

# TMPM471

## 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 .....	7
2. システムブロック図 .....	9
3. ソースファイル構成 .....	10
4. 評価環境について .....	12
4.1 評価ボードの設定 .....	12
4.1.1 SBK-M471 ボードの設定 .....	12
4.1.2 KES-P2 ボードの設定 .....	12
4.1.3 KES-PH ボードの設定 .....	12
4.1.4 SBK-M471 と KES-P2(KES-PH)の接続 .....	13
4.1.5 DAC 出力の接続設定 .....	14
4.1.6 基板修正(KES-PH) .....	15
4.2 開発ツール .....	16
4.3 プロジェクトの立ち上げ方 .....	16
4.4 DAC 出力 .....	21
4.5 ポート出力 .....	21
4.5.1 ベクトル制御ソフトウェア処理時間 .....	22
4.5.2 電流検出状態 .....	22
4.6 ステージ履歴 .....	22
4.7 ユーザーインターフェースについて .....	23
5. モジュール構成 .....	25
6. マイコンハードウェア割り付け .....	26
6.1 周辺インターフェース .....	26
6.2 割り込み .....	27
7. ジェネラルフローチャート .....	28
7.1 メインルーチン (main) .....	28
7.2 メインループ (main_loop) .....	29
7.3 割り込み .....	30
7.3.1 ソフトウェアベクトル処理 (INT_VectorControl_bySoft) .....	31
8. 状態遷移 .....	32
8.1 センサーレス .....	32
8.2 エンコーダー .....	33
8.3 エンコーダー+ホールセンサー .....	34
9. ユーザーアプリ .....	35
9.1 ユーザー制御 .....	35
9.1.1 タイマーカウント(timer_count) .....	36
9.1.2 AD 値取得(soft_adc_getdata) .....	36

9.1.3	キー制御(Uikeyscan) .....	36
9.1.4	ユーザー設定(user_interface).....	36
9.1.5	LED 表示制御(led_display) .....	37
9.1.6	UART 送信データ設定(send_data_set).....	37
9.1.7	温度測定(temp_check).....	37
9.1.8	ステータス確認モニター .....	38
10.	機能説明.....	40
10.1	制御コマンド.....	40
10.1.1	制御方法 (usr.com_user).....	40
10.1.2	制御目標速度 .....	40
10.1.3	始動電流 .....	40
10.2	駆動コマンド.....	41
10.2.1	駆動方法 (drv.command) .....	41
10.2.2	ベクトル制御コマンド (drv.vector_cmd).....	41
10.3	駆動状態.....	43
10.3.1	エラー状態 (drv.state) .....	43
10.4	モーター制御構造体 .....	44
10.4.1	変数一覧 .....	44
10.5	関数詳細.....	48
10.5.1	エンコーダ初期設定 (init_ENCen) .....	48
10.5.2	ADC 初期設定 (init_ADCen).....	48
10.5.3	PMD 初期設定 (init_PMDen) .....	49
10.5.4	モーター制御初期設定 (B_Motor_Init) .....	49
10.5.5	DAC 制御初期設定 (init_Dac) .....	50
10.5.6	周期タイマー初期設定 (init_Timer_interval4kHz).....	52
10.5.7	ユーザー制御初期設定 (init_user_control).....	52
10.5.8	ユーザー制御 (user_control) .....	53
10.5.9	ユーザーモーター制御 (B_User_MotorControl) .....	54
10.6	モーター制御関数 .....	55
10.6.1	状態遷移処理関数 (C_Control_Ref_Model) .....	55
10.6.2	モーター制御共通処理関数 (C_Common) .....	57
10.6.3	停止状態関数 (C_Stage_Stop).....	58
10.6.4	ブートストラップ状態関数 (C_Stage_Bootstrap).....	59
10.6.5	位置決め状態関数 (C_Stage_Initposition) .....	60
10.6.6	強制転流状態関数 (C_Stage_Force) .....	61
10.6.7	強制定常切換状態関数 (C_Stage_Change_up) .....	62
10.6.8	定常状態関数 (C_Stage_Steady_A) .....	63
10.6.9	保護状態関数 (C_Stage_Emergency).....	64
10.7	モーター駆動関数 .....	65
10.7.1	用語説明 .....	65

10.7.2	モーター電流、電源電圧取得 (D_GetMotorCurrentPowerVolt) .....	66
10.7.3	入力座標変換 (D_InputTransformation) .....	69
10.7.4	クラーク変換 (E_Clarke) .....	70
10.7.5	パーク変換 (E_Park).....	71
10.7.6	位置推定関数 (D_Detect_Rotor_Position) .....	73
10.7.7	エンコーダー制御関数 (H_Encoder) .....	75
10.7.8	ホールセンサー+エンコーダー制御関数 (H_HallCon H_Hall_Encoder).....	77
10.7.9	速度制御関数 (D_Control_Speed) .....	78
10.7.10	電流制御関数 (D_Control_Current).....	79
10.7.11	出力座標変換 (D_OutputTransformation).....	80
10.7.12	逆パーク変換 (E_InvPark).....	82
10.7.13	セクター演算 (D_CalSector).....	83
10.7.14	逆クラーク変換 (E_InvClarke) .....	84
10.7.15	空間ベクトル変調 (D_SVM) .....	85
10.7.16	トリガータイミング演算 (D_CalTrgTiming).....	91
10.7.17	PWM レジスター設定 (PMD_RegDataSet).....	95
10.7.18	電流誤検知検出関数 (D_Check_DetectCurrentError) .....	96
10.7.19	インバーター出力電圧の算出関数 (Cal_Vdq).....	97
11.	定数定義説明 .....	98
11.1	モータードライバー設定用パラメーター共通 (D_Para.h) .....	98
11.1.1	モーター制御チャンネル選択 .....	98
11.1.2	DAC 出力選択 .....	98
11.1.3	疑似温度調整選択 .....	98
11.1.4	指令速度単位選択 .....	98
11.1.5	共通パラメーター一覧 .....	98
11.2	モータードライバー設定用パラメーターモーターチャンネル別 (D_Para_Chx.h) x=0,1 .....	99
11.2.1	制御選択 .....	99
11.2.2	モーターチャンネル別パラメーター一覧 .....	99
11.2.3	定数設定値と波形の関係 .....	105
11.3	ユーザー制御関連定数.....	106
12.	制御、データ更新のタイミング .....	110
12.1	ソフトウェアによるベクトル制御 .....	110
12.1.1	3 シャント制御 .....	110
12.1.2	1 シャント制御 .....	111
13.	ペリフェラルドライバー .....	112
13.1	マイコン周辺回路アドレス.....	112
13.1.1	データ構造 .....	112
13.2	モーター制御回路(PMD).....	112
13.2.1	関数仕様 .....	112
13.2.2	データ構造 .....	114

---

13.3 アナログデジタルコンバーター(ADC).....	115
13.3.1 関数仕様.....	115
13.3.2 データ構造.....	116
14. 付録.....	117
14.1 固定小数点処理.....	117
14.2 正規化 (Normalize).....	117
14.3 データフォーマット.....	118
14.4 固定小数点での演算.....	120
14.5 assert 関数.....	120
15. 改訂履歴.....	121
製品取り扱い上のお願い.....	122

## 1. 概要

### 1.1 はじめに

本モーター制御用サンプルソフトはイーエスピー企画製 SBK-M471(TMPM471F10FG マイコン評価ボード)と KES-P2,KES-PH (インバーターボード)を使用して動作確認を行っております。

本評価ボードの詳細については(株)イーエスピー企画(<http://esp.co.jp/>)にお問い合わせください。

### 1.2 仕様概要

◆ マイコン	.....	TMPM471F10FG
◆ クロック	.....	160MHz(ADC, A-PMD, A-ENC32) 80MHz(TSPI, T32A, UART)
◆ 使用モーター	.....	KES-P2 : ツカサ電工(株) – TG611B KES-PH : 三菱電機FA産業機器(株) – EM-AMF 0.1kW
◆ 開発環境	.....	IAR Embedded Workbench V9.70 Keil $\mu$ Vision MDK V5.43 SEGGER Embedded Studio V8.24
◆ 直流リンク電圧	.....	KES-P2 : DC24V
◆ 電源電圧	.....	KES-PH : AC100V

#### ◆ 制御概要

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

◆ モーター制御内容

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(スイッチ切替)
回転方向	CW 固定
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、1 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。

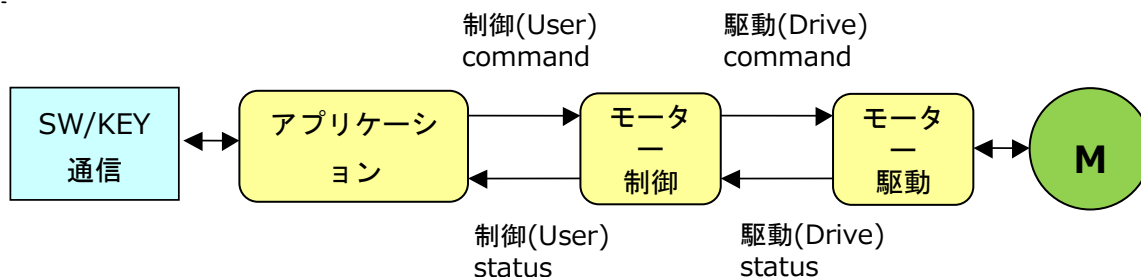


図 1.1 ベクトル制御ソフトウェアの構造

- 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。

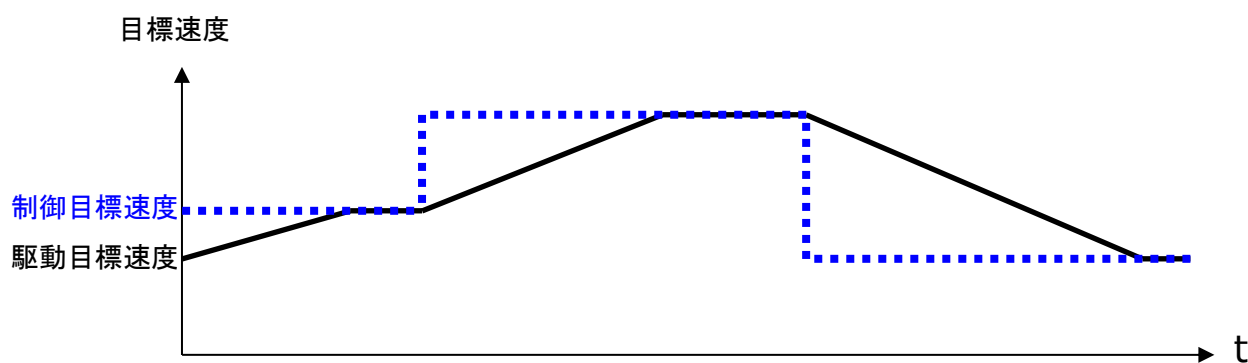


図 1.2 制御目標速度と駆動目標速度

2. システムブロック図

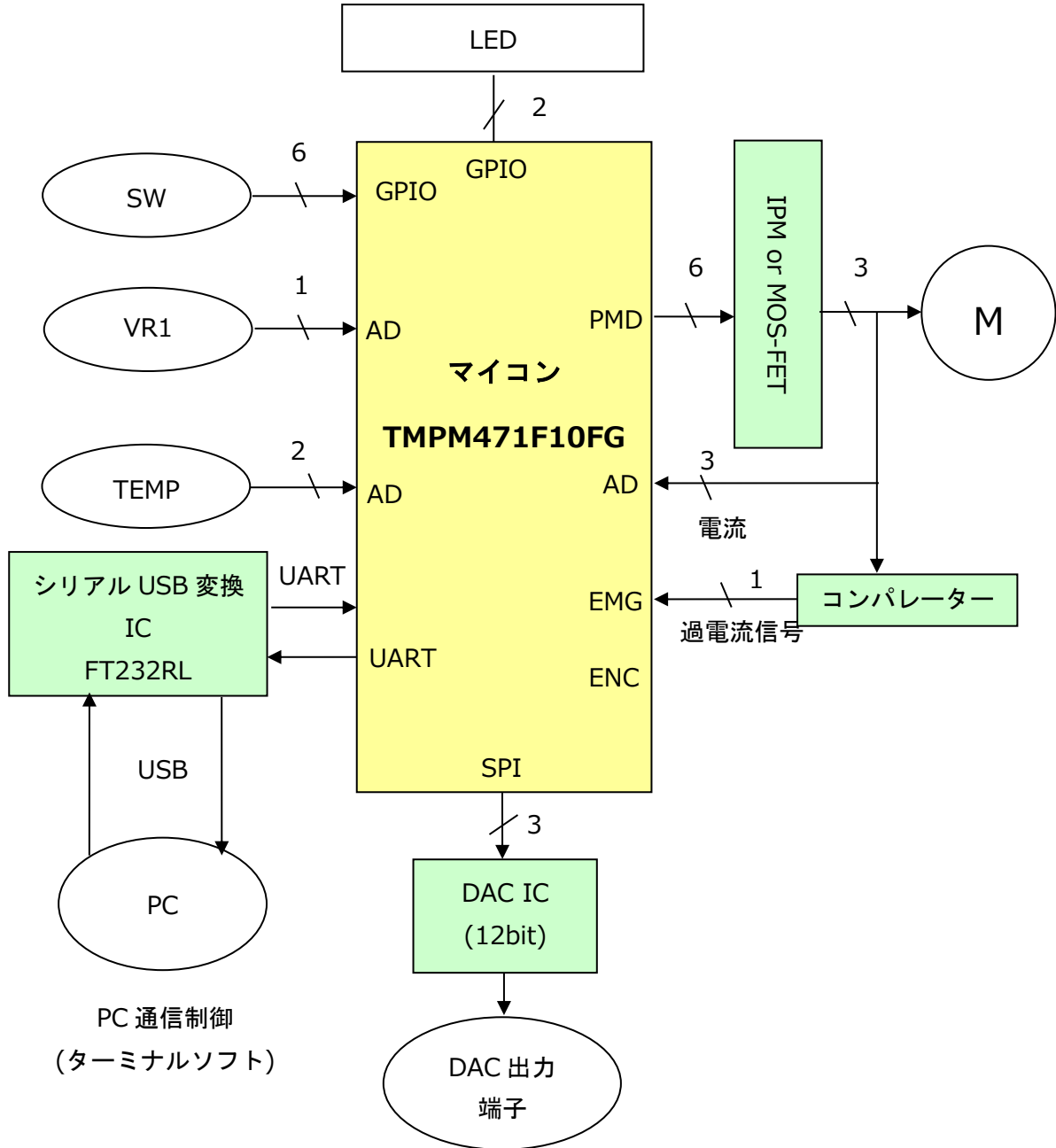


図 2.1 システムブロック図

## 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
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 ドライバースソースファイル
dac_drv.h	DAC IC ドライバースヘッダーファイル
D_Driver.c	ベクトル制御ドライバースソースファイル
D_Driver.h	ベクトル制御ドライバースヘッダーファイル
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 1)
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 1)
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 1)
D_Para_ch1.h	ベクトル制御パラメーター(チャンネル 1 用)ヘッダーファイル
E_Sub.c	演算用関数ソースファイル
E_Sub.h	演算用関数ヘッダーファイル
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スソースファイル
mcuip_drv.h	マイコンハード設定ドライバースヘッダーファイル
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 2)
usercon_KES-P2.c	ユーザー制御用ソースファイル(注 2)
usercon_KES-PH.c	ユーザー制御用ソースファイル(注 2)
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。

└─M471F10

└─CMSIS

CMSIS コアが格納されています。

		system_TMPM471F10.c	マイコン初期設定用ソースファイル		
		system_TMPM471F10.h	マイコン初期設定用ヘッダーファイル		
		TMPM471F10.h	マイコンレジスタ定義ヘッダーファイル		
		└─startup			
			└─arm		
				startup_TMPM471F10.s	スタートアップアセンブラソース (arm 用)
				TMPM471F10.scat	リンカーファイル (arm 用)
				└─iar	
				startup_TMPM471F10.s	スタートアップアセンブラソース (IAR 用)
				TMPM471F10.icf	リンカーファイル (IAR 用)
				└─segger	
				TMPM471F10_Vectors.s	スタートアップアセンブラソース (SEGGER 用)
				SEGGER_THUMB_Startup.s	スタートアップアセンブラソース (SEGGER 用)
				TX04_M471_Startup.s	スタートアップアセンブラソース (SEGGER 用)
				TX04_M471_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: 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 2: 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

#### 1.4 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。

##### 1.4.1 SBK-M471 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-M471 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-BOOT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	BOOT	M4KN-BOOT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 オープン オープン	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	

##### 1.4.2 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	

##### 1.4.3 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。

### 1.4.4 SBK-M471 と KES-P2(KES-PH)の接続

SBK-M471 と KES-P2 間は付属の 26pin フラットケーブルで接続します。

SBK-M471 と KES-PH 間には下図のように、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。

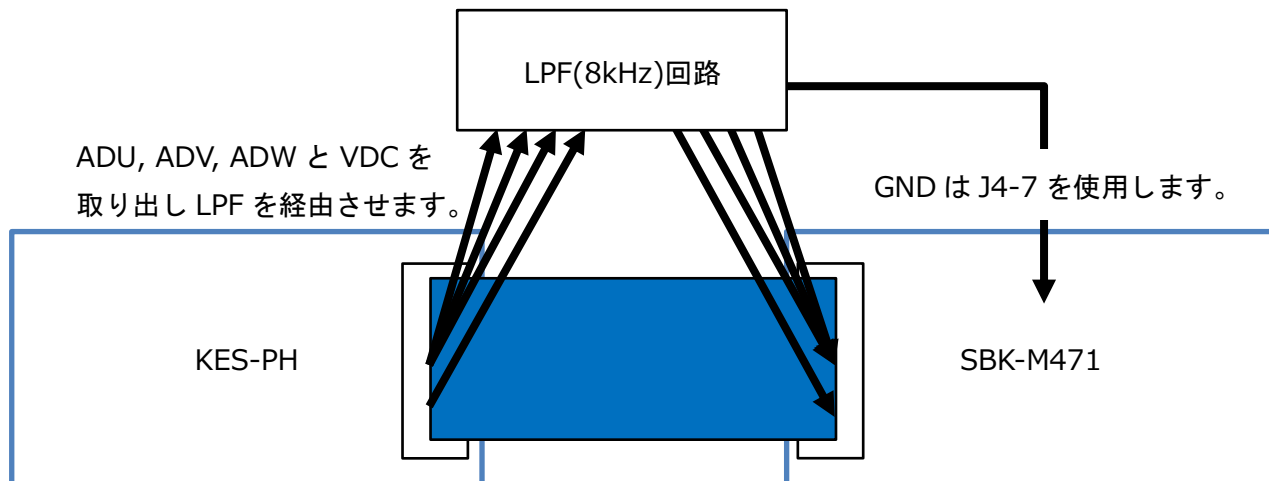


図 4.1 LPF の追加

### 1.4.5 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-M471 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-M471基板 J4	
番号	信号名		番号	信号名
1	DVDD	→	5	VDD
2	GND	→	6	GND
			SBK-M471基板 J7	
3	TSPI1RXD	→	5	TSPI2RXD/UT2RXD/PD6
4	TSPI1TXD	→	4	TSPI2TXD/UT2TXDA/PD5
5	TSPI1CLK	→	6	TSPI2SCK/PD4

