DC 300 V Input BLDC Motor Sensorless Control Circuit Using TPD4165K

Software Guide

RD179d-SWGUIDE-01

Toshiba Electronic Devices & Storage Corporation

Table of Contents

1. Overview

This document describes the software specifications of DC 300 V Input BLDC Motor Sensorless Control Circuit reference design.

1.1 Outline of Specifications

- Microcontroller: TMPM374FWUG (Operation Clock 80 MHz)
- Intelligent Power Device: TPD4165K
- Motor: Brushless motors for indoor unit fan of air conditioner (operation voltage: DC280 V)
- Development environment: EWARM Ver8.50.1.24811 or KEIL Ver5.29.0.0

- Control Overview

  ・Brushless DC Motor Sensorless Vector Control

  ・Support for external speed control

  ・Support for LED output (LED for monitor) and DAC output (for variable value analog output) for evaluation

- Motor Control Details

Item	Control Details
Current Detection Type	3-shunt or 1-shunt
Position Detection Type	Sensorless
PWM Modulation Type	3-phase modulation or 2-phase modulation

1.2 Processing Overview

The vector control software consists of three layers: an application layer that performs user interface processing, a motor control layer that controls the motor operation status via state transitions, and a motor drive layer that directly accesses the motor drive circuit to drive the motor.

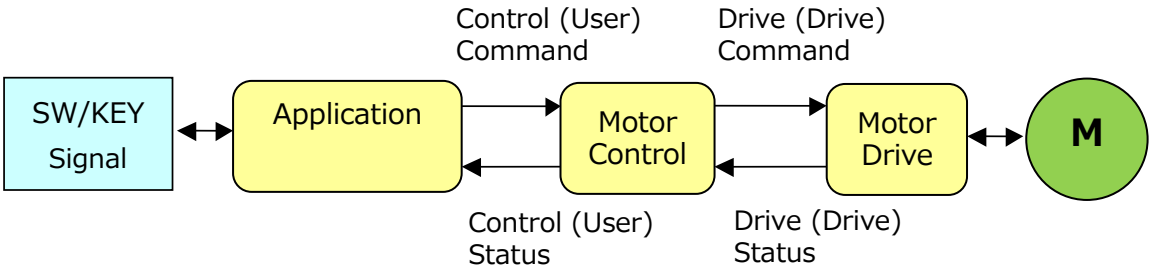


Fig. 1 Structure of Vector Control Software

1

i. The application layer takes input control commands set by switches, keys, communication, etc. and gives them to the motor control layer together with other control commands. In addition, the control status is obtained from the motor control layer, necessary processing is performed, and it is output to a port, etc.

ii. Motor control layer reads the control commands provided by the application layer and converts them into more specific drive commands according to the motor operating status, giving them to the motor drive layer. It also acquires the drive status from the motor drive layer, performs the necessary processing, and transfers it to the application layer.

The motor drive layer reads the drive command given from the motor control layer and drives the motor. It also monitors the operation of the motor, performs the necessary processing according to the condition, and transfers the drive status to the motor control layer.

For example, when a new control target speed received from the application layer while the motor is rotating, the motor drive layer cannot respond to a sudden change in the target speed, so it is converted to a gradually changing drive target speed within the motor control layer, and then it is given to the motor drive layer.

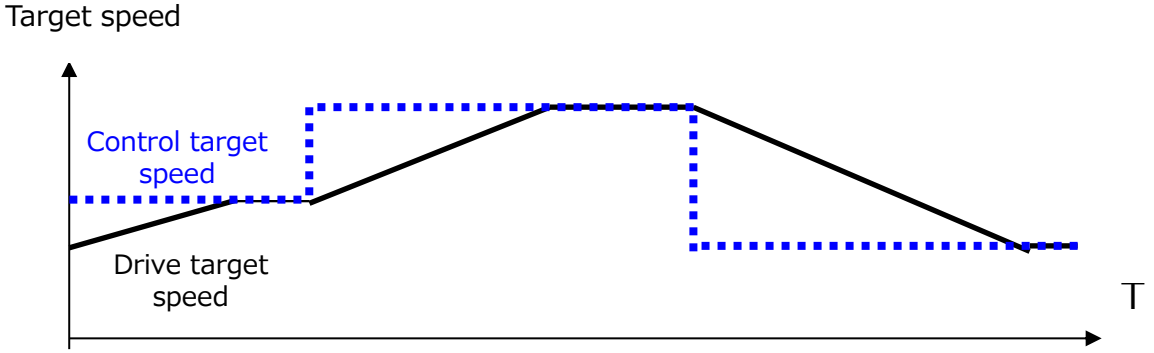


Fig. 2 Control Target Speed and Drive Target Speed2

System Block Diagram

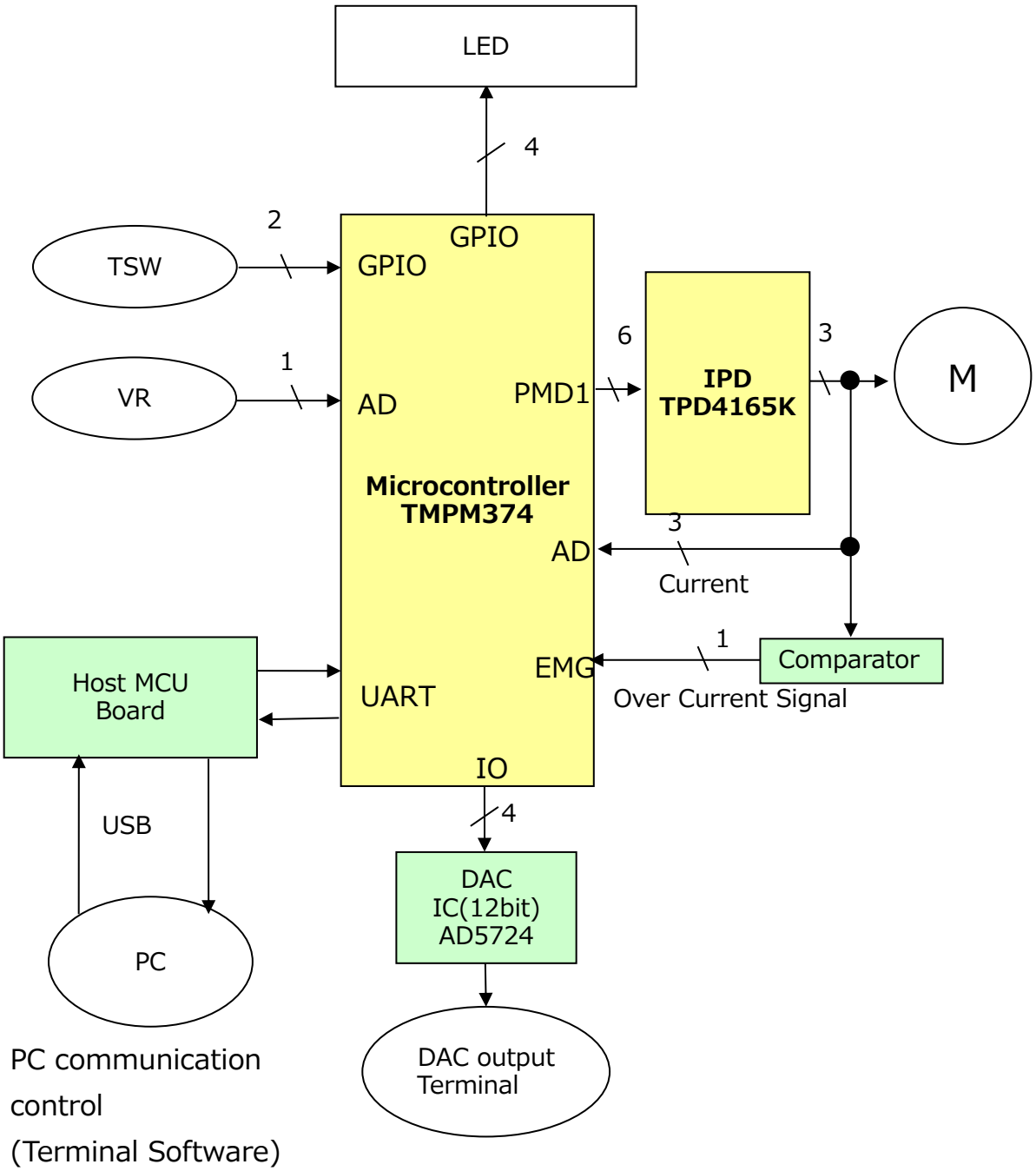


Fig. 3 System Block Diagram3

2. Source File Configuration

```
metis_motor_sample_r01
 |
 └──common                  Contains files related to the vector control driver
 |  B_User.c                Vector control user configuration related source files
 |  B_User.h                Vector control user configuration related header files
 |  C_Control.c             Vector control related source files
 |  C_Control.h             Vector control related header files
 |  D_Driver.c              Vector control driver source file
 |  D_Driver.h              Vector control driver header file
 |  E_Sub.c                 Operation function source file
 |  E_Sub.h                 Operation function header file
 |  ipdefine.h              MCU setting related header file
 |  ipdrv.c                 MCU hardware reference source file
 |  ipdrv.h                 MCU hardware reference header file
 |  sys_macro.h             Macro definition header file
 |
 └──M374
    |
    └──source
       |  initial.c          MCU initialization related source file
       |  initial.h          MCU initialization related header file
       |  main.c             Main routine
       |  system_int.c       Interrupt function source file
       |  system_int.h       Interrupt function header file
       |  usercon.c          Source file for user control
       |  usercon.h          Header file for user control
       |
       └──driver            Contains file related to device driver
       |  D_Para.h           Vector control arguments: (common for channels) header file
       |  D_Para_ch1.h       Vector control arguments: (for channel 1) header file
       |  dac_drv.c          DAC IC driver source file
       |  dac_drv.h          DAC IC driver header file
       |  interrupt.c        Interrupt control related source files
       |  interrupt.h        Interrupt control related header file
       |  mcuip_drv.c        MCU hardware setting driver source file
       |  mcuip_drv.h        MCU hardware setting driver header file
       |  CheckVdq.h         Vdq operation function source file
       |  CheckVdqq.h        Vdq operation function header file
       |
       └──Libraries                 Contains CMSIS cores and libraries for IP
          └──CMSIS                        Contains CMSIS cores
          |  |  system_TMPM374.c         Source file for initializing the MCU
          |  |  system_TMPM374.h         Header file for initializing the MCU
          |  |  TMPM374.h                MCU register definition header file
          |  └──startup
          |     ├──arm
          |     |        startup_TMPM374.s      Startup assembler source (for arm)
          |     └──iar
          |              startup_TMPM374.s      Startup assembler source (for IAR)
          |
          └──IP_Driver          Peripheral driver is stored
               tmpm374_adc.c
               tmpm374_adc.h
               tmpm374_gpio.c
               tmpm374_gpio.h
               tmpm374_tmrb.c
               tmpm374_tmrb.h
               tmpm374_uart.c
               tmpm374_uart.h
               tmpm374_wdt.c
               tmpm374_wdt.h
               tx03_common.h
```

3. Evaluation Environment

3.1 Development tools

This software was developed using the following development tools.

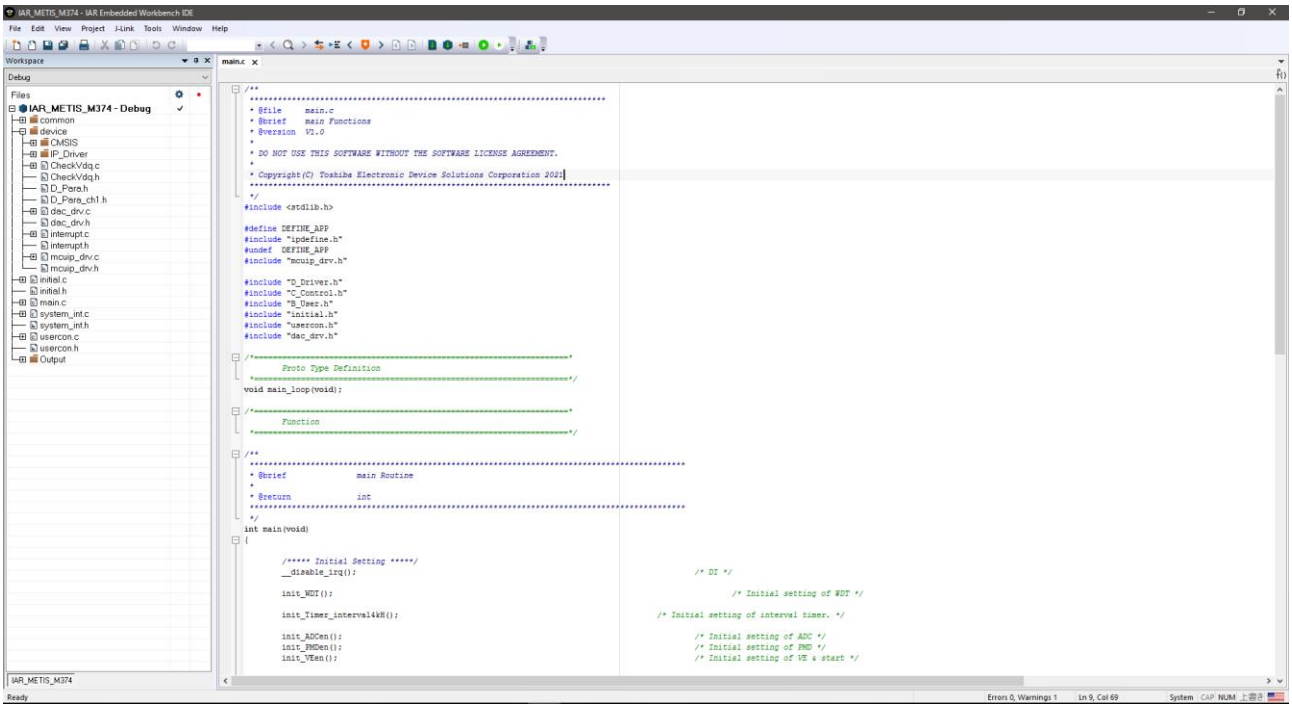
IAR Embeded Workbench for ARM 8.50.1.24811
KEIL µVision MDK-Lite 5.29.0.0

3.2 How to start up the project

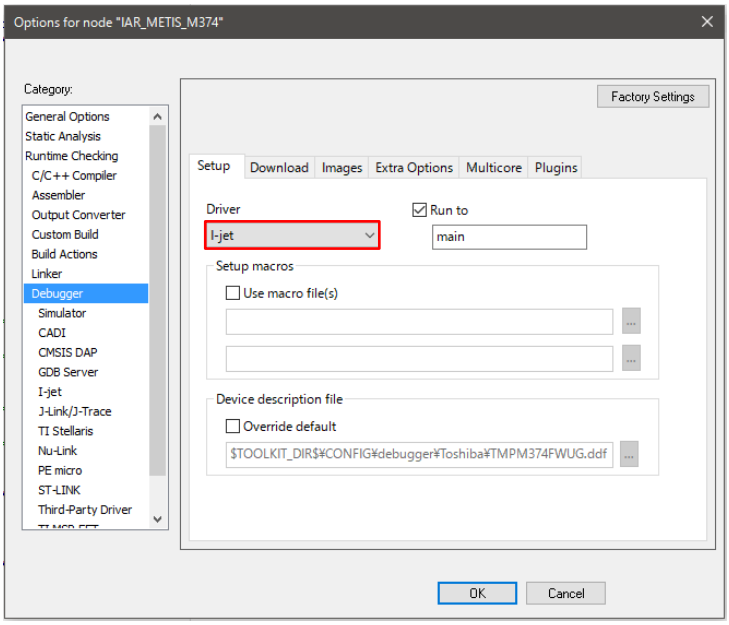
Illustrated below using IAR Embeded Workbench.

1. Double-click M374¥iar_V8501¥ IAR_METIS_M 374.eww or
   Open IAR_METIS_M374.eww from [File] > [Open] > [Workspace].

2. The following screen will appear.



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



Select the tool to be used.

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

3.3 DAC Output

The DAC output function is implemented for viewing changes in variables on an oscilloscope, etc.
To enable DAC output, enable the following definition of D_Para.h.
#define __USE_DAC

The variables to be DAC-printed are described in the function UiOutDataStartByVE() of the file usercon.c.
If no variables are to be checked, add them as necessary.

« DAC-Output Setting Variable »
dac.select　　DAC output selection
dac.motch　　Set the motor CH to be output using DAC output
dac.datsft0 - 3　Set the amount of data shift

The default is as follows:
Modes can be switched by file usercon.c.
dac_t dac = {0,1,0,0,0,0};

« DAC Output Default Variable »

1. MODE 0
   VoA: U-phase current
   VoB: V-phase current
   VoC: W-phase current
   VoD: Electric angle

2. MODE 1
   VoA: Id reference (target value)
   VoB: Id (current value)
   VoC: Iq reference (target value)
   VoD: Iq (current value)

3. MODE 2
   VoA: Angular velocity (target value)
   VoB: Angular velocity (current value)
   VoC: Angular velocity (difference)
   VoD: Iq (current value)

4. MODE 3
   VoA: U-phase current
   VoB: Iq reference (target value)
   VoC: Id reference (target value)
   VoD: Angular velocity (current value)

5. MODE 4
   VoA: Vdc (current value)
   VoB: Vdq (current value)
   VoC: Vd (current value)
   VoD: Vq (current value)

3.4 User Interface

The following functions are available as user interfaces.

« User Interface Description »

1. Motor speed adjustment
   The motor speed can be specified by operating the volume switch (VR1).
   Speed range: cHZ_MIN to 60rps (electric angle)

2. Motor rotation direction switching
   Rotation direction of the motor can be changed by the toggle switch (TSW1).

Hi:CW Low:CCW

3. LED display

   Communication status and motor status can be checked by LED1～LED3.

