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

V850シリーズによるインバータ制御

エンコーダによるベクトル制御編

V850E/IA1
V850E/IA2
V850E/IA3
V850E/IA4

資料番号　U17324JJ1V0AN00（第1版）
発行年月　September 2004 NS　CP(K)

〔メ　モ〕

目次要約

第1章　制御方式　…　12

第2章　ハードウエア構成　…　15

第3章　ソフトウエア構成　…　27

第4章　プログラム・リスト　…　56

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。

本製品のうち，外国為替及び外国貿易法の規定により規制貨物等（または役務）に該当するものについては，日本国外に輸出する際に，同法に基づき日本国政府の輸出許可が必要です。

非該当品　　：μPD70F3114, 70F3114(A), 70F3116, 70F3116(A), 70F3116(A1), 70F3184, 70F3186

ユーザ判定品：μPD703114, 703114(A), 703116, 703116(A), 703116(A1), 703183, 703185, 703186

- **本資料に記載されている内容は2004年9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，**V850E/IA1, V850E/IA2, V850E/IA3, V850E/IA4**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。**

- V850E/IA1
  標準品：μPD703116, 70F3116
  特別品：μPD703116(A), 703116(A1), 70F3116(A), 70F3116(A1)
- V850E/IA2
  標準品：μPD703114, 70F3114
  特別品：μPD703114(A), 70F3114(A)
- V850E/IA3
  標準品：μPD703183, 70F3184
- V850E/IA4
  標準品：μPD703185, 703186, 70F3186

目　　的　**このアプリケーション・ノートでは，**V850E/IA1, V850E/IA2, V850E/IA3, V850E/IA4**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，エンコーダによるベクトル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。**

注意1.　**このマニュアル中の使用例は，一般電子機器用の『標準』品質水準品用に作成してあります。『特別』品質水準を要求する用途にこのマニュアル中の使用例を使用する場合は，実際に使用する各部品および回路について，その品質水準についてご検討のうえご使用ください。**

2.　**特別品のマニュアルとして使用する場合には，次のように読み替えてください。**

μPD703114　→　μPD703114(A)
μPD70F3114　→　μPD70F3114(A)
μPD703116　→　μPD703116(A), 703116(A1)
μPD70F3116　→　μPD70F3116(A), 70F3116(A1)

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V850E/IA1　**ユーザーズ・マニュアル　ハードウエア編，**V850E/IA2　**ユーザーズ・マニュアル　ハードウエア編，**V850E/IA3, V850E/IA4　**ユーザーズ・マニュアル　ハードウエア編**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V850E1　**ユーザーズ・マニュアル　アーキテクチャ編**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。

V850E/IA1, V850E/IA2, V850E/IA3, V850E/IA4**に関する資料**

資　料　名	資料番号
V850E1　**ユーザーズ・マニュアル　アーキテクチャ編**	U14559J
V850E/IA1　**ユーザーズ・マニュアル　ハードウエア編**	U14492J
V850E/IA2　**ユーザーズ・マニュアル　ハードウエア編**	U15195J
V850E/IA3, V850E/IA4　**ユーザーズ・マニュアル　ハードウエア編**	U16543J
V850E/IA1, V850E/IA2　**アプリケーション・ノート　ベクトル演算によるACモータ・インバータ制御編**	U14868J
V850**シリーズ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**	U17209J
V850**シリーズによるインバータ制御　アプリケーション・ノート　エンコーダによるベクトル制御編**	**このマニュアル**

開発ツールに関する資料（ユーザーズ・マニュアル）

資　料　名		資料番号
IE-V850E-MC（V850E/IA1, V850E/IA2用インサーキット・エミュレータ）		U14487J
QB-V850EIA4（V850E/IA3, V850E/IA4用インサーキット・エミュレータ）		U17167J
IE-V850E1-CD-NW（V850E/IA3, V850E/IA4用PCMCIAカード型オンチップ・ディバグ・エミュレータ）		U16647J
IE-703116-MC-EM1（V850E/IA1用インサーキット・エミュレータ・オプション・ボード）		U14700J
IE-703114-MC-EM1（V850E/IA2用インサーキット・エミュレータ・オプション・ボード）		U16533J
CA850　Ver.3.00　Cコンパイラ・パッケージ	操作編	U17293J
	C言語編	U17291J
	アセンブリ言語編	U17292J
	リンク・ディレクティブ編	U17294J
PM plus　Ver.6.00		U17178J
ID850　Ver.2.50　統合ディバッガ	操作編	U16217J
ID850NWC　Ver.2.51　統合ディバッガ（V850E/IA3, V850E/IA4用）	操作編	U16525J
ID850QB　Ver.2.80　統合ディバッガ（V850E/IA3, V850E/IA4用）	操作編	U16973J
SM850　Ver.2.50　システム・シミュレータ（V850E/IA1, V850E/IA2用）	操作編	U16218J
SM850　Ver.2.00以上　システム・シミュレータ（V850E/IA1, V850E/IA2用）	外部部品ユーザ・オープン・インタフェース仕様編	U14873J
SM plus　システム・シミュレータ（V850E/IA1, V850E/IA2用）	操作編	U17246J
	ユーザ・オープン・インタフェース編	U17247J
RX850　Ver.3.13以上　リアルタイムOS	基礎編	U13430J
	インストレーション編	U13410J
	テクニカル編	U13431J
RX850 Pro　Ver.3.15　リアルタイムOS	基礎編	U13773J
	インストレーション編	U13774J
	テクニカル編	U13772J
RX-NET　Ver.1.30　TCP/IPライブラリ		U15083J
RD850　Ver.3.01　タスク・ディバッガ		U13737J
RD850 Pro　Ver.3.01　タスク・ディバッガ		U13916J
AZ850　Ver.3.20　システム・パフォーマンス・アナライザ		U14410J
PG-FP4　フラッシュ・メモリ・プログラマ		U15260J

目　次

第1章　制御方式 … 12

1. 1　**ベクトル制御の原理** … 12

第2章　ハードウエア構成 … 15

2. 1　**構　　成** … 15
2. 2　**回 路 図** … 17

第3章　ソフトウエア構成 … 27

3. 1　**制御ブロック** … 27
3. 2　**速度制御** … 28
3. 3　**エンコーダによる位置速度検出** … 28
3. 4　**電流制御** … 29
3. 5　3**相電圧変換** … 29
3. 6　PWM**変換** … 30
3. 7　**周辺**I/O … 30
3. 8　**ソフトウエア処理構造** … 32
3. 9　**フロー・チャート** … 34
　3. 9. 1　**メイン処理** … 34
　3. 9. 2　LED**表示** … 42
　3. 9. 3　**モータ制御タイマ割り込み処理** … 43
　3. 9. 4　**モータ制御処理** … 43
　3. 9. 5　10 mSEC**インターバル割り込み処理** … 48
　3. 9. 6　A/D**コンバータ・チャネル**0**割り込み処理** … 49
　3. 9. 7　A/D**コンバータ・チャネル**1**割り込み処理** … 50
　3. 9. 8　**ハードウエア初期化** … 51
　3. 9. 9　**コモン・エリア初期化** … 52
　3. 9. 10　**回転開始初期化** … 52
　3. 9. 11　sin**計算** … 53
3. 10　**コモン・エリア** … 54
3. 11　**テーブル類** … 55
3. 12　**定数定義** … 55

第4章　プログラム・リスト … 56

4. 1　**プログラム・リスト（**V850E/IA1**用）** … 56
　4. 1. 1　**シンボル定義** … 56
　4. 1. 2　**定数定義** … 57
　4. 1. 3　**割り込みハンドラ設定** … 59
　4. 1. 4　**スタートアップ・ルーチン設定** … 60
　4. 1. 5　**メイン処理関数** … 63
　4. 1. 6　LED**表示関数** … 67
　4. 1. 7　**モータ制御割り込み処理関数** … 68

4. 1. 8　10 mSEC**インターバル割り込み処理関数**　…　71
4. 1. 9　A/D**コンバータ割り込み処理関数**　…　71
4. 1. 10　**ハードウエア初期化処理関数**　…　72
4. 1. 11　**コモン・エリア初期化処理関数**　…　73
4. 1. 12　**回転開始初期化処理関数**　…　74
4. 1. 13　sin**計算処理関数**　…　74
4. 1. 14　V850E/IA1**用のリンク・ディレクティブ・ファイル**　…　75
4. 2　**プログラム・リスト（**V850E/IA2**用）**　…　77
4. 2. 1　**シンボル定義**　…　77
4. 2. 2　**定数定義**　…　78
4. 2. 3　**割り込みハンドラ設定**　…　80
4. 2. 4　**スタートアップ・ルーチン設定**　…　81
4. 2. 5　**メイン処理関数**　…　84
4. 2. 6　LED**表示関数**　…　88
4. 2. 7　**モータ制御割り込み処理関数**　…　89
4. 2. 8　10 mSEC**インターバル割り込み処理関数**　…　92
4. 2. 9　A/D**コンバータ割り込み処理関数**　…　92
4. 2. 10　**ハードウエア初期化処理関数**　…　93
4. 2. 11　**コモン・エリア初期化処理関数**　…　94
4. 2. 12　**回転開始初期化処理関数**　…　94
4. 2. 13　sin**計算処理関数**　…　95
4. 2. 14　V850E/IA2**用のリンク・ディレクティブ・ファイル**　…　96
4. 3　**プログラム・リスト（**V850E/IA3**用）**　…　97
4. 3. 1　**シンボル定義**　…　97
4. 3. 2　**定数定義**　…　98
4. 3. 3　**割り込みハンドラ設定**　…　100
4. 3. 4　**スタートアップ・ルーチン設定**　…　101
4. 3. 5　**メイン処理関数**　…　104
4. 3. 6　LED**表示関数**　…　108
4. 3. 7　**モータ制御割り込み処理関数**　…　109
4. 3. 8　10 mSEC**インターバル割り込み処理関数**　…　112
4. 3. 9　A/D**コンバータ割り込み処理関数**　…　112
4. 3. 10　**ハードウエア初期化処理関数**　…　113
4. 3. 11　**コモン・エリア初期化処理関数**　…　114
4. 3. 12　**回転開始初期化処理関数**　…　115
4. 3. 13　sin**計算処理関数**　…　115
4. 3. 14　V850E/IA3**用のリンク・ディレクティブ・ファイル**　…　116
4. 4　**プログラム・リスト（**V850E/IA4**用）**　…　118
4. 4. 1　**シンボル定義**　…　118
4. 4. 2　**定数定義**　…　119
4. 4. 3　**割り込みハンドラ設定**　…　121
4. 4. 4　**スタートアップ・ルーチン設定**　…　122
4. 4. 5　**メイン処理関数**　…　125
4. 4. 6　LED**表示関数**　…　129
4. 4. 7　**モータ制御割り込み処理関数**　…　130
4. 4. 8　10 mSEC**インターバル割り込み処理関数**　…　133
4. 4. 9　A/D**コンバータ割り込み処理関数**　…　133
4. 4. 10　**ハードウエア初期化処理関数**　…　134
4. 4. 11　**コモン・エリア初期化処理関数**　…　135

4. 4. 12　**回転開始初期化処理関数**　…　136
4. 4. 13　sin**計算処理関数**　…　136
4. 4. 14　V850E/IA4**用のリンク・ディレクティブ・ファイル**　…　137

第1章　制御方式

1. 1　ベクトル制御の原理

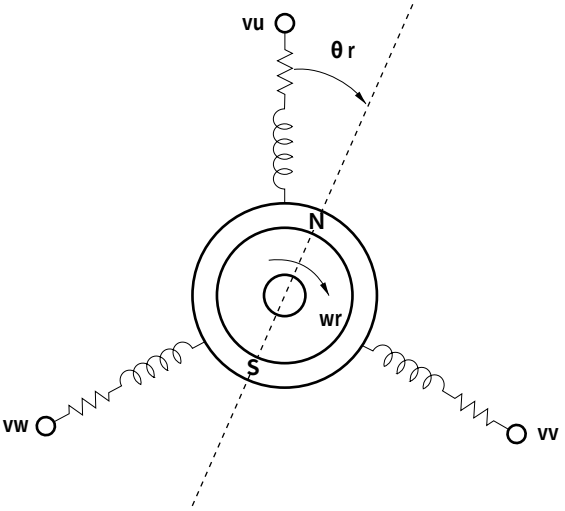
通常3相モータを制御する場合，電圧や電流は3相交流で表すことができます。3相交流で表すより2相交流で表すほうがより簡単となります。さらに，2相交流で表すより2軸直流で表すほうがより制御が簡単となります。

2軸直流（d-q軸）変換を行うと図1－1（c）のように電機子巻線がDCモータのように整流子に接続されて半径方向に無数にあり，それらに界磁と同じ速度で回転するd-q軸上に配置されたブラシを通して，vd（d軸電圧値），vq（q軸電圧値）が印加され，id（d軸電流値），iq（q軸電流値）が流れることがわかります。

レファレンス・システムでは，d-q軸上でDCモータのように電流，電圧等の制御を行っています。

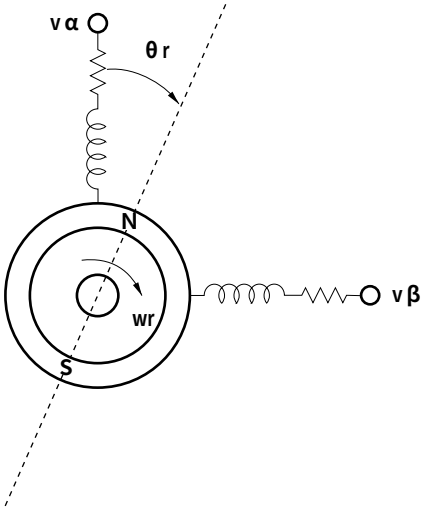
図1－1　等価回路（1/3）

（a）3相同期モータの等価回路



備考　vu：u相電圧値
vv：v相電圧値
vw：w相電圧値
θr：回転角
N：N極
S：S極
wr：角速度

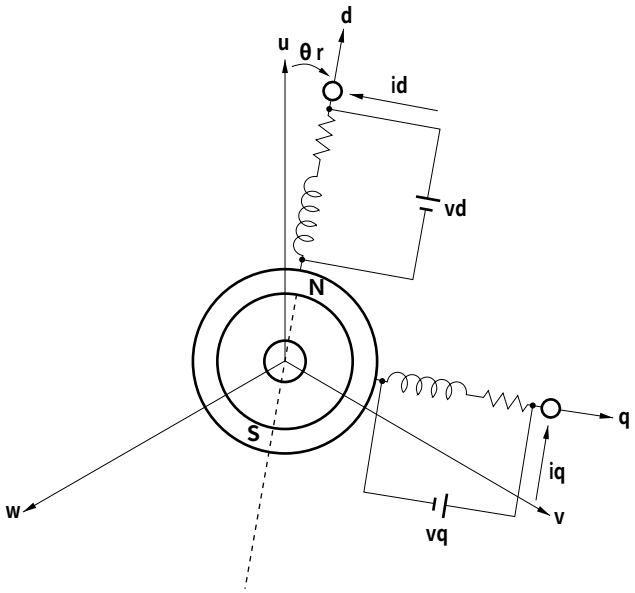
（b）2相交流で表した等価回路



備考　vα：α軸電圧値
vβ：β軸電圧値
θr：回転角
N：N極
S：S極
wr：角速度

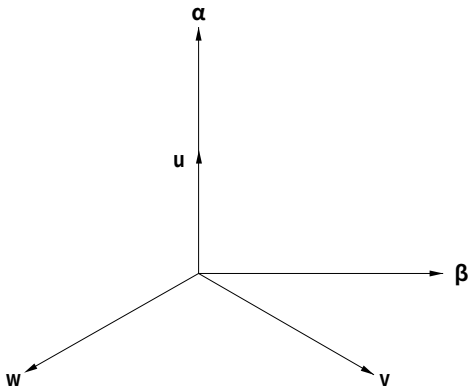
図1－1　等価回路（2/3）

（c）2軸直流（d-q）軸で表した等価回路



備考　u：u相
v：v相
w：w相
θr：回転角
N：N極
S：S極
d：d軸
id：d軸電流値
vd：d軸電圧値
q：q軸
iq：q軸電流値
vq：q軸電圧値

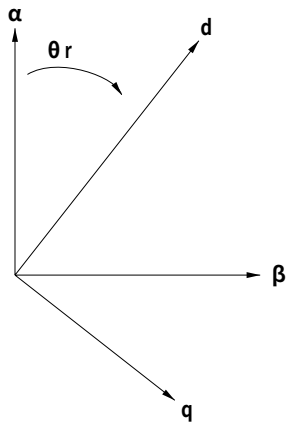
（d）3相および2相交流座標の関係



備考　u：u相
v：v相
w：w相
α：α軸
β：β軸

図1－1　等価回路（3/3）

（e）2相交流と2軸直流座標の関係



備考　α：α軸

β：β軸

d：d軸

q：q軸

θr：回転角

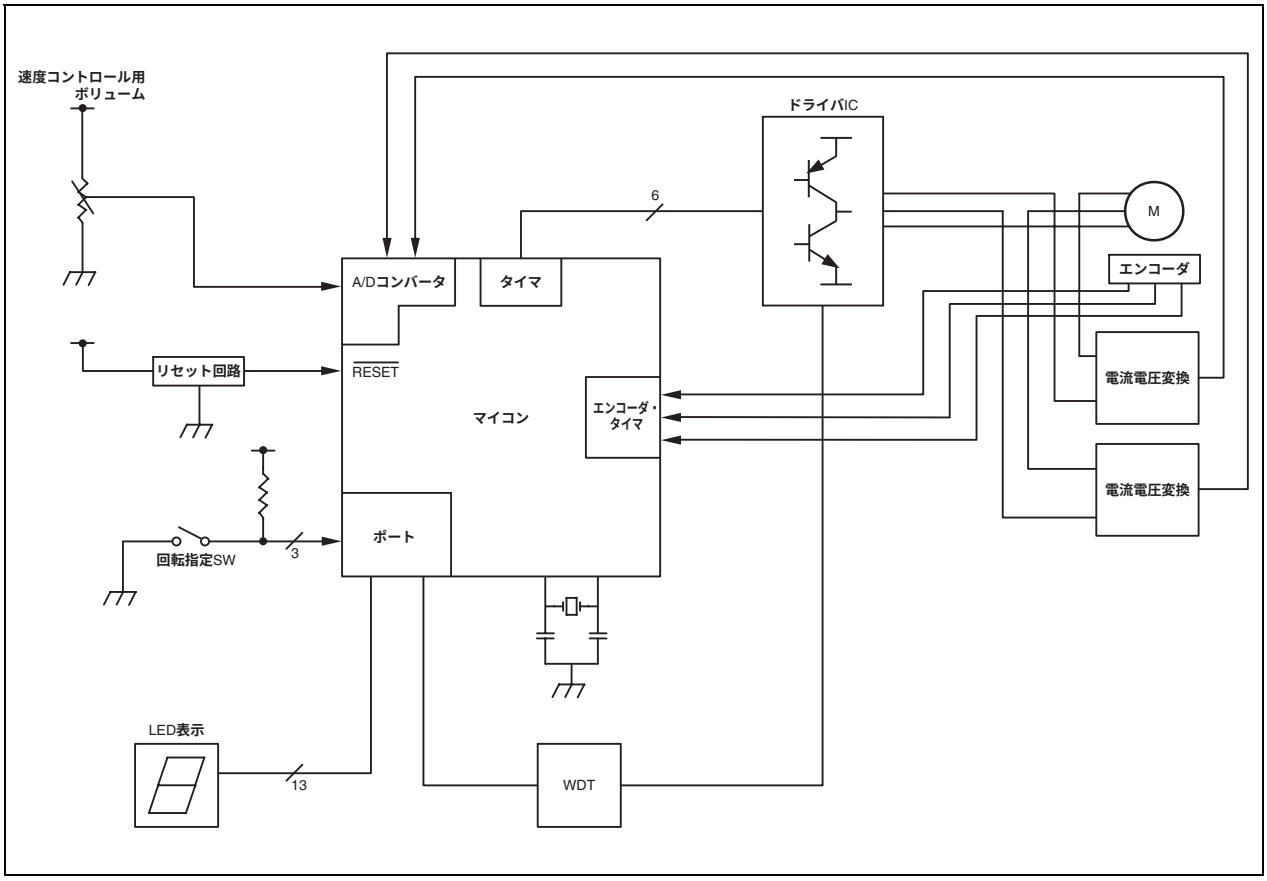
第2章　ハードウエア構成

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) エンコーダ・タイマ

モータからの回転，位置情報

2.2 回 路 図

レファレンス・システムの回路図を図2－2から図2－5に示します。

レファレンス・システムは，V850E/IA1，V850E/IA2，V850E/IA3，V850E/IA4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

V850E/IA1，V850E/IA2，V850E/IA3，V850E/IA4，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