### 1.4.6 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。

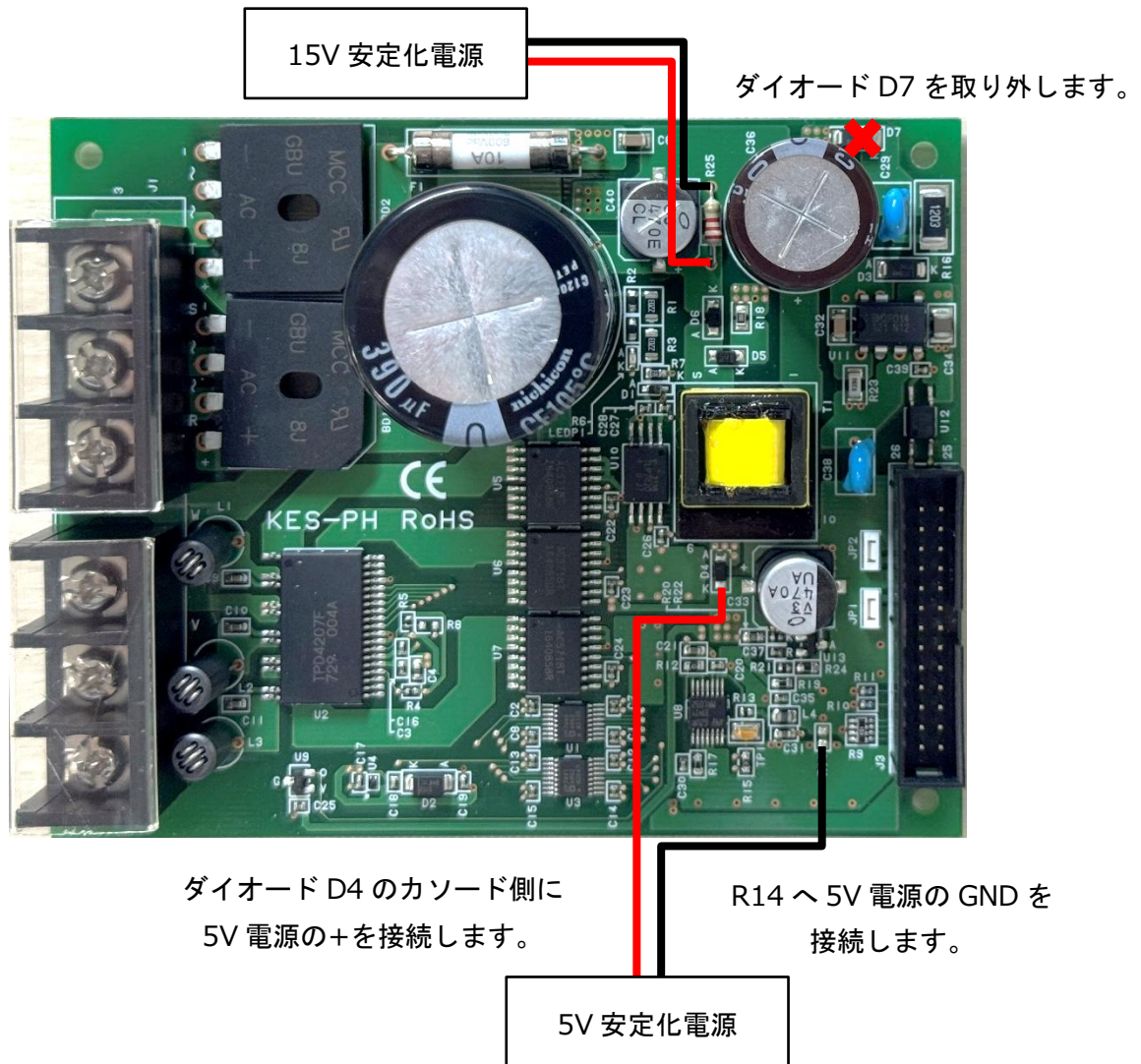


図 4.2 基板修正(KES-PH)

## 1.5 開発ツール

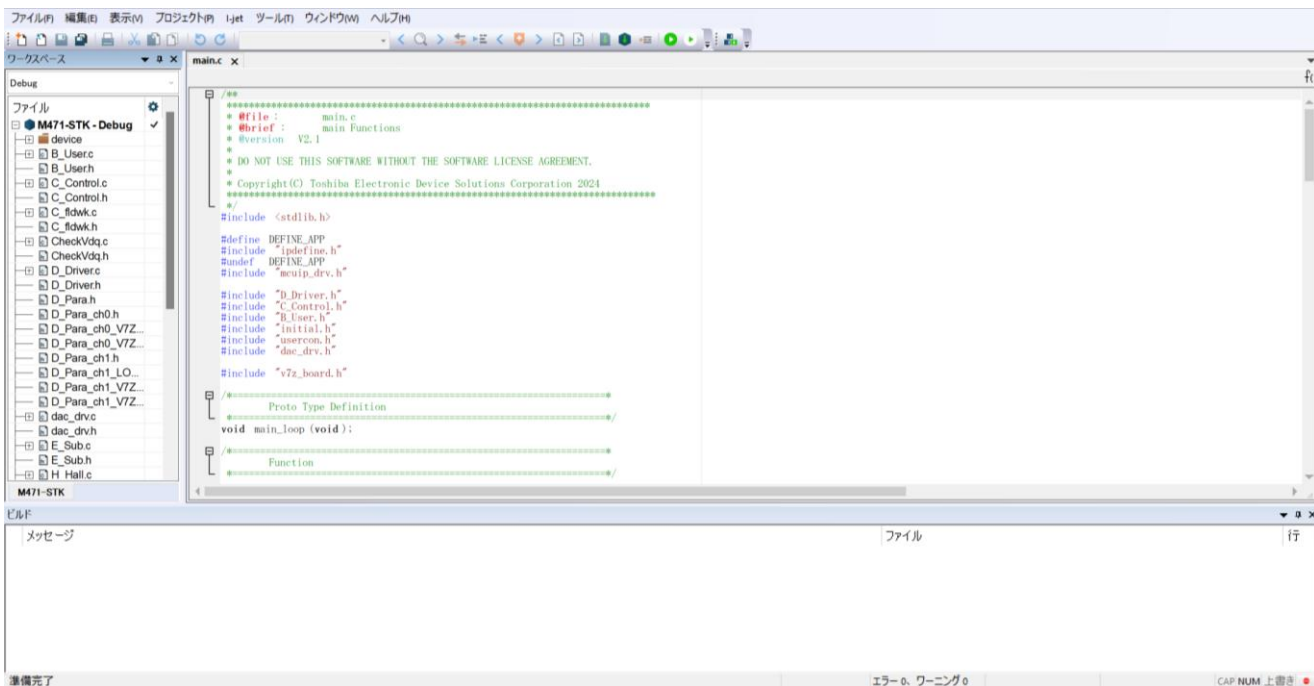
本ソフトは以下の開発ツールに対応しています。

IAR Embedded Workbench for ARM	9.70
Keil $\mu$ Vision MDK	5.43
Segger Embedded Studio	8.24

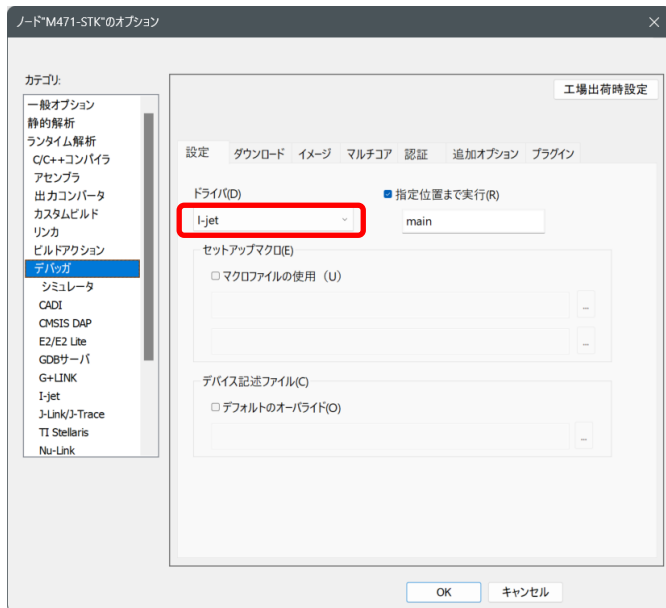
## 1.6 プロジェクトの立ち上げ方

IAR Embedded Workbench の場合を説明します。

1. `iar¥M471-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  
M471-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] > [デバッグ]の<設定>タブ



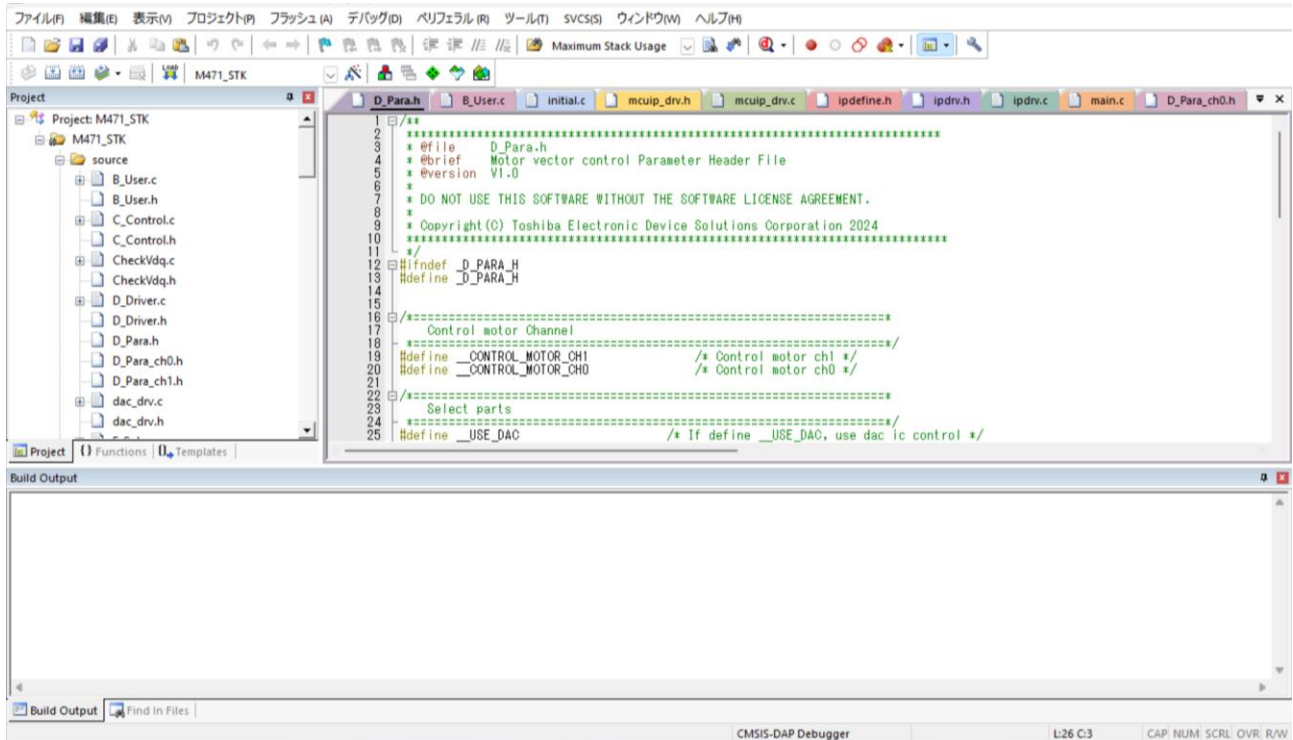
使用するツールを選択してください。

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。

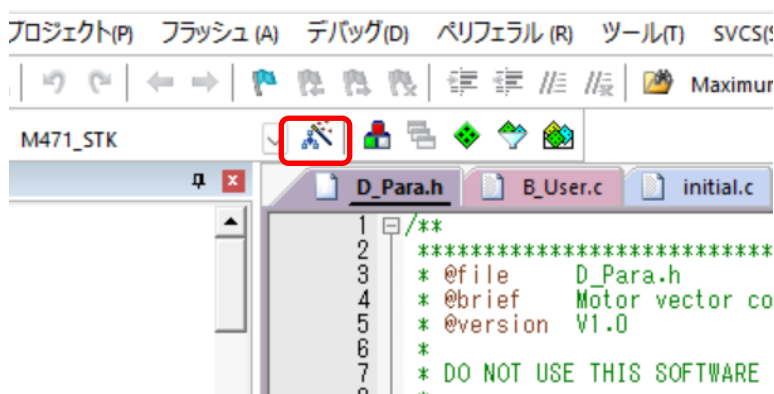


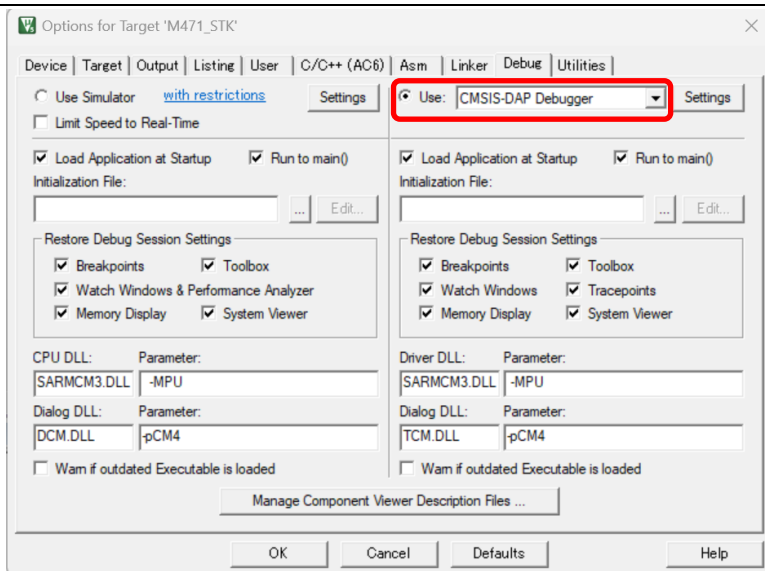
Keil μVision MDK の場合を説明します。

1. M471\_STK.uvprojx をダブルクリック、または[ファイル] > [開く]から M471\_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





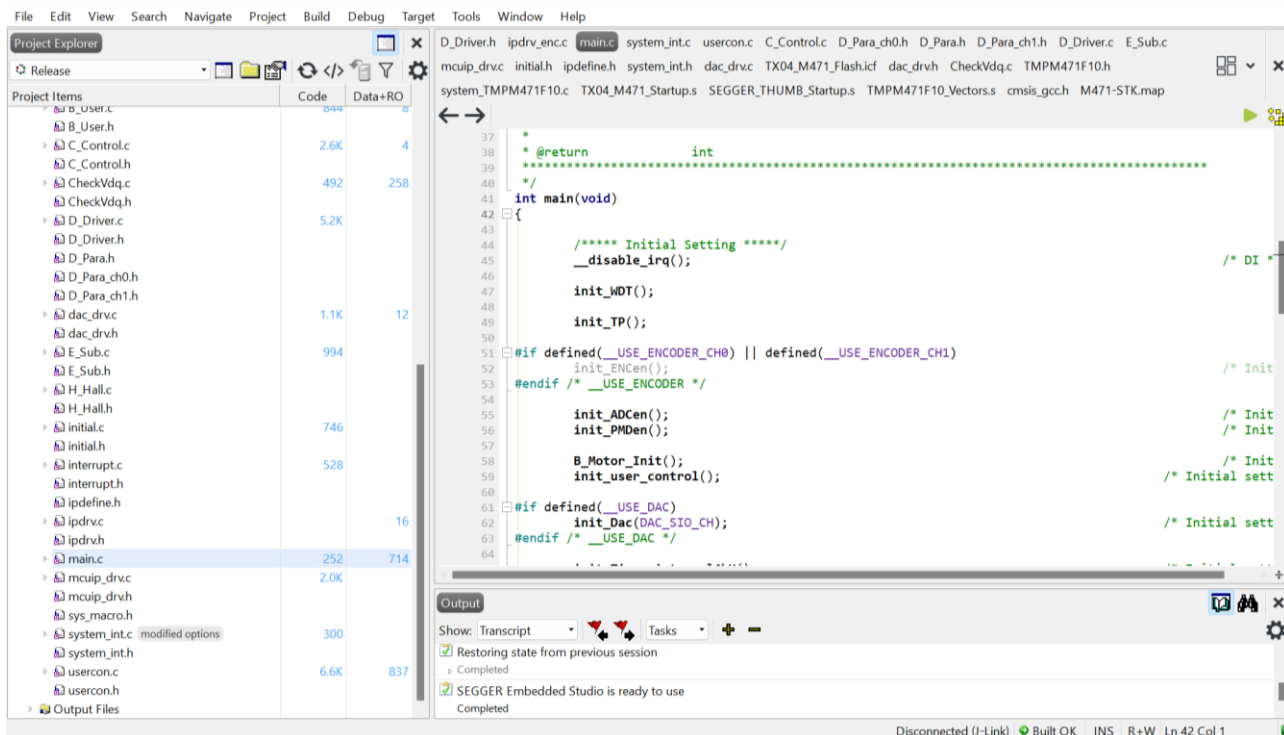
使用するツールを選択してください。

4. 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。



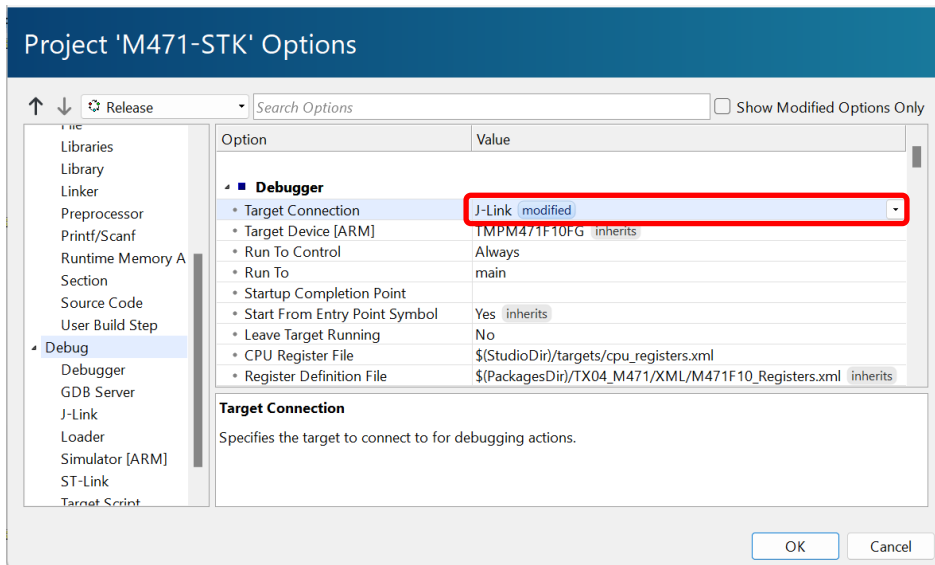
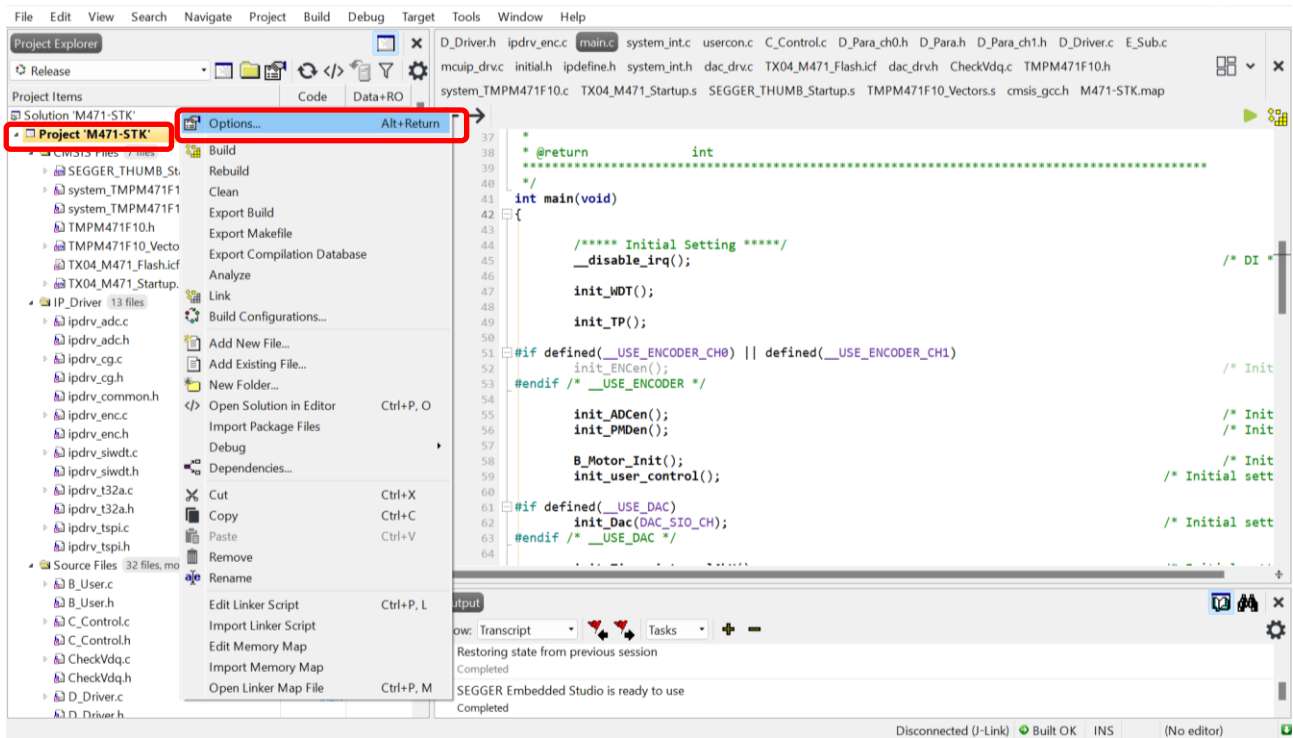
Segger Embedded Studio の場合を説明します。

