TMPM375
Motor Sample Software
Application Note

Table of Contents

List of Figures

List of Tables

1. Overview

1.1. Introduction

The operation of this motor control sample software has been checked using the SBK-M375 (TMPM375 microcontroller evaluation board) by ESP. Please contact ESP (http://esp.co.jp/) for details on this evaluation board.

1.2. Specifications

- Microcontroller TMPM375FSDMG
- Clock 40 MHz
- Motor Tsukasa Electric - TG611B
- Development Environment IAR Embedded Workbench for ARM V9.70.1
  KEIL µVision MDK-Lite V5.43.1.0
  Segger Embedded Studio for ARM V8.24
- Operating Voltage DC24V

- Overview
  - Brushless DC Motor Vector Control (Speed Control)
  - Evaluation Purpose
    - DAC Output (variable analogue output)
    - Toggle switch (Rotation direction) and Slide volume (Rotation count) inputs

- Motor Control

Table 1.1 Motor Control

Item	Control
Rotating Speed Control Method	Constant Rotating Speed Control (variable resistance operation)
Rotating Direction	CW/CCW (switch changeover)
Current Detection	3-shunt or 1 shunt
Position Detection	Sensor-less and encoder (Note1)
Op Amplifier	Built-In Microcontroller or External

Note1: Unevaluated

Precautions

When an EMG occurs, please turn off the motor drive using the VR controls to cancel it.
When operating with 1 shunt, check the setups as follows:
- D_Para_CH1.h
  #define cSHUNT_TYPE_CH1
  #define cBOOT_TYPE_CHx (cBoot_v)

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.

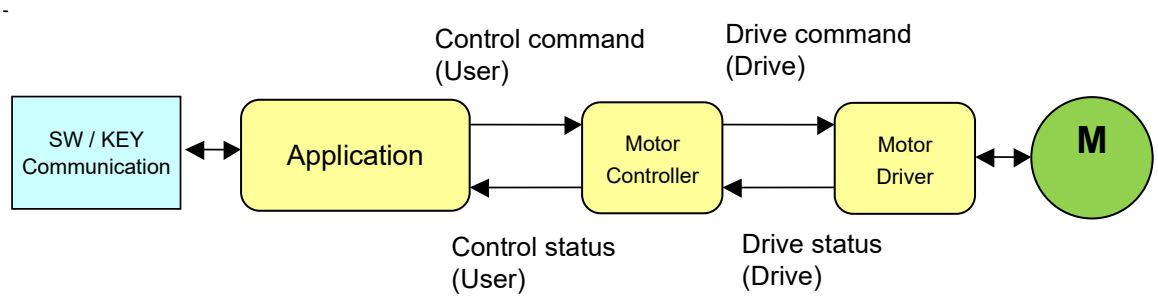


Figure 1.1 Vector Control Software Configuration

i. The application will input the control commands that are set up through the switches, keys, communication, etc. and give them to the motor controller together with other control commands. Additionally, it obtains the control status from the motor control layer, performs the necessary processing, and displays it on devices such as LEDs.

ii. The motor controller reads the control commands that are given from the application, converts them 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 target speed is given from the application layer while the motor is running, the motor drive layer cannot respond to a sudden change in target speed. Therefore, the motor control first converts it into a gradually changing drive target speed before passing it to the motor drive layer.

