TMPM375
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

目次

図目次

表目次

1. 概要

1.1. はじめに

本モーター制御用サンプルソフトはイーエスピー企画製 SBK-M375 (TMPM375 ブラシレス DC モーターベクトル制御開発キット)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　TMPM375FSDMG
- ◆ クロック　40MHz
- ◆ 使用モーター　ツカサ電工(株) - TG611B
- ◆ 開発環境　IAR Embedded Workbench for ARM V9.70.1
  KEIL μVision MDK-ARM V5.43.1.0
  Segger Embedded Studio for ARM V8.24
- ◆ 直流リンク電圧　DC24V

- ◆ 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)
  -トグル SW(回転方向)、スライドボリューム(回転数)入力

- ◆ 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
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(スイッチ切り替え)
電流検出方式	3 シャント/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)
オペアンプ	マイコン内蔵/外部

注 1: 未評価

・注意事項
  ・EMG が発生した場合、VR 操作でモーター駆動を OFF にして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_CH1.h
      #define cSHUNT_TYPE_CH1(1)
      #define cBOOT_TYPE_CHx (cBoot_v)

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層、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層、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層の 3 階層で構成されます。

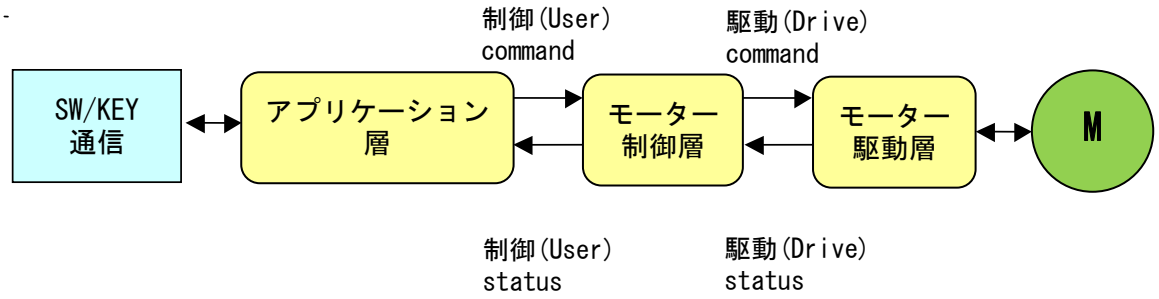


図 1.1　ベクトル制御ソフトウェアの構造

i. アプリケーション層は、スイッチ、キー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層に与えます。また制御ステータスをモーター制御層から取得し、必要な処理を行うとともに、LED などに表示します。

ii. モーター制御層はアプリケーション層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層から取得し、必要な処理を行うとともにアプリケーション層に転送します。

iii. モーター駆動層はモーター制御層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層に転送します。

例えば、モーター回転中にアプリケーション層から新たな制御目標速度が与えられたとき、モーター駆動層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層に与えます。

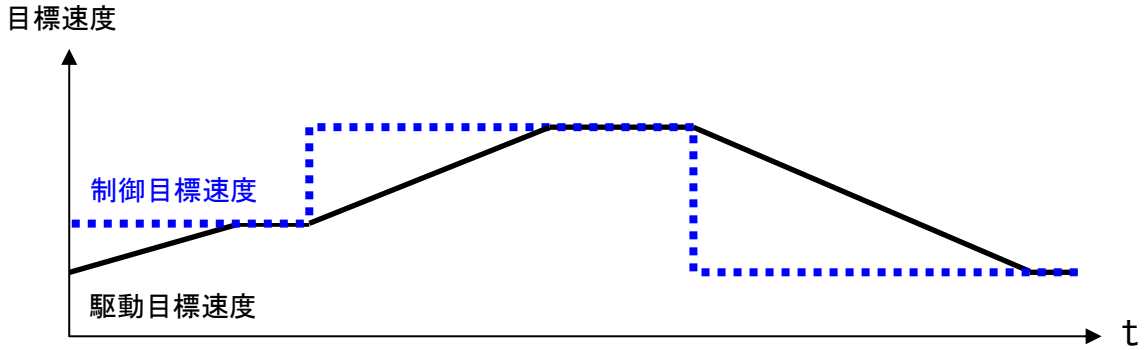


図 1.2　制御目標速度と駆動目標速度

2. システムブロック図

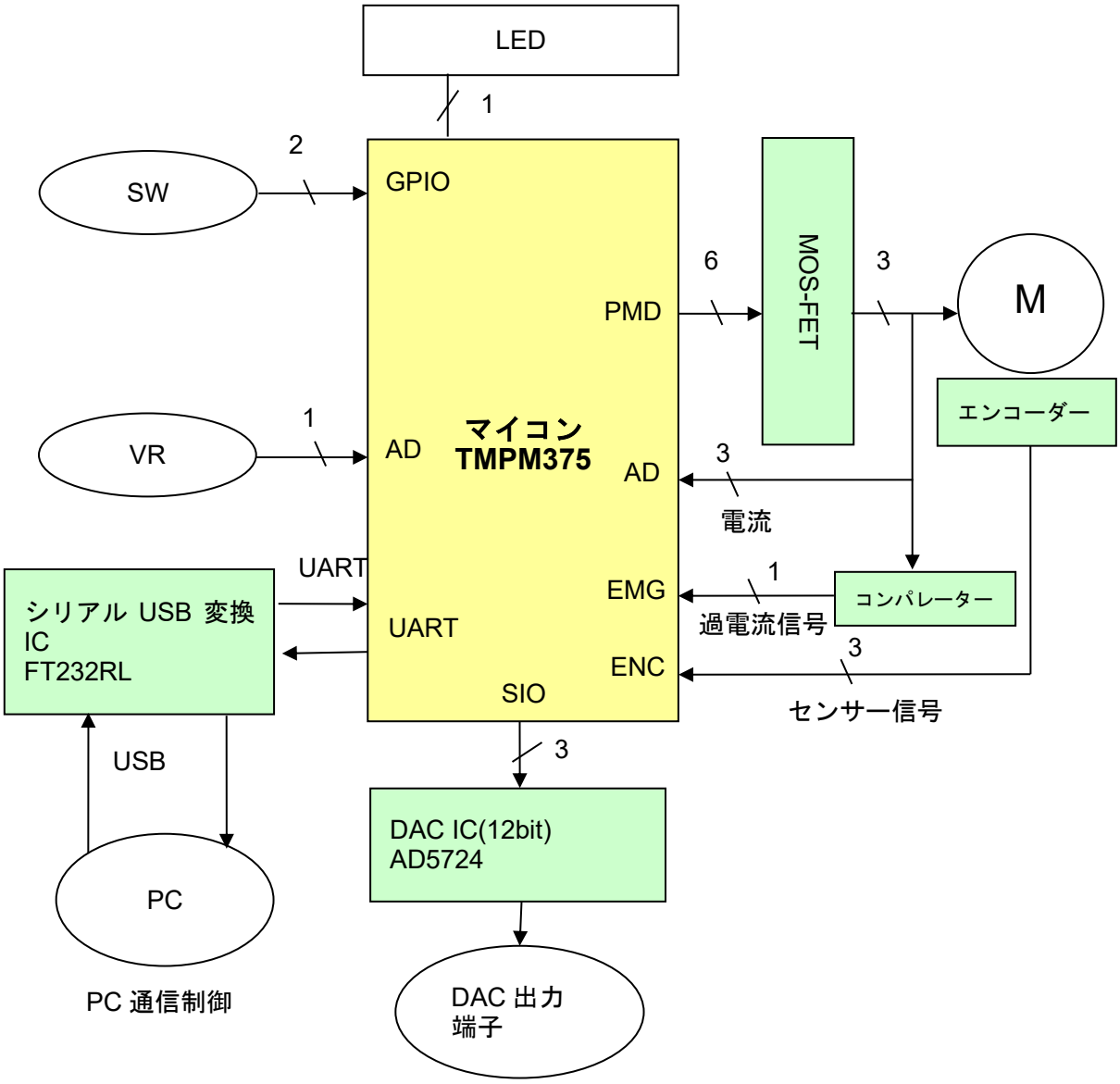


図 2.1　システムブロック図

3. ソースファイル構成

```
source
 │   B_User.c          ベクトル制御用ユーザー設定関連ソースファイル
 │   B_User.h          ベクトル制御用ユーザー設定関連ヘッダーファイル
 │   C_Control.c       ベクトル制御用コントロール関連ソースファイル
 │   C_Control.h       ベクトル制御用コントロール関連ヘッダーファイル
 │   dac_drv.c         DAC IC ドライバーソースファイル
 │   dac_drv.h         DAC IC ドライバーヘッダーファイル
 │   D_Driver.c        ベクトル制御用ドライバーソースファイル
 │   D_Driver.h        ベクトル制御用ドライバーヘッダーファイル
 │   D_Para.h          ベクトル制御用パラメーター(チャネル共通)ヘッダーファイル
 │   D_Para_ch1.h      ベクトル制御用パラメーター(チャネル 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
 │   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_v.c     マイコンハード設定ドライバーソースファイル
 │   mcuip_drv_v.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
 │   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
 │   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。
    └──M375
        ├──CMSIS                    CMSISコアが格納されています。
        │   │   system_TMPM375.c     マイコン初期設定用ソースファイル
        │   │   system_TMPM375.h     マイコン初期設定用ヘッダーファイル
        │   │   TMPM375.h            マイコンレジスター定義ヘッダーファイル
        │   └──startup
        │       ├──arm
        │       │   startup_TMPM375.s        スタートアップアセンブラーソース
        │       │                                (arm 用)
        │       │   TMPM375.scat             リンカーファイル (arm 用)
        │       └──iar
        │           startup_TMPM375.s        スタートアップアセンブラーソース
        │                                    (IAR 用)
        │           TMPM375.icf              リンカーファイル (IAR 用)
        │       └──segger
        │           TMPM375_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)
        │           Cortex_M_Startup.s        スタートアップアセンブラーソース
        │                                    (SEGGER 用)
        │           SEGGER_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。
            tmpm375_adc.c
            tmpm375_adc.h
            tmpm375_cg.c
```

tmpm375_cg.h
tmpm375_dnf.c
tmpm375_dnf.h
tmpm375_enc.c
tmpm375_enc.h
tmpm375_fc.c
tmpm375_fc.h
tmpm375_gpio.c
tmpm375_gpio.h
tmpm375_ofd.c
tmpm375_ofd.h
tmpm375_pmd.c
tmpm375_pmd.h
tmpm375_sbi.c
tmpm375_sbi.h
tmpm375_tmrb.c
tmpm375_tmrb.h
tmpm375_trmosc.c
tmpm375_trmosc.h
tmpm375_uart.c
tmpm375_uart.h
tmpm375_vltd.c
tmpm375_vltd.h
tmpm375_wdt.c
tmpm375_wdt.h
tx03_common.h

4. 評価環境について

4.1. 開発ツール

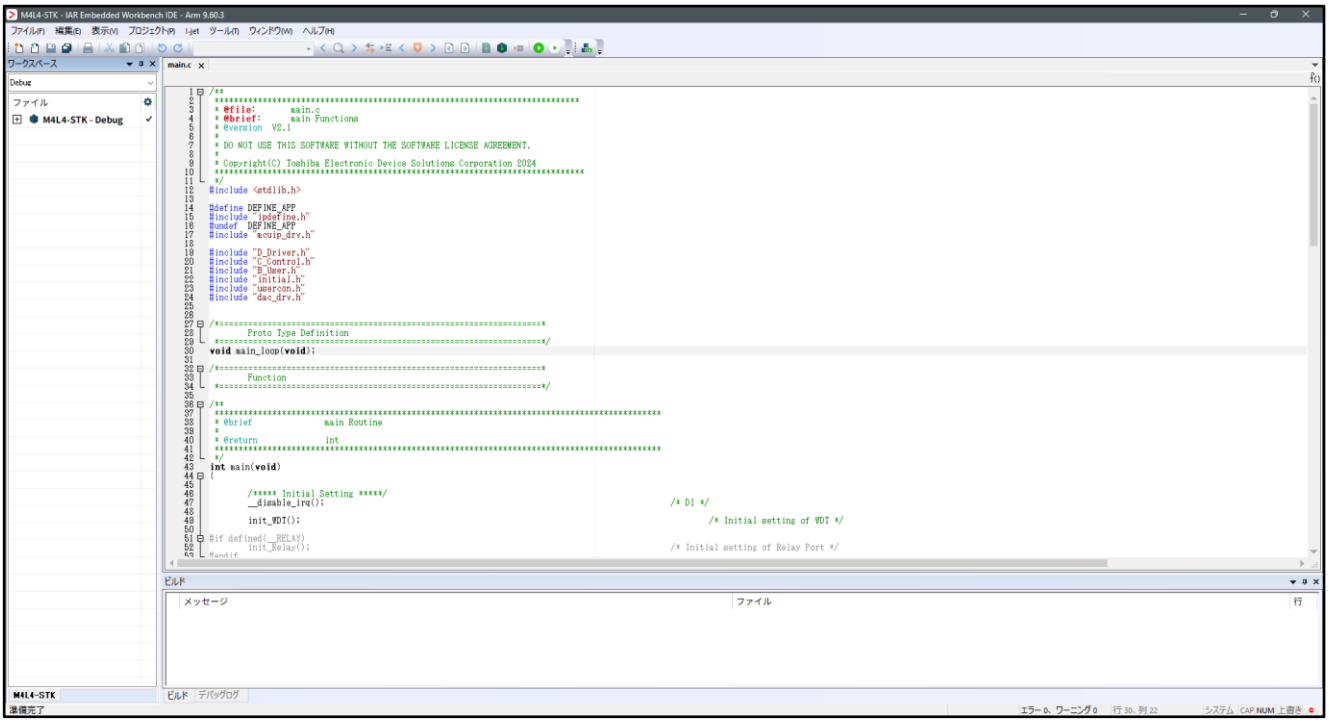
本ソフトは以下の開発ツールに対応しています。

IAR Embeded Workbench for ARM	9.70.1
Keil µVision MDK	5.43.1.0
Segger Embedded Studio	8.24

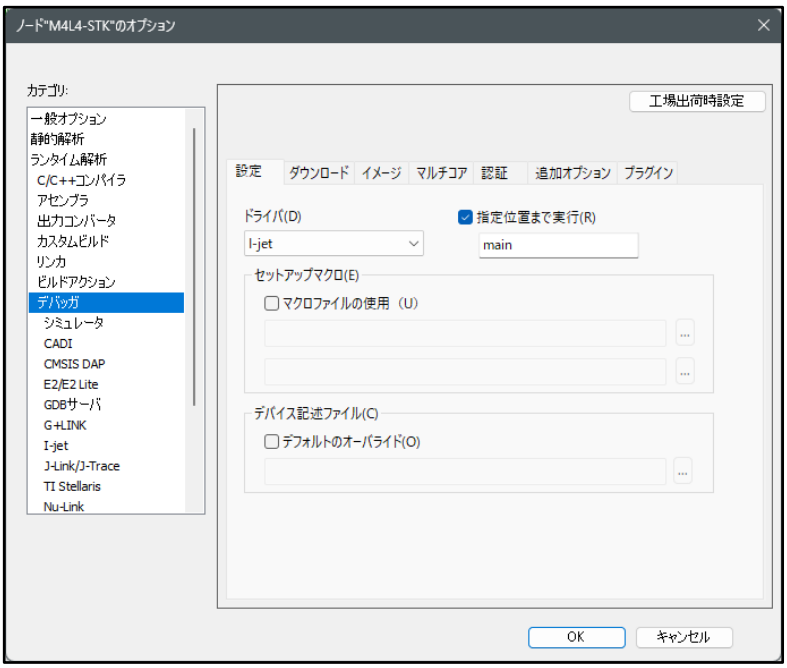
4.2. プロジェクトの立ち上げ方

IAR Embeded Workbench の場合を説明します。

1. iar¥M375-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 M375-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] > [デバッガー]の<設定>タブ

