# Air Conditioner Outdoor Unit Circuit

# Software Guide

## RD219-SWGUIDE-01

TOSHIBA ELECTRONIC DEVICES & STORAGE CORPORATION

© 2024
Toshiba Electronic Devices Solutions Corporation

# Table of Contents



© 2024
Toshiba Electronic Devices Solutions Corporation

© 2024
Toshiba Electronic Devices Solutions Corporation

© 2024
Toshiba Electronic Devices Solutions Corporation

© 2024
Toshiba Electronic Devices Solutions Corporation

# 1. Overview

This document describes the software specifications of the Air Conditioner Outdoor Unit Circuit Reference Design (RD219). This reference design requires a host MCU board (MikroElektronika's Clicker 4 for TMPM4K) to control it.

## 1.1 Outline of Specifications

- Microcontroller(MCU) used: TMPM4KLFYAUG (operation clock of 160 MHz)
- Powered Devices used: TPD4204F (Fan Inveter), TK20A60W5 (Compressor Inverter), GT30J65MRB, and TRS24N65FB (PFC)
- Motors used: Brushless motor for fan, brushless motor for compressor
- Development Environment: EWARM Ver9.30.1 or KEIL Ver6.195.0.0

- Control Overview
  ・Brushless DC Motor Sensorless Vector Control
  ・Supports external speed control and PFC control (On/Off)
  ・DAC output (for variable-value analogue output) and LED output (LED for monitoring) for evalation

- Motor control details

| Item | Control Details |
|---|---|
| Current detection type | 1 shunt |
| Position detection type | Sensorless |
| PWM modulation type | 2-phase modulation |

- PFC control details

| Item | Control Details |
|---|---|
| PFC type | CCM |
| Number of coils | 1 |
| Current detection type | 1 shunt |

© 2024
Toshiba Electronic Devices Solutions Corporation

### 1.2 Outline of Motor Control Processing

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.

-

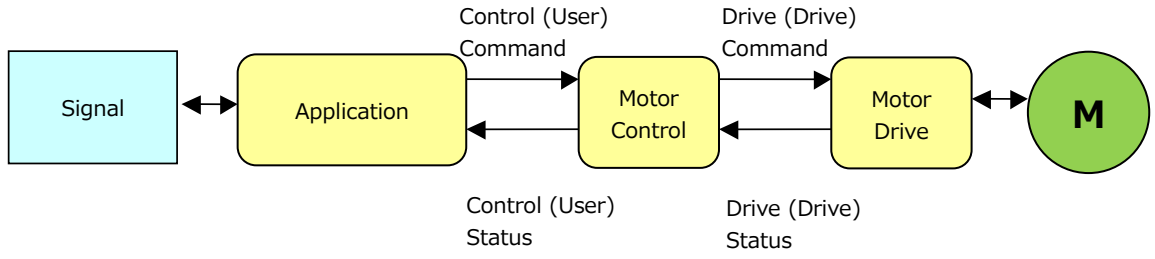



**Fig. 1 Structure of Vector Control Software**

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.

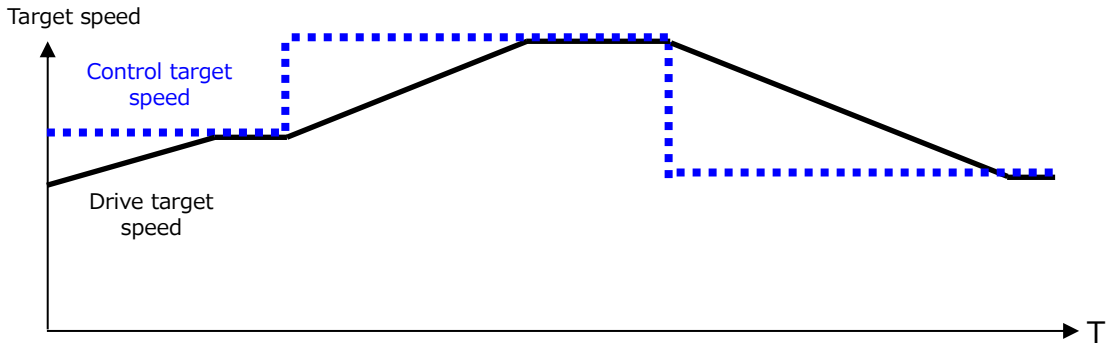



**Fig. 2 Control Target Speed and Drive Target Speed**

© 2024
Toshiba Electronic Devices Solutions Corporation

System Diagram

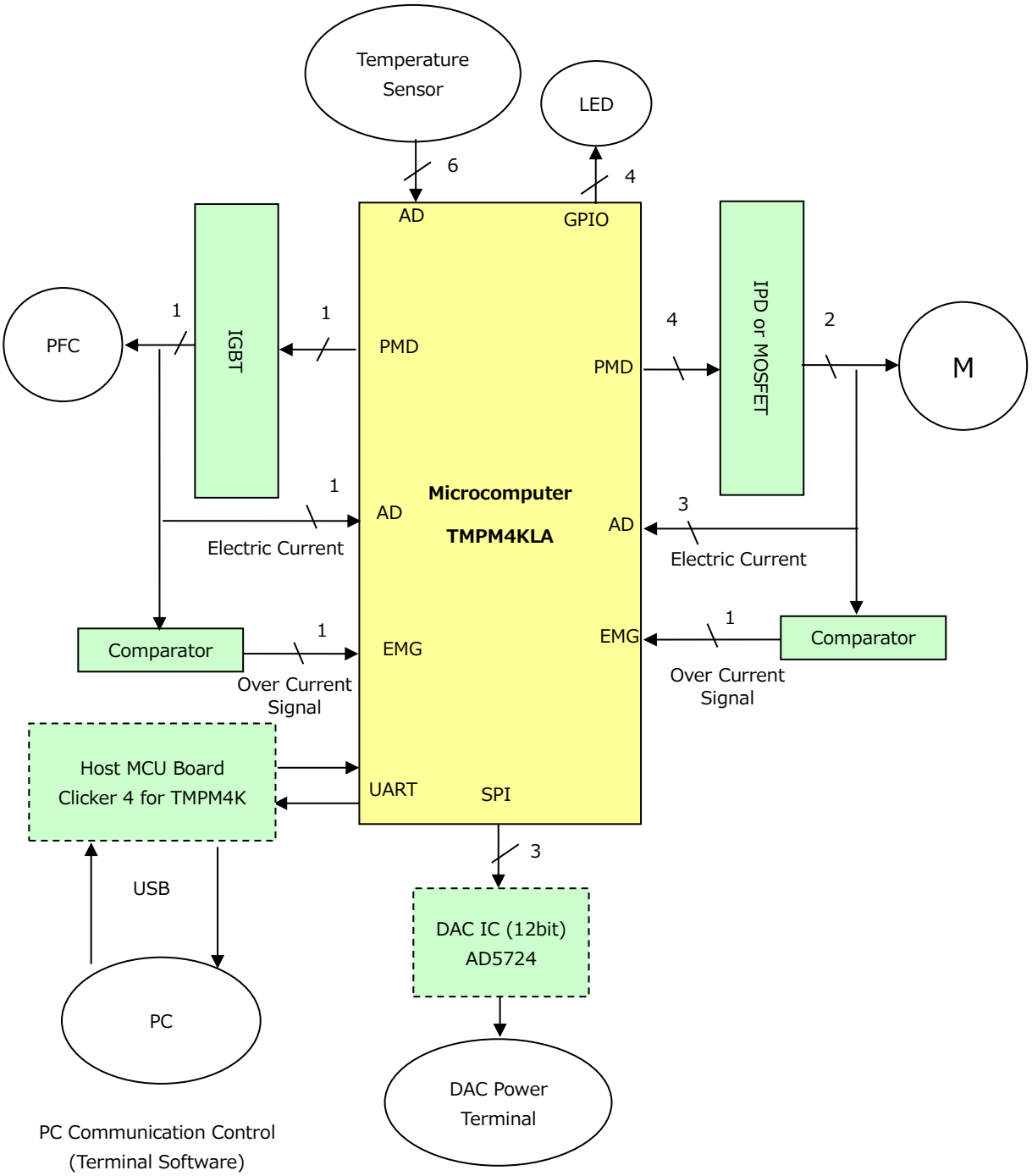



**Fig. 3 System block diagram**

© 2024
Toshiba Electronic Devices Solutions Corporation

## 2. Source File Configuration

```
Source
 | B_User.c                Vector control user settings source file
 | B_User.h                Vector control user settings header file
 | C_Control.c             Vector control source file
 | C_Control.h             Vector control header file
 | dac_drv.c               DAC IC driver source file
 | dac_drv.h               DAC IC driver header file
 | D_Driver.c              Vector control driver source file
 | D_Driver.h              Vector control driver header file
 | D_Para.h                Vector control parameter (channel common) header file
 | D_Para_Comp.h           Vector control parameter (for channel 0) header file
 | D_Para_Fan.h            Vector control parameter (for channel 1) header file
 | dac_drv.c               DAC source file
 | dac_drv.h               DAC header file
 | E_Sub.a                 Operation function library file
 | E_Sub.h                 Function library header file for arithmetic operation
 | HPFC_drv.c              PFC source file
 | HPFC_drv.h              PFC header file
 | HPFC_Para.h             PFC control parameter header file
 | initial.c               MCU initialization source file
 | initial.h               MCU initialization header file
 | interrupt.c             Interrupt control source file
 | interrupt.h             Interrupt control header file
 | ipdefine.h              MCU settings header file
 | main.c                  Main routine
 | mcuip_drv.c             MCU hardware setting driver source file
 | mcuip_drv.h             MCU hardware setting driver header file
 | sys_macro.h             Macro definition header file
 | system_int.c            Interrupt function source file
 | system_int.h            Interrupt function header file
 | usercon.c               User control source file
 | usercon.h               User control header file
 └──Libraries              Contain libraries for CMSIS core and individual IPs
    └──M4Kx
       ├──CMSIS                Contain files for CMSIS core
       |  | system_TMPM4KyA.c  MCU initialization source file
       |  | system_TMPM4KyA.h  MCU initialization header file
       |  | TMPM4KLA.h         MCU register definition header file
       |  | TMPM4KyA.h         MCU register common definition header file
       |  └──startup
       |     ├──arm
       |     |     startup_TMPM4KLA.s     Start-up assembler source (for arm)
       |     └──iar
       |           startup_TMPM4KLA.s     Start-up assembler source (for IAR)
       |
       └──IP_Driver        Contain peripheral drivers
            ipdrv_adc.c
            ipdrv_adc.h
            ipdrv_cg.c
            ipdrv_cg.h
            ipdrv_common.h
            ipdrv_siwdt.c
            ipdrv_siwdt.h
            ipdrv_t32a.c
            ipdrv_t32a.h
            ipdrv_tspi.c
            ipdrv_tspi.h
            ipdrv_uart.c
            ipdrv_uart.h
```

© 2024
Toshiba Electronic Devices Solutions Corporation

# 3. Evaluation Environment

## 3.1 Development Tools

This software is developed using the following development tools.

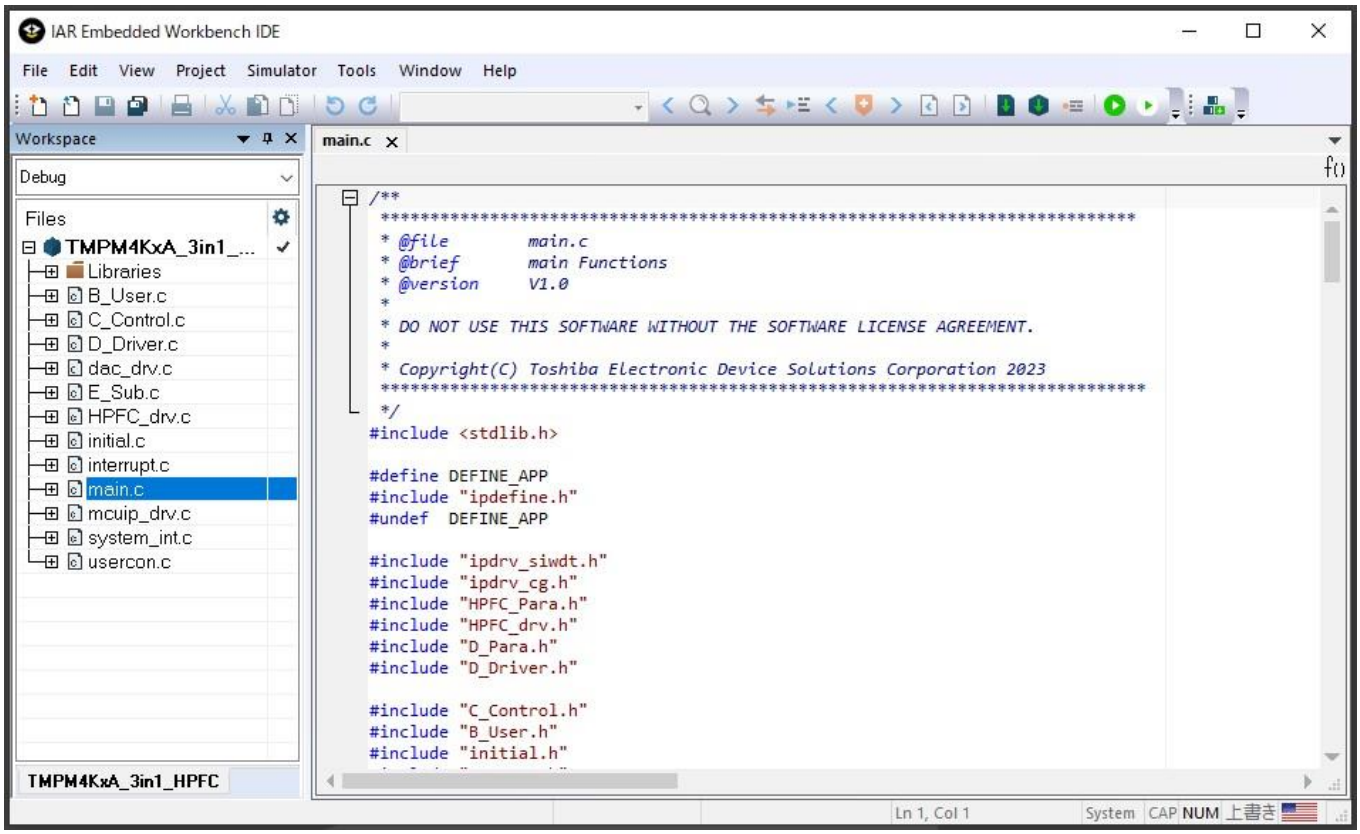
| | |
|---|---|
| IAR Embeded Workbench for ARM | 9.30.1 |
| KEIL µVision MDK-Lite | 6.195.0.0 |

## 3.2 How to Start Up the Project

The following is an example of IAR Embeded Workbench.

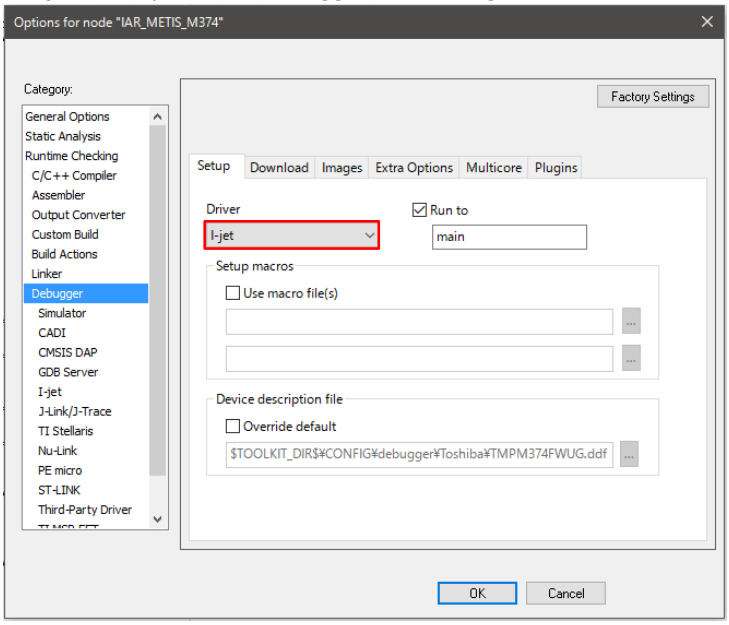
1. Double click iar¥TMPM4KxA_3in1_HPFC.eww, or [File] > [Open] > Open TMPM4KxA_3in1_HPFC.eww from [Workspace].

2. The following screen will be displayed.



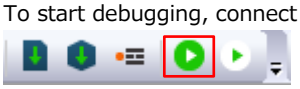

© 2024
Toshiba Electronic Devices Solutions Corporation

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




Select the tool you want to use.

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



### 3.3 DAC Power

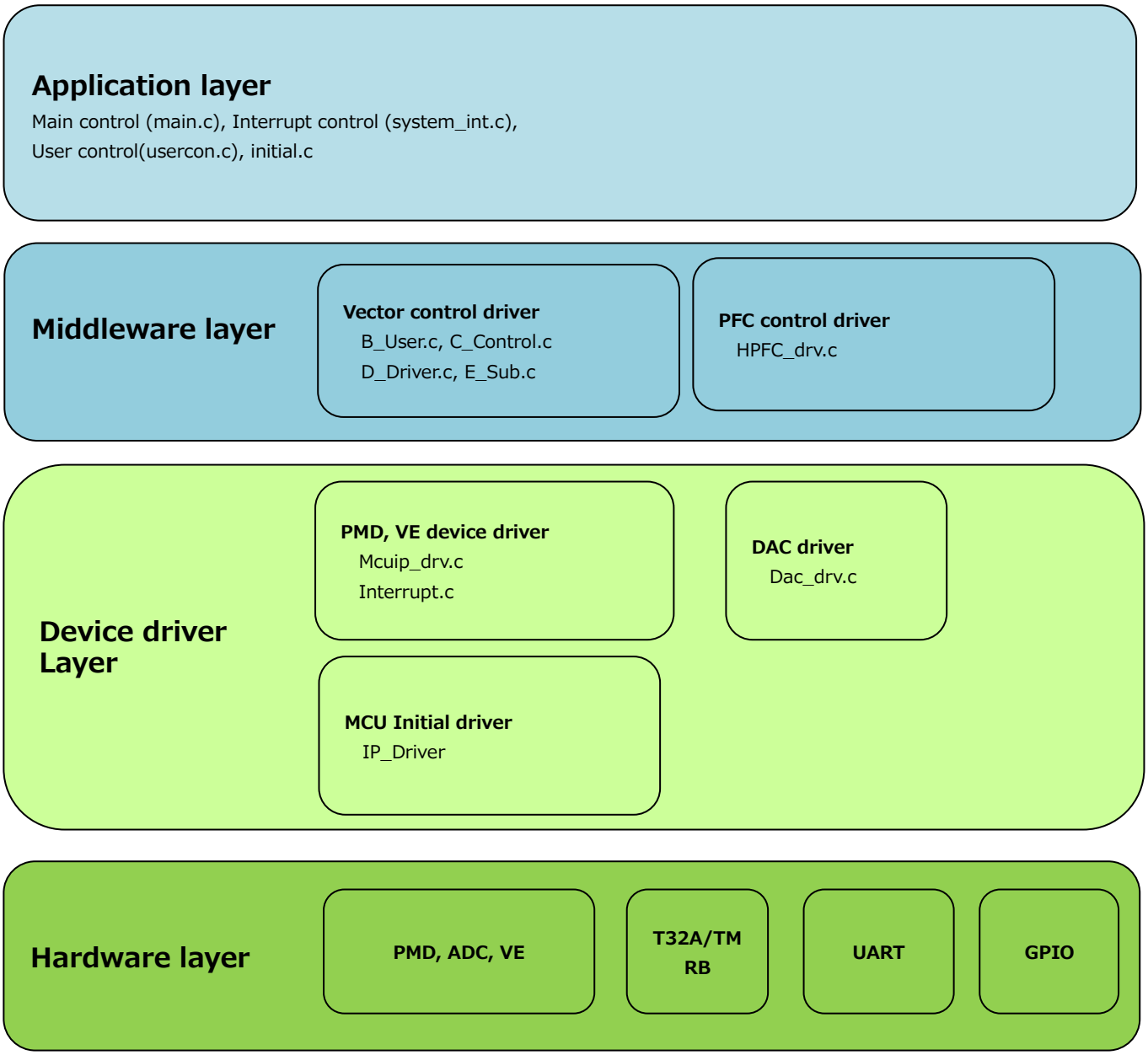
The DAC output function is implemented for viewing changes in variables on an oscilloscope, etc.
To enable DAC output, enable the following definition of D_Para.h.
#define　__USE_DAC

The variables to be DAC-printed are described in the function UiOutDataStart() 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

© 2024
Toshiba Electronic Devices Solutions Corporation

## 4. Module Configuration




© 2024
Toshiba Electronic Devices Solutions Corporation

# 5. Assigning MCU Hardware Resources

## 5.1 Peripheral Interface

| IP | | Details | Remark |
|---|---|---|---|
| T32A | CH0 | 4 kHz period interrupt | Timer A |
| | CH1 | Not used | |
| | CH2 | Not used | |
| | CH3 | Not used | |
| | CH4 | Not used | |
| | CH5 | Not used | |
| TSPI | CH0 | Not used | |
| | CH1 | DAC IC control | SIO |
| | CH2 | Not used | |
| | CH3 | Not used | |
| UART | CH0 | Main PCB communication | |
| | CH1 | Not used | |
| | CH2 | Not used | |
| | CH3 | Not used | |
| ADC | UNITA | Motor CH0 coil current, power supply voltage | |
| | UNITB | Motor CH1 coil current | |
| | UNITC | Acquire PFC current and AC voltage. | |
| A-PMD | CH0 | Motor CH0 control | |
| | CH2 | Motor CH1 control | |
| | CH3 | PFC control | |
| A-VE+ | CH0 | Motor CH0 control | |

## 5.2 Interrupt

| Factor Name | Processing Details | Function Name |
|---|---|---|
| INTT32A0AC_IRQn | 4 kHz period timing creation | INTT32A0AC_IRQHandler |
| INTVCN0_IRQn | ・Vector control software for FAN<br>(for vector control by VE)<br>・PFC voltage control | INTVCN0_IRQHandler |
| INTADBPDB_IRQn | Vector control software for COMP<br>(for vector control by software) | INTADBPDB_IRQHandler |
| INTADCPDB_IRQn | PFC current control process | INTADCPDB_IRQHandler |
| INTSC1TX_IRQn | DAC IC control | INTSC1TX_IRQHandler |
| INTSC0RX_IRQn | Main board reception | INTSC0RX_IRQHandler |
| INTSC0TX_IRQn | Main board transmission | INTSC0TX_IRQHandler |

Interrupt priorities can be changed with ipdefine.h constants below.