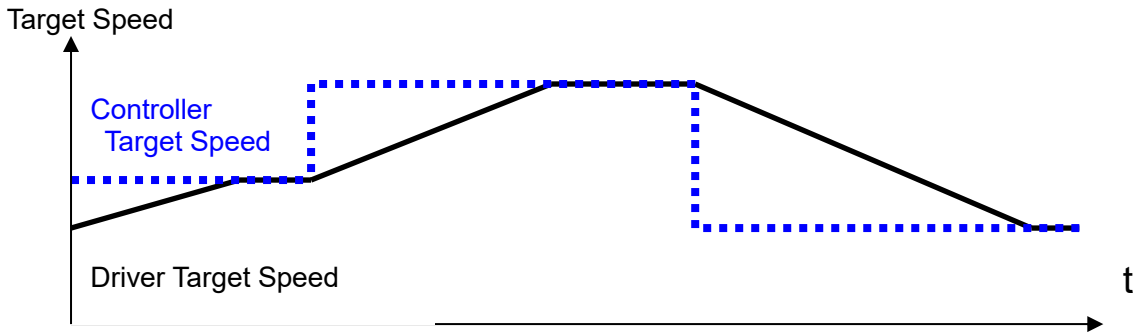


Figure 1.2 Controller Target Speed and Driver Target Speed

2. System Block Diagram

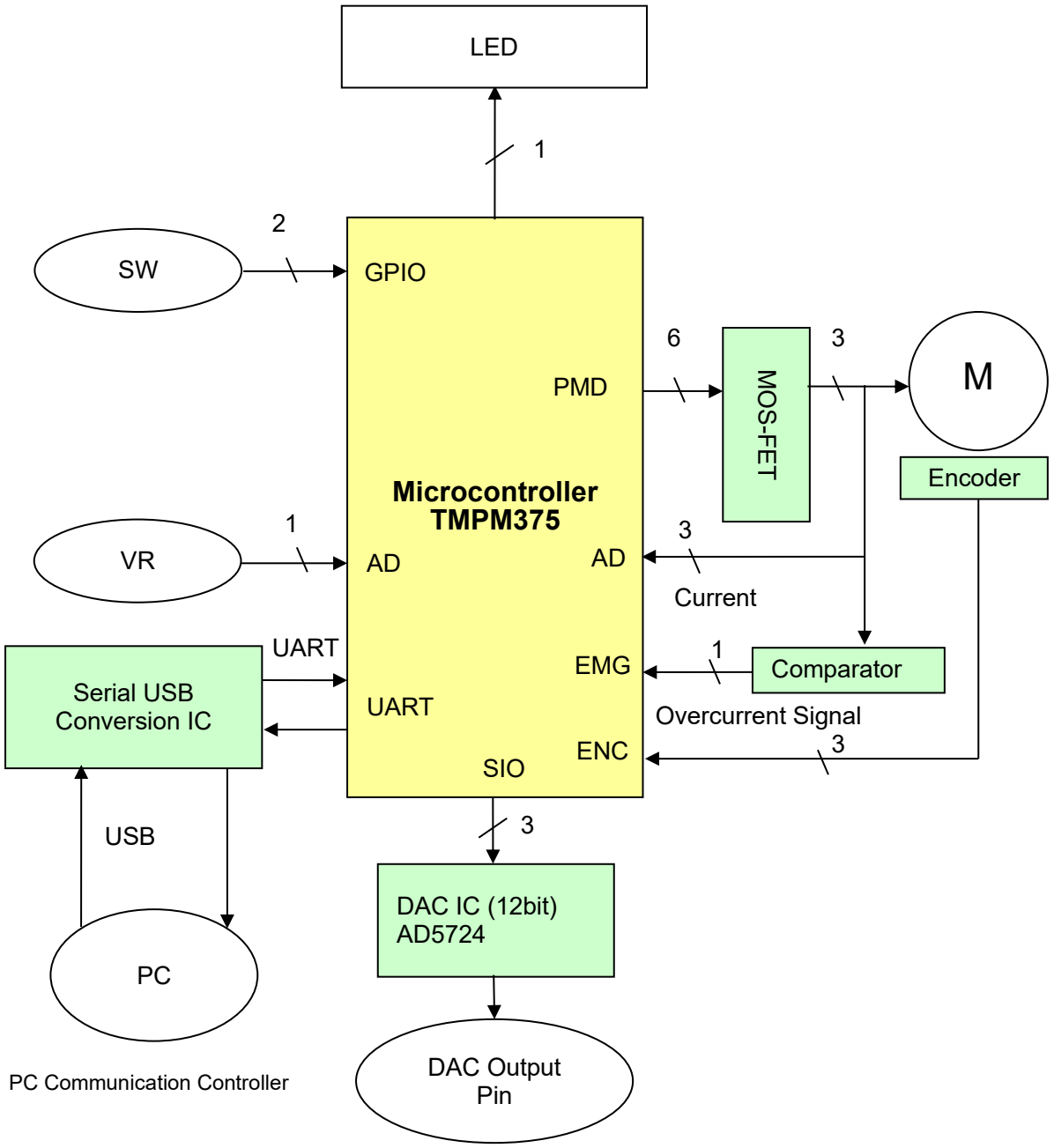


Figure 2.1 System Block Diagram

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
 ├ 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_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
 ├ 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
 ├ sys_macro.h           Macro Definition Header File
 ├ usercon.c             User Control Source File
 ├ usercon.h             User Control Header File
 │
 └Libraries              The library for CMSIS Core and individual IPs are stored.
   └M375
     └CMSIS                          CMSIS Core is stored.
     │  ├ system_TMPM375.c           Microcontroller Initial Setup Source File
     │  ├ system_TMPM375.h           Microcontroller Initial Setup Header File
     │  └ TMPM375.h                  Microcontroller Register Definition Header File
     │  └startup
     │     ├arm
     │     │     startup_TMPM375.s          Start Up Assembler Source (arm)
     │     │     TMPM375.scat
     │     └iar
     │     │   startup_TMPM375.s            Start Up Assembler Source (IAR)
     │     │     TMPM375.icf                Linker File (IAR)
     │     └segger
     │           TMPM375_Vectors.s          Start Up Assembler Source (Segger)
     │           SEGGER_THUMB_Startup.s      Start Up Assembler Source (Segger)
     │           TXZ4Aplus_M4K2_Startup.s    Start Up Assembler Source (Segger)
     │           Cortex_M_Startup.s          Start Up Assembler Source (Segger)
     │           SEGGER_Flash.icf            Linker File (Segger)
     │
     └IP_Driver          Peripheral Driver is stored.
         tmpm375_adc.c
         tmpm375_adc.h
         tmpm375_cg.c
         tmpm375_cg.h
         tmpm375_dnf.c
         tmpm375_dnf.h
         tmpm375_enc.c
         tmpm375_enc.h
         tmpm375_fc.c
         tmpm375_fc.h
```

tmpm375_gpio.c
tmpm375_gpio.h
tmpm375_ofd.c
tmpm375_ofd.h
tmpm375_pmd.c
tmpm375_pmd.h
tmpm375_sbi.c
tmpm375_sbi.h
tmpm375_tmrb.c
tmpm375_tmrb.h
tmpm375_trmosc.c
tmpm375_trmosc.h
tmpm375_uart.c
tmpm375_uart.h
tmpm375_vltd.c
tmpm375_vltd.h
tmpm375_wdt.c
tmpm375_wdt.h
tx03_common.h

4. Evaluation Environment

4.1. Development Environment

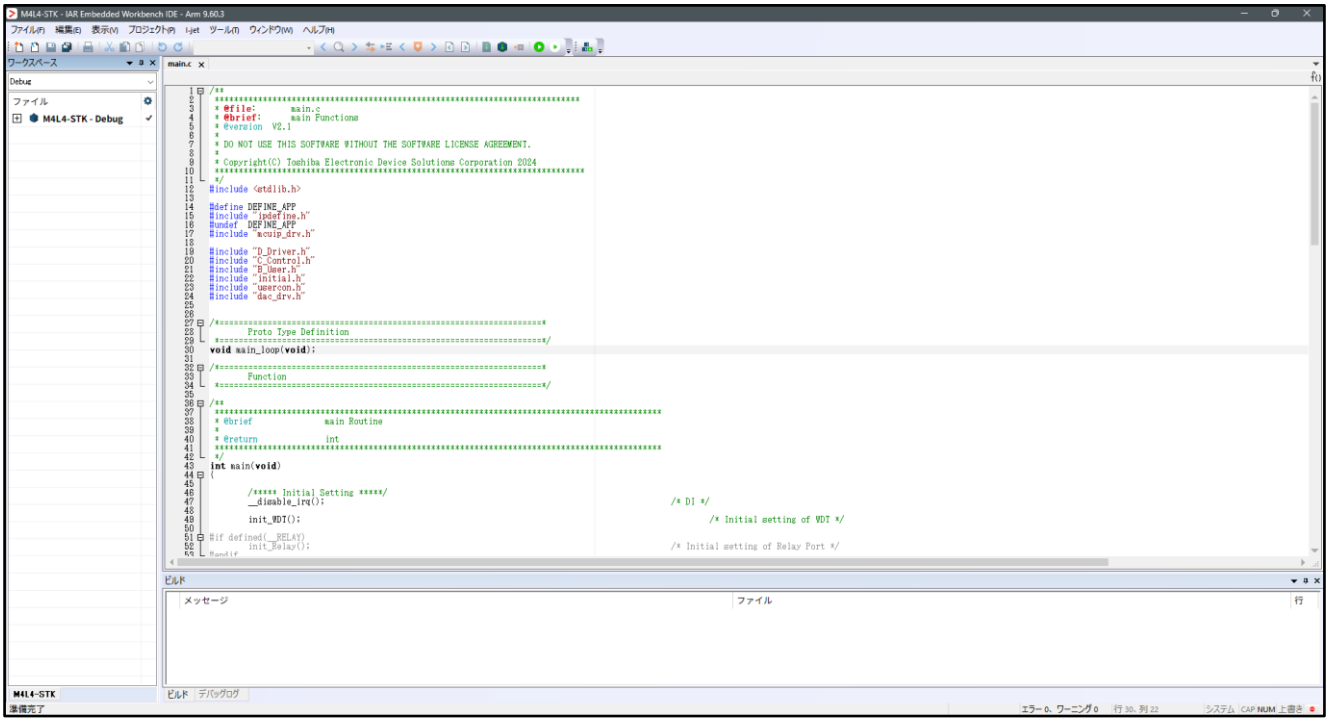
This software supports the following development tools.

IAR Embedded Workbench for ARM V9.70.1
KEIL µVision MDK-Lite V5.43.1.0
Segger Embedded Studio for ARM V8.24

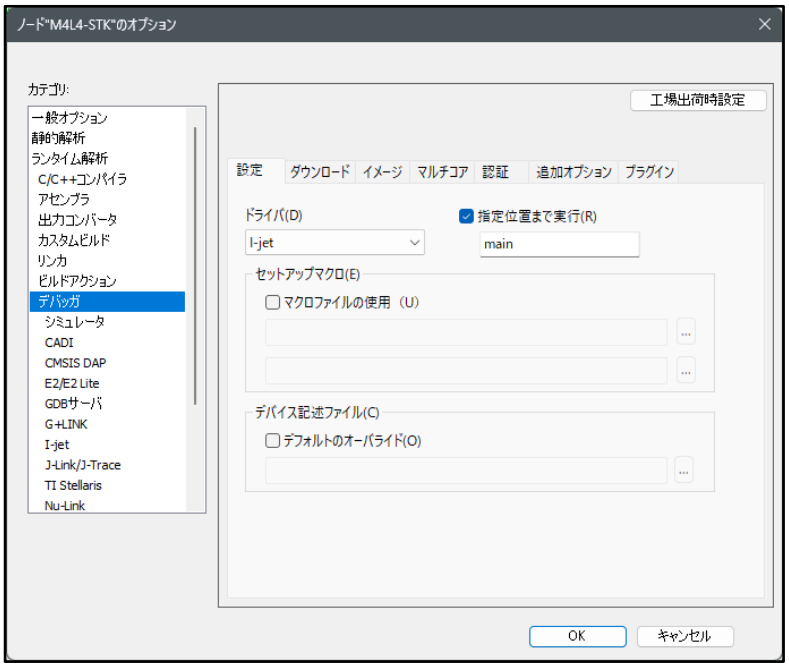
4.2. How to Start a Project

This explains the case for IAR Embedded Workbench.

1. Double-click iar¥M375-STK.eww, or open M375-STK.eww from [File] > [Open] > [Workspace].
2. The following screen will appear.



3. Open the options and select the tool you want to use.
[Project] > [Options] > [Debugger] tab <Settings>

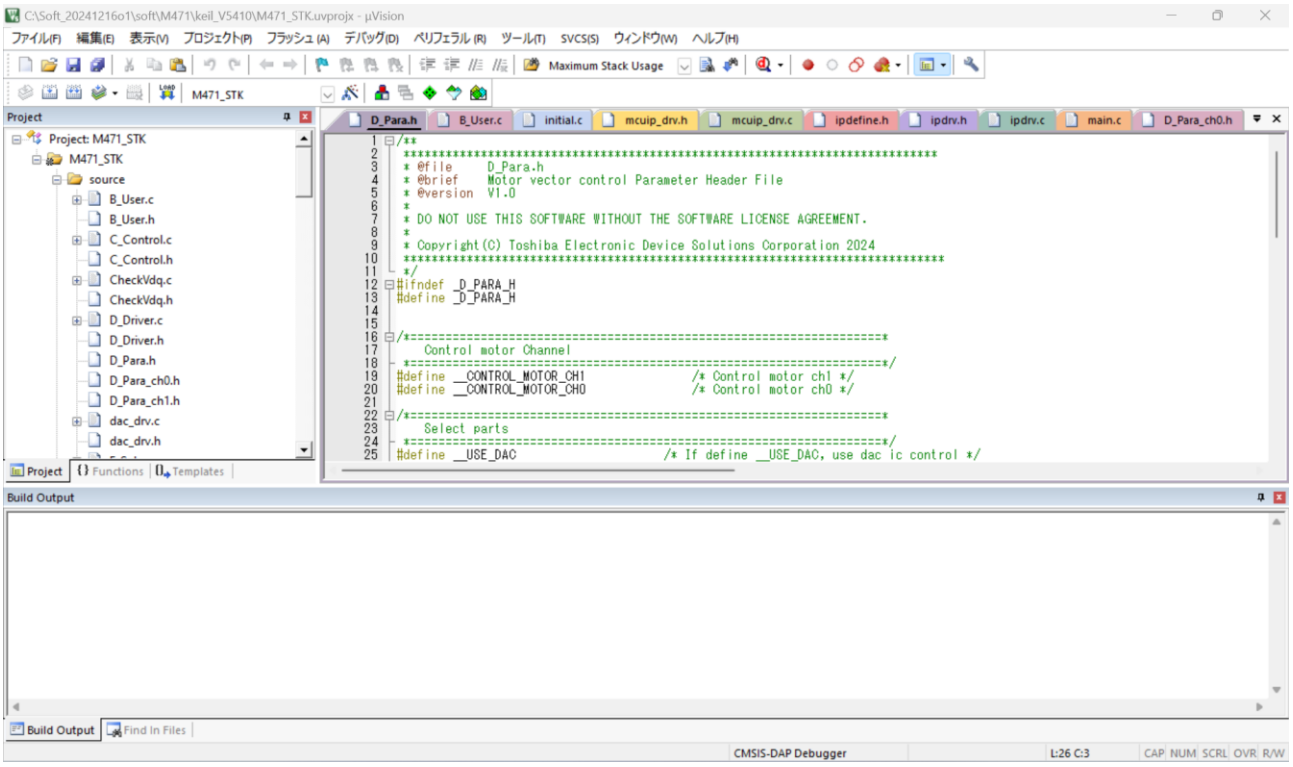


Select the tool to use.

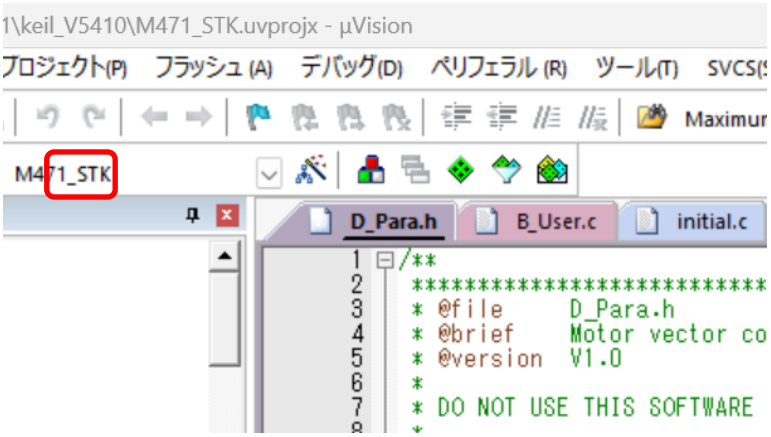
4. To start debugging, connect the tool and select [Project] > [Download and Debug], or press the button below.

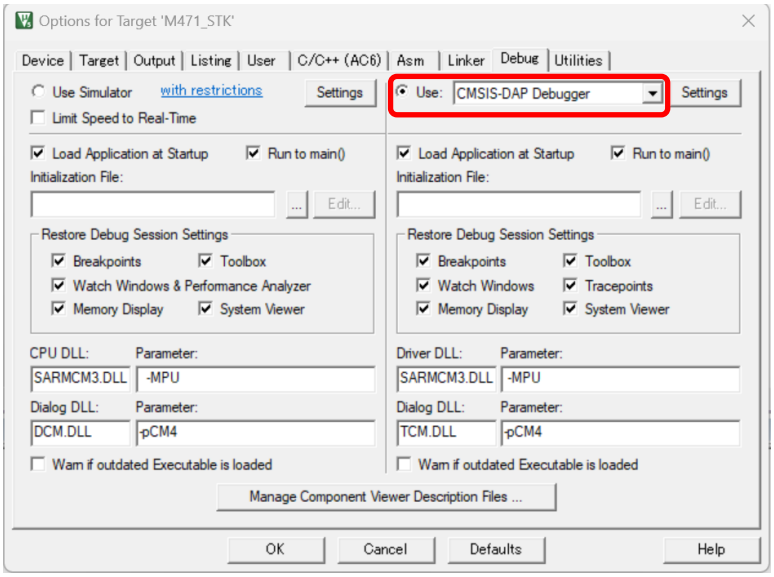
This explains the case for Keil µVision MDK.

1. Double-click M375_STK.uvprojx, or open M375_STK.uvprojx from [File] > [Open].
2. The following screen will appear.



3. Open the target settings and select the tool you want to use.
[Target Settings] > [Debug] in the <Use> field



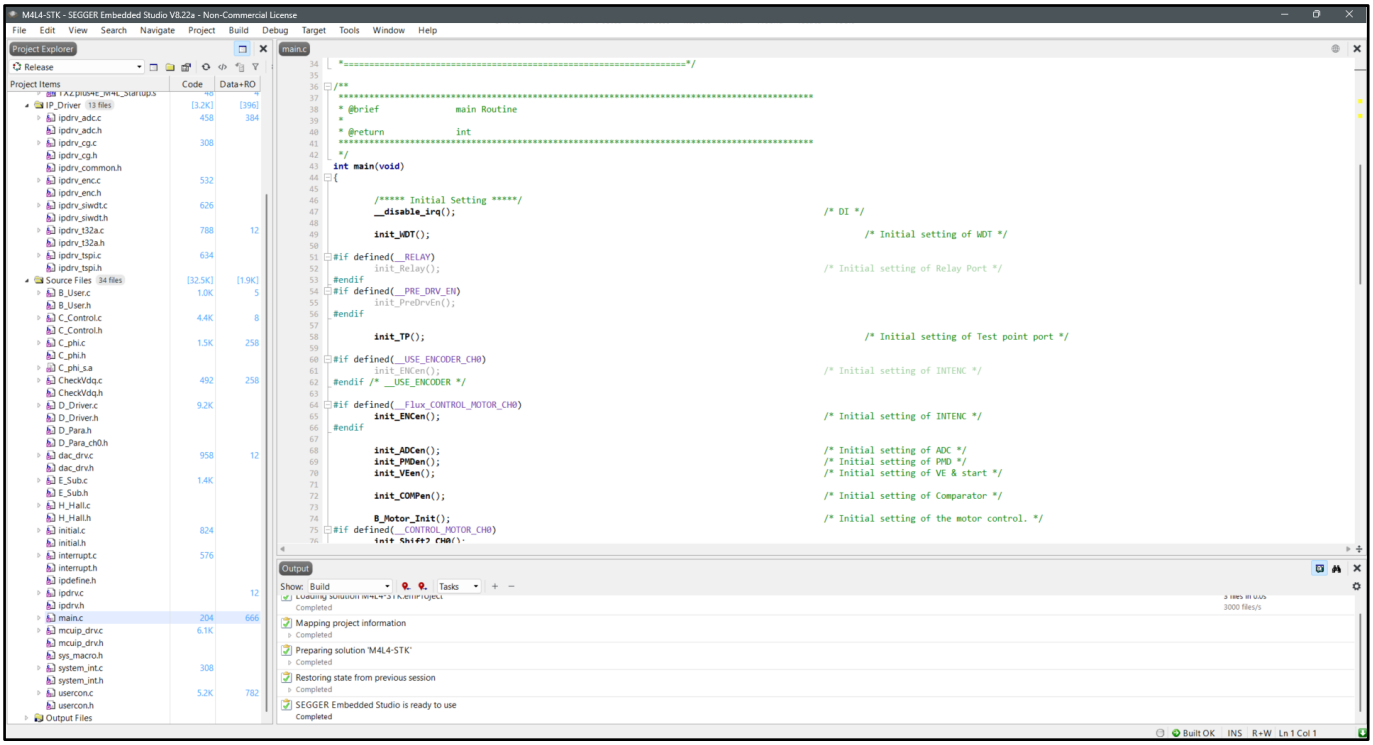


Select the tool to use.

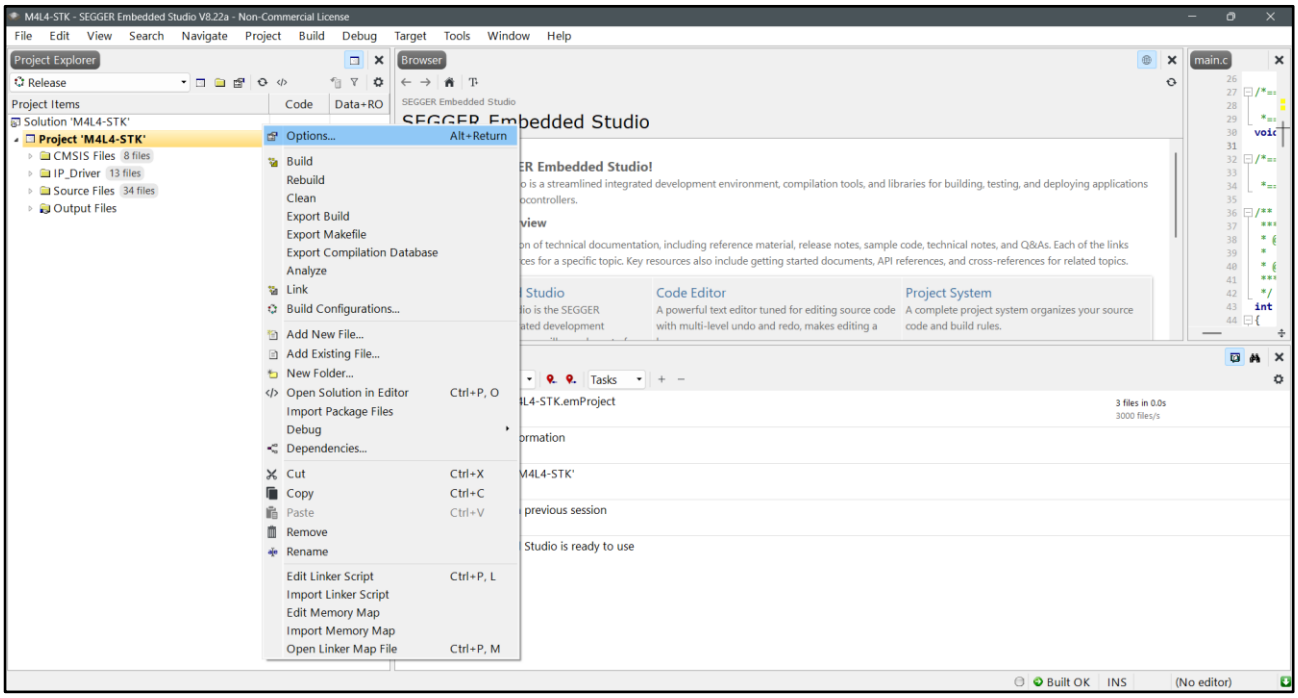
4. To start debugging, connect the tool and select [Debug] > [Start/Stop Debug Session], or click the button below.

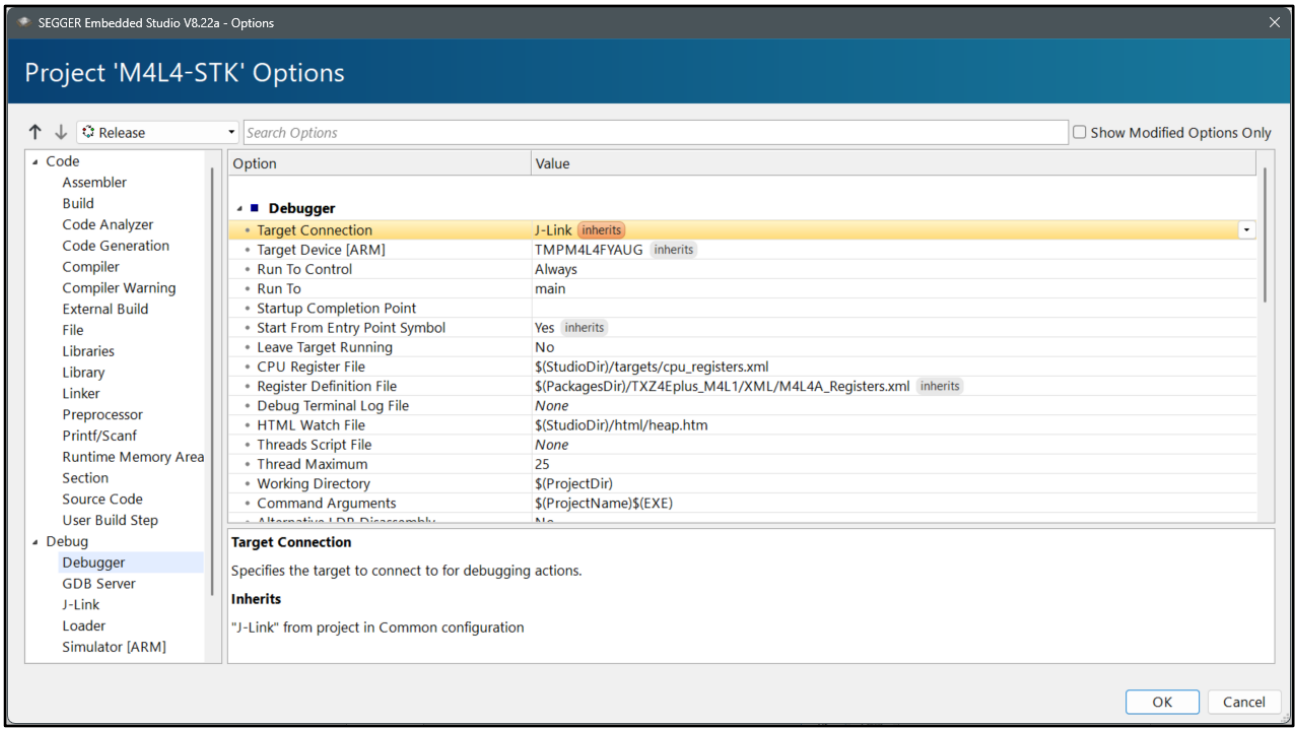
This explains the case for Segger Embedded Studio.

1. Double-click M375-STK.emProject, or open it from [File] > [Open].
2. The following screen will appear.



3. Open Options and select the tool you want to use.
   Right-click the project name > [Options] > [Debugger] > <Target Connection>





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 any variable for checking is missing.

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

4.4. PORT Output

It is possible to output various types of data to the selected ports. The ports that are configured to output data by default are listed below. Please change them as needed.

4.4.1. Vector Control Software Processing Time

It is possible to output the software processing time of vector control. To enable this feature, please enable the following definition in the file ipdefine.h:
#define __MOTOR_DBGOUT_VECTOR_TIME_INT

Table 4.1 PORT Output

Motor Channel	Output Port
CH1	PE bit1

H: While processing

4.4.2. Current Detection State

It is possible to output the current detection status. To enable this feature, please activate the following definition in the file ipdefine.h.
#define __MOTOR_DBGOUT_IDET

Table 4.2 PORT Output

Motor Channel	Output Port
CH1	PE bit1

H: Normal detection state, L: False detection state

4.5. Stage History

By keeping a history of the main stage, it is possible to check from which state it transitioned to an abnormal state. The stage history is stored in the following variable:
stage_his[]
To enable this feature, activate the following definition in the file ipdefine.h:
#define __MOTOR_DEBUG

4.6. User Interface

The following functions are provided as the user interface:

<<User Interface>>

1. Rotation Direction Switch
   The rotation direction of the motor can be changed by switching SW (TSW1).

off: CW, on: CCW

2. VR (Volume resistor)
   By operating VR1, the rotation speed of the motor can be adjusted.
   Speed Range: 12 rps to 200 rps (electrical angle)

3. LED Display
   The motor error status can be checked using the LED.
   In the case of a hard EMG during the STOP stage, the LED will not illuminate.
   Enabling the LED display requires configuration.
   For more details, please refer to Section 11.1.4 LED/PC UART Output Selection
   When driving Motor CH1 (LED5):
   Error Status LED
   No EMG: Off
   EMG: On
   The EMG status is cleared when the motor drive stopped.

4. Initial Display via UART
   After releasing the reset, it is possible to display the message and board name on the PC once.
   For details, please refer to Section 9.1.6 Terminal Software Display.

5. Module Configuration

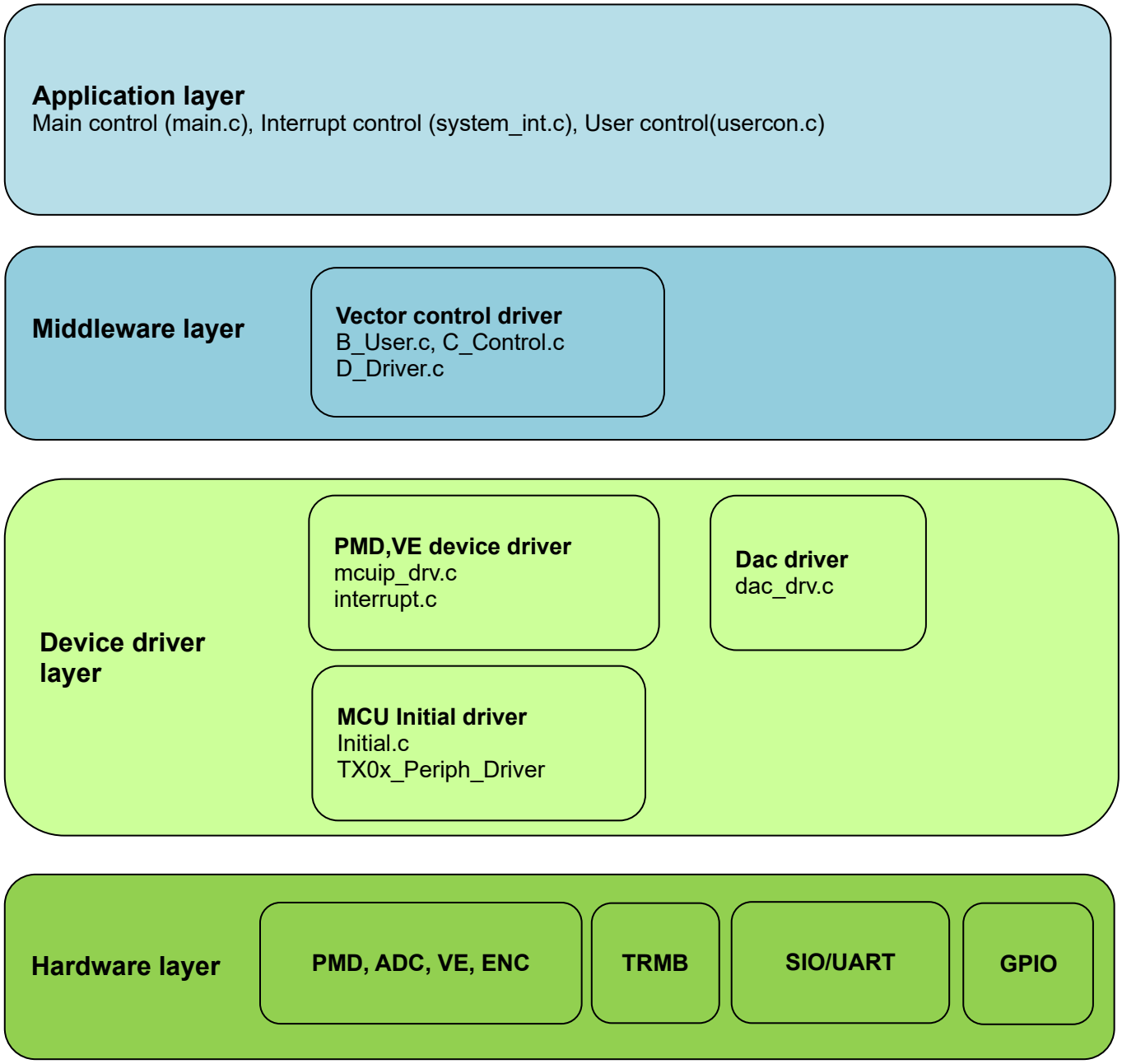


Figure 5.1 Module Configuration

6. Microcontroller Hardware Assignment

6.1. IP

Table 6.1 IP

IP		Control	Remarks
TMRB	TMRB0	4-kHz cycle interrupt	
	TMRB4	Unused.	
	TMRB5	Unused.	
SIO/UART	Channel 0	DAC IC control	Used for SIO
	Channel 1	PC / UART	Used for UART
	Channel 2	Unused.	
	Channel 3	Unused.	
ADC	Unit A	Unused.	
	Unit B	Motor CH1: Acquisition of motor current and VDC voltage values VR: Acquisition of ADKey Voltage Value	
PMD	Channel 1	Motor CH1 control	
VE	Channel 1	Motor CH1 control	
ENC	Channel 1	Motor CH1 encoder signal control	

6.2. Interruption

Table 6.2 Interruption

Cause	Process	Priority	Function
INTTB00_IRQn	4-kHz cycle timing generation	5	INTTB00_IRQHandler
INTVCNB_IRQn	Vector control software process for CH1	3	INTVCNB_IRQHandler
INTTX0_IRQn	DAC IC control	6	INTTX0_IRQHandler

The priority of interruption can be changed using the following constants in ipdefine.h.

