LV8548MCSLDGEVK

ブラシDCモータドライバモジュール ソリューションキット アプリケーションノート

ON Semiconductor®

www.onsemi.jp

APPLICATION NOTE

このアプリケーションノートは、LV8548MCSLDGEVKモータドライバソリューションキットのサポート情報を提供します。

キット概要

オンセミコンダクター社が提供するモータドライバ製品を使用してモータ制御アプリケーションシステムを開発する場合、まず最初にモータドライバ製品の仕様を理解した上でハードウエアの設計を行い、次にドライバ製品に入力する動作制御信号(回転方向、回転速度、回転角度など)を生成する必要があります。一般的にこの様な動作制御信号はマイコンなどを使用して生成するため、上述ハードウエア設計に加えてマイコン用ソフトウェアの開発も必要となります。

本キットは、オンセミコンダクター社のモータドライバ製品(LV8548MC)をArduino MICRO マイコンから制御するためのモータドライバ制御用API 関数ライブラリを提供します。

また本キットは、本API関数ライブラリを組込んだArduino MICROをパソコンからUSB通信を介してターゲットモータを制御するための専用GUIも提供しておりますので、Arduino MICRO用モータ制御ソフトを開発することなくモータの制御シーケンスや動作パラメータをチューニングするなどのデバッグ作業を容易に行うことが可能となっています。

更には、本GUIはデバッグした制御シーケンスや動作パラメータをArduino MICROマイコンで実現するための制御ソフト（ソースコード）をArduino IDEでコンパイル可能な形式(スケッチ)で出力する、自動コード生成機能も有しております。

従って、本キットを使用することにより、モータドライバの仕様や、制御用API関数を使用したソフト開発など、特別な知識を有していなくてもモータ制御アプリケーションシステムの雛形(プロトタイプ)を容易に開発することを可能としているため、開発期間の短縮と大幅なコスト低減を実現します。

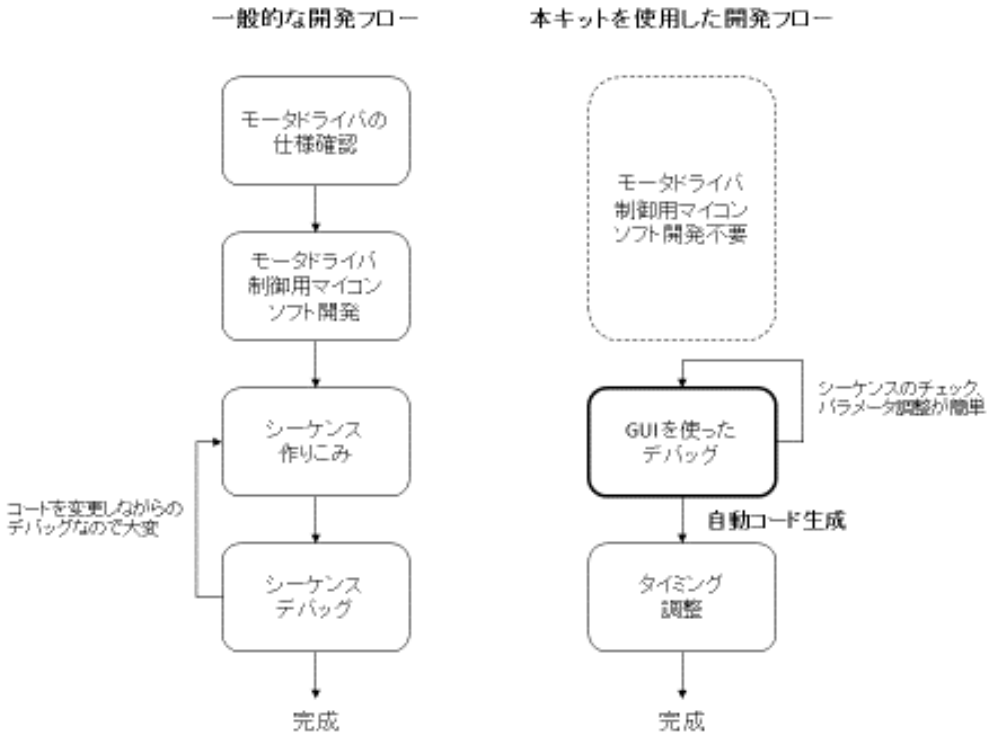


図 1. 一般的な開発フローと本キットを使用した開発フロー

GUIを使ったデバッグモードとスタンドアローン開発モード

本キットは、GUIを使ったデバッグモードと自動コード生成機能を用いたスタンドアローン開発モード、オリジナルスケッチを用いたスタンドアローン開発モードの計3種類のデバッグおよび開発モードの使用を想定しています。