1. M471-STK.emProject をダブルクリック、または[File] > [Open]から M471-STK.emProject を開いてください。
2. 下記画面が立ち上がります。



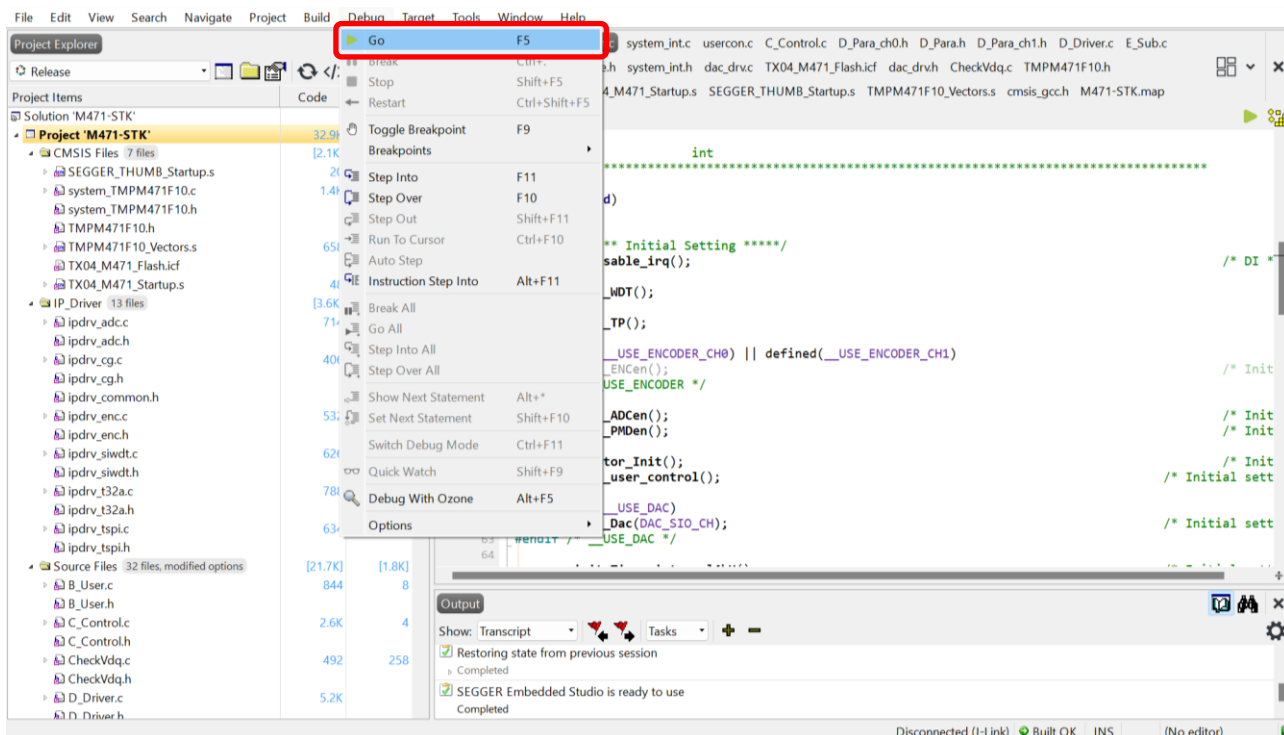
3. Options を開き、使用するツールの選択を行ってください。

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



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[Debug] > [Go]を選択してください。



### 1.7 DAC 出力

変数の変化をオシロスコープなどで見るための DAC 出力機能を実装しています。

DAC 出力を有効にするためには、D\_Para.h の下記定義を有効にしてください。

```
#define __USE_DAC
```

DAC 出力させる変数は、ファイル usercon.c の関数 UiOutDataStart()に記載しています。

確認する変数がない場合などは、必要に応じて追加してください。

《DAC 出力設定用変数》

dac.select	DAC 出力選択
dac.motch	DAC 出力させるモーターCH を設定してください。
dac.datsft0 - 3	データのシフト量を設定してください。
	計算式(x) : data × (2^x)

### 1.8 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。

### 1.8.1 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

モーターチャンネル	出力ポート
CH0	PL bit0
CH1	PL bit1

H : 演算中

### 1.8.2 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

モーターチャンネル	出力ポート
CH0、CH1	PL bit0

L : 正常検出状態、H : 誤検出状態

### 1.9 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
```

## 1.10 ユーザーインターフェースについて

ユーザーインターフェースとして下記機能を用意しています。

《ユーザーインターフェース内容》

### 1. モーターch0 操作

SW(S\_SW1)を切り替えることで、  
モーターch0 の動作を切り替えることができます。

off : 停止 on : 始動(0Hz)

### 2. モーターch1 操作

SW(S\_SW2)を切り替えることで、  
モーターch1 の動作を切り替えることができます。

off : 停止 on : 始動(0Hz)

### 3. 回転数/DAC 状態表示の操作対象切り替え

SW(S\_SW3)を切り替えることで、  
回転数/DAC 状態表示の操作対象を切り替えることができます。

off : モーターch0 on : モーターch1

### 4. 回転数 Up/Down 切り替え

SW(S\_SW4)を切り替えることで、  
回転数を Up させるか Down させるかを切り替えることができます。

off : Up on : Down

### 5. 回転数調整

SW(USW1)を押下することで、  
下記表に従い回転数を調整することができます。

0Hz(min)~100Hz(max)

表 4.3 回転数調整

No.	S_SW4	USW1 押下後
1	Off : Up	10Hz Up
2	On : Down	10Hz Down

## 6. DAC 表示

SW(USW2)を押下することで、

下記表に従い DAC 状態を表示することができます。

通常 : DAC 表示 3 秒間長押し : DAC No 切り替え

表 4.4 DAC 表示

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]

## 7. ボリューム

VR1 を操作することにより、疑似温度調整をすることができます。

疑似温度調整を使用にするためには、D\_Para.h の下記定義を有効にしてください。

