TMPM471
Motor Sample Software
Application Note

Contents

1. Overview

1.1 Introduction

The operation of this motor control sample software has been checked using the SBK-M471 (TMPM471F10FG microcontroller evaluation board) by ESP and the KES-P2 (inverter board). Please contact ESP (http://esp.co.jp/) for details on this evaluation board.

1.2 Overview of Specifications

- Microcontroller --- TMPM471F10FG
- Clock --- 160MHz(ADC, A-PMD, A-ENC32) 80MHz(TSPI, T32A, UART)
- Evaluation motor --- Tsukasa Electric - TG611B
- Development environment --- IAR Embedded Workbench V9.60.2
  Keil µVision MDK V5.40.0.5
  SEGGER Embedded Studio V8.18
- DC link voltage --- 24 VDC

- Control outline
  - 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
Motor Operation	Stop or start
Rotation Speed Control Method	Constant rotation speed control (switching)
Rotation Direction	Fixed as CW
PWM Modulation	Three-phase or two-phase modulation
Current Detection	3-shunt or 1-shunt
Position Detection	Sensor-less, encoder(Note1) , or Encoder and Hall (Note1)

Note1: Not evaluated

Note: Precautions

- When an EMG occurs, use "Reset" for releasing.
- When operating with 1-shunt, check the setups as follows:

  - D_Para_CHx.h (Note1)

    #define cSHUNT_TYPE_CHx (1) (Note1)

    #define cBOOT_TYPE_CHx (cBoot_v) (Note1)

  Note1: Choose a suitable number from "0,1" 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.

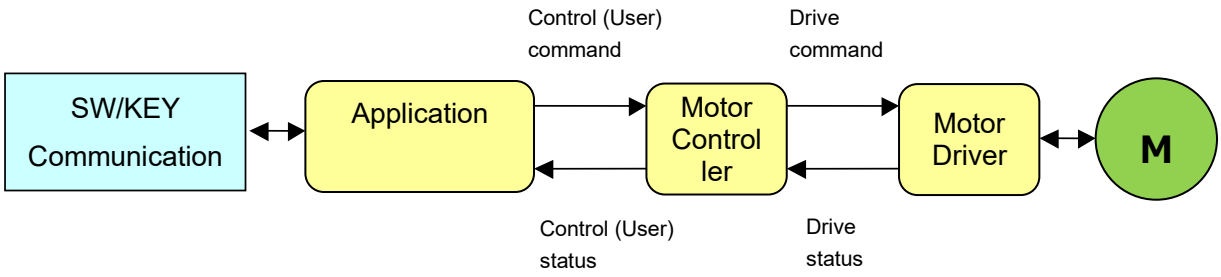


Figure 1.1 Vector Control Software Configuration

i. 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.

ii. 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.

iii. 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.

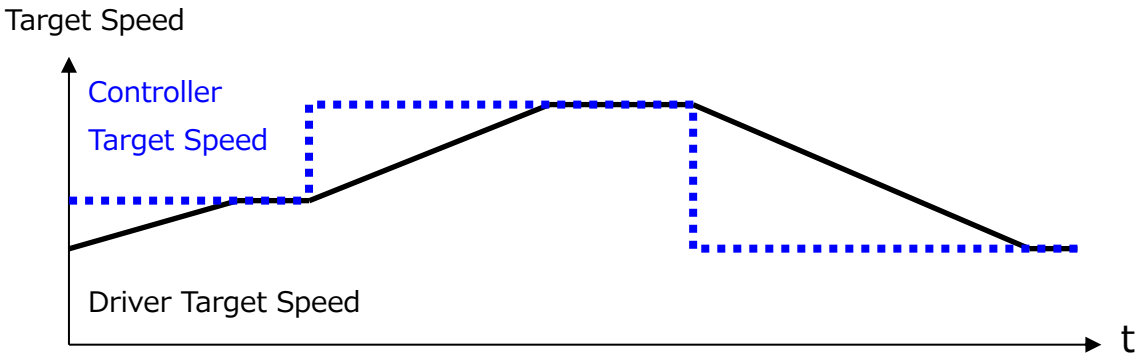


Figure 1.2 Controller Target Speed and Driver Target Speed

2. System Block Diagram

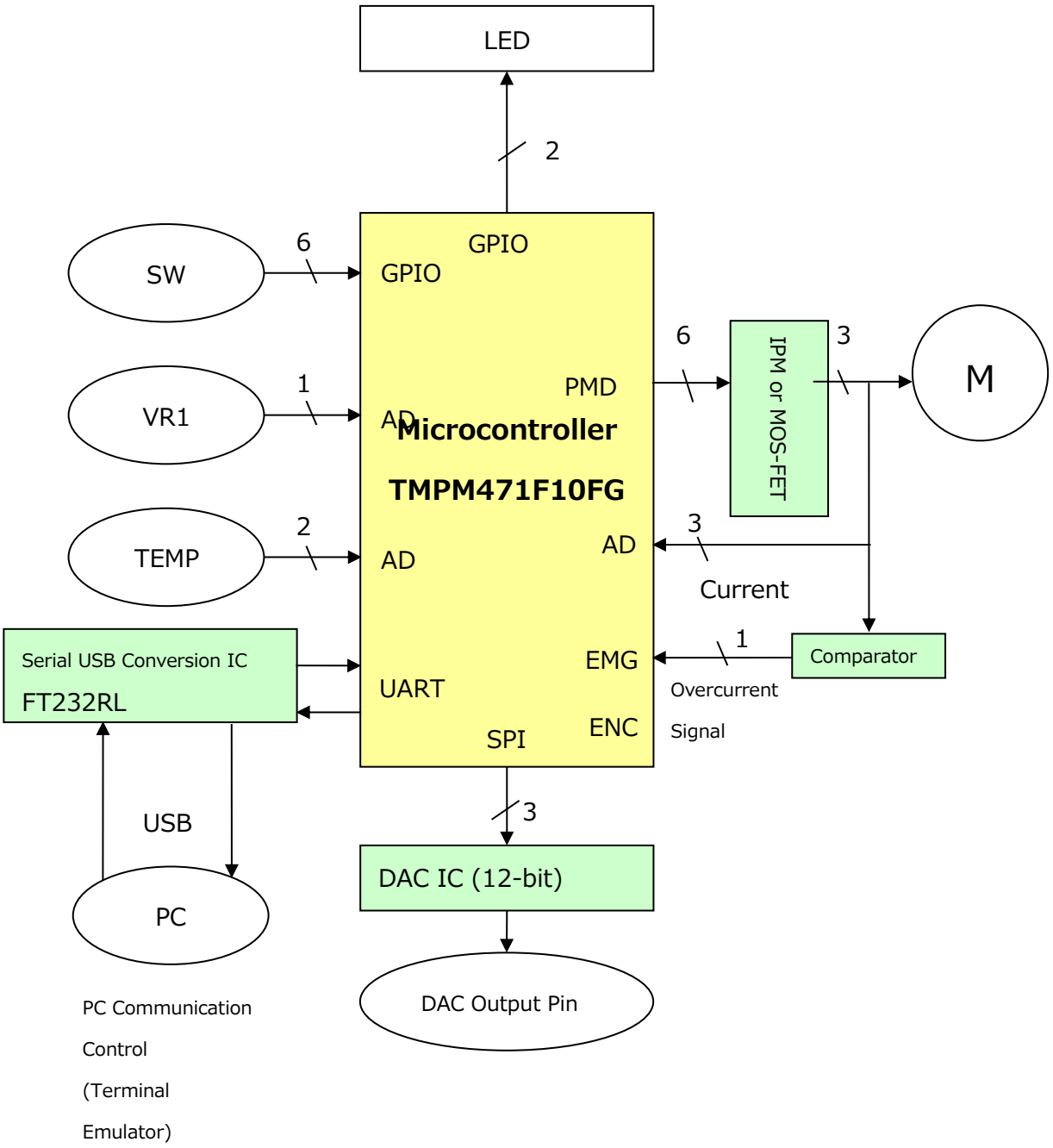


Figure 2.1 System Block Diagram

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
│   CheckVdq.c              dq-Axis Voltage Check Source File
│   CheckVdq.h              dq-Axis Voltage Check Header File
│   dac_drv.c               DAC IC Driver Source File
│   dac_drv.h               DAC IC Driver 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
│   D_Para_ch1.h            Vector Control Parameter (Channel 1) Header File
│   E_Sub.c                 Computing Function Source File
│   E_Sub.h                 Computing 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
│   H_Hall.c                Vector Control for Hall Sensor Source File
│   H_Hall.h                Vector Control for Hall Sensor header File
│   ipdefine.h              Microcontroller Setup 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
│   ipdrv.c                 Microcontroller Hardware Reference Source File
│   ipdrv.h                 Microcontroller Hardware Reference Header File
│   sys_macro.h             Macro Definition Header File
│   usercon.c               User Control Source File
│   usercon.h               User Control Header File
│
└─Libraries The libraries for CMSIS Core and individual IPs are stored.
    └─M471F10
        ├─CMSIS CMSIS Core is stored.
        │   │   system_TMPM471F10.c      Microcontroller Initial Setup Source File
        │   │   system_TMPM471F10.h      Microcontroller Initial Setup Header File
        │   │   TMPM471F10.h             Microcontroller Register Definition Header File
        │   └─startup
        │       ├─arm
        │       │       startup_TMPM471F10.s      Start Up Assembler Source (arm)
        │       │       TMPM471F10.scat           Linker File (arm)
        │       └─iar
        │       │       startup_TMPM471F10.s      Start Up Assembler Source (IAR)
```