V850E/IA1，V850E/IA2，V850E/IA3，V850E/IA4から1 ms以上の期間パルスがなかった場合，停止信号を出力する仕様となっています。

(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　V850E/IA1の回路図

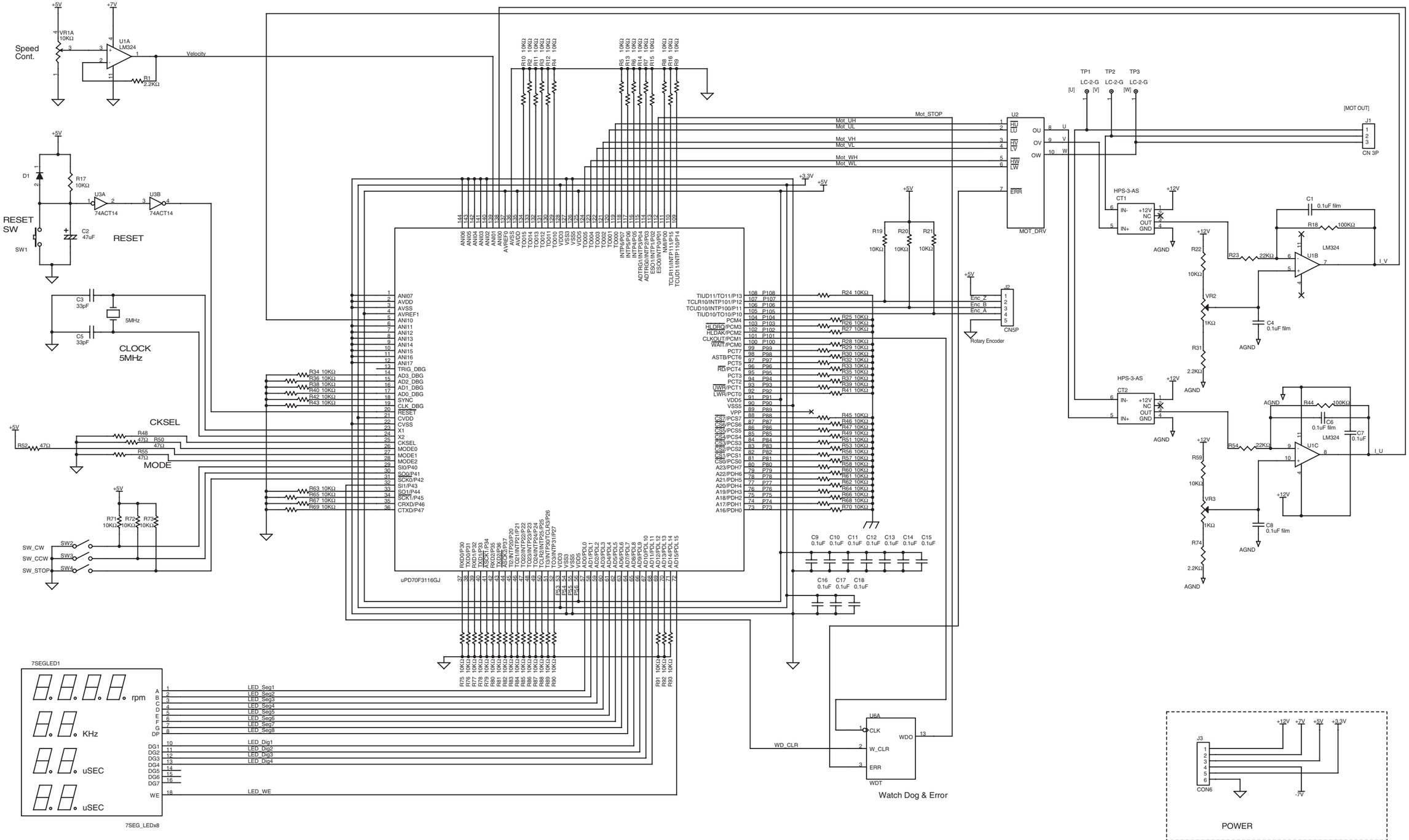


図2－3　V850E/IA2の回路図

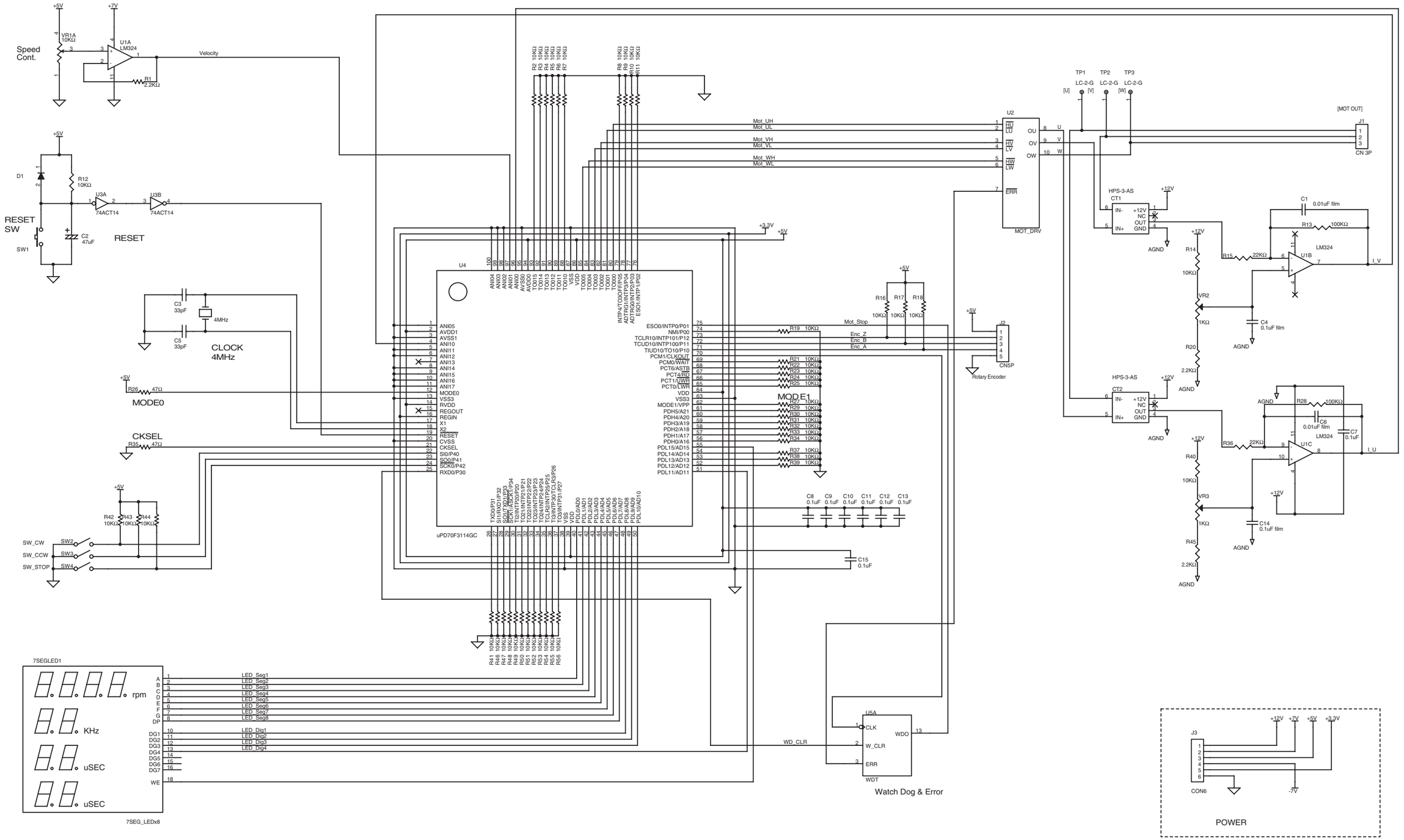


図2－4　V850E/IA3の回路図

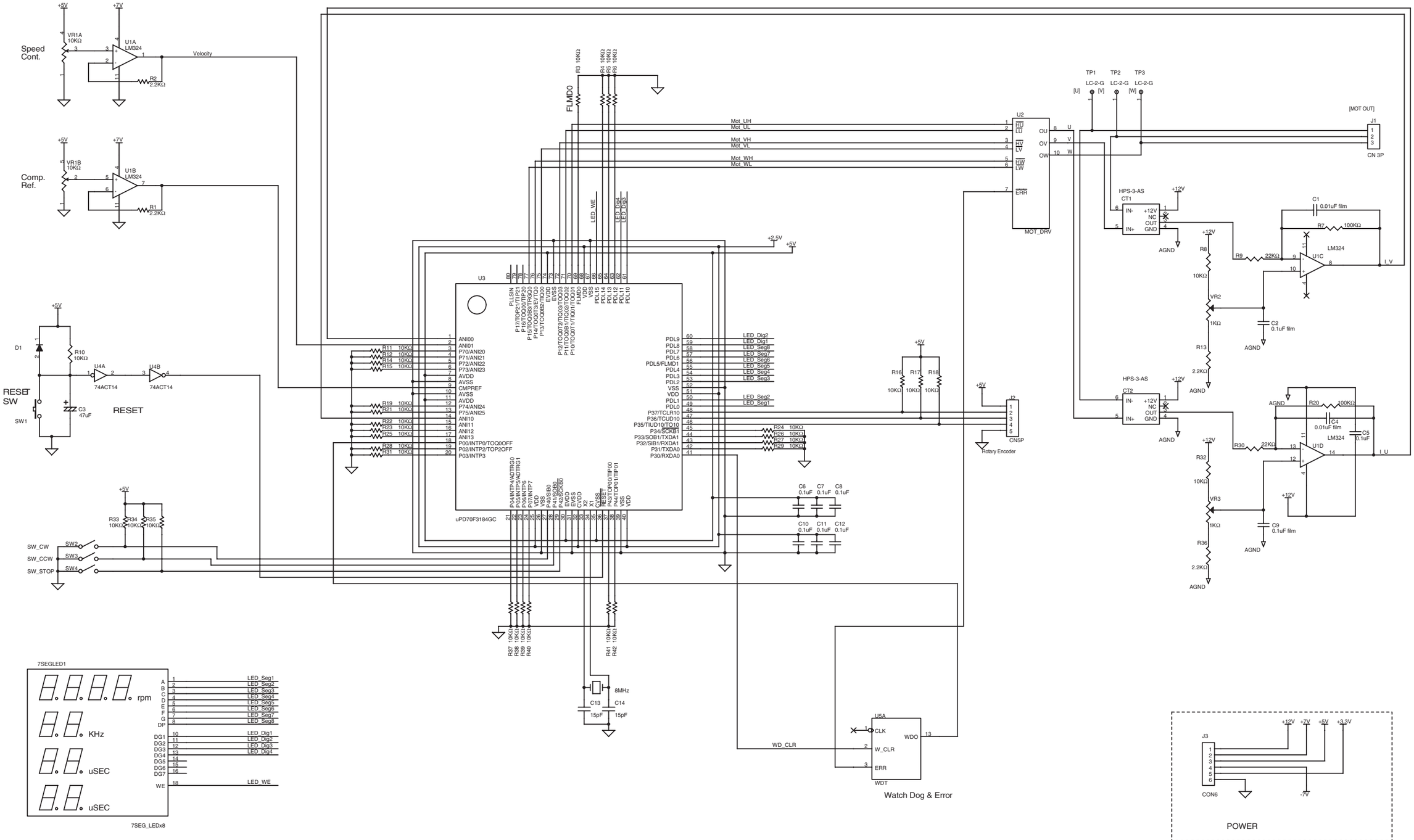
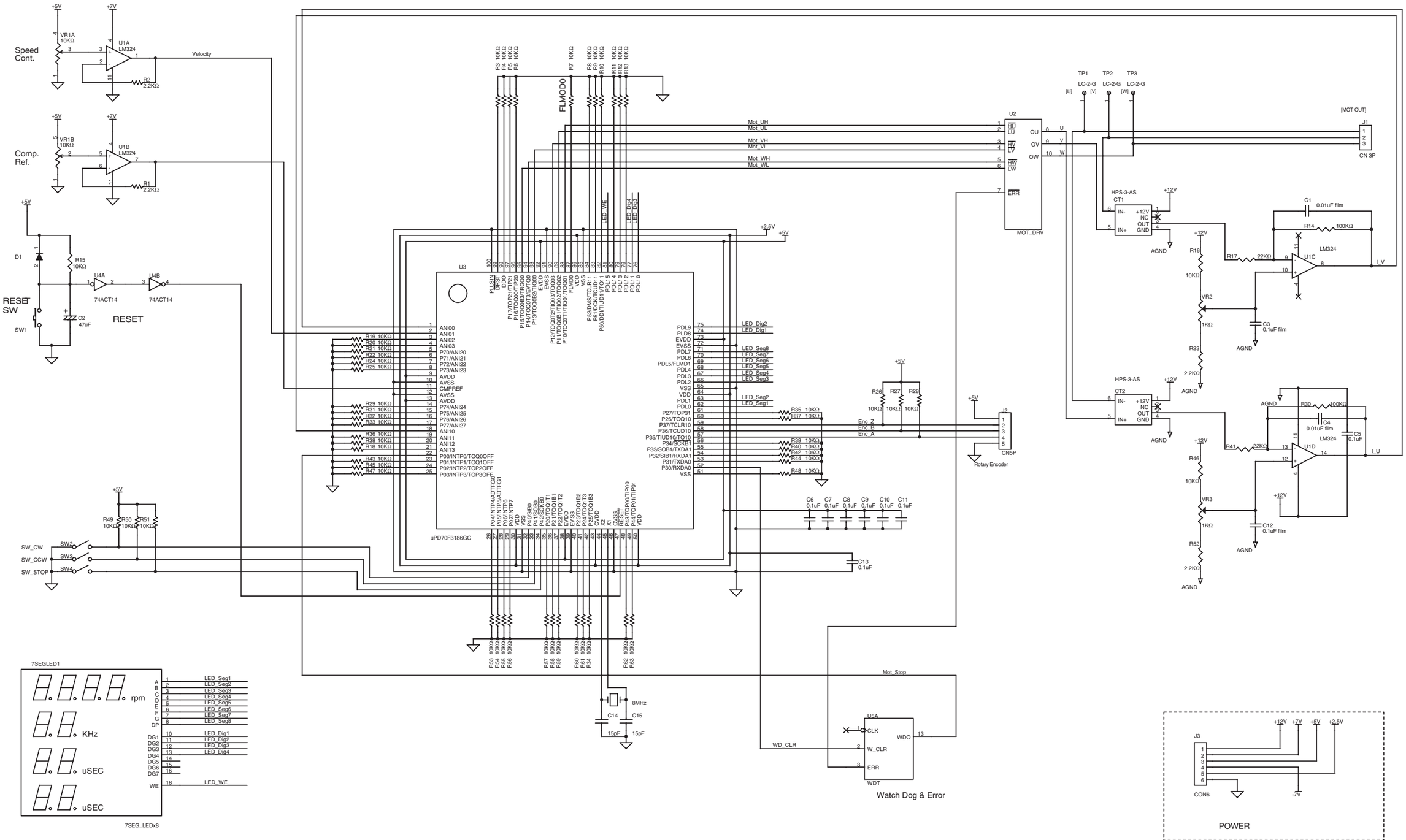


図2－5　V850E/IA4の回路図

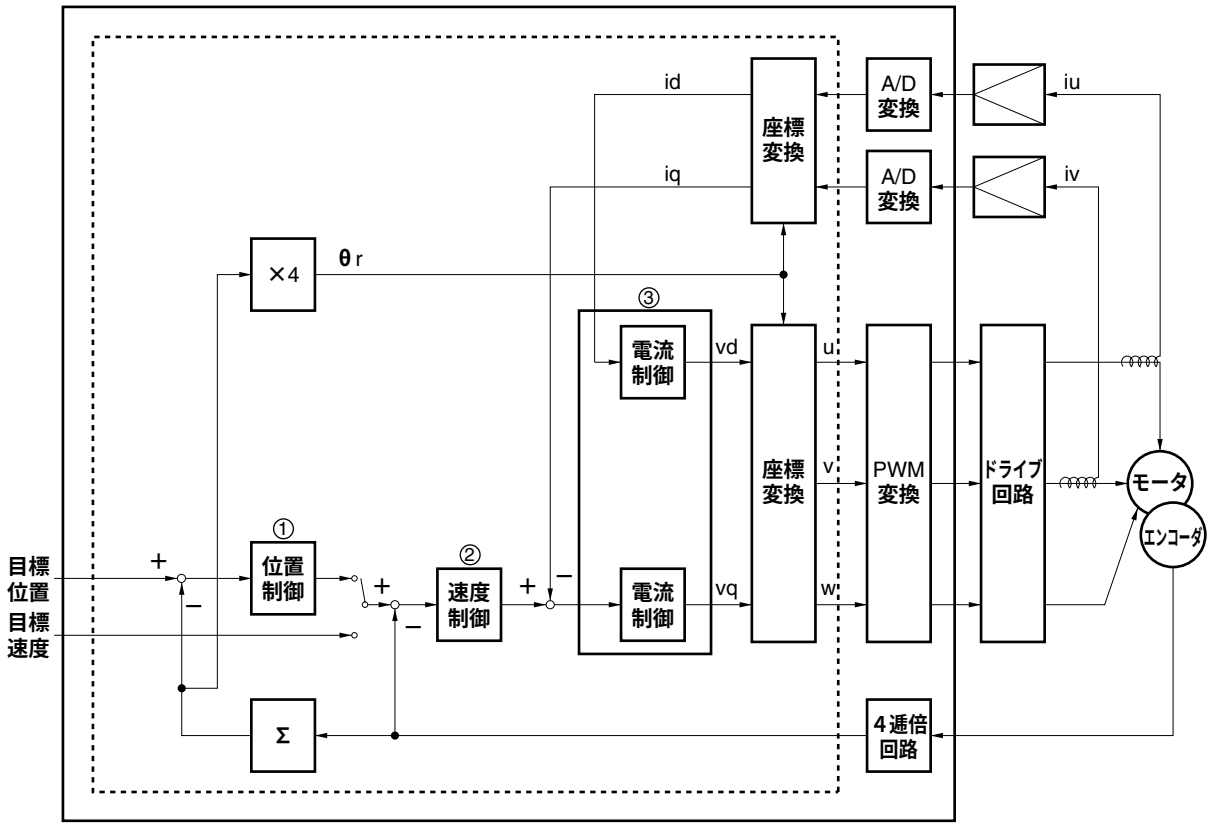


第3章　ソフトウエア構成

3. 1　制御ブロック

レファレンス・システムでは，d-q軸上で図3－1に示すような制御をタイマ割り込み処理で計算を行い，U, V, W相電圧を対象マイコンの持つインバータ・タイマ機能により最終出力を行っています。

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 目標位置を設定した場合，目標位置と現在位置の差分により速度変換を行います。
② 設定した目標速度，または計算された目標速度と現在速度の差分により目標の電流値を計算します。
③ 目標の電流値とd-q軸変換された電流戻り値によりd-q軸目標電圧を計算します。

備考　実線内は対象マイコン内部での処理を表します。
破線内はソフトウエアでの処理を表します。

3. 2　速度制御

レファレンス・システムでは，速度制御部にPI**制御（**Proportion**：比例，**Integral**：積分）を使用しています。使用している式を次に示します。**

d_speed = o_speed－now_speed

o_iqp　= ksp×d_speed

o_iqi　= o_iqi（n－1）＋（ksi×d_speed（n－1））

o_iq　= o_iqp＋o_iqi（n）

備考

d_speed	**：目標速度と現在速度の差**
o_speed	**：目標速度**
now_speed	**：現在速度**
o_iqp	**：速度比例成分電流値**
ksp	**：速度比例定数**
o_iqi	**：速度積分成分電流値**
ksi	**：速度積分定数**
o_iq	**：目標電流値**
n	**：今回成分**
n－1	**：前回成分**

3. 3　エンコーダによる位置速度検出

エンコーダは，2**相エンコーダ入力用タイマを使用して**4**逓倍にしています。モータ軸上の絶対位置を検出する必要があるため**Z**相でエンコーダ値をクリアし，絶対位置を検出しています。**

3. 4　電流制御

電流制御は，d軸電流（id）およびq軸電流（iq）を次に示す式で各軸の目標電圧に変換しています。

o_vd = ki×（－id）
o_vq = ki×（o_iq－iq）

備考　o_vd ：目標d軸電圧
ki　　：電流比例ゲイン
id　　：d軸電流値
o_vq ：目標q軸電圧
o_iq　：目標電流値
iq　　：q軸電流値

id, iqは，u, v相の電流値をd-q軸座標変換した値です。変換式を次に示します。

id = iv×sinθr－iu×sin（θr－2π/3）
iq = iv×cosθr－iu×cos（θr－2π/3）

備考　id　　：d軸電流値
iq　　：q軸電流値
iu　　：u相電流値
iv　　：v相電流値
θr　：回転角

3. 5　3相電圧変換

d-q軸で計算された電圧（vd, vq）を3相座標へ変換する式を次に示します。

o_vu = o_vd×cos θr－o_vq×sin θr
o_vv = o_vd×cos（θr－2π/3）－o_vq×sin（θr－2π/3）
o_vw = －o_vu－o_vv

備考　o_vu ：目標u相電圧
o_vv ：目標v相電圧
o_vw：目標w相電圧
o_vd ：目標d軸電圧
o_vq ：目標q軸電圧
θr　：回転角

3. 6　PWM変換

計算された目標電圧は，インバータ・タイマにより出力されます。

3. 7　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（V850E/IA1）	周辺I/O機能名（V850E/IA2）	周辺I/O機能名（V850E/IA3）	周辺I/O機能名（V850E/IA4）
インバータ・タイマ	タイマ00（TM00）	タイマ00（TM00）	タイマQ0（TMQ0）	タイマQ0（TMQ0）
10 msタイマ	タイマ21（TM21）	タイマ21（TM21）	タイマP0（TMP0）	タイマP0（TMP0）
モータ制御タイマ	タイマ3（TM3）	タイマ3（TM3）	タイマP1（TMP1）	タイマP1（TMP1）
時間計測用タイマ	タイマ3（TM3）	タイマ3（TM3）	タイマP1（TMP1）	タイマP1（TMP1）
U相電流	ANI00	ANI00	ANI00	ANI00
V相電流	ANI10	ANI10	ANI10	ANI10
設定スピード（ボリューム）	ANI01	ANI01	ANI01	ANI01
A相エンコーダ	TIUD10	TIUD10	TIUD10	TIUD10
B相エンコーダ	TCUD10	TCUD10	TCUD10	TCUD10
Z相エンコーダ	TCLR10	TCLR10	TCLR10	TCLR10
CWキー入力	P40	P40	P40	P40
CCWキー入力	P41	P41	P41	P41
STOPキー入力	P42	P42	P42	P42
WDTリセット出力	P43	P30	P30	P30
LED出力	PDL0-PDL11, PDL15	PDL0-PDL11, PDL15	PDL0-PDL11, PDL15	PDL0-PDL11, PDL15

(1) 周辺I/O機能の説明

(a) インバータ・タイマ

各インバータ・タイマを使用してPWM波形を出力します。

レファレンス・システムでは次のような設定になっています。

- 20 kHz対称三角波モード
- デッド・タイム：6 μs
- インバータ・タイマ出力：ロウ・アクティブ
- ESO0, TOQ0OFF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

各モータ制御用タイマを使用し，50 μsインターバル割り込みを発生させます。

(c) 10 msタイマ

各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) 速度計測用タイマ

モータの回転スピード計測用に使用します。

(e) 電流値入力

ANI00：U相電流値（−5〜+5 A）

ANI10：V相電流値（−5〜+5 A）

(f) 速度指令ボリューム値入力

ANI01を使用して0-1023までの値を入力します。

3. 8　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

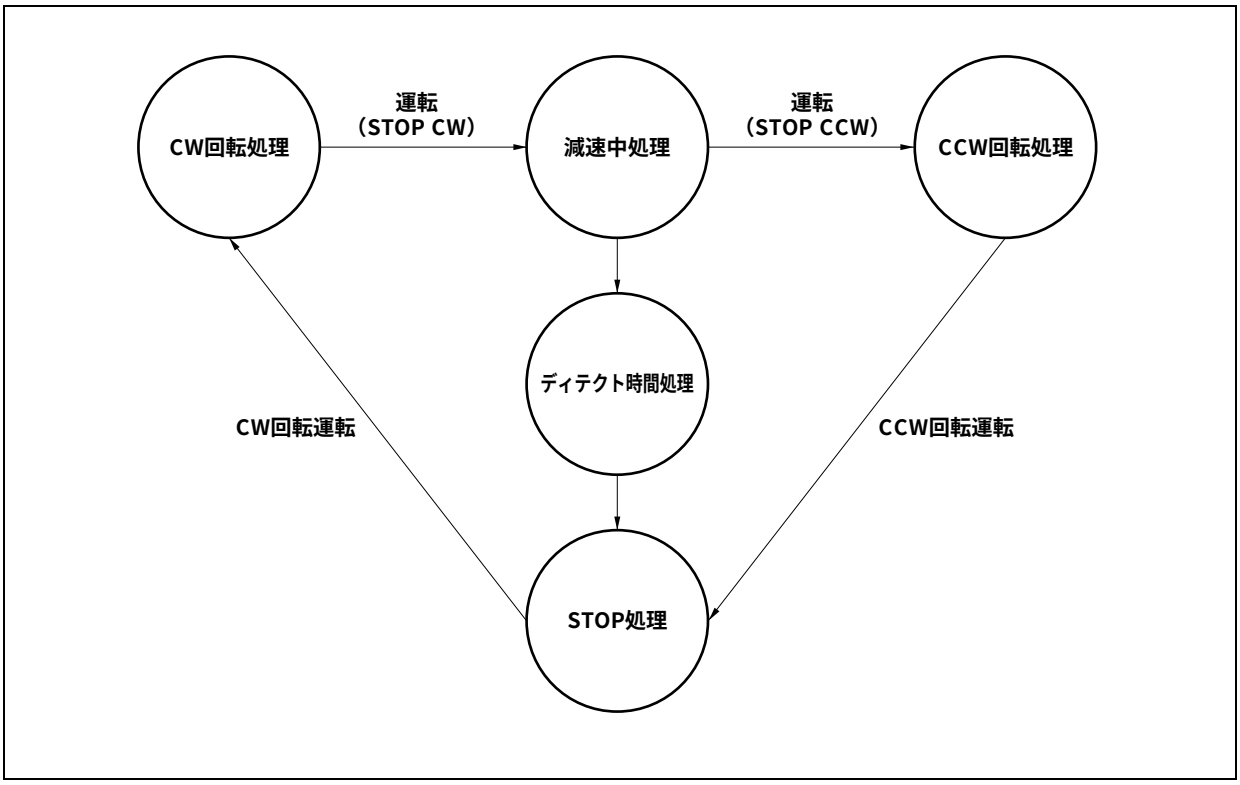
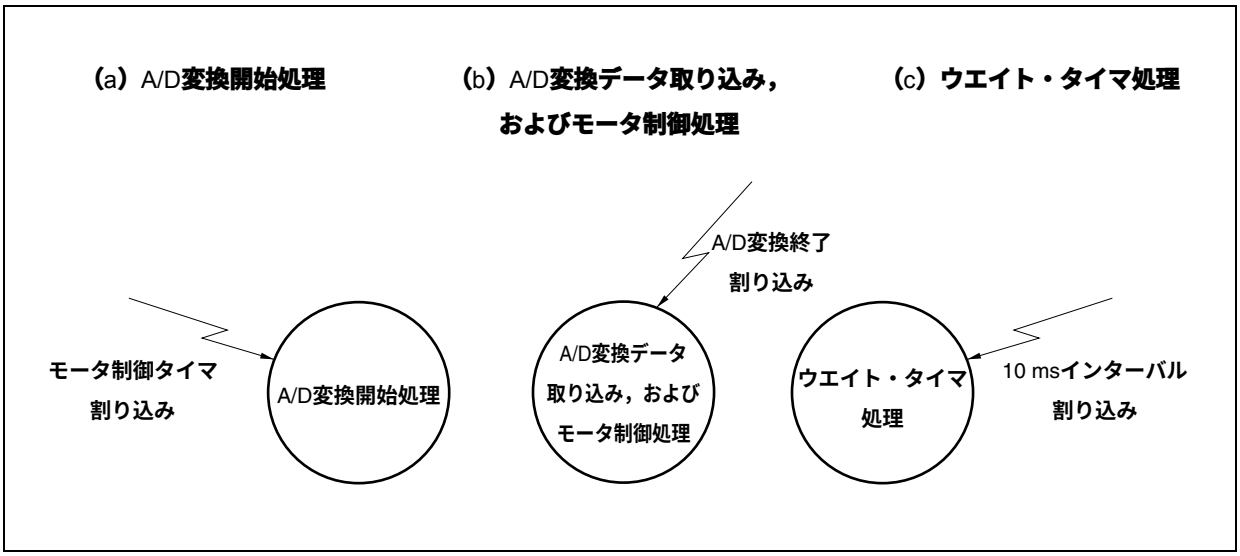


図3－3　割り込み処理構造



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。指示された状態をモータ制御タイマ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

  エンコーダからの位置情報を入力するまで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. 9　フロー・チャート

3. 9. 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

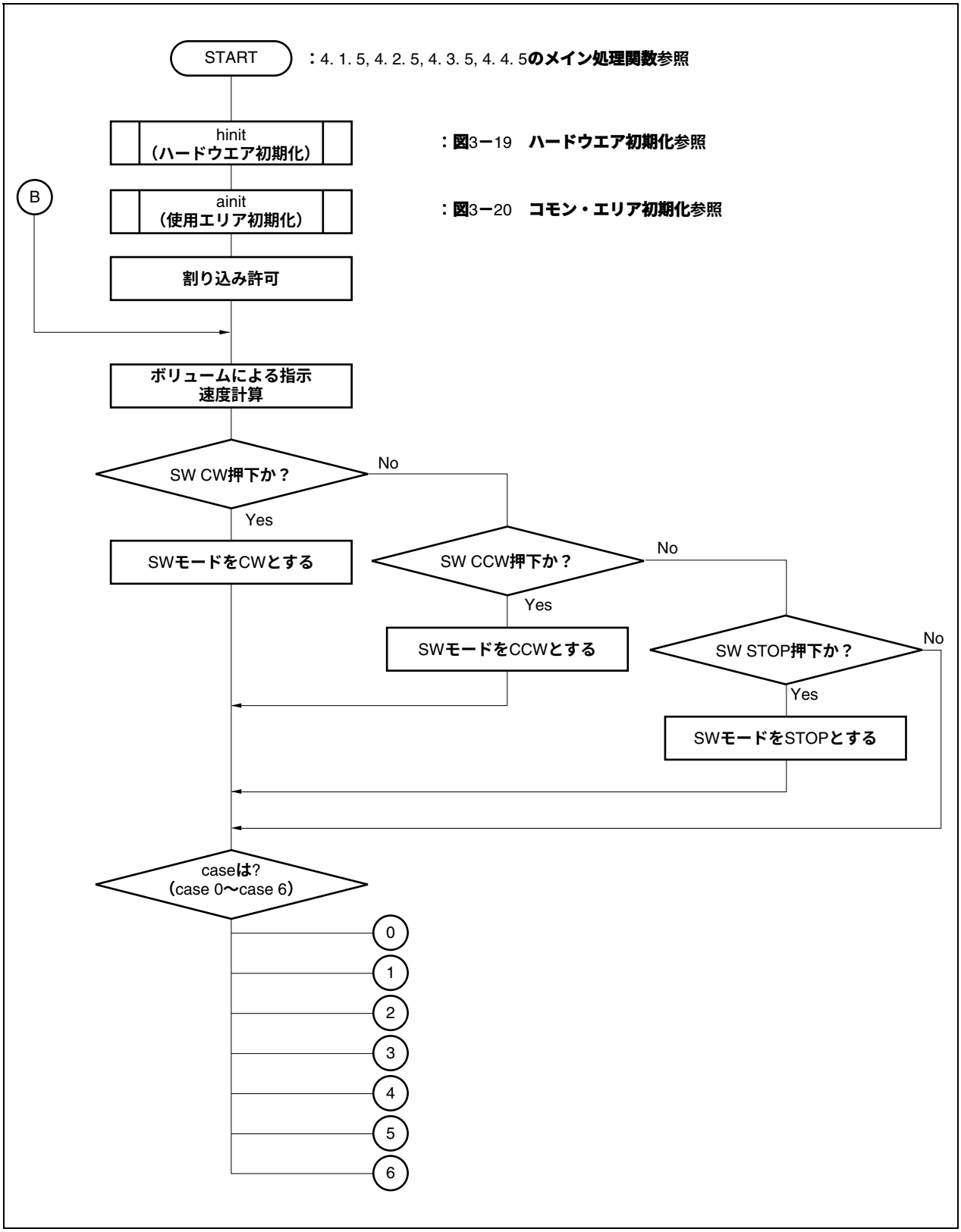

START
：4. 1. 5, 4. 2. 5, 4. 3. 5, 4. 4. 5のメイン処理関数参照
hinit
（ハードウエア初期化）
：図3－19　ハードウエア初期化参照
B
ainit
（使用エリア初期化）
：図3－20　コモン・エリア初期化参照
割り込み許可
ボリュームによる指示
速度計算
SW CW押下か？
No
Yes
SWモードをCWとする
SW CCW押下か？
No
Yes
SWモードをCCWとする
SW STOP押下か？
No
Yes
SWモードをSTOPとする
caseは?
（case 0～case 6）
0
1
2
3
4
5
6

