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。

H8/300L SLP シリーズ

1-2 相励磁方式ステッピングモータ

要旨

H8/38024 の内蔵機能のうち，P63～P60 とタイマ F アウトプットコンペア機能を用いて，2 相ステッピングモータを，1-2 相励磁方式で制御します。

動作確認デバイス

H8/38024

目次

1. 仕様

H8/38024 の内蔵機能のうち，P63～P60 とタイマ F アウトプットコンペア機能を用いて 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，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 Slew-up および Slew-down 処理を行ないます。

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。

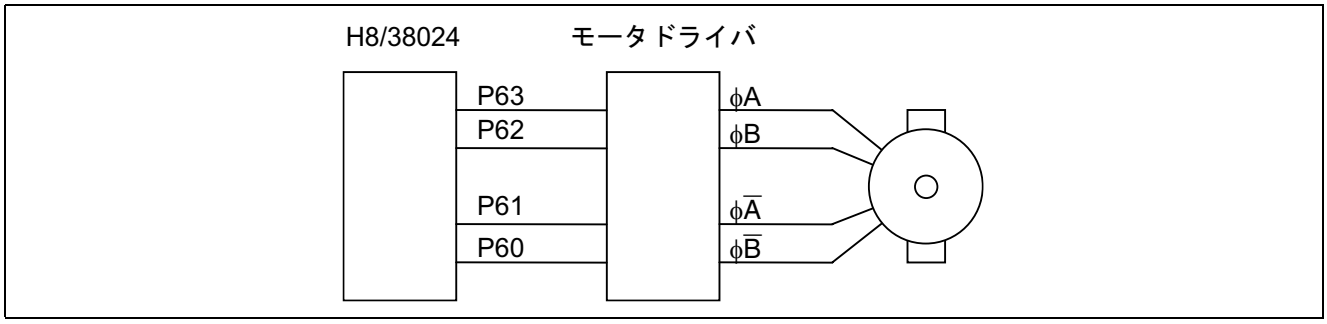


図 1　2 相ステッピングモータ制御の接続図

2. 使用機能説明

2.1 モータ仕様

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。KP6P8-701 の標準仕様を表 1 に示します。

表 1　KP6P8-701 標準仕様

項目	値
型名	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 使用機能

ステッピングモータ制御における H8/38024 の使用機能について説明します。本タスク例における使用機能のブロック図を図 2 に示します。

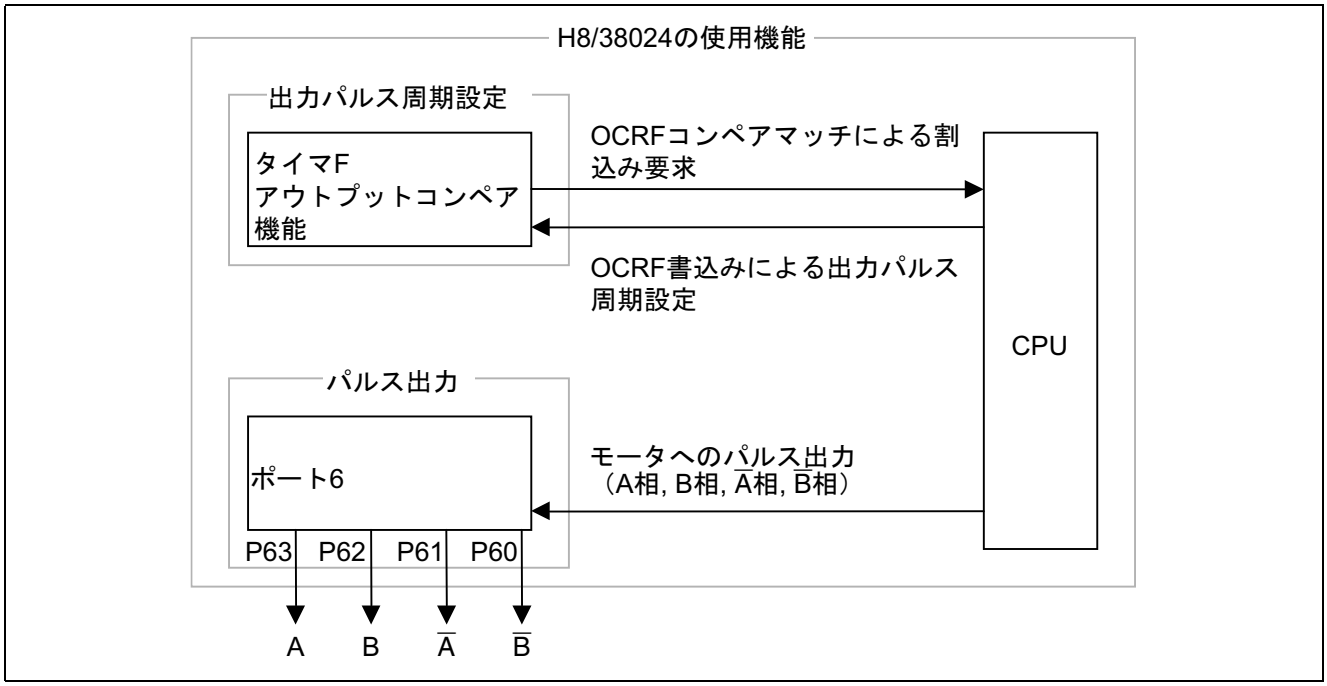


図 2　H8/38024 の使用機能

2.3 タイマ F 機能

タイマ F は，アウトプットコンペア機能を内蔵した 16 ビットのタイマです。本タスク例では，タイマ F のアウトプットコンペア機能を使用します。タイマ F のブロック図を図 3 に示します。以下にタイマ F 機能のブロック図について説明します。

- タイマコントロールレジスタ F (TCRF)
  8 ビットのリード/ライト可能なレジスタで，16 ビットモード，8 ビットモードの切換え，4 種類の内部クロックおよび外部イベントの選択を行ないます。
- タイマコントロールステータスレジスタ F (TCSRF)
  8 ビットのレジスタで，カウンタクリアの選択，オーバフローフラグのセット，コンペアマッチフラグのセット，オーバフローによる割込み要求の許可の制御を行ないます。
- タイマカウンタ F (TCF) (TCFH, TCFL)
  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。入力するクロックはシステムクロックを 4 分周，16 分周，32 分周したもの，サブクロックを 4 分周したもの，または外部クロックの計 5 種類から選択可能です。本タスク例では，TCF の入力クロックにシステムクロックを 4 分周 (ϕ/4) したものを選択しています。
- アウトプットコンペアレジスタ F (OCRF) (OCRFH, OCRFL)
  16 ビットのリード/ライト可能なレジスタで，OCRF の内容は TCF と常に比較されています。両者の値が一致するとコンペアマッチ FH が発生し，割込みが発生します。

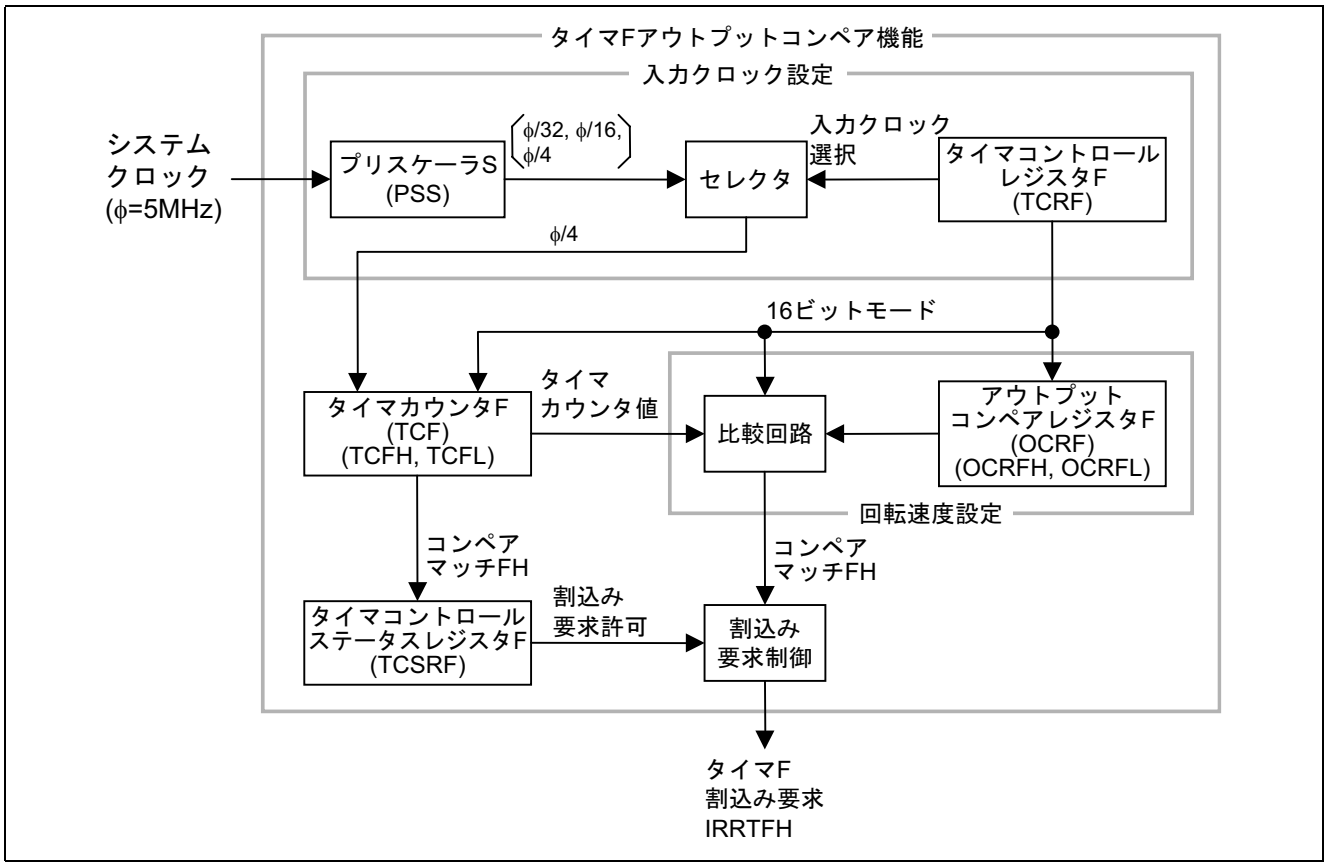


図 3　タイマ F のブロック図

2.4 ポートの設定

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。

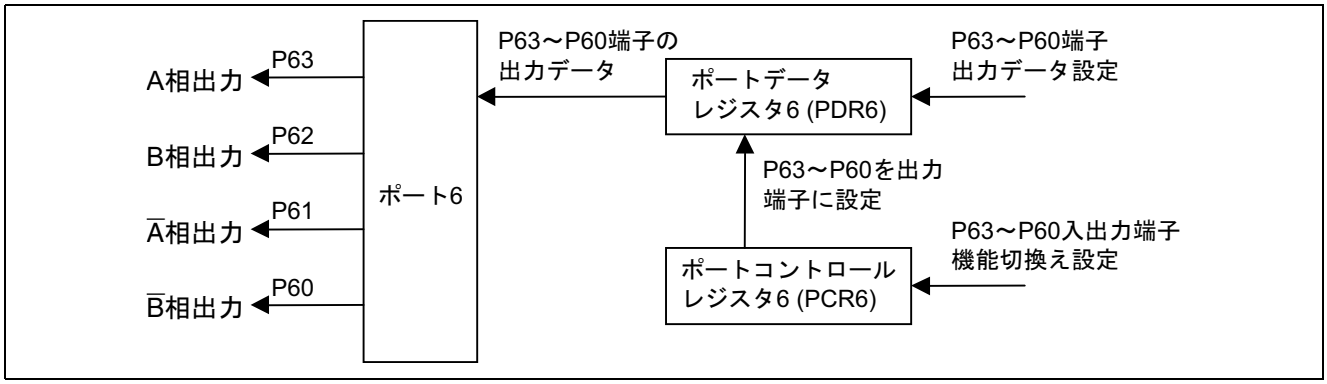


図 4　ポート 6 機能のブロック図

2.5 機能割付け

本タスク例の機能割付けを表 2 に示します。

表 2　機能割付け

機能	機能割付け
PSS	システムクロック (φ) を入力とする 13 ビットのアップカウンタ
TCRF	TCF の入力クロックを設定。タイマ F を 16 ビットモードに設定。
TCSRF	コンペアマッチフラグ，TCF のクリア設定，TCF のクリア方法の設定
TCF (TCFH, TCFL)	φ/16 をクロック入力とする 16 ビットのカウンタ
OCRF (OCRFH, OCRFL)	ステッピングモータ 1 ステップ時間の設定
PDR6	P63～P60 からステッピングモータ励磁相駆動信号を出力
PCR6	P63～P60 を出力端子に設定
IENTFH	タイマ FH 割込み要求の許可
IRRTFH	タイマ FH 割込み要求フラグ

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，以下のとおりです。

- 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, $\overline{A}$ 相→$\overline{A}$ 相→$\overline{A}$, $\overline{B}$ 相→$\overline{B}$ 相→$\overline{B}$, 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→$\overline{B}$ 相→$\overline{A}$, $\overline{B}$ 相→$\overline{A}$ 相→B, $\overline{A}$ 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。

