DC300 V 入力
ブラシレス DC モーターセンサーレス駆動回路
(TPD4164K 適用)

ソフトウェアガイド

RD179c-SWGUIDE-01

東芝デバイス＆ストレージ株式会社

目次

1. 概要 6
1.1 仕様概要 6
1.2 処理概要 7
2. ソースファイル構成 9
3. 評価環境について 10
3.1 開発ツール 10
3.2 プロジェクトの立ち上げ方 10
3.3 DAC 出力 11
3.4 ユーザーインターフェースについて 12
4. モジュール構成 14
5. マイコンハードウェアリソースのアサイン 15
5.1 周辺インターフェース 15
5.2 割り込み 15
5.3 GPIO 16
6. ジェネラルフロー 17
6.1 メインルーチン(main) 17
6.2 メインループ(main_loop) 18
6.3 割り込み 19
6.3.1 VE ベクトル処理(INT_VectorControl_byVE) 21
6.3.2 PMD 割り込み処理 (INT_PMD1) 22

7. 状態遷移 23

7.1 センサーレス 23

8. ユーザーアプリケーション 24

8.1 ユーザー制御 24
8.1.1 タイマーカウント制御(timer_conut) 25
8.1.2 AD 値取得(soft_adc_getdata) 25
8.1.3 キー制御(Uikeyscan) 25
8.1.4 ユーザー設定(user_setting) 25
8.1.5 LED 表示制御(led_display) 25
8.1.6 通信制御(uart_control) 25

9. 機能説明 26

9.1 制御コマンド 26
9.1.1 制御方法(usr.com_user) 26
9.1.2 制御目標速度 26
9.1.3 始動電流 26

9.2 駆動コマンド 26
9.2.1 駆動方法(drv.command) 26
9.2.2 ベクトル制御コマンド(drv.vector_cmd) 26

9.3 駆動状態 27
9.3.1 エラー状態(drv.state) 27

9.4 モーター制御構造体 29
9.4.1 変数一覧 29

9.5 関数詳細 31
9.5.1 ADC 初期設定(init_ADCen) 31
9.5.2 PMD 初期設定(init_PMDen) 31
9.5.3 VE 初期設定(init_VEen) 31
9.5.4 モーター制御初期設定(B_Motor_Init) 32
9.5.5 DAC 制御初期設定(init_Dac) 33
9.5.6 周期タイマー初期設定(init_Timer_interval4kHz) 33
9.5.7 ユーザー制御初期設定(init_user_control) 34
9.5.8 ユーザー制御(uart_control) 34
9.5.9 ユーザーモーター制御(B_User_MotorControl) 34

9.6 モーター制御関数 35
9.6.1 状態遷移処理関数(C_Control_Ref_Model) 35
9.6.2 モーター制御共通処理関数(C_Common) 36
9.6.3 停止状態関数(C_Stage_Stop) 37
9.6.4 ブートストラップ状態関数(C_Stage_Bootstrap) 37
9.6.5 位置決め状態関数(C_Stage_Initposition) 38
9.6.6 強制転流状態関数(C_Stage_Force) 39
9.6.7 強制定常切替状態関数(C_Stage_Change_up) 40
9.6.8 定常状態関数(C_Stage_Steady_A) 41
9.6.9 保護状態関数(C_Stage_Emergency) 41
9.6.10 シフト PWM 制御 (C_ShiftPWM_Control) 42
9.6.11 Vdq 算出 (Cal_Vdq) 43

9.7 モーター駆動関数 44
9.7.1 用語説明 44
9.7.2 位置推定関数(D_Detect_Rotor_Position) 44
9.7.3 速度制御関数(D_Control_Speed) 46
9.7.4 3 シャント電流誤検知検出関数 (D_Check_DetectCurrentError_3shunt) 46
9.7.5 1 シャント電流誤検知検出関数 (D_Check_DetectCurrentError_1shunt) 47

10. 定数定義説明 49

10.1 モータードライバー設定用引数：(D_Para.h) 49
10.1.1 DAC 出力選択 49
10.1.2 引数：一覧 49

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_ch1.h) 49
10.2.1 モーターチャネル別引数：一覧 49
10.2.2 定数設定値と波形の関係 55

10.3 ユーザー制御関連定数 56

11. 制御、データ更新のタイミング 62

11.1 VE 使用によるベクトル制御 62
11.1.1 3 シャント制御 62
11.1.2 1 シャント制御 63

12. ペリフェラルドライバー 64

12.1 IP テーブル 64
12.1.1 データ構造 64

12.2 ベクトルエンジン(VE) 64
12.2.1 関数仕様 64
12.2.2 データ構造 71

12.3 モーター制御回路(PMD) 72
12.3.1 関数仕様 72
12.3.2 データ構造 73

12.4 アナログデジタルコンバーター(ADC) 74
12.4.1 関数仕様 74
12.4.2 データ構造 74
12.4.3 VE 搭載マイコン用定数(mcuip_drv.h) 74

13. 付録 78

13.1 固定小数点処理 78

13.2 正規化(Normalize) 78

13.3 データフォーマット 78

13.4 固定小数点での演算 79

1. 概要

本ドキュメントは DC 300V Input BLDC Motor Sensorless Control Circuit リファレンスデザインのソフトウェア仕様を記述したものである。

1.1 仕様概要

- ◆ 使用マイコン・・・・・・TMPM374FWUG(動作クロック 80 MHz)
- ◆ 使用インテリジェントパワーデバイス・・・・・・TPD4164K
- ◆ 使用モーター：エアコン室内機ファン用ブラシレスモーター（動作電圧 DC280 V）
- ◆ 開発環境・・・・・・EWARM Ver8.50.1.24811　または KEIL Ver5.29.0.0

- ◆ 制御概要
  ・ブラシレス DC 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に対応
  ・評価用 DAC 出力(変数値アナログ出力用)ならびに 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	3 シャント or 1 シャント
位置検出方式	センサーレス
PWM 変調方式	3 相変調 or 2 相変調

