お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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。お客様の機器の設計において、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には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、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に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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おりますが、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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。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、一切その責任を負いません。
11. 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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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その総株主の議決権の過半数を直接または間接に保有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。

RENESAS

アプリケーション・ノート

μPD78F0714によるインバータ制御

ゼロクロス検出による120度通電方式制御編

μPD78F0714

資料番号　U17297JJ1V0AN00（第1版）
発行年月　May 2005 NS　CP(K)

〔メ　モ〕

目次要約

CMOSデバイスの一般的注意事項

① 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デバイス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。入力レベルが固定な場合はもちろん，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等が入らないようご使用ください。

② 未使用入力の処理

CMOSデバイスの未使用端子の入力レベルは固定してください。

未使用端子入力については，CMOSデバイスの入力に何も接続しない状態で動作させるのではなく，プルアップかプルダウンによって入力レベルを固定してください。また，未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③ 静電気対策

MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイス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，または導電性の緩衝材，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，MOSデバイス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。

④ 初期化以前の状態

電源投入時，MOSデバイスの初期状態は不定です。

電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

⑤ 電源投入切断順序

内部動作および外部インタフェースで異なる電源を使用するデバイスの場合，原則として内部電源を投入した後に外部電源を投入してください。切断の際には，原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により，内部素子に過電圧が印加され，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「電源投入切断シーケンス」についての記載のある製品については，その内容を守ってください。

⑥ 電源OFF時における入力信号

当該デバイスの電源がOFF状態の時に，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OFF時における入力信号」についての記載のある製品については，その内容を守ってください。

- 本資料に記載されている内容は2005年5月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 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。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。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。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関する情報を使用する場合には，お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し，当社は，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  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

  特別水準：輸送機器（自動車，電車，船舶等），交通用信号機器，防災・防犯装置，各種安全装置，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

(注)

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいう。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E　02.11

はじめに

対 象 者　このアプリケーション・ノートは，μPD78F0714の機能を理解し，それらを使用した応用システムを設計するユーザを対象とします。対象製品を次に示します。

- μPD78F0714

目　　的　このアプリケーション・ノートでは，μPD78F0714のタイマ／カウンタ機能のシステム例としてPWM出力，A/Dコンバータ入力を使用したブラシレスDCモータを使って，センサレス駆動の120度通電方式による制御をユーザに理解していただくことを目的としています。

構　　成　このアプリケーション・ノートは大きく分けて次の内容で構成しています。

- 制御方式
- ハードウエア構成
- ソフトウエア構成
- プログラム・リスト

読 み 方　このマニュアルの読者には，電気，論理回路，およびマイクロコンピュータに関する一般知識を必要とします。

ハードウエア機能の詳細（特にレジスタ機能とその設定方法など），および電気的特性を知りたいとき

→別冊のμPD78F0714　**ユーザーズ・マニュアル**（U16928J）を参照してください。

命令機能の詳細を理解しようとするとき

→別冊の78K/0**シリーズ　ユーザーズ・マニュアル　命令編**（U12326J）を参照してください。

凡　例　データ表記の重み：左が上位桁，右が下位桁

アクティブ・ロウの表記：$\overline{xxx}$（端子，信号名称に上線）

メモリ・マップのアドレス：上部-上位，下部-下位

注　：本文中に付けた注の説明

注意：気を付けて読んでいただきたい内容

備考：本文の補足説明

数の表記：2進数 … xxxxまたはxxxxB

10進数… xxxx

16進数… xxxxH

2のべき数を示す接頭語（アドレス空間，メモリ容量）：

K（キロ）　…　$2^{10} = 1024$

M（メガ）　…　$2^{20} = 1024^2$

G（ギガ）　…　$2^{30} = 1024^3$

データ・タイプ：ワード　…　32ビット

ハーフワード　…　16ビット

バイト　…　8ビット

開連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

デバイスの関連資料

資　料　名	資料番号	
	和　　文	英　　文
μPD78F0714　ユーザーズ・マニュアル	U16928J	U16928E
78K/0シリーズ　ユーザーズ・マニュアル　命令編	U12326J	U12326E
μPD78F0714によるインバータ制御　アプリケーション・ノート　ゼロクロス検出による120度通電方式制御編	このマニュアル	U17297E

注意　上記関連資料は予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

開発ツール（ソフトウエア）の資料（ユーザーズ・マニュアル）

資　料　名		資料番号	
		和　　文	英　　文
RA78K0　Ver.3.80　アセンブラ・パッケージ	操作編	U17199J	U17199E
	言語編	U17198J	U17198E
	構造化アセンブリ言語編	U17197J	U17197E
CC78K0　Ver.3.70　Cコンパイラ	操作編	U17201J	U17201E
	言語編	U17200J	U17200E
SM+　システム・シミュレータ	操作編	U17246J	U17246E
	ユーザ・オープン・インタフェース編	U17247J	U17247E
ID78K0-QB Ver.2.81　統合ディバッガ	操作編	U16996J	U16996E
PM plus Ver.5.20		U16934J	U16934E

開発ツール（ハードウエア）の資料（ユーザーズ・マニュアル）

資　料　名	資料番号	
	和　　文	英　　文
QB-78K0KX1H　インサーキット・エミュレータ	U17081J	U17081E

フラッシュ・メモリ書き込み用の資料

資　料　名	資料番号	
	和　　文	英　　文
PG-FP3　フラッシュ・メモリ・プログラマ　ユーザーズ・マニュアル	U13502J	U13502E
PG-FP4　フラッシュ・メモリ・プログラマ　ユーザーズ・マニュアル	U15260J	U15260E

その他の資料

資　料　名	資料番号	
	和　　文	英　　文
SEMICONDUCTOR SELECTION GUIDE　- Products and Packages -	X13769X	
半導体デバイス　実装マニュアル	注	
NEC半導体デバイスの品質水準	C11531J	C11531E
NEC半導体デバイスの信頼性品質管理	C10983J	C10983E
静電気放電（ESD）破壊対策ガイド	C11892J	C11892E
半導体　品質／信頼性ハンドブック	C12769J	－
マイクロコンピュータ関連製品ガイド　社外メーカ編	U11416J	－

注　「半導体デバイス実装マニュアル」のホーム・ページ参照

和文：http://www.necel.com/pkg/ja/jissou/index.html

英文：http://www.necel.com/pkg/en/mount/index.html

注意　上記関連資料は予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

目　次

第1章　制御方式

1.1　ブラシレスDCモータ制御の概要

ブラシレスDC（BLDC）モータは，固定子部分（ステータ）のコイルが発生する磁界の作用により，永久磁石でできた回転部分（ロータ）が回転します。

ステータに巻かれたコイルを一定の順に通電することで回転磁界を発生させ，その強弱，周期をマイコン制御することで，トルク応答性や回転数制御を行います。

μPD78F0714を用いたBLDCモータのセンサレス制御について説明します。

図1－3は，3相ブラシレスDCモータの回路例を示したものです。マイコン内蔵のPWM出力機能が6つのトランジスタで構成されたトランジスタ・アレイを制御し，そのトランジスタ・アレイによってモータに流れる電流を制御できます。

表1－1のように6つのトランジスタの通電パターンを制御することで，回転磁界を発生させます。

表1－1　通電パターン

通電パターン	上アーム			下アーム			通電方向
	U	V	W	$\overline{U}$	$\overline{V}$	$\overline{W}$	
<1>	アクティブ	インアクティブ	インアクティブ	インアクティブ	アクティブ	インアクティブ	U→$\overline{V}$
<2>	アクティブ	インアクティブ	インアクティブ	インアクティブ	インアクティブ	アクティブ	U→$\overline{W}$
<3>	インアクティブ	アクティブ	インアクティブ	インアクティブ	インアクティブ	アクティブ	V→$\overline{W}$
<4>	インアクティブ	アクティブ	インアクティブ	アクティブ	インアクティブ	インアクティブ	V→$\overline{U}$
<5>	インアクティブ	インアクティブ	アクティブ	アクティブ	インアクティブ	インアクティブ	W→$\overline{U}$
<6>	インアクティブ	インアクティブ	アクティブ	インアクティブ	アクティブ	インアクティブ	W→$\overline{V}$

図1－1　3相DCモータ電圧波形

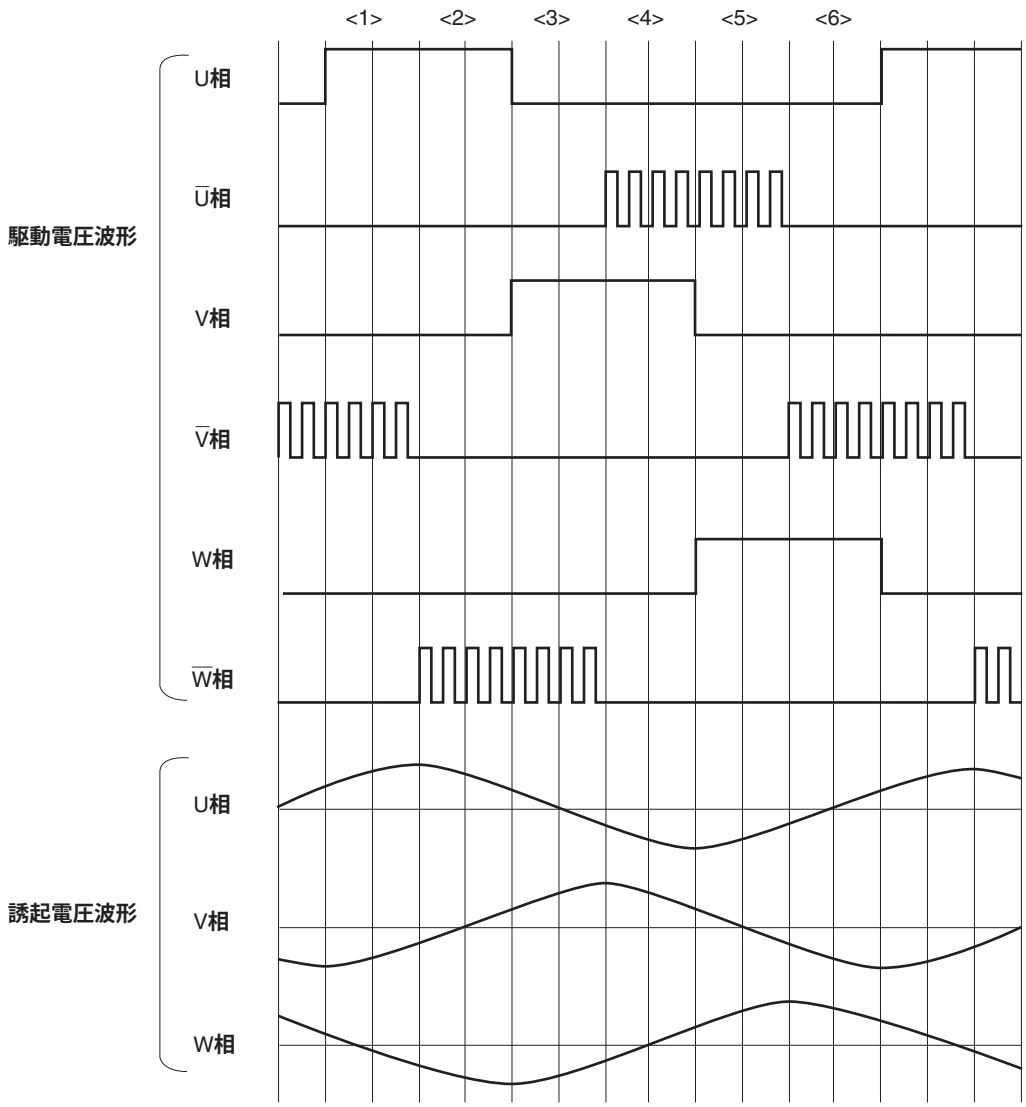
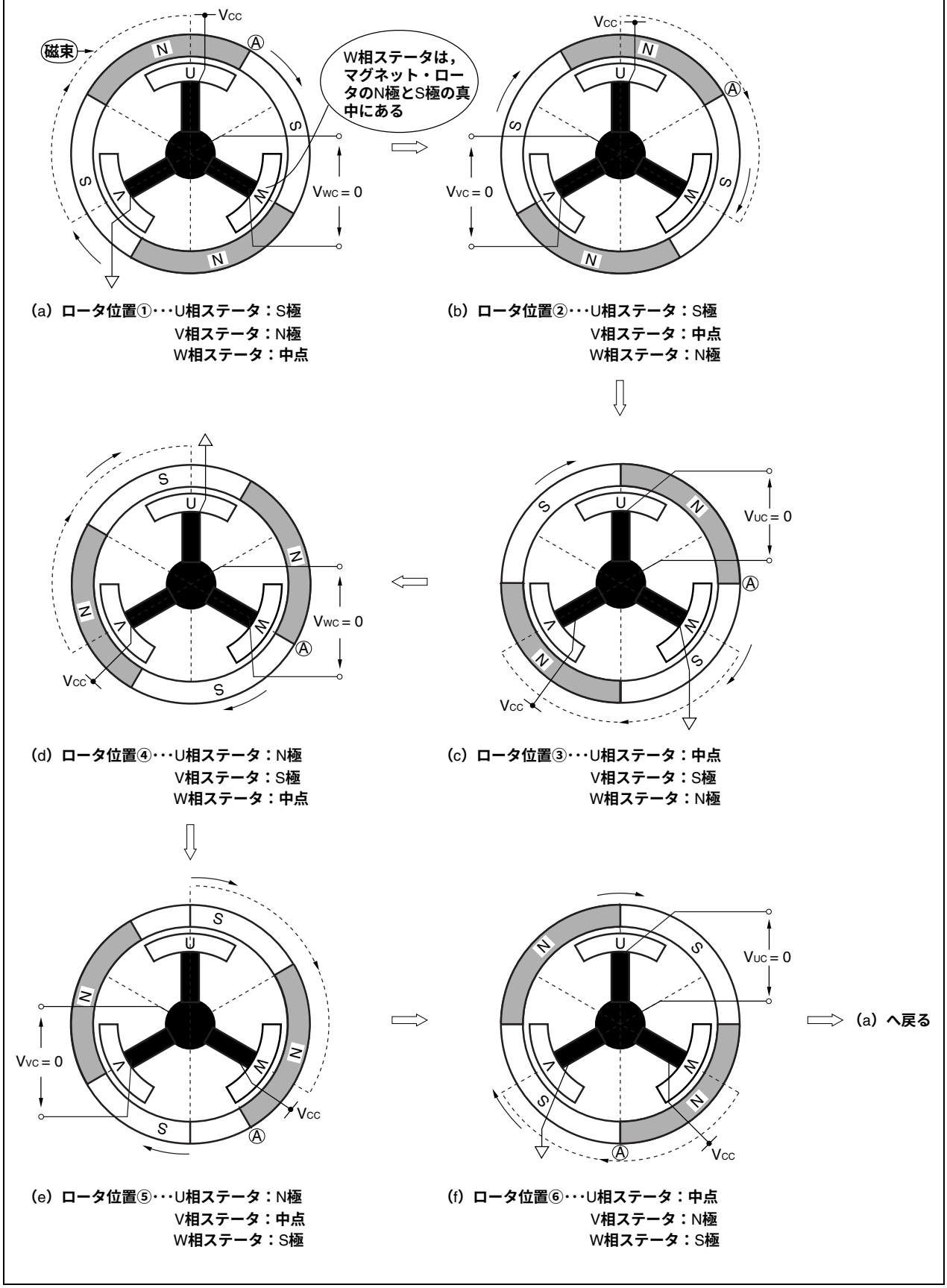


図1－2　ロータ位置検出原理



(a) ロータ位置①･･･U相ステータ：S極
V相ステータ：N極
W相ステータ：中点

(b) ロータ位置②･･･U相ステータ：S極
V相ステータ：中点
W相ステータ：N極

(c) ロータ位置③･･･U相ステータ：中点
V相ステータ：S極
W相ステータ：N極

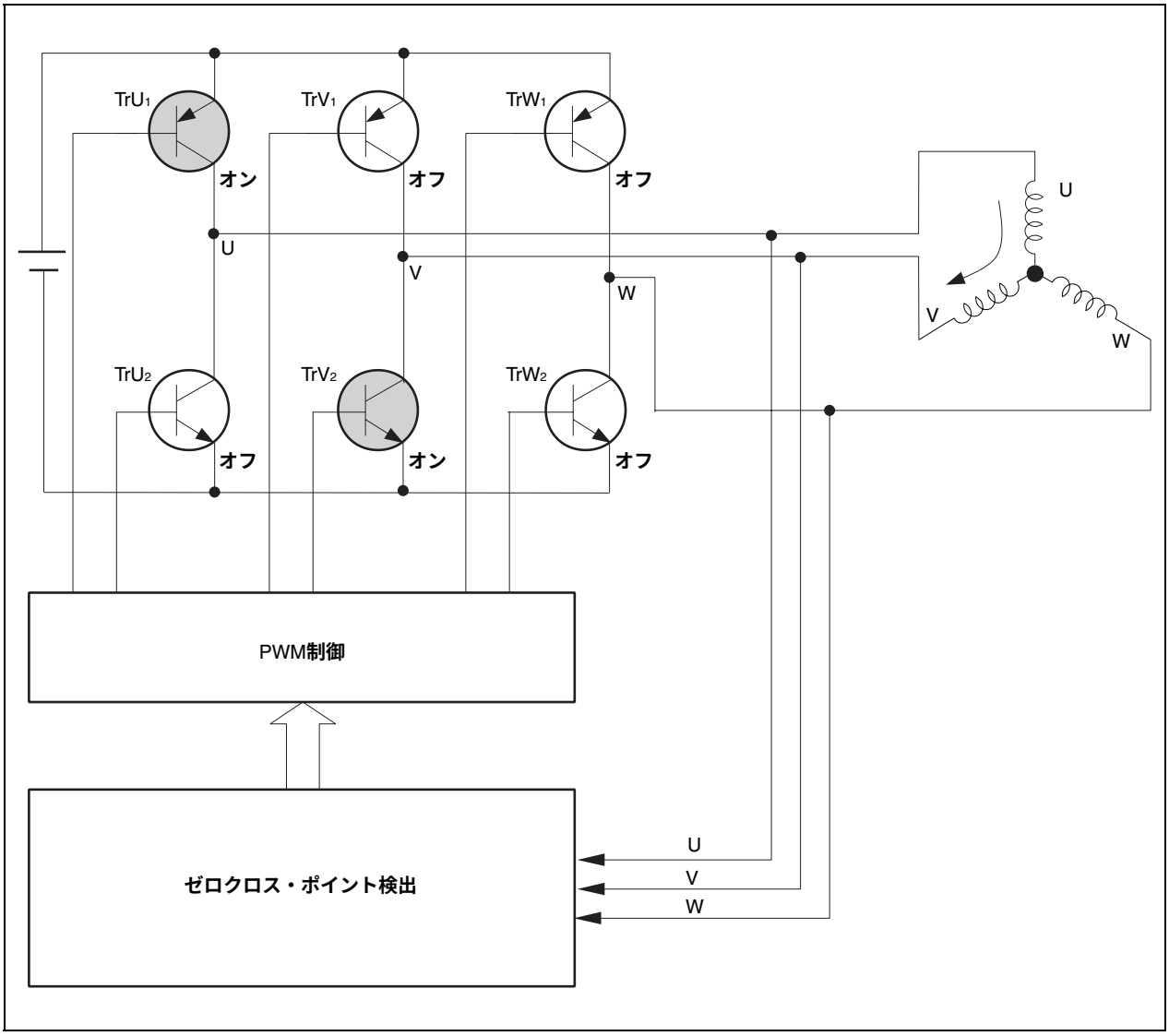
(d) ロータ位置④･･･U相ステータ：N極
V相ステータ：S極
W相ステータ：中点

(e) ロータ位置⑤･･･U相ステータ：N極
V相ステータ：中点
W相ステータ：S極

(f) ロータ位置⑥･･･U相ステータ：中点
V相ステータ：N極
W相ステータ：S極

(a) へ戻る

図1－3　3相ブラシレスDCモータ構成



センサレスBLDCモータの120度制御方法について次に説明します。

BLDCモータを制御するには，ロータの位置を知る必要があります。

BLDCモータのセンサレス制御の場合は，誘起電圧を用いてロータの位置を推定します。

図1－1のように，誘起電圧は駆動電圧波形と同じ位相で，正弦波に近い波形となります。また，図1－2は，モータのステータの極性切り替えとマグネット・ロータの回転の様子です。

図1－1と図1－2からわかるように，3相DCモータは，3相巻き線に3通りの駆動電流パターンを切り替えて流してロータを回転させます。

たとえば，<1>の期間では，U相駆動端子は出力トランジスタTrU_1がオンし，またV相駆動端子はTrV_2がオンとなりU相駆動端子からV相駆動端子に向かって流れます。このとき，W相巻き線は見かけ上，駆動回路から切り離された状態になり誘起電圧が発生します。