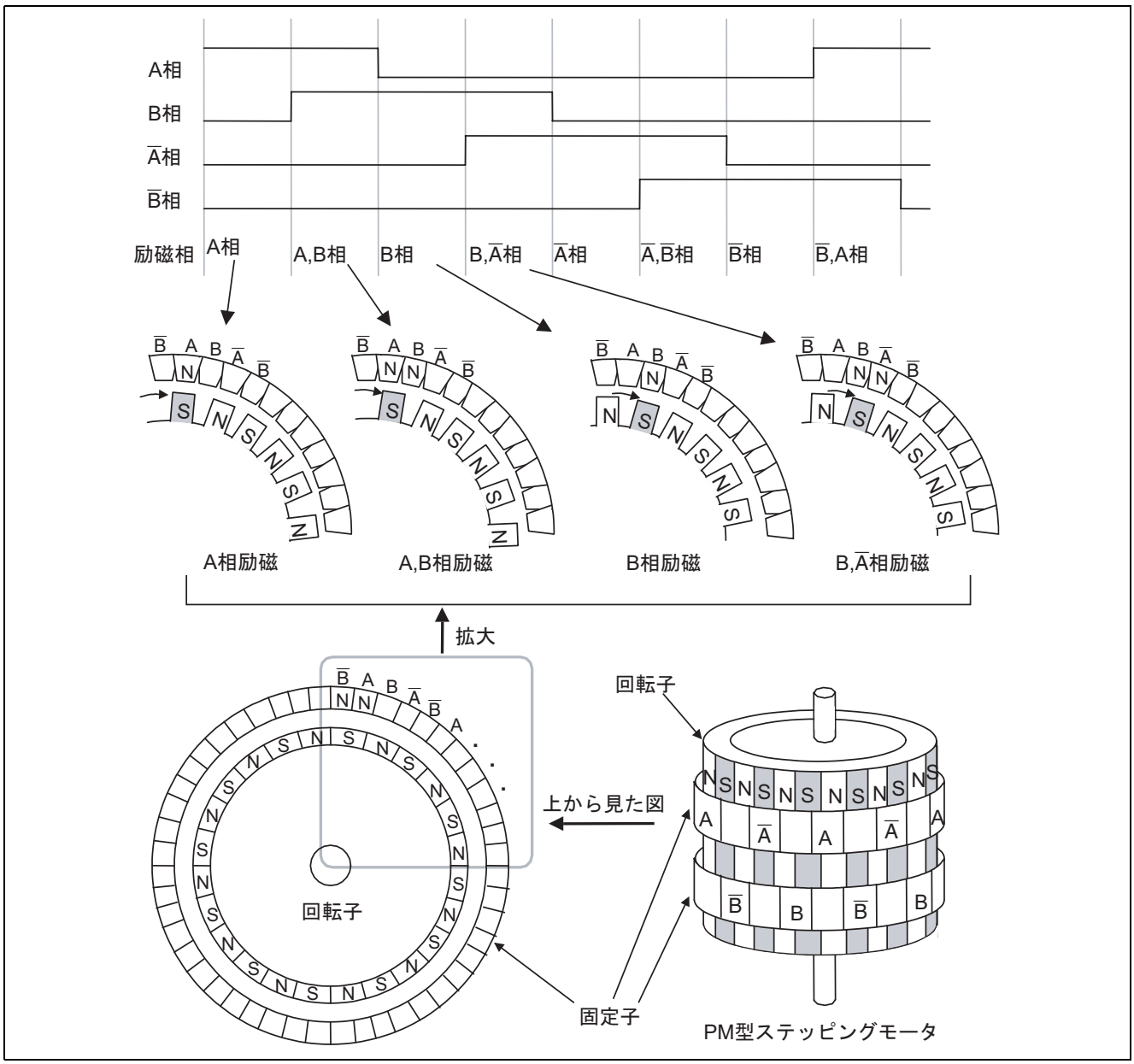


図 5　ステッピングモータ動作例

3.2 Slew-up, Slew-down 動作

Slew-up, Slew-down 動作を行なうことにより，モータの脱調を防止します。ここで脱調とは，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，回転しないことを言います。これを防止する対策として Slew-up および Slew-down を行ないます。動作原理を以下に示します。

- 徐々にパルス周期を短くし，設定した数のパルスを出力する。(Slew-up)
- 一定のパルス周期で設定した数のパルスを出力する。(定速)
- 徐々にパルス周期を長くし，設定した数のパルスを出力する。(Slew-down)