```
#define __USE__FAKETEMP
```

## 8. “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)

モーターCH1 駆動時(LED5)

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。

## 9. 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 [9.1.8 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

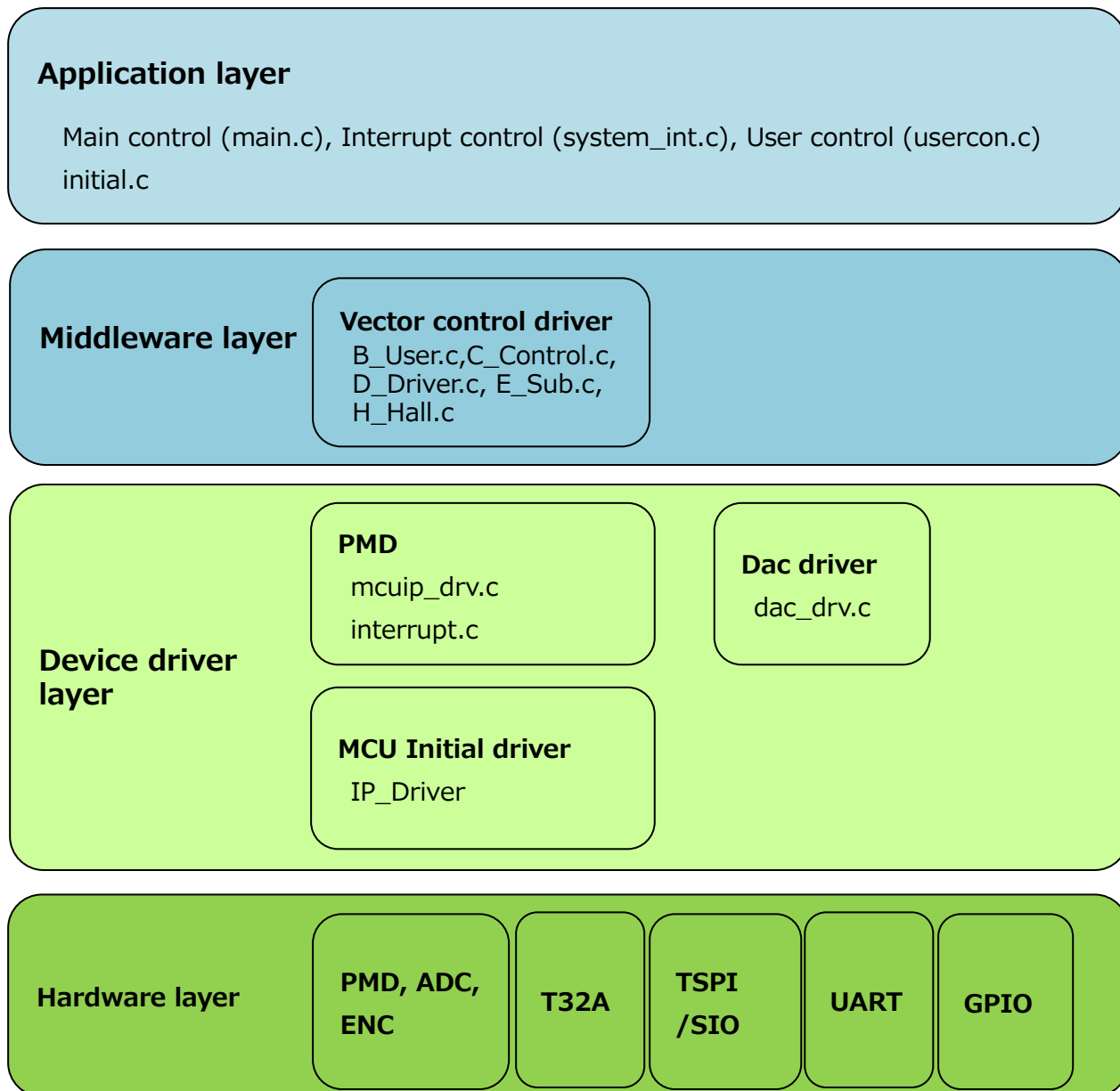


図 5.1 モジュール構成

## 6. マイコンハードウェア割り付け

## 1.11 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	
	チャンネル 5	未使用	
TSPI	チャンネル 0	未使用	
	チャンネル 1	未使用	
	チャンネル 2	DAC IC 制御	SIO モード
	チャンネル 3	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	
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	モーターCH1 コイル電流、電源電圧値 取得 TEMP1 電圧値取得	
PMD	チャンネル 0	モーターCH0 制御	
	チャンネル 1	モーターCH1 制御	
ENC	チャンネル 0	モーターCH0 エンコーダ信号制御	
	チャンネル 1	モーターCH1 エンコーダ信号制御	

### 1.12 割り込み

表 6.1 割り込み

要因名	処理内容	優先度	関数名
INTT32A0AC_IRQn	4kHz 周期タイミング作成	5	INTT32A0AC_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
INTADBPDB_IRQn	ベクトル制御ソフト処理 for CH1 (ソフトウェアによるベクトル制御用)	3	INTADBPDB_IRQHandler
INTSC0TX_IRQn	UART 送信完了割り込み処理	6	INTSC0TX_IRQHandler
INTSC2TX_IRQn	DAC IC 制御	6	INTSC2TX_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_ADCA_LEVEL         3      /* ADC A interrupt */
INT_ADCB_LEVEL         3      /* ADC B 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

### 1.13 メインルーチン (main)

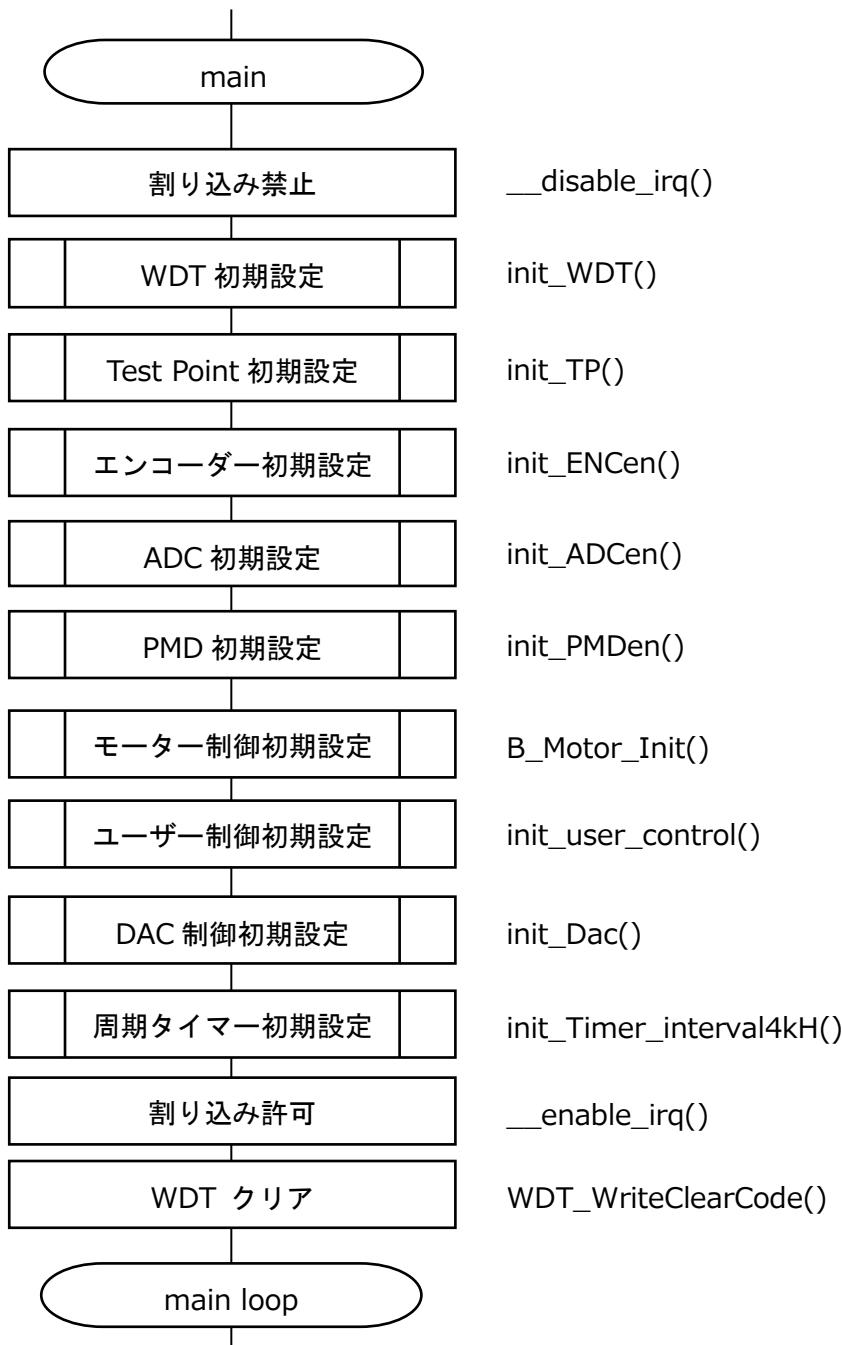


図 7.1 メインルーチン (main)

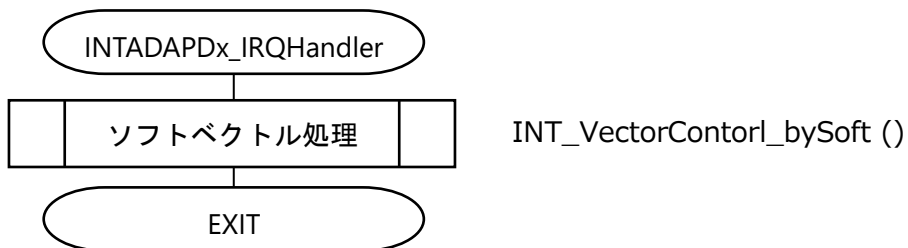
## 1.14 メインループ (main\_loop)



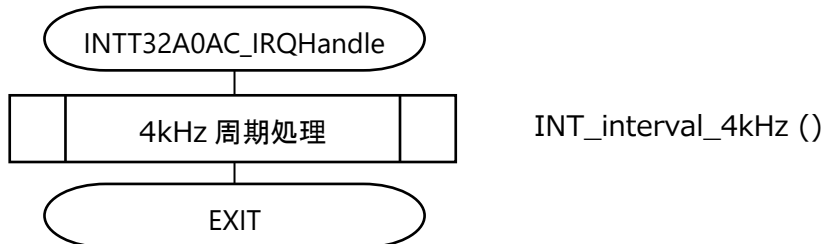
図 7.2 メインループ (main\_loop)

## 1.15 割り込み

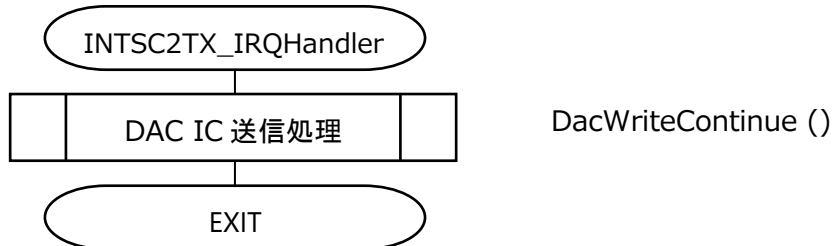
AD 変換終了割り込み(x=A,B)



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

1.15.1 ソフトウェアベクトル処理 (INT\_VectorControl\_bySoft)

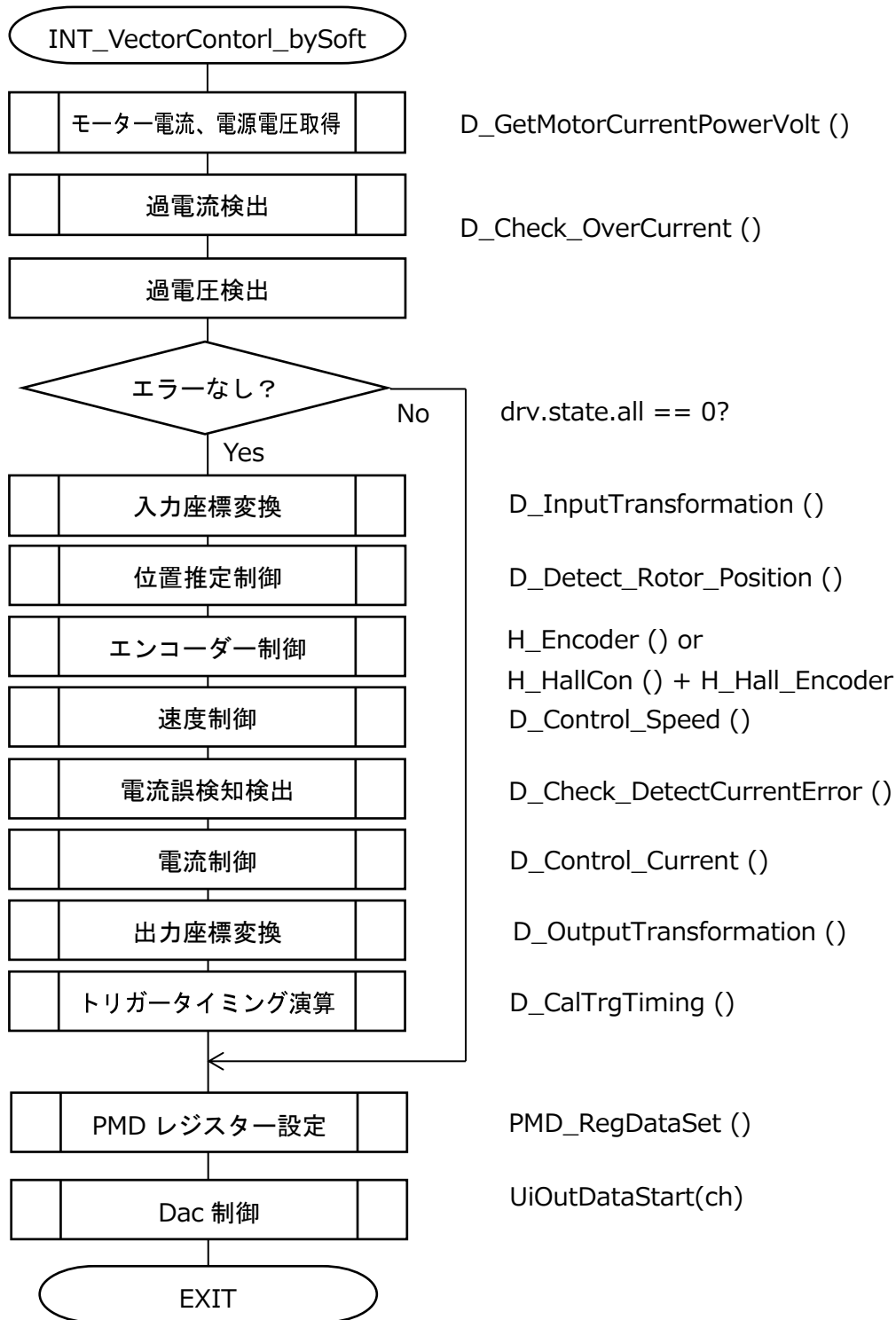


図 7.4 ソフトウェアベクトル処理 (INT\_VectorControl\_bySoft)

8. 状態遷移

1.16 センサーレス

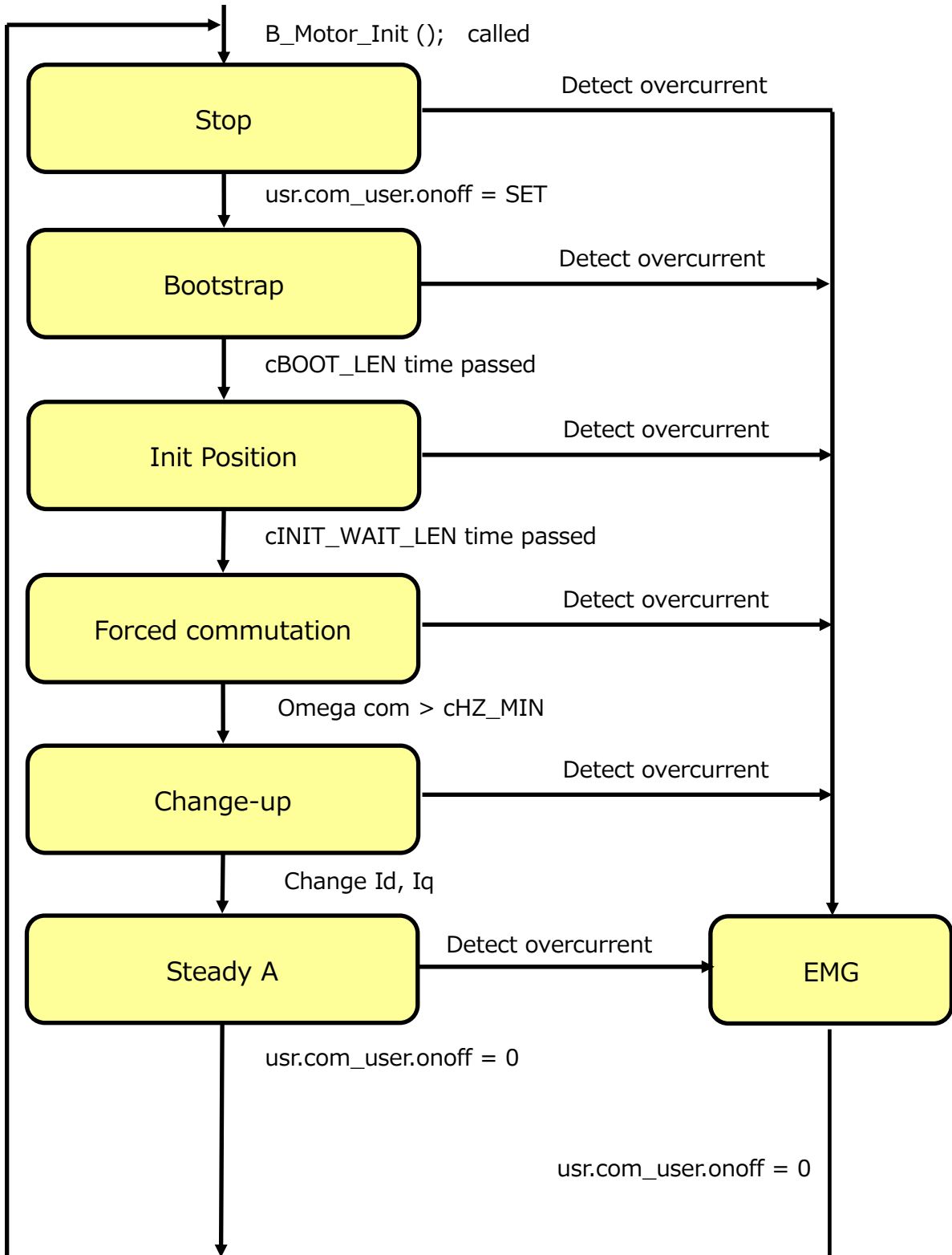


図 8.1 センサーレス

1.17 エンコーダー

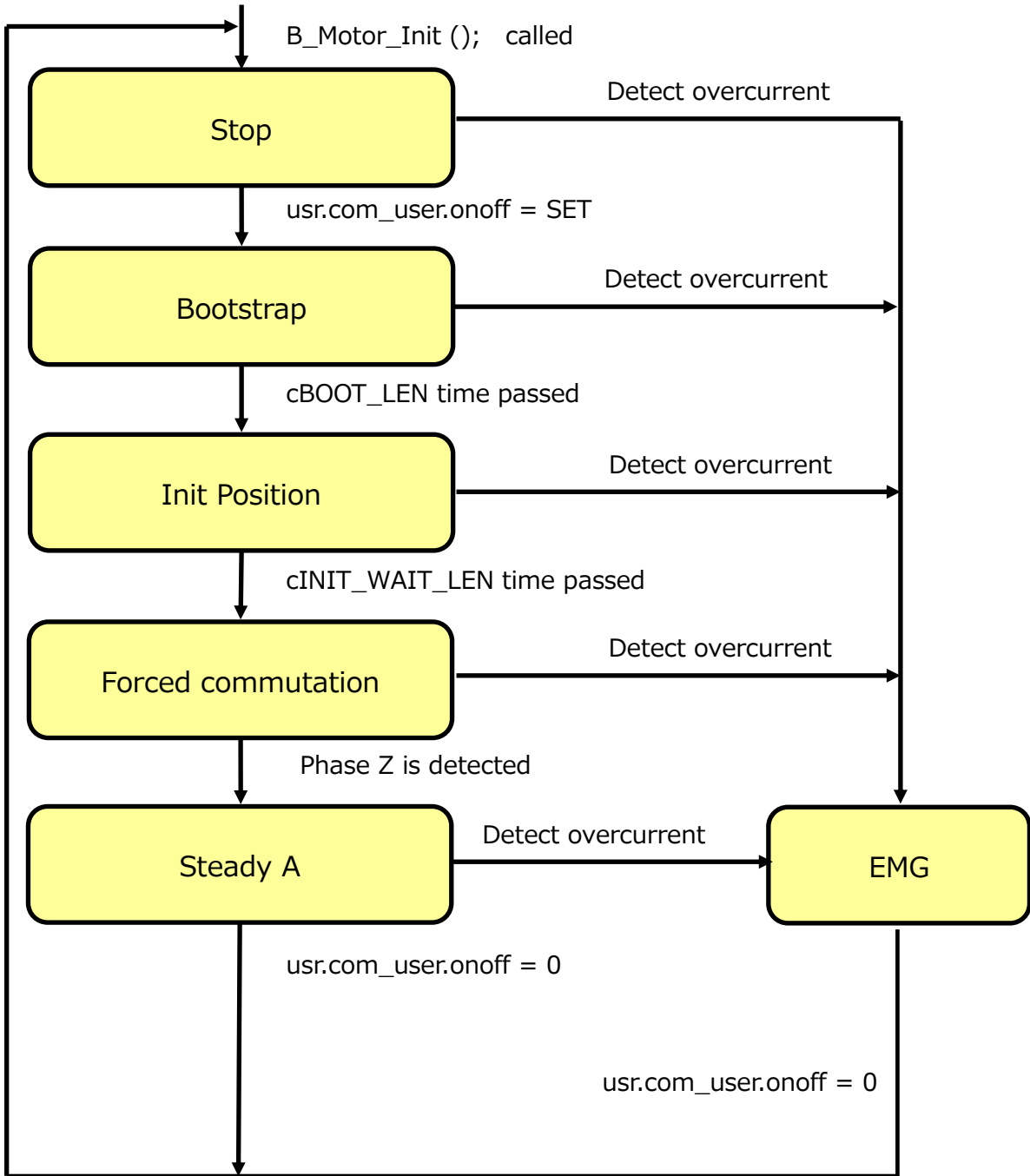


図 8.2 エンコーダー

1.18 エンコーダー+ホールセンサー

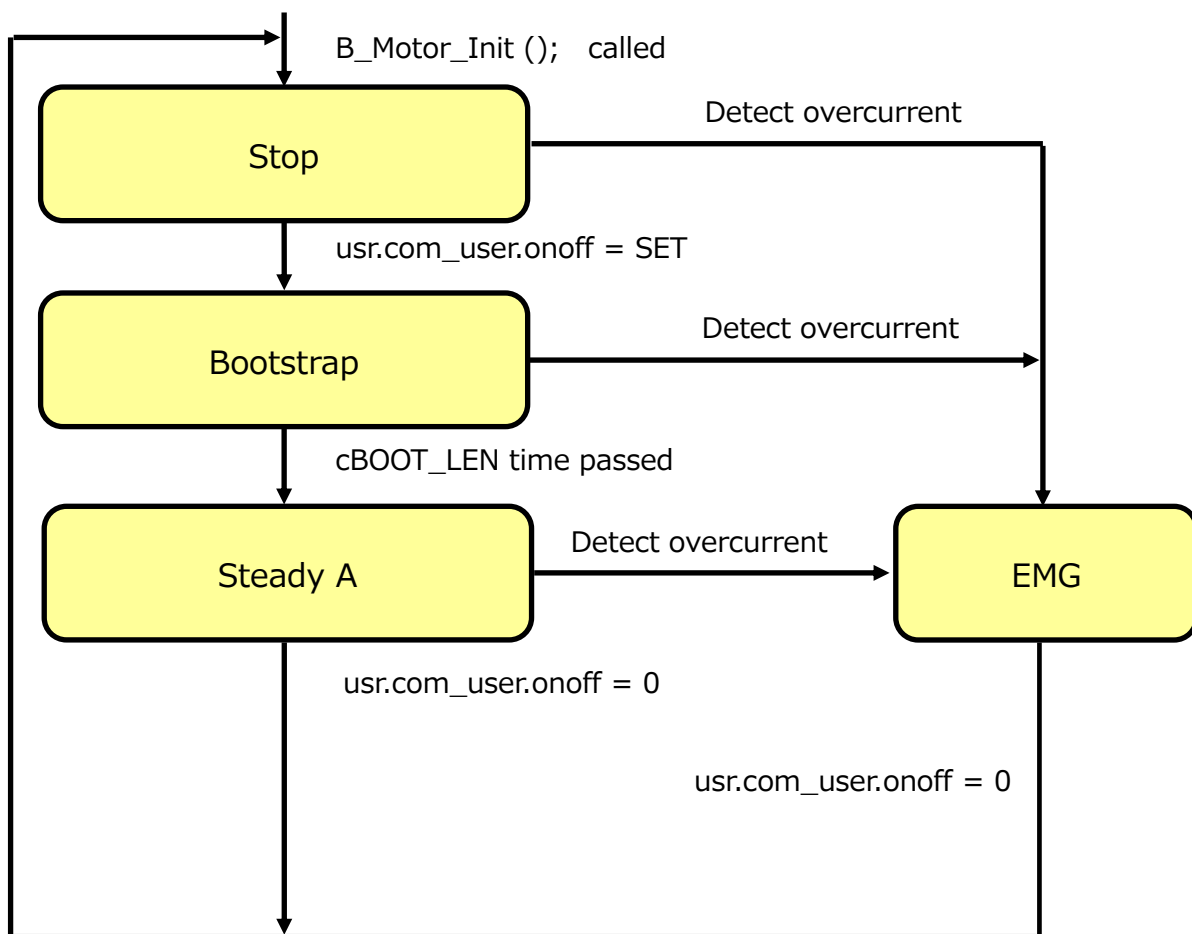


図 8.3 エンコーダー+ホールセンサー

## 9. ユーザーアプリ

### 1.19 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。

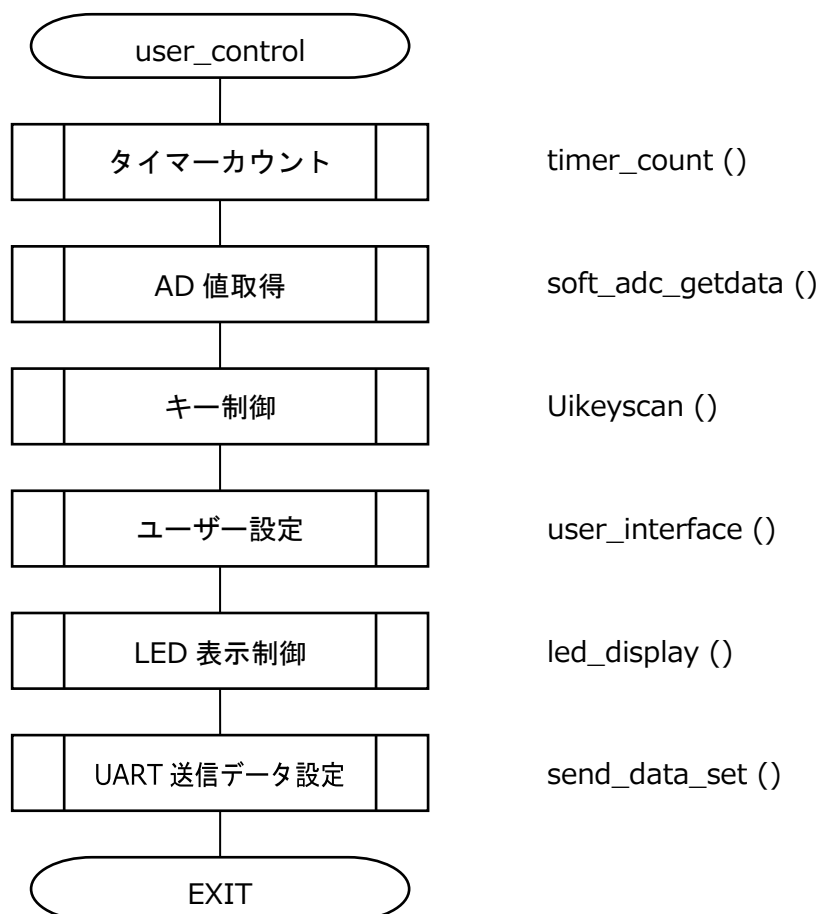


図 9.1 ユーザー制御

### 1.19.1 タイマーカウント(timer\_count)

KEY 処理の長押し判定のための 3 秒カウント処理を行います。

### 1.19.2 AD 値取得(soft\_adc\_getdata)

疑似温度(VR1)、モーターch0・ch1 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\_temp1.avedat に格納されます。

- ・疑似温度調整(\_\_USE\_FAKETEMP)が有効の場合

疑似温度(VR1)の値が ad\_temp0.avedat, ad\_temp1.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。

モーターch1 のインバーター温度が ad\_temp1.avedat に格納されます。

### 1.19.3 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。

USW2 の長押し確定時間は 3 秒以上、3 秒未満の場合は短押し確定です。

### 1.19.4 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. モーターch0 操作

sswdata.bit.sw1 の値に従い、Motor\_ch0.usr.com\_user.spd\_ctrl\_en を切り替えます。

sswdata.bit.sw1 が 1 の場合、on : 始動(0Hz)

sswdata.bit.sw1 が 0 の場合、off : 停止

#### 2. モーターch1 操作

sswdata.bit.sw2 の値に従い、Motor\_ch1.usr.com\_user.spd\_ctrl\_en を切り替えます。

sswdata.bit.sw2 が 1 の場合、on : 始動(0Hz)

sswdata.bit.sw2 が 0 の場合、off : 停止

#### 3. 回転数/DAC 状態表示の操作対象切り替え

sswdata.bit.sw3 の値に従い、disp\_ch、dac.motch を切り替えます。

sswdata.bit.sw3 が 1 の場合、on : モーターch1

sswdata.bit.sw3 が 0 の場合、off : モーターch0

#### 4. 回転数 Up/Down 切り替え

sswdata.bit.sw4 の値に従い、flg.motor.motor\_spd\_updown を切り替えます。

sswdata.bit.sw4 が 1 の場合、on : Down

sswdata.bit.sw4 が 0 の場合、off : Up

#### 5. 回転数調整

uswdata.bit.sw1 が 1 となるたびに target\_spdx(x = 0,1)の値を更新します。

0 : 0(0Hz)~10(100Hz)

定数定義値 : cSPD\_UPDOWN\_RESOLUTION を使用し、

target\_spdx(x = 0,1)を更新。

例)

target\_spd : 0→最低回転数→…→90→100(最大回転数)

#### 6. DAC 表示

uswdata.bit.sw2 が 1 となるたびに DAC 表示を行います。

uswdata.bit.sw2 が長押しされるたびに dac.selectchx(x = 0,1)を切り替えます。

### 1.19.5 LED 表示制御(led\_display)

EMG ステータスに従い、LED4、LED5 の制御を行います。

詳細は [4.6 ユーザーインターフェースについて](#)を参照ください

### 1.19.6 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。

### 1.19.7 温度測定(temp\_check)

VR1、TEMP0,TEMP1 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°C × 68 / 351 = 1.9°C (1.9373…)

30°C + 1.9°C = 31.9°C

- ・温度範囲

-20℃～100℃までとし、範囲外はエラーを表示します。

### 1.19.8 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

S\_SW3 で選択されている制御 ch の初期ステータスを表示します。

制御 ch = CH0 または CH1

キャリア一周波数 = xxxxx Hz

デッドタイム = x.x μs

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)

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(TEMP0,TEMP1)を読み込み演算

- ・初回以降表示内容

モーター回転指示が発生後に、

S\_SW3 で選択されている制御 ch の現在の回転速度を 1 秒ごとに表示します。

例)

40Hz

60Hz

80Hz

:

## ・ EMG 表示内容

EMG 状態になった場合、

S\_SW3 で選択されている制御 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 または CH1

キャリア一周波数 = xxxxx Hz

デッドタイム = x.x  $\mu$ 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)

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 S または プッシュ SW

注 1: AD(VDC)を読み込み演算

注 2: AD(TEMP0,TEMP1)を読み込み演算

## 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。

### 1.20 制御コマンド

制御コマンドを以下に記します。

#### 1.20.1 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）

```
typedef struct {  
    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。

#### 1.20.2 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。

#### 1.20.3 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。

### 1.21 駆動コマンド

駆動コマンドを以下に記します。

#### 1.21.1 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。

#### 1.21.2 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 $\omega$ とします。CLEAR では指令値を速度 $\omega$ 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。

## 1.22 駆動状態

### 1.22.1 エラー状態 (drv.state)

```
typedef union {  
    struct {  
        uint16_t reserve:11;    /* reserve */  
        uint16_t Loss_sync:1;   /* 0:normal, 1: Loss of synchronism */  
        uint16_t emg_DC:1;     /* 0:normal, 1: 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) */  
    } 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。

### 1.23 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 */
vector_t  Motor_ch1;    /* Motor data for ch1 */
```

#### 1.23.1 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	ロータ一位置	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	---	(1.21.1 参照)
vectorcmd_t	drv.vector_cmd	ベクトル制御コマンド	---	(1.21.2 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	
q31_u	usr.omega_user	制御目標速度	Q31	
q15_t	usr.Id_st_user	始動 Id 電流値	Q15	
q15_t	usr.Iq_st_user	始動 Iq 電流値	Q15	
uint16_t	usr.lambda_user	初期ロータ一位置	Q0	
command_t	usr.com_user	制御方法	---	(1.20.1 参照)
command_t	usr.com_user_1	制御方法前回	---	
q15_t	para.omega_min	強制転流終了速度	Q15	
q15_t	para.omega_v2i	電流制御切換速度	Q15	
q31_t	para.vd_pos	位置決め指令電圧	Q31	
q31_t	para.spd_coef	出力電圧係数	Q15	
q31_u	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
q31_u	para.sp_up_lim_s	駆動速度増加リミット値	Q31	

q31_u	para.sp_dn_lim_s	駆動速度減少リミット値	Q31	
uint16_t	para.time.initpos	位置決め時間	Q0	
uint16_t	para.time.initpos2	位置決め状態待ち時間	Q0	
uint16_t	para.time.bootstp	ブートストラップ時間	Q0	
uint16_t	para.time.go_up	強制定常切換後待ち時間	Q0	
q31_t	para.iq_lim	q 軸電流制限値	Q31	
q31_t	para.id_lim	d 軸電流制限値	Q31	
q15_t	para.err_ovc	過電流設定値	Q15	
q31_t	para.pos.kp	位置推定比例ゲイン	Q15	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_adjust	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	
uint32_t	stage_counter	ステージカウンター	Q0	
shunt_type_e	shunt_type	シャントタイプ	---	c1shunt c3shunt
boot_type_e	boot_type	起動タイプ	---	cBoot_i cBoot_v