1.2 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。

-

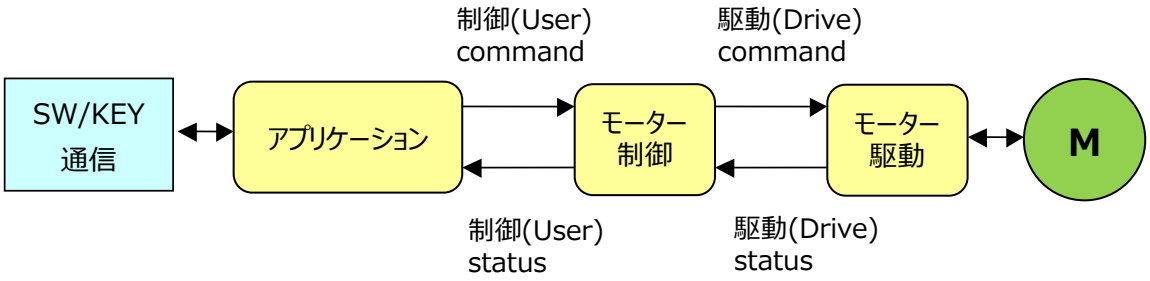


図 1 ベクトル制御ソフトウェアの構造

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。

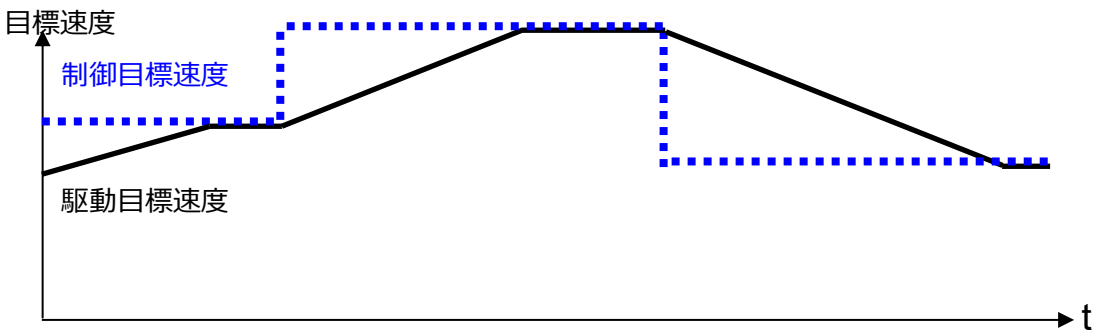


図 2 制御目標速度と駆動目標速度

システムブロック図

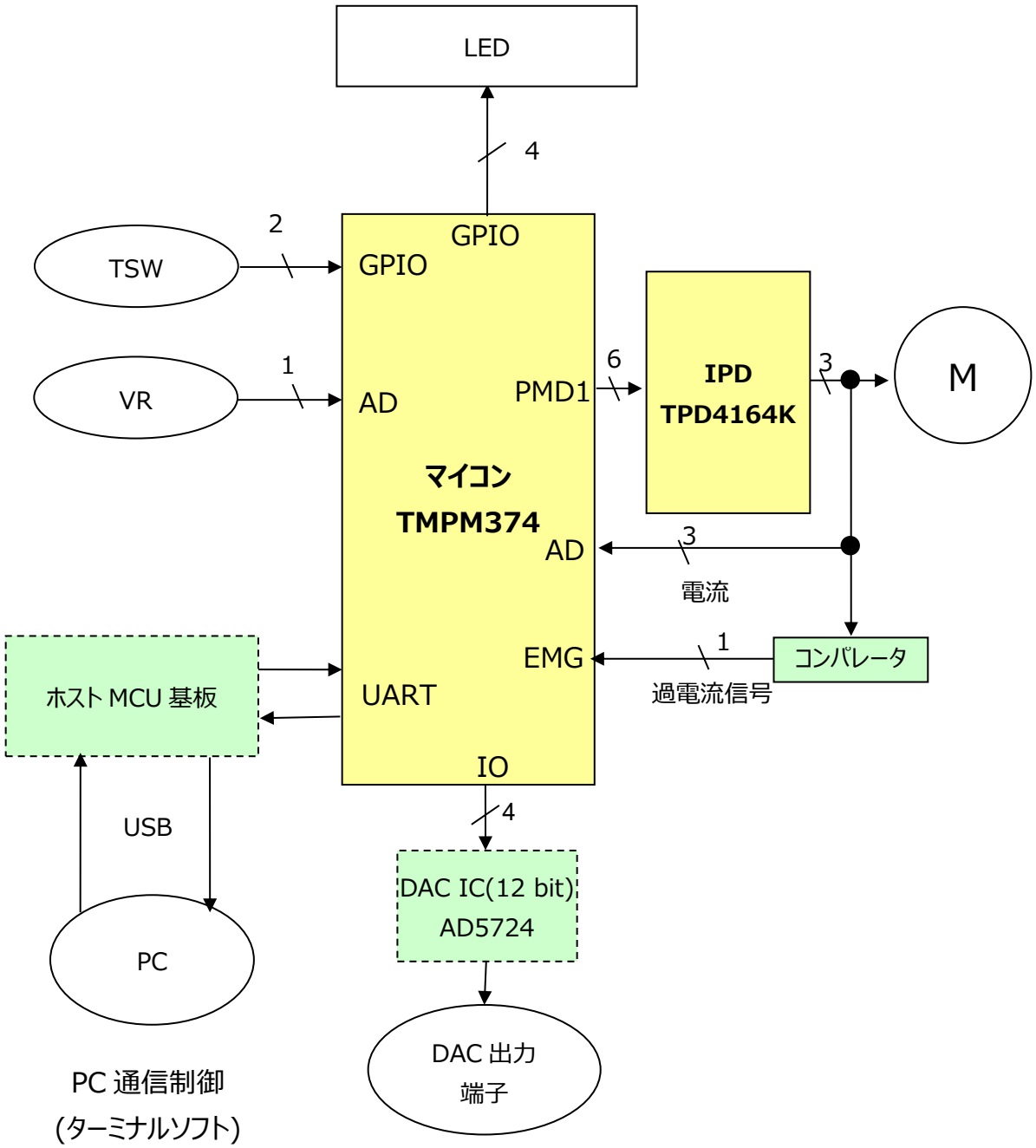


図 3 システムブロック図

2. ソースファイル構成

```
metis_motor_sample_r01
│
└──common        ベクトル制御ドライバー関連のファイルが格納されています。
│   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
│   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
│   sys_macro.h      マクロ定義用ヘッダーファイル
└──M374
    └──source
        │   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
        │   main.c    メインルーチン
        │   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
        └──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   ベクトル制御引数：(チャネル共通)ヘッダーファイル
        │   D_Para_ch1.h ベクトル制御引数：(チャネル 1 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
        │   CheckVdq.h     Vdq 演算用関数ソースファイル
        │   CheckVdqq.h   Vdq 演算用関数ヘッダーファイル
        └──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_TMPM374.c  マイコン初期設定用ソースファイル
            │   │   system_TMPM374.h  マイコン初期設定用ヘッダーファイル
            │   │   TMPM374.h         マイコンレジスター定義ヘッダーファイル
            │   └──startup
            │       ├──arm
            │       │       startup_TMPM374.s      スタートアップアセンブラーソース(arm 用)
            │       └──iar
            │               startup_TMPM374.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。
                tmpm374_adc.c
                tmpm374_adc.h
                tmpm374_gpio.c
                tmpm374_gpio.h
                tmpm374_tmrb.c
                tmpm374_tmrb.h
                tmpm374_uart.c
                tmpm374_uart.h
                tmpm374_wdt.c
                tmpm374_wdt.h
                tx03_common.h
```

3. 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。

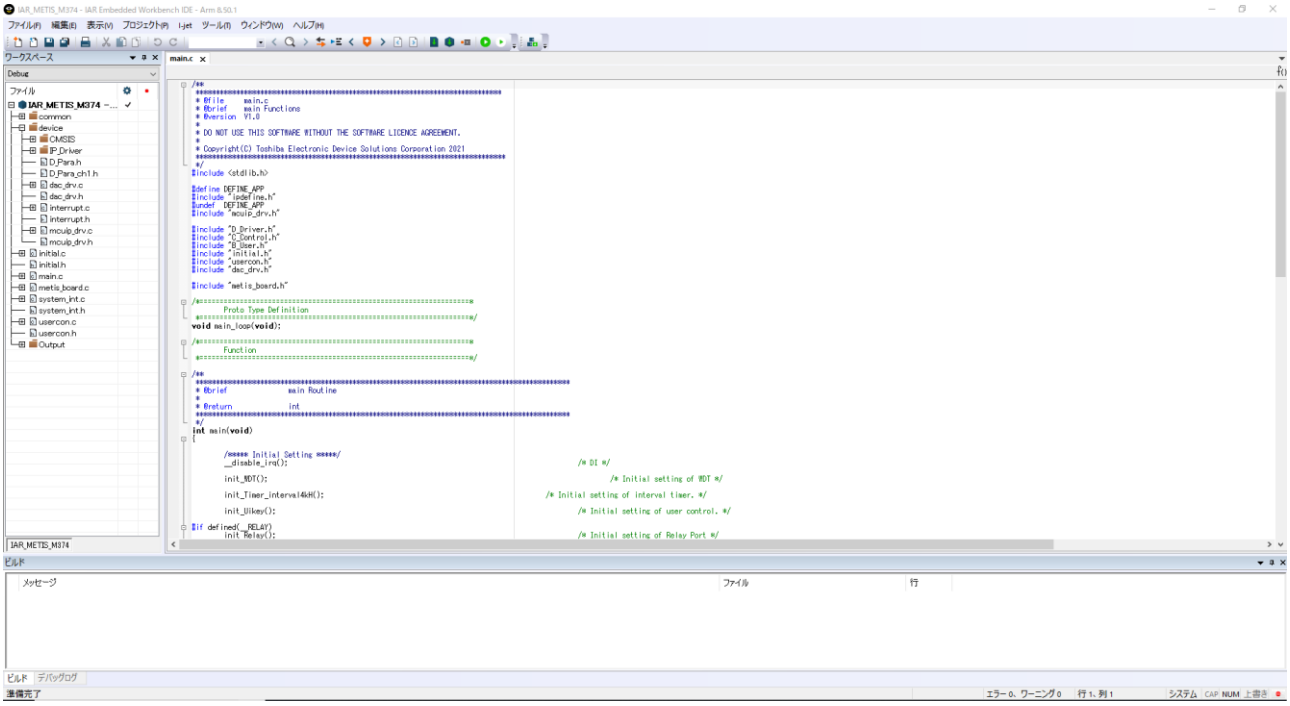
IAR Embeded Workbench for ARM　　8.50.1.24811
KEIL µVision MDK-Lite　　5.29.0.0

3.2 プロジェクトの立ち上げ方

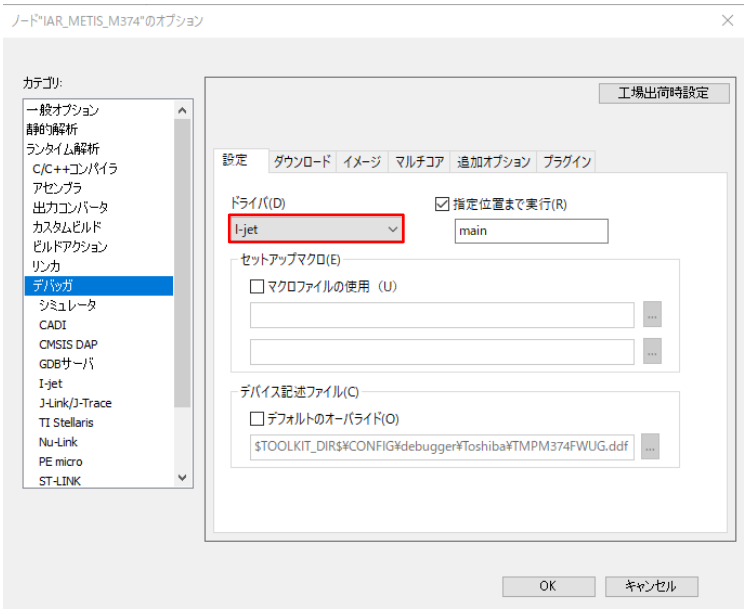
IAR Embeded Workbench を例として説明します。

1. M374¥iar_V8501¥ IAR_METIS_M374.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 IAR_METIS_M374.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 UiOutDataStartByVE()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。

デフォルトでは以下のとおりになっています。
各モードはファイル usercon.c によって切替可能です。
dac_t　dac = {0,1,0,0,0,0 };

«DAC 出力デフォルト変数»
1．　MODE 0
　VoA：U 相電流
　VoB：V 相電流
　VoC：W 相電流

VoD：電気角

2． MODE 1
VoA：Id リファレンス(目標値)
VoB：Id(現在値)
VoC：Iq リファレンス(目標値)
VoD：Iq(現在値)

3． MODE 2
VoA：角速度(目標値)
VoB：角速度(現在値)
VoC：角速度(差)
VoD：Iq(現在値)

4． MODE 3
VoA：U 相電流
VoB：Iq リファレンス(目標値)
VoC：Id リファレンス(目標値)
VoD：角速度(現在値)

5． MODE 4
VoA：Vdc(現在値)
VoB：Vdq(現在値)
VoC：Vd(現在値)
VoD：Vq(現在値)

3.4 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. モーター速度調整
   ボリューム SW(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：cHZ_MIN～60 Hz(電気角)

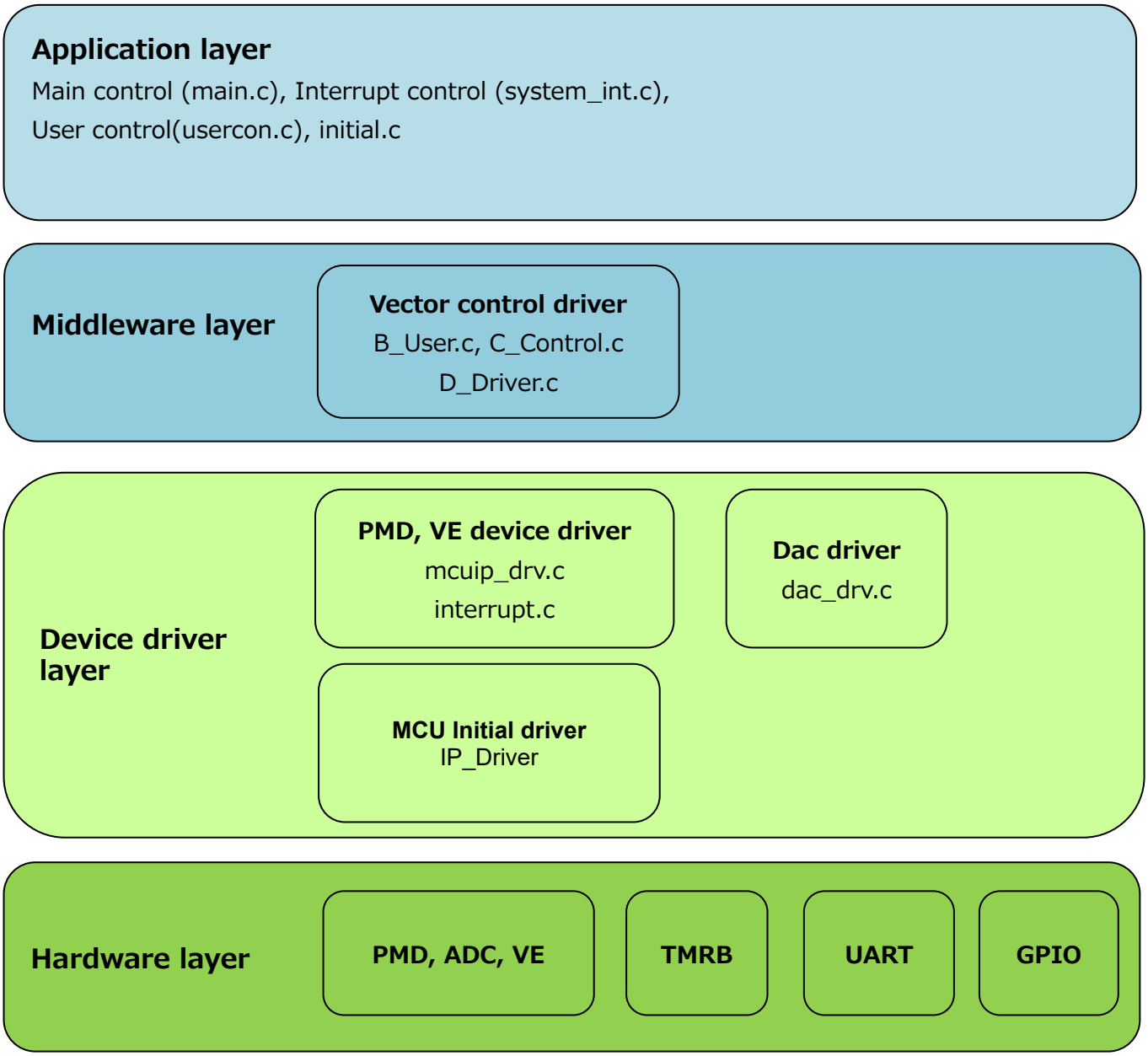
2. モーター回転方向切替
   トグル SW(TSW1)を変更することでモーターの回転方向を切り替えることができます。
   Hi：CW　Low：CCW

