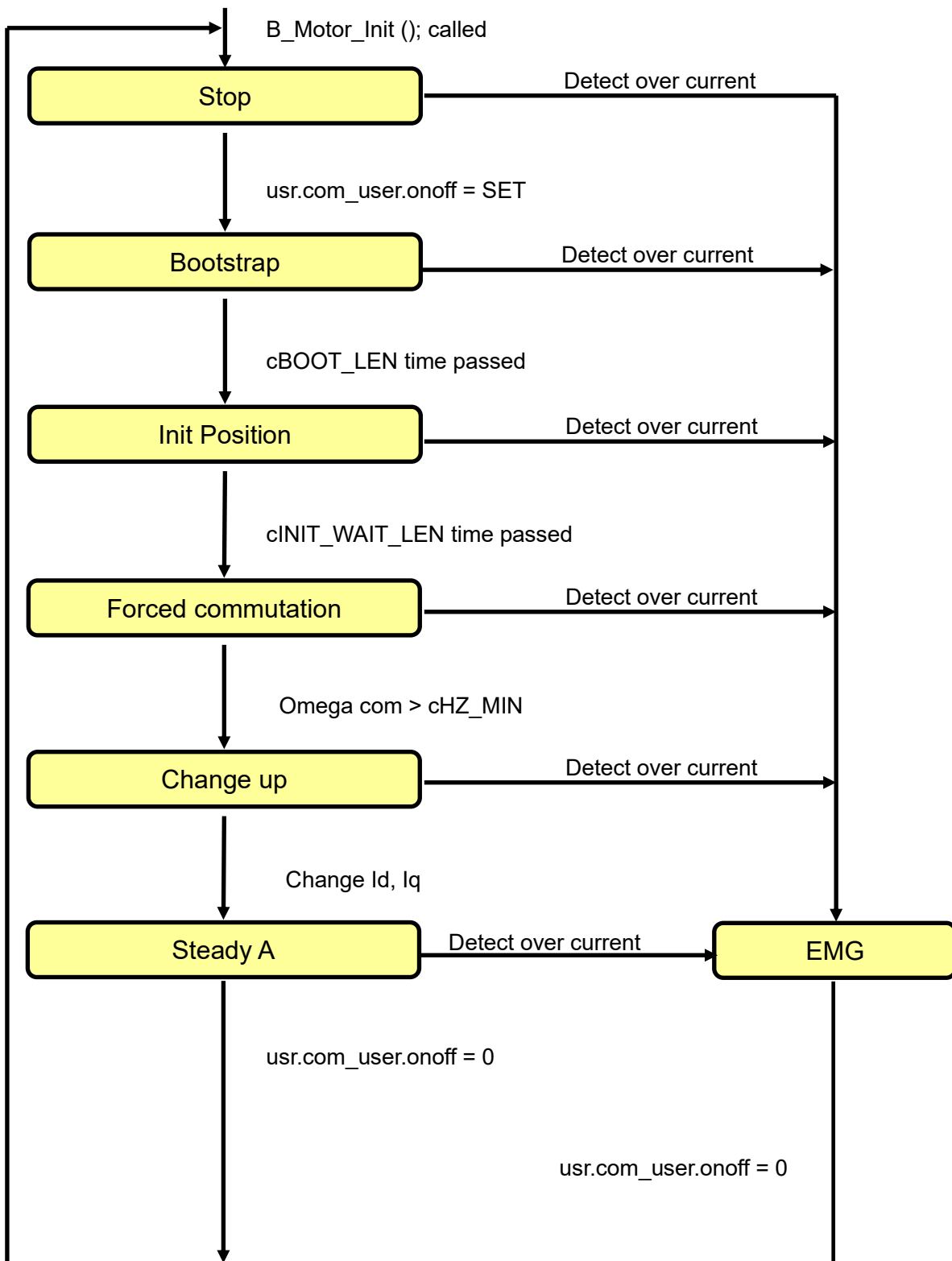


Figure 6.4 VE Vector Processing

**7. Motor Operation Status Transition (stage)**

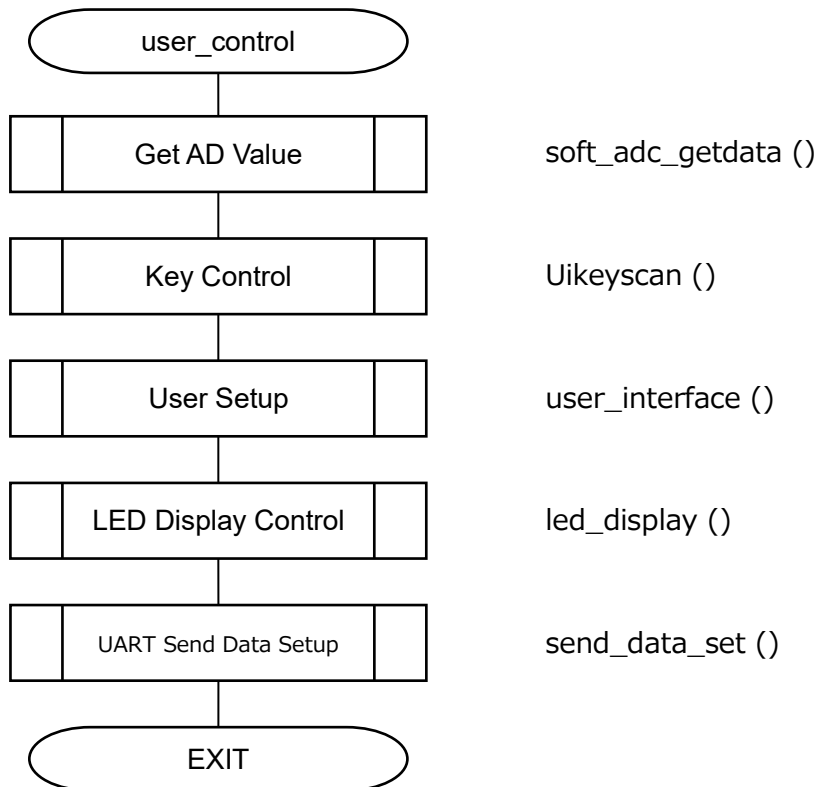


**Figure 7.1 Motor Operation Status Transition**

## 8. 8. User Application

### 8.1. User Control

Each of user application is processed once every lapse of main cycle (1 ms).



**Figure 8.1 User Control**

### 8.1.1. Get AD Value (soft\_adc\_getdata)

Get AD values of VR1 and Inverter Temperature (12 bit) with the individual conversion setup. After getting 10 times, the average of 8 times of data excluding the max. and min. for VR1 and the inverter temperature will be stored in ad\_vr.avedat and ad\_inv.avedat, respectively, as the effective values for each.

### 8.1.2. Key Control (Uikeyscan)

Process the key scan for S\_SW1, S\_SW2 and USW1

Each of the key data will be finalized when an identical value is continuously acquired 20 times and the S\_SW1 and S\_SW2 will be stored in the keydata and the USW1 in swdata.

### 8.1.3. User Setup (user\_interface)

Set up the following according to the VR1 input by the user operation.

1. Motor Speed Control  
Set the ad\_vr.avedata to the 8-bit resolution and the following are determined according to the value:  
Less than 0 x 10: 0 (0 Hz)  
0 x F0 and over: 200 (max. speed)  
0 x 10 to 0 x EF: Taken as an effective value and the speed is computed.
2. Motor Rotating Direction Changeover  
The setup of rotating direction rote\_dir according to the value of keydata.sw1.  
When keydata.sw1 = 1, rote\_dir: 1 (forward)  
When keydata.sw1 = 0, rote\_dir: 0 (reverse)
3. Modulation Type Changeover  
According to the value of keydata.sw2, the setup of modulation type Motor\_chx.usr.co\_user.modul is changed over (x = 0, 1).  
When keydata.sw2 = 1, Motor\_chx.usr.com\_user.modul: 1 (two-phase modulation)  
When keydata.sw2 = 0, Motor\_chx.usr.com\_user.modul: 0 (three-phase modulation)
4. DAC Mode  
Every time swdata.sw becomes "1," the setup of dac.select is changed over.

### 8.1.4. LED Display Control (led\_display)

The lighting of LED4 is controlled according to the rotating status.  
See 3.2 User Interface for details.

### 8.1.5. UART Transmitting Data Setup (send\_data\_set)

Set up the motor initial setups and the current rotating speed monitoring data.

The setups for communication are, 115,200 bps, data 8-bit, stop bit 1 bit, without parity and without flow control.

## 9. Functions

The following are the interface between the application and the motor controller, and the motor controller and the motor driver.

### 9.1. Control Commands

The control commands are listed as below:

#### 9.1.1. Control Method (usr.com\_user)

- Motor start and stop
- Encoder equipped or not
- Modulation Type (two-phase modulation, three-phase modulation)
- Shift PWM ON/OFF (Enabled only when VE is used and 1-shunt two-phase modulation)

```
typedef struct {
    uint16_t reserve: 12;      /* reserve */
    uint16_t spwm: 1;        /* Shift PWM 0=off 1=on */
    uint16_t modul: 1;       /* PWM Moduration 0=3phase modulation 1=2phase modulation */
    uint16_t encoder: 1;     /* Position detect 0=Current 1=Encoder */
    uint16_t onoff: 1;      /* PWM output 0=off 1=on */
} command_t;
```

```
command_t com_user;
```

usr.com\_user is set up as a control command in the application.

#### 9.1.2. Control Target Speed (usr.omega\_user)

```
q31_u omega_user;          /* [Hz/maxHz] OMEGA command, Q31 */
```

usr.omega\_user is set up as a control target speed in the application.

#### 9.1.3. Starting Current (usr.ld\_st\_user, usr.lq\_st\_user)

```
q15_t ld_st_user;         /* [A/maxA] d-Axis start current reference, Q15 */
```

```
q15_t lq_st_user;        /* [A/maxA] q-Axis start current reference, Q15 */
```

usr.ld\_st\_user and usr.lq\_st\_user are set up as the start current references in the application.

## 9.2. Drive Command

Drive Commands are listed as below:

### 9.2.1. Driving Method (drv.command)

```
command_t command;
```

The motor control will transfer the driving method to the motor control driver side via the drv.command.

### 9.2.2. Vector Control Command (drv.vector\_cmd)

A command for executing vector control computation and it administers the stage by stage processing.

```
typedef struct
```

```
{
  uint16_t reserve: 9;           /* reserve */
  uint16_t F_vcomm_theta: 1; /* Omega to Theta 0=command value 1=Calculate the theta from omega.*/
  uint16_t F_vcomm_omega: 1; /* Omega by 0=command value 1=Result of Estimation position */
  uint16_t F_vcomm_current: 1; /* Current by 0=command value 1=Result of Speed Control */
  uint16_t F_vcomm_volt: 1; /* Voltage by 0=command value 1=Result of Current Control */
  uint16_t F_vcomm_Edetect: 1; /* Position detect 0=off 1=on */
  uint16_t F_vcomm_Idetect: 1; /* Current detect 0=off 1=on */
  uint16_t F_vcomm_onoff: 1; /* Motor output 0=off 1=on */
} vectorcmd_t;
```

The vector control drive command (drv.vector\_cmd) is set up as below in each stage and a command is given to the motor drive.

In the vector control using VE, F\_vcomm\_volt or F\_vcomm\_Idetect are not used.

**Table 9.1 Vector Control Command**

cmd Stage	theta	omega	current	Volt	Edetect	Idetect	onoff
Stop	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR
Bootstrap	-	-	-	-	-	-	-
InitPosition	CLEAR	CLEAR	CLEAR	SET	CLEAR	SET	SET
Forced	SET	CLEAR	CLEAR	SET	SET	SET	SET
Change_up	SET	SET	CLEAR	SET	SET	SET	SET
Steady	SET	SET	SET	SET	SET	SET	SET
Emergency	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR

1. F\_vcomm\_theta  
For the rotor position estimation computation, the estimated value is adopted as the rotor position when SET. The reference value is adopted as the rotor position when CLEAR.
2. F\_vcomm\_omega  
For the rotor position estimation computation, the estimated value is adopted as the speed  $\omega$  when SET. The reference value is adopted as the speed  $\omega$  when CLEAR.
3. F\_vcomm\_current  
For the speed control, gives a command for the calculation method for the standard values of d- and q-axis currents.  
The value acquired using the speed deviation through the PI control is adopted as a standard value when SET. The PI control is not executed when CLEAR but the reference value is directly adopted as a standard value.
4. F\_vcomm\_volt

For the current control, the calculation methods for the d- and q-axis voltage standard values are given as a command.

The value acquired using the speed deviation through the PI control is adopted as a standard value when SET.

The PI control is not executed when CLEAR but the reference value is directly adopted as a standard value.

5. F\_vcomm\_Edetect

The induced voltage computation is executed when SET, and the rotor position estimation computation is executed.

The induced voltage computation is not executed when CLEAR, but the induced voltage is set to 0 and the reference value is adopted as the rotor position.

6. F\_vcomm\_Idetect

The Input coordinate axis conversion computation is executed when SET.

The computation is not executed when CLEAR, but the value is cleared.

## 9.3. Driver Status

### 9.3.1. Error Status (drv.state)

```
typedef union {
    struct {
        uint16_t reserve:11; /* reserve */
        uint16_t Loss_sync: 1; /* 0:normal, 1: Loss of synchronism */
        uint16_t emg_DC:1; /* 0:normal, 1: Over Vdc */
        uint16_t emg_I:1; /* 0:normal, 1: Current detect error */
        uint16_t emg_S:1; /* 0:normal, 1: Over current(soft) */
        uint16_t emg_H:1; /* 0:normal, 1: Over current(hard) */
    } flg;
    uint16_t all;
} state_t;
```

- |    |           |                                |   |
|----|-----------|--------------------------------|---|
| 1) | Loss_sync | Step-Out Detection             | SET when a step-out is detected (not implemented).                            |
| 2) | emg_DC    | Abnormal Voltage Detection     | SET when an abnormal voltage is detected.                                     |
| 3) | emg_I     | Current Detection Abnormality  | SET when a current detection abnormality is detected (not implemented).       |
| 4) | emg_S     | Software Overcurrent Detection | SET when an overcurrent is detected during the software processing.           |
| 5) | emg_H     | Hardware Overcurrent Detection | SET when an overcurrent is detected in the microcontroller hardware function. |

## 9.4. Motor Control Structure

Motor Control Structure (vector\_t) is defined in the ipdefine.h.

The variables are declared per motor channel as below:

e.g.)

```
vector_t    Motor_ch0; /* Motor data for ch0 */
vector_t    Motor_ch1; /* Motor data for ch1 */
vector_t    Motor_ch2; /* Motor data for ch2 */
```

### 9.4.1. List of Variables

**Table 9.2 List of Variables**

Type	Code	Description	Q Form at	Remarks
main_stage_e	stage.main	Main Stage	---	cStop cBootstrap cInitposition cForce cChange_up cSteady_A cEmergency
sub_stage_e	stage.sub	Sub Stage	---	cStep0 cStep1 cStep2 cStep3 cStepEnd
itr_stage_e	stage.itr	Interruption Stage	---	ciStop ciBootstrap ciInitposition_i ciInitposition_v ciForce_i ciForce_v ciChange_up ciSteady_A ciEmergency
q31_u	drv.omega_com	Driving Speed Reference	Q31	

		Value		
q31_u	drv.omega	Estimated Speed	Q31	
q15_t	drv.omega_dev	Speed Deviation	Q15	
q31_u	drv.ld_com	d-Axis Current Reference Value	Q31	
q31_u	drv.lq_com	q-Axis Current Reference Value	Q31	
q15_t	drv.ld_ref	d-Axis Current Standard Value	Q15	
q15_t	drv.lq_ref	q-Axis Current Standard Value	Q15	
q15_t	drv.ld	d-Axis Current	Q15	
q15_t	drv.lq	q-Axis Current	Q15	
q31_u	drv.lq_ref_l	q-Axis Current Integrated Value	Q31	
uint16_t	drv.theta_com	Electrical Angle Reference Value	Q0	
uint32_u	drv.theta	Rotor Position	Q0	
q15_t	drv.Vdc	Power supply voltage	Q15	
q31_u	drv.Vdc_ave	(Not Used)	---	
q31_u	drv.Vd	d-Axis Voltage	Q31	
q31_u	drv.Vq	q-Axis Voltage	Q31	
q15_t	drv.Vdq	(Not Used)	---	
q31_u	drv.Vdq_ave	(Not Used)	---	
q31_t	drv.Vd_out	Output Voltage	Q31	
q15_t	drv.Ed	d-Axis Induced Voltage	Q15	
q15_t	drv.Eq	(Not Used)	---	
q31_t	drv.Ed_l	d-Axis Induced Voltage Integrated Value	Q31	
q31_t	drv.Ed_PI	d-Axis Induced Voltage PI Value	Q31	
state_t	drv.state	Motor Abnormality Status	---	
command_t	drv.command	Driving Method	---	(See 9.2.1)
vectorcmd_t	drv.vector_cmd	Vector Control Command	---	(See 9.2.2)
q15_t	drv.spwm_threshold	Shift Changeover Speed	Q15	1-shunt only
uint16_t	drv.chkpls	Current Detection Protection Duty Width	Q0	
uint8_t	drv.idetect_error	Current Detection Status	---	0: Detection Enabled 1: Detection Disabled
q15_t	drv.la_raw	Phase-a Current (raw data)	Q15	
q15_t	drv.lb_raw	Phase-b Current (raw data)	Q15	
q15_t	drv.lc_raw	Phase-c Current (raw data)	Q15	
q15_t	drv.la	Phase-a Current (detection error protection)	Q15	
q15_t	drv.lb	Phase-b Current (detection error protection)	Q15	
q15_t	drv.lc	Phase-c Current (detection error protection)	Q15	
q31_u	drv.lao_ave	Phase-a Zero Current Average AD	Q0	
q31_u	drv.lbo_ave	Phase-b Zero Current Average AD	Q0	

q31_u	drv.lco_ave	Phase-c Zero Current Average AD	Q0	
uint8_t	drv.spdprd	Speed Control Cycle	Q0	
q15_t	drv.omega_enc	Encoder Speed	Q15	
q15_t	drv.omega_enc_raw	Encoder Speed	Q15	
q31_u	drv.omega_enc_ave	Encoder Average Speed	Q31	
uint32_t	drv.theta_enc	Encoder Angle	Q0	
int16_t	drv.EnCnt	Encoder Counter Value	Q0	
int16_t	drv.EnCnt1	Encoder Counter Previous Value	Q0	
int16_t	drv.EnCnt_dev	Encoder Counter Deviation	Q0	
q31_u	usr.omega_user	Controller Target Speed	Q31	
q15_t	usr.ld_st_user	Starting Id-Current	Q15	
q15_t	usr.lq_st_user	Starting Iq-Current	Q15	
uint16_t	usr.lambda_user	Initial Rotor Position	Q0	
command_t	usr.com_user	Control Method	---	(See 9.1.1)
command_t	usr.com_user_1	Control Method Previous	---	
q15_t	para.omega_min	Forced Commutation End Speed	Q15	
q15_t	para.omega_v2i	Current Control Changeover Speed	Q15	
q15_t	para.spwm_threshold	Shift PWM Changeover Speed	Q15	
q31_t	para.vd_pos	Positioning Reference Voltage	Q31	
q31_t	para.spd_coef	Output Voltage Factor	Q15	
q31_u	para.sp_ud_lim_f	Drive Speed Increase/Decrease Limit	Q31	
q31_u	para.sp_up_lim_s	Drive Speed Increase Limit	Q31	
q31_u	para.sp_dn_lim_s	Drive Speed Decrease Limit	Q31	
uint16_t	para.time.initpos	Positioning Time	Q0	
uint16_t	para.time.initpos2	Positioning Status Standby Time	Q0	
uint16_t	para.time.bootstp	Bootstrap Time	Q0	
uint16_t	para.time.go_up	Standby Time after Change Up	Q0	
q31_t	para.iq_lim	q-Axis Current Limit	Q31	
q31_t	para.id_lim	d-Axis Current Limit	Q31	
q15_t	para.err_ovc	Overcurrent Setup Value	Q15	
q31_t	para.pos.kp	Position Estimation Proportional Gain	Q15	
q31_t	para.pos.ki	Position Estimation Integration Gain	Q15	
int32_t	para.pos.ctrlprd	Position Estimation Control Cycle	Q16	
q31_t	para.spd.kp	Speed Control Proportional Gain	Q15	
q31_t	para.spd.ki	Speed Control Integration Gain	Q15	
uint8_t	para.spd.pi_prd	(Not Used)	Q0	
q31_t	para.crt.dkp	d-Axis Current Control Proportional Gain	Q15	

q31_t	para.crt.dki	d-Axis Current Control Integration Gain	Q15	
q31_t	para.crt.qkp	q-Axis Current Control Proportional Gain	Q15	
q31_t	para.crt.qki	q-Axis Current Control Integration Gain	Q15	
q31_t	para.motor.r	Motor Winding Resistance	Q15	
q31_t	para.motor.Lq	Motor q-Axis Inductance	Q15	
q31_t	para.motor.Ld	Motor d-Axis Inductance	Q15	
uint32_t	para.enc.pls2theta	Computation factor from the number of pulses to the angle	Q32	
q15_t	para.enc.pls2omega	Computation factor from the number of pulses to the speed	Q15	
int32_t	para.enc.plsnum	Number of encoder pulses	Q0	
int32_t	para.enc.ctrlprd	Encoder Control Cycle	Q16	
uint32_t	para.enc.deg_adjust	Electrical Angle Adjustment	Q0	
int32_t	para.delta_lambda	Current changeover phase when changing up	Q0	
uint16_t	para.chkpls	Current Detection Protection Duty Width	Q0	
uint32_t	stage_counter	Stage Counter	Q0	
shunt_type_e	shunt_type	Shunt Type	---	c1shunt c3shunt
boot_type_e	boot_type	Starting Type	---	cBoot_i cBoot_v

