エアコン室外機リファレンスモデル
ソフトウェアガイド

RD219-SWGUIDE-01

東芝デバイス＆ストレージ株式会社

© 2024
Toshiba Electronic Devices Solutions Corporation

目次

© 2024
Toshiba Electronic Devices Solutions Corporation

© 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. 概要

本ドキュメントはエアコン室外機制御回路リファレンスデザイン(RD219)のソフトウェア仕様を記述したものです。本リファレンスデザインでは制御用ホスト MCU として MikroElektronika 製 Clicker 4 for TMPM4K を使用します。

1.1 仕様概要

- 使用 MCU：TMPM4KLFYAUG (動作クロック 160 MHz)
- 使用パワーデバイス ：TPD4204F (ファン駆動インバーター), TK20A60W5 (コンプレッサー駆動インバーター), GT30J65MRB ならびに TRS24N65FB (PFC)
- 使用モーター ： ファン用ブラシレスモーター、コンプレッサー用ブラシレスモーター
- 開発環境 ：EWARM Ver9.30.1　または KEIL Ver6.195.0.0

- 制御概要
  ・ブラシレスモーターセンサーレスベクトル制御
  ・外部からの速度制御、PFC 制御(On/Off)に対応
  ・評価用 DAC 出力(変数値アナログ出力用)と LED 出力(モニター用 LED)に対応

- モーター制御内容

項目	制御内容
電流検出方式	1 シャント
位置検出方式	センサーレス
PWM 変調方式	2 相変調

- PFC 制御内容

項目	制御内容
PFC 方式	CCM 方式
コイル数	1
電流検出方式	1 シャント

© 2024
Toshiba Electronic Devices Solutions Corporation

1.2 モーター制御処理概要

ベクトル制御ソフトウェアは、ユーザーインターフェース処理を行うアプリケーション、状態遷移（State transition）によりモーター動作状態（Motor operation status）を制御するモーター制御、モーター駆動回路を直接アクセスしてモーターの駆動処理を行うモーター駆動の 3 階層で構成されます。

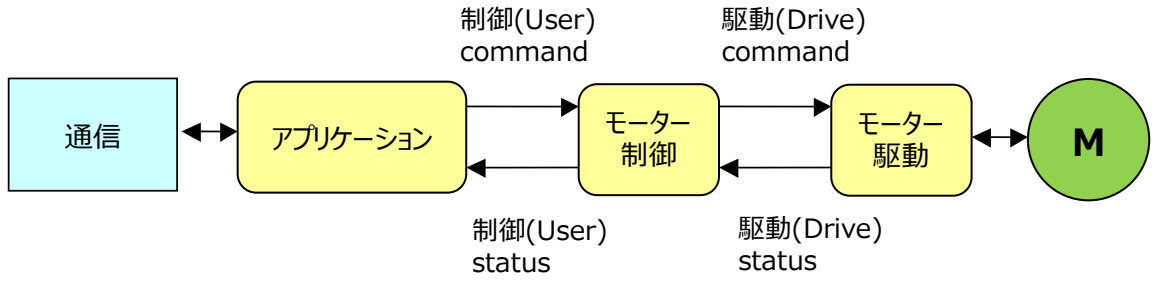


図 1 ベクトル制御ソフトウェアの構造

i. アプリケーションは、スイッチ、キー、通信などで設定した制御コマンドを入力し、その他の制御コマンドとともにモーター制御に与えます。また制御ステータスをモーター制御から取得し、必要な処理を行うとともに、ポートなどに出力します。

ii. モーター制御はアプリケーションから与えられる制御コマンドを読み取り、モーター動作状態に従ってより具体的な駆動コマンドに変換し、モーター駆動に与えます。また駆動ステータスをモーター駆動から取得し、必要な処理を行うとともにアプリケーションに転送します。

モーター駆動はモーター制御から与えられる駆動コマンドを読み取り、モーターを駆動します。またモーターの動作を監視し、その状態に従って必要な処理を行うとともに、駆動ステータスをモーター制御に転送します。

例えば、モーター回転中にアプリケーションから新たな制御目標速度が与えられたとき、モーター駆動は急激な目標速度の変化に対応できないため、いったんモーター制御内で徐々に変化する駆動目標速度に変換してからモーター駆動に与えます。

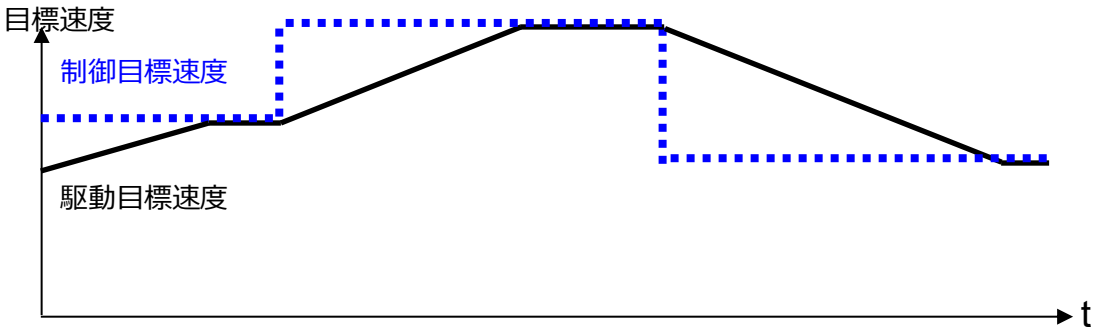


図 2 制御目標速度と駆動目標速度

© 2024
Toshiba Electronic Devices Solutions Corporation

システムブロック図

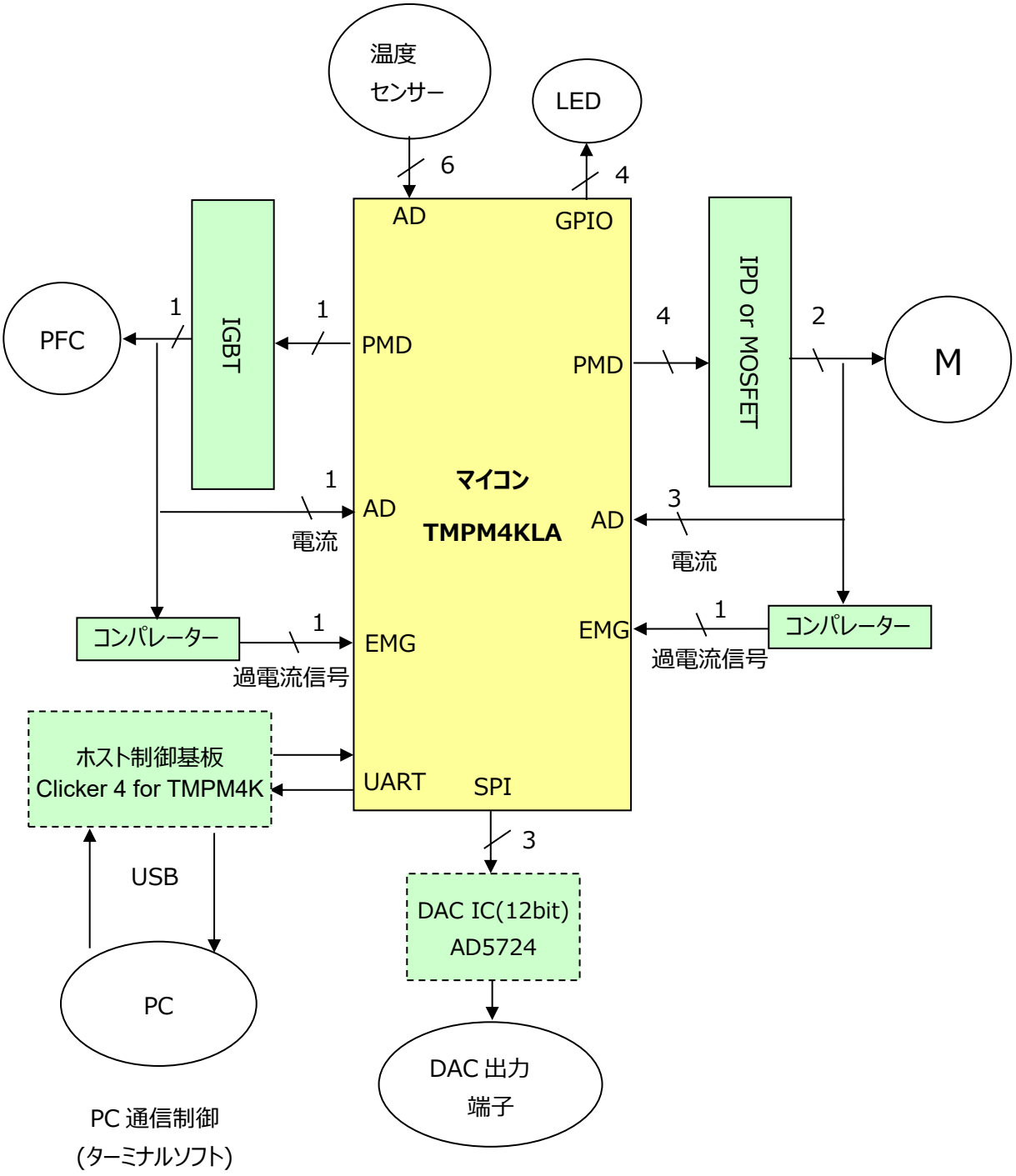


図 3 システムブロック図

© 2024
Toshiba Electronic Devices Solutions Corporation

2. ソースファイル構成

```
source
|   B_User.c              ベクトル制御ユーザー設定関連ソースファイル
|   B_User.h              ベクトル制御ユーザー設定関連ヘッダーファイル
|   C_Control.c           ベクトル制御コントロール関連ソースファイル
|   C_Control.h           ベクトル制御コントロール関連ヘッダーファイル
|   dac_drv.c             DAC IC ドライバーソースファイル
|   dac_drv.h             DAC IC ドライバーヘッダーファイル
|   D_Driver.c            ベクトル制御ドライバーソースファイル
|   D_Driver.h            ベクトル制御ドライバーヘッダーファイル
|   D_Para.h              ベクトル制御パラメーター(チャネル共通)ヘッダーファイル
|   D_Para_Comp.h         ベクトル制御パラメーター(チャネル 0 用)ヘッダーファイル
|   D_Para_Fan.h          ベクトル制御パラメーター(チャネル 1 用)ヘッダーファイル
|   dac_drv.c             DAC 関連ソースファイル
|   dac_drv.h             DAC 関連ヘッダーファイル
|   E_Sub.a               演算用関数ライブラリーファイル
|   E_Sub.h               演算用関数ライブラリーヘッダーファイル
|   HPFC_drv.c            PFC 関連ソースファイル
|   HPFC_drv.h            PFC 関連ヘッダーファイル
|   HPFC_Para.h           PFC 制御パラメーターヘッダーファイル
|   initial.c             マイコン初期設定関連ソースファイル
|   initial.h             マイコン初期設定関連ヘッダーファイル
|   interrupt.c           割り込み制御関連ソースファイル
|   interrupt.h           割り込み制御関連ヘッダーファイル
|   ipdefine.h            マイコン設定関連ヘッダーファイル
|   main.c                メインルーチン
|   mcuip_drv.c           マイコンハード設定ドライバーソースファイル
|   mcuip_drv.h           マイコンハード設定ドライバーヘッダーファイル
|   sys_macro.h           マクロ定義用ヘッダーファイル
|   system_int.c          割り込み関数ソースファイル
|   system_int.h          割り込み関数ヘッダーファイル
|   usercon.c             ユーザー制御用ソースファイル
|   usercon.h             ユーザー制御用ヘッダーファイル
└──Libraries        CMSIS コア、各 IP 用ライブラリーが格納されています。
    └──M4Kx
        ├──CMSIS            CMSISコアが格納されています。
        |   |   system_TMPM4KyA.c   マイコン初期設定用ソースファイル
        |   |   system_TMPM4KyA.h   マイコン初期設定用ヘッダーファイル
        |   |   TMPM4KLA.h          マイコンレジスター定義ヘッダーファイル
        |   |   TMPM4KyA.h          マイコンレジスター共通定義ヘッダーファイル
        |   └──startup
        |       ├──arm
        |       |       startup_TMPM4KLA.s      スタートアップアセンブラーソース(arm 用)
        |       └──iar
        |               startup_TMPM4KLA.s      スタートアップアセンブラーソース(IAR 用)
        |
        └──IP_Driver        ペリフェラルドライバーが格納されています。
            ipdrv_adc.c
```

© 2024
Toshiba Electronic Devices Solutions Corporation

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. 評価環境について

3.1 開発ツール

本ソフトは以下の開発ツールを使用して開発されたものとなります。

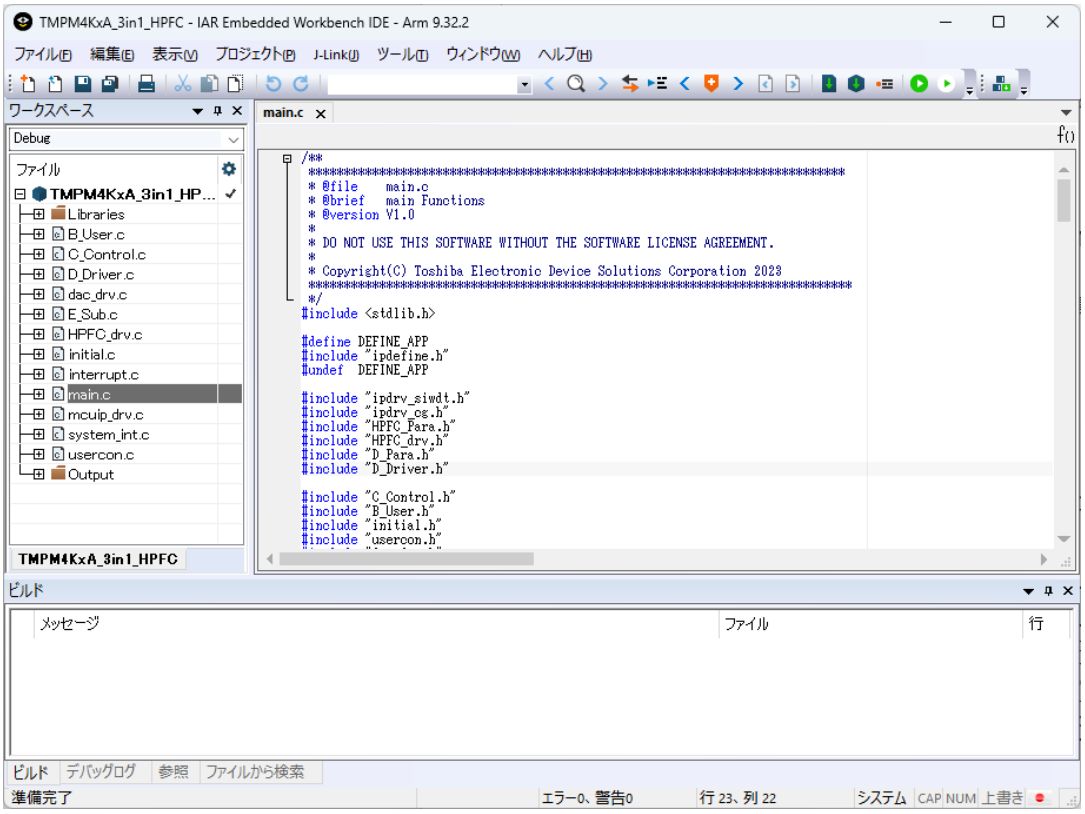
IAR Embeded Workbench for ARM	9.30.1
KEIL µVision MDK-Lite	6.195.0.0

3.2 プロジェクトの立ち上げ方

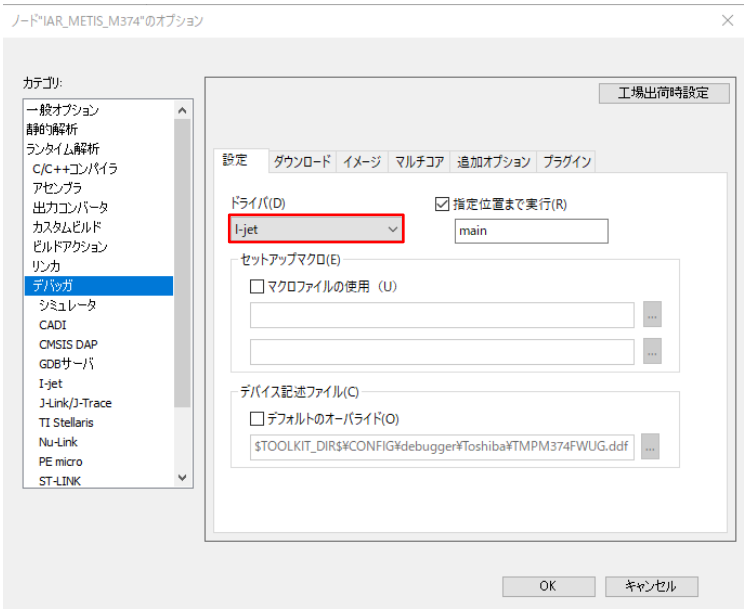
IAR Embeded Workbench を例として説明します。

1. iar¥TMPM4KxA_3in1_HPFC.eww をダブルクリック、または[ファイル] > [開く] > [ワークスペース]から TMPM4KxA_3in1_HPFC.eww を開いてください。

2. 下記画面が立ち上がります。



3. オプションを開き、使用するツールの選択を行ってください。
   [プロジェクト] > [オプション] > [デバッガ]の<設定>タブ



使用するツールを選択してください。

4. デバッグを開始するときは、ツールを接続し[プロジェクト] > ダウンロードしてデバッグ を選択するか下記ボタンを押してください。

3.3 DAC 出力

変数の変化をオシロスコープなどで見るための DAC 出力機能を実装しています。
DAC 出力を有効にするためには、D_Para.h の下記定義を有効にしてください。
#define __USE_DAC

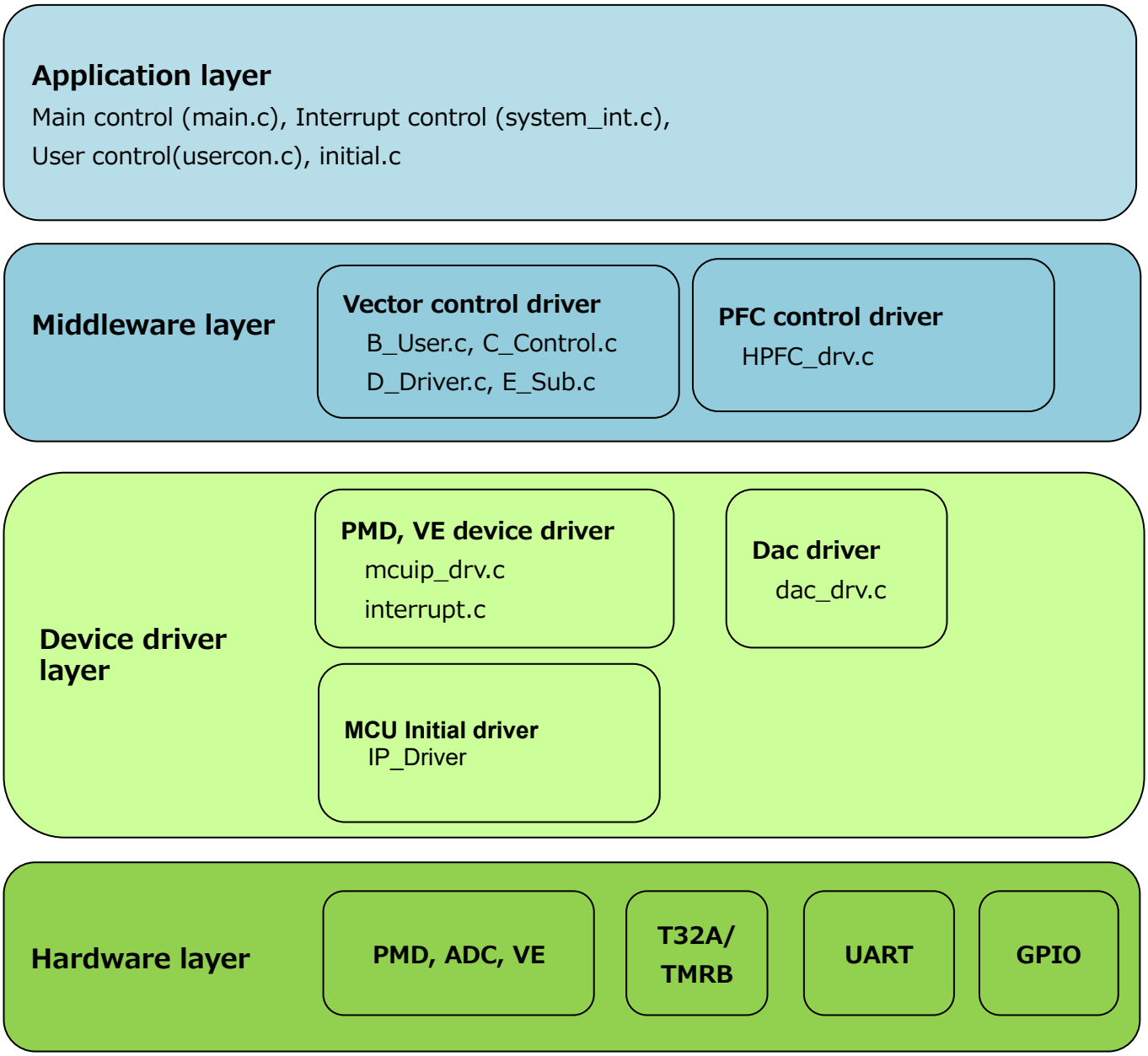
DAC 出力させる変数は、ファイル usercon.c の関数 UiOutDataStart()に記載しています。
確認する変数がない場合などは、必要に応じて追加してください。

«DAC 出力設定用変数»

dac.select	DAC 出力選択
dac.motch	DAC 出力させるモーターCH を設定してください。
dac.datsft0 - 3	データ量シフト量を設定してください。

© 2024
Toshiba Electronic Devices Solutions Corporation

4. モジュール構成



© 2024
Toshiba Electronic Devices Solutions Corporation

5. マイコンハードウェアリソースのアサイン

5.1 周辺インターフェース

IP		制御内容	備考
T32A	CH0	4kHz 周期割り込み	タイマーA
	CH1	未使用	
	CH2	未使用	
	CH3	未使用	
	CH4	未使用	
	CH5	未使用	
TSPI	CH0	未使用	
	CH1	DAC IC 制御	SIO
	CH2	未使用	
	CH3	未使用	
UART	CH0	メイン基板通信	
	CH1	未使用	
	CH2	未使用	
	CH3	未使用	
ADC	UNITA	モーターCH0 コイル電流、電源電圧値取得	
	UNITB	モーターCH1 コイル電流	
	UNITC	PFC 電流取得、AC 電圧取得	
A-PMD	CH0	モーターCH0 制御	
	CH2	モーターCH1 制御	
	CH3	PFC 制御	
A-VE+	CH0	モーターCH0 制御	

5.2 割り込み

要因名	処理内容	関数名
INTT32A0AC_IRQn	4kHz 周期タイミング作成	INTT32A0AC_IRQHandler
INTVCN0_IRQn	・FAN 用ベクトル制御ソフト処理 (VE によるベクトル制御用) ・PFC 電圧制御	INTVCN0_IRQHandler
INTADBPDB_IRQn	COMP 用ベクトル制御ソフト処理 (ソフトウェアによるベクトル制御用)	INTADBPDB_IRQHandler
INTADCPDB_IRQn	PFC 電流制御処理	INTADCPDB_IRQHandler
INTSC1TX_IRQn	DAC IC 制御	INTSC1TX_IRQHandler
INTSC0RX_IRQn	メイン基板受信	INTSC0RX_IRQHandler
INTSC0TX_IRQn	メイン基板送信	INTSC0TX_IRQHandler