   1. Communication indicator (LED1)

      No error: Off

      Hardware EMG: On

      Soft EMG: 1 s blinking

      Current detection error at startup: 0.5 s blinking

      Vdc error: 0.25 s blinking

   2. VE interrupt processing time indicator (LED2)

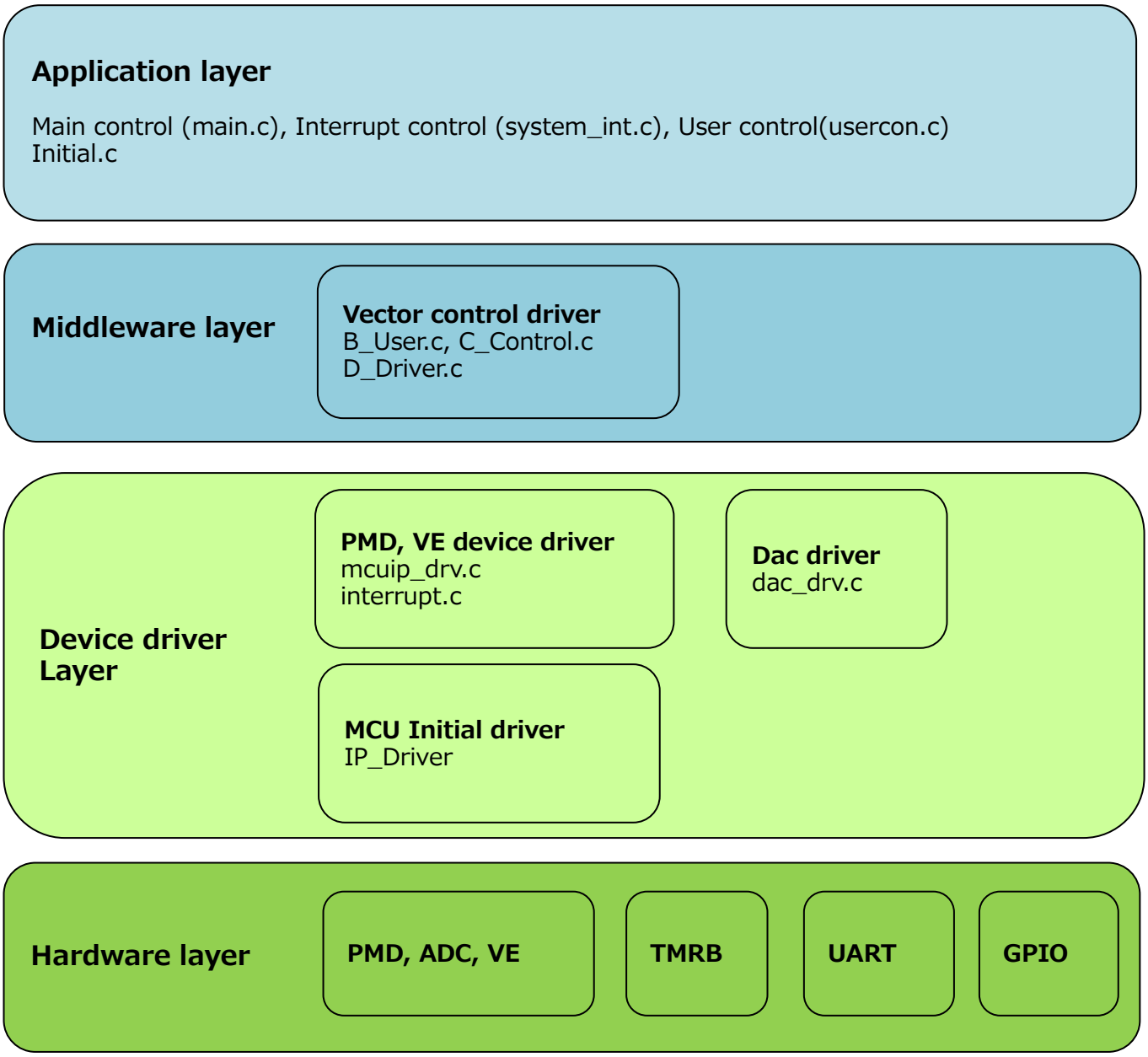
      On during VE interrupt processing

   3. Communication indicator (LED3)

      Communication established: On

      Communication error: Blinking (250 ms ON/250 ms OFF)

4. Module Configuration



5. Assigning MCU Hardware Resources

5.1 Peripheral Interface

IP		Details	Remarks
TMRB	CH0	4 kHz periodic interrupt	
	CH1	Not used	
	CH2	Not used	
	CH3	Not used	
	CH4	Not used	
	CH5	Not used	
	CH6	Not used	
	CH7	Not used	
SIO/UART	CH0	Main board communication control	UART
	CH1	DAC IC control	SIO
	CH2	Not used	
ADC	UNITB	Slide volume	Control mode: Single unit only
		Obtains motor CH1 coiled current and power supply voltage	
PMD	CH1	Motor CH1 control	
VE	CH1	Motor CH1 control	

5.2 Interrupt

Factor Name	Details of Processing	Function Name
INTTB00IRQn	4 kHz cycle timing creation	INTTB00_IRQHandler
INTVCN1_IRQn	Vector control software processing for CH1	INTVCN1_IRQHandler
INTTX0_IRQn	Main PCB communication control (transmission)	INTTX0_IRQHandler
INTRX0_IRQn	Main PCB communication control (reception)	INTRX0_IRQHandler
INTTX1_IRQn	DAC IC control	INTTX1_IRQHandler
INTPMD1_IRQn	VE start (1-shunt only)	INTPMD1_IRQHandler

The interrupt priority can be changed using the constants shown below in ipdefine.h.

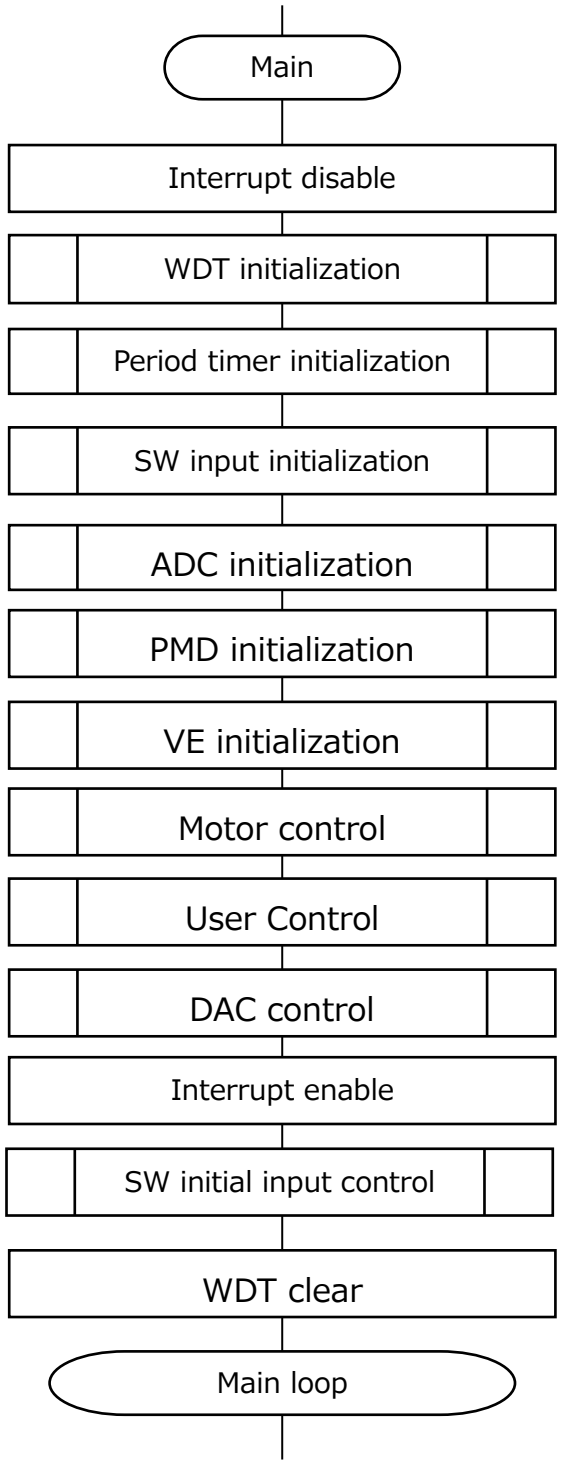
```
/* High Low */
/*   0 ---- 7   */

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

5.3 GPIO

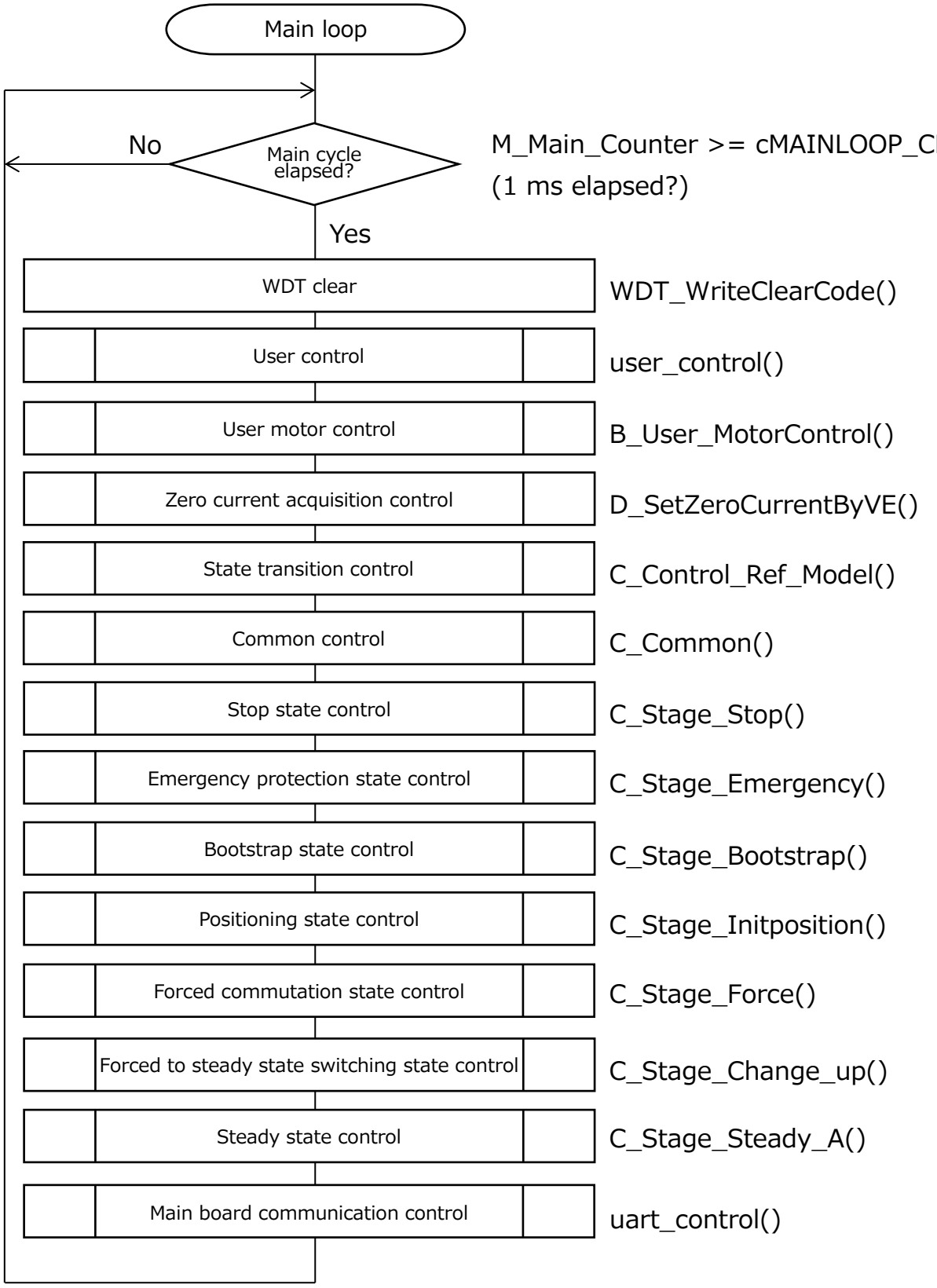
PORT	METIS Pin Name	Function
PG0	UH_1	Motor U
PG1	UL_1	Motor V
PG2	VH_1	Motor W
PG3	VL_1	Motor X
PG4	WH_1	Motor Y
PG5	WL_1	Motor Z
PG6(EMG1)	XEMG_1	Motor EMG
PG7	RELAY	Inrush current prevention
PE0	TXD_UA_MCU	UART_TX
PE1	RXD_UA_MCU	UART_RX
PF2	LED1	Turn On during EMG
PF3	LED2	Turn On during VE interrupt processing
PF4	LED3	Control mode = Turn On by communication
PI3(AINB2)	AD_UP_1	U-phase current
PJ0(AINB3)	AD_VP_1	V-phase current
PJ6(AINB9)	AD_WP_1	W-phase current
PJ7(AINB10)	AD_VDC_1	VDC voltage
PK0	KEY_A	Volume resistance
PK1	KEY_3	Slide switch 3 (rotation direction)
PE7	KEY_2	Slide switch 2 (control mode)
PA6	XSYNC_SPI	Serial for DAC
PA5	SDO_SPI	Serial for DAC
PA4	SCLK_SPI	Serial for DAC

6. General Flow

6.1 Main Routine (main)

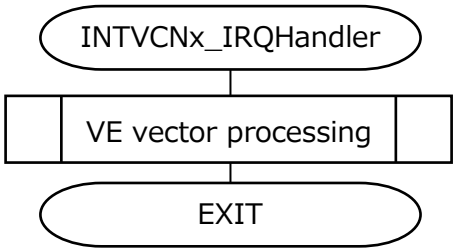


6.2 Main Loop (main_loop)



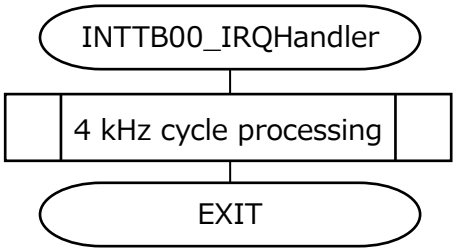
6.3 Interrupt

When the VE schedule is completed



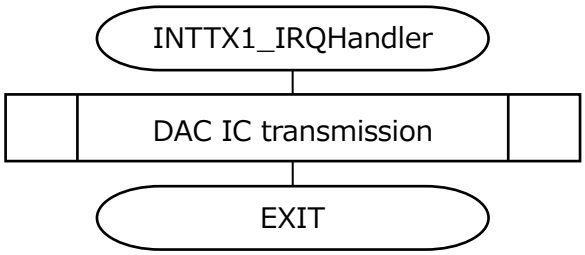
INT_VectorContorlByVE_CH1 ()

Every 4 kHz cycle



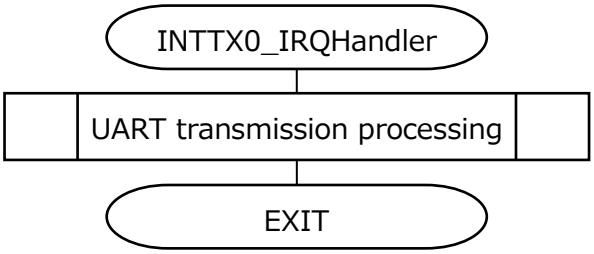
INT_interval_4kHz ()

Upon completion of 8-bit transmission of DAC communication



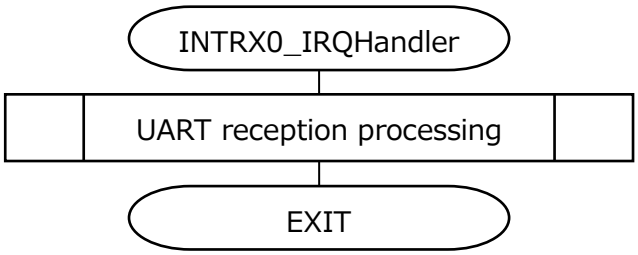
DacWriteContinue ()

When main PCB communication 8-bit transmission is completed



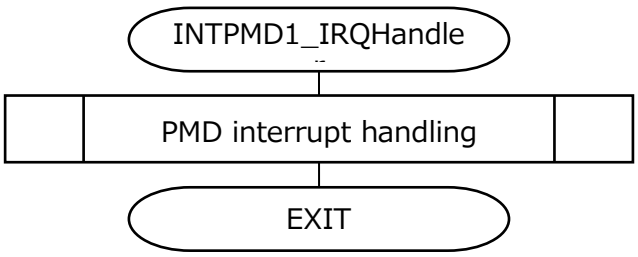
INT_uart0tx ()

When main PCB communication 8-bit reception is completed



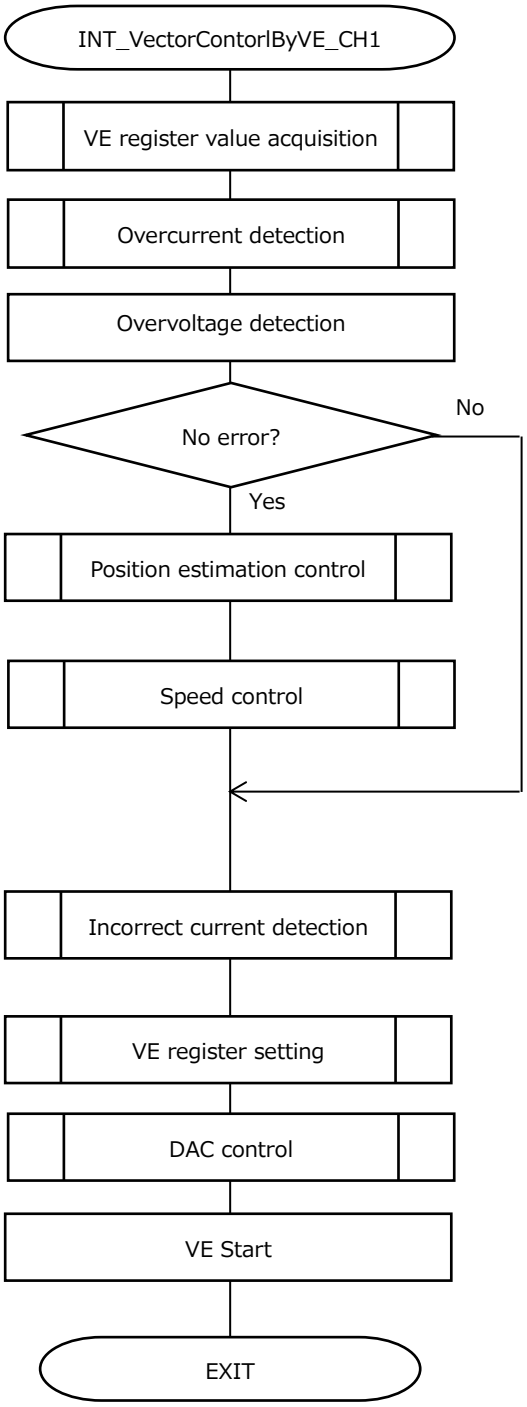
INT_uart0rx ()

At PWM interrupt (1-shunt only)



INT_PWM1()

6.3.1 VE Vector Processing (INT_VectorControl_byVE)



VE_GetdataFromVEreg()

D_Check_OverCurrent ()

drv.state.all == 0?

D_Detect_Rotor_Position ()

D_Control_Speed ()

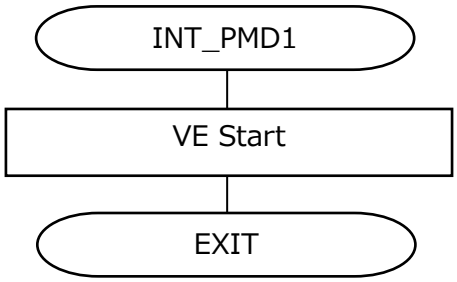
D_Check_DetectCurrentError_3shunt()/
D_Check_DetectCurrentError_1shunt()

VE_SetdataToVEreg_xxx()

UiOutDataStartByVE(ch)

VE_Start ()
(Note) 3-shunt only

6.3.2 PMD Interrupt Handling (INT_PMD1)

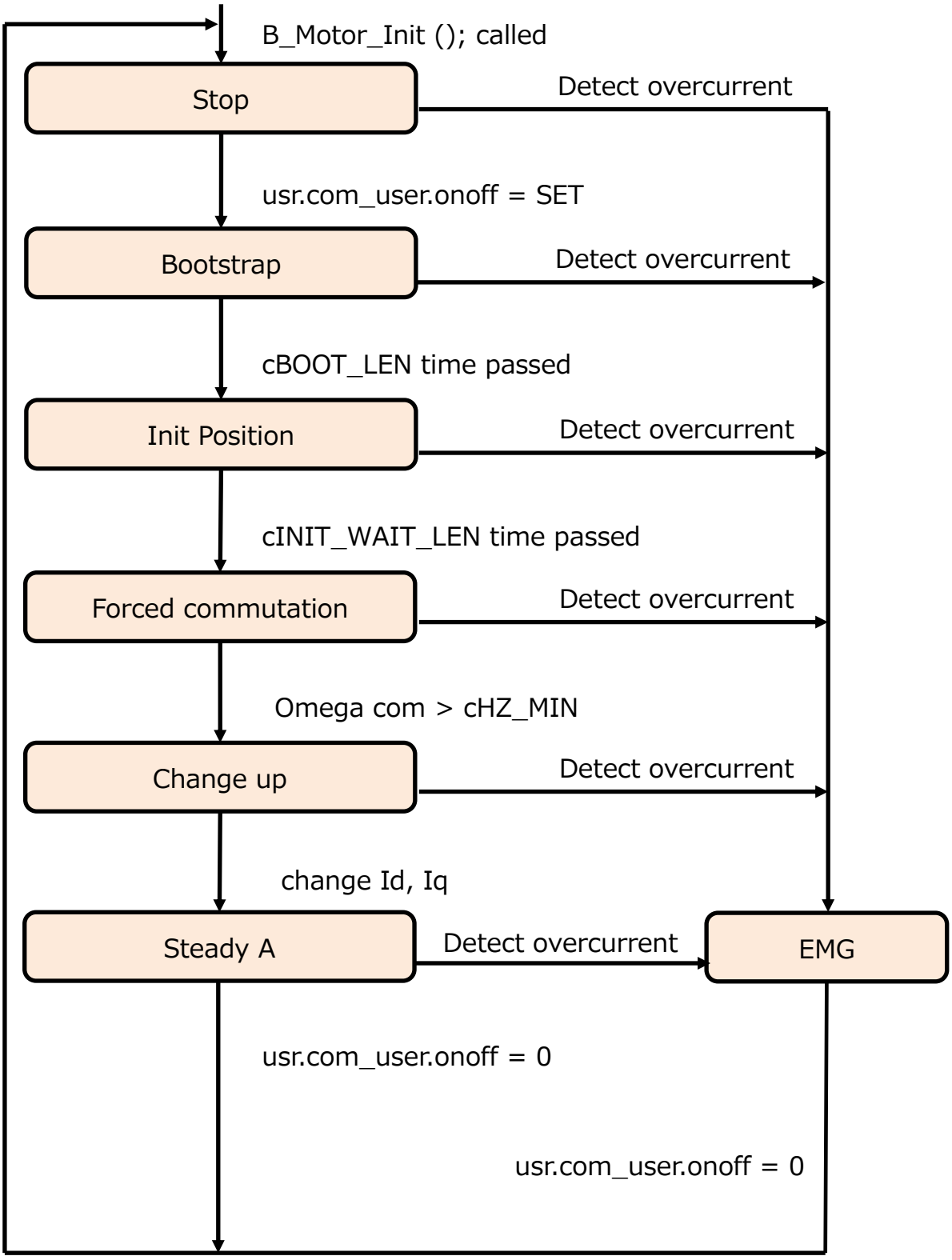


VE_Start ()

(Note) 1-shunt only

7. State Transition

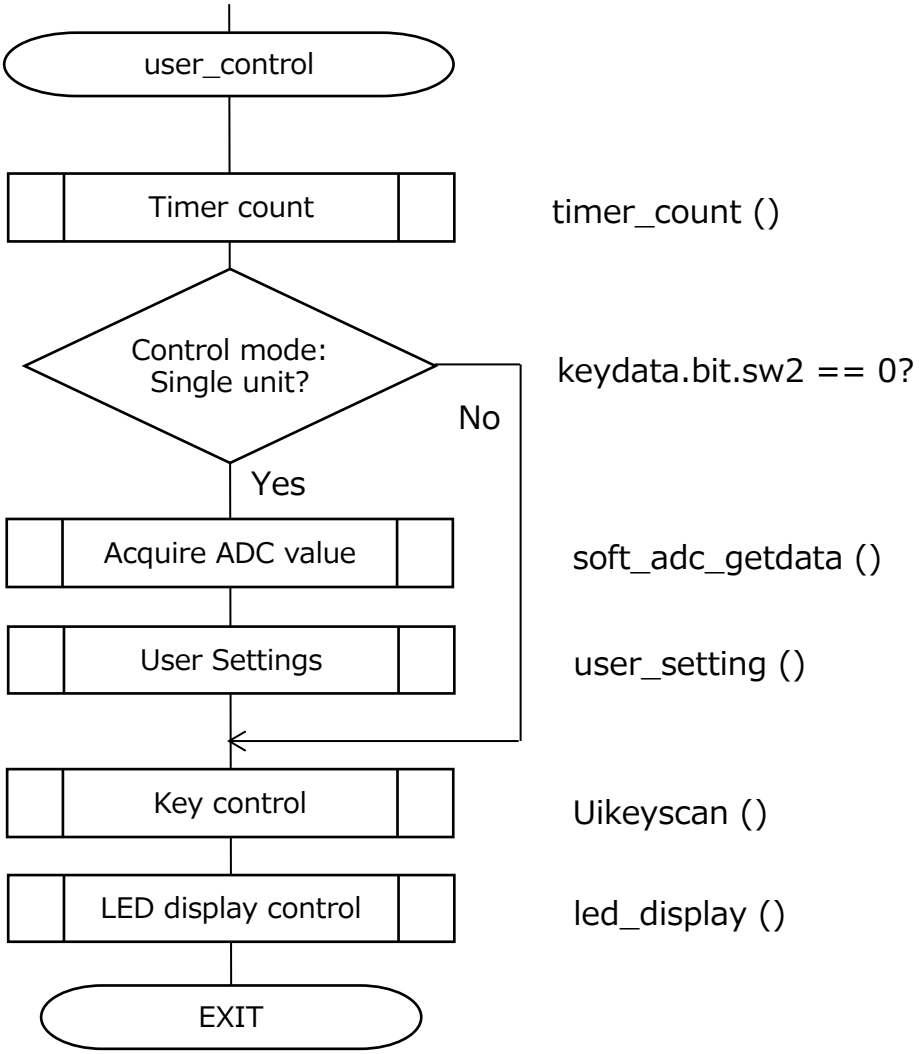
7.1 Sensorless



8. User Application

8.1 User Control

Each processing of the user application part is performed in each main period (1 ms).



8.1.1 Timer Count Control (timer_conut)

The following timer counts are performed for each main cycle.

1. Communication error LED blinking timer (1 ms × 250 counts)
   Counting is performed to switch LED3 status every 250 ms when a communication error occurs.
2. EMG status blinking timer (1 ms × 250 counts/500 counts/1000 counts)
   It is counted to switch LED1 status according to the EMG status.
3. Communication TimeOut timer (1 ms × 2000 counts)
   The count is used to check the communication status with the main board during communication.

8.1.2 Acquire AD-Value (soft_adc_getdata)

Used to acquire the AD-value (12-bit) of the volume switch (VR1) using the independent conversion setting.

After 10 times of acquisition, 8 averages excluding the max. and min. values are stored as valid values of VR1 in ad_vr. avedat.

8.1.3 Key Control (Uikeyscan)

Performs a key scan of TSW2,3.

Each key data is confirmed by 20 consecutive matches and stored in keydata.

8.1.4 Custom (user_setting)

Follows VR1 entered by the user and performs following settings.

1. Motor speed control
   At 8-bit resolution of ad_vr. avedat
   Less than 0x10: 0 (0 Hz)
   0xF0 or more: 60 (maximum speed)
   0x10 to 0xEF: Speed calculation as effective value

8.1.5 LED-Display Control (led_display)

Performs output control of LED1 to 3 (always ON, always OFF, blinking) according to EMG type and communication status.

It also performs time management (0.25 s, 0.5 s, 1 s) for blink display timing.

For details on the indications, refer to section 3.4 About the User Interface.

8.1.6 Communication Control (uart_control)

Receives the rotation speed and information status command sent from the main board, and transmits the corresponding information to the main board.

The communication settings are:

115200bps, 8-bit data, 1-bit stop bit, no parity, no flow control.

9. Function Description

Functions are the interface between the application layer and motor control layer and between motor control layer and motor drive layer. These are described below.

9.1 Control Commands

Control commands are described below.

9.1.1 Control Methods (usr.com_user)

・Starts and stops the motor
・Modulation type (2-phase modulation, 3-phase modulation)
・Shift PWM on, off (enabled only for 1-shunt, 2-phase modulation)

