

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ひ申し上げます。

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のですが、誤りが無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、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と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。

# μPD78F0714によるモータ制御

## ホールICによる120度通電方式編

---

μPD78F0714

〔メモ〕

# 目次要約

第1章 概 説 ...	10
第2章 BLDCモータ制御の原理 ...	11
第3章 システム概要 ...	16
第4章 制御プログラム ...	22
付 録 ...	65

### 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。

CMOSデバイスの入力にノイズなどに起因して、 $V_{IL}$  (MAX.) から  $V_{IH}$  (MIN.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定な場合はもちろん、 $V_{IL}$  (MAX.) から  $V_{IH}$  (MIN.) までの領域を通過する遷移期間中にチャタリングノイズ等が入らないようご使用ください。

### 未使用入力の処理

CMOSデバイスの未使用端子の入力レベルは固定してください。

未使用端子入力については、CMOSデバイスの入力に何も接続しない状態で動作させるのではなく、プルアップかプルダウンによって入力レベルを固定してください。また、未使用の入出力端子が出力となる可能性（タイミングは規定しません）を考慮すると、個別に抵抗を介して  $V_{DD}$  または GND に接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については、その内容を守ってください。

### 静電気対策

MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジン・ケース、または導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。

また、MOSデバイスを実装したボードについても同様の扱いをしてください。

### 初期化以前の状態

電源投入時、MOSデバイスの初期状態は不定です。

電源投入時の端子の出力状態や入出力設定、レジスタ内容などは保証しておりません。ただし、リセット動作やモード設定で定義している項目については、これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は、まずリセット動作を実行してください。

### 電源投入切断順序

内部動作および外部インタフェースで異なる電源を使用するデバイスの場合、原則として内部電源を投入した後に外部電源を投入してください。切断の際には、原則として外部電源を切断した後に内部電源を切断してください。逆の電源投入切断順により、内部素子に過電圧が印加され、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源投入切断シーケンス」についての記載のある製品については、その内容を守ってください。

### 電源OFF時における入力信号

当該デバイスの電源がOFF状態の時に、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。

資料中に「電源OFF時における入力信号」についての記載のある製品については、その内容を守ってください。

- 本資料に記載されている内容は2007年5月現在のもので、今後、予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は、本資料の誤りに関し、一切その責を負いません。
- 当社は、本資料に記載された当社製品の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、一切その責を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路、ソフトウェアおよびこれらに関する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責を負いません。
- 当社は、当社製品の品質、信頼性の向上に努めておりますが、当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命、身体および財産に対する損害の危険を最小限度にするために、冗長設計、延焼対策設計、誤動作防止設計等安全設計を行ってください。
- 当社は、当社製品の品質水準を「標準水準」、「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。

標準水準：コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

特別水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器

特定水準：航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器、生命維持のための装置またはシステム等

当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には、事前に当社販売窓口までお問い合わせください。

(注)

- (1) 本事項において使用されている「当社」とは、NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社とその総株主の議決権の過半数を直接または間接に保有する会社をいう。
- (2) 本事項において使用されている「当社製品」とは、(1)において定義された当社の開発、製造製品をいう。

# はじめに

**対象者** このアプリケーション・ノートは、 $\mu$  PD78F0714の機能を理解し、それらを使用した応用システムを設計するユーザを対象とします。対象製品を次に示します。

・ $\mu$  PD78F0714

**目的** このアプリケーション・ノートでは、 $\mu$  PD78F0714を利用し、ホールIC付きブラシレスDCモータ( Brush Less DC Motor )を120度通電方式で駆動するシステムについてユーザに理解していただくことを目的としています。

**構成** このアプリケーション・ノートは大きく分けて次の内容で構成しています。

- ・概説
- ・システム概要
- ・BLDCモータ制御の原理
- ・制御プログラム

**読み方** このマニュアルの読者には、電気、論理回路、およびマイクロコンピュータに関する一般知識を必要とします。

ハードウェア機能の詳細（特にレジスタ機能とその設定方法など）、および電気的特性を知りたいとき

別冊の $\mu$  PD78F0714 **ユーザズ・マニュアル (U16928J)** を参照してください。

命令機能の詳細を理解しようとするとき

別冊の78K/0シリーズ **ユーザズ・マニュアル 命令編 (U12326J)** を参照してください。



- 凡 例** データ表記の重み：左が上位桁，右が下位桁  
 アクティブ・ロウの表記： $\overline{xxx}$ （端子，信号名称に上線）  
 メモリ・マップのアドレス：上部-上位，下部-下位  
 注：本文中に付けた注の説明  
 注意：気を付けて読んでいただきたい内容  
 備考：本文の補足説明  
 数の表記：2進数 ... xxxxまたはxxxxB  
           10進数... xxxx  
           16進数... xxxxH  
 2のべき数を示す接頭語（アドレス空間，メモリ容量）：  
           K（キロ） ...  $2^{10} = 1024$   
           M（メガ） ...  $2^{20} = 1024^2$   
           G（ギガ） ...  $2^{30} = 1024^3$   
 データ・タイプ：ワード ... 32ビット  
                   ハーフワード ... 16ビット  
                   バイト ... 8ビット

**関連資料** 関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

**デバイスの関連資料**

資料名	資料番号	
	和 文	英 文
$\mu$ PD78F0714 ユーザーズ・マニュアル	U16928J	U16928E
78K/0シリーズ ユーザーズ・マニュアル 命令編	U12326J	U12326E
$\mu$ PD78F0714によるインバータ制御 アプリケーション・ノート ゼロクロス検出による120度通電方式制御編	U17297J	U17297E
$\mu$ PD78F0714による単相インダクション・モータ制御 アプリケーション・ノート V/f制御による2相インバータ正弦波駆動編	U17481J	U17481E
$\mu$ PD78F0714によるモータ制御 アプリケーション・ノート ホールICによる120度通電方式編	このマニュアル	作成予定
$\mu$ PD78F0714によるモータ制御 アプリケーション・ノート センサレス（BEMF）による120度通電方式編	U18051J	作成予定

**注意** 上記関連資料は予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

### 開発ツール（ソフトウェア）の資料（ユーザズ・マニュアル）

資料名	資料番号	
	和文	英文
RA78K0 Ver.3.80 アセンブラ・パッケージ	操作編	U17199J U17199E
	言語編	U17198J U17198E
	構造化アセンブリ言語編	U17197J U17197E
CC78K0 Ver.3.70 Cコンパイラ	操作編	U17201J U17201E
	言語編	U17200J U17200E
SM+ システム・シミュレータ	操作編	U17246J U17246E
	ユーザ・オープン・インタフェース編	U17247J U17247E
ID78K0-QB Ver.2.81 統合デバッガ	操作編	U16996J U16996E
PM plus Ver.5.20		U16934J U16934E

### 開発ツール（ハードウェア）の資料（ユーザズ・マニュアル）

資料名	資料番号	
	和文	英文
QB-78K0KX1H インサーキット・エミュレータ	U17081J	U17081E

### フラッシュ・メモリ書き込み用の資料

資料名	資料番号	
	和文	英文
PG-FP3 フラッシュ・メモリ・プログラマ ユーザズ・マニュアル	U13502J	U13502E
PG-FP4 フラッシュ・メモリ・プログラマ ユーザズ・マニュアル	U15260J	U15260E

### その他の資料

資料名	資料番号	
	和文	英文
SEMICONDUCTOR SELECTION GUIDE - Products and Packages -	X13769X	
半導体デバイス 実装マニュアル	注	
NEC半導体デバイスの品質水準	C11531J	C11531E
NEC半導体デバイスの信頼性品質管理	C10983J	C10983E
静電気放電（ESD）破壊対策ガイド	C11892J	C11892E
半導体 品質 / 信頼性ハンドブック	C12769J	-
マイクロコンピュータ関連製品ガイド 社外メーカ編	U11416J	-

注 「半導体デバイス実装マニュアル」のホーム・ページ参照

和文：<http://www.necel.com/pkg/ja/jissou/index.html>

英文：<http://www.necel.com/pkg/en/mount/index.html>

注意 上記関連資料は予告なしに内容を変更することがあります。設計などには、必ず最新の資料をご使用ください。

# 目 次

## 第1章 概 説 ... 10

- 1.1 動作環境 ... 10
- 1.2 関連マニュアル ... 10

## 第2章 BLDCモータ制御の原理 ... 11

- 2.1 回転方向の定義 ... 11
- 2.2 回転原理 ... 11
- 2.3 通電パターン ... 12
- 2.4 インバータ ... 13
- 2.5 120度通電方式 ... 13
- 2.6 位置情報 ... 14
- 2.7 通電パターン切り替え ... 14
- 2.8 速度検出 ... 14
- 2.9 電圧制御 ... 15
- 2.10 速度制御 ... 15
  - 2.10.1 PID制御 ... 15

## 第3章 システム概要 ... 16

- 3.1 構 成 ... 16
- 3.2 インタフェース ... 17
- 3.3 機 能 ... 19
- 3.4 周辺I/O ... 20
- 3.5 割り込み ... 21

## 第4章 制御プログラム ... 22

- 4.1 通電パターン ... 22
  - 4.1.1 低電圧インバータ・セット ... 22
- 4.2 通電パターン切り替え ... 23
- 4.3 速度検出 ... 23
- 4.4 電圧制御 ... 24
- 4.5 速度制御 ... 24
  - 4.5.1 PID制御 ... 24
- 4.6 モジュール構成 ... 25
- 4.7 制御プログラムの関数一覧とフローチャート ... 25
  - 4.7.1 関数一覧 ... 25
  - 4.7.2 フローチャート ... 28
- 4.8 制御プログラムの変数・定数一覧 ... 48
  - 4.8.1 main.hの#defineで定義している値 ... 48
  - 4.8.2 lib\_eu.hの#defineで定義している値 ... 49
  - 4.8.3 変 数 ... 51
- 4.9 制御プログラムのソースファイル ... 52

## 付 録 ... 65

# 第1章 概 説

このシステムはホールIC付きブラシレスDCモータ（Brush Less DC Motor：以降BLDCモータ）を120度通電方式で駆動します。

- ・ このシステム（サンプル・プログラム）は NEC エレクトロニクスのモータ・スタータ・キット（ $\mu$ PD78F0714）<sup>※</sup>を利用し，ホールIC付きBLDCモータを120度通電方式で駆動します。
- ・ 制御ゲインは動作環境の特定のモータに合わせて調整しています。

注 モータ・スタータ・キット（ $\mu$ PD78F0714）については，弊社特約店にお問い合わせください。

## 1.1 動作環境

このシステムは以下の環境で使用することを前提に作成しています。

- ・ モータ・スタータ・キット（ $\mu$ PD78F0714）ボード一式
- ・ 低電圧インバータセット  
BLDCモータ PITTMAN(N2311A011)
  - ・ 基準電圧[V] : 12
  - ・ 無負荷回転速度[r/min] : 7197
  - ・ 連続トルク[Nm] : 0.11
  - ・ 最大トルク[Nm] : 0.23
  - ・ 駆動コイル : 3相（Y結線）
  - ・ 磁極ロータ : 4極（2極対）
  - ・ ステータ : 6スロット
  - ・ 位置センサ : ホールIC
- ・ PM plus 環境プラットフォーム V5.20
- ・ CC78K0 コンパイラ W3.60
- ・ RA78K0 アセンブラ W3.70
- ・ DF0714.78K デバイスファイル V1.10

## 1.2 関連マニュアル

開発環境およびボードにつきましては次に示すマニュアルを参照してください。

- ・ 低電圧モータ・スタータ・キット マニュアル
- ・ PM plus Ver.5.20 ユーザーズ・マニュアル
- ・ CC78K0 Ver.3.60 C コンパイラ 各ユーザーズ・マニュアル
- ・ RA78K0 Ver.3.70 アセンブラ・パッケージ 各ユーザーズ・マニュアル

## 第2章 BLDCモータ制御の原理

BLDCモータは固定子部分(ステータ)のコイルが発生する磁界の作用により、永久磁石でできた回転部分(ロータ)が回転します。

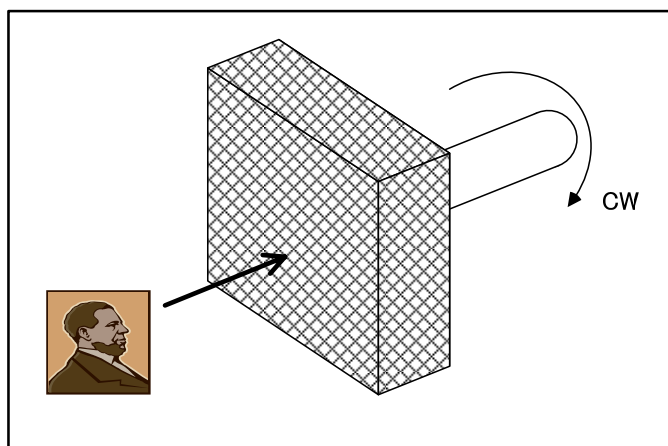
### 2.1 回転方向の定義

まず、モータの回転方向についての定義をします。

モータの回転方向は、CW(時計回り) / CCW(反時計回り)があります。

モータが回す対象物の回転方向を基準にしてCW / CCWが決定します。モータの軸がある面を対象物側に向けたときの回転方向を基準にします。したがって、下図のようになります。

図 2-1 モータの回転方向



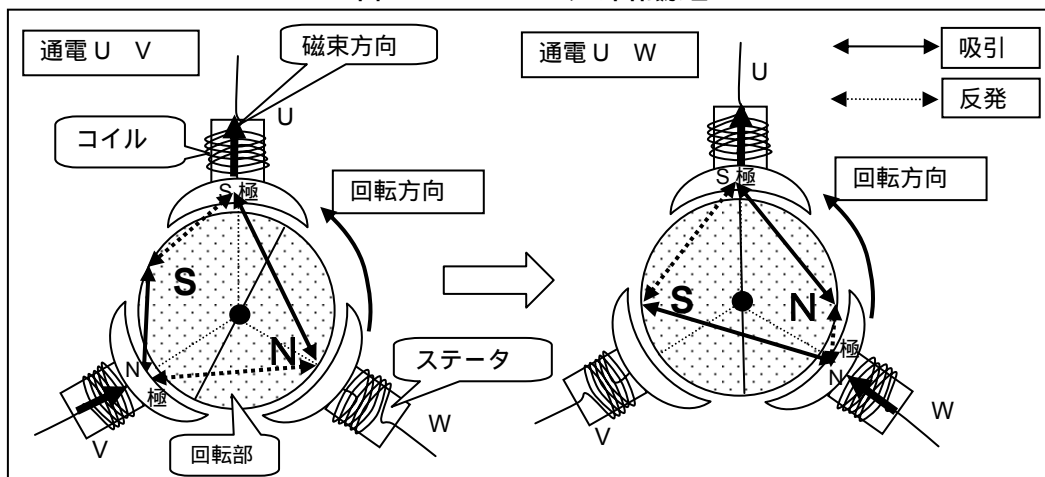
### 2.2 回転原理

BLDCモータの回転原理を記述します。

次の図に示したBLDCモータは3相2極3スロットY結線でインナーロータ型のSPM (Surface Permanent Magnet: 永久磁石を表面に配置した表面磁石構造)です。

ステータの極と回転部永久磁石の磁極との吸引および反発によって発生するマグネット・トルクで回転します

図 2-2 BLDCモータの回転原理

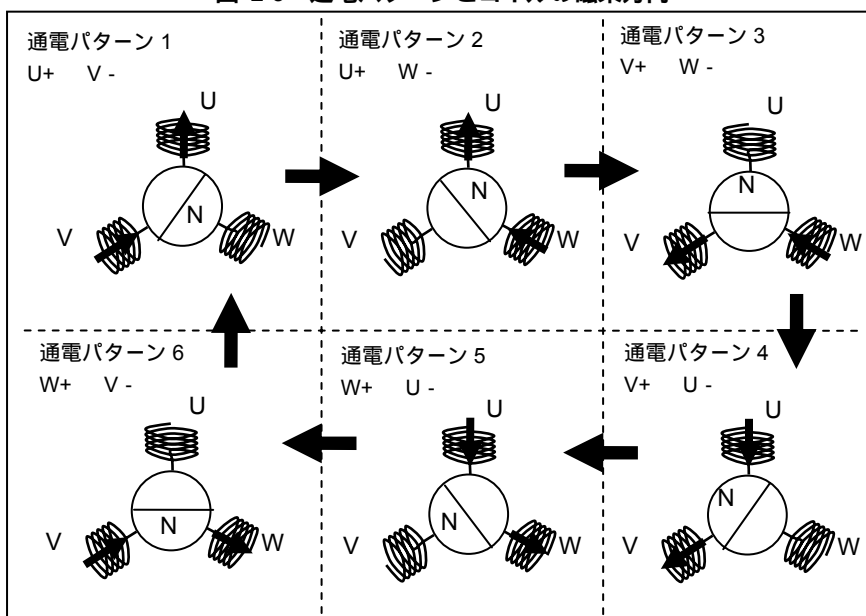


ステータの極はコイルの巻線方向に依存します。  
 回転部の磁極が逆の場合、回転方向が逆になります。

### 2.3 通電パターン

次の図に通電パターンとコイルで発生する電流磁束方向(ステータの極)と回転部の磁極の関係を示します。

図 2-3 通電パターンとコイルの磁束方向

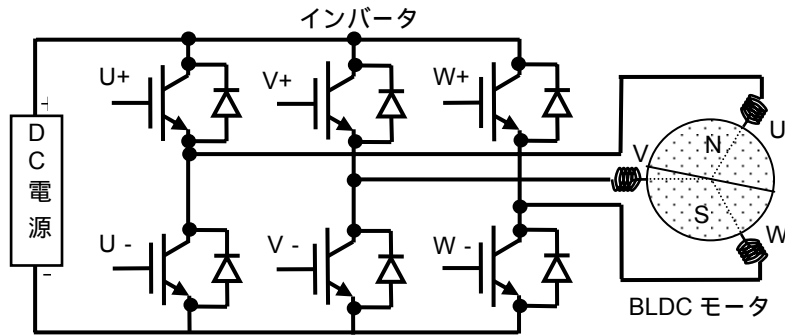


コイルの極は(コイルの)巻き線方向に依存します。

## 2.4 インバータ

ブラシと整流子のないBLDCモータは、コイルと電流の向きをインバータで切り替えます。次に3相Y結線BLDCモータとインバータとの結線図を示します。

図 2-4 インバータとBLDCモータ

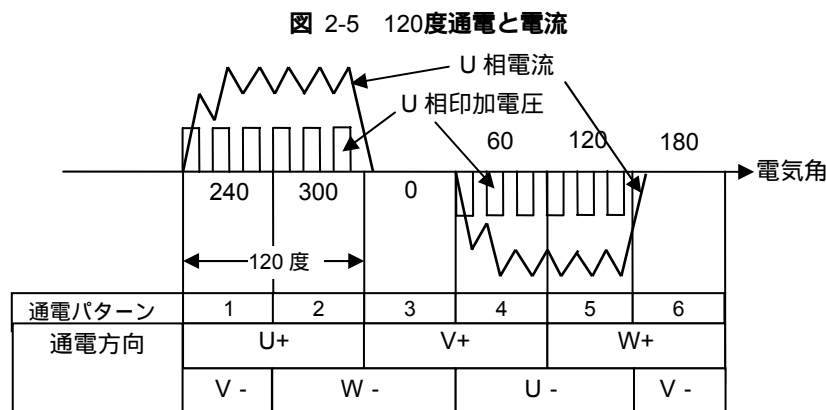


6個のスイッチング素子で通電時間や通電方向を制御します。

## 2.5 120度通電方式

次の図は2.3で示した通電パターンで各相に矩形波の電圧を印加したときのU相の状態（電圧，電流）を示したものです。U相には120度の通電期間と60度の無通電期間が繰り返し発生します。

各相の通電期間を120度で行うこの方式を120度通電方式といいます。

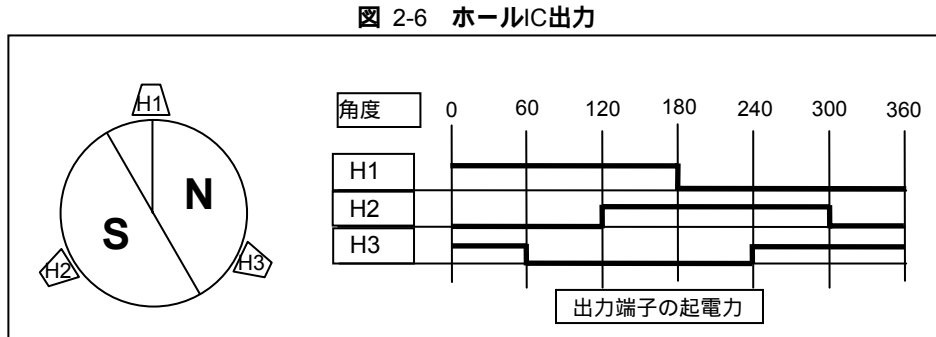


このシステムは、BLDCモータを120度通電方式で駆動します。

## 2.6 位置情報

120度通電方式ではインバータの切り換えを行うための位置情報が必要です。

次の図にホールICを利用した磁極位置センサの出力例を示します。



180度ごとに变化するホールICを，120度ごとに3個配置することで，60度ごとの位置検出が可能です。

ホールICの値と通電パターン（2.3参照）と位置情報（図2-6参照）を組み合わせた例を次の表に示します。

**表 2-1 ホールICと通電パターンの関係**

通電パターン ホール IC	1	2	3	4	5	6
H1	Lo	Hi	Hi	Hi	Lo	Lo
H2	Lo	Lo	Lo	Hi	Hi	Hi
H3	Hi	Hi	Lo	Lo	Lo	Hi

