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。

お客様各位

資料中の「日立製作所」、「日立XX」等名称の株式会社ルネサス テクノロジへの変更について

2003年4月1日を以って三菱電機株式会社及び株式会社日立製作所のマイコン、ロジック、アナログ、ディスクリート半導体、及びDRAMを除くメモリ（フラッシュメモリ・SRAM等）を含む半導体事業は株式会社ルネサス テクノロジに承継されました。従いまして、本資料中には「日立製作所」、「株式会社日立製作所」、「日立半導体」、「日立XX」といった表記が残っておりますが、これらの表記は全て「株式会社ルネサス テクノロジ」に変更されておりますのでご理解の程お願い致します。尚、会社商標・ロゴ・コーポレートステートメント以外の内容については一切変更しておりませんので資料としての内容更新ではありません。

ルネサステクノロジ ホームページ（http://www.renesas.com）

2003年4月1日
株式会社ルネサス テクノロジ
カスタマサポート部

RENESAS

ご注意

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ (http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。

アプリケーションノート

2 相励磁方式ステッピングモータ

要旨

H8/3664 の内蔵機能のうち、P83～P80 とタイマ W コンペアマッチ機能を用いて 2 相ステッピングモータを制御します。
ステッピングモータは、2 相励磁方式で制御します。

動作確認デバイス

H8/300H Tiny シリーズ －H8/3664－

目次

ご注意

1. 本書に記載の製品及び技術のうち「外国為替及び外国貿易法」に基づき安全保障貿易管理関連貨物・技術に該当するものを輸出する場合，または国外に持ち出す場合は日本国政府の許可が必要です。
2. 本書に記載された情報の使用に際して，弊社もしくは第三者の特許権，著作権，商標権，その他の知的所有権等の権利に対する保証または実施権の許諾を行うものではありません。また本書に記載された情報を使用した事により第三者の知的所有権等の権利に関わる問題が生じた場合，弊社はその責を負いませんので予めご了承ください。
3. 製品及び製品仕様は予告無く変更する場合がありますので，最終的な設計，ご購入，ご使用に際しましては，事前に最新の製品規格または仕様書をお求めになりご確認ください。
4. 弊社は品質・信頼性の向上に努めておりますが，宇宙，航空，原子力，燃焼制御，運輸，交通，各種安全装置，ライフサポート関連の医療機器等のように，特別な品質・信頼性が要求され，その故障や誤動作が直接人命を脅かしたり，人体に危害を及ぼす恐れのある用途にご使用をお考えのお客様は，事前に弊社営業担当迄ご相談をお願い致します。
5. 設計に際しては，特に最大定格，動作電源電圧範囲，放熱特性，実装条件及びその他諸条件につきましては，弊社保証範囲内でご使用いただきますようお願い致します。保証値を越えてご使用された場合の故障及び事故につきましては，弊社はその責を負いません。また保証値内のご使用であっても半導体製品について通常予測される故障発生率，故障モードをご考慮の上，弊社製品の動作が原因でご使用機器が人身事故，火災事故，その他の拡大損害を生じないようにフェールセーフ等のシステム上の対策を講じて頂きますようお願い致します。
6. 本製品は耐放射線設計をしておりません。
7. 本書の一部または全部を弊社の文書による承認なしに転載または複製することを堅くお断り致します。
8. 本書をはじめ弊社半導体についてのお問い合わせ，ご相談は弊社営業担当迄お願い致します。

1. 仕様

1. H8/3664 の内蔵機能のうち、P83～P80 とタイマ W コンペアマッチ機能を用いて 2 相ステッピングモータを制御します。
2. ステッピングモータは、2 相励磁方式で制御します。
3. 本タスクでは、ステッピングモータを正転→停止→逆転→停止の動作を繰り返します。
4. 本タスクでは、ソフトウェアによる Slue Up 及び Slue Down 処理を行います。
5. 2 相ステッピングモータ制御の接続図を図 1.1 に示します。

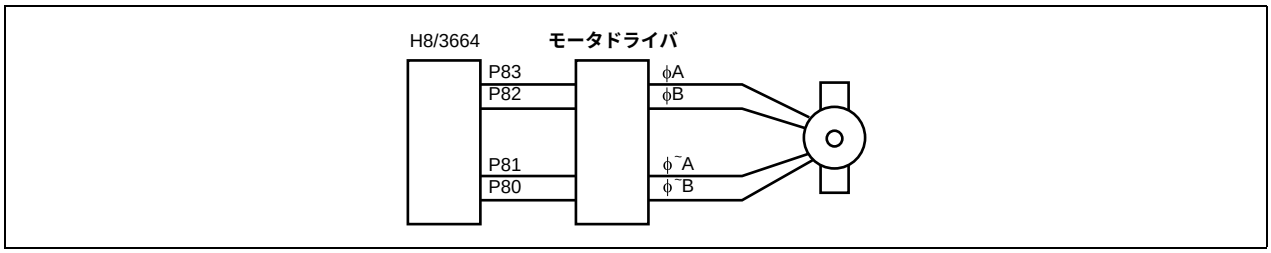


図 1.1　2 相ステッピングモータ制御の接続図

2. 考え方

1. ステッピングモータ動作例

   ステップ角 7.5[deg./step]の 2 相ステッピングモータを 2 相励磁方式で動作させる例を図 2.1 に示します。動作概要は、以下の通りです。

   （a）図 2.1 のようにパルスが High のとき、対応する相を励磁します。

   （b）まず、¯B 相と A 相を同時に励磁します。このときロータは、¯B 相と A 相の中間に位置します。

   （c）次に、A 相と B 相を同時に励磁します。このときロータは、A 相と B 相の中間に位置します。以下、2 相励磁方式は、隣り合う 2 つの相（¯B,A 相、A,B 相、B,¯A 相、¯A,¯B 相）を励磁し、ロータを回転させます。

   （d）逆転動作の場合は、¯A,¯B 相→B,¯A 相→A,B 相→¯B,A 相を順番に励磁することでステッピングモータを回転させます。

   （e）停止動作は、正転動作の最後の相または、逆転動作の最後の相を一定時間励磁し続けることでステッピングモータを停止させます。

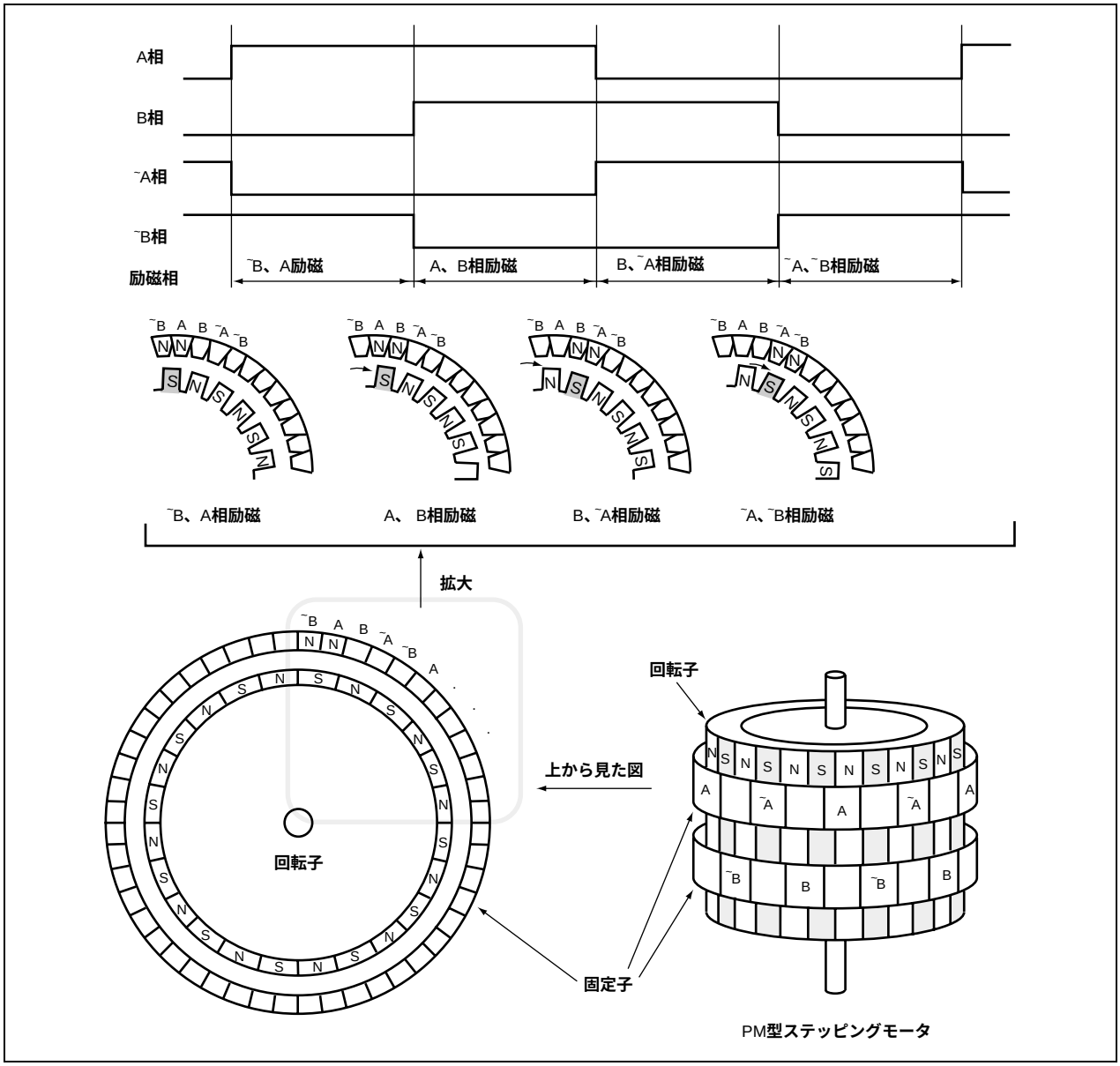


図 2.1　ステッピングモータ動作例

2. ノンオーバラップ時間

出力パターン切り替え時に貫通電流防止期間 n（ノンオーバラップ時間）を挿入します。励磁相の切り替え時に生じるターンオフの遅れにより、ドライバが破壊する危険性があり、これを防ぐためにノンオーバラップ時間を挿入し、時間遅れを持たせて対策します。

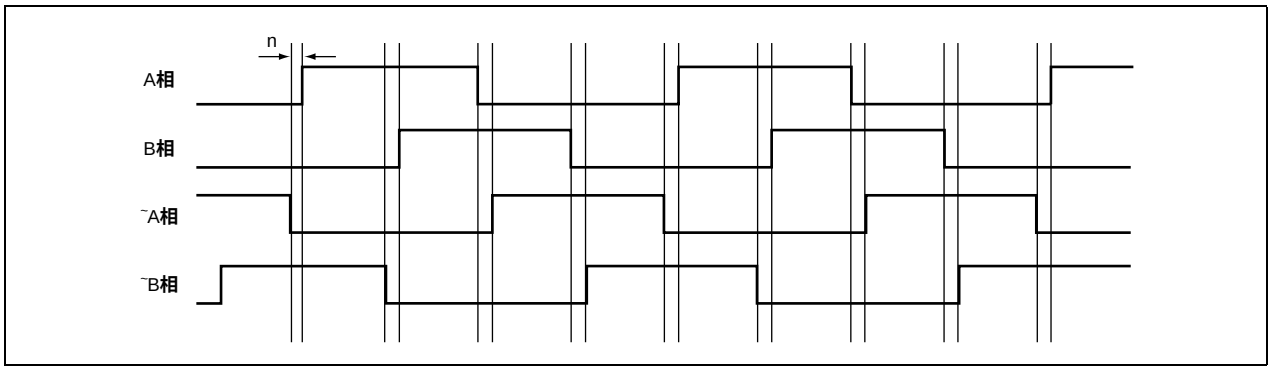


図 2.2　ノンオーバラップ時間出力例

3. Slue Up、Slue Down 動作

加減速制御されたパルスを出力します。Slue Up、Slue Down 動作を行うことにより、モータの脱調を防止します。モータを動作させる際に、急に周期の短いパルスを出力すると、モータは、負荷に追いつけず、回転しないことがあります。これを防止する対策として Slue Up および Slue Down を行います。動作原理を以下に示します。

（a）徐々にパルス周期を短くし、設定した量のパルスを出力する。（Slue Up）

（b）一定のパルス周期で設定した量のパルスを出力する。

（c）徐々にパルス周期を長くし、設定した量のパルスを出力する。（Slue Down）

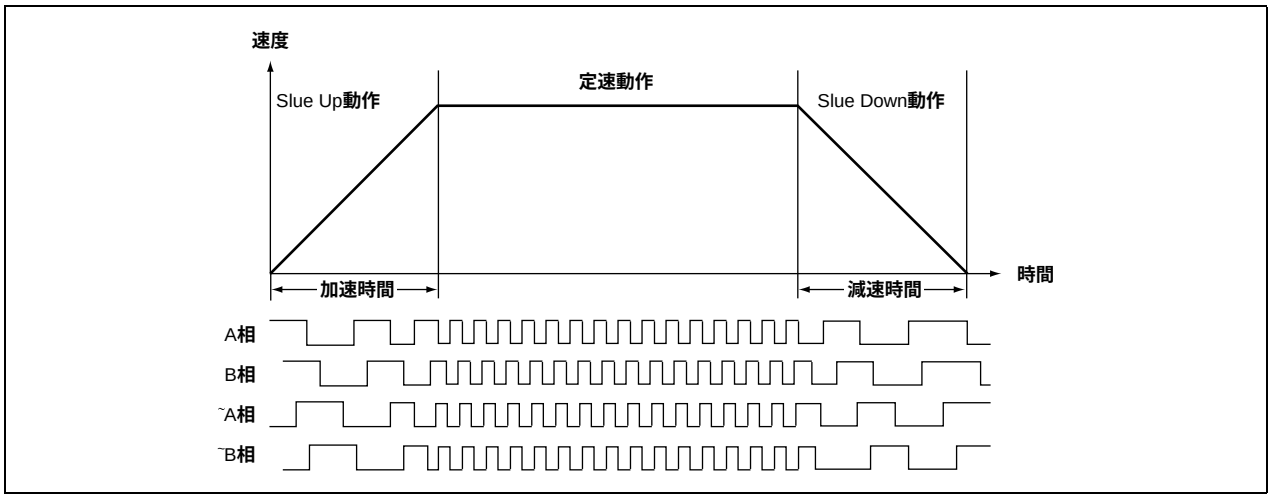


図 2.3　Slue Up、Slue Down 動作例

3. 使用機能説明

1. 本タスク例ではパーマネントマグネット型のステッピングモータ（KP6P8-701、日本サーボ株式会社）を使用しています。表 3.1 に KP6P8-701 の標準仕様を示します。