```
typedef struct {
  uint16_t spwm:1;   /* Shift PWM            0=off, 1=on */
  uint16_t modul:1;  /* PWM Moduration       0=3-phase modulation, 1=2-phase modulation */
  uint16_t onoff:1;  /* Motor start command  0=off, 1=on*/
} command_t;
```

Command_t Com_user;
The application sets the usr.com_user as a control command.

9.1.2 Control Target Speed

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

The application sets the usr.omega_user as the control target speed.

9.1.3 Startup Current

```
q15_t    Id_st_user;      /* [A/maxA] d-axis starting current command, Q15 */
q15_t    Iq_st_user;      /* [A/maxA] q-axis starting current command, Q15 */
```

In the application, the usr.Id_st_user and usr.Iq_st_user are set as startup current commands.

9.2 Drive Command

The driving commands are described below.

9.2.1 Drive Method (drv. command)

```
Command_t  Command;
```

The application sends the drive method to the motor control driver via the drv. command.

9.2.2 Vector Control Command (drv.vector_cmd)

This command is a vector control operation command that manages processing for each 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 (unuse)*/
  uint16_t F_vcomm_Edetect:1;      /* Position detect    0=off, 1=on */
  uint16_t F_vcomm_Idetect:1;      /* Current detect    0=off, 1=on (unuse)*/
  uint16_t F_vcomm_onoff: 1;       /* Motor output 0=off 1=on */
} vectorcmd_t;
```

In each stage, the vector control drive command (drv.vector_cmd) is set as follows and the command is sent to motor drive.
Note that F_vcomm_volt and F_vcomm_Idetect are not used.

cmd / Stage	theta	omega	current	Edetect
Stop	CLEAR	CLEAR	CLEAR	CLEAR
Bootstrap	CLEAR	CLEAR	CLEAR	CLEAR
InitPosition	CLEAR	CLEAR	CLEAR	CLEAR
Forced	SET	CLEAR	CLEAR	SET
Change_up	SET	SET	CLEAR	SET
Steady	SET	SET	SET	SET
Emergency	CLEAR	CLEAR	CLEAR	CLEAR

1) F_vcomm_theta
   In the rotor position estimation operation, when it is SET, the estimated value is taken as the rotor position. And when it is CLEAR, the command is taken as the rotor position.

2) F_vcomm_omega
   In the rotor position estimation operation, when it is SET, the estimated value is the speed ω. And when it is CLEAR, the command is the speed ω.

3) F_vcomm_current
   Specifies the calculation method of the reference value of d and q-axis currents in speed control.
   When it is SET, the reference value is the value obtained by PI control from the speed deviation. And when it is CLEAR, PI control is not executed, and the reference value is set as it is.

4) F_vcomm_Edetect
   When it is SET, the induced voltage is calculated and the rotor position estimation calculation is performed. And when it is CLEAR, the induced voltage is not calculated and the induced voltage is set to 0, and the rotor position is set to the command value.

9.3 Driving State

9.3.1 Error Status (drv. state)

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

Loss_sync Step-out detection It is set when step-out is detected (not implemented)

Emg_DC Abnormal voltage detection It is set when abnormal voltage is detected.

Emg_I Current detection error It is set when abnormal current is detected. (not used)

Emg_S Soft overcurrent detection It is set when overcurrent is detected by software processing.

Emg_H Hardware overcurrent detection It is set when an overcurrent is detected by the MCU hardware function.

9.4 Motor Control Structure

The motor control struct (vector_t) is defined in ipdefine.h. Variables are declared per motor channel as follows:

Example)

Vector_tMotor_ch1; /* Motor data for ch1 */

9.4.1 List of Variables

Type	Name	Meaning	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	Interrupt stage	---	CiStop CiBootstrap CiInitposition_i CiInitposition_v

				CiForce_i CiForce_v CiChange_up CiSteady_A CiEmergency
q31_u	Drv.omega_com	Drive speed command 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 command value	Q31	
q31_u	Drv.Iq_com	q-axis current command 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	Electric angle command value	Q0	
Uint32_u	Drv.theta	Rotor position	Q0	
q15_t	Drv.Vdc	Power supply voltage	Q15	
q31_u	Drv.Vdc_ave	(not used)	---	
q31_u	Drv.Vd	d-axis voltage	Q31	
q31_u	Drv.Vq	q-axis voltage	Q31	
q15_t	Drv.Vdq	(not used)	---	
q31_u	Drv.Vdq_ave	(not used)	---	
q31_t	Drv.Vd_out	Output voltages	Q31	
q15_t	Drv.Ed	d-axis induced voltage	Q15	
q15_t	Drv.Eq	(not used)	---	
q31_t	Drv.Ed_I	d-axis induced voltage integral value	Q31	
q31_t	Drv.Ed_PI	d-axis induced voltage PI value	Q31	
State_t	Drv.state	Motor error status	---	
Command_t	Drv.command	Driving method	---	(Refer to 9.2.1.)
Vectorcmd_t	Drv.vector_cmd	Vector control command	---	(Refer to 9.2.2.)
q15_t	Drv.spwm_threshold	Shift switching speed	Q15	1-Shunt only
Uint16_t	Drv.chkpls	Current detection protection Duty range	Q0	
Uint8_t	Drv.idetect_error	Current detection status	---	0:Detectable 1:Undetectable
q15_t	Drv.Ia_raw	a-phase current (raw data)	Q15	
q15_t	Drv.Ib_raw	b-phase current (raw data)	Q15	
q15_t	Drv.Ic_raw	c-phase current (raw data)	Q15	

q15_t	Drv.Ia	a-phase current (false detection protection)	Q15	
q15_t	Drv.Ib	b-hase current (false detection protection)	Q15	
q15_t	Drv.Ic	c-phase current (false detection protection)	Q15	
q31_u	Drv.Iao_ave	a-phase zero current average ADC value	Q0	
q31_u	Drv.Ibo_ave	b-phase zero current average ADC value	Q0	
q31_u	Drv.Ico_ave	c-phase zero current average ADC value	Q0	
Uint8_t	Drv.spdprd	Speed control period	Q0	
q31_u	Usr.omega_user	Control target speed	Q31	
q15_t	Usr.Id_st_user	Starting Id current value	Q15	
q15_t	Usr.Iq_st_user	Starting Iq current value	Q15	
Uint16_t	Usr.lambda_user	Initial rotor position	Q0	
Command_t	Usr.com_user	Control method	---	(Refer to 9.1.1.)
Command_t	Usr.com_user_1	Previous control method	---	
q15_t	Para.omega_min	Forced commutation termination speed	Q15	
q15_t	Para.omega_v2i	Current control switching speed	Q15	
q15_t	Para.spwm_threshold	Shift PWM switching speed	Q15	
q31_t	Para.vd_pos	Positioning command voltage	Q31	
q31_t	Para.spd_coef	Output voltage coefficient	Q15	
q31_u	Para.sp_ud_lim_f	Drive speed increase/decrease limit value	Q31	
q31_u	Para.sp_up_lim_s	Drive speed increase limit value	Q31	
q31_u	Para.sp_dn_lim_s	Drive speed decrease limit value	Q31	
Uint16_t	Para.time.initpos	Positioning time	Q0	
Uint16_t	Para.time.initpos2	Positioning status wait time	Q0	
Uint16_t	Para.time.bootstp	Bootstrap time	Q0	
Uint16_t	Para.time.go_up	Wait time after change-up	Q0	
q31_t	Para.iq_lim	q-axis current limit value	Q31	

q31_t	Para.id_lim	d-axis current limit value	Q31	
q15_t	Para.err_ovc	Overcurrent setting value	Q15	
q31_t	Para.pos.kp	Position estimation proportional gain	Q15	
q31_t	Para.pos.ki	Position estimation integral gain	Q15	
Int32_t	Para.pos.ctrlprd	Position estimation control period	Q16	
q31_t	Para.spd.kp	Speed control proportional gain	Q15	
q31_t	Para.spd.ki	Speed control integral gain	Q15	
Uint8_t	Para.spd.pi_prd	(not used)	Q0	
q31_t	Para.crt.dkp	d-axis current control proportional gain	Q15	
q31_t	Para.crt.dki	d-axis current control integral gain	Q15	
q31_t	Para.crt.qkp	q-axis current control proportional gain	Q15	
q31_t	Para.crt.qki	q-axis current control integral gain	Q15	
q31_t	Para.motor.r	Motor winding resistance	Q15	
q31_t	Para.motor.Lq	Motor q-axis inductance	Q15	
q31_t	Para.motor.Ld	Motor d-axis inductance	Q15	
Int32_t	Para.delta_lambda	Current switching phase at the time of change-up	Q0	
Uint16_t	Para.chkpls	Current detection protection Duty range	Q0	
Uint32_t	Stage_counter	Stage counter	Q0	
Shunt_type_e	Shunt_type	Shunt type	---	c3shunt
Boot_type_e	Boot_type	Activation type	---	cBoot_i cBoot_v

9.5 Function Details

9.5.1 ADC Initialization (init_ADCen)

9.5.1.1 Syntax

Void init_ADCen(void)

Argument :

None

Return value :

None

9.5.1.2 **Details of Processing**

Perform the initialization of the ADC.

