

# Training Text

マイコントレーニングボード  
MT-R300



DCモーター基板 (MT-E508) 編

VPort Lab. 代表  
元長野県工科短期大学教授  
工学博士 星野 俊行 著




**Sunhayato**

# 安全上のご注意





このたびは弊社製品をご使用いただき、誠にありがとうございます。本項では、誤った取り扱いによる事故を未然に防ぐための安全上の注意事項を説明しています。弊社製品をご使用になる前に必ずお読みください。

 <b>警告</b>	この表記を無視して誤った取り扱いをすると、死亡や重傷など、人体への重大な障害をもたらす恐れのある内容について示しています。
 <b>注意</b>	この表記を無視して誤った取り扱いをすると、軽傷または中程度の障害をもたらす恐れのある内容について示しています。また、本製品や本製品に接続している機器に損傷を与える可能性がある事項についても示しています。




## 警告

 <b>水場禁止</b>	▶水分の多いところ、水がかかる場所では、本製品は使用しないでください 風呂場や台所など水分の多いところ、水がかかる場所では、本製品は使用しないでください。火災、感電、故障の原因となります。
 <b>禁止</b>	▶医療、軍事、航空宇宙、列車、運送、原子力などの制御設備へは使用しないでください 医療機器、軍事機器、航空宇宙機器、運送、原子力などの制御設備などの人命に関わるシステムへの使用は意図しておりません。
 <b>接触禁止</b>	▶雷が鳴りはじめたらご使用をお控えください 近くに雷が発生したときは、パソコンから本製品を抜いてご使用をお控えください。また、ご使用のパソコンの取扱説明書に従って、電源プラグをコンセントから抜くなどしてください。雷によっては、火災、感電、故障の原因となることがあります。

## 注意

 <b>プラグの差し込み</b>	▶プラグは確実に差し込んでください 差し込みが不完全ですと火災、感電、過熱、故障の原因になります。
 <b>発火注意</b>	▶発火、発煙、異臭への対処 発火、発煙、異臭がするなどの異常がありましたら使用を直ちに中止してください。そのまま使用すると、火災、故障の原因となります。 すぐにパソコンから抜き、煙などの異常が出なくなるのを確認し、販売店などに修理をご依頼ください。
 <b>分解禁止</b>	▶分解・改造しないでください 分解、改造しないでください。怪我、感電、故障の原因となります。本製品の分解、改造による怪我や事故について、当社は責任を負いかねます。
 <b>接触禁止</b>	▶濡れた手での操作は避けてください 濡れた手で電源ケーブル・プラグを抜き差ししないでください。また、製品に触れないでください。感電の原因となることがあります。

# 注意

 禁止	<p>▶ 以下のような場所では使用しないでください 本製品を以下のような場所で使用すると、動作不良、故障の原因となります。</p> <ul style="list-style-type: none"><li>・ 振動や衝撃が加わる場所</li><li>・ 直射日光のあたる場所</li><li>・ 湿気やホコリが多い場所</li><li>・ 温度差の激しい場所</li><li>・ 熱を発生するもの（暖房器具など）の近く</li><li>・ 強い磁力、電波が発生するもの（磁石、ディスプレイ、スピーカー、ラジオ、無線機など）の近く</li><li>・ 湿気の多い場所</li></ul>
 子供注意	<p>▶ 子供の手の届かない場所に置いてください 本製品に装着されている電子部品など子供が飲み込まないように注意してください。</p>
 引抜禁止	<p>▶ 通信中はケーブルを引抜かないでください 本製品がパソコンと通信中の場合はケーブルを引抜かないでください。ケーブルの引抜きは、必ず通信していないときに行ってください。故障の原因になることがあります。</p>
 安全設計	<p>▶ 安全設計をしてください 本製品を、高度な信頼性を必要とするシステムに使用する場合は、冗長設計、誤動作防止設計など十分な安全設計を必ず行ってください。本製品の故障、傷害により生じるいかなる損害、事故について当社は責任を負いかねます。</p>
 保管注意	<p>▶ 長期間使用しない場合の保管について 長期間使用しない場合は、帯電防止袋などに入れ、ホコリなどが入らないようにしてください。ホコリや汚れが付着すると短絡、接触不良などの原因になります。</p>
 ホコリ注意	<p>▶ 製品の清掃について 製品にホコリや汚れなどが付着すると短絡、故障の原因になりますので、下記の「▶ お手入れについて」に従って清掃してください。</p>
 薬品注意	<p>▶ お手入れについて ホコリが付着した場合はサンハヤト製ジェットブロー（JBK-480）などのガススプレーで吹き飛ばしてください。</p>
 使用注意	<p>▶ 故障、破損時の処理について 本製品が故障もしくは破損した場合は、速やかに使用を中止してください。そのまま使用しますと火災、感電、怪我の原因になるおそれがあります。</p>
 廃棄注意	<p>▶ 本製品の廃棄について 本製品の廃棄は、各自治体の廃棄ルールに従ってください。詳しくは各自治体にお問い合わせください。</p>

# 本資料についてのご注意

---

---

## 本資料について

- ・本資料は、電子工作や電子回路、パーソナルコンピュータの操作について一般的な知識をお持ちの方を対象にしています。
- ・本資料を元に操作するには、株式会社ルネサス エレクトロニクス製 H8/300H マイコンについての知識や開発環境などが必要です。
- ・Microsoft<sup>®</sup>、Windows<sup>®</sup>は米国 Microsoft 社の米国およびその他の国における登録商標です。
- ・その他、記載されている会社名、製品名は各社の商標または登録商標です。

## 本資料のご利用にあたって

- ・この取扱説明書に掲載している内容は、お客様が用途に応じた適切な製品をご購入頂くことを目的としています。その使用により当社及び第三者の知的財産権その他の権利に対する保証、又は実施権の許諾を意味するものではありません。また、権利の侵害に関して当社は責任を負いません。
- ・本資料に記載した情報を流用する場合は、お客様のシステム全体で充分評価し適用可能かご判断願います。当社では適用可能判断についての責任を負いません。
- ・本資料に記載してある内容は、一般的な電子機器（学習教材、事務機器、計測機器、パーソナル機器、コンピュータ機器など）に使用されることを目的としています。高い品質や信頼性が要求され、故障や誤作動が直接人命を脅かしたり人体に危害を及ぼす恐れのある、医療、軍事、航空宇宙、原子力制御、運輸、移動体、各種安全装置などの機器への使用は意図も保証もしていません。
- ・この取扱説明書の一部、又は全部を著者および当社の承諾なしで、いかなる形でも転載又は複製されることは堅くお断りします。
- ・全ての情報は本資料発行時点のものであり、当社は予告なしに本資料に記載した内容を変更することがあります。
- ・この資料の内容は慎重に制作しておりますが、万一記述誤りによってお客様に損害が生じても当社はその責任を負いません。
- ・本資料に関してのお問合せ、その他お気づきの点がございましたら、当社までお問合せください。
- ・本資料に関する最新の情報はサンハヤト株式会社ホームページ（<http://www.sunhayato.co.jp/>）に掲載しております。

# このテキストについて

本テキストでは、サンハヤト H8 マイコンレーニングボード MT-R300 のプログラム開発を、HEW (High-Performance Embedded Workshop) を使用して行った場合について説明しています。HEW の詳細な使用方法や注意事項につきましては、ルネサス エレクトロニクス社発行のマニュアルを参照してください。

## サンハヤト H8 マイコンレーニングボード MT-R300 概要

- ターゲットマイコン : H8/300H シリーズ 3062 グループ HD64H3062BF (以下 H8/3062BF マイコン)
- 書き込みボード : MT-R300 (H8/3062BF マイコン モード 5 (内蔵 ROM 有効拡張 16M バイトモード)、モード 7 (シングルチップアドバンスモード) 対応)

## 開発に必要なもの

- 開発用 PC (以下の条件を満たすもの)

PC 本体	Pentium III 600MHz 以上を搭載した IBM PC/AT 互換機
OS	Windows XP, Windows Me, Windows 98SE, Windows 2000 (Windows XP 推奨)
メモリ	128MB 以上 (ロードモジュールの 2 倍以上)
ハードディスク	エミュレータ用ソフトウェアのインストールには 100MB 以上の空き容量が必要
入力デバイス	マウス、またはマウス相当のポインティングデバイス
インターフェイス	1 ポート以上の USB インターフェイス

- サンハヤト H8 マイコンレーニングボード MT-R300
- 付属の USB ケーブル

## このマニュアルで使用しているツール類

このスタートアップガイドでは、以下の OS、ツールのバージョンで動作したものと説明しています。

- ホスト PC の OS : Windows XP Professional
- 統合化開発環境 : HEW V.4.03.00001
- C コンパイラ : H8S,H8/300 Standard Toolchain (V.6.02 Release00)
- フラッシュ書き込みツール : FDT (FlashDevelopmentToolKit) Ver4.02

