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。

SH7145 グループ

CMT によるステッピングモータの加減速制御

要旨

CMT(Compare Match Timer)モジュールを用いた、1-2 相励磁方式によるステッピングモータの加減速制御について述べます。正転→停止→逆転→停止を繰り返します。

動作確認デバイス

SH7145F

目次

1. 仕様

SH7145 の CMT(Compare Match Timer)のチャネル 0 と汎用ポート（PC12～PC15）を用いてパルスを発生させ、2 相ステッピングモータの加減速制御を行います。本タスク例では、1-2 相励磁方式により制御し、正転→静止→逆転→静止を繰り返します。また、回転時には Slew-up および Slew-down 処理を行っています。

図 1 に SH7145F とステッピングモータの接続図を示します。

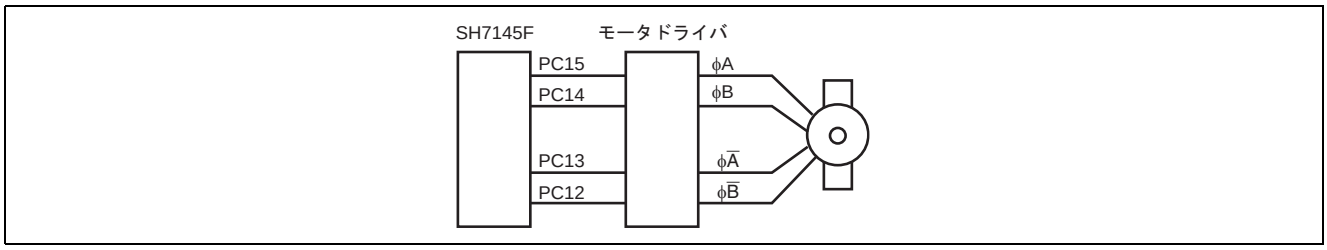


図 1　ステッピングモータとの接続図

2. 使用機能説明

本タスク例では、CMT(Compare Match Timer)と汎用ポート（ポート C）を用いて、ステッピングモータ制御用のパルスを発生させます。

2.1 ステッピングモータ

本タスク例では、パーマネン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　ステッピングモータの標準仕様

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/φ]	0.33
抵抗[Ω/φ]	36
インダクタンス[mH/φ]	28
最大静止トルク[mN･m(kgf･cm)]	78.4(0.8)
ディテントトルク[mN･m(gf･cm)]	1.3(180)
ロータイナーシャ[g･cm^2]	23.7

2.2 CMT(Compare Match Timer)

設定した周期ごとに割り込みを発生させます。図 2 に CMT モジュールチャネル 0（ch0）のブロック図を示し、以下に図 2 の機能説明をします。

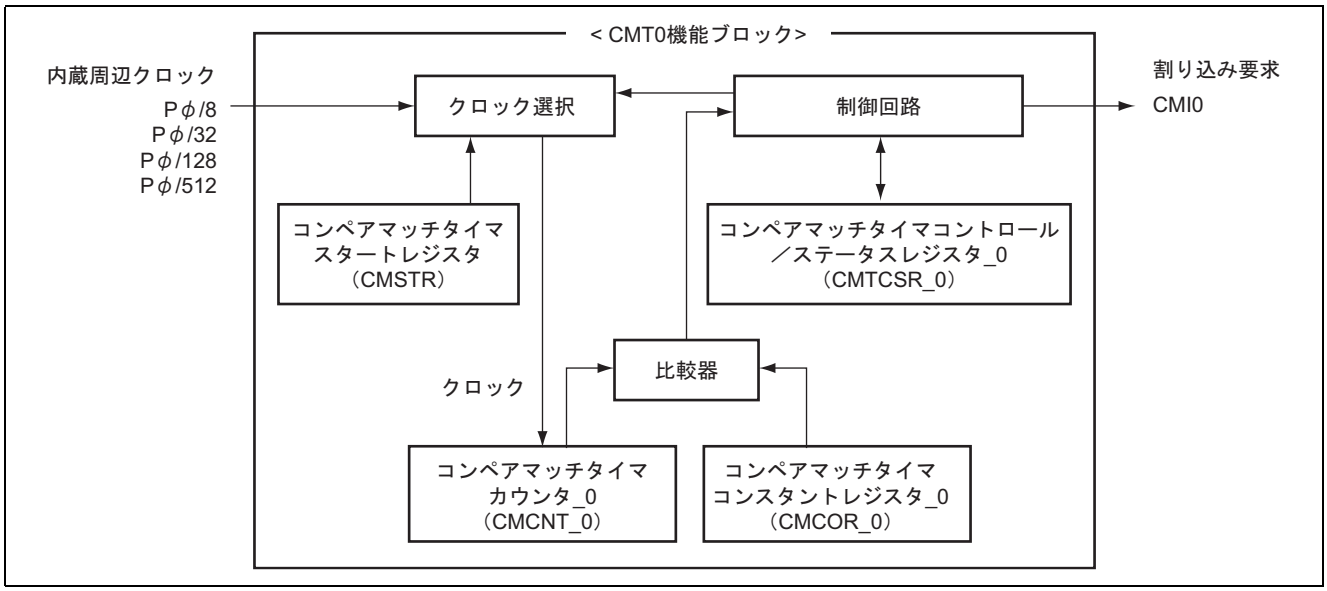


図 2　CMT(ch0)のブロック図

- CMT は、16 ビットのカウンタを持ち、設定した周期ごとの割り込みの発生が可能です。
- 内蔵周辺クロック Pφを分周したクロックを選択することができます。そして、選択したクロックにより、カウントアップを行います。
- コンペアマッチタイマスタートレジスタ(CMSTR)は、カウントの開始および停止の設定を行います。
- コンペアマッチタイマコントロール/ステータスレジスタ(CMCSR_0)は、コンペアマッチ発生の表示、割り込みの設定、カウントアップ用クロックの選択を行います。
- コンペアマッチタイマカウンタ(CMCNT_0)は、割り込み要求を発生させるためのアップカウンタです。
- コンペアマッチタイマコンスタントレジスタ(CMCOR_0)は、コンペアマッチ周期を設定します。

2.3 汎用ポート（ポート C）

本タスク例では、汎用ポート（ポート C）の制御により、1-2 相励磁用のパルスを出力します。パルス出力には、PC12～PC15 を使用します。図 3 にポート C のブロック図を示し、以下に図 3 の機能説明をします。

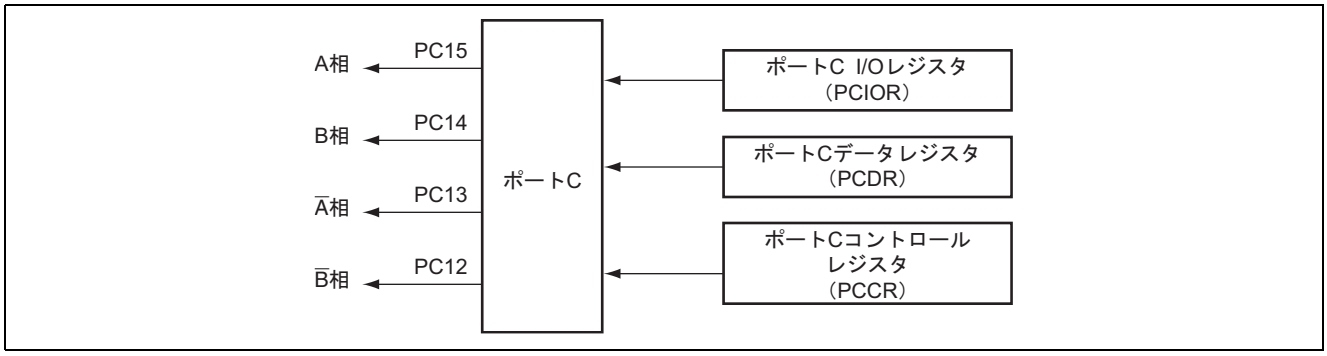


図 3　ポート C のブロック図

- ポート C は、16 ビットの汎用入出力ポートです。
- ポート C コントロールレジスタ(PCCR)は、マルチプレクス端子の機能を選択するレジスタです。
- ポート C･I/O･レジスタ(PCIOR)は、端子の入出力方向を選択するレジスタです。PCIOR は、ポート C の端子機能が汎用入出力の場合に有効で、それ以外の場合は無効です。
- ポート C データレジスタ(PCDR)は、ポート C のデータを格納するレジスタです。汎用出力の場合、PCDR に書き込んだ値が対応する端子から直接出力されます。また、汎用入力の場合に PCDR を読み出すと、端子の状態が直接読み出されます。

3. 動作説明

3.1 1-2 相励磁用パルスの出力

パルスの出力は、CMT の割り込み機能とポート C の汎用出力により行います。ポート C の端子を 4 つ(PC12～PC15)使用してパルスを出力します。CMT によるコンペアマッチ割り込みが入るたびに、ポート C からの出力を変化させます。図 4-1 に機能ブロック図を、図 4-2 に 1-2 相励磁用パルス（ポート C の出力変化）を示します。