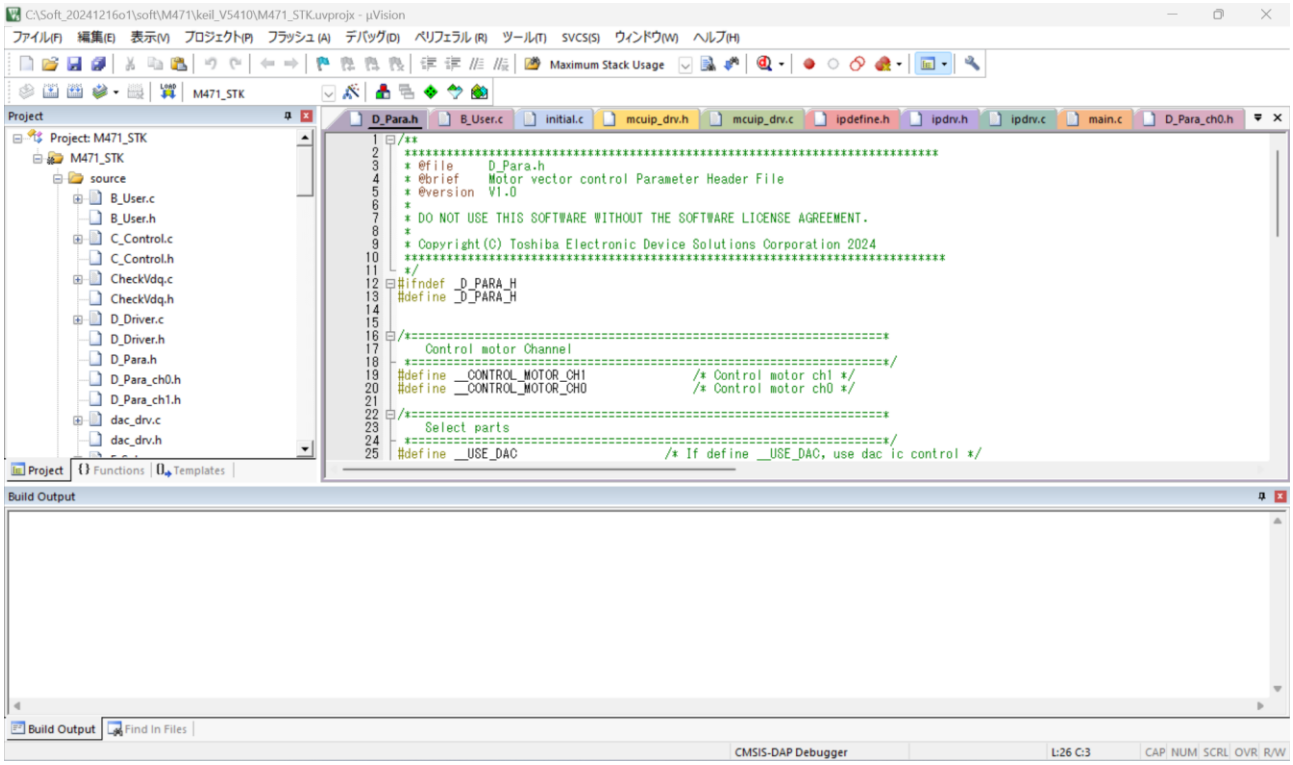


使用するツールを選択してください。

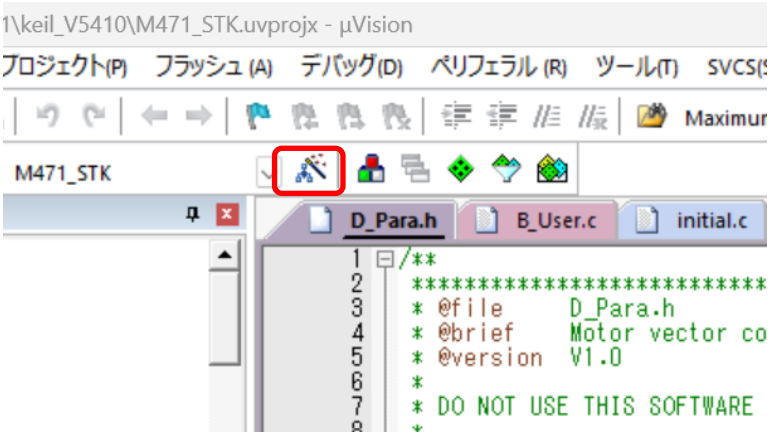
4. デバッグを開始するときは、ツールを接続し[プロジェクト] > [ダウンロードしてデバッグ]を選択するか、下記ボタンを押してください。

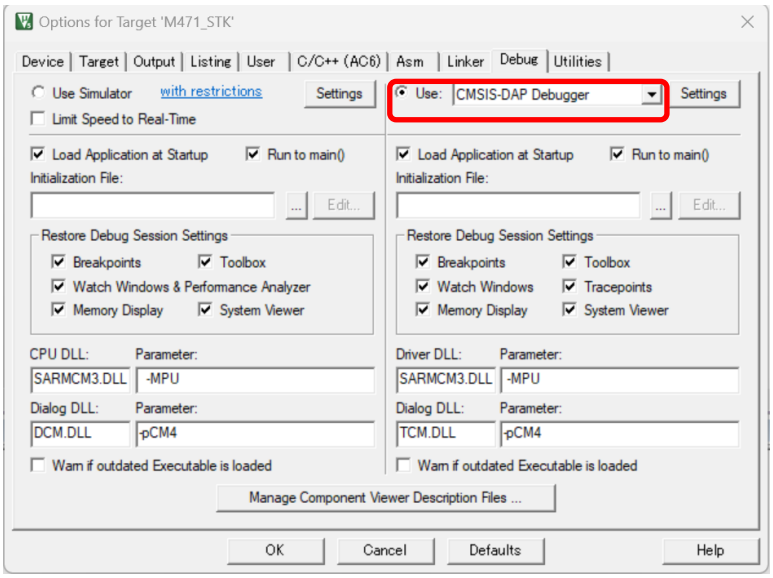
Keil µVision MDK の場合を説明します。

1. M375_STK.uvprojx をダブルクリック、または[ファイル] > [開く]から M375_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



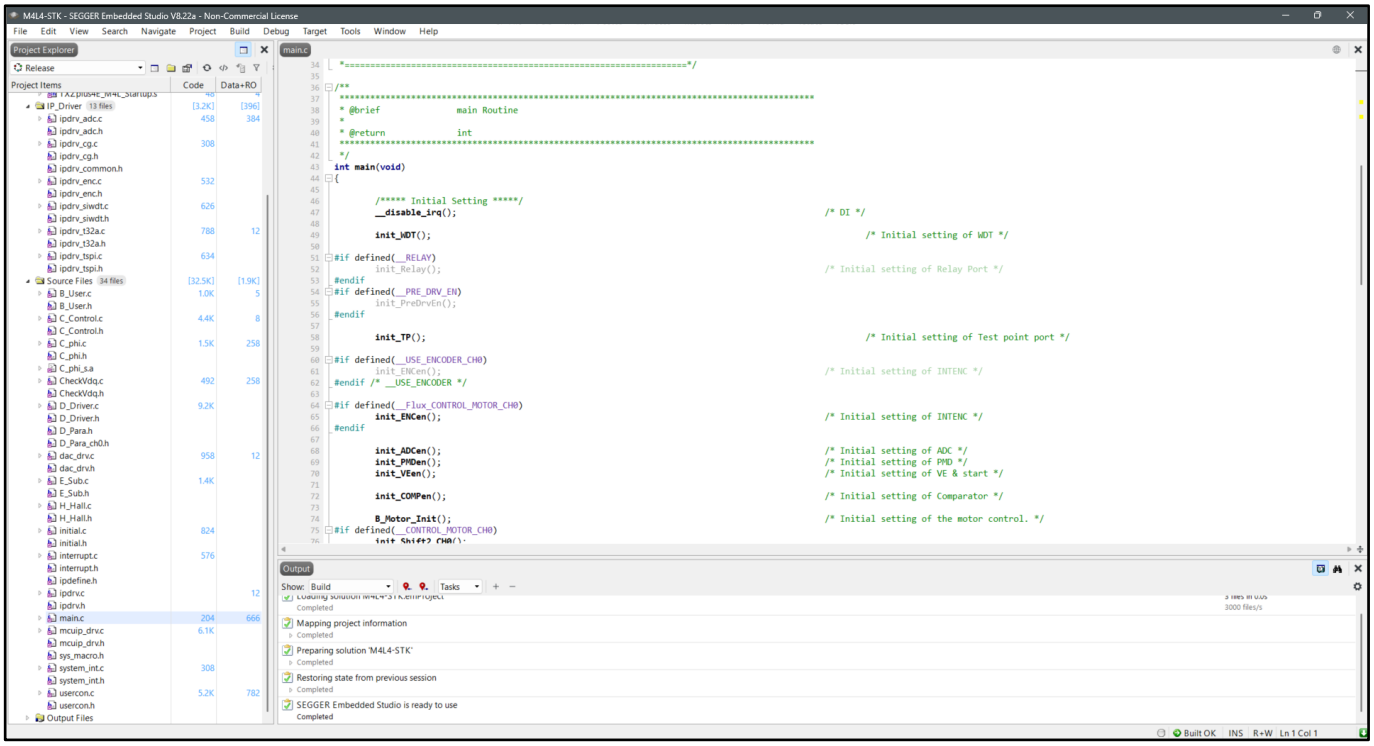


使用するツールを選択してください。

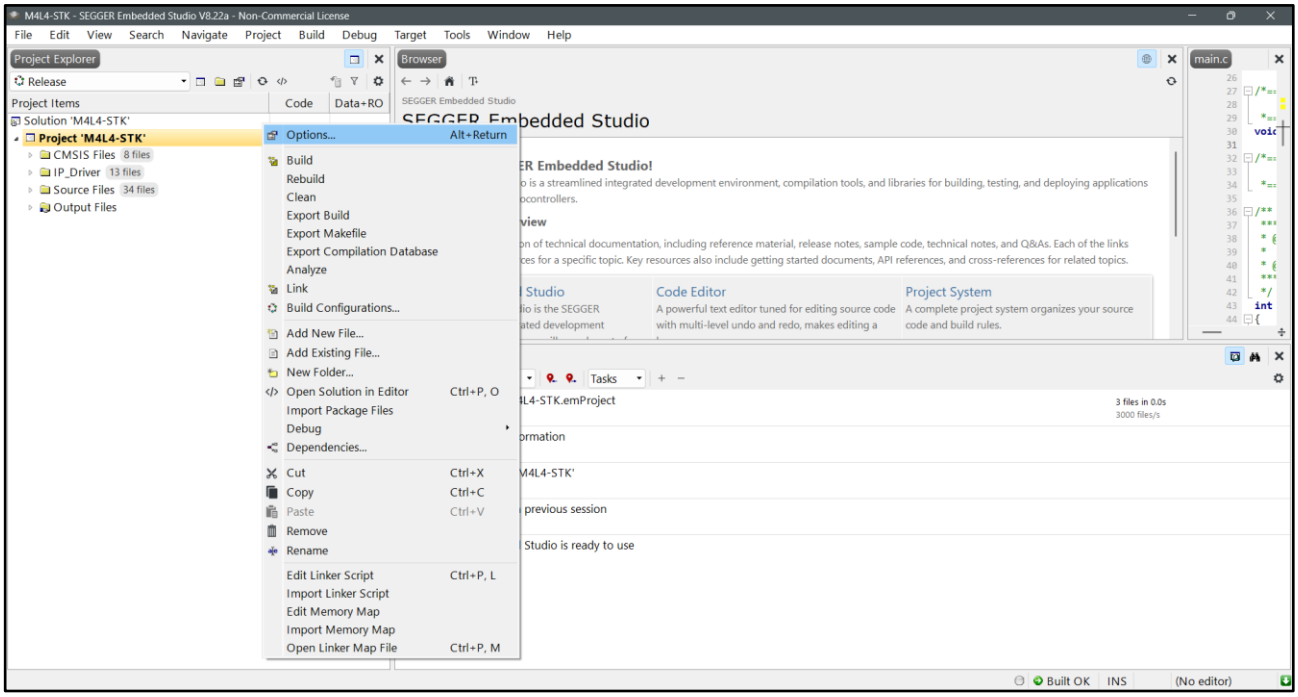
4. デバッグを開始するときは、ツールを接続し[デバッグ] > [デバッグセッションの開始/停止]を選択するか、下記ボタンを押してください。

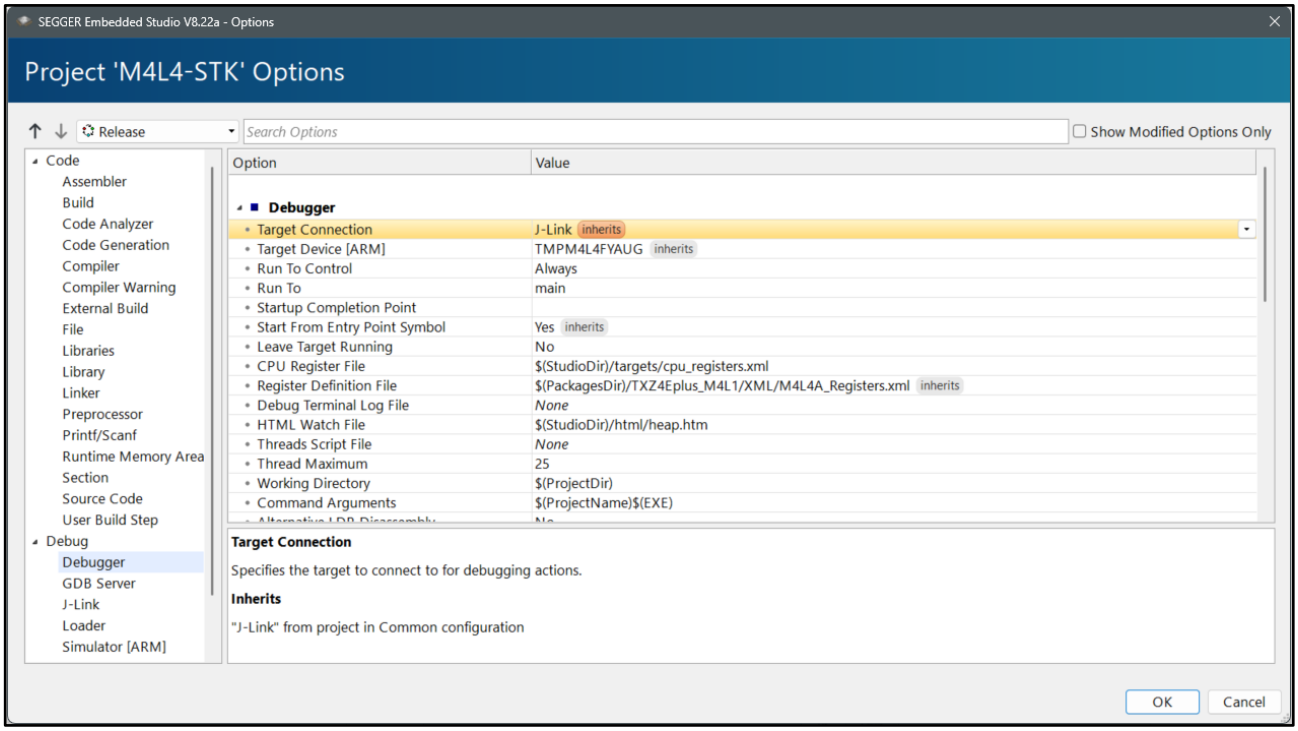
Segger Embedded Studio の場合を説明します。

1. M375-STK.emProject をダブルクリック、または[File] > [Open]から M375-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>





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。

4.3. DAC 出力

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

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

≪DAC 出力設定用変数≫
dac.select　　DAC 出力選択
dac.motch　　DAC 出力させるモーターCH を設定
dac.datsft0 - 3　ビットデータを左シフトする量(x)を設定
　　計算式(x)：data × (2^x)

