TMPM4KN
モーターサンプルソフトウェア
アプリケーションノート
Rev. 1.0

目次

1. 概要

1.1 はじめに

本モーター制御用サンプルソフトはイーエスピー企画製 SBK-M4KN(TMPM4KN マイコン評価ボード)と KES-P2(インバーターボード)を使用して動作確認を行ております。本評価ボードの詳細については(株)イーエスピー企画(http://esp.co.jp/)にお問い合わせください。

1.2 スペック

- ◆ 応用製品 ・・・・・・ ブラシレス DC モーター制御用ソフト(ベクトルエンジン"VE"使用)
- ◆ マイコン ・・・・・・ TMPM4KNFYAFG
- ◆ クロック ・・・・・・ 80MHz

- ◆ 開発環境

・IAR Embedded Workbench for ARM 8.50.6
・Arm® KEIL® MDK 5.31.0.0
・評価ボード：(株)イーエスピー企画
　SBK-M4KN + KES-P2
・モーター：ツカサ電工(株) – TG611B
・動作電圧 24V

- ◆ 概要

・ブラシレス DC モーターベクトル制御
・速度制御
・評価用
　-DAC 出力(変数値アナログ出力用)
　-インバーターボード温度検出
　-UART 出力(Tera Term などを使用した初期ステータス確認)
　-LED 出力(モニター用 LED)

- ◆ モーター制御内容

項目	制御内容
回転制御方式	一定回転数制御(ボリューム抵抗操作)
回転方向	CW/CCW(スイッチ切替)
変調方式	3 相変調 or 2 相変調
電流検出方式	3 シャント or 1 シャント
位置検出方式	センサーレス
オペアンプ	マイコン内蔵 or 外部

◆ 注意事項

EMG が発生した場合、リセットによる解除を行ってください。

1 シャントで動作させる場合、下記の設定となっているか確認してください。

・D_Para_CHx.h

```
#define cSHUNT_TYPE_CHx (1)
#define cBOOT_TYPE_CHx (cBoot_v)
```

・B_User.c

```
Motor_chx.usr.com_user.spwm = 1;
```

※"x"には使用モーターのチャネルに合わせ、0~2 を当てはめてください。

1.3 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。

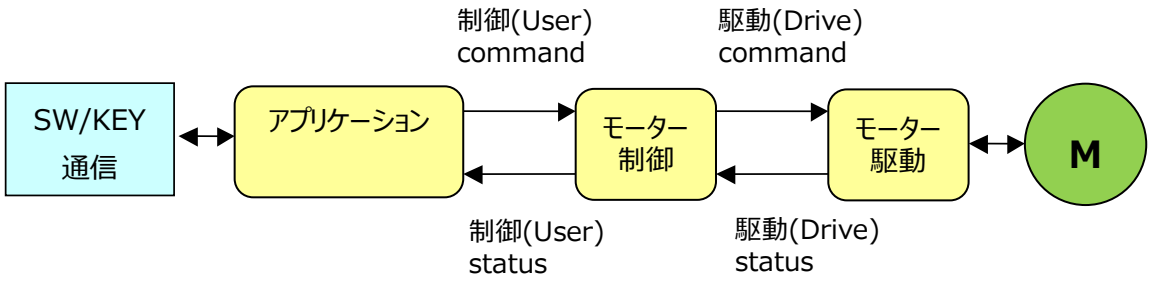


図 1.1　ベクトル制御ソフトウェアの構造

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。

iii. 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。

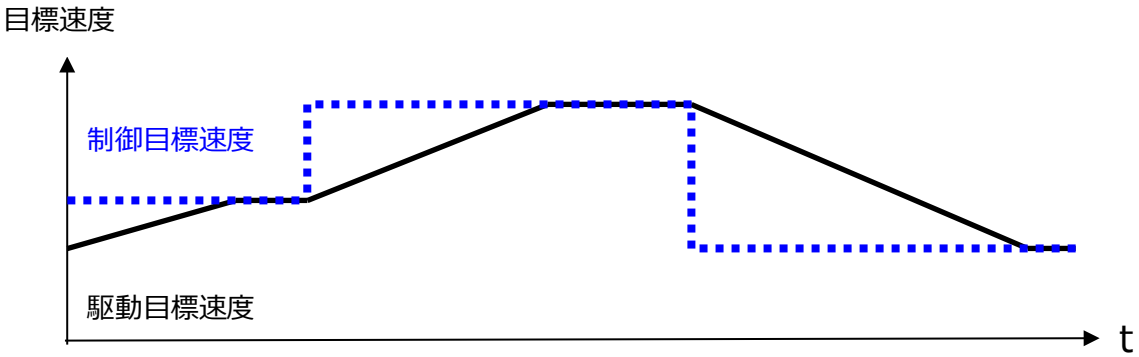


図 1.2　制御目標速度と駆動目標速度

2. システムブロック図

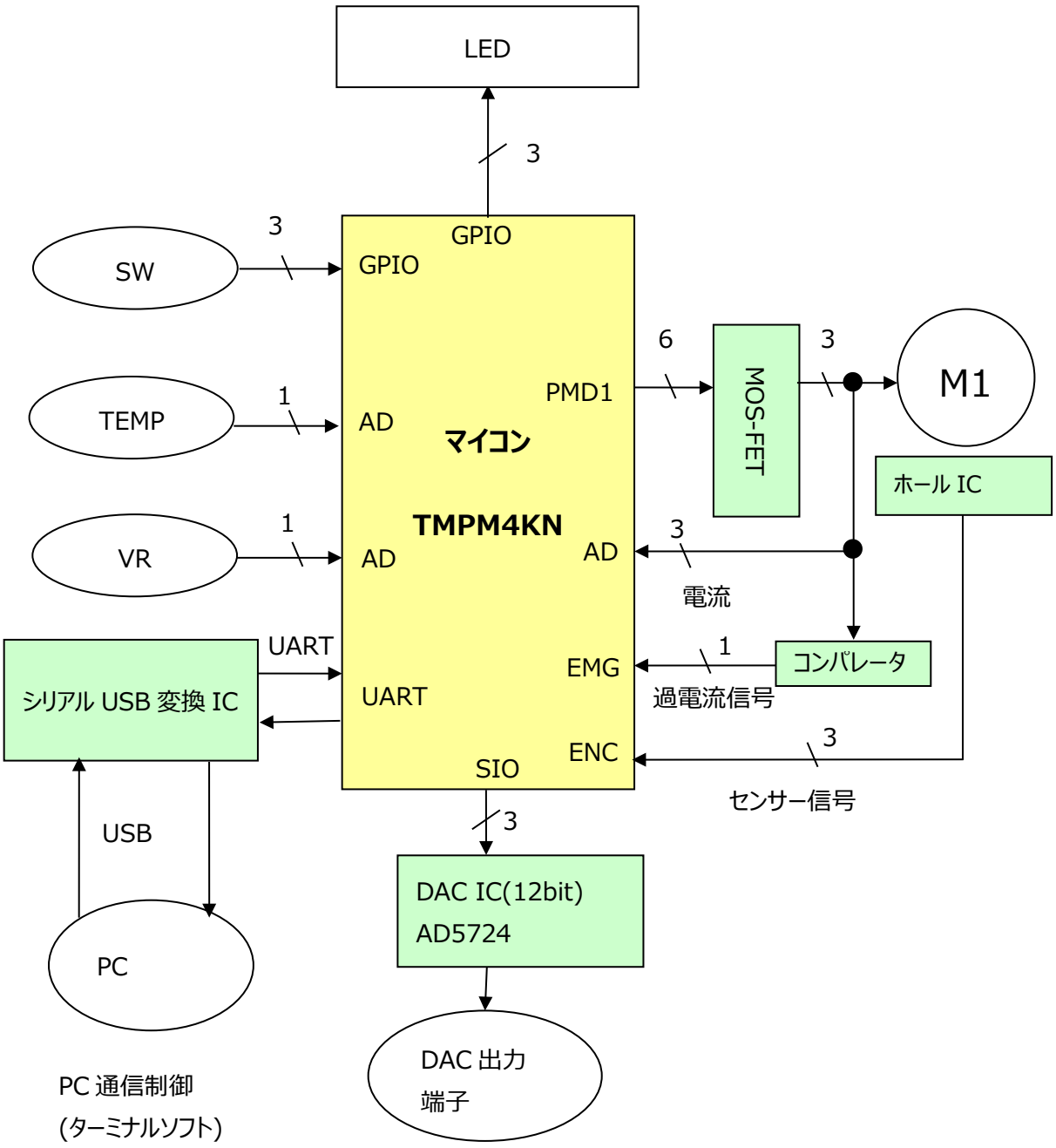


図 2.1　システムブロック図

3. 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
|   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
|   E_Sub.c                 演算用関数ソースファイル
|   E_Sub.h                 演算用関数ヘッダーファイル
|   H_Hall.c                ホールセンサー用関数ソースファイル
|   H_Hall.h                ホールセンサー用関数ヘッダーファイル
|   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
|   ipdefine.h              マイコン設定関連ヘッダーファイル
|   ipdrv.c                 マイコンハード参照用ソースファイル
|   ipdrv.h                 マイコンハード参照用ヘッダーファイル
|   main.c                  メインルーチン
|   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
|   v7z_board.c             V7Z ボード動作用ソースファイル
|   v7z_board.h             V7Z ボード動作用ヘッダーファイル
|
└──driver                   デバイスドライバー関連のファイルが格納されています。
|   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_ch0.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_ch1.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
|   D_Para_ch2.h            ベクトル制御パラメーター(チャネル 2 用)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
|   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
|   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
|
└──Libraries                CMSIS コア、各 IP 用ライブラリーが格納されています。
    └──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
    |   |   TMPM4KNA.h          マイコンレジスター定義ヘッダーファイル
    |   |   TMPM4KyA.h          TMPM4KyA グループ定義ヘッダーファイル
```

```
│   └─startup
│       ├─arm                                   スタートアップアセンブラーソース(arm 用)
│       │       startup_TMPM4KHA.s
│       │       startup_TMPM4KLA.s
│       │       startup_TMPM4KMA.s
│       │       startup_TMPM4KNA.s
│       └─iar                                   スタートアップアセンブラーソース(IAR 用)
│               startup_TMPM4KHA.s
│               startup_TMPM4KLA.s
│               startup_TMPM4KMA.s
│               startup_TMPM4KNA.s
│
└─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
    ipdrv_adc.h
    ipdrv_cg.c
    ipdrv_cg.h
    ipdrv_common.h
    ipdrv_enc.c
    ipdrv_enc.h
    ipdrv_siwdt.c
    ipdrv_siwdt.h
    ipdrv_t32a.c
    ipdrv_t32a.h
    ipdrv_tspi.c
    ipdrv_tspi.h
```

3.1 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
dac.datsft0 - 3	ビットデータを左シフトする量(x)を設定 計算式(x)：data × (2^x)

3.2 ユーザーインターフェースについて

ユーザーインターフェースとして下記機能を用意しています。

«ユーザーインターフェース内容»

1. 回転方向切替
   SW(S_SW1)を切り替えることでモーターの回転方向を切り替えることができます。
   Hi：CW　Low：CCW

2. 位相切替
   SW(S_SW2)を切り替えることでモーターの回転方向を切り替えることができます。
   Hi：2 相変調　Low：3 相変調

3. 回転数制御
   VR1 を操作することにより、モーターの回転速度を指定することができます。
   速度範囲：12rps～200rps(電気角)

4. DAC モード切替
   SW(USW1)を押下することで、下記表に従い DAC モードを切り替えることができます。

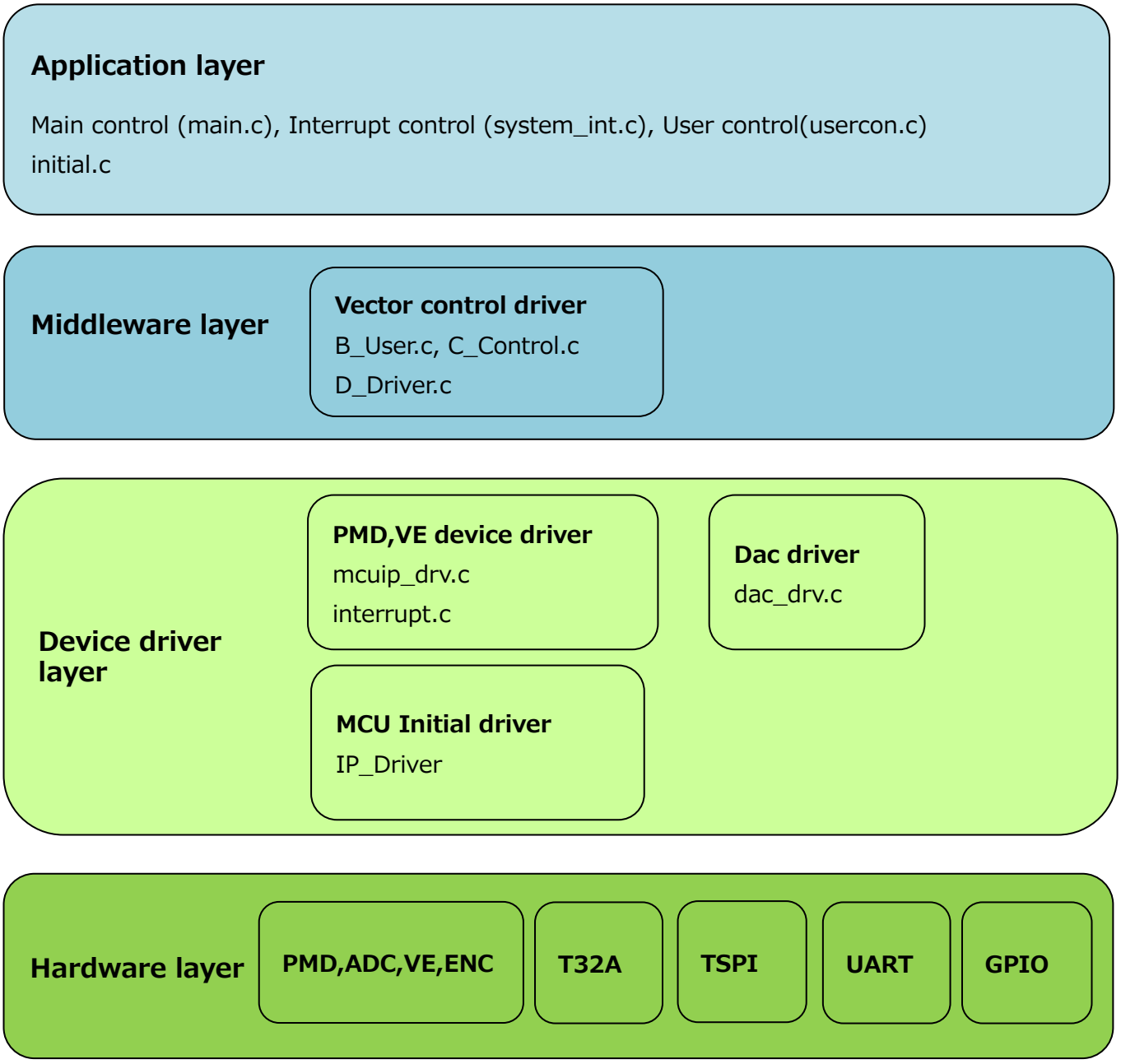
No.	現在のモード	SW 押下後
1	モード 0	モード 1
2	モード 1	モード 2
3	モード 2	モード 3
4	モード 3	モード 0

5. LED 表示
   LED4～LED6 でモーター状態を確認することができます。
   1. 回転ステータス LED(LED4)
      回転停止：消灯
      回転中(強制転流)：点灯
      回転中(定常)：消灯
   2. VE 割り込み表示 LED(LED5)
      VE 割り込み処理中：点灯
      それ以外：消灯
   3. EMG ステータス LED(LED6)
      EMG 未検出：消灯
      EMG 検出：点灯

6. 初期ステータス表示
   リセット解除後に初期ステータスを確認することができます。

詳細は 14. ステータス確認モニターを参照ください。

4. モジュール構成



5. マイコンハードウェア割り付け

5.1 IP

IP		制御内容	備考
TMRB	チャネル 0	4kHz 周期割り込み	
	チャネル 1	未使用	
	チャネル 2	未使用	
	チャネル 3	未使用	
	チャネル 4	未使用	
	チャネル 5	未使用	
TSPI	チャネル 0	未使用	
	チャネル 1	DAC IC 制御	SIO モード
UART	チャネル 0	PC 接続	UART0
	チャネル 1	未使用	
	チャネル 2	未使用	
	チャネル 3	未使用	
ADC	ユニット A	モーター電流、温度取得	
	ユニット B	未使用	
	ユニット C	回転数制御 VR	
PMD	チャネル 0	モーターCH0 制御	
	チャネル 1	モーターCH1 制御	
	チャネル 2	モーターCH2 制御	
VE	チャネル 0	モーターCH0 制御	
EMC	チャネル 0	モーターCH0 エンコーダー信号制御	
	チャネル 1	モーターCH1 エンコーダー信号制御	
	チャネル 2	モーターCH2 エンコーダー信号制御	

5.2 割り込み

要因名	処理内容	関数名
INTTB00IRQn	4kHz 周期タイミング作成	INTTB00_IRQHandler
INTVCN0_IRQn	ベクトル制御ソフト処理 for CH0	INTVCN0_IRQHandler
INTADAPDA_IRQn	ベクトル制御中断 for CH0	INTADAPDA_IRQHandler
INTSC0TX_IRQn	UART 送信完了割り込み処理	INTSC0TX_IRQHandler
INTSC1TX_IRQn	DAC IC 制御	INTSC1TX_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        3       /* VE interrupt */
#define INT_ADC_LEVEL       3       /* ADC interrupt */
#define INT_DAC_LEVEL       6       /* SIO interrupt for Dac */
```