```
│       │       TMPM471F10.icf          Linker File (IAR)
│       └─segger
│               TMPM471F10_Vectors.s     Start Up Assembler Source (SEGGER)
│               SEGGER_THUMB_Startup.s   Start Up Assembler Source (SEGGER)
│               TX04_M471_Startup.s      Start Up Assembler Source (SEGGER)
│               TX04_M471_Flash.icf      Linker File (SEGGER)
│
└─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
```

4. Evaluation Environment

4.1 Development Tools

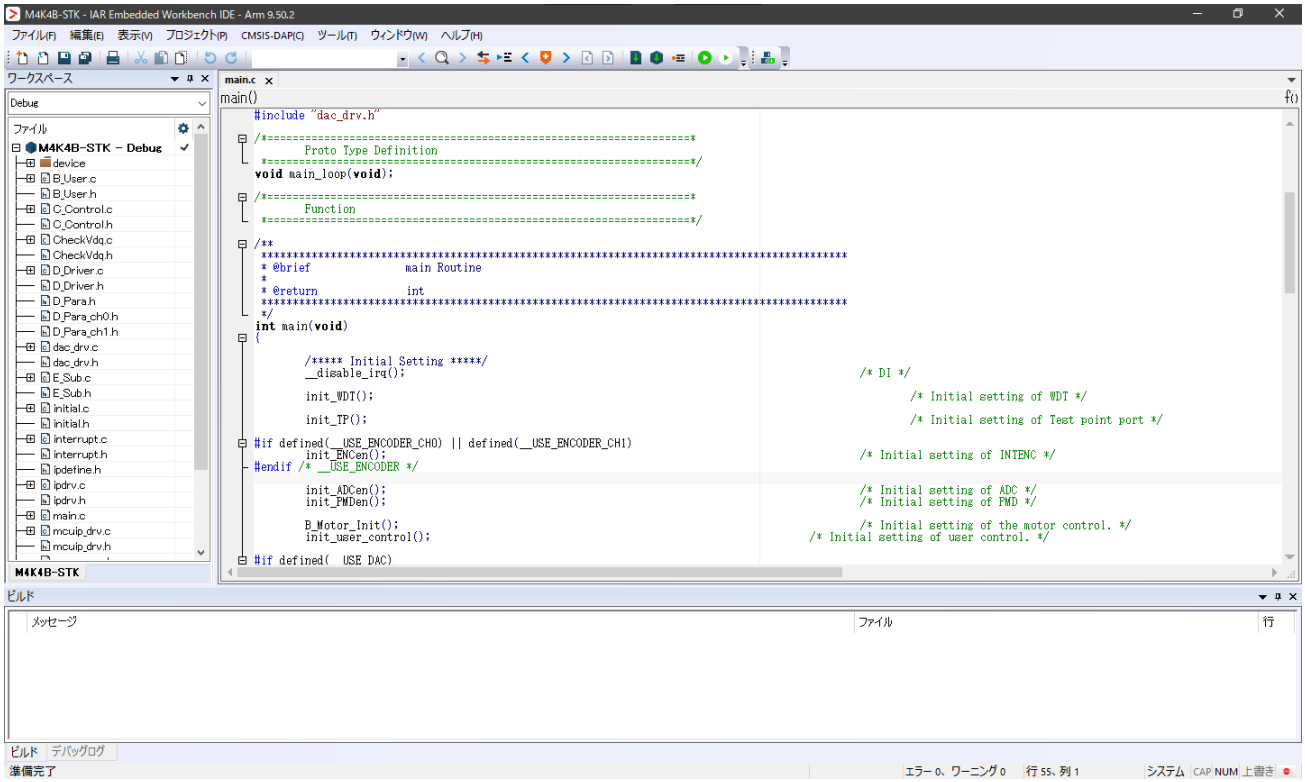
This software was developed using the following development tools:

IAR Embedded Workbench for ARM	9.60.2
Keil µVision MDK	5.40.0.5
Segger Embedded Studio	8.18

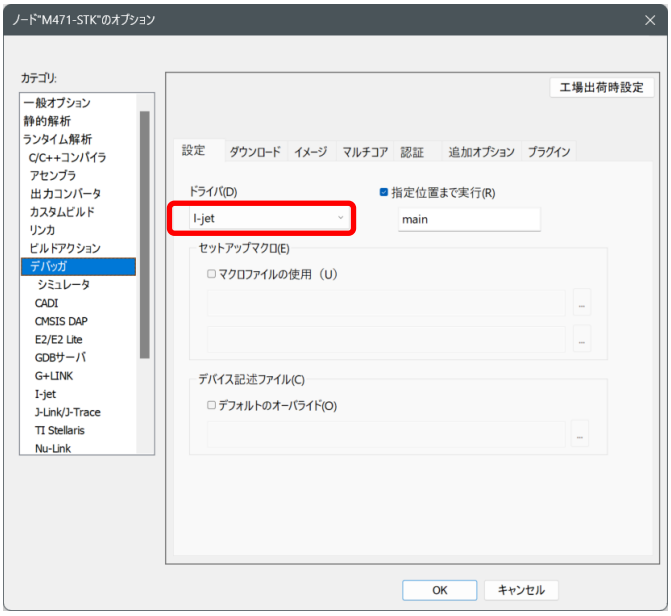
4.2 How to Start Up a Project

The procedure is explained using IAR Embedded Workbench as an example.

1. Double-click iar¥M471-STK.eww, or select [File] > [Open] > [Workspace], then open M471-STK.eww.
2. The following window appears.



3. Open the Option window to select a tool to use.
   [Project] > [Option] > [Debugger], then the <Setup> tab page

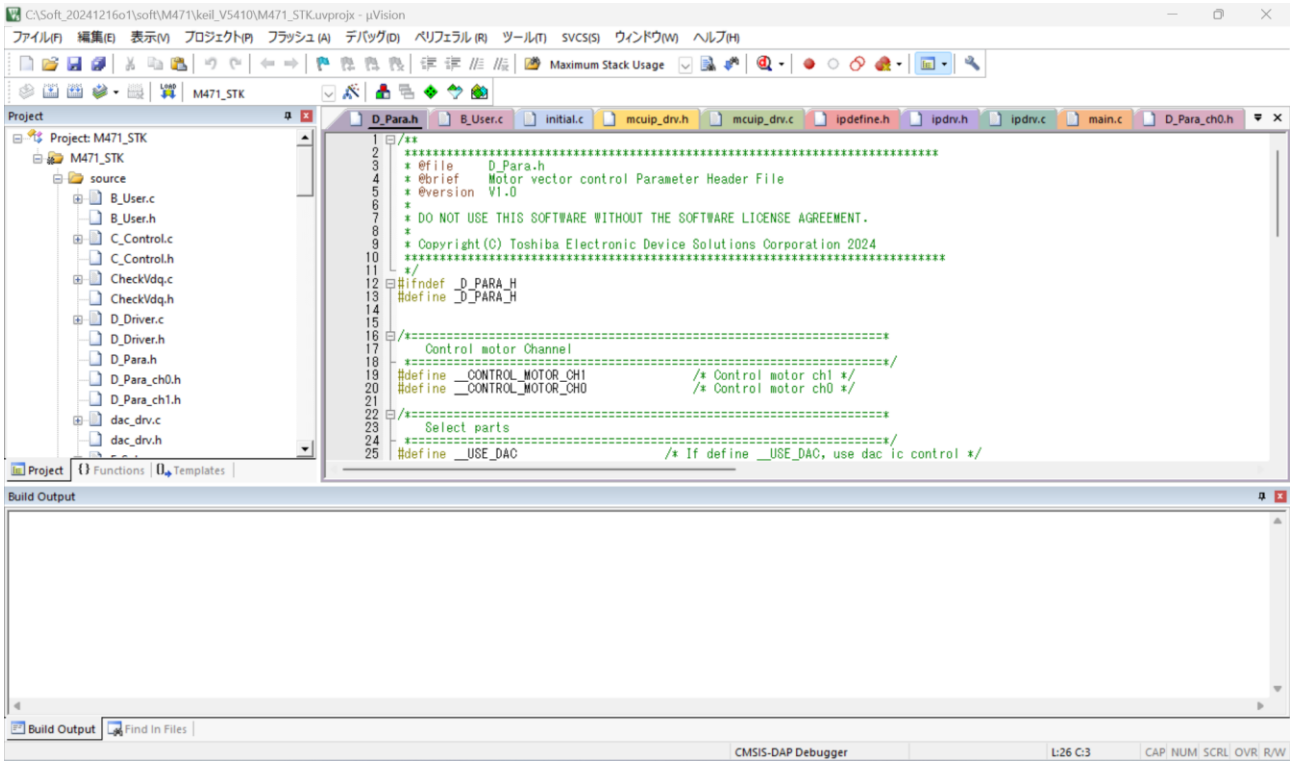


Select a tool to use.

4. To start debugging, connect the tool and select [Project] > [Download and Debug], or press the following button.

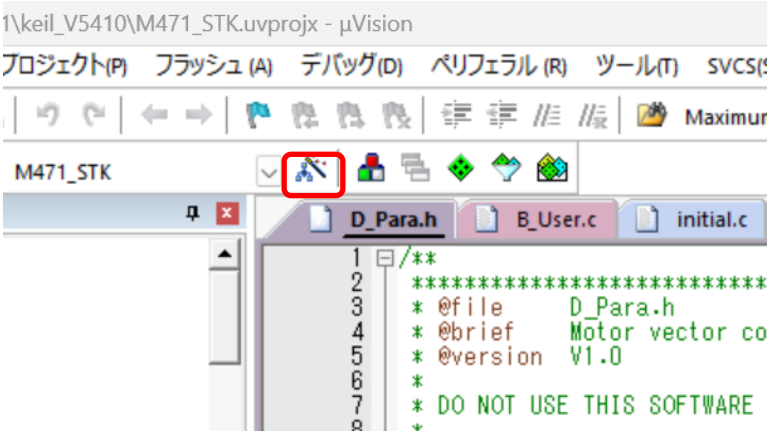
The following explains the case of Take Keil µVision MDK.

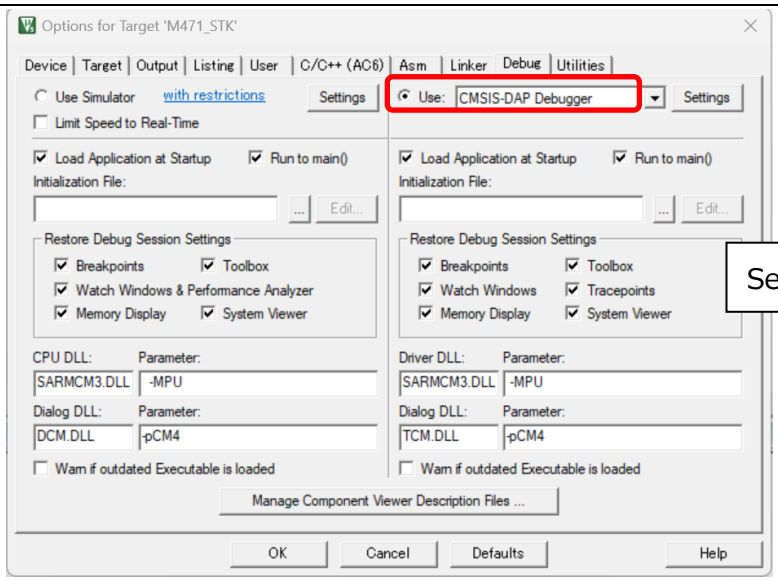
1. Double-click M471_STK.uvprojx, or select [File] > [Open], then open M471_STK.uvprojx.
2. The following window appears.



3. Open the Options for Target window and select a tool to use.

   [Options for Target] > [Debug], then the <Use> field



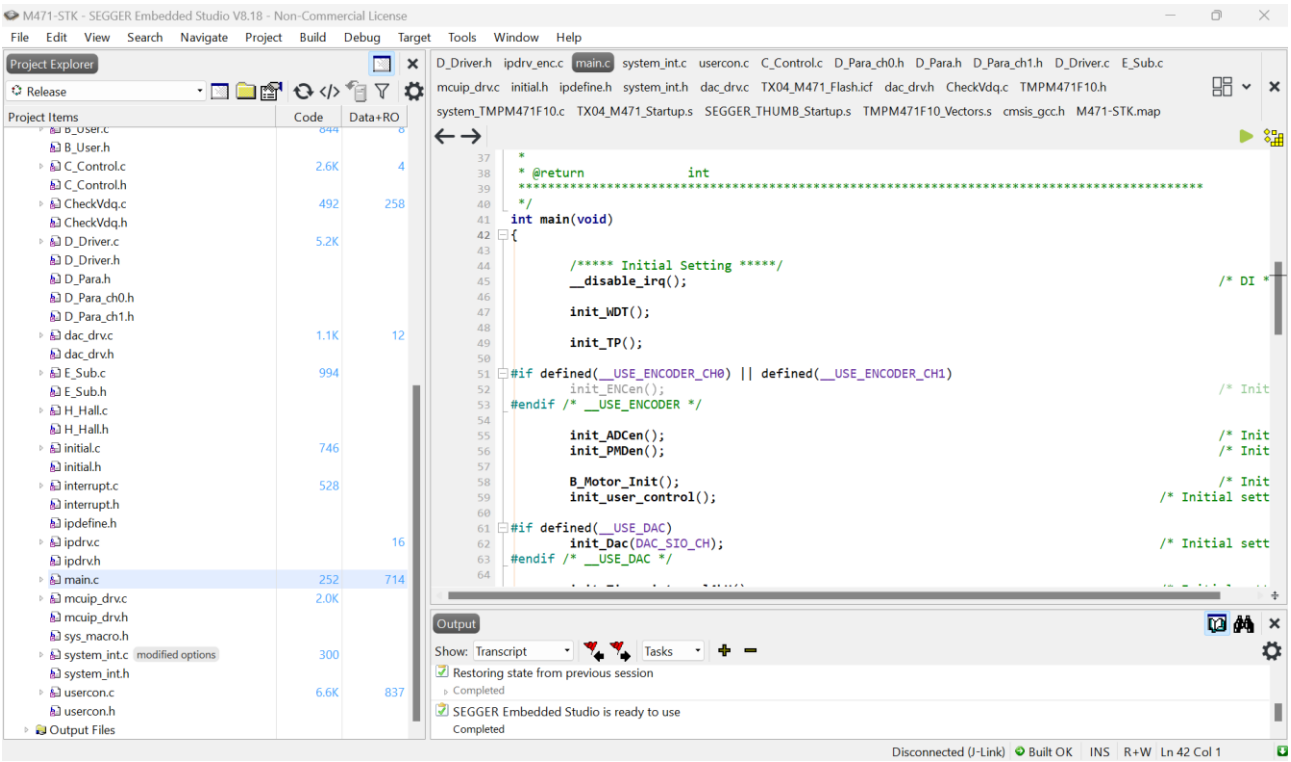


Select a tool to use.

4. To start debugging, connect the tool and select [Debug] > [Start/Stop Debug Session], or press the following button.

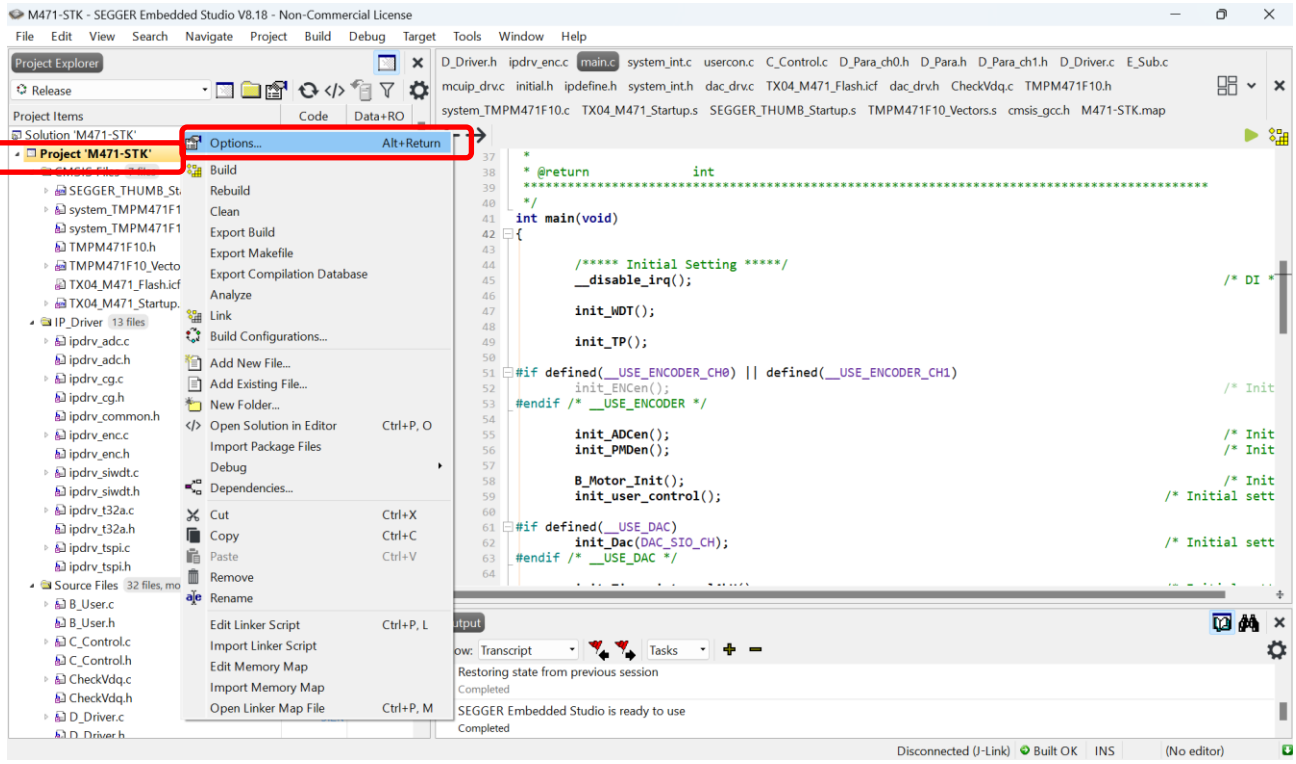
The following explains the case of the Segger Embedded Studio.

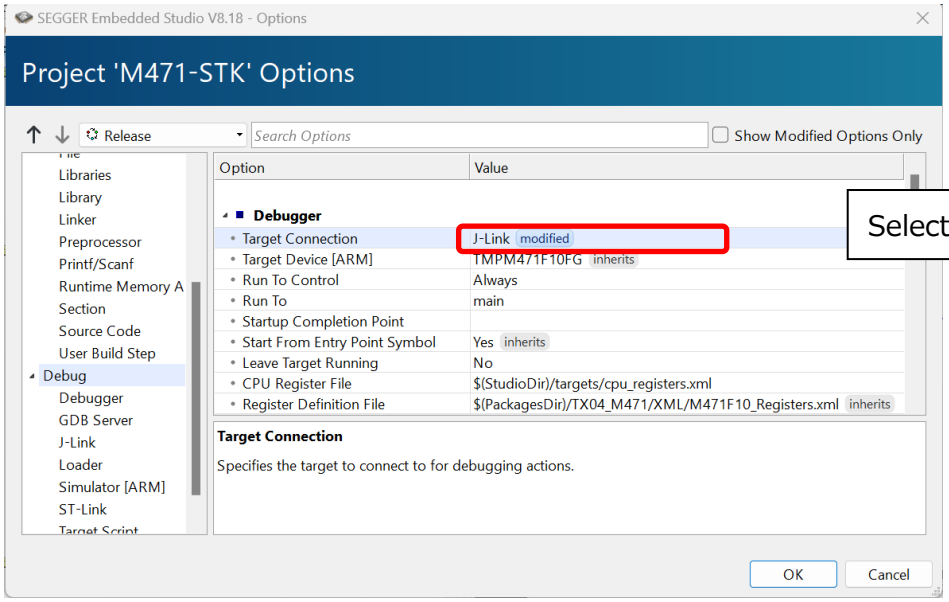
1. Double-click M471_STK.emProject, or select [File] > [Open], then open M471_STK.emProject.
2. The following window appears.



3. Open Options and select a tool to use.

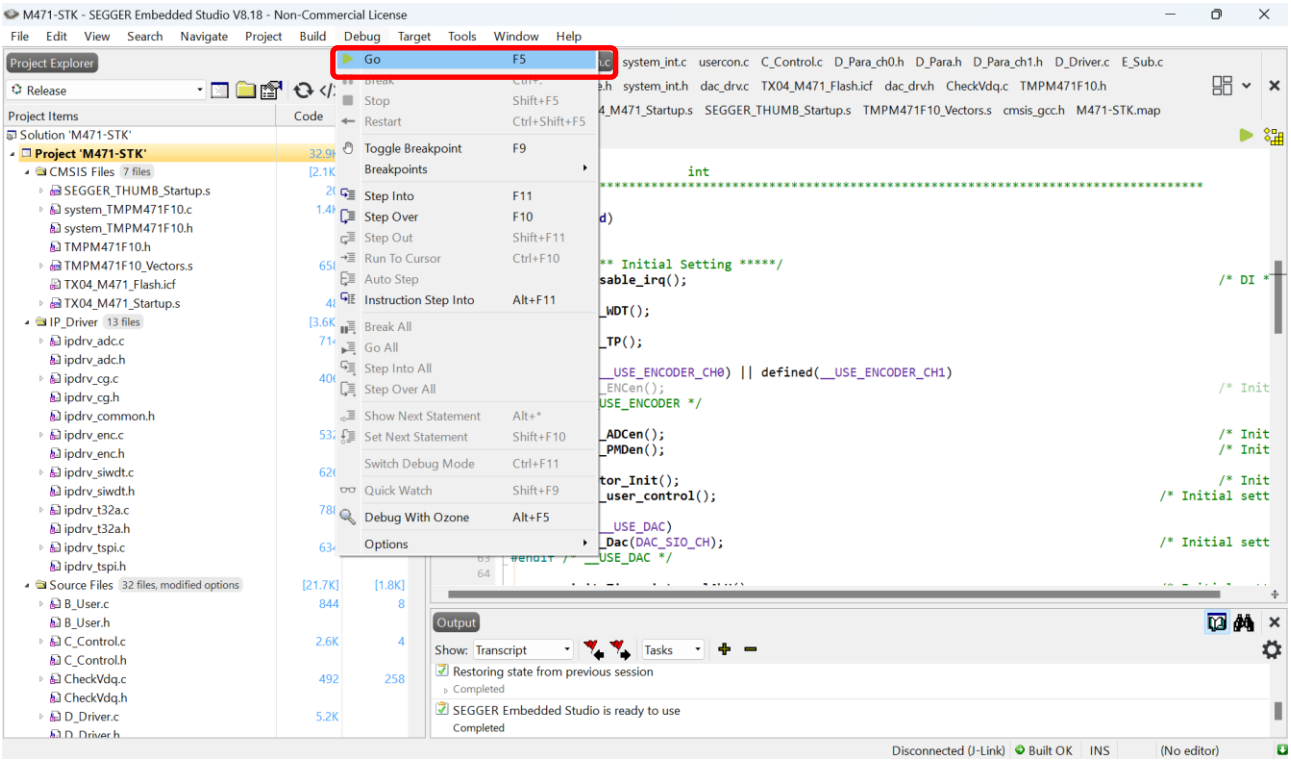
Right-click on the project name > [Options] > [Debugger], then <Target Connection>.





Select a tool to use.

4. To start debugging, connect the tool and select [Debug] > [Go].



4.3 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 the function does not have any variable for checking.

<<DAC Output Setup Variables>>

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

4.4 Port Output

It is possible to output various timings on the port.

The initial setting of the port to be output is as follows, so change it as necessary

4.4.1 Vector Control Software Processing Time

It is possible to output the software processing time of vector control.

To enable this function, please enable the following definition in ipdefine.h file.

#define __MOTOR_DBGOUT_VECTOR_TIME_INT

Table 4.1 Output port

Motor Channel	Output port
CH0	PL bit0
CH1	PL bit1

H：Under calculation

4.4.2 Current Detection State

It is possible to output the current detection state.

To enable this function, please enable the following definition in ipdefine.h file.

#define __MOTOR_DBGOUT_IDET

Table 4.2 Output port

Motor Channel	Output Port
CH0,CH1	PL bit0

L：Normal detection state, H：False detection state

4.5 Stage history

By leaving the history of the main stage, it is possible to check from which state an abnormal condition has come.

The history of the stage is stored in the following variables.

stage_his[]

To enable this function, please enable the following definition in ipdefine.h.

#define __MOTOR_DEBUG

4.6 User Interface

The following functions are provided as the user interface:

<<User Interface>>

1. Motor ch0 Operation

   The operation of Motor ch0 is changed over by switching the SW (S_SW1).

   off: Stop on: Start (0 Hz)

2. Motor ch1 Operation

The operation of Motor ch1 is changed over by switching the SW (S_SW2).

off: Stop on: Start (0 Hz)

3. Change of operation target in rotation speed/DAC status display

   The operation target in the rotation speed/DAC status display is changed over by switching the SW (S_SW3).

   off: Motor ch0 on: Motor ch1

4. Changeover between Up and Down of rotation speed

   Whether to make the rotation speed Up or Down is changed over by switching the SW (S_SW4).

   off: Up on: Down

5. Rotation speed adjustment

   The rotation speed is adjusted according to the table below by pressing the SW (USW1).

   0Hz(min)～100Hz(max)

Table 4.3 Rotation speed adjustment

No.	S_SW4	After Pressing USW1
1	Off: Up	10 Hz Up
2	On: Down	10 Hz Down

6. DAC Display

   The DAC status is displayed according to the table below by pressing the SW (USW2).

   Press normally: DAC display 3seconds Hold down: DAC No. change

Table 4.4 DAC Display

No.	DAC Display
1	A: Ia B: Ib C: Ic D: theta.half[1]
2	A: Id_ref B: Id C: Iq_ref D: Iq
3	A: omega_com.half[1] B: omega.half[1] C: omega_ref D: Iq_ref
4	A: Ia B: Iq_ref C: Id_ref D: omega.half[1]

7. Variable resistor

   Fake Temperature Adjustment is performed by operating VR1.

   To use Fake Temperature Adjustment, enable the following definition in D_Para.h.

   #define __USE__FAKETEMP

8. “LED Display”

   The motor error status is checked by the LED.

   For hardware EMG in the STOP stage, the LED is not lit.

   When Motor CH0 is driving (LED4)

   When Motor CH1 is driving (LED5)

   1. Error Status LED

      No EMG: Lights OFF

      EMG exists: Lights ON

9. Status display through UART connection

   The settings, motor rotation speed, and EMG status are displayed on the PC.

   The initial status can be checked once after releasing the reset.

   For details, refer to 9.1.8 Status Check Monitor.

5. Module Configuration

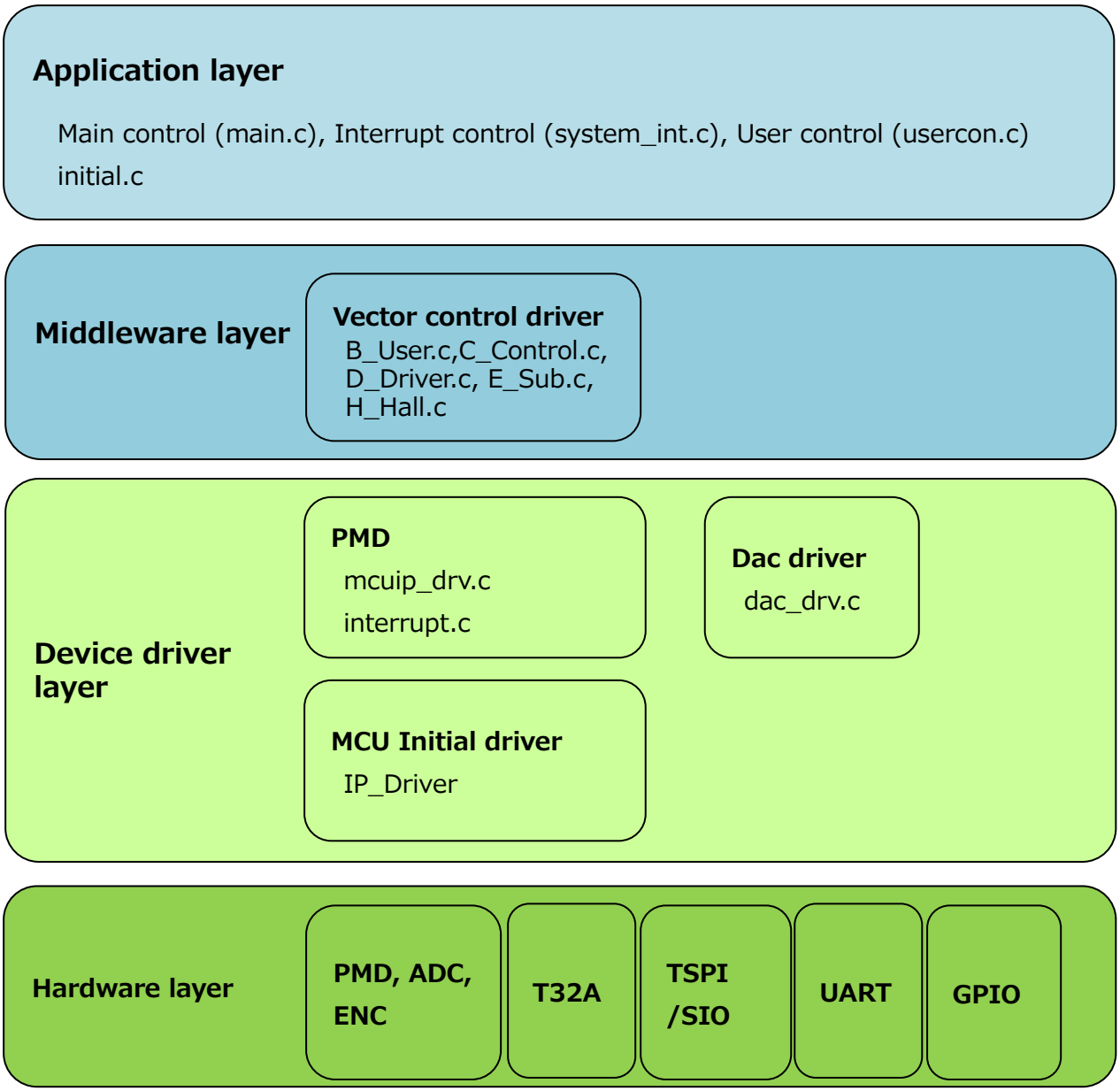


Figure 5.1 Module Configuration

6. Microcontroller Hardware Assignment

6.1 IP

IP		Control	Remarks
T32A	channel 0	4 kHz Interruption	Timer A
	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	Not Used	
	channel 2	DAC IC Control	SIO Mode
	channel 3	Not Used	
UART	channel 0	PC Connection	
	channel 1	Not Used	
	channel 2	Not Used	
	channel 3	Not Used	
ADC	Unit A	Motor CH0 Acquisition of Coil Current, Power Supply Voltage VR1, TEMP0 Voltage Value Acquisition	
	Unit B	Motor CH1 Acquisition of Coil Current, Power Supply Voltage TEMP1 Voltage Value Acquisition	
PMD	channel 0	Motor CH0 Control	
	channel 1	Motor CH1 Control	
ENC	channel 0	Motor CH0 Encoder Signal Control	
	channel 1	Motor CH1 Encoder Signal Control	

6.2 Interruption

Table 6.1 Interruption

Cause	Process	Priority	Function
INTT32A0AC_IRQn	4 kHz Interval Timing Generation	5	INTT32A0AC_IRQHandler
INTADAPDA_IRQn	Vector Control Software Processing for CH0 (Vector Control Using Software)	3	INTADAPDA_IRQHandler
INTADBPDB_IRQn	Vector Control Software Processing for CH1 (Vector Control Using Software)	3	INTADBPDB_IRQHandler
INTSC0TX_IRQn	UART Transmission Complete Interruption Processing	6	INTSC0TX_IRQHandler
INTSC2TX_IRQn	DAC IC Control	6	INTSC2TX_IRQHandler

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