回転部の磁極が4極の場合，1つのホールICは90度ごとに値が変化し，通電パターンの切り替えは30度ごとに行います。

通常，1サイクル（6つの通電パターン）を電気角の360度，モータ軸の1回転を機械角の360度と定義します（このマニュアルではすべて電気角で記述しています）。

機械角：電気角 / 極対数

極対数：極数 / 2

## 2.7 通電パターン切り替え

ホールICによる回転部の磁極位置情報から回転方向を考慮して，通電パターンを設定します。

## 2.8 速度検出

ホールICの値が变化する時間を計測することで，モータの回転速度を求めます。



## 2.9 電圧制御

モータのコイルに加える電圧は、120度通電の矩形波をスイッチング素子のいずれかの導通期間を高い周波数でチョッパ動作させて通流率（平均電圧）を調整するPWM（Pulse Width Modulation）で制御します。

## 2.10 速度制御

モータのコイルに加える電圧（PWMの通流率）をPID制御で変更することで速度を制御します。

### 2.10.1 PID制御

指定された速度と検出した回転速度の偏差でPID制御を行い、PWM電圧制御法の通流率（以降、デューティ比と記述）を変更します。

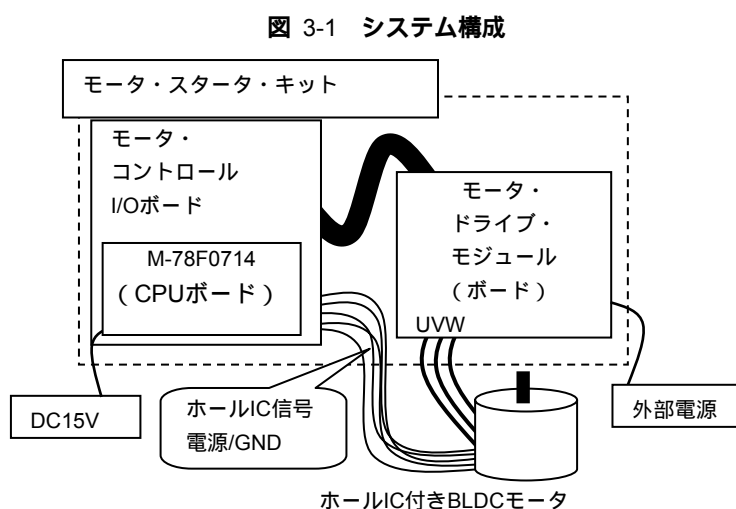
PID制御は、偏差に比例する出力を出す比例動作（Proportional action:P動作）、偏差の積分に比例する出力を出す積分動作（Integral action:I動作）、偏差の微分に比例する出力を出す微分動作（Derivative action:D動作）からPWMのデューティ比操作量を求めます。

## 第3章 システム概要

このシステムの概要について記述します。

### 3.1 構成

このシステムの構成を次の図に示します。

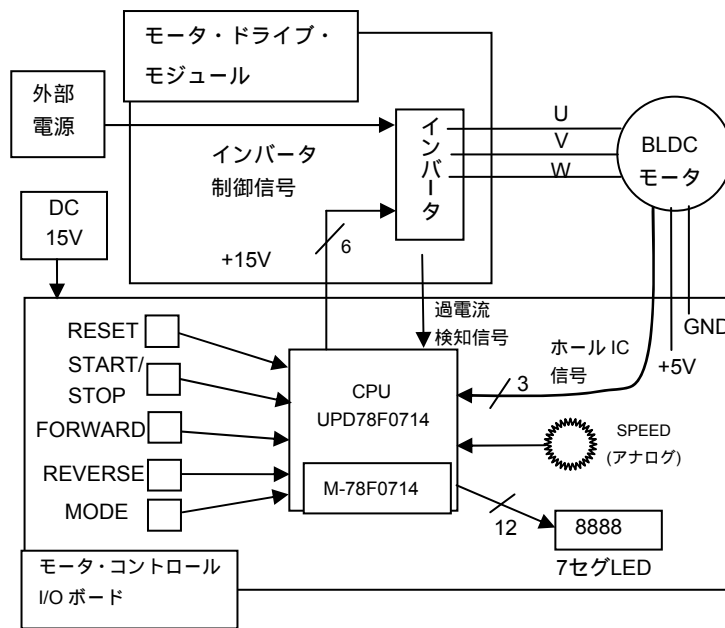


モータを駆動するモータ・ドライブ・モジュール，モータ制御用スイッチを搭載したモータ・コントロールI/Oボード，CPUを搭載したM-78F0714で構成されます。

BLDCモータは3相4極（2極対）のホールIC付きモータです。

モータ・スタータ・キットのブロック構成を次の図に示します。

図 3-2 ブロック構成



モータ・コントロールI/Oボード上のスイッチでモータの制御を行います。

### 3.2 インタフェース

表 3-1にユーザ・インタフェース一覧を示します。

表 3-1 ユーザ・インタフェース

機能名	部品番号	機能
RESET	SW1	リセット
START/STOP	SW2	スタート/ストップ
FORWARD	SW3	回転方向 (右回転, 時計回り, CW)
REVERSE	SW4	回転方向 (左回転, 反時計回り, CCW)
MODE	SW5	速度指定切り替え 速度表示切り替え
SPEED	R52	指定速度変更
7セグLED	DISP1 DISP2 DISP3 DISP4	速度の表示 (rpm) <sup>注</sup>

注 モータ停止中は常に指定速度を表示します。  
 指定速度固定時には右下に “.” を表示します。  
 モータ回転中は回転速度を表示します。  
 モータ回転中は MODE スイッチを押している間, 指定速度を表示します。  
 RESET を押している間はリセット継続, 離すとリセット解除です。  
 リセット解除直後 “ SELF ” と 1 秒表示します。

表 3-2にエラー一覧を示します。

表 3-2 エラー一覧

エラー	LEDの表示	状 況
H/W 過電流	O.C.	モータの電流が異常
ホールIC	HALL	ホールICの値が異常
装置異常	FAIL	モータが回転していない

表 3-3に $\mu$ PD78F0714端子のインタフェース一覧を示します。

表 3-3 端子のインタフェース

端子番号	端子名	機 能
8	RESET	RESET ( SW1 ) <sup>注</sup>
27-32	TW0TO0-TW0TO5	三相PWMインバータ切り替え
11	P01/INTP1	ホールIC信号 ( HALL1 )
10	P02/INTP2	ホールIC信号 ( HALL2 )
9	P03/INTP3	ホールIC信号 ( HALL3 )
49-52	P64-P67	7セグLED選択 ( LD_LED0-LD_LED3 )
56	P73	START/STOP ( SW2 )
55	P72	FORWARD ( SW3 )
54	P71	REVERSE ( SW4 )
53	P70	MODE ( SW5 )
41-48	P40-P47	7セグLEDへの出力データ
12	TW0TOFFP/INTP0	過電流検知 ( +5V 0V )
60	ANI4	速度変更 ( R52 )
20	P53/TI000/INTP5	タイマ・キャプチャ・トリガ
21	P54	モータ・ドライブ・モジュール制御

注 2JP7の1-2をショートします。

### 3.3 機能

表 3-4にこのシステムの機能と動作概要を示します。

表 3-4 システムの機能と動作概要 (1/2)

機能	概要
起動 (電源供給)	<ul style="list-style-type: none"> <li>・ LEDに “ SELF ” を1秒表示する</li> <li>・ SPEEDボリュームの指定速度 ( rpm ) をLEDに表示する</li> </ul>
RESETスイッチ	<ul style="list-style-type: none"> <li>・ モータの制御状態に関係なく、システムの再起動を行う</li> </ul>
START/STOP スイッチ	モータ制御停止中：モータ制御を開始する
	<ul style="list-style-type: none"> <li>・ LEDに “ 0 ” を表示する</li> <li>・ モータはCWで回転を開始する</li> <li>・ モータの回転速度 ( rpm ) をLEDに表示する</li> </ul>
	モータ制御中：モータ制御を停止する
	<ul style="list-style-type: none"> <li>・ モータの回転速度 ( rpm ) が0になるまでLEDに表示する</li> <li>・ SPEEDボリュームの指定速度 ( rpm ) をLEDに表示する</li> </ul>
	押し続けてもSTARTとSTOPはトグルしない
FORWARDスイッチ	モータ制御停止中：機能しない
	<ul style="list-style-type: none"> <li>・ 変化なし</li> </ul>
	モータ制御中：回転方向を変える
	<ul style="list-style-type: none"> <li>・ 回転方向がCCWの場合、一度停止してからCWになる</li> <li>・ 回転方向がCWの場合、変化しない</li> </ul>
REVERSEスイッチ	モータ制御停止中：機能しない
	<ul style="list-style-type: none"> <li>・ 変化なし</li> </ul>
	モータ制御中：回転方向を変える
	<ul style="list-style-type: none"> <li>・ 回転方向がCCWの場合、変化しない</li> <li>・ 回転方向がCWの場合、一度停止してからCCWになる</li> </ul>
MODEスイッチ	モータ制御停止中：速度指定を切り替える
	<ul style="list-style-type: none"> <li>・ SPEEDボリュームによる指定速度変更を無効 / 有効にする</li> <li>・ 無効時はLEDの数値右下に “ . ” が表示される</li> </ul>
	モータ制御中：LEDの表示を変える
	<ul style="list-style-type: none"> <li>・ 押している間、SPEEDボリュームの指定速度 ( rpm ) をLEDに表示する</li> </ul>
SPEEDボリューム	モータ制御停止中：指定速度の変更
	<ul style="list-style-type: none"> <li>・ LEDの表示速度 ( rpm ) が200 ~ 3200に変化する</li> <li>・ MODEスイッチで無効に設定されている場合は変化しない</li> </ul>
	モータ制御中：指定速度の変更
	<ul style="list-style-type: none"> <li>・ モータの回転速度 ( 平均 ) が指定した速度になる</li> </ul>
H/W過電流	<ul style="list-style-type: none"> <li>・ モータ・ドライブ・モジュールの許容電流超過で発生する</li> <li>・ モータ制御を停止する</li> <li>・ LEDに “ O.C. ” を表示する</li> <li>・ RESETスイッチ以外の機能を無効にする</li> </ul>

**注意** 複数のスイッチが押された場合の動作は保証していません。

表 3-4 システムの機能と動作概要 (2/2)

機 能	概 要
無回転	<ul style="list-style-type: none"> <li>・STARTから約2秒以上モータが回転しない場合、回転中で約0.05秒以上モータが停止した場合に発生する</li> <li>・モータ制御を停止する</li> <li>・LEDに“ FAIL ”を表示する</li> <li>・RESETスイッチ以外の機能を無効にする</li> </ul>
ホールIC異常	<ul style="list-style-type: none"> <li>・ホールICからの値が不正の場合発生する</li> <li>・モータ制御を停止する</li> <li>・LEDに“ HALL ”を表示する</li> <li>・RESETスイッチ以外の機能を無効にする</li> </ul>

注意 複数のスイッチが押された場合の動作は保証していません。

### 3.4 周辺I/O

このシステムでは次のような周辺I/Oを使用しています。

表 3-5 使用周辺I/O一覧

機 能	周辺I/O機能名 (μPD78F0714)
インバータ・タイマ	<ul style="list-style-type: none"> <li>・PWM出力用</li> <li>・10ビット・インバータ制御用タイマ (TW0UDCなど)</li> <li>・キャリア (変調) 周波数は13 KHz対称三角波</li> <li>・キャリア (変調) 同期割り込みが76.9 μs間隔で発生する</li> </ul> (メインのPID制御でデューティ比が計算され、キャリア同期割り込みで更新される)
リアルタイム出力	<ul style="list-style-type: none"> <li>・通電パタン切り替え用</li> <li>・リアルタイム出力ポート (RTBH01, RTBL01など)</li> <li>・16ビット・タイマ・キャプチャ/コンペア・レジスタ01 (CR01)</li> </ul> (キャリア同期割り込みで通電パタンが変更される)
ウェイト処理	<ul style="list-style-type: none"> <li>・タイミング調整用</li> <li>・8ビット・タイマ/イベントカウンタ50 (TM50, CR50)</li> </ul>
指定速度読み出し	<ul style="list-style-type: none"> <li>・可変抵抗値の電圧を指定速度に変換</li> <li>・A/Dコンバータ (ANI4)</li> </ul>
キャプチャ割り込み	<ul style="list-style-type: none"> <li>・速度計測用</li> <li>・割り込み機能 (TI000/INTP5)</li> </ul>
H/W過電流割り込み	<ul style="list-style-type: none"> <li>・割り込み機能 (INTP0)</li> <li>・モータ・ドライブ・モジュールの過電流発生 (LOWアクティブ)</li> </ul>
フェールセーフ	<ul style="list-style-type: none"> <li>・ウォッチドッグ・タイマ</li> </ul>

## 3.5 割り込み

このシステムで使用する割り込みの一覧を表3-6に示します。

表 3-6 使用割り込み一覧

名 称	機 能	発生条件
INTP0	過電流発生の検知	外部端子
INTP5	HALL1の変化検知（速度計測）	外部端子
INTTW0UD	キャリア同期割り込み発生	インバータ・タイマ・カウンタの アンダフロー
RESET	リセット発生	RESET端子
WDT	内部リセット発生	プログラム暴走によるウォッチ ドッグ・タイマ・オーバフロー

備考 INTTM50, INTAD は割り込み要求フラグのポーリングで処理

## 第4章 制御プログラム

このシステムでは基本的な制御プログラムを用いて実際にモータの速度制御を行います。

### 4.1 通電パターン

BLDCモータのスペックから端子情報と通電パターンとホールICの値を記述します。

#### 4.1.1 低電圧インバータ・セット

表 4-1 BLDCモータの端子スペック

カラー	機能	備考
BROWN	MOTOR $\phi$ A	U相
RED	MOTOR $\phi$ B	V相
ORANGE	MOTOR $\phi$ C	W相
GREY	SENSOR $\phi$ 1	HALL1
BLUE	SENSOR $\phi$ 2	HALL2
WHITE	SENSOR $\phi$ 3	HALL3
VIOLET	SENSOR Vcc	
BLACK	SENSOR GND	

表 4-2 通電パターンとホールICの値

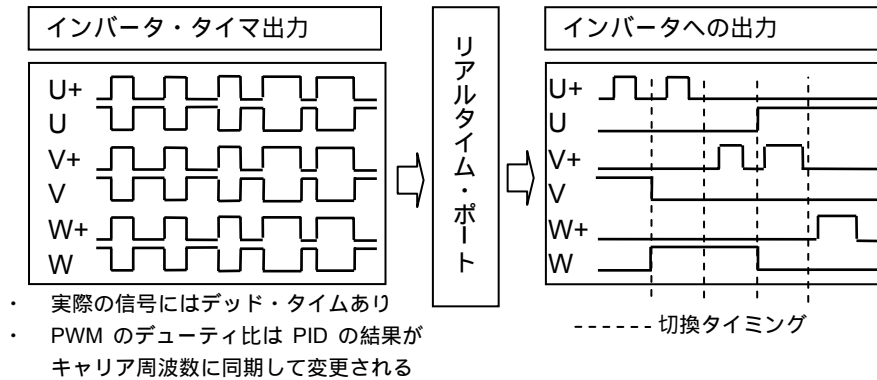
通電（正回転）	+A	+A	+B	+B	+C	+C
	-B	-C	-C	-A	-A	-B
SENSOR $\phi$ 1	L	L	H	H	H	L
SENSOR $\phi$ 2	H	L	L	L	H	H
SENSOR $\phi$ 3	H	H	H	L	L	L
通電（逆回転）	+B	+C	+C	+A	+A	+B
	-A	-A	-B	-B	-C	-C



## 4.2 通電パターン切り替え

通電パターンの切り替えはインバータ・タイマ出力 (PWM 波形) に同期して行う必要があるため、インバータ・タイマのキャリア同期割り込み (76.9  $\mu$ s 周期) でホール IC の値に対応する通電パターンをリアルタイム・ポートに設定して行います。

図 4-1 通電パターン切り替えタイミング



## 4.3 速度検出

タイマ・キャプチャ機能を用いてホールICの変化時点でのタイマのカウンタ値を瞬時に保存することで、通常の処理でのカウンタ値の保存の際に発生する遅延の影響がない速度計算により精度の高い速度検出を行います。

このシステムのBLDCモータ (3相4極) の場合、1つのホールICは1回転で4回変化しますが、使用するタイマ機能の仕様により立ち上がり側の変化のみ処理を行うため、1/2回転に変化したタイマ値から速度を計算します。

$$N = \frac{60}{s \times n \times 2} = \frac{2343750}{n}$$

$N$ : 1 分間の回転数 ( $rpm$ )

$s$ : タイマの分解能 (12.8  $\mu$ s)

$n$ : 発生回数

2: 変化回数

このシステムでは 200 rpm ~ 3200 rpm の回転速度範囲をサポートしています。

## 4.4 電圧制御

モータに印加する電圧は、インバータのプラス側を 13 kHz のキャリア周波数でチョッパ動作 (PWM) させた平均電圧で制御します。

マイナス側は PWM による相補動作を行わないため、減速時のブレーキ・トルクは発生しません(自然減速)。

## 4.5 速度制御

150 ms 周期に PWM のデューティ比を調整することでモータの回転速度を制御します。

デューティ比の調整量は指定速度とモータの回転速度の差を PID 制御で求めています。

### 4.5.1 PID制御

回転速度と指定速度の差をフィードバックして PWM のデューティ比 (平均電圧) に PID 制御を行うことで速度調整を行っています。PWM のデューティ比操作量は、サンプリング方式 (離散値) に適した以下の速度形 PID アルゴリズムを使用しています。

$$MV_n = MV_{n-1} + \Delta MV_n$$

$$\Delta MV_n = Kp(e_n - e_{n-1}) + Ki \times e_n + Kd((e_n - e_{n-1}) - (e_{n-1} - e_{n-2}))$$

$MV_n$ : 今回操作量

$MV_{n-1}$ : 前回操作量

$\Delta MV_n$ : 今回操作量差分

$e_n$ : 今回の偏差 (指定速度と実速度の差分)

$e_{n-1}$ : 前回の偏差

$e_{n-2}$ : 前々回の偏差

$Kp$ : フィードバックゲイン (比例要素)

$Ki$ : フィードバックゲイン (積分要素)

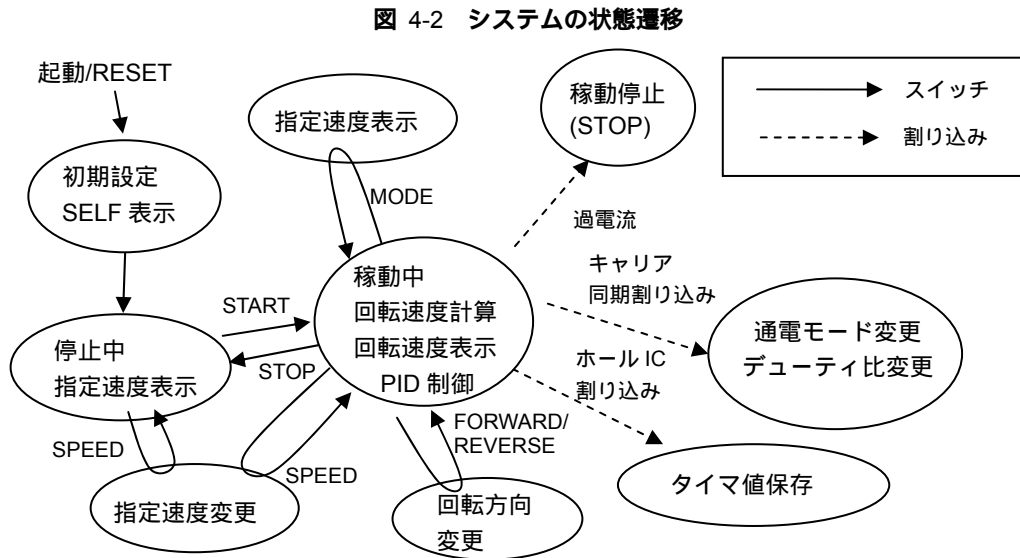
$Kd$ : フィードバックゲイン (微分要素)