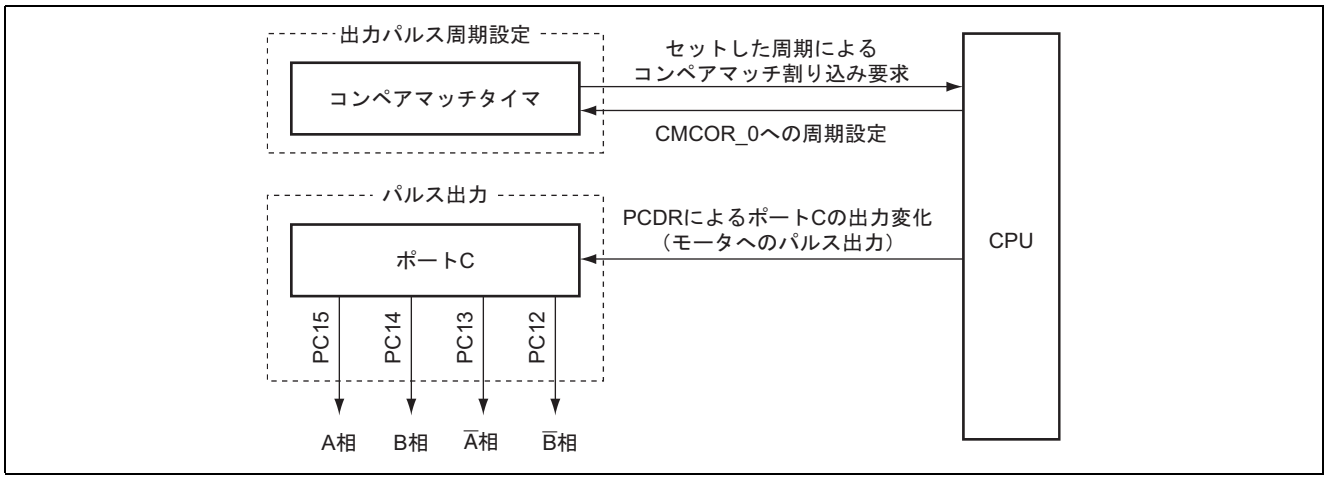


図 4-1 機能ブロック図

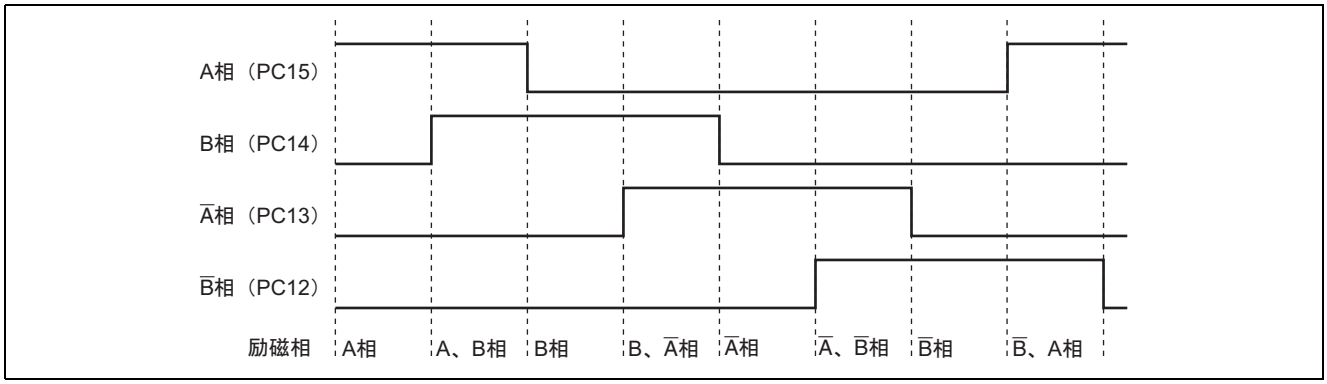


図 4-2 1-2 相励磁用パルス

3.2 ステッピングモータの動作例

ステッピングモータの 1-2 相励磁での動作例を図 5 に示し、以下に動作概要を説明します。

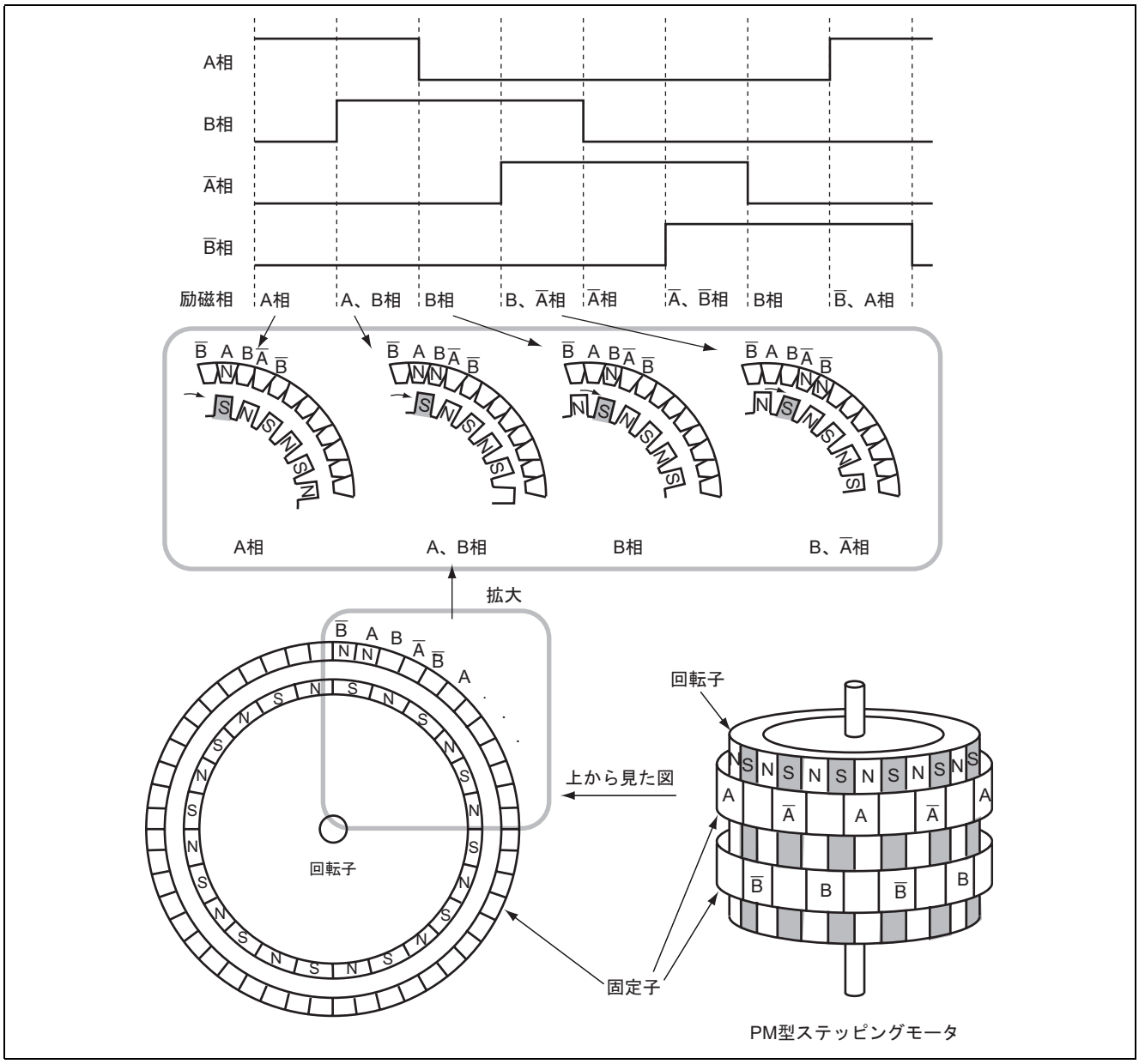


図 5　1-2 相励磁でのステッピングモータの動作例

(1)まず、A 相を励磁します。このときロータ（回転子）は、A 相に位置します。
(2)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,Ā 相→Ā 相→Ā,B̄ 相…の順番に励磁し、ロータを回転させます。
(3)逆転動作の場合は、B,A 相→A 相→A,B̄ 相→B̄ 相→Ā,B̄ 相…の順番に励磁することでロータを回転させます。
(4)停止動作は、最後に励磁した相を一定時間励磁し続けることで、ロータの回転を止めます。

3.3 モータの加減速制御

Slew-up, Slew-down 動作により、モータの加減速制御を行います。これにより、回転・停止時のモータの脱調を防止します。図 6 に動作例を示し、以下に動作概要を説明します。

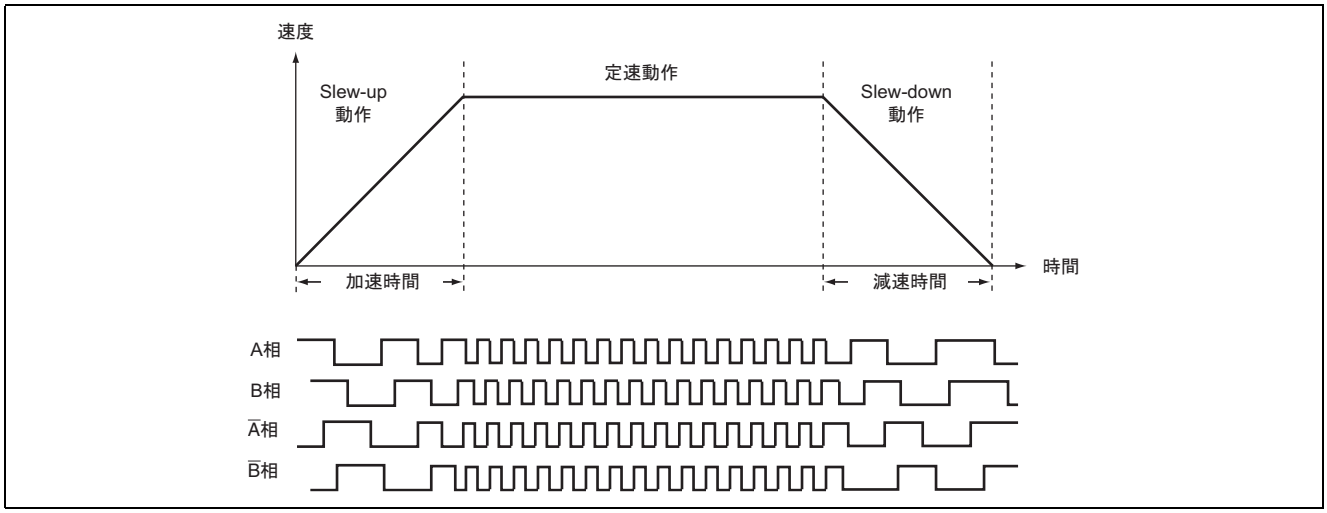


図 6　Slew-up, Slew-down 動作例

(1)徐々にパルスの周期を短くすることで、モータの回転速度を加速させます。（Slew-up）
(2)一定の周期でパルスを発生させることで、モータの回転速度を一定に保ちます。
(3)徐々にパルスの周期を長くすることで、モータの回転速度を減速させます。（Slew-down）