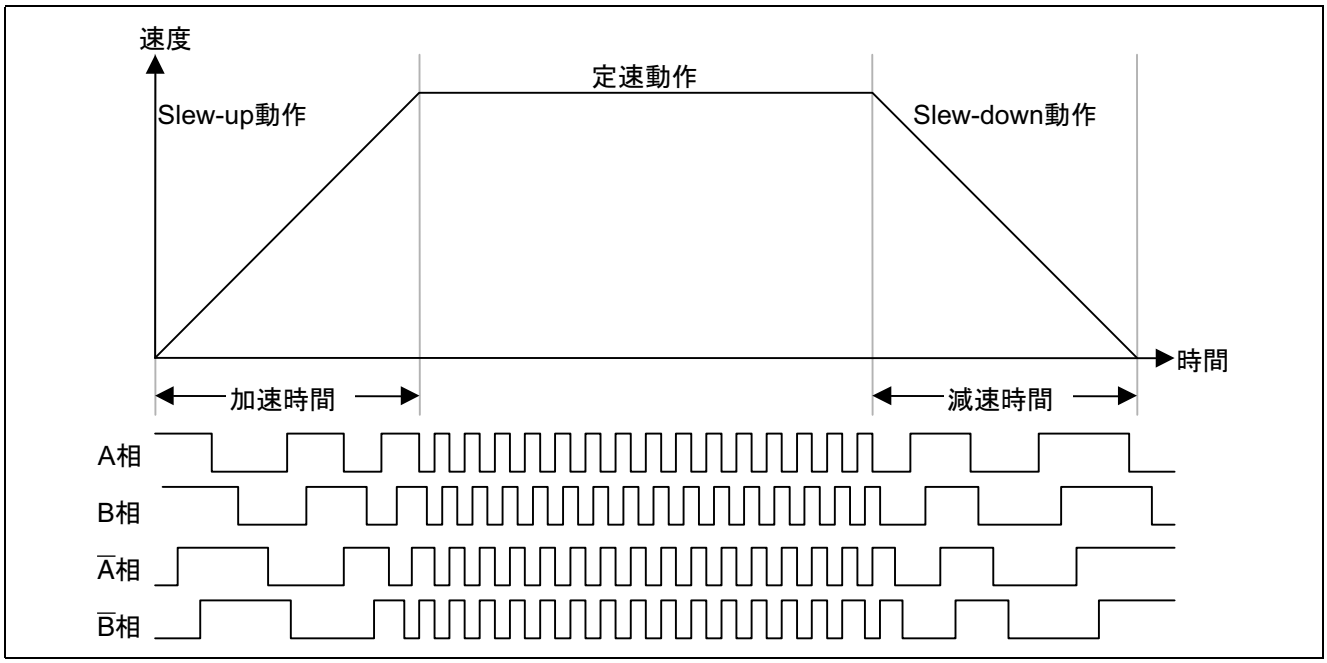


図 6　Slew-up, Slew-down 動作例

3.3 ステッピングモータ制御

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。

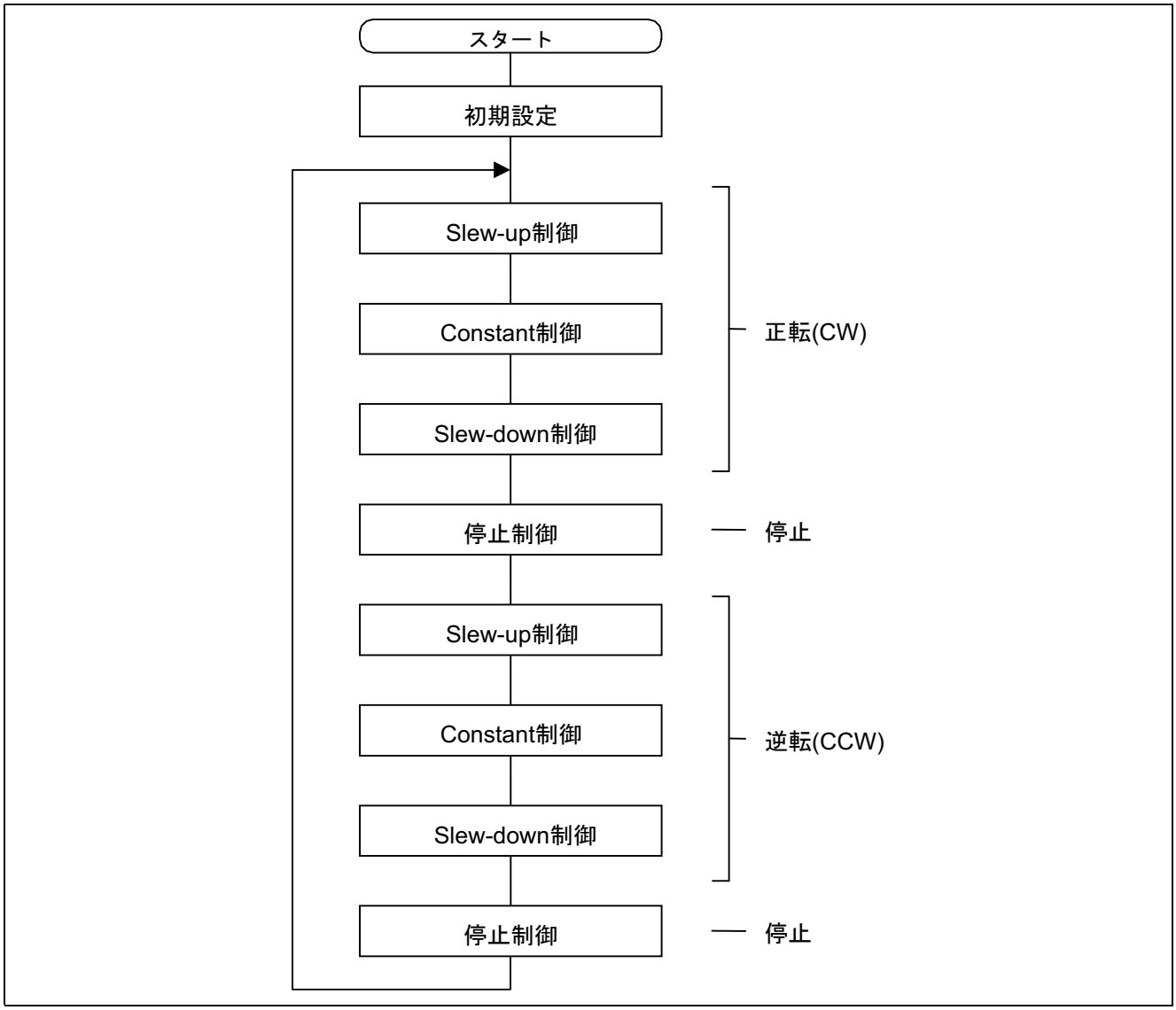


図 7　ステッピングモータ制御のフローチャート

3.4 タイマ F 割込み時間の計算式

タイマ F 割込み時間は，アウトプットコンペアレジスタ (OCRF) を設定することにより，以下のとおり計算できます。

$$\text{タイマF割込み時間} = \frac{\text{OCRF} + 1}{(\text{システムクロック}\phi/4)}$$

$$= \frac{\text{OCRF} + 1}{(5\text{MHz}/4)}$$

$$= 0.8 \times (\text{OCRF} + 1)\ [\mu\text{s}]$$

3.5 正転時 Slew-up 制御

正転時 Slew-up 制御の動作原理を図 8 に示します。

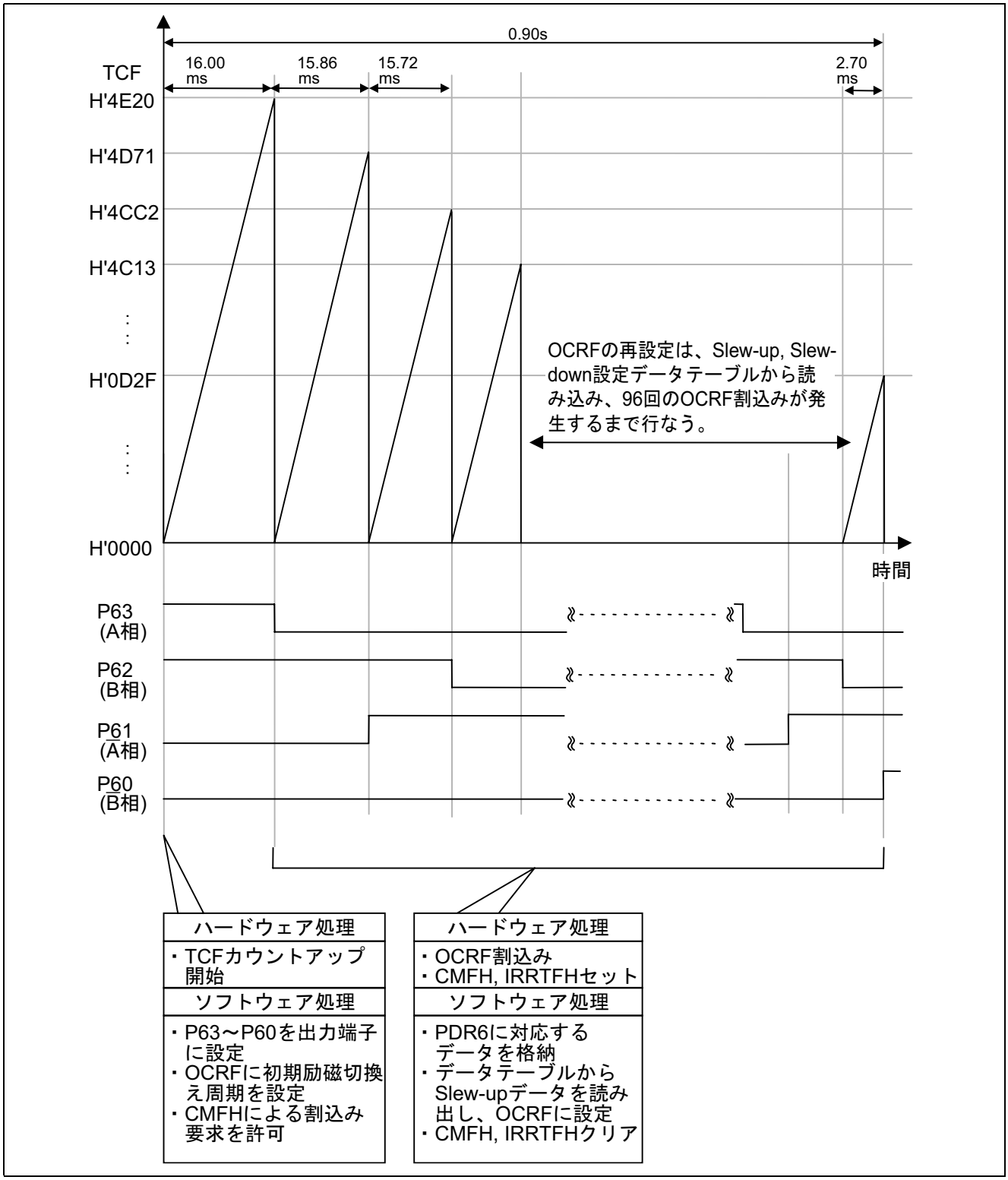


図 8　正転時 Slew-up 制御の動作原理

3.6 正転時 Constant 制御

正転時 Constant 制御の動作原理を図 9 に示します。

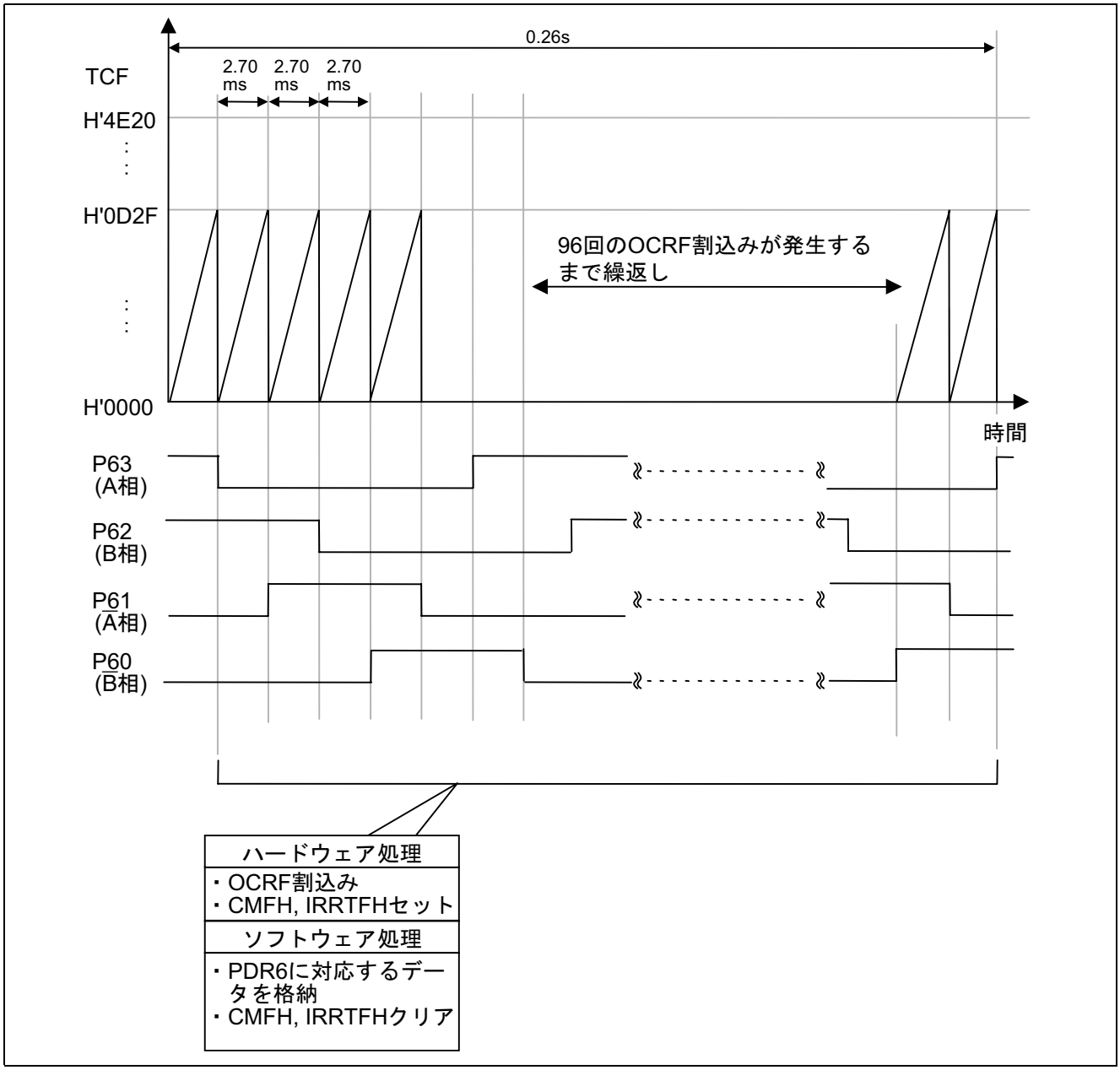


図 9　正転時 Constant 制御の動作原理

3.7 正転時 Slew-down 制御

正転時 Slew-down 制御の動作原理を図 10 に示します。

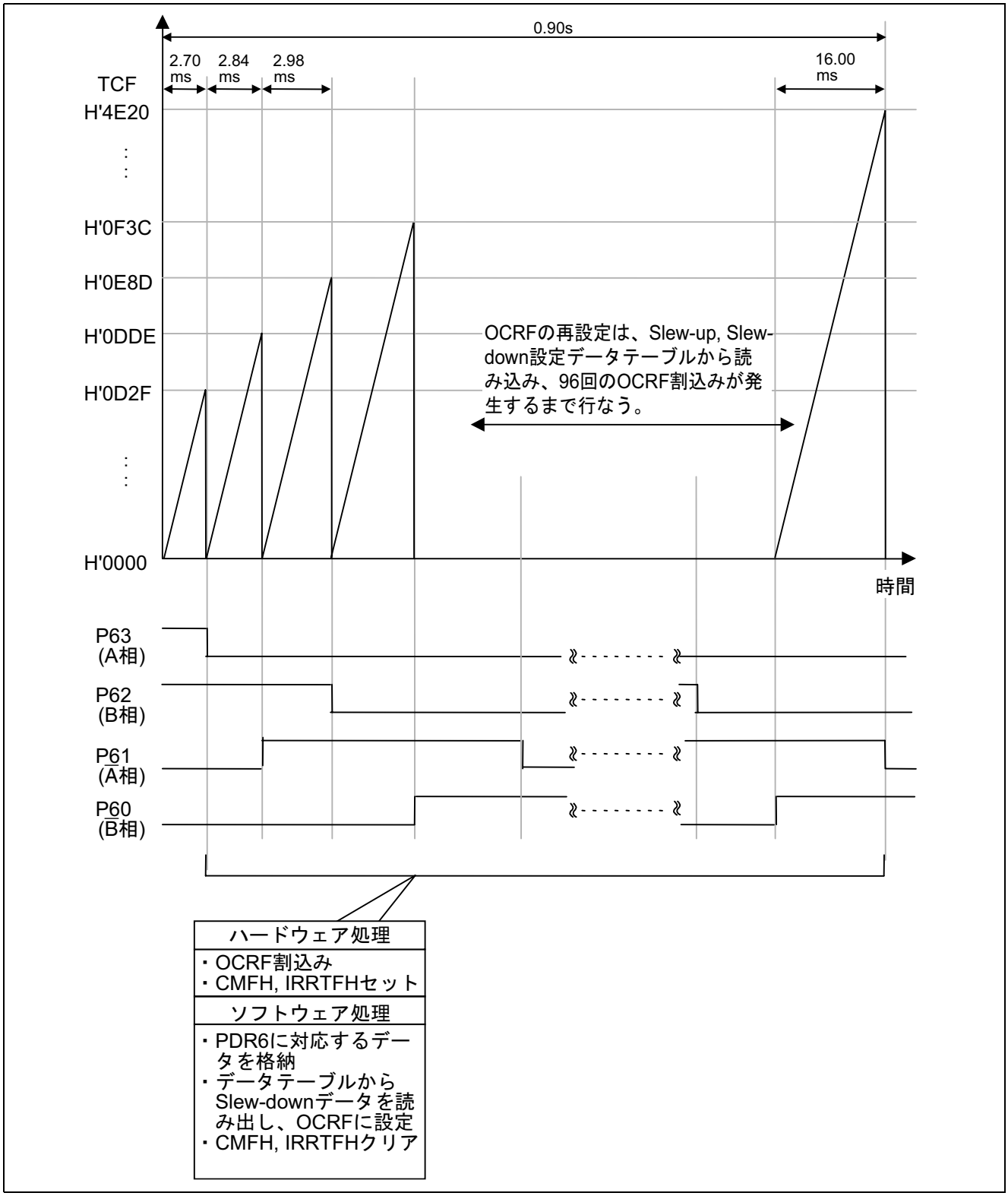


図 10　正転時 Slew-down 制御の動作原理

3.8 停止制御

停止制御の動作原理を図 11 に示します。

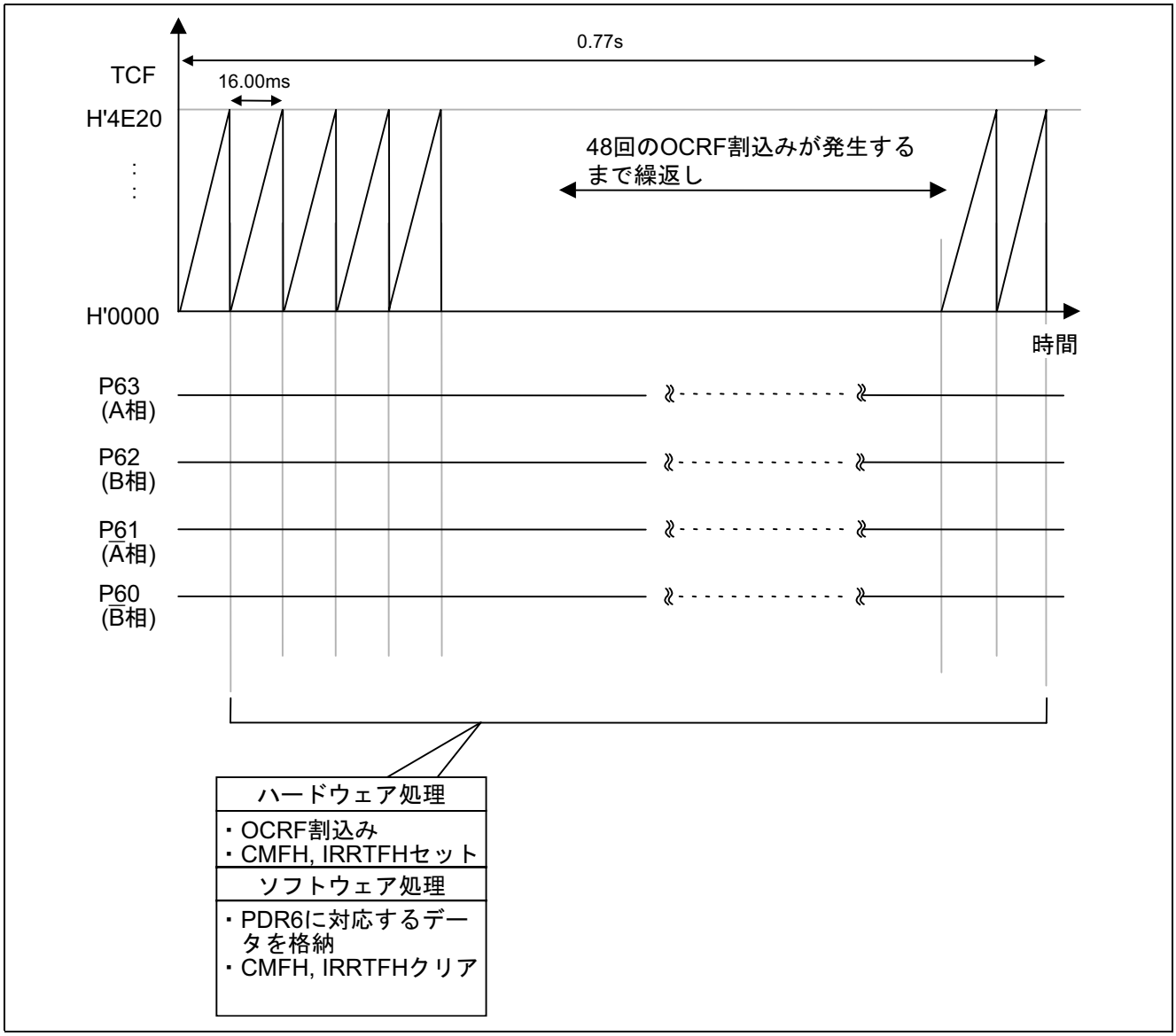


図 11　停止制御の動作原理

3.9 逆転時 Slew-up 制御

逆転時 Slew-up 制御の動作原理を図 12 に示します。

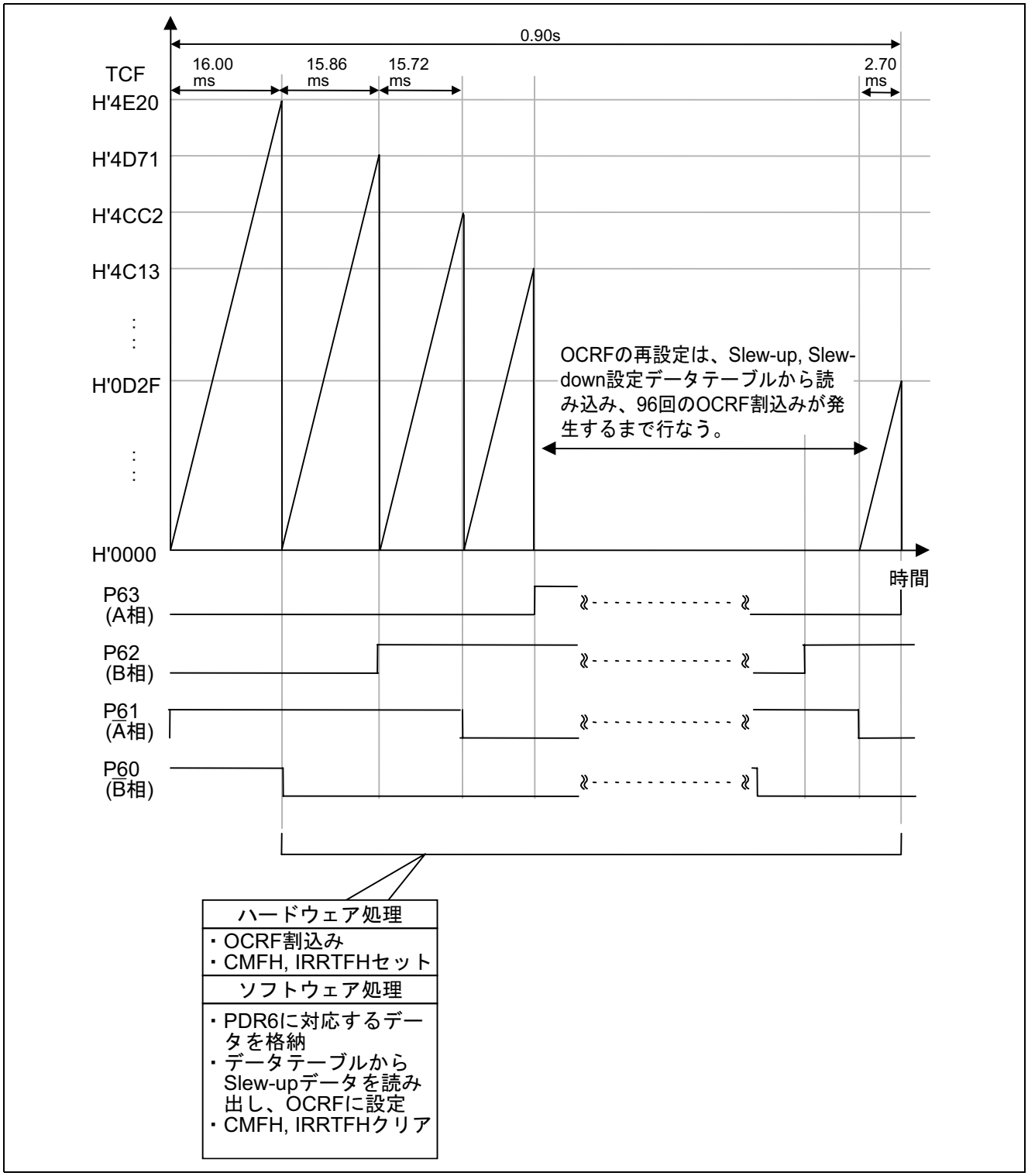


図 12　逆転時 Slew-up 制御の動作原理