```
/* High     Low */
/*   0 ---- 7   */
INT4KH_LEVEL        5       /* 4kH interval timer interrupt */
INT_VC_LEVEL        3       /* VE interrupt */
INT_DAC_LEVEL       6       /* SIO interrupt for DAC */
```

7. General Flow

7.1. Main Routine (main)

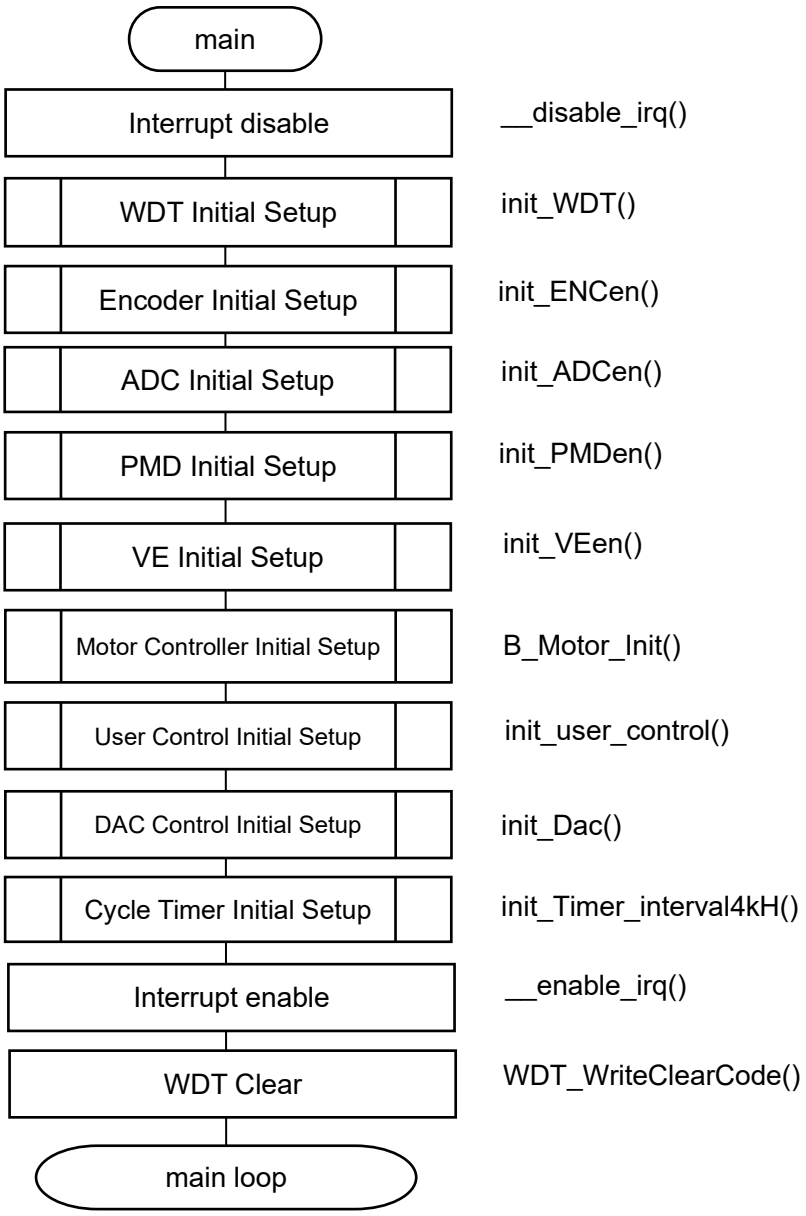


Figure 7.1 Main Routine

7.2. Main Loop (main_loop)

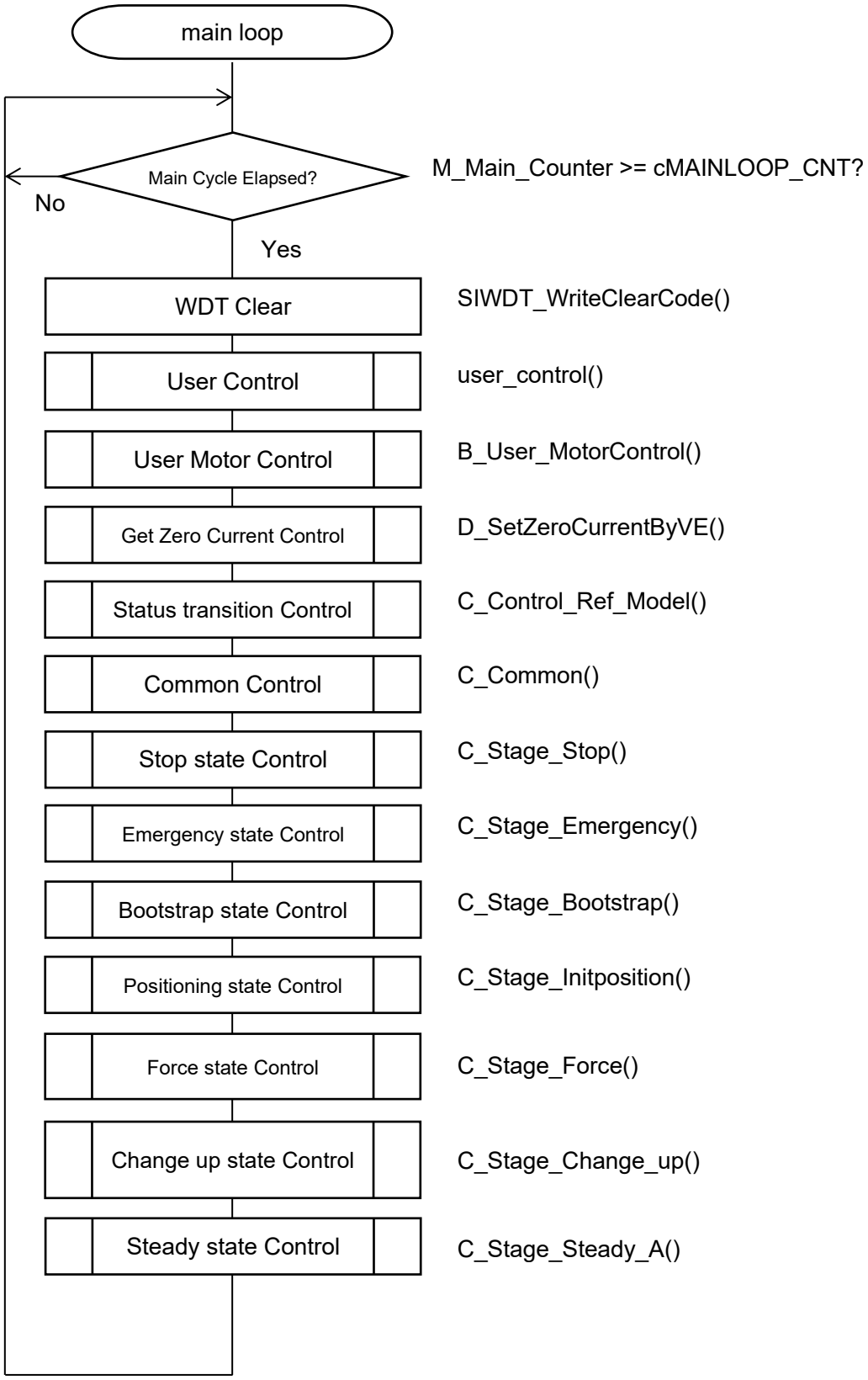


Figure 7.2 Main Loop

7.3. Interruption (interrupt.c)

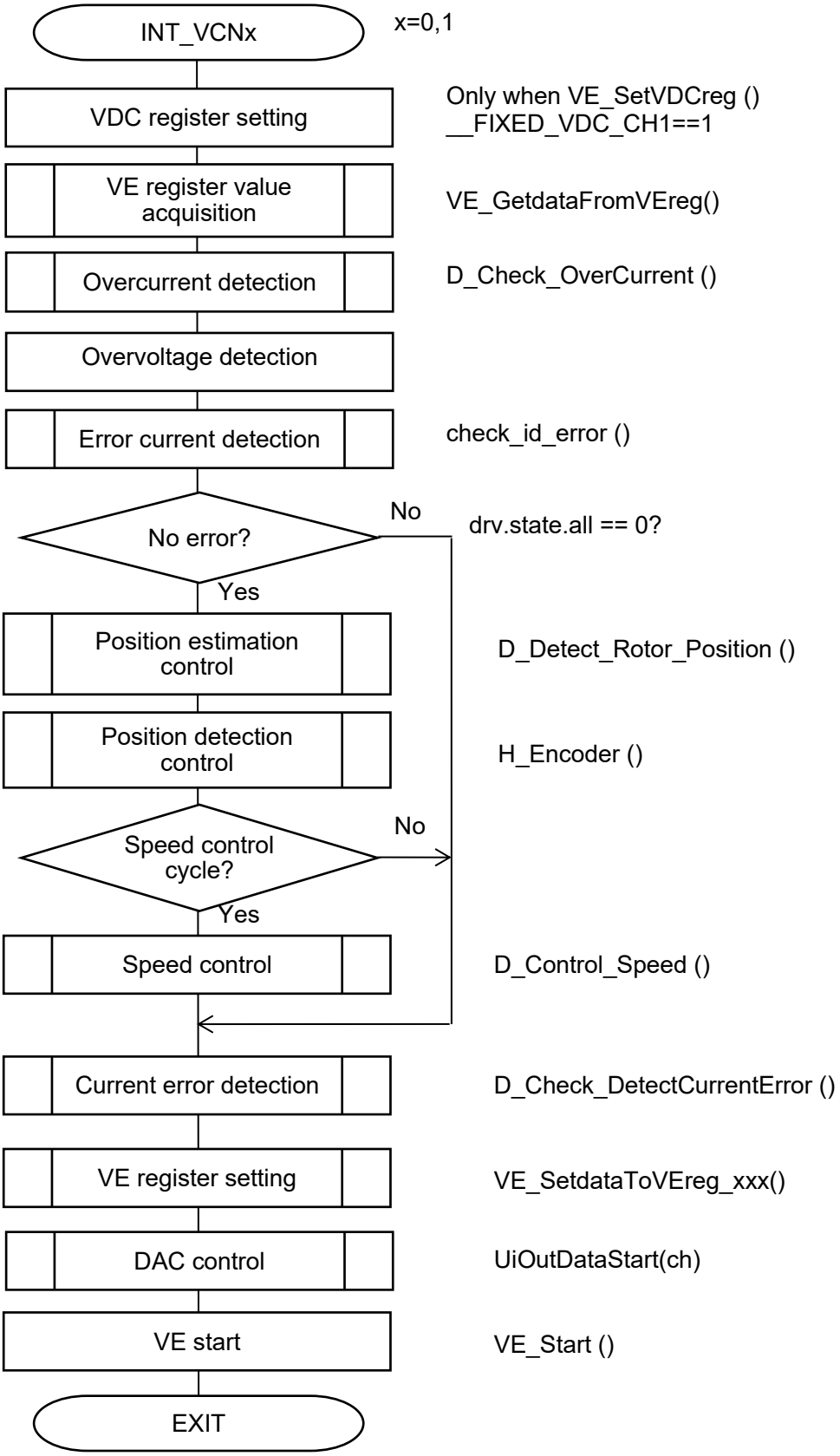


Figure 7.3 Interruption

8. Motor Operation Status Transition

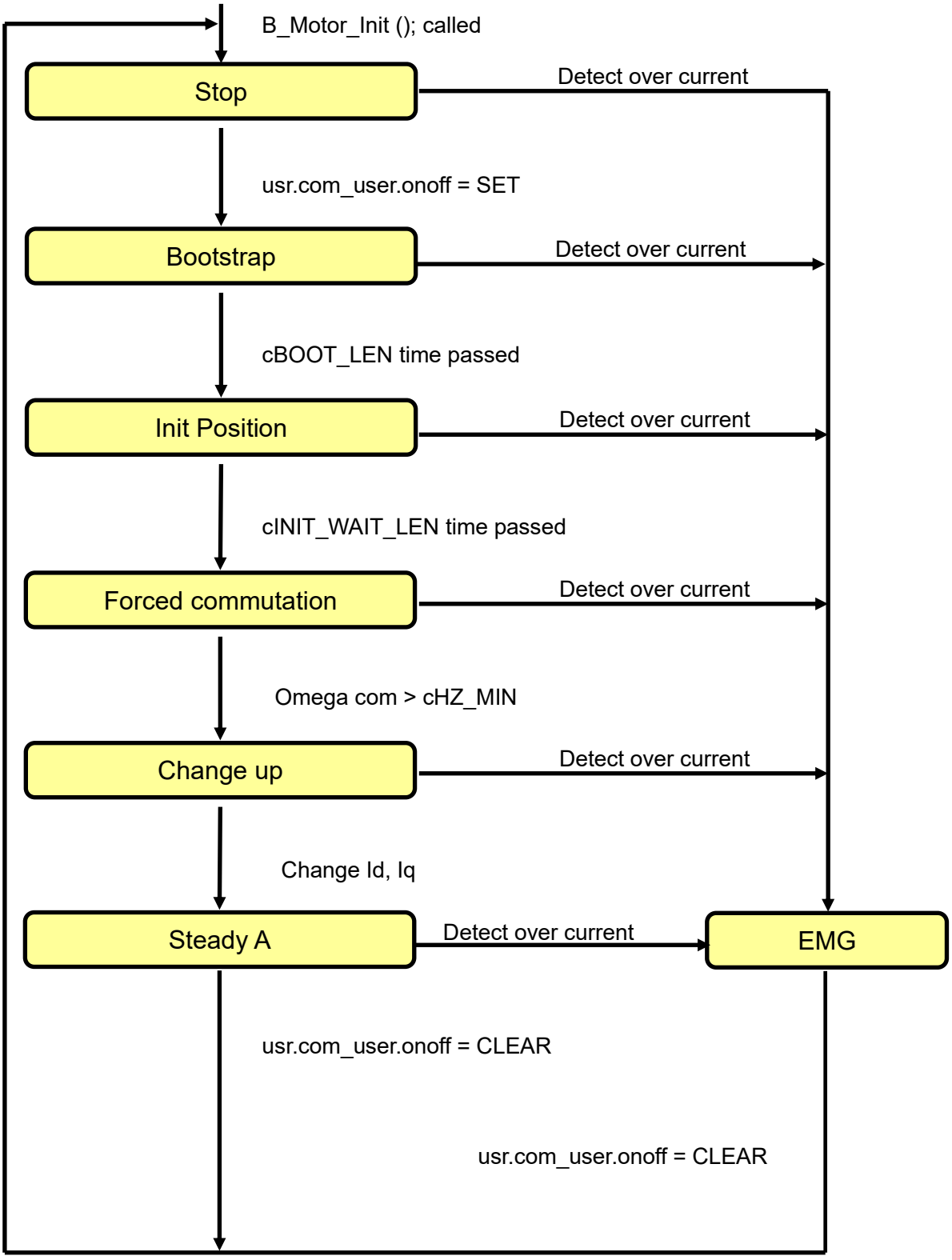


Figure 8.1 Motor Operation Status Transition

9. User Application

9.1. User Control

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

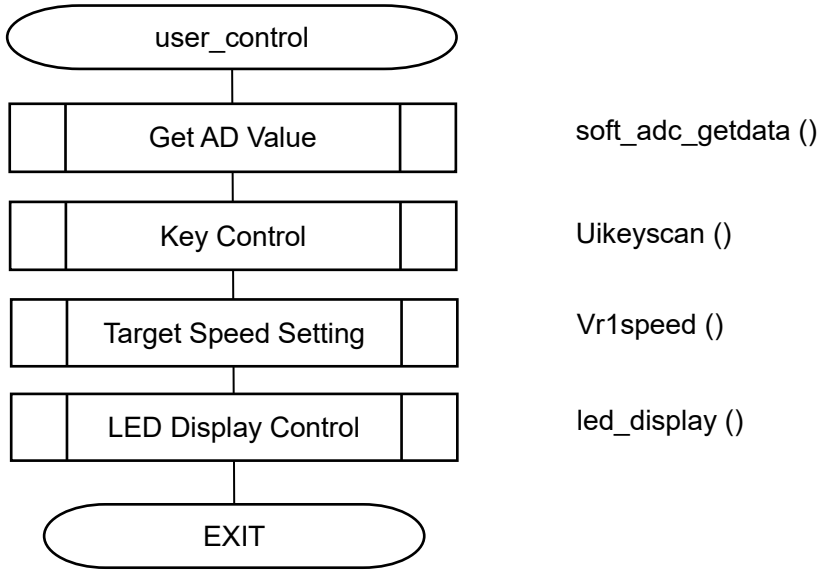


Figure 9.1 User Control

9.1.1. Get AD Value (soft_adc_getdata)

The AD value (12-bit) of VR1 is acquired using the single conversion setting. After 10 acquisitions, the average of the 8 values excluding the maximum and minimum is stored as the respective effective values in ad_vr.avedat.

9.1.2. Key Control (Uikeyscan)

Key scan processing for TSW1 will be performed. Each key data is confirmed after 20 consecutive matches and then stored in keydata.

9.1.3. User Setup (user_interface)

The following settings will be configured according to the SW input by user operation.

1. Rotation Direction Switching
   This process is carried out when the motor is stopped (when the main stage of the motor is in a stopped state).
   The rotation direction is switched according to the value of keydata.tsw1.
   - If keydata.tsw1 is 1: CW (vr1_speed: positive)
   - If keydata.tsw1 is 0: CCW (vr1_speed: negative)

2. Motor Speed Control
   When adjusting the rotation speed, ad_vr.avedat is converted to 8-bit resolution, and the values are interpreted as follows:
   - Less than 0x10 : 0 Hz
   - 0xF0 or higher : 200 Hz (maximum speed)
   - 0x10 to 0xEF : cHz_MIN to 199 Hz

9.1.4. LED Display Control (led_display)

LED5 lighting will be controlled according to the EMG status. For further details, please refer to Section 4.6 User Interface.

9.1.5. UART Transmission Data Settings (UartDrv_putc)

Initial display data settings will be configured.
The communication settings shall be 115200 bps, 8-bit data, 1 stop bit, no parity, and no flow control.

9.1.6. Terminal Software Display

It is possible to display text strings on terminal software such as Tera Term.

- TeraTerm Communication Settings:
  115200 bps, 8 data bits, 1 stop bit, no parity, no flow control

- Display Content:
  The following will be displayed once after the reset is released:
  Hello!
  SBK-M375

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 below:

10.1.1. Control instruction (usr.com_user)

- Motor start and stop
- Enable or disable of the encoder.
- Modulation type (2-phase modulation or 3-phase modulation)
- Shift PWM ON/OFF (Effective only when using VE with one shunt, and two-phase modulation.)

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