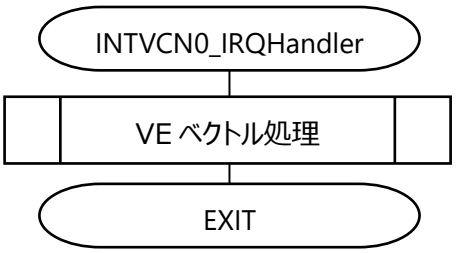
6. ジェネラルフロー

6.1 メインルーチン(main)

6.2 メインループ(main_loop)

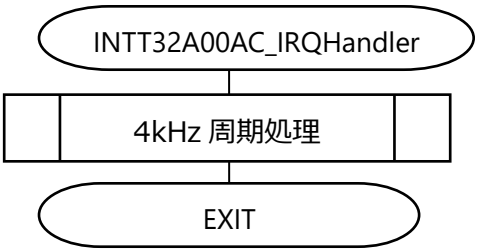
6.3 割り込み

VE スケジュール完了時



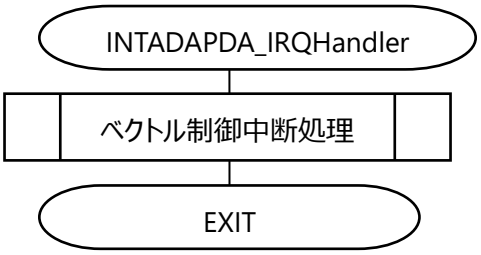
INT_VectorContorlByVE_CH0 ()

4kHz 周期ごと



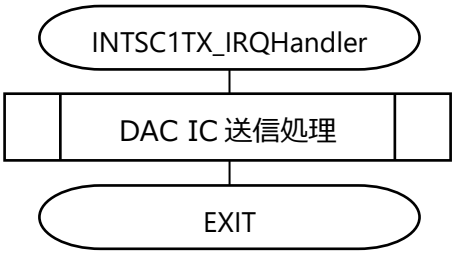
INT_interval_4kHz ()

AD 割り込み時



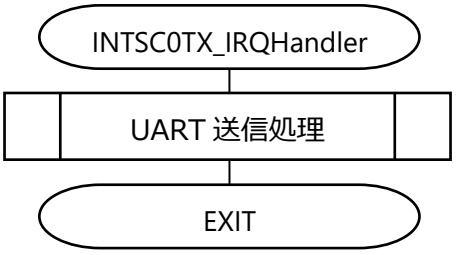
INT_VectorControlBySoft_CH0 ()

DAC 通信 8bit 送信完了時



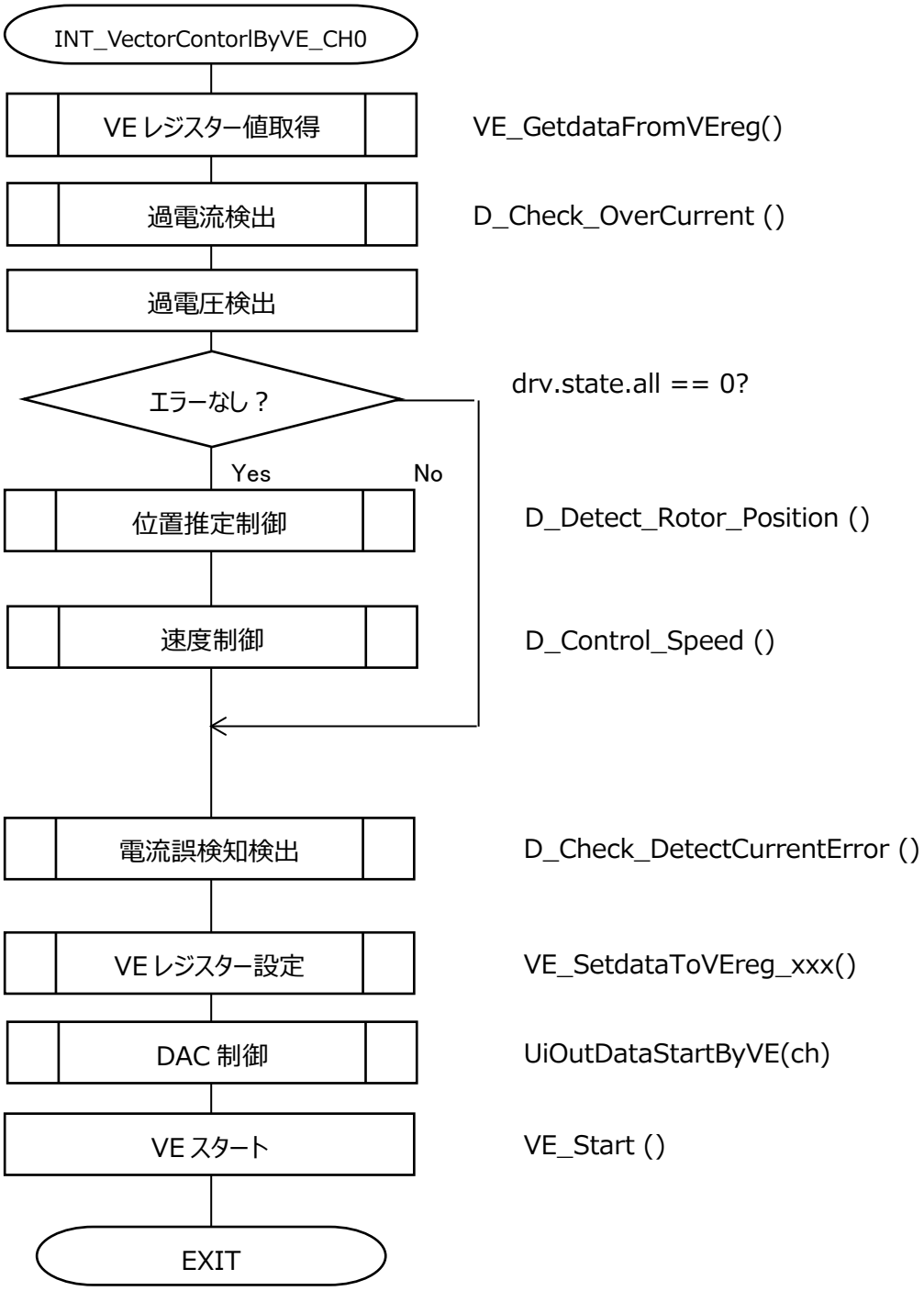
DacWriteContinue ()

UART8bit 送信完了時

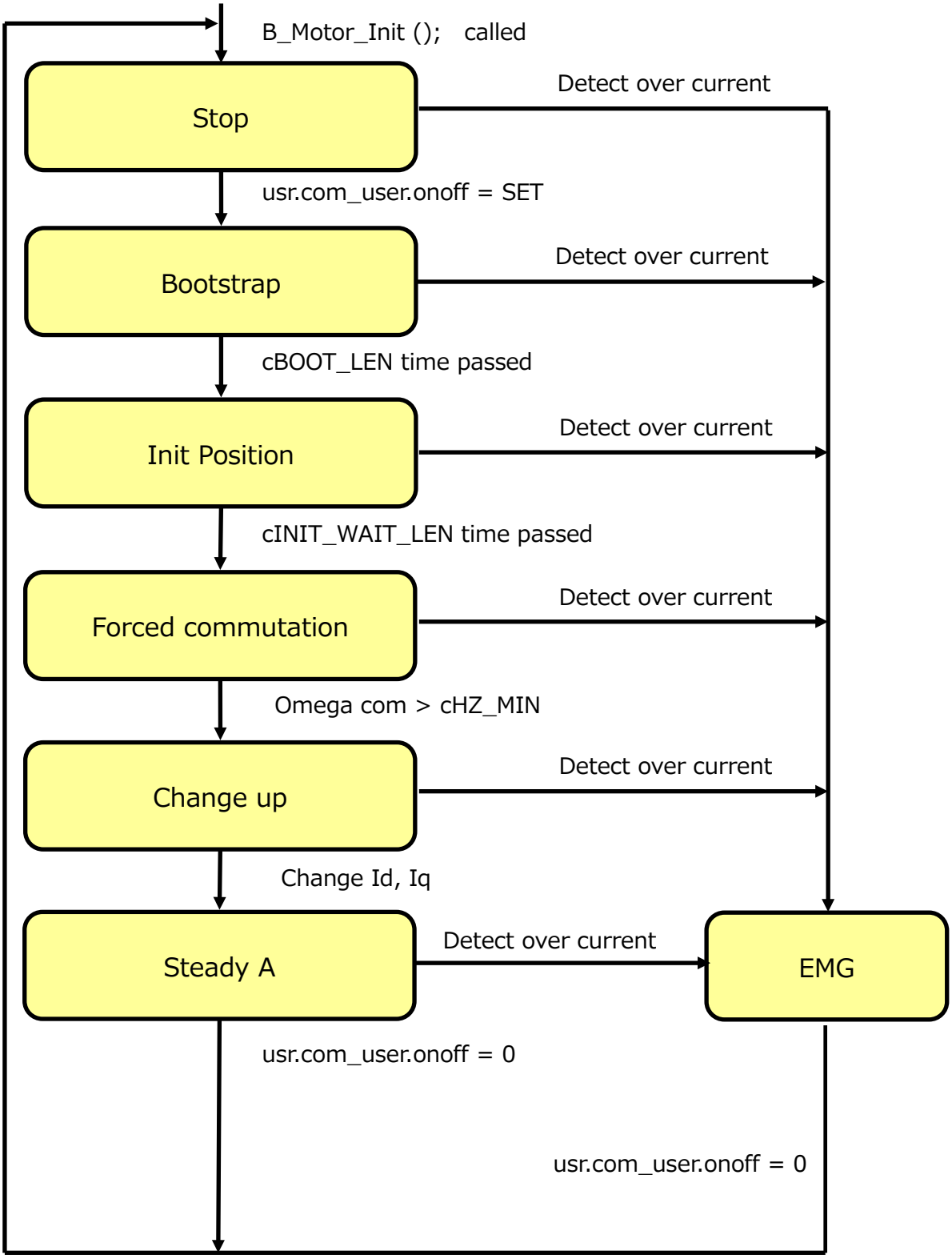


INT_uart0tx ()

6.3.1 VE ベクトル処理(INT_VectorControl_byVE)



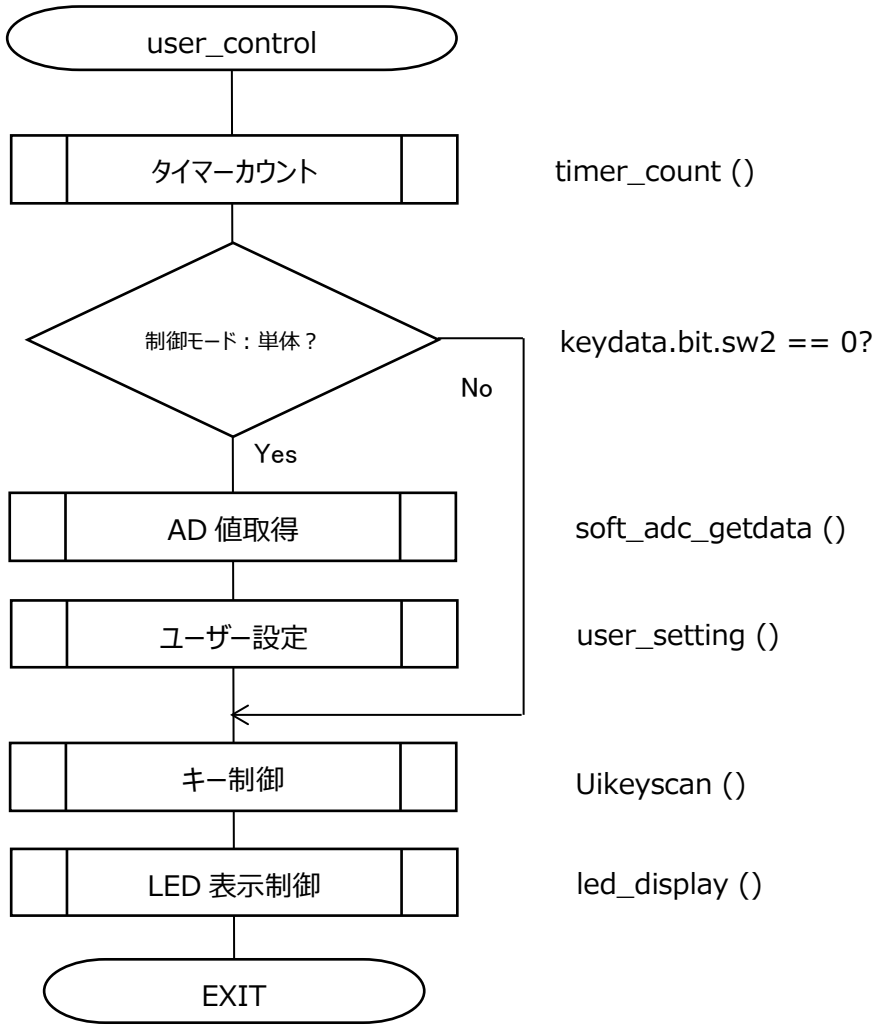
7. モーター動作の状態遷移(ステージ)



8. ユーザーアプリ

8.1 ユーザー制御

メイン周期(1ms)ごとにユーザーアプリケーション部の各処理を行います。



8.1.1 AD 値取得(soft_adc_getdata)

VR1、インバーター温度の AD 値(12bit)を単独変換設定で取得します。
10 回取得後、最大・最小値を除く 8 回の平均値をそれぞれの有効値として VR1 は ad_vr.avedat、インバーター温度は ad_inv.avedat に格納されます。

8.1.2 キー制御(Uikeyscan)

S_SW1、S_SW2、USW1 のキースキャン処理を行います。
各キーデータは 20 回連続一致で確定し、S_SW1、S_SW2 は keydata、USW1 は swdata に格納されます。