フィードバック・ゲインの最適値はモータ特性や負荷の有無で変化します。

PID 制御の周期は最低速度 (200 rpm) での速度更新間隔 (150 ms) に設定しています。

## 4.6 モジュール構成

システムの状態遷移を図 4-2に示します。



キャリア同期割り込みは76.9  $\mu$ sごとに発生します。

## 4.7 制御プログラムの関数一覧とフローチャート

### 4.7.1 関数一覧

制御プログラムは複数の関数で構成されています。表 4-5に各関数の機能概要を示します。より詳細な処理については4.7.2 フローチャートを参照してください

表 4-5 関数の機能一覧 (1/2)

関数名	機能概要
void main(void)	<ul style="list-style-type: none"> <li>・メインルーチン</li> <li>・起動, 初期設定, スイッチ入力待ち無限ループ(起動, 停止, PID制御)</li> </ul>
void int_carrier(void)	<ul style="list-style-type: none"> <li>・キャリア同期割り込み</li> <li>・モータ無回転の検出(エラー処理)</li> <li>・速度情報の取得</li> <li>・PWMのデューティ比変更</li> <li>・通電パターン切り替え</li> </ul>
void int_faulta(void)	<ul style="list-style-type: none"> <li>・過電流割り込み</li> <li>・制御停止</li> <li>・エラー表示</li> </ul>
void system_init(void)	<ul style="list-style-type: none"> <li>・初期設定</li> <li>・内蔵周辺の初期化</li> <li>・LEDへの“SELF”表示</li> </ul>
void system_start(void)	<ul style="list-style-type: none"> <li>・モータ制御開始</li> <li>・変数に開始時の値設定</li> <li>・内蔵周辺タイマ起動</li> <li>・割り込みマスク解除</li> </ul>
void system_restart(void)	<ul style="list-style-type: none"> <li>・反転停止後のモータ制御再開</li> <li>・変数に再開時の値設定</li> </ul>
void system_stop(void)	<ul style="list-style-type: none"> <li>・モータ制御停止</li> <li>・割り込みマスク設定</li> <li>・変数に停止時の値設定</li> <li>・内蔵周辺タイマ停止</li> </ul>
void INTP0_on(void)	<ul style="list-style-type: none"> <li>・過電流割り込み許可</li> <li>・割り込みマスク解除</li> </ul>
void print_error(char)	<ul style="list-style-type: none"> <li>・エラー表示</li> <li>・7セグLEDにエラーを表示</li> <li>・STOP命令でプログラムを停止</li> </ul>
void pwm_pid(int)	<ul style="list-style-type: none"> <li>・速度のPID制御</li> <li>・キャリア同期割り込み回数やタイマの値から速度を計算</li> <li>・指定時間ごとに指定速度との差分でPID制御を行い, PWMデューティ比を計算</li> </ul>
void get_speed(void)	<ul style="list-style-type: none"> <li>・指定速度読み出し</li> <li>・可変抵抗の電圧をA/D変換した値から指定速度を計算</li> <li>・200 ~ 3200 rpmの指定速度を変数に設定</li> </ul>
unsigned char get_sw(void)	<ul style="list-style-type: none"> <li>・スイッチの読み出し</li> <li>・制御スイッチの状態を返す</li> </ul>
void init_PORT(void)	I/Oポートの初期設定
void init_OSC(void)	CPUクロックの切替
void init_TW0(void)	10ビット・インバータ・タイマの初期設定
void set_TW0(int, int, int, int)	PWMのデューティ比変更
void start_TW0(void)	PWMの出力開始
void stop_TW0(void)	PWMの出力停止

表 4-5 関数の機能一覧 (2/2)

関数名	機能概要
void init_TM00(void)	16ビット・タイマ(TM00)初期設定
void start_TM00(void)	16ビット・タイマ(TM00)起動
void stop_TM00(void)	16ビット・タイマ(TM00)停止
void init_RTPM01(void)	リアルタイム・ポートの初期設定
void set_RTPM01 (unsigned char)	リアルタイム・ポートの設定 通電パターン (PWMを出力するポート) を変更
void start_RTPM01(void)	リアルタイム・ポート機能動作
void stop_RTPM01(void)	リアルタイム・ポート機能動作停止
void init_AD(void)	A/D機能初期設定 SPEEDボリュームのA/D変換を設定
void start_AD(void)	A/D変換開始
unsigned char get_AD(char)	A/D変換値の読み出し
void init_WDTM(void)	ウォッチドッグ・タイマの初期設定
void clear_WDTM(void)	ウォッチドッグ・タイマのカウンタ・クリア
void reset_WDTM(void)	ウォッチドッグ・タイマによる内部リセット信号発生
void init_TM50(void)	8ビット・タイマ (TM50) の初期設定 1 msタイマで使用
void start_TM50(void)	8ビット・タイマ (TM50) 起動
void wait_TM50(void)	8ビット・タイマ (TM50) 待ち
void speed_print(int)	LEDに速度を表示する モータ制御停止中は指定速度を表示する モータ制御中でMODEスイッチが押されている場合は、モータの回転速度を表示する
void INTTW0UD_on(void)	キャリア同期割り込み許可
void INTTW0UD_off(void)	キャリア同期割り込み禁止
void INTP5_on(void)	INTP5割り込み許可
void INTP5_off(void)	INTP5割り込み禁止
void led_print(int, char)	LEDに1～4桁の値を表示する
void led_set (unsigned char, unsigned char)	指定されたLEDに値を出力する
void wait(int)	指定時間 (ms) 経過待ち
char read_Hall_IC(void)	ホールICの値を読み出す
void int_speed(void)	INTP5割り込み

### 4.7.2 フローチャート

図 4-3から図 4-46に、各関数のフローチャートを示します。

<メインルーチン:main( )>

図 4-3 メイン処理 (1/2)

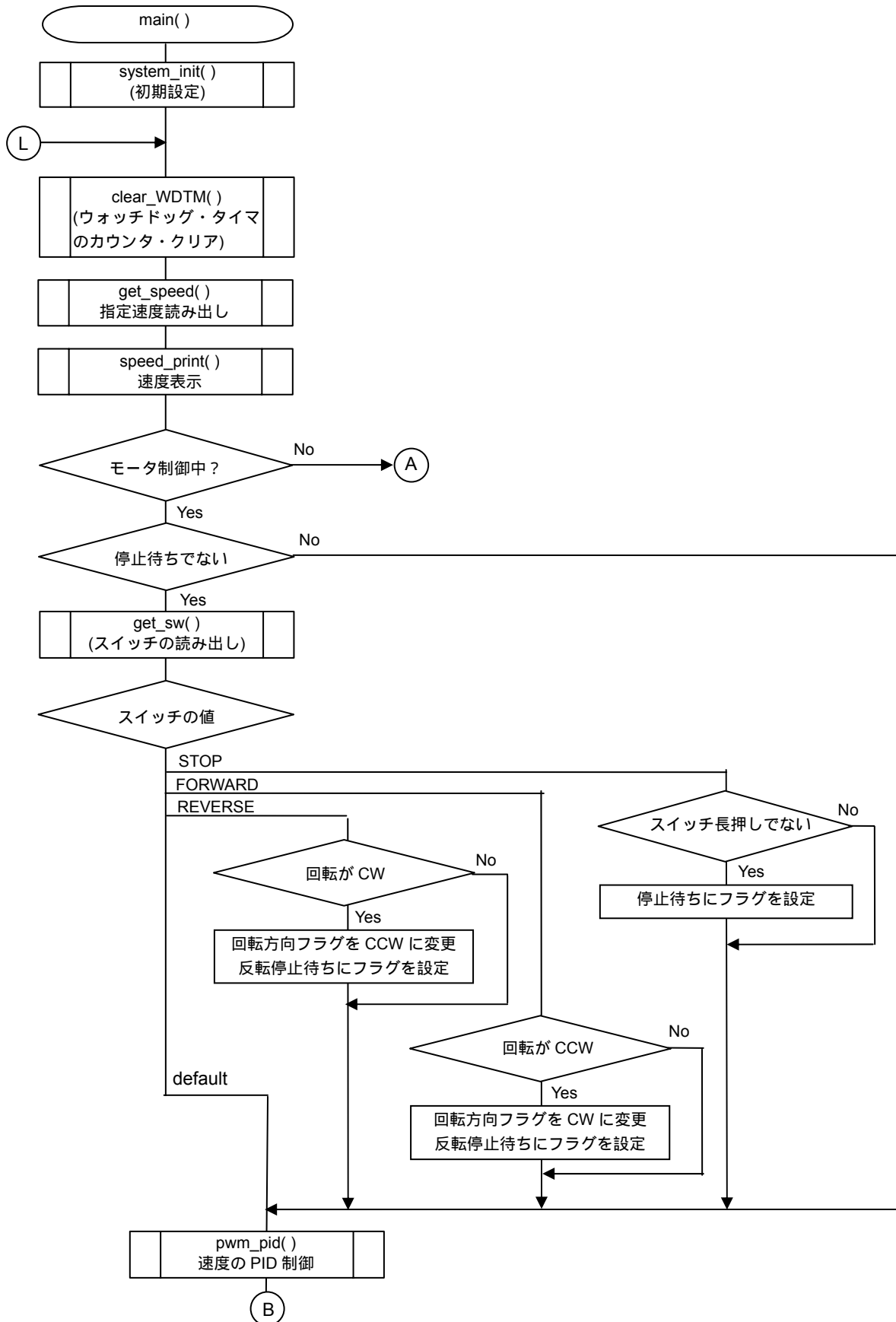
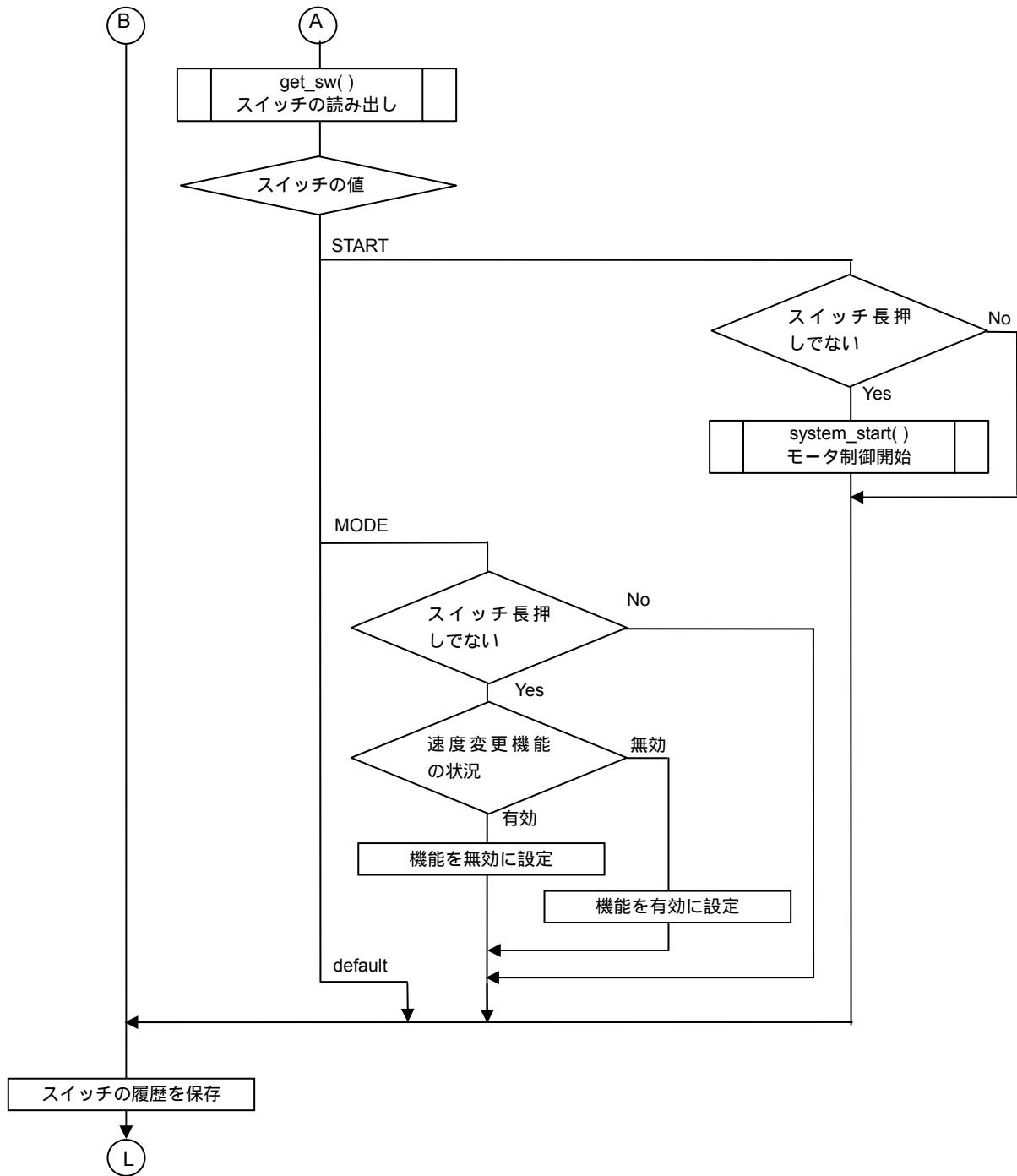


図 4-3 メイン処理 (2/2)



< キャリア同期割り込み >

図 4-4 キャリア同期割り込み処理 (1/2)

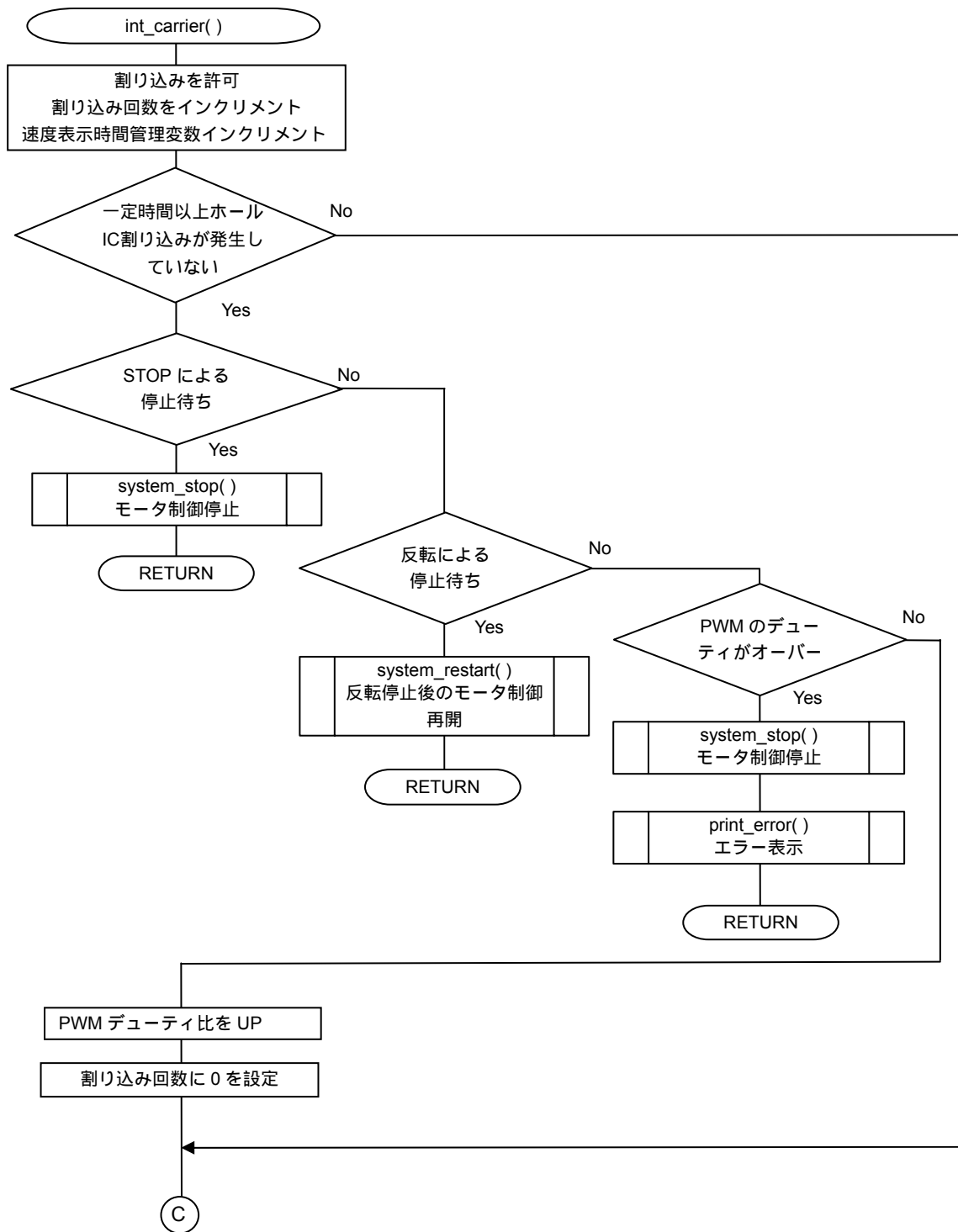
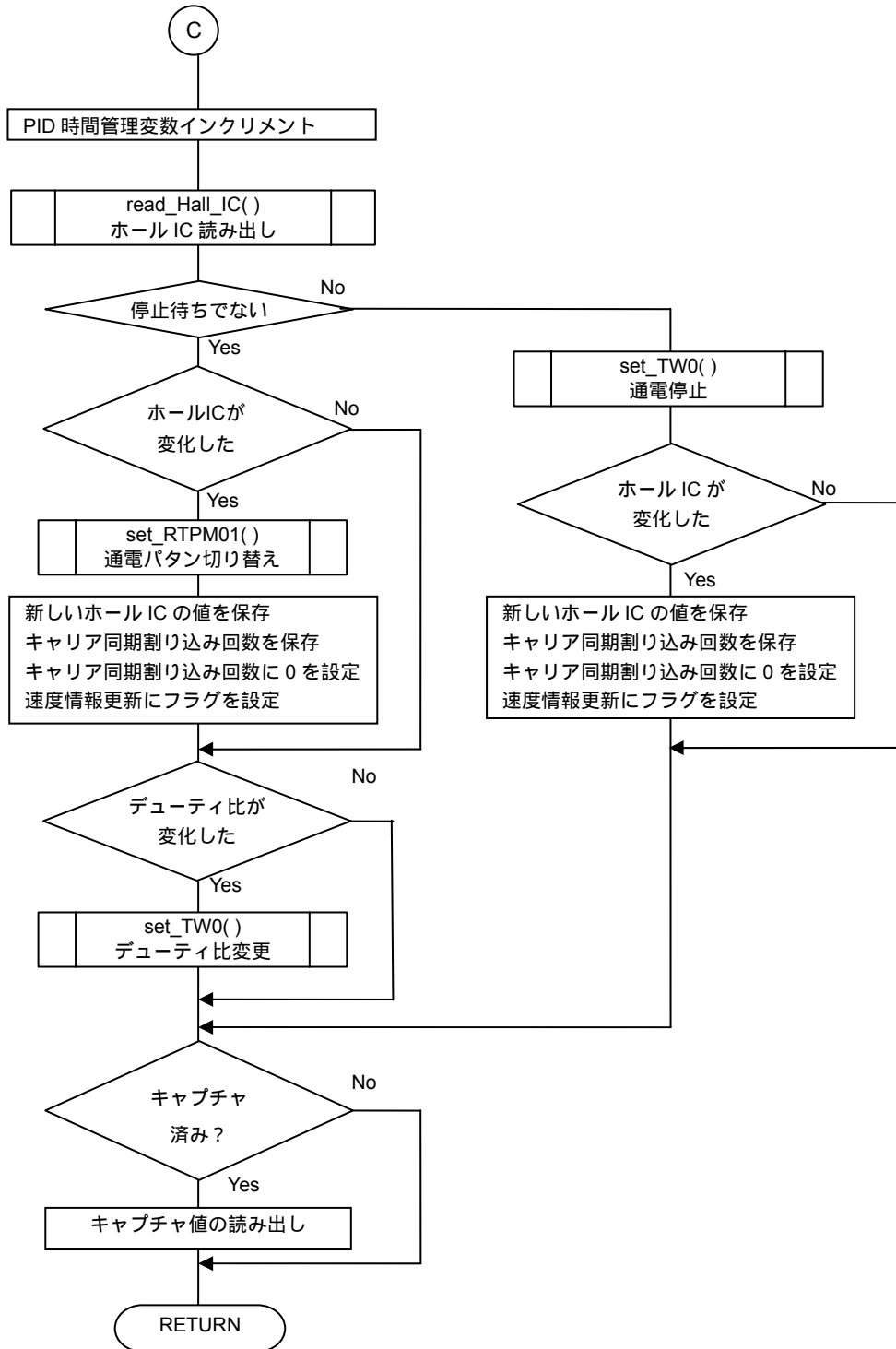


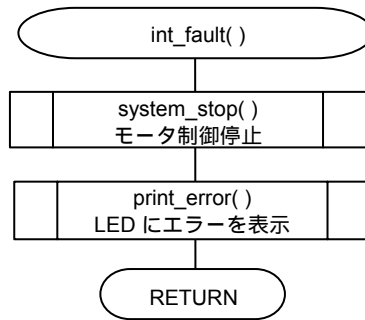


図 4-4 キャリア同期割り込み処理 (2/2)