表 3.1 KP6P8-701 標準仕様

項　　目	値
型式名	KP6P8-701
相数	2
ステップ角[deg./step]	7.5
電圧[V]	12
電流[A/PHASE]	0.33
巻線抵抗[Ω/PHASE]	36
インダクタンス[mH/PHASE]	28
最大静止トルク[mN･m]	78.4
ディテントトルク[mN･m]	1.3
ロータイナーシャ[g･cm^2]	23.7

2. ステッピングモータ制御における H8/3664 の使用機能について説明します。図 3.1 に本タスク例における使用機能のブロック図を示します。

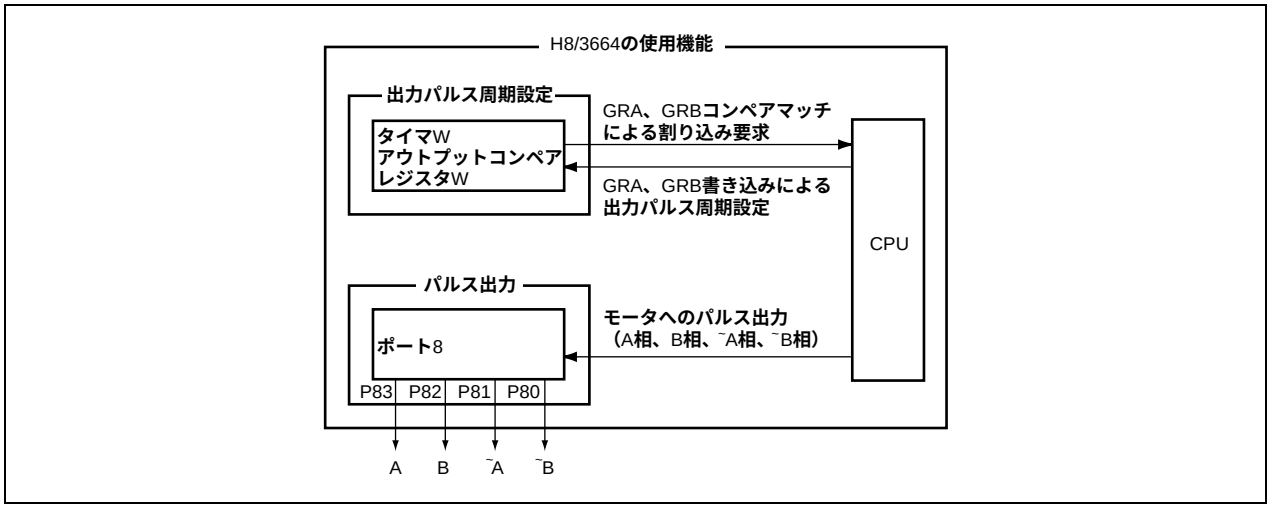


図 3.1　H8/3664 の使用機能

3. タイマ W の各機能を説明します。

（a）タイマ W のブロック図を図 3.2 に示します。

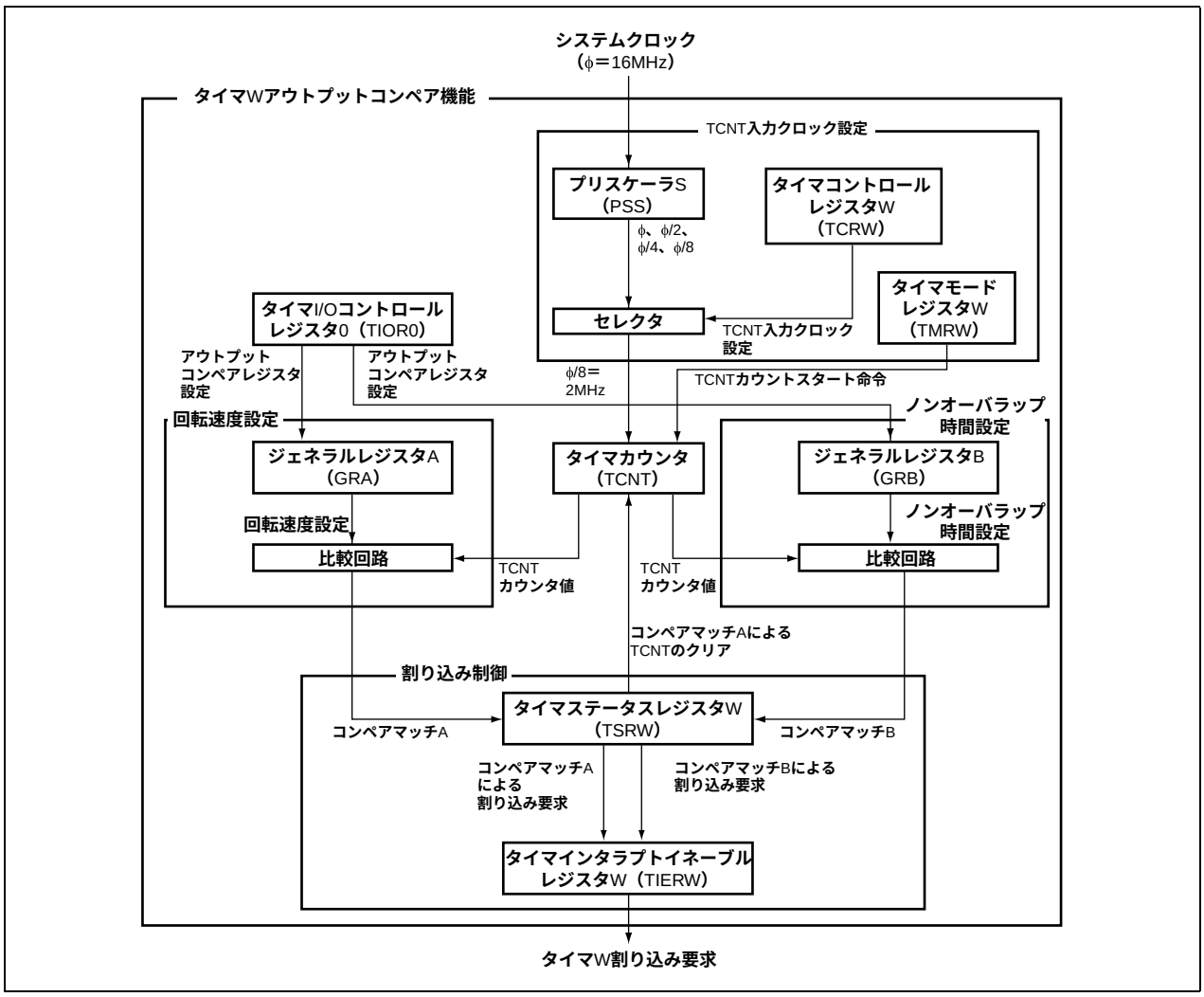


図 3.2　タイマ W 機能のブロック図

（b）タイマ W は、アウトプットコンペア機能、インプットキャプチャ機能を内蔵した 16 ビットの多機能タイマです。本タスク例では、タイマ W のアウトプットコンペア機能を使用します。以下にタイマ W 機能のブロック図について説明します。

- システムクロック（φ）
  16MHz の OSC クロックで、CPU および周辺機能を動作させるための基準クロックです。
- プリスケーラ S（PSS）
  φを入力とする 13 ビットのカウンタで、1 サイクルごとにカウントアップします。
- タイマモードレジスタ W（TMRW）
  TCNT のカウンタを開始します。
- タイマコントロールレジスタ W（TCRW）
  8 ビットのリード／ライト可能なレジスタで、TCNT 入力クロックの選択、およびコンペアマッチ A による TCNT クリアの設定を行います。
- タイマインタラプトイネーブルレジスタ（TIERW）
  8 ビットのリード／ライト可能なレジスタで、各割り込み要求の許可／禁止を制御します。
- タイマステータスレジスタ W（TSRW）
  8 ビットのレジスタで、各割り込み要求信号の制御を行います。
- タイマ I/O コントロールレジスタ 0（TIOR0）
  8 ビットのリード／ライト可能なレジスタで、アウトプットコンペアレジスタの設定を行います。
- タイマカウンタ（TCNT）
  16 ビットのリード／ライト可能なアップカウンタで、入力するシステムクロック／外部クロックにより、カウントアップされます。入力するクロックはφ、φ/2、φ/4、φ/8 及び外部クロックの計 5 種類のクロックから選択可能です。本タスク例では、TCNT の入力クロックにφ/8 を選択しています。
- ジェネラルレジスタ A（GRA）
  16 ビットのリード／ライト可能なレジスタで、GRA の内容は TCNT と常に比較されており、両者の値が一致すると、TSRW の IMFA が 1 にセットされます。この時、TIERW の IMIEA が 1 ならば、CPU に割り込みを要求します。
- ジェネラルレジスタ B（GRB）
  16 ビットのリード／ライト可能なレジスタで、GRB の内容は TCNT と常に比較されており、両者の値が一致すると、TSRW の IMFB が 1 にセットされます。この時、TIERW の IMIEB が 1 ならば、CPU に割り込みを要求します。

4. ポート 8 の各機能を説明します。

（a）ポート 8 のブロック図を図 3.3 に示します。

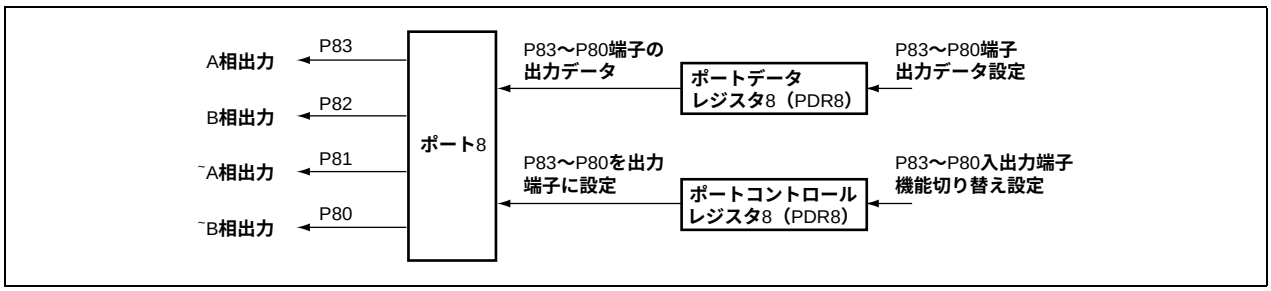


図 3.3　ポート 8 機能のブロック図

（b）ポート 8 は、8 ビット入出力ポートです。本タスク例では、ポート 8 のうち P83～P80 を使用します。以下にポート 8 の機能について説明します。

- ポートデータレジスタ 8（PDR8）

  P83～P80 をステッピングモータの励磁相駆動に使用します。
- ポートコントロールレジスタ 8（PCR8）

  P83～P80 を出力端子に設定します。

5. 本タスク例の機能割り付けを表 3.2 に示します。

表 3.2　機能割付け

機　　能	機能割付け
システムクロック PSS TCNT	ステッピングモータを動作させる基準クロック
TMRW	TCNT のカウンタを開始
TCRW	TCNT 動作設定
TIERW	各割り込み要求の許可／禁止を制御
TSRW	各割り込み要求信号の制御
TIOR0	アウトプットコンペアレジスタの設定
GRA	ステッピングモータ 1 ステップ時間の設定
GRB	ノンオーバラップ時間の設定
PDR8 PCR8	ステッピングモータの励磁相駆動