3. LED 表示
   LED1～LED3 で通信状態、モーター状態を確認することができます。
   1. 通信モード表示(LED1)

異常なし：消灯
ハード EMG：点灯
ソフト EMG：1 s 点滅
起動時電流検出異常：0.5 s 点滅
Vdc 異常：0.25 s 点滅

2. VE 割り込み処理時間表示(LED2)
   VE 割り込み処理中点灯
3. 通信モード表示(LED3)
   通信確立：点灯
   通信エラー：点滅(250 ms ON/250 ms OFF)

4. モジュール構成



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
TMRB	CH0	4 kHz 周期割り込み	
	CH1	未使用	
	CH2	未使用	
	CH3	未使用	
	CH4	未使用	
	CH5	未使用	
	CH6	未使用	
	CH7	未使用	
SIO/UART	CH0	メイン基板通信制御	UART
	CH1	DAC IC 制御	SIO
	CH2	未使用	
ADC	UNITB	スライドボリューム	制御モード：単体時のみ
		モーターCH1 コイル電流、電源電圧値取得	
PMD	CH1	モーターCH1 制御	
VE	CH1	モーターCH1 制御	

5.2 割り込み

要因名	処理内容	関数名
INTTB00IRQn	4 kHz 周期タイミング作成	INTTB00_IRQHandler
INTVCN1_IRQn	ベクトル制御ソフト処理 for CH1	INTVCN1_IRQHandler
INTTX0_IRQn	メイン基板通信制御(送信)	INTTX0_IRQHandler
INTRX0_IRQn	メイン基板通信制御(受信)	INTRX0_IRQHandler
INTTX1_IRQn	DAC IC 制御	INTTX1_IRQHandler
INTPMD1_IRQn	VE スタート(1 シャントのみ)	INTPMD1_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 */
#define  INT_UART_LEVEL     6  /* UART interrupt */
```

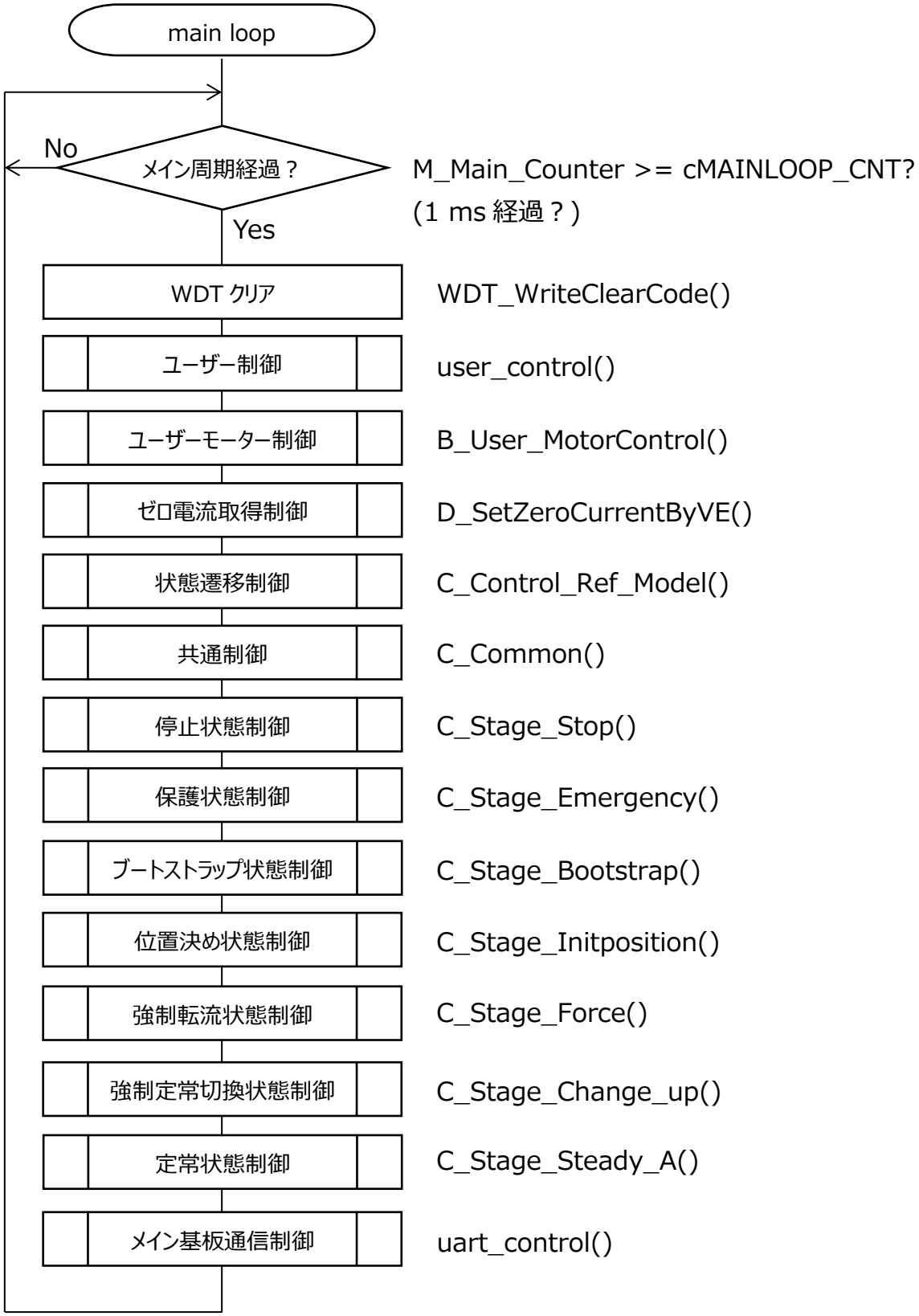
5.3 GPIO

PORT	METIS ピン名称	機能
PG0	UH_1	モーターU
PG1	UL_1	モーターV
PG2	VH_1	モーターW
PG3	VL_1	モーターX
PG4	WH_1	モーターY
PG5	WL_1	モーターZ
PG6(EMG1)	XEMG_1	モーターEMG
PG7	RELAY	突入電流防止
PE0	TXD_UA_MCU	UART_TX
PE1	RXD_UA_MCU	UART_RX
PF2	LED1	EMG 中点灯
PF3	LED2	VE 割り込み処理中点灯
PF4	LED3	制御モード=通信で点灯
PI3(AINB2)	AD_UP_1	U 相電流
PJ0(AINB3)	AD_VP_1	V 相電流
PJ6(AINB9)	AD_WP_1	W 相電流
PJ7(AINB10)	AD_VDC_1	VDC 電圧
PK0	KEY_A	ボリューム抵抗
PK1	KEY_3	スライドスイッチ 3(回転方向)
PE7	KEY_2	スライドスイッチ 2(制御モード)
PA6	XSYNC_SPI	DAC 用シリアル
PA5	SDO_SPI	DAC 用シリアル
PA4	SCLK_SPI	DAC 用シリアル

6. ジェネラルフロー

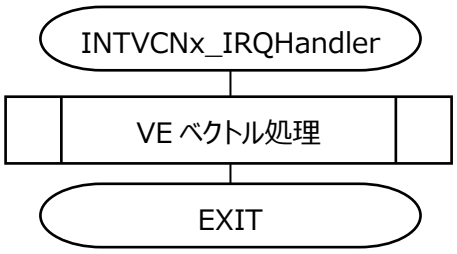
6.1 メインルーチン(main)

6.2 メインループ(main_loop)



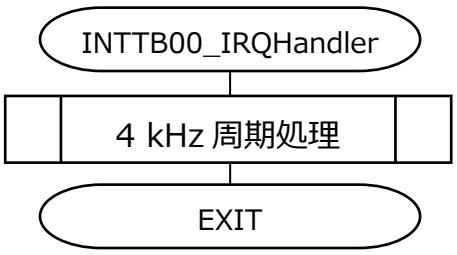
6.3 割り込み

VE スケジュール完了時



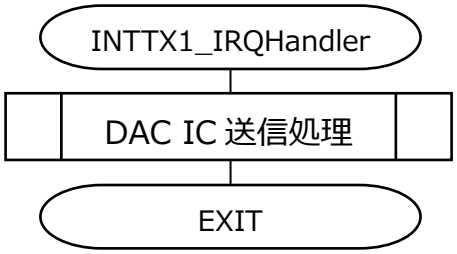
INT_VectorContorlByVE_CH1 ()

4 kHz 周期ごと



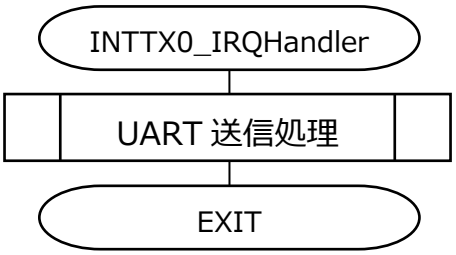
INT_interval_4kHz ()