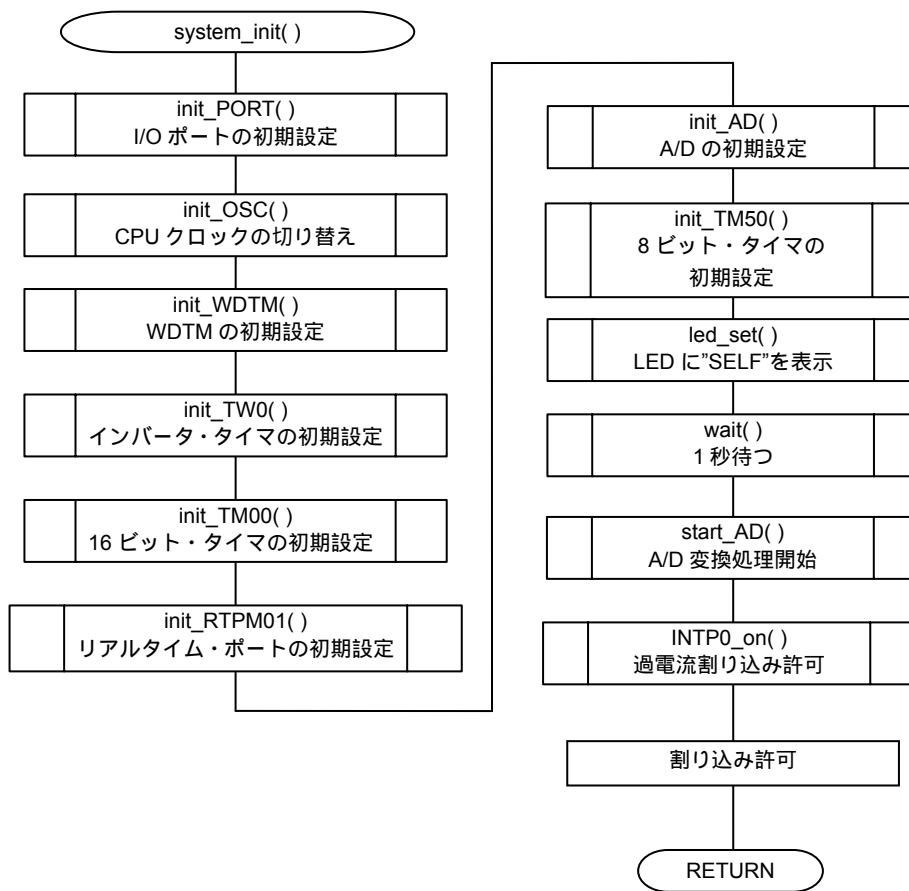
< 過電流割り込み >

図 4-5 過電流割り込み処理



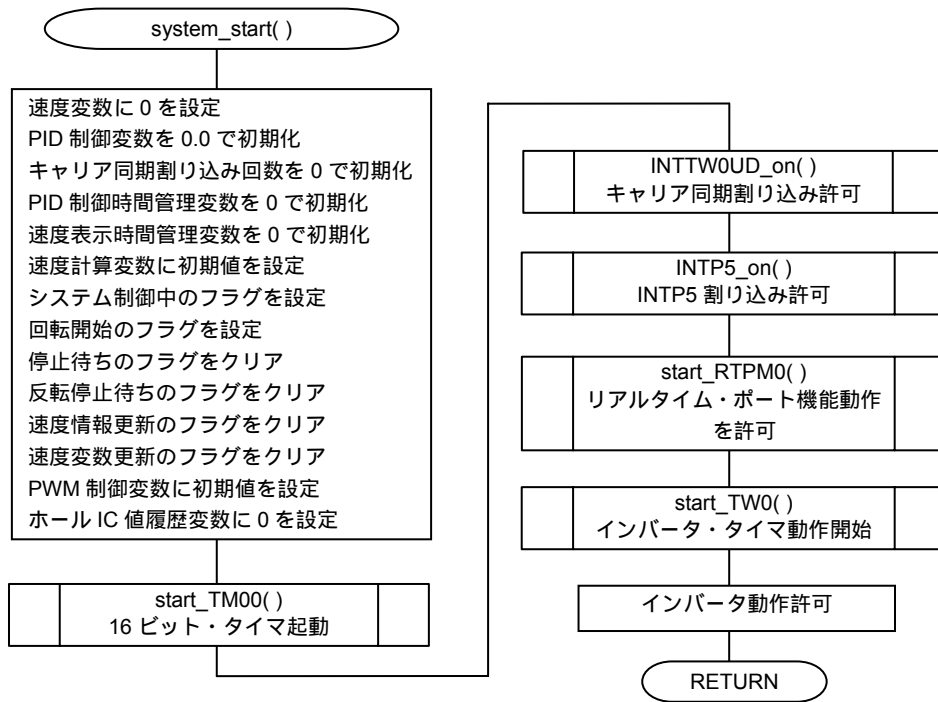
< 初期設定 >

図 4-6 初期設定処理



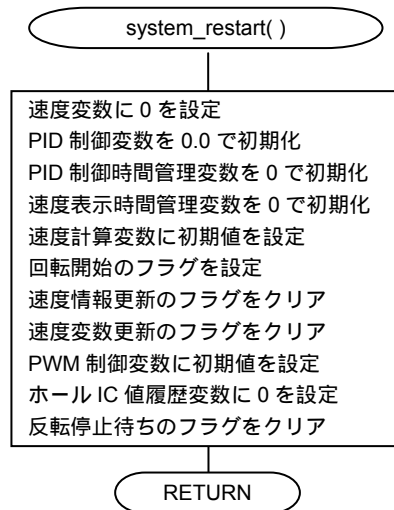
< モータ制御開始 >

図 4-7 モータ制御開始処理



< 反転停止後のモータ制御再開 >

図 4-8 反転停止後のモータ制御再開処理



< モータ制御停止 >

図 4-9 モータ制御停止処理



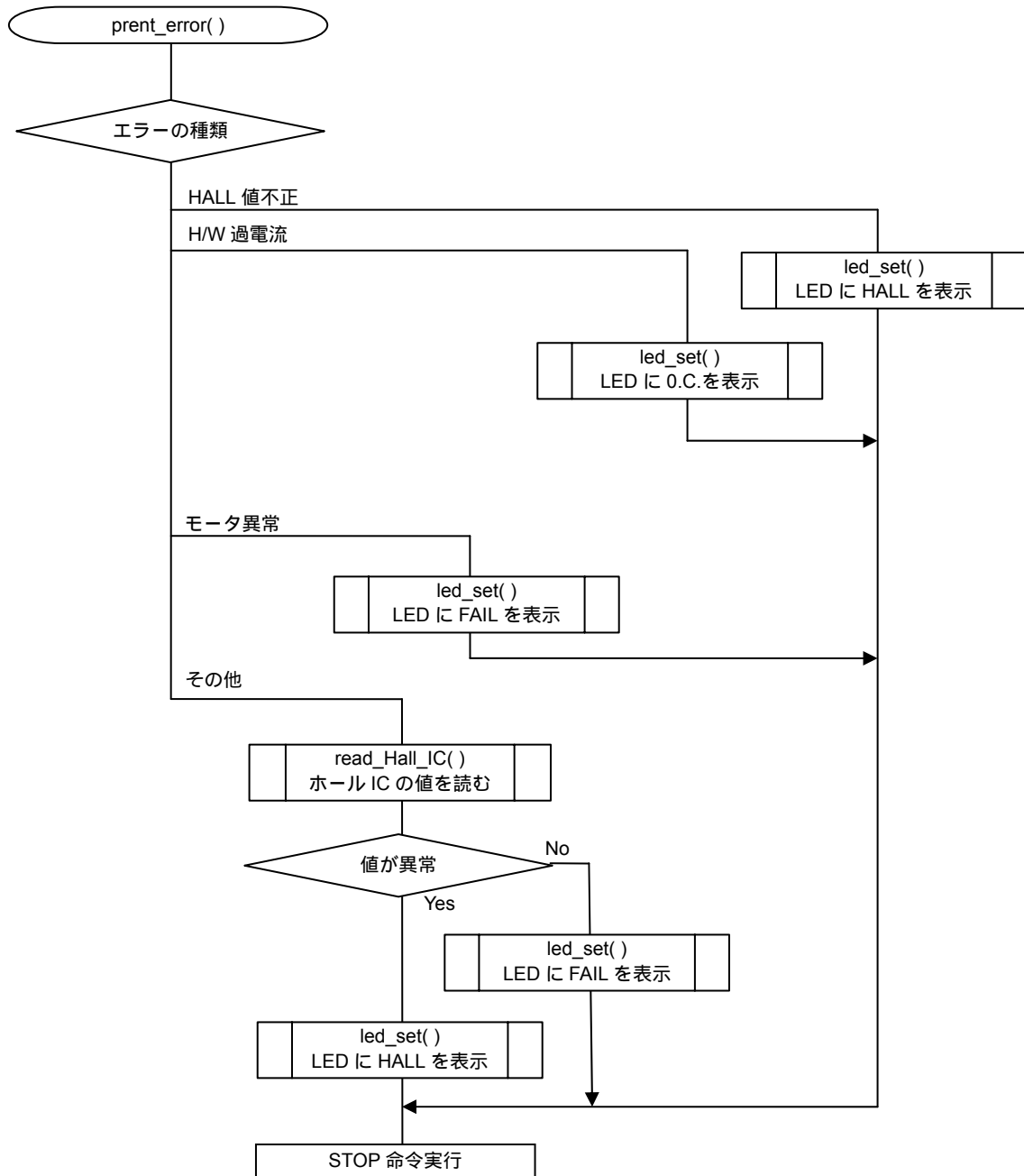
< 過電流割り込み許可 >

図 4-10 過電流割り込み許可処理



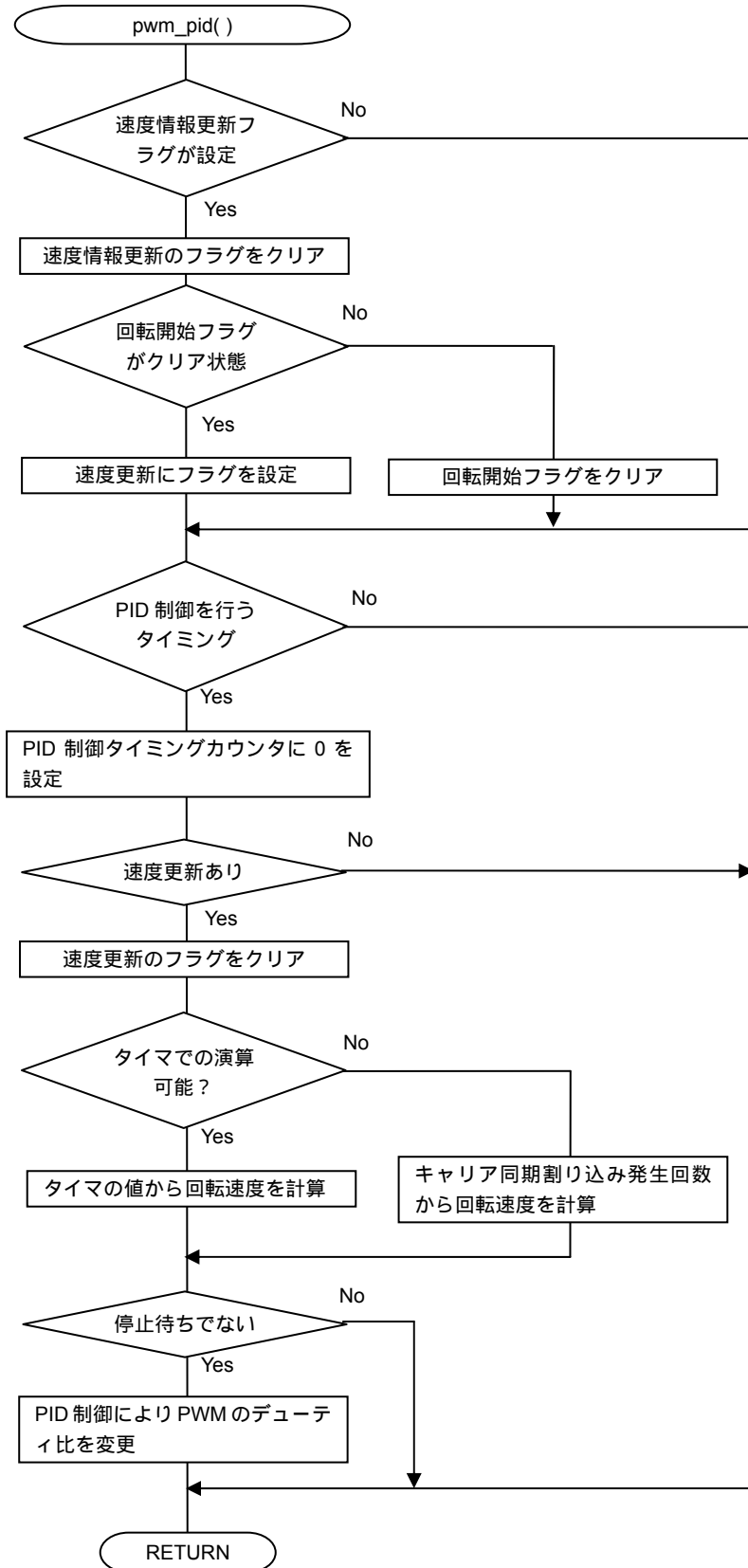
<エラー表示>

図 4-11 エラー表示処理



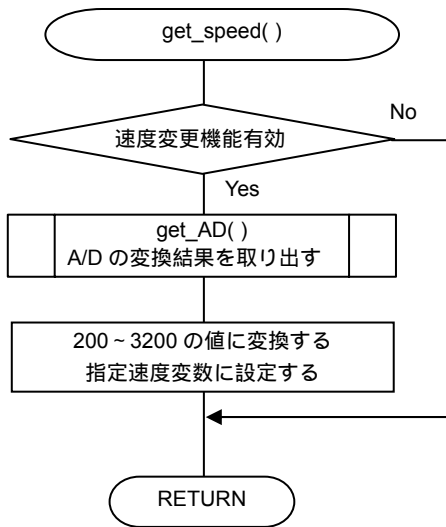
< 速度のPID制御 >

図 4-12 速度のPID制御処理



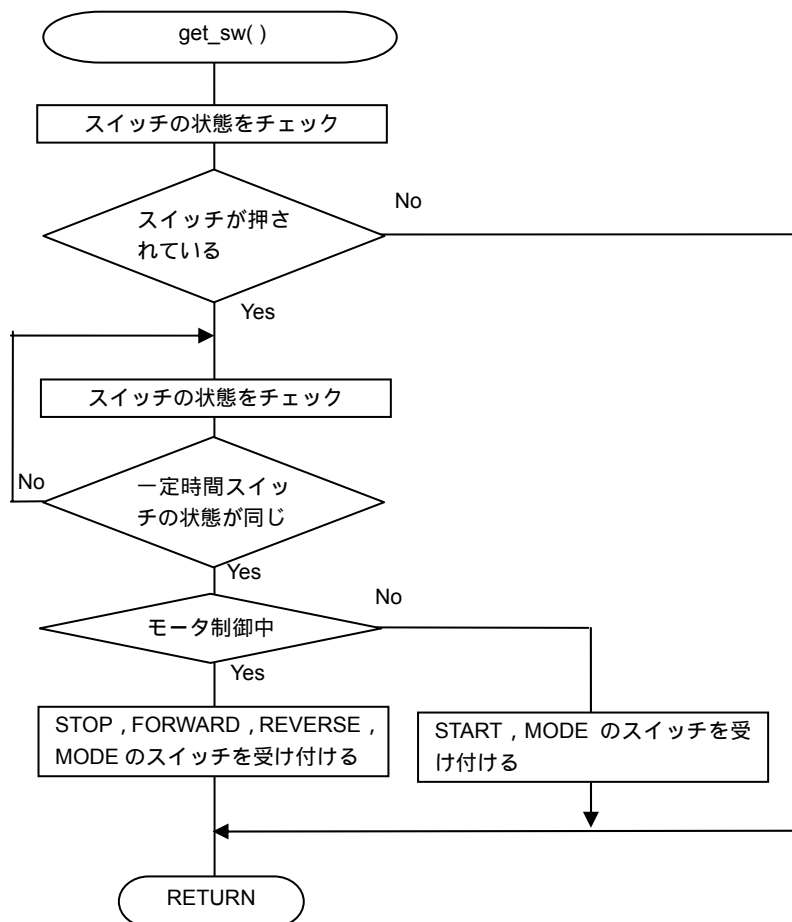
< 指定速度の読み出し >

図 4-13 指定速度の読み出し処理



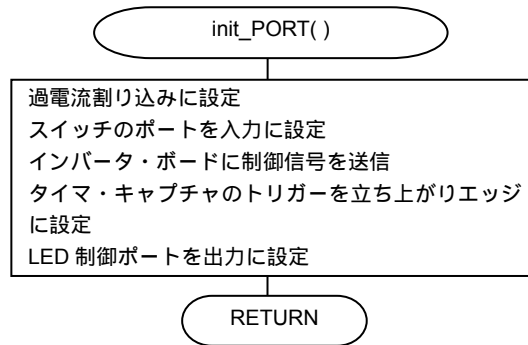
< スイッチ読み出し >

図 4-14 スイッチ読み出し処理



< I/Oポートの初期設定 >

図 4-15 I/Oポートの初期設定処理



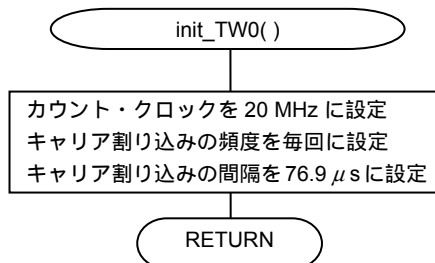
< CPUクロックの切り替え >

図 4-16 CPUクロック切り替え処理



< 10ビット・インバータ・タイマの初期設定 >

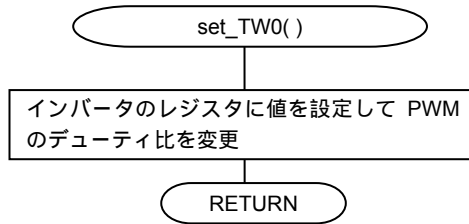
図 4-17 10ビット・インバータ・タイマの初期設定処理





< PWMのデューティ比変更 >

図 4-18 PWMのデューティ比変更処理



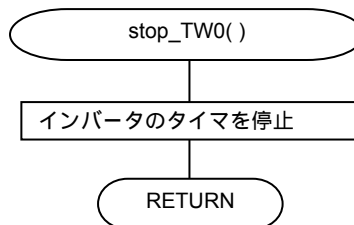
< PWMの出力開始 >

図 4-19 PWMの出力開始処理



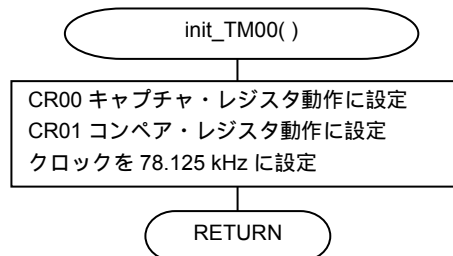
< PWM出力停止 >

図 4-20 PWMの出力停止処理



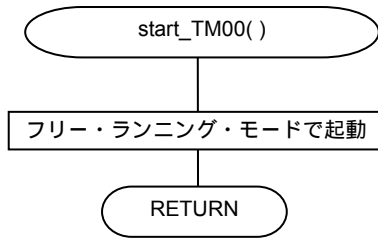
< 16ビット・タイマ (TM00) 初期設定 >

図 4-21 16ビット・タイマ (TM00) 初期設定処理



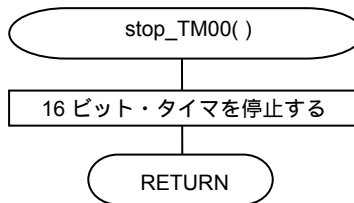
< 16ビット・タイマ (TM00) 起動 >

図 4-22 16ビット・タイマ (TM00) 起動処理



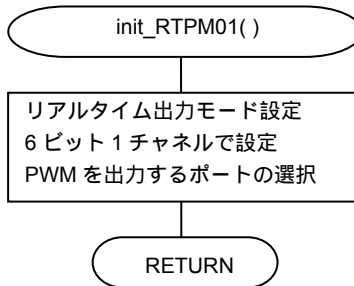
< 16ビット・タイマ (TM00) 停止 >

図 4-23 16ビット・タイマ (TM0) 停止処理



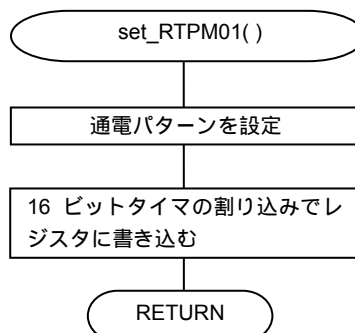
< リアルタイム・ポートの初期設定 >

図 4-24 リアルタイム・ポートの初期設定処理



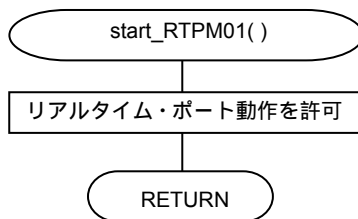
< リアルタイム・ポートの設定 >

図 4-25 リアルタイム・ポートの設定処理



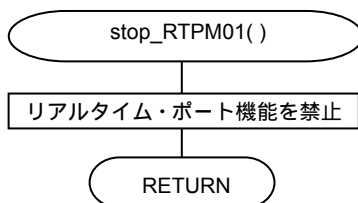
<リアルタイム・ポート起動>

図 4-26 リアルタイム・ポート機能動作処理



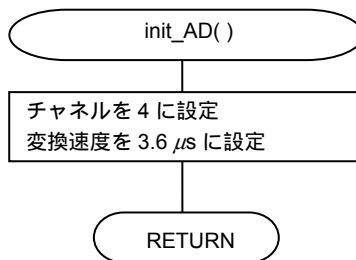
<リアルタイム・ポート停止>

図 4-27 リアルタイム・ポート機能動作停止処理



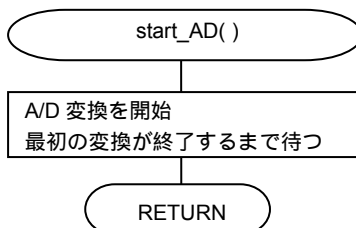
<A/D機能初期設定>

図 4-28 A/D機能初期設定処理



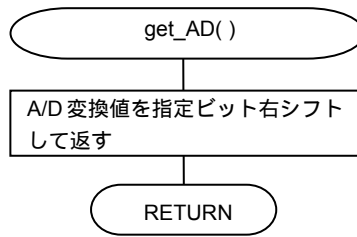
<A/D変換開始>

図 4-29 A/D変換開始処理



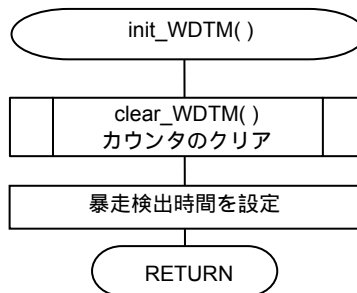
< A/D 変換値の読み出し >

図 4-30 A/D変換値の読み出し処理



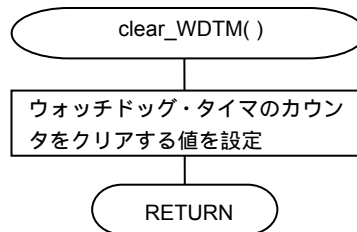
< ウォッチドッグ・タイマ初期設定 >

図 4-31 ウォッチドッグ・タイマの初期設定処理



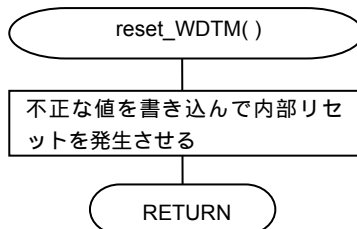
< ウォッチドッグ・タイマのカウンタクリア >

図 4-32 ウォッチドッグ・タイマのカウンタ・クリア処理



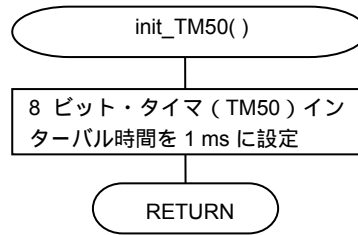
< ウォッチドッグ・タイマによる内部リセット信号発生 >

図 4-33 ウォッチドッグ・タイマによる内部リセット信号発生処理



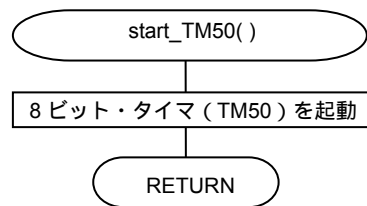
<8ビット・タイマ (TM50) の初期設定>

図 4-34 8ビット・タイマ (TM50) の初期設定処理



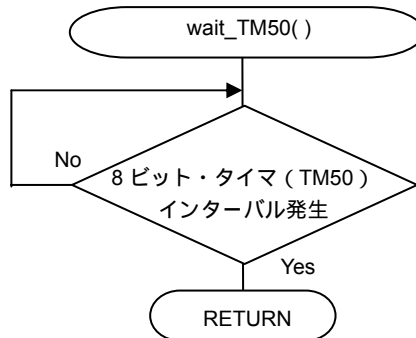
<8ビット・タイマ (TM50) 起動>

図 4-35 8ビット・タイマ (TM50) 起動処理



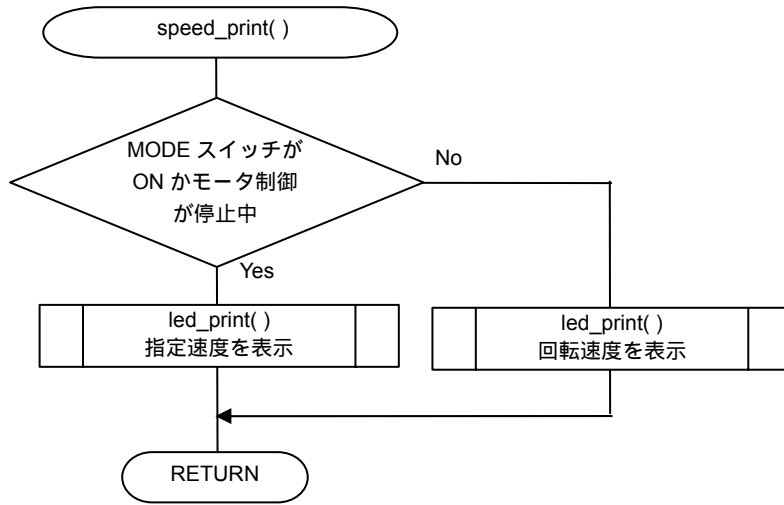
<8ビット・タイマ (TM50) 待ち>

図 4-36 8ビット・タイマ (TM50) 待ち処理



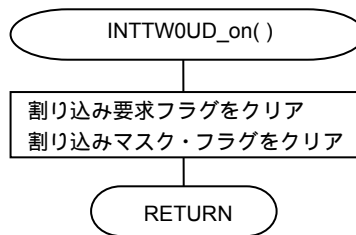
< LED に速度を表示する >

図 4-37 LEDに速度を表示する処理



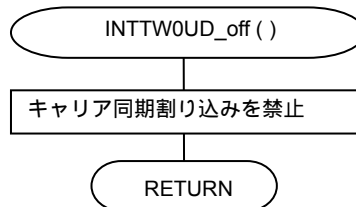
< キャリア同期割り込み許可 >

図 4-38 キャリア同期割り込み許可処理



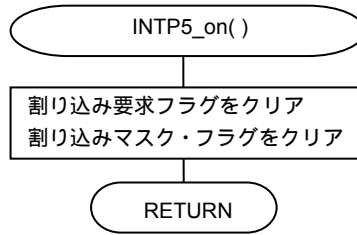
< キャリア同期割り込み禁止 >

図 4-39 キャリア同期割り込み禁止処理



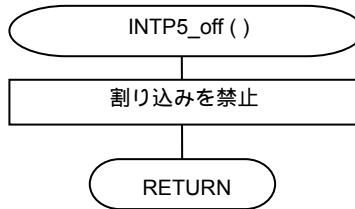
< INTP5 割り込み許可 >

図 4-40 INTP5割り込み許可処理



< INTP5 割り込み禁止 >

図 4-41 INTP5割り込み禁止処理



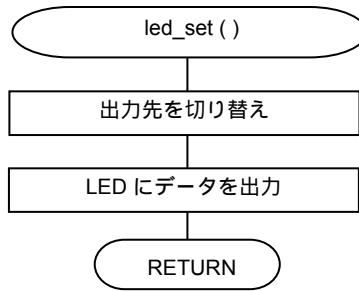
< LED に 1 ~ 4 桁の値を表示する >

図 4-42 LEDに1～4桁の値を表示する処理



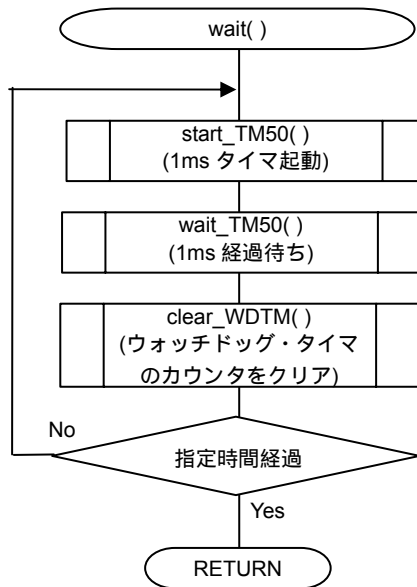
< 指定された LED に値を出力する >

図 4-43 指定されたLEDに値を出力する処理



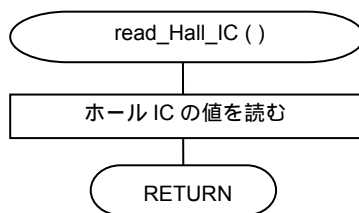
< 指定時間 (ms) 経過待ち >

図 4-44 指定時間 (ms) 経過待ち処理



< ホール IC の値を読み出す >

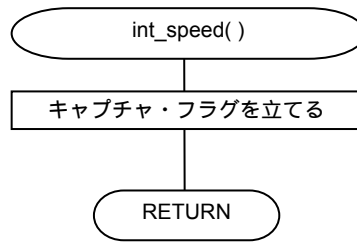
図 4-45 ホールICの値を読み出す処理





< INTP5 割り込み >

図 4-46 INTP5割り込み処理



## 4.8 制御プログラムの変数・定数一覧

## 4.8.1 main.hの#defineで定義している値

名称	意味	設定値	備考
FLG_ON	フラグ値	1	フラグの有効/無効を示します
FLG_OFF	フラグ値	0	
CW	回転方向	0	モータの回転方向を示します
CCW	回転方向	1	
START_TR	状態設定用	0x01	制御開始
STOP_TR	状態設定用	0x02	制御停止
FORWARD_TR	状態設定用	0x04	回転をCWに変更
REVERSE_TR	状態設定用	0x08	回転をCCWに変更
MODE_TR	状態設定用	0x10	MODEスイッチが押された状態
ERROR_HALL	エラー・フラグ・ビット	0x01	ホールICのエラー
ERROR_OC	エラー・フラグ・ビット	0x02	過電流によるエラー
ERROR_MOTOR	エラー・フラグ・ビット	0x04	モータ異常のエラー
ERROR_S_OC	エラー・フラグ・ビット	0x08	過電流によるエラー
P_OFF	通電パターン	0x3f	リアルタイム・ポートの通電
P_STOP	通電パターン	0x15	
P_T1	通電パターン	0x36	
P_T2	通電パターン	0x1e	
P_T3	通電パターン	0x1b	
P_T4	通電パターン	0x39	
P_T5	通電パターン	0x2d	
P_T6	通電パターン	0x27	

## 4.8.2 lib\_eu.hの#defineで定義している値

(1/2)

名称	意味	設定値	備考
KP_DEF	フィードバック・ゲイン	0.020	PID制御で使用
KI_DEF	フィードバック・ゲイン	0.001	
KD_DEF	フィードバック・ゲイン	0.003	
PWM_LIMIT_H	PWMの操作量上限	49.0	
PWM_LIMIT_L	PWMの操作量下限	-49.0	
UNIT_RPM	速度演算定数	2343750	速度算出に使用
PWM_BASE_REF	PWMベース	769	PWM出力で使用
PWM_F_REF	PWM初期値	0	
PWM_F_START	PWM開始値	60	
PWM_F_MIN	PWM最小値	10	
PWM_F_MAX	PWM最大値	769	
PWM_DTM_REF	デッド・タイム	0	
IN	ポート設定用	1	
OUT	ポート設定用	0	
CLEAR	レジスタ・ビット設定用	0	レジスタのビット・アクセスで使用
SET	レジスタ・ビット設定用	1	
INV_OFF	インバータ制御用	1/0	低電圧インバータ
INV_ON	インバータ制御用	0/1	
INVERTER_SW	インバータ制御ポート	P54	インバータ制御ポート
INVERTER_SW_MODE	インバータ制御ポート 制御レジスタ	PM54	インバータ制御ポート
INTP0	ポート制御レジスタ	PM00	過電流検知ポート
SW2	ポート制御レジスタ	PM73	START/STOP用ポート
SW3	ポート制御レジスタ	PM72	FORWARD用ポート
SW4	ポート制御レジスタ	PM71	REVERSE用ポート
SW5	ポート制御レジスタ	PM70	MODE用ポート
LD_LED0	ポート制御レジスタ	PM64	LED選択用ポート
LD_LED1	ポート制御レジスタ	PM65	
LD_LED2	ポート制御レジスタ	PM66	
LD_LED3	ポート制御レジスタ	PM67	
LD_DATA	ポート制御レジスタ	PM4	LEDへのデータ出力ポート

名称	意味	設定値	備考
LED_0	LED表示データ	0xc0	“0”を表示
LED_1		0xf9	“1”を表示
LED_2		0xa4	“2”を表示
LED_3		0xb0	“3”を表示
LED_4		0x99	“4”を表示
LED_5		0x92	“5”を表示
LED_6		0x82	“6”を表示
LED_7		0xf8	“7”を表示
LED_8		0x80	“8”を表示
LED_9		0x98	“9”を表示
LED_O		0xc0	“O”の代わりに“0”を表示
LED_I		0xcf	“I”を表示
LED_C		0xc6	“C”を表示
LED_H		0x89	“H”を表示
LED_A		0x88	“A”を表示
LED_L		0xc7	“L”を表示
LED_		0xff	“ ”を表示
LED_S		0x92	“S”を表示
LED_E		0x86	“E”を表示
LED_F		0x8e	“F”を表示
LED_P	0x8c	“P”を表示	
LED_dot	0x7f	“.”を表示	
IMS_DATA	メモリ・サイズ	0xc8	メモリサイズ切替設定に使用
WDTM_OFF	停止	0x77	ウォッチドッグ・タイマ制御用
WDTM_SET	設定	0x67	
WDTE_CLR	クリア	0xac	
WDTE_RESET	リセット	0x00	
KEY_WAIT	スイッチ観察時間	10	チャタリング除去で使用
SW	スイッチ	(P7&0xf)	スイッチ接続ポート
START_SW	起動	0x7	スイッチの状態
STOP_SW	停止	0x7	
FORWARD_SW	CW	0xb	
REVERSE_SW	CCW	0xd	
MODE_SW	モード	0xe	

## 4.8.3 変数

変数名	型	意味	備考
sys_flag	char	制御フラグ	制御状態
stop_wait	char	停止指令フラグ	停止指令の有無
