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。

H8/300H Tiny シリーズ

1-2 相励磁方式ステッピングモータ

要旨

H8/3687 の内蔵機能のうち、P63～P60 とタイマ Z 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、1-2 相励磁方式で制御します。

動作確認デバイス

H8/3687

目次

1. 仕様

- H8/3687 の内蔵機能のうち、P63～P60 とタイマ Z 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、1-2 相励磁方式で制御します。
- 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。
- 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。
- 2 相ステッピングモータ制御の接続図を図 1に示します。

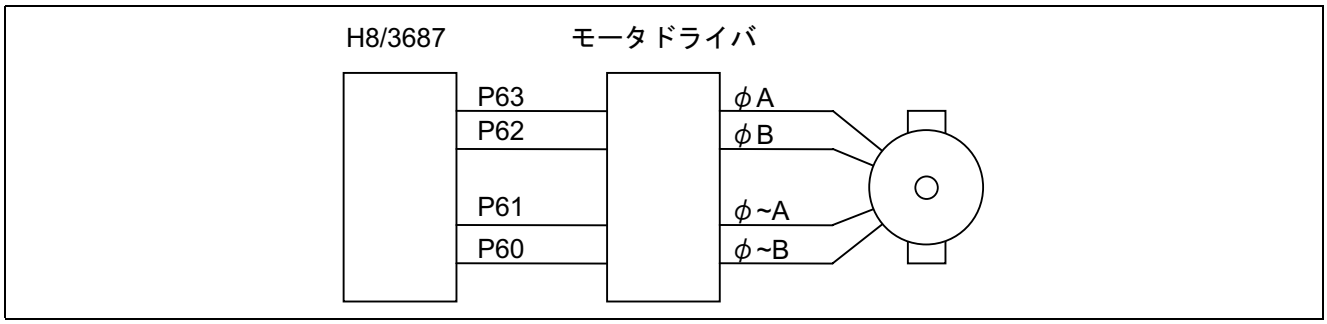


図1　2 相ステッピングモータ制御の接続図

2. 使用機能説明

2.1 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 1に KP6P8-701 の標準仕様を示します。

表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

2.2 **ステッピングモータ制御における H8/3687 の使用機能について説明します。図 2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。**

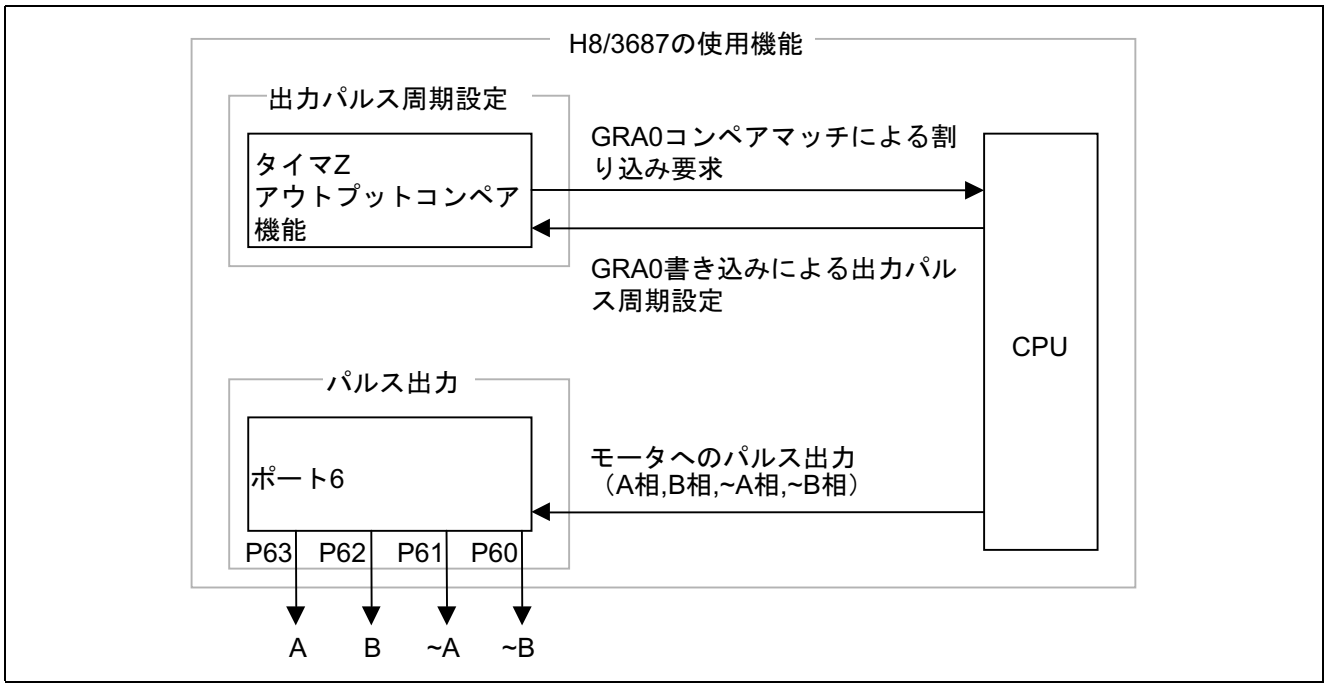


図2　H8/3687 の使用機能

2.3 **タイマ Z 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 Z のアウトプットコンペア機能を使用します。タイマ Z のブロック図を図 3に示します。以下にタイマ Z 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。
- タイマコントロールレジスタ 0（TCR0）
  TCNT0 の入力クロック、クリア方法を選択します。本タスク例では、入力クロックをφ/4、クロックの立ち上がりエッジでカウント、GRA0 のコンペアマッチ／インプットキャプチャで TCNT0 をクリアに設定しています。
- タイマ I/O コントロールレジスタ A0（TIORA0）
  GRA0 を制御します。本タスク例では、GRA0 をアウトプットコンペアレジスタとし、FTIOA0 端子は、出力禁止に設定します。
- タイマステータスレジスタ 0（TSR0）
  タイマ Z の状態を表わします。本タスク例では、GRA0 コンペアマッチ時、インプットキャプチャ／コンペアマッチフラグ A（IMFA）が"1"にセットされます。
- タイマインタラプトイネーブルレジスタ（TIER0）
  各割り込み要求の許可／禁止を制御します。本タスク例では、TSR0 の IMFA、IMFB フラグによる割り込み要求を許可し、それ以外の割り込みは禁止にしています。

- タイマカウンタ 0（TCNT0）
  16 ビットのリード/ライト可能なアップカウンタで、入力する内部クロック/外部クロックによりカウントアップされます。本タスク例では、入力クロックをφ/4、クロックの立ち上がりエッジでカウントします。
- ジェネラルレジスタ A0（GRA0）
  16 ビットのリード/ライト可能なレジスタです。GRA0 の内容は TCNT0 と常に比較されており、両者の値が一致すると TSR0 の IMFA が"1"にセットされます。この時、TIER0 の IMIEA が"1"ならば、CPU に割り込みを要求します。
- タイマスタートレジスタ（TSTR）
  TCNT0、TCNT1 の動作／停止を選択します。本タスク例では、TCNT0 をカウント動作、TCNT1 をカウント停止に設定しています。
- タイマモードレジスタ（TMDR）
  TCNT0、TCNT1 のタイマ同期/独立を選択します。本タスク例では、TCNT0 と TCNT1 は、独立動作に設定しています。

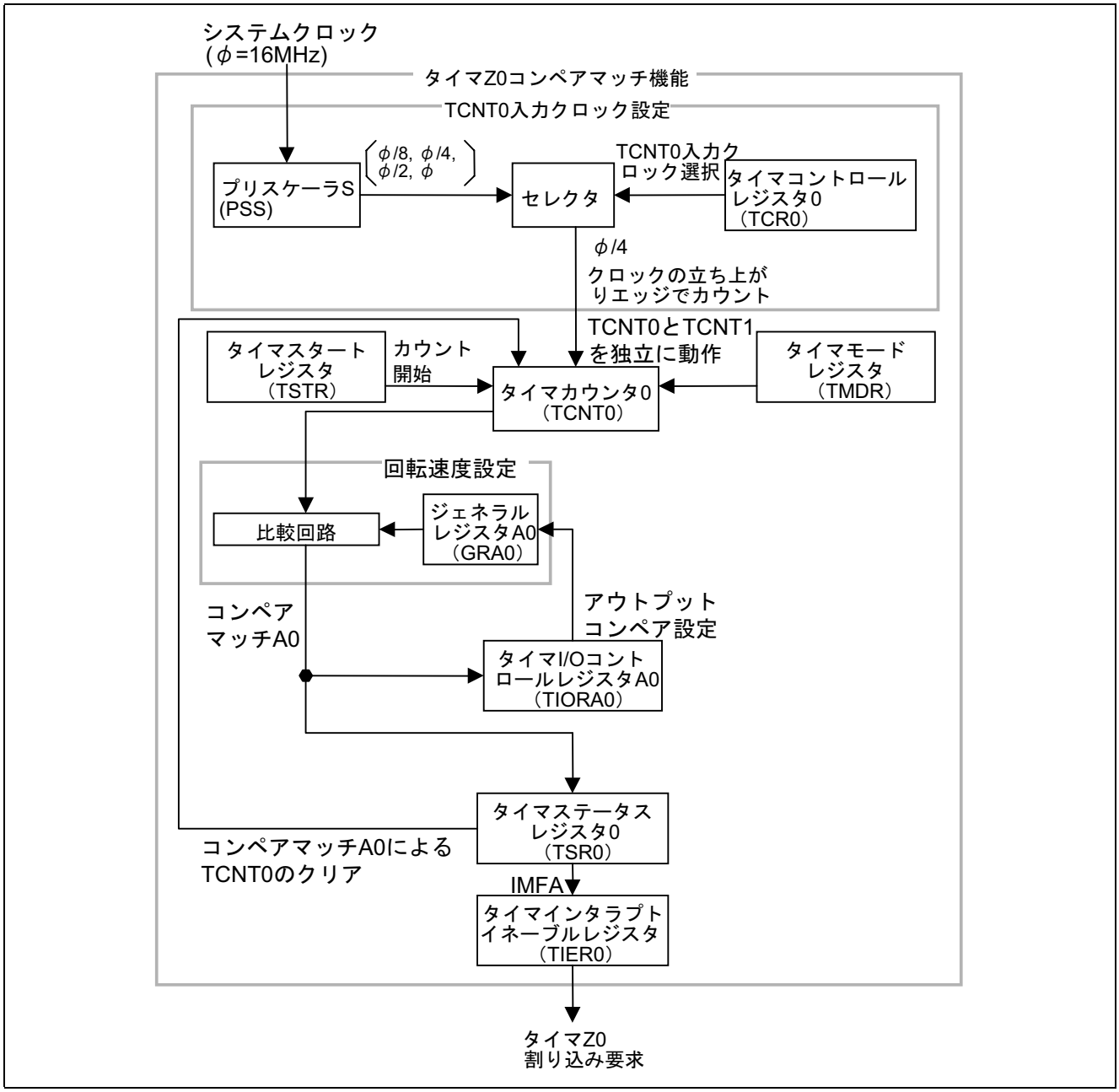


図3　タイマ Z0 のブロック図

2.4　ポート 6 は、8 ビット入出力ポートです。本タスク例では、ポート 6 のうち P63～P60 を使用します。ポート 6 のブロック図を図 4に示します。以下にポート 6 の機能について説明します。

- ポートデータレジスタ 6（PDR6）
  P63～P60 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 6（PCR6）
  P63～P60 を出力端子に設定します。

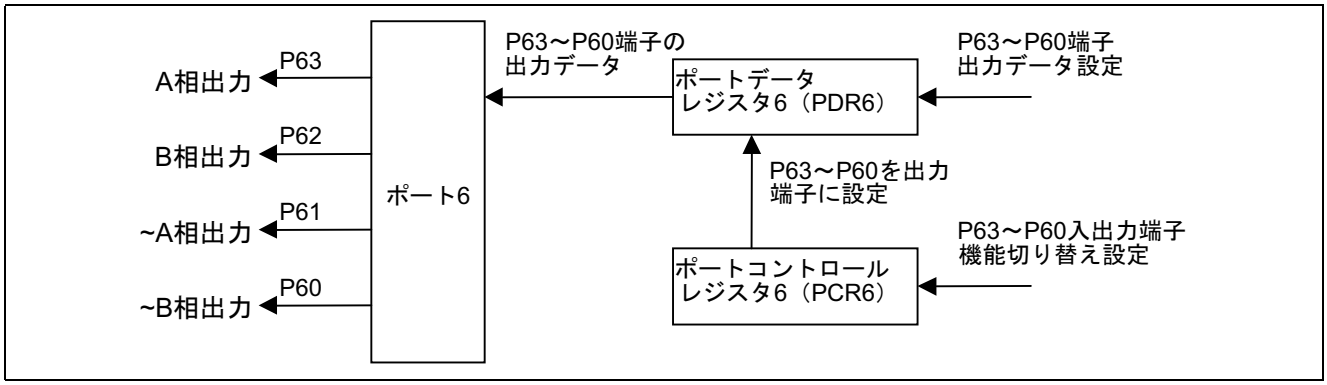


図4　ポート 6 機能のブロック図

2.5　**本タスク例の機能割り付けを表 2に示します。**

表2　機能割り付け

機能	機能割り付け
システムクロック PSS TCNT0	**ステッピングモータを動作させる基準クロック**
TCR0	TCNT0 **動作設定**
TIORA0	**アウトプットコンペアレジスタの設定**
TSR0	**タイマ** Z **の状態を表わす**
TIER0	**各割り込み要求の許可／禁止を制御**
TSTR	TCNT0 **のカウントを開始**
TMDR	TCNT0 **と** TCNT1 **を独立動作に設定**
GRA0	**ステッピングモータ** 1 **ステップ時間の設定**
PDR6 PCR6	**ステッピングモータの励磁相駆動**

3. 動作説明

3.1 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 1-2 相励磁方式で動作させる例を図 5に示します。動作概要は、以下の通りです。

- 図 5のようにパルスが High のとき、対応する相を励磁します。
- まず、A 相を励磁します。このときロータは、A 相に位置します。
- 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 B 相→B,~A 相→~A 相→~A,~B 相→~B 相→~B,A 相の順番に励磁し、ロータを回転させます。
- 逆転動作の場合は、B,A 相→~B 相→~A,~B 相→~A 相→B,~A 相→B 相→A,B 相→ A 相の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。
- 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。

