

TMPM4L4

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

Arm および Keil は、Arm Limited（またはその子会社）の米国およびその他の国における登録商標です。

この資料に記載されている社名・商品名・サービス名などは、それぞれ各社が商標として使用している場合があります。

目次

1. 概要	6
1.1 はじめに	6
1.2 仕様概要	6
1.3 処理概要	8
2. システムブロック図	10
3. ソースファイル構成	11
4. 評価環境について	13
4.1 評価ボードの設定	13
4.1.1 SBK-M4L4 ボードの設定	13
4.1.2 KES-P2 ボードの設定	13
4.1.3 KES-PH ボードの設定	13
4.1.4 SBK-M4L4 と KES-P2 (KES-PH) の接続	14
4.1.5 DAC 出力の接続設定	15
4.1.6 基板修正 (KES-PH)	16
4.2 開発ツール	17
4.3 プロジェクトの立ち上げ方	17
4.4 DAC 出力	23
4.5 ポート出力	23
4.5.1 ベクトル制御ソフトウェア処理時間	23
4.5.2 電流検出状態	23
4.5.3 デバッグ出力機能	24
4.6 ステージ履歴	24
4.7 過電流検出	24
4.8 ユーザーインターフェースについて	25
5. モジュール構成	27
6. マイコンハードウェア割り付け	28
6.1 周辺インターフェース	28
6.2 割り込み	29
7. ジェネラルフローチャート	30
7.1 メインルーチン (main)	30
7.2 メインループ (main_loop)	31
7.3 割り込み	32
7.3.1 VE (ベクトルエンジン) ベクトル処理 (INT_VectorControl_byVE)	33
7.3.2 ソフトウェアベクトル処理 (INT_VectorControl_bySoft)	35
8. 状態遷移	37
8.1 センサーレス	37
8.2 エンコーダー	38
8.3 エンコーダー+ホールセンサー	39
9. ユーザーアプリ	40

9.1 ユーザー制御	40
9.1.1 AD 値取得 (soft_adc_getdata)	41
9.1.2 キー制御 (Uikeyscan)	41
9.1.3 ユーザー設定 (user_interface)	41
9.1.4 LED 表示制御 (led_display)	42
9.1.5 UART 送信データ設定 (send_data_set)	42
9.1.6 温度測定 (temp_check)	42
9.1.7 ステータス確認モニター	43
10. 機能説明	45
10.1 制御コマンド	45
10.1.1 制御方法 (usr.com_user)	45
10.1.2 制御目標速度	45
10.1.3 始動電流	45
10.2 駆動コマンド	46
10.2.1 駆動方法 (drv.command)	46
10.2.2 ベクトル制御コマンド (drv.vector_cmd)	46
10.3 駆動状態	48
10.3.1 エラー状態 (drv.state)	48
10.4 モーター制御構造体	49
10.4.1 変数一覧	49
10.5 関数詳細	56
10.5.1 エンコーダー初期設定 (init_ENCen)	56
10.5.2 ADC 初期設定 (init_ADCen)	56
10.5.3 PMD 初期設定 (init_PMDen)	57
10.5.4 VE 初期設定 (init_VEen)	57
10.5.5 シフト 2 初期設定 (init_Shift2_CHx)	58
10.5.6 コンパレータ初期設定 (init_COMPen)	58
10.5.7 モーター制御初期設定 (B_Motor_Init)	59
10.5.8 DAC 制御初期設定 (init_Dac)	61
10.5.9 周期タイマー初期設定 (init_Timer_interval4kHz)	62
10.5.10 ユーザー制御初期設定 (init_user_control)	62
10.5.11 ユーザー制御 (user_control)	63
10.5.12 ユーザーモーター制御 (B_User_MotorControl)	64
10.6 モーター制御関数	65
10.6.1 状態遷移処理関数 (C_Control_Ref_Model)	65
10.6.2 モーター制御共通処理関数 (C_Common)	67
10.6.3 停止状態関数 (C_Stage_Stop)	68
10.6.4 ブートストラップ状態関数 (C_Stage_Bootstrap)	69
10.6.5 位置決め状態関数 (C_Stage_Initposition)	70
10.6.6 強制転流状態関数 (C_Stage_Force)	71

10.6.7 強制定常切換状態関数 (C_Stage_Change_up)	73
10.6.8 定常状態関数 (C_Stage_Steady_A)	74
10.6.9 保護状態関数 (C_Stage_Emergency)	76
10.6.10 PWM シフト制御 (C_ShiftPWM_Control)	77
10.7 モーター駆動関数	78
10.7.1 用語説明	78
10.7.2 モーター電流、電源電圧取得 (D_GetMotorCurrentPowerVolt)	79
10.7.3 入力座標変換 (D_InputTransformation)	82
10.7.4 クラーク変換 (E_Clarke)	83
10.7.5 パーク変換 (E_Park)	84
10.7.6 誘起電圧位置推定関数 (D_Detect_Rotor_Position)	86
10.7.7 磁束オブザーバー位置推定関数 (D_Detect_Flux_Rotor_Position)	87
10.7.8 エンコーダー制御関数 (H_Encoder)	89
10.7.9 ホールセンサー+エンコーダー制御関数 (H_HallCon H_Hall_Encoder)	91
10.7.10 速度制御関数 (D_Control_Speed)	92
10.7.11 電流制御関数 (D_Control_Current)	93
10.7.12 出力座標変換 (D_OutputTransformation)	94
10.7.13 逆パーク変換 (E_InvPark)	96
10.7.14 セクター演算 (D_CalSector)	97
10.7.15 逆クラーク変換 (E_InvClarke)	98
10.7.16 空間ベクトル変調 (D_SVM)	99
10.7.17 トリガータイミング演算 (D_CalTrgTiming)	105
10.7.18 PWM レジスター設定 (PMD_RegDataSet)	110
10.7.19 電流誤検知検出関数 (D_Check_DetectCurrentError)	111
10.7.20 3 ショント電流誤検知検出関数 (D_Check_DetectCurrentError_3shunt)	112
10.7.21 1 ショント電流誤検知検出関数 (D_Check_DetectCurrentError_1shunt)	113
10.7.22 インバーター出力電圧の算出関数 (Cal_Vdq)	115
11. 特殊機能説明	116
11.1 磁束オブザーバー適用位置推定方式	116
11.1.1 関数仕様	117
11.1.2 データ構造	122
11.1.3 パラメーター	122
12. 定数定義説明	124
12.1 モータードライバー設定用パラメーター共通 (D_Para.h)	124
12.1.1 モーター制御チャネル選択	124
12.1.2 DAC 出力選択	124
12.1.3 疑似温度調整選択	124
12.1.4 指令速度単位選択	124
12.1.5 特殊制御選択	124
12.1.6 共通パラメーター一覧	124

12.2 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別 (D_Para_Chx. h) x=0	125
12.2.1 制御選択	125
12.2.2 モーターチャネル別パラメーター一覧	126
12.2.3 定数設定値と波形の関係	134
12.3 ユーザー制御関連定数	135
13. 制御、データ更新のタイミング	138
13.1 ソフトウェアによるベクトル制御	138
13.1.1 3シャント制御	138
13.1.2 1シャント制御	139
13.2 VE 使用によるベクトル制御	141
13.2.1 3シャント制御	141
13.2.2 1シャント制御	142
14. ペリフェラルドライバー	144
14.1 マイコン周辺回路アドレス	144
14.1.1 データ構造	144
14.2 ベクトルエンジン (VE)	144
14.2.1 関数仕様	144
14.2.2 データ構造	162
14.3 モーター制御回路 (PMD)	163
14.3.1 関数仕様	163
14.3.2 データ構造	165
14.4 アナログデジタルコンバーター (ADC)	166
14.4.1 関数仕様	166
14.4.2 データ構造	167
15. 付録	168
15.1 固定小数点処理	168
15.2 正規化 (Normalize)	168
15.3 データフォーマット	169
15.4 固定小数点での演算	171
15.5 assert 関数	171
16. 改訂履歴	172
製品取り扱い上のお願い	173

◆ モーター制御内容

表1.1モーター制御内容

項目	制御内容
モーター操作	停止または始動
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)
PWM 変調方式	3 相変調または 2 相変調
電流検出方式	3 シャントまたは 1 シャント
位置検出方式	センサーレス、エンコーダー(注 1)、 またはエンコーダー+ホール(注 1)

注 1: 未評価

・ 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(注 1)

#define cSHUNT_TYPE_CHx(1) (注 1)

#define cBOOT_TYPE_CHx(cBoot_v) (注 1)

注 1: ” x” には使用モーターのチャンネルに合わせ、0 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。

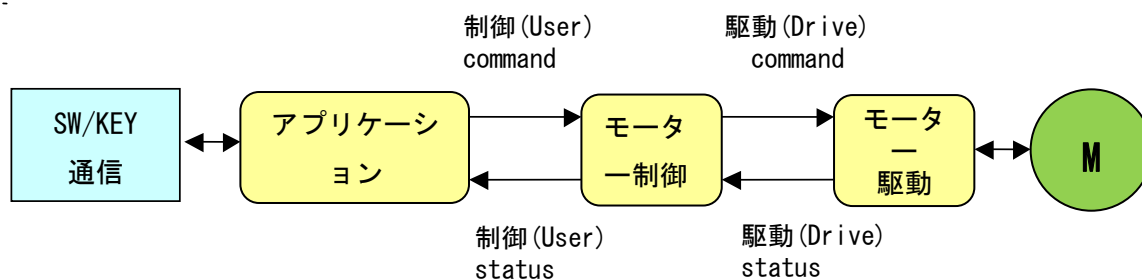


図 1.1 ベクトル制御ソフトウェアの構造

- 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。

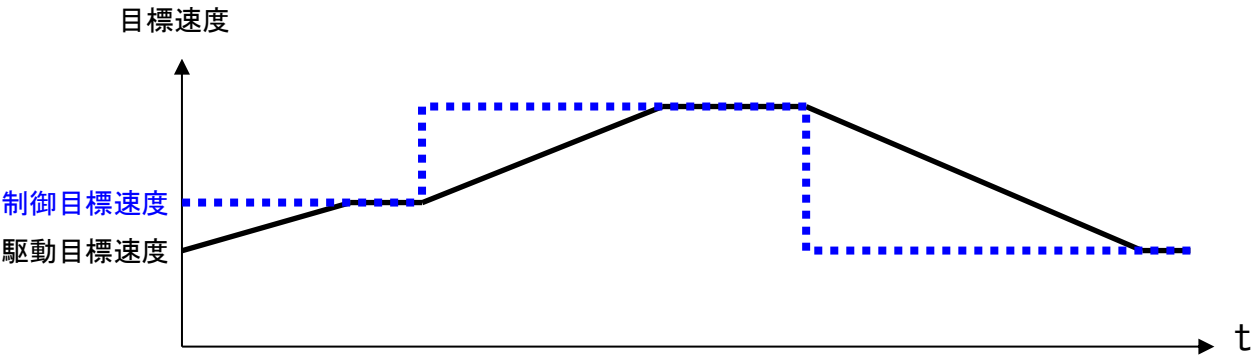


図 1.2 制御目標速度と駆動目標速度

2. システムブロック図

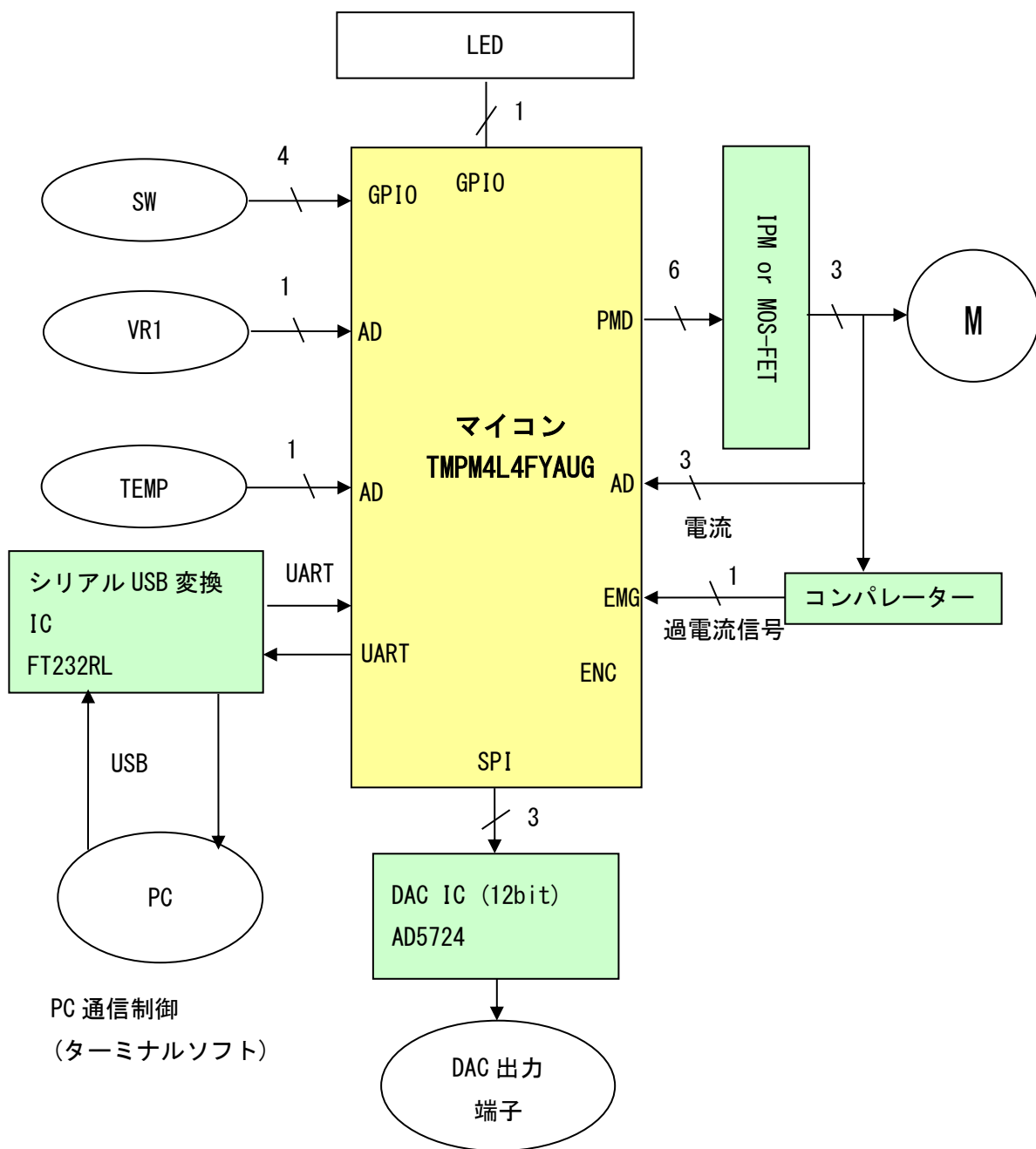


図 2.1 システムブロック図

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
C_phi.lib	磁束オブザーバー制御ライブラリーファイル(arm)
C_phi_i.a	磁束オブザーバー制御ライブラリーファイル(IAR)
C_phi_s.a	磁束オブザーバー制御ライブラリーファイル(SEGGER)
C_phi.h	磁束オブザーバー制御ヘッダーファイル(注 1)
C_phi_KES-P2.h	磁束オブザーバー制御ヘッダーファイル(注 1)
C_phi_KES-PH.h	磁束オブザーバー制御ヘッダーファイル(注 1)
CheckVdq.c	dq 軸電圧チェックソースファイル
CheckVdq.h	dq 軸電圧チェックヘッダーファイル
dac_drv.c	DAC IC ドライバーソースファイル
dac_drv.h	DAC IC ドライバーヘッダーファイル
D_Driver.c	ベクトル制御ドライバーソースファイル
D_Driver.h	ベクトル制御ドライバーヘッダーファイル
D_Para.h	ベクトル制御パラメーター(チャンネル共通)ヘッダーファイル
D_Para_ch0.h	ベクトル制御パラメーター(チャンネル 0 用)ヘッダーファイル(注 2)
D_Para_ch0_KES-P2.h	ベクトル制御パラメーター(チャンネル 0 用)ヘッダーファイル(注 2)
D_Para_ch0_KES-PH.h	ベクトル制御パラメーター(チャンネル 0 用)ヘッダーファイル(注 2)
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
initial.c	マイコン初期設定関連ソースファイル
initial.h	マイコン初期設定関連ヘッダーファイル
interrupt.c	割り込み制御関連ソースファイル
interrupt.h	割り込み制御関連ヘッダーファイル
H_Hall.c	ホールセンサーベクトル制御用ソースファイル
H_Hall.h	ホールセンサーベクトル制御用ヘッダーファイル
ipdefine.h	マイコン設定関連ヘッダーファイル
main.c	メインルーチン
mcuip_drv.c	マイコンハード設定ドライバーソースファイル
mcuip_drv.h	マイコンハード設定ドライバーヘッダーファイル
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
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
usercon.c	ユーザー制御用ソースファイル(注 3)
usercon_KES-P2.c	ユーザー制御用ソースファイル(注 3)
usercon_KES-PH.c	ユーザー制御用ソースファイル(注 3)
usercon.h	ユーザー制御用ヘッダーファイル

Libraries

CMSIS コア、各 IP 用ライブラリーが格納されています。

M4L4

CMSIS

CMSIS コアが格納されています。

| system_TMPM4LxA.c

マイコン初期設定用ソースファイル

| system_TMPM4LxA.h

マイコン初期設定用ヘッダーファイル

| TPM4L4A.h

マイコンレジスター定義ヘッダーファイル

| TPM4LxA.h

マイコンレジスター定義選択用ヘッダーファイル

startup

arm

| startup_TMPM4L4A.s

スタートアップアセンブラソース

		(arm 用)
	M4L4.scat	リンカーファイル (arm 用)
	└─iar	
	startup_TPM4L4A.s	スタートアップアセンブラーソース
		(IAR 用)
	└─segger	
	TMPM4Lx.icf	リンカーファイル (IAR 用)
	TMPM4L4A_Vectors.s	スタートアップアセンブラーソース
		(SEGGER 用)
	SEGGER_THUMB_Startup.s	スタートアップアセンブラーソース
		(SEGGER 用)
	TXZplus4E_M4L_Startup.s	スタートアップアセンブラーソース
		(SEGGER 用)
	TXZplus4E_M4L_Flash.icf	リンカーファイル (SEGGER 用)
	└─IP_Driver	
		ペリフェラルドライバーが格納されています。
	ipdrv_adc.c	
	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	

注 1: C_phi.h の初期状態は KES-P2 用コードです。KES-PH を使用する際は C_phi_KES-PH.h のコードへ書き換えてください。

注 2: D_Para_ch0.h の初期状態は KES-P2 用コードです。KES-PH を使用する際は D_Para_ch0_KES-PH.h のコードへ書き換えてください。

注 3: usercon.c の初期状態は KES-P2 用コードです。KES-PH を使用する際は usercon_KES-PH.c のコードへ書き換えてください。

4. 評価環境について

4.1 評価ボードの設定

本ソフトは以下の評価ボードを使用して開発されたものです。

4.1.1 SBK-M4L4 ボードの設定

電源供給元選択がインバーター基板になるようにジャンパ設定します。

表 4.1 SBK-M4L4 ボードの設定

ジャンパ名	端子名 (シルク標示)	機能	オープン	ショート	評価時の設定
JP1	DAPEN	オンボードICE使用 (基板上のCMSIS-DAPを使用する場合は ショート)	未使用	使用	ショート
JP2	DnBOOT	M366-B00Tモード設定 (通常は、必ずオープンにしてくだ さい)	禁止	許可	オープン
JP3	B00T	M4KN-B00Tモード設定 (通常はオープンにしてください)	禁止	許可	オープン
JP4	LVSFT	レベルシフト回路許可 (CMSIS-DAPを使用する場合はショート)	禁止	許可	ショート
JP5 JP6 JP7	USB OBJT 5VIN	マイコン5V電源供給元選択 JP5, JP6, JP7は必ず表の通り設定してくだ さい。 (ショートは1つのジャンパのみにして下 さい。)	電源供給元 JP5 ショート JP6 オープン JP7 オープン	USB2 (USB) ショート USB1 (DAP) オープン ショート オープン	インバーター基板 オープン オープン ショート
JP8	(VDD-AVDD)	デジタル/アナログ 5V電源分離	分離	共通	共通
JP9	(GND-AGND)	デジタル/アナログ GND分離	分離	共通	共通

4.1.2 KES-P2 ボードの設定

ソフトウェアの設定と同じになるように1シャントまたは3シャントのジャンパ設定にします。

表 4.2 KES-P2 ボードの設定

位置検出	JPR	3シャント方式	1シャント方式	
3/1シャント切替	JPR1	オープン	ショート	ソフトウェア側の設定と必ず合わせ るようにしてください。
	JPR2	オープン	オープン	
	JPR3	オープン	オープン	
	JPR4	ショート	オープン	

4.1.3 KES-PH ボードの設定

KES-PHは3シャント専用基板です。ジャンパ設定はありません。

4. 1. 4 SBK-M4L4 と KES-P2 (KES-PH) の接続

SBK-M4L4 と KES-P2 間は付属の 26pin フラットケーブルで接続します。

SBK-M4L4 と KES-PH 間は下図のように、モーター電流（ADU, ADV, ADW）と基準電圧 VDC に約 8kHz の LPF を追加します。

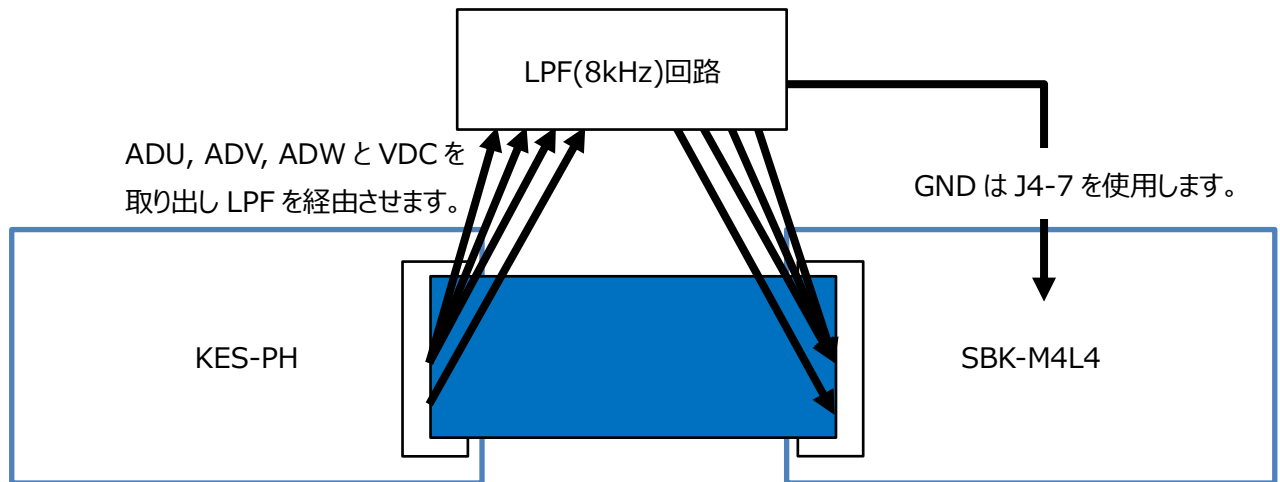


図 4.1 LPF の追加

4. 1. 5 DAC 出力の接続設定

DAC 出力機能を使用するには、KES-P2 に付属の DAC 基板(KES-DAC12)と SBK-M4L4 基板との接続が必要です。

下表のとおりに接続します。

表 4.3 DAC 出力の接続設定

DAC基板 KES-DAC12 J28			SBK-M4L4基板 J4	
番号	信号名		番号	信号名
1	DVDD	→	5	VDD
2	GND	→	6	GND
			SBK-M4L4基板 J7	
3	TSPI1RXD	→	5	TSPI1RXD/PC3
4	TSPI1TXD	→	4	TSPI1TXD/PC2
5	TSPI1CLK	→	6	TSPI1SCK/PC1

4.1.6 基板修正 (KES-PH)

KES-PH 上の DC-DC コンバーターを無効化します。(電源電圧と DC-DC 間のダイオード D7 を取り外します)

5V と 15V にはそれぞれ安定化電源を使用します。

抵抗 R25 の PGND 側に 15V 電源の GND を、
反対側に+を接続します。

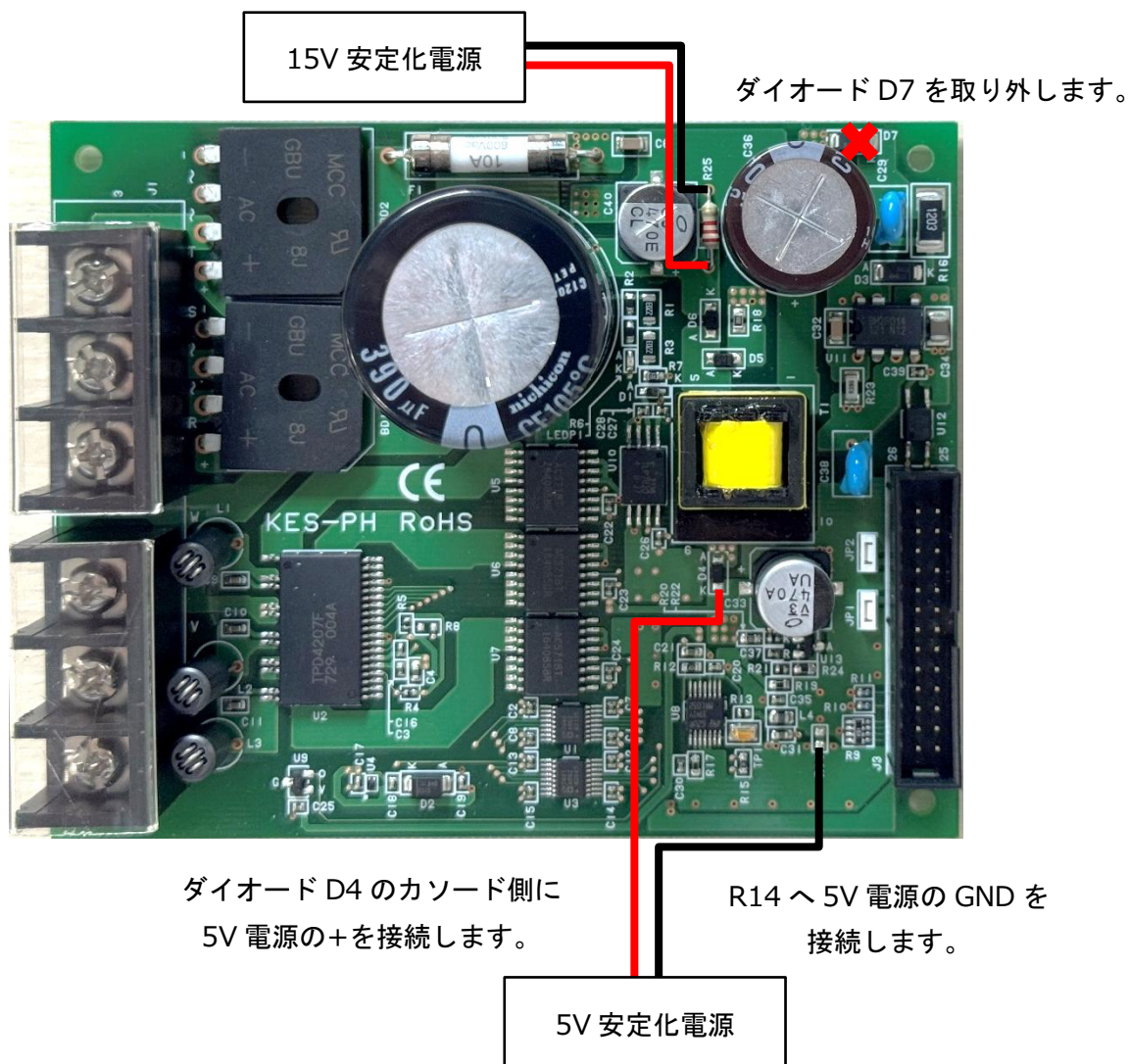


図 4.2 基板修正(KES-PH)

4.2 開発ツール

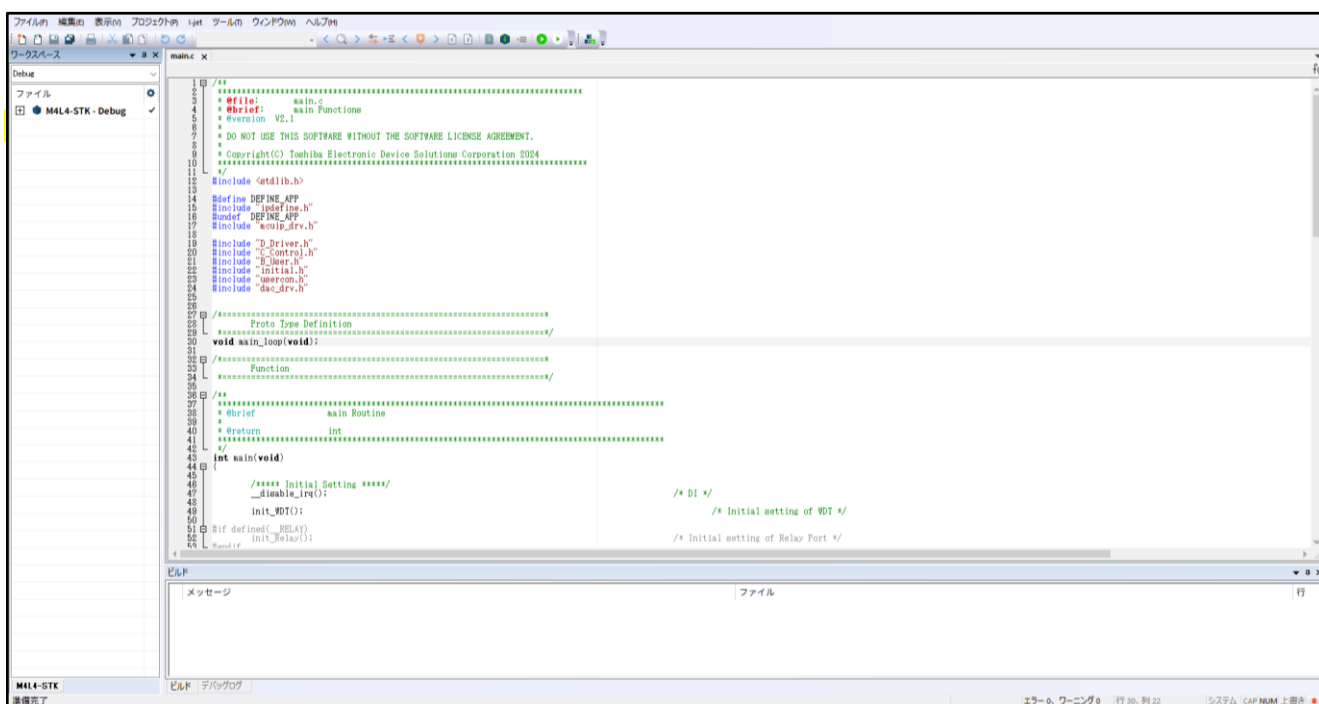
本ソフトは以下の開発ツールに対応しています。

IAR Embedded Workbench for ARM	9.70
Keil μ Vision MDK	5.43
Segger Embedded Studio	8.24

4.3 プロジェクトの立ち上げ方

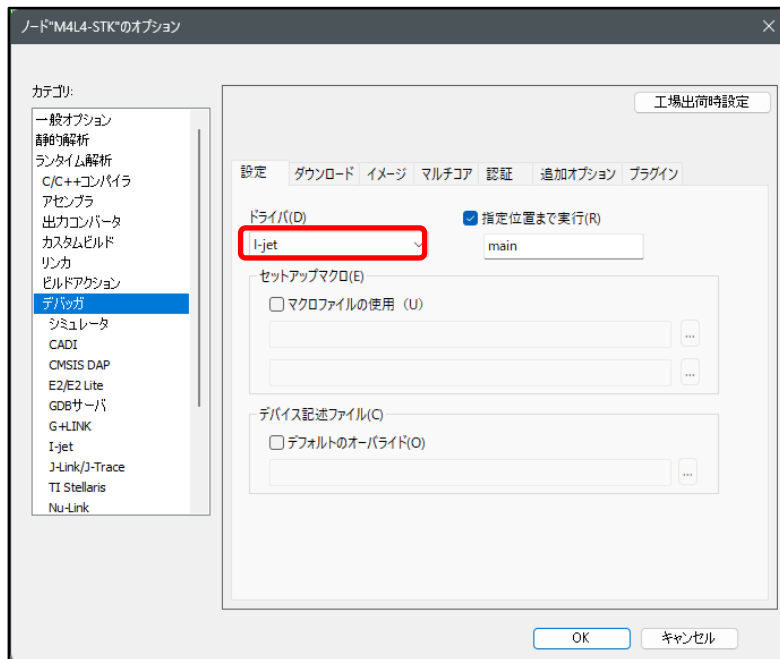
IAR Embedded Workbench の場合を説明します。

1. iar¥M4L4-STK.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 M4L4-STK.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