4.4. 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。

4.4.1. 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
CH1	PE bit1

H：演算中

4.4.2. 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
CH1	PE bit1

L：正常検出状態、H：誤検出状態

4.5. 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

4.6. ユーザーインターフェースについて

ユーザーインターフェースとして下記機能を用意しています。

≪ユーザーインターフェース内容≫

1. 回転方向切替

SW(TSW1)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. 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ができます。
回転数調整：0, cHz_MIN～200Hz

3. 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。
LED 表示を有効にするには設定が必要となります。
詳細は 11.1.4　LED/PC UART 出力選択を参照してください。
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
　モーター駆動停止で EMG 状態が解除されます。

4. UART による初期表示

リセット解除後に 1 回 PC 上に、メッセージとボード名を表示することができます。
詳細は 9.1.6　ターミナルソフト表示を参照ください。

5. モジュール構成

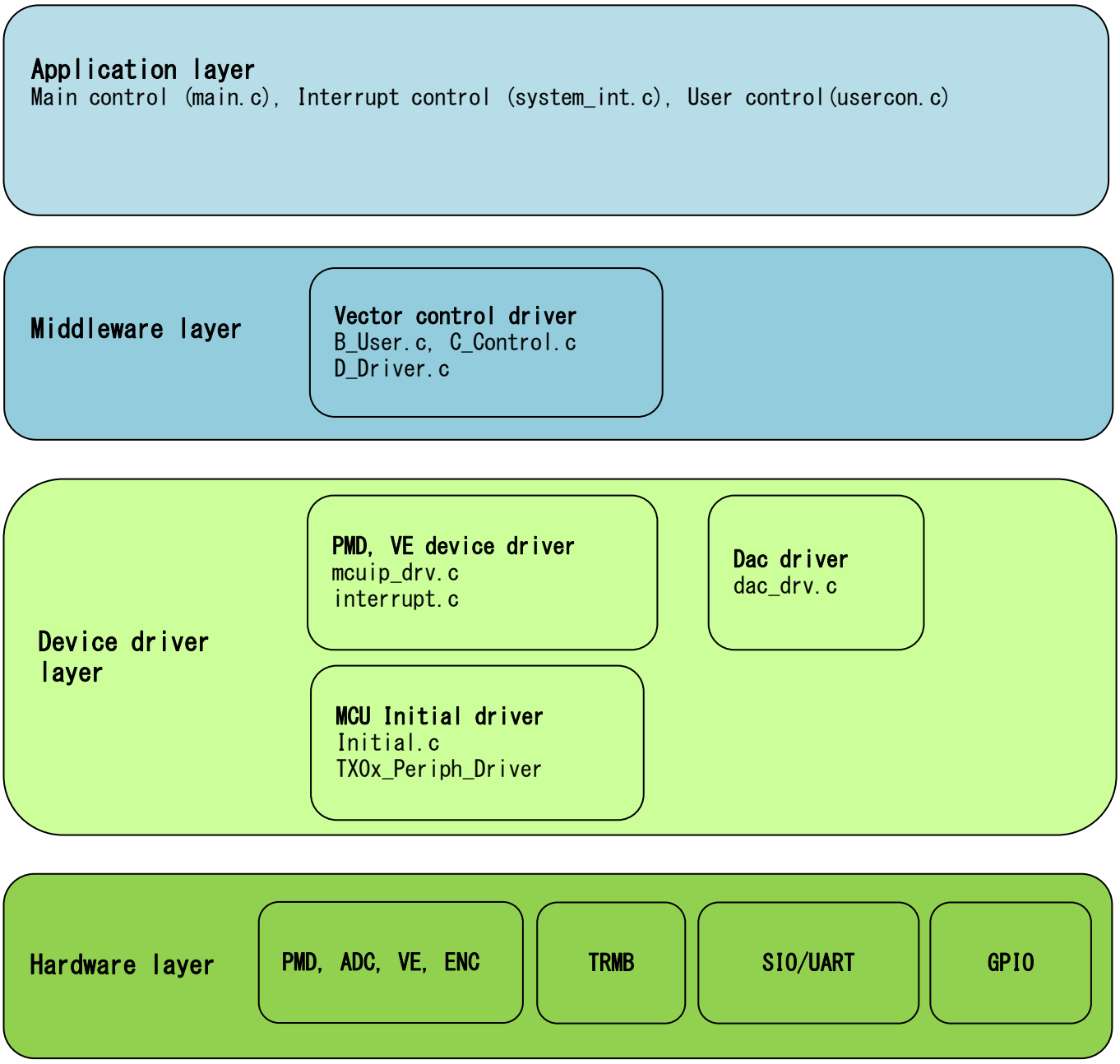


図 5.1　モジュール構成

6. マイコンハードウェア割り付け

6.1. IP

表 6.1 IP

IP		制御内容	備考
TMRB	TRMB0	4kHz 周期割り込み	
	TRMB4	未使用	
	TRMB5	未使用	
SIO/UART	チャネル 0	DAC IC 制御	SIO で使用
	チャネル 1	PC/UART	UART で使用
	チャネル 2	未使用	
	チャネル 3	未使用	
ADC	ユニット A	未使用	
	ユニット B	モーターCH1 モーター電流、VDC 電圧値取得 VR: ADKey 電圧値取得	
PMD	チャネル 1	モーターCH1 制御	
VE	チャネル 1	モーターCH1 制御	
ENC	チャネル 1	モーターCH1 エンコーダー信号制御	

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
INTTB00_IRQn	4kHz 周期タイミング作成	5	INTTB00_IRQHandler
INTVCNB_IRQn	ベクトル制御ソフト処理 for CH1	3	INTVCNB_IRQHandler
INTTX0_IRQn	DAC IC 制御	6	INTTX0_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_DAC_LEVEL   6    /* SIO interrupt for DAC */
```

7. ジェネラルフロー

7.1. メインルーチン(main)

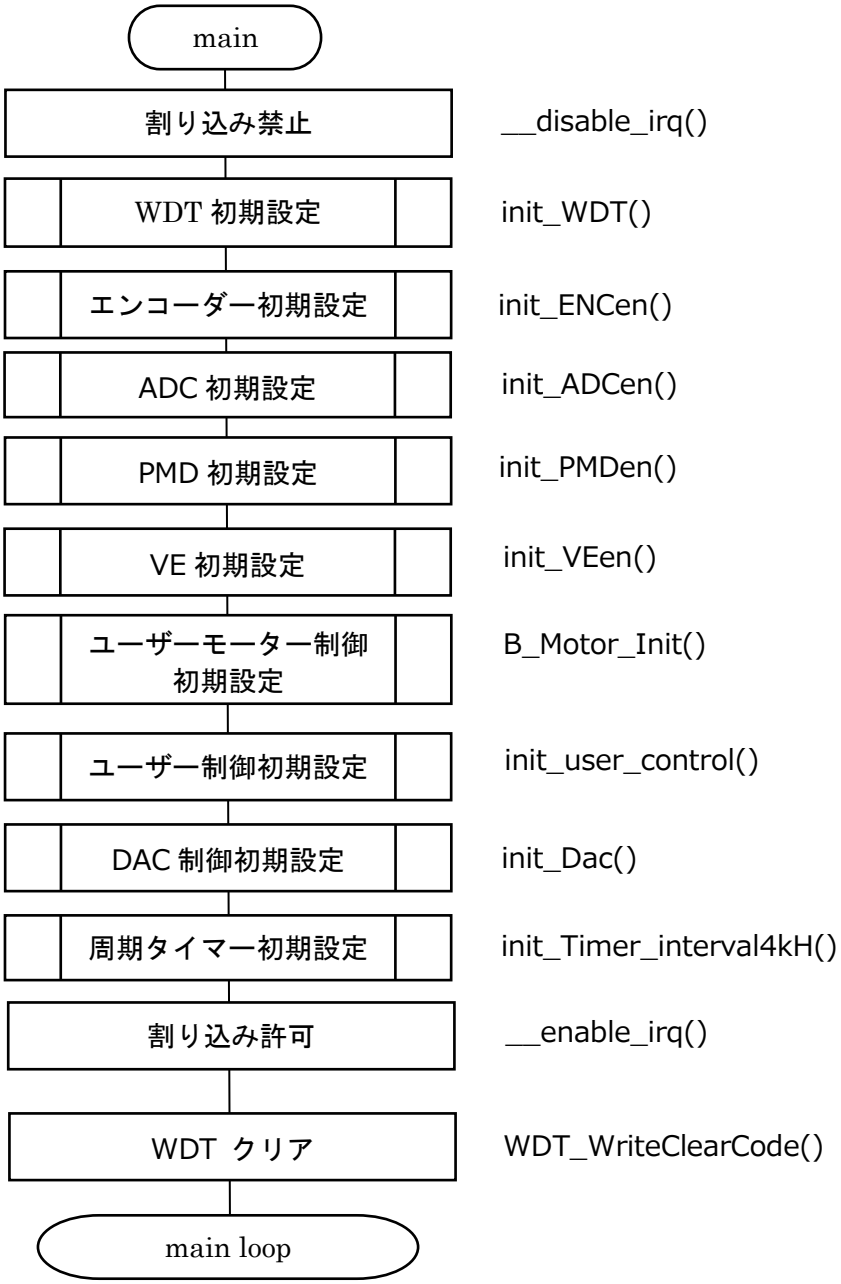


図 7.1　メインルーチン

7.2. メインループ(main_loop)

図 7.2　メインループ

7.3. 割り込み(interrupt.c)

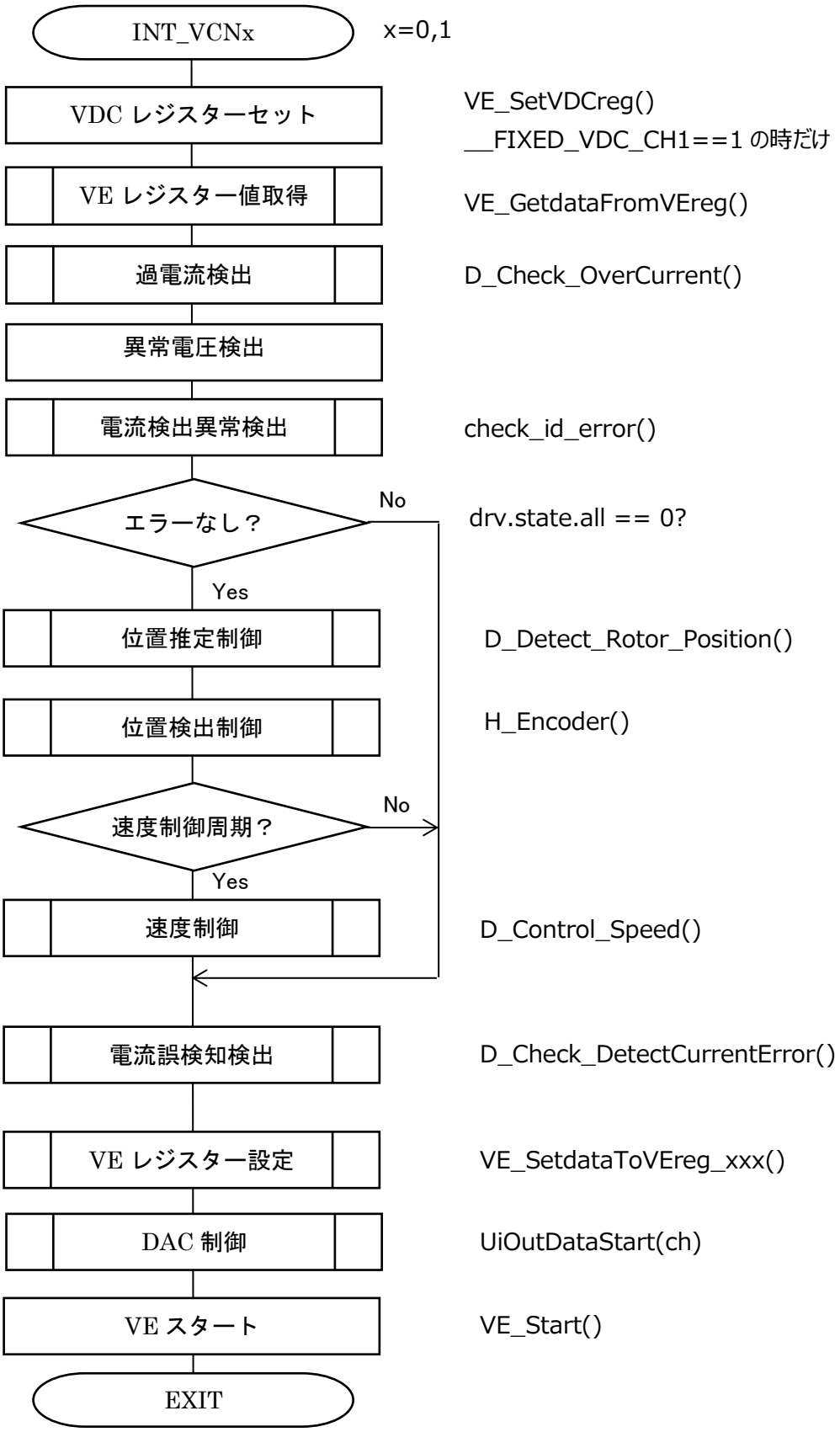


図 7.3　割り込み

8. 状態遷移

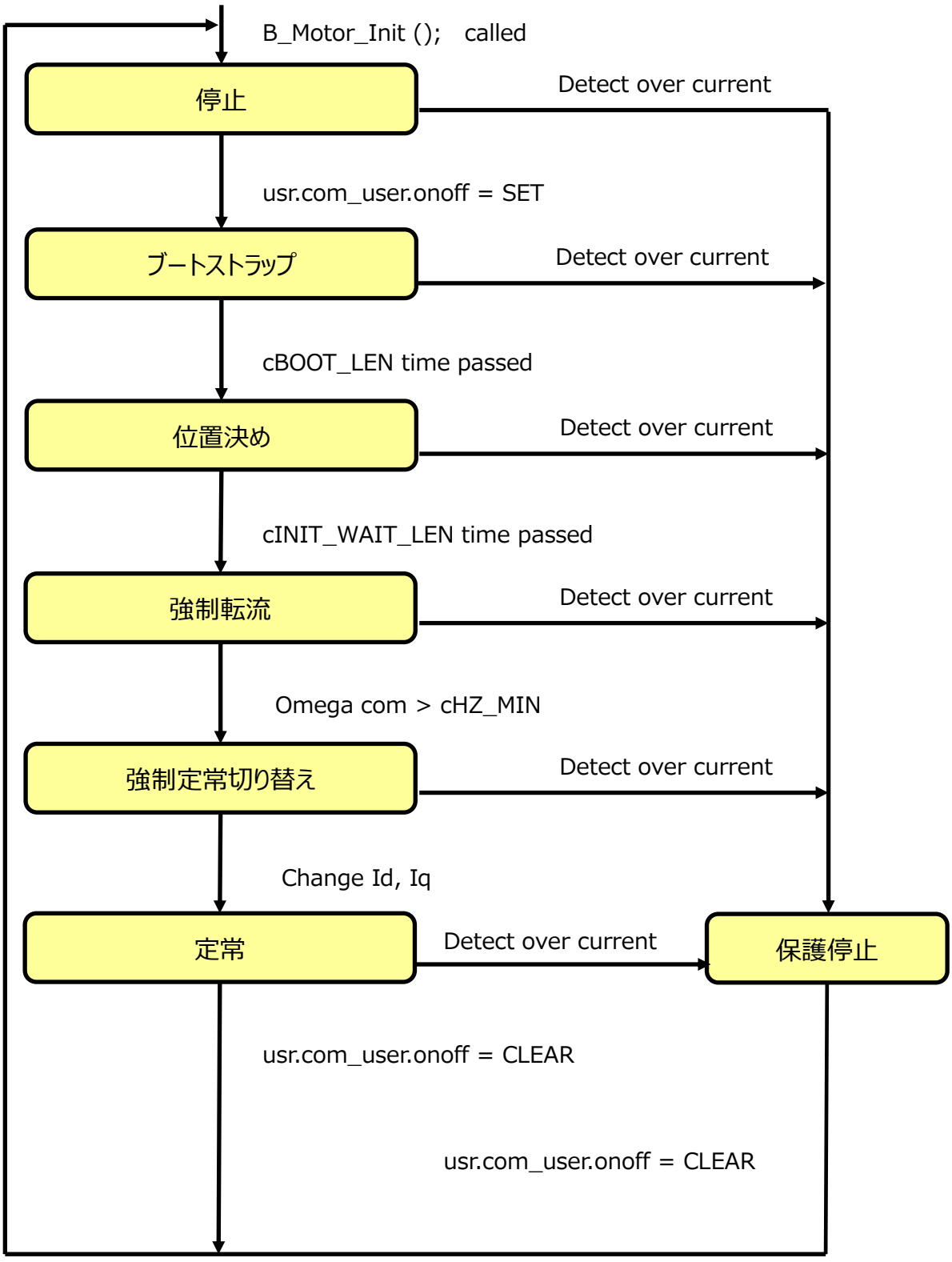


図 8.1　モーター動作の状態遷移

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。

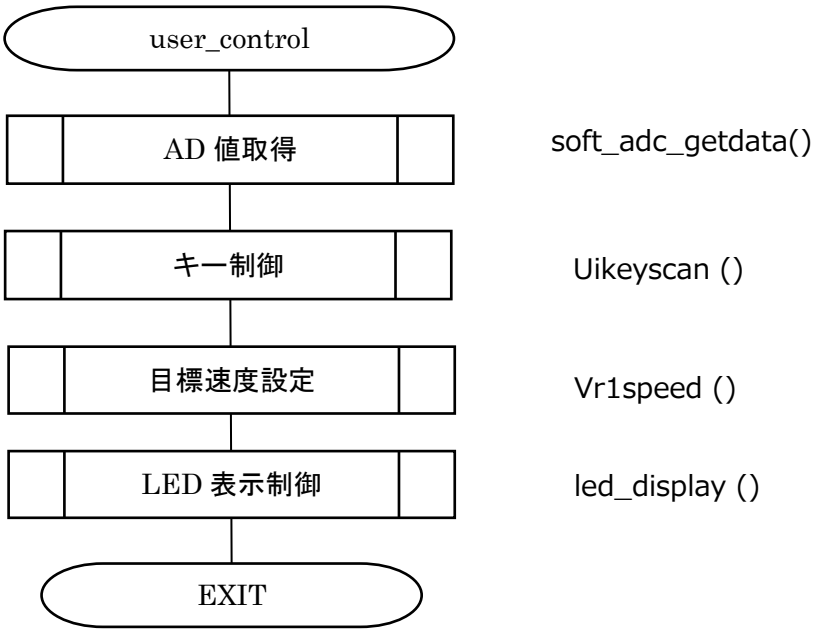


図 9.1　ユーザー制御

9.1.1. AD 値取得(soft_adc_getdata)

VR1 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 ad_vr.avedat に格納されます。

9.1.2. キー制御(Uikeyscan)

TSW1 のキースキャン処理を行います。
各キーデータは 20 回連続一致で確定し、keydata に格納されます。

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)
   keydata.tsw1 の値に従い、回転方向を切り替えます。
   keydata.tsw1 が 1 の場合、CW(vr1_speed：正)
   keydata.tsw1 が 0 の場合、CCW(vr1_speed：負)

2. 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： cHz_MIN～199Hz

9.1.4. LED 表示制御(led_display)

EMG ステータスに従い、LED5 の点灯制御を行います。
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。

9.1.5. UART 送信データ設定(UartDrv_putc)

初期表示用データ設定を行います。
通信設定は
115200bps、データ 8bit、ストップビット 1bit、パリティーなし、フロー制御なし
となります。

9.1.6. ターミナルソフト表示

TeraTerm 上などのターミナルソフトで文字列を表示することができます。
・TeraTerm 通信設定
　115200bps、データ 8bit、ストップビット 1bit、パリティーなし、フロー制御なし

・表示内容
　リセット解除後に 1 回、下記のとおりに表示を行います。
　Hello!
　SBK-M375

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. 制御指令(usr.com_user)

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

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