DAC 通信 8 bit 送信完了時



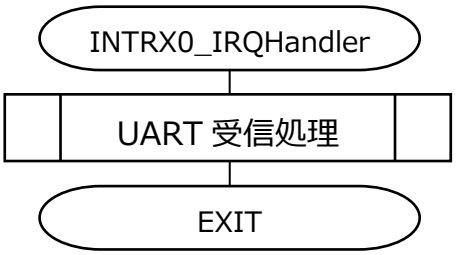
DacWriteContinue ()

メイン基板通信 8 bit 送信完了時



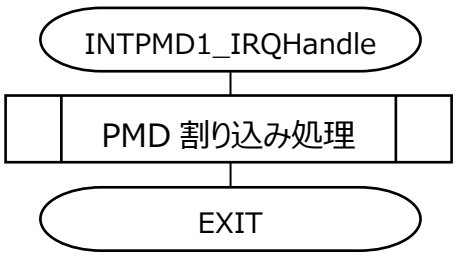
INT_uart0tx ()

メイン基板通信 8 bit 受信完了時



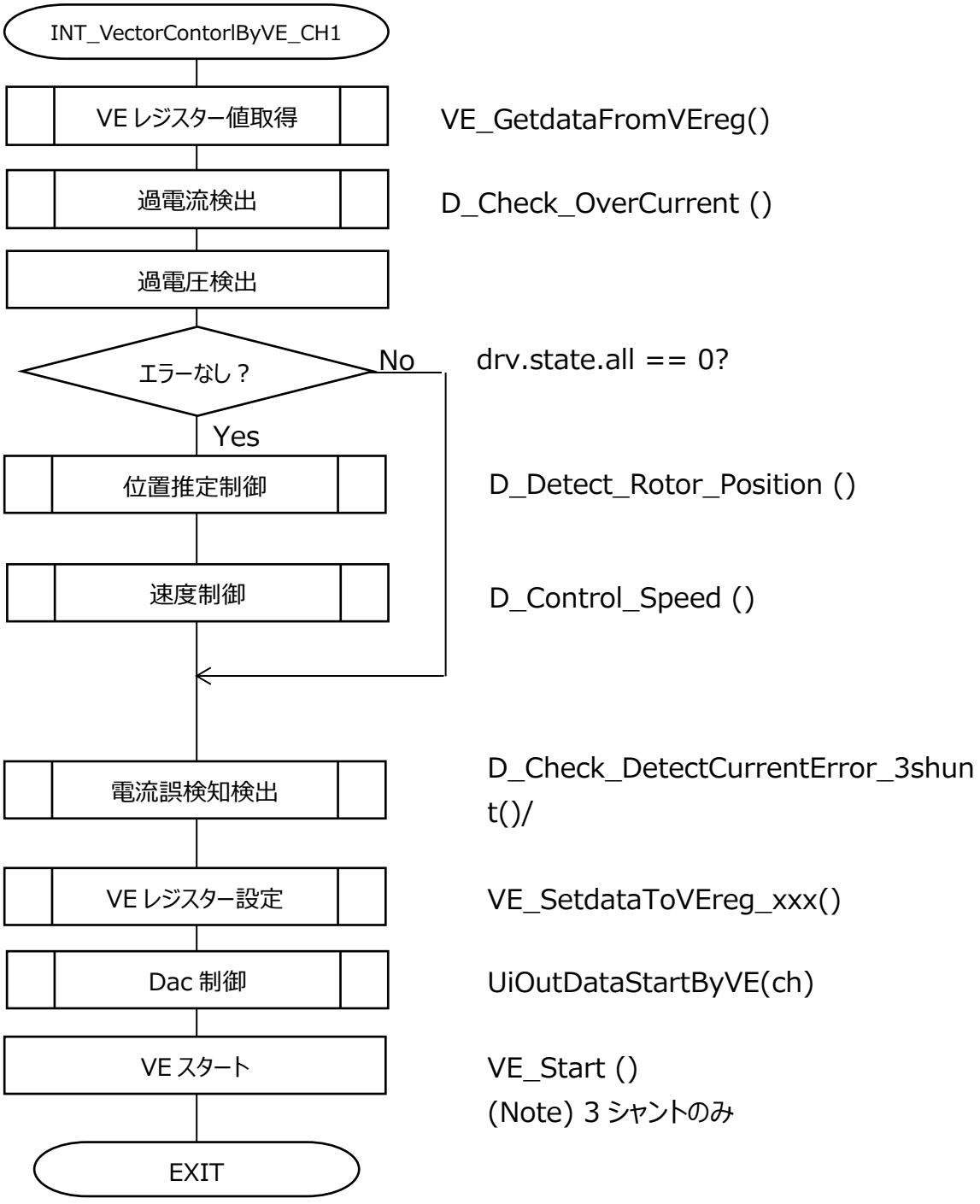
INT_uart0rx ()

PWM 割り込み時（1 シャントのみ）

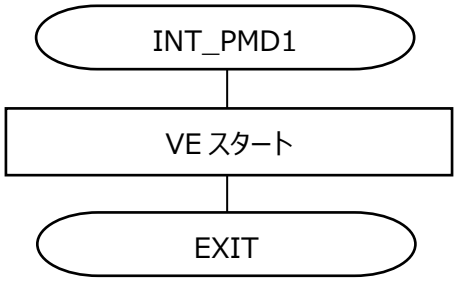


INT_PWM1()

6.3.1 VE ベクトル処理(INT_VectorControl_byVE)



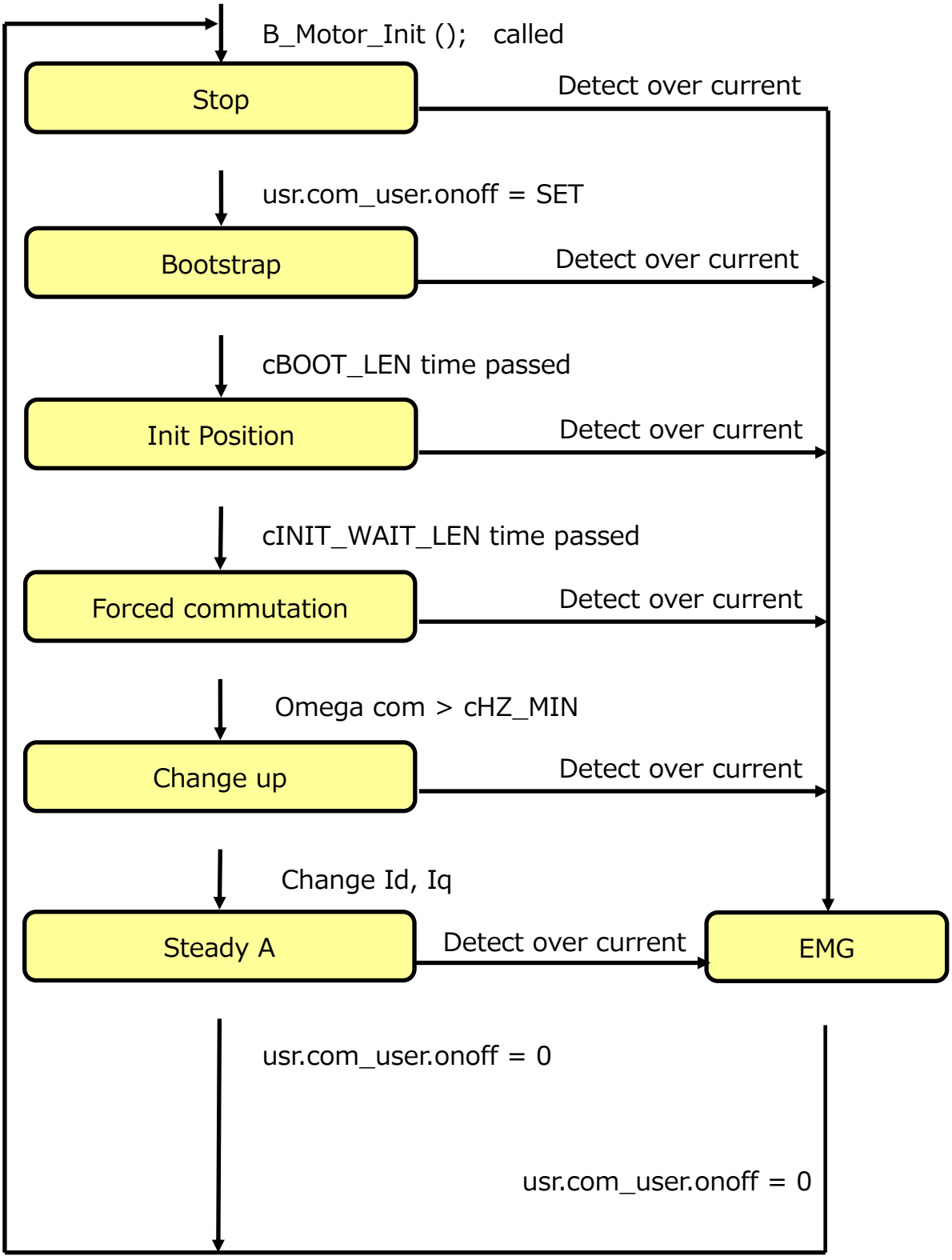
6.3.2 PMD 割り込み処理 (INT_PMD1)



VE_Start ()
(Note)1 シャントのみ

7. 状態遷移

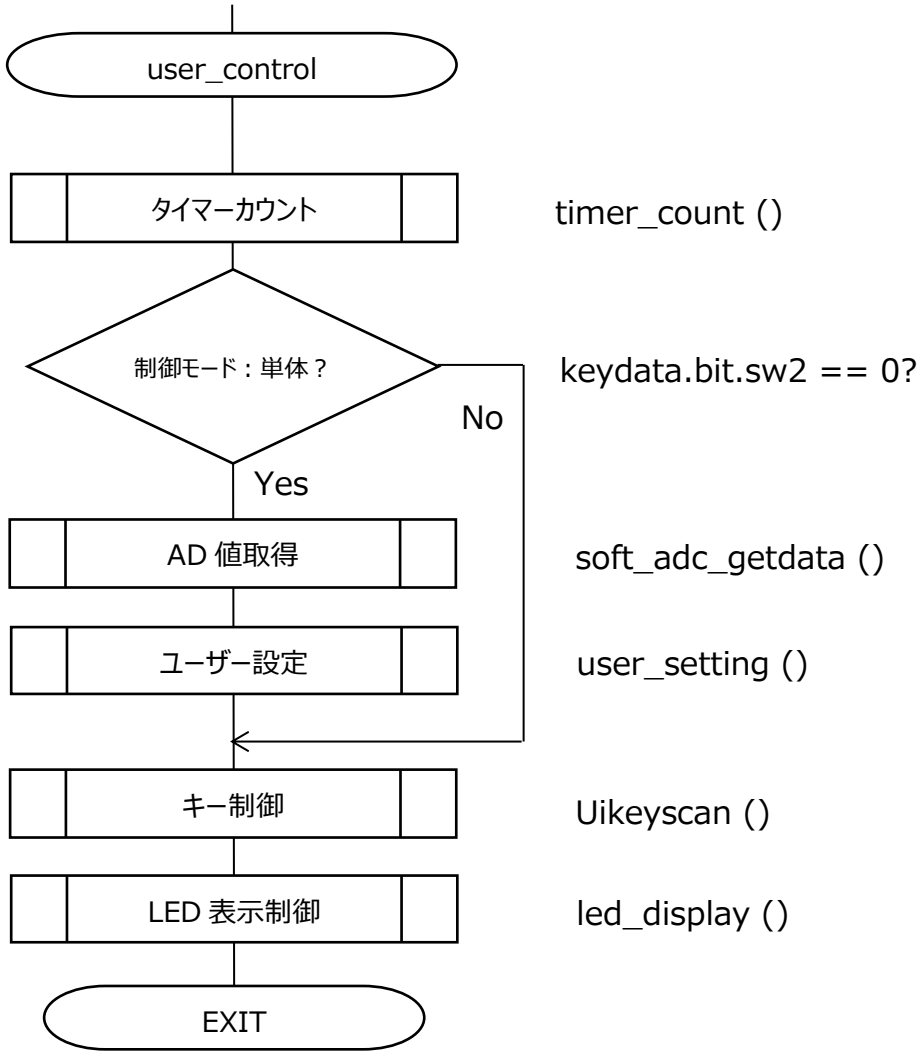
7.1 センサーレス



8. ユーザーアプリケーション

8.1 ユーザー制御

メイン周期(1 ms)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 タイマーカウント制御(timer_conut)

メイン周期ごとに下記のタイマーのカウントを行います。

1. 通信エラーLED 点滅タイマー(1 ms × 250 カウント)
   通信エラー時、250 ms ごとに LED3 の状態を切り替えるためのカウントとなります。
2. EMG 状態点滅タイマー(1 ms × 250 カウント/500 カウント/1000 カウント)
   EMG 状態により LED1 の状態を切り替えるためのカウントとなります。
3. 通信 TimeOut タイマー(1 ms × 2000 カウント)
   通信時、メイン基板との通信状態を確認するためのカウントとなります。

8.1.2 AD 値取得(soft_adc_getdata)

ボリューム SW(VR1)の AD 値(12 bit)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 VR1 の有効値として ad_vr.avedat に格納します。

8.1.3 キー制御(Uikeyscan)

TSW2,3 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、keydata に格納されます。

8.1.4 ユーザー設定(user_setting)

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 8 bit 分解能にし、この値が
   0x10 未満：0(0 Hz)
   0xF0 以上：60(最大速度)
   0x10～0xEF：有効値として速度計算

8.1.5 LED 表示制御(led_display)

EMG の種類、通信状態による LED1～3 表示の出力制御(常時 ON、常時 OFF、点滅)を行います。
また、点滅表示タイミング用の時間管理(0.25 s、0.5 s、1 s)も行います。
各表示については 3.4 ユーザーインターフェースについてを参照してください。

8.1.6 通信制御(uart_control)

メイン基板より送信される回転数・情報ステータスコマンドを受信し、対応する情報をメイン基板へ送信を行います。
通信設定は
115200 bps、データ 8 bit、ストップビット 1 bit、パリティ―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
・変調方式（２相変調、３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 spwm:1; /* 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。

9.1.2 制御目標速度