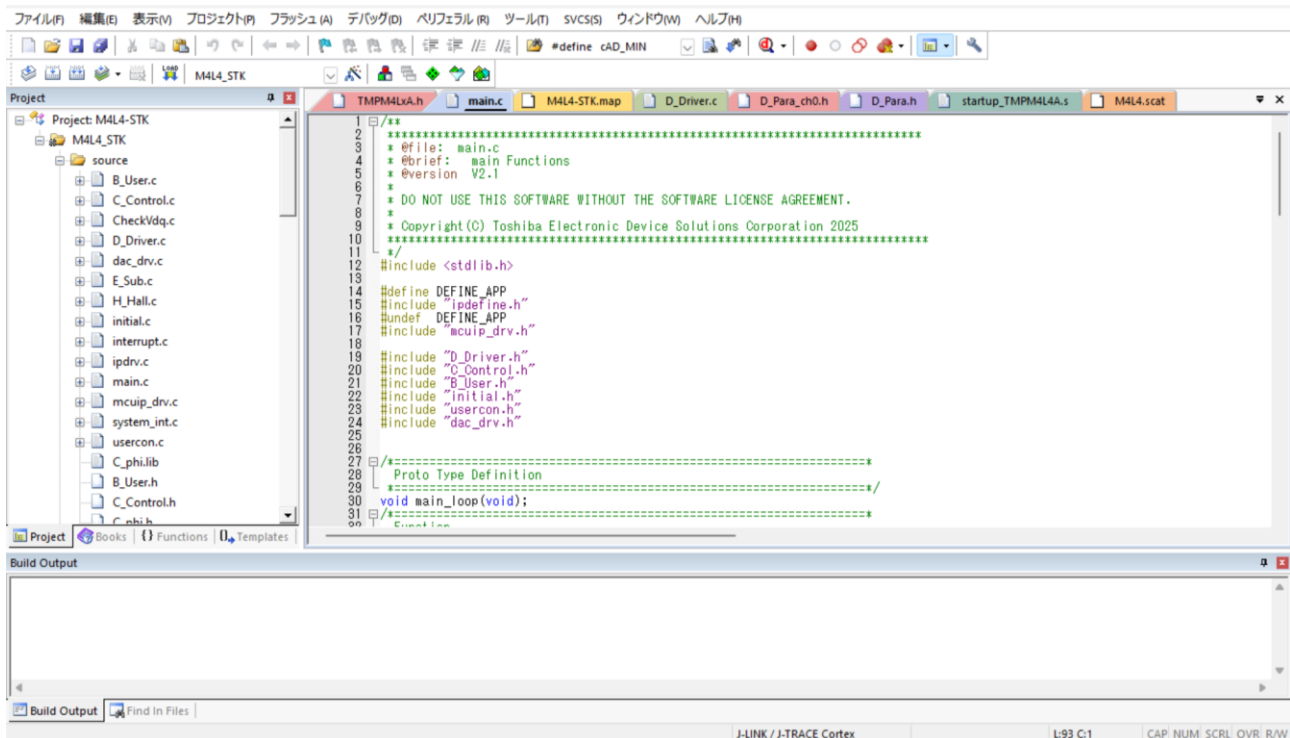
使用するツールを選択してください。

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。

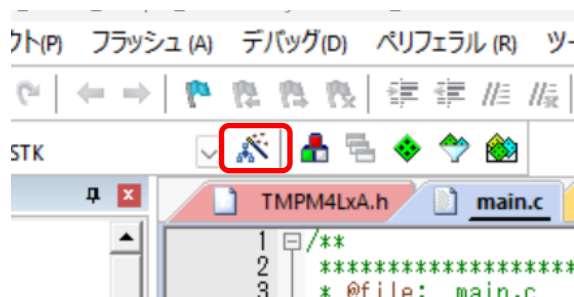


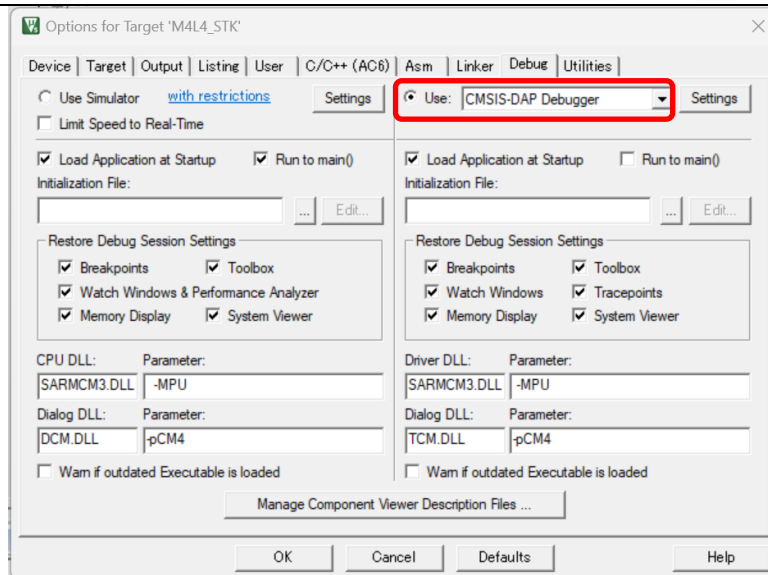
Keil μ Vision MDK の場合を説明します。

1. M4L4-STK.uvprojx をダブルクリック、または[ファイル] > [開く]から
M4L4-STK.uvprojx を開いてください。
2. 下記画面が立ち上がります。



3. ターゲットの設定を開き、使用するツールの選択を行ってください。
[ターゲットの設定] > [Debug]の<Use>欄





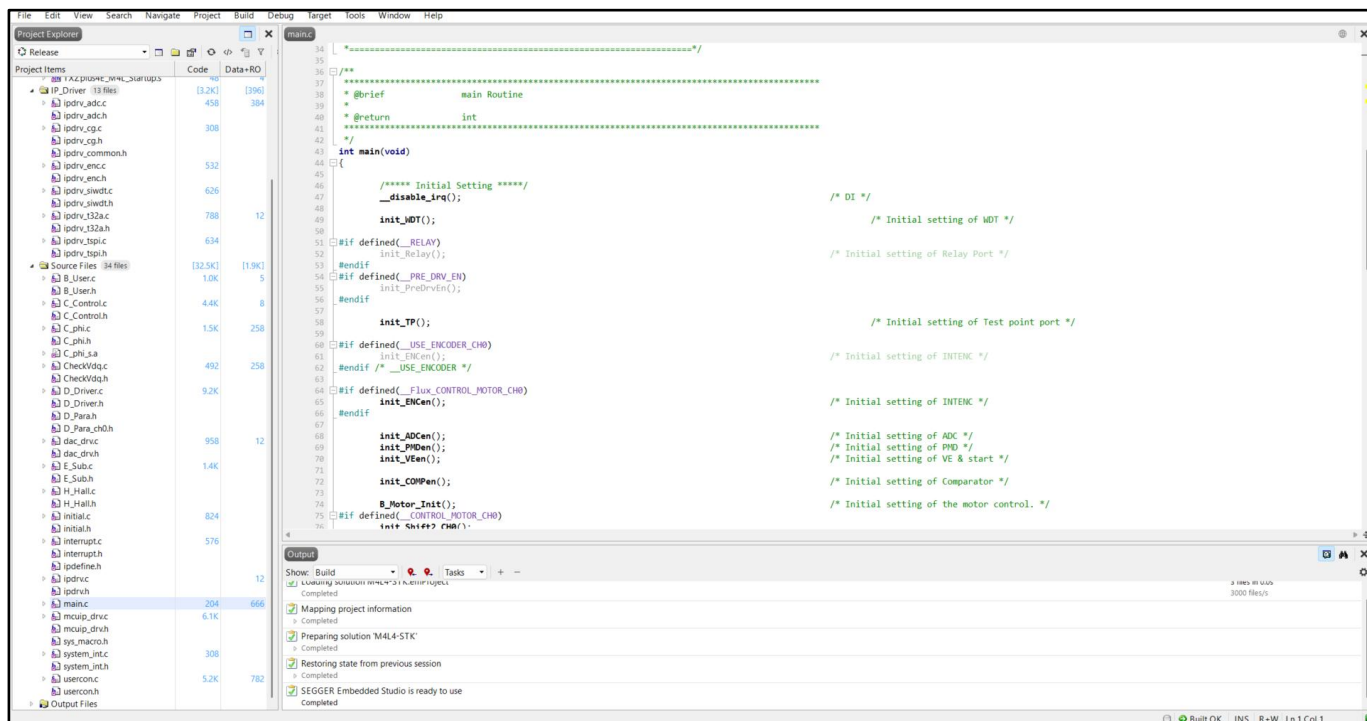
使用するツールを選択してください。

- デバッグを開始するときは、ツールを接続し[デバッグ] > [デバッグセッションの開始/停止]を選択するか、下記ボタンを押してください。

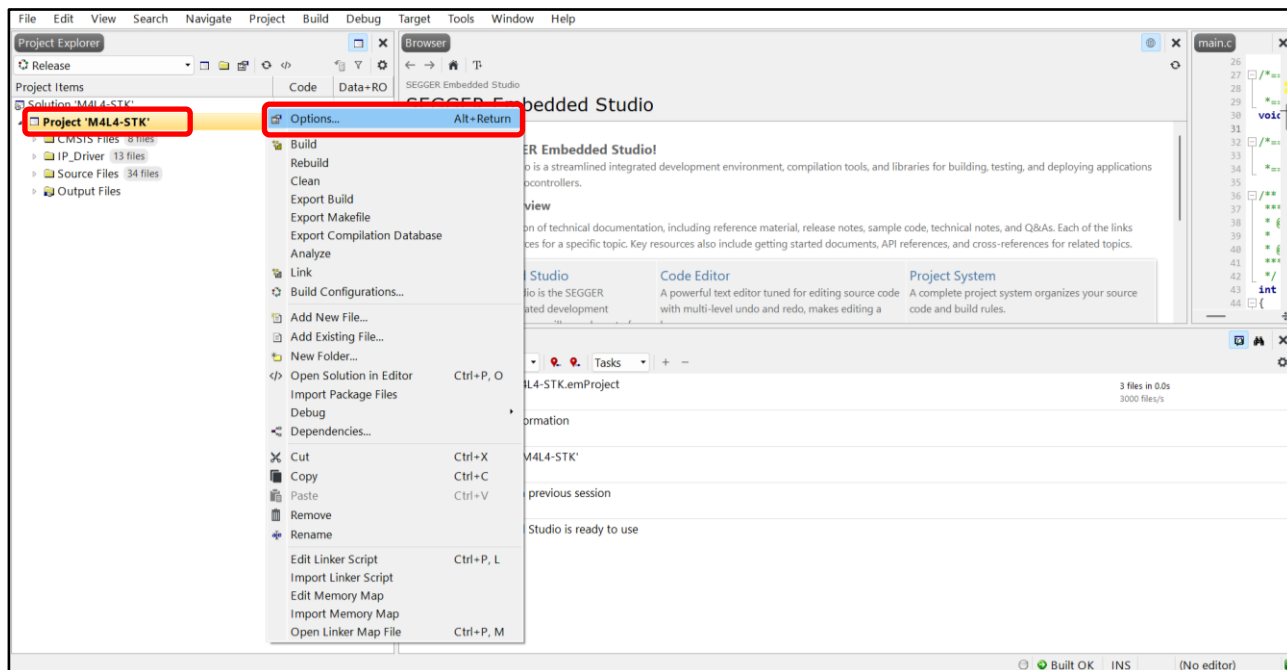


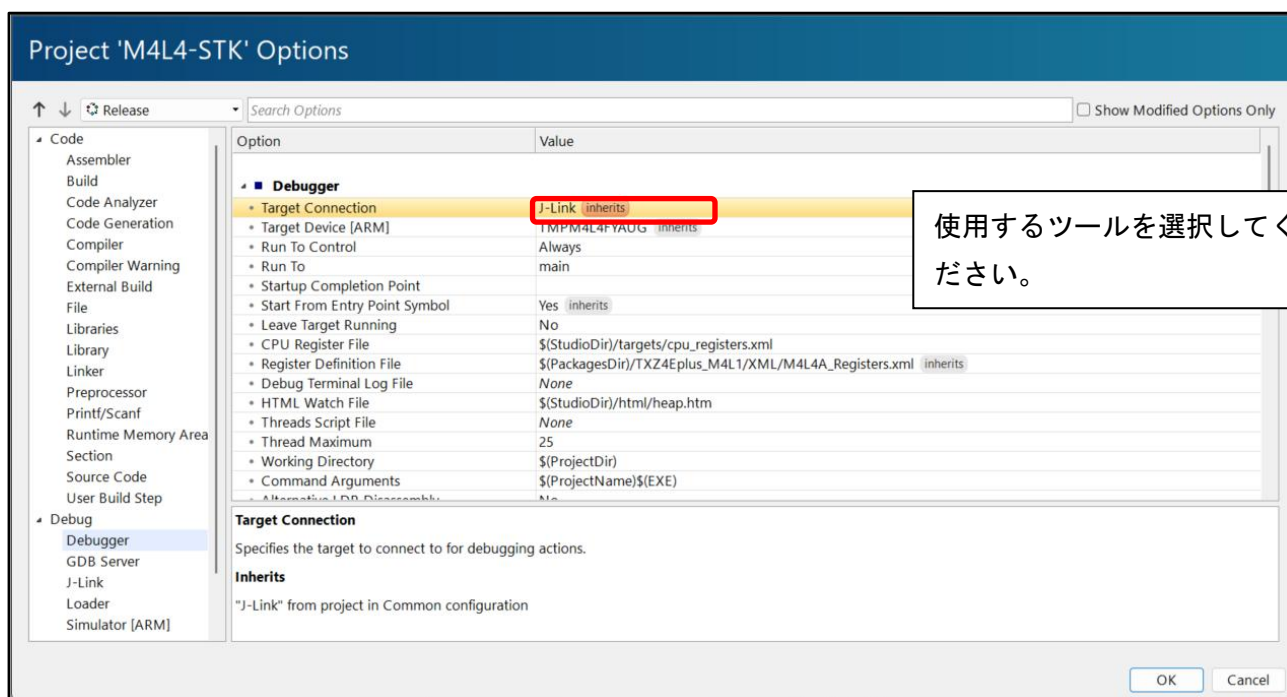
Segger Embedded Studio の場合を説明します。

1. M4L4-STK.emProject をダブルクリック、または[File] > [Open]から M4L4-STK.emProject を開いてください。
2. 下記画面が立ち上がります。

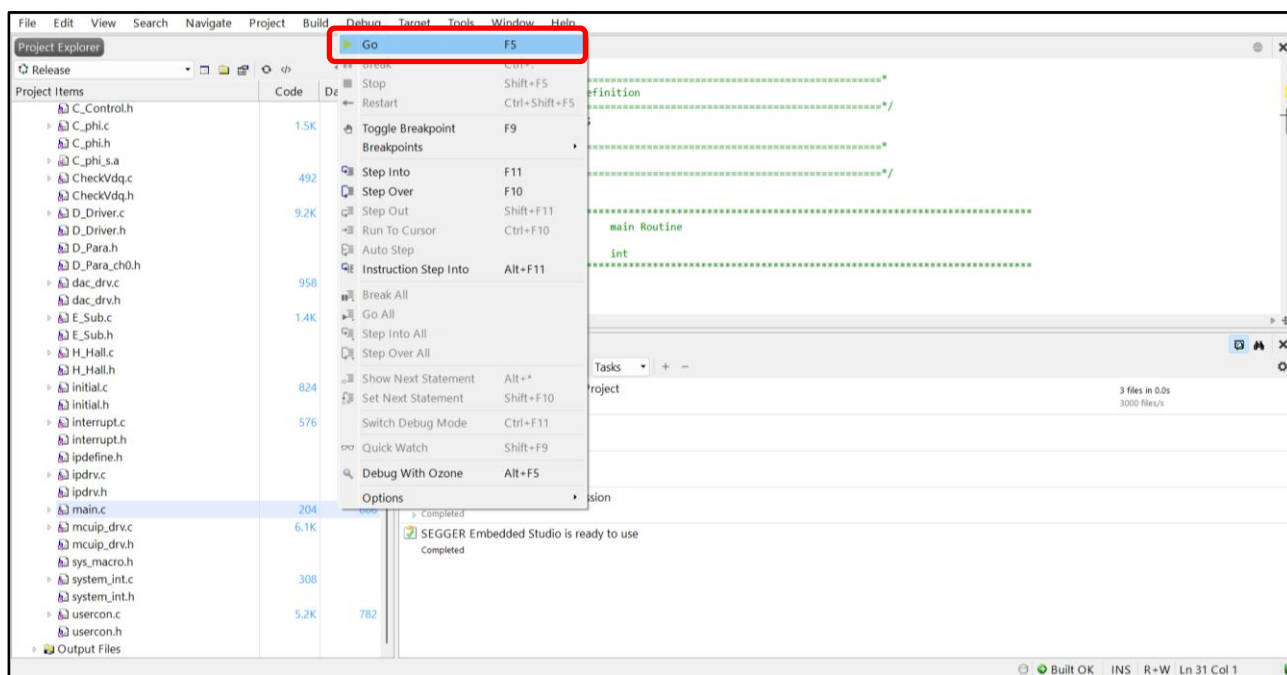


プロジェクト名を右クリック > [Options] > [Debugger]の<Target Connection>





4. デバッグを開始するときは、ツールを接続し[Debug] > [Go]を選択してください。



4.4 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。

確認する変数がその関数に無い場合などは、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 データのシフト量を設定してください。
 計算式(x) : data × (2^x)

4.5 ポート出力

各種データを選択したポートに出力することが可能です。

初期設定でデータを出力する設定になっているポートは下記のとおりです。必要に応じて変更してください。

4.5.1 ベクトル制御ソフトウェア処理時間

ベクトル制御のソフトウェア処理時間を出力することが可能です。

この機能を有効にするには、ファイル ipdefine.h の下記定義を有効にしてください。

```
#define    __MOTOR_DBGOUT_VECTOR_TIME_INT
```

表 4.1 出力ポート

モーターチャネル	出力ポート
CH0	PA bit5

H : 演算中

4.5.2 電流検出状態

電流検出の状態を出力することが可能です。

この機能を有効にするには、ファイル ipdefine.h の下記定義を有効にしてください。

```
#define    __MOTOR_DBGOUT_IDET
```

表 4.2 出力ポート

モーターチャネル	出力ポート
CH0	PA bit6

L : 正常検出状態、H : 誤検出状態

4.5.3 デバッグ出力機能

PMD のデバッグ出力機能 (PMD、ADC、VE から) の信号を出力することが可能です。
この機能を有効にするには、ファイル ipdefine.h の下記定義を有効にしてください。

```
#define __MOTOR_PMDDBG
```

表 4.3 出力ポート

モーターチャネル	出力ポート
CH0	PC bit4

出力させる信号を変更する場合は、ファイル mcuip_drv.c 関数 IP_PMD_init() の DBGOUTCR レジスタの設定を変更してください。デフォルトは、AD 変換タイミングモニター信号に設定されています。

詳細はデータシート PMD デバッグ出力機能の項を参照してください。

4.6 ステージ履歴

メインステージの履歴を残すことで、どの状態から異常状態になったかなどを確認することが可能です。

ステージの履歴は下記変数に格納されます。

```
stage_his[]
```

この機能を有効にするには、ファイル ipdefine.h の下記定義を有効にしてください。

```
#define __MOTOR_DEBUG
```

4.7 過電流検出

PWM シフト 2 モード時は Vd/Vq 出力制限を行うため過電流検出は行いませんが、Vd/Vq 出力制限値を決定する際に過電流検出を行うことが可能です。

この機能を有効にするには、ipdefine.h の下記定義を有効にしてください。

```
#define __MOTOR_OVC_CHECK_SHIFT2
```

4.8 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. 回転方向切換

モーター停止中に 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。

off : CW on : CCW

2. 回転数調整/疑似温度調整

モーター停止中に SW(S_SW3) を切り替えることで、
VR1 の操作を
モーターの回転数調整または疑似温度調整に
切り替えることができます。

off : 回転数調整 on : 疑似温度調整

3. DAC 表示内容を更新

SW(U_SW1) を押下することで、
最新状態の DAC を表示できます。

4. DAC No 切り替え

SW(U_SW2) を押下することで、
下記の表に従い、DAC No を切り替えることができます。

No.	DAC 表示
1	A: Ia B: Ib C: Ic D: theta.half[1]
2	A: Id_ref B: Id C: Iq_ref D: Iq
3	A: omega_com.half[1] B: omega.half[1] C: omega_ref D: Iq_ref
4	A: Ia B: Iq_ref C: Id_ref D: omega.half[1]

5. ボリューム

VR1 を操作することにより、
モーターの回転数調整/疑似温度調整ができます。
疑似温度調整を使用するためには、D_Para.h の下記定義を有効にしてください。

```
#define __USE__FAKETEMP
```

疑似温度調整 : -20℃～100℃

回転数調整 : 0, cHz_MIN～200Hz

6. “LED 表示”

LED でモーターのエラー状態を確認することができます。

STOP ステージ中のハード EMG の場合、LED の点灯は行われません。

モーターCH0 駆動時(LED4)

エラーステータス LED

EMG なし：消灯

EMG あり：点灯

リセットボタンで EMG 状態が解除されます。

7. UART 接続によるステータス表示

PC 上に、設定、モーター回転数、EMG ステータスを表示することができます。

リセット解除後に 1 回初期ステータスを確認することができます。

詳細は [9.1.7 ステータス確認モニター](#)を参照ください。

5. モジュール構成

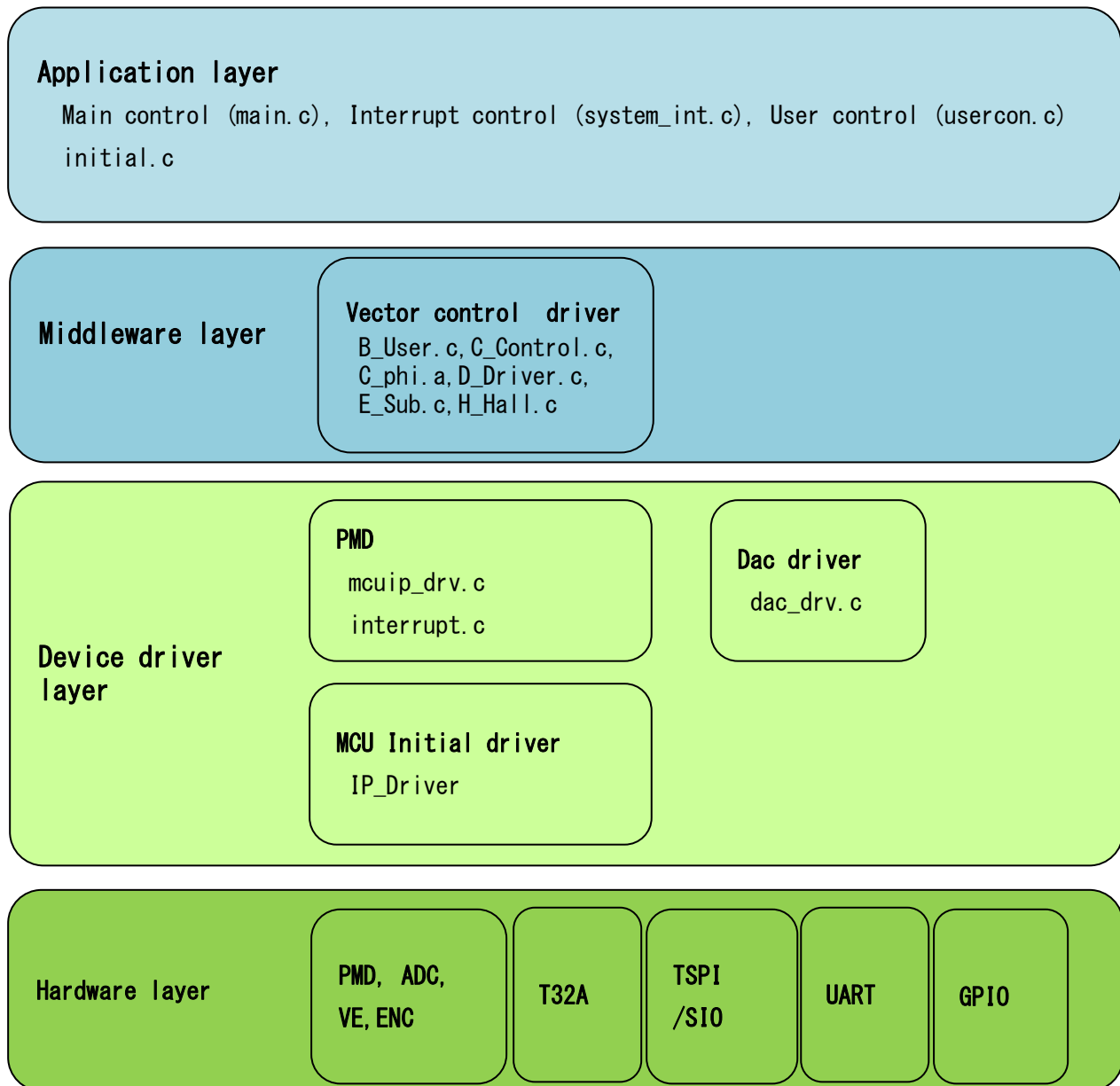


図 5.1 モジュール構成

6. マイコンハードウェア割り付け

6.1 周辺インターフェース

表 6.1 周辺インターフェース

IP		制御内容	備考
T32A	チャンネル 0	4kHz 周期割り込み	タイマーA
	チャンネル 1	未使用	
	チャンネル 2	未使用	
	チャンネル 3	未使用	
	チャンネル 4	未使用	
TSPI	チャンネル 0	未使用	
	チャンネル 1	DAC IC 制御	SIO モード
	チャンネル 2	未使用	
UART	チャンネル 0	PC 接続	
	チャンネル 1	未使用	
	チャンネル 2	未使用	
	チャンネル 3	未使用	
	チャンネル 4	未使用	
	チャンネル 5	未使用	
ADC	ユニット A	モーターCH0 コイル電流、電源電圧値取得 VR1、TEMPO 電圧値取得	
	ユニット B	未使用	
A-PMD2	チャンネル 0	モーターCH0 制御	
A-VE2	チャンネル 0	モーターCH0 制御	
A-ENC2	チャンネル 0	モーターCH0 エンコーダー信号制御	
COMP	チャンネル 0	U 相モーター電流 (正) 基準電圧入力	
	チャンネル 1	U 相モーター電流 (負) 基準電圧入力	
	チャンネル 2	V 相モーター電流 (正) 基準電圧入力	
	チャンネル 3	V 相モーター電流 (負) 基準電圧入力	
	チャンネル 4	W 相モーター電流 (正) 基準電圧入力	
	チャンネル 5	W 相モーター電流 (負) 基準電圧入力	
DAC	チャンネル 0	モーター電流 (正) 基準電圧出力	
	チャンネル 1	モーター電流 (負) 基準電圧出力	

6.2 割り込み

表 6.2 割り込み

要因名	処理内容	優先度	関数名
INTT32A00AC_IRQn	4kHz 周期タイミング作成	5	INTT32A00AC_IRQHandler
INTVCNO_IRQn	CH0 用ベクトル制御ソフト処理 (VE によるベクトル制御用)	3	INTVCNO_IRQHandler
INTADAPDA_IRQn	ベクトル制御ソフト処理 for CH0 (ソフトウェアによるベクトル制御用)	3	INTADAPDA_IRQHandler
INTUART0TX_IRQn	UART 送信完了割り込み処理	6	INTUART0TX_IRQHandler
INTT1TX_IRQn	DAC IC 制御	6	INTT1TX_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  */