図3ー5　case 0（停止中処理）

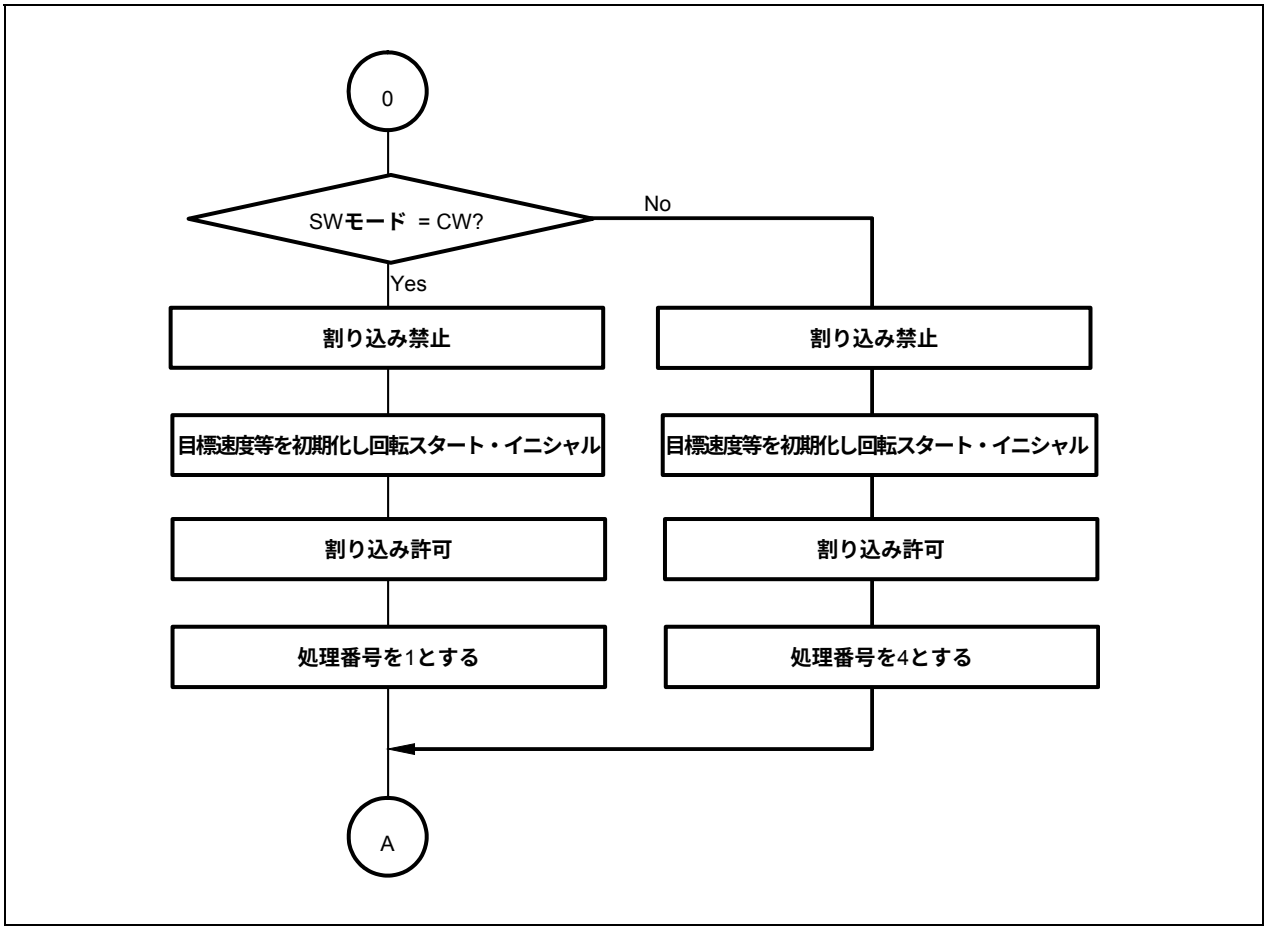


図3－6　case 1（CW加速処理）

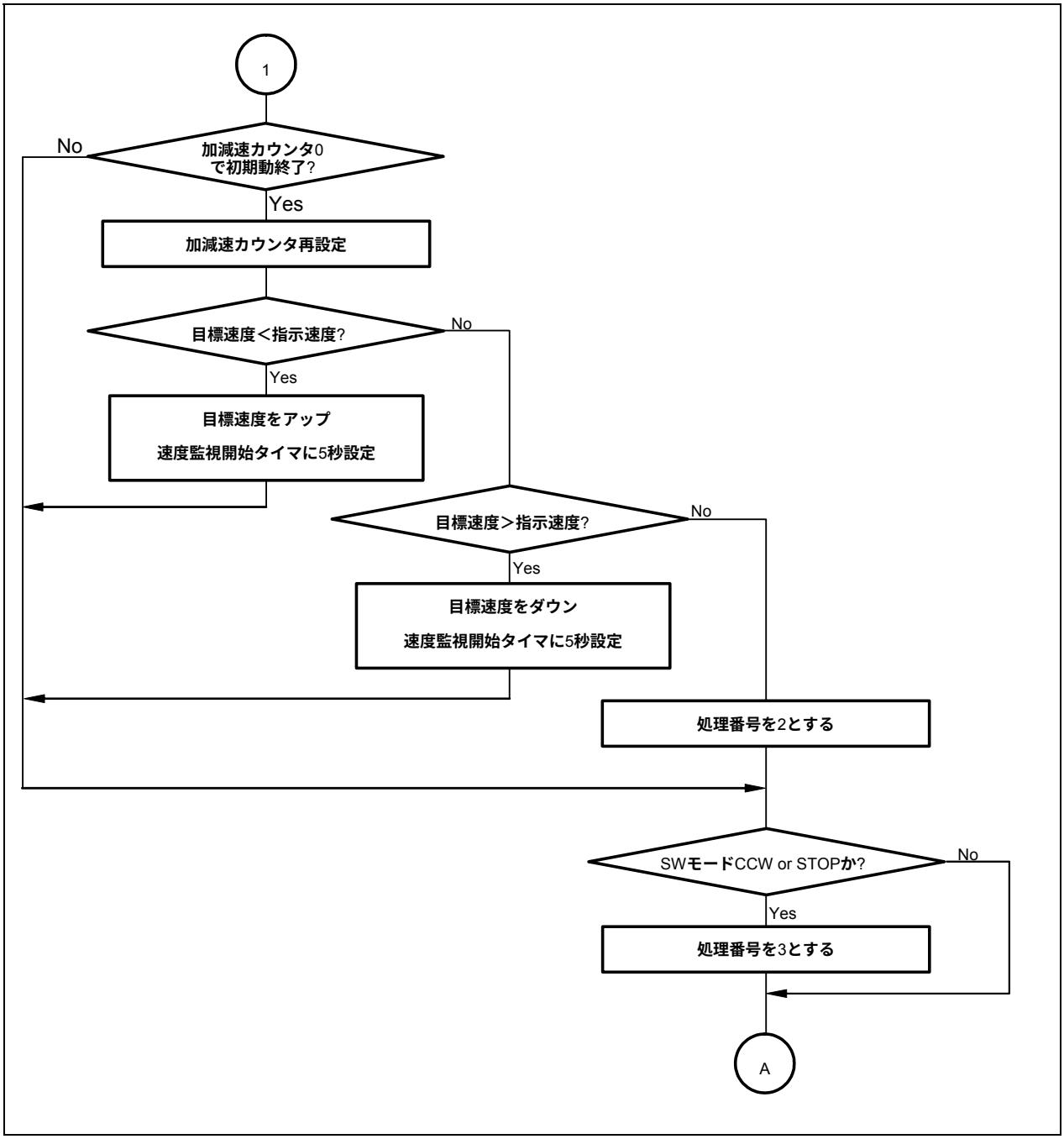


図3－7　case 2（CW定速処理）

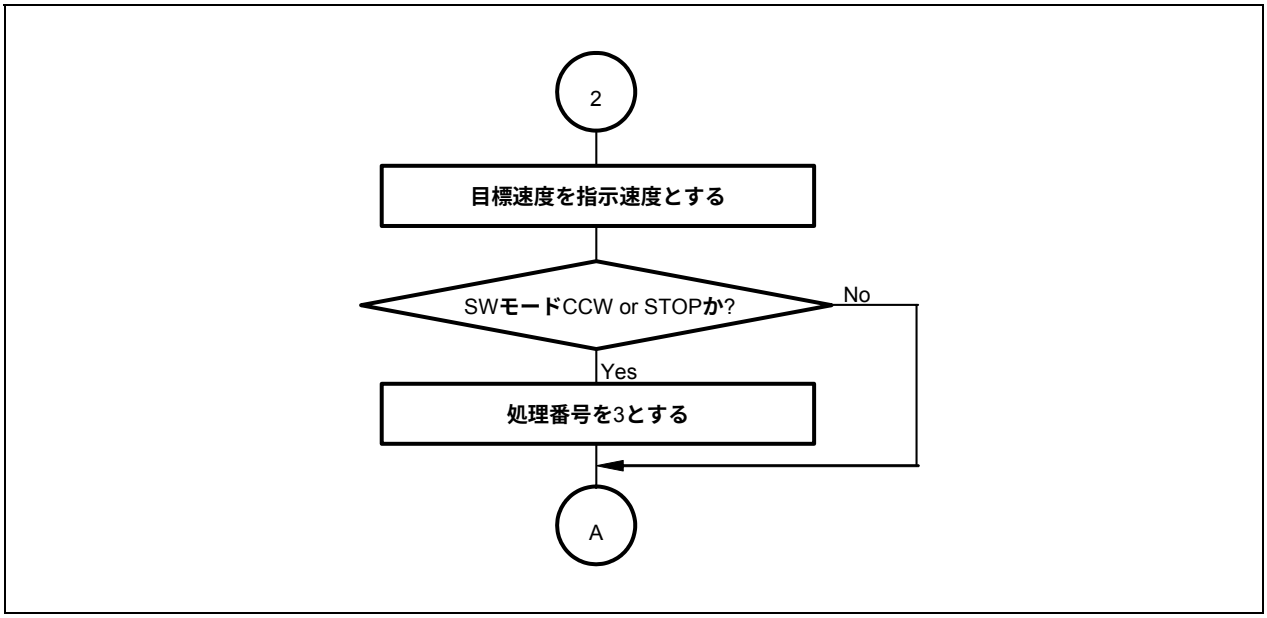


図3－8　case 3（CW停止処理）

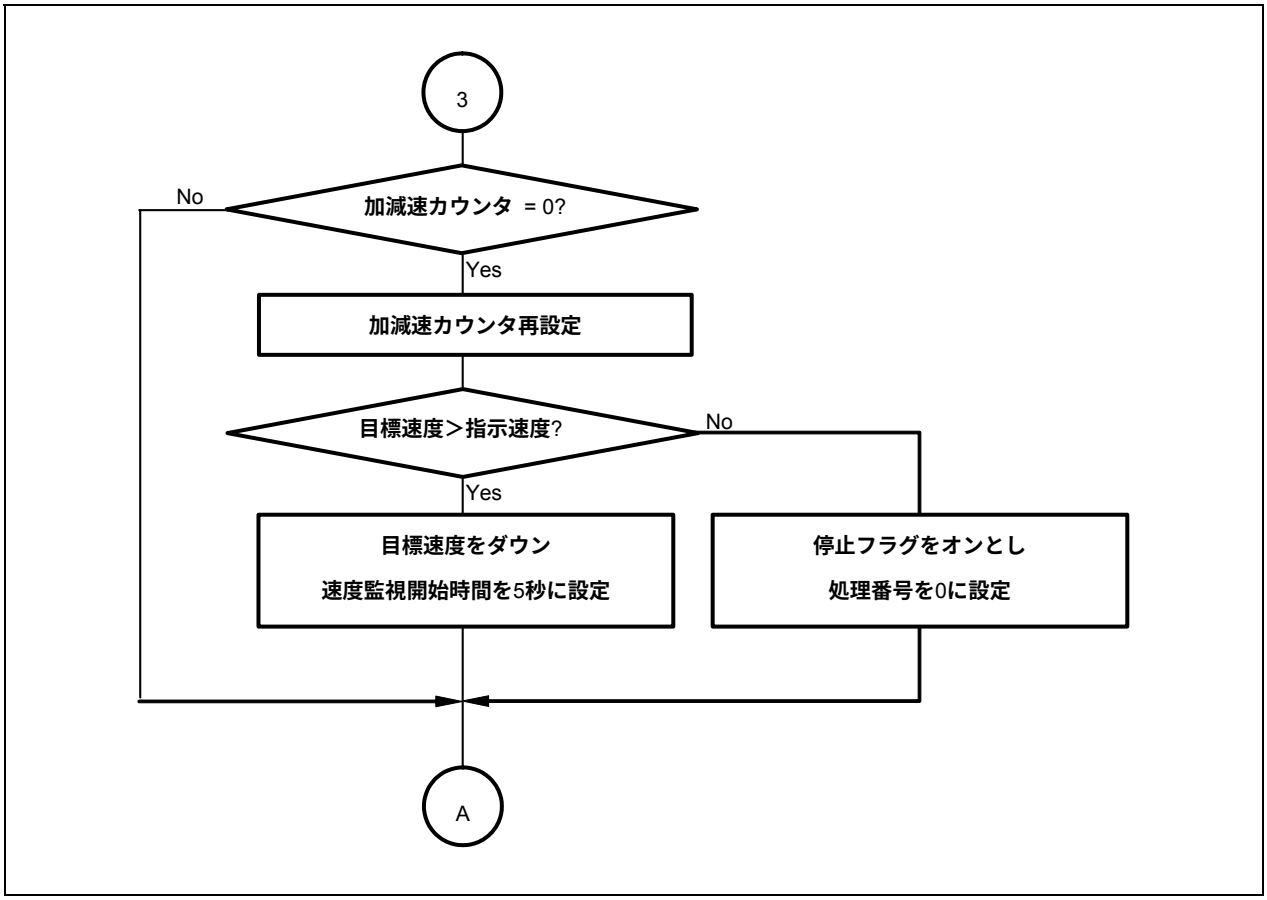


図3－9　case 4（CCW加速処理）

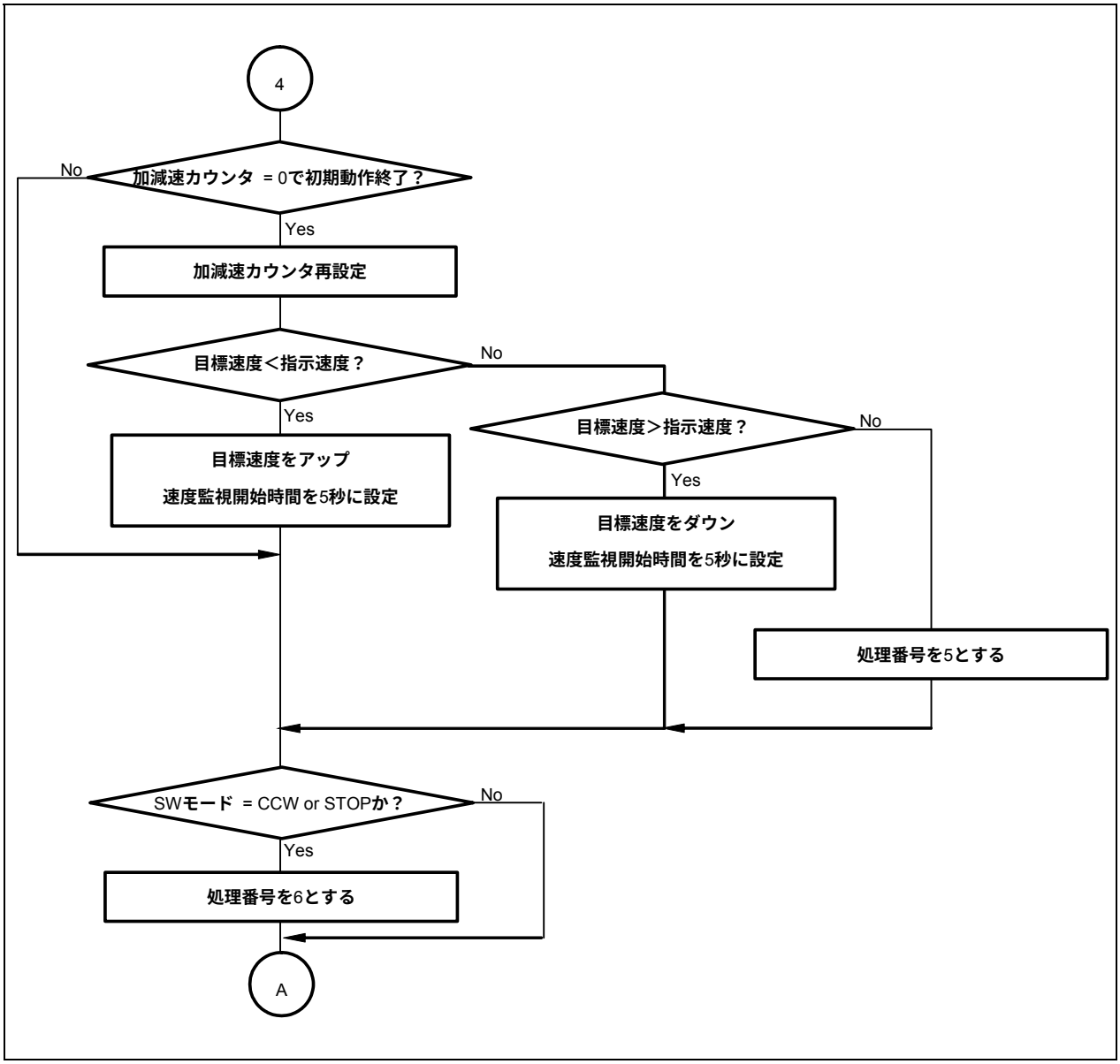


図3－10　case 5（CCW定速処理）

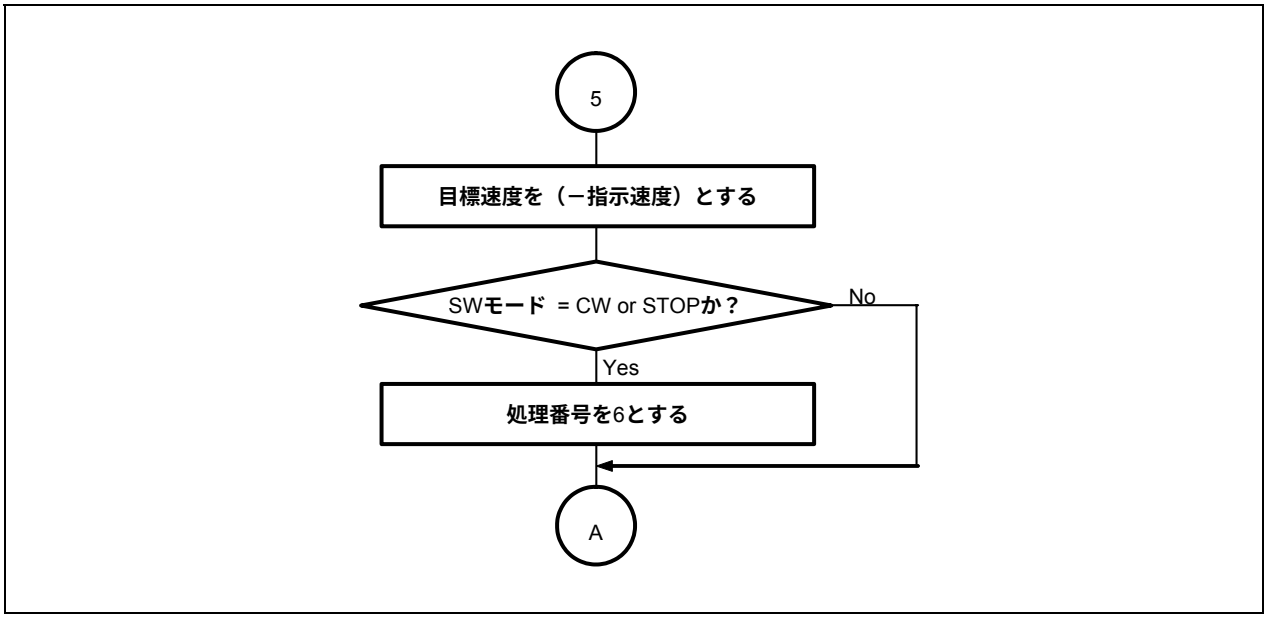


図3－11　case 6（CCW停止処理）

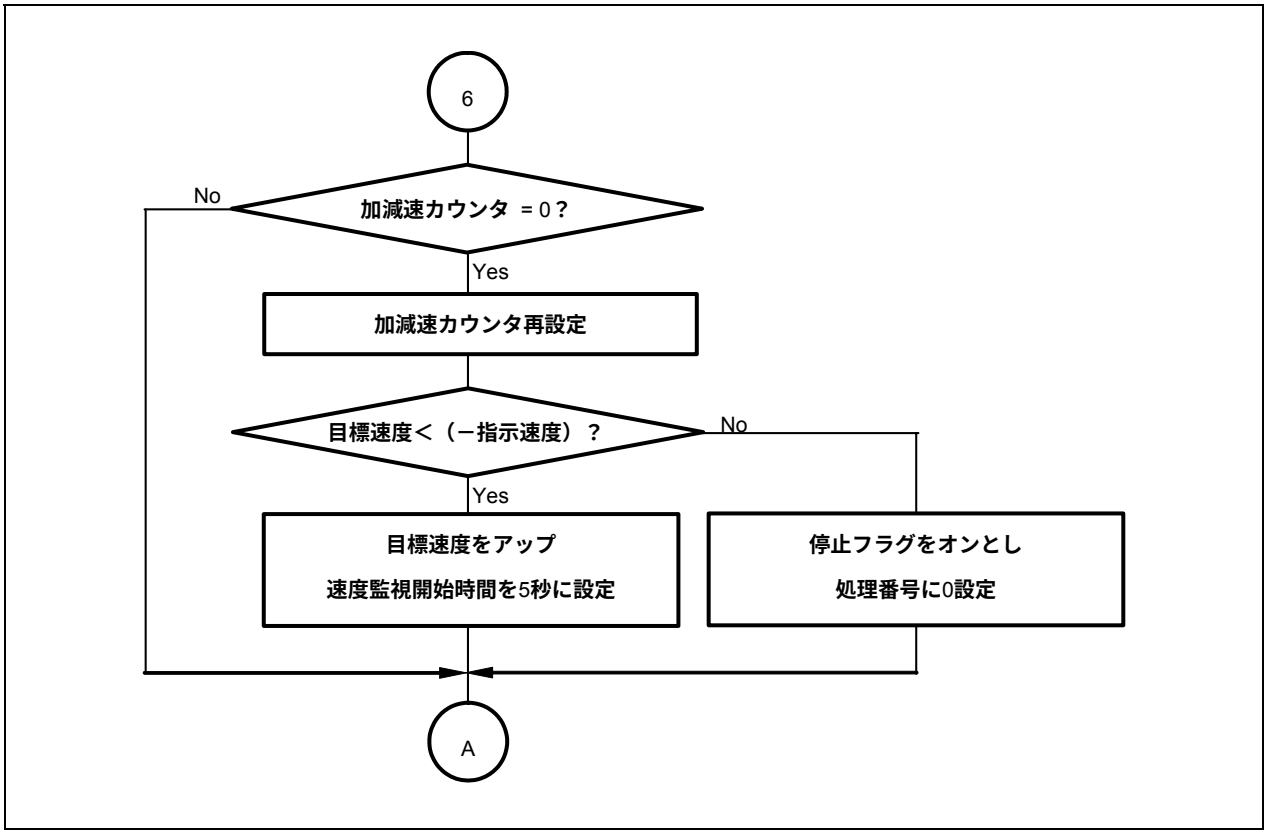
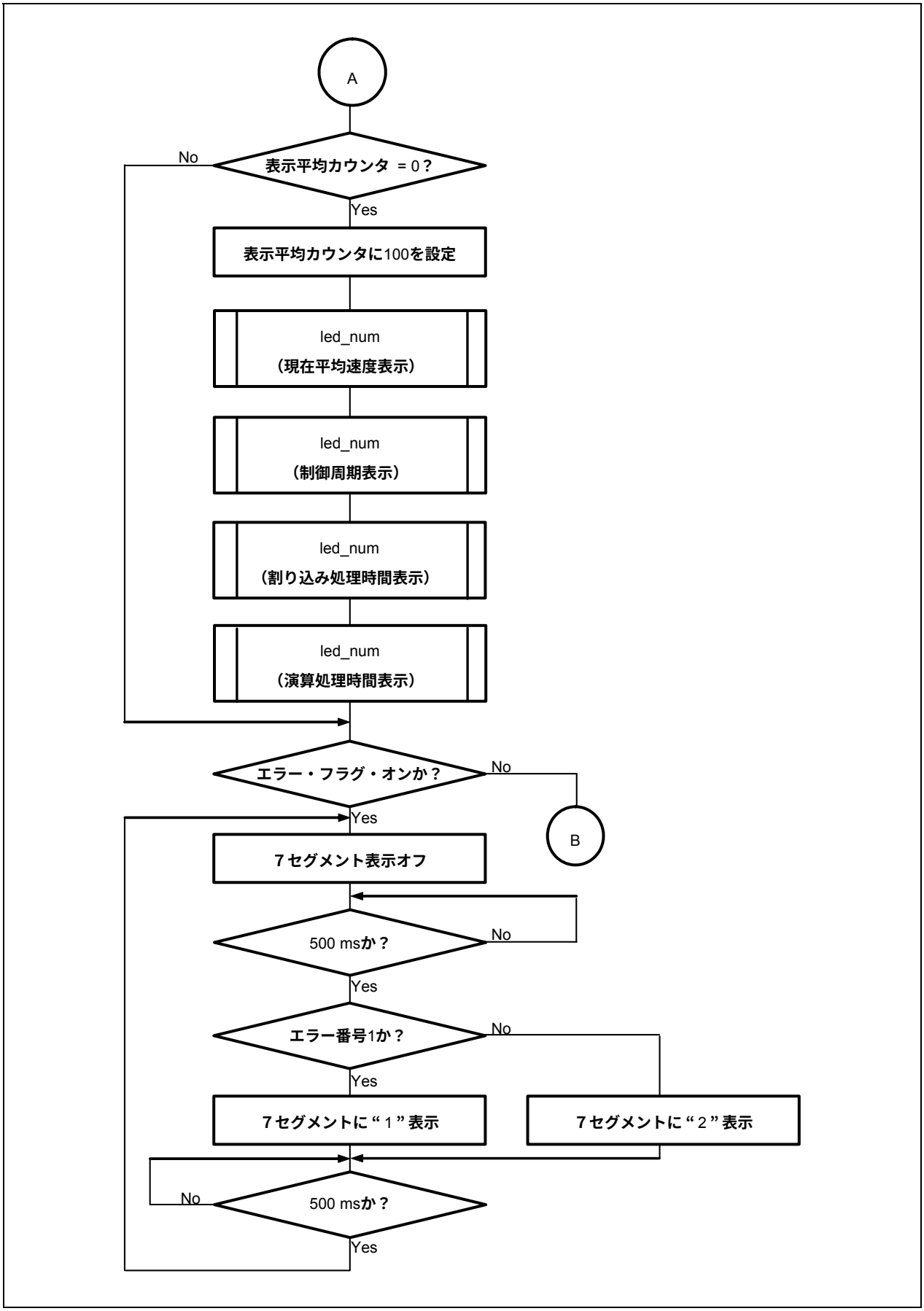
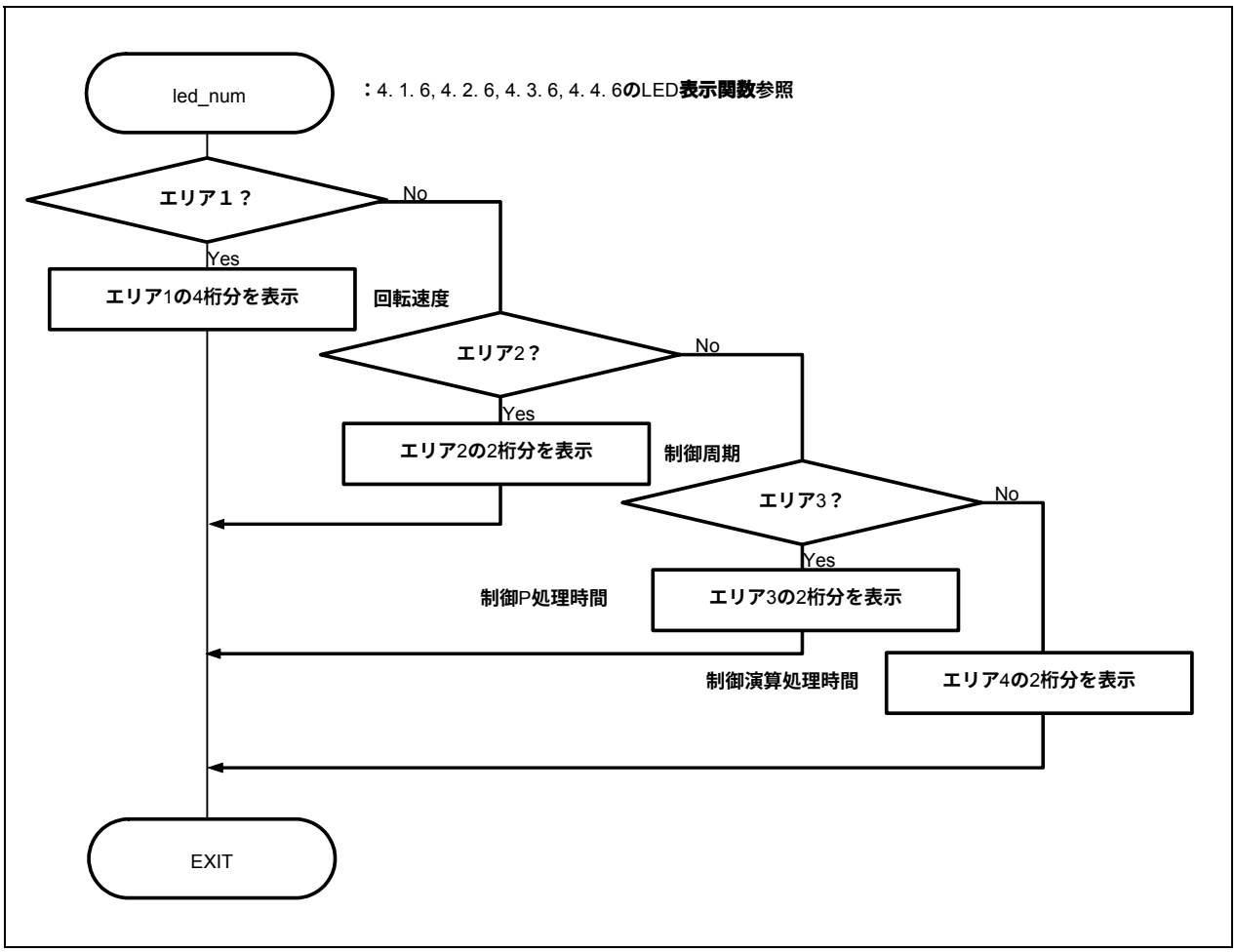