8.1.3 ユーザー設定(user_interface)

ユーザー操作により入力された VR1 に従い、下記設定を行います。

1. モーター速度制御
   ad_vr.avedat を 8bit 分解能にし、この値が
   0x10 未満：0(0Hz)
   0xF0 以上：200(最大速度)
   0x10～0xEF：有効値として速度計算
2. モーター回転方向切替
   keydata.sw1 の値に従い、回転方向 rote_dir の設定を切り替える。
   keydata.sw1 が 1 の場合、rote_dir：1(正転)
   keydata.sw1 が 0 の場合、rote_dir：0(逆転)
3. 変調方式切替
   keydata.sw2 の値に従い、変調方式 Motor_chx.usr.co_user.modul の設定を切り替える(x=0,1)
   keydata.sw2 が 1 の場合、Motor_chx.usr.com_user.modul：1(2 相変調)
   keydata.sw2 が 0 の場合、Motor_chx.usr.com_user.modul：0(3 相変調)
4. DAC モード
   swdata.sw が 1 となるたびに dac.select の設定を切り替える。

8.1.4 LED 表示制御(led_display)

回転ステータスに従い、LED4 の点灯制御を行います。
詳細は 3.2 ユーザーインターフェースについてを参照ください。

8.1.5 UART 送信データ設定(send_data_set)

モーター初期設定や現在の回転速度のモニター用データ設定を行います。
通信設定は
115200bps、データ 8bit、ストップビット 1bit、パリティーなし、フロー制御なし
となります。

9. 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。

9.1 制御コマンド

制御コマンドを以下に記します。

9.1.1 制御方法(usr.com_user)

・モーターの起動スタート、ストップ
・エンコーダーの有無
・変調方式（２相変調、３相変調）
・シフト PWM オン、オフ(VE 使用時かつ 1 シャント、2 相変調のときだけ有効)

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

```
command_t       com_user;
```

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。

9.1.2 制御目標速度(usr.omega_user)

```
q31_u      omega_user;      /* [Hz/maxHz] OMEGA 指令，Q31 */
```

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。

9.1.3 始動電流(usr.Id_st_user, usr.Iq_st_user)

```
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 が始動電流指令として設定されます。

9.2 駆動コマンド

駆動コマンドを以下に記します。

9.2.1 駆動方法(drv.command)

command_t command;

モーター制御は、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。

9.2.2 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 */
  uint16_t F_vcomm_Edetect: 1; /* Position detect 0=off 1=on */
  uint16_t F_vcomm_Idetect: 1; /* Current detect 0=off 1=on */
  uint16_t F_vcomm_onoff: 1;    /* Motor output 0=off 1=on */
} vectorcmd_t;
```

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。

VE を使用したベクトル制御では、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	-	-	-	-	-	-	-
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。

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_volt

電流制御で、d、q 軸電圧の基準値の算出方法を指令します。

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 制御を実行しないで、指令値をそのまま基準値とします。

5) 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。

6) 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。

9.3 駆動状態

9.3.1 エラー状態(drv.state)

```
typedef union {
  struct {
        uint16_t reserve:12;        /* reserve */
        uint16_t Loss_sync: 1;      /* 0:normal, 1: Loss of synchronism  */
        uint16_t emg_DC:1;          /* 0:normal, 1: Over Vdc             */
        uint16_t emg_I:1;           /* 0:normal, 1: Current detect error */
        uint16_t emg_S:1;           /* 0:normal, 1: Over current(soft)   */
        uint16_t emg_H:1;           /* 0:normal, 1: Over current(hard)   */
  } flg;
  uint16_t all;
} state_t;
```

①	Loss_sync	脱調検出	脱調を検出したとき SET されます(未実装)
②	emg_DC	異常電圧検出	異常電圧を検出したとき SET されます。
③	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。
④	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。
⑤	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。