```
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

```
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。

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;
```

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。

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

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	Edetect
Stop	CLEAR	CLEAR	CLEAR	CLEAR
Bootstrap	CLEAR	CLEAR	CLEAR	CLEAR
InitPosition	CLEAR	CLEAR	CLEAR	CLEAR
Forced	SET	CLEAR	CLEAR	SET
Change_up	SET	SET	CLEAR	SET
Steady	SET	SET	SET	SET
Emergency	CLEAR	CLEAR	CLEAR	CLEAR

1) F_vcomm_theta

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_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

9.3 駆動状態

9.3.1 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_DC:1;    /* 0:normal, 1: Over Vdc         */
    uint16_t   emg_I:1;  /* 0:normal, 1: Current detect error    */
    uint16_t   emg_S:1;  /* 0:normal, 1: Over current(soft)  */
    uint16_t   emg_H:1; /* 0:normal, 1: Over current(hard) */
  } flg;
  uint16_tall;
} state_t;
```

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。

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_tMotor_ch1; /* Motor data for ch1 */

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	シフト切替速度	Q15	1 シャント時のみ
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	---	(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	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	
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	---	c3shunt
boot_type_e	boot_type	起動タイプ	---	cBoot_i cBoot_v

9.5 関数詳細

9.5.1 ADC 初期設定(init_ADCen)

9.5.1.1 構文

void init_ADCen(void)

引数：

なし

戻り値：

なし

9.5.1.2 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

9.5.2 PMD 初期設定(init_PMDen)

9.5.2.1 構文

void init_PMDen(void)

引数：

なし

戻り値：

なし

9.5.2.2 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。

9.5.3 VE 初期設定(init_VEen)

9.5.3.1 構文

void init_VEen(void)

引数：

なし

戻り値：

なし

9.5.3.2 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

9.5.4 モーター制御初期設定(B_Motor_Init)

9.5.4.1 **構文**

void B_Motor_Init(void)

引数：

なし

戻り値：

なし

9.5.4.2 **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	---	
	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 切替速度	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	

9.5.4.3 **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。

9.5.5 DAC 制御初期設定(init_Dac)

9.5.5.1 **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

9.5.5.2 **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

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

9.5.6.1 **構文**

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

9.5.6.2 **処理内容**

4 kHz 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。

9.5.7 ユーザー制御初期設定(init_user_control)

9.5.7.1 **構文**

void init_user_control(void)

引数：

なし

戻り値：

なし

9.5.7.2 **処理内容**

ユーザーなど外部からの指令制御用の初期設定を行います。

・キー入力処理初期設定(init_Uikey)

・アナログ電圧入力処理初期設定(init_soft_adc)

・LED 表示処理初期設定(init_led)

・UART 処理初期設定(init_uart)

9.5.8 ユーザー制御(uart_control)

9.5.8.1 **構文**

void user_control(void)

引数：

なし

戻り値：

なし

9.5.8.2 **処理内容**

ユーザーなど外部からの指令を制御します。

・メイン周期ごとのタイマーカウント

・ボリューム SW(VR1)入力による回転速度制御切換

・トグル SW1 入力によるモーター回転方向切換

・LED 出力

9.5.9 ユーザーモーター制御(B_User_MotorControl)

9.5.9.1 **構文**

void B_User_MotorControl(void)

引数：

なし

戻り値：

なし

9.5.9.2 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	---	

9.5.9.3 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。

目標速度 0 Hz 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。

9.6 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、強制定常切替状態、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)、強制定常切替(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。

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。

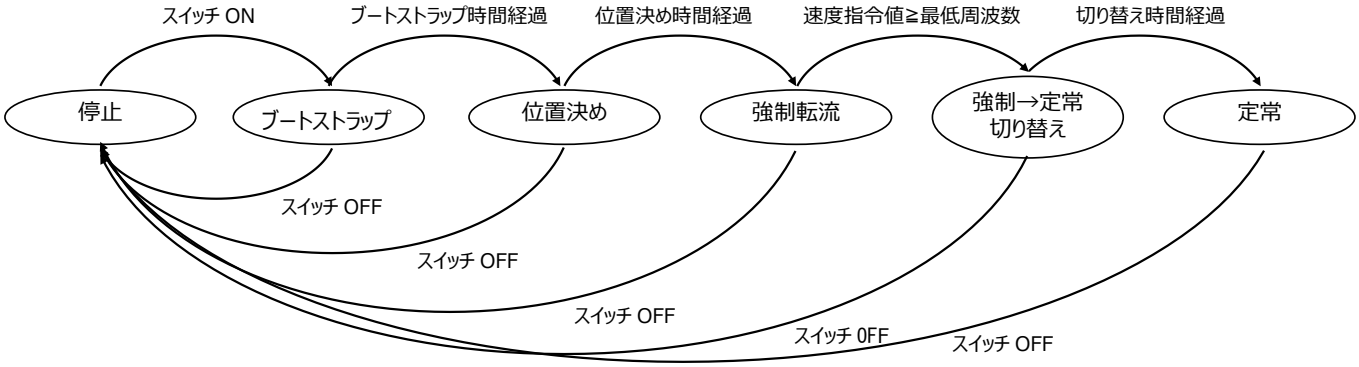


図 4 モーター制御の状態遷移図

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.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	---	

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。
シフト PWM 制御(9.6.10 C_ShiftPWM_Control)、および Vdq の算出(9.6.11 Cal_Vdq)と変調率の算出を行います
変調率は以下の演算式により算出します。
変調率(Vdq_per) = 100.0(%) × Vdq 平均値(Vdq_ave) / Vdc 平均値(Vdc_ave)

9.6.3 停止状態関数(C_Stage_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.vector_cmd	ベクトル制御コマンド	---	(9.2.2 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.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	---	
	drv.vector_cmd	ベクトル制御コマンド	---	(9.2.2 参照)

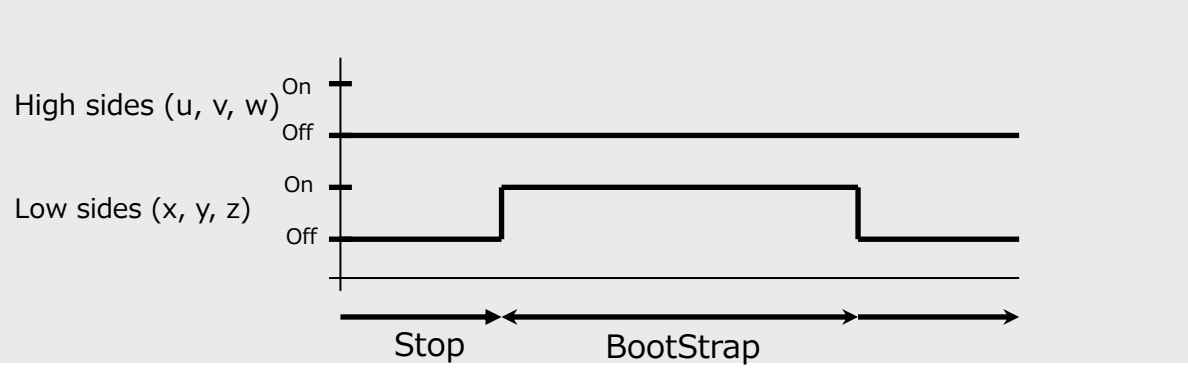
9.6.4.3 **処理内容**

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	---	(9.2.2 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	

9.6.5.3 処理内容

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	---	(9.2.2 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

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($\theta=\omega_{com}t$)。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	---	(9.2.2 参照)
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	