割り込み優先度は、ipdefine.h の下記定数で変更可能です。

```
/* 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

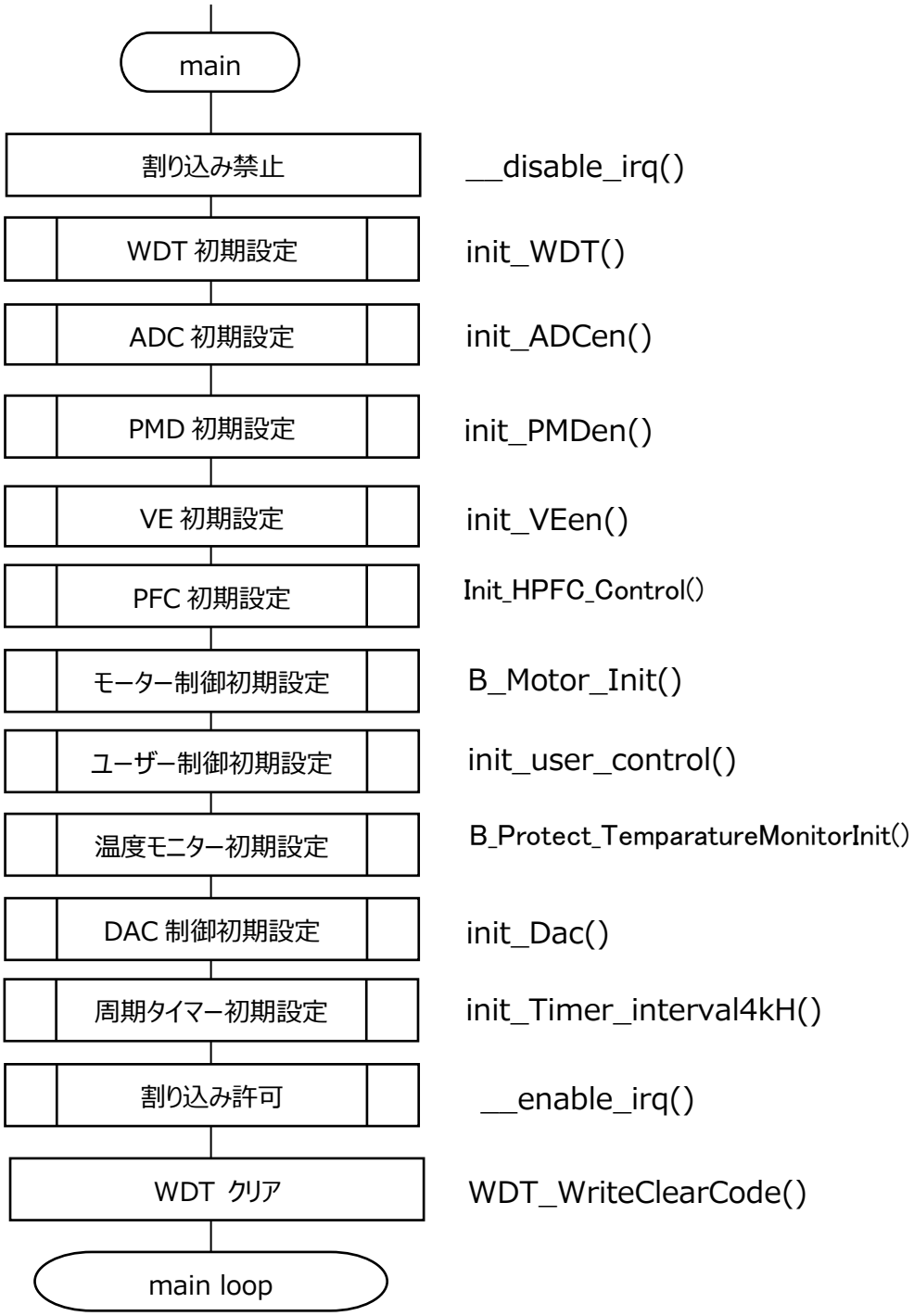
5.3 GPIO

PORT	機能
PF0	SWD-SWDIO
PU0	予約(室内機通信用 UART-RXD2)
PU1	予約(室内機通信用 UART-TXD2)
PU2	予約(膨張弁 A 制御用 GPIO)
PU3	予約(膨張弁 B 制御用 GPIO)
PU4	PFC 用 PWM 出力
PU5	LED4 用 GPIO
PU6	PFC EMG 入力
PA2	LED3/PMD2DBG 用 GPIO
PA3	予約(膨張弁 C 制御用 GPIO)
PA4	予約(膨張弁 D 制御用 GPIO)
PL0	CH0(FAN)モーター用コイル電流検出(OP-AMP+)
PL1	CH0(FAN)モーター用コイル電流検出(OP-AMP-)
PL2	CH1(コンプレッサー)モーター用コイル電流検出(OP-AMP+)
PL3	CH0(コンプレッサー)モーター用コイル電流検出(OP-AMP-)
PL4	IGBT 温度検出
PL5	Diode 温度検出
PL6	直流リンク電圧検出
PL7	予約(PTC リレー制御用 GPIO)
PK2	Out pipe temperature sample
PK1	予約(熱排出パイプ温度検出)
PK0	予約(室外温度検出)
PJ2	HVMOS 温度検出
PJ1	PFC 用コイル電流検出
PJ0	PFC 用 AC 電圧検出
PC0	メイン基板通信用 UART-TXD0
PC1	メイン基板通信用 UART-RXD0
PC2	LED1/PMD0DBG 用 GPIO
PC3	LED2/PMD1DBG 用 GPIO
PB0	CH0(FAN)モーターU 相 PWM 出力
PB1	CH0(FAN)モーターX 相 PWM 出力
PB2	CH0(FAN)モーターV 相 PWM 出力
PB3	CH0(FAN)モーターY 相 PWM 出力
PB4	CH0(FAN)モーターW 相 PWM 出力
PB5	CH0(FAN)モーターZ 相 PWM 出力
PB6	CH0(FAN)モーターEMG 入力
PG2	Low 固定(シングルチップモード)
PG3	予約(4way バルブリレー制御用 GPIO)
PG4	DAC 通信用 SYNC 出力
PG5	DAC 通信用 SDIN 出力
PG6	DAC 通信用 SCLK 出力
PE0	CH0(コンプレッサー)モーターU 相 PWM 出力
PE1	CH0(コンプレッサー)モーターX 相 PWM 出力
PE2	CH0(コンプレッサー)モーターV 相 PWM 出力
PE3	CH0(コンプレッサー)モーターY 相 PWM 出力
PE4	CH0(コンプレッサー)モーターW 相 PWM 出力
PE5	CH0(コンプレッサー)モーターZ 相 PWM 出力
PE6	CH0(コンプレッサー)モーターEMG 入力
PH0	予約(10 MHz Crystal Oscillator)
PH1	予約(10 MHz Crystal Oscillator)
PF1	SWD-SWCLK

© 2024
Toshiba Electronic Devices Solutions Corporation

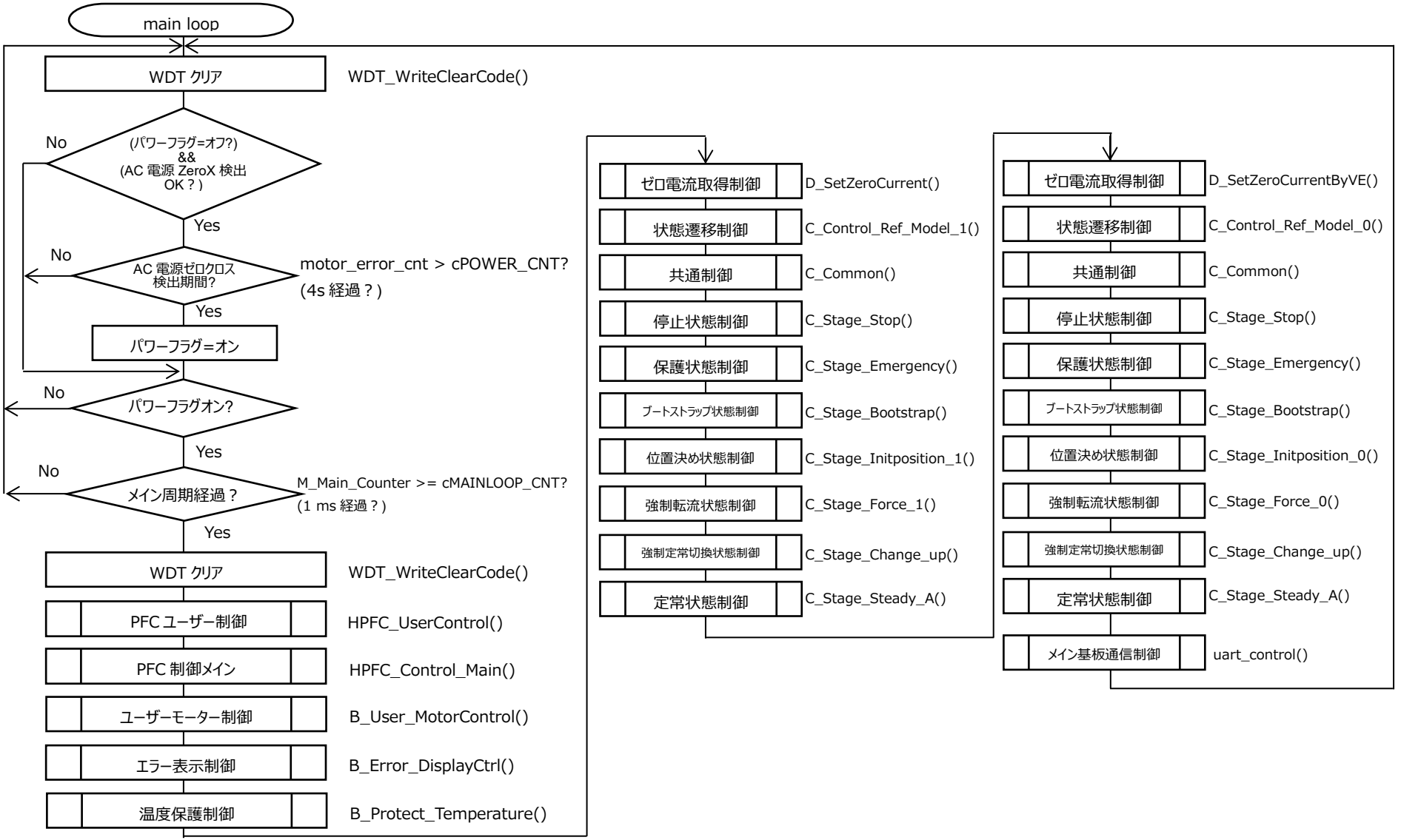
6. ジェネラルフロー

6.1 メインルーチン(main)



© 2024
Toshiba Electronic Devices Solutions Corporation

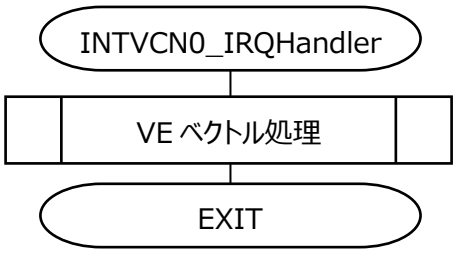
6.2 メインループ(main_loop)



© 2023
Toshiba Electronic Devices Solutions Corporation

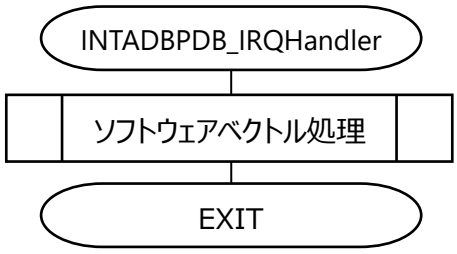
6.3 割り込み

VE スケジュール完了時



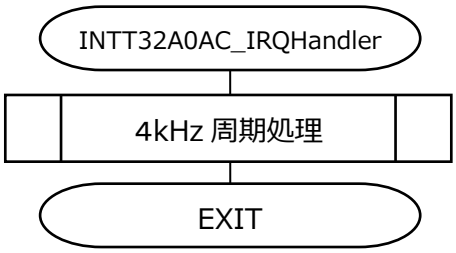
INT_VectorContorlByVE()

AD 変換終了割り込み



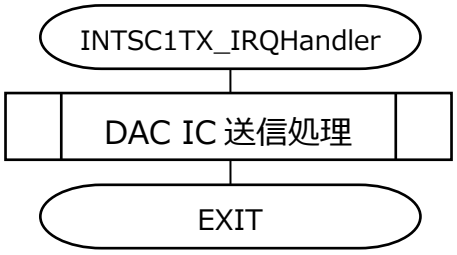
INT_VectorContorlBySoft()

4kHz 周期ごと



INT_interval_4kHz ()

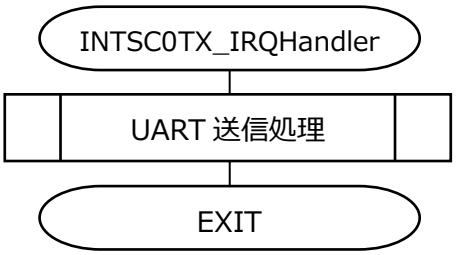
DAC 通信 24 bit 送信完了時



DacWriteContinue ()

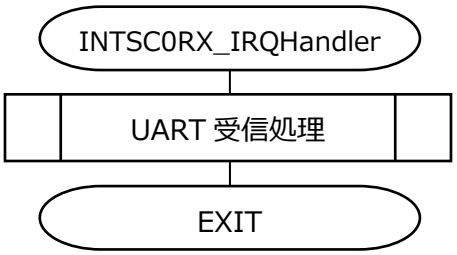
© 2024
Toshiba Electronic Devices & Storage Corporation

メイン基板通信送信完了時



INT_uart0tx ()

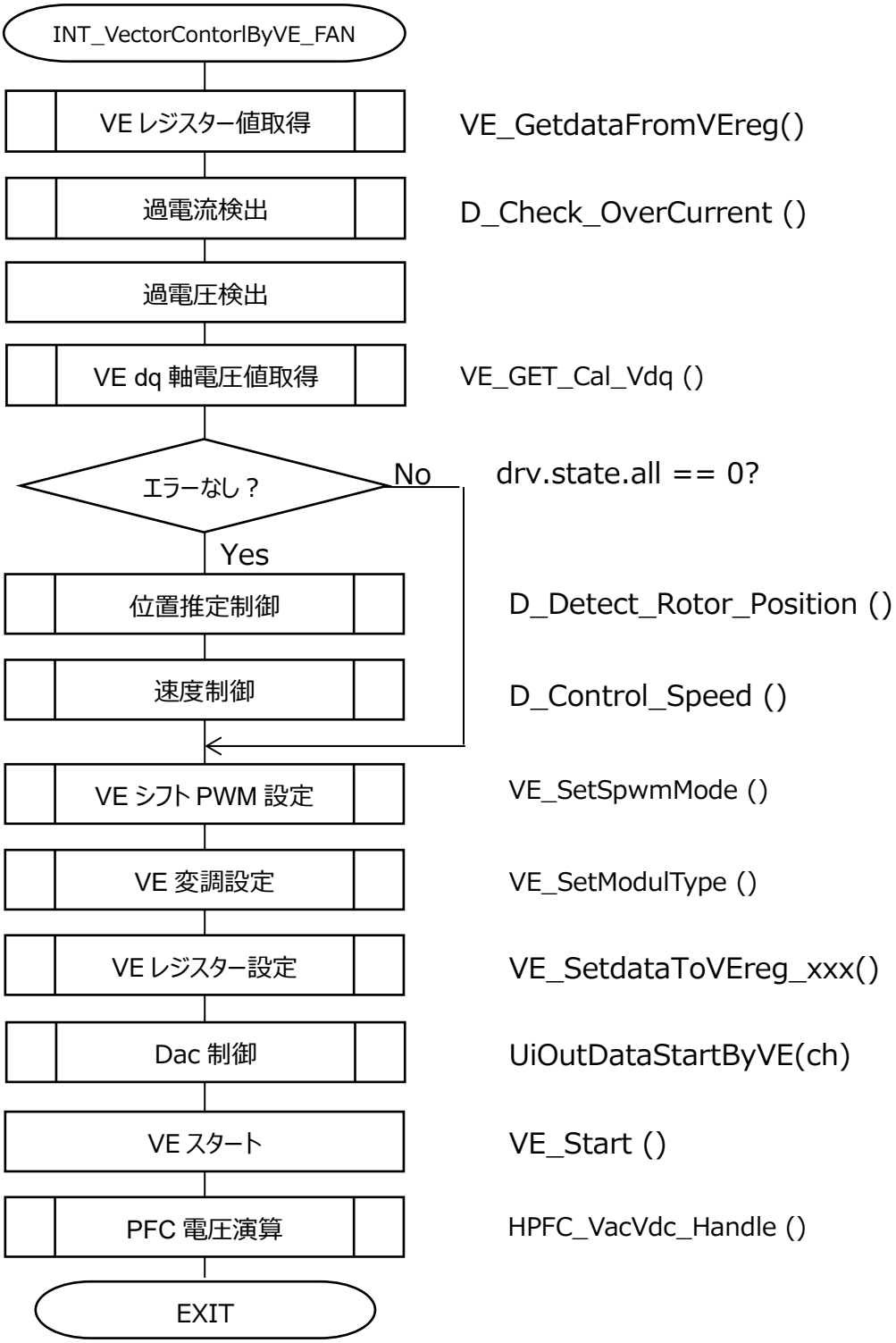
メイン基板通信受信完了時



INT_uart0rx ()

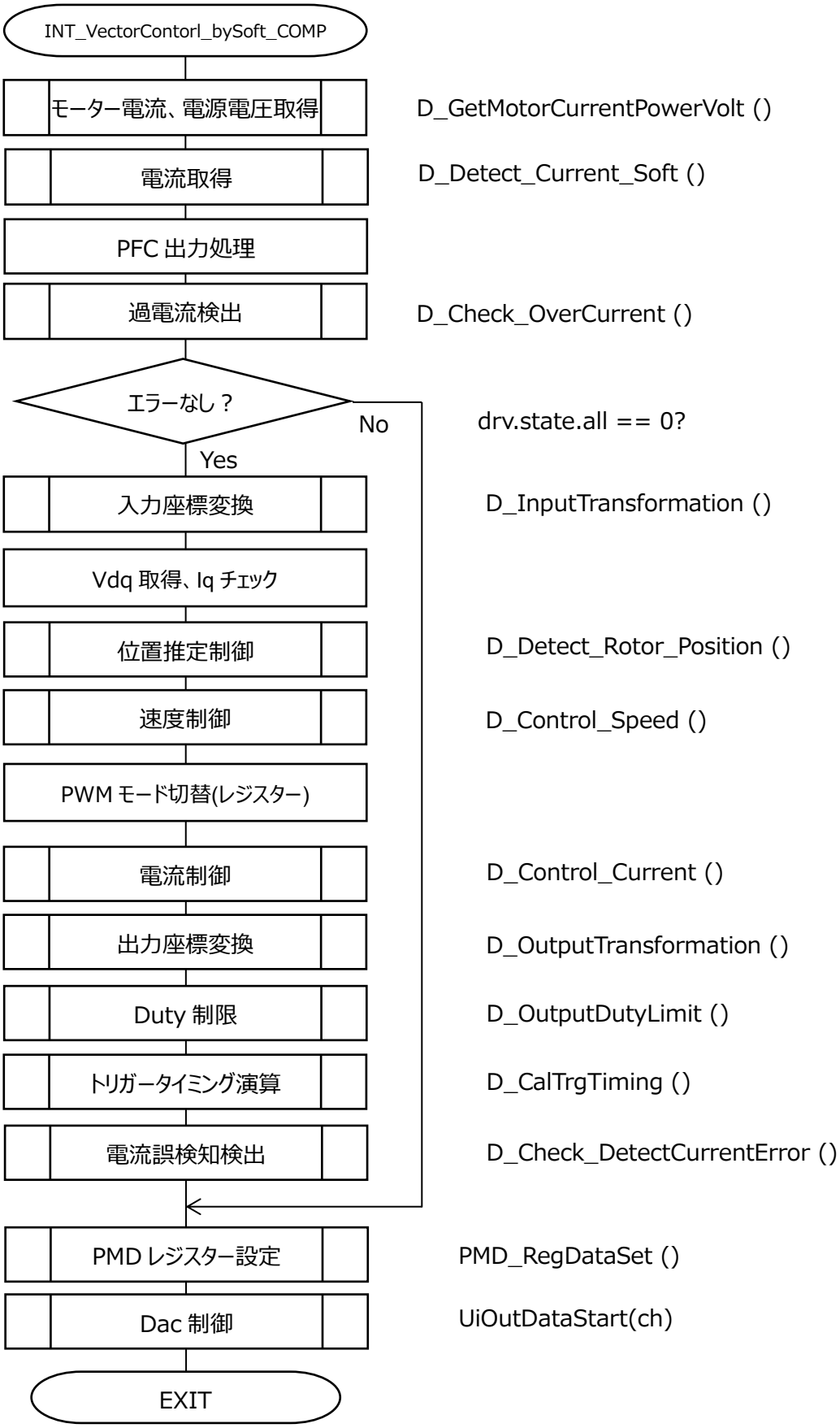
© 2024
Toshiba Electronic Devices & Storage Corporation

6.3.1 VE ベクトル処理(INT_VectorControlByVE_FAN)



© 2024
Toshiba Electronic Devices & Storage Corporation

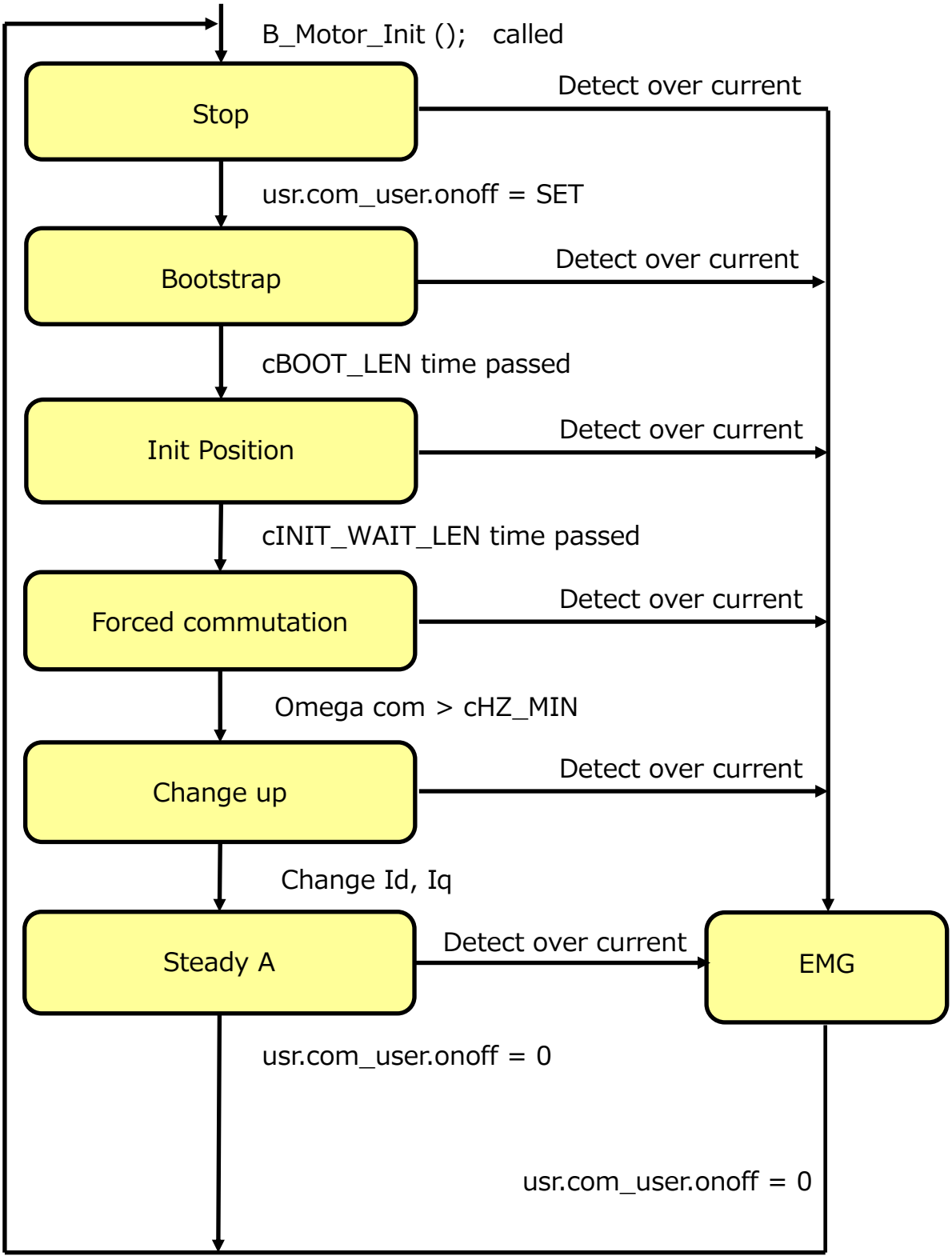
6.3.2 ソフトウェアベクトル処理（INT_VectorControlBySoft_COMP）



© 2024
Toshiba Electronic Devices & Storage Corporation

7. 状態遷移

7.1 センサーレス



© 2024
Toshiba Electronic Devices & Storage Corporation

8. 機能説明

アプリケーションとモーター制御間およびモーター制御とモーター駆動間のインターフェースを以下に記します。

8.1 制御コマンド

制御コマンドを以下に記します。

8.1.1 制御方法(usr.com_user)

・モーターの起動スタート、ストップ
・変調方式（２相変調）
・シフト PWM オン、オフ(1 シャント、2 相変調のときだけ有効)

```
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;  /* モーター起動指令       0=off, 1=on*/
} command_t;
```

command_t　com_user;
アプリケーションでは、usr.com_user が制御指令として設定されます。

8.1.2 制御目標速度

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

アプリケーションでは、usr.omega_user が制御目標速度として設定されます。

8.1.3 始動電流

```
q15_t     Id_st_user;         /* [A/maxA] d-軸起動電流指令、Q15 */
q15_t     Iq_st_user;         /* [A/maxA] q-軸起動電流指令、Q15 */
```

アプリケーションでは、usr.Id_st_user と usr.Iq_st_user が起動電流指令として設定されます。

© 2024
Toshiba Electronic Devices & Storage Corporation

8.2 駆動コマンド

駆動コマンドを以下に記します。

8.2.1 駆動方法(drv.command)

command_t command;

アプリケーションは、drv.command を介して、駆動方法をモーター制御ドライバー側へ渡します。

8.2.2 ベクトル制御コマンド(drv.vector_cmd)

ベクトル制御演算の指令コマンドで、ステージごとの処理を管理します。

```
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;
```

それぞれのステージではベクトル制御駆動コマンド（drv.vector_cmd）を以下のように設定し、モーター駆動に指令しています。

なお、F_vcomm_volt と F_vcomm_Idetect は使用しておりません。

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

ローター位置推定演算で、SET のときは推定値をローター位置とします。CLEAR では指令値をローター位置とします。

2) F_vcomm_omega

ローター位置推定演算で、SET のときは推定値を速度ωとします。CLEAR では指令値を速度ωとします。

3) F_vcomm_current

速度制御で、d、q 軸電流の基準値の算出方法を指令します。

© 2024
Toshiba Electronic Devices & Storage Corporation

SET のとき、速度偏差から PI 制御で求めた値を基準値とします。CLEAR では PI 制御を実行せず、指令値をそのまま基準値とします。

4) F_vcomm_Edetect
SET のとき、誘起電圧の演算を行い、ローター位置推定演算を行います。CLEAR のときは、誘起電圧の演算は行わず誘起電圧値を 0 とし、ローター位置は指令値とします

5) F_vcomm_Idetect
SET のとき、入力座標軸変換の演算を行います。CLEAR のときは、演算は行わず値をクリアします。

6) F_vcomm_onoff
SET のとき、モーター出力波形を ON します。CLEAR のときは、モーター出力波形を OFF します。

8.3 駆動状態

8.3.1 エラー状態(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_tall;
} state_t;
```

Loss_sync	脱調検出	脱調を検出したとき SET されます(未実装)
emg_DC	異常電圧検出	異常電圧を検出したとき SET されます。
emg_I	電流検出異常	電流検出の異常を検出したとき SET されます。(未使用)
emg_S	ソフト過電流検出	ソフト処理で過電流を検出したとき SET されます。
emg_H	ハード過電流検出	マイコンハード機能で過電流を検出したとき SET されます。
pfc_relate	PFC 異常検出	PFC 関連の異常を検出したとき SET されます。

© 2024
Toshiba Electronic Devices & Storage Corporation

8.4 モーター制御構造体

モーター制御構造体(vector_t)は、ipdefine.h に定義されています。変数は以下のようにモーターチャネルごとに宣言されます。

例)

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

8

型	名前	意味	Q フォーマット	備考
main_stage_e	stage.main	メインステージ	---	cStop cBootstrap cInitposition cForce cChange_up cSteady_A cEmergency
sub_stage_e	stage.sub	サブステージ	---	cStep0 cStep1 cStep2 cStep3 cStepEnd
itr_stage_e	stage.itr	割り込みステージ	---	ciStop ciBootstrap ciInitposition_i ciInitposition_v ciForce_i ciForce_v ciChange_up ciSteady_A ciEmergency
Uint8_t	Error_flag[2]	エラー情報	---	
q15_t	target_spd	目標速度	Q15	
q31_u	drv.omega_com	駆動速度指令値	Q31	
q31_u	drv.omega	推定速度	Q31	
q15_t	drv.omega_dev	速度偏差	Q15	
q31_u	drv.Id_com	d 軸電流指令値	Q31	
q31_u	drv.Iq_com	q 軸電流指令値	Q31	
q15_t	drv.Id_ref	d 軸電流基準値	Q15	
q15_t	drv.Iq_ref	q 軸電流基準値	Q15	
q15_t	drv.Id	d 軸電流	Q15	
q15_t	drv.Iq	q 軸電流	Q15	
q31_u	drv.Iq_ref_I	q 軸電流積分値	Q31	
uint16_t	drv.theta_com	電気角指令値	Q0	
uint32_u	drv.theta	ローター位置	Q0	
q15_t	drv.Vdc	電源電圧	Q15	
q15_t	drv.Vdc_temp	前回電源電圧	Q15	
q31_u	drv.Vdc_ave	(未使用)	---	
q31_u	drv.Vd	d 軸電圧	Q31	

© 2024
Toshiba Electronic Devices & Storage Corporation

q31_u	drv.Vq	q 軸電圧	Q31	
q15_t	drv.Vdq	(未使用)	---	
q31_u	drv.Vdq_ave	(未使用)	---	
q31_t	drv.Vd_out	出力電圧値	Q31	
q15_t	drv.Ed	d 軸誘起電圧	Q15	
q15_t	drv.Eq	(未使用)	---	
q31_t	drv.Ed_I	d 軸誘起電圧積分値	Q31	
q31_t	drv.Ed_PI	d 軸誘起電圧 PI 値	Q31	
state_t	drv.state	モーター異常ステータス	---	
command_t	drv.command	駆動方法	---	(8.2.1 参照)
vectorcmd_t	drv.vector_cmd	ベクトル制御コマンド	---	(8.2.2 参照)
uint16_t	drv.chkpls	電流検出保護 Duty 幅	Q0	
uint8_t	drv.idetect_error	電流検出状態	---	0:検出可能 1:検出不能
q15_t	drv.Ia_raw	a 相電流(生データ)	Q15	
q15_t	drv.Ib_raw	b 相電流(生データ)	Q15	
q15_t	drv.Ic_raw	c 相電流(生データ)	Q15	
q15_t	drv.Ia	a 相電流(誤検知保護)	Q15	
q15_t	drv.Ib	b 相電流(誤検知保護)	Q15	
q15_t	drv.Ic	c 相電流(誤検知保護)	Q15	
q31_u	drv.Iao_ave	a 相ゼロ電流平均 AD 値	Q0	
q31_u	drv.Ibo_ave	b 相ゼロ電流平均 AD 値	Q0	
q31_u	drv.Ico_ave	c 相ゼロ電流平均 AD 値	Q0	
uint8_t	drv.spdprd	速度制御周期	Q0	
q31_u	usr.omega_user	制御目標速度	Q31	
q15_t	usr.Id_st_user	始動 Id 電流値	Q15	
q15_t	usr.Iq_st_user	始動 Iq 電流値	Q15	
uint16_t	usr.lambda_user	初期ローター位置	Q0	
command_t	usr.com_user	制御方法	---	(8.1.1 参照)
command_t	usr.com_user_1	制御方法前回	---	
uint16_t	para.V_max	最大電圧	---	
uint16_t	para.I_max	最大電流	---	
uint16_t	para.Hz_max	最大速度	---	
q15_t	para.pos_degadj	初期位置	---	
q15_t	para.omega_min	強制転流終了速度	Q15	
q15_t	para.omega_v2i	電流制御切替速度	Q15	
q15_t	para.spwm_threshold	シフト PWM 切替速度	Q15	
q31_t	para.vd_pos	位置決め指令電圧	Q31	
q31_t	para.spd_coef	出力電圧係数	Q15	
q31_u	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
q31_u	para.sp_up_lim_s	駆動速度増加リミット値	Q31	
q31_u	para.sp_dn_lim_s	駆動速度減少リミット値	Q31	
uint16_t	para.time.initpos	位置決め時間	Q0	
uint16_t	para.time.initpos2	位置決め状態待ち時間	Q0	
uint16_t	para.time.bootstp	ブートストラップ時間	Q0	
uint16_t	para.time.go_up	強制定常切換後待ち時間	Q0	
q31_t	para.iq_lim	q 軸電流制限値	Q31	
q31_t	para.id_lim	d 軸電流制限値	Q31	
q15_t	para.err_ovc	過電流設定値	Q15	
q31_t	para.pos.kp	位置推定比例ゲイン	Q15	
q31_t	para.pos.ki	位置推定積分ゲイン	Q15	
int32_t	para.pos.ctrlprd	位置推定制御周期	Q16	
q31_t	para.spd.kp	速度制御比例ゲイン	Q15	
q31_t	para.spd.ki	速度制御積分ゲイン	Q15	
uint8_t	para.spd.pi_prd	(未使用)	Q0	
q31_t	para.crt.dkp	d 軸電流制御比例ゲイン	Q15	

© 2024
Toshiba Electronic Devices & Storage Corporation

q31_t	para.crt.dki	d 軸電流制御積分ゲイン	Q15	
q31_t	para.crt.qkp	q 軸電流制御比例ゲイン	Q15	
q31_t	para.crt.qki	q 軸電流制御積分ゲイン	Q15	
q31_t	para.motor.r	モーター巻線抵抗	Q15	
q31_t	para.motor.Lq	モーターq 軸インダクタンス	Q15	
q31_t	para.motor.Ld	モーターd 軸インダクタンス	Q15	
int32_t	para.delta_lambda	強制定常切換時電流切替位相	Q0	
uint16_t	para.chkpls	電流検出保護 Duty 幅	Q0	
uint32_t	stage_counter	ステージカウンター	Q0	
shunt_type_e	shunt_type	シャントタイプ	---	c1shunt
boot_type_e	boot_type	起動タイプ	---	cBoot_v

ソフトウェアによるベクトル制御用

型	名前	意味	Q フォーマット	備考
uint32_t*	drv.ADxREG0	AD 変換結果レジスターアドレス	---	
uint32_t*	drv.ADxREG1	AD 変換結果レジスターアドレス	---	
uint32_t*	drv.ADxREG2	AD 変換結果レジスターアドレス	---	
uint32_t*	drv.ADxREG3	AD 変換結果レジスターアドレス	---	
q15_t	drv.Vdc_adc	電源電圧変換結果	Q15	
uint16_t	drv.Ia_adc	U 相電流変換結果	Q15	
uint16_t	drv.Ib_adc	V 相電流変換結果	Q15	
uint16_t	drv.Ic_adc	W 相電流変換結果	Q15	
q15_t	drv.Idc1_adc	1st 相電流変換結果	Q15	
q15_t	drv.Idc2_adc	2nd 相電流変換結果	Q15	
q31_u	drv.Idco_ave	ゼロ電流平均 AD 値	Q0	
q15_t	drv.Ialpha	α軸電流	Q15	
q15_t	drv.Ibeta	β軸電流	Q15	
q15_t	drv.Id_dev	d 軸電流偏差	Q15	
q15_t	drv.Iq_dev	q 軸電流偏差	Q15	
q31_u	drv.Vd_com	d 軸電圧指令値	Q31	
q31_u	drv.Vq_com	q 軸電圧指令値	Q31	
q31_u	drv.Vd_I	d 軸電圧積分値	Q31	
q31_u	drv.Vq_I	q 軸電圧積分値	Q31	
q31_u	drv.Valpha	α軸電圧	Q31	
q31_u	drv.Vbeta	β軸電圧	Q31	
q15_t	drv.DutyU	U 相 PWM Duty 比	Q15	
q15_t	drv.DutyV	V 相 PWM Duty 比	Q15	
q15_t	drv.DutyW	W 相 PWM Duty 比	Q15	
int32_t	drv.AdTrg0	AD トリガータイミング 0 位置	Q15	
int32_t	drv.AdTrg1	AD トリガータイミング 1 位置	Q15	
uint8_t	drv.Sector0	セクター	Q0	0 ～ 5
uint8_t	drv.Sector1	前回セクター	Q0	0 ～ 5

© 2024
Toshiba Electronic Devices & Storage Corporation

8.5 関数詳細

8.5.1 ADC 初期設定(init_ADCen)

8.5.1.1 構文

void init_ADCen(void)

引数：

なし

戻り値：

なし

8.5.1.2 処理内容

ADC の初期設定を行います。

・モーター用 AD 設定

・ADC 許可

8.5.2 PMD 初期設定(init_PMDen)

8.5.2.1 構文

void init_PMDen(void)

引数：

なし

戻り値：

なし

8.5.2.2 処理内容

PMD(プログラマブルモータードライバー)の初期設定を行います。

8.5.3 VE 初期設定(init_VEen)

8.5.3.1 構文

void init_VEen(void)

引数：

なし

戻り値：

なし

8.5.3.2 処理内容

VE(ベクトルエンジン)の初期設定を行います。

・VE 割り込み設定

・VE 設定

© 2024
Toshiba Electronic Devices & Storage Corporation

8.5.4 モーター制御初期設定(B_Motor_Init)

8.5.4.1 **構文**

void B_Motor_Init(void)