```
command_t   com_user;
```

アプリケーション層では、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(usr.omega_user)

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

アプリケーション層では、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(usr.Id_st_user, usr.Iq_st_user)

```
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 */
```

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

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;
モーター制御は、drv.command を介して、駆動方法をモーター制御ドライバー側へ渡します。

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

それぞれのステージではベクトル制御駆動コマンド（drv.vector_cmd）を以下のように設定し、モーター駆動に指令しています。
VE を使用したベクトル制御では、F_vcomm_volt と F_vcomm_Idetect は使用しておりません。

表 10.1　ベクトル制御コマンド

cmd / Stage	theta	omega	current	Volt	Edetect	Idetect	onoff
Stop	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR
Bootstrap	-	-	-	-	-	-	-
InitPosition	CLEAR	CLEAR	CLEAR	SET	CLEAR	SET	SET
Forced	SET	CLEAR	CLEAR	SET	SET	SET	SET
Change_up	SET	SET	CLEAR	SET	SET	SET	SET
Steady	SET	SET	SET	SET	SET	SET	SET
Emergency	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR	CLEAR

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

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

3) F_vcomm_current
速度制御で、d、q 軸電流の基準値の算出方法を指令します。
SET のとき、速度偏差から PI 制御で求めた値を基準値とします。CLEAR では PI 制御を実行せず、指令値をそのまま基準値とします。

4) F_vcomm_volt

電流制御で、d、q 軸電圧の基準値の算出方法を指令します。
SET のとき、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 制御を実行しないで、指令値をそのまま基準値とします。

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

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

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

① Loss_sync　脱調検出
脱調を検出したとき SET されます(未実装)

② emg_DC　異常電圧検出
異常電圧を検出したとき SET されます。

③ emg_I　電流検出異常
電流検出の異常を検出したとき SET されます。(未実装)

④ emg_S　ソフト過電流検出
ソフト処理で過電流を検出したとき SET されます。

⑤ emg_H　ハード過電流検出
マイコンハード機能で過電流を検出したとき SET されます。

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_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	---	
q31_u	drv.Vdq_ave	dq 軸電圧平均値	---	
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	---	
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	
q31_u	drv.Iao_ave	a 相ゼロ電流平均 AD 値	Q0	
q31_u	drv.Ibo_ave	b 相ゼロ電流平均 AD 値	Q0	
q31_u	drv.Ico_ave	c 相ゼロ電流平均 AD 値	Q0	

uint8_t	drv.spdprd	速度制御周期	Q0	
q15_t	drv.omega_enc	エンコーダー速度	Q15	
q15_t	drv.omega_enc_raw	エンコーダー速度(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	para.omega_min	強制転流終了速度	Q15	
q15_t	para.omega_v2i	電流制御切替速度	Q15	
q15_t	para.spwm_threshold	PWM シフト切り替え速度	Q15	1 シャント時だけ
q31_t	para.vd_pos	位置決め指令電圧	Q31	
q31_u	drv.Valpha	α軸電圧	Q31	
q31_u	drv.Vbeta	β軸電圧	Q31	
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.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	強制定常切替後待ち時間	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替位相	Q0	
uint16_t	para.chkpls	電流検出保護 Duty 幅	Q0	
uint32_t	stage_counter	ステージカウンター	Q0	
shunt_type_e	shunt_type	シャントタイプ	---	c1shunt c3shunt
boot_type_e	boot_type	起動タイプ	---	cBoot_i cBoot_v
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.Vdq_dc_rate	電圧利用率	Q15	

10.5. ユーザー関数

ユーザー、モーター制御、モーター駆動の各処理で使用される関数を説明します。
ユーザー処理は main 関数および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。

10.5.1. モーター制御初期設定(B_Motor_Init)

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。

表 10.3 モーター制御初期設定変数

名前	意味	Q フォーマット	備考
shunt_type	シャントタイプ	—	
boot_type	駆動タイプ	—	
usr.com_user.encoder	エンコーダー制御	—	
stage.main	メインステージ	—	
stage.sub	サブステージ	—	
drv.Iao_ave	U 相ゼロ電流平均 AD 値	Q0	
drv.Ibo_ave	V 相ゼロ電流平均 AD 値	Q0	
drv.Ico_ave	W 相ゼロ電流平均 AD 値	Q0	
usr.Iq_st_user	始動 Iq 電流値	Q15	
usr.Id_st_user	始動 Id 電流値	Q15	
usr.lambda_user	初期ローター位置	Q0	
usr.com_user.modul	変調方式	—	
usr.com_user.spwm	シフト PWM ON/OFF	—	
para.motor.r	モーター巻線抵抗	Q15	
para.motor.Lq	モーターq 軸インダクタンス	Q15	
para.motor.Ld	モーターd 軸インダクタンス	Q15	
para.spwm_threshold	シフト PWM 切り替え速度	Q15	1 シャント 2 相変調だけ
para.chkpls	最小パルス幅 or 最大パルス幅	Q0	
para.vd_pos	位置決め指令電圧	Q31	
para.spd_coef	出力電圧係数	Q15	
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	強制定常切り替え後待ち時間	Q0	
para.omega_min	強制転流終了速度	Q15	
para.omega_v2i	電流制御切り替え速度	Q15	
para.delta_lambda	強制定常切り替え時電流切り替え位相	Q0	
para.pos.ki	位置推定積分ゲイン	Q15	
para.pos.kp	位置推定比例ゲイン	Q15	
para.pos.ctrlprd	位置推定制御周期	Q0	

para.spd.ki	速度制御積分ゲイン	Q15	
para.spd.kp	速度制御比例ゲイン	Q15	
para.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	Q0	エンコーダー使用時
para.enc.deg_adjust	電気角調整	Q0	エンコーダー使用時
para.enc.plsnum	エンコーダーパルス数	Q0	エンコーダー使用時
para.enc.width2omega	パルス幅時間から速度への演算係数	Q15	エンコーダー使用時
para.enc.pls2omega	パルス数から速度への演算係数	Q15	エンコーダー使用時
para.enc.ctrlprd	エンコーダー制御周期	Q16	エンコーダー使用時

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

モーターの ON/OFF、回転数、駆動方式などの設定を行います。
過電流(ハードウェア EMG)状態のチェックを行います。

表 10.4　ユーザーモーター制御変数

名前	意味	Q フォーマット	備考
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	---	

10.5.3. ユーザー制御(user_control)

・TSW1 入力
モーターの回転方向の切り替えを行います。
モーターが回転中に回転方向の切り替えを行うとモーターが停止します。

・VR(ボリューム抵抗)
モーターの速度指令値の設定を行います。
回転数を 12Hz～200Hz の範囲で設定が可能です。
cSPEED_ACT_MIN と cSPEED_ACT_MAX を変更すると上記の値を変更可能です。

・LED 出力
EMG 状態中は LED5 が点灯します。EMG 状態が解除されると LED5 を消灯します。

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、強制定常切り替え状態、 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)、強制定常切り替え(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 のとき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. 処理内容

アプリケーションから与えられる制御コマンドおよび現在の状態を監視し、状態遷移を実行します。

各状態はさらに詳細化されたサブ状態に分かれます。サブ状態の遷移は状態遷移処理関数ではなく、各状態の処理関数内で実行されます。

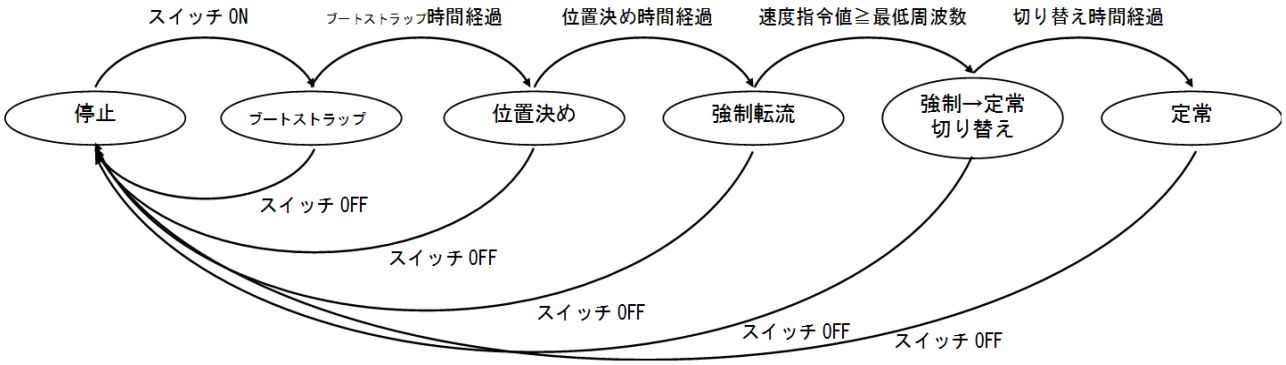


図 10.1　モーター制御の状態遷移図(センサーレス制御)

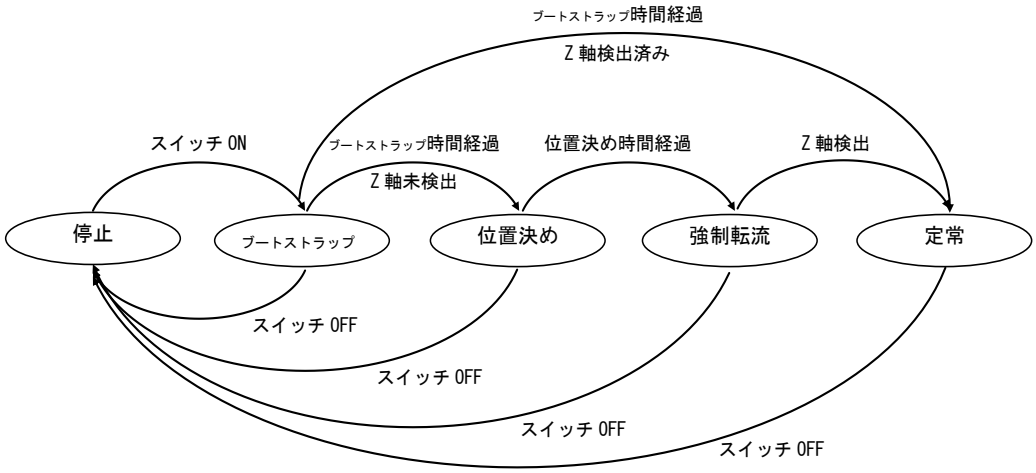


図 10.2　モーター制御の状態遷移図(エンコーダー制御)

10.6.2. モーター制御共通処理関数(C_Common)

モーター制御の各状態に共通な処理を実行します。

10.6.3. 停止状態関数(C_Stege_Stop)

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)

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

ブートストラップ状態を以下のサブ状態に分けて制御します。

a) 初期状態
ブートストラップ状態の初期設定を行います。

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