・ADC setting for motor

・ADC enable

9.5.2 PMD Initialization (init_PMDen)

9.5.2.1 **Syntax**

Void init_PMDen(void)

Argument :

None

Return value :

None

9.5.2.2 **Details of Processing**

Performs initialization of PMD (Programmable Motor Driver).

9.5.3 VE Initialization (init_VEen)

9.5.3.1 **Syntax**

Void init_VEen(void)

Argument :

None

Return value :

None

9.5.3.2 **Details of Processing**

Default setting of VE (Vector Engine).

・VE interrupt setting

・VE setting

9.5.4 Motor Control Initialization (B_Motor_Init)

9.5.4.1 **Syntax**

Void B_Motor_Init(void)

Argument :

None

Return value :

None

9.5.4.2 **Variable**

Direction	Name	Meaning	Q Format	Remarks
Output	shunt_type	Shunt type	---	
	boot_type	Drive type	---	
	stage.main	Main stage	---	

	stage.sub	Sub-stage	---	
	drv.Iao_ave	U-phase zero current average ADC value	Q0	
	drv.Ibo_ave	V-phase zero current average ADC value	Q0	
	drv.Ico_ave	W-phase zero current average ADC value	Q0	
	usr.Iq_st_user	Starting Iq current value	Q15	
	usr.Id_st_user	Starting Id current value	Q15	
	usr.lambda_user	Initial rotor position	Q0	
	para.motor.r	Motor winding resistance	Q15	
	para.motor.Lq	Motor q-axis inductance	Q15	
	para.motor.Ld	Motor d-axis inductance	Q15	
	para.spwm_threshold	Shift PWM switching speed	Q15	
	para.chkpls	Current detection protection duty range	Q0	
	para.vd_pos	Positioning command voltage	Q31	Only at voltage startup
	para.spd_coef	Output voltage coefficient	Q15	Only at voltage startup
	para.sp_ud_lim_f	Drive speed increase/decrease limit value	Q31	
	para.sp_up_lim_s	Drive speed increase limit value	Q31	
	para.sp_dn_lim_s	Drive speed decrease limit value	Q31	
	para.time.bootstp	Bootstrap time	Q0	
	para.time.initpos	Positioning time	Q0	
	para.time.initpos2	Positioning status wait time	Q0	
	para.time.go_up	Waiting time after change-up	Q0	
	para.omega_min	Forced commutation termination speed	Q15	
	para.omega_v2i	Current control switching speed	Q15	
	para.delta_lambda	Current switching phase at the time of change-up	Q0	
	para.pos.ki	Position estimation integral gain	Q15	
	para.pos.kp	Position estimation proportional gain	Q15	
	para.pos.ctrlprd	Position estimation control period	Q16	
	para.spd.ki	Speed control integral gain	Q15	
	para.spd.kp	Speed control proportional gain	Q15	
	para.crt.dki	d-axis current proportional gain	Q15	
	para.crt.dkp	d-axis current integral gain	Q15	
	para.crt.qki	q-axis current proportional gain	Q15	
	para.crt.qkp	q-axis current integral gain	Q15	
	para.iq_lim	q-axis current limit value	Q31	
	para.id_lim	d-axis current limit value	Q31	
	para.err_ovc	Overcurrent setting value	Q15	

9.5.4.3 Details of Processing

Initializes the motor control parameter.

Performs the setting of variables which are not changed during control.

9.5.5 DAC Control Initialization (init_Dac)

9.5.5.1 Syntax

void init_Dac(TSB_SC_TypeDef* const SCx)

Argument :

TSB_SC_TypeDef* const SCx: Selects the SC-address.

Return value :

None

9.5.5.2 Details of Processing

DAC IC control initialization

SIO enable

Port initialization

SIO initialization

DAC IC initialization communication

SIO interrupt level setting

Pending SIO interrupt clear

Transmission interrupt enable

9.5.6 Cyclic Timer Initialization (init_Timer_interval4kHz)

9.5.6.1 Syntax

void init_Timer_interval4kHz(void)

Argument :

None

Return value :

None

9.5.6.2 Details of Processing

Default setting of timer for creating 4 kHz cycle timing.

9.5.7 User Control Initialization (init_user_control)

9.5.7.1 Syntax

void init_user_control(void)

Argument :

None

Return value :

None

9.5.7.2 Details of Processing

Default setting for command control from outside such as by user.

- ・Key input processing initialization (init_Uikey)
- ・Analog voltage input processing initialization (init_soft_adc)
- ・LED display processing initialization (init_led)
- ・UART processing initialization (init_uart)

9.5.8 User Control (uart_control)

9.5.8.1 Syntax

void user_control(void)

Argument :

None

Return value :

None

9.5.8.2 Details of Processing

Controls command from outside such as by user.

- ・Timer count for each main period
- ・Rotation speed control switching from volume switch (VR1) input
- ・Motor Rotation Direction Switching from Toggle SW1 input
- ・LED output

9.5.9 User Motor Control (B_User_MotorControl)

9.5.9.1 Syntax

void B_User_MotorControl(void)

Argument :

None

Return value :

None

9.5.9.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	target_spd	Target speed	---	User command
Output	usr.omega_user	Control target speed	Q31	
	usr.com_user.onoff	Motor ON/OFF	---	
	drv.state	Motor error status	---	

9.5.9.3 Details of Processing

Handles the user's commands for motor-related items.

- ・Checks the overcurrent (hardware) status

  Checks the status of hardware overcurrent and update the motor fault status.

- ・Motor ON/OFF command

Determines the motor ON/OFF (usr.com_user.onoff) from target speed (target_spd1) status.

Clears the motor error status when the target speed is 0 Hz.

・Drive speed normalization, etc.

Normalizes the target speed.

9.6 Motor Control Function

The motor control process is implemented by the following functions called from within the main loop of main function:

The motor operation is controlled as a state transition between stop state, bootstrap state, positioning state, forced commutation state, forced to steady-state switching state, steady state, short-circuit brake state, and emergency protection state.

The main stage moves in the following order according to the motor operation starting instruction, this order is positioning (Initposition), forced commutation (Force), forced to steady-state switching (Change_up), and steady state (Steady_A). The sub-stage ranges from Step0 to StepEnd, and at the time of StepEnd the main stage shifts and starts from Step0. If a motor error is detected, the system shifts to the protective shutdown Emergency.

9.6.1 State Transition Processing Function (C_Control_Ref_Model)

9.6.1.1 Syntax

void C_Control_Ref_Model(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.6.1.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	usr.com_user.onoff	Control command	---	
	drv.state.all	Error condition	---	
Input/Output	stage.main	Main stage	---	
	stage.sub	Sub-stage	---	
	usr.com_user_1.onoff	Last control command	---	

9.6.1.3 Details of Processing

Monitors the control commands given by the application and the current state, and then executes state transition.

Each state is divided into more detailed sub-states. The transition of the sub-state is not executed by the state transition processing function, but by the processing function of each state.

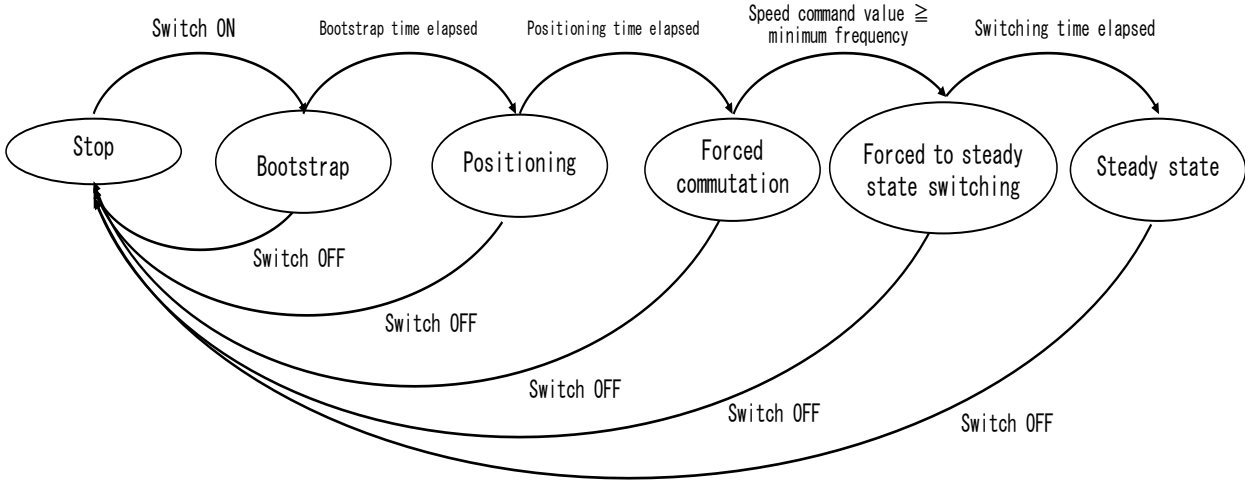


Fig. 4 Motor Control State Transition Diagram4

9.6.2 Motor Control Common Processing Function (C_Common)

9.6.2.1 **Syntax**

void C_Common(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.6.2.2 **Variable**

Direction	Name	Meaning	Q Format	Remarks
Input	usr.com_user.modul	Modulation type control command	---	2-phase or 3-phase modulation
	para.chkpls	Current detection protection Duty range setting	---	
Output	drv.command.modul	Modulation type drive command	---	2-phase or 3-phase modulation
	drv.chkpls	Current detection protection Duty range	---	

9.6.2.3 **Details of Processing**

Executes common processing for each state of motor control.

Performs the calculation of shift PWM control (10.6.10 C_ShiftPWM_Control) and calculation of Vdq (10.6.11 Cal_Vdq) and modulation rate.9.6.109.6.11

The modulation rate is calculated using the following formula.

Modulation rate (Vdq_per) = 100.0 (%) × Vdq average value (Vdq_ave)/Vdc average value (Vdc_ave)

9.6.3 Stop State Function (C_Stage_Stop)

9.6.3.1 Syntax

void C_Stage_Stop(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.6.3.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	stage.main	Main stage	---	
Input/Output	stage.sub	Sub-stage	---	
Output	stage.itr	Interrupt stage	---	
	drv.vector_cmd	Vector control command	---	(Refer to 9.2.2.)
	drv.theta_com	Electric angle command value	Q0	
	drv.theta	Rotor position	Q0	
	drv.omega_com	Drive speed command value	Q31	
	drv.omega	Speed	Q31	
	drv.Id_com	d-axis current command value	Q31	
	drv.Iq_com	d-axis current command value	Q31	

9.6.3.3 Details of Processing

Stops the motor. (stops the PWM output)

9.6.4 Bootstrap State Function (C_Stage_Bootstrap)

9.6.4.1 Syntax

void C_Stage_Bootstrap(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.6.4.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	stage.main	Main stage	---	
	para.time.bootstp	Bootstrap time	Q0	* MainLoopPrd(s)
Input/Output	stage.sub	Sub-stage	---	
	stage_counter	Stage counter	Q0	* MainLoopPrd(s)
Output	stage.itr	Interrupt stage	---	
	drv.vector_cmd	Vector control command	---	(Refer to 9.2.2.)

9.6.4.3 Details of Processing

Outputs the waveform for upper phase All OFF and lower phase All ON and charges the bootstrap capacitor. Continues this process to "bootstrap time". Determines the amount of charge in the capacitor from "bootstrap time".

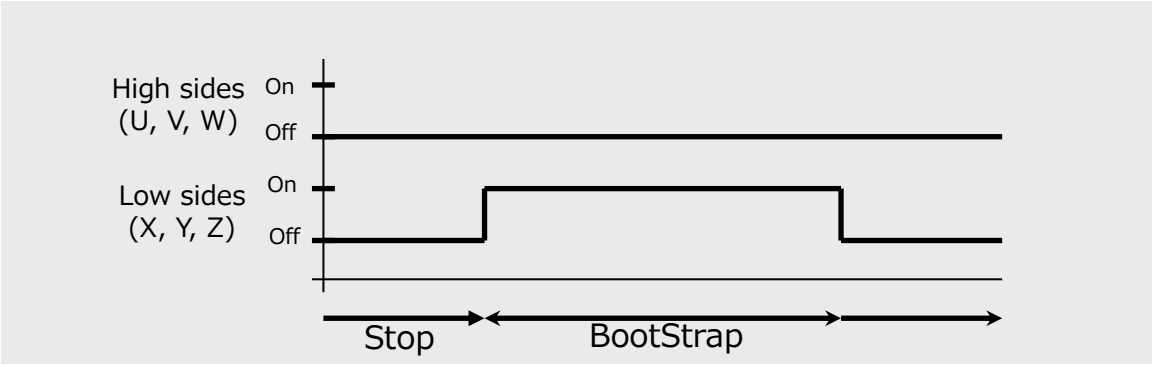
Controls the bootstrap state by dividing it into the following sub-states:

a) Initial state

Initializes the bootstrap state.

b) Delayed state

Waits for the specified bootstrap time to elapse and then transits to the positioning state.



9.6.5 Positioning State Function (C_Stage_Initposition)

9.6.5.1 Syntax

void C_Stage_Initposition(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.6.5.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	stage.main	Main stage	---	
	boot_type	Activation type	---	
	usr.Id_st_user	Starting Id current value	Q15	
	usr.lambda_user	Initial rotor position	Q0	
	para.vd_pos	Positioning command voltage	Q31	
	para.time.initpos	Positioning time	Q0	* MainLoopPrd(s)
	para.time.initpos2	Positioning status wait time	Q0	* MainLoopPrd(s)
Input/Output	stage.sub	Sub-stage	---	
	stage_counter	Stage counter	Q0	* MainLoopPrd(s)
	drv.Vd_out	Output voltage	Q31	Valid only during voltage drive
	drv.Id_com	d-axis current command value	Q31	
Output	stage.itr	Interrupt stage	---	
	drv.vector_cmd	Vector control command	---	(Refer to 9.2.2.)
	drv.Iq_com	d-axis current command value	Q31	
	drv.omega_com	Drive speed command value	Q31	
	drv.theta_com	Electric angle command value	Q0	

9.6.5.3 Details of Processing

Fixs θ to "initial position", ω to 0, and gradually increase Id from 0 while fixing Iq to 0.

This process continues for the "positioning time" and eventually Id becomes the "start Id current value". The amount of increase in Id per unit time is determined from "Positioning time" and "Start Id current value".
After Id reaches "Start Id current value", and after [Positioning wait time] has elapsed, it transitions to the next stage.

The positioning state is divided into the following sub states for control.

a) Initial state
   Performs initialization of positioning state.
b) Id increment state
   Gradually increase Id to the set value.

9.6.6 Forced Commutation State Function (C_Stage_Force)

9.6.6.1 **Syntax**

void C_Stage_Force(vector_t* const _motor)
Argument :
vector_t* const _motor : Motor control structure
Return value :
None

9.6.6.2 **Variable**

Direction	Name	Meaning	Q Format	Remarks
Input	stage.main	Main stage	---	
	boot_type	Activation type	---	Current or voltage
	usr.Id_st_user	Starting Id current value	Q15	
	usr.omega_user	Control target speed	Q31	
	para.omega_v2i	Current control switching speed	Q15	Valid only during voltage dive
	para.spd_coef	Output voltage coefficient	Q15	Valid only during voltage dive
	para.vd_pos	Positioning output voltage	Q31	Valid only during voltage drive
	para.omega_min	Forced commutation termination speed	Q15	
	para.sp_ud_lim_f	Drive speed increase/decrease limit value	Q31	
Input/Output	stage.sub	Sub-stage	---	
	drv.omega_com	Drive speed command value	Q31	
Output	drv.vector_cmd	Vector control command	---	(Refer to 9.2.2.)
	stage.itr	Interrupt stage	---	
	drv.Iq_com	d-axis current command value	Q31	
	drv.Id_com	d-axis current command value	Q31	
	drv.Vd_out	Output voltage	Q31	Valid only during voltage drive

9.6.6.3 Details of Processing

Starts the rotation of the rotor. In this stage, rather than vector-controlled feedback processing, a rotating magnetic field is forced to cause the rotor to rotate accordingly.
The driving speed command value ω_{com} is gradually incremented while the Id is fixed to "start Id current value" and the Iq is fixed to 0.
θ is obtained from ω_{com} ($\theta = \omega_{com}t$). This process continues until ω reaches "Forced commutation termination speed".
The "drive target speed" is increased by a fixed value to approach the "control target speed". "Drive target speed" becomes ω.

9.6.7 Forced to Steady-State Switching State Function (C_Stage_Change_up)

9.6.7.1 Syntax

void C_Stage_Change_up(vector_t* const _motor)
Argument :
vector_t* const _motor : Motor control structure
Return value :
None

9.6.7.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	stage.main	Main stage	---	
	usr.Id_st_user	Starting Id current value	Q15	
	usr.Iq_st_user	Starting Iq current value	Q15	
	usr.omega_user	Control target speed	Q31	
	para.delta_lambda	Current switching phase at the time of change-up	Q0	
	para.sp_ud_lim_f	Drive speed increase/decrease limit value	Q31	
	para.time.go_up	Waiting time after change-up	Q0	* MainLoopPrd(s)
Input/Output	stage.sub	Sub-stage	---	
	stage_counter	Stage counter	Q0	* MainLoopPrd(s)
	drv.omega_com	Drive speed command value	Q31	
Output	stage.itr	Interrupt stage	---	
	drv.vector_cmd	Vector control command	---	(Refer to 9.2.2.)
	drv.Id_com	d-axis current command value	Q31	
	drv.Iq_com	q-axis current command value	Q31	

9.6.7.3 Details of Processing

Decreases Id to 0 and increases Iq to "starting Iq current", and controls so that the direction of the magnetic field is directly below the rotor.
Generates a torque component.
ω and θ are obtained by the position estimation operation.

The "drive target speed" is increased by a fixed value to approach the "control target speed". However, speed control is not performed in this stage, so "drive target speed" is not used for control.

Control is performed by dividing the forced to steady-state switching state into the following sub-states.

a) Initial state
   Performs the initialization of the forced to steady-state switching state.
b) Id, Iq switching status
   Gradually decrements the Id to 0 and gradually increases the Iq to the specified value at the same time. Increasing and decreasing curves are not linear, they are trigonometric curves. After the switch is completed, the transition to the time elapsed wait state is made.
c) Delayed state
   Waits for the specified forced to steady-state switching time to elapse, and then transits to the steady state.