引数：

なし

戻り値：

なし

8.5.4.2 **変数**

方向	名前	意味	Q フォーマット	備考
出力	shunt_type	シャントタイプ	---	
	boot_type	駆動タイプ	---	
	usr.com_user.encoder	エンコーダー制御	---	
	stage.main	メインステージ	---	
	stage.sub	サブステージ	---	
	drv.Iao_ave	U 相ゼロ電流平均 AD 値	Q0	
	drv.Ibo_ave	V 相ゼロ電流平均 AD 値	Q0	
	drv.Ico_ave	W 相ゼロ電流平均 AD 値	Q0	
	drv.Idco_ave	ゼロ電流変換結果	Q15	1 シャント時だけ
	usr.Iq_st_user	始動 Iq 電流値	Q15	
	usr.Id_st_user	始動 Id 電流値	Q15	
	usr.lambda_user	初期ローター位置	Q0	
	para.motor.r	モーター巻線抵抗	Q15	
	para.motor.Lq	モーターq 軸インダクタンス	Q15	
	para.motor.Ld	モーターd 軸インダクタンス	Q15	
	para.spwm_threshold	シフト PWM 切換速度	Q15	
	para.chkpls	電流検出保護 Duty 幅	Q0	
	para.vd_pos	位置決め指令電圧	Q31	電圧起動時だけ
	para.spd_coef	出力電圧係数	Q15	電圧起動時だけ
	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
	para.sp_up_lim_s	駆動速度増加リミット値	Q31	
	para.sp_dn_lim_s	駆動速度減少リミット値	Q31	
	para.time.bootstp	ブートストラップ時間	Q0	
	para.time.initpos	位置決め時間	Q0	
	para.time.initpos2	位置決め状態待ち時間	Q0	
	para.time.go_up	強制定常切換後待ち時間	Q0	
	para.omega_min	強制転流終了速度	Q15	
	para.omega_v2i	電流制御切換速度	Q15	
	para.delta_lambda	強制定常切換時電流切換位相	Q0	
	para.pos.ki	位置推定積分ゲイン	Q15	Q12 対応あり
	para.pos.kp	位置推定比例ゲイン	Q15	Q12 対応あり
	para.pos.ctrlprd	位置推定制御周期	Q16	
	para.spd.ki	速度制御積分ゲイン	Q15	Q12 対応あり
	para.spd.kp	速度制御比例ゲイン	Q15	Q12 対応あり
	para.current.dki	d 軸電流比例ゲイン	Q15	Q12 対応あり
	para.current.dkp	d 軸電流積分ゲイン	Q15	Q12 対応あり
	para.current.qki	q 軸電流比例ゲイン	Q15	Q12 対応あり
	para.current.qkp	q 軸電流積分ゲイン	Q15	Q12 対応あり
	para.iq_lim	q 軸電流制限値	Q31	

© 2024
Toshiba Electronic Devices & Storage Corporation

	para.id_lim	d 軸電流制限値	Q31	
	para.err_ovc	過電流設定値	Q15	
	usr.com_user.modul	変調方式制御コマンド	---	2 相変調
	para.TrgPosMd	電流検出位置モード	---	
	para.TrgComp	トリガータイミング補正値	Q15	

8.5.4.3 **処理内容**

モーター制御引数：の初期化を行います。
制御中に設定変更を行わない変数設定を行います。

8.5.5 DAC 制御初期設定(init_Dac)

8.5.5.1 **構文**

void init_Dac(TSB_SC_TypeDef* const SCx)
引数：
TSB_SC_TypeDef* const SCx：SC アドレスを選択します。
戻り値：
なし

8.5.5.2 **処理内容**

DAC IC 制御の初期設定を行います。
SIO 許可
ポート初期設定
SIO 初期設定
DAC IC 初期化通信
SIO 割り込みレベル設定
SIO 割り込み保留クリア
送信割り込み許可

8.5.6 周期タイマー初期設定(init_Timer_interval4kHz)

8.5.6.1 **構文**

void init_Timer_interval4kHz(void)
引数：
なし
戻り値：
なし

8.5.6.2 **処理内容**

4kHz 周期タイミングを作成するためのタイマーの初期設定を行います。

© 2024
Toshiba Electronic Devices & Storage Corporation

8.5.7 PFC 制御初期設定 (init_HPFC_Control)

8.5.7.1 構文

void init_HPFC_Control(void)

引数：

なし

戻り値：

なし

8.5.7.2 処理内容

PFC を制御するための初期設定を行います。

PWM、ADC など

8.5.8 UART 初期設定(init_uart)

8.5.8.1 構文

void init_uart(void)

引数：

なし

戻り値：

なし

8.5.8.2 処理内容

UART の初期設定を行います。

8.5.9 ユーザー制御(uart_control)

8.5.9.1 構文

void user_control(void)

引数：

なし

戻り値：

なし

8.5.9.2 処理内容

ユーザーなど外部からの指令を制御します。

・回転数増減

・制御モーターの切り替え(ファン／コンプレッサー)

・ファンモーター、コンプレッサーモーター、PFC の停止

・情報取得の切り替え(ファンモーター／コンプレッサーモーター／PFC)

・DAC モードの切り替え

© 2024
Toshiba Electronic Devices & Storage Corporation

8.5.10 ユーザーモーター制御(B_User_MotorControl)

8.5.10.1 構文

void B_User_MotorControl(void)

引数：

なし

戻り値：

なし

8.5.10.2 変数

方向	名前	意味	Q フォーマット	備考
入力	target_spd	目標速度	---	ユーザー指令
	keydata.sw	スイッチ ON,OFF 状態	---	ユーザー指令
出力	usr.omega_user	制御目標速度	Q31	
	usr.com_user.onoff	モーター ON/OFF	---	
	drv.state	モーター異常ステータス	---	

8.5.10.3 処理内容

モーター関連に対するユーザーからの指令を処理します。

・過電流(ハードウェア)状態のチェック

ハードウェア過電流の状態をチェックし、モーター異常ステータスを更新します。

・モーターON/OFF 指令

目標速度(target_spd1)の状態からモーターON/OFF(usr.com_user.onoff)を決定します。

モーターOFF 指令の時に、モーター異常ステータスをクリアします。

・駆動速度正規化など

目標速度を正規化します。

© 2024
Toshiba Electronic Devices & Storage Corporation

8.6 モーター制御関数

モーター制御処理は main 関数のメインループ内からコールされる以下の関数によって実現されます。
モーター動作を、停止状態、ブートストラップ状態、位置決め状態、強制転流状態、強制定常切換状態、定常状態、短絡ブレーキ状態、保護状態間の状態遷移として制御します。
メインステージはモーター動作開始指示で位置決め((Initposition)、強制転流(Force)、強制定常切換(Change_up)、定常(Steady_A)の順に移行します。サブステージは Step0 から StepEnd まであり、StepEnd の時にメインステージが移行し Step0 から始まります。また、モーターの異常を検出した場合は保護停止 Emergency に移行します。

8.6.1 状態遷移処理関数(C_Control_Ref_Model_0, C_Control_Ref_Model_1)

8.6.1.1 構文

void C_Control_Ref_Model(vector_t* const _motor)
引数：
　vector_t* const _motor　：　モーター制御構造体
戻り値：
　なし

8.6.1.2 変数

方向	名前	意味	Q フォーマット	備考
入力	usr.com_user.onoff	制御指令	---	
	drv.state.all	エラー状態	---	
入出力	stage.main	メインステージ	---	
	stage.sub	サブステージ	---	
	usr.com_user_1.onoff	前回制御指令	---	

8.6.1.3 処理内容

アプリケーションから与えられる制御コマンドおよび現在の状態を監視し、状態遷移を実行します。
各状態はさらに詳細化されたサブ状態に分かれます。サブ状態の遷移は状態遷移処理関数ではなく、各状態の処理関数内で実行されます。

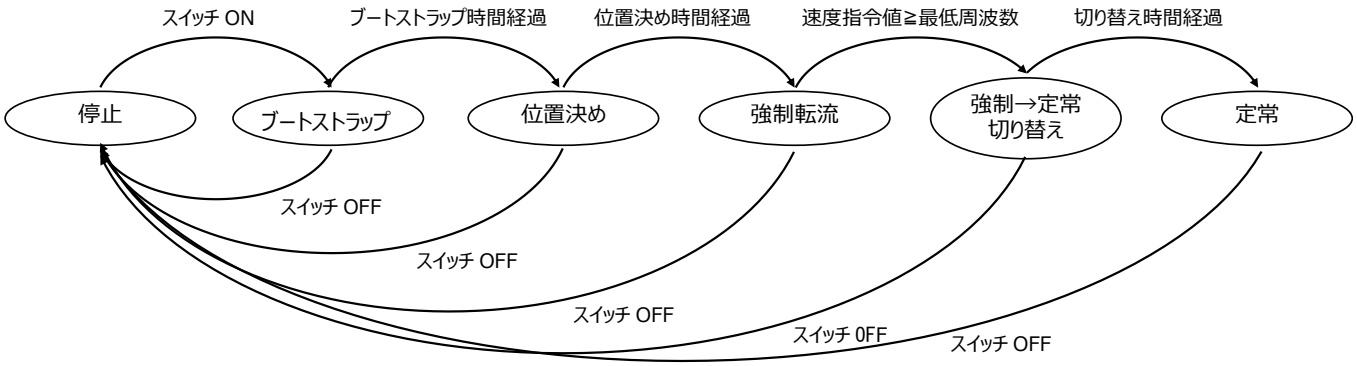


図 4 モーター制御の状態遷移図

© 2024
Toshiba Electronic Devices & Storage Corporation

8.6.2 モーター制御共通処理関数(C_Common)

8.6.2.1 構文

void C_Common(vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.6.2.2 変数

方向	名前	意味	Q フォーマット	備考
入力	usr.com_user.modul	変調方式制御コマンド	---	2 相 or 3 相変調
	para.chkpls	電流検出保護 Duty 幅設定値	---	
出力	drv.command.modul	変調方式駆動コマンド	---	2 相 or 3 相変調
	drv.chkpls	電流検出保護 Duty 幅	---	

8.6.2.3 処理内容

モーター制御の各状態に共通な処理を実行します。

シフト PWM 制御(8.6.10 C_ShiftPWM_Control)の算出を行います

8.6.3 停止状態関数(C_Stage_Stop)

8.6.3.1 構文

void C_Stage_Stop(vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.6.3.2 変数

方向	名前	意味	Q フォーマット	備考
入力	stage.main	メインステージ	---	
	motor_type	駆動モーター	---	
入出力	stage.sub	サブステージ	---	
出力	stage.itr	割り込みステージ	---	
	drv.vector_cmd	ベクトル制御コマンド	---	(8.2.2 参照)
	drv.theta_com	電気角指令値	Q0	
	drv.theta	ローター位置	Q0	
	drv.omega_com	駆動速度指令値	Q31	
	drv.omega	速度	Q31	
	drv.idetect_error	電流検出状態	---	
	drv.Vd_com	d 軸電圧指令値	Q31	ソフトウェアベクトル制御のみ
	drv.Vq_com	q 軸電圧指令値	Q31	ソフトウェアベクトル制御のみ
	drv.Id_com	d 軸電流指令値	Q31	

© 2024
Toshiba Electronic Devices & Storage Corporation

	drv.Iq_com	q 軸電流指令値	Q31	

8.6.3.3 処理内容

モーターを停止させます。(PWM 出力を停止します)

8.6.4 ブートストラップ状態関数(C_Stage_Bootstrap)

8.6.4.1 構文

void C_Stage_Bootstrap(vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.6.4.2 変数

方向	名前	意味	Q フォーマット	備考
入力	stage.main	メインステージ	---	
	para.time.bootstp	ブートストラップ時間	Q0	* MainLoopPrd(s)
入出力	stage.sub	サブステージ	---	
	stage_counter	ステージカウンター	Q0	* MainLoopPrd(s)
出力	stage.itr	割り込みステージ	---	
	drv.vector_cmd	ベクトル制御コマンド	---	(8.2.2 参照)

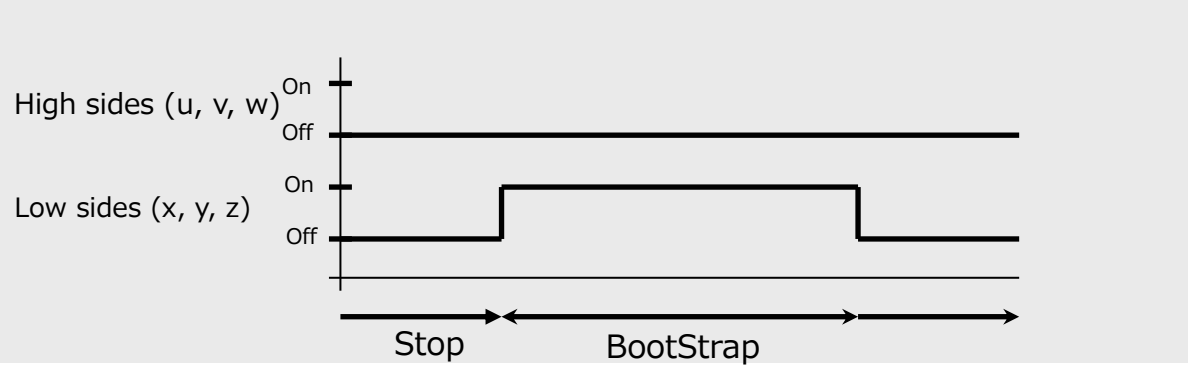
8.6.4.3 処理内容

上相 All OFF、下相 All ON の波形を出力し、ブートストラップコンデンサーの充電を行います。
この処理を「ブートストラップ時間」継続させます。「ブートストラップ時間」からコンデンサーの充電量を決定します。

ブートストラップ状態を以下のサブ状態に分けて制御します。

a) 初期状態
ブートストラップ状態の初期設定を行います。

b) 時間経過待ち状態
指定されたブートストラップ時間が経過するのを待ち、位置決め状態へ遷移します。



© 2024
Toshiba Electronic Devices & Storage Corporation

8.6.5 位置決め状態関数(C_Stage_Initposition_0, C_Stage_Initposition_1)

8.6.5.1 構文

void C_Stage_Initposition(vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.6.5.2 変数

方向	名前	意味	Q フォーマット	備考
入力	stage.main	メインステージ	---	
	boot_type	起動タイプ	---	
	usr.Id_st_user	始動 Id 電流値	Q15	
	usr.lambda_user	初期ローター位置	Q0	
	para.vd_pos	位置決め指令電圧	Q31	
	para.time.initpos	位置決め時間	Q0	* MainLoopPrd(s)
	para.time.initpos2	位置決め状態待ち時間	Q0	* MainLoopPrd(s)
入出力	stage.sub	サブステージ	---	
	stage_counter	ステージカウンター	Q0	* MainLoopPrd(s)
	drv.Vd_out	出力電圧値	Q31	電圧駆動時だけ有効
	drv.Id_com	d 軸電流指令値	Q31	
出力	stage.itr	割り込みステージ	---	
	drv.vector_cmd	ベクトル制御コマンド	---	(8.2.2 参照)
	drv.Iq_com	q 軸電流指令値	Q31	
	drv.omega_com	駆動速度指令値	Q31	
	drv.theta_com	電気角指令値	Q0	

8.6.5.3 処理内容

ローターを初期位置に固定させます。

θを「初期位置」に固定、ωを 0 に固定、Iq を 0 に固定しながら、Id を 0 から徐々に増加させていきます。

この処理を「位置決め時間」継続させ、最終的に Id が「始動 Id 電流値」になります。「位置決め時間」と「始動 Id 電流値」から単位時間当たりの Id の増加量を決定します。

Id が「始動 Id 電流値」到達後、[位置決め待ち時間]経過後、次ステージに遷移します。

位置決め状態を以下のサブ状態に分けて制御します。

a) 初期状態
位置決め状態の初期設定を行います。

b) Id 増加状態
Id を設定値まで徐々に増加させます。

8.6.6 強制転流状態関数(C_Stage_Force_0, C_Stage_Force_1)

8.6.6.1 構文

© 2024
Toshiba Electronic Devices & Storage Corporation

void C_Stage_Force(vector_t* const _motor)

引数：

vector_t* const _motor　：　モーター制御構造体

戻り値：

なし

8.6.6.2 変数

方向	名前	意味	Q フォーマット	備考
入力	stage.main	メインステージ	---	
	boot_type	起動タイプ	---	電流 or 電圧
	usr.Id_st_user	始動 Id 電流値	Q15	
	usr.omega_user	制御目標速度	Q31	
	para.omega_v2i	電流制御切換速度	Q15	電圧駆動時だけ有効
	para.spd_coef	出力電圧係数	Q15	電圧駆動時だけ有効
	para.vd_pos	位置決め出力電圧	Q31	電圧駆動時だけ有効
	para.omega_min	強制転流終了速度	Q15	
	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
入出力	stage.sub	サブステージ	---	
	drv.omega_com	駆動速度指令値	Q31	
出力	drv.vector_cmd	ベクトル制御コマンド	---	(8.2.2 参照)
	stage.itr	割り込みステージ	---	
	drv.Iq_com	q 軸電流指令値	Q31	
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Vd_out	出力電圧値	Q31	電圧駆動時だけ有効

8.6.6.3 処理内容

ローターの回転を開始します。このステージではベクトル制御によるフィードバック処理ではなく、強制的に回転磁界を与えて、ローターがそれに追随して回転します。

Id を「始動 Id 電流値」に、Iq を 0 に固定しながら、駆動速度指令値ω_{com}を徐々に増加させます。

θはω_{com}から求めます($\theta=\omega_{com}t$)。この処理をωが「強制転流終了速度」に達するまで継続します。

「駆動目標速度」を一定値ずつ増加させて「制御目標速度」に近づけます。「駆動目標速度」がωになります。

8.6.7 強制定常切換状態関数(C_Stage_Change_up)

8.6.7.1 構文

void C_Stage_Change_up(vector_t* const _motor)

引数：

vector_t* const _motor　：　モーター制御構造体

戻り値：

なし

8.6.7.2 変数

方向	名前	意味	Q フォーマット	備考
入力	stage.main	メインステージ	---	

© 2024
Toshiba Electronic Devices & Storage Corporation

	usr.Id_st_user	始動 Id 電流値	Q15	
	usr.Iq_st_user	始動 Iq 電流値	Q15	
	usr.omega_user	制御目標速度	Q31	
	para.delta_lambda	強制定常切換時電流切換位相	Q0	
	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
	para.time.go_up	強制定常切換後待ち時間	Q0	* MainLoopPrd(s)
入出力	stage.sub	サブステージ	---	
	stage_counter	ステージカウンター	Q0	* MainLoopPrd(s)
	drv.omega_com	駆動速度指令値	Q31	
出力	stage.itr	割り込みステージ	---	
	drv.vector_cmd	ベクトル制御コマンド	---	(8.2.2 参照)
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	

8.6.7.3 **処理内容**

Id を 0 に減少、Iq を「始動 Iq 電流」まで増加させ、磁界の方向がローターと直下になるように制御します。
トルク成分を発生させます。
ω、θは位置推定演算により求めます。
「駆動目標速度」を一定値ずつ増加させて「制御目標速度」に近づけます。ただしこのステージでは速度制御は行っていないため、「駆動目標速度」は制御には使用されません。

強制定常切換え状態を以下のサブ状態に分けて制御します。

a) 初期状態
強制定常切換え状態の初期設定を行います。

b) Id、Iq 切換え状態
Id を 0 まで徐々に減少させ、同時に Iq を指定値まで徐々に増加させます。増加および減少の曲線は線形ではなく、三角関数曲線によります。切り替え完了後、時間経過待ち状態へ遷移します。

c) 時間経過待ち状態
指定された強制定常切換え時間が経過するのを待ち、定常状態へ遷移します。

8.6.8 定常状態関数(C_Stage_Steady_A)

8.6.8.1 **構文**

void C_Stage_Steady_A(vector_t* const _motor)
引数：
vector_t* const _motor ： モーター制御構造体
戻り値：
なし

8.6.8.2 **変数**

方向	名前	意味	Q フォーマット	備考
入力	stage.main	メインステージ	---	
	usr.omega_user	制御目標速度	Q31	
	para.sp_up_lim_s	駆動速度増加リミット値	Q31	

© 2024
Toshiba Electronic Devices & Storage Corporation

	para.sp_dn_lim_s	駆動速度減少リミット値	Q31	
入出力	stage.sub	サブステージ	---	
	drv.omega_com	駆動速度指令値	Q31	
出力	drv.vector_cmd	ベクトル制御コマンド	Q0	(8.2.2 参照)
	stage.itr	割り込みステージ	---	
	drv.Id_com	d 軸電流指令値	Q31	

8.6.8.3 処理内容

定常状態の処理を実行します。

「駆動目標速度」を一定ずつ増加させて「制御目標速度」に近づけます。

8.6.9 保護状態関数(C_Stage_Emergency)

8.6.9.1 構文

void C_Stage_Emergency(vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.6.9.2 変数

方向	名前	意味	Q フォーマット	備考
入力	stage.main	メインステージ	---	
入出力	stage.sub	サブステージ	---	
出力	drv.vector_cmd	ベクトル制御コマンド	---	(8.2.2 参照)
	stage.itr	割り込みステージ	---	

8.6.9.3 処理内容

過電流が発生したとき、この状態へ遷移します。

ハードウェア過電流を検出したときは、モーター駆動出力 u、v、w、x、y、z は全て Hi-z となっています。

ソフトウェア過電流を検出したときは、モーター駆動出力 u、v、w、x、y、z は全て OFF となっています。

ローターは惰性で回転します。過電流状態復帰処理を実施するまでこのステージを維持します。

8.6.10 シフト PWM 制御 (C_ShiftPWM_Control)

8.6.10.1 構文

static void C_ShiftPWM_Control(vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

© 2024
Toshiba Electronic Devices & Storage Corporation

8.6.10.2 変数

方向	名前	意味	Q フォーマット	備考
入力	shunt_type	シャントタイプ	---	
	stage.itr	割り込みステージ	---	
	usr.com_user.spwm	シフト PWM 制御方法	---	
	drv.omega_com	駆動速度指令値	Q31	
	para.spwm_threshold	シフト PWM 切換速度	Q15	
出力	drv.command.spwm	シフト PWM 駆動コマンド	---	

8.6.10.3 処理内容

この制御は VE 使用かつ 1 シャントのときだけ有効です。

シフト PWM の ON/OFF の設定を行います。

シフト PWM 制御方法、割り込みステージ、目標速度から、シフト PWM 駆動方法を決定します。サンプルソフトでは、**エラー! 参照元が見つかりません。**の条件設定となっています。

表 1　シフト PWM 駆動方法条件

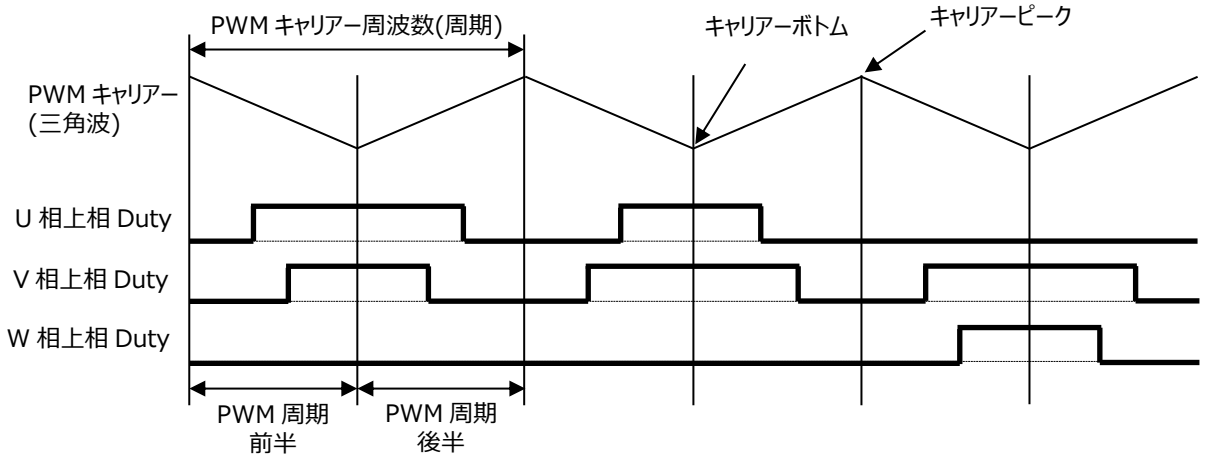
シフト PWM 制御方法 usr.com_user.spwm	割り込みステージ stage.itr	目標速度 drv.omega_com	シフト PWM 駆動方法 drv.command.spwm
OFF (0)	All	All	OFF (0)
ON　(1)	Stop	All	OFF (0)
	Emergency	All	OFF (0)
	ciInitposition_i Force_i Change_up Steady_A	目標速度 >= シフト切換速度 drv.omega_com >= para.spwm_threshold	OFF (0)
		目標速度 < シフト切換速度 drv.omega_com < para.spwm_threshold	ON (1)

© 2024
Toshiba Electronic Devices & Storage Corporation

8.7 モーター駆動関数

8.7.1 用語説明

8.7.1.1 2相変調の場合



8.7.2 モーター電流、電源電圧取得（D_GetMotorCurrentPowerVolt)

8.7.2.1 構文

void D_GetMotorCurrentPowerVolt (vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.7.2.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.Sector1	前回セクター	Q0	0 ～ 5
	para.TrgPosMd	電流検出位置モード	---	1 シャント前半 1 シャント後半
	drv.ADxREG0	AD 変換結果レジスターアドレス	---	
	drv.ADxREG1	AD 変換結果レジスターアドレス	---	
	drv.ADxREG2	AD 変換結果レジスターアドレス	---	
	drv.ADxREG3	AD 変換結果レジスターアドレス	---	
	drv.Iao_ave	U 相ゼロ電流平均	Q15	3 シャント時だけ
	drv.Ibo_ave	V 相ゼロ電流平均	Q15	3 シャント時だけ
	drv.Ico_ave	W 相ゼロ電流平均	Q15	3 シャント時だけ
	drv.Idco_ave	ゼロ電流平均	Q15	1 シャント時だけ
	stage.itr	割り込みステージ	---	
	drv.idetect_error	電流検出状態	---	
	drv.shift2_mode	シフト 2 制御モード	---	
入出力	drv.Ia_raw	U 相電流(生データ)	Q15	
	drv.Ib_raw	V 相電流(生データ)	Q15	
	drv.Ic_raw	W 相電流(生データ)	Q15	
出力	drv.Ia	U 相電流(誤検知保護)	Q15	
	drv.Ib	V 相電流(誤検知保護)	Q15	

© 2024
Toshiba Electronic Devices & Storage Corporation

	drv.Ic	W 相電流(誤検知保護)	Q15	
	drv.Ia_adc	U 相電流変換結果	Q15	3 シャント時だけ
	drv.Ib_adc	V 相電流変換結果	Q15	3 シャント時だけ
	drv.Ic_adc	W 相電流変換結果	Q15	3 シャント時だけ
	drv.Idc1_adc	1st 相電流変換結果	Q15	1 シャント時だけ
	drv.Idc2_adc	2nd 相電流変換結果	Q15	1 シャント時だけ
	drv.Vdc	電源電圧	Q15	
	drv.Vdc_adc	電源電圧変換結果	Q15	

8.7.2.3 **処理内容**

AD 変換結果値を取得し、電源電圧とモーター電流を演算します。

1．PMD トリガーによる AD 変換の終了を確認し、変換が終了していれば下記表に従って変換結果を取得する。
変換終了していない場合は、変換が終わるまでループで待つ(通常ではあり得ないがフェールセーフ処理で待つ処理を行う)。
ただし、シフト PWM2 で動作している場合はオフセットを加えて算出し、
通常 PWM/シフト PWM2 の切り替えたタイミングでは取得・演算は行いわない。

2．取得した AD 変換結果から電源電圧、モーター電流を演算する。

- 電源電圧

電源電圧変換結果 drv.Vdc_adc	AD 変換結果 3 ADxREG3

電源電圧は符号なしのため、AD 変換結果を 1bit 右シフトして Q15 フォーマットの変数とする。

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

- モーター電流

<電流検出モード：1 シャント前半、1 シャント後半のとき>

電流検出モード：1 シャント前半のときは、PWM 周期前半の位置で AD 変換結果を取得
電流検出モード：1 シャント後半のときは、PWM 周期後半の位置で AD 変換結果を取得

電流取得位置 / 変換結果	PWM 周期後半	PWM 周期前半
電流変換結果 1 drv.Idc1_adc	AD 変換結果 0 ADREG0	AD 変換結果 1 ADREG 1
電流変換結果 2 drv.Idc2_adc	AD 変換結果 1 ADREG1	AD 変換結果 0 ADREG0

モーター停止時のＡＤ変換結果をゼロ電流として、下記変数に格納する。

ゼロ電流変換結果 drv.Idco_ave	電流変換結果 1 drv.Idc1_adc

3 相電流の値は、ゼロ電流(モーター停止)時の AD 変換結果「ゼロ電流変換結果」と 2 つのタイミング時の AD 変換結果から下記表に従って演算する。

	セクター					
	0	1	2	3	4	5
U 相電流 drv.Ia	-(Idco_ave-Idc2_adc)	-Ib-Ic	Idco_ave-Idc1_adc	Idco_ave-Idc1_adc	-Ib-Ic	-(Idco_ave-Idc2_adc)
V 相電流	-Ia-Ic	-(Idco_ave-	-(Idco_ave-	-Ia-Ic	Idco_ave-	Idco_ave-

© 2024
Toshiba Electronic Devices & Storage Corporation

drv.Ib		Idc2_adc)	Idc2_adc)		Idc1_adc	Idc1_adc
W 相電流 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 入力座標変換（D_InputTransformation)

8.7.3.1 構文

void D_InputTransformation(vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.7.3.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_Idetect	電流検出コマンド	---	
	drv.Ia	U 相電流	Q15	
	drv.Ib	V 相電流	Q15	
	(drv.Ic)	W 相電流	Q15	
	drv.theta	ローター位置	Q0	[deg/360]
	para.pos.ctrlprd	ベクトル制御周期	Q0	
	drv.omega	推定速度	Q15	
	drv.idetect_error	電流検出状態	---	0:検出可能 1:検出不能
	drv.shift2_mode	シフト 2 制御モード	---	
出力	drv.Ialpha	α軸電流	Q15	
	drv.Ibeta	β軸電流	Q15	
	drv.Id	d 軸電流	Q15	
	drv.Iq	q 軸電流	Q15	

8.7.3.3 処理内容

3 相の電流(Ia, Ib, Ic)から、dq 軸電流に変換します。

1．8.7.4 クラーク変換により、3 相の電流(Ia, Ib, Ic)から 2 相の電流(Iα,Iβ)に変換します。
2．8.7.5 パーク変換により、静止座標である 2 相の電流(Iα,Iβ)から回転座標の dq 軸変換に変換します。
パーク変換に使用するローター位置は、位置推定処理で演算した時点から実際のローター位置は進んでいるため、速度ωと演算周期から予測したθest を使用します。 θest = θ ＋ 速度ω×ベクトル制御周期

なお、電流検出が正常にできない状態のとき、または通常 PWM とシフトの切換のときは、変換を行わず、Id,Iq は更新しません。

8.7.4 クラーク変換（E_Clarke)

8.7.4.1 構文

© 2024
Toshiba Electronic Devices & Storage Corporation

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

引数：

_iu　U 相電流
_iv　V 相電流
_iw　W 相電流
_ialpha　α軸電流格納アドレス
_ibeta　β軸電流格納アドレス

戻り値：

なし

8.7.4.2 **処理内容**

3 相の電流(Iu, Iv, Iw)を 2 相(Iα, Iβ)に変換します。

下記演算式により Iα、Iβを演算します。

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

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

係数$\frac{2}{3}$は、3 相電流を 2 相のαβ軸電流に変換すると振幅が$\frac{3}{2}$倍になるため、振幅を等価とするための係数。

3 相電流の総和は 0 となるため、Iu+Iv+Iw=0 とすると、

$$I\alpha = Iu$$

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

となります。

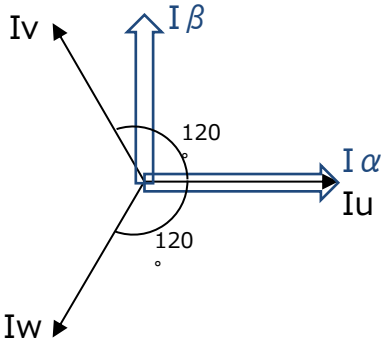


図 5 クラーク変換

8.7.5 パーク変換 (E_Park)