表 10.5 ブートストラップ状態関数変数

名前	意味	Q フォーマット	備考
para.time.bootstp	ブートストラップ時間	Q0	* MainLoopPrd(s)
stage_counter	ステージカウンター	Q0	* MainLoopPrd(s)

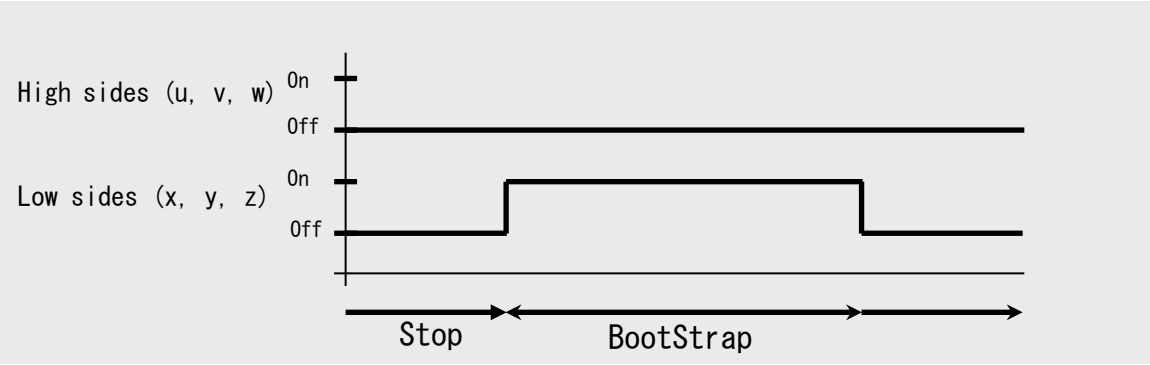


図 10.3 ブートストラップ時間

10.6.5. 位置決め状態関数(C_Stage_Initposition)

ローターを初期位置に固定させます。

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

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

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

位置決め状態を以下のサブ状態に分けて制御します。

a) 初期状態
位置決め状態の初期設定を行います。

b) Id 増加状態
Id を設定値まで徐々に増加させます。

表 10.6 位置決め状態関数変数

名前	意味	Q フォーマット	備考
drv.vector_cmd	駆動コマンド	—	
boot_type	起動タイプ	—	
stage_counter	ステージカウンター	Q0	* MainLoopPrd(s)
usr.Id_st_user	始動 Id 電流値	Q15	
usr.lambda_user	初期ローター位置	Q0	
drv.Id_com	d 軸電流指令値	Q31	
drv.Iq_com	q 軸電流指令値	Q31	
drv.omega_com	周波数指令値	Q31	
drv.theta_com	電気角指令値	Q0	
drv.Vd_out	出力電圧値	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	
para.vd_pos	位置決め指令電圧	Q31	

10.6.6. 強制転流状態関数(C_Stage_Force)

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

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

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

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

表 10.7　強制転流状態関数変数

名前	意味	Q フォーマット	備考
drv.vector_cmd	駆動コマンド	—	
boot_type	起動タイプ	—	電流 または 電圧
usr.Id_st_user	始動 Id 電流値	Q15	
usr.omega_user	制御目標速度	Q31	
drv.Id_com	d 軸電流指令値	Q31	
drv.Iq_com	q 軸電流指令値	Q31	
drv.omega_com	駆動速度指令値	Q31	
drv.Vd_out	出力電圧値	Q31	電圧駆動時だけ有効
para.sp_ud_lim_f	駆動速度増減リミット値	Q31	
para.omega_min	強制転流終了速度	Q15	
para.omega_v2i	電流制御切り替え速度	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

10.6.7. 強制定常切替状態関数(C_Stage_Change_up)

Id を 0 に減少、Iq を「始動 Iq 電流」まで増加させ、磁界の方向がローターと直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。

「駆動目標速度」を一定値ずつ増加させて「制御目標速度」に近づけます。ただしこのステージでは速度制御は行っていないため、「駆動目標速度」は制御には使用されません。

強制定常切換え状態を以下のサブ状態に分けて制御します。

a) 初期状態
強制定常切換え状態の初期設定を行います。

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

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

表 10.8 強制定常切替状態関数変数

名前	意味	Q フォーマット	備考
drv.vector_cmd	駆動コマンド	Q0	
stage_counter	ステージカウンター	Q0	ms
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	
drv.Id_com	d 軸電流指令値	Q31	
drv.Iq_com	q 軸電流指令値	Q31	
drv.omega_com	駆動速度指令値	Q31	
para.time.go_up	強制定常切り替え後待ち時間	Q0	
para.sp_ud_lim_f	駆動速度増減リミット値	Q31	

10.6.8. 定常状態関数(C_Stage_Steady_A)

定常状態の処理を実行します。
「駆動目標速度」を一定ずつ増加させて「制御目標速度」に近づけます。

表 10.9　定常状態関数変数

名前	意味	Q フォーマット	備考
drv.vector_cmd	駆動コマンド	Q0	
usr.omega_user	制御目標速度	Q31	
usr.lambda_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
drv.Id_com	d 軸電流指令値	Q31	
drv.omega_com	駆動速度指令値	Q31	速度制御時
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	

10.6.9. 保護状態関数(C_Stage_Emergency)

過電流が発生したとき、この状態へ遷移します。
ハードウェア過電流を検出したときは、モーター駆動出力 u、v、w、x、y、z は全て Hi-z となっています。
ソフトウェア過電流を検出したときは、モーター駆動出力 u、v、w、x、y、z は全て OFF となっています。
ローターは惰性で回転します。過電流状態復帰処理を実施するまでこのステージを維持します。

10.6.10. シフト PWM 制御(C_ShiftPWM_Control)

この制御は 1 シャントのときだけ有効です。
シフト PWM の ON/OFF および電流検知前回値を使用する最小 Duty 値の設定を行います。
シフト PWM 制御方法、割り込みステージ、目標速度からシフト PWM 駆動方法を決定します。
サンプルソフトでは表 10.10　シフト PWM 駆動方法条件の条件設定となっています。

表 10.10　シフト PWM 駆動方法条件

シフト PWM 制御方法 usr.com_user.spwm	割り込みステージ stage.itr	目標速度 drv.omega_com	シフト PWM 駆動方法 drv.command.spwm
OFF (0)	All	All	OFF (0)
ON　(1)	Stop	All	OFF (0)
	Emergency	All	OFF (0)
	Initposition_i	All	OFF (0)
	Force_i Change_up Steady_A	目標速度 ≧シフト切り替え速度 drv.omega_com ≧ para.spwm_threshold	OFF (0)
		目標速度 <シフト切り替え速度 drv.omega_com < para.spwm_threshold	ON (1)

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

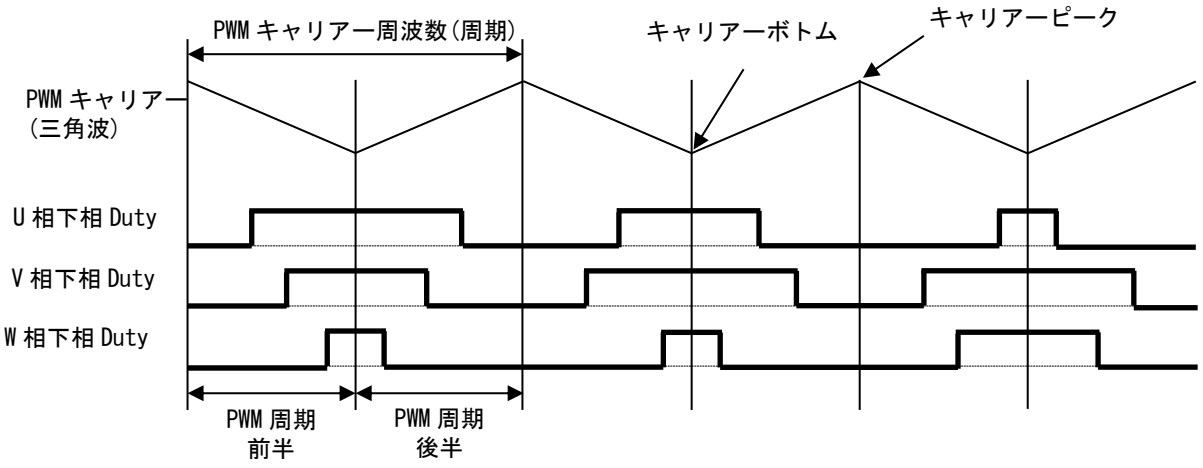


図 10.4　3 相変調の場合

10.7.1.2. 2 相変調の場合

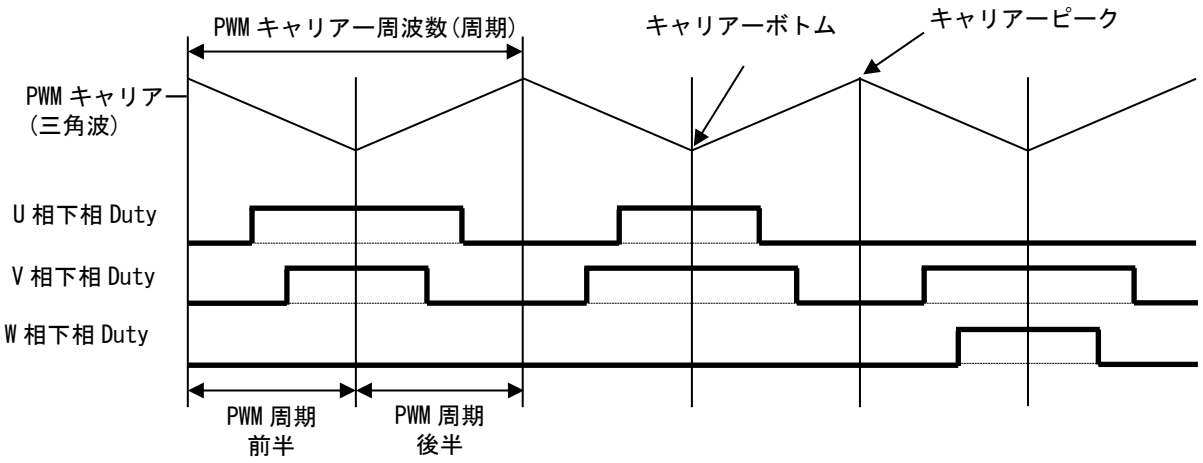


図 10.5　2 相変調の場合

10.7.2. 位置推定関数(D_Detect_Rotor_Position)

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

モーターの d 軸に関する等価回路方程式は、下記で表すことができます。

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

p=d/dt, Id≈一定値より pId=0 とすることができる。)

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

よって、d 軸誘起電圧 Ed は下記演算式で求めることができます。

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

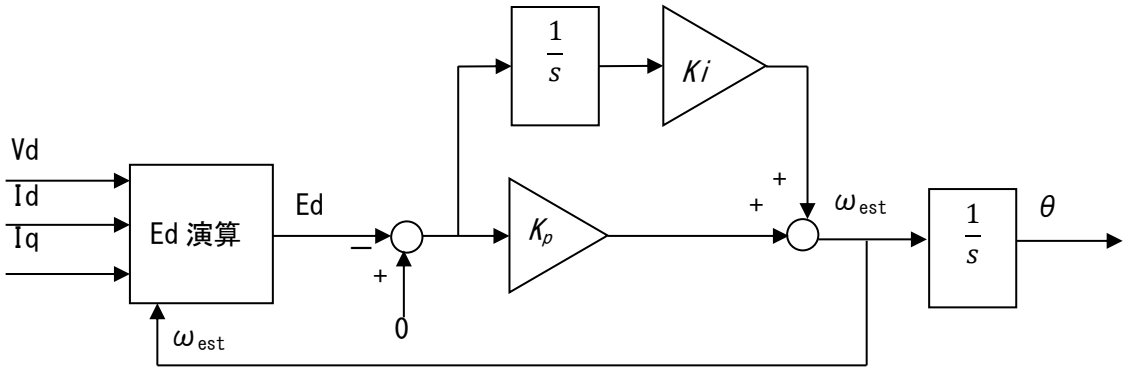


図 10.6　d 軸誘起電圧 Ed による位置推定ブロック図

表 10.11　位置推定関数変数

名前	意味	Q フォーマット	備考
drv.Ed	d 軸誘起電圧	Q15	
drv.Vd	d 軸電圧	Q31	
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.omega	推定速度	Q31	
drv.theta	ローター位置	Q31	
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	
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.pos.kp	位置推定比例ゲイン	Q15	
para.pos.ki	位置推定積分ゲイン	Q15	

10.7.3. エンコーダー制御関数(H_Encoder)

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。

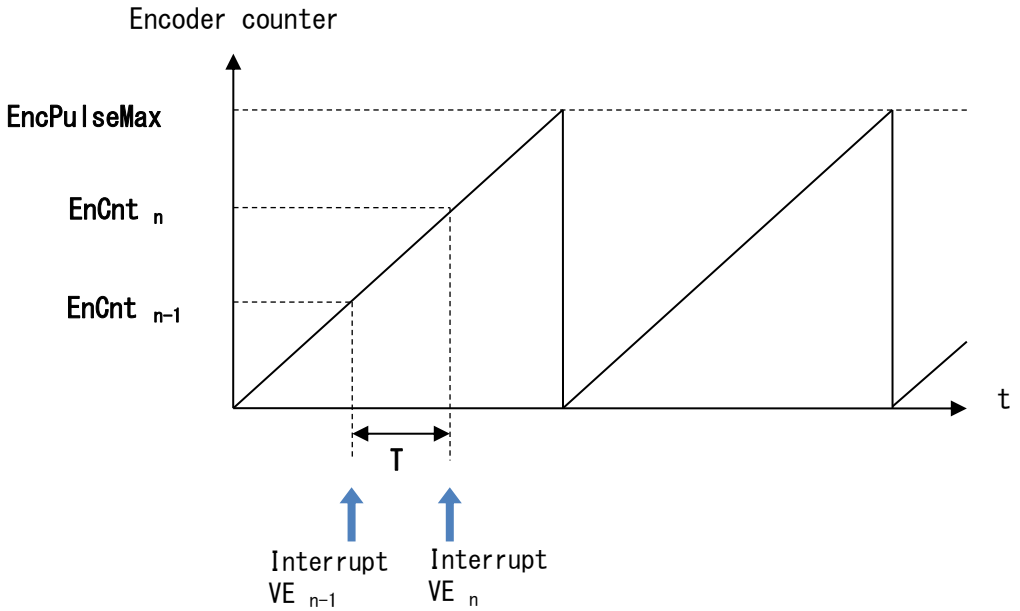


図 10.7　エンコーダーパルスからの位置および速度演算

VE 割り込みごとに、パルス数を読み込み、下記演算を行います。

現在のエンコーダーパルスカウンター値から下記式により θ を演算します。

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

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

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

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

また、M375 ではエンコーダーパルス間隔時間から下記式により速度 ω を演算します。

$$\omega=(Pole\div2)\div[\{(PulseTimerDiv\div PreFreq)\times CapPulseCnt\} \times(EncPulseMax\div EncPulseDiv)]$$

CapPulseCnt	エンコーダーパルス間隔測定キャプチャーカウント
EncPulseDiv	エンコーダーパルス信号入力分周比
PulseTimerDiv	エンコーダーパルス計測タイマー分周比
PreFreq	プリスケーラークロック[Hz]

10.7.4. 速度制御関数(D_Control_Speed)

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

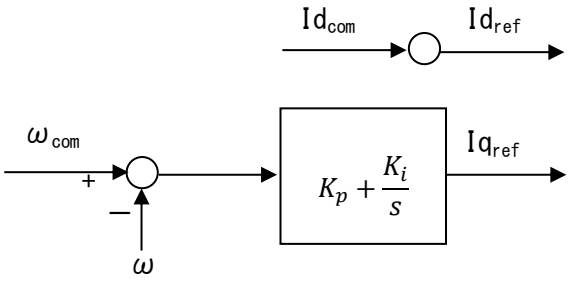


図 10.8　速度制御ブロック図

表 10.12　速度制御関数変数

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

10.7.5. 3 シャント電流誤検知検出関数 (D_Check_DetectCurrentError_3shunt)

10.7.5.1. 構文

int D_Check_DetectCurrentError_3shunt (uint32_t _duty, uint16_t _chkplswidth)

引数：
　　_duty　　Duty の中間値
　　_chkplswidth　　パルスチェック幅
戻り値：
　　電流検出状態

10.7.5.2. 変数

表 10.13　3 シャント電流誤検知検出関数変数

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

10.7.5.3. 処理内容

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

10.7.6. 1 シャント電流誤検知検出関数 (D_Check_DetectCurrentError_1shunt)

10.7.6.1. 構文