図3ー12　LED表示処理



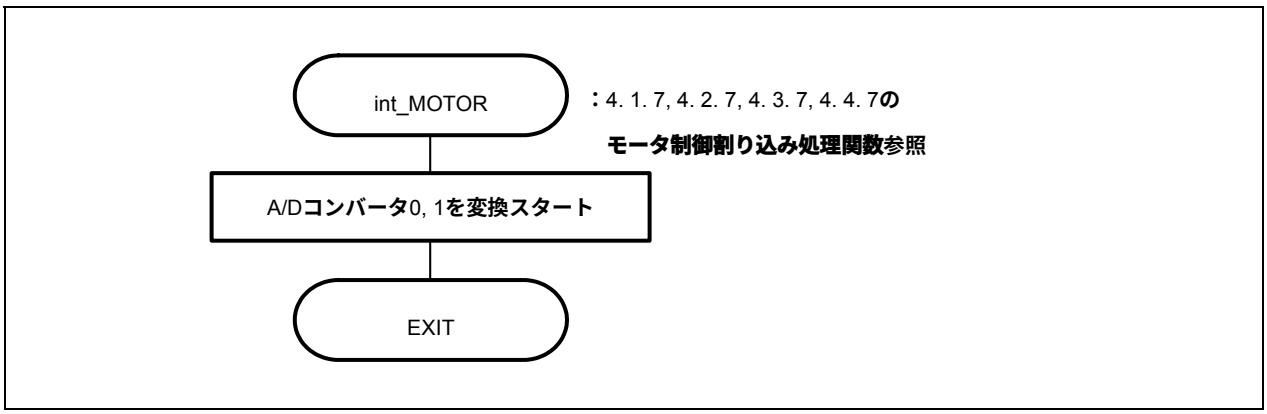
3. 9. 2　LED表示

図3ー13　LED表示



3. 9. 3　モータ制御タイマ割り込み処理

図3－14　モータ制御タイマ割り込み処理



3. 9. 4　モータ制御処理

図3－15　制御割り込み処理（1/5）

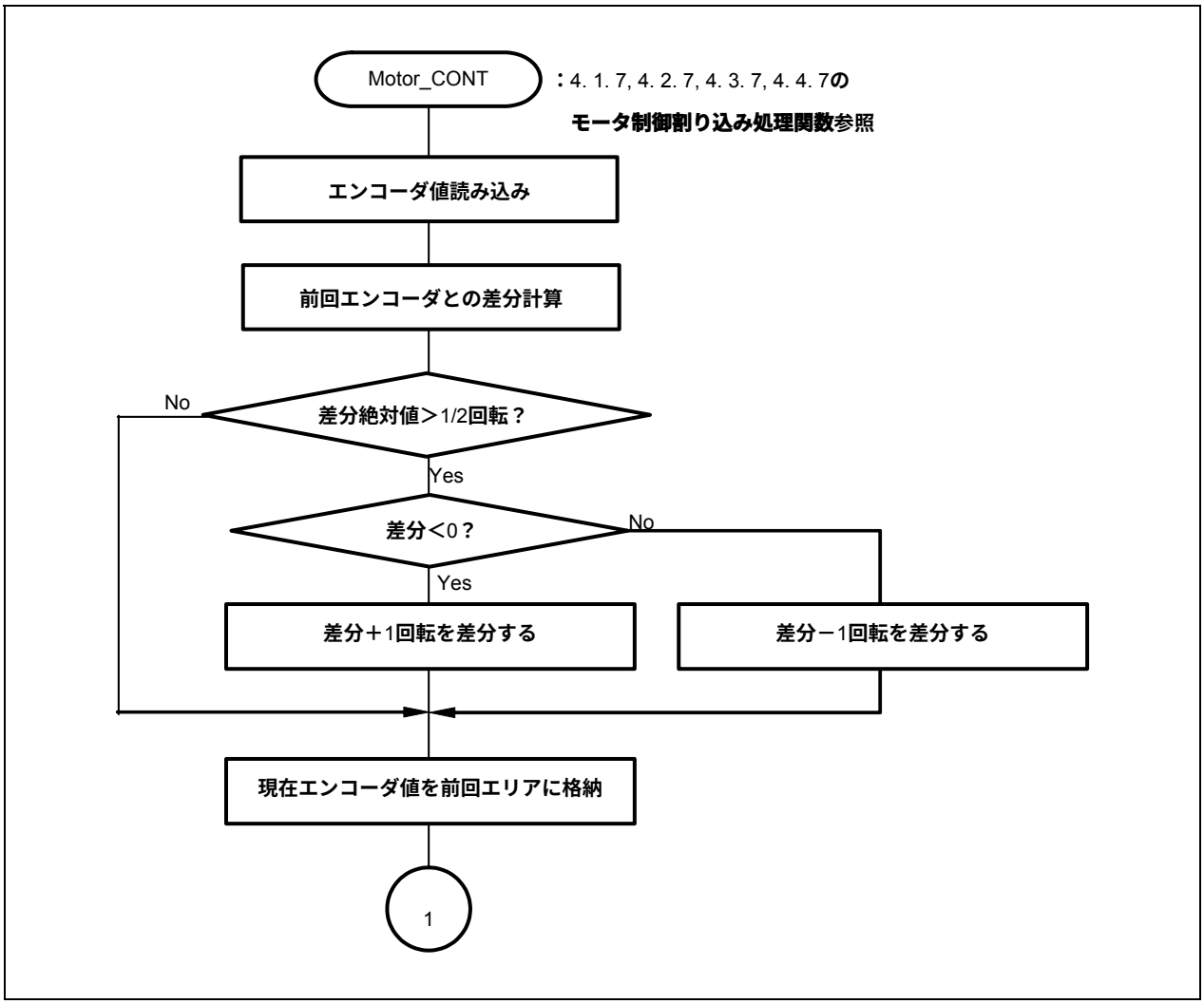
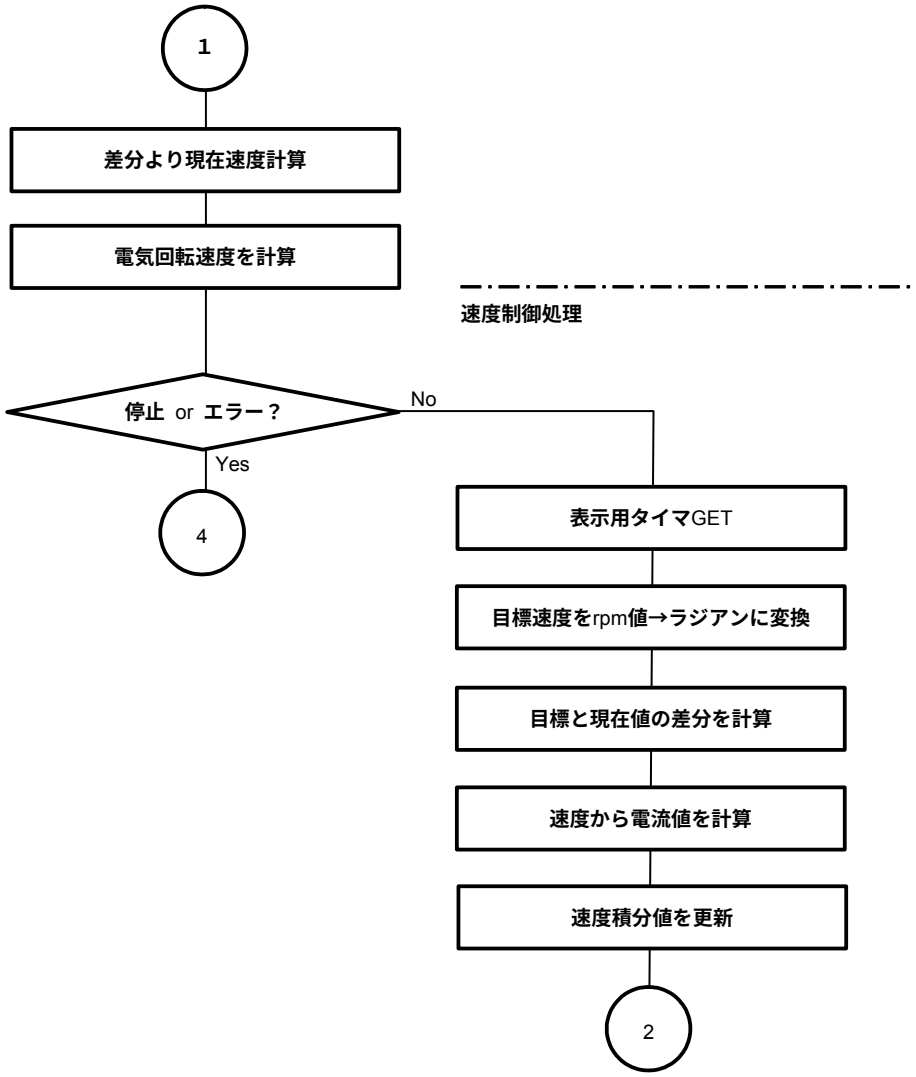


図3－15　制御割り込み処理（2/5）



備考1.　**電気回転速度 = 速度×極数**

2.　**電流値 =（KSP×速度差＋速度積分値）**/KSIGETA

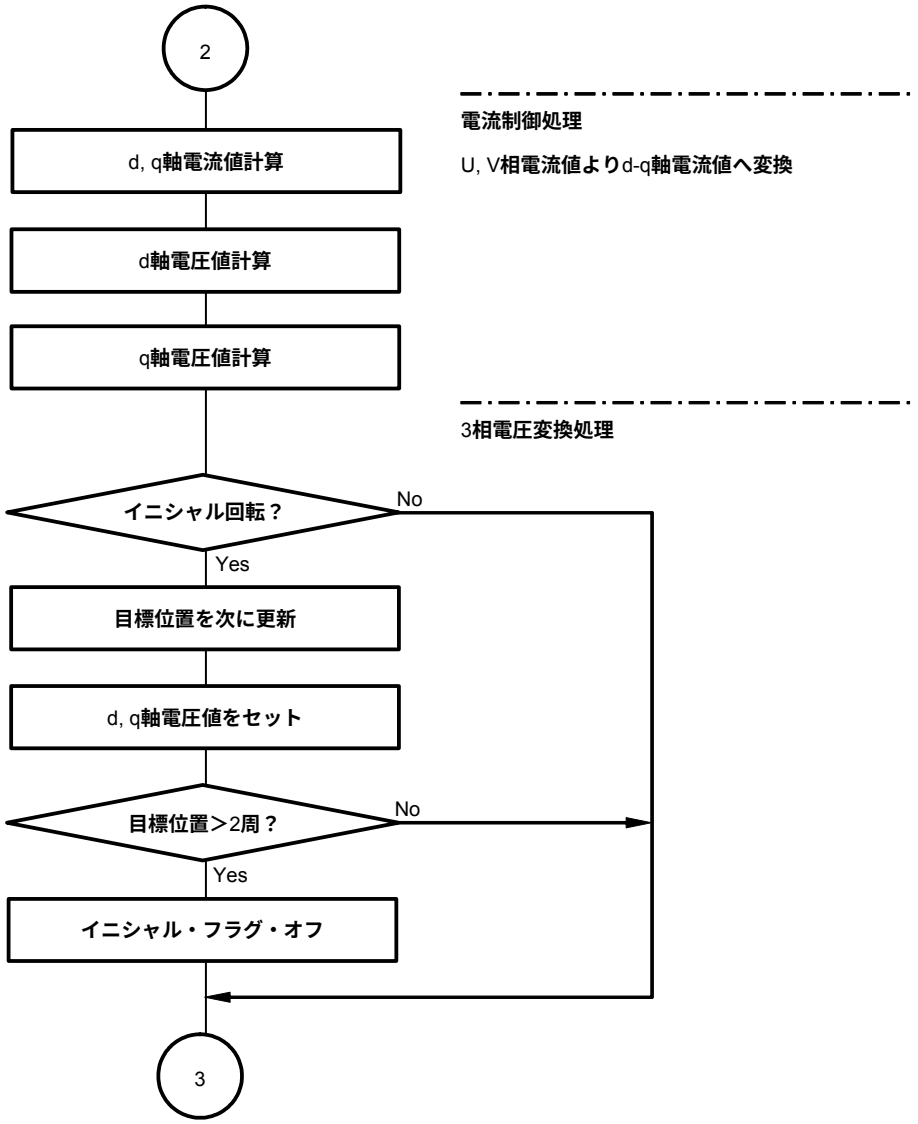
3.　**速度積分値 = 前回分の速度積分値**＋KSI×**速度差**

4.　KSI　　**：速度積分定数**

KSIGETA　**：速度積分定数ゲタ定数**

KSP　　**：速度比例定数**

図3－15　制御割り込み処理（3/5）



備考1.　d軸電圧値 = KI×（－d軸電流値）/KIGETA

2.　q軸電圧値 = KI×（目標q軸電流値－q軸電流値）/KIGETA

3.　KI　　：電流変換定数

KIGETA ：電流変換定数ゲタ定数

図3－15　制御割り込み処理（4/5）

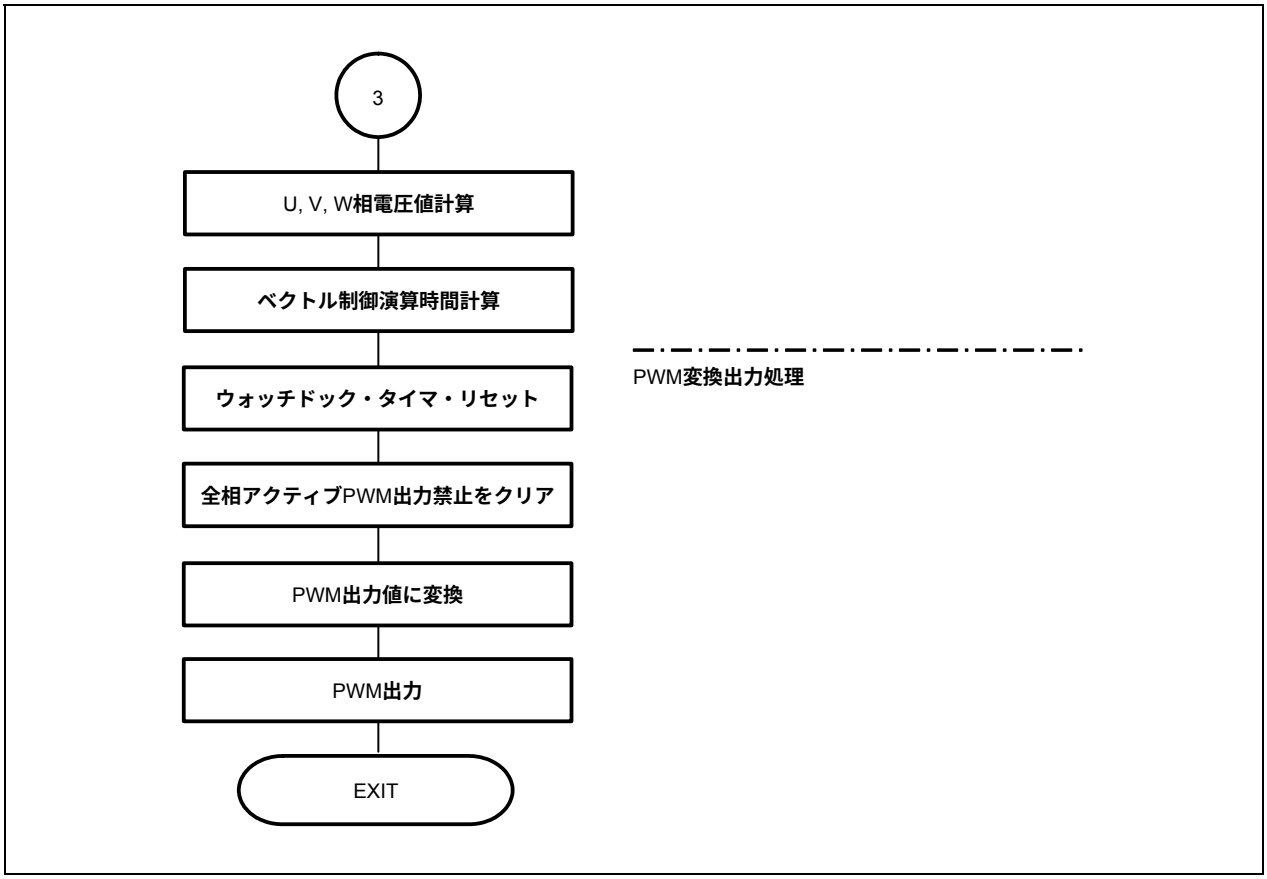
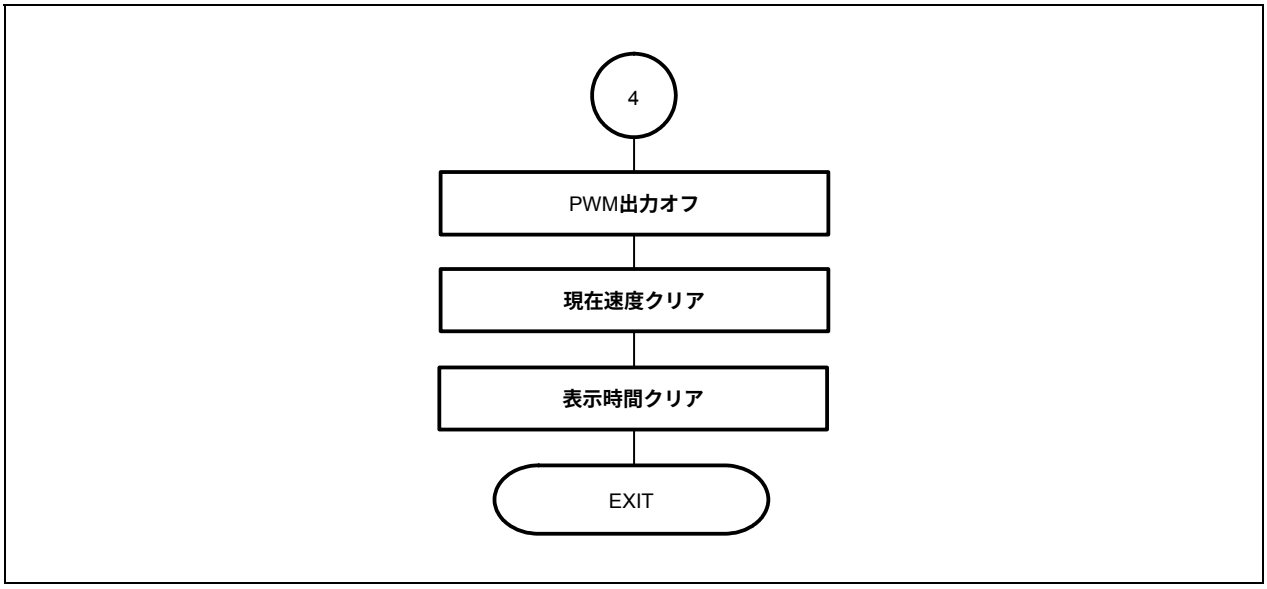
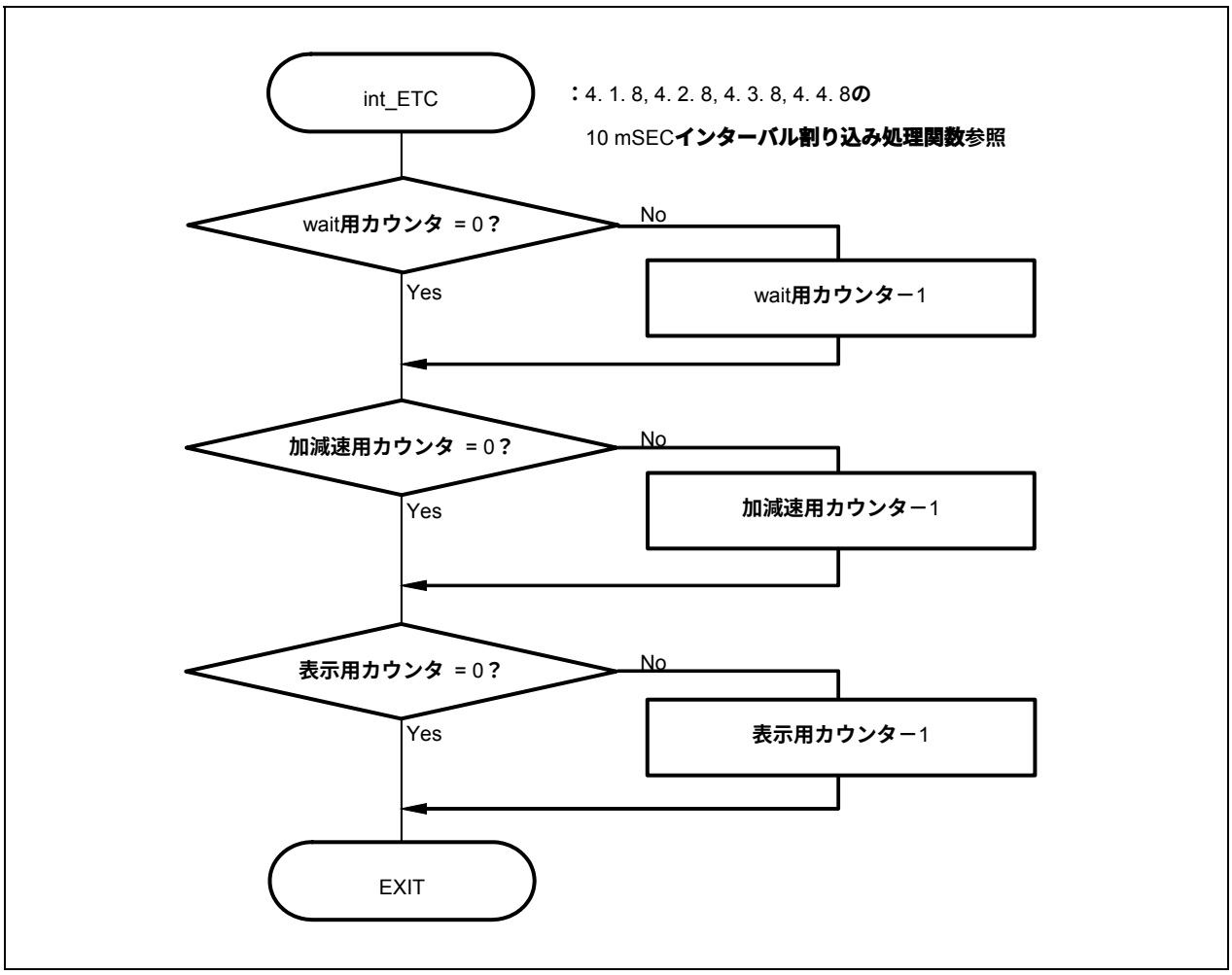