```
/* High Low */
/* 0 ------ 7 */
INT4KH_LEVEL 5 /* 4kH interval timer interrupt */
INT_ADCA_LEVEL 3 /* ADC A interrupt */
INT_ADCB_LEVEL 3 /* ADC B interrupt */
INT_DAC_LEVEL 6 /* SIO interrupt for Dac */
INT_UART_LEVEL 6 /* SIO interrupt for UART */
```

7. General Flowchart

7.1 Main Routine (main)

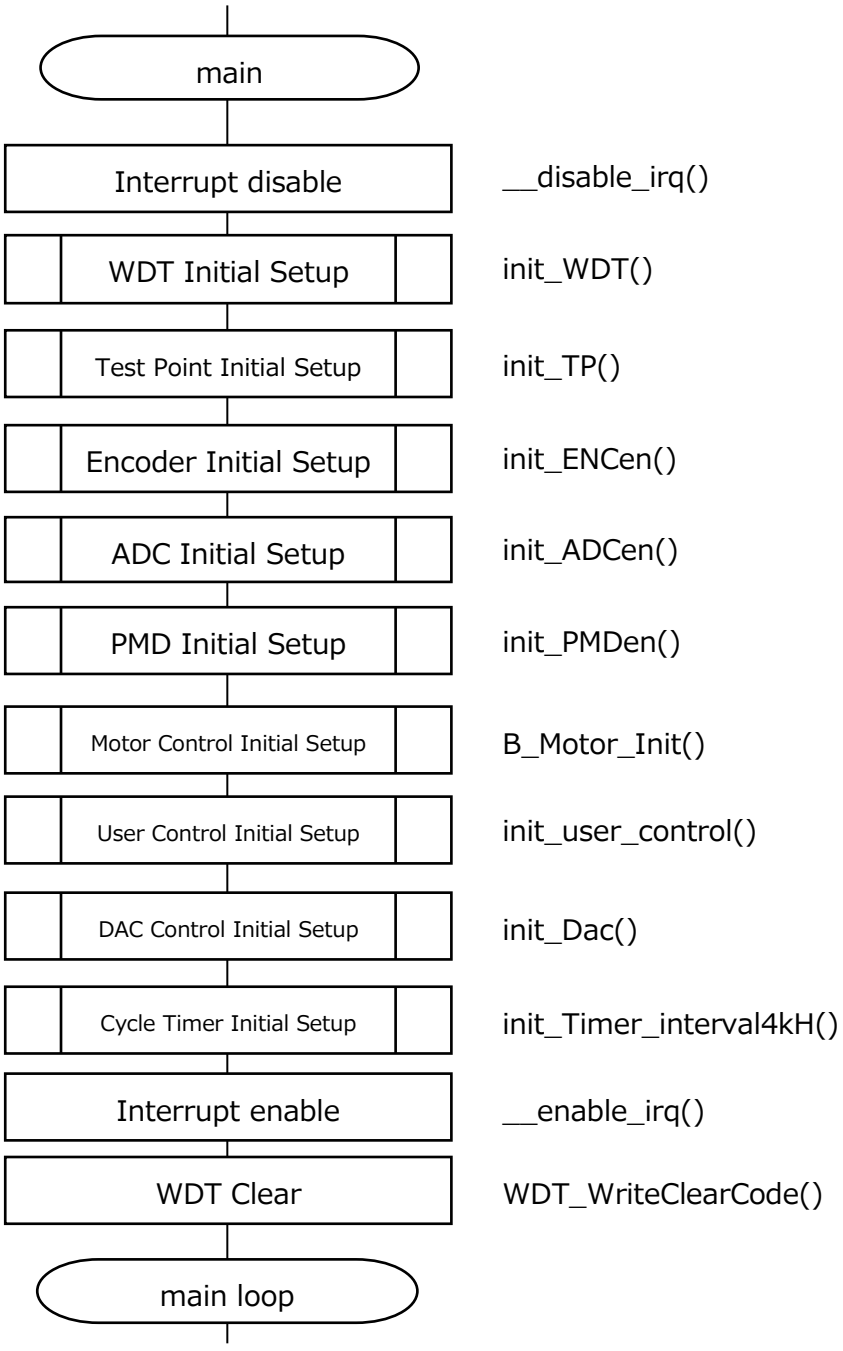


Figure 7.1 Main Routine (main)

7.2 Main Loop (main_loop)

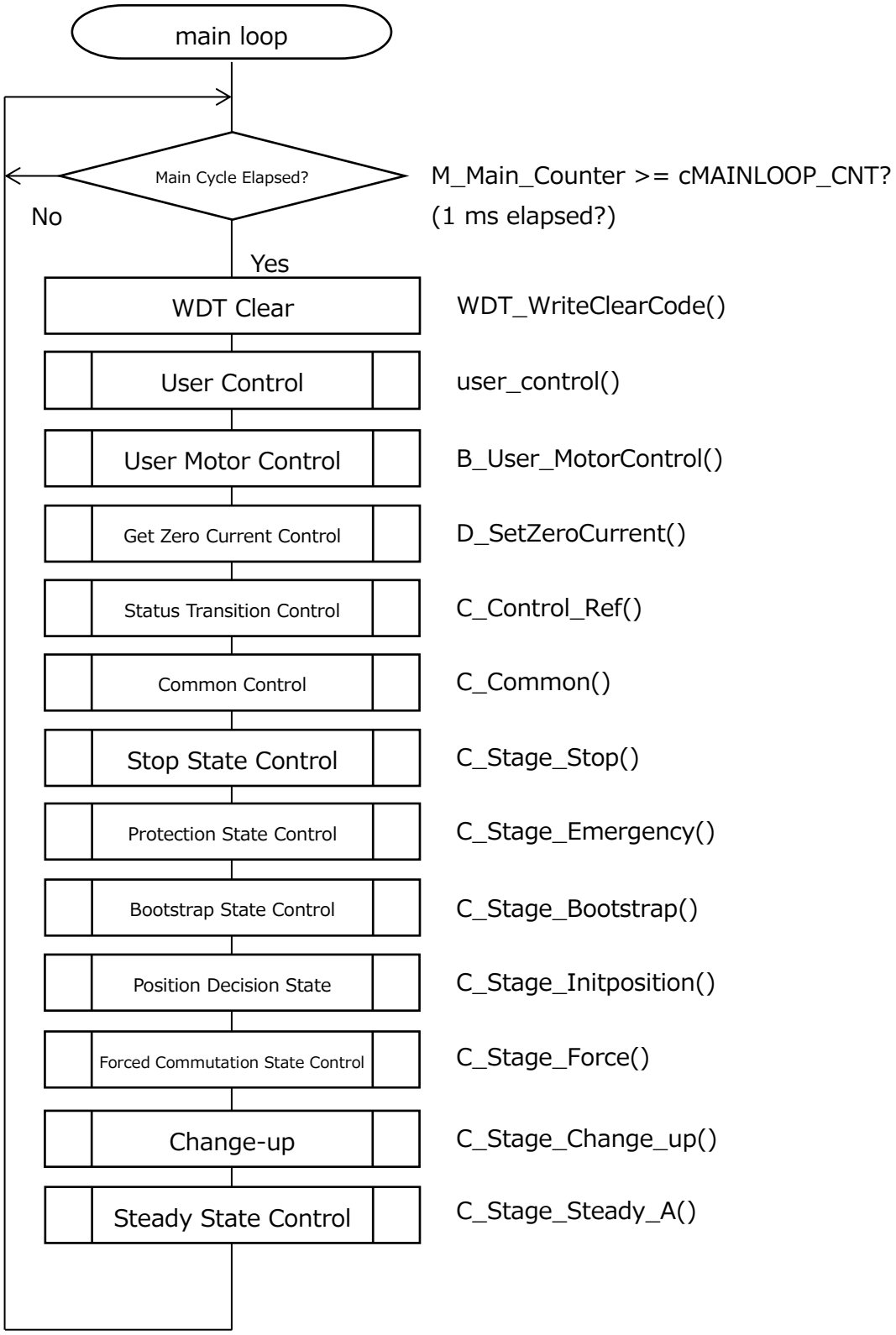
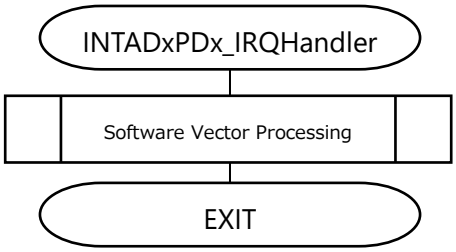


Figure 7.2 Main Loop (main_loop)

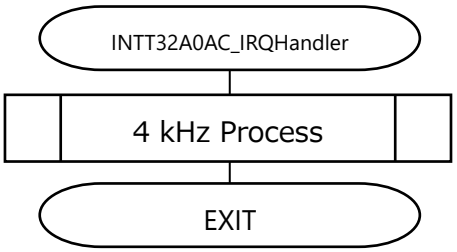
7.3 Interruption

AD Conversion Complete Interrupt (x = A,B)



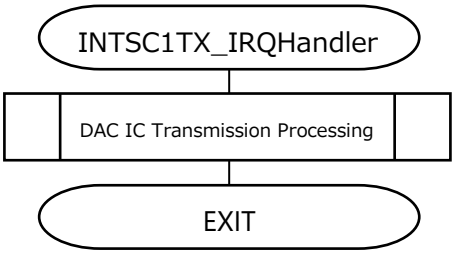
INT_VectorContorl_bySoft ()

Every 4 kHz cycle



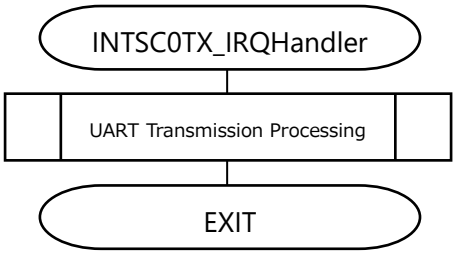
INT_interval_4kHz ()

DAC Communication 24-bit Transmission Complete



DacWriteContinue ()

UART 8-bit Transmission Complete



INT_uart0tx ()

Figure 7.3 Interruption

7.3.1 Software Vector Processing (INT_VectorControl_bySoft)

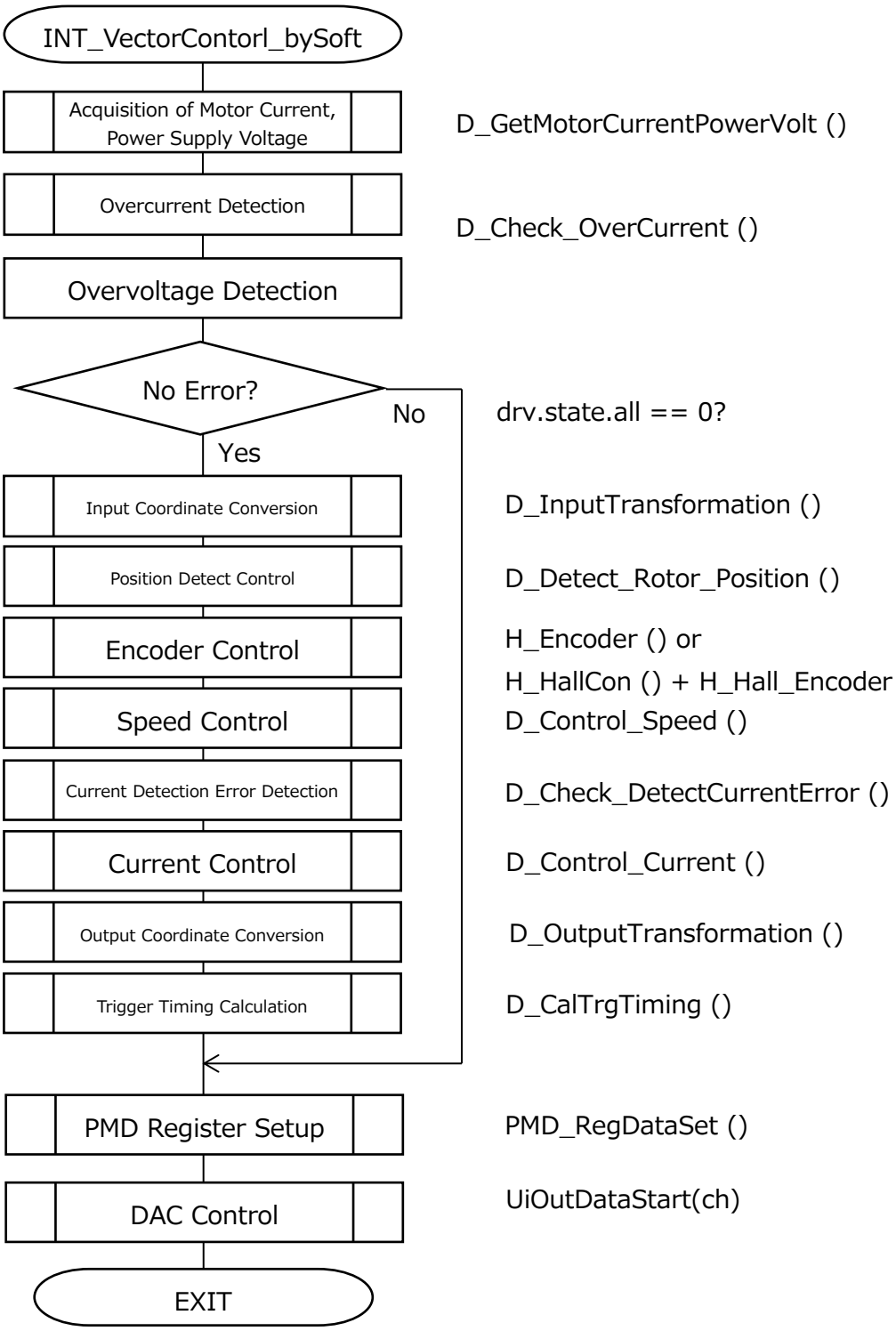


Figure 7.4 Software Vector Processing (INT_VectorControl_bySoft)

8. Status Transition

8.1 Sensor-less

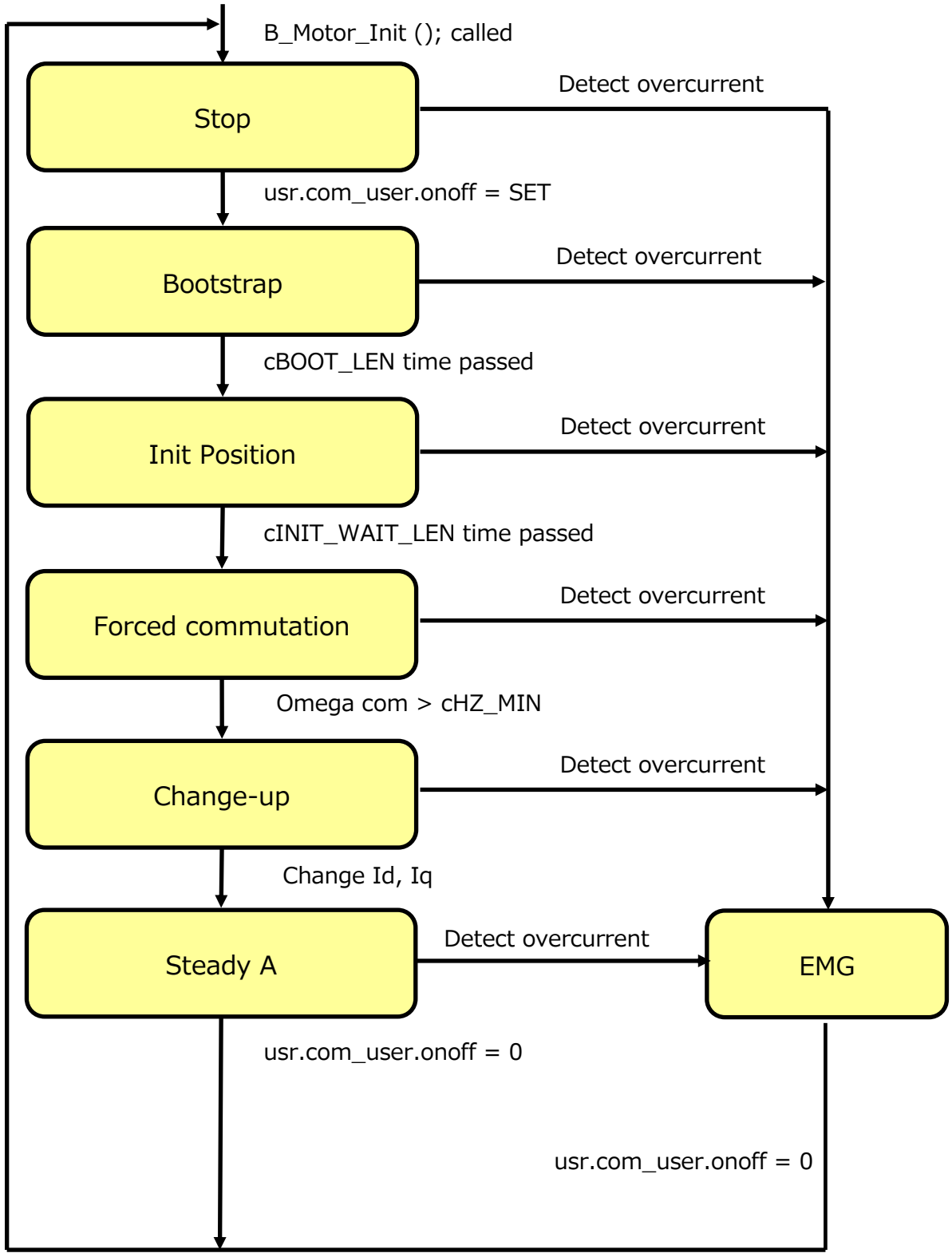


Figure 8.1 Sensor-less

8.2 Encoder

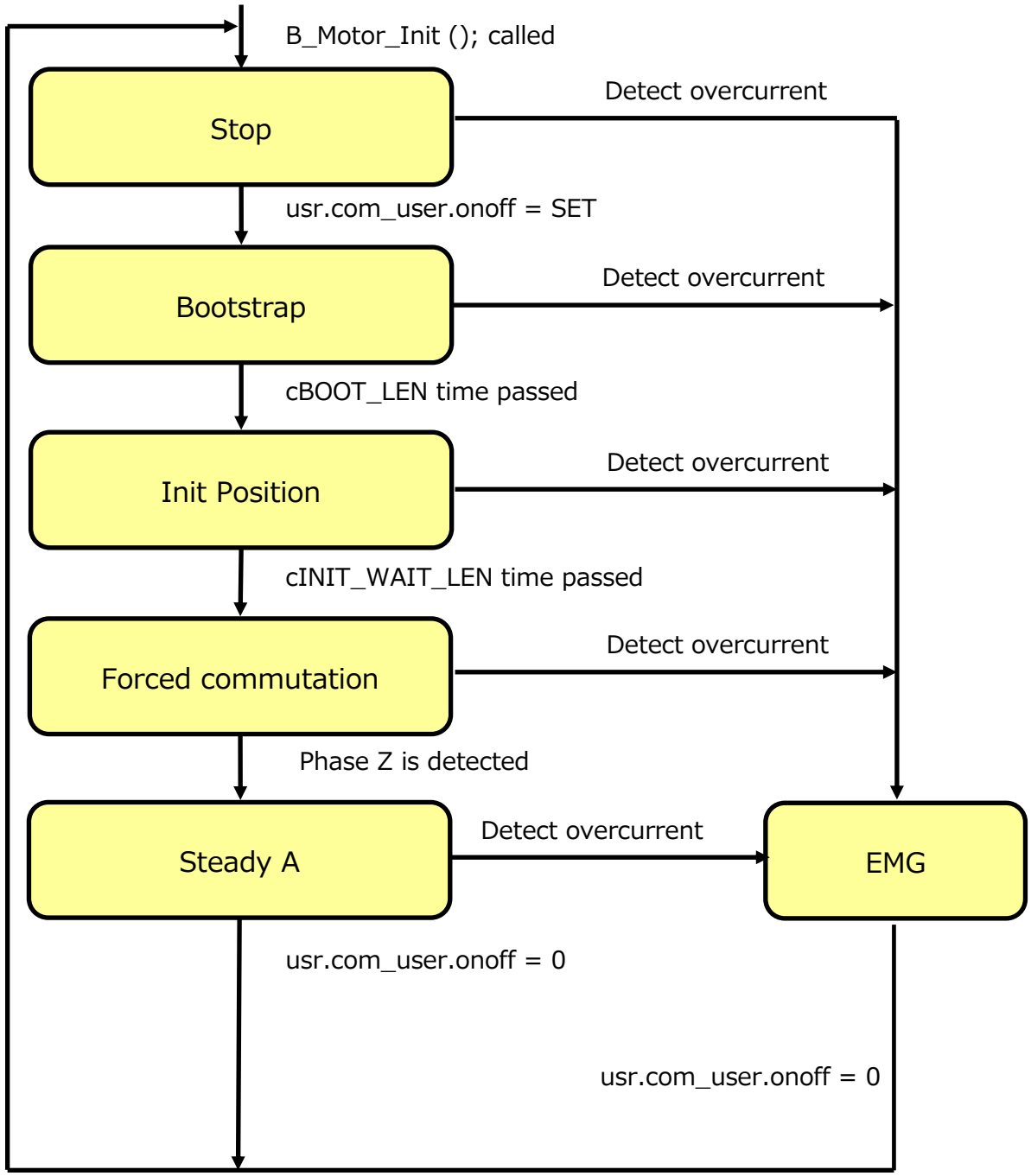


Figure 8.2 Encoder

8.3 Encoder and Hall Sensor

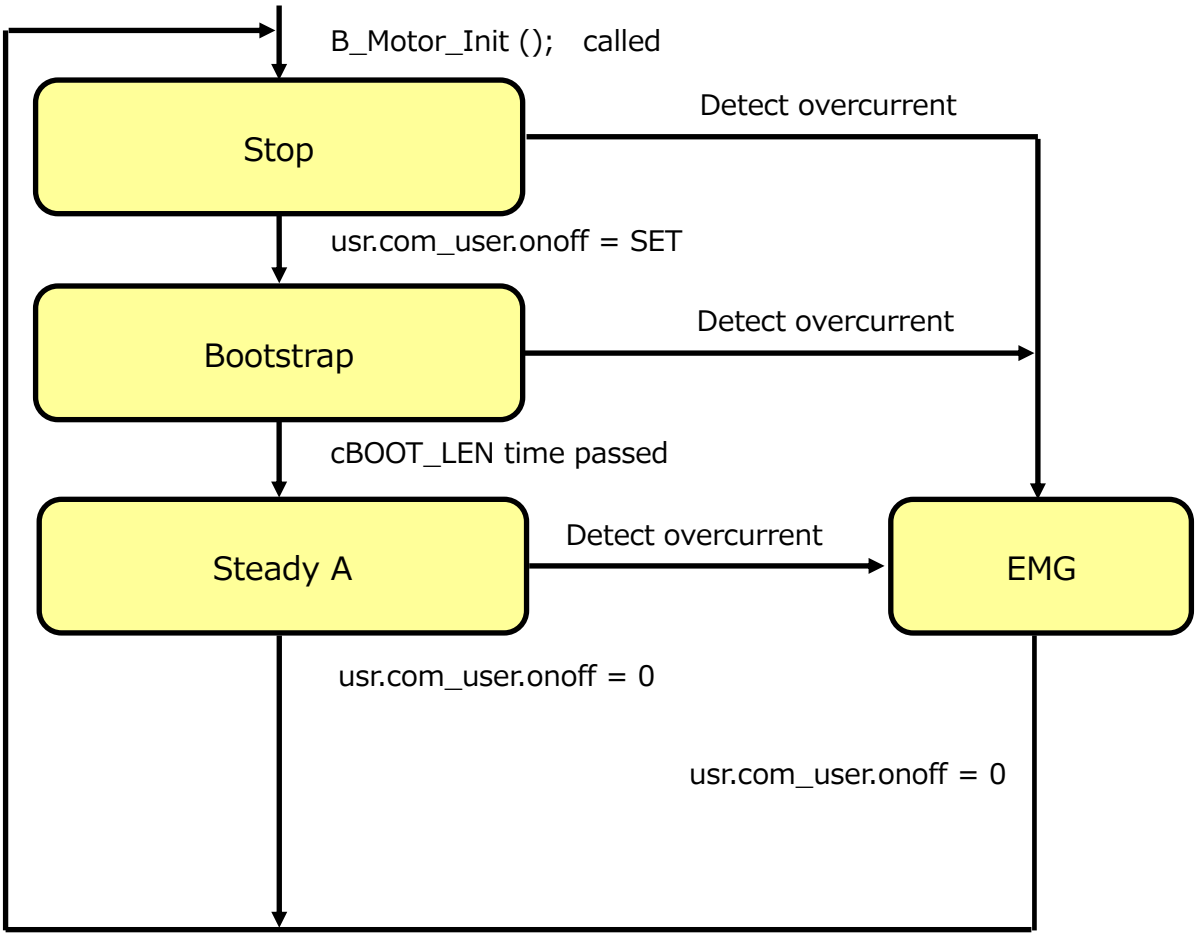


Figure 8.3 Encoder and Hall Sensor

9. User Application

9.1 User Control

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

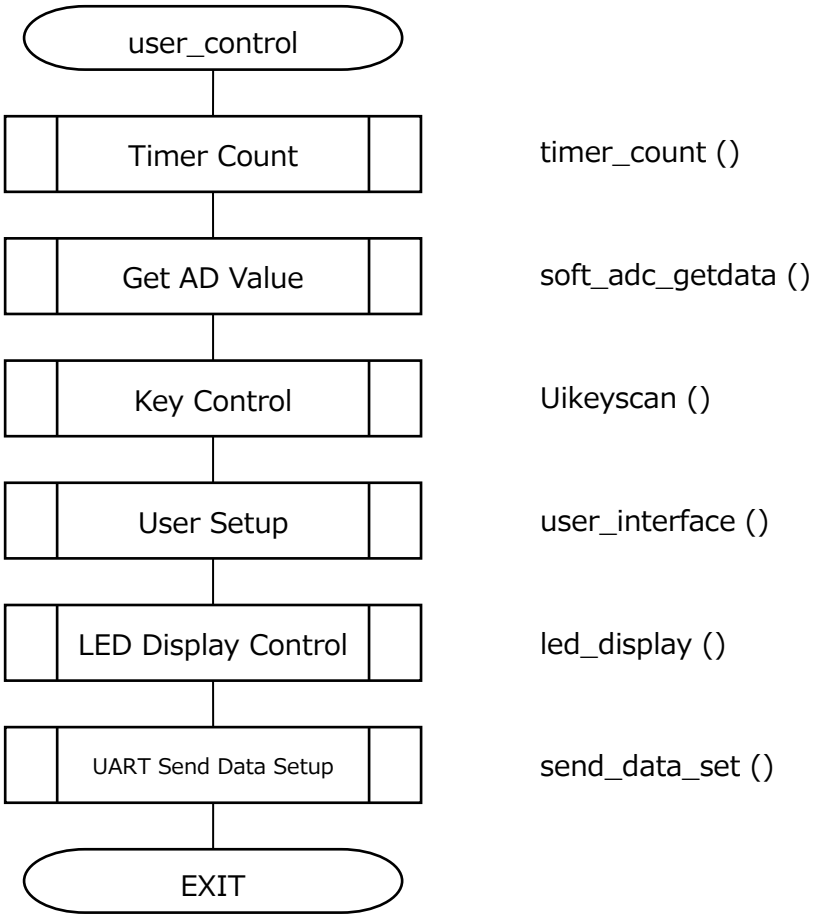


Figure 9.1 User Control

9.1.1 Timer Count (timer_count)

Perform three seconds count processing to judge that the key is held down in KEY processing.

9.1.2 Get AD Value (soft_adc_getdata)

Get AD values (12 bits) of Fake Temperature (VR1) and Motor ch0･ch1Inverter Temperature with the individual conversion setup.
After values are obtained 10 times, for values obtained 8 times excluding the max and min values, the average value for each temperature will be calculated and stored in ad_temp0.avedat, ad_temp1.avedat as the effective value under the following conditions:
- When Fake Temperature Adjustment (__USE_FAKETEMP) is enabled
  The Fake Temperature (VR1) value will be stored in ad_temp0.avedat, ad_temp1.avedat.
- When Fake Temperature Adjustment (__USE_FAKETEMP) is disabled
  The Motor ch0 Inverter Temperature value will be stored in ad_temp0.avedat.
  The Motor ch1 Inverter Temperature value will be stored in ad_temp1.avedat.

9.1.3 Key Control (Uikeyscan)

Process the key scan for S_SW1, S_SW2, S_SW3, S_SW4, USW1, USW2, and USW3.
Each set of the key data will be finalized when an identical value is continuously acquired 20 times, and for S_SW1, S_SW2, S_SW3, and S_SW4, data will be stored in sswdata, and for USW1, USW2, and USW3 data is stored in uswdata.
The time to finalize judgment on USW2 being held down is equal to or more than three seconds, and the time to finalize judgment on USW2 being pressed is less than three seconds.

9.1.4 User Setup (user_interface)

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

1. Motor ch0 Operation
   According to the value of sswdata.bit.sw1, change over Motor_ch0.usr.com_user.spd_ctrl_en.
   When sswdata.bit.sw1 is 1, on: Start (0Hz)
   When sswdata.bit.sw1 is 0, off: Stop
2. Motor ch1 Operation
   According to the value of sswdata.bit.sw2, change over Motor_ch1.usr.com_user.spd_ctrl_en.
   When sswdata.bit.sw2 is 1, on: Start (0Hz)
   When sswdata bit.sw2 is 0, off: Stop
3. Change of operation target in rotation speed/DAC status display
   According to the value of sswdata.bit.sw3, change over disp_ch, dac.motch.
   When sswdata.bit.sw3 is 1, on: Motor ch1
   When sswdata.bit.sw3 is 0, off: Motor ch0

4. Changeover between Up and Down of rotation speed
   According to the value of sswdata.bit.sw4, change over flg.motor.motor_spd_updown.
   When sswdata.bit.sw4 is 1, on: Down
   When sswdata.bit.sw4 is 0, off: Up
5. Rotation speed adjustment
   Each time uswdata.bit.sw1 becomes 1, update the value of target_spdx(x = 0,1).
   0:0 (0Hz) to 10 (100Hz)
   Using the value defined for constant: cSPD_UPDOWN_RESOLUTION, target_spdx(x = 0,1) is updated.
   e.g.)
   target_spd：0→Minimum rotation speed→…→90→100 (maximum rotation speed)

6. DAC Display
   Each time uswdata.bit.sw2 becomes 1, perform DAC display.
   Each time uswdata.bit.sw2 is held down, change over dac.selectchx(x = 0,1).

9.1.5 LED Display Control (led_display)

Control LED4, LED5-according to the EMG status.
See 4.6 User Interface for details.

9.1.6 UART Transmitting Data Setup (send_data_set)

Set up the motor initial settings, current rotation speed, and EMG status monitoring data.
The setups for communication are, 115,200 bps, data 8-bit, stop bit 1 bit, without parity and without flow control.

9.1.7 Temperature Measurement (temp_check)

Obtain the AD values of VR1 TEMP0, and TEMP1.
Computed to one decimal place between temperatures using a linear equation by referring to the values in the table.

Example: When Temp (12bitAD) = 1790
30°C AD Judgment Value = 1858 (table) / 40°C AD Judgment Value = 1507 (table)
1858–1507 = 351 (fixed gradient within 30°C to 40°C)
1858–1790 = 68 (deviation from 30°C)
Interpolation 10°C × 68 / 351 = 1.9°C (1.9373...)
30°C + 1.9°C = 31.9°C

- Temperature range
  The temperature range is from -20˚C to 100˚C, and an error is displayed when the temperature is outside the range.

9.1.8 Status Check Monitor

Monitoring of current rotation 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.

- Details of Initial Display

As shown below, display the initial status of the control channel selected by S_SW3, once after the reset is released.

Control ch = CH or CH
Carrier frequency = xxxxx Hz
Dead time = x.x μs
Gate active = H/H or L/L, etc.
Position Detection = 1-shunt (with/without SPWM) or 3-shunt
VDC Voltage = xx.x V (Note1)
Board Temperature = xx.x °C (Note2)
U,V,W Voltage = x.xx V, x.xx V, x.xx V
DAC Use = Yes/No
Built-in Amplifier Use = Yes/No
Rotation Direction = CW or CCW
Modulation Type = two-phase modulation or three-phase modulation
Rotation Speed Command = Variable Resistor S or Push SW
Note1: Calculated by loading AD(VDC)
Note2: Calculated by loading AD(TEMP0,TEMP1)

- Details Displayed after Initial Display

After a motor speed command is generated, display the current rotation speed of the control ch selected in S_SW3, every second.

e.g.)
40 Hz
60 Hz
80 Hz
:

- Details of Displayed EMG

If a state of EMG arises, display the EMG status of the control ch selected in S_SW3, every second.

For hardware EMG in the STOP stage, the EMG status is not displayed.

EMG_S

Control ch = CH or CH

Carrier frequency = xxxxx Hz

Dead time = x.x μs

Gate active = H/H or L/L, etc.

Position Detection = 1-shunt (with/without SPWM) or 3-shunt

VDC Voltage = xx.x V (Note1)

Board Temperature = xx.x °C (Note2)

U,V,W Voltage = x.xx V, x.xx V, x.xx V

DAC Use = Yes/No

Built-in Amplifier Use = Yes/No

Rotation Direction = CW or CCW

Modulation Type = two-phase modulation or three-phase modulation

Rotation Speed Command = Variable Resistor SW or Push SW

Note1: Calculated by loading AD(VDC)

Note2: Calculated by loading AD(TEMP0,TEMP1)

10. Functions

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

10.1 Control Commands

The control commands are listed as below:

10.1.1 Control Method (usr.com_user)

- Motor start and stop
- Encoder equipped or not
- Modulation Type (two-phase modulation, three-phase modulation)