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 変換結果レジスターアドレス	---	
uint32_t*	drv.ADxREG1	AD 変換結果レジスターアドレス	---	
uint32_t*	drv.ADxREG2	AD 変換結果レジスターアドレス	---	
uint32_t*	drv.ADxREG3	AD 変換結果レジスターアドレス	---	
q15_t	drv.Vdc_adc	電源電圧変換結果	Q15	
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	$\alpha$ 軸電流	Q15	
q15_t	drv.Ibeta	$\beta$ 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	$\alpha$ 軸電圧	Q31	
q31_u	drv.Vbeta	$\beta$ 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.TrigComp	トリガータイミング補正值	Q15	
trgpos_e	para.TrigPosMd	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)

### 1.24 関数詳細

#### 1.24.1 エンコーダー初期設定 (init\_ENCen)

##### 1.24.1.1 構文

void init\_ENCen(void)

引数 :

なし

戻り値 :

なし

##### 1.24.1.2 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

#### 1.24.2 ADC 初期設定 (init\_ADCen)

##### 1.24.2.1 構文

void init\_ADCen(void)

引数 :

なし

戻り値 :

なし

##### 1.24.2.2 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

## 1.24.3 PMD 初期設定 (init\_PMDen)

### 1.24.3.1 構文

```
void init_PMDen(void)
```

引数 :

なし

戻り値 :

なし

### 1.24.3.2 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。

## 1.24.4 モーター制御初期設定 (B\_Motor\_Init)

### 1.24.4.1 構文

```
void B_Motor_Init(void)
```

引数 :

なし

戻り値 :

なし

### 1.24.4.2 変数

方向	名前	意味	Q フォーマット	備考
出力	shunt_type	シャントタイプ	---	
	boot_type	駆動タイプ	---	
	usr.com_user.encoder	エンコーダー制御	---	
	stage.main	メインステージ	---	
	stage.sub	サブステージ	---	
	drv.Iao_ave	U 相ゼロ電流平均 AD 値	Q0	
	drv.Ibo_ave	V 相ゼロ電流平均 AD 値	Q0	
	drv.Ico_ave	W 相ゼロ電流平均 AD 値	Q0	
	drv.Idco_ave	ゼロ電流変換結果	Q15	1 シャント時だけ
	usr.Iq_st_user	始動 Iq 電流値	Q15	
	usr.Id_st_user	始動 Id 電流値	Q15	
	usr.lambda_user	初期ローター位置	Q0	
	para.motor.r	モーター巻線抵抗	Q15	
	para.motor.Lq	モーター q 軸インダクタンス	Q15	
	para.motor.Ld	モーター d 軸インダクタンス	Q15	
	para.chkpls	電流検出保護 Duty 幅	Q0	
	para.vd_pos	位置決め指令電圧	Q31	電圧起動時だけ
para.spd_coef	出力電圧係数	Q15	電圧起動時だけ	

para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
para.sp_up_lim_s	駆動速度増加リミット値	Q31	
para.sp_dn_lim_s	駆動速度減少リミット値	Q31	
para.time.bootstp	ブートストラップ時間	Q0	
para.time.initpos	位置決め時間	Q0	
para.time.initpos2	位置決め状態待ち時間	Q0	
para.time.go_up	強制定常切換後待ち時間	Q0	
para.omega_min	強制転流終了速度	Q15	
para.omega_v2i	電流制御切換速度	Q15	
para.delta_lambda	強制定常切換時電流切換位相	Q0	
para.pos.ki	位置推定積分ゲイン	Q15	Q12 対応あり
para.pos.kp	位置推定比例ゲイン	Q15	Q12 対応あり
para.pos.ctrlprd	位置推定制御周期	Q16	
para.spd.ki	速度制御積分ゲイン	Q15	Q12 対応あり
para.spd.kp	速度制御比例ゲイン	Q15	Q12 対応あり
para.current.dki	d 軸電流比例ゲイン	Q15	Q12 対応あり
para.current.dkp	d 軸電流積分ゲイン	Q15	Q12 対応あり
para.current.qki	q 軸電流比例ゲイン	Q15	Q12 対応あり
para.current.qkp	q 軸電流積分ゲイン	Q15	Q12 対応あり
para.iq_lim	q 軸電流制限値	Q31	
para.id_lim	d 軸電流制限値	Q31	
para.err_ovc	過電流設定値	Q15	
para.enc.pls2theta	パルス数から角度への演算係数	Q32	
para.enc.deg_adjust	電気角調整	Q0	
para.enc.plsnum	エンコーダーパルス数	Q0	
para.enc.pls2omega	パルス数から速度への演算係数	Q15	
para.enc.ctrlprd	エンコーダー制御周期	Q16	
para.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	

#### 1.24.4.3 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。

#### 1.24.5 DAC 制御初期設定 (init\_Dac)

##### 1.24.5.1 構文

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

引数：

なし

戻り値：

なし

### 1.24.5.2 処理内容

DAC IC 制御の初期設定を行います。

- SIO 許可

- ポート初期設定

- SIO 初期設定

- DAC IC 初期化通信

- SIO 割り込みレベル設定

- SIO 割り込み保留クリア

- 送信割り込み許可

### 1.24.6 周期タイマー初期設定 (init\_Timer\_interval4kH)

#### 1.24.6.1 構文

```
void init_Timer_interval4kH(void)
```

引数 :

なし

戻り値 :

なし

#### 1.24.6.2 処理内容

4kHz 周期タイミングを作成するためのタイマーの初期設定を行います。

### 1.24.7 ユーザー制御初期設定 (init\_user\_control)

#### 1.24.7.1 構文

```
void init_user_control(void)
```

引数 :

なし

戻り値 :

なし

#### 1.24.7.2 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)

## 1.24.8 ユーザー制御 (user\_control)

### 1.24.8.1 構文

```
void user_control(void)
```

引数 :

なし

戻り値 :

なし

### 1.24.8.2 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、モーターch0 操作
- ・ S\_SW2 による、モーターch1 操作
- ・ S\_SW3 による、回転数/DAC 状態表示の操作対象の切換
- ・ S\_SW4 による、回転数 Up/Down 切換
- ・ USW1 による、回転数調整
- ・ USW2 による、DAC 表示
- ・ VR1 による、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: SW 切り替え直後はモーター動作を停止します。

- ・ アナログ入力電圧取得

疑似温度(VR1)、モーターch0・ch1 のインバーター温度(TEMP0/TEMP1)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。

## 1.24.9 ユーザーモーター制御 (B\_User\_MotorControl)

## 1.24.9.1 構文

void B\_User\_MotorControl(void)

引数：

なし

戻り値：

なし

## 1.24.9.2 変数

方向	名前	意味	Q フォーマット	備考
入力	target_spd	目標速度	---	ユーザー指令
	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	---	
	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

## 1.24.9.3 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。

## 1.25 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。

### 1.25.1 状態遷移処理関数 (C\_Control\_Ref\_Model)

#### 1.25.1.1 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

#### 1.25.1.2 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	---	

#### 1.25.1.3 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。

注意: 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。

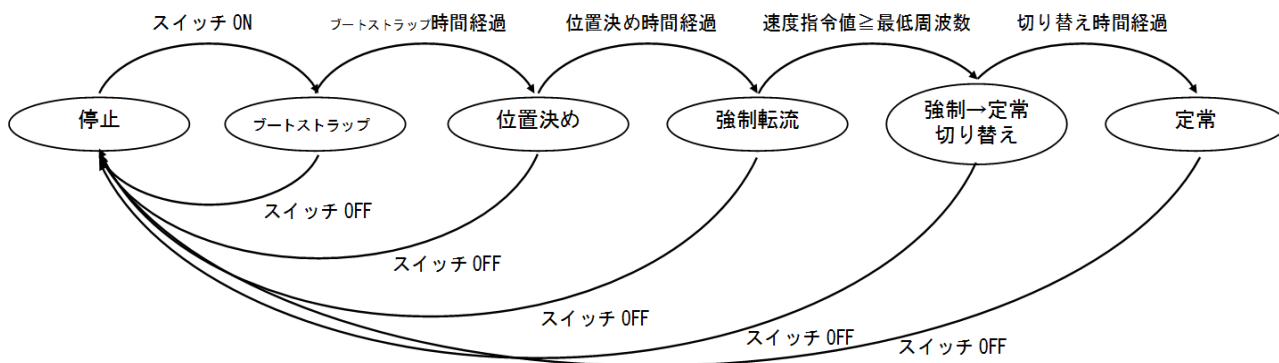


図 10.1 モーター制御の状態遷移図

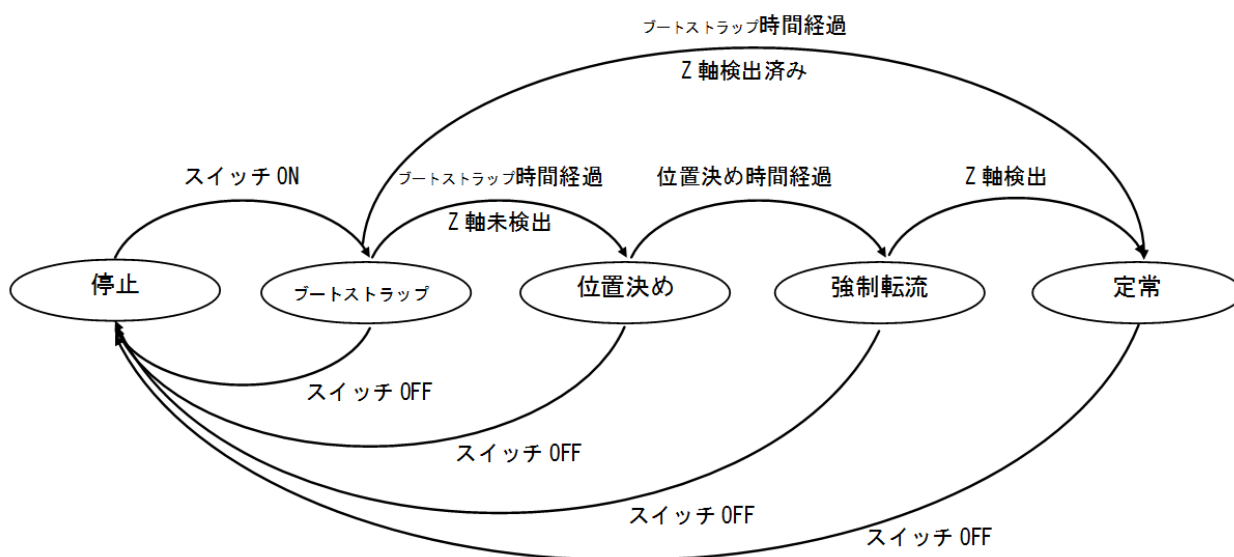


図 10.2 モーター制御の状態遷移図 (エンコーダー制御)

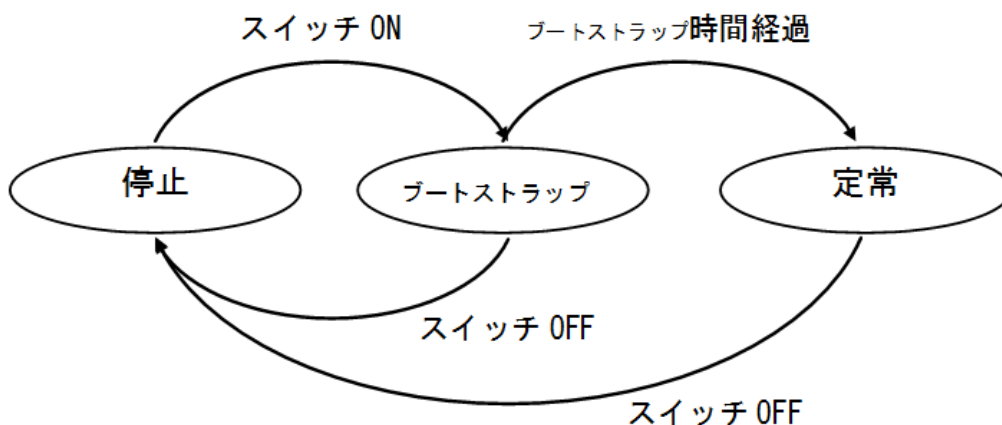


図 10.3 モーター制御の状態遷移図 (ホールセンサー+エンコーダー制御)

### 1.25.2 モーター制御共通処理関数 (C\_Common)

#### 1.25.2.1 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

#### 1.25.2.2 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 or3 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 or3 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	---	[%]

#### 1.25.2.3 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。

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

### 1.25.3 停止状態関数 (C\_Stage\_Stop)

#### 1.25.3.1 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

#### 1.25.3.2 変数

方向	名前	意味	Qフォーマット	備考
入力	stage.main	メインステージ	---	
入出力	stage.sub	サブステージ	---	
出力	stage.itr	割り込みステージ	---	
	drv.vector_cmd	ベクトル制御コマンド	---	(1.21.2 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	

#### 1.25.3.3 処理内容

モーターを停止させます。(PWM 出力を停止します)

### 1.25.4 ブートストラップ状態関数 (C\_Stage\_Bootstrap)

#### 1.25.4.1 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

#### 1.25.4.2 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	---	(1.21.2 参照)

#### 1.25.4.3 処理内容

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

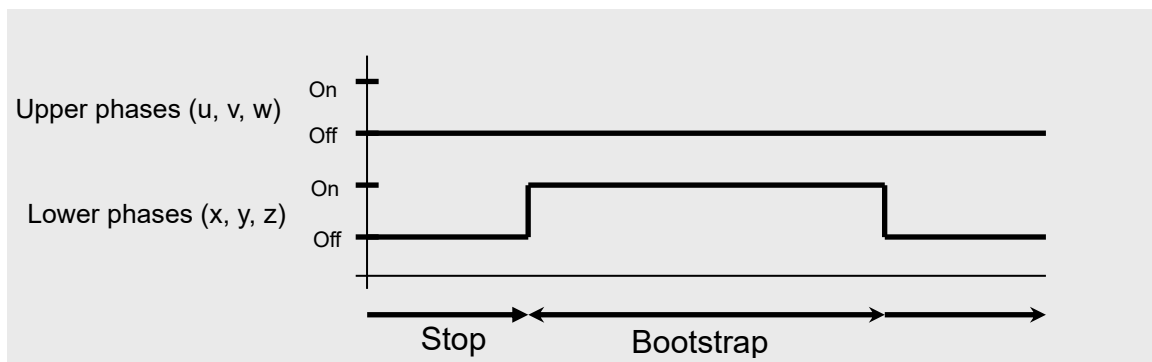


図 10.4 ブートストラップ時間

### 1.25.5 位置決め状態関数 (C\_Stage\_Initposition)

#### 1.25.5.1 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

#### 1.25.5.2 変数

方向	名前	意味	Qフォーマット	備考
入力	stage.main	メインステージ	---	
	boot_type	起動タイプ	---	
	usr.Id_st_user	始動 Id 電流値	Q15	
	usr.lambda_user	初期ローター位置	Q0	
	para.vd_pos	位置決め指令電圧	Q31	
	para.time.initpos	位置決め時間	Q0	* MainLoopPrd(s)
	para.time.initpos2	位置決め状態待ち時間	Q0	* MainLoopPrd(s)
入出力	stage.sub	サブステージ	---	
	stage_counter	ステージカウンター	Q0	* MainLoopPrd(s)
	drv.Vd_out	出力電圧値	Q31	電圧駆動時だけ有効
	drv.Id_com	d 軸電流指令値	Q31	
出力	stage.itr	割り込みステージ	---	
	drv.vector_cmd	ベクトル制御コマンド	---	(1.21.2 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	

#### 1.25.5.3 処理内容

ローターを初期位置に固定させます。

$\theta$  を「初期位置」に固定、 $\omega$  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。

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

$I_d$  を設定値まで徐々に増加させます。

### 1.25.6 強制転流状態関数 (C\_Stage\_Force)

#### 1.25.6.1 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

#### 1.25.6.2 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	
入出力	para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
	stage.sub	サブステージ	---	
出力	drv.omega_com	駆動速度指令値	Q31	
	drv.vector_cmd	ベクトル制御コマンド	---	(1.21.2 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

#### 1.25.6.3 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。

Id を「始動 Id 電流値」に、Iq を 0 に固定しながら、駆動速度指令値  $\omega_{com}$  を徐々に増加させます。 $\theta$  は  $\omega_{com}$  から求めます ( $\theta = \omega_{com}t$ )。この処理を  $\omega$  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  $\omega$  になります。

### 1.25.7 強制定常切換状態関数 (C\_Stage\_Change\_up)

#### 1.25.7.1 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

#### 1.25.7.2 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)
入出力	stage.sub	サブステージ	---	
	stage_counter	ステージカウンター	Q0	* MainLoopPrd(s)
	drv.omega_com	駆動速度指令値	Q31	
出力	stage.itr	割り込みステージ	---	
	drv.vector_cmd	ベクトル制御コマンド	---	(1.21.2 参照)
	drv.Id_com	d 軸電流指令値	Q31	
	drv.Iq_com	q 軸電流指令値	Q31	

#### 1.25.7.3 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。

$\omega$ 、 $\theta$  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。

### 1.25.8 定常状態関数 (C\_Stage\_Steady\_A)

#### 1.25.8.1 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

#### 1.25.8.2 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	
	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
入出力	stage.sub	サブステージ	---	
	drv.omega_com	駆動速度指令値	Q31	
出力	drv.vector_cmd	ベクトル制御コマンド	Q0	(1.21.2 参照)
	stage.itr	割り込みステージ	---	
	drv.Id_com	d 軸電流指令値	Q31	

#### 1.25.8.3 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。

### 1.25.9 保護状態関数 (C\_Stage\_Emergency)

#### 1.25.9.1 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

#### 1.25.9.2 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	---	(1.21.2 参照)
	stage.itr	割り込みステージ	---	

#### 1.25.9.3 処理内容

過電流が発生したとき、この状態へ遷移します。

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

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

ローターは惰性で回転します。

過電流状態復帰処理を実施するまでこのステージを維持します。

## 1.26 モーター駆動関数

### 1.26.1 用語説明

#### 1.26.1.1 3相変調の場合

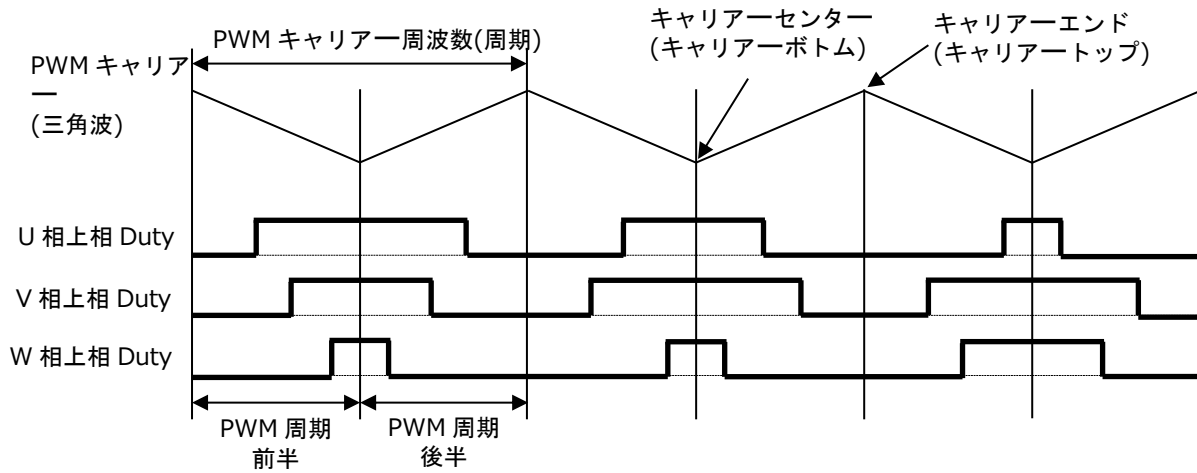


図 10.5 3相変調の場合

#### 1.26.1.2 2相変調の場合

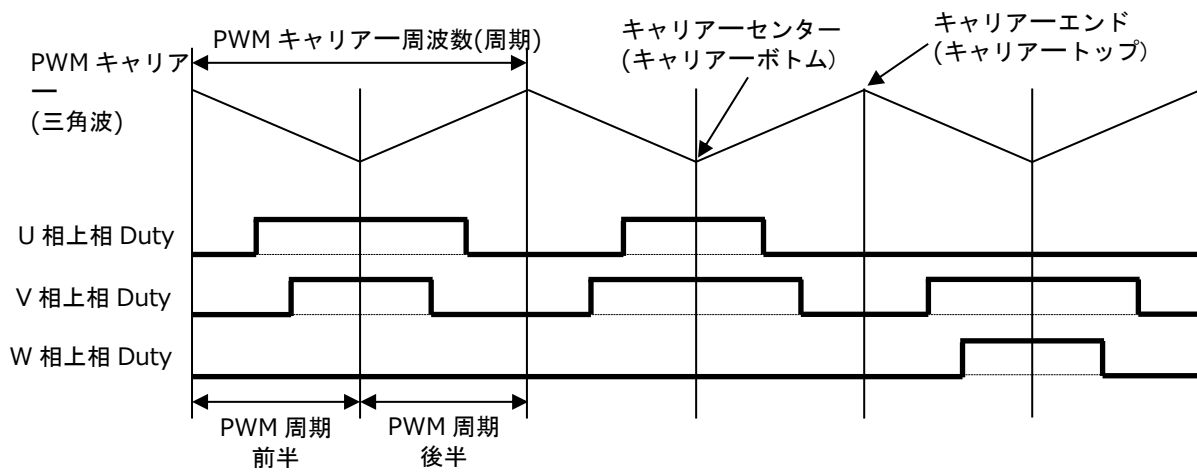


図 10.6 2相変調の場合

### 1.26.2 モーター電流、電源電圧取得 (D\_GetMotorCurrentPowerVolt)

#### 1.26.2.1 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

#### 1.26.2.2 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.TrigPosMd	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	---	
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		