8.7.5.1 **構文**

```
void E_Park(   q15_t _ialpha,
               q15_t _ibeta,
```

uint16_t _theta,
q15_t* _id,
q15_t* _iq)

引数：

_ialpha　　α軸電流 Iα
_ibeta　　β軸電流 Iβ
_theta　　ローター位置θ: 0≤位置<360°(0 to 0xffff)
_id　　d 軸電流 Id 格納アドレス
_iq　　q 軸電流 Iq 格納アドレス

戻り値：

なし

8.7.5.2 処理内容

αβ軸の静止座標から回転座標の dq 軸に変換します。

下記演算式により Id、Iq を演算します。

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

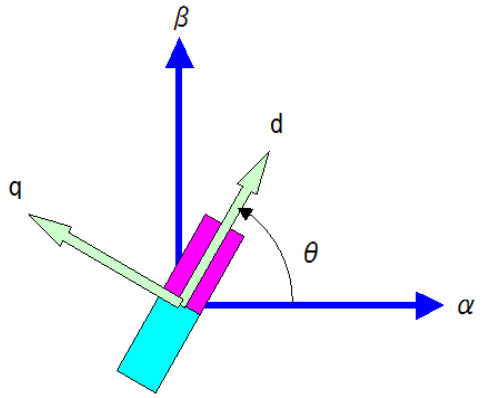


図 6 パーク変換

8.7.6 位置推定関数(D_Detect_Rotor_Position)

8.7.6.1 構文

void D_Detect_Rotor_Position(vector_t* const _motor)

引数：

vector_t* const _motor　：　モーター制御構造体

戻り値：

なし

8.7.6.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_Edetect	誘起電圧制御コマンド	---	
	drv.vector_cmd.F_vcomm_omega	推定速度制御コマンド	---	
	drv.vector_cmd.F_vcomm_theta	推定ローター位置制御コマンド	---	
	drv.Id	d 軸電流	Q15	
	drv.Iq	q 軸電流	Q15	
	drv.omega_com	駆動速度指令値	Q31	
	drv.theta_com	電気角指令値	Q0	
	drv.Vd	d 軸電圧	Q31	
	para.motor.Lq	モーターq 軸インダクタンス	Q12	
	para.motor.r	モーター巻線抵抗	Q12	
	para.pos.ctrlprd	位置推定制御周期	Q0	
	para.pos.ki	位置推定積分ゲイン	Q15	
	para.pos.kp	位置推定比例ゲイン	Q15	
入出力	drv.Ed	d 軸誘起電圧	Q15	
	drv.Ed_I	d 軸誘起電圧積分値	Q31	
	drv.Ed_PI	d 軸誘起電圧 PI 値	Q31	
	drv.omega	推定速度	Q31	
出力	drv.theta	ローター位置	Q0	

8.7.6.3 **処理内容**

操作量がモーター駆動信号の推定速度ω_{est}、制御量が d 軸誘起電圧 Ed として PI 制御を行います。
なお Ed の目標値は常に 0 です。つまり偏差は-Ed となります。
PI 制御により求めた推定速度ω_{est}を積分することで、ローター位置θ(角度)を求めます。
推定速度 omega は、F_vcomm_omega が CLEAR の場合は、駆動速度指令値 omega_com を推定速度 omega として使用します。

モーターの d 軸に関する等価回路方程式は、下記で表すことができます。

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

(p=d/dt, Id≈一定値より pId=0 とすることができる。)

Vd :モーター印加電圧　　Id,Iq :モーター電流
ω_{est} :推定角速度
R :抵抗　　Ld,Lq :インダクタンス

よって、d 軸誘起電圧 Ed は下記演算式で求めることができます。

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

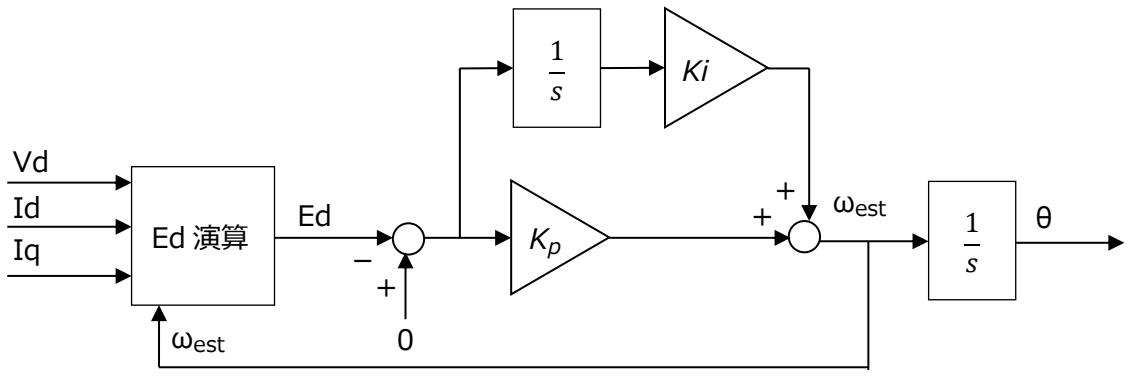


図 7 d 軸誘起電圧 Ed による位置推定ブロック図

© 2024
Toshiba Electronic Devices & Storage Corporation

8.7.7 速度制御関数(D_Control_Speed)

8.7.7.1 構文

void D_Control_Speed(vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.7.7.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_current	電流指令制御コマンド	---	
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	
	drv.omega	推定速度	Q31	
	drv.omega_com	駆動速度指令値	Q31	
	para.id_lim	d 軸電流制限値	Q31	
	para.iq_lim	q 軸電流制限値	Q31	
	para.spd.ki	速度制御積分ゲイン	Q15	
	para.spd.kp	速度制御比例ゲイン	Q15	
入出力	drv.Iq_ref_I	q 軸電流積分値	Q31	
	drv.omega_dev	速度偏差	Q15	
出力	drv.Id_ref	d 軸電流基準値	Q15	
	drv.Iq_ref	q 軸電流基準値	Q15	

8.7.7.3 処理内容

制御量が出力周波数ω、操作量が q 軸電流 Iq として PI 制御を行います。

速度指令値を実測の速度の偏差から、d 軸と q 軸電流基準値を決定します。

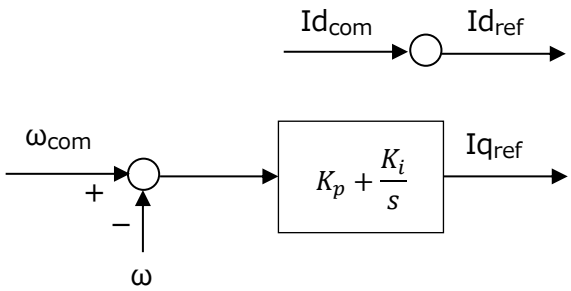


図 8 速度制御ブロック図

8.7.8 電流制御関数 (D_Control_Current)

8.7.8.1 構文

void D_Control_Current(vector_t* const _motor)

引数：

vector_t* const _motor　：　モーター制御構造体

戻り値：

なし

8.7.8.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_volt	電圧指令制御コマンド	---	
	stage.itr	割り込みステージ	---	
	drv.Id_ref	d 軸電流基準値	Q15	
	drv.Iq_ref	q 軸電流基準値	Q15	
	drv.Id	d 軸電流	Q15	
	drv.Iq	q 軸電流	Q15	
	drv.Vd_com	d 軸電圧指令値	Q31	
	drv.Vq_com	q 軸電圧指令値	Q31	
	drv.Vd_out	出力電圧値		電圧駆動時だけ有効
	para.crt.dkp	d 軸電流制御比例ゲイン	Q15	Q12 対応あり
	para.crt.dki	d 軸電流制御積分ゲイン	Q15	Q12 対応あり
	para.crt.qkp	q 軸電流制御比例ゲイン	Q15	Q12 対応あり
	para.crt.qki	q 軸電流制御積分ゲイン	Q15	Q12 対応あり
	drv.shift2_mode	シフト 2 制御モード	---	
	drv.Vd_lim_shift2	d 軸電圧リミット値	Q31	
	drv.Vq_lim_shift2	q 軸電圧リミット値	Q31	
入出力	drv.Vd_I	d 軸電圧積分値	Q31	
	drv.Vq_I	q 軸電圧積分値	Q31	
出力	drv.Vd	d 軸電圧	Q31	
	drv.Vq	q 軸電圧	Q31	

8.7.8.3 処理内容

電流基準値と実際の電流の偏差から、d 軸と q 軸電圧を決定します。電圧指令制御コマンドが SET のときは、下記 PI 制御により d 軸電圧、q 軸電圧を演算し、CLEAR のときは、d 軸電圧、q 軸電圧、積分値を指令値にします。制御量が q 軸電流 Iq、操作量が q 軸電圧 Vq として PI 制御を行います。

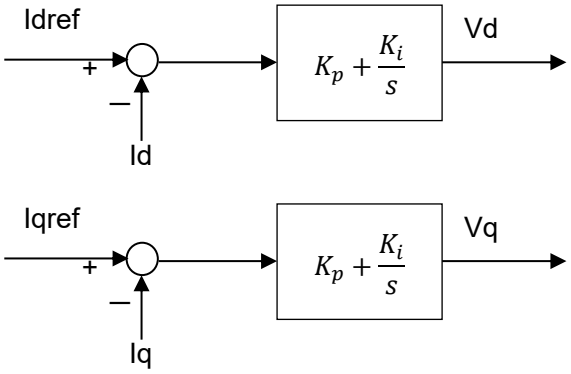


図 9 電流制御ブロック図

8.7.9 出力座標変換（D_OutputTransformation）

8.7.9.1 構文

void D_OutputTransformation (vector_t* const _motor)

引数：

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

8.7.9.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_onoff	モーター出力制御コマンド	---	
	drv.Vd	d 軸電圧	Q31	
	drv.Vq	q 軸電圧	Q31	
	drv.Vdc	電源電圧	Q15	
	drv.theta	ローター位置	Q0	[deg/360]
	drv.PhCvMd	相変換モード	---	0:空間ベクトル 1:逆クラーク
入出力	drv.Sector	セクター	---	
出力	drv.Valpha	α軸電圧	Q31	
	drv.Vbeta	β軸電圧	Q31	
	drv.DutyU	U 相 PWM Duty 比	Q15 (符号無し)	0.0 ～ 1.0
	drv.DutyV	V 相 PWM Duty 比	Q15 (符号無し)	0.0 ～ 1.0
	drv.DutyW	W 相 PWM Duty 比	Q15 (符号無し)	0.0 ～ 1.0
	drv.Sector1	前回セクター	---	

8.7.9.3 処理内容

dq 軸電圧(Vd,Vq)から 3 相の PWMDuty 比(DutyU, DutyV, DutyW)に変換します。

1．8.7.10 逆パーク変換により、dq 軸電圧(Vd,Vq)からαβ軸電圧(Vα、Vβ)に変換します。
2．αβ軸電圧(Vα、Vβ)から現在の「セクター」を演算します。演算前の「セクター」を「前回セクター」に保存し、モーター電流取得時に使用します。
3．8.7.12 空間ベクトル変調により、αβ軸電圧(Vα、Vβ)から 3 相 Duty 比(DutyV, DutyW, DutyW)に変換します。

8.7.10 逆パーク変換（E_InvPark）

8.7.10.1 構文

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

© 2024
Toshiba Electronic Devices & Storage Corporation

引数：

_d　　d 軸

_q　　q 軸

_theta　　ローター位置θ: 0≤位置<360°(0 to 0xffff)

_alpha　　α軸格納アドレス

_beta　　β軸格納アドレス

戻り値：

なし

8.7.10.2 処理内容

回転座標の dq 軸から静止座標のαβ軸に変換します。

下記演算式により Vα、Vβを演算します。

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

8.7.11 セクター演算（D_CalSector)

8.7.11.1 構文

```
uint8_t D_CalSector(    q31_t _valpha,
                        q31_t _vbeta)
```

引数：

_valpha　　α軸電圧

_vbeta　　β軸電圧

戻り値：

セクター

8.7.11.2 処理内容

αβ軸電圧から「セクター」を演算します。

「前回のセクター」値を保存し、下記条件式により今回の「セクター」値を決定します。

条件		セクター値
条件 1	条件 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

© 2024
Toshiba Electronic Devices & Storage Corporation

	$\lvert V\alpha\rvert < (\lvert V\beta\rvert/\sqrt{3})$	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 空間ベクトル変調（D_SVM）

8.7.12.1 構文

```
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)
```

引数：

_valpha	α軸 Vα
_vbeta	β軸 Vβ
_vdc	電源電圧
_sector	セクター
_modul	変調方式
_p_vu	U 相 Duty 比 DutyU 格納アドレス
_p_vv	V 相 Duty 比 DutyV 格納アドレス
_p_vw	W 相 Duty 比 DutyW 格納アドレス

戻り値：

なし

8.7.12.2 処理内容

2 相のαβ軸電圧から 3 相 PWM の Duty 比を求めます。

空間ベクトル変調により各相の PWM Duty 比を求めます。

- 空間ベクトル変調とは?

インバーター駆動回路のスイッチング素子の状態は、上相の素子は ON or OFF、下相の素子は上相の逆状態(デッドタイムは無視)となるため、

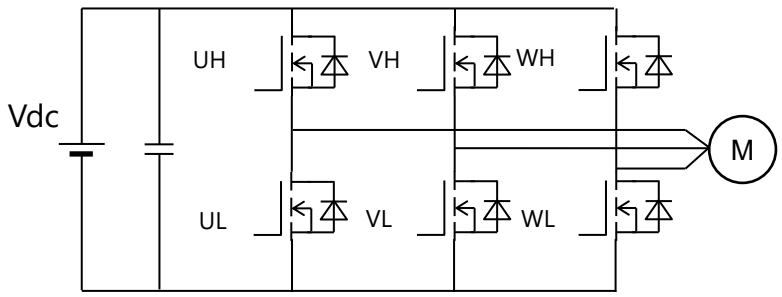
表 2 のように 8 パターン存在します。

この 8 パターンを V0 から V7 ベクトルとし、図に表すと、V0 と V7 ベクトルは、線間電圧がかからないため原点に存在し、残りの 6 つのベクトル(V1 から V6)は図 11 空間ベクトル図 11 のように 60°ごとに表されます。

このうち隣り合う 2 つのベクトルを組み合わすことにより、任意の出力電圧ベクトル V を得ることができます。

出力電圧ベクトル V がセクター0 に存在するときの PWM Duty の演算方法を次ページから説明します。

© 2024
Toshiba Electronic Devices & Storage Corporation



UH: U 相上相スイッチング素子　UL: U 相下相スイッチング素子
VH: V 相上相スイッチング素子　VL: V 相下相スイッチング素子
WH: W 相上相スイッチング素子　WL: W 相下相スイッチング素子
Vdc: 電源電圧

表 2 インバータースイッチング状態

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: 上相 OFF、下相 ON
1: 上相 ON、 下相 OFF

図 10 インバーター駆動回路

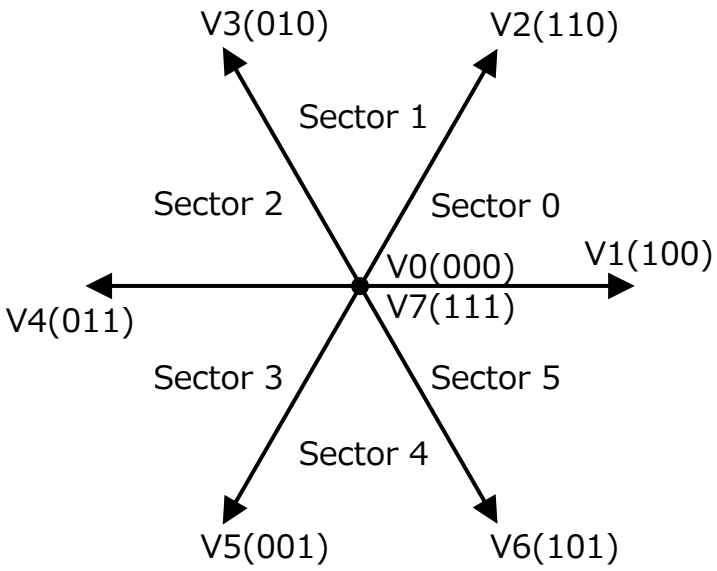


図 11 空間ベクトル

・出力電圧 V がセクター 0 内に存在する場合

例えば図 12 で、Vα、Vβの合成ベクトル V はセクター 0 上にあるため電圧ベクトル V1 と V2 にそれぞれ係数 t1、t2 をかけて得られるベクトル V1'、V2' の合成ベクトルとして得ることができます。PWM 波形で表すと、図 13 のように PWM 半周期で、V1、V2 をそれぞれ時間 t1、t2 だけ発生させることで 出力電圧ベクトル V を合成します。

© 2024
Toshiba Electronic Devices & Storage Corporation

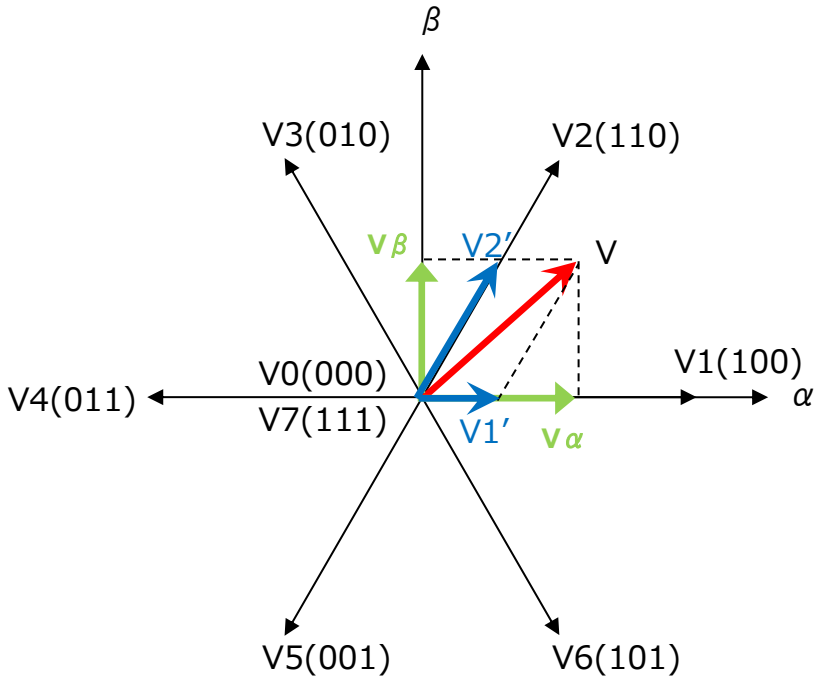


図 12 セクター0 時の電圧ベクトル

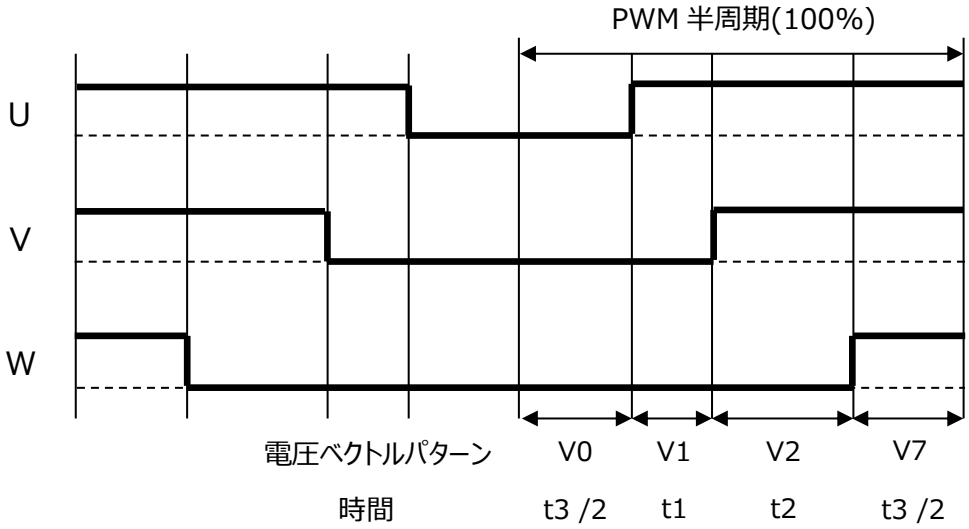


図 13 3 相変調時の PWM 波形

図 12 より、Vα、Vβは、

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

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

と表すことができます。

よって、V1′,V2′は、

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

© 2024
Toshiba Electronic Devices & Storage Corporation

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

となります。

モーター電源電圧を Vdc とすると、

$$V1' = t1 \times Vdc$$
$$V2' = t2 \times Vdc$$

よって、t1,t2,t3 の比率は、

$$t1 = k \times \frac{V\alpha - \frac{1}{\sqrt{3}}V\beta}{Vdc}$$
$$t2 = k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$$
$$t3 = 100\% - t1 - t2$$

となります。

k は、3 相から 2 相変換時に大きさを一定とするための変換係数で 3/2 です。

V0,V7 ベクトルの発生時間をそれぞれ t3/2 としたのが 3 相変調で、V0 ベクトル発生時間を t3,V7 ベクトル発生時間を 0 としたのが 2 相変調となります。

よって、3 相の Duty は下記で t1,t2,t3 から下記計算で求めることができます。

U 相 Duty = t1+t2+(t3/2)

V 相 Duty＝t2+(t3/2)

W 相 Duty＝t3/2

以下同様に出力電圧のセクターごとに、表 3 セクターごとの Duty ON 時間の演算式で OFF Duty D0, 1 相のみ ON の Duty D1, 2 相 ON の Duty D2 を演算します。

表 3 セクターごとの Duty ON 時間の演算式

セクター	1 相のみ ON の Duty D1	2 相 ON の Duty 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	$-k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$	$k \times \frac{-V\alpha + \frac{1}{\sqrt{3}}V\beta}{Vdc}$	

© 2024
Toshiba Electronic Devices & Storage Corporation

4	$k \times \frac{-V\alpha - \frac{1}{\sqrt{3}}V\beta}{Vdc}$	$k \times \frac{V\alpha - \frac{1}{\sqrt{3}}V\beta}{Vdc}$	
5	$k \times \frac{V\alpha + \frac{1}{\sqrt{3}}V\beta}{Vdc}$	$-k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$	

k は、変換前後の振幅を一定にするための係数で 3/2。

D0,D1,D2 から U,V,W 相の Duty は、表 4 の演算式で算出します。

表 4 セクターと各相の Duty の関係

セクター	3 相変調			2 相変調		
	U 相 Duty	V 相 Duty	W 相 Duty	U 相 Duty	V 相 Duty	W 相 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 トリガータイミング演算（D_CalTrgTiming)

8.7.13.1 構文

void D_CalTrgTiming(vector_t* const _motor)

引数：

_motor　　モーター制御構造体

戻り値：

なし

8.7.13.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.DutyU	U 相 PWM Duty 比	Q15	0.0 ～ 1.0
	drv.DutyV	V 相 PWM Duty 比	Q15	0.0 ～ 1.0
	drv.DutyW	W 相 PWM Duty 比	Q15	0.0 ～ 1.0
	drv.command.modul	変調方式	---	2 相変調
	drv.TrgPosMd	電流検出位置モード	---	1 シャント(前半) 1 シャント(後半)
	para.TrgComp	トリガータイミング補正値	Q15	−1.0 ～ 1.0
出力	drv.AdTrg0	AD トリガータイミング 0(TRG0)	Q15	−1.0 ～ 1.0
	drv.AdTrg1	AD トリガータイミング 1(TRG1)	Q15	−1.0 ～ 1.0

© 2024
Toshiba Electronic Devices & Storage Corporation

8.7.13.3 処理内容

モーター電流および電源電圧 Vdc の AD 変換値の取得トリガータイミングを演算します。

- 1 シャントの場合

基本トリガー位置は、U,V,W 相の PWM Duty の値から演算します。
トリガータイミング補正値に 0 以外が入っていたら、演算したトリガー位置から補正します。
電源電圧 Vdc のトリガー位置は、キャリアーエンドの位置に固定されます。（初期設定時に設定し、この関数では処理しない）

	変調方式	トリガータイミング 0 (TRG0) モーター電流	トリガータイミング 1 (TRG1) モーター電流	トリガータイミング 2 (TRG2) 電源電圧 Vdc
PWM 周期後半	3 相変調	Duty 中間値	Duty 最大値	キャリアーエンド固定
	2 相変調	Duty 中間値	Duty 最大値	キャリアーエンド固定
PWM 周期前半	3 相変調	−Duty 中間値	−Duty 最小値	キャリアーエンド固定
	2 相変調	−Duty 中間値	Duty 中間値	キャリアーエンド固定

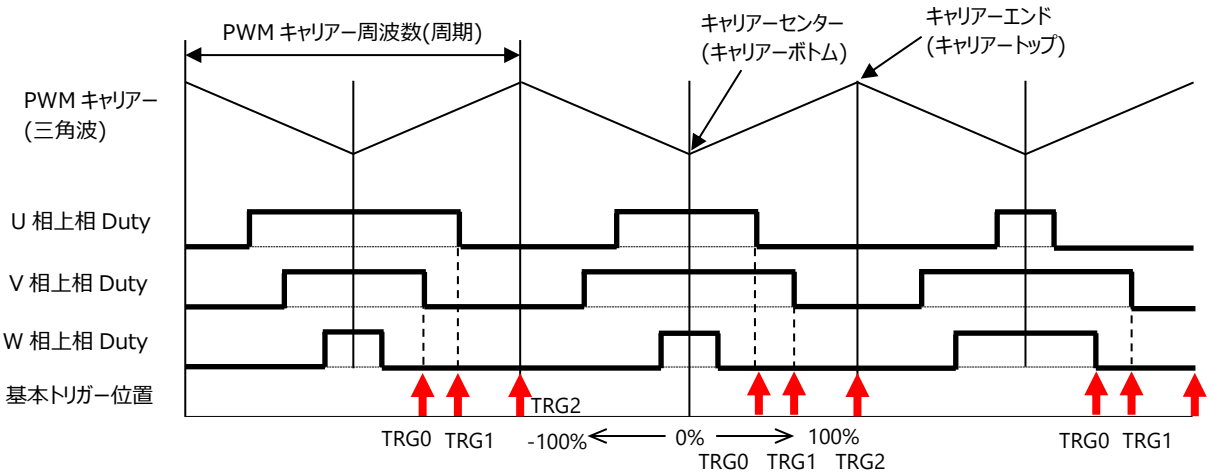


図 14 1 シャント、PWM 周期後半、3 相変調時のトリガー位置

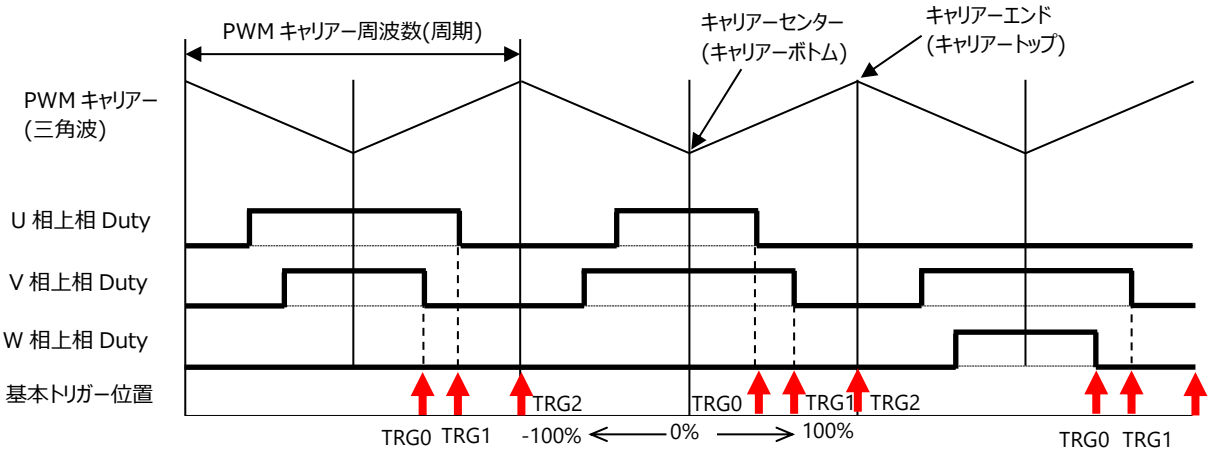


図 15 1 シャント、PWM 周期後半、2 相変調時のトリガー位置

© 2024
Toshiba Electronic Devices & Storage Corporation

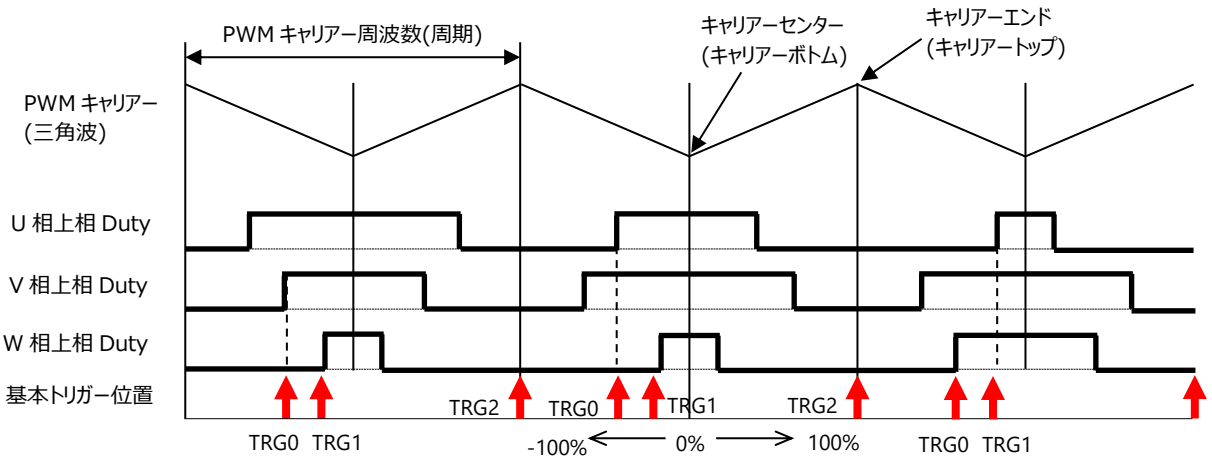


図 16 1 シャント、PWM 周期前半、3 相変調時のトリガー位置

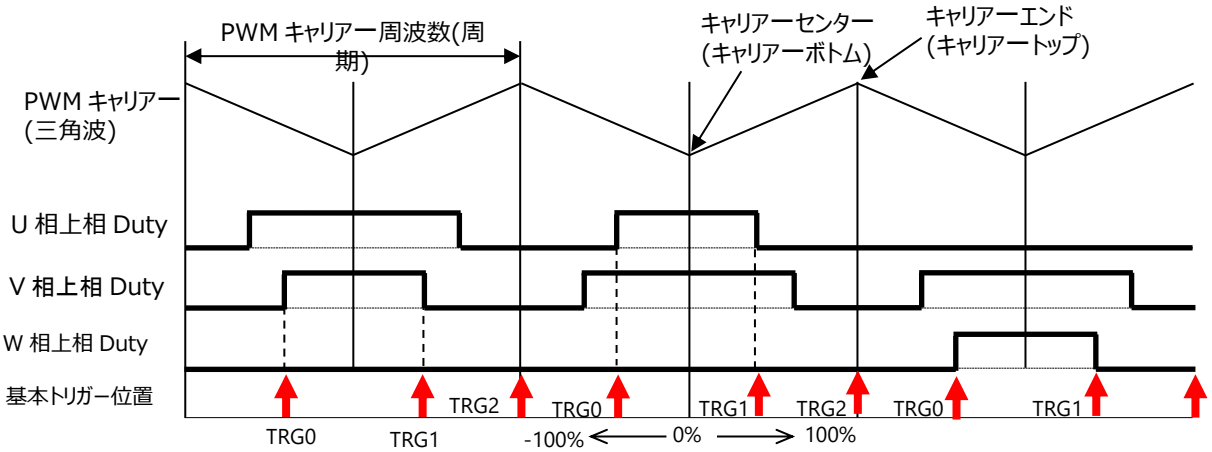


図 17 1 シャント、PWM 周期前半、2 相変調時のトリガー位置

8.7.14 PWM レジスター設定 (PMD_RegDataSet)

8.7.14.1 構文

```
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)
```

引数：

PMDx	PMD レジスターアドレス
duty_u,	U 相 PWM Duty 比
duty_v,	V 相 PWM Duty 比
duty_w,	W 相 PWM Duty 比
adtrg0,	AD トリガータイミング 0 位置