図 7 にステッピングモータにおける加減速制御のフローチャートを示します。

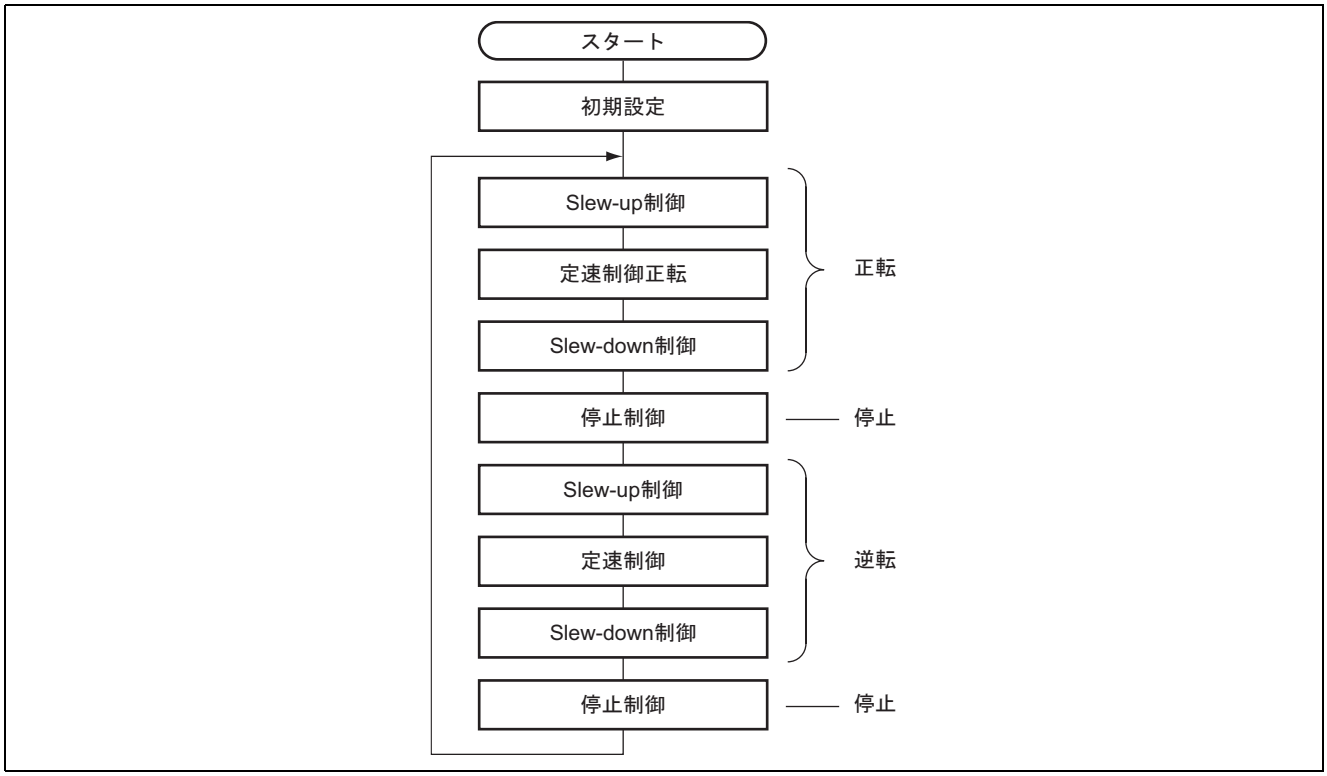


図 7　加減速制御のフローチャート

3.4 Slew-up 制御

Slew-up 時の動作原理を図 8 に示します。また、図 8 の説明として表 2 にソフトウェアおよびハードウェア処理の内容を示します。図 8 は正転時の処理であり、逆転時にはポートからの H 出力の順番が逆になります。

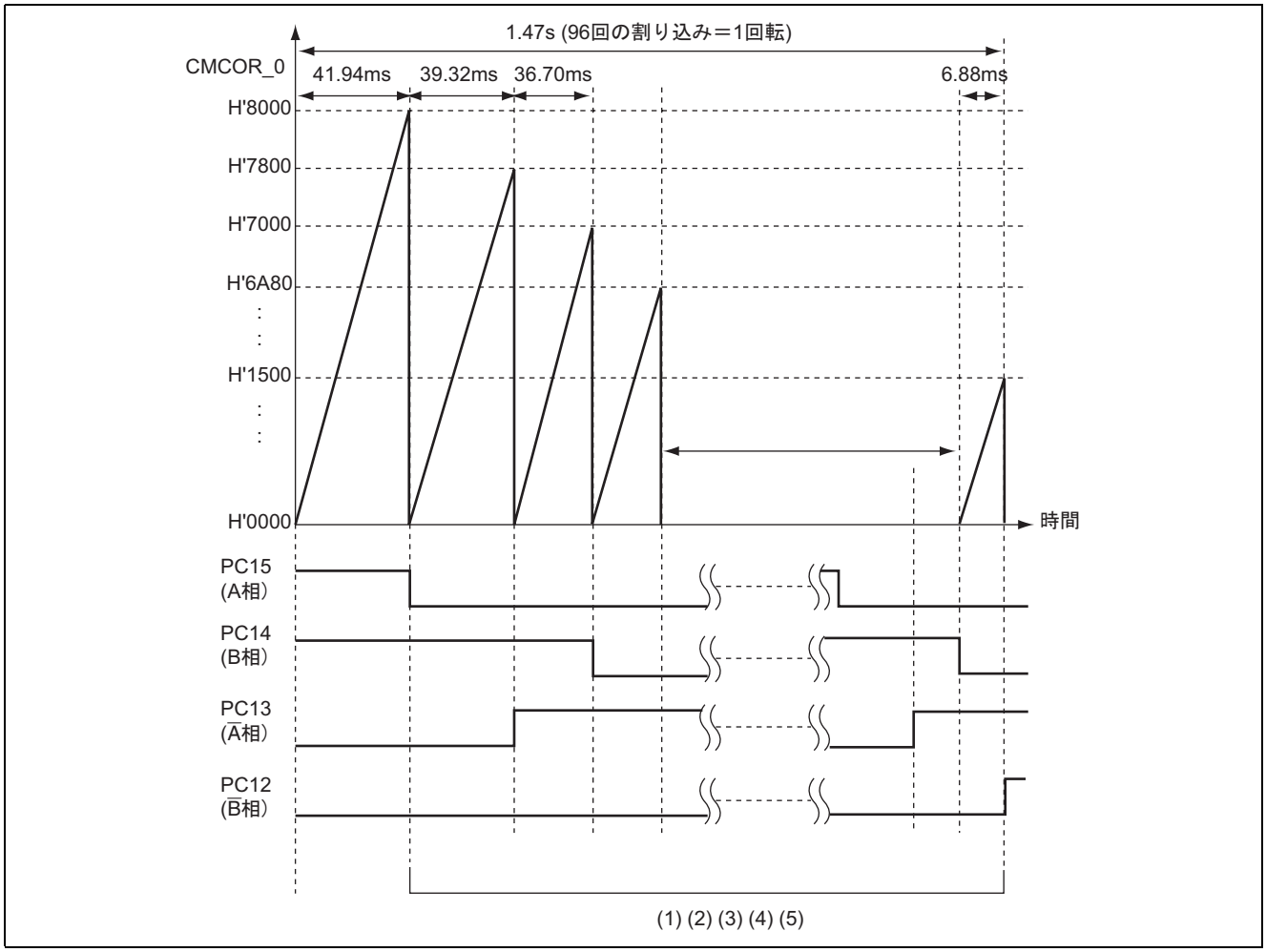


図 8　Slew-up 動作原理

表 2　処理内容

	ソフトウェア処理	ハードウェア処理
(1)	—	CMF フラグセット（コンペアマッチ割り込み発生）
(2)	CMCOR_0 に次の励磁切り替え周期をセット	—
(3)	CMF フラグクリア	CMT_0 のカウントスタート
(4)	PCDR をセットし励磁する相を切り替える	—
(5)	ロータが一回転するまで(1)～(4)を繰り返し	ロータが一回転するまで(1)～(4)を繰り返し

3.5 Slew-down 制御

Slew-down 制御の動作原理を図 9 に示します。また、図 9 の説明として表 3 にソフトウェアおよびハードウェア処理の内容を示します。図 9 は正転時の処理であり、逆転時にはポートからの H 出力の順番が逆になります。

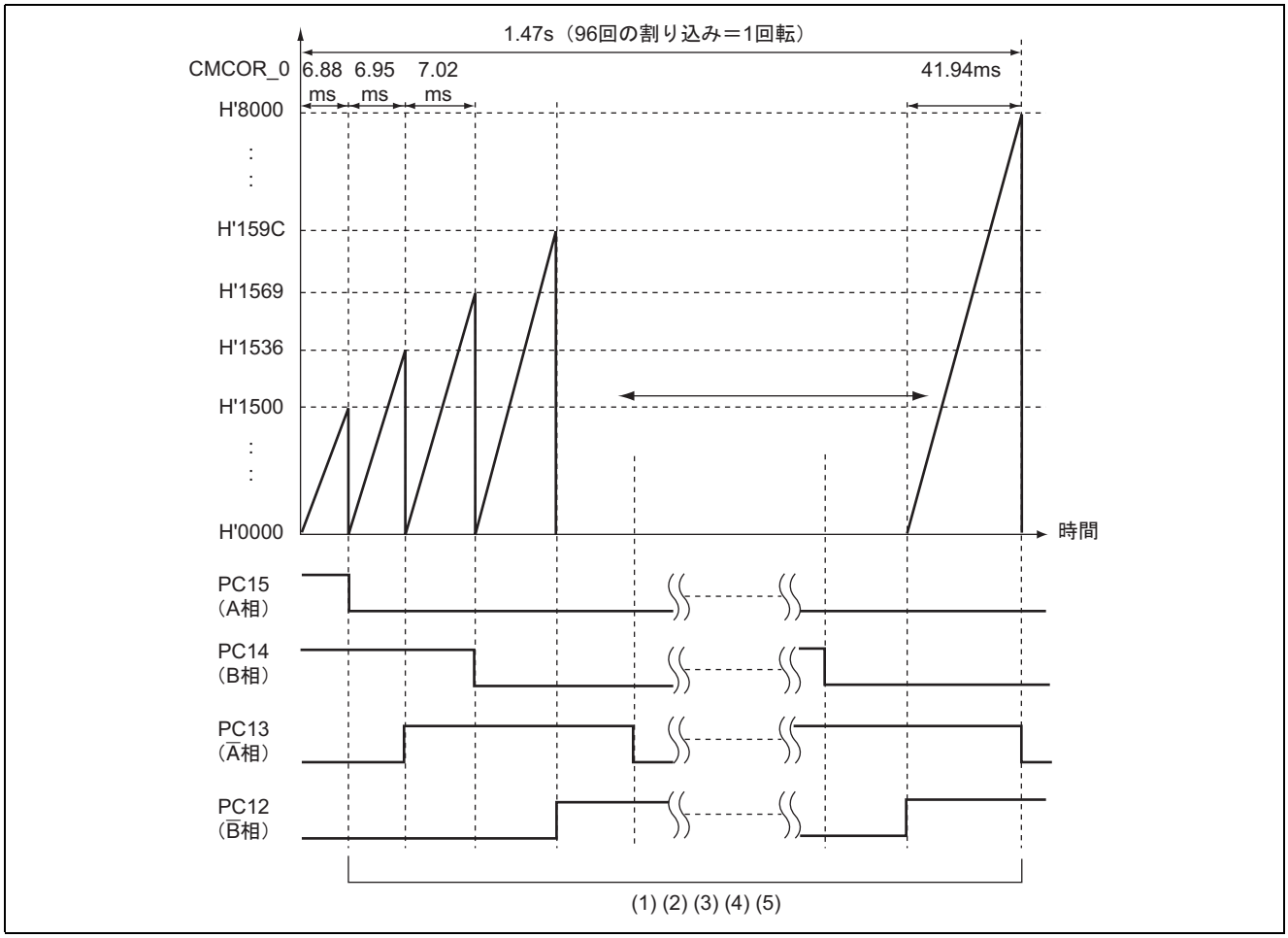


図 9　Slew-down 動作原理

表 3　処理内容

	ソフトウェア処理	ハードウェア処理
(1)	—	CMF フラグセット（コンペアマッチ割り込み発生）
(2)	CMCOR_0 に次の励磁切り替え周期をセット	—
(3)	CMF フラグクリア	CMT_0 のカウントスタート
(4)	PCDR をセットし励磁する相を切り替える	—
(5)	ロータが一回転するまで(1)～(4)を繰り返し	ロータが一回転するまで(1)～(4)を繰り返し

3.6 定速制御

定速制御の動作原理を図 10 に示します。また、図 10 の説明として表 4 にソフトウェアおよびハードウェアの処理内容を示します。図 10 は正転時の処理であり、逆転時にはポートからの H 出力の順番が逆になります。