```
int D_Check_DetectCurrentError_1shunt (uint32_t _pwma, uint32_t _pwmb,
                                       uint32_t _pwmc, uint16_t _chkplswidth,
                                       int _sfhpwm, int _modu, int _sector,
                                       uint32_t _mdprd, 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
_chkplswidth	パルスチェック幅
_sfhpwm	シフト PWM モード状態
_modu	変調方式
_sector	セクター情報
_mdprd	PWM 周期

戻り値：
電流検出状態

10.7.6.2. 変数

表 10.14 1 シャント電流誤検知検出関数変数

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

10.7.6.3. 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。

10.7.7. インバーター出力電圧の算出関数 (Cal_Vdq)

10.7.7.1. 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)

10.7.7.2. 変数

表 10.15　インバーター出力電圧の算出関数変数

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

10.7.7.3. 処理内容

インバーター出力電圧(Vdq)を算出します。

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

11. 定数定義説明

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)

11.1.1. モーター制御チャネル選択

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

11.1.2. 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。

11.1.3. 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

11.1.4. LED/PC UART 出力選択

__LED_SW　　LED を使用する場合は、こちらを「1」に設定してください。
__PC_CH1　　PCUART を使用する場合は、こちらを「1」に設定してください。

11.1.5. パラメーター一覧

表 11.1　パラメーター一覧

パラメーター名	説明
cMAINLOOP_PRD	メイン周期設定
	単位[s] 分解能 4kHz
	メイン周期の時間を設定してください。
cIXO_AVE	ゼロ電流値用フィルター係数
	--
	フィルター係数を設定してください。 大きな値を設定すると安定しますが、反応が遅れます。
cVDQ_AVE	電源電圧 Vdc,出力電圧 Vdq 用フィルター係数
	--
	フィルター係数を設定してください。 大きな値を設定すると安定しますが、反応が遅れます。
cOMEGAENC_AVE	エンコーダーパルス数からの速度検出値用フィルター係数
	--
	速度検出用のフィルター係数を設定してください。 エンコーダーパルス数が小さい場合は、値を大きめにしてください。 ただし、値を大きくすると検出の遅れが発生しますので、遅れが問題ない値に設定してください。
cOMEGAPLS_FLTRE	エンコーダーパルス数からの速度検出値用フィルター係数
	--
	速度検出用のフィルター係数を設定してください。

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ネル(D_Para_chx.h) x=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。

11.2.1. 制御選択

11.2.1.1. AMP 選択

__USE_INAMP　　内蔵 AMP を使用する場合、定義してください。

11.2.1.2. センサーレス/センサー選択

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

11.2.2. モーターチャネル別パラメーター一覧

表 11.2　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
	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
	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]
	AD 変換が 1LSB 変化する電源電圧の変化量[V]×2^12 の値を設定してください。
cA_MAX	最大電流値の設定
	単位[A]
	AD 変換が 1LSB 変化する相電流の変化量[A]×2^11 の値を設定してください。
cSHUNT_TYPE	電流取得方式(3 シャント または 1 シャント)の設定
	1:1 シャント/ 3:3 シャント
	電流取得方式の設定を行ってください。
cBOOT_TYPE	起動時の駆動方式の設定
	cBoot_i:電流型駆動/ cBoot_v:電圧型駆動
	起動時の駆動方式を設定してください。 1 シャント駆動の起動時など電流が取得できない場合などに、電圧型駆動を選択します。
cSHUNT_ZERO_OFFSET	オフセット電圧の設定
	単位[V]
	電流が流れていないときのシャント電圧を設定してください。 この値は、ゼロ電流平均値の初期値に使用します。
cADCH_CURRENT_U	U 相電流取得 AD チャネル設定(3 シャント用)
	AINx
	U 相電流を検出する AD チャネルを設定してください。

パラメーター名	説明
cADCH_CURRENT_V	V 相電流取得 AD チャネル設定(3 シャント用)
	AINx
	V 相電流を検出する AD チャネルを設定してください。
cADCH_CURRENT_W	W 相電流取得 AD チャネル設定(3 シャント用)
	AINx
	W 相電流を検出する AD チャネルを設定してください。
cADCH_CURRENT_IDC	電流取得 AD チャネル設定(1 シャント用)
	AINx
	電流を検出する AD チャネルを設定してください。
cADCH_VDC	電源電圧取得 AD チャネル設定
	AINx
	電源電圧 Vdc を検出する AD チャネルを設定してください。
cOVC	過電流検出値の設定
	単位[A]
	過電流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、出力をソフトで OFF にします。
cVDC_MINLIM	電源電圧 Vdc 最低値の設定
	単位[V]
	電源電圧 Vdc の最低値を設定してください。 この値未満の電圧値を検出すると、モーターを停止させます。
cVDC_MAXLIM	電源電圧 Vdc 最高値の設定
	単位[V]
	電源電圧 Vdc の最高値を設定してください。 この値より大きな値の電圧値を検出すると、モーターを停止させます。
cPWMPRD	PWM 周期の設定
	単位[μs] 分解能 25ns@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[μs] 分解能 100ns@80MHz
	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 使用の場合のみ)
	単位[回] 1~15
	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] または [RPS] または [RPM] 単位は__**TGTSPD_UNIT** の設定による

パラメーター名	説明
	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[Ω]
	モーターコイル 1 相分の抵抗値を設定してください。
cMOTOR_LQ	q 軸インダクタンス値
	単位[mH]
	モーターコイルの q 軸インダクタンス値を設定してください。
cMOTOR_LD	d 軸インダクタンス値(未使用)
	単位[mH]
	モーターコイルの d 軸インダクタンス値を設定してください。
cPOLE	モーター極数の設定
	単位[pole]
	モーターの極数を設定してください。
cID_KP	d 軸電流制御比例ゲイン
	単位[V/A]
	d 軸電流制御比例ゲインを設定してください。
cID_KI	d 軸電流制御積分ゲイン
	単位[V/As]
	d 軸電流制御積分ゲインを設定してください。
cIQ_KP	q 軸電流制御比例ゲイン
	単位[V/A]
	q 軸電流制御比例ゲインを設定してください。
cIQ_KI	q 軸電流制御積分ゲイン
	単位[V/As]
	q 軸電流制御積分ゲインを設定してください。
cPOSITION_KP	位置推定比例ゲイン

パラメーター名	説明
	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	強制定常切り替え後待ち時間
	単位[s] 分解能:1ms
	強制定常切り替え後待ち時間を設定してください。

パラメーター名	説明
cHZ_MAX	最大周波数
	単位[Hz]
	マイコンで検出させる最大周波数を設定してください。 制御で使用する最大周波数の 10~20%増し程度の値を設定してください。 値が小さいほど演算精度が上がりますが、検出値がこの値を超えると制御が破たんします。
cHZ_MIN	強制転流から定常への切り替え速度
	単位[Hz]
	強制転流から定常へ移行させる速度を設定してください。 位置推定ができる(誘起電圧が検出できる)速度を設定します。
cHZ_SPWM	シフト PWM 切り替え速度 (1 シャント 2 相変調用)
	単位[Hz]
	シフト PWM から通常 PWM へ切り替える速度を設定してください。 指令速度の値で切り替えます。
cMINPLS	最小パルス幅設定(1 シャント用)
	単位[μs]
	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しないで、前回値を使用して制御を行います。
cMAXPLS	最大パルス幅設定(3 シャント用)
	単位[μs]
	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しないで、前回値を使用して制御を行います。
cID_LIM	d 軸電流 limit
	単位[A]
	d 軸電流のリミット値を設定してください。
cIQ_LIM	q 軸電流 limit
	単位[A]
	q 軸電流のリミット値を設定してください。
cINITIAL_POSITION	初期位置
	単位[deg.]
	位置決め時の角度を電気角で設定してください。
cVD_POS	電圧駆動時の位置決め出力電圧
	単位[V]
	位置決め時の電圧を設定してください。 ※cBOOT_TYPE が"cBoot_v"のときだけ有効 詳細は 11.2.3.2　電圧型起動時の定数値を参照してください。
cSPD_COEF	電圧駆動時の強制転流出力電圧係数
	0 < x <1 の値を設定してください。
	強制転流時の出力電圧を決めます。

パラメーター名	説明
	この設定値と指令速度を掛けた値を出力電圧としています。 Vd = cSPD_COEF × Omega_com ※cBOOT_TYPE が"cBoot_v"のときだけ有効 詳細は 11.2.3.2　電圧型起動時の定数値を参照してください。
cHZ_V2I	電圧制御から電流制御への切り替え速度
	単位[Hz]
	電圧制御から電流制御へ切り替える速度を設定してください。 cHZ_MIN より大きな値を設定した場合、cHZ_MIN が以上の速度になると、定常状態へ移行し、電流制御に切り換わります。 ※cBOOT_TYPE が"cBoot_v"のときだけ有効
cENC_PULSE_NUM	エンコーダーパルス数設定
	単位[ppr]
	エンコーダーのパルス数を設定してください。 エンコーダーパルス数が小さい場合、速度検出精度が悪くなります。 検出速度が安定しない場合は、フィルター係数 cOMEGAENC_AVE の値を大きくしてください。
cENC_DEG_ADJUST	エンコーダー位置調整
	単位[deg.]
	電気角の 0°と Z 相の位置ずれ角度を設定してください。
__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
cSPEED_ACT_MIN	速度指令最小値
	単位[Hz] (電気角)
	VR の速度制御指令の最小値を設定してください。
cSPEED_ACT_MAX	速度指令最大値
	単位[Hz](電気角)
	VR の速度制御指令の最大値を設定してください。

11.2.3. 定数設定値と波形の関係

11.2.3.1. 電流型起動時の定数値

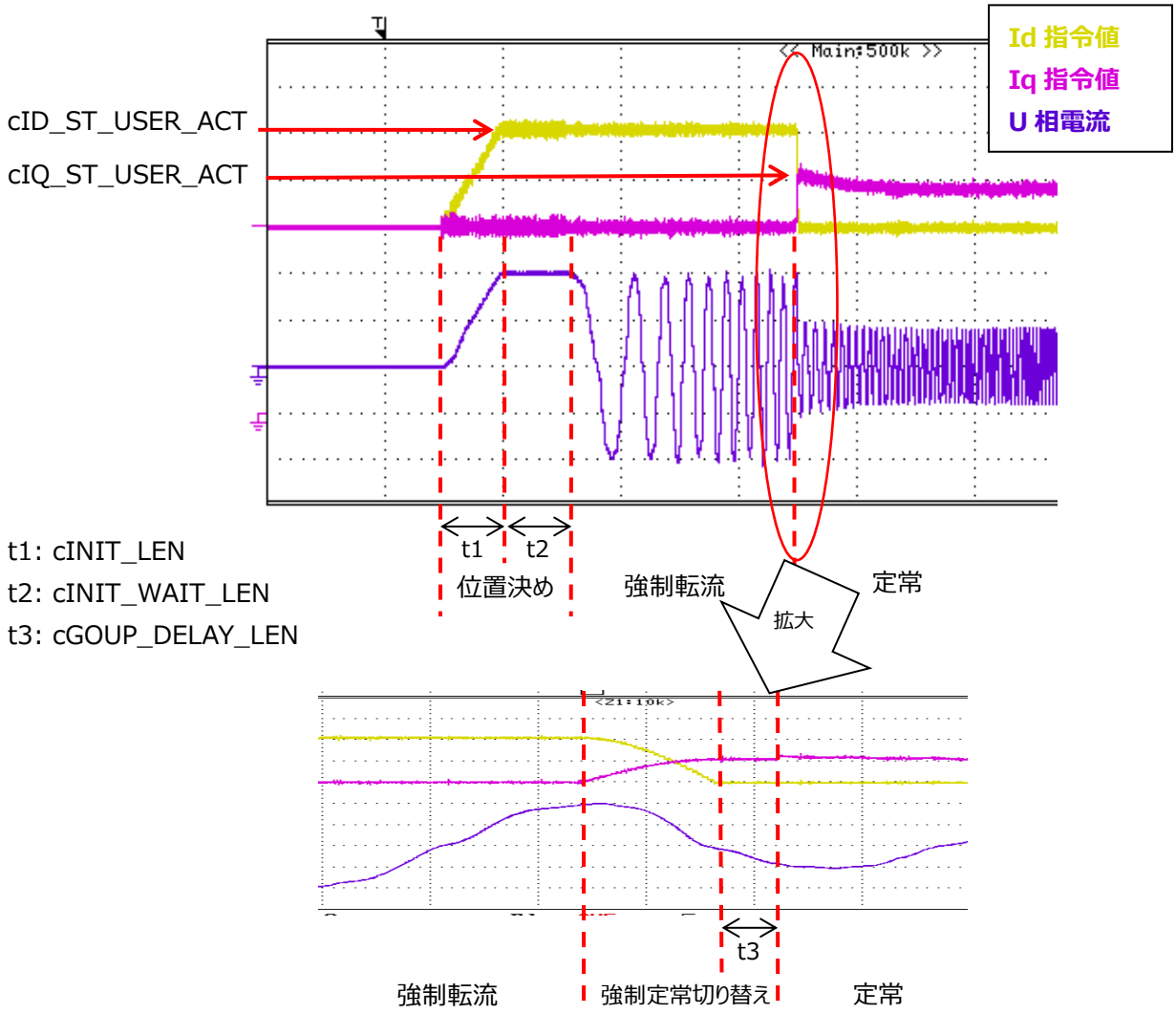


図 11.1　始動電流波形(電気角 0°に位置決め)

強制定常切り替えステージで、cID_ST_USER_ACT と cIQ_ST_USER_ACT の値を入れ替える。
入れ替え後、cGOUP_DELAY_LEN の時間 Iq 指令値一定値で制御。
定常移行後、Iq 指令値は PI 制御により演算。

11.2.3.2. 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。安全のため小さい値から
調整を始めてください。
目標値より小さいなら値を大きく、大きいなら値を小さくしてください。

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。電流の振幅値が小さくなっていくなら値を大きく、大きくなっていくなら値小さくしてください。