```
typedef struct {
  uint16_t modul:1; /* PWM Modulation     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.

10.1.2 Control Target Speed

```
q31_u          usr.omega_user; /* [Hz/maxHz] Target omega by user */
```

usr.omega_user is set up as a control target speed in the application.

10.1.3 Starting Current

```
q15_t          Id_st_user; /* [A/maxA] Start d-axis current by user */
q15_t          Iq_st_user; /* [A/maxA] Start q-axis current by user */
```

usr.Id_st_user and usr.Iq_st_user are set up as the start current references in the application.

10.2 Drive Command

Drive Commands are listed as below:

10.2.1 Driving Method (drv.command)

command_t command;

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

10.2.2 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 (unuse)*/
  uint16_t F_vcomm_Edetect:1;      /* Position detect    0=off, 1=on */
  uint16_t F_vcomm_Idetect:1;      /* Current detect     0=off, 1=on (unuse)*/
  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.

Table 10.1 Vector Control Command

cmd / Stage	theta	omega	current	volt	Edetect	Idetect	onoff
StopcForce	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR
Bootstrap	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	SET
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 ω when SET. The reference value is adopted as the speed ω 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 current 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.

7) F_vcomm_onoff

The motor output waveform is turned ON when SET. The motor output waveform is turned OFF when CLEAR.

10.3 Driver Status

10.3.1 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: Overvoltage (Vdc) */
        uint16_t   emg_I:1;           /* 0:normal, 1: Current detection error */
        uint16_t   emg_S:1;           /* 0:normal, 1: Overcurrent (Software) */
        uint16_t   emg_H:1;           /* 0:normal, 1: Overcurrent (Hardware) */
  } flg;
  uint16_t          all;
} state_t;
```

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

10.4 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 */
```

10.4.1 List of Variables

Table 10.2 List of Variables

Type	Code	Description	Q Format	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 Value	Q31	
q31_u	drv.omega	Estimated Speed	Q31	
q15_t	drv.omega_dev	Speed Deviation	Q15	
q31_u	drv.Id_com	d-Axis Current Reference Value	Q31	
q31_u	drv.Iq_com	q-Axis Current Reference Value	Q31	
q15_t	drv.Id_ref	d-Axis Current Standard Value	Q15	
q15_t	drv.Iq_ref	q-Axis Current Standard Value	Q15	

q15_t	drv.Id	d-Axis Current	Q15	
q15_t	drv.Iq	q-Axis Current	Q15	
q31_u	drv.Iq_ref_I	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	DC Voltage Average Value	Q31	
q31_u	drv.Vd	d-Axis Voltage	Q31	
q31_u	drv.Vq	q-Axis Voltage	Q31	
q15_t	drv.Vdq	dq-Axis Voltage	Q15	
q31_u	drv.Vdq_ave	dq-Axis Voltage Average Value	Q31	
q31_t	drv.Vd_out	Output Voltage	Q31	
q15_t	drv.Ed	d-Axis Induced Voltage	Q15	
q15_t	drv.Eq	q-Axis Induced Voltage	Q15	
q31_t	drv.Ed_I	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	---	(Refer to 10.2.1.)
vectorcmd_t	drv.vector_cmd	Vector Control Command	---	(Refer to 10.2.2.)
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.Ia_raw	Phase-a Current (raw data)	Q15	
q15_t	drv.Ib_raw	Phase-b Current (raw data)	Q15	
q15_t	drv.Ic_raw	Phase-c Current (raw data)	Q15	
q15_t	drv.Ia	Phase-a Current (detection error protection)	Q15	
q15_t	drv.Ib	Phase-b Current (detection error protection)	Q15	
q15_t	drv.Ic	Phase-c Current (detection error protection)	Q15	
int32_t	drv.Iao_ave	Phase-a Zero Current Average AD	Q0	
int32_t	drv.Ibo_ave	Phase-b Zero Current Average AD	Q0	
int32_t	drv.Ico_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 (raw data)	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 Reading	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.Id_st_user	Starting Id-Current	Q15	
q15_t	usr.Iq_st_user	Starting Iq-Current	Q15	
uint16_t	usr.lambda_user	Initial Rotor Position	Q0	
command_t	usr.com_user	Control Method	---	(Refer to 10.1.1.)
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	
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	Wait 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	Q12 Support
q31_t	para.pos.ki	Position Estimation Integration Gain	Q15	Q12 Support
int32_t	para.pos.ctrlprd	Position Estimation Control Cycle	Q16	
q31_t	para.spd.kp	Speed Control Proportional Gain	Q15	Q12 Support
q31_t	para.spd.ki	Speed Control Integration Gain	Q15	Q12 Support
uint8_t	para.spd.pi_prd	(Not Used)	Q0	
q31_t	para.crt.dkp	d-Axis Current Control Proportional Gain	Q15	Q12 Support
q31_t	para.crt.dki	d-Axis Current Control Integration Gain	Q15	Q12 Support

q31_t	para.crt.qkp	q-Axis Current Control Proportional Gain	Q15	Q12 Support
q31_t	para.crt.qki	q-Axis Current Control Integration Gain	Q15	Q12 Support
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 for Change-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

Table 10.3 Vector Control Using Software

Type	Code	Description	Q Format	Remarks
uint32_t*	drv.ADxREG0	AD Conversion Result Register Address	---	
uint32_t*	drv.ADxREG1	AD Conversion Result Register Address	---	
uint32_t*	drv.ADxREG2	AD Conversion Result Register Address	---	
uint32_t*	drv.ADxREG3	AD Conversion Result Register Address	---	
q15_t	drv.Vdc_adc	Power Supply Voltage Conversion Result	Q15	
q15_t	drv.Ia_adc	Phase-U Current Conversion Result	Q15	3-shunt only
q15_t	drv.Ib_adc	Phase-V Current Conversion Result	Q15	3-shunt only
q15_t	drv.Ic_adc	Phase-W Current Conversion Result	Q15	3-shunt only
q15_t	drv.Idc1_adc	Phase-1st Current Conversion Result	Q15	1-shunt only
q15_t	drv.Idc2_adc	Phase-2nd Current Conversion Result	Q15	1-shunt only
q31_u	drv.Idco_ave	Zero Current Average AD	Q0	1-shunt only
q15_t	drv.Ialpha	α-Axis Current	Q15	
q15_t	drv.Ibeta	β-Axis Current	Q15	
q15_t	drv.Id_dev	d-Axis Current Deviation	Q15	
q15_t	drv.Iq_dev	q-Axis Current Deviation	Q15	
q31_u	drv.Vd_com	d-Axis Voltage Reference Value	Q31	
q31_u	drv.Vq_com	q-Axis Voltage Reference Value	Q31	
q31_u	drv.Vd_I	d-Axis Voltage Integrated Value	Q31	
q31_u	drv.Vq_I	q-Axis Voltage Integrated Value	Q31	
q31_u	drv.Valpha	α-Axis Voltage	Q31	
q31_u	drv.Vbeta	β-Axis Voltage	Q31	
q15_t	drv.DutyU	Phase-U PWM Duty Ratio	Q15	
q15_t	drv.DutyV	Phase-V PWM Duty Ratio	Q15	
q15_t	drv.DutyW	Phase-W PWM Duty Ratio	Q15	
int32_t	drv.AdTrg0	AD Trigger Timing 0 Position	Q15	
int32_t	drv.AdTrg1	AD Trigger Timing 1 Position	Q15	
uint8_t	drv.Sector	Sector	Q0	0 to 5
uint8_t	drv.Sector1	Previous Sector	Q0	0 to 5
int16_t	para.TrgComp	Trigger Timing Compensation	Q15	
trgpos_e	para.TrgPosMd	Current Detection Position Mode	---	0:3-shunt 1:1-shunt (first half) 2:1-shunt (second half)
phcvmd_e	para.PhCvMd	Phase Conversion Mode	---	0: Spatial Vector 1: Inverse Clarke (not implemented)

10.5 Function Details

10.5.1 Encoder Initial Setup (init_ENCen)

10.5.1.1 Syntax

void init_ENCen(void)

Parameter:

No

Returned Value:

No

10.5.1.2 Process

Initial setup of encoder circuit

- Encoder circuit setup
- Port setup

10.5.2 ADC Initial Setup (init_ADCen)

10.5.2.1 Syntax

void init_ADCen(void)

Parameter:

No

Returned Value:

No

10.5.2.2 Process

Initial setup of ADC

- Motor AD Setup (trigger)
- ADC Permission

10.5.3 PMD Initial Setup (init_PMDen)

10.5.3.1 Syntax

void init_PMDen(void)

Parameter:

No

Returned Value:

No

10.5.3.2 Process

Initial Setup of PMD (programmable motor driver)

10.5.4 Motor Control Initial Setup (B_Motor_Init)

10.5.4.1 Syntax

void B_Motor_Init(void)

Parameter:

No

Returned Value:

No

10.5.4.2 Variable

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.Iao_ave	Phase-u Zero Current Average AD	Q0	
	drv.Ibo_ave	Phase-v Zero Current Average AD	Q0	
	drv.Ico_ave	Phase-w Zero Current Average AD	Q0	
	drv.Idco_ave	Zero Current Conversion Result	Q15	1-shunt only
	usr.Iq_st_user	Starting Iq-Current	Q15	
	usr.Id_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.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	Wait 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 for Change-up	Q0	
	para.pos.ki	Position Estimation Integration Gain	Q15	Q12 Support
	para.pos.kp	Position Estimation Proportional Gain	Q15	Q12 Support
	para.pos.ctrlprd	Position Estimation Control Cycle	Q16	
	para.spd.ki	Speed Control Integration Gain	Q15	Q12 Support
	para.spd.kp	Speed Control Proportional Gain	Q15	Q12 Support
	para.current.dki	d-Axis Current Proportional Gain	Q15	Q12 Support
	para.current.dkp	d-Axis Current Integration Gain	Q15	Q12 Support
	para.current.qki	q-Axis Current Proportional Gain	Q15	Q12 Support
	para.current.qkp	q-Axis Current Integration Gain	Q15	Q12 Support
	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.TrgPosMd	Current Detection Position Mode	---	
	para.TrgComp	Trigger Timing Compensation	Q15	

10.5.4.3 Process

Initializing the motor control parameters.

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

10.5.5 DAC Control Initial Setup (init_Dac)

10.5.5.1 Syntax

void init_Dac(TSB_TSPI_TypeDef* const TSPIx)

Parameter:

No

Returned Value:

No

10.5.5.2 Process

Initial Setup of DAC IC Control

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

10.5.6 Cycle Timer Initial Setup (init_Timer_interval4kH)

10.5.6.1 Syntax

void init_Timer_interval4kH(void)

Parameter:

No

Returned Value:

No

10.5.6.2 Process

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

10.5.7 User Control Initial Setup (init_user_control)

10.5.7.1 Syntax

void init_user_control(void)

Parameter:

No

Returned Value:

No

10.5.7.2 Process

Initial setting for command control from outside such as users.

Key Input Processing Initial Setup (init_Uikey)

Analog Voltage Input Processing Initial Setup (init_soft_adc)

LED Initial Setup (init_led)

UART Initial Setup (init_uart)

10.5.8 User Control (user_control)

10.5.8.1 Syntax

void user_control(void)

Parameter:

No

Returned Value:

No

10.5.8.2 Process

Detection of a command from an outside source, such as the user and execution of control according to the command.

- Motor ch0 operation using S_SW1
- Motor ch1 operation using S_SW2
- Change of operation target in rotation speed/DAC status display using S_SW3
- Changeover between Up and Down of rotation speed using S_SW4
- Rotation speed adjustment using USW1
- DAC display using USW2
- Fake Temperature Adjustment using VR1
- Output of settings and rotation speed information to UART
- Displaying the internal status on LED

Note: The motor operation will be stopped immediately after a changeover of SW.

- Acquisition of Analog Input Voltage

Acquisition and averaging of analog voltage values of fake temperature (VR1) and Motor ch0・ch1 inverter temperature(TEMP0/TEMP1).

- Digital Port Input

Processing of key chatter.

10.5.9 User Motor Control (B_User_MotorControl)

10.5.9.1 Syntax

void B_User_MotorControl(void)

Parameter:

No

Returned Value:

No

10.5.9.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	target_spd	Target Speed	---	User Command
	sswdata.sw	S_SW ON/OFF State	---	User Command
	uswdata.sw	USW ON/OFF State	---	User Command
Output	usr.omega_user	Controller Target Speed	Q31	
	usr.com_user.onoff	Motor ON/OFF	---	
	drv.state	Motor Abnormality Status	---	
	usr.lamlxla_user	Initial Rotor Position	Q0	When Hall sensor and encoder control is avarable

10.5.9.3 Process

Processing of motor-related commands from the user.

- Overcurrent (Hardware) Status Check

  Checking of the hardware overcurrent status to upgrade the motor abnormality status.

- Motor ON/OFF Command

  Determination of Motor ON/OFF(usr.com_user.onoff) according to the key status.

- Normalization of Drive Speed, etc.

  Normalization of the target speed.

10.6 Motor Control Function

Motor control processing is realized by the following functions that are called from inside the main loop of the main function. Motor operation is controlled as a state transition between the following stages: Stop, Bootstrap, Positioning, Forced Commutation, Change-up, Steady, Brake by Short-Circuit, and Protection.

In the main stages, transition is made by the motor operation start command in the following order: Positioning (Initpositon), Forced Commutation (Force), Change-up (Change_up), and Steady (Steady_A). The sub-stages have steps starting from Step0 to StepEnd, and when the sub-stage is in StepEnd, the main stages transition and the sub-stages start with Step0. When a motor abnormality is detected, transition is made to Protection Stop Emergency.

10.6.1 Status Transfer Processing Function (C_Control_Ref_Model)

10.6.1.1 Syntax

void C_Control_Ref_Model(vector_t* const _motor) Without ENC_Z state transition
void C_Control_Ref_Model_ENC(vector_t* const _motor) With ENC_Z state transition

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.1.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	usr.com_user.onoff	Control Command	---	
	drv.state.all	Error Status	---	
	drv.Z_detect	Z Pulse Detection	---	Used only for ENC_Z state transition
I/O	stage.main	Main Stage	---	
	stage.sub	Sub Stage	---	
	usr.com_user_1.onoff	Previous Control Command	---	

10.6.1.3 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 substages is executed within the processing function for each stage instead of within the stage transfer processing function.

In the case of being with ENC_Z state transition, when a Z-pulse is detected, the stage transitions from Forced Commutation to Steady.

Note: ENC_Z state transition is made when the encoder is enabled, and 3-shunt is adopted.

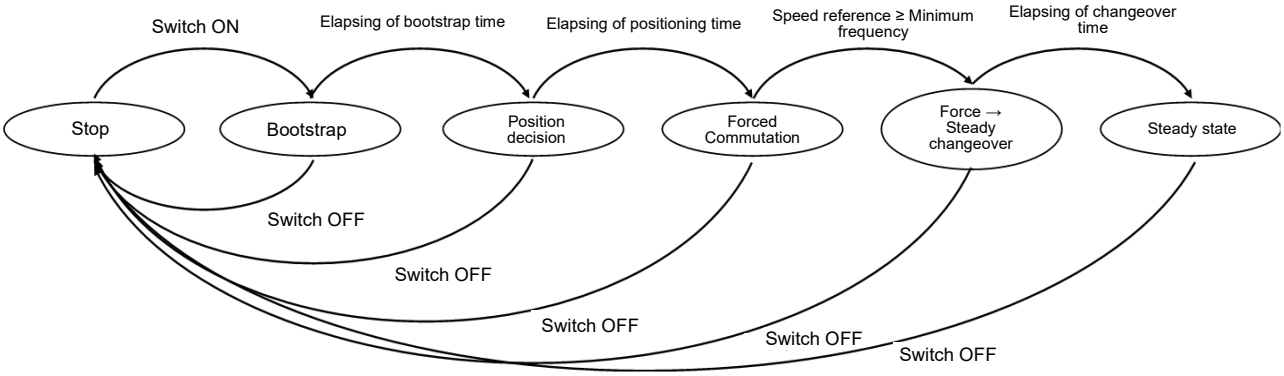


Figure 10.1 State transition chart of Motor control

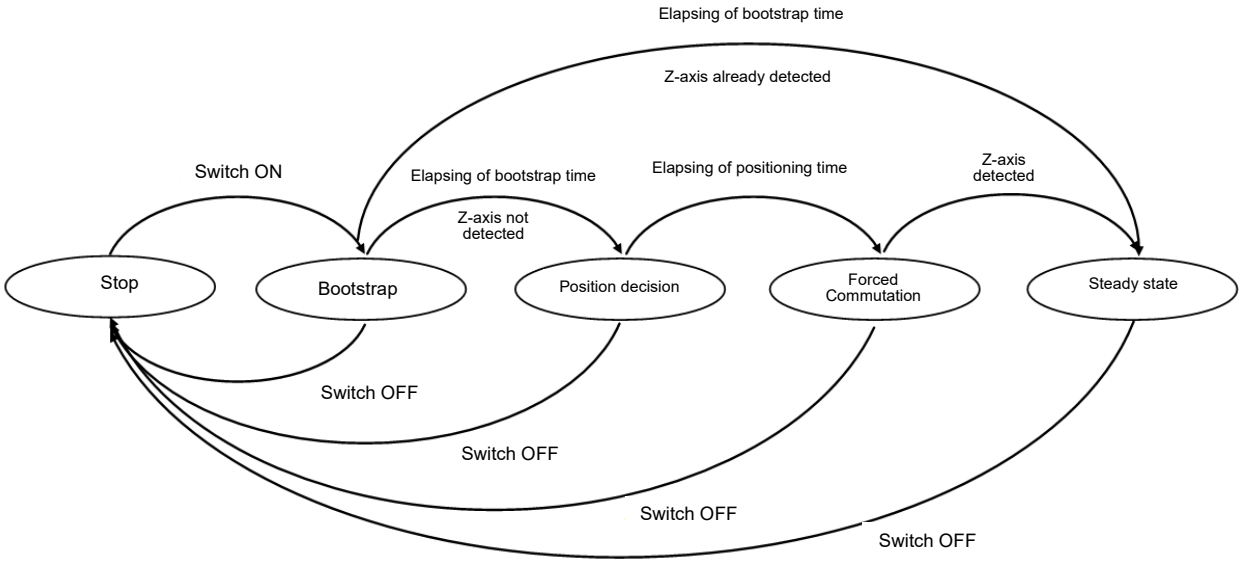


Figure 10.2 State transition chart of Motor control (Encoder Control)

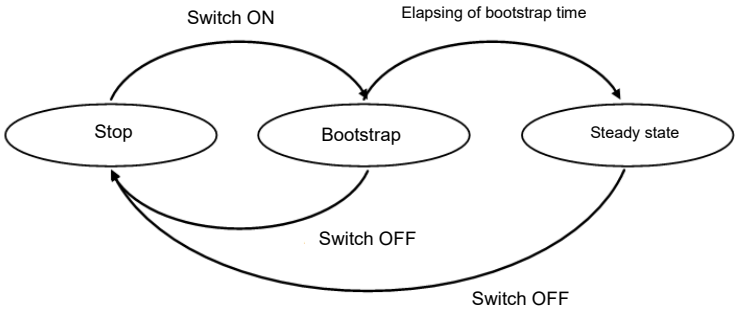


Figure 10.3 Stage transition of motor control (Hall sensor and encoder control)

10.6.2 Motor Control Common Processing Function (C_Common)

10.6.2.1 Syntax

void C_Common(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.2.2 Variable

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
	para.chkpls	Current Detection Protection Duty Width Setup Value	---	
	drv.Vd	d-Axis Voltage	Q32	
	drv.Vq	q-Axis Voltage	Q32	
	drv.Vdc	Power Supply Voltage	Q15	
Output	drv.command.encoder	Encoder Drive Command	---	
	drv.command.modul	Modulation Method Drive Command	---	Two-phase or Three-phase Modulation
	drv.chkpls	Current Detection Protection Duty Width	---	
	drv.Vdq_per	Ratio of dq-Axis Voltage against Power Supply Voltage	---	[%]

10.6.2.3 Process

A common process to various motor control statuses is executed.