GUIを使ったデバッグモード

GUIを使ったデバッグモードは、ユーザがPCにインストールされた専用GUIを操作することで、LV8548に対してモータの制御シーケンスや動作パラメータを変更でき、実際にモータを動かしながらパラメータのチューニングが行えます。

また、ユーザがGUIを操作して設定したモータの制御シーケンスや動作パラメータを反映し、Arduino MICROにコンパイル・書き込みできるスケッチ(.inoファイル)として出力できる自動コード生成機能が備わっています。

このモードを使用するためには、Arduino IDEを使用し、キットに同梱されるGUI専用スケッチ(LV8548_DC_Program.ino)とArduino MICRO専用モータ駆動API関数ライブラリ(LV8548_DC_Lib.cpp/h)をコンパイルすることで生成されるファームウェアをArduino MICROに書き込む必要があります。

GUIを使ったデバッグモードの概要を図2に示します。

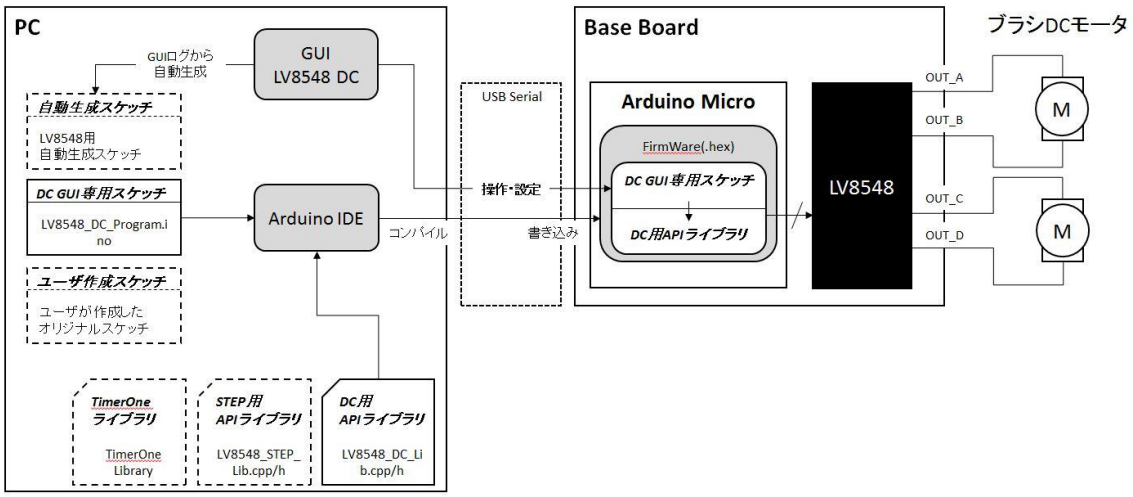


図 2. GUIを使ったデバッグモードの概要

スタンドアローン開発モード

スタンドアローン開発モードは、GUIを使ったデバッグモードで作成した自動生成スケッチを元に、モータ駆動タイミングの調整やユーザオリジナルのソースコードの追加などを行い、GUI専用スケッチの代わりにArduino MICROに書き込むことで、本キットを用いたスタンドアローンでのDCモータ駆動を容易に行える開発モードです。

