RL78/F24

R01AN6310JJ0110
Rev.1.10
2023.06.30

MCU による PMSM モータのセンサレスベクトル制御編（単一シャント）

要旨

本アプリケーションノートは RL78/F24 の機能を使って永久磁石同期モータ（以降 PMSM モータと称す）をセンサレスベクトル制御で駆動するサンプルプログラムについて説明することを目的としています。

サンプルプログラムはあくまで参考用途であり、弊社がこの動作を保証するものではありません。サンプルプログラムを使用する場合、適切な環境で十分な評価を行ったうえで使用下さい。

動作確認デバイス

サンプルプログラムの動作確認は下記のデバイスで行っております。

・RL78/F24 (R7F124FGJ)

目次

1. 概説

本アプリケーションノートは、RL78/F24 を使用し、PMSM モータ（Permanent Magnet Synchronous Motor）のセンサレスベクトル制御のサンプルプログラムについて説明するものです。

1.1 システムの利用

本システム（サンプルプログラム）は、RL78/F24 マイコン搭載ボード（RTK7F124FGS00000BJ）および、BLDC モータ（TG-55N-KA）を使用し、センサレスベクトル制御を実現しています。

備考：

- TG-55N-KA モータは、ツカサ電工株式会社の製品です。
  ツカサ電工株式会社 (https://www.tsukasa-d.co.jp/)

1.2 開発環境

本アプリケーションノートが対象とするサンプルプログラムの開発環境を表 1-1、表 1-2 に示します。

表 1-1 ソフトウェア開発環境

Renesas CS+		
	IDE Version	CS+ for CC E8.07.00 [01 Dec 2021]
	Compiler	CC-RL E1.11.00
IAR		
	IDE Version	IAR Embedded Workbench IDE 8.5.2.7561 (8.5.2.7561)
	Compiler	IAR C/C++ Compiler for Renesas RL78 4.21.3.2447 (4.21.3.2447)

表 1-2 ハードウェア開発環境

On-chip Debugging Emulator	E2 emulator Lite
	E2 emulator
MCU Part Name	RL78/F24 (R7F124FGJ)
Inverter Board	RTK7F124FGS00000BJ
BLDC Motor	PMSM (TG-55N-KA)

2.　システム概要

2.1　ハードウェア構成

2.1.1　ハードウェア構成図

ハードウェアの構成図を図 2-1 に示します。

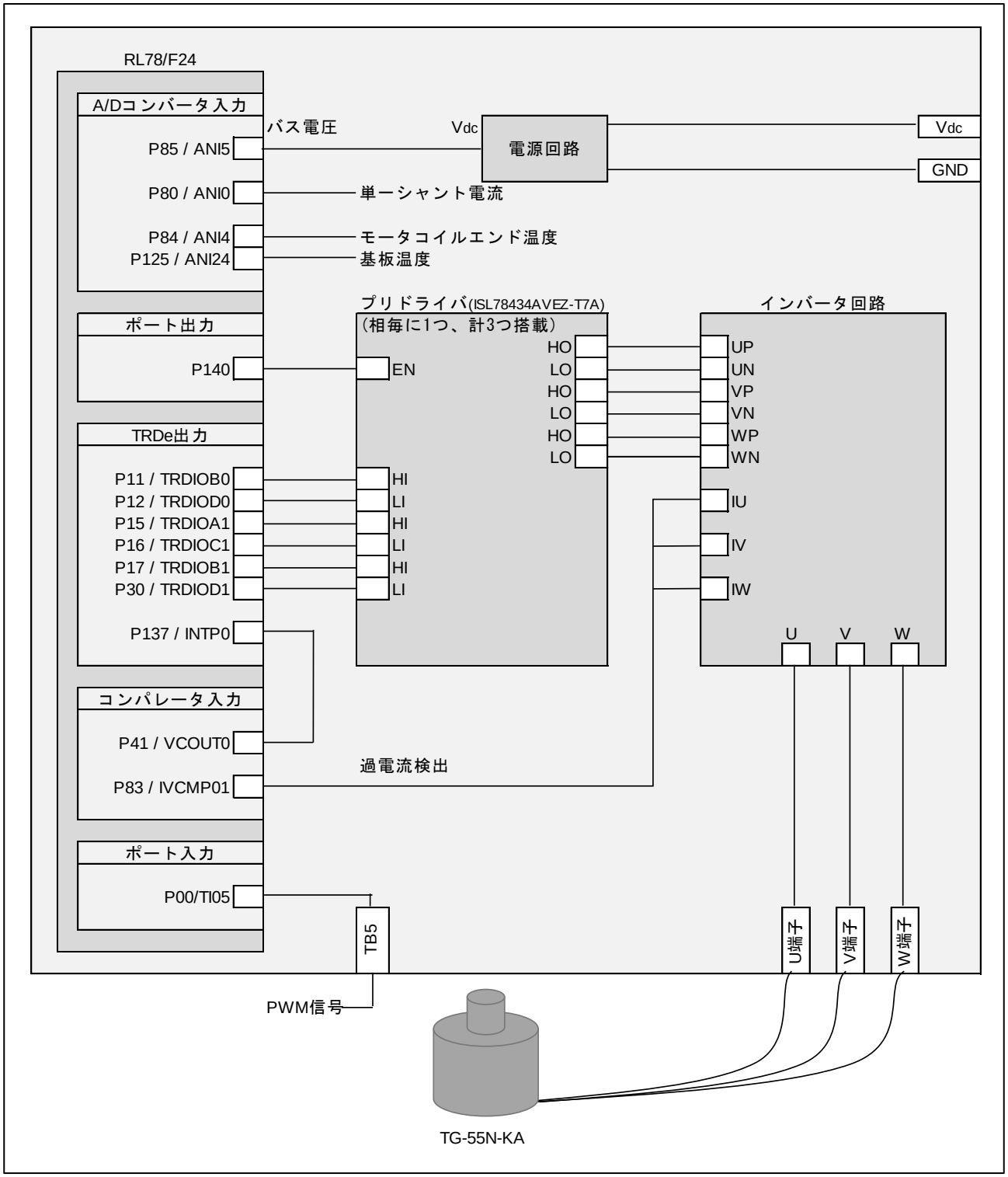


図 2-1 ハードウェア構成

2.1.2　ハードウェアの加工

本システムのインバータボード（RTK7F124FGS00000BJ）は 3 シャント電流検出用回路として提供しています。単一シャント電流検出に対応するためには以下に示すハードウェアの加工を行なう必要があります。

- インバータボード上のシャント抵抗 R55、R56 を取り外す
- R55、R56 部の “1 – 3” 間を 0Ω 抵抗で結線する

上記にて、シャント抵抗 R57 を使用した単一シャント電流を検出します。

2.2　ハードウェア仕様

2.2.1　端子インタフェース

本システムの RL78/F24 端子インタフェース一覧を表 2-1 に示します。

表 2-1 端子インタフェース一覧

端子名	機能
P85 / ANI5	Vdc 電圧測定
P80 / ANI0	単一シャント電流測定
P84 / ANI4	モータコイルエンド温度測定
P125 / ANI24	基板温度測定
P140	ゲートドライバ有効化 / 無効化信号出力
P11 / TRDIOB0	モータ制御出力 U 相（正相）
P12 / TRDIOD0	モータ制御出力 U 相（逆相）
P15 / TRDIOA1	モータ制御出力 V 相（正相）
P16 / TRDIOC1	モータ制御出力 V 相（逆相）
P17 / TRDIOB1	モータ制御出力 W 相（正相）
P30 / TRDIOD1	モータ制御出力 W 相（逆相）
P137 / INTP0	過電流検出信号入力
P41 / VCOUT0	過電流検出用コンパレータ出力
P83 / IVCMP01	過電流検出用コンパレータ入力
P00 / TI05	PWM デューティ指令値入力

2.2.2　周辺機能

本システムで使用する周辺機能の一覧を表 2-2 に示します。

表 2-2 周辺機能一覧

周辺機能	用途
A/D コンバータ	• 電圧測定（Vdc 電源電圧取得） • モータ電流測定（単一シャント電流取得） • モータコイルエンド温度測定 • 基板温度測定
タイマ・アレイ・ユニット	• 1ms インターバル・タイマ • PWM 信号指令値入力
タイマ RDe	• 拡張相補 PWM モードを使用した PWM 出力（正相 3 本、逆相 3 本） • PWM 出力強制遮断（PWMOPA を使用）
D/A コンバータ	内蔵コンパレータのしきい値出力
コンパレータ	過電流検出用
DTC	A/D 変換結果転送
AAU 注	• クラーク／パーク変換＋PI 制御 • 逆パーク／クラーク変換

【注】AAU：アプリケーション・アクセラレータ・ユニット（以降、AAU と称す）

2.2.2.1　A/D コンバータ

A/D コンバータは、アナログ入力をデジタル値に変換します。対象マイコン（RL78/F24 48 ピン製品）では、12 ビットの A/D コンバータを 1 回路搭載しています。変換チャネルを制御することで最大 19 チャネルのアナログ入力をデジタル値に変換できます。

本システムでは、A/D コンバータを表 2-3、表 2-4 のように使用しています。

表 2-3 A/D コンバータの使用内容（INTTM01 同期取得）

チャネル	項目	A/D 変換値の 1 ビットあたりの物理量
ANI4	モータコイルエンド温度測定	表 2-6 を参照
ANI24	基板温度測定	表 2-7 を参照

備考　INTTM01: タイマ・アレイ・ユニット 01 の割り込み要求

表 2-4 A/D コンバータの使用内容（INTAD）

チャネル	項目	A/D 変換値の 1 ビットあたりの物理量
ANI5	V_{dc} 電源電圧測定	65.0 [V] / 4095 = 0.0159 [V]
ANI0	単一シャント電流測定	50.0 [A] / 4095 = 0.0122 [A]

備考　INTAD: A/D 変換終了割り込み要求

表 2-5 A/D 変換対象一覧

取得対象	変数名	チャネル
V_{dc} 電源電圧	–	ANI5
相電流	focCurrentIu, focCurrentIv, focCurrentIw	ANI0
モータコイルエンド温度	userThCoilEnd	ANI4
基板温度	userThOnBoard	ANI24

備考　focCurrentIu（U 相電流）、focCurrentIv（V 相電流）、focCurrentIw（W 相電流）は ANI0 の取得値と位相パターンにより求めます。

温度測定においては、予めサーミスタと回路抵抗から算出されたテーブルを基に温度の変換を行います。以下にモータコイルエンドと基板の電圧変換テーブルを示します。

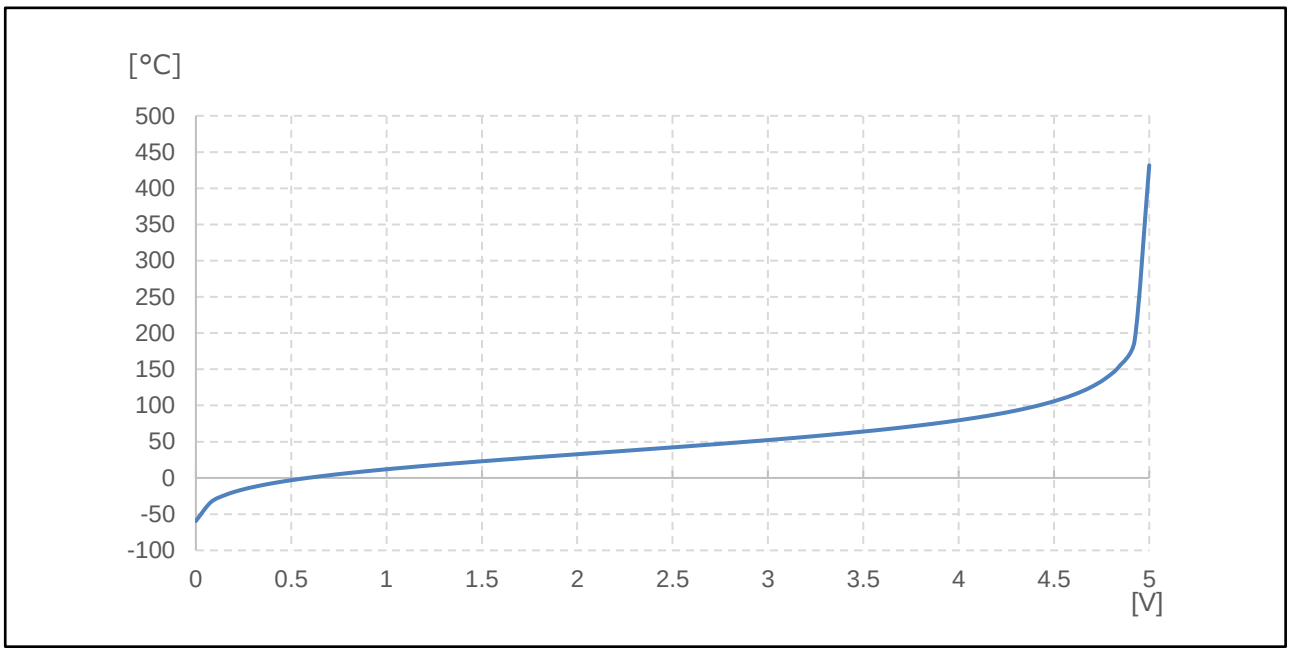


図 2-2 コイルエンド温度測定 A/D 端子電圧対温度グラフ

表 2-6 コイルエンド温度測定 A/D 端子電圧対温度表

電圧［V］	温度［°C］	電圧［V］	温度［°C］	電圧［V］	温度［°C］	電圧［V］	温度［°C］
0.000	-59.393	1.328	19.469	2.657	45.264	3.985	79.031
0.078	-33.636	1.407	21.111	2.735	46.818	4.063	82.049
0.156	-23.353	1.485	22.716	2.813	48.396	4.142	85.326
0.234	-16.808	1.563	24.289	2.891	50.002	4.220	88.917
0.313	-11.869	1.641	25.836	2.969	51.641	4.298	92.896
0.391	-7.838	1.719	27.362	3.048	53.317	4.376	97.367
0.469	-4.391	1.797	28.870	3.126	55.035	4.454	102.478
0.547	-1.354	1.875	30.364	3.204	56.801	4.532	108.450
0.625	1.381	1.954	31.849	3.282	58.620	4.611	115.640
0.703	3.884	2.032	33.326	3.360	60.501	4.689	124.664
0.781	6.205	2.110	34.799	3.438	62.450	4.767	136.738
0.860	8.378	2.188	36.272	3.516	64.477	4.845	154.778
0.938	10.430	2.266	37.748	3.595	66.592	4.923	189.159
1.016	12.382	2.344	39.228	3.673	68.808	5.000	431.619
1.094	14.250	2.422	40.717	3.751	71.141		
1.172	16.047	2.501	42.217	3.829	73.607		
1.250	17.783	2.579	43.732	3.907	76.228		

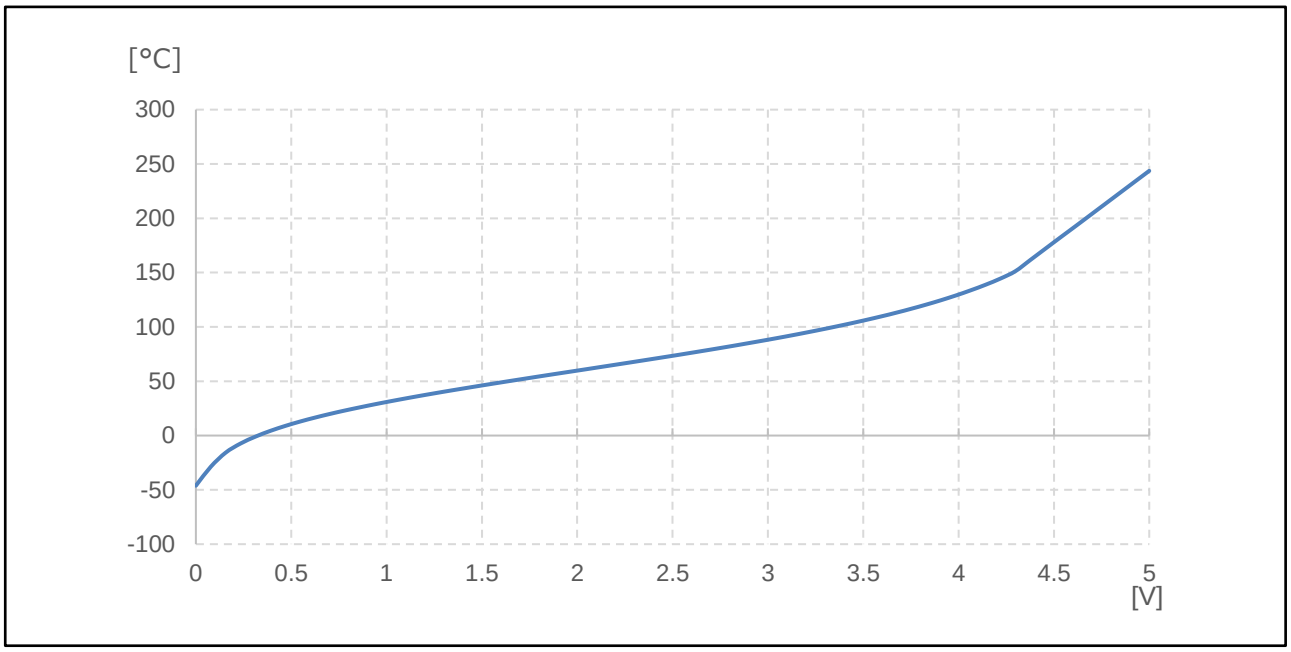


図 2-3 基板温度測定 A/D 端子電圧対温度グラフ

表 2-7 基板温度測定 A/D 端子電圧対温度表

電圧［V］	温度［°C］	電圧［V］	温度［°C］	電圧［V］	温度［°C］	電圧［V］	温度［°C］
0.000	0.000	1.328	41.174	2.657	77.864	3.985	128.909
0.078	-28.744	1.407	43.453	2.735	80.135	4.063	133.649
0.156	-15.757	1.485	45.688	2.813	82.450	4.142	138.830
0.234	-7.357	1.563	47.887	2.891	84.814	4.220	144.548
0.313	-0.950	1.641	50.055	2.969	87.233	4.298	151.285
0.391	4.326	1.719	52.200	3.048	89.716	4.376	161.566
0.469	8.869	1.797	54.326	3.126	92.270	4.454	171.848
0.547	12.898	1.875	56.440	3.204	94.905	4.532	182.129
0.625	16.546	1.954	58.545	3.282	97.629	4.611	192.410
0.703	19.903	2.032	60.647	3.360	100.456	4.689	202.691
0.781	23.029	2.110	62.750	3.438	103.397	4.767	212.973
0.860	25.969	2.188	64.858	3.516	106.468	4.845	223.254
0.938	28.758	2.266	66.975	3.595	109.688	4.923	233.535
1.016	31.420	2.344	69.107	3.673	113.076	5.000	243.656
1.094	33.978	2.422	71.257	3.751	116.659		
1.172	36.447	2.501	73.430	3.829	120.465		
1.250	38.842	2.579	75.631	3.907	124.533		

2.2.2.2　タイマ・アレイ・ユニット

タイマ・アレイ・ユニット TAU は、8 個の 16 ビット・タイマを搭載しています。各 16 ビット・タイマは「チャネル」と呼び、それぞれを単独のタイマとして使用することはもちろん、複数のチャネルを組み合わせて高度なタイマ機能として使用することもできます。対象マイコン（RL78/F24）では、2 ユニット（計 16 チャネル）を搭載しています。

本システムでは、TAU を表 2-8 のように使用しています。

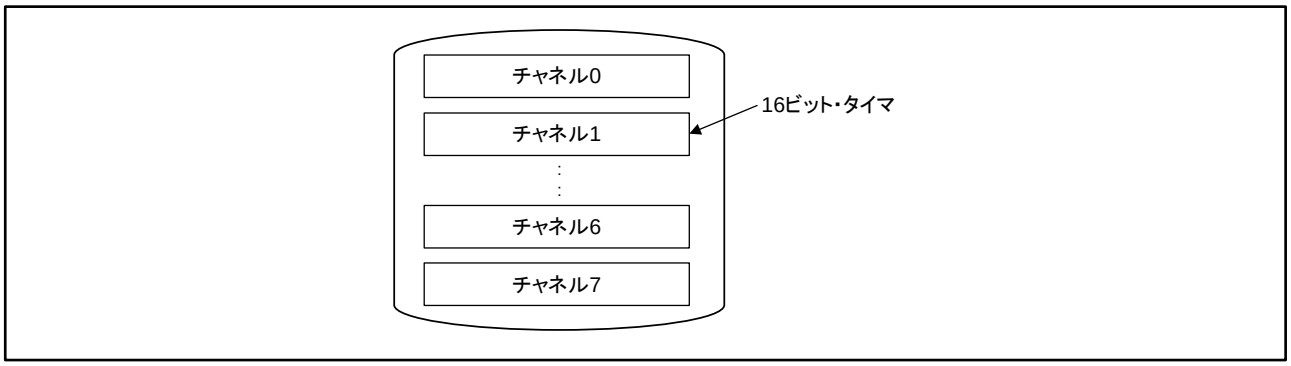


図 2-4 タイマ・アレイ・ユニット

表 2-8 タイマ・アレイ・ユニットの使用内容

Unit	CH	項目	内容	用途
0	1	タイマの動作モード	インターバル・タイマ機能	1 [ms]生成用タイマ
		ソース・クロック	CK00	
		カウント・クロック周波数	40 [MHz]	
		割り込み周期	1 [ms]	
		タイマ・データ・レジスタ 1（TDR01）設定値	39999 (1 [ms] / (1/40 [MHz]) - 1)	
	5	タイマの動作モード	キャプチャ機能	PWM 周期測定用タイマ
		ソース・クロック	CK01	
		カウント・クロック周波数	625 [kHz]	
		割り込み周期	両エッジ検出毎	
		タイマ・データ・レジスタ 5（TDR05）設定値	–	
		備考	使用内容を 4.1.1.2 (1)に記載	
	6	タイマの動作モード	ワンカウント機能	PWM 周期測定タイムアウト生成用タイマ
		ソース・クロック	CK01	
		カウント・クロック周波数	625 [kHz]	
		割り込み周期	100 [ms]	
		タイマ・データ・レジスタ 6（TDR06）設定値	62499 (100 [ms] / (1/625 [kHz]) - 1)	
		備考	使用内容を 4.1.1.2 (2)に記載	

2.2.2.3　タイマ RDe

タイマ RDe は、16 ビットタイマを 2 本（タイマ RD0、タイマ RD1）持ちます。また、タイマ RDe には、以下の 6 つのモードがあります。

- タイマモード
- リセット同期 PWM モード
- 相補 PWM モード
- PWM3 モード
- 拡張 PWM モード
- 拡張相補 PWM モード

本システムでは、タイマ RDe を表 2-9 のように使用しています。

表 2-9 タイマ RDe の使用内容

使用タイマ	項目	内容	用途
タイマ RDe	使用モード	拡張相補 PWM モード（非対称波形出力）	三相相補 PWM 出力
	PWM 周期	50 [μs]	
	短絡防止時間（Dead Time）	1.0 [μs]	
	カウント周波数	80 [MHz]	
	出力レベル	初期出力 “Low”、アクティブ・レベル “High”	
	バッファ動作	あり	
	パルス出力強制遮断制御	有効（PWM オプション・ユニット A を使用）（遮断時の出力値：ハイインピーダンス出力）	
	A/D トリガ	有効（ダウンカウント時にコンペアマッチ）	

注意：拡張相補 PWM モードは、タイマ RD0 とタイマ RD1 のカウンタやレジスタを組み合わせて三相相補 PWM 波形を出力します。

単一シャント電流検出では、相間電圧が小さい場合や、電圧のクロスポイントで電流測定期間が短くなる場合があります。この場合、電流期間を確保するために PWM 波形出力の変調を行います。本サンプルソフトウェアでは、単一シャント電流検出用にタイマ RDe の拡張相補 PWM モード（非対称波形出力）を使用し実現しています。拡張相補 PWM モード（非対称波形出力）の出力波形例を図 2-5 に示します。