3.10 逆転時 Constant 制御

逆転時 Constant 制御の動作原理を図 13 に示します。

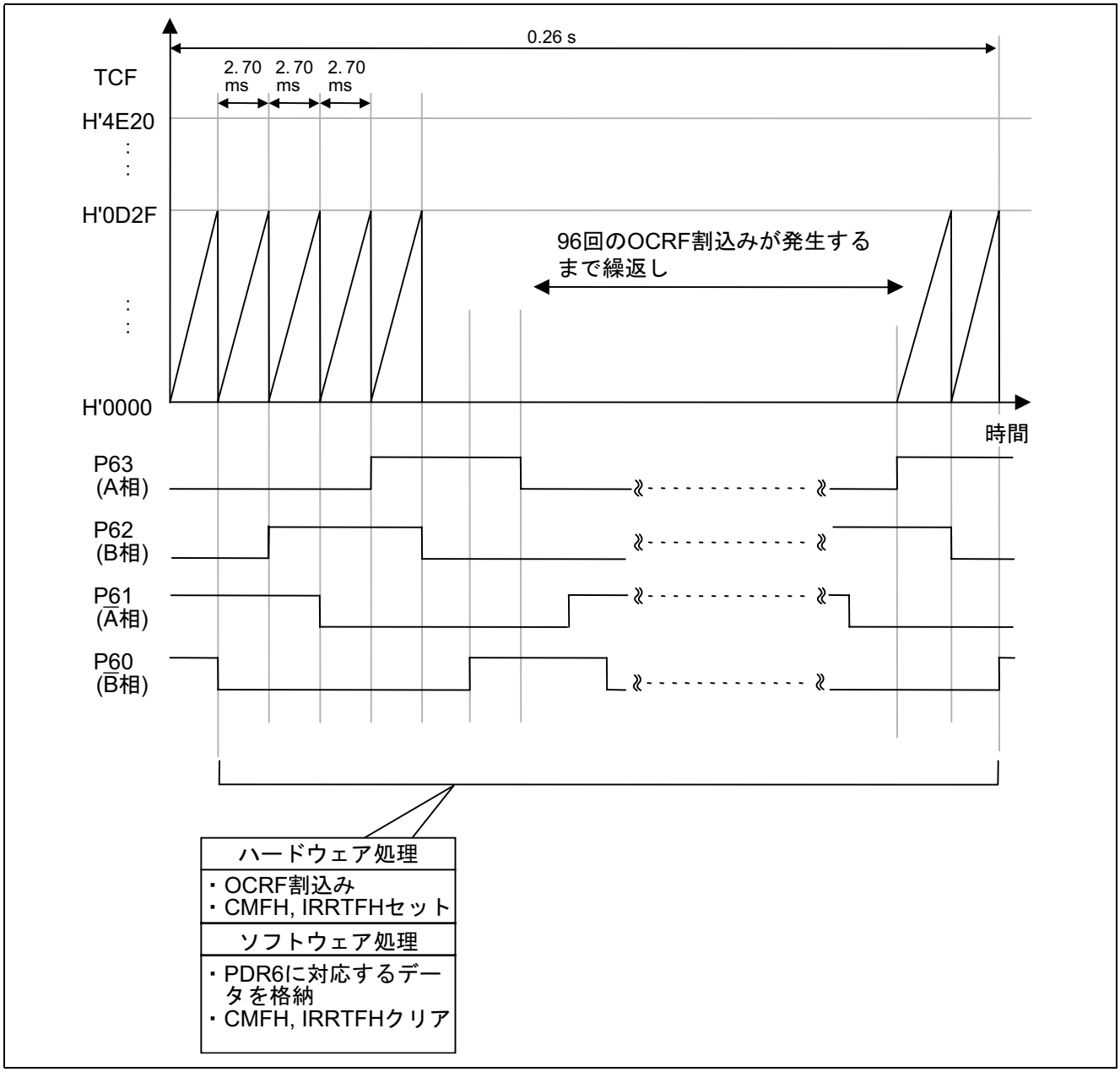


図 13　逆転時 Constant 制御の動作原理

3.11 逆転時 Slew-down 制御

逆転時 Slew-down 制御の動作原理を図 14 に示します。

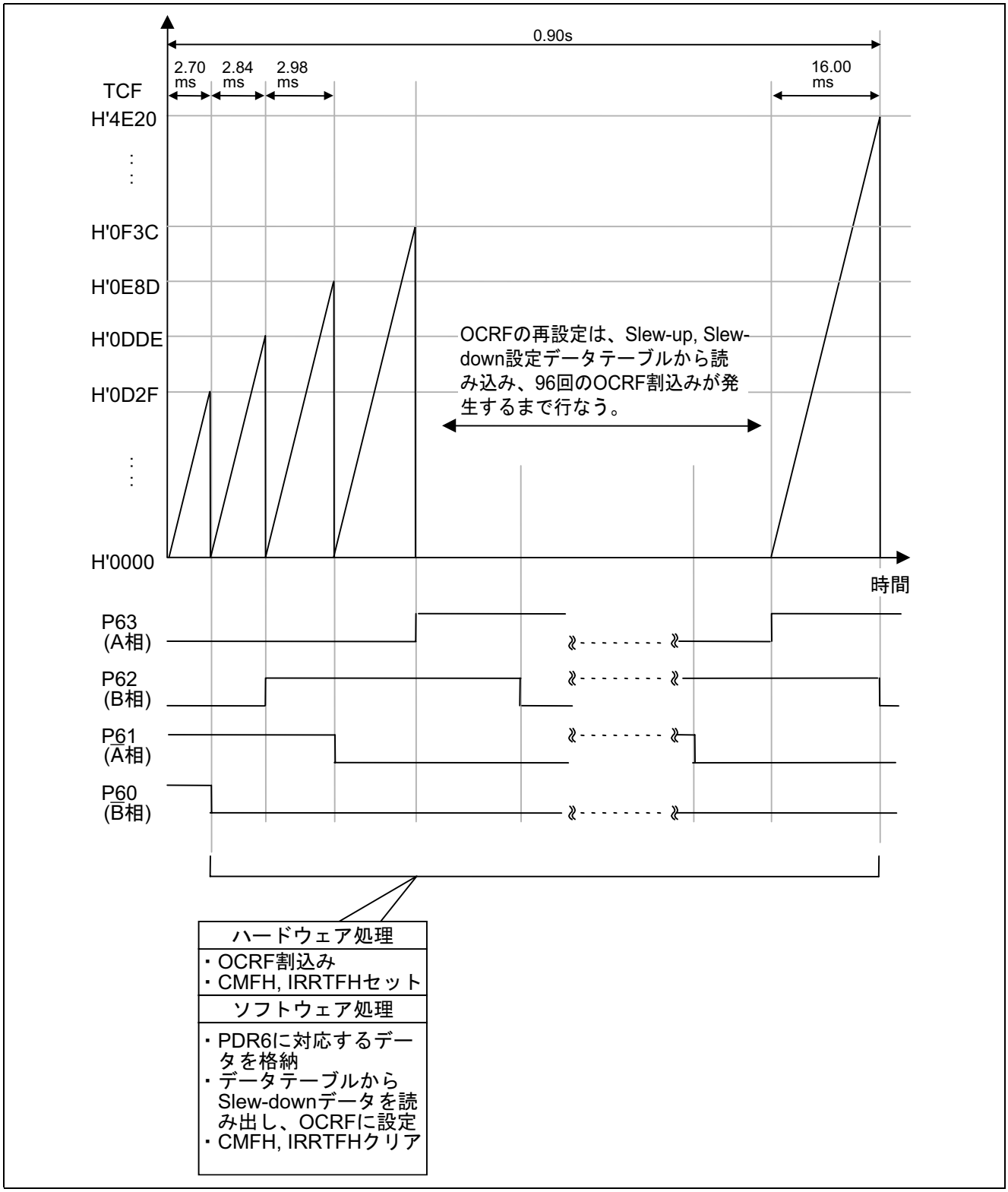


図 14　逆転時 Slew-down 制御の動作原理

4. ソフトウェア説明

4.1 モジュール一覧

本タスク例のモジュールを表 3 に示します。本タスク例の階層構造を図 15 に示します。

表 3　モジュール説明

ラベル名	機能
main	**メインルーチン** **グローバル変数，**I/O **ポート，タイマ** F **の初期設定，および割込みの許可を行なう**
tfhint	**タイマ** FH **割込み処理** **ステッピングモータのメインルーチン**
fslueup	**正転時の** Slew-up **制御**
fsluedwn	**正転時の** Slew-down **制御**
fconst	**正転時の** Constant **制御**
frstop	**正転/逆転の回転停止**
rslueup	**逆転時の** Slew-up **制御**
rsluedwn	**逆転時の** Slew-down **制御**
rconst	**逆転時の** Constant **制御**

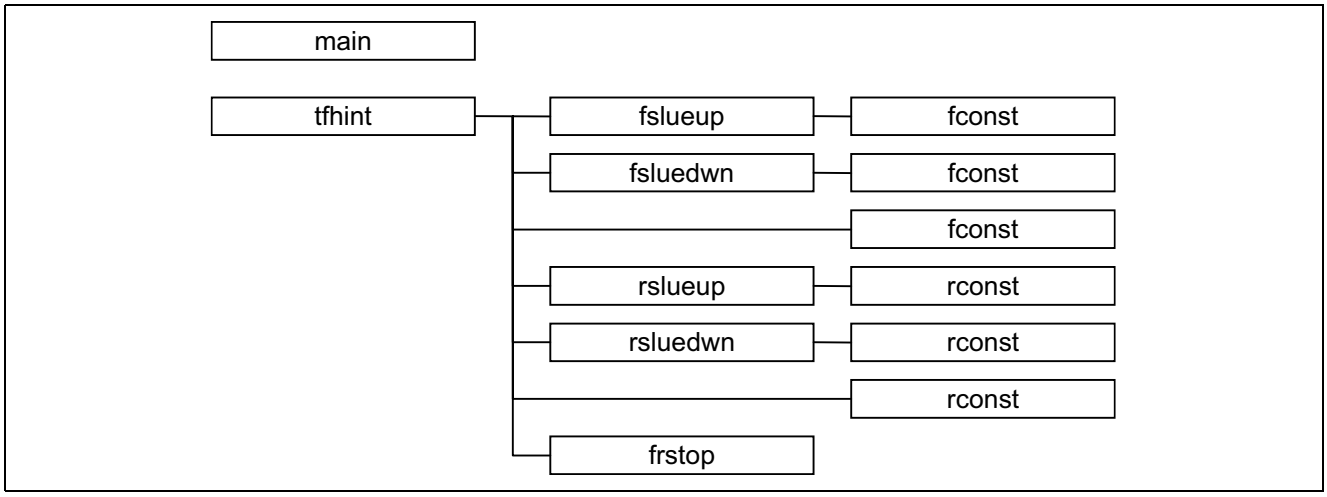


図 15　階層構造

4.2 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換えデータテーブル

```
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),Ā 相 (P61) を励磁する。
   0x02, ・・・・Ā 相 (P61) を励磁する。
   0x03, ・・・・Ā 相 (P61),B̄ 相 (P60) を励磁する。
   0x01, ・・・・B̄ 相 (P60) を励磁する。
   0x09  ・・・・B̄ 相 (P60),A 相 (P63) を励磁する。
};
```