### 1.26.2.3 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)。
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	-Ib-Ic	Iao_ave - Ia_adc	Iao_ave- Ia_adc	Iao_ave- Ia_adc	Iao_ave- Ia_adc	-Ib-Ic
V 相電流 drv.Ib	Ibo_ave - Ib_adc	-Ia-Ic	-Ia-Ic	Ibo_ave - Ib_adc	Ibo_ave- Ib_adc	Ibo_ave- Ib_adc
W 相電流 drv.Ic	Ico_ave - Ic_adc	Ico_ave - Ic_adc	Ico_ave- Ic_adc	-Ia-Ib	-Ia-Ib	Ico_ave - Ic_adc

<電流検出モード：1シャント前半、1シャント後半のとき>

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

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

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

モーター停止時のAD変換結果をゼロ電流として、下記変数に格納する。

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

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

	セクター					
	0	1	2	3	4	5
U相電流 drv.Ia	$-(Idco\_ave - Idc2\_adc)$	$-Ib - Ic$	$Idco\_ave - Idc1\_adc$	$Idco\_ave - Idc1\_adc$	$-Ib - Ic$	$-(Idco\_ave - Idc2\_adc)$
V相電流 drv.Ib	$-Ia - Ic$	$-(Idco\_ave - Idc2\_adc)$	$-(Idco\_ave - Idc2\_adc)$	$-Ia - Ic$	$Idco\_ave - Idc1\_adc$	$Idco\_ave - Idc1\_adc$
W相電流 drv.Ic	$Idco\_ave - Idc1\_adc$	$Idco\_ave - Idc1\_adc$	$-Ia - Ib$	$-(Idco\_ave - Idc2\_adc)$	$-(Idco\_ave - Idc2\_adc)$	$-Ia - Ib$

### 1.26.3 入力座標変換 (D\_InputTransformation)

#### 1.26.3.1 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

#### 1.26.3.2 変数

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

#### 1.26.3.3 処理内容

3相の電流(Ia, Ib, Ic)から、dq軸電流に変換します。

- 1.26.4 クラーク変換により、3相の電流(Ia, Ib, Ic)から2相の電流( $I_\alpha, I_\beta$ )に変換します。
- 1.26.5 パーク変換により、静止座標である2相の電流( $I_\alpha, I_\beta$ )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 $\omega$ と演算周期から予測した $\theta_{est}$ 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は、変換を行わず、Id, Iqは更新しません。

## 1.26.4 クラーク変換 (E\_Clarke)

## 1.26.4.1 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	$\alpha$ 軸電流格納アドレス
_ibeta	$\beta$ 軸電流格納アドレス

戻り値：

なし

## 1.26.4.2 処理内容

3 相の電流( $I_u$ ,  $I_v$ ,  $I_w$ )を 2 相( $I_\alpha$ ,  $I_\beta$ )に変換します。

下記演算式により  $I_\alpha$ 、 $I_\beta$  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  $\alpha$   $\beta$  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。

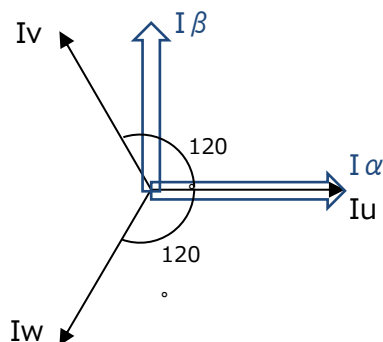


図 10.7 クラーク変換

### 1.26.5 パーク変換 (E\_Park)

#### 1.26.5.1 構文

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

引数 :

_ialpha	$\alpha$ 軸電流 $I_\alpha$
_ibeta	$\beta$ 軸電流 $I_\beta$
_theta	ローター位置 $\theta$ : $0 \leq \text{位置} < 360^\circ$ (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

#### 1.26.5.2 処理内容

$\alpha$   $\beta$  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。

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

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

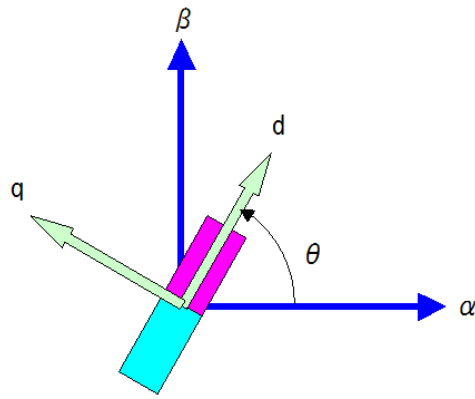


図 10.8 パーク変換

### 1.26.6 位置推定関数 (D\_Detect\_Rotor\_Position)

#### 1.26.6.1 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

#### 1.26.6.2 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	



### 1.26.7 エンコーダー制御関数 (H\_Encoder)

#### 1.26.7.1 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

#### 1.26.7.2 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.ctrlprd	エンコーダー制御周期	Q16	
	para.enc.deg_adjust	電気角調整	Q0	
	para.enc.pls2omega	パルス数から速度への演算係数	Q15	
	para.enc.pls2theta	パルス数から角度への演算係数	Q32	
	para.enc.plsnum	エンコーダーパルス数	Q0	
入出力	drv.EnCnt	エンコーダーカウンター値	Q0	
	drv.EnCnt_dev	エンコーダーカウンター偏差	Q0	
	drv.EnCnt1	エンコーダーカウンター前回値	Q0	
	drv.omega_enc	エンコーダー速度	Q15	
	drv.omega_enc_ave	エンコーダー速度平均値	Q31	
	drv.omega_enc_raw	エンコーダー速度	Q15	
	drv.theta_enc	エンコーダー角度	Q0	
出力	drv.omega	速度	Q31	
	drv.theta	ローター位置	Q0	

### 1.26.7.3 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） $\theta$ と、速度 $\omega$ の算出処理を行います。

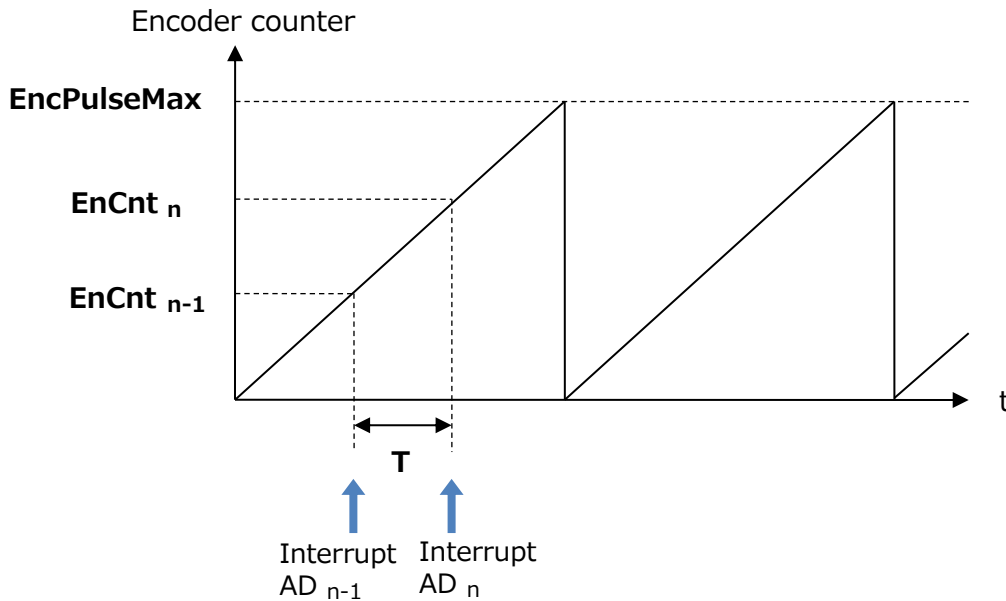


図 10.10 エンコーダーパルスからの位置および速度演算

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  $\theta$  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  $\omega$  を演算します。

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

EnCnt <sub>n</sub>	現在エンコーダーカウンタ値
EnCnt <sub>n-1</sub>	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]×逡倍数(4)
Pole	極数
$\theta$	角度[deg]
$\omega$	速度[Hz]
T	速度演算周期[s]

## 1.26.8 ホールセンサー+エンコーダー制御関数 (H\_HallCon H\_Hall\_Encoder)

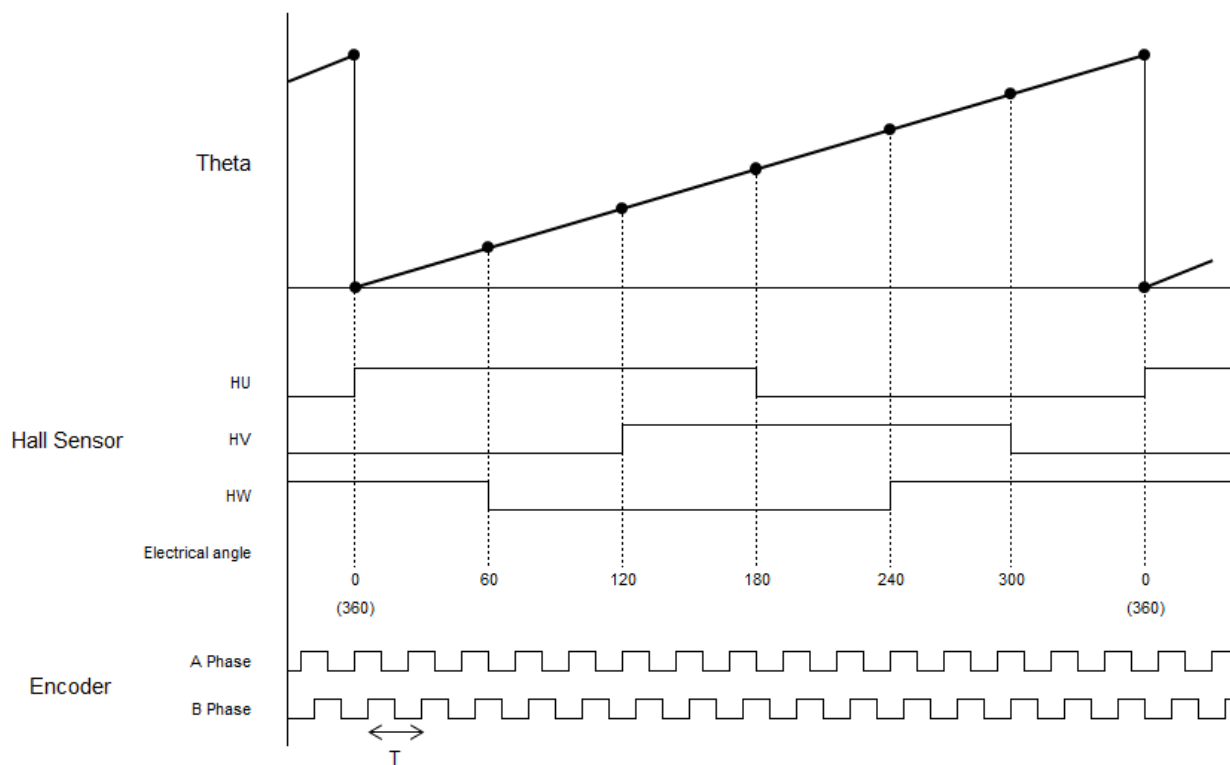


図 10.11 エンコーダーZ 信号検出前の位置検出制御

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。

### 1.26.9 速度制御関数 (D\_Control\_Speed)

#### 1.26.9.1 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

#### 1.26.9.2 変数

方向	名前	意味	Qフォーマット	備考
入力	drv.vector_cmd.F_vcomm_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_I	q 軸電流積分値	Q31	
	drv.omega_dev	速度偏差	Q15	
出力	drv.Id_ref	d 軸電流基準値	Q15	
	drv.Iq_ref	q 軸電流基準値	Q15	

#### 1.26.9.3 処理内容

制御量が出力周波数 $\omega$ 、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。

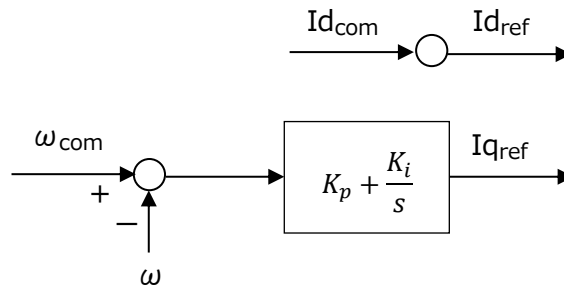


図 10.12 速度制御ブロック図

## 1.26.10 電流制御関数 (D\_Control\_Current)

## 1.26.10.1 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

## 1.26.10.2 変数

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

## 1.26.10.3 処理内容

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

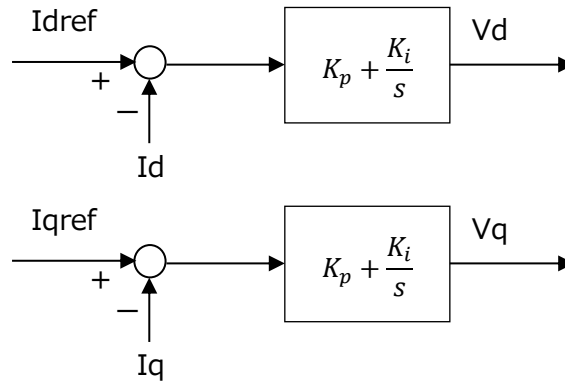


図 10.13 電流制御ブロック図

### 1.26.11 出力座標変換 (D\_OutputTransformation)

#### 1.26.11.1 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

#### 1.26.11.2 変数

方向	名前	意味	Q フォーマット	備考
入力	drv.vector_cmd.F_vcomm_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	---	
出力	drv.Valpha	$\alpha$ 軸電圧	Q31	
	drv.Vbeta	$\beta$ 軸電圧	Q31	
	drv.DutyU	U 相 PWM Duty 比	Q15 (符号無し)	0.0 to 1.0
	drv.DutyV	V 相 PWM Duty 比	Q15 (符号無し)	0.0 to 1.0
	drv.DutyW	W 相 PWM Duty 比	Q15 (符号無し)	0.0 to 1.0
	drv.Sector1	前回セクター	---	

## 1.26.11.3 処理内容

dq 軸電圧( $V_d, V_q$ )から 3 相の PWM Duty 比(DutyU, DutyV, DutyW)に変換します。

1. 1.26.12 逆パーク変換により、dq 軸電圧( $V_d, V_q$ )から  $\alpha \beta$  軸電圧( $V_\alpha, V_\beta$ )に変換します。
2.  $\alpha \beta$  軸電圧( $V_\alpha, V_\beta$ )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. 1.26.15 空間ベクトル変調あるいは 1.26.14 逆クラーク変換により、 $\alpha \beta$  軸電圧( $V_\alpha, V_\beta$ )から 3 相 Duty 比(DutyV, DutyW, DutyW)に変換します。

## 1.26.12 逆パーク変換 (E\_InvPark)

### 1.26.12.1 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 $\theta$ : $0 \leq \text{位置} < 360^\circ$ (0 to 0xffff)
_alpha	$\alpha$ 軸格納アドレス
_beta	$\beta$ 軸格納アドレス

戻り値 :

なし

### 1.26.12.2 処理内容

回転座標の dq 軸から静止座標の  $\alpha$   $\beta$  軸に変換します。

下記演算式により  $V_\alpha$ 、 $V_\beta$  を演算します。

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

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

## 1.26.13 セクター演算 (D\_CalSector)

### 1.26.13.1 構文

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

引数 :

  \_valpha             $\alpha$  軸電圧

  \_vbeta             $\beta$  軸電圧

戻り値 :

  セクター

### 1.26.13.2 処理内容

$\alpha$   $\beta$  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 10.4 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

**1.26.14 逆クラーク変換 (E\_InvClarke)**

本ソフトでは未実装です

**1.26.14.1 構文**

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

引数 :

_alpha	$\alpha$ 軸 $V_\alpha$
_vbeta	$\beta$ 軸 $V_\beta$
_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

**1.26.14.2 処理内容**

2 相( $V_\alpha$ ,  $V_\beta$ )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。