4. 動作説明

1. ステッピングモータ制御のフローチャートを図 4.1 に示します。

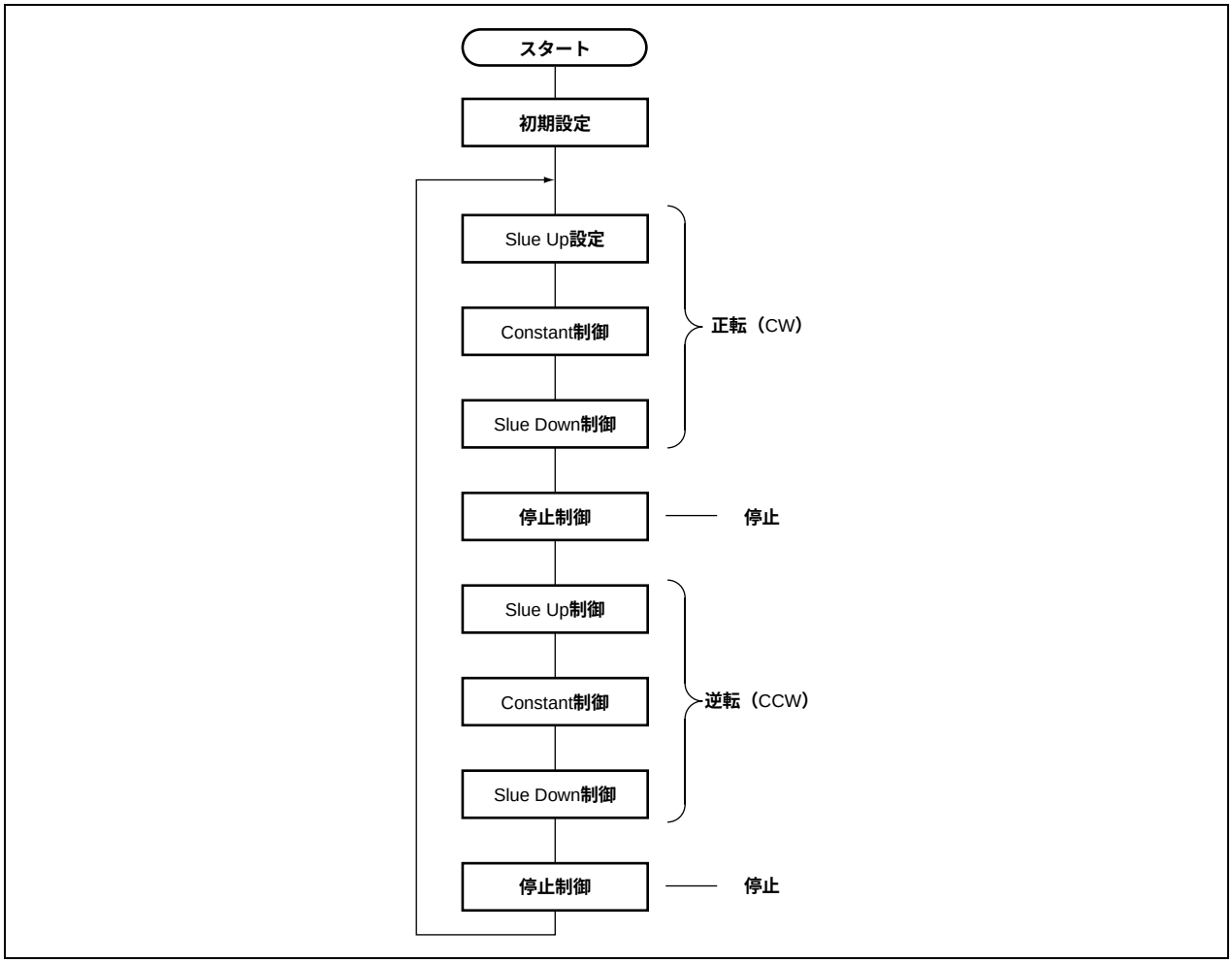


図 4.1 ステッピングモータ制御のフローチャート

2. タイマ W 割り込み時間の計算式

- アウトプットコンペアレジスタである GRA、GRB の設定によるタイマ W 割り込み時間の計算式を以下に示します。

$$\text{タイマ W 割り込み時間} = \frac{\text{GRA または GRB}}{(\text{システムクロック}\phi/8)}$$

$$= \frac{\text{GRA または GRB}}{(16\text{MHz}/8)}$$

$$= \frac{\text{GRA または GRB}}{2} \ [\mu\text{s}]$$

3. 正転時 Slue Up 制御の動作原理を図 4.2 に示します。

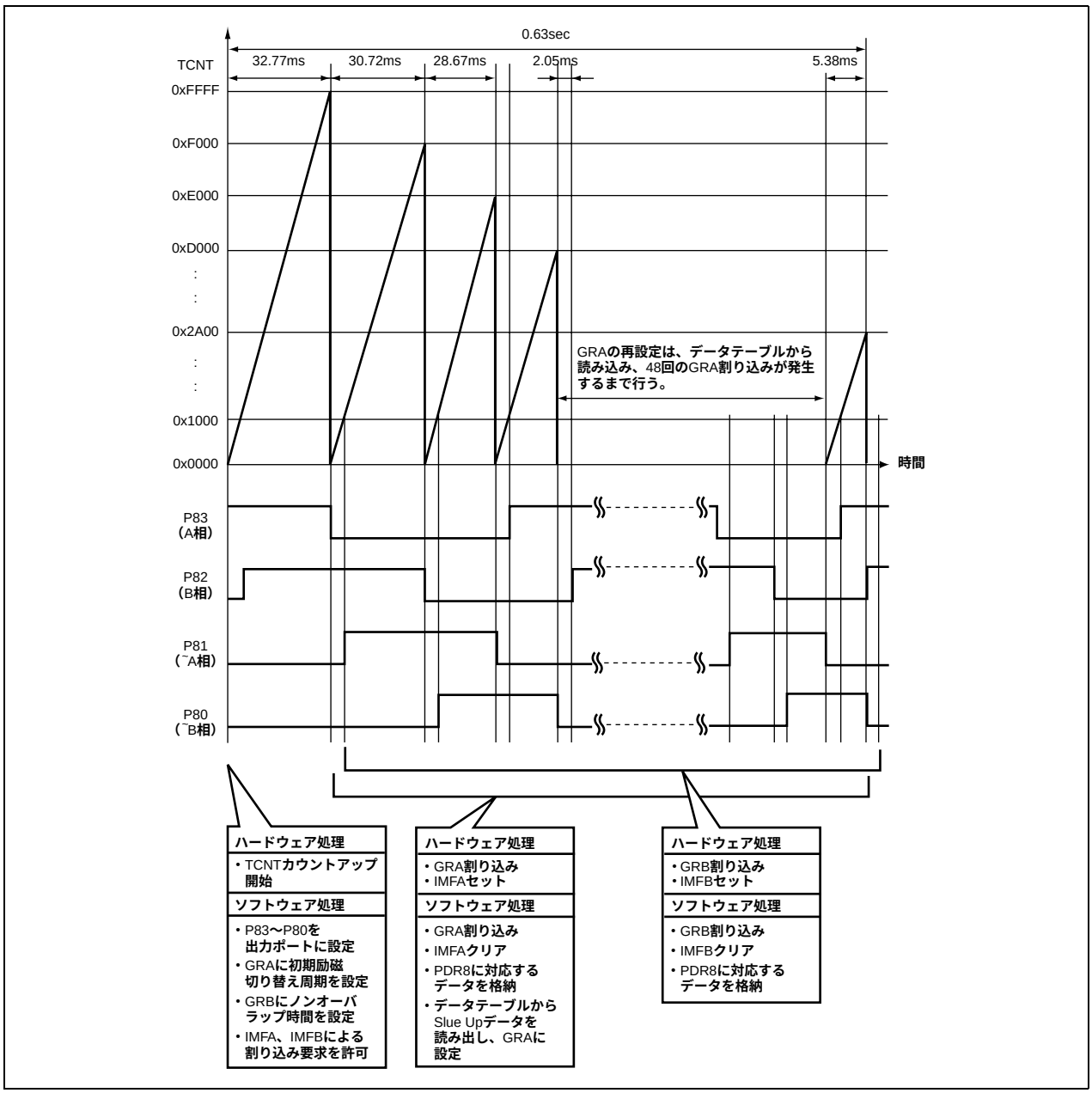


図 4.2 正転時 Slue Up 制御の動作原理

4. 正転時 Constant 制御の動作原理を図 4.3 に示します。

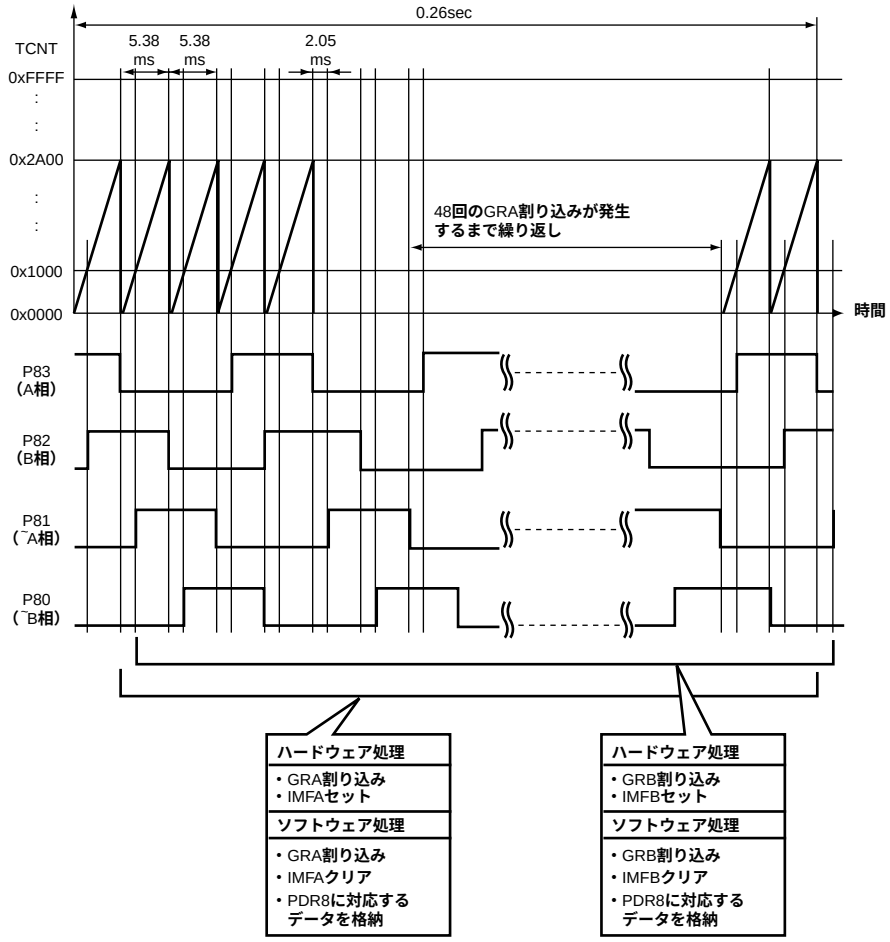


図 4.3 正転時 Constant 制御の動作原理

5. 正転時 Slue Down 制御の動作原理を図 4.4 に示します。

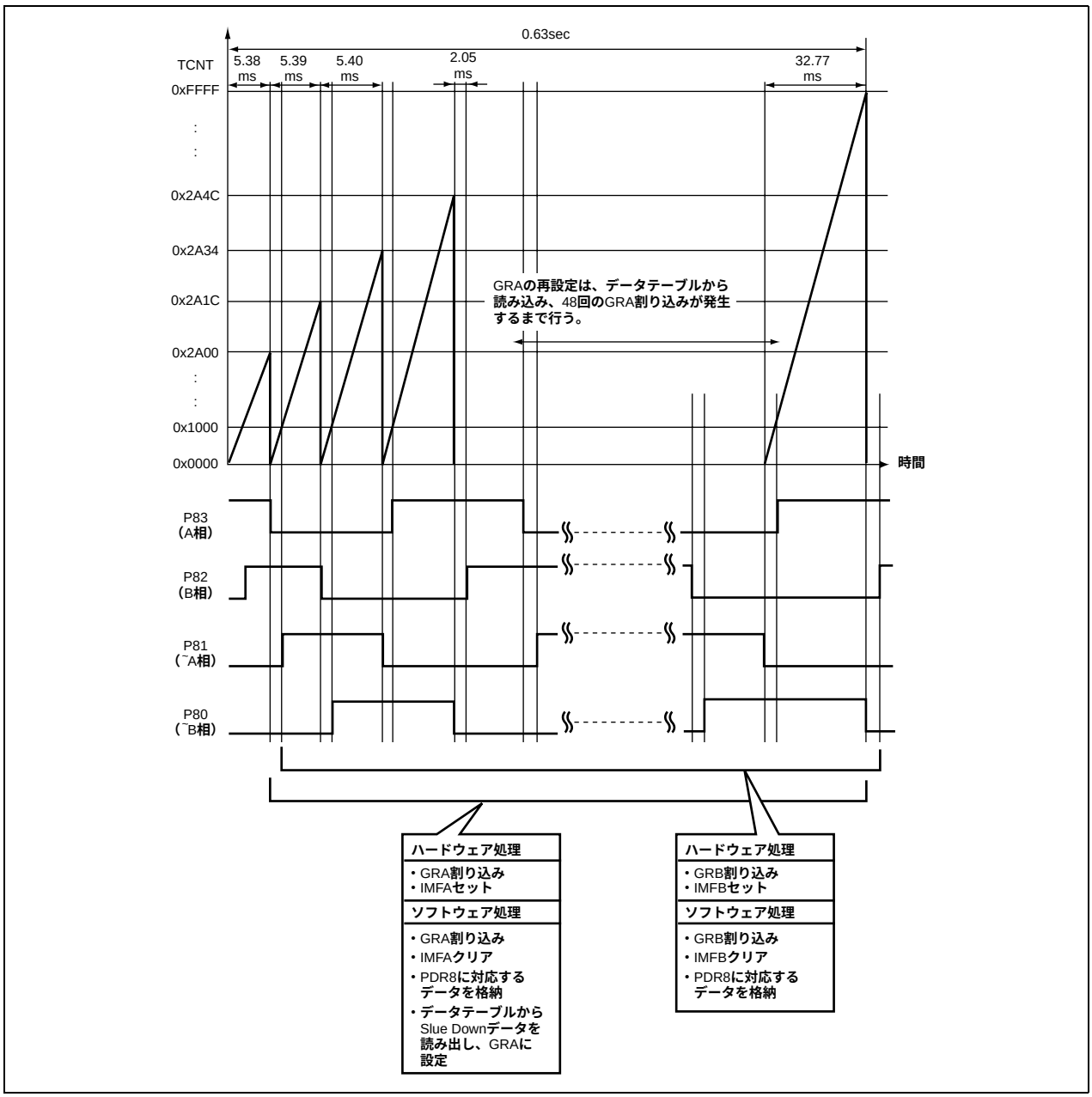


図 4.4 正転時 Slue Down 制御の動作原理

6. 停止制御の動作原理を図 4.5 に示します。

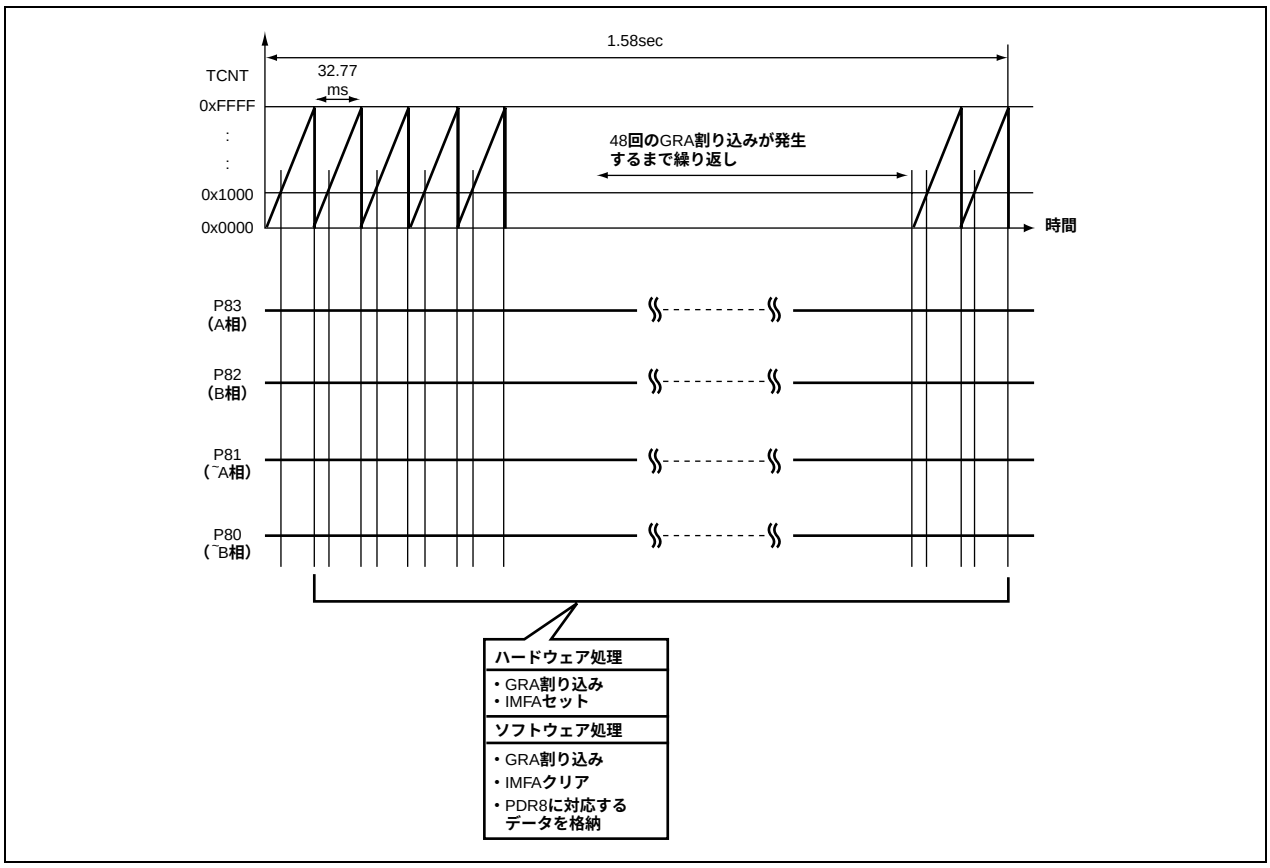


図 4.5 停止制御の動作原理