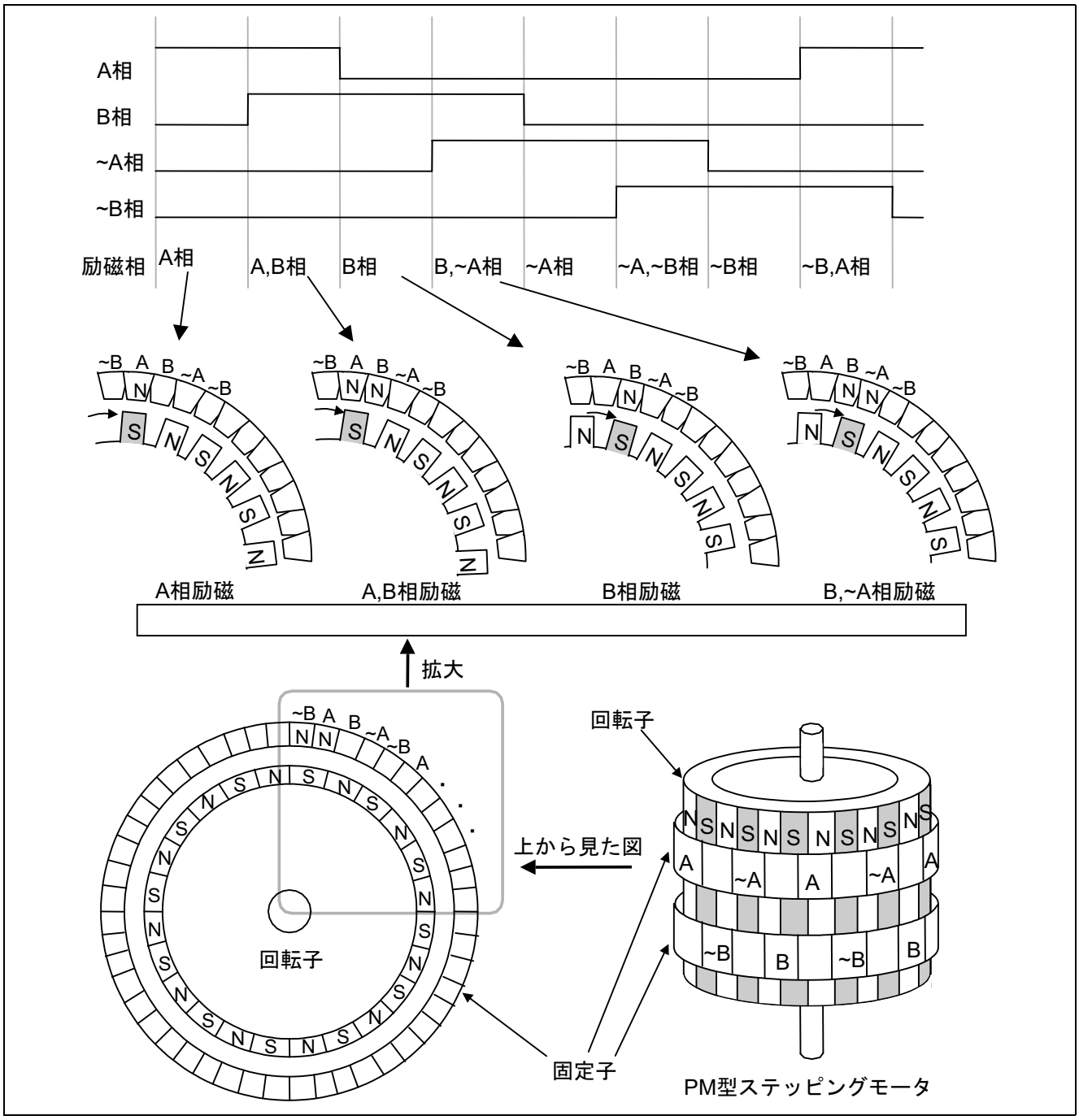


図5　ステッピングモータ動作例

3.2 Slue Up、Slue Down 動作

3.2.1 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。

3.2.2 動作原理を以下に示します。

- 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）
- 一定のパルス周期で設定した量のパルスを出力する。（定速動作）
- 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）