## 9.5. Function Details

### 9.5.1. Encoder Initial Setup (init\_ENCen)

#### 9.5.1.1. Syntax

void init\_ENCen(void)

Parameter:

No

Returned Value:

No

#### 9.5.1.2. Process

Initial setup of encoder circuit

- Encoder circuit setup
- Port setup

### 9.5.2. ADC Initial Setup (init\_ADCen)

#### 9.5.2.1. Syntax

void init\_ADCen(void)

Parameter:

No

Returned Value:

No

#### 9.5.2.2. Process

Initial setup of ADC

- Motor AD Setup
- ADC Permission

### 9.5.3. PMD Initial Setup (init\_PMDen)

#### 9.5.3.1. Syntax

void init\_PMDen(void)

Parameter:

No

Returned Value:

No

#### 9.5.3.2. Process

Initial Setup of PMD (programmable motor driver)

### 9.5.4. VE Initial Setup (init\_VEen)

#### 9.5.4.1. Syntax

void init\_VEen(void)

Parameter:

No

Returned Value:

No

#### 9.5.4.2. Process

Initial Setup of VE (vector engine)

- VE Interruption Setup
- VE Setup

### 9.5.5. Motor Control Initial Setup (B\_Motor\_Init)

#### 9.5.5.1. Syntax

void B\_Motor\_Init(void)

Parameter:  
No  
Returned Value:  
No

## 9.5.5.2. Variable

**Table 9.3 List of Variables**

Direction	Code	Description	Q Format	Remarks
Output	shunt_type	Shunt Type	---	
	boot_type	Driver Type	---	
	usr.com_user.encoder	Encoder Control	---	
	stage.main	Main Stage	---	
	stage.sub	Sub Stage	---	
	drv.lao_ave	Phase-u Zero Current Average AD	Q0	
	drv.lbo_ave	Phase-v Zero Current Average AD	Q0	
	drv.lco_ave	Phase-w Zero Current Average AD	Q0	
	usr.lq_st_user	Starting Iq-Current	Q15	
	usr.ld_st_user	Starting Id-Current	Q15	
	usr.lambda_user	Initial Rotor Position	Q0	
	para.motor.r	Motor Winding Resistance	Q15	
	para.motor.Lq	Motor q-Axis Inductance	Q15	
	para.motor.Ld	Motor d-Axis Inductance	Q15	
	para.spwm_threshold	Shift PWM Changeover Speed	Q15	
	para.chkpls	Current Detection Protection Duty Width	Q0	
	para.vd_pos	Positioning Reference Voltage	Q31	Voltage Start only
	para.spd_coef	Output Voltage Factor	Q15	Voltage Start only
	para.sp_ud_lim_f	Drive Speed Increase/Decrease Limit	Q31	
	para.sp_up_lim_s	Drive Speed Increase Limit	Q31	
	para.sp_dn_lim_s	Drive Speed Decrease Limit	Q31	
	para.time.bootstp	Bootstrap Time	Q0	
	para.time.initpos	Positioning Time	Q0	
	para.time.initpos2	Positioning Status Standby Time	Q0	
	para.time.go_up	Standby Time after Change Up	Q0	
	para.omega_min	Forced Commutation End Speed	Q15	
	para.omega_v2i	Current Control Changeover Speed	Q15	
	para.delta_lambda	Current changeover phase when changing up	Q0	
	para.pos.ki	Position Estimation Integration Gain	Q15	
	para.pos.kp	Position Estimation Proportional Gain	Q15	
	para.pos.ctrlprd	Position Estimation Control Cycle	Q16	
	para.spd.ki	Speed Control Integration Gain	Q15	
	para.spd.kp	Speed Control Proportional Gain	Q15	
	para.crt.dki	d-Axis Current Proportional Gain	Q15	
	para.crt.dkp	d-Axis Current Integration Gain	Q15	
para.crt.qki	q-Axis Current Proportional Gain	Q15		
para.crt.qkp	q-Axis Current Integration Gain	Q15		
para.iq_lim	q-Axis Current Limit	Q31		
para.id_lim	d-Axis Current Limit	Q31		

para.err_ovc	Overcurrent Setup Value	Q15	
para.enc.pls2theta	Computation factor from the number of pulses to the angle	Q32	
para.enc.deg_adjust	Electrical Angle Adjustment	Q0	
para.enc.plsnum	Number of encoder pulses	Q0	
para.enc.pls2omega	Computation factor from the number of pulses to the speed	Q15	
para.enc.ctrlprd	Encoder Control Cycle	Q16	
para.TrigComp	Trigger Timing Compensation	Q15	

### 9.5.5.3. Process

Initializing the motor control parameters.  
Setup of the variable whose setups are not changed during the control.

### 9.5.6. DAC Control Initial Setup (init\_Dac)

#### 9.5.6.1. Syntax

```
void init_DAC(TSB_TSPI_TypeDef* const TSPIx)
```

Parameter:

TSB\_TSPI\_TypeDef\* const TSPIx: Select a TSPI address.

Returned Value:

No

#### 9.5.6.2. Process

Initial Setup of DAC IC Control

- SIO Permission
- Port Initial Setup
- SIO Initial Setup
- DAC IC Initialization Communication
- SIO Interruption Level Setup
- SIO Interruption Suspension Clear
- Transmission Interruption Permission

### 9.5.7. Cycle Timer Initial Setup (init\_Timer\_interval4kHz)

#### 9.5.7.1. Syntax

```
void init_Timer_interval4kHz(void)
```

Parameter:

No

Returned Value:

No

#### 9.5.7.2. Process

Initial setup of timer for generating the 4 kHz cycle timing.

## 9.5.8. User Control Initial Setup (init\_user\_control)

### 9.5.8.1. Syntax

void init\_user\_control(void)

Parameter:

No

Returned Value:

No

### 9.5.8.2. Process

Initial setups for executing the user control, such as the SW.

- Digital Port Initial Setup
- Analogue Port Initial Setup
- LED Initial Setup
- UART Initial Setup

## 9.5.9. User Control (user\_control)

### 9.5.9.1. Syntax

void user\_control(void)

Parameter:

No

Returned Value:

No

### 9.5.9.2. Process

Control according to the external statuses, such as VR and SW.

- Drive speed control using VR1
- Rotating direction changeover using S\_SW1 (\*)
- two-phase/three-phase modulation changeover using S\_SW2 (\*)
- DAC changeover using USW1
- Outputting the rotating speed information to UART
- Displaying the internal status on LED

\* The motor operation will be stopped immediately after a changeover of SW.  
Set the speed setup of VR to 0 (0 V) after stopping.

## 9.5.10. User Motor Control (B\_User\_MotorControl)

### 9.5.10.1. Syntax

void B\_User\_MotorControl(void)

Parameter:

No

Returned Value:

No

### 9.5.10.2. Variable

**Table 9.4 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	target_spd	Target Speed	---	User Command
Output	usr.omega_user	Controller Target Speed	Q31	
	usr.com_user.onoff	Motor ON/OFF	---	
	drv.state	Motor Abnormality Status	---	

### 9.5.10.3. Process

Setup Motor ON/OFF, rotating speed, driving method, etc.

Check overcurrent (hardware EMG) status.

## 9.6. Motor Control Function

The motor control processing is controlled using the following functions:

C\_Control\_Ref\_Model

C\_Common()

C\_Stage\_Stop()

C\_Stage\_Bootstrap()

C\_Stage\_Initposition()

C\_Stage\_Force()

C\_Stage\_Change\_up()

C\_Stage\_Steady\_A()

C\_Stage\_Emergency()

Each stage has the internal status (sub stages).

The internal status (sub stages) starts from Step0 and when the processing is complete until StepEnd, the stage will be transferred to the next.

When a motor abnormality is detected, it will be transferred to the protection stop status (emergency).

## 9.6.1. Status Transfer Processing Function (C\_Control\_Ref\_Model)

### 9.6.1.1. Syntax

```
void C_Control_Ref_Model(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.6.1.2. Variable

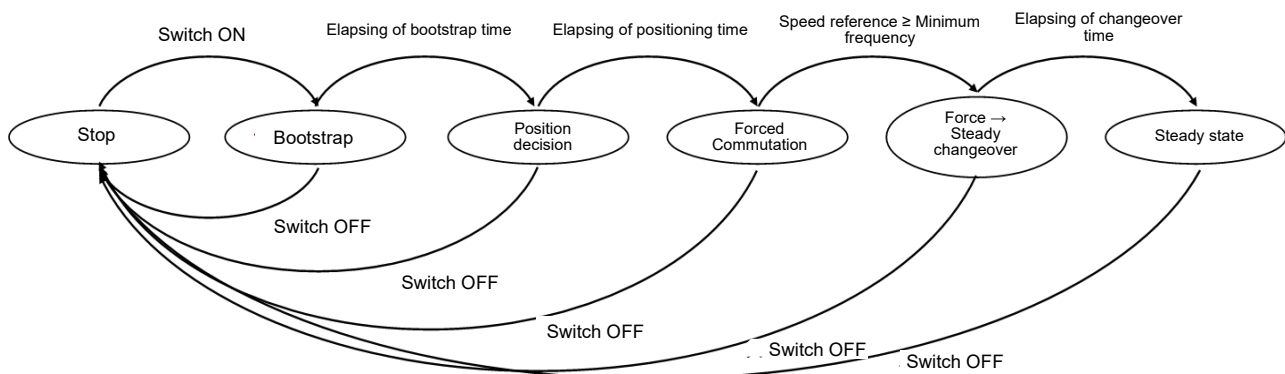
**Table 9.5 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	usr.com_user.onoff	Control Command	---	
	drv.state.all	Error Status	---	
I/O	stage.main	Main Stage	---	
	stage.sub	Sub Stage	---	
	usr.com_user_1.onoff	Previous Control Command	---	

### 9.6.1.3. Process

Monitor the control command given from the application and the current statuses and executes the status transferring.

Each status is separated into the more detailed sub statuses. The transfer of sub stages is executed within the processing function for each stage instead of within the stage transfer processing function.



**Figure 9.1 Motor Control Status Transition Diagram**

## 9.6.2. Motor Control Common Processing Function (C\_Common)

### 9.6.2.1. Syntax

```
void C_Common(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.6.2.2. Variable

**Table 9.6 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	usr.com_user.encoder	Encoder Control Command	---	
	usr.com_user.modul	Modulation Method Control Command	---	two-phase or three-phase Modulation
Output	drv.command.encoder	Encoder Drive Command	---	
	drv.command.modul	Modulation Method Drive Command	---	two-phase or three-phase Modulation

### 9.6.2.3. Process

A common process to various motor control statuses is executed.

9.6.10 Shift PWM Control (C\_ShiftPWM\_Control) is executed.

## 9.6.3. Stop Stage Function (C\_Stage\_Stop)

### 9.6.3.1. Syntax

```
void C_Stage_Stop(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.6.3.2. Variable

**Table 9.7 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
I/O	stage.sub	Sub Stage	---	
Output	stage.itr	Interruption Stage	---	
	drv.theta_com	Electrical Angle Reference Value	Q0	
	drv.theta	Rotor Position	Q0	
	drv.omega_com	Driving Speed Reference Value	Q31	
	drv.omega	Speed	Q31	
	drv.Id_com	d-Axis Current Reference Value	Q31	
	drv.Iq_com	q-Axis Current Reference Value	Q31	

### 9.6.3.3. Process

Stops the motor. (Stops the PWM output)

## 9.6.4. Bootstrap Stage Function (C\_Stage\_Bootstrap)

### 9.6.4.1. Syntax

```
void C_Stage_Bootstrap(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.6.4.2. Variable

**Table 9.8 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
	para.time.bootstp	Bootstrap Time	Q0	* MainLoopPrd(s)
I/O	stage.sub	Sub Stage	---	
	stage_counter	Stage Counter	Q0	* MainLoopPrd(s)
Output	stage.itr	Interruption Stage	---	

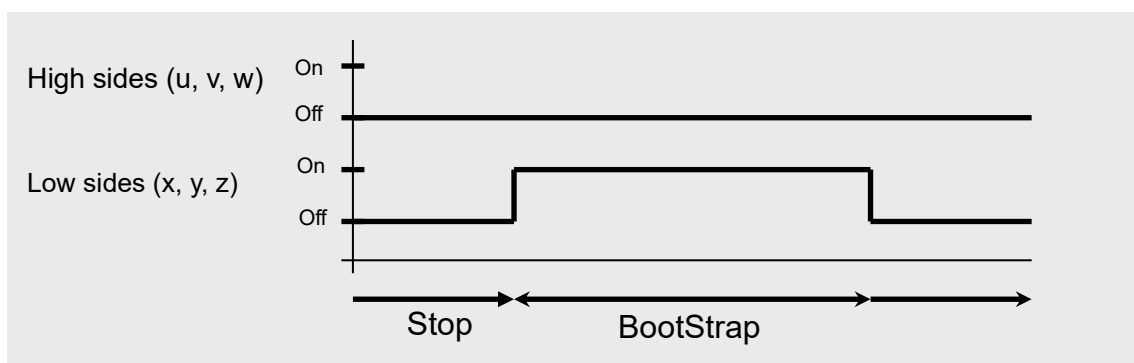
### 9.6.4.3. Process

Outputs the waveform of upper phase All OFF and Lower phase All ON and charges the bootstrap capacitor.

Continues this process during the Bootstrap Time and determines the charging amount for the capacitor based on the Bootstrap Time

Controls the bootstrap status by separating it into the following sub-stages.

1. Initial Status  
Initial setup of bootstrap status.
2. Time Lapse Standby Stage  
Wait for the lapse of specified bootstrap time and transfers to the positioning stage.



**Figure 9.2 Bootstrap**

## 9.6.5. Positioning Stage Function (C\_Stage\_Initposition)

### 9.6.5.1. Syntax

```
void C_Stage_Initposition(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.6.5.2. Variable

**Table 9.9 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
	boot_type	Starting Type	---	
	usr.Id_st_user	Starting Id-Current	Q15	
	usr.lambda_user	Initial Rotor Position	Q0	
	para.vd_pos	Positioning Reference Voltage	Q31	
	para.time.initpos	Positioning Time	Q0	* MainLoopPrd(s)
	para.time.initpos2	Positioning Status Standby Time	Q0	* MainLoopPrd(s)
I/O	stage.sub	Sub Stage	---	
	stage_counter	Stage Counter	Q0	* MainLoopPrd(s)
	drv.Vd_out	Output Voltage	Q31	Enabled only during voltage drive
	drv.Id_com	d-Axis Current Reference Value	Q31	
Output	stage.itr	Interruption Stage	---	
	drv.vector_cmd	Vector Control Command	---	
	drv.Iq_com	q-Axis Current Reference Value	Q31	
	drv.omega_com	Driving Speed Reference Value	Q31	
	drv.theta_com	Electrical Angle Reference Value	Q0	

### 9.6.5.3. Process

Fixes the rotor to the initial position.

While fixing  $\theta$  to the Initial Position,  $\omega$  to 0 and  $I_q$  to 0, gradually increases  $I_d$  from 0.

Continues this process during the Positioning Time and finally  $I_d$  reaches the Start Id Current. Determines the increment of  $I_d$  per unit time based on the Positioning Time and the Start Id Current

After  $I_d$  reaches the Start Id Current and after the lapse of Positioning Standby Time, the stage will be transferred to the next.

The positioning stages will be separated into the following sub stages for controlling.

1. Initial Status  
Initial setup of positioning stages.
2. Id Increasing Stage  
Id will be gradually increased to the setup value.

## 9.6.6. Forced Commutation Stage Function (C\_Stage\_Force)

### 9.6.6.1. Syntax

```
void C_Stage_Force(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:  
No

## 9.6.6.2. Variable

**Table 9.10 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
	boot_type	Starting Type	---	Current or Voltage
	usr.Id_st_user	Starting Id-Current	Q15	
	usr.omega_user	Controller Target Speed	Q31	
	para.omega_v2i	Current Control Changeover Speed	Q15	Enabled only during voltage drive
	para.spd_coef	Output Voltage Factor	Q15	Enabled only during voltage drive
	para.vd_pos	Positioning Output Voltage	Q31	Enabled only during voltage drive
	para.omega_min	Forced Commutation End Speed	Q15	
	para.sp_ud_lim_f	Drive Speed Increase/Decrease Limit	Q31	
I/O	stage.sub	Sub Stage	---	
	drv.omega_com	Driving Speed Reference Value	Q31	
Output	drv.vector_cmd	Vector Control Command	---	
	stage.itr	Interruption Stage	---	
	drv.lq_com	q-Axis Current Reference Value	Q31	
	drv.Id_com	d-Axis Current Reference Value	Q31	
	drv.Vd_out	Output Voltage	Q31	Enabled only during voltage drive

## 9.6.6.3. Process

Starts the rotation of rotor. In this stage, a rotating magnetic field is forcibly applied instead of the feedback processing with vector control, and the rotor will follow it and be rotated.

The drive speed reference,  $\omega_{com}$ , will be gradually increased while fixing the Id to Start Id Current and Iq to 0.

Acquire  $\theta$  from  $\omega_{com}$  ( $\theta = \omega_{com} \times t$ ). This process will be continued until  $\omega$  reaches the Forced Commutation End Speed

The Drive Target Speed will be increased with a constant increment for approaching Control Target Speed

The Drive Target Speed reaches  $\omega$ .

## 9.6.7. Forced Steady Changeover Stage Function (C\_Stage\_Change\_up)

### 9.6.7.1. Syntax

```
void C_Stage_Change_up(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.6.7.2. Variable

**Table 9.11 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
	usr.Id_st_user	Starting Id-Current	Q15	
	usr.lq_st_user	Starting Iq-Current	Q15	
	usr.omega_user	Controller Target Speed	Q31	
	para.delta_lambda	Current changeover phase when changing up	Q0	
	para.sp_ud_lim_f	Drive Speed Increase/Decrease Limit	Q31	
	para.time.go_up	Standby Time after Change Up	Q0	* MainLoopPrd(s)
I/O	stage.sub	Sub Stage	---	
	stage_counter	Stage Counter	Q0	* MainLoopPrd(s)
	drv.omega_com	Driving Speed Reference Value	Q31	
Output	stage.itr	Interruption Stage	---	
	drv.vector_cmd	Vector Control Command	---	
	drv.Id_com	d-Axis Current Reference Value	Q31	
	drv.lq_com	q-Axis Current Reference Value	Q31	

### 9.6.7.3. Process

Decreases Id to 0 and increases Iq to Start Iq Current and controls the orientation of magnetic field to directly downwards with respect to the rotor.

Generates the torque component.

Acquire  $\omega$  and  $\theta$  through the position estimation computation.

The Drive Target Speed will be increased with a constant increment for approaching Control Target Speed. However, the Drive Target Speed will not be used for controlling because the speed control is not conducted in this stage.

The forced steady changeover stage will be controlled by separating it into the following sub-stages.

1. Initial Status  
Initial setup of forced steady changeover stages.
2. Id and Iq Changeover Stages  
Gradually decreases Id to 0 and gradually increases Iq to the specified value simultaneously. The increasing/decreasing curves are not linear but follows the triangular function curve. After the completion of changeover, the stage will be transferred to the time lapse standby.
3. Time Lapse Standby Stage  
Waits for the lapse of the specified forced steady changeover time and transfers to the steady stage.

## 9.6.8. Steady Stage Function (C\_Stage\_Steady\_A)

### 9.6.8.1. Syntax

```
void C_Stage_Steady_A(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.6.8.2. Variable

**Table 9.12 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
	usr.omega_user	Controller Target Speed	Q31	
	para.sp_up_lim_s	Drive Speed Increase Limit	Q31	
	para.sp_dn_lim_s	Drive Speed Decrease Limit	Q31	
I/O	stage.sub	Sub Stage	---	
	drv.omega_com	Driving Speed Reference Value	Q31	
Output	drv.vector_cmd	Vector Control Command	Q0	
	stage.itr	Interruption Stage	---	
	drv.Id_com	d-Axis Current Reference Value	Q31	

### 9.6.8.3. Process

Executes the process of steady stage.