- Slew-up, Slew-down 設定データテーブル

```
uptbl[96] = {
   0x4E20,0x4D71,0x4CC2,0x4C13,0x4B64,0x4AB5,0x4A06,0x4957,0x48A8,0x47F9,
   0x474A,0x469B,0x45EC,0x453D,0x448E,0x43DF,0x4330,0x4281,0x41D2,0x4123,
   0x4074,0x3FC5,0x3F16,0x3E67,0x3DB8,0x3D09,0x3C5A,0x3BAB,0x3AFC,0x3A4D,
   0x399E,0x38EF,0x3840,0x3791,0x36E2,0x3633,0x3584,0x34D5,0x3426,0x3377,
   0x32C8,0x3219,0x316A,0x30BB,0x300C,0x2F5D,0x2EAE,0x2DFF,0x2D50,0x2CA1,
   0x2BF2,0x2B43,0x2A94,0x29E5,0x2936,0x2887,0x27D8,0x2729,0x267A,0x25CB,
   0x251C,0x246D,0x23BE,0x230F,0x2260,0x21B1,0x2102,0x2053,0x1FA4,0x1EF5,
   0x1E46,0x1D97,0x1CE8,0x1C39,0x1B8A,0x1ADB,0x1A2C,0x197D,0x18CE,0x181F,
   0x1770,0x16C1,0x1612,0x1563,0x14B4,0x1405,0x1356,0x12A7,0x11F8,0x1149,
   0x109A,0x0FEB,0x0F3C,0x0E8D,0x0DDE,0x0D2F
};
```

Slew-up, Slew-down 動作時の OCRF 割込みで uptbl[]を順次 OCRF に書き込む。書込みは, 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。

4.3 モジュール説明

4.3.1 main 関数

(1) モジュール仕様

機能概要：グローバル変数，I/O ポート，タイマ F の初期設定，および割込みの許可を行なう。

表 4 モジュール仕様

	型	変数名	内容
引数	なし	なし	なし
使用 RAM	unsigned char	tcnt	ステッピングモータの励磁データである配列 pattbl[]の要素
	unsigned char	sluecnt	Slew-up, Slew-down 時に使用する配列 uptbl[]の要素
	unsigned char	nextmode	ステッピングモータの動作モードを設定する 0：正転時 Slew-up 制御　4：逆転時 Slew-up 制御 1：正転時 Constant 制御　5：逆転時 Constant 制御 2：正転時 Slew-down 制御　6：逆転時 Slew-down 制御 3：停止制御　7：停止制御
	unsigned short	modecnt	ステッピングモータの動作モードの割込み回数を設定する
使用 ROM	unsigned char	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 short	uptbl[96]	Slew-up, Slew-down 時の割込み時間データテーブル

(2) 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。

- TCRF　タイマコントロールレジスタ F　アドレス：H'FFB6

ビット	ビット名	設定値	機能
6 5 4	CKSH2 CKSH1 CKSH0	0 0 0	クロックセレクト H CKSH2～0=B'000,B'001,B'010：TCF は 16 ビットカウンタとして動作
2 1 0	CKSL2 CKSL1 CKSL0	1 1 0	クロックセレクト L CKSH2～0=B'110：TCF は内部クロックφ/4 でカウント

- TCSRF　タイマコントロールステータスレジスタ F　アドレス：H'FFB7

ビット	ビット名	設定値	機能
6	CMFH	0	コンペアマッチフラグ H CMFH=0：コンペアマッチ F が発生していない CMFH=1：コンペアマッチ F が発生した
4	CCLRH	1	CCLRH=0：16 ビットモード時，コンペアマッチによる TCF のクリアを禁止 CCLRH=1：16 ビットモード時，コンペアマッチによる TCF のクリアを許可

- TCF　16 ビットタイマカウンタ F　アドレス：H'FFB8
  機能：φ/4 を入力とする 16 ビットのカウンタ
  設定値：H'0000

- OCRF　16 ビットアウトプットコンペアレジスタ F　アドレス：H'FFBA
  機能：OCRF の設定値と TCF 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

- 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

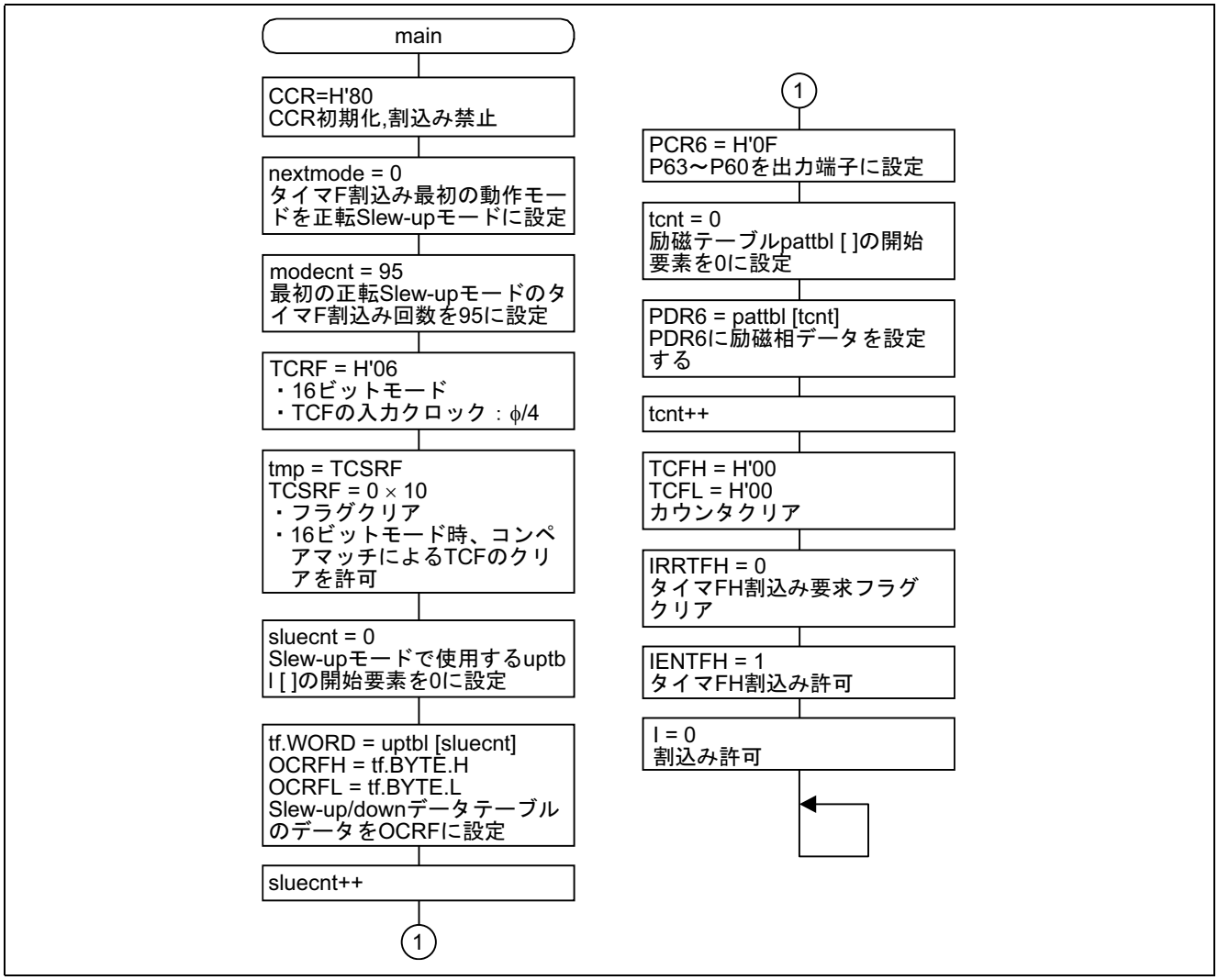
- IENR2　　割込み許可レジスタ 2　　アドレス：H'FFF4

ビット	ビット名	設定値	機能
3	IENTFH	1	**タイマ** FH **割込みイネーブル** IENTFH=0：**タイマ** FH **割込み要求を禁止** IENTFH=1：**タイマ** FH **割込み要求を許可**

- IRR2　　割込み要求レジスタ 2　　アドレス：H'FFF7

ビット	ビット名	設定値	機能
3	IRRTFH	0	**タイマ** FH **割込み要求フラグ** IRRTFH=0：**タイマ** FH **割込みが要求されていない** IRRTFH=1：**タイマ** FH **割込みが要求されている**

(3) フローチャート


main
CCR=H'80
CCR初期化,割込み禁止
nextmode = 0
タイマF割込み最初の動作モードを正転Slew-upモードに設定
modecnt = 95
最初の正転Slew-upモードのタイマF割込み回数を95に設定
TCRF = H'06
・16ビットモード
・TCFの入力クロック：φ/4
tmp = TCSRF
TCSRF = 0 × 10
・フラグクリア
・16ビットモード時、コンペアマッチによるTCFのクリアを許可
sluecnt = 0
Slew-upモードで使用するuptbl [ ]の開始要素を0に設定
tf.WORD = uptbl [sluecnt]
OCRFH = tf.BYTE.H
OCRFL = tf.BYTE.L
Slew-up/downデータテーブルのデータをOCRFに設定
sluecnt++
1
1
PCR6 = H'0F
P63～P60を出力端子に設定
tcnt = 0
励磁テーブルpattbl [ ]の開始要素を0に設定
PDR6 = pattbl [tcnt]
PDR6に励磁相データを設定する
tcnt++
TCFH = H'00
TCFL = H'00
カウンタクリア
IRRTFH = 0
タイマFH割込み要求フラグクリア
IENTFH = 1
タイマFH割込み許可
I = 0
割込み許可

4.3.2 tfhint 関数

(1) モジュール仕様

機能概要：タイマ FH 割込み処理/ステッピングモータのメイン処理

表 5 モジュール仕様

	型	変数名	内容
引数	なし	なし	なし
使用 RAM	unsigned char	tcnt	ステッピングモータの励磁データである配列 pattbl[]の要素
	unsigned char	sluecnt	Slew-up, Slew-down 時に使用する配列 uptbl[]の要素
	unsigned char	nextmode	ステッピングモータの動作モードを設定する 0：正転時 Slew-up 制御　4：逆転時 Slew-up 制御 1：正転時 Constant 制御　5：逆転時 Constant 制御 2：正転時 Slew-down 制御　6：逆転時 Slew-down 制御 3：停止制御　7：停止制御
	unsigned short	modecnt	ステッピングモータの動作モードの割込み回数を設定する

(2) 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。

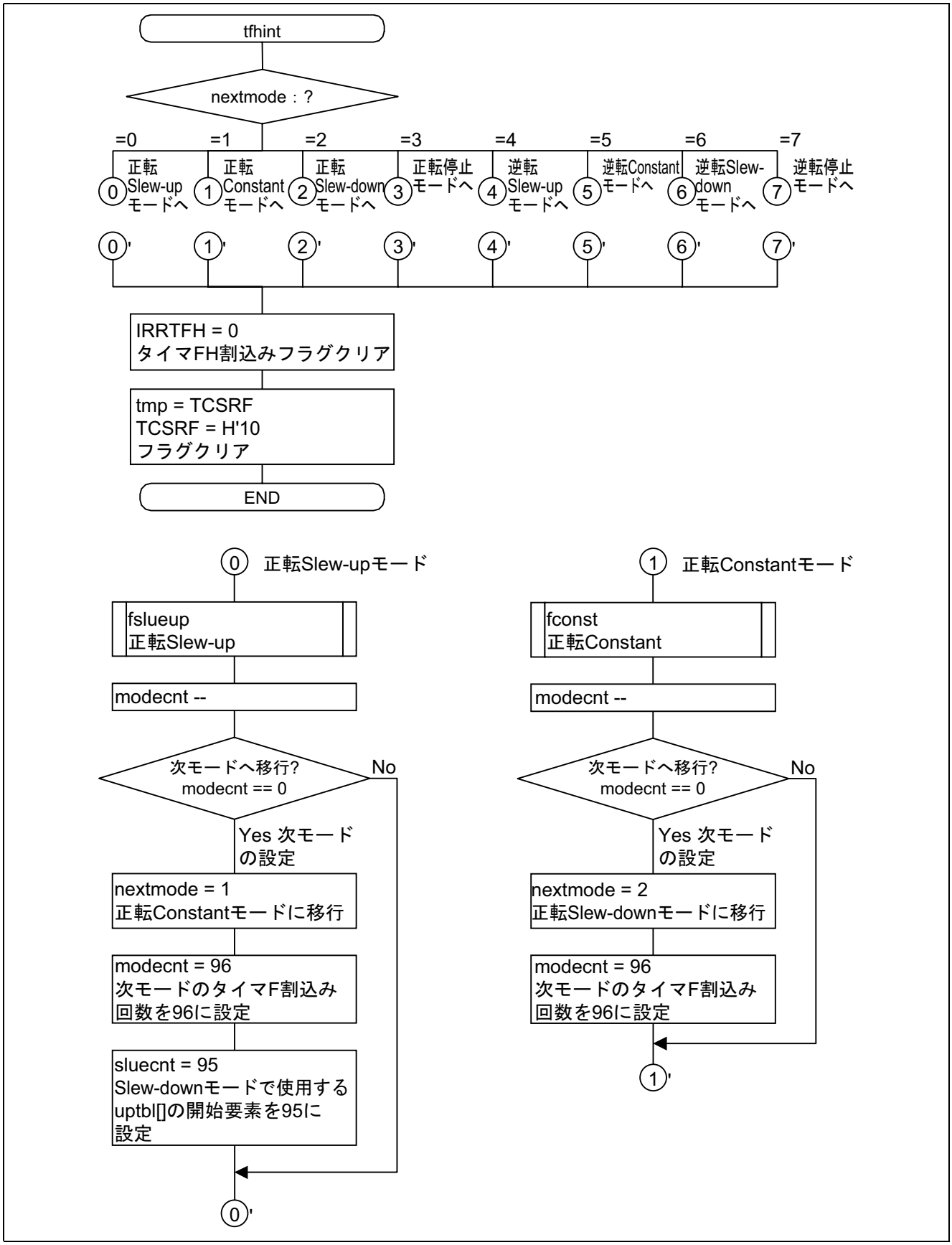
- TCSRF　タイマコントロールステータスレジスタ F　アドレス：H'FFB7