図3－15　制御割り込み処理（5/5）



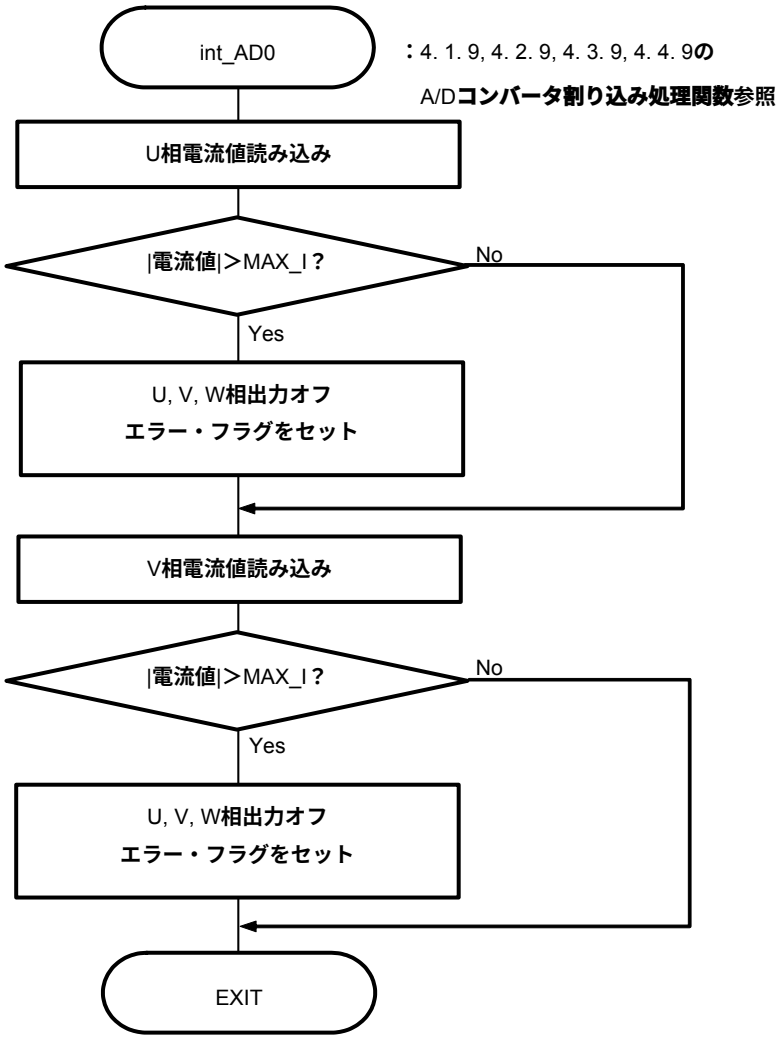
3. 9. 5　10 mSECインターバル割り込み処理

図3－16　10 mSECインターバル割り込み処理



3. 9. 6　A/Dコンバータ・チャネル0割り込み処理

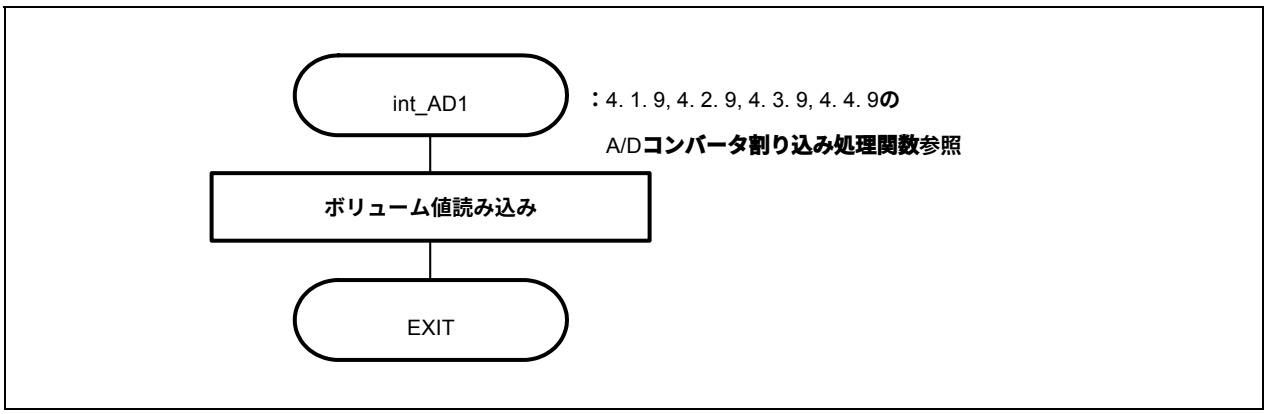
図3－17　A/Dコンバータ・チャネル0割り込み処理



備考　MAX_I：電流最大値

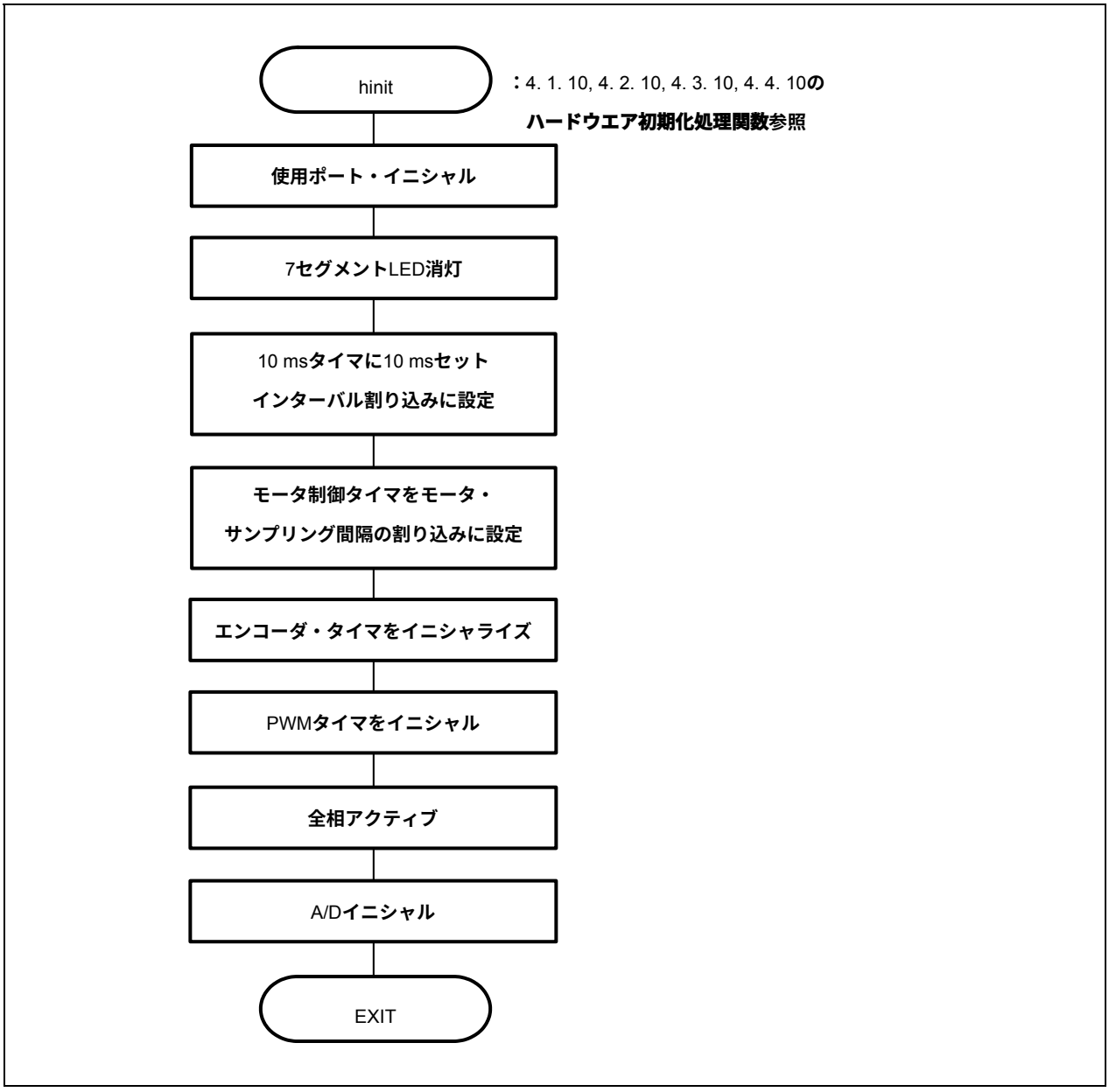
3. 9. 7　A/Dコンバータ・チャネル1割り込み処理

図3－18　A/Dコンバータ・チャネル1割り込み処理



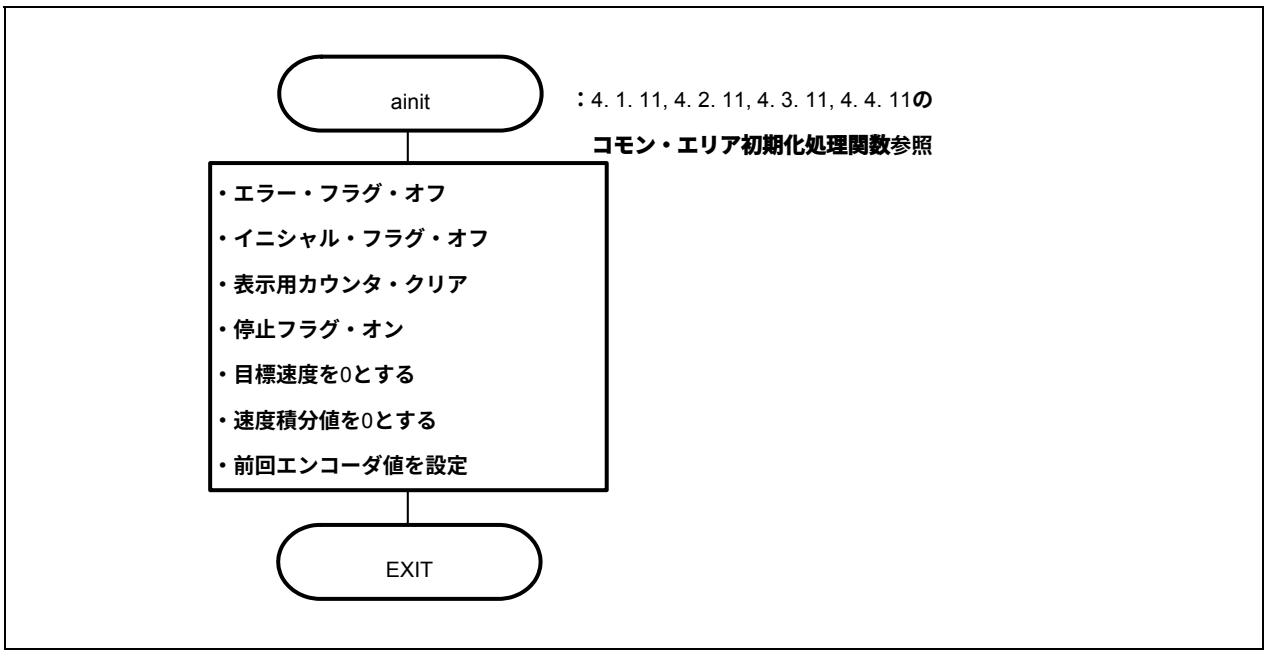
3. 9. 8　ハードウエア初期化

図3－19　ハードウエア初期化



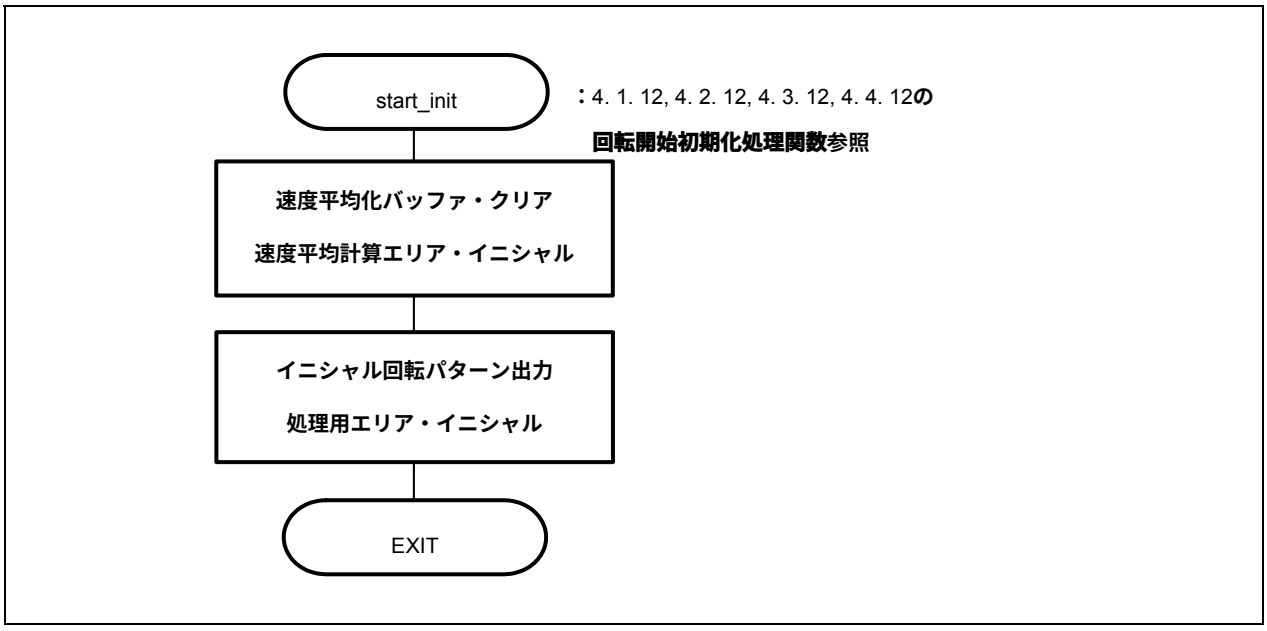
3. 9. 9　コモン・エリア初期化

図3－20　コモン・エリア初期化



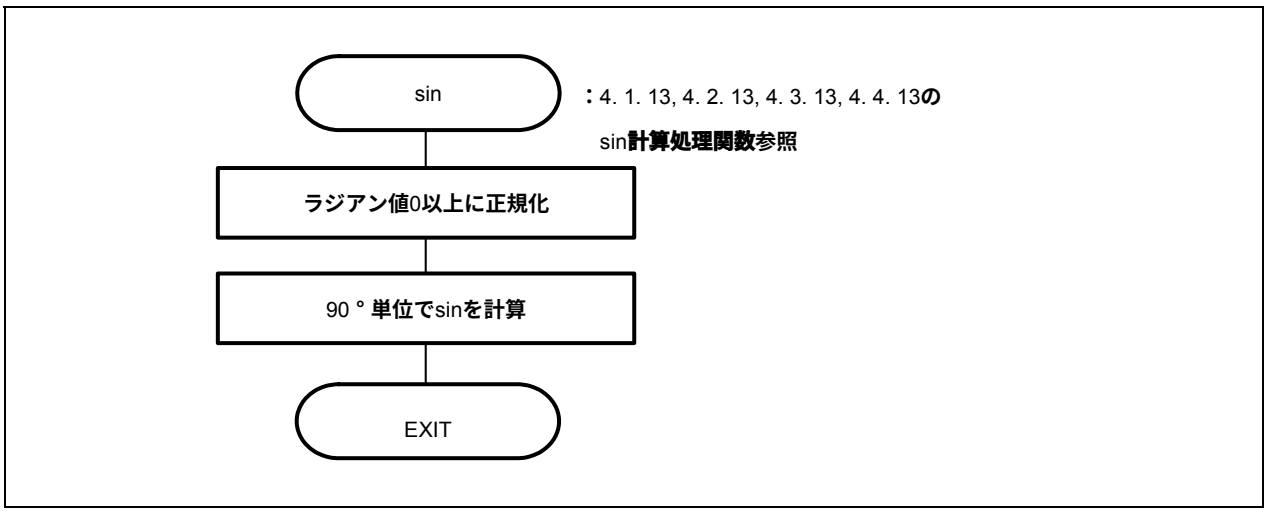
3. 9. 10　回転開始初期化

図3－21　回転開始初期化



3. 9. 11　sin計算

図3－22　sin計算



3. 10　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_NO2 ：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 char	停止フラグ	ON　：停止中 OFF：回転中
before_posi[21][2]	signed short	位置バッファ	
total_sa	signed short	位置トータル差分	
sum_speed	signed int	速度平均計算用合計値エリア	0，1, …
speed_co	signed 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	unsigned 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	速度推定値基準点	
sa_time	unsigned int	速度計測値	
timer_count	unsigned short	時間待ちカウンタ	
accel_count	unsigned short	加減速操作時間カウンタ	
o_trm	signed int	イニシャル用目標位置	
bfore_enc	signed short	前回エンコーダ値	

3. 11　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, 〜 };
```

3. 12　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[注]
PAI	π	3.141592
TH_U	ラジアン値，ゲタ定数	1000
RAD	1回転のラジアン値	2×PAI×TH_U
OFFSET	原点OFFSET	1733
RPM_RADS	rpm→ラジアン変換定数	2×PAI×TH_U／60
KSP	速度比例定数	500
KSI	速度積分定数	50
KI	電流変換定数	1
P	モータ極数	2
KSPGETA	速度比例定数ゲタ定数	0
KSIGETA	速度積分定数ゲタ定数	9
KIGETA	電流変換定数ゲタ定数	9
SGETA	sinゲタ定数	14
PWM_TS	PWM周期	80 μs
PWM_DATA	PWM設定値	PWM_TS／0.03125
SPEED_MAX	最大速度	500（3000 rpm）
SPEED_MINI	最低速度	100（600 rpm）
SA_SPEED_MAX	最大速度差	800
IQAMAX	最大速度積分値	40000000
MAX_I	電流最大値	800
TS	モータ制御周期	80 μs
WATCH_START	速度監視開始時間　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
MAXPULSE	1回転のパルス数	10000

注　V850E/IA3, V850E/IA4で使用する値。

V850E/IA1, V850E/IA2で使用する値については，プログラム・リストで確認してください。

第4章　プログラム・リスト

4. 1　プログラム・リスト（V850E/IA1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 short    before_posi[21][2] ;            /* 位置バッファ */
signed short    total_sa ;                      /* 位置トータル差分 */
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 ;

#pragma section const begin
const unsigned short led_pat[10] = { 0xfc, 0x60, 0xda, 0xf2, 0x66, 0xb6, 0xbe, 0xe0, 0xfe,
                                     0xe6 } ;
#pragma section const end
/************************************************************************** /
/*      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 short     before_posi[21][2] ;    /* 位置バッファ */
```

```
extern signed short     total_sa ;              /* 位置トータル差分 */
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 ;

#pragma section const begin
extern const unsigned short led_pat[] ;;
#pragma section const end
/************************************************************************** /
/*      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 int       o_iqai ;                /* 速度積分値エリア */
extern signed int       o_trm ;                 /* イニシャル用目標位置 */
extern signed int       base_position ;         /* 速度推定値基準点 */
extern unsigned int     sa_time ;               /* 速度計測値 */
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 signed short     before_enc ;            /* 前回エンコーダ値 */
```

4.1.2　定数定義

```
/************************************************************************** /
/*      I/O                                                                 */
/************************************************************************** /
#define BASE_IO      0xc200000
#define LED11        3
#define LED12        2
#define LED13        1
#define LED14        0
#define LED21        5
#define LED22        4
#define LED31        7
#define LED32        6
#define LED41        9
#define LED42        8

#define DIPSW        0x10
#define SW           0x20
#define DA1          0x30
#define DA2          0x40
#define DA3          0x50
#define WRESET       0x60
#define MODE         0x70
```

```
/************************************************************************** /
/*      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                                                         */
/************************************************************************** /
#define PAI             3.14159265          /* π */
#define TH_U            1000                /* ラジアン値　ゲタ定数 */
#define RAD             (int)(2*PAI*TH_U)   /* 1回転のラジアン値 */
#define OFFSET          1945                /* 原点OFFSET */
#define RPM_RADS        (int)((2*PAI*TH_U)/60)  /* rpm->ラジアン変換定数 */
/* モータ定数 */
#define KSP             500                 /* 速度比例定数 */
#define KSI             50                  /* 速度積分定数 */
#define KI              1                   /* 電流変換定数 */
#define P               2                   /* 極数 */

#define KSPGETA         0                   /* KSP  ゲタ定数 */
#define KSIGETA         9                   /* KSI ゲタ定数 */
#define KIGETA          9                   /* KI  ゲタ定数 */
#define SGETA           14                  /* sin ゲタ定数 */

#define PWM_TS          50                  /* PWM 周期 */
#define PWM_DATA        (PWM_TS/0.05)       /* PWM 設定値 */
#define SPEED_MAX       3000                /* 最大速度 rpm */
#define SPEED_MINI      600                 /* 最低速度 rpm */
#define SA_SPEED_MAX    800                 /* 最大速度差分 rpm */
#define IQAMAX          40000000            /* 最大速度積分値 */
#define MAX_I           800                 /* 電流 MAX値 */
#define TS              200                 /* モータ制御時間間隔 uSEC */
#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 MAXPULSE        10000               /* 1回転のパルス数 */
/************************************************************************** /
/*      関数定義                                                           */
/************************************************************************** /
void            fcalcu( signed int *wrm, signed int *trm );
void            OUT_data( unsigned short reg, unsigned short data );
unsigned short  IN_data( int reg );
void            led_num( int no, long data ) ;
```

```
/************************************************************************** /
/*      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 int      o_iqai ;                        /* 速度積分値エリア */
signed int      o_trm ;                         /* イニシャル用目標位置 */
signed int      base_position ;                 /* 速度推定値基準点 */
unsigned int    sa_time ;                       /* 速度計測値 */
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 */
signed short    before_enc ;                    /* 前回エンコーダ値 */
```

4.1.3　割り込みハンドラ設定

```
/************************************************************************** /
/*      割り込みシンボル・テーブル                                          */
/************************************************************************** /
        .extern _ _start
        .extern _int_MOTOR
        .extern _int_AD0
        .extern _int_AD1
        .extern _int_ETC

        .globl V_RESET
        .globl V_ETC
        .globl V_MOTOR
        .globl V_AD0
        .globl V_AD1
#************************************************************************
        .section ".handler",text
V_RESET:
        jr      _start
V_ETC:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_ETC                        -- その他タイマ
V_MOTOR:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_MOTOR                      -- 速度制御タイマ
V_AD0:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_AD0                        -- A/Dコンバータ CH0
V_AD1:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_AD1                        -- A/Dコンバータ CH1
```

```
        .extern V_RESET
        .extern V_ETC
        .extern V_MOTOR
        .extern V_AD0
        .extern V_AD1
/************************************************************************** /
/*      割り込みジャンプ・テーブル                                          */
/************************************************************************** /
        .section ".vect_RESET",text
        mov     #V_RESET,r1
        jmp     [r1]

        .section ".id_NO",text
        .byte   0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff

        .section ".vect_ETC",text
        add     -4,sp
        st.w    r1,[r3]
        mov     #V_ETC,r1
        jmp     [r1]

        .section ".vect_MOTOR",text
        add     -4,sp
        st.w    r1,[r3]
        mov     #V_MOTOR,r1
        jmp     [r1]

        .section ".vect_AD0",text
        add     -4,sp
        st.w    r1,[r3]
        mov     #V_AD0,r1
        jmp     [r1]

        .section ".vect_AD1",text
        add     -4,sp
        st.w    r1,[r3]
        mov     #V_AD1,r1
        jmp     [r1]
```

4.1.4　スタートアップ・ルーチン設定

```
#==============================================================================
# DESCRIPTIONS:
#     This assembly program is a sample of start-up module for ca850.
#   If you modified this program, you must assemble this file, and
#   locate a given directory.
#
#   Unless -G is specified, sections are located as the following.
#
#                 |       :       |
```

```
#                  |       :       |
#           tp -> -+-------------- + _ _start    _ _tp_TEXT
#                  | start up      |
#                  |-------------- |
# text section     |               |
#                  | user program  |
#                  |               |
#                  |-------------- |
#                  | library       |
#                 -+-------------- +
#                  |               |
# sdata section    |               |
#                  |               |
#           gp -> -+-------------- +            _ _ssbss
#                  |               |
# sbss section     |               |
#                  |               |
#                   +-------------- + _ _stack    _ _esbss        _ _sbss
#                  | stack area    |
# bss section      |               |
#                  | 0x400 bytes   |
#           sp -> -+-------------- + _ _stack + STACKSIZE       _ _ebss
#                  |       :       |
#                  |       :       |
#                  |       :       |
#           ep -> -+-------------- + _ _ep_DATA
# tidata section   |               |
#                 -+-------------- +
# sidata section   |               |
#                 -+-------------- +
#                  |       :       |
#                  |       :       |
#
#===============================================================================
#-------------------------------------------------------------------------------
#       special symbols
#-------------------------------------------------------------------------------
        .extern _ _tp_TEXT, 4
        .extern _ _gp_DATA, 4
        .extern _ _ep_DATA, 4
        .extern _ _ssbss, 4
        .extern _ _esbss, 4
        .extern _ _sbss, 4
        .extern _ _ebss, 4


#-------------------------------------------------------------------------------
#       C program main function
#-------------------------------------------------------------------------------
        .extern _main
```

```
#------------------------------------------------------------------------------
#       dummy data declaration for creating sbss section
#------------------------------------------------------------------------------
        .sbss
        .lcomm  _ _sbss_dummy, 0, 0


#------------------------------------------------------------------------------
#       system stack
#------------------------------------------------------------------------------
        .set    STACKSIZE, 0x400
        .bss
        .lcomm  _ _stack, STACKSIZE, 4

#------------------------------------------------------------------------------
#       start up
#            pointers:  tp - text pointer
#                       gp - global pointer
#                       sp - stack pointer
#                       ep - element pointer
#       exit status is set to r10
#------------------------------------------------------------------------------
        .text
        .align 4
        .globl _ _start
        .globl _exit
        .globl _ _exit
_ _start:
        mov     0x12,r10
        st.b    r10,VSWC[r0]                    -- 周辺I/Oウエイト設定

        mov     0x07,r10                        -- ×10
        st.b    r0,PHCMD[r0]
        st.b    r10,CKC[r0]                     -- PLL xx逓倍
        nop
        nop
        nop
        nop
        nop

        mov     #_ _tp_TEXT, tp                 -- set tp register
        mov     #_ _gp_DATA, gp                 -- set gp register offset
        add     tp, gp                          -- set gp register
        mov     #_ _stack+STACKSIZE, sp         -- set sp register
        mov     #_ _ep_DATA, ep                 -- set ep register
#
        mov     #_ _ssbss, r13                  -- clear sbss section
        mov     #_ _esbss, r12
```

```
        cmp     r12, r13
        jnl     .L11
.L12:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L12
.L11:
#
        mov     #_ _sbss, r13                   -- clear bss section
        mov     #_ _ebss, r12
        cmp     r12, r13
        jnl     .L14
.L15:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L15
.L14:
#
        jarl    _main, lp                       -- call main function
_ _exit:
        halt                                    -- end of program
_ _startend:
        nop
#                                                                               #
#-------------------------- end of start up module ----------------------------#
#                                                                               #
```

4.1.5　メイン処理関数

```
#include    "Common.h"
#include    "Motor.h"
#pragma     ioreg                               /* 周辺I/Oレジスタ定義 */

static int save_psw;
/*************************************************************************** /
/*      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 ) * 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 ;
      }
      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 / 100 );             /* 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 );                 /* 処理時間全体 */
        led_num(4, cont_time1 );                /* ベクトル演算処理時間 */
    }
    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 ) ;
          }
       }
    }
}
```

4.1.6　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 ) {
                P4.3 = 0;
                data = 1;                       /* ダミー・ステップ */
                P4.3 = 1;
```

```
        } else {
                PDL = data | ( reg << 8 );
                PDL = reg | ( reg << 8 ) | 0x8000;
        }
}
/************************************************************************** /
/*      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 P4;
        } else {
                return 0;
        }
}
```

4.1.7　モータ制御割り込み処理関数

```
#include    "Common.h"
#include    "Motor.h"
#pragma     ioreg                                   /* 周辺I/Oレジスタ定義 */
/************************************************************************** /
/*      モータ制御タイマ割り込み処理                                        */
/************************************************************************** /
_ _interrupt
void    int_MOTOR(void)
{
   ADSCM00 = 0x8001 ;                               /* AD0 起動 */
   ADSCM10 = 0x8000 ;                               /* AD1 起動 */
}
/************************************************************************** /
/*      モータ制御処理                                                      */
/************************************************************************** /
void    Motor_CONT(void)
{
signed int      wrm, wre, trm, tre ;
signed int      o_wre, we, o_iqap, o_iqa ;
signed int      ida, iqa, o_vda1, o_vqa1 ;
signed int      o_vd0, o_vq0, o_vda, o_vqa ;
signed int      o_vua, o_vva, o_vwa, wiua, wiva ;
signed int      s_time, wk ;
signed short    now_enc, sa_enc ;
/*  */
```

```
/*************************************************************************** */
/*      速度およびロータ位置の計算処理                                       */
/*************************************************************************** */
   now_enc = -TM10 ;
   sa_enc = now_enc - before_enc ;
   if  ( abs( sa_enc ) > ( MAXPULSE / 2 ) ) {
      if  ( sa_enc < 0 ) {
         sa_enc += MAXPULSE ;
      } else {
         sa_enc -= MAXPULSE ;
      }
   }
   before_enc = now_enc ;
   wrm = now_speed = (signed long)sa_enc * RAD / MAXPULSE;     /* 現在速度セット */

   wre = wrm * P ;
   tre = ( trm * P + OFFSET ) % RAD ;
/*************************************************************************** */
/*      速度制御処理                                                         */
/*************************************************************************** */
   if  ( ( stop_flag == OFF ) && ( error_flag == 0 ) ) {
      s_time = TM3 ;
      o_wre = object_speed * RPM_RADS * P / TH_U ;             /* rpm->ラジアン変換 */
      we = o_wre - wre ;
      o_iqap = ( ( wre * KSP ) + ( we * KSP ) ) >> KSPGETA ;
      o_iqa  = o_iqap + ( o_iqai >> KSIGETA ) ;

      if  ( o_iqai > IQAMAX ) {
         o_iqai = IQAMAX ;
      } else if ( o_iqai < -IQAMAX ) {
         o_iqai = -IQAMAX ;
      } else {
         o_iqai += ( KSI * we ) ;
      }
/*************************************************************************** */
/*      電流制御処理                                                         */
/*************************************************************************** */
      ida = ( ( ( iva * sin( tre ) ) - ( iua * sin( tre + 2*RAD/3 ) ) ) ) >> SGETA ;
      iqa = ( ( ( iva * sin( tre + RAD/4 ) ) - ( iua * sin( tre + 11*RAD/12 ) ) ) ) >> SGETA ;

      o_vda = ( KI * -ida ) >> KIGETA ;
      o_vqa = ( KI * ( o_iqa  - iqa )  ) >> KIGETA ;
/*************************************************************************** */
/*      3相電圧変換処理                                                      */
/*************************************************************************** */
      if ( init_flag == 2 ) {                                  /* CW初起動処理 */
         o_trm += ( SPEED_MINI * TS / 20000 ) ;
         tre = ( o_trm * P ) % RAD ;
         o_vda = 0 ;
         o_vqa = 100 ;
```