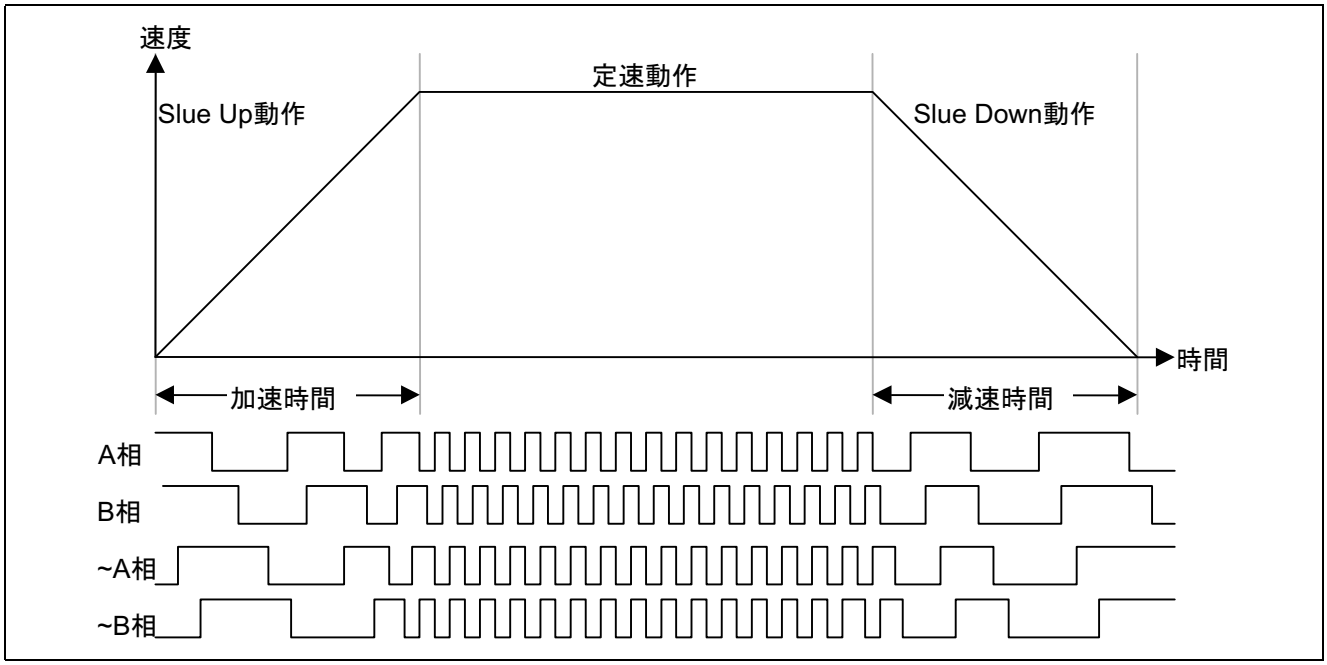


図6 Slue Up、Slue Down 動作例

3.3 ステッピングモータ制御のフローチャートを図 7に示します。

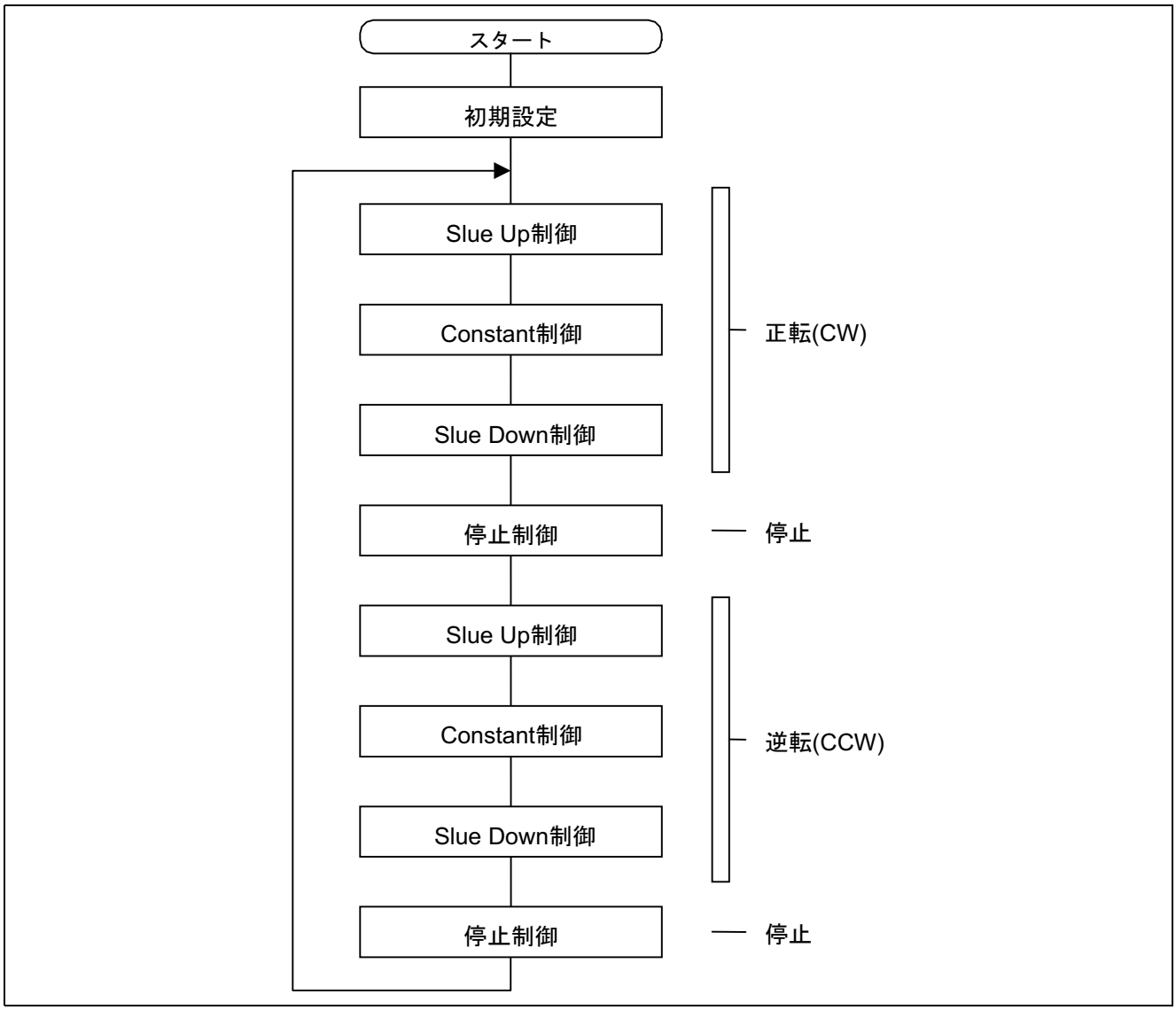


図7 ステッピングモータ制御のフローチャート

3.4 タイマ Z 割り込み時間の計算式

- アウトプットコンペアレジスタ（GRA0）の設定による、タイマ Z 割り込み時間の計算式を以下に示します。

$$\text{タイマ Z 割り込み時間} = \frac{\text{GRA0} + 1}{(\text{システムクロック}\phi / 4)}$$

$$= \frac{\text{GRA0} + 1}{(16\text{MHz} / 4)}$$

$$= \frac{\text{GRA0} + 1}{4} \quad [\mu\text{s}]$$

3.5 正転時 Slue Up 制御の動作原理を図 8に示します。

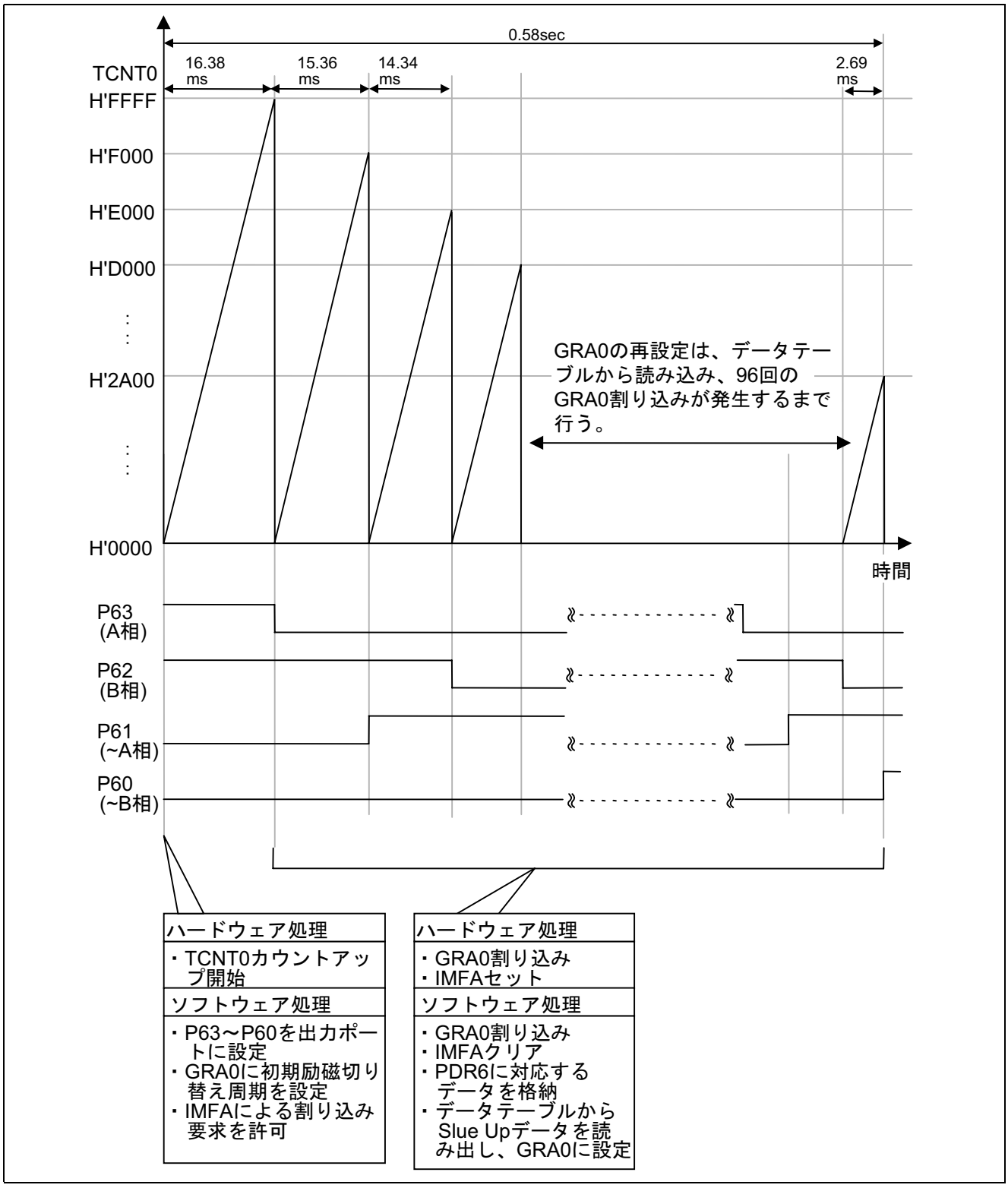


図8 正転時 Slue Up 制御の動作原理

3.6 **正転時** Constant **制御の動作原理を図** 9**に示します。**

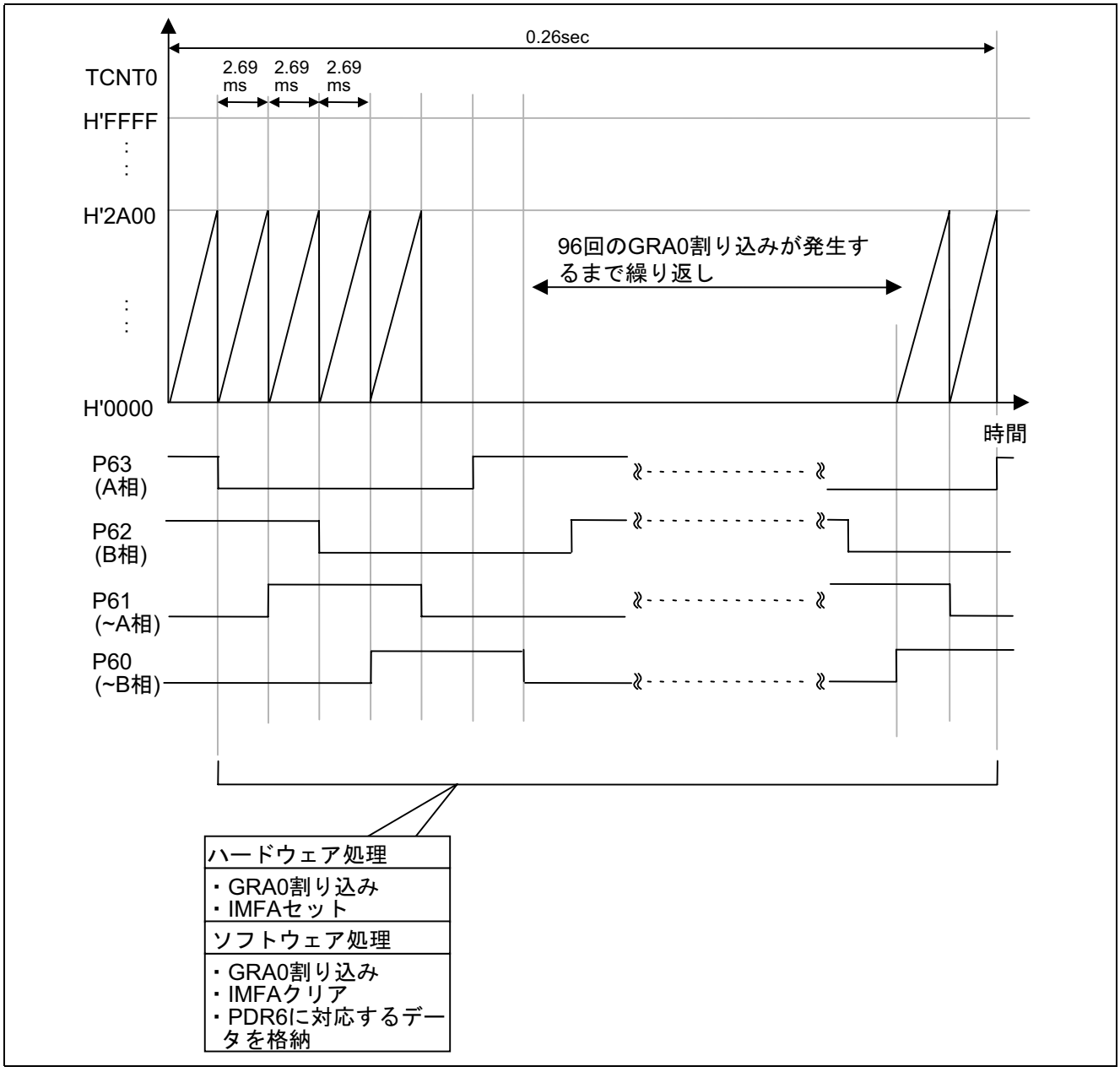


図9　**正転時** Constant **制御の動作原理**

3.7 **正転時** Slue Down **制御の動作原理を図** 10**に示します。**

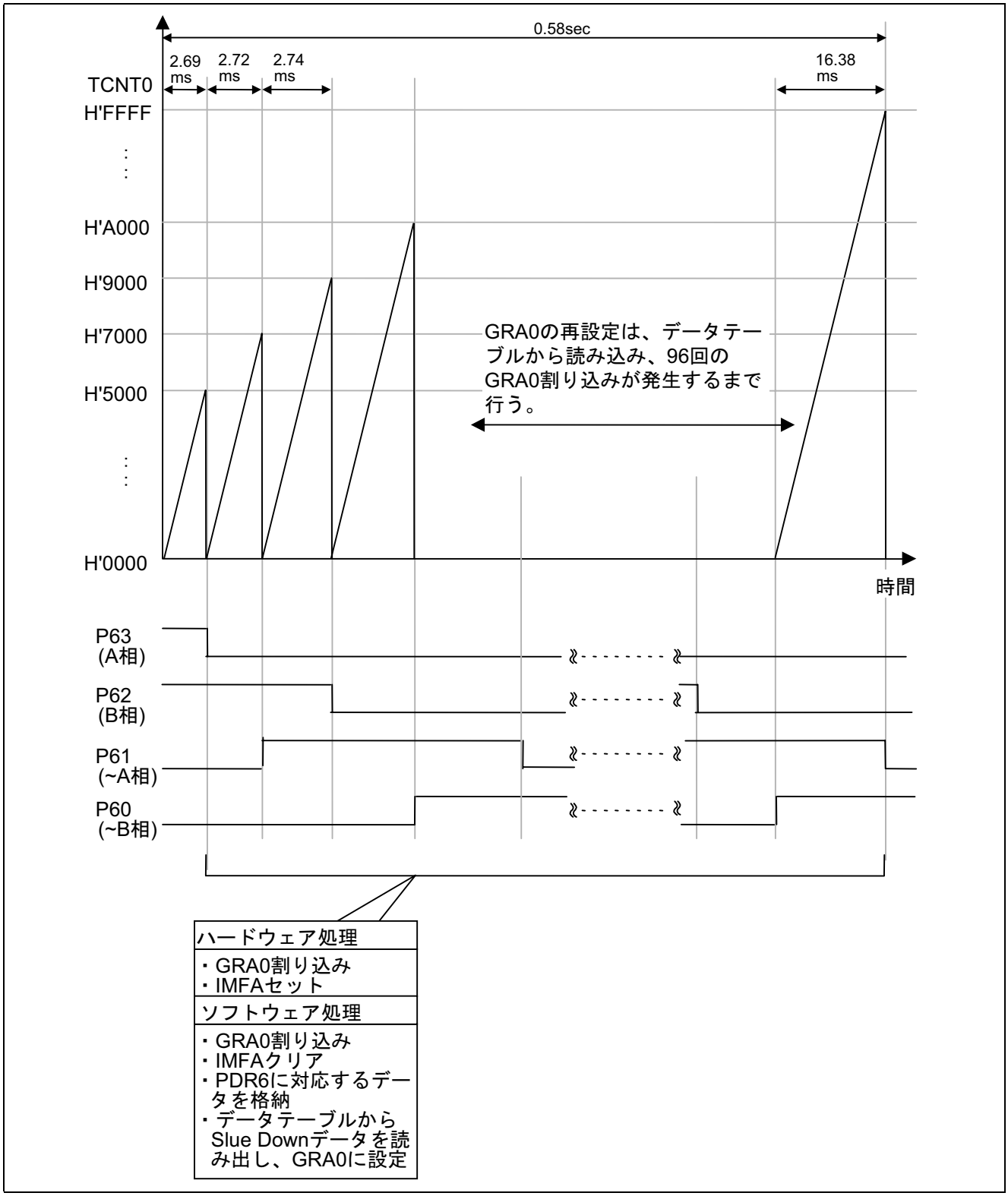


図10　**正転時** Slue Down **制御の動作原理**

3.8 停止制御の動作原理を図 11に示します。

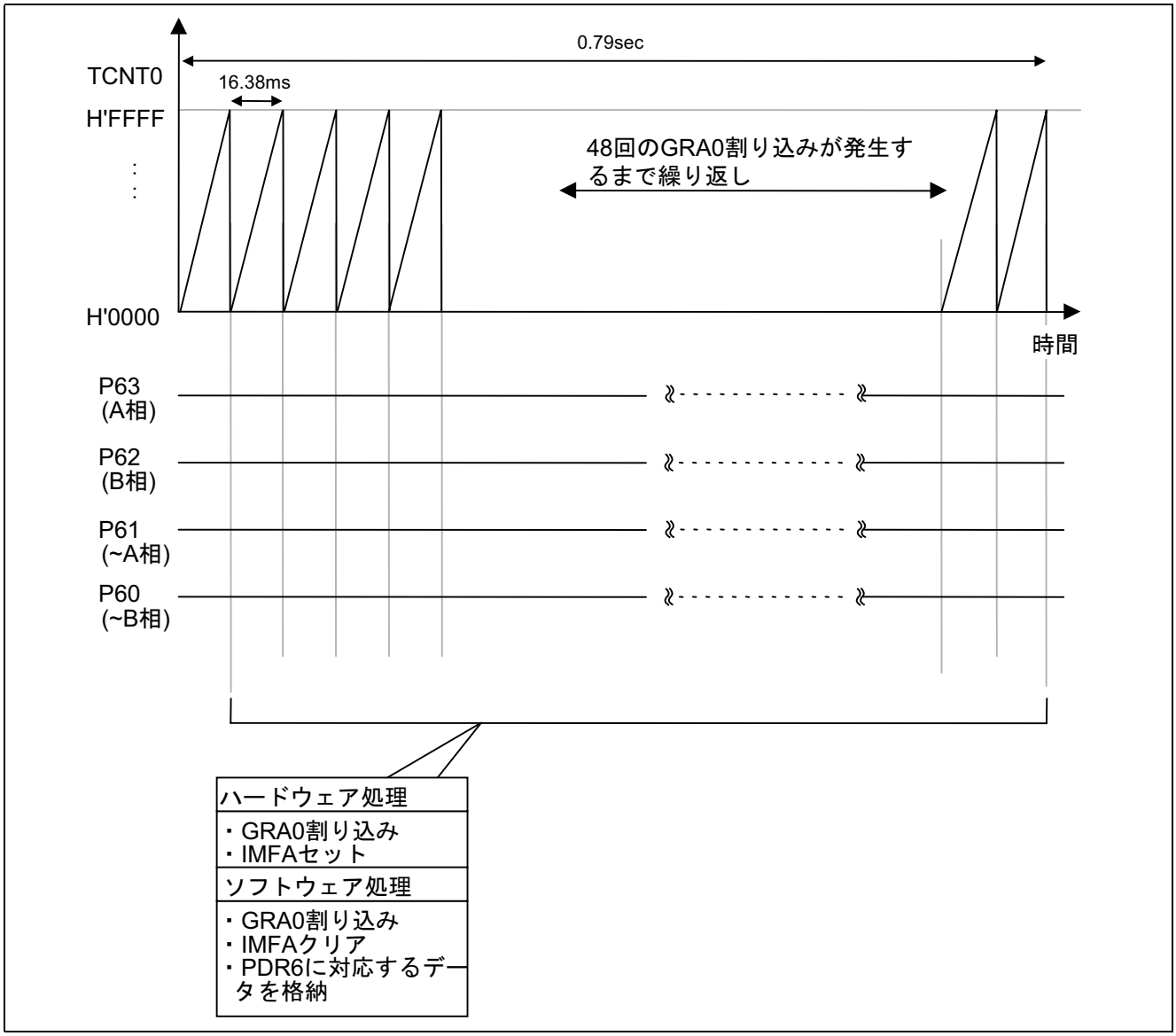


図11 停止制御の動作原理

3.9 **逆転時** Slue Up **制御の動作原理を図** 12**に示します。**

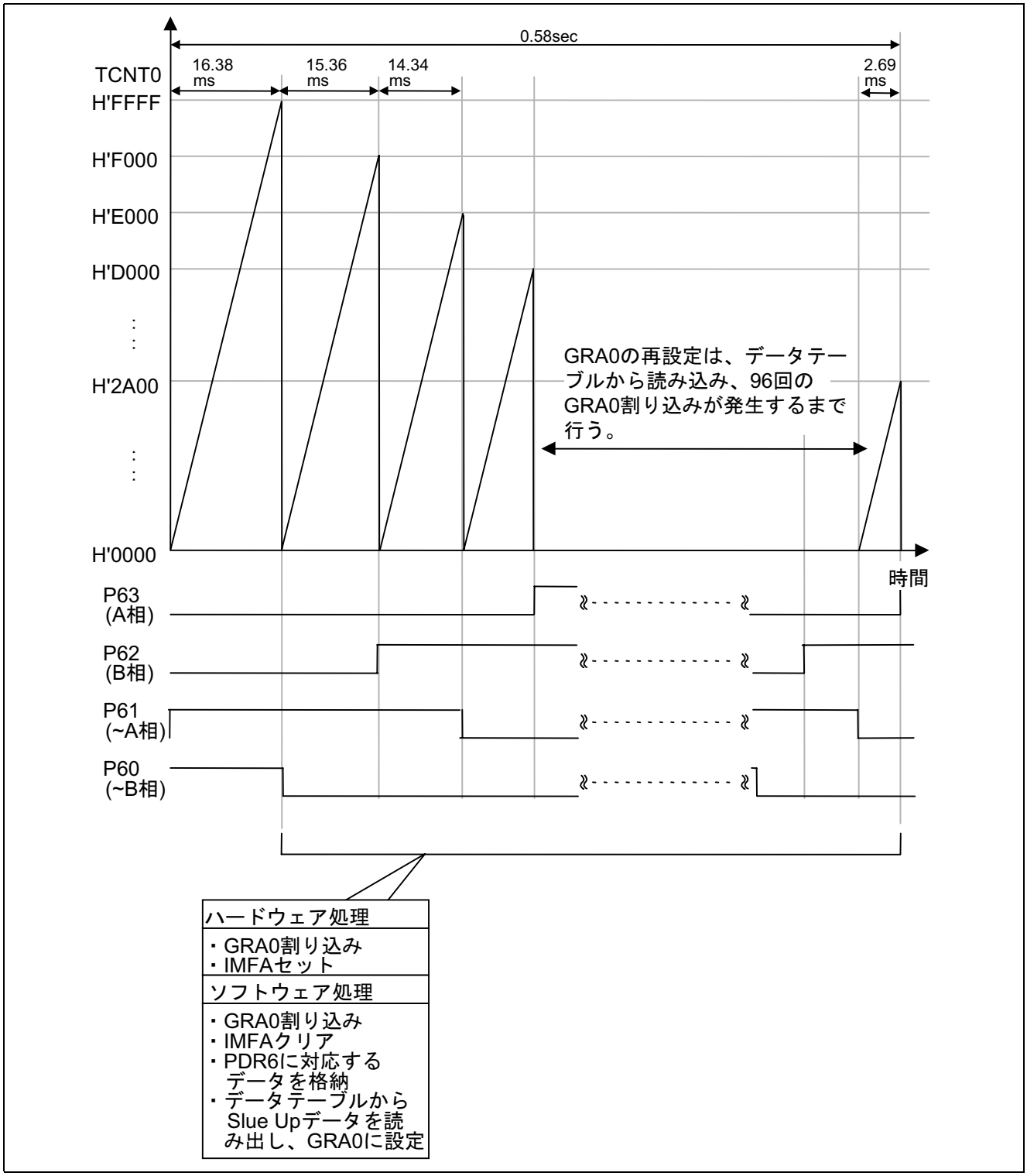


図12 **逆転時** Slue Up **制御の動作原理**

3.10 **逆転時** Constant **制御の動作原理を図** 13**に示します。**

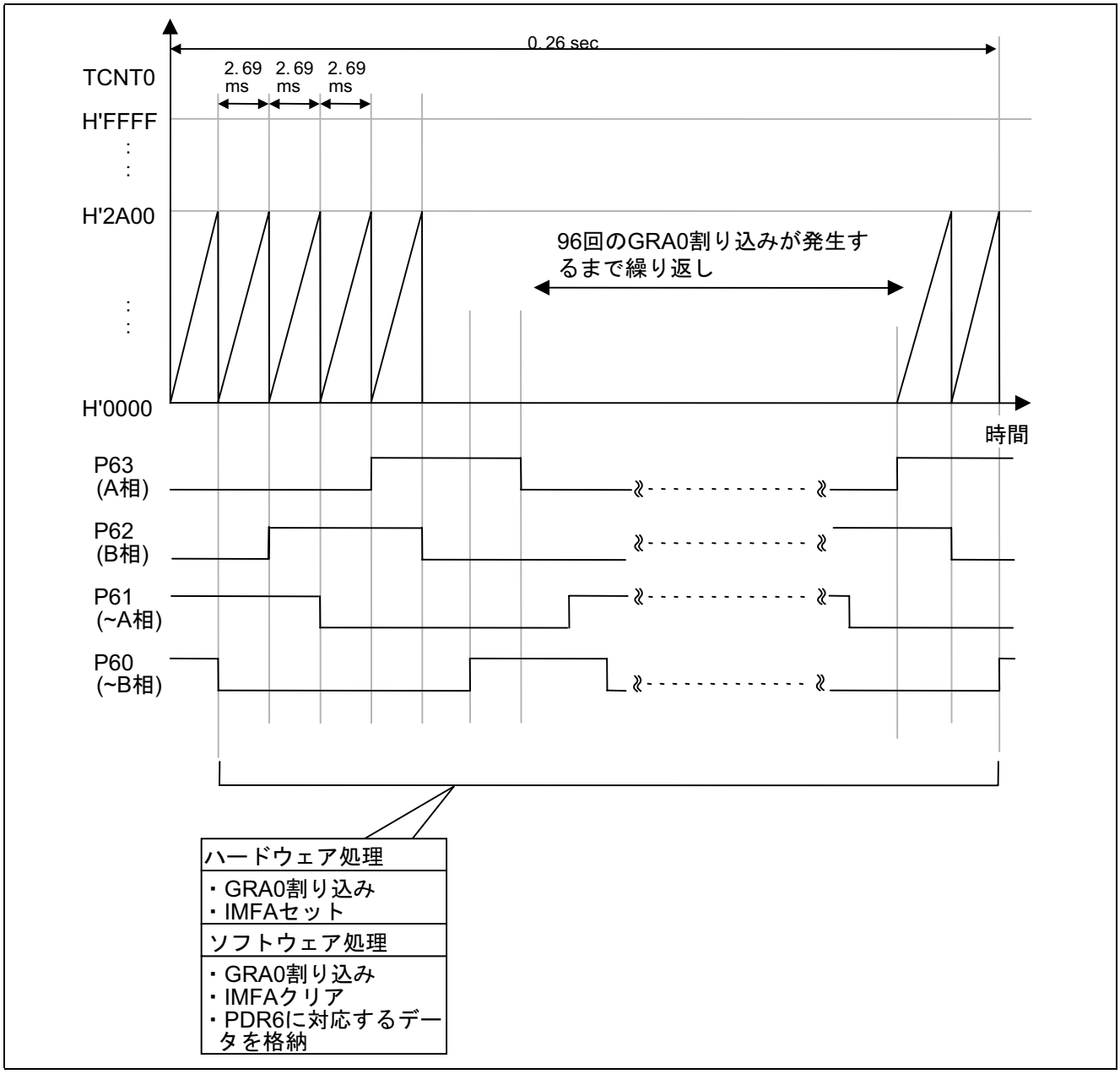


図13 **逆転時** Constant **制御の動作原理**

3.11 **逆転時** Slue Down **制御の動作原理を図** 14**に示します。**

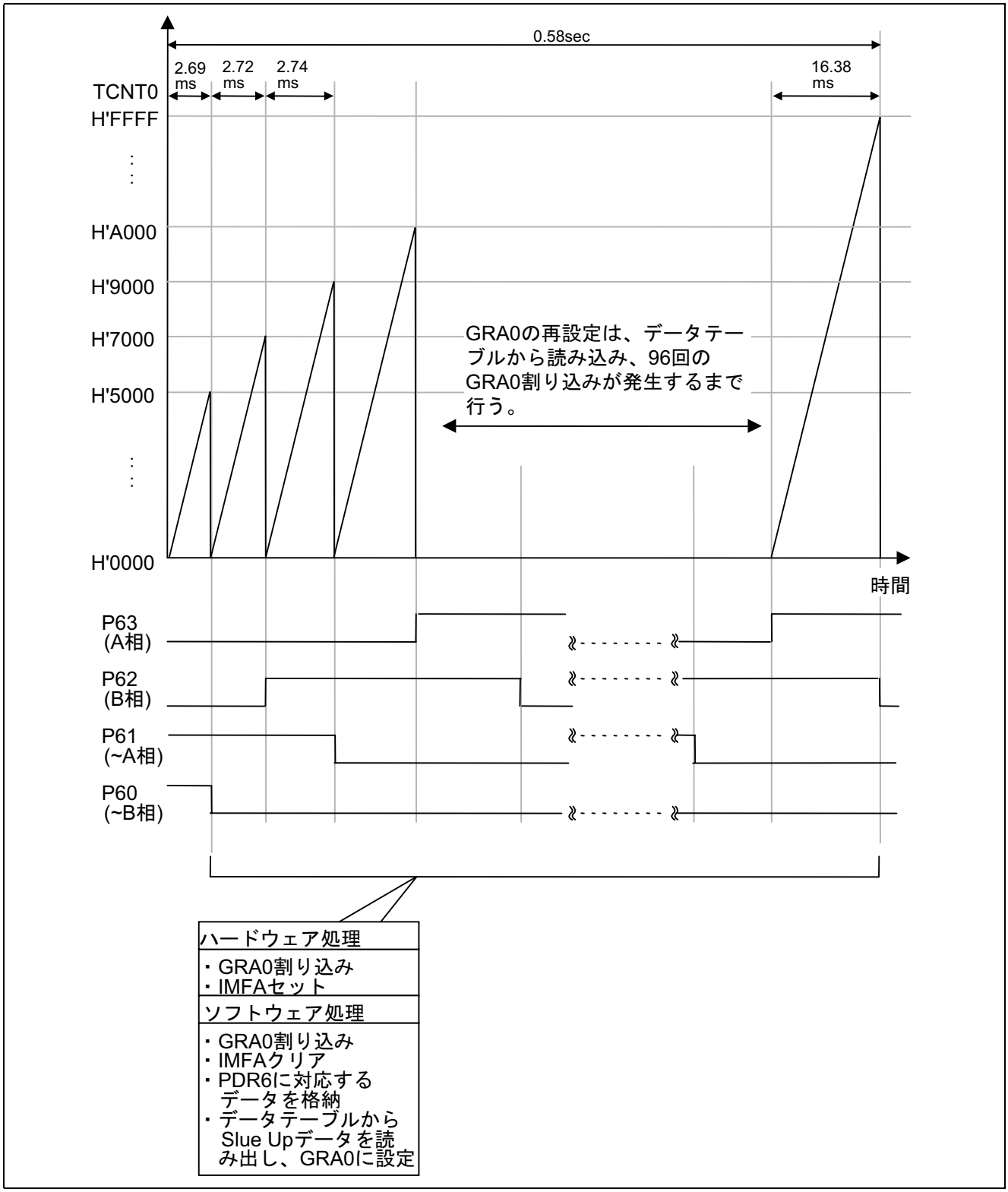


図14 **逆転時** Slue Down **制御の動作原理**

4. ソフトウェア説明

4.1 モジュール説明

本タスク例のモジュールを表 3に示します。

表3 モジュール説明

モジュール名	ラベル名	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 Z の初期設定、および割り込みの許可を行う
タイマ Z 割り込み処理	tzint	ステッピングモータのメインルーチン
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

4.2 引数の説明

本タスク例では、引数を使用しません。

4.3 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以下に示します。

- TCR0　タイマコントロールレジスタ 0　　アドレス：H'F700

ビット	ビット名	設定値	機能
7 6 5	CCLR2 CCLR1 CCLR0	CCLR2=0 CCLR1=0 CCLR0=1	カウンタクリア 2～0 CCLR2=0、CCLR1=0、CCLR0=1：GRA0 のコンペアマッチ／インプットキャプチャで TCNT0 クリア
4 3	CKEG1 CKEG0	CKEG1=0 CKEG0=0	クロックエッジ 1～0 CKEG1=0、CKEG0=0：クロックの立ち上がりでカウント
2 1 0	TPSC2 TPSC1 TPSC0	TPSC2=0 TPSC1=1 TPSC0=0	タイマプリスケーラ 2～0 TPSC2=0、TPSC1=1、TPSC0=0：φ/4 でカウント

- TIORA0　タイマ I/O コントロールレジスタ A0　　アドレス：H'F701

ビット	ビット名	設定値	機能
2 1 0	IOA2 IOA1 IOA0	IOA2=0 IOA1=0 IOA0=0	I/O コントロール A2～0 IOA2=0、IOA1=0、IOA0=0：GRA0 をアウトプットコンペアレジスタとし、コンペアマッチによる端子出力を禁止

- TSR0　タイマステータスレジスタ 0　　アドレス：H'F703

ビット	ビット名	設定値	機能
0	IMFA	0	インプットキャプチャ／コンペアマッチフラグ A IMFA=0：GRA0 と TCNT0 の値が一致していない IMFA=1：GRA0 と TCNT0 の値が一致した

- TIER0　タイマインタラプトイネーブルレジスタ 0　　アドレス：H'F704

ビット	ビット名	設定値	機能