この誘起電圧は，ロータ位置を検出するために利用します。

回転数を制御するためには，モータに加える電圧を制御し，コイルに流れる電流を変化させます。電圧を変化させるためにPWM制御した波形をトランジスタに加えます。

上アーム側トランジスタ（TrU_1, TrV_1, TrW_1）をオンさせたまま，下アーム側トランジスタ（TrU_2, TrV_2, TrW_2）に加えたい電圧に比例したデューティの波形（PWM波形）を加えることによって，電圧を変化させます。

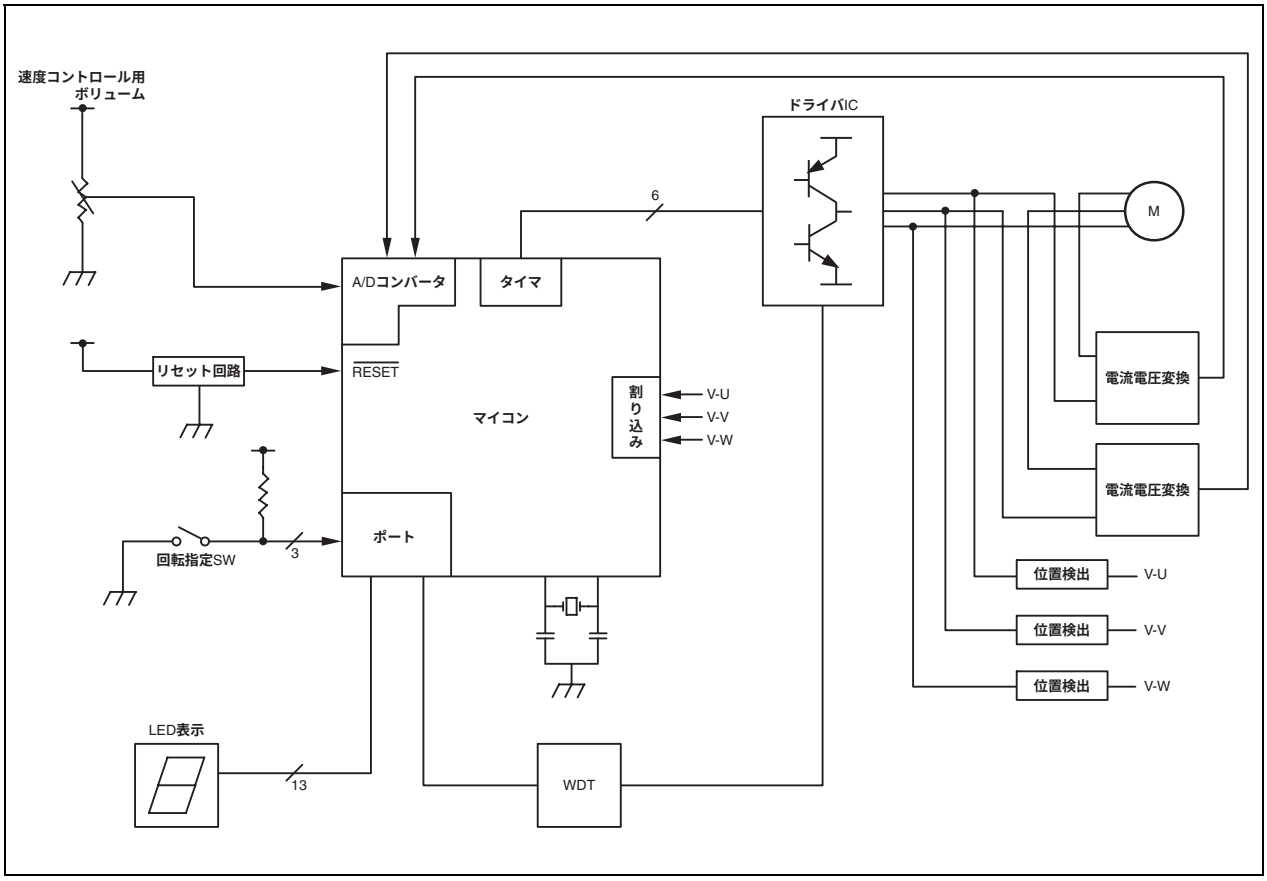
第2章　ハードウエア構成

ハードウエア構成について説明します。

2. 1　構　　成

次にレファレンス・システムの主な機能を示します。このレファレンス・システムは，電源を投入したあと，回転指定SWスイッチを押すと，モータが指定の方向に回転を開始します。

図2－1　全体のシステム構成



(1) 速度コントロール用ボリューム

モータの回転数の増減設定ボリューム

(2) 回転指定SW

CW, CCW, **および**STOP**用スイッチ**

(3) LED**表示**

回転数，演算時間等の表示LED

(4) WDT

ウォッチドッグ・タイマ

(5) ドライバIC

モータ駆動用ドライバ

(6) 電流電圧変換回路

モータの駆動電流を電圧に変換，過電流検出用

(7) 位置検出回路

誘起電圧からロータの位置推定信号出力

2.2 回 路 図

レファレンス・システムの回路図を図2－2に示します。

レファレンス・システムは，μPD78F0714と，リセット回路，発振回路，端子処理のマイコン周辺部，および運転モードSW部，LED出力部，ウォッチドッグ・タイマ回路部，ドライブ回路部，モータ制御部，モータ回転指示部で構成されています。

(1) マイコンおよびマイコン周辺部

μPD78F0714，リセット回路，発振子を使用した発振回路およびMODE端子と未使用端子の端子処理で構成されています。

(2) 運転モードSW部

CW回転またはCCW回転運転の設定を行うSWで構成されています。

(3) LED出力部

回転数，エラー表示などを行う16個のLEDで構成されています。

(4) ウォッチドッグ・タイマ回路部

μPD74HC123Aを使用して，μPD78F0714から1 ms以上の期間パルスがなかった場合，停止信号を出力します。

(5) ドライブ回路部

インバータ・タイマの6相出力からモータ･ドライブ用U, V, W相出力に変換しています。このドライブ回路は，モータ仕様によるため具体例は示しておりません。

(6) モータ制御部

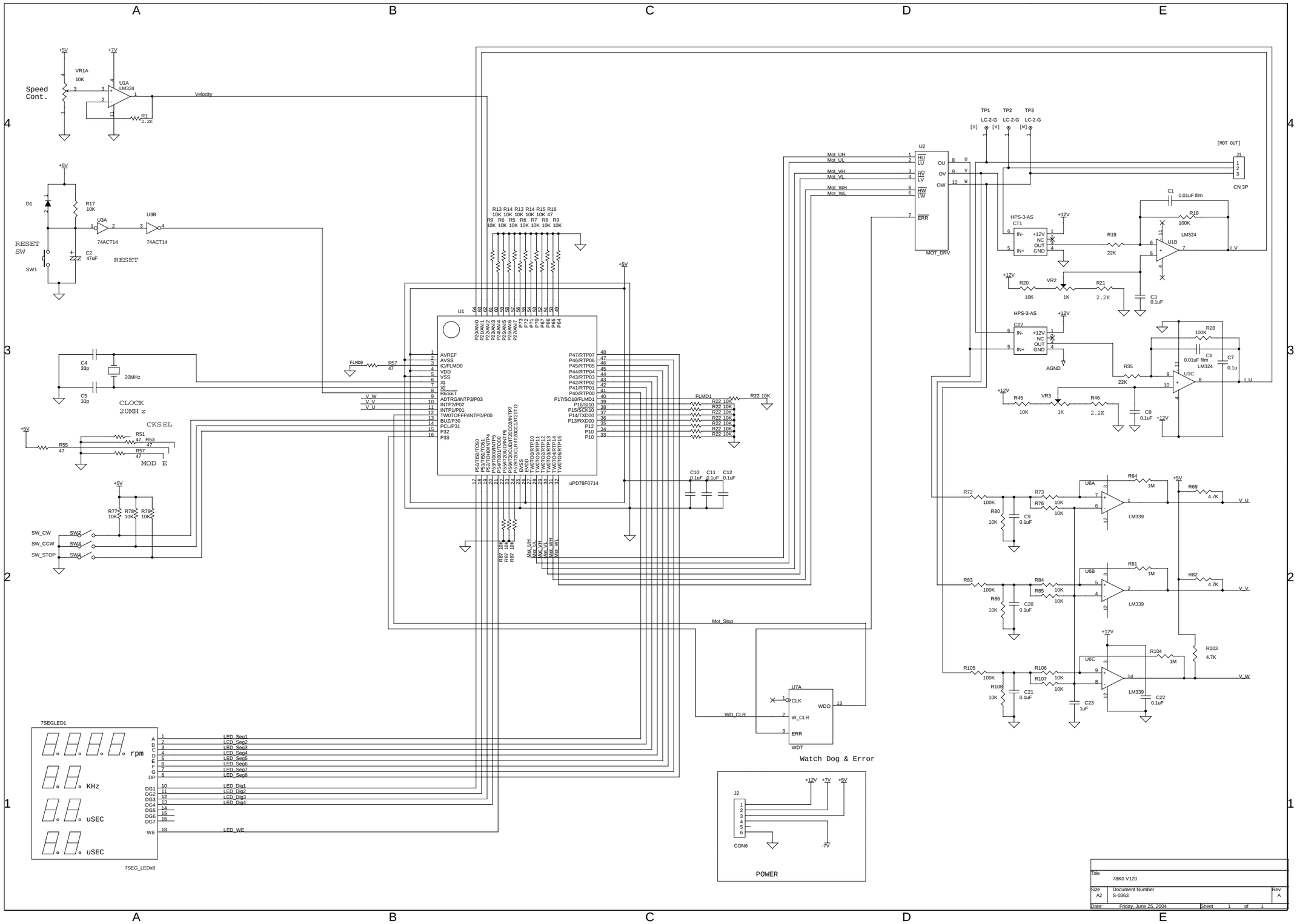
モータの駆動電流U, VをA/D変換により測定するため，HPS-3-AS, LM324などで構成されています。

(7) モータ回転指示部

モータの回転数を設定するため，ボリューム，LM324で構成されています。

〔メ　モ〕

図2－2　μPD78F0714の回路図



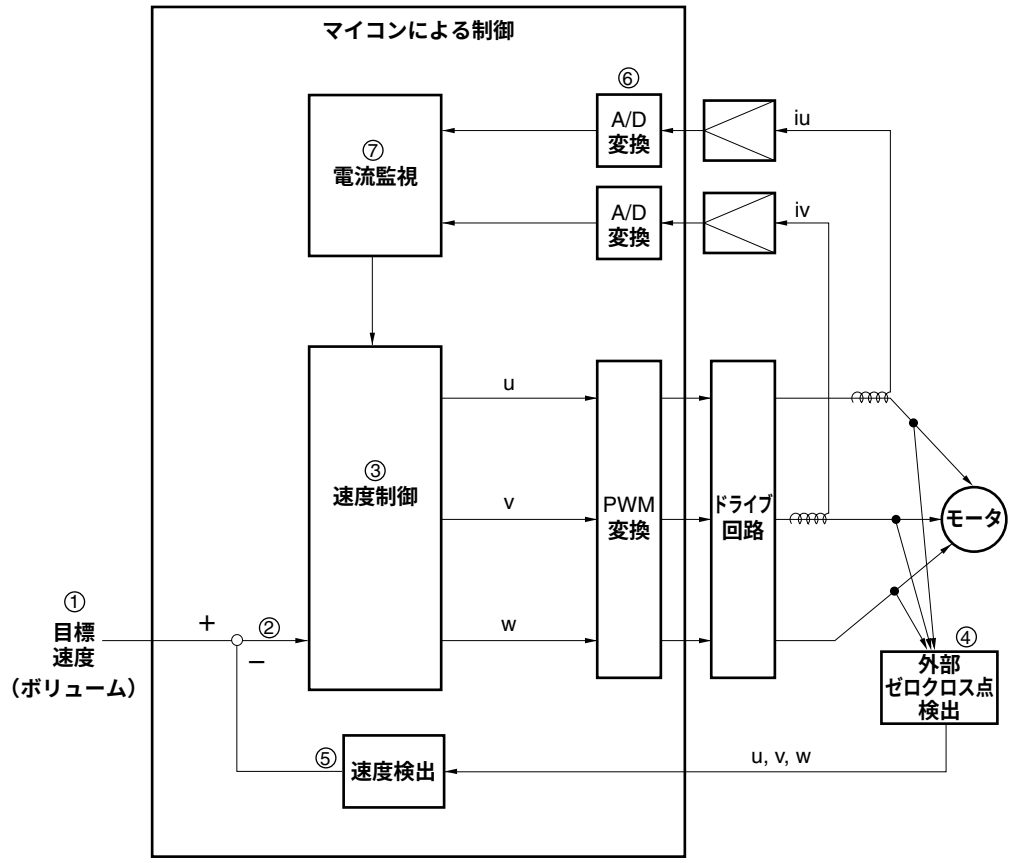
〔メ　モ〕

第3章　ソフトウエア構成

3. 1　制御ブロック

レファレンス・システムのソフトウエア制御ブロック図を次に示します。

図3－1　レファレンス・システムのソフトウエア制御ブロック図



① ボリュームをA/Dコンバータで目標速度として取り込みます。
② 現在の速度との差分により，求める速度を決めます。
③ U, V, Wの各相に値を設定します。
④ 誘起電圧によりゼロクロス・ポイントを検出します（ハードウエアによる）。
⑤ 各相のゼロクロス・ポイントの間隔により速度を求めます。
⑥ A/Dコンバータにより過電流有無の検出をします。
⑦ 電流値により速度制御を行います。

3. 2　周辺I/O

レファレンス・システムでは，次のように周辺I/Oを使用しています。

表3－1　使用周辺I/O一覧

機能	周辺I/O機能名（μPD78F0714）
インバータ・タイマ	タイマW0（TMW0）
10 msタイマ	タイマ00（TM00）
モータ制御タイマ	タイマ00（TM00）
遅延制御タイマ	タイマ51（TM51）
U相電流	ANI0
V相電流	ANI1
設定スピード（ボリューム）	ANI2
U相ゼロクロス入力	INTP1
V相ゼロクロス入力	INTP2
W相ゼロクロス入力	INTP3
CWキー入力	P30
CCWキー入力	P31
STOPキー入力	P32
WDTリセット出力	P33
LED出力	P40-P47, P50-P57

(1) 周辺I/O機能の説明

(a) インバータ・タイマ

インバータ・タイマを使用してPWM波形を出力します。

応用回路例では次のような設定になっています。

- 10 kHz対称三角波モード
- インバータ・タイマ出力：ロウ・アクティブ
- TW0TOFFP端子入力がハイ・レベルのとき，PWM出力停止

(b) モータ制御用タイマ

モータ制御用タイマを使用し，2 msインターバル割り込みを発生させます。

(c) 10 msタイマ

10 msタイマを使用し，10 msインターバル割り込みを発生させます。

(d) 遅延制御タイマ

遅延制御タイマを使用し，30度遅延切り替え用割り込みを発生させます。

(e) 電流値入力

ANI0：U相電流値（－5～＋5 A）

ANI1：V相電流値（－5～＋5 A）

(f) 速度指令ボリューム値入力

ANI2を使用して0-1023までの値を入力します。

3. 3　ソフトウエア処理構造

次にソフトウエア処理構造を示します。

図3－2　メイン処理構造

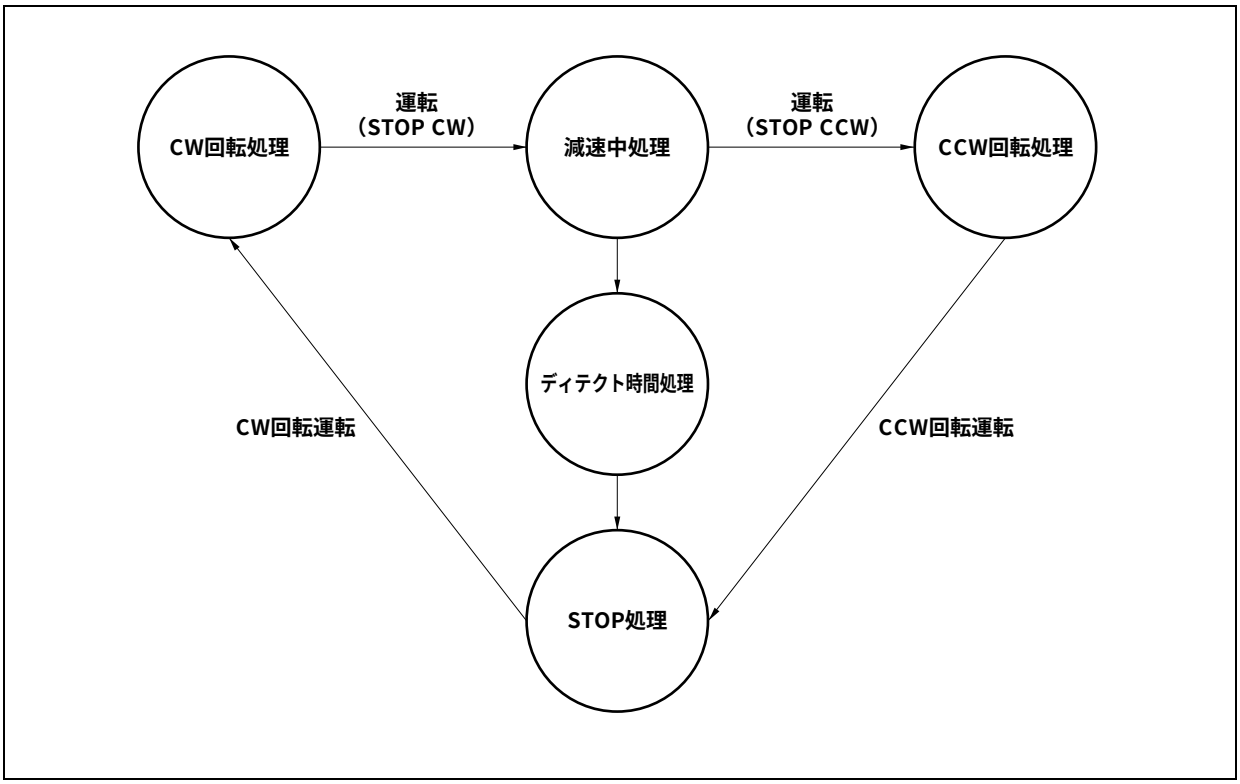
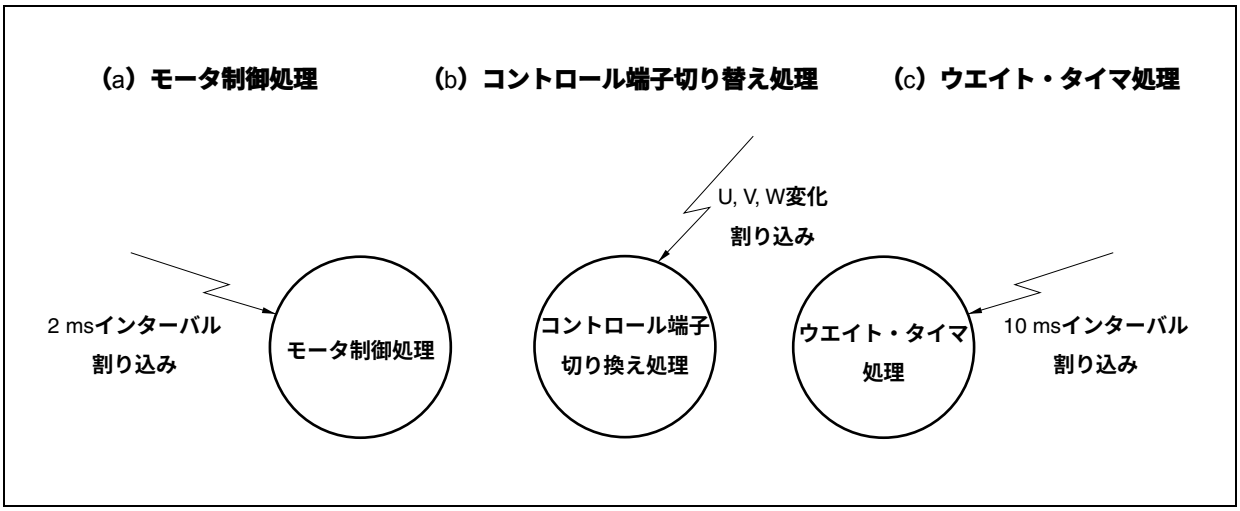