Vdq calculation (Cal_Vdq) is performed to compute the ratio of Vdq against Vdc.

10.6.3 Stop Stage Function (C_Stage_Stop)

10.6.3.1 Syntax

void C_Stage_Stop(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.3.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
I/O	stage.sub	Sub Stage	---	
Output	stage.itr	Interruption Stage	---	
	drv.vector_cmd	Vector Control Command	---	(Refer to 10.2.2.)
	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.Vd_com	d-Axis Voltage Reference Value	Q31	Only Vector Control Using Software
	drv.Vq_com	q-Axis Voltage Reference Value	Q31	Only Vector Control Using Software
	drv.Id_com	d-Axis Current Reference Value	Q31	
	drv.Iq_com	q-Axis Current Reference Value	Q31	

10.6.3.3 Process

Stops the motor. (Stops the PWM output)

10.6.4 Bootstrap Stage Function (C_Stage_Bootstrap)

10.6.4.1 Syntax

void C_Stage_Bootstrap(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.4.2 Variable

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	---	
	drv.vector_cmd	Vector Control Command	---	(Refer to 10.2.2.)

10.6.4.3 Process

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

Continues this process for the Bootstrap Time. Determines the charging amount for the capacitor based on the Bootstrap Time

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

a) Initial Status

Initial setup of bootstrap status.

b) Time Lapse Standby Stage

Wait for the lapse of specified bootstrap time and transfers to the positioning stage.

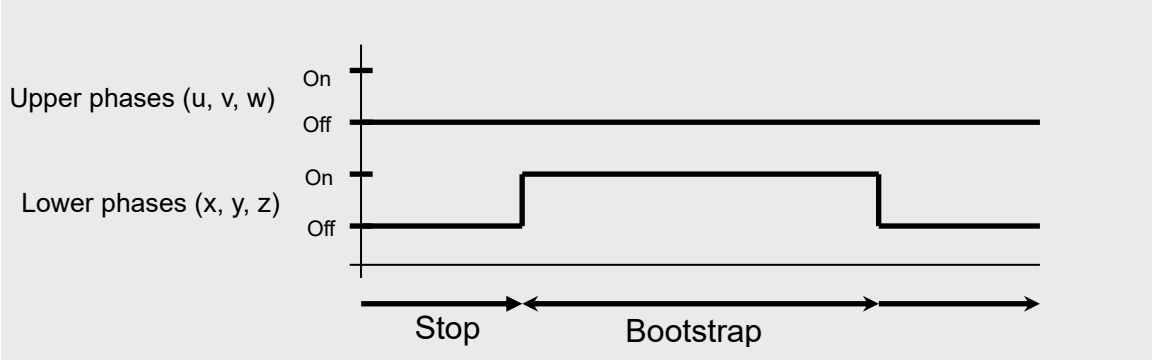


Figure 10.4 Bootstrap Time

10.6.5 Positioning Stage Function (C_Stage_Initposition)

10.6.5.1 Syntax

void C_Stage_Initposition(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.5.2 Variable

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	---	(Refer to 10.2.2.)
	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	

10.6.5.3 Process

Fixes the rotor to the initial position.

While fixing θ to the Initial Position, ω to 0 and Iq to 0, gradually increases Id from 0. Continues this process during the Positioning Time and finally Id reaches the Start Id Current. Determines in advance the increment of Id per unit time based on the Positioning Time and the Start Id Current. After Id 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.

a) Initial Status

Initial setup of positioning stages.

b) Id Increasing Stage

Id will be gradually increased to the setup value.

10.6.6 Forced Commutation Stage Function (C_Stage_Force)

10.6.6.1 Syntax

void C_Stage_Force(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.6.2 Variable

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	---	(Refer to 10.2.2.)
	stage.itr	Interruption Stage	---	
	drv.Iq_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

10.6.6.3 Process

Starts the rotation of rotor. In this stage, a rotation 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, ωcom, will be gradually increased while fixing the Id to Start Id Current and Iq to 0. Acquire θ from ωcom (θ=ωcomt). This process will be continued until ω reaches the Forced Commutation End Speed.

The Drive Target Speed will be increased with a constant increment for approaching Control Target Speed. The Estimated Speed reaches ω.

10.6.7 Forced Steady Changeover Stage Function (C_Stage_Change_up)

10.6.7.1 Syntax

void C_Stage_Change_up(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.7.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
	usr.Id_st_user	Starting Id-Current	Q15	
	usr.Iq_st_user	Starting Iq-Current	Q15	
	usr.omega_user	Controller Target Speed	Q31	
	para.delta_lambda	Current Changeover Phase for Change-up	Q0	
	para.sp_ud_lim_f	Drive Speed Increase/Decrease Limit	Q31	
	para.time.go_up	Wait 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	---	(Refer to 10.2.2.)
	drv.Id_com	d-Axis Current Reference Value	Q31	
	drv.Iq_com	q-Axis Current Reference Value	Q31	

10.6.7.3 Process

Decreases Id to 0 and increases Iq to Start Iq Current and controls the orientation of magnetic field so that the magnetic field and the rotor form a right angle.

In other words, generates the torque component.

Acquire ω and θ through the position estimation computation. The Driving Speed Reference Value will be increased with a constant increment for approaching Control Target Speed However, the Estimated Speed will not be used for control because the speed control is not conducted in this stage.

The forced steady changeover stage will be controlled by separating it into the following substages.

a) Initial Status

Initial setup of forced steady changeover stages.

b) 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.

c) Time Lapse Standby Stage

Waits for the lapse of the specified forced steady changeover time and transfers to the steady stage.

10.6.8 Steady Stage Function (C_Stage_Steady_A)

10.6.8.1 Syntax

void C_Stage_Steady_A(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.8.2 Variable

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	
	usr.lamlxla_user	Initial Rotor Position	Q0	When the hall sensor and encoder control are enabled.
I/O	stage.sub	Sub Stage	---	
	drv.omega_com	Driving Speed Reference Value	Q31	
Output	drv.vector_cmd	Vector Control Command	Q0	(Refer to 10.2.2.)
	stage.itr	Interruption Stage	---	
	drv.ld_com	d-Axis Current Reference Value	Q31	

10.6.8.3 Process

Executes the process of steady stage.

The Driving Speed Reference Value will be increased with a constant increment for approaching Control Target Speed

10.6.9 Protection Stage Function (C_Stage_Emergency)

10.6.9.1 Syntax

void C_Stage_Emergency(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.6.9.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	stage.main	Main Stage	---	
I/O	stage.sub	Sub Stage	---	
Output	drv.vector_cmd	Vector Control Command	---	(Refer to 10.2.2.)
	stage.itr	Interruption Stage	---	

10.6.9.3 Process

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

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

When a software overcurrent is detected, the motor drive outputs 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.

10.7 Motor Drive Function

10.7.1 Explanation of Terms

10.7.1.1 3-phase Modulation

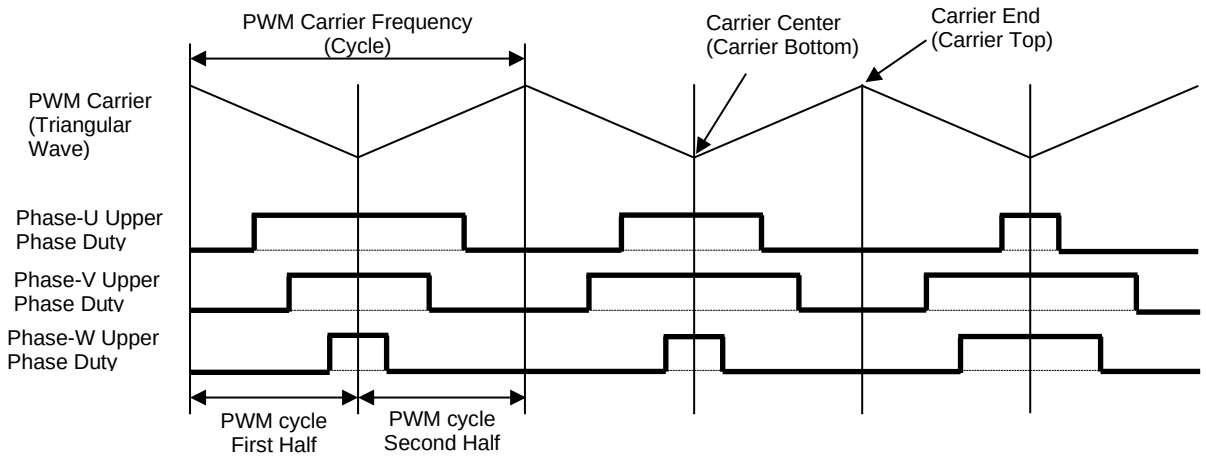


Figure 10.5 **3-phase Modulation**

10.7.1.2 2-phase Modulation

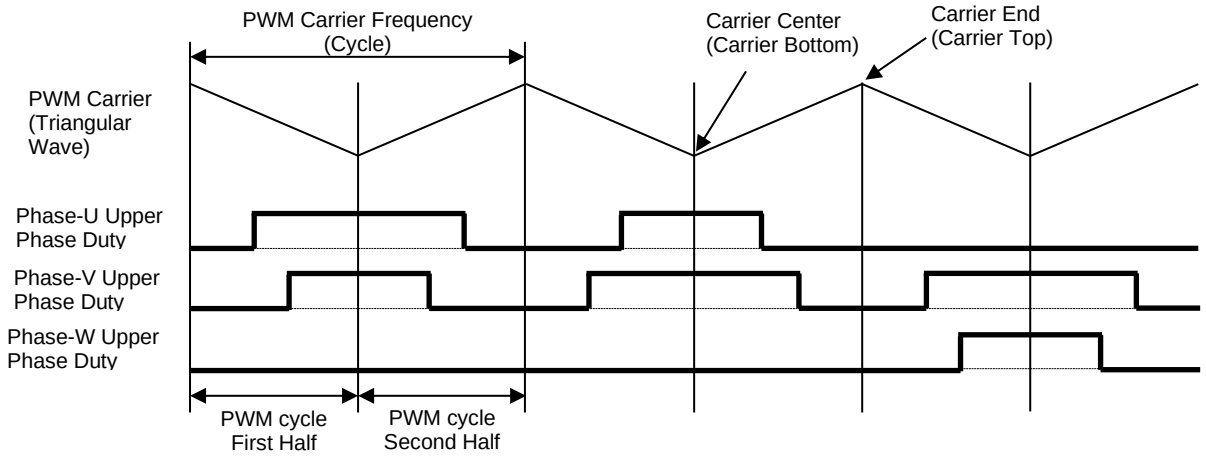


Figure 10.6 **2-phase Modulation**

10.7.2 Acquisition of Motor Current, Power Supply Voltage (D_GetMotorCurrentPowerVolt)

10.7.2.1 Syntax

void D_GetMotorCurrentPowerVolt (vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.7.2.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	drv.Sector1	Previous Sector	Q0	0 to 5
	para.TrgPosMd	Current Detection Position Mode	---	3-shunt 1-shunt first half 1-shunt second half
	drv.ADxREG0	AD Conversion Result Register Address	---	
	drv.ADxREG1	AD Conversion Result Register Address	---	
	drv.ADxREG2	AD Conversion Result Register Address	---	
	drv.ADxREG3	AD Conversion Result Register Address	---	
	drv.Iao_ave	Phase-U Zero Current Average	Q15	3-shunt only
	drv.Ibo_ave	Phase-V Zero Current Average	Q15	3-shunt only
	drv.Ico_ave	Phase-W Zero Current Average	Q15	3-shunt only
	drv.Idco_ave	Zero Current Average	Q15	1-shunt only
	stage.itr	Interruption Stage	---	
	drv.idetect_error	Current Detection Status	---	
I/O	drv.Ia_raw	Phase-U Current (raw data)	Q15	
	drv.Ib_raw	Phase-V Current (raw data)	Q15	
	drv.Ic_raw	Phase-W Current (raw data)	Q15	
Output	drv.Ia	Phase-U Current (detection error protection)	Q15	
	drv.Ib	Phase-V Current (detection error protection)	Q15	
	drv.Ic	Phase-W Current (detection error protection)	Q15	
	drv.Ia_adc	Phase-U Current Conversion Result	Q15	3-shunt only
	drv.Ib_adc	Phase-V Current Conversion Result	Q15	3-shunt only
	drv.Ic_adc	Phase-W Current Conversion Result	Q15	3-shunt only
	drv.Idc1_adc	Phase-1st Current Conversion Result	Q15	1-shunt only
	drv.Idc2_adc	Phase-2nd Current Conversion Result	Q15	1-shunt only
	drv.Vdc	Power Supply Voltage	Q15	
	drv.Vdc_adc	Power Supply Voltage Conversion Result	Q15	

10.7.2.3 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, continues standing by until the completion of conversion. (Never occurs under normal conditions, but as a failsafe 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 is unsigned, the AD conversion result is right shifted by 1 bit and converted to the Q15 format variable.

```
drv.Vdc_adc = (q15_t)((*_motor->drv.ADxREG3 & 0xFFF0) >> 1)
drv.Vdc_adc = drv.Vdc_adc
```

- Motor Current

<Current Detection Mode: 3-shunt>

	Sector					
	0	1	2	3	4	5
Phase-U Current Conversion Result drv.Ia_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.Ib_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.Ic_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.Iao_ave	Phase-U Current Conversion Result drv.Ia_adc
Phase-V Zero Current Conversion Result drv.Ibo_ave	Phase-V Current Conversion Result drv.Ib_adc
Phase-W Zero Current Conversion Result drv.Ico_ave	Phase-W Current Conversion Result drv.Ic_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.

	Sector					
	0	1	2	3	4	5
Phase-U Current drv.Ia	-Ib-Ic	Iao_ave - Ia_adc	Iao_ave- Ia_adc	Iao_ave- Ia_adc	Iao_ave- Ia_adc	-Ib-Ic
Phase-V Current drv.Ib	Ibo_ave - Ib_adc	-Ia-Ic	-Ia-Ic	Ibo_ave - Ib_adc	Ibo_ave- Ib_adc	Ibo_ave- Ib_adc
Phase-W Current drv.Ic	Ico_ave - Ic_adc	Ico_ave - Ic_adc	Ico_ave- Ic_adc	-Ia-Ib	-Ia-Ib	Ico_ave - Ic_adc

<Current Detection Mode: 1-shunt First Half and the Second Half>

Current Detection Mode: During the first half of 1-shunt, the AD conversion result is acquired at the position of the first half of the PWM cycle.

Current Detection Mode: During the second half of 1-shunt, the AD conversion result is acquired at the position of the second half of the PWM cycle.

Current Acquisition Position / Conversion Result	PWM Cycle Second Half	PWM Cycle First Half
Current Conversion Result 1 drv.Idc1_adc	AD Conversion Result 0 ADREG0	AD Conversion Result 1 ADREG1
Current Conversion Result 2 drv.Idc2_adc	AD Conversion Result 1 ADREG1	AD Conversion Result 0 ADREG0

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.Idco_ave	Current Conversion Result 1 drv.Idc1_adc

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.

	Sector					
	0	1	2	3	4	5
Phase-U Current drv.Ia	-(Idco_ave-Idc2_adc)	-Ib-Ic	Idco_ave-Idc1_adc	Idco_ave-Idc1_adc	-Ib-Ic	-(Idco_ave-Idc2_adc)
Phase-V Current drv.Ib	-Ia-Ic	-(Idco_ave-Idc2_adc)	-(Idco_ave-Idc2_adc)	-Ia-Ic	Idco_ave-Idc1_adc	Idco_ave-Idc1_adc
Phase-W Current drv.Ic	Idco_ave-Idc1_adc	Idco_ave-Idc1_adc	-Ia-Ib	-(Idco_ave-Idc2_adc)	-(Idco_ave-Idc2_adc)	-Ia-Ib

10.7.3 Input Coordinate Conversion (D_InputTransformation)

10.7.3.1 Syntax

void D_InputTransformation(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.7.3.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	drv.vector_cmd.F_vcomm_Idetect	Current Detection Command	---	
	drv.Ia	Phase-U Current	Q15	
	drv.Ib	Phase-V Current	Q15	
	(drv.Ic)	Phase-W Current	Q15	
	drv.theta	Rotor Position	Q0	[deg/360]
	para.pos.ctrlprd	Vector Control Cycle	Q0	
	drv.omega	Estimated Speed	Q15	
	drv.idetect_error	Current Detection Status	---	0:Detection Enabled 1:Detection Disabled
Output	drv.Ialpha	α-Axis Current	Q15	
	drv.Ibeta	β-Axis Current	Q15	
	drv.Id	d-Axis Current	Q15	
	drv.Iq	q-Axis Current	Q15	

10.7.3.3 Process

Three-phase currents (Ia, Ib, Ic) are converted to dq-axis currents.

1. Through 10.7.4 Clarke Transformation, three-phase currents (Ia, Ib, Ic) are converted to two-phase currents (Iα, Iβ).
2. Through 10.7.5 Park Transformation, two-phase currents (Iα,Iβ), which are static coordinates, are converted to dq-axis conversions, which are rotational coordinates.
   For the rotor position used for Park Transformation, since the actual rotor position is advanced in relation to the time it was computed in the position estimation processing, θest predicted based on the speed ω and computing cycle is used. θest = θ ＋ speed ω×Vector Control Cycle

If currents cannot be detected normally, conversion is not made, and Id and Iq are not updated.

10.7.4 Clarke Transformation (E_Clarke)

10.7.4.1 Syntax

```
void E_Clarke( q15_t _iu,
               q15_t _iv,
               q15_t _iw,
               q15_t* _ialpha,
               q15_t* _ibeta)
```

Parameter:

_iu	Phase-U Current
_iv	Phase-V Current
_iw	Phase-W Current
_ialpha	α-Axis Current Storage Address
_ibeta	β-Axis Current Storage Address

Returned Value:

No

10.7.4.2 Process

The three-phase currents (Iu, Iv, Iw) are converted to two-phase (Iα, Iβ).

Iα and Iβ are computed using the following computation formulas:

$$I\alpha = \frac{2}{3} \times (Iu \times \cos 0 + Iv \times \cos 120 + Iw \times \cos 240) = \frac{2}{3} \times \left(Iu - \frac{1}{2}Iv - \frac{1}{2}Iw\right)$$

$$I\beta = \frac{2}{3} \times (Iu \times \sin 0 + Iv \times \sin 120 + Iw \times \sin 240) = \frac{2}{3} \times \left(\frac{\sqrt{3}}{2}Iv - \frac{\sqrt{3}}{2}Iw\right)$$

*When three-phase currents are converted to two-phase αβ-axis currents, the amplitude becomes times of itself. Factor is used for making the amplitude equal.

Because the total sum of three-phase currents will be 0, when Iu + Iv + Iw = 0, the result is as follows:

$$I\alpha = Iu$$

$$I\beta = (Iu + 2Iv)/\sqrt{3}$$

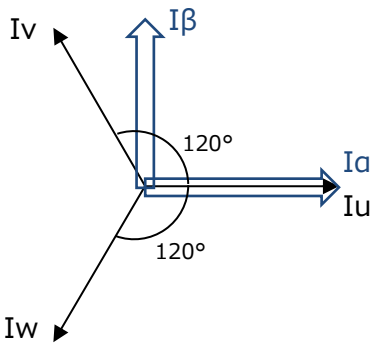


Figure 10.7 Clarke Transformation

10.7.5 Park Transformation (E_Park)

10.7.5.1 Syntax

```
void E_Park( q15_t _ialpha,
             q15_t _ibeta,
             uint16_t _theta,
             q15_t* _id,
             q15_t* _iq)
```

Parameter:

_ialpha	α-Axis Current Iα
_ibeta	β-Axis Current Iβ
_theta	Rotor Position θ: 0≤Position<360°(0 to 0xffff)
_id	d-Axis Current Id Storage Address
_iq	q-Axis Current Iq Storage Address

Returned Value:

No

10.7.5.2 Process

Conversion from the static coordinate of αβ-axes to the dq-axes of rotational coordinate.

Id and Iq are computed using the following computation formulas:

$$Id = I\alpha \times \cos\theta + I\beta \times \sin\theta$$

$$Iq = I\alpha \times (-\sin\theta) + I\beta \times \cos\theta$$

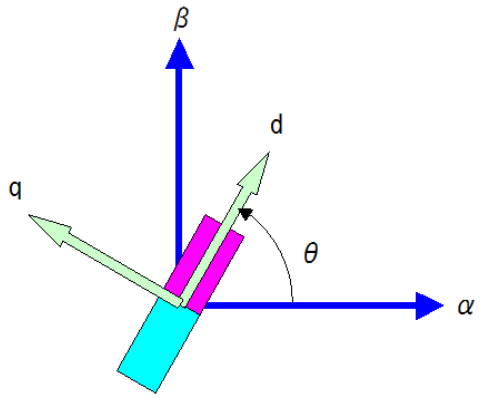


Figure 10.8 Park Transformation

10.7.6 Position Estimation Function (D_Detect_Rotor_Position)

10.7.6.1 Syntax

void D_Detect_Rotor_Position(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.7.6.2 Variable

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	Q12 Support
	para.pos.kp	Position Estimation Proportional Gain	Q15	Q12 Support
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	

10.7.6.3 Process

The PI control is executed by regarding the estimate speed ωest of motor drive signal as the amount of operation and the d-axis induced voltage Ed as the amount of control.

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

The rotor position θ (angle) is acquired by integrating the estimated speed ωest acquired through the PI control.

For the estimated speed omega, when F_vcomm_omega is CLEAR, the driving speed reference value omega_com is used.

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

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

(p = d/dt, Id ≈ constant, then regarded as pId = 0)

Vd : Motor Applied Voltage Id,Iq : Motor Current
ωest : Estimated Angular Speed
R : Resistance Ld,Lq : Inductance

Consequently, the induced voltage Ed of the d-axis is acquired using the following computation formula:

Ed = Vd – R･Id + ωest･Lq･Iq

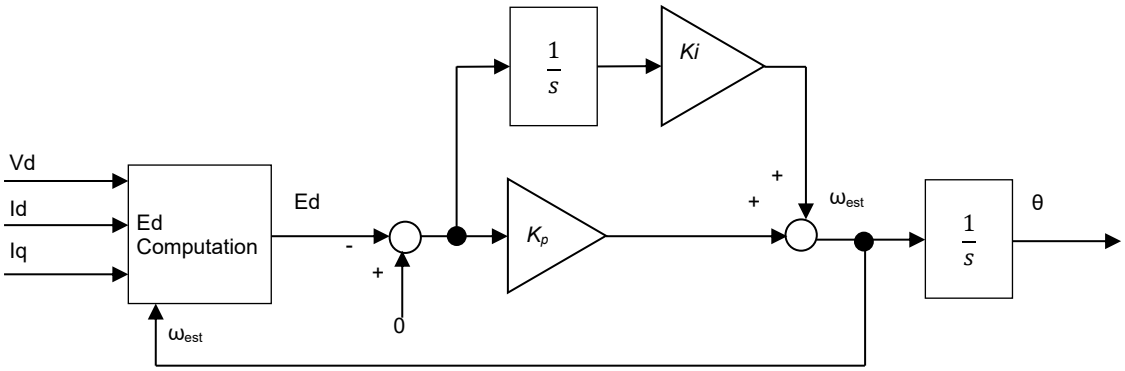


Figure10.9 Position Estimation Block Diagram based on Induced Voltage Ed of the d-Axis

10.7.7 Encoder Control Function (H_Encoder)

10.7.7.1 Syntax

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

Parameter:

_motor Motor Control Structure

ENx ENC Address

Returned Value:

No

10.7.7.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	drv.command.encoder	Encoder Drive Command	---	
	drv.vector_cmd.F_vcomm_omega	Estimated Speed Control Command	---	
	drv.vector_cmd.F_vcomm_theta	Estimated Rotor Position Control Command	---	
	drv.omega_com	Driving Speed Reference Value	Q31	
	drv.theta_com	Electrical Angle Reference Value	Q0	
	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 Reading	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	

10.7.7.3 Process

Reads the number of incremental encoder pulses and processes the calculation of rotor position (angle) θ and speed ω.