自動生成機能を用いたスタンドアローン開発モードの概要を図3に示します。

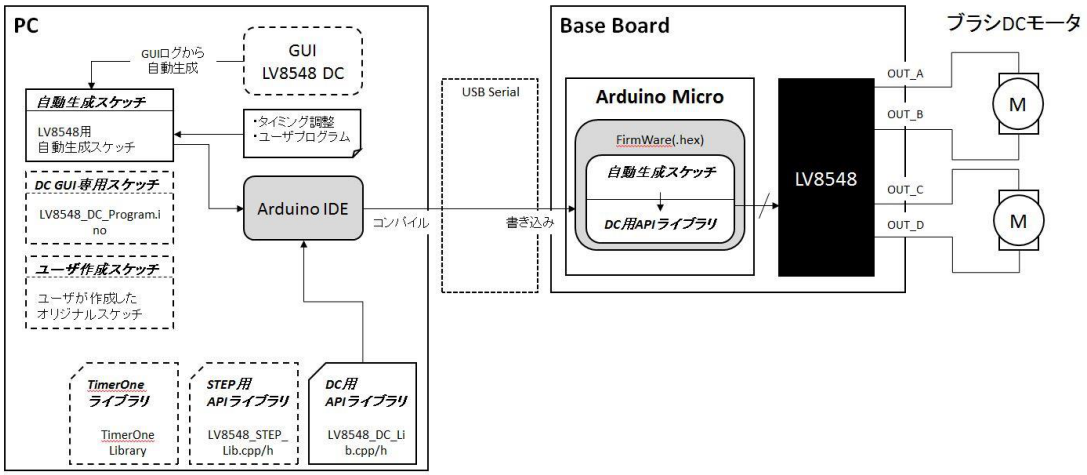


図 3. 自動生成機能を用いたスタンドアローン開発モード

自動コード生成機能を用いずに、API関数ライブラリを利用したオリジナルスケッチを利用することもできます。高度なプログラミング知識があれば、より複雑で高度なモータ制御アプリケーションを開発することも可能です。

ユーザオリジナルスケッチを用いたスタンドアローン開発モードの概要を図4に示します。

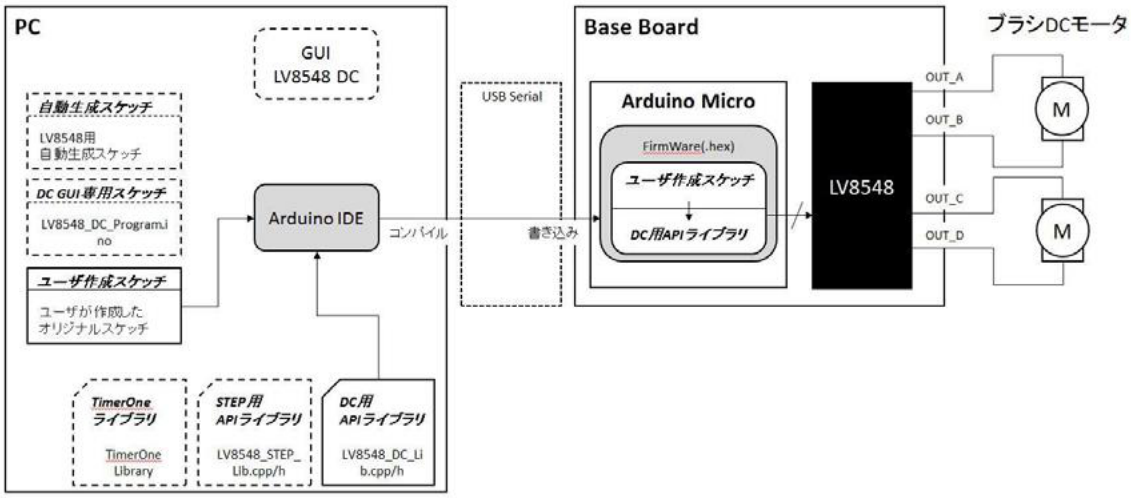


図 4. オリジナルスケッチを用いたスタンドアローン開発モード

Arduinoのスケッチについて

スケッチの概要

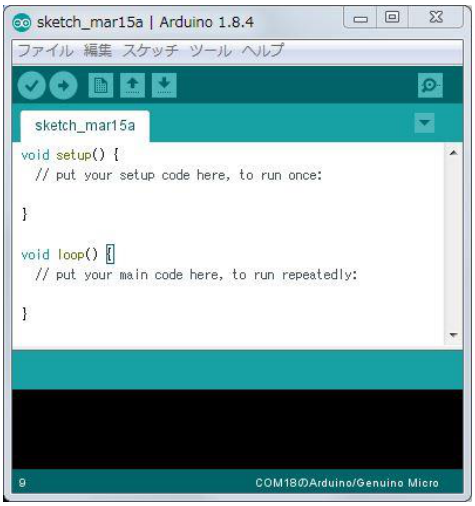
スケッチは、Arduino言語で記述されたArduinoのためのプログラムの名称であり、Arduino Boardにアップロードされて動くコードのまとまりを指します。Arduino言語はC/C++をベースに構築されており、C言語の全構造とC++の一部機能をサポートしています。