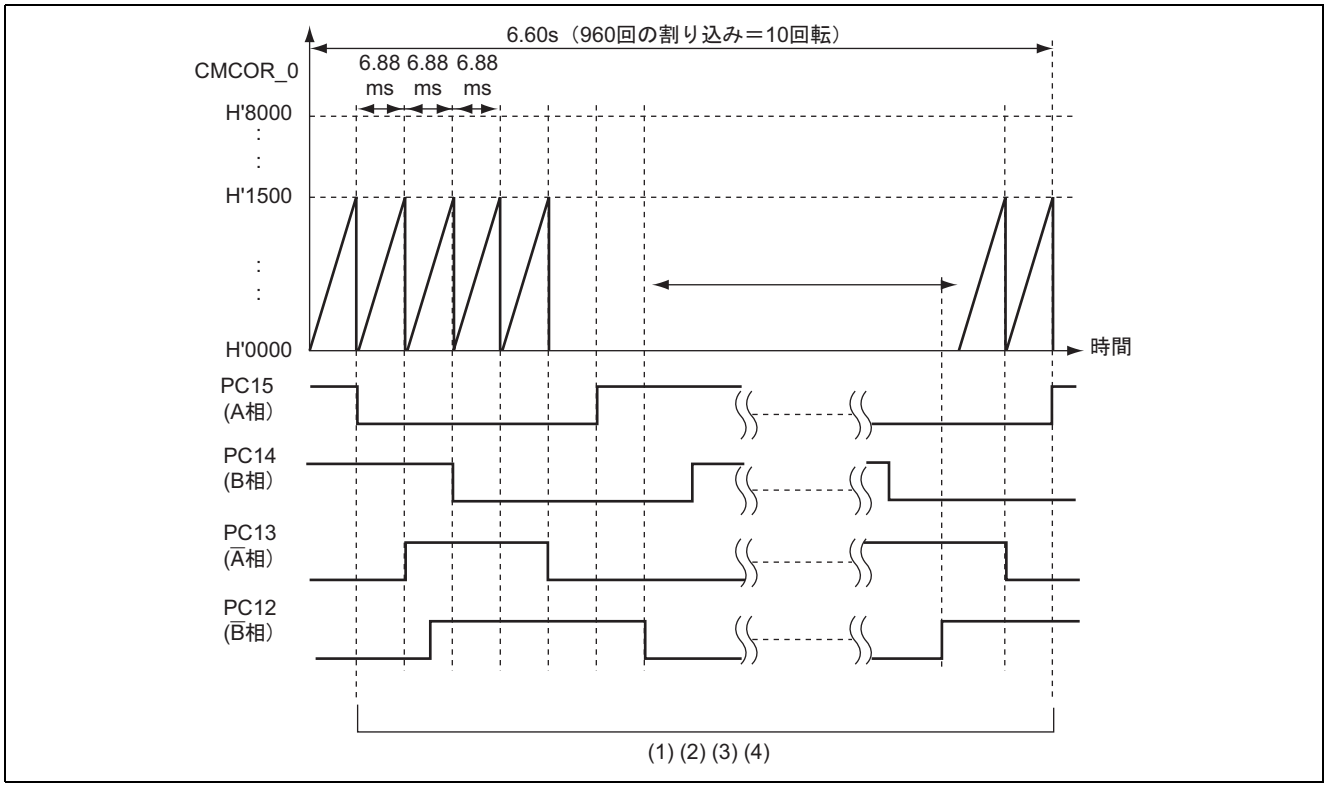


図 10　定速制御の動作原理

表 4　処理内容

	ソフトウェア処理	ハードウェア処理
(1)	—	CMF フラグセット（コンペアマッチ割り込み発生）
(2)	CMF フラグクリア	CMT_0 のカウントスタート
(3)	PCDR をセットし励磁する相を切り替える	—
(4)	(1)～(3)を指定回数繰り返し	(1)～(3)を指定回数繰り返し

4. ソフトウェア説明

4.1 モジュール説明

表 5 に本タスク例のモジュールを示します。

表 5 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、コンペアマッチタイマの初期設定、および割り込みの許可を行う
コンペアマッチ割り込み	cmt_int	ステッピングモータのメインルーチン
正転 Slew-up 制御	fslueup	正転時の Slew-up 制御を行う
正転 Slew-down 制御	fsluedwn	正転時の Slew-down 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 Slew-up 制御	rslueup	逆転時の Slew-up 制御を行う
逆転 Slew-down 制御	rsluedwn	逆転時の Slew-down 制御を行う
逆転 Constant 制御	rconst	逆転時の Constant 制御を行う

4.2 内部レジスタ説明

表 6 に本タスク例で使用する内部レジスタを示します。なお、設定値は本タスク例において使用している値であり、初期値とは異なります。

表 6 使用内部レジスタ説明（1）

レジスタ名			設定値	機能
	ビット	ビット名		
MSTCR2				モジュールスタンバイコントロールレジスタ 2
	12	MSTP12	0	CMT のスタンバイ制御ビット MSTP12=0 のとき CMT のスタンバイ解除
CMSTR			H’01	コンペアマッチタイマスタートレジスタ
	15-2	—	0	リザーブビット
	1	STR1	0	カウントスタート 1 STR1=0 で CMCNT_1 はカウント動作停止
	0	STR0	1	カウントスタート 0 STR0=1 で CMCNT_0 はカウント動作
CMCSR_0				コンペアマッチタイマコントロール/ステータスレジスタ_0
	15-8	—	0	リザーブビット
	7	CMF	※	コンペアマッチフラグ CMF=1 のとき CMCNT と CMCOR の値が一致
	6	CMIE	1	コンペアマッチ割り込みイネーブル コンペアマッチ割り込みの許可と禁止の選択 CMIE=1 のときコンペアマッチ割り込み許可
	5-2	—	0	リザーブビット
	1-0	CKS1 CKS0	0 1	CMCNT_0 に入力するクロックの選択 本タスク例では Pφ/32 を選択

【注】※ 0 クリアのみ可能で、1 のセットはハードウェアにより自動的に行われます。
※1 Slew-up および Slew-down 制御時には変化します。

表 6　使用内部レジスタ説明（2）

レジスタ名	ビット	ビット名	設定値	機能
CMCNT_0			—	コンペアマッチタイマカウンタ_0 割り込み要求を発生させるためのアップカウンタ
CMCOR_0			※1	コンペアマッチタイマコンスタントレジスタ CMCNT_0 とのコンペアマッチ周期
IPRG			H'00F0	インタラプトプライオリティレジスタ G 割り込み要因の優先順位の設定
	7-4	IPR7 IPR6 IPR5 IPR4	1 1 1 1	CMT_0 の優先順位（0-15）の設定
PCCR			H'00	ポート C コントロールレジスタ ポート C の端子機能の設定
	15	PC15MD	0	PC15MD=0 のとき対応する端子は汎用ポート機能
	14	PC14MD	0	PC14MD=0 のとき対応する端子は汎用ポート機能
	13	PC13MD	0	PC13MD=0 のとき対応する端子は汎用ポート機能
	12	PC12MD	0	PC12MD=0 のとき対応する端子は汎用ポート機能
PCIOR			H'03	ポート C･I/O･レジスタ ポート C の入出力の設定
	15	PC15IOR	1	PC15IOR=1 のとき PC15 は出力
	14	PC14IOR	1	PC14IOR=1 のとき PC14 は出力
	13	PC13IOR	1	PC13IOR=1 のとき PC13 は出力
	12	PC12IOR	1	PC12IOR=1 のとき PC12 は出力
PCDR				ポート C データレジスタ
	15	PC15DR	※2	PC15 が汎用出力のとき PC15DR の値が出力される
	14	PC14DR	※2	PC14 が汎用出力のとき PC14DR の値が出力される
	13	PC13DR	※2	PC13 が汎用出力のとき PC13DR の値が出力される
	12	PC12DR	※2	PC12 が汎用出力のとき PC12DR の値が出力される

【注】※2　コンペアマッチ割り込みが入るたびに変化します。

4.3 使用 RAM 説明

表 7 に本タスク例の使用 RAM を示します。

表 7　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
ppcnt	ステッピングモータの励磁データである配列 pattb[]の要素	1 バイト	メインルーチン コンペアマッチ割り込み処理 正転 Slew-up 制御 正転 Slew-down 制御 正転 Constant 制御 回転停止 逆転 Slew-up 制御 逆転 Slew-down 制御 逆転 Constant 制御
sluecnt	Slew-up, Slew-down 時に使用する配列 int_cyc[]の要素	1 バイト	メインルーチン コンペアマッチ割り込み処理 正転 Slew-up 制御 正転 Slew-down 制御 逆転 Slew-up 制御 逆転 Slew-down 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 コンペアマッチ割り込み処理
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 コンペアマッチ割り込み処理
pattb[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 Slew-up 制御 正転 Slew-down 制御 正転 Constant 制御 回転停止 逆転 Slew-up 制御 逆転 Slew-down 制御 逆転 Slew-down 制御
int_cyc[96]	Slew-up, Slew-down 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 Slew-up 制御 正転 Slew-down 制御 逆転 Slew-up 制御 逆転 Slew-down 制御

4.4 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

```
pattb[8]={
    0x8000,     …A 相（PC15）を励磁する。
    0xC000,     …A 相（PC15）,B 相（PC14）を励磁する。
    0x4000,     …B 相（PC14）を励磁する。
    0x6000,     …B 相（PC14）,Ā 相（PC13）を励磁する。
    0x2000,     …Ā 相（PC13）を励磁する。
    0x3000,     …Ā 相（PC13）,B̄ 相（PC12）を励磁する。
    0x1000,     …B̄ 相（PC12）を励磁する。
    0x9000,     …B̄ 相（PC12）,A 相（PC15）を励磁する。
}
```