7. 逆転時 Slue Up 制御の動作原理を図 4.6 に示します。

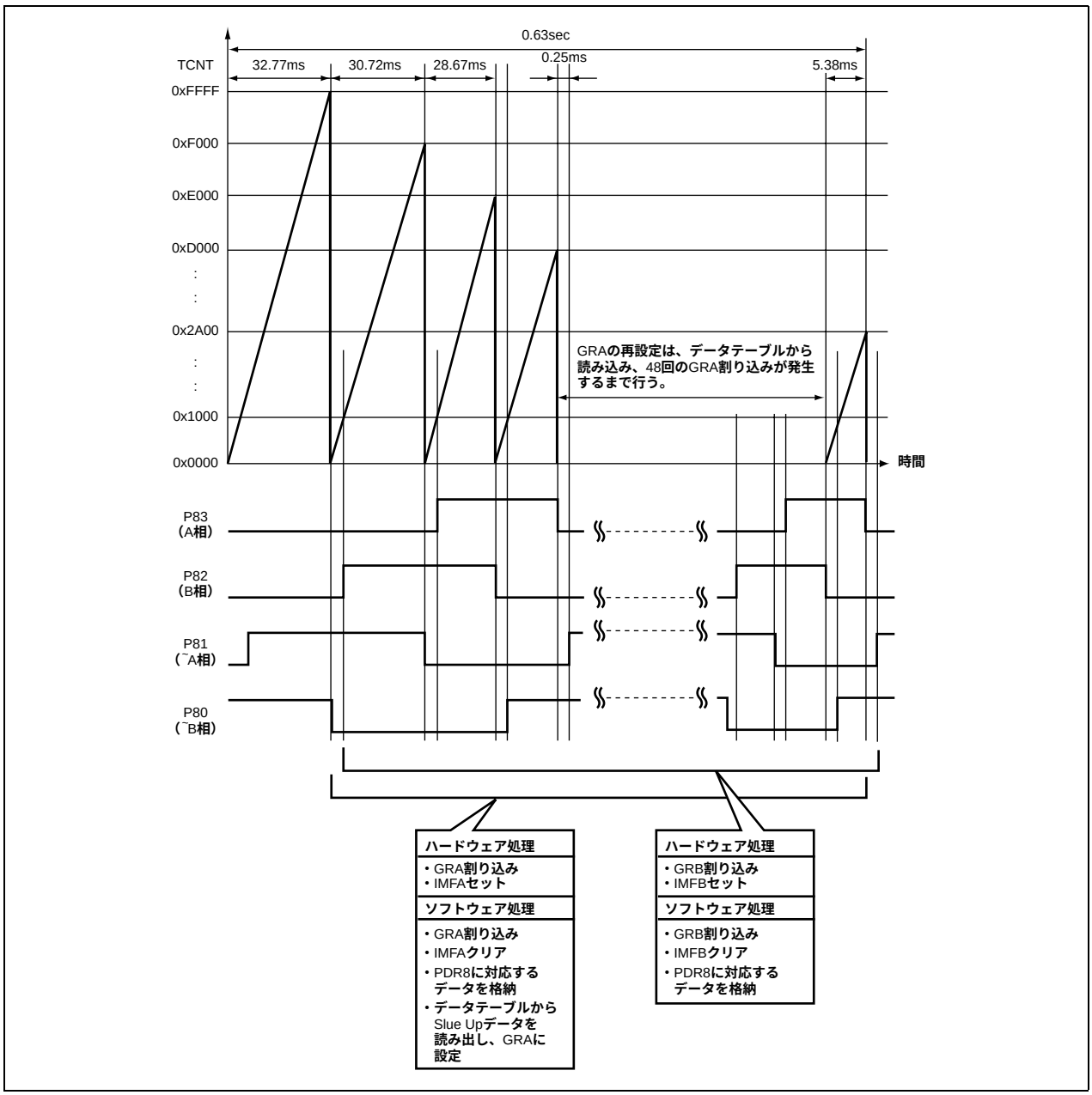


図 4.6 逆転時 Slue Up 制御の動作原理

8. 逆転時 Constant 制御の動作原理を図 4.7 に示します。

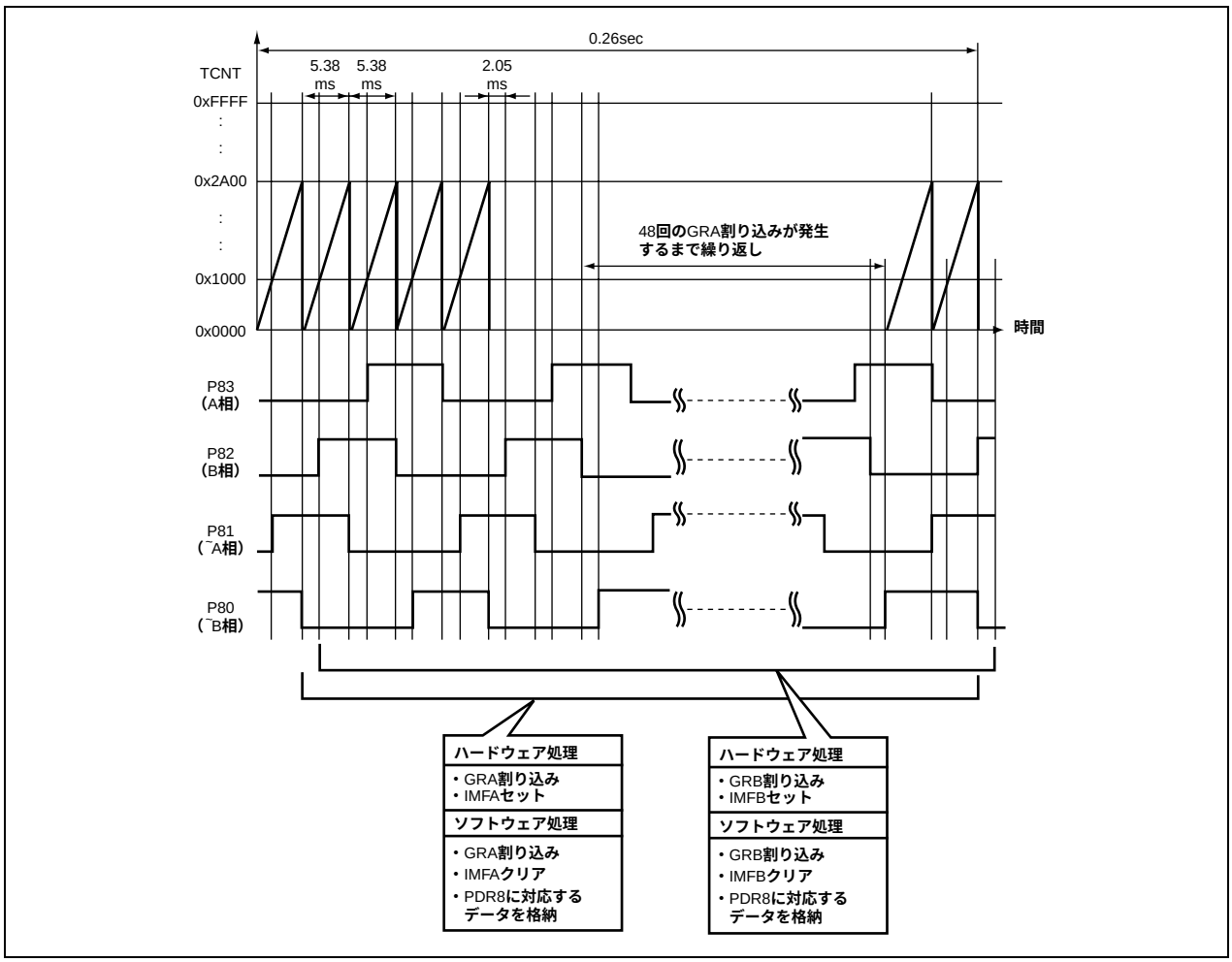


図 4.7 逆転時 Constant 制御の動作原理

9. 逆転時 Slue Down 制御の動作原理を図 4.8 に示します。

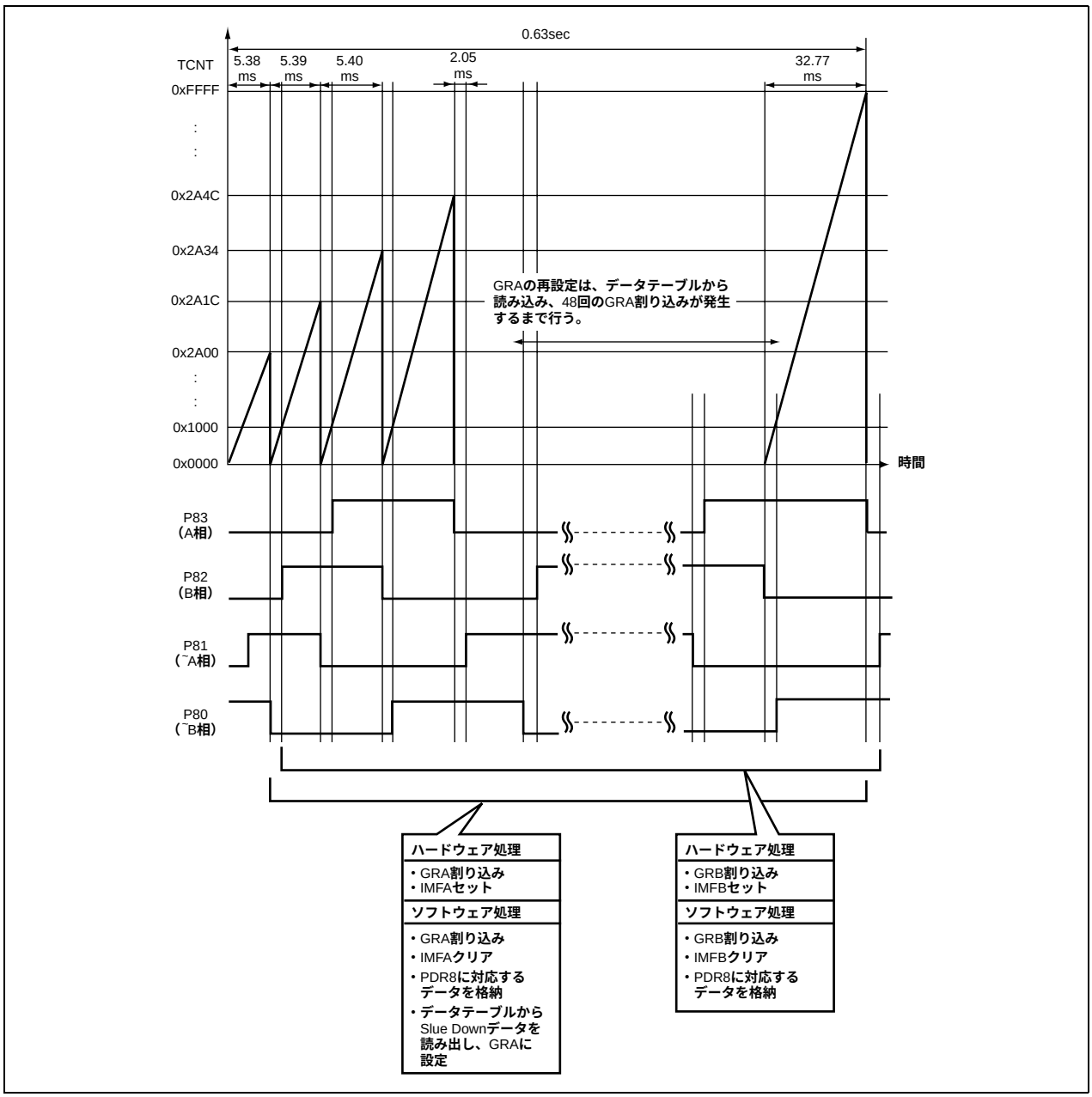


図 4.8 逆転時 Slue Down 制御の動作原理

5. 使用機能説明

5.1 モジュール説明

本タスク例のモジュールを表 5.1 に示します。

表 5.1 モジュール説明

モジュール名	関数名	機　　能
メインルーチン	main	グローバル変数、I/O ポート、タイマ W の初期設定、および割り込みの許可を行う
タイマ W 割り込み処理	twint	ステッピングモータをメイン動作ルーチン
正転 SlueUp 制御	fslueup	正転時の SlueUp 制御を行う
正転 SlueDown 制御	fsluedwn	正転時の SlueDown 制御を行う
正転 Constant 制御	fconst	正転時の Constant 制御を行う
回転停止	frstop	正転／逆転の停止を行う
逆転 SlueUp 制御	rslueup	逆転時の SlueUp 制御を行う
逆転 SlueDown 制御	rsluedwn	逆転時の SlueDown 制御を行う
逆転 Constant 制御	rconst	逆転時の Constant 制御を行う