9.4 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_ch0;    /* Motor data for ch0 */
vector_t    Motor_ch1;    /* Motor data for ch1 */
vector_t    Motor_ch2;    /* Motor data for ch2 */
```

9.4.1 変数一覧

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
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	
q31_u	drv.Vdc_ave	(未使用)	---	
q31_u	drv.Vd	d 軸電圧	Q31	
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	---	(9.2.1 参照)
vectorcmd_t	drv.vector_cmd	ベクトル制御コマンド	---	(9.2.2 参照)
q15_t	drv.spwm_threshold	シフト切り替え速度	Q15	1 シャント時だけ
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	
q15_t	drv.omega_enc	エンコーダー速度	Q15	
q15_t	drv.omega_enc_raw	エンコーダー速度	Q15	
q31_u	drv.omega_enc_ave	エンコーダー速度平均	Q31	

uint32_t	drv.theta_enc	エンコーダー角度	Q0	
int16_t	drv.EnCnt	エンコーダーカウンター値	Q0	
int16_t	drv.EnCnt1	エンコーダーカウンター前回値	Q0	
int16_t	drv.EnCnt_dev	エンコーダーカウンター偏差	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	---	(9.1.1 参照)
command_t	usr.com_user_1	制御方法前回	---	
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 切り替え速度	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	チェンジアップ後待ち時間	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	
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	
uint32_t	para.enc.pls2theta	パルス数から角度への演算係数	Q32	
q15_t	para.enc.pls2omega	パルス数から速度への演算係数	Q15	
int32_t	para.enc.plsnum	エンコーダーパルス数	Q0	
int32_t	para.enc.ctrlprd	エンコーダー制御周期	Q16	
uint32_t	para.enc.deg_adjust	電気角調整	Q0	
int32_t	para.delta_lambda	チェンジアップ時電流切替位相	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 c3shunt
boot_type_e	boot_type	起動タイプ	---	cBoot_i cBoot_v

9.5 関数詳細

9.5.1 エンコーダー初期設定(init_ENCen)

9.5.1.1 構文

void init_ENCen(void)

引数：

なし

戻り値：

なし

9.5.1.2 処理内容

エンコーダー回路の初期設定を行います。

・エンコーダー回路設定

・ポート設定

9.5.2 ADC 初期設定(init_ADCen)

9.5.2.1 構文

void init_ADCen(void)

引数：

なし

戻り値：

なし

9.5.2.2 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

9.5.3 PMD 初期設定(init_PMDen)

9.5.3.1 構文

void init_PMDen(void)

引数：

なし

戻り値：

なし

9.5.3.2 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。

9.5.4 VE 初期設定(init_VEen)

9.5.4.1 構文

void init_VEen(void)

引数：

なし

戻り値：

なし

9.5.4.2 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

9.5.5 モーター制御初期設定(B_Motor_Init)

9.5.5.1 構文

void B_Motor_Init(void)

引数：

なし

戻り値：

なし

9.5.5.2 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	
	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 切り替え速度	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	チェンジアップ後待ち時間	Q0	
	para.omega_min	強制転流終了速度	Q15	
	para.omega_v2i	電流制御切替速度	Q15	
	para.delta_lambda	チェンジアップ時電流切替位相	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	
	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	
	para.spd.kp	速度制御比例ゲイン	Q15	
	para.crt.dki	d 軸電流比例ゲイン	Q15	
	para.crt.dkp	d 軸電流積分ゲイン	Q15	
	para.crt.qki	q 軸電流比例ゲイン	Q15	
	para.crt.qkp	q 軸電流積分ゲイン	Q15	
	para.iq_lim	q 軸電流制限値	Q31	
	para.id_lim	d 軸電流制限値	Q31	
	para.err_ovc	過電流設定値	Q15	
	para.enc.pls2theta	パルス数から角度への演算係数	Q32	
	para.enc.deg_adjust	電気角調整	Q0	
	para.enc.plsnum	エンコーダーパルス数	Q0	
	para.enc.pls2omega	パルス数から速度への演算係数	Q15	
	para.enc.ctrlprd	エンコーダー制御周期	Q16	
	para.TrgComp	トリガータイミング補正値	Q15	

9.5.5.3 処理内容

モーター制御パラメーター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。

9.5.6 DAC 制御初期設定(init_Dac)

9.5.6.1 構文

void init_DAC(TSB_TSPI_TypeDef* const TSPIx)

引数：

TSB_TSPI_TypeDef* const TSPIx：TSPI アドレスを選択します。

戻り値：

なし

9.5.6.2 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

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

9.5.7.1 構文

void init_Timer_interval4kHz(void)

引数：

なし

戻り値：

なし

9.5.7.2 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。

9.5.8 ユーザー制御初期設定(init_user_control)

9.5.8.1 構文

void init_user_control(void)

引数：

なし

戻り値：

なし

9.5.8.2 処理内容

SW などのユーザー制御を行うための初期設定を行います。

・デジタルポート初期設定

・アナログポート初期設定

・LED 初期設定

・UART 初期設定

9.5.9 ユーザー制御(uart_control)

9.5.9.1 構文

void user_control(void)

引数：

なし

戻り値：

なし

9.5.9.2 処理内容

VR、SW など外部状態に従って制御します。

・VR1 による、駆動速度制御

・S_SW1 による、回転方向切替(※)

・S_SW2 による、2 相/3 相変調切替(※)

・USW1 による、DAC の切替

・回転速度情報を UART 出力

・内部状態を LED 表示

※SW 切替直後はモーター動作を停止します。

停止後は VR の速度設定を 0(0V)にしてください。

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

9.5.10.1 構文

void B_User_MotorControl(void)

引数：

なし

戻り値：

なし

9.5.10.2 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
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	---	

9.5.10.3 処理内容

モーターの ON/OFF、回転数、駆動方式などの設定を行います。

過電流(ハードウェア EMG)状態のチェックを行います。

9.6 モーター制御関数

モーター制御処理は以下の関数で制御します。

- C_Control_Ref_Model
- C_Common
- C_Stage_Stop
- C_Stage_Bootstrap
- C_Stage_Initposition
- C_Stage_Force
- C_Stage_Change_up
- C_Stage_Steady_A
- C_Stage_Emergency

各状態(ステージ)は、内部状態(サブステージ)を持ちます。

内部状態(サブステージ)は Step0 から開始し、StepEnd まで完了したら、次の状態(ステージ)に遷移します。

またモーターの異常を検出した場合、保護停止状態(Emergency)に遷移します。

9.6.1 状態遷移処理関数(C_Control_Ref_Model)

9.6.1.1 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

9.6.1.2 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	---	

9.6.1.3 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。

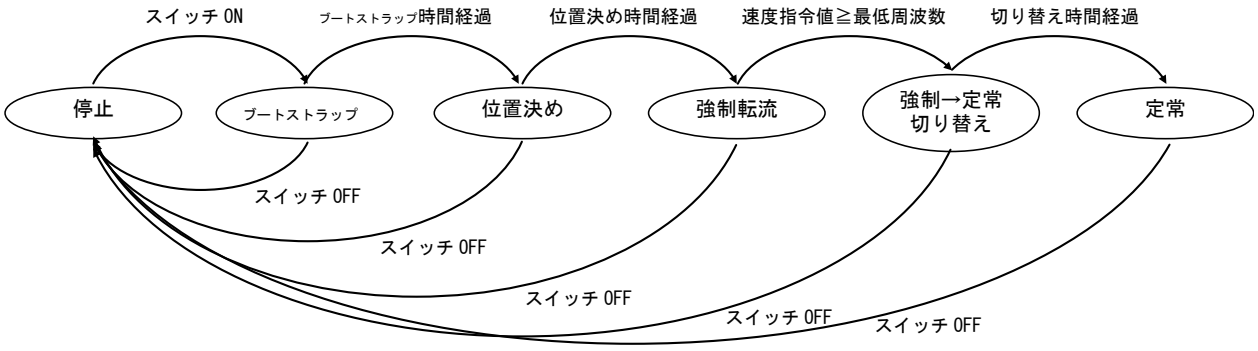


図 9.1 モーター制御の状態遷移図

9.6.2 モーター制御共通処理関数(C_Common)

9.6.2.1 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

9.6.2.2 変数

方向	名前	意味	Q フォーマット	備考
入力	usr.com_user.encoder	エンコーダー制御コマンド	—	
	usr.com_user.modul	変調方式制御コマンド	—	2 相または 3 相変調
出力	drv.command.encoder	エンコーダー駆動コマンド	—	
	drv.command.modul	変調方式駆動コマンド	—	2 相または 3 相変調

9.6.2.3 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。

9.6.10 シフト PWM 制御(C_ShiftPWM_Control)を行います。

9.6.3 停止状態関数(C_Stege_Stop)

9.6.3.1 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

9.6.3.2 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	---	
入出力	stage.sub	サブステージ	---	
出力	stage.itr	割り込みステージ	---	
	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.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	

9.6.3.3 処理内容

モーターを停止させます。(PWM 出力を停止します)

9.6.4 ブートストラップ状態関数(C_Stage_Bootstrap)

9.6.4.1 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

9.6.4.2 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	---	

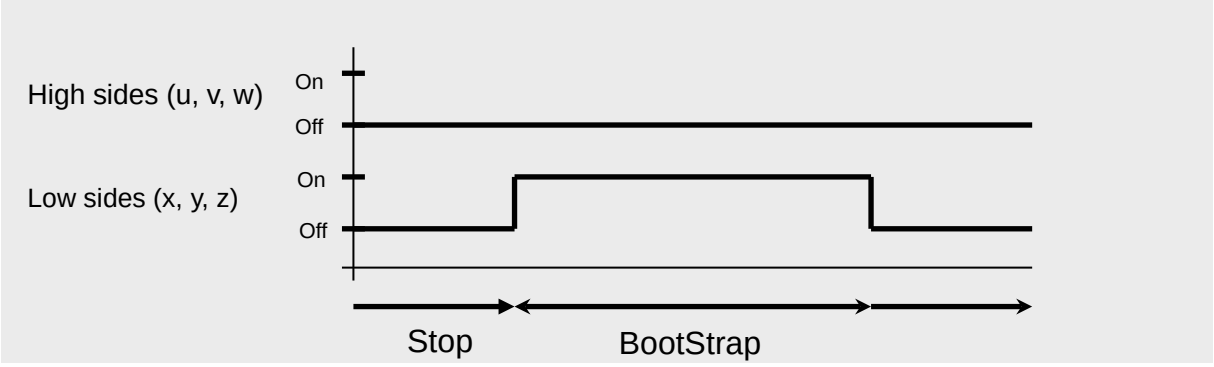
9.6.4.3 処理内容

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。



9.6.5 位置決め状態関数(C_Stage_Initposition)

9.6.5.1 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

9.6.5.2 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	---	
	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	

9.6.5.3 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。

9.6.6 強制転流状態関数(C_Stage_Force)

9.6.6.1 構文

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

9.6.6.2 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	---	
	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

9.6.6.3 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($\theta=\omega_{com} \times t$)。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。

9.6.7 強制定常切替状態関数(C_Stage_Change_up)

9.6.7.1 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

9.6.7.2 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	---	
	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	チェンジアップ時電流切替位相	Q0	
	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
	para.time.go_up	チェンジアップ後待ち時間	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	---	
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	

9.6.7.3 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。

9.6.8 定常状態関数(C_Stage_Steady_A)

9.6.8.1 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

9.6.8.2 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	
	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	
	stage.itr	割り込みステージ	---	
	drv.Id_com	d 軸電流指令値	Q31	

9.6.8.3 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。

9.6.9 保護状態関数(C_Stage_Emergency)

9.6.9.1 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

9.6.9.2 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	---	
	stage.itr	割り込みステージ	---	

9.6.9.3 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。

9.6.10 シフト PWM 制御(C_ShiftPWM_Control)

9.6.10.1 構文

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

9.6.10.2 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 切り替え速度	Q15	
	para.minpls	最小パルス幅	Q0	
出力	drv.minpls	最小パルス幅	Q0	
	drv.command.spwm	シフト PWM 駆動コマンド	—	

9.6.10.3 処理内容

この制御は 1 シャントのときだけ有効です。

シフト PWM の ON/OFF および電流検知前回値を使用する最小 Duty 値の設定を行います。

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表の条件設定となっています。

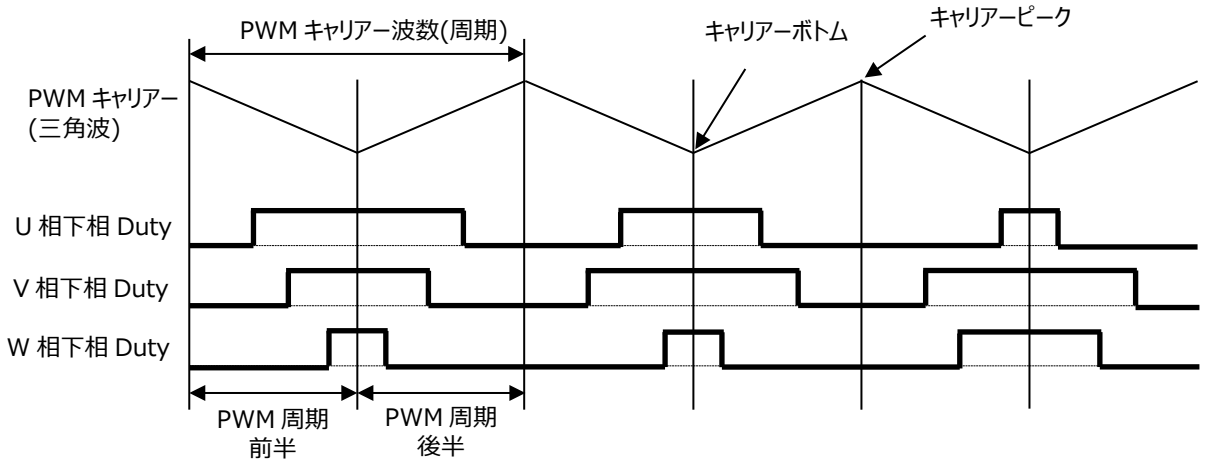
表 9.1　シフト PWM 駆動方法条件

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)
	Initposition_i	All	OFF (0)
	Force_i Change_up Steady_A	目標速度 ≧シフト切り替え速度 drv.omega_com ≧ para.spwm_threshold	OFF (0)
		目標速度 <シフト切り替え速度 drv.omega_com < para.spwm_threshold	ON (1)

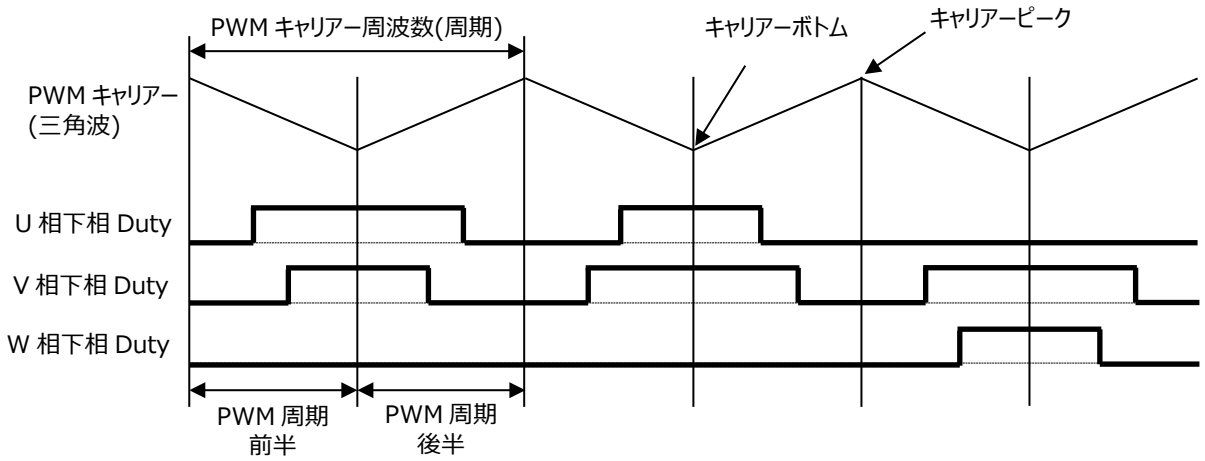
9.7 モーター駆動関数

9.7.1 用語説明

9.7.1.1 3相変調の場合



9.7.1.2 2相変調の場合



9.7.2 モーター電流、電源電圧取得(VE_GetdataFromVEreg)

9.7.2.1 構文

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

引数：

ipdrv_t* const _ipdrv ： IP テーブルアドレスを選択します。

vector_t* const _motor ： モーター制御構造体

戻り値：

なし

9.7.2.2 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
	drv.TrgMd	電流検出モード	―	3 シャント 1 シャント前半 1 シャント後半
入出力	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
出力	drv.Ia	U 相電流	Q15	
	drv.Ib	V 相電流	Q15	
	drv.Ic	W 相電流	Q15	
	drv.Vdc	電源電圧	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_adc	電源電圧変換結果	Q15	

9.7.2.3 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します。
   変換が終了していない場合は、変換が終わるまでループで待機します(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実行)。
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します。

- 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します。

drv.Vdc = drv.Vdc_adc >>1

- モーター電流

<電流検出モード: 3 シャントのとき>

	セクター					
	0	1	2	3	4	5
U 相電流変換結果 drv.Ia_adc	AD 変換結果 2 ADxREG2	AD 変換結果 1 ADxREG1	AD 変換結果 1 ADxREG1	AD 変換結果 0 ADxREG0	AD 変換結果 0 ADxREG0	AD 変換結果 2 ADxREG2
V 相電流変換結果	AD 変換結果 0	AD 変換結果 2	AD 変換結果 2	AD 変換結果 1	AD 変換結果 1	AD 変換結果 0

drv.Ib_adc	ADxREG0	ADxREG2	ADxREG2	ADxREG1	ADxREG1	ADxREG0
W 相電流変換結果 drv.Ic_adc	AD 変換結果 1 ADxREG1	AD 変換結果 0 ADxREG0	AD 変換結果 0 ADxREG0	AD 変換結果 2 ADxREG2	AD 変換結果 2 ADxREG2	AD 変換結果 1 ADxREG1

モーター停止時の AD 変換結果をゼロ電流として、下記変数に格納します。

U 相ゼロ電流変換結果 drv.Iao_ave	U 相電流変換結果 drv.Ia_adc
V 相ゼロ電流変換結果 drv.Ibo_ave	V 相電流変換結果 drv.Ib_adc
W 相ゼロ電流変換結果 drv.Ico_ave	W 相電流変換結果 drv.Ic_adc

3 相電流の値は、ゼロ電流(モーター停止)時の AD 変換結果と相電流変換結果から下記表に従って演算します。

	セクター					
	0	1	2	3	4	5
U 相電流 drv.Ia	-Ib-Ic	.Iao_ave - .Ia_adc	.Iao_ave - .Ia_adc	.Iao_ave - .Ia_adc	.Iao_ave - .Ia_adc	-Ib-Ic
V 相電流 drv.Ib	.Ibo_ave - .Ib_adc	-Ia-Ic	-Ia-Ic	.Ibo_ave - .Ib_adc	.Ibo_ave - .Ib_adc	.Ibo_ave - .Ib_adc
W 相電流 drv.Ic	.Ico_ave - .Ic_adc	.Ico_ave - .Ic_adc	.Ico_ave - .Ic_adc	-Ia-Ib	-Ia-Ib	.Ico_ave - .Ic_adc

<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 ADxREG0	AD 変換結果 1 ADxREG1
電流変換結果 2 drv.Idc2_adc	AD 変換結果 1 ADxREG1	AD 変換結果 0 ADxREG0

モーター停止時の AD 変換結果をゼロ電流として、下記変数に格納します。

ゼロ電流変換結果 drv.Idco_ave	AD 変換結果 0 ADxREG0

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します。

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

9.7.3 クラーク変換(E_Clarke)

9.7.3.1 構文

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

引数：

q15_t _iu	: U 相電流
q15_t _iv	: V 相電流
q15_t _iw	: W 相電流
q15_t* _ialpha	: α軸電流格納アドレス
q15_t* _ibeta	: β軸電流格納アドレス

戻り値：

なし

9.7.3.2 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、

$I\alpha = Iu$

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

となります。

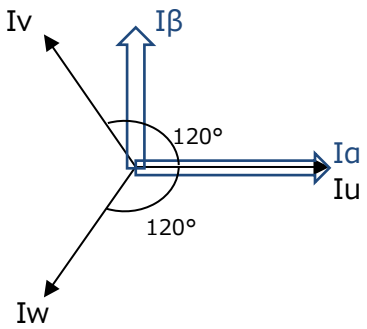


図 9.2　クラーク変換

9.7.4 パーク変換(E_Park)

9.7.4.1 構文

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

引数：

q15_t _ialpha	: α軸電流 Iα
q15_t _ibeta	: β軸電流 Iβ
q15_t _theta	: ローター位置θ：0 ≦ 位置 <360°(0~0xFFFF)
q15_t* _id	: d 軸電流 Id 格納アドレス
q15_t* _iq	: q 軸電流 Iq 格納アドレス

戻り値：

なし

9.7.4.2 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。

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

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

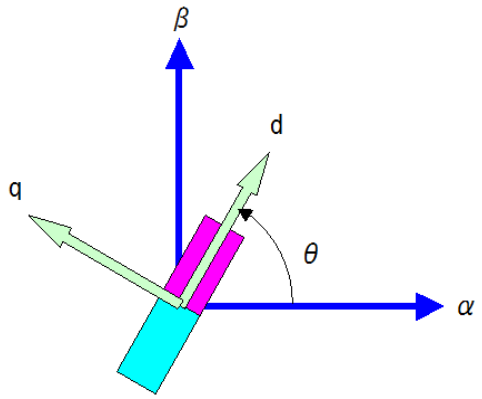


図 9.3　パーク変換

9.7.5 位置推定関数(D_Detect_Rotor_Position)

9.7.5.1 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

9.7.5.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.command.encoder	エンコーダー駆動コマンド	---	
	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	

9.7.5.3 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。

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。

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

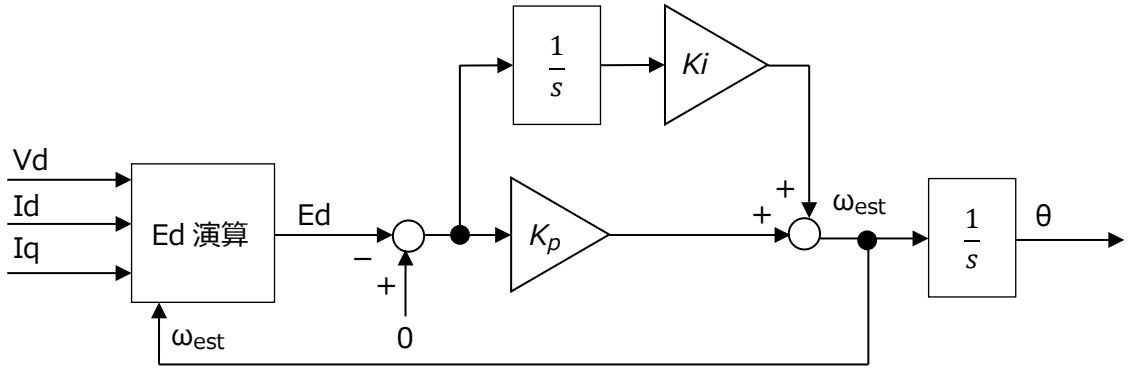


図 9.4 d 軸誘起電圧 Ed による位置推定ブロック図

9.7.6 エンコーダー制御関数(H_Encoder)

9.7.6.1 構文

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

引数：

vector_t* const _motor　：モーター制御構造体

TSB_EN_TypeDef* const ENx　：ENC アドレス

戻り値：

なし

9.7.6.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.command.encoder	エンコーダー駆動コマンド	—	
	drv.omega_com	駆動速度指令値	Q31	
	drv.theta_com	電気角指令値	Q0	
	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	—	
	para.enc.ctrlprd	エンコーダー制御周期	Q16	
	para.enc.deg_adjust	電気角調整	Q0	
	para.enc.pls2omega	パルス数から速度への演算係数	Q15	
	para.enc.pls2theta	パルス数から角度への演算係数	Q32	
	para.enc.plsnum	エンコーダーパルス数	Q0	

入出力	drv.EnCnt	エンコーダーカウンター値	Q0	
	drv.EnCnt_dev	エンコーダーカウンター偏差	Q0	
	drv.EnCnt1	エンコーダーカウンター前回値	Q0	
	drv.omega_enc	エンコーダー速度	Q15	
	drv.omega_enc_ave	エンコーダー速度平均値	Q31	
	drv.omega_enc_raw	エンコーダー速度	Q15	
	drv.theta_enc	エンコーダー角度	Q0	
出力	drv.omega	速度	Q31	
	drv.theta	ローター位置	Q0	

9.7.6.3 処理内容

インクリメンタル・エンコーダーパルス数を読み出し、ローター位置(角度)θ と、速度ωの算出処理を行います。

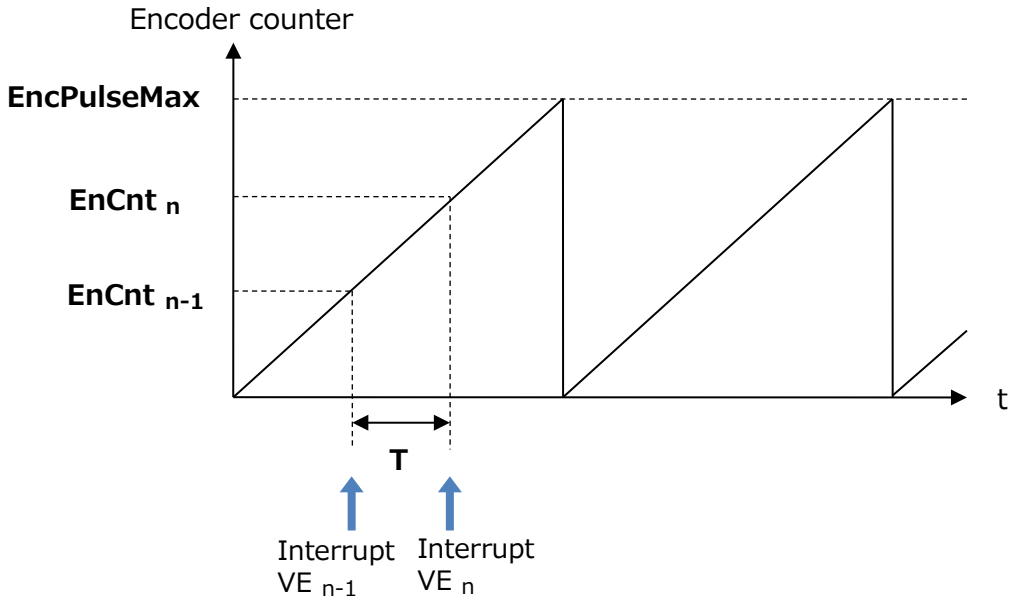


図 9.5　エンコーダーパルスからの位置および速度演算

VE 割り込みごとに、パルス数を読み込み、下記演算を行います。

現在のエンコーダーパルスカウンター値から下記式により θ を演算します。

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

前回 n-1 と現在 n のエンコーダーパルスカウンター値との差から下記式により速度 ω を演算します。

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

$\mathrm{EnCnt_n}$　　現在エンコーダーカウンター値

$EnCnt_{n-1}$	前回エンコーダーカウンター値
EncPulseMax	エンコーダーパルス数[ppr]×逓倍数(4)
Pole	極数
θ	角度[deg.]
ω	速度[Hz]
T	速度演算周期[s]

9.7.7 速度制御関数(D_Control_Speed)

9.7.7.1 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

9.7.7.2 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	d 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	

9.7.7.3 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。

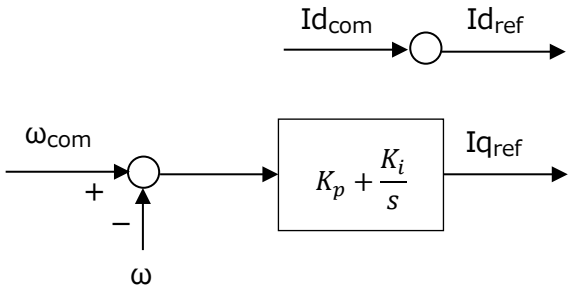


図9.6速度制御ブロック図

9.7.8 逆パーク変換(E_InvPark)

9.7.8.1 構文

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

引数：

q31_t _d	: d 軸
q31_t _q	: q 軸
q31_t _theta	: ローター位置θ：0 ≦ 位置 ＜ 360°(0~0xFFFF)
q31_t _alpha	: α軸格納アドレス
q31_t _beta	: β軸格納アドレス

戻り値：

なし

9.7.8.2 処理内容

回転座標の dq 軸から αβ 軸の静止座標に変換します。

下記演算式により Vα、Vβ を演算します。

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

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

9.7.9 セクター演算

9.7.9.1 構文

uint8_t D_CalSector(q31_t _valpha, q31_t _vbeta)

引数：

q31_t _valpha	: α軸電圧
q31_t _vbeta	: β軸電圧

戻り値：

セクター

9.7.9.2 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
	$\lvert V\alpha\rvert < (\lvert V\beta\rvert/\sqrt{3})$	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

9.7.10 空間ベクトル変調(D_SVM)

9.7.10.1 構造