HEW のバージョンは、HEW の「ヘルプ」メニューの「High-Performance Embedded Workshop のバージョン情報」で確認できます。

## このマニュアルで紹介している各社サイト、ツールについて

本スタートアップガイドで紹介している各社サイトやツールは、本スタートアップガイド発行時のものを掲載しております。各社サイトの構成やツールのバージョン、またバージョンに伴ってツールのファイル名などが変わっている場合がありますのでご了承ください。

## 目 次

<b>このテキストについて</b> .....	<b>5</b>
サンハヤト H8 マイコントレーニングボード MT-R300 概要 .....	5
開発に必要なもの .....	5
このマニュアルで使用しているツール類 .....	5
このマニュアルで紹介している各社サイト、ツールについて .....	5
<b>1. はじめに</b> .....	<b>7</b>
<b>2. 基礎知識</b> .....	<b>8</b>
2.1 DC モーター .....	8
2.2 DC モーターの制御 .....	9
2.3 DC モーター用ドライバー IC .....	10
2.4 フォトインタラプタ .....	11
<b>3 接続図の概要</b> .....	<b>12</b>
<b>4 基本実習</b> .....	<b>14</b>
4.1 基本プログラム .....	14
<b>実習 4.1.1</b> DIP_SW で制御信号 (BRK、ENB、PWM、CW) を設定し、DC モータをオンオフ制御する .....	15
<b>実習 4.1.2</b> DIP_SW (b6 ~ 0) でデューティ比 (0 ~ 100) を設定し、DC モータを PWM 制御する .....	16
<b>実習 4.1.3</b> DIP_SW でデューティ比(0 ~ 100)を設定し、DC モータを PWM 制御する(ITU の PWM モード) .....	17
4.2 DC モーター制御関数のライブラリー作成 .....	19
<b>実習 4.2.1</b> DIP_SW (b7) で DC モータの起動 / 停止を制御し、DIP_SW (b6 ~ 0) でデューティ比を調整する .....	19
<b>実習 4.2.2</b> DC モータを回転し、フォトインタラプタのカウント数を評価用 LED (8 ビット) に表示する .....	21
4.3 LCD への表示 .....	22
<b>実習 4.3.1</b> DC モータを回転し、LCD に回転数を表示する .....	23
<b>実習 4.3.2</b> DC モータを回転し、LCD に回転数と回転速度を表示する .....	24
<b>実習 4.3.3</b> DC モータを回転し、LCD に回転数と回転速度を表示する (タイマ割り込み関数使用) .....	26
<b>5 おわりに</b> .....	<b>29</b>

---

# 1. はじめに

---

DC モーターは直流電源で簡単に回すことができます。その電流の向きにより正逆転します。これは、起動トルクが大きく、出力効率が良いです。高速で電流の ON/OFF を繰り返す PWM 制御でトルクや回転数を制御できます。H ブリッジ型回路で正逆転の制御もできます。制御技術者はこの知識が必要不可欠です。

このボードは DC モーター、ドライバー IC、入力信号変換回路、モーターの回転を検知するフォトインタラプタ（フォトセンサ）、印加電圧調整用可変抵抗器などを設置しています。

このモーターは印加電圧が DC 1.0 ~ 5V、消費電流が 35 ~ 550mA です。印加電圧を 3.5V 以下にしているため、VPort with Power からの電流で十分動作します。入力信号変換（ゲート）回路とドライバー IC（H ブリッジ、PWM 制御回路を含む）で、モーターの回転方向切換、ブレーキ制御、動作イネーブル、PWM 制御ができます。

フォトインタラプタは、モーターに取り付けられた回転板による光の透過／遮断を検知します。マイコンのソフトウェアで、その信号からモーターの回転数を読み取ることができます。

可変抵抗器は、印加電圧を調整し、モーターの回転数を調整します。実験時には、擬似的な負荷変動としても使えます。

このボードを用いて、DC モーターの原理、ドライバー IC、ゲート回路などのハードウェアの知識を習得します。実習ではモーターの回転方向、回転速度（PWM 制御）などの制御プログラムを作成します。

モニター LED により、ドライバー IC の各制御信号、センサーの信号、をモニターできます。各信号の変化を視認できるため、より効果的な実習ができます。

## 2. 基礎知識

### 2.1 DC モーター

DC モーターは、①起動トルクが大きく、②回転数が印加電圧に比例し、③出力トルクが入力電流に比例し、④出力効率がよいため、制御用モーターとして優れています。

図 2-1 は DC モーターの構造を示します。

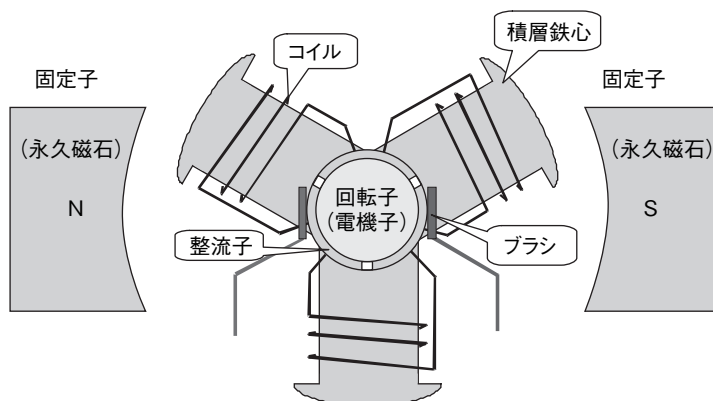


図 2-1 DC モーターの構造

DC モーターは外周部の固定子（永久磁石、ブラシ）と中心部の回転子（積層鉄心、コイル、整流子）で構成されています。外部電源からブラシ経由で整流子に電流を流します。コイルはそれぞれ整流子に接続されています。積層鉄心にはコイルが巻かれています。

図 2-2 は回転の動作原理を示します。

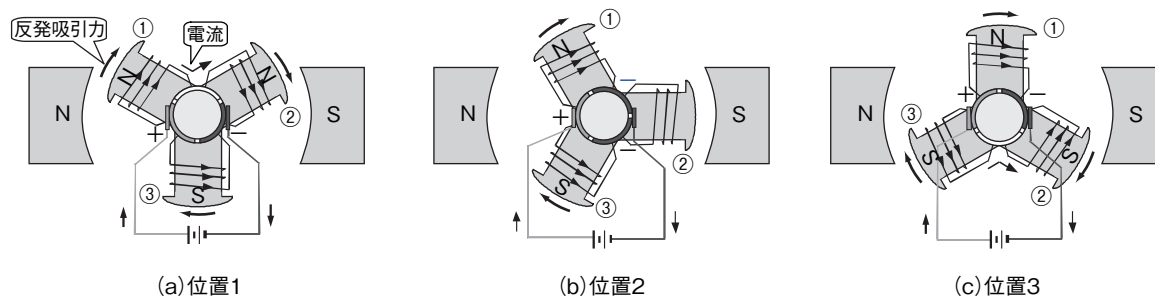


図 2-2 動作原理 (位置と励磁の向き)

(a) の位置 1 のとき、図のように 3 つのコイルに電流が流れます。その結果、積層鉄心③は S 極に磁化され、積層鉄心①と②は N 極に磁化されます。このとき、コイル①と②は直列に接続されるため、電流は③の半分になります。積層鉄心は固定子との間でそれぞれ反発・吸引力が生じ、回転子は時計方向に回転します。

(b) の位置 2 のとき、図のように 2 つのコイルに電流が流れます。その結果、積層鉄心①は N 極、③は S 極に磁化されます。②は電流が流れないため磁化されません。積層鉄心は固定子との間でそれぞれ反発・吸引力が生じ、回転子は時計方向に回転します。

(c) の位置 3 のとき、図のように 3 つのコイルに電流が流れます。その結果、鉄心①は N 極、積層鉄心②と③は S 極に磁化されます。積層鉄心は固定子との間でそれぞれ反発・吸引力が生じ、回転子は時計方向に回転します。

このように回転子の回転により整流子も回転し、コイルに流れる電流（積層鉄心の磁化）の向きが順次に切り替わり、回転子は回転を続けます。

このボードに使われているモーターは S.T.L.JAPAN 社の HS-D1S で、その仕様は次のとおりです。

- ・印加電圧 : DC1.0 ~ 5V
- ・消費電流 : 35 ~ 550mA
- ・定格トルク : 2gcm/3V
- ・回転数 : 15200rpm/3V