INT4KH_LEVEL      5      /* 4kH interval timer interrupt */
INT_VC_LEVEL      3      /* VE interrupt */
INT_ADCA_LEVEL    3      /* ADC A interrupt */
INT_DAC_LEVEL     6      /* SIO interrupt for Dac */
INT_UART_LEVEL    6      /* SIO interrupt for UART */

```

7. ジェネラルフローチャート

7.1 メインルーチン (main)



図 7.1 メインルーチン (main)

7.2 メインループ (main_loop)

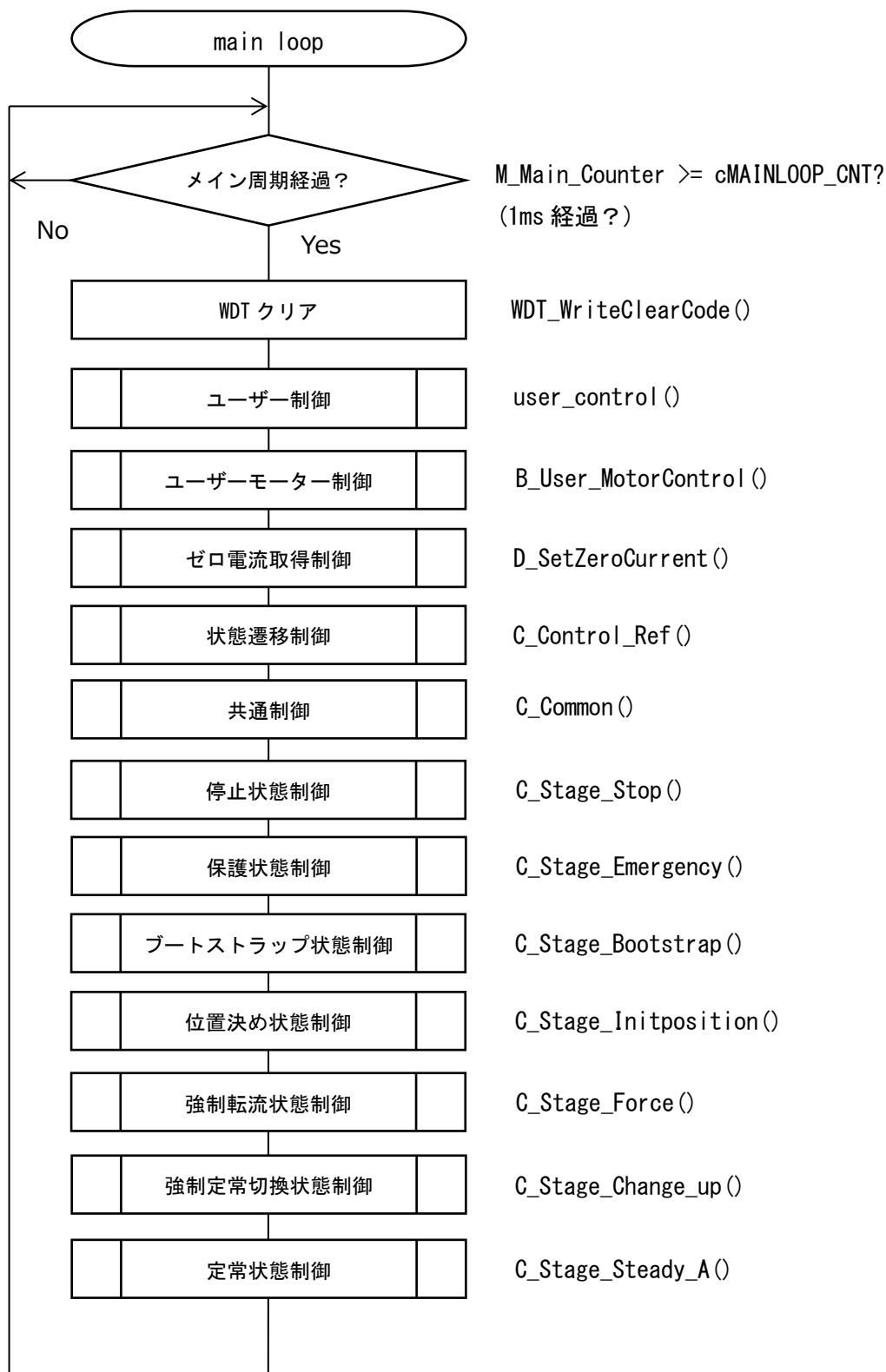
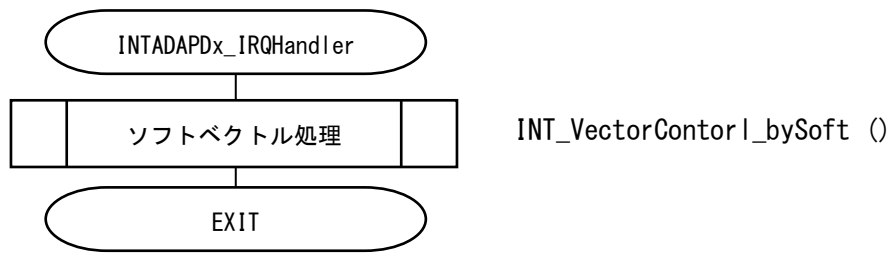


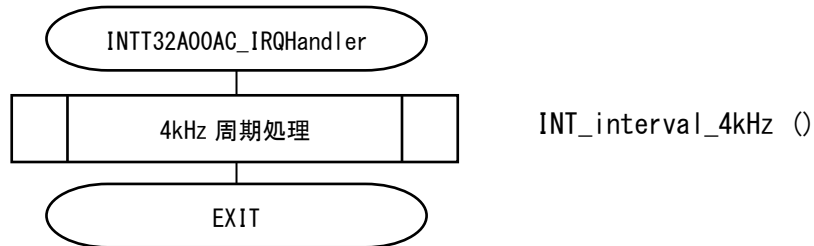
図 7.2 メインループ (main_loop)

7.3 割り込み

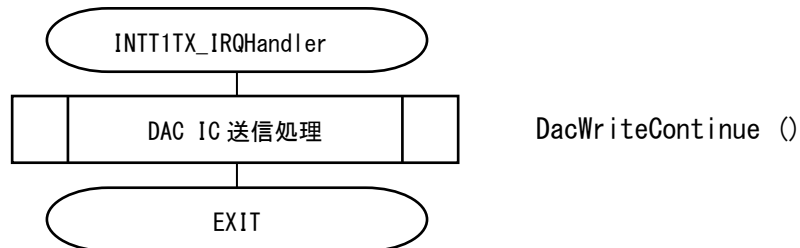
AD 変換終了割り込み (x=A)



4kHz 周期ごと



DAC 通信 24bit 送信完了時



UART8bit 送信完了時



図 7.3 割り込み

7.3.1 VE (ベクトルエンジン) ベクトル処理 (INT_VectorControl_byVE)

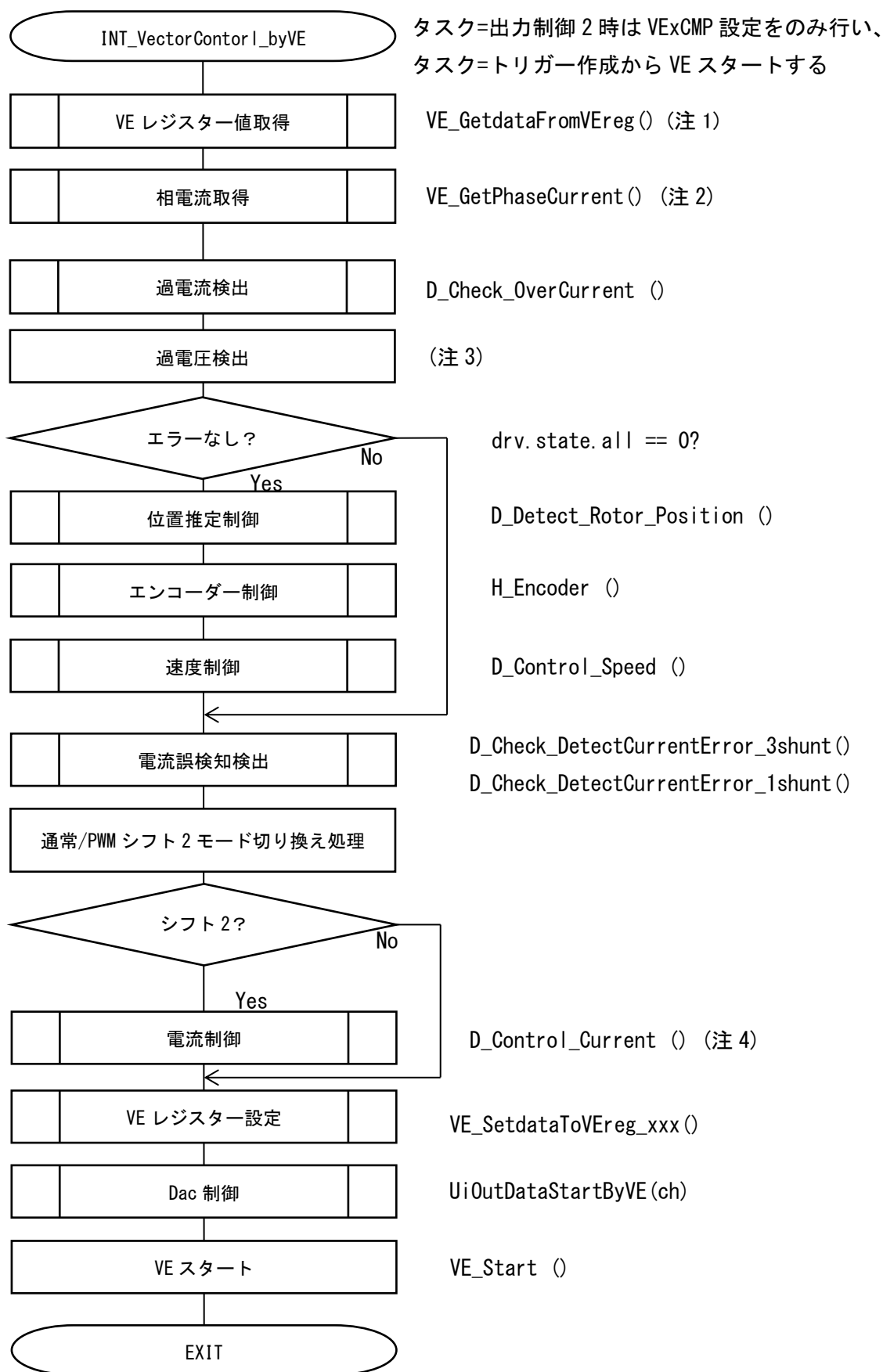


図 7.4 ベクトル処理 (INT_VectorControl_bySoft)

- 注 1: 磁束オブザーバー方式時、 V_{α}/V_{β} を取得 (VE レジスターで取得)
- 注 2: 磁束オブザーバー方式時、 I_{α}/I_{β} を取得 (VE レジスターで取得)
- 注 3: 磁束オブザーバー方式時
- 注 4: シフト 2⇔通常の切り換え時は処理なし

7.3.2 ソフトウェアベクトル処理 (INT_VectorControl_bySoft)

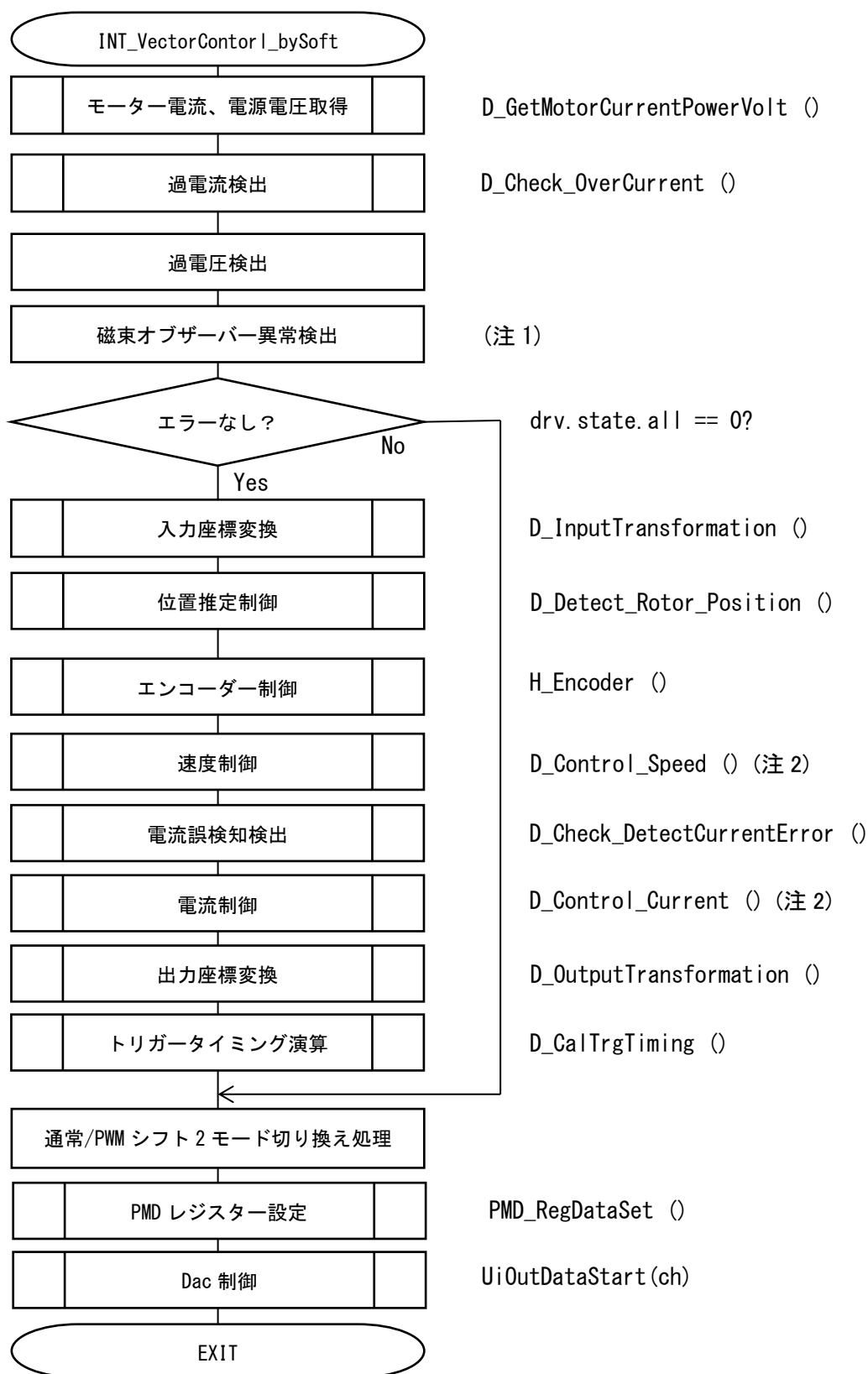


図 7.5 ソフトウェアベクトル処理 (INT_VectorControl_bySoft)

注 1: 磁束オブザーバー方式時

注 2: シフト 2⇔通常の切り換え時は処理なし

8. 状態遷移

8.1 センサーレス

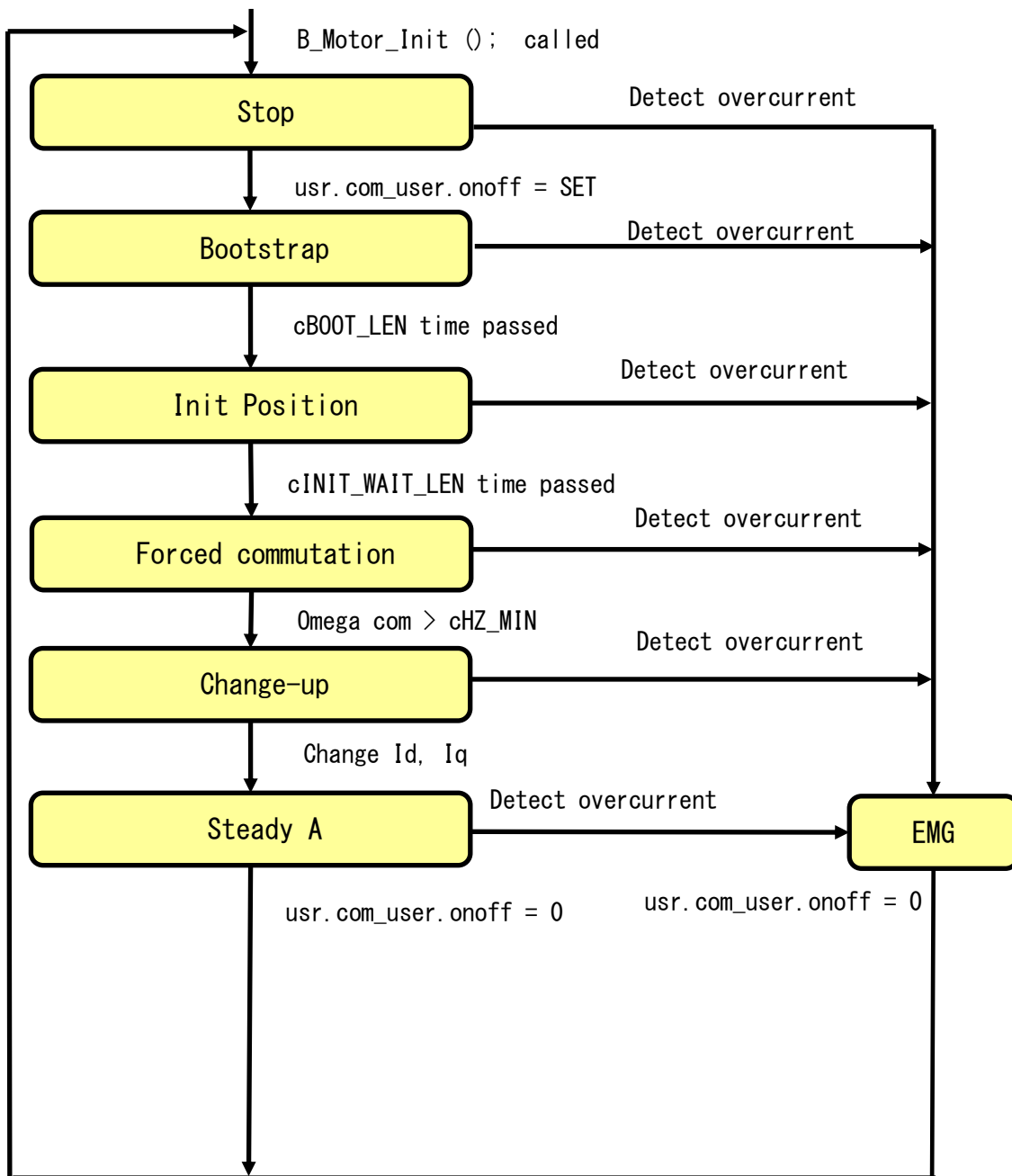


図 8.1 センサーレス

8.2 エンコーダー

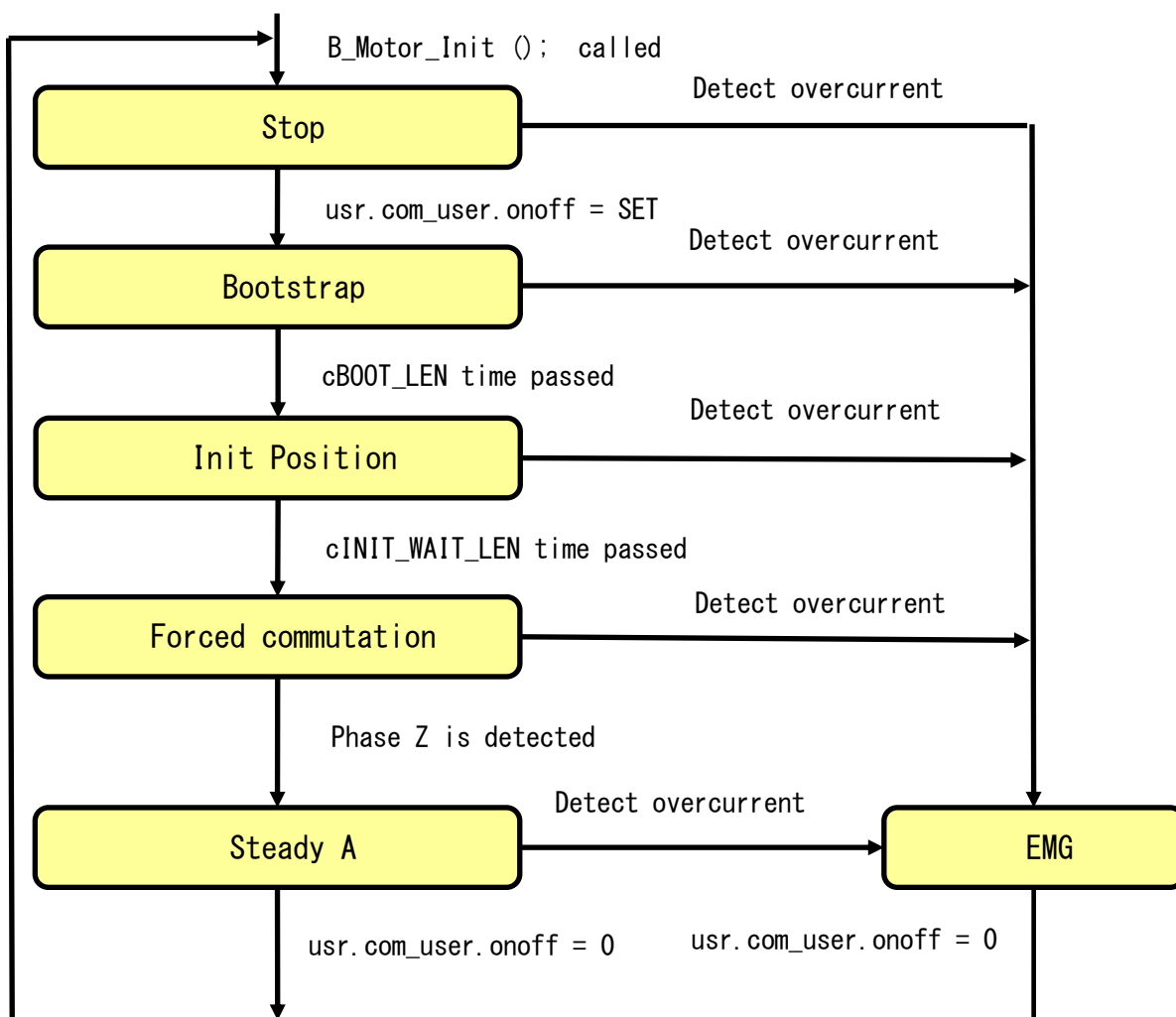


図 8.2 エンコーダー

8.3 エンコーダー+ホールセンサー

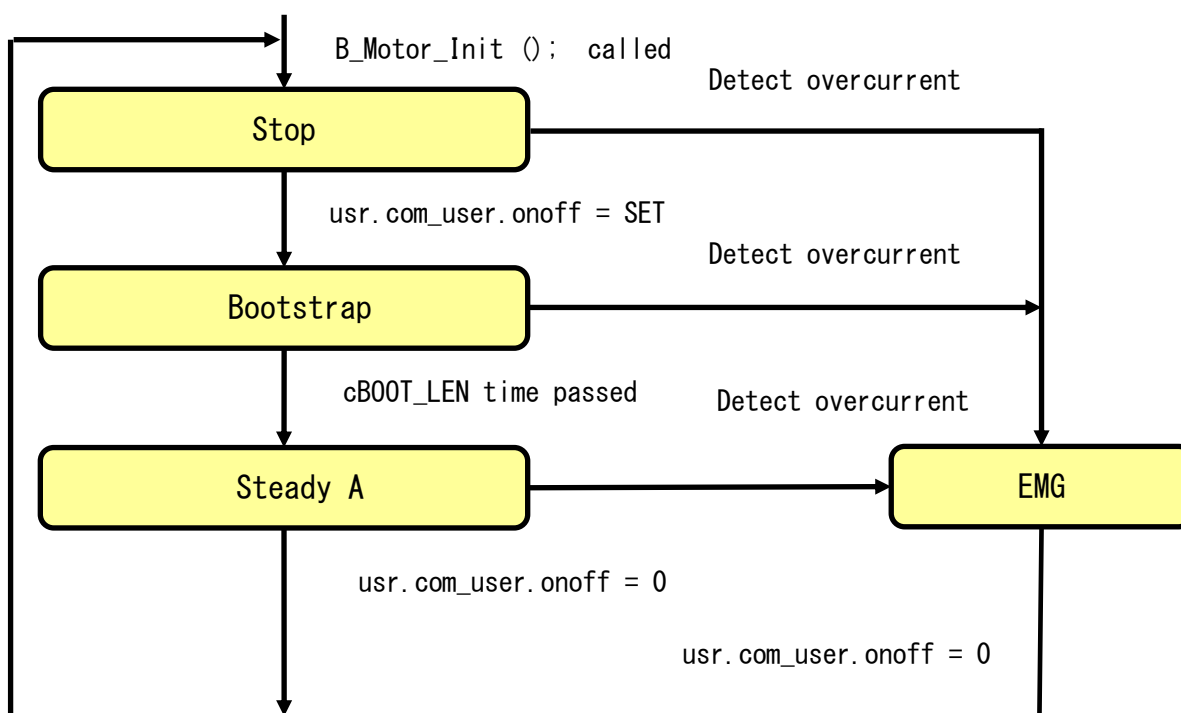


図 8.3 エンコーダー+ホールセンサー

9. ユーザーアプリ

9.1 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。

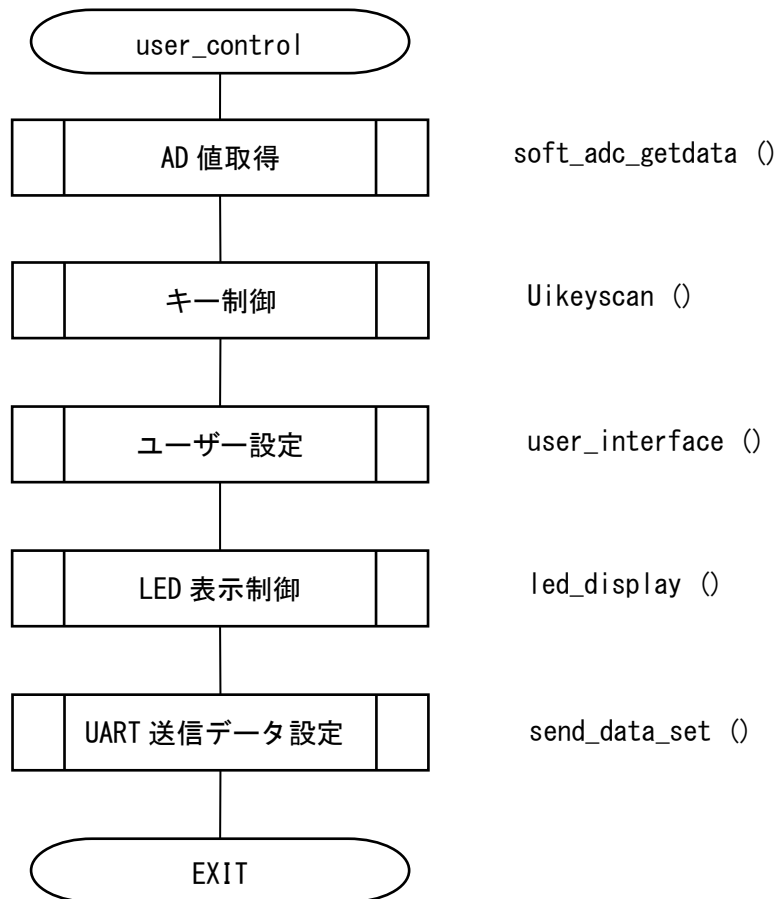


図 9.1 ユーザー制御

9.1.1 AD 値取得(soft_adc_getdata)

VR1、モーターch0 の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、次の条件によって ad_temp0.avedat、ad_vr.avedat に格納されます。

- ・ 疑似温度調整(__USE_FAKETEMP)が有効で、S_SW3 が ON の場合
VR1 の値が ad_vr.avedat に格納されます。
- ・ 疑似温度調整(__USE_FAKETEMP) が無効の場合
モーターch0 のインバーター温度が ad_temp0.avedat に格納されます。
- ・ S_SW3 が OFF の場合
VR1 の値が ad_vr.avedat に格納されます。

9.1.2 キー制御(Uikeyscan)

S_SW1、S_SW2、S_SW3、S_SW4、USW1、USW2、USW3 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、S_SW3、S_SW4 は sswdata、USW1、USW2、USW3 は uswdata に格納されます。

9.1.3 ユーザー設定(user_interface)

ユーザー操作により入力された SW に従い、下記設定を行います。

1. 回転方向切り替え

モーター停止時に処理を行う。(モーターのメインステージが停止状態時)

sswdata.bit.sw1 の値に従い、rota_dir を切り替えます。

sswdata.bit.sw1 が 1 の場合、on : CCW

sswdata.bit.sw1 が 0 の場合、off : CW

2. 回転数調整/疑似温度調整の切り替え

モーター停止時に切り替え処理を行う。(モーターのメインステージが停止状態時)

sswdata.bit.sw3 の値に従い、spd_temp_chg を切り替えます。

sswdata.bit.sw3 が 1 の場合、on : 疑似温度調整

sswdata.bit.sw3 が 0 の場合、off : 回転数調整

注意：疑似温度調整→回転数調整の切り替え時は VR1 を最小位置まで下げるまでモーターは停止状態です。

3. DAC 表示内容を更新

uswdata.bit.sw1 の値が 1 となるたびに最新の DAC 表示を行います

4. DAC No 切り替え

uswdata.bit.sw2 が 1 となるたびに dac.select の値を更新します。

5. モーター速度制御

回転数調整時、ad_vr.avedat を 8bit 分解能にし、この値が

0x10 未満 : 0Hz

0xF0 以上 : 200Hz (最大速度)

0x10～0xEF : 40Hz～199Hz

9.1.4 LED 表示制御(led_display)

EMG ステータスに従い、LED4 の制御を行います。

詳細は [4.7 ユーザーインターフェースについて](#)を参照ください

9.1.5 UART 送信データ設定(send_data_set)

モーター初期設定、現在の回転速度、EMG ステータス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、パリティなし、フロー制御なし

となります。

9.1.6 温度測定(temp_check)

VR1、TEMP0 の AD 値を取得、

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

例) Temp(12bitAD) = 1790 の時

30°C AD 判定値 = 1858(テーブル値) / 40°C AD 判定値 = 1507(テーブル値)

$1858 - 1507 = 351$ (30°C ～ 40°C 傾きは固定値)

$1858 - 1790 = 68$ (30°C からの差)

補間値 $10^{\circ}\text{C} \times 68 / 351 = 1.9^{\circ}\text{C}$ (1.9373…)

$30^{\circ}\text{C} + 1.9^{\circ}\text{C} = 31.9^{\circ}\text{C}$

・ 温度範囲

-20℃～100℃までとし、範囲外はエラーを表示します。

9.1.7 ステータス確認モニター

TeraTerm 上など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、パリティなし、フロー制御なし

- ・ 初回表示内容

リセット解除後に 1 回、下記のとおり

制御 ch = CH0

キャリア一周波数 = xxxxx Hz

デッドタイム = x.x μ s

ゲートアクティブ = H/H または L/L など

位置検出 = 1 シャント (SPWM 有/無) または 3 シャント

VDC 電圧 = xx.x V (注 1)

基板温度 = xx.x $^{\circ}$ C (注 2) (注 3)

U, V, W 電圧 = x.xx V、x.xx V、x.xx V

DAC 使用 = 有/無

内蔵アンプ使用 = 有/無

回転方向 = CW または CCW

変調方式 = 2 相変調 または 3 相変調

回転数指示 = ボリューム SW または プッシュ SW

注 1: AD (VDC) を読み込み演算

注 2: AD (VR1, TEMP0) を読み込み演算

注 3: S_SW3 が回転数調整で疑似温度定数を有効している場合「---」と表示。

- ・ 初回以降表示内容

モーター回転指示が発生後に、

制御 ch の現在の回転速度を 1 秒ごとに表示します。

例)

40Hz

58Hz

103Hz

:

・ EMG 表示内容

EMG 状態になった場合、

制御 ch の EMG 状態を 1 秒ごとに表示します。

STOP ステージ中のハード EMG の場合、EMG 状態は表示されません。

EMG_S

制御 ch = CH0

キャリア一周波数 = xxxxx Hz

デッドタイム = x.x μ s

ゲートアクティブ = H/H または L/L など

位置検出 = 1 シャント (SPWM 有/無) または 3 シャント

VDC 電圧 = xx.x V (注 1)

基板温度 = xx.x °C (注 2) (注 3)

U, V, W 電圧 = x.xx V、x.xx V、x.xx V

DAC 使用 = 有/無

内蔵アンプ使用 = 有/無

回転方向 = CW または CCW

変調方式 = 2 相変調 または 3 相変調

回転数指示 = ボリューム SW または プッシュ SW

注 1: AD (VDC) を読み込み演算

注 2: AD (VR1, TEMP0) を読み込み演算

注 3: S_SW3 が回転数調整で疑似温度定数を有効している場合「---」と表示。

10. 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。

10.1 制御コマンド

制御コマンドを以下に記します。

10.1.1 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）
- ・PWM シフトオン、オフ

(シフト 1:VE 使用時かつ 1 シャント、2 相変調のときだけ有効)

シフト 2:1 シャント、3 相変調のときだけ有効)

```
typedef struct {  
    uint16_t reserve:12; /* reserve */  
    uint16_t spwm:1;      /* PWM Shift          0=off, 1=on */  
    uint16_t modul:1;     /* PWM Modulation      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。

10.1.2 制御目標速度

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

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

10.1.3 始動電流

```
q15_t Id_st_user; /* [A/maxA] Start d-axis current by user */
```

```
q15_t Iq_st_user; /* [A/maxA] Start q-axis current by user */
```

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。

10.2 駆動コマンド

駆動コマンドを以下に記します。

10.2.1 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。

10.2.2 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。

表 10.1 ベクトル制御コマンド

cmd \ Stage	theta	omega	current	volt	Edetect	Idetect	onoff
StopcForce	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR
Bootstrap	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	SET
InitPosition	CLEAR	CLEAR	CLEAR	SET	CLEAR	SET	SET
Forced	SET	CLEAR	CLEAR	SET	SET	SET	SET
Change_up	SET	SET	CLEAR	SET	SET	SET	SET
Steady	SET	SET	SET	SET	SET	SET	SET
Emergency	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR

1) F_vcomm_theta

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

2) F_vcomm_omega

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

3) F_vcomm_current

速度制御で、d、q 軸電流の基準値の算出方法を指令します。

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、電流偏差から 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。

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。

7) F_vcomm_onoff

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

10.3 駆動状態

10.3.1 エラー状態 (drv.state)

```
typedef union {  
    struct {  
        uint16_t reserve:10;        /* reserve */  
        uint16_t Loss_sync:1;        /* 0:normal, 1: Loss of synchronism */  
        uint16_t emg_DC:1;           /* 0:normal, 1: Overvoltage (Vdc) */  
        uint16_t emg_I:1;            /* 0:normal, 1: Current detection error */  
        uint16_t emg_S:1;            /* 0:normal, 1: Overcurrent (Software) */  
        uint16_t emg_H:1;            /* 0:normal, 1: Overcurrent (Hardware) */  
        uint16_t emg_F:1;            /* 0:normal, 1: Flux observer error */  
    } flg;  
    uint16_t all;  
} 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。(未実装)
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。
emg_F	磁束オブザーバー異常検出 磁束オブザーバーが異常動作、逆転などしたときに、SET されます。(磁束オブザーバー方式時)

10.4 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ンネルごとに宣言されます。

例)