図3－3　割り込み処理構造



メイン処理で運転モードSWの状態を監視して，CW, CCWおよび停止状態へ処理を移行します。指示された状態を2 msインターバル割り込みでモータ制御を行います。

モータ制御状態には次の3つがあります。

・停止状態

　モータの制御を行いません。

・初期動作状態

　起電力のゼロクロス点を検出できる速度まで見込み回転制御を行います。

・速度制御状態

　指示された速度となるようにフィードバック回転制御を行います。

3. 4　フロー・チャート

3. 4. 1　メイン処理

図3－4にメイン処理についてのフローを示します。

図3－4　メイン処理

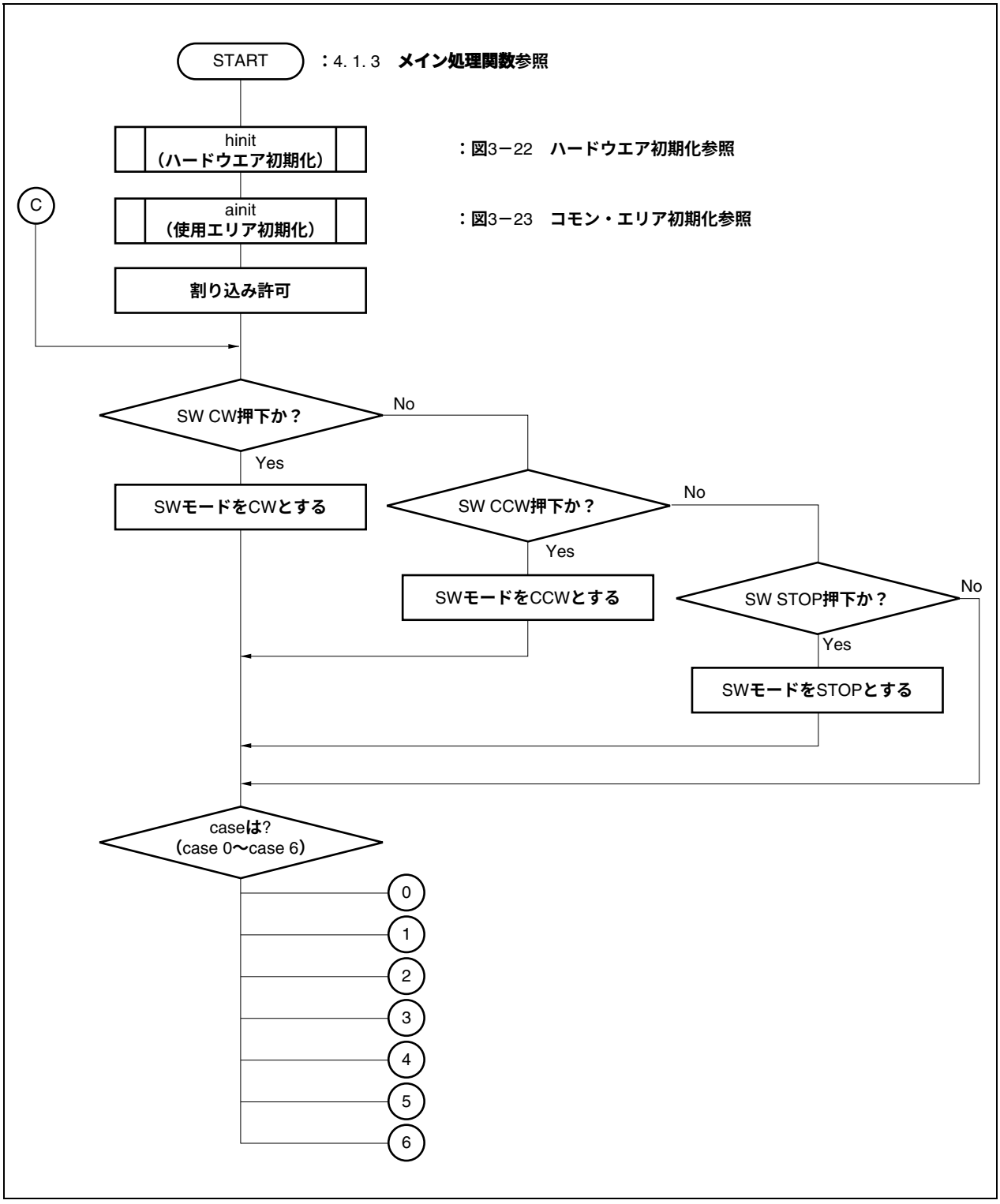


図3ー5　case 0（停止中処理）

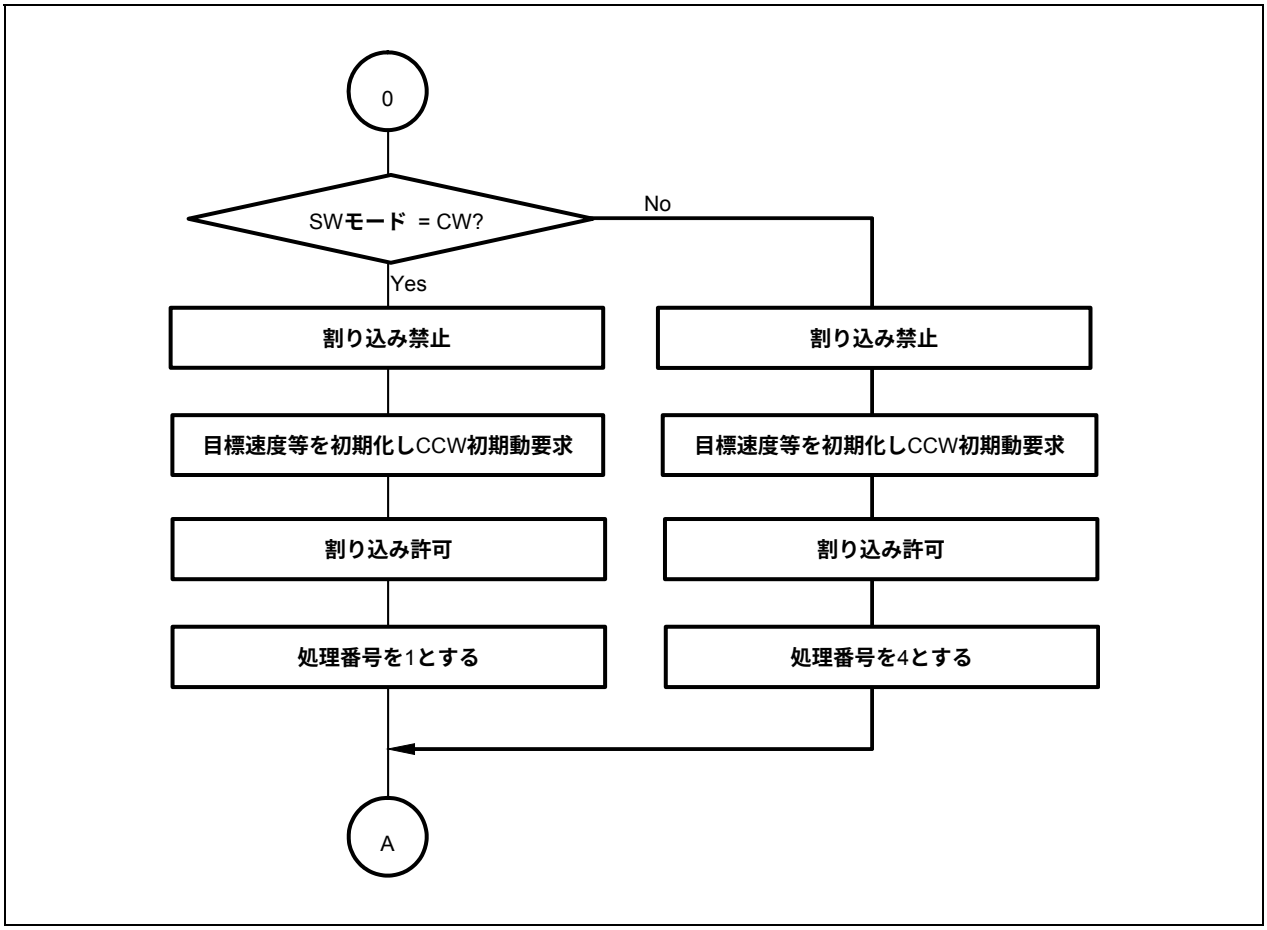


図3－6　case 1（CW加速処理）

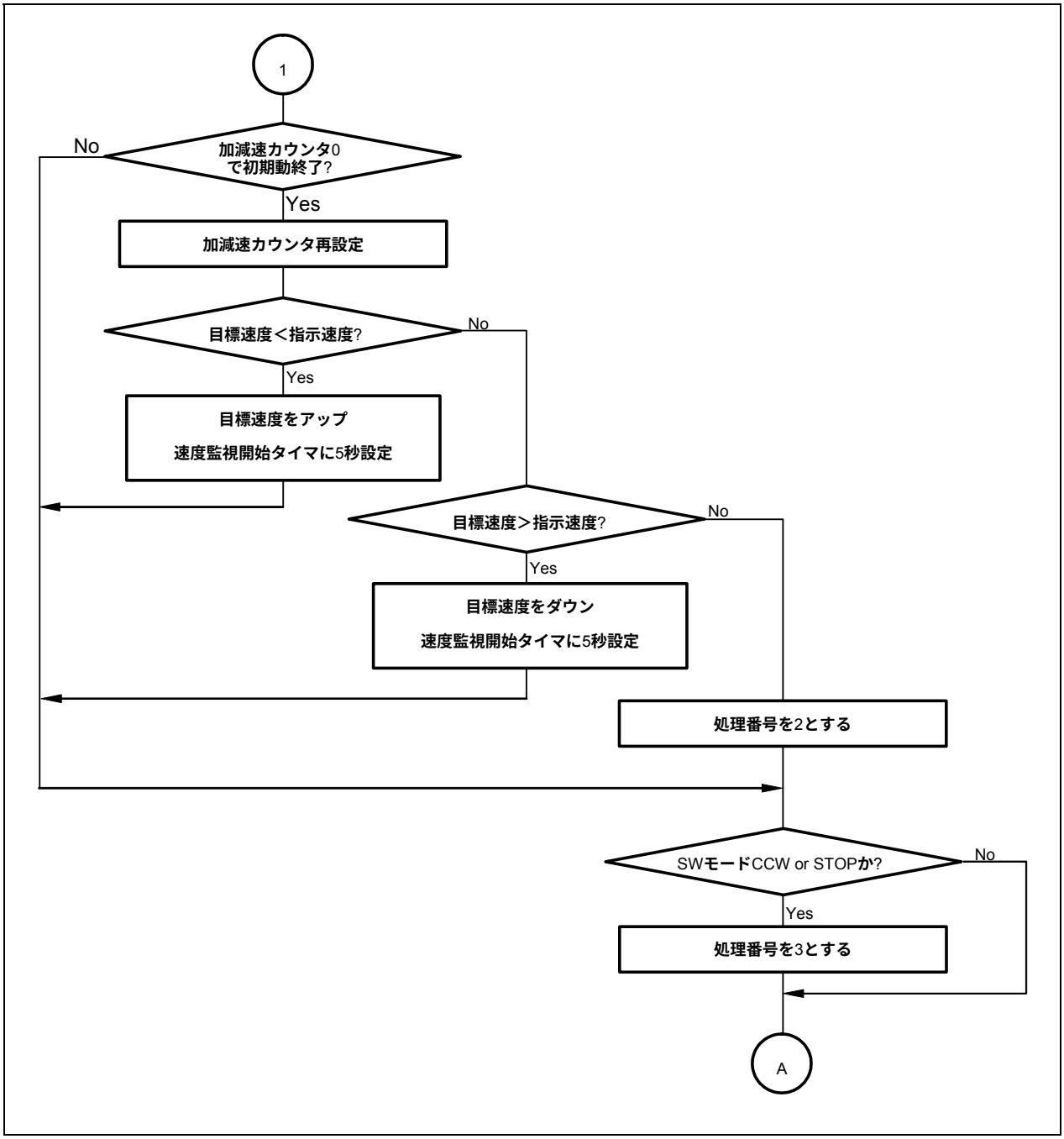