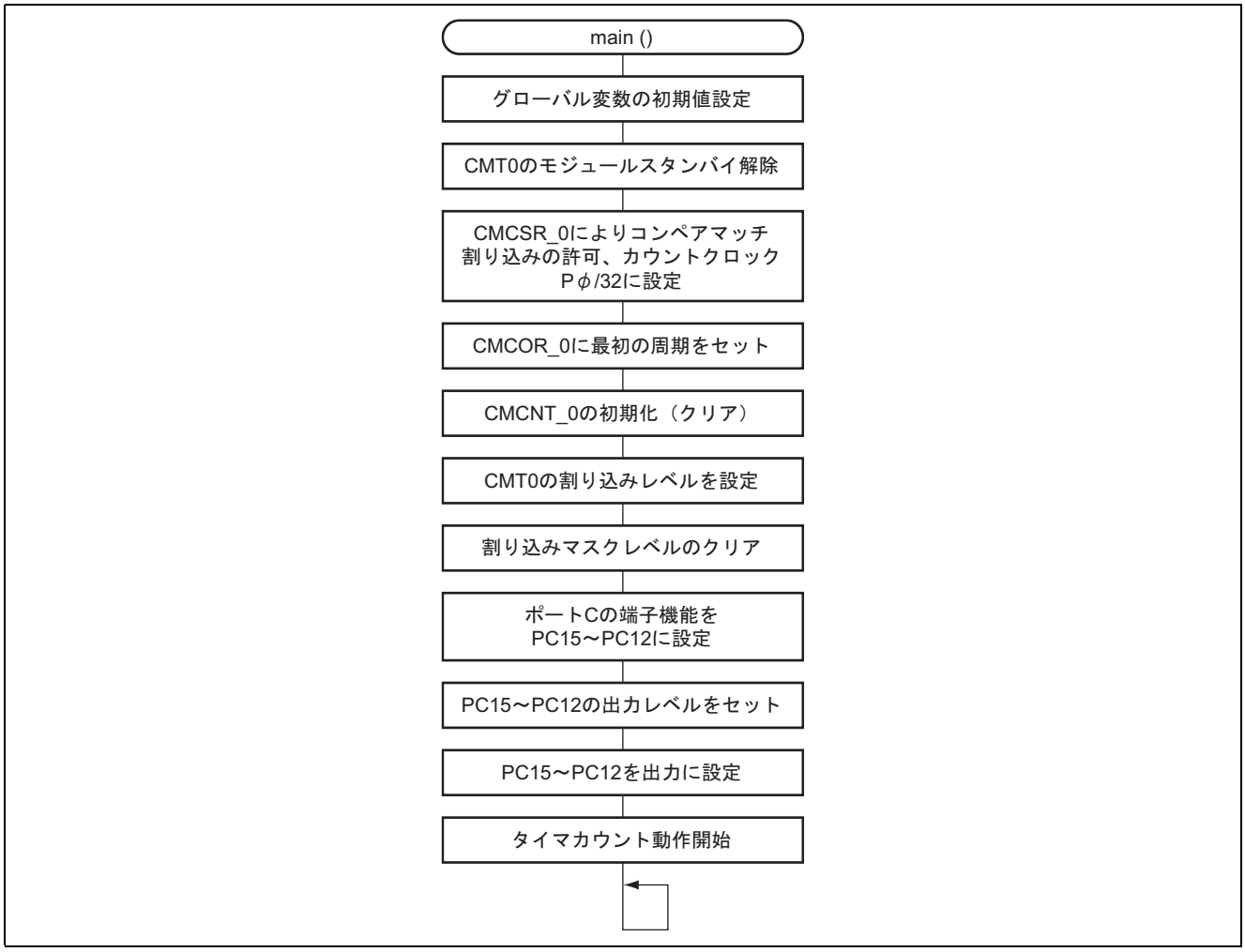
- Slew-up, Slew-down 周期設定データテーブル

```
int_cyc[96] = {
    0x8000,0x7800,0x7000,0x6A80,0x6720,0x639C,0x6018,0x5C94,0x5910,0x558C,
    0x5208,0x4EE8,0x4BC8,0x48A8,0x4650,0x43F8,0x41A0,0x4010,0x3E80,0x3D54,
    0x3C28,0x3AFC,0x3A5E,0x38A4,0x37DC,0x3714,0x364C,0x3584,0x34BC,0x33F4,
    0x335E,0x32C8,0x3232,0x319C,0x3106,0x30A2,0x303E,0x2FDA,0x2F76,0x2EE5,
    0x2E54,0x2DC3,0x2D32,0x2CA1,0x2C10,0x2B7F,0x2AEE,0x2A5D,0x29CC,0x293B,
    0x28AA,0x2819,0x2788,0x26F7,0x2666,0x25D5,0x2544,0x24B3,0x2422,0x2391,
    0x2300,0x226F,0x21DE,0x214D,0x20BC,0x202B,0x1F9A,0x1F09,0x1E78,0x1DE2,
    0x1D1A,0x1C48,0x1BCE,0x1AF4,0x1A4A,0x19EB,0x198C,0x192D,0x18CE,0x186F,
    0x1823,0x17F2,0x17B8,0x177B,0x173E,0x1701,0x16CE,0x169B,0x1668,0x1635,
    0x1602,0x15CF,0x159C,0x1569,0x1536,0x1500,
};
```

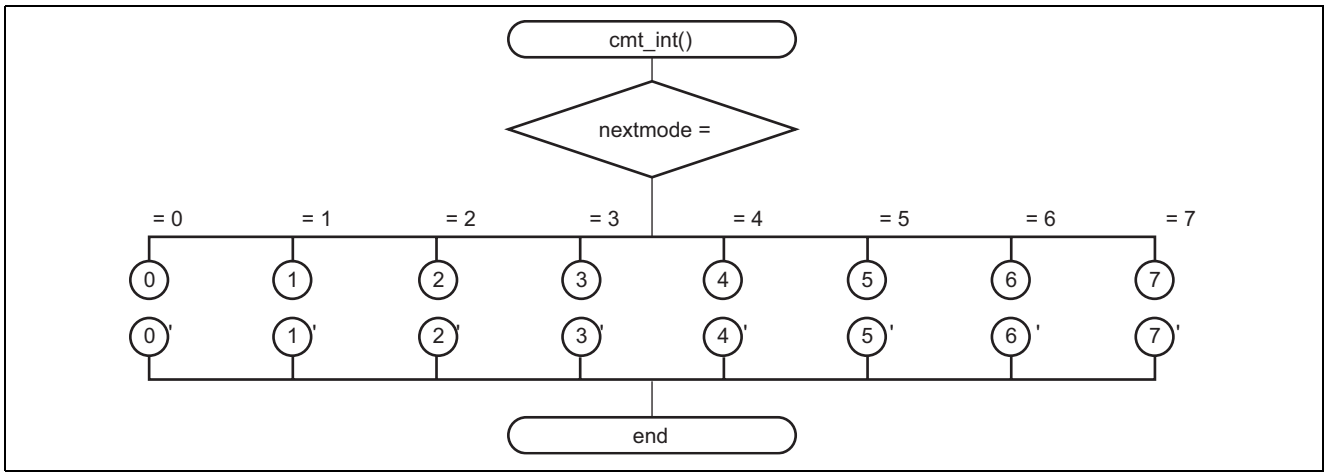
Slew-up, Slew-down 動作時、コンペアマッチ割り込みが入るたびに CMCOR_0 に書き込み、ステッピングモータの加減速制御を行う。