```
vector_t      Motor_ch0;      /* Motor data for ch0 */
```

10.4.1 変数一覧

表 10.2 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	DC 電圧平均値	Q31	
q31_u	drv.Vd	d 軸電圧	Q31	
q31_u	drv.Vq	q 軸電圧	Q31	
q15_t	drv.Vdq	dq 軸電圧	Q15	
q31_u	drv.Vdq_ave	dq 軸電圧平均値	Q31	
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	q 軸誘起電圧	Q15	
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	---	(10.2.1 参照)
vectorcmd_t	drv.vector_cmd	ベクトル制御コマンド	---	(10.2.2 参照)
uint16_t	drv.chkpls	電流検出保護 Duty 幅	Q0	
uint8_t	drv.idetect_error	電流検出状態	---	0: 検出可能 1: 検出不能
q15_t	drv.Ia_raw	a 相電流 (生データ)	Q15	
q15_t	drv.Ib_raw	b 相電流 (生データ)	Q15	
q15_t	drv.Ic_raw	c 相電流 (生データ)	Q15	
q15_t	drv.Ia	a 相電流 (誤検知保護)	Q15	
q15_t	drv.Ib	b 相電流 (誤検知保護)	Q15	
q15_t	drv.Ic	c 相電流 (誤検知保護)	Q15	
int32_t	drv.Iao_ave	a 相ゼロ電流平均 AD 値	Q0	
int32_t	drv.Ibo_ave	b 相ゼロ電流平均 AD 値	Q0	
int32_t	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 (生データ)	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	
uint8_t	drv.shift2_mode	シフト 2 制御モード	---	cNoShift2 cShift2_Speed cShift2_End cNormal_End
uint8_t	drv.shift2_mode_start	初期シフト 2 制御モード	---	cNoShift2 cShift2_Speed
q15_t	drv.spwm2_threshold	通常-シフト 2 切り換え速度	Q15	シフト 2 用
q15_t	drv.spwm2_threshold_hys	通常-シフト 2 切り換え速度ヒステリシス	Q15	シフト 2 用
uint8_t	drv.adtrg0_modulation_shift2	AD 変換トリガタイミング (1st)	---	シフト 2 用

uint8_t	drv. adtrg1_mod uration_shift2	AD 変換トリガタイミング (2nd)	---	シフト 2 用
uint16_t	drv. adtrg_comp _shift2	AD 変換トリガタイミング補正值	---	シフト 2 用
q15_t	drv. Ia_offset_ shift2	U 相電流オフセット	Q15	シフト 2 用
q15_t	drv. Ib_offset_ shift2	V 相電流オフセット	Q15	シフト 2 用
q15_t	drv. Ic_offset_ shift2	W 相電流オフセット	Q15	シフト 2 用
q31_u	drv. Vq_lim_shi ft2	q 軸電圧リミット値	Q31	シフト 2 用
q31_u	drv. Vd_lim_shi ft2	d 軸電圧リミット値	Q31	シフト 2 用
uint8_t	drv. shift2_cha nge	シフト 2 切り換えタイミング	---	シフト 2 用
uint8_t	drv. shift2_cha nge_bf	前回シフト 2 切換タイミング	---	シフト 2 用
uint16_t	drv. deadtime	デッドタイム	---	シフト 2 用
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_use r	初期ローター位置	Q0	
command_t	usr. com_user	制御方法	---	(10.1.1 参照)
command_t	usr. com_user_1	制御方法前回	---	
q15_t	usr. Id_st_user 2	始動 Id 電流値 2	Q15	磁束オブザーバー 方式時
q15_t	para. omega_min	強制転流終了速度	Q15	
q15_t	para. omega_v2i	電流制御切換速度	Q15	
q15_t	para. spwm_thre shold	PWM シフト切換速度	Q15	1 シャント時だけ
q15_t	para. spwm_thre shold_hys	PWM シフト切換速度ヒステリシス	Q15	1 シャント時だけ
q15_t	para. spwm2_thr eshold	PWM シフト 2 切換速度	Q15	1 シャント時だけ
q15_t	para. spwm2_thr eshold_hys	PWM シフト 2 切換速度ヒステリシス	Q15	1 シャント時だけ
uin16_t	para. spwm2_rat e_threshold_hy s	PWM シフト 2Duty 閾値ヒステリシス	Q0	シフト 2 用
q31_t	para. vd_pos	位置決め指令電圧	Q31	
q15_t	drv. Ialpha	α 軸電流	Q15	
q15_t	drv. Ibeta	β 軸電流	Q15	
q15_t	drv. Id_dev	d 軸電流偏差	Q15	
q15_t	drv. Iq_dev	q 軸電流偏差	Q15	
q31_u	drv. Vd_com	d 軸電圧指令値	Q31	
q31_u	drv. Vq_com	q 軸電圧指令値	Q31	
q31_u	drv. Vd_I	d 軸電圧積分値	Q31	

q31_u	drv. Vq_I	q 軸電圧積分値	Q31	
q31_u	drv. Valpha	α 軸電圧	Q31	
q31_u	drv. Vbeta	β 軸電圧	Q31	
q31_u	drv. Valpha_pre	前回 α 軸電圧	Q31	
q31_u	drv. Vbeta_pre	前回 β 軸電圧	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. init_pos	位置決め時間	Q0	
uint16_t	para. time. init_pos2	位置決め状態待ち時間	Q0	
uint16_t	para. time. boot_stp	ブートストラップ時間	Q0	
uint16_t	para. time. go_up	強制定常切換後待ち時間	Q0	
q31_t	para. iq_lim	q 軸電流制限値	Q31	
q31_t	para. id_lim	d 軸電流制限値	Q31	
q15_t	para. err_ovc	過電流設定値	Q15	
q31_t	para. pos. kp	位置推定比例ゲイン	Q15	Q12 対応あり
q31_t	para. pos. ki	位置推定積分ゲイン	Q15	Q12 対応あり
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	Q12 対応あり
q31_t	para. spd. ki	速度制御積分ゲイン	Q15	Q12 対応あり
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	Q12 対応あり
q31_t	para. crt. dki	d 軸電流制御積分ゲイン	Q15	Q12 対応あり
q31_t	para. crt. qkp	q 軸電流制御比例ゲイン	Q15	Q12 対応あり
q31_t	para. crt. qki	q 軸電流制御積分ゲイン	Q15	Q12 対応あり
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_adj	電気角調整	Q0	

int32_t	para. delta_lambda	強制定常切換時電流切換位相	Q0	
uint16_t	para. chkpls	電流検出保護 Duty 幅	Q0	
uint16_t	para. trgcomp0	シフト 2 トリガータイミング 0	---	
uint16_t	para. trgcomp1	シフト 2 トリガータイミング 1	---	
uint8_t	para. shift2	PWM シフト 2 センター選択	---	0:U 相センター 1:V 相センター 2:W 相センター
uint8_t	para. sensorless_method	センサーレス制御方式	---	0:誘起電圧 1:磁束オブザーバー
uint32_t	stage_counter	ステージカウンタ	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
q31_u	drv. phi_omega	磁束オブザーバー推定速度	Q31	磁束オブザーバー 方式時
q31_u	drv. phi_omega_ave	磁束オブザーバー推定速度平均値	Q31	磁束オブザーバー 方式時
uint32_u	drv. phi_tq_ave2	モーター出力トルク平均値 2	Q31	磁束オブザーバー 方式時
uint32_u	drv. theta_phi	磁束オブザーバー角度	Q0	磁束オブザーバー 方式時
uint32_u	drv. theta_phi_ie	前回磁束オブザーバー角度	Q0	磁束オブザーバー 方式時
uint32_u	drv. theta_phi2	補正後磁束オブザーバー角度	Q0	磁束オブザーバー 方式時
uint32_t	drv. theta_phi_bf	前回磁束オブザーバー角度	Q0	磁束オブザーバー 方式時
uint16_t	drv. theta_phi_rate	強制転流/センサーレス角度比重	Q0	磁束オブザーバー 方式時
uint16_t	drv. force_rate	強制転流角度割合	Q0	磁束オブザーバー 方式時
q15_t	drv. phi_omega_div	前回と今回の磁束オブザーバー速度差	Q15	磁束オブザーバー 方式時
q31_u	drv. phi_omega_bf	前回磁束オブザーバー速度	Q31	磁束オブザーバー 方式時
uint8_t	drv. phi_omega_div_emg_f	磁束オブザーバー用 EMG フラグ	---	磁束オブザーバー 方式時
q31_t	drv. Id_com2	d 軸電流指令値 2	Q31	磁束オブザーバー 方式時
q15_t	drv. DutyU	U 相 PWM Duty 比	Q15	
q15_t	drv. DutyV	V 相 PWM Duty 比	Q15	
q15_t	drv. DutyW	W 相 PWM Duty 比	Q15	
q15_t	drv. DutyU_bf	前回 U 相 PWM Duty 比	Q15	
q15_t	drv. DutyV_bf	前回 V 相 PWM Duty 比	Q15	
q15_t	drv. DutyW_bf	前回 W 相 PWM Duty 比	Q15	
float32_t	para. phi_SOGI_G	SOGI ゲイン	---	磁束オブザーバー 方式時

float32_t	para. phi. SOGI_G_low_speed	SOGI ゲイン (低速)	---	磁束オブザーバー方式時
float32_t	para. phi. SOGI_G_high	SOGI ゲイン (高速)	---	磁束オブザーバー方式時
q15_t	para. phi. SOGI_G_Change_Hz	SOGI ゲイン切り換え速度	Q15	磁束オブザーバー方式時
float32_t	para. phi. theta_phi_adj_tq_th	角度調整比率	---	磁束オブザーバー方式時
float32_t	para. phi. Id_CH_ini	低速時 Id 印加倍率	---	磁束オブザーバー方式時
q15_t	para. phi. Id_CH_ini_Hz	低速時 Id 印加速度	Q15	磁束オブザーバー方式時
float32_t	para. phi. phi_omega_div_emg	速度差エラー判定係数	---	磁束オブザーバー方式時
int32_t	para. phi. phi_CH_time	Id 印加周期	---	磁束オブザーバー方式時
uint32_u	para. phi. theta_phi_adj	角度補正值	---	磁束オブザーバー方式時
uint32_u	para. phi. theta_phi_adj_tq	位置補正+負荷トルクによるオブザーバー角度補正一定値	---	磁束オブザーバー方式時
int32_t	para. phi. phi_prd	磁束オブザーバー制御周期	---	磁束オブザーバー方式時
uint8_t	enable_theta_phi_adj	角度調整許可フラグ	---	磁束オブザーバー方式時
q15_t	para. phi. omega_phi_force_end	強制転流 2 終了速度	Q15	磁束オブザーバー方式時
detect_state_omega_e	state_omega_type	磁束オブザーバー速度用ステージ	---	Initposition_o Force_o Change_up_o Steady_F_o
detect_state_theta_e	state_theta_type	磁束オブザーバー角度用ステージ	---	Initposition_t Force_t Change_up_t Steady_F_t
phi_t	Phi	磁束オブザーバーパラメータ	---	(11.1.2.1 参照)
force2_onoff	force2	強制転流 2 使用情報	---	0 : 強制転流 2 不使用 1 : 強制転流 2 使用

表 10.3 ソフトウェアによるベクトル制御用

型	名前	意味	Q フォーマット	備考
uint32_t*	drv. ADxREG0	AD 変換結果レジスタアドレス	---	
uint32_t*	drv. ADxREG1	AD 変換結果レジスタアドレス	---	
uint32_t*	drv. ADxREG2	AD 変換結果レジスタアドレス	---	
uint32_t*	drv. ADxREG3	AD 変換結果レジスタアドレス	---	
q15_t	drv. Vdc_adc	電源電圧変換結果	Q15	
q15_t	drv. Ia_adc	U 相電流変換結果	Q15	3 シャント時だけ
q15_t	drv. Ib_adc	V 相電流変換結果	Q15	3 シャント時だけ
q15_t	drv. Ic_adc	W 相電流変換結果	Q15	3 シャント時だけ
q15_t	drv. Idc1_adc	1st 相電流変換結果	Q15	1 シャント時だけ
q15_t	drv. Idc2_adc	2nd 相電流変換結果	Q15	1 シャント時だけ
q31_u	drv. Idco_ave	ゼロ電流平均 AD 値	Q0	1 シャント時だけ
q15_t	drv. Ialpha	α 軸電流	Q15	
q15_t	drv. Ibeta	β 軸電流	Q15	
q15_t	drv. Id_dev	d 軸電流偏差	Q15	
q15_t	drv. Iq_dev	q 軸電流偏差	Q15	
q31_u	drv. Vd_com	d 軸電圧指令値	Q31	
q31_u	drv. Vq_com	q 軸電圧指令値	Q31	
q31_u	drv. Vd_I	d 軸電圧積分値	Q31	
q31_u	drv. Vq_I	q 軸電圧積分値	Q31	
q31_u	drv. Valpha	α 軸電圧	Q31	
q31_u	drv. Vbeta	β 軸電圧	Q31	
q15_t	drv. DutyU	U 相 PWM Duty 比	Q15	
q15_t	drv. DutyV	V 相 PWM Duty 比	Q15	
q15_t	drv. DutyW	W 相 PWM Duty 比	Q15	
int32_t	drv. AdTrg0	AD トリガータイミング 0 位置	Q15	
int32_t	drv. AdTrg1	AD トリガータイミング 1 位置	Q15	
uint8_t	drv. Sector	セクター	Q0	0 to 5
uint8_t	drv. Sector1	前回セクター	Q0	0 to 5
int16_t	para. TrgComp	トリガータイミング補正值	Q15	
trgpos_e	para. TrgPosMd	電流検出位置モード	---	0:3 シャント 1:1 シャント(前半) 2:1 シャント(後半)
phcvmd_e	para. PhCvMd	相変換モード	---	0:空間ベクトル 1:逆クランク(未実装)

10.5 関数詳細

10.5.1 エンコーダー初期設定 (init_ENCen)

10.5.1.1 構文

```
void init_ENCen(void)
```

引数 :

なし

戻り値 :

なし

10.5.1.2 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

10.5.2 ADC 初期設定 (init_ADCen)

10.5.2.1 構文

```
void init_ADCen(void)
```

引数 :

なし

戻り値 :

なし

10.5.2.2 処理内容

ADC の初期設定を行います。

- ・モーター用 AD 設定 (トリガー)
- ・ADC 許可

10.5.3 PMD 初期設定 (init_PMDen)

10.5.3.1 構文

```
void init_PMDen(void)
```

引数：

なし

戻り値：

なし

10.5.3.2 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。

10.5.4 VE 初期設定 (init_VEen)

10.5.4.1 構文

```
void init_VEen(void)
```

引数：

なし

戻り値：

なし

10.5.4.2 処理内容

VE の初期設定を行います。

- ・ VE 割り込み設定
- ・ VE 設定

プログラムスケジュールは下記で設定

n	CODE	タスク名	<ADC>	<END>
0	0	出力制御 1	0	0
1	1	トリガー生成	1	0
2	2	入力処理 1	0	1
3	6	SIN/COS 演算 1	0	0
4	7	出力座標軸変換(逆 Park 変換)	0	0
5	8	出力相変換 1(空間ベクトル変換)	0	0
6	0	出力制御 1	0	0
7	1	トリガー生成	1	0
8	2	入力処理 1	0	0
9	3	入力相変換(Clark 変換)	0	0
10	4	入力座標軸変換(Park 変換)	0	1
11	5	電流制御	0	0
12	6	SIN/COS 演算 1	0	0
13	7	出力座標軸変換(逆 Park 変換)	0	0
14	8	出力相変換 1(空間ベクトル変換)	0	0
15	0	出力制御 1	0	0

16	1	トリガー生成	1	0
17	2	入力処理 1	0	0
18	3	入力相変換 (Clark 変換)	0	0
19	4	入力座標軸変換 (Park 変換)	0	1
20	5	電流制御	0	0
21	6	SIN/COS 演算 1	0	0
22	7	出力座標軸変換 (逆 Park 変換)	0	0
23	8	出力相変換 1 (空間ベクトル変換)	0	0
24	9	出力制御 2	0	0
25	1	トリガー生成	1	0
26	10	入力処理 2	0	0
27	3	入力相変換 (Clark 変換)	0	0
28	4	入力座標軸変換 (Park 変換)	0	1
29	–	–	–	–
30	–	–	–	–
31	–	–	–	–

10.5.5 シフト 2 初期設定 (init_Shift2_CHx)

10.5.5.1 構文

```
void init_Shift2_CHx(void)
```

引数：

なし

戻り値：

なし

10.5.5.2 処理内容

PWM シフト 2 モードの初期設定を行います。

- AD 設定
- VE 設定 (VE 設定時)
- PMD 設定

10.5.6 コンパレータ初期設定 (init_COMPen)

10.5.6.1 構文

```
void init_COMPen(void)
```

引数：

なし

戻り値：

なし

10.5.6.2 処理内容

過電流検出用の内蔵コンパレーターの初期設定を行います。

- COMP 設定
- DAC 設定 (正電流/負電流の基準電圧)

10.5.7 モーター制御初期設定 (B_Motor_Init)

10.5.7.1 構文

```
void B_Motor_Init(void)
```

引数 :

なし

戻り値 :

なし

10.5.7.2 変数

方向	名前	意味	Q フォーマット	備考
入出力	drv.shift2_mode_start	初期シフト 2 制御モード	---	
	usr.Id_st_user2	始動 Id 電流値 2	Q15	磁束オブザーバー方式時
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.lao_ave	U 相ゼロ電流平均 AD 値	Q0	
	drv.lbo_ave	V 相ゼロ電流平均 AD 値	Q0	
	drv.lco_ave	W 相ゼロ電流平均 AD 値	Q0	
	drv.Idco_ave	ゼロ電流変換結果	Q15	1 シャント時だけ
	usr.lq_st_user	始動 lq 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	PWM シフト切換速度	Q15	1 シャント時だけ
	para.spwm_threshold_hys	PWM シフト切換速度ヒステリシス	Q15	1 シャント時だけ
	para.chkpls	電流検出保護 Duty 幅	Q0	
	para.vd_pos	位置決め指令電圧	Q31	電圧起動時だけ
	para.spd_coef	出力電圧係数	Q15	電圧起動時だけ
	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
	para.sp_up_lim_s	駆動速度増加リミット値	Q31	
	para.sp_dn_lim_s	駆動速度減少リミット値	Q31	
	para.time.bootstp	ブートストラップ時間	Q0	
	para.time.initpos	位置決め時間	Q0	
	para.time.initpos2	位置決め状態待ち時間	Q0	
	para.time.go_up	強制定常切換後待ち時間	Q0	

para. omega_min	強制転流終了速度	Q15	
para. omega_v2i	電流制御切換速度	Q15	
para. delta_lambda	強制定常切換時電流切換位相	Q0	
para. pos.ki	位置推定積分ゲイン	Q15	Q12 対応あり
para. pos.kp	位置推定比例ゲイン	Q15	Q12 対応あり
para. pos.ctr_lprd	位置推定制御周期	Q16	
para. spd.ki	速度制御積分ゲイン	Q15	Q12 対応あり
para. spd.kp	速度制御比例ゲイン	Q15	Q12 対応あり
para. current.dki	d 軸電流比例ゲイン	Q15	Q12 対応あり
para. current.dkp	d 軸電流積分ゲイン	Q15	Q12 対応あり
para. current.qki	q 軸電流比例ゲイン	Q15	Q12 対応あり
para. current.qkp	q 軸電流積分ゲイン	Q15	Q12 対応あり
para. iq_lim	q 軸電流制限値	Q31	
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.ctr_lprd	エンコーダー制御周期	Q16	
para. shift2	シフト 2 センター相設定	---	
para. spwm2_threshold	通常-シフト 2 切り換え速度	Q15	シフト 2 用
para. spwm2_threshold_hys	通常-シフト 2 切り換え速度ヒステリシス	Q15	シフト 2 用
drv. la_offset_shift2	U 相電流オフセット	Q15	シフト 2 用
drv. lb_offset_shift2	V 相電流オフセット	Q15	シフト 2 用
drv. lc_offset_shift2	W 相電流オフセット	Q15	シフト 2 用
usr.com_user.modul	変調方式制御コマンド	---	2 相 or 3 相変調
drv.adtrg0_moduration_shift2	AD 変換トリガータイミング(1st)	---	シフト 2 用
drv.adtrg1_moduration_shift2	AD 変換トリガータイミング(2nd)	---	シフト 2 用
drv.adtrg_comp_shift2	AD 変換トリガータイミング補正值	---	シフト 2 用
drv.deadtime	デッドタイム	---	
drv.Vq_lim_shift2	q 軸電圧リミット値	Q31	シフト 2 用
drv.Vd_lim_shift2	d 軸電圧リミット値	Q31	シフト 2 用
para.TrgPosMd	電流検出位置モード	---	
para.TrgComp	トリガータイミング補正值	Q15	
detect_type	位置推定方式	---	磁束オブザーバー方式時
para.motor.Pole_pare	モーター極数	---	磁束オブザーバー方式時
para.phi.phi_omega_div_emg	磁束オブザーバー用速度 EMG 判定閾値	Q0	磁束オブザーバー方式時
para.phi.phi_CH_time	Id 印加周期	---	磁束オブザーバー方式時
para.phi.SOGI_G	SOGI 制御ゲイン	Q31	磁束オブザーバー方式時
para.phi.SOGI_G_low_speed	SOGI ゲイン(低速)	---	磁束オブザーバー方式時
para.phi.SOGI_G_high	SOGI ゲイン(高速)	---	磁束オブザーバー方式時
para.phi.SOGI_G_Change_Hz	SOGI ゲイン切り換え速度	Q15	磁束オブザーバー方式時
para.phi.theta_phi_adj_tq_th	角度調整比率	---	磁束オブザーバー方式時
enable_theta_phi_adj	角度調整許可フラグ	---	磁束オブザーバー方式時

para.phi.Id_CH_ini	低速時 Id 印加倍率	---	磁束オブザーバー方式時
para.phi.Id_CH_ini_Hz	低速時 Id 印加速度	Q15	磁束オブザーバー方式時
para.phi.omega_phi_force_end	強制転流 2 終了速度	Q15	磁束オブザーバー方式時
para.phi.phi_prd	磁束オブザーバー制御周期	Q0	磁束オブザーバー方式時
drv.Id_com2	d 軸電流指令値 2	Q31	磁束オブザーバー方式時
force2	強制転流 2 使用状態	---	強制転流 2 使用 or 不使用

10.5.7.3 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。

10.5.8 DAC 制御初期設定 (init_Dac)

10.5.8.1 構文

```
void init_Dac(TSB_TSPI_TypeDef* const TSPIx)
```

引数：

なし

戻り値：

なし

10.5.8.2 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

10.5.9 周期タイマー初期設定 (init_Timer_interval4kH)

10.5.9.1 構文

```
void init_Timer_interval4kH(void)
```

引数 :

なし

戻り値 :

なし

10.5.9.2 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。

10.5.10 ユーザー制御初期設定 (init_user_control)

10.5.10.1 構文

```
void init_user_control(void)
```

引数 :

なし

戻り値 :

なし

10.5.10.2 処理内容

ユーザーなどによる外部からの指令を検出するため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 初期設定 (init_led)

UART 初期設定 (init_uart)

10.5.11 ユーザー制御 (user_control)

10.5.11.1 構文

```
void user_control(void)
```

引数：

なし

戻り値：

なし

10.5.11.2 処理内容

ユーザーなどによる外部からの指令を検出し、指令に従って制御を行います。

- ・ S_SW1 による、回転方向切換
- ・ S_SW3 による、回転数調整/疑似温度調整の切換
- ・ USW1 による、DAC 表示内容更新
- ・ USW2 による、DAC No 切り替え
- ・ VR1 による、回転数調整/疑似温度調整
- ・ 設定や回転速度情報を UART 出力
- ・ 内部状態を LED 表示

注意：S_SW の切り替えはモーター動作を停止してからでないと有効になりません。

- ・ アナログ入力電圧取得

回転数調整/疑似温度調整 (VR1)、モーターch0 のインバーター温度 (TEMP0) のアナログ電圧値を取得し平均処理を行います。

- ・ デジタルポート入力

キーのチャタリング処理を行います。

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

10.5.12.1 構文

```
void B_User_MotorControl(void)
```

引数：

なし

戻り値：

なし

10.5.12.2 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
	sswdata.sw	S_SWON, OFF 状態	---	ユーザー指令
	uswdata.sw	USW の ON, OFF 状態	---	ユーザー指令
出力	usr.omega_user	制御目標速度	Q31	
	usr.com_user.onoff	モーター ON/OFF	---	
	drv.state	モーター異常ステータス	---	
	Detect_type	位置推定方式	---	磁束オブザーバー方式時
	usr.lamsla_user	初期ローター位置	Q0	ホールセンサー+エンコーダー制御時

10.5.12.3 処理内容

モーター関連に関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

キーの状態からモーター ON/OFF(usr.com_user.onoff)を決定します。

- ・駆動速度正規化など

目標速度を正規化します。

10.6 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り実現されます。モーター動作を、停止状態、ブートストラップ状態、位置決め状態、強制転流状態、強制定常切換状態、定常状態、短絡ブレーキ状態、保護状態間の状態遷移として制御します。

メインステージはモーター動作開始指示で位置決め (Initpositon)、強制転流 (Force)、強制定常切換 (Change_up)、定常 (Steady_A) の順に移行します。サブステージは Step0 から StepEnd まであり、StepEnd の時にメインステージが移行し Step0 から始まります。またモーターの異常を検出した場合は保護停止 Emergency に移行します。

10.6.1 状態遷移処理関数 (C_Control_Ref_Model)

10.6.1.1 構文

```
void C_Control_Ref_Model(vector_t* const _motor)          • ENC_Z 状態遷移なし
void C_Control_Ref_Model_ENC(vector_t* const _motor)      • ENC_Z 状態遷移あり
```

引数：

_motor モーター制御構造体

戻り値：

なし

10.6.1.2 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	---	
	drv.Z_detect	Z パルス検知	---	ENC_Z 状態遷移時のみ使用
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	---	
出力	drv.phi_omega_div_emg_f	磁束オブザーバー用 EMG フラグ	---	磁束オブザーバー方式時

10.6.1.3 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。

ENC_Z 状態遷移ありの場合、Z パルス検知で強制転流から定常に遷移します。

※ ENC_Z 状態遷移は、エンコーダーが有効で 3 シャントの場合に行います。

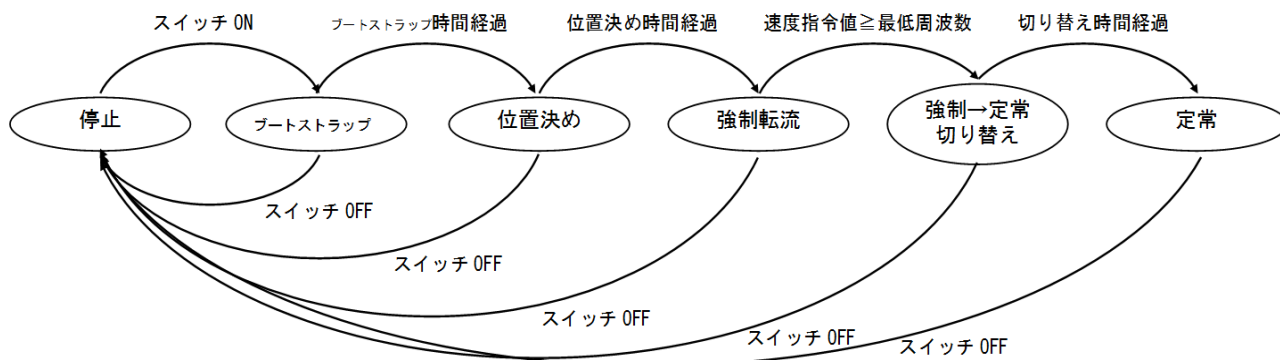


図 10.1 モーター制御の状態遷移図

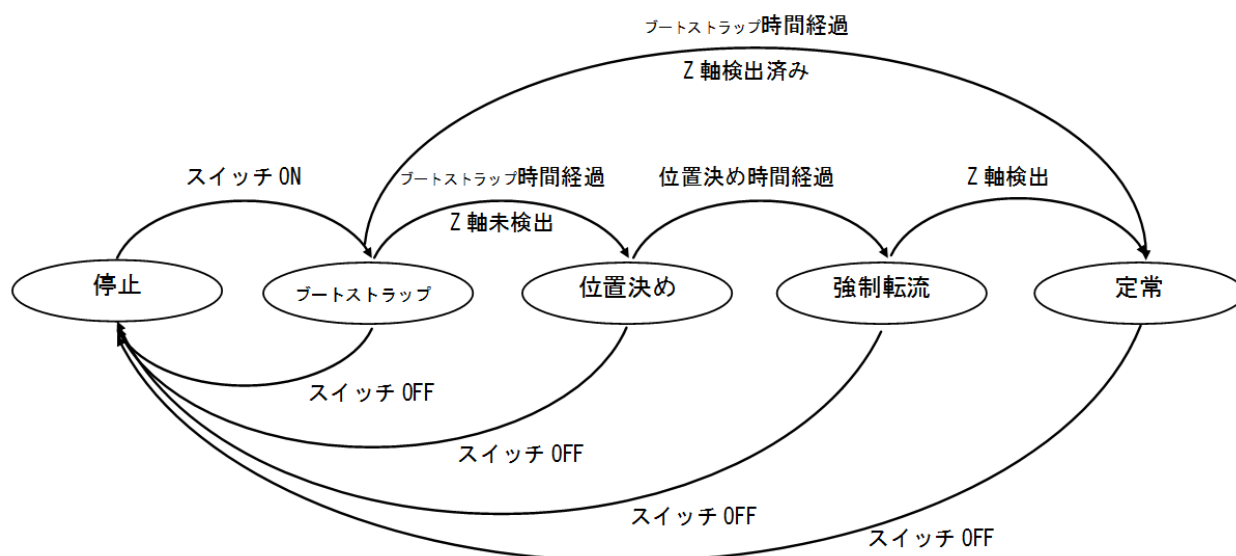


図 10.2 モーター制御の状態遷移図 (エンコーダー制御)

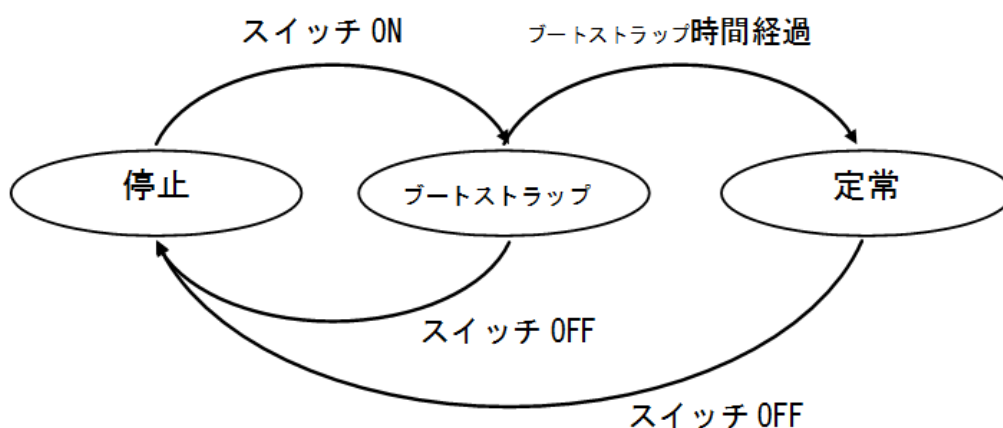


図 10.3 モーター制御の状態遷移図 (ホールセンサー+エンコーダー制御)

10.6.2 モーター制御共通処理関数 (C_Common)

10.6.2.1 構文

```
void C_Common(vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

なし

10.6.2.2 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 or 3相変調
	para.chkpls	電流検出保護 Duty 幅設定値	---	
	drv.Vd	d 軸電圧	Q32	
	drv.Vq	q 軸電圧	Q32	
	drv.Vdc	電源電圧	Q15	
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 or 3相変調
	drv.chkpls	電流検出保護 Duty 幅	---	
	drv.Vdq_per	電源電圧に対する dq 軸電圧の割合	---	[%]

10.6.2.3 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。

PWM シフト制御(C_ShiftPWM_Control)を行います。

Vdq 演算 (Cal_Vdq) を行い、Vdc に対する Vdq の割合を算出します。

10.6.3 停止状態関数 (C_Stage_Stop)

10.6.3.1 構文

```
void C_Stage_Stop(vector_t* const _motor)
```

引数：

_motor モーター制御構造体

戻り値：

なし

10.6.3.2 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	---	
	drv.shift2_mode_start	初期シフト 2 制御モード	---	
	Phi	磁束オブザーバー構造体	---	磁束オブザーバー方式時
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	---	(10.2.2 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.Vd_com	d 軸電圧指令値	Q31	ソフトウェアベクトル制御のみ
	drv.Vq_com	q 軸電圧指令値	Q31	ソフトウェアベクトル制御のみ
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	
	drv.theta_phi_rate	強制転流/センサーレス角度比重	Q0	磁束オブザーバー方式時
	drv.force_rate	強制転流角度割合	Q0	磁束オブザーバー方式時
	drv.phi_tq_ave2	モーター出力トルク平均値 2	Q31	磁束オブザーバー方式時
	drv.phi_omega_ave	磁束オブザーバー推定速度平均値	Q31	磁束オブザーバー方式時
	drv.shift2_change	シフト 2 切り換えタイミング	---	シフト 2 用
	drv.Vd_I	d 軸電圧積分値	Q31	
	drv.Vq_I	q 軸電圧積分値	Q31	

10.6.3.3 処理内容

モーターを停止させます。(PWM 出力を停止します)

10.6.4 ブートストラップ状態関数 (C_Stage_Bootstrap)

10.6.4.1 構文

```
void C_Stage_Bootstrap(vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

なし

10.6.4.2 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	---	(10.2.2 参照)

10.6.4.3 処理内容

上相 A11 OFF、下相 A11 ON の波形を出力し、ブートストラップコンデンサの充電を行います。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。

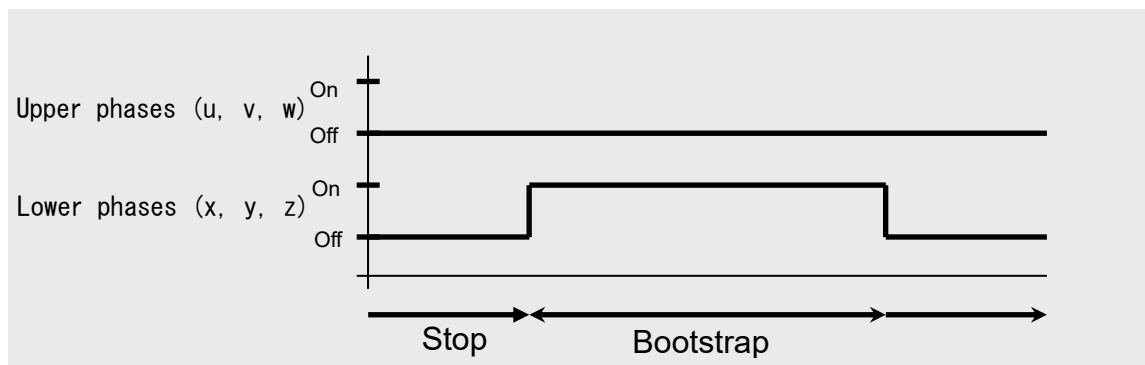


図 10.4 ブートストラップ時間

10.6.5 位置決め状態関数 (C_Stage_Initposition)

10.6.5.1 構文

```
void C_Stage_Initposition(vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

 なし

10.6.5.2 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	
	drv.shift2_mode_start	初期シフト 2 制御モード	---	シフト 2 用
	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	---	(10.2.2 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	
	drv.shift2_mode	シフト 2 制御モード	---	
	usr.com_user.modul	変調方式制御コマンド	---	2 相 or 3 相変調

10.6.5.3 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、 I_q を 0 に固定しながら、 I_d 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 I_d 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 I_d の増加量をあらかじめ決定しておきます。 I_d 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。

初期シフト 2 制御モードがシフト 2 モード設定の場合は PWM シフト 2 用の初期設定を行います。

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。

10.6.6 強制転流状態関数 (C_Stage_Force)

10.6.6.1 構文

```
void C_Stage_Force(vector_t* const _motor)
```

引数：

_motor モーター制御構造体

戻り値：

なし

10.6.6.2 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	---	(10.2.2 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

10.6.6.3 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従して回転します。

I_d を「始動 I_d 電流値」に、 I_q 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」に達するまで続けます。

「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。「推定速度」が ω になります。

10.6.7 強制定常切換状態関数 (C_Stage_Change_up)

10.6.7.1 構文

```
void C_Stage_Change_up(vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

なし

10.6.7.2 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	強制定常切換時電流切換位相	Q0	
	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
	para.time.go_up	強制定常切換後待ち時間	Q0	* MainLoopPrd(s)
	drv.Id_com2	d 軸電流指令値 2	Q31	磁束オブザーバー方式時
入出力	stage.sub	サブステージ	---	
	stage_counter	ステージカウンタ	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	---	(10.2.2 参照)
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	

10.6.7.3 処理内容

Id を 0 に減少、Iq を「始動 Iq 電流値」まで増加させ、磁界の方向がローターと直角になるように制御します。

つまり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。「駆動速度指令値」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、「推定速度」は制御には使用されません。

強制定常切換状態を以下のサブ状態に分けて制御します。

a) 初期状態

強制定常切換状態の初期設定を行います。

b) Id、Iq 切り換え状態

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時間が経過するのを待ち、定常状態へ遷移します。

10.6.8 定常状態関数 (C_Stage_Steady_A)

10.6.8.1 構文