cw_ccw_flag	char	回転方向指令フラグ	回転方向状態
cw_ccw_wait	char	反転停止指令フラグ	反転による停止指令の有無
old_hdata	char	ホールIC履歴	1つ前のホールICの値
speed_flag	char	速度情報フラグ	速度計算に必要なデータの有無
ad_flag	char	速度変更フラグ	指定速度変更機能を制限
int_cnt	unsigned int	割り込みカウンタ	キャリア同期割り込み回数
pid_cnt	unsigned int	PID制御タイミング	キャリア同期割り込み回数
print_cnt	unsigned int	速度表示タイミング	キャリア同期割り込み回数
cnt_data1 cnt_data2 cnt_data3	unsigned int	速度計算用データ	キャリア同期割り込み回数
pwm_base	int	PWMベース・クロック値	PWMのキャリア幅
pwm_ff	int	PWM指令値	PWMの流通率
old_ff	int	PWM指令値履歴	1つ前のpwm_ffの値
m_speed	int	実速度	モータの回転速度(rpm)
led_data[ ]	unsigned char	LED出力データ	LEDに表示する数値
up_flag	char	実速度計算状態フラグ	回転速度計算用
capture_flag	char	タイマ値取得フラグ	
clk_flag	char	タイマ値演算可能フラグ	
old_clk	unsigned int	前回のクロック値	
new_clk	unsigned int	最新のクロック値	
kp_ref	float	Pゲイン	速度制御
ki_ref	float	Iゲイン	
kd_ref	float	Dゲイン	
mvn	float	前回の操作量	
en	float	今回の差分	
en_1	float	前回の差分	
en_2	float	前々回の差分	

## 4.9 制御プログラムのソースファイル

メモリサイズ ROM: 16F0h

RAM: 3B8h

```

/*
  BLDC モータ 120 度通電方式 位置センサ(ホール IC)

  target : uPD78F0714 モータ・スタータ・キット

  date   : 2006/12/13
  filename: main.h

  NEC Micro Systems,Ltd
*/

#define FLG_ON      1
#define FLG_OFF    0

#define CW          0          /* 時計回り          */
#define CCW        1          /* 反時計周り        */
#define START_TR   0x01      /* 押されているスイッチ判定 */
#define STOP_TR    0x02
#define FORWARD_TR 0x04
#define REVERSE_TR 0x08
#define MODE_TR    0x10

#define ERROR_HALL 0x01      /* ホール IC 異常          */
#define ERROR_OC   0x02      /* 過電流                  */
#define ERROR_MOTOR 0x04     /* モータ異常              */
#define ERROR_S_OC 0x08      /* 過電流(ソフトウェア検出) */

#define P_OFF      0x3f      /* リアルタイム出力ポート OFF */
#define P_STOP    0x15
#define P_T1      0x36      /* 通電ボタン 1 U V          */
#define P_T2      0x1e      /* 通電ボタン 2 U W          */
#define P_T3      0x1b      /* 通電ボタン 3 V W          */
#define P_T4      0x39      /* 通電ボタン 4 V U          */
#define P_T5      0x2d      /* 通電ボタン 5 W U          */
#define P_T6      0x27      /* 通電ボタン 6 W V          */

extern char      sys_flag;
extern char      stop_wait;
extern char      cw_ccw_flag;
extern char      cw_ccw_wait;
extern char      old_hdata;
extern char      speed_flag;
extern char      ad_flag;
extern char      clk_flag;
extern char      capture_flag;
extern unsigned int new_clk;
extern unsigned int old_clk;
extern unsigned int int_cnt;
extern unsigned int pid_cnt;
extern unsigned int print_cnt;
extern unsigned int cnt_data;
extern unsigned int cnt_data1;
extern unsigned int cnt_data2;
extern unsigned int cnt_data3;
extern int        pwm_ff;
extern int        old_ff;
extern int        pwm_base;
extern int        m_speed;
extern unsigned int ad_data;

extern void      system_init(void);
extern void      get_speed(void);
extern void      speed_print(int);
extern unsigned char get_sw(void);
extern void      pwm_pid(int);
extern void      system_start(void);
extern void      system_stop(void);
extern void      system_restart(void);
extern char      read_Hall_IC(void);
extern void      set_RTPM01(unsigned char);
extern void      set_TW0(int, int, int, int);
extern void      print_error(char);
extern void      clear_WDTM(void);
extern void      reset_WDTM(void);

```

```

/*
  BLDC モータ 120 度通電方式 位置センサ(ホール IC)

  target : uPD78F0714 モータ・スタータ・キット

```

```

date : 2006/12/18
filename: main.c

NEC Micro Systems,Ltd
*/

#include "main.h"

/*
メイン関数
*/
void
main(void)
{
    unsigned char    sw = 0;
    unsigned char    tmp_sw = 0;

    system_init();          /* 初期設定          */
    while(1){
        clear_WDTM();      /* ウォッチドッグタイマクリア */
        get_speed();      /* 指定速度読み出し      */
        speed_print(200); /* 速度表示(ms 間隔)    */
        if(sys_flag != FLG_OFF){ /* 起動中              */
            if((cw_ccw_wait == FLG_OFF) && /* 反転の停止待ちでない */
               (stop_wait == FLG_OFF)){ /* 停止待ちでない      */
                sw = get_sw(); /* スイッチチェック    */
                switch(sw){
                    case STOP_TR:
                        if(sw != tmp_sw){ /* トグル対処          */
                            stop_wait = FLG_ON; /* 停止待ちを設定      */
                        }
                        break;

                    case FORWARD_TR:
                        if(cw_ccw_flag == CCW){ /* CCW で回転中        */
                            cw_ccw_wait = FLG_ON; /* 反転の停止待ちを設定 */
                            cw_ccw_flag = CW; /* 回転方向を設定      */
                        }
                        break;

                    case REVERSE_TR:
                        if(cw_ccw_flag == CW){ /* CW で回転中          */
                            cw_ccw_wait = FLG_ON; /* 反転停止待ちを設定   */
                            cw_ccw_flag = CCW; /* 回転方向を設定      */
                        }
                        break;

                    default:
                        ;
                }
            }
            pwm_pid(150); /* PID 制御 150ms 周期    */
        }else{ /* 停止中                */
            sw = get_sw(); /* スイッチチェック      */
            switch(sw){
                case START_TR:
                    if(sw != tmp_sw){ /* トグル対処          */
                        system_start(); /* 起動                */
                    }
                    break;

                case MODE_TR:
                    if(sw != tmp_sw){ /* トグル対処          */
                        if(ad_flag == FLG_ON){ /* 機能停止中          */
                            ad_flag = FLG_OFF; /* 機能復帰            */
                        }else{
                            ad_flag = FLG_ON; /* 機能停止            */
                        }
                    }
                    break;

                default:
                    ;
            }
        }
        tmp_sw = sw;
    }
}