## 2.2 DC モーターの制御

### (1) 正・逆転などの制御

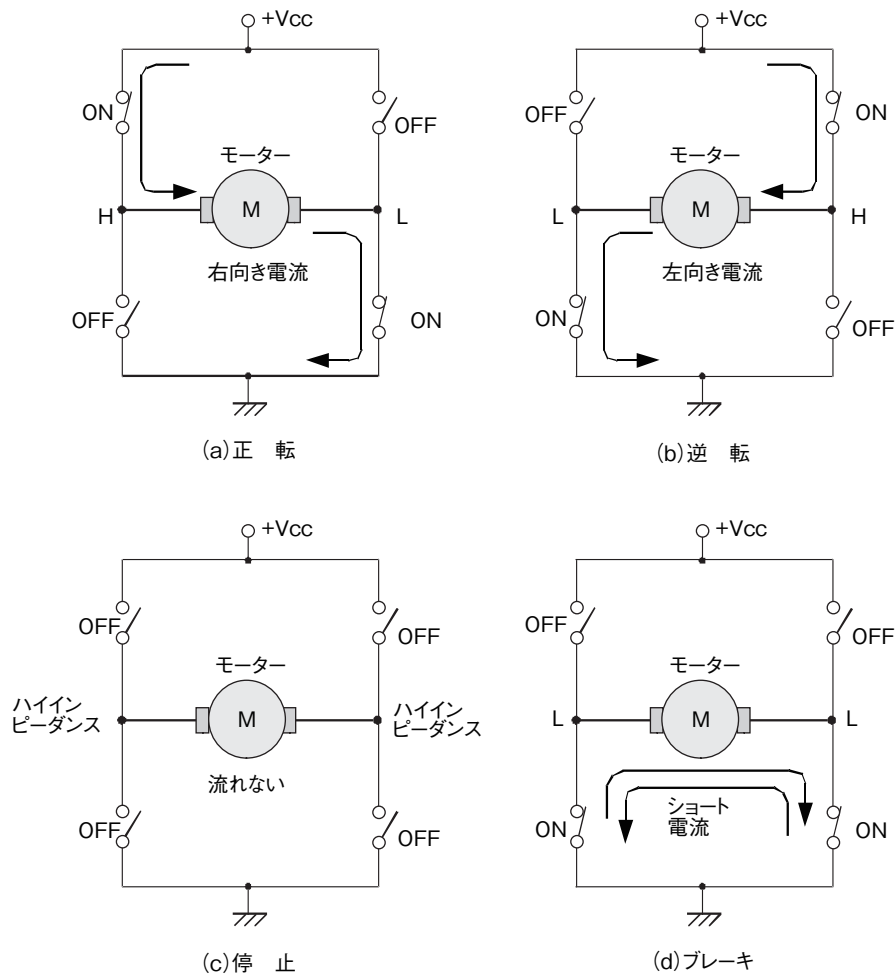


図 2-3 正・逆転の切り換え

図 2-3 は正転、逆転、停止、ブレーキの動作原理を示します。H 型のフルブリッジ回路の中心部に DC モーターを接続し、4 つのスイッチの切り換えで正転、逆転、停止、ブレーキの 4 モードを切り換えます。(a) の場合はモーターに右向き電流が流れ、モーターは特定の方向に回転（正転）します。(b) の場合はモーターに左向き電流が流れ、モーターは (a) とは逆の方向に回転（逆転）します。(c) の場合はモーターに電流が流れないためモーターは自然に停止します。(d) の場合は回転しているモーターに発生する逆起電力をショートさせて、コイルに流れる電流と磁界間の電磁誘導作用でブレーキをかけます。実際の回路では、機械的なスイッチではなく半導体スイッチ（トランジスタ、FET など）を用います。

## (2) 速度制御

DC モーターは、入力電流に比例した出力トルクが得られます。出力トルクが大きくなると回転速度も高くなります。従って入力電流を調整することで回転速度を制御できます。

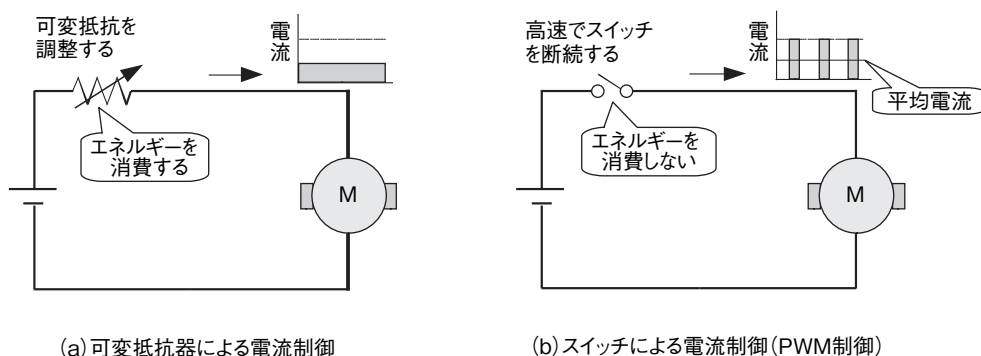


図 2-4 電流の制御

図 2-4 は DC モーターの電流制御の原理を示します。(a) の回路では、直列に可変抵抗器を挿入して電流を調整します。可変抵抗器にはモーターと同じ電流が流れるため、大きな電力をロスします。(b) の回路では、スイッチの断続により電流をオン/オフさせ、その平均電流を調整します。この方法は抵抗器を使わないため、電力をロスしません。

周期に対する通電電流の比をデューティ比といいます。これが 100% のときは電流が流れ続けるためもっとも高速で回転します。小さくするとゆっくり回転し、ある値以下になると停止します。これを PWM (Pulse Width Modulation: パルス幅変調) 制御といいます。PWM 信号の周波数は 100Hz ~ 1KHz が適切です。これが低すぎると回転が滑らかになりません。これが高すぎるとモーター (コイル) に流れる PWM 信号の波形が乱れ、電力のロスになります。実際の回路では、機械的なスイッチではなく半導体スイッチ (トランジスタ、FET など) を用います。

## 2.3 DC モーター用ドライバー IC

図 2-5 は DC モーター用ドライバー IC (東芝 :TA7291FG) のブロック図を示します。

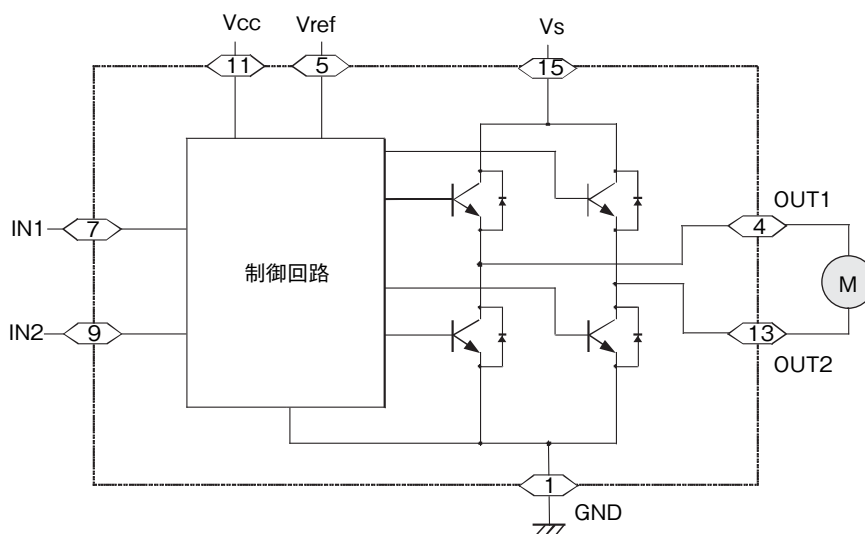


図 2-5 TA7291FG のブロック図

この内部にはパワートランジスタを用いたフルブリッジ回路が構成されています。出力端子 OUT1、OUT2 間にモーター

ターを接続します。入力端子 IN1、IN2 のロジック信号で、モーターの正転・逆転・停止・ブレーキの 4 モードを切り換えます。

+Vcc はロジック側電源端子で +5V を供給します。+Vs は出力側電源端子でモーター駆動用高電圧電源を供給します。+Vref は制御用電源端子でモータへの印加電圧を調整します。+Vref は +Vs より小さくします。

表 2-1 はこのドライバー IC の動作モード（真理値表）を示します。

表 2-1 ドライバー IC の動作モード

入力		出力		動作モード
IN1	IN2	OUT1	OUT2	
L	L	∞	∞	ストップ
H	L	H	L	CW/CCW
L	H	L	H	CCW/CW
H	H	L	L	ブレーキ

∞ : ハイインピーダンス

## 2.4 フォトインタラプタ

フォトインタラプタ（フォトセンサ）は、赤外線を発光する LED とその光を検出する光センサが対向しています。モーターに取り付けられた回転体がフォトインタラプタの光を遮断、通過することでモーターの回転を検知します。この回転体が 1/4 程度回転するたびに出力信号が変化するので、ステータス LED で確認しましょう。この信号をマイコンでカウントして、モーターの回転を求めます。フォトインタラプタの概要はステッピングモーター MT-E506 のテキストにありますので、そちらを参照願います。