The Drive Target Speed is increased with a constant increment and approaches the Control Target Speed

## 9.6.9. Protection Stage Function (C\_Stage\_Emergency)

### 9.6.9.1. Syntax

```
void C_Stage_Emergency(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.6.9.2. Variable

**Table 9.13 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
I/O	stage.sub	Sub Stage	---	
Output	drv.vector_cmd	Vector Control Command	---	
	stage.itr	Interruption Stage	---	

### 9.6.9.3. Process

When an overcurrent occurs, the stage will be transferred to this.

When a hardware overcurrent is detected, the motor drive output u, v, w, x, y, and z are all Hi-z.

When a software overcurrent is detected, the motor drive output u, v, w, x, y, and z are all OFF. The rotor will be rotated by the inertia. This stage will be maintained until the overcurrent status restoration process is executed.

## 9.6.10. Shift PWM Control (C\_ShiftPWM\_Control)

### 9.6.10.1. Syntax

```
void C_ShiftPWM_Control(vector_t* const _motor)
Parameter:
    vector_t* const _motor: Motor Control Structure
Returned Value:
    No
```

### 9.6.10.2. Variable

**Table 9.14 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	shunt_type	Shunt Type	—	
	stage.itr	Interruption Stage	—	
	usr.com_user.spwm	Shift PWM Control Method	—	
	drv.omega_com	Driving Speed Reference Value	Q31	
	para.spwm_threshold	Shift PWM Changeover Speed	Q15	
	para.minpls	Minimum Pulse Width	Q0	
Output	drv.minpls	Minimum Pulse Width	Q0	
	drv.command.spwm	Shift PWM Drive Command	—	

### 9.6.10.3. Process

This control is enabled only during one-shunt. Setup of minimum duty value for using the shift PWM ON/OFF and the previous current detection value. Determines the shift PWM drive method based on the shift PWM control method, interruption stage and the target speed. The sample software is set up according to the conditions in the table.

**Table 9.15 Shift PWM Drive Method Conditions**

Shift PWM Control Method usr.com_user.spwm	Interruption Stage stage.itr	Target Speed drv.omega_com	Shift PWM Drive Method drv.command.spwm	
OFF (0)	All	All	OFF (0)	
ON (1)	Stop	All	OFF (0)	
	Emergency	All	OFF (0)	
	Initposition_i	All	OFF (0)	
	Force_i Change_up Steady_A	Target Speed ≥ Shift Changeover Speed drv.omega_com ≥ para.spwm_threshold		OFF (0)
		Target Speed < Shift Changeover Speed drv.omega_com < para.spwm_threshold		ON (1)

## 9.7. Motor Drive Function

### 9.7.1. Explanation of Terms

#### 9.7.1.1. 3-phase Modulation

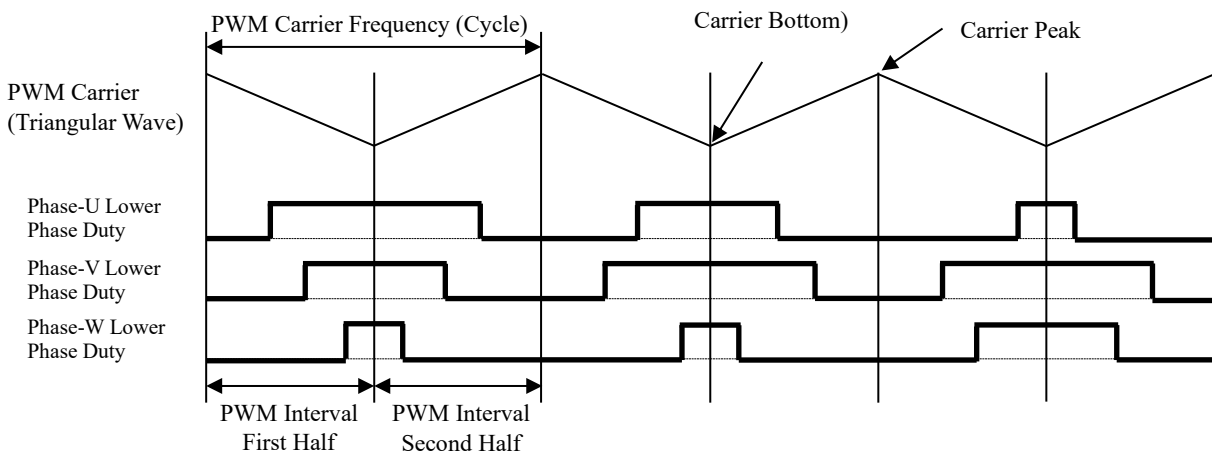


Figure 9.3 3-phase Modulation

#### 9.7.1.2. 2-phase Modulation

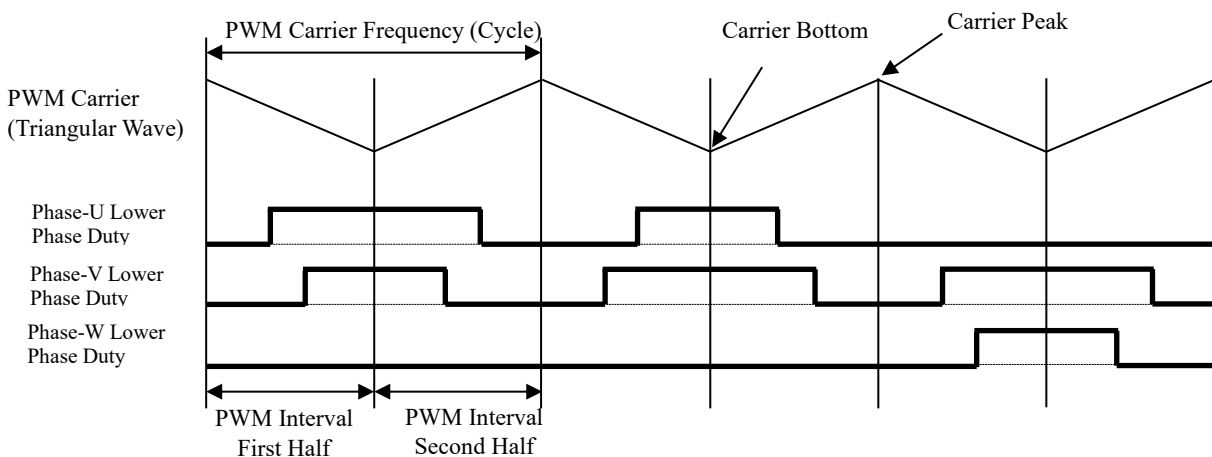


Figure 9.4 2-phase Modulation

## 9.7.2. Motor Current, Power Supply Voltage Gain (VE\_GetdataFromVEreg)

### 9.7.2.1. Syntax

```
void VE_GetdataFromVEreg(const ipdrv_t* const _ipdrv, vector_t* const _motor)
```

Parameter:

ipdrv\_t\* const \_ipdrv: Select an IP table address.

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

## 9.7.2.2. Variable

**Table 9.16 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	drv.Sector1	Previous Sector	Q0	0 to 5
	drv.TrqMd	Current Detection Mode	—	Three-shunt Single shunt first half Single shunt second half
I/O	drv.lao_ave	Phase-U Zero Current Average	Q15	3-shunt only
	drv.lbo_ave	Phase-V Zero Current Average	Q15	3-shunt only
	drv.lco_ave	Phase-W Zero Current Average	Q15	3-shunt only
	drv.ldco_ave	Zero Current Average	Q15	1-shunt only
Output	drv.la	Phase-U Current	Q15	
	drv.lb	Phase-V Current	Q15	
	drv.lc	Phase-W Current	Q15	
	drv.Vdc	Power supply voltage	Q15	
	drv.la_adc	Phase-U Current Conversion Result	Q15	3-shunt only
	drv.lb_adc	Phase-V Current Conversion Result	Q15	3-shunt only
	drv.lc_adc	Phase-W Current Conversion Result	Q15	3-shunt only
	drv.ldc1_adc	Phase-1st Current Conversion Result	Q15	1-shunt only
	drv.ldc2_adc	Phase-2nd Current Conversion Result	Q15	1-shunt only
	drv.Vdc_adc	Power Supply Voltage Conversion Result	Q15	

### 9.7.2.3. Process

Acquires the AD conversion result and computes the power supply voltage and the motor current.

1. Checks the completion of AD conversion with the PMD trigger and when the conversion is complete, acquires the conversion results according to the table below.  
When the conversion is not complete, continue standing by until the completion of conversion (Never occurs under normal conditions, but as a fail safe process, the standby process is executed).
2. Computes the power supply voltage and the motor current based on the acquired AD conversion results.

- Power supply voltage

Power Supply Voltage Conversion Result drv.Vdc_adc	AD Conversion Result 3 ADxREG3
---	-----------------------------------

As the power supply voltage being unsigned, the AD conversion result is right shifted by 1 bit and converted to the Q15 format variable.

$$\text{drv.Vdc} = \text{drv.Vdc\_adc} \gg 1$$

- Motor Current

<Current Detection Mode: Three-Shunt>

**Table 9.17 Current Conversion Result**

	Sector					
	0	1	2	3	4	5
Phase-U Current Conversion Result drv.la_adc	AD Conversion Result 2 ADxREG2	AD Conversion Result 1 ADxREG1	AD Conversion Result 1 ADxREG1	AD Conversion Result 0 ADxREG0	AD Conversion Result 0 ADxREG0	AD Conversion Result 2 ADxREG2
Phase-V Current Conversion Result drv.lb_adc	AD Conversion Result 0 ADxREG0	AD Conversion Result 2 ADxREG2	AD Conversion Result 2 ADxREG2	AD Conversion Result 1 ADxREG1	AD Conversion Result 1 ADxREG1	AD Conversion Result 0 ADxREG0
Phase-W Current Conversion Result drv.lc_adc	AD Conversion Result 1 ADxREG1	AD Conversion Result 0 ADxREG0	AD Conversion Result 0 ADxREG0	AD Conversion Result 2 ADxREG2	AD Conversion Result 2 ADxREG2	AD Conversion Result 1 ADxREG1

The AD conversion result when the motor is stopped is regarded as the zero current and stored in the following variables.

Phase-U Zero Current Conversion Result drv.lao_ave	Phase-U Current Conversion Result drv.la_adc
Phase-V Zero Current Conversion Result drv.lbo_ave	Phase-V Current Conversion Result drv.lb_adc
Phase-W Zero Current Conversion Result drv.lco_ave	Phase-W Current Conversion Result drv.lc_adc

The values of three-phase current will be computed in accordance with the following table based on the AD conversion result with zero current (motor stop) and the phase current conversion result.

**Table 9.18 Current Calculation**

	Sector					
	0	1	2	3	4	5
Phase-U Current drv.la	-lb-ic	.lao_ave - .la_adc	.lao_ave - .la_adc	.lao_ave - .la_adc	.lao_ave - .la_adc	-lb-ic
Phase-V Current drv.lb	.lbo_ave - .lb_adc	-la-ic	-la-ic	.lbo_ave - .lb_adc	.lbo_ave - .lb_adc	.lbo_ave - .lb_adc
Phase-W Current drv.lc	.lco_ave - .lc_adc	.lco_ave - .lc_adc	.lco_ave - .lc_adc	-la-lb	-la-lb	.lco_ave - .lc_adc

<Current Detection Mode: Single Shunt First Half and the Second Half>

Current Detection Mode: During the first half of single shunt, the AD conversion result is acquired at the position of the first half of PWM cycle.

Current Detection Mode: During the second half of single shunt, the AD conversion result is acquired at the position of the second half of PWM cycle.

**Table 9.19 Conversion Result**

Current Acquisition Position Conversion Result	PWM Cycle Second Half	PWM Cycle First Half
Current Conversion Result 1 drv.ldc1_adc	AD Conversion Result 0 ADxREG0	AD Conversion Result 1 ADxREG1
Current Conversion Result 2 drv.ldc2_adc	AD Conversion Result 1 ADxREG1	AD Conversion Result 0 ADxREG0

The AD conversion result when the motor is stopped is regarded as the zero current and stored in the following variables.

Zero Current Conversion Result drv.ldco_ave	AD Conversion Result 0 ADxREG0
--	-----------------------------------

The values for the three-phase current are computed in accordance with the table below based on the AD conversion result, Zero Current Conversion Result, during the zero current (motor stop) and the AD conversion results at the two timings.

**Table 9.20 Current Calculation**

	Sector					
	0	1	2	3	4	5
Phase-U Current drv.la	-(ldco_adc- ldc2_adc)	-lb-ic	ldco_adc- ldc1_adc	ldco_adc- ldc1_adc	-lb-ic	-(ldco_adc- ldc2_adc)
: Phase-V Current drv.lb	-la-ic	-(ldco_adc- ldc2_adc)	-(ldco_adc- ldc2_adc)	-la-ic	ldco_adc- ldc1_adc	ldco_adc- ldc1_adc
: Phase-W Current drv.lc	ldco_adc- ldc1_adc	ldco_adc- ldc1_adc	-la-lb	-(ldco_adc- ldc2_adc)	-(ldco_adc- ldc2_adc)	-la-lb

## 9.7.3. Clarke Conversion (E\_Clarke)

### 9.7.3.1. Syntax

```
void E_Clarke(q15_t_iu, q15_t_iv, q15_t_iw, q15_t*_ialpha, q15_t*_ibeta)
```

Parameter:

q15\_t\_iu : Phase-U Current  
 q15\_t\_iv : Phase-V Current  
 q15\_t\_iw : Phase-W Current  
 q15\_t\*\_ialpha :  $\alpha$ -Axis Current Storage Address  
 q15\_t\*\_ibeta :  $\beta$ -Axis Current Storage Address

Returned Value:

No

### 9.7.3.2. Process

The three-phase currents ( $I_u$ ,  $I_v$ ,  $I_w$ ) are converted to two-phase ( $I_\alpha$ ,  $I_\beta$ ).

$I_\alpha$  and  $I_\beta$  are computed using the following computation formulas:

$$I_\alpha = \frac{2}{3} \times (I_u \times \cos 0 + I_v \times \cos 120 + I_w \times \cos 240) = \frac{2}{3} \times \left( I_u - \frac{1}{2} I_v - \frac{1}{2} I_w \right)$$

$$I_\beta = \frac{2}{3} \times (I_u \times \sin 0 + I_v \times \sin 120 + I_w \times \sin 240) = \frac{2}{3} \times \left( \frac{\sqrt{3}}{2} I_v - \frac{\sqrt{3}}{2} I_w \right)$$

\* Factors  $\frac{2}{3}$  are used for making the amplitude equivalent, which is  $\frac{3}{2}$  when the three-phase current is converted to those of two-phase  $\alpha\beta$  axes.

Because the total sum of three-phase currents will be 0,  $I_\alpha = I_u$   $I_\beta = (I_u + 2 \times I_v)/\sqrt{3}$  is derived with  $I_u + I_v + I_w = 0$ .

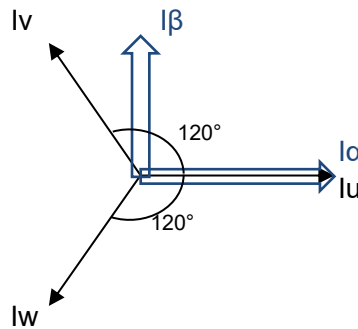


Figure 9.5 Clarke Conversion

## 9.7.4. Park Conversion (E\_Park)

### 9.7.4.1. Syntax

```
void E_Park(q15_t_ialpha, q15_t_ibeta, uint16_t_theta, q15_t*_id, q15_t*_iq)
```

Parameter:

q15\_t\_ialpha :  $\alpha$ -Axis Current  $I_\alpha$   
 q15\_t\_ibeta :  $\beta$ -Axis Current  $I_\beta$   
 q15\_t\_theta : Rotor Position  $\theta$ :  $0 \leq \text{Position} < 360^\circ$  (0 to 0xFFFF)  
 q15\_t\*\_id : d-Axis Current  $I_d$  Storage Address  
 q15\_t\*\_iq : q-Axis Current  $I_q$  Storage Address

Returned Value:

No

### 9.7.4.2. Process

Conversion from the static coordinate of  $\alpha\beta$ -axes to the  $dq$ -axes of rotational coordinate.

$I_d$  and  $I_q$  are computed using the following computation formulas:

$$I_d = I_\alpha \times \cos\theta + I_\beta \times \sin\theta$$

$$I_q = I_\alpha \times (-\sin\theta) + I_\beta \times \cos\theta$$

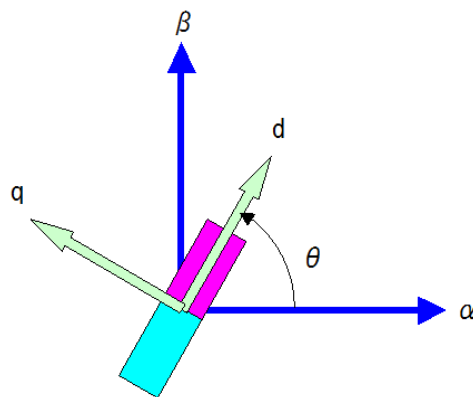


Figure 9.6 Park Conversion

## 9.7.5. Position Estimation Function (D\_Detect\_Rotor\_Position)

### 9.7.5.1. Syntax

```
void D_Detect_Rotor_Position(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.7.5.2. Variable

**Table 9.21 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	drv.command.encoder	Encoder Drive Command	---	
	drv.vector_cmd.F_vcomm_Edetect	Induced Voltage Control Command	---	
	drv.vector_cmd.F_vcomm_omega	Estimated Speed Control Command	---	
	drv.vector_cmd.F_vcomm_theta	Estimated Rotor Position Control Command	---	
	drv.Id	d-Axis Current	Q15	
	drv.Iq	q-Axis Current	Q15	
	drv.omega_com	Driving Speed Reference Value	Q31	
	drv.theta_com	Electrical Angle Reference Value	Q0	
	drv.Vd	d-Axis Voltage	Q31	
	para.motor.Lq	Motor q-Axis Inductance	Q12	
	para.motor.r	Motor Winding Resistance	Q12	
	para.pos.ctrlprd	Position Estimation Control Cycle	Q0	
	para.pos.ki	Position Estimation Integration Gain	Q15	
para.pos.kp	Position Estimation Proportional Gain	Q15		
I/O	drv.Ed	d-Axis Induced Voltage	Q15	
	drv.Ed_I	d-Axis Induced Voltage Integrated Value	Q31	
	drv.Ed_PI	d-Axis Induced Voltage PI Value	Q31	
	drv.omega	Estimated Speed	Q31	
Output	drv.theta	Rotor Position	Q0	

### 9.7.5.3. Process

The PI control is executed by regarding the estimate speed  $\omega_{est}$  of motor drive signal as the amount of operation and the d-axis induced voltage  $E_d$  as the amount of control.

For information, the target value for  $E_d$  is always 0. Accordingly, the deviation is  $-E_d$ .

The rotor position  $\theta$  (angle) is acquired by integrating the estimated speed  $\omega_{est}$  acquired through the PI control.

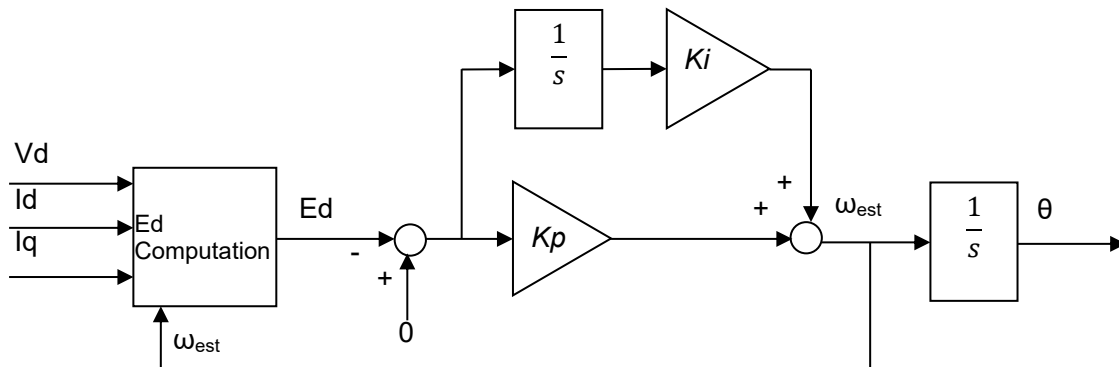
The equivalent circuit equation with regard to the d-axis of motor is expressed as below:

$$V_d = R \cdot I_d + L_d \cdot pI_d - \omega_{est} \cdot L_q \cdot I_q + E_d$$

( $p = d/dt$ ,  $I_d \approx \text{constant}$ , then regarded as  $pI_d = 0$ )

$V_d$  : Motor Applied Voltage  
 $I_d, I_q$  : Motor Current  
 $\omega_{est}$  : Estimated Angular Speed  
 $R$  : Resistance  
 $L_d, L_q$  : Inductance

Consequently, the induced voltage  $E_d$  of the d-axis is acquired using the following computation formula:  
 $E_d = V_d - R \cdot I_d + \omega_{est} \cdot L_q \cdot I_q$



**Figure 9.7** Position Estimation Block Diagram based on Induced Voltage  $E_d$  of the d-Axis

## 9.7.6. Encoder Control Function (H\_Encoder)

### 9.7.6.1. Syntax

```
void H_Encoder(vector_t* const _motor, TSB_EN_TypeDef* const ENx)
```

Parameter:

vector\_t\* const \_motor : Motor Control Structure

TSB\_EN\_TypeDef\* const ENx : ENC Address

Returned Value:

No

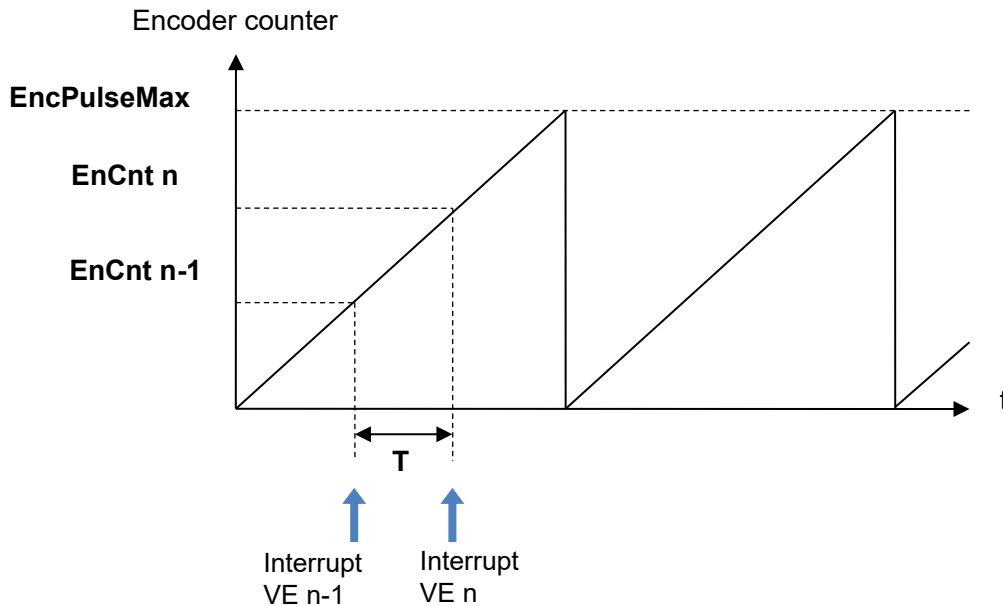
### 9.7.6.2. Variable

**Table 9.22 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	drv.command.encoder	Encoder Drive Command	—	
	drv.omega_com	Driving Speed Reference Value	Q31	
	drv.theta_com	Electrical Angle Reference Value	Q0	
	drv.vector_cmd.F_vcomm_omega	Estimated Speed Control Command	—	
	drv.vector_cmd.F_vcomm_theta	Estimated Rotor Position Control Command	—	
	para.enc.ctrlprd	Encoder Control Cycle	Q16	
	para.enc.deg_adjust	Electrical Angle Adjustment	Q0	
	para.enc.pls2omega	Computation factor from the number of pulses to the speed	Q15	
	para.enc.pls2theta	Computation factor from the number of pulses to the angle	Q32	
	para.enc.plsnum	Number of encoder pulses	Q0	
I/O	drv.EnCnt	Encoder Counter Value	Q0	
	drv.EnCnt_dev	Encoder Counter Deviation	Q0	
	drv.EnCnt1	Encoder Counter Previous Value	Q0	
	drv.omega_enc	Encoder Speed	Q15	
	drv.omega_enc_ave	Encoder Average Speed	Q31	
	drv.omega_enc_raw	Encoder Speed	Q15	
	drv.theta_enc	Encoder Angle	Q0	
Output	drv.omega	Speed	Q31	
	drv.theta	Rotor Position	Q0	

### 9.7.6.3. Process

Reads the number of incremental encoder pulses and processes the calculation of rotor position (angle)  $\theta$  and speed  $\omega$ .