```
command_t        com_user;
```

In the application software stage, “usr.com_user” is set as a control instruction.

10.1.2. Control Target Speed (usr.omega_user)

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

In the application software stage, “usr.omega_user” is set as a control target speed.

10.1.3. Starting Current (usr.Id_st_user, usr.Iq_st_user)

```
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 */
```

In the application software stage, “usr.Id_st_user” and “usr.Iq_st_user” are set as start-up current instructions.

10.2. Drive Command

Drive Commands are listed as below:

10.2.1. Drive Type (drv.command)

```
command_t        command;
```

In the motor control stage, the drive type is transferred to the motor control driver through “drv.command”.

10.2.2. Vector Control Command (drv.vector_cmd)

The instruction commands of the vector control calculation manage the processes per stage.

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

The vector control drive command (drv.vector_cmd) is set in each stage as follows, and it is instructed to the motor drive stage.
“F_vcomm_volt” and “F_vcomm_Idetect” are not used for the vector control which uses the VE.

Table 10.1 Vector Control Command

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

1) F_vcomm_theta
“SET” sets the estimation value to the rotor position in the rotor position estimation calculation. “CLEAR” sets the instructed value to the rotor position.

2) F_vcomm_omega
“SET” sets the estimation value to the speed Ω in the rotor position estimation calculation. “CLEAR” sets the instructed value to the speed Ω.

3) F_vcomm_current
It instructs the calculation type for the reference values of the d-axis current and the q-axis current in the current control process.
“SET” sets the value which is acquired by the PI control using the speed deviation value to the reference value. “CLEAR” sets the instructed value to the reference value without executing the PI control.

4) F_vcomm_volt
It instructs the calculation type for the reference values of the d-axis voltage and the q-axis voltage in the current control process.
“SET” sets the value which is acquired by the PI control using the current deviation value to the reference value. “CLEAR” sets the instructed value to the reference value without executing the PI control.

5) F_vcomm_Edetect
“SET” executes the calculation of the induced voltage and the estimation of the rotor position. “CLEAR” sets the induced voltage to 0 and the instructed value to the rotor position without executing the calculation of the induced voltage.

6) F_vcomm_Idetect
“SET” executes the input coordinate conversion. “CLEAR” clears the value without the execution.

7) F_vcomm_onoff
“SET” executes the motor output waveform ON. “CLEAR” executes the motor output waveform OFF.

10.3. Driver Status

10.3.1. Error Status (drv.state)

```
typedef union {
  struct {
        uint16_t reserve:10;        /* reserve */
        uint16_t unreached:1;       /* 0:normal, 1: Unreached error */
        uint16_t Loss_sync: 1;      /* 0:normal, 1: Loss of synchronism      */
        uint16_t emg_DC:1;          /* 0:normal, 1: Over Vdc                 */
        uint16_t emg_I:1;           /* 0:normal, 1: Current detect error     */
        uint16_t emg_S:1;           /* 0:normal, 1: Over current(soft)       */
        uint16_t emg_H:1;           /* 0:normal, 1: Over current(hard)       */
  } flg;
  uint16_tall;
  } state_t;
```

(1) Loss_sync Detection of loss of synchronism
It is set when loss of synchronism is detected (unimplemented).

(2) emg_DC Abnormal voltage detection
It is set when abnormal voltage is detected.

(3) emg_I Abnormal current detection
It is set when abnormal current is detected (unimplemented).

(4) emg_S Software overcurrent detection
It is set when overcurrent is detected by the corresponding software.

(5) emg_H Hardware overcurrent detection
It is set when overcurrent is detected by a hardware in a microcontroller.

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 */
```

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 instructed 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 instructed Value	Q31	
q31_u	drv.Iq_com	q-Axis Current instructed Value	Q31	
q15_t	drv.Id_ref	d-Axis Current reference Value	Q15	
q15_t	drv.Iq_ref	q-Axis Current reference 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 integral Value	Q31	
uint16_t	drv.theta_com	Electrical Angle instructed 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_u	drv.Vd	d-Axis Voltage	Q31	
q31_u	drv.Vq	q-Axis Voltage	Q31	
q15_t	drv.Vdq	dq-Axis Voltage	---	
q31_u	drv.Vdq_ave	dq-Axis Voltage average value	---	
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	---	
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	---	(See 10.2.1)
vectorcmd_t	drv.vector_cmd	Vector Control Command	---	(See10.2.2)
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	

q31_u	drv.Iao_ave	Phase-a Zero Current Average AD	Q0	
q31_u	drv.Ibo_ave	Phase-b Zero Current Average AD	Q0	
q31_u	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 Value	Q0	
int16_t	drv.EnCnt1	Encoder Counter Previous Value	Q0	
int16_t	drv.EnCnt_dev	Encoder Counter Deviation	Q0	
q31_u	usr.omega_user	Controller Target Speed	Q31	
q15_t	usr.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	---	(See 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 Switching Speed	Q15	
q15_t	para.spwm_threshold	PWM shift switching speed	Q15	1-shunt only
q31_t	para.vd_pos	Positioning command Voltage	Q31	
q31_u	drv.Valpha	α-axis voltage	Q31	
q31_u	drv.Vbeta	β-axis voltage	Q31	
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 command value	Q31	
q31_u	drv.Vq_com	q-axis voltage command value	Q31	
q31_u	drv.Vd_I	d-axis voltage integral value	Q31	
q31_u	drv.Vq_I	q-axis voltage integral value	Q31	
q31_u	drv.Valpha	α-axis voltage	Q31	
q31_u	drv.Vbeta	β-axis voltage	Q31	
q31_u	drv.Valpha_pre	Previous α-axis voltage	Q31	
q31_u	drv.Vbeta_pre	Previous β-axis voltage	Q31	
q31_t	para.spd_coef	Output Voltage Factor	Q15	
q31_u	para.sp_ud_lim_f	Drive Speed Increase/Decrease Limit	Q31	
q31_u	para.sp_up_lim_s	Drive Speed Increase Limit	Q31	
q31_u	para.sp_dn_lim_s	Drive Speed Decrease Limit	Q31	
uint16_t	para.time.initpos	Positioning Time	Q0	
uint16_t	para.time.initpos2	Positioning Status Standby Time	Q0	

uint16_t	para.time.bootstp	Bootstrap Time	Q0	
uint16_t	para.time.go_up	Standby Time after Change Up	Q0	
q31_t	para.iq_lim	q-Axis Current Limit	Q31	
q31_t	para.id_lim	d-Axis Current Limit	Q31	
q15_t	para.err_ovc	Overcurrent Setup Value	Q15	
q31_t	para.pos.kp	Position Estimation Proportional Gain	Q15	Supports Q12
q31_t	para.pos.ki	Position Estimation Integration Gain	Q15	Supports Q12
int32_t	para.pos.ctrlprd	Position Estimation Control Cycle	Q16	
q31_t	para.spd.kp	Speed Control Proportional Gain	Q15	Supports Q12
q31_t	para.spd.ki	Speed Control Integration Gain	Q15	Supports Q12
uint8_t	para.spd.pi_prd	(Not Used)	Q0	
q31_t	para.crt.dkp	d-Axis Current Control Proportional Gain	Q15	Supports Q12
q31_t	para.crt.dki	d-Axis Current Control Integration Gain	Q15	Supports Q12
q31_t	para.crt.qkp	q-Axis Current Control Proportional Gain	Q15	Supports Q12
q31_t	para.crt.qki	q-Axis Current Control Integration Gain	Q15	Supports Q12
q31_t	para.motor.r	Motor Winding Resistance	Q15	
q31_t	para.motor.Lq	Motor q-Axis Inductance	Q15	
q31_t	para.motor.Ld	Motor d-Axis Inductance	Q15	
uint32_t	para.enc.pls2theta	Computation factor from the number of pulses to the angle	Q32	
q15_t	para.enc.pls2omega	Computation factor from the number of pulses to the speed	Q15	
int32_t	para.enc.plsnum	Number of encoder pulses	Q0	
int32_t	para.enc.ctrlprd	Encoder Control Cycle	Q16	
uint32_t	para.enc.deg_adjust	Electrical Angle Adjustment	Q0	
int32_t	para.delta_lambda	Current changeover phase when changing up	Q0	
uint16_t	para.chkpls	Current Detection Protection Duty Width	Q0	
uint32_t	stage_counter	Stage Counter	Q0	
shunt_type_e	shunt_type	Shunt Type	---	c1shunt c3shunt
boot_type_e	boot_type	Starting Type	---	cBoot_i cBoot_v
q15_t	drv.DutyU	U-phase PWM duty ratio	Q15	
q15_t	drv.DutyV	V-phase PWM duty ratio	Q15	
q15_t	drv.DutyW	W-phase PWM duty ratio	Q15	
q15_t	drv.Vdq_dc_rate	Voltage utilization	Q15	

10.5. User Function

This section explains the functions used in user processing, motor control, and motor drive operations. The user processing is implemented through the following functions, which are called from the main function and within the main loop.

10.5.1. Motor Control Initial Setup (B_Motor_Init)

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

Table 10.3 List of Motor Control Initial Setup Variables

Name	Description	Q Format	Remarks
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	
usr.Iq_st_user	Starting Iq-Current	Q15	
usr.Id_st_user	Starting Id-Current	Q15	
usr.lambda_user	Initial Rotor Position	Q0	
usr.com_user.modul	Modulation Method	---	
usr.com_user.spwm	Shift PWM ON/OFF	---	
para.motor.r	Motor Winding Resistance	Q15	
para.motor.Lq	Motor q-Axis Inductance	Q15	
para.motor.Ld	Motor d-Axis Inductance	Q15	
para.spwm_threshold	Shift PWM Changeover Speed	Q15	
para.chkpls	Current Detection Protection Duty Width	Q0	
para.vd_pos	Positioning command Voltage	Q31	Voltage Start only
para.spd_coef	Output Voltage Factor	Q15	Voltage Start only
para.sp_ud_lim_f	Drive Speed Increase/Decrease Limit	Q31	
para.sp_up_lim_s	Drive Speed Increase Limit	Q31	
para.sp_dn_lim_s	Drive Speed Decrease Limit	Q31	
para.time.bootstp	Bootstrap Time	Q0	
para.time.initpos	Positioning Time	Q0	
para.time.initpos2	Positioning Status Standby Time	Q0	
para.time.go_up	Standby Time after Change Up	Q0	
para.omega_min	Forced Commutation End Speed	Q15	
para.omega_v2i	Current Control Changeover Speed	Q15	
para.delta_lambda	Current changeover phase when changing up	Q0	
para.pos.ki	Position Estimation Integration Gain	Q15	
para.pos.kp	Position Estimation Proportional Gain	Q15	
para.pos.ctrlprd	Position Estimation Control Cycle	Q16	
para.spd.ki	Speed Control Integration Gain	Q15	
para.spd.kp	Speed Control Proportional Gain	Q15	
para.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	When using the encoder
para.enc.deg_adjust	Electrical Angle Adjustment	Q0	When using the encoder
para.enc.plsnum	Number of encoder pulses	Q0	When using the encoder
para.enc.width2omega	Computation Coefficient from Pulse Width Duration to Velocity	Q15	When using the encoder
para.enc.pls2omega	Computation factor from the number of pulses to the speed	Q15	When using the encoder
para.enc.ctrlprd	Encoder Control Cycle	Q16	When using the encoder

10.5.2. User Motor Control (B_User_MotorControl)

Setup Motor ON/OFF, rotating speed, driving method, etc.
Check overcurrent (hardware EMG) status.

Table 10.4 List of User Motor Control Variables

Name	Description	Q Format	Remarks
usr.omega_user	Controller Target Speed	Q31	
usr.com_user.onoff	Motor ON/OFF	---	
drv.state	Motor Abnormality Status	---	

10.5.3. User Control (user_control)

- TSW1 input
The rotation direction of a motor is switched.
If the rotation direction is switched while the motor is rotating, the motor stops.

- VR (Volume resistor)
The setting of the motor speed instructed value is done.
The rotational speed can be set within the range of 12 Hz to 200 Hz. The values can be adjusted by modifying cSPEED_ACT_MIN and cSPEED_ACT_MAX.

- LED output
The LED5 lights during the emergency state. The LED5 turns off after the emergency state is dissolved.