## 3 接続図の概要

図 3-1 はこのボードの接続図の概要を示します。

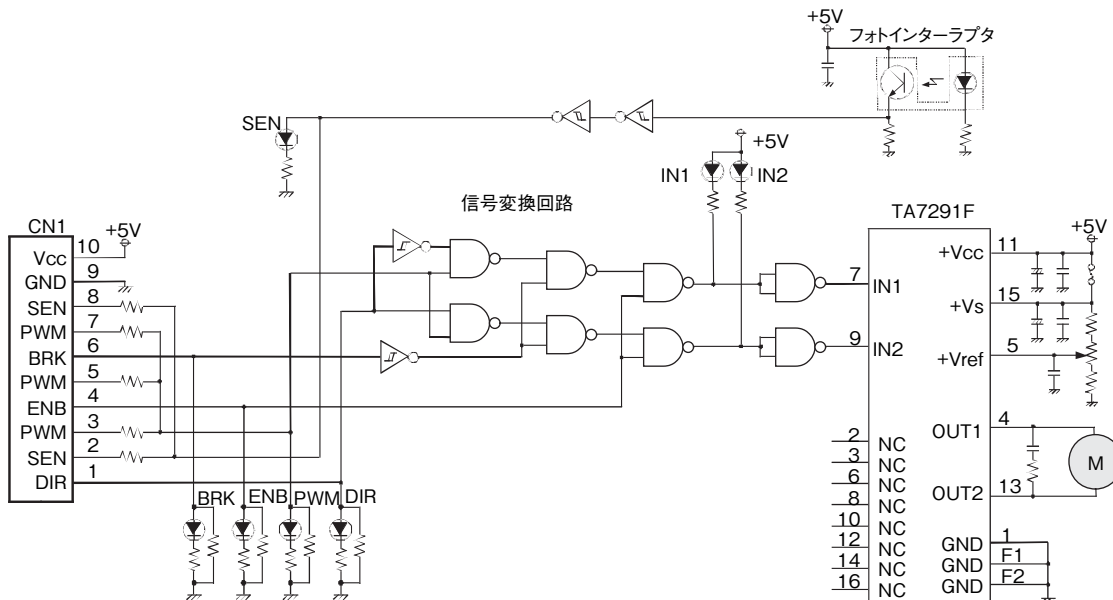


図 3-1 接続図の概要

ドライバー IC の論理回路用電源端子 +Vcc には +5V を供給します。

モーターの推奨電圧が 0 ~ +5V のため、同 IC のモーター用電源端子 +Vs にも +5V を供給します。

モーター制御用電源端子 +Vref には、実際にモーターに印加される電圧を入力します。ここでは可変抵抗器で分圧した 2.5 ~ 3.5V の電圧を供給します。

同図の中央付近の NAND ゲートなどで構成された信号変換回路を見てください。この回路で制御信号 (DIR、PWM、ENB、BRK) をドライバー IC の入力信号 (IN1、IN2) に変換します。ドライバー IC は、それから信号 (OUT1、OUT2) を出力し、モーターを制御します。

表 3-1 はモーターの制御信号と動作モードの関係を示します。

表 3-1 制御信号と動作モード

分類	入力				変換後 (IC 入力)		IC 出力		動作モード	
	BRK	ENB	PWM	DIR	IN1	IN2	OUT1	OUT2		
①	X	L	X	X	L	L	∞	∞	無効 (停止)	
②	H	H	X	X	H	H	L	L	ブレーキ	
③	L	H	H	L	H	L	H	L	時計方向	回転
			L		L	L	L	停止		
④	L	H	H	H	L	H	L	H	反時計方向	回転
			L		L	L	L	停止		

∞: ハイインピーダンス

X: H or L

分類①: ENB (イネーブル) 信号が L レベル (無効) の場合、他の 3 つの信号に関わらず OUT1、OUT2 がハイインピーダンス状態となり、モーターは停止します。

分類②: ENB (イネーブル) 信号が H レベル (制御信号有効)、BRK (ブレーキ) 信号が H レベル (ブレーキあり) の場合、OUT1、OUT2 が共に L レベルでその間が短絡状態になりモーターはブレーキがかかります。

分類③、④：ENB（イネーブル）信号が H レベル（制御信号有効）、BRK（ブレーキ）信号が L レベル（ブレーキなし）の場合です。③、④は、回転方向がそれぞれ時計方向、反時計方向となります。PWM 信号が H レベルの場合はモーターに電流が流れ、L レベルの場合は電流が流れません。従って、この端子に PWM 信号を加え、そのデューティ比を変化させることでモーターの回転速度を制御します。

この 4 つの信号とフォトインタラプタからの信号（H/L レベル）は、ステータス LED でモニターできます。接続図の詳細は MT-E508 取扱説明書をご参照ください。

## 4 基本実習

DC モーターの起動と停止、ブレーキ制御、回転方向の切り換え、回転速度の制御、などの実習を行います。それぞれの制御関数を作成し、それらをまとめたライブラリを作成します。必要なライブラリを用いて、LCD ボードにモーターの回転数、回転速度を表示します。

### 4.1 基本プログラム

ここでは DC モーターを制御する種々の基本プログラムを作成します。図 4-1 のとおり、付属の 10P ケーブルを用いて MT-R300 と MT-E508、MT-E501 を接続します。

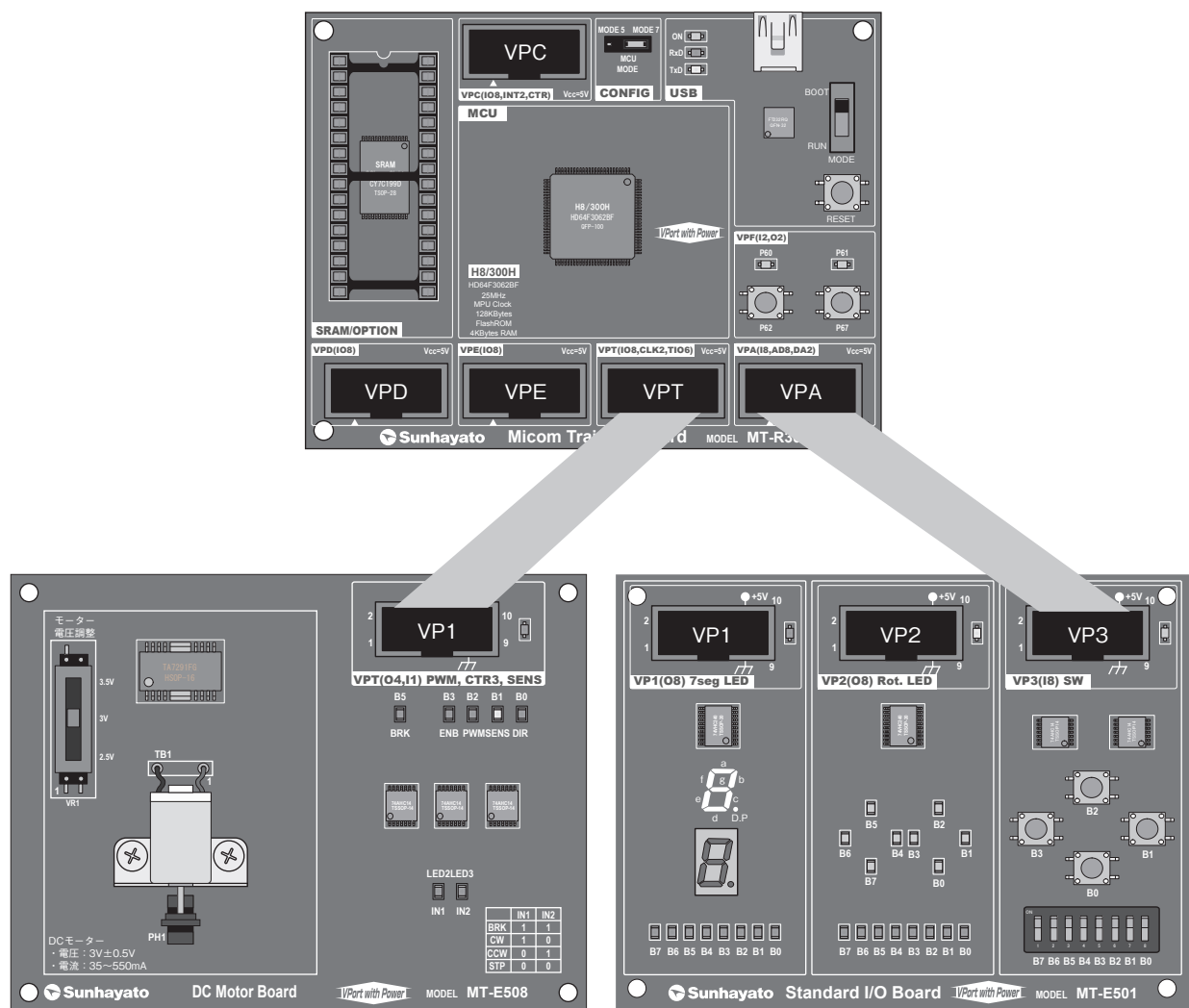


図 4-1 基板間の接続

"X:\¥xxxx¥workspace" に、ワークスペース、プロジェクト (dcm) を新規作成します。他の実習と同様に種々の設定をします。"X:\¥xxxx¥workspace¥@library" フォルダ内の "mt\_r300.c" をプロジェクトに追加します。相対パスで、① "Project directory"、② "custom directory" で "..¥..¥..¥@library" を指定します。