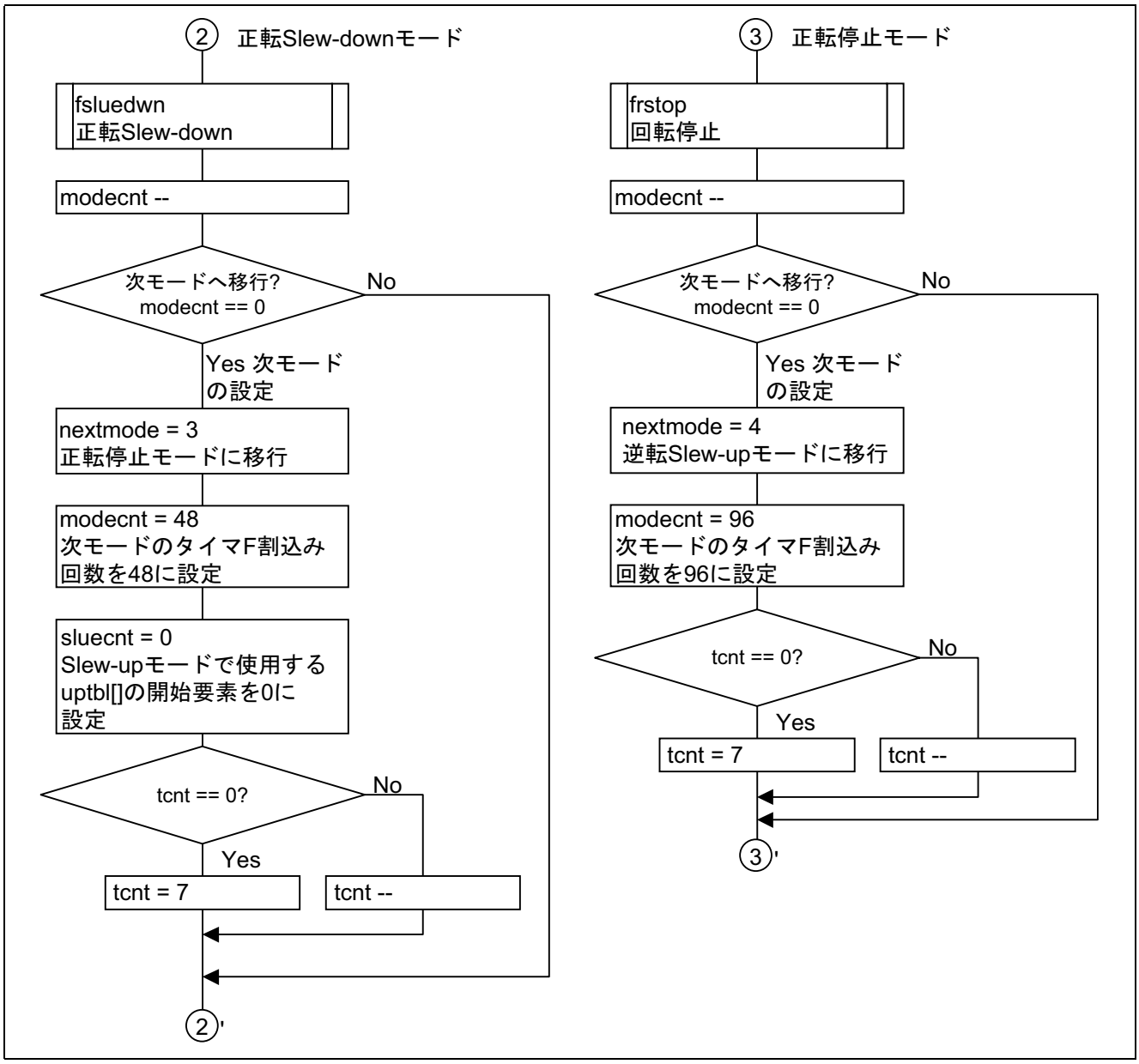
ビット	ビット名	設定値	機能
6	CMFH	0	コンペアマッチフラグ H CMFH=0：コンペアマッチ F が発生していない CMFH=1：コンペアマッチ F が発生した
4	CCLRH	1	CCLRH=0：16 ビットモード時，コンペアマッチによる TCF のクリアを禁止 CCLRH=1：16 ビットモード時，コンペアマッチによる TCF のクリアを許可

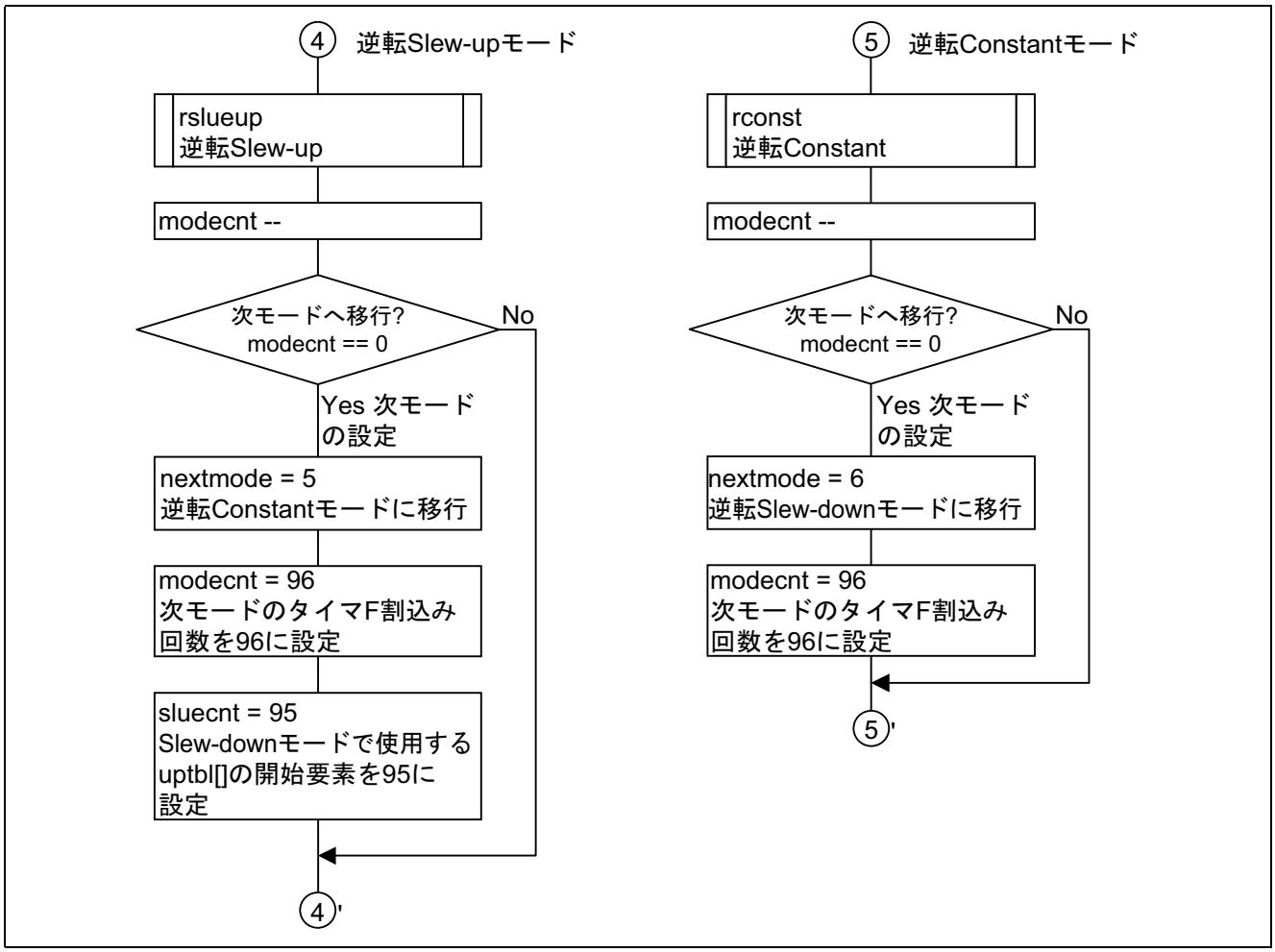
- IRR2　割込み要求レジスタ 2　アドレス：H'FFF7

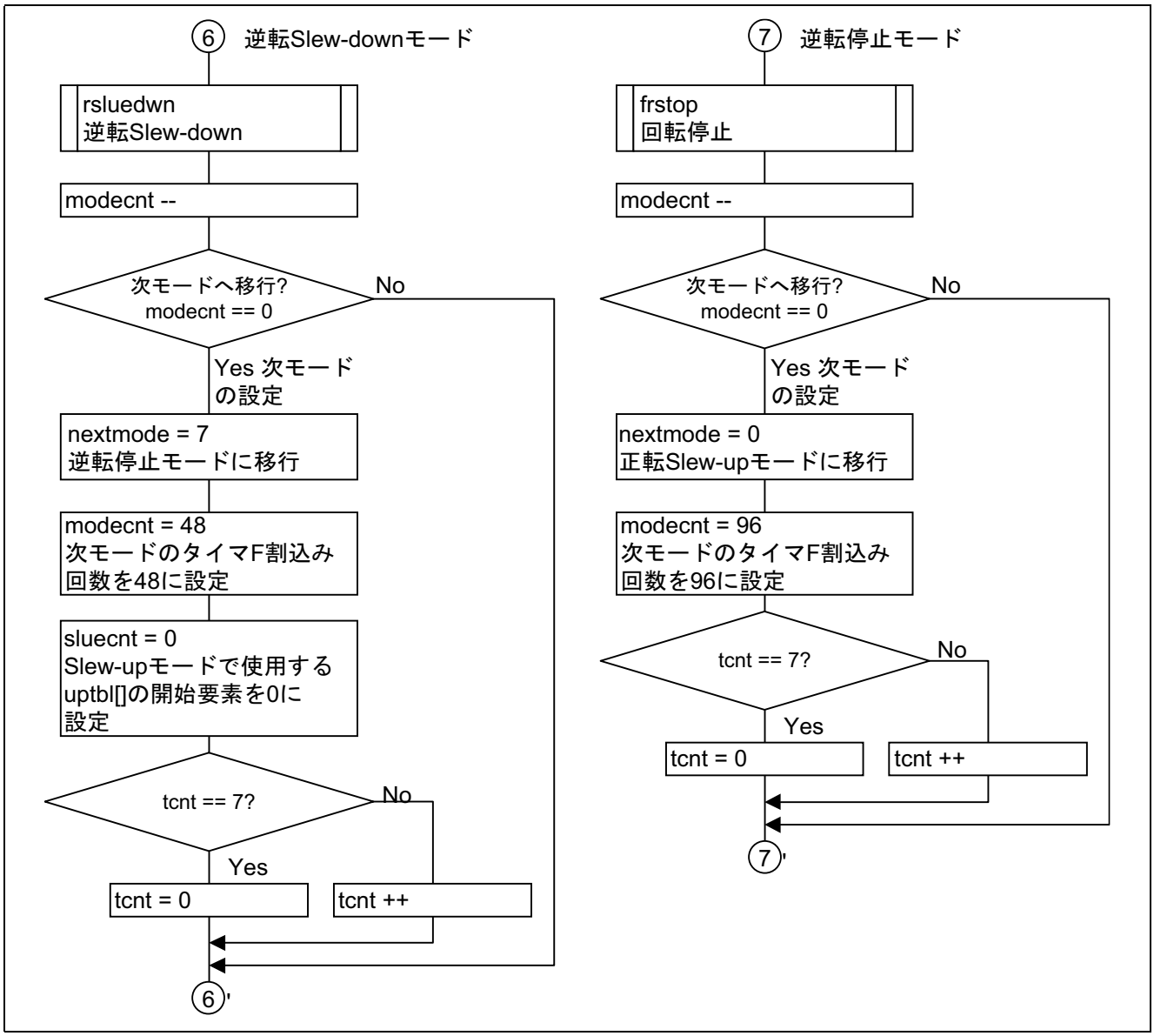
ビット	ビット名	設定値	機能
3	IRRTFH	0	タイマ FH 割込み要求フラグ IRRTFH=0：タイマ FH 割込みが要求されていない IRRTFH=1：タイマ FH 割込みが要求されている

(3) フローチャート


tfhint
nextmode：?
=0
=1
=2
=3
=4
=5
=6
=7
正転 Slew-up モードへ
正転 Constant モードへ
正転 Slew-down モードへ
正転停止 モードへ
逆転 Slew-up モードへ
逆転Constant モードへ
逆転Slew-down モードへ
逆転停止 モードへ
IRRTFH = 0
タイマFH割込みフラグクリア
tmp = TCSRF
TCSRF = H'10
フラグクリア
END
正転Slew-upモード
fslueup
正転Slew-up
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F割込み回数を96に設定
sluecnt = 95
Slew-down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Slew-downモードに移行
modecnt = 96
次モードのタイマF割込み回数を96に設定