```
            if  ( o_trm > ( 2 * RAD ) ) {
                init_flag = 0;
            }
        } else if ( init_flag == 3 ) {          /* CCW初起動処理 */
            o_trm -= ( SPEED_MINI * TS / 20000 ) ;
            tre = ( o_trm * P ) % RAD ;
            o_vda = 0 ;
            o_vqa = 100 ;
            if  ( o_trm < -( 2 * RAD ) ) {
                init_flag = 0;
            }
        } else {
            o_trm = 0 ;
        }
        o_vua = ( ( ( o_vda * sin( tre + RAD/4 ) ) - ( o_vqa * sin( tre ) ) ) ) >> SGETA ;
        o_vva = ( ( ( o_vda * sin( tre + 11*RAD/12 ) ) - ( o_vqa * sin( tre + 2*RAD/3 ) ) ) ) >> SGETA ;
        o_vwa = -o_vua - o_vva ;

        cont_time1 = ( TM3 - s_time ) * 10 / 16; /* uSECに変換 */
/************************************************************************** /
/*      PWM変換出力処理                                                    */
/************************************************************************** /
        OUT_data( WRESET, 0 ) ;                 /* ウォッチドック・タイマRESET */
        POER0 = 0x3f ;                          /* 全相アクティブ */
        TUC00 = 0x02 ;                          /* PWM 出力禁止クリア */
/* PWMカウンタ値計算出力 */
        o_vua += ( PWM_DATA  / 2 ) ;
        o_vva += ( PWM_DATA  / 2 ) ;
        o_vwa += ( PWM_DATA  / 2 ) ;

        if  ( o_vua <= 0 ) {
            o_vua = 1 ;
        } else if  ( o_vua >= PWM_DATA  ) {
            o_vua = PWM_DATA  - 1 ;
        }
        if  ( o_vva <= 0 ) {
            o_vva = 1 ;
        } else if  ( o_vva >= PWM_DATA  ) {
            o_vva = PWM_DATA  - 1 ;
        }
        if  ( o_vwa <= 0 ) {
            o_vwa = 1 ;
        } else if  ( o_vwa >= PWM_DATA  ) {
            o_vwa = PWM_DATA  - 1 ;
        }

        BFCM00 = o_vua ;
        BFCM01 = o_vva ;
        BFCM02 = o_vwa ;
    } else {
```

```
        POER0 = 0 ;                                 /* PWM出力OFF */
        now_speed = 0;
        cont_time1 = 0;
    }
}
```

4.1.8　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）                    */
/************************************************************************** /
_ _multi_interrupt
void    int_ETC(void)
{
unsigned short dummy ;
/* 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 += abs( now_speed ) ;
        disp_co -= 1 ;
    }
}
```

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                    */
/************************************************************************** /
_ _multi_interrupt
void    int_AD0(void)
{
    iua = (( ADCR00 & 0x3ff ) - 0x200) ;
    if  ( abs(iua) > MAX_I ) {
        POER0 = 0 ;                             /* PWM出力OFF */
        error_flag = ERR_NO1 ;                  /* エラーNOセット */
    }
    volume = 1023 - ( ADCR01 & 0x3ff ) ;        /* ボリューム値セット */
    Motor_CONT() ;
    cont_time = TM3 * 10 / 16;                  /* uSECに変換 */
}
```

```
/************************************************************************** /
/*      V相電流用A/Dコンバータ割り込み処理                                  */
/************************************************************************** /
_ _interrupt
void    int_AD1(void)
{
    iva = (( ADCR10 & 0x3ff ) - 0x200) ;
    if  ( abs(iva) > MAX_I ) {
        POER0 = 0 ;                             /* PWM出力OFF */
        error_flag = ERR_NO1 ;                  /* エラーNOセット */
    }
}
```

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 )
{
short   dummy ;
/* ポート・モード・レジスタ初期化 */
    PMC1  = 0x3f ;                              /* ENC select */
    PMC2  = 0x07 ;
    PM2   = 0x07 ;

    PM4 = 0xf7 ;
    PMDL = 0x0000 ;

    OUT_data( LED11, 0xff ) ;                   /* LED OFF */
    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 ) ;
/* 10 mSECタイマ  TM2設定 */
    STOPTE0 = 0x0000;
    PRM02 = 0x01;                               /* fXX/2セレクト */
    CSE0 = 0x0028;                              /* fCLK/64(3.2 uSEC)セレクト */
    TCRE0 = 0x2000;                             /* タイマ・スタート */
    CVSE50 = 1562*2 ;                           /* 10 mSEC */
    CMSE050 = 0x2400;
    CC2IC5 = 0x06;
/* モータ制御割り込み用タイマ  TM3設定 */
    PRM03 = 1 ;                                 /* fCLK = fXX */
    TMC30 = 0x51;                               /* fXX = 4 MHz*10/64(1.6 uSEC) */
```

```
    TMC31 = 0x09 ;
    CC30  = TS * 10 / 16 ;                  /* TS uSEC インターバル */
    TMC30 = 0x53;                           /* タイマ・スタート */
    CC3IC0 = 0x02;                          /* 割り込みマスクをリセット */
/* TM10 初期化 */
    PRM02 = 1 ;                             /* fCLK = fXX/2 */
    TUM0 = 0x80 ;                           /* UDC モード選択 */
    PRM10 = 0x07 ;                          /* 動作モード4選択 */
    TMC10 = 0x40 ;                          /* カウント開始 */
/* TM00 初期化 */
    PRM01 = 0 ;                             /* fCLK = fXX/2 */
    SPEC0 = 0x0000 ;
    TOMR0 = 0x80 ;                          /* 出力モード設定 */
    PSTO0 = 0x00 ;                          /* リアルタイム出力禁止 */
    BFCM00 = PWM_DATA /2 ;                  /* デューティ50設定 */
    BFCM01 = PWM_DATA /2 ;                  /* デューティ50設定 */
    BFCM02 = PWM_DATA /2 ;                  /* デューティ50設定 */
    BFCM03 = PWM_DATA ;                     /* PWM周期 設定 */
    DTRR0 = 40*3 ;                          /* デットタイム 6 uSEC */
    POER0 = 0x3f ;                          /* 全相アクティブ */
    TMC00 = 0x8018 ;                        /* TMPWM タイマ開始 */
/* A/D 設定 */
    ADSCM00 = 0x0001 ;
    ADSCM10 = 0x0000 ;
    ADIC0 = 0x03 ;
    ADIC1 = 0x03 ;
}
```

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 */
    o_trm = 0 ;
    before_enc  = -TM10 ;                   /* 前回エンコーダ値を設定 */
}
```

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;
/*  */
    for ( i = 0 ; i < 21 ; i++ ) before_posi[i][1] = 0;
    total_sa = 0 ;
    sum_speed = 0 ;
    speed_co = 100000 / TS ;
}
```

4. 1. 13　sin計算処理関数

```
/*************************************************************************** */
/*      sin x                                                                */
/*          データ                                                           */
/*              ラジアン単位                                                 */
/*          戻り値                                                           */
/*              サイン値*16384                                               */
/*************************************************************************** */
int sin( int x )
{
    x = x % RAD ;
    if  ( x < 0 ) x += RAD ;
    if  ( x < (RAD/4) ){
        return sins( x ) ;
    } else if ( x < (RAD/2) ) {
        return sins( (RAD/2) - x ) ;
    } else if ( x < (RAD*3/4) ) {
        return -sins( x - (RAD/2) ) ;
    } else {
        return -sins( RAD - x ) ;
    }
}
/*************************************************************************** */
int sins( int x )
{
short z1, z2, z3, z4, z5 ;
/*  */
    if  ( x <= (RAD/8) ) {
        z1 = (x << SGETA) / (RAD/8) ;
        z2 = z1 * z1 >> SGETA ;
        z3 = z1 * z2 >> SGETA ;
        z5 = z2 * z3 >> SGETA ;
        return ( (12868*z1) - (1322*z3) + (40*z5) ) >> SGETA ;
    } else {
```

```
        x = (RAD/4) - x ;
        z1 = (x << SGETA) / (RAD/8) ;
        z2 = z1 * z1 >> SGETA ;
        z4 = z2 * z2 >> SGETA ;
        return ( (268432772) - (5050*z2) + (252*z4) ) >> SGETA ;
    }
}
```

4. 1. 14　V850E/IA1用のリンク・ディレクティブ・ファイル

```
/************************************************************************** /
/*      V850E/IA1用のリンク・ディレクティブ・ファイル                        */
/************************************************************************** /
VECT_RESET: !LOAD ?RX V0x0000000 {
        .vect_RESET  = $PROGBITS ?AX .vect_RESET;
};
ID_NO: !LOAD ?RX V0x0000070 {
        .id_NO  = $PROGBITS ?AX .id_NO;
};
VECT_ETC: !LOAD ?RX V0x0000240 {
        .vect_ETC  = $PROGBITS ?AX .vect_ETC;
};
VECT_MOTOR: !LOAD ?RX V0x0000260 {
        .vect_MOTOR  = $PROGBITS ?AX .vect_MOTOR;
};
VECT_AD0: !LOAD ?RX V0x00003a0 {
        .vect_AD0  = $PROGBITS ?AX .vect_AD0;
};
VECT_AD1: !LOAD ?RX V0x00003b0 {
        .vect_AD1  = $PROGBITS ?AX .vect_AD1;
};

HANDLER: !LOAD ?RX V0x00001000 {
        .handler  = $PROGBITS ?AX .handler;
};
TEXT:  !LOAD ?RX {
        .text  = $PROGBITS ?AX .text;
};

CONST : !LOAD ?R {
        .const = $PROGBITS ?A .const;
};

DATA : !LOAD ?RW V0x0fffc000 {
        .data  = $PROGBITS ?AW  ;
        .sdata = $PROGBITS ?AWG ;
        .sbss  = $NOBITS    ?AWG ;
        .bss   = $NOBITS    ?AW  ;
};
```

```
_ _tp_TEXT @ %TP_SYMBOL;
_ _gp_DATA @ %GP_SYMBOL &_ _tp_TEXT{DATA};
_ _ep_DATA @ %EP_SYMBOL;
```

4. 2　プログラム・リスト（V850E/IA2用）

4. 2. 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 short    before_posi[21][2] ;        /* 位置バッファ */
signed short    total_sa ;                  /* 位置トータル差分 */
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 ;

#pragma section const begin
const unsigned short led_pat[10] = { 0xfc, 0x60, 0xda, 0xf2, 0x66, 0xb6, 0xbe, 0xe0, 0xfe,
                                     0xe6 } ;
#pragma section const end
/************************************************************************** /
/*      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 short     before_posi[21][2] ;/* 位置バッファ */
extern signed short     total_sa ;          /* 位置トータル差分 */
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 ;

#pragma section const begin
extern const unsigned short led_pat[] ;;
#pragma section const end
/************************************************************************** /
/*      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 int       o_iqai ;                /* 速度積分値エリア */
extern signed int       o_trm ;                 /* イニシャル用目標位置 */
extern signed int       base_position ;         /* 速度推定値基準点 */
extern unsigned int     sa_time ;               /* 速度計測値 */
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 signed short     before_enc ;            /* 前回エンコーダ値 */
```

4.2.2　定数定義

```
/************************************************************************** /
/*      I/O                                                                 */
/************************************************************************** /
#define BASE_IO      0xc200000
#define LED11        3
#define LED12        2
#define LED13        1
#define LED14        0
#define LED21        5
#define LED22        4
#define LED31        7
#define LED32        6
#define LED41        9
#define LED42        8

#define DIPSW        0x10
#define SW           0x20
#define DA1          0x30
#define DA2          0x40
#define DA3          0x50
#define WRESET       0x60
#define MODE         0x70
/************************************************************************** /
/*      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                                                          */
/************************************************************************** /
#define PAI             3.14159265             /* π */
#define TH_U            1000                   /* ラジアン値　ゲタ定数 */
#define RAD             (int)(2*PAI*TH_U)      /* 1回転のラジアン値 */
#define OFFSET          1945                   /* 原点OFFSET */
#define RPM_RADS        (int)((2*PAI*TH_U)/60) /* rpm->ラジアン変換定数 */
/* モータ定数 */
#define KSP             500                    /* 速度比例定数 */
#define KSI             50                     /* 速度積分定数 */
#define KI              1                      /* 電流変換定数 */
#define P               2                      /* 極数 */

#define KSPGETA         0                      /* KSP  ゲタ定数 */
#define KSIGETA         9                      /* KSI ゲタ定数 */
#define KIGETA          9                      /* KI  ゲタ定数 */
#define SGETA           14                     /* sin ゲタ定数 */

#define PWM_TS          50                     /* PWM 周期 */
#define PWM_DATA        (PWM_TS/0.05)          /* PWM 設定値 */
#define SPEED_MAX       3000                   /* 最大速度 rpm */
#define SPEED_MINI      600                    /* 最低速度 rpm */
#define SA_SPEED_MAX    800                    /* 最大速度差分 rpm */
#define IQAMAX          40000000               /* 最大速度積分値 */
#define MAX_I           800                    /* 電流 MAX値 */
#define TS              200                    /* モータ制御時間間隔  uSEC */
#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 MAXPULSE        10000                  /* 1回転のパルス数 */
/************************************************************************** /
/*      関数定数                                                            */
/************************************************************************** /
void            fcalcu( signed int *wrm, signed int *trm );
void            OUT_data( unsigned short reg, unsigned short data );
unsigned short  IN_data( int reg );
void            led_num( int no, long data ) ;
/************************************************************************** /
/*      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 int      o_iqai ;                       /* 速度積分値エリア */
signed int      o_trm ;                        /* イニシャル用目標位置 */
signed int      base_position ;                /* 速度推定値基準点 */
```

```
unsigned int    sa_time ;                       /* 速度計測値 */
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 */
signed short    before_enc ;                    /* 前回エンコーダ値 */
```

4. 2. 3　割り込みハンドラ設定

```
/************************************************************************** /
/*      割り込みシンボル・テーブル                                          */
/************************************************************************** /
        .extern _ _start
        .extern _int_MOTOR
        .extern _int_AD0
        .extern _int_AD1
        .extern _int_ETC

        .globl V_RESET
        .globl V_ETC
        .globl V_MOTOR
        .globl V_AD0
        .globl V_AD1
#************************************************************************
        .section ".handler",text
V_RESET:
        Jr      _ _start
V_ETC:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_ETC                        -- その他タイマ
V_MOTOR:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_MOTOR                      -- 速度制御タイマ
V_AD0:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_AD0                        -- A/Dコンバータ CH0
V_AD1:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_AD1                        -- A/Dコンバータ CH1

        .extern V_RESET
        .extern V_ETC
        .extern V_MOTOR
        .extern V_AD0
        .extern V_AD1
```

```
/************************************************************************** /
/*      割り込みジャンプ・テーブル                                          */
/************************************************************************** /
        .section ".vect_RESET",text
        mov     #V_RESET,r1
        jmp     [r1]

        .section ".id_NO",text
        .byte   0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff

        .section ".vect_ETC",text
        add     -4,sp
        st.w    r1,[r3]
        mov     #V_ETC,r1
        jmp     [r1]

        .section ".vect_MOTOR",text
        add     -4,sp
        st.w    r1,[r3]
        mov     #V_MOTOR,r1
        jmp     [r1]

        .section ".vect_AD0",text
        add     -4,sp
        st.w    r1,[r3]
        mov     #V_AD0,r1
        jmp     [r1]

        .section ".vect_AD1",text
        add     -4,sp
        st.w    r1,[r3]
        mov     #V_AD1,r1
        jmp     [r1]
```

4.2.4　スタートアップ・ルーチン設定

```
#==============================================================================
# DESCRIPTIONS:
#     This assembly program is a sample of start-up module for ca850.
#   If you modified this program, you must assemble this file, and
#   locate a given directory.
#
#   Unless -G is specified, sections are located as the following.
#
#                  |       :       |
#                  |       :       |
#          tp -> -+--------------- + _ _start    _ _tp_TEXT
#                  | start up      |
#                  |-------------- |
# text section     |               |
```

```
#                  | user program  |
#                  |               |
#                  |-------------- |
#                  | library       |
#                 -+-------------- +
#                  |               |
# sdata section    |               |
#                  |               |
#           gp -> -+-------------- +            _ _ssbss
#                  |               |
# sbss section     |               |
#                  |               |
#                   +-------------- + _ _stack    _ _esbss        _ _sbss
#                  | stack area    |
# bss section      |               |
#                  | 0x400 bytes   |
#           sp -> -+-------------- + _ _stack + STACKSIZE       _ _ebss
#                  |       :       |
#                  |       :       |
#                  |       :       |
#           ep -> -+-------------- + _ _ep_DATA
# tidata section   |               |
#                 -+-------------- +
# sidata section   |               |
#                 -+-------------- +
#                  |       :       |
#                  |       :       |
#
#==============================================================================
#------------------------------------------------------------------------------
#       special symbols
#------------------------------------------------------------------------------
        .extern _ _tp_TEXT, 4
        .extern _ _gp_DATA, 4
        .extern _ _ep_DATA, 4
        .extern _ _ssbss, 4
        .extern _ _esbss, 4
        .extern _ _sbss, 4
        .extern _ _ebss, 4


#------------------------------------------------------------------------------
#       C program main function
#------------------------------------------------------------------------------
        .extern _main


#------------------------------------------------------------------------------
#       dummy data declaration for creating sbss section
#------------------------------------------------------------------------------
```

```
        .sbss
        .lcomm  _ _sbss_dummy, 0, 0


#------------------------------------------------------------------------------
#       system stack
#------------------------------------------------------------------------------
        .set    STACKSIZE, 0x400
        .bss
        .lcomm  _ _stack, STACKSIZE, 4

#------------------------------------------------------------------------------
#       start up
#            pointers:  tp - text pointer
#                       gp - global pointer
#                       sp - stack pointer
#                       ep - element pointer
#       exit status is set to r10
#------------------------------------------------------------------------------
        .text
        .align 4
        .globl _ _start
        .globl _exit
        .globl _ _exit
_ _start:
        mov     0x02,r10
        st.b    r10,VSWC[r0]                    -- 周辺I/Oウエイト設定

        mov     0x07,r10                        -- ×10
        st.b    r0,PHCMD[r0]
        st.b    r10,CKC[r0]                     -- PLL xx逓倍
        nop
        nop
        nop
        nop
        nop

        mov     #_ _tp_TEXT, tp                 -- set tp register
        mov     #_ _gp_DATA, gp                 -- set gp register offset
        add     tp, gp                          -- set gp register
        mov     #_ _stack+STACKSIZE, sp         -- set sp register
        mov     #_ _ep_DATA, ep                 -- set ep register
#
        mov     #_ _ssbss, r13                  -- clear sbss section
        mov     #_ _esbss, r12
        cmp     r12, r13
        jnl     .L11
.L12:
        st.w    r0, [r13]
        add     4, r13
```

```
        cmp     r12, r13
        jl      .L12
.L11:
#
        mov     #_ _sbss, r13                   -- clear bss section
        mov     #_ _ebss, r12
        cmp     r12, r13
        jnl     .L14
.L15:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L15
.L14:
#
        jarl    _main, lp                       -- call main function
_ _exit:
        halt                                    -- end of program
_ _startend:
        nop
#                                                                               #
#-------------------------- end of start up module ----------------------------#
#                                                                               #
```

4.2.5　メイン処理関数

```
#include    "Common.h"
#include    "Motor.h"
#pragma     ioreg                               /* 周辺I/Oレジスタ定義 */

static int save_psw;
/*****************************************************************************/
/*      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 ) * 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 ;
        }
        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 / 100 );          /* 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 );              /* 処理時間全体 */
          led_num(4, cont_time1 );             /* ベクトル演算処理時間 */
       }
       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 ) ;
         }
      }
   }
}
```

4.2.6　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 ) {
                P3.0 = 0;
                data = 1;                       /* ダミー・ステップ */
                P3.0 = 1;
        } else {
                PDL = data | ( reg << 8 );
                PDL = reg | ( reg << 8 ) | 0x8000;
        }
}
```

```
/************************************************************************** /
/*      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 P4;
        } else {
                return 0;
        }
}
```

4.2.7　モータ制御割り込み処理関数

```
#include    "Common.h"
#include    "Motor.h"
#pragma     ioreg                                   /* 周辺I/Oレジスタ定義 */
/************************************************************************** /
/*      モータ制御タイマ割り込み処理                                        */
/************************************************************************** /
_ _interrupt
void    int_MOTOR(void)
{
   ADSCM00 = 0x8001 ;                               /* AD0 起動 */
   ADSCM10 = 0x8000 ;                               /* AD1 起動 */
}
/************************************************************************** /
/*      モータ制御処理                                                      */
/************************************************************************** /
void    Motor_CONT(void)
{
signed int      wrm, wre, trm, tre ;
signed int      o_wre, we, o_iqap, o_iqa ;
signed int      ida, iqa, o_vda1, o_vqa1 ;
signed int      o_vd0, o_vq0, o_vda, o_vqa ;
signed int      o_vua, o_vva, o_vwa, wiua, wiva ;
signed int      s_time, wk ;
signed short    now_enc, sa_enc ;
/*  */
/************************************************************************** /
/*      速度およびロータ位置の計算処理                                      */
/************************************************************************** /
   now_enc = -TM10 ;
   sa_enc = now_enc - before_enc ;
   if  ( abs( sa_enc ) > ( MAXPULSE / 2 ) ) {
      if  ( sa_enc < 0 ) {
         sa_enc += MAXPULSE ;
```

```
        } else {
            sa_enc -= MAXPULSE ;
        }
    }
    before_enc = now_enc ;
    wrm = now_speed = (signed long)sa_enc * RAD / MAXPULSE;     /* 現在速度セット */

    wre = wrm * P ;
    tre = ( trm * P + OFFSET ) % RAD ;
/************************************************************************** /
/*      速度制御処理                                                        */
/************************************************************************** /
    if  ( ( stop_flag == OFF ) && ( error_flag == 0 ) ) {
        s_time = TM3 ;
        o_wre = object_speed * RPM_RADS * P / TH_U ;            /* rpm->ラジアン変換 */
        we = o_wre - wre ;
        o_iqap = ( ( wre * KSP ) + ( we * KSP ) ) >> KSPGETA ;
        o_iqa = o_iqap + ( o_iqai >> KSIGETA ) ;

        if  ( o_iqai > IQAMAX ) {
            o_iqai = IQAMAX ;
        } else if ( o_iqai < -IQAMAX ) {
            o_iqai = -IQAMAX ;
        } else {
            o_iqai += ( KSI * we ) ;
        }
/************************************************************************** /
/*      電流制御処理                                                        */
/************************************************************************** /
        ida = ( ( ( iva * sin( tre ) ) - ( iua * sin( tre + 2*RAD/3 ) ) ) ) >> SGETA ;
        iqa = ( ( ( iva * sin( tre + RAD/4 ) ) - ( iua * sin( tre + 11*RAD/12 ) ) ) ) >> SGETA ;

        o_vda = ( KI * -ida ) >> KIGETA ;
        o_vqa = ( KI * ( o_iqa  - iqa )  ) >> KIGETA ;
/************************************************************************** /
/*      3相電圧変換処理                                                     */
/************************************************************************** /
        if ( init_flag == 2 ) {                                 /* CW初起動処理 */
            o_trm += ( SPEED_MINI * TS / 20000 ) ;
            tre = ( o_trm * P ) % RAD ;
            o_vda = 0 ;
            o_vqa = 100 ;
            if  ( o_trm > ( 2 * RAD ) ) {
                init_flag = 0;
            }
        } else if ( init_flag == 3 ) {                          /* CCW初起動処理 */
            o_trm -= ( SPEED_MINI * TS / 20000 ) ;
            tre = ( o_trm * P ) % RAD ;
            o_vda = 0 ;
            o_vqa = 100 ;
```

```
            if  ( o_trm < -( 2 * RAD ) ) {
                init_flag = 0;
            }
        } else {
            o_trm = 0 ;
        }
        o_vua = ( ( ( o_vda * sin( tre + RAD/4 ) ) - ( o_vqa * sin( tre ) ) ) ) >> SGETA ;
        o_vva = ( ( ( o_vda * sin( tre + 11*RAD/12 ) ) - ( o_vqa * sin( tre + 2*RAD/3 ) ) ) ) >> SGETA ;
        o_vwa  = -o_vua - o_vva ;

        cont_time1 = ( TM3 - s_time ) * 10 / 16; /* uSECに変換 */
/************************************************************************** /
/*      PWM変換出力処理                                                    */
/************************************************************************** /
        OUT_data( WRESET, 0 ) ;                 /* ウォッチドック・タイマRESET */
        POER0 = 0x3f ;                          /* 全相アクティブ */
        TUC00 = 0x02 ;                          /* PWM 出力禁止クリア */
/* PWMカウンタ値計算出力 */
        o_vua += ( PWM_DATA  / 2 ) ;
        o_vva += ( PWM_DATA  / 2 ) ;
        o_vwa += ( PWM_DATA  / 2 ) ;

        if  ( o_vua <= 0 ) {
            o_vua = 1 ;
        } else if  ( o_vua >= PWM_DATA  ) {
            o_vua = PWM_DATA  - 1 ;
        }
        if  ( o_vva <= 0 ) {
            o_vva = 1 ;
        } else if  ( o_vva >= PWM_DATA  ) {
            o_vva = PWM_DATA  - 1 ;
        }
        if  ( o_vwa <= 0 ) {
            o_vwa = 1 ;
        } else if  ( o_vwa >= PWM_DATA  ) {
            o_vwa = PWM_DATA  - 1 ;
        }

        BFCM00 = o_vua ;
        BFCM01 = o_vva ;
        BFCM02 = o_vwa ;
    } else {
        POER0 = 0 ;                             /* PWM出力OFF */
        now_speed = 0;
        cont_time1 = 0;
    }
}
```

4. 2. 8　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）                    */
/************************************************************************** /
_ _multi_interrupt
void    int_ETC(void)
{
unsigned short dummy ;
/* 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 += abs( now_speed ) ;
        disp_co -= 1 ;
    }
}
```

4. 2. 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                    */
/************************************************************************** /
_ _multi_interrupt
void    int_AD0(void)
{
    iua = (( ADCR00 & 0x3ff ) - 0x200) ;
    if  ( abs(iua) > MAX_I ) {
        POER0 = 0 ;                             /* PWM出力OFF */
        error_flag = ERR_NO1 ;                  /* エラーNOセット */
    }
    volume = 1023 - ( ADCR01 & 0x3ff ) ;        /* ボリューム値セット */
    Motor_CONT() ;
    cont_time = TM3 * 10 / 16;                  /* uSECに変換 */
}
/************************************************************************** /
/*      V相電流用A/Dコンバータ割り込み処理                                   */
/************************************************************************** /
_ _interrupt
void    int_AD1(void)
{
    iva = (( ADCR10 & 0x3ff ) - 0x200) ;
    if  ( abs(iva) > MAX_I ) {
        POER0 = 0 ;                             /* PWM出力OFF */
```

```
        error_flag = ERR_NO1 ;                  /* エラーNOセット */
    }
}
```

4. 2. 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 )
{
short   dummy ;
/*ポート・モード・レジスタ初期化 */
    PMC1 = 0x3f ;                               /* ENC select */
    PMC2 = 0x07 ;
    PM2 = 0x07 ;

    PM3 = 0xfe ;
    PM4 = 0xf7 ;
    PMDL = 0x0000 ;