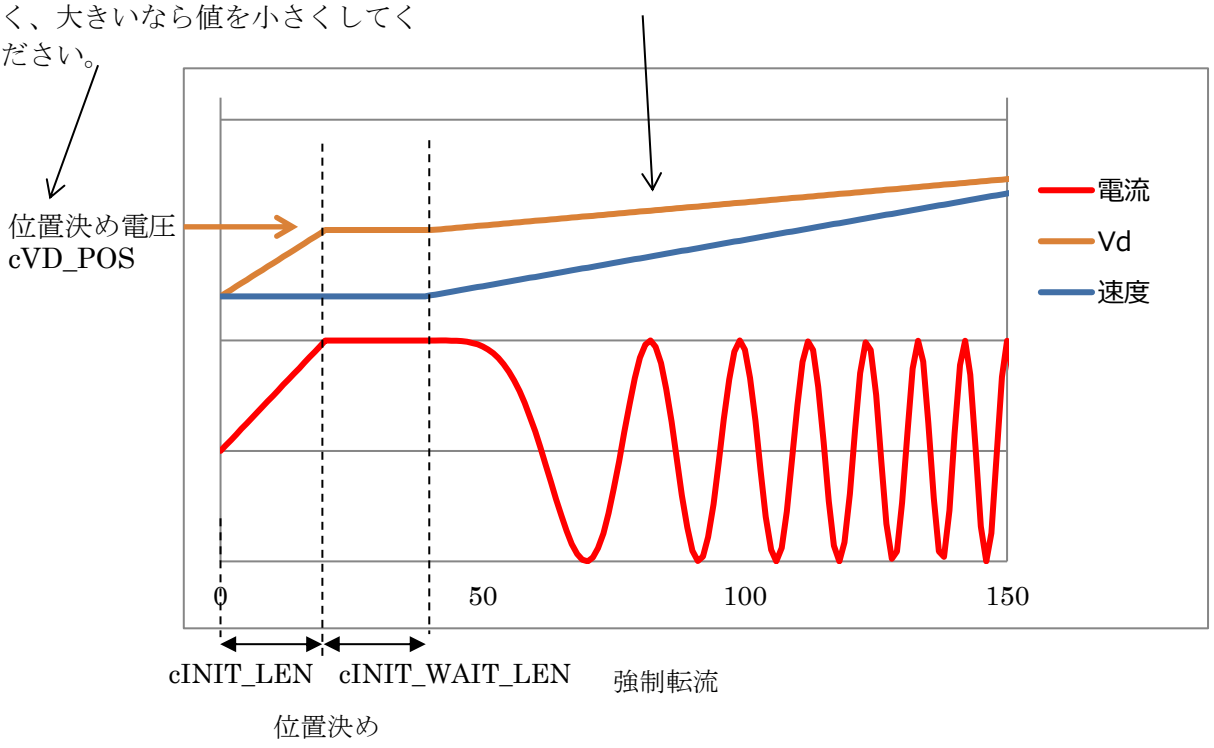


図 11.2 電圧駆動時のパラメーター調整方法

11.3. ユーザー制御関連定数

usercon.c

```
/* Key settting */
#define P_TSW1                  TSB_PB_DATA_PB5     /* TSW1 */
#define P_SW1                   TSB_PB_DATA_PB6     /* SW1 */※未使用
#define cKEY_CHATA_CNT          (20)                /* [cnt] chattering counter for KEY SW */

/* Soft ADC Setting */
#define  cADUNIT_USR            TSB_ADB             /* User ad data ADCUnit */
#define cADCH_VR                ADC_AIN9            /* ADC Channel for VR */
#define cADREG_VR               ADC_REG7            /* Result register No for VR */
#define cADAVECNT               (10)                /* ADC average count */

/* LED setting */
#define P_LED5                  TSB_PF_DATA_PF0     /* LED5 */

/* UART setting */
#define SIOCH_PCCOM             TSB_SC1             /* UART Channel for PC COM */
```

12. ペリフェラルドライバー

本ドライバーは、以下ファイルで構成されています。
mcuip_drv.c
mcuip_drv.h

12.1. ベクトルエンジン

12.1.1. 関数仕様

12.1.1.1. IP_VE_init

VE 初期設定
API：
void
IP_VE_init(TSB_VE_TypeDef* const VEx, VE_InitTypeDef* const _initdata)
パラメーター：
VEx：VE アドレスを選択します。
_initdata： VE 初期設定データ構造体
詳細は 12.1.2.1 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。
リターンパラメーター：
なし

12.1.1.2. VE_Start

VE スタート
API：
void
VE_Start(TSB_VE_TypeDef* const VEx);

パラメーター：
VEx：VE アドレスを選択します。

機能：
VE を開始します。
補足：
特になし
リターンパラメーター：
なし

12.1.1.3. VE_GetPhaseCurrent

相電流の取得

API：

```
void
VE_GetPhaseCurrent(TSB_VE_TypeDef* const    VEx,
                   q15_t*    _ia, q15_t*    _ib, q15_t*    _ic);
```

パラメーター：

VEx： VE アドレスを選択します。
_ia： a 相電流を格納する変数アドレスを設定します。
_ib： b 相電流を格納する変数アドレスを設定します。
_ic： c 相電流を格納する変数アドレスを設定します。

機能：

各相の電流値を取得します。
a 相には U 相電流値、b 相には V 相電流値、c 相には W 相電流値が入ります。

補足：

特になし

リターンパラメーター：

なし

12.1.1.4. VE_GetCurrentAdcData

電流 AD 値取得

API：

```
void
VE_GetCurrentAdcData(TSB_VE_TypeDef* const    VEx,
                     uint32_t*    _adc_ia, uint32_t*    _adc_ib, uint32_t*    _adc_ic);
```

パラメーター：

VEx： IP テーブルアドレスを選択します。
_adc_ia： a 相電流 AD 値を格納する変数アドレスを設定します。
_adc_ib： b 相電流 AD 値を格納する変数アドレスを設定します。
_adc_ic： c 相電流 AD 値を格納する変数アドレスを設定します。

機能：

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

補足：

特になし

リターンパラメーター：

なし

12.1.1.5. VE_GetdataFromVEreg

VE レジスターデータ取得

API：

```
void
VE_GetdataFromVEreg(TSB_VE_TypeDef* const    VEx,
                    vector_t* const    _motor)
```

パラメーター：

VEx： VE アドレスを選択します。
_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：

VE レジスターからモーター電圧 Vdc、d 軸電圧 Vd、q 軸電圧 Vq、d 軸電流 Id、q 軸電流 Iq の値をベクトル制御の各変数へ格納します。
Duty 幅により電流が検出できないタイミングでは、d 軸電流 Id、q 軸電流 Iq の値を VE レジスターへ前回検出値を書きこみます。

補足：

特になし

リターンパラメーター：

なし

12.1.1.6. **VE_GetPWM_Period**

電流 AD 値取得

API：
uint32_t
VE_GetPWM_Period(TSB_VE_TypeDef* const VEx)
パラメーター：
VEx：VE アドレスを選択します。

機能：
PWM 周期レジスターの値を取得します。
補足：
特になし
リターンパラメーター：
PWM 周期

12.1.1.7. **VE_GetPWM_DutyU**

U 相 Duty 値取得
API：
uint32_t
VE_GetPWM_DutyU(TSB_VE_TypeDef* const VEx)

パラメーター：
VEx：VE アドレスを選択します。

機能：
U 相 Duty レジスターの値を取得します。
補足：
特になし

リターンパラメーター：
U 相 Duty

12.1.1.8. **VE_GetPWM_DutyV**

V 相 Duty 値取得
API：
uint32_t
VE_GetPWM_DutyV(TSB_VE_TypeDef* const VEx)

パラメーター：
VEx：VE アドレスを選択します。
機能：
V 相 Duty レジスターの値を取得します。
補足：
特になし
リターンパラメーター：
V 相 Duty

12.1.1.9. **VE_GetPWM_DutyW**

W 相 Duty 値取得
API：
uint32_t
VE_GetPWM_DutyW(TSB_VE_TypeDef* const VEx)

パラメーター：
VEx：VE アドレスを選択します。
機能：
W 相 Duty レジスターの値を取得します。

補足：
特になし
リターンパラメーター：
W 相 Duty

12.1.1.10. VE_GetPWM_DutyMed

Duty 中間値取得
API：
uint32_t
VE_GetPWM_DutyMed(TSB_VE_TypeDef* const VEx)

パラメーター：
VEx：VE アドレスを選択します。
機能：
U、V、W 相 Duty 値の中間の値を取得します。
補足：
特になし
リターンパラメーター：
Duty 中間値

12.1.1.11. VE_GetPWM_Modulation

変調方式取得
API：
int
VE_GetPWM_Modulation(TSB_VE_TypeDef* const VEx)

パラメーター：
VEx：VE アドレスを選択します。

機能：
変調方式を取得します。
補足：
特になし
リターンパラメーター：
変調方式 0:3 相変調、1:2 相変調

12.1.1.12. VE_GetSector

現在セクター取得
API：
uint32_t VE_GetSector(const ipdrv_t* const _ipdrv)
パラメーター：
_ipdrv： IP テーブルアドレスを選択します。
機能：
現在のセクター値を取得します。
補足：
特になし
リターンパラメーター：
現在セクター値 [0-11]

12.1.1.13. VE_GetShiftPWMState

シフト PWM 状態取得
API：
int
VE_GetShiftPWMState(TSB_VE_TypeDef* const VEx)

パラメーター：
VEx：VE アドレスを選択します。

機能：
シフト PWM 状態を取得します。
補足：
特になし
リターンパラメーター：
シフト PWM 状態 0：通常 PWM、1：シフト PWM

12.1.1.14. VE_GetOutPutMode

PWM 出力状態取得
API：

```
int
VE_GetOutputMode(TSB_VE_TypeDef* const VEx)
```

パラメーター：
VEx：VE アドレスを選択します。

機能：
PWM 出力状態を取得。
補足：
特になし
リターンパラメーター：

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

12.1.1.15. VE_SetdataToVEreg_Stop

VE レジスターへのデータセット(Stop)
API：

```
void
VE_SetdataToVEreg_Stop(TSB_VE_TypeDef* const VEx,
                       const vector_t* const _motor);
```

パラメーター：
VEx： VE アドレスを選択します。
_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：
停止状態の VE レジスターへの設定処理を行います。
電流制御ゲイン積分値レジスターなどの初期化を行います。
補足：
特になし
リターンパラメーター：
なし

12.1.1.16. VE_SetdataToVEreg_Bootstrap

VE レジスターへのデータセット(Bootstrap)
API：

```
void
VE_SetdataToVEreg_ Bootstrap(TSB_VE_TypeDef* const VEx,
                             const vector_t* const _motor);
```

パラメーター：
VEx： VE アドレスを選択します。
_motor： ベクトル制御変数の構造体アドレスを設定します。

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

特になし

リターンパラメーター：

なし

12.1.1.17. VE_SetdataToVEreg_Initposition_i

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

API：

```
void
VE_SetdataToVEreg_ Initposition_i (TSB_VE_TypeDef* const VEx,
                                   const vector_t* const _motor);
```

パラメーター：

VEx： VE アドレスを選択します。

_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：

電流制御タイプで位置決め状態の VE レジスターへの設定処理を行います。

補足：

特になし

リターンパラメーター：

なし

12.1.1.18. VE_SetdataToVEreg_Initposition_v

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

API：

```
void
VE_SetdataToVEreg_ Initposition_v (TSB_VE_TypeDef* const VEx
                                   const vector_t* const _motor);
```

パラメーター：

VEx： VE アドレスを選択します。

_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：

電圧制御タイプで位置決め状態の VE レジスターへの設定処理を行います。

補足：

特になし

リターンパラメーター：

なし

12.1.1.19. VE_SetdataToVEreg_Force_i

VE レジスターへのデータセット(強制転流 電流制御タイプ)

API：

```
void
VE_SetdataToVEreg_Force_i (const ipdrv_t* const VEx,
                           const vector_t* const _motor)
```

パラメーター：

VEx： VE アドレスを選択します。

_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：

電流制御タイプで強制転流状態の VE レジスターへの設定処理を行います。

補足：

特になし

リターンパラメーター：

なし

12.1.1.20. VE_SetdataToVEreg_Force_v

VE レジスターへのデータセット(強制転流 電圧制御タイプ)

API：

```
void
VE_SetdataToVEreg_Force_v(TSB_VE_TypeDef* const VEx,
                          const vector_t* const _motor);
```

パラメーター：

VEx： VE アドレスを選択します。
_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：

電圧制御型の強制転流状態の VE レジスターへの設定処理を行います。

補足：

特になし

リターンパラメーター：

なし

12.1.1.21. VE_SetdataToVEreg_Change_up

VE レジスターへのデータセット(強制定常切り替え)

API：

```
void
VE_SetdataToVEreg_ Change_up (TSB_VE_TypeDef* const VEx,
                              const vector_t* const _motor);
```

パラメーター：

VEx： VE アドレスを選択します。
_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：

強制定常切り替え状態の VE レジスターへの設定処理を行います。

補足：

特になし

リターンパラメーター：

なし

12.1.1.22. VE_SetdataToVEreg_Steady_A

VE レジスターへのデータセット(定常)

API：

```
void
VE_SetdataToVEreg_Steady_A (TSB_VE_TypeDef* const VEx,
                            const vector_t* const _motor);
```

パラメーター：

VEx： VE アドレスを選択します。
_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：

定常状態の VE レジスターへの設定処理を行います。

補足：

特になし

リターンパラメーター：

なし

12.1.1.23. VE_SetdataToVEreg_Emergency

VE レジスターへのデータセット(EMG)

API：

```
void
VE_SetdataToVEreg_ Emergency (TSB_VE_TypeDef* const VEx,
                              const vector_t* const _motor);
```

パラメーター：

VEx： VE アドレスを選択します。
_motor： ベクトル制御変数の構造体アドレスを設定します。

機能：
Emergency 状態の VE レジスターへの設定処理を行います。
補足：
特になし
リターンパラメーター：
なし

12.1.1.24. VE_SetZeroCurrentData

ゼロ電流設定
API：

```
void
VE_SetZeroCurrentData(TSB_VE_TypeDef* const    VEx,
                      uint32_t    _z_ia, uint32_t    _z_ib, uint32_t    _z_ic)
```

パラメーター：
VEx： VE アドレスを選択します。
_z_ia： a 相ゼロ電流 AD 値を設定します。
_z_ib： b 相ゼロ電流 AD 値を設定します。
_z_ic： c 相ゼロ電流 AD 値を設定します。
機能：
ゼロ電流時の AD 値を VE レジスターに設定します。
補足：
特になし
リターンパラメーター：
なし

12.1.1.25. VE_SetVDCreg

モーター電圧と Vdc 設定
API：

```
void VE_SetVDCreg(TSB_VE_TypeDef* const VEx, q15_t _dat);
```

パラメーター：
VEx： VE アドレスを選択します。
_dat： モーター電圧を設定します。
機能：
モーター電圧を VE レジスターに設定します。
補足：
特になし
リターンパラメーター：
なし

12.1.1.26. VE_SetSpwmMode

シフト PWM モード設定
API：

```
void VE_SetSpwmMode(TSB_VE_TypeDef* const VEx, uint8_t _dat);
```

パラメーター：
VEx： VE アドレスを選択します。
_dat： シフト PWM モードを設定します。
機能：
シフト PWM モード状態を VE レジスターに設定します。
補足：
特になし
リターンパラメーター：
なし

12.1.1.27. **VE_SetModulType**

変調方式設定

API：

void VE_SetModulType (TSB_VE_TypeDef* const VEx, uint8_t _dat);

パラメーター：

VEx： VE アドレスを選択します。

_dat： 変調方式を設定します。

機能：

変調方式を VE レジスターに設定します。

補足：

特になし

リターンパラメーター：

なし

12.1.2. データ構造

12.1.2.1. VE_InitTypeDef

Data Fields :

uint8_t **ve_ch** : VE チャネル
cCH0 : VE チャネル 0
cCH1 : VE チャネル 1

uint8_t **shunt** : シャントタイプ
1 : 1 シャント
3 : 3 シャント

uint16_t **pwmfreq** : PWM 周波数(PWM 出力用)
VEMDPRD レジスターに設定する値

uint16_t **reptime** : リピート回数(1～15 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
TRGMODE_UNITB : ADCB PMD1 トリガー同期変換終了割り込みで起動

uint16_t **tpwm** : PWM 周期時間(位相補間用)
VETPWM レジスターに設定する値

uint16_t **idkp** : d 軸電流制御比例ゲイン

uint16_t **idki** : d 軸電流制御積分ゲイン

uint16_t **iqkp** : q 軸電流制御比例ゲイン

uint16_t **iqki** : q 軸電流制御積分ゲイン

uint16_t **zerooffset** : ゼロ電流オフセット値

12.2. モーター制御回路(PMD)

12.2.1. 関数仕様

12.2.1.1. IP_PMD_init

PMD 初期設定

API：

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

パラメーター：

PMDx：　PMD アドレスを選択します。
_initdata：　PMD 初期設定データ構造体
詳細は 12.2.2.1　PMD_InitTypeDef を参照してください。