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

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	---	(9.2.2 参照)
	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

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

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 切換速度	Q15	
出力	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 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、表 1 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 <	ON (1)

		para.spwm_threshold	

9.6.11 Vdq 算出 (Cal_Vdq)

9.6.11.1 構文

q15_t Cal_Vdq(q15_t _vd, q15_t _vq)

引数：

q15_t _vd ： d 軸電圧

q15_t _vq ： q 軸電圧

戻り値：

q15_t Vdq ： dq 軸電圧

9.6.11.2 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	
	_vq	q 軸電圧	Q15	
出力	_Vdq	dq 軸電圧	Q15	

9.6.11.3 処理内容

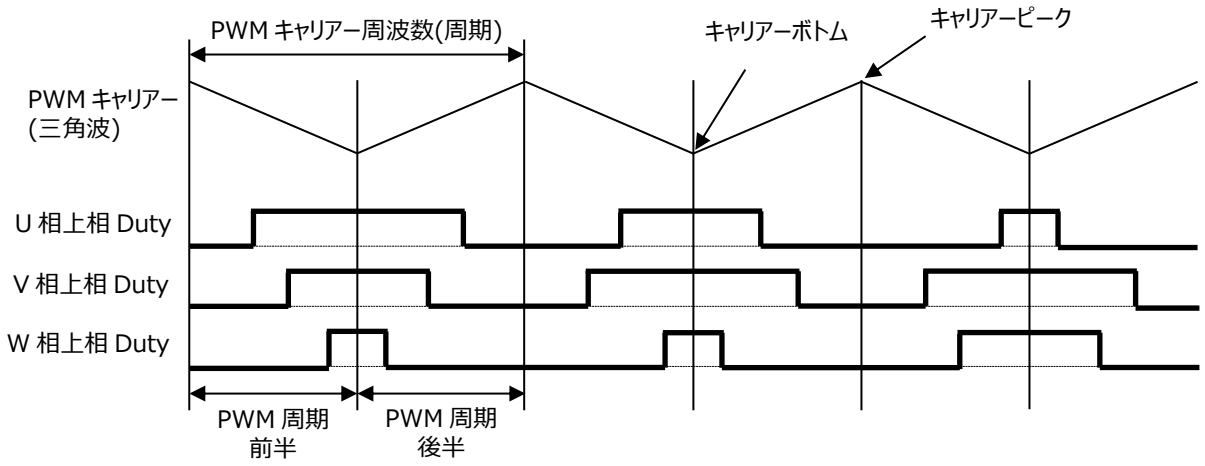
下記演算式により Vdq を算出します。

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

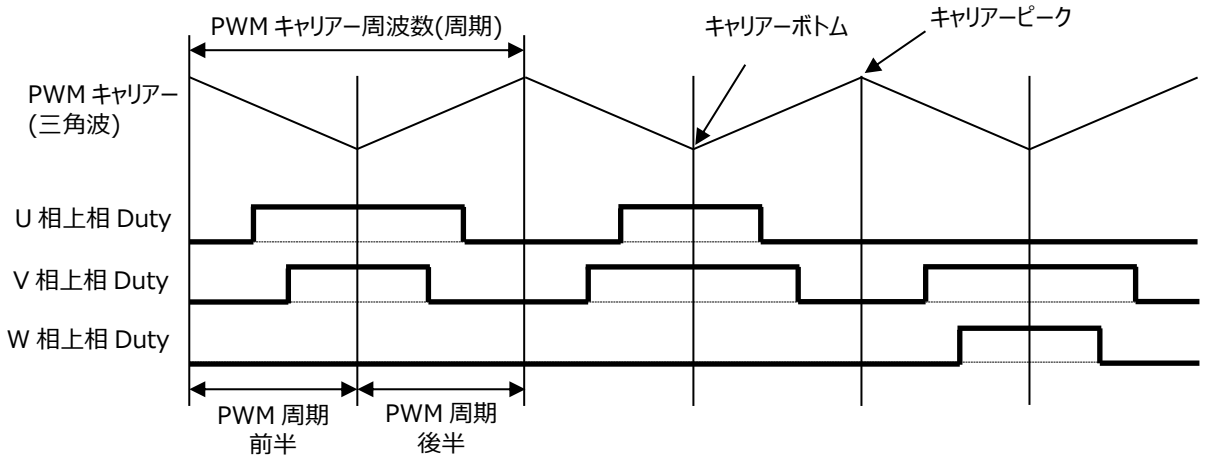
9.7 モーター駆動関数

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 位置推定関数(D_Detect_Rotor_Position)

9.7.2.1 **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

9.7.2.2 **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	

9.7.2.3 **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。

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

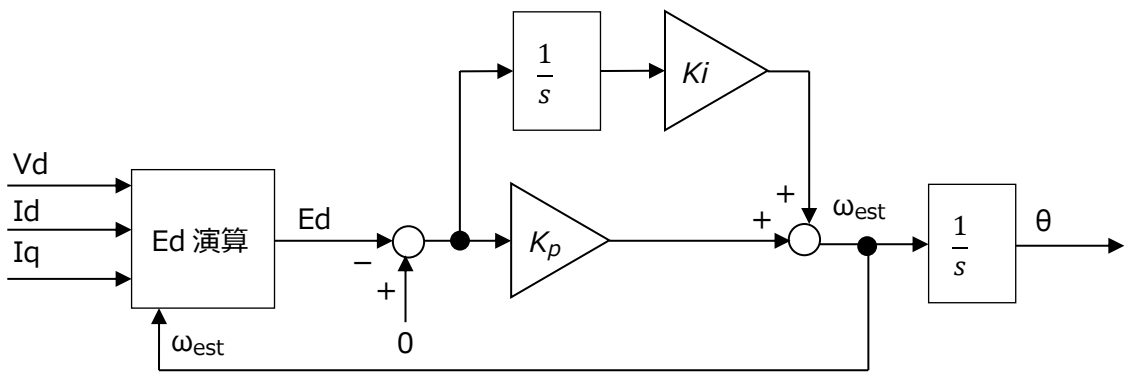


図 5 d 軸誘起電圧 Ed による位置推定ブロック図

9.7.3 速度制御関数(D_Control_Speed)

9.7.3.1 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

9.7.3.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_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	

9.7.3.3 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。

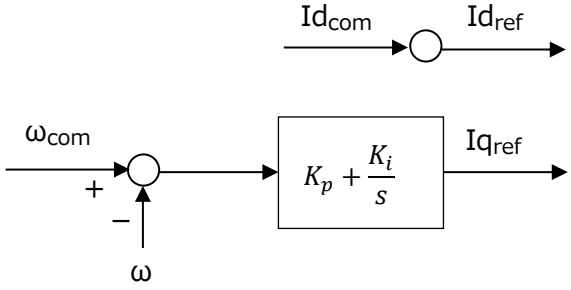


図 6 速度制御ブロック図

9.7.4 3 シャント電流誤検知検出関数 (D_Check_DetectCurrentError_3shunt)

9.7.4.1 構文

int D_Check_DetectCurrentError_3shunt (uint32_t _duty, uint16_t _chkplswidth)

引数：

_duty　　Duty の中間値

_chkplswidth パルスチェック幅

戻り値：

電流検出状態

9.7.4.2 変数

方向	名前	意味	Q フォーマット	備考
入力	_duty	中間の PWM Duty 比	Q15	0.0 to 1.0
	_chkplswidth	パルスチェック幅	---	
出力	---	電流検出状態	---	0:検出可能 1:検出不能

9.7.4.3 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。

Duty の中間値が設定値より大きくなることを検出します。

この関数で電流が検出できない Duty 幅であることを確認し、電流検出処理で検出値を使用しないで、前回の値を使用することで、電流誤検知による制御乱れを無くします。

PWM 最大相の電流検出 AD 変換値は、演算には使用していないため、PWM 中間相の電流検出値が設定値より大きくなると検出 NG と判定します。

9.7.5 1 シャント電流誤検知検出関数（D_Check_DetectCurrentError_1shunt)

9.7.5.1 構文

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

引数：

_pwma　　U 相 PWM Duty 比

_pwmb　　V 相 PWM Duty 比

_pwmc　　W 相 PWM Duty 比

_chkplswidth パルスチェック幅

_sfhpwm　シフト PWM モード状態

_modu　　変調方式

_sector　　セクター情報

_mdprd　　PWM 周期

戻り値：

電流検出状態

9.7.5.2 変数

方向	名前	意味	Q フォーマット	備考
入力	_pwma	U 相 PWM Duty 比	Q15	0.0 to 1.0
	_pwmb	V 相 PWM Duty 比	Q15	0.0 to 1.0
	_pwmc	W 相 PWM Duty 比	Q15	0.0 to 1.0
	_chkplswidth	パルスチェック幅	---	
	_sfhpwm	シフト PWM モード状態	---	
	_modu	変調方式駆動コマンド	---	2 相または 3 相変調
	_sector	セクター情報	---	
	_mdprd	PWM 周期	---	
出力	---	電流検出状態	---	0:検出可能 1:検出不能

9.7.5.3 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。
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。
この関数で電流が検出できない Duty 幅であることを確認し、電流検出処理で検出値を使用しないで、前回の値を使用することで、電流誤検知による制御乱れを無くします。
以下のいずれかの場合に検出 NG と判定します。

- PWM Duty 幅の最小値が設定値より小さい場合
- 2 つの PWM の Duty 幅差の最小値が設定値より小さい場合

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。

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 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。

10.1.2 引数：一覧

引数：名	説明
cMAINLOOP_PRD	メイン周期設定
	単位[s] 分解能 4 kHz
	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。

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_ch1.h)

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 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
	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/ 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。
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(3 シャント用)
	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(3 シャント用)
	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(3 シャント用)
	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[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。 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]
	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。
cGOUP_DELAY_LEN	チェンジアップ後待ち時間
	単位[s] 分解能:1 ms
	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 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。

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 10.2.2.2 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 10.2.2.2 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
__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

10.2.2 定数設定値と波形の関係

10.2.2.1 電流型起動時の定数値

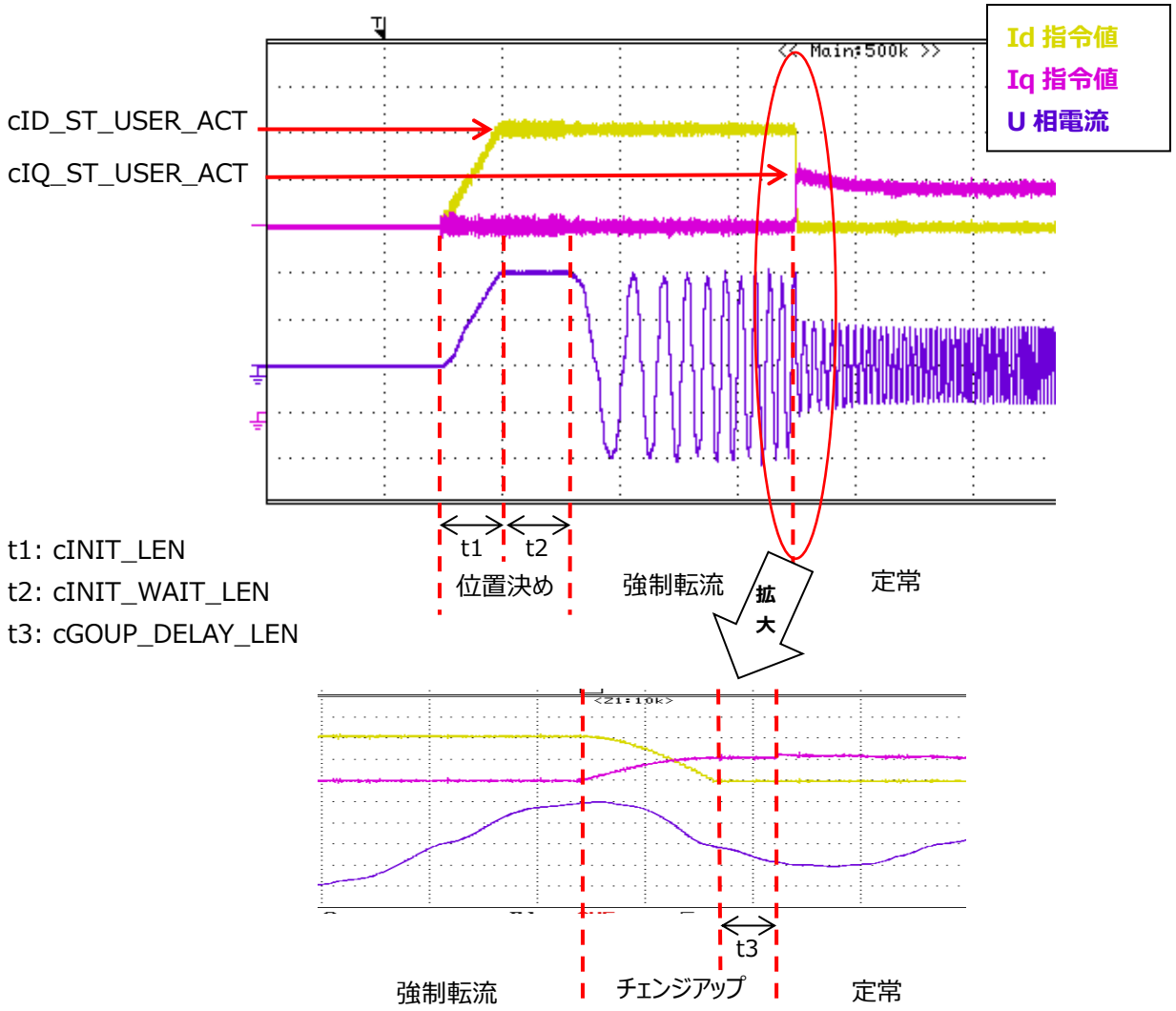


図 7 起動電流波形（電気角 0°に位置決め）

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。

10.2.2.2 **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。

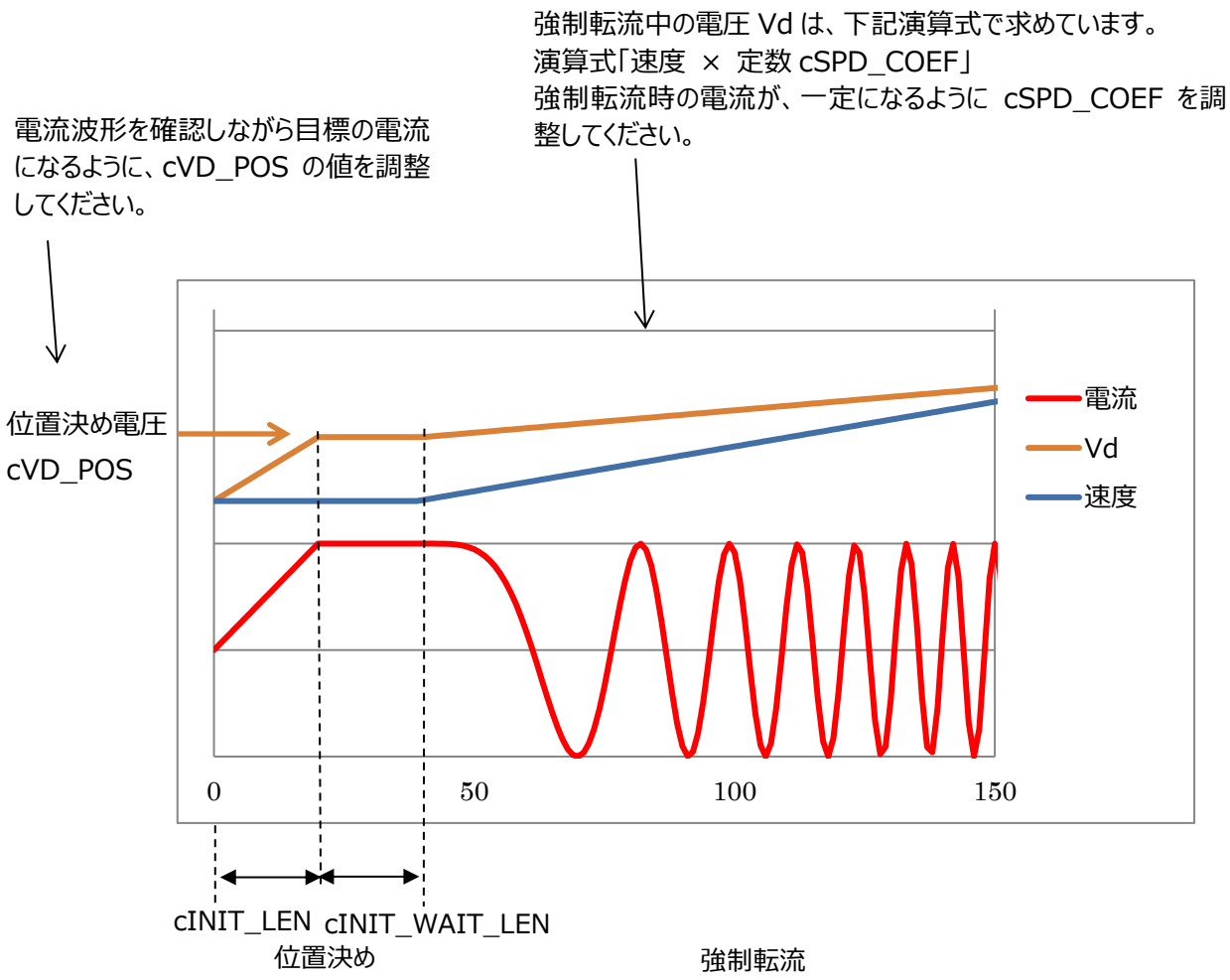


図 8 電圧駆動時の引数：調整方法

10.3 ユーザー制御関連定数

usercon.c