**実習 4.1.1** DIP\_SW で制御信号 (BRK、ENB、PWM、CW) を設定し、DC モータをオンオフ制御する

**[ 解説 ]**

図 4-2 は VPT ポートのピン割り付けを示します。

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
NC	ADIE	BRK	NC	ENB	PWM	SENS	DIR
(入力)	(入力)	(出力)	(入力)	(出力)	(出力)	(入力)	(出力)
		ブレーキ信号		イネーブル信号	PWM 信号	光センサ信号	回転方向信号
未接続	未接続	1: 有効	未接続	1: 信号有効	1: 電流オン	1: 回転体なし	1: C CW
		0: 無効		0: 信号無効 (モータ停止)	0: 電流オフ	0: 回転体検出	0: CW

図 4-2 VPT ポートのピン割り付け

bit5 (BRK: ブレーキ信号)、bit3 (ENB: イネーブル信号)、bit2 (PWM: PWM 信号)、bit0 (DIR: 回転方向信号) の 4 つの信号でモーターを制御します。

初期値を bit5 (0: ブレーキなし)、bit3 (0: モーター停止)、bit2 (0: 電流オフ)、bit0 (0: 時計方向) とし、VPT に 0x00 を出力します。その後は DIP\_SW (VPA) を入力し、それをモータードライバ IC (VPT) に出力し、モーターを制御します。4 ビットの DIP\_SW をそれぞれ動かしてみましょう。

**[ 解答 ]**

リスト 4.1.1 にプログラム例を示します。

リスト 4.1.1 DIP\_SW で制御信号 (brk、enb、pwm、dir) を設定し、DC モータをオンオフ制御する

```

/** main.c *****/
#include <mt_r300.h> // マイコンボード用ライブラリー
void main (void)
{
    unsigned char sw;
    setup_VPT(0x2d); // 初期化 b7: 入力(), b6: 入力(), b5: 出力 (BRK), b4: 入力(),
                    // b3: 出力 (ENB), b2: 出力 (PWM), b1: 入力 (SENS), b0: 出力 (DIR)
// setup_VPA(0x00); // 初期化 (VPA は全入力固定のため省略可)
    out_VPT(0x00); // 初期値 (b5: 0 (ブレーキなし), b3: 0 (停止), b2: 0 (電流オフ), b0: 0 (時計方向))
    while(1){
        sw = in_VPA(); // DIP_SW で設定 (b5: brk, b3: enb, b2: pwm, b0: dir)
        out_VPT(sw & 0x2d); // DC モータを制御 (有効ビットを出力)
    }
}

```

## 実習 4.1.2 DIP\_SW (b6 ~ 0) でデューティ比 (0 ~ 100) を設定し、DC モータを PWM 制御する

### [ 解説 ]

PWM 制御によりモータの回転速度を制御します。

周期を 10ms とすると、

- ・ 電流オンの時間は、デューティ比 / 100 × 10ms = デューティ比 / 100 μs
- ・ 電流オフの時間は、(100 - デューティ比) / 100 × 10ms = (100 - デューティ比) / 100 μs

となります。

### [ 解答 ]

リスト 4.1.2 にプログラム例を示します。

リスト 4.1.2 DIP\_SW (b6 ~ 0) でデューティ比 (0 ~ 100) を設定し、DC モータを PWM 制御する

```
/** main.c *****/
#include <mt_r300.h> // マイコンボード用ライブラリー
#define BRK 0x20 // ブレーキあり
#define NBRK 0x00 // ブレーキなし
#define ENB 0x08 // イネーブル(制御信号有効)
#define DSB 0x00 // ディスエーブル(制御信号無効、モーター停止)
#define PWMH 0x04 // PWM 信号 H(電流オン)
#define PWML 0x00 // PWM 信号 L(電流オフ)
#define CCW 0x01 // 時計方向
#define CW 0x00 // 反時計方向
void main (void)
{
    unsigned char dt;
    setup_VPT(0x2d); // 初期化 b7: 入力(), b6: 入力(), b5: 出力(BRK), b4: 入力(),
                    // b3: 出力(ENB), b2: 出力(PWM), b1: 入力(SENS), b0: 出力(DIR)
    out_VPT(NBRK | ENB | PWML | CW); // 初期値(ブレーキなし, 信号有効, 電流オフ, 時計方向)
    while(1){
        dt = in_VPA() & 0x7f; // DIP_SW(b6 ~ b0) でデューティ比(0 ~ 100) を入力する
        dt = (dt > 100) ? 100 : dt; // 100 を越えない値(小さい方)
        // PWM 信号の周波数; 100Hz(周期: 10ms) (適正周波数; 100Hz ~ 1kHz(周期: 10 ~ 1ms))
        out_VPT(NBRK | ENB | PWMH | CW); // 電流オン
        wait_us_itu2(dt * 100); // 継続時間 dt * 100[μs]
        out_VPT(NBRK | ENB | PWML | CW); // 電流オフ
        wait_us_itu2((100-dt) * 100); // 継続時間 (100-dt) * 100[μs]
    }
}
```

**実習 4.1.3** DIP\_SW でデューティ比 (0 ~ 100) を設定し、DC モータを PWM 制御する (ITU の PWM モード)

**[ 解説 ]**

16 ビットタイマー ITU2 ~ 0 は PWM モードで動作させることができます。図 4-3 は PWM モードの動作例を示します。16TCNT の値が GRB0 の値に一致した時 0 クリアされる設定をした場合についての説明です。

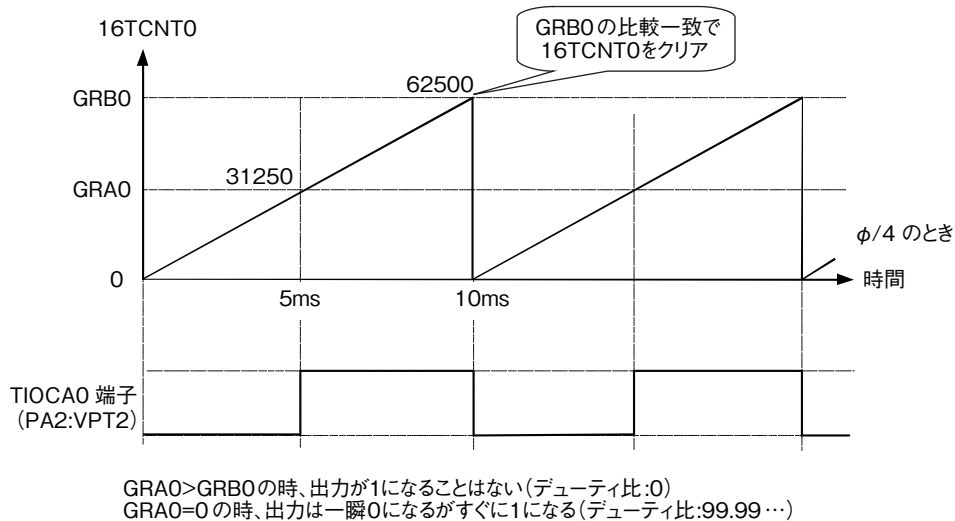


図 4-3 PWM モードの動作

TIOCA0 端子 (PA2、VPT2) に PWM 信号が出力されます。具体的には、16TCNT0 の値が GRA0 の値に一致した時 1 が出力され、GRB0 の値に一致した時 0 が出力されます。GRA0 > GRB0 の時は、16TCNT0 の値が GRA0 の値に一致することがないので、出力は常に 0 です (デューティ比:0)。GRA0 = 1 - 1 のときは 1 カウント分だけ 0 が出力され、すぐに 1 が出力されます (デューティ比: 99.99... (これが最大値で、以下これを便宜的に 100 とします))。デューティ比を完全に 100 にするためには、16TCNT の値が GRA0 の値に一致した時 0 クリアされる別の設定法が必要ですが、ここでは省略します。

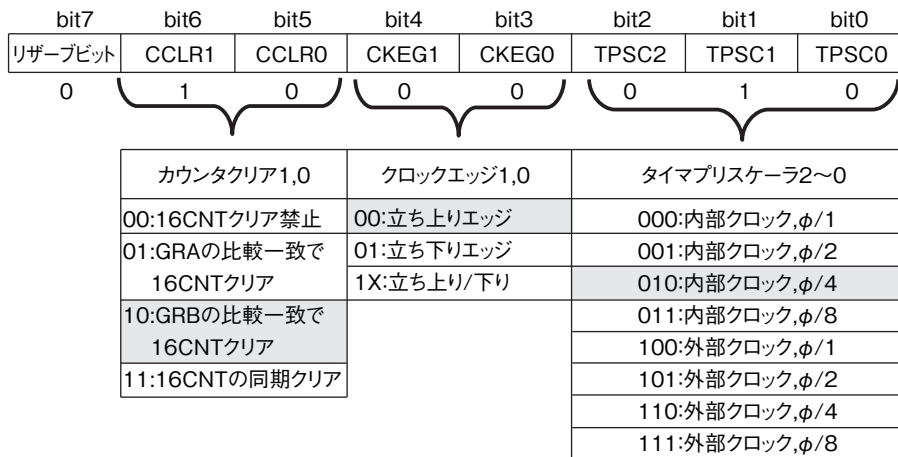
図 4-4 は PWM モードでの ITU0 のレジスタの設定法を示します。DC モーターのイニシャライズ関数を作成し、その中で ITU0 のレジスタを設定します。