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

#define INT4KH_LEVEL        5   /* 4kH interval timer interrupt */
#define INT_VC_LEVEL        4   /* VE interrupt */
#define INT_ADC_LEVEL       5   /* ADC interrupt */
#define INT_DAC_LEVEL       6   /* SIO interrupt for Dac */
#define INT_UART_LEVEL      7   /* UART interrupt */
```

© 2024
Toshiba Electronic Devices Solutions Corporation

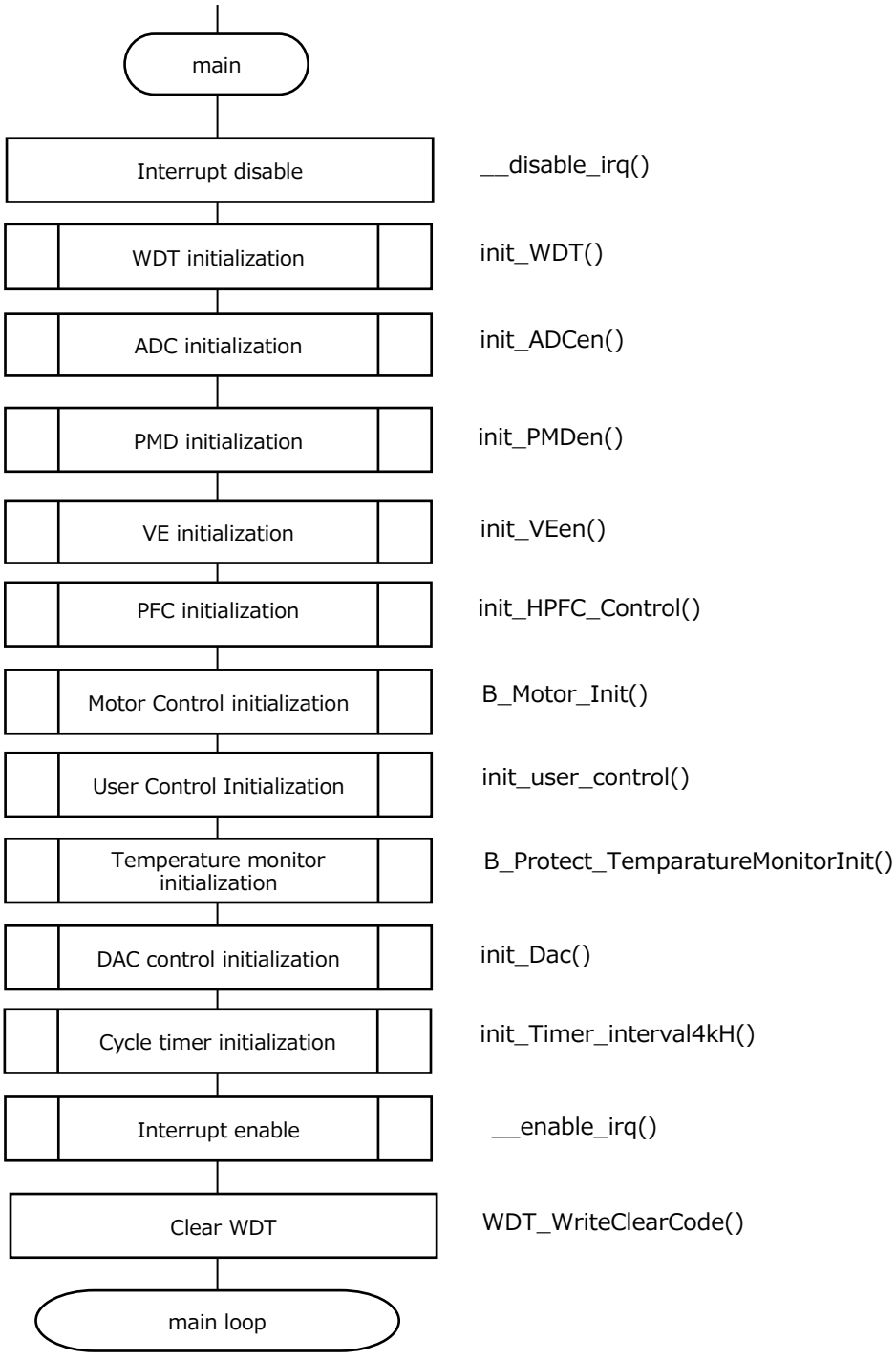
### 5.3 GPIO

| PORT | Function |
|---|---|
| PF0 | SWD-SWDIO |
| PU0 | Reserved (UART-RXD2 for indoor unit communication) |
| PU1 | Reserved (UART-TXD2 for indoor unit communication) |
| PU2 | Reserved (GPIO for controlling expansion valve A) |
| PU3 | Reserved (GPIO for controlling expansion valve B) |
| PU4 | PWM output for PFC |
| PU5 | GPIO for LED4 |
| PU6 | PFC EMG In |
| PA2 | GPIO for LED3/PMD2DBG |
| PA3 | Reserved (GPIO for controlling expansion valve C) |
| PA4 | Reserved (GPIO for controlling expansion valve D) |
| PL0 | CH0(FAN) Motor-coil current detection (OP-AMP +) |
| PL1 | CH0(FAN) Motor-coil current detection (OP-AMP -) |
| PL2 | CH1 (Compressor) Motor-coil current detection (OP-AMP +) |
| PL3 | CH0 (Compressor) Motor-coil current detection (OP-AMP -) |
| PL4 | IGBT temperature detection |
| PL5 | Diode temperature detection |
| PL6 | DC link voltage detection |
| PL7 | Reserved (GPIO for PTC control) |
| PK2 | Out pipe temperature sample |
| PK1 | Reservation (Heat exhaust pipe temperature detection) |
| PK0 | Reservation (Outdoor temperature detection) |
| PJ2 | HVMOS temperature detection |
| PJ1 | Coil current detection for PFC |
| PJ0 | AC voltage detection for PFC |
| PC0 | UART-TXD0 for main-board communication |
| PC1 | UART-RXD0 for main-board communication |
| PC2 | GPIO for LED1/PMD0DBG |
| PC3 | GPIO for LED2/PMD1DBG |
| PB0 | CH0(FAN) Motor U-phase PWM output |
| PB1 | CH0(FAN) Motor X phase PWM output |
| PB2 | CH0(FAN) Motor V-phase PWM output |
| PB3 | CH0(FAN) Motor Y-phase PWM output |
| PB4 | CH0(FAN) Motor W-phase PWM output |
| PB5 | CH0(FAN) Motor Z-phase PWM output |
| PB6 | CH0(FAN) Motor EMG input |
| PG2 | Fixed to Low (single-chip mode) |
| PG3 | Reserved (GPIO for 4-way valve relay control) |
| PG4 | SYNC for DAC communication |
| PG5 | SDIN for DAC communication |
| PG6 | SCLK for DAC communication |
| PE0 | CH0 (Compressor) motor U-phase PWM output |
| PE1 | CH0 (Compressor) Motor X-phase PWM output |
| PE2 | CH0 (Compressor) Motor V-phase PWM output |
| PE3 | CH0 (Compressor) Motor Y-phase PWM output |
| PE4 | CH0 (Compressor) Motor W-phase PWM output |
| PE5 | CH0 (Compressor) Motor Z-phase PWM output |
| PE6 | CH0 (compressor) Motor EMG input |
| PH0 | Reserved (10 MHz Crystal Oscillator) |
| PH1 | Reserved (10 MHz Crystal Oscillator) |
| PF1 | SWD-SWCLK |

© 2024
Toshiba Electronic Devices Solutions Corporation

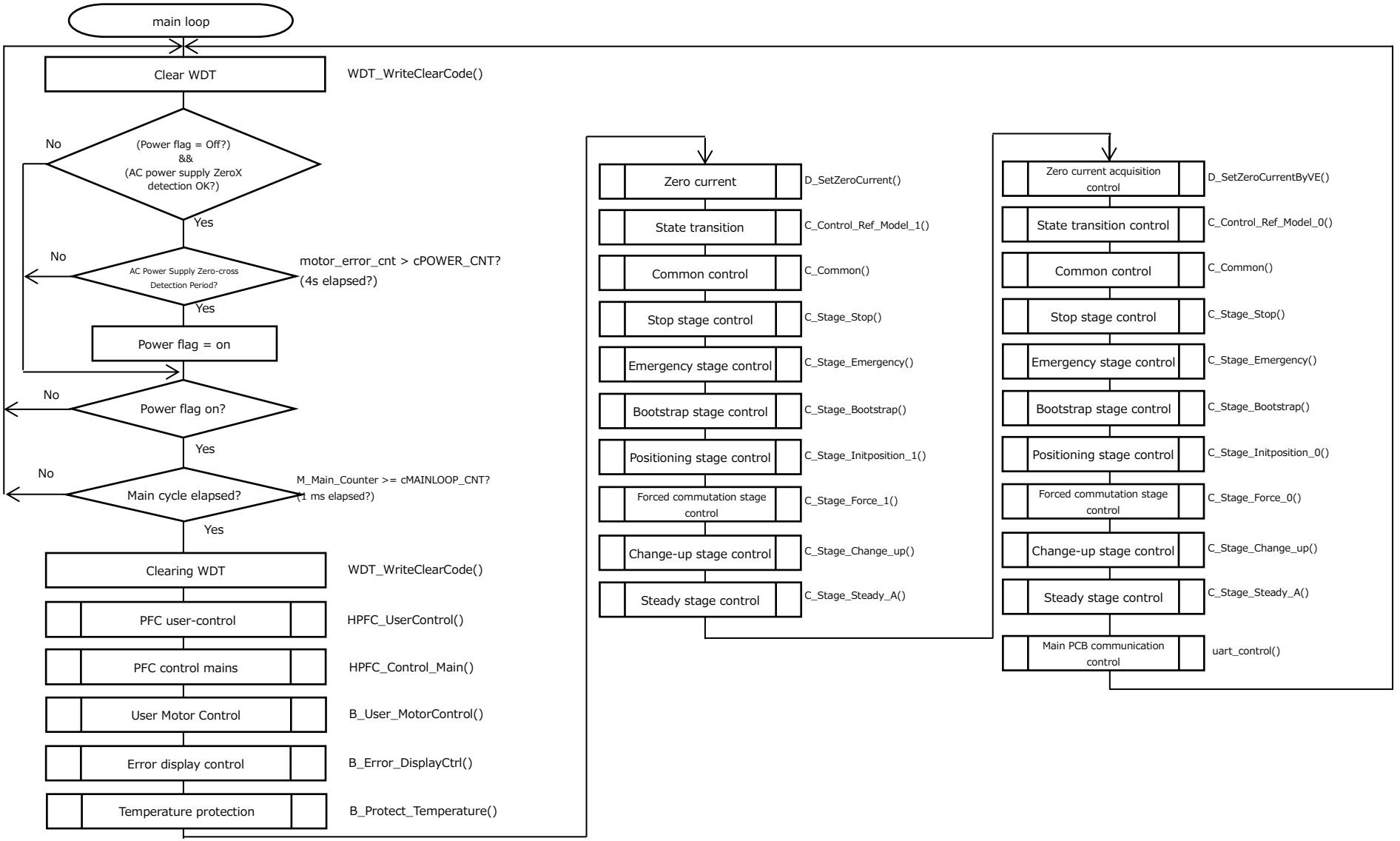
# 6. General Flow

## 6.1 Main Routine (main)




© 2024
Toshiba Electronic Devices Solutions Corporation

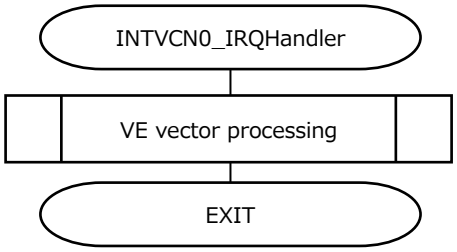
## 6.2 Main Loop (main_loop)




© 2023
Toshiba Electronic Devices Solutions Corporation

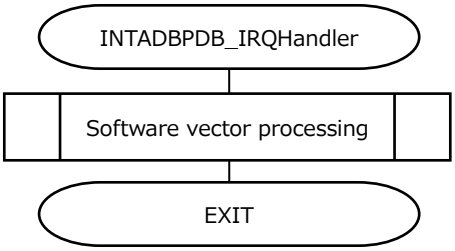
### 6.3 Interrupt

At the completion of the VE schedule



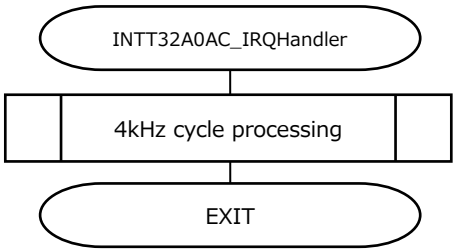

INT_VectorContorlByVE()

AD conversion end interrupt



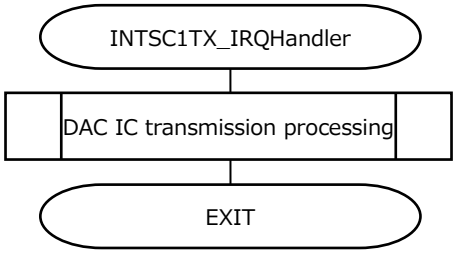

INT_VectorContorlBySoft()

At the every 4 kHz period




INT_interval_4kHz ()

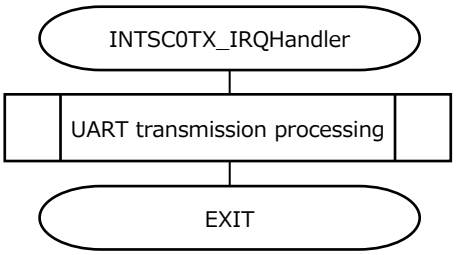
At the DAC (I/O) communication 24 bit transmission completion




DacWriteContinue ()

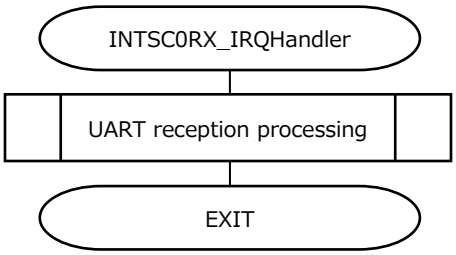
© 2024
Toshiba Electronic Devices & Storage Corporation

At the main board communication transmission completion




INT_uart0tx ()

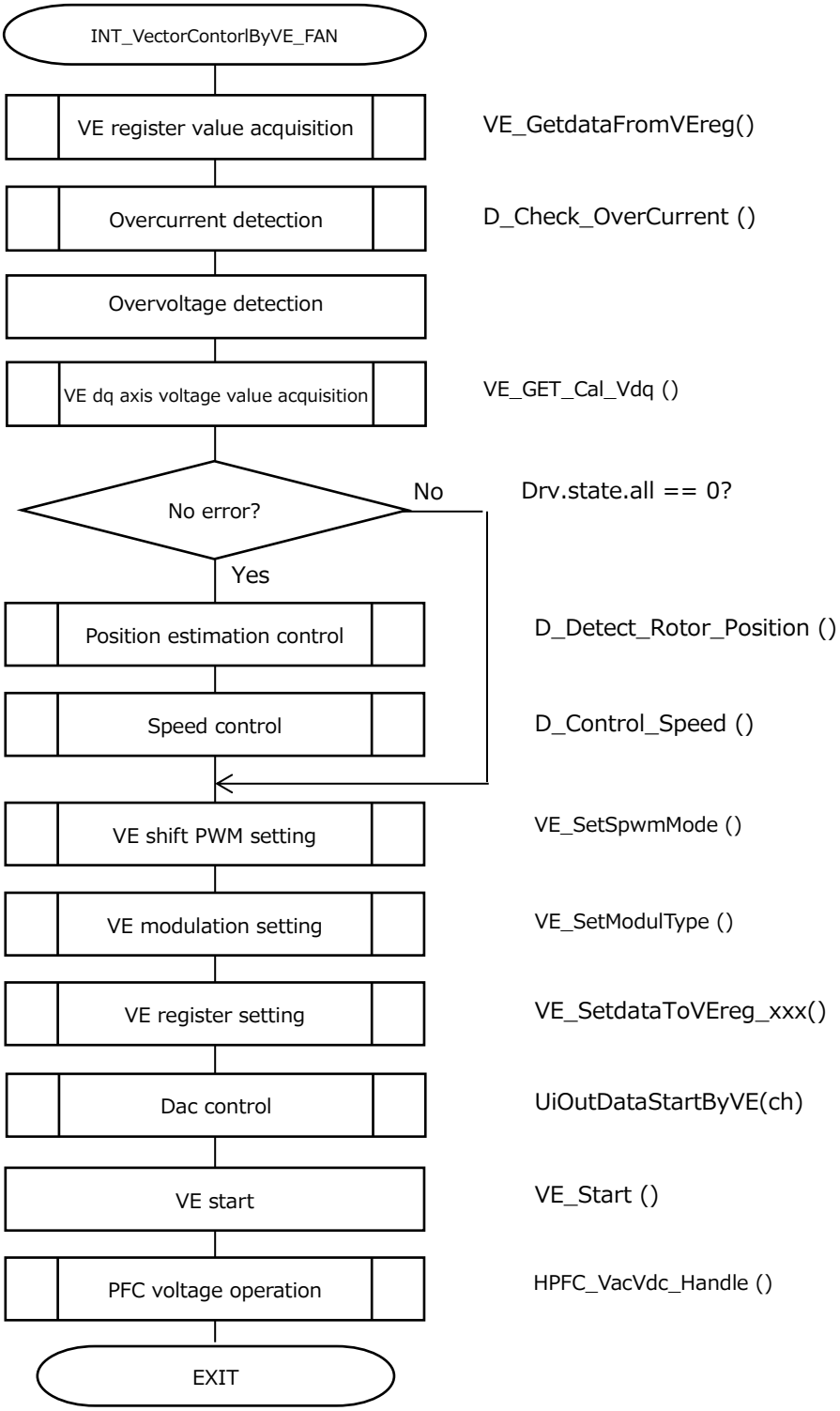
At the main board communication reception completion




INT_uart0rx ()

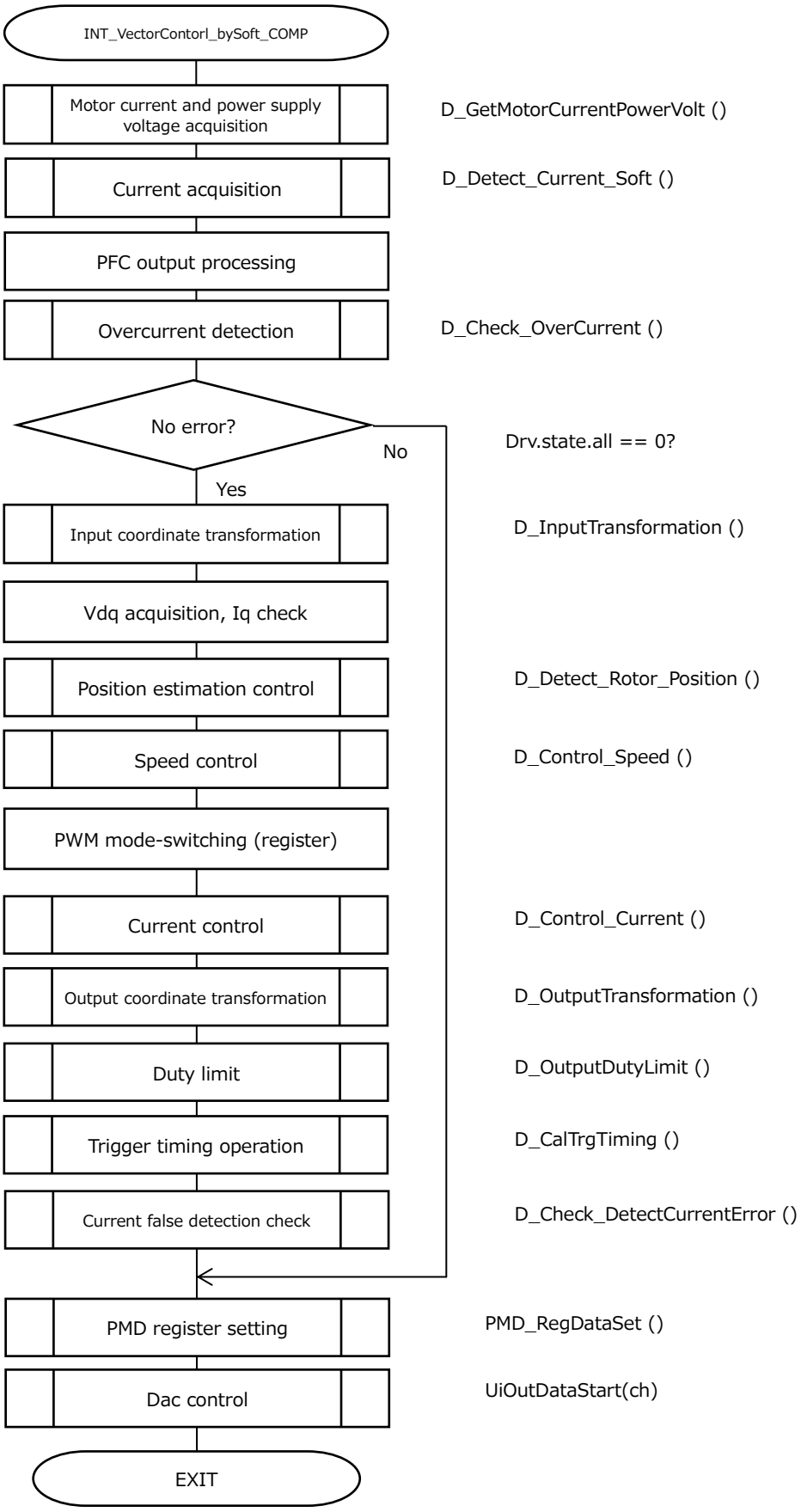
© 2024
Toshiba Electronic Devices & Storage Corporation

### 6.3.1 VE Vector Processing (INT_VectorControlByVE_FAN)




© 2024
Toshiba Electronic Devices & Storage Corporation

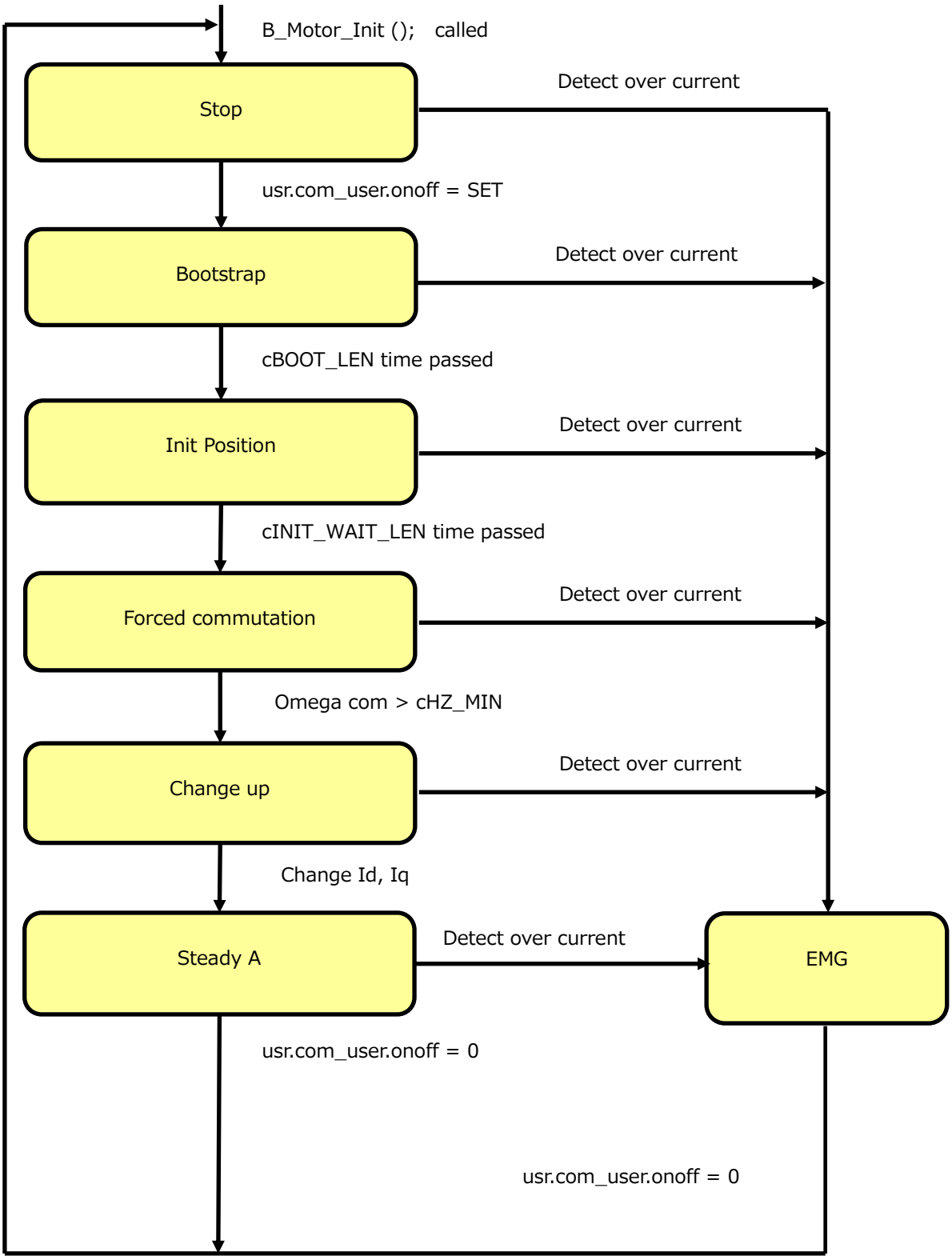
### 6.3.2 Software Vector Processing (INT_VectorControlBySoft_COMP)




© 2024
Toshiba Electronic Devices & Storage Corporation

# 7. State Transition

## 7.1 Sensorless




© 2024
Toshiba Electronic Devices & Storage Corporation

# 8. Functional Description

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

## 8.1 Control Command

Control commands are described below.

### 8.1.1 Control Method (usr.com_user)

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

```
typedef struct {
    uint16_t reserved:12; /* reserved */
    uint16_t spwm:2;   /* Shift PWM      0=off, 1=on */
    uint16_t modul:1;  /* PWM Moduration 0=3phase modulation, 1=2phase modulation */
    uint16_t onoff:1;    /* Motor start command 0=off, 1=on*/
} command_t;
```

command_t　Com_user;
In applications, the usr.com_user is set as a control command.

### 8.1.2 Control Target Speed

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

The application sets the usr.omega_user as the target rate of control.

### 8.1.3 Striking Current

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

In the application, the usr.Id_st_user and usr.Iq_st_user are set as start-up current commands.

© 2024
Toshiba Electronic Devices & Storage Corporation

## 8.2 Drive Command

The driving commands are described below.

### 8.2.1 Driving (drv.command)

Command_t Command;

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

### 8.2.2 Vector Control Command (drv.vector_cmd)

The command is a vector control operation that manages the 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 output0=off 1=on */
} vectorcmd_t;
```

In each stage, a vector control drive command (drv.vector_cmd) is set as shown below and the command is sent to the motor drive.

Note that F_vcomm_volt and F_vcomm_Idetect are not used.

| Cmd / Stage | Theta | Omega | Current | Volt | Edetect | Idetect | Onoff |
|---|---|---|---|---|---|---|---|
| Stop | CLEAR | CLEAR | CLEAR | CLEAR | CLEAR | CLEAR | CLEAR |
| Bootstrap | CLEAR | CLEAR | CLEAR | CLEAR | CLEAR | CLEAR | SET |
| InitPosition | CLEAR | CLEAR | CLEAR | SET | CLEAR | SET | SET |
| Forced | SET | CLEAR | CLEAR | SET | SET | SET | SET |
| Change_up | SET | SET | CLEAR | SET | SET | SET | SET |
| Steady | SET | SET | SET | SET | SET | SET | SET |
| Emergency | CLEAR | CLEAR | CLEAR | CLEAR | CLEAR | CLEAR | CLEAR |

1) F_vcomm_theta

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.

5) F_vcomm_Idetect

When it is SET, input coordinate axis transformation calculation is performed. And when it is CLEAR, the calculation is not performed, and the value is cleared.

© 2024
Toshiba Electronic Devices & Storage Corporation

6) F_vcomm_onoff

When it is SET, motor-output wave form is ON. And when it is CLEAR, the motor-output wave form is OFF.

## 8.3 Driving State

### 8.3.1 Error Condition (drv.state)

```
typedef union {
  struct {
    uint16_t reserve:11;   /* reserve */
    uint16_t Loss_sync: 1;/* 0:normal, 1: Loss of synchronism */
    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)   */
    uint16_t pfc_relate:1; /* 0:normal, 1: PFC error   */
} 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 | Abnormal current detection | It is SET when abnormal current is detected. (unused) |
| emg_S | Soft overcurrent detection | It is SET when overcurrent is detected by software processing. |
| emg_H | Hard overcurrent detection | It is SET when overcurrent is detected by the MCU hardware function. |
| pfc_relate | PFC abnormality detection | It is SET when abnormality in PFC is detected. |

© 2024
Toshiba Electronic Devices & Storage Corporation

## 8.4 Motor Control Structure

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

Example)