スケッチの詳細は以下を参照してください。

https://www.arduino.cc/en/Tutorial/Sketch

setup()とloop()関数

Arduino IDEから新規スケッチの作成を行うと、下図のように自動的にsetup()とloop()が挿入されます。



setup()は、Arduino Boardの電源オンの後、またはリセット後に一度だけ呼び出される関数で、変数やピンモードの初期化ライブラリの使用準備などを記述します。

setup()の詳細は以下を参照してください。

https://www.arduino.cc/reference/en/language/structure/sketch/setup/

loop()は名前のとおり、繰り返し**実**行される関数で、この中に**実**際に繰り返し動作させたいプログラムを記述します。

loop()の詳細は以下を参照してください。

https://www.arduino.cc/reference/en/language/structure/sketch/loop/

DCモータ制御**API**ライブラリの概要

DCモータ制御APIライブラリ(LV8548_DC_APILibrary)は、Arduino Microからオンセミコンダクター社製 モータドライバLV8548を使用して、DCモータを制御するためのライブラリを提供します。ユーザーは、本APIライブラリをArduino IDEからインクルードし、目的に沿ったAPI関数をスケッチに記述することで容易にLV8548を使用したDCモータ制御が可能となります。

表 1. DCモータ制御APIライブラリファイル一覧

#	ファイル名	内容
1	keywords.txt	キーワードファイル (スケッチ内の表示色変更対象ワードの設定)
2	LV8548_DC_Lib.cpp	ソースファイル
3	LV8548_DC_Lib.h	ヘッダファイル

DCモータ制御**API**ライブラリの使い方

ライブラリのインクルードについては、クイックスタートガイドを参照してください。

DCモータ制御APIライブラリをArduinoで利用する場合、以下のようにスケッチの冒頭でDCモータ制御APIライブラリのヘッダファイルをインクルードし、使用するクラスのインスタンス化を行います。

GUIツールを使用する場合は、別途シリアル通信APIを呼び出す必要があります。詳細は、API関数仕様を参照してください。

また使用上の注意点として、DCモータ制御APIライブラリでは
ATmega32U4(Arduino Microマイクロコントローラ)のタイマレジスタ・TCCR0を使用しており、Arduino Microのデフォルトライブラリ関数に影響があります。(備考1)

DCモータ制御APIライブラリをインクルードしたスケッチを以下に示します。

DCモータ制御APIライブラリのインクルード

```
#include <LV8548_DC_Lib.h>
//LV8548DC用API関数を使用するためのヘッダファイルの取り込み
Lib_LV8548DC  Lib;        //LV8548DCクラスのインスタンス化(備考2)

void setup() {            //スタート時に呼び出される関数(setup()とloop()関数 参照)
}
void loop() {             //繰り返し実行される関数(setup()とloop()関数 参照)
}
```

1. TCCR0の分周比を変更する場合、下記関数の動作に影響します。
   分周費に応じた計算処理を追加する、または同等関数を自作して使用してください。
   tone()、analogRead()、micros()、millis()、delayMicroseconds()、delay()
2. 今回の例では「Lib」という名前でインスタンス化しているため、接頭に「Lib.」をつけることでLib_LV8548DCクラスのAPI関数を呼び出し可能になります。**例**：Lib.initLib();

スケッチのコンパイル、書き込み

スケッチ(Arduinoプログラム)のコンパイル、書き込み手順については、クイックスタートガイド「Arduinoプログラムのコンパイル・Arduinoへの書き込み」を参照してください。