図3－7　case 2（CW定速処理）

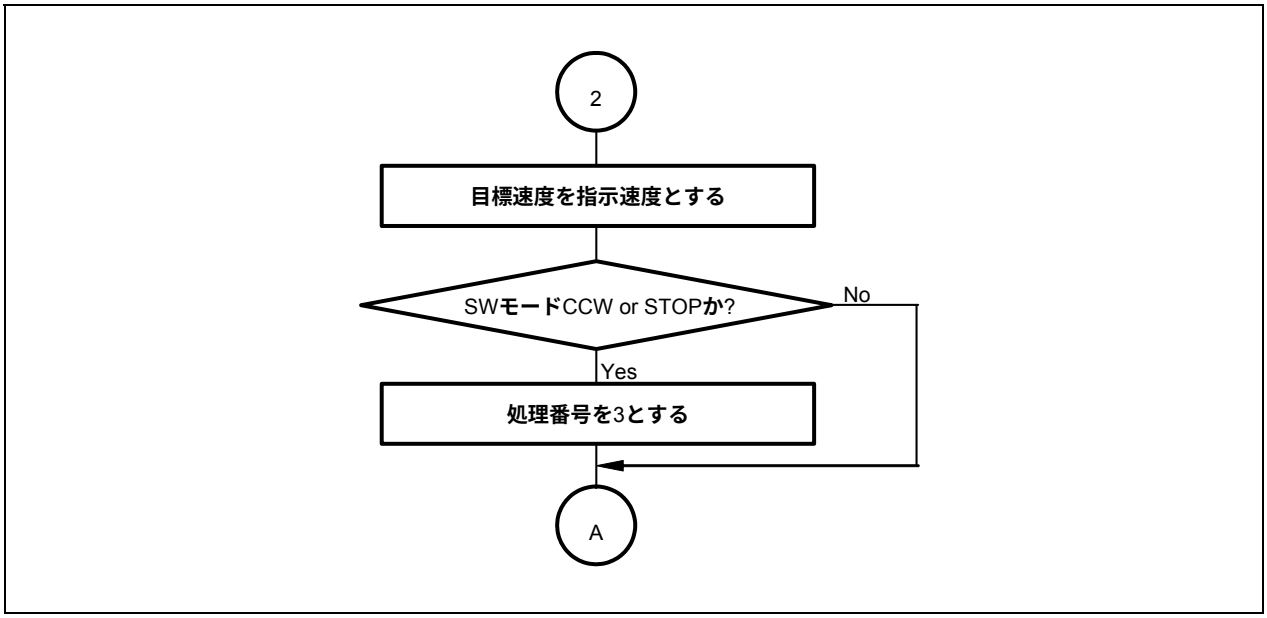


図3－8　case 3（CW停止処理）

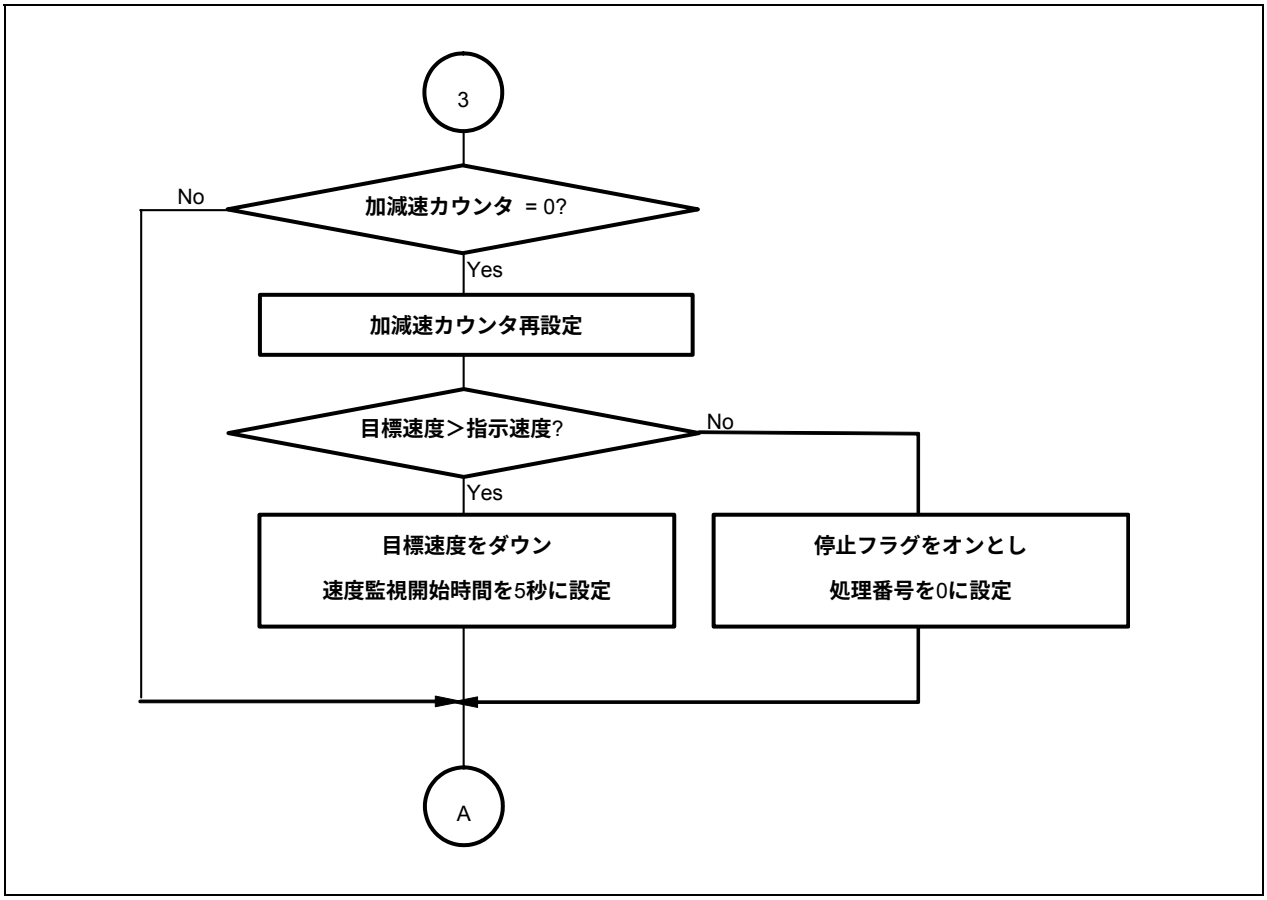


図3－9　case 4（CCW加速処理）

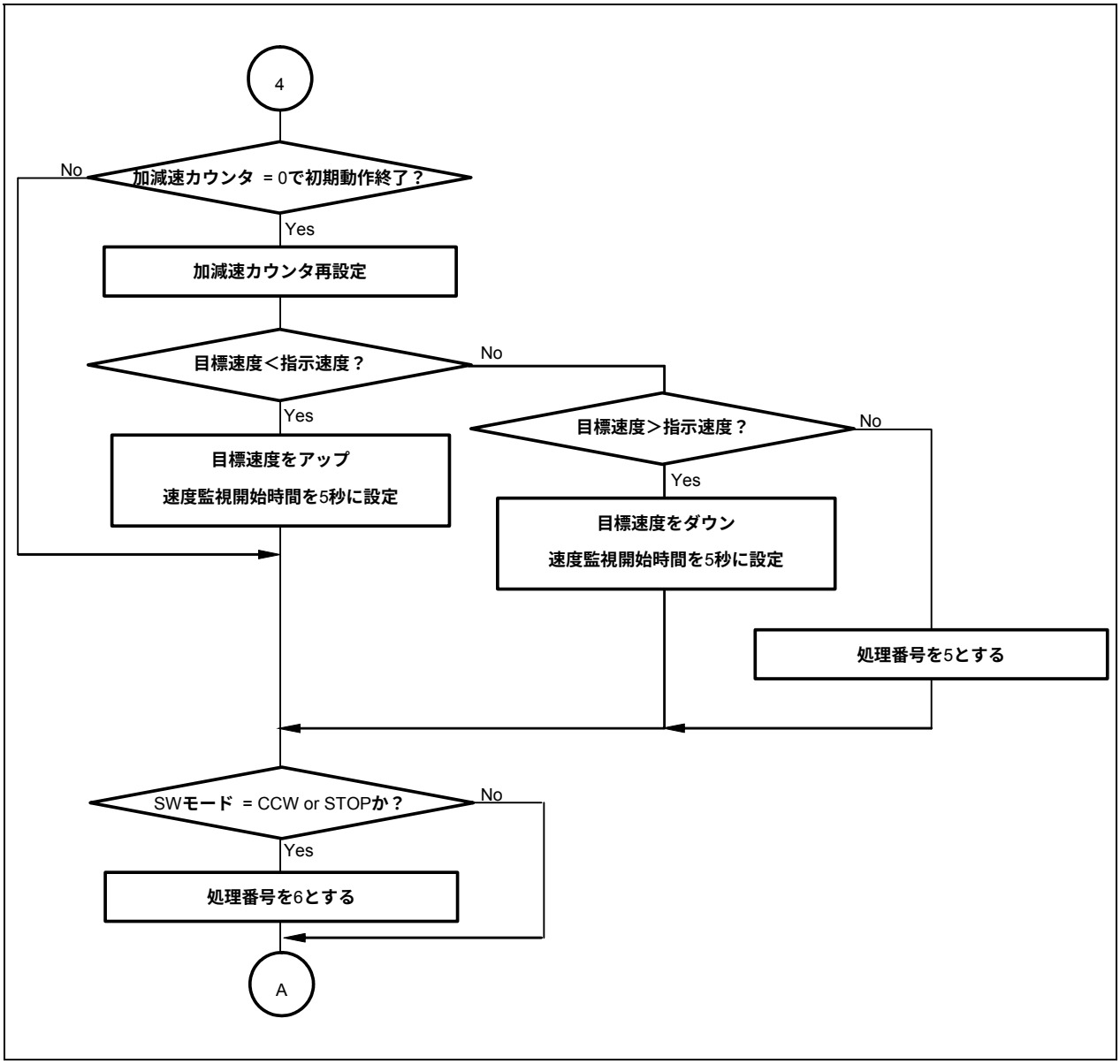


図3－10　case 5（CCW定速処理）

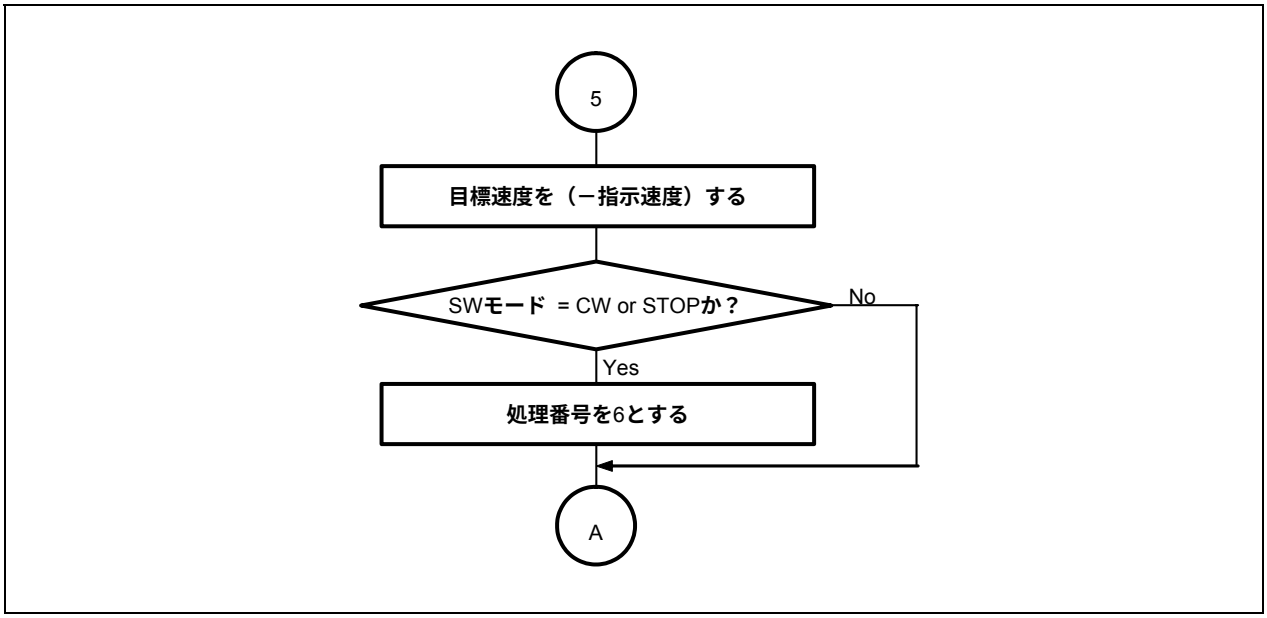


図3－11　case 6（CCW停止処理）

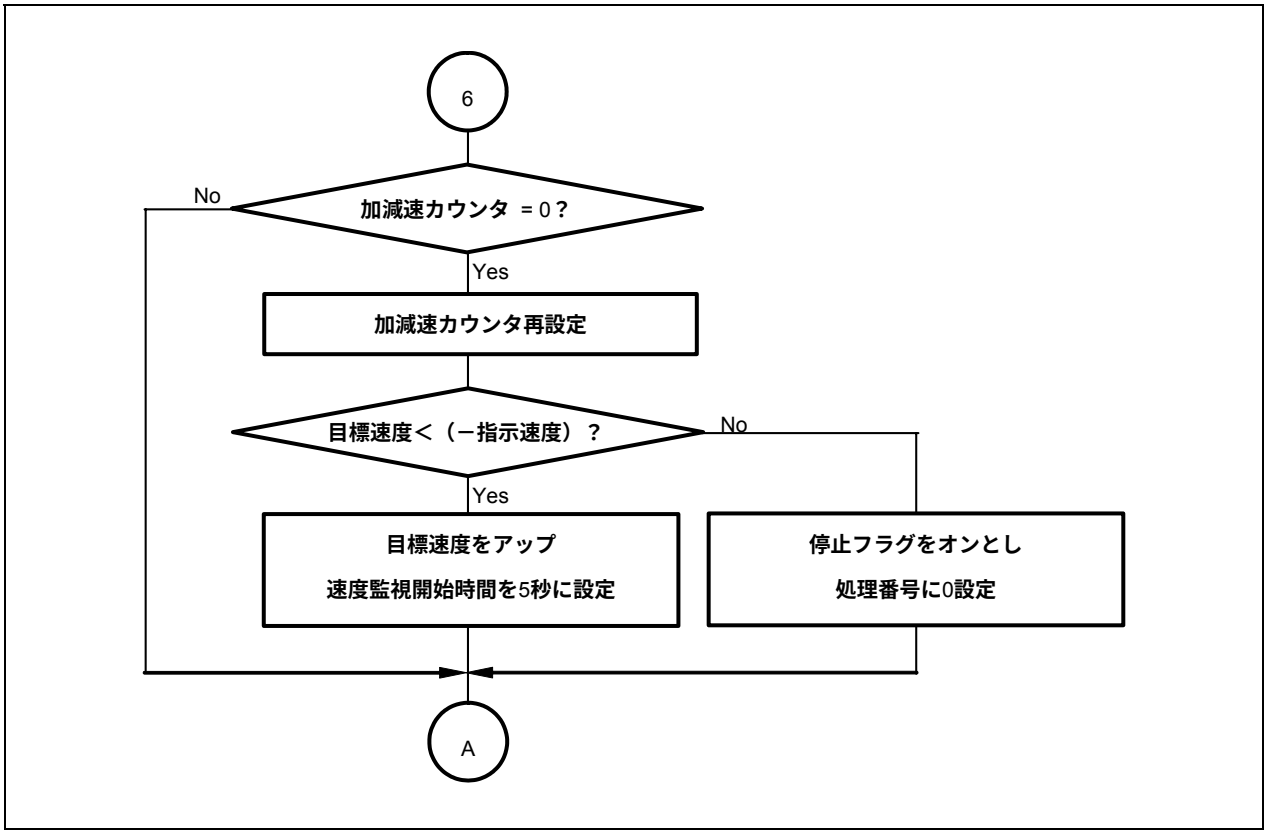


図3－12　ディテクト待ち（1/2）

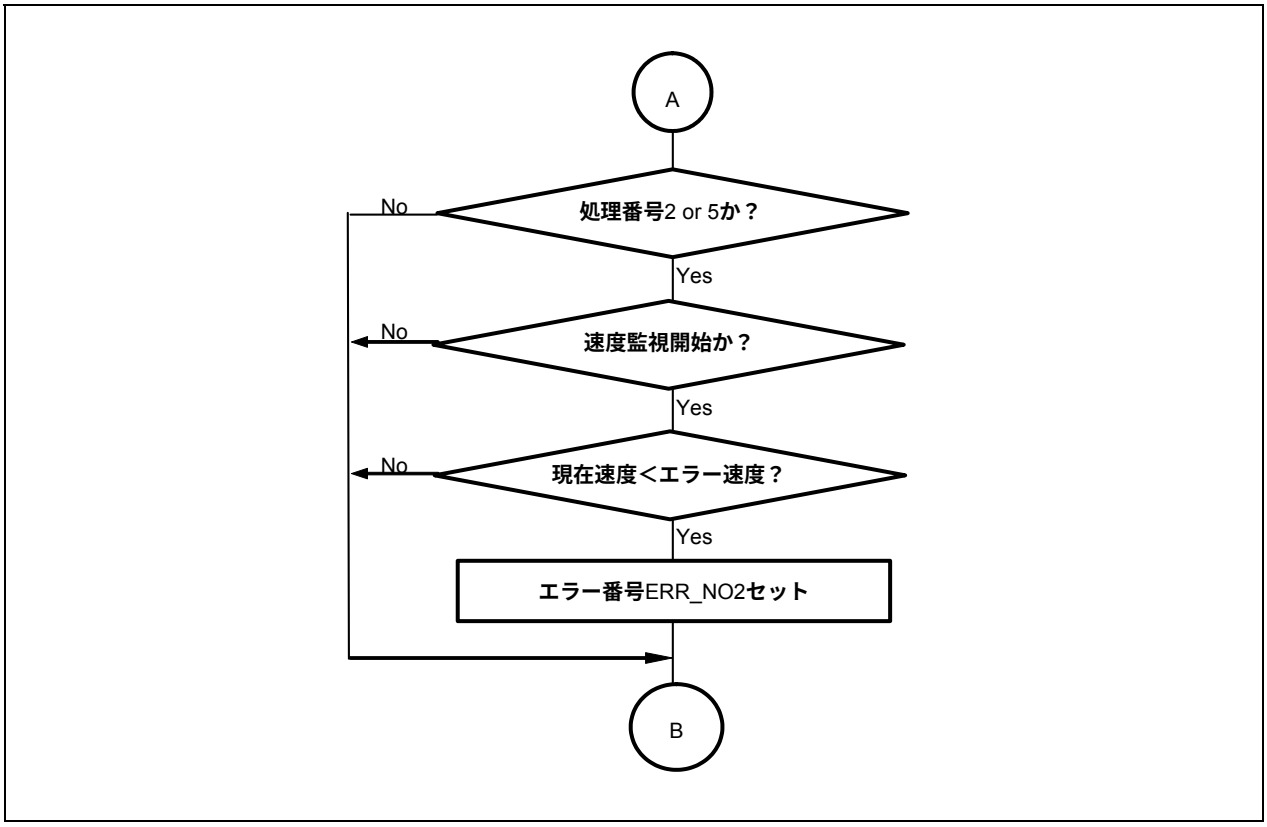
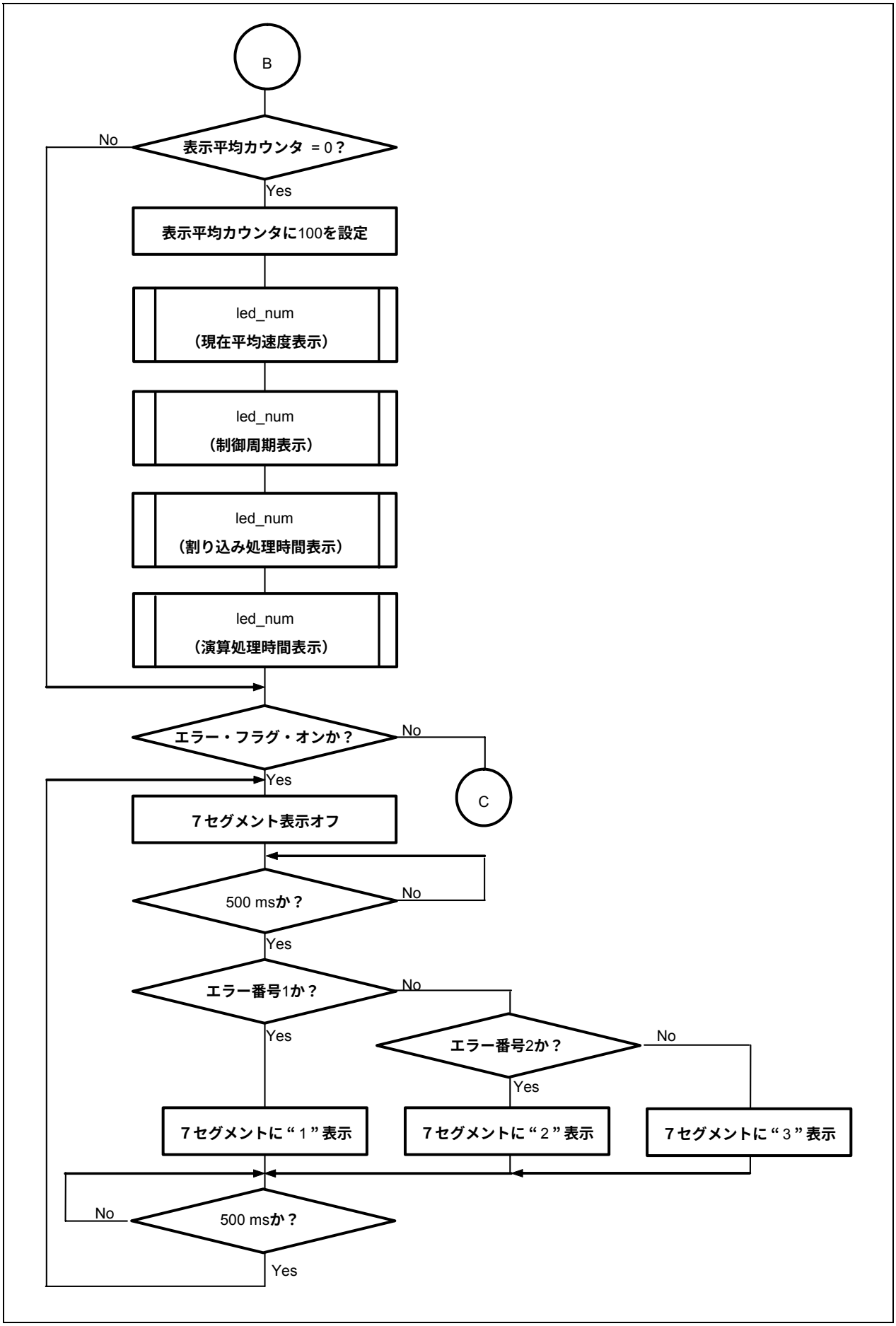