```
void C_Stage_Steady_A(vector_t* const _motor)
```

引数：

_motor モーター制御構造体

戻り値：

なし

10.6.8.2 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	
	drv.shift2_mode_start	初期シフト 2 制御モード	Q0	シフト 2 用
	para.spwm2_threshold	通常-シフト 2 切り換え速度	Q15	
	para.spwm2_threshold_hys	通常-シフト 2 切り換え速度ヒステリシス	Q15	シフト 2 用
	detect_type	位置推定方式	---	磁束オブザーバー方式時
	para.phi.Id_CH_ini	低速時 Id 印加倍率	---	磁束オブザーバー方式時
	para.phi.Id_CH_ini_Hz	低速時 Id 印加速度	Q15	磁束オブザーバー方式時
	para.phi.SOGI_G_low_speed	SOGI ゲイン(低速)	---	磁束オブザーバー方式時
	para.phi.SOGI_G_high	SOGI ゲイン(高速)	---	磁束オブザーバー方式時
	para.omega_min	強制転流終了速度	Q15	磁束オブザーバー方式時
	para.omega_min2	定常強制転流切り換え速度	Q15	磁束オブザーバー方式時
	drv.Id_com2	d 軸電流指令値 2	Q31	磁束オブザーバー方式時
	usr.Id_st_user2	開始電流	Q15	磁束オブザーバー方式時
	usr.lam1xla_user	初期ローター位置	Q0	ホールセンサー+エンコーダー制御時
入出力	stage.sub	サブステージ	---	
	drv.omega_com	駆動速度指令値	Q31	
	para.phi.SOGI_G	SOGI 制御ゲイン	Q31	磁束オブザーバー方式時

	drv.shift2_mode	シフト 2 制御モード	---	シフト 2 用
出力	drv.vector_cmd	ベクトル制御コマンド	Q0	(10.2.2 参照)
	stage.itr	割り込みステージ	---	
	drv.Id_com	d 軸電流指令値	Q31	
	drv.shift2_change	シフト 2 切り換えタイミング	---	シフト 2 用

10.6.8.3 処理内容

定常状態の処理を実行します。

「駆動速度指令値」を一定値ずつ増加させて「制御目標速度」に近づけます。

(1 シャント)

駆動速度指令値 > (通常-シフト 2 切り換え速度) ならば通常 PWM、駆動速度指令値 ≤ (通常-シフト 2 切り換え速度) ならば PWM シフト 2 で駆動します。

注意：通常 PWM と PWM シフトの切り換えについては PWM シフト制御 (C_ShiftPWM_Control) の説明を参照

10.6.9 保護状態関数 (C_Stage_Emergency)

10.6.9.1 構文

```
void C_Stage_Emergency(vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

 なし

10.6.9.2 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	---	(10.2.2 参照)
	stage.itr	割り込みステージ	---	

10.6.9.3 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 となります。

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、全てオフとなります。

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。

10.6.10 PWM シフト制御 (C_ShiftPWM_Control)

10.6.10.1 構文

```
static void C_ShiftPWM_Control(vector_t* const _motor)
```

引数:

<code>_motor</code>	モーター制御構造体
---------------------	-----------

戻り値:

なし

10.6.10.2 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	PWM シフト制御方法	---	
	drv.shift2_mode	シフト 2 制御モード	---	
	drv.omega_com	駆動速度指令値	Q31	
	para.spwm_threshold	PWM シフト切換速度	Q15	
	para.spwm_threshold_hys	PWM シフト切換速度ヒステリシス	Q15	
出力	drv.command.spwm	PWM シフト駆動コマンド	---	

10.6.10.3 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。

PWM シフトの ON/OFF の設定を行います。

PWM シフト制御方法、割り込みステージ、目標速度により PWM シフト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 10.4 PWM シフト駆動方法条件(切り換え設定: 速度)

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	PWM シフト駆動方法 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)
		drv.shift2_mode == cNoShift2	
		目標速度 < シフト切換速度 drv.omega_com < para.spwm_threshold - para.spwm_threshold_hys drv.shift2_mode != cNoShift2	ON (1)

10.7 モーター駆動関数

10.7.1 用語説明

10.7.1.1 3 相変調の場合

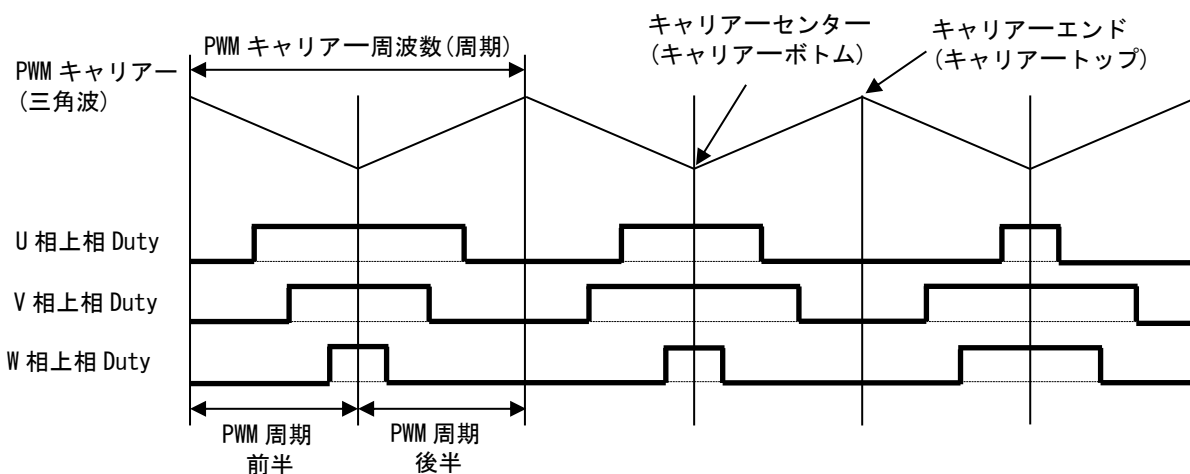


図 10.5 3 相変調の場合

10.7.1.2 2 相変調の場合

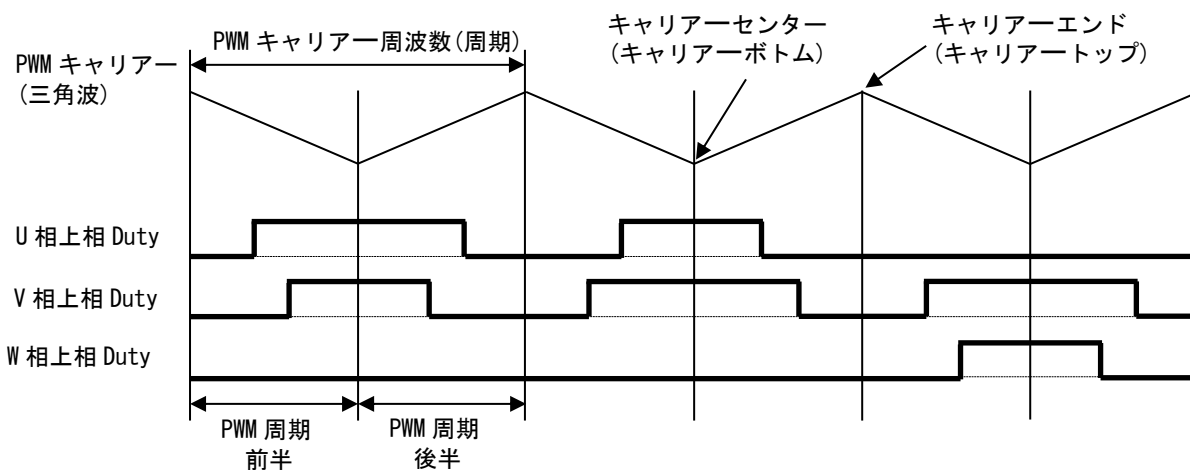


図 10.6 2 相変調の場合

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

10.7.2.1 構文

```
void D_GetMotorCurrentPowerVolt (vector_t* const _motor)
```

引数：

_motor モーター制御構造体

戻り値：

なし

10.7.2.2 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 to 5
	para.TrgPosMd	電流検出位置モード	---	3 シャント 1 シャント前半 1 シャント後半
	drv.ADxREG0	AD 変換結果レジスターアドレス	---	
	drv.ADxREG1	AD 変換結果レジスターアドレス	---	
	drv.ADxREG2	AD 変換結果レジスターアドレス	---	
	drv.ADxREG3	AD 変換結果レジスターアドレス	---	
	drv.Iao_ave	U 相ゼロ電流平均	Q15	3 シャント時だけ
	drv.Ibo_ave	V 相ゼロ電流平均	Q15	3 シャント時だけ
	drv.Ico_ave	W 相ゼロ電流平均	Q15	3 シャント時だけ
	drv.Idco_ave	ゼロ電流平均	Q15	1 シャント時だけ
	stage.itr	割り込みステージ	---	
	drv.idetect_error	電流検出状態	---	
	drv.shift2_mode	シフト 2 制御モード	---	シフト 2 用
	drv.shift2_change_bf	前回シフト 2 切換タイミング	---	シフト 2 用
入出力	drv.Ia_raw	U 相電流 (生データ)	Q15	
	drv.Ib_raw	V 相電流 (生データ)	Q15	
	drv.Ic_raw	W 相電流 (生データ)	Q15	
出力	drv.Ia	U 相電流 (誤検知保護)	Q15	
	drv.Ib	V 相電流 (誤検知保護)	Q15	
	drv.Ic	W 相電流 (誤検知保護)	Q15	
	drv.Ia_adc	U 相電流変換結果	Q15	3 シャント時だけ
	drv.Ib_adc	V 相電流変換結果	Q15	3 シャント時だけ
	drv.Ic_adc	W 相電流変換結果	Q15	3 シャント時だけ
	drv.Idc1_adc	1st 相電流変換結果	Q15	1 シャント時だけ
	drv.Idc2_adc	2nd 相電流変換結果	Q15	1 シャント時だけ
	drv.Vdc	電源電圧	Q15	
	drv.Vdc_adc	電源電圧変換結果	Q15	

10.7.2.3 処理内容

AD 変換結果値を取得し、電源電圧とモーター電流を演算します。

1. PMD トリガーによる AD 変換の終了を確認し、変換が終了していれば下記表に従って変換結果を取得する。

変換終了していない場合は、変換が終わるまでループで待つ(通常ではあり得ないがフェールセーフ処理で待つ処理を行う)。

ただし、1 シャントにおいて PWM シフト 2 で動作している場合はオフセットを加えて算出し、通常 PWM/PWM シフト 2 の切り換えたタイミングでは取得・演算は行わない

2. 取得した AD 変換結果から電源電圧、モーター電流を演算する。

● 電源電圧

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

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

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

```
drv.Vdc_adc = drv.Vdc_adc
```

● モーター電流

<電流検出モード : 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 drv.Ib_adc	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	AD 変換結果 1 ADxREG1	AD 変換結果 0 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、下記変数に格納する。

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する。

	セクター					
	0	1	2	3	4	5
U相電流 drv. Ia	$-I_b - I_c$	$I_{ao_ave} - I_{a_adc}$	$I_{ao_ave} - I_{a_adc}$	$I_{ao_ave} - I_{a_adc}$	$I_{ao_ave} - I_{a_adc}$	$-I_b - I_c$
V相電流 drv. Ib	$I_{bo_ave} - I_{b_adc}$	$-I_a - I_c$	$-I_a - I_c$	$I_{bo_ave} - I_{b_adc}$	$I_{bo_ave} - I_{b_adc}$	$I_{bo_ave} - I_{b_adc}$
W相電流 drv. Ic	$I_{co_ave} - I_{c_adc}$	$I_{co_ave} - I_{c_adc}$	$I_{co_ave} - I_{c_adc}$	$-I_a - I_b$	$-I_a - I_b$	$I_{co_ave} - I_{c_adc}$

〈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、下記変数に格納する。

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

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

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

10.7.3 入力座標変換 (D_InputTransformation)

10.7.3.1 構文

```
void D_InputTransformation(vector_t* const _motor)
```

引数：

`_motor` モーター制御構造体

戻り値：

なし

10.7.3.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_Idetect	電流検出コマンド	---	
	drv.Ia	U 相電流	Q15	
	drv.Ib	V 相電流	Q15	
	(drv.Ic)	W 相電流	Q15	
	drv.theta	ローター位置	Q0	[deg/360]
	para.pos.ctr_lprd	ベクトル制御周期	Q0	
	drv.omega	推定速度	Q15	
	drv.idetect_error	電流検出状態	---	0:検出可能 1:検出不能
	drv.shift2_change_bf	前回シフト2切り換えタイミング	---	シフト2用
	drv.shift2_mode	シフト2制御モード	---	シフト2用
出力	drv.Ialpha	α 軸電流	Q15	
	drv.Ibeta	β 軸電流	Q15	
	drv.Id	d 軸電流	Q15	
	drv.Iq	q 軸電流	Q15	

10.7.3.3 処理内容

3 相の電流(Ia, Ib, Ic)から、dq 軸電流に変換します。

- 10.7.4 クラーク変換により、3 相の電流(Ia, Ib, Ic)から 2 相の電流(I_α, I_β)に変換します。
- 10.7.5 パーク変換により、静止座標である 2 相の電流(I_α, I_β)から回転座標の dq 軸変換に変換します。

パーク変換に使用するローター位置は、位置推定処理で演算した時点から実際のローター位置は進んでいるため、速度 ω と演算周期から予測した θ_{est} を使用します。 $\theta_{est} = \theta + \text{速度}\omega \times \text{ベクトル制御周期}$

なお、電流検出が正常にできない状態のとき、または通常 PWM と PWM シフト 2 の切り換えのときは、変換を行わず、Id, Iq は更新しません。

10.7.4 クラーク変換 (E_Clarke)

10.7.4.1 構文

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

引数：

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

戻り値：

なし

10.7.4.2 処理内容

3 相の電流(I_u , I_v , I_w)を 2 相(I_α , I_β)に変換します。

下記演算式により I_α 、 I_β を演算します。

$$I_\alpha = \frac{2}{3} \times (I_u \times \cos 0 + I_v \times \cos 120 + I_w \times \cos 240) = \frac{2}{3} \times \left(I_u - \frac{1}{2} I_v - \frac{1}{2} I_w \right)$$

$$I_\beta = \frac{2}{3} \times (I_u \times \sin 0 + I_v \times \sin 120 + I_w \times \sin 240) = \frac{2}{3} \times \left(\frac{\sqrt{3}}{2} I_v - \frac{\sqrt{3}}{2} I_w \right)$$

係数 $\frac{2}{3}$ は、3 相電流を 2 相の α β 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、 $I_u + I_v + I_w = 0$ とすると、

$$I_\alpha = I_u$$

$$I_\beta = (I_u + 2I_v)/\sqrt{3}$$

となります。

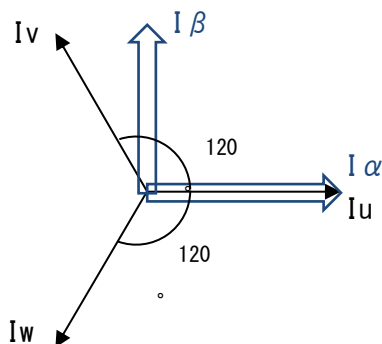


図 10.7 クラーク変換

10.7.5 パーク変換 (E_Park)

10.7.5.1 構文

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

引数：

_ialpha	α 軸電流 I_α
_ibeta	β 軸電流 I_β
_theta	ローター位置 θ : $0 \leq \text{位置} < 360^\circ$ (0 to 0xffff)
_id	d 軸電流 I_d 格納アドレス
_iq	q 軸電流 I_q 格納アドレス

戻り値：

なし

10.7.5.2 処理内容

α β 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 I_d 、 I_q を演算します。

$$I_d = I_\alpha \times \cos\theta + I_\beta \times \sin\theta$$

$$I_q = I_\alpha \times (-\sin\theta) + I_\beta \times \cos\theta$$

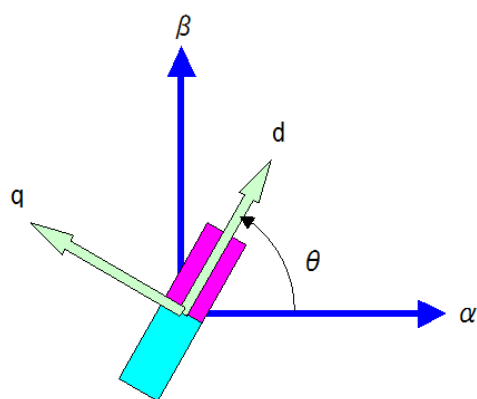


図 10.8 パーク変換

10.7.6 誘起電圧位置推定関数 (D_Detect_Rotor_Position)

10.7.6.1 構文

```
void D_Detect_Rotor_Position(vector_t* const _motor)
```

引数：

_motor モーター制御構造体

戻り値：

なし

10.7.6.2 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.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	Q12 対応あり
	para.pos.kp	位置推定比例ゲイン	Q15	Q12 対応あり
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	

10.7.6.3 処理内容

操作量がモーター駆動信号の推定速度 ω_{est} 、制御量が d 軸誘起電圧 E_d として PI 制御を行います。

なお E_d の目標値は常に 0 です。つまり偏差は $-E_d$ になります。

PI 制御より求めた推定速度 ω_{est} を積分することで、ローター位置 θ （角度）を求めます。

推定速度 ω は、F_vcomm_omega が CLEAR の場合は、駆動速度指令値 ω_{com} を推定速度 ω として使用します。

モーターの d 軸に関する等価回路方程式は、下記で表すことができます。

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

($p=d/dt$, $I_d \approx$ 一定値より $pI_d=0$ とすることができる。)

V_d : モーター印加電圧

I_d, I_q : モーター電流

ω_{est} : 推定角速度

R : 抵抗

L_d, L_q : インダクタンス

よって、d 軸誘起電圧 E_d は下記演算式で求めることができます。

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

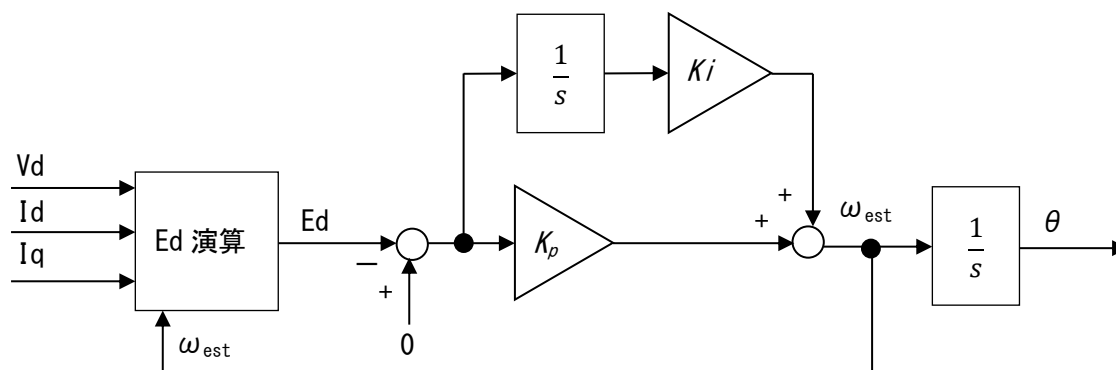


図 10.9 d 軸誘起電圧 E_d による位置推定ブロック図

10.7.7 磁束オブザーバー位置推定関数 (D_Detect_Flux_Rotor_Position)

10.7.7.1 構文

```
void D_Detect_Flux_Rotor_Position (vector_t* const _motor)
```

引数:

`_motor`

モーター制御構造体

戻り値:

なし

10.7.7.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv. idetect_error	電流検出状態	---	0: 検出可能 1: 検出不能
	drv. shift2_change_bf	前回シフト 2 切り換えタイミング	---	
	drv. omega_com	駆動速度指令値	Q31	
	para. phi. phi_prd	磁束オブザーバー制御周期	Q0	
	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. command. encoder	エンコーダー駆動コマンド	---	
	para. phi. phi_omega_div_emg	磁束オブザーバー用速度 EMG 判定閾値	Q0	
	drv. vector_cmd. F_vcomm_theta	推定ローター位置制御コマンド	---	
	para. phi. enable_theta_phi_adj	角度調整許可フラグ	---	
	para. phi. theta_phi_adj_tq_th	角度調整比率	---	
	para. phi. omega_phi_force_end	強制転流 2 終了速度	Q15	
入出力	drv. phi_omega_bf	前回磁束オブザーバー速度	Q31	
	drv. theta_phi	磁束オブザーバー角度	Q0	
	drv. phi_omega	磁束オブザーバー推定速度	Q31	
	drv. phi_omega_ave	磁束オブザーバー推定速度平均値	Q31	
	drv. phi_omega_div	前回と今回の磁束オブザーバー速度差	Q15	
	drv. phi_tq_ave2	モーター出力トルク平均値 2	Q31	
	drv. theta_phi2	補正後磁束オブザーバー角度	Q0	
	drv. theta_phi_rate	強制転流/センサーレス角度比重	Q0	
	drv. force_rate	強制転流角度割合	Q0	
	drv. theta_phi_bf	前回磁束オブザーバー角度	Q0	
出力	drv. theta_phi_ie	前回磁束オブザーバー角度	Q0	
	drv. omega	推定速度	Q31	
	state_omega_type	磁束オブザーバー速度用ステージ	---	
	drv. phi_omega_div_emg_f	磁束オブザーバー用 EMG フラグ	---	
	state_theta_type	磁束オブザーバー角度用ステージ	---	

10.7.7.3 処理内容

磁束オブザーバー制御により、推定角度、推定速度を演算する。

モーター電流が検出できない場合、前回の速度平均値と周期から角度を演算する。

また、負荷トルクによって発生するトルク角を用いて、推定角度を補正する(補正 ON/OFF 設定可能)。

10.7.8 エンコーダー制御関数 (H_Encoder)

10.7.8.1 構文

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

引数：

_motor モーター制御構造体
ENx ENC アドレス

戻り値：

なし

10.7.8.2 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_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.omega_com	駆動速度指令値	Q31	
	drv.theta_com	電気角指令値	Q0	
	para.enc.ctrIprd	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	

10.7.8.3 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。

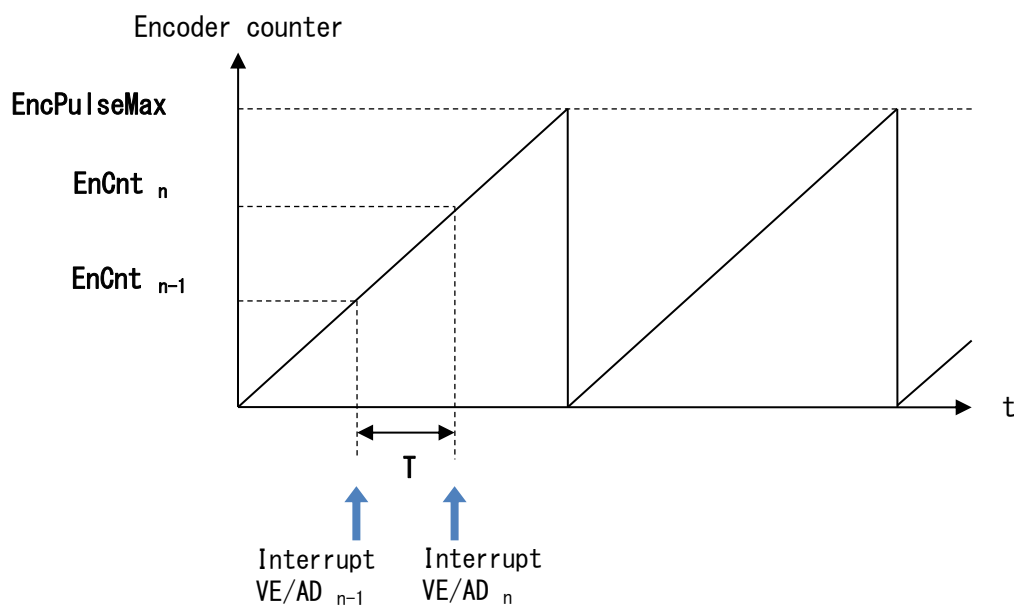


図 10.10 エンコーダーパルスからの位置および速度演算

VE/AD 変換終了割り込みごとに、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値から下記式により θ を演算します。

$$\theta = EnCnt_n \times (360^\circ \div EncPulseMax) \times (Pole \div 2)$$

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

$$\omega = \{(EnCnt_n - EnCnt_{n-1}) \times (360^\circ \div (EncPulseMax \div (Pole \div 2)))\} \div T$$

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

10.7.9 ホールセンサー+エンコーダー制御関数 (H_HallCon H_Hall_Encoder)

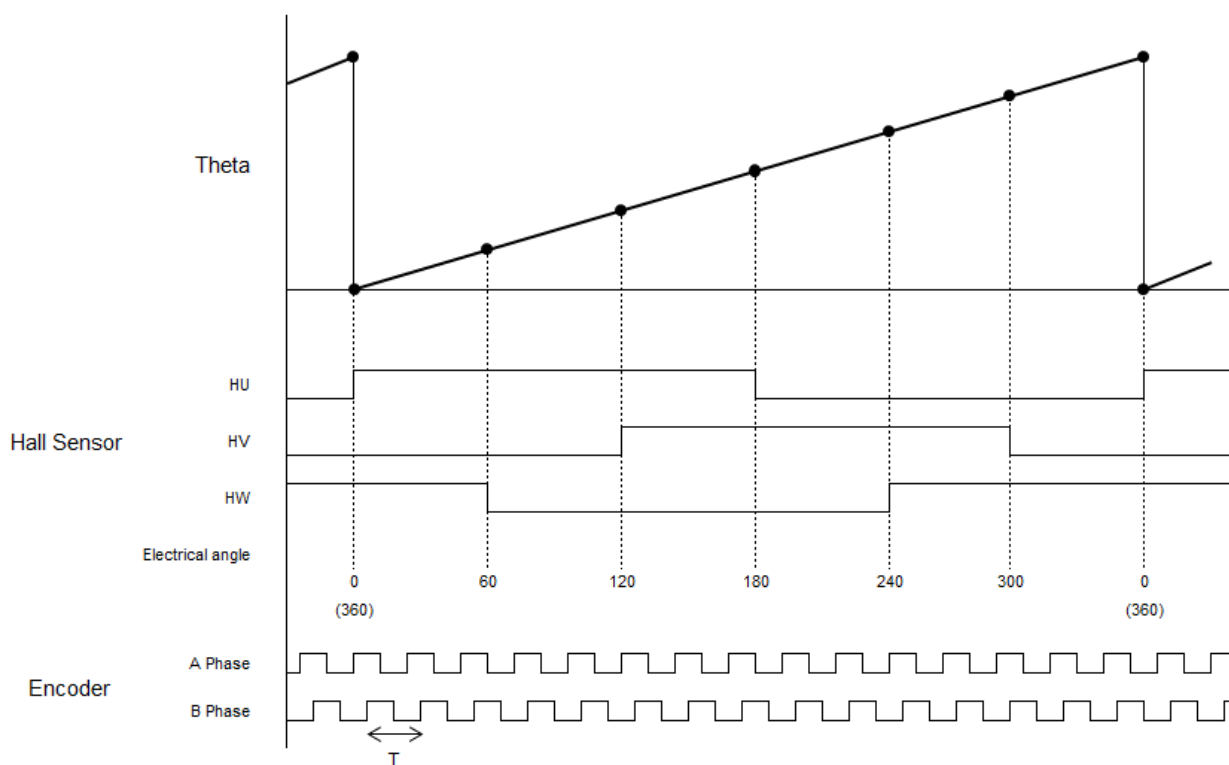


図 10.11 エンコーダーZ 信号検出前の位置検出制御

1. マイコンリセット直後は、ホール IC 信号から位置を検出する。(60 [deg]単位)
2. エンコーダーの Z 信号を検出するまでは、ホール IC 信号間をエンコーダーAB 相より補間し位置を検出する。
3. エンコーダーの Z 信号検出後は、エンコーダーABZ 相により位置を検出する。

10.7.10 速度制御関数 (D_Control_Speed)

10.7.10.1 構文

```
void D_Control_Speed(vector_t* const _motor)
```

引数：

`_motor` モーター制御構造体

戻り値：

なし

10.7.10.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_current	電流指令制御コマンド	---	
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	
	drv.omega	推定速度	Q31	
	drv.omega_com	駆動速度指令値	Q31	
	para.id_lim	d 軸電流制限値	Q31	
	para.iq_lim	q 軸電流制限値	Q31	
	para.spd.ki	速度制御積分ゲイン	Q15	Q12 対応あり
	para.spd.kp	速度制御比例ゲイン	Q15	Q12 対応あり
入出力	drv.Iq_ref_l	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	

10.7.10.3 処理内容

制御量が出力周波数 ω 、操作量が q 軸電流 I_q として PI 制御を行います。
速度指令値と実際の速度の偏差から、d 軸と q 軸電流基準値を決定します。

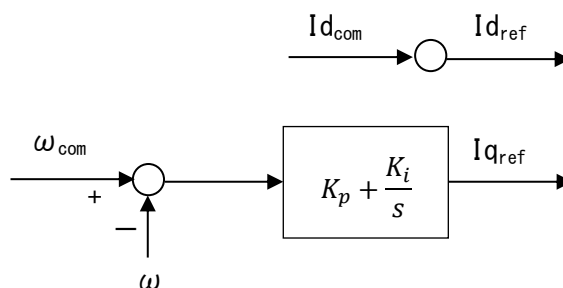


図 10.12 速度制御ブロック図

10.7.11 電流制御関数 (D_Control_Current)

10.7.11.1 構文

```
void D_Control_Current(vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

なし

10.7.11.2 変数

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

10.7.11.3 処理内容

電流基準値と実際の電流の偏差から、d 軸と q 軸電圧を決定します。電圧指令制御コマンドが SET のときは、下記 PI 制御により d 軸電圧、q 軸電圧を演算し、CLEAR のときは、d 軸電圧、q 軸電圧、積分値を指令値にします。制御量が q 軸電流 Iq、操作量が q 軸電圧 Vq として PI 制御を行います。
注意：シフト 2 モードの場合はリミットを設けています。

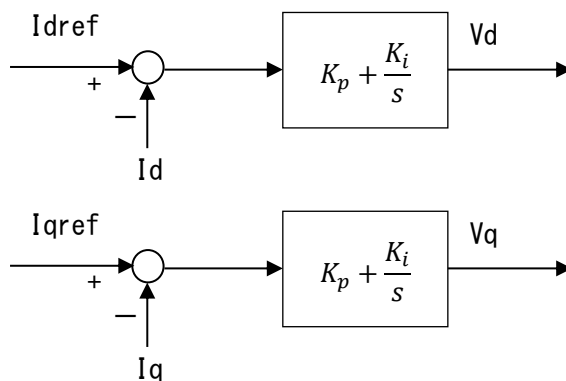


図 10.13 電流制御ブロック図

10.7.12 出力座標変換 (D_OutputTransformation)

10.7.12.1 構文

```
void D_OutputTransformation (vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

なし

10.7.12.2 変数

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

10. 7. 12. 3 処理内容

dq 軸電圧 (V_d, V_q) から 3 相の PWM Duty 比 ($Duty_U, Duty_V, Duty_W$) に変換します。

1. 10. 7. 133 逆パーク変換により、dq 軸電圧 (V_d, V_q) から $\alpha \beta$ 軸電圧 (V_α, V_β) に変換します。
2. $\alpha \beta$ 軸電圧 (V_α, V_β) から現在の「セクター」を演算します。演算前の「セクター」を「前回セクター」に保存し、モーター電流取得時に使用します。
3. 10. 7. 166 空間ベクトル変調あるいは 10. 7. 155 逆クラーク変換により、 $\alpha \beta$ 軸電圧 (V_α, V_β) から 3 相 Duty 比 ($Duty_U, Duty_V, Duty_W$) に変換します。

10.7.13 逆パーク変換 (E_InvPark)

10.7.13.1 構文

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

引数：

_d	d 軸
_q	q 軸
_theta	ローター位置 θ : $0 \leq \text{位置} < 360^\circ$ (0 to 0xffff)
_alpha	α 軸格納アドレス
_beta	β 軸格納アドレス

戻り値：

なし

10.7.13.2 処理内容

回転座標の dq 軸から静止座標の α β 軸に変換します。

下記演算式により V_α 、 V_β を演算します。

$$V_\alpha = V_d \times \cos\theta - V_q \times \sin\theta$$

$$V_\beta = V_d \times \sin\theta + V_q \times \cos\theta$$

10.7.14 セクター演算 (D_CalSector)

10.7.14.1 構文

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

引数 :

 _valpha α 軸電圧
 _vbeta β 軸電圧

戻り値 :

 セクター

10.7.14.2 処理内容

α β 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。

表 10.5 セクター演算

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
$(V\alpha < 0) \ \& \ (V\beta \geq 0)$	$ V\alpha < (V\beta/\sqrt{3})$	1
	$ V\alpha \geq (V\beta/\sqrt{3})$	2
$(V\alpha < 0) \ \& \ (V\beta < 0)$	$ V\alpha \geq (V\beta /\sqrt{3})$	3
	$ V\alpha < (V\beta /\sqrt{3})$	4
$(V\alpha \geq 0) \ \& \ (V\beta < 0)$	$V\alpha < (V\beta /\sqrt{3})$	4
	$V\alpha \geq (V\beta /\sqrt{3})$	5

10.7.15 逆クラーク変換 (E_InvClarke)

本ソフトでは未実装です

10.7.15.1 構文

```
void E_InvClarke(      q31_t _valpha,  
                       q31_t _vbeta,  
                       uint16_t* _p_vu,  
                       uint16_t* _p_vv,  
                       uint16_t* _p_vw)
```