© 2024
Toshiba Electronic Devices & Storage Corporation

adrtg1　　AD トリガータイミング 1 位置

戻り値：

なし

8.7.14.2 処理内容

3 相 PWM Duty とトリガータイミングの値をレジスターに設定します。

入力された引数を、下記演算を行い PMD レジスターに設定します。

(M4KNA)

TRGCMP0 = (adtrg0 + 32768) ÷2
TRGCMP1 = (adtrg1 + 32768) ÷2
CMPU = duty_u
CMPV = duty_v
CMPW = duty_w

8.7.15 電流誤検知検出関数 (D_Check_DetectCurrentError)

8.7.15.1 構文

int D_Check_DetectCurrentError (vector_t* const _motor)

引数：

_motor　　モーター制御構造体

戻り値：

電流検出状態

8.7.15.2 変数

方向	名前	意味	Q フォーマット	備考
入力	para.chkpls	パルスチェック幅	---	
	shunt_type	シャントタイプ	---	1:1 シャント
	drv.command.modul	変調方式駆動コマンド	---	2 相変調
	drv.DutyU	U 相 PWM Duty 比	Q15	0.0 ～ 1.0
	drv.DutyV	V 相 PWM Duty 比	Q15	0.0 ～ 1.0
	drv.DutyW	W 相 PWM Duty 比	Q15	0.0 ～ 1.0
出力	drv.idetect_error	電流検出状態	---	0:検出可能 1:検出不能

8.7.15.3 処理内容

PWM Duty 幅により電流を検出できないタイミングであることを検出します。

3 シャント時は、Duty の中間値が設定値より大きくなることを検出します。

1 シャント時は、最小 Duty 幅、Duty 幅差の大きさが設定値より小さくなることを検出します。なお、1 シャントで 2 相変調のときの最小 Duty 幅は 3 相 Duty 幅の中間 Duty 幅となります。

© 2024
Toshiba Electronic Devices & Storage Corporation

・1 シャントの場合

PWM Duty 自体の幅と、PWM Duty の差の幅が、設定値より小さくなると検出 NG と判定します。

8.7.16 インバーター出力電圧の算出関数（VE_GET_Cal_Vdq)

8.7.16.1 構文

q15_t VE_GET_Cal_Vdq (　q15_t _vd, q15_t _vq)

引数：

_vd　　d 軸電圧

_vq　　q 軸電圧

戻り値：

インバーター出力電圧(Vdq)

8.7.16.2 変数

方向	名前	意味	Q フォーマット	備考
入力	_vd	d 軸電圧	Q15	0.0 ～ 1.0
	_vq	q 軸電圧	Q15	0.0 ～ 1.0
出力	---	インバーター出力電圧(Vdq)	Q15	0.0 ～ 1.0

8.7.16.3 処理内容

インバーター出力電圧(Vdq)を算出します。

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

© 2024
Toshiba Electronic Devices & Storage Corporation

8.8 温度保護制御関数

8.8.1 温度保護制御関数（B_Protect_Temperature)

8.8.1.1 構文

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

引数：

_temp　温度保護制御構造体

_motor　モーター制御構造体

戻り値：

なし

8.8.1.2 処理内容

各温度センサーの AD 値を取得し、エラー状態の確認を行います。

8.8.2 温度取得関数（B_Protect_GetTemperature)

8.8.2.1 構文

void B_Protect_GetTemperature(temperature_t * const _temp)

引数：

_temp　温度保護制御構造体

戻り値：

なし

8.8.2.2 処理内容

次の温度を AD 変換により取得します。

OutRoom、OutPipe、OutEXHAUST、HVMOS、IGBT、Diode

8.8.3 温度エラーチェック関数（B_Protect_CheckTemperatureError)

8.8.3.1 構文

void B_Protect_CheckTemperatureError(temperature_t * const _temp)

引数：

_temp　温度保護制御構造体

戻り値：

なし

8.8.3.2 処理内容

© 2024
Toshiba Electronic Devices & Storage Corporation

各温度センサーの AD 値から開放・短絡のエラーを判定します。

8.8.4 温度制御状態選択関数（B_Protect_TemperatureStatusSel)

8.8.4.1 構文

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

引数：

_tc	速度制限構造体
_t	温度
_times	サンプリング回数

戻り値：

なし

8.8.4.2 処理内容

現在の温度により、エラー状態判定する。

8.8.5 温度保護エラー判定関数（B_Error_Collection)

8.8.5.1 構文

void B_Error_Collection(void)

引数：

なし

戻り値：

なし

8.8.5.2 処理内容

温度に関してファンモーター・コンプレッサーモーター・PFC のエラー判定を行い、設定された優先順位により表示するエラーを決定します。

8.8.6 エラー表示制御関数（B_Error_DisplayCtrl)

8.8.6.1 構文

void B_Error_DisplayCtrl(void)

引数：

なし

戻り値：

なし

© 2024
Toshiba Electronic Devices & Storage Corporation

8.8.6.2 処理内容

各エラー状態に対して、LED1～4 の表示設定を行います。エラーがないときは常時点灯、エラー時は指定の LED について 2 秒のインターバルごとに繰り返す 500ms 間隔の点滅の回数を設定します。
各エラーの表示内容は下記となります。

・ファンモーター

エラーなし	: LED2 消灯
過電流(ハード)	: LED2 点滅 1 回
過電流(ソフト)	: LED2 点滅 2 回

・コンプレッサーモーター

エラーなし	: LED3 消灯
過電流(ハード)	: LED3 点滅 1 回
過電流(ソフト)	: LED3 点滅 2 回

・PFC

エラーなし	: LED4 点灯
過電流(ハード)	: LED4 点滅 1 回
過電流(ソフト)	: LED4 点滅 2 回
AC 過電圧	: LED4 点滅 3 回
AC 電圧不足	: LED4 点滅 4 回
DC 過電圧	: LED4 点滅 5 回
DC 電圧不足	: LED4 点滅 6 回
AC 電圧周波数	: LED4 点滅 7 回
ゼロクロス	: LED4 点滅 8 回

・温度

エラーなし	: LED1 点灯
センサー開放/短絡	: LED1 点滅 1 回
温度異常	: LED1 点滅 2 回

8.8.7 エラー表示関数（B_Error_Display）

8.8.7.1 構文

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

引数：

_time	点滅回数
_sLED	選択 LED
i	LED 番号

戻り値：

© 2024
Toshiba Electronic Devices & Storage Corporation

なし

8.8.7.2 処理内容

LED の点滅間隔と ON/OFF の切換、点滅のインターバル期間の制御を行います。また、設定に従い LED 出力を行います。

© 2024
Toshiba Electronic Devices & Storage Corporation

9. PFC 制御

9.1 概要

PFC とはインバーター負荷を電力システムに対する抵抗負荷のように動作させることで、正弦波入力電圧に応じて入力電流波形を正弦波に近づけることができます。これにより、無駄なエネルギーの消費を抑え、高調波を抑制することで電子機器の誤動作やけがの防止につながります。

PFC は AC 電源の PF 値(皮相電力に占める実駆動電力の割合)を向上させて 1.0 に近づけるための機能です。

本システムでは、ダイオードブリッジを用いた単相 PFC の制御を行います。

9.2 単相 PFC のシステム構成

単相 PFC のシステム構成は下記となります。

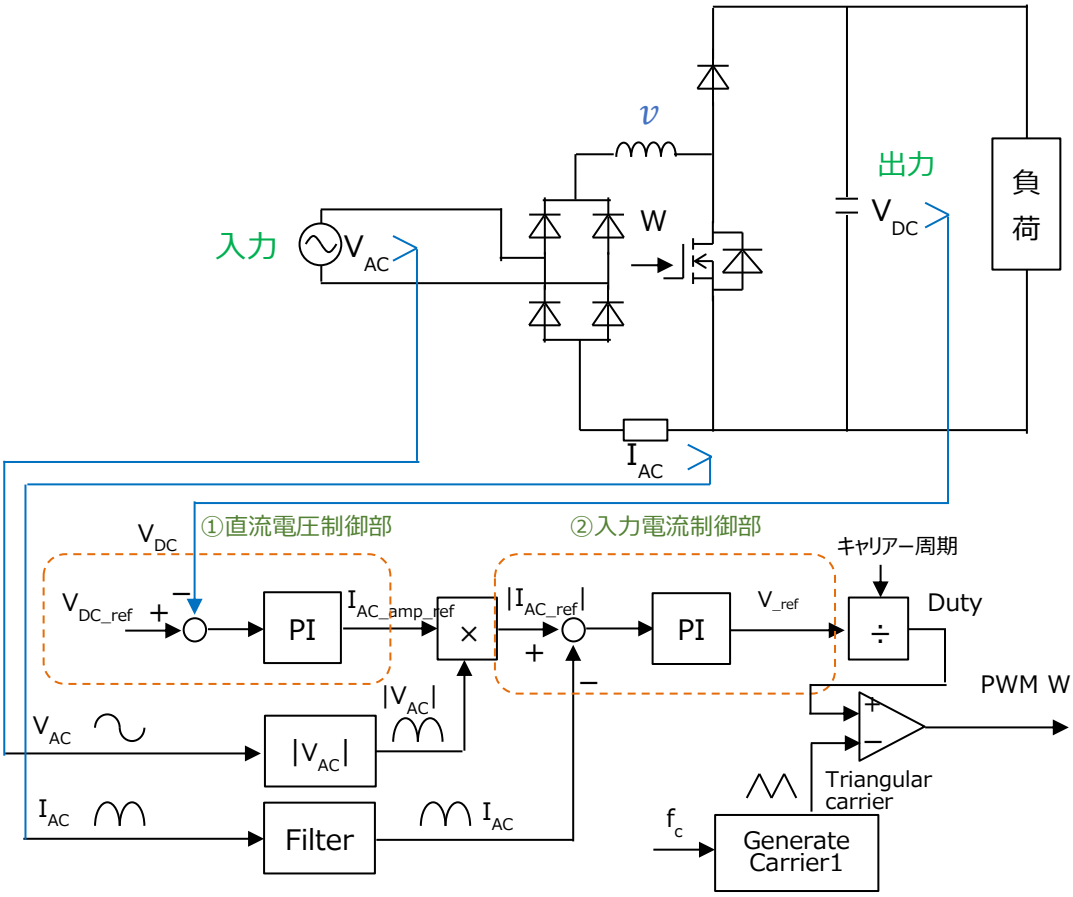


図 18 PFC 制御のブロック図

9.3 PFC 駆動関数

9.3.1 PFC 割り込み制御 (HPFC_Control_Int)

9.3.1.1 構文

void HPFC_Control_Int(void)

引数：

なし

戻り値：

なし

9.3.1.2 処理内容

AD 変換完了時の PFC 制御を行います。
エラーがなければ電流制御、PWM の Duty、AD トリガーのレジスター設定、エラー時は PWM 出力を OFF します。

9.3.2 PFC AD 変換完了時割り込み処理（HPFC_INT_ADC_Fin)

9.3.2.1 構文

void HPFC_INT_ADC_Fin(void)

引数：

なし

戻り値：

なし

9.3.2.2 処理内容

PFC AD 変換完了時割り込みの処理を行います。
相電流値を取得し、AD 変換完了時の PFC 制御を行います。

9.3.3 PFC 制御メイン関数（HPFC_Control_Main)

9.3.3.1 構文

void HPFC_Control_Main(void)

引数：

なし

戻り値：

なし

9.3.3.2 処理内容

DC 電圧の制御を行います。
開始電圧まで一定間隔で昇圧していき、開始電圧到達後は目標電圧で制御を行います。

9.3.4 PFC ユーザー制御（HPFC_UserControl)

9.3.4.1 構文

void HPFC_UserControl(void)

© 2024
Toshiba Electronic Devices & Storage Corporation

引数：
なし
戻り値：
なし

9.3.4.2 処理内容

PFC の ON/OFF を制御します。
ON は定常状態で AC 電流が一定値以上の場合、OFF は AC 電流が一定値未満になったタイミングで行います。

9.3.5 PFC エラーチェック（HPFC_INT_ERROR_CHECK_16k)

9.3.5.1 構文

void HPFC_INT_ERROR_CHECK_16k(void)

引数：
なし
戻り値：
なし

9.3.5.2 処理内容

PFC に関して次のエラー判定を行います。
コンプレッサーモーターエラー、DC 電圧(過電圧、電圧不足)、AC 電圧(過電圧、電圧不足、周波数、ゼロクロス)

9.3.6 PFC　AC/DC 電圧割り込み処理（HPFC_VacVdc_Handle)

9.3.6.1 構文

void HPFC_VacVdc_Handle(pfc_t* const _pfc)

引数：
_pfc　　PFC 制御構造体
戻り値：
なし

9.3.6.2 処理内容

割り込み時に DC 電圧取得/演算、AC 電圧取得/演算、電圧制御を行います。

9.3.7 PFC 電流制御（HPFC_ControlCurrent)

9.3.7.1 構文

void HPFC_ControlCurrent(pfc_t * const _pfc, phase_mode_e _mode)

© 2024
Toshiba Electronic Devices & Storage Corporation

引数：

_pfc　　PFC 制御構造体

_mode　　相設定

戻り値：

なし

9.3.7.2 処理内容

AC 電流から PWM の Duty を演算します。

9.3.8 PFC 目標 DC 電圧設定（HPFC_VdcTarget)

9.3.8.1 構文

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

引数：

_pfc　　PFC 制御構造体

_motor　　モーター制御構造体

戻り値：

なし

9.3.8.2 処理内容

AC 電圧から算出された現在の DC 電圧と目標 DC 電圧の差が 5V を超えたとき、一定の割合で目標電圧まで昇圧します。

9.3.9 PFC 電流制御（HPFC_INT_pwmA)

9.3.9.1 構文

void HPFC_INT_pwmA(void)

引数：

なし

戻り値：

なし

9.3.9.2 処理内容

算出された PWM Duty に対してトリガー値を算出します。

9.3.10 PFC 電流制御（HPFC_CalVacFilter_16k)

9.3.10.1 構文

void HPFC_CalVacFilter_16k(pfc_t * const _pfc)

© 2024
Toshiba Electronic Devices & Storage Corporation

引数：

_pfc　　PFC 制御構造体

戻り値：

なし

9.3.10.2 処理内容

取得した AC 電圧からピーク電圧とボトム電圧を記憶して、ゼロクロス判定を行います。
AC 電圧は、ゼロクロス電圧を基準に入力電圧の負電圧分を正電圧に置き換えます。

9.3.11 PFC 電圧制御（HPFC_ControlVoltage_16k)

9.3.11.1 構文

void HPFC_ControlVoltage_16k(pfc_t * const _pfc)

引数：

_pfc　　PFC 制御構造体

戻り値：

なし

9.3.11.2 処理内容

目標 DC 電圧から増幅 AC 電流指令値を算出します。

© 2024
Toshiba Electronic Devices & Storage Corporation

10. 定数定義説明

10.1 モータードライバー設定用引数：(D_Para.h)

10.1.1 モーター制御チャネル選択

__CONTROL_MOTOR_FAN ファンモーター(CH0)を制御するとき定義してください。
__CONTROL_MOTOR_COMP コンプレッサーモーター(CH1)を制御するとき定義してください。
__CONTROL_PFC PFC(CH2)を制御するとき定義してください。

10.1.2 DAC 出力選択

__USE_DAC 評価用の変数値出力用の DAC 制御を行うとき定義してください。

10.1.3 RAMScope(NBD)出力選択

__USE_NBD RAMScope を使用するとき定義してください。

10.1.4 指令速度単位選択

__TGTSPD_UNIT 指令速度の単位を選択できます
0: Hz of Electrical angle
1: RPS of Mechanical angle
2: RPM of Mechanical angle

10.1.5 モーター制御および PFC ON/OFF 選択

__USE_mikroE_COMMUNICATION_FUNC モーター制御および PFC ON/OFF を選択できます。
なし: ライブウォッチから RAM の値を書き換えての制御
あり: UART 通信による制御

10.1.6 共通パラメーター一覧

引数：名	説明
cMAINLOOP_PRD	メイン周期設定
	単位[s] 分解能 4kHz
	メイン周期の時間を設定してください。
cIXO_AVE	ゼロ電流値用フィルター係数
	--
	フィルター係数を設定してください。 大きな値を設定すると安定しますが、反応が遅れます。
cVDQ_AVE	電源電圧 Vdc,出力電圧 Vdq 用フィルター係数
	--

© 2024
Toshiba Electronic Devices & Storage Corporation

	フィルター係数を設定してください。 大きな値を設定すると安定しますが、反応が遅れます。

10.2 モータードライバー設定用引数：モーターチャネル(D_Para_x.h) (x=Fan,Comp)

モータードライバー部の引数：を変更することにより、さまざまなモーターを駆動することが可能です。

10.2.1 制御選択

10.2.1.1 AMP 選択

__USE_INAMP　　内蔵 AMP を使用する場合、定義してください。

10.2.2 モーターチャネル別引数：一覧

引数：名	説明
cPOLH	上相ドライバー論理設定
	0: Low active/ 1: High active
	基板設計に応じて値を変更してください。 上相ドライバーの論理の設定を行ってください。
cPOLL	下相ドライバー論理設定
	0: Low active/ 1: High active
	基板設計に応じて値を変更してください。 下相ドライバーの論理の設定を行ってください。
cV_MAX	最大電圧値の設定
	単位[V]
	基板設計に応じて値を変更してください。 AD 変換後の 12 ビットカウント値でフルスケール(4095)に相当する電源電圧[単位 V]を設定してください。
cA_MAX	最大電流値の設定
	単位[A]
	基板設計に応じて値を変更してください。 AD 変換後の 11 ビットカウント値でフルスケール(2047)に相当する相電流[単位 A]を設定してください。
cSHUNT_TYPE	電流取得方式（3 シャントまたは 1 シャント）の設定
	1: 1 シャント/ 3: 3 シャント(未対応)
	基板設計に応じて値を変更してください。 電流取得方式の設定を行ってください。
cBOOT_TYPE	起動時の駆動方式の設定
	cBoot_i:電流型駆動(未対応)/ cBoot_v:電圧型駆動
	起動時の駆動方式を設定してください。1 シャント駆動の起動時など電流が取得できない場合などに、電圧型駆動を選択します。

© 2024
Toshiba Electronic Devices & Storage Corporation

cSHUNT_ZERO_OFFSET	オフセット電圧の設定
	単位[V]
	電流が流れていないときのシャント電圧を設定してください。 この値は、ゼロ電流平均値の初期値に使用します。
cADCH_CURRENT_IDC	電流取得 AD チャネル設定(1 シャント用)
	AINx
	電流を検出する AD チャネルを設定してください。
cADCH_VDC	電源電圧取得 AD チャネル設定
	AINx
	電源電圧 Vdc を検出する AD チャネルを設定してください。
cOVC	過電流検出値の設定
	単位[A]
	過電流電流値を設定してください。 この設定値以上のコイル電流を検出すると、出力をソフトで OFF にします。
cVDC_MINLIM	電源電圧 Vdc 最低値の設定
	単位[V]
	電源電圧 Vdc の最低値を設定してください。 この値未満の電圧値を検出すると、モーターを停止させます。
cVDC_MAXLIM	電源電圧 Vdc 最高値の設定
	単位[V]
	電源電圧 Vdc の最高値を設定してください。 この値より大きな値の電圧値を検出すると、モーターを停止させます。
cPWMPRD	PWM 周期の設定
	単位[us] 分解能 25 ns@80 MHz
	PWM キャリアー周期を設定してください。
cDEADTIME	デッドタイム値の設定
	単位[us] 分解能 0.1 μs@80 MHz
	デッドタイムの値を設定してください。
cREPTIME	ソフトウェア制御間引き回数の設定
	単位[回] 1 to 15
	ベクトル演算ソフトウェア処理を行うタイミングを間引きすることができます。 1 と設定すると、PWM1 周期ごとにベクトル演算のソフト処理を行います。 2 と設定すると、PWM2 周期に 1 回ベクトル演算のソフト処理を行います。
cID_ST_USER_ACT	d 軸始動電流の設定
	単位[A]
	d 軸始動電流の値を設定してください。

© 2024
Toshiba Electronic Devices & Storage Corporation

	この値の電流値で、位置決めおよび強制転流を行います。 定格転流の 10%程度の値を設定してください。 モーターが動かない場合は、動くまで徐々に値を大きくしてください。
cIQ_ST_USER_ACT	q 軸始動電流の設定
	単位[A]
	q 軸始動電流の値を設定してください。 d 軸始動電流の 1/2 程度の値を設定してください。 強制転流(d 軸制御)から定常(q 軸制御)移行時に、モーターが急加速する場合は、値を小さく、停止してしまう場合は、値を大きくしてください。
cMOTOR_R	モーターコイル抵抗値
	単位[Ohm]
	モーターコイル 1 相分の抵抗値を設定してください。
cMOTOR_LQ	q 軸インダクタンス値
	単位[mH]
	モーターコイルの q 軸インダクタンス値を設定してください。
cMOTOR_LD	d 軸インダクタンス値(未使用)
	単位[mH]
	モーターコイルの d 軸インダクタンス値を設定してください。
cPOLE	モーター極数の設定
	単位[pole]
	モーターの極数を設定してください。
cID_KP	d 軸電流制御比例ゲイン
	単位[V/A]
	d 軸電流制御比例ゲインを設定してください。
cID_KI	d 軸電流制御積分ゲイン
	単位[V/As]
	d 軸電流制御積分ゲインを設定してください。
cIQ_KP	q 軸電流制御比例ゲイン
	単位[V/A]
	q 軸電流制御比例ゲインを設定してください。
cIQ_KI	q 軸電流制御積分ゲイン
	単位[V/As]
	q 軸電流制御積分ゲインを設定してください。
cPOSITION_KP	位置推定比例ゲイン
	単位[Hz/V]

© 2024
Toshiba Electronic Devices & Storage Corporation

	位置推定の比例ゲインを設定してください。
cPOSITION_KI	位置推定積分ゲイン
	単位[Hz/Vs]
	位置推定の積分ゲインを設定してください。
cSPEED_KP	速度制御比例ゲイン
	単位[A/Hz]
	速度制御の比例ゲインを設定してください。
cSPEED_KI	速度制御積分ゲイン
	単位[A/Hzs]
	速度制御の積分ゲインを設定してください。
cSPD_PI_PRD	速度 PI 制御周期設定
	[回]
	速度 PI 制御周期を設定してください。 1 と設定すると、PWM1 周期ごとに速度 PI 演算を行います。 2 と設定すると、PWM2 周期に 1 回速度 PI 演算を行います。
cFCD_UD_LIM	駆動速度加速率(強制転流時)
	単位[Hz/s]
	強制転流時の速度加速率を設定してください。 強制転流の出力に、モーターの回転が追従してこない場合は、値を小さくしてください。
cSTD_UP_LIM	駆動速度加速率(定常時)
	単位[Hz/s]
	定常時の速度加速率を設定してください。
cSTD_DW_LIM	駆動速度減速率(定常時)
	単位[Hz/s]
	定常時の速度減速率を設定してください。
cBOOT_LEN	ブートストラップ波形出力時間
	単位[s] 分解能:1 ms
	ブートストラップ波形の出力時間を設定してください。
cINIT_LEN	位置決め時間
	単位[s] 分解能:1 ms
	位置決め時間を設定してください。
cINIT_WAIT_LEN	位置決め後待ち時間
	単位[s] 分解能:1 ms
	位置決め後の待ち時間を設定してください。

© 2024
Toshiba Electronic Devices & Storage Corporation

cGOUP_DELAY_LEN	強制定常切換後待ち時間
	単位[s] 分解能:1 ms
	強制定常切換後待ち時間を設定してください。
cHZ_MAX	最大周波数
	単位[Hz]
	マイコンで検出させる最大周波数を設定してください。 制御で使用する最大周波数の 10 から 20%増し程度の値を設定してください。 値が小さいほど演算精度が上がりますが、検出値がこの値を超えると制御が破たんします。
cHZ_MIN	強制転流から定常への切り替え速度
	単位[Hz]
	強制転流から定常へ移行させる速度を設定してください。 位置推定ができる(誘起電圧が検出できる)速度を設定します。
cHZ_SPWM	シフト PWM 切り替え速度（1 シャント 2 相変調用)
	単位[Hz]
	シフト PWM から通常 PWM へ切り替える速度を設定してください。 指令速度の値で切り替えます。
cMINPLS	最小パルス設定（1 シャント用）
	単位[us]
	1 シャント駆動時、電流が取得できなくなるときの、PWM パルス幅時間を設定してください。(※シフト PWM 動作中は、未評価です。シフト PWM 機能を使用する場合は、電圧型で起動するなどの適用をお願いします) PWM パルス幅が、この設定値未満の値になると、検出した電流値は使用せず、前回値を使用して制御を行います。
cID_LIM	d 軸電流 limit
	単位[A]
	d 軸電流のリミット値を設定してください。
cIQ_LIM	q 軸電流 limit
	単位[A]
	q 軸電流のリミット値を設定してください。
cINITIAL_POSITION	初期位置
	単位[deg]
	位置決め時の角度を電気角で設定してください。
cVD_POS	電圧駆動時の位置決め出力電圧
	単位[V]
	位置決め時の電圧を設定してください。 cBOOT_TYPE が"cBoot_v"のときだけ有効 詳細は 10.2.3.2 電圧型起動時の定数値を参照してください。

© 2024
Toshiba Electronic Devices & Storage Corporation

cSPD_COEF	電圧駆動時の強制転流出力電圧係数
	0 < x <1 の値を設定してください。
	強制転流時の出力電圧を決めます。 この設定値と指令速度を掛けた値を出力電圧としています。 Vd = cSPD_COEF × Omega_com cBOOT_TYPE が"cBoot_v"のときだけ有効 詳細は 10.2.3.2 電圧型起動時の定数値を参照してください。
cHZ_V2I	電圧制御から電流制御への切換速度
	単位[Hz]
	電圧制御から電流制御へ切り替える速度を設定してください。 cHZ_MIN より大きな値を設定すると、cHZ_MIN 以上になると定常へ移行し、電流制御に切り替わります。 cBOOT_TYPE が"cBoot_v"のときだけ有効
cPFC_ON_CUR	PFC 制御開始電流設定
	単位[A]
	PFC 制御を開始する電流を設定してください。
cOPEN	温度センサーオープン判定値
	AD 値
	温度センサーの開放判定値を設定してください。
cSHORTC	温度センサーショート判定値
	AD 値
	温度センサーの短絡判定値を設定してください。
cOPEN_SHORT_CNT	温度センサーオープン/ショート確定回数
	単位[回]
	温度センサーオープン/ショート状態の確定回数を設定してください。
cTEMP_CNT	温度エラー確定回数
	単位[回]
	温度異常状態のエラー確定回数を設定してください。
cTIMES	温度情報分解能設定
	分解能
	温度情報分解能を設定してください。 例：10→0.1℃
cEXH_OVER_EMG	Exhaust 異常判定温度
	単位[℃]
	Exhaust の温度異常判定値を設定してください。
cHVMOS_OVER_EMG	HVMOS 異常判定温度

© 2024
Toshiba Electronic Devices & Storage Corporation

	単位[℃]
	HVMOS の温度異常判定値を設定してください。
cIGBT_OVER_EMG	IGBT 異常判定温度
	単位[℃]
	IGBT の温度異常判定値を設定してください。
cDIODE_OVER_EMG	Diode 異常判定温度
	単位[℃]
	Diode の温度異常判定値を設定してください。
cOUTPIPE_OVER_EMG	Outpipe 異常判定温度
	単位[℃]
	Outpipe の温度異常判定値を設定してください。
cTEMP_UP1	センサー最大温度
	単位[℃]
	センサーの最大温度値を設定してください。
cTEMP_DW1	センサー最小温度
	単位[℃]
	センサーの最小温度値を設定してください。
__FIXED_VDC	電源電圧固定設定
	0:検出値 1:固定値
	電源電圧を固定値にする場合は 1 を設定してください。
cVDC	電源電圧固定値
	単位[V]
	電源電圧の値を設定してください。 __FIXED_VDC が"1"のときだけ有効

© 2024
Toshiba Electronic Devices & Storage Corporation

10.2.3 定数設定値と波形の関係

10.2.3.1 電流型起動時の定数値

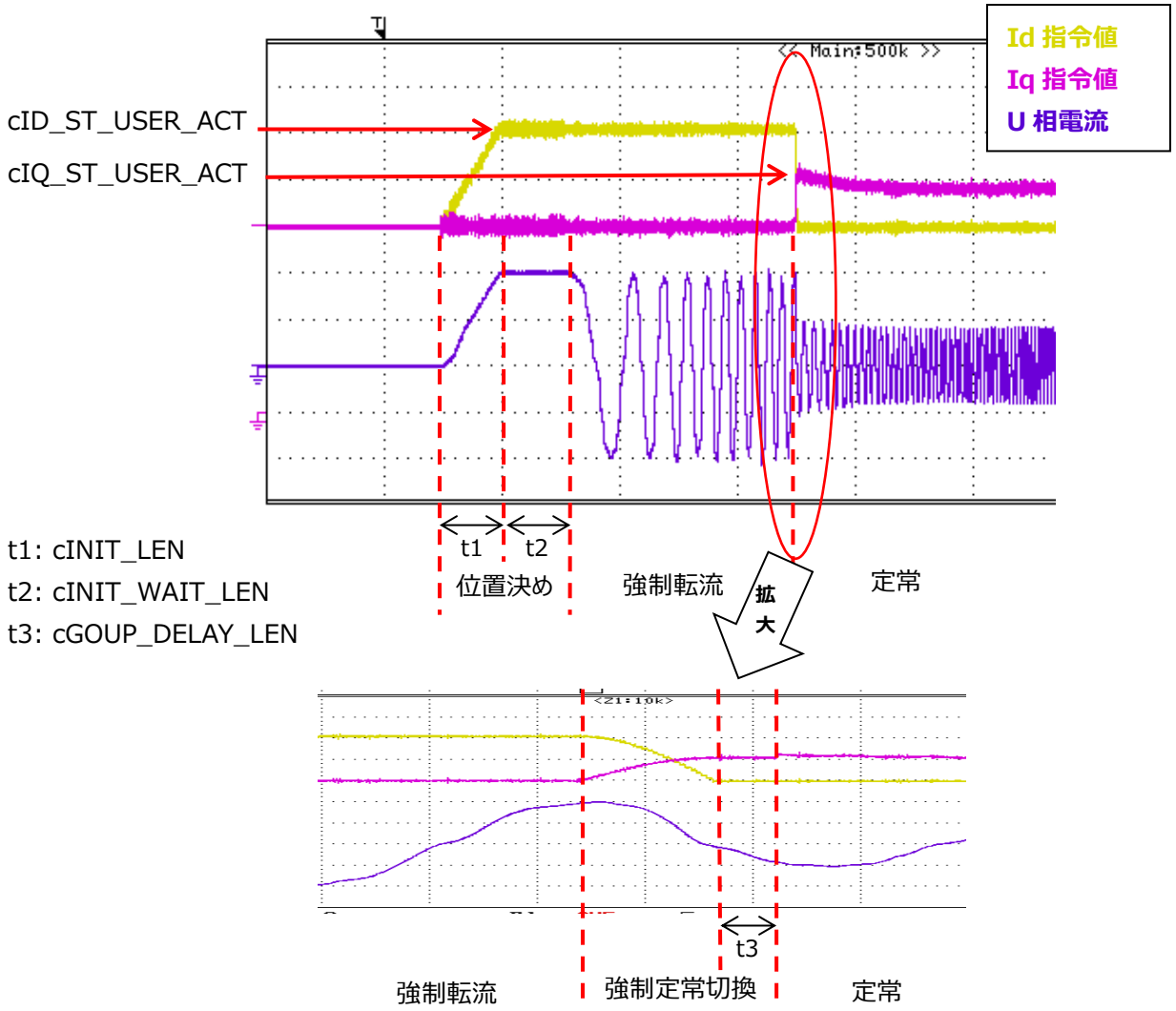


図 19 起動電流波形（電気角 0°に位置決め）

強制定常切換ステージで、cID_ST_USER_ACT と cIQ_ST_USER_ACT の値を更新入れ換える。
入れ換え後、cGOUP_DELAY_LEN の時間 Iq 指令値一定値で制御します。
定常移行後、Iq 指令値は PI 制御により演算されます。