```
vector_t    Motor_fan;    /* Motor data for FAN motor */
vector_t    Motor_comp;   /* Motor data for COMPRESSOR */
```

### 8.4.1 List of Variables

| Type | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Main_stage_e | Stage.main | Main stage | --- | cStop<br>cBootstrap<br>cInitposition<br>cForce<br>cChange_up<br>cSteady_A<br>cEmergency |
| Sub_stage_e | Stage.sub | Sub stage | --- | cStep0<br>cStep1<br>cStep2<br>cStep3<br>cStepEnd |
| Itr_stage_e | Stage.itr | Interrupt stage | --- | CiStop<br>CiBootstrap<br>CiInitposition_i<br>CiInitposition_v<br>CiForce_i<br>CiForce_v<br>CiChange_up<br>CiSteady_A<br>CiEmergency |
| Uint8_t | Error_flag[2] | Error information | --- | |
| q15_t | Target_spd | Target speed | Q15 | |
| 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 | |
| q15_t | Drv.Vdc_temp | Previous power supply voltage | Q15 | |
| q31_u | Drv.Vdc_ave | (unused) | --- | |
| q31_u | Drv.Vd | d-axis voltage | Q31 | |
| q31_u | Drv.Vq | q-axis voltage | Q31 | |
| q15_t | Drv.Vdq | (unused) | --- | |
| q31_u | Drv.Vdq_ave | (unused) | --- | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| q31_t | Drv.Vd_out | Output voltages | Q31 | |
| q15_t | Drv.Ed | d-axis induced voltage | Q15 | |
| q15_t | Drv.Eq | (unused) | --- | |
| q31_t | Drv.Ed_I | d-axis induced voltage integral value | Q31 | |
| q31_t | Drv.Ed_PI | d-axis induced-voltage PI | Q31 | |
| State_t | Drv.state | Motor failure status | --- | |
| Command_t | Drv.command | Drive method | --- | (Refer to 8.2.1) |
| vectorcmd_t | Drv.vector_cmd | Vector control commands | --- | (Refer to 8.2.2) |
| Uint16_t | Drv.chkpls | Current detection protection duty range | Q0 | |
| Uint8_t | Drv.idetect_error | Current detection status | --- | 0:Detectable<br>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-phase 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 | Q15 | |
| q15_t | Usr.Iq_st_user | Starting Iq current | Q15 | |
| Uint16_t | Usr.lambda_user | Initial rotor position | Q0 | |
| Command_t | Usr.com_user | Control method | --- | (Refer to 8.1.1) |
| Command_t | Usr.com_user_1 | Control method last time | --- | |
| Uint16_t | Para.V_max | Maximum voltage | --- | |
| Uint16_t | Para.I_max | Maximum current | --- | |
| Uint16_t | Para.Hz_max | Maximum speed | --- | |
| q15_t | Para.pos_degadj | Initial position | --- | |
| 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 | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| 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 | 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 cycle | 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 | (unused) | 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 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 | --- | c1shunt |
| Boot_type_e | Boot_type | Startup type | --- | cBoot_v |

For vector control by software

| Type | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Uint32_t* | Drv.ADxREG0 | ADC Conversion Result Register Address | --- | |
| Uint32_t* | Drv.ADxREG1 | ADC Conversion Result Register Address | --- | |
| Uint32_t* | Drv.ADxREG2 | ADC Conversion Result Register Address | --- | |
| Uint32_t* | Drv.ADxREG3 | ADC Conversion Result Register Address | --- | |
| q15_t | Drv.Vdc_adc | Power supply voltage conversion result | Q15 | |
| Uint16_t | Drv.Ia_adc | U-phase current conversion result | Q15 | |
| Uint16_t | Drv.Ib_adc | V-phase current conversion result | Q15 | |
| Uint16_t | Drv.Ic_adc | W-phase current conversion result | Q15 | |
| q15_t | Drv.Idc1_adc | 1st phase current conversion | Q15 | |
| q15_t | Drv.Idc2_adc | 2nd phase current conversion | Q15 | |
| q31_u | Drv.Idco_ave | Zero-current average ADC value | Q0 | |
| q15_t | Drv.Ialpha | α-axis current | Q15 | |
| q15_t | Drv.Ibeta | β-axis current | Q15 | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| q15_t | Drv.Id_dev | d-axis current deviation | Q15 | |
| q15_t | Drv.Iq_dev | q-axis current deviation | Q15 | |
| q31_u | Drv.Vd_com | d-axis voltage command value | Q31 | |
| q31_u | Drv.Vq_com | q-axis voltage command value | Q31 | |
| q31_u | Drv.Vd_I | d-axis voltage integral value | Q31 | |
| q31_u | Drv.Vq_I | q-axis voltage integral value | Q31 | |
| q31_u | Drv.Valpha | α-axis voltage | Q31 | |
| q31_u | Drv.Vbeta | β-axis voltage | Q31 | |
| q15_t | Drv.DutyU | U-phase PWM Duty rate | Q15 | |
| q15_t | Drv.DutyV | V-phase PWM Duty rate | Q15 | |
| q15_t | Drv.DutyW | W-phase PWM Duty rate | Q15 | |
| Int32_t | Drv.AdTrg0 | ADC trigger timing 0 position | Q15 | |
| Int32_t | Drv.AdTrg1 | ADC trigger timing 1 position | Q15 | |
| Uint8_t | Drv.Sector0 | Sector | Q0 | 0 to 5 |
| Uint8_t | Drv.Sector1 | Previous sector | Q0 | 0 to 5 |

© 2024
Toshiba Electronic Devices & Storage Corporation

## 8.5 Function Details

### 8.5.1 ADC Initialization (init_ADCen)

8.5.1.1 **Syntax**

void init_ADCen(void)
Arguments:
None
Return value:
None

8.5.1.2 **Processing Details**

Performs the initialization for ADC.
・ADC setting for motors
・ADC enable

### 8.5.2 PMD Initialization (init_PMDen)

8.5.2.1 **Syntax**

void init_PMDen(void)
Arguments:
None
Return value:
None

8.5.2.2 **Processing Details**

Performs the initialization for PMD (Programmable Motor Driver).

### 8.5.3 VE Initialization (init_VEen)

8.5.3.1 **Syntax**

void init_VEen(void)
Arguments:
None
Return value:
None

8.5.3.2 **Processing Details**

Performs the initialization for VE (vector engine).
・VE interrupt setting
・VE setting

### 8.5.4 Motor Control Initialization (B_Motor_Init)

8.5.4.1 **Syntax**

void B_Motor_Init(void)
Arguments:
None
Return value:
None

8.5.4.2 **Variable**

© 2024
Toshiba Electronic Devices & Storage Corporation

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Output | Shunt_type | Shunt type | --- | |
| | Boot_type | Drive type | --- | |
| | Usr.com_user.encoder | Encoder control | --- | |
| | 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 | |
| | Drv.Idco_ave | Zero current conversion result | Q15 | 1 shunt only |
| | Usr.Iq_st_user | Starting Iq current | Q15 | |
| | Usr.Id_st_user | Starting Id current | Q15 | |
| | Usr.lambda_user | Initial rotor position | Q0 | |
| | Para.motor.r | Motor winding resistance | Q15 | |
| | Para.motor.Lq | Motor q-axis inductance | Q15 | |
| | Para.motor.Ld | Motor d-axis inductance | Q15 | |
| | Para.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 | Wait time after change-up | Q0 | |
| | Para.omega_min | Forced commutation termination rate | Q15 | |
| | Para.omega_v2i | Current control switching speed | Q15 | |
| | Para.delta_lambda | Current switching phase at change-up | Q0 | |
| | Para.pos.ki | Position estimation integral gain | Q15 | Q12 available |
| | Para.pos.kp | Position estimation proportional gain | Q15 | Q12 available |
| | Para.pos.ctrlprd | Position estimation control cycle | Q16 | |
| | Para.spd.ki | Speed control integral gain | Q15 | Q12 available |
| | Para.spd.kp | Speed control proportional gain | Q15 | Q12 available |
| | Para.current.dki | d-axis current proportional gain | Q15 | Q12 available |
| | Para.current.dkp | d-axis current integral gain | Q15 | Q12 available |
| | Para.current.qki | q-axis current proportional gain | Q15 | Q12 available |
| | Para.current.qkp | q-axis current integral gain | Q15 | Q12 available |
| | Para.iq_lim | q-axis current limit value | Q31 | |
| | Para.id_lim | d-axis current limit value | Q31 | |
| | Para.err_ovc | Overcurrent setting | Q15 | |
| | Usr.com_user.modul | Modulation type control command | --- | 2-phase modulation |
| | Para.TrgPosMd | Current detection position mode | --- | |
| | Para.TrgComp | Trigger timing correction value | Q15 | |

8.5.4.3 **Processing Details**

Initializes the motor control argument.

Sets a variable that does not change the setting during control.

© 2024
Toshiba Electronic Devices & Storage Corporation

### 8.5.5 DAC Control Initialization (init_Dac)

8.5.5.1 **Syntax**

void init_Dac(TSB_SC_TypeDef* const SCx)
Arguments:
Select TSB_SC_TypeDef * const SCx:SC.
Return value:
None

8.5.5.2 **Processing Details**

Initializes DAC IC control.
Enables SIO
Initializes Port
Initializes SIO
Initializes DAC IC communication
Sets SIO interrupt level
Clears pending SIO interrupt
Transmits Interrupt enable

### 8.5.6 Cycle Timer Initialization (init_Timer_interval4kHz)

8.5.6.1 **Syntax**

void init_Timer_interval4kHz(void)
Arguments:
None
Return value:
None

8.5.6.2 **Processing Details**

Initializes the timer for creating 4 kHz cycle.

### 8.5.7 PFC Control Initialization (init_HPFC_Control)

8.5.7.1 **Syntax**

void init_HPFC_Control(void)
Arguments:
None
Return value:
None

8.5.7.2 **Processing Details**

Performs the initialization to control PFC.
PWM, ADC, etc.

### 8.5.8 UART Initialization (init_uart)

8.5.8.1 **Syntax**

void init_uart(void)
Arguments:
None
Return value:
None

© 2024
Toshiba Electronic Devices & Storage Corporation

8.5.8.2 **Processing Details**

Performs the initialization for UART.

### 8.5.9 User UART Control (uart_control)

8.5.9.1 **Syntax**

void user_control(void)
Arguments:
None
Return value:
None

8.5.9.2 **Processing Details**

Controls commands from external sources such as users.

・Increase/decrease of revolutions
・Control motor switching (fan/compressor)
・Stopping fan motors, compressor motors and PFC
・Switching of info-acquisition (fan motor/compressor motor /PFC)
・Switching DAC modes

### 8.5.10 User Motor Control (B_User_MotorControl)

8.5.10.1 **Syntax**

void B_User_MotorControl(void)
Arguments:
None
Return value:
None

8.5.10.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Target_spd | Target speed | --- | User command |
| | Keydata.sw | Switching ON, OFF status | --- | User command |
| Output | Usr.omega_user | Control target speed | Q31 | |
| | Usr.com_user.onoff | Motor ON/OFF | --- | |
| | Drv.state | Motor failure status | --- | |

8.5.10.3 **Processing Details**

Processes commands from the user related to the motor.

・Checking the overcurrent (hardware) status
Checks the hardware overcurrent status and update the motor fault status.
・Motor ON/OFF command
Determines the motor ON/OFF (usr.com_user.onoff) from the status of the target speed (target_spd1).
Clears the motor error status, when the motor OFF command is issued.
・Drive speed normalization, etc.
Normalizes the target speed.

© 2024
Toshiba Electronic Devices & Storage Corporation

## 8.6 Motor Control Functions

Various states of motor control are defined as stages in this application.
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 stage, bootstrap stage, positioning stage, forced commutation stage, change-up stage, steady stage, 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), change-up (Change_up), and steady (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.

### 8.6.1 State Transition Processing Function (C_Control_Ref_Model_0, C_Control_Ref_Model_1)

#### 8.6.1.1 **Syntax**

void C_Control_Ref_Model(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

#### 8.6.1.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| 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 | Previous control command | --- | |

#### 8.6.1.3 **Processing Details**

System 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-stage is not executed by the state transition processing function, but by the processing function of each state.
Various states of motor control are defined as stages in this application.

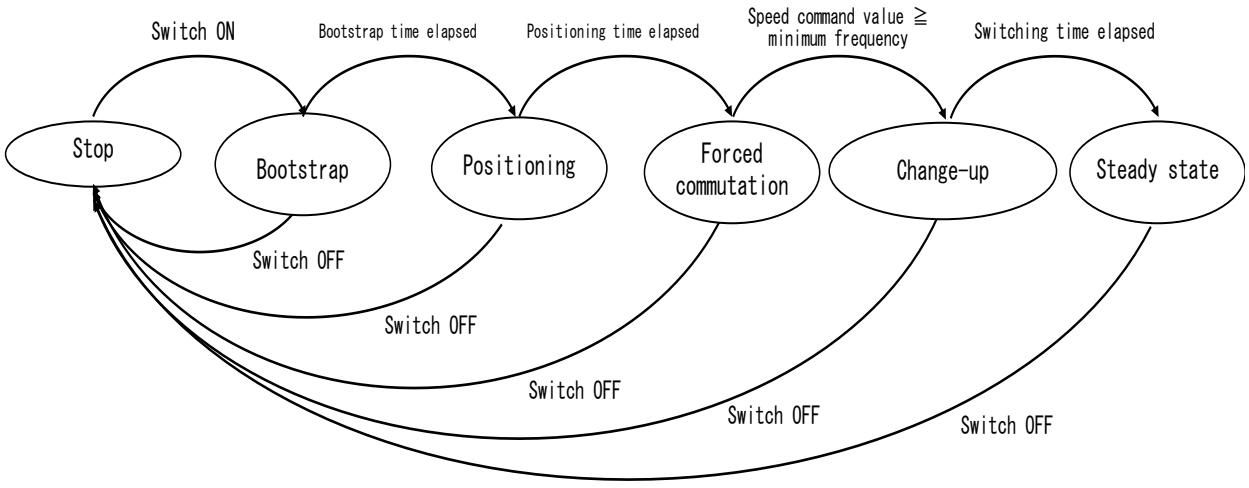



**Fig. 4 State Transition Diagram of Motor Control Stages**

© 2024
Toshiba Electronic Devices & Storage Corporation

### 8.6.2 Motor Control Common Processing Function (C_Common)

8.6.2.1 **Syntax**

void C_Common(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

8.6.2.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Usr.com_user.modul | Modulation type control command | --- | 2-phase or 3 phase modulation |
| | Para.chkpls | Current detection protection duty range | --- | |
| Output | Drv.command.modul | Modulation type drive command | --- | 2-phase or 3 phase modulation |
| | Drv.chkpls | Current detection protection duty range | --- | |

8.6.2.3 **Processing Details**

Executes the process common to each stage of motor control.
Calculate the shifted PWM control (8.6.10 C_ShiftPWM_Control).

### 8.6.3 Stop Stage Function (C_Stage_Stop)

8.6.3.1 **Syntax**

void C_Stage_Stop(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

8.6.3.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Stage.main | Main stage | --- | |
| | Motor_type | Drive motor | --- | |
| Input/Output | Stage.sub | Sub-stage | --- | |
| Output | Stage.itr | Interrupt stage | --- | |
| | Drv.vector_cmd | Vector control commands | --- | (Refer to 8.2.2.) |
| | 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.idetect_error | Current detection status | --- | |
| | Drv.Vd_com | d-axis voltage command value | Q31 | Software vector control only |
| | Drv.Vq_com | q-axis voltage command value | Q31 | Software vector control only |
| | Drv.Id_com | d-axis current command value | Q31 | |
| | Drv.Iq_com | q-axis current command value | Q31 | |

8.6.3.3 **Processing Details**

Stop the motor. (PWM outputting is stopped.)

### 8.6.4 Bootstrap Stage Function (C_Stage_Bootstrap)

8.6.4.1 **Syntax**

© 2024
Toshiba Electronic Devices & Storage Corporation

void C_Stage_Bootstrap(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

8.6.4.2 **Variable**

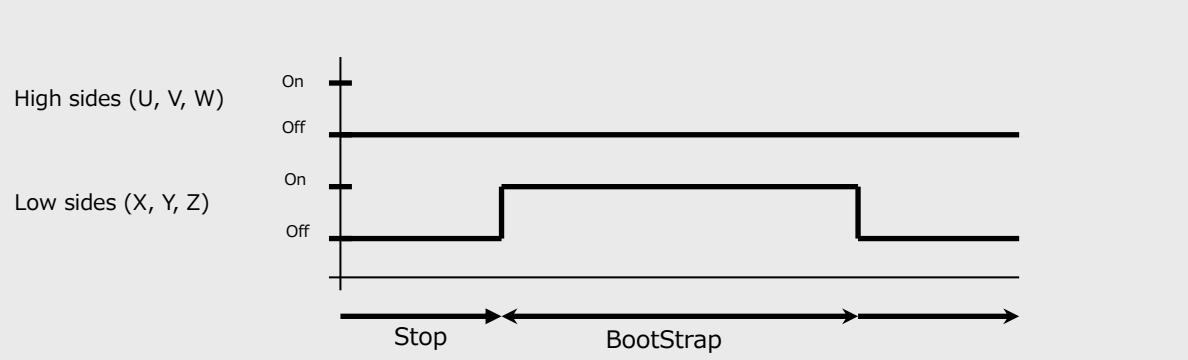
| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Stage.main | Main stage | --- | |
| | Para.time.bootstp | Bootstrap time | Q0 | * MainLoopPrd(s) |
| Input/Output | Stage.sub | Sub-stage | --- | |
| | Stage_counter | Stage counter | Q0 | * MainLoopPrd(s) |
| Output | Stage.itr | Interrupt stage | --- | |
| | Drv.vector_cmd | Vector control commands | --- | (Refer to 8.2.2.) |

8.6.4.3 **Processing Details**

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 stage by dividing it into the following sub-stages:

a) Initial stage
Initializes the bootstrap stage.
b) Delay stage
Waits for the specified bootstrap time to elapse and then transits to the positioning stage.




**8.6.5 Positioning Stage Function (C_Stage_Initposition_0, C_Stage_Initposition_1)**

8.6.5.1 **Syntax**

void C_Stage_Initposition(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

8.6.5.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Stage.main | Main stage | --- | |
| | Boot_type | Startup type | --- | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| | Usr.Id_st_user | Starting Id current | 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 voltages | Q31 | Enabled 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 8.2.2.) |
| | 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 | |

8.6.5.3 **Processing Details**

Fixes the rotor in the initial position.
Fixes 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 stage is divided into the following sub-stages for control.

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

### 8.6.6 Forced Commutation Stage Function (C_Stage_Force_0, C_Stage_Force_1)

8.6.6.1 **Syntax**

void C_Stage_Force(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

8.6.6.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Stage.main | Main stage | --- | |
| | Boot_type | Startup type | --- | Current or Voltage |
| | Usr.Id_st_user | Starting Id current | Q15 | |
| | Usr.omega_user | Control target speed | Q31 | |
| | Para.omega_v2i | Current control switching speed | Q15 | Enabled only during voltage drive |
| | Para.spd_coef | Output voltage coefficient | Q15 | Enabled only during voltage drive |
| | Para.vd_pos | Positioning output voltage | Q31 | Enabled only during voltage drive |
| | Para.omega_min | Forced commutation termination rate | Q15 | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| | 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 8.2.2.) |
| | 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 voltages | Q31 | Enabled only during voltage drive |

8.6.6.3 **Processing Details**

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 $\omega_{com}$ is gradually incremented while the Id is fixed to "start Id current value" and the Iq is fixed to 0.
θ is obtained from $\omega_{com}$ ($\theta = \omega_{com}t$). 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.

### 8.6.7 Change-up Stage Function (C_Stage_Change_up)

8.6.7.1 **Syntax**

void C_Stage_Change_up(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

8.6.7.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Stage.main | Main stage | --- | |
| | Usr.Id_st_user | Starting Id current | Q15 | |
| | Usr.Iq_st_user | Starting Iq current | Q15 | |
| | Usr.omega_user | Control target speed | Q31 | |
| | Para.delta_lambda | Current switching phase at change-up stage | Q0 | |
| | Para.sp_ud_lim_f | Drive speed increase/decrease limit value | Q31 | |
| | Para.time.go_up | Wait 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 8.2.2) |
| | Drv.Id_com | d-axis current command value | Q31 | |
| | Drv.Iq_com | q-axis current command value | Q31 | |

8.6.7.3 **Processing Details**

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.

© 2024
Toshiba Electronic Devices & Storage Corporation

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 Change-up stage into the following sub-stages.

a) Initial stage
   Performs the initialization of the change-up stage.
b) Id, Iq switching stage
   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 stage is made.
c) Delay stage
   Waits for the specified change-up time to elapse, and then transits to the steady stage.

### 8.6.8 Steady Stage Function (C_Stage_Steady_A)

#### 8.6.8.1 **Syntax**

void C_Stage_Steady_A(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

#### 8.6.8.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| 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 8.2.2) |
| | Stage.itr | Interrupt stage | --- | |
| | Drv.Id_com | d-axis current command value | Q31 | |

#### 8.6.8.3 **Processing Details**

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

### 8.6.9 Emergency Stage Function (C_Stage_Emergency)

#### 8.6.9.1 **Syntax**

void C_Stage_Emergency(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

#### 8.6.9.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Stage.main | Main stage | --- | |
| Input/Output | Stage.sub | Sub-stage | --- | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| Output | Drv.vector_cmd | Vector control command | --- | (Refer to 8.2.2) |
| | Stage.itr | Interrupt stage | --- | |

8.6.9.3 **Processing Details**

Transitions to this stage 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 stage recovery process is performed.

### 8.6.10 Shift PWM Control (C_ShiftPWM_Control)

#### 8.6.10.1 Syntax

Static void C_ShiftPWM_Control(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

#### 8.6.10.2 Variable

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Shunt_type | Shunt type | --- | |
| | Stage.itr | Interrupt stage | --- | |
| | Usr.com_user.spwm | Shift PWM control methods | --- | |
| | Drv.omega_com | Drive speed command value | Q31 | |
| | Para.spwm_threshold | Shift PWM switching speed | Q15 | |
| Output | Drv.command.spwm | Shift PWM driving command | --- | |

#### 8.6.10.3 Processing Details

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 Shift PWM driving method conditions

**Table 1 Shift PWM Driving Conditions**

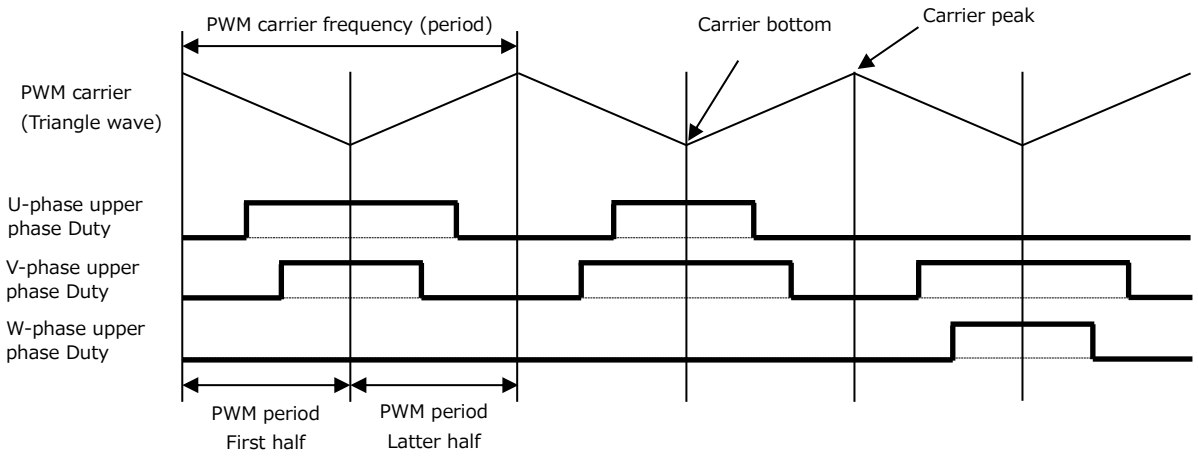
| Shift PWM Control Methods<br>Usr.com_user.spwm | Interrupt Stage<br>Stage.itr | Target Speed<br>Drv.omega_com | Shift PWM Driving Methods<br>Drv.command.spwm |
|---|---|---|---|
| OFF (0) | All | All | OFF (0) |
| ON (1) | Stop | All | OFF (0) |
| | Emergency | All | OFF (0) |
| | CiInitposition_i<br>Force_i<br>Change_up<br>Steady_A | Target speed >= shift switching speed<br>Drv.omega_com >= para.spwm_threshold | OFF (0) |
| | | Target speed < Shift switching speed<br>Drv.omega_com < para.spwm_threshold | ON (1) |

© 2024
Toshiba Electronic Devices & Storage Corporation

## 8.7 Motor Drive Functions

### 8.7.1 Definition and Terminology

#### 8.7.1.1 For 2-Phase Modulation




### 8.7.2 Motor Current and Supply Voltage Acquisition Function (D_GetMotorCurrentPowerVolt)

#### 8.7.2.1 Syntax

void D_GetMotorCurrentPowerVolt (vector_t* const _motor)

Arguments:

vector_t* const _motor : motor control struct

Return value:

None

#### 8.7.2.2 Variable

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Drv.Sector1 | Previous sector | Q0 | 0 to 5 |
| | Para.TrgPosMd | Current detection position mode | --- | 1 Shunt first half<br>1 Shunt latter half |
| | Drv.ADxREG0 | ADC conversion result register address | --- | |
| | Drv.ADxREG1 | ADC conversion result register address | --- | |
| | Drv.ADxREG2 | ADC conversion result register address | --- | |
| | Drv.ADxREG3 | ADC conversion result register address | --- | |
| | Drv.Iao_ave | U-phase zero current average | Q15 | 3 shunt only |
| | Drv.Ibo_ave | V-phase zero current average | Q15 | 3 shunt only |
| | Drv.Ico_ave | W-phase zero current average | Q15 | 3 shunt only |
| | Drv.Idco_ave | Zero current average | Q15 | 1 shunt only |
| | Stage.itr | Interrupt stage | --- | |
| | Drv.idetect_error | Current detection status | --- | |
| | Drv.shift2_mode | Shift 2 control mode | --- | |
| Input/Output | Drv.Ia_raw | U-phase current (raw data) | Q15 | |
| | Drv.Ib_raw | V-phase current (raw data) | Q15 | |
| | Drv.Ic_raw | W-phase current (raw data) | Q15 | |
| Output | Drv.Ia | U-phase current (false detection protection) | Q15 | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| | Drv.Ib | V-phase current (false detection protection) | Q15 | |
| | Drv.Ic | W-phase current (false detection protection) | Q15 | |
| | Drv.Ia_adc | U-phase current conversion result | Q15 | 3 shunt only |
| | Drv.Ib_adc | V-phase current conversion result | Q15 | 3 shunt only |
| | Drv.Ic_adc | W-phase current conversion result | Q15 | 3 shunt only |
| | Drv.Idc1_adc | 1st phase current conversion | Q15 | 1 shunt only |
| | Drv.Idc2_adc | 2nd phase current conversion | Q15 | 1 shunt only |
| | Drv.Vdc | Power supply voltage | Q15 | |
| | Drv.Vdc_adc | Power supply voltage conversion result | Q15 | |

8.7.2.3 **Processing Details**

Retrieves ADC conversion result. Calculates the power supply voltage and motor current.

1. Check the completion of ADC conversion by PMD trig. If the conversion has been completed, obtain the conversion result according to the table below.
   If the conversion is not completed, it waits in the loop until the conversion is completed. (The processing waits in the fail-safe process, though this is not normal.)
   However, when operating in shift PWM2, it is calculated by adding offset, acquisition/calculation is not performed at the timing when the normal PWM/ shift PWM2 is switched.
2. The power supply voltage/motor current is calculated from the obtained ADC conversion.

- Power supply voltage

| | |
|---|---|
| Power supply voltage conversion result<br>Drv.Vdc_adc | ADC Conversion Result 3<br>ADxREG3 |

Since the power supply is unsigned, ADC conversion is 1bit shifted to the right-hand side and used as Q15 format.

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

- Motor current

<Current detection mode: 1 shunt first half, 1 shunt second half>

Current detect mode: Acquires ADC converted value at the first half of PWM cycle in the first half of the shunt cycle.

Current detect mode: Acquires ADC converted value at the position in the second half of PWM cycle in the second half of a shunt cycle.

| Current Acquisition Position / Conversion Result | Latter Half of PWM Cycle | First Half of PWM Cycle |
|---|---|---|
| Current conversion result 1<br>Drv.Idc1_adc | ADC Conversion Result 0<br>ADREG0 | ADC Conversion Result 1<br>ADREG1 |
| Current conversion result 2<br>Drv.Idc2_adc | ADC Conversion Result 1<br>ADREG1 | ADC Conversion Result 0<br>ADREG0 |

Store ADC converted value when the motor is stopped as zero-current in the following parameter.

| | |
|---|---|
| Zero current conversion result<br>Drv.Idco_ave | Current conversion result 1<br>Drv.Idc1_adc |

The value of the three-phase current is calculated from ADC conversion result "zero current conversion result" at the time of zero current (motor-stop) and ADC conversion result at the following two timings (according to the table below).

| | Sector | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| U-phase | -(Idco_ave- | -Ib-Ic | Idco_ave-Idc1_adc | Idco_ave-Idc1_adc | -Ib-Ic | -(Idco_ave- |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | | | |
|---|---|---|---|---|---|---|
| current Drv.Ia | Idc2_adc) | | | | | Idc2_adc) |
| V-phase current Drv.Ib | -Ia-Ic | -(Idco_ave-Idc2_adc) | -(Idco_ave-Idc2_adc) | -Ia-Ic | Idco_ave-Idc1_adc | Idco_ave-Idc1_adc |
| W-phase current Drv.Ic | Idco_ave-Idc1_adc | Idco_ave-Idc1_adc | -Ia-Ib | -(Idco_ave-Idc2_adc) | -(Idco_ave-Idc2_adc) | -Ia-Ib |

### 8.7.3 Input Coordinate Transformation Function (D_InputTransformation)

8.7.3.1 **Syntax**

void D_InputTransformation(vector_t* const _motor)

Arguments:

vector_t* const _motor : motor control struct

Return value:

None

8.7.3.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Drv.vector_cmd.F_vcomm_Idetect | Current detection command | --- | |
| | Drv.Ia | U-phase current | Q15 | |
| | Drv.Ib | V-phase current | Q15 | |
| | (drv.Ic) | W-phase current | Q15 | |
| | Drv.theta | Rotor position | Q0 | [deg/360] |
| | Para.pos.ctrlprd | Vector control period | Q0 | |
| | Drv.omega | Estimated speed | Q15 | |
| | Drv.idetect_error | Current detection status | --- | 0:Detectable<br>1:Undetectable |
| | Drv.shift2_mode | Shift 2 control mode | --- | |
| Output | Drv.Ialpha | α-axis current | Q15 | |
| | Drv.Ibeta | β-axis current | Q15 | |
| | Drv.Id | d-axis current | Q15 | |
| | Drv.Iq | q-axis current | Q15 | |

8.7.3.3 **Processing Details**

Converts 3-phase current (Ia, Ib, Ic) to dq axis current.

1. 8.7.4 Clark transformation converts three-phase currents (Ia, Ib, Ic) into two-phase currents (Iα, Iβ).
2. 8.7.5 Park transformation converts the two-phase current (Iα, Iβ) in stationary coordinates to the dq-axis transformation in rotating coordinates.

   The rotor position to be used for the park transformation uses θest predicted from the speed ω and the calculation period, because the actual rotor position is advanced from the point of calculation in the position estimation process. θest = θ + speed ω × vector control period.

When the current cannot be detected normally or when switching between normal PWM and shift, conversion is not performed and Id,Iq are not updated.

### 8.7.4 Clarke Transformation Function (E_Clarke)

8.7.4.1 **Syntax**

void E_Clarke( q15_t _iu,

© 2024
Toshiba Electronic Devices & Storage Corporation

```
                q15_t _iv,
                q15_t _iw,
                q15_t* _ialpha,
                q15_t* _ibeta)
```

Arguments:

| | |
|---|---|
| _iu | U-phase current |
| _iv | V-phase current |
| _iw | W-phase current |
| _ialpha | α-axis current storage address |
| _ibeta | β-axis current storage address |

Return value:

None

8.7.4.2 **Processing Details**

Converts 3-phase current (Iu, Iv, Iw) into 2-phase (Iα, Iβ).

Iα and Iβ are calculated using the following formula.

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

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

The coefficient 2/3 is a coefficient for making the amplitudes equivalent, since the amplitude becomes 3/2 times when converting a three-phase current to a two-phase αβ-axis current.

Since the sum of the three-phase current is 0, when Iu+Iv+Iw=0,

$$I\alpha = Iu$$

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

*It becomes.*

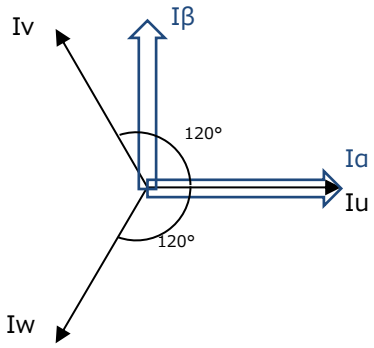



**Fig. 5 Clarke Transformation**

### 8.7.5 Park Transformation Function (E_Park)

8.7.5.1 **Syntax**