引数：

_valpha	α 軸 V_α
_vbeta	β 軸 V_β
_p_vu	U 相電圧 V_u 格納アドレス
_p_vv	V 相電圧 V_v 格納アドレス
_p_vw	W 相電圧 V_w 格納アドレス

戻り値：

なし

10.7.15.2 処理内容

2 相 (V_α , V_β) から 3 相の電圧 (V_u , V_v , V_w) に変換します。

下記演算式により V_u , V_v , V_w を算出します。

$$V_u = V_\alpha \times \cos 0^\circ + V_\beta \times \sin 0^\circ = V_\alpha$$

$$V_v = V_\alpha \times \cos 120^\circ + V_\beta \times \sin 120^\circ = -\frac{1}{2}V_\alpha + \frac{\sqrt{3}}{2}V_\beta$$

$$V_w = V_\alpha \times \cos 240^\circ + V_\beta \times \sin 240^\circ = -\frac{1}{2}V_\alpha - \frac{\sqrt{3}}{2}V_\beta$$

10.7.16 空間ベクトル変調 (D_SVM)

10.7.16.1 構文

```
void D_SVM(    q31_t _valpha,  
               q31_t _vbeta,  
               q15_t _vdc,  
               uint8_t _sector,  
               uint8_t _modul,  
               uint16_t* _p_vu,  
               uint16_t* _p_vv,  
               uint16_t* _p_vw)
```

引数：

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

戻り値：

なし

10.7.16.2 処理内容

2 相の α β 軸電圧から 3 相 PWM の Duty 比を求めます。
空間ベクトル変調により各相の PWM Duty 比を求めます。

・空間ベクトル変調とは？

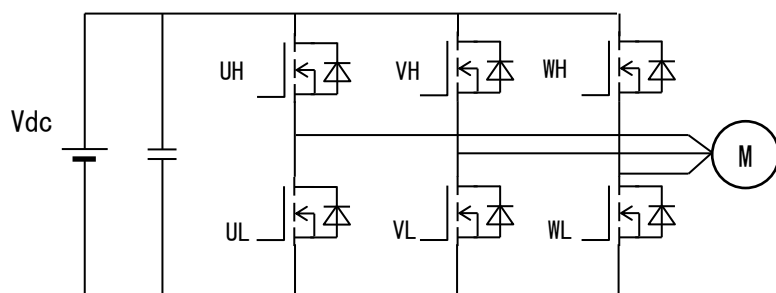
インバーター駆動回路のスイッチング素子の状態は、上相の素子は ON or OFF、下相の素子は上相の逆状態(デッドタイムは無視)となるため、表のように 8 パターン存在します。

この 8 パターンを V0 から V7 ベクトルとし、図に表すと、V0 と V7 ベクトルは、線間電圧がかからないため原点に存在し、残りの 6 つのベクトル(V1 から V6)は図 5 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せることにより、任意の出力電圧ベクトル V を得ることができます。

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

表 10.6 インバータースイッチング状態



UH: U 相上相スイッチング素子 UL: U 相下相スイッチング素子

VH: V 相上相スイッチング素子 VL: V 相下相スイッチング素子

WH: W 相上相スイッチング素子 WL: W 相下相スイッチング素子

Vdc: 電源電圧

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

0: 上相 OFF、下相 ON

1: 上相 ON、下相 OFF

図 10.14 インバーター駆動回路

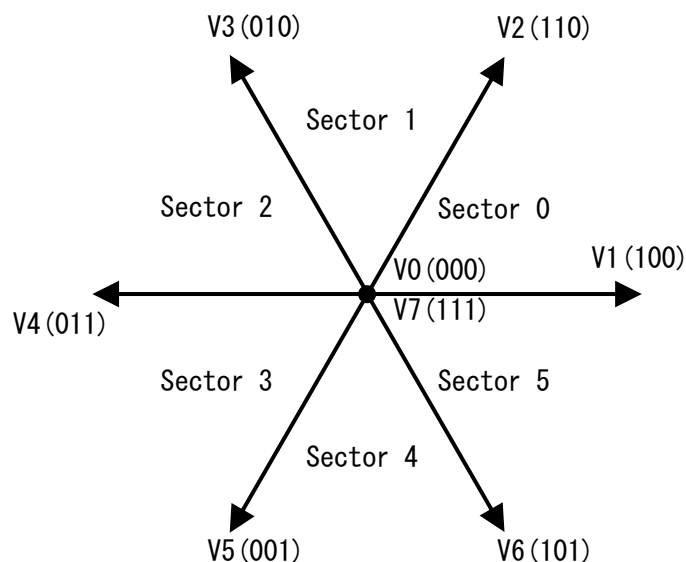


図 10.15 空間ベクトル

- ・出力電圧 V がセクター 0 内に存在する場合

例えば図 10.16 セクター 0 時の電圧ベクトルで、 V_α 、 V_β 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 V_1 と V_2 にそれぞれ係数 t_1 、 t_2 をかけて得られるベクトル V_1' 、 V_2' 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、図のように PWM 半周期で、 V_1 、 V_2 をそれぞれ時間 t_1 、 t_2 だけ発生させることで 出力電圧ベクトル V を合成します。

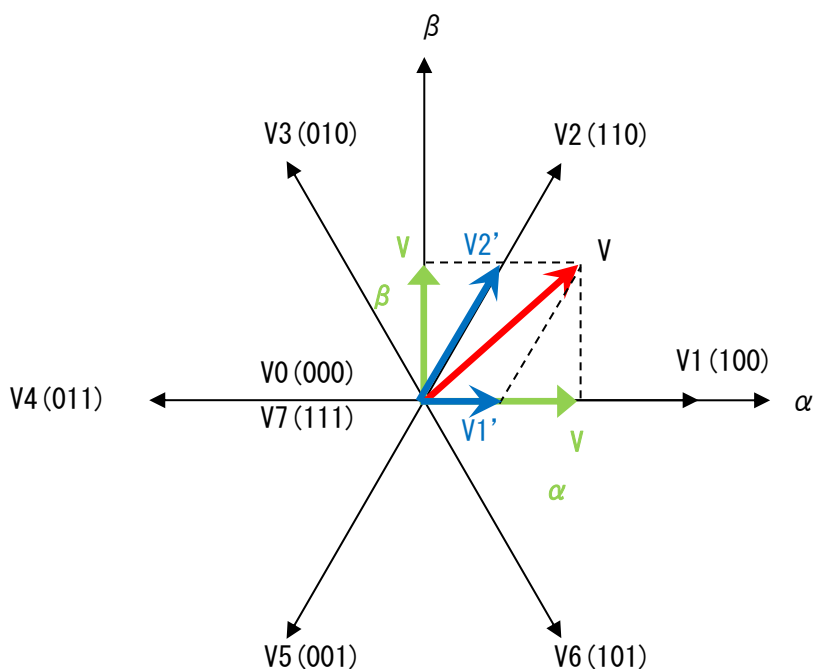


図 10.16 セクター 0 時の電圧ベクトル

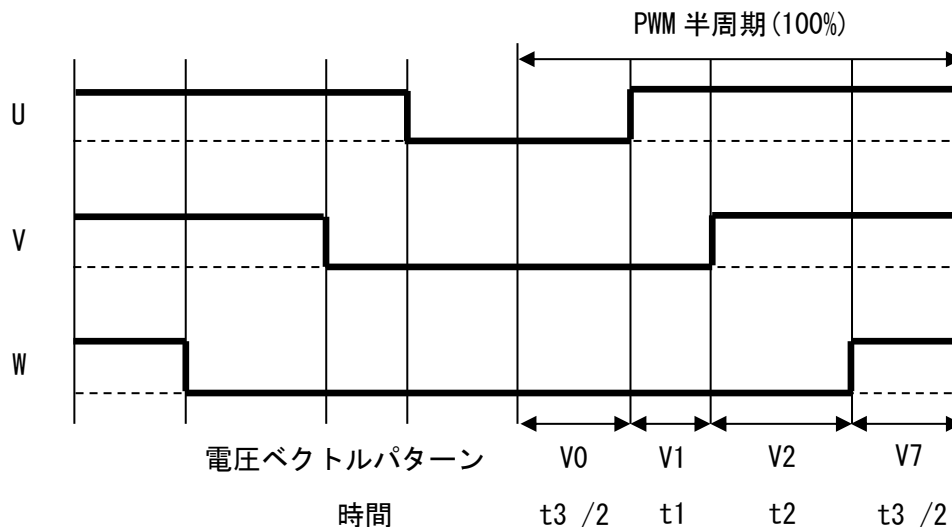


図 10.17 3 相変調時の PWM 波形

図 10.16 セクター0 時の電圧ベクトルより、 V_α, V_β は、

$$V_\alpha = V1' \times \cos 0^\circ + V2' \times \cos 60^\circ = V1' + \frac{V2'}{2}$$

$$V_\beta = V1' \times \sin 0^\circ + V2' \times \sin 60^\circ = \frac{\sqrt{3}}{2} \times V2'$$

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 V_{dc} とすると、

$$V1' = t1 \times V_{dc}$$

$$V2' = t2 \times V_{dc}$$

よって、 $t1, t2, t3$ の比率は、

$$t1 = k \times \frac{V_\alpha - \frac{1}{\sqrt{3}} V_\beta}{V_{dc}}$$

$$t2 = k \times \frac{\frac{2}{\sqrt{3}} V_\beta}{V_{dc}}$$

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

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。

$$\text{U 相 Duty} = t1 + t2 + (t3/2)$$

$$\text{V 相 Duty} = t2 + (t3/2)$$

$$\text{W 相 Duty} = t3/2$$

以下同様に出力電圧のセクターごとに、表 10.7 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 10.7 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、表の演算式で算出します。

表 10.8 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

10.7.17 トリガータイミング演算 (D_CalTrgTiming)

10.7.17.1 構文

```
void D_CalTrgTiming(vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

なし

10.7.17.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.DutyU	U 相 PWM Duty 比	Q15	0.0 to 1.0
	drv.DutyV	V 相 PWM Duty 比	Q15	0.0 to 1.0
	drv.DutyW	W 相 PWM Duty 比	Q15	0.0 to 1.0
	drv.command.modul	変調方式	---	2 相または 3 相変調
	drv.TrgPosMd	電流検出位置モード	---	3 シェント 1 シェント(前半) 1 シェント(後半)
	para.TrgComp	トリガータイミング補正值	Q15	-1.0 to 1.0
出力	drv.AdTrg0	AD トリガータイミング 0 (TRG0)	Q15	-1.0 to 1.0
	drv.AdTrg1	AD トリガータイミング 1 (TRG1)	Q15	-1.0 to 1.0

10.7.17.3 処理内容

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

3 シェント、1 シェントでトリガー位置が異なります。

・ 3 シャントの場合

基本トリガー位置は、-100%(キャリアエンド)。トリガータイミング補正値にプラス値が入っていたら、基本トリガー位置から後ろ方向へ補正します。マイナスの値の場合は、補正しないで-100%の位置となります。

変調方式	トリガータイミング (TRG) 電流 UVW、電源電圧 Vdc
2 相変調	100%
3 相変調	100%

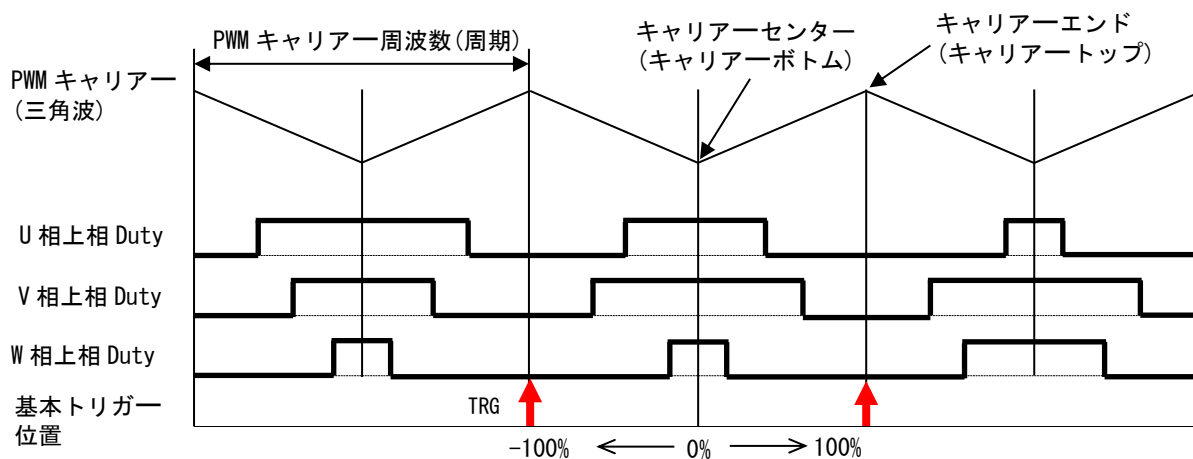


図 10.18 3 シャント時のトリガー位置

・ 1 シャントの場合

基本トリガー位置は、U, V, W 相の PWM Duty の値から演算します。

トリガータイミング補正值に 0 以外が入っていたら、演算したトリガー位置から補正します。

電源電圧 V_{dc} のトリガー位置は、キャリアーエンドの位置に固定されます。(初期設定時に設定し、この関数では処理しない)

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

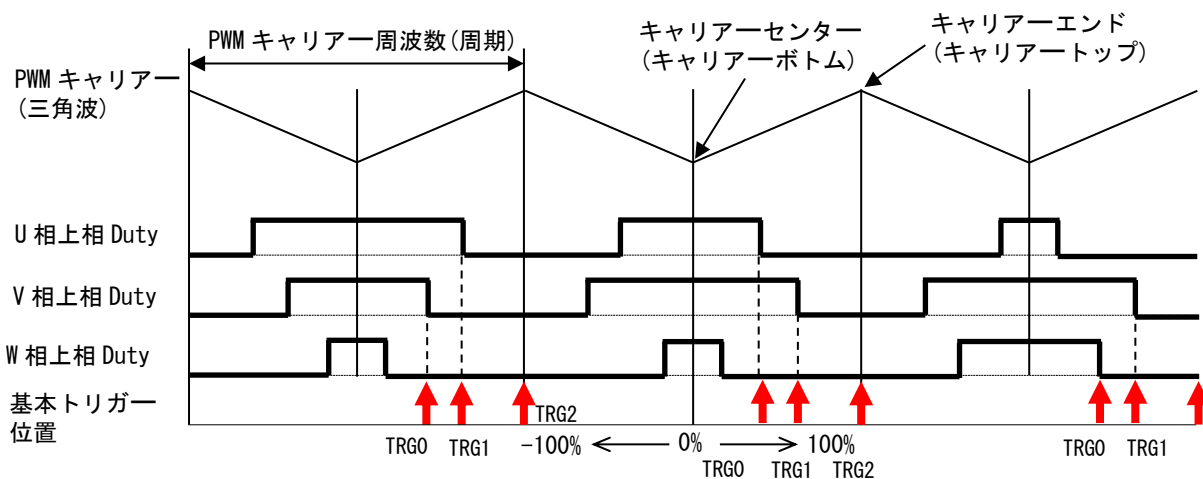


図 10.19 1 シャント、PWM 周期後半、3 相変調時のトリガー位置

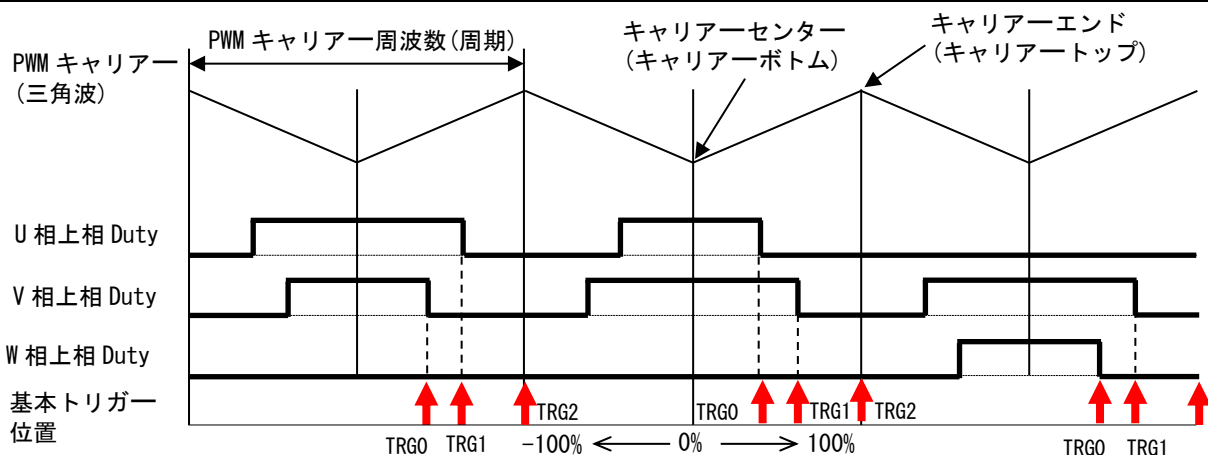


図 10.20 1 シェント、PWM 周期後半、2 相変調時のトリガー位置

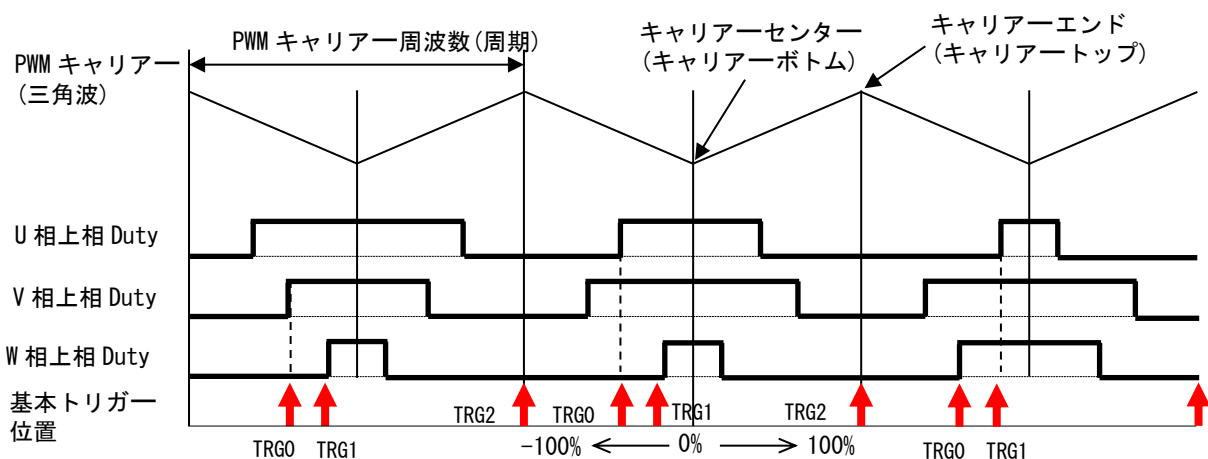


図 10.21 1 シェント、PWM 周期前半、3 相変調時のトリガー位置

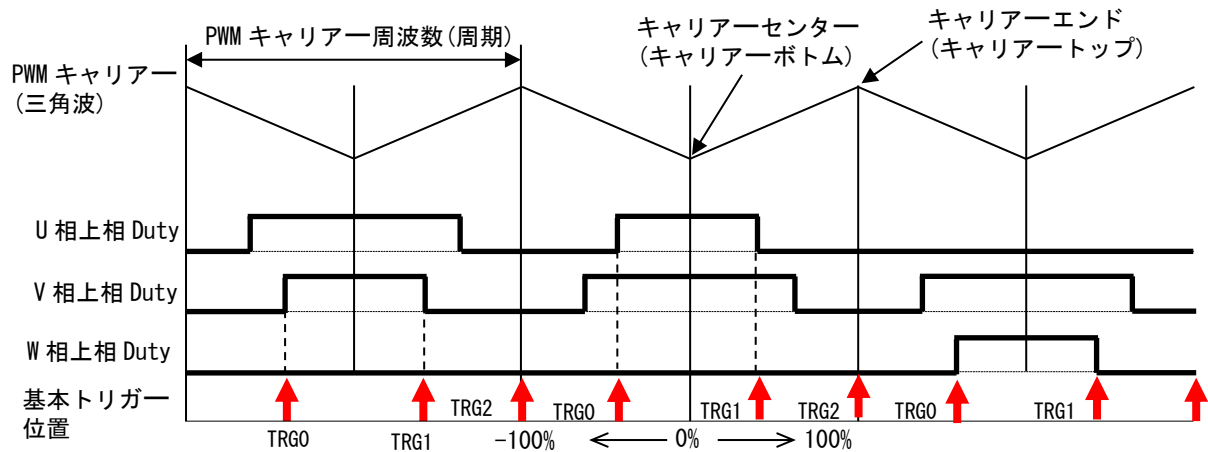


図 10.22 1 シャント、PWM 周期前半、2 相変調時のトリガー位置

10.7.18 PWM レジスタ設定 (PMD_RegDataSet)

10.7.18.1 構文

```
void PMD_RegDataSet(TSB_PMD_TypeDef* const PMDx,  
                    uint16_t duty_u,  
                    uint16_t duty_v,  
                    uint16_t duty_w,  
                    q15_t adtrg0,  
                    q15_t adrtgl)
```

引数：

PMDx	PMD レジスタアドレス
duty_u,	U 相 PWM Duty 比
duty_v,	V 相 PWM Duty 比
duty_w,	W 相 PWM Duty 比
adtrg0,	AD トリガタイミング 0 位置
adrtgl	AD トリガタイミング 1 位置

戻り値：

なし

10.7.18.2 処理内容

3 相 PWM Duty とトリガタイミングの値をレジスタに設定します。

入力された引数を、下記演算を行い PMD レジスタに設定します。

$$\text{TRGCMP0} = (\text{adtrg0} + 32768) \div 2$$
$$\text{TRGCMP1} = (\text{adrtgl} + 32768) \div 2$$
$$\text{CMPU} = \text{duty_u}$$
$$\text{CMPV} = \text{duty_v}$$
$$\text{CMPW} = \text{duty_w}$$

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

10.7.19.1 構文

```
void D_Check_DetectCurrentError (vector_t* const _motor)
```

引数：

 _motor モーター制御構造体

戻り値：

なし

10.7.19.2 変数

方向	名前	意味	Q フォーマット	備考
入力	para. chkpls	パルスチェック幅	---	
	shunt_type	シャントタイプ	---	1:1 シャント 3:3 シャント
	drv. command. modul	変調方式駆動コマンド	---	2 相または 3 相変調
	drv. DutyU	U 相 PWM Duty 比	Q15	0.0 to 1.0
	drv. DutyV	V 相 PWM Duty 比	Q15	0.0 to 1.0
	drv. DutyW	W 相 PWM Duty 比	Q15	0.0 to 1.0
出力	drv. idetect_error	電流検出状態	---	0:検出可能 1:検出不能

10.7.19.3 処理内容

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

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

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

ただし、シフト 2PWM モード時は検知を行いません。

注意：Vd/Vq 出力制限値の確認を行いたい場合はマクロを有効にして過電流検出を有効にしてください。

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。

・3 シャントの場合

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。

・1 シャントの場合

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

10.7.20.3 ショント電流誤検知検出関数 (D_Check_DetectCurrentError_3shunt)

10.7.20.1 構文

```
int D_Check_DetectCurrentError_3shunt (uint32_t _duty, uint16_t _chkplwidth)
```

引数：

_duty Duty の中間値
_chkplwidth パルスチェック幅

戻り値：

電流検出状態

10.7.20.2 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
	_chkplwidth	パルスチェック幅	---	
出力	drv.idetect_error	電流検出状態	---	0:検出可能 1:検出不能

10.7.20.3 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。

10.7.21 1 シャント電流誤検知検出関数 (D_Check_DetectCurrentError_1shunt)

10.7.21.1 構文

```
int D_Check_DetectCurrentError_1shunt (uint32_t _pwma, uint32_t _pwmb,
                                       uint32_t _pwmc, uint16_t _chkplwidth,
                                       int _sfhpwm, int _modu, int _sector,
                                       uint32_t _mdprd, uint16_t _trgcmp0,
                                       uint16_t _trgcmp1, uint16_t _shift2)
```

引数：

_pwma	U 相 PWM Duty 比
_pwmb	V 相 PWM Duty 比
_pwmc	W 相 PWM Duty 比
_chkplwidth	パルスチェック幅
_sfhpwm	PWM シフトモード状態
_modu	変調方式
_sector	セクター情報
_mdprd	PWM 周期
_trgcmp0	シフト 2 トリガータイミング 0
_trgcmp1	シフト 2 トリガータイミング 1
_shift2	PWM シフト 2 センター選択

戻り値：

電流検出状態

10.7.21.2 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
	_chkplwidth	パルスチェック幅	---	
	_sfhpwm	PWM シフトモード状態	---	
	_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	---	
	_trgcmp0	シフト 2 トリガータイミング 0	---	
	_trgcmp1	シフト 2 トリガータイミング 1	---	
	_shift2	PWM シフト 2 センター選択	---	0:PWM シフトなし 1:U 相センター 2:V 相センター

				3:W相センター
出力	---	電流検出状態	---	0:検出可能 1:検出不能

10.7.21.3 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。

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。

ただし、PWM シフトモード 2 の場合は下記判定条件を逸脱した場合、検出 NG とする。

- ・A判定:U相PWM出力の幅が最小となった場合、その幅が電流検出に必要な時間の2倍以上であること
- ・B判定:V相もしくは、W相のPWM出力の幅が最小となった場合、その幅が電流検出に必要な時間以上であること
- ・C判定:V相もしくは、W相のPWM出力の幅がPWM周期と電流検出に必要な時間との差分以下であること

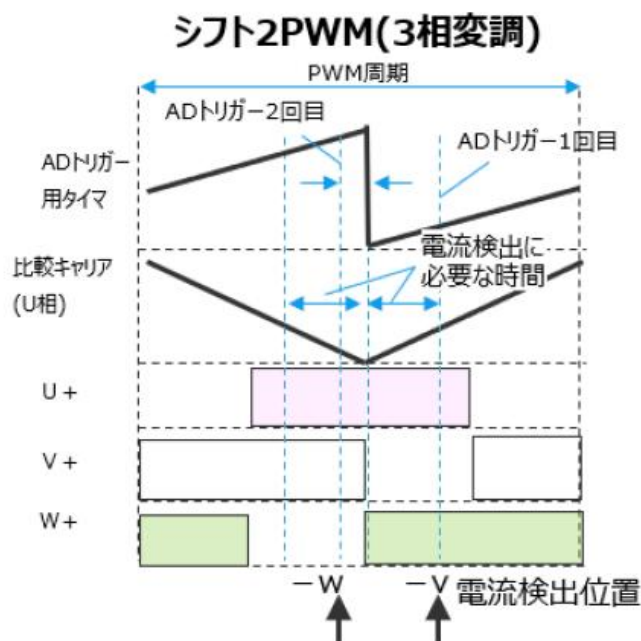


図 10.23 シフト 2PWM (3 相変調)

10.7.22 インバーター出力電圧の算出関数 (Cal_Vdq)

10.7.22.1 構文

q15_t Cal_Vdq (q15_t _vd, q15_t _vq)

引数 :

_vd d 軸電圧

_vq q 軸電圧

戻り値 :

インバーター出力電圧 (Vdq)

10.7.22.2 変数

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

10.7.22.3 処理内容

インバーター出力電圧 (Vdq) を算出します。

$$Vdq = \sqrt{3} \times \sqrt{vd^2 + vq^2}$$

11. 特殊機能説明

11.1 磁束オブザーバー適用位置推定方式

磁束オブザーバーを用いたセンサーレス位置推定制御は、主に α/β 軸の電流・電圧から磁束を演算し、そこから角度・トルク・速度を演算する。

ステージ遷移に応じたモーターの動作波形は以下となります。

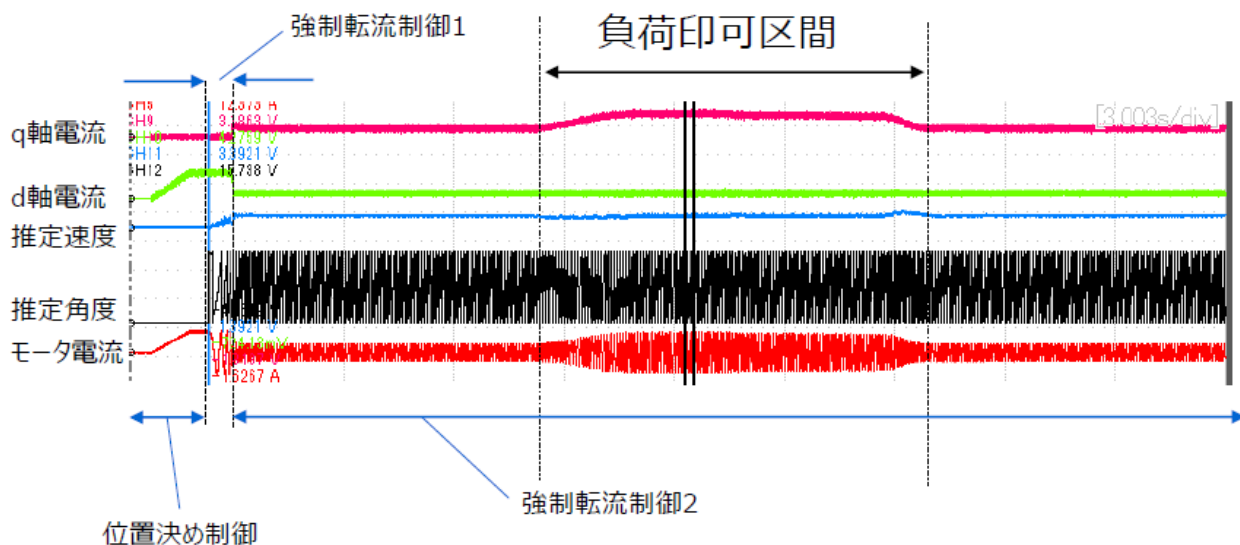


図 11.1 推定されたモーター磁束の角度に応じて、d 軸電流を印加する強制転流の場合のモーターの動作波形

強制転流制御 1 以降、指令回転数を上昇させる。指令回転数が、回転数閾値 1 を超えると強制転流制御 2 に移行する。強制転流 2 では、推定した角度を用いて、idref を固定値に設定しながら、iqref に速度制御の結果を用いることで、負荷に応じたモーター電流を流すことができる。

指令回転数が、回転数閾値 2 を超えると、センサーレス制御に移行し、idref を 0 に設定する。

また、関連処理はすべてライブラリーファイルにて管理する。

11.1.1 関数仕様

11.1.1.1 磁束オブザーバー制御の初期化関数 (Cal_phi_theta_ini)

構文:

```
q31_t Cal_phi_theta_ini ( phi_t* const _phi )
```

引数:

```
_phi          磁束オブザーバー構造体
```

戻り値:

```
なし
```

処理内容:

磁束オブザーバー適用位置推定方式の初期設定を行います。

11.1.1.2 電圧偏差の算出関数 (cal_sogil_alpha_voltage, cal_sogil_beta_voltage)

構文:

```
q31_t cal_sogil_alpha_voltage (phi_t* const _phi,  
                               q31_t _voltage,  
                               q15_t _current)  
q31_t cal_sogil_beta_voltage (phi_t* const _phi,  
                              q31_t _voltage,  
                              q15_t _current)
```

引数:

```
_phi          磁束オブザーバー構造体  
_voltage       $\alpha$  軸 ( $\beta$  軸) 電圧の前回値  
_current       $\alpha$  軸 ( $\beta$  軸) 電流
```

戻り値:

α 軸 (β 軸) 電圧偏差、および α 軸 (β 軸) 電圧偏差から α 軸 (β 軸) SOGI 中間値を差し引いた値

変数:

方向	名前	意味	Qフォーマット	備考
入力	_phi.phi_r	モーター巻き線抵抗	Q15	
	_voltage	α 軸 (β 軸) 電圧	Q31	
	_current	α 軸 (β 軸) 電流	Q15	
	_phi.alpha_SOGI4, beta_SOGI4	α 軸 (β 軸) SOGI 中間値	Q31	
出力	_phi.volt_dev_a, volt_dev_b	α 軸 (β 軸) 電圧偏差	Q15	

処理内容:

α 軸 (β 軸) 電圧偏差、および α 軸 (β 軸) 電圧偏差から α 軸 (β 軸) SOGI 中間値を差し引いた値を算出します。なお、後者の数値に SOGI ゲインをかけることで磁束オブザーバー適用位置推定に用いる α 軸 (β 軸) SOGI 中間値が算出できます。

11.1.1.3 推定角度演算関数 (Cal_phi_theta)

構文:

```
q31_t Cal_phi_theta (phi_t* const _phi,
                    q31_t _omega_com,
                    int32_t _ctrlprd,
                    q15_t _Ialpha,
                    q15_t _Ibeta,
                    q15_t _omega)
```

引数:

_phi	磁束オブザーバー構造体
_omega_com	駆動速度指令値
_ctrlprd	位置推定制御周期
_Ialpha	α 軸電流
_Ibeta	β 軸電流
_omega	推定速度

戻り値:

磁束オブザーバー推定角度

変数:

方向	名前	意味	Qフォーマット	備考
入力	_phi.phi_r	モーター巻き線抵抗	Q15	
	_phi.Lm	相互インダクタ値	Q31	
	_phi.volt_dev_a	α 軸電圧偏差	Q15	
	_phi.alpha_SOGI1	α 軸 SOGI 中間値	Q31	
	_phi.alpha_SOGI4	前回 α 軸 SOGI 中間値	Q31	
	_phi.alpha_SOGI5	前回 α 軸 SOGI 中間値	Q31	
	_phi.volt_dev_b	β 軸電圧偏差	Q15	
	_phi.beta_SOGI1	β 軸 SOGI 中間値	Q31	
	_phi.beta_SOGI4	前回 β 軸 SOGI 中間値	Q31	
	_phi.beta_SOGI5	前回 β 軸 SOGI 中間値	Q31	
	_Ialpha	α 軸電流	Q15	
	_Ibeta	β 軸電流	Q15	
	_omega_com	速度指令	Q31	
	_ctrlprd	位置推定制御周期	Q0	
	_SOGI_G	SOGI 制御ゲイン	Q31	
出力	_phi_th	磁束オブザーバー推定角度	Q0	
	_phi.alpha_SOGI4	α 軸 SOGI 中間値	Q31	
	_phi.alpha_SOGI5	α 軸 SOGI 中間値	Q31	
	_phi.beta_SOGI4	β 軸 SOGI 中間値	Q31	
	_phi.beta_SOGI5	β 軸 SOGI 中間値	Q31	

処理内容:

主に $I_{\alpha/\beta}$ 、 $V_{\alpha/\beta}$ から $\phi_{\alpha/\beta}$ を演算し、角度を算出します。

注意: $_SOGI_G$ 値は高速高負荷時に、位置推定が不安定となる場合があるため、速度指令に応じて可変としています。

以下、演算式・ブロック図となります。

$$\varphi_{\alpha} = \int (V_{\alpha} - R^* I_{\alpha}) dt - L_M I_{\alpha}$$

$$\varphi_{\beta} = \int (V_{\beta} - R^* I_{\beta}) dt - L_M I_{\beta}$$

$$SOGI_{\alpha 1} = (V_{\alpha} - R^* I_{\alpha}) - SOGI_{\alpha 4}^{*1}$$

$$SOGI_{\alpha 2} = SOGI_{\alpha 1} * G - SOGI_{\alpha 5}^{*1} * \omega_com^{*2}$$

$$SOGI_{\alpha 3} = SOGI_{\alpha 2} * \omega_com$$

$$SOGI_{\alpha 4} = \int (SOGI_{\alpha 3}) dt$$

$$SOGI_{\alpha 5} = \int (SOGI_{\alpha 4}) dt$$

$$\varphi_{\alpha} = SOGI_{\alpha 5} - L_M I_{\alpha}$$

$$SOGI_{\beta 1} = (V_{\beta} - R^* I_{\beta}) - SOGI_{\beta 4}$$

$$SOGI_{\beta 2} = SOGI_{\beta 1} * G - SOGI_{\beta 5} * \omega_com$$

$$SOGI_{\beta 3} = SOGI_{\beta 2} * \omega_com$$

$$SOGI_{\beta 4} = \int (SOGI_{\beta 3}) dt$$

$$SOGI_{\beta 5} = \int (SOGI_{\beta 4}) dt$$

$$\varphi_{\beta} = SOGI_{\beta 5} - L_M I_{\beta}$$

$$\Theta = \text{ATAN}(\varphi_{\beta} / \varphi_{\alpha})$$

*1: 前回値

*2: 絶対値

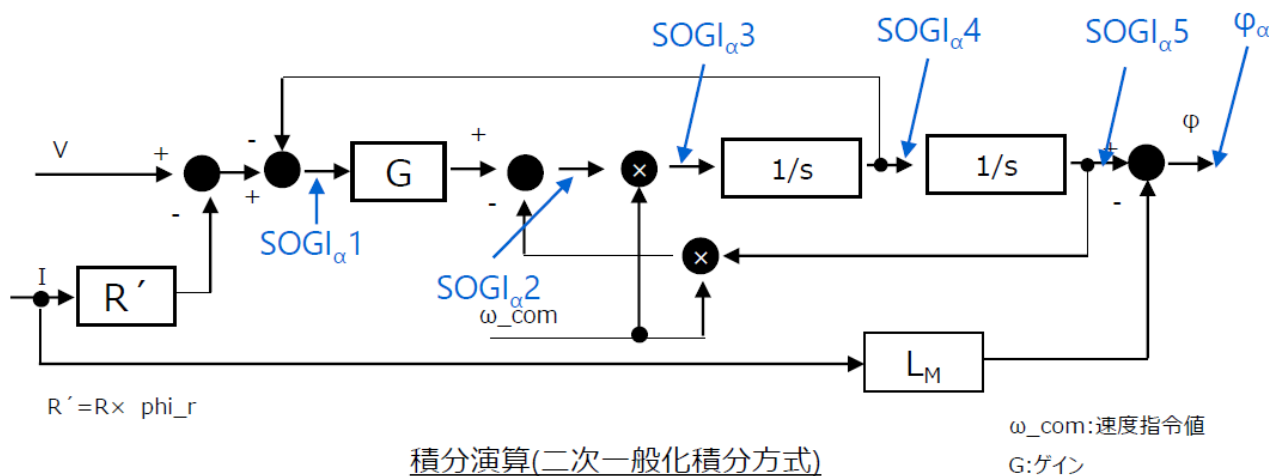


図 11.2 積分演算

11.1.1.4 角度補正演算関数 (Cal_theta_adj)

構文:

```
q31_t Cal_theta_adj (phi_t* const _phi,  
                    q31_t _ld,  
                    q31_t _lq,  
                    q15_t _id,  
                    q15_t _iq,  
                    q15_t _omega)
```

引数:

- _phi 磁束オブザーバー構造体
- _ld モーターd 軸インダクター
- _lq モーターq 軸インダクター
- _id d 軸電流
- _iq q 軸電流
- _omega 推定速度

戻り値:

角度補正值

変数:

方向	名前	意味	Q フォーマット	備考
入力	_phi.Phi	磁石磁束	Q31	
	_ld	モーターq 軸インダクタンス	Q31	
	_lq	モーターd 軸インダクタンス	Q31	
	_id	d 軸電流	Q15	
	_iq	q 軸電流	Q15	
出力	theta_adj	角度補正值	Q31	

処理内容:

負荷トルクによって発生するトルク角により、推定角度を補正します。

$$\delta = \tan^{-1}((_lq * _iq) / (_ld * _id + \phi \alpha))$$

$\phi \alpha$: 磁石磁束[Wb]

磁束オブザーバにおける推定角度の考え方

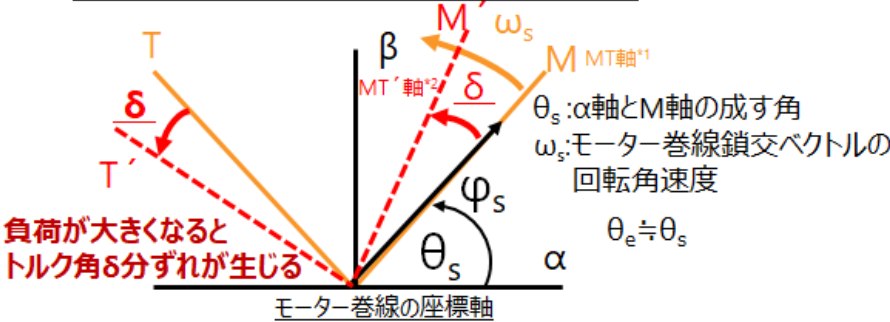


図 11.3 モーター巻線の座標軸

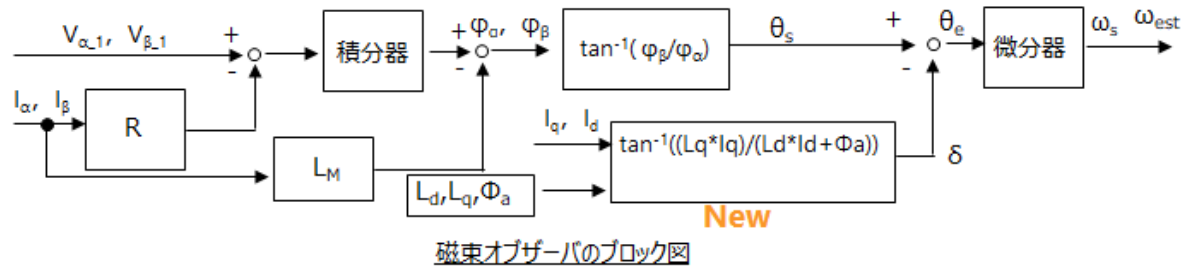


図 11.4 磁束オブザーバのブロック図

11.1.1.5 推定速度演算関数 (Cal_phi_omega)

構文:

```
q31_t Cal_phi_omega (phi_t* const _phi,
                    q31_t _theta_phi2,
                    q31_t _theta_phi_bf,
                    uint8_t _omegaphi_ave,
                    int32_t _ctrlprd)
```

引数:

_phi	磁束オブザーバー構造体
_theta_phi2	磁束オブザーバー適用推定角度の今回値
_theta_phi_bf	磁束オブザーバー適用推定角度の前回値
_omegaphi_ave	磁束オブザーバーの前回と今回の角度偏差演算、推定速度用フィルター係数
_ctrlprd	位置推定制御周期

戻り値:

磁束オブザーバー推定モーター速度

変数:

方向	名前	意味	Q フォーマット	備考
入力	_theta_phi_bf	磁束オブザーバー角度の前回値	Q31	
	_theta_phi2	磁束オブザーバー角度	Q31	
	_omegaphi_ave	磁束オブザーバーの前回と今回の角度偏差演算、推定速度用フィルター係数	Q0	
	_ctrlprd	位置推定制御周期	Q0	
入出力	Phi.theta_phi_div_ave	前回と今回の角度偏差	Q31	
出力	Phi.phi_om	磁束オブザーバー推定速度	Q31	

処理内容:

現在の推定角度と 1 周期前の推定角度から推定速度を演算します。

11.1.2 データ構造

11.1.2.1 phi_t

Data Fields:

q31_u alpha_SOGI4: [---] α 軸 SOGI 中間変数
 q31_u alpha_SOGI5: [---] α 軸 SOGI 中間変数
 q31_u beta_SOGI4: [---] β 軸 SOGI 中間変数
 q31_u beta_SOGI5: [---] β 軸 SOGI 中間変数
 q31_u theta_phi_div_ave: [deg] 前回と今回の角度偏差
 q31_t phi_om: [Hz] 磁束オブザーバー推定速度
 q31_t phi_al: [Wb] α 軸磁束
 q31_t phi_be: [Wb] β 軸磁束
 q31_t phi_th: [deg] 磁束オブザーバー推定角度
 q31_t Lm: [Vs/A] 相互インダクタ (Ld、Lq から算出)
 q31_t Phi: [Wb] 磁石磁束
 q31_t phi_r: [ohm] モーター巻き線抵抗
 q15_t volt_dev_a, volt_dev_b: [V] 電圧偏差
 q31_u alpha_SOGI1: [---] α 軸 SOGI 中間変数
 q31_u beta_SOGI1: [---] β 軸 SOGI 中間変数

11.1.3 パラメーター

パラメーター名	説明
phi_theta_time_CHx	磁束オブザーバー切り換え時、角度補正の切り換えタイムステップ
	単位[s]
	phi_CH_theta = 1 で有効となります。 Recommended value : 0.001~0.01
phi_SOGI_G_Chx	SOGI 用パラメーター

	磁束オブザーバーの応答性調整値を設定してください。 Recommended value : 2~5
phi_SOGI_G_high_CHx	高速時 SOGI 用パラメーター

	高速回転時の磁束オブザーバーの応答性調整値を設定してください。
phi_SOGI_G_high_Hz_CHx	SOGI 用パラメーター切り換え速度
	単位[Hz]
	SOGI 用パラメーターを切り換える速度を設定してください。
theta_phi_adj_CHx	磁束オブザーバー適用時位置推定誤差補正有無

	磁束オブザーバー適用時の位置推定誤差の補正有無を設定してください。 0 : トルクを考慮した角度補正なし 1 : トルクを考慮した角度補正あり
phi_EMG_CHx	磁束オブザーバーEMG 閾値

	EMG 判定値を設定してください。 磁束オブザーバーの推定速度が、急峻に変化した場合に EMG と判定する 閾値で $\text{omega_com} * \text{phi_EMG}$ 以上の推定速度の場合、EMG とする。 小さい値の場合、誤検知が増加する。
phi_r_CHx	磁束オブザーバー演算用抵抗値補正值
	--
	磁束オブザーバー演算時の抵抗値補正值を設定してください。 PTS 使用時、配線、インバーター分を含むため 0.7~0.9 を設定。 直接モーター巻線抵抗を測定する場合は、1.0 を設定。
phi_Id_ini_value_CHx	低速時 Id 印加倍率設定値
	--
	低速駆動時の Id を印加する倍率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。 Recommended value : 0.5~1.0
phi_Id_ini_Hz_CHx	低速時 Id 印加速度設定値
	[Hz]
	低速駆動時の Id を印加する周波数を設定してください。 センサーレス切り換え周波数 + phi_Id_CH_ini_Hz で印加します。
theta_phi_adj_tq_CHx	磁束オブザーバー位相調整比率
	--
	theta_phi_adj_CH = 1 で有効となります。 Recommended value : 0.5~2.0

12. 定数定義説明

12.1 モータードライバ設定用パラメーター共通 (D_Para.h)

12.1.1 モーター制御チャンネル選択

__CONTROL_MOTOR_CHO CHO を制御するとき定義してください。

12.1.2 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。

12.1.3 疑似温度調整選択

__USE_FAKETEMP 疑似温度調整を行うとき定義してください。

12.1.4 指令速度単位選択

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

12.1.5 特殊制御選択

__USE_PHI_DETECT_MOTOR_POSITION 磁束オブザーバー制御を行うとき定義してください。

12.1.6 共通パラメーター一覧

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。
cIX0_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
	--
	速度検出値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、値を大き目にしてください。 ただし、値を大きくすると検出値から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。

cOMEGAPI_AVE	磁束オブザーバーの前回と今回の角度偏差演算、推定速度用フィルター係数
	--
	フィルター係数を設定して下さい。 大きな値を設定すると安定しますが、反応が遅れます。 Recommended value : 2~4
cTORQUE_AVE	磁束オブザーバーのトルク負荷補正フィルター係数
	--
	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。 Recommended value : 2~4

12.2 モータードライバ設定用パラメーターモーターチャンネル別 (D_Para_Chx.h) x=0

モータードライバ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。

12.2.1 制御選択

12.2.1.1 センサーレス／センサー選択

- __USE_ENCODER_CHO エンコーダー制御を行う場合、定義してください。
__USE_HALL_CHO ホールセンサー制御を行う場合、定義してください。

両方定義しない場合は、センサーレス駆動となります。

同時に定義するとホールセンサー+エンコーダー駆動となります。

ホールセンサーのみの制御は対応しておりません。

12.2.1.2 VE 使用選択

- __USE_VE_CHO VE を使用する場合、定義してください。

12.2.1.3 COMP 使用選択

- __USE_COMP_CHO 内蔵コンパレータを使用する場合、定義してください。

12.2.1.4 PWM シフト 2 選択

- __SPWM2_CHO
0 : PWM シフト 2 (U 相センター) を使用する場合、定義してください。
1 : PWM シフト 2 (V 相センター) を使用する場合、定義してください。
2 : PWM シフト 2 (W 相センター) を使用する場合、定義してください。

12. 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] $\times 2^{12}$ 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。f (AD 変換が 1LSB 変化する相電流の変化量) [A] $\times 2^{11}$ 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 チャンネル設定 (3 シャント用)
	AINx
	U 相電流を検出する AD チャンネルを設定してください。
cADCH_CURRENT_V	V 相電流取得 AD チャンネル設定 (3 シャント用)
	AINx
	V 相電流を検出する AD チャンネルを設定してください。
cADCH_CURRENT_W	W 相電流取得 AD チャンネル設定 (3 シャント用)
	AINx
	W 相電流を検出する AD チャンネルを設定してください。
cADCH_CURRENT_IDC	電流取得 AD チャンネル設定 (1 シャント用)
	AINx
	電流を検出する AD チャンネルを設定してください。
cADCH_VDC	電源電圧取得 AD チャンネル設定
	AINx
	電源電圧 Vdc を検出する AD チャンネルを設定してください。
cOVC	過電流検出値の設定

	単位[A] 過電流検出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、出力をソフトウェアで OFF にします。
cVDC_MINLIM	電源電圧 Vdc 最小値の設定 単位[V] 電源電圧 Vdc の最小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 最大値の設定 単位[V] 電源電圧 Vdc の最大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 12.5ns@80MHz 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 50.0ns@20MHz 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 (VE 使用の時だけ) 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。
cSPEED_ACT	指令速度の設定 単位[Hz] or [RPS] or [RPM] 単位は__TGTSPD_UNIT の設定による モーター指令速度を設定してください。 SWx を ON にしたときに、この設定値でモーターが駆動します。
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: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	強制定常切換後待ち時間
	単位[s] 分解能:1ms
	強制定常切換後待ち時間を設定してください。
cHZ_MAX	最大周波数
	単位[Hz]
	マイコンで検出させる最大周波数を設定してください。
	制御で使用する最大周波数の 10 から 20%増し程度の値を設定してください。 値が小さいほど演算精度が上がりますが、検出値がこの値を超えると制御が破たんします。
cHZ_MIN	強制転流から定常への切り換え速度
	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	PWM シフト切り換え速度 (1 シャント 2 相変調用)
	単位[Hz]
	PWM シフト⇄通常 PWM の切り換え速度を設定してください。 指令速度の値で切り換えます。
cHZ_SPWM_HYS	PWM シフト切り換え速度ヒステリシス (1 シャント 2 相変調用)
	単位[Hz]
	PWM シフト⇄通常 PWM の切り換え速度のヒステリシスを設定してください。 指令速度を減算した値で切り換えます。
cMINPLS	最小パルス設定 (1 シャント用)
	単位[us]
	1 シャント駆動時、電流が取得できなくなるときの、PWM パルス幅時間を設定してください。(注 1) PWM パルス幅が、この設定値未満の値になると、検出した電流値は使用せず、前回値を使用して制御を行います。
cMAXPLS	最大パルス設定 (3 シャント用)
	単位[us]
	3 シャント駆動時、電流が取得できなくなるときの、中間相の PWM パルス幅時間を設定してください。 PWM パルス幅が、この設定値以上の値になると、検出した電流値は使用せず、前回値を使用して制御を行います。
cID_LIM	d 軸電流リミット値
	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 軸電流リミット値
	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 12. 2. 3. 2 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。 $V_d = cSPD_COEF \times \Omega_{com}$ cBOOT_TYPE が“cBoot_v”のときだけ有効 詳細は 12. 2. 3. 2 を参照してください。
cHZ_V2I	電圧制御から電流制御への切換速度
	単位[Hz]
	電圧制御から電流制御へ切り替える速度を設定してください。 モーターの推定速度が cHZ_MIN よりも速くなると強制転流状態から定常状態へ移行し、電圧制御から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 <u>cOMEGAENC_AVE</u> 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-ENC2 だけ有効)
	単位[us] 分解能 fc (12.5ns@80MHz)
	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
cTRGCMPO_SHIFT2	PWM シフト 2 のトリガー 0 保障時間設定 (VE)
	単位[ns]
	0 ~ (cPWMPRD/2) の範囲で設定してください。
cTRGCMP1_SHIFT2	PWM シフト 2 のトリガー 1 補償時間設定 (VE)
	単位[ns]
	0 ~ (cPWMPRD/2) の範囲で設定してください。
cLS_CONT_MD	PWM シフト 2 モード設定
	0: 通常 PWM/ 5: PWM シフト 2

	1 シャント時の PWM の設定をしてください。
cAD_TRG0_MODURATION_SHIFT2	PWM シフト 2 のトリガー0 時間設定 (ソフト)
	単位 [%]
	0~100%の範囲で設定してください。
cAD_TRG1_MODURATION_SHIFT2	PWM シフト 2 のトリガー1 時間設定 (ソフト)
	単位 [%]
	0~100%の範囲で設定してください。
cHZ_SPWM2	通常/PWM シフト 2 モード切り換え速度
	単位 [Hz]
	速度検出設定時に PWM シフト 2⇄通常 PWM へ切り換える速度を設定してください。 指令速度の値で切り換えます。
cHZ_SPWM2_HYS	通常/PWM シフト 2 モード切り換え速度ヒステリシス
	単位 [Hz]
	速度検出設定時に PWM シフト 2⇄通常 PWM へ切り換える速度のヒステリシスを設定してください。 指令速度を減算した値で切り換えます。
cIA_OFFSET_SHIFT2_U_CENTER	PWM シフト 2 用 U 相電流オフセット (U 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 (注 2)
cIB_OFFSET_SHIFT2_U_CENTER	PWM シフト 2 用 V 相電流オフセット (U 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 DAC 出力による電流値波形から補正してください。
cIC_OFFSET_SHIFT2_U_CENTER	PWM シフト 2 用 W 相電流オフセット (U 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 DAC 出力による電流値波形から補正してください。
cIA_OFFSET_SHIFT2_V_CENTER	PWM シフト 2 用 U 相電流オフセット (V 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 DAC 出力による電流値波形から補正してください。
cIB_OFFSET_SHIFT2_V_CENTER	PWM シフト 2 用 V 相電流オフセット (V 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 (注 3)
cIC_OFFSET_SHIFT2_V_CENTER	PWM シフト 2 用 W 相電流オフセット (V 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 DAC 出力による電流値波形から補正してください。

cIA_OFFSET_SHIFT2_W_CENTER	PWM シフト 2 用 U 相電流オフセット (W 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 DAC 出力による電流値波形から補正してください。
cIB_OFFSET_SHIFT2_W_CENTER	PWM シフト 2 用 V 相電流オフセット (W 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 DAC 出力による電流値波形から補正してください。
cIC_OFFSET_SHIFT2_W_CENTER	PWM シフト 2 用 W 相電流オフセット (W 相センター)
	単位 [A/AMAX]
	リップル電流による 3 相電流の補正值となります。 (注 2)
cVD_LIMIT_SHIFT2	PWM シフト 2 用 d 軸電圧制限
	単位 [V]
	PWM シフト 2 使用時の d 軸電流制限値を設定してください。 演算された d 軸電圧がこの値を超える場合は制限値が設定されます。
cVQ_LIMIT_SHIFT2	PWM シフト 2 用 q 軸電圧制限
	単位 [V]
	PWM シフト 2 使用時の q 軸電流制限値を設定してください。 演算された q 軸電圧がこの値を超える場合は制限値が設定されます。
cCOMP_OVC	内蔵コンパレータ過電流検出値 (正電流/負電流)
	単位 [A]
	内蔵コンパレータによる過電流検出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、出力をソフトウェアで OFF にします。
__Flux_CONTROL_MOTOR	磁束オブザーバー制御設定
	0: 無効/ 1: 有効
	磁束オブザーバー制御の有効/無効を設定してください。
cDETECT_TYPE	位置推定方式設定
	0: 誘起電圧方式/ 1: 磁束オブザーバー方式
	位置推定方式を設定してください。
cPHI_MAX	最大磁束値の設定
	単位 [Wb]
	基板設計に応じて値を変更してください。
cFORCE2_ONOFF	強制転流 2 の使用設定
	cForce2_on: 使用/cForce2_off: 不使用
	強制転流 2 の使用/不使用を設定して下さい。

注 1: PWM シフト動作中は、未評価です。PWM シフト機能を使用する場合は、電圧型で起動するなどの適用をお願いします

注 2: U 相電流は演算で算出するため設定は 0 固定となります

注 3: V 相電流は演算で算出するため設定は 0 固定となります

12.2.3 定数設定値と波形の関係

12.2.3.1 電流型起動時の定数値

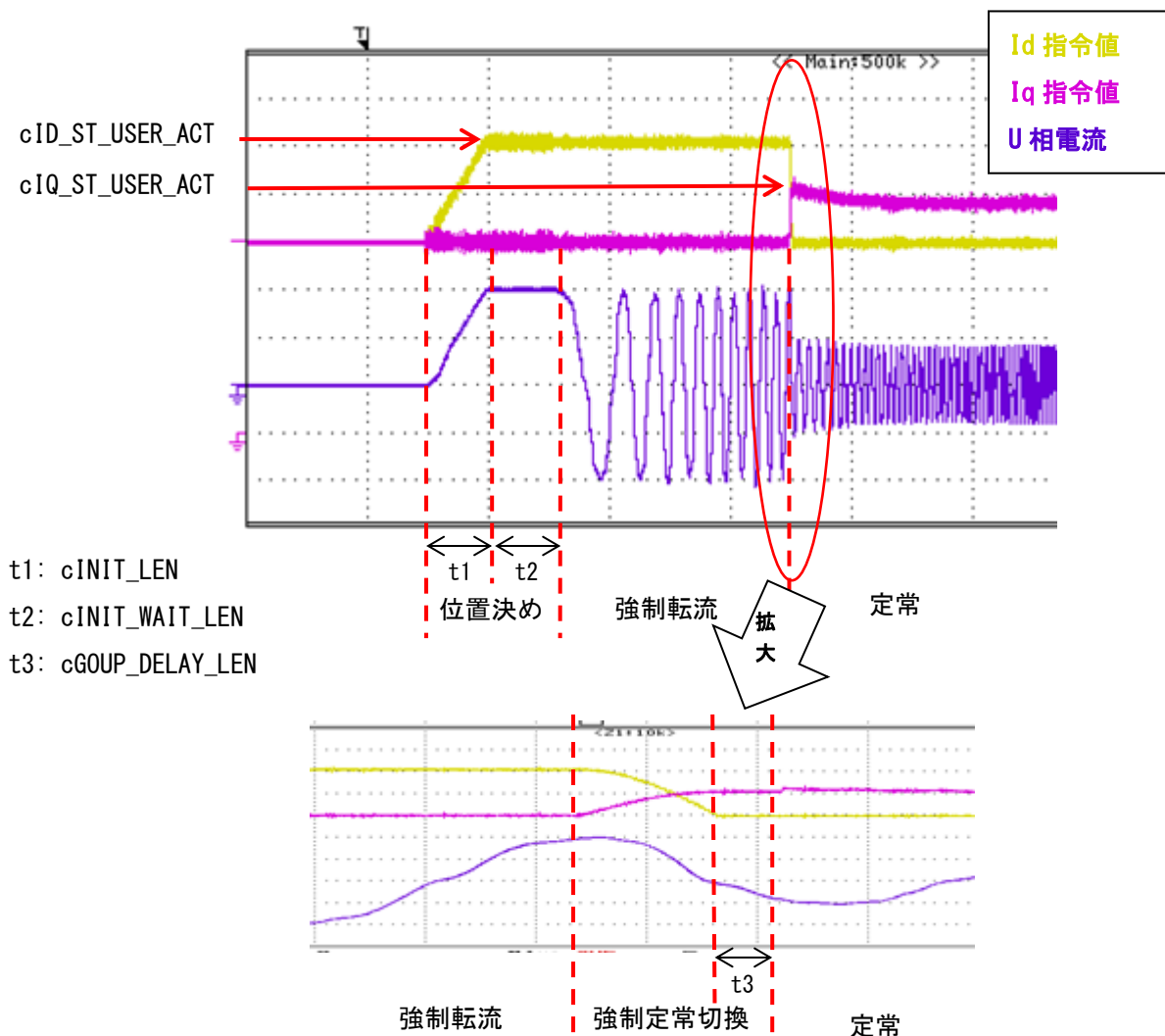


図 12.1 起動電流波形（電気角 0° に位置決め）

強制定常切換ステージで、cID_ST_USER_ACT と cIQ_ST_USER_ACT の値を更新する。
更新後、cGOUP_DELAY_LEN の時間だけ Iq 指令値一定値一定で制御します。
定常状態に移行した後、Iq 指令値は PI 制御により演算されます。

12.2.3.2 電圧型起動時の定数値

オシロスコープなどで電流波形を確認しながら、定数の値を決めてください。

電流波形を確認しながら目標の電流になるように、cVD_POS の値を調整してください。

強制転流中の電圧 Vd は、下記演算式で求めています。

演算式 「速度 × 定数 cSPD_COEF」

強制転流時の電流が、一定になるように cSPD_COEF を調整してください。

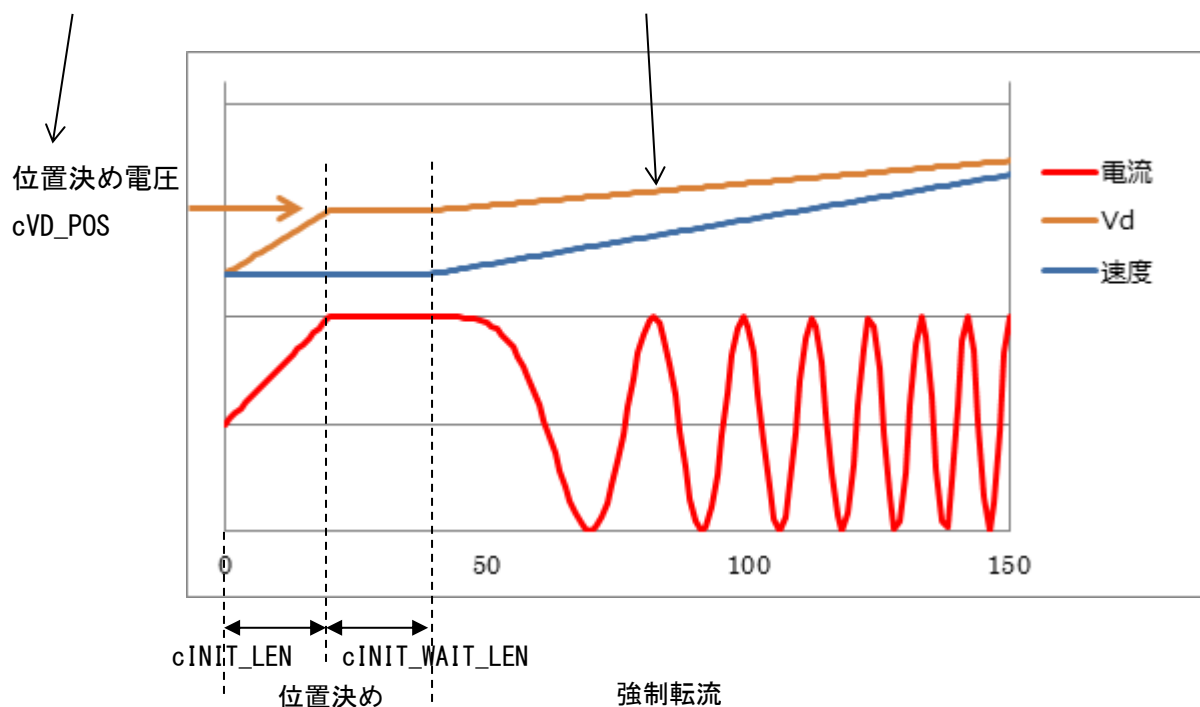


図 12.2 電圧駆動時のパラメーター調整方法

12.3 ユーザー制御関連定数

/*Initial value*/

#define cINIT_VALUE (0)

#define cINIT_FF (0xff)

/* Key setting */

#define cS_SW1 (*((__IO uint32_t *)BITBAND_PERI (&TSB_PH->DATA, 1))) /* SW1 */

#define cS_SW2 (*((__IO uint32_t *)BITBAND_PERI (&TSB_PA->DATA, 7))) /* SW2 */

#define cS_SW3 (*((__IO uint32_t *)BITBAND_PERI (&TSB_PE->DATA, 1))) /* SW3 */

#define cS_SW4 (*((__IO uint32_t *)BITBAND_PERI (&TSB_PE->DATA, 2))) /* SW4 */

#define cUSW1 (*((__IO uint32_t *)BITBAND_PERI (&TSB_PJ->DATA, 0))) /* USW1 */

#define cUSW2 (*((__IO uint32_t *)BITBAND_PERI (&TSB_PJ->DATA, 1))) /* USW2 */

#define cUSW3 (*((__IO uint32_t *)BITBAND_PERI (&TSB_PE->DATA, 3))) /* USW3 */

#define cKEY_CHATA_CNT (20) /* [cnt] chattering counter for KEY SW */

#define cKEY_NO_INPUT (0x00) /* USW1~3 no input */

#define cKEY_MASK_USW (0x07) /* USW1~3 mask */

```

#define cKEY_INPUT_USW1      (0x01)      /* USW1 input */
#define cKEY_INPUT_USW2      (0x02)      /* USW2 input */
#define cKEY_INPUT_USW3      (0x04)      /* USW3 input */

/* Soft ADC Setting */
#define cADUNIT_VR            TSB_ADA      /* VR ad data ADCUnit */
#define cADUNIT_TEMPO         TSB_ADA      /* TEMPO ad data ADCUnit */

#define cADTRG_USR            ADC_TRG_SINGLE /* AD Trigger Type */

#define cADCH_VR              ADC_AIN4      /* ADC Channel for VR */
#define cADREG_VR             ADC_REG8      /* Result register Number for VR */
#define cADCH_TEMPO           ADC_AIN13     /* ADC Channel for TEMPO */
#define cADREG_TEMPO          ADC_REG9      /* Result register Number for TEMPO */

#define cADAVECNT             (10)         /* ADC average count */
#define cONE_ABOVE            (1)

/* Speed Control Setting */
#define cSPEED_USER_STOP      (0)
#define cSPEED_USER_MIN_CH0   cHZ_MIN_CH0 /* [Hz] Min Target speed of motor ch0 */
#define cSPEED_USER_MAX       (200)       /* [Hz] Max Target speed of motor ch0 */
#define cAD_MIN               (0x10)       /* Voltage value 0.3V */
#define cAD_MAX               (0xF0)       /* Voltage value 4.7V */

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

/* LED setting */
#define cLED_EMG_CH0 (*( (__IO uint32_t *)BITBAND_PERI(&TSB_PD->DATA, 4))) /* LED4 */
#define cLED_ON               (1)          /* LED ON level */
#define cLED_OFF              (0)          /* LED OFF level */