機能：

PMD の初期設定を行います

補足：

PMD 停止、割り込み禁止で呼んでください。

リターンパラメーター：

なし

12.2.1.2. PMD_GetEMG_Status

EMG 保護状態取得

API：

emg_status_e
PMD_GetEMG_Status(TSB_PMD_TypeDef* const PMDx)

パラメーター：

PMDx：PMD アドレスを選択します。

機能：

EMG 保護状態を取得します。

補足：

特になし

リターンパラメーター：

emg_status_e：　EMG 保護状態
cNormal：　正常
cEMGProtected：　保護中

12.2.1.3. PMD_ReleaseEMG_Protection

EMG 保護状態解除

API：

void
PMD_ReleaseEMG_Protection(TSB_PMD_TypeDef* const　PMDx)

パラメーター：

PMDx：PMD アドレスを選択します。

機能：

EMG 保護状態を解除します。

補足：

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

リターンパラメーター：

なし

12.2.2. データ構造

12.2.2.1. PMD_InitTypeDef

Data Fields :

uint8_t **shunt** : シャントタイプ
3 : 3 シャント
1 : 1 シャント

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

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

uint16_t **pwmfreq** : PWM 周波数
PMDxMDPDR に設定する値

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

12.3. アナログデジタルコンバーター(ADC)

12.3.1. 関数仕様

12.3.1.1. IP_ADC_init

ADC 初期設定

API：

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

パラメーター：

ADx： ADC アドレスを選択します。
_initdata： ADC 初期設定データ構造体
詳細は 12.3.2.1　AD_InitTypeDef を参照してください。

機能：

ADC の初期設定を行います

補足：

ADC 停止、割り込み禁止で呼んでください。

リターンパラメーター：

なし

12.3.1.2. IP_ADC_init2Unit

ADC 初期設定

API：

void
IP_ADC_init2Unit(AD_InitTypeDef* const _initdata)

パラメーター：

ADx： ADC アドレスを選択します。
_initdata： ADC 初期設定データ構造体
詳細は 12.3.2.1　AD_InitTypeDef を参照してください。

機能：

ADC の初期設定を行います

補足：

ADC 停止、割り込み禁止で呼んでください。

リターンパラメーター：

なし

12.3.2. データ構造

12.3.2.1. 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 チャネル番号(3 シャント用)

uint8_t **ivch** : V 相電流取り込み AD チャネル番号(3 シャント用)

uint8_t **iwch** : W 相電流取り込み AD チャネル番号(3 シャント用)

uint8_t **idcch** : DC 電流取り込み AD チャネル番号(1 シャント用)

uint8_t **vdcch** : モーター電圧 Vdc 取り込み AD チャネル番号

uint8_t **pmd_ch** : 使用する PMD チャネル選択
cPMD0 : PMD チャネル 0
cPMD1 : PMD チャネル 1

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

12.4. 定数説明

12.4.1. VE+搭載マイコン用定数(mcuip_drv.h)

12.4.1.1. **PMD**

表 12.1 PMD 定数

MDCR レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cWPWMES	W-phase edge setting	PWMES_CENTER	Edge unfixed. (center-aligned PWM)
		PWMES_RISING	PWM rising-edge fixed
		PWMES_FALLING	PWM falling-edge fixed
cVPWMES	V-phase edge setting	PWMES_CENTER	Edge unfixed. (center-aligned PWM)
		PWMES_RISING	PWM rising-edge fixed
		PWMES_FALLING	PWM falling-edge fixed
cUPWMES	U-phase edge setting	PWMES_CENTER	Edge unfixed. (center-aligned PWM)
		PWMES_RISING	PWM rising-edge fixed
		PWMES_FALLING	PWM falling-edge fixed
cDSYNCS	Double buffer update timing for the duty compare register and PWM period register.	DSYNCS_INTCYC	Depends on interrupt cycle setting
		DSYNCS_BOTTOM	Updates at PWM carrier bottom
		DSYNCS_TOP	Updates at PWM carrier peak
		DSYNCS_BOTTOM_TOP	Updates at both PWM carrier peak and bottom
cDTCREN	Dead time correction	DTCREN_DISABLE	Correction disable
		DTCREN_ENABLE	Correction enable
cPWMEXCLK	PWM period extension mode	PWMCK_NORMAL	Normal period
		PWMCK_4FOLD	4x period
cPWMSYNT	Port output mode	SYNTMD_0	—
		SYNTMD_1	—
cPWMSPCFY	Duty mode	DTYMD_COMMON	3-phase common mode
		DTYMD_INDEPENDENT	3-phase independent mode
cPWMINT	PWM interrupt timing	PINT_BOTTOM	Interrupt request when PWM counter PMD1MDCNT<MDCNT[15:0]> = 0x0001
		PINT_TOP	Interrupt request when PWM counter PMD1MDCNT<MDCNT[15:0]> = <MDPRD[15:0]>
cPWMINTPRD	PWM interrupt period	INTPRD_HALF	Interrupt request at every 0.5 PWM period
		INTPRD_1	Interrupt request at every PWM period
		INTPRD_2	Interrupt request at every 2 PWM periods
		INTPRD_4	Interrupt request at every 4 PWM periods
cPWMWAVE	PWM carrier waveform	PWMMD_SAWTOOTH	Edge-aligned PWM Sawtooth wave
		PWMMD_TRIANGULAR	Center-aligned PWM Triangular wave

MDPOT レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cSYNCS	MDOUT transfer timing for trigger synchronous.	SYNCS_ASYNC	Asynchronous
		SYNCS_INTENC	when INTENCx (ENCx interrupt request) occurs
		SYNCS_NTTB	when INTTBx0 (TMRBx interrupt request) occurs
cPSYNCS	MDOUT transfer timing for PWM synchronous.	PSYNCS_WRITE	Reflect when write
		PSYNCS_BOTTOM	Reflect when PWM counter MDCNT="1"
		PSYNCS_TOP	Reflect when PWM counter MDCNT=PMDxMDPRD<MDPRD>
		PSYNCS_BOTTOM_TOP	Reflect when PWM counter MDCNT="1" or PMDxMDPRD<MDPRD>

PORTMD レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cPORTMD	Port control settings at tool break	PORTMD_ALLHIZ	Upper phases = High-z / lower phases = High-z
		PORTMD_UHIZ_LON	Upper phases = High-z / lower phases = PMD output
		PORTMD_UON_LHIZ	Upper phases = PMD output / lower phases = High-z
		PORTMD_ALLON	Upper phases = PMD output / lower phases = PMD output

TRGCR レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cTRGMD_3SHT	Trigger timing setting for 3shunt	TRGMD_DIS	Trigger output disabled
		TRGMD_DOWN	Trigger output at down-count match
		TRGMD_UP	Trigger output at up-count match
		TRGMD_UPDOWM	Trigger output at up-/down-count match
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak and bottom
cTRGMD_1SHT	Trigger timing setting for 1shunt	TRGMD_DIS	Trigger output disabled
		TRGMD_DOWN	Trigger output at down-count match
		TRGMD_UP	Trigger output at up-count match
		TRGMD_UPDOWM	Trigger output at up-/down-count match
		TRGMD_PEAK	Trigger output at PWM carrier peak
		TRGMD_BOTTOM	Trigger output at PWM carrier bottom
		TRGMD_PEAKBOTTOM	Trigger output at PWM carrier peak and bottom
cBUFSYNC	Triger Buffer update timing	TRGBE_SYNC	Sync
		TRGBE_ASYNC	Async The value written to PMDTRGx is

			immediately reflected.)

TRGMD レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cEMGTGE	Output enable in EMG protection state	EMGTGE_DISABLE	Disable trigger output in the protection state
		EMGTGE_ENABLE	Enable trigger output in the protection state

EMGCR レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cEMGCNT	EMG input detection time	0 to 15	The noise remove time value. cEMGCNT × 16/fsys. (resolution:200[ns] at 80MHz) cEMGCNT ≧ 0 to 15. When cEMGCNT=0, the noise filter is bypassed.
cINHEN	Tool break enable setting	INHEN_DISABLE	Tool break disable
		INHEN_ENABLE	Tool break enable
cEMGMD	EMG protection mode select	EMGMD_ALLHIZ	All phases High-Z
		EMGMD_UON_LHIZ	Lower phases High-Z
		EMGMD_UHIZ_LON	Upper phases High-Z
		EMGMD_ALLHIZ2	All phases High-Z

OVVCR レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cOVVCNT	OVV input detection time	0 to 15	The noise remove time value. cEMGCNT × 16/fsys. (resolution:200[ns] at 80MHz) cEMGCNT ≧ 0 to 15. When cEMGCNT=0, the noise filter is bypassed.
cADIN1EN	ADC B monitor interrupt input enable	ADIN1EN_DISABLE	Disable input
		ADIN1EN_ENABLE	Enable input
cADIN0EN	ADC A monitor interrupt input enable	ADIN0EN_DISABLE	Disable input
		ADIN0EN_ENABLE	Enable input
cOVVMD	OVV protection mode	OVVMD_NOCON	No output control
		OVVMD_UON_LOFF	All upper phases ON, all lower phases OFF
		OVVMD_UOFF_LON	All upper phases OFF, all lower phases ON
		OVVMD_ALLOFF	All phases OFF
cOVVISEL	OVV input select	OVVISEL_PORT	Port input
		OVVISEL_ADC	ADC monitor signal
cOVVRS	OVV protection release	OVVRS_NORMAL	Disable automatic release of OVV protection
		OVVRS_AUTO	Enable automatic release of OVV protection

12.4.1.2. **VE**

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

表 12.2　VE 定数

FMODE レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cSPWMMD	Selects a PWM shift mode	SPWMMD_SPWM1	Shift 1
		SPWMMD_SPWM2U	Shift 2 (U-phase Normal PWM)
		SPWMMD_SPWM2V	Shift 2 (V-phase Normal PWM)
		SPWMMD_SPWM2W	Shift 2 (W-phase Normal PWM)
cPHCVDIS	Phase transformation	PHCVDIS_ENABLE	2-3 phase transformation is enabled
		PHCVDIS_DISABLE	2-3 phase transformation is disabled
cMREGDIS	Keep the previous value of SIN/COS/SECTOR	MREGDIS_EFFECTIVE	Effective
		MREGDIS_NOEFFECTIVE	No effective
cCRCEN	Trigger correction	CRCEN_DISABLE	Disable trigger correction.
		CRCEN_ENABLE	Enable trigger correction.
cIDPLMD	Current polarity detection	IDPLMD_SHUNT	shunt
		IDPLMD_SENSOR	sensor

MODE レジスター設定用			
定数名	**定数の意味**	**選択データ**	**選択データの意味**
cT7SQRTEN	Task7 enable.	T7SQRTEN_DISABLE	Disable
		T7SQRTEN_ENABLE	Enable
cT4ATANEN	Task4 enable.	T4ATANEN_DISABLE	Disable
		T4ATANEN_ENABLE	Enable
cVDCSEL	Selects the supply voltage store register	VDCSEL_VDC	Stored in VExVDC.
		VDCSEL_VDCL	Stored in VExVDCL.
cZIEN	Zero current detection	ZIEN_DISABLE	Disable
		ZIEN_ENABLE	Enable
cPVIEN	Phase interpolation	PVIEN_DISABLE	Disable
		PVIEN_ENABLE	Enable

13. 付録

13.1. 固定小数点処理

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

13.2. 正規化(Normalize)

正規化とはデータを一定のルールに従って変形しデータを利用しやすくすることです。
本アプリケーションノートでは最上位を符号ビット(0 は正、1 は負)とし、次のビットとの間に小数点を置き、またそのデータがなりうる最大の値(0x7fff)を 1、最小の値(0x8000)を-1 となるように正規化を行います。

例)

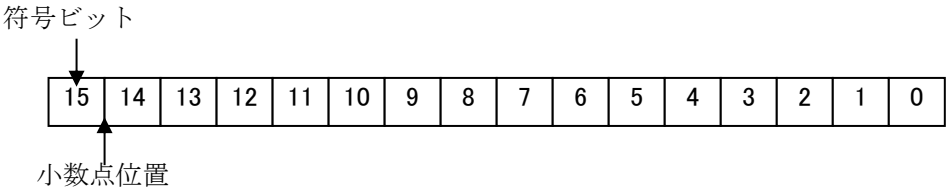


図 13.1　16 ビットデータの例

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

13.3. データフォーマット

本アプリケーションのモーター制御部では固定小数点演算を行っています。
固定小数点演算では小数部分のビット数を Q 表記(Q フォーマット)で表します。
基本的には 16 ビットデータの場合は Q15 フォーマット(小数部分 15 ビット)、32 ビットデータの場合は Q31 フォーマット(小数部分 31 ビット)で演算を行っています。
小数フォーマットで表すことのできる値はフォーマットにより異なります。

表 13.1　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 13.2　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

13.4. 固定小数点での演算

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

(1)乗算

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

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

(2)除算

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

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

13.5. 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あります。

(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
```

14. 改訂履歴

日付	Rev.	変更内容
2019/08/22	1.0	初版
2026/06/29	1.1	1.1 評価ボードを変更 1.2 開発環境を追加

製品取り扱い上のお願い

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

- 本製品に関する情報等、本資料の掲載内容は、技術の進歩などにより予告なしに変更されることがあります。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。また、文書による当社の事前の承諾を得て本資料を転載複製する場合でも、記載内容に一切変更を加えたり、削除したりしないでください。
- 当社は品質、信頼性の向上に努めていますが、半導体・ストレージ製品は一般に誤作動または故障する場合があります。本製品をご使用頂く場合は、本製品の誤作動や故障により生命・身体・財産が侵害されることのないように、お客様の責任において、お客様のハードウエア・ソフトウエア・システムに必要な安全設計を行うことをお願いします。なお、設計および使用に際しては、本製品に関する最新の情報（本資料、仕様書、データシート、アプリケーションノート、半導体信頼性ハンドブックなど）および本製品が使用される機器の取扱説明書、操作説明書などをご確認の上、これに従ってください。また、上記資料などに記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を使用する場合は、お客様の製品単独およびシステム全体で十分に評価し、お客様の責任において適用可否を判断してください。
- 本製品は、特別に高い品質・信頼性が要求され、またはその故障や誤作動が生命・身体に危害を及ぼす恐れ、膨大な財産損害を引き起こす恐れ、もしくは社会に深刻な影響を及ぼす恐れのある機器（以下“特定用途”という）に使用されることは意図されていませんし、保証もされていません。特定用途には原子力関連機器、航空・宇宙機器、医療機器（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。
- 本製品を分解、解析、リバースエンジニアリング、改造、改変、翻案、複製等しないでください。
- 本製品を、国内外の法令、規則及び命令により、製造、使用、販売を禁止されている製品に使用することはできません。
- 本資料に掲載してある技術情報は、製品の代表的動作・応用を説明するためのもので、その使用に際して当社及び第三者の知的財産権その他の権利に対する保証または実施権の許諾を行うものではありません。
- 別途、書面による契約またはお客様と当社が合意した仕様書がない限り、当社は、本製品および技術情報に関して、明示的にも黙示的にも一切の保証（機能動作の保証、商品性の保証、特定目的への合致の保証、情報の正確性の保証、第三者の権利の非侵害保証を含むがこれに限らない。）をしておりません。
- 本製品、または本資料に掲載されている技術情報を、大量破壊兵器の開発等の目的、軍事利用の目的、あるいはその他軍事用途の目的で使用しないでください。また、輸出に際しては、「外国為替及び外国貿易法」、「米国輸出管理規則」等、適用ある輸出関連法令を遵守し、それらの定めるところにより必要な手続を行ってください。
- 本製品の RoHS 適合性など、詳細につきましては製品個別に必ず当社営業窓口までお問い合わせください。本製品のご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用ある環境関連法令を十分調査の上、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は一切の責任を負いかねます。