10.6. Motor Control Function

The motor control process is realized by the functions which are called from the main loop in the “main” function. The motor operation state is selected from among the stop state, the bootstrap state, the position decision state, the forced commutation state, the forced steady switch state, the steady state, the short-circuit brake state, and the protection states. In the main stage, the operation proceeds in the order of the position decision (Initposition), the forced commutation (Force), the forced steady switch (Change-up), and the steady (Steady_A). The sub stage has the steps from Step0 to StepEnd. The main stage starts with Step0 after StepEnd. When an abnormal motor operation is detected, the state transits to the protection stop emergency state.

10.6.1. Status Transfer Processing Function (C_Control_Ref_Model)

This function checks the control command given by the application software and the present status. Then it executes a proper state transition. Each state is divided into ramified sub states. The transition among the sub states is executed in the process function of each state, not in the state transition process function.

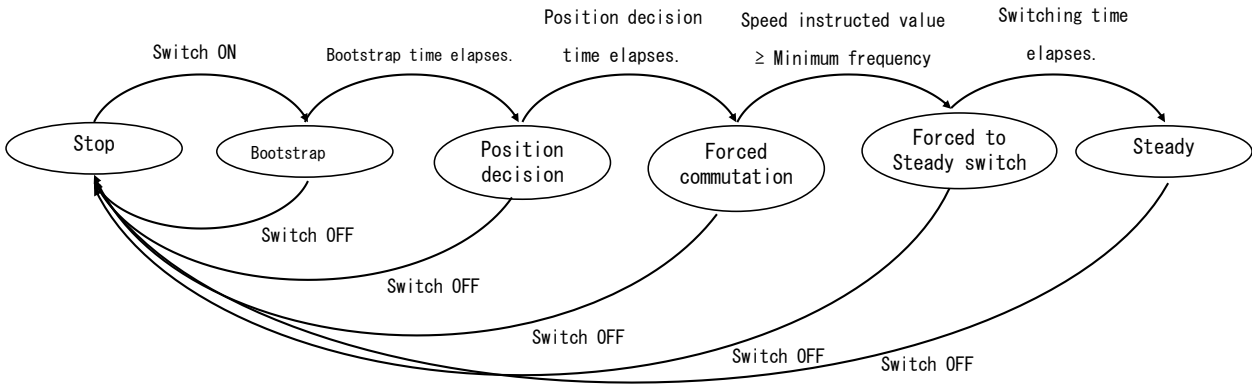


Figure 10.1 Motor Control Status Transition Diagram (Sensor less Control)

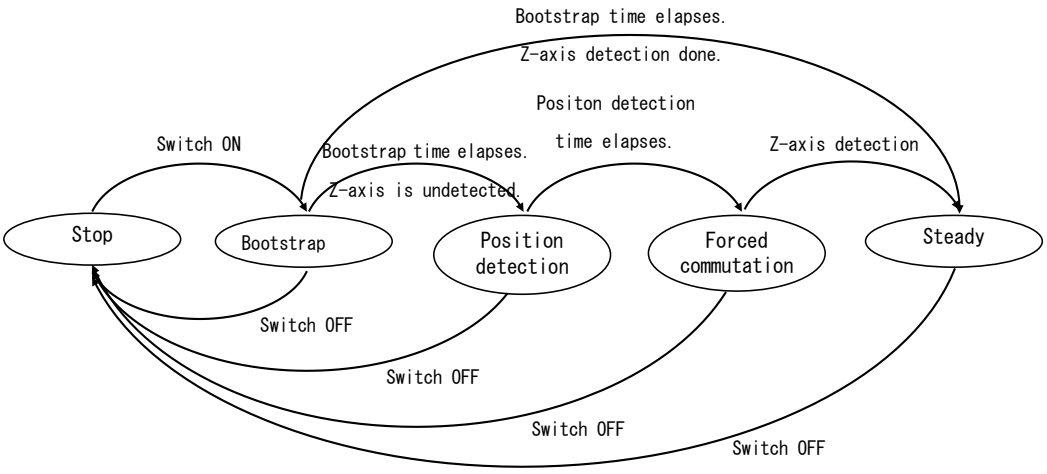


Figure 10.2 Motor Control Status Transition Diagram (Encoder control)

10.6.2. Motor Control Common Processing Function (C_Common)

Common process to each state of the motor control is executed.

10.6.3. Stop Stage Function (C_Stage_Stop)

Stops the motor. (Stops the PWM output)

10.6.4. Bootstrap Stage Function (C_Stage_Bootstrap)

This function outputs the upper phase all OFF signal and the lower phase all ON signal to charge the bootstrap capacitor.
This process continues for the interval of “Bootstrap time”. The charge amount in the capacitor depends on the Bootstrap time.

The bootstrap state is controlled by being divided into the following sub states.
a) Initial state
The initial setting of the bootstrap state is done.

b) Time lapse waiting state
After a specified bootstrap time elapses, the state is transited to the position decision state.

Table 10.5 List of Bootstrap Stage Function Variables

Name	Description	Q Format	Remarks
para.time.bootstp	Bootstrap Time	Q0	* MainLoopPrd(s)
stage_counter	Stage Counter	Q0	* MainLoopPrd(s)

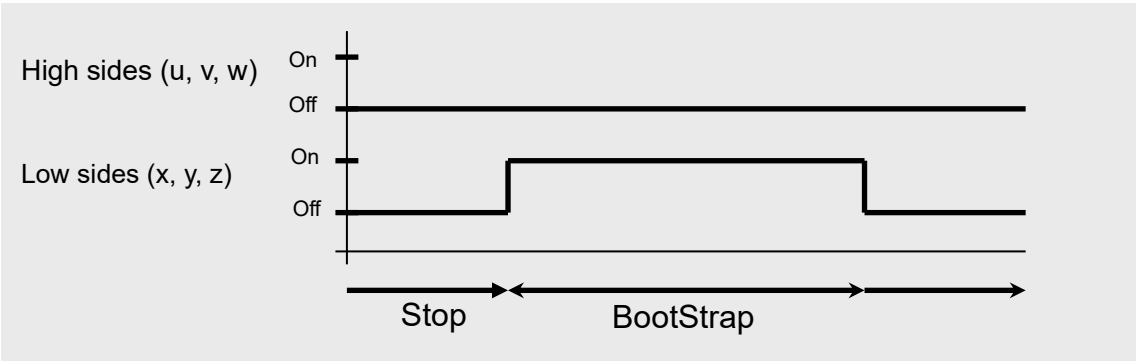


Figure 10.3 Bootstrap Time

10.6.5. Positioning Stage Function (C-Stage_Initposition)

The rotor is fixed to the initial position.
θ is fixed to the value of “the initial position”. Ω and Iq are fixed to 0. And Id is increased gradually from 0. This process continues for the interval of “the position decision time”. The last value of the Id is “the start Id current value”. The rise amount per unit time is decided using “the position decision time” and “the start Id current value”. The Id value reaches “the start Id current value”. Then, after “the position decision wait time” elapses, the state is transited to the next stage.

The position decision state is controlled by being divided into the following sub states.

a) Initial state
The initial setting of the position decision state is done.

b) Id increase state
Id is increased gradually to the setting value.

Table 10.6 List of Positioning Stage Function Variables

Name	Description	Q Format	Remarks
drv.vector_cmd	Drive command	---	
boot_type	Start-up type	---	
stage_counter	Stage counter	Q0	* MainLoopPrd(s)
usr.Id_st_user	Start Id current value	Q15	
usr.lambda_user	Initial rotor position	Q0	
drv.Id_com	d-axis current instructed value	Q31	
drv.Iq_com	q-axis current instructed value	Q31	
drv.omega_com	Frequency instructed value	Q31	
drv.theta_com	Electrical angle instructed value	Q0	
drv.Vd_out	Output voltage value	Q31	
para.time.initpos	Position decision time	Q0	* MainLoopPrd(s)
para.time.initpos2	Position decision waiting time	Q0	
para.vd_pos	Position decision instructed voltage	Q31	

10.6.6. Forced Commutation Stage Function (C_Stage_Force)

The rotation of the rotor starts. Rotation magnetic field is given forcibly to the rotor in this stage, and the rotor rotates according to the magnetic field.
Id and Iq are fixed to “the start Id current value” and 0, respectively. The drive speed instructed value Ωcom is increased gradually. θ is calculated with Ωcom (θ = Ωcom x t). This process continues till Ω reaches “the forced commutation end speed”.

“The drive target speed” is increased per constant value to increase the rotor speed to “the control target speed”. “The target speed” is Ω.

Table 10.7 List of Forced Commutation Stage Function Variables

Name	Description	Q Format	Remarks
drv.vector_cmd	Drive command	---	
boot_type	Drive type	---	Current or Voltage

usr.Id_st_user	Start Id current value	Q15	
usr.omega_user	Control target speed	Q31	
drv.Id_com	d-axis current instructed value	Q31	
drv.Iq_com	q-axis current instructed value	Q31	
drv.omega_com	Drive speed instructed value	Q31	
drv.Vd_out	Output voltage value	Q31	Valid only when Voltage drive is done.
para.sp_ud_lim_f	Drive speed increase and decrease limit values	Q31	
para.omega_min	Forced control end speed	Q15	
para.omega_v2i	Current control switch speed	Q15	Valid only when Voltage drive is done.
para.spd_coef	Output voltage coefficient	Q15	Valid only when Voltage drive is done.
para.vd_pos	Position decision output voltage	Q31	Valid only when Voltage drive is done.

10.6.7. Forced Steady Changeover Stage Function (C_Stage_Change_up)

The direction of the magnetic field is controlled to be perpendicular to the direction of the rotor by decreasing the Id to 0 and increasing the Iq to “the start Iq current value”. This generates a torque element. Ω and θ are acquired with the position estimation calculation.
The 'target drive speed' is gradually increased in fixed increments to approach the 'control target speed'. “The drive target speed” is not used because the speed control is not done in this stage.

The forced steady switch state is controlled by being divided into the following sub states.

a) Initial state

The initial setting of the force steady switch state is done.

b) Id and Iq switch state

The Id is decreased gradually to 0, and at the same time the Iq is increased gradually to the setting value. The decrease or the increase rate is not linear but trigonometric. After the switch completes, the state is transited to the time lapse wait state.

c) Time lapse wait state

After the specified forced steady switch time elapses, the state is transited to the steady state.

Table 10.8 List of Forced Steady Changeover Stage Function Variables

Name	Description	Q Format	Remarks
drv.vector_cmd	Drive command	Q0	
stage_counter	Stage counter	Q0	ms
usr.Id_st_user	Start Id current value	Q15	
usr.Iq_st_user	Start Iq current value	Q15	
usr.omega_user	Control target speed	Q31	
drv.Id_com	d-axis current instructed value	Q31	
drv.Iq_com	q-axis current instructed value	Q31	
drv.omega_com	Drive speed instructed value	Q31	
para.time.go_up	Wait time after Change-up	Q0	
para.sp_ud_lim_f	Drive speed increase and decrease limit values	Q31	

10.6.8. Steady Stage Function (C_Stage_Steady_A)

The steady state process is executed.
“The drive target speed” is increased per constant value to reach “the control target speed”.

Table 10.9 List of Steady Stage Function Variables

Code	Description	Q Format	Remarks
drv.vector_cmd	Drive command	Q0	
usr.omega_user	Control target speed	Q31	
usr.lambda_user	Initial rotor position	Q0	At Hall sensor + Encoder control
drv.Id_com	d-axis current instructed value	Q31	
drv.omega_com	Drive speed instructed value	Q31	At the speed control
para. sp_up_lim_s	Drive speed increase limit value	Q31	
para. sp_dn_lim_s	Drive speed decrease limit value	Q31	

10.6.9. Protection Stage Function (C_Stage_Emergency)

When overcurrent occurs, the state is transited to this state.
When hardware overcurrent is detected, all the motor drive outputs, u, v, w, x, y, and z become Hi-z.
When software overcurrent is detected, all the motor drive outputs, u, v, w, x, y, and z become OFF.
The rotor freewheels.
This stage continues until the process of the overcurrent state resumption is executed.

10.6.10. Shift PWM Control (C_ShiftPWM_Control)

This control is valid only for the 1-shunt circuit.
The ON or OFF setting of the shift PWM is done and the minimum duty value is set to use the previous current detection value.
The shift PWM drive type is decided by the shift PWM control type, the interrupt stage, and the target speed. This sample software has the conditions inTable 10.10 Shift PWM Drive Method Conditions.

Table 10.10 Shift PWM Drive Method Conditions

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

10.7. Motor Drive Function

10.7.1. Explanation of Terms

10.7.1.1. 3-phase Modulation

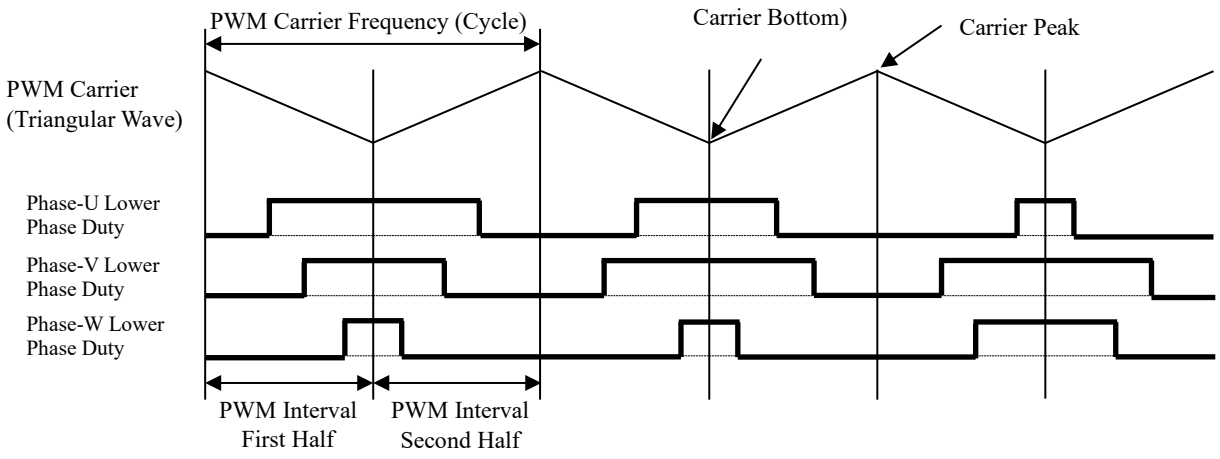


Figure 10.4 3-phase Modulation

10.7.1.2. 2-phase Modulation

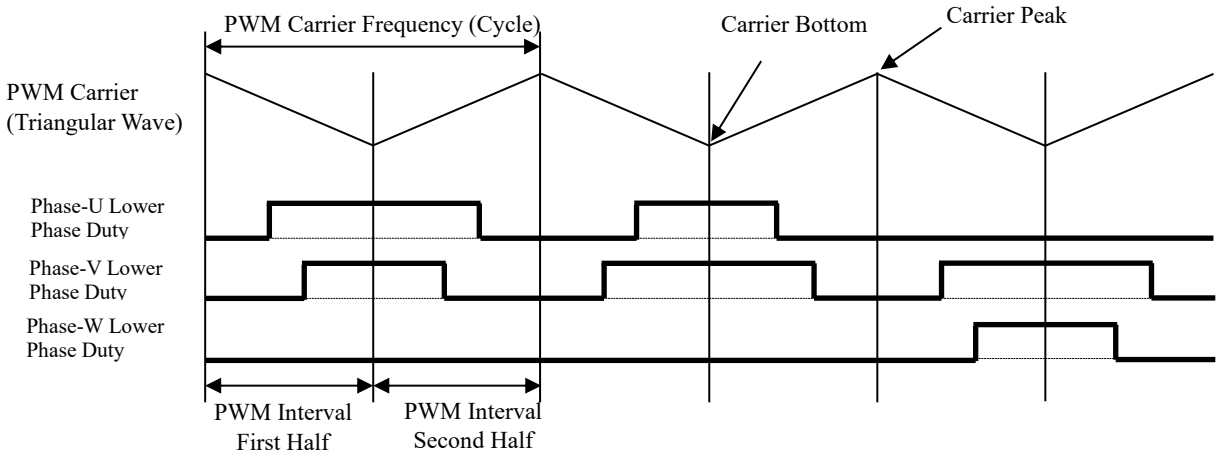


Figure 10.5 2-phase Modulation

10.7.2. Position Estimation Function (D_Detect_Rotor_Position)

The PI control is executed by regarding the estimate speed Ω_{est} 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 Ω_{est} acquired through the PI control.