② 正転Slew-downモード
fsluedwn
正転Slew-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
modecnt = 48
次モードのタイマF割込み回数を48に設定
sluecnt = 0
Slew-upモードで使用するuptbl[]の開始要素を0に設定
tcnt == 0?
No
Yes
tcnt = 7
tcnt --
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Slew-upモードに移行
modecnt = 96
次モードのタイマF割込み回数を96に設定
tcnt == 0?
No
Yes
tcnt = 7
tcnt --
③'


4
逆転Slew-upモード
rslueup
逆転Slew-up
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F割込み
回数を96に設定
sluecnt = 95
Slew-down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Slew-downモードに移行
modecnt = 96
次モードのタイマF割込み
回数を96に設定
5'


⑥ 逆転Slew-downモード
rsluedwn
逆転Slew-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
modecnt = 48
次モードのタイマF割込み
回数を48に設定
sluecnt = 0
Slew-upモードで使用する
uptbl[]の開始要素を0に
設定
tcnt == 7?
No
Yes
tcnt = 0
tcnt ++
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Slew-upモードに移行
modecnt = 96
次モードのタイマF割込み
回数を96に設定
tcnt == 7?
No
Yes
tcnt = 0
tcnt ++
⑦'

4.3.3 fslueup 関数

(1) モジュール仕様

機能概要：正転時の Slew-up 制御を行なう

表 6　モジュール仕様

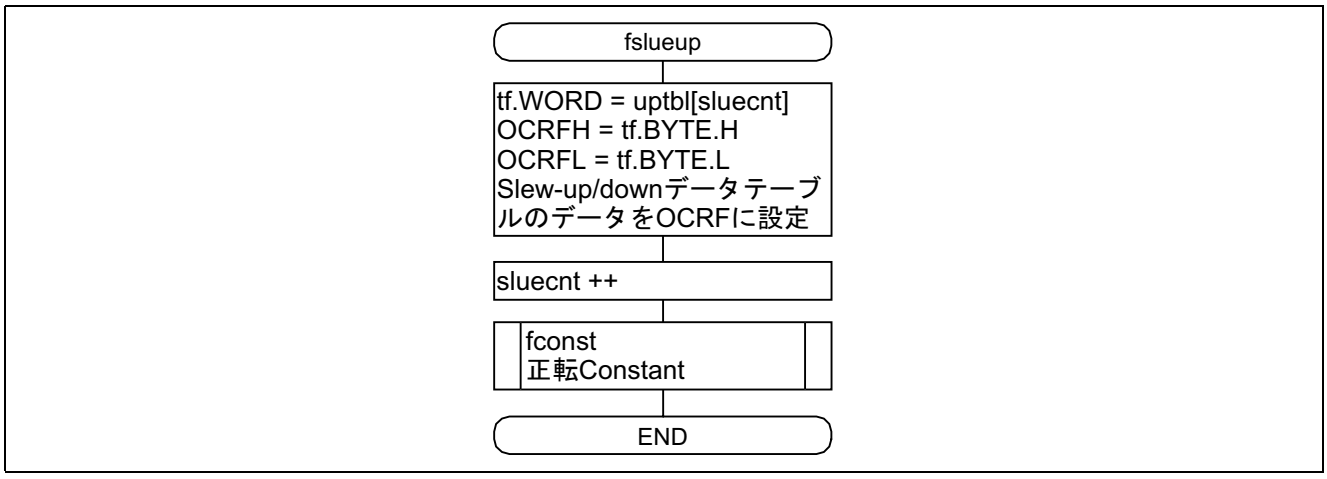
	型	変数名	内容
引数	なし	なし	なし
使用 RAM	unsigned char	sluecnt	Slew-up, Slew-down 時に使用する配列 uptbl[]の要素
使用 ROM	unsigned short	uptbl[96]	Slew-up, Slew-down 時の割込み時間データテーブル

(2) 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。

- OCRF　　16 ビットアウトプットコンペアレジスタ F　　アドレス：H'FFBA
  機能：OCRF の設定値と TCF 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：uptbl[sluecnt]

(3) フローチャート



4.3.4 fsluedwn 関数

(1) モジュール仕様

機能概要：正転時の Slew-down 制御を行なう

表 7　モジュール仕様

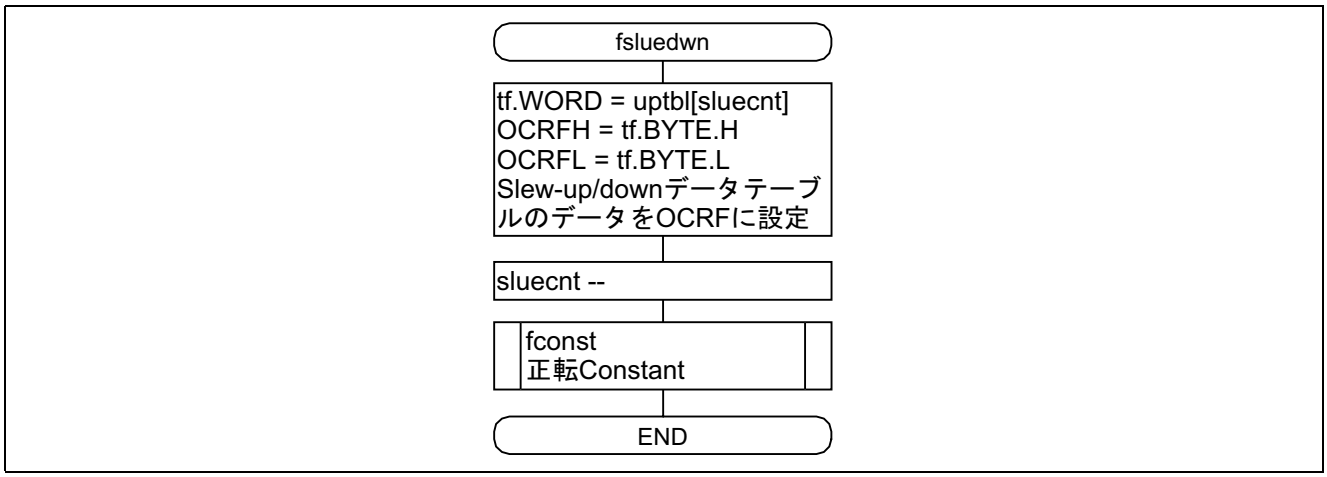
	型	変数名	内容
引数	なし	なし	なし
使用 RAM	unsigned char	sluecnt	Slew-up, Slew-down 時に使用する配列 uptbl[]の要素
使用 ROM	unsigned short	uptbl[96]	Slew-up, Slew-down 時の割込み時間データテーブル

(2) 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。

- OCRF　16 ビットアウトプットコンペアレジスタ F　アドレス：H'FFBA
  機能：OCRF の設定値と TCF 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：uptbl[sluecnt]

(3) フローチャート



4.3.5 fconst 関数

(1) モジュール仕様

機能概要：正転時の Constant 制御を行なう

表 8 モジュール仕様

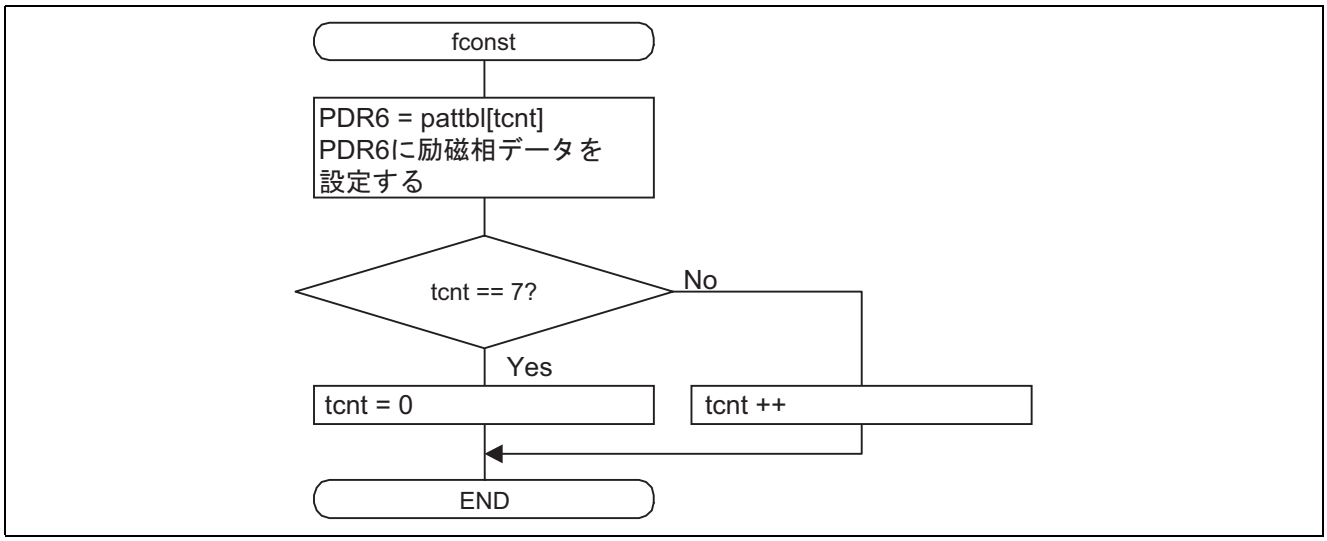
	型	変数名	内容
引数	**なし**	**なし**	**なし**
使用 RAM	unsigned char	tcnt	**ステッピングモータの励磁データである配列 pattbl[]の要素**
使用 ROM	unsigned char	pattbl[8]	**ステッピングモータの励磁パターンデータテーブル**

(2) 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。

- 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：pattbl[tcnt]

(3) フローチャート



4.3.6 frstop 関数

(1) モジュール仕様

機能概要：正転/逆転の停止を行なう

表 9　モジュール仕様

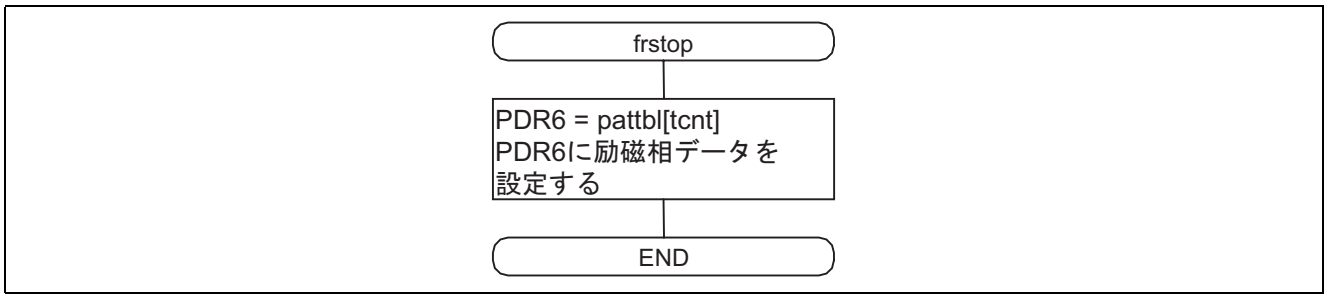
	型	変数名	内容
引数	なし	なし	なし
使用 RAM	unsigned char	tcnt	ステッピングモータの励磁データである配列 pattbl[]の要素
使用 ROM	unsigned char	pattbl[8]	ステッピングモータの励磁パターンデータテーブル

(2) 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。

- 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：pattbl[tcnt]

(3) フローチャート



4.3.7 rslueup 関数

(1) モジュール仕様

機能概要：逆転時の Slew-up 制御を行なう

表 10　モジュール仕様

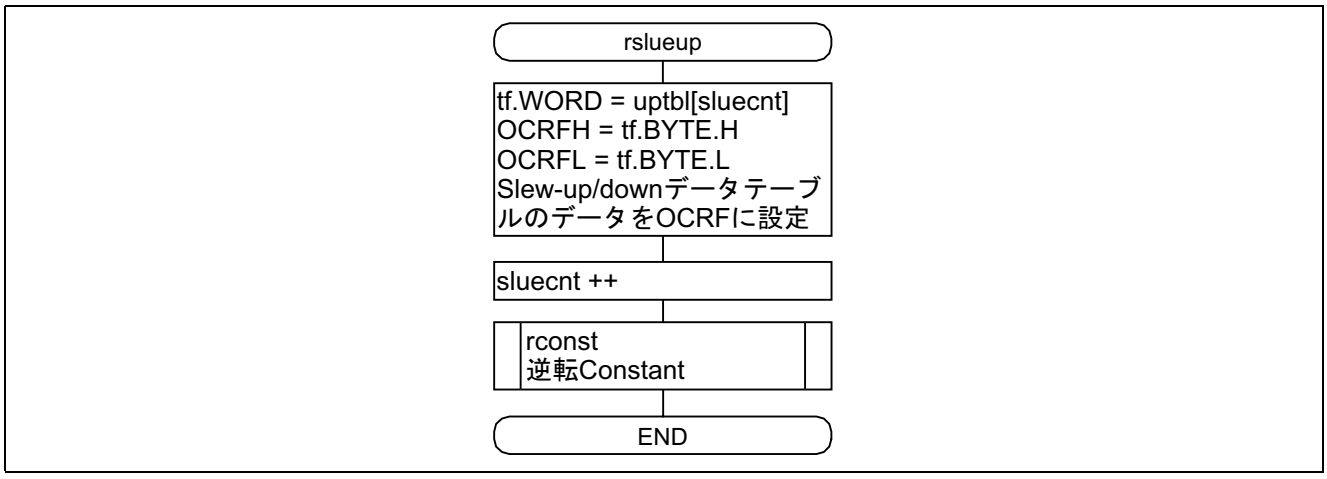
	型	変数名	内容
引数	なし	なし	なし
使用 RAM	unsigned char	sluecnt	Slew-up, Slew-down 時に使用する配列 uptbl[]の要素
使用 ROM	unsigned short	uptbl[96]	Slew-up, Slew-down 時の割込み時間データテーブル

(2) 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。

- OCRF　16 ビットアウトプットコンペアレジスタ F　アドレス：H'FFBA
  機能：OCRF の設定値と TCF 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：uptbl[sluecnt]