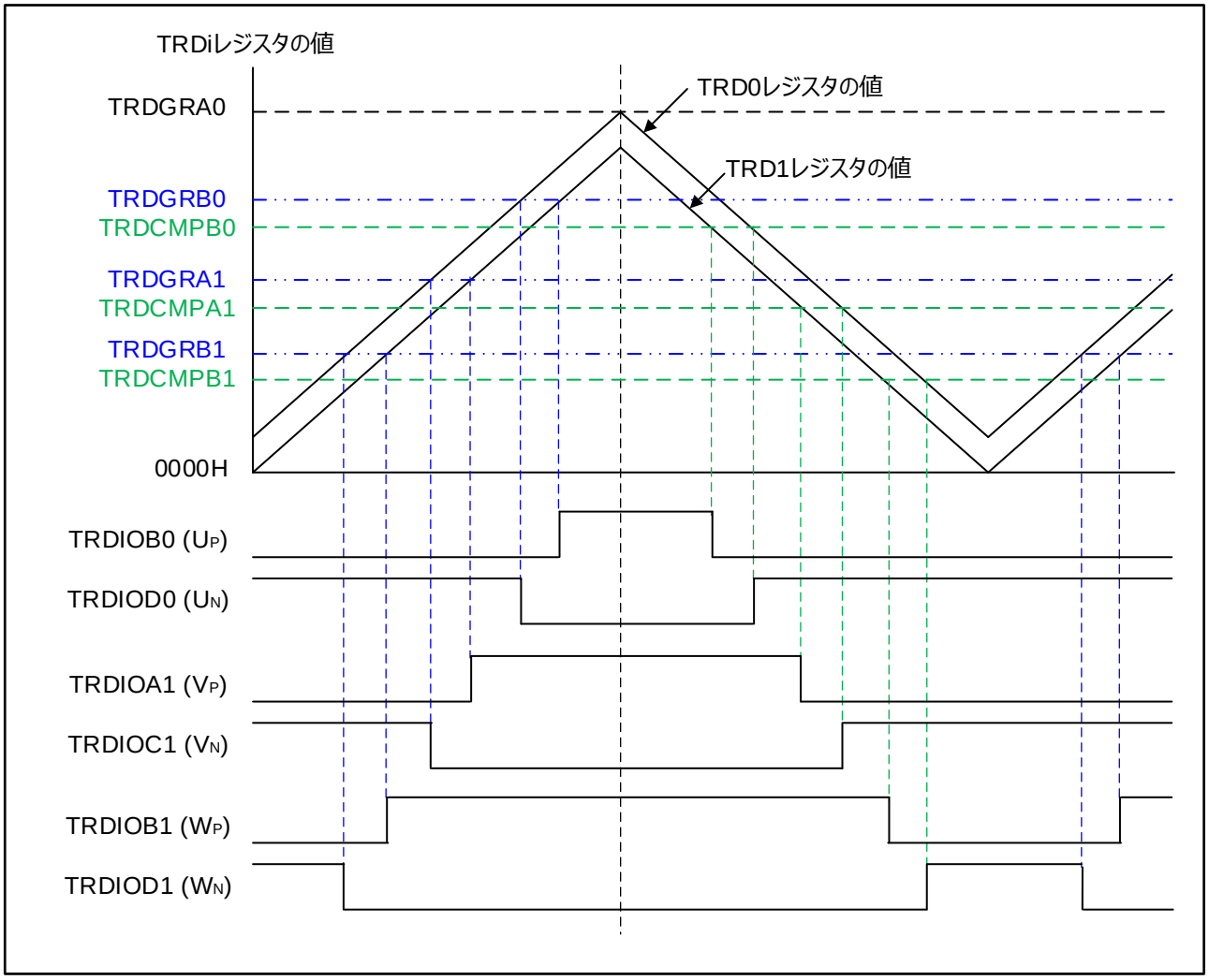


図 2-5 拡張相補 PWM モード（非対称波形出力）の出力波形例

PWM デューティ指令値から PWM のデューティ設定：

拡張相補 PWM モード（非対称波形出力）時のデューティの設定方法をまとめます。

正規化された PWM デューティ指令値と PWM 周期カウント値から正相アクティブ・レベル幅を求め、PWM 波形出力シフト処理を使用して、バッファ・レジスタ（アップカウント側の TRDGRD0、TRDGRC1、TRDGRD1、およびダウンカウント側の TRDCMPD0、TRDCMPC1、TRDCMPD1）に値を設定します。

正相アクティブ・レベル幅 = PWM デューティ指令値 × PWM 周期カウント値
各相のカウント値 = PWM 周期カウント値 – 正相アクティブ・レベル幅
【PWM 波形出力シフト処理】
バッファ・レジスタ（TRDGRD0, TRDGRC1, TRDGRD1, TRDCMPD0, TRDCMPC1, TRDCMPD1）に値を設定

PWM 波形出力シフト処理：

本サンプルソフトウェアでは、以下の条件をもとに出力波形をシフトさせます。

(1) A/D 変換時間やスイッチングノイズによる測定安定待ち時間を考慮し、システムのシフト幅を決めます。本サンプルプログラムでは 5 [µs]としています。1 周期で 2 回の A/D 変換を行うため計 10 [µs]のシフト幅を設けます。

(2) 単一シャント電流検出有効期間を各相の PWM 値より、最大 PWM と最小 PWM の差から求めます。
検出有効期間 = 最大 PWM 値 – 最小 PWM 値

(3) シフト幅と検出有効期間を判定し、各相の PWM 値を求めます。
（シフト幅 ×2）の期間 > 検出有効期間の場合、中間相を基準として 2 相の値をシフトします。
次に、（シフト幅 ×2）の期間 < 検出有効期間の場合、中間の相をシフトします。中間相のシフト幅は、中間相と最大 PWM と最小 PWM の位相差により求めます。

図 2-6 に PWM 波形出力シフトの例を示します。

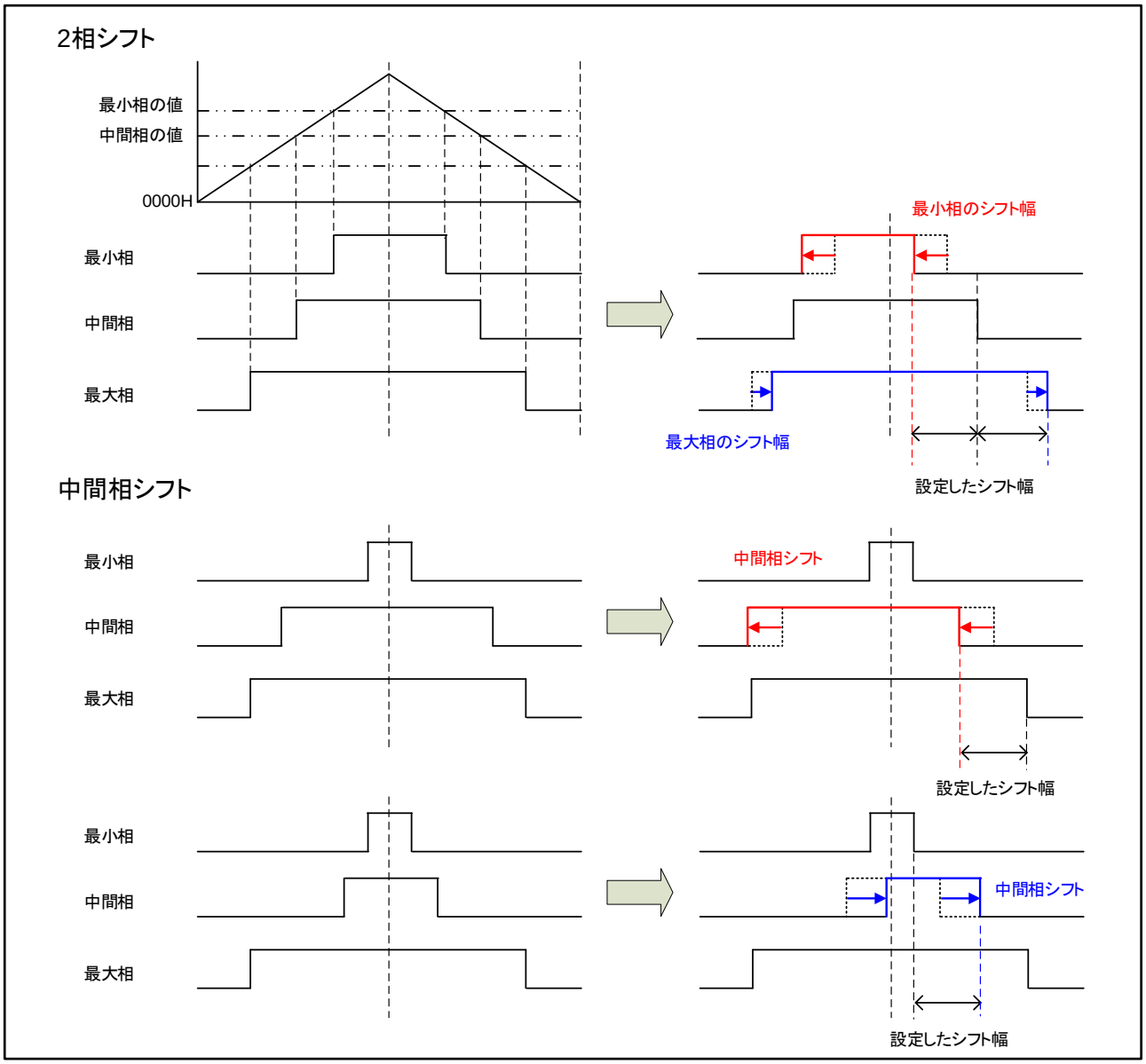


図 2-6 PWM 波形出力シフト例

A/D 変換トリガ設定：

単一シャント電流検出は、出力パターンに対応したタイミングで A/D 変換データの取得を行う必要があります。本サンプルプログラムでは、タイマ RDe の A/D トリガ生成機能を使用して、三角波のダウンカウント中に 2 回の A/D 変換トリガを要求します。また、間引き機能を使用して、2 回に 1 回の周期で A/D 変換トリガを要求しています（図 2-7 参照）。

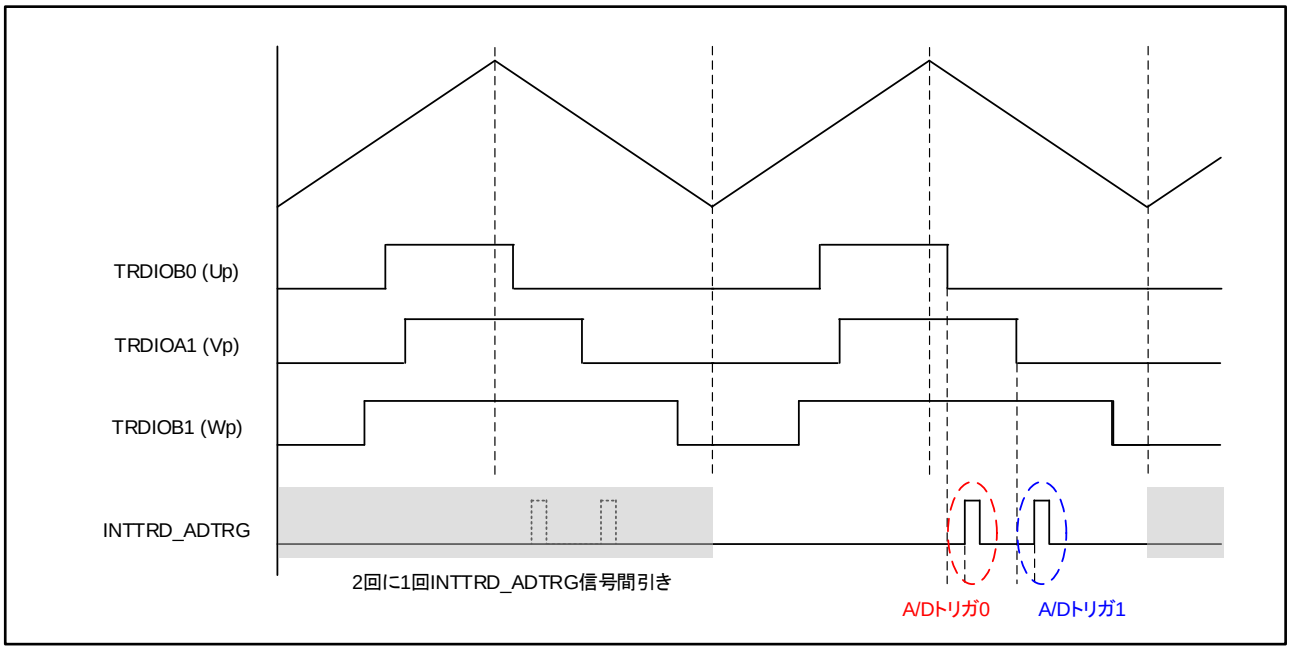


図 2-7 単一シャント電流検出タイミング例

単一シャント電流を取得するための A/D 変換トリガは、ダウンカウント中に 2 回要求します。A/D 変換トリガの要求タイミングは下表の条件で求めます。

表 2-10 A/D 変換トリガの要求タイミング

A/D トリガ生成条件	A/D トリガ 0	A/D トリガ 1
出力電圧：Vu > Vv > Vw	W 相基準	V 相基準
出力電圧：Vu > Vw > Vv	V 相基準	W 相基準
出力電圧：Vv > Vu > Vw	W 相基準	U 相基準
出力電圧：Vv > Vw > Vu	U 相基準	W 相基準
出力電圧：Vw > Vu > Vv	V 相基準	U 相基準
出力電圧：Vw > Vv > Vu	U 相基準	V 相基準

2.2.2.4　DTC

DTC の転送機能により、DTC ベクタ・テーブルに割り当てられた起動要因に紐づくデータを RAM 領域に転送させる事が可能です。

サンプルプログラムでは A/D 変換完了割り込みを起動要因として A/D コンバータのチャネル 0 変換結果を転送します。2 回転送が完了したタイミングで A/D 変換完了割り込み（INTAD）が発生します。

A/D 変換完了要因による DTC の動作イメージを図 2-8 に示します。

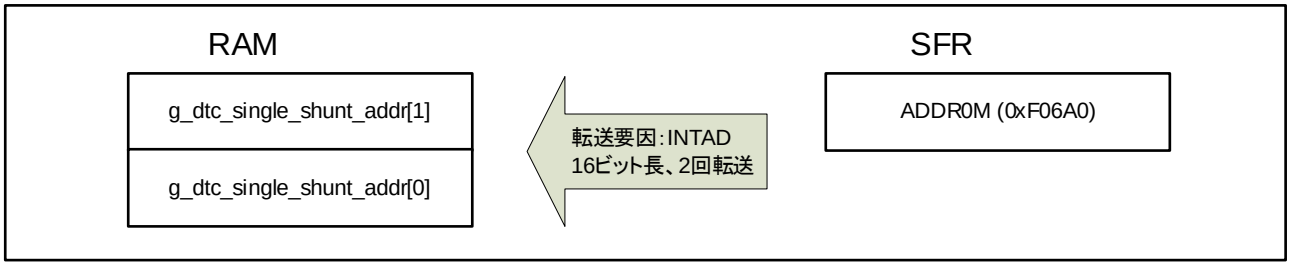


図 2-8 A/D 変換完了要因による DTC の動作イメージ

2.2.2.5　アプリケーション・アクセラレータ・ユニット（AAU）

アプリケーション・アクセラレータ・ユニットは、RL78/F23, F24 に搭載された演算回路です。BLDC モータのセンサレスベクトル制御（FOC：Field Oriented Control）や DC/DC コンバータの PI 制御で使用することができます。AAU を使用することで CPU のソフトウェア負荷を軽減させることができます。

本サンプルプログラムでは AAU を使用して、以下の演算を行います。

表 2-11 使用する AAU 機能

AAU 演算処理モード	処理概要
クラーク／パーク変換＋PI 制御（モータ）	クラーク／パーク変換（3 相–2 相変換、回転座標変換）および PI 制御演算を行う
逆パーク／クラーク変換	逆パーク／クラーク変換（回転座標変換、2 相–3 相変換）

(1)　クラーク／パーク変換＋PI 制御

AAU では、クラーク変換、パーク変換、PI 制御をまとめて処理することができます。

クラーク変換：3 相の電流（I_U, I_V, I_W）を 2 相に変換（I_α, I_β）

パーク変換：固定座標から回転座標に変換

PI 制御：P 制御（比例制御）と I 制御（積分制御）を行い、2 相の出力電流（I_d, I_q）を算出

(2)　逆パーク／クラーク変換

逆パーク変換：回転座標から固定座標に変換

逆クラーク変換：2 相電圧（V_α, V_β）を 3 相電圧（V_U, V_V, V_W）に変換

2.3　ソフトウェア構成

2.3.1　ファイル構成

サンプルプログラムのファイル構成を表 2-12 に示します。

表 2-12 サンプルプログラムのファイル構成 (1/2)

RL78F24_SSL_FOC180SS_V105		
prj	RL78F24_SSL_FOC180.mtpj	CS+プロジェクト・ファイル（CS+用）
	cstart.asm	スタートアップ・ルーチン定義ファイル（CC-RL 用）
	monitor.asm	デバッグモニタ用セクション定義ファイル（CC-RL 用）
	OPT_BYTE.asm	デバイス･オプション･バイト定義ファイル（CC-RL 用）
	security_id.asm	セキュリティ ID 定義ファイル（CC-RL 用）
	stkinit.asm	スタック領域初期化ファイル（CC-RL 用）
	iodefine.h	レジスタ定義ファイル（CC-RL 用）
	RL78F24_SSL_FOC180.dep	IAR ワークスペース・ファイル一式（IAR 用）
	RL78F24_SSL_FOC180.ewd	
	RL78F24_SSL_FOC180.ewp	
	RL78F24_SSL_FOC180.ewt	
	RL78F24_SSL_FOC180.eww	
inc	control_math.h	制御演算処理ヘッダ
	control_parameter.h	モータ制御パラメータ定義用ヘッダ
	error_management.h	エラー管理処理ヘッダ
	foc_current_ref_ctrl.h	ベクトル制御の速度制御・電流指令値制御ヘッダ
	foc_current_regulator.h	ベクトル制御の電流制御ヘッダ
	foc_math.h	ベクトル制御演算ヘッダ
	motor_parameter.h	モータ・パラメータ定義用ヘッダ
	mtr_speed_ref_ctrl.h	速度指令値制御ヘッダ
	rl78_common.h	デバイス依存共通定義用ヘッダ
	rl78_interrupt.h	割り込み処理定義用ヘッダ
	rssk_inv.h	インバータ依存処理ヘッダ
	sequence.h	システム状態管理ヘッダ
	thermistor.h	サーミスタ・テーブル定義用ヘッダ
	user_control.h	ユーザ定義処理ヘッダ
lib	r_fixed_point_math.h	固定小数点演算ライブラリ・ヘッダ
	r_foc_angle_est.h	位置推定ライブラリ・ヘッダ
	r_scale_adjust.h	スケーリング・マクロ
lib¥CC-RL	r_fixed_point_math.lib	固定小数点演算ライブラリ
	r_foc_angle_est.lib	位置推定ライブラリ

表 2-12 サンプルプログラムのファイル構成 (2/2)

RL78F24_SSL_FOC180SS_V105		
src	contrl_math.c	制御演算処理部
	error_management.c	エラー管理処理部
	foc_current_ref_ctrl.c	ベクトル制御の速度制御・電流指令値制御部
	foc_current_regulator.c	ベクトル制御の電流制御部
	foc_math.c	ベクトル制御演算部
	mtr_speed_ref_ctrl.c	速度指令値制御部
	rl78_interrupt.c	割り込み処理部
	rssk_inv.c	インバータ依存処理部
	sequence.c	システム状態管理部
	user_control.c	ユーザ定義処理部
	mcu_debug_option_for_iar.c	オプション・バイト設定（IAR コンパイラ用）
src¥drv	r_cg_aau.c	AAU ドライバ実装部
	r_cg_aau.h	AAU ドライバ実装ヘッダ
	r_cg_adc.c	A/D 変換ドライバ実装部
	r_cg_adc.h	A/D 変換ドライバ実装ヘッダ
	r_cg_adc_user.c	A/D 変換ドライバ・ユーザ定義実装部
	r_cg_cgc.c	クロック生成回路ドライバ実装部
	r_cg_cgc.h	クロック生成回路ドライバ実装ヘッダ
	r_cg_cgc_user.c	クロック生成回路ドライバ・ユーザ定義実装部
	r_cg_comp.c	内蔵コンパレータ・ドライバ実装部
	r_cg_comp.h	内蔵コンパレータ・ドライバ実装ヘッダ
	r_cg_comp_user.c	内蔵コンパレータ・ドライバ・ユーザ定義実装部
	r_cg_dac.c	D/A コンバータ・ドライバ実装部
	r_cg_dac.h	D/A コンバータ・ドライバ実装ヘッダ
	r_cg_dac_user.c	D/A コンバータ・ドライバ・ユーザ定義実装部
	r_cg_dtc. c	DTC ドライバ実装部
	r_cg_dtc. h	DTC ドライバ実装ヘッダ
	r_cg_dtc_user. c	DTC ドライバ・ユーザ定義実装部
	r_cg_macrodriver.h	CG 用マクロヘッダ
	r_cg_port.c	ポート入出力ドライバ実装部
	r_cg_port.h	ポート入出力ドライバ実装ヘッダ
	r_cg_port_user.c	ポート入出力ドライバ・ユーザ定義実装部
	r_cg_timer.c	タイマ・ドライバ実装部
	r_cg_timer.h	タイマ・ドライバ実装ヘッダ
	r_cg_timer_user.c	タイマ・ドライバ・ユーザ定義実装部
	r_cg_timer_rde.c	タイマ RDe ドライバ実装部
	r_cg_timer_rde.h	タイマ RDe ドライバ実装ヘッダ
	r_cg_timer_rde_user.c	タイマ RDe ドライバ・ユーザ定義実装部
	r_cg_userdefine.h	コード生成用ユーザ・ヘッダ
	r_cg_wdt.c	WDT ドライバ実装部
	r_cg_wdt.h	WDT ドライバ実装ヘッダ
	r_cg_wdt_user.c	WDT ドライバ・ユーザ定義実装部
	r_main.c	main 関数実装部
	r_systeminit.c	ハードウェア初期化定義

2.3.2 モジュール構成

サンプルプログラムの階層構造を図 2-9 に示します。

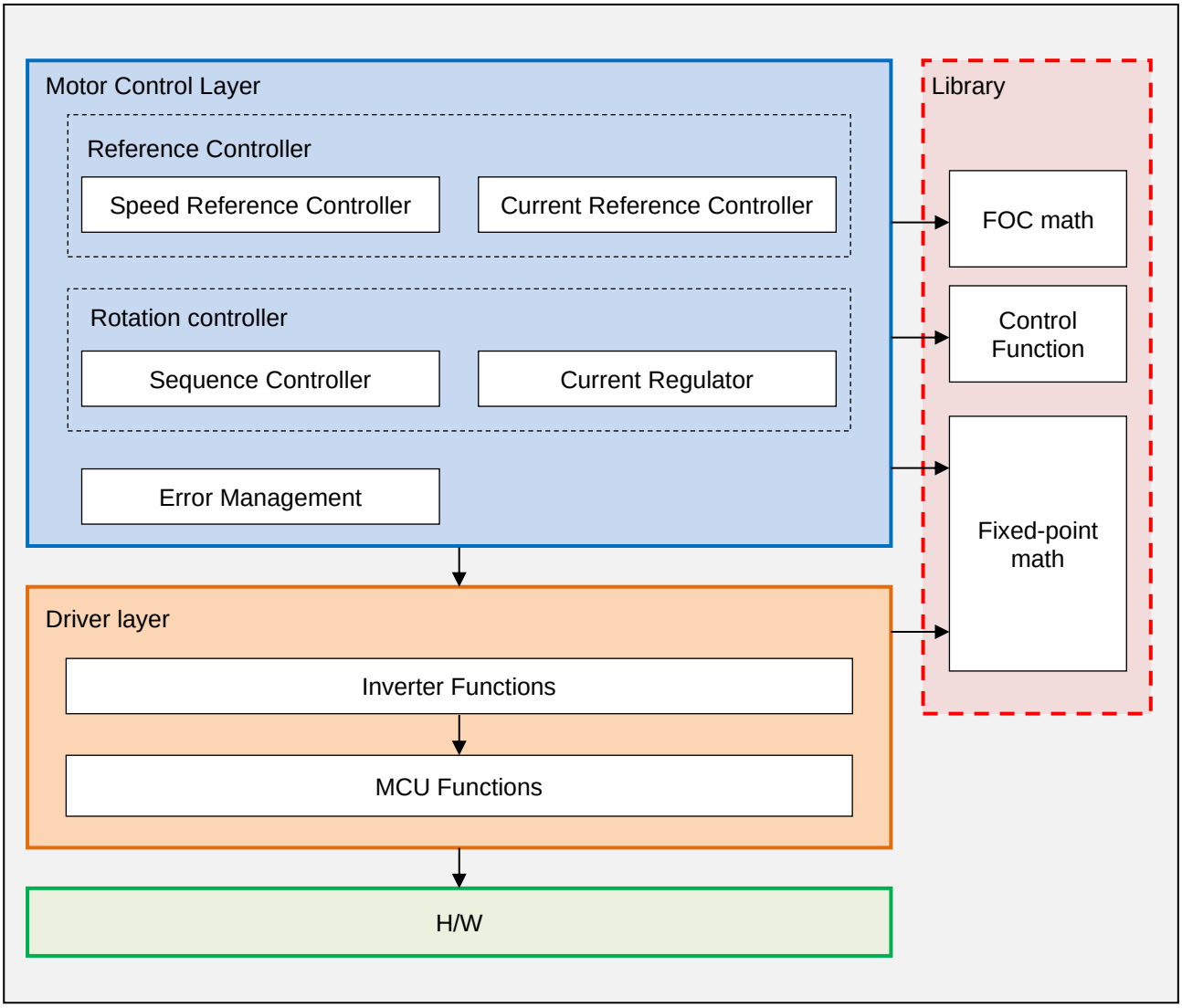


図 2-9 ソフトウェア全体構成

2.4　ソフトウェア仕様

本アプリケーションノートが対象とするソフトウェアの基本仕様を表 2-13 に示します。

表 2-13 ソフトウェアの基本仕様

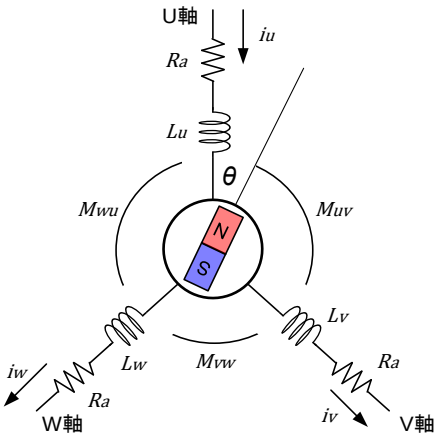
項目	内容
制御方式	センサレスベクトル制御（単一シャント方式）
モータ回転開始／停止	以下 2 つのうち、どちらか一方を選択して動作する • リセット解除後以下のタイミングで制御回転状態を遷移する – 3 [s]後、1000 [rpm]を速度指令値として動作開始 – 上記動作から 10 [s]後、2000 [rpm]に回転速度指令値を変更 – 上記動作から 10 [s]後、3000 [rpm]に回転速度指令値を変更 – 上記動作から 10 [s]後、モータ停止 – 上記動作から 1 [s]後、カウンタリセット • リセット解除後、PWM 信号入力により回転速度指令値を制御する – 入力する PWM 信号は 10 [Hz] ～ 1000 [Hz]とする – デューティが 100%のとき、回転速度指令値を 3000 [rpm]とする – デューティが 0%のとき、回転速度指令値を 0 [rpm]とする
回転子磁極位置検出	センサレス（電流推定誤差法）
キャリア周波数（PWM）	20 [kHz]
デッドタイム	1 [µs]
制御周期	100 [µs] （50 × 2）
回転速度制御範囲	CW/CCW 共に 500 [rpm] ～ 3000 [rpm]
保護停止処理	以下のいずれかの条件の時、モータ制御信号出力（6 本）を非アクティブにする – 3 相の電流値のいずれか 1 相でも±16.97 [A]（10Arms + 20%）を超過（100 [µs]毎に監視） – インバータ電源電圧 V_{dc} が 28.0 [V]を超過（100 [µs]毎に監視） – インバータ電源電圧 V_{dc} が 8.0 [V]を下回る（100 [µs]毎に監視） – 回転速度が 5000 [rpm]を超過（1 [ms]毎に監視） – INTP0 端子に Low レベル入力 （信号停止動作はリアルタイム、エラー検知は 100 [µs]毎に監視） – 基板温度が 120 [°C]を超過（1 [ms]毎に監視） – モータコイルエンド温度が 180 [°C]を超過（1 [ms]毎に監視）