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

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

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

## 1.26.15 空間ベクトル変調 (D\_SVM)

### 1.26.15.1 構文

```
void D_SVM(  q31_t _alpha,  
             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 :

_alpha	$\alpha$ 軸 $V_\alpha$
_vbeta	$\beta$ 軸 $V_\beta$
_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

### 1.26.15.2 処理内容

2相の $\alpha$   $\beta$ 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)は図のように 60° ごとに表されます。

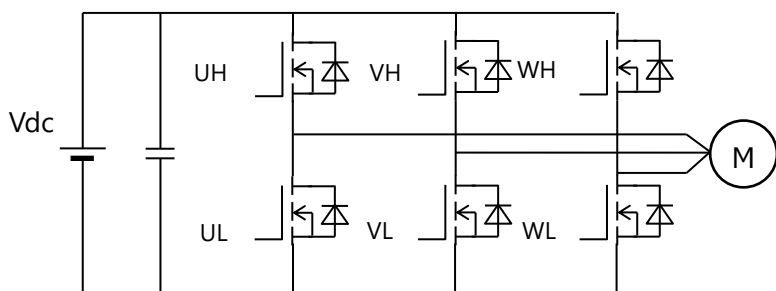
このうち隣り合う 2 つのベクトルを組み合わすことにより、任意の出力電圧ベクトル V を得ることができます。

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

表 10.5 インバータースイッチング状態

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

0: 上相 OFF、下相 ON  
1: 上相 ON、下相 OFF



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

図 10.14 インバーター駆動回路

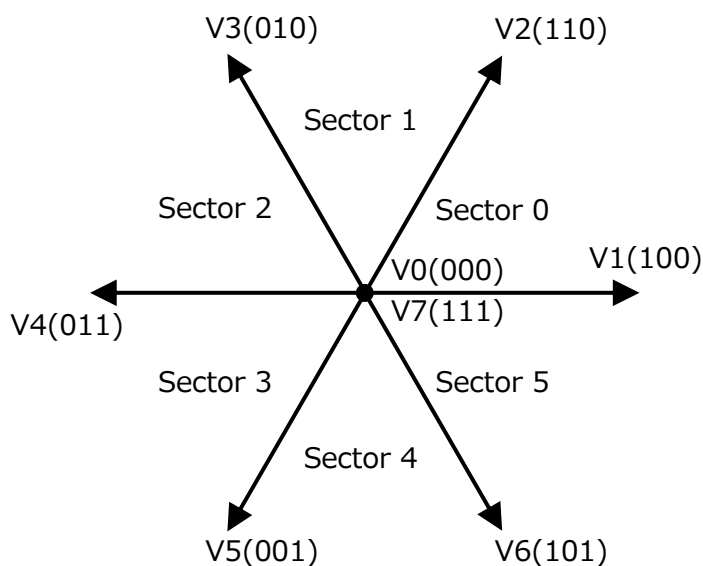


図 10.15 空間ベクトル

・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、 $V_\alpha$ 、 $V_\beta$  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。

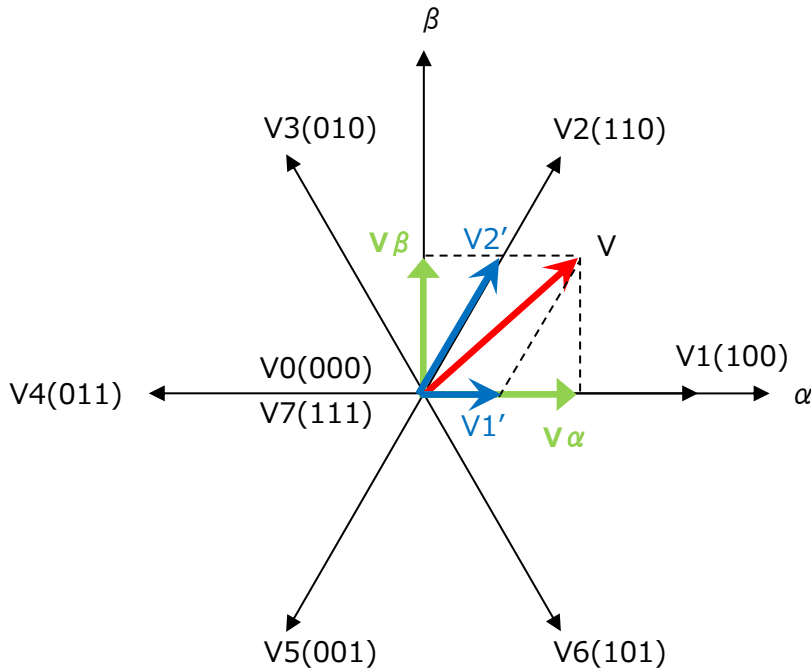


図 10.16 セクター 0 時の電圧ベクトル

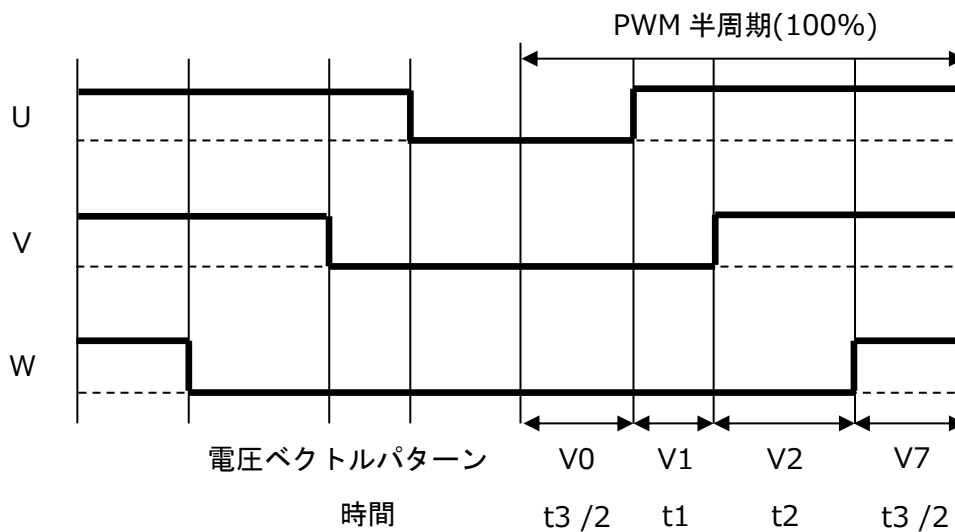


図 10.17 3 相変調時の PWM 波形

図 10.16 セクター0 時の電圧ベクトルより、 $V_{\alpha}, V_{\beta}$  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 10.6 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 10.6 セクターごとの Duty ON 時間の演算式

セクタ —	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 10.7 セクターと各相の Duty の関係

セクタ —	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

## 1.26.16 トリガータイミング演算 (D\_CalTrgTiming)

## 1.26.16.1 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

## 1.26.16.2 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.TrigPosMd	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)
	para.TrigComp	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

## 1.26.16.3 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シャント、1シャ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%(キャリアーエンド)。トリガータイミング補正值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%

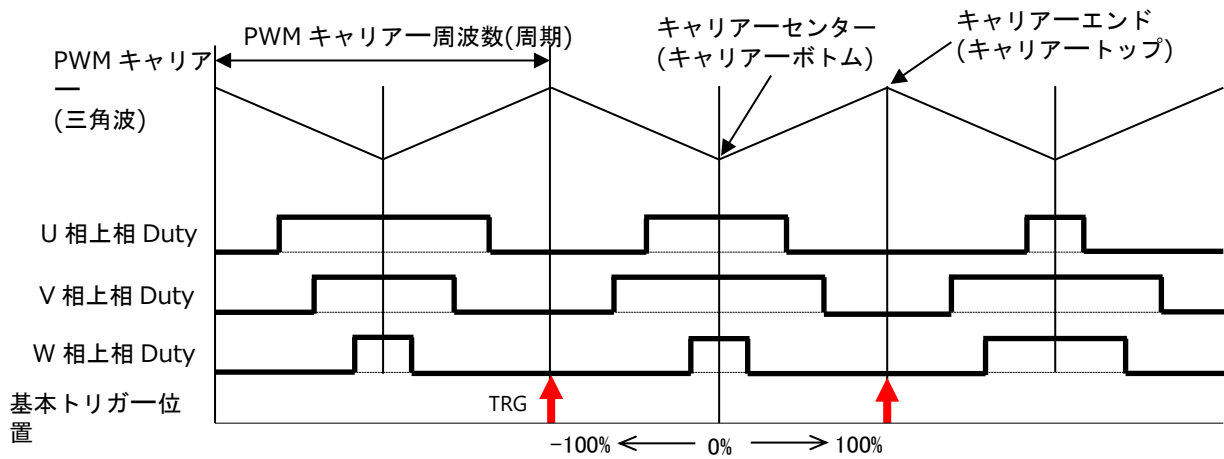


図 10.18 3 シャント時のトリガー位置

● 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 Vdc のトリガー位置は、キャリアエンドの位置に固定されます。(初期設定時に設定し、この関数では処理しない)

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

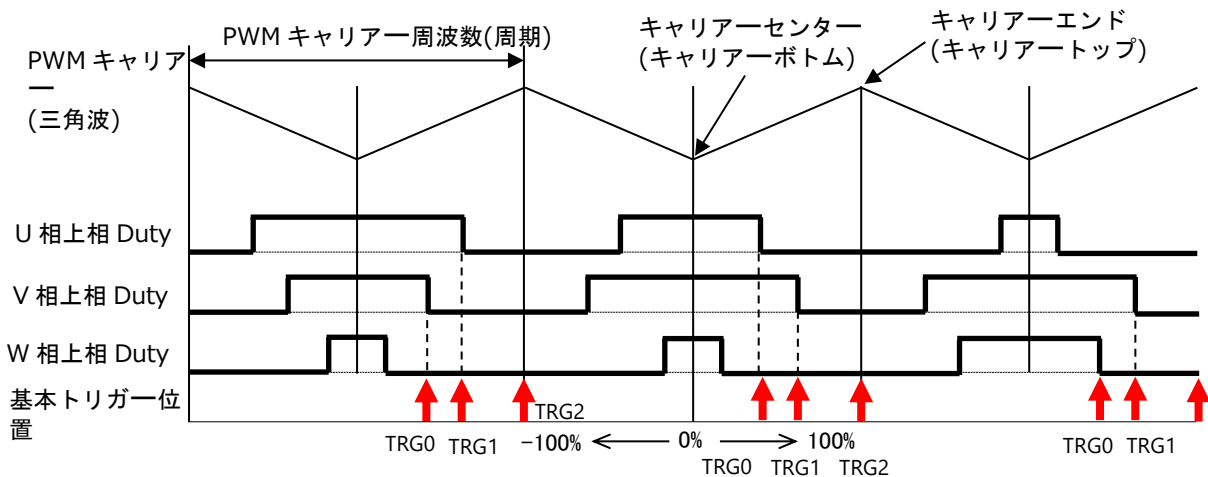


図 10.19 1 シャント、PWM 周期後半、3 相変調時のトリガー位置

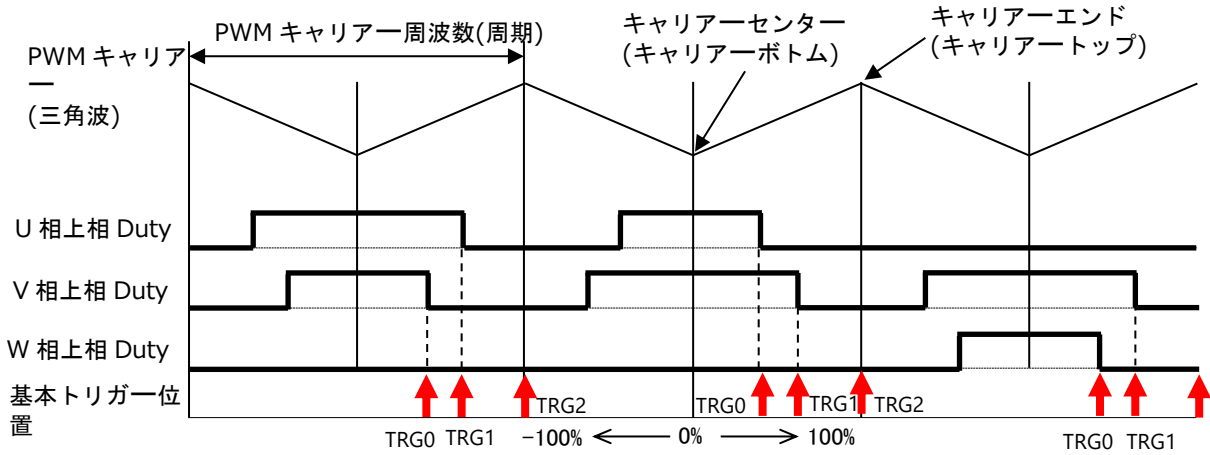


図 10.20 1 ショット、PWM 周期後半、2 相変調時のトリガー位置

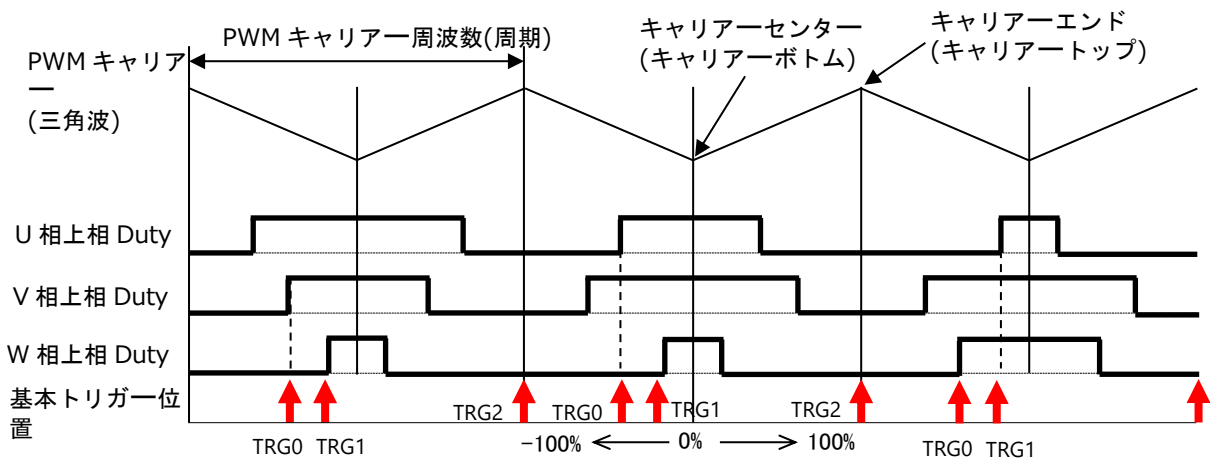


図 10.21 1 ショット、PWM 周期前半、3 相変調時のトリガー位置

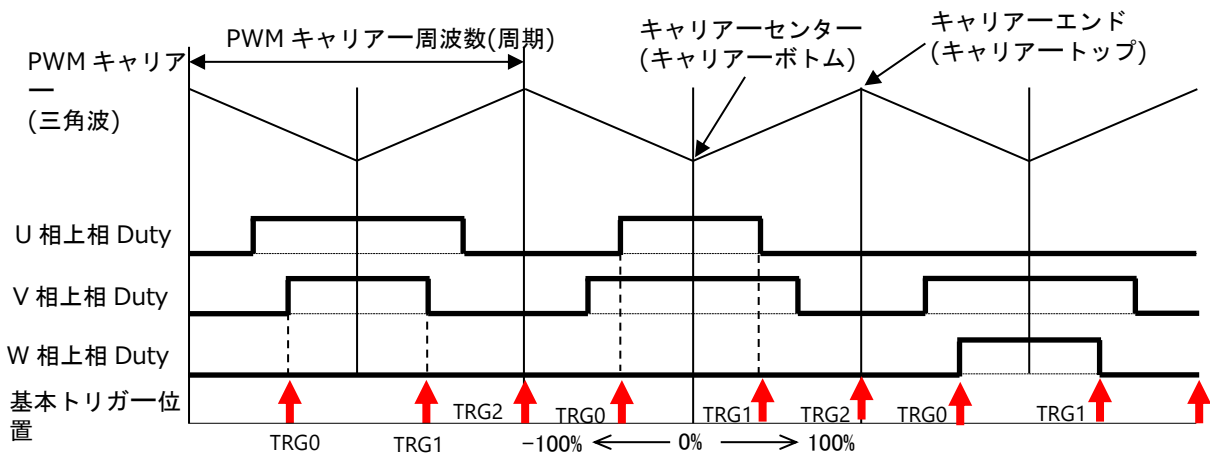


図 10.22 1 ショット、PWM 周期前半、2 相変調時のトリガー位置

### 1.26.17 PWM レジスタ設定 (PMD\_RegDataSet)

#### 1.26.17.1 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 adtrg1)
```

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 トリガータイミング 0 位置
adtrg1	AD トリガータイミング 1 位置

戻り値 :

なし

#### 1.26.17.2 処理内容

3相 PWM Duty とトリガータイミングの値をレジスタに設定します。

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。

$$\text{TRGCMP0} = (\text{adtrg0} + 32768) \div 2$$

$$\text{TRGCMP1} = (\text{adtrg1} + 32768) \div 2$$

$$\text{CMPU} = \text{duty\_u}$$

$$\text{CMPV} = \text{duty\_v}$$

$$\text{CMPW} = \text{duty\_w}$$

## 1.26.18 電流誤検知検出関数 (D\_Check\_DetectCurrentError)

## 1.26.18.1 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

## 1.26.18.2 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

## 1.26.18.3 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。

この関数で電流を検出できない 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。

## 1.26.19 インバーター出力電圧の算出関数 (Cal\_Vdq)

### 1.26.19.1 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)

### 1.26.19.2 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

### 1.26.19.3 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. 定数定義説明

### 1.27 モータードライバー設定用パラメーター共通 (D\_Para.h)

#### 1.27.1 モーター制御チャンネル選択

`__CONTROL_MOTOR_CH1` CH1 を制御するとき定義してください。  
`__CONTROL_MOTOR_CH0` CH0 を制御するとき定義してください。

#### 1.27.2 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。

#### 1.27.3 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。

#### 1.27.4 指令速度単位選択

`__TGTSPEED_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

#### 1.27.5 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。
cIXO_AVE	ゼロ電流値用フィルター係数
	--
	フィルター係数を設定してください。 大きな値を設定すると安定しますが、反応が遅れます。
cVDQ_AVE	電源電圧 Vdc,出力電圧 Vdq 用フィルター係数
	--
	フィルター係数を設定してください。 大きな値を設定すると安定しますが、反応が遅れます。
cOMEGAENC_AVE	エンコーダーパルス数からの速度検出値用フィルター係数
	--
	速度検出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。

### 1.28 モータードライバー設定用パラメーターモーターチャンネル別 (D\_Para\_Chx.h) x=0,1

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

#### 1.28.1 制御選択

##### 1.28.1.1 センサーレス/センサー選択

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

#### 1.28.2 モーターチャンネル別パラメーター一覧

パラメーター名	説明
cPOLH	上相ドライバー論理設定
	0: Low active/ 1: High active
	基板設計に応じて値を変更してください。 上相ドライバーの論理の設定を行ってください。
cPOLL	下相ドライバー論理設定
	0: Low active/ 1: High active
	基板設計に応じて値を変更してください。 下相ドライバーの論理の設定を行ってください。
cV_MAX	最大電圧値の設定
	単位[V]
	基板設計に応じて値を変更してください。 (AD 変換が 1LSB 変化する電源電圧の変化量)[V] × 2 <sup>12</sup> 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] × 2 <sup>11</sup> 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 16.67ns@120MHz PWM キャリアー周期を設定してください。
cDEADTIME	デッドタイム値の設定 単位[us] 分解能 66.67ns@120MHz デッドタイムの値を設定してください。
cREPTIME	ソフトウェア制御間引き回数の設定 単位[回] 1 to 15 ベクトル演算ソフトウェア処理を行うタイミングを間引きすることができます。 1 と設定すると、PWM1 周期ごとにベクトル演算のソフト処理を行います。 2 と設定すると、PWM2 周期に 1 回ベクトル演算のソフト処理を行います。
cSPEED_ACT	指令速度の設定 単位[Hz] or [RPS] or [RPM] 単位は <u>TGTSPD_UNIT</u> 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替え速度
	単位[Hz]
	強制転流から定常へ移行させる速度を設定してください。 位置推定ができる(誘起電圧が検出できる)速度を設定します。
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。 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]

	<p>3シャント駆動時、電流が取得できなくなる時の、中間相のPWMパルス幅時間を設定してください。</p> <p>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。</p>
cID_LIM	d軸電流リミット値
	単位[A]
cIQ_LIM	d軸電流のリミット値を設定してください。
	q軸電流リミット値
cINICIAL_POSITION	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1.28.3.2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1.28.3.2を参照してください。
cHZ_V2I	電圧制御から電流制御への切替速度
	単位[Hz]
	電圧制御から電流制御へ切り替える速度を設定してください。 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]
cENC_DEG_ADJUST	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。
	エンコーダー位置調整
cENC_NOISE_TIME	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。
	エンコーダー入力信号ノイズキャンセル時間設定 (A-ENC だけ有効)
__FIXED_VDC	単位[us] 分解能 fc (8.33ns@120MHz)
	エンコーダー入力信号のノイズキャンセル時間を設定してください。 この設定時間より短い信号は、キャンセルします。 この設定はA-ENC搭載マイコンのときだけ有効となります。
cVDC	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。
	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
	単位[ns]
	0～(cPWMPRD/2)の範囲で設定してください。