0	IMIEA	1	インプットキャプチャ／コンペアマッチインタラプトイネーブル A TIORA0 の IOA2=0（アウトプットコンペア設定）時、 IMIEA=0：TSR0 の IMFA フラグによる割り込みを禁止 IMIEA=1：TSR0 の IMFA フラグによる割り込みを許可

- TCNT0　タイマカウンタ 0　　アドレス：H'F706
  機能：φ/4 の立ち上がりエッジでカウントする 16 ビットのアップカウンタ
  設定値：H'0000

- GRA0　ジェネラルレジスタ A0　　アドレス：H'F708
  機能：GRA0 の設定値と TCNT0 のカウンタ値が一致すると、コンペアマッチが発生
  設定値：H'FFFF

- TSTR　タイマスタートレジスタ　　アドレス：H'F720

ビット	ビット名	設定値	機能
0	STR0	0	チャネル 0 カウンタスタート STR0=0：TCNT0 は、カウント動作停止 STR0=1：TCNT0 は、カウント動作開始

- TMDR　タイマモードレジスタ　　アドレス：H'F721

ビット	ビット名	設定値	機能
0	SYNC	0	**タイマ同期** SYNC=0：TCNT0,TCNT1 **は、独立動作** SYNC=1：TCNT0,TCNT1 **は、同期動作**

- TFCR　タイマファンクションコントロールレジスタ　　アドレス：H'F723

ビット	ビット名	設定値	機能
1 0	CMD1 CMD0	CMD1=0 CMD0=0	**コンビネーションモード** 1～0 CMD1=0、CMD0=0：**チャネル** 0、1 **は、通常動作**

- TOER　タイマアウトプットマスタイネーブルレジスタ　　アドレス：H'F724

ビット	ビット名	設定値	機能
0	EA0	1	**マスタイネーブル** A0 EA0=0：FTIOA0 **端子の出力を許可** EA0=1：FTIOA0 **端子の出力を禁止**

- TOCR　タイマアウトプットコントロールレジスタ　　アドレス：H'F725

ビット	ビット名	設定値	機能
0	TOA0	0	**出力レベルセレクト** A0 TOA0=0：FTIOA0 **の初期出力は** 0 TOA0=1：FTIOA0 **の初期出力は** 1

- PDR6　ポートデータレジスタ 6　　アドレス：H'FFD9
  機能：P63～P60 をステッピングモータの励磁相駆動に使用する
  設定値：H'08

- PCR6　ポートコントロールレジスタ 6　　アドレス：H'FFE9
  機能：PCR6=H'0F のとき、P63～P60 を出力端子に設定
  設定値：H'0F

4.4 使用 RAM 説明

本タスク例の使用 RAM を表 4に示します。

表4 使用 RAM 説明

ラベル名	機能	メモリ消費量	使用モジュール名
tzcnt	ステッピングモータの励磁データである配列 pattbl[]の要素	1 バイト	メインルーチン タイマ Z 割り込み処理 正転 SlueUp 制御 正転 SlueDown 制御 正転 Constant 制御 回転停止 逆転 SlueUp 制御 逆転 SlueDown 制御 逆転 Constant 制御
sluecnt	SlueUp,SlueDown 時に使用する配列 uptbl[]の要素	1 バイト	メインルーチン タイマ Z 割り込み処理 正転 SlueUp 制御 正転 SlueDown 制御 逆転 SlueUp 制御 逆転 SlueDown 制御
nextmode	ステッピングモータの動作モードを設定する	1 バイト	メインルーチン タイマ Z 割り込み処理
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	2 バイト	メインルーチン タイマ Z 割り込み処理

表 4　使用 RAM 説明（つづき）

ラベル名	機能	メモリ消費量	使用モジュール名
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	8 バイト	メインルーチン 正転 SlueUp 制御 正転 SlueDown 制御 正転 Constant 制御 回転停止 逆転 SlueUp 制御 逆転 SlueDown 制御 逆転 SlueDown 制御
uptbl[96]	SlueUp,SlueDown 時の割り込み時間 データテーブル	192 バイト	メインルーチン 正転 SlueUp 制御 正転 SlueDown 制御 逆転 SlueUp 制御 逆転 SlueDown 制御

4.5 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, ・・・・A 相（P63）を励磁する。
    0x0C, ・・・・A 相（P63）,B 相（P62）を励磁する。
    0x04, ・・・・B 相（P62）を励磁する。
    0x06, ・・・・B 相（P62）,~A 相（P61）を励磁する。
    0x02, ・・・・~A 相（P61）を励磁する。
    0x03, ・・・・~A 相（P61）,~B 相（P60）を励磁する。
    0x01, ・・・・~B 相（P60）を励磁する。
    0x09, ・・・・~B 相（P60）,A 相（P63）を励磁する。
}
```

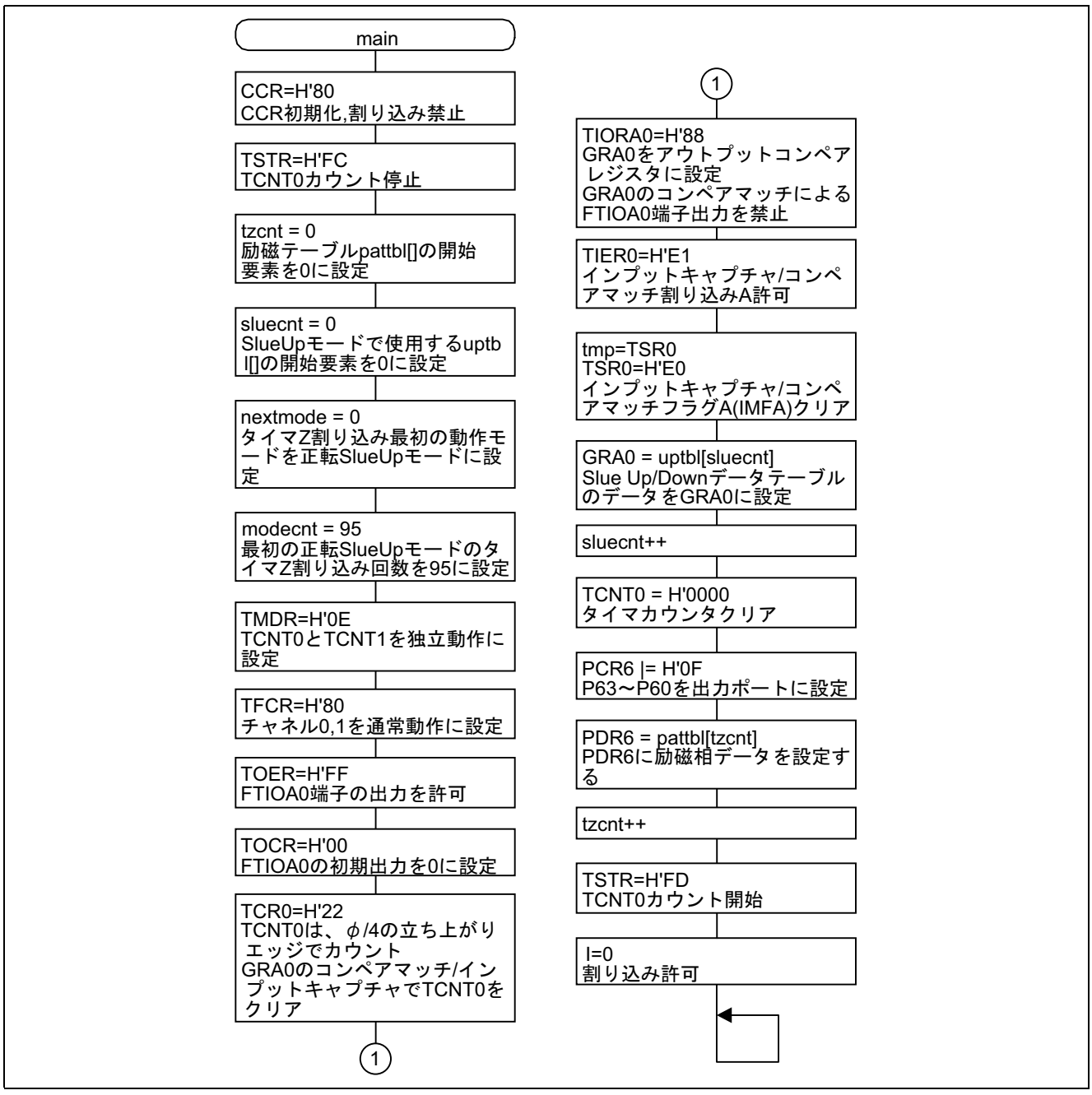
- SlueUp、SlueDown 設定データテーブル