```

```

/*
BLDC モータ 120 度通電方式 位置センサ(ホール IC)

target : uPD78F0714 モータ・スタータ・キット

date : 2006/12/13
filename: motor.c

NEC Micro Systems,Ltd
*/
#pragma sfr

```

```

#pragma INTERRUPT INTPO    int_faulta    rb1 /* 過電流発生 */
#pragma INTERRUPT INTTWOD  int_carrier    rb2 /* キャリア同期割り込み 75µs周期 */

/* ----- */
#include "main.h"

/* ----- */
#define INT_CNT_MAX    1333 /* 50ms 無回転の判定 x75us */
#define START_PWM_F_MAX 250 /* 無回転時の限界値 */

#define ISHUNT_AD      5 /* ISHUNT 端子が接続された AD チャンネル番号 */

#define ADDATA_MIN      0
#define ADDATA_100      0x1900
#define ADDATA_200      0x3200
#define ADDATA_300      0x4b00
#define ADDATA_400      0x6400
#define ADDATA_E_2      0x7400 /* 464<<6 */
#define ADDATA_500      0x7d00
#define ADDATA_600      0x9600
#define ADDATA_700      0xaf00
#define ADDATA_800      0xc800
#define ADDATA_900      0xe100
#define ADDATA_1000     0xfa00
#define ADDATA_MAX      0xfcc0

#define CLEAR          0
#define SET            1

/* ----- */
const unsigned char tr_sw[2][8] = { /* 通電ボタン */
    {P_OFF, P_T4, P_T6, P_T5, P_T2, P_T3, P_T1, P_OFF}, /* CW */
    {P_OFF, P_T1, P_T3, P_T2, P_T5, P_T6, P_T4, P_OFF} /* CCW */
};

/* ===== */
/*
ハードウェア過電流検知
*/
__interrupt void
int_faulta(void)
{
    system_stop(); /* システム停止 */
    print_error(ERROR_OC); /* エラー処理 */
}

/*
キャリア同期割り込み (谷側)
*/
__interrupt void
int_carrier(void)
{
    char h_data;
    int tmp_pwm;

    int_cnt++; /* 割り込み回数 */
    print_cnt++; /* 速度表示タイミング */
    if(int_cnt > INT_CNT_MAX){ /* 50ms 以上ホール IC に変化無し */
        if(stop_wait != FLG_OFF){ /* STOP での停止待ち */
            system_stop();
            return;
        }
        else if(cw_ccw_wait != FLG_OFF){ /* 反転で停止待ち */
            system_restart();
            return;
        }
        else if(pwm_ff > START_PWM_F_MAX){ /* START からの起動中 */
            system_stop();
            print_error(ERROR_MOTOR);
            return;
        }
        else{
            pwm_ff+=5; /* PWM を up */
            int_cnt = 0;
        }
    };

    pid_cnt++; /* PID 制御タイミング */
    h_data = read_Hall_IC();
    if((stop_wait == FLG_OFF) && (cw_ccw_wait == FLG_OFF)){ /* 停止待ちでない */
        if(h_data != old_hdata){ /* ホール IC が変化した */
            set_RTPM01(tr_sw[cw_ccw_flag][h_data]); /* 通電ボタンを設定 */
            old_hdata = h_data; /* ホール IC の値保存 */
            cnt_data3 = cnt_data2; /* 割り込み回数履歴保存 */
            cnt_data2 = cnt_data1; /* 割り込み回数履歴保存 */
            cnt_data1 = int_cnt; /* 割り込み回数保存 */
            cnt_data = cnt_data1 + cnt_data2 + cnt_data3;
            int_cnt = 0; /* 割り込み回数クリア */
            speed_flag = FLG_ON; /* 速度情報有効に設定 */
        };
        if(pwm_ff != old_ff){ /* デューティが変化した */
            tmp_pwm = pwm_base - pwm_ff;
            set_T2O(tmp_pwm, tmp_pwm, tmp_pwm, pwm_base); /* デューティ変更 */
            old_ff = pwm_ff;
        };
    };
};

```



```

    }else{
        /* 停止待ちでも速度表示を続けるための処理 */
        set_RTPM01(P_OFF);
        /* 通電を停止 */
        if(h_data != old_hdata){
            /* ホール IC が変化した */
            old_hdata = h_data;
            cnt_data3 = cnt_data2;
            cnt_data2 = cnt_data1;
            cnt_data1 = int_cnt;
            cnt_data = cnt_data1 + cnt_data2 + cnt_data3;
            int_cnt = 0;
            speed_flag = FLG_ON;
        }
    };

    if(capture_flag == FLG_ON){
        if(CR00 != new_clk){
            old_clk = new_clk;
            new_clk = CR00;
            clk_flag = FLG_ON;
            capture_flag = FLG_OFF;
        }
    };

    return;
}

```

```

/*
BLDC モータ 120 度通電方式 位置センサ(ホール IC)

target : uPD78F0714 モータ・スタータ・キット

date : 2006/12/13
filename: lib_eu.h

NEC Micro Systems,Ltd
*/

#define KP_DEF 0.020 /* For LowVoltage */
#define KI_DEF 0.001
#define KD_DEF 0.003
#define PWM_LIMIT_H 49.0 /* PWM の操作量の上限 */
#define PWM_LIMIT_L -49.0 /* PWM の操作量の下限 */

/*
m_speed 用定数: 400[rpm] * (0.15[s] / 12.8[us]) / 2
-----
種対数
TM00 カウントクロック 78.125[KHz](1/78.125[KHz])
400[rpm]時の周期(1/(400/60))
*/
#define UNIT_RPM 2343750 /* 2count */

#define MIN_SPEED 200
#define MAX_SPEED 3200

#define PWM_BASE_REF 769 /* PWM 周期の半周期 */
#define PWM_F_REF 0 /* PWM 波形の初期値 */
#define PWM_F_START 60 /* 始動時の初期値 */
#define PWM_F_MIN 10 /* 最小値 */
#define PWM_F_MAX 769 /* 最大値 */
#define PWM_DTM_REF 0 /* デッド・タイム */

#define IN 1 /* 入力 */
#define OUT 0 /* 出力 */

#define CLEAR 0
#define SET 1

#define INV_OFF 1 /* For LowVoltage */
#define INV_ON 0

#define INVERTER_SW P54 /* インバータ動作制御ポート */
#define INVERTER_SW_MODE PM54 /* インバータ動作制御ポート */
#define INTPO PM00 /* 過電流検知ポート */
#define SW2 PM73
#define SW3 PM72
#define SW4 PM71
#define SW5 PM70

#define LD_LED0 PM64
#define LD_LED1 PM65
#define LD_LED2 PM66
#define LD_LED3 PM67

#define LD_DATA PM4

#define LED_0 0xc0 /* LED 表示用データ */
#define LED_1 0xf9
#define LED_2 0xa4
#define LED_3 0xb0
#define LED_4 0x99
#define LED_5 0x92

```

```
#define LED_6      0x82
#define LED_7      0xf8
#define LED_8      0x80
#define LED_9      0x98
#define LED_O      0xc0      /* 0-C      */
#define LED_I      0xcf
#define LED_C      0xc6
#define LED_H      0x89      /* HALL      */
#define LED_A      0x88
#define LED_L      0xc7
#define LED_      0xff
#define LED_S      0x92      /* SELF      */
#define LED_E      0x86
#define LED_F      0x8e
#define LED_P      0x8c      /* PC      */
#define LED_dot    0x7f

#define IMS_DATA   0xc8

#define WDTM_OFF   0x77      /* ウォッチドッグ・タイマ制御用      */
#define WDTM_SET   0x67
#define WDTE_CLR   0xac
#define WDTE_RESET 0x00

#define KEY_WAIT   10      /* チャタリング除去 ms      */
#define SW         (P7&0xf) /* スwitchのポート      */

#define START_SW   0x7      /* 押されているスイッチ判定      */
#define STOP_SW    0x7
#define FORWARD_SW 0xb
#define REVERSE_SW 0xd
#define MODE_SW    0xe

static void      INTPO_on(void);
static void      led_set(unsigned char, unsigned char);
static void      led_print(int, char);
static void      wait(int);
static void      INTTWOUD_on(void);
static void      INTTWOUD_off(void);
static void      INTTWOUD3_on(void);
static void      INTTWOUD3_off(void);
static void      init_PORT(void);
static void      init_OSC(void);
static void      init_TWO(void);
static void      start_TWO(void);
static void      stop_TWO(void);
static void      init_TM00(void);
static void      start_TM00(void);
static void      stop_TM00(void);
static void      init_RTPM01(void);
static void      start_RTPM01(void);
static void      stop_RTPM01(void);
static void      init_AD(void);
static void      start_AD(void);
static unsigned int get_AD(char);
static void      init_WDTM(void);
static void      init_TM51(void);
static void      start_TM51(void);
static void      stop_TM51(void);
static void      init_TM50(void);
static void      start_TM50(void);
static void      wait_TM50(void);
static void      INTSR00_on(void);
static void      INTSR00_off(void);
static void      INTP5_on(void);
static void      INTP5_off(void);
```

```
/*
  BLDC モータ 120 度通電方式 位置センサ(ホール IC)

  target : uPD78F0714 モータ・スタータ・キット

  date : 2006/12/13
  filename: lib_eu.c

  NEC Micro Systems,Ltd
  */

#pragma sfr
#pragma stop
#pragma ei
#pragma di

#pragma INTERRUPT INTP5      int_speed      rb3

#include "main.h"
#include "lib_eu.h"

int      pwm_ff;      /* アクティブ幅:0 ~ 750      */
int      old_ff;      /* PWM 履歴      */
char      old_hdata; /* ホール IC データ履歴      */
int      pwm_base = PWM_BASE_REF; /* PWM の半周期:750 固定      */
```

```

char      start_flag;          /* 回転開始直後の判定 */
int       speed_ref = 200;     /* 指定速度 */
int       m_speed;            /* モータ回転速度 */
unsigned int int_cnt;         /* キャリア同期割り込み回数 */
unsigned int pid_cnt;         /* PID制御周期確認用 */
unsigned int print_cnt;       /* 速度表示間隔確認用 */
unsigned int cnt_data;        /*
unsigned int cnt_data1, cnt_data2, cnt_data3; /* キャリア同期割り込み回数履歴 */
char      cw_ccw_flag = CW;    /* 回転方向 */
char      speed_flag;          /* 速度情報の有無 */
char      cw_ccw_wait;        /* 反転の停止待ち */
char      sys_flag;           /* システム稼働状況 */
char      stop_wait;          /* 停止待ちの有無 */
char      ad_flag = FLG_OFF;   /* 指定速度変更機能制限 */

char      clk_flag;
char      capture_flag;
unsigned int new_clk, old_clk; /* 速度測定用タイマ値 */

/*
 * スタティック変数
 */
static const unsigned char led_data[10] = { /* 数字表示用 */
    LED_0, LED_1, LED_2, LED_3, LED_4,
    LED_5, LED_6, LED_7, LED_8, LED_9
};
static char up_flag = FLG_OFF; /* 速度更新確認フラグ */
static float kp_ref = KP_DEF; /* Kp 初期値 */
static float ki_ref = KI_DEF; /* Ki 初期値 */
static float kd_ref = KD_DEF; /* Kd 初期値 */
static float mvn; /* 操作量 */
static float en, en_1, en_2; /* 偏差 */

/*
 * 初期設定
 */
void system_init(void)
{
    init_PORT();
    init_OSC();
    init_WDTM();
    init_TWO();
    init_TMO0();
    init_RTPM01();
    init_AD();
    init_TMS0();
    led_set(0, LED_S);
    led_set(1, LED_E);
    led_set(2, LED_L);
    led_set(3, LED_F);
    wait(1000);
    start_AD(); /* AD変換開始 */
    INTPO_on(); /* 過電流割り込みマスク解除 */
    EI(); /* 割り込み許可 */
}

/*
 * システム起動
 */
void system_start(void)
{
    m_speed = 0;
    en_1 =
    en =
    mvn = 0.0;
    int_cnt =
    pid_cnt =
    print_cnt = 0;
    cnt_data1 =
    cnt_data2 =
    cnt_data3 = 500;
    cnt_data = 1500;
    sys_flag = FLG_ON;
    start_flag = FLG_ON;
    stop_wait = FLG_OFF;
    cw_ccw_wait = FLG_OFF;
    speed_flag = FLG_OFF;
    up_flag = FLG_OFF;
    clk_flag = FLG_OFF;
    capture_flag = FLG_OFF;
    pwm_ff = PWM_F_START;
    old_ff = 0;
    old_hdata = 0;
    start_TMO0();
    INTTWOUD_on(); /* キャリア同期割り込みマスク解除 */
    INTP5_on();
    start_RTPM01();
    start_TWO();
    INVERTER_SW = INV_ON; /* INVERTER enable */
}

```

```

/* 反転停止からの再起動
*/
void
system_restart(void)
{
    m_speed      = 0;
    mvn          =
    en_1        =
    en           = 0.0;
    pid_cnt      =
    print_cnt    = 0;
    cnt_data1    =
    cnt_data2    =
    cnt_data3    = 500;
    cnt_data     = 1500;
    start_flag   = FLG_ON;
    speed_flag   = FLG_OFF;
    up_flag      = FLG_OFF;
    clk_flag     = FLG_OFF;
    capture_flag = FLG_OFF;
    pwm_ff       = PWM_F_START;
    old_ff       = 0;
    old_hdata    = 0;
    cw_ccw_wait  = FLG_OFF;
}

/* システム停止
*/
void
system_stop(void)
{
    INVERTER_SW = INV_OFF;      /* INVERTER disable */
    INTTWOUD_off();
    INTPS_off();
    stop_RTPM01();
    stop_TWO();
    stop_TM00();
    sys_flag    = FLG_OFF;
    stop_wait   = FLG_OFF;
    cw_ccw_flag = CW;
    m_speed     = 0;          /* 速度 0 */
}

/* 過電流割り込み許可
*/
static void
INTPO_on(void)
{
    PROL.1 = 0;              /* 優先順位 */
    EGN.0  = 1;              /* 立ち下がリエッジ */
    IFOL.1 = CLEAR;         /* フラグクリア */
    MKOL.1 = CLEAR;         /* マスククリア */
}

/* エラー表示
*/
void
print_error(char eno)
{
    char data;

    switch(eno){
    case ERROR_HALL:
        led_set(0, LED_H);
        led_set(1, LED_A);
        led_set(2, LED_L);
        led_set(3, LED_L);
        break;

    case ERROR_OC:
        led_set(0, LED_);
        led_set(1, LED_O & LED_dot);
        led_set(2, LED_C & LED_dot);
        led_set(3, LED_);
        break;

    case ERROR_MOTOR:
        led_set(0, LED_F);
        led_set(1, LED_A);
        led_set(2, LED_L);
        led_set(3, LED_L);
        break;

    case ERROR_S_OC:
        led_set(0, LED_S & LED_dot);
        led_set(1, LED_);
        led_set(2, LED_O & LED_dot);
        led_set(3, LED_C & LED_dot);
        break;
}

```

```

default:
    data = read_Hall_IC();
    if((0 == data) || (7 == data)){
        led_set(0, LED_H);
        led_set(1, LED_A);
        led_set(2, LED_L);
        led_set(3, LED_L);
    }else{
        led_set(0, LED_F);
        led_set(1, LED_A);
        led_set(2, LED_L);
        led_set(3, LED_L);
    }
    }
    STOP();
}
}

/*
PID 制御
*/
void
pwm_pid(int cy_time)
{
    unsigned long tmp;
    unsigned int  new_tmp, old_tmp;
    int          pwm_tmp;

    if(speed_flag == FLG_ON){ /* 速度計測済み */
        speed_flag = FLG_OFF;
        if(start_flag == FLG_OFF){
            up_flag = FLG_ON; /* 速度更新済み */
        }else{
            start_flag = FLG_OFF; /* 先頭の速度情報は使用しない */
        }
    }

    if(pid_cnt >= (unsigned int)(cy_time*13)){ /* cy_time*1ms 経過 */
        pid_cnt = 0;
        if(up_flag == FLG_ON){ /* 速度更新あり */
            up_flag = FLG_OFF;
            if(clk_flag == FLG_ON){
                DI();
                clk_flag = FLG_OFF;
                new_tmp = new_clk;
                old_tmp = old_clk;
                EI();
                if(old_tmp > new_tmp){
                    tmp = (unsigned long)((65536 - (long)old_tmp) + (long)new_tmp);
                }else{
                    tmp = new_tmp - old_tmp;
                }
                m_speed = (int)(UNIT_RPM/tmp);
            }else{
                m_speed = (int)(200000/(unsigned long)cnt_data);
            }
        }
        if((sys_flag != FLG_OFF) && /* 起動中 */
           (stop_wait != FLG_ON) && /* 停止待ちでない */
           (cw_ccw_wait != FLG_ON)){ /* 反転の停止待ちでない */
            up_flag = FLG_OFF;
            en_2 = en_1;
            en_1 = en;
            en = (float)(speed_ref - m_speed);
            mvn = mvn +
                kp_ref*(en - en_1) +
                ki_ref*en +
                kd_ref*((en - en_1) - (en_1 - en_2));
            if(mvn > PWM_LIMIT_H){
                mvn = (float)PWM_LIMIT_H;
            }else if(mvn < PWM_LIMIT_L){
                mvn = (float)PWM_LIMIT_L;
            }
            pwm_tmp = pwm_ff + (int)mvn;
            if(pwm_tmp > PWM_F_MAX){ /* リミッタ */
                pwm_ff = PWM_F_MAX;
            }else if(pwm_tmp < PWM_F_MIN){
                pwm_ff = PWM_F_MIN;
            }else{
                pwm_ff = pwm_tmp;
            }
        }
    }
}

/*
速度指定可変抵抗の電圧を読み出し
指定速度に変換する
*/
void
get_speed(void)
{
    unsigned int data;
    unsigned long tmp;

```

```

if(ad_flag == FLG_OFF){
    data = get_AD(2);
    data = ((data - 3) * 12) + MIN_SPEED;
    if(data > MAX_SPEED){
        data = MAX_SPEED;
    }else if(data < MIN_SPEED) {
        data = MIN_SPEED;
    };
    tmp = (UNIT_RPM / (unsigned long)data);
    speed_ref = (int)(UNIT_RPM / tmp);
}
}

/*
スイッチ(P7)とUARTからの値読み出し

スイッチの値を読み出す
一定時間チェックして安定している値を返す

*/
unsigned char
get_sw(void)
{
    unsigned char data;

    int i;
    unsigned char tmp;
    static unsigned char start_stop_flag = FLG_OFF;

    data = SW;                                /* スイッチ読み込み */
    if(data != 0xf){
        for(i=0; i<KEY_WAIT;){                /* チャタリング除去 要調整 */
            tmp = SW;                          /* スイッチ再読み込み */
            if(data == tmp){
                i++;
            }else{
                i = 0;
                data = tmp;
            }
        }
        wait(2);
    }
    if(sys_flag != FLG_OFF){
        switch(data){
            case STOP_SW:
                if(start_stop_flag == FLG_OFF){
                    data = STOP_TR;
                    start_stop_flag = FLG_ON;
                }
                break;

            case FORWARD_SW: data = FORWARD_TR; break;
            case REVERSE_SW: data = REVERSE_TR; break;
            case MODE_SW: data = MODE_TR; break;
            default:
                ;
        }
    }else{
        if((data == START_SW) && (start_stop_flag == FLG_OFF)){
            data = START_TR;
            start_stop_flag = FLG_ON;
        }else if(data == MODE_SW){
            data = MODE_TR;
        }
    }
}
if(data == 0xf){ /* スイッチが何も押されていない */
    start_stop_flag = FLG_OFF; /* START/STOPのトグル防止 */
}
return(data);
}

/*
ボートの設定
*/
static void
init_PORT(void)
{
    INTP0 = IN;                                /* 過電流通知の割り込み */
    SW2 = IN;                                  /* START/STOP */
    SW3 = IN;                                  /* FORWARD */
    SW4 = IN;                                  /* REVERSE */
    SW5 = IN;                                  /* MODE */

    INVERTER_SW_MODE = OUT;                    /* INVERTER enable/disable */
    INVERTER_SW = INV_OFF;                     /* INVERTER disable */

    EGP5 = SET;                                /* 立ち上がりエッジ有効 */
    EGN5 = CLEAR;

    LD_LED0 = OUT;                             /* LED 選択 出力 */
    LD_LED1 = OUT;
    LD_LED2 = OUT;
    LD_LED3 = OUT;
}