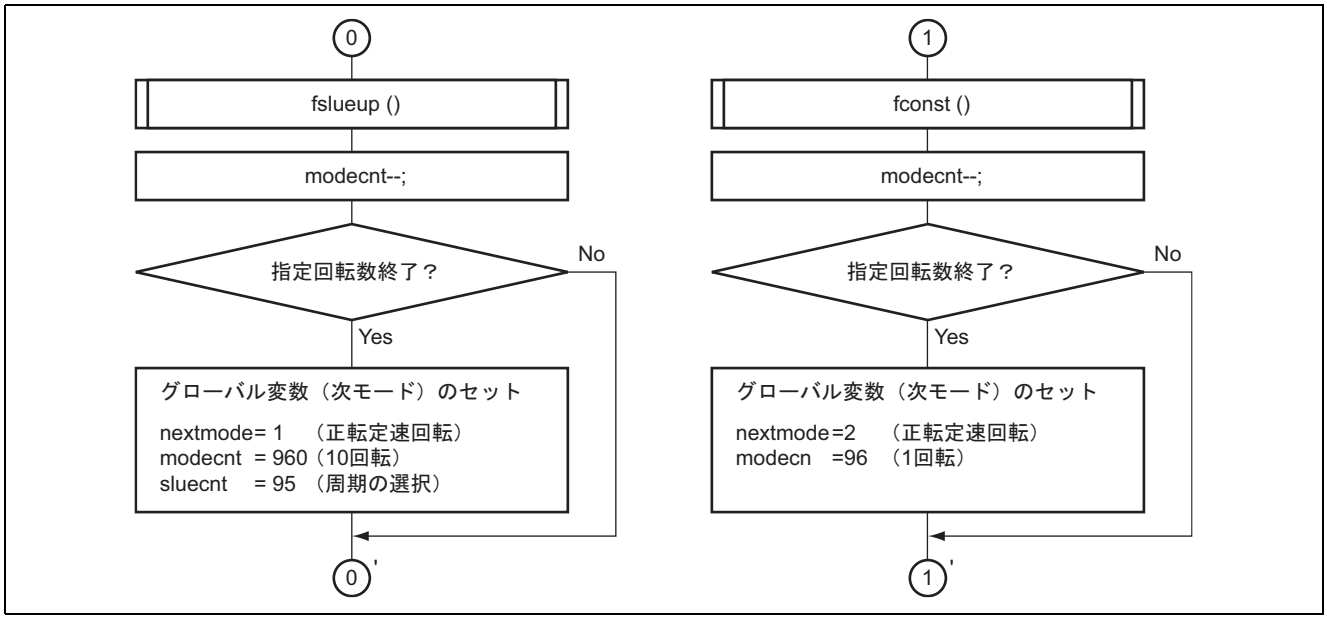
5. フローチャート

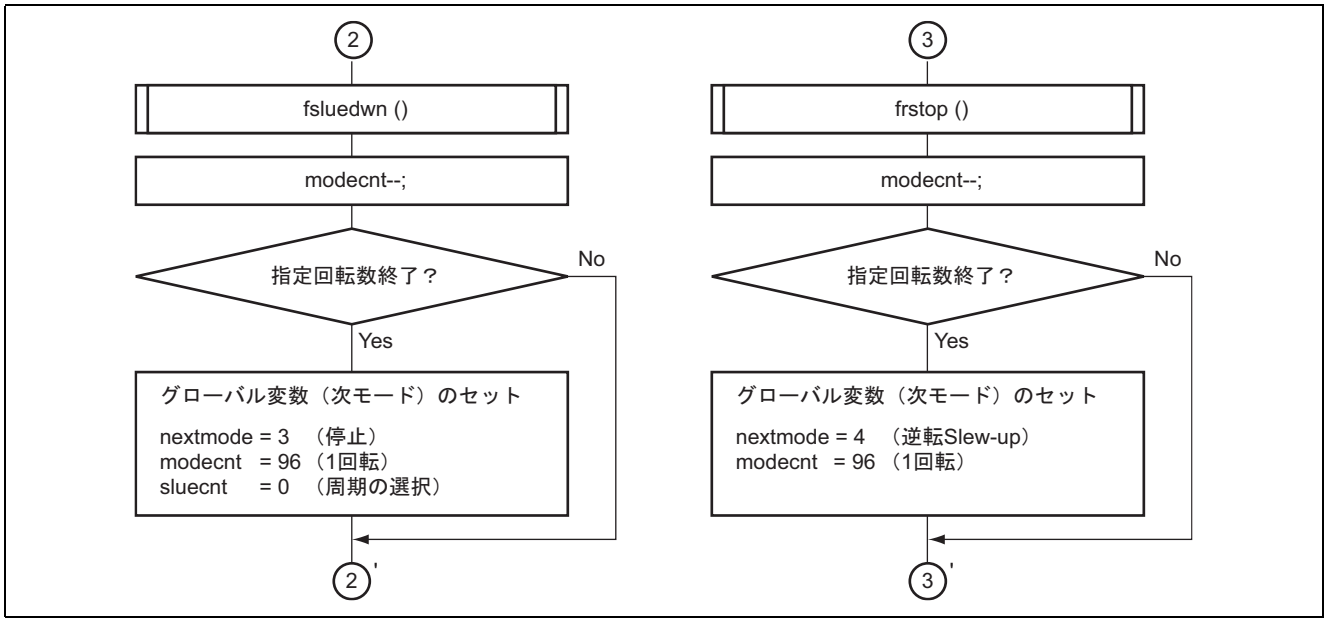
5.1 メインルーチン

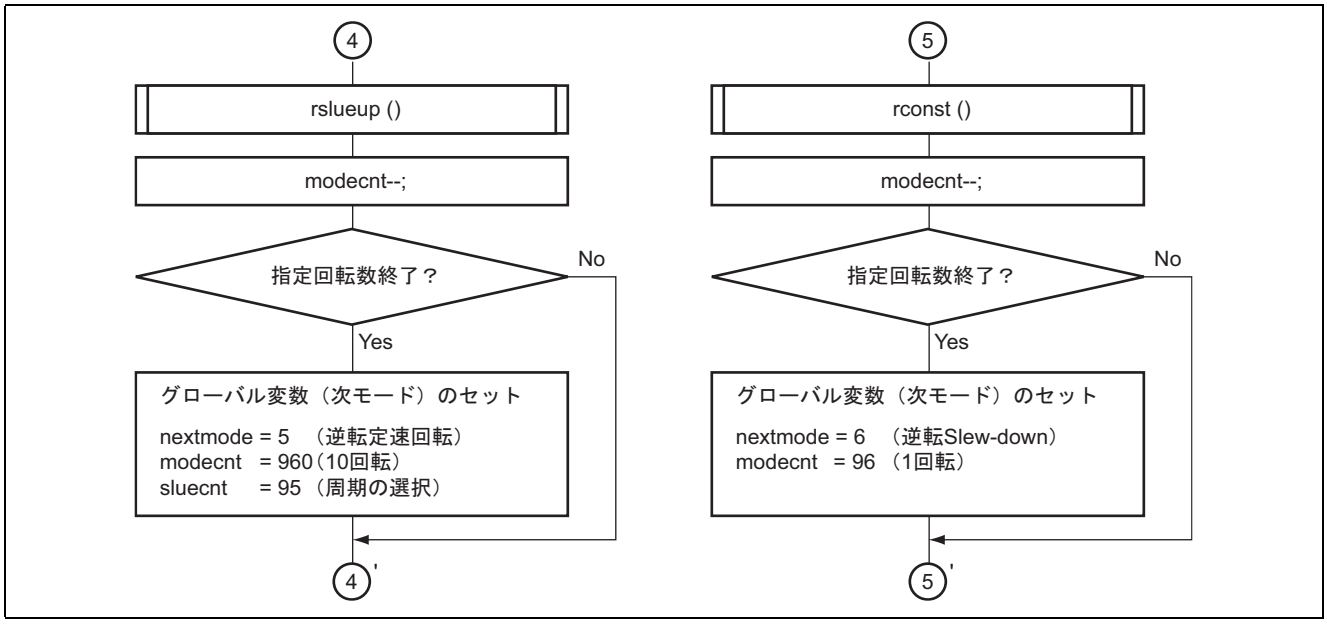


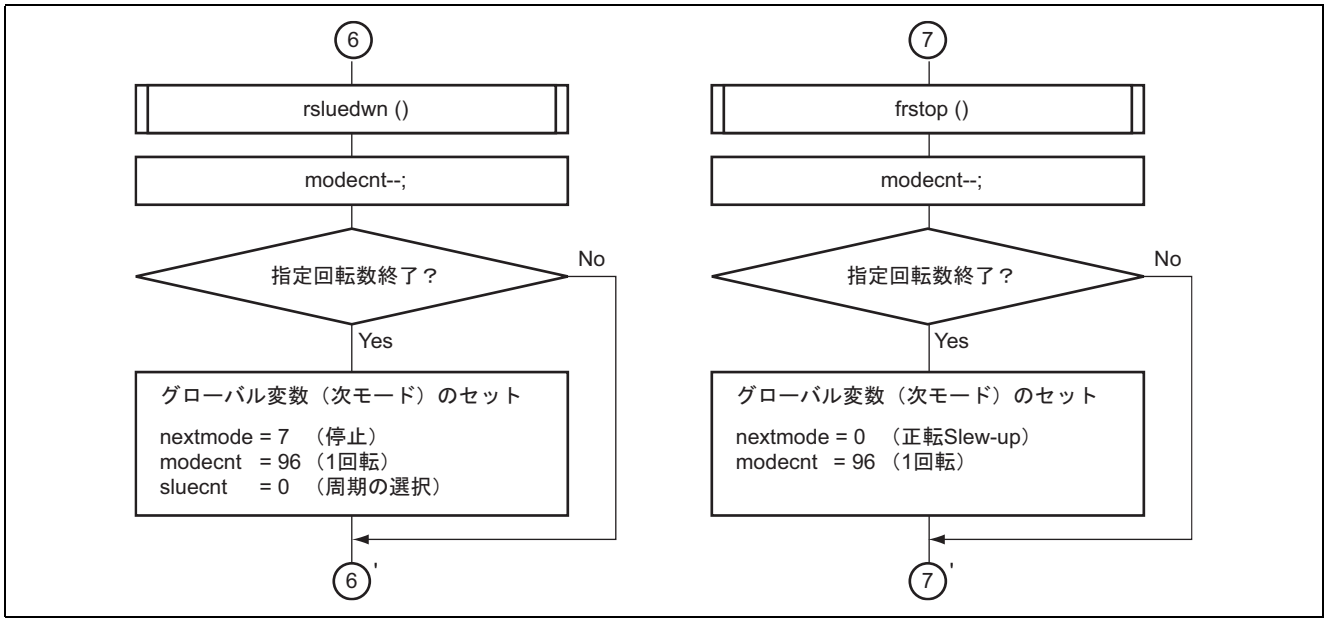
5.2 コンペアマッチ割り込みルーチン



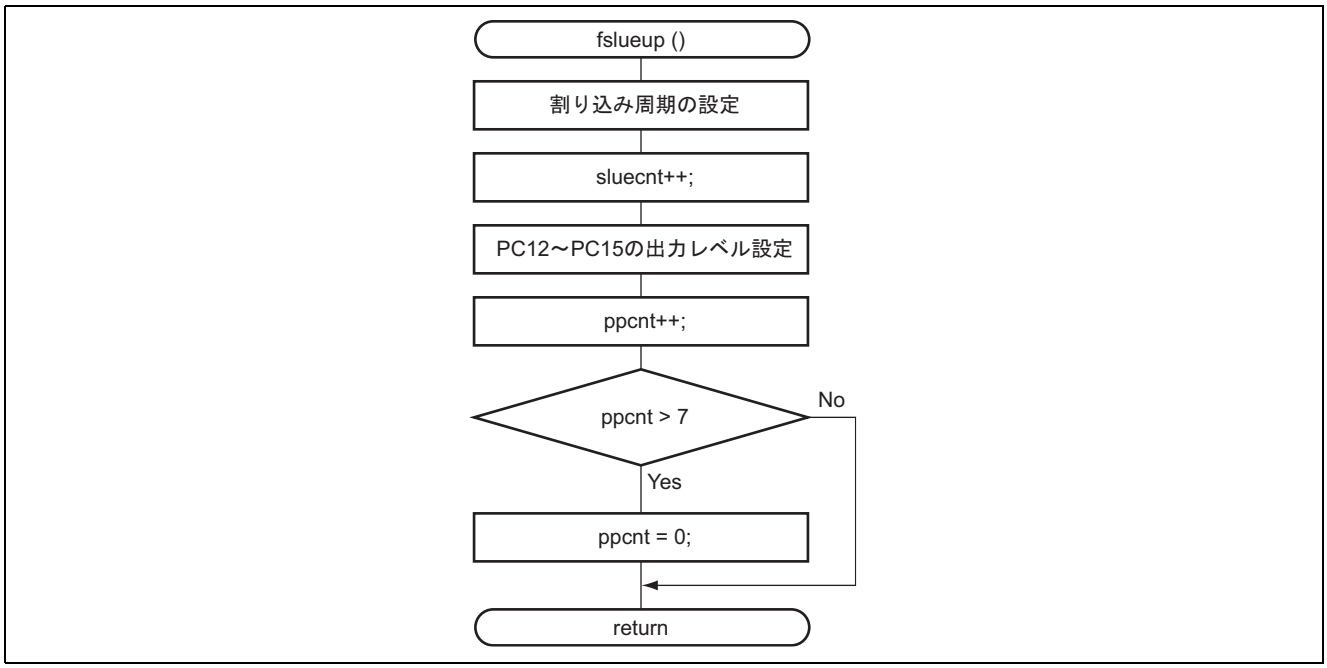



2
fsluedwn ()
modecnt--;
指定回転数終了？
No
Yes
グローバル変数（次モード）のセット
nextmode = 3 （停止）
modecnt = 96 （1回転）
sluecnt = 0 （周期の選択）
2'
3
frstop ()
modecnt--;
指定回転数終了？
No
Yes
グローバル変数（次モード）のセット
nextmode = 4 （逆転Slew-up）
modecnt = 96 （1回転）
3'

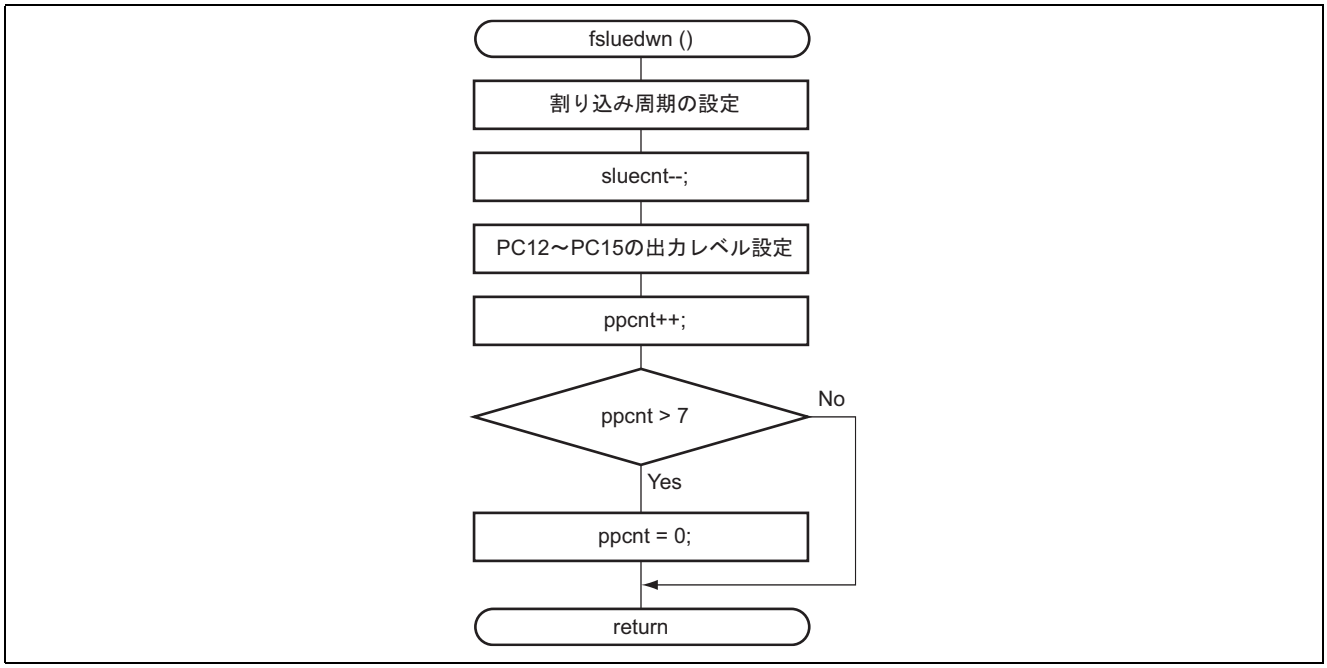

4
rslueup ()
modecnt--;
指定回転数終了？
No
Yes
グローバル変数（次モード）のセット
nextmode = 5 （逆転定速回転）
modecnt = 960（10回転）
sluecnt = 95（周期の選択）
4'
5
rconst ()
modecnt--;
指定回転数終了？
No
Yes
グローバル変数（次モード）のセット
nextmode = 6 （逆転Slew-down）
modecnt = 96 （1回転）
5'