```
uptbl[96]={
    0xFFFF,0xF000,0xE000,0xD500,0xCE40,0xC738,0xC030,0xB928,0xB220,0xAB18,
    0xA410,0x9DD0,0x9790,0x9150,0x8CA0,0x87F0,0x8340,0x8020,0x7D00,0x7AA8,
    0x7850,0x75F8,0x74BD,0x7148,0x6FB8,0x6E28,0x6C98,0x6B08,0x6978,0x67E8,
    0x66BC,0x6590,0x6464,0x6338,0x620C,0x6144,0x607C,0x5FB4,0x5EEC,0x5DCA,
    0x5CA8,0x5B86,0x5A64,0x5942,0x5820,0x56FE,0x55DC,0x54BA,0x5398,0x5276,
    0x5154,0x5032,0x4F10,0x4DEE,0x4CCC,0x4BAA,0x4A88,0x4966,0x4844,0x4722,
    0x4600,0x44DE,0x43BC,0x429A,0x4178,0x4056,0x3F34,0x3E12,0x3CF0,0x3BC4,
    0x3A34,0x3890,0x373C,0x35E8,0x3494,0x33D6,0x3318,0x325A,0x319C,0x30DE,
    0x3064,0x2FEA,0x2F70,0x2EF6,0x2E7C,0x2E02,0x2D9C,0x2D36,0x2CD0,0x2C6A,
    0x2C04,0x2B9E,0x2B38,0x2AD2,0x2A6C,0x2A00,
}
```

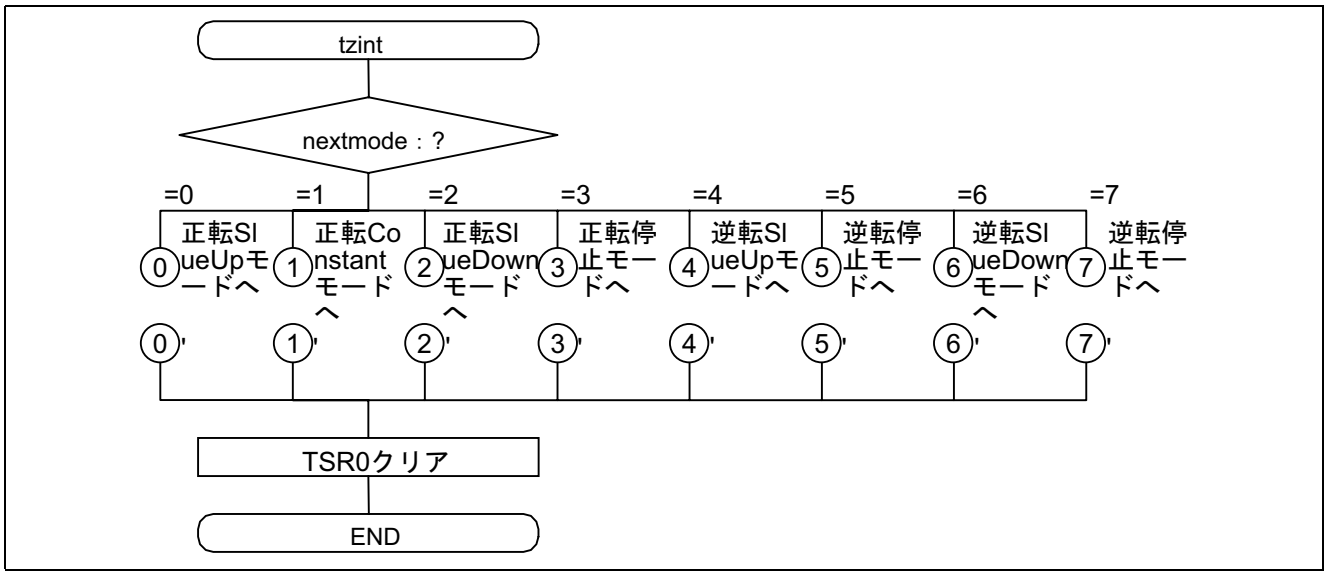
SlueUp、SlueDown 動作時の GRA0 割り込みで uptbl[]を順次 GRA0 に書き込む。書き込みは、ステッピングモータが 1 回転（96 ステップ）するまで続ける。

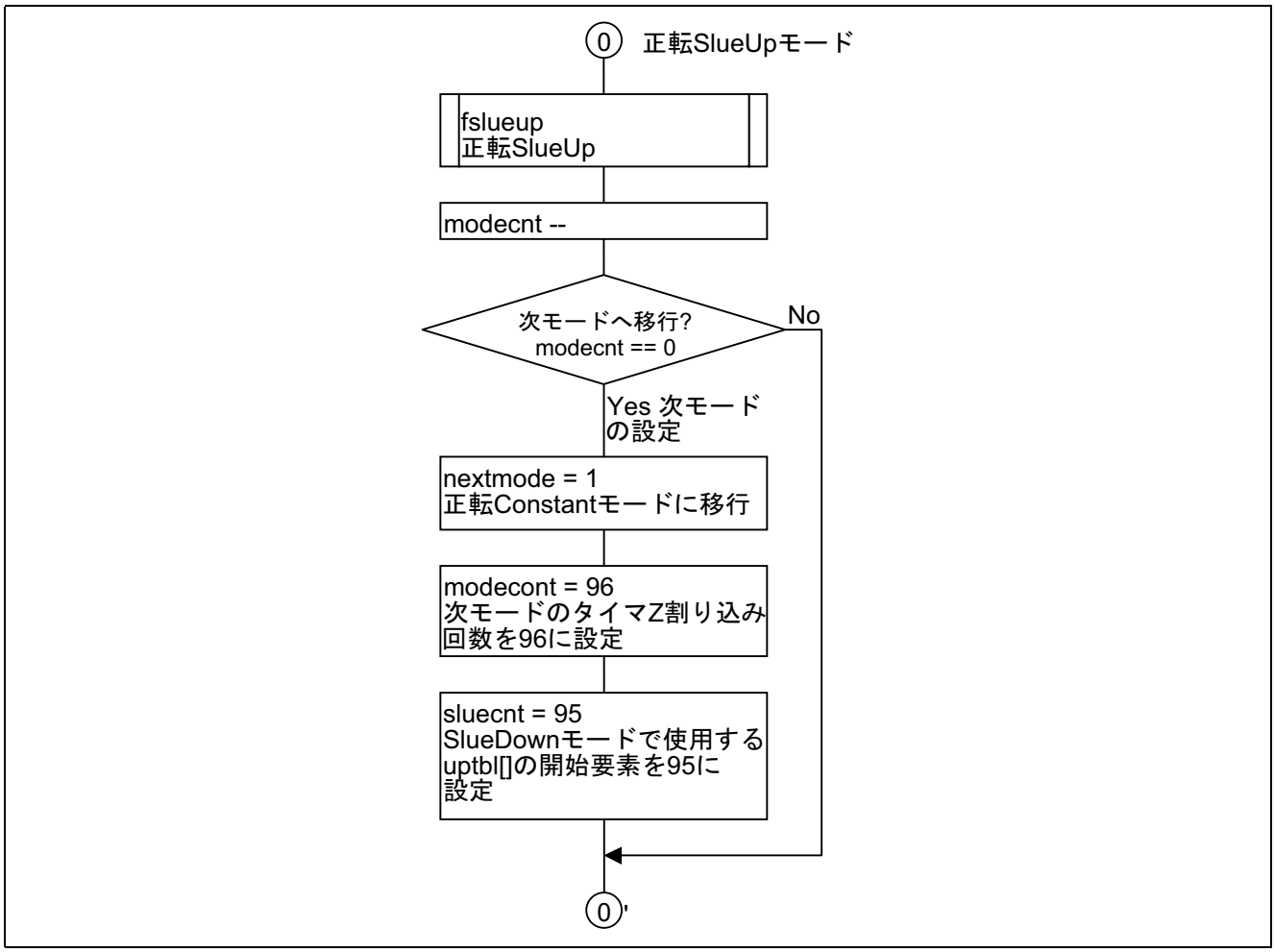
5. フローチャート

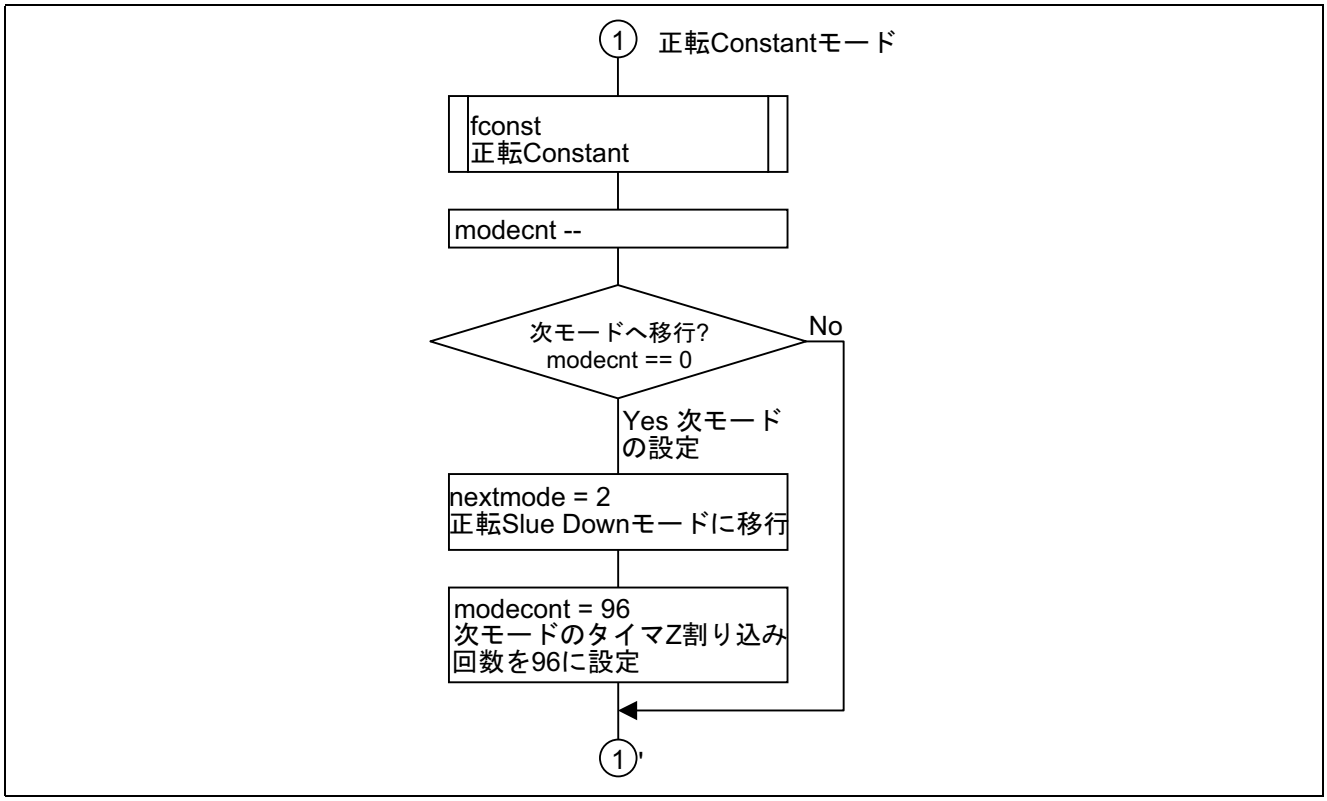
5.1 メインルーチン

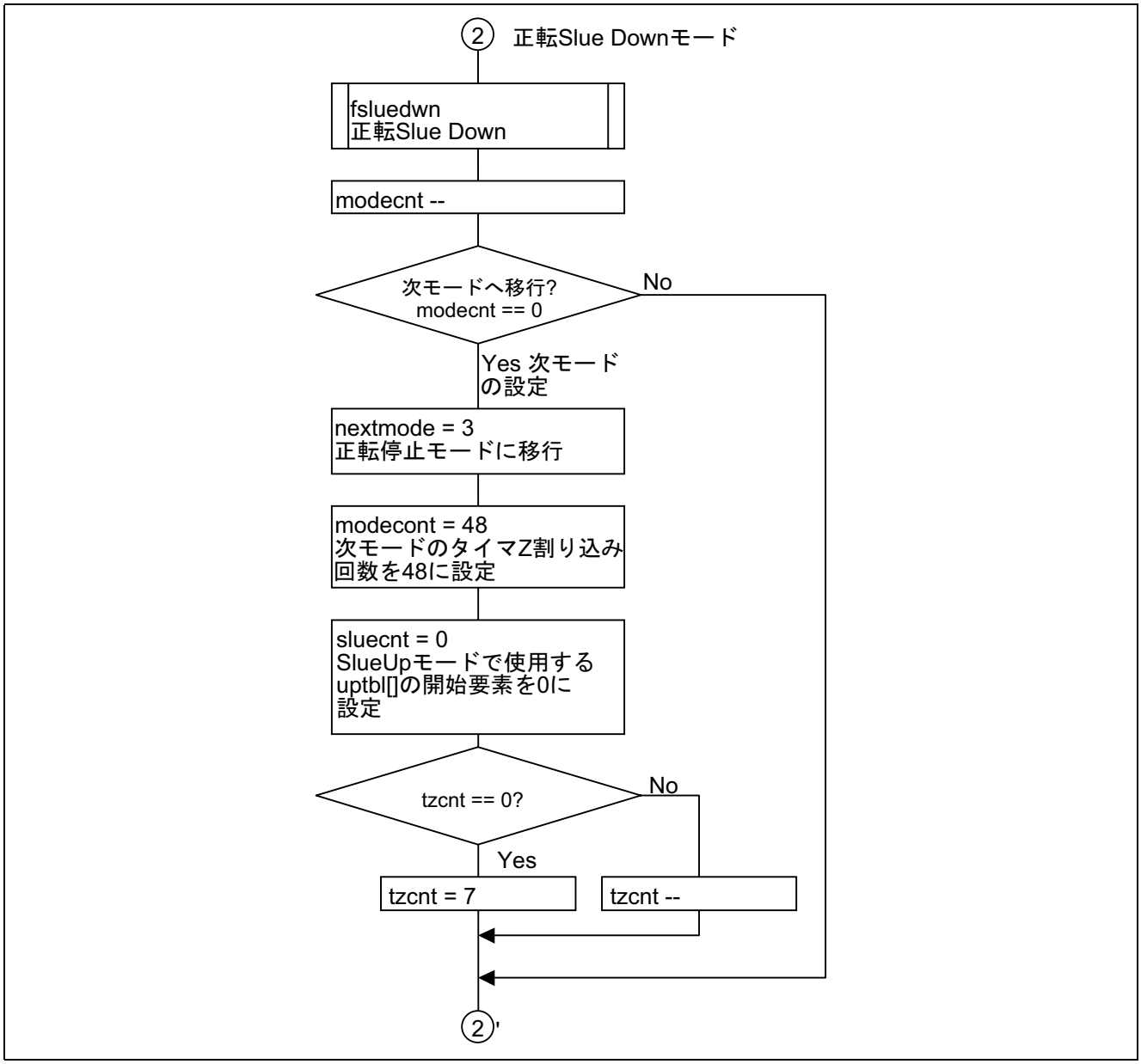


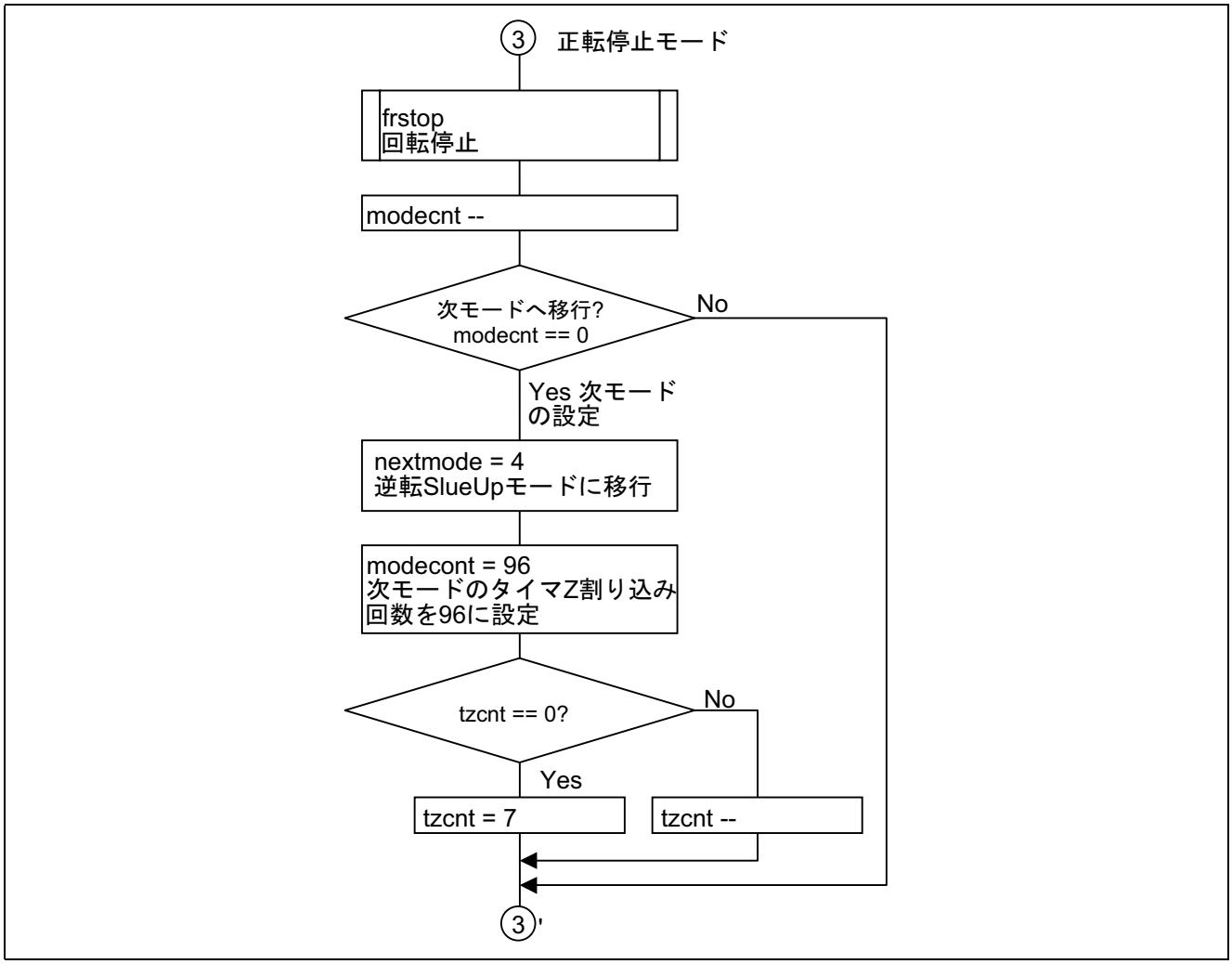
5.2 タイマ Z 割り込み

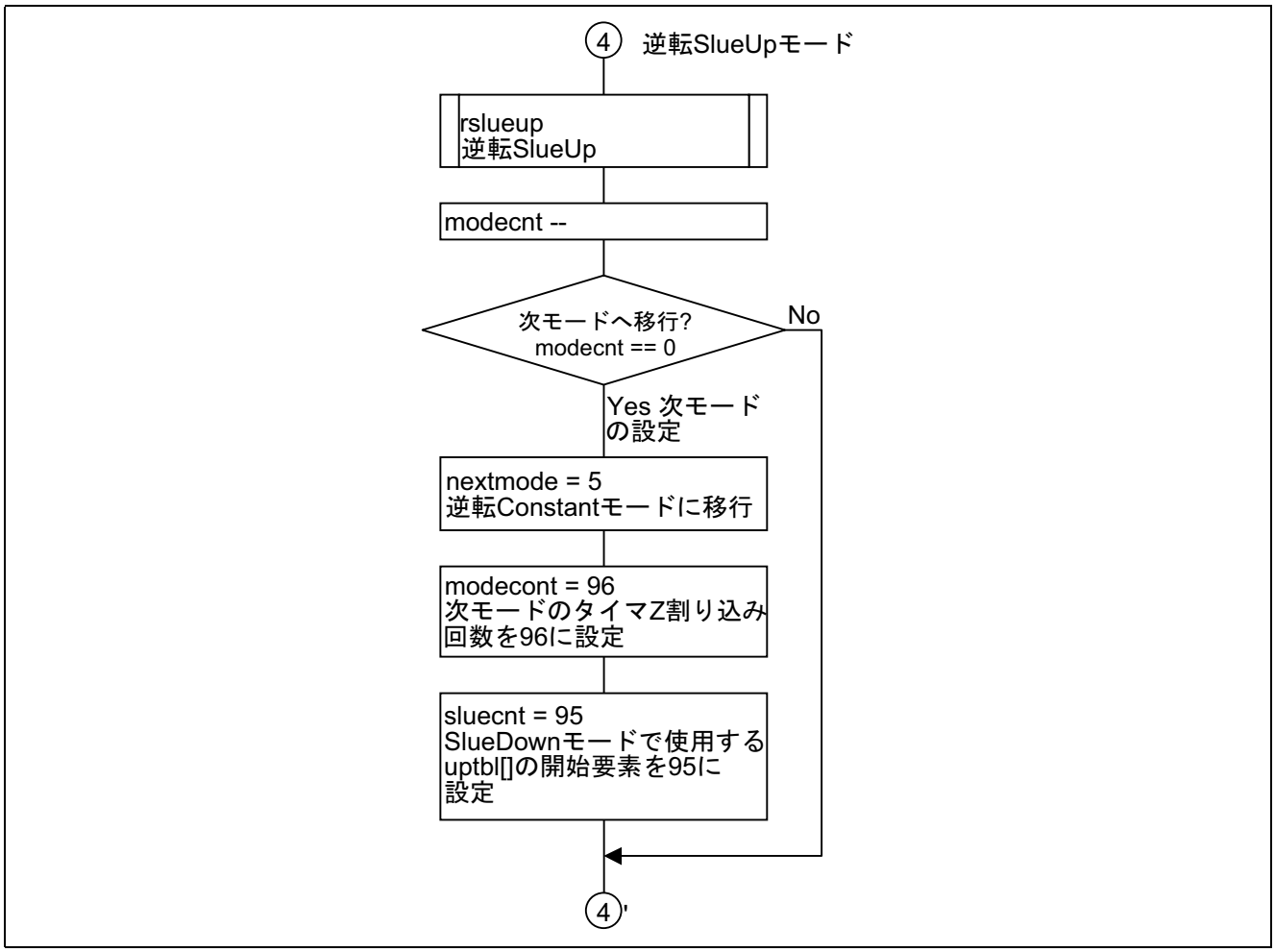

tzint
nextmode：?
=0
=1
=2
=3
=4
=5
=6
=7
0 正転SlueUpモードへ
1 正転Constantモードへ
2 正転SlueDownモードへ
3 正転停止モードへ
4 逆転SlueUpモードへ
5 逆転停止モードへ
6 逆転SlueDownモードへ
7 逆転停止モードへ
0'
1'
2'
3'
4'
5'
6'
7'
TSR0クリア
END

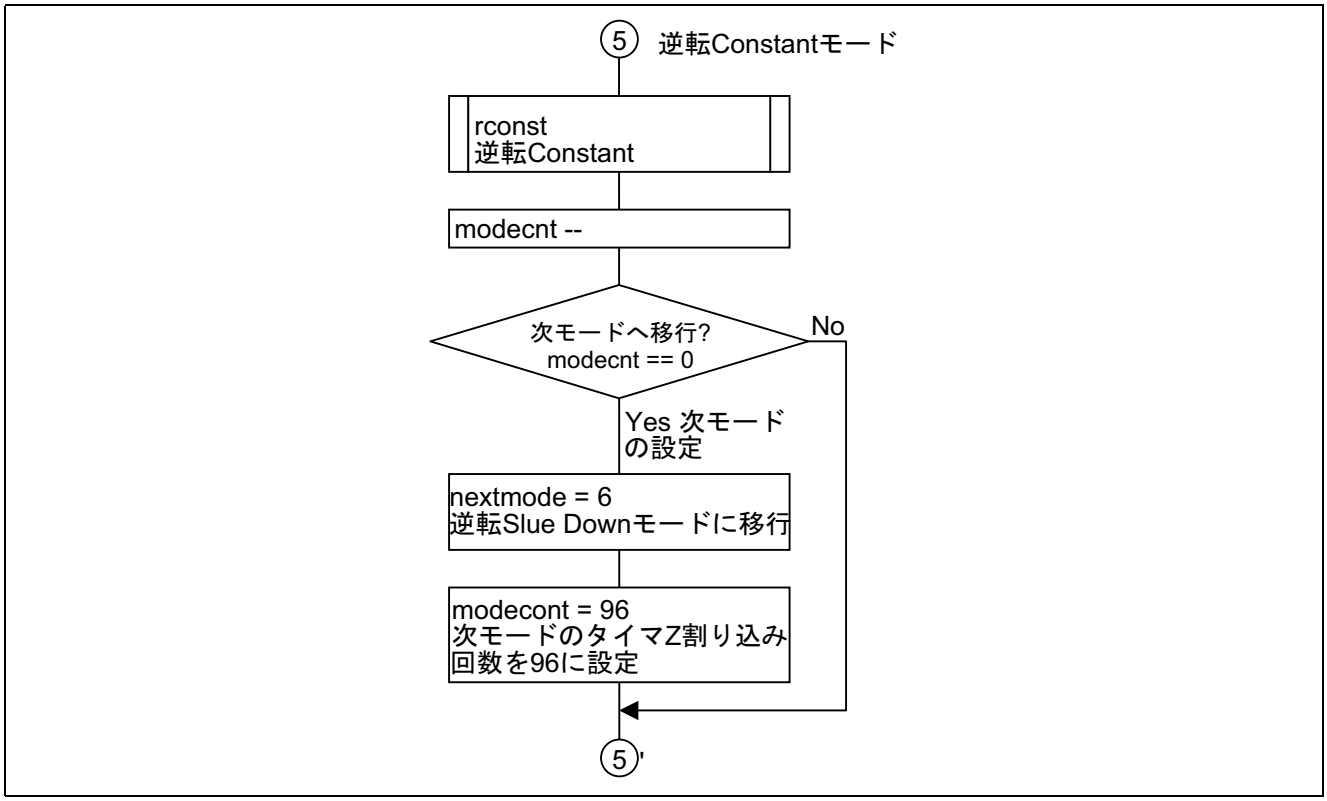

0 正転SlueUpモード
fslueup
正転SlueUp
modecnt --
次モードへ移行?
modecnt == 0
No
Yes 次モードの設定
nextmode = 1
正転Constantモードに移行
modecont = 96
次モードのタイマZ割り込み回数を96に設定
sluecnt = 95
SlueDownモードで使用するuptbl[]の開始要素を95に設定
0'

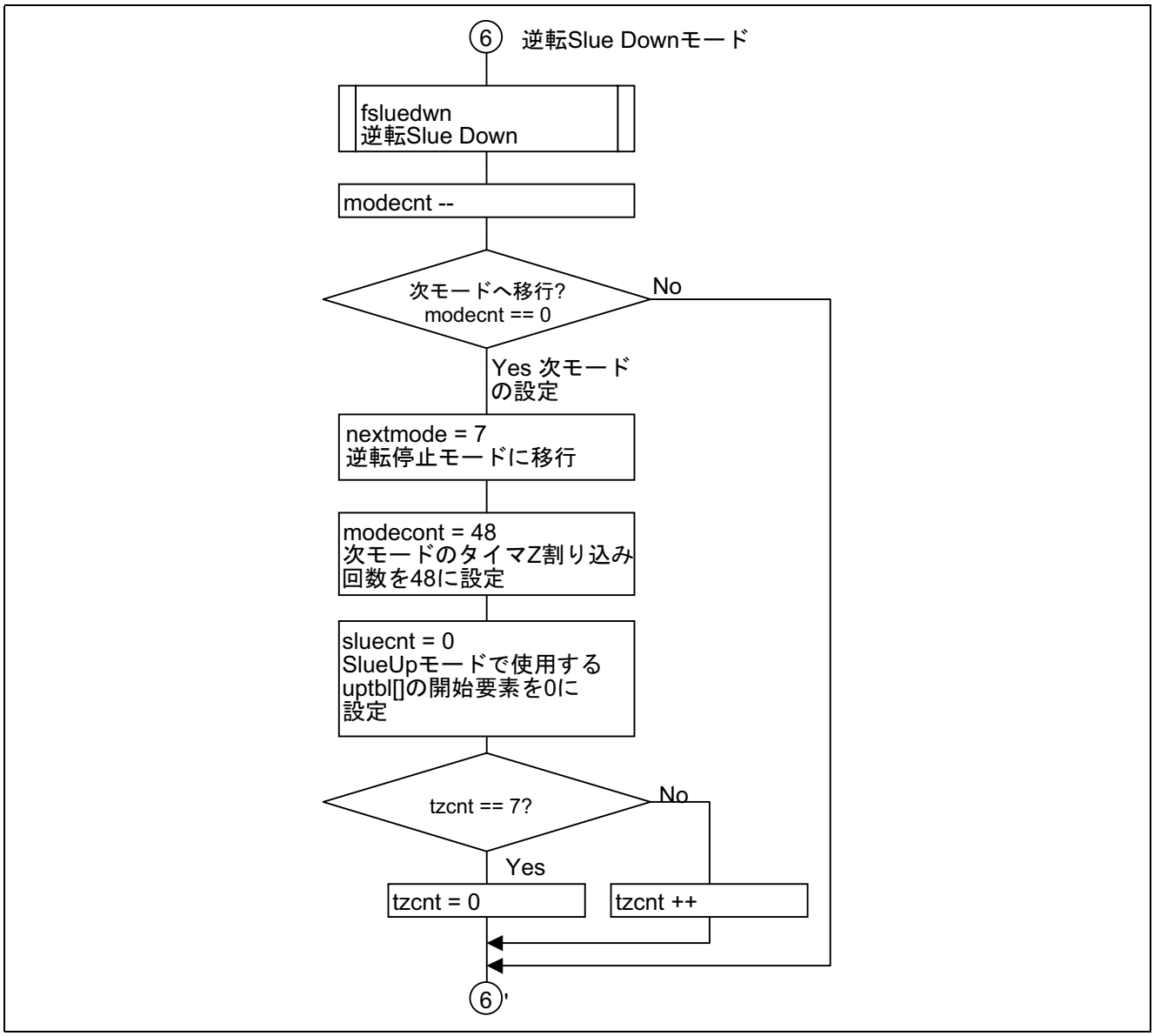

1
正転Constantモード
fconst
正転Constant
modecnt --
次モードへ移行?
modecnt == 0
No
Yes 次モード
の設定
nextmode = 2
正転Slue Downモードに移行