**Figure 9.8 Computation of Position and Speed using Encoder Pulse**

Reads the number of pulses at every VE interruption and computes the following:

Computes  $\theta$  using the formula below based on the current encoder pulse counter reading.

$$\theta = \text{EnCnt}_n \times (360^\circ / \text{EncPulseMax}) \times (\text{Pole} / 2)$$

Computes the speed  $\omega$  using the formula below based on the difference of the encoder pulse counter reading of previous (n-1) and current (n).

$$\omega = \{ (\text{EnCnt}_n - \text{EnCnt}_{n-1}) \times (360^\circ / (\text{EncPulseMax} / (\text{Pole} / 2))) \} / T$$

EnCntn	Current Encoder Counter Reading
EnCntn-1	Previous Encoder Counter Reading
EncPulseMax	Number of Encoder Pulses [ppr] x Multiplier (4)
Pole	Number of Poles
$\theta$	Angle [deg.]
$\omega$	Speed [Hz]
T	Speed Computing Cycle [s]

## 9.7.7. Speed Control Function (D\_Control\_Speed)

### 9.7.7.1. Syntax

```
void D_Control_Speed(vector_t* const _motor)
```

Parameter:

vector\_t\* const \_motor: Motor Control Structure

Returned Value:

No

### 9.7.7.2. Variable

**Table 9.23 List of Variables**

Direction	Code	Description	Q Format	Remarks
Input	drv.vector_cmd.F_vcomm_current	Current Reference Control Command	---	
	drv.id_com	d-Axis Current Reference Value	Q31	
	drv.iq_com	d-Axis Current Reference Value	Q31	
	drv.omega	Estimated Speed	Q31	
	drv.omega_com	Driving Speed Reference Value	Q31	
	para.id_lim	d-Axis Current Limit	Q31	
	para.iq_lim	q-Axis Current Limit	Q31	
	para.spd.ki	Speed Control Integration Gain	Q15	
	para.spd.kp	Speed Control Proportional Gain	Q15	
I/O	drv.iq_ref_i	q-Axis Current Integrated Value	Q31	
	drv.omega_dev	Speed Deviation	Q15	
Output	drv.id_ref	d-Axis Current Standard Value	Q15	
	drv.iq_ref	q-Axis Current Standard Value	Q15	

### 9.7.7.3. Process

Executes the PI control by regarding the output frequency  $\omega$  as the controlled amount and the q-axis current  $I_q$  as the operated amount.

Determines the d-axis and q-axis current standard values based on the deviation of the speed reference and the measured speed.

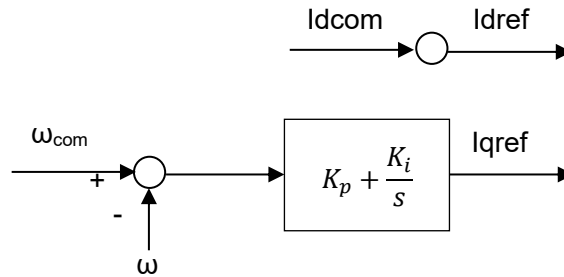


Figure 9.9 Speed Control Block Diagram

### 9.7.8. Inverse Park Conversion (E\_InvPark)

#### 9.7.8.1. Syntax

```
void E_InvPark(q31_t_d, q31_t_q, uint16_t_theta, q31_t*_alpha, q31_t*_beta)
```

Parameter:

q31\_t\_d : d-Axis  
 q31\_t\_q : q-Axis  
 q31\_t\_theta : Rotor Position  $\theta$ :  $0 \leq \text{Position} < 360^\circ$  (0 to 0xFFFF)  
 q31\_t\_alpha :  $\alpha$ -Axis Storage Address  
 q31\_t\_beta :  $\beta$ -Axis Storage Address

Returned Value:

No

#### 9.7.8.2. Process

Converts from the dq-axes of rotational coordinate to the static coordinate of the  $\alpha\beta$ -axes.

$V_\alpha$  and  $V_\beta$  are computed using the following computation formulas:

$$V_\alpha = V_d \times \cos\theta - V_q \times \sin\theta$$

$$V_\beta = V_d \times \sin\theta + V_q \times \cos\theta$$

## 9.7.9. Sector Computation

### 9.7.9.1. Syntax

```
uint8_t D_CalSector(q31_t_valpha, q31_t_vbeta)
```

Parameter:

q31\_t\_valpha :  $\alpha$ -Axis Voltage

q31\_t\_vbeta :  $\beta$ -Axis Voltage

Returned Value:

Sector

### 9.7.9.2. Process

Computes the sector based on the  $\alpha\beta$ -axis voltage.

Stores the previous sector values and determines the current sector values using the conditional formula below:

**Table 9.24 Sector Computation**

Condition		Sector Value
Condition 1	Condition 2	
$(V\alpha \geq 0) \& (V\beta \geq 0)$	$V\alpha \geq (V\beta/\sqrt{3})$	0
	$V\alpha < (V\beta/\sqrt{3})$	1
$(V\alpha < 0) \& (V\beta \geq 0)$	$ V\alpha  < (V\beta/\sqrt{3})$	1
	$ V\alpha  \geq (V\beta/\sqrt{3})$	2
$(V\alpha < 0) \& (V\beta < 0)$	$ V\alpha  \geq ( V\beta /\sqrt{3})$	3
	$ V\alpha  < ( V\beta /\sqrt{3})$	4
$(V\alpha \geq 0) \& (V\beta < 0)$	$V\alpha < ( V\beta /\sqrt{3})$	4
	$V\alpha \geq ( V\beta /\sqrt{3})$	5

## 9.7.10. Spatial Vector Modulation (D\_SVM)

### 9.7.10.1. Structure

```
void D_SVM(q31_t_valpha, q31_t_vbeta, q15_t_vdc, uint8_t_sector,
           uint8_t_modul, uint16_t*_p_vu, uint16_t*_p_vv, uint16_t*_p_vw)
```

Parameter:

q31\_t\_valpha :  $\alpha$ -Axis  $V\alpha$

q31\_t\_vbeta :  $\beta$ -Axis  $V\beta$

q15\_t\_vdc : Power supply voltage

uint8\_t\_sector : Sector

uint8\_t\_modul : Modulation

uint16\_t\*\_p\_vu : Phase-U Duty Ratio  $D_u$  Storage Address

uint16\_t\*\_p\_vv : Phase-U Duty Ratio  $D_v$  Storage Address

uint16\_t\*\_p\_vw : Phase-U Duty Ratio  $D_w$  Storage Address

Returned Value:

No

### 9.7.10.2. Process

Acquires the duty ratio of the three-phase PWM from the two-phase  $\alpha\beta$ -axis voltages.

Acquires the PWM duty ratio of each phase using the spatial vector modulation.

## What is the Spatial Vector Modulation?

As the switching elements in the inverter drive circuit are in the statuses where, the elements of the upper phase are ON or OFF and those of the lower phase are in the inverse status (the dead time is ignored); There are eight patterns as shown in Table 9.25.

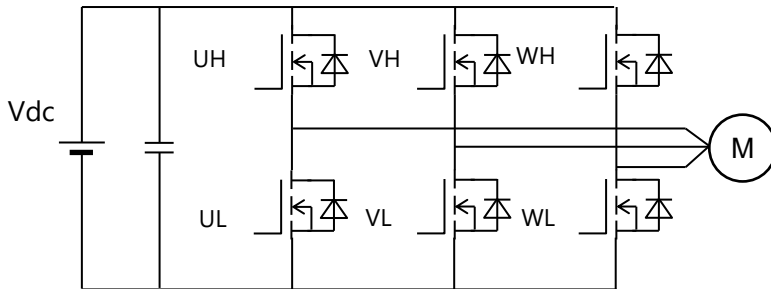
Where these eight patterns are defined as V0 thru V7, the vectors V0 and V7 are located at the origin due to no line voltage, and the remaining six vectors (V1 thru V6) are expressed at every 60 degrees as shown in Figure 9.10.

By combining the neighboring two vectors together, an arbitrary output voltage vector V is acquired.

**Table 9.25 Inverter Switching Status**

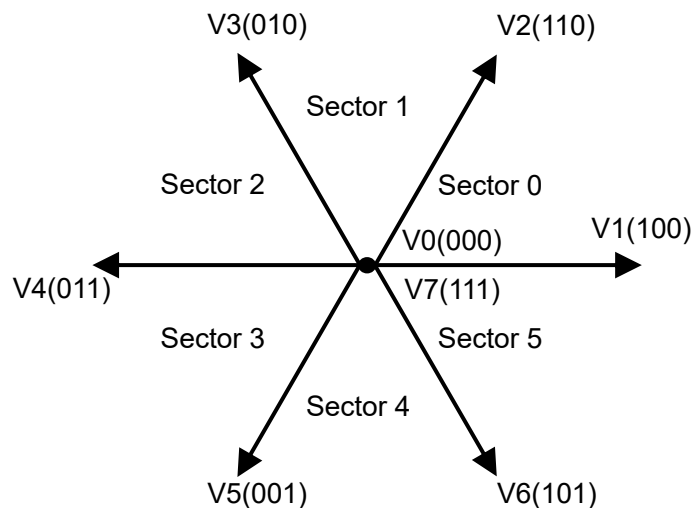
U	V	W	Vector
0	0	0	V0 (0 0 0)
1	0	0	V1 (1 0 0)
1	1	0	V2 (1 1 0)
0	1	0	V3 (0 1 0)
0	1	1	V4 (0 1 1)
0	0	1	V5 (0 0 1)
1	0	1	V6 (1 0 1)
1	1	1	V7 (1 1 1)

0: Upper Phase OFF, Lower Phase ON  
1: Upper Phase ON, Lower Phase OFF



UH: Phase-U Upper Phase Switching Element  
UL: Phase-U Lower Phase Switching Element  
VH: Phase-V Upper Phase Switching Element  
VL: Phase-V Lower Phase Switching Element  
WH: Phase-W Upper Phase Switching Element  
WL: Phase-W Lower Phase Switching Element  
Vdc: Power Supply Voltage

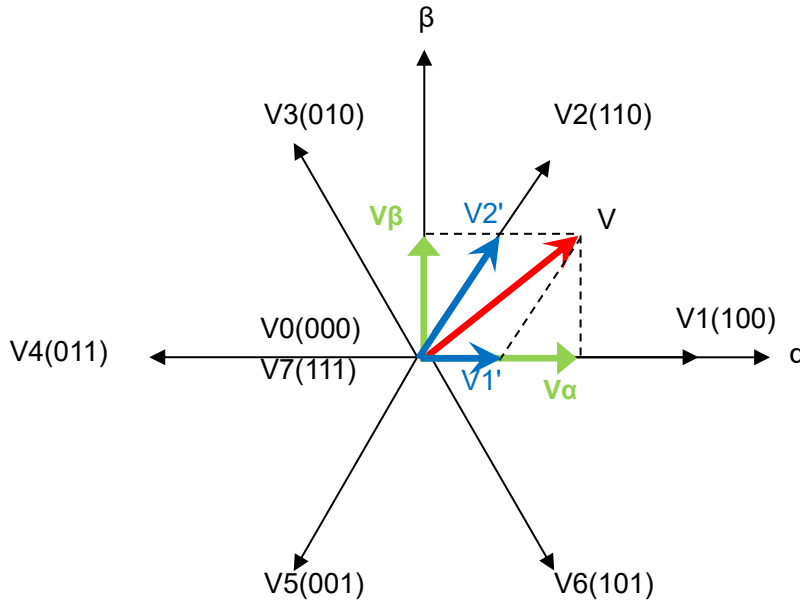
**Figure 9.10 Inverter Drive Circuit**



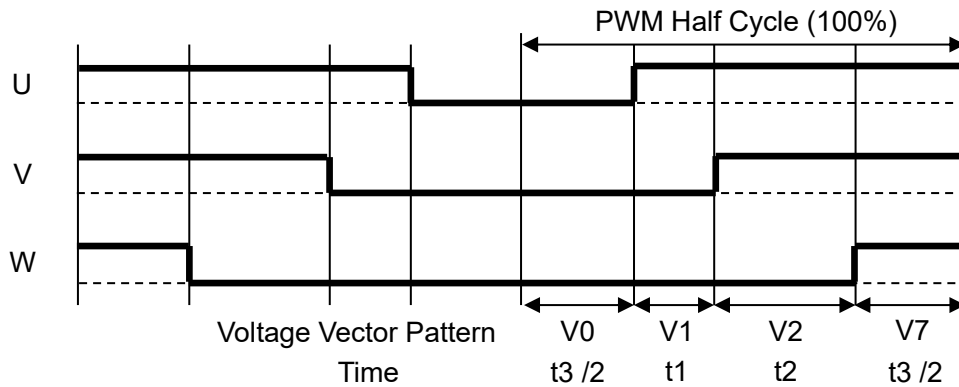
**Figure 9.11 Spatial Vector**

- When the output voltage vector  $V$  is at Sector 0

For example, in Figure 9.11, the composite vector  $V$  of  $V_\alpha$  and  $V_\beta$  is located at the sector 0, accordingly it is acquired as a composite vector of  $V1'$  and  $V2'$  acquired by multiplying the voltage vectors  $V1$  and  $V2$  with factors  $t1$  and  $t2$ , respectively. If they are expressed in the PWM waveform,  $V$  is composed by generating  $V1$  and  $V2$  for the duration of  $t1$  and  $t2$ , respectively, in the half cycle of PWM as shown in Figure 9.12.



**Figure 9.12 Voltage Vector at Sector 0**



**Figure 9.13 three-phase Modulation PWM Waveform**

From Figure 9.12,  $V\alpha$  and  $V\beta$  are expressed as below:

$$V\alpha = V1' \times \cos 0^\circ + V2' \times \cos 60^\circ = V1' + \frac{V2'}{2}$$

$$V\beta = V1' \times \sin 0^\circ + V2' \times \sin 60^\circ = \frac{\sqrt{3}}{2} \times V2'$$

Consequently,  $V1'$  and  $V2'$  are as below:

$$V2' = \frac{2}{\sqrt{3}} V\beta$$

$$V1' = V\alpha - \frac{V2'}{2} = V\alpha - \frac{1}{\sqrt{3}} V\beta$$

Where the motor power supply voltage is  $V_{dc}$ ,

$$V1' = t1 \times V_{dc}$$

$$V2' = t2 \times V_{dc}$$

Consequently, the ratio of  $t1$ ,  $t2$ , and  $t3$  are as below:

$$t1 = k \times \frac{V\alpha - \frac{1}{\sqrt{3}} V\beta}{V_{dc}}$$

$$t2 = k \times \frac{\frac{2}{\sqrt{3}} V\beta}{V_{dc}}$$

$$t3 = 100\% - t1 - t2$$

where,  $k$  is a conversion factor  $3/2$  for making the size to a certain level when converting from three-phase to two-phase.

The definition of occurrence duration for vectors  $V0$  and  $V7$  as  $t3/2$  respectively is the three-phase modulation, whereas the definitions of occurrence duration for vector  $V0$  as  $t3$  and that for  $V7$  as  $0$  are the two-phase modulation.

Consequently, the duties for the three phases are acquired using the following calculations from  $t1$ ,  $t2$  and  $t3$ .

$$\text{Phase-U Duty} = t1 + t2 + (t3/2)$$

$$\text{Phase-V Duty} = t2 + (t3/2)$$

$$\text{Phase-W Duty} = t3/2$$

Similarly, for every sector of the output voltage, the OFF Duty  $D0$ , the Duty  $D1$  when only one phase is ON and the Duty  $D2$  when two phases are ON are computed using the formula for the Duty On duration per the sectors in Table 9.26.

**Table 9.26 Formulas for Duty On Duration per Sector**

Sector	Duty when only one of the phases is ON. D1	Duty when two phases are ON D2	OFF Duty D0
0	$k \times \frac{V\alpha - \frac{1}{\sqrt{3}}V\beta}{Vdc}$	$k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$	100% – D1 – D2
1	$k \times \frac{-V\alpha + \frac{1}{\sqrt{3}}V\beta}{Vdc}$	$k \times \frac{V\alpha + \frac{1}{\sqrt{3}}V\beta}{Vdc}$	
2	$k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$	$k \times \frac{-V\alpha - \frac{1}{\sqrt{3}}V\beta}{Vdc}$	
3	$-k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$	$k \times \frac{-V\alpha + \frac{1}{\sqrt{3}}V\beta}{Vdc}$	
4	$k \times \frac{-V\alpha - \frac{1}{\sqrt{3}}V\beta}{Vdc}$	$k \times \frac{V\alpha - \frac{1}{\sqrt{3}}V\beta}{Vdc}$	
5	$k \times \frac{V\alpha + \frac{1}{\sqrt{3}}V\beta}{Vdc}$	$-k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$	

where, k is a factor 3/2 for making the amplitudes before and after conversion to a certain level. The duties for U, V and W phases are computed from D0, D1 and D2 using the formulas in Table 9.27.

**Table 9.27 Relationship of Sectors and Duties of Phases**

Sector	three-phase Modulation			two-phase Modulation		
	Phase-U Duty	Phase-V Duty	Phase-W Duty	Phase-U Duty	Phase-V Duty	Phase-W Duty
0	D1+D2+(D0/2)	D2+(D0/2)	D0/2	D1+D2	D2	0
1	D2+(D0/2)	D1+D2+(D0/2)	D0/2	D2	D1+D2	0
2	D0/2	D1+D2+(D0/2)	D2+(D0/2)	0	D1+D2	D2
3	D0/2	D2+(D0/2)	D1+D2+(D0/2)	0	D2	D1+D2
4	D2+(D0/2)	D0/2	D1+D2+(D0/2)	D2	0	D1+D2
5	D1+D2+(D0/2)	D0/2	D2+(D0/2)	D1+D2	0	D2

## 10. Explanation of Definitions for Constants

### 10.1. Motor Driver Setting Parameter (D-Para.h)

#### 10.1.1. Motor Control Channel Selection

\_\_CONTROL\_MOTOR\_CH0 : Define when controlling the CH0  
 \_\_CONTROL\_MOTOR\_CH1 : Define when controlling the CH1  
 \_\_CONTROL\_MOTOR\_CH2 : Define when controlling the CH2

#### 10.1.2. Selecting a DAC Output

\_\_USE\_DAC : Define when executing the DAC control for outputting the variables for evaluation.

#### 10.1.3. Selection of Unit for Command Speed

\_\_TGTSPD\_UINT: Selection of unit for command speed  
 0: Hz of Electrical angle  
 1: RPS of mechanical angle  
 2: RPM of mechanical angle

#### 10.1.4. Parameter List

**Table 10.1 Parameter List**

Parameter Name	Description
cMAINLOOP_PRD	Main Cycle Setup
	Unit [s] Resolution 4 kHz
	Set up the main cycle time.
cIXO_AVE	Zero Current Filtering Factor
	--
	Set up the filtering factor. The setup of larger value will stabilize but will delay the response.
cVDQ_AVE	Filtering Factors for Power Supply Voltage Vdc and Output Voltage Vdq
	--
	Set up the filtering factor. The setup of larger value will stabilize but will delay the response.
cOMEGAENC_AVE	Filtering Factor for Speed Detection Value using the Number of Encoder Pulses
	--
	Set up the filtering factor for speed detection. Use a larger value when the number of encoder pulses is small. However, the larger value will bring about the delay of detection. Set up a value that gives the tolerable delay.

## 10.2. Motor Driver Setting Parameter Motor Channel (D\_Para\_chx.h) x = 0, 1, 2

Various motors are driven by changing the parameters in the motor driver.

### 10.2.1. Selection of Control

#### 10.2.1.1. Selection of AMP

`__USE_INAMP` Define when the built-in AMP is used.

#### 10.2.1.2. Choice of Sensor-less/Sensor

`__USE_ENCODER` Define when the encoder control is used.