/* State control Setting */
#define cMOTOR_CH0            (0x00)       /* Control channel : CH0 */

/* UART Setting */
#define cUART_BRD_INIT        ((uint32_t)0x00A6002B)
#define cUART_CLK_INIT        ((uint32_t)0x00000000)

```

```
#define cUART_CRO_INIT          ((uint32_t)0x00000001)
#define cUART_CR1_INIT          ((uint32_t)0x00000040)
#define cUART_SEND_WAIT        (1000)
#define cUART_COUNT             (0)
#define c1MHz                    (1000000)
#define cLOW_ACTIVE             (0)
#define cSWRST10                (0x2U)
#define cSWRST01                (0x1U)
#define cSWRSTF                 (0x80)
#define cSendBufSIZE            (80+1)

#define cDISP_INI_STATUS        (0x00) /* Initial status display */
#define cDISP_CONTROL_CH        (0x01) /* Control channel display */
#define cDISP_CARRIER_FREQUENCY (0x02) /* Carrier frequency display */
#define cDISP_DEAD_TIME         (0x03) /* Dead time display */
#define cDISP_GATE_ACTIVE       (0x04) /* Gate active display */
#define cDISP_POSITION_DETECT    (0x05) /* position detect display */
#define cDISP_VDC_VOLTAGE        (0x06) /* VDC voltage display */
#define cDISP_INVERTER_TEMP     (0x07) /* Inverter Temp */
#define cDISP_UVW_VOLTAGE        (0x08) /* U-phase/V-phase/W-phase voltage display */
#define cDISP_DAC_DATA          (0x09) /* Dac data display */
#define cDISP_INTERNAL_AMP       (0x0A) /* Internal amp display */
#define cDISP_DIRECTION          (0x0B) /* Direction display */
#define cDISP_MODULATION         (0x0C) /* Modulation display */
#define cDISP_ROTATE_INSTRUCTION (0x0D) /* Rotate instruction display */
#define cPOSITION_DETECT_3SHUNT  (3) /* Position Ditect : 3shunt */
#define cPOSITION_DETECT_1SHUNT  (1) /* Position Ditect : 1shunt */

/* User_interface Setting */
#define cFLG_ON                  (1)
#define cFLG_OFF                 (0)
#define cSW_Hi                   (1)
#define cSW_Low                  (0)
#define cROTATION_CW              (1)
#define cROTATION_CCW             (0)
#define cSPDTEMP_CHG_SPD          (1)
#define cSPDTEMP_CHG_TEMP         (0)
#define cINV_MIN_TEMP             (-200)
#define cINV_MAX_TEMP             (1000)
```

13. 制御、データ更新のタイミング

13.1 ソフトウェアによるベクトル制御

13.1.1 3 シャント制御

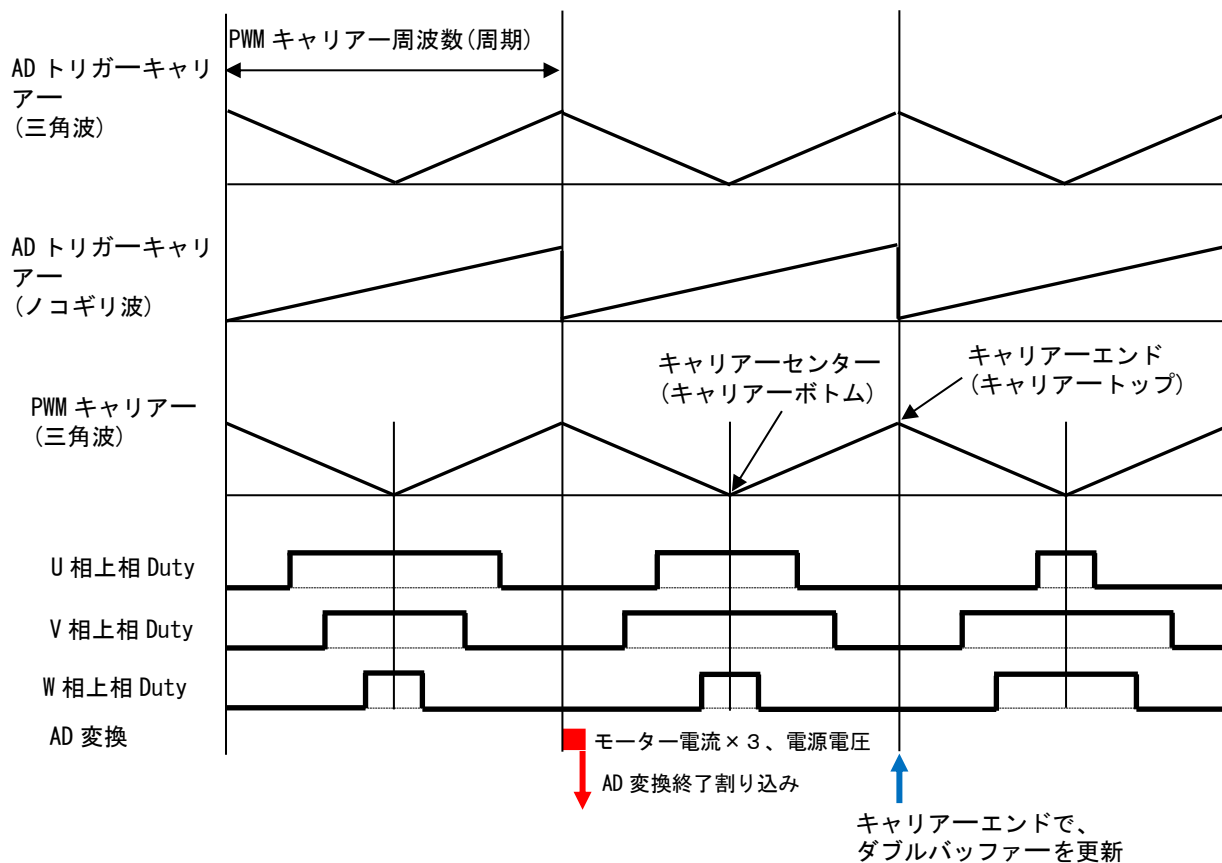


図 13.1 3 シャント制御

13.1.1.1 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

13.1.1.2 ダブルバッファ更新タイミング

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

13.1.1.3 割り込み関連

割り込み要求	割り込み	タイミング
PWM (PMD)	禁止	1 回、2 回、4 回ごと
AD 変換終了	許可	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

13.1.2 1 シャント制御

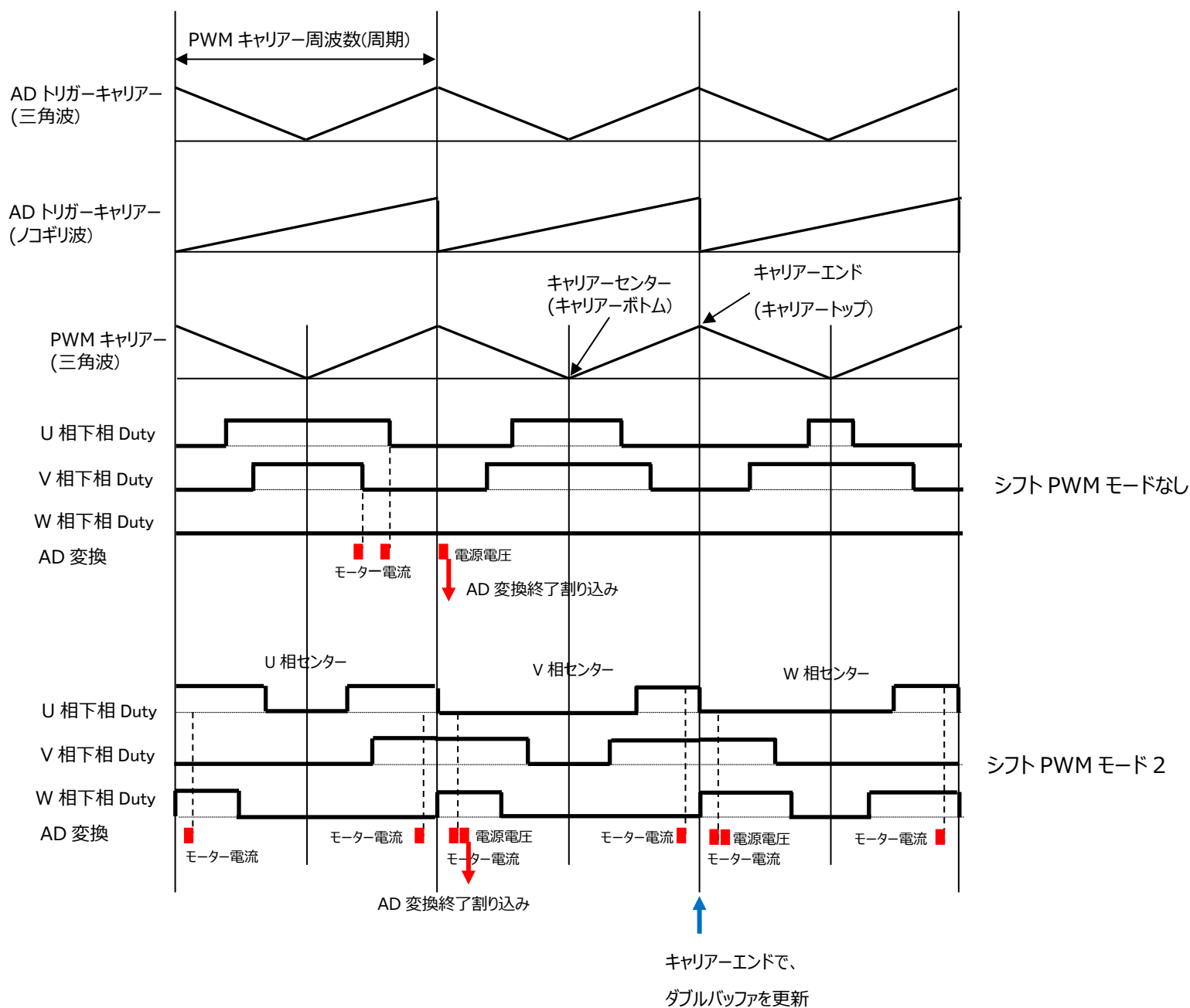


図 13.2 1 シャント制御

13.1.2.1 キャリアー波形

種類	波形	
	シフト PWM モードなし	シフト PWM モード 2
PWM	3 相とも三角波	センター(三角波)、センター以外(のこぎり波、及び逆のこぎり波)
AD トリガー	のこぎり波	のこぎり波

13.1.2.2 ダブルバッファ更新タイミング

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

13.1.2.3 割り込み関連

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

13.2 VE 使用によるベクトル制御

13.2.1.3 シャント制御

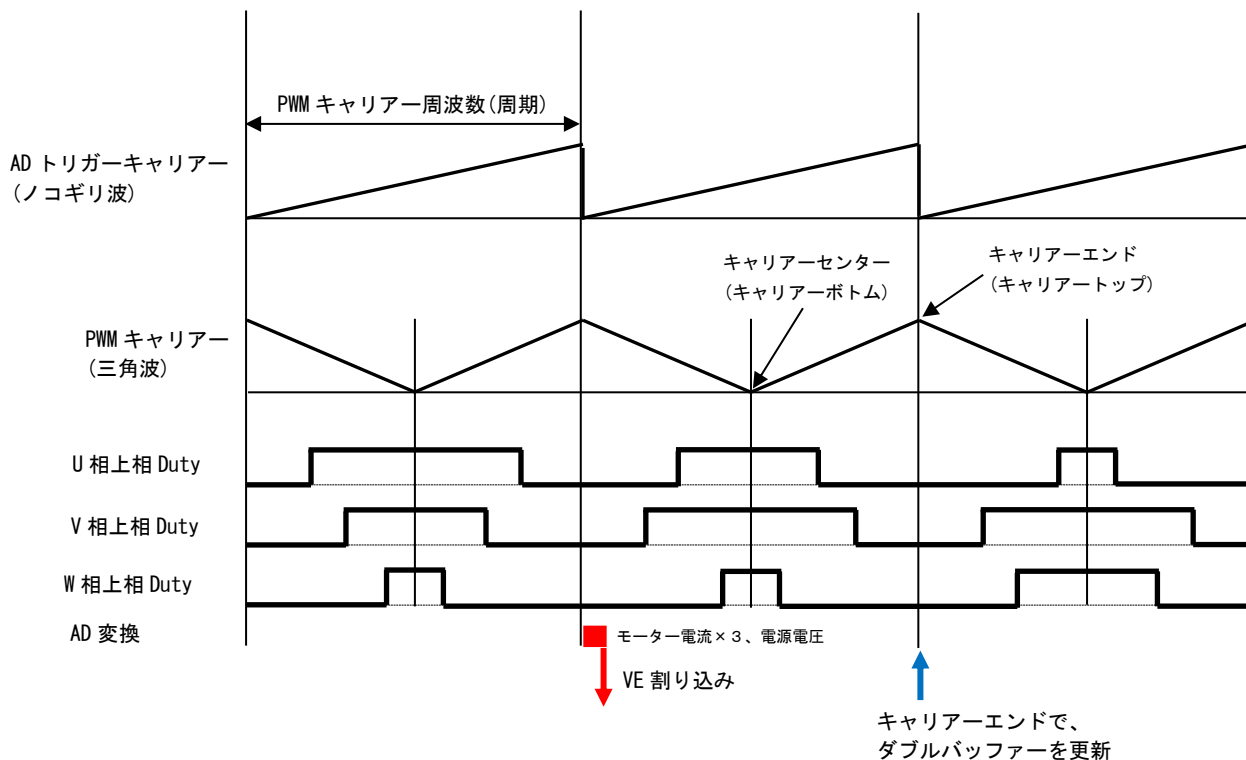


図 13.3 3 シャント制御

13.2.1.1 キャリアー波形

種類	波形
PWM	3 相とも三角波
AD トリガー	のこぎり波

13.2.1.2 ダブルバッファ更新タイミング

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

13.2.1.3 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

13.2.2.1 シェント制御

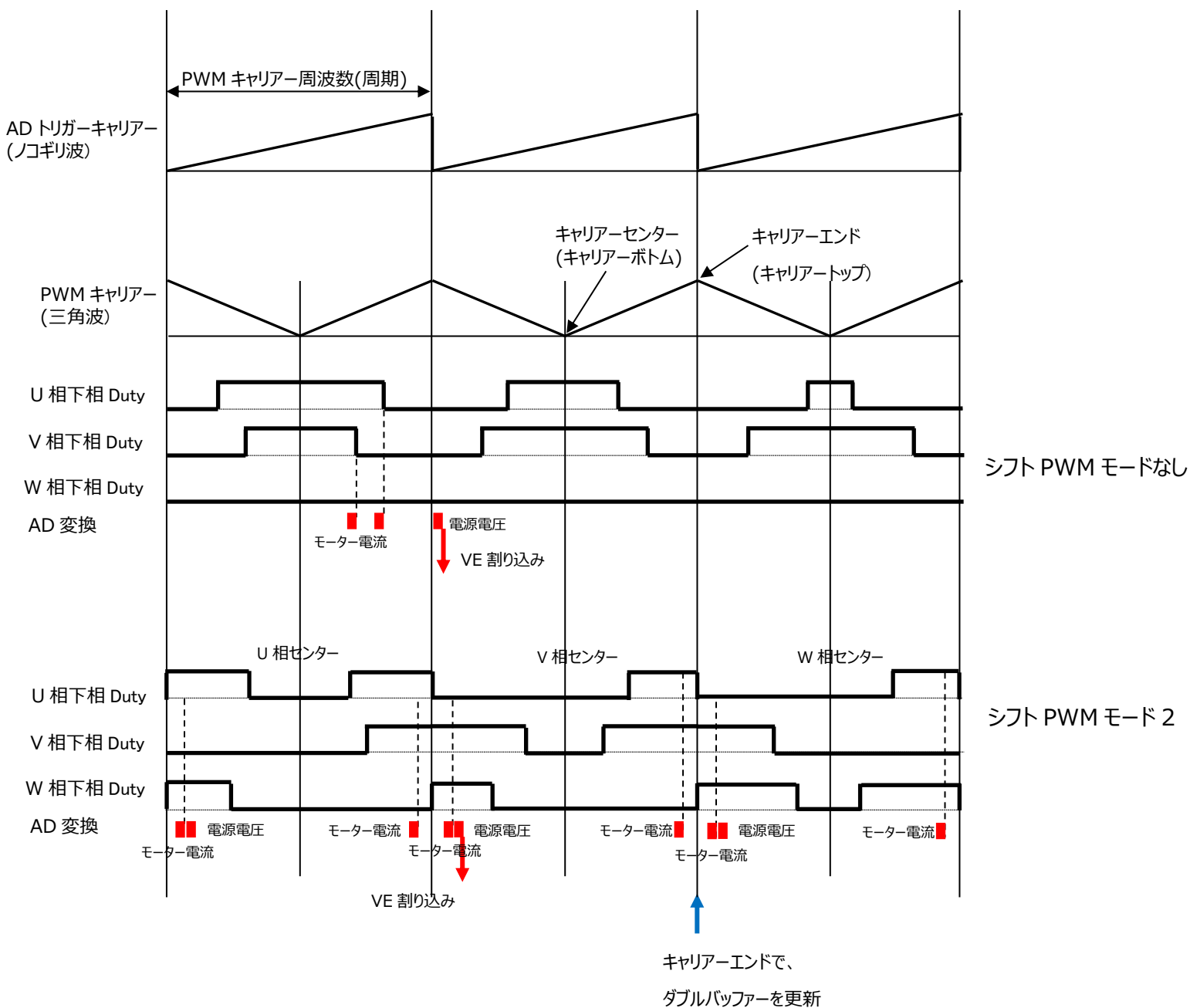


図 13.4 1 シェント制御

13.2.2.1 キャリアー波形

種類	波形	
	シフト PWM モードなし	シフト PWM モード 2
PWM	3 相とも三角波	センター(三角波)、センター以外(のこぎり波、及び逆のこぎり波)
AD トリガー	のこぎり波	のこぎり波

13.2.2.2 ダブルバッファ更新タイミング

PWM シフトモード 2 未使用（シフト 2 以外）

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

PWM シフトモード 2 使用

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

13.2.2.3 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

14. ペリフェラルドライバー

14.1 マイコン周辺回路アドレス

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

14.1.1 データ構造

14.1.1.1 ipdrv_t

Data Fields:

TSB_VE_TypeDef* const	VE_x : VE アドレスを選択します。
TSB_PMD_TypeDef* const	PMD_x : PMD アドレスを選択します。
TSB_AD_TypeDef* const	AD_x : ADC アドレスを選択します。

14.2 ベクトルエンジン(VE)

14.2.1 関数仕様

14.2.1.1 IP_VE_init

VE 初期設定

API:

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

引数:

VE_x: 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 14.2.2.1 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

14. 2. 1. 2 VE_Start

VE スタート

API:

```
void VE_Start(const ipdrv_t* const _ipdrv);
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14. 1. 1. 1 を参照してください。

機能:

VE を開始します。

補足:

特になし。

戻り値:

なし

14. 2. 1. 3 VE_GetPhaseCurrent

相電流の取得

API:

void

```
VE_GetPhaseCurrent(const ipdrv_t* const _ipdrv, q15_t* _ia, q15_t* _ib, q15_t* _ic);
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_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

14. 2. 1. 4 VE_GetCurrentAdcData

電流 AD 値取得

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_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 レジスタの値を各相の電流の AD 値として取得します。

補足:

特になし

戻り値:

なし

14. 2. 1. 5 VE_GetdataFromVEreg

VE レジスタデータ取得

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_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:

VE レジスタからモーター電源電圧 V_{dc} 、d 軸電圧 V_d 、q 軸電圧 V_q 、d 軸電流 I_d 、q 軸電流 I_q 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 I_d 、q 軸電流 I_q の値を VE レジスタへ前回検出値を書きこみます。

補足:

特になし

戻り値:

なし

14. 2. 1. 6 VE_GetPWM_DutyU

U 相 Duty 値取得

API:

```
uint32_t VE_GetPWM_DutyU(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

機能:

U 相 Duty レジスタの値を取得します。

補足:

特になし

戻り値:

U 相 Duty

14. 2. 1. 7 VE_GetPWM_DutyV

V 相 Duty 値取得

API:

```
uint32_t    VE_GetPWM_DutyV(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14. 1. 1. 1 を参照してください。

機能:

V 相 Duty レジスタの値を取得します。

補足:

特になし

戻り値:

V 相 Duty

14. 2. 1. 8 VE_GetPWM_DutyW

W 相 Duty 値取得

API:

```
uint32_t    VE_GetPWM_DutyW(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14. 1. 1. 1 を参照してください。

機能:

W 相 Duty レジスタの値を取得します。

補足:

特になし

戻り値:

W 相 Duty

14. 2. 1. 9 VE_GetPWM_DutyMed

Duty 中間値取得

API:

```
uint32_t VE_GetPWM_DutyMed(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14. 1. 1. 1 を参照してください。

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

14. 2. 1. 10 VE_GetPWM_Modulation

変調方式取得

API:

```
int VE_GetPWM_Modulation(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14. 1. 1. 1 を参照してください。

機能:

変調方式を取得します。

補足:

特になし

戻り値:

変調方式 0:3 相変調、1:2 相変調

14. 2. 1. 11 VE_GetSector

現在セクター取得

API:

```
uint32_t VE_GetSector(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14. 1. 1. 1 を参照してください。

機能:

現在のセクター値を取得します。

補足:

特になし

戻り値:

現在セクター値 [0-11]

14. 2. 1. 12 VE_GetShiftPWMState

PWM シフト状態取得

API:

```
int VE_GetShiftPWMState(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14. 1. 1. 1 を参照してください。

機能:

PWM シフト状態を取得します。

補足:

特になし

戻り値:

PWM シフト状態 0:通常 PWM、1:PWM シフト

14. 2. 1. 13 VE_GetOutputMode

PWM 出力状態取得

API:

```
int VE_GetOutputMode(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14. 1. 1. 1 を参照してください。

機能:

PWM 出力状態を取得します。

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

14. 2. 1. 14 VE_SetdataToVEreg_Stop

VE レジスタへのデータセット (Stop)

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_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 レジスタへの設定処理を行います。

電流制御積分ゲイン値レジスタなどの初期化を行います。

プログラムタスクポインター開始レジスタに” 0” を設定します。

補足:

特になし

戻り値:

なし

14. 2. 1. 15 VE_SetdataToVEreg_Bootstrap

VE レジスタへのデータセット(Bootstrap)

API:

```
Void          VE_SetdataToVEreg_ Bootstrap(const ipdrv_t* const _ipdrv,  
                                           const vector_t* const _motor);
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_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 レジスタへの設定処理を行います。

VE の短絡ブレーキ機能を利用して L 側だけ ON となる波形を出力します。

プログラムタスクポインター開始レジスタに” 3” を設定します。

補足:

特になし

戻り値:

なし

14. 2. 1. 16 VE_SetdataToVEreg_Initposition_i

VE レジスタへのデータセット(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: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_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 レジスタへの設定処理を行います。

プログラムタスクポインター開始レジスタに

PWM シフト 2 モードの場合” 20”、それ以外は” 11” を設定します。

補足:

特になし

戻り値:

なし

14. 2. 1. 17 VE_SetdataToVEreg_Initposition_v

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

API:

```
void VE_SetdataToVEreg_Initposition_v (const ipdrv_t* const _ipdrv,  
                                         const vector_t* const _motor);
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14.1.1.1 を参照してください。

_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 レジスタへの設定処理を行います。

プログラムタスクポインター開始レジスタに” 3” を設定します。

補足:

特になし

戻り値:

なし

14.2.1.18 VE_SetdataToVEreg_Force_i

VE レジスタへのデータセット (強制転流 電流制御タイプ)

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14.1.1.1 を参照してください。

_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 レジスタへの設定処理を行います。

プログラムタスクポインター開始レジスタに

PWM シフト 2 モードの場合” 20”、それ以外は” 11” を設定します。

補足:

特になし

戻り値:

なし

14.2.1.19 VE_SetdataToVEreg_Force_v

VE レジスタへのデータセット(強制転流 電圧制御タイプ)

API:

void

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14.1.1.1 を参照してください。

_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 レジスタへの設定処理を行います。

プログラムタスクポインター開始レジスタに” 3” を設定します。

補足:

特になし

戻り値:

なし

14. 2. 1. 20 VE_SetdataToVEreg_Change_up

VE レジスタへのデータセット(強制定常切換)

API:

```
void VE_SetdataToVEreg_Change_up (const ipdrv_t* const _ipdrv,  
                                   const vector_t* const _motor);
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_motor: ベクトル制御変数の構造体アドレスを設定します。

機能:

強制定常切換状態の VE レジスタへの設定処理を行います。

プログラムタスクポインター開始レジスタに

PWM シフト 2 モードの場合” 20”、それ以外は” 11” を設定します。

補足:

特になし

戻り値:

なし

14. 2. 1. 21 VE_SetdataToVEreg_Steady_A

VE レジスタへのデータセット(定常)

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_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 レジスタへの設定処理を行います。

プログラムタスクポインター開始レジスタに

PWM シフト 2 モードの場合” 20”、それ以外は” 11” を設定します。

補足:

特になし

戻り値:

なし

14. 2. 1. 22 VE_SetdataToVEreg_Change_down

VE レジスタへのデータセット(定常強制切換)

API:

```
void VE_SetdataToVEreg_Change_down (const ipdrv_t* const _ipdrv,  
                                     const vector_t* const _motor);
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_motor: ベクトル制御変数の構造体アドレスを設定します。

機能:

強制定常切換状態の VE レジスタへの設定処理を行います。

プログラムタスクポインター開始レジスタに

PWM シフト 2 モードの場合” 20”、それ以外は” 11” を設定します。

補足:

特になし

14. 2. 1. 23 VE_SetdataToVEreg_Emergency

VE レジスタへのデータセット(EMG)

API:

```
void VE_SetdataToVEreg_Emergency (const ipdrv_t* const _ipdrv,  
                                   const vector_t* const _motor);
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_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 レジスタへの設定処理を行います。

プログラムタスクポインター開始レジスタに” 0” を設定します。

補足:

特になし

戻り値:

なし

14.2.1.24 VE_SetZeroCurrentData

ゼロ電流設定

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14.1.1.1 を参照してください。

_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

14.2.1.25 VE_SetVDCreg

モーターの電源電圧 Vdc 設定

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14.1.1.1 を参照してください。

_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

14. 2. 1. 26 VE_SetSpwmMode

PWM シフトモード設定

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_dat: PWM シフトモードを設定します。

機能:

PWM シフトモード状態を VE レジスターに設定します。

補足:

特になし

戻り値:

なし

14. 2. 1. 27 VE_SetModulType

変調方式設定

API:

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

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。

詳細は 14. 1. 1. 1 を参照してください。

_dat: 変調方式を設定します。

機能:

変調方式を VE レジスタに設定します。

補足:

特になし

戻り値:

なし

14.2.2 データ構造

14.2.2.1 VE_InitTypeDef

Data Fields:

uint8_t	shunt: ショントタイプ 1: 1 ショント 3: 3 ショント
uint16_t	reptime: リピート回数(1 to 15 回)
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
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値
uint16_t	trgcomp0: トリガタイミング 0 設定 VExTRGCOMP0 に設定する値
uint16_t	trgcomp1: トリガタイミング 1 設定 VExTRGCOMP1 に設定する値
uint32_t	fmode: フロー制御設定 VExFMODE に設定する値
uint16_t	fpwmchg: PWM 切り換え速度設定 VExFPWMCHG に設定する値
uint16_t	piolim: PI 制御出力制限設定 VExPIOLIM に設定する値

14.3 モーター制御回路(PMD)

14.3.1 関数仕様

14.3.1.1 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 14.3.2.1PMD_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

14.3.1.2 PMD_GetEMG_Status

EMG 保護状態取得

API:

```
emg_status_e      PMD_GetEMG_Status(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14.1.1.1 を参照してください。

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: 保護中

14.3.1.3 PMD_ReleaseEMG_Protection

EMG 保護状態解除

API:

```
void      PMD_ReleaseEMG_Protection(const ipdrv_t* const _ipdrv)
```

引数:

_ipdrv: マイコン周辺回路アドレスが定義された構造体を指定します。
詳細は 14.1.1.1 を参照してください。

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

14.3.2 データ構造

14.3.2.1 PMD_InitTypeDef

Data Fields:

uint8_t	shunt: ショントタイプ
	1: 1 ショント
	3: 3 ショント
uint8_t	poll: L 側極性
	0: L active
	1: H active
uint8_t	polh: H 側極性
	0: L active
	1: H active
uint16_t	pwmrate: PWM 周波数
	PMDxRATE に設定する値
uint16_t	pwmfreq: PWM 周波数
	PMDxMDPRD に設定する値
uint16_t	deadtime: デッドタイム時間
	PMDxDTR に設定する値
uint8_t	busmode: モード選択
	PMDxMODESEL に設定する値

14.4 アナログデジタルコンバーター (ADC)

14.4.1 関数仕様

14.4.1.1 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 14.4.2.1AD_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

14.4.2 データ構造

14.4.2.1 AD_InitTypeDef

Data Fields:

`uint8_t` **shunt**: ショントタイプ
 1: 1 ショント
 3: 3 ショント

`uint8_t` **iuch**: U 相電流取り込み AD チャンネル番号 (3 ショント用)

`uint8_t` **ivch**: V 相電流取り込み AD チャンネル番号 (3 ショント用)

`uint8_t` **iwch**: W 相電流取り込み AD チャンネル番号 (3 ショント用)

`uint8_t` **idcch**: DC 電流取り込み AD チャンネル番号 (1 ショント用)

`uint8_t` **vdch**: モーターの電源電圧 Vdc 取り込み AD チャンネル番号

`uint8_t` **pmd_ch**: 使用する PMD チャンネル選択
 cPMD0: PMD チャンネル 0

`uint8_t` **pints**: PMD トリガー用割り込み選択
 cPINTS_A: INTADxPDA
 cPINTS_B: INTADxPDB

`uint32_t**` **p_adreg0**: AD 変換結果格納レジスターアドレス 0

`uint32_t**` **p_adreg1**: AD 変換結果格納レジスターアドレス 1

`uint32_t**` **p_adreg2**: AD 変換結果格納レジスターアドレス 2

`uint32_t**` **p_adreg3**: AD 変換結果格納レジスターアドレス 3

15. 付録

15.1 固定小数点処理

本サンプルソフトは、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。

15.2 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っています。

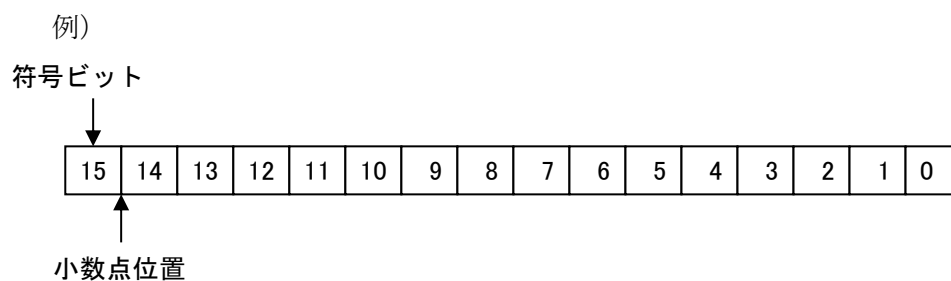


図 15.1 16 ビットデータの例

本アプリケーション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。

15.3 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 ビットデータでは 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 15.1 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 15.2 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	0.99999999534338	-1
Q30	30	1.99999999068670	-2
Q29	29	3.99999998137350	-4
Q28	28	7.99999996274700	-8
Q27	27	15.99999992549400	-16
Q26	26	31.99999985098800	-32
Q25	25	63.99999970197600	-64
Q24	24	127.99999940395000	-128
Q23	23	255.99999880790000	-256
Q22	22	511.99999761581000	-512
Q21	21	1023.99999523160000	-1024
Q20	20	2047.99999046320000	-2048
Q19	19	4095.99998092650000	-4096
Q18	18	8191.99996185300000	-8192
Q17	17	16383.99992370600000	-16384
Q16	16	32767.99984741200000	-32768
Q15	15	65535.99969482400000	-65536
Q14	14	131071.99938965000000	-131072
Q13	13	262143.99987793000000	-262144
Q12	12	524287.99975585900000	-524288
Q11	11	1048575.99951172000000	-1048576
Q10	10	2097151.99902344000000	-2097152
Q9	9	4194303.99804687000000	-4194304
Q8	8	8388607.99609375000000	-8388608
Q7	7	16777215.99218750000000	-16777216
Q6	6	33554431.98437500000000	-33554432
Q5	5	67108863.96875000000000	-67108864
Q4	4	134217727.93750000000000	-134217728
Q3	3	268435455.87500000000000	-268435456
Q2	2	536870911.75000000000000	-536870912
Q1	1	1073741823.50000000000000	-1073741824
Q0	0	2147483647.00000000000000	-2147483648

15.4 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、正しい小数点位置にする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。

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

15.5 assert 関数

本サンプルソフトには assert 関数が用意されております (DEBUG 設定時のみ)。ただし、関数処理はユーザーが設定する必要がございます。

(例)

```
#ifdef DEBUG
/*
 * @brief Deal with the error parameter
 * @param file: Pointer to the file where the error parameter locates
 * @param line: Number of the line in which the error parameter locates
 * @retval None
 */
void assert_failed(char *file, int32_t line)
{

}
#endif
```

16. 改訂履歴

日付	Rev.	変更内容
2026/05/29	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ェア、ソフトウェ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。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（生命直結機器）、車載・輸送機器、防衛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、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。