modecont = 96
次モードのタイマZ割り込み
回数を96に設定
1'

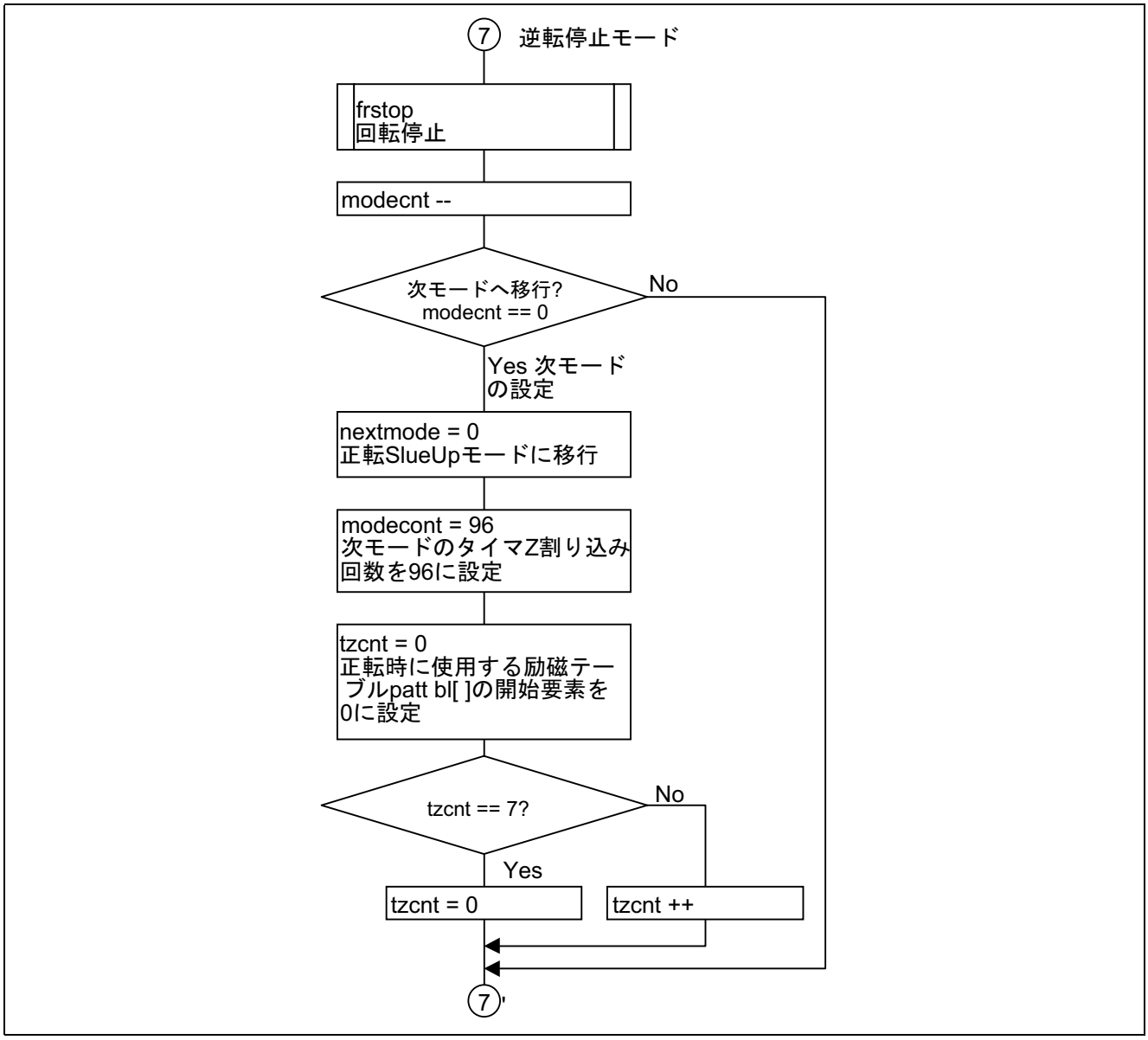

2
正転Slue Downモード
fsluedwn
正転Slue Down
modecnt --
次モードへ移行?
modecnt == 0
No
Yes 次モードの設定
nextmode = 3
正転停止モードに移行
modecont = 48
次モードのタイマZ割り込み回数を48に設定
sluecnt = 0
SlueUpモードで使用するuptbl[]の開始要素を0に設定
tzcnt == 0?
No
Yes
tzcnt = 7
tzcnt --
2'


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
modecont = 96
次モードのタイマZ割り込み
回数を96に設定
tzcnt == 0?
No
Yes
tzcnt = 7
tzcnt --
③'

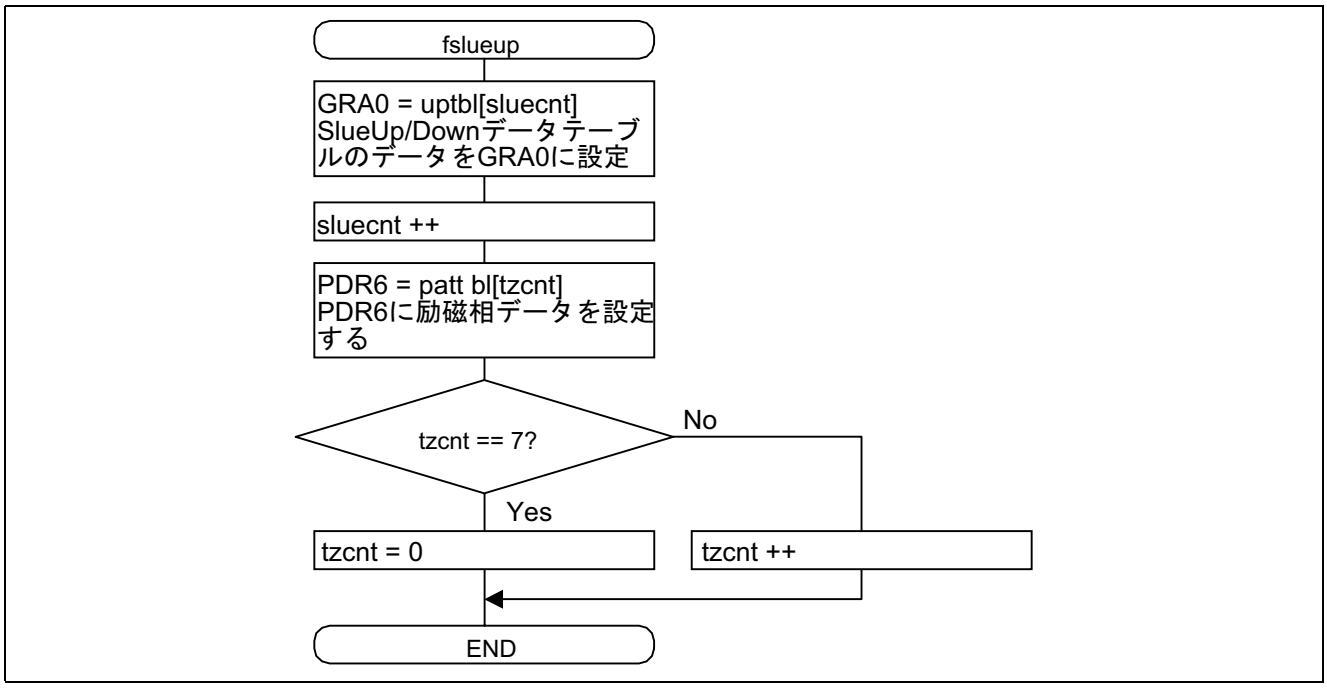

4
逆転SlueUpモード
rslueup
逆転SlueUp
modecnt --
次モードへ移行?
modecnt == 0
No
Yes 次モード
の設定
nextmode = 5
逆転Constantモードに移行
modecont = 96
次モードのタイマZ割り込み
回数を96に設定
sluecnt = 95
SlueDownモードで使用する
uptbl[]の開始要素を95に
設定
4'


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Slue Downモードに移行
modecont = 96
次モードのタイマZ割り込み
回数を96に設定
5'

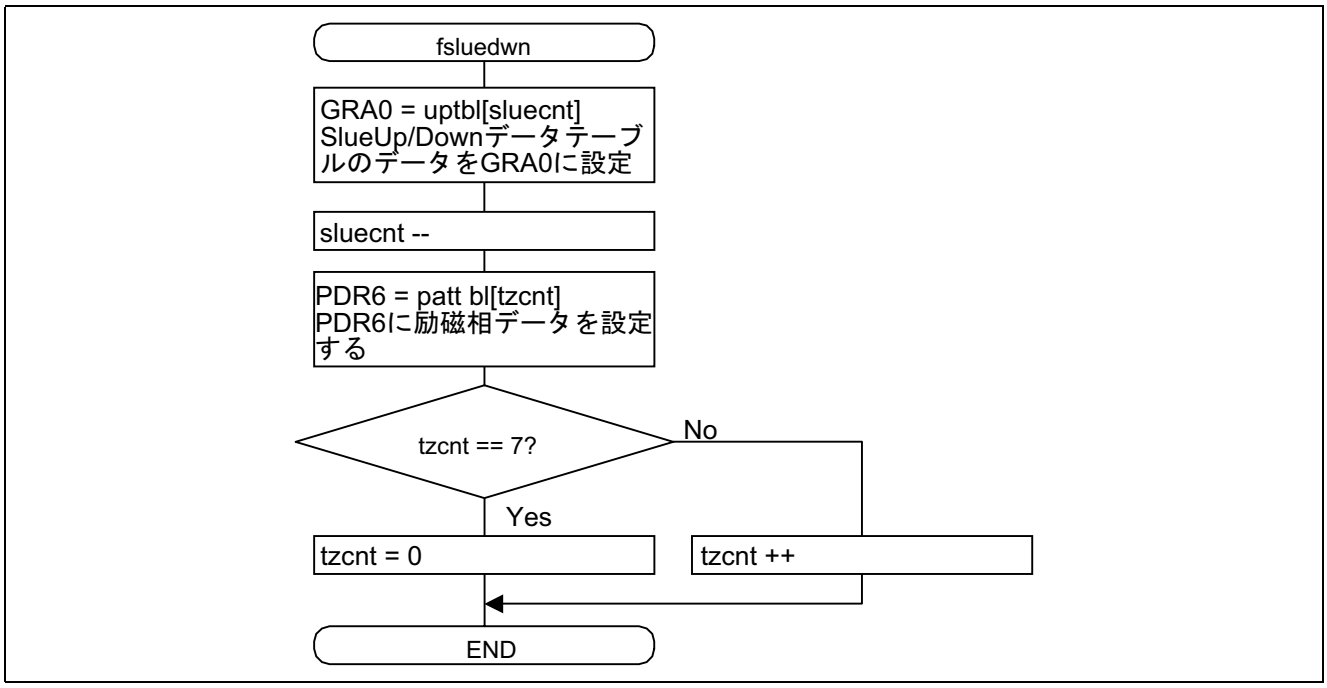

6
逆転Slue Downモード
fsluedwn
逆転Slue Down
modecnt --
次モードへ移行?
modecnt == 0
No
Yes 次モードの設定
nextmode = 7
逆転停止モードに移行
modecont = 48
次モードのタイマZ割り込み回数を48に設定
sluecnt = 0
SlueUpモードで使用するuptbl[]の開始要素を0に設定
tzcnt == 7?
No
Yes
tzcnt = 0
tzcnt ++
6'


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
modecont = 96
次モードのタイマZ割り込み回数を96に設定
tzcnt = 0
正転時に使用する励磁テーブルpatt bl[ ]の開始要素を0に設定
tzcnt == 7?
No
Yes
tzcnt = 0
tzcnt ++
⑦'

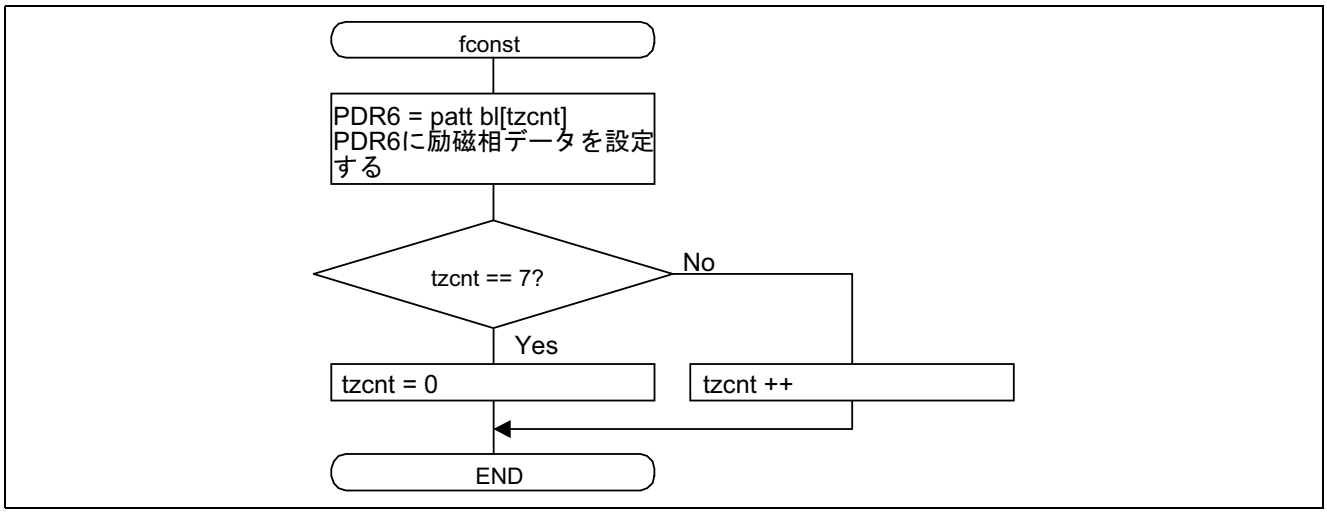
5.3 正転時 Slue UP 制御



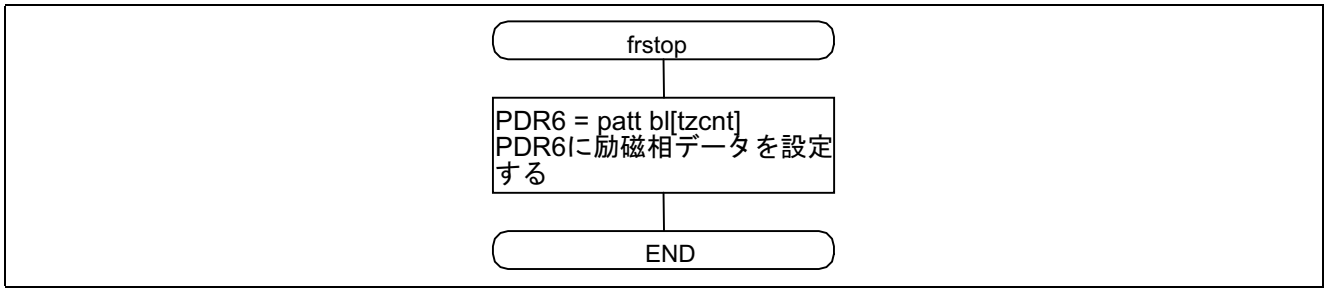
5.4 正転時 Slue Down 制御



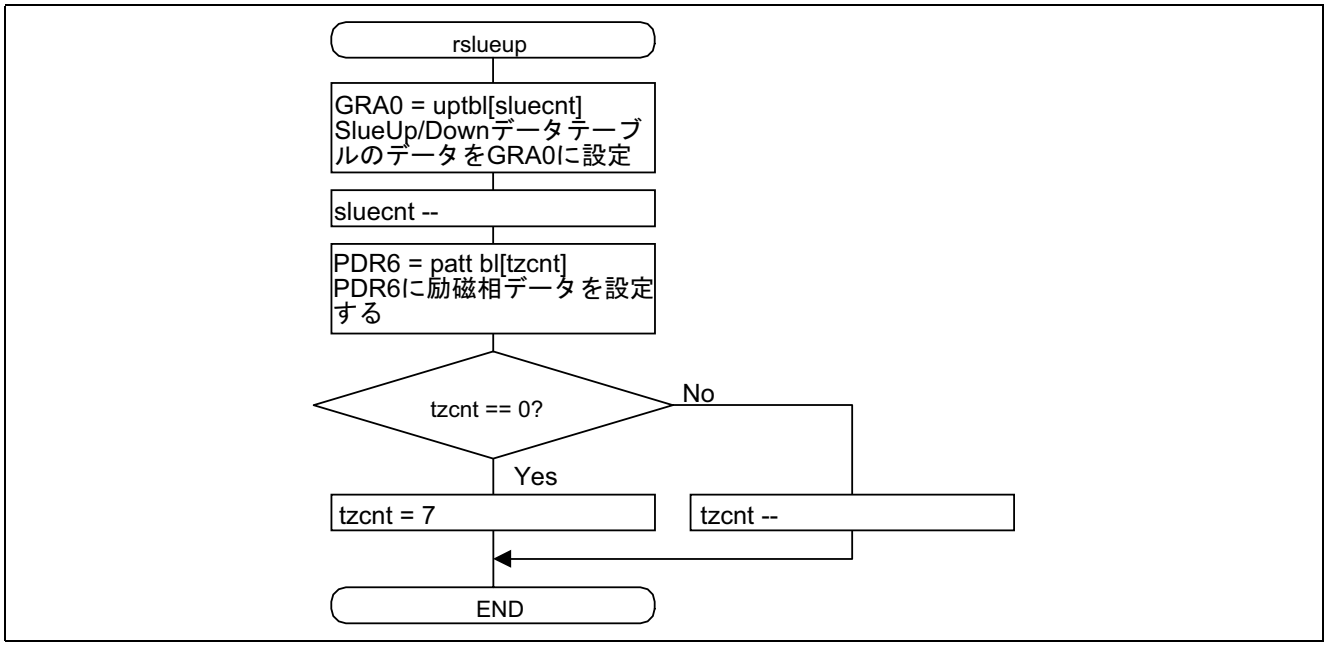
5.5 正転時 Constant 制御



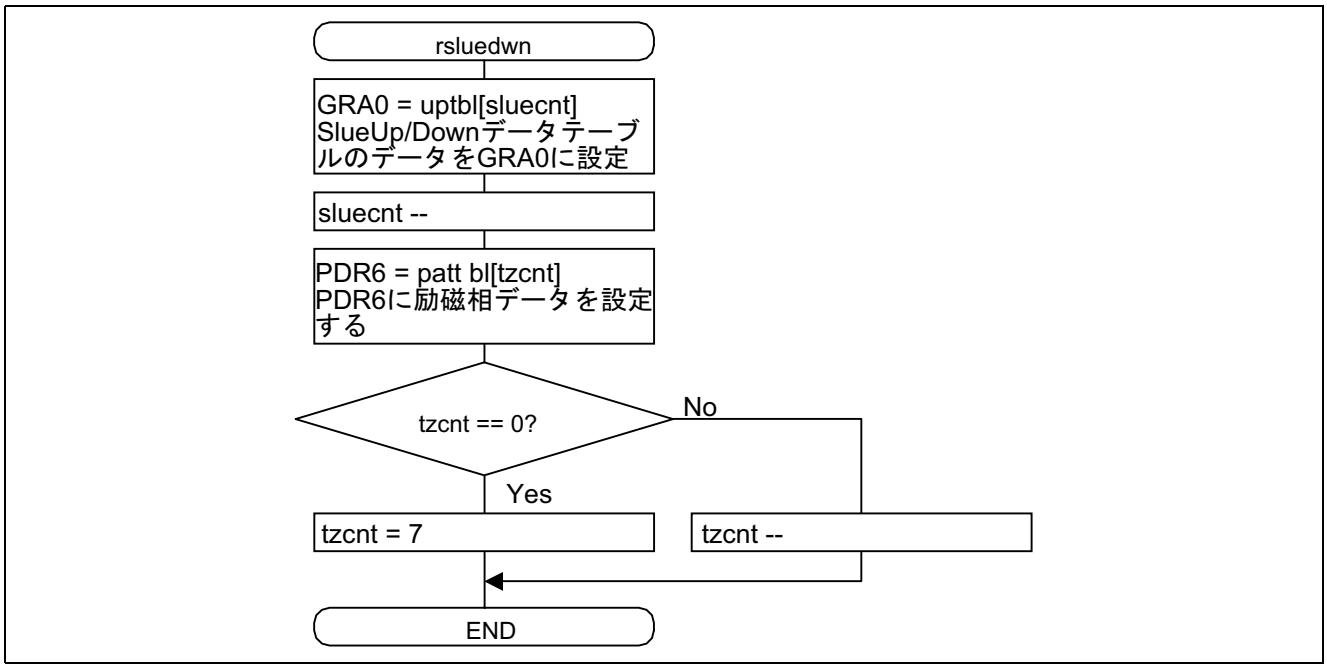
5.6 停止制御



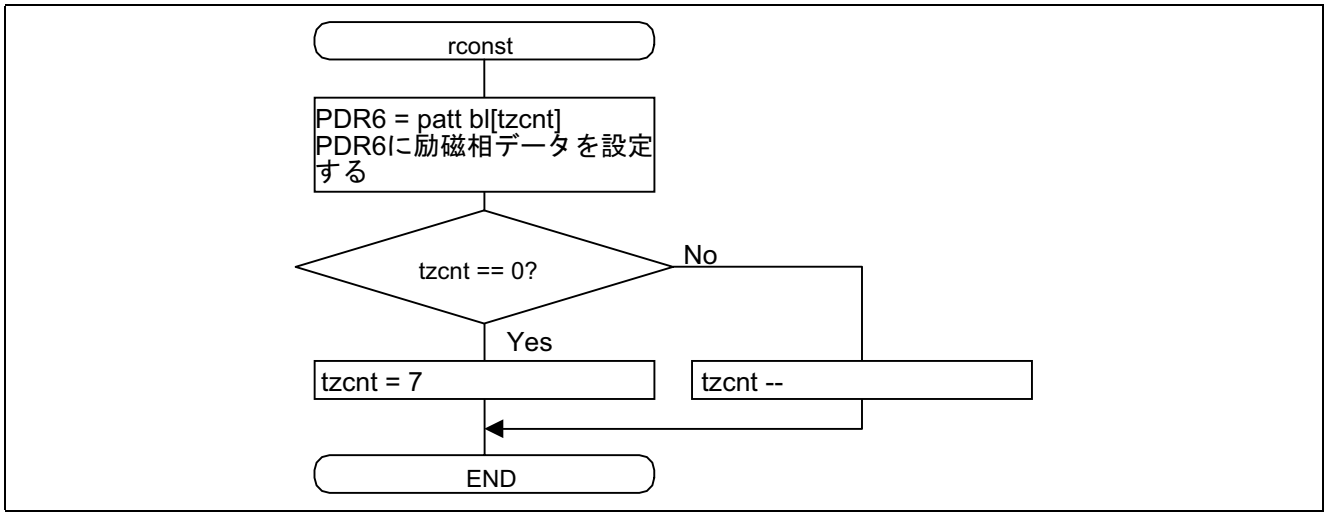
5.7 逆転時 Slue UP 制御



5.8 逆転時 Slue Down 制御



5.9 逆転時 Constant 制御



5.10 リンクアドレス指定

セクション名	アドレス
CV1	H'0000
CV2	H'0034
P	H'0100
C	H'0600
DOUTDT	H'0610
B	H'FB80

6. プログラムリスト