9.6.8 Steady State Function (C_Stage_Steady_A)

9.6.8.1 **Syntax**

void C_Stage_Steady_A(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.6.8.2 **Variable**

Direction	Name	Meaning	Q Format	Remarks
Input	stage.main	Main stage	---	
	usr.omega_user	Control target speed	Q31	
	para.sp_up_lim_s	Drive speed increase limit value	Q31	
	para.sp_dn_lim_s	Drive speed decrease limit value	Q31	
Input/Output	stage.sub	Sub-stage	---	
	drv.omega_com	Drive speed command value	Q31	
Output	drv.vector_cmd	Vector control command	Q0	(Refer to 9.2.2.)
	stage.itr	Interrupt stage	---	
	drv.Id_com	d-axis current command value	Q31	

9.6.8.3 **Details of Processing**

Executes steady-state processing.

The "drive target speed" is increased in fixed increments to get closer to the "control target speed".

9.6.9 Emergency Protection State Function (C_Stage_Emergency)

9.6.9.1 Syntax

void C_Stage_Emergency(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.6.9.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	stage.main	Main stage	---	
Input/Output	stage.sub	Sub-stage	---	
Output	drv.vector_cmd	Vector control command	---	(Refer to 9.2.2.)
	stage.itr	Interrupt stage	---	

9.6.9.3 Details of Processing

Transitions to this state when an overcurrent occurs.

When hardware overcurrent is detected, the motor drive outputs U, V, W, X, Y, and Z becomes all Hi-z.

When software overcurrent is detected, the motor drive outputs U, V, W, X, Y, and Z becomes all OFF.

The rotor rotates with inertia. This stage is maintained until the overcurrent state recovery process is performed.

9.6.10 Shift PWM Control (C_ShiftPWM_Control)

9.6.10.1 Syntax

static void C_ShiftPWM_Control(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.6.10.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	shunt_type	Shunt type	---	
	stage.itr	Interrupt stage	---	
	usr.com_user.spwm	Shift PWM control method	---	
	drv.omega_com	Drive speed command value	Q31	
	para.spwm_threshold	Shift PWM switching speed	Q15	
Output	drv.command.spwm	Shift PWM drive command	---	

9.6.10.3 Details of Processing

This control is only available while using VE and 1-shunt.

Performs ON/OFF setting of shift PWM.

Determines the shift PWM driving procedure from the shift PWM control method, interrupt stage, and target speed. In the sample software, the requirements are set as shown in Table 1.

Table 1 Shift PWM driving method conditions1

Table 1 Shift PWM driving method conditions1

Shift PWM Control Method usr.com_user.spwm	Interrupt 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)
	ciInitposition_i Force_i Change_up Steady_A	Target speed >= Shift switching speed drv.omega_com >= para.spwm_threshold	OFF (0)
		Target speed < Shift switching speed drv.omega_com < para.spwm_threshold	ON (1)

9.6.11 Vdq Calculation (Cal_Vdq)

9.6.11.1 Syntax

q15_t Cal_Vdq(q15_t _vd, q15_t _vq)

Argument :

q15_t _vd : d-axis voltage

q15_t _vq : q-axis voltage

Return value :

q15_t Vdq : dq-axis voltage

9.6.11.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	_vd	d-axis voltage	Q15	
	_vq	q-axis voltage	Q15	
Output	_Vdq	dq-axis voltage	Q15	

9.6.11.3 Details of Processing

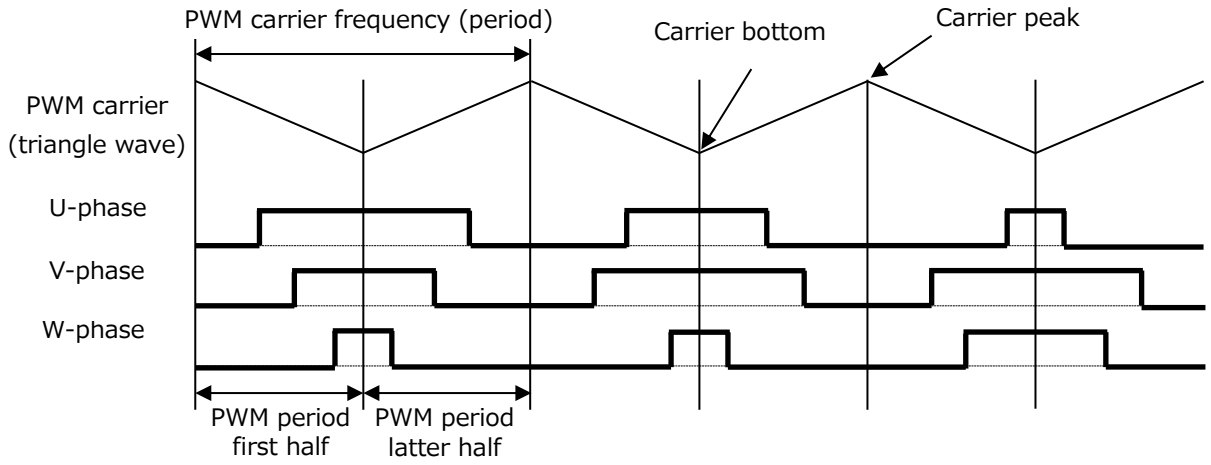
Calculates Vdq using the following formula.

$Vdq = \sqrt{3} \times \sqrt{(Vd^2 + Vq^2)}$

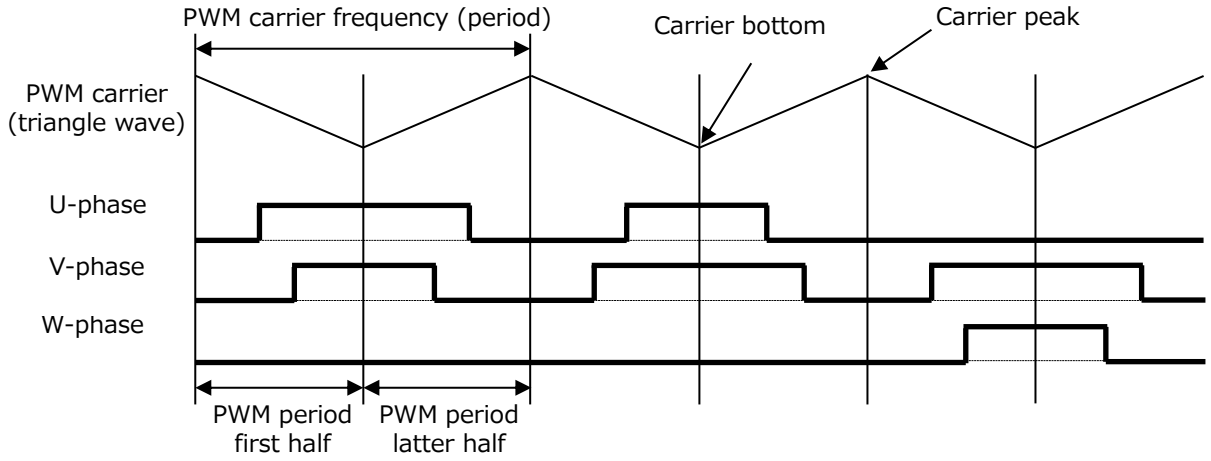
9.7 Motor Drive Function

9.7.1 Definitions

9.7.1.1 For 3-Phase Modulation



9.7.1.2 For 2-Phase Modulation



9.7.2 Position Estimation Function (D_Detect_Rotor_Position)

9.7.2.1 Syntax

void D_Detect_Rotor_Position(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.7.2.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	drv.vector_cmd.F_vcomm_Edetect	Induced voltage control command	---	
	drv.vector_cmd.F_vcomm_omega	Estimated speed control command	---	
	drv.vector_cmd.F_vcomm_theta	Estimated rotor position control command	---	
	drv.Id	d-axis current	Q15	
	drv.Iq	q-axis current	Q15	

	drv.omega_com	Drive speed command value	Q31	
	drv.theta_com	Electric angle command value	Q0	
	drv.Vd	d-axis voltage	Q31	
	para.motor.Lq	Motor q-axis inductance	Q12	
	para.motor.r	Motor winding resistance	Q12	
	para.pos.ctrlprd	Position estimation control period	Q0	
	para.pos.ki	Position estimation integral gain	Q15	
	para.pos.kp	Position estimation proportional gain	Q15	
Input/Output	drv.Ed	d-axis induced voltage	Q15	
	drv.Ed_I	d-axis induced voltage integral value	Q31	
	drv.Ed_PI	d-axis induced voltage PI value	Q31	
	drv.omega	Estimated speed	Q31	
Output	drv.theta	Rotor position	Q0	

9.7.2.3 **Details of Processing**

PI control is performed by taking the estimated speed ω_{est} of the motor drive signal as set variable and the d-axis induced voltage Ed as the process variable.
The target value of Ed is always 0. Thus, the deviation is -Ed.
The rotor position θ (angular) is obtained by integrating the estimated speed ω_{est} obtained by PI-control.

The equivalent circuit equation for the d-axis of the motor can be expressed as follows:

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

(since p = d/dt and Id ≈ constant value, pId=0 can be set.)

Vd : Motor applied voltage Id,Iq : Motor current
ω_{est} : Estimated angular speed
R : Resistance Ld,Lq : Inductance

Therefore, the d-axis induced voltage Ed can be calculated by the following formula.

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

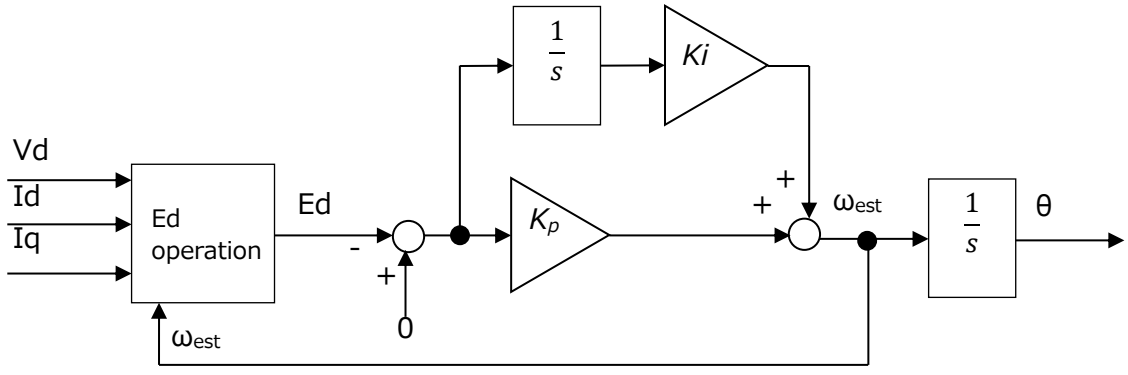


Fig. 5 Position Estimation Block Diagram with d-axis Induced Voltage Ed5

9.7.3 Speed Control Function (D_Control_Speed)

9.7.3.1 Syntax

void D_Control_Speed(vector_t* const _motor)

Argument :

vector_t* const _motor : Motor control structure

Return value :

None

9.7.3.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	drv.vector_cmd.F_vcomm_current	Current command control command	---	
	drv.Id_com	d-axis current command value	Q31	
	drv.Iq_com	q-axis current command value	Q31	
	drv.omega	Estimated speed	Q31	
	drv.omega_com	Drive speed command value	Q31	
	para.id_lim	d-axis current limit value	Q31	
	para.iq_lim	q-axis current limit value	Q31	
	para.spd.ki	Speed control integral gain	Q15	
	para.spd.kp	Speed control proportional gain	Q15	
Input / Output	drv.Iq_ref_I	q-axis current integral value	Q31	
	drv.omega_dev	Speed deviation	Q15	
Output	drv.Id_ref	d-axis current reference value	Q15	
	drv.Iq_ref	q-axis current reference value	Q15	

9.7.3.3 Details of Processing

PI control is performed by taking output frequency ω as the process variable and q-axis current Iq as the set variable.

The d-axis and q-axis current reference values are determined from the speed deviation of the actual measurement of speed command value.

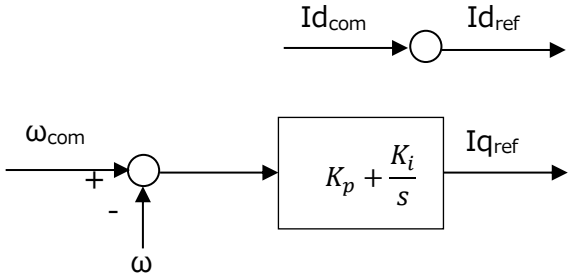


Fig. 6 Speed Control Block Diagram6

9.7.4 3-Shunt Current False Positive Detection Function (D_Check_DetectCurrentError_3shunt)

9.7.4.1 Syntax

int D_Check_DetectCurrentError_3shunt (uint32_t _duty, uint16_t _chkplswidth)

Argument :

_duty Mid-range of Duty

_chkplswidth Pulse check width

Return value :

Current detection status

9.7.4.2 Variable

Direction	Name	Meaning	Q Format	Remarks
Input	_duty	Intermediate PWM Duty fraction	Q15	0.0 to 1.0
	_chkplswidth	Pulse check width	---	
Output	---	Current detection status	---	0:Detectable 1:Undetectable

9.7.4.3 Details of Processing

Detects the timing at which the current cannot be detected by PWM Duty width.
Detects that the intermediate value of Duty is greater than the set value.
This function confirms that Duty range is such that the current cannot be detected. By using the previous value without using the detected value from the current detection process, the control disturbance caused by the current erroneous detection is eliminated.

Since the current detection ADC conversion value of the PWM maximum phase is not used for calculation, it is judged as “Not Good” when the current detection value of the PWM intermediate phase becomes larger than the set value.

9.7.5 Shunt Current False Positive Detection Function (D_Check_DetectCurrentError_1shunt)

9.7.5.1 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)

Argument :

_pwma U-phase PWM Duty

_pwmb V-phase PWM Duty

_pwmc W-phase PWM Duty
_chkplswidth Pulse check width
_sfhpwm Shift PWM mode status
_modu Modulation type
_sector Sector information
_mdprd PWM period

Return value :
Current detection status

9.7.5.2 **Variable**

Direction	Name	Meaning	Q Format	Remarks
Input	_pwma	U-phase PWM Duty	Q15	0.0 to 1.0
	_pwmb	V-phase PWM Duty	Q15	0.0 to 1.0
	_pwmc	W-phase PWM Duty	Q15	0.0 to 1.0
	_chkplswidth	Pulse check width	---	
	_sfhpwm	Shift PWM mode status	---	
	_modu	Modulation type drive command	---	2-phase or 3-phase modulation
	_sector	Sector information	---	
	_mdprd	PWM period	---	
Output	---	Current detection status	---	0:Detectable 1:Undetectable

9.7.5.3 **Details of Processing**

Detects the timing at which the current cannot be detected by PWM Duty width.
Detects that the difference of smallest Duty width and Duty width has become smaller than the set value. Note that the smallest Duty width when 2-phase modulation is performed with 1-shunt is the intermediate Duty width of the 3-phase Duty width.

By confirming that the Duty width of the current which cannot be detected by this function, and using the previous value instead of using the detected value in the current detection process, control disturbance due to current detection miss is supressed.

It is judged as “Not Good Detection” in any of the following cases.

- When the minimum value of PWM Duty width is smaller than the set value
- When the minimum value of the Duty width difference between the two PWMs is smaller than the set value

If the width of the PWM Duty itself and the width of the difference between the PWM Duty are smaller than the set value, it is judged as Not Good Detection.

10. Constant Definition Description

10.1 Argument for Setting the Motor Driver: (D_Para.h)

10.1.1 DAC Output Selection

__USE_DAC : Define this parameter when performing DAC control for outputting variable values for evaluation.

10.1.2 Argument: List

Argument: Name	Description
cMAINLOOP_PRD	Main cycle setting
	Unit [s] resolution 4 kHz
	Set the time of the main cycle.
cIXO_AVE	Filter factor for zero current value
	--
	Set the filter coefficient. Setting a larger value stabilizes, but the reaction is delayed.
cVDQ_AVE	Filter Factor for Supply Voltage Vdc, Output Voltage Vdq
	--
	Set the filter coefficient. Setting a larger value stabilizes, but the reaction is delayed.

10.2 Motor Driver Setting Parameters: Motor Channel (D_Para_ch1.h)

Parameters of the motor driver section: Various motors can be driven by changing these parameters.