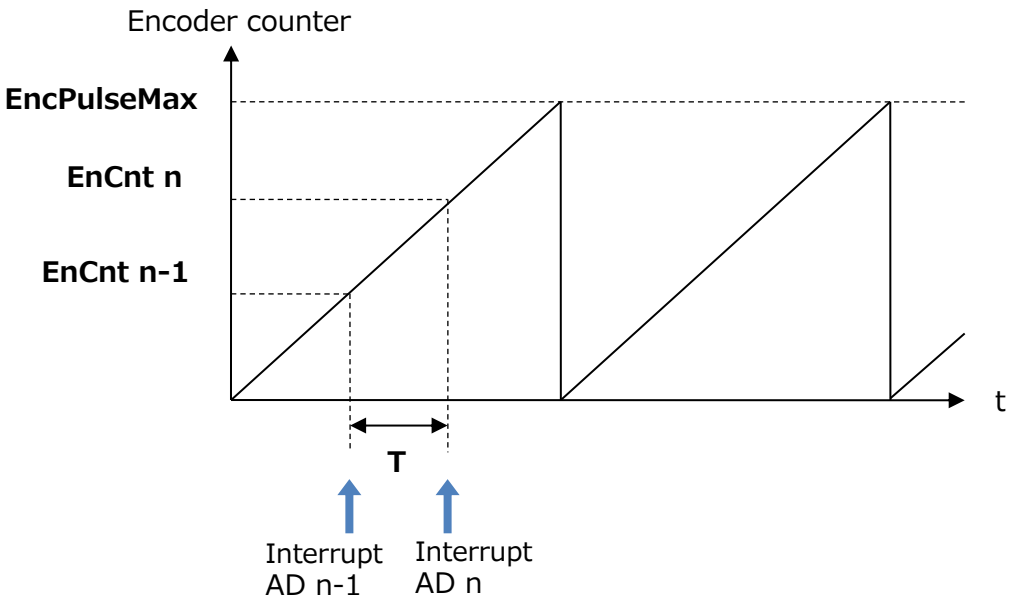


Figure 10.10 Computation of Position and Speed using Encoder Pulse

Reads the number of pulses at every AD conversion completion interrupt and computes the following:

Computes θ using the formula below based on the current encoder pulse counter reading.

$$\theta = EnCnt_n \times (360°/EncPulseMax) \times (Pole/2)$$

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

$$\omega = \{(EnCnt_n - EnCnt_{n-1}) \times (360°/(EncPulsMax/(Pole/2))\}/T$$

EnCntn	Current Encoder Counter Reading
EnCntn-1	Previous Encoder Counter Reading
EncPulseMax	Number of Encoder Pulses [ppr] x Multiplier (4)
Pole	Number of poles
Θ	Angle [deg.]
Ω	Speed [Hz]
T	Speed Computing Cycle [s]

10.7.8 Hall sensor and Encoder control function (H_HallCon H_Hall_Encoder)

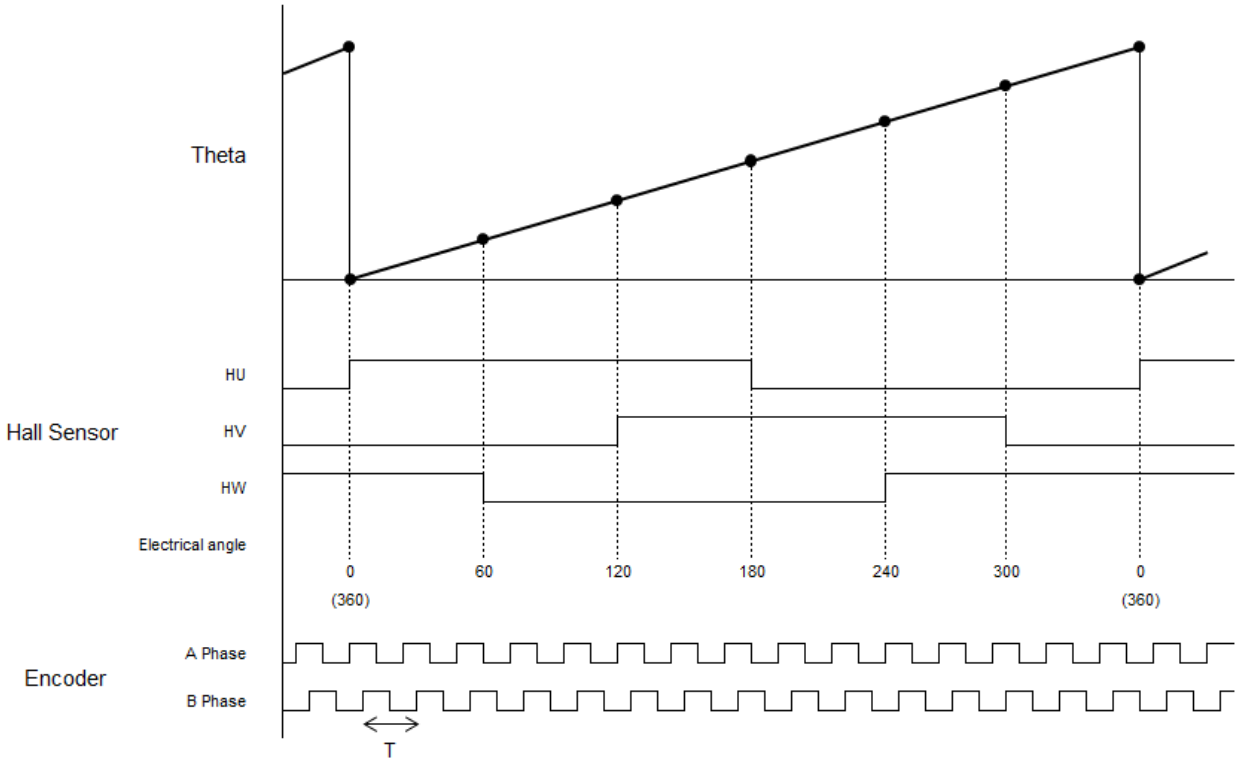


Figure 10.11 Angle detection control before detection encoder Z-signal

1. When the reset of MCU is released, a motor angle is detected by the hall IC. (Every 60 [deg])
2. Until the encoder's Z phase is detected, a motor angle is complemented by the encoder's A and B phases between hall IC signals.
3. After the Z phase of the encoder is detected, a motor angle is detected by the encoder's A, B and Z phases.

10.7.9 Speed Control Function (D_Control_Speed)

10.7.9.1 Syntax

void D_Control_Speed(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.7.9.2 Variable

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	q-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	Q12 Support
	para.spd.kp	Speed Control Proportional Gain	Q15	Q12 Support
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	

10.7.9.3 Process

Executes the PI control by regarding the output frequency ω as the controlled amount and the q-axis current Iq as the operated amount.

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

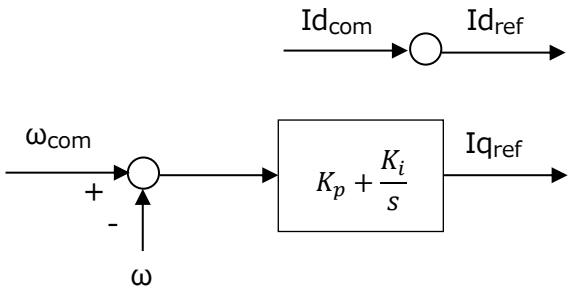


Figure 3.12 Speed Control Block Diagram

10.7.10 Current Control Function (D_Control_Current)

10.7.10.1 Syntax

void D_Control_Current(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.7.10.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	drv.vector_cmd.F_vcomm_volt	Voltage Reference Control Command	---	
	stage.itr	Interruption Stage	---	
	drv.Id_ref	d-Axis Current Standard Value	Q15	
	drv.Iq_ref	q-Axis Current Standard Value	Q15	
	drv.Id	d-Axis Current	Q15	
	drv.Iq	q-Axis Current	Q15	
	drv.Vd_com	d-Axis Voltage Reference Value	Q31	
	drv.Vq_com	q-Axis Voltage Reference Value	Q31	
	drv.Vd_out	Output Voltage		Enabled only during voltage drive
	para.crt.dkp	d-Axis Current Control Proportional Gain	Q15	Q12 Support
	para.crt.dki	d-Axis Current Control Integration Gain	Q15	Q12 Support
	para.crt.qkp	q-Axis Current Control Proportional Gain	Q15	Q12 Support
	para.crt.qki	q-Axis Current Control Integration Gain	Q15	Q12 Support
I/O	drv.Vd_I	d-Axis Voltage Integrated Value	Q31	
	drv.Vq_I	q-Axis Voltage Integrated Value	Q31	
Output	drv.Vd	d-Axis Voltage	Q31	
	drv.Vq	q-Axis Voltage	Q31	

10.7.10.3 Process

Determines the d-axis and q-axis voltages based on the deviation of the current reference and the measured current. When the Voltage Reference Control Command is SET, calculates the d-Axis Voltage and q-Axis Voltage through the following PI control, and when it is CLEAR, adopts the d-Axis Voltage Integrated and q-Axis Voltage Integrated Values as the reference values. Executes the PI control by regarding q-axis current Iq as the controlled amount and the q-axis voltage Vq as the operated amount.

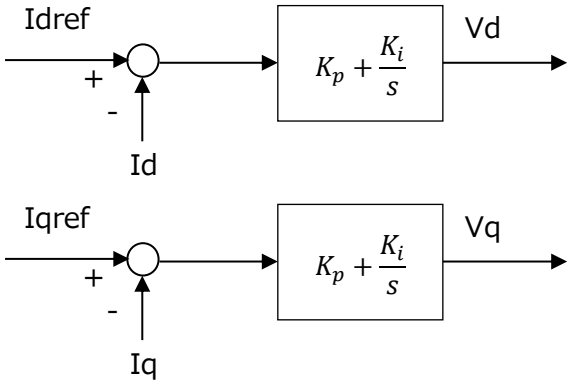


Figure 4 Current Control Block Diagram

10.7.11 Output Coordinate Conversion (D_OutputTransformation)

10.7.11.1 Syntax

void D_OutputTransformation (vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.7.11.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	drv.vector_cmd.F_vcomm_onoff	Motor Output Control Command	---	
	drv.Vd	d-Axis Voltage	Q31	
	drv.Vq	q-Axis Voltage	Q31	
	drv.Vdc	Power Supply Voltage	Q15	
	drv.theta	Rotor Position	Q0	[deg/360]
	drv.PhCvMd	Phase Conversion Mode	---	0: Spatial Vector 1: Inverse Clarke
I/O	drv.Sector	Sector	---	
Output	drv.Valpha	α-Axis Voltage	Q31	
	drv.Vbeta	β-Axis Voltage	Q31	
	drv.DutyU	Phase-U PWM Duty Ratio	Q15 (Unsigned)	0.0 to 1.0
	drv.DutyV	Phase-V PWM Duty Ratio	Q15 (Unsigned)	0.0 to 1.0
	drv.DutyW	Phase-W PWM Duty Ratio	Q15 (Unsigned)	0.0 to 1.0
	drv.Sector1	Previous Sector	---	

10.7.11.3 Process

Converts dq-axis voltages (Vd, Vq) to three-phase PWMDuty ratios (DutyU, DutyV, DutyW).

1. Through 10.7.12 Inverse Park Transformation, converts dq-axis voltages (Vd, Vq) to αβ-axis voltages (Vα, Vβ).
2. Computes the current Sector based on the αβ-axis voltages (Vα, Vβ). Saves the Sector before computation to "Previous Sector", so that it is used for motor current acquisition.
3. Through 10.7.13 Spatial Vector Modulation or Inverse Clarke Transformation, converts αβ-axis voltages (Vα, Vβ) to three-phase Duty ratios (DutyV, DutyW, DutyW).

10.7.12 Inverse Park Transformation (E_InvPark)

10.7.12.1 Syntax

```
void E_InvPark( q31_t _d,
                q31_t _q,
                uint16_t _theta,
                q31_t* _alpha,
                q31_t* _beta)
```

Parameter:

_d	d-Axis
_q	q-Axis
_theta	Rotor Position θ: 0≤Position<360°(0 to 0xffff)
_alpha	α-Axis Storage Address
_beta	β-Axis Storage Address

Returned Value:

No

10.7.12.2 Process

Converts from the dq-axes of rotational coordinate to the αβ-axes of static coordinate.

Vα and Vβ are computed using the following computation formulas:

$$V\alpha = Vd \times \cos\theta - Vq \times \sin\theta$$

$$V\beta = Vd \times \sin\theta + Vq \times \cos\theta$$

10.7.13 Sector Calculation (D_CalSector)

10.7.13.1 Syntax

uint8_t D_CalSector(q31_t _valpha,
q31_t _vbeta)

Parameter:

_valpha α-Axis Voltage

_vbeta β-Axis Voltage

Returned Value:

Sector

10.7.13.2 Process

Computes the sector based on the αβ-axis voltage.

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

Table 10.4 Sector Calculation

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

10.7.14 Inverse Clarke Transformation (E_InvClarke)

Not implemented in this software.

10.7.14.1 Syntax

void E_InvClarke(q31_t _valpha,
q31_t _vbeta,
uint16_t* _p_vu,
uint16_t* _p_vv,
uint16_t* _p_vw)

Parameter:

_valpha	α-Axis Vα
_vbeta	β-Axis Vβ
_p_vu	Phase-U Voltage Vu Storage Address
_p_vv	Phase-V Voltage Vv Storage Address
_p_vw	Phase-W Voltage Vw Storage Address

Returned Value:

No

10.7.14.2 Process

Converts two-phase (Vα, Vβ) to three-phase (Vu, Vv, Vw) voltages.

Vu, Vv, and Vw are computed using the following computation formulas:

$$\mathrm{Vu} = \mathrm{V\alpha} \times \cos 0° + \mathrm{V\beta} \times \sin 0° = \mathrm{V\alpha}$$

$$\mathrm{Vv} = \mathrm{V\alpha} \times \cos 120° + \mathrm{V\beta} \times \sin 120° = -\frac{1}{2}\mathrm{V\alpha} + \frac{\sqrt{3}}{2}V\beta$$

$$\mathrm{Vw} = \mathrm{V\alpha} \times \cos 240° + \mathrm{V\beta} \times \sin 240° = -\frac{1}{2}V\alpha - \frac{\sqrt{3}}{2}V\beta$$