図3－12　ディテクト待ち（2/2）



3. 4. 2　モータ制御処理

図3－13　制御割り込み処理（1/4）

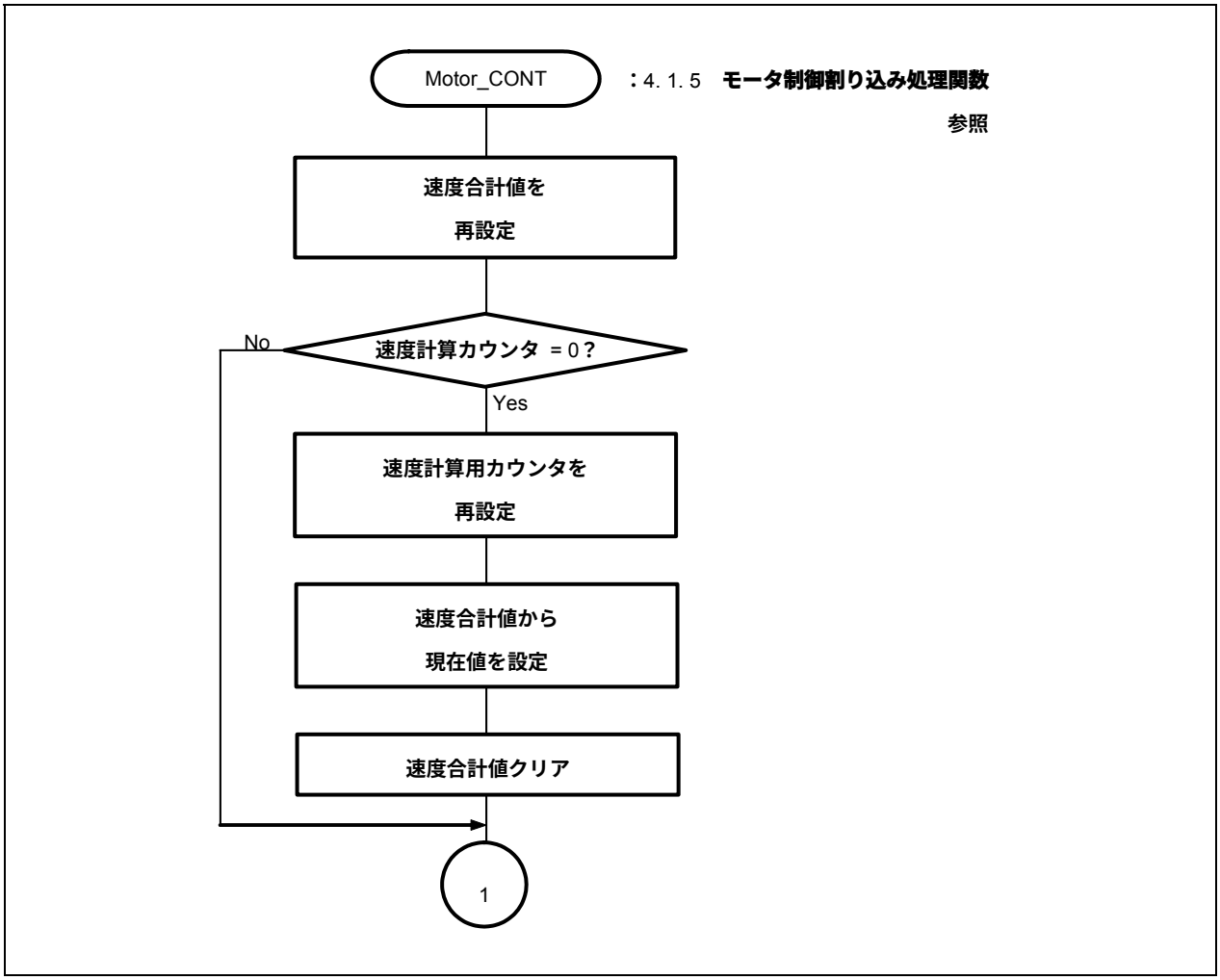


図3－13　制御割り込み処理（2/4）

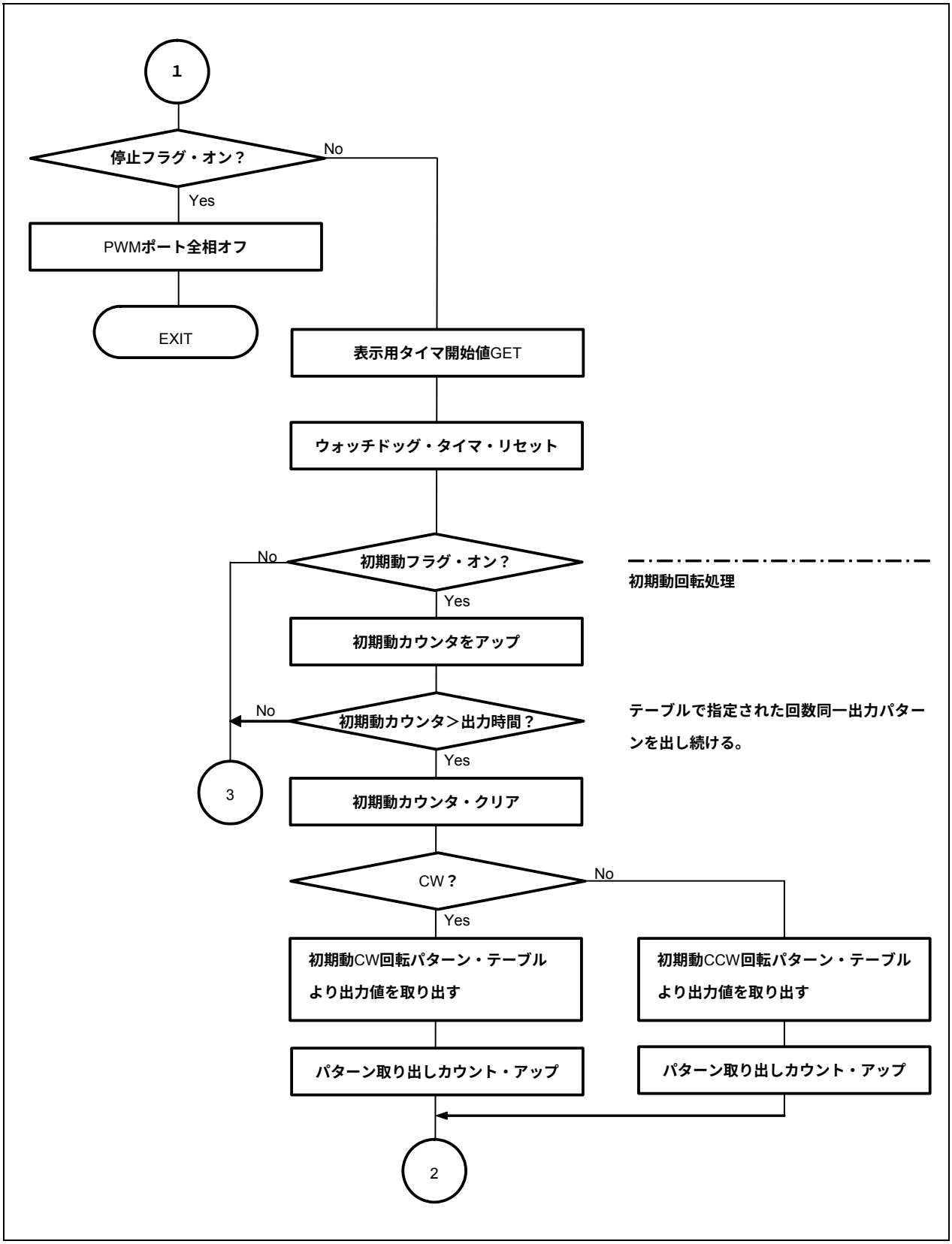


図3－13　制御割り込み処理（3/4）

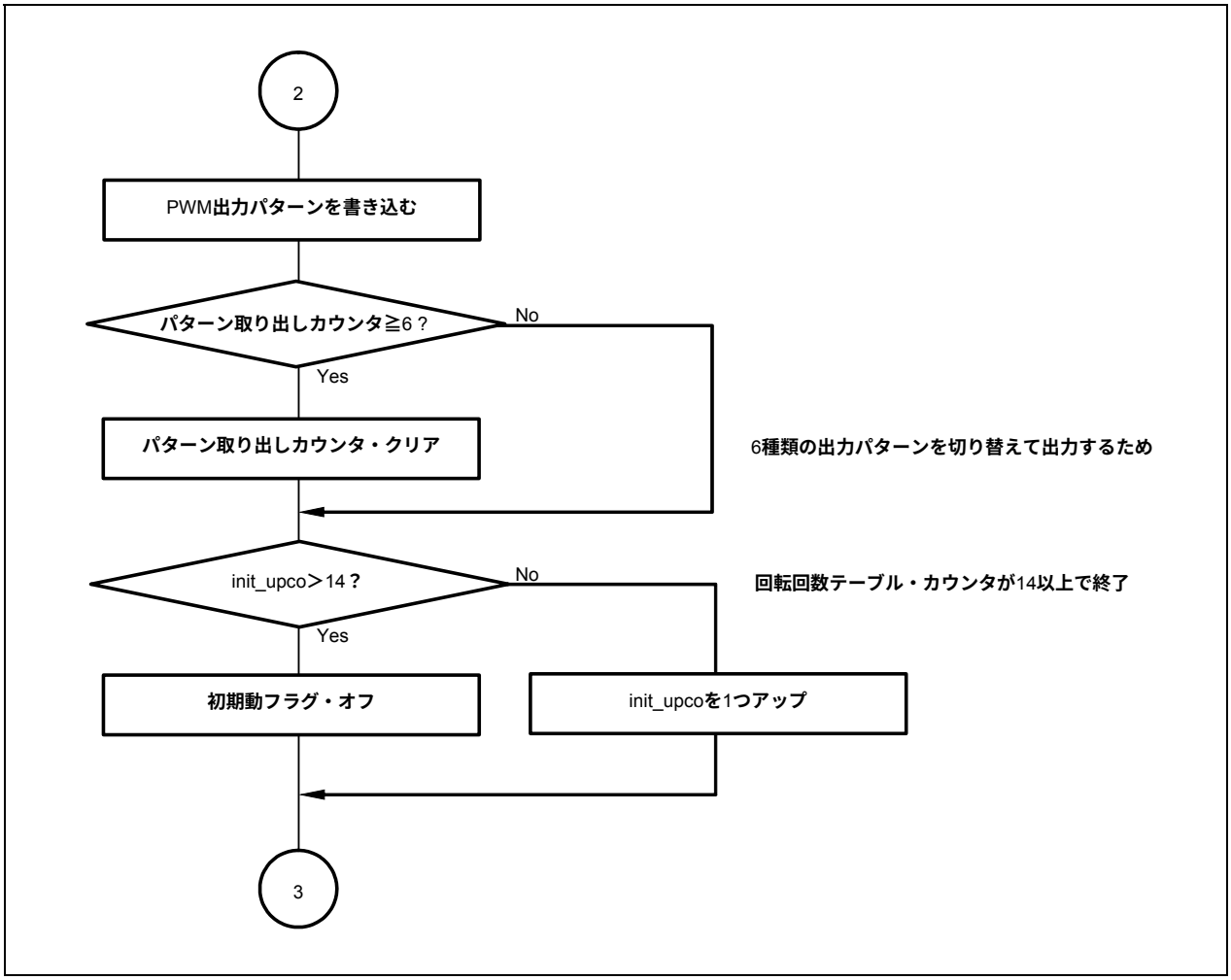
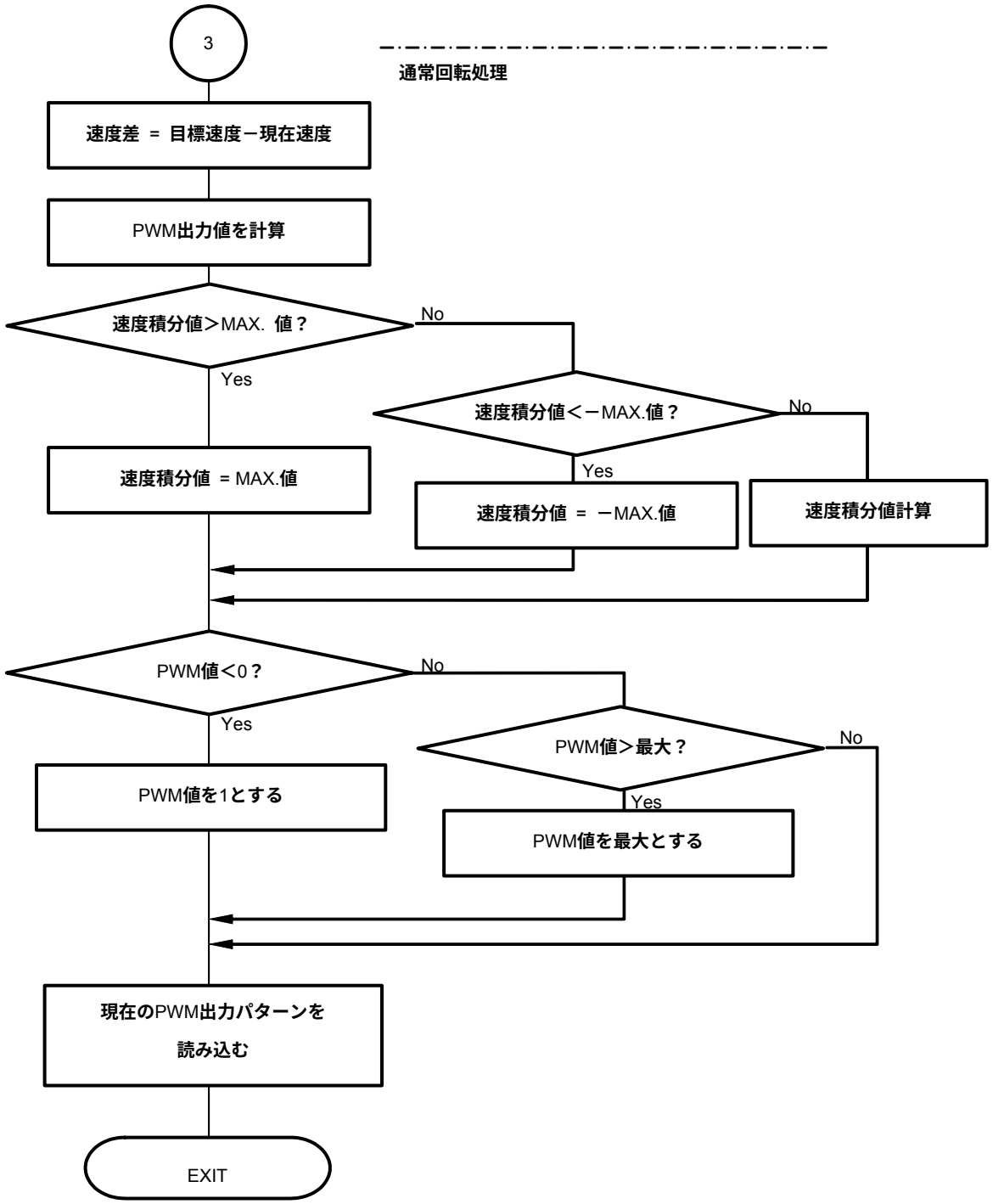