```

```

LD_DATA = OUT;          /* LED表示データ 出力 */
}
/*
 * クロック切り替え
 */
static void
init_OSC(void)
{
    IMS = IMS_DATA;      /* メモリサイズ切り替え */
    clear_WDTM();
    while(OSTC != 0x1f); /* 発振安定待ち */
    PCC = 0x00;          /* 分周比設定 */
    MCM.0 = 1;          /* X1 入力クロック */
    VSWC.1 = 1;
}
/*
 * インバータタイマ
 */
static void
init_TWO(void)
{
    TCL02 = 0;          /* カウント・クロック:20MHz */
    TCL01 = 0;
    TCL00 = 0;
    IDEV02 = 0;        /* 毎回割り込み発生 */
    IDEV01 = 0;
    IDEV00 = 0;
    TWOM = 0;
    TWOTRGS = 0;
    TWOC = 0;
    TWOCM3 = PWM_BASE_REF;
    TWOCM0 = PWM_BASE_REF - PWM_F_REF;
    TWOCM1 = PWM_BASE_REF - PWM_F_REF;
    TWOCM2 = PWM_BASE_REF - PWM_F_REF;
    TWODTIME = PWM_DTM_REF; /* デッド・タイム */
    TWOBFCM3 = PWM_BASE_REF;
    TWOBFCM0 = PWM_BASE_REF - PWM_F_REF;
    TWOBFCM1 = PWM_BASE_REF - PWM_F_REF;
    TWOBFCM2 = PWM_BASE_REF - PWM_F_REF;
    TWOTRGS = 1;
}

void
set_TWO(int a, int b, int c, int base)
{
    TWOBFCM3 = (unsigned int)base;
    TWOBFCM0 = (unsigned int)a;
    TWOBFCM1 = (unsigned int)b;
    TWOBFCM2 = (unsigned int)c;
}

static void
start_TWO(void)
{
    TWOC.7 = SET;      /* タイマスタート */
}

static void
stop_TWO(void)
{
    TWOC.7 = CLEAR;   /* タイマストップ */
}
/*
 * 16ビットタイマ
 * CR01 リアルタイム出力ポート・トリガ
 */
static void
init_TM00(void)
{
    ES000 = 0;
    ES001 = 0;        /* T1000 端子の有効エッジ 立下りエッジ */
    CRC000 = 1;      /* CR00 キャプチャレジスタ */
    CRC001 = 1;      /* T1000 端子の有効エッジの逆相でキャプチャ */
    CRC002 = 0;      /* CR01 コンペアレジスタ */
    PRM001 = SET;    /* カウント・クロック 78.125kHz */
    PRM000 = CLEAR;
}

static void
start_TM00(void)
{
    TMIF00 = CLEAR;
    TMC00 = 0x04;    /* フリーランニングモード */
}

static void
stop_TM00(void)
{

```

```

    TMC00 = CLEAR;          /* タイマ停止          */
}

/*
リアルタイムポート
*/
static void
init_RTPM01(void)
{
    RTPM01 = 0x3f;          /* リアルタイム出力モード */
    RTPC01 = 0x20;          /* 6ビット x1チャンネル    */
    DCCTL01 = 0xc0;          /* PWM変調出力            */
    RTBL01 = 0x3f;          /* 出力バッファ           */
}

void
set_RTPM01(unsigned char data)
{
    RTBL01 = data;          /* 通電ボタン設定        */
    CRO1 = TMO0 + 1;
}

static void
start_RTPM01(void)
{
    RTPC01.7 = SET;          /* 動作許可              */
}

static void
stop_RTPM01(void)
{
    RTPC01.7 = CLEAR;          /* 動作禁止              */
}

/*
AD
*/
static void
init_AD(void)
{
    ADS = 4;                /* SPEED                  */
    ADM = 0x04;              /* 3.6us                  */
}

static void
start_AD(void)
{
    ADIF = CLEAR;           /* 割り込み通知フラグクリア */
    ADCS = SET;             /* 変換動作許可            */
    while(ADIF != SET);     /* 割り込み通知待ち        */
}

static unsigned int
get_AD(char s_bit)
{
    return((((unsigned int)ADCR)>>(s_bit+6))&0x3ff);
}

/*
ウォッチドッグタイマ
*/
static void
init_WDTM(void)
{
    clear_WDTM();
    WDTM = WDTM_SET;
}

void
clear_WDTM(void)
{
    WDTE = WDTE_CLR;
}

void
reset_WDTM(void)
{
    WDTE = WDTE_RESET;
}

/*
8ビット・タイマ50
*/
static void
init_TM50(void)
{
    TCL50 = 0x06;           /* 1ms                    */
    CR50 = 78;
}

static void
start_TM50(void)
{
    TMIF50 = CLEAR;          /* 割り込み通知フラグクリア */
}

```



```

TCE50 = SET;          /* タイマ開始          */
}

static void
wait_TM50(void)
{
    while(TMIF50 != SET);      /* 割り込み通知待ち      */
    TCE50 = CLEAR;           /* タイマ停止            */
}

/*
 * 速度表示
 */
void
speed_print(int ms)
{
    if((MODE_SW == SW) || (sys_flag == FLG_OFF)){
        led_print(speed_ref, ad_flag);
    }else{
        if(print_cnt > (unsigned int)(ms*13)){
            led_print(m_speed, FLG_OFF);
            print_cnt = 0;
        }
    }
}

/*
 * キャリア同期割り込み許可
 */
static void
INTTWOUD_on(void)
{
    UDIFW0 = CLEAR;          /* 要求フラグクリア */
    UDMKW0 = CLEAR;         /* 割り込み許可      */
}

/*
 * キャリア同期割り込み禁止
 */
static void
INTTWOUD_off(void)
{
    UDMKW0 = SET;           /* 割り込み禁止 */
    UDIFW0 = CLEAR;
}

/*
 * INTP5 割り込み許可
 */
static void
INTP5_on(void)
{
    PIF5 = CLEAR;
    PMK5 = CLEAR;
}

/*
 * INTP5 割り込み禁止
 */
static void
INTP5_off(void)
{
    PMK5 = SET;
}

/*
 * 7 セグ LED に値を表示する
 */
static void
led_print(int data, char flag)
{
    unsigned int tmp;
    int flg = FLG_OFF;

    tmp = (unsigned int)(data / 1000);
    if(tmp != 0){          /* 最上位が0でない */
        flg = FLG_ON;     /* 最上位に数値を表示した */
        led_set(0, led_data[tmp]);
    }else{
        led_set(0, LED_);
    }
    data %= 1000;
    tmp = (unsigned int)(data / 100);
    if((tmp != 0) || (flg == FLG_ON)){ /* 値が0でない or すでに数字を表示した */
        flg = FLG_ON;
        led_set(1, led_data[tmp]);
    }else{
        led_set(1, LED_);
    }
    data %= 100;
    tmp = (unsigned int)(data / 10);
    if((tmp != 0) || (flg == FLG_ON)){ /* 値が0でない or すでに数字を表示した */
        led_set(2, led_data[tmp]);
    }else{
}

```

```
        led_set(2, LED_);
    }
    data %= 10;
    if(flag == FLAG_OFF){
        led_set(3, led_data[data]);
    }else{
        led_set(3, (unsigned char)(led_data[data]&LED_dot);
    }
}

/*
7セグLEDにデータを表示する
no: 表示するLEDの場所
data: 出力するデータ
*/
static void
led_set(unsigned char no, unsigned char data)
{
    unsigned char p;

    P6 = 0x00;
    P4 = data;

    switch(no){
        case 0: p = 0x80; break; /* 表示する場所 */ /* 左端 */ /*
        case 1: p = 0x40; break;
        case 2: p = 0x20; break;
        default: p = 0x10; break; /* 右端 */ /*
    }
    P6 = p;
}

/*
時間調整 ms
*/
static void
wait(int cnt)
{
    int i;

    for(i=0; i<cnt; i++){
        start_TM50(); /* タイマスタート */ /*
        wait_TM50(); /* 割り込み要求発生待ち */ /*
        clear_WDTM(); /* ウォッチドック・タイマのクリア */ /*
    }
    return;
}

/*
ホールICの読み出し
*/
char
read_Hall_IC(void)
{
    return((char)((P0>>1)&0x7));
}

/*
INTP5
*/
__interrupt void
int_speed(void)
{
    capture_flag = FLAG_ON;
}
}
```

# 付 録

別途提供するモータ操作パネル（GUI プログラム）と、モータ・コントロール I/O ボード上の RS-232C 端子を接続して、このシステムをコントロールする場合のプログラム例を添付します。

```
/*
BLDC モータ 120 度通電方式 位置センサ(ホール IC)

target : uPD78F0714 モータ・スタータ・キット
        コンパイラオプション GUI : GUI 使用

date   : 2006/12/13
filename: lib_eu.h

  NEC Micro Systems,Ltd
*/

#define KP_DEF          0.020      /* For LowVoltage */
#define KI_DEF          0.001
#define KD_DEF          0.003
#define PWM_LLIMIT_H   49.0       /* PWM の操作量の上限 */
#define PWM_LLIMIT_L   -49.0      /* PWM の操作量の下限 */

/*
   m_speed 用定数: 400[rpm] * (0.15[s] / 12.8[us]) / 2
                   -----
                   極対数
                   TM00 カウントクロック 78.125[KHz](1/78.125[KHz])
                   400[rpm]時の周期(1/(400/60))
*/
#define UNIT_RPM        2343750 /* 2count */

#define MIN_SPEED       200
#define MAX_SPEED       3200

#define PWM_BASE_REF    769       /* PWM 周期の半周期 */
#define PWM_F_REF       0         /* PWM 波形の初期値 */
#define PWM_F_START     60        /* 始動時の初期値 */
#define PWM_F_MIN       10        /* 最小値 */
#define PWM_F_MAX       769       /* 最大値 */
#define PWM_DTM_REF     0         /* デッド・タイム */

#define IN               1         /* 入力 */
#define OUT              0         /* 出力 */

#define CLEAR           0
#define SET              1

#define INV_OFF         1         /* For LowVoltage */
#define INV_ON          0

#define INVERTER_SW     P54        /* インバータ動作制御ポート */
#define INVERTER_SW_MODE PM54     /* インバータ動作制御ポート */
#define INTPO           PM00       /* 過電流検知ポート */
#define SW2             PM73
#define SW3             PM72
#define SW4             PM71
#define SW5             PM70

#define LD_LED0         PM64
#define LD_LED1         PM65
#define LD_LED2         PM66
#define LD_LED3         PM67

#define LD_DATA         PM4

#define LED_0           0xc0       /* LED 表示用データ */
#define LED_1           0xf9
#define LED_2           0xa4
#define LED_3           0xb0
#define LED_4           0x99
#define LED_5           0x92
#define LED_6           0x82
#define LED_7           0xf8
#define LED_8           0x80
#define LED_9           0x98
#define LED_0           0xc0       /* 0-C */
#define LED_I           0xcf
#define LED_C           0xc6
#define LED_H           0x89       /* HALL */
#define LED_A           0x88
#define LED_L           0xc7
```

```

#define LED_          0xff
#define LED_S        0x92      /* SELF */
#define LED_E        0x86
#define LED_F        0x8e
#define LED_P        0x8c      /* PC */
#define LED_dot      0x7f

#define IMS_DATA      0xc8

#define WDTM_OFF      0x77      /* ウォッチドッグ・タイマ制御用 */
#define WDTM_SET      0x67
#define WDTE_CLR      0xac
#define WDTE_RESET    0x00

#define KEY_WAIT      10       /* チャタリング除去 ms */
#define SW             (P7&0xf) /* スイッチのポート */

#define START_SW      0x7       /* 押されているスイッチ判定 */
#define STOP_SW       0x7
#define FORWARD_SW    0xb
#define REVERSE_SW    0xd
#define MODE_SW       0xe

#ifdef GUI
#define MD_ERROR_HALL  0xf0      /* ホール IC 異常 */
#define MD_ERROR_OC    0xf1      /* 過電流 */
#define MD_ERROR_MOTOR 0xf2      /* モータ異常 */
#define RTS            P11
#define CTS            P10
#define RX_BUFF_SIZE   6        /* UART00 の受信バッファサイズ */
#define MD_CMD_START   0x20      /* START */
#define MD_CMD_STOP    0x21      /* STOP */
#define MD_CMD_RESET   0x2f      /* RESET */
#define MD_CMD_GETID   0x10      /* ID 要求 */
#define MD_CMD_GETVER  0x11      /* バージョン要求 */
#define MD_CMD_SETSSPEED 0x30     /* 指定速度変更 */
#define MD_CMD_GETSSPEED 0x31     /* 指定速度読み出し */
#define MD_CMD_SETPIDPARAM 0x40    /* PID 変更 */
#define MD_CMD_GETPIDPARAM 0x41    /* PID 読み出し */
#define MD_CMD_GETSPEED 0x50     /* 実速度読み出し */
#define MY_ID          0x21      /* ファーム識別 ID */
#define VER_MAJOR      0x01      /* バージョン情報 */
#define VER_MINOR      0x00
#define VER_SEQ        0x01
#endif

static void INTPO_on(void);
static void led_set(unsigned char, unsigned char);
static void led_print(int, char);
static void wait(int);
static void INTTWOUD_on(void);
static void INTTWOUD_off(void);
static void INTTWOCM3_on(void);
static void INTTWOCM3_off(void);
static void init_PORT(void);
static void init_OSC(void);
static void init_TWO(void);
static void start_TWO(void);
static void stop_TWO(void);
static void init_TM00(void);
static void start_TM00(void);
static void stop_TM00(void);
static void init_RTPM01(void);
static void start_RTPM01(void);
static void stop_RTPM01(void);
static void init_AD(void);
static void start_AD(void);
static unsigned int get_AD(char);
static void init_WDTM(void);
static void init_TMS1(void);
static void start_TMS1(void);
static void stop_TMS1(void);
static void init_TM50(void);
static void start_TM50(void);
static void wait_TM50(void);
static void INTSRO0_on(void);
static void INTSRO0_off(void);
static void INTP5_on(void);
static void INTP5_off(void);

#ifdef GUI
static void init_UART00(void);
static unsigned char get_UART00(void);
static unsigned char wait_UART00(void);
static void set_UART00(unsigned char);
#endif
#endif

```

```

/*
BLDC モータ 120 度通電方式 位置センサ(ホール IC)

target : uPD78F0714 モータ・スタータ・キット
        コンパイラオプション GUI : GUI 使用

date   : 2006/12/13
filename: lib_eu.c

NEC Micro Systems,Ltd
*/

#pragma sfr
#pragma stop
#pragma ei
#pragma di

#pragma INTERRUPT INTP5          int_speed  rb3

#ifdef GUI
#pragma INTERRUPT INTSR00       int_uart00 rb1 /* UART00 コマンド受信用 */
#endif

#include "main.h"
#include "lib_eu.h"

int          pwm_ff;                /* アクティブ幅 : 0 - 750 */
int          old_ff;                /* PWM 履歴 */
char         old_hdata;             /* ホール IC データ履歴 */
int          pwm_base = PWM_BASE_REF; /* PWM の半周期 : 750 固定 */
char         start_flag;            /* 回転開始直後の判定 */
int          speed_ref = 200;        /* 指定速度 */
int          m_speed;               /* モータ回転速度 */
unsigned int int_cnt;               /* キャリア同期割り込み回数 */
unsigned int pid_cnt;               /* PID 制御同期確認用 */
unsigned int print_cnt;             /* 速度表示間隔確認用 */
unsigned int cnt_data;
unsigned int cnt_data1, cnt_data2, cnt_data3; /* キャリア同期割り込み回数履歴 */
char         cw_ccw_flag = CW;      /* 回転方向 */
char         speed_flag;            /* 速度情報の有無 */
char         cw_ccw_wait;           /* 反転の停止待ち */
char         sys_flag;              /* システム稼働状況 */
char         stop_wait;             /* 停止待ちの有無 */
char         ad_flag = FLG_OFF;     /* 指定速度変更機能制限 */

char         clk_flag;
char         capture_flag;
unsigned int new_clk, old_clk;       /* 速度測定用タイマ値 */

/*
スタティック変数
*/
static const unsigned char led_data[10] = { /* 数字表示用 */
    LED_0, LED_1, LED_2, LED_3, LED_4,
    LED_5, LED_6, LED_7, LED_8, LED_9
};
static char up_flag = FLG_OFF;        /* 速度更新確認フラグ */
static float kp_ref = KP_DEF;        /* Kp 初期値 */
static float ki_ref = KI_DEF;        /* Ki 初期値 */
static float kd_ref = KD_DEF;        /* Kd 初期値 */
static float mvn;                    /* 操作量 */
static float en, en_1, en_2;        /* 偏差 */
#if 0
static char i_check_count;
static float area_high, area_low;
#endif
#ifdef GUI
static unsigned char err_flag = FLG_OFF; /* エラー通知フラグ */
static unsigned char uart00_data[RX_BUFFER_SIZE]; /* UART00 受信バッファ */
static unsigned char read_p, write_p; /* バッファポインタ */
static int kp_tmp, ki_tmp, kd_tmp; /* Kp,Ki,Kd の通信データ */
#endif

/*
初期設定
*/
void system_init(void)
{
    init_PORT();
    init_OSC();
    init_WDTM();
    init_TMO();
    init_TMOO();
    init_RTPMO1();
    init_AD();
    init_TMSO();
#ifdef GUI
    init_UART00();
    led_set(0, LED_P);
    led_set(1, LED_C);
    kp_tmp = (int)(kp_ref * 10000);

```

```

ki_tmp = (int)(ki_ref * 10000);
kd_tmp = (int)(kd_ref * 10000);
#else
led_set(0, LED_S);
led_set(1, LED_E);
led_set(2, LED_L);
led_set(3, LED_F);
wait(1000);
#endif
start_AD();          /* AD 変換開始          */
INTP0_on();         /* 過電流割り込みマスク解除 */
EI();              /* 割り込み許可          */
}

/* システム起動 */
void
system_start(void)
{
    m_speed      = 0;
    en_1         =
    en           =
    mvn          = 0.0;
    int_cnt      =
    pid_cnt      =
    print_cnt    = 0;
    cnt_data1    =
    cnt_data2    =
    cnt_data3    = 500;
    cnt_data     = 1500;
    sys_flag     = FLG_ON;
    start_flag   = FLG_ON;
    stop_wait    = FLG_OFF;
    cw_ccw_wait  = FLG_OFF;
    speed_flag   = FLG_OFF;
    up_flag      = FLG_OFF;
    clk_flag     = FLG_OFF;
    capture_flag = FLG_OFF;
    pwm_ff       = PWM_F_START;
    old_ff       = 0;
    old_hdata    = 0;
    start_TMOO();
    INTTWOUD_on(); /* キャリア同期割り込みマスク解除 */
    INTP5_on();
    start_RTPM01();
    start_TWO();
    INVERTER_SW = INV_ON; /* INVERTER enable */
}

/* 反転停止からの再起動 */
void
system_restart(void)
{
    m_speed      = 0;
    mvn          =
    en_1         =
    en           = 0.0;
    pid_cnt      =
    print_cnt    = 0;
    cnt_data1    =
    cnt_data2    =
    cnt_data3    = 500;
    cnt_data     = 1500;
    start_flag   = FLG_ON;
    speed_flag   = FLG_OFF;
    up_flag      = FLG_OFF;
    clk_flag     = FLG_OFF;
    capture_flag = FLG_OFF;
    pwm_ff       = PWM_F_START;
    old_ff       = 0;
    old_hdata    = 0;
    cw_ccw_wait  = FLG_OFF;
}

/* システム停止 */
void
system_stop(void)
{
    INVERTER_SW = INV_OFF; /* INVERTER disable */
    INTTWOUD_off();
    INTP5_off();
    stop_RTPM01();
    stop_TWO();
    stop_TMOO();
    sys_flag     = FLG_OFF;
    stop_wait    = FLG_OFF;
    cw_ccw_flag  = CW;
    m_speed      = 0; /* 速度 0 */
}