その概要は次のとおりです。PWM 信号の周期は 10m 秒とします。クロック 25MHz を 1/4 分周し 6.25MHz (周期:0.16 μs) を生成します。0.16 μs を 62500 回カウントし、周期 10m 秒を得ます。

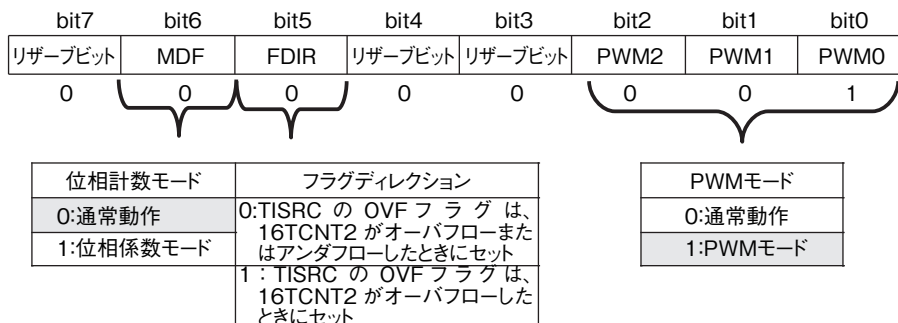
- |   |                        |
|---|------------------------|
| ・ ITU0 を PWM モードに設定                               | ITU.TMDR.BIT.PWM0 = 1; |
| ・ GRB0 と 16TCNT0 のコンペアマッチで 16TCNT0 クリア、1/4 分周     | ITU0.TCR.BYTE = 0x42;  |
| ・ 25MHz/4=6.25MHz (0.16 μs)、0.16 μs × 62500=10m 秒 | GRB0=62500-1;          |
| ・ デューティ比を 0 にするため、GRA0 > GRB0 にする (最大値設定)         | ITU0.GRA = 65535;      |
| ・ PWM 動作を開始する                                     | ITU.TSTR.BIT.STR0 = 1; |

プログラム中ではデューティ比を変更します。100 にする時には、GRA0=1 - 1 に設定します。

## タイマコントロールレジスタ(16TCR)



## タイマモードレジスタ(TMDR)



## タイマスタートレジスタ(TSTR)

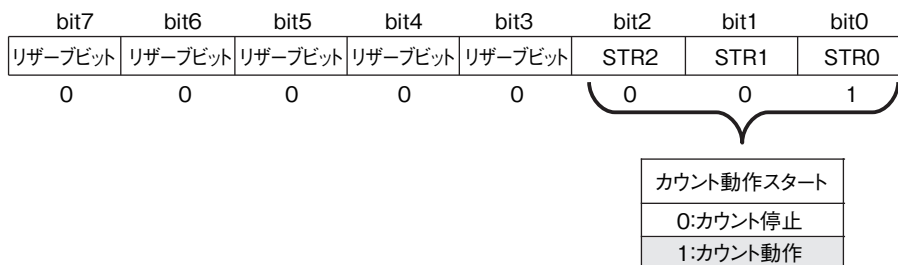


図 4-4 PWM モードの ITU0 のレジスタの設定

### [ 解答 ]

リスト 4.1.3 にプログラム例を示します。

リスト 4.1.3 DIP\_SW でデューティ比 (0 ~ 100) を設定し、DC モータを PWM 制御する (ITU の PWM モード)

```

/** main.c *****/
#include <iodef.h>
#include <mt_r300.h> // マイコンボード用ライブラリー
#define BRK 0x20 // ブレーキあり
#define NBRK 0x00 // ブレーキなし
#define ENB 0x08 // イネーブル (制御信号有効)
#define DSB 0x00 // ディスエーブル (制御信号無効、モーター停止)
#define CCW 0x01 // 時計方向
    
```

```
#define CW      0x00    // 反時計方向
void initilize_dcm()
{
    setup_VPT(0x29);      // ポートの初期化      b7: 入力(), b6: 入力(), b5: 出力(BRK), b4: 入力(),
                        // b3: 出力(ENB), b2: 入力((PWM時は出力), b1: 入力(SENS), b0: 出力(DIR)
    out_VPT(NBRK | ENB | CW);    // 初期値(ブレーキなし, 信号有効, 時計方向)
    // PWM 信号の周波数; 100Hz(周期: 10ms) (適正周波数; 100Hz ~ 1KHz(周期: 10 ~ 1ms))
    ITU.TMDR.BIT.PWM0 = 1;      // チャンネル0 を PWM モードに設定
    ITU0.TCR.BYTE = 0x42;      // GRB と TCNT のコンペアマッチ, TCNT クリア, 1/4 分周
    ITU0.GRB = 62500 - 1;      // 周期: 25MHz/4=6.25MHz(0.16 μs), 0.16 μs × 62500=10ms
    ITU0.GRA = 65535;      // GRA>GRB とし、デューティ比を 0 にする
    ITU.TSTR.BIT.STRO = 1;      // PWM 動作開始
}
void main (void)
{
    unsigned char dty;
    initilize_dcm();
    while(1){
        dty = in_VPA() & 0x7f;    // DIP_SW(b6 ~ b0) でデューティ比 (0 ~ 100) を入力
        if(dty == 0)
            ITU0.GRA = 65535;    // GRA>GRB とし、デューティ比を 0 にする
        else if(dty >= 100)
            ITU0.GRA = 1 - 1;    // GRA=0 とし、デューティ比を 100(99.99...) にする
        else
            ITU0.GRA = 62500 * (100 - dty) / 100 - 1; // PWM_L の時間
    }
}
```

## 4.2 DC モーター制御関数のライブラリー作成

ここでは 4.1 で作成したモーター制御関数をライブラリーにまとめます。"X:¥xxxx¥workspace" にワークスペース、プロジェクト (dcm\_lib) を新規作成し、4.1 と同様に種々の設定をします。

ライブラリー ("mt\_e508.c", "mt\_e508.h") を新規作成し、"X:¥xxxx¥workspace¥@library" に保存します。"mt\_r300.c" と同様に "mt\_e508.c" もプロジェクトに追加します。

### 実習 4.2.1 DIP\_SW (b7) で DC モータの起動 / 停止を制御し、DIP\_SW (b6 ~ 0) でデューティ比を調整する

#### [解説]

ライブラリー "mt\_e508.c" に実習 4.1.1 のモーター制御関数を移植し、新規作成します。合わせて、"mt\_e508.h" にその関数のプロトタイプ宣言を作成します。

#### [解答]

リスト 4.2.1 にプログラム例を示します。

# Sunhayato

リスト 4.2.1 DIP\_SW (b7) で DC モータの起動 / 停止を制御し、DIP\_SW (b6 ~ 0) でデューティ比を調整する

```
/** main.c *****/
#include <mt_r300.h> // マイコンボード用ライブラリー
#include <mt_e508.h> // DC モータボード用ライブラリー
void main (void)
{
    unsigned char dt;
    initilize_dcm(); // DC モータ用ドライバーと ITU0(PWM モード)の初期化
    while(1){
        while(!(in_VPA() & 0x80)); // DIP_SW(b7) が 0 の間待つ
        start_dcm(CW); // 起動 (時計方向)
        while(in_VPA() & 0x80){ // DIP_SW(b7) が 1 の間繰り返す
            dt = in_VPA() & 0x7f; // DIP_SW(b6 ~ 0) でデューティ比 (0 ~ 100) を入力する
            set_dt(dt); // デューティ比設定
        }
        stop_dcm(); // 停止
    }
}
```

```
/** mt_e508.c (新規作成) *****/
#include <iodef.h> // IO 定義用ヘッダーファイル
#include "mt_r300.h" // マイコンボード用ライブラリー
#include <mt_e508.h> // DC モータボード用ライブラリー
/**** DC モータを設定する関数 ****/
void setup_dcm(unsigned char brk, unsigned char enb, unsigned char dir)
{ // brk(1: ブレーキあり / 0: なし), enb(1: 制御信号有効 / 0: 停止), dir(1: 反時計方向 / 0: 時計方向)
    out_VPT(brk | enb | dir);
}
/**** DC モータと ITU0(PWM モード)を初期設定する関数 ****/
void initilize_dcm(void)
{
    setup_VPT(0x29); // ポートの初期化 b7: 入力(), b6: 入力(), b5: 出力 (BRK), b4: 入力(),
    // b3: 出力 (ENB), b2: 入力 (PWM 時は出力), b1: 入力 (SENS), b0: 出力 (DIR)
    setup_dcm(BRK, ENB, CW); // DC モーターの初期設定 (ブレーキあり, 信号有効, 時計方向)
    ITU.TMDR.BIT.PWMO = 1; // チャンネル 0 を PWM モードに設定
    ITU0.TCR.BYTE = 0x42; // GRB と TCNT のコンペアマッチ, TCNT クリア, 1/4 分周
    ITU0.GRB = 62500 - 1; // 周期: 25MHz/4=6.25MHz(0.16 μs), 0.16 μs × 62500=10ms
    ITU0.GRA = 65535; // GRA>GRB とし、デューティ比を 0 にする
    ITU.TSTR.BIT.STRO = 1; // PWM 動作開始
}
/**** DC モータを起動 (指定方向) する関数 ****/
void start_dcm(unsigned char dir) // direct(CW: 時計方向, CCW: 反時計方向)
{
    setup_dcm(NBRK, ENB, dir); // 設定 (ブレーキなし, 制御信号有効, 設定方向)
}
```

```

/**** DC モータを停止する関数 ****/
void stop_dcm(void)
{
    setup_dcm(NBRK, DSB, CW);          // 設定 (ブレーキなし, モーター停止, 時計方向)
}

/**** DC モータをブレーキングする関数 ****/
void break_dcm(void)
{
    setup_dcm(BRK, ENB, CW);          // 設定 (ブレーキあり, 制御信号有効, 時計方向)
}

/**** PWM 信号のデューティ比を設定する関数 ****/
void set_dty(unsigned char dty)      // dty(デューティ比:0~100)
{
    if(dty == 0)
        ITU0.GRA = 65535;             // GRA>GRB とし、デューティ比を 0 にする
    else if(dty >= 100)
        ITU0.GRA = 1 - 1;             // GRA=0 とし、デューティ比を 100(99.99...) にする
    else
        ITU0.GRA = 62500 * (100 - dty) / 100 - 1; // PWM_L の時間
}