```
/* Timer setting */
#define cEMG_S        (1)    /* LED1 blinking request(over detect(soft)) */
 ソフト EMG 点滅要求
#define cEMG_I    (1)    /* LED1 blinking request(curret detect error) */
 電流検出異常点滅要求
#define cEMG_DC       (1)    /* LED1 blinking request(over vdc) */
 Vdc 異常点滅要求
#define c2000MS_TIMER(2000)   /* [ms] (4 kHz * 4) * 2000 */
 通信判定 2s タイマー
```

```
/* Key settting */
#define S_SW2 TSB_PE_DATA_PE7   /* SW2 or 9(KEY2) */
        SW2 ポート：PE7
#define S_SW3 TSB_PK_DATA_PK1   /* GPIO_15 */
        SW3 ポート：PK1
#define cKEY_CHATA_CNT  (20)      /* [cnt] chattering counter for KEY SW */
 SW 用チャタリング除去カウンター数：20 回連続一致

/* Soft ADC Setting */
#define cADUNIT_USR    TSB_ADB    /* User ad data ADCUnit */
        VR1 入力用 ADC ユニット：ユニット B
#define cADTRG_USR     ADC_TRG_SW
#define cADCH_VR     ADC_AIN11 /* ADC Channel for VR */
        VR1 入力用 AD チャネル：CH11
#define cADREG_VR      ADC_REG5  /* Result register No for VR */
        VR1 入力用 AD 値 格納レジスター：REG5
#define cADAVECNT      (10)      /* ADC average count */
        AD 値確定用平均カウント値(10 回取得で確定)

/* Led setting */
#define LED_EMG      TSB_PF_DATA_PF2   /* LED1 */
        EMG LED ポート：PA0
#define LED_UART_ERR  TSB_PF_DATA_PF4   /* LED3 */
        UART 通信 LED ポート：PF4
#define   cFLASH_TYPE1_CYCLE  (1)       /* [s] led flash cycle type1 */
        LED 点滅時間 1：1s
#define   cFLASH_TYPE2_CYCLE  (0.5)     /* [s] led flash cycle type2 */
        LED 点滅時間 2：0.5s
#define   cFLASH_TYPE3_CYCLE  (0.25)       /* [s] led flash cycle type3 */
        LED 点滅時間 3：0.25s
#define   cLED_ON       (1)       /* LED ON level */
        LED レベル：ON
#define   cLED_OFF      (0)       /* LED OFF level */
        LED レベル：OFF

/* User Setting */
#define   cROTATION_CW      (1)       /* Direction: plus */
        回転方向：CW
#define   cROTATION_CCW (0)       /* Direction: minus */
        回転方向：CCW
#define   cCONTROL_SINGLE  (0)       /* UART control: single */
```

通信制御モード：単体
#define cCONTROL_UART (1) /* UART control: uart */
通信制御モード：通信

/* Speed Control Setting */
#define cAD_MIN (0x10) /* motor speed ADC min value */
VR1 AD 値：最小速度
#define cAD_MAX (0xF0) /* motor speed ADC max value */
VR1 AD 値：最大速度
#define cSPEED_USER_MIN (10) /* [Hz] Min Target speed of motor */
モーター速度：最小速度 12H
#define cSPEED_USER_MAX (60) /* [Hz] Max Target speed of motor */
モーター速度：最大速度 200 Hz

/* UART Setting */
#define UART_ch UART0 /* UART Channel */
UART チャネル：CH0
#define INTERRUPT_TX INTTX0_IRQn /* UART Interrupt request */
UART0 送信割り込み要求
#define INTERRUPT_RX INTRX0_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 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 */
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_KEY_OPERATION (0x93)      /* Key operation */
        回転制御取得コマンド(※本システムでは未使用)
#define cGET_MOTOR_SPEED      (0x94)      /* motor rotation speed */
        モーター回転数取得コマンド
#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 */
```

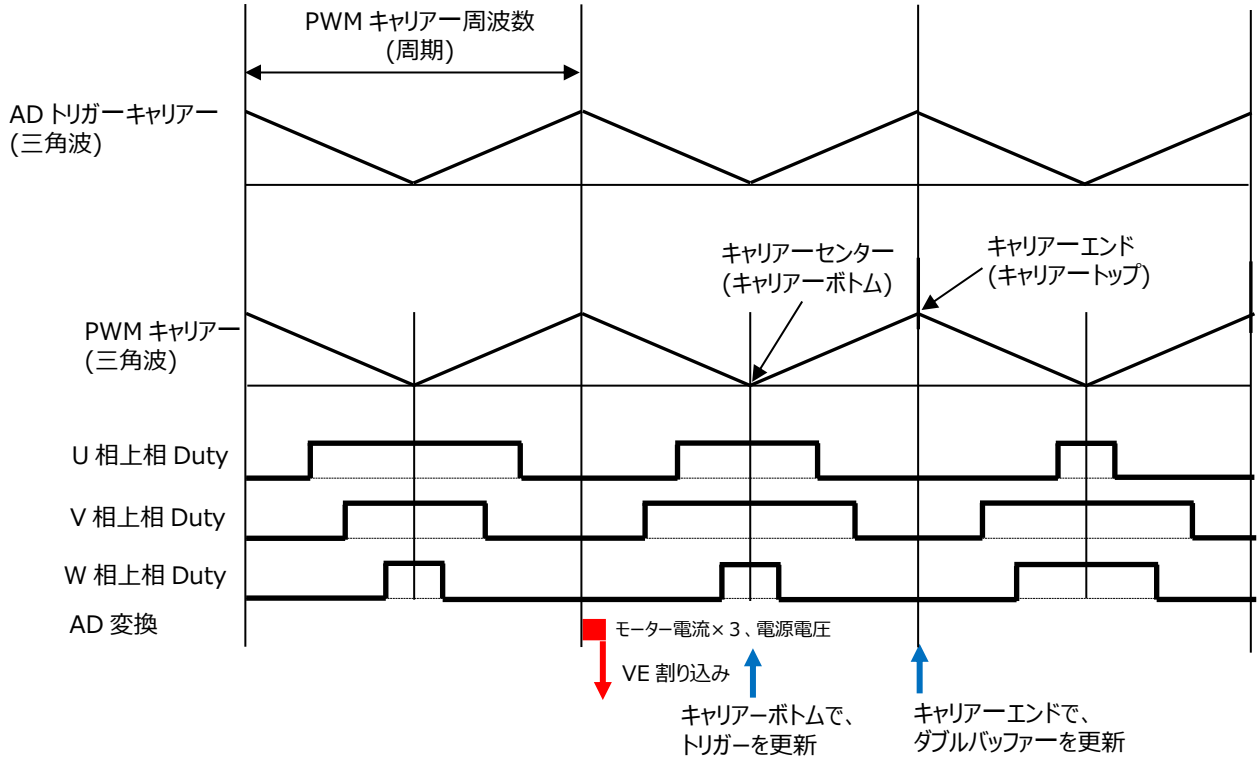
```
        EMG ステート：VDC 異常
#define cCONTROL_CH_CH0     (0x00)      /* Motor control channel：ch0 */
        モーター制御：CH0
#define cCONTROL_CH_CH1     (0x01)      /* Motor control channel：ch1 */
        モーター制御：CH1
#define cCONTROL_CH_CH2     (0x02)      /* Motor control channel：ch2 */
        モーター制御：CH2
#define cCONTROL_CH    (cCONTROL_CH_CH1)  /* User Motor control channel */
        使用モーター制御
#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
#define cPOSITION_DETECT_1SHUNT(1)       /* Position Ditect：1shunt */
        シャントタイプ： 1 シャント
#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] */
```

```
        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
```