(3) フローチャート



4.3.8 rsluedwn 関数

(1) モジュール仕様

機能概要：逆転時の Slew-down 制御を行なう

表 11　モジュール仕様

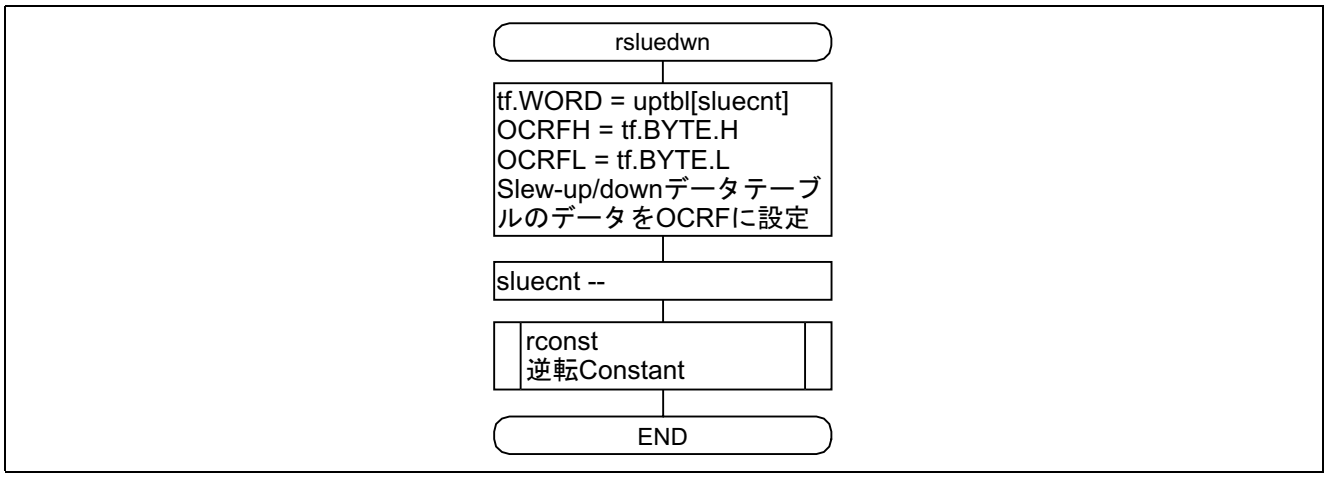
	型	変数名	内容
引数	なし	なし	なし
使用 RAM	unsigned char	sluecnt	Slew-up, Slew-down 時に使用する配列 uptbl[]の要素
使用 ROM	unsigned short	uptbl[96]	Slew-up, Slew-down 時の割込み時間データテーブル

(2) 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。

- OCRF　16 ビットアウトプットコンペアレジスタ F　アドレス：H'FFBA
  機能：OCRF の設定値と TCF 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：uptbl[sluecnt]

(3) フローチャート



4.3.9 rconst 関数

(1) モジュール仕様

機能概要：逆転時の Constant 制御を行なう

表 12　モジュール仕様

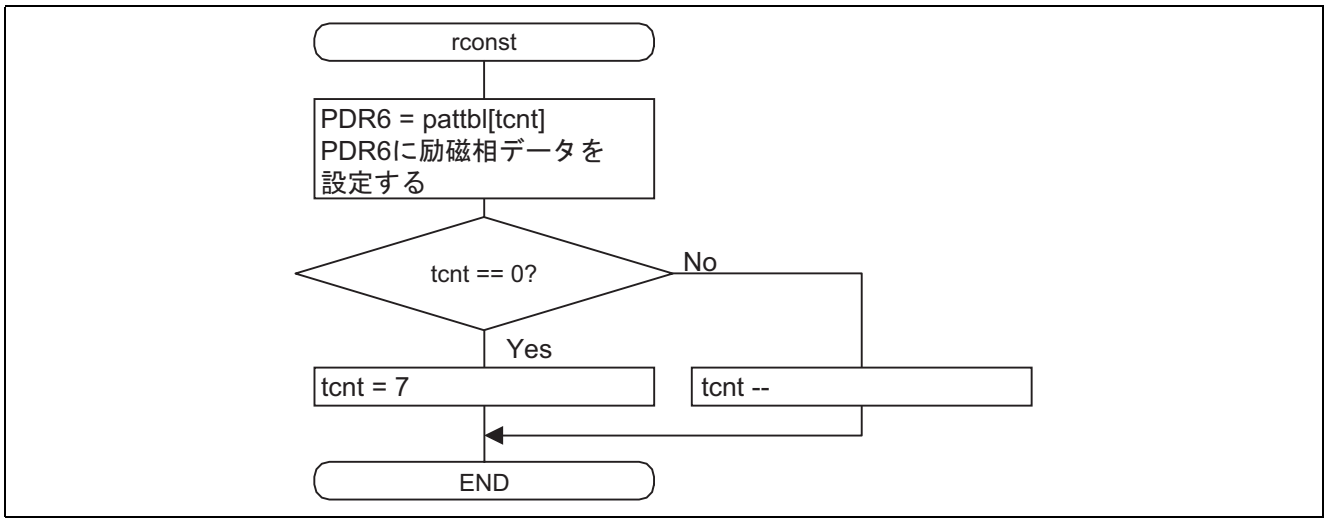
	型	変数名	内容
引数	なし	なし	なし
使用 RAM	unsigned char	tcnt	ステッピングモータの励磁データである配列 pattbl[]の要素
使用 ROM	unsigned char	pattbl[8]	ステッピングモータの励磁パターンデータテーブル

(2) 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。

- 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：pattbl[tcnt]

(3) フローチャート



4.4 リンクアドレス指定

セクション名	アドレス
CV1	H'0000
CV2	H'001E
P	H'0100
C	H'0800
DOUTDT	H'0810
B	H'FB80

5. プログラムリスト

```
/*********************************************************/
/*                                                       */
/*  H8/300L Super Low Power Series                        */
/*     -H8/38024 Series-                                 */
/*  Application Note                                     */
/*                                                       */
/*  '1-2 Phase Excitation Control for a Stepping Motor    */
/*                                                       */
/*  Function                                             */
/*  : Timer F Output Compare                             */
/*                                                       */
/*  External Clock : 10MHz                               */
/*  Internal Clock : 5MHz                                */
/*  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 TCRF       *(volatile unsigned char *)0xFFB6      /* Timer Control Register F        */
#define TCSRF      *(volatile unsigned char *)0xFFB7      /* Timer Control Status Register F  */
#define TCSRF_BIT   (*(volatile struct BIT *)0xFFB7)      /* Timer Control Status Register F  */
#define OVFH       TCSRF_BIT.b7                          /* Timer Overflow Flag H           */
#define CMFH       TCSRF_BIT.b6                          /* Compare Match Flag H            */
#define OVIEH      TCSRF_BIT.b5                          /* Timer Overflow Interrupt Enable  */
#define CCLRH      TCSRF_BIT.b4                          /* Output Select 3                 */
#define TCF        *(volatile unsigned short *)0xFFB8     /* Timer Counter F                 */
#define TCFH       *(volatile unsigned char *)0xFFB8      /* Timer Counter F                 */
#define TCFL       *(volatile unsigned char *)0xFFB9      /* Timer Counter F                 */
#define OCRF       *(volatile unsigned short *)0xFFBA     /* Output Compare Register F       */
#define OCRFH      *(volatile unsigned char *)0xFFBA      /* Output Compare Register F       */
#define OCRFL      *(volatile unsigned char *)0xFFBB      /* Output Compare Register F       */
#define PDR6       *(volatile unsigned char *)0xFFD9      /* Port Data Register 6            */
#define PCR6       *(volatile unsigned char *)0xFFE9      /* Port Control Register 6         */
#define IENR2_BIT   (*(volatile struct BIT *)0xFFF4)      /* Interrupt Enable Register 2     */
#define IENTFH     IENR2_BIT.b3                          /* Timer F Interrupt Enable        */
#define IRR2_BIT   (*(volatile struct BIT *)0xFFF7)       /* Interrupt Request Register 2    */
#define IRRTFH     IRR2_BIT.b3                           /* Timer F Interrupt Request Flag   */

#pragma interrupt   (tfhint)
/*********************************************************/
```

```
/*  Function define                                     */
/*********************************************************/
void main ( void );
void tfhint ( 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   tcnt,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] = {                          /* Slue Up/Down table            */
   0x4E20,0x4D71,0x4CC2,0x4C13,0x4B64,0x4AB5,0x4A06,0x4957,0x48A8,0x47F9,
   0x474A,0x469B,0x45EC,0x453D,0x448E,0x43DF,0x4330,0x4281,0x41D2,0x4123,
   0x4074,0x3FC5,0x3F16,0x3E67,0x3DB8,0x3D09,0x3C5A,0x3BAB,0x3AFC,0x3A4D,
   0x399E,0x38EF,0x3840,0x3791,0x36E2,0x3633,0x3584,0x34D5,0x3426,0x3377,
   0x32C8,0x3219,0x316A,0x30BB,0x300C,0x2F5D,0x2EAE,0x2DFF,0x2D50,0x2CA1,
   0x2BF2,0x2B43,0x2A94,0x29E5,0x2936,0x2887,0x27D8,0x2729,0x267A,0x25CB,
   0x251C,0x246D,0x23BE,0x230F,0x2260,0x21B1,0x2102,0x2053,0x1FA4,0x1EF5,
   0x1E46,0x1D97,0x1CE8,0x1C39,0x1B8A,0x1ADB,0x1A2C,0x197D,0x18CE,0x181F,
   0x1770,0x16C1,0x1612,0x1563,0x14B4,0x1405,0x1356,0x12A7,0x11F8,0x1149,
   0x109A,0x0FEB,0x0F3C,0x0E8D,0x0DDE,0x0D2F
};

/*********************************************************/
/*  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) = {
   tfhint                                            /* 1E Timer FH Interrupt         */
};

#pragma entry main(sp=0xFF80)
#pragma section                                        /* P                             */
/*********************************************************/
/*  Main Program                                        */
/*********************************************************/
```

```
void main ( void )
{
    unsigned char tmp;

    union {
        unsigned short  WORD;
        struct {
            unsigned char   H;
            unsigned char   L;
        }BYTE;
    }tf;

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

    nextmode = 0;
    modecnt = 95;                                   /* Motor Slue mode countset "95"    */

    TCRF = 0x06;                                    /* Initialize Clock Select         */
    tmp = TCSRF;
    TCSRF = 0x10;                                   /* Initialize Overflow Interrupt    */

    sluecnt = 0;                                    /* Slue Up/Down table counter set   */
    tf.WORD = uptbl[sluecnt];
    OCRFH = tf.BYTE.H;                               /* Set OCRF                        */
    OCRFL = tf.BYTE.L;
    sluecnt++;

    PCR6 |= 0x0F;                                   /* Port8 Output                     */
    tcnt = 0;                                      /* Output Pattern table counter set */
    PDR6 = pattbl[tcnt];                             /* PDR6 Set Output Pattern         */
    tcnt++;

    TCFH = 0x00;                                    /* Set TCF                         */
    TCFL = 0x00;
    IRRTFH = 0;                                     /* Clear IRRTFH                     */
    IENTFH = 1;                                     /* Timer FH Interrupt Enable       */
    set_imask_ccr(0);                                /* Interrupt Enable                */

    while(1);
}

/*********************************************************/
/*  Timer FH Interrupt                                    */
/*********************************************************/
void tfhint ( 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(tcnt==0)
            tcnt = 7;
        else
            tcnt--;
    }
    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(tcnt==0)
            tcnt = 7;
        else
            tcnt--;
    }
    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(tcnt==7)
                    tcnt = 0;
                else
                    tcnt++;
            }
            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(tcnt==7)
                    tcnt = 0;
                else
                    tcnt++;
            }
            break;
    }

    IRRTFH = 0;
    tmp = TCSRF;
    TCSRF = 0x10;                                       /* Interrupt Flag Clear          */
}

/*********************************************************/
/*  Forward Slue Up                                      */
/*********************************************************/
void fslueup ( void )
{
    union {
        unsigned short  WORD;
        struct {
            unsigned char   H;
            unsigned char   L;
        }BYTE;
    }tf;

    tf.WORD = uptbl[sluecnt];
    OCRFH = tf.BYTE.H;                                  /* OCRF Set Slue Up/Down table    */
    OCRFL = tf.BYTE.L;
    sluecnt++;

    fconst();
}

/*********************************************************/
/*  Forward Slue Down                                    */
/*********************************************************/
```

```
void fsluedwn ( void )
{
    union {
        unsigned short  WORD;
        struct {
            unsigned char   H;
            unsigned char   L;
        }BYTE;
    }tf;

    tf.WORD = uptbl[sluecnt];
    OCRFH = tf.BYTE.H;                              /* OCRF Set Slue Up/Down table     */
    OCRFL = tf.BYTE.L;
    sluecnt--;

    fconst();
}

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

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

/************************************************************/
/*  Reverse Slue Up                                         */
/************************************************************/
void rslueup ( void )
{
    union {
        unsigned short  WORD;
        struct {
            unsigned char   H;
            unsigned char   L;
        }BYTE;
    }tf;

    tf.WORD = uptbl[sluecnt];
    OCRFH = tf.BYTE.H;                              /* OCRF Set Slue Up/Down table     */
    OCRFL = tf.BYTE.L;
    sluecnt++;

    rconst();
```

```
}

/*********************************************************/
/*  Reverse Slue Down                                    */
/*********************************************************/
void rsluedwn ( void )
{
    union {
        unsigned short  WORD;
        struct {
            unsigned char   H;
            unsigned char   L;
        }BYTE;
    }tf;

    tf.WORD = uptbl[sluecnt];
    OCRFH = tf.BYTE.H;                              /* OCRF Set Slue Up/Down table     */
    OCRFL = tf.BYTE.L;
    sluecnt--;

    rconst();
}

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2004.07.28	—	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。