© 2024
Toshiba Electronic Devices & Storage Corporation

10.2.3.2 電圧型起動時の定数値

オシロスコープなどで電流波形を確認しながら、定数の値を決めてください。

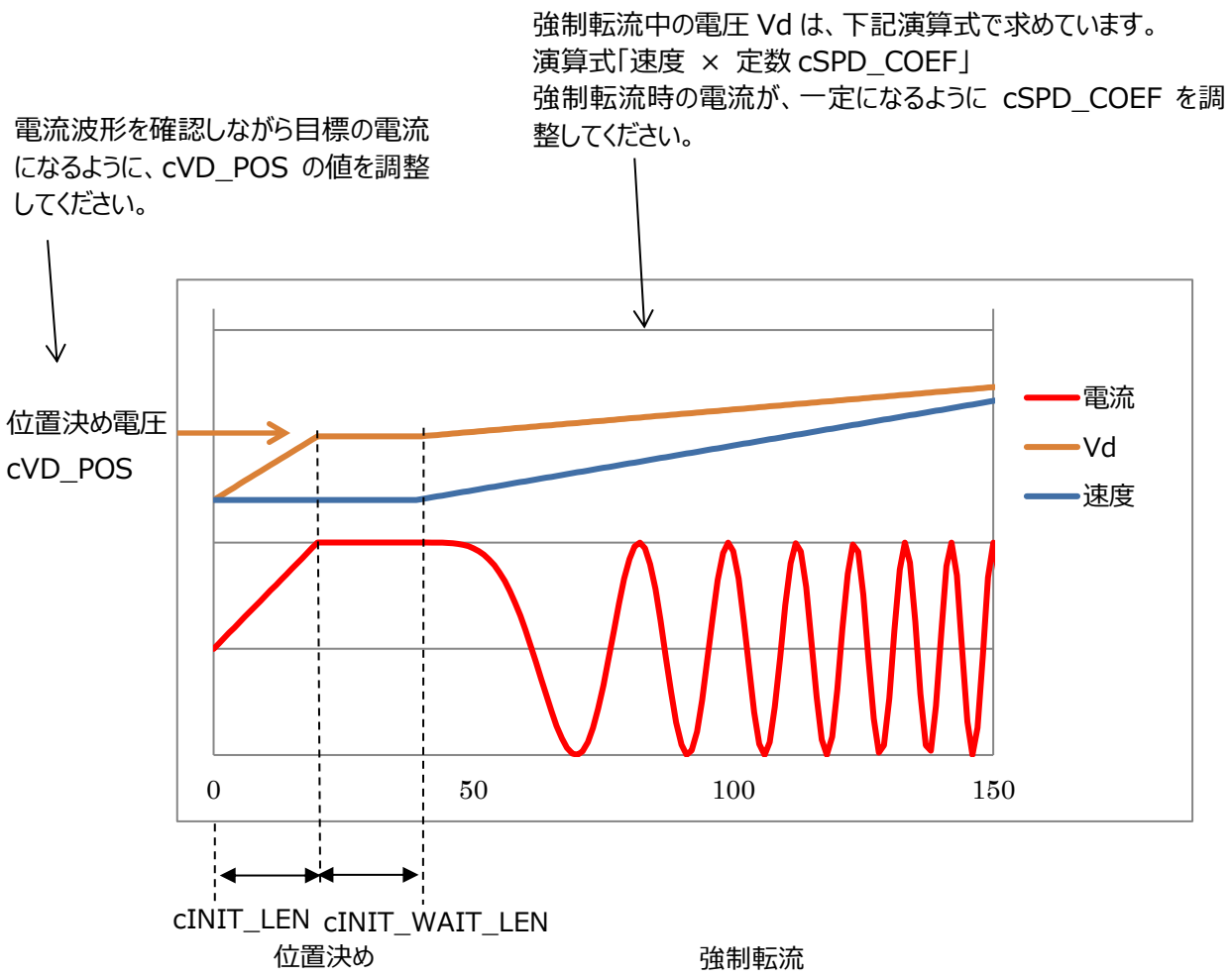


図 20 電圧駆動時の引数：調整方法

10.3 ユーザー制御関連定数

usercon.c

```
#define c2000MS_TIMER  (2000)   /* [ms] (4 kHz * 4) * 2000 */
  通信判定 2s タイマー

/* UART Setting */
#define UART_ch     UART0           /* UART Channel */
        UART チャネル：CH0
#define INTERRUPT_TX INTSC0TX_IRQn      /* UART Interrupt request */
        UART0 送信割り込み要求
#define INTERRUPT_RX INTSC0RX_IRQn      /* UART Interrupt request */
        UART0 受信割り込み要求
#define cSEND_DATA_NUM        (7)       /* Send data size */
        送信データ数
#define cRECEIVE_DATA_NUM       (6)       /* Receive data size */
        受信データ数
#define cUART_RECEIVE_WAIT   (0x00)       /* UART mode : data receive wait */
        UART モード：受信完了待ち
#define cUART_ERR          (0x01)       /* UART mode : error */
        UART モード：通信エラー
#define cREQ_SYSTEM_START       (0x10)       /* System start request */
        システム起動要求コマンド
#define cREQ_ROTATE_MOTOR       (0x11)       /* Target speed update request */
        目標速度更新要求コマンド
#define cREQ_CHANGE_MOTOR       (0x12)       /* Control Rpm request */
        モーター切り替え要求コマンド
#define  cREQ_ON_OFF_PFC        (0x13)       /* Control PFC request */
        PFC On/Off 要求コマンド
#define  cREQ_ALL_STOP_MOTOR   (0x14)       /* Stop PFC request */
        モーター、PFC 停止要求コマンド
#define  cREQ_STATUS_CH         (0x15)       /* Change channel request */
        状態表示切替要求コマンド
#define  cREQ_STATUS_DAC        (0x16)       /* Chnage Dac mode request */
        状態表示要求コマンド
#define cGET_MOTOR_ENABLE       (0x80)       /* Operating status */
        モーター動作 EN/DI 取得コマンド(※本システムでは未使用)
#define cGET_STATE_EMG          (0x81)       /* Emergency status */
        EMG ステータス取得コマンド
#define cGET_STAGE              (0x82)       /* Main stage */
        メインステージ取得コマンド
#define cGET_CONTROL_CH         (0x83)       /* Control channel */
```

© 2024
Toshiba Electronic Devices & Storage Corporation

```
        モーター制御 CH 取得コマンド
#define cGET_CARRIER_FREQUENCY   (0x84)      /* Carrier frequency */
        キャリアー周波数取得コマンド
#define cGET_MOTOR_SPEED_MIN  (0x85)      /* Minimum rotation speed */
        最小回転数取得コマンド
#define cGET_MOTOR_SPEED_MAX  (0x86)      /* Maximum rotation speed */
        最大回転数取得コマンド
#define cGET_DEAD_TIME        (0x87)      /* Dead time */
        デッドタイム取得コマンド
#define cGET_GATE_ACTIVE      (0x88)      /* Gate active */
        ゲートアクティブ取得コマンド
#define cGET_POSITION_DITECT(0x89)      /* Position detect */
        シャントタイプ取得コマンド
#define cGET_VDC_VOLTAGE      (0x8A)      /* VDC voltage */
        電源電圧取得コマンド
#define cGET_INVETER_TEMP     (0x8B)      /* Inverter temp */
        温度取得コマンド
#define cGET_U_VOLTAGE        (0x8C)      /* U-phase voltage */
        U 相電圧取得コマンド
#define cGET_V_VOLTAGE        (0x8D)      /* V-phase voltage */
        V 相電圧取得コマンド
#define cGET_W_VOLTAGE        (0x8E)      /* W-phase voltage */
        W 相電圧取得コマンド
#define cGET_DAC_DATA         (0x8F)      /* Daca date */
        DAC データ取得コマンド
#define cGET_INTERNAL_AMP     (0x90)      /* Internal amp */
        内蔵アンプ取得コマンド
#define cGET_DIRECTION        (0x91)      /* Direction */
        回転方向取得コマンド
#define cGET_MODULATION       (0x92)      /* Modulation */
        モデュレーション取得コマンド
#define cGET_MOTOR_SPEED      (0x94)      /* motor rotation speed */
        モーター回転数取得コマンド
#define  cGET_OUTPIPE_TEMP    (0x95)      /* outpipe templeture */
        パイプ温度取得コマンド
#define  cGET_EXHAUST_TEMP    (0x96)      /* exhaust templeture */
        熱排出温度取得コマンド
#define  cGET_DIODE_TEMP      (0x97)      /* diode templeture */
        PFC ダイオード温度取得コマンド
#define  cGET_IGBT_TEMP       (0x98)      /* IGBT templeture */
        PFC IGBT 温度取得コマンド
```

© 2024
Toshiba Electronic Devices & Storage Corporation

```
#define  cGET_HVMOS_TEMP        (0x99)      /* HVMOS templeture */
        MOSFET 温度取得コマンド
#define cEMG_STATE_EMG_H     (0x00)      /* emergency status : over detect(hard) */
        EMG ステート：ハード EMG
#define cEMG_STATE_EMG_S     (0x01)      /* emergency status : over detect(soft) */
        EMG ステート：ソフト EMG
#define cEMG_STATE_EMG_I        (0x02)      /* emergency status : curret detect error */
        EMG ステート：電流検出異常
#define cEMG_STATE_EMG_DC       (0x03)      /* emergency status : over vdc */
        EMG ステート：VDC 異常
#define cGATE_ACTIVE_H_H        (0)         /* Gate active : H/H */
        ゲートアクティブ：H/H
#define cGATE_ACTIVE_L_L        (1)         /* Gate active : L/L */
        ゲートアクティブ：L/L
#define cGATE_ACTIVE_H_L        (2)         /* Gate active : H/L */
        ゲートアクティブ：H/L
#define cGATE_ACTIVE_L_H        (3)         /* Gate active : L/H */
        ゲートアクティブ：L/H
#define cPOSITION_DETECT_3SHUNT  (0)        /* Position Ditect : 3shunt */
        シャントタイプ： 3 シャント(※本システムでは未使用)
#define cPOSITION_DETECT_1SHUNT  (1)        /* Position Ditect : 1shunt */
        シャントタイプ： 1 シャント(※本システムでは未使用)
#define cDAC_DATA_TMPREG0       (0x00)      /* Dac data : TMPREG0 */
        DAC データ：U 相電流
#define cDAC_DATA_TMPREG1       (0x01)      /* Dac data : TMPREG1 */
        DAC データ：V 相電流
#define cDAC_DATA_TMPREG2       (0x02)      /* Dac data : TMPREG2 */
        DAC データ：W 相電流
#define cDAC_DATA_THETAHALF(0x03)       /* Dac data : theta.half[1] */
        DAC データ：電気角
#define cDAC_DATA_IDREF         (0x04)      /* Dac data : Id_ref */
        DAC データ：Id リファレンス(目標値)
#define cDAC_DATA_ID            (0x05)      /* Dac data : Id */
        DAC データ：Id(現在値)
#define cDAC_DATA_IQREF         (0x06)      /* Dac data : Iq_ref */
        DAC データ：Iq リファレンス(目標値)
#define cDAC_DATA_IQ            (0x07)      /* Dac data : Iq */
        DAC データ：Iq(現在値)
#define cDAC_DATA_OMEGACOMHALH (0x08)       /* Dac data : omega_com.half[1] */
        DAC データ：角速度(目標値)
#define cDAC_DATA_OMEGAHALF    (0x09)       /* Dac data : omega.half[1] */
```

© 2024
Toshiba Electronic Devices & Storage Corporation

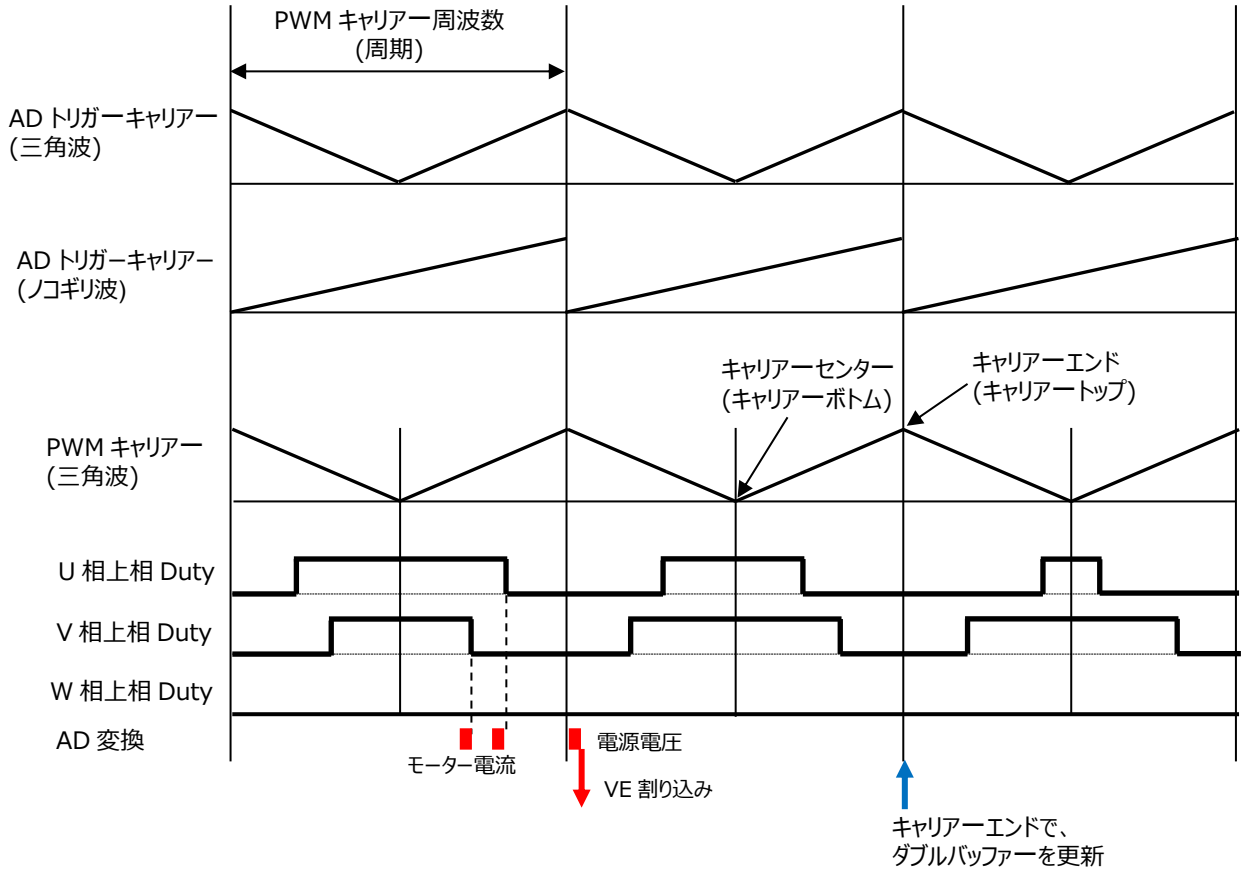
```
        DAC データ：角速度(現在値)
#define cDAC_DATA_OMEGADEV (0x0A)       /* Dac data：omega_dev */
        DAC データ：角速度(差)
#define cINTERNAL_AMP_NO     (0)        /* Internal amp：External */
        内蔵アンプ：未使用
#define cINTERNAL_AMP_YES    (1)        /* Internal amp：Internal */
        内蔵アンプ：使用
#define cDIRECTION_CW            (0)        /* Direction：plus */
        回転方向：CW
#define cDIRECTION_CCW       (1)        /* Direction：minus */
        回転方向：CCW(※本システムでは未使用)
#define  cMOTOR_CTRL_FAN         (0)        /* motor control：fan */
        モーター定義：Fan モーター
#define  cMOTOR_CTRL_COMP        (1)        /* motor control：compressor */
        モーター定義：Conpressor モーター
#define  cMOTOR_CTRL_PFC         (2)        /* motor control：pfc */
        PFC 定義：PFC
```

© 2024
Toshiba Electronic Devices & Storage Corporation

11. 制御、データ更新のタイミング

11.1 VE 使用によるベクトル制御

11.1.1 1 シャント制御



11.1.1.1 キャリアー波形

種類	波形
PWM	3 相とも三角波
AD トリガー	のこぎり波

11.1.1.2 ダブルバッファー更新タイミング

データ	更新タイミング
PWM 周期（RATE）	キャリアーエンド
Duty 値（CMPU,CMPV,CMPW）	キャリアーエンド
トリガー位置（TRGCMPx）	キャリアーエンド
出力設定（MDOUT）	キャリアーエンド

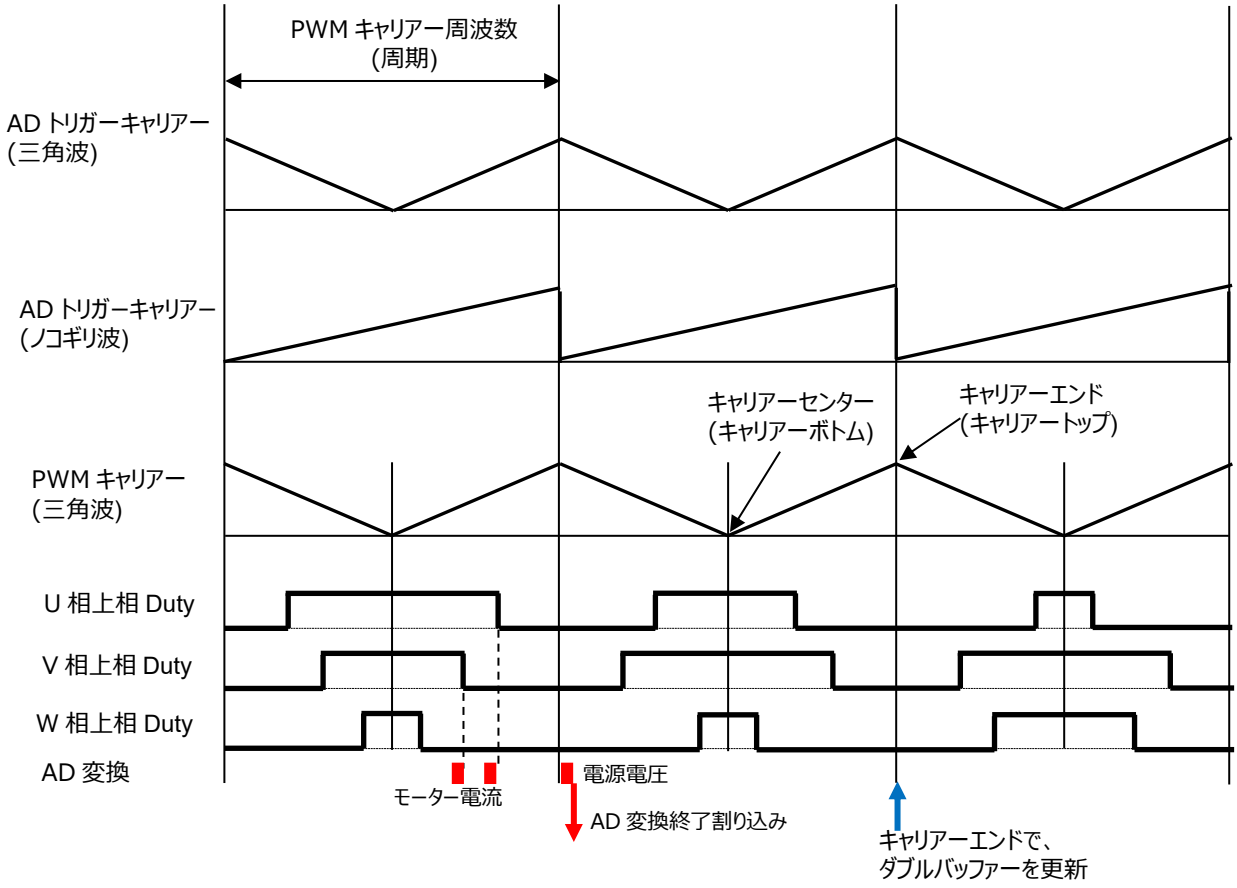
11.1.1.3 割り込み関連

割り込み要求	割り込み	タイミング
VE	許可	VE 入力スケジュール終了後
PWM(PMD)	禁止	1 回、2 回、4 回ごと
AD 変換終了	禁止	電源電圧の AD 変換終了後
AD トリガー	—	PWM と同じ頻度

© 2024
Toshiba Electronic Devices & Storage Corporation

11.2 ソフトウェアによるベクトル制御

11.2.1 1 シャント制御



11.2.1.1 キャリアー波形

種類	波形
PWM	3 相とも三角波
AD トリガー	のこぎり波

11.2.1.2 ダブルバッファー更新タイミング

データ	更新タイミング
PWM 周期（RATE)	キャリアーエンド
Duty 値（CMPU,CMPV,CMPW)	キャリアーエンド
トリガー位置（TRGCMPx)	キャリアーエンド
出力設定（MDOUT)	キャリアーエンド

11.2.1.3 割り込み関連

割り込み要求	割り込み	タイミング
PWM(PMD)	禁止	1 回、2 回、4 回ごと
AD 変換終了	許可	電源電圧の AD 変換終了後
AD トリガー	—	PWM と同じ頻度

© 2024
Toshiba Electronic Devices & Storage Corporation

12. ペリフェラルドライバー

12.1 マイコン周辺回路アドレス

ペリフェラルドライバーAPI に渡すマイコン周辺回路レジスターのベースアドレスを定義します。

12.1.1 データ構造

12.1.1.1 **Ipdrv_t**

Data Fields

TSB_VE_TypeDef* const **VEx**：VE アドレスを選択します。

TSB_PMD_TypeDef* const **PMDx**：PMD アドレスを選択します。

TSB_AD_TypeDef* const **ADx**：AD アドレスを選択します。

12.2 ベクトルエンジン(VE)

12.2.1 関数仕様

12.2.1.1 **IP_VE_init**

VE 初期設定

API：

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

引数：

VEx：VE アドレスを選択します。

_initdata：VE 初期設定データ構造体

詳細は 12.2.2.1 VE_InitTypeDef を参照してください。

機能：

VE の初期設定を行います。

補足：

VE 停止、割り込み禁止で呼んでください。

戻り値：

なし

12.2.1.2 **VE_Start**

VE スタート

API：

VE_Start(const ipdrv_t* const _ipdrv)

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

機能：

VE を開始します。

補足：

© 2024
Toshiba Electronic Devices & Storage Corporation

特になし

戻り値：

なし

12.2.1.3 **VE_GetPhaseCurrent**

相電流の取得

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_ia：a 相電流を格納する変数アドレスを設定します。

_ib：b 相電流を格納する変数アドレスを設定します。

_ic：c 相電流を格納する変数アドレスを設定します。

機能：

各相の電流値を取得します。

a 相には U 相電流値、b 相には V 相電流値、c 相には W 相電流値が入ります。

補足：

特になし

戻り値：

なし

12.2.1.4 **VE_GetCurrentAdcData**

電流 AD 値取得

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_adc_ia：a 相電流 AD 値を格納する変数アドレスを設定します。

_adc_ib：b 相電流 AD 値を格納する変数アドレスを設定します。

_adc_ic：c 相電流 AD 値を格納する変数アドレスを設定します。

機能：

IAADCx、IBADCx、ICADCx レジスターの値を各相の電流 AD 値に取得します。

補足：

特になし

戻り値：

なし

© 2024
Toshiba Electronic Devices & Storage Corporation

12.2.1.5 **VE_GetdataFromVEreg**

VE レジスターデータ取得

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_motor：ベクトル制御変数の構造体アドレスを選択します。

機能：

VE レジスターからモーター電源電圧 Vdc、d 軸電圧 Vd、q 軸電圧 Vq、d 軸電流 Id、q 軸電流 Iq の値をベクトル制御の各変数へ格納します。

Duty 幅により電流が検出できないタイミングでは、d 軸電流 Id、q 軸電流 Iq の値を VE レジスターへ前回検出値を書きこみます。

補足：

特になし

戻り値：

なし

12.2.1.6 **VE_GetPWM_DutyU**

U 相 Duty 値取得

API:

uint32_t　VE_GetPWM_DutyU(const ipdrv_t* const _ipdrv)

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

機能:

U 相 Duty レジスターの値を取得します。

補足:

特になし

戻り値:

U 相 Duty

12.2.1.7 **VE_GetPWM_DutyV**

V 相 Duty 値取得

API:

uint32_t　VE_GetPWM_DutyV(const ipdrv_t* const _ipdrv)

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

機能:

V 相 Duty レジスターの値を取得します。

補足:

© 2024
Toshiba Electronic Devices & Storage Corporation

特になし

戻り値:

V 相 Duty

12.2.1.8 **VE_GetPWM_DutyW**

W 相 Duty 値取得

API:

uint32_t　VE_GetPWM_DutyW(const ipdrv_t* const _ipdrv)

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

機能:

W 相 Duty レジスターの値を取得します。

補足:

特になし

戻り値:

W 相 Duty

12.2.1.9 **VE_GetPWM_DutyMed**

Duty 中間値取得

API：

uint32_t VE_GetPWM_DutyMed(const ipdrv_t* const _ipdrv)

引数：

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

機能：

U、V、W 相 Duty 値の中間の値を取得します。

補足：

特になし

戻り値：

Duty 中間値

12.2.1.10 **VE_GetPWM_Modulation**

変調方式取得

API:

int　VE_GetPWM_Modulation(const ipdrv_t* const _ipdrv)

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

機能:

変調方式を取得します。

補足:

特になし

© 2024
Toshiba Electronic Devices & Storage Corporation

戻り値:

変調方式 0:3 相変調、1:2 相変調

12.2.1.11 **VE_GetSector**

現在セクター取得

API:

uint32_t VE_GetSector(const ipdrv_t* const _ipdrv)

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

機能:

現在のセクター値を取得します。

補足:

特になし

戻り値:

現在セクター値 [0-11]

12.2.1.12 **VE_GetShiftPWMState**

シフト PWM 状態取得

API:

int VE_GetShiftPWMState(const ipdrv_t* const _ipdrv)

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

機能:

シフト PWM 状態を取得します。

補足:

特になし

戻り値:

シフト PWM 状態 0:通常 PWM、1:シフト PWM

12.2.1.13 **VE_GetOutPutMode**

PWM 出力状態取得

API :

int VE_GetOutputMode(const ipdrv_t* const _ipdrv)

引数 :

_ipdrv : マイコン周辺回路アドレスが定義された構造体を指定します。

機能 :

PWM 出力状態を取得。

補足 :

特になし

戻り値 :

© 2024
Toshiba Electronic Devices & Storage Corporation

PWM 状態 OCRMD_OUT_OFF：出力 OFF
OCRMD_OUT_ON：出力 ON
OCRMD_OUT_ON_LOWPH：下相のみ ON

12.2.1.14 **VE_SetdataToVEreg_Stop**

VE レジスターへのデータセット(Stop)

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。
_motor：ベクトル制御変数の構造体アドレスを設定します。

機能：

停止状態の VE レジスターへの設定処理を行います。
電流制御ゲイン積分値レジスターなどの初期化を行います。

補足：

特になし

戻り値：

なし

12.2.1.15 **VE_SetdataToVEreg_Bootstrap**

VE レジスターへのデータセット(Bootstrap)

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。
_motor：ベクトル制御変数の構造体アドレスを設定します。

機能：

ブートストラップ状態の VE レジスターへの設定処理を行います。
VE では、2 相変調かつ出力電圧を 0 とすることで、L 側だけ ON となる波形を出力します。

補足：

特になし

戻り値：

なし

12.2.1.16 **VE_SetdataToVEreg_Initposition_i**

VE レジスターへのデータセット(Initposition 電流制御タイプ)

API：

© 2024
Toshiba Electronic Devices & Storage Corporation

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_motor：ベクトル制御変数の構造体アドレスを設定します。

機能：

電流制御型の位置決め状態の VE レジスターへの設定処理を行います。

補足：

特になし

戻り値：

なし

12.2.1.17 **VE_SetdataToVEreg_Initposition_v**

VE レジスターへのデータセット(Initposition 電圧制御タイプ)

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_motor：ベクトル制御変数の構造体アドレスを設定します。

機能：

電圧制御型の位置決め状態の VE レジスターへの設定処理を行います。

補足：

特になし

戻り値：

なし

12.2.1.18 **VE_SetdataToVEreg_Force_i**

VE レジスターへのデータセット(強制転流　電流制御タイプ)

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_motor：ベクトル制御変数の構造体アドレスを設定します。

機能：

電流制御型の強制転流状態の VE レジスターへの設定処理を行います。

補足：

© 2024
Toshiba Electronic Devices & Storage Corporation

特になし

戻り値：

なし

12.2.1.19 **VE_SetdataToVEreg_Force_v**

VE レジスターへのデータセット(強制転流 電圧制御タイプ)

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_motor：ベクトル制御変数の構造体アドレスを設定します。

機能：

電圧制御型の強制転流状態の VE レジスターへの設定処理を行います。

補足：

特になし

戻り値：

なし

12.2.1.20 **VE_SetdataToVEreg_Change_up**

VE レジスターへのデータセット(強制定常切換)

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_motor：ベクトル制御変数の構造体アドレスを設定します。

機能：

強制定常切換状態の VE レジスターへの設定処理を行います。

補足：

特になし

戻り値：

なし

12.2.1.21 **VE_SetdataToVEreg_Steady_A**

VE レジスターへのデータセット(定常)

API：

```
void
```

© 2024
Toshiba Electronic Devices & Storage Corporation

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_motor：ベクトル制御変数の構造体アドレスを設定します。