5.2 引数の説明

本タスク例では、引数を使用しません。

5.3 使用内部レジスタ説明

本タスク例の使用内部レジスタを表 5.2 に示します。

表 5.2 使用内部レジスタ説明

レジスタ名		機　能	アドレス	設定値
TMRW	CTS	タイマモードレジスタ W（タイマカウンタスタート） ：CTS＝0 のとき、TCNT のカウンタ停止 ：CTS＝1 のとき、TCNT のカウンタ開始	0xFF80 ビット 7	1
TCRW	CCLR	タイマコントロールレジスタ W（カウンタクリア） ：CCLR＝0 のとき、コンペアマッチ A であっても、TCNT をクリアしない ：CCLR＝1 のとき、コンペアマッチ A によって TCNT をクリアする	0xFF81 ビット 7	1
	CKS2 CKS1 CKS0	タイマコントロールレジスタ W（クロックセレクト 2～0） ：CKS2＝0、CKS1＝1、CKS0＝1 のとき、TCNT の入力クロックをシステムクロックφ/8 に設定	0xFF81 ビット 6 ビット 5 ビット 4	CKS2＝0 CKS1＝1 CKS0＝1
TIERW	IMIEB	タイマインタラプトイネーブルレジスタ W （インプットキャプチャ／コンペアマッチ割り込みイネーブル B） ：IMIEB＝0 のとき、IMFB による割り込み要求を禁止 ：IMIEB＝1 のとき、IMFB による割り込み要求を許可	0xFF82 ビット 1	1
	IMIEA	タイマインタラプトイネーブルレジスタ W （インプットキャプチャ／コンペアマッチ割り込みイネーブル A） ：IMIEA＝0 のとき、IMFA による割り込み要求を禁止 ：IMIEA＝1 のとき、IMFA による割り込み要求を許可	0xFF82 ビット 0	1
TSRW	IMFB	タイマステータスレジスタ W （インプットキャプチャ／コンペアマッチフラグ B） ：IMFB＝0 のとき、TCNT と GRB がコンペアマッチしていないことを示す ：IMFB＝1 のとき、TCNT と GRB がコンペアマッチしたことを示す	0xFF83 ビット 1	0
	IMFA	タイマステータスレジスタ W （インプットキャプチャ／コンペアマッチフラグ A） ：IMFA＝0 のとき、TCNT と GRA がコンペアマッチしていないことを示す ：IMFA＝1 のとき、TCNT と GRA がコンペアマッチしたことを示す	0xFF83 ビット 0	0
TIOR0	IOB2	タイマ I/O コントロールレジスタ 0（I/O コントロール B2） ：IOB2＝0 のとき、GRB をアウトプットコンペアレジスタとして機能 ：IOB2＝1 のとき、GRB をインプットキャプチャレジスタとして機能	0xFF84 ビット 6	0
	IOB1 IOB0	タイマ I/O コントロールレジスタ 0（I/O コントロール B1～B0） ：IOB1＝0,IOB0＝0 のとき、コンペアマッチによる端子出力を禁止	0xFF84 ビット 5 ビット 4	IOB1＝0 IOB0＝0
	IOA2	タイマ I/O コントロールレジスタ 0（I/O コントロール A2） ：IOA2＝0 のとき、GRA をアウトプットコンペアレジスタとして機能 ：IOA2＝1 のとき、GRA をインプットキャプチャレジスタとして機能	0xFF84 ビット 2	0
	IOA1 IOA0	タイマ I/O コントロールレジスタ 0 （I/O コントロール A1～A0） ：IOA1＝0,IOA0＝0 のとき、コンペアマッチによる端子出力を禁止	0xFF84 ビット 1 ビット 0	IOA1＝0 IOA0＝0

表 5.2　使用内部レジスタ説明（つづき）

レジスタ名	機　　能	アドレス	設定値
TCNT	タイマカウンタ ：システムクロックφ/8 を入力とする 16 ビットのカウンタ	0xFF86	0x0000
GRA	ジェネラルレジスタ A ：GRA の設定値と TCNT のカウンタ値が一致すると、コンペアマッチ A が発生する	0xFF88	0xFFFF
GRB	ジェネラルレジスタ B ：GRB の設定値と TCNT のカウンタ値が一致すると、コンペアマッチ B が発生する	0xFF8A	0x1000
PDR8	ポートデータレジスタ 8 ：P83～P80 をステッピングモータの励磁相駆動に使用する	0xFFDB	0x08
PCR8	ポートコントロールレジスタ 8 ：PCR8＝0x0F のとき、P83～P80 を出力端子に設定	0xFFEB	0x0F

5.4　グローバル変数説明

本タスク例のグローバル変数を表 5.3 に示します。

表 5.3　グローバル変数説明

変数名	機　　能	データ型／サイズ	使用関数
twcnt	ステッピングモータの励磁データである配列 pattbl[ ]の要素	char／1 バイト	main,twint,fslueup, fsluedwn,fconst,frstop, rslueup,rsluedwn,rconst
sluecnt	SlueUp,SlueDown 時に使用する配列 uptbl[ ]の要素	char／1 バイト	main,twint,fslueup, fsluedwn,rslueup,rsluedwn
nextmode	ステッピングモータの動作モードを設定する	char／1 バイト	main,twint
modecnt	ステッピングモータの動作モードの割り込み回数を設定する	short／2 バイト	main,twint
pattbl[8]	ステッピングモータの励磁パターンデータテーブル	unsigned char ／8 バイト	main,fslueup,fsluedwn, fconst,frstop,rslueup, rsluedwn,rconst
uptbl[48]	SlueUp,SlueDown 時の割り込み時間データテーブル	unsigned short ／96 バイト	main,fslueup,fsluedwn, rslueup,rsluedwn

5.5 データテーブル変数説明

- ステッピングモータ励磁パターン切り替えデータテーブル

  pattbl[8]＝{

  0x08, ……GRB 割り込みで P8 出力。A 相（P83）を励磁する。

  0x0C, ……GRA 割り込みで P8 出力。A 相（P83）、B 相（P82）を励磁する。

  0x04, ……GRB 割り込みで P8 出力。B 相（P82）を励磁する。

  0x06, ……GRA 割り込みで P8 出力。B 相（P82）、~A 相（P81）を励磁する。

  0x02, ……GRB 割り込みで P8 出力。~A 相（P81）を励磁する。

  0x03, ……GRA 割り込みで P8 出力。~A 相（P81）、~B 相（P80）を励磁する。

  0x01, ……GRB 割り込みで P8 出力。~B 相（P80）を励磁する。

  0x09, ……GRA 割り込みで P8 出力。~B 相（P80）、A 相（P83）を励磁する。

  }

- SlueUp、SlueDown 設定データテーブル

  uptbl[48]＝{

  0xFFFF,0xF000,0xE000,0xD000,0xC670,0xBC48,0xB1BC,0xAA50,0xA21C,0x98BC,

  0x9218,0x8D68,0x88B8,0x8408,0x7F58,0x7AA8,0x75F8,0x7148,0x6C98,0x6720,

  0x6338,0x5E24,0x5B04,0x56B8,0x5398,0x5140,0x4D58,0x4970,0x4650,0x4330,

  0x4010,0x3CF0,0x3AFC,0x3908,0x3714,0x3520,0x332C,0x3138,0x2F44,0x2DB4,

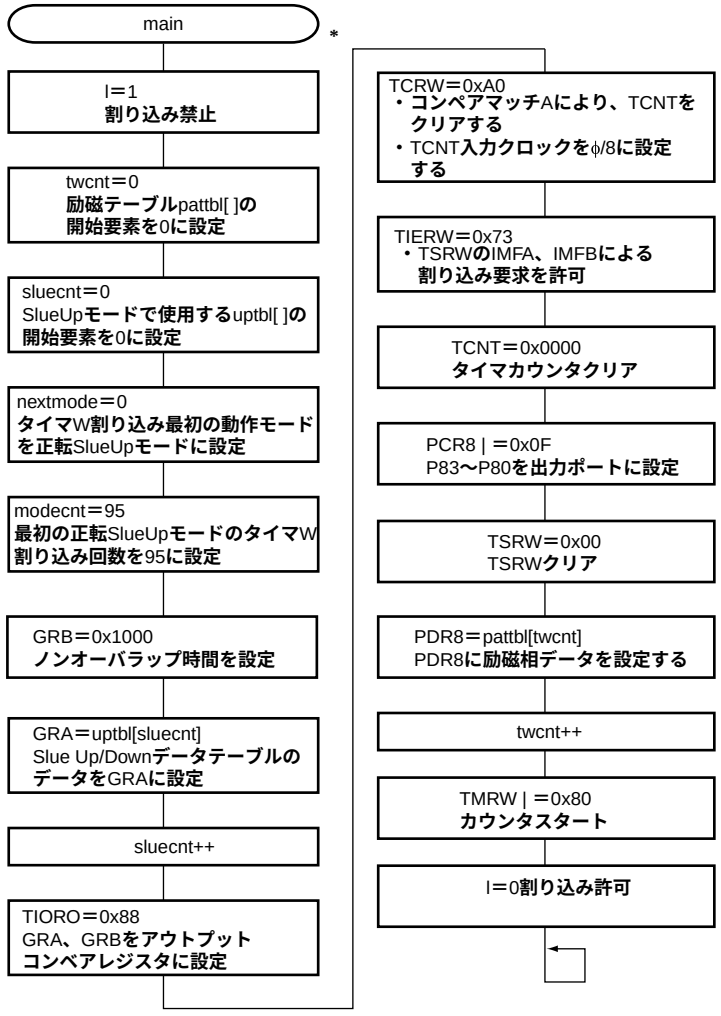
  0x2C24,0x2A94,0x2A7C,0x2A64,0x2A4C,0x2A34,0x2A1C,0x2A00,

  }

  SlueUp、SlueDown 動作時の GRA 割り込みで uptbl[]を順次 GRA に書き込む。書き込みは、ステッピングモータが 1 回転（48 ステップ）するまで続ける。