3.　モータ制御方法

本章ではサンプルプログラムで用いる、永久磁石同期モータ（PMSM）のベクトル制御について説明します。

3.1　モータ制御システムの電圧方程式

正弦波状の磁束分布を持った永久磁石同期モータの電圧方程式は図 3-1 のように表すことができます。



$$\begin{pmatrix} v_u \\ v_v \\ v_w \end{pmatrix} = R_a \begin{pmatrix} i_u \\ i_v \\ i_w \end{pmatrix} + p \begin{pmatrix} \varphi_u \\ \varphi_v \\ \varphi_w \end{pmatrix}$$

$$\begin{pmatrix} \varphi_u \\ \varphi_v \\ \varphi_w \end{pmatrix} = \begin{pmatrix} L_u & M_{uv} & M_{wu} \\ M_{uv} & L_v & M_{vw} \\ M_{wu} & M_{vw} & L_w \end{pmatrix} \begin{pmatrix} i_u \\ i_v \\ i_w \end{pmatrix} + \Psi \begin{pmatrix} \cos\theta \\ \cos(\theta - 2\pi/3) \\ \cos(\theta + 2\pi/3) \end{pmatrix}$$

v_u, v_v, v_w: 各相電機子電圧
i_u, i_v, i_w: 各相電機子電流
$\varphi_u, \varphi_v, \varphi_w$: 各相電機子鎖交磁束
R_a: 各相電機子電流
p: 微分演算子
L_u, L_v, L_w: 各相自己インダクタンス
M_{uv}, M_{vw}, M_{wu}: 各相間相互インダクタンス
ψ: 永久磁石による電機子鎖交磁束の最大値
θ: U 相からの永久磁石（回転子）の進み角

図 3-1 三相永久磁石同期モータの概要図

ここで自己インダクタンスと相互インダクタンスは次式のように表されます。

$$\begin{cases} L_u = l_a + L_a - L_{as}\, cos(2\theta) \\ L_v = l_a + L_a - L_{as}\, cos(2\theta + 2\pi/3) \\ L_w = l_a + L_a - L_{as}\, cos(2\theta - 2\pi/3) \end{cases}$$

$$\begin{cases} M_{uv} = -L_a/2 - L_{as}\, cos(2\theta - 2\pi/3) \\ M_{vw} = -L_a/2 - L_{as} \cos(2\theta) \\ M_{wu} = -L_a/2 - L_{as}\, cos(2\theta + 2\pi/3) \end{cases}$$

l_a:　一相あたりの漏れインダクタンス
L_a:　一相あたりの有効インダクタンスの平均値
L_{as}:　一相あたりの有効インダクタンスの振幅

3.2　ベクトル制御

回転子の永久磁石の磁束（N 極）方向に d 軸を定め、d 軸から 90 度進んだ方向に q 軸を取ることにすると、dq 座標系から見た永久磁石同期モータの電圧方程式を得るためには以下の変換行列を用いればよいことになります。

$$C = \sqrt{\frac{2}{3}}\begin{pmatrix} \cos\theta & \cos(\theta - 2\pi/3) & \cos(\theta + 2\pi/3) \\ -\sin\theta & -\sin(\theta - 2\pi/3) & -\sin(\theta + 2\pi/3) \end{pmatrix}$$

$$\begin{pmatrix} v_d \\ v_q \end{pmatrix} = C\begin{pmatrix} v_u \\ v_v \\ v_w \end{pmatrix}$$

上記の座標変換により dq 座標系での永久磁石同期モータの電圧方程式は以下のように表すことができます。

$$\begin{pmatrix} v_d \\ v_q \end{pmatrix} = \begin{pmatrix} R_a + pL_d & -\omega L_q \\ \omega L_d & R_a + pL_q \end{pmatrix}\begin{pmatrix} i_d \\ i_q \end{pmatrix} + \begin{pmatrix} 0 \\ \omega\psi_a \end{pmatrix}$$

v_d, v_q: 各相電機子電圧　　L_d, L_q: 各相自己インダクタンス
i_d, i_q: 各相電機子電流　　$L_d = l_a + 3/2(L_a - L_{as})$、$L_q = l_a + 3/2(L_a + L_{as})$
ψ_a: 永久磁石による電機子鎖交磁束の実効値　　R_a: 各相電機子抵抗
$\psi_a = \sqrt{3/2}\psi$

これにより静止している三相固定子に流れていた交流は、回転子である永久磁石と同期して回転している 2 相の固定子に直流として現れると見なすことができます。

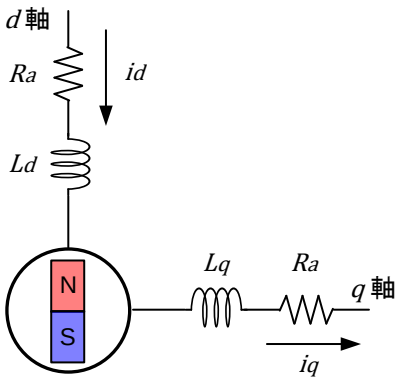


図 3-2 二相直流モータの概念図

モータに生じるトルクの大きさは電流ベクトルと電機子鎖交磁束の外積より下記のように求まります。この式の右辺第一項をマグネットトルク、右辺第二項をリラクタンストルクと呼びます。

$$T = P_n\{\psi_a i_q + (L_d - L_q) i_d i_q\}$$

T: モータトルク　　P_n: 極対数

d 軸と q 軸のインダクタンスの差がないモータを突極性が無いモータと呼びます。この場合、リラクタンストルクは 0 になるので、q 軸電流に比例してトルクは大きくなります。このため、q 軸電流をトルク電流と呼ぶことがあります。一方、d 軸電流は、その大きさを変化させることで q 軸電圧にとってあたかも永久磁石の磁束の大きさが変化しているかのように見なせる働きをするので励磁電流と呼ぶことがあります。

一般的に SPMSM（表面磁石型ブラシレス DC モータ）は突極性が無いので、速度制御の際、トルクを発生させるのに不要な d 軸電流は 0 に制御します。これを id=0 制御と呼びます。一方で、この時のモータの運動方程式は、下記のように表されるので、速度を上昇させたい場合は、q 軸電流 iq を上昇させればよいことが解ります。

$$I\frac{d\omega}{dt} = P_n\psi_a i_q - T_L$$

T_L: 負荷トルク　　I: モータの慣性モーメント

速度制御は、この運動方程式を解く事によってではなく、PI 制御によって行います。速度 PI 制御によって q 軸電流の指令値を得ます。

$$i_q^* = (K_{P\omega} + \frac{K_{I\omega}}{s})(\omega^* - \omega)$$

$K_{P\omega}$: 速度 PI 比例ゲイン　　$K_{I\omega}$: 速度 PI 積分ゲイン　　s: ラプラス演算子

d 軸と q 軸の電流指令値により早く安定させるため、電流値にも PI 制御を行います。電流 PI 制御によって指令電圧を得ます。

$$v_d^* = (K_{Pi_d} + \frac{K_{Ii_d}}{s})(i_d^* - i_d)$$

K_{Pi_d}: d軸電流PI比例ゲイン　　K_{Ii_d}: d軸電流PI積分ゲイン

$$v_q^* = (K_{Pi_q} + \frac{K_{Ii_q}}{s})(i_q^* - i_q)$$

K_{Pi_q}: q軸電流 PI 比例ゲイン　　K_{Ii_q}: q軸電流 PI 積分ゲイン

モータが回転すると誘起電圧が発生し、d 軸電圧には q 軸電流による影響が、また q 軸電圧には d 軸電流と永久磁石磁束による影響が、速度が大きくなるにつれて顕著になります。この d 軸と q 軸の干渉は、電流値の安定を遅らせる働きをしてしまうことがあります。これを避けるために、各軸の干渉項をあらかじめキャンセルするようにフィードフォワードして各軸の電圧を算出します。

$$v_d^* = (K_{Pi_d} + \frac{K_{Ii_d}}{s})(i_d^* - i_d) - \omega L_q i_q$$

$$v_q^* = (K_{Pi_q} + \frac{K_{Ii_q}}{s})(i_q^* - i_q) + \omega(L_d i_d + \psi_a)$$

このようにして干渉項の影響を無くす方法を非干渉制御と呼びます。これにより、d 軸と q 軸を独立に制御することが可能になります。

ベクトル制御は、互いに独立して制御することができなかった三相の交流モータを独立に制御可能な 2 相の直流モータへと変換し、トルクや回転子の速度、位置を管理しながら制御する方法といえます。

ベクトル制御の制御フローを図 3-3 に示します。

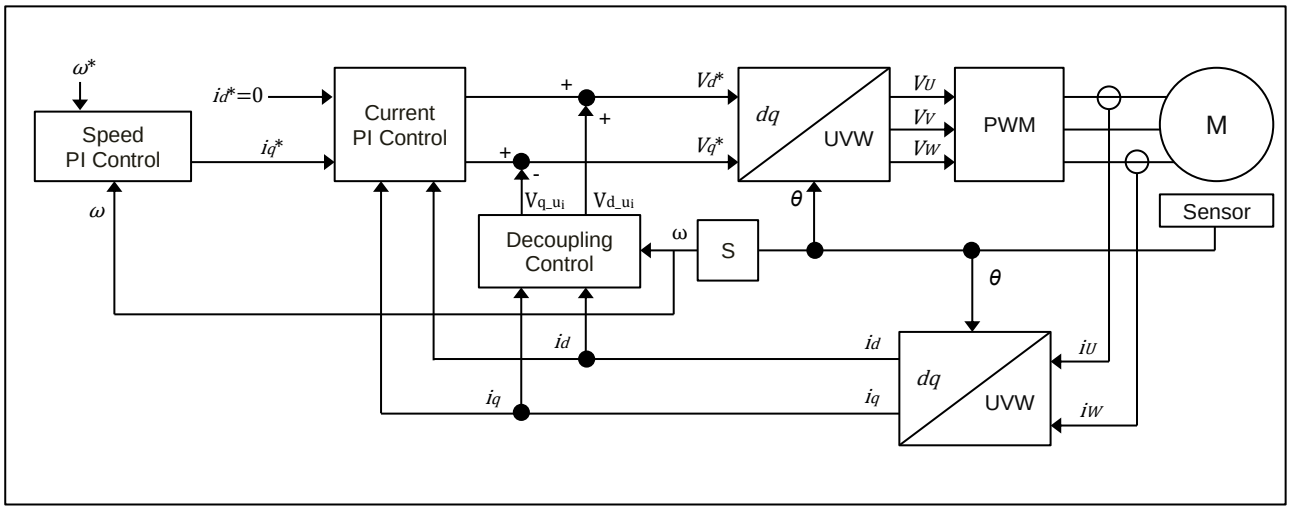
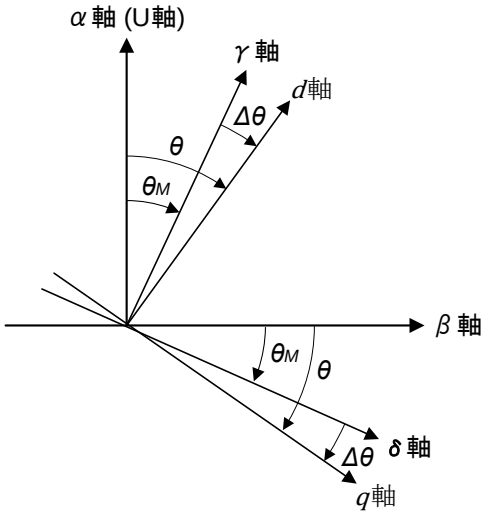


図 3-3 ベクトル制御の制御フロー

3.3　電流推定誤差に基づくセンサレスベクトル制御

ベクトル制御は回転子の位置に応じて電圧を設定するので、エンコーダやレゾルバ等の位置センサが必要になります。これらの位置センサを使わない、即ちセンサレスベクトル制御を行う場合、位置情報を何らかの方法で推定する必要があります。昨今は、センサレスでのモータ制御の需要が高まり、位置情報を推定する為に様々な方法が提案されています。ここでは、本システムで採用している電流推定誤差に基づくセンサレスベクトル制御について紹介します。

実モータの位置情報はありませんので d 軸の位置は分かりません。図のように d 軸から $\Delta\theta$ 遅れた所に γ 軸を定め、γ 軸から 90 度進んだところに δ 軸を取ることにすると、dq 軸から γδ 軸への変換式は図 3-4 のように書く事ができます。



$$\begin{pmatrix}\gamma\\ \delta\end{pmatrix}=\begin{pmatrix}\cos\Delta\theta & \sin\Delta\theta\\ -\sin\Delta\theta & \cos\Delta\theta\end{pmatrix}\begin{pmatrix}d\\ q\end{pmatrix}$$

図 3-4 dq 軸と γδ 軸の関係

これを SPMSM の電圧方程式に施し、電流の状態方程式の形に書くと、下記のようになります。

$$p\begin{pmatrix}i_\gamma\\ i_\delta\end{pmatrix}=-\begin{pmatrix}\frac{R}{L} & -\dot{\theta}_M\\ \dot{\theta}_M & \frac{R}{L}\end{pmatrix}\begin{pmatrix}i_\gamma\\ i_\delta\end{pmatrix}+\frac{1}{L}\begin{pmatrix}v_\gamma\\ v_\delta\end{pmatrix}-\frac{K_E\dot{\theta}}{L}\begin{pmatrix}-\sin\Delta\theta\\ \cos\Delta\theta\end{pmatrix}$$

この状態方程式に後退微分近似（オイラー近似）を使用し離散化します。

$$\begin{pmatrix}i_\gamma(n)\\ i_\delta(n)\end{pmatrix}=\begin{pmatrix}i_\gamma(n-1)\\ i_\delta(n-1)\end{pmatrix}+\frac{T}{L}\left\{\begin{pmatrix}v_\gamma(n-1)\\ v_\delta(n-1)\end{pmatrix}-R\begin{pmatrix}i_\gamma(n-1)\\ i_\delta(n-1)\end{pmatrix}\right.$$
$$\left.-\dot{\theta}_M(n-1)L\begin{pmatrix}-i_\delta(n-1)\\ i_\gamma(n-1)\end{pmatrix}-e(n-1)\begin{pmatrix}-\sin\Delta\theta(n-1)\\ \cos\Delta\theta(n-1)\end{pmatrix}\right\}$$
$$\because\ e(n-1)=K_E\dot{\theta}(n-1)$$

ここで、モータモデルとして、モータ・パラメータを実モータのパラメータと等しい値として R_M、L_M、e_M のように示し、$\Delta\theta$ を 0 とした場合を考えます。この時、サンプル点 n の電流値は、以下のように書くことができます。

$$\begin{pmatrix} i_{\gamma M}(n) \\ i_{\delta M}(n) \end{pmatrix} = \begin{pmatrix} i_\gamma(n-1) \\ i_\delta(n-1) \end{pmatrix} + \frac{T}{L_M}\left\{ \begin{pmatrix} v_\gamma(n-1) \\ v_\delta(n-1) \end{pmatrix} - R_M \begin{pmatrix} i_\gamma(n-1) \\ i_\delta(n-1) \end{pmatrix} \right. \\ \left. -\dot{\theta}_M(n-1) L_M \begin{pmatrix} -i_\delta(n-1) \\ i_\gamma(n-1) \end{pmatrix} - e_M(n-1) \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

実モータとモータモデルの電流式の差分より、電流推定誤差は以下のように書くことができます。

$$\begin{pmatrix} \Delta i_\gamma(n) \\ \Delta i_\delta(n) \end{pmatrix} = \frac{T}{L} \begin{pmatrix} e(n-1)\sin\Delta\theta(n-1) \\ e_M(n-1) - e(n-1)\cos\Delta\theta(n-1) \end{pmatrix}$$

$\Delta\theta$ が十分に小さいときは以下のように近似することができます。

$$\begin{pmatrix} \Delta i_\gamma(n) \\ \Delta i_\delta(n) \end{pmatrix} \approx \frac{T}{L} \begin{pmatrix} e(n-1)\Delta\theta(n-1) \\ -\Delta e(n-1) \end{pmatrix} \\ \Delta e(n-1) = e(n-1) - e_M(n-1)$$

Δe と $\Delta\theta$ が共に 0 であれば、モータモデルと実モデルが合致して回転している状況とみなすことができます。Δe が 0 になるように e_M に Δi_δ をフィードバックする事で推定し、同様に $\Delta\theta$ が 0 になるように θ_M の値に Δi_γ をフィードバックする事で推定し、モータモデルと実モデルを一致させます。e_M の推定式は以下で表せます。

$$e_M(n) = e_M(n-1) - K_e \Delta i_\delta(n)$$

K_e は速度起電力ゲインを表しています。同様に θ_M の推定式は、1 サンプル周期間の回転子の回転角を加えて以下のように書く事ができます。

$$\theta_M(n) = \theta_M(n-1) + \frac{T}{K_{EM}} e_M(n) + K_\theta \, sgn\{\dot{\theta}_M(n-1)\} \Delta i_\gamma(n) \\ sgn\{\dot{\theta}_M(n-1)\} = \begin{cases} 1; \dot{\theta}_M(n-1) \geq 0 \\ -1; \dot{\theta}_M(n-1) < 0 \end{cases}$$

K_{EM} はモータモデルの起電力係数で、K_θ は位置推定ゲインを示します。また、$p\theta$ の符号の代わりに $p\theta_M$ の符号を使用しています。速度は、上式より以下のように書く事ができます。

$$\dot{\theta}_M = \frac{1}{T}\{\theta_M(n) - \theta_M(n-1)\} = \frac{e_M}{K_{EM}} + \Delta\dot{\theta}_M(n) \\ \Delta\dot{\theta}_M(n) = \frac{K_\theta}{T} sgn\{\dot{\theta}_M(n-1)\} \Delta i_\gamma(n)$$

サンプルプログラムの制御では、この速度式の右辺第 2 項の速度補正項に LPF をかけたものを使用します。ここでは係数 K は $0 < K < 1$ としています。

$$\dot{\theta}_{Mo}(n) = \frac{e_M(n)}{K_{EM}} + \Delta\dot{\theta}_{Mo}(n) \\ \Delta\dot{\theta}_{Mo}(n) = \Delta\dot{\theta}_{Mo}(n-1) + K\{\Delta\dot{\theta}_M(n) - \Delta\dot{\theta}_{Mo}(n-1)\}$$

この制御方法の制御フローを図 3-5 に示します。

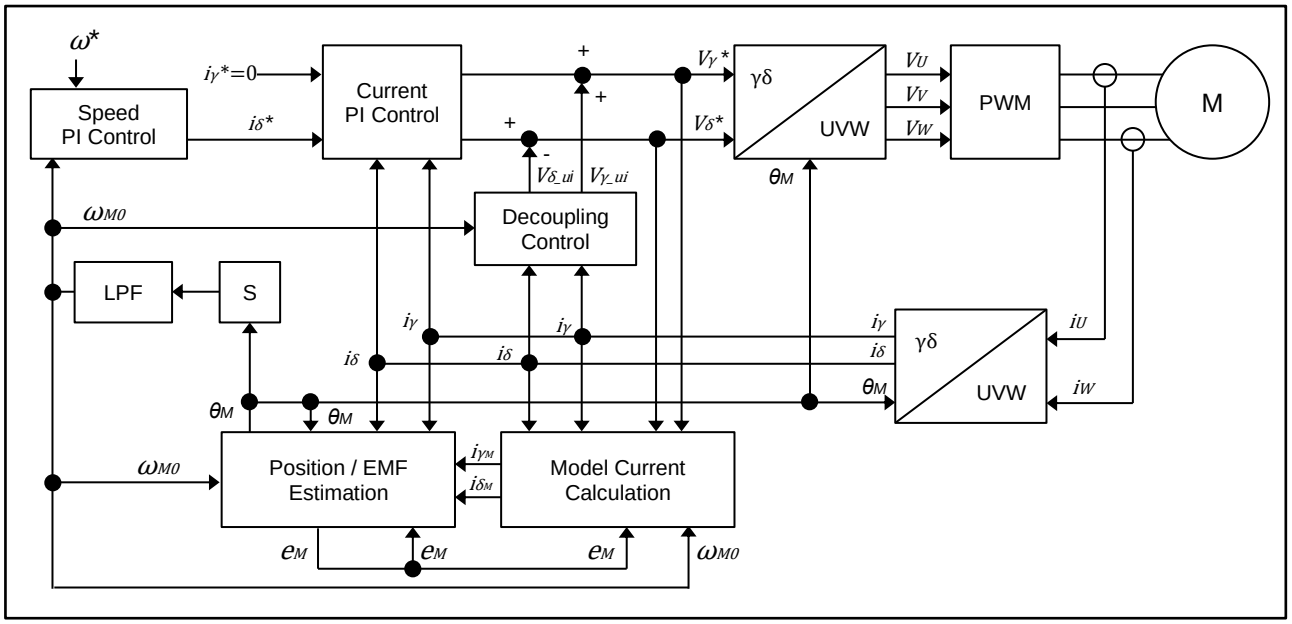


図 3-5 電流推定誤差法によるセンサレスベクトル制御の制御フロー

3.4　三角波比較法

指令値電圧を実際に出力するためには、キャリア波形（三角波）と指令値電圧波形を比較することで出力電圧のパルス幅を決める三角波比較法を用います。この PWM 方式により、正弦波状の指令値電圧を疑似的に出力することができます。