図3－13　制御割り込み処理（4/4）



備考1. PWM =（現在速×ksp）＋（速度差×ksp/10）＋速度積分値＋速度オフセット値

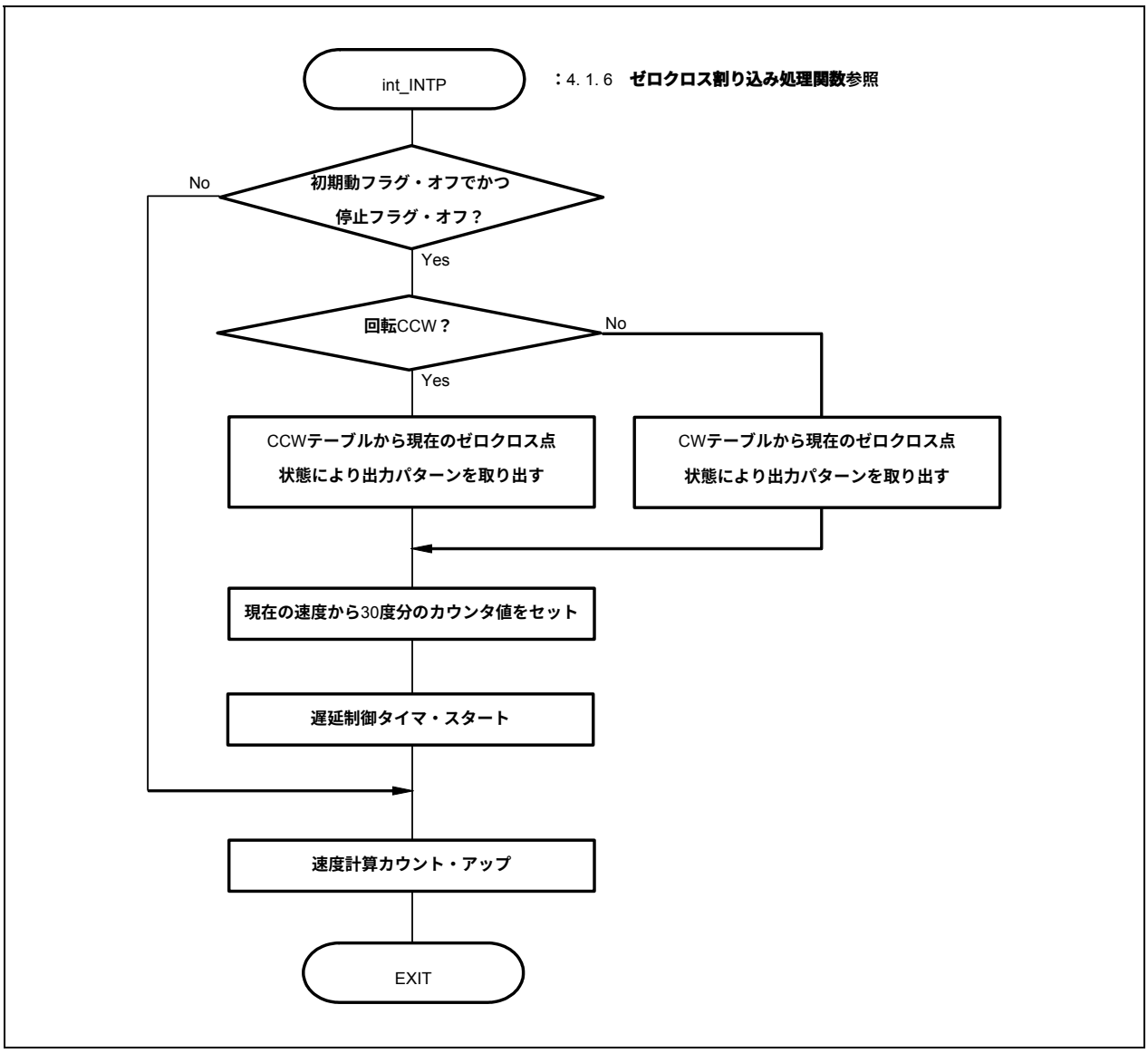
2. 速度積分値 = 速度積分値＋（速度差×ksi）

3. ksp ：速度比例ゲイン

   ksi ：速度積分ゲイン

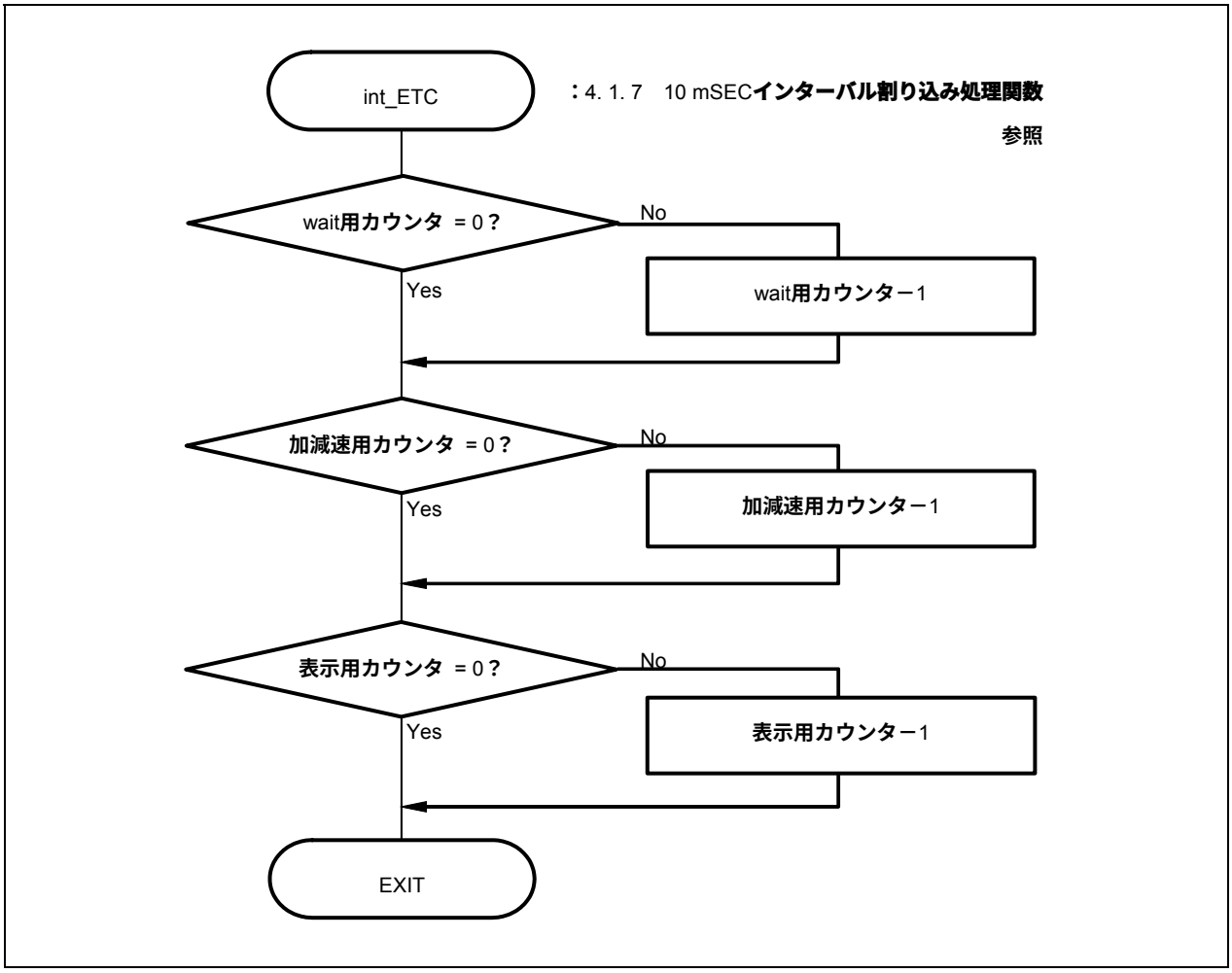
3. 4. 3　U, V, Wゼロクロス点割り込み処理

図3－14　U, V, Wゼロクロス点割り込み処理



3. 4. 4　10 mSECインターバル割り込み処理

図3－15　10 mSECインターバル割り込み処理



3. 4. 5　A/Dコンバータ割り込み処理

図3－16　A/Dコンバータ割り込み処理

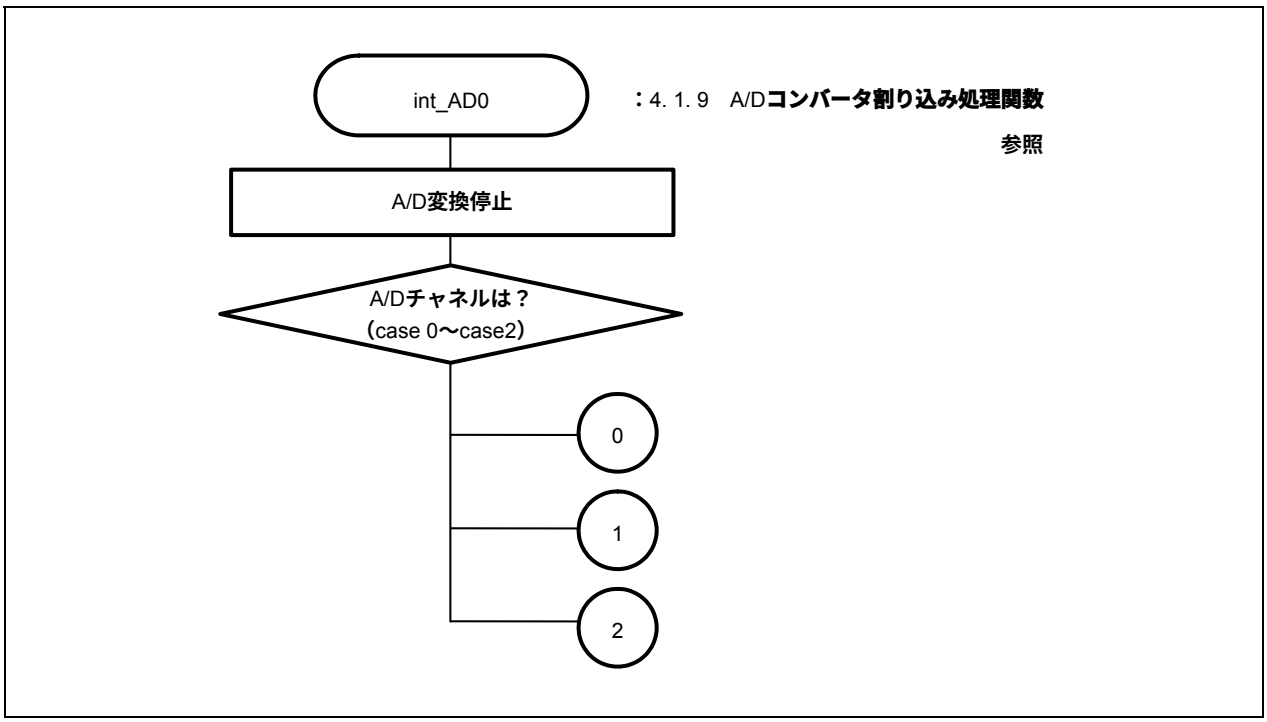


図3－17　case 0（A/Dコンバータ・チャネル1割り込み処理）

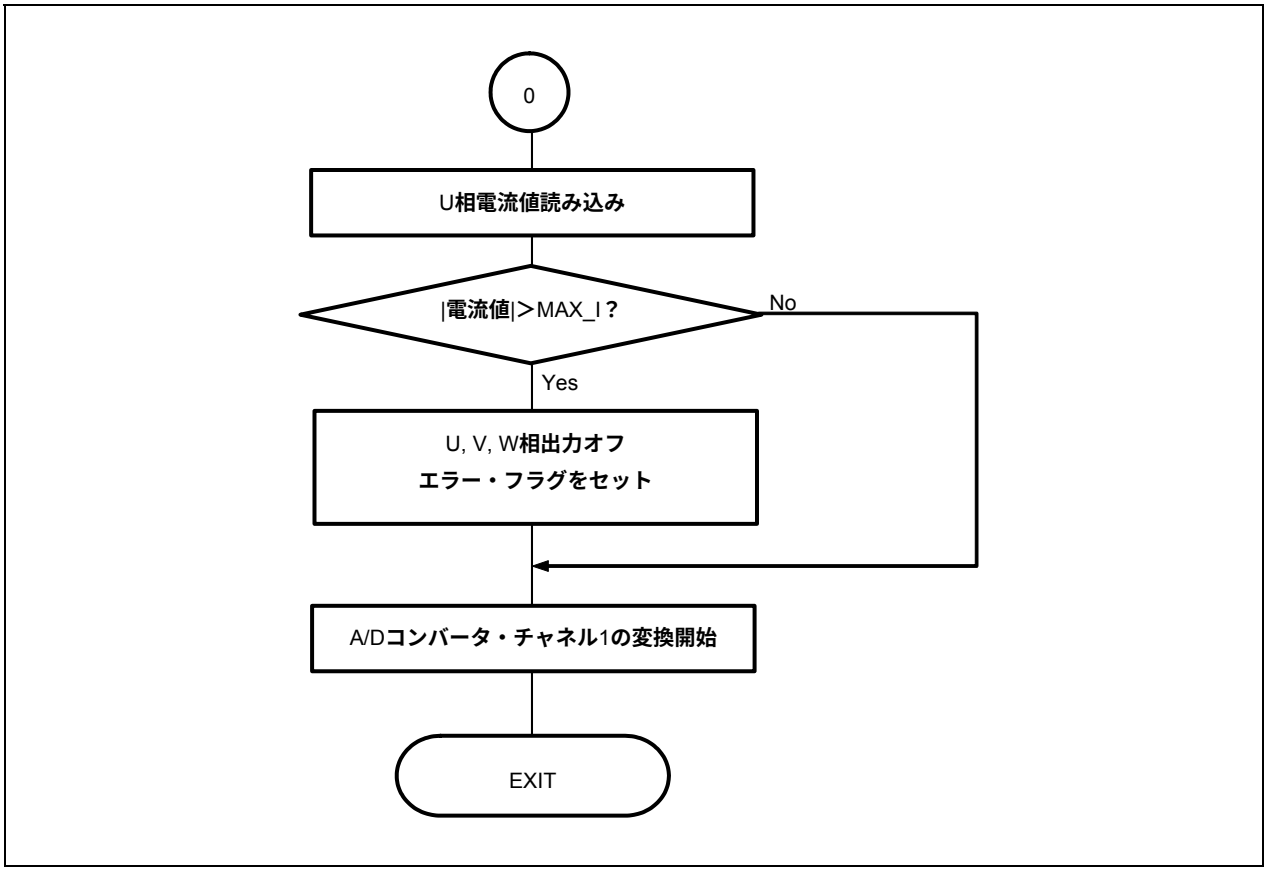


図3－18　case 1（A/Dコンバータ・チャネル2割り込み処理）

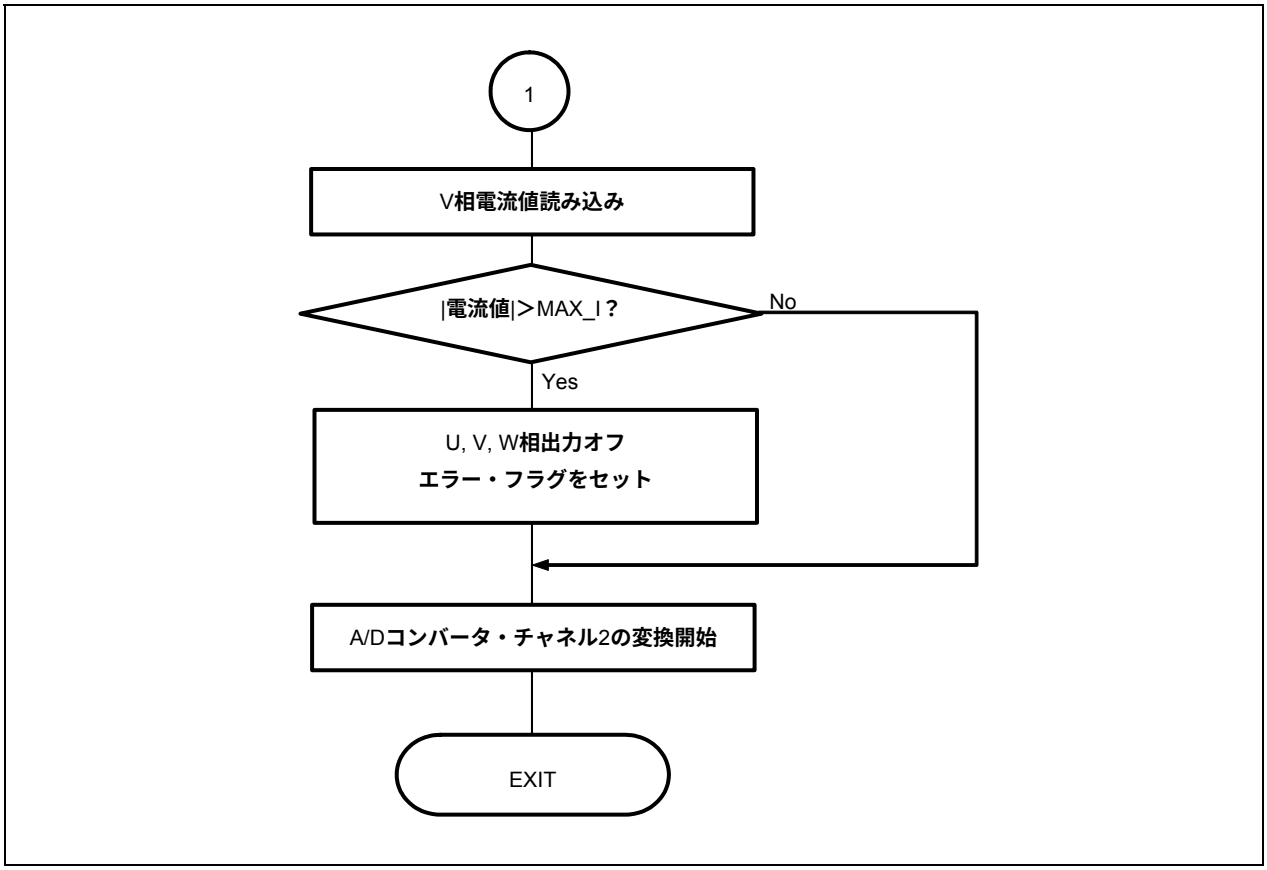
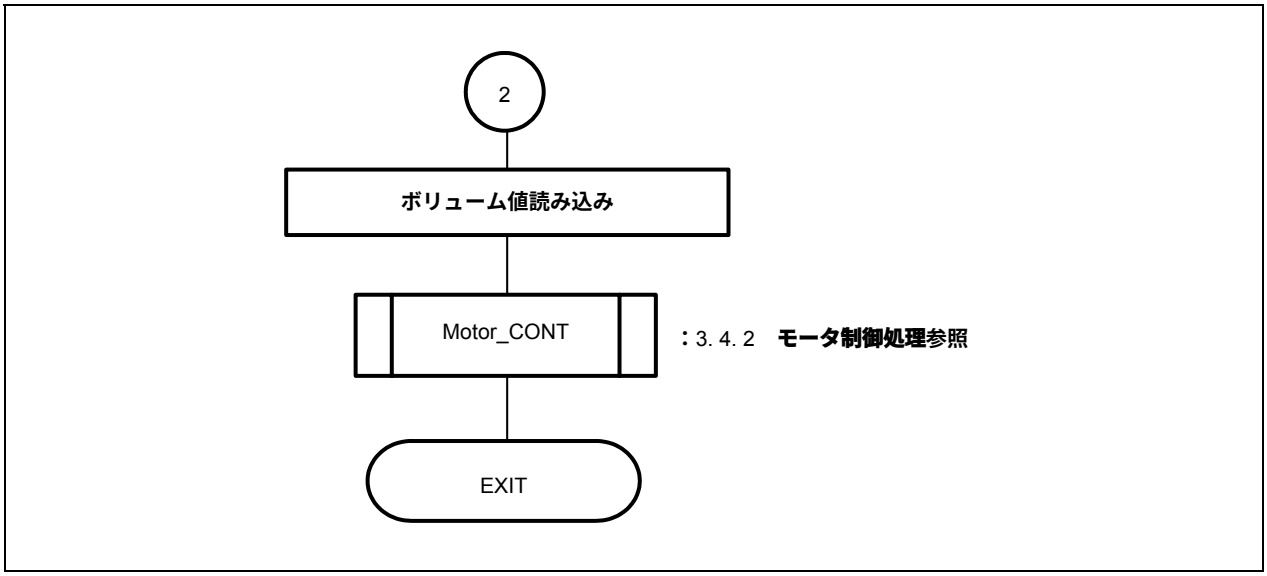
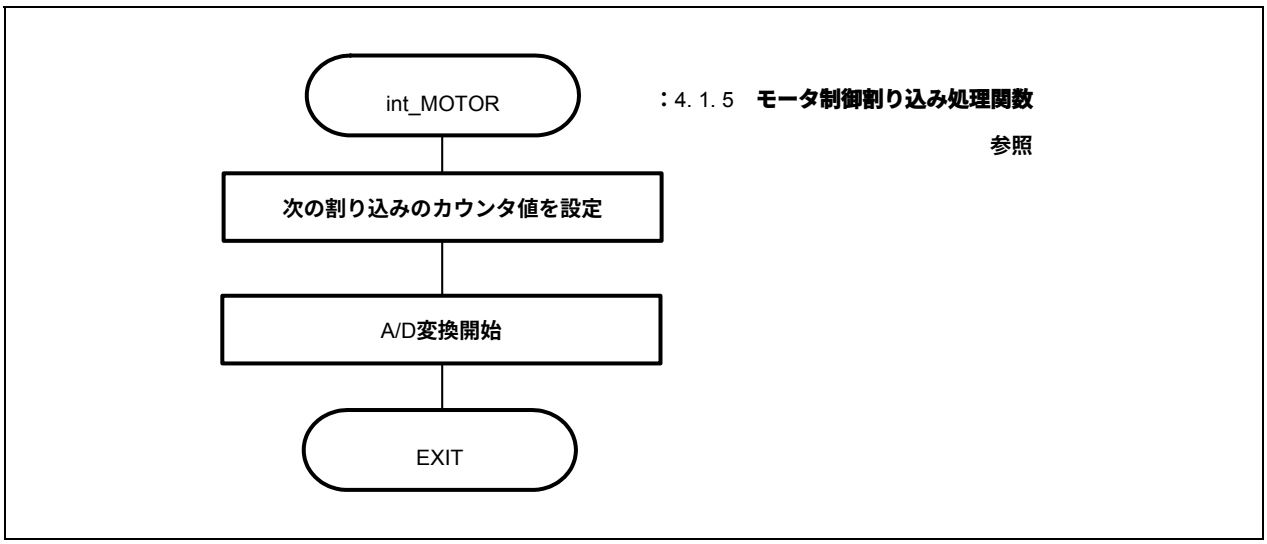