6
rsluedwn ()
modecnt--;
指定回転数終了？
No
Yes
グローバル変数（次モード）のセット
nextmode = 7 （停止）
modecnt = 96 （1回転）
sluecnt = 0 （周期の選択）
6'
7
frstop ()
modecnt--;
指定回転数終了？
No
Yes
グローバル変数（次モード）のセット
nextmode = 0 （正転Slew-up）
modecnt = 96 （1回転）
7'

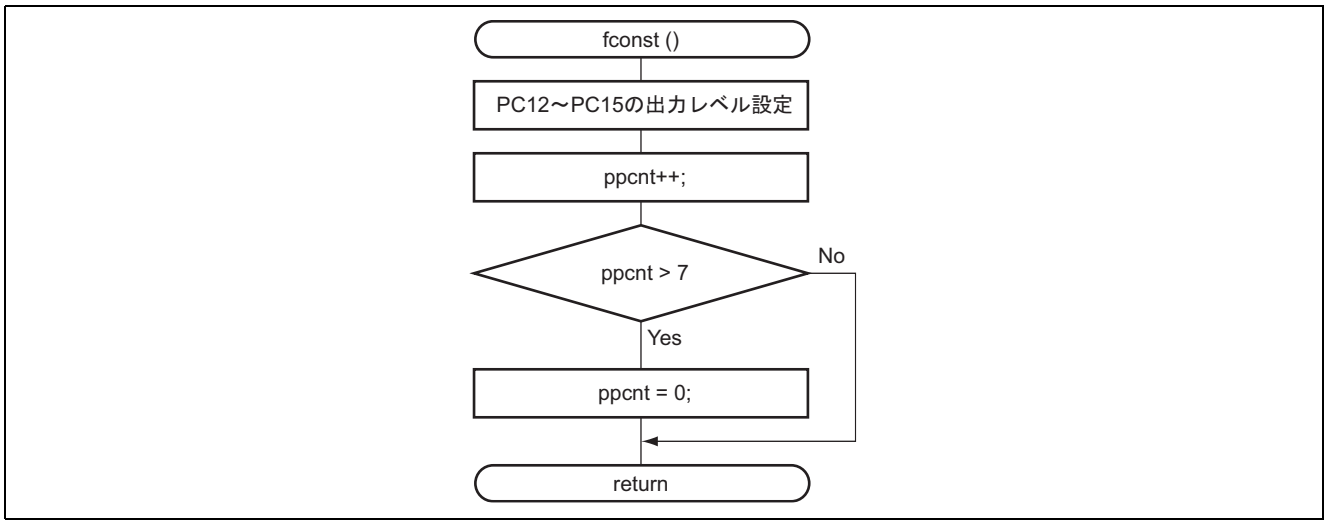
5.3 正転時 Slew-up ルーチン



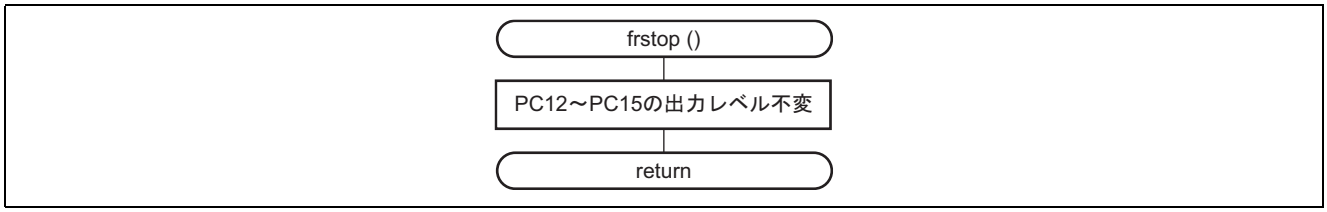
5.4 正転時 Slew-down ルーチン



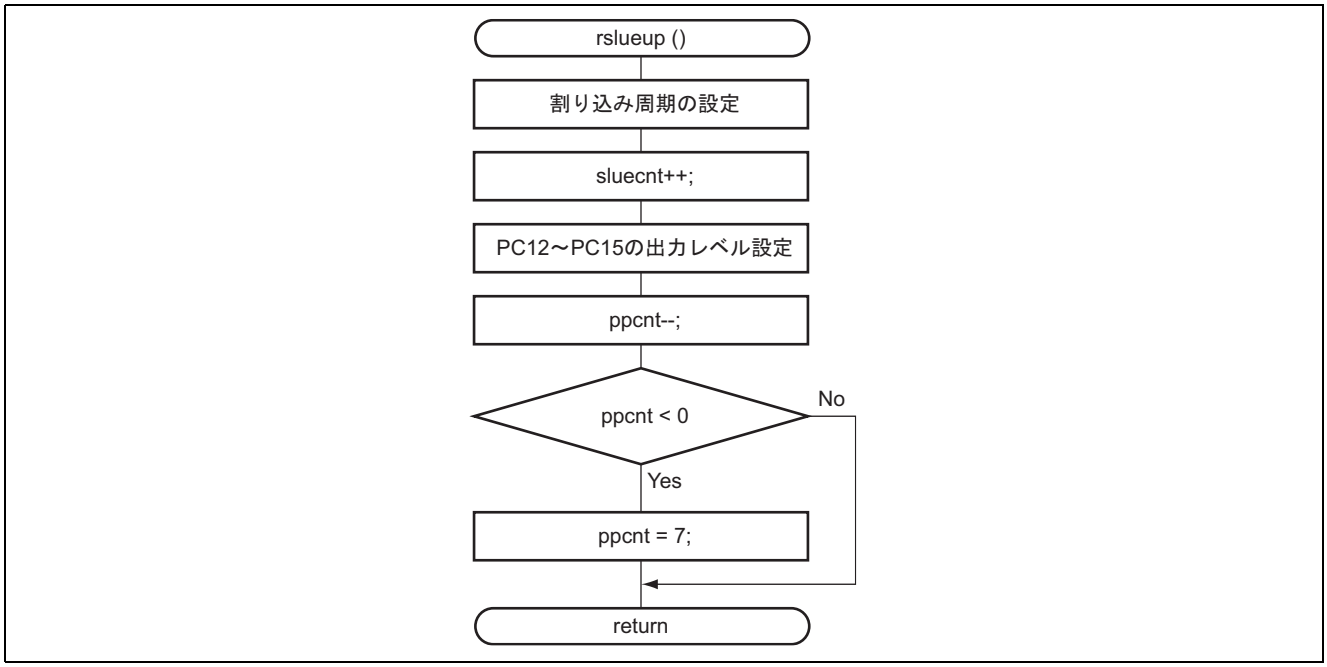
5.5 正転時定速回転ルーチン



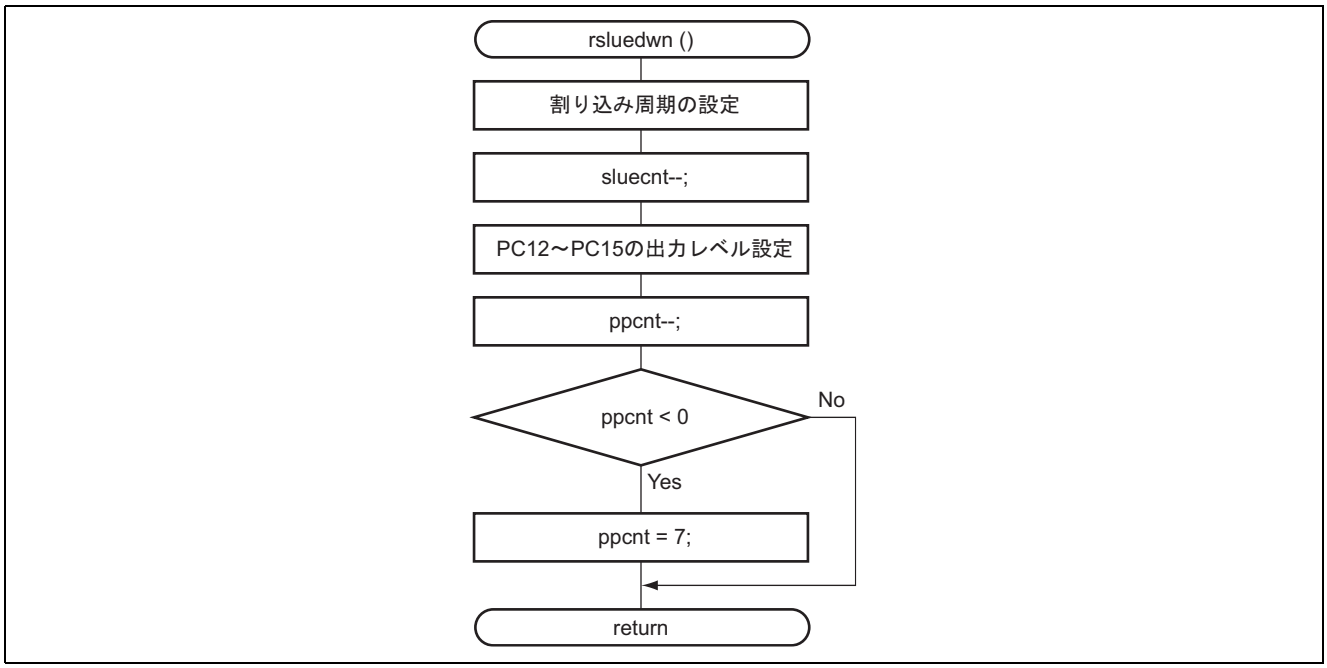
5.6 停止ルーチン



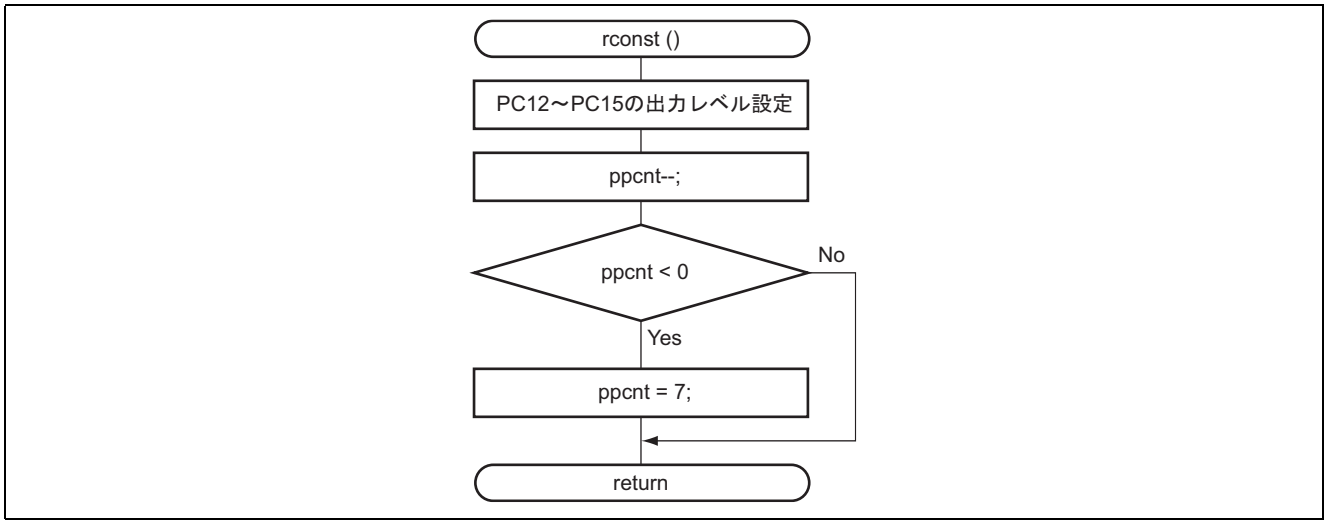
5.7 逆転時 Slew-up ルーチン



5.8 逆転時 Slew-down ルーチン



5.9 逆転時定速回転ルーチン



6. プログラムリスト

```
/************************************************************/
/* SH7145F Application Note                                 */
/*                                                          */
/* Function                                                 */
/* :CMT0(Stepping motor)                                    */
/*                                                          */
/* External input clock       :12.5MHz                      */
/* Internal CPU clock         :50MHz                        */
/* Internal peripheral clock  :25MHz                        */
/*                                                          */
/* Written  :2003/10   Rev.1.0                              */
/************************************************************/

#include "iodefine.h"
#include <machine.h>
/*------------ Function Definition ---------------------------------------*/
void main(void);
extern void _INITSCT(void);

void fslueup(void);
void fsluedwn(void);
void fconst(void);
void frstop(void);
void rslueup(void);
void rsluedwn(void);
void rconst(void);

void cmt_int(void);

void dummy_f(void);
/************************************************************/
/* Global Variable                                          */
/************************************************************/
char ppcnt;
unsigned char sluecnt;
unsigned char nextmode;
long modecnt;
/************************************************************/
/* Stepping Motor Output Patarn Table                       */
/************************************************************/

unsigned short pattb[8] = {                  /* Output Pulse Pattern Table */
   0x8000,0xC000,0x4000,0x6000,0x2000,0x3000,0x1000,0x9000
};

unsigned short int_cyc[96] = {          /* Interrupt Cycle Table           */
   0x8000,0x7800,0x7000,0x6A80,0x6720,0x639C,0x6018,0x5C94,0x5910,0x558C,
   0x5208,0x4EE8,0x4BC8,0x48A8,0x4650,0x43F8,0x41A0,0x4010,0x3E80,0x3D54,
   0x3C28,0x3AFC,0x3A5E,0x38A4,0x37DC,0x3714,0x364C,0x3584,0x34BC,0x33F4,
   0x335E,0x32C8,0x3232,0x319C,0x3106,0x30A2,0x303E,0x2FDA,0x2F76,0x2EE5,
   0x2E54,0x2DC3,0x2D32,0x2CA1,0x2C10,0x2B7F,0x2AEE,0x2A5D,0x29CC,0x293B,
   0x28AA,0x2819,0x2788,0x26F7,0x2666,0x25D5,0x2544,0x24B3,0x2422,0x2391,
   0x2300,0x226F,0x21DE,0x214D,0x20BC,0x202B,0x1F9A,0x1F09,0x1E78,0x1DE2,
   0x1D1A,0x1C48,0x1BCE,0x1AF4,0x1A4A,0x19EB,0x198C,0x192D,0x18CE,0x186F,
   0x1823,0x17F2,0x17B8,0x177B,0x173E,0x1701,0x16CE,0x169B,0x1668,0x1635,
```

```
   0x1602,0x15CF,0x159C,0x1569,0x1536,0x1500,
};
/************************************************************/
/* main Program                                             */
/************************************************************/
void main( void )
{
    ppcnt = 0;               /* Output Pulse Pattern table counter set */
    sluecnt = 0;                  /* Slue Up/Down table counter set   */
    nextmode = 0;
    modecnt = 95;            /* Motor Slue mode countset "96"          */

   P_STBY.MSTCR2.BIT.MSTP12 = 0;   /* Disable  CMT_0 standby mode      */

   P_CMT.CMCSR_0.WORD = 0x0041;    /* Set CMCSR_0                      */
         /* [15-8] = 0;Reserve                        */
         /* [7]    = 0;CMF                            */
         /* [6]    = 1;CMT interrupt enable           */
         /* [5-2]  = 0;Reserve                        */
         /* [1]    = 0                          */
         /* [0]    = 1;count clock=Pφ/32              */
   P_CMT.CMCOR_0 = int_cyc[sluecnt];     /* Set interrupt cycle table        */
   sluecnt++;

   P_CMT.CMCNT_0 = 0;              /* Clear CMCNT_0              */

   P_INTC.IPRG.BIT.CMT0 = 0xF;           /* Set CMT_0 interrupt level  */
   set_imask(0);             /* Set interrupt mask level               */

   P_PORTC.PCCR.WORD = 0;          /* Set function                     */

   P_PORTC.PCDR.WORD = pattb[ppcnt];     /* Set portC output pattern table   */
   ppcnt++;

   P_PORTC.PCIOR.WORD |= 0xF000;   /* PortC output                     */

   P_CMT.CMSTR.BIT.STR = 1;        /* Timer count start                */

   while(1);                 /* LOOP                           */
}
/************************************
   割込み処理
************************************/
#pragma interrupt(cmt_int)
void cmt_int(void)
{
   P_CMT.CMCSR_0.BIT.CMF = 0;            /* Clear CMF                  */

switch(nextmode){
   case 0:
         fslueup();          /* Forward Slue Up                */
         modecnt--;
         if(modecnt <= 0){               /* Next mode?                 */