### 10.2.2. Parameter List per Motor Channel

**Table 10.2 Parameter List per Motor Channel**

Parameter Name	Description
cPOLH	Upper Phase Drive Logical Setup
	0: Low active/ 1: High active
	Change the values according to the board design. Set up the logic for the upper phase driver.
cPOLL	Logical Setup for Lower Phase Driver
	0: Low active/ 1: High active
	Change the values according to the board design. Setup the logic for the lower phase driver.
cV_MAX	Setup of Max Voltage
	Unit [V]
	Change the values according to the board design. Setup the value of power supply voltage variation [A] x 2 <sup>12</sup> with regard to the variation of 1LSB in the AD conversion.
cA_MAX	Setup of Max Current
	Unit [A]
	Change the values according to the board design. Setup the value of phase current variation [A] x 2 <sup>11</sup> with regard to the variation of 1 LSB in the AD conversion.
cSHUNT_TYPE	Setup of Current Acquisition Method (3-shunt)
	3:3 shunt
	Setting of other than three-shunts will cause an error.
cBOOT_TYPE	Setup of Drive Type when Starting
	cBoot_i: Current Type Drive/ cBoot_v: Voltage Type Drive
	Set up the drive type for starting.
cSHUNT_ZERO_OFFSET	Setup of Offset Voltage
	Unit [V]
	Set up the shunt voltage when no current flows. This value will be used as the initial value of average zero current.

cADCH_CURRENT_U	Setup of Phase-U Current Acquisition AD Channel
	AINx
	Set up the AD channel for detecting the Phase-U current
cADCH_CURRENT_V	Setup of Phase-V Current Acquisition AD Channel
	AINx
	Set up the AD channel for detecting the Phase-V current
cADCH_CURRENT_W	Setup of Phase-W Current Acquisition AD Channel
	AINx
	Set up the AD channel for detecting the Phase-W current
cADCH_CURRENT_IDC	Setup of Current Acquisition AD Channel (single shunt)
	AINx
	Set up the AD channel for detecting the current.
cADCH_VDC	Setup of Power Supply Voltage Acquisition AD Channel
	AINx
	Set up the AD channel for detecting the power supply voltage Vdc.
cOVC	Setup of Overcurrent Detecting Value
	Unit [A]
	Set up the overcurrent value. When a coil current exceeding this setup value is detected, the output will be turned OFF by the software.
cVDC_MINLIM	Setup of Min Power Supply Voltage Vdc
	Unit [V]
	Set up the Min value of power supply voltage Vdc. When voltage lower than this value is detected, the motor will be stopped.
cVDC_MAXLIM	Setup of Max Power Supply Voltage Vdc
	Unit [V]
	Set up the Max of power supply voltage Vdc. When voltage higher than this value is detected, the motor will be stopped.
cPWMPRD	Setup of PWM Cycle
	Unit [us] Resolution 16.67ns@120MHz
	Set up the cycle for the PWM carrier.
cDEADTIME	Setup of Dead Time
	Unit [us] Resolution 66.67ns@120MHz
	Set up the dead time.
cREPTIME	Setup of the number of skipped software control.
	Unit [times]: 1 to 15
	The timing for executing the vector computation software processing may be skipped.

	Setting to "1" will execute the software processing of vector computation once every PWM cycle. Setup of "2" will execute the software processing of vector computation one time every two PWM cycles.
cID_ST_USER_ACT	Setup of d-Axis Start Current
	Unit [A]
	Set up the d-axis start current. This current value will be used for positioning and forced commutation. Set up the value of approximately 10% of the rated commutation. When the motor does not operate, gradually increase the value until it operates.
cIQ_ST_USER_ACT	Setup of q-Axis Start Current
	Unit [A]
	Set up the q-axis start current. Set up a value of approximately 1/2 of d-axis start current. If the motor abruptly accelerates when transferring from the forced commutation (d-axis control) to the steady (q-axis control), set a smaller value and, if the motor stops, set a larger value.
cMOTOR_R	Motor Coil Resistance
	Unit [Ohm]
	Set up a resistance equivalent to that of the single phase of motor coil.
cMOTOR_LQ	q-Axis Inductance
	Unit [mH]
	Set up the inductance of q-axis of motor coil
cMOTOR_LD	d-Axis Inductance (not used)
	Unit [mH]
	Set up the inductance of d-axis of motor coil
cPOLE	Setup of the Number of Motor Poles
	Unit [poles]
	Set up the number of motor poles.
cID_KP	d-Axis Current Control Proportional Gain
	Unit [V/A]
	Set up the d-axis current control proportional gain.
cID_KI	d-Axis Current Control Integration Gain
	Unit [V/As]
	Set up the d-axis current control integration gain.
cIQ_KP	q-Axis Current Control Proportional Gain
	Unit [V/A]
	Set up the q-axis current control proportional gain.
cIQ_KI	q-Axis Current Control Integration Gain
	Unit [V/As]

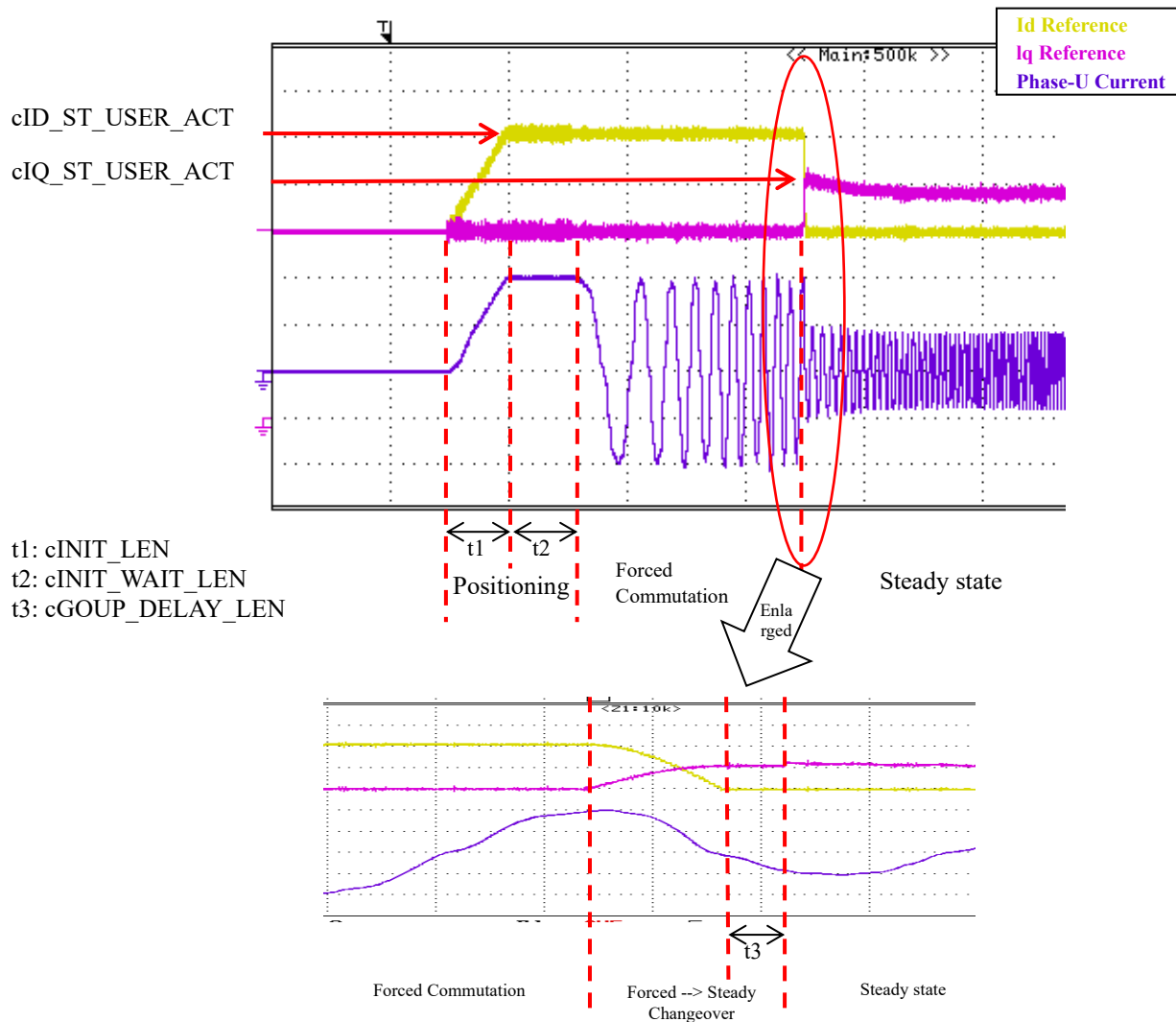
	Set up the q-axis current control integration gain.
cPOSITION_KP	Position Estimation Proportional Gain
	Unit [Hz/V]
	Set up the position estimation proportional gain.
cPOSITION_KI	Position Estimation Integration Gain
	Unit [Hz/Vs]
	Set up the position estimation integration gain.
cSPEED_KP	Speed Control Proportional Gain
	Unit [A/Hz]
	Set up the speed control proportional gain.
cSPEED_KI	Speed Control Integration Gain
	Unit [A/Hz]
	Set up the speed control integration gain.
cSPD_PI_PRD	Setup of Speed PI Control Cycle
	[Time]
	Set up the speed PI control cycle. Setting to "1" will execute the speed PI computation once every PWM cycle. Setting to "2" will execute the speed PI computation once every two PWM cycles.
cFCD_UD_LIM	Drive Speed Acceleration Rate (Forced Commutation)
	Unit [Hz/s]
	Setup the speed acceleration rate during the forced commutation. Decrease the value when the motor speed does not follow the output of forced commutation.
cSTD_UP_LIM	Drive Speed Acceleration Rate (Steady)
	Unit [Hz/s]
	Set up the speed acceleration rate during the steady state.
cSTD_DW_LIM	Drive Speed Deceleration Rate (Steady)
	Unit [Hz/s]
	Set up the deceleration rate during the steady state.
cBOOT_LEN	Bootstrap Waveform Output Duration
	Unit [s] Resolution: 1 ms
	Set up the output duration for the bootstrap waveform.
cINIT_LEN	Positioning Time
	Unit [s] Resolution: 1 ms
	Set up the positioning duration.
cINIT_WAIT_LEN	Standby Time after Positioning

	Unit [s] Resolution: 1 ms
	Set up the standby time after positioning.
cGOUP_DELAY_LEN	Standby Time after Change Up
	Unit [s] Resolution: 1 ms
	Set up the standby time after change up.
cHZ_MAX	Max Frequency
	Unit [Hz]
	Set up the max frequency to be detected by the microcontroller. Set up a value increased by 10% to 20% than the max frequency used for controlling. A smaller value will enhance the precision of computation, but if the detected value exceeds this figure, the control will collapse.
cHZ_MIN	Changeover speed from forced commutation to steady.
	Unit [Hz]
	Set up a speed for transferring from the forced commutation to steady. Set up a speed that permits the position estimation (the induced voltage will be detected).
cID_LIM	d-Axis Current Limit
	Unit [A]
	Set up the d-axis current limit.
cIQ_LIM	q-Axis Current Limit
	Unit [A]
	Set up the q-axis current limit.
cINITIAL_POSITION	Initial Position
	Unit [deg]
	Set up the angle for positioning in the electrical angle.
cVD_POS	Positioning output voltage during the voltage drive
	Unit [V]
	Set up the voltage for positioning. Enabled only when the cBOOT_TYPE is "cBoot_v" See 10.2.3.2 Constants for Voltage Type Starting for details.
cSPD_COEF	Forced Commutation Output Voltage Factor during Voltage Drive
	Set a value x: $0 < x < 1$
	Determine the output voltage during the forced commutation. The multiplication of this setup and the reference speed is defined as the output voltage. $V_d = \text{cSPD\_COEF} \times \text{Omega\_com}$ Enabled only when the cBOOT_TYPE is "cBoot_v" See 10.2.3.2 Constants for Voltage Type Starting for details.
cHZ_V2I	Changeover Speed from Voltage Control to Current Control
	Unit [Hz]
	Set up a speed for changeover from the voltage control to the current control.

	Setting to a value larger than the cHZ_MIN will execute the transfer to a steady value equal to cHZ_MIN and over and will change over to the current control. Enabled only when the cBOOT_TYPE is "cBoot_v"
cENC_PULSE_NUM	Setup of the Number of Encoder Pulses
	Unit [ppr]
	Set up the number of pulses for the encoder. Small value for the number of encoder pulses will degrade the speed detection accuracy. If the detected speed is not stable, increase the filtering factor cOMEGAENC_AVE.
cENC_DEG_ADJUST	Encoder Position Adjustment
	Unit [deg.]
	Set up the shifted positional angle of phase-Z with regard to the electrical angle 0°
cENC_NOISE_TIME	Setup of Encoder Input Signal Noise Cancellation Time (Only A-ENC is enabled)
	Unit [μs] Resolution fc (8.33ns@120MHz)
	Set up the noise cancellation time of the encoder input signal. The signals shorter than this setup time will be canceled. This setup will be enabled only for the microcontroller equipped with A-ENC.
__FIXED_VDC	Setup of Power Supply Voltage Fixing
	0: Detected Value, 1: Fixed Value
	When setting the power supply voltage to a fixed value, set up to "1"
cVDC	Fixed Power Supply Voltage
	Unit [V]
	Set up the power supply voltage. Enabled only when the __FIXED_VDC is "1"

## 10.2.3. Relationship of Constant Setups and Waveform

### 10.2.3.1. Constants for Current Type Starting

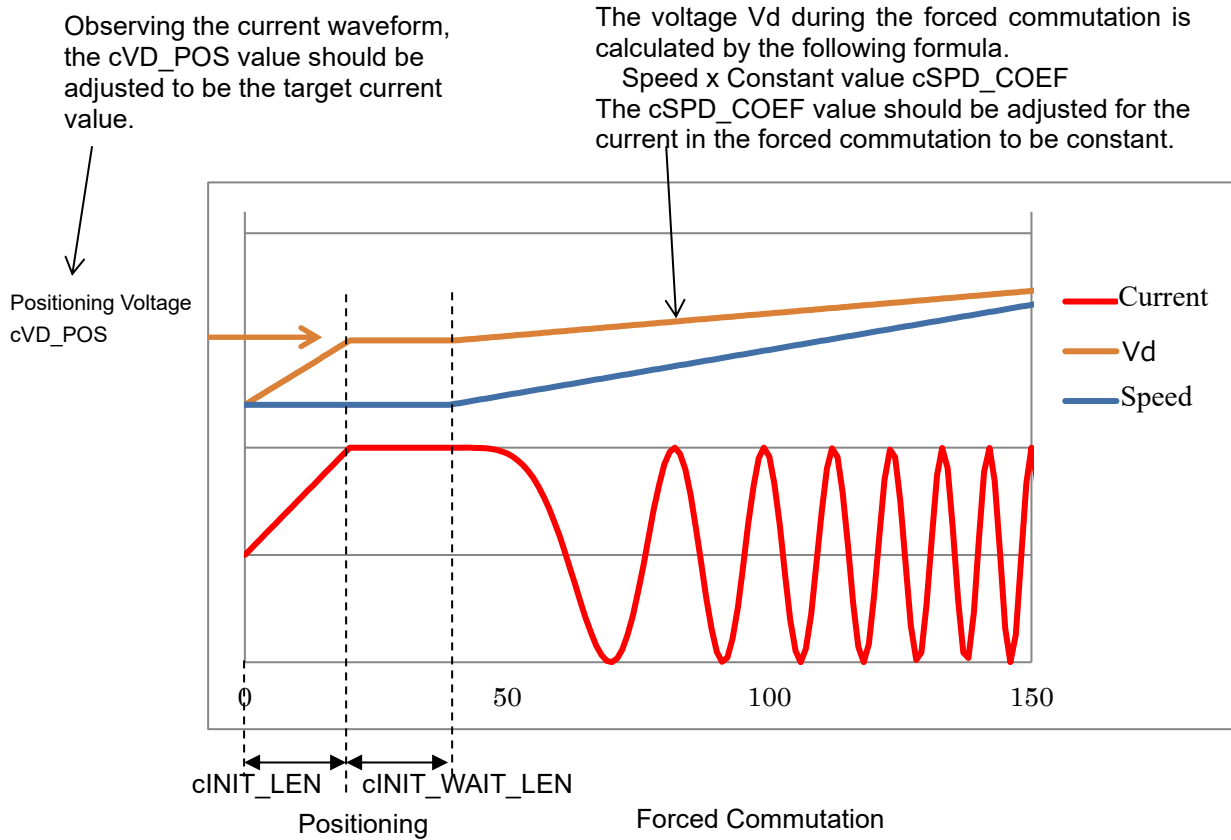


**Figure 10.1 Starting Current Waveform (Positioned to electrical angle 0°)**

Switch the values `cID_ST_USER_ACT` and `cIQ_ST_USER_ACT` during the change up stage. After switching, the control will be conducted with a constant reference value of `Iq` during the time of `cGOUP_DELAY_LEN`. After transferring to steady, the `Iq` reference value will be computed by the PI control.

**10.2.3.2. Constants for Voltage Type Starting**

Determine the constants while checking the current waveform on an oscilloscope.



**Figure 10.2 How to Adjust Parameters during Voltage Drive**

**10.3. Constants for User Control**

```

/*Initial value*/
#define cININT_VALUE    (0)
#define cININT_FF      (0xff)

/* Key setting */
#define P_SW1          TSB_PG_DATA_PG3    /* slideSW1 or 8(KEY1) */
#define P_SW2          TSB_PG_DATA_PG4    /* slideSW2 or 9(KEY2) */
#define P_PSW1         TSB_PD_DATA_PD1    /* pushSW1 DACSW */

#define cKEY_CHATA_CNT    (20)    /* [cnt] chattering counter for KEY SW */
#define cNO_ENTER_KEY    (0)    /* Soft ADC Setting */
#define cADUNIT_USR      TSB_ADC      /* User ad data ADCUnit */
#define cADAUNIT_USR     TSB_ADA      /* User ad data ADAUnit */
#define cADTRG_USR       ADC_TRG_SINGLE /* ADC Trigger Type */
#define cADCH_ADKEY      ADC_AIN0     /* ADC Channel for AD Key */
#define cADREG_ADKEY     ADC_REG4     /* Result register No for AD Key */
#define cADCH_VR         ADC_AIN0     /* ADC Channel for VR */
#define cADREG_VR        ADC_REG4     /* Result register No for VR */
#define cADCH_TEMP       ADC_AIN13    /* ADA Channel for TEMP */
#define cADREG_TEMP      ADC_REG7     /* Result register No for TEMP */
#define cADAVECNT        (10)        /* ADC average count */
#define cPUSHSW_CHATA_CNT (5)        /* [cnt] Chattering counter for Push SW */
    
```

```

/* AD speed control setting */
#define cAD_MIN          (0x10)          /* Voltage value 0.3V */
#define cAD_MAX          (0xF0)          /* Voltage value 4.7V */
#define cSPEED_USER_MIN (12)            /* [Hz] Min Target speed of motor */
#define cSPEED_USER_MAX (200)           /* [Hz] Max Target speed of motor */

```

```

/* AD Temp setting */
#define cONE_ABOVE      (1)

```

```

/* DACMODE setting */
#define cDACMODE_MIN (0)          /* DACMODE count0 */
#define cDacMODE_MAX (3)         /* DACMODE count3 */

```

```

/* LED setting */
#define LED_FORCED   TSB_PV_DATA_PV0      /* LED4 */
#define LED_EMG      TSB_PB_DATA_PB6      /* LED6 */
#define cLED_ON      (1)                  /* LED ON level */
#define cLED_OFF     (0)                  /* LED OFF level */

```

```

/* flag status */
#define cFLG_ON      (1)
#define cFLG_OFF    (0)

```

```

/* UART Setting */
#define UART_BRD_INIT      ((uint32_t)0x00A6002B)
#define UART_CLK_INIT     ((uint32_t)0x00000000)
#define UART_CR0_INIT     ((uint32_t)0x00000001)
#define UART_CR1_INIT     ((uint32_t)0x00000040)
#define UART_SEND_WAIT    (1000)
#define UART_COUNT        (0)
#define c10MHz             (1000000)
#define cLOW_ACTIVE       (0)
#define cSWRST10          (0x2U)
#define cSWRST01          (0x1U)
#define cSWRSTF           (0x80)

```

```

/* User_interface Setting */
#define cROTATION_CW (1)
#define cROTATION_CCW (0)
#define cPSW1_Hi (1)
#define cPSW1_Low (0)
#define cStrCH0 ("CH0")
#define cINV_MIN_TEMP (-20)
#define cINV_MAX_TEMP (100)

#define cSendBufSIZE (80+1)

```

## 11. Timings for Control and Data Update

### 11.1. Vector Control Using VE

#### 11.1.1. 3-Shunt Control

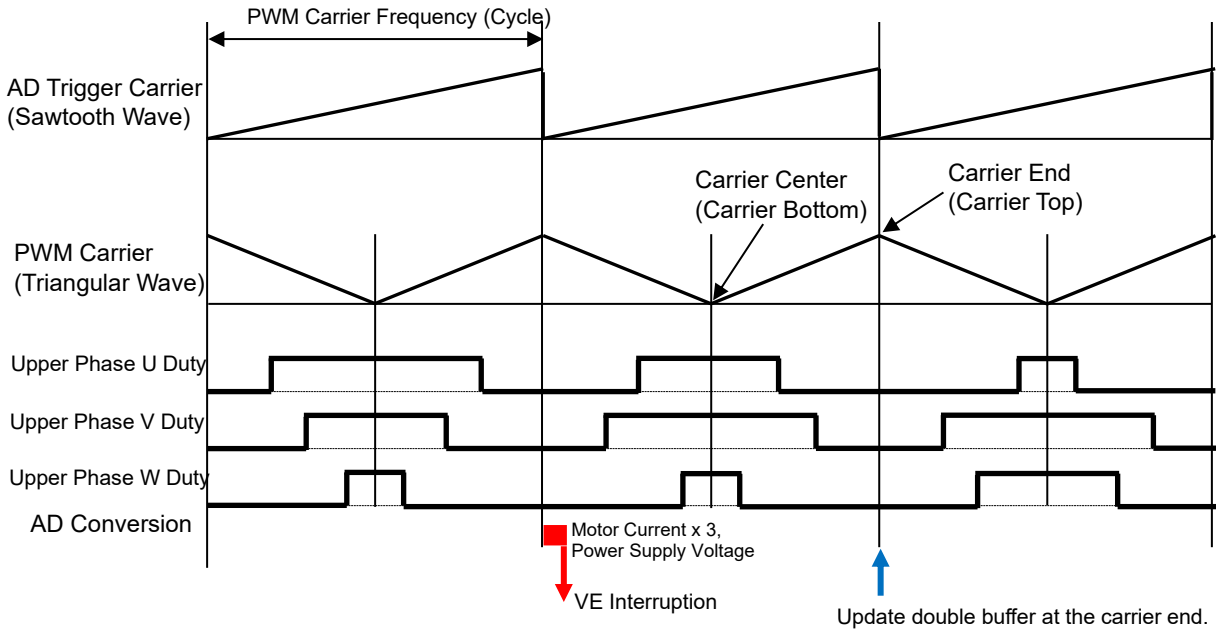


Figure 11.1 3-Shunt Control

#### 11.1.1.1. Carrier Waveform

Table 11.1 Carrier Waveform

Type	Waveform
PWM	Triangular Wave (for all three phases)
AD Trigger	Sawtooth Wave