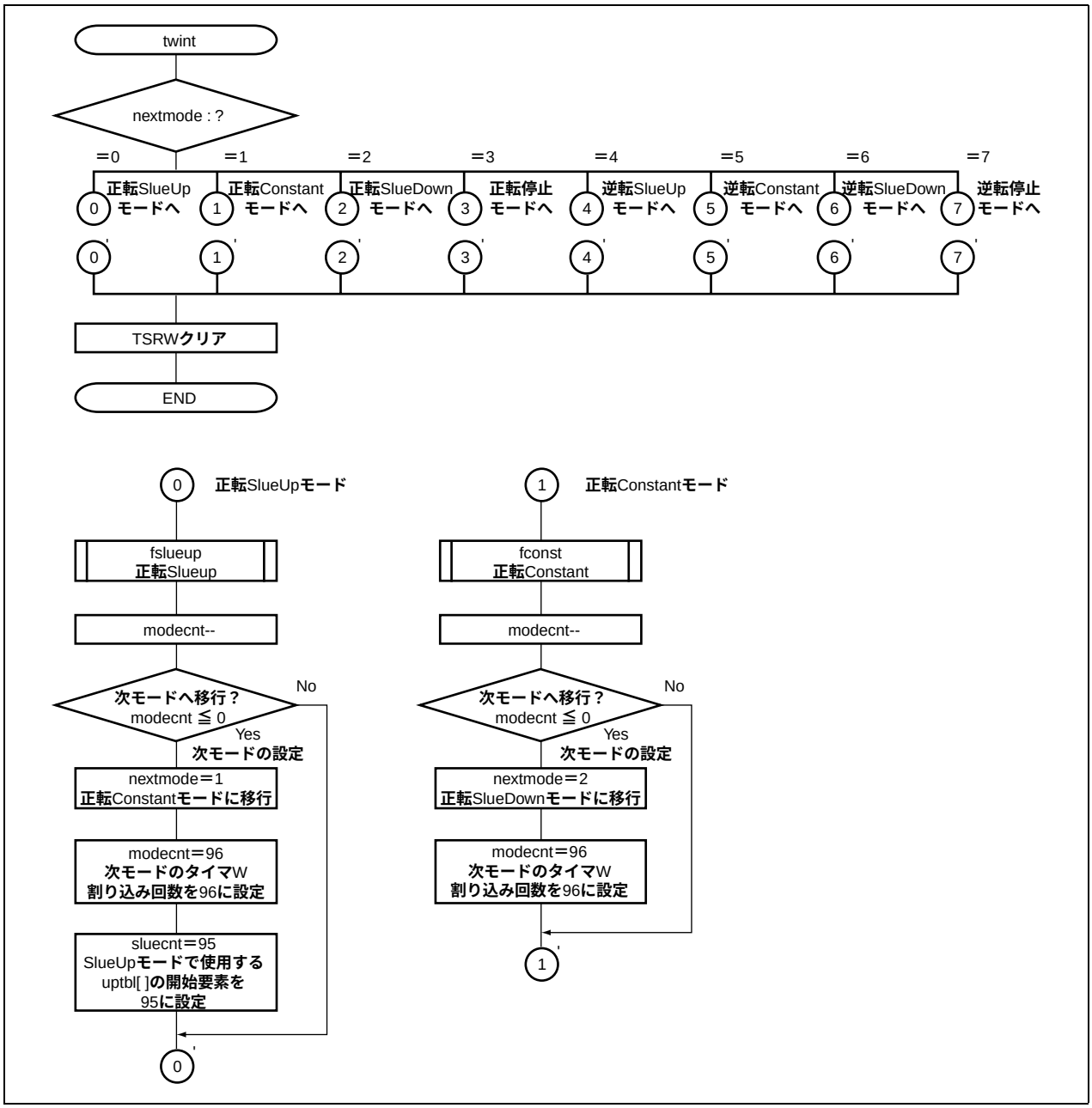
6. フローチャート

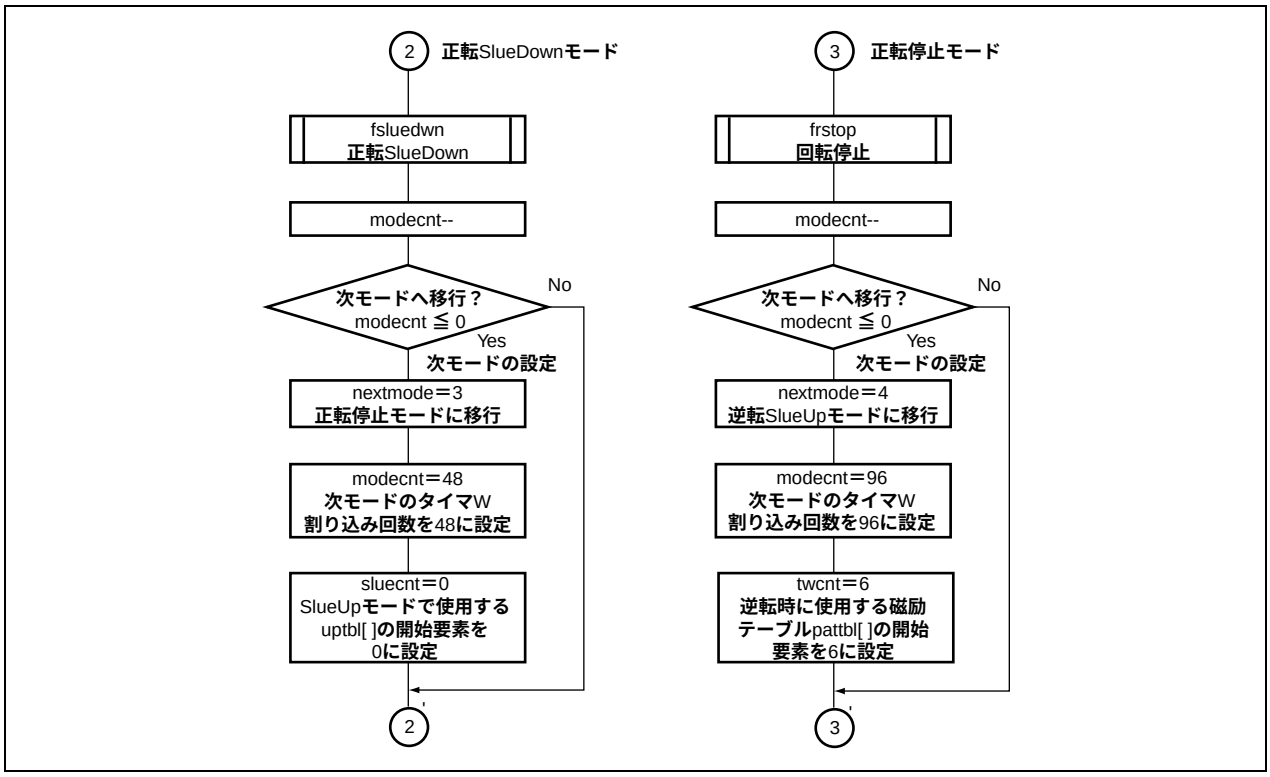
（a）main 関数

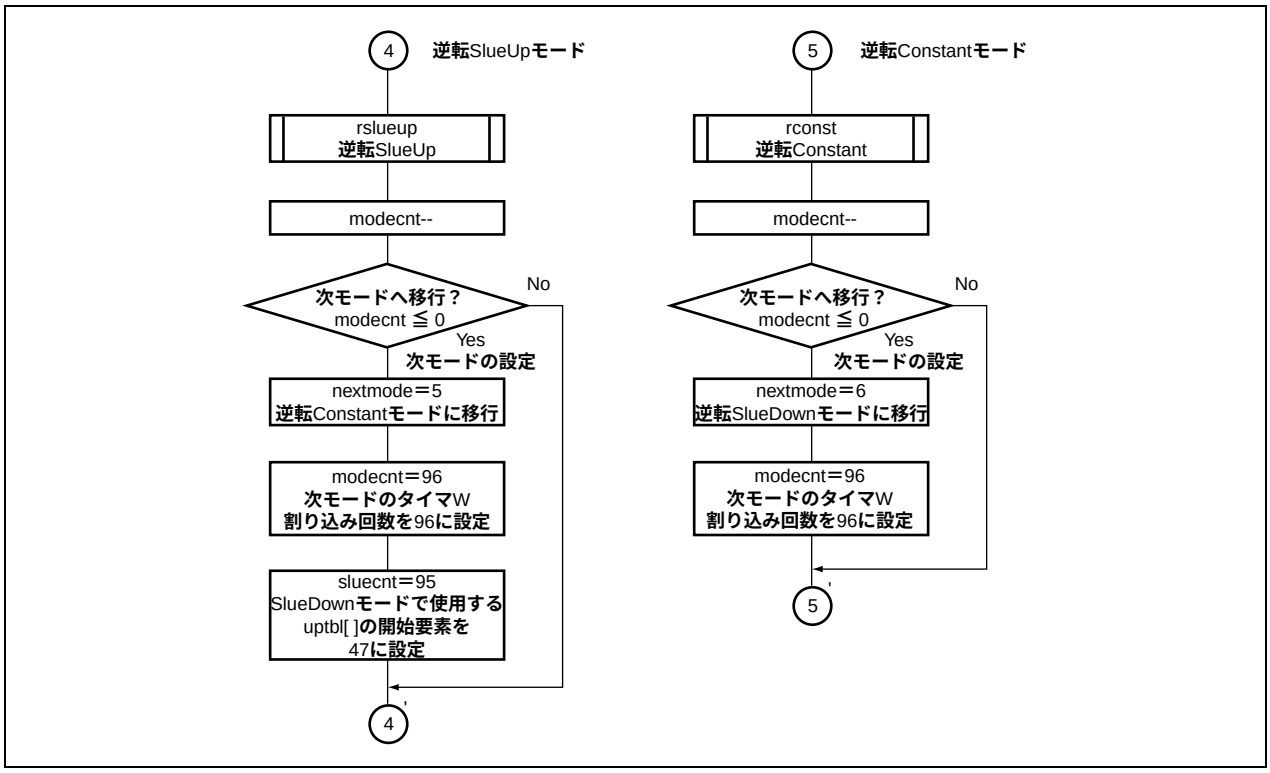


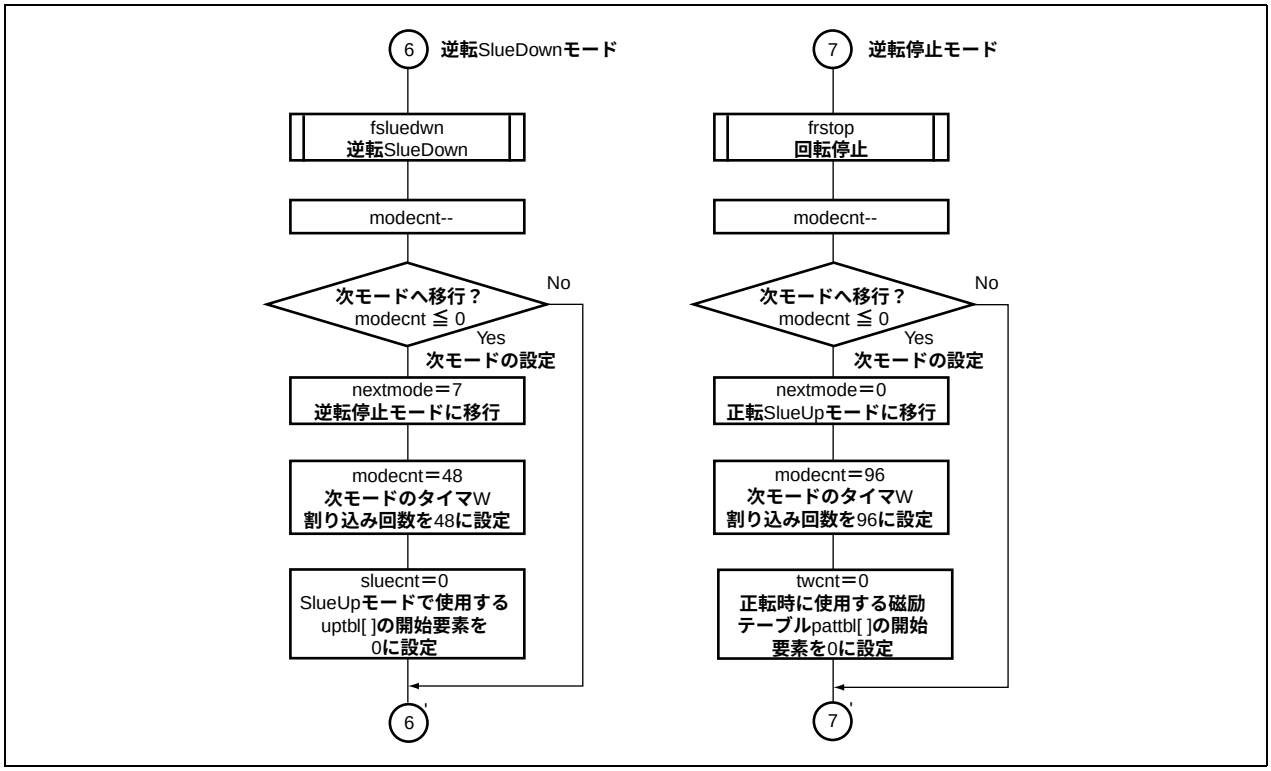
【注】＊ 本例ではスタックポインタはINT.SRC（アセンブリ言語）で設定してあります。

(b) タイマ W 割り込み

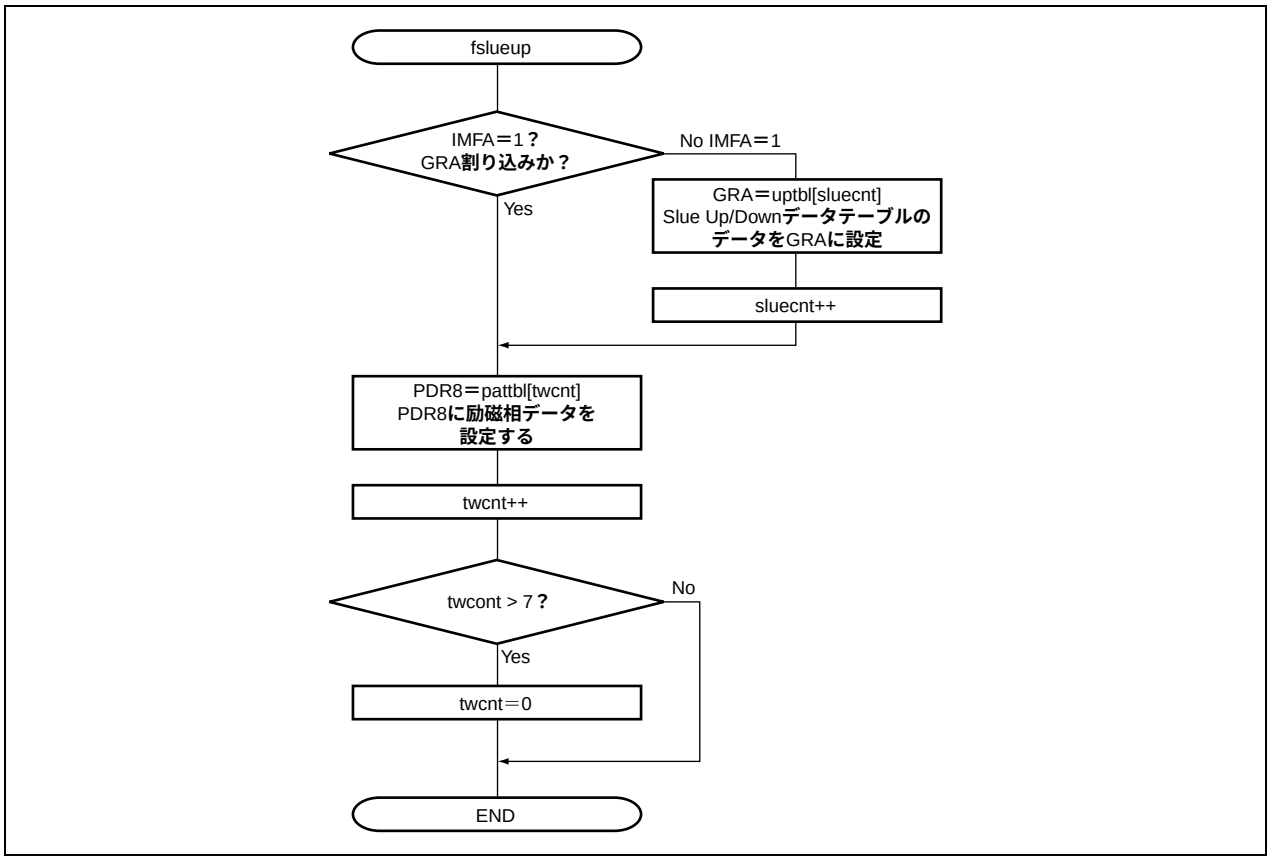

twint
nextmode : ?
=0
=1
=2
=3
=4
=5
=6
=7
正転SlueUp モードへ
正転Constant モードへ
正転SlueDown モードへ
正転停止 モードへ
逆転SlueUp モードへ
逆転Constant モードへ
逆転SlueDown モードへ
逆転停止 モードへ
TSRWクリア
END
正転SlueUpモード
fslueup 正転Slueup
modecnt--
次モードへ移行？ modecnt ≦ 0
No
Yes
次モードの設定
nextmode＝1 正転Constantモードに移行
modecnt＝96 次モードのタイマW 割り込み回数を96に設定
sluecnt＝95 SlueUpモードで使用する uptbl[ ]の開始要素を 95に設定
正転Constantモード
fconst 正転Constant
modecnt--
次モードへ移行？ modecnt ≦ 0
No
Yes
次モードの設定
nextmode＝2 正転SlueDownモードに移行
modecnt＝96 次モードのタイマW 割り込み回数を96に設定


2
正転SlueDownモード
fsluedwn
正転SlueDown
modecnt--
次モードへ移行？
modecnt ≦ 0
No
Yes
次モードの設定
nextmode＝3
正転停止モードに移行
modecnt＝48
次モードのタイマW
割り込み回数を48に設定
sluecnt＝0
SlueUpモードで使用する
uptbl[ ]の開始要素を
0に設定
2
3
正転停止モード
frstop
回転停止
modecnt--
次モードへ移行？
modecnt ≦ 0
No
Yes
次モードの設定
nextmode＝4
逆転SlueUpモードに移行
modecnt＝96
次モードのタイマW
割り込み回数を96に設定
twcnt＝6
逆転時に使用する磁励
テーブルpattbl[ ]の開始
要素を6に設定
3

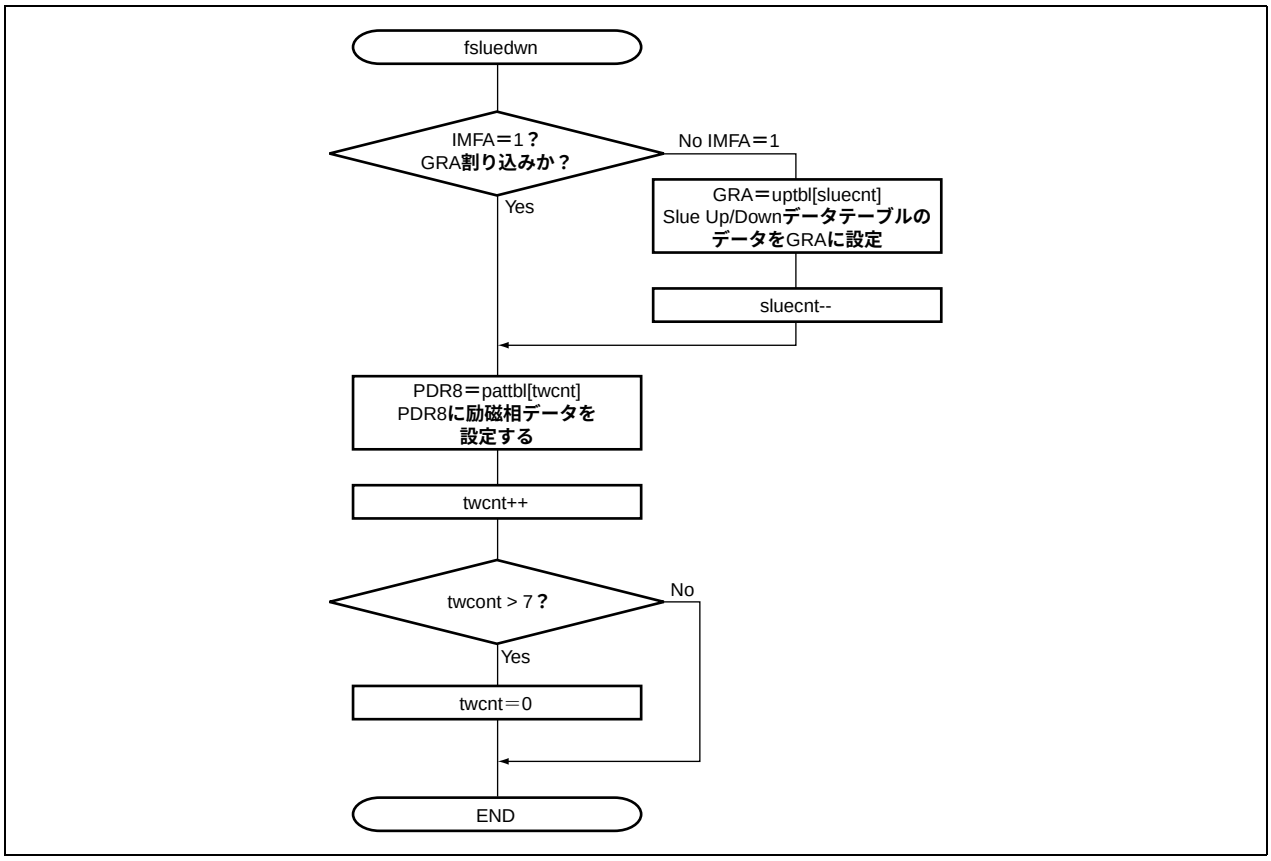

4
逆転SlueUpモード
rslueup
逆転SlueUp
modecnt--
次モードへ移行？
modecnt ≦ 0
No
Yes
次モードの設定
nextmode＝5
逆転Constantモードに移行
modecnt＝96
次モードのタイマW
割り込み回数を96に設定
sluecnt＝95
SlueDownモードで使用する
uptbl[ ]の開始要素を
47に設定
4
5
逆転Constantモード
rconst
逆転Constant
modecnt--
次モードへ移行？
modecnt ≦ 0
No
Yes
次モードの設定
nextmode＝6
逆転SlueDownモードに移行
modecnt＝96
次モードのタイマW
割り込み回数を96に設定
5