```

```

/** mt_e508.h (新規作成) *****/
#include <mt_r300.h> // マイコンボード用ライブラリー
#define BRK 0x20 // ブレーキあり
#define NBRK 0x00 // ブレーキなし
#define ENB 0x08 // イネーブル (制御信号有効)
#define DSB 0x00 // ディスエーブル (制御信号無効、モーター停止)
#define CCW 0x01 // 反時計方向
#define CW 0x00 // 時計方向
void initilize_dcm(void);
void start_dcm(unsigned char);
void stop_dcm(void);
void break_dcm(void);
void set_dty(unsigned char);

```

#### 実習 4.2.2 DC モータを回転し、フォトインタラプタのカウンタ数を評価用 LED (8 ビット) に表示する

##### [ 解説 ]

ライブラリー "mt\_e508.c" にフォトセンサの関数を追加移植します。合わせて、"mt\_e508.h" にその関数のプロトタイプ宣言を追加移植します。

##### [ 解答 ]

リスト 4.2.2 にプログラム例を示します。

# Sunhayato

リスト 4.2.2 DC モーターを回転し、フォトセンサのカウンタ数を評価用 LED (8 ビット) に表示する

```
/** main.c *****/
#include <mt_r300.h> // マイコンボード用ライブラリー
#include <mt_e508.h> // DC モーターボード用ライブラリー
void main (void)
{
    unsigned char dty, sens, sens_s, sens_cnt=0;
    setup_VPE(0xff);
    initilize_dcm(); // DC モーターと ITU0(PWM モード)の初期化
    while(1){
        while(!(in_VPA() & 0x80)); // DIP_SW(b7) が 0 の間待つ
        sens_s = get_sensor(); // センサの初期値を保存
        start_dcm(CW); // 起動 (時計方向)
        while(in_VPA() & 0x80){ // DIP_SW(b7) が 1 の間繰り返す
            dty = in_VPA() & 0x7f; // DIP_SW(b6 ~ 0) でデューティ比 (0 ~ 100) を入力する
            set_dty(dty); // デューティ比設定
            sens = get_sensor(); // センサの状態を取り込む
            if(sens_s != sens){ // センサの状態が変化した時
                sens_s = sens; // センサの状態を保存
                sens_cnt++; // カウンタアップ (1/4 回転毎)
                out_VPE(sens_cnt); // 評価用 LED に表示
            }
        }
        stop_dcm(); // 停止
    }
}
```

```
/** mt_e508.c (以下を追加します) *****/
**** フォトセンサの状態を取り込む関数 ****/
unsigned char get_sensor(void)
{
    unsigned char sens;
    sens = (in_VPT() & 0x02) >> 1;
    return sens;
}
```

```
/** mt_e508.h (以下を追加します) *****/
unsigned char get_sensor(void);
```

## 4.3 LCD への表示

ここでは LCD ボードを用いて、モーターの回転数、回転速度を表示します。"X:¥xxxx¥workspace" にワークスペース、プロジェクト (dcm\_lcd) を新規作成します。4.2 と同様に種々の設定をします。プロジェクトに "mt\_e508.c"、"mt\_e502.c" を追加します。"mt\_e502.c" については「トレーニングテキスト LCD 基板 (MT-E502) 編」を参照してください。また、LCD 基板 (MT-E502) は VPD ポートに接続してください。

**実習 4.3.1 DC モータを回転し、LCD に回転数を表示する**

**[ 解説 ]**

フォトインタラプタで回転体の有無を検知し、その信号の変化をカウントします。約 1/4 回転で 1 カウントするため、カウント数を 4 で割った値が回転数になります。これを LCD に表示します。

**[ 解答 ]**

リスト 4.3.1 にプログラム例を示します。

リスト 4.3.1 DC モータを回転し、LCD に回転数を表示する

```

/** main.c *****/
#include <mt_r300.h> // マイコンボード用ライブラリー
#include <mt_e502.h> // LCD ボード用ライブラリー
#include <mt_e508.h> // DC モータボード用ライブラリー
void main (void)
{
    unsigned char sw, sw_s = 0, dty, sens, sens_s;
    unsigned long sens_cnt = 0;
    initilize_dcm(); // DC モータと ITU0(PWM モード) の初期化
    initialize_lcd(); // LCD ボードの初期化
    clear_display();
    locate_cursor(1, 1);
    prints_lcd("DIP_SW_b7->start");
    locate_cursor(2, 1);
    prints_lcd("rev = ");
    locate_cursor(2, 7);
    printi_lcd(10, 0); // 回転数 (0) を表示
    sens_s = get_sensor(); // センサの初期状態を保存
    while(1){
        sw = (in_VPA() & 0x80) >> 7; // DIP_SW(b7) を入力
        if(sw != sw_s){ // SW の状態が変化した時
            if(sw){
                start_dcm(CW); // 起動 (時計方向)
            }
            else{
                stop_dcm(); // 停止
            }
            sw_s = sw;
        }
        dty = in_VPA() & 0x7f; // DIP_SW(b6 ~ 0) でデューティ比 (0 ~ 100) を入力する
        set_dty(dty); // デューティ比設定
        sens = get_sensor(); // センサの状態を取り込む
        if(sens_s != sens){ // センサの状態が変化した時
            sens_s = sens; // センサの状態を保存
            sens_cnt++; // カウントアップ (1/4 回転毎)
        }
    }
}

```

```
locate_cursor(2, 7);
printi_lcd(10, sens_cnt/4);           // 回転数 (カウント数 / 4) を表示
}
}
}
```

## 実習 4.3.2 DC モータを回転し、LCD に回転数と回転速度を表示する

### [ 解説 ]

1/4 回転毎にカウントアップするため、0.25 秒毎に回転速度を計算すると分かりやすくなります。この場合の回転速度は、次のように求められます。

回転速度 [rps] = 回転数変化 / 0.25 = (カウント数変化 / 4) / 0.25 = カウント数変化

回転速度は通常 1 分間の回転数で表すため、回転速度 [rpm] = カウント数 × 60 となります。

ITU1 を用いて 10ms の基準カウンタの関数を作成します。これを 25 回カウントすることで 0.25 秒 (250ms) の間隔がとれます。ITU1 の TCNT はゼロクリア後もカウント動作を続けているので、while(1) ループ内の処理時間は除外されます。この方法は、while(1) ループの処理時間 (printf() 文など) が、基準カウンタの 10ms より長くなると正しく動作しないので注意が必要です。