図3－19　case 2（モータ制御処理へ）



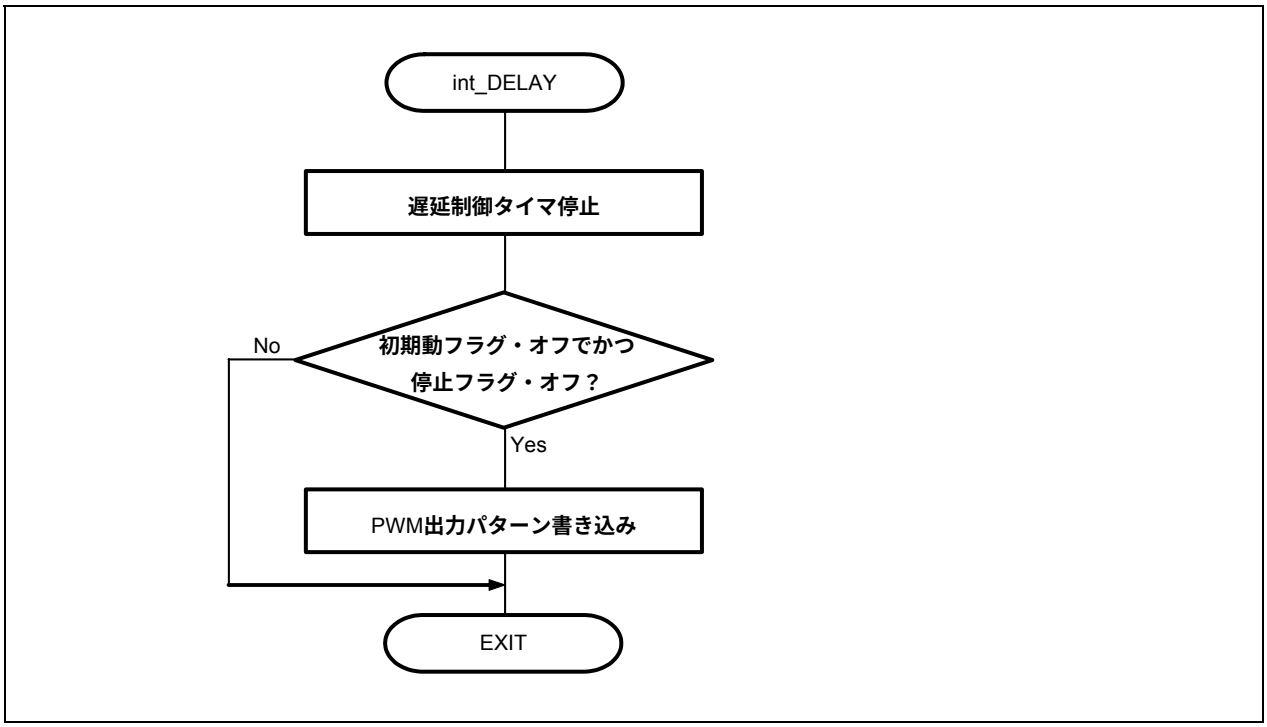
3. 4. 6　モータ制御タイマ割り込み処理

図3－20　モータ制御タイマ割り込み処理



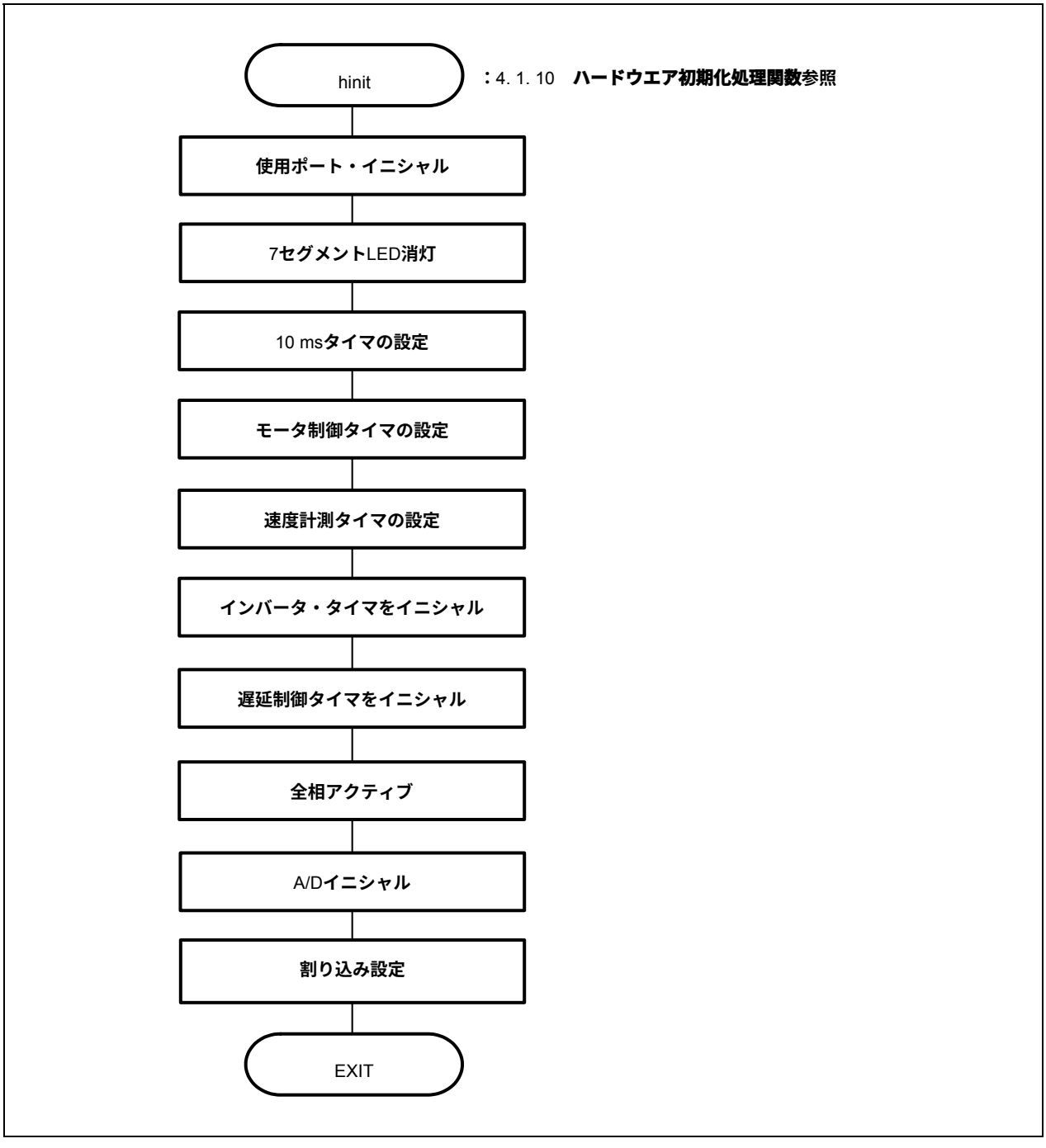
3. 4. 7　遅延制御タイマ割り込み処理

図3－21　遅延制御タイマ割り込み処理



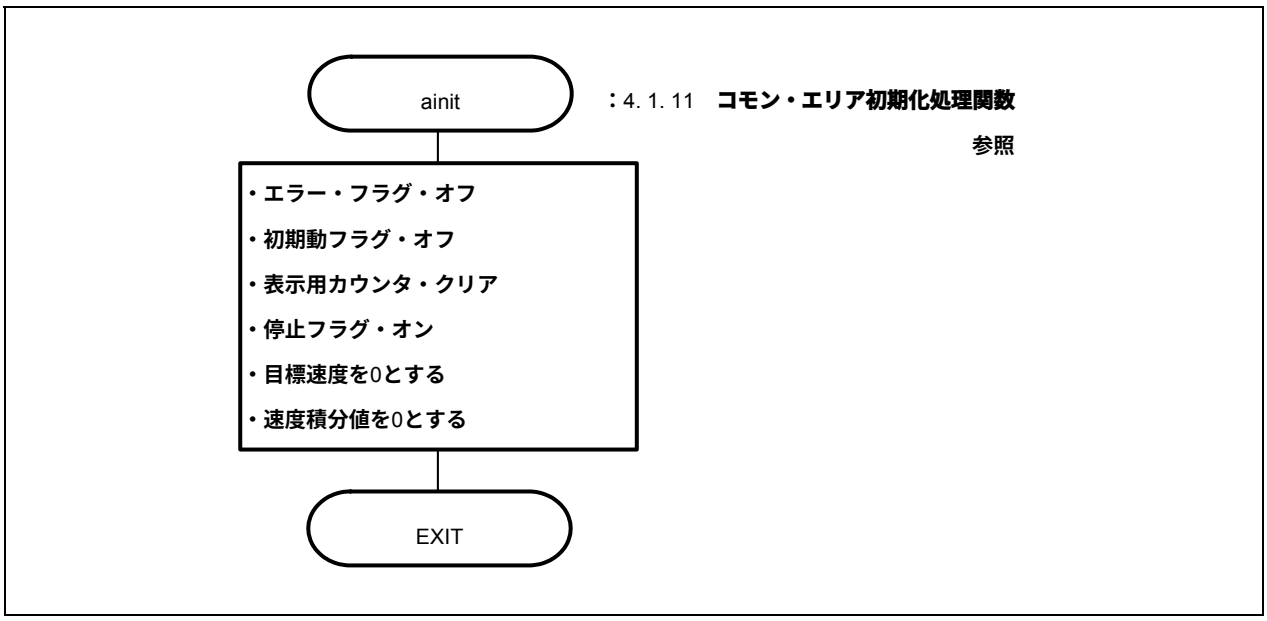
3. 4. 8　ハードウエア初期化

図3－22　ハードウエア初期化



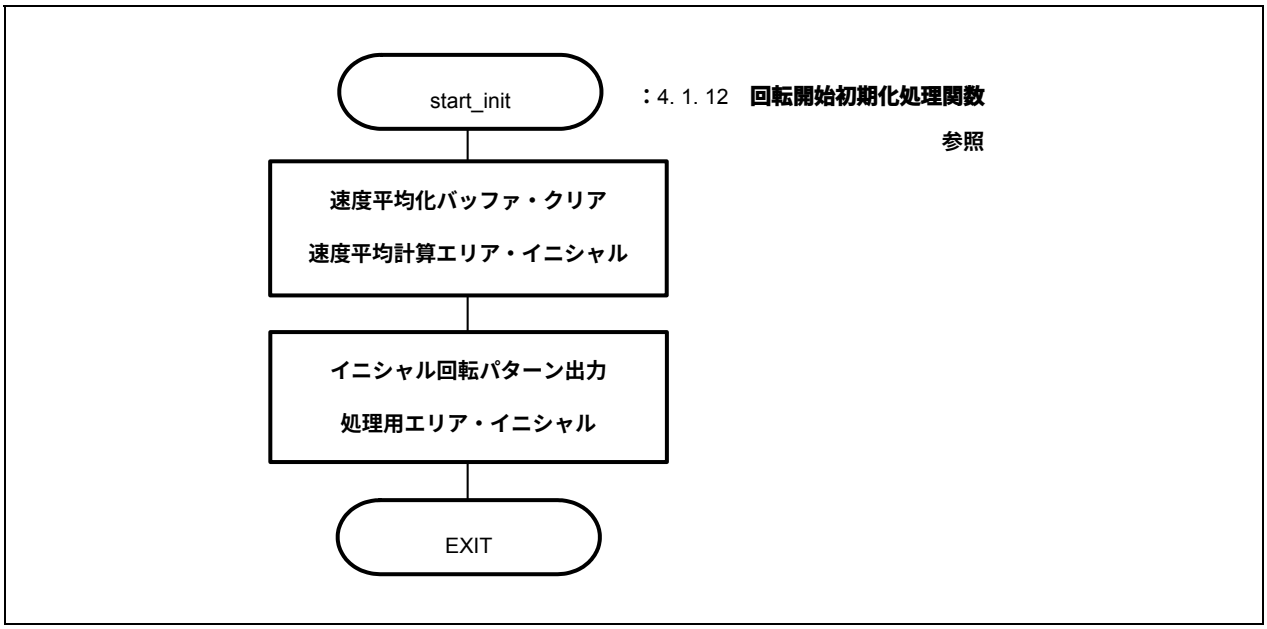
3. 4. 9　コモン・エリア初期化

図3－23　コモン・エリア初期化



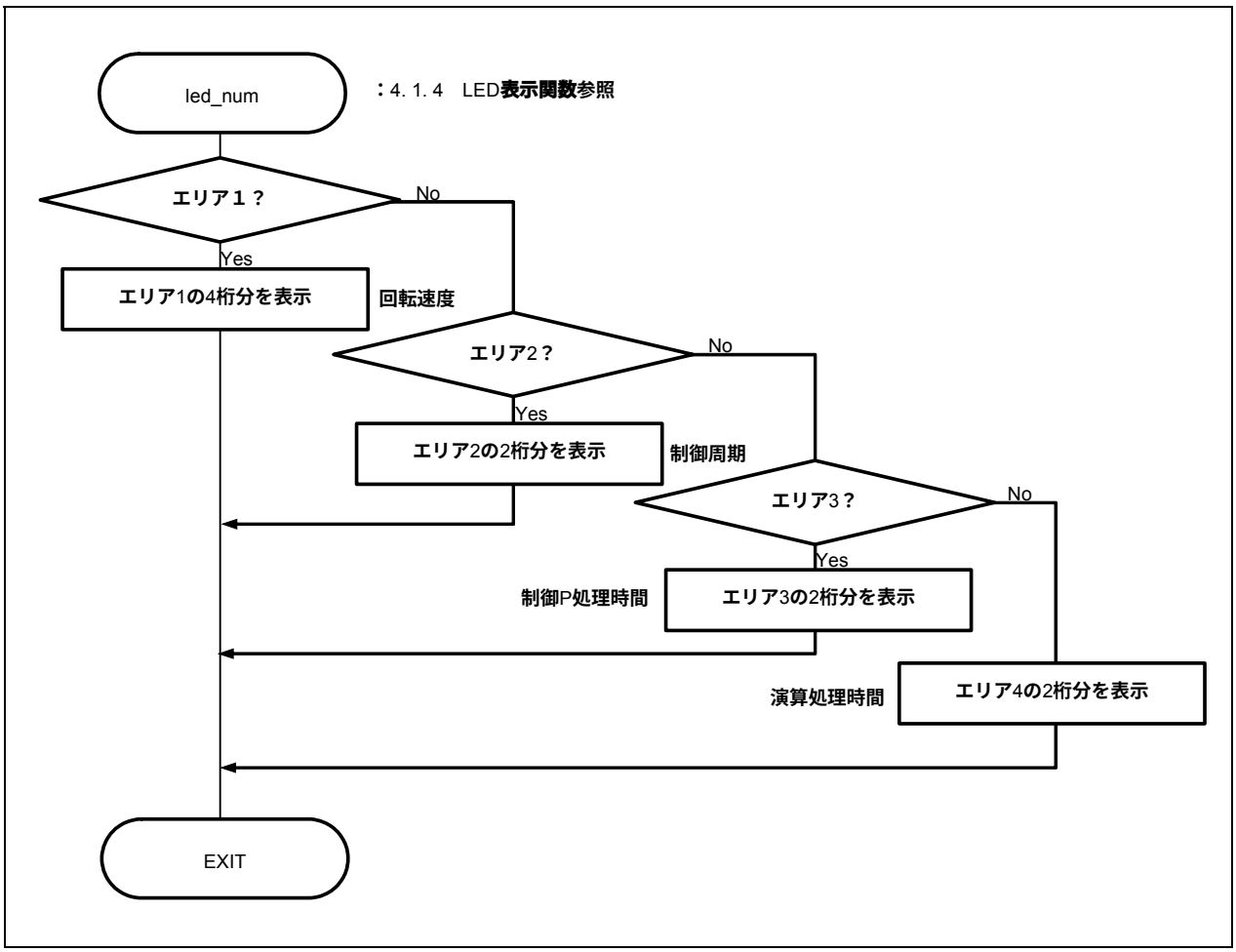
3. 4. 10　回転開始初期化

図3－24　回転開始初期化



3. 4. 11　LED表示

図3ー25　LED表示



3. 5　コモン・エリア

レファレンス・システムで使用する主なコモン・エリアを示します。

表3－2　コモン・エリア一覧

シンボル	型	用　途	設定値
error_flag	unsigned char	エラー・フラグ	0　：エラーなし ERR_NO1 ：過電流 ERR_NO3 ：速度差エラー
init_flag	unsigned char	初期動回転を表す	ON　：初期動回転中 OFF ：停止またはノーマル回転中
cont_time	unsigned short	割り込み処理時間	1 μs単位
cont_time1	unsigned short	ベクトル演算時間	1 μs単位
disp_co	unsigned short	表示用速度平均カウンタ	
volume	unsigned short	速度ボリューム値	
timer_count	unsigned short	時間待ち用カウンタ	10 ms単位
accel_count	unsigned short	加減速操作時間カウンタ	10 ms単位
stop_flag	unsigned short	停止フラグ	ON　：停止中 OFF：回転中
sum_speed	unsigned int	速度平均計算用合計値エリア	0，1, …
speed_co	unsigned int	速度平均計算用カウンタ	0，1, …
now_speed	signed int	現在速度	rpm
object_speed	signed int	目標速度	rpm
d_speed	signed int	表示速度	rpm
iua	signed short	U相電流	
iva	signed short	V相電流	
o_iqai	signed int	速度積分値	
base_position	signed int	速度推定基準点	
sa_time	unsigned int	速度計測用	
init_co	unsigned short	初期動回転時の出力切り替え監視カウンタ	0，1, …
init_pat	unsigned char	初期動回転出力パターン番号	0-5
init_upco	unsigned short	初期動回転出力回数テーブル番号	0-10
int_co	unsigned int	U, V, W割り込み回数カウンタ	0，1, …
pwm_value	signed int	PWM出力値	出力ビット・パターン

3. 6　テーブル類

(1) LED出力パターン

0-9までの表示パターン・データが入っています。

```
unsigned short led_pat[10] = { 0xfc. 0x60, 〜 };
```

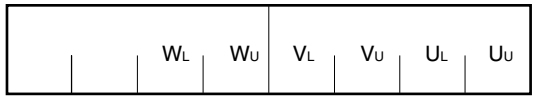
(2) 初期動CW出力パターン

CW初期動を行うときの出力パターンが入っています。

```
unsigned short cw_data[6][2] = { { 0x09, 0x00 }, { 0x21, 0x00 }, 〜 };注
```

注　下線部の値は，対象マイコンによって異なります。

図3－26　ビット割り付け



(3) 初期動CCW出力パターン

CCW初期動を行うときの出力パターンが入っています。

```
unsigned short ccw_data[6][2] = { { 0x18, 0x00 }, { 0x12, 0x00 }, 〜 };注
```

注　下線部の値は，対象マイコンによって異なります。

(4) 初期動回転パターン出力時間

初期動パターンをこのテーブルの割り込み回数ごとに回転速度を上げながら出力を行います。

```
unsigned short up_data[ ] = { 255, 242, 〜 };
```

(5) 通常CW回転時出力パターン

通常CW回転時ゼロクロス点の状態による出力パターンが入っています。

```
unsigned char run_cw_data[8][2] = { { 0x00, 0x00 }, { 0x21, 0x00 }, 〜 };注
```

注　下線部の値は，対象マイコンによって異なります。

(6) 通常CCW回転時出力パターン

通常CCW回転時ゼロクロス点の状態による出力パターンが入っています。

```
unsigned char run_ccw_data[8][2] = { { 0x00, 0x00 }, { 0x09, 0x00 }, 〜 };注
```

注　下線部の値は，対象マイコンによって異なります。

3. 7　定数定義

レファレンス・システムで使用する主な定数を示します。

シンボル	用　　途	値
KSP	速度比例定数	1100L
KSI	速度積分定数	50L
P	モータ極数	2
KSPGETA	速度比例定数ゲタ定数	10
KSIGETA	速度積分定数ゲタ定数	14
SGETA	sinゲタ定数	14
PWM_TS	PWM周期	100 μs
PWM_DATA	PWM設定値	PWM_TS／0.05
SPEED_MAX	最大速度	3000 rpm
SPEED_MINI	最低速度	800 rpm
SPEED_INIT	イニシャル回転速度	700 rpm
SA_SPEED_MAX	最大速度差	800 rpm
IQMAX	最大速度積分値	200000
MAX_I	電流最大値	800
TS	モータ制御周期	2000 μs
ACCEL_TIME	加減速時間定数　（単位：10 ms）	1
ACCEL_DATA	加減速増分回転数	40 rpm
WATCH_START	速度監視開始時間　（単位：10 ms）	500
ACCEL_VAL_1ST	初期加減速時間定数	50
ACCEL_VAL	加減速時間定数	3
ACCEL_SPD	加減速定数	50
PWM_INIT	PWMイニシャル値	PWM_DATA×3/4

第4章　プログラム・リスト

4. 1　プログラム・リスト（μPD78F0714用）

4. 1. 1　シンボル定義

```
/************************************************************************/
/*      コモン・エリア                                                  */
/************************************************************************/
unsigned char   ram_start ;
unsigned char   error_flag ;                    /* エラー・フラグ */
unsigned char   init_flag ;                     /* イニシャル・フラグ */
unsigned short  cont_time ;                     /* 割り込み制御時間 uSEC */
unsigned short  cont_time1 ;                    /* ベクトル演算時間 uSEC */
unsigned short  disp_co ;                       /* 割り込み制御時間表示タイマ */
unsigned short  volume ;                        /* ボリューム値 */
unsigned short  timer_count ;                   /* 時間待ち用カウンタ */
unsigned short  accel_count ;                   /* 加減速操作時間カウンタ */
unsigned char   stop_flag ;                     /* 停止フラグ */
signed int      sum_speed ;
signed int      speed_co ;
signed int      now_speed ;                     /* 現在速度 rms */
signed int      object_speed ;                  /* 目標速度 rms */
unsigned int    d_speed ;                       /* 表示速度 rms */
unsigned char   ram_end ;

const unsigned short led_pat[10] = { 0xfc, 0x60, 0xda, 0xf2, 0x66, 0xb6, 0xbe, 0xe0, 0xfe,
                                     0xe6 } ;
/************************************************************************/
/*      コモン・フラグ類                                                */
/************************************************************************/
extern unsigned char    ram_start ;
extern unsigned char    error_flag ;            /* エラー・フラグ */
extern unsigned char    init_flag ;             /* イニシャル・フラグ */
extern unsigned short   cont_time ;             /* 割り込み制御時間 uSEC */
extern unsigned short   cont_time1 ;            /* ベクトル演算時間 uSEC */
extern unsigned short   disp_co ;               /* 割り込み制御時間表示タイマ */
extern unsigned short   volume ;                /* ボリューム値 */
extern unsigned short   timer_count ;           /* 時間待ち用カウンタ */
extern unsigned short   accel_count ;           /* 加減速操作時間カウンタ */
extern unsigned char    stop_flag ;             /* 停止フラグ */
extern signed int       sum_speed ;
extern signed int       speed_co ;
extern signed int       now_speed ;             /* 現在速度 rms */
extern signed int       object_speed ;          /* 目標速度 rms */
extern unsigned int     d_speed ;               /* 表示速度 rms */
```

```
extern unsigned char    ram_end ;

extern const unsigned short led_pat[] ;

/************************************************************************/
/*      モータ・コモン定義                                              */
/************************************************************************/
extern signed short     iua ;                   /* U相電流 */
extern signed short     iva ;                   /* V相電流 */
extern signed long      o_iqai ;                /* 速度積分値エリア */
extern signed int       base_position ;         /* 速度推定値基準点 */
extern unsigned int     sa_time ;               /* 速度計測値 */
extern signed int       o_speed ;               /* 目標回転速度ゼロクロスパルス／100 mSEC */

extern unsigned short   init_co ;               /* イニシャル割り込みカウンタ */
extern unsigned char    init_pat ;              /* イニシャル・パターン・カウンタ */
extern unsigned short   init_upco ;             /* イニシャル・スピードアップ・カウンタ */
extern unsigned int     int_co ;                /* UVW割り込み回数カウンタ */
extern signed int       pwm_value ;             /* PWMの出力値 */
extern unsigned short out_pat;                  /* 出力パターン */
extern unsigned short CR01_int;                 /* 変換完了時間テンポラリ変数 */

extern const unsigned char cw_data[][2] ;
extern const unsigned char ccw_data[][2] ;
extern const unsigned char up_data[] ;
extern const unsigned char run_cw_data[][2] ;
extern const unsigned char run_ccw_data[][2] ;
```

4.1.2　定数定義

```
/************************************************************************/
/*      I/O                                                             */
/************************************************************************/
#define BASE_IO      0xc200000
#define LED11        0x03
#define LED12        0x02
#define LED13        0x01
#define LED14        0x00
#define LED21        0x05
#define LED22        0x04
#define LED31        0x07
#define LED32        0x06
#define LED41        0x09
#define LED42        0x08

#define DIPSW        0x0f
#define SW           0x0e
#define DA1          0x0a
#define DA2          0x0b
#define DA3          0x0c
```

```
#define WRESET       0x0d
#define MODE         0x10
/*************************************************************************/
/*      定　数                                                           */
/*************************************************************************/
#define ON           1
#define OFF          0
#define CW           1                          /* CW運転モード */
#define CCW          2                          /* CCW運転モード */
#define STOP         0                          /* 運転停止モード */
#define ERR_NO1      1                          /* 過電流エラー */
#define ERR_NO2      2                          /* 速度差エラー */
/*************************************************************************/
/*      モータ定数                                                       */
/*************************************************************************/
/* モータ定数 */
#define KSP             1100L                   /* 速度比例定数 */
#define KSI             50L                     /* 速度積分定数 */
#define P               2                       /* 極数 */

#define KSPGETA         10                      /* KP  ゲタ定数 */
#define KSIGETA         14                      /* KSI ゲタ定数 */
#define SGETA           14                      /* sin ゲタ定数 */

#define PWM_TS          100                     /* PWM 周期（uS）10kHz */
#define CPU_CLOCK_US    0.8                     // 1/TW0C-CLOCK*1000*1000
#define PWM_DATA        (PWM_TS/CPU_CLOCK_US/2) /* PWM 設定値 */
#define SPEED_MAX       3000                    /* 最大速度 3000 rpm */
#define SPEED_MINI      800                     /* 最低速度 800 rpm */
#define SPEED_INIT      700                     /* イニシャル回転速度 rpm */
#define SA_SPEED_MAX    800                     /* 最大速度差分 rpm */
#define IQAMAX          200000                  /* 最大速度積分値 */
#define MAX_I           800                     /* 電流 MAX値 */
#define TS              2000                    /* モータ制御時間間隔 uSEC */
#define ACCEL_TIME      1                       /* 加減速時間定数 10 mSEC */
#define ACCEL_DATA      40                      /* 加減速増分回転数 rpm */
#define WATCH_START     500                     /* 速度監視開始時間 10 mSEC */
#define ACCEL_VAL_1ST   50                      /* 初期加減速時間定数 */
#define ACCEL_VAL       3                       /* 加減速時間定数 */
#define ACCEL_SPD       50                      /* 加減速度定数 */
#define PWM_INIT        (int)(PWM_DATA*3/4)     /* PWM イニシャル値 */

#define LOW             0
#define HIGH            1
#define ENABLE          0

#define UP_TO_CPU(x)    ((x)*5U)                // us -> TM00値　プリスケーラで0.2us*5=1us
#define CPU_TO_US(x)    ((x+4U)/5U)             // TM00値  -> us

#define ADM_DEF         0x05                    // STOP,Select,time=3.6us,
```

```
                                                    Comparator=ON
#define ADM_START       (ADM_DEF|0x80)          // AD comversion start
#define ADM_STOP        ADM_DEF                 // AD comversion stop

/*************************************************************************/
/*      関数定数                                                         */
/*************************************************************************/
void            OUT_data( unsigned short reg, unsigned short data );
unsigned short  IN_data( int reg );
void            led_num( int no, long data ) ;
void            hinit( void );
void            ainit( void );
void            start_init( void );
/*************************************************************************/
/*      モータ関係コモン・エリア                                         */
/*************************************************************************/
signed short    iua ;                           /* U相電流 */
signed short    iva ;                           /* V相電流 */
signed long     o_iqai ;                        /* 速度積分値エリア */
signed int      base_position ;                 /* 速度推定値基準点 */
unsigned int    sa_time ;                       /* 速度計測値 */
signed int      o_speed ;                       /* 目標回転速度ゼロクロスパルス／100mSEC */

unsigned short  init_co ;                       /* イニシャル割り込みカウンタ */
unsigned char   init_pat ;                      /* イニシャル・パターン・カウンタ */
unsigned short  init_upco ;                     /* イニシャル・スピードアップ・カウンタ */
unsigned int    int_co ;                        /* UVW割り込み回数カウンタ */
signed int      pwm_value ;                     /* PWMの出力値 */
unsigned short out_pat ;                        /* 出力パターン */
unsigned short CR01_int ;                       /* 変換完了時間計算用テンポラリ変数 */

const unsigned char cw_data[6][2] = { {0x09,0x00},{0x21,0x00},{0x24,0x00},
                                      {0x06,0x00},{0x12,0x00},{0x18,0x00} };
const unsigned char ccw_data[6][2] = { {0x18,0x00},{0x12,0x00},{0x06,0x00},
                                       {0x24,0x00},{0x21,0x00},{0x09,0x00} };
const unsigned char up_data[] = { 255, 242, 229, 217, 206, 195, 185, 176, 166, 158,
                                  158, 158, 158, 158, 158, 158, 158, 158, 158, 158 } ;
const unsigned char run_cw_data[8][2] = { {0x00,0x00},{0x21,0x00},{0x06,0x00},{0x24,0x00},
                                     {0x18,0x00},{0x09,0x00},{0x12,0x00},{0x00,0x00}
                                      };
const unsigned char run_ccw_data[8][2] = { {0x00,0x00},{0x09,0x00},{0x24,0x00},
                                           {0x21,0x00}, {0x12,0x00},{0x18,0x00},
                                           {0x06,0x00},{0x00,0x00} };
```