    OUT_data( LED11, 0xff ) ;                   /* LED OFF */
    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 ) ;
/* 10 mSECタイマ　TM2設定 */
    STOPTE0 = 0x0000;
    PRM02 = 0x01;                               /* fXX/2セレクト */
    CSE0 = 0x0028;                              /* fCLK/64(3.2 uSEC)セレクト */
    TCRE0 = 0x2000;                             /* タイマ・スタート */
    CVSE50 = 1562*2 ;                           /* 10 mSEC */
    CMSE050 = 0x2400;
    CC2IC5 = 0x06;
/* モータ制御割り込み用タイマ　TM3設定 */
    PRM03 = 1 ;                                 /* fCLK = fXX */
    TMC30 = 0x51;                               /* fXX = 4MHz*10/64(1.6 uSEC) */
    TMC31 = 0x09 ;
    CC30  = TS * 10 / 16 ;                      /* TS uSEC インターバル */
    TMC30 = 0x53;                               /* タイマ・スタート */
    CC3IC0 = 0x02;                              /* 割り込みマスクをリセット */
/* TM10 初期化 */
    PRM02 = 1 ;                                 /* fCLK = fXX/2 */
    TUM0 = 0x80 ;                               /* UDC モード選択 */
    PRM10 = 0x07 ;                              /* 動作モード4選択 */
```

```
    TMC10 = 0x40 ;                          /* カウント開始 */
/* TM00 初期化 */
    PRM01 = 0 ;                             /* fCLK = fXX/2 */
    SPEC0 = 0x0000 ;
    TOMR0 = 0x80 ;                          /* 出力モード設定 */
    PSTO0 = 0x00 ;                          /* リアルタイム出力禁止 */
    BFCM00 = PWM_DATA /2 ;                  /* デューティ50設定 */
    BFCM01 = PWM_DATA /2 ;                  /* デューティ50設定 */
    BFCM02 = PWM_DATA /2 ;                  /* デューティ50設定 */
    BFCM03 = PWM_DATA ;                     /* PWM周期 設定 */
    DTRR0 = 40*3 ;                          /* デットタイム 6 uSEC */
    POER0 = 0x3f ;                          /* 全相アクティブ */
    TMC00 = 0x8018 ;                        /* TMPWM タイマ開始 */
/* A/D 設定 */
    ADSCM00 = 0x0001 ;
    ADSCM10 = 0x0000 ;
    ADIC0 = 0x03 ;
    ADIC1 = 0x03 ;
}
```

4. 2. 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 */
    o_trm = 0 ;
    before_enc  = -TM10 ;                   /* 前回エンコーダ値を設定 */
}
```

4. 2. 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;
/*  */
    for ( i = 0 ; i < 21 ; i++ ) before_posi[i][1] = 0;
```

```
    total_sa = 0 ;
    sum_speed = 0 ;
    speed_co = 100000 / TS ;
}
```

4. 2. 13　sin計算処理関数

```
/************************************************************************** /
/*      sin x                                                              */
/*          データ                                                         */
/*              ラジアン単位                                               */
/*          戻り値                                                         */
/*              サイン値*16384                                             */
/************************************************************************** /
int sin( int x )
{
    x = x % RAD ;
    if  ( x < 0 ) x += RAD ;
    if  ( x < (RAD/4) ){
        return sins( x ) ;
    } else if ( x < (RAD/2) ) {
        return sins( (RAD/2) - x ) ;
    } else if ( x < (RAD*3/4) ) {
        return -sins( x - (RAD/2) ) ;
    } else {
        return -sins( RAD - x ) ;
    }
}
/*************************************************************************/
int sins( int x )
{
short z1, z2, z3, z4, z5 ;
/*  */
    if  ( x <= (RAD/8) ) {
        z1 = (x << SGETA) / (RAD/8) ;
        z2 = z1 * z1 >> SGETA ;
        z3 = z1 * z2 >> SGETA ;
        z5 = z2 * z3 >> SGETA ;
        return ( (12868*z1) - (1322*z3) + (40*z5) ) >> SGETA ;
    } else {
        x = (RAD/4) - x ;
        z1 = (x << SGETA) / (RAD/8) ;
        z2 = z1 * z1 >> SGETA ;
        z4 = z2 * z2 >> SGETA ;
        return ( (268432772) - (5050*z2) + (252*z4) ) >> SGETA ;
    }
}
```

4. 2. 14　V850E/IA2用のリンク・ディレクティブ・ファイル

```
/************************************************************************** /
/*      V850E/IA2用のリンク・ディレクティブ・ファイル                        */
/************************************************************************** /
VECT_RESET: !LOAD ?RX V0x0000000 {
        .vect_RESET  = $PROGBITS ?AX .vect_RESET;
};
ID_NO: !LOAD ?RX V0x0000070 {
        .id_NO  = $PROGBITS ?AX .id_NO;
};
VECT_ETC: !LOAD ?RX V0x0000240 {
        .vect_ETC  = $PROGBITS ?AX .vect_ETC;
};
VECT_MOTOR: !LOAD ?RX V0x0000260 {
        .vect_MOTOR  = $PROGBITS ?AX .vect_MOTOR;
};
VECT_AD0: !LOAD ?RX V0x00003a0 {
        .vect_AD0  = $PROGBITS ?AX .vect_AD0;
};
VECT_AD1: !LOAD ?RX V0x00003b0 {
        .vect_AD1  = $PROGBITS ?AX .vect_AD1;
};

HANDLER: !LOAD ?RX V0x00001000 {
        .handler  = $PROGBITS ?AX .handler;
};
TEXT:  !LOAD ?RX {
        .text  = $PROGBITS ?AX .text;
};

CONST : !LOAD ?R {
        .const = $PROGBITS ?A .const;
};

DATA : !LOAD ?RW V0x0fffc000 {
        .data  = $PROGBITS ?AW  ;
        .sdata = $PROGBITS ?AWG ;
        .sbss  = $NOBITS   ?AWG ;
        .bss   = $NOBITS   ?AW  ;
};

_ _tp_TEXT @ %TP_SYMBOL;
_ _gp_DATA @ %GP_SYMBOL &_ _tp_TEXT{DATA};
_ _ep_DATA @ %EP_SYMBOL;
```

4. 3　プログラム・リスト（V850E/IA3用）

4. 3. 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 short    before_posi[21][2] ;        /* 位置バッファ */
signed short    total_sa ;                  /* 位置トータル差分 */
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 ;

#pragma section const begin
const unsigned short led_pat[10] = { 0xfc, 0x60, 0xda, 0xf2, 0x66, 0xb6, 0xbe, 0xe0, 0xfe,
                                     0xe6 } ;
#pragma section const end
/************************************************************************** /
/*      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 short     before_posi[21][2] ;/* 位置バッファ */
extern signed short     total_sa ;          /* 位置トータル差分 */
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 ;

#pragma section const begin
extern const unsigned short led_pat[] ;;
#pragma section const end
/************************************************************************** /
/*      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 int       o_iqai ;                /* 速度積分値エリア */
extern signed int       o_trm ;                 /* イニシャル用目標位置 */
extern signed int       base_position ;         /* 速度推定値基準点 */
extern unsigned int     sa_time ;               /* 速度計測値 */
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 signed short     before_enc ;            /* 前回エンコーダ値 */
```

4.3.2 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                                                          */
/************************************************************************** /
#define PAI             3.14159265             /* π */
#define TH_U            1000                   /* ラジアン値　ゲタ定数 */
#define RAD             (int)(2*PAI*TH_U)      /* 1回転のラジアン値 */
#define OFFSET          1733                   /* 原点OFFSET */
#define RPM_RADS        (int)((2*PAI*TH_U)/60) /* rpm->ラジアン変換定数 */
/* モータ定数 */
#define KSP             500                    /* 速度比例定数 */
#define KSI             50                     /* 速度積分定数 */
#define KI              1                      /* 電流変換定数 */
#define P               2                      /* 極数 */

#define KSPGETA         0                      /* KSP  ゲタ定数 */
#define KSIGETA         9                      /* KSI ゲタ定数 */
#define KIGETA          9                      /* KI  ゲタ定数 */
#define SGETA           14                     /* sin ゲタ定数 */

#define PWM_TS          80                     /* PWM 周期 */
#define PWM_DATA        (PWM_TS/0.03125)       /* PWM 設定値 */
#define SPEED_MAX       3000                   /* 最大速度 rpm */
#define SPEED_MINI      600                    /* 最低速度 rpm */
#define SA_SPEED_MAX    800                    /* 最大速度差分 rpm */
#define IQAMAX          40000000               /* 最大速度積分値 */
#define MAX_I           800                    /* 電流 MAX値 */
#define TS              80                     /* モータ制御時間間隔  uSEC */
#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 MAXPULSE        10000                  /* 1回転のパルス数 */
/************************************************************************** /
/*      関数定義                                                            */
/************************************************************************** /
void            fcalcu( signed int *wrm, signed int *trm );
void            OUT_data( unsigned short reg, unsigned short data );
unsigned short  IN_data( int reg );
void            led_num( int no, long data ) ;
/************************************************************************** /
/*      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 int      o_iqai ;                       /* 速度積分値エリア */
signed int      o_trm ;                        /* イニシャル用目標位置 */
signed int      base_position ;                /* 速度推定値基準点 */
```

```
unsigned int    sa_time ;                       /* 速度計測値 */
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 */
signed short    before_enc ;                    /* 前回エンコーダ値 */
```

4.3.3　割り込みハンドラ設定

```
/************************************************************************** /
/*      割り込みシンボル・テーブル                                          */
/************************************************************************** /
        .extern _ _start
        .extern _int_MOTOR
        .extern _int_AD0
        .extern _int_AD1
        .extern _int_ETC

        .globl V_RESET
        .globl V_ETC
        .globl V_MOTOR
        .globl V_AD0
        .globl V_AD1
#************************************************************************
        .section ".handler",text
V_RESET:
        jr      _ _start
V_ETC:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_ETC                        -- その他タイマ
V_MOTOR:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_MOTOR                      -- 速度制御タイマ
V_AD0:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_AD0                        -- A/Dコンバータ CH0
V_AD1:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_AD1                        -- A/Dコンバータ CH1

        .extern V_RESET
        .extern V_ETC
        .extern V_MOTOR
        .extern V_AD0
        .extern V_AD1
```

```
/************************************************************************** /
/*      割り込みジャンプ・テーブル                                          */
/************************************************************************** /
        .section ".vect_RESET",text
        mov     #V_RESET,r1
        jmp     [r1]

        .section ".id_NO",text
        .byte   0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff

        .section ".vect_ETC",text
        add     -4,sp
        st.w    r1,[sp]
        mov     #V_ETC,r1
        jmp     [r1]

        .section ".vect_MOTOR",text
        add     -4,sp
        st.w    r1,[sp]
        mov     #V_MOTOR,r1
        jmp     [r1]

        .section ".vect_AD0",text
        add     -4,sp
        st.w    r1,[sp]
        mov     #V_AD0,r1
        jmp     [r1]

        .section ".vect_AD1",text
        add     -4,sp
        st.w    r1,[sp]
        mov     #V_AD1,r1
        jmp     [r1]
```

4.3.4　スタートアップ・ルーチン設定

```
#===============================================================================
# DESCRIPTIONS:
#     This assembly program is a sample of start-up module for ca850.
#   If you modified this program, you must assemble this file, and
#   locate a given directory.
#
#   Unless -G is specified, sections are located as the following.
#
#                  |       :       |
#                  |       :       |
#          tp -> -+--------------- + _ _start    _ _tp_TEXT
#                  | start up      |
#                  |-------------- |
# text section     |               |
```

```
#                  | user program  |
#                  |               |
#                  |-------------- |
#                  | library       |
#                 -+-------------- +
#                  |               |
# sdata section    |               |
#                  |               |
#          gp -> -+-------------- +            _ _ssbss
#                  |               |
# sbss section     |               |
#                  |               |
#                   +-------------- + _ _stack    _ _esbss        _ _sbss
#                  | stack area    |
# bss section      |               |
#                  | 0x400 bytes   |
#          sp -> -+-------------- + _ _stack + STACKSIZE       _ _ebss
#                  |       :       |
#                  |       :       |
#                  |       :       |
#           ep -> -+-------------- + _ _ep_DATA
# tidata section   |               |
#                 -+-------------- +
# sidata section   |               |
#                 -+-------------- +
#                  |       :       |
#                  |       :       |
#
#==============================================================================
#------------------------------------------------------------------------------
#       special symbols
#------------------------------------------------------------------------------
        .extern _ _tp_TEXT, 4
        .extern _ _gp_DATA, 4
        .extern _ _ep_DATA, 4
        .extern _ _ssbss, 4
        .extern _ _esbss, 4
        .extern _ _sbss, 4
        .extern _ _ebss, 4


#------------------------------------------------------------------------------
#       C program main function
#------------------------------------------------------------------------------
        .extern _main


#------------------------------------------------------------------------------
#       dummy data declaration for creating sbss section
#------------------------------------------------------------------------------
```

```
        .sbss
        .lcomm  _ _sbss_dummy, 0, 0


#------------------------------------------------------------------------------
#       system stack
#------------------------------------------------------------------------------
        .set    STACKSIZE, 0x400
        .bss
        .lcomm  _ _stack, STACKSIZE, 4

#------------------------------------------------------------------------------
#       start up
#            pointers:  tp - text pointer
#                       gp - global pointer
#                       sp - stack pointer
#                       ep - element pointer
#       exit status is set to r10
#------------------------------------------------------------------------------
        .text
        .align 4
        .globl _ _start
        .globl _exit
        .globl _ _exit
_ _start:
        mov     13,r10                          -- 500 kHz～64 MHz
        st.b    r10,VSWC[r0]                    -- 周辺I/Oウエイト設定

        mov     0x03,r10                        -- PLL
        st.b    r10,PLLCTL[r0]

        mov     0x00,r10
        st.b    r10,PRCMD[r0]
        st.b    r10,PCC[r0]
        nop
        nop
        nop
        nop
        nop

        mov     #_ _tp_TEXT, tp                 -- set tp register
        mov     #_ _gp_DATA, gp                 -- set gp register offset
        add     tp, gp                          -- set gp register
        mov     #_ _stack+STACKSIZE, sp         -- set sp register
        mov     #_ _ep_DATA, ep                 -- set ep register
#
        mov     #_ _ssbss, r13                  -- clear sbss section
        mov     #_ _esbss, r12
        cmp     r12, r13
        jnl     .L11
```

```
.L12:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L12
.L11:
#
        mov     #_ _sbss, r13                   -- clear bss section
        mov     #_ _ebss, r12
        cmp     r12, r13
        jnl     .L14
.L15:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L15
.L14:
#
        jarl    _main, lp                       -- call main function
_ _exit:
        halt                                    -- end of program
_ _startend:
        nop
#                                                                               #
#------------------------- end of start up module ----------------------------#
#                                                                               #
```

4.3.5　メイン処理関数

```
#include    "Common.h"
#include    "Motor.h"
#pragma     ioreg                               /* 周辺I/Oレジスタ定義 */

static int save_psw;
/*****************************************************************************/
/*      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 ) * 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 ;
        }
        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 / 100 );             /* 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 );                 /* 処理時間全体 */
        led_num(4, cont_time1 );                /* ベクトル演算処理時間 */
    }
    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 ) ;
        }
    }
}
```

4.3.6　LED表示関数

```
/*************************************************************************** /
/*      LED数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 ) {
                P3.0 = 0;
                data = 1;                       /* ダミー・ステップ */
                P3.0 = 1;
        } else {
                PDL = data | ( reg << 8 );
```

```
                PDL = reg | ( reg << 8 ) | 0x8000;
        }
}
/************************************************************************** /
/*      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 P4;
        } else {
                return 0;
        }
}
```

4.3.7　モータ制御割り込み処理関数

```
#include    "Common.h"
#include    "Motor.h"
#pragma     ioreg                                   /* 周辺I/Oレジスタ定義 */
/************************************************************************** /
/*      モータ制御タイマ割り込み処理                                        */
/************************************************************************** /
_ _interrupt
void    int_MOTOR(void)
{
   ADA0M0 = 0xB0 ;                                  /* AD0 起動 */
   ADA1M0 = 0xA0 ;                                  /* AD1 起動 */
}
/************************************************************************** /
/*      モータ制御処理                                                      */
/************************************************************************** /
void    Motor_CONT(void)
{
signed int      wrm, wre, trm, tre ;
signed int      o_wre, we, o_iqap, o_iqa ;
signed int      ida, iqa, o_vda1, o_vqa1 ;
signed int      o_vd0, o_vq0, o_vda, o_vqa ;
signed int      o_vua, o_vva, o_vwa, wiua, wiva ;
signed int      s_time, wk ;
signed short    now_enc, sa_enc ;
/*  */
/************************************************************************** /
/*      速度およびロータ位置の計算処理                                      */
/************************************************************************** /
   now_enc = -TMENC10 ;
   sa_enc = now_enc - before_enc ;
```

```
   if  ( abs( sa_enc ) > ( MAXPULSE / 2 ) ) {
      if  ( sa_enc < 0 ) {
         sa_enc += MAXPULSE ;
      } else {
         sa_enc -= MAXPULSE ;
      }
   }
   before_enc = now_enc ;
   wrm = now_speed = (signed long)sa_enc * RAD / MAXPULSE;     /* 現在速度セット */

   wre = wrm * P ;
   tre = ( trm * P + OFFSET ) % RAD ;
/************************************************************************** /
/*      速度制御処理                                                        */
/************************************************************************** /
   if  ( ( stop_flag == OFF ) && ( error_flag == 0 ) ) {
      s_time = TP1CNT ;
      o_wre = object_speed * RPM_RADS * P / TH_U ;            /* rpm->ラジアン変換 */
      we =  o_wre - wre ;
      o_iqap  = ( ( wre * KSP ) + ( we * KSP ) ) >> KSPGETA ;
      o_iqa   = o_iqap + ( o_iqai >> KSIGETA ) ;

      if  ( o_iqai > IQAMAX ) {
         o_iqai = IQAMAX ;
      } else if ( o_iqai < -IQAMAX ) {
         o_iqai = -IQAMAX ;
      } else {
         o_iqai += ( KSI * we ) ;
      }
/************************************************************************** /
/*      電流制御処理                                                        */
/************************************************************************** /
      ida = ( ( ( iva * sin( tre ) ) - ( iua * sin( tre + 2*RAD/3 ) ) ) ) >> SGETA ;
      iqa = ( ( ( iva * sin( tre + RAD/4 ) ) - ( iua * sin( tre + 11*RAD/12 ) ) ) ) >> SGETA ;

      o_vda = ( KI * -ida ) >> KIGETA ;
      o_vqa = ( KI * ( o_iqa  - iqa )  ) >> KIGETA ;
/************************************************************************** /
/*      3相電圧変換処理                                                     */
/************************************************************************** /
      if ( init_flag == 2 ) {                                 /* CW初起動処理 */
         o_trm += ( SPEED_MINI * TS / 20000 ) ;
         tre = ( o_trm * P ) % RAD ;
         o_vda = 0 ;
         o_vqa = 100 ;
         if  ( o_trm > ( 2 * RAD ) ) {
            init_flag = 0;
         }
      } else if ( init_flag == 3 ) {                          /* CCW初起動処理 */
         o_trm -= ( SPEED_MINI * TS / 20000 ) ;
```

```
            tre = ( o_trm * P ) % RAD ;
            o_vda = 0 ;
            o_vqa = 100 ;
            if  ( o_trm < -( 2 * RAD ) ) {
                init_flag = 0;
            }
        } else {
            o_trm = 0 ;
        }
        o_vua = ( ( ( o_vda * sin( tre + RAD/4 ) ) - ( o_vqa * sin( tre ) ) ) ) >> SGETA ;
        o_vva = ( ( ( o_vda * sin( tre + 11*RAD/12 ) ) - ( o_vqa * sin( tre + 2*RAD/3 ) ) ) ) >> SGETA ;
        o_vwa  = -o_vua - o_vva ;

        cont_time1 = ( TP1CNT - s_time ) ;       /* uSECに変換 */
/************************************************************************** /
/*      PWM変換出力処理                                                    */
/************************************************************************** /
        OUT_data( WRESET, 0 ) ;                  /* ウォッチドック・タイマRESET */
        HZA0CTL0 |= 0x04;                        /* PWM出力ON */
/* PWMカウンタ値計算出力 */
        o_vua += ( PWM_DATA  / 2 ) ;
        o_vva += ( PWM_DATA  / 2 ) ;
        o_vwa += ( PWM_DATA  / 2 ) ;

        if  ( o_vua <= 0 ) {
            o_vua = 1 ;
        } else if  ( o_vua >= PWM_DATA  ) {
            o_vua = PWM_DATA  - 1 ;
        }
        if  ( o_vva <= 0 ) {
            o_vva = 1 ;
        } else if  ( o_vva >= PWM_DATA  ) {
            o_vva = PWM_DATA  - 1 ;
        }
        if  ( o_vwa <= 0 ) {
            o_vwa = 1 ;
        } else if  ( o_vwa >= PWM_DATA  ) {
            o_vwa = PWM_DATA  - 1 ;
        }

        TQ0CCR1 = o_vua ;
        TQ0CCR2 = o_vva ;
        TQ0CCR3 = o_vwa ;
    } else {
        HZA0CTL0 |= 0x08;                        /* PWM出力OFF */
        now_speed = 0;
        cont_time1 = 0;
    }
}
```

4. 3. 8　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）                    */
/************************************************************************** /
_ _multi_interrupt
void    int_ETC(void)
{
/* 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 += abs( now_speed ) ;
        disp_co -= 1 ;
    }
}
```

4. 3. 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                   */
/************************************************************************** /
_ _multi_interrupt
void    int_AD0(void)
{
    iua = (( ( ADA0CR0 >> 6 ) & 0x3ff ) - 0x200) ;
    if  ( abs(iua) > MAX_I ) {
        HZA0CTL0 |= 0x08 ;                          /* PWM出力OFF */
        error_flag = ERR_NO1 ;                      /* エラーNOセット */
    }
    volume = 1023 - ( ( ADA0CR1 >> 6 ) & 0x3ff ) ;      /* ボリューム値セット */
    Motor_CONT() ;
    cont_time = TP1CNT;                             /* uSECに変換 */
}
/************************************************************************** /
/*      ボリューム用A/Dコンバータ割り込み処理                                */
/************************************************************************** /
_ _interrupt
void    int_AD1(void)
{
    iva = (( ADA1CR0 & 0x3ff ) - 0x200) ;
    if  ( abs(iva) > MAX_I ) {
        HZA0CTL0 |= 0x08 ;                          /* PWM出力OFF */
        error_flag = ERR_NO1 ;                      /* エラーNOセット */
```

```
    }
}
```

4. 3. 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 )
{
/* ポート・モード・レジスタ初期化 */
    PM3 = 0xfe ;
    PM4 = 0xff ;
    PMDL = 0x0000 ;

    OUT_data( LED11, 0xff ) ;                   /* LED OFF */
    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 ) ;
/* 10 mSECタイマ  TMP0設定 */
    TP0CTL0 = 0x05 ;                            /* fXX/64セレクト */
    TP0CTL1 = 0x00 ;                            /* インターバル・タイマ・モード・セレクト */
    TP0CCR0 = 10000 ;                           /* 10 mSEC */
    TP0CTL0 |= 0x80 ;                           /* タイマ・スタート */
    TP0CCIC0= 0x06;
/* モータ制御割り込み用タイマ  TMP1設定 */
    TP1CTL0 = 0x05 ;                            /* fXX/64セレクト */
    TP1CTL1 = 0x00 ;                            /* インターバル・タイマ・モード・セレクト */
    TP1CCR0 = TS ;                              /* TS uSEC */
    TP1CTL0 |= 0x80 ;                           /* タイマ・スタート */
    TP1CCIC0= 0x01;
/* TMENC10 初期化 */
    TUM10 = 0x80 ;                              /* UDC モード選択 */
    PRM10 = 0x07 ;                              /* 動作モード4選択 */
    TMC10 = 0x40 ;                              /* カウント開始 */
/* TMQ0 初期化 */
    TQ0CTL0 = 0x00;                             /* fXX/2  (64 MHz/2 = 32 MHz) */
    TQ0CTL1 = 0x07;                             /* 6相PWM出力モード・セレクト */
    TQ0IOC0 = 0x55;                             /* 正相 通常出力,出力許可 */
    TQ0IOC1 = 0x00;                             /* TMQ0のTIQ00-TIQ03,EVTQ0,TRGQ0端子 */
    TQ0IOC2 = 0x00;                             /* は使用しない */
    TQ0OPT0 = 0x00;                             /* コンペア・モード・セレクト */
    TQ0CCR0 = PWM_DATA ;                        /* 搬送波周期 20 kHz */
    TQ0CCR1 = PWM_DATA /2;                      /* U相　デューディ50設定 */
```

```
    TQ0CCR2 = PWM_DATA /2;                  /* V相　デューディ50設定 */
    TQ0CCR3 = PWM_DATA /2;                  /* W相　デューディ50設定 */
    TQ0DTC = 180;                           /* デットタイム　6 uSEC */
    TQ0OPT1 = 0x00;                         /* 間引きなし,山／谷割り込み */
                                            /* を使用しない */

    TQ0OPT2 = 0x04;                         /* ・リロード間の間引きなし */
                                            /* ・デット・タイム・カウンタは */
                                            /*　クリア再カウント */
                                            /* ・INTTP0CC0割り込みのA/Dトリガ*/
                                            /*　出力をアップ・カウント時に出力 */
                                            /* ・INTTP0CC0割り込みのA/Dトリガ*/
                                            /*　出力許可 */

    TQ0IOC3 = 0xfc;                         /* 逆相 反転出力,出力許可 */
    PMC1 = 0x3F;                            /* 兼用機能モード */
    PFCE1 = 0x00;                           /* TOQ0T1-TOQ0T3,TOQ0B1-TOQ0B3出力選択 */
    PFC1 = 0x00;

    HZA0CTL0 = 0x00;
    HZA0CTL0 = 0xD0;                        /* ハイ・インピーダンス動作許可 */
                                            /* TOQ0OFF端子立ち上がりエッジ有効 */
    HZA2CTL0 = 0x00;                        /* アナログ入力によるハイ・インピーダンスOFF */
    TQ0CTL0 |= 0x80;                        /* 6相PWM出力モード・スタート */
/* A/D 設定 */
    ADA0M0 = 0x30 ;                         /* ANI00,ANI01 */
    ADA0M1 = 0x01 ;
    ADA0S = 0x01 ;
    OP0CTL0 = 0x00 ;                        /* オペアンプ無効 */
    OP0CTL1 = 0x00 ;                        /* コンパレータ無効 */
    AD0IC = 0x03 ;

    ADA1M0 = 0x20 ;                         /* ANI10 */
    ADA1M1 = 0x01 ;
    ADA1S = 0x00 ;
    OP1CTL0 = 0x00 ;                        /* オペアンプ無効 */
    OP1CTL1 = 0x00 ;                        /* コンパレータ無効 */
    AD1IC = 0x03 ;
    PIC4 = 0x01 ;
}
```

4. 3. 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 */
    o_trm = 0 ;
    before_enc  = -TMENC10 ;                    /* 前回エンコーダ値を設定 */
}
```

4. 3. 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;
/*  */
    for ( i = 0 ; i < 21 ; i++ ) before_posi[i][1] = 0;
    total_sa = 0 ;
    sum_speed = 0 ;
    speed_co = 100000 / TS ;
}
```