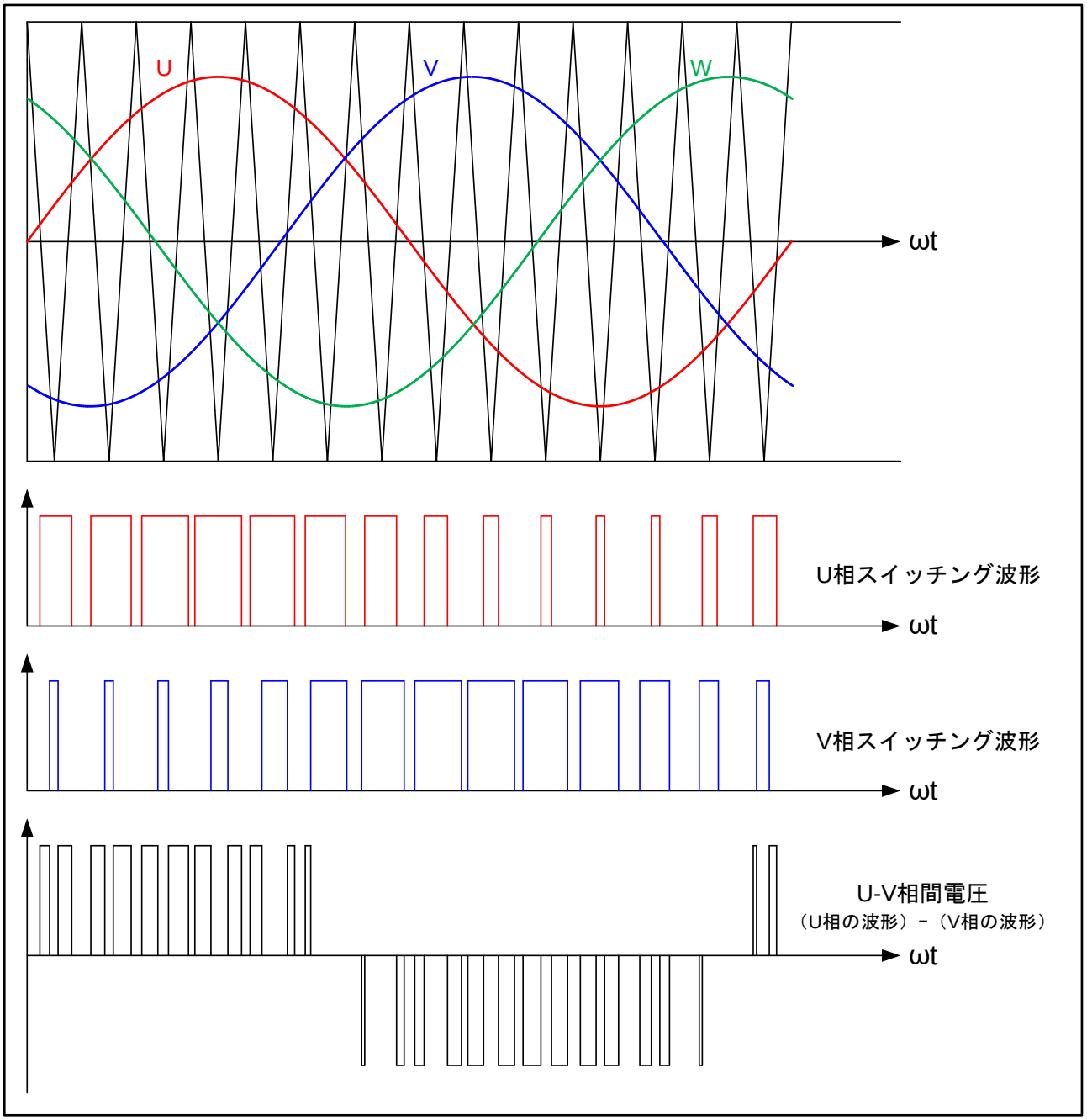


図 3-6 三角波比較法の概念図

図 3-7 のように、出力電圧パルスのキャリア波に対する割合をデューティと呼びます。

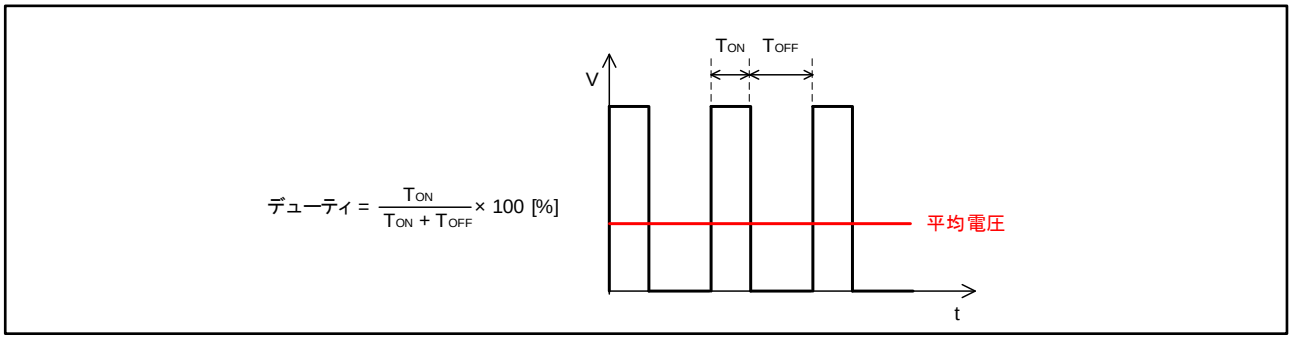


図 3-7 デューティの定義

また、変調率 m を以下のように定義します。

$$m = \frac{V}{E}$$

m:変調率　　V:指令値電圧　　E:インバータ母線電圧

この変調率を、PWM デューティを決めるレジスタに反映させることで制御を行います。

3.5　単一シャント電流検出

スイッチングパターンに関連するシャント電流の読み取りを図 3-8 に示します。下側スイッチの 1 つまたは 2 つがオンの場合のみ、シャント電流を使用して相電流を取得できます。また下側スイッチの状態に応じて算出方法が異なります。

<下側スイッチの 1 つがオンのとき>

- シャント電流はそのスイッチに対応する相電流になります。

<下側スイッチの 2 つがオンのとき>

- シャント電流は対応するスイッチの相電流の合計値になります。また三相の電流の合計値は 0 [A]となることから以下の式が成り立ちます。

下側スイッチがオフの相の電流 ＝ －シャント電流値（下側スイッチ 2 つオンの相の電流の合計値）

単一シャント電流検出では、常に相電流を読み取れるわけではありません。出力整流の後、少なくとも読み取り回路ハードウェアの整定時間や A/D コンバータのサンプリング時間が必要です。したがって PWM 周期において、異なるチャネルの整流間に十分な時間差があるときにのみ、相電流を読み取ることが出来ます。

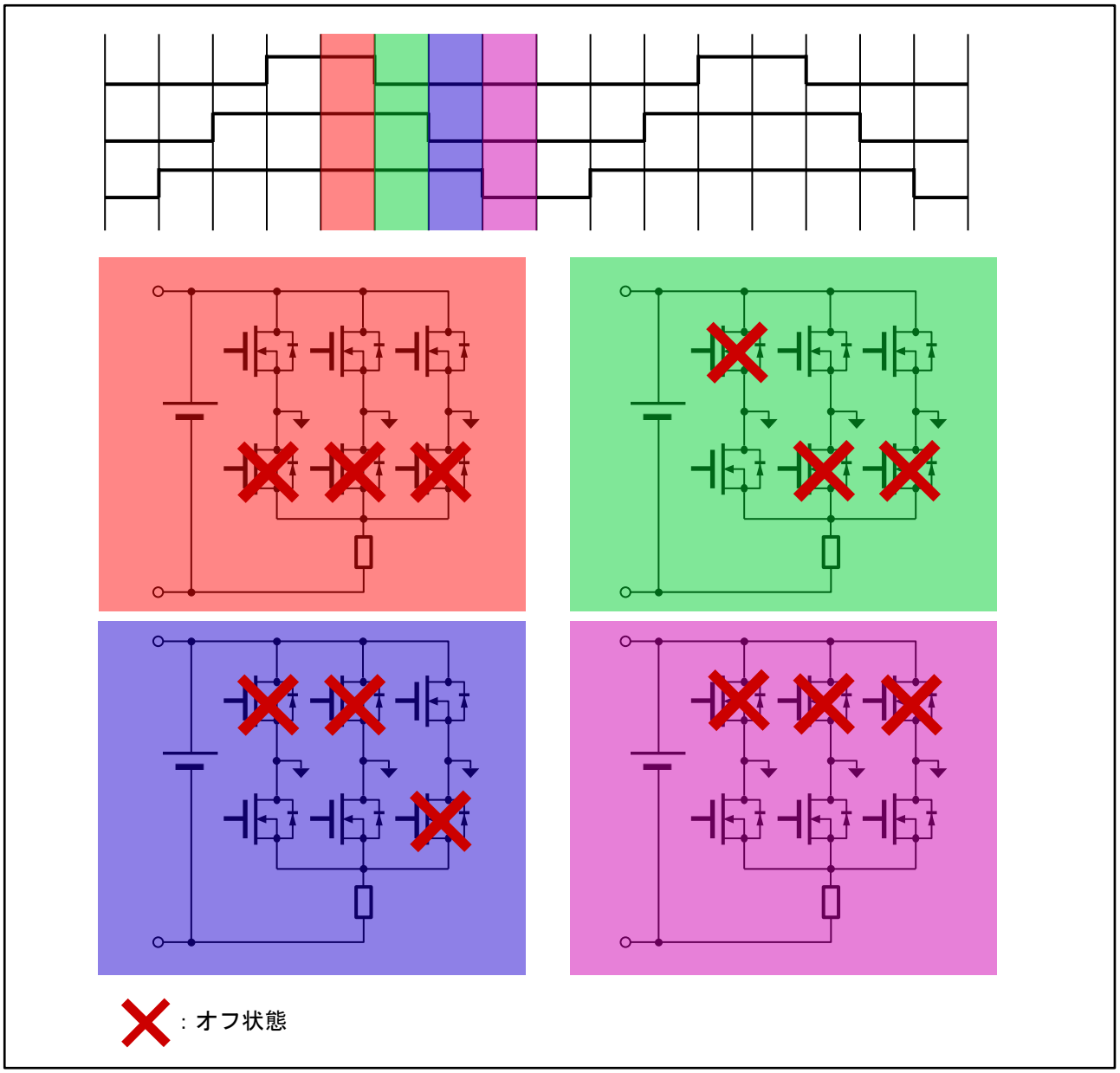


図 3-8 単一シャント電流検出における電流スイッチングパターン

単一シャント電流測定によるセンサレスベクトル制御の実装は、2 つのシャント電流測定による実装とよく似ています。取得した 2 つの A/D 変換結果と各相の出力電圧条件より、各相の電流を求めます。図 3-9 の例では W 相が最大 PWM 値であり、V 相がこれより小さな値、U 相が最小値となっています。V 相と W 相の立ち下がりエッジ間のシャント電流を測定した場合、W 相の電流を測定したことになります。次に、U 相と V 相の立ち下がりエッジ間のシャント電流を測定した場合、V 相と W 相の両方を合わせた電流が測定されます。3 つの相電流すべての合計はゼロになることより、U 相の電流が測定されることになります。表 3-1 に単一シャント電流検出時の各相電流値の検出方法を示します。

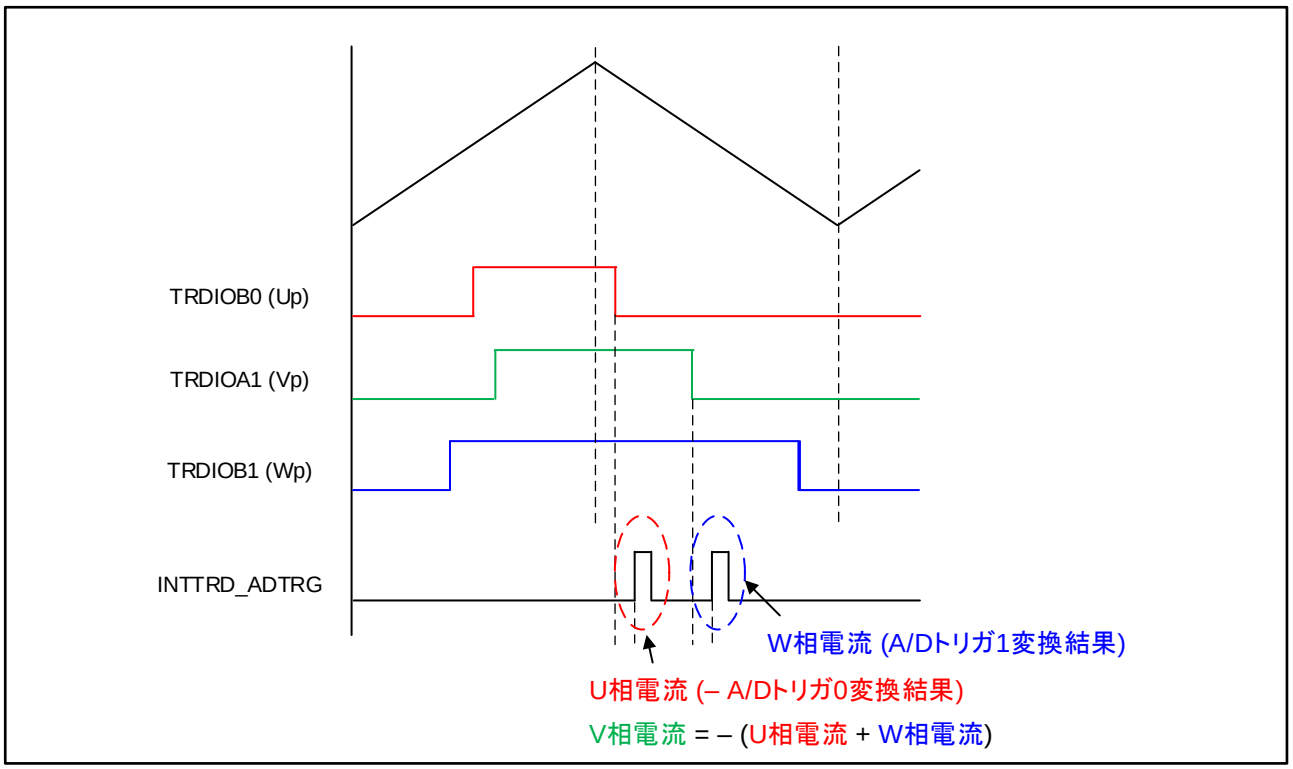


図 3-9 単一シャント電流の検出例

表 3-1 各相の電流検出

出力電圧条件	U 相電流	V 相電流	W 相電流
Vu > Vv > Vw	A/D トリガ 1 変換結果	– (U 相電流 + W 相電流)	– A/D トリガ 0 変換結果
Vu > Vw > Vv	A/D トリガ 1 変換結果	– A/D トリガ 0 変換結果	– (U 相電流 + V 相電流)
Vv > Vu > Vw	– (V 相電流 + W 相電流)	A/D トリガ 1 変換結果	– A/D トリガ 0 変換結果
Vv > Vw > Vu	– A/D トリガ 0 変換結果	A/D トリガ 1 変換結果	– (U 相電流 + V 相電流)
Vw > Vu > Vv	– (V 相電流 + W 相電流)	– A/D トリガ 0 変換結果	A/D トリガ 1 変換結果
Vw > Vv > Vu	– A/D トリガ 0 変換結果	– (U 相電流 + W 相電流)	A/D トリガ 1 変換結果

4. 制御プログラム説明

4.1 制御内容

4.1.1 モータ起動/停止

本サンプルソフトウェアでは以下の 2 通りの制御方法があります。

制御方法の競合を避けるため、初期状態では 4.1.1.1 で示す制御を有効とし、その他はコンパイルスイッチにより無効化しています。

4.1.1.1 電源投入後の経過時間によるモータ動作

モータは H/W リセット解除後、制御を開始します。1 [ms]インターバル・タイマの処理中にカウンタを用意し、カウント数に応じて制御を変更します。制御の変更によって回転速度が変化します。

サンプルソフトウェアでは、リセット解除 3 [s]経過後に 1000 [rpm]の速度指令値を設定し、動作モードを変更して回転制御を開始します。それから 10 [s]経過後に 2000 [rpm]へ速度指令値を変更、さらに 10 [s]経過後に 3000 [rpm]へ速度指令値を変更します。3000 [rpm]動作から 10 [s]経過後、動作モードを変更し回転制御を停止します。停止後 1 [s]でカウンタをリセットし、再度上記の動作を繰り返し実行します。

4.1.1.2　PWM 入力の指令値によるモータ動作

PWM 通信ポートに与えられたデューティ比からモータの回転数を生成します。

PWM 通信ポートにおける電圧レベルの High 期間と Low 期間の時間を比較してデューティ比を生成し、得られたデューティ比からモータ回転数を演算します。

デューティ比 100 [%]の状態を 0 [rpm]（通電停止状態）、0 [%]を 3000 [rpm]として回転指令値を生成します。

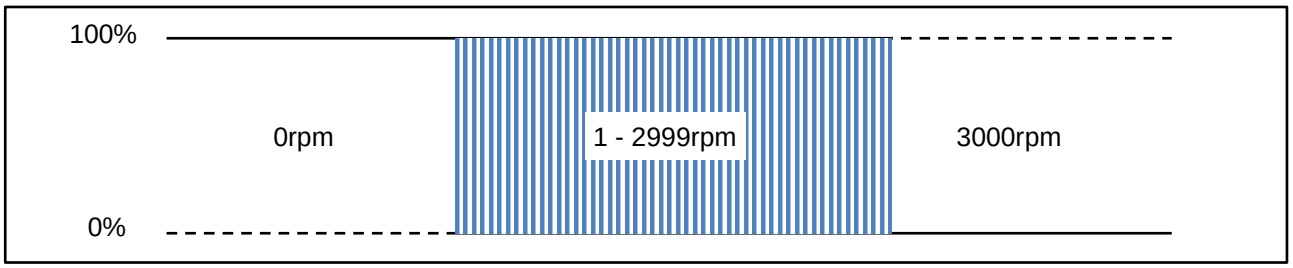


図 4-1 PWM から回転指令値への変換

(1)　デューティ測定方法

デューティの測定には TAU0（TI05 入力）を使用します。

入力パルス間隔測定機能を使用し、H 期間、L 期間両方の時間を測定する必要があるため入力エッジは両エッジに設定します。

エッジ検出の割り込み時にポート機能による入力レベルのチェックを行い、立ち上がり、または立ち下がりのどちらの入力であるかを判断します。立ち下がりエッジから立ち上がりエッジまでの間を Low 幅期間として測定します。また、立ち下がりエッジの検出タイミングで High 幅、Low 幅期間が取得できている状態である場合はその 2 つの値を使用してデューティ比を算出します。

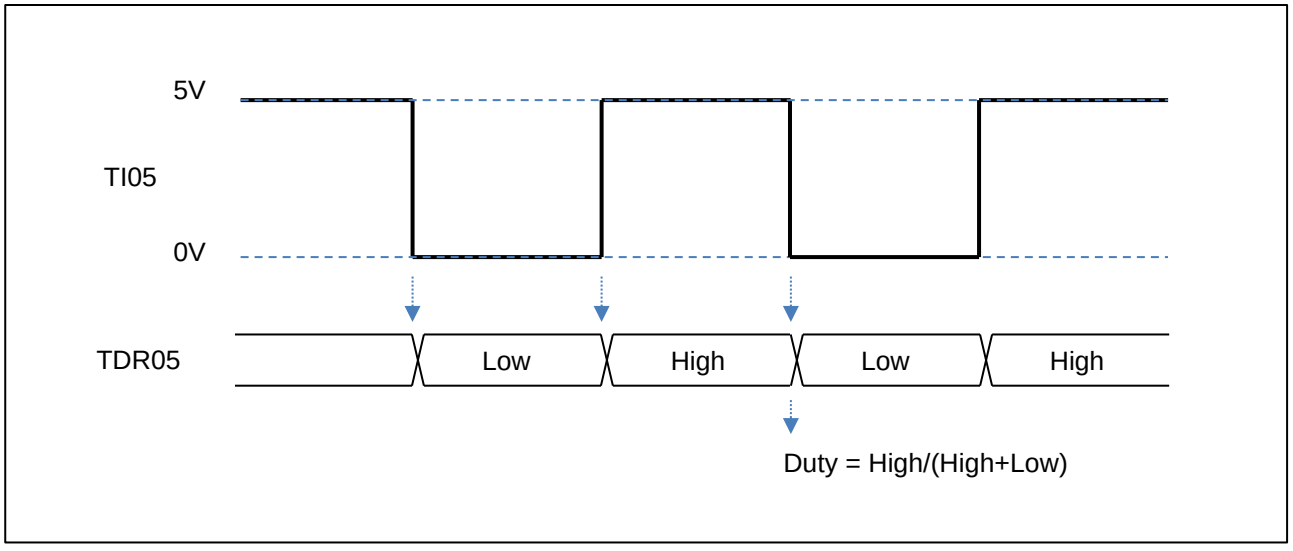


図 4-2 PWM 検出方法

(2)　0%, 100%の検出

入力パルス間隔測定を利用する場合、入力電圧が H 固定、または L 固定のときにエッジ検出が発生しないため、0%と 100%の検出ができません。そのため、エッジ検出による測定とは別に、カウンタスタートおよび最後のエッジ検出から一定時間以内に次のエッジが検出されない場合に 0%もしくは 100%デューティであることと判別するタイムアウトのイベントを作成します。

エッジ検出時にこのタイマは毎回リセットされ、1 度イベントを生成すると再度エッジ検出するまでは再開されません。

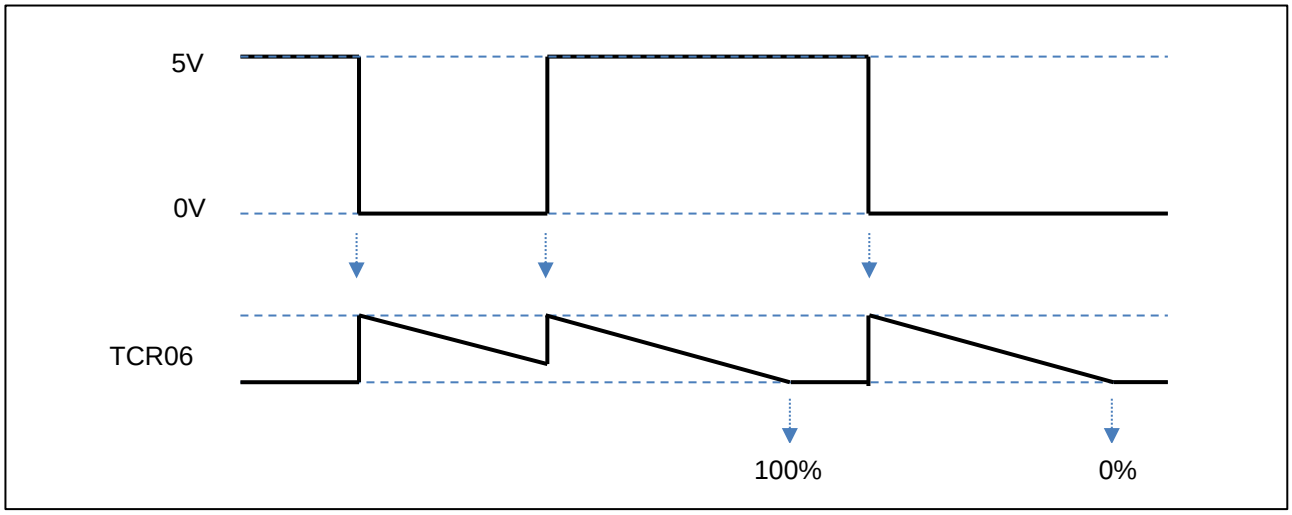


図 4-3 デューティ 0%、100%検出方法

(3)　タイマカウンタ設定

TI05 への入力周波数の対応は 10 [Hz] ～ 1000 [Hz]とします。

本条件を満たすため、サンプルプログラムではカウントソースおよび、カウンタ設定値を以下のように設定します。

表 4-1 PWM 指令値入力によるモータ動作時のタイマ仕様

タイマ	動作クロック	コンペア・キャプチャ値	分解能 [%]	100%=3000rpm 設定時の誤差量 [rpm]
入力パルス間隔測定	62.5kHz (40MHz/2^6)	1 [ms] (625) @ 1kHz	0.16	4.8
		100 [ms] (62,500) @ 10Hz	0.0016	0.048
ワンカウント・タイマ	62.5kHz (40MHz/2^6)	100 [ms] (62,500)	—	—

注意：入力周波数の非対応周波数の入力（High 幅+Low 幅合計値が 1ms 未満の場合、または 100ms を超える場合）については期待するシステム動作となりません。

4.1.2　始動方法

センサレスベクトル制御の位置検出は、三相の電流値を測定しモータモデルとの電流誤差を利用することで回転子の移動量と回転速度を推定しています。モータ停止状態、または低速回転状態ではモータの回転による誘起電圧が小さいため、誤差量が十分に発生せず推定ができません。

サンプルソフトウェアでは、始動時に位置決め電流を印加し強制転流させることにより、モータを回転させるオープンループ制御（同期運転）を行います。

モータ・システムを SEQ_MODE_RUN 状態に変更すると、電流指令値制御は d 軸、q 軸の電流指令値をそれぞれのオープンループ制御時の印加量設定に従って増加（CCW 時は減少）させます。d 軸電流の印加量がオープンループ時 d 軸電流リクエスト値に達すると回転指令値の増減を開始します。オープンループ制御時の回転子角度は PWM のキャリア周期毎に回転指令値から算出した更新角度を加算して生成します。