```
void E_Park(    q15_t _ialpha,
                q15_t _ibeta,
                Uint16_t _theta,
                q15_t* _id,
                q15_t* _iq)
```

Arguments:

| | |
|---|---|
| _ialpha | α-axis current Iα |
| _ibeta | β-axis current Iβ |
| _theta | Rotor position θ: 0 ≤ position < 360°(0 to 0xffff) |
| _id | d-axis current Id storage address |
| _iq | q-axis current Iq storage address |

Return value:

None

8.7.5.2 **Processing Details**

Converts the stationary coordinate of αβ axis to dq axis of the rotational coordinate.

Id, Iq is calculated using the formula below.

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

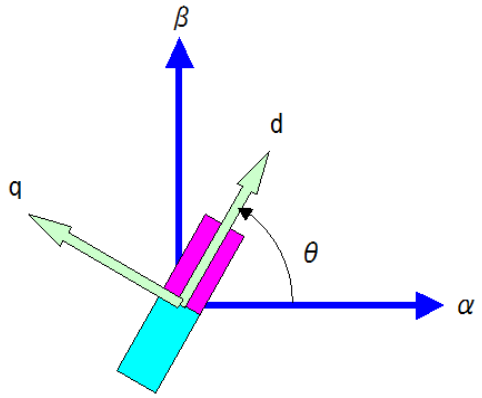



**Fig. 6 Park Transformation**

### 8.7.6 Position Estimation Function (D_Detect_Rotor_Position)

8.7.6.1 **Syntax**

void D_Detect_Rotor_Position(vector_t* const _motor)

Arguments:

vector_t* const _motor : motor control struct

Return value:

None

8.7.6.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| 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 cycle | 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 | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| | Drv.Ed_PI | d-axis induced-voltage PI | Q31 | |
| | Drv.omega | Estimated speed | Q31 | |
| Output | Drv.theta | Rotor position | Q0 | |

8.7.6.3 **Processing Details**

PI control is performed by taking the estimated speed $\omega_{est}$ 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 $\omega_{est}$ obtained by PI-control.
The estimated speed omega uses the drive speed command value omega_com as the estimated speed omega if F_vcomm_omega is CLEAR.

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

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

(pId=0 can be set from p = d/dt, Id ≠ fixed value.)

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

Therefore, the d-axis induced-voltage Ed can be calculated using the formula below.

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

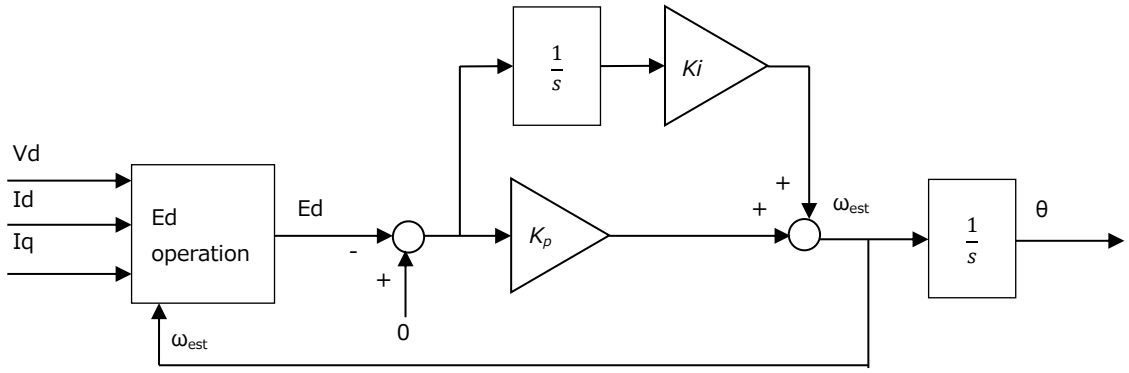



**Fig. 7 Block-Diagram of Position Estimation by d-axis Induced Voltage Ed**

### 8.7.7 Speed Control Function (D_Control_Speed)

8.7.7.1 **Syntax**

void D_Control_Speed(vector_t* const _motor)
Arguments:
vector_t* const _motor : motor control struct
Return value:
None

8.7.7.2 **Variable**

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| 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 | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| | 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 | |

8.7.7.3 **Processing Details**

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.

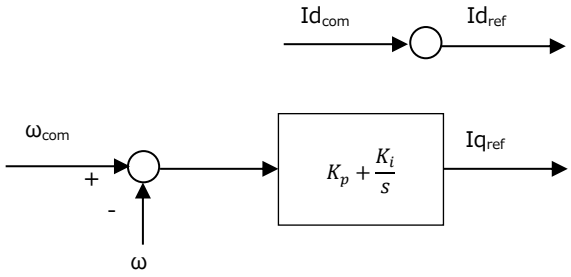



**Fig. 8 Block Diagram of Speed Control**

## 8.7.8 Current Control Function (D_Control_Current)

### 8.7.8.1 Syntax

void D_Control_Current(vector_t* const _motor)

Arguments:
vector_t* const _motor : motor control struct
Return value:
None

### 8.7.8.2 Variable

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Drv.vector_cmd.F_vcomm_volt | Voltage command control command | --- | |
| | Stage.itr | Interrupt stage | --- | |
| | Drv.Id_ref | d-axis current reference value | Q15 | |
| | Drv.Iq_ref | q-axis current reference value | Q15 | |
| | Drv.Id | d-axis current | Q15 | |
| | Drv.Iq | q-axis current | Q15 | |
| | Drv.Vd_com | d-axis voltage command value | Q31 | |
| | Drv.Vq_com | q-axis voltage command value | Q31 | |
| | Drv.Vd_out | Output voltages | | Enabled only during voltage drive |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| | Para.crt.dkp | d-axis current control proportional gain | Q15 | Q12 available |
| | Para.crt.dki | d-axis current control integral gain | Q15 | Q12 available |
| | Para.crt.qkp | q-axis current control proportional gain | Q15 | Q12 available |
| | Para.crt.qki | q-axis current control integral gain | Q15 | Q12 available |
| | Drv.shift2_mode | Shift 2 control mode | --- | |
| | Drv.Vd_lim_shift2 | d-axis voltage limit value | Q31 | |
| | Drv.Vq_lim_shift2 | q-axis voltage limit value | Q31 | |
| Input/Output | Drv.Vd_I | d-axis voltage integral value | Q31 | |
| | Drv.Vq_I | q-axis voltage integral value | Q31 | |
| Output | Drv.Vd | d-axis voltage | Q31 | |
| | Drv.Vq | q-axis voltage | Q31 | |

#### 8.7.8.3 Processing Details

Determines the d-axis and q-axis voltages from the deviation between the current reference value and the actual current. When the voltage command control command is SET, the d-axis voltage and q-axis voltage are calculated by PI control below. When it is CLEAR, the d-axis voltage, q-axis voltage, and integral are set to the command values. PI control is performed as the controlled variable is the q-axis current Iq and the manipulated variable is q-axis voltage Vq.

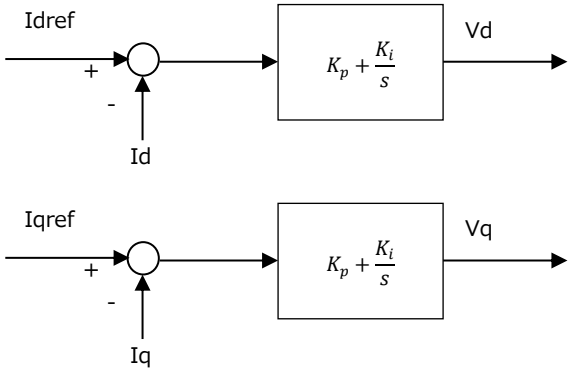



**Fig. 9 Block Diagram of Current Control**

### 8.7.9 Output Coordinate Transformation Function (D_OutputTransformation)

#### 8.7.9.1 Syntax

void D_OutputTransformation (vector_t* const _motor)

Arguments:
vector_t* const _motor : motor control struct
Return value:
None

#### 8.7.9.2 Variable

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Drv.vector_cmd.F_vcomm_onoff | Motor output control command | --- | |
| | Drv.Vd | d-axis voltage | Q31 | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | |
|---|---|---|---|---|
| | Drv.Vq | q-axis voltage | Q31 | |
| | Drv.Vdc | Power supply voltage | Q15 | |
| | Drv.theta | Rotor position | Q0 | [deg/360] |
| | Drv.PhCvMd | Phase conversion mode | --- | 0:Space vector<br>1:Inverse Clarke |
| Input/Output | Drv.Sector | Sector | --- | |
| Output | Drv.Valpha | α-axis voltage | Q31 | |
| | Drv.Vbeta | β-axis voltage | Q31 | |
| | Drv.DutyU | U-phase PWM Duty rate | Q15<br>(unsigned) | 0.0 to 1.0 |
| | Drv.DutyV | V-phase PWM Duty rate | Q15<br>(unsigned) | 0.0 to 1.0 |
| | Drv.DutyW | W-phase PWM Duty rate | Q15<br>(unsigned) | 0.0 to 1.0 |
| | Drv.Sector1 | Previous sector | --- | |

#### 8.7.9.3 Processing Details

Converts dq axis voltage (Vd,Vq) to PWMDuty ratio (DutyU, DutyV, DutyW) of the three phases.

1. Converts the dq-axis voltage (Vd, Vq) to the αβ-axis voltage (Vα, Vβ) using 8.7.10 reverse park transformation.
2. Calculates the current "sector" from the αβ axis voltage (Vα, Vβ). Save the "sector" before operation to the "last sector" and use it when acquiring the motor current.
3. Converts αβ axis voltage (Vα, Vβ) to 3-phase duty ratio (DutyV, DutyW, DutyW) using 8.7.12 spatial vector modulation.

### 8.7.10 Reverse Park Transformation Function (E_InvPark)

#### 8.7.10.1 Syntax

```
void E_InvPark( q31_t _d,
                q31_t _q,
                Uint16_t _theta,
                q31_t* _alpha,
                q31_t* _beta)
```

Arguments:

| | |
|---|---|
| _d | d axis |
| _q | q-axis |
| _theta | Rotor position θ: 0 ≤ position < 360°(0 to 0xffff) |
| _alpha | α-axis stored address |
| _beta | β-axis storage address |

Return value:

None

#### 8.7.10.2 Processing Details

Converts dq axis of the rotational coordinate to the αβ axis of the stationary coordinate.

Vα and Vβ are calculated by the following formula.

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

### 8.7.11 Sector Operation Function (D_CalSector)

#### 8.7.11.1 Syntax

```
Uint8_t D_CalSector(    q31_t _valpha,
                        q31_t _vbeta)
```

© 2024
Toshiba Electronic Devices & Storage Corporation

Arguments:

_valpha α-axis voltage

_vbeta β-axis voltage

Return value:

Sector

#### 8.7.11.2 Processing Details

Calculates "sector" from αβ axis voltage.

Save the "Last Sector" value and determine the current "Sector" value according to the conditional expression below.

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

### 8.7.12 Spatial Vector Modulation Function (D_SVM)

#### 8.7.12.1 Syntax

```
void D_SVM(    q31_t _valpha,
               q31_t _vbeta,
               q15_t _vdc,
               Uint8_t _sector,
               Uint8_t _modul,
               Uint16_t* _p_vu,
               Uint16_t* _p_vv,
               Uint16_t* _p_vw)
```

Arguments:

_valpha α-axis Vα

_vbeta β-axis Vβ

_vdc Power supply voltage

_sector Sector

_modul Modulation type

_p_vu U-phase Duty ratio DutyU storage address

_p_vv V-phase Duty ratio DutyV storage address

_p_vw W-phase Duty ratio DutyW storage address

Return value:

None

#### 8.7.12.2 Processing Details

Calculates Duty rate of the 3-phase PWM from the αβ-axis of the 2-phase.

© 2024
Toshiba Electronic Devices & Storage Corporation

PWM Duty factor of each phase is obtained by spatial vector modulation.

- What is spatial vector modulation?

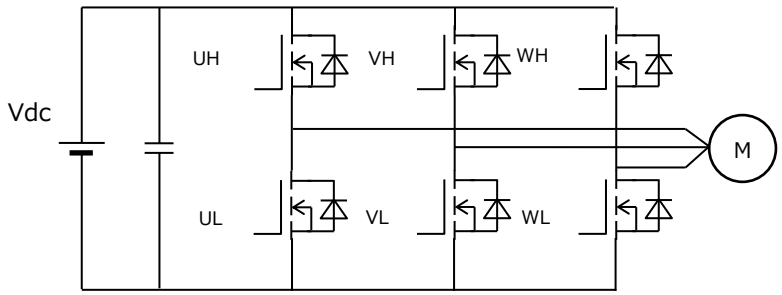
As shown in Table 2, the status of the switching elements in the inverter drive circuit is 8 patterns, since the elements in the upper phase are in ON or OFF state and the elements in the lower phase are in the opposite state compared to the upper phase (ignoring dead time).

Table 2 Inverter switching status1

These 8 patterns are referred to as V0 to V7 vectors, and when expressed in a diagram, V0 and V7 vectors exist at the origin because no line voltage is applied, and the remaining six vectors (V1 to V6) are expressed in increments of 60° as shown in Figure 11 Space Vector Diagram.

By combining two adjacent vectors, an arbitrary output voltage vector V can be obtained.

This section explains how to calculate PWM Duty when the output-voltage vector V exists in sector 0.



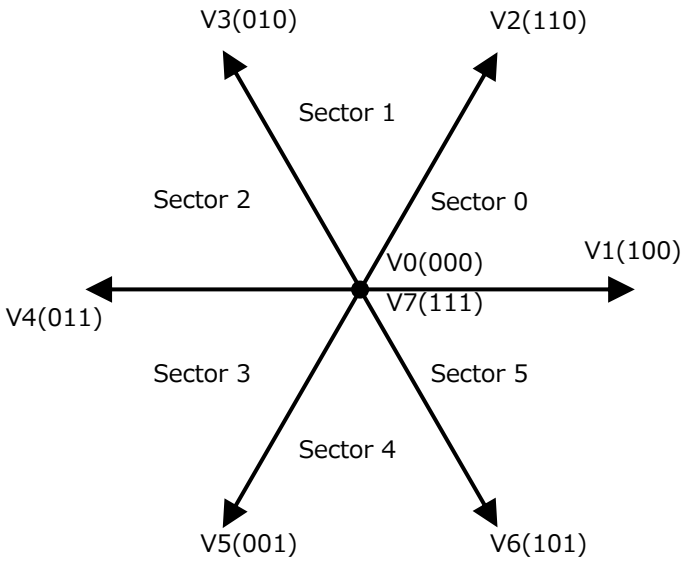

UH: U-phase upper phase switching element
UL: U-phase lower phase switching element
VH: V-phase upper phase switching element
VL: V-phase lower phase switching element
WH: W-phase upper phase switching element
WL: W-phase lower phase switching element
Vdc: Power supply voltage

Table 2 Inverter switching status1

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

0: Upper phase OFF, lower phase ON
1: Upper phase ON, lower phase OFF

**Fig. 10 Inverter Drive Circuit**




© 2024
Toshiba Electronic Devices & Storage Corporation

**Fig. 11 Space Vector**

・When output voltage V exists in sector 0

For example, in Fig. 12, the composite vector V of Vα and Vβ lies on sector 0 and can be obtained as a composite vector of vector V1', V2' obtained by multiplying voltage-vector V1 and V2 by coefficient t1, t2, respectively. When expressed as a PWM wave form, an output-voltage vector V is synthesized by generating V1, V2 for a t1, t2 of times in PWM half cycle as shown in Fig. 13 PWM Waveform for 3-Phase Modulation

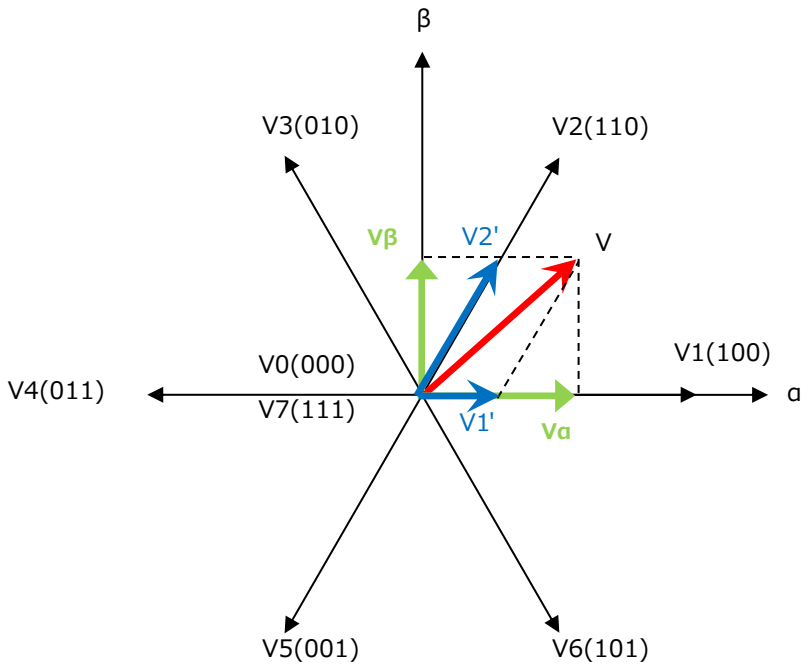



**Fig. 12 Voltage Vector for Sector 0**

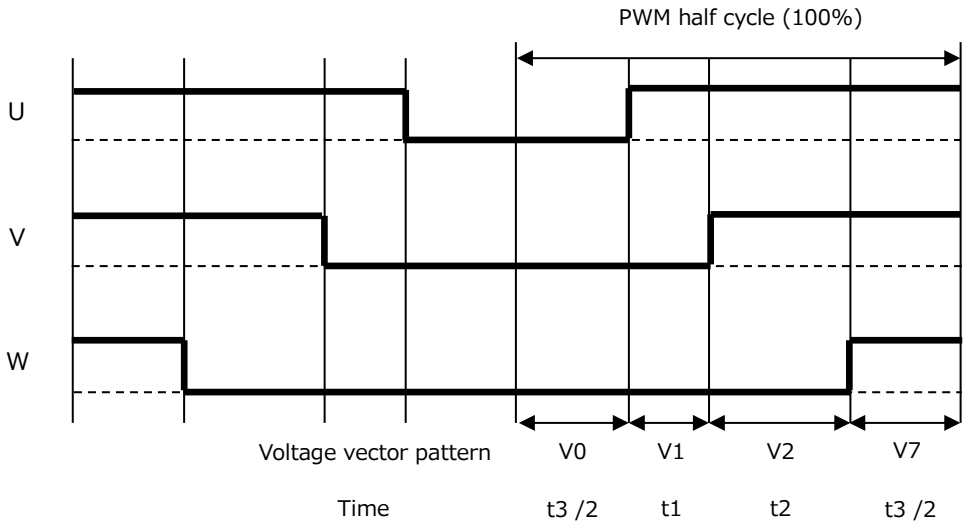



**Fig. 13 PWM Waveform for 3-Phase Modulation**

Fig. 12 Voltage Vector for Sector 0

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

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

Can be expressed as.

Therefore, V1′, V2′ is

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

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

When the power supply is turned Vdc,

$$V1' = t1 \times Vdc$$

$$V2' = t2 \times Vdc$$

Therefore, the ratio of t1,t2,t3, becomes

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

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

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

K is the conversion factor, and is 3/2 for keeping the amplitude constant during 3-phase to 2-phase conversion.

Three-phase modulation takes V0,V7 vector generation time as a t3/2, and two-phase modulation takes V0 vector generation time as t3 and V7 vector generation time as 0.

Therefore, Duty of 3-phase can be calculated from t1,t2,t3 below.

U-phaseDuty = t1+t2+(t3/2)

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

W-phase Duty=t3/2

Similarly, for each sector of the output voltage, calculate OFF Duty D0, Duty D1 for 1-phase ON, and Duty D2 for 2-phase ON using the formula for Duty ON time for each sector in Table 3.

**Table 3 Calculation of Duty ON Time for Each Sector**

| Sector | Duty when only 1-phase is ON D1 | Duty when 2-phases are ON D2 | OFF Duty D0 |
|---|---|---|---|
| 0 | $k \times \frac{V\alpha - \frac{1}{\sqrt{3}} V\beta}{Vdc}$ | $k \times \frac{\frac{2}{\sqrt{3}} V\beta}{Vdc}$ | 100% − D1 − D2 |
| 1 | $k \times \frac{-V\alpha + \frac{1}{\sqrt{3}} V\beta}{Vdc}$ | $k \times \frac{V\alpha + \frac{1}{\sqrt{3}} V\beta}{Vdc}$ | |
| 2 | $k \times \frac{\frac{2}{\sqrt{3}} V\beta}{Vdc}$ | $k \times \frac{-V\alpha - \frac{1}{\sqrt{3}} V\beta}{Vdc}$ | |
| 3 | $-k \times \frac{\frac{2}{\sqrt{3}} V\beta}{Vdc}$ | $k \times \frac{-V\alpha + \frac{1}{\sqrt{3}} V\beta}{Vdc}$ | |
| 4 | $k \times \frac{-V\alpha - \frac{1}{\sqrt{3}} V\beta}{Vdc}$ | $k \times \frac{V\alpha - \frac{1}{\sqrt{3}} V\beta}{Vdc}$ | |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | |
|---|---|---|---|
| 5 | $k \times \frac{V\alpha + \frac{1}{\sqrt{3}}V\beta}{Vdc}$ | $-k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$ | |

k is the coefficient for keeping the amplitude constant before and after the conversion, and it is 3/2.

Duty of U,V,W phase from D0,D1,D2 is calculated using the calculation formulas in Table4.

**Table 4 Relation Between Sector and Phase Duty**

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

### 8.7.13 Trigger Timing Operation Function (D_CalTrgTiming)

#### 8.7.13.1 Syntax

void D_CalTrgTiming(vector_t* const _motor)

Arguments:
_motor Motor control structure
Return value:
None

#### 8.7.13.2 Variable

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Drv.DutyU | U-phase PWM Duty rate | Q15 | 0.0 to 1.0 |
| | Drv.DutyV | V-phase PWM Duty rate | Q15 | 0.0 to 1.0 |
| | Drv.DutyW | W-phase PWM Duty rate | Q15 | 0.0 to 1.0 |
| | Drv.command.modul | Modulation type | --- | 2-phase modulation |
| | Drv.TrgPosMd | Current detection position mode | --- | 1 shunt (first half)<br>1 shunt (latter half) |
| | Para.TrgComp | Trigger timing correction value | Q15 | -1.0 to 1.0 |
| Output | Drv.AdTrg0 | ADC Trigger Timing 0 (TRG0) | Q15 | -1.0 to 1.0 |
| | Drv.AdTrg1 | ADC Trigger Timing 1 (TRG1) | Q15 | -1.0 to 1.0 |

#### 8.7.13.3 Processing Details

Calculates the trigger timing for acquiring ADC conversion value of the motor current and supply voltage Vdc.

- For 1 shunt

The basic trigger position is calculated from PWM Duty of U,V,W phase.

© 2024
Toshiba Electronic Devices & Storage Corporation

If a value other than 0 is entered in the trigger timing compensation value, it will be corrected from the calculated trigger position.

The triggering position of the supply-voltage Vdc is fixed at the position of the carrier end. (set at the time of initialization and it is not handled by this function)

| | Modulation Type | Trigger Timing 0 (TRG0) Motor Current | Trigger Timing 1 (TRG1) Motor Current | Trigger Timing 2 (TRG2) Power Supply Vdc |
|---|---|---|---|---|
| Latter Half of PWM Cycle | 3-phase modulation | Duty intermediate value | Duty maximum value | Carrier end (fixed) |
| | 2-phase modulation | Duty intermediate value | Duty maximum value | Carrier end (fixed) |
| First Half of PWM Cycle | 3-phase modulation | −Duty intermediate value | −Duty minimum value | Carrier end (fixed) |
| | 2-phase modulation | −Duty intermediate value | Duty intermediate value | Carrier end (fixed) |

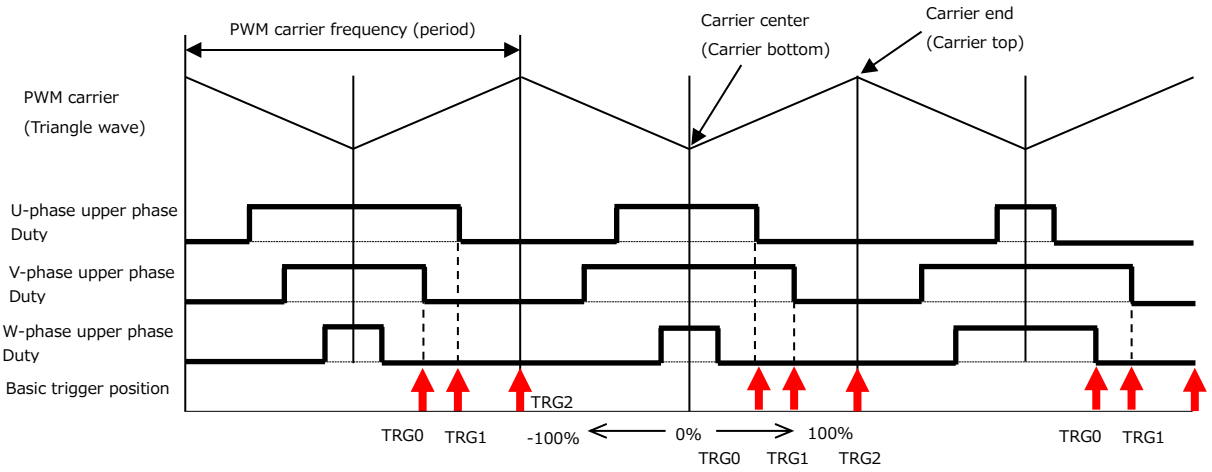



**Fig. 14 Trigger Positions for 1-shunt, Second Half of PWM Period, and 3-phase Modulation**

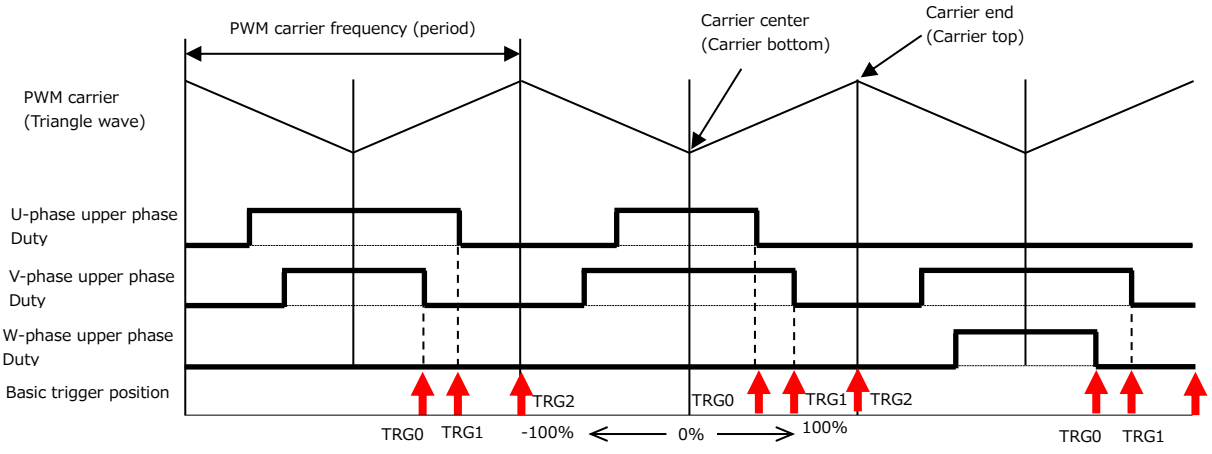



**Fig. 15 Trigger Positions for 1-shunt, Second Half of PWM Period and 2-phase Modulation**

© 2024
Toshiba Electronic Devices & Storage Corporation

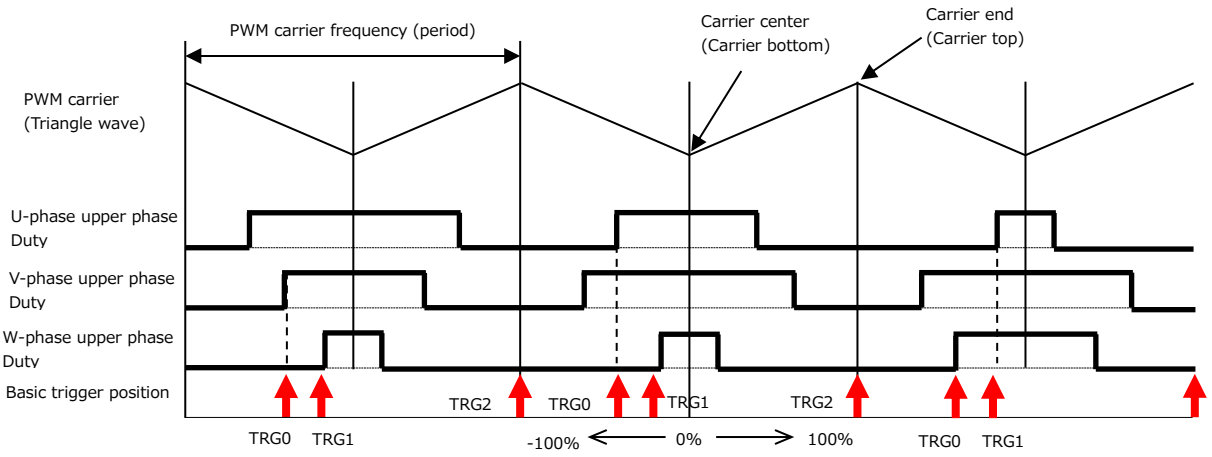



**Fig. 16 Trigger Positions for 1-shunt, First Half of PWM Period and 3-phase Modulation**

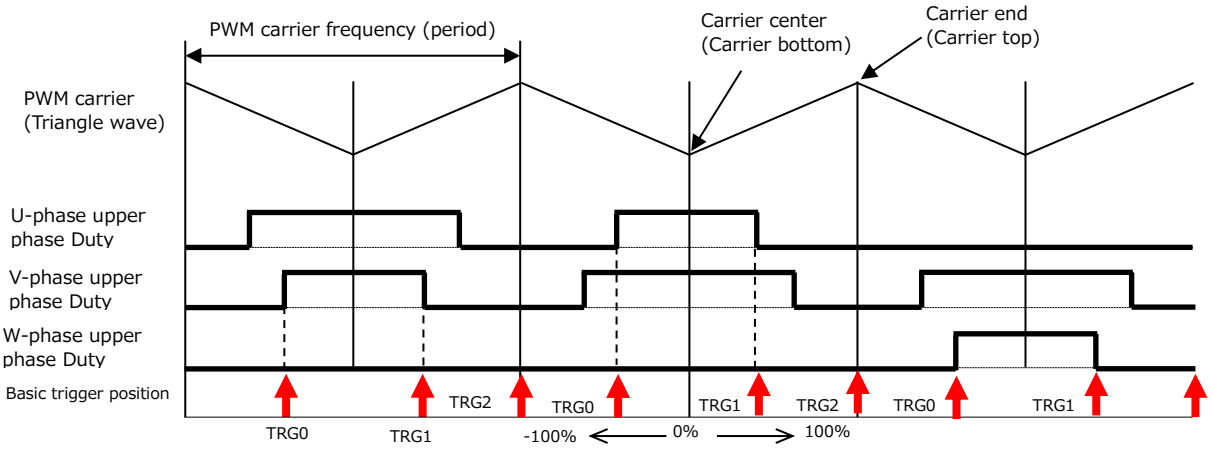



**Fig. 17 Trigger Positions for 1-shunt, First Half of PWM Period and 2-phase Modulation**

### 8.7.14 PWM Register Setting Function (PMD_RegDataSet)

#### 8.7.14.1 Syntax