6
逆転SlueDownモード
fsluedwn
逆転SlueDown
modecnt--
次モードへ移行？
modecnt ≦ 0
No
Yes
次モードの設定
nextmode＝7
逆転停止モードに移行
modecnt＝48
次モードのタイマW
割り込み回数を48に設定
sluecnt＝0
SlueUpモードで使用する
uptbl[ ]の開始要素を
0に設定
6'
7
逆転停止モード
frstop
回転停止
modecnt--
次モードへ移行？
modecnt ≦ 0
No
Yes
次モードの設定
nextmode＝0
正転SlueUpモードに移行
modecnt＝96
次モードのタイマW
割り込み回数を96に設定
twcnt＝0
正転時に使用する磁励
テーブルpattbl[ ]の開始
要素を0に設定
7'

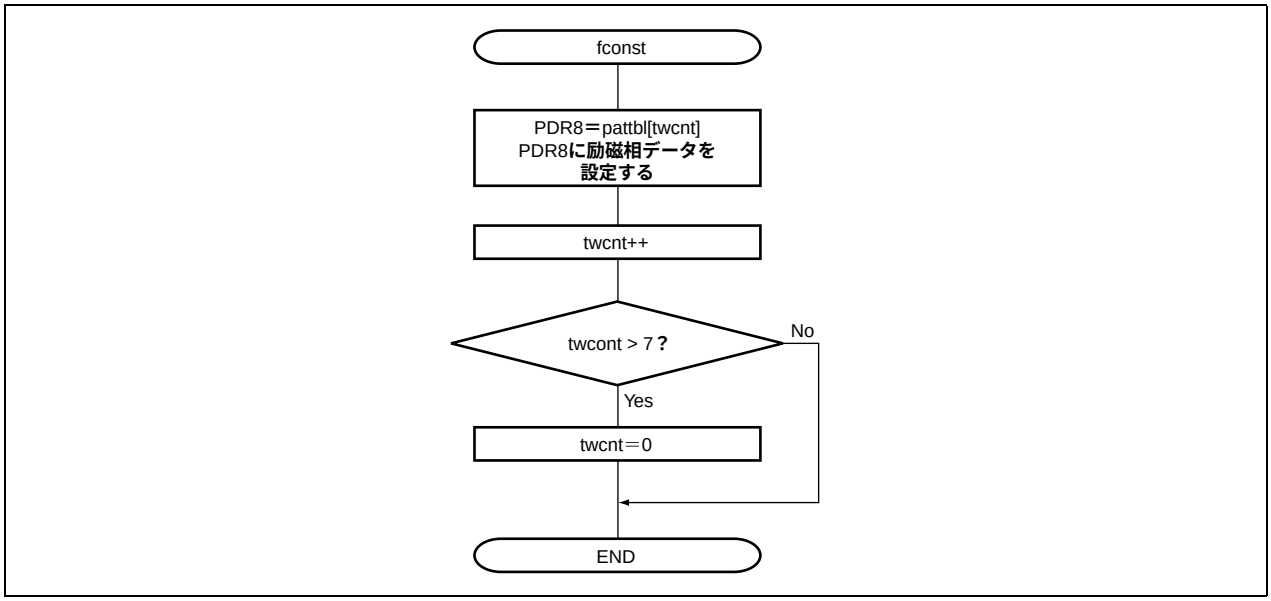
（c）正転時 Slue UP 制御



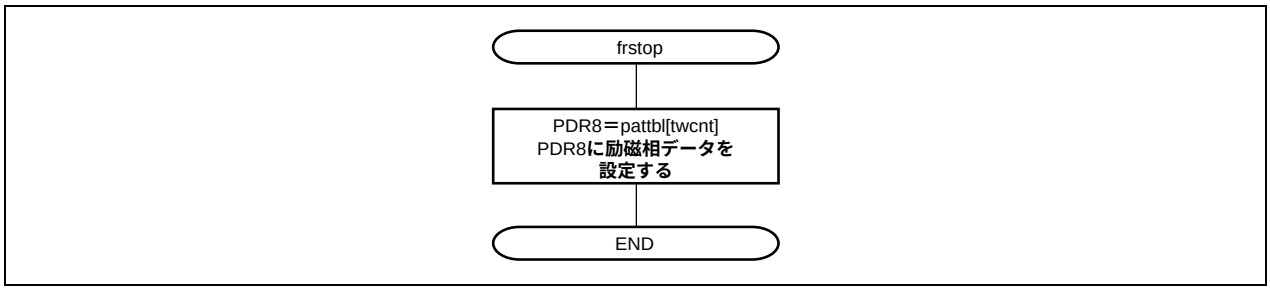
（d）正転時 Slue Down 制御



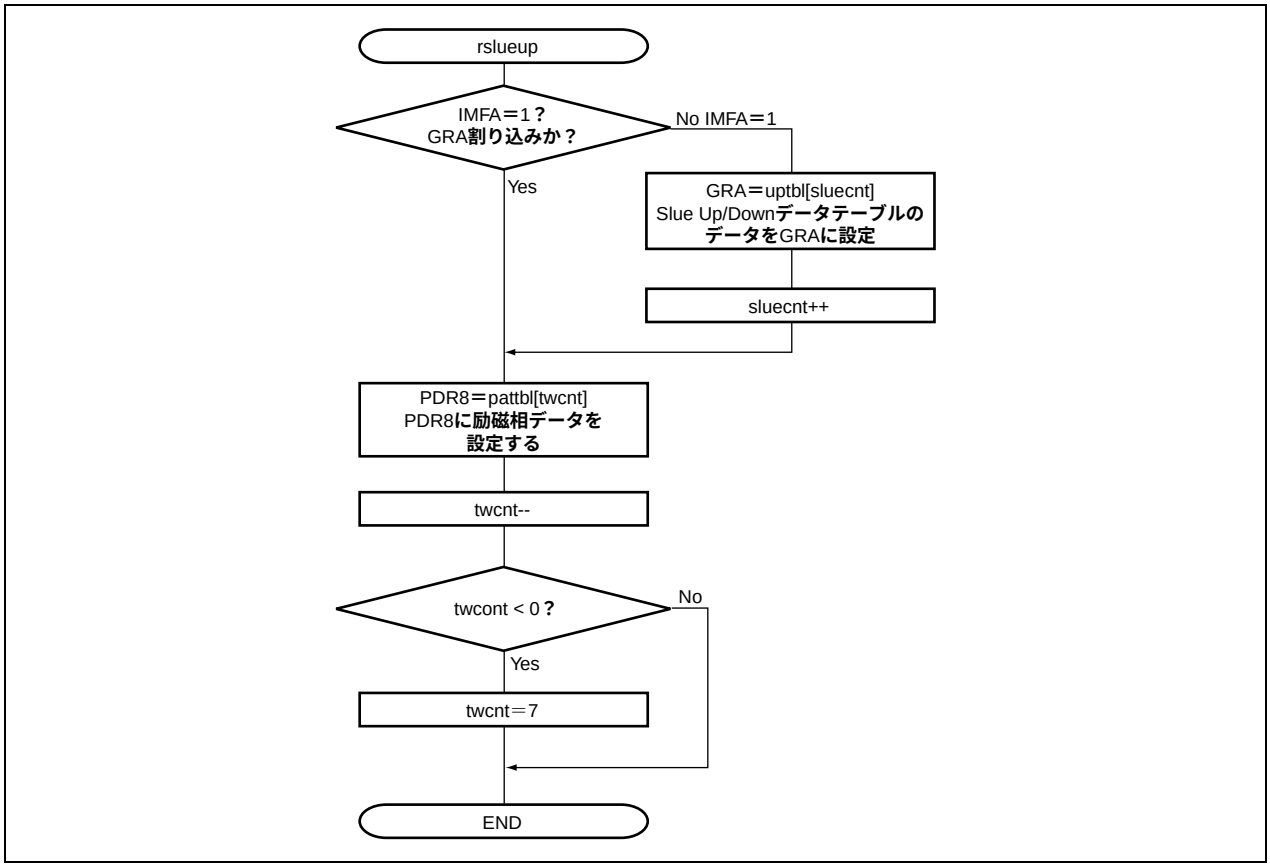
（e）正転時 Constant 制御



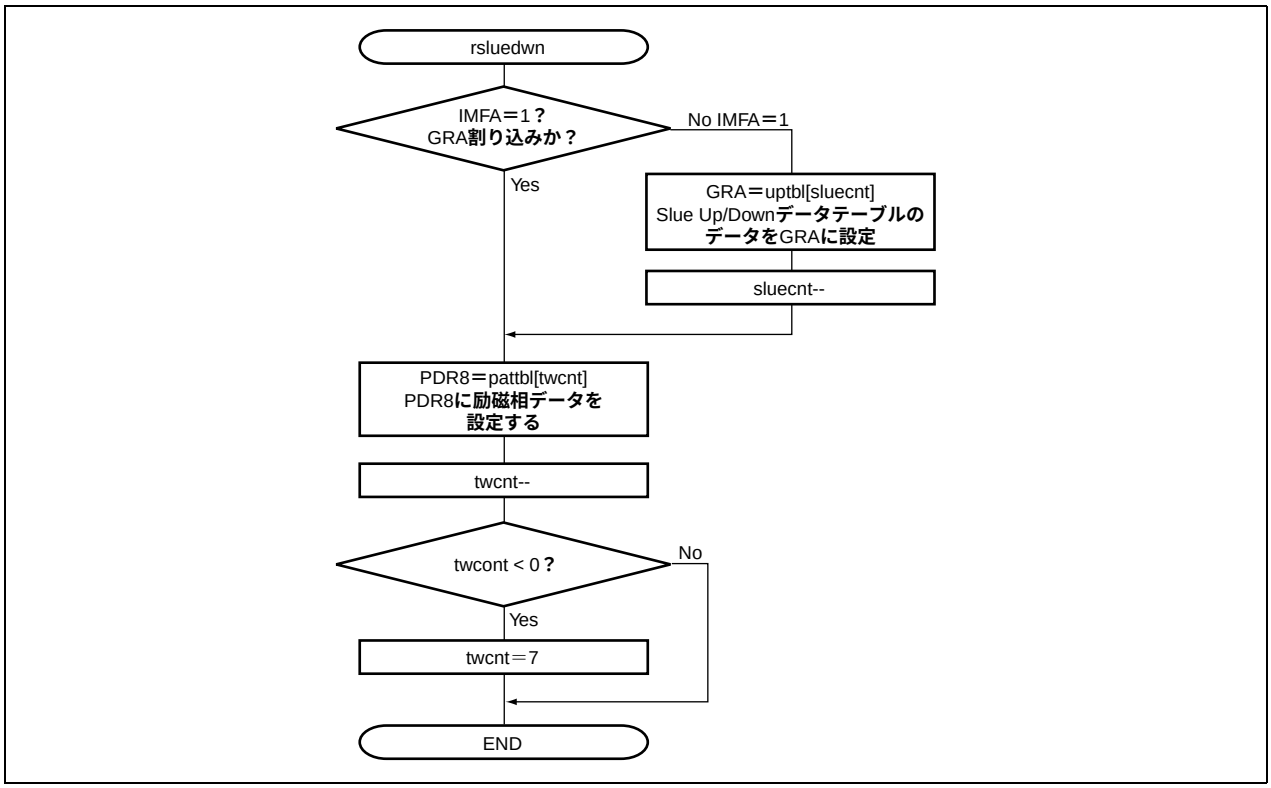
（f）停止制御



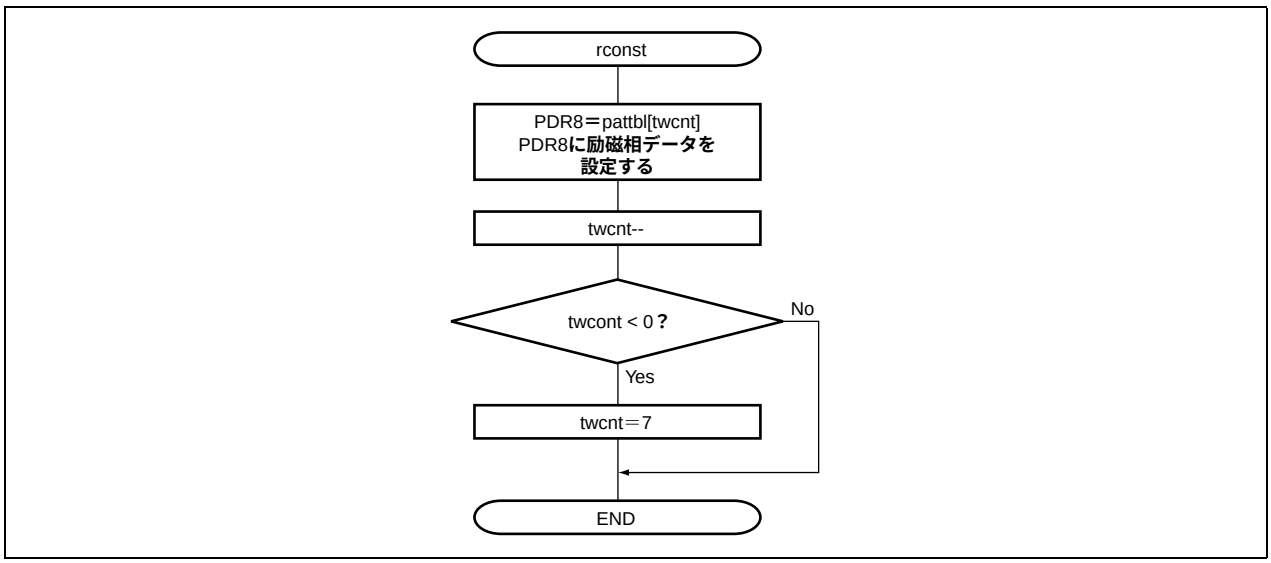
（g）逆転時 Slue UP 制御



（h）逆転時 Slue Down 制御



（i）逆転時 Constant 制御



7. プログラムリスト

INIT.SRC （プログラムリスト）

```
                .EXPORT             _INIT
                .IMPORT             _main
 ;
                .SECTION    P,CODE
 _INIT:
                MOV.W               #H'FF80,R7
                LDC.B               #B'10000000,CCR
                JMP                 @_main
 ;
                .END
```

```
/****************************************************************************************************/
/*                                                                                                  */
/*  H8/300HN Series -H8/3664-                                                                       */
/*  Application Note                                                                                */
/*                                                                                                  */
/*  'Stepping Motor                                                                                 */
/*                                                                                                  */
/*  Function                                                                                        */
/*  : Timer W Output Compare                                                                        */
/*                                                                                                  */
/*  External Clock : 16MHz                                                                          */
/*  Internal Clock : 16MHz                                                                          */
/*  Sub-Clock      : 32.768kHz                                                                      */
/*                                                                                                  */
/****************************************************************************************************/

#include   <machine.h>

/****************************************************************************************************/
/*  Symbol Definition                                                                               */
/****************************************************************************************************/
struct BIT {
   unsigned char   b7:1;      /* bit7                                                               */
   unsigned char   b6:1;      /* bit6                                                               */
   unsigned char   b5:1;      /* bit5                                                               */
   unsigned char   b4:1;      /* bit4                                                               */
   unsigned char   b3:1;      /* bit3                                                               */
   unsigned char   b2:1;      /* bit2                                                               */
```

```
    unsigned char   b1:1;      /* bit1                                                    */
    unsigned char   b0:1;      /* bit0                                                    */
};

#define    TMRW       *(volatile unsigned char *)0xFF80   /* Timer Mode Register W                    */
#define    TCRW       *(volatile unsigned char *)0xFF81   /* Timer Control Register W                 */
#define    TCRW_BIT    (*(struct BIT *)0xFF81)            /* Timer Control Register W                 */
#define    CCLR       TCRW_BIT.b7                        /* Counter Clear                            */
#define    CKS2       TCRW_BIT.b6                        /* Clock Select 2                           */
#define    CKS1       TCRW_BIT.b5                        /* Clock Select 1                           */
#define    CKS0       TCRW_BIT.b4                        /* Clock Select 0                           */
#define    TIERW      *(volatile unsigned char *)0xFF82   /* Timer Interrupt Enable Register          */
#define    TIERW_BIT   (*(struct BIT *)0xFF82)           /* Timer Interrupt Enable Register          */
#define    IMIEB      TIERW_BIT.b1                       /* Output Compare Interrupt B Enable        */
#define    IMIEA      TIERW_BIT.b0                       /* Output Compare Interrupt A Enable        */
#define    TSRW       *(volatile unsigned char *)0xFF83   /* Timer Status Register W                  */
#define    TSRW_BIT    (*(struct BIT *)0xFF83)           /* Timer Status Register W                  */
#define    IMFB       TSRW_BIT.b1                        /* Output Compare Flag B                    */
#define    IMFA       TSRW_BIT.b0                        /* Output Compare Flag A                    */
#define    TIOR0      *(volatile unsigned char *)0xFF84   /* Timer I/O Control Register 0             */
#define    TIOR0_BIT   (*(struct BIT *)0xFF84)           /* Timer I/O Control Register 0             */
#define    IOB2       TIOR0_BIT.b6                       /* I/O Control Register B2                  */
#define    IOB1       TIOR0_BIT.b5                       /* I/O Control Register B1                  */
#define    IOB0       TIOR0_BIT.b4                       /* I/O Control Register B0                  */
#define    TCNT       *(volatile unsigned int *)0xFF86    /* Time Counter                             */
#define    GRA        *(volatile unsigned int *)0xFF88    /* General Register A                       */
#define    GRB        *(volatile unsigned int *)0xFF8A    /* General Register B                       */
#define    PDR8       *(volatile unsigned int *)0xFFDB    /* Port Data Register 8                     */
#define    PCR8       *(volatile unsigned int *)0xFFEB    /* Port Control Register 8                  */

#pragma interrupt  (twint)
/************************************************************************************************/
/*  Function define                                                                             */
/************************************************************************************************/
extern void INIT ( void );                              /* SP Set                                   */
void       main ( void );
void       twint ( void );

void       fslueup ( void );
void       fsluedwn ( void );
void       fconst ( void );
void       frstop ( void );
void       rslueup ( void );
```

```
void       rsluedwn ( void );
void       rconst ( void );

/************************************************************************************************************/
char   twcnt,sluecnt,nextmode;
short  modecnt;

/************************************************************************************************************/
#pragma section OUTDT
unsigned char   pattbl[8] = {                          /* Stepping Motor Output Pattern Table         */
   0x08,0x0C,0x04,0x06,0x02,0x03,0x01,0x09,
};

unsigned short  uptbl[48] = {                          /* Stepping Motor Output Pattern Table         */
   0xFFFF,0xF000,0xE09C,0xD034,0xC670,0xBC48,0xB1BC,0xAA50,0xA21C,0x98BC,
   0x9218,0x8D68,0x88B8,0x8408,0x7F58,0x7AA8,0x75F8,0x7148,0x6C98,0x6720,
   0x6338,0x5E24,0x5B04,0x56B8,0x5398,0x5140,0x4D58,0x4970,0x4650,0x4330,
   0x4010,0x3CF0,0x3AFC,0x3908,0x3714,0x3520,0x332C,0x3138,0x2F44,0x2DB4,
   0x2C24,0x2A94,0x2A7C,0x2A64,0x2A4C,0x2A34,0x2A1C,0x2A00,
};

/*   0xFFFF,0xE000,0xD000,0xD000,0xC000,0xC000,0xB000,0xB000,0xA000,0xA000,
/*   0x9000,0x9000,0x9000,0x8000,0x8000,0x8000,0x8000,0x8000,0x7000,0x7000,
/*   0x7000,0x7000,0x6000,0x6000,0x6000,0x6000,0x5000,0x5000,0x5000,0x5000,
/*   0x5000,0x4000,0x4000,0x4000,0x4000,0x4000,0x4000,0x3000,0x3000,0x3000,
/*   0x3000,0x3000,0x3000,0x3000,0x3000,0x2A00,0x2A00,0x2A00,
*/

/************************************************************************************************************/
/*  Vector Address                                                                                          */
/************************************************************************************************************/
#pragma section    V1                                  /* VECTOR SECTION SET                          */
void (*const VEC_TBL1[])(void) = {                     /* 0x00 - 0x0f                                 */
   INIT                                                /* 00 Reset                                    */
};
#pragma section    V2                                  /* VECTOR SECTION SET                          */
void (*const VEC_TBL2[])(void) = {
   twint                                               /* 2A Timer W Interrupt                        */
};

#pragma section                                        /* P                                           */
```

```
/*****************************************************************************************************/
/*  Main Program                                                                                     */
/*****************************************************************************************************/
void main ( void )
{
    unsigned char tmp;

    set_imask_ccr(1);                           /* Disable interrupts                           */

    twcnt = 0;                                  /* Output Pattern table counter set             */
    sluecnt = 0;                                /* Slue Up/Down table counter set               */