```
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：

q31_t _valpha	: α軸 Vα
q31_t _vbeta	: β軸 Vβ
q31_t _vdc	: 電源電圧
uint8_t _sector	: セクター
uint8_t _modul	: 変調方式
uint16_t* _p_vu	: U 相 Duty 比 Du 格納アドレス
uint16_t* _p_vv	: U 相 Duty 比 Dv 格納アドレス
uint16_t* _p_vw	: U 相 Duty 比 Dw 格納アドレス

戻り値：

なし

9.7.10.2 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 または 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 9.1 のように 8 パターン存在します。

この 8 パターンを V0～V7 ベクトルとすると、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～V6)は図 9.7 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。

表 9.1 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

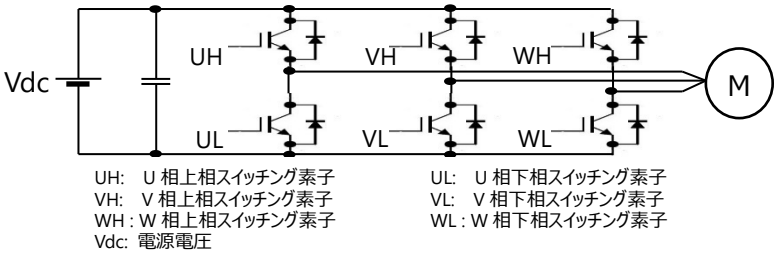


図 9.7 インバーター駆動回路

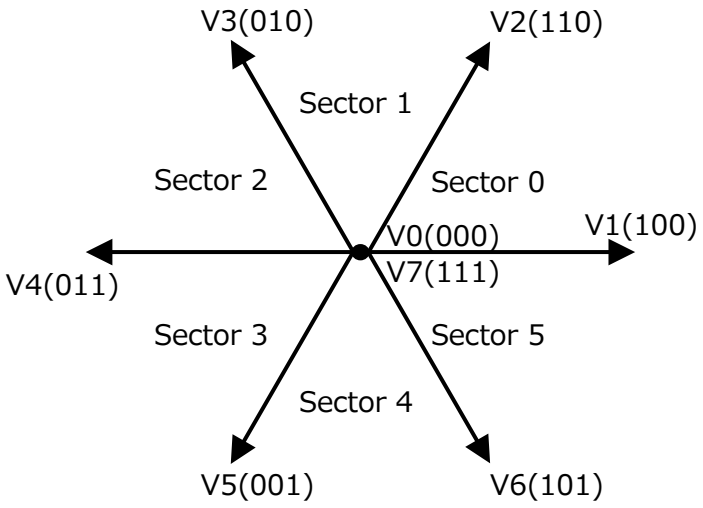


図 9.8 空間ベクトル

・出力電圧ベクトル V がセクター0 にある場合

例えば図 9.8 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 9.9 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 V を合成します。

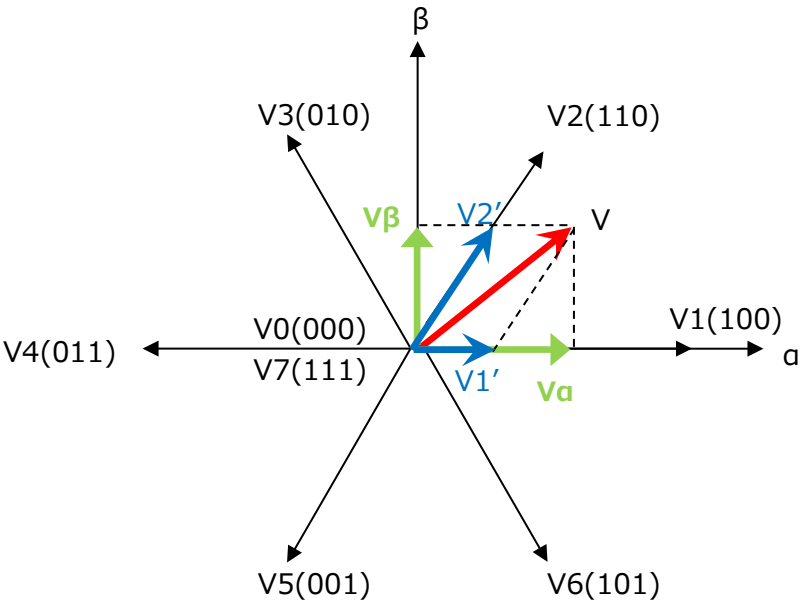


図 9.9　セクター0 時の電圧ベクトル

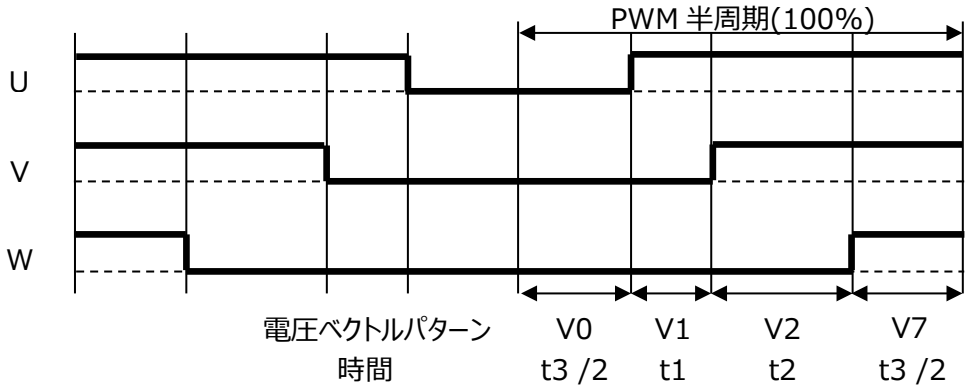


図 9.10　3 相変調時の PWM 波形

図 9.8 より、Vα,Vβ は、

$$V\alpha = V1' \times \cos 0° + V2' \times \cos 60° = V1' + \frac{V2'}{2}$$

$$V\beta = V1' \times \sin 0° + V2' \times \sin 60° = \frac{\sqrt{3}}{2} \times V2'$$

と表すことができます。

よって、V1′,V2′は、

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

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

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 9.2 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 9.2 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	$-k \times \frac{\frac{2}{\sqrt{3}}V\beta}{Vdc}$	$k \times \frac{-V\alpha + \frac{1}{\sqrt{3}}V\beta}{Vdc}$	
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 9.3 の演算式で算出します。

表 9.3　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

10. 定数定義説明

10.1 モータードライバー設定用パラメーター(D_Para.h)

10.1.1 モーター制御チャネル選択

__CONTROL_MOTOR_CH0 ：CH0 を制御するとき定義してください。
__CONTROL_MOTOR_CH1 ：CH1 を制御するとき定義してください。
__CONTROL_MOTOR_CH2 ：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 指令速度単位選択

__TGTSPD_UINT ：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

10.1.4 パラメーター一覧

パラメーター名	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
	--
	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。
cOMEGAENC_AVE	エンコーダーパルス数からの速度検出値用フィルター係数
	--
	速度検出用のフィルター係数を設定してください。 エンコーダーパルス数が小さい場合は、値を大きめにしてください。

	ただし、値を大きくすると検出の遅れが発生しますので、遅れが問題ない値に設定してください。

10.2 モータードライバー設定用パラメーターモーターチャネル(D_Para_chx.h) x=0,1,2

モータードライバー部のパラメーター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。

10.2.1.2 センサーレス/センサー選択

__USE_ENCODER　　エンコーダー制御を行う場合、定義してください。

10.2.2 モーターチャネル別パラメーター一覧

パラメーター名	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 変換が 1LSB 変化する電源電圧の変化量[V]×2^12 の値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 変換が 1LSB 変化する相電流の変化量[A]×2^11 の値を設定してください。
cSHUNT_TYPE	電流取得方式（3 シャント）の設定
	3:3 シャント
	3 シャント以外を設定するとエラーとなります。

cBOOT_TYPE	起動時の駆動方式の設定
	cBoot_i:電流型駆動/ cBoot_v:電圧型駆動
	起動時の駆動方式を設定してください。
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_U	U 相電流取得 AD チャネル設定
	AINx
	U 相電流を検出する AD チャネルを設定してください。
cADCH_CURRENT_V	V 相電流取得 AD チャンル設定
	AINx
	V 相電流を検出する AD チャネルを設定してください。
cADCH_CURRENT_W	W 相電流取得 AD チャネル設定
	AINx
	W 相電流を検出する AD チャネルを設定してください。
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[μs] 分解能 16.67ns@120MHz
	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[μs] 分解能 66.67ns@120MHz
	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。 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[Ω]
	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]
	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:1ms
	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:1ms
	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:1ms
	位置決め後の待ち時間を設定してください。
cGOUP_DELAY_LEN	チェンジアップ後待ち時間
	単位[s] 分解能:1ms
	チェンジアップ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 ～ 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。
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 詳細は電圧型起動時の定数値を参照してください。
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替速度
	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
cENC_PULSE_NUM	エンコーダーパルス数設定
	単位[ppr]
	エンコーダーのパルス数を設定してください。 エンコーダーパルス数が小さい場合、速度検出精度が悪くなります。 検出速度が安定しない場合は、フィルター係数 cOMEGAENC_AVE の値を大きくしてください。
cENC_DEG_ADJUST	エンコーダー位置調整
	単位[deg.]
	電気角の 0°と Z 相の位置ずれ角度を設定してください。
cENC_NOISE_TIME	エンコーダー入力信号ノイズキャンセル時間設定 (A-ENC だけ有効)
	単位[μs] 分解能 fc (8.33ns@120MHz)

	エンコーダー入力信号のノイズキャンセル時間を設定してください。 この設定時間より短い信号は、キャンセルします。 この設定は A-ENC 搭載マイコンのときだけ有効となります。
__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

10.2.3 定数設定値と波形の関係

電流型起動時の定数値

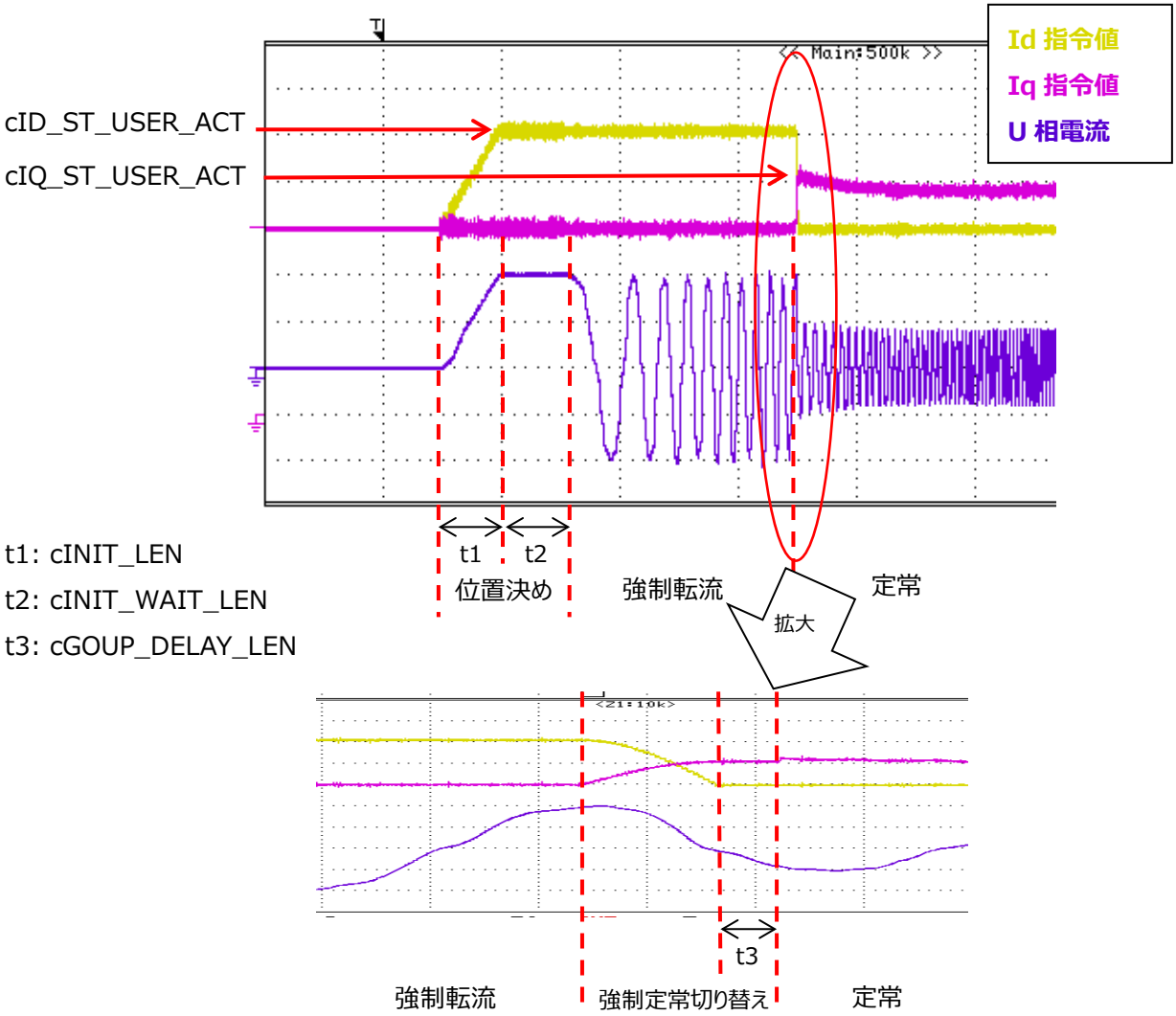


図 10.1　始動電流波形(電気角 0°に位置決め)

チェンジアップステージで、cID_ST_USER_ACT と cIQ_ST_USER_ACT の値を入れ替える。
入れ替え後、cGOUP_DELAY_LEN の時間 Iq 指令値一定値で制御。
定常移行後、Iq 指令値は PI 制御により演算。