```

```

/*
  過電流割り込み許可
*/
static void
INTPO_on(void)
{
  PROL.1 = 0;           /* 優先順位 */
  EGN.0 = 1;           /* 立ち下がりエッジ */
  IFOL.1 = CLEAR;      /* フラグクリア */
  MKOL.1 = CLEAR;      /* マスククリア */
}

/*
  エラー表示
*/
void
print_error(char eno)
{
  char data;

  switch(eno){
  case ERROR_HALL:
#ifdef GUI
    err_flag = MD_ERROR_HALL;
#endif
    led_set(0, LED_H);
    led_set(1, LED_A);
    led_set(2, LED_L);
    led_set(3, LED_L);
    break;

  case ERROR_OC:
#ifdef GUI
    err_flag = MD_ERROR_OC;
#endif
    led_set(0, LED_);
    led_set(1, LED_0 & LED_dot);
    led_set(2, LED_C & LED_dot);
    led_set(3, LED_);
    break;

  case ERROR_MOTOR:
#ifdef GUI
    err_flag = MD_ERROR_MOTOR;
#endif
    led_set(0, LED_F);
    led_set(1, LED_A);
    led_set(2, LED_I);
    led_set(3, LED_L);
    break;

  case ERROR_S_OC:
#ifdef GUI
    err_flag = MD_ERROR_OC;
#endif
    led_set(0, LED_S & LED_dot);
    led_set(1, LED_);
    led_set(2, LED_0 & LED_dot);
    led_set(3, LED_C & LED_dot);
    break;

  default:
    data = read_Hall_IC();
    if((0 == data) || (7 == data)){
#ifdef GUI
      err_flag = MD_ERROR_HALL;
#endif
      led_set(0, LED_H);
      led_set(1, LED_A);
      led_set(2, LED_L);
      led_set(3, LED_L);
    }else{
#ifdef GUI
      err_flag = MD_ERROR_MOTOR;
#endif
      led_set(0, LED_F);
      led_set(1, LED_A);
      led_set(2, LED_I);
      led_set(3, LED_L);
    }
  }
#ifdef GUI
  STOP();
#endif
}

/*
  PID制御
*/
void
pwm_pid(int cy_time)
{
  unsigned long tmp;

```

```

unsigned int new_tmp, old_tmp;
int pwm_tmp;

if(speed_flag == FLG_ON){ /* 速度計測済み */
    speed_flag = FLG_OFF;
    if(start_flag == FLG_OFF){
        up_flag = FLG_ON; /* 速度更新済み */
    }else{
        start_flag = FLG_OFF; /* 先頭の速度情報は使用しない */
    }
}

if(pid_cnt >= (unsigned int)(cy_time*13)){ /* cy_time*1ms 経過 */
    pid_cnt = 0;
    if(up_flag == FLG_ON){ /* 速度更新あり */
        up_flag = FLG_OFF;
        if(clk_flag == FLG_ON){
            DI();
            clk_flag = FLG_OFF;
            new_tmp = new_clk;
            old_tmp = old_clk;
            EI();
            if(old_tmp > new_tmp){
                tmp = (unsigned long)((65536 - (long)old_tmp) + (long)new_tmp);
            }else{
                tmp = new_tmp - old_tmp;
            }
            m_speed = (int)(UNIT_RPM/tmp);
        }else{
            m_speed = (int)(200000/(unsigned long)cnt_data);
        }
        if((sys_flag != FLG_OFF) && /* 起動中 */
            (stop_wait != FLG_ON) && /* 停止待ちでない */
            (cw_ccw_wait != FLG_ON)){ /* 反転の停止待ちでない */
            up_flag = FLG_OFF;
            en_2 = en_1;
            en_1 = en;
            en = (float)(speed_ref - m_speed);
            mvn = mvn +
                kp_ref*(en - en_1) +
                ki_ref*en +
                kd_ref*((en - en_1) - (en_1 - en_2));
            if(mvn > PWM_LIMIT_H){
                mvn = (float)PWM_LIMIT_H;
            }else if(mvn < PWM_LIMIT_L){
                mvn = (float)PWM_LIMIT_L;
            }
            pwm_tmp = pwm_ff + (int)mvn;
            if(pwm_tmp > PWM_F_MAX){ /* リミット */
                pwm_ff = PWM_F_MAX;
            }else if(pwm_tmp < PWM_F_MIN){
                pwm_ff = PWM_F_MIN;
            }else{
                pwm_ff = pwm_tmp;
            }
        }
    }
}

/*
速度指定可変抵抗の電圧を読み出し
指定速度に変換する
*/
void
get_speed(void)
{
#ifdef GUI
    unsigned int data;
    unsigned long tmp;

    if(ad_flag == FLG_OFF){
        data = get_AD(2);
        data = ((data - 3) * 12) + MIN_SPEED;
        if(data > MAX_SPEED){
            data = MAX_SPEED;
        }else if(data < MIN_SPEED) {
            data = MIN_SPEED;
        }
        tmp = (UNIT_RPM / (unsigned long)data);
        speed_ref = (int)(UNIT_RPM / tmp);
    }
#endif
}

/*
スイッチ(P7)とUARTからの値読み出し

スイッチの値を読み出す
一定時間チェックして安定している値を返す
*/
unsigned char

```



```

get_sw(void)
{
    unsigned char data;

#ifdef GUI
    int i;
    unsigned char tmp;
    static unsigned char start_stop_flag = FLG_OFF;

    data = SW;                                /* スイッチ読み込み */
    if(data != 0xf){
        for(i=0;i<KEY_WAIT;){                /* チャタリング除去 要調整 */
            tmp = SW;                          /* スイッチ再読み込み */
            if(data == tmp){
                i++;
            }else{
                i = 0;
                data = tmp;
            }
            wait(2);
        }
        if(sys_flag != FLG_OFF){
            switch(data){
                case STOP_SW:
                    if(start_stop_flag == FLG_OFF){
                        data = STOP_TR;
                        start_stop_flag = FLG_ON;
                    }
                    break;

                case FORWARD_SW: data = FORWARD_TR; break;
                case REVERSE_SW: data = REVERSE_TR; break;
                case MODE_SW: data = MODE_TR; break;
                default:
                    ;
            }
        }else{
            if((data == START_SW) && (start_stop_flag == FLG_OFF)){
                data = START_TR;
                start_stop_flag = FLG_ON;
            }else if(data == MODE_SW){
                data = MODE_TR;
            }
        }
    }
    if(data == 0xf){ /* スイッチが何も押されていない */
        start_stop_flag = FLG_OFF; /* START/STOP のトグル防止 */
    }
#else
    int ss;
    unsigned char hi, lo;

    data = get_UART00();
    switch(data){
        case MD_CMD_GETID: /* Get ID */
            set_UART00(data);
            set_UART00(MY_ID);
            break;

        case MD_CMD_GETVER: /* Get Version */
            set_UART00(data);
            set_UART00(VER_MAJOR);
            set_UART00(VER_MINOR);
            set_UART00(VER_SEQ);
            break;

        case MD_CMD_START: /* Start */
            set_UART00(data);
            data = START_TR;
            break;

        case MD_CMD_STOP: /* Stop */
            set_UART00(data);
            data = STOP_TR;
            break;

        case MD_CMD_RESET: /* Reset */
            set_UART00(data);
            while(STIF00 != SET);
            reset_WDTM();
            break;

        case MD_CMD_SETSSPEED: /* Set Setting Speed */
            lo = wait_UART00();
            hi = wait_UART00();
            set_UART00(data);
            ss = ((int)hi<<8) + lo;
            if(hi > 0x80){ /* CCW */
                ss = -ss + 1;
                data = REVERSE_TR;
                if(sys_flag == FLG_OFF){
                    cw_ccw_flag = CCW;
                }
            }
        }else{

```

```

        data = FORWARD_TR;
        if(sys_flag == FLG_OFF){
            cw_ccw_flag = CW;
        }
    }
    speed_ref = ss;
    break;

case MD_CMD_GETSSPEED:          /* Get Setting Speed */
    set_UART00(data);
    ss = speed_ref;
    if(cw_ccw_flag == CCW){
        ss = ~ss + 1;
    }
    lo = (unsigned char)((unsigned int)(ss)&0xff);
    hi = (unsigned char)(((unsigned int)(ss)>>8)&0xff);
    set_UART00(lo);
    set_UART00(hi);
    break;

case MD_CMD_SETPIDPARAM:       /* PIDのフィードバックゲイン設定 */
    lo = wait_UART00();
    hi = wait_UART00();
    kp_tmp = (((int)(hi)<<8) + lo);
    lo = wait_UART00();
    hi = wait_UART00();
    ki_tmp = (((int)(hi)<<8) + lo);
    lo = wait_UART00();
    hi = wait_UART00();
    kd_tmp = (((int)(hi)<<8) + lo);
    set_UART00(data);
    kp_ref = ((float)kp_tmp)/10000;
    ki_ref = ((float)ki_tmp)/10000;
    kd_ref = ((float)kd_tmp)/10000;
    break;

case MD_CMD_GETPIDPARAM:       /* PIDのフィードバックゲイン読み出し */
    set_UART00(data);
    kp_tmp = ((int)kp_ref)*10000;
    set_UART00((unsigned char)((unsigned int)(kp_tmp)&0xff));
    set_UART00((unsigned char)(((unsigned int)(kp_tmp)>>8)&0xff));
    ki_tmp = ((int)ki_ref)*10000;
    set_UART00((unsigned char)((unsigned int)(ki_tmp)&0xff));
    set_UART00((unsigned char)(((unsigned int)(ki_tmp)>>8)&0xff));
    kd_tmp = ((int)kd_ref)*10000;
    set_UART00((unsigned char)((unsigned int)(kd_tmp)&0xff));
    set_UART00((unsigned char)(((unsigned int)(kd_tmp)>>8)&0xff));
    break;

case MD_CMD_GETSPEED:          /* 回転速度読み出し */
    if(err_flag != FLG_OFF){
        set_UART00(err_flag);
        err_flag = FLG_OFF;
    }else{
        set_UART00(data);
        if(m_speed == 0){
            set_UART00(0);
            set_UART00(0);
        }else{
            ss = m_speed;
            if(cw_ccw_wait == FLG_OFF){
                if(cw_ccw_flag == CCW){
                    ss = ~ss + 1;
                }
            }else{ /* 反転停止待ち */
                if(cw_ccw_flag == CW){
                    ss = ~ss + 1;
                }
            }
            set_UART00((unsigned char)((unsigned int)(ss)&0xff));
            set_UART00((unsigned char)(((unsigned int)(ss)>>8)&0xff));
        }
    }
    break;

default:
    ;
}
#endif
return(data);
}

/*
ボートの設定
*/
static void
init_PORT(void)
{
    INTPO = IN;          /* 過電流通知の割り込み */
    SW2 = IN;           /* START/STOP */
    SW3 = IN;           /* FORWARD */
    SW4 = IN;           /* REVERSE */
    SW5 = IN;           /* MODE */
}

```

```

INVERTER_SW_MODE = OUT;          /* INVERTER enable/disable */
INVERTER_SW      = INV_OFF;      /* INVERTER disable */

EGP5 = SET;                       /* 立ち上がりエッジ有効 */
EGN5 = CLEAR;

LD_LED0 = OUT;                    /* LED 選択 出力 */
LD_LED1 = OUT;
LD_LED2 = OUT;
LD_LED3 = OUT;

LD_DATA = OUT;                    /* LED 表示データ 出力 */
}

/*
 クロック切り替え
*/
static void
init_OSC(void)
{
    IMS = IMS_DATA;                /* メモリサイズ切り替え */
    clear_WDTM();
    while(OSTC != 0x1f);           /* 発振安定待ち */
    PCC = 0x00;                    /* 分周比設定 */
    MCM.0 = 1;                     /* X1 入力クロック */
    VSWC.1 = 1;
}

/*
 インバータタイマ
*/
static void
init_TWO(void)
{
    TCL02 = 0;                     /* カウント・クロック:20MHz */
    TCL01 = 0;
    TCL00 = 0;
    IDEV02 = 0;                    /* 毎回割り込み発生 */
    IDEV01 = 0;
    IDEV00 = 0;
    TWOM = 0;
    TWOTRGS = 0;
    TWOOC = 0;
    TWOCM3 = PWM_BASE_REF;
    TWOCM0 = PWM_BASE_REF - PWM_F_REF;
    TWOCM1 = PWM_BASE_REF - PWM_F_REF;
    TWOCM2 = PWM_BASE_REF - PWM_F_REF;
    TWODTIME = PWM_DTM_REF;        /* デッド・タイム */
    TWOBFCM3 = PWM_BASE_REF;
    TWOBFCM0 = PWM_BASE_REF - PWM_F_REF;
    TWOBFCM1 = PWM_BASE_REF - PWM_F_REF;
    TWOBFCM2 = PWM_BASE_REF - PWM_F_REF;
    TWOTRGS = 1;
}

void
set_TWO(int a, int b, int c, int base)
{
    TWOBFCM3 = (unsigned int)base;
    TWOBFCM0 = (unsigned int)a;
    TWOBFCM1 = (unsigned int)b;
    TWOBFCM2 = (unsigned int)c;
}

static void
start_TWO(void)
{
    TWOC.7 = SET;                  /* タイマスタート */
}

static void
stop_TWO(void)
{
    TWOC.7 = CLEAR;               /* タイマストップ */
}

/*
 16 ビットタイマ
 CR01 リアルタイム出力ポート・トリガ
*/
static void
init_TM00(void)
{
    ES000 = 0;
    ES001 = 0;                    /* T1000 端子の有効エッジ 立下りエッジ */
    CRC000 = 1;                   /* CR00 キャプチャレジスタ */
    CRC001 = 1;                   /* T1000 端子の有効エッジの逆相でキャプチャ */
    CRC002 = 0;                   /* CR01 コンペアレジスタ */
    PRM001 = SET;                  /* カウント・クロック 78.125kHz */
    PRM000 = CLEAR;
}

static void

```

```

start_TM00(void)
{
    TMIF00 = CLEAR;
    TMC00 = 0x04;          /* フリーランニングモード */
}

static void
stop_TM00(void)
{
    TMC00 = CLEAR;        /* タイマ停止 */
}

/*
リアルタイムポート
*/
static void
init_RTPM01(void)
{
    RTPM01 = 0x3f;        /* リアルタイム出力モード */
    RTPC01 = 0x20;        /* 6ビットx1チャンネル */
    DCCTL01 = 0xc0;       /* PWM 変調出力 */
    RTBL01 = 0x3f;        /* 出力バッファ */
}

void
set_RTPM01(unsigned char data)
{
    RTBL01 = data;        /* 通電パタン設定 */
    CR01 = TM00 + 1;
}

static void
start_RTPM01(void)
{
    RTPC01.7 = SET;       /* 動作許可 */
}

static void
stop_RTPM01(void)
{
    RTPC01.7 = CLEAR;     /* 動作禁止 */
}

/*
AD
*/
static void
init_AD(void)
{
    ADS = 4;              /* SPEED */
    ADM = 0x04;           /* 3.6us */
}

static void
start_AD(void)
{
    ADIF = CLEAR;        /* 割り込み通知フラグクリア */
    ADCS = SET;          /* 変換動作許可 */
    while(ADIF != SET);  /* 割り込み通知待ち */
}

static unsigned int
get_AD(char s_bit)
{
    return((((unsigned int)ADCR)>>(s_bit+6))&0x3ff);
}

/*
ウォッチドッグタイマ
*/
static void
init_WDTM(void)
{
    clear_WDTM();
    WDTM = WDTM_SET;
}

void
clear_WDTM(void)
{
    WDTE = WDTE_CLR;
}

void
reset_WDTM(void)
{
    WDTE = WDTE_RESET;
}

/*
8ビット・タイマ50
*/
static void
init_TM50(void)

```

```

{
    TCL50 = 0x06;
    CR50 = 78;          /* 1ms */
}

static void
start_TM50(void)
{
    TMIF50 = CLEAR;    /* 割り込み通知フラグクリア */
    TCE50 = SET;      /* タイマ開始 */
}

static void
wait_TM50(void)
{
    while(TMIF50 != SET); /* 割り込み通知待ち */
    TCE50 = CLEAR;      /* タイマ停止 */
}

/*
 * 速度表示
 */
void
speed_print(int ms)
{
    if((MODE_SW == SW) || (sys_flag == FLG_OFF)){
        led_print(speed_ref, ad_flag);
    }else{
        if(print_cnt > (unsigned int)(ms*13)){
            led_print(m_speed, FLG_OFF);
            print_cnt = 0;
        }
    }
}

/*
 * キャリア同期割り込み許可
 */
static void
INTTWOUD_on(void)
{
    UDIFW0 = CLEAR;    /* 要求フラグクリア */
    UDMKW0 = CLEAR;   /* 割り込み許可 */
}

/*
 * キャリア同期割り込み禁止
 */
static void
INTTWOUD_off(void)
{
    UDMKW0 = SET;      /* 割り込み禁止 */
    UDIFW0 = CLEAR;
}

/*
 * INTP5 割り込み許可
 */
static void
INTP5_on(void)
{
    PIF5 = CLEAR;
    PMK5 = CLEAR;
}

/*
 * INTP5 割り込み禁止
 */
static void
INTP5_off(void)
{
    PMK5 = SET;
}

/*
 * 7セグLEDに値を表示する
 */
static void
led_print(int data, char flag)
{
    unsigned int tmp;
    int fig = FLG_OFF;

    tmp = (unsigned int)(data / 1000);
    if(tmp != 0){
        fig = FLG_ON; /* 最上位が0でない */
        led_set(0, led_data[tmp]); /* 最上位に数値を表示した */
    }else{
        led_set(0, LED_);
    }
    data %= 1000;
    tmp = (unsigned int)(data / 100);
    if((tmp != 0) || (fig == FLG_ON)){ /* 値が0でない or すでに数字を表示した */
        fig = FLG_ON;
    }
}

```

```

        led_set(1, led_data[tmp]);
    }else{
        led_set(1, LED_);
    }
    data %= 100;
    tmp = (unsigned int)(data / 10);
    if((tmp != 0) || (fig == FLG_ON)){ /* 値が0でない or すでに数字を表示した */
        led_set(2, led_data[tmp]);
    }else{
        led_set(2, LED_);
    }
    data %= 10;
    if(flag == FLG_OFF){
        led_set(3, led_data[data]);
    }else{
        led_set(3, (unsigned char)(led_data[data])&LED_dot);
    }
}

/*
7セグLEDにデータを表示する
no: 表示するLEDの場所
data: 出力するデータ
*/
static void
led_set(unsigned char no, unsigned char data)
{
    unsigned char p;

    P6 = 0x00;
    P4 = data;
    switch(no){
        case 0: p = 0x80; break; /* 表示する場所 */ /* 左端 */ /*
        case 1: p = 0x40; break;
        case 2: p = 0x20; break;
        default: p = 0x10; break; /* 右端 */ /*
    }
    P6 = p;
}

/*
時間調整 ms
*/
static void
wait(int cnt)
{
    int i;

    for(i=0;i<cnt;i++){
        start_TMS0(); /* タイマスタート */ /*
        wait_TMS0(); /* 割り込み要求発生待ち */ /*
        clear_WDTM(); /* ウォッチドック・タイマのクリア */ /*
    }
    return;
}

/*
ホールICの読み出し
*/
char
read_Hall_IC(void)
{
    return((char)((P0>>1)&0x7));
}

/*
INTP5
*/
__interrupt void
int_speed(void)
{
    capture_flag = FLG_ON;
}

#ifdef GUI
/*
UART00の設定
*/
static void
init_UART00(void)
{
    PM10 = IN;
    PM11 = OUT;
    PM13 = IN;
    PM14 = OUT;
    RTS = 1;
    P14 = 1;
    BRGC00 = 0x56; /* 115200 */
    PS001 = CLEAR;
    PS000 = CLEAR;
    CLOO = SET; /* 8bit */
    SLOO = CLEAR;
    POWER00 = SET;
    TXE00 = SET;
}

```

```
RXE00 = SET;
STIF00 = SET;
SRIF00 = CLEAR;
SRMK00 = CLEAR;          /* 受信割り込み許可 */
RTS = 0;
read_p = 0;
write_p = 0;
}

/*
UART00 の受信バッファデータを返す
*/
static unsigned char
get_UART00(void)
{
    unsigned char data = 0;

    if(read_p != write_p){          /* 受信データ有り */
        data = uart00_data[read_p++]; /* データを取り出す */
        read_p %= 6;                /* 読み出しポインタを更新 */
    }
    return(data);
}

/*
UART00 の受信待ち
*/
static unsigned char
wait_UART00(void)
{
    unsigned char data;

    while(read_p == write_p);      /* バッファにデータが入るまで待つ */
    data = uart00_data[read_p++];  /* データを取り出す */
    read_p %= 6;                   /* 読み出しポインタを更新 */
    return(data);
}

/*
UART00 から送信
*/
static void
set_UART00(unsigned char data)
{
    while(STIF00 != SET);          /* 送信完了待ち */
    STIF00 = CLEAR;
    TXS00 = data;                 /* 送信 */
}

/*
UART00 のコマンド受信用
*/
__interrupt void
int_uart00(void)
{
    unsigned char tmp;

    tmp = ASIS00;                 /* エラー・ステータス読み出し */
    uart00_data[write_p++] = RXB00; /* 受信バッファに格納 */
    write_p %= RX_BUFF_SIZE;      /* 書き込みポインタ更新 */
}
#endif
```

## 【発 行】

NECエレクトロニクス株式会社

〒211-8668 神奈川県川崎市中原区下沼部1753

電話（代表）：044(435)5111

---

お問い合わせ先

## 【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。

URL(アドレス) <http://www.necel.co.jp/>

## 【営業関係，技術関係お問い合わせ先】

半導体ホットライン

(電話：午前 9:00～12:00，午後 1:00～5:00)

電 話 : 044-435-9494

E-mail : [info@necel.com](mailto:info@necel.com)

## 【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。