The equivalent circuit equation regarding the d-axis of motor is expressed as below:

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

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

Vd	: Motor Applied Voltage
Id,Iq	: Motor Current
Ω_{est}	: Estimated Angular Speed
R	: Resistance
Ld,Lq	: Inductance

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

$$\mathrm{Ed} = \mathrm{Vd} - \mathrm{R} \cdot \mathrm{Id} + \Omega\mathrm{est} \cdot \mathrm{Lq} \cdot \mathrm{Iq}$$

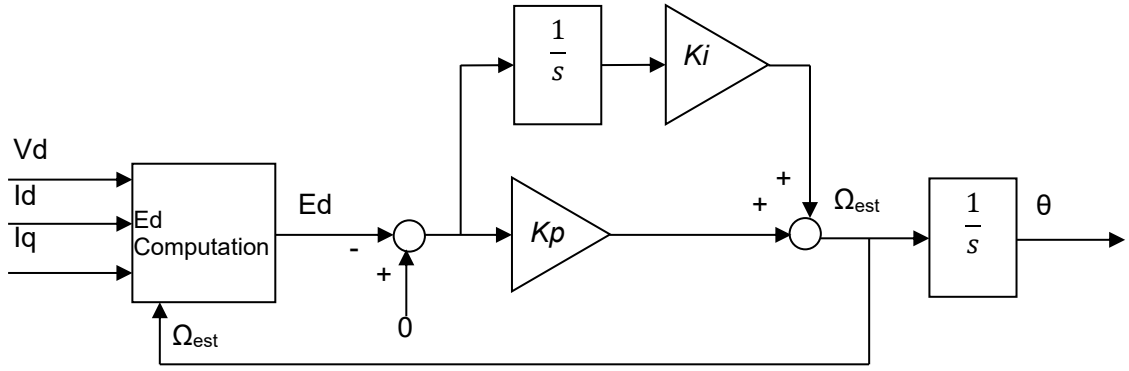


Figure 10.6 Position Estimation Block Diagram based on Induced Voltage Ed of the d-Axis

Table 10.11 List of Position Estimation Function Variables

Name	Description	Q Format	Remarks
drv.Ed	d-axis induced voltage	Q15	
drv.Vd	d-axis voltage	Q31	
drv.Id	d-axis current	Q15	
drv.Iq	q-axis current	Q15	
drv.omega_com	Drive speed instructed value	Q31	
drv.omega	Estimated speed	Q31	
drv.theta	Rotor position	Q31	
drv.Ed_I	d-axis induced voltage integral value	Q31	
drv.Ed_PI	d-axis induced voltage PI value	Q31	
para.motor.r	Motor winding resistance	Q15	
para.motor.Lq	Motor q-axis inductance	Q15	
para.pos.kp	Position estimation proportional gain	Q15	
para.pos.ki	Position estimation integral gain	Q15	

10.7.3. Encoder Control Function (H_Encoder)

The count of the incremental encoder pulses is read. The rotor position (angle) θ and the speed Ω are calculated using the count value.

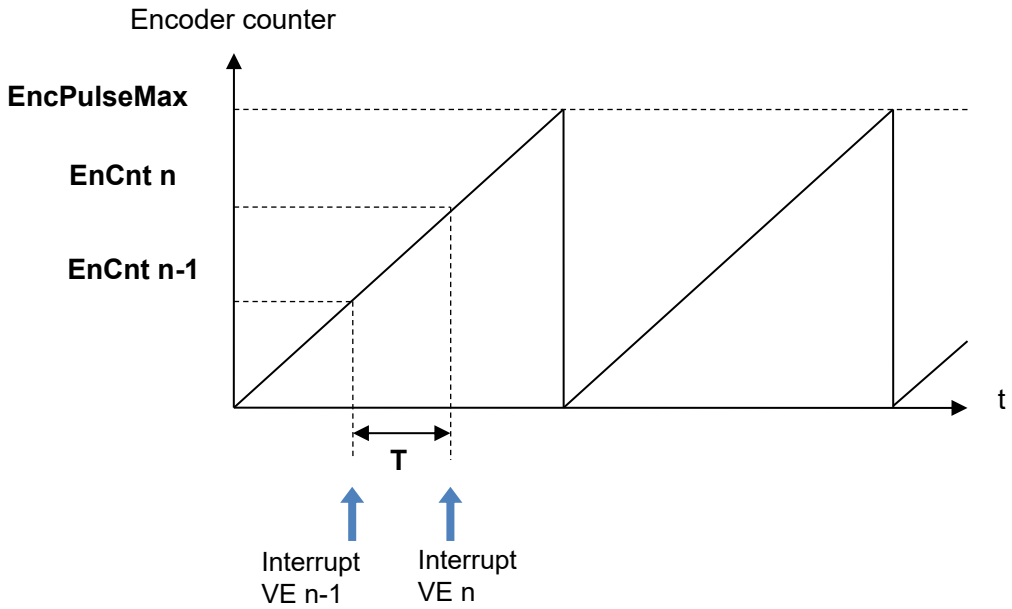


Figure 10.7 Computation of Position and Speed using Encoder Pulse

The pulse count is read at every VE interrupt, and the following calculation is done.

θ is calculated with the following formula using the current encoder pulse count.

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

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).

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

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]

In the M375 system, the speed Ω is calculated from the encoder pulse interval time using the following formula.

$$\Omega = (\mathrm{Pole} \div 2) \div [\{(\mathrm{PulseTimerDiv} \div \mathrm{PreFreq}) \times \mathrm{CapPulseCnt}\} \times (\mathrm{EncPulseMax} \div \mathrm{EncPulseDiv})]$$

CapPulseCnt	Encoder pulse interval measurement capture count
EncPulseDiv	Encoder pulse signal input division ratio
PulseTimerDiv	Encoder pulse measurement timer division ratio
PreFreq	Prescaler clock [Hz]

10.7.4. Speed Control Function (D_Control_Speed)

10.7.4.1. 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 measured speed.

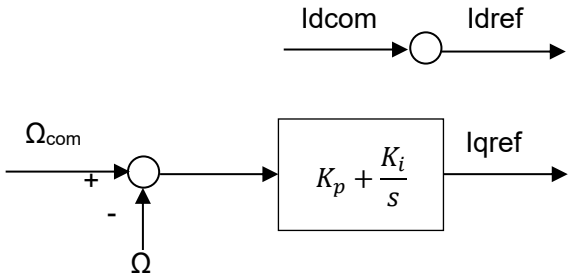


Figure 10.8 Speed Control Block Diagram

Table 10.12 List of Speed Control Function Variables

Name	Description	Q Format	Remarks
drv.omega_com	Drive speed instructed value	Q31	
drv.omega	Estimated speed	Q31	
drv.omega_dev	Speed deviation	Q15	
drv.Id_com	d-axis current instructed value	Q31	
drv.Id_ref	d-axis current reference value	Q15	
drv.Iq_ref	q-axis current reference value	Q15	
drv.Iq_ref_I	q-axis current integral value	Q31	
para.spd.kp	Speed control proportional gain	Q15	
para.spd.ki	Speed control integral gain	Q15	
para.id_lim	d-axis current limit value	Q31	
para.iq_lim	q-axis current limit value	Q31	

10.7.5. 3shunt Current Detection Error Detection Function (D_Check_DetectCurrentError_3shunt)

10.7.5.1. Syntax

int D_Check_DetectCurrentError_3shunt (uint32_t _duty, uint16_t _chkplswidth)

Parameter:

_duty	Intermediate value of Duty
_chkplswidth	Pulse Check Width

Returned Value:

Current detection state

10.7.5.2. Variable

Table 10.13 List of 3shunt Current Detection Error Detection Function Variables

Name	Description	Q Format	Remarks
_duty	Intermediate PWM Duty Ratio	Q15	0.0 to 1.0
_chkplswidth	Pulse Check Width	---	
---	Current detection state	---	0:Detectable 1:Undetectable

10.7.5.3. Process

This detects timing when current cannot be detected due to the PWM duty width. It also detects if the midpoint of the duty exceeds the set value. By confirming that the duty width is one at which current cannot be detected in this function, and using the previous value instead of the detected value in the current detection process, it eliminates control disturbances caused by erroneous current detection.

Since the AD conversion value of the current detection for the PWM maximum phase is not used in calculations, if the current detection value of the PWM intermediate phase exceeds the set value, it will be judged as a detection NG.

10.7.6. 1shunt Current Detection Error Detection Function (D_Check_DetectCurrentError_1shunt)

10.7.6.1. Syntax

```
int D_Check_DetectCurrentError_1shunt    (uint32_t _pwma, uint32_t _pwmb,
                                          uint32_t _pwmc, uint16_t _chkplswidth,
                                          int _sfhpwm, int _modu, int _sector,
                                          uint32_t _mdprd, uint16_t _trgcmp0,
                                          uint16_t _trgcmp1, uint16_t _shift2)
```

Parameter:

_pwma	U-phase PWM duty ratio
_pwmb	V-phase PWM duty ratio
_pwmc	W-phase PWM duty ratio
_chkplswidth	Pulse check width
_sfhpwm	Shift PWM mode status
_modu	Modulation method
_sector	Sector information
_mdprd	PWM period

Returned Value:

Current detection state

10.7.6.2. Variable

Table 10.14 List of 1shunt Current Detection Error Detection Function Variables

Name	Description	Q Format	Remarks
_pwma	U-phase PWM duty ratio	Q15	0.0 to 1.0
_pwmb	V-phase PWM duty ratio	Q15	0.0 to 1.0
_pwmc	W-phase PWM duty ratio	Q15	0.0 to 1.0
_chkplswidth	Pulse check width	---	
_sfhpwm	Shift PWM mode status	---	
_modu	Modulation method	---	2-phase or 3-phase modulation
_sector	Sector information	---	
_mdprd	PWM period	---	
---	U-phase PWM duty ratio	---	0:Detectable 1:Undetectable

10.7.6.3. Process

It detects the timing when the current cannot be detected due to the PWM duty width. It detects when the minimum duty width and the magnitude of the duty width difference become smaller than the set value. Note that the minimum duty width during two-phase modulation with one shunt becomes the intermediate duty width of the three-phase duty width.

Confirm that the duty width is one at which current cannot be detected with this function, and by not using the detected value in the current detection process and using the previous value instead, eliminate control disturbances caused by false current detection. If the width of the PWM duty itself and the difference in PWM duty width become smaller than the set values, it is judged as detection NG.

10.7.7. Function for Calculating Inverter Output Voltage (Cal_Vdq)

10.7.7.1. 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)

10.7.7.2. Variable

Table 10.15 List of Function for Calculating Inverter Output Voltage Variables

Direction	Name	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

10.7.7.3. Process

Calculating the inverter output voltage (Vdq).

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

11. Explanation of Definitions for Constants

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

11.1.1. Motor Control Channel Selection

__CONTROL_MOTOR_CH1 Define when controlling the CH1

11.1.2. Selecting DAC Output

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

11.1.3. Selection of Unit for Command Speed

__TGTSPD_UINT: Selection of unit for command speed
0: Hz of Electrical angle
1: RPS of mechanical angle
2: RPM of mechanical angle

11.1.4. LED/PC UART Output Selection

__LED_SW: Set this to "1" when using an LED.
__ PC_CH1: Set this to "1" when using PC UART.

11.1.5. Parameter List

Table 11.1 Parameter List

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

11.2. Motor Driver Setting Parameter Motor Channel (D_Para_chx.h) x = 1

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

11.2.1. Selection of Control

11.2.1.1. Selection of AMP

__USE_INAMP Define when the built-in AMP is used.

11.2.1.2. Choice of Sensor-less/Sensor

__USE_ENCODER Define when the encoder control is used.

11.2.2. Parameter List per Motor Channel

Table 11.2 Parameter List per Motor Channel

Parameter Name	Description
cPOLH	Upper Phase Drive Logical Setup
	0: Low active/ 1: High active
	Set up the logic for the upper phase driver.
cPOLL	Logical Setup for Lower Phase Driver
	0: Low active/ 1: High active
	Setup the logic for the lower phase driver.
cV_MAX	Setup of Max Voltage
	Unit [V]
	Setup the value of power supply voltage variation [A] x 2^12 regarding the variation of 1LSB in the AD conversion.
cA_MAX	Setup of Max Current
	Unit [A]
	Setup the value of phase current variation [A] x 2^11 regarding 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
	The current acquisition type should be set.
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. In cases such as during startup of shunt-driven systems where current cannot be obtained, a voltage-driven method is selected.
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 (for 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 (for 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 (for 3-shunt)

	AINx
	Set up the AD channel for detecting the Phase-W current
cADCH_CURRENT_IDC	Setup of Current Acquisition AD Channel (for 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 value. When a coil current exceeding this setup value is detected, the output will be turned OFF by the software.
cVDC_MINLIM	Setup of Min Power Supply Voltage Vdc
	Unit [V]
	Set up the Min value of power supply voltage Vdc. When voltage lower than this value is detected, the motor will be stopped.
cVDC_MAXLIM	Setup of Max Power Supply Voltage Vdc
	Unit [V]
	Set up the Max of power supply voltage Vdc. When voltage higher than this value is detected, the motor will be stopped.
cPWMPRD	Setup of PWM Cycle
	Unit [us] Resolution 25ns@80MHz
	Set up the cycle for the PWM carrier.
cDEADTIME	Setup of Dead Time
	Unit [us] Resolution 100ns@80MHz
	Set up the dead time.
cREPTIME	Setup of the number of skipped software control (Only when using VE).
	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	Command Speed Setting
	Unit [Hz] or [RPS] or [RPM] The unit depends on the setting of __TGTSPD_UNIT
	Sets the motor command speed. When SWx is turned ON, the motor operates at this set value.
cID_ST_USER_ACT	Setup of d-Axis Start Current
	Unit [A]
	Set up 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 q-axis start current.

	Set up a value of approximately 1/2 of d-axis start current. If the motor abruptly accelerates when transferring from the forced commutation (d-axis control) to the steady (q-axis control), set a smaller value and, if the motor stops, set a larger value.
cMOTOR_R	Motor Coil Resistance
	Unit [Ohm]
	Set up a resistance equivalent to that of the single phase of motor coil.
cMOTOR_LQ	q-Axis Inductance
	Unit [mH]
	Set up the inductance of q-axis of motor coil
cMOTOR_LD	d-Axis Inductance (not used)
	Unit [mH]
	Set up the inductance of d-axis of motor coil
cPOLE	Setup of the Number of Motor Poles
	Unit [poles]
	Set up the number of motor poles.
cID_KP	d-Axis Current Control Proportional Gain
	Unit [V/A]
	Set up the d-axis current control proportional gain.
cID_KI	d-Axis Current Control Integration Gain
	Unit [V/As]
	Set up the d-axis current control integration gain.
cIQ_KP	q-Axis Current Control Proportional Gain
	Unit [V/A]
	Set up the q-axis current control proportional gain.
cIQ_KI	q-Axis Current Control Integration Gain
	Unit [V/As]
	Set up the q-axis current control integration gain.
cPOSITION_KP	Position Estimation Proportional Gain
	Unit [Hz/V]
	Set up the position estimation proportional gain.
cPOSITION_KI	Position Estimation Integration Gain
	Unit [Hz/Vs]
	Set up the position estimation integration gain.
cSPEED_KP	Speed Control Proportional Gain
	Unit [A/Hz]
	Set up the speed control proportional gain.
cSPEED_KI	Speed Control Integration Gain
	Unit [A/Hz]
	Set up the speed control integration gain.
cSPD_PI_PRD	Setup of Speed PI Control Cycle
	[Time]
	Set up the speed PI control cycle. Setting to "1" will execute the speed PI computation once every PWM cycle. Setting to "2" will execute the speed PI computation once every two PWM cycles.
cFCD_UD_LIM	Drive Speed Acceleration Rate (Forced Commutation)
	Unit [Hz/s]

	Set up the speed acceleration rate during the forced commutation. Decrease the value when the motor speed does not follow the output of forced commutation.
cSTD_UP_LIM	Drive Speed Acceleration Rate (Steady)
	Unit [Hz/s]
	Set up the speed acceleration rate during the steady state.
cSTD_DW_LIM	Drive Speed Deceleration Rate (Steady)
	Unit [Hz/s]
	Set up the deceleration rate during the steady state.
cBOOT_LEN	Bootstrap Waveform Output Duration
	Unit [s] Resolution: 1 ms
	Set up the output duration for the bootstrap waveform.
cINIT_LEN	Positioning Time
	Unit [s] Resolution: 1 ms
	Set up the positioning duration.
cINIT_WAIT_LEN	Standby Time after Positioning
	Unit [s] Resolution: 1 ms
	Set up the standby time after positioning.
cGOUP_DELAY_LEN	Standby Time after Change Up
	Unit [s] Resolution: 1 ms
	Set up the standby time after change up.
cHZ_MAX	Max Frequency
	Unit [Hz]
	Set up the max frequency to be detected by the microcontroller. Set up a value increased by 10% to 20% than the max frequency used for controlling. A smaller value will enhance the precision of computation, but if the detected value exceeds this figure, the control will collapse.
cHZ_MIN	Changeover speed from forced commutation to steady.
	Unit [Hz]
	Set up a speed for transferring from the forced commutation to steady. Set up a speed that permits the position estimation (the induced voltage will be detected).
cHZ_SPWM	Shift PWM Switching Speed (for 1 Shunt 2-Phase Modulation)
	Unit [Hz]
	Set up the speed at which PWM switches from shift PWM to normal PWM. Switches at the value of command speed.
cMINPLS	Minimum Pulse Width Setting (for 1 Shunt)
	Unit [μs]
	When driving 1 shunt, set the PWM pulse width time for when the current cannot be obtained. If the PWM pulse width becomes less than this set value, the detected current value will not be used, and the previous value will be used for control.
cMAXPLS	Maximum Pulse Width Setting (for 3 Shunts)
	Unit [μs]
	When driving with 3 shunts, set the PWM pulse width time of the intermediate phase when the current cannot be obtained. If the PWM pulse width exceeds this set value, the detected current value will not be used, and control will be performed using the previous 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" See 11.2.3.2 Constants for Voltage Type Starting for details.
cSPD_COEF	Forced Commutation Output Voltage Factor during Voltage Drive
	Set a value x: 0 < x <1
	Determine the output voltage during the forced commutation. The multiplication of this setup and the reference speed is defined as the output voltage. Vd = cSPD_COEF × Omega_com Enabled only when the cBOOT_TYPE is "cBoot_v" See 11.2.3.2 Constants for Voltage Type Starting for details.
cHZ_V2I	Changeover Speed from Voltage Control to Current Control
	Unit [Hz]
	Set up a speed for changeover from the voltage control to the current control. Setting to a value larger than the cHZ_MIN will execute the transfer to a steady value equal to cHZ_MIN and over and will change over to the current control. Enabled only when the cBOOT_TYPE is "cBoot_v"
cENC_PULSE_NUM	Setup of the Number of Encoder Pulses
	Unit [ppr]
	Set up the number of pulses for the encoder. Small value for the number of encoder pulses will degrade the speed detection accuracy. If the detected speed is not stable, increase the filtering factor cOMEGAENC_AVE.
cENC_DEG_ADJUST	Encoder Position Adjustment
	Unit [deg.]
	Set up the shifted positional angle of phase-Z regarding the electrical angle 0°
__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"
cSPEED_ACT_MIN	Minimum Speed Command
	Unit [Hz] (electrical angle)
	Set up the minimum value of the VR speed control command.
cSPEED_ACT_MAX	Maximum Speed Command
	Unit [Hz] (electrical angle)
	Set up the maximum value of the VR speed control command.