機能：

定常状態の VE レジスターへの設定処理を行います。

補足：

特になし

戻り値：

なし

12.2.1.22 **VE_SetdataToVEreg_Emergency**

VE レジスターへのデータセット(EMG)

API：

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

引数：

_ipdrv： マイコン周辺回路アドレスが定義された構造体を指定します。

_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：

Emergency 状態の VE レジスターへの設定処理を行います。

補足：

特になし

戻り値：

なし

12.2.1.23 **VE_SetZeroCurrentData**

ゼロ電流設定

API：

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

引数：

_ipdrv： マイコン周辺回路アドレスが定義された構造体を指定します。

_z_ia：a 相ゼロ電流 AD 値を設定します。

_z_ib：b 相ゼロ電流 AD 値を設定します。

_z_ic：c 相ゼロ電流 AD 値を設定します。

機能：

ゼロ電流時の AD 値を VE レジスターに設定します。

© 2024
Toshiba Electronic Devices & Storage Corporation

補足：

特になし

戻り値：

なし

12.2.1.24 **VE_SetVDCreg**

DC リンク電圧 Vdc の設定

API：

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

引数：

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

_dat: モーターの電源電圧を設定します。

機能：

モーターの電源電圧を VE レジスターに設定します。

補足：

特になし

戻り値：

なし

12.2.1.25 **VE_SetSpwmMode**

シフト PWM モード設定

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

_dat: シフト PWM モードを設定します。

機能:

シフト PWM モード状態を VE レジスターに設定します。

補足:

特になし

戻り値:

なし

12.2.1.26 **VE_SetModulType**

変調方式設定

API：

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

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

_dat：変調方式を設定します。

© 2024
Toshiba Electronic Devices & Storage Corporation

機能：

変調方式を VE レジスターに設定します。

補足：

特になし

戻り値：

なし

12.2.2 データ構造

12.2.2.1 VE_InitTypeDef

Data Fields：

uint8_t shunt：シャントタイプ
1：1 シャント
uint16_t reptime：リピート回数(1 ～ 15)
uint16_t trgmode：起動トリガー
TRGMODE_UNITA：ADCA PMD0 トリガー同期変換終了割り込みで起動
TRGMODE_UNITB：ADCB PMD1 トリガー同期変換終了割り込みで起動
uint16_t tpwm：PWM 周期時間(位相補間用)
VETPWM レジスターに設定する値
uint16_t idkp：d 軸電流制御比例ゲイン
uint16_t idki：d 軸電流制御積分ゲイン
uint16_t iqkp：q 軸電流制御比例ゲイン
uint16_t iqki：q 軸電流制御積分ゲイン
uint16_t zerooffset：ゼロ電流オフセット
uint16_t **trgcomp0:** トリガータイミング 0 設定
VExTRGCOMP0 に設定する値
uint16_t **trgcomp1:** トリガータイミング 1 設定
VExTRGCOMP1 に設定する値
uint32_t **fmode:** フロー制御設定
VExFMODE に設定する値
uint16_t **fpwmchg:** PWM 切り替え速度設定
VExFPWMCHG に設定する値

12.3 モーター制御回路(PMD)

12.3.1 関数仕様

12.3.1.1 IP_PMD_init

PMD 初期設定

API：

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

© 2024
Toshiba Electronic Devices & Storage Corporation

引数：

PMDx：PMD アドレスを選択します。

_initdata：PMD 初期設定データ構造体

詳細は 12.3.2.1 PMD_InitTypeDef を参照してください。

機能：

PMD の初期設定を行います

補足：

PMD 停止、割り込み禁止で呼んでください。

戻り値：

なし

12.3.1.2 **PMD_GetEMG_Status**

EMG 保護状態取得

API：

emg_status_e PMD_GetEMG_Status(const ipdrv_t* const _ipdrv)

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

機能：

EMG 保護状態を取得します。

補足：

特になし

戻り値：

emg_status_e：EMG 保護状態

cNormal：正常

cEMGProtected：EMG 発生による PWM 出力の禁止

12.3.1.3 **PMD_ReleaseEMG_Protection**

EMG 保護状態解除

API：

void PMD_ReleaseEMG_Protection(const ipdrv_t* const _ipdrv)

引数：

_ipdrv：マイコン周辺回路アドレスが定義された構造体を指定します。

機能：

EMG 保護状態を解除します。

補足：

この関数を呼んでも、MDOUT が 0 かつ EMG ポートが H の状態でないと、保護状態は解除されません。

戻り値：

なし

© 2024
Toshiba Electronic Devices & Storage Corporation

12.3.2 データ構造

12.3.2.1 **PMD_InitTypeDef**

Data Fields :

uint8_t **shunt :** シャントタイプ
1：1 シャント

uint8_t **poll :** L 側極性
0：L active
1：H active

uint8_t **polh :** H 側極性
0：L active
1：H active

uint16_t **pwmrate :** PWM 周波数
PMDxRATE に設定する値

uint16_t **deadtime :** デッドタイム時間
PMDxDTR に設定する値

uint8_t **busmode:** モード選択
PMDxMODESEL に設定する値

12.4 アナログデジタルコンバーター(ADC)

12.4.1 関数仕様

12.4.1.1 **IP_ADC_init**

ADC 初期設定

API :

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

引数 :

ADx : ADC アドレスを選択します。

_initdata : ADC 初期設定データ構造体
詳細は 12.4.2.1 AD_InitTypeDef を参照してください。

機能 :

ADC の初期設定を行います

補足 :

ADC 停止、割り込み禁止で呼んでください。

戻り値 :

なし

12.4.2 データ構造

12.4.2.1 **AD_InitTypeDef**

Data Fields :

© 2024
Toshiba Electronic Devices & Storage Corporation

uint8_t shunt：シャントタイプ
1：1 シャント
uint8_t iuch：U 相電流取り込み AD チャネル番号(3 シャント用)
uint8_t ivch：V 相電流取り込み AD チャネル番号(3 シャント用)
uint8_t iwch：W 相電流取り込み AD チャネル番号(3 シャント用)
uint8_t idcch：DC 電流取り込み AD チャネル番号(1 シャント用)
uint8_t vdcch：モーター電源電圧 Vdc 取り込み AD チャネル番号
uint8_t pmd_ch：使用する PMD チャネル選択
cPMD0：PMD チャネル 0
cPMD1：PMD チャネル 1
cPMD2：PMD チャネル 2
uint8_t pints：PMD トリガー用割り込み選択
cPINTS_A：ITTADxPDA
cPINTS_B：ITTADxPDB
cPINTS_C：ITTADxPDC
cPINTS_D：ITTADxPDD
uint32_t** **p_adreg0**: AD 変換結果格納レジスターアドレス 0
uint32_t** **p_adreg1**: AD 変換結果格納レジスターアドレス 1
uint32_t** **p_adreg2**: AD 変換結果格納レジスターアドレス 2
uint32_t** **p_adreg3**: AD 変換結果格納レジスターアドレス 3

12.5 定数説明

12.5.1 A-VE+搭載マイコン用定数（mcuip_drv.h)

12.5.1.1 A-PMD

MDCR レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cWPWMMD	W 相 PWM キャリアー波形	PWMMD_SAWTOOTH	ノコギリ波(エッジ PWM)
		PWMMD_TRIANGULAR	三角波(センターPWM)
		PWMMD_SAWTOOTH_REV	逆ノコギリ波(エッジ PWM)
		PWMMD_TRIANGULAR_REV	逆三角波(センターPWM)
cVPWMMD	V 相 PWM キャリアー波形	PWMMD_SAWTOOTH	ノコギリ波(エッジ PWM)
		PWMMD_TRIANGULAR	三角波(センターPWM)
		PWMMD_SAWTOOTH_REV	逆ノコギリ波(エッジ PWM)
		PWMMD_TRIANGULAR_REV	逆三角波(センターPWM)
cUPWMMD	U 相 PWM キャリアー波形	PWMMD_SAWTOOTH	ノコギリ波(エッジ PWM)
		PWMMD_TRIANGULAR	三角波(センターPWM)
		PWMMD_SAWTOOTH_REV	逆ノコギリ波(エッジ PWM)
		PWMMD_TRIANGULAR_REV	逆三角波(センターPWM)
cDSYNCS	PWM デューティー コンペアレジスターの 実行バッファー更新 タイミング	DSYNCS_INTCYC	割り込み周期設定
		DSYNCS_CENTER	相別 PWM センターで更新
		DSYNCS_END	相別 PWM エンドで更新
		DSYNCS_CENTER_END	相別 PWM エンドとセンターで更新
cDTCREN	デッドタイム	DTCREN_DISABLE	補正禁止

© 2024
Toshiba Electronic Devices & Storage Corporation

	補正許可	DTCREN_ENABLE	補正許可
cDCMEN	実行バッファー更新/トリガー出力の間引き制御	DCMEN_DISABLE	禁止
		DCMEN_ENABLE	許可
cPWMSYNT	ポート出力モード設定	SYNTMD_0	
		SYNTMD_1	
cPWMSPCFY	デューティーモード選択	DTYMD_COMMON	3 相共通
		DTYMD_INDEPENDENT	3 相独立
cPWMINT	PWM 割り込み要求タイミング選択	PINT_CENTER	U 相 PWM センターで割り込み要求発生
		PINT_END	U 相 PWM エンドで割り込み要求発生
cPWMINTPRD	PWM 割り込み要求周期選択	INTPRD_HALF	PWM 0.5 周期ごとに割り込み要求
		INTPRD_1	PWM 1 周期ごとに割り込み要求
		INTPRD_2	PWM 2 周期ごとに割り込み要求
		INTPRD_4	PWM 4 周期ごとに割り込み要求

MDPOT レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cSYNCS	MDOUT 設定転送タイミング選択	SYNCS_ASYNC	非同期
		SYNCS_INTENC	A-ENCx 割り込み要求発生時
		SYNCS_NTTB	汎用タイマー割り込み要求発生時
		SYNCS_CTRGO	A-ENCx MCMP 成立時
cPSYNCS	MDOUT、MDCR 実行バッファー更新タイミング選択	PSYNCS_WRITE	PWM 非同期(書き込み時に反映)
		PSYNCS_CENTER	相別 PWM センター
		PSYNCS_END	相別 PWM エンド
		PSYNCS_CENTER_END	相別 PWM エンドおよびセンター

PORTMD レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cPORTMD	デバッグホールト時のポート出力制御	PORTMD_ALLHIZ	上相ハイインピーダンス / 下相ハイインピーダンス
		PORTMD_UHIZ_LON	上相ハイインピーダンス / 下相 PMD 出力
		PORTMD_UON_LHIZ	上相 PMD 出力 / 下相ハイインピーダンス
		PORTMD_ALLON	上相 PMD 出力 / 下相 PMD 出力

TRGCR レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cTRGMD_3SHT	3 シャント時のトリガータイミング	TRGMD_DIS	トリガー出力禁止
		TRGMD_TRI_FIRST_HALF	三角波キャリアー時の PWM 前半のコンペア成立でトリガー出力
		TRGMD_TRI_SECND_HALF	三角波キャリアー時の PWM 後半のコンペア成立でトリガー出力
		TRGMD_TRI_F_S_CMP	三角波キャリアー時の PWM 前半/後半のコンペア成立でトリガー出力
		TRGMD_END	PWM エンドタイミングでトリガー出力
		TRGMD_CENTER	PWM センタータイミングでトリガー出力
		TRGMD_ENDCENTER	PWM エンドタイミング/センタータイミングでトリガー出力
		TRGMD_DIS7	トリガー出力禁止

© 2024
Toshiba Electronic Devices & Storage Corporation

cTRGMD_1SHT	1 シャント時のトリガータイミング	TRGMD_DIS	トリガー出力禁止
		TRGMD_TRI_FIRST_HALF	三角波キャリアー時の PWM 前半のコンペア成立でトリガー出力
		TRGMD_TRI_SECND_HALF	三角波キャリアー時の PWM 後半のコンペア成立でトリガー出力
		TRGMD_TRI_F_S_CMP	三角波キャリアー時の PWM 前半/後半のコンペア成立でトリガー出力
		TRGMD_END	PWM エンドタイミングでトリガー出力
		TRGMD_CENTER	PWM センタータイミングでトリガー出力
		TRGMD_ENDCENTER	PWM エンドタイミング/センタータイミングでトリガー出力
		TRGMD_DIS7	トリガー出力禁止
cBUFSYNC	実行バッファーの非同期更新許可	TRGBE_SYNC	同期更新
		TRGBE_ASYNC	非同期更新
cCARSEL	比較キャリアーの選択	CARSEL_BASE	基本キャリアーと比較
		CARSEL_PHASE	相別キャリアーと比較

TRGMD レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cEMGTGE	EMG 保護動作中の出力許可設定	EMGTGE_DISABLE	保護動作時トリガー出力禁止
		EMGTGE_ENABLE	保護動作時トリガー出力許可

EMGCR レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cEMGCNT	EMG 入力検出時間	0 ～ 31	異常検出入力(EMG)のノイズ除去時間を設定 cEMGCNT × 16/fsys "00000"設定時はノイズフィルターをスルー
cINHEN	デバッグホールト時の PMD の許可/禁止	INHEN_DISABLE	禁止
		INHEN_ENABLE	許可
cEMGMD	EMG 保護モード選択	EMGMD_ALLHIZ	全相ハイインピーダンス
		EMGMD_UON_LHIZ	全上相オン／全下相ハイインピーダンス
		EMGMD_UHIZ_LON	全上相ハイインピーダンス／全下相オン
		EMGMD_ALLHIZ2	全相ハイインピーダンス

OVVCR レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cOVVCNT	OVV 入力検出時間	0 ～ 31	OVV のノイズ除去時間を設定 cOVVCNT × 16/fsys "00000"設定時はノイズフィルターをスルー
cADIN1EN	ADC 監視機能 1 入力許可	ADIN1EN_DISABLE	入力禁止
		ADIN1EN_ENABLE	入力許可
cADIN0EN	ADC 監視機能 0 入力許可	ADIN0EN_DISABLE	入力禁止
		ADIN0EN_ENABLE	入力許可
cOVVMD	OVV 保護モード選択	OVVMD_NOCON	出力制御なし
		OVVMD_UON_LOFF	全上相オン、全下相オフ
		OVVMD_UOFF_LON	全上相オフ、全下相オン
		OVVMD_ALLOFF	全相オフ

© 2024
Toshiba Electronic Devices & Storage Corporation

cOVVISEL	OVV 端子入力制御	OVVISEL_PORT	ポート入力許可
		OVVISEL_ADC	ADC モニター信号

12.5.1.2 A-VE+

cTADC　　1 シャントシフト PWM 用 AD 変換時間　　を設定してください。

FMODE レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cSPWMMD	PWM シフトモード選択	SPWMMD_SPWM1	シフト 1
		SPWMMD_SPWM2U	シフト 2 (U 相センター)
		SPWMMD_SPWM2V	シフト 2 (V 相センター)
		SPWMMD_SPWM2W	シフト 2 (W 相センター)
cCCVMD	相変換モード選択	CCVMD_RELATIVE	相対変換
		CCVMD_ABSOLUTE	絶対変換
cPHCVDIS	相変換禁止	PHCVDIS_ENABLE	2-3 相変換許可(3 相交流出力)
		PHCVDIS_DISABLE	2-3 相変換禁止(2 相交流出力)
cVSLIMMD	電圧スカラー制限制御	VSLIMMD_DIS	スカラー制限禁止
		VSLIMMD_D	d 軸方向に制限
		VSLIMMD_Q	q 軸方向に制限
		VSLIMMD_DQ	dq 軸方向に制限
cMREGDIS	SIN/COS/SECTOR レジスターおよび *[VExMCTLF]*の前回値フラグの使用	MREGDIS_EFFECTIVE	有効
		MREGDIS_NOEFFECTIVE	無効
cCRCEN	トリガー補正許可	CRCEN_DISABLE	トリガー補正禁止
		CRCEN_ENABLE	トリガー補正許可
cIAPLMD	電流(IA)検出方向設定	IPLMD_SHUNT	シャントモード (IA = VExIAO - VExIAADC)
		IPLMD_SENSOR	センサーモード (IA = VExIAADC - VExIAO)
cIBPLMD	電流(IB)検出方向設定	IPLMD_SHUNT	シャントモード (IB = VExIBO - VExIBADC)
		IPLMD_SENSOR	センサーモード (IB = VExIBADC - VExIBO)
cICPLMD	電流(IC)検出方向設定	IPLMD_SHUNT	シャントモード (IC = VExICO - VExICADC)
		IPLMD_SENSOR	センサーモード (IC = VExICADC - VExICO)
cIDQSEL	非干渉制御入力電流選択	IDQSEL_FEEDBACK	フィードバック電流 (ID and IQ)
		IDQSEL_INSTRUCTION	指令電流 (IDREF, IQREF)

MODE レジスター設定用

定数名	定数の意味	選択データ	選択データの意味
cIPDEN	電流極性判定制御	IPDEN_DISABLE	電流極性判定禁止
		IPDEN_ENABLE	電流極性判定許可
cPMDDTCEN	デッドタイム補償時、PMD 回路のデッドタイム補正への対応制御	PMDDTCEN_DISABLE	デッドタイム補正を無効
		PMDDTCEN_ENABLE	デッドタイム補正を有効
cPWMFLEN	PWM 上限設定時、デューティー100%出力を許可	PWMFLEN_DISABLE	100% 禁止
		PWMFLEN_ENABLE	100% 許可
cPWMBLEN	PWM 下限設定時、デューティー0%出力を許可	PWMBLEN_DISABLE	0% 出力禁止
		PWMBLEN_ENABLE	0% 出力許可

© 2024
Toshiba Electronic Devices & Storage Corporation

cNICEN	タスク 5 の非干渉制御の許可	NICEN_DISABLE	非干渉制御禁止
		NICEN_ENABLE	非干渉制御許可
cT5ECEN	タスク 5 の拡張制御(非干渉制御,電圧スカラー制限)の許可	T5ECEN_DISABLE	拡張制御(非干渉制御,電圧スカラー制限)禁止
		T5ECEN_ENABLE	拡張制御(非干渉制御,電圧スカラー制限)許可
cAWUMD	PI 制御出力制限時のアンチワインドアップ(AWU)制御	AWUMD_DISABLE	AWU 禁止
		AWUMD_ENABLE4	制限量 /4 を積分項に反映
		AWUMD_ENABLE2	制限量 /2 を積分項に反映
		AWUMD_ENABLE1	制限量を積分項に反映
cCLPEN	位相補間時の位相クリッピング制御	CLPEN_DISABLE	クリッピング禁止
		CLPEN_ENABLE	クリッピング許可
cATANMD	ATAN 演算制御	ATANMD_DISABLE	演算禁止
		ATANMD_IDQ	電流ベクトルの d-q 座標軸上の偏角算出
		ATANMD_EDQ	誘起電圧ベクトルの d-q 座標軸上の偏角算出
cVDCSEL	電源電圧保存レジスター選択	VDCSEL_VDC	*[VExVDC]*保存
		VDCSEL_VDCL	*[VExVDCL]*保存
cZIEN	ゼロ電流検出制御	ZIEN_DISABLE	通常電流検出
		ZIEN_ENABLE	ゼロ電流検出
cPVIEN	位相補間制御	PVIEN_DISABLE	禁止
		PVIEN_ENABLE	許可

© 2024
Toshiba Electronic Devices & Storage Corporation

13. UART 通信プロトコル

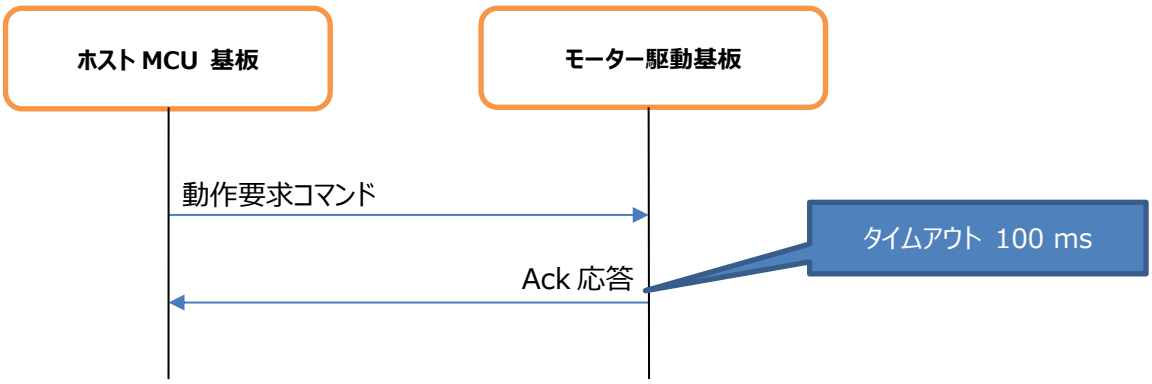
ホスト MCU 基板は、UART 通信プロトコルを使用してこのリファレンスデザイン（モーター駆動基板）を制御します。この UART 通信プロトコルについて説明します。

13.1 UART コマンド

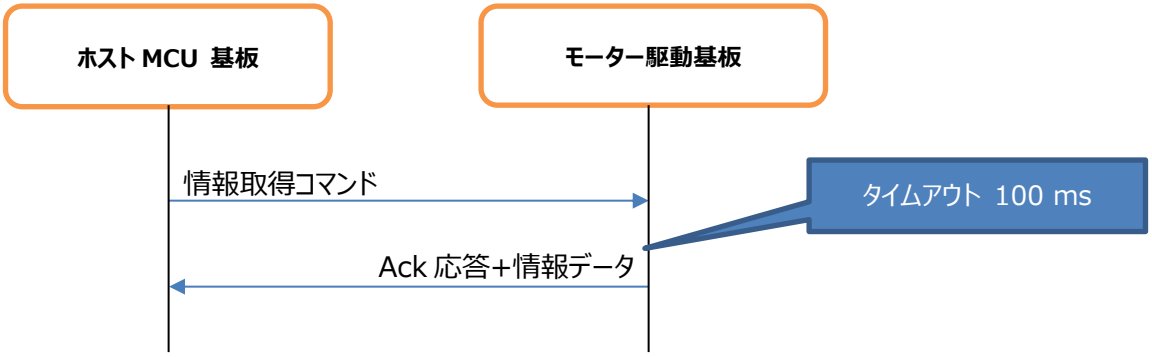
13.1.1 UART コマンドの種類

このリファレンスデザインでは、次の 2 種類の UART コマンドによって制御されます。それぞれのコマンドシーケンスを以下に示します。

コマンドタイプ 1： 動作要求コマンド送信



コマンドタイプ 2： 情報取得コマンド送信と情報データ受信



© 2024
Toshiba Electronic Devices & Storage Corporation

動作要求と情報取得の各コマンドの送信データフォーマットを以下に示します。

	Bit							
	7	6	5	4	3	2	1	0
Byte0	コマンド ID							
Byte1	データ 0（下位バイト）							
Byte2	データ 1							
Byte3	データ 2							
Byte4	データ 3（上位バイト）							
Byte5	CHECKSUM							

Ack 応答ならびに情報データの受信データフォーマットを以下に示します。

	Bit							
	7	6	5	4	3	2	1	0
Byte0	コマンド ID (受信コマンド ID)							
Byte1	0	0	0	0	0	EMG	0	ACK
Byte2	データ 0（下位バイト）							
Byte3	データ 1							
Byte4	データ 2							
Byte5	データ 3（上位バイト）							
Byte6	CHECKSUM							

コマンド ID：(1byte)

UART コマンド仕様を参照ください。

CHECKSUM：(1byte)

CHECKSUM は、CHECKSUM を除く全データ加算値の下位 8 ビットを表します。

モーター駆動基板は、受信データと CHECKSUM の値を確認し、Ack 応答で結果を返します。

ホスト MCU 基板は、受信データと CHECKSUM の値を確認し、誤りがあればデータは無効とします。

EMG bit：(1bit)

EMG 状態を判断します。

0: 通常状態(EMG 以外の状態)

1: EMG 状態

ACK bit：(1bit)

コマンド応答

0: 受付 NG (Nack)（CHECKSUM エラー時／無効コマンド時）

© 2024
Toshiba Electronic Devices & Storage Corporation

1: 受付 OK (Ack)

データ : (4byte)

UART コマンド仕様を参照ください。

13.1.2 UART 通信設定

このリファレンスデザインで使用されている UART 通信設定は次のとおりです。

- ボーレート: 9600 bps
- データ:8 ビット
- ストップビット: 1 ビット
- パリティ:なし
- フロー制御:なし

13.1.3 UART コマンド概要

コマンドタイプ	コマンド名	コマンド ID (1 バイト)	各状態での有効コマンド	
			初期状態 (リセット後)	通常状態
動作要求	REQ_SYSTEM_START	0x10	✓ (→通常状態)	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	✓
情報取得	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	✓

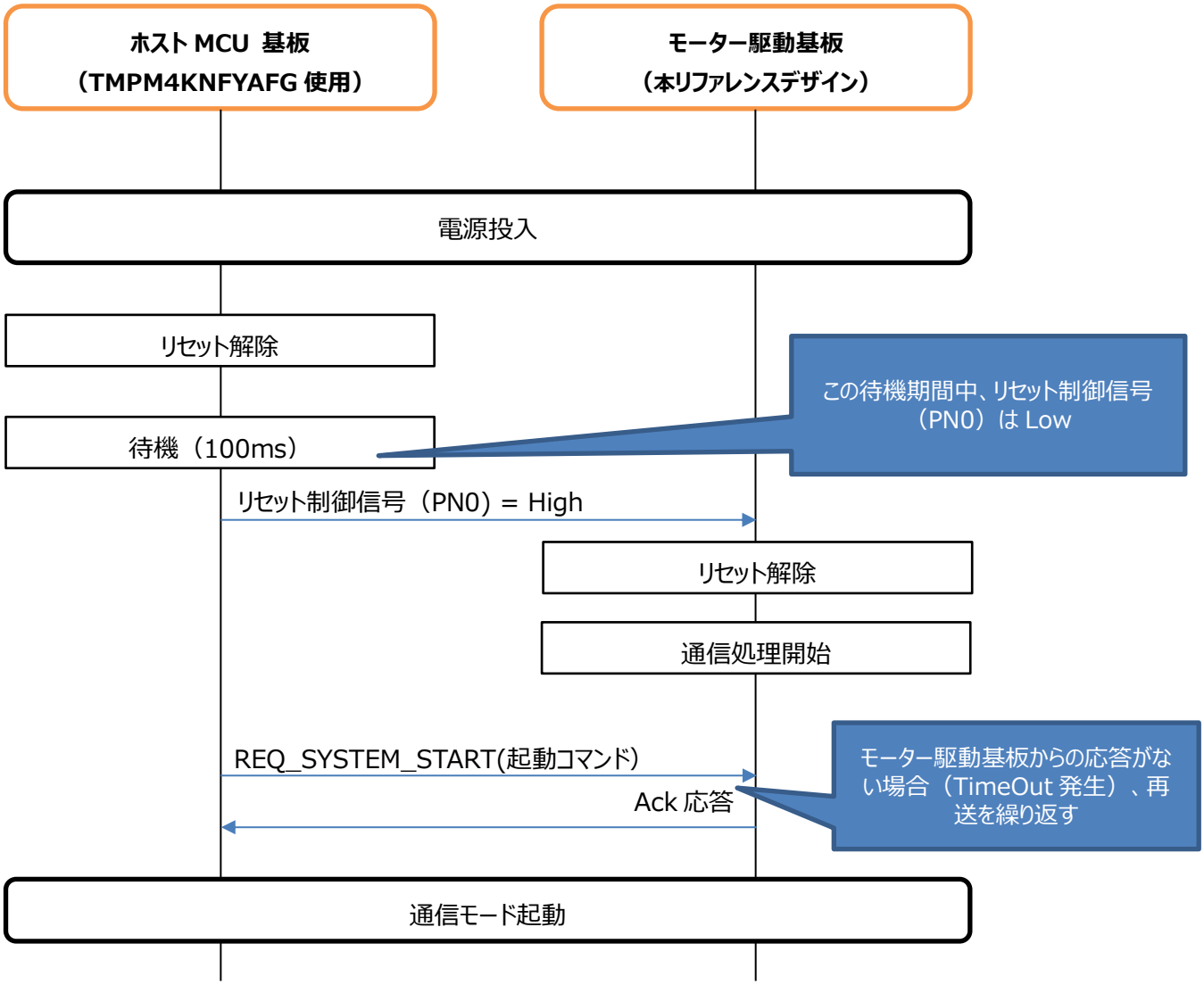
© 2024
Toshiba Electronic Devices & Storage Corporation