### [ 解答 ]

リスト 4.3.2 にプログラム例を示します。

#### リスト 4.3.2 DC モータを回転し、LCD に回転数と回転速度を表示する

```
/** main.c *****/
#include <iodefne.h>           // I/O 定義用ヘッダファイル
#include <mt_r300.h>          // マイコンボード用ライブラリー
#include <mt_e502.h>          // LCD ボード用ライブラリー
#include <mt_e508.h>          // DC モータボード用ライブラリー

unsigned char count_10ms(void)
{
    if(ITU.TISRA.BIT.IMFA1 == 1){           // 比較一致フラグ == 1
        ITU.TISRA.BIT.IMFA1 = 0;          // ITU0 の比較一致フラグクリア
        return 1;
    }
    else
        return 0;
}

void main (void)
{
    unsigned char sw, sw_s = 0, dty, sens, sens_s, loop_cnt = 0;
    unsigned long sens_cnt = 0, sens_cnt_s = 0, rps;
    initilize_dcm();                 // DC モータと ITU0(PWM モード) の初期化
    initialize_lcd();                // LCD ボードの初期化
    clear_display();
```

```

locate_cursor(1, 1);
prints_lcd("rev = ");
locate_cursor(1, 7);
printi_lcd(10, 0); // 回転数 (0) を表示
locate_cursor(2, 1);
prints_lcd("rpm = ");
locate_cursor(2, 7);
printi_lcd(10, 0); // 回転速度 (0) を表示
// ITU1 タイマの設定
ITU1.TCR.BYTE = 0x23; // TCNT と GRA の比較一致で TCNT クリア ,1/8 φ
ITU1.GRA = 31250 - 1; // 割り込み周期 :25MHz/8=3.125MHz(0.32 μs),0.32 μs × 31250=10m 秒
ITU1.TCNT = 0; // TCNT クリア
ITU.TISRA.BIT.IMFA1 = 0; // TCNT と GRA の比較一致フラグクリア
ITU.TISRA.BIT.IMIEA1 = 0; // ITU の割り込み禁止
ITU.TSTR.BIT.STR1 = 1; // TCNT のカウントスタート
sens_s = get_sensor(); // センサの初期状態を取り込む
while(1){
    sw = (in_VPA() & 0x80) >> 7;
    if(sw != sw_s){ // SW の状態が変化した時
        if(sw){
            start_dcm(CW); // 起動 (時計方向)
        }
        else{
            stop_dcm(); // 停止
        }
        sw_s = sw;
    }
    dty = in_VPA() & 0x7f; // DIP_SW(b6 ~ b0) でデューティを入力する
    set_dty(dty); // デューティ設定
    sens = get_sensor(); // センサの状態を取り込む
    if(sens_s != sens){ // センサの状態が変化した時
        sens_s = sens; // センサの状態を保存
        sens_cnt++; // カウントアップ (1/4 回転毎)
        locate_cursor(1, 7);
        printi_lcd( 10, sens_cnt/4); // 回転数 ( カウント数 /4) を表示 ( 毎回 )
    }
    if(count_10ms()){ // 10m 秒経過した時
        loop_cnt++;
        if(loop_cnt >= 25){ // 25 回 [10ms × 25=250ms] 経過した時
            loop_cnt = 0;
            rps = sens_cnt - sens_cnt_s; // 回転速度 [rps]= 回転数変化 /0.25= カウント数変化
            sens_cnt_s = sens_cnt;
            locate_cursor(2, 7);
            printi_lcd(10, rps*60); // 回転速度 [rpm](rps*60) を表示 ((250ms 毎))
        }
    }
}

```

```
}  
}  
}
```

while(1) ループは printi\_lcd() 文などがあるため、10ms 弱の時間がかかります。従って、センサの信号がこれより速く変化する、正しくカウントできません。

1/4 回転が 10ms 程度とすると回転速度は 25[rps]=1500[rpm] 程度となり、これが測定の限界です。

## 実習 4.3.3 DC モータを回転し、LCD に回転数と回転速度を表示する（タイマ割り込み関数使用）

### [ 解説 ]

実習 4.3.2 では回転速度が 1500rpm 程度しか対応できませんでした。ここでは短い（250  $\mu$  秒）間隔のタイマ割り込みで、センサの信号を取り込み適切な処理をします。原理的には 250  $\mu$  秒毎に 1/4 回転する速度（1 回転 /1ms、1000r/s、6000rpm）まで検知できます。従って、このモーター（回転速度 :15200rpm/3V）の実験で十分対応します。

### [ 解答 ]

リスト 4.3.3 にプログラム例を示します。

#### リスト 4.3.3 DC モータを回転し、LCD に回転数と回転速度を表示する（タイマ割り込み関数使用）

```
/** main.c *****/
#include <iodefne.h> // IO 定義用ヘッダーファイル
#include <mt_r300.h> // マイコンボード用ライブラリー
#include <mt_e502.h> // LCD ボード用ライブラリー
#include <mt_e508.h> // DC モータボード用ライブラリー
unsigned char rps_flag = 0;
unsigned long sens_cnt = 0, rps;
void main (void)
{
    unsigned char sw, sw_s = 0, dt;
    initilize_dcm(); // DC モータと ITU0(PWM モード)の初期化
    initialize_lcd(); // LCD ボードの初期化
    clear_display();
    locate_cursor(1, 1);
    prints_lcd("rev = ");
    locate_cursor(1, 7);
    printi_lcd(10, 0); // 回転数 (0) を表示
    locate_cursor(2, 1);
    prints_lcd("rpm = ");
    locate_cursor(2, 7);
    printi_lcd(10, 0); // 回転速度 (0) を表示
    // ITU1 タイマの割り込み関数 (intprg.c) の準備
    ITU1.TCR.BYTE = 0x20; // TCNT と GRA の比較一致で TCNT クリア ,1/1  $\phi$ 
    ITU1.GRA = 6250 - 1; // 割り込み周期 :25MHz/1=25MHz(0.04  $\mu$  s), 0.04  $\mu$  s  $\times$  6250=250  $\mu$  秒
    ITU1.TCNT = 0; // TCNT クリア
    ITU.TISRA.BIT.IMFA1 = 0; // TCNT と GRA の比較一致フラグクリア
```

```

ITU.TISRA.BIT.IMIEA1 = 1;           // ITU の割り込み許可
ITU.TSTR.BIT.STR1 = 1;             // TCNT のカウントスタート
while(1){
    sw = (in_VPA() & 0x80) >> 7;
    if(sw != sw_s){                // SW の状態が変化した時
        if(sw){
            start_dcm(CW);         // 起動 (時計方向)
        }
        else{
            stop_dcm();           // 停止
        }
        sw_s = sw;
    }
    dty = in_VPA() & 0x7f;         // DIP_SW(b6 ~ b0) でデューティ比を入力する
    set_dty(dty);                 // デューティ比設定
    locate_cursor(1, 7);
    printi_lcd(10, sens_cnt/4);    // 回転数 (カウント数 /4) を表示 (毎回)
    if(rps_flag){                 // rps の演算終了 (250ms) 毎に
        rps_flag = 0;
        locate_cursor(2, 7);
        printi_lcd( 10, rps*60);  // 回転速度 [rpm=rps*60] を表示
    }
}
}

```

```

/** intprg.c *****/
#include <iodefne.h>                // IO 定義用ヘッダーファイル
#include <mt_r300.h>                // マイコンボード用ライブラリー
#include <mt_e508.h>                // DC モーターボード用ライブラリー
extern unsigned char rps_flag;
extern unsigned long sens_cnt, rps;
__interrupt(vect=28) void INT_IMIA1(void){ // タイマ割り込み関数 (250 μ秒毎)
    static unsigned char sens_s = 9;     // 未定
    static unsigned long sens_cnt_s = 0;
    static unsigned int loop_cnt = 0;
    unsigned char sens;
    ITU.TISRA.BIT.IMFA1 = 0;           // TCNT と GRA の比較一致フラグクリア
    if(sens_s == 9)                    // 初期 (未定) 時
        sens_s = get_sensor();         // センサの初期状態 (1/0) を取り込む
    sens = get_sensor();                // センサの状態を取り込む
    if(sens_s != sens){                 // センサの状態が変化した時
        sens_s = sens;                 // センサの状態を保存
        sens_cnt++;                    // カウントアップ (1/4 回転毎)
    }
    loop_cnt++;
}

```

# Sunhayato

---

```
if(loop_cnt >= 1000){ // 1000回(250 μs*1000=250ms)毎に
    loop_cnt = 0;
    rps = sens_cnt - sens_cnt_s; // 回転速度[rps]=回転数変化/0.25=カウント数変化
    sens_cnt_s = sens_cnt; // カウント数の保存
    rps_flag = 1; // 回転速度の演算終了
}
}
```

## 5 おわりに

---

この実習により、DC モーターを制御する基本プログラムが理解できたことと思います。モータードライバー用 IC、IC ロジック回路についても理解されたことと思います。各信号のレベル（H/L）の変化を視認するモニタ LED も効果的だったと思います。それらの制御プログラムをライブラリーにまとめましたので、今後はそれを活用して種々の制御プログラムを作成しましょう。

モーターは電源や負荷の変動により回転速度が変動します。センサの信号を読み取り、回転速度を一定に保つ PI 制御などのプログラムも作成できますが、ここでは省略します。

---

**マイコントレーニング基板 MT-R300**  
**トレーニングテキスト DC モーター基板編**

発行日 2011-8-20 Rev1.00  
発行 サンハヤト株式会社  
〒170-0005 東京都豊島区南大塚3丁目40番1号

©2011 Sunhayato Corp. All rights reserved.

SG11006

---