4.1.3　メイン処理関数

```
#pragma     sfr
#pragma     EI
#pragma     DI

#include    <stdlib.h>
#include    "Common.h"
#include    "Motor.h"

/*****************************************************************************/
/*      3相モータ制御プログラム                                              */
/*****************************************************************************/
void    main()
{
unsigned char   proc_no ;                       /* 現在処理番号 */
signed int      speed ;                         /* 指示速度 rms */
signed int      accel_spd ;
int             sw, sw_mode ;
/* */
   hinit() ;                                    /* ハードウエア初期化 */
   ainit() ;                                    /* 使用エリア初期化 */
   proc_no = 0 ;
   EI();
   while( 1 ) {
      accel_spd = ( SPEED_MAX - SPEED_MINI ) / 100;
      speed  = ( ( SPEED_MAX - SPEED_MINI ) * (long) volume  / 1024 )
               + SPEED_MINI ;                   /* ボリュームによる指示速度計算 */
      sw =  ~IN_data( SW ) & 0x07 ;             /* 運転ボタン読み込み */
      if  ( sw == 1 ) {
         sw_mode = CW ;
      } else if  ( sw == 2 ) {
         sw_mode = CCW ;
      } else if  ( sw == 4 ) {
         sw_mode = STOP ;
      }
      o_speed = ( int ) (abs(object_speed)/(100/P)); /* rmp=>ゼロクロスパルス／100mSEC */
      switch( proc_no ) {
/* STOP 処理 */
         case 0 :
            if  ( sw_mode == CW ) {
               DI() ;
               object_speed  = SPEED_MINI ;     /* 目標速度を最低値に設定 */
               stop_flag = OFF ;
               timer_count = WATCH_START ;      /* 速度監視開始時間5 SEC設定 */
               accel_count = ACCEL_VAL_1ST ;    /* 加減速カウンタ設定 */
               init_flag = 2 ;                  /* CCW初起動要求 */
               start_init() ;                   /* 回転スタート・イニシャル */
               EI() ;
               proc_no = 1 ;                    /* 次の処理番号設定 */
```

```
            } else if  ( sw_mode == CCW ) {
                DI() ;
                stop_flag = OFF ;              /* 停止フラグOFF */
                object_speed = -SPEED_MINI ;   /* 目標速度を最低値に設定 */
                timer_count = WATCH_START ;    /* 速度監視開始時間5 SEC設定 */
                accel_count = ACCEL_VAL_1ST ;  /* 加減速カウンタ設定 */
                init_flag = 3 ;                /* CCW初起動要求 */
                start_init() ;                 /* 回転スタート・イニシャル */
                EI() ;
                proc_no = 4 ;                  /* CCW 処理番号設定 */
            }
            break ;
/* CW 処理 加速*/
        case 1 :
            if  ( accel_count == 0 ) {
                accel_count = ACCEL_VAL ;      /* 加減速カウンタ設定 */
                if  ( object_speed  < speed ) {
                    object_speed += accel_spd ;
                    if  ( object_speed > speed ) object_speed = speed;
                    timer_count = WATCH_START ;/* 速度監視開始時間5 SEC設定 */
                } else if ( object_speed  > speed ) {
                    object_speed -= accel_spd ;
                    if  ( object_speed < speed ) object_speed = speed;
                    timer_count = WATCH_START ;/* 速度監視開始時間5 SEC設定 */
                } else {
                    proc_no = 2 ;              /* 定速処理 */
                }
            }
            if  ( (sw_mode == CCW) || (sw_mode == STOP) ) {
                proc_no = 3 ;                  /* 減速中 処理番号設定 */
            }
            break ;
/* CW 処理 定速*/
        case 2 :
            object_speed = speed ;
            if  ( (sw_mode == CCW) || (sw_mode == STOP) ) {
                proc_no = 3 ;                  /* 減速中 処理番号設定 */
            }
            break ;
/* CW停止 処理 */
        case 3 :
            if  ( accel_count == 0 ) {
                accel_count = ACCEL_VAL ;      /* 加減速カウンタ設定 */
                if  ( object_speed  > SPEED_MINI ) {
                    object_speed -= accel_spd ;
                    if  ( object_speed < SPEED_MINI ) object_speed = SPEED_MINI;
                    timer_count = WATCH_START ;/* 速度監視開始時間5 SEC設定 */
                } else {
                    stop_flag = ON ;           /* 停止フラグON */
                    proc_no = 0 ;              /* 停止 処理番号設定 */
```

```
                }
            }
            break ;
/* CCW 処理 加速*/
        case 4 :
            if  ( accel_count == 0 ) {
                accel_count = ACCEL_VAL ;       /* 加減速カウンタ設定 */
                if  ( object_speed  < -speed ) {
                    object_speed += accel_spd ;
                    if  ( object_speed > -speed ) object_speed = -speed;
                    timer_count = WATCH_START ; /* 速度監視開始時間5 SEC設定 */
                } else if ( object_speed  > -speed ) {
                    object_speed -= accel_spd ;
                    if  ( object_speed < -speed ) object_speed = -speed;
                    timer_count = WATCH_START ; /* 速度監視開始時間5 SEC設定 */
                } else {
                    proc_no = 5 ;               /* 定速処理 */
                }
            }
            if  ( (sw_mode == CW) || (sw_mode == STOP) ) {
                proc_no = 6 ;                   /* 減速中 処理番号設定 */
            }
            break ;
/* CCW 処理 定速*/
        case 5 :
            object_speed = -speed ;
            if  ( (sw_mode == CW) || (sw_mode == STOP) ) {
                proc_no = 6 ;                   /* 減速中 処理番号設定 */
            }
            break ;
/* CCW停止 処理 */
        case 6 :
            if  ( accel_count == 0 ) {
                accel_count = ACCEL_VAL ;       /* 加減速カウンタ設定 */
                if  ( object_speed  < -SPEED_MINI ) {
                    object_speed += accel_spd ;
                    if  ( object_speed > -SPEED_MINI ) object_speed = -SPEED_MINI;
                    timer_count = WATCH_START ; /* 速度監視開始時間5 SEC設定 */
                } else {
                    stop_flag = ON ;            /* 停止フラグON */
                    proc_no = 0 ;               /* 停止 処理番号設定 */
                }
            }
            break ;
    }

    if  ( ( proc_no == 2 ) || ( proc_no == 5 ) ) {
        if  (  timer_count == 0 ) {
            if  ( abs( object_speed - now_speed ) > SA_SPEED_MAX ) {
                error_flag = ERR_NO2 ;          /* エラーNOセット */
```

```
            }
        }
    }

    if  ( disp_co == 0 ) {
        led_num(1, d_speed / P );            /* 回転数 */
        d_speed = 0 ;
        disp_co = 100 ;
        if  ( abs(now_speed) == 0 ) {
            disp_co = 0;
        }
        led_num(2, 1000/PWM_TS );            /* キャリア周波数 */
        led_num(3, cont_time / 10 );         /* 処理時間全体 */
        led_num(4, cont_time1 / 10);         /* 演算処理時間 */
    }
    if  ( error_flag ) {
        while( 1 ) {
            OUT_data( LED41, ~0x00 ) ;       /* LED表示OFF */
            OUT_data( LED42, ~0x00 ) ;
            timer_count = 50 ;
            while( timer_count ) ;
            if  ( error_flag == ERR_NO1 ) {
                OUT_data( LED41, ~0x9e ) ;   /* E1表示 */
                OUT_data( LED42, ~0x60 ) ;
            } else if ( error_flag == ERR_NO2 ) {
                OUT_data( LED41, ~0x9e ) ;   /* E2表示 */
                OUT_data( LED42, ~0xda ) ;
            } else {
                OUT_data( LED41, ~0x9e ) ;   /* E3表示 */
                OUT_data( LED42, ~0xf2 ) ;
            }
            timer_count = 50 ;
            while( timer_count ) ;
        }
    }
  }
}
```

4.1.4　LED表示関数

```
/*****************************************************************************/
/*      LED値表示サブルーチン                                                */
/*              no   : 表示エリア番号(1-4)                                   */
/*              data : 表示データ(0-99)                                      */
/*****************************************************************************/
void led_num( int no, long data )
{
   if  ( no == 1 ) {
      data = data % 10000;
      OUT_data( LED11, ~led_pat[data/1000]&0xff ) ;
      OUT_data( LED12, ~led_pat[(data%1000)/100]&0xff ) ;
      OUT_data( LED13, ~led_pat[(data%100)/10]&0xff ) ;
      OUT_data( LED14, ~led_pat[data%10]&0xff ) ;
   } else if  ( no == 2 ) {
      OUT_data( LED21, ~led_pat[(data%100)/10]&0xff ) ;
      OUT_data( LED22, ~led_pat[data%10]&0xff ) ;
   } else if  ( no == 3 ) {
      OUT_data( LED31, ~led_pat[(data%100)/10]&0xff ) ;
      OUT_data( LED32, ~led_pat[data%10]&0xff ) ;
   } else {
      OUT_data( LED41, ~led_pat[(data%100)/10]&0xff ) ;
      OUT_data( LED42, ~led_pat[data%10]&0xff ) ;
   }
}
/*****************************************************************************/
/*      外部I/O出力サブルーチン                                              */
/*              reg  : 出力レジスタ番号                                      */
/*              data : 出力データ                                            */
/*****************************************************************************/
void    OUT_data( unsigned short reg, unsigned short data )
{
        if  ( reg == WRESET ) {
                P3.3 = 0;
                data = 1;                       /* ダミー・ステップ */
                P3.3 = 1;
        } else {
                P5 = (unsigned char) reg ;
                P4 = (unsigned char) ( data & 0xff) ;
                PM4 = 0x00 ;
                P54 = LOW ;

                P54 = HIGH ;
        }
}
/*****************************************************************************/
/*      外部I/O入力サブルーチン                                              */
/*              reg  : 入力レジスタ番号                                      */
/*****************************************************************************/
```

```
unsigned short  IN_data( int reg )
{
unsigned char *po;
/*  */
        if    ( reg == SW ) {
                return P3;
        } else {
                return 0;
        }
}
```

4.1.5　モータ制御割り込み処理関数

```
#pragma     sfr
#pragma     EI
#pragma     DI
#pragma     INTERRUPT INTTM00 int_ETC rb1
#pragma     INTERRUPT INTTM01 int_MOTOR rb1
#pragma     INTERRUPT INTAD int_AD0 rb2
#pragma     INTERRUPT INTP1 int_INTP rb1
#pragma     INTERRUPT INTP2 int_INTP rb1
#pragma     INTERRUPT INTP3 int_INTP rb1
#pragma     INTERRUPT INTTM51 int_DELAY rb3

#include    <stdlib.h>
#include    "Common.h"
#include    "Motor.h"
/*****************************************************************************/
/*      モータ制御タイマ割り込み処理                                         */
/*****************************************************************************/
_ _interrupt
void    int_MOTOR(void)
{
    CR01_int = CR01;
    CR01 = US_TO_CPU(TS)+CR01 ;
    ADS = 0x00 ;                                    // ANI0 選択
    ADM = ADM_START ;
}
/*****************************************************************************/
/*      モータ制御処理                                                       */
/*****************************************************************************/
void    Motor_CONT(void)
{
signed int      sa_speed, o_iqap, o_iqa ;
signed int      s_time, cow ;
unsigned char   wk ;
/* */
/*****************************************************************************/
/*      速度の計算処理                                                       */
/*****************************************************************************/
```

```
   sum_speed += int_co ;
   int_co = 0 ;
   if  ( --speed_co == 0 ) {
      speed_co = 100000 / TS ;                  /* 100 mSECカウンタ値設定 */
      now_speed = sum_speed ;                   /* ゼロクロスパルス／100mSEC */
      sum_speed = 0 ;
   }

   if  ( ( stop_flag == OFF ) && ( error_flag == 0 ) ) {
      s_time = TM00 ;
      OUT_data( WRESET, 0 ) ;                   /* ウォッチドック・タイマRESET */
/*************************************************************************/
/*      イニシャル回転処理                                               */
/*************************************************************************/
      if  ( init_flag ) {
         cow = init_upco ;
         if  ( cow > 4 ) cow = 4;
         if  ( ++init_co > ( (long)up_data[ cow ] * 34000L / ( SPEED_INIT * TS ) ) ) {
            init_co = 0 ;
            if  ( init_flag == 2 ) {
               wk  = cw_data[ init_pat++  ][0] ;
            } else {
               wk  = ccw_data[ init_pat++ ][0] ;
            }
                     RTPM01 = ~wk;

            if  ( init_pat >= 6 ) {
               init_pat = 0 ;
               if  ( init_upco > 14 ) {
                  init_flag = 0 ;
               } else {
                  init_upco++ ;
               }
            }
         }
      } else {
/*************************************************************************/
/*      ノーマル回転処理                                                 */
/*************************************************************************/
         se_speed = o_speed - now_speed ;

         o_iqap  = (int) ( ( (now_speed + sa_speed) * KSP ) >> KSPGETA ) ;
         o_iqa   = (int) (o_iqap + ( o_iqai >> KSIGETA ) ) ;

         if  ( o_iqai > IQAMAX ) {
            o_iqai = IQAMAX ;
         } else if ( o_iqai < -IQAMAX ) {
            o_iqai = -IQAMAX ;
         } else {
            o_iqai += ( KSI * sa_speed ) ;
```

```
          }

          pwm_value = o_iqa ;
          if  ( pwm_value <= 0 ) {
              pwm_value = 1 ;
          } else if  ( pwm_value >= PWM_DATA  ) {
              pwm_value = ( PWM_DATA  ) - 1 ;
          }
                TW0BFCM0 =  PWM_DATA - pwm_value ;
                TW0BFCM1 =  PWM_DATA - pwm_value ;
                TW0BFCM2 =  PWM_DATA - pwm_value ;

                cont_time1 = CPU_TO_US( TM00 - s_time );        // uSECに変換
       }
   } else {
                CE0 = OFF ;                                     // PWM出力OFF
                RTPM01 = 0xFF ;                                 // ラッチデータ反転出力
       now_speed = 0;
       cont_time1 = 0;
   }
}
```

4.1.6　ゼロクロス割り込み処理関数

```
/*************************************************************************/
/*      Uゼロクロス点割り込み                                            */
/*      Vゼロクロス点割り込み                                            */
/*      Mゼロクロス点割り込み                                            */
/*************************************************************************/
_interrupt void int_INTP(void)
{
   int co;
/* */
   if  ( ( ( init_flag == 0 ) && ( stop_flag == OFF) ) ) {
        if  ( object_speed < 0 ) {
                out_pat  = run_ccw_data[ ( P0 >> 1 ) & 0x07 ][0] ;
        } else {
                out_pat  = run_cw_data [ ( P0 >> 1 ) & 0x07 ][0] ;
        }

        co = 781 / abs(now_speed);                              /* 30度分のカウンタ値 */
        if  ( co == 0 ) co = 1;
        CR51 = co;
        TMC51 = 0x80;                                           // タイマ・スタート
   }
   int_co++ ;
}
```

4.1.7　10 mSECインターバル割り込み処理関数

```
/*****************************************************************************/
/*      その他のタイマ割り込み処理（10 mSECインターバル）                    */
/*****************************************************************************/
_interrupt void int_ETC(void)
{
        EI() ;
        CR00 = US_TO_CPU (10000) + CR00 ;

/* wait タイマ処理 */
   if  ( timer_count != 0 ) {
       timer_count -= 1 ;
   }
/* 加減速 タイマ処理 */
   if  ( accel_count != 0 ) {
       accel_count -= 1 ;
   }
/* */
   if  ( disp_co != 0 ) {
       d_speed += now_speed ;
       disp_co -= 1 ;
   }
}
```

4.1.8　遅延制御割り込み処理関数

```
/*****************************************************************************/
/*     30度遅延タイマ割り込み処理                                            */
/*****************************************************************************/
__interrupt void   int_DELAY(void)
{
        TMC51 = 0x00;                                          // タイマ停止
        if  ( ( ( init_flag == 0 ) && ( stop_flag == OFF) ) ) {
                RTPM01 = ~out_pat ;
        }
}
```

4.1.9　A/Dコンバータ割り込み処理関数

```
/*****************************************************************************/
/*      U相電流&速度ボリューム用A/Dコンバータ割り込み処理                    */
/*      ボリューム用A/Dコンバータ割り込み処理                                */
/*****************************************************************************/
_interrupt void int_AD0(void)
{
        ADM = ADM_STOP ;
        IF1H.4 = 0 ;
        EI() ;
        switch ( ADS & 0x07 ) {
        case 0x00 :
                // ANI0 comversion end
                iua = ((( ADCR >> 6 ) & 0x3ff ) - 0x200) ;
                if  ( abs(iua) > MAX_I ) {
                        CE0 = OFF ;             // PWM出力OFF
                        RTPM01 = 0xFF ;         // ラッチデータ反転出力
                        error_flag = ERR_NO1 ;  /* エラーNOセット */
                }
                ADS = 0x01 ;                    // ANI1を選択
                ADM = ADM_START ;
                break ;
        case 0x01 :
                // ANI1 comversion end
                iva = (( ADCR & 0x3ff ) - 0x200) ;
                if  ( abs(iva) > MAX_I ) {
                        CE0 = OFF ;             // PWM出力OFF
                        RTPM01 = 0xFF ;         // ラッチデータ反転出力
                        error_flag = ERR_NO1 ;  /* エラーNOセット */
                }
                ADS = 0x02 ;                    // ANI2を選択
                ADM = ADM_START ;
                break ;
        default :
                // ANI2 comversion end
                volume = 1023 - (( ADCR >> 6 ) & 0x3ff ) ;      // ボリューム値セット

                EI() ;
                Motor_CONT () ;
                cont_time = CPU_TO_US(TM00-CR01_int) ;          // uSECに変換
                break ;
        }
}
```

4. 1. 10　ハードウエア初期化処理関数

```
/*****************************************************************************/
/*      ハードウエア（周辺I/O)初期化                                         */
/*****************************************************************************/
void    hinit( void )
{
    IMS = 0xC8 ;                                    // メモリ・エリア

    /*  WDT停止 */
    WDTM = 0x77 ;

    /*  メイン・クロック発振，高速モード設定 */
    while ( OSTC != 0x1f ) ;                        // 発振安定まで待つ
    MCM = 0x03 ;                                    // X1入力クロックを選択

    // FPGAアクセス ポート・モード・レジスタ初期化
    PM5 = 0xE0 ;
    PM4 = 0x00 ;

    // LED OFF
    OUT_data( LED11, 0xff ) ;
    OUT_data( LED12, 0xff ) ;
    OUT_data( LED13, 0xff ) ;
    OUT_data( LED14, 0xff ) ;
    OUT_data( LED21, 0xff ) ;
    OUT_data( LED22, 0xff ) ;
    OUT_data( LED31, 0xff ) ;
    OUT_data( LED32, 0xff ) ;
    OUT_data( LED41, 0xff ) ;
    OUT_data( LED42, 0xff ) ;

    // 16ビット・タイマ  TM00設定
    TMC00 = 0x00 ;                                  // 動作を禁止してから設定
    CRC00 = 0x00 ;                                  // CR00,CR01コンペア動作
    TOC00 = 0x00 ;                                  // 出力禁止
    PRM00 = 0x01 ;                                  // fX/2^2 (5 MHz = 0.2us)
    CR00 = US_TO_CPU (10000) + TM00 ;               // 10 mSEC
    CR01 = US_TO_CPU ( TS ) + TM00 ;                // TS uSEC
    TMC00 = 0x04 ;                                  // フリーラン，オーバフローなし

    // 8ビット・タイマ TM51設定
    TCL51 = 0x05 ;
    CR51  = 0xff ;
    TMC51 = 0x80 ;

    // TMW0 設定
    TW0C = 0x20 ;                                   // タイマ停止，fx/16 = 1.25 MHz，
                                                    // アンダフローごとに割り込み発生
    TW0M = 0x04 ;                                   //
```

```
    TW0OC = 0x00 ;                          // TW0TO0-TW0TO5出力許可

/* リアルタイム出力ポート設定 */
    RTBH01 = 0xFF ;                         // 出力データ・ラッチ
    RTBL01 = 0xFF ;                         // 出力データ・ラッチ
    DCCTL01 = 0xB0 ;                        // 7: PWM変調RTP出力
                                            // 6: RTP10,12,14 PWM変調動作許可
                                            // 5: RTP11,13,15 PWM変調動作許可
                                            // 4: 反転許可
    // A/D 設定
    ADM = ADM_DEF ;                         // 動作停止，セレクト・モード，
                                            // 3.6us，コンパレータON
    ADS = 0x00 ;
    PFM = 0x00 ;
    PFT = 0x00 ;

    // 外部割り込み設定
    EGP = 0x0e ;                            // 立ち上がりエッジ
    EGN = 0x0e ;                            // 立ち下がりエッジ

    // 割り込みマスク設定
    TMMK00 = ENABLE ;
    TMMK01 = ENABLE ;
    PMK1 = ENABLE ;
    PMK2 = ENABLE ;
    PMK3 = ENABLE ;
    ADMK = ENABLE ;
    TMMK51 = ENABLE ;

    // 優先順位設定
    ADPR = 0 ;
    PPR1 = 0 ;
    PPR2 = 0 ;
    PPR3 = 0 ;
    TMPR51 = 0 ;

    // ポート3設定
    PM3 = 0xF7 ;
}
```

4.1.11　コモン・エリア初期化処理関数

```
/************************************************************************/
/*      コモン・エリア初期化                                            */
/************************************************************************/
void    ainit( void )
{
/* フラグ類初期化 */
    error_flag = 0 ;                        /* エラー・フラグ・クリア */
    init_flag = OFF ;                       /* イニシャル・フラグOFF */
```

```
    disp_co = 100 ;
    d_speed = 0 ;
/* モータ制御エリア初期化 */
    stop_flag   = ON ;                          /* 停止フラグON */
    object_speed = 0 ;                          /* 目標速度を0とする */
    o_iqai = 0 ;                                /* 速度積分値を0とする */
}
```

4. 1. 12　回転開始初期化処理関数

```
/*****************************************************************************/
/*      回転開始初期化                                                       */
/*****************************************************************************/
void    start_init( void )
{
    int  i;
/*  */
    sum_speed = 0 ;
    speed_co = 100000 / TS ;

    init_co = 0 ;
    init_pat = 0 ;
    init_upco = 0 ;

    // リアルタイム出力ポート設定
    RTPM01    = 0xFF ;                          // ラッチ・データ反転出力
    RTPC01    = 0xA0 ;                          // 7: 動作許可
                                                // 5: 6ビット×1チャネル
    // PWM周期をイニシャル回転用に設定
    TW0BFCM3  = PWM_DATA ;
    TW0BFCM0  = PWM_INIT ;
    TW0BFCM1  = PWM_INIT ;
    TW0BFCM2  = PWM_INIT ;
    pwm_value = PWM_DATA - PWM_INIT ;
    CE0 = ON ;                                  // PWM出力ON
}
```

〔メ　モ〕

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。