位置決めにかかる時間は、d 軸電流印加速度とオープンループ時の d 軸電流指令値に依存します。

強制転流時間は内部速度指令値の速度制御量と、FOC フィードバック制御切り替え回転数の相互関係により決定します。

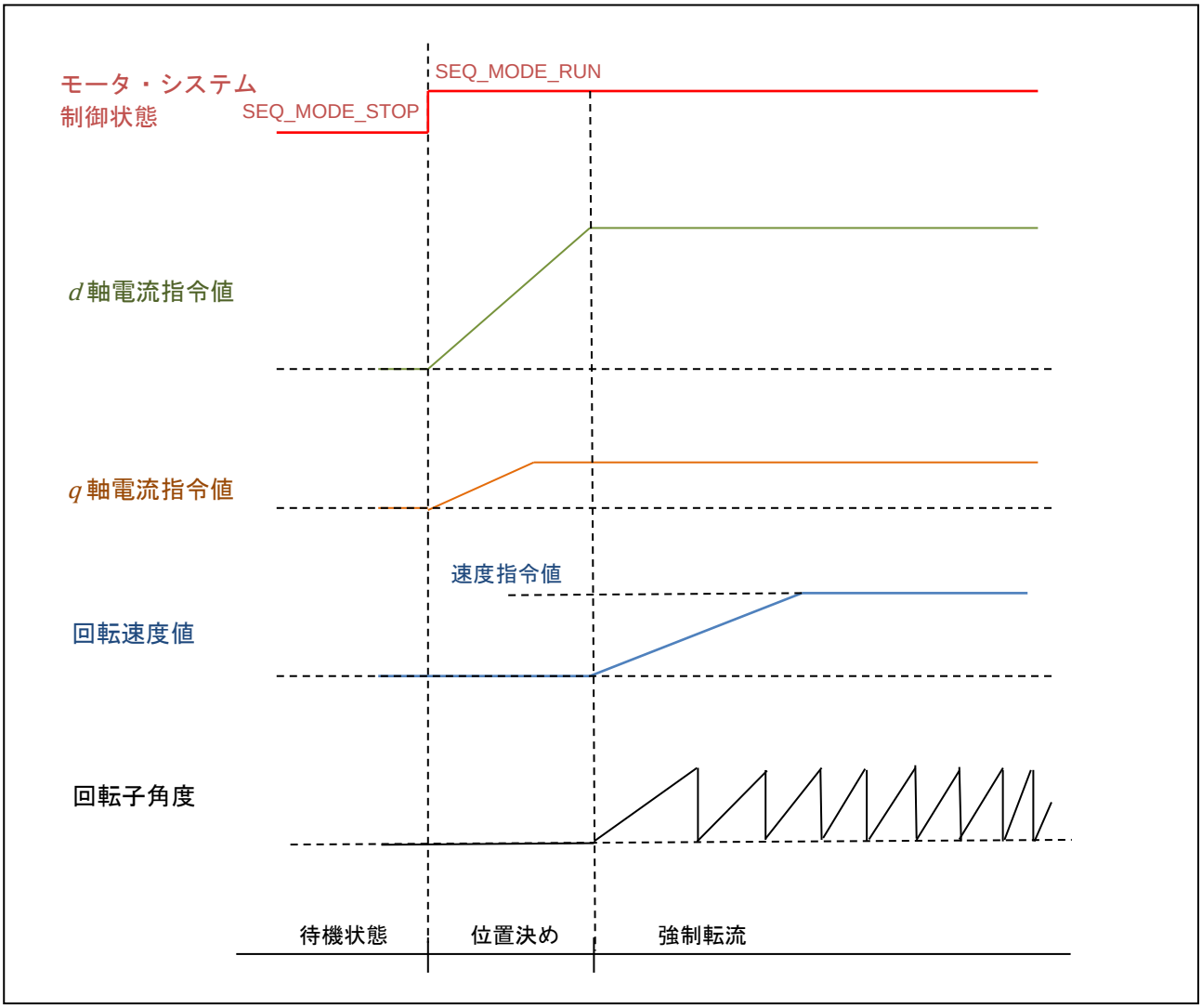


図 4-4 同期制御の動作イメージ

4.1.3　クローズドループ制御

オープンループ制御で一定回転数以上の速度になったとき、制御状態をクローズドループ制御に切り替えます。クローズドループ制御では、速度のフィードバックによって *q* 軸電流指令値を増減させて目標値へ近づけていきます。このフィードバック処理には PI 制御を使用します。*d* 軸電流は低損失・高効率な制御とするため *Id=0* 制御を行い、*d* 軸電流指令値が 0 [A]になるように下げていきます。

また、オープンループ制御からクローズドループ制御への切り替え直後は電流制御の安定待ち時間とするため一定の期間速度指令値の増減を行いません。

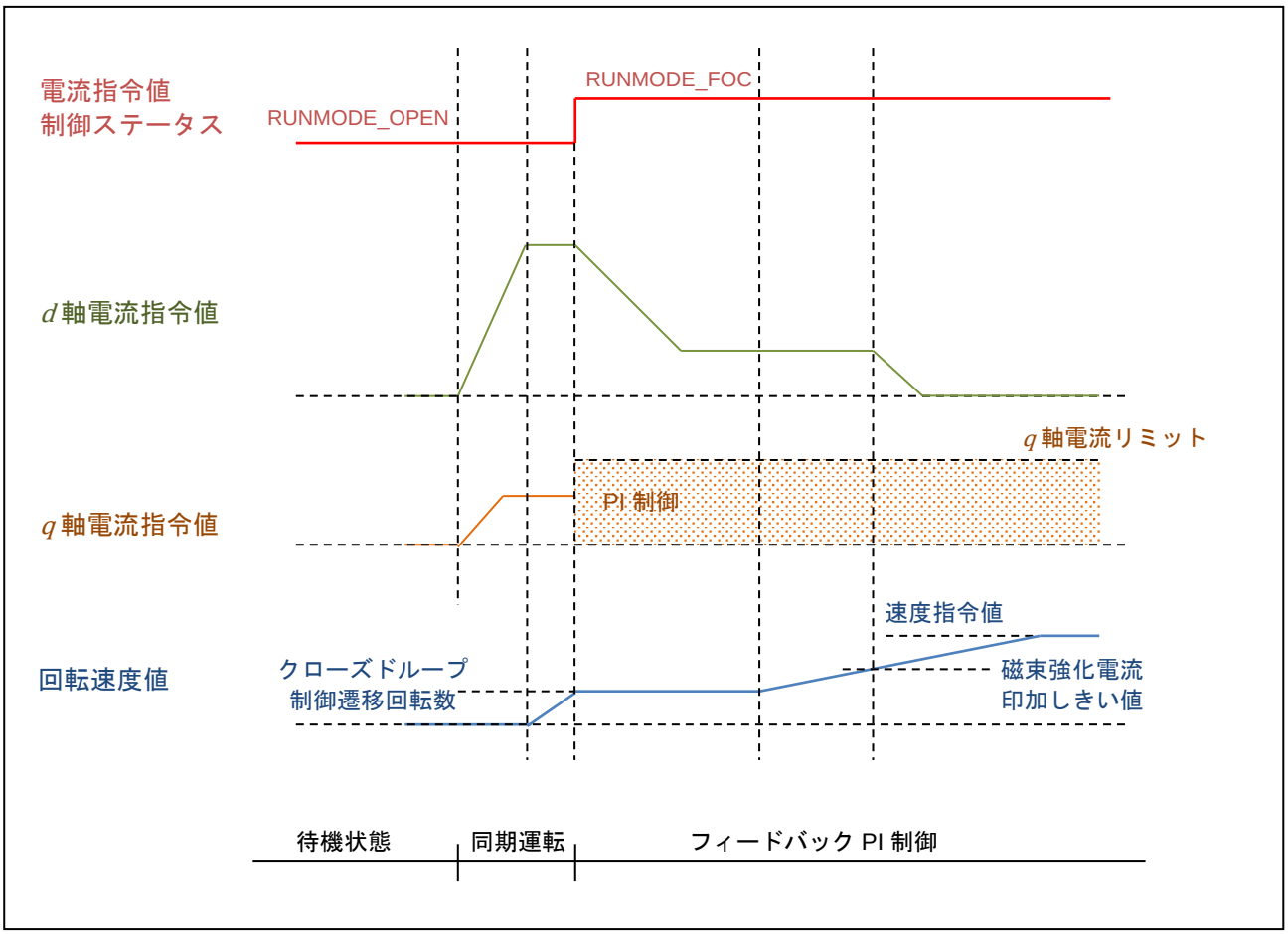


図 4-5 速度制御の動作イメージ

4.1.4　状態遷移

サンプルソフトウェアの状態遷移図を記載します。

4.1.4.1　Current Reference Controller

電流指令値制御ステータスを管理します。

電流指令制御では動作モードが RUNMODE_OPEN（オープンループ制御）と RUNMODE_FOC（FOC 電流フィードバック制御）の 2 つの状態を持ちます。

初期化後、動作モードは RUNMODE_OPEN で動作を開始します。RUNMODE_OPEN 状態でかつモータ・システムの制御状態が SEQ_MODE_RUN 状態の場合、回転を開始します。回転速度がクローズドループ制御遷移回転数に到達すると RUNMODE_FOC へ遷移します。

モータ・システムの制御状態が SEQ_MODE_STOP や、SEQ_MODE_ERROR の場合は RUNMODE_OPEN 状態となります。

電流指令値制御ステータスの状態遷移図は図 4-6 のようになります。

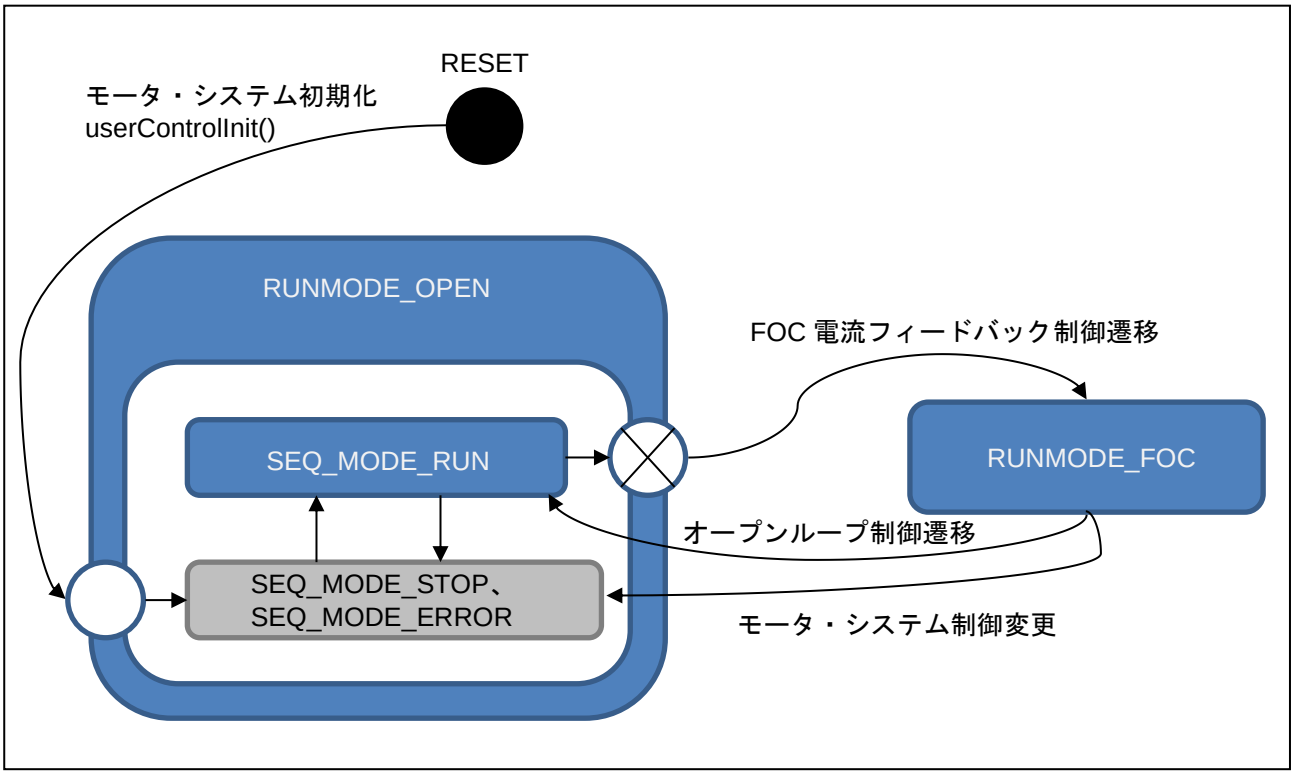


図 4-6 電流指令制御の状態遷移

注意：モータ・システム制御の状態遷移については 4.1.4.3 Sequencer Controller を参照してください。

4.1.4.2　Current Regulator

ベクトル制御の回転制御ステータスを管理します。

回転制御ステータスは初期化で電流センサの調整状態（FOC_STATE_ADJUST）に遷移します。電流センサのオフセット調整を行った後、モータ動作可能状態（FOC_STATE_ACTIVE）に遷移します。

それぞれの状態でエラーが発生した場合は、エラー状態（FOC_STATE_ERROR）へ遷移します。復帰するには再度初期化を行う必要があります。

回転制御ステータスの状態遷移図は図 4-7 のようになります。

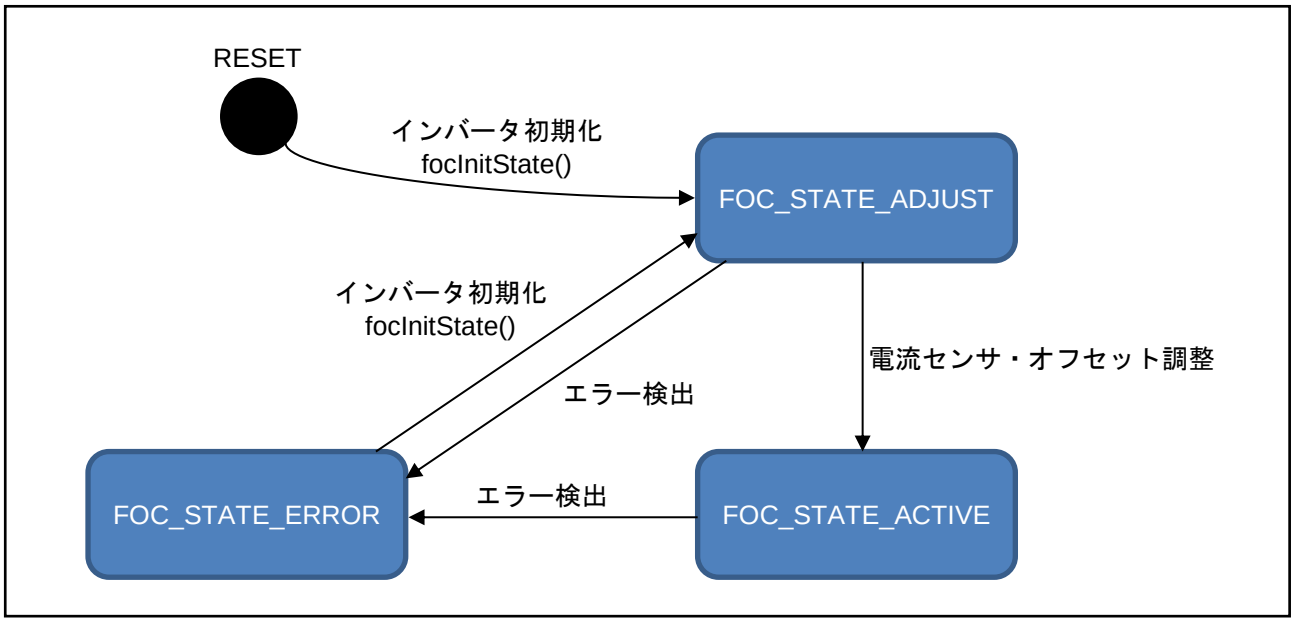


図 4-7 回転制御の状態遷移

4.1.4.3　Sequence Controller

モータ・システムの制御状態を管理します。

モータ・システムは初期化後 SEQ_MODE_STOP に設定されます。1ms 毎にモータ・システムのリクエストに応じて状態を変化させます。SEQ_MODE_RUN 状態に遷移するためにはインバータの出力を伴うため、回転制御ステータスが FOC_STATE_ACTIVE（モータ動作可能状態）である必要があります。

モータ・システムの状態遷移図は図 4-8 のようになります。

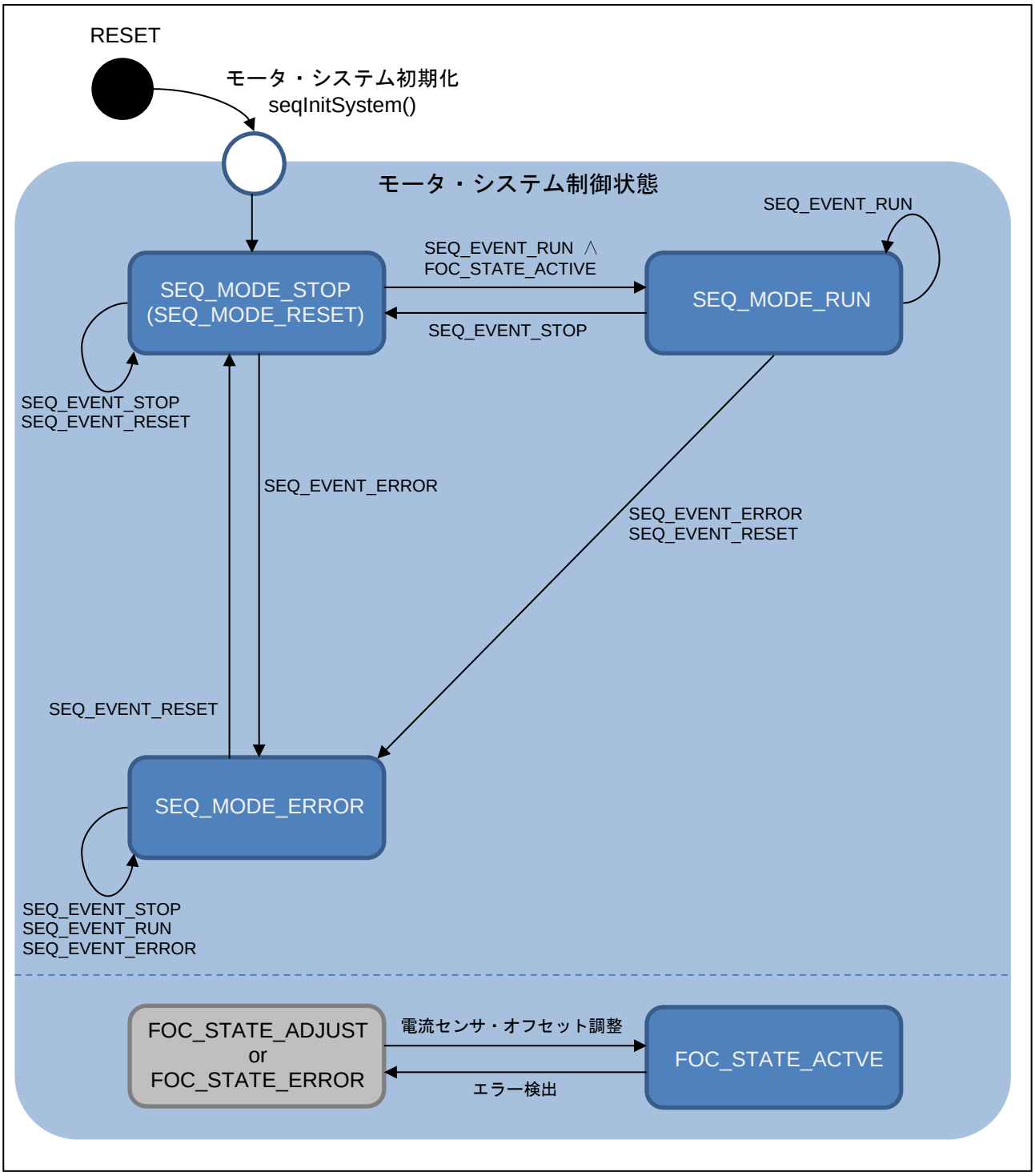


図 4-8 モータ・システムの状態遷移

注意：ベクトル制御の回転制御ステータスの状態遷移については 4.1.4.2 Current Regulator を参照してください。

4.1.5　システム保護機能

サンプルプログラムは、以下のエラー状態を持ち、それぞれの場合に停止処理を実行します。

(1)　エラー識別

- 過電流検出（H/W）

  ハードウェアからの緊急停止信号（過電流検出）により、パルス出力強制遮断を行います（CPU を介さない緊急停止）。マイコン内蔵のコンパレータを使用し過電流検出を行います。

- 過電流検出（S/W）

  PWM キャリア割り込み毎に電流値を監視します。電流検出回路の 3 相のいずれか 1 相が正または負の過電流検出しきい値を超えた場合に過電流と判断します。過電流を検出した場合、出力の停止処理を行い、エラー・コードを生成します。

  またサンプルプログラムでは電流検出しきい値を 16.97 [A]（10Arms + 20%）としています。

表 4-2 過電流検出パラメータ

パラメータ名	値	単位	説明
invOverCurLevel	16.97	A	モータ過電流検出しきい値

- 過電圧検出

  PWM キャリア割り込み毎にバス電圧検出回路を使用して V_{dc} 電源電圧の過電圧を監視します。過電圧を検出した場合、出力の停止処理を行い、エラー・コードを生成します。

表 4-3 過電圧検出パラメータ

パラメータ名	値	単位	説明
invOverVolLevel	28.0	V	V_{dc} 過電圧検出しきい値

- 低電圧検出

  PWM キャリア割り込み毎にバス電圧検出回路を使用して V_{dc} 電源電圧の低電圧を監視します。低電圧を検出した場合、出力の停止処理を行い、エラー・コードを生成します。

表 4-4 低電圧検出パラメータ

パラメータ名	値	単位	説明
invUnderVolLevel	8.0	V	V_{dc} 低電圧検出しきい値

- 基板過熱検出

  インターバル・タイマ割り込み毎に基板温度を監視します。基板過熱を検出した場合、出力の停止処理を行い、エラー・コードを生成します。また、過熱エラー検出温度より 10°C 低い温度で警告を出します。警告の解除は過熱警告検出温度より 5°C 下がった場合に解除します。

表 4-5 基板過熱検出パラメータ

パラメータ名	値	単位	説明
errTempBoardOHD	120	°C	基板過熱エラー検出温度
errTempBoardWD	110	°C	基板過熱警告検出温度
errTempBoardWC	105	°C	基板過熱警告解除温度

- モータコイルエンド過熱検出

  インターバル・タイマ割り込み毎にモータコイルエンド温度を監視します。モータコイルエンド過熱を検出した場合、出力の停止処理を行い、エラー・コードを生成します。また、過熱エラー検出温度より 10°C 低い温度で警告を出します。警告の解除は過熱警告検出温度より 5°C 下がった場合に解除します。

表 4-6 モータコイルエンド過熱検出パラメータ

パラメータ名	値	単位	説明
errTempCoilOHD	180	°C	モータコイルエンド過熱エラー検出温度
errTempCoilWD	170	°C	モータコイルエンド過熱警告検出温度
errTempCoilWC	165	°C	モータコイルエンド過熱警告解除温度

- 加速度、モータロック検出

  推定速度を使用し脱調を監視します。制御範囲を超える異常回転時に脱調と判断し停止処理を行います。高回転では過速度検出回転数を超える速度となった場合に停止処理を行います。低回転ではモータロック検出回転数が 1 秒以上断続的に続く場合にエラーと判断して停止処理を行います。

表 4-7 モータロック検出パラメータ

パラメータ名	値	単位	説明
errRpmOsd	5000	rpm	過速度検出回転数
errRpmUsd	150	rpm	モータロック検出回転数

(2)　エラー番号

駆動中にエラーを検出した場合、発生したエラーを示す番号を errErrorStatus に格納します。
errErrorStatus は 16bit の変数であり、各ビットの示す意味を図 4-9、表 4-8 に記載します。

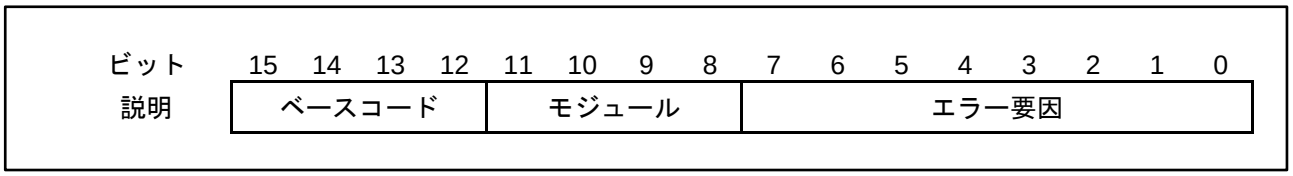


図 4-9 ビット割り当て

表 4-8 エラー番号

項目	値	概要
ベースコード	0x8	Warning code
	0xC	Error code
モジュール	0x0	MCU
	0x1	INV
	0x8	FOC
	0x9	APP
エラー要因	0x00	過電流検出 ※モジュールが MCU の場合のみ、初期化エラー
	0x10	過電圧検出
	0x11	低電圧検出
	0x20	過熱検出
	0x30	過速度検出
	0x31	モータロック検出
	0x80	状態遷移エラー
	0xFF	UNKNOWN

4.2　システム・リソース

4.2.1　割り込み

本制御プログラムで使用する割り込みの一覧を表 4-9 に示します。

表 4-9 割り込み機能一覧

割り込み	割り込みハンドラ	割り込み条件	主な機能
A/D 変換終了割り込み（DTC 転送完了）(INTAD)	r_adc_interrupt ()	DTC による 2 回の A/D 変換結果格納時（キャリア周期の 2 回に 1 回の頻度で割り込みが発生）	• 間引き済みキャリア周期処理呼び出し • ユーザ処理 • 電流制御ループ • エラー監視
インターバル・タイマ割り込み (INTTM01)	intIntervalTimerInterrupt()	1 [ms] (1 [kHz])	• ユーザ処理 • 速度制御 • 電流制御 • エラー監視
キャプチャ・タイマ割り込み (INTTM05)	r_tau0_channel5_interrupt()	TI05 入力信号のエッジ検出	PWM 信号入力制御
ワンカウント・タイマ割り込み (INTTM06)	r_tau0_channel6_interrupt()	100 [ms] (10 [Hz])	最後の INTTM05 から 100ms 経過後に発生

(1)　割り込み優先レベル

対象マイコンは割り込み優先レベルを 4 つ持ち、多重割り込み設定との組み合わせで最大 5 つの優先度が利用できます。本システムで利用する各種割り込みタスクの優先レベルは以下の通りです。