10.2.1 Arguments by Motor Channel: List

Argument: Name	Description
cPOLH	Upper phase driver logic setting
	0: Low active/ 1: High active
	Change the value according to the board design. Set the upper phase driver logic.
cPOLL	Lower phase driver logic setting
	0: Low active/ 1: High active
	Change the value according to the board design. Set the logic of the lower phase driver.
cV_MAX	Maximum voltage value setting
	Unit [V]
	Change the value according to the board design. Set the power supply voltage [unit V] equivalent to full scale (4095) with the 12-bit count value after A/D conversion.
cA_MAX	Maximum current value setting

	Unit [A]
	Change the value according to the board design. Set the phase current [unit A] equivalent to full scale (2047) with the 11-bit count value after A/D conversion.
cSHUNT_TYPE	Current acquisition type (3-shunt or 1-shunt) setting
	1: 1-shunt/3: 3-shunt
	Change the value according to the board design. Set the current acquisition type.
cBOOT_TYPE	Startup driving type setting
	cBoot_i: Current type drive/cBoot_v: Voltage type drive
	Set the driving type at startup. Select the voltage type drive in case of 1-shunt drive startup, or if current cannot be obtained.
cSHUNT_ZERO_OFFSET	Offset voltage setting
	Unit [V]
	Set the shunt voltage when no current is flowing. This value is used for the initial value of the zero current average value.
cADCH_CURRENT_U	U-phase current acquisition ADC channel setting (for 3-shunt)
	AINx
	Set the ADC channel for detecting the U-phase current.
cADCH_CURRENT_V	V-phase current acquisition ADC channel setting (for 3-shunt)
	AINx
	Set the ADC channel for detecting the V-phase current.
cADCH_CURRENT_W	W-phase current acquisition ADC channel setting (for 3-shunt)
	AINx
	Set the ADC channel for detecting the W-phase current.
cADCH_CURRENT_IDC	Current acquisition ADC channel setting (for 1-shunt)
	AINx
	Set the ADC channel for detecting the current.
cADCH_VDC	Power supply voltage acquisition ADC channel setting
	AINx
	Set the ADC channel for detecting the power supply voltage Vdc.
cOVC	Overcurrent detection value setting
	Unit [A]
	Set the overcurrent value. When coil current exceeding this set value is detected, the output is turned OFF by software.

cVDC_MINLIM	Minimum value setting of the power supply voltage Vdc
	Unit [V]
	Set the lowest value of the power supply voltage Vdc. If a voltage value smaller than this value is detected, the motor stops.
cVDC_MAXLIM	Maximum value setting of the power supply voltage Vdc
	Unit [V]
	Set the maximum value of the power supply voltage Vdc. If a voltage value larger than this value is detected, the motor stops.
cPWMPRD	PWM Cycle setting
	Unit [us] resolution 25 ns@80 MHz
	Set the PWM carrier period.
cDEADTIME	Dead time value setting
	Unit [us] resolution 0.1 μs@80 MHz
	Set the dead time value.
cREPTIME	Software control skipping count setting
	Units [times] 1 to 15
	Timing of vector calculation software processing can be reduced. When it is set to 1, software processing of vector operation is performed in each PWM cycle. When it is set to 2, software processing of vector operation is performed once in two PWM cycles.
cID_ST_USER_ACT	d-axis starting current setting
	Unit [A]
	Set the d-axis starting current value. Positioning and forced commutation are performed with the current value of this value. Set a value of about 10% of the rated commutation. If the motor does not move, increase the value gradually until it moves.
cIQ_ST_USER_ACT	q-axis starting current setting
	Unit [A]
	Set the value of q-axis starting current. Set the value about 1/2 of the d-axis starting current. If the motor accelerates suddenly when shifting from forced commutation (d-axis control) to steady (q-axis control), reduce the value. If the motor stops, increase the value.

cMOTOR_R	Motor coil resistance
	Unit [Ω]
	Set the resistance value for one phase of the motor coil.
cMOTOR_LQ	q-axis inductance value
	Unit [mH]
	Set the q-axis inductance value of the motor coil.
cMOTOR_LD	d-axis inductance value (not used)
	Unit [mH]
	Set the d-axis inductance value of the motor coil.
cPOLE	Setting of the number of motor poles
	Units [pole]
	Set the number of motor poles.
cID_KP	d-axis current control proportional gain
	Unit [V/A]
	Set d-axis current control proportional gain.
cID_KI	d-axis current control integral gain
	Units [V/As]
	Set the d-axis current control integral gain.
cIQ_KP	q-axis current control proportional gain
	Unit [V/A]
	Set the q-axis current control proportional gain.
cIQ_KI	q-axis current control integral gain
	Units [V/As]
	Set the q-axis current control integral gain.
cPOSITION_KP	Position estimation proportional gain
	Units [Hz/V]
	Set the proportional gain for position estimation.
cPOSITION_KI	Position estimation integral gain
	Units [Hz/Vs]
	Set the integral gain for position estimation.
cSPEED_KP	Speed control proportional gain
	Units [A/Hz]
	Set the proportional gain of the speed control.
cSPEED_KI	Speed control integral gain

	Units [A/Hzs]
	Set the integral gain of speed control.
cSPD_PI_PRD	Speed PI control cycle setting
	[Times]
	Set the speed PI control cycle. When this bit is set to 1, speed-PI calculation is performed for each PWM cycle. When set to 2, speed PI calculation is performed once in two PWM cycles.
cFCD_UD_LIM	Drive Speed Acceleration Rate (Forced Commutation)
	Units [Hz/s]
	Set the speed acceleration rate for forced commutation. If motor rotation does not follow the output of forced commutation, reduce the value.
cSTD_UP_LIM	Drive speed acceleration rate (steady state)
	Units [Hz/s]
	Set the steady state speed acceleration rate.
cSTD_DW_LIM	Drive speed deceleration rate (steady state)
	Units [Hz/s]
	Set the steady state speed deceleration rate.
cBOOT_LEN	Bootstrap waveform output time
	Unit [s] Resolution: 1 ms
	Set the output time of the bootstrap waveform.
cINIT_LEN	Positioning time
	Unit [s] Resolution: 1 ms
	Set the positioning time.
cINIT_WAIT_LEN	Wait time after positioning
	Unit [s] Resolution: 1 ms
	Set the wait time after positioning.
cGOUP_DELAY_LEN	Waiting time after change-up
	Unit [s] Resolution: 1 ms
	Set the wait time after change-up.
cHZ_MAX	Maximum Frequency
	Unit [Hz]
	Set the maximum frequency to be detected by the MCU. Set a value about 10 to 20% higher than the maximum frequency

	used in control. The smaller the value, the more accurate the operation will be, but if the detected value exceeds this value, the control will be broken.
cHZ_MIN	Switching speed from forced commutation to steady state
	Unit [Hz]
	Set the speed to shift from forced commutation to steady state. Set the speed at which position estimation can be performed (induced voltage detected).
cHZ_SPWM	Shift PWM switching speed (for 1-shunt 2-phase modulation)
	Unit [Hz]
	Set the speed to switch from shift PWM to normal PWM. Switchs according to the command speed value.
cID_LIM	d-axis current limit
	Unit [A]
	Set the limit value of d-axis current.
cIQ_LIM	q-axis current limit
	Unit [A]
	Set the limit value of the q-axis current.
cINITIAL_POSITION	Initial position
	Unit [deg]
	Set the angle at the time of positioning by the electrical angle.
cVD_POS	Positioning output voltage during voltage drive
	Unit [V]
	Set the voltage to be used during positioning. Valid only when cBOOT_TYPE is "cBoot_v" For more information, see section 10.2.2.2 Constant Value at Voltage Type Startup.
cSPD_COEF	Forced commutation output voltage coefficient during voltage drive
	Set a value of 0 < x < 1.
	Determines the output voltage during forced commutation. The output voltage is the value obtained by multiplying this setting value by the commanded speed. Vd = cSPD_COEF × Omega_com Valid only when cBOOT_TYPE is "cBoot_v" For more information, see section 10.2.2.2 Constant Value at Voltage Type Startup.

cHZ_V2I	Switching speed from voltage control to current control
	Unit [Hz]
	Set the speed for switching from voltage control to current control. If a value larger than cHZ_MIN is set, and if it becomes cHZ_MIN or higher, it shifts to steady state and switches to current control. Valid only when cBOOT_TYPE is "cBoot_v"
__FIXED_VDC	Fixed setting of power supply voltage
	0:Detected value 1: Fixed value
	To set the power supply voltage to a fixed value, set 1.
cVDC	Fixed value of power supply voltage
	Unit [V]
	Set the value of the power supply voltage. Valid only when __FIXED_VDC is set to "1"

10.2.2 Relationship between Constant Setting Value and Waveform

10.2.2.1 Constant Value at Current Type Start

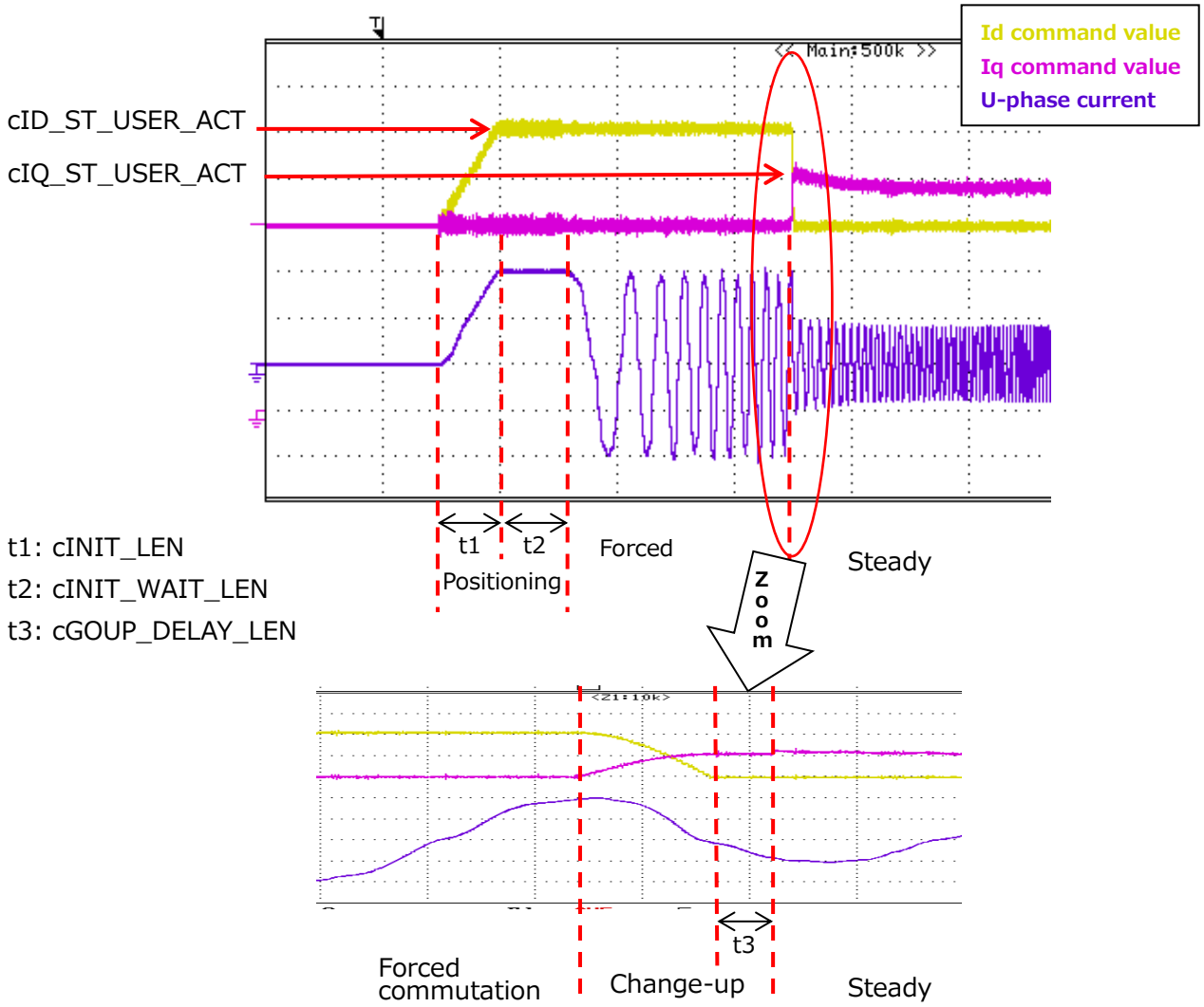


Fig. 7 Starting Current Waveform (Positioned at Electrical Angle 0°)7

In the change-up stage, cID_ST_USER_ACT and cIQ_ST_USER_ACT valuesare swapped.
After swapping, motor is controlled during cGOUP_DELAY_LEN by Iq command value constant.
After steady state transition, the Iq command value is calculated by PI control.

10.2.2.2 Constant Value at Voltage Type Startup

Determine the constant value while checking the current waveform with an oscilloscope, etc.

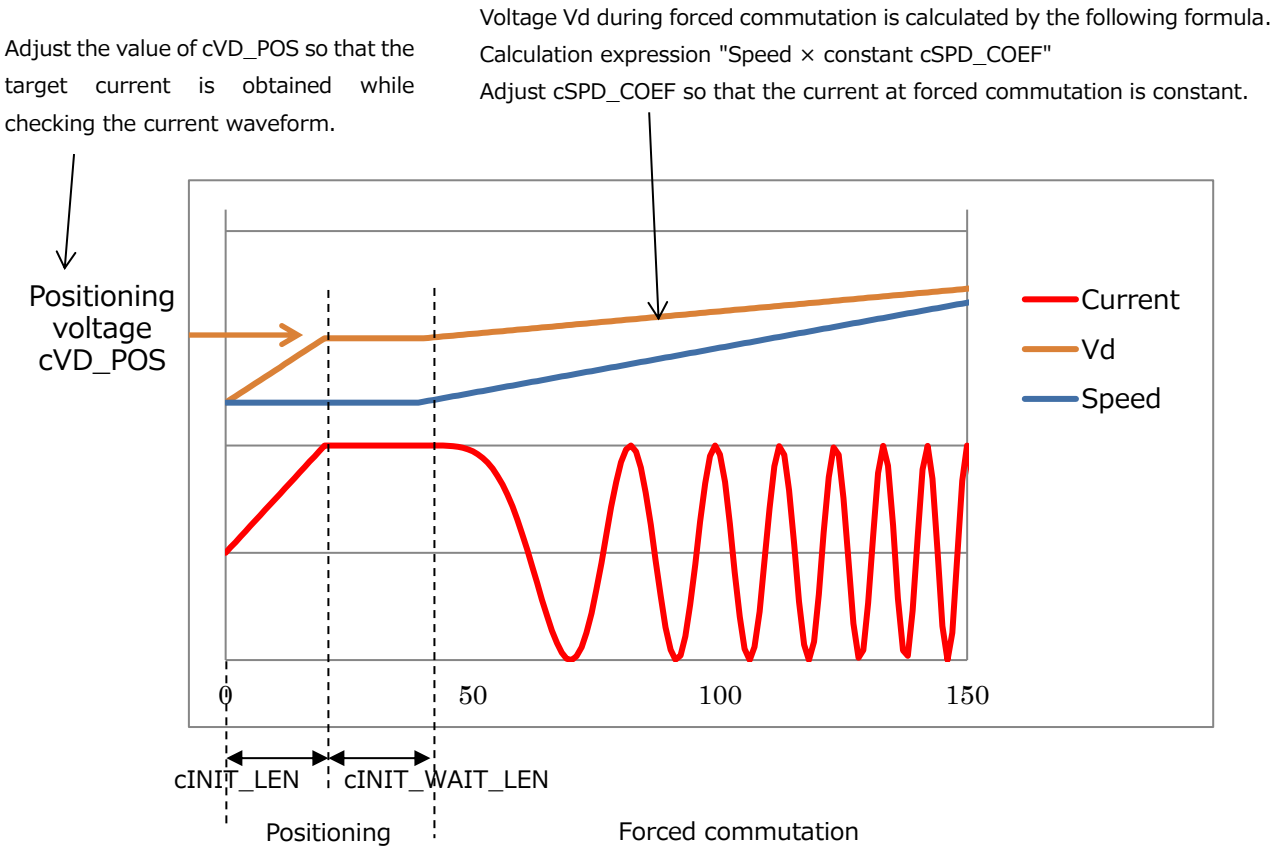


Fig. 8 Arguments for Voltage Drive: How to Adjust8

10.3 User Control Related Constants

usercon.c

```
/* Timer setting */
#define cEMG_S        (1)                    /* LED1 blinking request(over detect(soft)) */
 Soft EMG blinking request
#define cEMG_I     (1)                       /* LED1 blinking request(curret detect error) */
 Current detection error blinking request
#define cEMG_DC       (1)                    /* LED1 blinking request(over vdc) */
 Vdc error blinking request
#define c2000MS_TIMER(2000)                  /* [ms] (4 kHz * 4) * 2000 */
 Communication judgment 2 s timer

/* Key settting */
#define S_SW2 TSB_PE_DATA_PE7                /* SW2 or 9(KEY2) */
        SW2 port :PE7
#define S_SW3 TSB_PK_DATA_PK1                /* GPIO_15 */
        SW3 port :PK1
```

```
#define cKEY_CHATA_CNT   (20)                  /* [cnt] chattering counter for KEY SW */
 Number of chattering removal counters for SW: 20 consecutive matches

/* Soft ADC Setting */
#define cADUNIT_USR    TSB_ADB                 /* User ad data ADCUnit */
       ADC unit for VR1 input: Unit B
#define cADTRG_USR     ADC_TRG_SW
#define cADCH_VR     ADC_AIN11                 /* ADC Channel for VR */
       A/D channels for VR1 inputs: CH11
#define cADREG_VR       ADC_REG5               /* Result register No for VR */
       AD-value storage register for VR1 entry: REG5
#define cADAVECNT       (10)                   /* ADC average count */
       Average count value for ADC value determination (confirmed by 10 acquisitions)

/* Led setting */
#define LED_EMG      TSB_PF_DATA_PF2           /* LED1 */
       EMG LED port: PA0
#define LED_UART_ERR  TSB_PF_DATA_PF4          /* LED3 */
       UART communication LED-port: PF4
#define    cFLASH_TYPE1_CYCLE   (1)            /* [s] led flash cycle type1 */
       LED flashing time 1:1 s
#define    cFLASH_TYPE2_CYCLE   (0.5)          /* [s] led flash cycle type2 */
       LED flashing time 2: 0.5 s
#define    cFLASH_TYPE3_CYCLE   (0.25)         /* [s] led flash cycle type3 */
       LED flashing time 3: 0.25 s
#define    cLED_ON        (1)                  /* LED ON level */
       LED Level: ON
#define    cLED_OFF       (0)                  /* LED OFF level */
       LED Level: OFF

/* User Setting */
#define    cROTATION_CW       (1)             /* Direction: plus */
       Direction of rotation :CW
#define    cROTATION_CCW (0)                  /* Direction: minus */
       Rotation direction: CCW
#define    cCONTROL_SINGLE   (0)              /* UART control: single */
       Communication control mode: Single unit
#define    cCONTROL_UART (1)                  /* UART control: uart */
       Communication control mode: Communication

/* Speed Control Setting */
```

```
#define    cAD_MIN        (0x10)              /* motor speed ADC min value */
        VR1 AD: Min. speed
#define    cAD_MAX        (0xF0)              /* motor speed ADC max value */
        VR1 AD: Max. speed
#define    cSPEED_USER_MIN   (10)             /* [Hz] Min Target speed of motor */
        Motor speed: Minimum speed 12H
#define    cSPEED_USER_MAX  (60)              /* [Hz] Max Target speed of motor */
        Motor speed: Maximum speed 200 Hz

/* UART Setting */
#define UART_ch        UART0                  /* UART Channel */
        UART channels: CH0
#define INTERRUPT_TX    INTTX0_IRQn           /* UART Interrupt request */
        UART0 Transmit Interrupt Requests
#define INTERRUPT_RX    INTRX0_IRQn           /* UART Interrupt request */
        UART0 Receive Interrupt Requests
#define cSEND_DATA_NUM       (7)              /* Send data size */
        Number of send data
#define cRECEIVE_DATA_NUM       (6)           /* Receive data size */
        Number of receive data
#define cUART_RECEIVE_WAIT (0x00)             /* UART mode : data receive wait */
        UART mode: Wait for reception to complete
#define cUART_ERR        (0x01)               /* UART mode : error */
        UART mode: Communication failure
#define cREQ_SYSTEM_START       (0x10)        /* System start request */
        System startup request command
#define cREQ_ROTATE_MOTOR       (0x11)        /* Target speed update request */
        Target speed update request command
#define cGET_MOTOR_ENABLE       (0x80)        /* Operating status */
        Motor operation EN/DI retrieval command (*Not used in this system)
#define cGET_STATE_EMG        (0x81)          /* Emergency status */
        EMG status acquisition command
#define cGET_STAGE          (0x82)            /* Main stage */
        Main Stage Acquisition Command
#define cGET_CONTROL_CH      (0x83)           /* Control channel */
        Motor control CH acquisition command
#define cGET_CARRIER_FREQUENCY  (0x84)        /* Carrier frequency */
        Carrier Frequency Acquisition Command
#define cGET_MOTOR_SPEED_MIN (0x85)           /* Minimum rotation speed */
        Minimum speed acquisition command
#define cGET_MOTOR_SPEED_MAX(0x86)            /* Maximum rotation speed */
```

```
        Maximum speed acquisition command
#define cGET_DEAD_TIME      (0x87)          /* Dead time */
        Dead Time Acquisition Command
#define cGET_GATE_ACTIVE    (0x88)          /* Gate active */
        Gate active acquisition command
#define cGET_POSITION_DITECT (0x89)         /* Position detect */
        Shunt Type Acquisition Command
#define cGET_VDC_VOLTAGE    (0x8A)          /* VDC voltage */
        Power supply voltage acquisition command
#define cGET_INVETER_TEMP   (0x8B)          /* Inverter temp */
        Temperature acquisition command
#define cGET_U_VOLTAGE      (0x8C)          /* U-phase voltage */
        U-phase voltage acquisition command
#define cGET_V_VOLTAGE      (0x8D)          /* V-phase voltage */
        V-phase voltage acquisition command
#define cGET_W_VOLTAGE      (0x8E)          /* W-phase voltage */
        W-phase voltage acquisition command
#define cGET_DAC_DATA       (0x8F)          /* Daca date */
        DAC data acquisition command
#define cGET_INTERNAL_AMP   (0x90)          /* Internal amp */
        Internal amplifier acquisition command
#define cGET_DIRECTION      (0x91)          /* Direction */
        Rotation direction acquisition command
#define cGET_MODULATION     (0x92)          /* Modulation */
        Get Modulation Command
#define cGET_KEY_OPERATION (0x93)           /* Key operation */
        Rotation control acquisition command (*Not used in this system)
#define cGET_MOTOR_SPEED    (0x94)          /* motor rotation speed */
        Motor speed acquisition command
#define cEMG_STATE_EMG_H    (0x00)          /* emergency status : over detect(hard) */
        EMG state: Hard EMG
#define cEMG_STATE_EMG_S    (0x01)          /* emergency status : over detect(soft) */
        EMG state: Soft EMG
#define cEMG_STATE_EMG_I    (0x02)          /* emergency status : curret detect error */
        EMG state: Abnormal current detection
#define cEMG_STATE_EMG_DC   (0x03)          /* emergency status : over vdc */
        EMG state: VDC error
#define cCONTROL_CH_CH0     (0x00)          /* Motor control channel : ch0 */
        Motor control: CH0
#define cCONTROL_CH_CH1     (0x01)          /* Motor control channel : ch1 */
        Motor drives :CH1
```