#### 11.1.1.2. Double Buffer Update Timing

Table 11.2 Double Buffer Update Timing

Data	Update Timing
PWN Cycle (RATE)	Carrier End
Duty Value (CMPU, CMPV, CMPW)	Carrier End
Trigger Position (TRGCMPx)	Carrier End
Output Setup (MDOUT)	Carrier End

## 11.1.1.3. Interruption

**Table 11.3 Interruption**

Interruption Request	Interruption	Timing
VE	Permitted	After completion of VE input schedule
PWM(PMD)	Prohibited	Once every one time, twice and four times.
AD Conversion Complete	Prohibited	After completing the AD conversion for all of motor current x 3 and power supply voltage.
AD Trigger	—	Same frequency as PWM

## 11.1.2. 1-Shunt Control

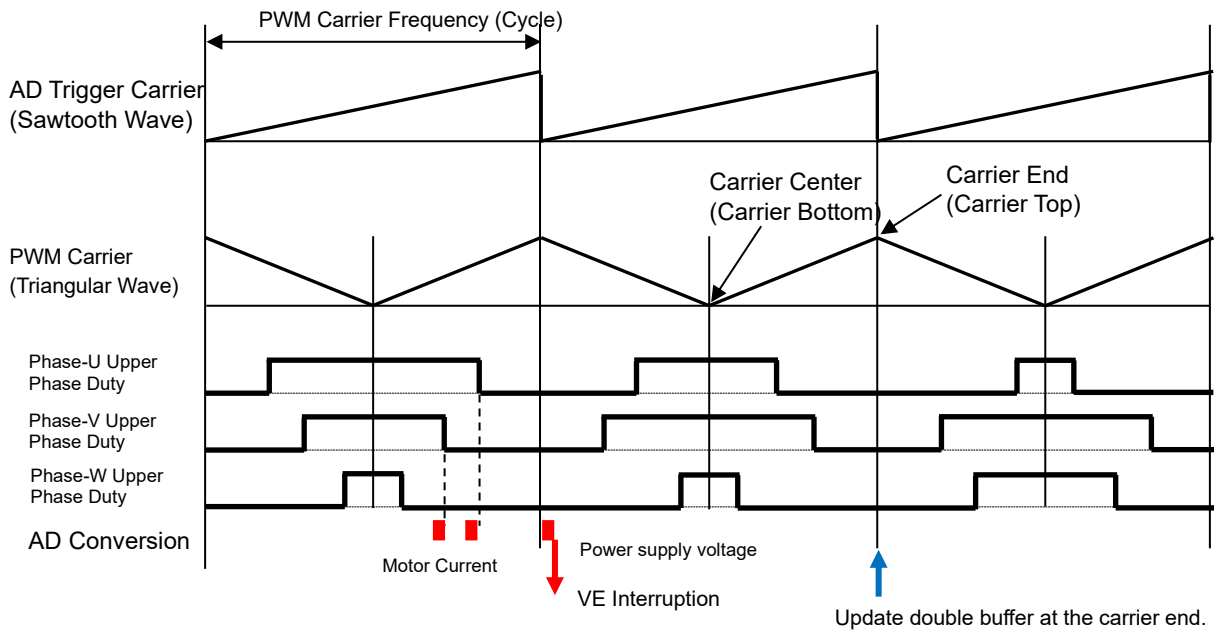


Figure 11.2 1-Shunt Control

### 11.1.2.1. Carrier Waveform

Table 11.4 Carrier Waveform

Type	Waveform
PWM	Triangular Wave (for all three phases)
AD Trigger	Sawtooth Wave

### 11.1.2.2. Double Buffer Update Timing

Table 11.5 Double Buffer Update Timing

Data	Update Timing
PWN Cycle (RATE)	Carrier End
Duty Value (CMPU, CMPV, CMPW)	Carrier End
Trigger Position (TRGCMPx)	Carrier End
Output Setup (MDOU)	Carrier End

### 11.1.2.3. Interruption

Table 11.6 Interruption

Interruption Request	Interruption	Timing
VE	Permitted	After completion of VE input schedule

PWM(PMD)	Prohibited	Once every one time, twice and four times.
AD Conversion Complete	Prohibited	After completion of AD conversion of power supply voltage
AD Trigger	—	Same frequency as PWM

## 12. Peripheral Driver

### 12.1. Vector Engine (A-VE+)

#### 12.1.1. Specifications of Functions

##### 12.1.1.1. IP\_VE\_init

VE Initial Setup

**API:**

```
void IP_VE_init(TSB_VE_TypeDef* const VEx, VE_InitTypeDef* const _initdata)
```

**Parameters:**

**VEx: Selection of VE address**

**\_initdata: VE Initial Setup Data Structure**

See 12.1.2.1 VE\_InitTypeDef for details.

**Function:**

Initial Setup of VE

**Supplement:**

Call up during VE suspension with prohibited interruption.

**Return Parameter:**

No

##### 12.1.1.2. VE\_Start

VE Start

**API:**

```
VE_Start(const ipdrv_t* const _ipdrv)
```

**Parameters:**

**\_ipdrv: Select an IP table address.**

**Function:**

Starts VE.

**Supplement:**

None

**Return Parameter:**

No

##### 12.1.1.3. VE\_GetPhaseCurrent

Acquisition of Phase Current

**API:**

```
void VE_GetPhaseCurrent(const ipdrv_t* const _ipdrv,
                        q15_t* _ia, q15_t* _ib, q15_t* _ic)
```

**Parameters:**

**\_ipdrv: Select an IP table address.**

**\_ia: Setup of an address for the variable to store the phase-a current.**

**\_ib: Setup of an address for the variable to store the phase-b current.**

**\_ic: Setup of an address for the variable to store the phase-c current.**

**Function:**

Acquires the current values for each phase.

The value of phase-U current is stored in phase-a, phase-V current in phase-b and phase-W current in phase-c.

**Supplement:**

None

**Return Parameter:**

No

##### 12.1.1.4. VE\_GetCurrentAdcData

Acquisition of Current AD Value

**API:**

void

```
VE_GetCurrentAdcData(const ipdrv_t* const _ipdrv,
```

uint32\_t\* \_adc\_ia, uint32\_t\* \_adc\_ib, uint32\_t\* \_adc\_ic)

**Parameters:**

**\_ipdrv:** Select an IP table address.

**\_adc\_ia:** Setup of an address for a variable to store the AD value for the phase-a current.

**\_adc\_ib:** Setup of an address for a variable to store the AD value for the phase-b current.

**\_adc\_ic:** Setup of an address for a variable to store the AD value for the phase-c current.

**Function:**

Acquires the values of IAADCx, IBADCx, and ICADCx registers in the current AD values for each phase.

**Supplement:**

None

**Return Parameter:**

No

### 12.1.1.5. VE\_GetdataFromVEreg

Acquisition of VE Register Data

**API:**

void VE\_GetdataFromVEreg(const ipdrv\_t\* const \_ipdrv, vector\_t\* const \_motor)

**Parameters:**

**\_ipdrv:** Select an IP table address.

**\_motor:** Selecting an address for the vector control variable structure

**Function:**

Stores the values; motor voltage Vdc, d-axis voltage Vd, q-axis voltage Vq, d-axis current Id, and q-axis current Iq from the VE register to each variable.

At the time when the current is not detectable due to the duty width, the previously detected values of d-axis current Id and q-axis current Iq will be written in the VE register.

**Supplement:**

None

**Return Parameter:**

No

### 12.1.1.6. VE\_GetPWM\_DutyU

Acquisition of Phase-U Duty Value

**API:**

uint32\_t VE\_GetPWM\_DutyU(const ipdrv\_t\* const \_ipdrv)

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the value of phase-U duty register.

**Supplement:**

None

**Return Parameter:**

Phase-U Duty

### 12.1.1.7. VE\_GetPWM\_DutyV

Acquisition of Phase-V Duty Value

**API:**

uint32\_t VE\_GetPWM\_DutyV(const ipdrv\_t\* const \_ipdrv)

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the value of phase-V duty register.

**Supplement:**

None

**Return Parameter:**

Phase-V Duty

## 12.1.1.8. VE\_GetPWM\_DutyW

Acquisition of Phase-W Duty Value

**API:**

```
uint32_t VE_GetPWM_DutyW(const ipdrv_t* const _ipdrv)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the value of phase-W duty register.

**Supplement:**

None

**Return Parameter:**

Phase-W Duty

## 12.1.1.9. VE\_GetPWM\_DutyMed

Acquisition of Duty Median

**API:**

```
uint32_t VE_GetPWM_DutyMed(const ipdrv_t* const _ipdrv)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the duty medians for phases U, V, and W.

**Supplement:**

None

**Return Parameter:**

Duty Median

## 12.1.1.10. VE\_GetPWM\_Modulation

Acquisition of Modulation Type

**API:**

```
int VE_GetPWM_Modulation(const ipdrv_t* const _ipdrv)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the modulation type.

**Supplement:**

None

**Return Parameter:**

Modulation Type 0: three-phase modulation, 1: two-phase modulation

## 12.1.1.11. VE\_GetSector

Acquisition of Current Sector

**API:**

```
uint32_t VE_GetSector(const ipdrv_t* const _ipdrv)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the current sector value.

**Supplement:**

None

**Return Parameter:**

Current Sector Value [0 - 11]

## 12.1.1.12. VE\_GetShiftPWMState

Acquisition of Shift PWM Status

**API:**

```
int VE_GetShiftPWMState(const ipdrv_t* const _ipdrv)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the shift PWM status.

**Supplement:**

None

**Return Parameter:**

Shift PWM Status 0: Ordinary PWM, 1: Shift PWM

### 12.1.1.13. VE\_GetOutputMode

Acquisition of PWM Output Status

**API:**

```
int VE_GetOutputMode(const ipdrv_t* const _ipdrv)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the PWM output status.

**Supplement:**

None

**Return Parameter:**

PWM Status      OCRMD\_OUT\_OFF: Output OFF  
                   OCRMD\_OUT\_ON: Output ON  
                   OCRMD\_OUT\_ON\_LOWPH: Only Lower Phase ON

### 12.1.1.14. VE\_SetdataToVEreg\_Stop

Data Setting to VE Register (Stop)

**API:**

```
void  
VE_SetdataToVEreg_Stop(const ipdrv_t* const _ipdrv, const vector_t* const _motor)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.  
**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setup of VE register during the stop status.  
 Initializes the current control gain integration register.

**Supplement:**

None

**Return Parameter:**

No

### 12.1.1.15. VE\_SetdataToVEreg\_Bootstrap

Data Setting to VE Register (Bootstrap)

**API:**

```
void  
VE_SetdataToVEreg_Bootstrap(const ipdrv_t* const _ipdrv,  
                             const vector_t* const _motor)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.  
**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setup of VE register during the bootstrap status.  
 Through the VE register, a waveform that is ON only at the L-side is output by setting to the two-phase modulation and setting the output voltage to 0.

**Supplement:**

None

**Return Parameter:**

No

### 12.1.1.16. VE\_SetdataToVEreg\_Initposition\_i

Data Setup to VE Register (nitposition current control type)

**API:**

```
void  
VE_SetdataToVEreg_Initposition_i (const ipdrv_t* const _ipdrv,  
                                   const vector_t* const _motor)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.  
**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setup of the current control type positioning status to the VE register.

**Supplement:**

None

**Return Parameter:**

No

**12.1.1.17. VE\_SetdataToVEreg\_Initposition\_v**

Data Setup to VE Register (Initposition voltage control type)

**API:**

```
void  
VE_SetdataToVEreg_Initposition_v (const ipdrv_t* const _ipdrv,  
                                   const vector_t* const _motor)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.  
**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setup of voltage control type positioning status to the VE register.

**Supplement:**

None

**Return Parameter:**

No

**12.1.1.18. VE\_SetdataToVEreg\_Force\_i**

Data Setup to VE Register (forced commutation, current control type)

**API:**

```
void  
VE_SetdataToVEreg_Force_i (const ipdrv_t* const _ipdrv,  
                            const vector_t* const _motor)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.  
**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setup of current control type forced commutation status to the VE register.

**Supplement:**

None

**Return Parameter:**

No

**12.1.1.19. VE\_SetdataToVEreg\_Force\_v**

Data Setting to VE Register (forced commutation, voltage control type)

**API:**

```
void  
VE_SetdataToVEreg_Force_v(const ipdrv_t* const _ipdrv,  
                           const vector_t* const _motor)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.  
**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setup of voltage control type forced commutation status to the VE register.

**Supplement:**

None

**Return Parameter:**

No

**12.1.1.20. VE\_SetdataToVEreg\_Change\_up**

Data Setup to VE Register (change up)

**API:**

void

VE\_SetdataToVEreg\_Change\_up (const ipdrv\_t\* const \_ipdrv,  
const vector\_t\* const \_motor)

**Parameters:**

**\_ipdrv:** Select an IP table address.

**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setting of change up status to the VE register.

**Supplement:**

None

**Return Parameter:**

No

**12.1.1.21. VE\_SetdataToVEreg\_Steady\_A**

Data Setup to VE Register (Steady)

**API:**

void

VE\_SetdataToVEreg\_Steady\_A (const ipdrv\_t\* const \_ipdrv,  
const vector\_t\* const \_motor)

**Parameters:**

**\_ipdrv:** Select an IP table address.

**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setup of steady status to the VE register.

**Supplement:**

None

**Return Parameter:**

No

**12.1.1.22. VE\_SetdataToVEreg\_Emergency**

Data Setup to VE Register (EMG)

**API:**

void

VE\_SetdataToVEreg\_Emergency (const ipdrv\_t\* const \_ipdrv,  
const vector\_t\* const \_motor)

**Parameters:**

**\_ipdrv:** Select an IP table address.

**\_motor:** Setup of an address for the vector control variable structure.

**Function:**

Processes the setup of emergency status to the VE register.

**Supplement:**

None

**Return Parameter:**

No

**12.1.1.23. VE\_SetZeroCurrentData**

Zero Current Setup

**API:**

void

VE\_SetZeroCurrentData(const ipdrv\_t\* const \_ipdrv,  
uint32\_t \_z\_ia, uint32\_t \_z\_ib, uint32\_t \_z\_ic)

**Parameters:**

- `_ipdrv`: Select an IP table address.
- `_z_ia`: Setup of the phase-a zero current AD value
- `_z_ib`: Setup of the phase-b zero current AD value
- `_z_ic`: Setup of the phase-c zero current AD value

**Function:**

Setup of zero current AD value to the VE register.

**Supplement:**

None

**Return Parameter:**

No

### 12.1.1.24. VE\_SetVDCreg

Setup of Motor Voltage and Vdc

**API:**

```
void VE_SetVDCreg(const ipdrv_t* const _ipdrv, q15_t _dat)
```

**Parameters:**

- `_ipdrv`: Select an IP table address.
- `_dat`: Setup of motor voltage.

**Function:**

Setup of motor voltage to the VE register.

**Supplement:**

None

**Return Parameter:**

No

### 12.1.1.25. VE\_SetSpwmMode

Shift PWM Mode Setting

**API:**

```
void VE_SetSpwmMode(TSB_VE_TypeDef* const VEx, uint8_t _dat);
```

**Parameters:**

- `VEx`: Selection of VE address
- `_dat`: Setup of shift PWM mode

**Function:**

Setup of shift PWM mode status to the VE register.

**Supplement:**

None

**Return Parameter:**

No

### 12.1.1.26. VE\_SetModulType

Setup of Modulation Type

**API:**

```
void VE_SetModulType (const ipdrv_t* const _ipdrv, uint8_t _dat)
```

**Parameters:**

- `_ipdrv`: Select an IP table address.
- `_dat`: Setup of modulation type

**Function:**

Setup of modulation type to VE register.

**Supplement:**

None

**Return Parameter:**

No

## 12.1.2. Data Structure

### 12.1.2.1. VE\_InitTypeDef

**Data Fields:**

uint8\_t ve\_ch: Vector Engine Channel  
0: Channel 0  
1: Channel 1

uint8\_t shunt: Shunt Type  
3:3 shunt  
1:1 shunt

uint16\_t pwmfreq: Carrier Frequency

uint16\_t reptime: Repeat Frequency (1 to 15)

uint16\_t trgmode: Start Trigger  
TRGMODE\_UNITA: Starts with an interruption when the ADCA PMD0 trigger synchronized conversion is complete.  
TRGMODE\_UNITB: Starts with an interruption when the ADCB PMD1 trigger synchronized conversion is complete.

uint16\_t tpwm: PWM Cycle Time (phase interpolation)  
A value for setting to the VETPWM register.

uint16\_t idkp: d-Axis Current Control Proportional Gain

uint16\_t idki: d-Axis Current Control Integration Gain

uint16\_t iqkp: q-Axis Current Control Proportional Gain

uint16\_t iqki: q-Axis Current Control Integration Gain

uint16\_t zerooffset: Zero Current Offset

## 12.2. Motor Control Circuit (PMD)

### 12.2.1. Specifications of Functions

#### 12.2.1.1. IP\_PMD\_init

PMD Initial Setup

**API:**

void  
IP\_PMD\_init(TSB\_PMD\_TypeDef\* const PMDx, PMD\_InitTypeDef\* const \_initdata)

**Parameters:**

**PMDx:** Select a PMD address  
**\_initdata:** PMD Initial Setup Data Structure  
See 12.2.2.1 PMD\_InitTypeDef for details.

**Function:**

Initial setup of PMD.

**Supplement:**

Call up during PMD suspension with prohibited interruption.

**Return Parameter:**

No

#### 12.2.1.2. PMD\_GetEMG\_Status

Acquisition of EMG Protection Status

**API:**

emg\_status\_e PMD\_GetEMG\_Status(const ipdrv\_t\* const \_ipdrv)

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Acquires the EMG protection status.

**Supplement:**

None

**Return Parameter:**

**emg\_status\_e:** EMG Protection Status  
**cNormal:** Normal  
**cEMGProtected:** Protected

### 12.2.1.3. PMD\_ReleaseEMG\_Protection

EMG Protection Status Released

**API:**

```
void PMD_ReleaseEMG_Protection(const ipdrv_t* const _ipdrv)
```

**Parameters:**

**\_ipdrv:** Select an IP table address.

**Function:**

Releases the EMG protection status.

**Supplement:**

Even if this function is called up, the protection status will not be released unless the MDOUT is 0 and the EMG port is H.

**Return Parameter:**

No

### 12.2.2. Data Structure

#### 12.2.2.1. PMD\_InitTypeDef

**Data Fields:**

uint8_t	<b>shunt:</b> Shunt Type 3:3 shunt 1:1 shunt
uint8_t	<b>poll:</b> L-Side Polarity 0:L active 1:H active
uint8_t	<b>polh:</b> H-Side Polarity 0:L active 1:H active
uint16_t	<b>pwmfreq:</b> PWM Frequency A value to be set up to PMDxMDPDR
uint16_t	<b>deadtime:</b> Dead Time A value to be set up to PMDxDTR

## 12.3. Analogue Digital Converter (ADC)

### 12.3.1. Specifications of Functions

#### 12.3.1.1. IP\_ADC\_init

ADC Initial Setup

**API:**

```
void IP_ADC_init(TSB_AD_TypeDef* const ADx, AD_InitTypeDef* const _initdata)
```

**Parameters:**

**ADx:** Select an ADC Address

**\_initdata:** ADC Initial Setup Data Structure

See 12.3.2.1 AD\_InitTypeDef for details.

**Function:**

Initial setup of ADC

**Supplement:**

Call up during ADC suspension with prohibited interruption.