表 4-10 割り込み優先レベル設定

割り込みレベル	割り込みタスク	備考
0 (Disable Interrupt)	該当なし	多重割り込み不可
0 (Enable Interrupt)	該当なし	Lv0 割り込みであれば多重割り込み可
1 (Enable Interrupt)	INTAD	Lv0 割り込みであれば多重割り込み可
2 (Enable Interrupt)	INTTM01	Lv0–1 割り込みであれば多重割り込み可
3 (Enable Interrupt)	INTTM05, INTTM06	Lv0–2 割り込みであれば多重割り込み可

(2)　レジスタバンク設定

レジスタバンクを使用することで割り込み発生時のレジスタ退避時間を短縮し処理速度を向上させることができます。各割り込みタスクのレジスタバンク設定は以下の通りです。

表 4-11 レジスタバンク設定

割り込みタスク	バンク設定	説明
INTAD	RB1	RB1（レジスタバンク 1）を他で使用しないこと
INTTM01	指定なし（RB0）	デフォルト動作
INTTM05	指定なし（RB0）	デフォルト動作
INTTM06	指定なし（RB0）	デフォルト動作

4.2.2　PWM 出力部

本制御プログラムで使用する PWM 出力部の一覧を次に示します。

表 4-12 PWM 出力機能

入出力	出力端子	機　能	備　考
出力	TRDIOB0 / P11	U 相上アームモータ制御信号 PWM 出力（U_P）	論理設定は “High” アクティブ
	TRDIOD0 / P12	U 相下アームモータ制御信号 PWM 出力（U_N）	
	TRDIOA1 / P15	V 相上アームモータ制御信号 PWM 出力（V_P）	
	TRDIOC1 / P16	V 相下アームモータ制御信号 PWM 出力（V_N）	
	TRDIOB1 / P17	W 相上アームモータ制御信号 PWM 出力（W_P）	
	TRDIOD1 / P30	W 相下アームモータ制御信号 PWM 出力（W_N）	

4.3　関数仕様

サンプルプログラムの関数一覧を表 4-13 に示します。

表 4-13 関数一覧 (1/9)

関数仕様	処理概要
ファイル：control_math.c	
関数名：ctrlPiControl() 入力：pi_prm_t * obj 出力：int16_t ret_value	PI 制御 obj：PI 制御対象 ret_value：PI 制御結果
関数名：ctrlLimInt16() 入力：int16_t src int16_t limit 出力：int16_t ret_value	リミット制御 src：制御値 limit：制限値 ret_value：リミット制御結果
ファイル：error_management.c	
関数名：errClearStatus() 入力：なし 出力：なし	エラー・ステータスのクリア
関数名：errCheckError() 入力：なし 出力：なし	• エラー・チェック • 警告の解除チェック • 警告の発生チェック • エラー発生時処理 • モータ・システム更新
ファイル：foc_current_ref_ctrl.c	
関数名：crefFocControlTimer() 入力：なし 出力：なし	電流指令値制御 オープンループとクローズドループの制御モードを判別し関数呼び出し
関数名：crefForcedCommutation() 入力：なし 出力：なし	強制転流制御
関数名：crefFluxCurrentRefControl() 入力：なし 出力：なし	磁束電流値制御 • d軸電流のユーザ定義処理呼び出し • d軸電流指令値の加減算
関数名：crefSpeedControl() 入力：なし 出力：なし	速度フィードバック制御による d軸電流生成

表 4-13 関数一覧 (2/9)

関数仕様	処理概要
ファイル：foc_current_regulator.c	
関数名：focInitState() 入力：なし 出力：なし	インバータ初期化 • 回転制御ステータス、エラー・ステータスのリセット • インバータの初期化 • 制御パラメータのクリア • 位置推定ライブラリのステータス初期化と再設定
関数名：focFocCarrierLoop() 入力：なし 出力：なし	電流制御ループ処理 • AAU 機能によるフィードバック演算の実施 • 位置推定演算の実施 • 非干渉項の演算 • 出力電圧の設定 • エラー・チェックの実行
関数名：focErrorCheck() 入力：なし 出力：なし	エラー・チェックの実行
関数名：focStartDrive() 入力：なし 出力：なし	• 回転制御ステータスが待機状態（FOC_STATE_ACTIVE）の場合、ゲートドライバの出力を開始 • 位置推定ライブラリのパラメータを駆動状態に更新
関数名：focStopDrive() 入力：なし 出力：なし	• 位置推定ライブラリのパラメータを停止状態に更新 • ゲートドライバの出力を停止
関数名：focGetStatus() 入力：なし 出力：uint8_t focStatus	回転制御ステータスを取得 focStatus：回転制御ステータス
関数名：focGetError() 入力：なし 出力：uint8_t focError	FOC エラー・コードを取得 focError：FOC エラー・コード
関数名：focCompensateAngle() 入力：int16_t theta int16_t angular_velocity int16_t comp_time 出力：int16_t ret_value	推定角度値、推定速度値、補正時間より補正角度値を算出 theta：推定角度値 [rad] angular_velocity：推定速度値 [rad/s] comp_time：補正時間 [s] ret_value：補正角度値 [rad]
ファイル：foc_math.c	
関数名：focCalcVqLim() 入力：int16_t vd_ref int16_t vdq_limit 出力：int16_t ret_value	q 軸電圧制限値演算 vd_ref：d 軸電圧指令値 vdq_limit：dq 軸電圧制限値 ret_value：q 軸電圧制限値
ファイル：mtr_speed_ref_ctrl.c	
関数名：spdSlopeControl() 入力：なし 出力：なし	速度指令値制御処理 • 速度指令値の上限・下限をチェック • 速度指令値にスロープ値を加算 • オープンループ・クローズドループの切り替え判定を実施
ファイル：rl78_interrupt.c	
関数名：intIntervalTimerInterrupt() 入力：なし 出力：なし	インターバル・タイマ割り込み処理 • ユーザ周期割り込み処理関数呼び出し • 速度スロープ制御関数呼び出し • 周期割り込み処理関数呼び出し • エラー検出処理関数呼び出し
関数名：intCarrierInterrupt() 入力：なし 出力：なし	キャリア周期割り込み処理（INTAD で呼び出し） • 電流制御ループ関数呼び出し • ユーザ・キャリア周期割り込み処理関数呼び出し

表 4-13 関数一覧 (3/9)

関数仕様	処理概要
ファイル：rssk_inv. c (1/2)	
関数名：invInitBoard() 入力：なし 出力：uint8_t error_type	• インバータボードの初期化 • 過電流検出チェック error_type：インバータエラー・コード
関数名：invEnableGateDriver() 入力：なし 出力：なし	ゲートドライバの出力を有効化
関数名：invDisableGateDriver() 入力：なし 出力：なし	ゲートドライバの出力を無効化
関数名：invSetUVW() 入力：int16_t ref_vu int16_t ref_vv int16_t ref_vw 出力：なし	三相 PWM 出力値更新処理 • デューティ比の算出 • 変調処理の実施 • 三相 PWM 出力値の更新 ref_vu：U 相電圧指令値 ref_vv：V 相電圧指令値 ref_vw：W 相電圧指令値
関数名：invModulation () 入力：int16_t * mu int16_t * mv int16_t * mw uint16_t mode 出力：なし	PWM 変調処理 mu：U 相電圧指令値 mv：V 相電圧指令値 mw：W 相電圧指令値 mode：変調モード（通常 or 1/2 電圧加算方式）
関数名：invGetError() 入力：なし 出力：uint8_t error_type	インバータエラー判定処理 error_type：インバータエラー・コード
関数名：invGetCurrent() 入力：なし 出力：int16_t * p_iu int16_t * p_iv int16_t * p_iw	• 三相電流値の取得 • オフセット除去 • 各相の電流値の更新 p_iu：U 相電流値 p_iv：V 相電流値 p_iw：W 相電流値
関数名：invAdjustCurrent() 入力：なし 出力：int16_t * p_iu int16_t * p_iv int16_t * p_iw	• 三相電流値の取得 • オフセット値の演算 • オフセット除去 • 各相の電流値の更新 p_iu：U 相電流値 p_iv：V 相電流値 p_iw：W 相電流値
関数名：invGetSingleShuntCurrent() 入力：uint8_t index 出力：int16_t calc_buf	単一シャント電流値の取得 index：電流値取得番号 calc_buf：変換結果
関数名：invGetVdc() 入力：なし 出力：int16_t calc_buf	V_{dc} 電圧の取得 calc_buf：電源電圧値
関数名：invGetBoardTemp() 入力：なし 出力：int16_t temp1	基板温度の取得 temp1：基板温度
関数名：invGetCoilendTemp() 入力：なし 出力：int16_t temp1	モータコイルエンド温度の取得 temp1：モータコイルエンド温度

表 4-13 関数一覧 (4/9)

関数仕様	処理概要
ファイル：rssk_inv. c (2/2)	
関数名：invThreePhaseSamplingCall() 入力：int16_t ref_u int16_t ref_v int16_t ref_w 出力：uint16_t phase_pattern	位相パターンの算出 ref_u：U 相電圧指令値 ref_v：V 相電圧指令値 ref_w：W 相電圧指令値 phase_pattern：位相パターン
ファイル：sequence. c	
関数名：seqInitSystem() 入力：なし 出力：なし	モータ・システム初期化 • シーケンスエラー・ステータスのクリア • 回転子の状態を初期化
関数名：seqExecEvent() 入力：uint8_t sys_mode_req 出力：なし	現在のモータ・システムと要求モータ・システムの組み合わせからサブタスクを実行 sys_mode_req：要求モータ・システム
関数名：seqActRun() 入力：uint8_t req_state 出力：uint8_t ret	モータ駆動開始 req_state：要求モータ・システム ret：要求モータ・システム
関数名：seqActStop() 入力：uint8_t req_state 出力：uint8_t req_state	モータ駆動停止 req_state：要求モータ・システム req_state：要求モータ・システム
関数名：seqActNone() 入力：uint8_t req_state 出力：uint8_t req_state	処理なし req_state：要求モータ・システム req_state：要求モータ・システム
関数名：seqActReset() 入力：uint8_t req_state 出力：uint8_t req_state	回転制御のパラメータを初期化 req_state：要求モータ・システム req_state：要求モータ・システム
関数名：seqActError() 入力：uint8_t req_state 出力：uint8_t req_state	モータ駆動停止 req_state：要求モータ・システム req_state：要求モータ・システム
関数名：seqGetSystemMode() 入力：なし 出力：uint8_t seqSystemMode	モータ・システムの取得 seqSystemMode：モータ・システム
ファイル：user_control. c	
関数名：userControlInit() 入力：なし 出力：なし	ユーザ・パラメータを初期化
関数名：userControlTimer() 入力：なし 出力：なし	ユーザ周期割り込み処理を実施
関数名：userControlCarrier() 入力：なし 出力：なし	ユーザ・キャリア周期割り込み処理を実施
関数名：userIdReferenceControl() 入力：なし 出力：なし	*d* 軸電流制御方式を選択
関数名：userErrorDetected() 入力：uint8_t * const sys_mode uint16_t const err_status 出力：なし	ユーザ・エラー検出処理を実施 sys_mode：モータ・システム err_status：エラー・ステータス
関数名：userWarningDetected() 入力：uint8_t * const sys_mode uint16_t const err_status 出力：なし	ユーザ・ワーニング検出処理を実施 sys_mode：モータ・システム err_status：エラー・ステータス

表 4-13 関数一覧 (5/9)

関数仕様	処理概要
ファイル：r_cg_aau.c	
関数名：R_AAU_Create() 入力：なし 出力：なし	AAU 機能の初期化
関数名：R_AAU_MotorPI_Set_Reference() 入力：int16_t id_ref int16_t iq_ref 出力：なし	AAU 機能の PI 制御目標値を設定 id_ref：d 軸目標値設定 iq_ref：q 軸目標値設定
関数名：R_AAU_MotorPI_Set_Params() 入力：float kp_d float ki_d float kp_q float ki_q int16_t i_limit int16_t pi_limit 出力：なし	AAU 機能の PI 制御パラメータを設定 kp_d：d 軸比例係数 ki_d：d 軸積分係数 kp_q：q 軸比例係数 ki_q：q 軸成分係数 i_limit：積分項 limit pi_limit：PI 制御出力 limit
関数名：R_AAU_MotorPI_Set_Integral() 入力：int16_t id_buf int16_t iq_buf 出力：なし	AAU 機能の PI 制御積分項を更新 id_buf：d 軸積分値 iq_buf：q 軸積分値
関数名：R_AAU_UW2DQ_withPI() 入力：int16_t src_iu int16_t src_iw int16_t theta 出力：int16_t * p_dst_id int16_t * p_dst_iq int16_t * p_dst_vd int16_t * p_dst_vq	• Clarke & Park 変換の実施 • 電流 PI 制御演算 src_iu：U 相電流値 [A] src_iw：W 相電流値 [A] theta：角度値 [rad] p_dst_id：d 軸電流値格納アドレス p_dst_iq：q 軸電流値格納アドレス p_dst_vd：d 軸電圧値格納アドレス p_dst_vq：q 軸電圧値格納アドレス
関数名：R_AAU_DQ2UVW() 入力：int16_t src_vd int16_t src_vq int16_t theta 出力：int16_t * p_dst_vu int16_t * p_dst_vv int16_t * p_dst_vw	I-Park & I-Clarke 変換の実施 src_vd：d 軸電圧値 [V] src_vq：q 軸電圧値 [V] theta：角度値 [rad] p_dst_vu：U 相電圧値格納アドレス p_dst_vv：V 相電圧値格納アドレス p_dst_vw：W 相電圧値格納アドレス
ファイル：r_cg_adc_user.c	
関数名：r_adc_interrupt() 入力：なし 出力：なし	INTAD 割り込みハンドラ 間引き済みキャリア周期処理呼び出し

表 4-13 関数一覧 (6/9)

関数仕様	処理概要
ファイル：r_cg_adc. c	
関数名：R_ADC_Create() 入力：なし 出力：なし	ADC 機能の初期化
関数名：R_ADC_Start() 入力：なし 出力：なし	ADC 機能の起動許可
関数名：R_ADC_Get_Result() 入力：uint8_t channel 出力：uint16_t * const buffer	ADC 機能の変換結果取得 channel：取得チャネル buffer：変換結果格納アドレス
ファイル：r_cg_cgc.c	
関数名：R_CGC_Create() 入力：なし 出力：なし	クロック生成回路の初期化
ファイル：r_cg_comp.c	
関数名：R_COMP_Create() 入力：なし 出力：なし	コンパレータ機能の初期化
関数名：R_COMP0_Start() 入力：なし 出力：なし	コンパレータ機能の動作開始
ファイル：r_cg_dac. c	
関数名：R_DAC_Create() 入力：なし 出力：なし	DAC 機能の初期化
関数名：R_DAC0_Start() 入力：なし 出力：なし	DAC 機能の動作開始
ファイル：r_cg_dtc.c	
関数名：R_DTC_Create() 入力：なし 出力：なし	DTC 機能の初期化
関数名：R_DTCD2_Start() 入力：なし 出力：なし	DTC 許可（起動要因：A/D 変換完了）
関数名：r_dtcd2_get_shunt_buff() 入力：uint8_t index 出力：uint16_t addr_data	DTC 転送結果（A/D 変換値）取得 index：取得データ番号 addr_data：データ格納先アドレス
ファイル：r_cg_port.c	
関数名：R_PORT_Create() 入力：なし 出力：なし	端子入出力の初期化
関数名：R_PORT140_Set() 入力：uint8_t level 出力：なし	P14. 0 のポート操作 level：ポート出力レベル

表 4-13 関数一覧 (7/9)

関数仕様	処理概要
ファイル：r_cg_timer_rde. c	
関数名：R_TMR_RD0_Create() 入力：なし 出力：なし	タイマ RDe 機能の初期化
関数名：R_TMR_RD0_Start() 入力：なし 出力：なし	タイマ RDe 機能のカウント開始
関数名：R_TMR_RD0_Stop() 入力：なし 出力：なし	タイマ RDe 機能のカウント停止
関数名：R_TMR_RD0_Reset() 入力：なし 出力：なし	タイマ RDe 機能のリセット
関数名：R_TMR_RD0_Enable_Output() 入力：なし 出力：なし	タイマ RDe 機能の出力許可
関数名：R_TMR_RD0_Disable_Output() 入力：なし 出力：なし	タイマ RDe 機能の出力禁止
関数名：R_TMR_RD0_Set_Duty() 入力：int16_t pwm1 int16_t pwm2 int16_t pwm3 出力：uint8_t ret	三相のデューティ比を設定 pwm1：PWM1 のデューティ比 pwm2：PWM2 のデューティ比 pwm3：PWM3 のデューティ比 ret：書き込み成否（0：成功、1：失敗）
関数名：R_TMR_RD0_Get_Cutoff_Status() 入力：なし 出力：uint8_t ret	PWMOPA の強制遮断状態の取得 ret：強制遮断状態
関数名：R_TMR_RD0_Release_Cutoff() 入力：なし 出力：uint8_t ret	PWMOPA の強制遮断状態の解除 ret：解除成否
ファイル：r_cg_timer_user. c	
関数名：r_tau0_channel1_interrupt() 入力：なし 出力：なし	タイマ・アレイ・ユニット 0 チャネル 1 割り込みハンドラ
関数名：r_tau0_channel5_interrupt() 入力：なし 出力：なし	タイマ・アレイ・ユニット 0 チャネル 5 割り込みハンドラ
関数名：r_tau0_channel6_interrupt() 入力：なし 出力：なし	タイマ・アレイ・ユニット 0 チャネル 6 割り込みハンドラ
関数名：r_tau0_channel5_get_duty() 入力：なし 出力：uint16_t g_tau0_ch5_duty	PWM で与えられたデューティ比の取得 g_tau0_ch5_duty：デューティ比

表 4-13 関数一覧 (8/9)

関数仕様	処理概要
ファイル：r_cg_timer. c	
関数名：R_TAU0_Create() 入力：なし 出力：なし	タイマ・アレイ・ユニット 0 機能の初期化
関数名：R_TAU0_Channel1_Start() 入力：なし 出力：なし	タイマ・アレイ・ユニット 0 機能チャネル 1 の カウント開始
関数名：R_TAU0_Channel1_Stop() 入力：なし 出力：なし	タイマ・アレイ・ユニット 0 機能チャネル 1 の カウント停止
関数名：R_TAU0_Channel5_Start() 入力：なし 出力：なし	タイマ・アレイ・ユニット 0 機能チャネル 5 の カウント開始
関数名：R_TAU0_Channel6_Start() 入力：なし 出力：なし	タイマ・アレイ・ユニット 0 機能チャネル 6 の カウント開始
ファイル：r_cg_wdt. c	
関数名：R_WDT_Create() 入力：なし 出力：なし	WDT 機能の初期化
関数名：R_WDT_Restart() 入力：なし 出力：なし	WDT 機能のカウンタリセット
ファイル：r_main. c	
関数名：main() 入力：なし 出力：なし	メイン関数
関数名：R_MAIN_UserInit() 入力：なし 出力：なし	各種初期化関数呼び出し
ファイル：r_systeminit. c	
関数名：R_Systeminit() 入力：なし 出力：なし	各種マクロの初期化
関数名：hdwinit() 入力：なし 出力：なし	ハードウェア設定値の初期化 注意. 本関数はユーザ・ソフトウェアの初期化処理で呼び出してください。

表 4-13 関数一覧 (9/9)

関数仕様	処理概要
ファイル：r_fixed_point_math.lib	
関数名：R_SinCos() 入力：signed short sc_rad signed short * p_sin signed short * p_cos 出力：なし	Sin, Cos 計算処理 sc_rad：スケーリングされたラジアン角データ p_sin：Sin 値アドレス・ポインタ p_cos：Cos 値アドレス・ポインタ
関数名：R_MTR_AngleEstInit() 入力：est_configure_t * p_est_conf 出力：signed short	位置推定ライブラリ初期化処理 p_est_conf：初期化パラメータ・アドレス・ポインタ （0：初期化成功、1：初期化失敗）
関数名：R_MTR_AngleEstReset() 入力：なし 出力：なし	位置推定ライブラリ・リセット処理
関数名：R_MTR_AngleEstExec() 入力：signed short * p_speed_rad signed short int * p_angle_rad 出力：なし	回転子角度誤差推定処理 p_speed_rad：回転速アドレス・ポインタ p_angle_rad：回転子角度アドレス・ポインタ

4.4　変数仕様

サンプルプログラムの変数一覧を表 4-14 に示します。ただし、ローカル変数は記載していません。

表 4-14 変数一覧 (1/2)

変数名	型	内容
errErrorStatus	uint16_t	エラー・ステータス
errRpmOsd	int16_t	過速度エラー検出しきい値 [rpm(機械角)]
errRpmUsd	int16_t	モータロック検出用低速しきい値 [rpm(機械角)]
errTempBoardOHD	int16_t	基板過熱エラー検出しきい値 [°C]
errTempBoardWD	int16_t	基板過熱警告検出しきい値 [°C]
errTempBoardWC	int16_t	基板過熱警告解除しきい値 [°C]
errTempCoilOHD	int16_t	モータコイルエンド過熱エラー検出しきい値 [°C]
errTempCoilWD	int16_t	モータコイルエンド過熱警告検出しきい値 [°C]
errTempCoilWC	int16_t	モータコイルエンド過熱警告解除しきい値 [°C]
errLockDetectFilterSet	uint16_t	モータロック検出用フィルタ [ms]
errLockDetectCountSet	uint16_t	モータロック検出しきい値 [ms]
mathSqrt_3d2	int16_t	$\sqrt{3/2}$ の値を定義
mathSqrt_2d3	int16_t	$\sqrt{2/3}$ の値を定義
mathSqrt_3_2	int16_t	$\sqrt{3}/2$ の値を定義
mathSqrt_2	int16_t	$\sqrt{2}$ の値を定義
mathSqrt_2_2	int16_t	$\sqrt{2}/2$ の値を定義
crefRunMode	uint8_t	モータ動作モード
crefRpmRef	int16_t	速度指令値 [rpm(機械角)]
crefRpmEst	int16_t	速度推定値 [rpm(機械角)]
crefSpeedRadRef	int16_t	角速度指令値 [rad(電気角)]
crefIdRefRequest	int16_t	d 軸電流リクエスト値 [A]
crefIdSlopeUpVec32	int32_t	d 軸電流印加上昇スロープ [A/0.001s]
crefIdSlopeDownVec32	int32_t	d 軸電流印加下降スロープ [A/0.001s]
spdfRpmSlopeOl32	int32_t	オープンループ制御時加速・減速量 [rpm/0.001s]
crefIdRefOlRequest	int16_t	オープンループ d 軸電流リクエスト値 [A]
crefIdSlopeOl32	int32_t	オープンループ d 軸電流印加スロープ [A/0.001s]
crefIqRefOlRequest	int16_t	オープンループ q 軸電流リクエスト値 [A]
crefIqSlopeOl32	int32_t	オープンループ q 軸電流印加スロープ [A/0.001s]
crefPiPrmSpeed	PI_PRM_T	速度 PI 制御パラメータ構造体
crefIdRefBuffer32	int32_t	d 軸電流リクエスト値用バッファ
crefIqRefBuffer32	int32_t	q 軸電流リクエスト値用バッファ
focTimeSettingOffset	uint16_t	オフセット電流取得時間 [s/0.00005]
focTimeCountOffset	uint16_t	オフセット電流取得時間用カウンタ
focStatus	uint8_t	回転制御ステータス
focError	uint8_t	電流制御エラー・ステータス
focIdRef	int16_t	d 軸電流指令値 [A]
focIqRef	int16_t	q 軸電流指令値 [A]
focCurrentIu	int16_t	U 相電流値 [A]
focCurrentIv	int16_t	V 相電流値 [A]
focCurrentIw	int16_t	W 相電流値 [A]

表 4-14 変数一覧 (2/2)

変数名	型	内容
focCurrentId	int16_t	*d* 軸電流値 [A]
focCurrentIq	int16_t	*q* 軸電流値 [A]
focVdRef	int16_t	*d* 軸電圧指令値 [V]
focVqRef	int16_t	*q* 軸電圧指令値 [V]
focVuRef	int16_t	U 相電圧指令値 [V]
focVvRef	int16_t	V 相電圧指令値 [V]
focVwRef	int16_t	W 相電圧指令値 [V]
focVdqLimit	int16_t	*dq* 軸電圧制限値 [V]
focAngleRad	int16_t	回転子角度 [rad(電気角)]
focSpeedRad	int16_t	回転子角速度 [rad(電気角)/s]
focCompTimeAd	int16_t	電流取得角度補償時間 [s]
focCompTimePWM	int16_t	電圧出力角度補償時間 [s]
focMtrLd	int16_t	*d* 軸インダクタンス [H]
focMtrLq	int16_t	*q* 軸インダクタンス [H]
focMtrKe	int16_t	逆起電力係数 [Vs/rad]
spdRpmRefRequest	int16_t	速度リクエスト値 [rpm]
spdRpmLimitMax	int16_t	速度上限制限値 [rpm]
spdRpmLimitMin	int16_t	速度下限制限値 [rpm]
spdRpmSlopeUp32	int32_t	加速スロープ [rpm/s]
spdRpmSlopeDw32	int32_t	減速スロープ [rpm/s]
spdRpmOlToFoc	int16_t	オープンループからクローズドループ制御遷移回転数 [rpm]
spdRpmFocToOl	int16_t	クローズドループからオープンループ制御遷移回転数 [rpm]
spdCntDelayOlToFoc	uint16_t	電流制御安定待ち時間 [s/0.001]
invOverCurLevel	int16_t	過電流検出しきい値 [A]
invOverVolLevel	int16_t	過電圧検出しきい値 [V]
invUnderVolLevel	int16_t	低電圧検出しきい値 [V]
sscCurrentOffset	int16_t	オフセット電流値 [A]
invPhasePattern	uint16_t	位相パターン
userRpmEnhanced	int16_t	磁束強化電流印加しきい値 [rpm]
userIdRefEnhRequest	int16_t	磁束強化電流印加量 [A]
userThOnBoard	int16_t	回路温度 [°C]
userThCoilEnd	int16_t	モータコイルエンド温度 [°C]
seqSystemMode	uint8_t	モータ・システム制御状態
seqErrorStatus	uint8_t	モータ・システム制御エラー・ステータス
g_opsr_clear_flag	uint8_t	PWM 出力強制遮断クリアフラグ
g_tau0_ch5_width	uint32_t	TAU05 入力 PWM 信号パルス幅
g_tau0_ch5_Hwidth	uint32_t	TAU05 入力 PWM 信号 High 幅
g_tau0_ch5_Lwidth	uint32_t	TAU05 入力 PWM 信号 Low 幅
g_tau0_ch5_duty	uint16_t	TAU05 入力 PWM 信号デューティ値
g_trde_adt0_adj	uint16_t	A/D トリガ 0 遅延時間
g_trde_adt1_adj	uint16_t	A/D トリガ 1 遅延時間