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。

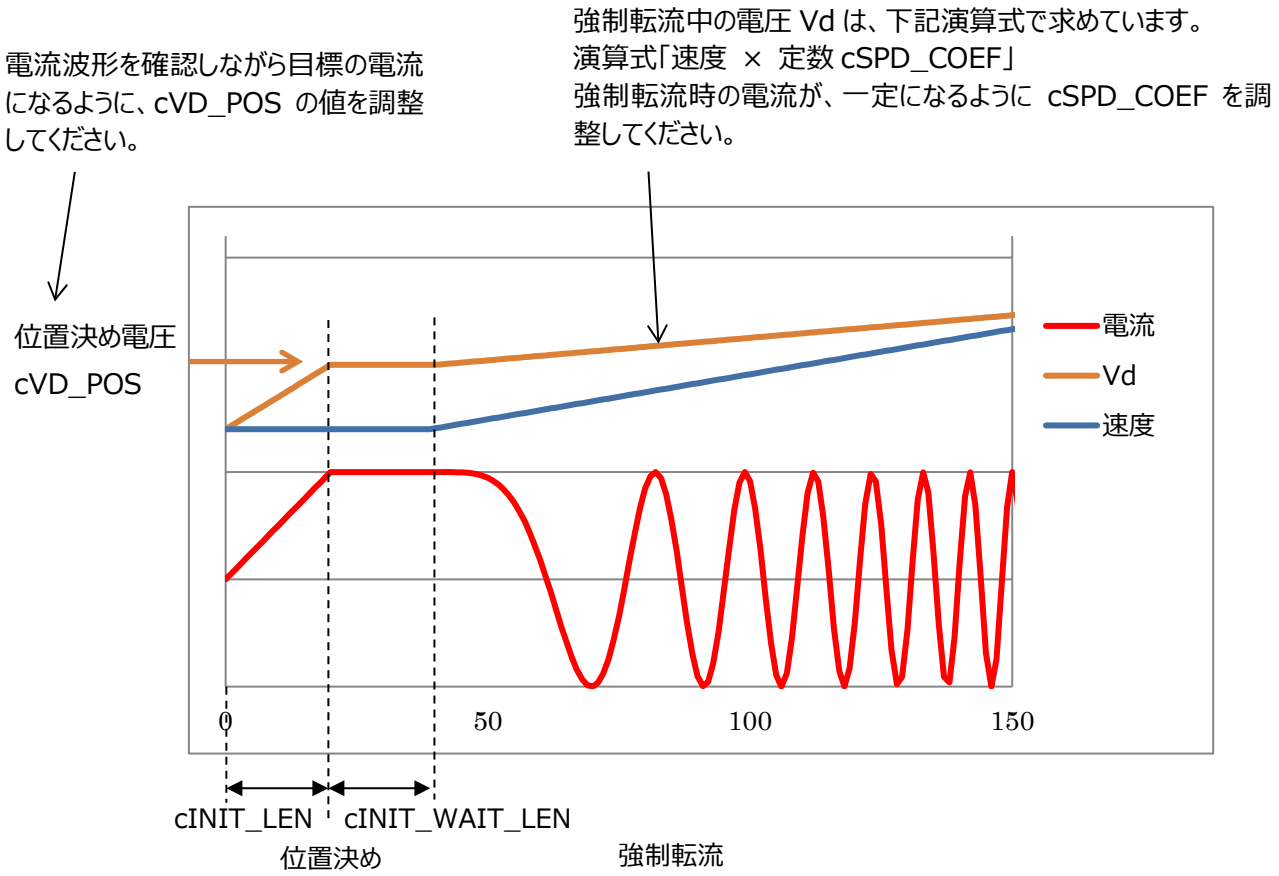


図 10.2 電圧駆動時のパラメーター調整方法

10.3 ユーザー制御関連定数

```
/*Initial value*/
#define cININT_VALUE  (0)
#define cININT_FF             (0xff)

/* Key settting */
#define P_SW1         TSB_PG_DATA_PG3     /* slideSW1 or 8(KEY1) */
#define P_SW2         TSB_PG_DATA_PG4     /* slideSW2 or 9(KEY2) */
#define P_PSW1                TSB_PD_DATA_PD1     /* pushSW1 DACSW */

#define cKEY_CHATA_CNT      (20)    /* [cnt] chattering counter for KEY SW */
#define cNO_ENTER_KEY       (0)     /* Soft ADC Setting */
#define cADUNIT_USR  TSB_ADC        /* User ad data ADCUnit */
```

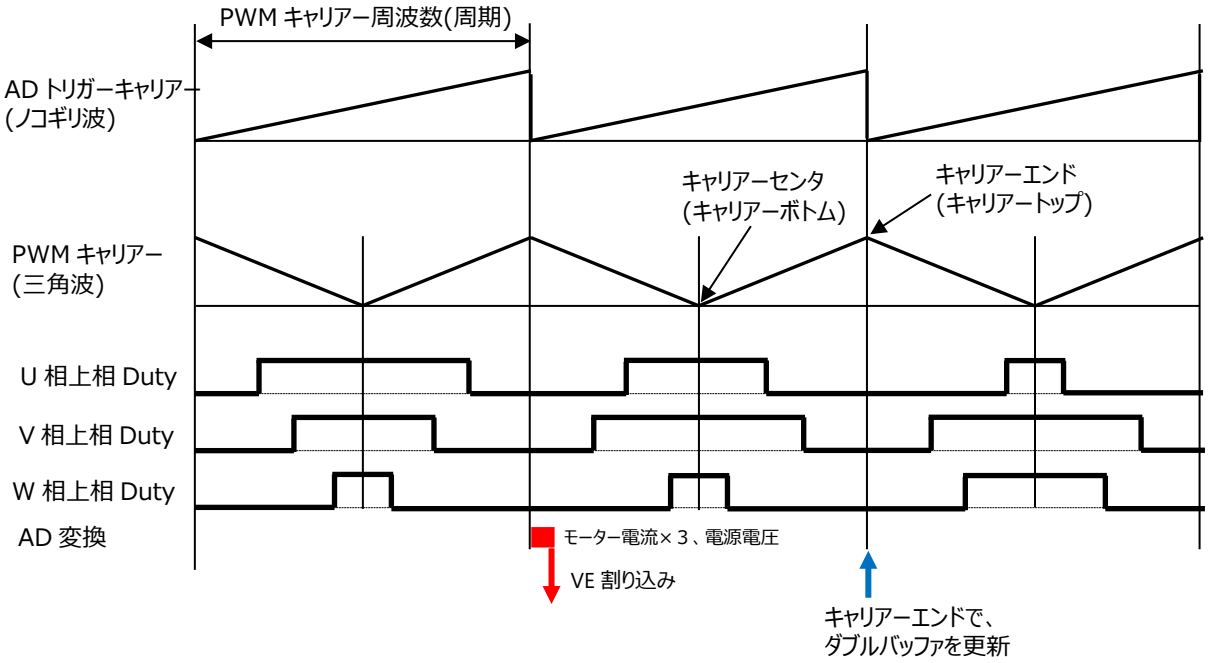
```
#define cADAUNIT_USR          TSB_ADA          /* User ad data ADAUnit */
#define cADTRG_USR            ADC_TRG_SINGLE       /* ADC Trigger Type */
#define cADCH_ADKEY           ADC_AIN0         /* ADC Channel for AD Key */
#define cADREG_ADKEY          ADC_REG4         /* Result register No for AD Key */
#define cADCH_VR              ADC_AIN0         /* ADC Channel for VR */
#define cADREG_VR             ADC_REG4         /* Result register No for VR */
#define cADCH_TEMP            ADC_AIN13        /* ADA Channel for TEMP */
#define cADREG_TEMP           ADC_REG7         /* Result register No for TEMP */
#define cADAVECNT             (10)             /* ADC average count */
#define cPUSHSW_CHATA_CNT (5)                  /* [cnt] Chattering counter for Push SW */


/* AD speed control setting */
#define cAD_MIN               (0x10)           /* Voltage value 0.3V */
#define cAD_MAX               (0xF0)           /* Voltage value 4.7V */
#define cSPEED_USER_MIN       (12)             /* [Hz] Min Target speed of motor */
#define cSPEED_USER_MAX       (200)            /* [Hz] Max Target speed of motor */


/* AD Temp setting */
#define cONE_ABOVE            (1)


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


/* LED setting */
#define LED_FORCED    TSB_PV_DATA_PV0              /* LED4 */
#define LED_EMG       TSB_PB_DATA_PB6              /* LED6 */
#define cLED_ON               (1)                  /* LED ON level */
#define cLED_OFF              (0)                  /* LED OFF level */


/* flag status */
#define cFLG_ON               (1)
#define cFLG_OFF              (0)
```

```
/* UART Setting */
#define UART_BRD_INIT       ((uint32_t)0x00A6002B)
#define UART_CLK_INIT       ((uint32_t)0x00000000)
#define UART_CR0_INIT       ((uint32_t)0x00000001)
#define UART_CR1_INIT       ((uint32_t)0x00000040)
#define UART_SEND_WAIT      (1000)
#define UART_COUNT          (0)
#define c10MHz              (1000000)
#define cLOW_ACTIVE         (0)
#define cSWRST10            (0x2U)
#define cSWRST01            (0x1U)
#define cSWRSTF             (0x80)


/* User_interface Setting */
#define cROTATION_CW        (1)
#define cROTATION_CCW       (0)
#define cPSW1_Hi            (1)
#define cPSW1_Low           (0)
#define cStrCH0             ("CH0")
#define cINV_MIN_TEMP       (-20)
#define cINV_MAX_TEMP       (100)

#define cSendBufSIZE        (80+1)
```

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 3 シャント制御



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

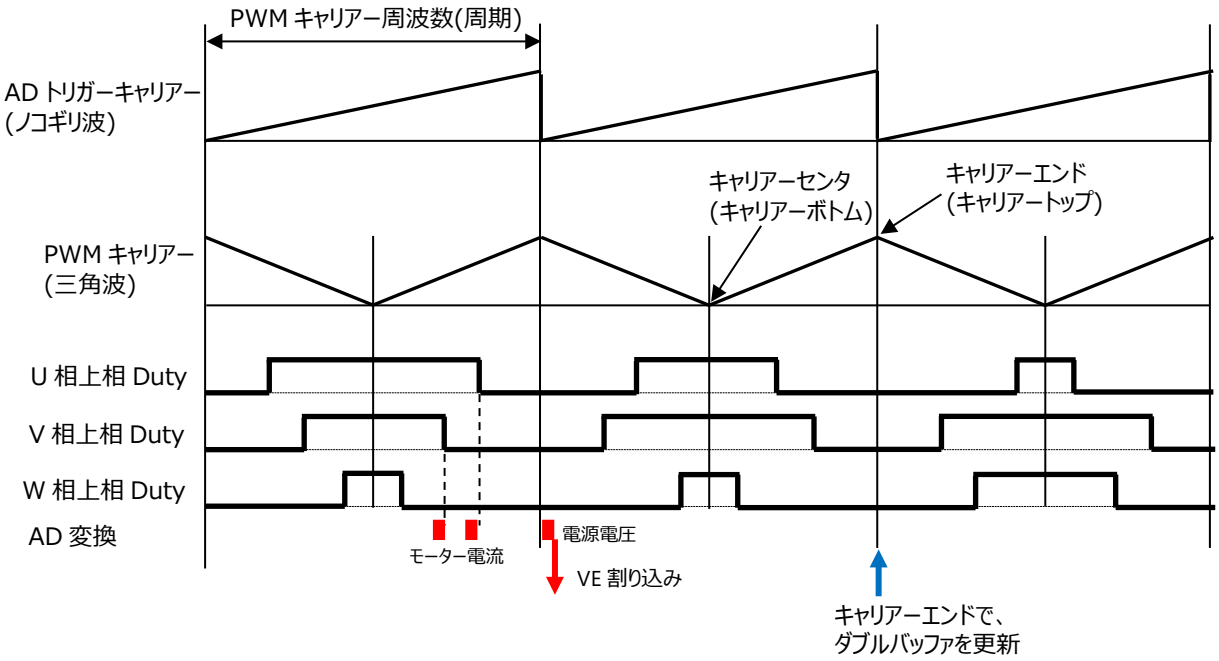
ダブルバッファー更新タイミング

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

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	モーター電流×3、電源電圧の全ての AD 変換終了後
AD トリガー	—	PWM と同じ頻度

11.1.2 1 シャント制御



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

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

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

12. ペリフェラルドライバー

12.1 ベクトルエンジン(A-VE+)

12.1.1 関数仕様

IP_VE_init

VE 初期設定

API：

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

パラメーター：

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 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。

リターンパラメーター：

なし

VE_Start

VE スタート

API：

VE_Start(const ipdrv_t* const _ipdrv)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

機能：

VE を開始します。

補足：

特になし

リターンパラメーター：

なし

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)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。
_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

リターンパラメーター：

なし

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)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。
_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

リターンパラメーター：

なし

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)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

_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

リターンパラメーター：

なし

VE_GetPWM_DutyU

U 相 Duty 値取得

API:

uint32_t VE_GetPWM_DutyU(const ipdrv_t* const _ipdrv)

パラメーター:

_ipdrv：IP テーブルアドレスを選択します。

機能:

U 相 Duty レジスターの値を取得します。

補足:

特になし

リターンパラメーター:

U 相 Duty

VE_GetPWM_DutyV

V 相 Duty 値取得

API:

uint32_t VE_GetPWM_DutyV(const ipdrv_t* const _ipdrv)

パラメーター:

_ipdrv：IP テーブルアドレスを選択します。

機能:

V 相 Duty レジスターの値を取得します。

補足:

特になし

リターンパラメーター:

V 相 Duty

VE_GetPWM_DutyW

W 相 Duty 値取得

API:

uint32_t VE_GetPWM_DutyW(const ipdrv_t* const _ipdrv)

パラメーター:

_ipdrv：IP テーブルアドレスを選択します。

機能:

W 相 Duty レジスターの値を取得します。

補足:

特になし

リターンパラメーター:

W 相 Duty

VE_GetPWM_DutyMed

Duty 中間値取得

API：

uint32_t VE_GetPWM_DutyMed(const ipdrv_t* const _ipdrv)

パラメーター：

_ipdrv: IP テーブルアドレスを選択します。

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

リターンパラメーター：

Duty 中間値

VE_GetPWM_Modulation

変調方式取得

API:

int VE_GetPWM_Modulation(const ipdrv_t* const _ipdrv)

パラメーター:

_ipdrv: IP テーブルアドレスを選択します。

機能:

変調方式を取得します。

補足:

特になし

リターンパラメーター:

変調方式 0:3 相変調、1:2 相変調

VE_GetSector

現在セクター取得

API:

uint32_t VE_GetSector(const ipdrv_t* const _ipdrv)

パラメーター:

_ipdrv: IP テーブルアドレスを選択します。

機能:

現在のセクター値を取得します。

補足:

特になし

リターンパラメーター:

現在セクター値 [0-11]

VE_GetShiftPWMState

シフト PWM 状態取得

API:

int VE_GetShiftPWMState(const ipdrv_t* const _ipdrv)

パラメーター:

_ipdrv: IP テーブルアドレスを選択します。

機能:

シフト PWM 状態を取得します。

補足:

特になし

リターンパラメーター:

シフト PWM 状態 0:通常 PWM、1:シフト PWM

VE_GetOutPutMode

PWM 出力状態取得

API：

int VE_GetOutputMode(const ipdrv_t* const _ipdrv)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

機能：

PWM 出力状態を取得。

補足：

特になし

リターンパラメーター：

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

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)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

_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

リターンパラメーター：

なし

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)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

_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

リターンパラメーター：

なし

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：

void

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

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

_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

リターンパラメーター：

なし

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)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

_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

リターンパラメーター：

なし

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)
```

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

_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：

特になし

リターンパラメーター：

なし

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)
```

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

_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

リターンパラメーター：

なし

VE_SetdataToVEreg_Change_up

VE レジスターへのデータセット(チェンジアップ)