```
void PMD_RegDataSet(TSB_PMD_TypeDef* const PMDx,
                    Uint16_t duty_u,
                    Uint16_t duty_v,
                    Uint16_t duty_w,
                    q15_t adtrg0,
                    q15_t adrtg1)
```

Arguments:

| | |
|---|---|
| PMDx | PMD register address |
| Duty_u, | U-phase PWM Duty rate |
| Duty_v, | V-phase PWM Duty rate |
| Duty_w, | W-phase PWM Duty rate |
| Adtrg0, | ADC trigger timing 0 position |
| Adrtg1 | ADC trigger timing 1 position |

Return value:

None

#### 8.7.14.2 Processing Details

Set the three-phase PWM Duty and trigger timing to registers.

© 2024
Toshiba Electronic Devices & Storage Corporation

Perform the operation below and set the input arguments in PMD register.
(M4KNA)
TRGCMP0 = (adtrg0 + 32768) ÷2
TRGCMP1 = (adtrg1 + 32768) ÷2
CMPU = duty_u
CMPV = duty_v
CMPW = duty_w

### 8.7.15 Current False Positive Detection Function (D_Check_DetectCurrentError)

#### 8.7.15.1 Syntax

Int D_Check_DetectCurrentError (vector_t* const _motor)

Arguments:
_motor Motor control structure
Return value:
Current detection status

#### 8.7.15.2 Variable

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | Para.chkpls | Pulse check width | --- | |
| | Shunt_type | Shunt type | --- | 1:1 shunt |
| | Drv.command.modul | Modulation type drive command | --- | 2-phase modulation |
| | Drv.DutyU | U-phase PWM Duty rate | Q15 | 0.0 to 1.0 |
| | Drv.DutyV | V-phase PWM Duty rate | Q15 | 0.0 to 1.0 |
| | Drv.DutyW | W-phase PWM Duty rate | Q15 | 0.0 to 1.0 |
| Output | Drv.idetect_error | Current detection status | --- | 0:Detectable<br>1:Undetectable |

#### 8.7.15.3 Processing Details

Detects that current cannot be detected due to PWM Duty width.
During 3-shunt, it detects if the middle value of Duty is larger than the set value.
During 1-shunt, this function detects that the magnitude of difference between the minimum Duty width and the Duty width is smaller than the set value. Note that the minimum Duty width when using 1-shunt and 2-phase modulation is the mid Duty width of the 3-phase Duty width.

・For 1-shunt
When the difference between the PWM Duty width and the PWM Duty is smaller than the set value, the detection result is determined to be NG.

### 8.7.16 Inverter Output Voltage Calculation Function (VE_GET_Cal_Vdq)

#### 8.7.16.1 Syntax

q15_t VE_GET_Cal_Vdq ( q15_t _vd, q15_t _vq)

Arguments:
_vd d-axis voltage
_vq q-axis voltage

Return value:
Inverter output-voltage (Vdq)

© 2024
Toshiba Electronic Devices & Storage Corporation

#### 8.7.16.2 Variable

| Direction | Name | Meaning | Q Format | Remark |
|---|---|---|---|---|
| Input | _vd | d-axis voltage | Q15 | 0.0 to 1.0 |
| | _vq | q-axis voltage | Q15 | 0.0 to 1.0 |
| Output | --- | Inverter output-voltage (Vdq) | Q15 | 0.0 to 1.0 |

#### 8.7.16.3 Processing Details

The inverter output-voltage (Vdq) is calculated.

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

© 2024
Toshiba Electronic Devices & Storage Corporation

## 8.8 Temperature Protection Control Function

### 8.8.1 Temperature Protection Control Function (B_Protect_Temperature)

#### 8.8.1.1 Syntax

void B_Protect_Temperature(temperature_t * const _temp, vector_t * const _motor)

Arguments:

| | |
|---|---|
| _temp | Temperature protection control structure |
| _motor | Motor control structure |

Return value:

None

#### 8.8.1.2 Processing Details

Obtains ADC value of each temperature-sensor and checks the error status.

### 8.8.2 Temperature Acquisition Function (B_Protect_GetTemperature)

#### 8.8.2.1 Syntax

void B_Protect_GetTemperature(temperature_t * const _temp)

Arguments:

| | |
|---|---|
| _temp | Temperature protection control structure |

Return value:

None

#### 8.8.2.2 Processing Details

Obtains the following temperatures by ADC conversion:
OutRoom, OutPipe, OutEXHAUST, HVMOS, IGBT and Diode.

### 8.8.3 Temperature Error Checking Function (B_Protect_CheckTemperatureError)

#### 8.8.3.1 Syntax

void B_Protect_CheckTemperatureError(temperature_t * const _temp)

Arguments:

| | |
|---|---|
| _temp | Temperature protection control structure |

Return value:

None

#### 8.8.3.2 Processing Details

Determines open or short circuit errors from ADC values of the temperature-sensors.

### 8.8.4 Temperature Control Status Selection Function (B_Protect_TemperatureStatusSel)

#### 8.8.4.1 Syntax

void B_Protect_TemperatureStatusSel(tc_Lim_t * _tc, int16_t _t, uint16_t times)

Arguments:

| | |
|---|---|
| _tc | Speed limiting structure |
| _t | Temperature |
| _times | Number of sampling |

Return value:

© 2024
Toshiba Electronic Devices & Storage Corporation

None

#### 8.8.4.2 Processing Details

Determines the error status according to the current temperature.

### 8.8.5 Temperature Protection Error Determination Function (B_Error_Collection)

#### 8.8.5.1 Syntax

void B_Error_Collection(void)

Arguments:
None
Return value:
None

#### 8.8.5.2 Processing Details

Determines the error of the fan motor/compressor motor PFC in relation to the temperature and displays the error according to the set priorities.

### 8.8.6 Error Display Control Function (B_Error_DisplayCtrl)

#### 8.8.6.1 Syntax

void B_Error_DisplayCtrl(void)

Arguments:
None
Return value:
None

#### 8.8.6.2 Processing Details

Set the displays of LED1 to 4 for the error status. Sets the number of blinks of 500 ms interval to repeat at every 2-second interval for always on when there is no error and for the specified LED when there is an error.
The display contents of each error are as follows.

・Fan motors

| | |
|---|---|
| No error | : LED2 off |
| Overcurrent (Hard) | : LED2 blinks once |
| Over current (Soft) | : LED2 blinks twice |

・Compressor motor

| | |
|---|---|
| No error | : LED3 off |
| Overcurrent (Hard) | : LED3 blinks once |
| Overcurrent (Soft) | : LED3 blinks twice |

・PFC

| | |
|---|---|
| No error | : LED4 ON |
| Overcurrent (Hard) | : LED4 blinks once |
| Overcurrent (Soft) | : LED4 blinks twice |
| AC overvoltage | : LED4 blinks three times |
| AC under-voltage | : LED4 blinks 4 times |
| DC overvoltage | : LED4 blinks 5 times |
| DC under-voltage | : LED4 blinks 6 times |
| AC voltage-frequency | : LED4 blinks 7 times |

© 2024
Toshiba Electronic Devices & Storage Corporation

Zero cross : LED4 blinks 8 times

・Temperature

No error : LED1 ON
Sensor open/short circuit : LED1 blinks once
Abnormal temperature : LED1 blinks twice

### 8.8.7 Error Display Function (B_Error_Display)

#### 8.8.7.1 Syntax

void B_Error_Display(uint8_t _time, uint8_t _sLED, uint8_t i)

Arguments:

_time Number of blinks
_sLED Selection LED
I LED number

Return value:

None

#### 8.8.7.2 Processing Details

Controls LED blinking interval, ON/OFF switching, and the blinking interval period. In addition, LED output is according to the setting.

© 2024
Toshiba Electronic Devices & Storage Corporation

# 9. PFC Control

## 9.1 Overview

PFC is an inverter load that operates like a resistive load to the power system. This makes the input current wave closer to a sine wave in response to the sine wave input voltage. This reduces unnecessary energy dissipation and suppresses harmonics, thereby preventing electronic equipment from malfunctioning and causing injuries.

PFC function improves power factor (PF) of AC power supply (the ratio of the actual drive power to the apparent power) to bring it close to 1.0.

This device controls single-phase PFC using diode bridges.

## 9.2 Configuration of Single-Phase PFC

The configuration of a single-phase PFC is shown below.

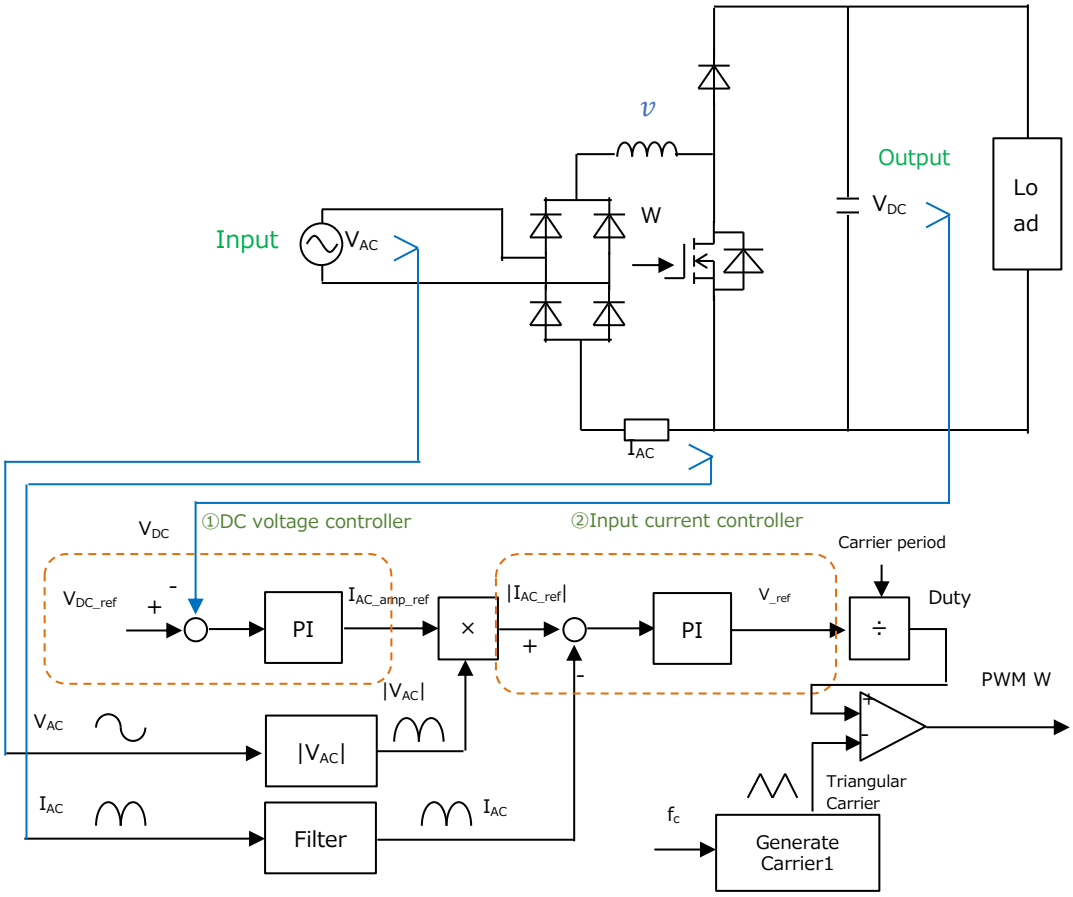



**Fig. 18 Block Diagram of PFC Control**

## 9.3 PFC Drive

### 9.3.1 PFC Interrupt Control (HPFC_Control_Int)

#### 9.3.1.1 Syntax

void HPFC_Control_Int(void)

Arguments:
None
Return value:
None

#### 9.3.1.2 Processing Details

Executes PFC control when ADC conversion is completed.
If there is no error, register setting is done for current control, PWM Duty and ADC trigger, and in case of an error, PWM output is OFF.

### 9.3.2 PFC ADC Conversion Completion Interrupt Handling (HPFC_INT_ADC_Fin)

#### 9.3.2.1 Syntax

void HPFC_INT_ADC_Fin(void)

Arguments:
None
Return value:
None

#### 9.3.2.2 Processing Details

This function handles interrupts when PFC ADC conversion is completed.
Used to acquire the phase current and to perform PFC control when ADC conversion is complete.

### 9.3.3 PFC Control Main Function (HPFC_Control_Main)

#### 9.3.3.1 Syntax

void HPFC_Control_Main(void)

Arguments:
None
Return value:
None

#### 9.3.3.2 Processing Details

Controls DC voltage.
The voltage is boosted to the start voltage at a fixed interval. After the start voltage is reached, control is performed with the target voltage.

### 9.3.4 PFC User Control Function (HPFC_UserControl)

#### 9.3.4.1 Syntax

void HPFC_UserControl(void)

Arguments:
None
Return value:
None

#### 9.3.4.2 Processing Details

Controls ON/OFF of PFC.
ON is set when during steady stage the AC current exceeds a certain value, and is set OFF at the timing when AC current falls below a certain value.

### 9.3.5 PFC Error Checking Function (HPFC_INT_ERROR_CHECK_16k)

#### 9.3.5.1 Syntax

© 2024
Toshiba Electronic Devices & Storage Corporation

void HPFC_INT_ERROR_CHECK_16k(void)

Arguments:
None
Return value:
None

#### 9.3.5.2 Processing Details

The following error checks are performed on the PFC.
Compressor motor error, DC voltage (overvoltage, undervoltage) and AC voltage (overvoltage, undervoltage, frequency, zero-crossing).

### 9.3.6 PFC AC/DC Interrupt Handling Function (HPFC_VacVdc_Handle)

#### 9.3.6.1 Syntax

void HPFC_VacVdc_Handle(pfc_t* const _pfc)

Arguments:
_pfc PFC control constructs
Return value:
None

#### 9.3.6.2 Processing Details

During interrupts, DC voltage acquisition/calculation, AC voltage acquisition/calculation, and voltage control are performed.

### 9.3.7 PFC Current Control Function (HPFC_ControlCurrent)

#### 9.3.7.1 Syntax

void HPFC_ControlCurrent(pfc_t * const _pfc, phase_mode_e _mode)

Arguments:
_pfc PFC control constructs
_mode Phase setting
Return value:
None

#### 9.3.7.2 Processing Details

Calculates Duty of PWM from AC current.

### 9.3.8 PFC Target DC Voltage Setting Function (HPFC_VdcTarget)

#### 9.3.8.1 Syntax

void HPFC_VdcTarget(pfc_t * const _pfc, vector_t * const _motor)

Arguments:
_pfc PFC control constructs
_motor Motor control structure
Return value:
None

#### 9.3.8.2 Processing Details

© 2024
Toshiba Electronic Devices & Storage Corporation

When the difference between the present DC voltage calculated from AC voltage and the target DC voltage exceeds 5 V, the voltage is boosted to the target voltage at a fixed rate.

### 9.3.9 PFC Current Control Function (HPFC_INT_pwmA)

#### 9.3.9.1 Syntax

void HPFC_INT_pwmA(void)

Arguments:
None
Return value:
None

#### 9.3.9.2 Processing Details

The trigger is calculated for the calculated PWM Duty.

### 9.3.10 PFC Current Control Function (HPFC_CalVacFilter_16k)

#### 9.3.10.1 Syntax

void HPFC_CalVacFilter_16k(pfc_t * const _pfc)

Arguments:
_pfc PFC control constructs
Return value:
None

#### 9.3.10.2 Processing Details

The peak voltage and bottom voltage are memorized from the acquired AC voltage, and the zero-cross judgment is performed. AC voltage replaces the negative voltage component of the incoming voltage with the positive voltage based on the zero-cross voltage.

### 9.3.11 PFC Voltage Control Function (HPFC_ControlVoltage_16k)

#### 9.3.11.1 Syntax

void HPFC_ControlVoltage_16k(pfc_t * const _pfc)

Arguments:
_pfc PFC control constructs
Return value:
None

#### 9.3.11.2 Processing Details

Calculate the amplified AC current command from the target DC.

© 2024
Toshiba Electronic Devices & Storage Corporation

# 10. Constant Definition Description

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

### 10.1.1 Motor Control Channel Selection

| | |
|---|---|
| **__CONTROL_MOTOR_FAN** | Define this when controlling the fan motor (CH0). |
| **__CONTROL_MOTOR_COMP** | Define this when controlling the compressor motor (CH1). |
| **__CONTROL_PFC** | Define this when controlling PFC(CH2). |

### 10.1.2 DAC Output Selection

| | |
|---|---|
| **__USE_DAC** | Define this bit when performing DAC control for evaluating variable-value output. |

### 10.1.3 RAMScope(NBD) Output Selection

| | |
|---|---|
| **__USE_NBD** | Define this when using RAMScope. |

### 10.1.4 Command Speed Unit Selection

| | |
|---|---|
| **__TGTSPD_UNIT** | The command speed unit can be selected.<br>0: Hz of Electrical angle<br>1: RPS of Mechanical angle<br>2: RPM of Mechanical angle |

### 10.1.5 Motor Control and PFC ON/OFF Selection

| | |
|---|---|
| **__USE_mikroE_COMMUNICATION_FUNC** | Motor-controlled and PFC ON/OFF can be selected.<br>None: Control by rewriting RAM from the live watch<br>Yes: Control by UART communication |

### 10.1.6 Common Parameter 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 factor.<br>Setting a larger value will stabilize but will delay the response. |
| cVDQ_AVE | Filter factor for Supply voltage Vdc, Output voltage Vdq. |
| | -- |
| | Set the filter factor.<br>Setting a larger value will stabilize but will delay the response. |

## 10.2 Motor Driver Setting Parameter: Motor Channel (D_Para_x.h) (x = Fan,Comp)

It is possible to drive various motors by changing the argument: in the motor driver section.

© 2024
Toshiba Electronic Devices & Storage Corporation

### 10.2.1 Control Selection

#### 10.2.1.1 AMP Selection

**__USE_INAMP** When using the built-in AMP, define it.

### 10.2.2 Argument 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.<br>Set the logic of the upper-phase driver. |
| cPOLL | Lower driver logic setting |
| | 0: Low active/ 1: High active |
| | Change the value according to the board design.<br>Set the logic of the bottom phase driver. |
| cV_MAX | Maximum voltage value setting |
| | Unit [V] |
| | Change the value according to the board design.<br>Set the power supply [V] corresponding to the full scale (4095) using the 12-bit count after ADC conversion. |
| cA_MAX | Maximum current value setting |
| | Unit [A] |
| | Change the value according to the board design.<br>Set the phase current [A] corresponding to the full scale (2047) with the 11-bit count after ADC conversion. |
| cSHUNT_TYPE | Setting of current acquisition type (3 shunt or 1 shunt) |
| | 1: 1 shunt/3: 3 shunt (not supported) |
| | Change the value according to the board design.<br>Set the current acquisition type. |
| cBOOT_TYPE | Setting of drive type at startup |
| | cBoot_i: Current Drive (not supported)/cBoot_v: Voltage Drive |
| | Set the drive type at start. Select the voltage type drive when current cannot be obtained, such as when starting the 1 shunt drive. |
| cSHUNT_ZERO_OFFSET | Setting the offset voltage |
| | Unit [V] |
| | Set the shunt voltage when no current is flowing.<br>This value is used for the initial value of the zero current average value. |
| cADCH_CURRENT_IDC | Current acquisition ADC channels setting (for 1-shunt) |
| | AINx |
| | Set ADC channels for detecting current. |
| cADCH_VDC | Power supply ADC channels setting |
| | AINx |
| | Set ADC channels for detecting the power supply voltage Vdc. |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | |
|---|---|
| cOVC | Setting the overcurrent detection value |
| | Unit [A] |
| | Set the overcurrent value.<br>If a coil current larger than this setting is detected, the output is turned OFF by software. |
| cVDC_MINLIM | Setting the Minimum Supply-voltage Vdc |
| | Unit [V] |
| | Set the lowest line-voltage Vdc.<br>When a voltage value less than this value is detected, the motor is stopped. |
| cVDC_MAXLIM | Setting the Maximum Supply-voltage Vdc |
| | Unit [V] |
| | Set the maximum line-voltage Vdc.<br>If a voltage value greater than this value is detected, the motor stops. |
| cPWMPRD | Setting PWM cycle |
| | Unit [us] Resolution 25 ns@80 MHz |
| | Set PWM carrier cycle. |
| cDEADTIME | Setting the dead time value |
| | Unit [us] Resolution 0.1 μs@80 MHz |
| | Set the dead time value. |
| cREPTIME | Setting of software control skipping count |
| | Unit [times] 1 to 15 |
| | The timing of vector calculation software processing can be skipped.<br>When this bit is set to 1, vector calculation is performed by software for each PWM1 cycle.<br>When this bit is set to 2, vector calculation is performed by software once in PWM2 cycle. |
| cID_ST_USER_ACT | Setting of d-axis start current |
| | Unit [A] |
| | Set the d-axis start current value.<br>The current value of this value is used for positioning and forced commutation.<br>Set a value about 10% of the rated commutation.<br>If the motor does not move, increase the value gradually until it moves. |
| cIQ_ST_USER_ACT | Setting of q-axis start current |
| | Unit [A] |
| | Set the value of q-axis start current.<br>Set a value about 1/2 of the d-axis starting current.<br>If the motor accelerates suddenly during transition from forced commutation (d-axis control) to steady (q-axis control), decrease the value. If it 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 |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | |
|---|---|
| | 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 the number of motor poles |
| | Unit [pole] |
| | Set the number of motor poles. |
| cID_KP | d-axis current control proportional gain |
| | Unit [V/A] |
| | Set the d-axis current control proportional gain. |
| cID_KI | d-axis current control integral gain |
| | Unit [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 |
| | Unit [V/As] |
| | Set the q-axis current control integral gain. |
| cPOSITION_KP | Position estimation proportional gain |
| | Unit [Hz/V] |
| | Set the proportional gain for position estimation. |
| cPOSITION_KI | Position estimation integral gain |
| | Unit [Hz/Vs] |
| | Set the integral gain for position estimation. |
| cSPEED_KP | Speed control proportional gain |
| | Unit [A/Hz] |
| | Set the proportional gain of speed control. |
| cSPEED_KI | Speed control integral gain |
| | Unit [A/Hzs] |
| | Set the integral gain of speed control. |
| cSPD_PI_PRD | Speed PI control cycle setting |
| | [times] |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | |
|---|---|
| | Set the speed PI control period.<br>When 1 is set, the speed PI is calculated for each PWM1 cycle.<br>When 2 is set, the speed PI is calculated once in PWM2 cycle. |
| cFCD_UD_LIM | Drive speed acceleration rate (at forced commutation) |
| | Unit [Hz/s] |
| | Set the speed acceleration rate for forced commutation.<br>If the motor rotation does not follow the forced commutation output, decrease the value. |
| cSTD_UP_LIM | Drive speed acceleration rate (steady stage) |
| | Unit [Hz/s] |
| | Set the speed acceleration rate at steady stage. |
| cSTD_DW_LIM | Drive speed deceleration rate (at steady stage) |
| | Unit [Hz/s] |
| | Set the steady stage speed deceleration rate. |
| cBOOT_LEN | Bootstrap waveform output time |
| | Unit [s] Resolution: 1 ms |
| | Set the bootstrap waveform output time. |
| 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 | Wait 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.<br>Set a value that is approximately 20% greater than the maximum frequency used for control from 10.<br>The smaller the value, the more accurate the operation will be. However, the control will be broken when the detected value exceeds this value. |
| cHZ_MIN | Switching speed from forced commutation to steady |
| | Unit [Hz] |
| | Set the speed to shift from forced commutation to steady.<br>Set the speed at which position estimation can be performed (induced voltage can be detected). |
| cHZ_SPWM | Shift PWM Switching Rate (for 1-shunt, 2-phase modulations) |
| | Unit [Hz] |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | |
|---|---|
| | Set the switching rate from the shift PWM to the normal PWM.<br>Switched by the value of the command speed. |
| cMINPLS | Minimum pulse setting (for 1 shunt) |
| | Unit [us] |
| | 1 Set PWM pulse-width times when current cannot be obtained during shunt driving. (* Not evaluated during shift PWM operation. When using shift PWM function, please apply starting by voltage-type, etc.)<br>When PWM pulse-width becomes a value less than this setting value, the detected current value is not used, and control is performed using the previous 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 with the electrical angle. |
| cVD_POS | Positioning output voltage during voltage drive |
| | Unit [V] |
| | Set the voltage during positioning.<br>Valid only when cBOOT_TYPE is "cBoot_v"<br>For more information, see section 10.2.3.2 Constant Value at Voltage-Type Startup. |
| cSPD_COEF | Forced commutation output voltage coefficient at voltage drive |
| | Set the value 0 < x < 1. |
| | Determines the output voltage during forced commutation.<br>The output voltage is the value obtained by multiplying this set value by the command speed.<br>Vd = cSPD_COEF × Omega_com<br>Valid only when cBOOT_TYPE is "cBoot_v"<br>For more information, see section 10.2.3.2 Constant Value at Voltage-Type Startup. |
| cHZ_V2I | Switching speed from voltage control to current control |
| | Unit [Hz] |
| | Set the speed to switch from voltage control to current control.<br>If a value larger than cHZ_MIN is set, it will shift to steady stage when the value becomes cHZ_MIN or more, and then switch to current control.<br>Valid only when cBOOT_TYPE is "cBoot_v" |
| cPFC_ON_CUR | PFC control starting current setting |
| | Unit [A] |
| | Set the current at which PFC control starts. |
| cOPEN | Temperature sensor open judgment value |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | |
|---|---|
| | ADC value |
| | Set the open judgment value of the temperature sensor. |
| cSHORTC | Temperature sensor short judgment value |
| | ADC value |
| | Set the short-circuit judgment value of the temperature sensor. |
| cOPEN_SHORT_CNT | Temperature sensor open/short decision count |
| | Unit [times] |
| | Set the number of times to confirm the temperature sensor open/short status. |
| cTEMP_CNT | Temperature error establishment count |
| | Unit [times] |
| | Set the number of times to determine the error of the abnormal temperature condition. |
| cTIMES | Temperature information resolution setting |
| | Resolution |
| | Set the temperature information resolution.<br>Example :10→0.1℃ |
| cEXH_OVER_EMG | Exhaust error judgment temp. |
| | Unit [℃] |
| | Set the temp. error threshold of Exhaust. |
| cHVMOS_OVER_EMG | HVMOS error judgment temp. |
| | Unit [℃] |
| | Set the temp. error threshold of HVMOS. |
| cIGBT_OVER_EMG | IGBT error judgment temp. |
| | Unit [℃] |
| | Set the temp. error threshold of IGBT. |
| cDIODE_OVER_EMG | Diode error judgment temp. |
| | Unit [℃] |
| | Set the temp. error threshold of Diode. |
| cOUTPIPE_OVER_EMG | Outpipe error judgment temp. |
| | Unit [℃] |
| | Set the temp. error threshold of Outpipe. |
| cTEMP_UP1 | Maximum sensor temperature |
| | Unit [℃] |
| | Set the maximum temperature value of the sensor. |
| cTEMP_DW1 | Sensor minimum temperature |
| | Unit [℃] |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | |
|---|---|
| | Set the minimum temperature value of the sensor. |
| __FIXED_VDC | Power supply voltage fixed setting |
| | 0:Detection 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 voltage.<br>Valid only when __FIXED_VDC is "1" |

© 2024
Toshiba Electronic Devices & Storage Corporation

### 10.2.3 Relationship Between Constant Setting Value and Waveform

#### 10.2.3.1 Constant Value at Current Type Startup

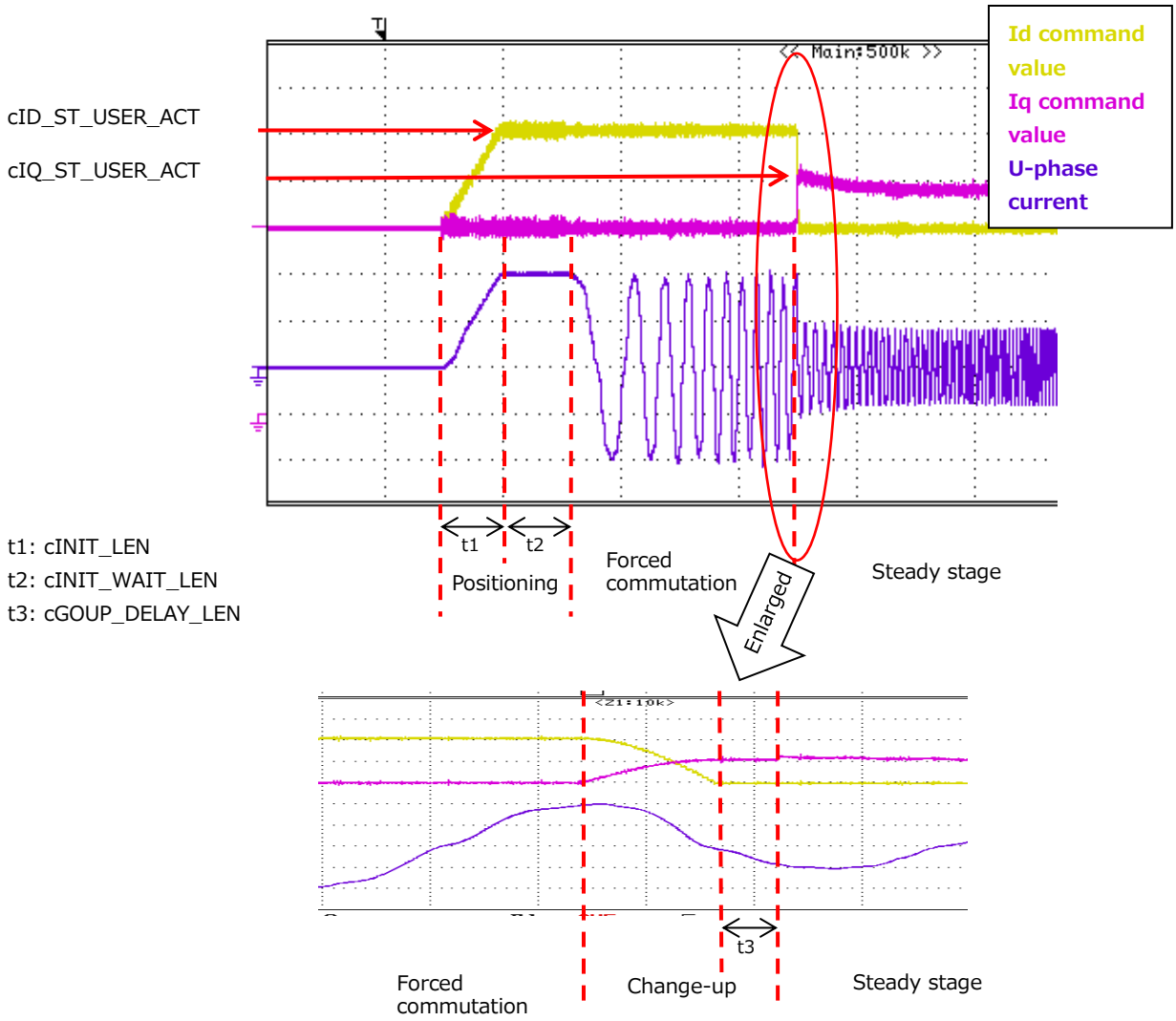



**Fig. 19 Starting Current Waveform (Positioning at an Electrical Angle of 0°)**

The cID_ST_USER_ACT and cIQ_ST_USER_ACT are updated at the change-up stage.
After switching, control is performed with the cGOUP_DELAY_LEN time period Iq command value constant value.
After steady stage transition, Iq command is calculated by PI control.

© 2024
Toshiba Electronic Devices & Storage Corporation

10.2.3.2 **Constant Value at Voltage Type Startup**

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

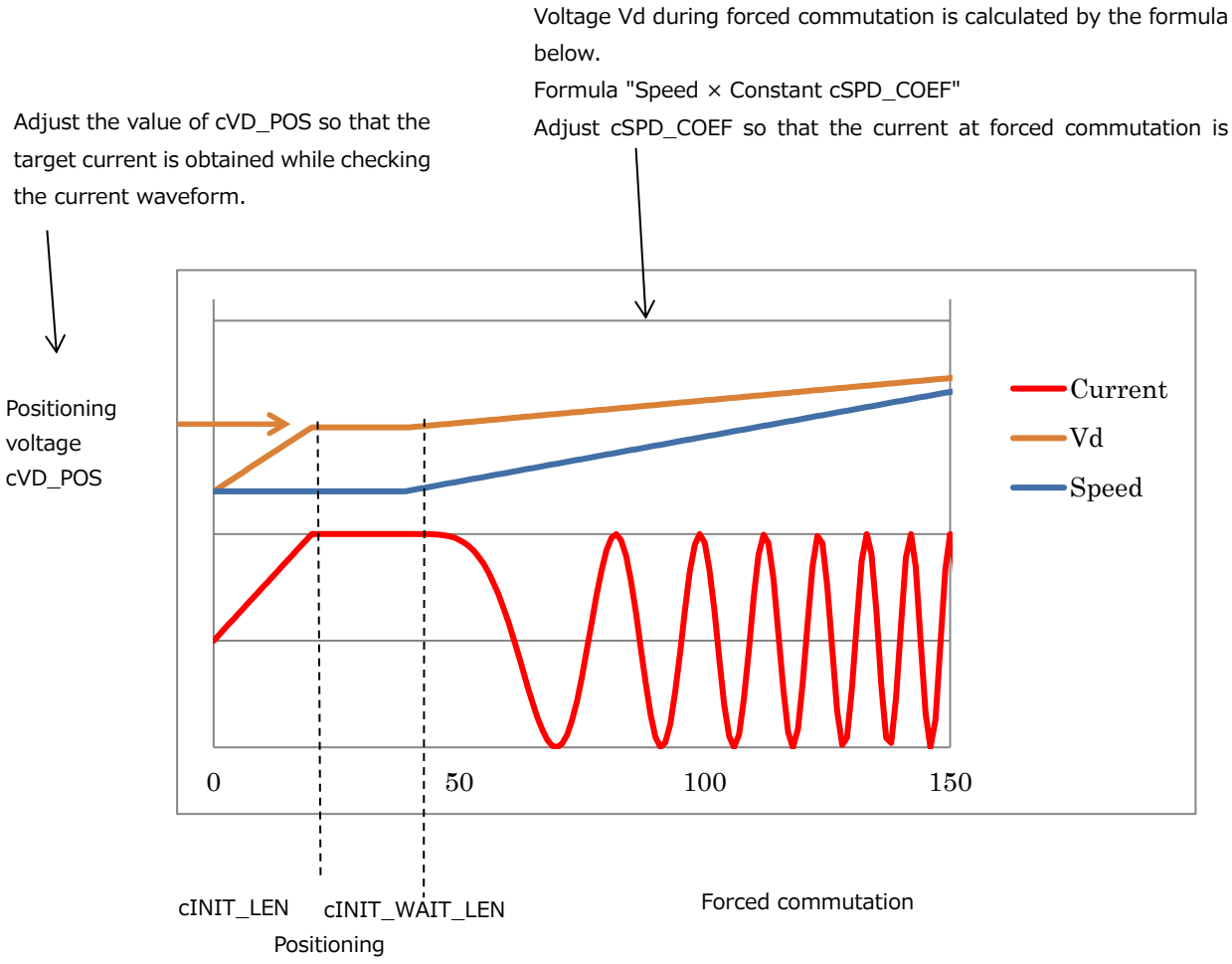



**Fig. 20 Argument for Voltage Driving: Adjustment Method**

© 2024
Toshiba Electronic Devices & Storage Corporation

### 10.3 User Control Related Constants

**Usercon.c**