```
/************************************************************/
/*                                                          */
/*  H8/300HN Series -H8/3687-                               */
/*  Application Note                                        */
/*                                                          */
/*  '1-2 Phase Excitation Control for a Stepping Motor      */
/*                                                          */
/*  Function                                                */
/*  : Timer Z 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 TCR0       *(volatile unsigned char *)0xF700      /* Timer control
register_0        */
#define TIORA0     *(volatile unsigned char *)0xF701      /* Timer I/O Control
Register A_0   */
#define TSR0       *(volatile unsigned char *)0xF703      /* Timer status
register_0        */
#define TSR0_BIT   (*(struct BIT *)0xF703)              /* Timer status
register_0        */
#define IMFB       TSR0_BIT.b1                         /* Input Capture/Compare
Match FlagB*/
#define IMFA       TSR0_BIT.b0                         /* Input Capture/Compare
Match FlagA*/
#define TIER0      *(volatile unsigned char *)0xF704      /* Timer interrupt enable
register0 */
#define TIER0_BIT   (*(struct BIT *)0xF704)              /* Timer interrupt enable
register0 */
#define IMIEA      TIER0_BIT.b0                        /* Input Capture/Compare
Match     */
                                                  /*             Interrupt
Enable A */
```

```
#define TCNT0      *(volatile unsigned short *)0xF706     /* Timer counter_0
*/
#define GRA0       *(volatile unsigned short *)0xF708     /* General register A_0
*/
#define TSTR       *(volatile unsigned char *)0xF720      /* Timer start register
*/
#define TMDR       *(volatile unsigned char *)0xF721      /* Timer mode register
*/
#define TFCR       *(volatile unsigned char *)0xF723      /* Timer function control
register  */
#define TOER       *(volatile unsigned char *)0xF724      /* Timer output master
enable      */
                                                  /*
register */
#define TOCR       *(volatile unsigned char *)0xF725      /* Timer output control
register    */
#define PDR6       *(volatile unsigned char *)0xFFD9      /* Port Data Register 6
*/
#define PCR6       *(volatile unsigned char *)0xFFE9      /* Port Control Register
6         */