API：

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

パラメーター：

_ipdrv： IP テーブルアドレスを選択します。

_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：

チェンジアップ状態の VE レジスターへの設定処理を行います。

補足：

特になし

リターンパラメーター：

なし

VE_SetdataToVEreg_Steady_A

VE レジスターへのデータセット(定常)

API：

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

パラメーター：

_ipdrv： IP テーブルアドレスを選択します。

_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

リターンパラメーター：

なし

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)

パラメーター：

_ipdrv： IP テーブルアドレスを選択します。

_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

リターンパラメーター：

なし

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)

パラメーター：

_ipdrv： IP テーブルアドレスを選択します。

_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。

補足：

特になし

リターンパラメーター：

なし

VE_SetVDCreg

モーター電圧と Vdc 設定

API：

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

パラメーター：

_ipdrv: IP テーブルアドレスを選択します。

_dat: モーター電圧を設定します。

機能：

モーター電圧を VE レジスターに設定します。

補足：

特になし

リターンパラメーター：

なし

VE_SetSpwmMode

シフト PWM モード設定

API:

void

VE_SetSpwmMode(TSB_VE_TypeDef* const VEx, uint8_t _dat);

パラメーター:

VEx:VE アドレスを選択します。

_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

リターンパラメーター:

なし

VE_SetModulType

変調方式設定

API：

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

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

_dat：変調方式を設定します。

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

リターンパラメーター：

なし

12.1.2 データ構造

VE_InitTypeDef

Data Fields :

uint8_t ve_ch：ベクトルエンジンチャネル
0：チャネル 0
1：チャネル 1
uint8_t shunt：シャントタイプ
3：3 シャント
1：1 シャント
uint16_t pwmfreq：キャリアー周波数
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

12.2 モーター制御回路(PMD)

12.2.1 関数仕様

IP_PMD_init

PMD 初期設定

API :

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

パラメーター：

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 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。

リターンパラメーター：

なし

PMD_GetEMG_Status

EMG 保護状態取得

API：

emg_status_e PMD_GetEMG_Status(const ipdrv_t* const _ipdrv)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

機能：

EMG 保護状態を取得します。

補足：

特になし

リターンパラメーター：

emg_status_e：EMG 保護状態

cNormal：正常

cEMGProtected：保護中

PMD_ReleaseEMG_Protection

EMG 保護状態解除

API：

void PMD_ReleaseEMG_Protection(const ipdrv_t* const _ipdrv)

パラメーター：

_ipdrv：IP テーブルアドレスを選択します。

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。

リターンパラメーター：

なし

12.2.2 データ構造

PMD_InitTypeDef

Data Fields :

uint8_t **shunt :** シャントタイプ
3：3 シャント
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 **pwmfreq :** PWM 周波数
PMDxMDPDR に設定する値

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

12.3 アナログデジタルコンバーター(ADC)

12.3.1 関数仕様

IP_ADC_init

ADC 初期設定

API :

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

パラメーター :

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 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。

リターンパラメーター :

なし

12.3.2 データ構造

AD_InitTypeDef

Data Fields :

uint8_t shunt：シャントタイプ
3：3 シャント
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
cPMD：PMD チャネル 0
cPMD：PMD チャネル 1
cPMD：PMD チャネル 2

uint8_t pints：PM トリガー用割り込み選択
cPINTS_B：ITTADxPDA
cPINTS_B：ITTADxPDB

12.4 定数説明

12.4.1 A-VE 搭載マイコン用定数(mcuip_drv.h)

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-phase mode	PWMMD_SWTOOTH	edge-aligned PWM, sawtooth wave
		PWMMD_TRIANGLAR	center-aligned PWM, triangular wave
		PWMMD_SWRROTH_REV	
		PWMMD_TRAIN	
cVPWMMD	V-phase mode	PWMMD_SWTOOTH	edge-aligned PWM, sawtooth wave
		PWMMD_TRIANGLAR	center-aligned PWM, triangular wave
		PWMMD_SWRROTH_REV	
		PWMMD_TRAIN	
cUPWMMD	U-phase mode	PWMMD_SWTOOTH	edge-aligned PWM, sawtooth wave
		PWMMD_TRIANGLAR	center-aligned PWM, triangular wave
		PWMMD_SWRROTH_REV	
		PWMMD_TRAIN	
cDSYNCS	Double buffer	DSYNCS_INTCYC	Depends on interrupt cycle

	update timing for the duty compare register and PWM period register.		setting
		DSYNCS_CENTER	Updates at PWM carrier center
		DSYNCS_END	Updates at PWM carrier end
		DSYNCS_CENTER_END	Updates at both PWM carrier center and end
cDCMEN	Port output mode	DCMEN_DISABLE	Correction disable
		DCMEN_ENABLE	Correction enable
cDTCREN	Dead time correction	DTCREN_DISABLE	Correction disable
		DTCREN_ENABLE	Correction enable
cPWMSYNT	Port output mode	SYNTMD_0	
		SYNTMD_1	
cPWMSPCFY	Duty mode	DTYMD_COMMON	3-phase common mode
		DTYMD_INDEPENDENT	3-phase independent mode
cPWMINT	PWM interrupt timing	PINT_BOTTOM	Interrupt request when PWM counter PMD1MDCNT<MDCNT[15:0]> = 0x0001
		PINT_TOP	Interrupt request when PWM counter PMD1MDCNT<MDCNT[15:0]> = <MDPRD[15:0]>
cPWMINTPRD	PWM interrupt period	INTPRD_HALF	Interrupt request at every 0.5 PWM period
		INTPRD_1	Interrupt request at every PWM period
		INTPRD_2	Interrupt request at every 2 PWM periods
		INTPRD_4	Interrupt request at every 4 PWM periods

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 transfer timing for trigger synchronous.	SYNCS_ASYNC	Asynchronous
		SYNCS_INTENC	when INTENCx (ENCx interrupt request) occurs
		SYNCS_NTTB	when INTT32Ax0 (TMRBx interrupt request) occurs
		SYNCS_CTRGO	when CTRGO(ENCx MCMP completed) occurs
cPSYNCS	MDOUT transfer timing for PWM synchronous.	PSYNCS_WRITE	Reflect when write
		PSYNCS_CENTER	Reflect when PWM carrier center
		PSYNCS_END	Reflect when PWM carrier end
		PSYNCS_CENTER_END	Reflect when PWM carrier center and end

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	Port control settings at tool break	PORTMD_ALLHIZ	Upper phases = High-z / lower phases = High-z
		PORTMD_UHIZ_LON	Upper phases = High-z / lower phases = PMD output

		PORTMD_UON_LHIZ	Upper phases = PMD output / lower phases = High-z
		PORTMD_ALLON	Upper phases = PMD output / lower phases = PMD output

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	Trigger timing setting for 3shunt	TRGMD_DIS	Trigger output disabled
		TRGMD_DOWN	Trigger output at down-count match
		TRGMD_UP	Trigger output at up-count match
		TRGMD_UPDOWM	Trigger output at up-/down-count match
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak and bottom
cTRGMD_1SHT	Trigger timing setting for 1shunt	TRGMD_DIS	Trigger output disabled
		TRGMD_DOWN	Trigger output at down-count match
		TRGMD_UP	Trigger output at up-count match
		TRGMD_UPDOWM	Trigger output at up-/down-count match
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak and bottom
cBUFSYNC	Triger Buffer update timing	TRGBE_SYNC	Sync
		TRGBE_ASYNC	Async The value written to PMDTRGx is immediately reflected.)
cCARSEL	Selection of a comparison career	CARSEL_BASE	Compared by the base carrier
		CARSEL_PHASE	Compared by TEMPREG0/1=U,REG2=V,REG3=W-phase carrier

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	Output enable in EMG protection state	EMGTGE_DISABLE	Disable trigger output in the protection state
		EMGTGE_ENABLE	Enable trigger output in the protection state

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

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

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	Selects a PWM shift mode	SPWMMD_SPWM1	Shift 1
		SPWMMD_SPWM2U	Shift 2 (U-phase Normal PWM)
		SPWMMD_SPWM2V	Shift 2 (V-phase Normal PWM)
		SPWMMD_SPWM2W	Shift 2 (W-phase Normal PWM)
cCCVMD	Phase-change mode selection	CCVMD_RELATIVE	relative
		CCVMD_ABSOLUTE	absolute
cPHCVDIS	Phase transformation	PHCVDIS_ENABLE	2-3 phase transformation is enabled
		PHCVDIS_DISABLE	2-3 phase transformation is disabled
cVSLIMMD	Controls voltage scalar limitation	VSLIMMD_DIS	Scalar limitation is disabled
		VSLIMMD_D	Limits the voltage on the d-axis
		VSLIMMD_Q	Limits the voltage on the q-axis
		VSLIMMD_DQ	dq proportional limitation
cMREGDIS	Keep the previous value of SIN/COS/SECTOR	MREGDIS_EFFECTIVE	Effective
		MREGDIS_NOEFFECTIVE	No effective
cCRCEN	Trigger correction	CRCEN_DISABLE	Disable trigger correction.
		CRCEN_ENABLE	Enable trigger correction.
cIDQSEL	De-coupling mode selection	IDQSEL_FEEDBACK	Feedback current(ID, IQ)
		IDQSEL_INSTRUCTION	Instruction current(IDREF, IQREF)
cIAPLMD	Direction of current detection	IPLMD_SHUNT	Shunt
		IPLMD_SENSOR	Sensor
cIBPLMD	Direction of current detection	IPLMD_SHUNT	Shunt
		IPLMD_SENSOR	Sensor
cICPLMD	Direction of current detection	IPLMD_SHUNT	Shunt
		IPLMD_SENSOR	Sensor

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	Current polarity determination	IPDEN_DISABLE	Current polarity determination is disabled.
		IPDEN_ENABLE	Current polarity determination is enabled.
cPMDDTCEN	Dead time correction control of the PMD	PMDDTCEN_DISABLE	Dead time correction of the PMD is disabled.
		PMDDTCEN_ENABLE	Dead time correction of the PMD is enabled.
cPWMFLEN	Duty of 100% setting when the upper-limit	PWMFLEN_DISABLE	Duty of 100% setting is disabled.
		PWMFLEN_ENABLE	Duty of 100% setting is enabled.
cPWMBLEN	Duty of 0% setting when the lower-limit	PWMBLEN_DISABLE	Duty of 0% setting is disabled.
		PWMBLEN_ENABLE	Duty of 0% setting is enabled.
cNICEN	Non-interference control	NICEN_DISABLE	Non-interference control is disabled.
		NICEN_ENABLE	Non-interference control is enabled.
cT5ECEN	Expansion control	T5ECEN_DISABLE	Expansion control is disabled.
		T5ECEN_ENABLE	Expansion control is enabled.
cAWUMD	Specifies anti-windup (AWU) control	AWUMD_DISABLE	AWU control is disabled.
		AWUMD_ENABLE4	Substitutes the result from amount of limitation ÷ 4 into the integral term.
		AWUMD_ENABLE2	Substitutes the result from amount of limitation ÷ 2 into the integral term.
		AWUMD_ENABLE1	Substitutes the amount of limitation into the integral term.
cCLPEN	Phase clipping control	CLPEN_DISABLE	Clipping is disabled.
		CLPEN_ENABLE	Clipping is enabled.
cATANMD	Specifies ATAN calculation control	ATANMD_DISABLE	Calculation is disabled.
		ATANMD_IDQ	Calculates the declination angle on d-q coordinate of current vector.
		ATANMD_EDQ	Calculates the declination angle on d-q coordinate of induced voltage vector.
cVDCSEL	Selects the supply voltage store register	VDCSEL_VDC	Stored in VExVDC.
		VDCSEL_VDCL	Stored in VExVDCL.
cZIEN	Zero current detection	ZIEN_DISABLE	Disable
		ZIEN_ENABLE	Enable
cPVIEN	Phase interpolation	PVIEN_DISABLE	Disable
		PVIEN_ENABLE	Enable

13. 温度測定

13.1 測定方法

Temp0 の AD 値を取得、テーブル値を参照し温度間 1 次方程式により小数点 1 桁まで算出しています。

例)

20°C AD 判定値 = 2146(テーブル値) / 30°C AD 判定値 = 1755(テーブル値) / Temp0 = 1790

2146 – 1755 = 391 (20°C～30°C 傾きは固定値)

2146 – 1790 = 356 (20°C からの差)

補間値 10°C × 356 / 391 = 9.1°C

20°C + 9.1°C = 29.1°C

14. ステータス確認モニター

TeraTerm 上で現在の回転速度などをモニターすることができます。

・TeraTerm 通信設定

115200bps、データ 8bit、ストップビット 1bit、パリティーなし、フロー制御なし

14.1 初回表示内容

リセット解除後に 1 回下記の初期ステータスを送信します。

Initial Status
Control ch = CH0
Carrier Frequency = xxxxx Hz
Dead Time = x.x μs
Gate Active = H/H または L/L
Position Detect = 3Shunt
VDC Voltage = xx.x V
Inverter Temp = xx.x degree
UVW 0A Voltage = x.xx V/x.xx V/x.xx V
DAC Data = A:xx B:xx C:xx D:xx (DAC モード 0 を表示)
Internal AMP = Yes または No
Direction = CW または CCW
Modulation = 2Phase または 3Phase
Key Operation = Volume SW

14.2 回転中の表示内容

モーター回転指示が発生後に現在の回転数を表示します。

回転停止時、EMG 検出時含めた表示は下記のとおりです。

回転中(1 秒毎) : xxHz
回転中⇒停止 : 0Hz
回転中⇒EMG 検出 : EMG_S

14.3 DAC モード切替表示内容

DAC モード切り替わった際、下記 DAC モードを表示します。

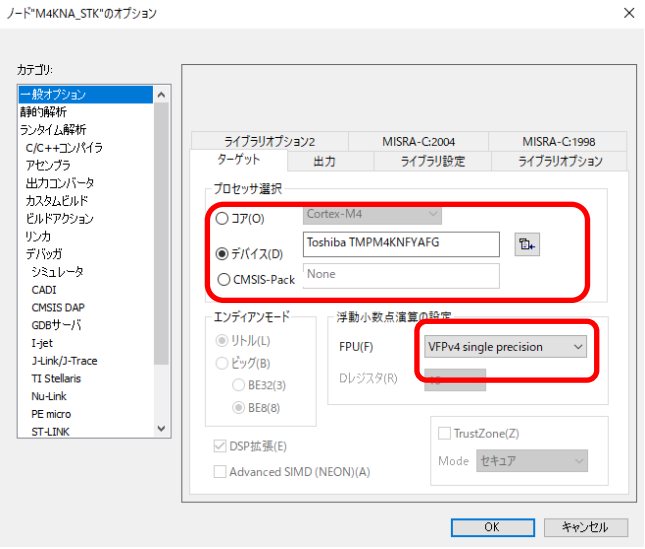
モード 0 : DAC Data = A:TMPREG0 B:TMPREG1 C:TMPREG2 D:theta.half[1]
モード 1 : DAC Data = A:Id_ref B:Id C:Iq_ref D:Iq
モード 2 : DAC Data = A:omega_com.half[1] B: omega.half[1] C: omega_def D:Iq_ref
モード 3 : DAC Data = A:TMPREG0 B:Iq_ref C:Id_ref D:omega.half[1]