    nextmode = 0;
    modecnt = 95;                               /* Motor Slue mode countset "96"                */

    GRB = 0x1000;                               /* Initialize GRB                               */
    GRA = uptbl[sluecnt];                       /* Initialize GRA                               */
    sluecnt++;

    TIOR0 = 0x88;                               /* Initialize Output Compare Function           */
    TCRW = 0xB0;                                /* Initialize TCNT Input Clock Period           */
    TIERW = 0x73;                               /* Initialize IMIEA/IMIEB Interrupt Enable      */

    TCNT = 0x0000;                              /* Initialize TCNT                              */

    PCR8 |= 0x0F;                               /* Port8 Output                                 */

    tmp = TSRW;                                 /* TSRW Clear                                   */
    TSRW = 0x00;

    PDR8 = pattbl[twcnt];                       /* PDR8 Set Output Pattern                      */
    twcnt++;

    TMRW |= 0x80;                               /* Initialize timer Mode Register               */
    set_imask_ccr(0);                           /* Interrupt Enable                             */

    while(1);
}
```

```
/*****************************************************************************************************/
/*  Timer W Interrupt                                                                                */
/*****************************************************************************************************/
void twint ( void )
{
   unsigned char tmp;

   switch(nextmode){
      case 0:
         fslueup();                                /* Forward Slue Up                          */
         modecnt--;
         if(modecnt <= 0){                         /* Next mode?                               */
            nextmode = 1;                          /* nextmode = 1 Constant Speed              */
            modecnt = 96;                          /* Next mode countset "96"                  */
            sluecnt = 47;                          /* Slue Up/Down table counter set           */
         }
         break;

      case 1:
         fconst();                                 /* Constant Speed                           */
         modecnt--;
         if(modecnt <= 0){                         /* Nextmode?                                */
            nextmode = 2;                          /* nextmode = 2 Forward Slue Down           */
            modecnt = 96;                          /* Nextmode countset "96"                   */
         }
         break;

      case 2:
         fsluedwn();                               /* Forward Slue Down                        */
         modecnt--;
         if(modecnt <= 0){                         /* Next mode?                               */
            nextmode = 3;                          /* nextmode = 3 Slue Stop                   */
            modecnt = 48;                          /* Next mode countset "48"                  */
            sluecnt = 0;                           /* Slue Up/Down table counter set           */
         }
         break;

      case 3:
         frstop();                                 /* Slue Stop                                */
         modecnt--;
         if(modecnt <= 0){                         /* Next mode?                               */
            nextmode = 4;                          /* nextmode = 4 Reverse Slue Up             */
            modecnt = 96;                          /* Next mode countset "96"                  */
```

```
twcnt = 6;                                  /* Output Pattern table counter set          */
          }
          break;

       case 4:
          rslueup();                        /* Reverse Slue Up                           */
          modecnt--;
          if(modecnt <= 0){                 /* Next mode?                                */
             nextmode = 5;                  /* nextmode = 5 Constant Speed               */
             modecnt = 96;                  /* Next mode countset "96"                   */
             sluecnt = 47;                  /* Slue Up/Down table counter set            */
          }
          break;

       case 5:
          rconst();                         /* Constant Speed                            */
          modecnt--;
          if(modecnt <= 0){                 /* Next mode?                                */
             nextmode = 6;                  /* nextmode = 6 Reverse Slue Down            */
             modecnt = 96;                  /* Next mode countset "96"                   */
          }
          break;

       case 6:
          rsluedwn();                       /* Reverse Slue Down                         */
          modecnt--;
          if(modecnt <= 0){                 /* Next mode?                                */
             nextmode = 7;                  /* nextmode = 7 Slue Stop                    */
             modecnt = 48;                  /* Next mode countset "48"                   */
             sluecnt = 0;                   /* Slue Up/Down table counter set            */
          }
          break;

       case 7:
          frstop();                         /* Slue Stop                                 */
          modecnt--;
          if(modecnt <= 0){                 /* Next mode?                                */
             nextmode = 0;                  /* nextmode = 0 Forward Slue Up              */
             modecnt = 96;                  /* Next mode countset "96"                   */
             twcnt = 0;                     /* Output Pattern table counter set          */
          }
          break;
    }
```

```
    tmp = TSRW;
    TSRW = 0x00;
}

/*****************************************************************************************************************/
/*  Forward Slue Up                                                                                              */
/*****************************************************************************************************************/
void fslueup ( void )
{
    if(IMFA == 1){
        GRA = uptbl[sluecnt];                           /* GRA Set Slue Up/Down table                  */
        sluecnt++;
    }

    PDR8 = pattbl[twcnt];                               /* PDR8 Set Output Pattern                     */
    twcnt++;

    if(twcnt>7)
        twcnt = 0;
}

/*****************************************************************************************************************/
/*  Forward Slue Down                                                                                            */
/*****************************************************************************************************************/
void fsluedwn ( void )
{
    if(IMFA == 1){
        GRA = uptbl[sluecnt];                           /* GRA Set Slue Up/Down table                  */
        sluecnt--;
    }

    PDR8 = pattbl[twcnt];                               /* PDR8 Set Output Pattern                     */
    twcnt++;

    if(twcnt>7)
        twcnt = 0;
}
```

```
/*****************************************************************************************************/
/*  Forward Constant Speed                                                                           */
/*****************************************************************************************************/
void fconst ( void )
{
   PDR8 = pattbl[twcnt];                          /* PDR8 Set Output Pattern                        */
   twcnt++;

   if(twcnt>7)
      twcnt = 0;
}

/*****************************************************************************************************/
/*  Slue/Reverse Stop                                                                                */
/*****************************************************************************************************/
void frstop ( void )
{
   PDR8 = pattbl[twcnt];                          /* PDR8 Set Output Pattern                        */
}

/*****************************************************************************************************/
/*  Reverse Slue Up                                                                                  */
/*****************************************************************************************************/
void rslueup ( void )
{
   if(IMFA == 1){
      GRA = uptbl[sluecnt];                       /* GRA Set Slue Up/Down table                     */
      sluecnt++;
   }

   PDR8 = pattbl[twcnt];                          /* PDR8 Set Output Pattern                        */
   twcnt--;

   if(twcnt < 0)
      twcnt = 7;
}
```

```
/*****************************************************************************************************/
/*  Reverse Slue Down                                                                                */
/*****************************************************************************************************/
void rsluedwn ( void )
{
   if(IMFA == 1){
      GRA = uptbl[sluecnt];                        /* GRA Set Slue Up/Down table                    */
      sluecnt--;
   }

   PDR8 = pattbl[twcnt];                           /* PDR8 Set Output Pattern                       */
   twcnt--;

   if(twcnt < 0)
      twcnt = 7;
}

/*****************************************************************************************************/
/*  Reverse Constant Speed                                                                           */
/*****************************************************************************************************/
void rconst ( void )
{
   PDR8 = pattbl[twcnt];                           /* PDR8 Set Output Pattern                       */
   twcnt--;

   if(twcnt < 0)
      twcnt = 7;
}
```

リンクアドレス指定

セクション名	アドレス
CV1	0x0000
CV2	0x002A
P	0x0100
C	0x0500
DOUTDT	0x0510
B	0xFB80