10.7.15 Spatial Vector Modulation (D_SVM)

10.7.15.1 Syntax

```
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:

_valpha	α-Axis Vα
_vbeta	β-Axis Vβ
_vdc	Power Supply Voltage
_sector	Sector
_modul	Modulation
_p_vu	Phase-U Duty Ratio DutyU Storage Address
_p_vv	Phase-V Duty Ratio DutyV Storage Address
_p_vw	Phase-W Duty Ratio DutyW Storage Address

Returned Value:

No

10.7.15.2 Process

Acquires the duty ratio of the three-phase PWM from the two-phase αβ-axis voltages.

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

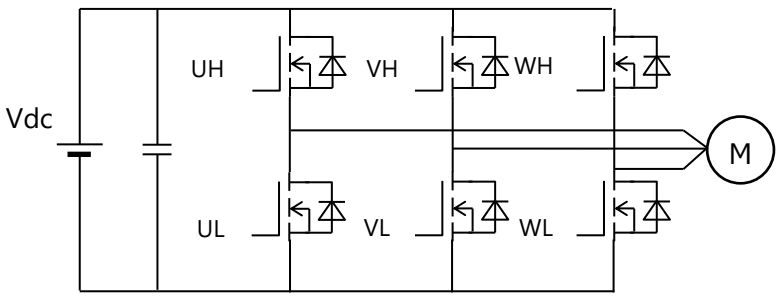
● What is the Spatial Vector Modulation?

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 0.5.

Where these eight patterns are defined as V0 thru V7, and the vectors V0 and V7 are located at the origin due to no line voltage, if graphically expressed, with the remaining six vectors (V1 thru V6) being expressed at every 60 degrees as shown in Figure 6.

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

How to compute PWM Duty when the output voltage vector V exists at Sector 0 will be shown in subsequent pages.



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 5 Inverter Drive Circuit

Table 10.5 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

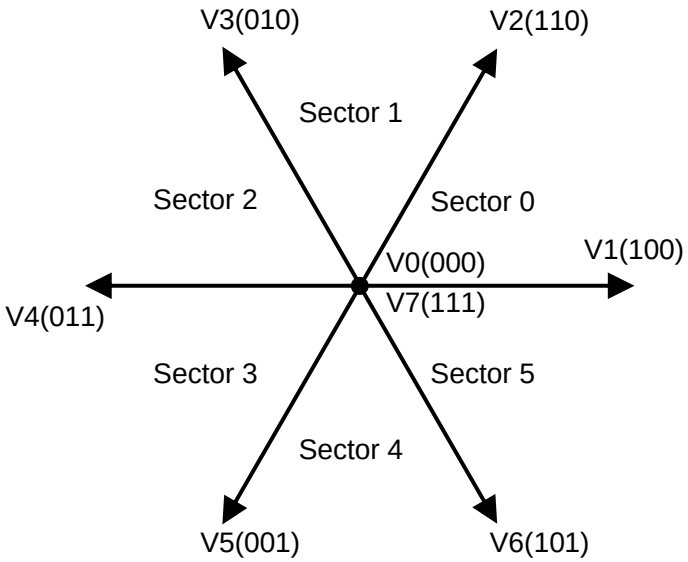


Figure 6 Spatial Vector

- When the output voltage vector V is in Sector 0

For example, in Figure 7, the composite vector V of Vα and Vβ 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, an output voltage vector 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 8.

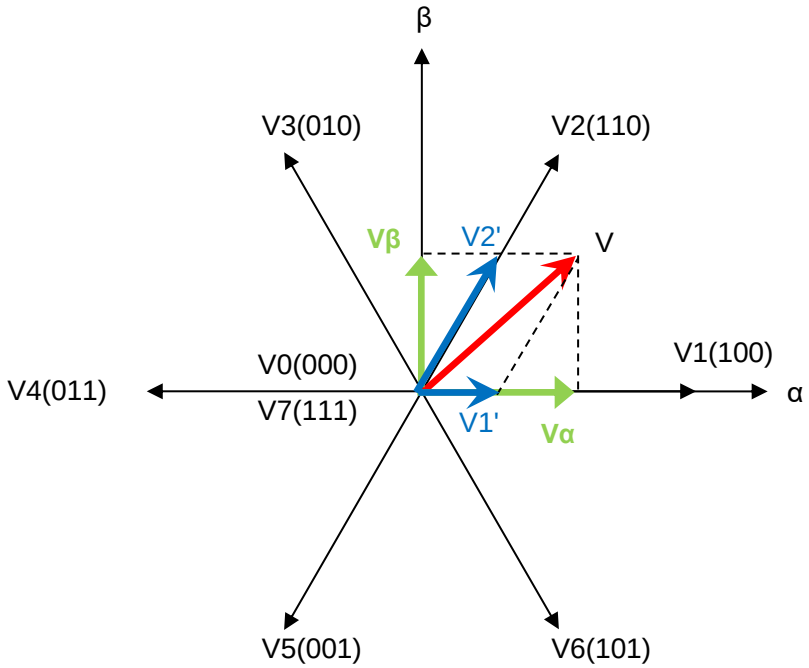


Figure 7 Voltage Vector at Sector 0

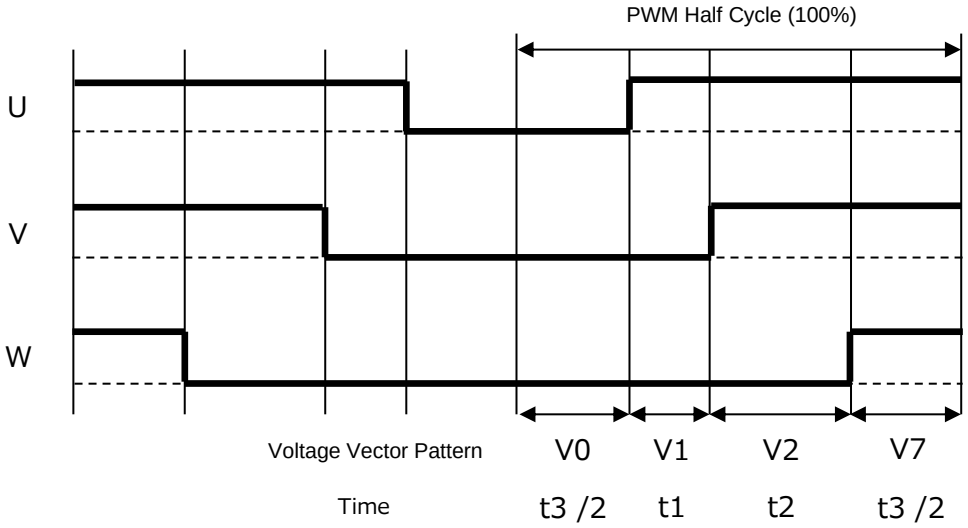


Figure 8 three-phase Modulation PWM Waveform

From Figure10.16 Voltage Vector at Sector 0, Vα and Vβ can be expressed as follows:

$$V\alpha = V1' \times cos0° + V2' \times cos60° = V1' + \frac{V2'}{2}$$

$$V\beta = V1' \times sin0° + V2' \times sin60° = \frac{\sqrt{3}}{2} \times V2'$$

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 Vdc, V1' and V2' are as below:

$$V1' = t1 \times Vdc$$

$$V2' = t2 \times Vdc$$

Consequently, the ratio between t1, t2, and t3 is as below:

$$\mathrm{t1} - k \times \frac{V\alpha - \frac{1}{\sqrt{3}} V\beta}{Vdc}$$

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

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

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

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.

Phase-U Duty = t1 + t2 + (t3/2)

Phase-V Duty = t2 + (t3/2)

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 Table 10.6 Formulas for Duty On Duration per Sector.

Table 10.6 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 10.7.

Table 10.7 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

10.7.16 Trigger Timing Calculation (D_CalTrgTiming)

10.7.16.1 Syntax

void D_CalTrgTiming(vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.7.16.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	drv.DutyU	Phase-U PWM Duty Ratio	Q15	0.0 to 1.0
	drv.DutyV	Phase-V PWM Duty Ratio	Q15	0.0 to 1.0
	drv.DutyW	Phase-W PWM Duty Ratio	Q15	0.0 to 1.0
	drv.command.modul	Modulation	---	two-phase or three-phase Modulation
	drv.TrgPosMd	Current Detection Position Mode	---	3-shunt 1-shunt (first half) 1-shunt (second half)
	para.TrgComp	Trigger Timing Compensation	Q15	-1.0 to 1.0
Output	drv.AdTrg0	AD Trigger Timing 0 (TRG0)	Q15	-1.0 to 1.0
	drv.AdTrg1	AD Trigger Timing 1 (TRG1)	Q15	-1.0 to 1.0

10.7.16.3 Process

Computes the trigger timing to acquire AD conversion values of motor current and power supply voltage Vdc.

The trigger position is different for 3-shunt or 1-shunt.

● For 3-shunt

The basic trigger position is -100% (carrier-end). When a positive value is entered for the trigger timing compensation, the value is compensated backward from the basic trigger position. When a negative value is entered, the position is -100% without compensation.

Modulation	Trigger Timing (TRG) Motor Currents UVW, Power Supply Voltage Vdc
two-phase Modulation	100%
three-phase Modulation	100%

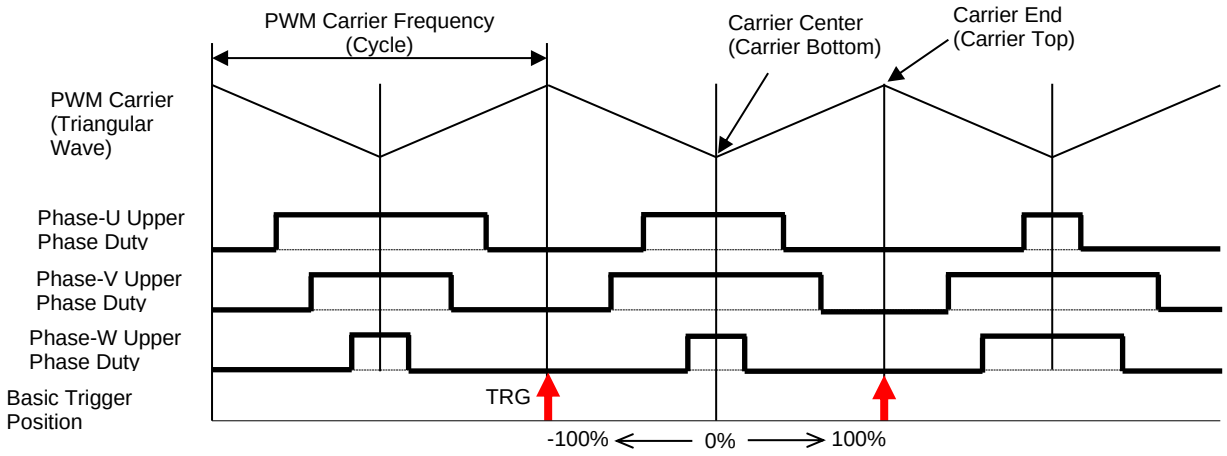


Figure 10.18 Trigger Position for 3-shunt

● For 1-shunt

Computes the basic trigger position from the PWM Duty values in Phases U, V, and W.

When a value other than 0 is entered for the trigger timing compensation, the value is compensated from the computed trigger position.

The Power Supply Voltage Vdc trigger position is fixed to the position of the carrier end. (Set up at initial setup, and not processed by this function.)

	Modulation	Trigger Timing 0 (TRG0) Motor Current	Trigger Timing 1 (TRG1) Motor Current	Trigger Timing 2 (TRG2) Power Supply Voltage Vdc
PWM Cycle Second Half	three-phase Modulation	Duty Median	Duty Max	Fixed to Carrier End
	two-phase Modulation	Duty Median	Duty Max	Fixed to Carrier End
PWM Cycle First Half	three-phase Modulation	-Duty Median	-Duty Min	Fixed to Carrier End
	two-phase Modulation	-Duty Median	Duty Median	Fixed to Carrier End

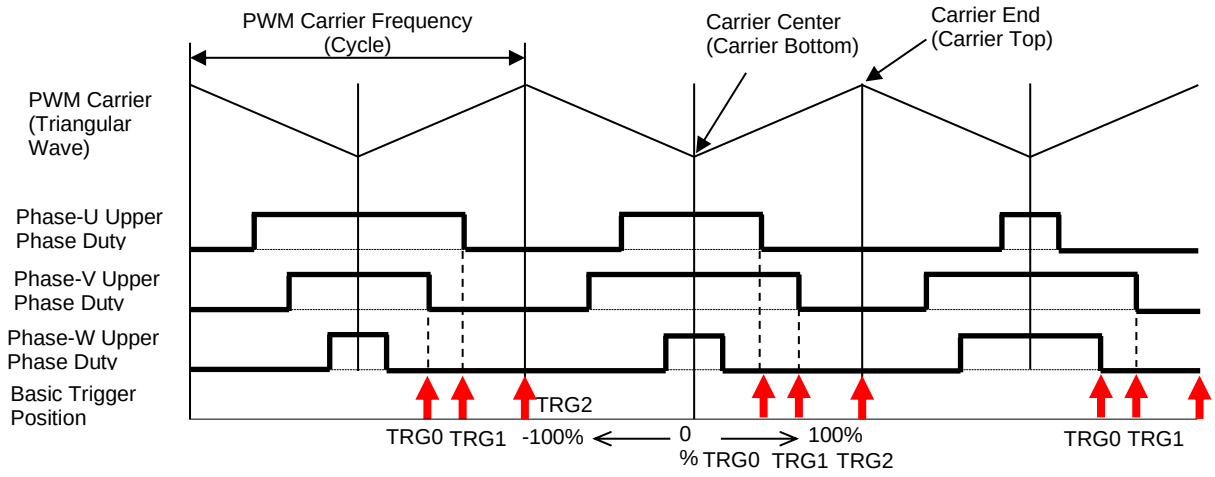


Figure 10.19 Trigger Position for 1-shunt, PWM Cycle Second Half, Three-phase Modulation

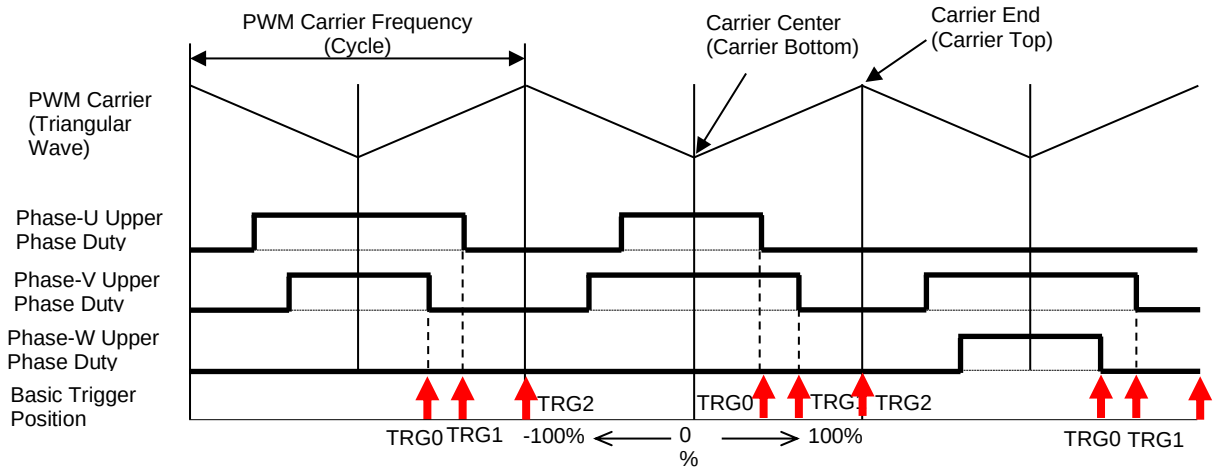


Figure 10.20 Trigger Position for 1-shunt, PWM Cycle Second Half, Three-phase Modulation

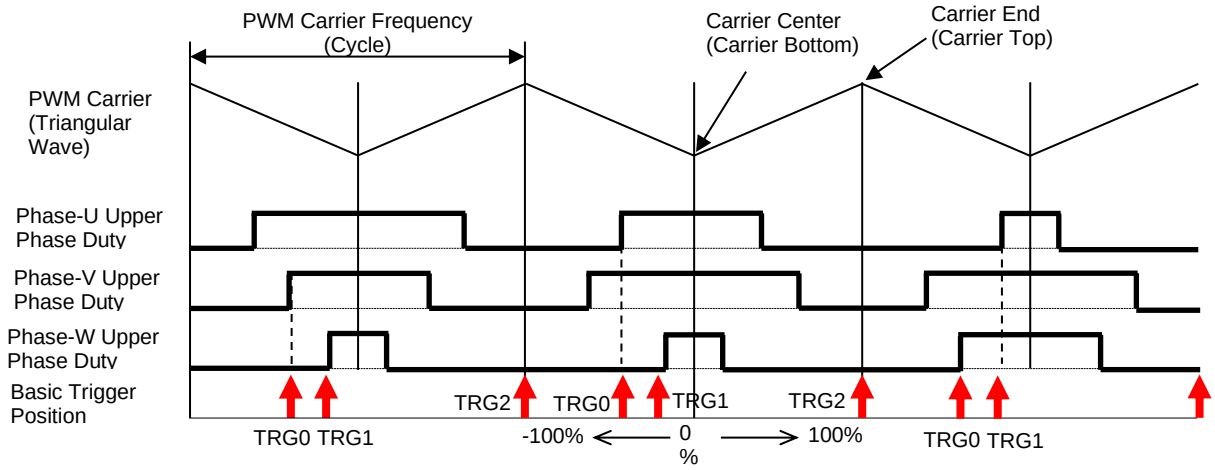


Figure 10.21 Trigger Position for 1-shunt, PWM Cycle First Half, Three-phase Modulation

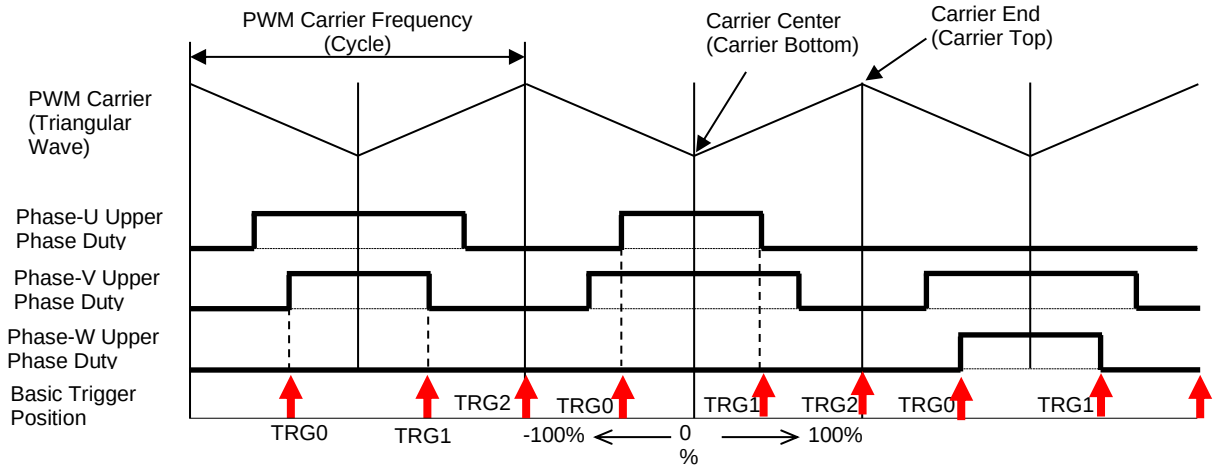


Figure 10.22 Trigger Position for 1-shunt, PWM Cycle First Half, Three-phase Modulation

10.7.17 PWM Register Setup (PMD_RegDataSet)

10.7.17.1 Syntax

```
void PMD_RegDataSet(TSB_PMD_TypeDef* const PMDx,
                    uint16_t duty_u,
                    uint16_t duty_v,
                    uint16_t duty_w,
                    q15_t adtrg0,
                    q15_t adtrg1)
```

Parameter:

PMDx	PMD Register Address
duty_u,	Phase-U PWM Duty Ratio
duty_v,	Phase-V PWM Duty Ratio
duty_w,	Phase-W PWM Duty Ratio
adtrg0,	AD Trigger Timing 0 Position
adtrg1	AD Trigger Timing 1 Position

Returned Value:

No

10.7.17.2 Process

Sets up the three-phase PWM Duty and trigger timing values to the register.

Sets up the entered parameters to the PMD register by executing the following computation.

TRGCMP0 = (adtrg0 + 32768) /2

TRGCMP1 = (adtrg1 + 32768) /2

CMPU = duty_u

CMPV = duty_v

CMPW = duty_w

10.7.18 Current Detection Error Detection Function (D_Check_DetectCurrentError)

10.7.18.1 Syntax

void D_Check_DetectCurrentError (vector_t* const _motor)

Parameter:

_motor Motor Control Structure

Returned Value:

No

10.7.18.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	para.chkpls	Pulse Check Width	---	
	shunt_type	Shunt Type	---	1:1-shunt 3:3-shunt
	drv.command.modul	Modulation Method Drive Command	---	two-phase or three-phase Modulation
	drv.DutyU	Phase-U PWM Duty Ratio	Q15	0.0 to 1.0
	drv.DutyV	Phase-V PWM Duty Ratio	Q15	0.0 to 1.0
	drv.DutyW	Phase-W PWM Duty Ratio	Q15	0.0 to 1.0
Output	drv.idetect_error	Current Detection Status	---	0:Detection Enabled 1:Detection Disabled

10.7.18.3 Process

From the PWM Duty width, detects that it is the time when the current is not detectable.
For 3-shunt, detects that the Duty Median is greater than the setup value.
For 1-shunt, detects that the minimum Duty width and Duty width difference each are smaller than the setup values. Note that the minimum Duty width for 1-shunt and two-phase modulation becomes the middle Duty width among the three-phase Duty widths.
This function checks that the current is not detectable by the relevant duty width, and for current detection processing, uses the previous value without using the detected value, so as to eliminate turbulence in control caused by current detection errors.

- For 3-shunt

Since an AD conversion value for current detection in the PWM maximum phase is not used in computation, when the current detection value in the PWM median phase becomes greater than the setup value, it is judged as not detectable.

- For 1-shunt

When the width of PWM Duty itself, or the width of the PWM Duty difference becomes smaller than the setup value, it is judged as not detectable.

10.7.19 Inverter Output Voltage Calculation Function (Cal_Vdq)

10.7.19.1 Syntax

q15_t Cal_Vdq (q15_t _vd, q15_t _vq)

Parameter:

_vd d-Axis Voltage

_vq q-Axis Voltage

Returned Value:

Inverter Output Voltage (Vdq)

10.7.19.2 Variable

Direction	Code	Description	Q Format	Remarks
Input	_vd	d-Axis Voltage	Q15	0.0 to 1.0
	_vq	q-Axis Voltage	Q15	0.0 to 1.0
Output	---	Inverter Output Voltage (Vdq)	Q15	0.0 to 1.0

10.7.19.3 Process

Computes Inverter Output Voltage (Vdq).

$$Vdq = \sqrt{3} \times \sqrt{\_vd^2 + \_vq^2}$$

11. Explanation of Definitions for Constants

11.1 Motor Driver Setting Parameter Common (D-Para.h)

11.1.1 Motor Control Channel Selection

__CONTROL_MOTOR_CH1 Define when controlling the CH1

__CONTROL_MOTOR_CH0 Define when controlling the CH0

11.1.2 Selecting a DAC Output

__USE_DAC Define when executing the DAC control for outputting the variables for evaluation.

11.1.3 Fake Temperature Adjustment Selection

__USE_FAKETEMP Define when executing Fake Temperature Adjustment

11.1.4 Selection of Unit for Command Speed

__TGTSPD_UNIT Selection of unit for command speed

0: Hz of Electrical angle

1: RPS of Mechanical angle

2: RPM of Mechanical angle

11.1.5 Common 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 values. Use a larger value when the number of encoder pulses is small. However, when the value for the filtering factor becomes larger, the detection value will cause a delay. Set up a value so that delay becomes tolerable.

11.2 Motor Driver Setting Parameter by Motor Channel (D_Para_Chx.h) x=0,1

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

11.2.1 Selection of Control

11.2.1.1 Sensorless / Sensor Selection

__USE_ENCODER	Define when encoder control is performed.
__USE_HALL	Define when hall sensor control is performed.

When both definitions are not defined, sensorless control is performed.

When both definitions are defined, hall sensor and encoder controls.

Only hall sensor control is not supported.

11.2.2 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 [V] x 2^12 (the value of power supply voltage variation 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 [A] x 2^11 (the value of phase current variation with regard to the variation of 1 LSB in the AD conversion).
cSHUNT_TYPE	Setup of Current Acquisition Method (3-shunt or 1-shunt)
	1: 1-shunt/ 3: 3-shunt
	Change the values according to the board design. Set up the current acquisition method.
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. Select Voltage Type Drive in such a case as when the current cannot be acquired at the startup of 1-shunt drive, and others.
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 (3-shunt)
	AINx
	Set up the AD channel for detecting the Phase-U current
cADCH_CURRENT_V	Setup of Phase-V Current Acquisition AD Channel (3-shunt)
	AINx
	Set up the AD channel for detecting the Phase-V current
cADCH_CURRENT_W	Setup of Phase-W Current Acquisition AD Channel (3-shunt)
	AINx
	Set up the AD channel for detecting the Phase-W current
cADCH_CURRENT_IDC	Setup of Current Acquisition AD Channel (1-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 detecting 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 value of power supply voltage Vdc. When voltage higher than this value is detected, the motor will be stopped.
cPWMPRD	Setup of PWM Cycle
	Unit [µs] Resolution 16.67ns@120MHz
	Set up the cycle for the PWM carrier.
cDEADTIME	Setup of Dead Time
	Unit [µs] 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.
cSPEED_ACT	Setup of the command speed
	Unit: [Hz] or [RPS] or [RPM] The unit is as specified in __TGTSPD_UNIT.
	Set up the motor command speed. When SWx is turned ON, the motor drives based on this setup value.
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/Hzs]

	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	Wait Time after Change-up
	Unit [s] Resolution: 1 ms
	Set up the wait 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% from 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).
cMINPLS	Setup of minimum pulse (for 1-shunt)
	Unit [us]
	Set up the PWM pulse width time that makes it impossible to acquire a current for the 1-shunt drive. When the PWM pulse width becomes less than this setup value, control is executed by using the previous value without using the detected current value.

cMAXPLS	Setup of maximum pulse (for thre-shunt)
	Unit [us]
	Set up the PWM pulse width time in the median phase that makes it impossible to acquire a current for the 3-shunt drive. When the PWM pulse width becomes equal to or more than this setup value, control is executed by using the previous value without using the detected current value.
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" For details, refer to 11.2.3.2.
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. Vd = cSPD_COEF × Omega_com Enabled only when the cBOOT_TYPE is "cBoot_v" For details, refer to 11.2.3.2.
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. When the motor estimated speed becomes faster than cHZ_MIN, the stage transitions from Forced Commutation to Steady, and voltage control is changed over to 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"
	Unit [ns]
	Set a value in the range 0 to (cPWMPRD/2).

11.2.3 Relationship of Constant Setups and Waveform

11.2.3.1 Constants for Current Type Starting

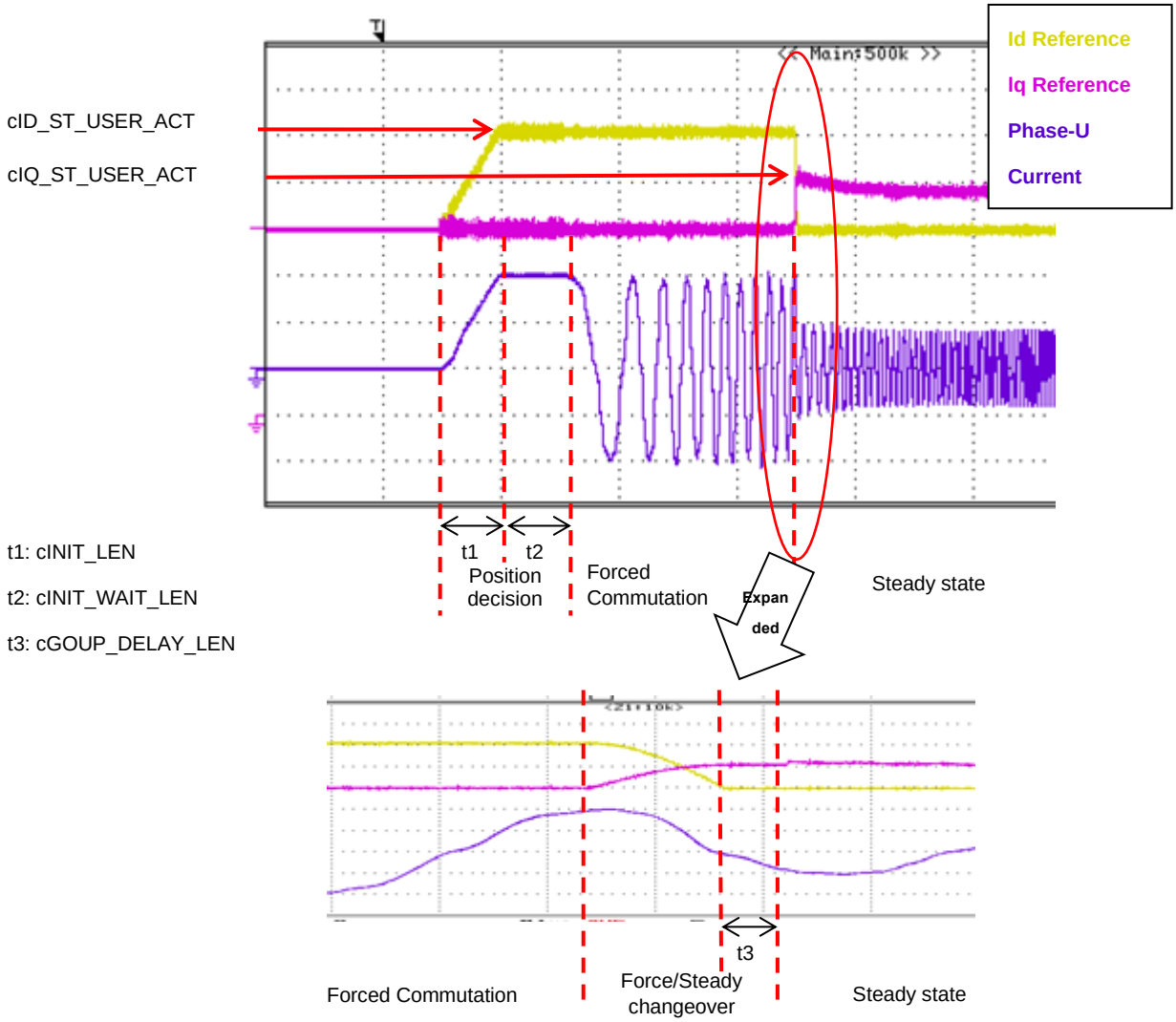


Figure 11.1 Starting Current Waveform (Positioned to electrical angle 0°)

In the Change-up Stage, the values of cID_ST_USER_ACT and cIQ_ST_USER_ACT will be updated. After upgrading, the control will be conducted with a constant reference value of Iq only for the time of cGOUP_DELAY_LEN.

After transferring to the steady state, the Iq reference value will be computed by the PI control.

11.2.3.2 Constants for Voltage Type Starting

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

Observing the current waveform, the cVD_POS value should be adjusted to be the target current value.

The voltage Vd during the forced commutation is calculated by the following formula.

Speed × Constant value cSPD_COEF

The cSPD_COEF value should be adjusted for the current in the forced

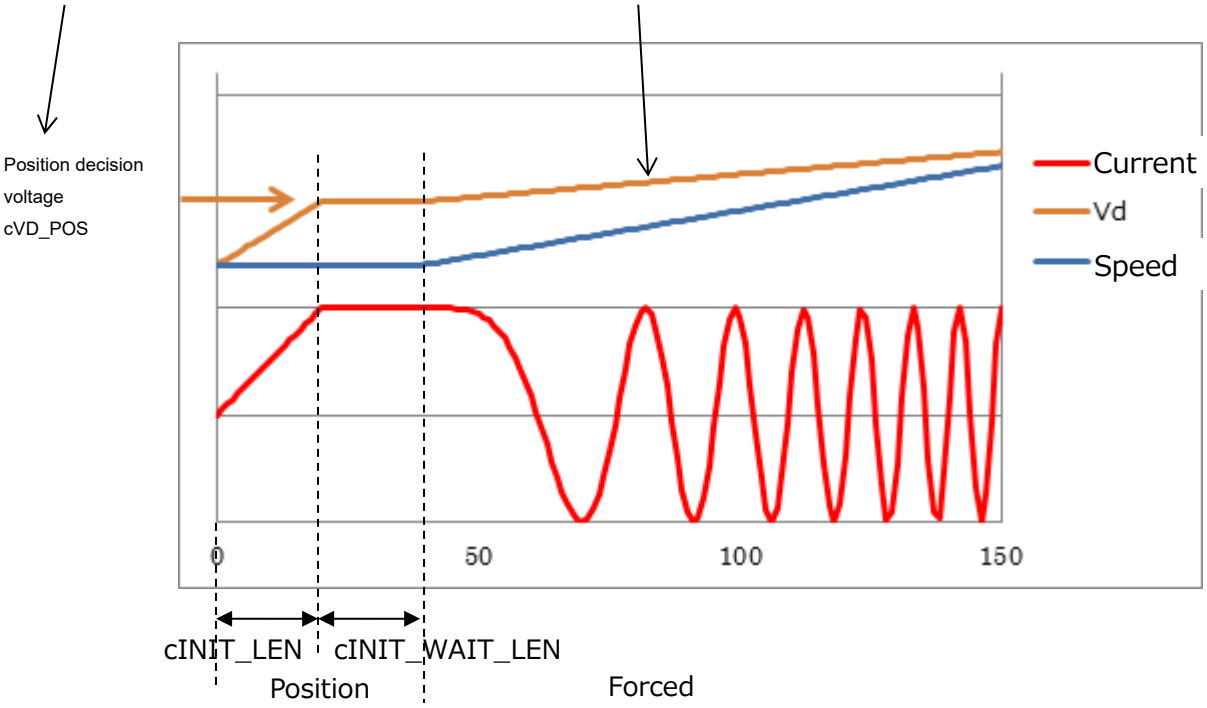


Figure 11.2 Parameter adjustment at Voltage drive

11.3 Constants for User Control