15. FAQ

15.1 EWARM のプロジェクト設定が消えたとき

バージョンが異なる IDE でプロジェクトを開いたとき、設定が消えることがあります。
下記に従い再設定してください。

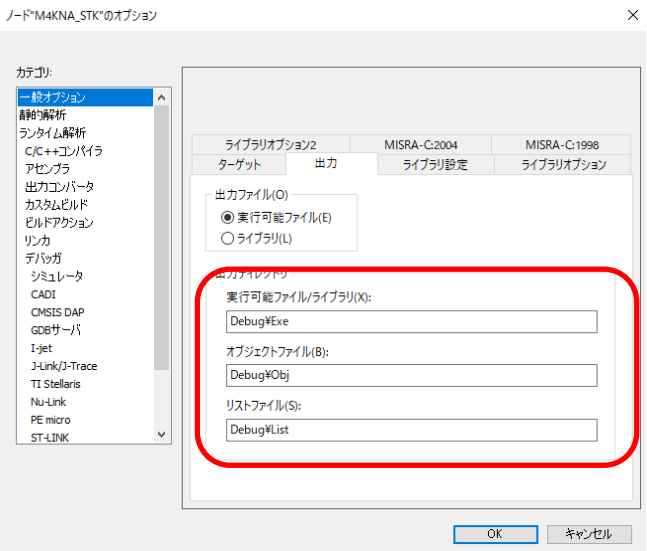
1．プロジェクトメニュー ＞ オプション ＞ 一般オプション
<ターゲット>タブ



TMPM4KNFYAFG を選択してください

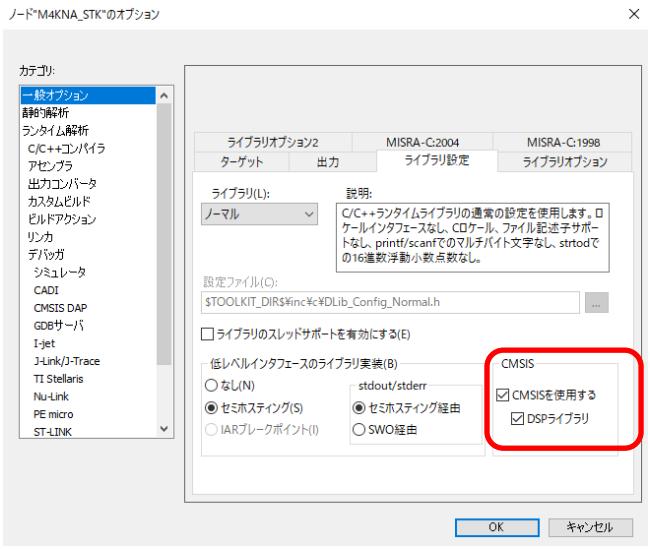
FPU の選択が可能な場合は
VFPv4 を選択してください。

<出力>タブ



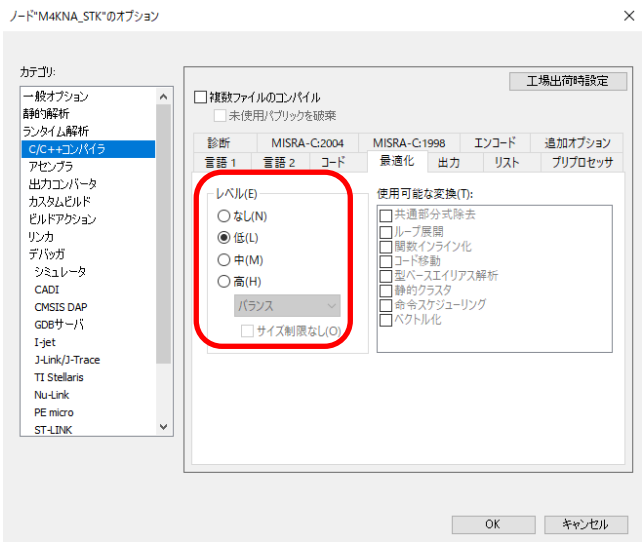
出力ディレクトリーの設定を行ってください。
デフォルトでも問題ありませんが、他プロジェクトと異なるディレクトリーに設定することをお勧めします。

<ライブラリ設定>タブ



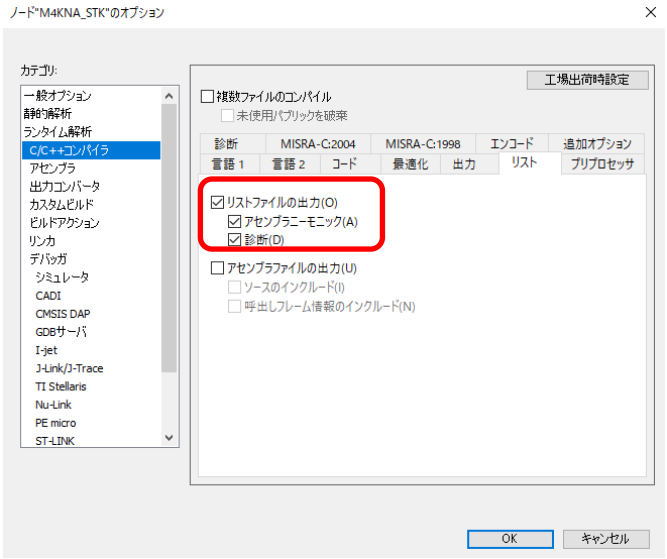
"CMCIS を使用する"と"DSP ライブラリ"
にチェックを入れてください。

2．プロジェクトメニュー ＞ オプション ＞ C/C++コンパイラー
<最適化>タブ



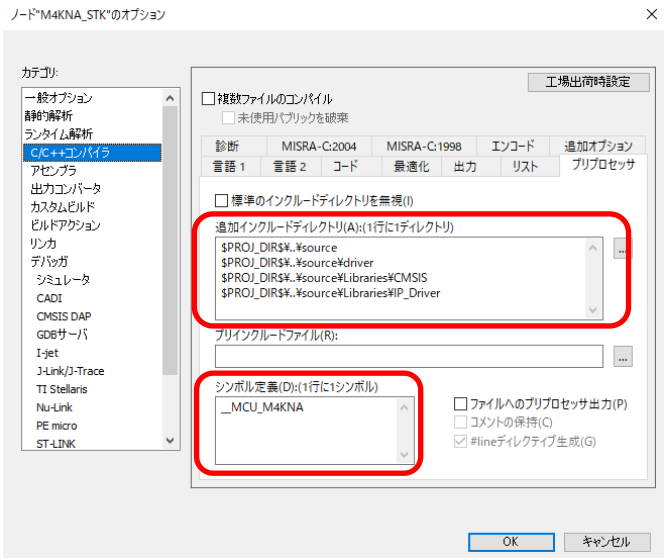
最適化レベルの設定を行ってください。

<リスト>タブ



リストファイルの出力にチェックを入れてください。
チェックを入れなくても、コンパイルはできますが、リストファイルを作成しておくと評価時などに便利です。

<プリプロセッサ>タブ

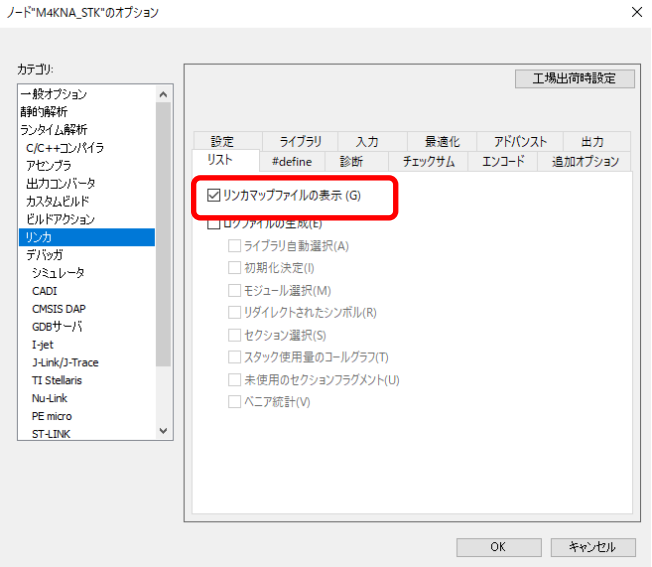


追加インクルードディレクトリーとシンボル定義に下記設定を追加してください。

【TMPM4KNA の場合】
・追加インクルードディレクトリー
$PROJ_DIR$¥..¥source
$PROJ_DIR$¥..¥source¥driver
$PROJ_DIR$¥..¥source¥Libraries¥CMSIS
$PROJ_DIR$¥..¥source¥Libraries¥IP_Driver
・シンボル定義
__MCU_M4KNA

3．プロジェクトメニュー ＞ オプション ＞ リンカ

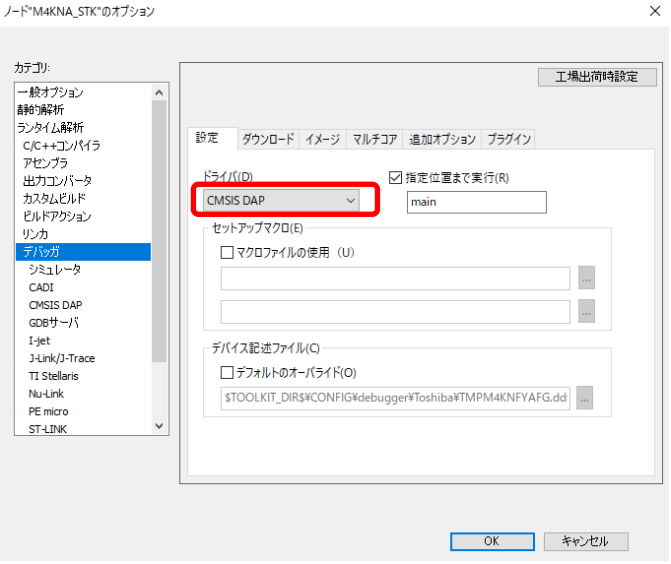
<リスト>タブ



リンカマップファイルの表示にチェックを入れてください。
チェックを入れなくても、コンパイルはできますが、マップファイルを作成しておくと評価時などに便利です。

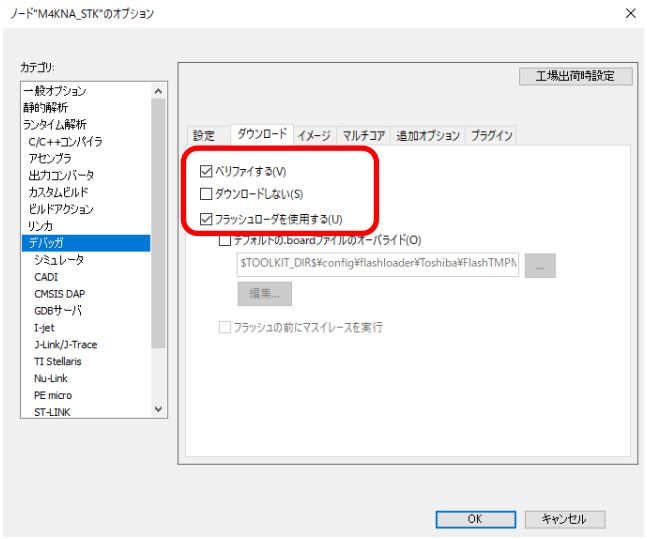
4．プロジェクトメニュー ＞ オプション ＞ デバッガー

<設定>タブ



使用するツールを選択してください。
オンボード ICE が搭載されているボードは「CMSIS-DAP」を選択してください。

<ダウンロード>タブ



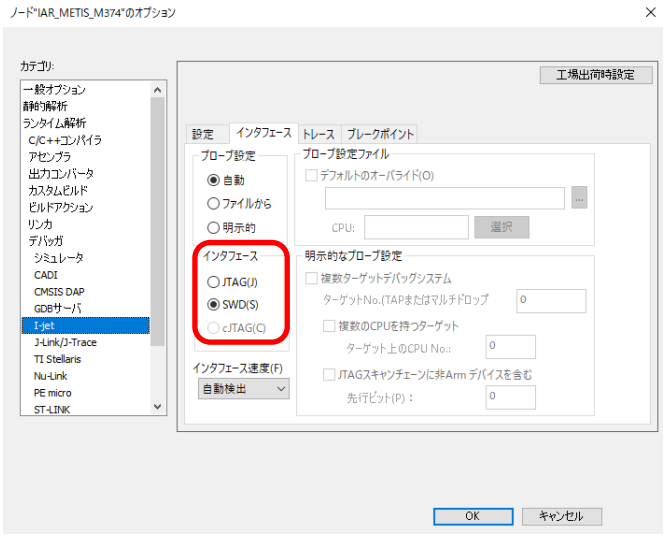
ベリファイするにチェックを入れてください。

フラッシュローダーを使用するにチェックを入れてください。

5．プロジェクトメニュー ＞ オプション ＞ CMSIS DAP or I-jet/JTAGjet

・I-jet の場合

<インターフェース>タブ



SWD を選択してください。

16. 付録

16.1 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。

16.2 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）

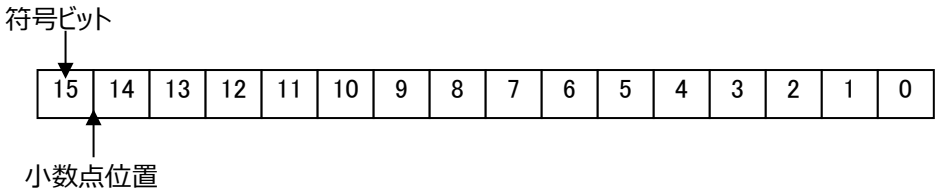


図 3 16 ビットデータの例

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。

16.3 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 4 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

表 5 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

16.4 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

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。

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

17. 改定履歴

日付	Rev.	変更内容
2021/11/08	1.0	初版

製品取り扱い上のお願い

株式会社東芝およびその子会社ならびに関係会社を以下「当社」といいます。
本資料に掲載されているハードウエア、ソフトウエアおよびシステムを以下「本製品」といいます。

- 本製品に関する情報等、本資料の掲載内容は、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。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。また、文書による当社の事前の承諾を得て本資料を転載複製する場合でも、記載内容に一切変更を加えたり、削除したりしないでください。
- 当社は品質、信頼性の向上に努めていますが、半導体･ストレージ製品は一般に誤作動または故障する場合があります。本製品をご使用頂く場合は、本製品の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、お客様のハードウエア・ソフトウエア・システムに必要な安全設計を行うことをお願いします。なお、設計および使用に際しては、本製品に関する最新の情報（本資料、仕様書、データシート、アプリケーションノート、半導体信頼性ハンドブックなど）および本製品が使用される機器の取扱説明書、操作説明書などをご確認の上、これに従ってください。また、上記資料などに記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を使用する場合は、お客様の製品単独および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ください。
- 本製品は、特別に高い品質・信頼性が要求され、またはその故障や誤作動が生命・身体に危害を及ぼす恐れ、膨大な財産損害を引き起こす恐れ、もしくは社会に深刻な影響を及ぼす恐れのある機器（以下“特定用途”という）に使用されることは意図されていませんし、保証もされていません。特定用途には原子力関連機器、航空・宇宙機器、医療機器（ヘルスケア除く）、車載・輸送機器、列車・船舶機器、交通信号機器、燃焼・爆発制御機器、各種安全関連機器、昇降機器、発電関連機器などが含まれますが、本資料に個別に記載する用途は除きます。特定用途に使用された場合には、当社は一切の責任を負いません。なお、詳細は当社営業窓口まで、または当社 Web サイトのお問い合わせフォームからお問い合わせください。
- 本製品を分解、解析、リバースエンジニアリング、改造、改変、翻案、複製等しないでください。
- 本製品を、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することはできません。
- 本資料に掲載してある技術情報は、製品の代表的動作・応用を説明するためのもので、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または実施権の許諾を行うものではありません。
- 別途、書面による契約またはお客様と当社が合意した仕様書がない限り、当社は、本製品および技術情報に関して、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。）をしておりません。
- 本製品、または本資料に掲載されている技術情報を、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ないでください。また、輸出に際しては、「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、それらの定めるところにより必要な手続を行ってください。
- 本製品の RoHS 適合性など、詳細につきましては製品個別に必ず当社営業窓口までお問い合わせください。本製品のご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用ある環境関連法令を十分調査の上、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は一切の責任を負いかねます。