## 1.28.3 定数設定値と波形の関係

### 1.28.3.1 電流型起動時の定数値

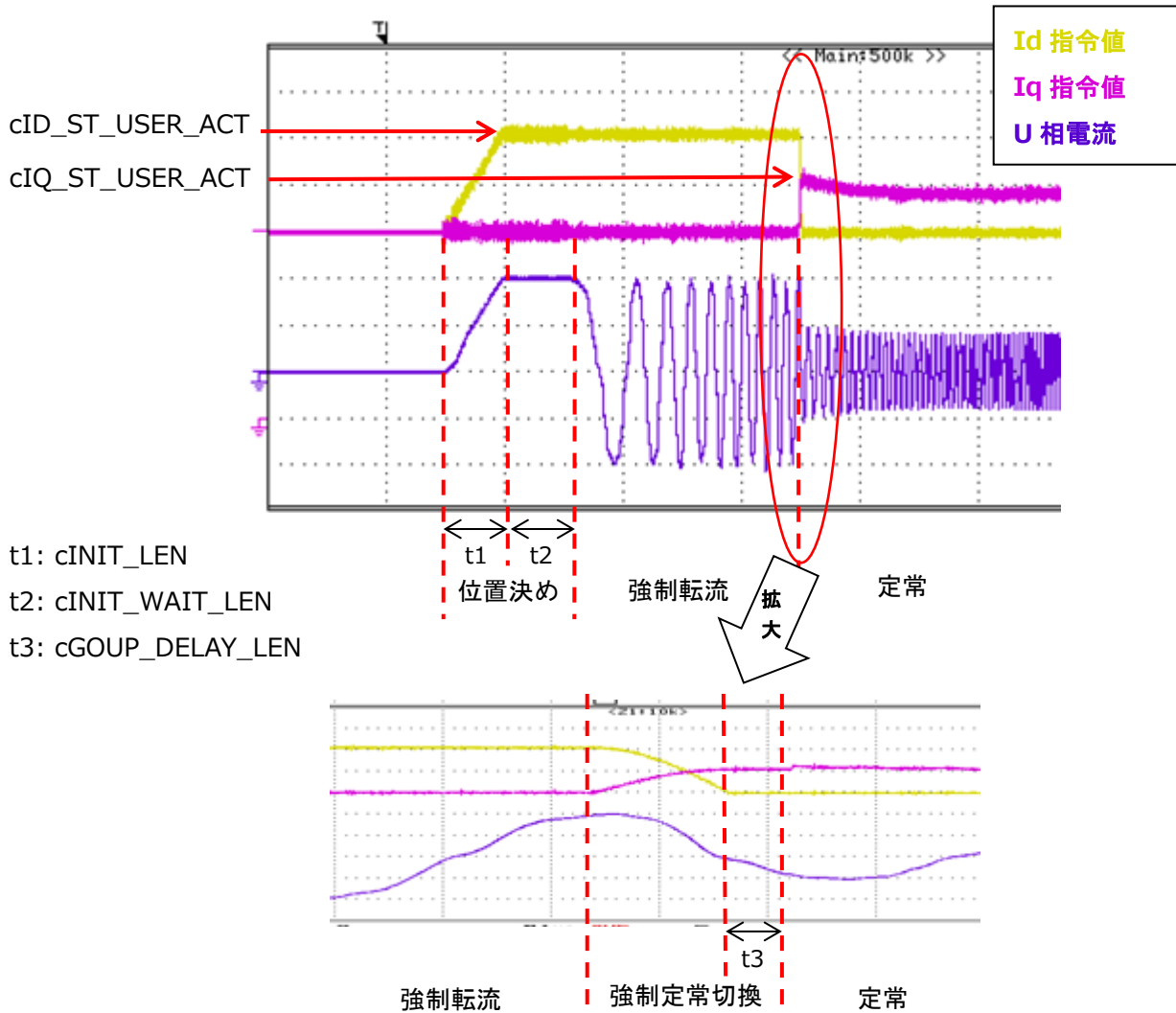


図 11.1 起動電流波形（電気角 0° に位置決め）

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。

### 1.28.3.2 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。

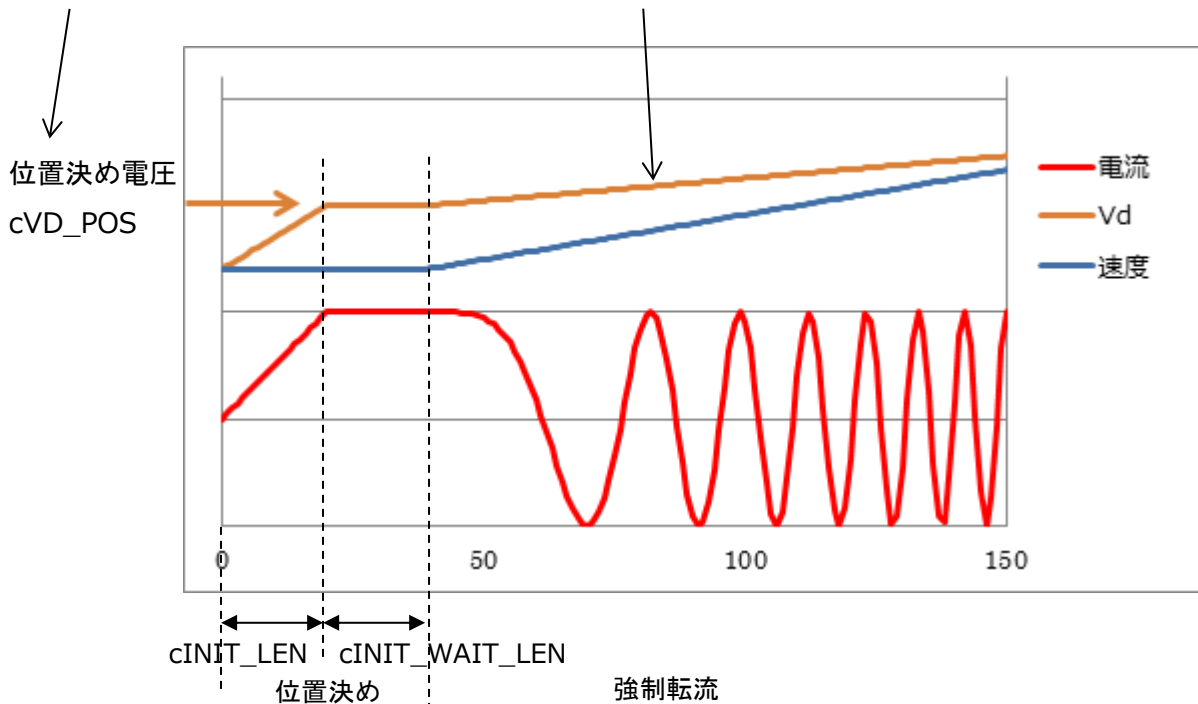


図 11.2 電圧駆動時のパラメーター調整方法

### 1.29 ユーザー制御関連定数

```

/*Initial value*/
#define cINIT_VALUE    (0)
#define cINIT_FF      (0xff)

/* Timer Setting */
#define c3000MS_TIMER      (3000)          /* [ms] (4kHz * 4) * 3000 */

/* Key settting */
#define cS_SW1              ((*((__IO uint32_t *)BITBAND_PERI(&TSB_PA->DATA,0))) /* SW1 */
#define cS_SW2              ((*((__IO uint32_t *)BITBAND_PERI(&TSB_PA->DATA,1))) /* SW2 */
#define cS_SW3              ((*((__IO uint32_t *)BITBAND_PERI(&TSB_PA->DATA,2))) /* SW3 */
#define cS_SW4              ((*((__IO uint32_t *)BITBAND_PERI(&TSB_PA->DATA,3))) /* SW4 */
#define cUSW1               ((*((__IO uint32_t *)BITBAND_PERI(&TSB_PE->DATA,5))) /* USW1 */
#define cUSW2               ((*((__IO uint32_t *)BITBAND_PERI(&TSB_PE->DATA,4))) /* 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_TEMP0      TSB_ADA      /* TEMP0 ad data ADCUnit */
#define cADUNIT_TEMP1      TSB_ADB      /* TEMP1 ad data ADCUnit */

#define cADTRG_USR          ADC_TRG_SINGLE /* AD Trigger Type */

#define cADCH_VR            ADC_AIN21    /* ADC Channel for VR */
#define cADREG_VR          ADC_REG4      /* Result register Number for VR */
#define cADCH_TEMP0        ADC_AIN17    /* ADC Channel for TEMP0 */
#define cADREG_TEMP0       ADC_REG5      /* Result register Number for TEMP0 */
#define cADCH_TEMP1        ADC_AIN19    /* ADC Channel for TEMP1 */
#define cADREG_TEMP1       ADC_REG6      /* Result register Number for TEMP1 */

#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_MIN_CH1 cHZ_MIN_CH1 /* [Hz] Min Target speed of motor ch1 */
#define cSPEED_USER_MAX    (100)        /* [Hz] Max Target speed of motor ch0,1*/

/* 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_PE->DATA,6))) /* LED4 */
#define cLED_EMG_CH1 (*((__IO uint32_t *)BITBAND_PERI(&TSB_PE->DATA,7))) /* LED5 */
#define cLED_ON            (1)           /* LED ON level */
#define cLED_OFF           (0)           /* LED OFF level */

```

```

/* State control Setting */
#define cMOTOR_CH0          (0x00)          /* Control channel : CH0 */
#define cMOTOR_CH1          (0x01)          /* Control channel : CH1 */
#define cDISP_CH0           (0x00)          /* Display channel 0 */
#define cDISP_CH1           (0x01)          /* Display channel 1 */

/* UART Setting */
#define cUART_BRD_INIT      ((uint32_t)0x00A6002B)
#define cUART_CLK_INIT     ((uint32_t)0x00000000)
#define cUART_CR0_INIT     ((uint32_t)0x00000001)
#define cUART_CR1_INIT     ((uint32_t)0x00000040)
#define cUART_SEND_WAIT    (1000)
#define cUART_COUNT        (0)
#define c10MHz              (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 cINV_MIN_TEMP (-200)
#define cINV_MAX_TEMP (1000)
#define cSPD_UPDOWN_RESOLUTION (10)
```

## 12. 制御、データ更新のタイミング

### 1.30 ソフトウェアによるベクトル制御

#### 1.30.1 3 シャント制御

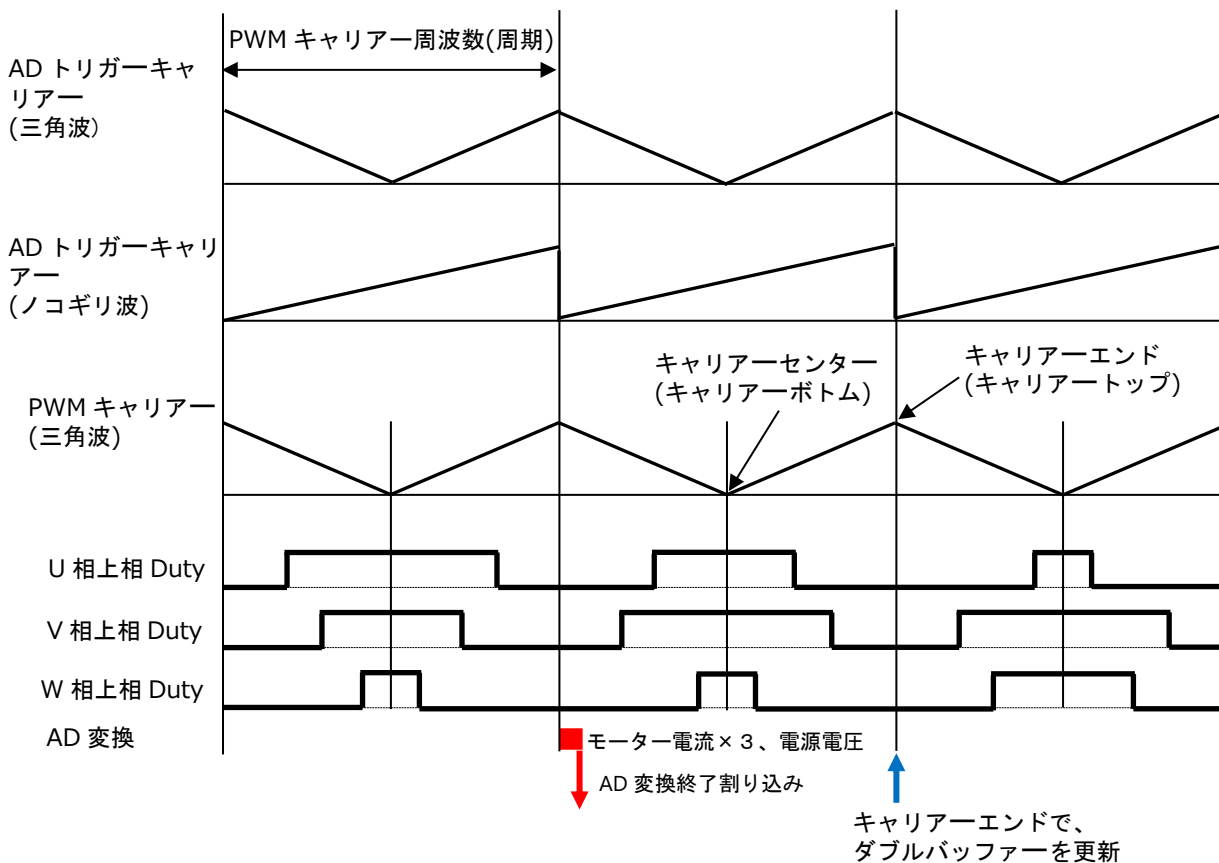


図 12.1 3 シャント制御

##### 1.30.1.1 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

##### 1.30.1.2 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

### 1.30.1.3 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

### 1.30.2 1シャント制御

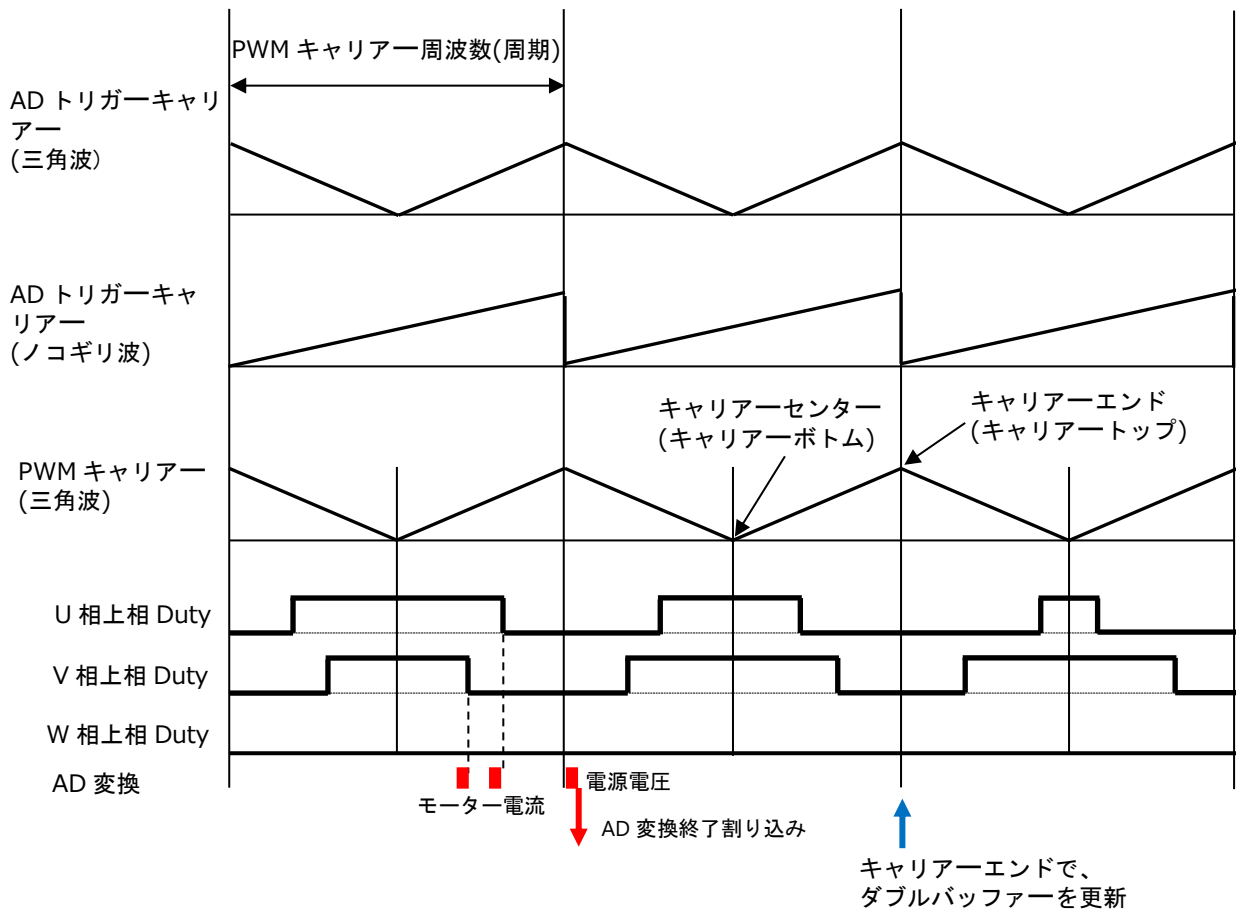


図 12.2 1 シャント制御

#### 1.30.2.1 キャリアー波形

種類	波形
PWM	3相とも三角波
AD トリガー	のこぎり波

#### 1.30.2.2 ダブルバッファー更新タイミング

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

## 1.30.2.3 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. ペリフェラルドライバー

## 1.31 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。

## 1.31.1 データ構造

## 1.31.1.1 ipdrv\_t

**Data Fields:**

TSB\_PMD\_TypeDef\* const **PMDx**: PMD アドレスを選択します。

TSB\_AD\_TypeDef\* const **ADx**: ADC アドレスを選択します。

## 1.32 モーター制御回路(PMD)

## 1.32.1 関数仕様

## 1.32.1.1 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 1.32.2.1PMD\_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

### 1.32.1.2 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 1.31.1.1 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

### 1.32.1.3 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 1.31.1.1 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

### 1.32.2 データ構造

#### 1.32.2.1 PMD\_InitTypeDef

**Data Fields:**

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

## 1.33 アナログデジタルコンバーター(ADC)

### 1.33.1 関数仕様

#### 1.33.1.1 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 1.33.2.1AD\_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

### 1.33.2 データ構造

#### 1.33.2.1 AD\_InitTypeDef

**Data Fields:**

uint8\_t **shunt:** シャントタイプ

**1:** 1 シャント

**3:** 3 シャント

uint8\_t **iuch:** U 相電流取り込み AD チャンネル番号(3 シャント用)

uint8\_t **ivch:** V 相電流取り込み AD チャンネル番号(3 シャント用)

uint8\_t **iwch:** W 相電流取り込み AD チャンネル番号(3 シャント用)

uint8\_t **idcch:** DC 電流取り込み AD チャンネル番号(1 シャント用)

uint8\_t **vdccch:** 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

**cPMD1:** PMD チャンネル 1

uint8\_t **pints:** PMD トリガー用割り込み選択

**cPINTS\_A:** INTADxPDA

**cPINTS\_B:** INTADxPDB

**cPINTS\_C:** INTADxPDC

**cPINTS\_D:** INTADxPDD

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

## 14. 付録

### 1.34 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。

### 1.35 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。

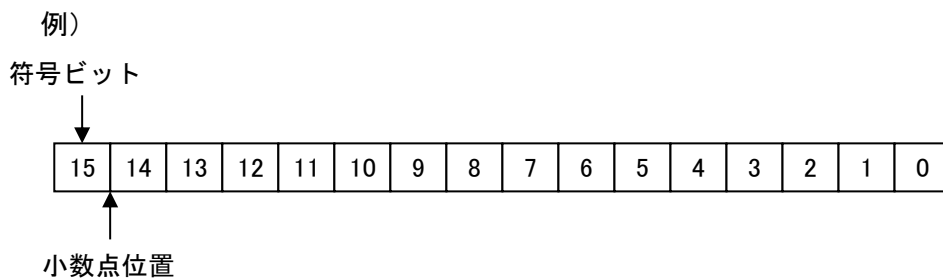


図 14.1 16 ビットデータの例

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。

## 1.36 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 14.1 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 14.2 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	0.999999999534338	-1
Q30	30	1.999999999068670	-2
Q29	29	3.999999998137350	-4
Q28	28	7.999999996274700	-8
Q27	27	15.999999992549400	-16
Q26	26	31.999999985098800	-32
Q25	25	63.999999970197600	-64
Q24	24	127.999999940395000	-128
Q23	23	255.999999880790000	-256
Q22	22	511.999999761581000	-512
Q21	21	1023.999999523160000	-1024
Q20	20	2047.999999046320000	-2048
Q19	19	4095.999998092650000	-4096
Q18	18	8191.999996185300000	-8192
Q17	17	16383.999992370600000	-16384
Q16	16	32767.999984741200000	-32768
Q15	15	65535.999969482400000	-65536
Q14	14	131071.999938965000000	-131072
Q13	13	262143.999877930000000	-262144
Q12	12	524287.999755859000000	-524288
Q11	11	1048575.999511720000000	-1048576
Q10	10	2097151.999023440000000	-2097152
Q9	9	4194303.998046870000000	-4194304
Q8	8	8388607.996093750000000	-8388608
Q7	7	16777215.992187500000000	-16777216
Q6	6	33554431.984375000000000	-33554432
Q5	5	67108863.968750000000000	-67108864
Q4	4	134217727.937500000000000	-134217728
Q3	3	268435455.875000000000000	-268435456
Q2	2	536870911.750000000000000	-536870912
Q1	1	1073741823.500000000000000	-1073741824
Q0	0	2147483647.000000000000000	-2147483648

### 1.37 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。

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

### 1.38 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
```

**15. 改訂履歴**

日付	Rev.	変更内容
2024/03/03	1.0	初版
2026/05/15	1.1	KES-PH に追加対応

## 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。

東芝デバイス&ストレージ株式会社

<https://toshiba.semicon-storage.com/jp/>