#pragma interrupt   (tz0int)
/************************************************************/
/*  Function define                                         */
/************************************************************/
void main ( void );
void tz0int ( void );
void fslueup ( void );
void fsluedwn ( void );
void fconst ( void );
void frstop ( void );
void rslueup ( void );
void rsluedwn ( void );
void rconst ( void );

/************************************************************/
/*  Ram define                                              */
/************************************************************/
unsigned char   tzcnt,sluecnt,nextmode;
unsigned short  modecnt;

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

unsigned short  uptbl[96] = {             /* Stepping Motor Output Patarn Table */
   0xFFFF,0xF000,0xE000,0xD500,0xCE40,0xC738,0xC030,0xB928,0xB220,0xAB18,
   0xA410,0x9DD0,0x9790,0x9150,0x8CA0,0x87F0,0x8340,0x8020,0x7D00,0x7AA8,
   0x7850,0x75F8,0x74BD,0x7148,0x6FB8,0x6E28,0x6C98,0x6B08,0x6978,0x67E8,
   0x66BC,0x6590,0x6464,0x6338,0x620C,0x6144,0x607C,0x5FB4,0x5EEC,0x5DCA,
   0x5CA8,0x5B86,0x5A64,0x5942,0x5820,0x56FE,0x55DC,0x54BA,0x5398,0x5276,
```

```
    0x5154,0x5032,0x4F10,0x4DEE,0x4CCC,0x4BAA,0x4A88,0x4966,0x4844,0x4722,
    0x4600,0x44DE,0x43BC,0x429A,0x4178,0x4056,0x3F34,0x3E12,0x3CF0,0x3BC4,
    0x3A34,0x3890,0x373C,0x35E8,0x3494,0x33D6,0x3318,0x325A,0x319C,0x30DE,
    0x3064,0x2FEA,0x2F70,0x2EF6,0x2E7C,0x2E02,0x2D9C,0x2D36,0x2CD0,0x2C6A,
    0x2C04,0x2B9E,0x2B38,0x2AD2,0x2A6C,0x2A00,
};

/************************************************************/
/*  Vector Address                                          */
/************************************************************/
#pragma section    V1                          /* VECTOR SECTOIN SET            */
void (*const VEC_TBL1[])(void) = {
    main                                       /* 00 Reset                      */
};

#pragma section    V2                          /* VECTOR SECTOIN SET            */
void (*const VEC_TBL2[])(void) = {
    tz0int                                     /* 34 Timer Z0 Interrupt         */
};

#pragma entry main(sp=0xFF80)
#pragma section                                /* P                            */
/************************************************************/
/*  Main Program                                            */
/************************************************************/
void main ( void )
{
    unsigned char tmp;

    set_ccr(0x80);                            /* Initialize CCR/Interrupt Disable */

    TSTR = 0xFC;                              /* TCNT0 count stop              */
    tzcnt = 0;                                /* Output Pattern table counter set */
    sluecnt = 0;                              /* Slue Up/Down table counter set   */
    nextmode = 0;
    modecnt = 95;                             /* Motor Slue mode countset "95"    */

    TMDR = 0x0E;                              /* TCNT0,TCNT1 Single Mode       */
    TFCR = 0x80;                              /* Chanel 0,1 is Normal Mode     */
    TOER = 0xFF;                              /* FTIOA0 Output Disable         */
    TOCR = 0x00;                              /* FTIOA0 initial outputs is 0   */
    TCR0 = 0x22;                              /* Rising edge, phi/4 Clock count   */
    TIORA0 = 0x88;                            /* FTIOA0 Toggle Output          */
    TIER0 = 0xE1;                             /* IMFA Interrupt Enable         */

    tmp = TSR0;
    TSR0 = 0xE0;                              /* Interrupt Flag Clear          */
    GRA0 = uptbl[sluecnt];                    /* Set GRA0                      */
    sluecnt++;

    TCNT0 = 0x0000;                           /* Set TCNT0                     */
    PCR6 |= 0x0F;                             /* Port8 Output                  */
    PDR6 = pattbl[tzcnt];                     /* PDR6 Set Output Pattern       */
    tzcnt++;
    TSTR = 0xFD;                              /* TCNT0 count start             */
```

```
    set_imask_ccr(0);                       /* Interrupt Enable             */

    while(1);
}

/************************************************************/
/*  Timer Z0 Interrupt                                      */
/************************************************************/
void tz0int ( 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 = 95;               /* 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 */
                if(tzcnt==0)
                    tzcnt = 7;
                else
                    tzcnt--;
            }
            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"      */
                if(tzcnt==0)
                    tzcnt = 7;
```

```
            else
                tzcnt--;
        }
        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 = 95;                   /* 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 */
            if(tzcnt==7)
                tzcnt = 0;
            else
                tzcnt++;
        }
        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"      */
            if(tzcnt==7)
                tzcnt = 0;
            else
                tzcnt++;
        }
        break;
    }

    tmp = TSR0;
    TSR0 = 0xE0;                            /* Interrupt Flag Clear         */
}
```

```
/*********************************************************/
/*  Forward Slue Up                                      */
/*********************************************************/
void fslueup ( void )
{
    GRA0 = uptbl[sluecnt];                  /* GRA Set Slue Up/Down table      */
    sluecnt++;

    PDR6 = pattbl[tzcnt];                   /* PDR6 Set Output Pattern         */
    if(tzcnt==7)
        tzcnt = 0;
    else
        tzcnt++;
}

/*********************************************************/
/*  Forward Slue Down                                    */
/*********************************************************/
void fsluedwn ( void )
{
    GRA0 = uptbl[sluecnt];                  /* GRA Set Slue Up/Down table      */
    sluecnt--;

    PDR6 = pattbl[tzcnt];                   /* PDR6 Set Output Pattern         */
    if(tzcnt==7)
        tzcnt = 0;
    else
        tzcnt++;
}

/*********************************************************/
/*  Forward Constant Speed                               */
/*********************************************************/
void fconst ( void )
{
    PDR6 = pattbl[tzcnt];                   /* PDR6 Set Output Pattern         */
    if(tzcnt==7)
        tzcnt = 0;
    else
        tzcnt++;
}

/*********************************************************/
/*  Slue/Reverse Stop                                    */
/*********************************************************/
void frstop ( void )
{
    PDR6 = pattbl[tzcnt];                   /* PDR6 Set Output Pattern         */
}

/*********************************************************/
/*  Reverse Slue Up                                      */
/*********************************************************/
void rslueup ( void )
```

```
{
    GRA0 = uptbl[sluecnt];                    /* GRA Set Slue Up/Down table      */
    sluecnt++;

    PDR6 = pattbl[tzcnt];                     /* PDR6 Set Output Pattern         */
    if(tzcnt==0)
        tzcnt = 7;
    else
        tzcnt--;
}

/*************************************************************/
/*  Reverse Slue Down                                        */
/*************************************************************/
void rsluedwn ( void )
{
    GRA0 = uptbl[sluecnt];                    /* GRA Set Slue Up/Down table      */
    sluecnt--;

    PDR6 = pattbl[tzcnt];                     /* PDR6 Set Output Pattern         */
    if(tzcnt==0)
        tzcnt = 7;
    else
        tzcnt--;
}

/*************************************************************/
/*  Reverse Constant Speed                                   */
/*************************************************************/
void rconst ( void )
{
    PDR6 = pattbl[tzcnt];                     /* PDR6 Set Output Pattern         */
    if(tzcnt==0)
        tzcnt = 7;
    else
        tzcnt--;
}
```

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.12.22	—	初版発行

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。