13.2 UART コマンドシーケンス

ここでは、UART 通信を使用する各種シーケンスについて説明します。

13.2.1 起動シーケンス

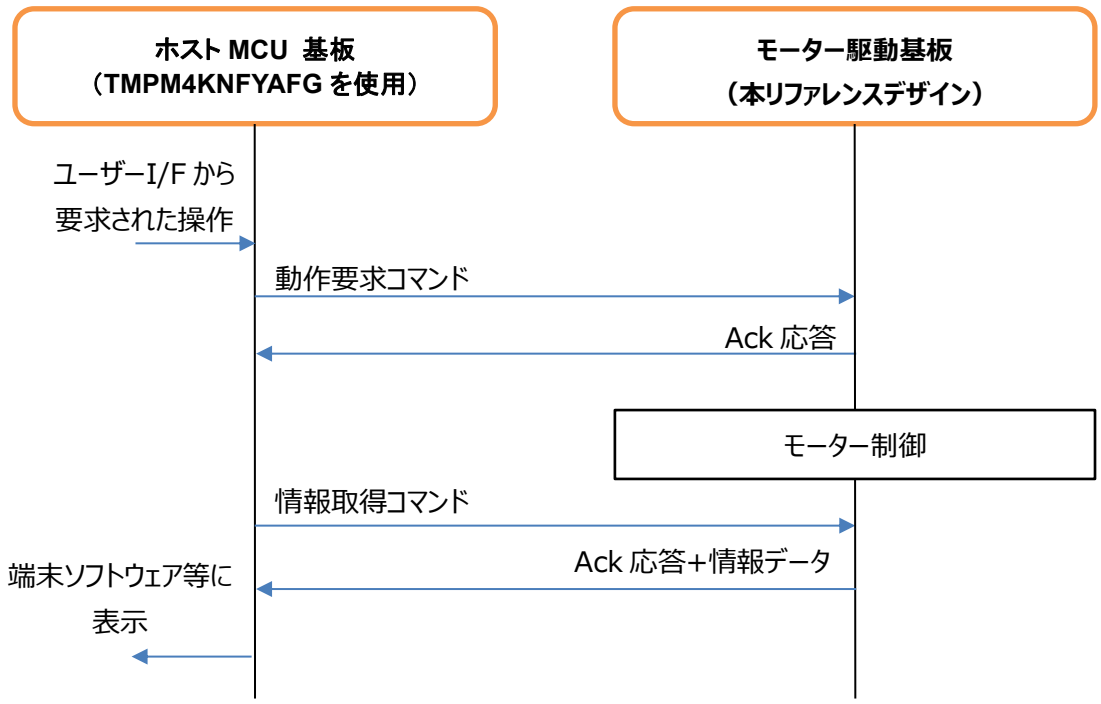
システムの電源投入後に UART 通信を開始するシーケンスです。



© 2024
Toshiba Electronic Devices & Storage Corporation

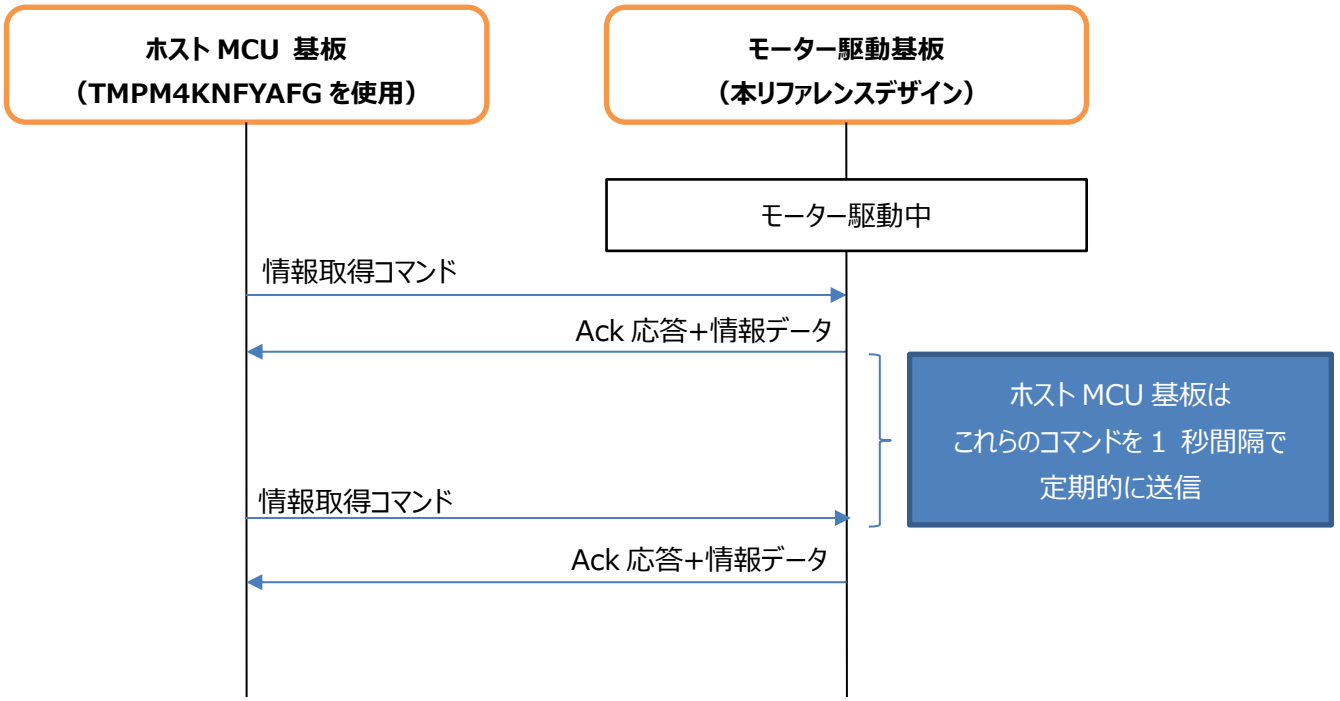
13.2.2 通常シーケンス（ユーザー操作時）

ユーザーによる操作が要求された時のシーケンスです。



13.2.3 通常シーケンス（モーター駆動中）

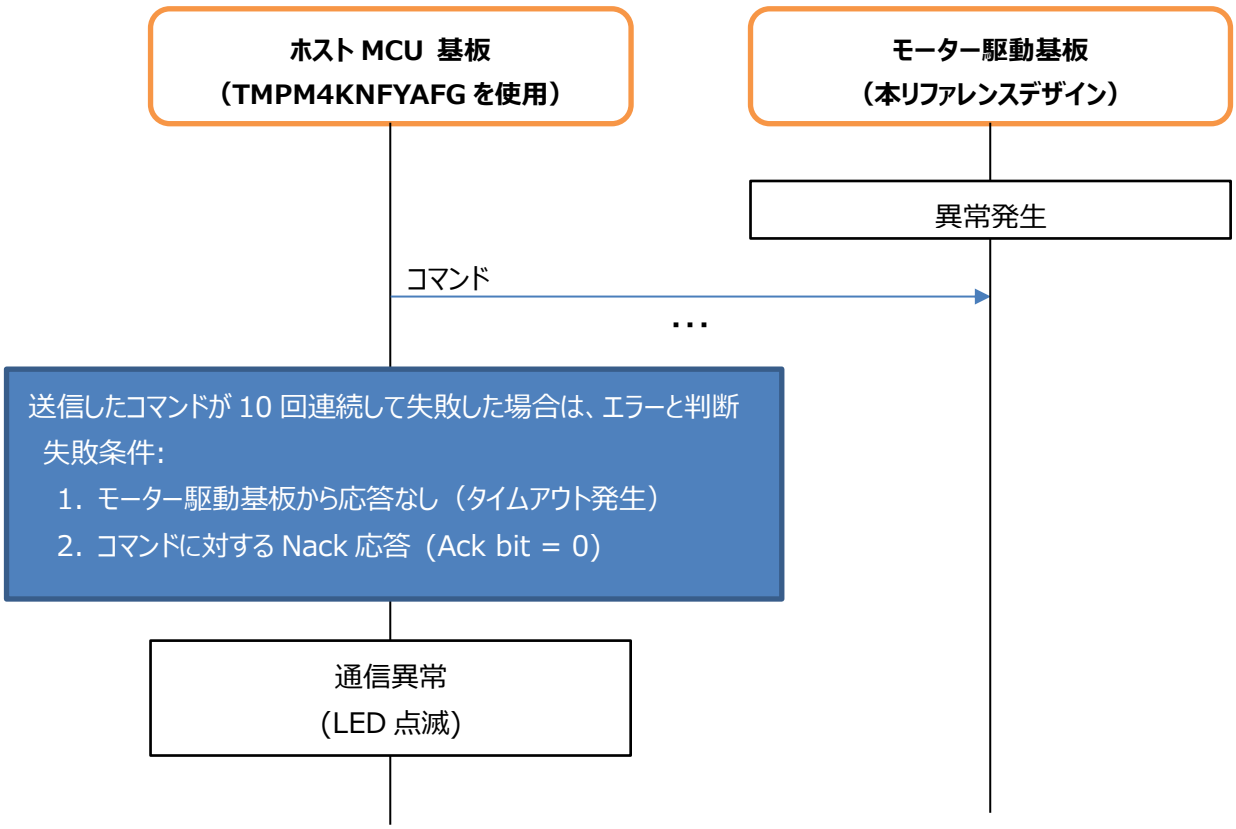
通常のモーター駆動中のシーケンスです。



© 2024
Toshiba Electronic Devices & Storage Corporation

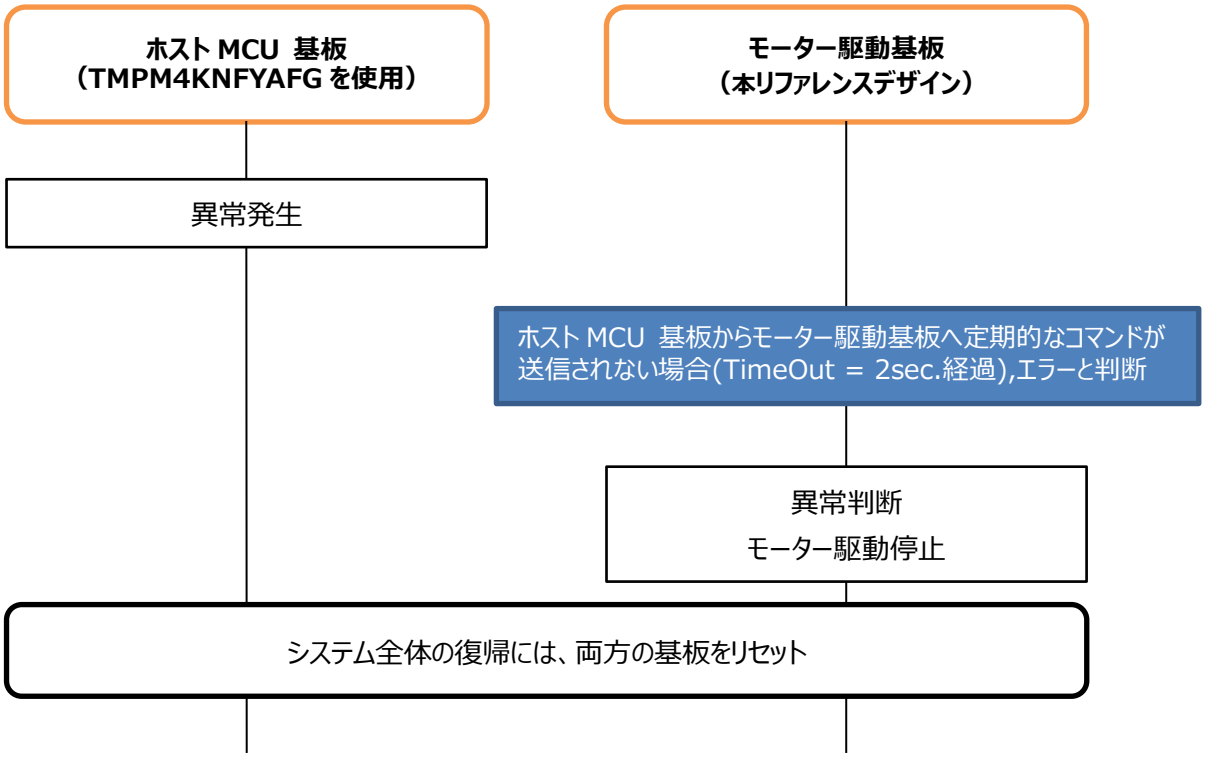
13.2.4 通信異常シーケンス

モーター駆動基板からのデータ受信でエラーが発生した場合のシーケンスです。



13.2.5 異常シーケンス

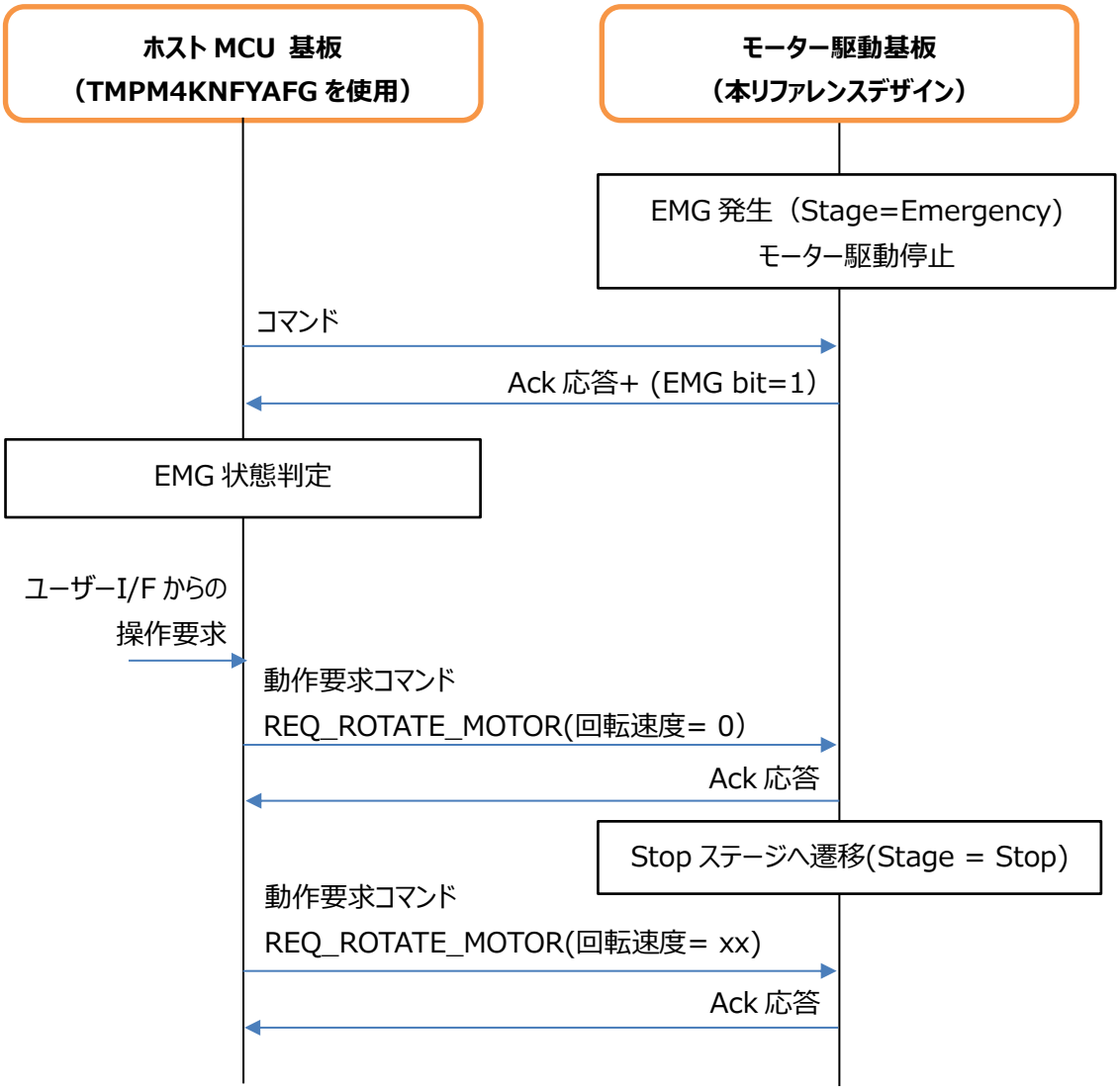
モーター駆動中にホスト MCU 基板で異常が発生した場合のシーケンスです。



© 2024
Toshiba Electronic Devices & Storage Corporation

13.2.6 EMG 状態シーケンス

モーター駆動等での異常状態(EMG)が発生したときのシーケンスです。



© 2024
Toshiba Electronic Devices & Storage Corporation

13.3 UART コマンド仕様

コマンドタイプ	コマンド名	コマンド ID（1 バイト）	データ		説明
			ホスト MCU 基板（4 バイト）	モーター駆動基板（4 バイト）	
動作要求	REQ_SYSTEM_START	0x10	- （ダミーデータ）	- （ダミーデータ）	初期状態から通常状態に遷移させ UART による制御を有効にします。
	REQ_ROTATE_MOTOR	0x11	データ 0~3: 回転速度（整数換算値）	- （ダミーデータ）	回転速度を設定します。 本リファレンスデザインでは、このコマンドは以下のプッシュスイッチ操作で使用されます: 回転速度 10 Hz 増加 回転速度 10 Hz 減少 モータ回転強制停止
	REQ_CHANGE_MOTOR	0x12	データ 1: 0x00-0x01: CH0-CH1 上記以外:予約	- （ダミーデータ）	回転速度を変更するチャンネルを選択します。
	REQ_ON_OFF_PFC	0x13	データ 1: 0x00: PFC オン 0x01: PFC オフ 上記以外: 予約	- （ダミーデータ）	PFC のオン、オフを要求します。
	REQ_ALL_STOP_MOTOR	0x14	データ 1: 0x00: 全モーター停止 上記以外: 予約	- （ダミーデータ）	全モーター停止を要求します。
	REQ_STATUS_CH	0x15	データ 1: 0x00-0x02:CH0-CH2 上記以外: 予約	- （ダミーデータ）	ステータスを取得するチャンネルを選択します。 CH0、CH1、CH2（PFC）
	REQ_STATUS_DAC	0x16	データ 1: 0x00-0x02:CH0-CH2 上記以外: 予約	- （ダミーデータ）	DAC 値を取得するチャンネルを選択します。 CH0、CH1、CH2（PFC）
情報収集	GET_STATE_EMG	0x81	- （ダミーデータ）	データ 0: 0x00: ハードウェア EMG_H 0x01: ソフトウェア EMG_S 0x02: 起動時電流検出異常 0x03: Vdc 異常 データ 1-3: 予約	EMG 状態を取得します。 ハードウェア EMG_H ソフトウェア EMG_S 起動時電流検出異常 Vdc 異常

© 2024
Toshiba Electronic Devices & Storage Corporation

	GET_STAGE	0x82	- (ダミーデータ)	データ 2: 0x00: Stop 0x01: Bootstrap 0x02: Initposition 0x03: Force 0x04: Change_up 0x05: Steady_A 0x06: Emergency 上記以外: 予約	メインステージ状態を取得します。 Stop Bootstrap Initposition Force Change_up Steady_A Emergency
	GET_CONTROL_CH	0x83	- (ダミーデータ)	データ 2: 0x00-0x02:CH0-CH2 上記以外: 予約	現在の制御チャンネルを取得します。 CH0、CH1、CH2（PFC）
	GET_CARRIER_FREQUENCY	0x84	- (ダミーデータ)	データ 0~3: 固定小数点整数変換値	搬送周波数を取得します。 xxxxx (Hz)
	GET_MOTOR_SPEED_MIN	0x85	- (ダミーデータ)	データ 0~3: 固定小数点整数変換値	切換回転速度(強制転流→定常)を取得します。 cHz MIN = xx (Hz)
	GET_MOTOR_SPEED_MAX	0x86	- (ダミーデータ)	データ 0~3: 固定小数点整数変換値	最高回転速度を取得します。 cHz MAX = xx (Hz)
	GET_DEAD_TIME	0x87	- (ダミーデータ)	データ 0~3: デッドタイム（μs） 例: 15.50 μs→1550	デッドタイムを取得します。 x.x (μs)
	GET_GATE_ACTIVE	0x88	- (ダミーデータ)	データ 0: 0x00: H/H 0x01: L/L データ 1-3:予約済み	ゲートアクティブ論理を取得します。 H/H または L/L
	GET_PSITION_DETECT	0x89	- (ダミーデータ)	データ 0: 0x00: 3 シャント 0x01: 1 シャント データ 1-3:予約済み	位置検出方式を取得します。 1 シャント(SPWM On/Off)または 3 シャント（本リファレンスデザインでは 1 シャントのみ使用可能です)
	GET_VDC_VOLTAGE	0x8A	- (ダミーデータ)	データ 0~3: 電圧（V） 例: 15.50 V→1550	Vdc 電圧を取得します。 xx.x (V)
	GET_U_VOLTAGE	0x8C	- (ダミーデータ)	データ 0~3: 電圧（V） 例: 15.50 V→1550	U 相- 0 A 時の電圧を取得します。 x.xx (V)
	GET_V_VOLTAGE	0x8D	- (ダミーデータ)	データ 0~3: 電圧（V） 例: 15.50 V→1550	V - 0 A 時電圧を取得します。 x.xx (V)
	GET_W_VOLTAGE	0x8E	- (ダミーデータ)	データ 0~3: 電圧（V） 例: 15.50 V→1550	W - 0 A 時の電圧を取得します。 x.xx (V)

© 2024
Toshiba Electronic Devices & Storage Corporation

	GET_DAC_DATA	0x8F	- （ダミーデータ）	データ 0:A データ 1:B データ 2:C データ 3:D 0x00:TMPREG0 0x01:TMPREG1 0x02:TMPREG2 0x03:θ.half［1］ 0x04: Id_ref 0x05: ID 0x06: Iq_ref 0x07:Iq 0x08:omega_com.half［1］ 0x09:ω.half［1］ 0x0A:omega_dev	DAC データを取得します。 A:xx B:xx C:xx D:xx または出力なし DAC 電源モード モード 0 DAC データ= A:TMPREG0 B:TMPREG1 C:TMPREG2 D:θ.half［1］ モード 1 DAC データ= A: Id_ref B:Id C: Iq_ref D: Iq モード 2 DAC データ= A: omega_com.half[1］B: オメガハーフ［1］C: omega_dev D: Iq_ref モード 3 DAC データ= A: TMPREG0 B: Iq_ref C:Id_ref D:omega.half[1]
	GET_INTERNAL_AMP	0x90	- （ダミーデータ）	データ 0: 0x00: 外部オペアンプ使用 0x01: MCU 内蔵オペアンプ使用 データ 1-3: 予約	MCU 内蔵オペアンプ使用状態を取得します。
	GET_DIRECTION	0x91	- （ダミーデータ）	データ 0: 0x00: CW 0x01: CCW データ 1-3: 予約	回転方向を取得します。 CW(時計回り)または CCW(反時計回り)（本デザインでは CW のみが使用可能です）
	GET_MODULATION	0x92	- （ダミーデータ）	データ 0: 0x00: 3 相変調 0x01: 2 相変調 データ 1-3: 予約	PWM 変調方式を取得します。
	GET_MOTOR_SPEED	0x94	- （ダミーデータ）	データ 0: 0x00-0xFF: 回転速度 データ 1-3: 予約	回転速度を取得します。 xx （Hz）
	REQ_CHANGE_MOTOR	0x12	データ 1: 0x00-0x01: CH0-CH1 上記以外:予約	- （ダミーデータ）	回転速度を変更するチャンネルを選択します。
	REQ_ON_OFF_PFC	0x13	データ 1: 0x00: PFC オン 0x01: PFC オフ 上記以外: 予約	- （ダミーデータ）	PFC のオン、オフを要求します。

© 2024
Toshiba Electronic Devices & Storage Corporation

	REQ_ALL_STOP_MOTOR	0x14	データ 1: 0x00: 全モーター停止 上記以外: 予約	- (ダミーデータ)	全モーター停止を要求します。
	REQ_STATUS_CH	0x15	データ 1: 0x00-0x02:CH0-CH2 上記以外: 予約	- (ダミーデータ)	ステータスを取得するチャンネルを選択します。 CH0、CH1、CH2（PFC）
	REQ_STATUS_DAC	0x16	データ 1: 0x00-0x02:CH0-CH2 上記以外: 予約	- (ダミーデータ)	DAC 値を取得するチャンネルを選択します。 CH0、CH1、CH2（PFC）
	GET_OUTROOM_TEMP	0x8B	- (ダミーデータ)	データ 0~3: 温度（℃） 例: 15.50 ℃ → 1550	サーミスタ温度(Outroom)を取得します。 xx.x ℃または N.A
	GET_OUTPIPE_TEMP	0x95	- (ダミーデータ)	データ 0~3: 温度（℃） 例: 15.50 ℃ → 1550	サーミスタ温度(Outpipe)を取得します。 xx.x ℃または N.A
	GET_EXHAUST_TEMP	0x96	- (ダミーデータ)	データ 0~3: 温度（℃） 例: 15.50 ℃ → 1550	サーミスタ温度(Exhaust)を取得します。 xx.x ℃または N.A
	GET_DIODE_TEMP	0x97	- (ダミーデータ)	データ 0~3: 温度（℃） 例: 15.50 ℃ → 1550	サーミスタ温度(Diode)を取得します。 xx.x ℃または N.A
	GET_IGBT_TEMP	0x98	- (ダミーデータ)	データ 0~3: 温度（℃） 例: 15.50 ℃ → 1550	サーミスタ温度(IGBT)を取得します。 xx.x ℃または N.A
	GET_HVMOS_TEMP	0x99	- (ダミーデータ)	データ 0~3: 温度（℃） 例: 15.50 ℃ → 1550	サーミスタ温度(HVMOS)を取得します。 xx.x ℃または N.A

© 2024
Toshiba Electronic Devices & Storage Corporation

14. 付録

14.1 固定小数点処理

本サンプルソフトには、小数演算は固定小数点で行っていますので、固定小数点演算について概要的に説明します。

14.2 正規化(Normalize)

正規化とはデータを一定のルールに従って変形しデータを利用しやすくすることです。

本アプリケーションノートでは最上位を符号ビット(0 は正、1 は負)とし、次のビットとの間に小数点を置き、またそのデータがなりうる最大の値(0x7fff)を 1、最小の値(0x8000)を-1 となるように正規化を行います。

例）

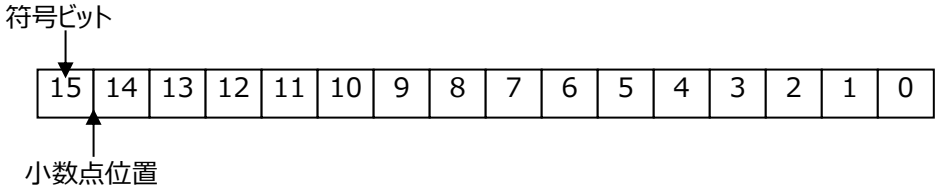


図 21 16 ビットデータの例

本アプリケーションノートでは電流データの最大の値を cA_Max と定義しており、例えば 16 ビット電流データが 0x7fff の場合は A_Max(A)、0x8000 の場合は-A_Max(A)となります。

14.3 データフォーマット

本アプリケーションのモーター制御部では固定小数点演算を行っています。

固定小数点演算では小数部分のビット数を Q 表記(Q フォーマット)で表します。

基本的には 16 ビットデータの場合は Q15 フォーマット(小数部分 15 ビット)、32 ビットデータの場合は Q31 フォーマット(小数部分 31 ビット)で演算を行っています。

小数フォーマットで表すことのできる値はフォーマットにより異なります。

表 5 単精度（16 ビット）小数フォーマット

Q フォーマット	小数ビット数	正の最大値(0x7FFF)	負の最大値（0x8000）
Q15	15	0.999969482421875	-1
Q14	14	1.999938964843750	-2
Q13	13	3.999877929687500	-4
Q12	12	7.999755859375000	-8
Q11	11	15.999511718750000	-16
Q10	10	31.999023437500000	-32
Q9	9	63.998046875000000	-64
Q8	8	127.996093750000000	-128
Q7	7	255.992187500000000	-256
Q6	6	511.984375000000000	-512
Q5	5	1023.968750000000000	-1024
Q4	4	2047.937500000000000	-2048
Q3	3	4095.875000000000000	-4096
Q2	2	8191.750000000000000	-8192
Q1	1	16383.500000000000000	-16384
Q0	0	32767.000000000000000	-32768

© 2024
Toshiba Electronic Devices & Storage Corporation

表 6 倍精度（32 ビット）小数フォーマット

Q フォーマット	小数ビット数	正の最大値(0x7FFFFFFF)	負の最大値（0x80000000）
Q31	31	0.999999999534338	-1
Q30	30	1.999999999068670	-2
Q29	29	3.999999998137350	-4
Q28	28	7.999999996274700	-8
Q27	27	15.999999992549400	-16
Q26	26	31.999999985098800	-32
Q25	25	63.999999970197600	-64
Q24	24	127.999999940395000	-128
Q23	23	255.999999880790000	-256
Q22	22	511.999999761581000	-512
Q21	21	1023.999999523160000	-1024
Q20	20	2047.999999046320000	-2048
Q19	19	4095.999998092650000	-4096
Q18	18	8191.999996185300000	-8192
Q17	17	16383.999992370600000	-16384
Q16	16	32767.999984741200000	-32768
Q15	15	65535.999969482400000	-65536
Q14	14	131071.999938965000000	-131072
Q13	13	262143.999877930000000	-262144
Q12	12	524287.999755859000000	-524288
Q11	11	1048575.999511720000000	-1048576
Q10	10	2097151.999023440000000	-2097152
Q9	9	4194303.998046870000000	-4194304
Q8	8	8388607.996093750000000	-8388608
Q7	7	16777215.992187500000000	-16777216
Q6	6	33554431.984375000000000	-33554432
Q5	5	67108863.968750000000000	-67108864
Q4	4	134217727.937500000000000	-134217728
Q3	3	268435455.875000000000000	-268435456
Q2	2	536870911.750000000000000	-536870912
Q1	1	1073741823.500000000000000	-1073741824
Q0	0	2147483647.000000000000000	-2147483648

14.4 固定小数点での演算

固定小数点演算の四則演算では、加算、減算はそのまま整数同士のように演算できますが、乗算、除算では演算結果の小数点位置が変化するため、元の小数点位置に戻す処理が必要になります。

(1)乗算

小数フォーマット同士の乗算の場合、例えば Q15 フォーマットデータを乗算する場合、結果は倍精度 Q30 フォーマットになります。Q31 フォーマットデータが必要なときは、演算結果を 1 ビット左シフトすることで倍精度 Q31 フォーマットになります。

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

(2)除算

© 2024
Toshiba Electronic Devices & Storage Corporation

小数フォーマット同士の除算の場合、例えば Q31 フォーマットを Q15 フォーマットで除算する場合、結果は Q16 フォーマットになります。Q15 フォーマットデータが必要なときは、演算前に除数を 1 ビット右シフトすることで、Q15 フォーマットデータを得ることができます。

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

© 2024
Toshiba Electronic Devices & Storage Corporation

ご利用規約

本規約は、お客様と東芝デバイス＆ストレージ株式会社（以下「当社」といいます）との間で、当社半導体製品を搭載した機器を設計する際に参考となるドキュメント及びデータ（以下「本リファレンスデザイン」といいます）の使用に関する条件を定めるものです。お客様は本規約を遵守しなければなりません。

第１条 禁止事項

お客様の禁止事項は、以下の通りです。

1．本リファレンスデザインは、機器設計の参考データとして使用されることを意図しています。信頼性検証など、それ以外の目的には使用しないでください。

2．本リファレンスデザインを販売、譲渡、貸与等しないでください。

3．本リファレンスデザインは、高低温・多湿・強電磁界などの対環境評価には使用できません。

4．本リファレンスデザインを、国内外の法令、規則及び命令により、製造、使用、販売を禁止されている製品に使用しないでください。

第２条 保証制限等

1．本リファレンスデザインは、技術の進歩などにより予告なしに変更されることがあります。

2．本リファレンスデザインは参考用のデータです。当社は、データ及び情報の正確性、完全性に関して一切の保証をいたしません。

3．半導体素子は誤作動したり故障したりすることがあります。本リファレンスデザインを参考に機器設計を行う場合は、誤作動や故障により生命・身体・財産が侵害されることのないように、お客様の責任において、お客様のハードウェア・ソフトウェア・システムに必要な安全設計を行うことをお願いします。また、使用されている半導体素子に関する最新の情報（半導体信頼性ハンドブック、仕様書、データシート、アプリケーションノートなど）をご確認の上、これに従ってください。

4．本リファレンスデザインを参考に機器設計を行う場合は、システム全体で十分に評価し、お客様の責任において適用可否を判断して下さい。当社は、適用可否に対する責任は負いません。

5．本リファレンスデザインは、その使用に際して当社及び第三者の知的財産権その他の権利に対する保証又は実施権の許諾を行うものではありません。

6．当社は、本リファレンスデザインに関して、明示的にも黙示的にも一切の保証（機能動作の保証、商品性の保証、特定目的への合致の保証、情報の正確性の保証、第三者の権利の非侵害保証を含むがこれに限らない。）をせず、また当社は、本リファレンスデザインに関する一切の損害（間接損害、結果的損害、特別損害、付随的損害、逸失利益、機会損失、休業損害、データ喪失等を含むがこれに限らない。）につき一切の責任を負いません。

第３条 契約期間

本リファレンスデザインをダウンロード又は使用することをもって、お客様は本規約に同意したものとみなされます。本規約は予告なしに変更される場合があります。当社は、理由の如何を問わずいつでも本規約を解除することができます。本規約が解除された場合は、お客様は本リファレンスデザインを破棄しなければなりません。さらに当社が要求した場合には、お客様は破棄したことを証する書面を当社に提出しなければなりません。

第４条 輸出管理

お客様は本リファレンスデザインを、大量破壊兵器の開発等の目的、軍事利用の目的、あるいはその他軍事用途の目的で使用してはなりません。また、お客様は「外国為替及び外国貿易法」、「米国輸出管理規則」等、適用ある輸出関連法令を遵守しなければなりません。

第５条 準拠法

本規約の準拠法は日本法とします。

第６条 管轄裁判所

本リファレンスデザインに関する全ての紛争については、別段の定めがない限り東京地方裁判所を第一審の専属管轄裁判所とします。

© 2024
Toshiba Electronic Devices & Storage Corporation