```
 #define c2000MS_TIMER    (2000)  /* [ms] (4 kHz * 4) * 2000 */
      Communication judgment 2s timer

 /* UART Setting */
 #define UART_ch          UART0            /* UART Channel */
          UART channels: CH0
#define INTERRUPT_TX  INTSC0TX_IRQn          /* UART Interrupt request */
       UART0 transmit interrupt requested
#define INTERRUPT_RX  INTSC0RX_IRQn          /* UART Interrupt request */
       UART0 receive interrupt requested
#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: Wait for completion of reception
#define cUART_ERR            (0x01)       /* UART mode : error */
       UART: Communication failure
#define cREQ_SYSTEM_START        (0x10)       /* System start request */
       System start request command
#define cREQ_ROTATE_MOTOR        (0x11)       /* Target speed update request */
       Target speed update request command
#define cREQ_CHANGE_MOTOR        (0x12)       /* Control Rpm request */
       Motor Switch Request Command
#define  cREQ_ON_OFF_PFC          (0x13)       /* Control PFC request */
       PFC On/Off requestcommand
#define  cREQ_ALL_STOP_MOTOR    (0x14)       /* Stop PFC request */
       Motor/stop PFC command
#define  cREQ_STATUS_CH           (0x15)       /* Change channel request */
       Status display change request command
#define  cREQ_STATUS_DAC          (0x16)       /* Chnage Dac mode request */
       Status display request command
#define cGET_MOTOR_ENABLE        (0x80)       /* Operating status */
       Obtain motor-operation EN/DI (*Not used in this system)
#define cGET_STATE_EMG           (0x81)       /* Emergency status */
       Get EMG Status
#define cGET_STAGE                    (0x82)       /* Main stage */
       Main Stage Acquisition Command
#define cGET_CONTROL_CH          (0x83)       /* Control channel */
       Obtain motor-control CH command
#define cGET_CARRIER_FREQUENCY     (0x84)       /* Carrier frequency */
       Carrier frequency acquisition command
#define cGET_MOTOR_SPEED_MIN   (0x85)       /* Minimum rotation speed */
       Minimum rotation speed acquisition command
#define cGET_MOTOR_SPEED_MAX  (0x86)       /* Maximum rotation speed */
       Maximum rotation speed acquisition command
#define cGET_DEAD_TIME           (0x87)       /* Dead time */
       Dead time acquisition command
#define cGET_GATE_ACTIVE         (0x88)       /* Gate active */
       Gate Active Acquire Command
#define cGET_POSITION_DITECT     (0x89)       /* Position detect */
       Shunt type acquisition command
```

© 2024
Toshiba Electronic Devices & Storage Corporation