4.5　マクロ定義仕様

サンプルプログラムのマクロ定義一覧を表 4-15、表 4-16 に示します。

表 4-15 マクロ定義一覧 (1/7)

マクロ名	定義値	内容
ファイル：control_parameter.h (1/2)		
CP_FREQ_SPEED_Hz	(1000)	速度制御周期 [Hz]
CP_RPM_MAX_SPEED	(3000)	回転速度指令値最大値 [rpm]
CP_RPM_MIN_SPEED	(500)	回転速度指令値最小値 [rpm]
CP_RPM_OL_TO_FOC	(300)	オープンループからクローズドループへの 制御切り替えしきい値 [rpm]
CP_RPM_FOC_TO_OL	(100)	クローズドループからオープンループへの 制御切り替えしきい値 [rpm]
CP_RPM_OSD_SPEED	(5000)	過速度エラー検出速度しきい値 [rpm]
CP_RPM_USD_SPEED	(150)	モータロック検出速度しきい値 [rpm]
CP_RPM_SLOPE_OL_REQ	(10000)	オープンループ時の加減速度 [rpm/s]
CP_ID_REF_OL_REQ	(1.02f)	オープンループ時の d 軸（位置決め） 電流指令値 [A]
CP_IQ_REF_OL_REQ	(0.30f)	オープンループ時の q 軸電流指令値 [A]
CP_ID_SLOPE_OL_REQ	(30.0f)	オープンループ時の d 軸電流スロープ [A/s]
CP_IQ_SLOPE_OL_REQ	(10.0f)	オープンループ時の q 軸電流スロープ [A/s]
CP_RPM_SLOPE_UP_REQ	(40000)	クローズドループ時の速度指令値 加速スロープ [rpm/s]
CP_RPM_SLOPE_DW_REQ	(25000)	クローズドループ時の速度指令値 減速スロープ [rpm/s]
CP_DELAY_OL_TO_FOC_s	(0.050f)	オープンループから電流制御切り替え時の 制御安定待ち時間 [s]
CP_ID_SLOPE_UP_REQ	(8.0f)	クローズドループ時の d 軸電流 上昇スロープ [A/s]
CP_ID_SLOPE_DOWN_REQ	(80.0f)	クローズドループ時の d 軸電流 下降スロープ [A/s]
CP_SPEED_PI_KP	(0.1f)	速度 PI 制御比例係数
CP_SPEED_PI_KI	(0.002f)	速度 PI 制御積分係数
CP_IQ_LIMIT	(2.88f)	q 軸電流リミット
CP_RPM_ENHANCE	(450)	磁束強化制御しきい値 [rpm]
CP_ID_REF_ENH_REQ	(0.5f)	磁束強化電流値 [A]

表 4-15 マクロ定義一覧 (2/7)

マクロ名	定義値	内容
ファイル：control_parameter.h (2/2)		
CP_RPMSLOPE_UP_DIV_FREQ	（注 1）	速度制御周期あたりのクローズドループ時の 速度指令値加速スロープ [rpm/s]
CP_RPMSLOPE_DW_DIV_FREQ	（注 2）	速度制御周期あたりのクローズドループ時の 速度指令値減速スロープ [rpm/s]
CP_RPMSLOPE_OL_DIV_FREQ	（注 3）	速度周期あたりのオープンループ時の 加減速度 [rpm/s]
CP_IDSLOPE_OL_DIV_FREQ	（注 4）	速度周期あたりのオープンループ時の d 軸電流スロープ [A/s]
CP_IQSLOPE_OL_DIV_FREQ	（注 5）	速度周期あたりのオープンループ時の q 軸電流スロープ [A/s]
CP_IDSLOPE_UP_DIV_FREQ	（注 6）	速度周期あたりのクローズドループ時の d 軸電流上昇スロープ [A/s]
CP_IDSLOPE_DW_DIV_FREQ	（注 7）	速度周期あたりのクローズドループ時の d 軸電流下降スロープ [A/s]
CP_PWM_CARRIER_Hz	(20000)	PWM キャリア周期 [Hz]
CP_TIME_OFFSET	(0.1f)	オフセット取得時間 [s]
CP_DECIMATION	(1)	キャリア割り込み間引き回数
CP_ID_PI_KP	(3.15564f)	d 軸電流 PI 制御比例係数
CP_ID_PI_KI	(0.02000ff)	d 軸電流 PI 制御積分係数
CP_IQ_PI_KP	(3.45939f)	q 軸電流 PI 制御比例係数
CP_IQ_PI_KI	(0.02200f)	q 軸電流 PI 制御積分係数
CP_THETA_EST_K	(0.331446f)	角度推定ゲイン
CP_SPEED_LPF_K	(0.070914f)	速度推定ゲイン
CP_EMF_EST_K	(0.356745f)	誘起電圧推定ゲイン
CP_CONTROL_FREQ	（注 8）	間引き後の PWM キャリア周波数 [Hz]
CP_CONTROL_INTERVAL	（注 9）	間引き後の PWM キャリア周期 [s]

【注】1. ((float)(CP_RPM_SLOPE_UP_REQ) / (float)CP_FREQ_SPEED_Hz)
2. ((float)(CP_RPM_SLOPE_DW_REQ) / (float)CP_FREQ_SPEED_Hz)
3. ((float)(CP_RPM_SLOPE_OL_REQ) / (float)CP_FREQ_SPEED_Hz)
4. ((float)(CP_ID_SLOPE_OL_REQ) / (float)CP_FREQ_SPEED_Hz)
5. ((float)(CP_IQ_SLOPE_OL_REQ) / (float)CP_FREQ_SPEED_Hz)
6. ((float)(CP_ID_SLOPE_UP_REQ) / (float)CP_FREQ_SPEED_Hz)
7. ((float)(CP_ID_SLOPE_DOWN_REQ) / (float)CP_FREQ_SPEED_Hz)
8. ((float)CP_PWM_CARRIER_Hz / (float)(CP_DECIMATION + 1))
9. ((float)(CP_DECIMATION + 1) / (float)CP_PWM_CARRIER_Hz)

表 4-15 マクロ定義一覧 (3/7)

マクロ名	定義値	内容
ファイル：error_management.h		
WAR	(0x8000U)	Warning コード定義
ERR	(0xC000U)	Error コード定義
MCU	(0x000U)	MCU モジュール定義
INV	(0x100U)	INV モジュール定義
FOC	(0x800U)	FOC モジュール定義
ERROR_NOT_OCCURRED	(0x0000U)	エラー未検出
E_MCU_INIT_ERR	(ERR + MCU + 0x00U)	初期化エラー
E_INV_OVER_CURRENT	(ERR + INV + 0x00U)	過電流エラー（H/W 検出）
E_INV_OVER_VOLTAGE	(ERR + INV + 0x10U)	過電圧エラー
E_INV_UNDER_VOLTAGE	(ERR + INV + 0x11U)	低電圧エラー
E_INV_OVER_TEMP	(ERR + INV + 0x20U)	基板過熱検出
E_FOC_OVER_CURRENT	(ERR + FOC + 0x00U)	過電流エラー（S/W 検出）
E_FOC_OVER_TEMP	(ERR + FOC + 0x20U)	モータ過熱検出
E_FOC_OVER_SPEED	(ERR + FOC + 0x30U)	過速度エラー
E_FOC_MOTOR_LOCKED	(ERR + FOC + 0x31U)	モータロック検出
E_FOC_INVALID_SEQUENCE	(ERR + FOC + 0x80U)	状態遷移エラー
E_FOC_UNKNOWN	(ERR + FOC + 0xFFU)	不明な電流制御エラー
W_INV_OVER_TEMP	(WAR + INV + 0x20U)	基板過熱警告
W_FOC_OVER_TEMP	(WAR + FOC + 0x20U)	モータ過熱警告
IS_WARNING(status) 入力：status 出力：TRUE/FALSE	(WAR == ((status) & 0xF000U))	Warning コード判定
IS_ERROR(status) 入力：status 出力：TRUE/FALSE	(ERR == ((status) & 0xF000U))	Error コード判定
ファイル：foc_current_ref_ctrl.h		
CREF_RUNMODE_OPEN	(1U)	動作モード：オープンループ
CREF_RUNMODE_CLOSE_SPD	(2U)	動作モード：クローズドループ

表 4-15 マクロ定義一覧 (4/7)

マクロ名	定義値	内容
ファイル：foc_current_regulator.h		
FOC_STATE_ADJUST	(0x00U)	電流制御状態：電流調整中
FOC_STATE_ACTIVE	(0x01U)	電流制御状態：電流制御有効
FOC_STATE_ERROR	(0xFFU)	電流制御状態：電流制御エラー
FOC_ERR_BASE	(0xE0U)	電流制御エラー・コードベース
FOC_ERR_NONE	(0x00U)	エラーなし
FOC_ERR_OCD_HW	(FOC_ERR_BASE + 0x0U)	H/W 検出過電流エラー
FOC_ERR_OCD	(FOC_ERR_BASE + 0x1U)	過電流エラー
FOC_ERR_OVD	(FOC_ERR_BASE + 0x2U)	V_{dc} 過電圧エラー
FOC_ERR_UVD	(FOC_ERR_BASE + 0x3U)	V_{dc} 低電圧エラー
FOC_ERR_SEQUENCE	(FOC_ERR_BASE + 0xDU)	状態遷移エラー
FOC_ERR_INIT	(FOC_ERR_BASE + 0xEU)	初期化エラー
FOC_ERR_UNKNOWN	(FOC_ERR_BASE + 0xFU)	不明なエラー
FOC_RPM2RADPS	（注 1）	回転速度 [rpm]から加速度 [rad/s]への変換係数
FOC_RADPS2RPM	（注 2）	加速度 [rad/s]から回転速度 [rpm]への変換係数
FOC_SUP2DQLIM	（注 3）	バス電圧から dq 軸電圧制限値への変換係数
FOC_DIV_PI	(1.0f / MATH_PI)	$1/\pi$ 定義
RECOGNIZE_SPD_LIM	(CP_RPM_OSD_SPEED)	過速度エラー検出速度しきい値 [rpm]
FOC_FP_RPM	（注 4）	スケーリング後の過速度エラー検出速度しきい値 [rpm]
FOC_FP_SPEED_RAD	（注 5）	スケーリング後の過速度エラー検出速度しきい値 [rad/s]
FOC_FP_ANGLE_RAD	(SC_GEN_FIX_PT(MATH_TWOPI))	スケーリング後の 2π 定義
FOC_FP_RPM2RADPS	(SC_GEN_FIX_PT(FOC_RPM2RADPS))	スケーリング後の回転速度 [rpm]から加速度 [rad/s]への変換係数
FOC_FP_RADPS2RPM	(SC_GEN_FIX_PT(FOC_RADPS2RPM))	スケーリング後の加速度 [rad/s]から回転速度 [rpm]への変換係数
FOC_FP_VSUP2VDQLIM	(SC_GEN_FIX_PT(FOC_SUP2DQLIM))	スケーリング後のバス電圧から *dq* 軸電圧制限値への変換係数
FOC_FP_DIV_PI	(SC_GEN_FIX_PT(FOC_DIV_PI))	スケーリング後の $1/\pi$ 定義
FOC_FP_LD	(SC_GEN_FIX_PT(MP_LD))	スケーリング後の *d* 軸インダクタンス [H]
FOC_FP_LQ	(SC_GEN_FIX_PT(MP_LQ))	スケーリング後の *q* 軸インダクタンス [H]
FOC_FP_KE	(SC_GEN_FIX_PT(MP_KE))	スケーリング後の逆起電力係数 [Vs/Rad]

【注】1. ((MATH_TWOPI * (float)MP_PP) / 60.0f)
2. (60.0f / (MATH_TWOPI * (float)MP_PP))
3. (MATH_SQRT_2_2 * INV_MAX_RATIO)
4. (SC_GEN_FIX_PT((float)(RECOGNIZE_SPD_LIM)))
5. (SC_GEN_FIX_PT((FOC_RPM2RADPS * (float)RECOGNIZE_SPD_LIM)))

表 4-15 マクロ定義一覧 (5/7)

マクロ名	定義値	内容
ファイル：foc_math.h		
MATH_PI	(3.14159265f)	円周率 π 定義
MATH_TWOPI	(2.0f * (MATH_PI))	2π 定義
MATH_SQRT_3d2	(1.224745f)	$\sqrt{3/2}$ 定義
MATH_SQRT_2d3	(0.816497f)	$\sqrt{2/3}$ 定義
MATH_SQRT_3_2	(0.866025f)	$\sqrt{3}/2$ 定義
MATH_SQRT_2	(1.414214f)	$\sqrt{2}$ 定義
MATH_SQRT_2_2	((MATH_SQRT_2) / 2.0f)	$\sqrt{2}/2$ 定義
FP_SQRT3D2	(SC_GEN_FIX_PT(MATH_SQRT_3d2))	スケーリング後の$\sqrt{3/2}$ 定義
FP_SQRT2D3	(SC_GEN_FIX_PT(MATH_SQRT_2d3))	スケーリング後の$\sqrt{2/3}$ 定義
FP_SQRT3_2	(SC_GEN_FIX_PT(MATH_SQRT_3_2))	スケーリング後の$\sqrt{3}/2$ 定義
FP_SQRT2	(SC_GEN_FIX_PT(MATH_SQRT_2))	スケーリング後の$\sqrt{2}$ 定義
FP_SQRT2_2	(SC_GEN_FIX_PT(MATH_SQRT_2_2))	スケーリング後の$\sqrt{2}/2$ 定義
ファイル：motor_parameter.h		
MP_PP	(2U)	極対数
MP_RA	(2.80f)	1 相あたりの巻線抵抗値 [Ω]
MP_LD	(0.00084150f)	d 軸インダクタンス [H]
MP_LQ	(0.00092250f)	q 軸インダクタンス [H]
MP_KE	(0.00853396f)	逆起電力係数 [Vs/Rad]
MP_MAX_TEMP	(180.0f)	モータ最大温度 [°C]
ファイル：sequence.h		
SEQ_MODE_STOP	(0U)	モータ・システム：停止
SEQ_MODE_RUN	(1U)	モータ・システム：実行
SEQ_MODE_ERROR	(2U)	モータ・システム：エラー
SEQ_SIZE_STATE	(3U)	モータ・システムサイズ
SEQ_EVENT_STOP	(0U)	モータ・システム要求：停止
SEQ_EVENT_RUN	(1U)	モータ・システム要求：実行
SEQ_EVENT_ERROR	(2U)	モータ・システム要求：エラー
SEQ_EVENT_RESET	(3U)	モータ・システム要求：リセット
SEQ_SIZE_EVENT	(4U)	モータ・システム要求サイズ
SEQ_ERR_NONE	(0x00U)	エラーなし
SEQ_ERR_UNKNOWN	(0xFFU)	不明なエラー
ファイル：sequence.c		
STATE_TABLE	表 4-16 を参照	状態遷移定義テーブル
ACTION_TABLE	表 4-16 を参照	動作定義テーブル

表 4-15 マクロ定義一覧 (6/7)

マクロ名	定義値	内容
ファイル：rssk_inv.h (1/2)		
INV_PWM_DEADTIME_s	(0.000001f)	dead time [s]
INV_SHUNT_R	(0.005f)	シャント抵抗 [Ω]
INV_AMP_GAIN	(20U)	電流シャントアンプゲイン
INV_AD2CUR	（注 1）	A/D 変換結果から電流値への変換係数
INV_AD2VPN	(65.0f / 4095.0f)	A/D 変換結果から電圧値への変換係数
INV_MAX_RATIO	（注 2）	1 キャリア周期内の dead time 以外の比率
INV_OV_LEVEL	(28.0f)	過電圧検出しきい値 [V]
INV_UV_LEVEL	(8.0f)	低電圧検出しきい値 [V]
INV_VOLTAGE_MAX	(40.0f)	最大検出可能電圧 [V]
INV_OC_LEVEL	(16.9706f)	過電流検出しきい値 [A]
INV_CURRENT_MAX	(20.0f)	最大検出可能電流 [A]
INV_CURRENT_OFFSET_K	(0.125f)	電流オフセットフィルタ係数
INV_OVERHEAT_DETECT_TEMP	(120.0f)	基板過熱検出温度 [°C]
INV_MODU_MODE_NONE	(1U)	変調制御なし
INV_MODU_MODE_CSVPWM	(2U)	中間電圧 1/2 加算方式
INV_ERR_BASE	(0xC0U)	INV エラー・コードベース
INV_ERR_NONE	(0x00U)	エラー未検出
INV_ERR_VDC_OVD	(INV_ERR_BASE + 0x0U)	過電圧検出
INV_ERR_VDC_UVD	(INV_ERR_BASE + 0x1U)	低電圧検出
INV_ERR_OCD	(INV_ERR_BASE + 0x2U)	過電流検出
INV_AD_CHANNEL_VDC	(5U)	A/D チャネル番号：Vdc
INV_AD_CHANNEL_T_RT1	(24U)	A/D チャネル番号：RT1
INV_AD_CHANNEL_T_NCT1	(4U)	A/D チャネル番号：NCT1
INV_FP_CURRENT	(SC_GEN_FIX_PT(INV_CURRENT_MAX))	スケーリング後の最大検出可能電流 [A]
INV_FP_VOLTAGE	(SC_GEN_FIX_PT(INV_VOLTAGE_MAX))	スケーリング後の最大検出可能電圧 [V]
INV_FP_AD2CUR	(SC_GEN_FIX_PT(INV_AD2CUR))	スケーリング後の A/D 変換結果から電流値への変換係数
INV_FP_AD2VPN	(SC_GEN_FIX_PT(INV_AD2VPN))	スケーリング後の A/D 変換結果から電圧値への変換係数
INV_FP_OFFSET_K	(SC_GEN_FIX_PT (INV_CURRENT_ OFFSET_K))	スケーリング後の電流オフセットフィルタ係数

【注】1. (((5.0f / (float)INV_AMP_GAIN) / INV_SHUNT_R) / 4095.0f)
2. (1.0f - (2.0f * INV_PWM_DEADTIME_s * (float)(CP_PWM_CARRIER_Hz)))

表 4-15 マクロ定義一覧 (7/7)

マクロ名	定義値	内容
ファイル：rssk_inv.h (2/2)		
INV_PHASE_UV	(0U)	位相パターン：電圧指令値が高い順に U → V → W
INV_PHASE_UW	(1U)	位相パターン：電圧指令値が高い順に U → W → V
INV_PHASE_VU	(2U)	位相パターン：電圧指令値が高い順に V → U → W
INV_PHASE_VW	(3U)	位相パターン：電圧指令値が高い順に V → W → U
INV_PHASE_WU	(4U)	位相パターン：電圧指令値が高い順に W → U → V
INV_PHASE_WV	(5U)	位相パターン：電圧指令値が高い順に W → V → U
ファイル：thermistor.h		
TH_TEMPTABLE_NTCG163JF103FT1S	表 2-7 を参照	NTCG163JF103FT1S 変換テーブル
TH_TEMPTABLE_103NT_4_R025H41G	表 2-6 を参照	103NT_4_R025H41G 変換テーブル
ファイル：user_control.c		
RPM2REF(mRpm)	（注 1）	スケーリング後の回転数指令値

【注】1.　(SC_FIXED_VAL16((mRpm), FOC_FP_RPM))

表 4-16 複数行マクロ定義

```
STATE_TABLE
{¥
/*              state       SEQ_MODE_STOP,   SEQ_MODE_RUN,    SEQ_MODE_ERROR */¥
/* request          */                                                          ¥
/* SEQ_EVENT_STOP   */{ SEQ_MODE_STOP,   SEQ_MODE_STOP,   SEQ_MODE_ERROR },¥
/* SEQ_EVENT_RUN    */{ SEQ_MODE_RUN,    SEQ_MODE_RUN,    SEQ_MODE_ERROR },¥
/* SEQ_EVENT_ERROR */{ SEQ_MODE_ERROR, SEQ_MODE_ERROR, SEQ_MODE_ERROR },¥
/* SEQ_EVENT_RESET */{ SEQ_MODE_STOP,   SEQ_MODE_ERROR, SEQ_MODE_STOP   }}
```

```
ACTION_TABLE
{¥
/*              state       SEQ_MODE_STOP,   SEQ_MODE_RUN,    SEQ_MODE_ERROR */¥
/* request          */                                                          ¥
/* SEQ_EVENT_STOP   */{ seqActNone,       seqActStop,       seqActNone       },¥
/* SEQ_EVENT_RUN    */{ seqActRun,        seqActNone,       seqActNone       },¥
/* SEQ_EVENT_ERROR */{ seqActError,      seqActError,      seqActNone       },¥
/* SEQ_EVENT_RESET */{ seqActReset,      seqActError,      seqActReset      }}
```