            nextmode = 1;          /* nextmode = 1 Constant Speed        */
            modecnt = 960;         /* Next mode count set "960"          */
            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 count set "96"                  */
        }
              break;

        case 2:
            fsluedwn();         /* Forward Slue Down                */
            modecnt--;
        if(modecnt <= 0){                   /* Next mode?                       */
              nextmode = 3;           /* nextmode = 3 Slue Stop               */
              modecnt = 96;           /* Next mode count set "96"             */
              sluecnt = 0;            /* Slue Up/Down table counter set       */
            }
            break;

        case 3:
            frstop();                   /* Rotation Stop                    */
            modecnt--;
            if(modecnt <= 0){               /* Next mode?                       */
              nextmode = 4;           /* nextmode = 4 Reverse Slue Up         */
              modecnt = 96;           /* Next mode count set "96"             */
            }
            break;

        case 4:
            rslueup();          /* Reverse Slue Up                  */
            modecnt--;
            if(modecnt <= 0){               /* Next mode?                       */
              nextmode = 5;           /* nextmode = 5 Constant Speed          */
              modecnt = 960;          /* Next mode count set "960"        */
              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 count set "96"         */
            }
            break;

        case 6:
            rsluedwn();         /* Reverse Slue Down                */
            modecnt--;
            if(modecnt <= 0){               /* Next mode?                       */
              nextmode = 7;           /* nextmode = 7 Slue Stop               */
              modecnt = 96;           /* Next mode count set "96"         */
              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 count set "96"         */
         }
         break;
   }
}
/************************************************************/
/*  Forward Slue Up                                         */
/************************************************************/
void fslueup ( void )
{
   P_CMT.CMCOR_0 = int_cyc[sluecnt];     /* Set interrupt cycle table    */
   sluecnt++;

   P_PORTC.PCDR.WORD = pattb[ppcnt];     /* Set portC output pattern table */
   ppcnt++;

   if(ppcnt>7)
      ppcnt = 0;
}
/************************************************************/
/*  Forward Slue Down                                       */
/************************************************************/
void fsluedwn ( void )
{
   P_CMT.CMCOR_0 = int_cyc[sluecnt];     /* Set interrupt cycle table    */
   sluecnt--;

   P_PORTC.PCDR.WORD = pattb[ppcnt];     /* Set portC output pattern table */
   ppcnt++;

   if(ppcnt>7)
      ppcnt = 0;
}
/************************************************************/
/*  Forward Constant Speed                                  */
/************************************************************/
void fconst ( void )
{
   P_PORTC.PCDR.WORD = pattb[ppcnt];     /* Set portC output pattern table */
   ppcnt++;

   if(ppcnt>7)
      ppcnt = 0;
}
/************************************************************/
/*  Slue/Reverse Stop                                       */
/************************************************************/
void frstop ( void )
{
   P_PORTC.PCDR.WORD = pattb[ppcnt];     /* Set portC output pattern table */
```

```
}
/************************************************************/
/*  Reverse Slue Up                                         */
/************************************************************/
void rslueup ( void )
{
   P_CMT.CMCOR_0 = int_cyc[sluecnt];     /* Set interrupt cycle table        */
   sluecnt++;

   P_PORTC.PCDR.WORD = pattb[ppcnt];     /* Set portC output pattern table   */
      ppcnt--;

   if(ppcnt < 0)
      ppcnt = 7;
}
/************************************************************/
/*  Reverse Slue Down                                       */
/************************************************************/
void rsluedwn ( void )
{
   P_CMT.CMCOR_0 = int_cyc[sluecnt];     /* Set interrupt cycle table        */
   sluecnt--;

   P_PORTC.PCDR.WORD = pattb[ppcnt];     /* Set portC output patarn table    */
   ppcnt--;

   if(ppcnt < 0)
      ppcnt = 7;
}
/************************************************************/
/*  Reverse Constant Speed                                  */
/************************************************************/
void rconst ( void )
{
   P_PORTC.PCDR.WORD = pattb[ppcnt];     /* Set portC output patarn table    */
   ppcnt--;

   if(ppcnt < 0)
      ppcnt = 7;
}
/************************************************************/
#pragma interrupt(dummy_f)
void dummy_f(void)
{
   /* Other Interrput  */
}
```

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2004.09.15	—	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。