```
#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 Acquisition Commands
#define cGET_INTERNAL_AMP           (0x90)      /* Internal amp */
        Built-in amplifier acquisition command
#define cGET_DIRECTION              (0x91)      /* Direction */
        Rotation direction acquisition command
#define cGET_MODULATION             (0x92)      /* Modulation */
        Get Modulation Command
#define cGET_MOTOR_SPEED            (0x94)      /* motor rotation speed */
        Motor speed acquisition command
#define  cGET_OUTPIPE_TEMP          (0x95)      /* outpipe templeture */
        Pipe temperature acquisition command
#define  cGET_EXHAUST_TEMP          (0x96)      /* exhaust templeture */
        Heat emission temperature acquisition command
#define  cGET_DIODE_TEMP            (0x97)      /* diode templeture */
        PFC Diode-Temperature acquisition command
#define  cGET_IGBT_TEMP             (0x98)      /* IGBT templeture */
        PFC IGBT Temp. acquisition command
#define  cGET_HVMOS_TEMP            (0x99)      /* HVMOS templeture */
        MOSFET Temp. 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: Current detect error
#define cEMG_STATE_EMG_DC           (0x03)      /* emergency status : over vdc */
        EMG state: VDC error
#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 (*Not used in this system)
#define cPOSITION_DETECT_1SHUNT     (1)     /* Position Ditect : 1shunt */
        Shunt type: 1 shunt (*Not used in this system)
#define cDAC_DATA_TMPREG0           (0x00)      /* Dac data : TMPREG0 */
        DAC: U-phase current
#define cDAC_DATA_TMPREG1           (0x01)      /* Dac data : TMPREG1 */
        DAC: V-phase current
```

© 2024
Toshiba Electronic Devices & Storage Corporation

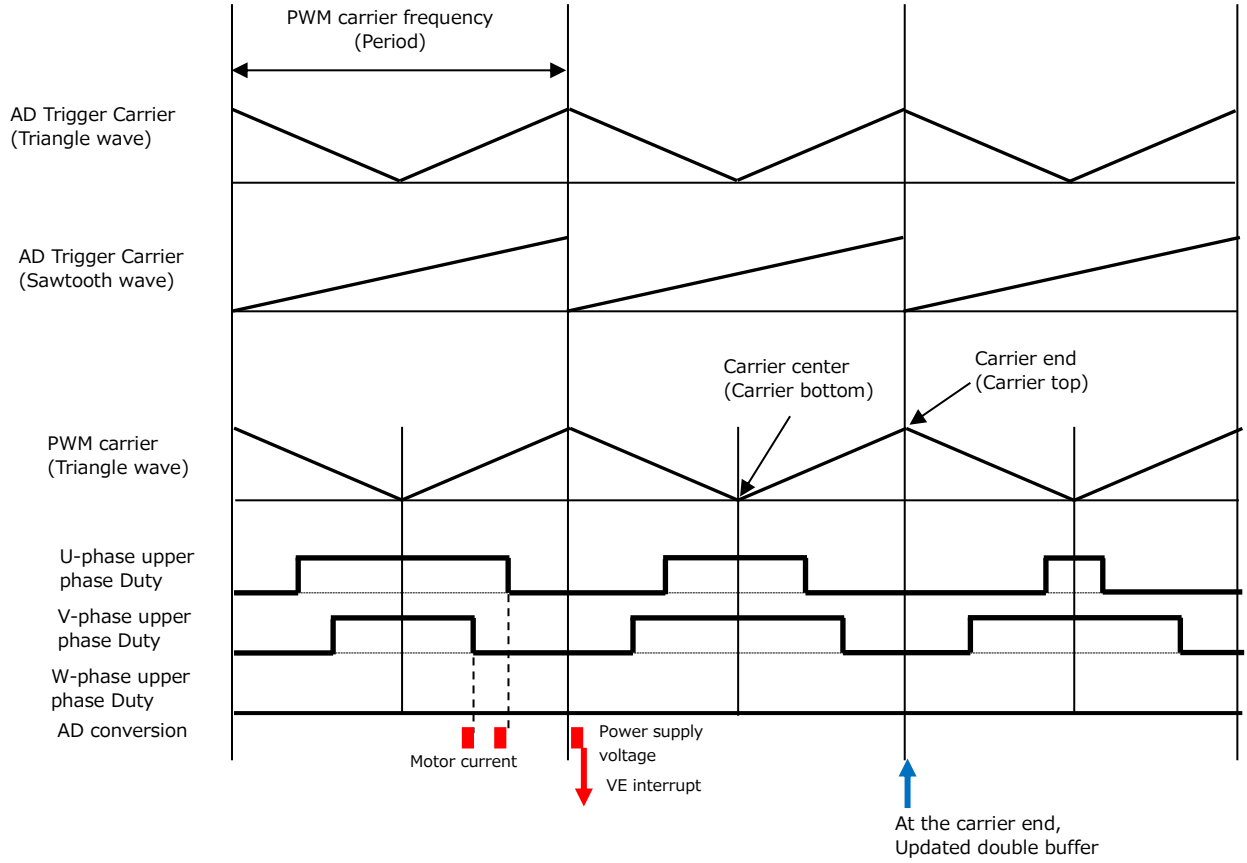
```
#define cDAC_DATA_TMPREG2       (0x02)      /* Dac data : TMPREG2 */
        DAC: W-phase current
#define cDAC_DATA_THETAHALF     (0x03)      /* Dac data : theta.half[1] */
        DAC: Electric angle
#define cDAC_DATA_IDREF         (0x04)      /* Dac data : Id_ref */
        DAC: Id Reference (Target)
#define cDAC_DATA_ID            (0x05)      /* Dac data : Id */
        DAC: Id (present value)
#define cDAC_DATA_IQREF         (0x06)      /* Dac data : Iq_ref */
        DAC: Iq Reference (Target)
#define cDAC_DATA_IQ            (0x07)      /* Dac data : Iq */
        DAC: Iq (present value)
#define cDAC_DATA_OMEGACOMHALH  (0x08)      /* Dac data : omega_com.half[1] */
        DAC: Angular speed (target value)
#define cDAC_DATA_OMEGAHALF     (0x09)      /* Dac data : omega.half[1] */
        DAC: Angular speed (present value)
#define cDAC_DATA_OMEGADEV      (0x0A)      /* Dac data : omega_dev */
        DAC: Angular speed (differential)
#define cINTERNAL_AMP_NO        (0)         /* Internal amp : External */
        Internal amplifier: Not used
#define cINTERNAL_AMP_YES       (1)         /* Internal amp : Internal */
        Built-in amplifier: Used
#define cDIRECTION_CW           (0)         /* Direction : plus */
        Rotation-direction: CW
#define cDIRECTION_CCW          (1)         /* Direction : minus */
        Rotational direction: CCW (*Not used in this system)
#define  cMOTOR_CTRL_FAN        (0)         /* motor control : fan */
        Motor Definition: Fan Motor
#define  cMOTOR_CTRL_COMP       (1)         /* motor control : compressor */
        Motor Definition: Conpressor Motor
#define  cMOTOR_CTRL_PFC        (2)         /* motor control : pfc */
        PFC Definition: PFC
```

© 2024
Toshiba Electronic Devices & Storage Corporation

# 11. Control and Data Update Timing

## 11.1 Vector Control Using VE

### 11.1.1 1 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 end |
| Power setting (MDOUT) | Carrier end |

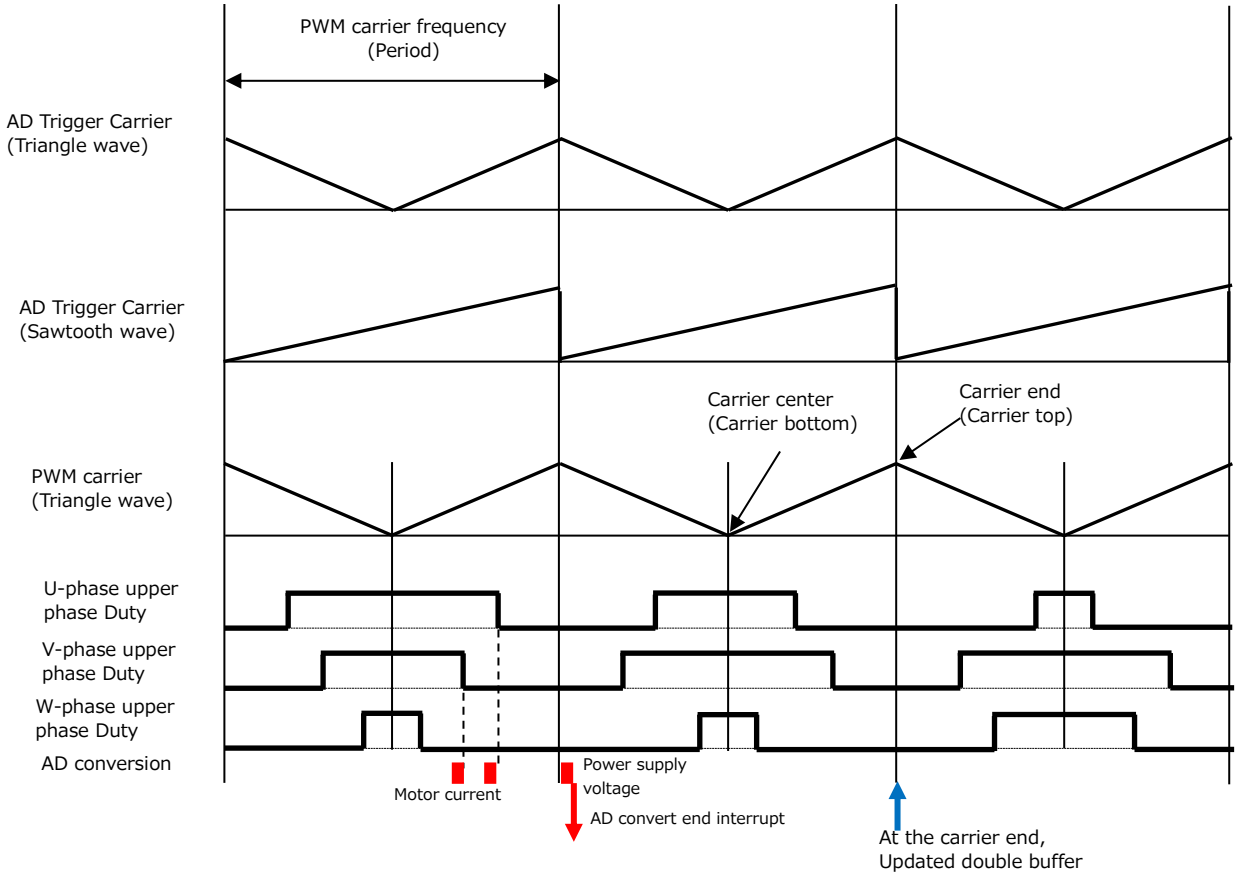
#### 11.1.1.3 **Interrupt**

| Interrupt Request | Interrupt | Timing |
|---|---|---|
| VE | Allowed | After VE entry is completed |
| PWM(PMD) | Prohibited | Once, twice, every four times |
| ADC conversion finish | Prohibited | After completion of ADC conversion of the power supply |
| ADC trigger | — | Same frequency as PWM |

© 2024
Toshiba Electronic Devices & Storage Corporation

## 11.2 Vector Control by Software

### 11.2.1 1 Shunt Control




#### 11.2.1.1 Carrier Waveform

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

#### 11.2.1.2 Double Buffer Update Timing

| Data | Update Timing |
|---|---|
| PWM period (RATE) | Carrier end |
| Duty (CMPU,CMPV,CMPW) | Carrier end |
| Trigger position (TRGCMPx) | Carrier end |
| Output setting (MDOUT) | Carrier end |

#### 11.2.1.3 Interrupt

| Interrupt Request | Interrupt | Timing |
|---|---|---|
| PWM(PMD) | Prohibited | Once, twice, every four times |
| ADC conversion finish | Allowed | After completion of ADC conversion of the power supply |
| ADC trigger | — | Same frequency as PWM |

© 2024
Toshiba Electronic Devices & Storage Corporation

# 12. Peripheral Driver

## 12.1 MCU Peripheral Circuit Address

This bit defines the base address of the MCU peripheral register 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 **Select VEx:VE adress.**
TSB_PMD_TypeDef* const **Select PMDx:PMD adress.**
TSB_AD_TypeDef* const **Select ADx:AD adress.**

## 12.2 Vector Engine (VE)

### 12.2.1 Function Specifications

#### 12.2.1.1 **IP_VE_init**

VE initial settings

**API:**

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

**Arguments:**

**Select VEx:VE adress.**

**_initdata:** VE initialization data structure

For more information, see section 12.2.2.1 VE_InitTypeDef.

**Function:**

Performs the initial settings for VE.

**Supplement:**

Call by stopping VE and disabling interrupt.

**Return value:**

None

#### 12.2.1.2 **VE_Start**

VE Start

**API:**

VE_Start(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

**Function:**

Starts VE.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.3 **VE_GetPhaseCurrent**

Acquisition of phase current

**API:**

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

© 2024
Toshiba Electronic Devices & Storage Corporation

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

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

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

**Function:**

Obtains the current value of each phase.

The U-phase current value enters a-phase, the V-phase current value enters b-phase, and the W-phase current value enters c-phase.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.4 **VE_GetCurrentAdcData**

Acquire current AD

**API:**

void

VE_GetCurrentAdcData(const ipdrv_t* const _ipdrv,

Uint32_t* _adc_ia, uint32_t* _adc_ib, uint32_t* _adc_ic)

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

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

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

**Function:**

Gets the value of IAADCx, IBADCx, ICADCx register to the ADC value of current of each phase.

**Supplement:**

None

**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)

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

**_motor:** Selects the structure address of the vector control variable.

**Function:**

Stores the values of the motor power supply voltage Vdc, d-axis voltage Vd, q-axis voltage Vq, d-axis current Id, and q-axis current Iq from VE register into the vector control variables.

At the timing when the current cannot be detected due to Duty width, the d-axis current Id and q-axis current Iq values are written to VE register.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.6 **VE_GetPWM_DutyU**

Obtains the U-phase Duty.

**API:**

Uint32_t VE_GetPWM_DutyU(const ipdrv_t* const _ipdrv)

**Arguments:**

© 2024
Toshiba Electronic Devices & Storage Corporation

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**Function:**

Gets the U-phase Duty register.

**Supplement:**

None

**Return value:**

U-phase Duty

#### 12.2.1.7 **VE_GetPWM_DutyV**

V-phase Duty reading

**API:**

Uint32_t VE_GetPWM_DutyV(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**Function:**

Gets the V-phase Duty register.

**Supplement:**

None

**Return value:**

V-phase Duty

#### 12.2.1.8 **VE_GetPWM_DutyW**

Obtains the W-phase Duty.

**API:**

Uint32_t VE_GetPWM_DutyW(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**Function:**

Gets the W-phase Duty register.

**Supplement:**

None

**Return value:**

W-phase Duty

#### 12.2.1.9 **VE_GetPWM_DutyMed**

Acquire Duty intermediate value

**API:**

Uint32_t VE_GetPWM_DutyMed(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**Function:**

Used to acquire the middle value of the U-, V-, or W-phase Duty value.

**Supplement:**

None

**Return value:**

Duty intermediate value

#### 12.2.1.10 **VE_GetPWM_Modulation**

Modulation type acquisition

**API:**

Int VE_GetPWM_Modulation(const ipdrv_t* const _ipdrv)

**Arguments:**

© 2024
Toshiba Electronic Devices & Storage Corporation

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**Function:**

Used to acquire the modulation type.

**Supplement:**

None

**Return value:**

Modulation type 0: 3-phase modulation, 1: 2-phase modulation

#### 12.2.1.11 **VE_GetSector**

Current sector acquisition

**API:**

Uint32_t VE_GetSector(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**Function:**

Gets the current sector value.

**Supplement:**

None

**Return value:**

Current sector value [0-11]

#### 12.2.1.12 **VE_GetShiftPWMState**

Acquire shifted PWM status

**API:**

Int VE_GetShiftPWMState(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**Function:**

Gets the shifted PWM status.

**Supplement:**

None

**Return value:**

Shift PWM status 0: Normal PWM, 1: Shift PWM

#### 12.2.1.13 **VE_GetOutPutMode**

Acquire PWM status

**API:**

Int VE_GetOutputMode(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

**Function:**

Used to acquire PWM output status.

**Supplement:**

None

**Return value:**

PWM condition OCRMD_OUT_OFF: Output OFF
OCRMD_OUT_ON: Output ON
OCRMD_OUT_ON_LOWPH: only lower phase is ON

#### 12.2.1.14 **VE_SetdataToVEreg_Stop**

Data set to VE register (Stop)

**API:**

© 2024
Toshiba Electronic Devices & Storage Corporation

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

**Function:**

This command sets VE register in the stopped status.

Initializes the current control gain integration value register, etc.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.15 **VE_SetdataToVEreg_Bootstrap**

Data set to VE register (Bootstrap)

**API:**

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

**Function:**

This command sets VE register in bootstrap status.

In VE, two-phase modulations and an output voltage of 0 result in output of a wave that turns ON only the L-side.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.16 **VE_SetdataToVEreg_Initposition_i**

Data set to VE register (Initposition current control)

**API:**

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

**Function:**

Performs the setting process of the current control positioning status to VE register.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.17 **VE_SetdataToVEreg_Initposition_v**

Data set to VE register (Initposition voltage-controlled)

**API:**

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

**Arguments:**

© 2024
Toshiba Electronic Devices & Storage Corporation

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

**Function:**

Settings to the voltage-controlled positioning status VE register are processed.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.18 **VE_SetdataToVEreg_Force_i**

Data set to VE register (forced commutation current control)

**API:**

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

**Function:**

Performs the setting process of the current control type forced commutation status to VE register.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.19 **VE_SetdataToVEreg_Force_v**

Data set to VE register (forced commutation voltage-controlled)

**API:**

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

**Function:**

Performs the setting process of the voltage-controlled forced commutation status to VE register.

**Supplement:**

None

**Return value:**

None

#### 12.2.1.20 **VE_SetdataToVEreg_Change_up**

Data set to VE register (change-up)

**API:**

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

**Function:**

Setting of change-up status to VE register is performed.

**Supplement:**

© 2024
Toshiba Electronic Devices & Storage Corporation

None

**Return value:**

None

12.2.1.21 **VE_SetdataToVEreg_Steady_A**

Data set to VE register (steady)

**API:**

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

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

**Function:**

Steady stage VE register settings are processed.

**Supplement:**

None

**Return value:**

None

12.2.1.22 **VE_SetdataToVEreg_Emergency**

Data set to VE register (EMG)

**API:**

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

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

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

**Function:**

Set Emergency status to VE register.

**Supplement:**

None

**Return value:**

None

12.2.1.23 **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)

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**_z_ia:** Sets the a-phase zero-current ADC value.

**_z_ib:** Sets the b-phase zero-current ADC value.

**_z_ic:** Sets the c-phase zero-current ADC value.

**Function:**

Sets ADC at zero-current to VE register.

**Supplement:**

None

**Return value:**

None

© 2024
Toshiba Electronic Devices & Storage Corporation

12.2.1.24 **VE_SetVDCreg**

Setting DC Linkvoltage Vdc

**API:**

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

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**_dat:** Sets the power supply voltage of the motor.

**Function:**

Sets the power supply to VE register.

**Supplement:**

None

**Return value:**

None

12.2.1.25 **VE_SetSpwmMode**

Shift PWM setting

**API:**

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

**Arguments:**

**_ipdrv:** Specifies the structure for which the MCU peripheral circuit address is defined.

**_dat:** Sets the shift PWM mode.

**Function:**

Set the shift PWM mode status to VE register.

**Supplement:**

None

**Return value:**

None

12.2.1.26 **VE_SetModulType**

Modulation type setting

**API:**

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

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

**_dat:** Sets the modulation type.

**Function:**

Set the modulating type to VE register.

**Supplement:**

None

**Return value:**

None

### 12.2.2 Data Structure

12.2.2.1 **VE_InitTypeDef**

**Data Fields:**

| | |
|---|---|
| Uint8_t | Shunt: Shunt type |
| | 1:1 shunt |
| Uint16_t | Reptime: Number of repeats (1 to 15) |
| Uint16_t | Trgmode: Activation trigger |
| | TRGMODE_UNITA: Activated by ADCA PMD0 trigger-synchronous conversion end interrupt |
| | TRGMODE_UNITB: Activated by ADCB PMD1 trigger-synchronous conversion end interrupt |

© 2024
Toshiba Electronic Devices & Storage Corporation

Uint16_t Tpwm:PWM period (for phase interpolating)
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
Uint16_t **Trgcomp0: Trigger timing 0 setting**
Value to set to VExTRGCOMP0
Uint16_t **Trgcomp1: Trigger Timing 1 Setting**
Value to set to VExTRGCOMP1
Uint32_t **Fmode: Flow control setting**
Value to set to VExFMODE
Uint16_t **Fpwmchg: PWM switching rate setting**
Value to set to VExFPWMCHG

## 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)

**Arguments:**

**PMDx:** select PMD adress.
**_initdata:** PMD initialization data structure
For more information, see section 12.3.2.1 PMD_InitTypeDef.

**Function:**

Performs PMD default settings.

**Supplement:**

Call with PMD stopped and interrupt disabled.

**Return value:**

None

#### 12.3.1.2 **PMD_GetEMG_Status**

Obtaining EMG protective status

**API:**

Emg_status_e PMD_GetEMG_Status(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

**Function:**

Used to acquire EMG protective status.

**Supplement:**

None

**Return value:**

**Emg_status_e:** EMG protected
**cNormal:** Normal
**cEMGProtected:** Disables PWM output according to EMG generation

#### 12.3.1.3 **PMD_ReleaseEMG_Protection**

Release of EMG protective status

**API:**

© 2024
Toshiba Electronic Devices & Storage Corporation

void PMD_ReleaseEMG_Protection(const ipdrv_t* const _ipdrv)

**Arguments:**

**_ipdrv:** Specifies a structure for which the MCU peripheral circuit address is defined.

**Function:**

Reset EMG protective status.

**Supplement:**

Calling this function does not release the protected state unless MDOUT is 0 and EMG port is H.

**Return value:**

None

### 12.3.2 Data Structure

#### 12.3.2.1 **PMD_InitTypeDef**

**Data Fields:**

Uint8_t **Shunt:** Shunt type
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 **Pwmrate:** PWM frequency
Value to set to PMDxRATE

Uint16_t **Deadtime:** Dead time
Value to set to PMDxDTR

Uint8_t **Busmode:** Mode selection
Value to set to PMDxMODESEL

## 12.4 Analog to Digital Converter (ADC)

### 12.4.1 Function Specifications

#### 12.4.1.1 **IP_ADC_init**

ADC default setting

**API:**

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

**Arguments:**

**Select ADx:ADC adress.**

**_initdata:** ADC initialization data structure

For more information, see section 12.4.2.1 AD_InitTypeDef.

**Function:**

Performs the initialization for ADC.

**Supplement:**

Call by stopping ADC and disabling interrupt.

**Return value:**

None

### 12.4.2 Data Structure

#### 12.4.2.1 **AD_InitTypeDef**

Data Fields:

Uint8_t Shunt: Shunt type
1:1 shunt

© 2024
Toshiba Electronic Devices & Storage Corporation

| | |
|---|---|
| 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 Acquisition ADC Channel Number (for 1-shunt) |
| Uint8_t | Vdcch: Motor supply voltage Vdc acquisition ADC channel Number |
| Uint8_t | Pmd_ch: Select PMD channels to be used |
| | cPMD0: PMD channel0 |
| | cPMD1: PMD-channel 1 |
| | cPMD2: PMD-channel 2 |
| Uint8_t | Pints: PMD Triggering Interrupt Select |
| | cPINTS_A: ITTADxPDA |
| | cPINTS_B: ITTADxPDB |
| | cPINTS_C: ITTADxPDC |
| | cPINTS_D: ITTADxPDD |
| Uint32_t** | **p_adreg0:** ADC conversion result storage register address 0 |
| Uint32_t** | **p_adreg1:** ADC conversion result storage register address 1 |
| Uint32_t** | **p_adreg2:** ADC conversion result storage register address 2 |
| Uint32_t** | **p_adreg3:** ADC conversion result storage register address 3 |

© 2024
Toshiba Electronic Devices & Storage Corporation

## 12.5 Constant Description

### 12.5.1 Constant for A-VE + Mounted MCU (mcuip_drv.h)

#### 12.5.1.1 A-PMD

For setting MDCR register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|
| cWPWMMD | W-phase PWM Carrier waveform | PWMMD_SAWTOOTH | Sawtooth (edge PWM) |
| | | PWMMD_TRIANGULAR | Triangle wave (center PWM) |
| | | PWMMD_SAWTOOTH_REV | Reverse sawtooth (edge PWM) |
| | | PWMMD_TRIANGULAR_REV | Inverse triangle wave (center PWM) |
| cVPWMMD | V-phase PWM Carrier waveform | PWMMD_SAWTOOTH | Sawtooth (edge PWM) |
| | | PWMMD_TRIANGULAR | Triangle wave (center PWM) |
| | | PWMMD_SAWTOOTH_REV | Reverse sawtooth (edge PWM) |
| | | PWMMD_TRIANGULAR_REV | Inverse triangle wave (center PWM) |
| cUPWMMD | U-phase PWM Carrier waveform | PWMMD_SAWTOOTH | Sawtooth (edge PWM) |
| | | PWMMD_TRIANGULAR | Triangle wave (center PWM) |
| | | PWMMD_SAWTOOTH_REV | Reverse sawtooth (edge PWM) |
| | | PWMMD_TRIANGULAR_REV | Inverse triangle wave (center PWM) |
| cDSYNCS | PWM duty Execution Buffer Update Timing of Compare Register | DSYNCS_INTCYC | Interrupt period setting |
| | | DSYNCS_CENTER | Renewed by phase-specific PWM center |
| | | DSYNCS_END | Updated at phase-specific PWM end |
| | | DSYNCS_CENTER_END | Updated at PWM end and center by phase |
| cDTCREN | Dead time Permission to amend | DTCREN_DISABLE | No correction |
| | | DTCREN_ENABLE | Permission to amend |
| cDCMEN | Run Buffer Update/Trigger Output Skipping Control | DCMEN_DISABLE | Prohibition |
| | | DCMEN_ENABLE | Permission |
| cPWMSYNT | Port output mode Setting | SYNTMD_0 | |
| | | SYNTMD_1 | |
| cPWMSPCFY | Duty mode selection | DTYMD_COMMON | Common for 3-phase |
| | | DTYMD_INDEPENDENT | 3-phase independent |
| cPWMINT | PWM Interrupt Request Timing Selection | PINT_CENTER | Interrupt generated at U-phase PWM center |
| | | PINT_END | Interrupt generated at U-phase PWM end |
| cPWMINTPRD | PWM Interrupt Request Period Selection | INTPRD_HALF | Interrupt requested for each PWM 0.5 cycle |
| | | INTPRD_1 | Interrupt requested for each PWM 1 cycle |
| | | INTPRD_2 | Interrupt requested for each PWM 2 cycle |
| | | INTPRD_4 | Interrupt requested for each PWM 4 cycle |

For setting MDPOT register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | |
|---|---|---|---|
| cSYNCS | MDOUT setting transfer timing selection | SYNCS_ASYNC | Asynchronous |
| | | SYNCS_INTENC | When a A-ENCx interrupt is generated |
| | | SYNCS_NTTB | When a general timer interrupt request is generated |
| | | SYNCS_CTRGO | When A-ENCx MCMP is met |
| cPSYNCS | Update MDOUT, MDCR run buffer Timing selection | PSYNCS_WRITE | PWM asynchronous (reflected when writing) |
| | | PSYNCS_CENTER | Phase-specific PWM center |
| | | PSYNCS_END | Phase-specific PWM end |
| | | PSYNCS_CENTER_END | Phase-specific PWM End and Center |

For setting PORTMD register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|
| cPORTMD | Port output control during debug halt | PORTMD_ALLHIZ | Upper phase high impedance/Lower phase high impedance |
| | | PORTMD_UHIZ_LON | Upper phase high impedance/lower phase PMD output |
| | | PORTMD_UON_LHIZ | Upper phase PMD output/lower phase high impedance |
| | | PORTMD_ALLON | Upper phase PMD output/lower phase PMD output |

For setting TRGCR register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|
| cTRGMD_3SHT | At 3 shunts Trigger Timing | TRGMD_DIS | Trigger output disabled |