**Return Parameter:**

No

### 12.3.2. Data Structure

#### 12.3.2.1. AD\_InitTypeDef

Data Fields:

```
uint8_t      shunt: Shunt Type
                3:3 shunt
                1:1 shunt

uint8_t      iuch: Phase-U Current Input AD Channel No. (three-shunts)
uint8_t      ivch: Phase-V Current Input AD Channel No. (three-shunts)
uint8_t      iwch: Phase-W Current Input AD Channel No. (three-shunts)
uint8_t      idcch: DC Current Input AD Channel No. (Single Shunt)
uint8_t      vdcch: Motor Voltage Vdc Input AD Channel No.
uint8_t      pmd_ch: Select a PMD Channel
                cPMD: PMD Channel 0
                cPMD: PMD Channel 1
                cPMD: PMD Channel 2

uint8_t      pints: Select Interruption for PM Trigger
                cPINTS_B:ITTADxPDA
                cPINTS_B:ITTADxPDB
```

## 12.4. Explanation of Constants

### 12.4.1. Constants for Microcontroller equipped with A--VE (mcuip\_drv.h)

#### 12.4.1.1. PMD

Table 12.1 PMD

For MDCR Register Setting			
Name of Constants	Description	Selected Data	Description
cWPWMMD	W-phase mode	PWMMD_SWTOOTH	edge-aligned PWM, sawtooth wave
		PWMMD_TRIANGLAR	center-aligned PWM, triangular wave
		PWMMD_SWRROTH_REV	
		PWMMD_TRAIN	
cVPWMMD	V-phase mode	PWMMD_SWTOOTH	edge-aligned PWM, sawtooth wave
		PWMMD_TRIANGLAR	center-aligned PWM, triangular wave

		PWMMD_SWRROTH_REV	
		PWMMD_TRAIN	
cUPWMMD	phase-U mode	PWMMD_SWTOOTH	edge-aligned PWM, sawtooth wave
		PWMMD_TRIANGLAR	center-aligned PWM, triangular wave
		PWMMD_SWRROTH_REV	
		PWMMD_TRAIN	
cDSYNCS	Double buffer update timing for the duty compare register and PWM period register.	DSYNCS_INTCYC	Depends on interrupt cycle setting
		DSYNCS_CENTER	Updates at PWM carrier center
		DSYNCS_END	Updates at PWM carrier end
		DSYNCS_CENTER_END	Updates at both PWM carrier center and end
cDCMEN	Port output mode	DCMEN_DISABLE	Correction disable
		DCMEN_ENABLE	Correction enable
cDTCREN	Dead time correction	DTCREN_DISABLE	Correction disable
		DTCREN_ENABLE	Correction enable
cPWMSYNT	Port output mode	SYNTMD_0	
		SYNTMD_1	
cPWMSPCFY	Duty mode	DTYMD_COMMON	three-phase common mode
		DTYMD_INDEPENDENT	three-phase independent mode
cPWMINT	PWM interrupt timing	PINT_BOTTOM	Interrupt request when PWM counter $PMD1MDCNT < MDCNT[15:0] > = 0x0001$
		PINT_TOP	Interrupt request when PWM counter $PMD1MDCNT < MDCNT[15:0] > = <MDPRD[15:0]>$
cPWMINTPRD	PWM interrupt period	INTPRD_HALF	Interrupt request at every 0.5 PWM period
		INTPRD_1	Interrupt request at every PWM period
		INTPRD_2	Interrupt request at every 2 PWM periods
		INTPRD_4	Interrupt request at every 4 PWM periods

For MDPOt Register Setting			
Name of Constants	Description	Selected Data	Description
cSYNCS	MDOUt transfer timing for trigger synchronous.	SYNCS_ASYNC	Asynchronous
		SYNCS_INTENC	when INTENCx (ENCx interrupt request) occurs
		SYNCS_NTTB	when INTT32Ax0 (TMRBx interrupt request) occurs
		SYNCS_CTRGO	when CTRGO(ENCx MCMP completed) occurs
cPSYNCS	MDOUt transfer timing for PWM synchronous.	PSYNCS_WRITE	Reflect when write
		PSYNCS_CENTER	Reflect when PWM carrier center
		PSYNCS_END	Reflect when PWM carrier end
		PSYNCS_CENTER_END	Reflect when PWM carrier center and end

**For PORTMD Register Setting**

Name of Constants	Description	Selected Data	Description
cPORTMD	Port control settings at tool break	PORTMD_ALLHIZ	Upper phases = High-z / lower phases = High-z
		PORTMD_UHIZ_LON	Upper phases = High-z / lower phases = PMD output
		PORTMD_UON_LHIZ	Upper phases = PMD output / lower phases = High-z
		PORTMD_ALLON	Upper phases = PMD output / lower phases = PMD output

For TRGCR Register Setting			
Name of Constants	Description	Selected Data	Description
cTRGMD_3SHT	Trigger timing setting for 3shunt	TRGMD_DIS	Trigger output disabled
		TRGMD_DOWN	Trigger output at down-count match
		TRGMD_UP	Trigger output at up-count match
		TRGMD_UPDOWNM	Trigger output at up-/down-count match
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak and bottom
cTRGMD_1SHT	Trigger timing setting for 1shunt	TRGMD_DIS	Trigger output disabled
		TRGMD_DOWN	Trigger output at down-count match
		TRGMD_UP	Trigger output at up-count match
		TRGMD_UPDOWNM	Trigger output at up-/down-count match
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak and bottom
cBUFSYNC	Triger Buffer update timing	TRGBE_SYNC	Sync
		TRGBE_ASYNC	Async The value written to PMDTRGx is immediately reflected.)
cCARSEL	Selection of a comparison career	CARSEL_BASE	Compared by the base carrier
		CARSEL_PHASE	Compared by TEMPREG0/1=U,REG2=V,REG3=W-phase carrier

For TRGMD Register Setting			
Name of Constants	Description	Selected Data	Description
cEMGTGE	Output enable in EMG protection state	EMGTGE_DISABLE	Disable trigger output in the protection state
		EMGTGE_ENABLE	Enable trigger output in the protection state

For EMGCR Register Setting			
Name of Constants	Description	Selected Data	Description

cEMGCNT	EMG input detection time	0 to 15	The noise remove time value. cEMGCNT × 16/fsys. (resolution:200[ns] at 80MHz) cEMGCNT ≥ 0 to 15. When cEMGCNT=0, the noise filter is bypassed.
cINHEN	Tool break enable setting	INHEN_DISABLE	Tool break disable
		INHEN_ENABLE	Tool break enable
cEMGMD	EMG protection mode select	EMGMD_ALLHIZ	All phases High-Z
		EMGMD_UON_LHIZ	Lower phases High-Z
		EMGMD_UHIZ_LON	Upper phases High-Z
		EMGMD_ALLHIZ2	All phases High-Z

For OVCR Register Setting			
Name of Constants	Description	Selected Data	Description
cOVVCNT	OVV input detection time	0 to 15	The noise remove time value. cEMGCNT × 16/fsys. (resolution:200[ns] at 80MHz) cEMGCNT ≥ 0 to 15. When cEMGCNT=0, the noise filter is bypassed.
cADIN1EN	ADC B monitor interrupt input enable	ADIN1EN_DISABLE	Disable input
		ADIN1EN_ENABLE	Enable input
cADIN0EN	ADC A monitor interrupt input enable	ADIN0EN_DISABLE	Disable input
		ADIN0EN_ENABLE	Enable input
cOVVMD	OVV protection mode	OVVMD_NOCON	No output control
		OVVMD_UON_LOFF	All upper phases ON, all lower phases OFF
		OVVMD_UOFF_LON	All upper phases OFF, all lower phases ON
		OVVMD_ALLOFF	All phases OFF
cOVVISEL	OVV input select	OVVISEL_PORT	Port input
		OVVISEL_ADC	ADC monitor signal
cOVVRS	OVV protection release	OVVRS_NORMAL	Disable automatic release of OVV protection
		OVVRS_AUTO	Enable automatic release of OVV protection

## 12.4.1.2. VE

cTADC Set up AD conversion time for single shunt shift PWM.

**Table 12.2 VE**

For FMODE Register Setting			
Name of Constants	Description	Selected Data	Description
cSPWMMD	Selects a PWM shift mode	SPWMMD_SPWM1	Shift 1
		SPWMMD_SPWM2U	Shift 2 (phase-U Normal PWM)
		SPWMMD_SPWM2V	Shift 2 (V-phase Normal PWM)
		SPWMMD_SPWM2W	Shift 2 (W-phase Normal PWM)
cCCVMD	Phase-change mode selection	CCVMD_RELATIVE	relative
		CCVMD_ABSOLUTE	absolute
cPHCVDIS	Phase transformation	PHCVDIS_ENABLE	2-3 phase transformation is enabled
		PHCVDIS_DISABLE	2-3 phase transformation is disabled
cVSLIMMD	Controls voltage scalar limitation	VSLIMMD_DIS	Scalar limitation is disabled
		VSLIMMD_D	Limits the voltage on the d-axis
		VSLIMMD_Q	Limits the voltage on the q-axis
		VSLIMMD_DQ	dq proportional limitation
cMREGDIS	Keep the previous value of SIN/COS/SECTOR	MREGDIS_EFFECTIVE	Effective
		MREGDIS_NOEFFECTIVE	No effective
cCRCEN	Trigger correction	CRCEN_DISABLE	Disable trigger correction.
		CRCEN_ENABLE	Enable trigger correction.
cIDQSEL	De-coupling mode selection	IDQSEL_FEEDBACK	Feedback current(ID, IQ)
		IDQSEL_INSTRUCTION	Instruction current(IDREF, IQREF)
cIAPLMD	Direction of current detection	IPLMD_SHUNT	Shunt
		IPLMD_SENSOR	Sensor
cIBPLMD	Direction of current detection	IPLMD_SHUNT	Shunt
		IPLMD_SENSOR	Sensor
cICPLMD	Direction of current detection	IPLMD_SHUNT	Shunt
		IPLMD_SENSOR	Sensor

For MODE Register Setting			
Name of Constants	Description	Selected Data	Description
cIPDEN	Current polarity determination	IPDEN_DISABLE	Current polarity determination is disabled.
		IPDEN_ENABLE	Current polarity determination is enabled.
cPMDTDCEN	Dead time	PMDTDCEN_DISABLE	Dead time correction of the PMD

	correction control of the PMD		is disabled.
		PMDDTCEN_ENABLE	Dead time correction of the PMD is enabled.
cPWMFLEN	Duty of 100% setting when the upper-limit	PWMFLEN_DISABLE	Duty of 100% setting is disabled.
		PWMFLEN_ENABLE	Duty of 100% setting is enabled.
cPWMBLEN	Duty of 0% setting when the lower-limit	PWMBLEN_DISABLE	Duty of 0% setting is disabled.
		PWMBLEN_ENABLE	Duty of 0% setting is enabled.
cNICEN	Non-interference control	NICEN_DISABLE	Non-interference control is disabled.
		NICEN_ENABLE	Non-interference control is enabled.
cT5ECEN	Expansion control	T5ECEN_DISABLE	Expansion control is disabled.
		T5ECEN_ENABLE	Expansion control is enabled.
cAWUMD	Specifies anti-windup (AWU) control	AWUMD_DISABLE	AWU control is disabled.
		AWUMD_ENABLE4	Substitutes the result from amount of limitation / 4 into the integral term.
		AWUMD_ENABLE2	Substitutes the result from amount of limitation / 2 into the integral term.
		AWUMD_ENABLE1	Substitutes the amount of limitation into the integral term.
cCLPEN	Phase clipping control	CLPEN_DISABLE	Clipping is disabled.
		CLPEN_ENABLE	Clipping is enabled.
cATANMD	Specifies ATAN calculation control	ATANMD_DISABLE	Calculation is disabled.
		ATANMD_IDQ	Calculates the declination angle on d-q coordinate of current vector.
		ATANMD_EDQ	Calculates the declination angle on d-q coordinate of induced voltage vector.
cVDCSEL	Selects the supply voltage store register	VDCSEL_VDC	Stored in VExVDC.
		VDCSEL_VDCL	Stored in VExVDCL.
cZIEN	Zero current detection	ZIEN_DISABLE	Disable
		ZIEN_ENABLE	Enable
cPVIEN	Phase interpolation	PVIEN_DISABLE	Disable
		PVIEN_ENABLE	Enable

## 13. Temperature Measuring

### 13.1. Measuring Method

Computed to one decimal place by acquiring the AD value of Temp0 and using a linear equation between temperatures in reference to the values in the table.

e.g.)

20°C AD Judgment Value = 2146 (table) / 30°C AD Judgment Value = 1755 (table) / Temp0 = 1790

2146–1755 = 391 (fixed gradient within 20°C to 30°C)

2146–1790 = 356 (deviation from 20°C)

Interpolation  $10^{\circ}\text{C} \times 356 / 391 = 9.1^{\circ}\text{C}$

20°C + 9.1°C = 29.1°C

## 14. Status Check Monitor

Monitoring of current rotating speed, etc. on the TeraTerm is provided.

- TeraTerm Communication Setting

115,200 bps, data 8-bit, stop bit 1-bit, without parity and without flow control.

### 14.1. Initial Display

The initial status below is transmitted one time after releasing the reset.

Initial Status

Control ch = CH0

Carrier Frequency = xxxxx Hz

Dead Time = x.x  $\mu$ s

Gate Active = H/H or L/L

Position Detect = 3Shunt

VDC Voltage = xx.x V

Inverter Temp = xx.x degree

UVW 0A Voltage = x.xx V/x.xx V/x.xx V

DAC Data = A:xx B:xx C:xx D:xx (DAC mode 0 is displayed)

Internal AMP = Yes or No

Direction = CW or CCW

Modulation = two-phase or three-phase

Key Operation = Volume SW

### 14.2. Display during Rotation

The current rotating speed is displayed after the occurrence of the motor rotation command.

The displays including when the rotation is stopped and when the EMG is detected are as follows:

During Rotation (every single second): xx Hz

Rotating --> Stop: 0 Hz

Rotating --> EMG Detection: EMG\_S

### 14.3. DAC Mode Changeover Display

The following DAC modes are displayed when the DAC mode is changed over.

Mode 0: DAC Data = A: TMPREG0, B: TMPREG1, C: TMPREG2, D: theta.half[1]

Mode 1: DAC Data = A: Id\_ref, B: Id, C: Iq\_ref, D: Iq

Mode 2: DAC Data = A: omega\_com.half[1], B: omega.half[1], C: omega\_def, D: Iq\_ref

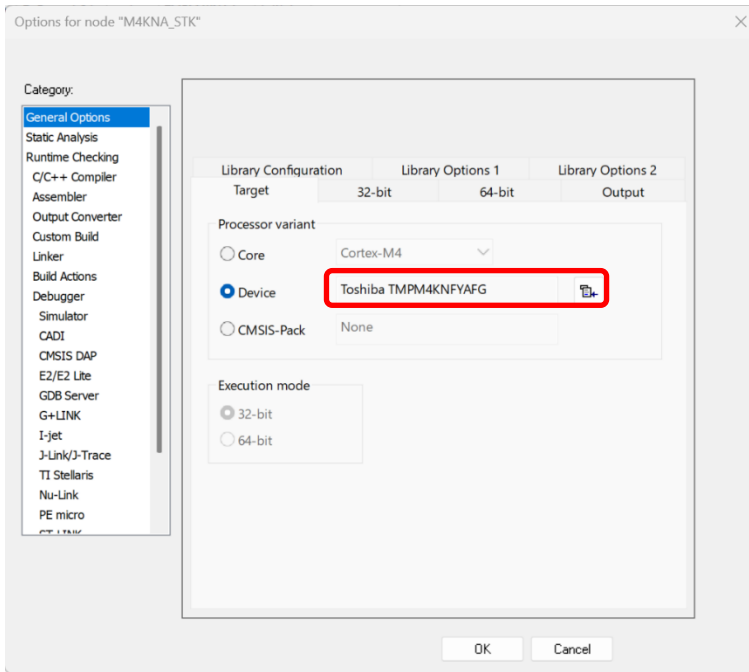
Mode 3: DAC Data = A: TMPREG0, B: Iq\_ref, C: Id\_ref, D: omega.half[1]

## 15. FAQ

### 15.1. When the EWARM project setup is deleted.

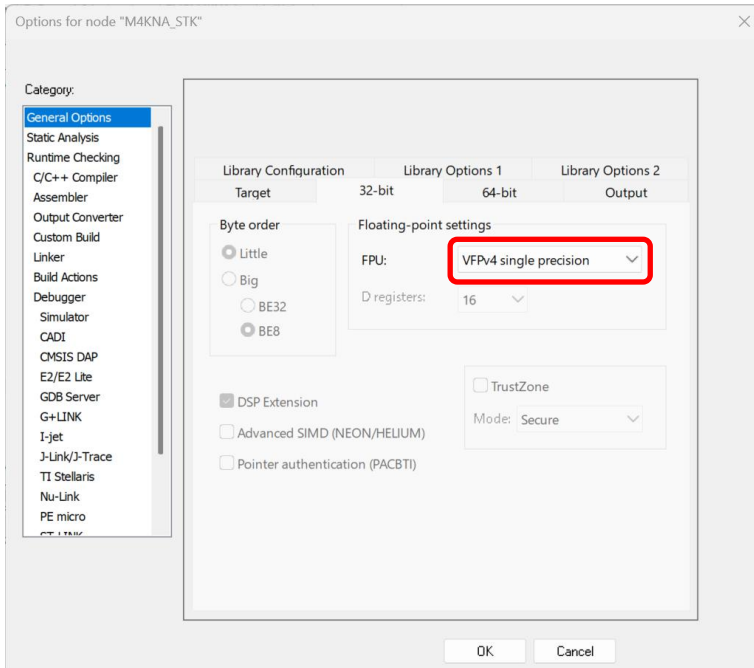
The setups may be deleted when a project is opened using an IDE of different version.  
Conduct the setup again in accordance with the following:

1. Project Menu > Option > General Option  
<Target> Tab



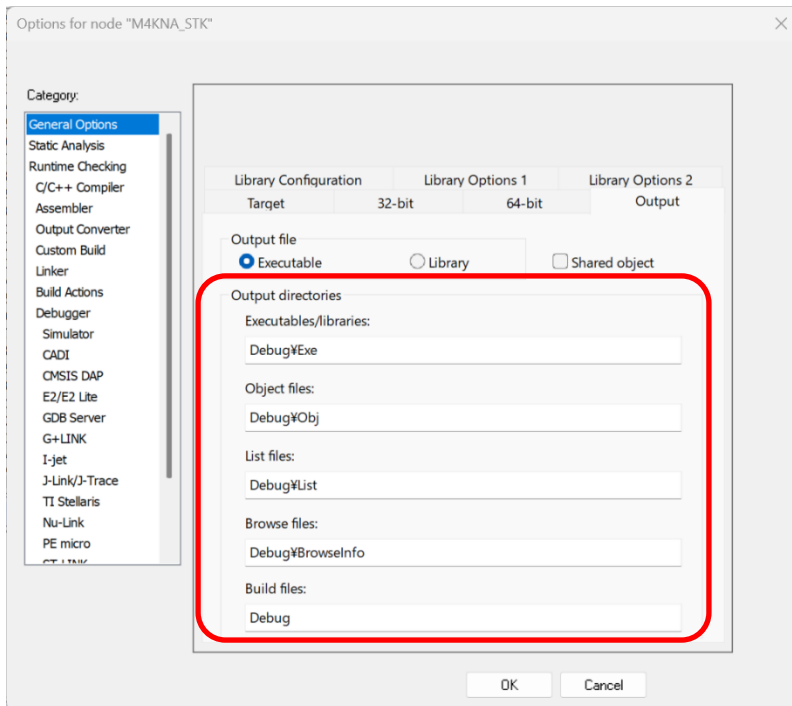
Select TMPM4KNFYAFG.

### <32-bit> Tab



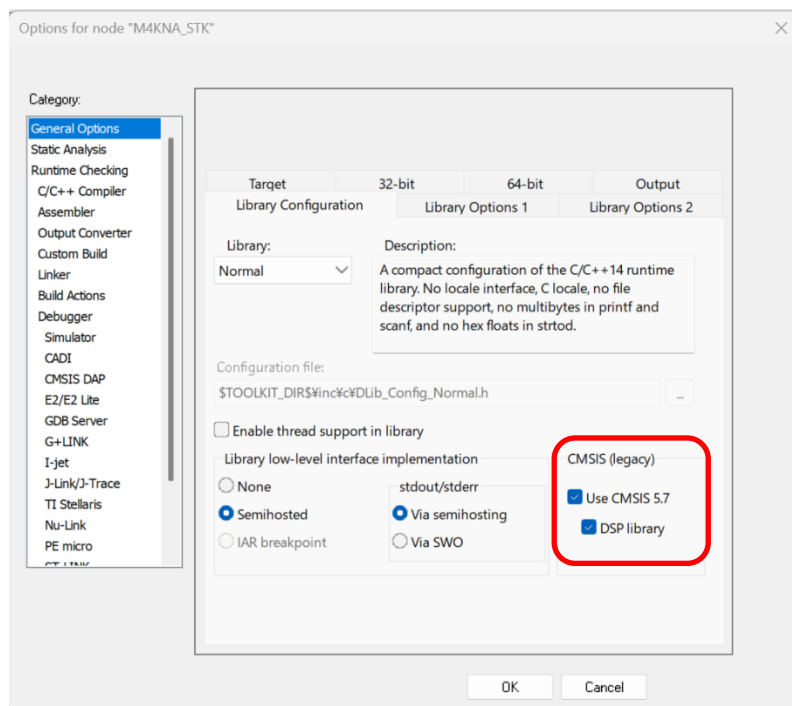
Select VFPv4 where the selection of FPU is possible.

## <Output> Tab



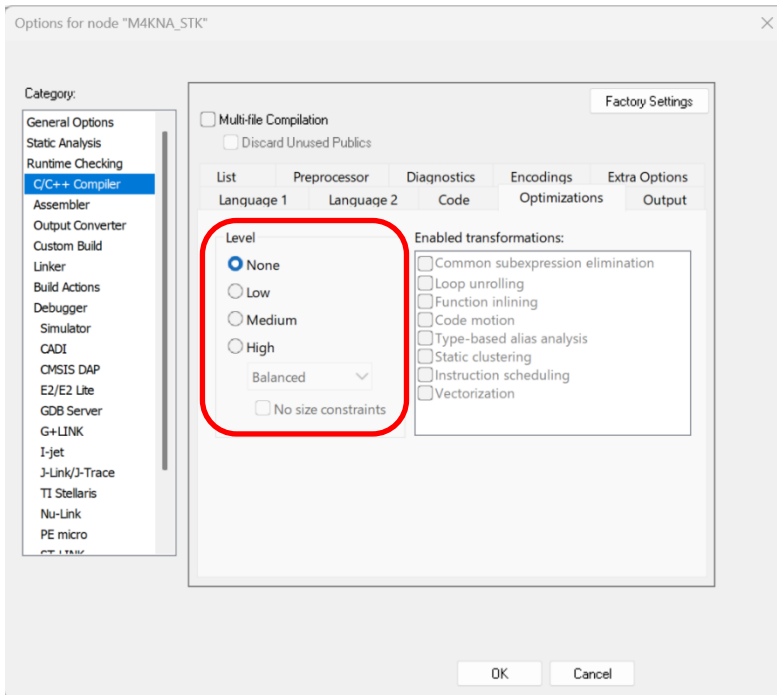
Set up the output directories. Defaults will not be problematic, but it is recommended to set up to a directory different from other projects.