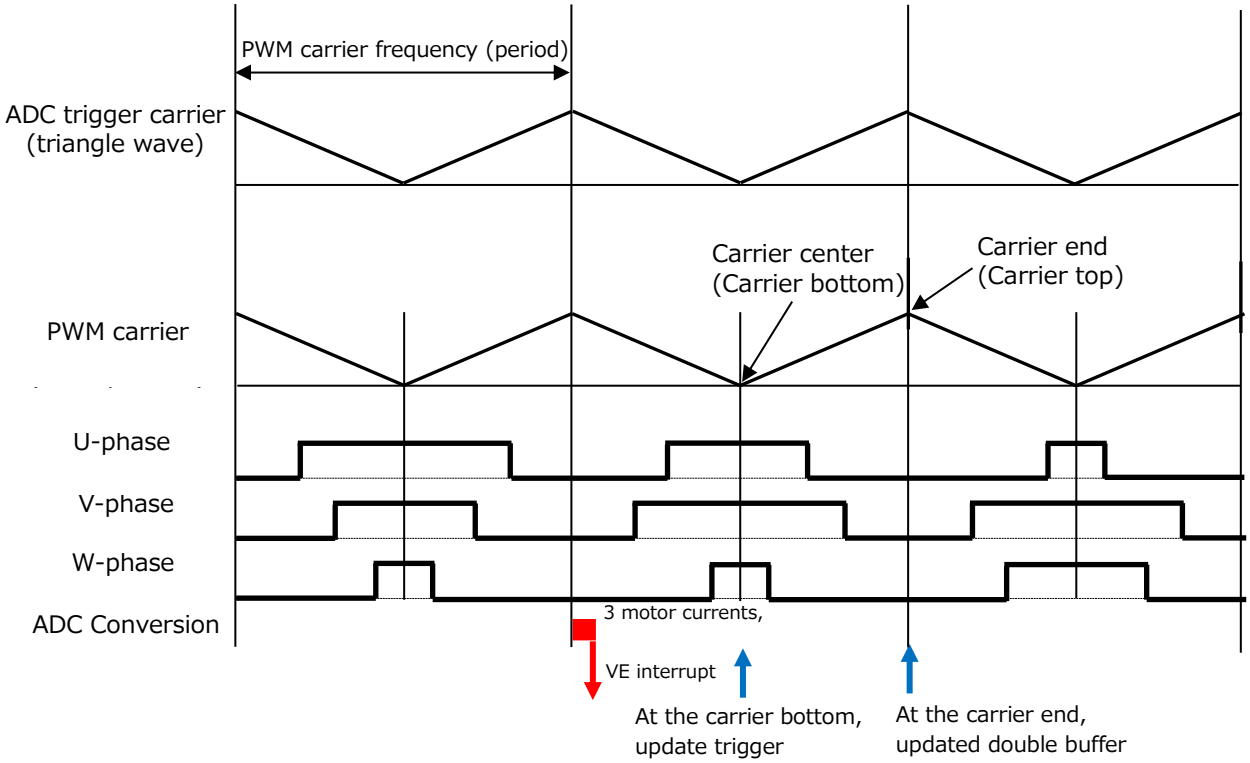
```
#define cCONTROL_CH_CH2     (0x02)              /* Motor control channel : ch2 */
        Motor drives :CH2
#define cCONTROL_CH    (cCONTROL_CH_CH1)  /* User Motor control channel */
        Motor control used
#define cGATE_ACTIVE_H_H    (0)                 /* Gate active : H/H */
        Gate active: H/H
#define cGATE_ACTIVE_L_L    (1)                 /* Gate active : L/L */
        Gate active: L/L
#define cGATE_ACTIVE_H_L    (2)                 /* Gate active : H/L */
        Gate active: H/L
#define cGATE_ACTIVE_L_H    (3)                 /* Gate active : L/H */
        Gate active: L/H
#define cPOSITION_DETECT_3SHUNT(0)          /* Position Ditect : 3shunt */
        Shunt type: 3-shunt
#define cPOSITION_DETECT_1SHUNT(1)          /* Position Ditect : 1shunt */
        Shunt type: 1-shunt
#define cDAC_DATA_TMPREG0      (0x00)           /* Dac data : TMPREG0 */
        DAC data: U-phase current
#define cDAC_DATA_TMPREG1      (0x01)           /* Dac data : TMPREG1 */
        DAC data: V-phase current
#define cDAC_DATA_TMPREG2      (0x02)           /* Dac data : TMPREG2 */
        DAC data: W-phase current
#define cDAC_DATA_THETAHALF    (0x03)           /* Dac data : theta.half[1] */
        DAC data: electrical angle
#define cDAC_DATA_IDREF        (0x04)           /* Dac data : Id_ref */
        DAC data: Id reference (target value)
#define cDAC_DATA_ID           (0x05)           /* Dac data : Id */
        DAC data: Id (current value)
#define cDAC_DATA_IQREF        (0x06)           /* Dac data : Iq_ref */
        DAC Data: Iq Reference (Target Value)
#define cDAC_DATA_IQ           (0x07)           /* Dac data : Iq */
        DAC data: Iq (current value)

#define cDAC_DATA_OMEGACOMHALH    (0x08)    /* Dac data : omega_com.half[1] */
        DAC data: Angular speed (target value)
#define cDAC_DATA_OMEGAHALF   (0x09)        /* Dac data : omega.half[1] */
        DAC data: Angular speed (current value)
#define cDAC_DATA_OMEGADEV    (0x0A)        /* Dac data : omega_dev */
        DAC data: Angular speed (difference)
#define cINTERNAL_AMP_NO       (0)              /* Internal amp : External */
        Internal amplifier: Not used
```

```
#define cINTERNAL_AMP_YES       (1)             /* Internal amp : Internal */
        Internal Amplifier: Used
#define cDIRECTION_CW     (0)                   /* Direction : plus */
        Direction of rotation :CW
#define cDIRECTION_CCW      (1)                 /* Direction : minus */
        Rotation direction: CCW
```

11. Timing of Control and Data Update

11.1 Vector Control using VE

11.1.1 3-Shunt Control



11.1.1.1 **Carrier Waveform**

Type	Waveform
PWM	Triangle wave for all three phases
ADC Trigger	Sawtooth wave

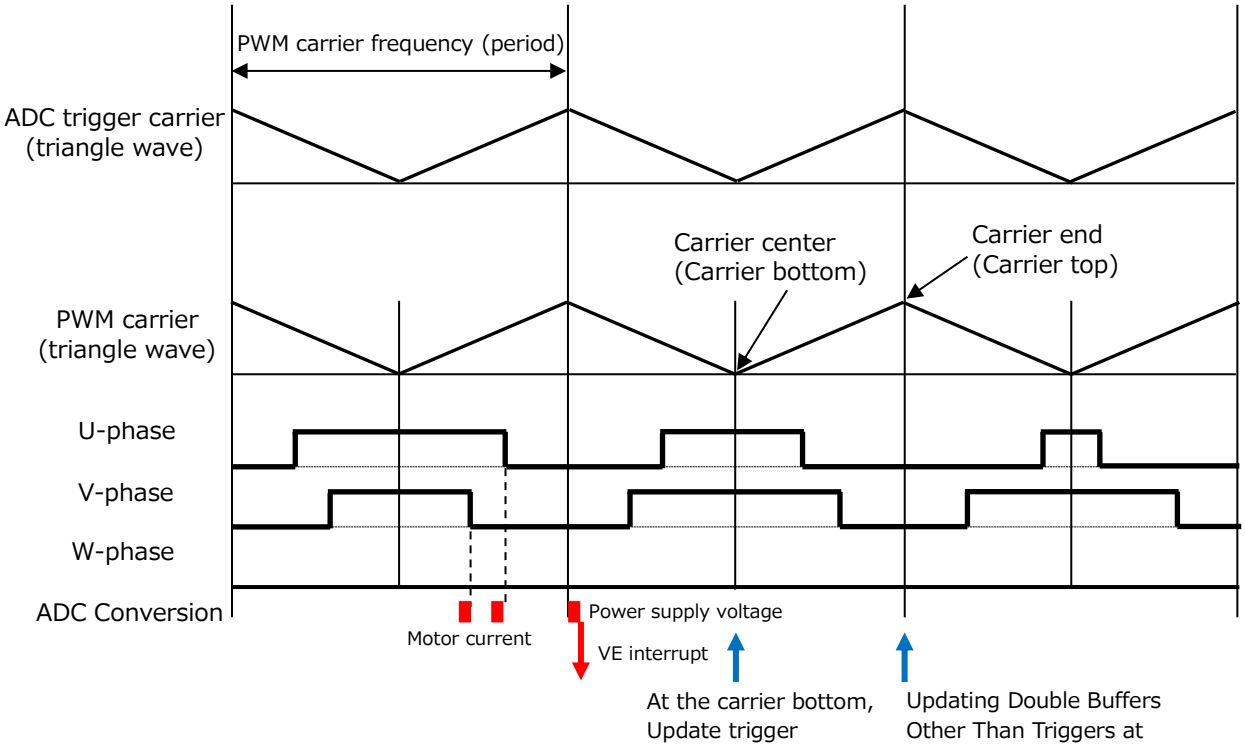
11.1.1.2 **Double Buffer Update Timing**

Data	Update Timing
PWM-period (RATE)	Carrier end
Duty (CMPU,CMPV,CMPW)	Carrier end
Trigger Position (TRGCMPx)	Carrier bottom
Output Setting (MDOUT)	Carrier end

11.1.1.3 **Interrupt Related Extensions**

Interrupt request	Interrupt	Timing
VE	Enable	After completion of VE input schedule
PWM(PMD)	Disable	Once, twice, every four times
End of ADC Conversion	Disable	After completion of all ADC conversion of motor current × 3 and power supply voltage
ADC Trigger	—	Same frequency as PWM

11.1.2 1-Shunt Control



11.1.2.1 **Carrier Waveform**

Type	Waveform
PWM	Triangle wave for all three phases
ADC Trigger	Sawtooth wave

11.1.2.2 **Double Buffer Update Timing**

Data	Update Timing
PWM-period (RATE)	Carrier end
Duty (CMPU,CMPV,CMPW)	Carrier end
Trigger Position (TRGCMPx)	Carrier bottom
Output Setting (MDOUT)	Carrier end

11.1.2.3 **Interrupt Related Extensions**

Interrupt Request	Interrupt	Timing
VE	Enable	After completion of VE input schedule
PWM(PMD)	Disable	Once, twice, every four times
End of ADC conversion	Disable	After completion of ADC conversion of power supply voltage
ADC Trigger	—	Same frequency as PWM

12. Peripheral Driver

12.1 IP Table

This bit defines the base address of the MCU peripheral circuit registers to be passed to the peripheral driver API.

12.1.1 Data Structure

12.1.1.1 Ipdrv_t

Data Fields

TSB_VE_TypeDef* const **VEx:** Selects the VE address.

TSB_PMD_TypeDef* const **PMDx:** Select the PMD address.

TSB_AD_TypeDef* const **ADx:** Selects the ADC address.

12.2 Vector Engine (VE)

12.2.1 Function Specifications

12.2.1.1 IP_VE_init

VE initialization

API:

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

Argument :

VEx: Selects the VE address.

_initdata: VE initialization data structure

For more information, refer to section 12.2.2.1 VE_InitTypeDef.

Function :

Performs initialization of VE.

Supplement :

Call with VE stop and interrupt disabled.

Return value :

None

12.2.1.2 VE_Start

VE Start

API:

VE_Start(const ipdrv_t* const _ipdrv)

Argument :

_ipdrv: Select the IP table address.

Function :

Start VE.

Supplement :

Nothing in particular

Return value :

None

12.2.1.3 **VE_GetPhaseCurrent**

Phase current acquisition

API:

Void
VE_GetPhaseCurrent(const ipderv_t* const _ipdrv,
q15_t* _ia, q15_t* _ib, q15_t* _ic)

Argument :

_ipdrv: Select the IP table address.

_ia: Set the variable address to store the a-phase current.

_ib: Set the variable address to store the b-phase current.

_ic: Sets the variable address to store the c-phase current.

Function :

Acquires the current value of each phase.

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

Supplement :

Nothing in particular

Return value :

None

12.2.1.4 **VE_GetCurrentAdcData**

Current ADC value acquisition

API:

void
VE_GetCurrentAdcData(const ipdrv_t* const _ipdrv,
uint32_t* _adc_ia, uint32_t* _adc_ib, uint32_t* _adc_ic)

Argument :

_ipdrv: Select the IP table address.

_adc_ia: Set the variable address to store the a-phase current ADC value.

_adc_ib: Set the variable address to store the b-phase current ADC value.

_adc_ic: Set the variable address to store the c-phase current ADC value.

Function :

Current A/D value of each phase is acquired from the values of IAADCx, IBADCx, ICADCx register.

Supplement :

Nothing in particular

Return value :

None

12.2.1.5 VE_GetdataFromVEreg

VE register data acquisition

API:

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

Argument :

_ipdrv: Select the IP table address.

_motor: Select the structure address of the vector control variable.

Function :

The values of motor power supply voltage Vdc, d-axis voltage Vd, q-axis voltage Vq, d-axis current Id, and q-axis current Iq are stored from the VE register to each variable of vector control.

When the current cannot be detected due to Duty range, the d-axis current Id and q-axis current Iq are written to the VE register.

Supplement :

Nothing in particular

Return value :

None

12.2.1.6 VE_GetPWM_DutyMed

Duty intermediate value acquisition

API:

uint32_t VE_GetPWM_DutyMed(const ipdrv_t* const _ipdrv)

Argument :

_ipdrv: Select the IP table address.

Function :

Acquires the middle value of the U, V, and W-phase Duty.

Supplement :

Nothing in particular

Return value :

Intermediate Duty value

12.2.1.7 VE_GetOutPutMode

PWM output status acquisition

API:

int VE_GetOutputMode(const ipdrv_t* const _ipdrv)

Argument :

_ipdrv: Select the IP table address.

Function :

Acquires the PWM output status.

Supplement :

Nothing in particular

Return value :

PWM state OCRMD_OUT_OFF: OutputOFF
OCRMD_OUT_ON: OutputON
OCRMD_OUT_ON_LOWPH: Only the lower phase is ON.

12.2.1.8 VE_SetdataToVEreg_Stop

Data set (Stop) for the VE register

API:

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

Argument :

_ipdrv: Select the IP table address.

_motor: Set the structure address of the vector control variable.

Function :

This register sets the VE register of the stop state.
Initializes the current control gain integral value register, etc.

Supplement :

Nothing in particular

Return value :

None

12.2.1.9 VE_SetdataToVEreg_Bootstrap

Data set (Bootstrap) for the VE register

API:

void
VE_SetdataToVEreg_ Bootstrap(const ipdrv_t* const _ipdrv,
const vector_t* const _motor)

Argument :

_ipdrv: Select the IP table address.

_motor: Set the structure address of the vector control variable.

Function :

This register sets the VE register of the bootstrap state.
In VE, a waveform that turns ON only on the L side is output by setting the 2-phase modulation and the output voltage to 0.

Supplement :

Nothing in particular

Return value :

None

12.2.1.10 VE_SetdataToVEreg_Initposition_i

Data Set for VE Register (Initposition Current Control Types)

API:

void

VE_SetdataToVEreg_ Initposition_i (const ipdrv_t* const _ipdrv,
const vector_t* const _motor)

Argument :

_ipdrv: Select the IP table address.

_motor: Set the structure address of the vector control variable.

Function :

Performs the setting processing of the VE register of positioning state of current control type.

Supplement :

Nothing in particular

Return value :

None

12.2.1.11 VE_SetdataToVEreg_Initposition_v

Data Set for VE Register (Initposition Voltage Control Type)

API:

void

VE_SetdataToVEreg_ Initposition_v (const ipdrv_t* const _ipdrv,
const vector_t* const _motor)

Argument :

_ipdrv: Select the IP table address.

_motor: Set the structure address of the vector control variable.

Function :

Performs the setting processing of the VE register of positioning state of voltage control type.

Supplement :

Nothing in particular

Return value :

None

12.2.1.12 VE_SetdataToVEreg_Force_i

Data set for VE register (forced commutation current control type)

API:

void

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

Argument :

_ipdrv: Select the IP table address.

_motor: Set the structure address of the vector control variable.

Function :

Performs the setting processing of the VE register of forced commutation state of current control type.