| | | TRGMD_TRI_FIRST_HALF | Trigger when a compare is met in the first half of a PWM on a triangle-wave carrier |
| | | TRGMD_TRI_SECND_HALF | Trigger when a compare is met in the second half of a PWM on a triangle-wave carrier |
| | | TRGMD_TRI_F_S_CMP | Trigger output when the compare of the first half and the second half of PWM is met in triangle-wave carrier |
| | | TRGMD_END | Trigger out at PWM end timing |
| | | TRGMD_CENTER | Trigger out at PWM center timing |
| | | TRGMD_ENDCENTER | Trigger out at PWM end timing/center timing |
| | | TRGMD_DIS7 | Trigger output disabled |
| cTRGMD_1SHT | At 1 shunt Trigger Timing | TRGMD_DIS | Trigger output disabled |
| | | TRGMD_TRI_FIRST_HALF | Triggering when a compare is met in the first half of a PWM on a triangle-wave carrier |
| | | TRGMD_TRI_SECND_HALF | Triggering when a compare is met in the second half of a PWM on a triangle-wave carrier |
| | | TRGMD_TRI_F_S_CMP | Trigger output when the compare of the first half and the second half of PWM is |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | |
|---|---|---|---|
| | | | met in triangle-wave carrier |
| | | TRGMD_END | Trigger out at PWM end timing |
| | | TRGMD_CENTER | Trigger out at PWM center timing |
| | | TRGMD_ENDCENTER | Trigger out at PWM end timing/center timing |
| | | TRGMD_DIS7 | Trigger output disabled |
| cBUFSYNC | In the execution buffer Asynchronous Update Enable | TRGBE_SYNC | Synchronous update |
| | | TRGBE_ASYNC | Asynchronous update |
| cCARSEL | Selecting the comparison carrier | CARSEL_BASE | Comparison with the basic carrier |
| | | CARSEL_PHASE | Comparison with the Phase Carrier |

For setting TRGMD register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|
| cEMGTGE | Setting for enabling outputting during EMG protective operation | EMGTGE_DISABLE | Trigger output disabled during protective operation |
| | | EMGTGE_ENABLE | Trigger output enabled during protection operation |

For setting EMGCR register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|
| cEMGCNT | EMG detection-time | 0 to 31 | Removes noises from the error detection input (EMG)<br>Set time<br>cEMGCNT × 16/fsys<br>When "00000", the noise filter is set through |
| cINHEN | Enable/disable PMD during debug halt | INHEN_DISABLE | Disable |
| | | INHEN_ENABLE | Enable |
| cEMGMD | EMG protection-mode Select | EMGMD_ALLHIZ | All-phase high-impedance |
| | | EMGMD_UON_LHIZ | All upper phase on/all lower phase high impedance |
| | | EMGMD_UHIZ_LON | All upper phase high impedance/all lower phase on |
| | | EMGMD_ALLHIZ2 | All-phase high-impedance |

For setting OVVCR register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|
| cOVVCNT | OVV detection Time | 0 to 31 | Sets OVV denoising duration.<br>cOVVCNT × 16/fsys<br>When "00000", the noise filter is set Through |
| cADIN1EN | Enable inputting ADC monitoring function 1 | ADIN1EN_DISABLE | Disable input |
| | | ADIN1EN_ENABLE | Enable input |
| cADIN0EN | Enable inputting | ADIN0EN_DISABLE | Disable input |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | |
|---|---|---|---|
| | ADC monitoring function 0 | ADIN0EN_ENABLE | Enable input |
| cOVVMD | OVV protection-mode Select | OVVMD_NOCON | No output control |
| | | OVVMD_UON_LOFF | All upper phase on, all lower phase off |
| | | OVVMD_UOFF_LON | All upper phase off, all lower phase on |
| | | OVVMD_ALLOFF | All phase off |
| cOVVISEL | OVV terminal Control | OVVISEL_PORT | Port input enable |
| | | OVVISEL_ADC | ADC monitoring |

**12.5.1.2 A-VE+**

cTADC Set ADC conversion-time for one-shunt shift PWM.

For setting FMODE register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|
| cSPWMMD | PWM shift mode Select | SPWMMD_SPWM1 | Shift 1 |
| | | SPWMMD_SPWM2U | Shift 2 (U-phase center) |
| | | SPWMMD_SPWM2V | Shift 2 (V-phase center) |
| | | SPWMMD_SPWM2W | Shift 2 (W-phase center) |
| cCCVMD | Phase conversion mode selection | CCVMD_RELATIVE | Relative conversion |
| | | CCVMD_ABSOLUTE | Absolute conversion |
| cPHCVDIS | Phase conversion disabled | PHCVDIS_ENABLE | Enable 2-3 phase conversion (3-phase AC output) |
| | | PHCVDIS_DISABLE | Disable 2-3 Phase conversion (2-phase AC output) |
| cVSLIMMD | Voltage scalar limit Control | VSLIMMD_DIS | Scalar restriction prohibited |
| | | VSLIMMD_D | Limited in the d-axis direction |
| | | VSLIMMD_Q | Limited in the q-axis direction |
| | | VSLIMMD_DQ | Limited to dq axis |
| cMREGDIS | Using the Last Value Flag for SIN/COS/SECTOR Registers and [VExMCTLF] | MREGDIS_EFFECTIVE | Enable |
| | | MREGDIS_NOEFFECTIVE | Disabled |
| cCRCEN | Trigger correction enable | CRCEN_DISABLE | Trigger correction disable |
| | | CRCEN_ENABLE | Trigger correction enable |
| cIAPLMD | Current (IA) Search Direction Setting | IPLMD_SHUNT | Shunt mode (IA = VExIAO - VExIAADC) |
| | | IPLMD_SENSOR | Sensor mode (IA = VExIAADC - VExIAO) |
| cIBPLMD | Current (IB) Search Direction Setting | IPLMD_SHUNT | Shunt mode (IB = VExIBO - VExIBADC) |
| | | IPLMD_SENSOR | Sensor mode (IB = VExIBADC - VExIBO) |
| cICPLMD | Current (IC) Search Direction Setting | IPLMD_SHUNT | Shunt mode (IC = VExICO - VExICADC) |
| | | IPLMD_SENSOR | Sensor mode (IC = VExICADC - VExICO) |
| cIDQSEL | Decoupling control input Current selection | IDQSEL_FEEDBACK | Feedback current (ID and IQ) |
| | | IDQSEL_INSTRUCTION | Command current (IDREF, IQREF) |

For setting MODE register

| Constant Name | Meaning of the Constant | Selected Data | Meaning of the Selected Data |
|---|---|---|---|

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | |
|---|---|---|---|
| cIPDEN | Current polarity determination Control | IPDEN_DISABLE | Disable current polarity determination |
| | | IPDEN_ENABLE | Enable current polarity determination |
| cPMDDTCEN | Control of response to dead time compensation in PMD circuit during dead time compensation | PMDDTCEN_DISABLE | Disable dead time correction |
| | | PMDDTCEN_ENABLE | Enable dead time correction |
| cPWMFLEN | Enable 100% duty cycle when PWM high limit is set. | PWMFLEN_DISABLE | 100% disable |
| | | PWMFLEN_ENABLE | 100% enable |
| cPWMBLEN | Enables 0% duty cycle when PWM low limit is set. | PWMBLEN_DISABLE | 0% output disable |
| | | PWMBLEN_ENABLE | 0% output enable |
| cNICEN | Permit decoupling control for task 5 | NICEN_DISABLE | Non-interference control disable |
| | | NICEN_ENABLE | Non-interference control enable |
| cT5ECEN | Expanded control of task 5 (decoupling control, Voltage scalar limit) | T5ECEN_DISABLE | Extended control (decoupling control, voltage scalar) disable |
| | | T5ECEN_ENABLE | Extended control (decoupling control, voltage scalar) enable |
| cAWUMD | Anti-windup (AWU) control when PI control outputlimit | AWUMD_DISABLE | AWU disable |
| | | AWUMD_ENABLE4 | Incorporates the limit amount/4 into the integral term |
| | | AWUMD_ENABLE2 | Incorporates the limit amount/2 into the integral term |
| | | AWUMD_ENABLE1 | Reflects the limit amount in the integral term |
| cCLPEN | Phase Clipping Control during Phase Interpolation | CLPEN_DISABLE | Clipping disable |
| | | CLPEN_ENABLE | Clipping enable |
| cATANMD | ATAN operation control | ATANMD_DISABLE | Operation disable |
| | | ATANMD_IDQ | Calculation of declination of the current vector on d-q coordinate axis |
| | | ATANMD_EDQ | Calculation of declination of the induced voltage vector on d-q coordinate axis |
| cVDCSEL | Power supply voltage storage Register selection | VDCSEL_VDC | *[VExVDC] Storage* |
| | | VDCSEL_VDCL | *[VExVDCL] Storage* |
| 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 |

© 2024
Toshiba Electronic Devices & Storage Corporation

# 13. UART Communication Protocol

The Host MCU board uses UART communication protocol to control this reference design (Motor drive board). This UART communication protocol is described in this section.

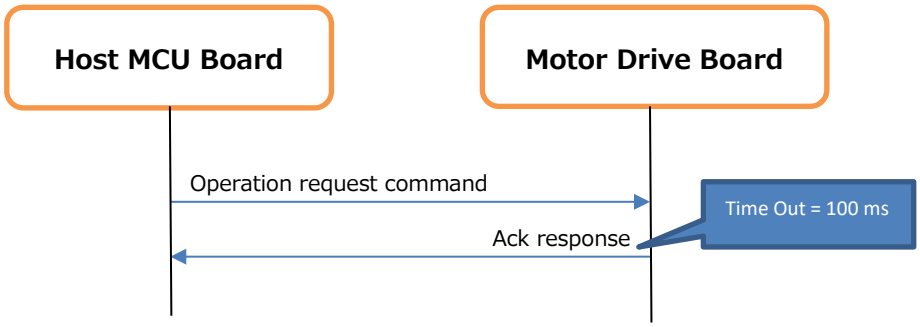
## 13.1 UART Commands

### 13.1.1 Types of UART Commands

This reference design uses two types of UART command as described below.

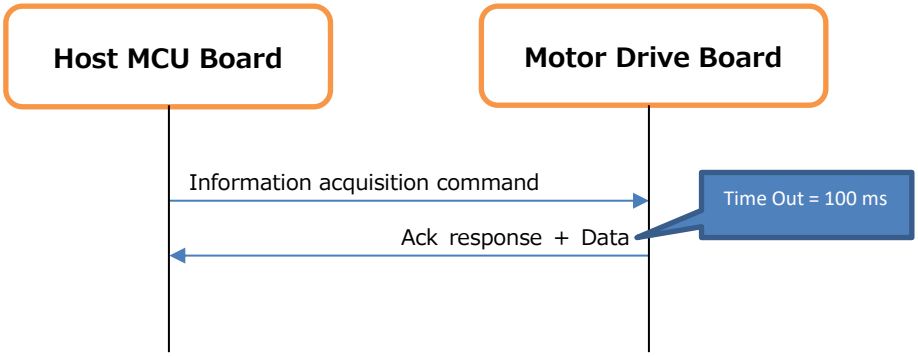
**Command Type 1: Commands to request an operation.**

The sequence and data format of these commands are shown below.




**Command Type 2: Commands to acquire information.**

The sequence and data format of these commands are shown below.




© 2024
Toshiba Electronic Devices & Storage Corporation

Command format of operation request and information acquisition are shown below.

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte0 | Command ID | | | | | | | |
| Byte1 | Data-0 (lower Byte) | | | | | | | |
| Byte2 | Data 1 | | | | | | | |
| Byte3 | Data 2 | | | | | | | |
| Byte4 | Data3 (upper Byte) | | | | | | | |
| Byte5 | CHECKSUM | | | | | | | |

Response format of Ack response and data response are shown below.

| | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte0 | Command ID (Received Command ID) | | | | | | | |
| Byte1 | 0 | 0 | 0 | 0 | 0 | EMG | 0 | ACK |
| Byte2 | Data-0 (lower Byte) | | | | | | | |
| Byte3 | Data 1 | | | | | | | |
| Byte4 | Data 2 | | | | | | | |
| Byte5 | Data3 (upper Byte) | | | | | | | |
| Byte6 | CHECKSUM | | | | | | | |

Command ID: (1byte)

Please refer to the specifications of the UART commands.

CHECKSUM: (1byte)

CHECKSUM represents the lower 8 bits of the sum of the whole data excluding CHECKSUM.

The motor drive board checks the received data and CHECKSUM value and returns the result with an Ack response.

The host MCU board checks the received data and the CHECKSUM value and invalidates the data if there is an error.

EMG bit : (1bit)

Determine EMG status.

0: Normal state (state other than EMG)

1: EMG state

ACK bit : (1bit)

Command response.

0: Reception NG (Nack) (In case of CHECKSUM error/ Invalid command)

1: Reception OK (Ack)

Data: (4byte)

Please refer to the specifications of the UART commands.

© 2024
Toshiba Electronic Devices & Storage Corporation

#### 13.1.2 Communication Settings

UART settings used in this reference design are as follows.

- Baud rate: 9600 bps
- Data: 8 bit
- Stop bit: 1 bit
- Parity :None
- Flow control: None

© 2024
Toshiba Electronic Devices & Storage Corporation

### 13.1.3 Summary of UART Commands

| Command Type | Command Name | Item ID (1 byte) | Command Availability for Each State | |
|---|---|---|---|---|
| | | | Initial State (after reset) | Normal State |
| Operation Request | REQ_SYSTEM_START | 0x10 | ✓ (→ Normal State) | x |
| | REQ_ROTATE_MOTOR | 0x11 | x | ✓ |
| | REQ_CHANGE_MOTOR | 0x12 | x | ✓ |
| | REQ_ON_OFF_PFC | 0x13 | x | ✓ |
| | REQ_ALL_STOP_MOTOR | 0x14 | x | ✓ |
| | REQ_STATUS_CH | 0x15 | x | ✓ |
| | REQ_STATUS_DAC | 0x16 | x | ✓ |
| Information Acquisition | GET_STATE_EMG | 0x81 | x | ✓ |
| | GET_STAGE | 0x82 | x | ✓ |
| | GET_CONTROL_CH | 0x83 | x | ✓ |
| | GET_CARRIER_FREQUENCY | 0x84 | x | ✓ |
| | GET_MOTOR_SPEED_MIN | 0x85 | x | ✓ |
| | GET_MOTOR_SPEED_MAX | 0x86 | x | ✓ |
| | GET_DEAD_TIME | 0x87 | x | ✓ |
| | GET_GATE_ACTIVE | 0x88 | x | ✓ |
| | GET_PSITION_DETECT | 0x89 | x | ✓ |
| | GET_VDC_VOLTAGE | 0x8A | x | ✓ |
| | GET_U_VOLTAGE | 0x8C | x | ✓ |
| | GET_V_VOLTAGE | 0x8D | x | ✓ |
| | GET_W_VOLTAGE | 0x8E | x | ✓ |
| | GET_DAC_DATA | 0x8F | x | ✓ |
| | GET_INTERNAL_AMP | 0x90 | x | ✓ |
| | GET_DIRECTION | 0x91 | x | ✓ |
| | GET_MODULATION | 0x92 | x | ✓ |
| | GET_MOTOR_SPEED | 0x94 | x | ✓ |
| | GET_OUTROOM_TEMP | 0x8B | x | ✓ |
| | GET_OUTPIPE_TEMP | 0x95 | x | ✓ |
| | GET_EXHAUST_TEMP | 0x96 | x | ✓ |
| | GET_DIODE_TEMP | 0x97 | x | ✓ |
| | GET_IGBT_TEMP | 0x98 | x | ✓ |
| | GET_HVMOS_TEMP | 0x99 | x | ✓ |

Note: "x" indicates invalid command, therefore Ack bit=0 response is received.

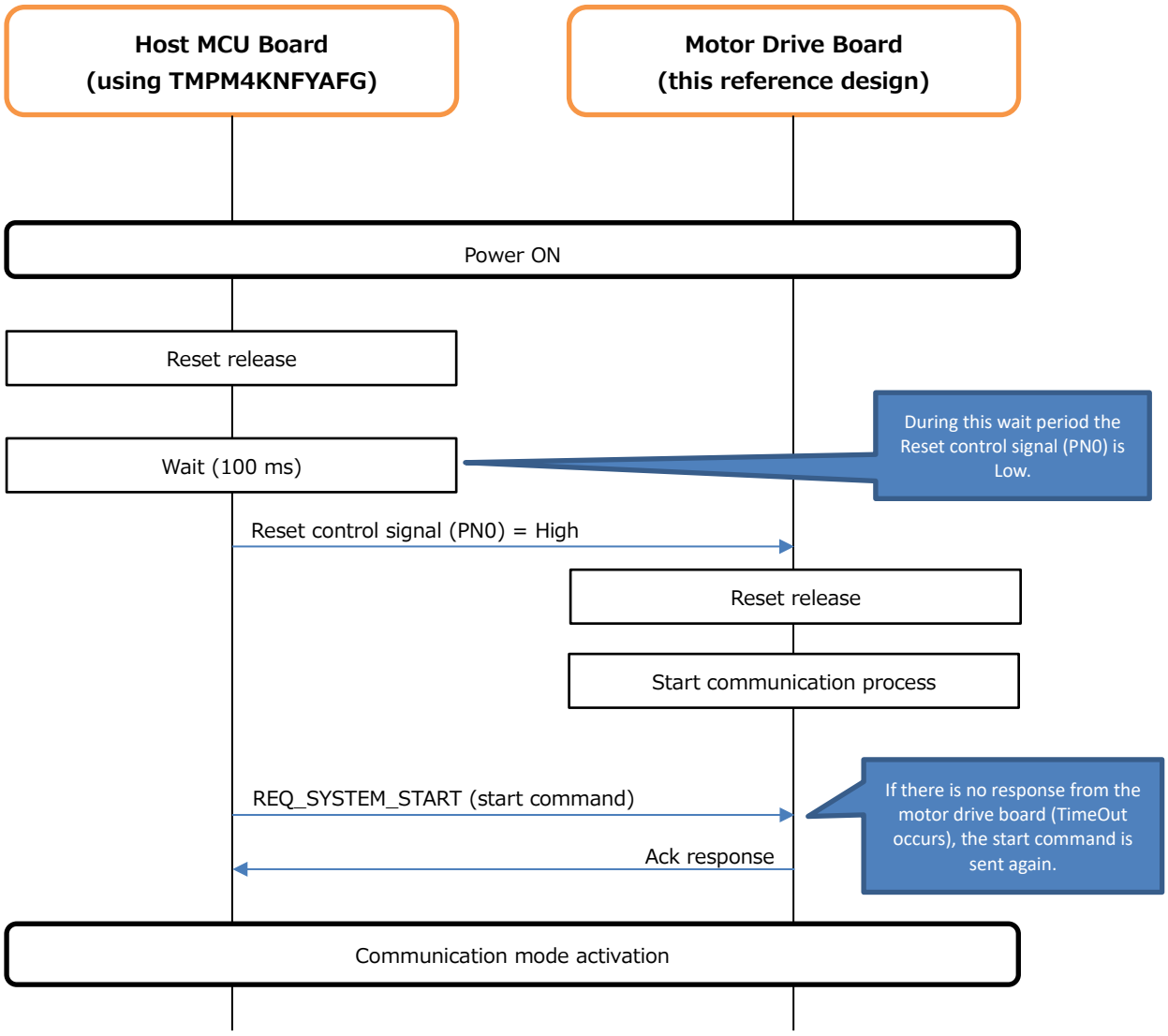
© 2024
Toshiba Electronic Devices & Storage Corporation

## 13.2 System Sequences using UART

This section shows various sequences which uses UART communication.

### 13.2.1 Startup Sequence

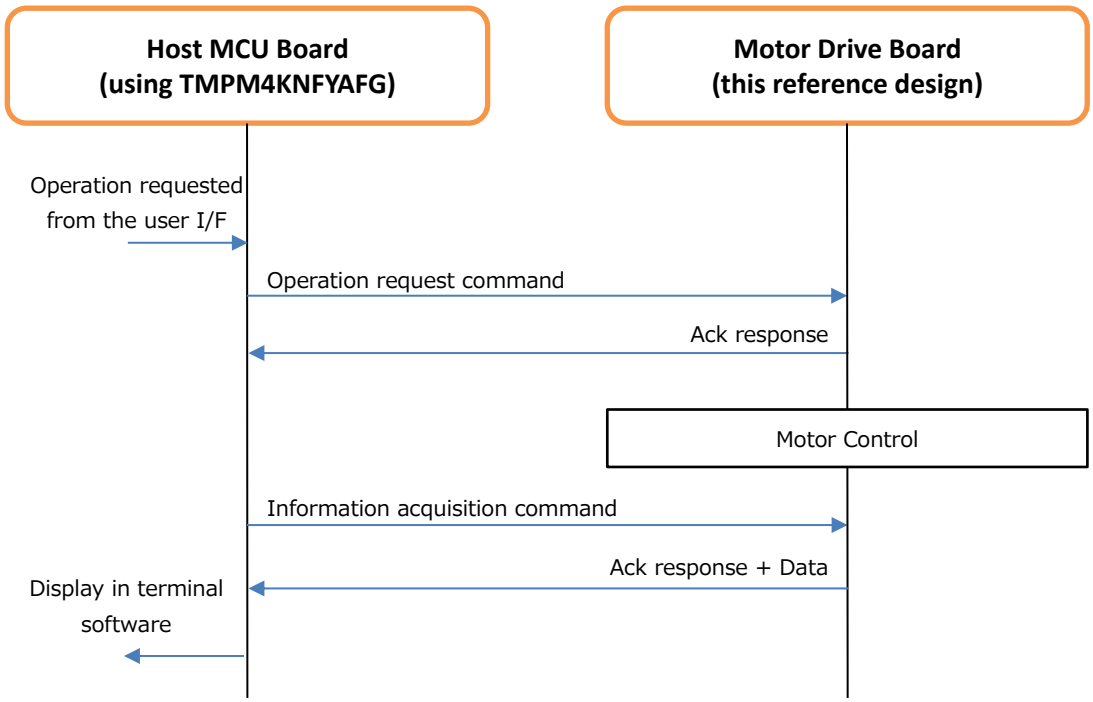
This sequence is followed at the system power up to start the motor.




© 2024
Toshiba Electronic Devices & Storage Corporation

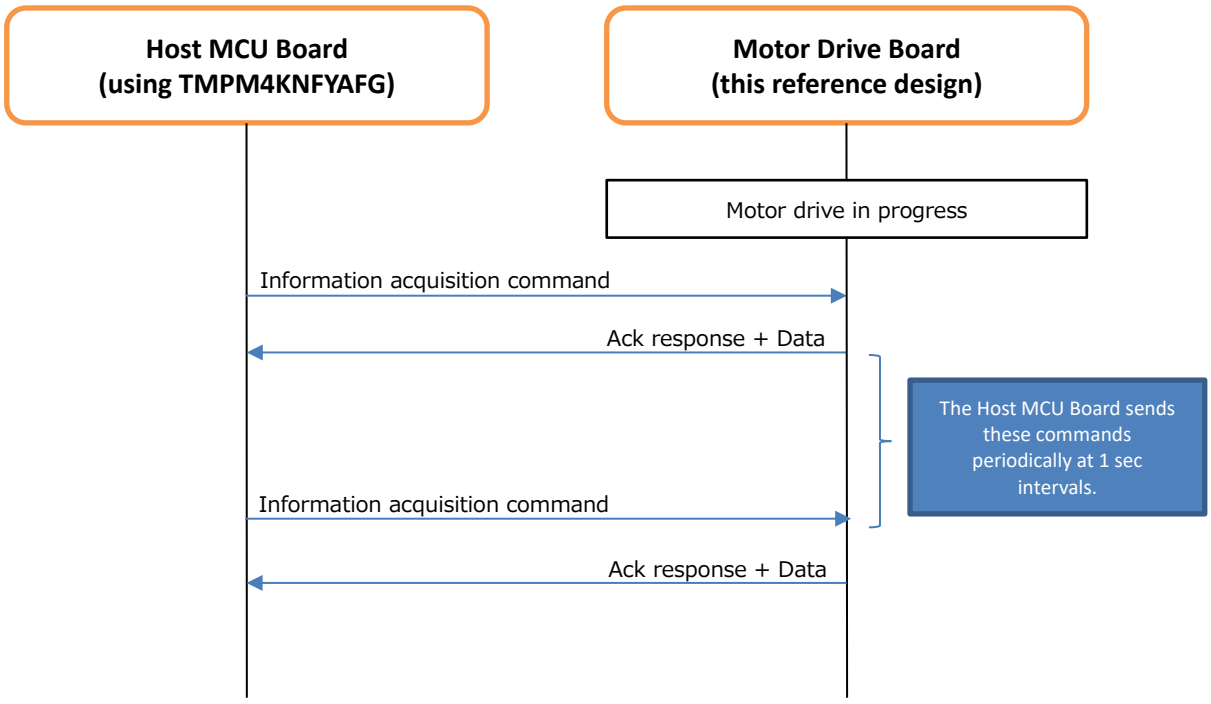
### 13.2.2 Normal Sequence (During Operation Request From User)

This sequence is followed when an operation is requested by the user.




### 13.2.3 Normal Sequence (During Motor Drive)

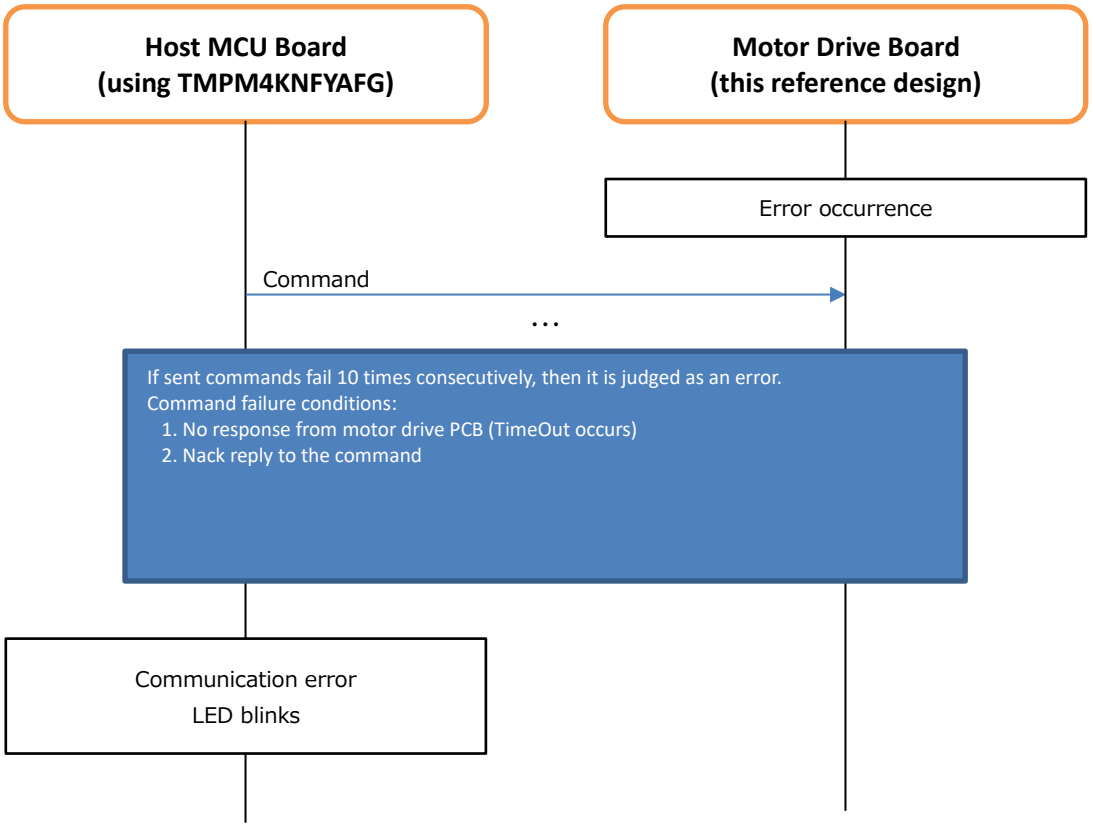
This sequence is followed during normal motor operation.




© 2024
Toshiba Electronic Devices & Storage Corporation

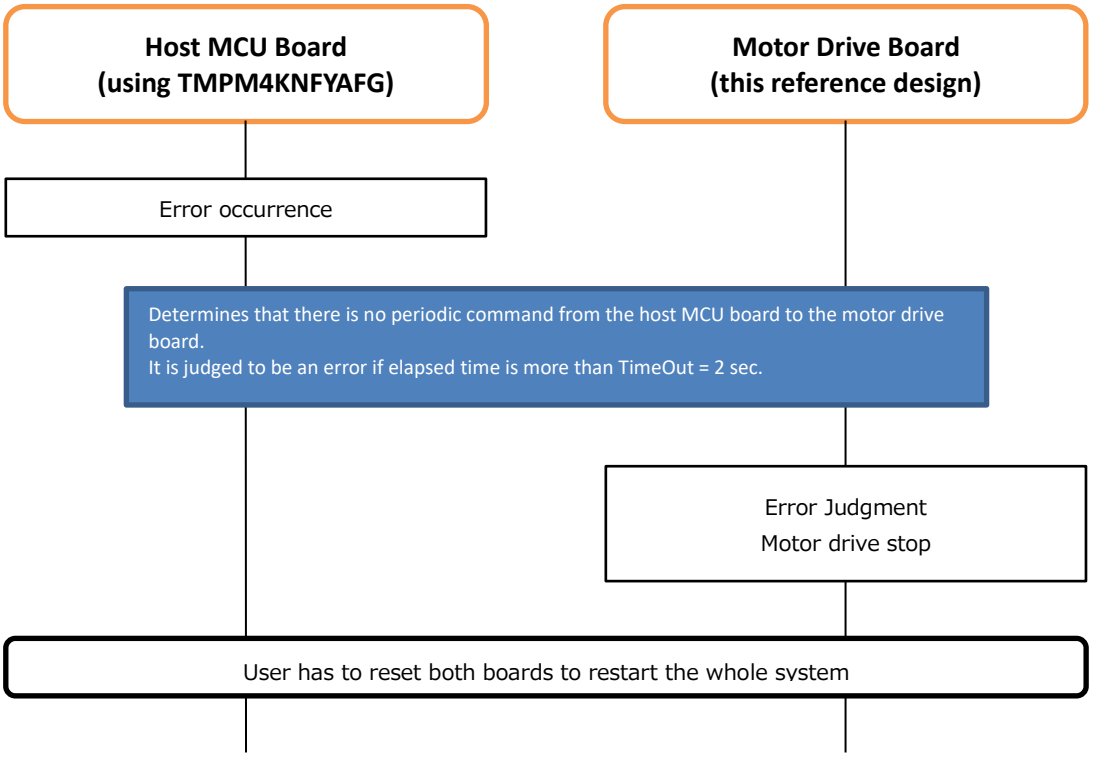
### 13.2.4 Communication Error Sequence

This is followed when a communication error occurs on the motor drive board.




### 13.2.5 Error Sequence

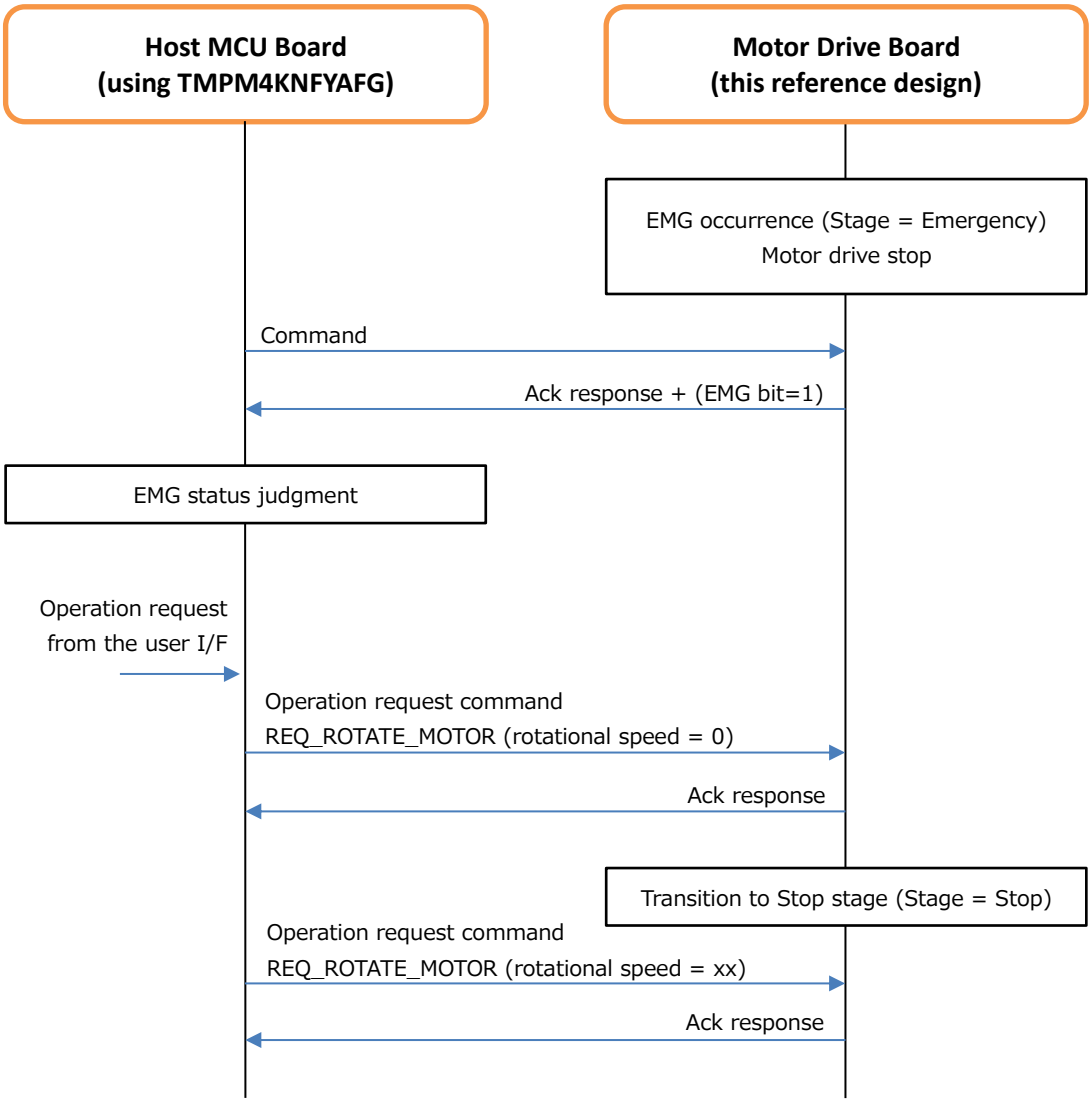
This is followed when error occurs on the host MCU board during motor drive.




© 2024
Toshiba Electronic Devices & Storage Corporation

### 13.2.6 EMG Status Sequence

This sequence is followed when an emergency occurs.




© 2024
Toshiba Electronic Devices & Storage Corporation

### 13.3 Specifications of UART Commands

| Command Type | Command Name | Item ID (1byte) | Data | | Description |
|---|---|---|---|---|---|
| | | | Host MCU Board (4byte) | Motor Drive Board (4byte) | |
| Operation Request | REQ_SYSTEM_START | 0x10 | -<br>(Dummy Data) | -<br>(Dummy Data) | To enable motor control and PFC control. |
| | REQ_ROTATE_MOTOR | 0x11 | Data 0-3:<br>Rotational speed (Integer converted reading) | -<br>(Dummy Data) | To set rotational speed.<br><br>In this reference design this command is used by following push switches:<br>Rotational speed 10 Hz increase<br>Rotational speed 10 Hz decrease<br>Motor rotation forced stop |
| | REQ_CHANGE_MOTOR | 0x12 | Data 1:<br>0x00-0x01:CH0-CH1<br>Other than above: Reserved | -<br>(Dummy Data) | Selects channel for which rotational speed is to be changed<br>CH0 or CH1 |
| | REQ_ON_OFF_PFC | 0x13 | Data 1:<br>0x00: PFC ON<br>0x01: PFC OFF<br>Other than above:<br>Reserved | -(Dummy Data) | PFC ON or OFF |
| | REQ_ALL_STOP_MOTOR | 0x14 | Data 1:<br>0x00: Stop all motors<br>Other than above:<br>Reserved | -<br>(Dummy Data) | Stop all motors |
| | REQ_STATUS_CH | 0x15 | Data 1:<br>0x00-0x02:CH0-CH2<br>Other than above:<br>Reserved | -<br>(Dummy Data) | Select channel whose status is to be acquired.<br>CH0 or CH1 or CH2(PFC) |
| | REQ_STATUS_DAC | 0x16 | Data 1:<br>0x00-0x02:CH0-CH2<br>Other than above:<br>Reserved | -<br>(Dummy Data) | Select channel whose DAC value is to be acquired.<br>CH0 or CH1 or CH2(PFC) |
| Information Acquisition | GET_STATE_EMG | 0x81 | -<br>(Dummy Data) | Data 0:<br>0x00: Hardware EMG_H<br>0x01: Software EMG_S<br>0x02: Current detection error at startup<br>0x03:Vdc error<br>Data 1-3:<br>Reserved | Get EMG condition<br>Hardware EMG_H<br>Software EMG_S<br>Current detection error at startup<br>Vdc error |
| | GET_STAGE | 0x82 | -<br>(Dummy Data) | Data 2:<br>0x00:Stop<br>0x01:Bootstrap<br>0x02:Initposition<br>0x03:Force<br>0x04:Change_up<br>0x05:Steady_A<br>0x06:Emergency<br>Other than above:<br>Reserved | Get main stage state<br>Stop<br>Bootstrap<br>Initposition<br>Force<br>Change_up<br>Steady_A<br>Emergency |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | | |
|---|---|---|---|---|---|
| | GET_CONTROL_CH | 0x83 | -<br>(Dummy Data) | Data 2:<br>0x00-0x02:CH0-CH2<br>Other than above:<br>Reserved | Get control channel<br>CH0 or CH1 or CH2(PFC) |
| | GET_CARRIER_FREQUENCY | 0x84 | -<br>(Dummy Data) | Data 0-3:<br>Fixed-point integer-converted value | Get carrier frequency<br>xxxxx Hz |
| | GET_MOTOR_SPEED_MIN | 0x85 | -<br>(Dummy Data) | Data 0-3:<br>Fixed-point integer-converted value | Get change-up rotational speed<br>cHz MIN = xx Hz |
| | GET_MOTOR_SPEED_MAX | 0x86 | -<br>(Dummy Data) | Data 0-3:<br>Fixed-point integer-converted value | Get maximum rotational speed<br>cHz MAX = xx Hz |
| | GET_DEAD_TIME | 0x87 | -<br>(Dummy Data) | Data 0-3:<br>Dead time(μs)<br>Example: 15.50 μs→1550 | Get dead time<br>x.x μs |
| | GET_GATE_ACTIVE | 0x88 | -<br>(Dummy Data) | Data 0:<br>0x00:H/H<br>0x01:L/L<br>Data 1-3: Reserved | Get gate active polarity<br>H/H or L/L |
| | GET_PSITION_DETECT | 0x89 | -<br>(Dummy Data) | Data 0:<br>0x00: 3 shunt<br>0x01: 1 shunt<br>Data 1-3: Reserved | Get implementation of position detect<br>1 Shunt (SPWM On/Off) or 3 Shunt<br>(Only 1 Shunt is available in this design) |
| | GET_VDC_VOLTAGE | 0x8A | -<br>(Dummy Data) | Data 0-3:<br>Voltage(V)<br>Example: 15.50 V→1550 | Get VDC Voltage<br>xx.x V |
| | GET_U_VOLTAGE | 0x8C | -<br>(Dummy Data) | Data 0-3:<br>Voltage(V)<br>Example: 15.50 V→1550 | Get U Voltage at 0 A<br>x.xxV |
| | GET_V_VOLTAGE | 0x8D | -<br>(Dummy Data) | Data 0-3:<br>Voltage(V)<br>Example: 15.50 V→1550 | Get V Voltage at 0 A<br>x.xxV |
| | GET_W_VOLTAGE | 0x8E | -<br>(Dummy Data) | Data 0-3:<br>Voltage(V)<br>Example: 15.50 V→1550 | Get W Voltage at 0 A<br>x.xxV |
| | GET_DAC_DATA | 0x8F | -<br>(Dummy Data) | Data 0:A<br>Data 1:B<br>Data 2:C<br>Data 3:D<br>0x00:TMPREG0<br>0x01:TMPREG1<br>0x02:TMPREG2<br>0x03:theta.half[1]<br>0x04:Id_ref<br>0x05:Id<br>0x06:Iq_ref<br>0x07:Iq<br>0x08:omega_com.half[1]<br>0x09:omega.half[1]<br>0x0A:omega_dev | Get DAC data<br>A: xx B:xx C:xx D:xx (Note 1) or No Output<br><br>DAC power Mode<br>Mode0 DAC Data = A:TMPREG0 B:TMPREG1 C:TMPREG2 D:theta.half[1]<br>Mode1 DAC Data = A: Id_ref B:Id C: Iq_ref D: Iq<br>Mode2 DAC Data = A: omega_com.half[1] B: omega.half[1] C: omega_dev D: Iq_ref<br>Mode3 DAC Data = A: TMPREG0 B: Iq_ref C:Id_ref D:omega.half[1] |
| | GET_INTERNAL_AMP | 0x90 | -<br>(Dummy Data) | Data 0:<br>0x00: External<br>0x01: Built-in MCU<br>Data 1-3:<br>Reserved | Get implementation of OpAmp |
| | GET_DIRECTION | 0x91 | -<br>(Dummy Data) | Data 0:<br>0x00:CW<br>0x01:CCW<br>Data 1-3:<br>Reserved | Get direction of motor rotation<br>CW or CCW<br>(Only CW is available in this design) |

© 2024
Toshiba Electronic Devices & Storage Corporation

| | | | | | |
|---|---|---|---|---|---|
| | GET_MODULATION | 0x92 | - (Dummy Data) | Data 0:<br>0x00: 3 phase modulation<br>0x01: 2 phase modulation<br>Data 1-3:<br>Reserved | Get current modulation<br>2 Phase or 3 Phase |
| | GET_MOTOR_SPEED | 0x94 | - (Dummy Data) | Data 0:<br>0x00-0xFF: RPM<br>Data 1-3:<br>Reserved | Get rotational speed<br>CH1 = 20 Hz |
| | GET_OUTROOM_TEMP | 0x8B | - (Dummy Data) | Data 0-3:<br>Temperature(degree)<br>Example: 15.50 degree → 1550 | Get inverter temperature(OUTROOM)<br>xx.x degree or N.A |
| | GET_OUTPIPE_TEMP | 0x95 | - (Dummy Data) | Data 0-3:<br>Temperature(degree)<br>Example: 15.50 degree → 1550 | Get inverter temperature(OUTPIPE)<br>xx.x degree or N.A |
| | GET_EXHAUST_TEMP | 0x96 | - (Dummy Data) | Data 0-3:<br>Temperature(degree)<br>Example: 15.50 degree → 1550 | Get inverter temperature(EXAUST)<br>xx.x degree or N.A |
| | GET_DIODE_TEMP | 0x97 | - (Dummy Data) | Data 0-3:<br>Temperature(degree)<br>Example: 15.50 degree → 1550 | Get inverter temperature(DIODE)<br>xx.x degree or N.A |
| | GET_IGBT_TEMP | 0x98 | - (Dummy Data) | Data 0-3:<br>Temperature(degree)<br>Example: 15.50 degree → 1550 | Get inverter temperature(IGBT)<br>xx.x degree or N.A |
| | GET_HVMOS_TEMP | 0x99 | - (Dummy Data) | Data 0-3:<br>Temperature(degree)<br>Example: 15.50 degree → 1550 | Get inverter temperature(HVMOS)<br>xx.x degree or N.A |

© 2024
Toshiba Electronic Devices & Storage Corporation

© 2024
Toshiba Electronic Devices & Storage Corporation

# 14. Appendix

## 14.1 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.

## 14.2 Normalization (Normalize)

Normalization is to transform the data according to a certain rule to make it easier to use the data.
In this application note, 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)

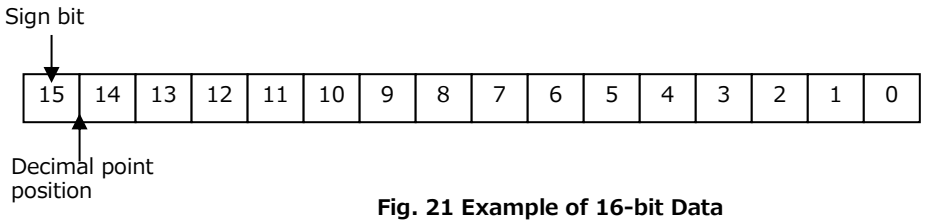



**Fig. 21 Example of 16-bit Data**

In this application note, 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).

## 14.3 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 5 Single Precision (16-bit) Decimal Format**

| Q Format | Number of Decimal Bits | Maximum Positive Value (0 X 7 FFF) | Maximum Negative Value (0 X 8000) |
|---|---|---|---|
| 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 6 Double Precision (32-bit) Decimal Format**

© 2024
Toshiba Electronic Devices & Storage Corporation

| Q Format | Number of Decimal Bits | Maximum Positive Value (0 x 7FFFFFF) | Maximum Negative Value (0 x 80000000) |
|---|---|---|---|
| Q31 | 31 | 0.999999999534338 | -1 |
| Q30 | 30 | 1.999999999068670 | -2 |
| Q29 | 29 | 3.999999998137350 | -4 |
| Q28 | 28 | 7.999999996274700 | -8 |
| Q27 | 27 | 15.999999992549400 | -16 |
| Q26 | 26 | 31.999999985098800 | -32 |
| Q25 | 25 | 63.999999970197600 | -64 |
| Q24 | 24 | 127.999999940395000 | -128 |
| Q23 | 23 | 255.999999880790000 | -256 |
| Q22 | 22 | 511.999999761581000 | -512 |
| Q21 | 21 | 1023.999999523160000 | -1024 |
| Q20 | 20 | 2047.999999046320000 | -2048 |
| Q19 | 19 | 4095.999998092650000 | -4096 |
| Q18 | 18 | 8191.999996185300000 | -8192 |
| Q17 | 17 | 16383.999992370600000 | -16384 |
| Q16 | 16 | 32767.999984741200000 | -32768 |
| Q15 | 15 | 65535.999969482400000 | -65536 |
| Q14 | 14 | 131071.999938965000000 | -131072 |
| Q13 | 13 | 262143.999877930000000 | -262144 |
| Q12 | 12 | 524287.999755859000000 | -524288 |
| Q11 | 11 | 1048575.999511720000000 | -1048576 |
| Q10 | 10 | 2097151.999023440000000 | -2097152 |
| Q9 | 9 | 4194303.998046870000000 | -4194304 |
| Q8 | 8 | 8388607.996093750000000 | -8388608 |
| Q7 | 7 | 16777215.992187500000000 | -16777216 |
| Q6 | 6 | 33554431.984375000000000 | -33554432 |
| Q5 | 5 | 67108863.968750000000000 | -67108864 |
| Q4 | 4 | 134217727.937500000000000 | -134217728 |
| Q3 | 3 | 268435455.875000000000000 | -268435456 |
| Q2 | 2 | 536870911.750000000000000 | -536870912 |
| Q1 | 1 | 1073741823.500000000000000 | -1073741824 |
| Q0 | 0 | 2147483647.000000000000000 | -2147483648 |

### 14.4 Operation at Fixed Point

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.

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

© 2024
Toshiba Electronic Devices & Storage Corporation

# 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.

© 2024
Toshiba Electronic Devices & Storage Corporation