## <Library Setup> Tab



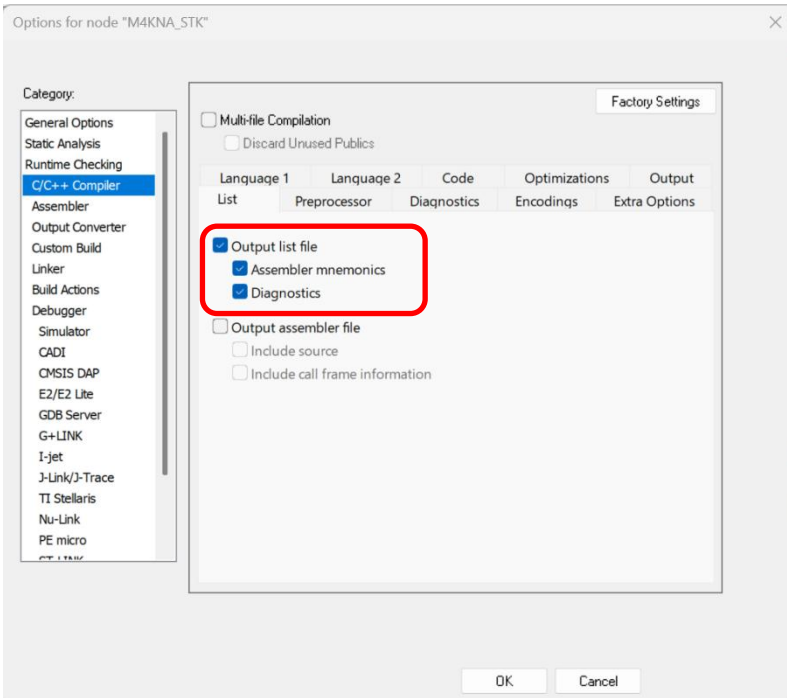
Check "Use CMCIS" and "DSP Library"

## 2. Project Menu > Option > C/C ++ Compiler <Optimization> Tab



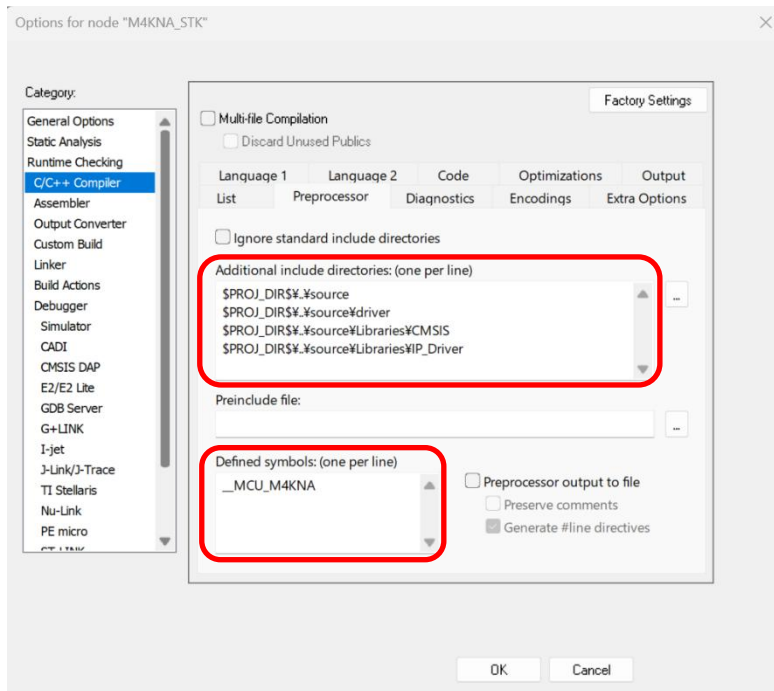
Set up the optimization level.

## <List> Tab



Check "Output List File"  
The compilation is possible without writing a check, but it would be convenient for evaluations if the list file is created.

## <Pre-Processor> Tab

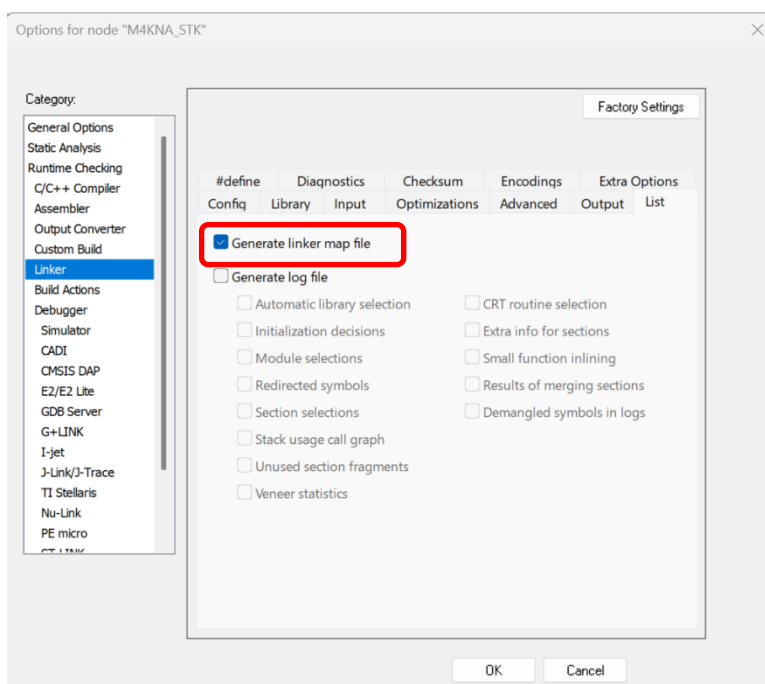


Add the setups below in the additionally included directories and the symbol definitions.

[TMPM4KNA]

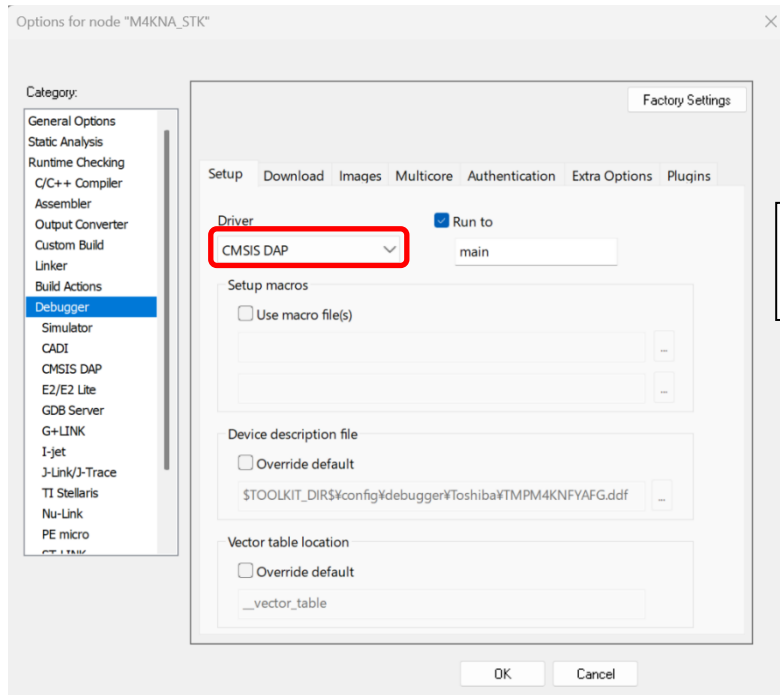
- Additionally Included Directory
  - \$PROJ\_DIR\$%.%source
  - \$PROJ\_DIR\$%.%source%driver
  - \$PROJ\_DIR\$%.%source%Libraries%CMSIS
  - \$PROJ\_DIR\$%.%source%Libraries%IP\_Driver
- Symbol Definition
  - \_\_MCU\_M4KNA

## 3. Project Menu > Option > Linker <List> Tab



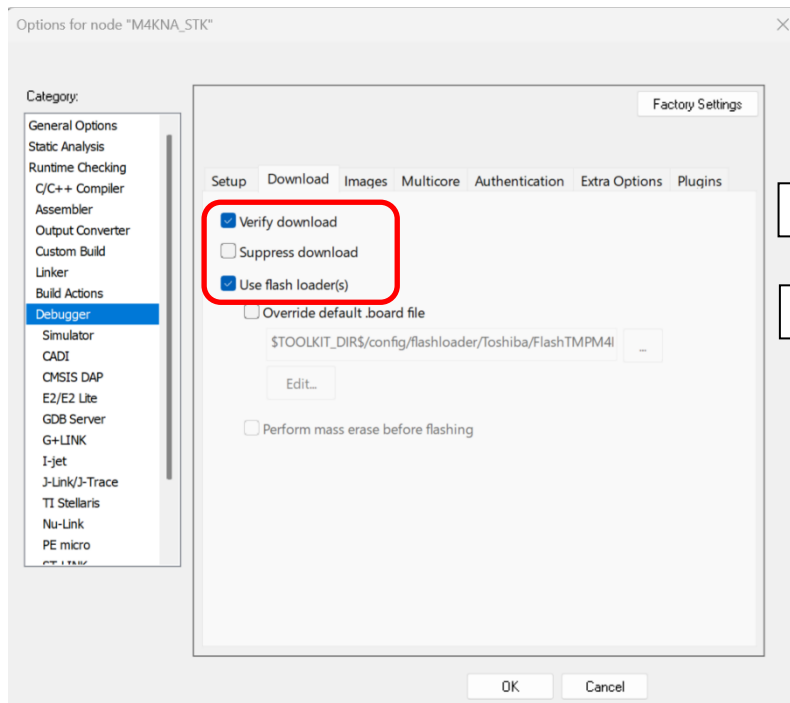
Check "Display Linker Map File"  
The compilation is possible without writing a check but it would be convenient for evaluations if the map file is created.

## 4. Project Menu > Option > Debugger <Setup> Tab



Select a tool to use.  
Select "CMSIS-DAP" for boards equipped with an onboard ICE.

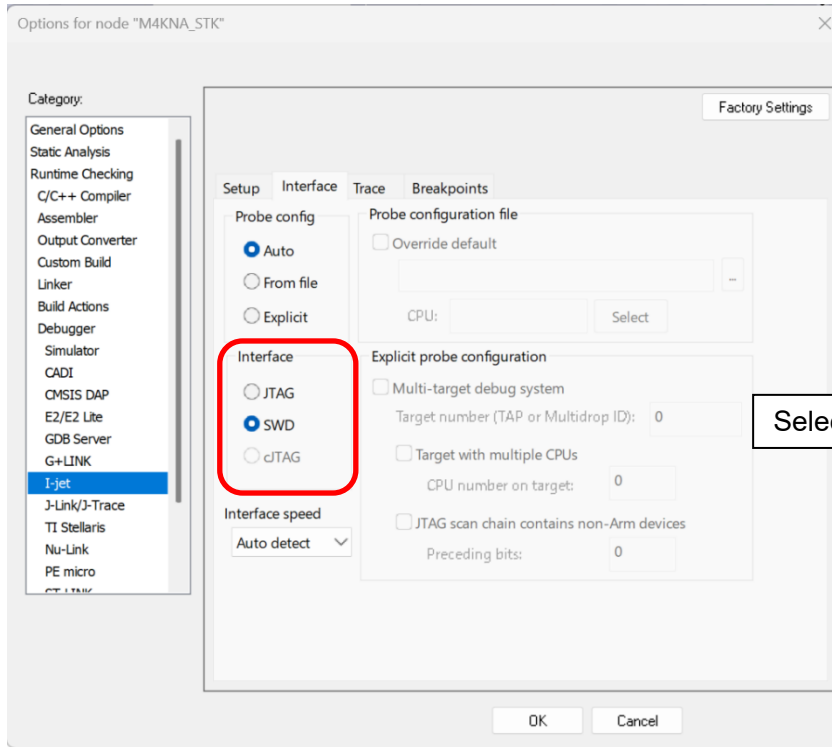
## <Download> Tab



Check "Verify"

Check "Use Flash Loader"

5. Project Menu > Option > CMSIS DAP or I-jet/JTAGjet  
- I-jet  
<Interface> Tab



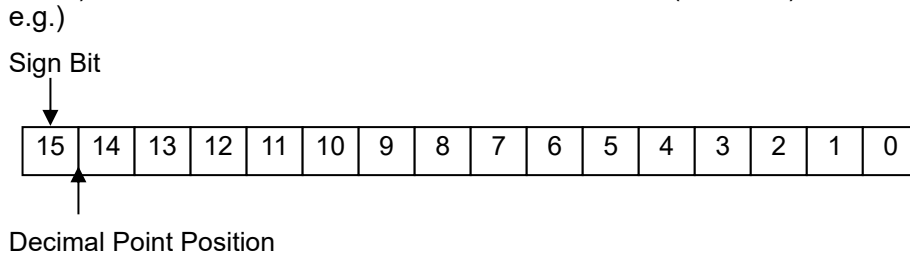
## 16. Appendix

### 16.1. Fixed Decimal Point Processing

The decimal arithmetic is executed with fixed decimal point in this software. A general explanation is given on the fixed decimal point computation.

### 16.2. Normalization

The normalization is to deform the data in accordance with the certain rules for ease of utilization. In this application note, the most significant bit is used as the sign bit (0: positive, 1: negative) and a decimal point is placed between the next bit, and normalized in such a way that the maximum possible value (0 x 7fff) for the data will be "1" and the minimum value (0 x 8000) will be "-1."



**Figure 16.1 Example of 16-Bit Data**

In this application note, the maximum value of current data is defined as cA\_Max: For example, A\_Max(A) when a 16 bit current data is 0x7fff and -A\_Max(A) for the case of 0x8000.

### 16.3. Data Format

The fixed decimal point computation is used in the motor controller of this application. The number of bits in the decimal fraction is expressed in Q format in the fixed decimal point computation. Basically for the case of 16-bit data, computed using Q15 format (15 bits for decimal fraction) and Q31 format for the case of 32-bit data (31 bits for decimal fraction). The value that can be displayed in the decimal format depends on the format.

**Table 16.1 Single Precision (16-bit) Decimal Fraction Format**

Q Format	Number of Bits for Decimal Fraction	Positive Maximum Value (0x7FFF)	Negative Maximum Value (0x8000)
Q15	15	0.999969482421875	-1
Q14	14	1.999938964843750	-2
Q13	13	3.999877929687500	-4
Q12	12	7.999755859375000	-8
Q11	11	15.999511718750000	-16
Q10	10	31.999023437500000	-32
Q9	9	63.998046875000000	-64
Q8	8	127.996093750000000	-128
Q7	7	255.992187500000000	-256
Q6	6	511.984375000000000	-512
Q5	5	1023.968750000000000	-1024
Q4	4	2047.937500000000000	-2048
Q3	3	4095.875000000000000	-4096
Q2	2	8191.750000000000000	-8192
Q1	1	16383.500000000000000	-16384

---

Q0	0	32767.0000000000000000	-32768
----	---	------------------------	--------

**Table 16.2 Double Precision (32-bit) Decimal Fraction Format**

Q Format	Number of Bits for Decimal Fraction	Positive Maximum Value (0x7FFFFFFF)	Negative Maximum Value (0x80000000)
Q31	31	0.99999999534338	-1
Q30	30	1.99999999068670	-2
Q29	29	3.99999998137350	-4
Q28	28	7.99999996274700	-8
Q27	27	15.99999992549400	-16
Q26	26	31.99999985098800	-32
Q25	25	63.99999970197600	-64
Q24	24	127.99999940395000	-128
Q23	23	255.99999880790000	-256
Q22	22	511.99999761581000	-512
Q21	21	1023.99999523160000	-1024
Q20	20	2047.99999046320000	-2048
Q19	19	4095.99998092650000	-4096
Q18	18	8191.99996185300000	-8192
Q17	17	16383.99992370600000	-16384
Q16	16	32767.99984741200000	-32768
Q15	15	65535.99969482400000	-65536
Q14	14	131071.99938965000000	-131072
Q13	13	262143.99877930000000	-262144
Q12	12	524287.99755859000000	-524288
Q11	11	1048575.99511720000000	-1048576
Q10	10	2097151.99902344000000	-2097152
Q9	9	4194303.99804687000000	-4194304
Q8	8	8388607.99609375000000	-8388608
Q7	7	16777215.99218750000000	-16777216
Q6	6	33554431.98437500000000	-33554432
Q5	5	67108863.96875000000000	-67108864
Q4	4	134217727.93750000000000	-134217728
Q3	3	268435455.87500000000000	-268435456
Q2	2	536870911.75000000000000	-536870912
Q1	1	1073741823.50000000000000	-1073741824
Q0	0	2147483647.00000000000000	-2147483648

## 16.4. Computation with Fixed Decimal Point

In the four arithmetic operations of fixed decimal point computation, addition and subtraction are performed as if they were integers, but in multiplication and division, the decimal point position of the result of computation will be shifted; accordingly, it is necessary to restore the original decimal point position.

### (1) Multiplication

In the case of multiplication between decimal fraction formats, for example, when multiplying Q15 format data, the result will be in the double precision Q30 format. When the Q31 format data is required, a double precision Q31 format is acquired by the left shifting of computation result by 1 bit.

$$Q15 * Q15 = 2^{-15} * 2^{-15} = 2^{(-15 + -15)} = 2^{-30} = Q30$$

### (2) Division

In the case of division between decimal fraction formats, for example, when the Q31 format is divided by the Q15 format, the result will be in the Q16 format. When the Q15 format data is required, the Q15 format data is acquired by right shifting of divisor by 1 bit before computing.

$$Q31 / Q15 = 2^{-31} / 2^{-15} = 2^{(-31 - (-15))} = 2^{-16} = Q16$$

**17. Revision History**

Date	Rev.	Description
2021-11-08	1.0	First release
2026-05-29	1.1	Added support for KES-PH.

## RESTRICTIONS ON PRODUCT USE

Toshiba Corporation and its subsidiaries and affiliates are collectively referred to as "TOSHIBA". Hardware, software and systems described in this document are collectively referred to as "Product".

- TOSHIBA reserves the right to make changes to the information in this document and related Product without notice.
- This document and any information herein may not be reproduced without prior written permission from TOSHIBA. Even with TOSHIBA's written permission, reproduction is permissible only if reproduction is without alteration/omission.
- Though TOSHIBA works continually to improve Product's quality and reliability, Product can malfunction or fail. Customers are responsible for complying with safety standards and for providing adequate designs and safeguards for their hardware, software and systems which minimize risk and avoid situations in which a malfunction or failure of Product could cause loss of human life, bodily injury or damage to property, including data loss or corruption. Before customers use the Product, create designs including the Product, or incorporate the Product into their own applications, customers must also refer to and comply with (a) the latest versions of all relevant TOSHIBA information, including without limitation, this document, the specifications, the data sheets and application notes for Product and the precautions and conditions set forth in the "TOSHIBA Semiconductor Reliability Handbook" and (b) the instructions for the application with which the Product will be used with or for. Customers are solely responsible for all aspects of their own product design or applications, including but not limited to (a) determining the appropriateness of the use of this Product in such design or applications; (b) evaluating and determining the applicability of any information contained in this document, or in charts, diagrams, programs, algorithms, sample application circuits, or any other referenced documents; and (c) validating all operating parameters for such designs and applications. **TOSHIBA ASSUMES NO LIABILITY FOR CUSTOMERS' PRODUCT DESIGN OR APPLICATIONS.**
- **PRODUCT IS NEITHER INTENDED NOR WARRANTED FOR USE IN EQUIPMENTS OR SYSTEMS THAT REQUIRE EXTRAORDINARILY HIGH LEVELS OF QUALITY AND/OR RELIABILITY, AND/OR A MALFUNCTION OR FAILURE OF WHICH MAY CAUSE LOSS OF HUMAN LIFE, BODILY INJURY, SERIOUS PROPERTY DAMAGE AND/OR SERIOUS PUBLIC IMPACT ("UNINTENDED USE").** Except for specific applications as expressly stated in this document, Unintended Use includes, without limitation, equipment used in nuclear facilities, equipment used in the aerospace industry, Class 3 medical devices, equipment used for automobiles, and military vehicles and munitions. **IF YOU USE PRODUCT FOR UNINTENDED USE, TOSHIBA ASSUMES NO LIABILITY FOR PRODUCT. For details, please contact your TOSHIBA sales representative or contact us via our website.**
- Product shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable laws or regulations.
- The information contained herein is presented only as guidance for Product use. No responsibility is assumed by TOSHIBA for any infringement of patents or any other intellectual property rights of third parties that may result from the use of Product. No license to any intellectual property right is granted by this document, whether express or implied, by estoppel or otherwise.
- **ABSENT A WRITTEN SIGNED AGREEMENT, EXCEPT AS PROVIDED IN THE RELEVANT TERMS AND CONDITIONS OF SALE FOR PRODUCT, AND TO THE MAXIMUM EXTENT ALLOWABLE BY LAW, TOSHIBA (1) ASSUMES NO LIABILITY WHATSOEVER, INCLUDING WITHOUT LIMITATION, INDIRECT, CONSEQUENTIAL, SPECIAL, OR INCIDENTAL DAMAGES OR LOSS, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, LOSS OF OPPORTUNITIES, BUSINESS INTERRUPTION AND LOSS OF DATA, AND (2) DISCLAIMS ANY AND ALL EXPRESS OR IMPLIED WARRANTIES AND CONDITIONS RELATED TO SALE, USE OF PRODUCT, OR INFORMATION, INCLUDING WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OF INFORMATION, OR NONINFRINGEMENT.**
- Do not use or otherwise make available Product or related software or technology for any military purposes, including without limitation, for the design, development, use, stockpiling or manufacturing of nuclear, chemical, or biological weapons or missile technology products (mass destruction weapons). Product and related software and technology may be controlled under the applicable export laws and regulations including, without limitation, the Japanese Foreign Exchange and Foreign Trade Law and the U.S. Export Administration Regulations. Export and re-export of Product or related software or technology are strictly prohibited except in compliance with all applicable export laws and regulations.
- Please contact your TOSHIBA sales representative for details as to environmental matters such as the RoHS compatibility of Product. Please use Product in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. **TOSHIBA ASSUMES NO LIABILITY FOR DAMAGES OR LOSSES OCCURRING AS A RESULT OF NONCOMPLIANCE WITH APPLICABLE LAWS AND REGULATIONS.**