```
/*Initial value*/
#define cINIT_VALUE      (0)
#define cINIT_FF         (0xff)

/* Timer Setting */
#define c3000MS_TIMER            (3000)   /* [ms] (4kHz * 4) * 3000 */

/* Key setting */
#define cS_SW1           (*((__IO uint32_t *)BITBAND_PERI(&TSB_PA->DATA,0))) /* SW1 */
#define cS_SW2           (*((__IO uint32_t *)BITBAND_PERI(&TSB_PA->DATA,1))) /* SW2 */
#define cS_SW3           (*((__IO uint32_t *)BITBAND_PERI(&TSB_PA->DATA,2))) /* SW3 */
#define cS_SW4           (*((__IO uint32_t *)BITBAND_PERI(&TSB_PA->DATA,3))) /* SW4 */
#define cUSW1            (*((__IO uint32_t *)BITBAND_PERI(&TSB_PE->DATA,5))) /* USW1 */
#define cUSW2            (*((__IO uint32_t *)BITBAND_PERI(&TSB_PE->DATA,4))) /* USW2 */
#define cUSW3            (*((__IO uint32_t *)BITBAND_PERI(&TSB_PE->DATA,3)))  /* USW3 */
```

```
#define cKEY_CHATA_CNT      (20)          /* [cnt] chattering counter for KEY SW */
#define cKEY_NO_INPUT       (0x00)        /* USW1 to 3 no input */
#define cKEY_MASK_USW       (0x07)        /* USW1 to 3 mask */
#define cKEY_INPUT_USW1     (0x01)        /* USW1 input */
#define cKEY_INPUT_USW2     (0x02)        /* USW2 input */
#define cKEY_INPUT_USW3     (0x04)        /* USW3 input */

/* Soft ADC Setting */
#define cADUNIT_VR      TSB_ADA           /* VR ad data ADCUnit */
#define cADUNIT_TEMP0       TSB_ADA       /* TEMP0 ad data ADCUnit */
#define cADUNIT_TEMP1       TSB_ADB       /* TEMP1 ad data ADCUnit */

#define cADTRG_USR      ADC_TRG_SINGLE    /* AD Trigger Type */

#define cADCH_VR        ADC_AIN21         /* ADC Channel for VR */
#define cADREG_VR       ADC_REG4          /* Result register Number for VR */
#define cADCH_TEMP0     ADC_AIN17         /* ADC Channel for TEMP0 */
#define cADREG_TEMP0    ADC_REG5          /* Result register Number for TEMP0 */
#define cADCH_TEMP1         ADC_AIN19     /* ADC Channel for TEMP1 */
#define cADREG_TEMP1        ADC_REG6      /* Result register Number for TEMP1 */

#define cADAVECNT (10) /* ADC average count */
#define cONE_ABOVE (1)

/* Speed Control Setting */
#define cSPEED_USER_STOP   (0)
#define cSPEED_USER_MIN_CH0         cHZ_MIN_CH0   /* [Hz] Min Target speed of motor ch0 */
#define cSPEED_USER_MIN_CH1         cHZ_MIN_CH1   /*[Hz] Min Target speed of motor ch1 */
#define cSPEED_USER_MAX     (100)           /* [Hz] Max Target speed of motor ch0,1*/

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

/* LED setting */
#define cLED_EMG_CH0  (*((__IO uint32_t *)BITBAND_PERI(&TSB_PE->DATA,6)))     /* LED4 */
#define cLED_EMG_CH1  (*((__IO uint32_t *)BITBAND_PERI(&TSB_PE->DATA,7)))     /* LED5 */
#define cLED_ON         (1)     /* LED ON level */
#define cLED_OFF        (0)     /* LED OFF level */
```

```
/* State control Setting */
#define cMOTOR_CH0      (0x00)   /* Control channel : CH0 */
#define cMOTOR_CH1      (0x01)   /* Control channel : CH1 */
#define cDISP_CH0       (0x00)   /* Display channel 0 */
#define cDISP_CH1       (0x01)   /* Display channel 1 */

/* UART Setting */
#define cUART_BRD_INIT          ((uint32_t)0x00A6002B)
#define cUART_CLK_INIT          ((uint32_t)0x00000000)
#define cUART_CR0_INIT          ((uint32_t)0x00000001)
#define cUART_CR1_INIT          ((uint32_t)0x00000040)
#define cUART_SEND_WAIT         (1000)
#define cUART_COUNT             (0)
#define c10MHz                  (1000000)
#define cLOW_ACTIVE             (0)
#define cSWRST10                (0x2U)
#define cSWRST01                (0x1U)
#define cSWRSTF                 (0x80)
#define cSendBufSIZE            (80+1)

#define cDISP_INI_STATUS        (0x00)          /* Initial status display */
#define cDISP_CONTROL_CH        (0x01)          /* Control channel display */
#define cDISP_CARRIER_FREQUENCY     (0x02)      /* Carrier frequency display */
#define cDISP_DEAD_TIME             (0x03)      /* Dead time display */
#define cDISP_GATE_ACTIVE           (0x04)      /* Gate active display */
#define cDISP_POSITION_DETECT       (0x05)      /* position detect display */
#define cDISP_VDC_VOLTAGE           (0x06)      /* VDC voltage display */
#define cDISP_INVERTER_TEMP         (0x07)      /* Inverter Temp */
#define cDISP_UVW_VOLTAGE           (0x08)      /* U-phase/V-phase/W-phase voltage display */
#define cDISP_DAC_DATA              (0x09)      /* Dac data display */
define cDISP_CONTROL_CH             (0x0A)      /* Control channel display */
define cDISP_GATE_ACTIVE            (0x0B)      /* Gate active display */
define cDISP_POSITION_DETECT        (0x0C)      /* position detect display */
define cDISP_GATE_ACTIVE            (0x0D)      /* Gate active display */

#define cPOSITION_DETECT_3SHUNT     (3)         /* Position Ditect : 3shunt */
#define cPOSITION_DETECT_1SHUNT     (1)         /* Position Ditect : 1shunt */

/* User_interface Setting */
```

```
#define cFLG_ON                 (1)
#define cFLG_OFF                (0)
#define cSW_Hi                  (1)
#define cSW_Low                 (0)
#define cINV_MIN_TEMP           (-200)
#define cINV_MAX_TEMP           (1000)
#define cSPD_UPDOWN_RESOLUTION  (10)
```

12. Timings for Control and Data Update

12.1 Vector Control Using Software

12.1.1 3-shunt Control

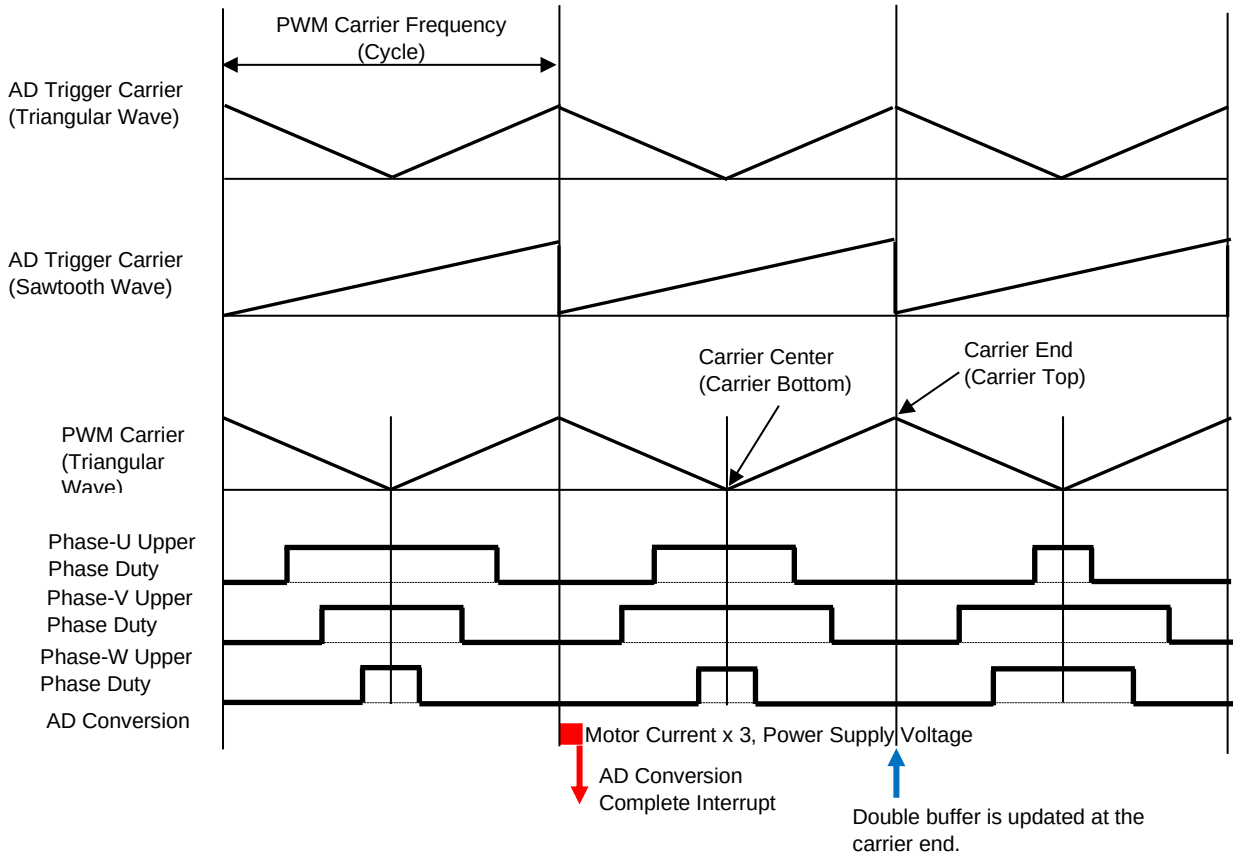


Figure 12.1 3-shunt Control

12.1.1.1 Carrier Waveform

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

12.1.1.2 Double Buffer Update Timing

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

12.1.1.3 Interruption

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

12.1.2 1-shunt Control

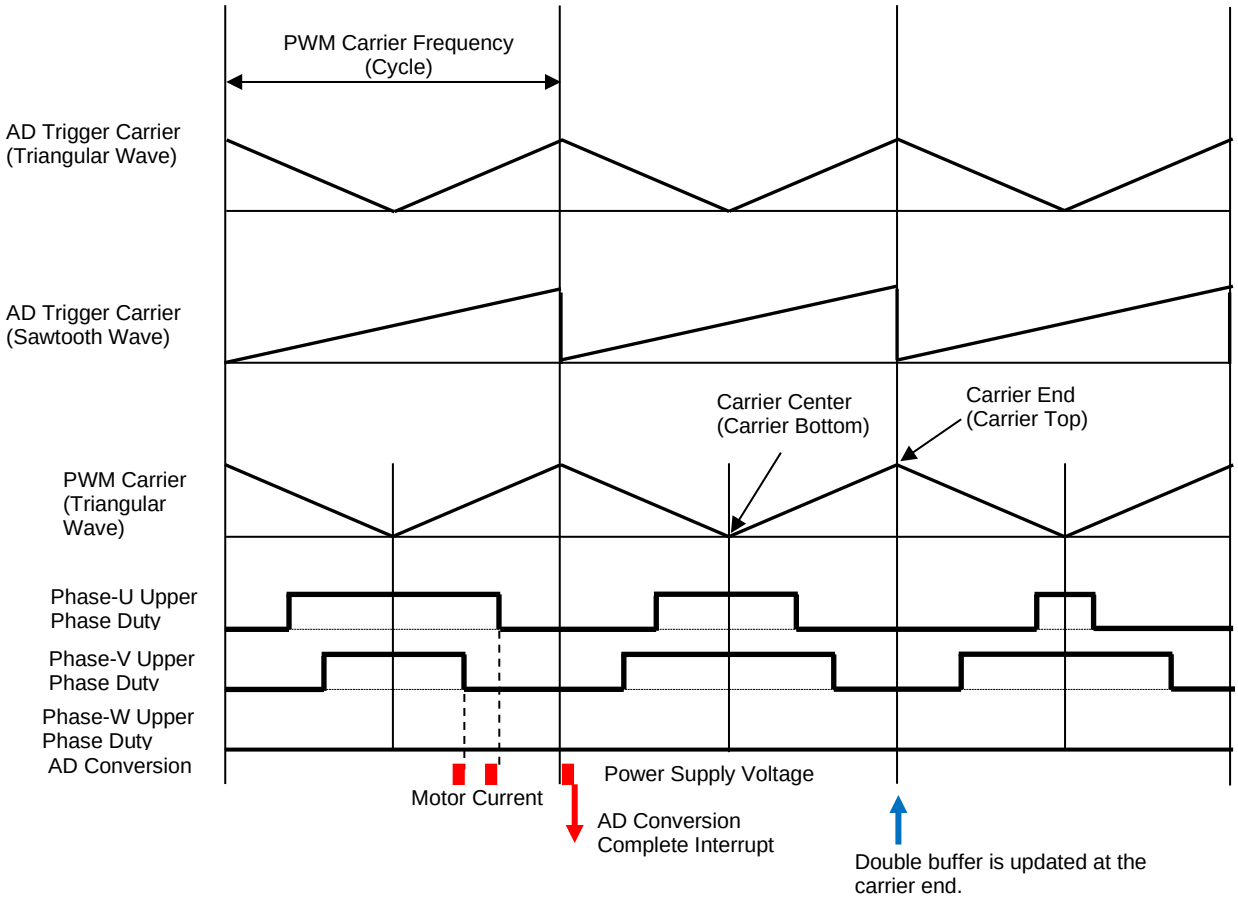


Figure 12.2 1-shunt Control

12.1.2.1 Carrier Waveform

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

12.1.2.2 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

12.1.2.3 Interruption

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

13. Peripheral Driver

13.1 Microcontroller Peripheral Circuit Addresses

Define the base addresses of microcontroller peripheral circuit registers to be passed to the peripheral driver API.

13.1.1 Data Structure

13.1.1.1 ipdrv_t

Data Fields:

TSB_PMD_TypeDef* const	**PMDx:** Select an PMD Address.
TSB_AD_TypeDef* const	**ADx:** Select an ADC Address.

13.2 Motor Control Circuit (PMD)

13.2.1 Specifications of Functions

13.2.1.1 IP_PMD_init

PMD Initial Setup

API:

void IP_PMD_init(TSB_PMD_TypeDef* const PMDx, PMD_InitTypeDef* const _initdata)

Parameter:

PMDx: Select a PMD address

_initdata: PMD Initial Setup Data Structure

For details, refer to 13.2.2.1 PMD_InitTypeDef.

Function:

Initial setup of PMD

Supplement:

Call up during PMD suspension with prohibited interruption.

Returned Value:

No

13.2.1.2 PMD_GetEMG_Status

Acquisition of EMG Protection Status

API:

emg_status_e PMD_GetEMG_Status(const ipdrv_t* const _ipdrv)

Parameter:

_ipdrv: Specify the structure where the microcontroller peripheral circuit addresses are defined.

For details, refer to 13.1.1.1.

Function:

Acquires the EMG protection status.

Supplement:

None

Returned Value:

emg_status_e: EMG Protection Status

cNormal: Normal

cEMGProtected: Protected

13.2.1.3 PMD_ReleaseEMG_Protection

EMG Protection Status Released

API:

Void PMD_ReleaseEMG_Protection(const ipdrv_t* const _ipdrv)

Parameter:

_ipdrv: Specify the structure where the microcontroller peripheral circuit addresses are defined.

For details, refer to 13.1.1.1.

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.

Returned Value:

No

13.2.2 Data Structure

13.2.2.1 PMD_InitTypeDef

Data Fields:

uint8_t shunt: Shunt Type

1: 1-shunt

3: 3-shunt

uint8_t poll: L-Side Polarity

0: L active

1: H active

uint8_t polh: H-Side Polarity

0: L active

1: H active

uint16_t pwmrate: PWM Frequency

A value to be set up to PMDxRATE

uint16_t pwmfreq: PWM Frequency

A value to be set up to PMDxMDPRD

uint16_t deadtime: Dead Time

A value to be set up to PMDxDTR

uint8_t busmode: Mode Selection

A value to be set up to PMDxMODESEL

13.3 Analogue Digital Converter (ADC)

13.3.1 Specifications of Functions

13.3.1.1 IP_ADC_init

ADC Initial Setup

API:

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

Parameter:

ADx: Select an ADC Address

_initdata: ADC Initial Setup Data Structure

For details, refer to 13.3.2.1 AD_InitTypeDef.

Function:

Initial setup of ADC

Supplement:

Call up during ADC suspension with prohibited interruption.

Returned Value:

No

13.3.2 Data Structure

13.3.2.1 AD_InitTypeDef

Data Fields:

uint8_t shunt: Shunt Type

1: 1-shunt

3: 3-shunt

uint8_t **iuch:** Phase-U Current Input AD Channel No.(3-shunt)

uint8_t **ivch:** Phase-V Current Input AD Channel No.(3-shunt)

uint8_t **iwch:** Phase-W Current Input AD Channel No.(3-shunt)

uint8_t **idcch:** DC Current Input AD Channel No.(1-shunt)

uint8_t **vdcch:** Motor Power Supply Voltage Vdc Input AD Channel No.

uint8_t **pmd_ch:** Select a PMD Channel

cPMD0: PMD Channel 0

cPMD1: PMD Channel 1

uint8_t pints: Select Interruption for PM Trigger

cPINTS_A: INTADxPDA

cPINTS_B: INTADxPDB

cPINTS_C: INTADxPDC

cPINTS_D: INTADxPDD

uint32_t** **p_adreg0:** AD Conversion Result Storage Register Address 0

uint32_t** **p_adreg1:** AD Conversion Result Storage Register Address 1

uint32_t** **p_adreg2:** AD Conversion Result Storage Register Address 2

uint32_t** **p_adreg3:** AD Conversion Result Storage Register Address 3

14. Appendix

14.1 Fixed Decimal Point Processing

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

14.2 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 (0x7fff) for the data will be "1" and the minimum value (0x8000) will be "-1."

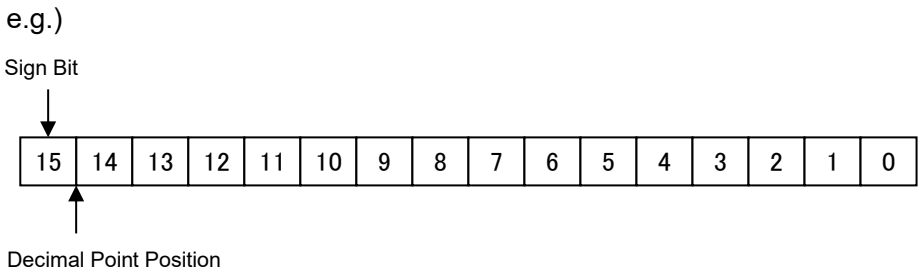


Figure 14.1 Example of 16-Bit Data

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.

14.3 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 16-bit data, computation is performed using Q15 format (15 bits for decimal fraction) and Q31 format for 32-bit data (31 bits for decimal fraction). The value that can be displayed in the decimal format depends on the format.

Table 14.1 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	32767.000000000000000	-32768

Table 14.2 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	0.999999999534338	-1
Q30	30	1.999999999068670	-2
Q29	29	3.999999998137350	-4
Q28	28	7.999999996274700	-8
Q27	27	15.999999992549400	-16
Q26	26	31.999999985098800	-32
Q25	25	63.999999970197600	-64
Q24	24	127.999999940395000	-128
Q23	23	255.999999880790000	-256
Q22	22	511.999999761581000	-512
Q21	21	1023.999999523160000	-1024
Q20	20	2047.999999046320000	-2048
Q19	19	4095.999998092650000	-4096
Q18	18	8191.999996185300000	-8192
Q17	17	16383.999992370600000	-16384
Q16	16	32767.999984741200000	-32768
Q15	15	65535.999969482400000	-65536
Q14	14	131071.999938965000000	-131072
Q13	13	262143.999877930000000	-262144
Q12	12	524287.999755859000000	-524288
Q11	11	1048575.999511720000000	-1048576
Q10	10	2097151.999023440000000	-2097152
Q9	9	4194303.998046870000000	-4194304
Q8	8	8388607.996093750000000	-8388608
Q7	7	16777215.992187500000000	-16777216
Q6	6	33554431.984375000000000	-33554432
Q5	5	67108863.968750000000000	-67108864
Q4	4	134217727.937500000000000	-134217728
Q3	3	268435455.875000000000000	-268435456
Q2	2	536870911.750000000000000	-536870912
Q1	1	1073741823.500000000000000	-1073741824
Q0	0	2147483647.000000000000000	-2147483648

14.4 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 set the correct 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

14.5 Assert Function

To report the name of the source file and source line number where the assert_param error has occurred, "DEBUG" must be defined.

And detailed definition of assert_failed() is needed to be implemented, which can be done, for example, in the main.c file.

(example)

```
#ifdef DEBUG
/*
  * @brief   Deal with the error parameter
  * @param   file: Pointer to the file where the error parameter locates
  * @param   line: Number of the line in which the error parameter locates
  * @retval None
 */
void assert_failed(char *file, int32_t line)
{


}
#endif
```

15. Revision History

Date	Rev.	Description
2025-03-03	1.0	First release

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, lifesaving and/or life supporting medical equipment, equipment used for automobiles, trains, ships and other transportation, traffic signaling equipment, equipment used to control combustions or explosions, safety devices, elevators and escalators, and devices related to power plant. **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.
- Do not disassemble, analyze, reverse-engineer, alter, modify, translate or copy Product, whether in whole or in part.
- 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.**