4. 3. 13　sin計算処理関数

```
/************************************************************************** /
/*      sin x                                                               */
/*          データ                                                          */
/*              ラジアン単位                                                */
/*          戻り値                                                          */
/*              サイン値*16384                                              */
/************************************************************************** /
int sin( int x )
{
    x = x % RAD ;
    if  ( x < 0 ) x += RAD ;
    if  ( x < (RAD/4) ){
        return sins( x ) ;
    } else if ( x < (RAD/2) ) {
        return sins( (RAD/2) - x ) ;
    } else if ( x < (RAD*3/4) ) {
        return -sins( x - (RAD/2) ) ;
    } else {
        return -sins( RAD - x ) ;
    }
}
```

```
/*****************************************************************************/
int sins( int x )
{
short z1, z2, z3, z4, z5 ;
/*  */
   if  ( x <= (RAD/8) ) {
       z1 = (x << SGETA) / (RAD/8) ;
       z2 = z1 * z1 >> SGETA ;
       z3 = z1 * z2 >> SGETA ;
       z5 = z2 * z3 >> SGETA ;
       return ( (12868*z1) - (1322*z3) + (40*z5) ) >> SGETA ;
   } else {
       x = (RAD/4) - x ;
       z1 = (x << SGETA) / (RAD/8) ;
       z2 = z1 * z1 >> SGETA ;
       z4 = z2 * z2 >> SGETA ;
       return ( (268432772) - (5050*z2) + (252*z4) ) >> SGETA ;
   }
}
```

4. 3. 14　V850E/IA3用のリンク・ディレクティブ・ファイル

```
/*****************************************************************************/
/*      V850E/IA3用のリンク・ディレクティブ・ファイル                        */
/*****************************************************************************/
VECT_RESET: !LOAD ?RX V0x00000000 {
        .vect_RESET  = $PROGBITS ?AX .vect_RESET;
};
ID_NO: !LOAD ?RX V0x00000070 {
        .id_NO  = $PROGBITS ?AX .id_NO;
};
VECT_ETC: !LOAD ?RX V0x00000250 {
        .vect_ETC  = $PROGBITS ?AX .vect_ETC;
};
VECT_MOTOR: !LOAD ?RX V0x00000280 {
        .vect_MOTOR  = $PROGBITS ?AX .vect_MOTOR;
};
VECT_AD0: !LOAD ?RX V0x00000400 {
        .vect_AD0  = $PROGBITS ?AX .vect_AD0;
};
VECT_AD1: !LOAD ?RX V0x00000410 {
        .vect_AD1  = $PROGBITS ?AX .vect_AD1;
};

HANDLER: !LOAD ?RX V0x00001000 {
        .handler  = $PROGBITS ?AX .handler;
};
TEXT:  !LOAD ?RX {
        .text  = $PROGBITS ?AX .text;
};
```

```
CONST : !LOAD ?R {
        .const = $PROGBITS ?A .const;
};

DATA : !LOAD ?RW  V0xffffd800 {
        .data  = $PROGBITS ?AW  ;
        .sdata = $PROGBITS ?AWG ;
        .sbss  = $NOBITS   ?AWG ;
        .bss   = $NOBITS   ?AW  ;
};

_ _tp_TEXT @ %TP_SYMBOL;
_ _gp_DATA @ %GP_SYMBOL &_ _tp_TEXT{DATA};
_ _ep_DATA @ %EP_SYMBOL;
```

4. 4　プログラム・リスト（V850E/IA4用）

4. 4. 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 short    before_posi[21][2] ;        /* 位置バッファ */
signed short    total_sa ;                  /* 位置トータル差分 */
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 ;

#pragma section const begin
const unsigned short led_pat[10] = { 0xfc, 0x60, 0xda, 0xf2, 0x66, 0xb6, 0xbe, 0xe0, 0xfe,
                                     0xe6 } ;
#pragma section const end
/************************************************************************** /
/*      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 short     before_posi[21][2] ;/* 位置バッファ */
extern signed short     total_sa ;          /* 位置トータル差分 */
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 ;

#pragma section const begin
extern  const unsigned short led_pat[] ;;
#pragma section const end
/************************************************************************** /
/*      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 int       o_iqai ;                /* 速度積分値エリア */
extern signed int       o_trm ;                 /* イニシャル用目標位置 */
extern signed int       base_position ;         /* 速度推定値基準点 */
extern unsigned int     sa_time ;               /* 速度計測値 */
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 signed short     before_enc ;            /* 前回エンコーダ値 */
```

4.4.2 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                                                          */
/************************************************************************** /
#define PAI             3.14159265             /* π */
#define TH_U            1000                   /* ラジアン値　ゲタ定数 */
#define RAD             (int)(2*PAI*TH_U)      /* 1回転のラジアン値 */
#define OFFSET          1733                   /* 原点OFFSET */
#define RPM_RADS        (int)((2*PAI*TH_U)/60) /* rpm->ラジアン変換定数 */
/* モータ定数 */
#define KSP             500                    /* 速度比例定数 */
#define KSI             50                     /* 速度積分定数 */
#define KI              1                      /* 電流変換定数 */
#define P               2                      /* 極数 */

#define KSPGETA         0                      /* KSP  ゲタ定数 */
#define KSIGETA         9                      /* KSI ゲタ定数 */
#define KIGETA          9                      /* KI  ゲタ定数 */
#define SGETA           14                     /* sin ゲタ定数 */

#define PWM_TS          80                     /* PWM 周期 */
#define PWM_DATA        (PWM_TS/0.03125)       /* PWM 設定値 */
#define SPEED_MAX       3000                   /* 最大速度 rpm */
#define SPEED_MINI      600                    /* 最低速度 rpm */
#define SA_SPEED_MAX    800                    /* 最大速度差分 rpm */
#define IQAMAX          40000000               /* 最大速度積分値 */
#define MAX_I           800                    /* 電流 MAX値 */
#define TS              80                     /* モータ制御時間間隔  uSEC */
#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 MAXPULSE        10000                  /* 1回転のパルス数 */
/************************************************************************** /
/*      関数定数                                                            */
/************************************************************************** /
void            fcalcu( signed int *wrm, signed int *trm );
void            OUT_data( unsigned short reg, unsigned short data );
unsigned short  IN_data( int reg );
void            led_num( int no, long data ) ;
/************************************************************************** /
/*      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 int      o_iqai ;                       /* 速度積分値エリア */
signed int      o_trm ;                        /* イニシャル用目標位置 */
signed int      base_position ;                /* 速度推定値基準点 */
```

```
unsigned int    sa_time ;                       /* 速度計測値 */
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 */
signed short    before_enc ;                    /* 前回エンコーダ値 */
```

4.4.3　割り込みハンドラ設定

```
/************************************************************************** /
/*      割り込みシンボル・テーブル                                          */
/************************************************************************** /
        .extern _ _start
        .extern _int_MOTOR
        .extern _int_AD0
        .extern _int_AD1
        .extern _int_ETC

        .globl V_RESET
        .globl V_ETC
        .globl V_MOTOR
        .globl V_AD0
        .globl V_AD1
#************************************************************************
        .section ".handler",text
V_RESET:
        Jr      _ _start
V_ETC:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_ETC                        -- その他タイマ
V_MOTOR:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_MOTOR                      -- 速度制御タイマ
V_AD0:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_AD0                        -- A/Dコンバータ CH0
V_AD1:
        ld.w    [sp],r1
        add     4,sp
        jr      _int_AD1                        -- A/Dコンバータ CH1

        .extern V_RESET
        .extern V_ETC
        .extern V_MOTOR
        .extern V_AD0
        .extern V_AD1
```

```
/************************************************************************** /
/*      割り込みジャンプ・テーブル                                          */
/************************************************************************** /
        .section ".vect_RESET",text
        mov     #V_RESET,r1
        jmp     [r1]

        .section ".id_NO",text
        .byte   0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff

        .section ".vect_ETC",text
        add     -4,sp
        st.w    r1,[sp]
        mov     #V_ETC,r1
        jmp     [r1]

        .section ".vect_MOTOR",text
        add     -4,sp
        st.w    r1,[sp]
        mov     #V_MOTOR,r1
        jmp     [r1]

        .section ".vect_AD0",text
        add     -4,sp
        st.w    r1,[sp]
        mov     #V_AD0,r1
        jmp     [r1]

        .section ".vect_AD1",text
        add     -4,sp
        st.w    r1,[sp]
        mov     #V_AD1,r1
        jmp     [r1]
```

4.4.4　スタートアップ・ルーチン設定

```
#==============================================================================
# DESCRIPTIONS:
#     This assembly program is a sample of start-up module for ca850.
#   If you modified this program, you must assemble this file, and
#   locate a given directory.
#
#   Unless -G is specified, sections are located as the following.
#
#                 |       :       |
#                 |       :       |
#          tp -> -+--------------- + _ _start    _ _tp_TEXT
#                 | start up      |
#                 |-------------- |
# text section    |               |
```

```
#                  | user program  |
#                  |               |
#                  |-------------- |
#                  | library       |
#                 -+-------------- +
#                  |               |
# sdata section    |               |
#                  |               |
#           gp -> -+-------------- +            _ _ssbss
#                  |               |
# sbss section     |               |
#                  |               |
#                   +-------------- + _ _stack    _ _esbss        _ _sbss
#                  | stack area    |
# bss section      |               |
#                  | 0x400 bytes   |
#           sp -> -+-------------- + _ _stack + STACKSIZE       _ _ebss
#                  |       :       |
#                  |       :       |
#                  |       :       |
#           ep -> -+-------------- + _ _ep_DATA
# tidata section   |               |
#                 -+-------------- +
# sidata section   |               |
#                 -+-------------- +
#                  |       :       |
#                  |       :       |
#
#==============================================================================
#------------------------------------------------------------------------------
#       special symbols
#------------------------------------------------------------------------------
        .extern _ _tp_TEXT, 4
        .extern _ _gp_DATA, 4
        .extern _ _ep_DATA, 4
        .extern _ _ssbss, 4
        .extern _ _esbss, 4
        .extern _ _sbss, 4
        .extern _ _ebss, 4


#------------------------------------------------------------------------------
#       C program main function
#------------------------------------------------------------------------------
        .extern _main


#------------------------------------------------------------------------------
#       dummy data declaration for creating sbss section
#------------------------------------------------------------------------------
```

```
        .sbss
        .lcomm  _ _sbss_dummy, 0, 0


#------------------------------------------------------------------------------
#       system stack
#------------------------------------------------------------------------------
        .set    STACKSIZE, 0x400
        .bss
        .lcomm  _ _stack, STACKSIZE, 4

#------------------------------------------------------------------------------
#       start up
#            pointers:  tp - text pointer
#                       gp - global pointer
#                       sp - stack pointer
#                       ep - element pointer
#       exit status is set to r10
#------------------------------------------------------------------------------
        .text
        .align 4
        .globl _ _start
        .globl _exit
        .globl _ _exit
_ _start:
        mov     13,r10                          -- 500 kHz～64 MHz
        st.b    r10,VSWC[r0]                    -- 周辺I/Oウエイト設定

        mov     0x03,r10                        -- PLL
        st.b    r10,PLLCTL[r0]

        mov     0x00,r10
        st.b    r10,PRCMD[r0]
        st.b    r10,PCC[r0]
        nop
        nop
        nop
        nop
        nop

        mov     #_ _tp_TEXT, tp                 -- set tp register
        mov     #_ _gp_DATA, gp                 -- set gp register offset
        add     tp, gp                          -- set gp register
        mov     #_ _stack+STACKSIZE, sp         -- set sp register
        mov     #_ _ep_DATA, ep                 -- set ep register
#
        mov     #_ _ssbss, r13                  -- clear sbss section
        mov     #_ _esbss, r12
        cmp     r12, r13
        jnl     .L11
```

```
.L12:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L12
.L11:
#
        mov     #_ _sbss, r13                   -- clear bss section
        mov     #_ _ebss, r12
        cmp     r12, r13
        jnl     .L14
.L15:
        st.w    r0, [r13]
        add     4, r13
        cmp     r12, r13
        jl      .L15
.L14:
#
        jarl    _main, lp                       -- call main function
_ _exit:
        halt                                    -- end of program
_ _startend:
        nop
#                                                                               #
#------------------------- end of start up module ----------------------------#
#                                                                               #
```

4.4.5　メイン処理関数

```
#include    "Common.h"
#include    "Motor.h"
#pragma     ioreg                               /* 周辺I/Oレジスタ定義 */

static int save_psw;
/*****************************************************************************/
/*      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 ) * 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 ;
        }
        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 / 100 );             /* 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 );                 /* 処理時間全体 */
        led_num(4, cont_time1 );                /* ベクトル演算処理時間 */
    }
    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 ) ;
        }
    }
}
```

4.4.6　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 ) {
                P3.0 = 0;
                data = 1;                       /* ダミー・ステップ */
                P3.0 = 1;
        } else {
                PDL = data | ( reg << 8 );
```

```
                PDL = reg | ( reg << 8 ) | 0x8000;
        }
}
/************************************************************************** /
/*      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 P4;
        } else {
                return 0;
        }
}
```

4.4.7　モータ制御割り込み処理関数

```
#include    "Common.h"
#include    "Motor.h"
#pragma     ioreg                                   /* 周辺I/Oレジスタ定義 */
/************************************************************************** /
/*      モータ制御タイマ割り込み処理                                        */
/************************************************************************** /
_ _interrupt
void    int_MOTOR(void)
{
   ADA0M0 = 0xB0 ;                                  /* AD0 起動 */
   ADA1M0 = 0xA0 ;                                  /* AD1 起動 */
}
/************************************************************************** /
/*      モータ制御処理                                                      */
/************************************************************************** /
void    Motor_CONT(void)
{
signed int      wrm, wre, trm, tre ;
signed int      o_wre, we, o_iqap, o_iqa ;
signed int      ida, iqa, o_vda1, o_vqa1 ;
signed int      o_vd0, o_vq0, o_vda, o_vqa ;
signed int      o_vua, o_vva, o_vwa, wiua, wiva ;
signed int      s_time, wk ;
signed short    now_enc, sa_enc ;
/*  */
/************************************************************************** /
/*      速度およびロータ位置の計算処理                                      */
/************************************************************************** /
   now_enc = -TMENC10 ;
   sa_enc = now_enc - before_enc ;
```

```
   if  ( abs( sa_enc ) > ( MAXPULSE / 2 ) ) {
      if  ( sa_enc < 0 ) {
         sa_enc += MAXPULSE ;
      } else {
         sa_enc -= MAXPULSE ;
      }
   }
   before_enc = now_enc ;
   wrm = now_speed = (signed long)sa_enc * RAD / MAXPULSE;     /* 現在速度セット */

   wre = wrm * P ;
   tre = ( trm * P + OFFSET ) % RAD ;
/************************************************************************** /
/*      速度制御処理                                                        */
/************************************************************************** /
   if  ( ( stop_flag == OFF ) && ( error_flag == 0 ) ) {
      s_time = TP1CNT ;
      o_wre = object_speed * RPM_RADS * P / TH_U ;            /* rpm->ラジアン変換 */
      we     =  o_wre - wre ;
      o_iqap  = ( ( wre * KSP ) + ( we * KSP ) ) >> KSPGETA ;
      o_iqa   = o_iqap + ( o_iqai >> KSIGETA ) ;

      if  ( o_iqai > IQAMAX ) {
         o_iqai = IQAMAX ;
      } else if ( o_iqai < -IQAMAX ) {
         o_iqai = -IQAMAX ;
      } else {
         o_iqai += ( KSI * we ) ;
      }
/************************************************************************** /
/*      電流制御処理                                                        */
/************************************************************************** /
      ida = ( ( ( iva * sin( tre ) ) - ( iua * sin( tre + 2*RAD/3 ) ) ) ) >> SGETA ;
      iqa = ( ( ( iva * sin( tre + RAD/4 ) ) - ( iua * sin( tre + 11*RAD/12 ) ) ) ) >> SGETA ;

      o_vda = ( KI * -ida ) >> KIGETA ;
      o_vqa = ( KI * ( o_iqa  - iqa )  ) >> KIGETA ;
/************************************************************************** /
/*      3相電圧変換処理                                                     */
/************************************************************************** /
      if ( init_flag == 2 ) {                                 /* CW初起動処理 */
         o_trm += ( SPEED_MINI * TS / 20000 ) ;
         tre = ( o_trm * P ) % RAD ;
         o_vda = 0 ;
         o_vqa = 100 ;
         if  ( o_trm > ( 2 * RAD ) ) {
            init_flag = 0;
         }
      } else if ( init_flag == 3 ) {                          /* CCW初起動処理 */
         o_trm -= ( SPEED_MINI * TS / 20000 ) ;
```

```
            tre = ( o_trm * P ) % RAD ;
            o_vda = 0 ;
            o_vqa = 100 ;
            if  ( o_trm < -( 2 * RAD ) ) {
                init_flag = 0;
            }
        } else {
            o_trm = 0 ;
        }
        o_vua = ( ( ( o_vda * sin( tre + RAD/4 ) ) - ( o_vqa * sin( tre ) ) ) ) >> SGETA ;
        o_vva = ( ( ( o_vda * sin( tre + 11*RAD/12 ) ) - ( o_vqa * sin( tre + 2*RAD/3 ) ) ) ) >> SGETA ;
        o_vwa  = -o_vua - o_vva ;

        cont_time1 = ( TP1CNT - s_time ) ;       /* uSECに変換 */
/************************************************************************** /
/*      PWM変換出力処理                                                    */
/************************************************************************** /
        OUT_data( WRESET, 0 ) ;                  /* ウォッチドック・タイマRESET */
        HZA0CTL0 |= 0x04;                        /* PWM出力ON */
/* PWMカウンタ値計算出力 */
        o_vua += ( PWM_DATA  / 2 ) ;
        o_vva += ( PWM_DATA  / 2 ) ;
        o_vwa += ( PWM_DATA  / 2 ) ;

        if  ( o_vua <= 0 ) {
            o_vua = 1 ;
        } else if  ( o_vua >= PWM_DATA  ) {
            o_vua = PWM_DATA  - 1 ;
        }
        if  ( o_vva <= 0 ) {
            o_vva = 1 ;
        } else if  ( o_vva >= PWM_DATA  ) {
            o_vva = PWM_DATA  - 1 ;
        }
        if  ( o_vwa <= 0 ) {
            o_vwa = 1 ;
        } else if  ( o_vwa >= PWM_DATA  ) {
            o_vwa = PWM_DATA  - 1 ;
        }

        TQ0CCR1 = o_vua ;
        TQ0CCR2 = o_vva ;
        TQ0CCR3 = o_vwa ;
    } else {
        HZA0CTL0 |= 0x08;                        /* PWM出力OFF */
        now_speed = 0;
        cont_time1 = 0;
    }
}
```

4.4.8　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）                    */
/************************************************************************** /
_ _multi_interrupt
void    int_ETC(void)
{
/* 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 += abs( now_speed ) ;
        disp_co -= 1 ;
    }
}
```

4.4.9　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                   */
/************************************************************************** /
_ _multi_interrupt
void    int_AD0(void)
{
    iua = (( ( ADA0CR0 >> 6 ) & 0x3ff ) - 0x200) ;
    if  ( abs(iua) > MAX_I ) {
        HZA0CTL0 |= 0x08 ;                          /* PWM出力OFF */
        error_flag = ERR_NO1 ;                      /* エラーNOセット */
    }
    volume = 1023 - ( ( ADA0CR1 >> 6 ) & 0x3ff ) ;      /* ボリューム値セット */
    Motor_CONT() ;
    cont_time = TP1CNT;                             /* uSECに変換 */
}
/************************************************************************** /
/*      ボリューム用A/Dコンバータ割り込み処理                                */
/************************************************************************** /
_ _interrupt
void    int_AD1(void)
{
    iva = (( ADA1CR0 & 0x3ff ) - 0x200) ;
    if  ( abs(iva) > MAX_I ) {
        HZA0CTL0 |= 0x08 ;                          /* PWM出力OFF */
        error_flag = ERR_NO1 ;                      /* エラーNOセット */
```

```
    }
}
```

4. 4. 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 )
{
/* ポート・モード・レジスタ初期化 */
    PM3 = 0xfe ;
    PM4 = 0xff ;
    PMDL  = 0x0000 ;

    OUT_data( LED11, 0xff ) ;                   /* LED OFF */
    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 ) ;
/* 10 mSECタイマ  TMP0設定 */
    TP0CTL0 = 0x05 ;                            /* fXX/64セレクト */
    TP0CTL1 = 0x00 ;                            /* インターバル・タイマ・モード・セレクト */
    TP0CCR0 = 10000 ;                           /* 10 mSEC */
    TP0CTL0 |= 0x80 ;                           /* タイマ・スタート */
    TP0CCIC0= 0x06;
/* モータ制御割り込み用タイマ  TMP1設定 */
    TP1CTL0 = 0x05 ;                            /* fXX/64セレクト */
    TP1CTL1 = 0x00 ;                            /* インターバル・タイマ・モード・セレクト */
    TP1CCR0 = TS ;                              /* TS uSEC */
    TP1CTL0 |= 0x80 ;                           /* タイマ・スタート */
    TP1CCIC0= 0x01;
/* TMENC10 初期化 */
    TUM10  = 0x80 ;                             /* UDC モード選択 */
    PRM10 = 0x07 ;                              /* 動作モード4選択 */
    TMC10 = 0x40 ;                              /* カウント開始 */
/* TMQ0 初期化 */
    TQ0CTL0 = 0x00;                             /* fXX /2  (64 MHz/2 = 32 MHz) */
    TQ0CTL1 = 0x07;                             /* 6相PWM出力モード・セレクト */
    TQ0IOC0 = 0x55;                             /* 正相 通常出力,出力許可 */
    TQ0IOC1 = 0x00;                             /* TMQ0のTIQ00-TIQ03,EVTQ0,TRGQ0端子 */
    TQ0IOC2 = 0x00;                             /* は使用しない */
    TQ0OPT0 = 0x00;                             /* コンペア・モード・セレクト */
    TQ0CCR0 = PWM_DATA ;                        /* 搬送波周期 20 kHz */
    TQ0CCR1 = PWM_DATA /2;                      /* U相　デューディ50設定 */
```

```
    TQ0CCR2 = PWM_DATA /2;                  /* V相　デューディ50設定 */
    TQ0CCR3 = PWM_DATA /2;                  /* W相　デューディ50設定 */
    TQ0DTC  = 180;                          /* デットタイム　6 uSEC */
    TQ0OPT1 = 0x00;                         /* 間引きなし,山／谷割り込み */
                                            /* を使用しない */

    TQ0OPT2 = 0x04;                         /* ・リロード間の間引きなし */
                                            /* ・デット・タイム・カウンタは */
                                            /*   クリア再カウント */
                                            /* ・INTTP0CC0割り込みのA/Dトリガ*/
                                            /*   出力をアップ・カウント時に出力 */
                                            /* ・INTTP0CC0割り込みのA/Dトリガ*/
                                            /*   出力許可 */

    TQ0IOC3 = 0xfc;                         /* 逆相 反転出力,出力許可 */
    PMC1 = 0x3F;                            /* 兼用機能モード */
    PFCE1 = 0x00;                           /* TOQ0T1-TOQ0T3,TOQ0B1-TOQ0B3出力選択 */
    PFC1 = 0x00;

    HZA0CTL0 = 0x00;
    HZA0CTL0 = 0xD0;                        /* ハイ・インピーダンス動作許可 */
                                            /* TOQ0OFF端子立ち上がりエッジ有効 */
    HZA2CTL0 = 0x00;                        /* アナログ入力によるハイ・インピーダンスOFF */
    TQ0CTL0 |= 0x80;                        /* 6相PWM出力モード・スタート */
/* A/D 設定 */
    ADA0M0 = 0x30 ;                         /* ANI00,ANI01 */
    ADA0M1 = 0x01 ;
    ADA0S  = 0x01 ;
    OP0CTL0 = 0x00 ;                        /* オペアンプ無効 */
    OP0CTL1 = 0x00 ;                        /* コンパレータ無効 */
    AD0IC  = 0x03 ;

    ADA1M0 = 0x20 ;                         /* ANI10 */
    ADA1M1 = 0x01 ;
    ADA1S  = 0x00 ;
    OP1CTL0 = 0x00 ;                        /* オペアンプ無効 */
    OP1CTL1 = 0x00 ;                        /* コンパレータ無効 */
    AD1IC  = 0x03 ;
}
```

4.4.11　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 */
    o_trm = 0 ;
    before_enc  = -TMENC10 ;                    /* 前回エンコーダ値を設定 */
}
```

4.4.12　回転開始初期化処理関数

```
/************************************************************************** /
/*      回転開始初期化                                                      */
/************************************************************************** /
void    start_init( void )
{
    int i;
/*  */
    for ( i = 0 ; i < 21 ; i++ ) before_posi[i][1] = 0;
    total_sa = 0 ;
    sum_speed = 0 ;
    speed_co = 100000 / TS ;
}
```

4.4.13　sin計算処理関数

```
/************************************************************************** /
/*      sin x                                                               */
/*          データ                                                          */
/*              ラジアン単位                                                */
/*          戻り値                                                          */
/*              サイン値*16384                                              */
/************************************************************************** /
int sin( int x )
{
    x = x % RAD ;
    if  ( x < 0 ) x += RAD ;
    if  ( x < (RAD/4) ){
        return sins( x ) ;
    } else if ( x < (RAD/2) ) {
        return sins( (RAD/2) - x ) ;
    } else if ( x < (RAD*3/4) ) {
        return -sins( x - (RAD/2) ) ;
    } else {
        return -sins( RAD - x ) ;
    }
}
```

```
/*****************************************************************************/
int sins( int x )
{
short z1, z2, z3, z4, z5 ;
/*  */
   if  ( x <= (RAD/8) ) {
      z1 = (x << SGETA) / (RAD/8) ;
      z2 = z1 * z1 >> SGETA ;
      z3 = z1 * z2 >> SGETA ;
      z5 = z2 * z3 >> SGETA ;
      return ( (12868*z1) - (1322*z3) + (40*z5) ) >> SGETA ;
   } else {
      x = (RAD/4) - x ;
      z1 = (x << SGETA) / (RAD/8) ;
      z2 = z1 * z1 >> SGETA ;
      z4 = z2 * z2 >> SGETA ;
      return ( (268432772) - (5050*z2) + (252*z4) ) >> SGETA ;
   }
}
```

4. 4. 14　V850E/IA4用のリンク・ディレクティブ・ファイル

```
/*************************************************************************** */
/*      V850E/IA4用のリンク・ディレクティブ・ファイル                          */
/*************************************************************************** */
VECT_RESET: !LOAD ?RX V0x00000000 {
        .vect_RESET  = $PROGBITS ?AX .vect_RESET;
};
ID_NO: !LOAD ?RX V0x00000070 {
        .id_NO  = $PROGBITS ?AX .id_NO;
};
VECT_ETC: !LOAD ?RX V0x00000250 {
        .vect_ETC  = $PROGBITS ?AX .vect_ETC;
};
VECT_MOTOR: !LOAD ?RX V0x00000280 {
        .vect_MOTOR  = $PROGBITS ?AX .vect_MOTOR;
};
VECT_AD0: !LOAD ?RX V0x00000400 {
        .vect_AD0  = $PROGBITS ?AX .vect_AD0;
};
VECT_AD1: !LOAD ?RX V0x00000410 {
        .vect_AD1  = $PROGBITS ?AX .vect_AD1;
};

HANDLER: !LOAD ?RX V0x00001000 {
        .handler  = $PROGBITS ?AX .handler;
};
TEXT:  !LOAD ?RX {
        .text  = $PROGBITS ?AX .text;
};
```

```
CONST : !LOAD ?R {
        .const = $PROGBITS ?A .const;
};

DATA : !LOAD ?RW  V0xffffd800 {
        .data  = $PROGBITS ?AW  ;
        .sdata = $PROGBITS ?AWG ;
        .sbss  = $NOBITS   ?AWG ;
        .bss   = $NOBITS   ?AW  ;
};

_ _tp_TEXT @ %TP_SYMBOL;
_ _gp_DATA @ %GP_SYMBOL &_ _tp_TEXT{DATA};
_ _ep_DATA @ %EP_SYMBOL;
```

〔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。