4.6　制御フロー

本制御プログラムのフローチャートを示します。

4.6.1　メイン関数

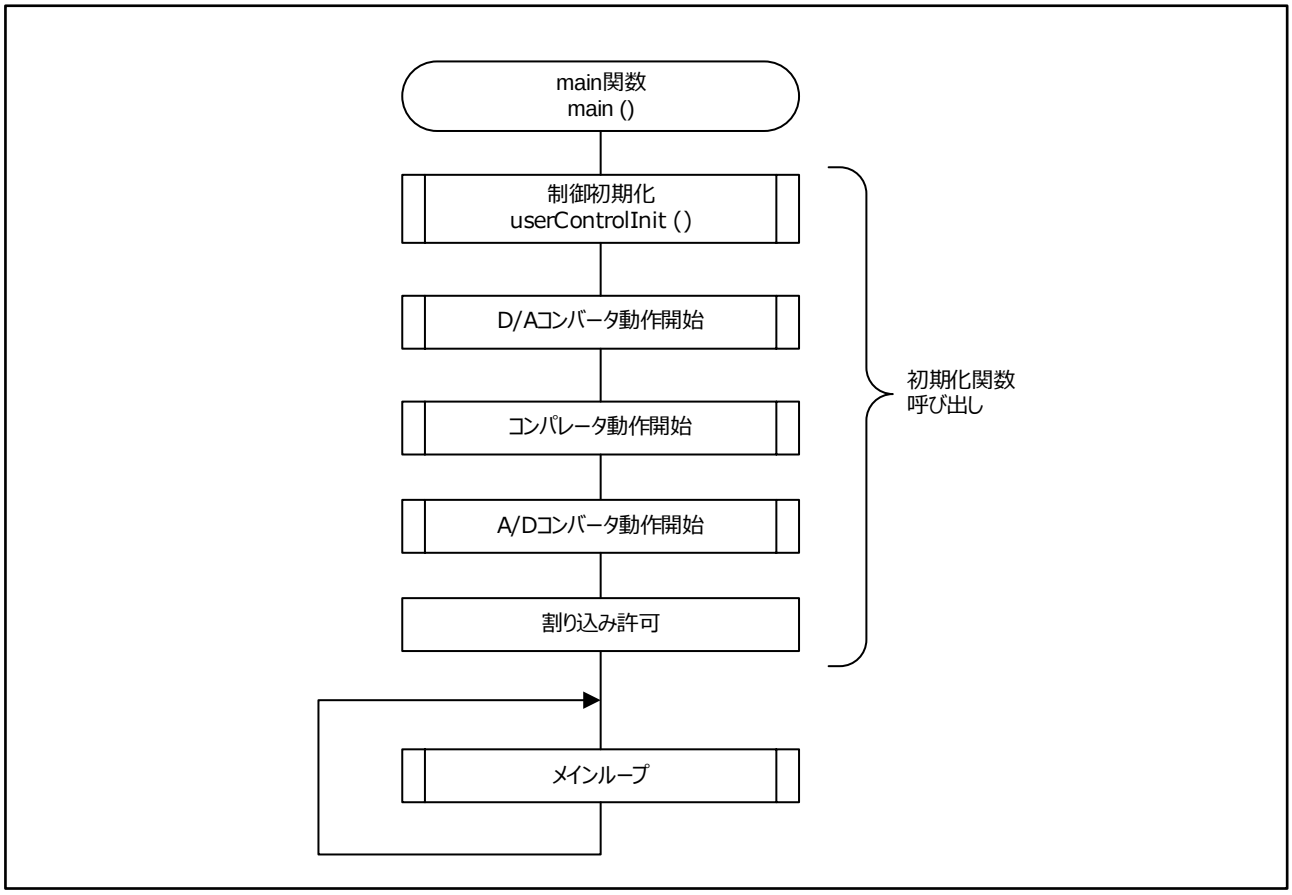


図 4-10 メイン関数

4.6.2 制御初期化関数

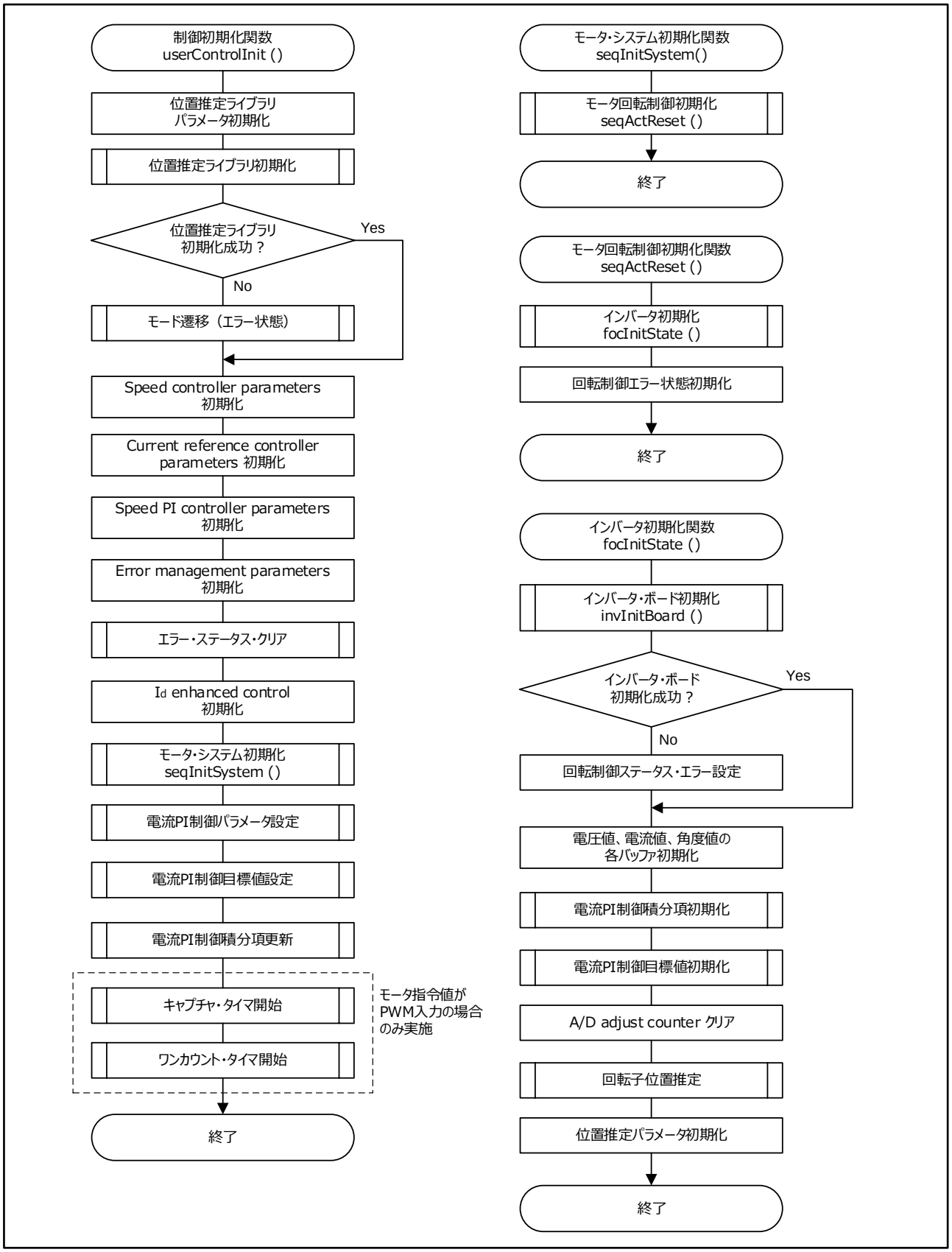


図 4-11 制御初期化関数

4.6.3　インバータ・ボード初期化

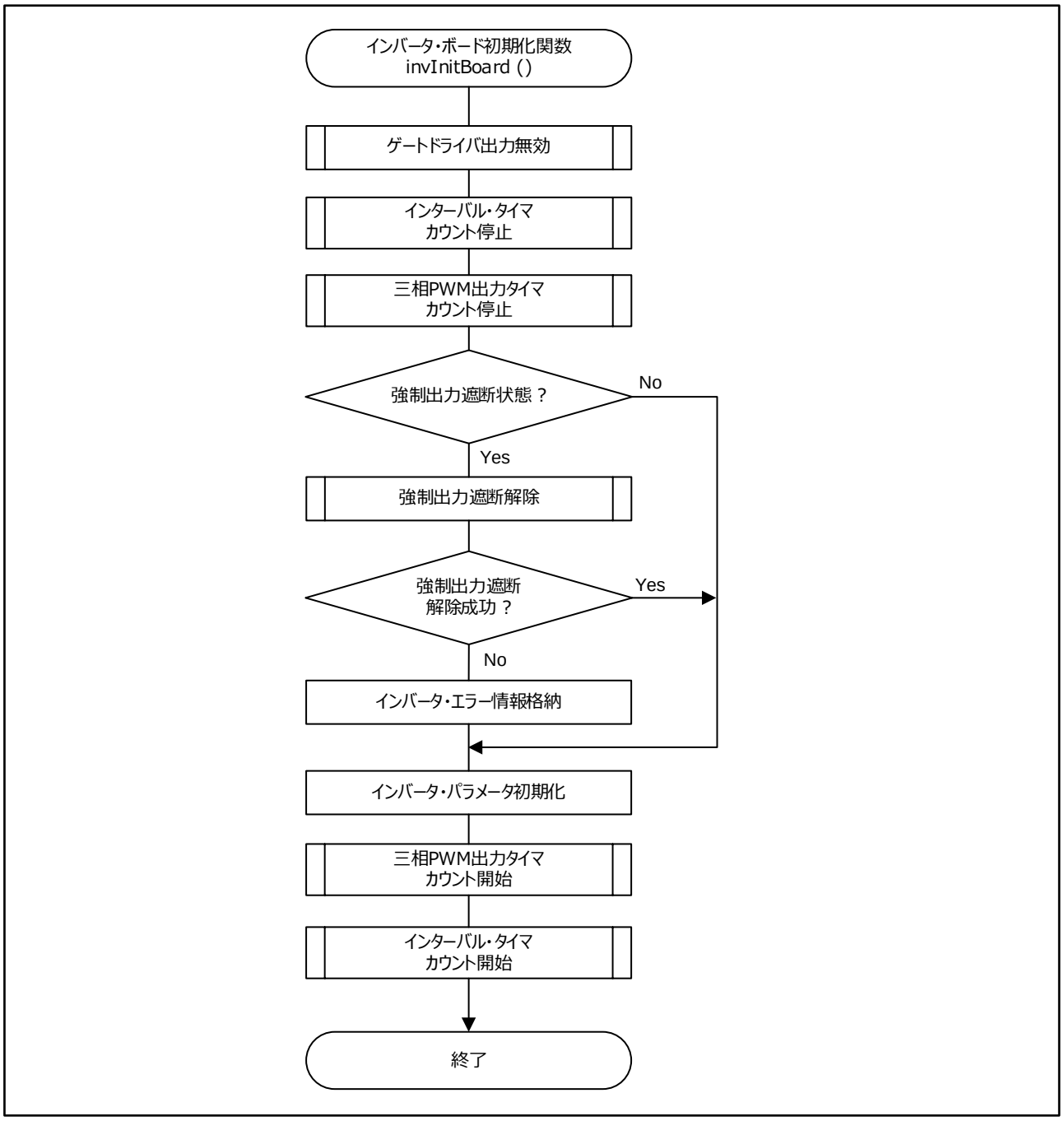


図 4-12 インバータ初期化関数

4.6.4　インターバル・タイマ割り込みハンドラ

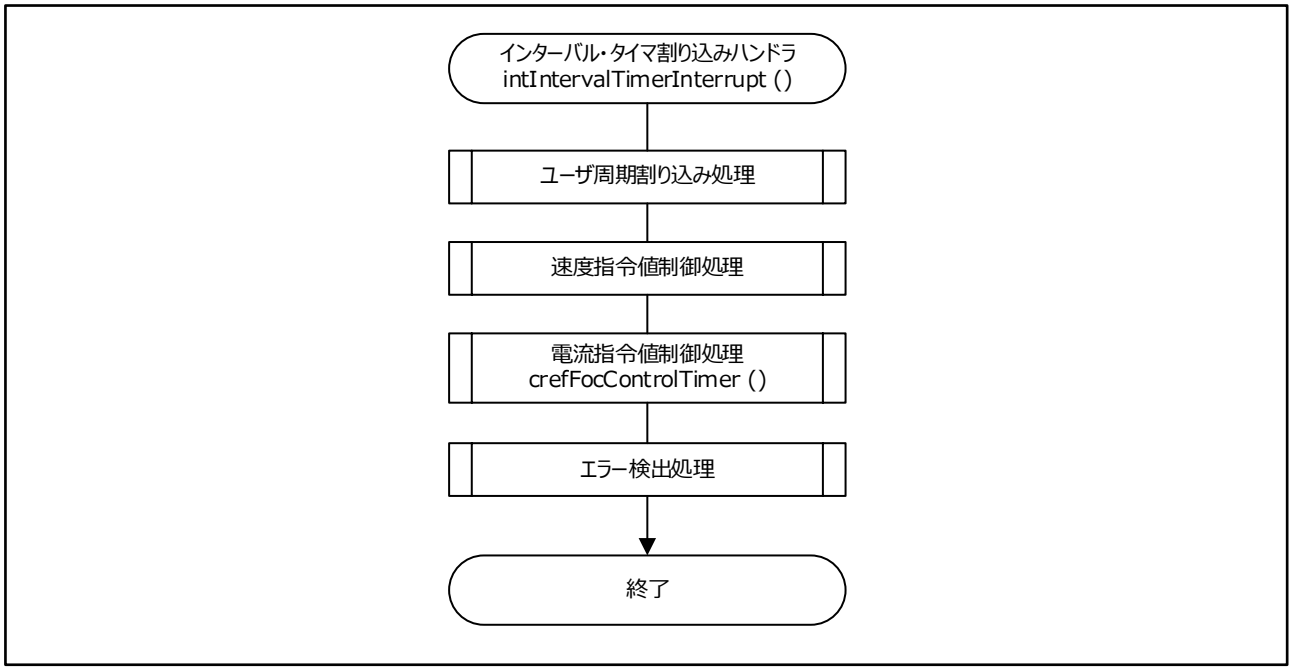


図 4-13 インターバル・タイマ割り込みハンドラ

4.6.5 電流指令値制御

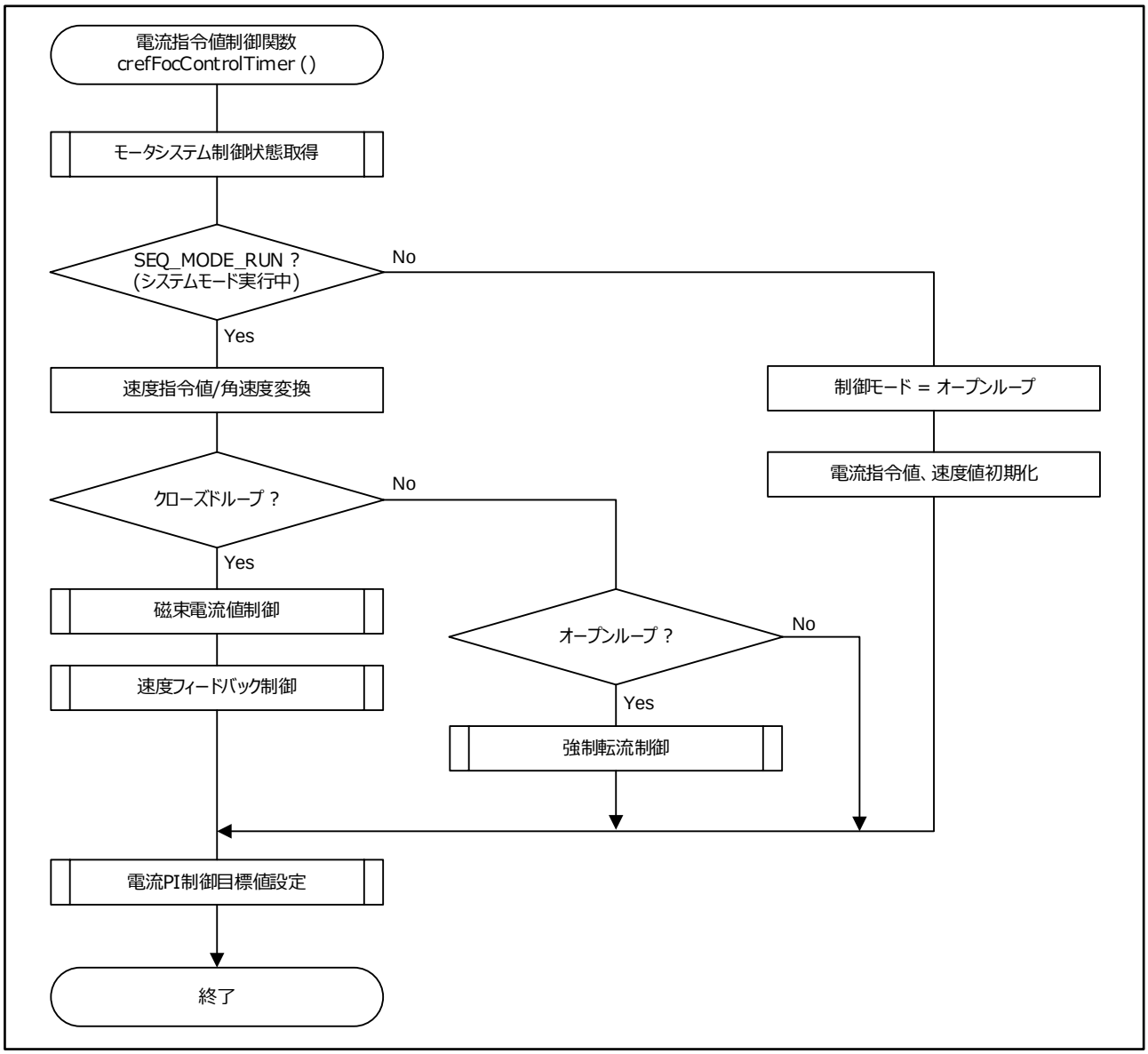


図 4-14 電流指令値制御

4.6.6　キャリア周期割り込みハンドラ

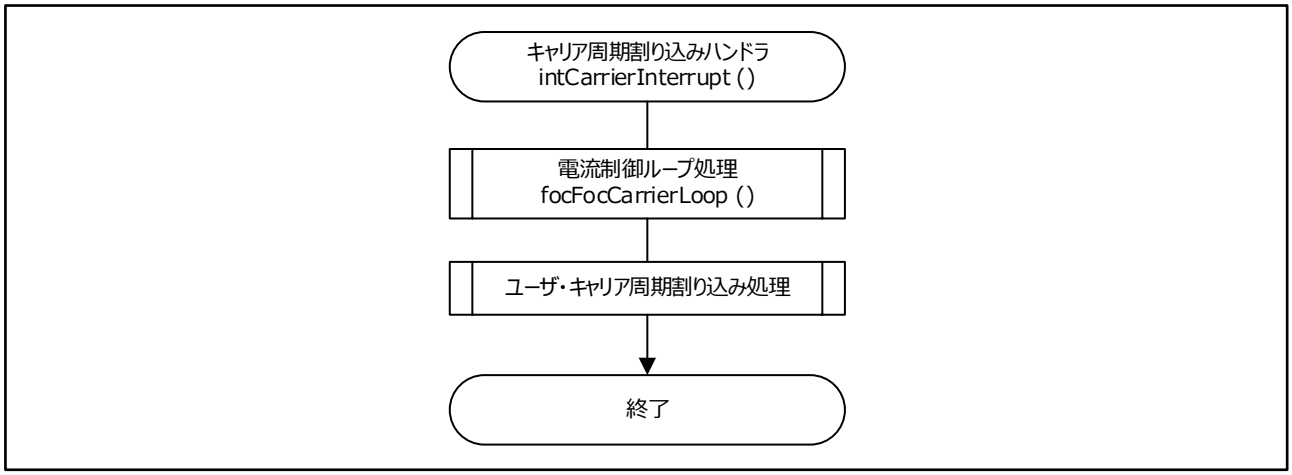


図 4-15 キャリア周期割り込みハンドラ

4.6.7 電流制御ループ処理

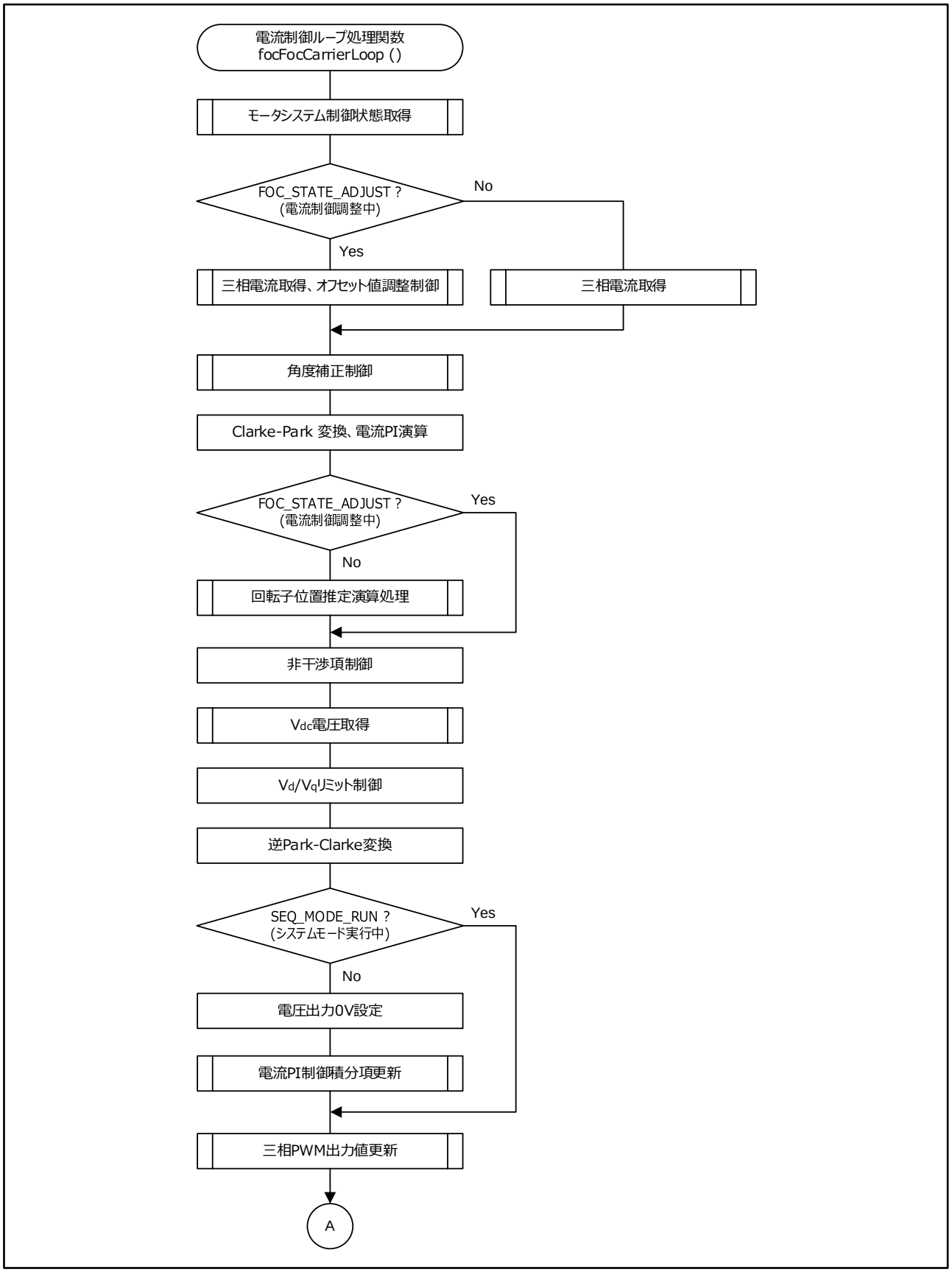


図 4-16 電流制御ループ処理 (1/2)

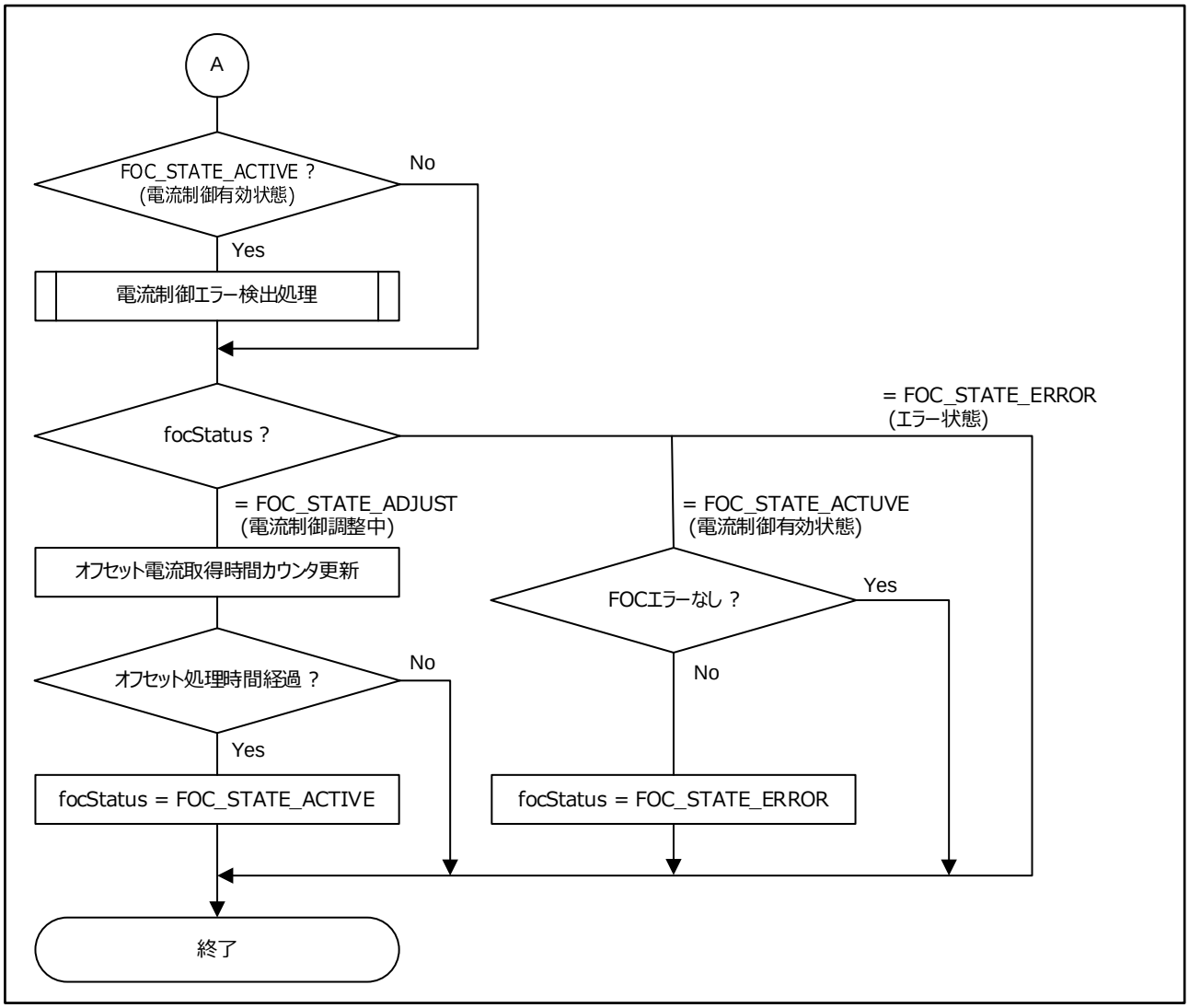


図 4-16 電流制御ループ処理 (2/2)

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2022. 9.30	–	初版発行
1.10	2023. 6.30	P.16, 17	ソフトウェアのバージョンを V.1.04 から V.1.05 に更新

		P.44	ctrlPiControl () 関数の計算式を修正
			変更前：
			obj->ibuffer = ((int32_t)refi << 16U) + (int32_t)((int16_t) obj->ibuffer);
			変更後：
			obj->ibuffer = (((int32_t)refi << 16U) & 0xFFFF0000) | (obj->ibuffer&0x0000FFFF);

		P.49	r_cg_dtc.c ファイルの DTC テーブル定義を IAR コンパイラに対応した記載に修正
		P.51	hdwinit () 関数に注記を追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の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合せください。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、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。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、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。