11.2.3. Relationship of Constant Setups and Waveform

11.2.3.1. Constants for Current Type Starting

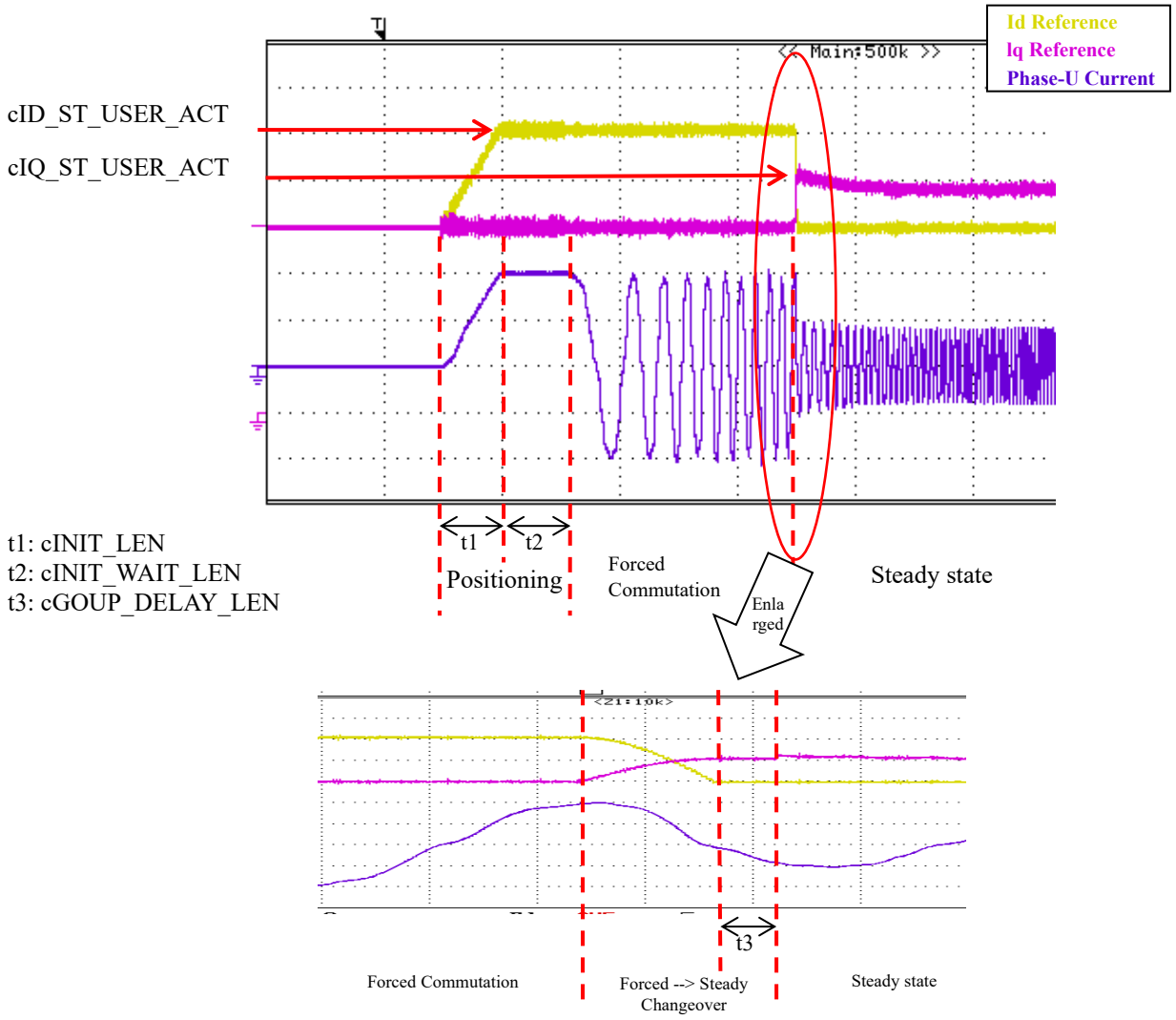


Figure 11.1 Starting Current Waveform (Positioned to electrical angle 0°)

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

11.2.3.2. Constants for Voltage Type Starting

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

The cVD_POS value should be adjusted to reach the target position decision current with viewing the current waveform. For safety, the adjustment should be done using the smallest value at start. Then the value should be increased. If the position decision current is smaller than the target one, the value should be increased. And if, larger, the value should be decreased.

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 commutation to be constant. If the current amplitude decreases, the value should be lager. If the current amplitude increases, the value should be smaller.

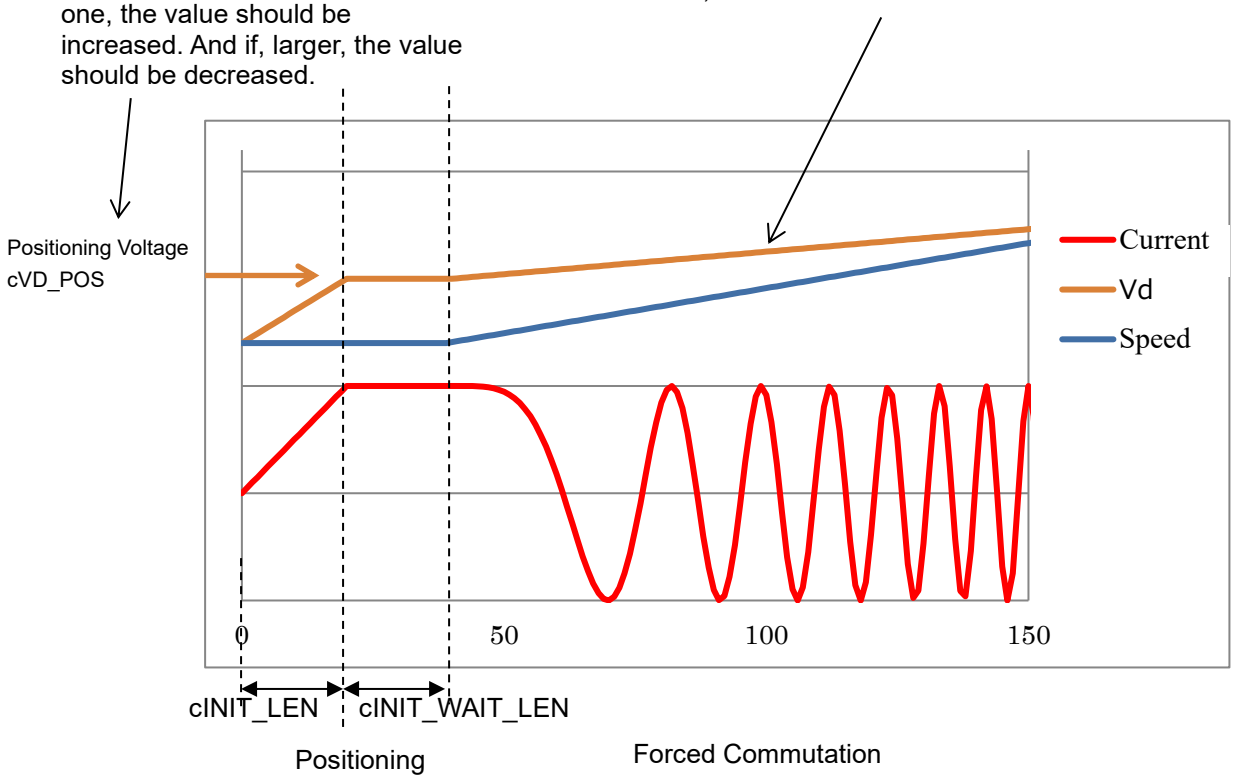


Figure 11.2 How to Adjust Parameters during Voltage Drive

11.3. Constants for User Control

```
usercon.c
/* Key settting */
#define P_TSW1                  TSB_PB_DATA_PB5     /* TSW1 */
#define P_SW1                   TSB_PB_DATA_PB6     /* SW1 */ Unused
#define cKEY_CHATA_CNT          (20)                /* [cnt] chattering counter for KEY SW */

/* Soft ADC Setting */
#define  cADUNIT_USR            TSB_ADB             /* User ad data ADCUnit */
#define cADCH_VR                ADC_AIN9            /* ADC Channel for VR */
#define cADREG_VR               ADC_REG7            /* Result register No for VR */
#define cADAVECNT               (10)                /* ADC average count */

/* LED setting */
#define P_LED5                  TSB_PF_DATA_PF0     /* LED5 */

/* UART setting */
#define SIOCH_PCCOM             TSB_SC1             /* UART Channel for PC COM */
```

12. Peripheral Driver

The driver is composed of the following files:
mcuip_drv.c
mcuip_drv.h

12.1. Vector Engine

12.1.1. Specifications of Functions

12.1.1.1. IP_VE_init

VE Initial Setup
API:
void IP_VE_init(TSB_VE_TypeDef* const VEx, VE_InitTypeDef* const _initdata)
Parameters:
VEx: Selection of VE address
_initdata: VE Initial Setup Data Structure
See 12.1.2.1 VE_InitTypeDef for details.
Function:
Initial Setup of VE
Supplement:
Call up during VE suspension with prohibited interruption.
Return Parameter:
No

12.1.1.2. VE_Start

VE Start
API:

```
void
VE_Start(TSB_VE_TypeDef* const    VEx);
```

Parameters:
Vex: Select VE address.
Function:
Starts VE.
Supplement:
None
Return Parameter:
No

12.1.1.3. VE_GetPhaseCurrent

Acquisition of Phase Current
API:

```
void
VE_GetPhaseCurrent(TSB_VE_TypeDef* const    VEx,
                   q15_t*    _ia, q15_t*    _ib, q15_t*    _ic);
```

Parameters:
VEx: Select VE address.
_ia: Setup of an address for the variable to store the phase-a current.
_ib: Setup of an address for the variable to store the phase-b current.
_ic: Setup of an address for the variable to store the phase-c current.
Function:
Acquires the current values for each phase.
The value of phase-U current is stored in phase-a, phase-V current in phase-b and phase-W current in phase-c.
Supplement:
None
Return Parameter:

No

12.1.1.4. VE_GetCurrentAdcData

Acquisition of Current AD Value

API:

```
void
VE_GetCurrentAdcData(TSB_VE_TypeDef* const    VEx,
                     uint32_t*    _adc_ia, uint32_t*    _adc_ib, uint32_t*    _adc_ic);
```

Parameters:

VEx: Select an IP table address.
_adc_ia: Setup of an address for a variable to store the AD value for the phase-a current.
_adc_ib: Setup of an address for a variable to store the AD value for the phase-b current.
_adc_ic: Setup of an address for a variable to store the AD value for the phase-c current.

Function:

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

Supplement:

None

Return Parameter:

No

12.1.1.5. VE_GetdataFromVEreg

Acquisition of VE Register Data

API:

```
void
VE_GetdataFromVEreg(TSB_VE_TypeDef* const    VEx,
                    vector_t* const    _motor)
```

Parameters:

VEx: Select VE address.
_motor: Selecting an address for the vector control variable structure

Function:

Stores the values; motor voltage Vdc, d-axis voltage Vd, q-axis voltage Vq, d-axis current Id, and q-axis current Iq from the VE register to each variable.
At the time when the current is not detectable due to the duty width, the previously detected values of d-axis current Id and q-axis current Iq will be written in the VE register.

Supplement:

None

Return Parameter:

No

12.1.1.6. VE_GetPWM_Period

Current AD value acquisition

API:

```
uint32_t
VE_GetPWM_Period(TSB_VE_TypeDef* const VEx)
```

Parameters:

VEx: VE address is selected.

Function:

The value in the PWM cycle register is acquired.

Supplement:

None

Return parameter:

PWM cycle

12.1.1.7. VE_GetPWM_DutyU

Acquisition of Phase-U Duty Value

API:
VE_GetPWM_DutyU(TSB_VE_TypeDef* const VEx)

Parameters:
VEx: Select VE address.
Function:
Acquires the value of phase-U duty register.
Supplement:
None

Return Parameter:
Phase-U Duty

12.1.1.8. VE_GetPWM_DutyV

Acquisition of Phase-V Duty Value
API:
uint32_t
VE_GetPWM_DutyV(TSB_VE_TypeDef* const VEx)

Parameters:
VEx: Select VE address.
Function:
Acquires the value of phase-V duty register.
Supplement:
None
Return Parameter:
Phase-V Duty

12.1.1.9. VE_GetPWM_DutyW

Acquisition of Phase-W Duty Value
API:
uint32_t
VE_GetPWM_DutyW(TSB_VE_TypeDef* const VEx)

Parameters:
VEx: Select VE address.
Function:
Acquires the value of phase-W duty register.
Supplement:
None
Return Parameter:
Phase-W Duty

12.1.1.10. VE_GetPWM_DutyMed

Acquisition of Duty Median
API:
uint32_t
VE_GetPWM_DutyMed(TSB_VE_TypeDef* const VEx)

Parameters:
VEx: Select VE address.
Function:
Acquires the duty medians for phases U, V, and W.
Supplement:
None
Return Parameter:
Duty Median

12.1.1.11. VE_GetPWM_Modulation

Acquisition of Modulation Type

API:

int
VE_GetPWM_Modulation(TSB_VE_TypeDef* const VEx)

Parameters:

VEx: Select VE address.

Function:

Acquires the modulation type.

Supplement:

None

Return Parameter:

Modulation Type 0: 3 phase modulation, 1: 2 phase modulation

12.1.1.12. VE_GetSector

Acquisition of Current Sector

API:

uint32_t VE_GetSector(const ipdrv_t* const _ipdrv)

Parameters:

_ipdrv: Select an IP table address.

Function:

Acquires the current sector value.

Supplement:

None

Return Parameter:

Current Sector Value [0 - 11]

12.1.1.13. VE_GetShiftPWMState

Acquisition of Shift PWM Status

API:

int
VE_GetShiftPWMState(TSB_VE_TypeDef* const VEx)

Parameters:

VEx: Select VE address.

Function:

Acquires the shift PWM status.

Supplement:

None

Return Parameter:

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

12.1.1.14. VE_GetOutPutMode

Acquisition of PWM Output Status

API:

int
VE_GetOutputMode(TSB_VE_TypeDef* const VEx)

Parameters:

VEx: Select VE address.

Function:

Acquires the PWM output status.

Supplement:

None

Return Parameter:

PWM Status OCRMD_OUT_OFF: Output OFF

OCRMD_OUT_ON: Output ON
OCRMD_OUT_ON_LOWPH: Only Lower Phase ON

12.1.1.15. VE_SetdataToVEreg_Stop

Data Setting to VE Register (Stop)

API:

```
void
VE_SetdataToVEreg_Stop(TSB_VE_TypeDef* const VEx,
                       const vector_t* const _motor);
```

Parameters:

VEx: Select VE address.
_motor: Setup of an address for the vector control variable structure.

Function:

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

Supplement:

None

Return Parameter:

No

12.1.1.16. VE_SetdataToVEreg_Bootstrap

Data Setting to VE Register (Bootstrap)

API:

```
void
VE_SetdataToVEreg_ Bootstrap(TSB_VE_TypeDef* const VEx,
                             const vector_t* const _motor);
```

Parameters:

VEx: Select VE address.
_motor: Setup of an address for the vector control variable structure.

Function:

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

Supplement:

None

Return Parameter:

No

12.1.1.17. VE_SetdataToVEreg_Initposition_i

Data Setup to VE Register (nitposition current control type)

API:

```
void
VE_SetdataToVEreg_ Initposition_i (TSB_VE_TypeDef* const VEx,
                                   const vector_t* const _motor);
```

Parameters:

VEx: Select VE address.
_motor: Setup of an address for the vector control variable structure.

Function:

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

Supplement:

None

Return Parameter:

No

12.1.1.18. VE_SetdataToVEreg_Initposition_v

Data Setup to VE Register (Initposition voltage control type)

API:

```
void
VE_SetdataToVEreg_ Initposition_v (TSB_VE_TypeDef* const VEx
                                   const vector_t* const _motor);
```

Parameters:

VEx: Select VE address.

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

Function:

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

Supplement:

None

Return Parameter:

No

12.1.1.19. VE_SetdataToVEreg_Force_i

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

API:

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

Parameters:

VEx: Select VE address.