Supplement :

Nothing in particular

Return value :

None

12.2.1.13 VE_SetdataToVEreg_Force_v

Data set for VE register (forced commutation voltage control type)

API:

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

Argument :

_ipdrv: Select the IP table address.

_motor: Set the structure address of the vector control variable.

Function :

Performs the setting processing of the VE register of forced commutation state of voltage control type.

Supplement :

Nothing in particular

Return value :

None

12.2.1.14 VE_SetdataToVEreg_Change_up

Data set for the VE register (change-up)

API:

void
VE_SetdataToVEreg_ Change_up (const ipdrv_t* const _ipdrv,
const vector_t* const _motor)

Argument :

_ipdrv: Select the IP table address.

_motor: Set the structure address of the vector control variable.

Function :

Performs the setting processing of the VE register of change-up state.

Supplement :

Nothing in particular

Return value :

None

12.2.1.15 VE_SetdataToVEreg_Steady_A

Data set for VE register (steady)

API:

void
VE_SetdataToVEreg_Steady_A (const ipdrv_t* const _ipdrv,
const vector_t* const _motor)

Argument :

_ipdrv: Select the IP table address.

_motor: Set the structure address of the vector control variable.

Function :

Performs the setting processing of VE register of steady state.

Supplement :

Nothing in particular

Return value :

None

12.2.1.16 VE_SetdataToVEreg_Emergency

Data set (EMG) for the VE register

API:

void
VE_SetdataToVEreg_ Emergency (const ipdrv_t* const _ipdrv,
const vector_t* const _motor)

Argument :

_ipdrv: Select the IP table address.

_motor: Sets the structure address of a vector control variable.

Function :

Performs setting processing of VE register of Emergency state.

Supplement :

Nothing in particular

Return value :

None

12.2.1.17 VE_SetZeroCurrentData

Zero current setting

API:

void
VE_SetZeroCurrentData(const ipdrv_t* const _ipdrv,
uint32_t _z_ia, uint32_t _z_ib, uint32_t _z_ic)

Argument :

_ipdrv: Select the IP table address.

_z_ia: Set a-phase zero current ADC value.

_z_ib: Set b-phase zero current ADC value.

_z_ic: Set c-phase zero current ADC value.

Function :

Sets the ADC value at zero current in the VE register.

Supplement :

Nothing in particular

Return value :

None

12.2.1.18 **VE_SetVDCreg**

Setting of DC link voltage Vdc

API:

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

Argument :

_ipdrv: Select the IP table address.

_dat: Set the motor power voltage.

Function :

Sets the power supply voltage of the motor in the VE register.

Supplement :

Nothing in particular

Return value :

None

12.2.1.19 **VE_SetModulType**

Modulation type setting

API:

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

Argument :

_ipdrv: Select the IP table address.

_dat: Set the modulation type.

Function :

Sets the modulation type in the VE register.

Supplement :

Nothing in particular

Return value :

None

12.2.2 Data Structure

12.2.2.1 **VE_InitTypeDef**

Data Fields:

uint8_t ve_ch: Vector engine channel

0:Channel 0
1:Channel 1

uint8_t shunt: Shunt type
3:3-shunt
1:1-shunt

uint16_t pwmfreq: Carrier Frequency

uint16_t reptime: Number of repeats (1 to 15)

uint16_t trgmode: Activation trigger
TRGMODE_UNITA: Activated by ADCA PMD0 trigger synchronous conversion completion interrupt
TRGMODE_UNITB: Activated by ADCB PMD1 trigger synchronous conversion completion interrupt

uint16_t tpwm: PWM period (for phase interpolation)
Value set in VETPWM register

uint16_t idkp: d-axis current control proportional gain

uint16_t idki: d-axis current control integral gain

uint16_t iqkp: q-axis current control proportional gain

uint16_t iqki: q-axis current control integral gain

uint16_t zerooffset: Zero current offset

12.3 Motor Control Circuit (PMD)

12.3.1 Function Specifications

12.3.1.1 **IP_PMD_init**

PMD Initialization

API:

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

Argument :

PMDx: Select the PMD address.

_initdata: PMD initialization data structure
For more information, refer to section 12.3.2.1 PMD_InitTypeDef.

Function :

Performs the initialization of the PMD.

Supplement :

Call while PMD is stopped and interrupt disabled.

Return value :

None

12.3.1.2 **PMD_GetEMG_Status**

EMG protection status acquisition

API:

emg_status_e PMD_GetEMG_Status(const ipdrv_t* const _ipdrv)

Argument :

_ipdrv: Select the IP table address.

Function :

Acquires the EMG protection status.

Supplement :

Nothing in particular

Return value :

emg_status_e: EMG protection status

cNormal: Normal

cEMGProtected: Disables the PWM output due to EMG occurance

12.3.1.3 PMD_ReleaseEMG_Protection

EMG protection status release

API:

void PMD_ReleaseEMG_Protection(const ipdrv_t* const _ipdrv)

Argument :

_ipdrv: Select the IP table address.

Function :

Cancels the EMG protection status.

Supplement :

Even if this function is called, if MDOUT is 0 and the EMG port is not H, the emergency protection status is not released.

Return value :

None

12.3.2 Data Structure

12.3.2.1 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

Set to PMDxMDPDR

uint16_t **Deadtime:** Dead time

Set to PMDxDTR

12.4 Analog-to-Digital Converter (ADC)

12.4.1 Function Specifications

12.4.1.1 **IP_ADC_init**

ADC initialization

API:

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

Argument :

ADx: Selects the ADC address.

_initdata: ADC initialization data structure

For more information, refer to section 12.4.2.1 AD_InitTypeDef.

Function :

Initializes ADC.

Supplement :

Call while ADC is stopped and interrupt disabled.

Return value :

None

12.4.2 Data Structure

12.4.2.1 **AD_InitTypeDef**

Data Fields:

uint8_t shunt: Shunt type
3:3-shunt
1:1-shunt

uint8_t iuch: U-phase current capture ADC channel number (for 3-shunt)

uint8_t ivch: V-phase current capture ADC channel number (for 3-shunt)

uint8_t iwch: W-phase current capture ADC channel number (for 3-shunt)

uint8_t idcch: DC current capture ADC channel number (for 1-shunt)

uint8_t vdcch: Motor Power Voltage Vdc Capture ADC Channel Number

uint8_t pmd_ch: Select PMD channel to be used
cPMD: PMD-channel 1

uint8_t pints: Interrupt selection for PMD triggering
cPINTS_B:ITTADxPDB

12.4.3 Parameters for VE equipped MCU (mcuip_drv.h)

12.4.3.1 **PMD**

For setting MDCR register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cPWMEXCLK	PWM period extension mode specification	PWMCK_NORMAL	Normal cycle
		PWMCK_4FOLD	Quadruple period

cPWMSYNT	Port output mode setting	SYNTMD_0	
		SYNTMD_1	
cPWMSPCFY	Duty mode select	DTYMD_COMMON	Common for all three phases
		DTYMD_INDEPENDENT	Independent for all three phases
cPWMINT	PWM interrupt request timing selection	PINT_BOTTOM	Interrupt request generated at PWM carrier bottom PMD1MDCNT<MDCNT[15:0]>= 0x0001
		PINT_TOP	Interrupt request at PWM carrier peak occurrence PMD1MDCNT<MDCNT[15:0]>=<MDPRD[15:0]>
cPWMINTPRD	PWM interrupt request period selection	INTPRD_HALF	PWM 0.5 interrupt request for each cycle
		INTPRD_1	Interrupt request for each PWM 1 period
		INTPRD_2	Interrupt request for each PWM 2 period
		INTPRD_4	Interrupt request for each PWM 4 period
cPWMWAVE	PWM carrier waveform selection	PWMMD_SAWTOOTH	Edge PWM, sawtooth wave
		PWMMD_TRIANGULAR	Center PWM, triangle wave

For setting MDPOT register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cPSYNCS	MDOUT setting transfer timing selection	PSYNCS_WRITE	PWM asynchronous (reflected when writing)
		PSYNCS_BOTTOM	Carrier bottom MDCNT= 1
		PSYNCS_TOP	Carrier peak MDCNT = PMDxMDPRD<MDPRD>
		PSYNCS_BOTTOM_TOP	Carrier peak and carrier bottom MDCNT = 1 or PMDxMDPRD<MDPRD>

For setting PORTMD register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cPORTMD	Setting of port control during a tool break	PORTMD_ALLHIZ	Upper phase High-z/Lower phase High-z
		PORTMD_UHIZ_LON	Upper phase High-z/Lower phase PMD output
		PORTMD_UON_LHIZ	Upper phase PMD output/Lower phase High-z
		PORTMD_ALLON	Upper phase PMD output/Lower phase PMD output

For setting TRGCR register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cTRGMD_3SHT	Trigger timing	TRGMD_DIS	Trigger output disable

	at 3-shunt	TRGMD_DOWN	Trigger output at match during down counting
		TRGMD_UP	Trigger output at match in up counting
		TRGMD_UPDOWM	Trigger output at up/down counting
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak/bottom
		TRGMD_DIS7	Trigger output disable
cTRGMD_1SHT	Trigger timing at 1-shunt	TRGMD_DIS	Trigger output disable
		TRGMD_DOWN	Trigger output at match during down counting
		TRGMD_UP	Trigger output at match in up counting
		TRGMD_UPDOWM	Trigger output at up/down counting
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak/bottom
		TRGMD_DIS7	Trigger output disable
cBUFSYNC	Buffer asynchronous update enable	TRGBE_SYNC	Synchronous Updates
		TRGBE_ASYNC	Asynchronous update

For setting TRGMD register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cEMGTGE	Output enable setting during EMG protection operation	EMGTGE_DISABLE	Trigger output disable during protection operation
		EMGTGE_ENABLE	Trigger output enable during protection operation

For setting EMGCR register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cEMGCNT	EMG input detection time	0 ～ 15	Noise rejection time setting for error detection input (EMG) cEMGCNT × 16/fsys cEMGCNT >= 0 ～ 15 When "0" is set, the noise filter is turned on.
cINHEN	During a tool break enable/disable	INHEN_DISABLE	Disable
		INHEN_ENABLE	Enable

cEMGMD	EMG protection mode select	EMGMD_ALLHIZ	Full phase high impedance
		EMGMD_UON_LHIZ	All upper phase on/all lower phase high impedance
		EMGMD_UHIZ_LON	All upper phases high impedance/all lower phases on
		EMGMD_ALLHIZ2	Full-phase high impedance

For setting OVVCR register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cOVVCNT	OVV input detection time	0 ～ 15	Noise rejection time setting for OVV input cOVVCNT × 16/fsys cOVVCNT >= 0 ～ 15 When this bit is set to "0", this bit becomes 1.
cADIN1EN	ADC (unit B) monitoring function 1 input enable	ADIN1EN_DISABLE	Input disable
		ADIN1EN_ENABLE	Input enable
cADIN0EN	ADC (unit A) monitoring function 0 input enable	ADIN0EN_DISABLE	Input disable
		ADIN0EN_ENABLE	Input enable
cOVVMD	OVV protection mode select	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 phase off
cOVVISEL	OVV terminal input control	OVVISEL_PORT	Port input
		OVVISEL_ADC	ADC monitoring signal
cOVVRS	Return from the OVV emergency protection state	OVVRS_NORMAL	Disable automatic recovery from emergency protection state
		OVVRS_AUTO	Enable automatic recovery from emergency protection state

12.4.3.2 **VE**

cTADC ADC conversion time for 1-shunt shift PWM Please set up.

For setting FMODE register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cMREGDIS	SIN/COS/SECTOR previous value retention selection	MREGDIS_EFFECTIVE	Enable
		MREGDIS_NOEFFECTIVE	Disable
cADCSEL1	VE-channel 1 ADC unit selection	ADCSEL_UNITB	Unit B
		ADCSEL_UNITAB	Unit AB
cADCSEL0	VE-channel 0 ADC unit	ADCSEL_UNITA	Unit A
		ADCSEL_UNITAB	Unit AB

	selection		

For setting MODE register

Constant Name	Meaning of Constant	Selected Data	Meaning of the Selected Data
cZIEN	Zero current detection control	ZIEN_DISABLE	Normal current detection
		ZIEN_ENABLE	Zero current detection
cPVIEN	Phase interpolation control	PVIEN_DISABLE	Disable
		PVIEN_ENABLE	Enable

13. Appendix

13.1 Fixed-Point Processing

In this sample software, the decimal arithmetic is performed with a fixed decimal point, so the fixed point arithmetic operation is outlined.

13.2 Normalization (Normalize)

Normalization is to transform the data according to a certain rule to make it easier to use the data.

In this design, the most significant bit is the sign bit (0 is positive and 1 is negative), and the decimal point is placed between the next bit, and normalization is performed so that the maximum value (0x7fff) at which the data can be obtained is 1 and the minimum value (0x8000) is-1.

Example)

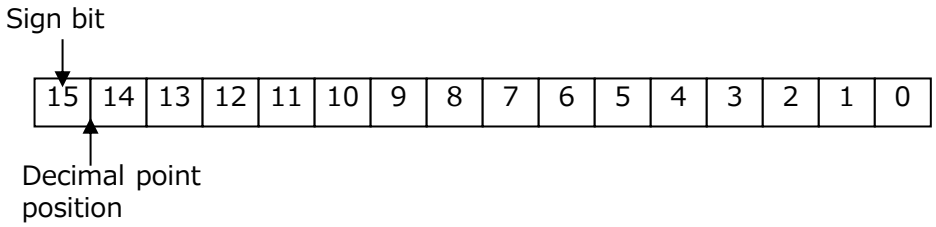


Fig. 9 Example of 16-bit data9

In this application design, the maximum value of the current data is defined as cA_Max. For example, when the 16-bit current data is 0x7fff, it is A_Max(A), and when it is 0x8000, it is-A_Max(A).

13.3 Data Format

The motor controller of this application performs fixed-point arithmetic.

In fixed-point arithmetic, the number of bits in the decimal part is expressed in Q notation (Q format).

Basically, operations are performed in Q15 format (fractional part 15 bits) for 16-bit data and Q31 format (fractional part 31 bits) for 32-bit data.

The values that can be represented in decimal format varies according to the format.

Table 2 Single-precision (16-bit) decimal format2

Q Format	Number of Fractional Bits	Maximum Positive Value (0x7FFF)	Maximum Negative 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 3 Double-precision (32-bit) Decimal Format3

Q Format	Number of Fractional Bits	Maximum Positive Value (0x7FFFFFFF)	Maximum Negative 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 Fixed-Point Arithmetic

In the four arithmetic operations of fixed-point arithmetic, addition and subtraction can be operated as if they were integers. However, in multiplication and division, the decimal point position of the operation result changes, so it is necessary to return to the original decimal point position.

(1)Multiplication

For multiplication between decimal formats, for example, when multiplying Q15 format data, the result is in double precision Q30 format. When Q31 format data is required, double precision Q31 format is achieved by shifting the operation result one bit to the left.

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

(2)Division

For division between decimal formats, for example, if you divide the Q31 format by the Q15 format, the result is in Q16 format. When Q15 format data is required, the Q15 format data can be obtained by right-shifting the divisor by 1 bit before the operation.

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

Terms of Use

This terms of use is made between Toshiba Electronic Devices and Storage Corporation (“We”) and Customer who downloads or uses this Reference Design. Customer shall comply with this terms of use. This Reference Design means all documents and data in order to design electronics applications on which our semiconductor device is embedded.

Section 1. Restrictions on usage

1. This Reference Design is provided solely as reference data for designing electronics applications. Customer shall not use this Reference Design for any other purpose, including without limitation, verification of reliability.
2. Customer shall not use this Reference Design for sale, lease or other transfer.
3. Customer shall not use this Reference Design for evaluation in high or low temperature, high humidity, or high electromagnetic environments.
4. This Reference Design shall not be used for or incorporated into any product or system whose manufacture, use, or sale is prohibited under any applicable laws or regulations.

Section 2. Limitations

1. We reserve the right to make changes to this Reference Design without notice.
2. This Reference Design should be treated as a reference only. WE ARE NOT RESPONSIBLE FOR ANY INCORRECT OR INCOMPLETE DATA AND INFORMATION.
3. Semiconductor devices can malfunction or fail. When designing electronics applications by referring to this Reference Design, Customer is 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 semiconductor devices could cause loss of human life, bodily injury or damage to property, including data loss or corruption. Customer must also refer to and comply with the latest versions of all relevant our information, including without limitation, specifications, data sheets and application notes for semiconductor devices, as well as the precautions and conditions set forth in the "Semiconductor Reliability Handbook".
4. Designing electronics applications by referring to this Reference Design, Customer must evaluate the whole system sufficiently. Customer is solely responsible for applying this Reference Design to Customer's own product design or applications. WE ASSUME NO LIABILITY FOR CUSTOMER'S PRODUCT DESIGN OR APPLICATIONS.
5. WE SHALL NOT BE RESPONSIBLE FOR ANY INFRINGEMENT OF PATENTS OR ANY OTHER INTELLECTUAL PROPERTY RIGHTS OF THIRD PARTIES THAT MAY RESULT FROM THE USE OF THIS REFERENCE DESIGN. NO LICENSE TO ANY INTELLECTUAL PROPERTY RIGHT IS GRANTED BY THIS TERMS OF USE, WHETHER EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE.
6. THIS REFERENCE DESIGN IS PROVIDED "AS IS". WE (a) ASSUME 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 (b) DISCLAIM ANY AND ALL EXPRESS OR IMPLIED WARRANTIES AND CONDITIONS RELATED TO THIS REFERENCE DESIGN, INCLUDING WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF FUNCTION AND WORKING, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OF INFORMATION, OR NONINFRINGEMENT.

Section 3. Terms and Termination

It is assumed that Customer agrees to any and all this terms of use if Customer downloads or uses this Reference Design. We may, at its sole and exclusive discretion, change, alter, modify, add, and/or remove any part of this terms of use at any time without any prior notice. We may terminate this terms of use at any time and without any cause. Upon termination of this terms of use, Customer shall eliminate this Reference Design. Furthermore, upon our request, Customer shall submit to us a written confirmation to prove elimination of this Reference Design.

Section 4. Export Control

Customer shall not use or otherwise make available this Reference Design 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). This Reference Design may be controlled under the applicable export laws and regulations including, without limitation, the Japanese Foreign Exchange and Foreign Trade Act and the U.S. Export Administration Regulations. Export and re-export of this Reference Design is strictly prohibited except in compliance with all applicable export laws and regulations.

Section 5. Governing Laws

This terms of use shall be governed and construed by laws of Japan, without reference to conflict of law principle.

Section 6. Jurisdiction

Unless otherwise specified, Tokyo District Court in Tokyo, Japan shall be exclusively the court of first jurisdiction for all disputes under this terms of use.