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



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

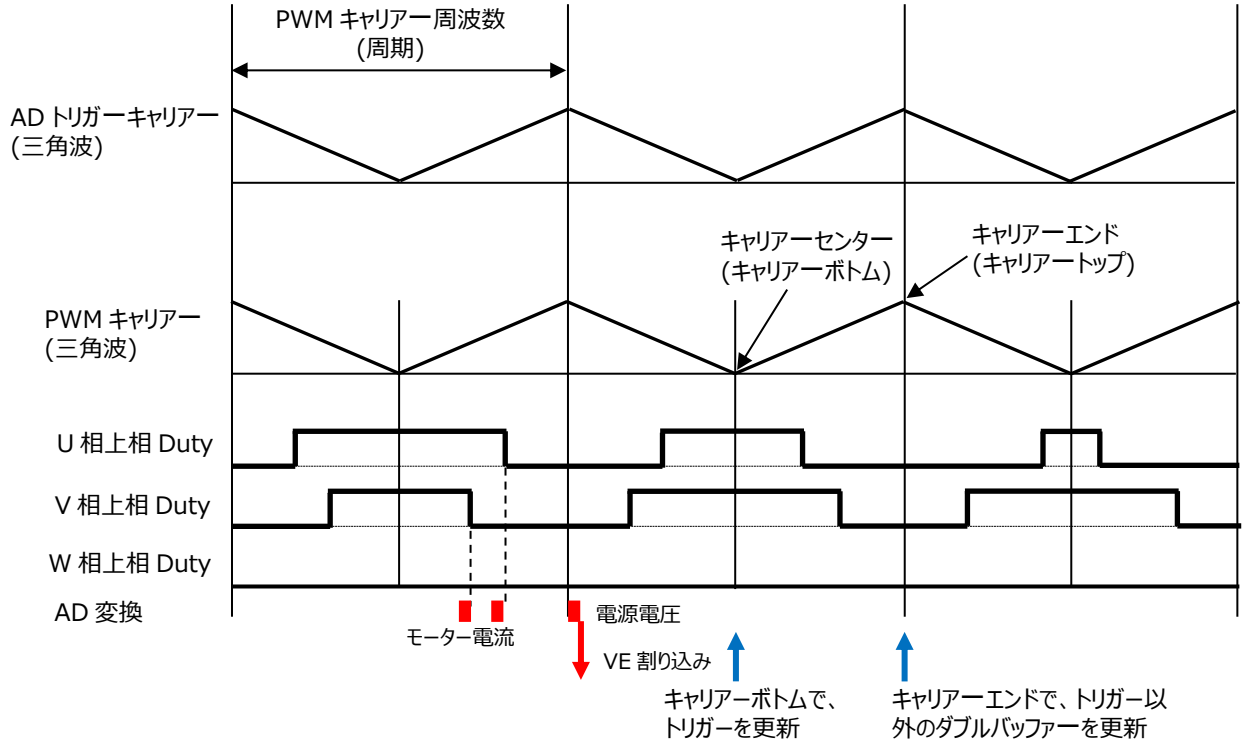
11.1.1.2 ダブルバッファー更新タイミング

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)	キャリアーボトム
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	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



11.1.2.1 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

11.1.2.2 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)	キャリアーボトム
出力設定 (MDOUT)	キャリアーエンド

11.1.2.3 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

12. ペリフェラルドライバー

12.1 IP テーブル

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：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

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：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

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：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

戻り値：

なし

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：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_DutyMed**

Duty 中間値取得

API：

uint32_t VE_GetPWM_DutyMed(const ipdrv_t* const _ipdrv)

引数：

_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

戻り値：

Duty 中間値

12.2.1.7 **VE_GetOutPutMode**

PWM 出力状態取得

API：

int VE_GetOutputMode(const ipdrv_t* const _ipdrv)

引数：

_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

戻り値：

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

12.2.1.8 **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：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

戻り値：

なし

12.2.1.9 **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：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

戻り値：

なし

12.2.1.10 **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)

引数：

_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

戻り値：

なし

12.2.1.11 **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：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

戻り値：

なし

12.2.1.12 **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：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

戻り値：

なし

12.2.1.13 **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：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

戻り値：

なし

12.2.1.14 **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)
```

引数：

_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

戻り値：

なし

12.2.1.15 **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)

引数：

_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

戻り値：

なし

12.2.1.16 **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： 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

戻り値：

なし

12.2.1.17 **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： 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

戻り値：

なし

12.2.1.18 **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: IP テーブルアドレスを選択します。

_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

12.2.1.19 **VE_SetModulType**

変調方式設定

API：

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

引数：

_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

戻り値：

なし

12.2.2 データ構造

12.2.2.1 **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 to 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

12.3 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)

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 : 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

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 : 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。

戻り値 :

なし

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
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

12.4 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：

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

cPMD：PMD チャネル 1

uint8_t pints：PMD トリガー用割り込み選択

cPINTS_B：ITTADxPDB

12.4.3 VE 搭載マイコン用定数(mcuip_drv.h)

12.4.3.1 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
cPWMEXCLK	PWM 周期延長	PWMCK_NORMAL	通常周期

	モード指定	PWMCK_4FOLD	4 倍周期
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_BOTTOM	PWM キャリアーボトムで割り込み要求発生 PMD1MDCNT<MDCNT[15:0]> = 0x0001
		PINT_TOP	PWM キャリアーピークで割り込み要求発生 PMD1MDCNT<MDCNT[15:0]> = <MDPRD[15:0]>
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
cPWMWAVE	PWM キャリアー波形選択	PWMMD_SAWTOOTH	エッジ PWM、ノコギリ波
		PWMMD_TRIANGULAR	センターPWM、三角波

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
cPSYNCS	MDOUT設定転送タイミング選択	PSYNCS_WRITE	PWM 非同期(書き込み時に反映)
		PSYNCS_BOTTOM	キャリアーボトム MDCNT= 1
		PSYNCS_TOP	キャリアーピーク MDCNT = PMDxMDPRD<MDPRD>
		PSYNCS_BOTTOM_TOP	キャリアーピークおよびキャリアーボトム MDCNT = 1 または PMDxMDPRD<MDPRD>

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	ツールブレーク時のポート制御の設定	PORTMD_ALLHIZ	上相 High-z / 下相 High-z
		PORTMD_UHIZ_LON	上相 High-z / 下相 PMD 出力
		PORTMD_UON_LHIZ	上相 PMD 出力 / 下相 High-z
		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_DOWN	ダウンカウント時の一致でトリガー出力
		TRGMD_UP	アップカウント時の一致でトリガー出力
		TRGMD_UPDOWM	アップ/ダウンカウント時にトリガー出力
		TRGMD_PEAK	PWM キャリアーピークでトリガー出力
		TRGMD_BOTTOM	PWM キャリアーボトムでトリガー出力
		TRGMD_PEAKBOTTOM	PWM キャリアーピーク/ボトムでトリガー出力
		TRGMD_DIS7	トリガー出力禁止
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_DOWN	ダウンカウント時の一致でトリガー出力
		TRGMD_UP	アップカウント時の一致でトリガー出力

		TRGMD_UPDOWM	アップ/ダウンカウント時にトリガー出力
		TRGMD_PEAK	PWM キャリアーピークでトリガー出力
		TRGMD_BOTTOM	PWM キャリアーボトムでトリガー出力
		TRGMD_PEAKBOTTOM	PWM キャリアーピーク/ボトムでトリガー出力
		TRGMD_DIS7	トリガー出力禁止
cBUFSYNC	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

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 ～ 15	異常検出入力(EMG)のノイズ除去時間設定 cEMGCNT × 16/fsys cEMGCNT >= 0 ～ 15 "0"設定時はノイズフィルターをスルーする
cINHEN	ツールブレーク時の許可/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 ～ 15	OVV 入力のノイズ除去時間設定 cOVVCNT × 16/fsys cOVVCNT >= 0 ～ 15 "0"設定時は 1 になります。
cADIN1EN	ADC(ユニット B)監視機能 1 入力許可	ADIN1EN_DISABLE	入力禁止
		ADIN1EN_ENABLE	入力許可
cADIN0EN	ADC(ユニット A)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
cOVVISEL	OVV 端子入力制御	OVVISEL_PORT	ポート入力
		OVVISEL_ADC	ADC 監視信号
cOVVRS	OVV 保護状態からの復帰	OVVRS_NORMAL	保護状態からの自動復帰禁止
		OVVRS_AUTO	保護状態からの自動復帰許可

12.4.3.2 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
cMREGDIS	SIN/COS/SECTOR 前回値保持選択	MREGDIS_EFFECTIVE	有効
		MREGDIS_NOEFFECTIVE	無効
cADCSEL1	VE チャネル 1 ADC ユニット選択	ADCSEL_UNITB	ユニット B
		ADCSEL_UNITAB	ユニット AB
cADCSEL0	VE チャネル 0 ADC ユニット選択	ADCSEL_UNITA	ユニット A
		ADCSEL_UNITAB	ユニット AB

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
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

13. 付録

13.1 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。

13.2 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）

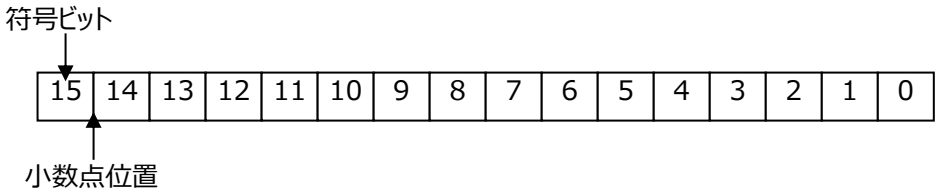


図 9 16 ビットデータの例

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。

13.3 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 2 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 3 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

13.4 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。

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

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。