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

Function:

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

Supplement:

None

Return Parameter:

No

12.1.1.20. VE_SetdataToVEreg_Force_v

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

API:

```
void
VE_SetdataToVEreg_Force_v(TSB_VE_TypeDef* const VEx,
                          const vector_t* const _motor);
```

Parameters:

VEx: Select VE address.

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

Function:

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

Supplement:

None

Return Parameter:

No

12.1.1.21. VE_SetdataToVEreg_Change_up

Data Setup to VE Register (change up)

API:

```
void
VE_SetdataToVEreg_ Change_up (TSB_VE_TypeDef* const VEx,
                              const vector_t* const _motor);
```

Parameters:

VEx: Select VE address.

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

Function:

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

Supplement:

None

Return Parameter:

No

12.1.1.22. VE_SetdataToVEreg_Steady_A

Data Setup to VE Register (Steady)

API:

```
void
VE_SetdataToVEreg_Steady_A (TSB_VE_TypeDef* const VEx,
const vector_t* const _motor);
```

Parameters:

VEx: Select VE address.

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

Function:

Processes the setup of steady status to the VE register.

Supplement:

None

Return Parameter:

No

12.1.1.23. VE_SetdataToVEreg_Emergency

Data Setup to VE Register (EMG)

API:

```
void
VE_SetdataToVEreg_ Emergency (TSB_VE_TypeDef* const VEx,
                              const vector_t* const _motor);
```

Parameters:

VEx: Select VE address.

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

Function:

Processes the setup of emergency status to the VE register.

Supplement:

None

Return Parameter:

No

12.1.1.24. VE_SetZeroCurrentData

Zero Current Setup

API:

```
void
VE_SetZeroCurrentData(TSB_VE_TypeDef* const   VEx,
                      uint32_t   _z_ia, uint32_t   _z_ib, uint32_t   _z_ic)
```

Parameters:

VEx: Select VE address.

_z_ia: Setup of the phase-a zero current AD value

_z_ib: Setup of the phase-b zero current AD value

_z_ic: Setup of the phase-c zero current AD value

Function:

Setup of zero current AD value to the VE register.

Supplement:

None

Return Parameter:
No

12.1.1.25. VE_SetVDCreg

Setup of Motor Voltage and Vdc

API:
void VE_SetVDCreg(TSB_VE_TypeDef* const VEx, q15_t _dat);

Parameters:
VEx: Select VE address.
_dat: Setup of motor voltage.

Function:
Setup of motor voltage to the VE register.

Supplement:
None

Return Parameter:
No

12.1.1.26. VE_SetSpwmMode

Shift PWM Mode Setting

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

Parameters:
VEx: Selection of VE address
_dat: Setup of shift PWM mode

Function:
Setup of shift PWM mode status to the VE register.

Supplement:
None

Return Parameter:
No

12.1.1.27. VE_SetModulType

Setup of Modulation Type

API:
void VE_SetModulType (TSB_VE_TypeDef* const VEx, uint8_t _dat);

Parameters:
VEx: Select VE address.
_dat: Setup of modulation type

Function:
Setup of modulation type to VE register.

Supplement:
None

Return Parameter:
No

12.1.2. Data Structure

12.1.2.1. VE_InitTypeDef

Data Fields:

uint8_t	**ve_ch:**	Vector Engine Channel cCH0: Channel 0 cCH1: Channel 1
uint8_t	**shunt:**	Shunt Type **3:** 3 shunt **1:** 1 shunt
uint16_t	**pwmfreq:**	PWM Frequency (for PWM output) Value to be set in the VEMDPRD register
uint16_t	**reptime:**	Repeat Frequency (1 to 15)
uint16_t	**trgmode:**	Start Trigger **TRGMODE_UNITA:** Starts with an interruption when the ADCA PMD0 trigger synchronized conversion is complete. **TRGMODE_UNITB:** Starts with an interruption when the ADCB PMD1 trigger synchronized conversion is complete.
uint16_t	**tpwm:**	PWM Cycle Time (phase interpolation) A value for setting to the VETPWM register.
uint16_t	**idkp:**	d-Axis Current Control Proportional Gain
uint16_t	**idki:**	d-Axis Current Control Integration Gain
uint16_t	**iqkp:**	q-Axis Current Control Proportional Gain
uint16_t	**iqki:**	q-Axis Current Control Integration Gain
uint16_t	**zerooffset:**	Zero Current Offset

12.2. Motor Control Circuit (PMD)

12.2.1. Specifications of Functions

12.2.1.1. IP_PMD_init

PMD Initial Setup

API:

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

Parameters:

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

Function:

Initial setup of PMD.

Supplement:

Call up during PMD suspension with prohibited interruption.

Return Parameter:

No

12.2.1.2. PMD_GetEMG_Status

Acquisition of EMG Protection Status

API:

emg_status_e
PMD_GetEMG_Status(TSB_PMD_TypeDef* const PMDx)

Parameters:

PMDx: Select PMD address.

Function:

Acquires the EMG protection status.

Supplement:

None

Return Parameter:

emg_status_e:	EMG Protection Status
cNormal:	Normal
cEMGProtected:	Protected

12.2.1.3. PMD_ReleaseEMG_Protection

EMG Protection Status Released

API:

void
PMD_ReleaseEMG_Protection(TSB_PMD_TypeDef* const PMDx)

Parameters:

PMDx: Select PMD address.

Function:

Releases the EMG protection status.

Supplement:

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

Return Parameter:

No

12.2.2. Data Structure

12.2.2.1. PMD_InitTypeDef

Data Fields:

uint8_t	**shunt:**	Shunt Type 3:3 shunt 1:1 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	**pwmfreq:**	PWM Frequency A value to be set up to PMDxMDPDR
uint16_t	**deadtime:**	Dead Time A value to be set up to PMDxDTR

12.3. Analogue Digital Converter (ADC)

12.3.1. Specifications of Functions

12.3.1.1. IP_ADC_init

ADC Initial Setup

API:

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

Parameters:

ADx: Select an ADC Address

_initdata: ADC Initial Setup Data Structure
See 12.3.2.1 AD_InitTypeDef for details.

Function:

Initial setup of ADC

Supplement:

Call up during ADC suspension with prohibited interruption.

Return Parameter:

No

12.3.1.2. IP_ADC_init2Unit

ADC Initial Setup

API:

void
IP_ADC_init2Unit(AD_InitTypeDef* const _initdata)

Parameters:

ADx: Select an ADC Address

_initdata: ADC Initial Setup Data Structure
See 12.3.2.1 AD_InitTypeDef for details.

Function:

Initial setup of ADC

Supplement:

Call up during ADC suspension with prohibited interruption.

Return Parameter:

No

12.3.2. Data Structure

12.3.2.1. AD_InitTypeDef

Data Fields:

uint8_t **shunt:** Shunt Type
3: 3 shunt
1: 1 shunt

uint8_t **iuch:** Phase-U Current Input AD Channel No. (three-shunts)

uint8_t **ivch:** Phase-V Current Input AD Channel No. (three-shunts)

uint8_t **iwch:** Phase-W Current Input AD Channel No. (three-shunts)

uint8_t **idcch:** DC Current Input AD Channel No. (Single Shunt)

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

uint8_t **pmd_ch:** Select a PMD Channel
cPMD: PMD Channel 0
cPMD: PMD Channel 1

uint8_t **pints:** Select Interruption for PM Trigger
cPINTS_A: ITTADxPDA
cPINTS_B: ITTADxPDB

12.4. Explanation of Constants

12.4.1. Constants for Microcontroller equipped with VE+ (mcuip_drv.h)

12.4.1.1. PMD

Table 12.1 Constants of PMD

For MDCR Register Setting			
Name of Constants	**Description**	**Selected Data**	**Description**
cWPWMES	W-phase edge setting	PWMES_CENTER	Edge unfixed. (center-aligned PWM)
		PWMES_RISING	PWM rising-edge fixed
		PWMES_FALLING	PWM falling-edge fixed
cVPWMES	V-phase edge setting	PWMES_CENTER	Edge unfixed. (center-aligned PWM)
		PWMES_RISING	PWM rising-edge fixed
		PWMES_FALLING	PWM falling-edge fixed
cUPWMES	U-phase edge setting	PWMES_CENTER	Edge unfixed. (center-aligned PWM)
		PWMES_RISING	PWM rising-edge fixed
		PWMES_FALLING	PWM falling-edge fixed
cDSYNCS	Double buffer update timing for the duty compare register and PWM period register.	DSYNCS_INTCYC	Depends on interrupt cycle setting
		DSYNCS_BOTTOM	Updates at PWM carrier bottom
		DSYNCS_TOP	Updates at PWM carrier peak
		DSYNCS_BOTTOM_TOP	Updates at both PWM carrier peak and bottom
cDTCREN	Dead time correction	DTCREN_DISABLE	Correction disable
		DTCREN_ENABLE	Correction enable
cPWMEXCLK	PWM period extension mode	PWMCK_NORMAL	Normal period
		PWMCK_4FOLD	4x period
cPWMSYNT	Port output mode	SYNTMD_0	---
		SYNTMD_1	---
cPWMSPCFY	Duty mode	DTYMD_COMMON	3-phase common mode
		DTYMD_INDEPENDENT	3-phase independent mode
cPWMINT	PWM interrupt timing	PINT_BOTTOM	Interrupt request when PWM counter PMD1MDCNT<MDCNT[15:0]> = 0x0001
		PINT_TOP	Interrupt request when PWM counter PMD1MDCNT<MDCNT[15:0]> = <MDPRD[15:0]>
cPWMINTPRD	PWM interrupt period	INTPRD_HALF	Interrupt request at every 0.5 PWM period
		INTPRD_1	Interrupt request at every PWM period
		INTPRD_2	Interrupt request at every 2 PWM periods
		INTPRD_4	Interrupt request at every 4 PWM periods
cPWMWAVE	PWM carrier waveform	PWMMD_SAWTOOTH	Edge-aligned PWM Sawtooth wave

		PWMMD_TRIANGULAR	Center-aligned PWM Triangular wave

For MDPOT Register Setting			
Name of Constants	**Description**	**Selected Data**	**Description**
cSYNCS	MDOUT transfer timing for trigger synchronous.	SYNCS_ASYNC	Asynchronous
		SYNCS_INTENC	when INTENCx (ENCx interrupt request) occurs
		SYNCS_NTTB	when INTT32Ax0 (TMRBx interrupt request) occurs
cPSYNCS	MDOUT transfer timing for PWM synchronous.	PSYNCS_WRITE	Reflect when write
		PSYNCS_BOTTOM	Reflect when PWM counter MDCNT="1"
		PSYNCS_TOP	Reflect when PWM counter MDCNT=PMDxMDPRD<MDPRD>
		PSYNCS_BOTTOM_TOP	Reflect when PWM counter MDCNT="1" or PMDxMDPRD<MDPRD>

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

For TRGCR Register Setting			
Name of Constants	**Description**	**Selected Data**	**Description**
cTRGMD_3SHT	Trigger timing setting for 3shunt	TRGMD_DIS	Trigger output disabled
		TRGMD_DOWN	Trigger output at down-count match
		TRGMD_UP	Trigger output at up-count match
		TRGMD_UPDOWM	Trigger output at up-/down-count match
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak and bottom
cTRGMD_1SHT	Trigger timing setting for 1shunt	TRGMD_DIS	Trigger output disabled
		TRGMD_DOWN	Trigger output at down-count match
		TRGMD_UP	Trigger output at up-count match
		TRGMD_UPDOWM	Trigger output at up-/down-count match
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak and

			bottom
cBUFSYNC	Triger Buffer update timing	TRGBE_SYNC	Sync
		TRGBE_ASYNC	Async The value written to PMDTRGx is immediately reflected.)

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

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

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

cOVVRS	OVV protection release	OVVRS_NORMAL	Disable automatic release of OVV protection
		OVVRS_AUTO	Enable automatic release of OVV protection

12.4.1.2. VE

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

Table 12.2 Constants of VE

For FMODE Register Setting			
Name of Constants	**Description**	**Selected Data**	**Description**
cSPWMMD	Selects a PWM shift mode	SPWMMD_SPWM1	Shift 1
		SPWMMD_SPWM2U	Shift 2 (U-phase Normal PWM)
		SPWMMD_SPWM2V	Shift 2 (V-phase Normal PWM)
		SPWMMD_SPWM2W	Shift 2 (W-phase Normal PWM)
cPHCVDIS	Phase transformation	PHCVDIS_ENABLE	2-3 phase transformation is enabled
		PHCVDIS_DISABLE	2-3 phase transformation is disabled
cMREGDIS	Keep the previous value of SIN/COS/SECTOR	MREGDIS_EFFECTIVE	Effective
		MREGDIS_NOEFFECTIVE	No effective
cCRCEN	Trigger correction	CRCEN_DISABLE	Disable trigger correction.
		CRCEN_ENABLE	Enable trigger correction.
cIDPLMD	Current polarity detection	IDPLMD_SHUNT	shunt
		IDPLMD_SENSOR	sensor

For MODE Register Setting			
Name of Constants	**Description**	**Selected Data**	**Description**
cT7SQRTEN	Task7 enable.	T7SQRTEN_DISABLE	Disable
		T7SQRTEN_ENABLE	Enable
cT4ATANEN	Task4 enable.	T4ATANEN_DISABLE	Disable
		T4ATANEN_ENABLE	Enable
cVDCSEL	Selects the supply voltage store register	VDCSEL_VDC	Stored in VExVDC.
		VDCSEL_VDCL	Stored in VExVDCL.
cZIEN	Zero current detection	ZIEN_DISABLE	Disable
		ZIEN_ENABLE	Enable
cPVIEN	Phase interpolation	PVIEN_DISABLE	Disable

13. Appendix

13.1. Fixed Decimal Point Processing

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

13.2. Normalization

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

e.g.)

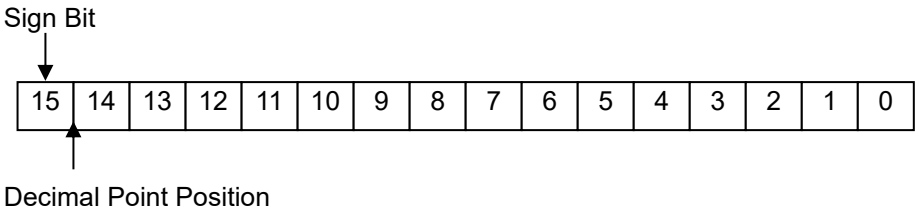


Figure 13.1 Example of 16-Bit Data

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.

13.3. Data Format

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

Table 13.1 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 13.2 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

13.4. Computation with Fixed Decimal Point

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

(1) Multiplication

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

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

(2) Division

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

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

13.5. assert function

This sample software includes an assert function (available only when DEBUG mode is enabled). However, the user must define the function's behavior.

(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
```

14. Revision History

Date	Rev.	Description
2019-08-22	1.0	First release
2026-06-29	1.1	1.1 Changed the Evaluation Board. 1.2 Added Development Environment Information.

RESTRICTIONS ON PRODUCT USE

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

- TOSHIBA reserves the right to make changes to the information in this document and related Product without notice.
- This document and any information herein may not be reproduced without prior written permission from TOSHIBA. Even with TOSHIBA's written permission, reproduction is permissible only if reproduction is without alteration/omission.
- Though TOSHIBA works continually to improve Product's quality and reliability, Product can malfunction or fail. Customers are responsible for complying with safety standards and for providing adequate designs and safeguards for their hardware, software and systems which minimize risk and avoid situations in which a malfunction or failure of Product could cause loss of human life, bodily injury or damage to property, including data loss or corruption. Before customers use the Product, create designs including the Product, or incorporate the Product into their own applications, customers must also refer to and comply with (a) the latest versions of all relevant TOSHIBA information, including without limitation, this document, the specifications, the data sheets and application notes for Product and the precautions and conditions set forth in the "TOSHIBA Semiconductor Reliability Handbook" and (b) the instructions for the application with which the Product will be used with or for. Customers are solely responsible for all aspects of their own product design or applications, including but not limited to (a) determining the appropriateness of the use of this Product in such design or applications; (b) evaluating and determining the applicability of any information contained in this document, or in charts, diagrams, programs, algorithms, sample application circuits, or any other referenced documents; and (c) validating all operating parameters for such designs and applications. **TOSHIBA ASSUMES NO LIABILITY FOR CUSTOMERS' PRODUCT DESIGN OR APPLICATIONS.**
- **PRODUCT IS NEITHER INTENDED NOR WARRANTED FOR USE IN EQUIPMENTS OR SYSTEMS THAT REQUIRE EXTRAORDINARILY HIGH LEVELS OF QUALITY AND/OR RELIABILITY, AND/OR A MALFUNCTION OR FAILURE OF WHICH MAY CAUSE LOSS OF HUMAN LIFE, BODILY INJURY, SERIOUS PROPERTY DAMAGE AND/OR SERIOUS PUBLIC IMPACT ("UNINTENDED USE").** Except for specific applications as expressly stated in this document, Unintended Use includes, without limitation, equipment used in nuclear facilities, equipment used in the aerospace industry, Class 3 medical devices, equipment used for automobiles, and military vehicles and munitions. **IF YOU USE PRODUCT FOR UNINTENDED USE, TOSHIBA ASSUMES NO LIABILITY FOR PRODUCT.** For details, please contact your TOSHIBA sales representative or contact us via our website.
- 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.**

Toshiba Electronic Devices & Storage Corporation
https://toshiba.semicon-storage.com/