プログラム（スケッチ）のコーディング

自動コード生成機能詳細

自動コード生成機能で出力したスケッチの一例を用いて、スケッチを出力する際に自動で記述する関数等の役割を説明します。

GUIデバッグ操作によって記述されるDCモータ制御APIライブラリの各API関数詳細は、 API関数仕様を参照してください。

自動生成コード例

```
#include <LV8548_DC_Lib.h>  //DCモータ制御APIライブラリの使い方 参照
Lib_LV8548DC  Lib;          //DCモータ制御APIライブラリの使い方 参照
void setup() {              //setup()とloop()関数 参照
  Serial.begin(19200);      //Baud rateを19200に設定してポートを開く(3)
  Lib.initLib();            //Arduinoパラメータとレジスタの初期化
  delay(5000);              //Arduino起動後のインターバル時間[ms](4)
  Lib.setPWMFrequency(1);
  delay(0);                 //API関数実行後のインターバル時間[ms]
  Lib.setRotation(0, 0);
  delay(0);                 //API関数実行後のインターバル時間[ms]
  Lib.setCtlVoltage(0, 20);
  delay(0);                 //API関数実行後のインターバル時間[ms]
  Lib.setStartFlag(0, 1);
  delay(0);                 //モータ駆動時間[ms] (5)
  Lib.setStartFlag(0, 0);
  delay(0);                 //API関数実行後のインターバル時間[ms]
}
void loop() {               //setup()とloop()関数 参照
}
```

3. シリアル通信を使用する場合に必要な関数です。
   使用しない場合は削除しても動作に影響はありません。
4. デフォルト5000として設定されます。
   Arduino起動後のインターバル時間の実時間はPWM周波数設定により変動し、0.977 kHzを設定した場合5秒となります。PWM周波数とインターバル時間の関係についてはクイックスタートガイド 補足資料の「delay関数について」 を参照してください。
5. モータスタート後のdelay()はモータの駆動時間になります。
   デフォルト0として設定されるため、変更してください。

自動生成されたスケッチの利用

自動生成されたスケッチは、プログラミング初心者でも使いやすいようシンプルな構成になっています。これをカスタマイズすることで、より実践的なプログラムになります。

今回は、アプリケーション適用例 で生成したスケッチのカスタマイズ例として、Arduinoセットアップ部分とモータ制御部を関数化し、それぞれsetup()とloop()で呼び出すスケッチを紹介します。

```
#include <LV8548_DC_Lib.h>
Lib_LV8548DC  Lib;
void setup() {
   motorSetup(); //関数化したArduinoの初期設定をsetup()で呼び出します。
}
void loop() {
   motorControl(); //関数化したモータ制御部をloop()で呼び出します。
}
//Arduinoの初期設定を関数化します。//
void motorSetup() {
   Serial.begin(19200);
   Lib.initLib();
   delay(5000);
}
//モータ制御部を関数化します。//
void motorControl() {
   Lib.setPWMFrequency(1);
   Lib.setRotation(0, 0);
   Lib.setCtlVoltage(0, 20);
   Lib.setStartFlag(0, 1);
   delay(5000);            //delayを5000に変更し、モータ1駆動時間を5000[ms]に設定します。
   Lib.setStartFlag(0, 0);
}
```

アプリケーション適用例

概要

本キットを使用して、DC ブラシモータ（おもちゃの電車）を Bluetooth で操作するアプリケーションを開発します。DC ブラシモータは本キットの LV8548モータドライバを用いてPWM で駆動され、Arduino は Bluetooth モジュールRN-42(*)を通してシリアル通信でコマンドを受信し、モータの駆動を制御します。

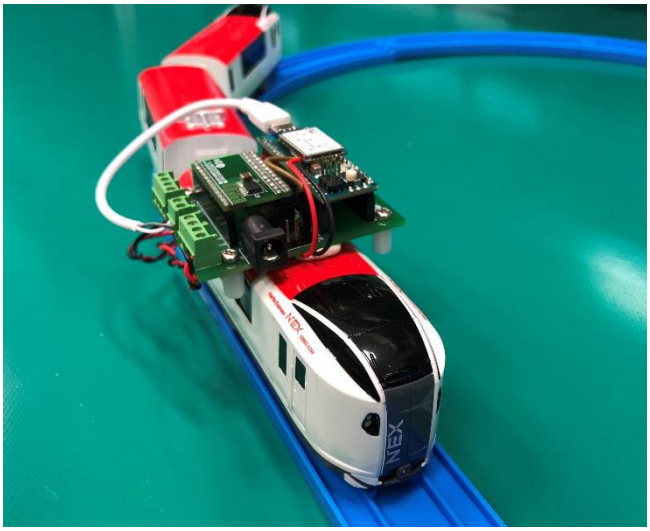


図 5. おもちゃの電車アプリ

必要なもの

- PC（Bluetooth 2.1 対応ノートパソコンを想定、非対応の場合はドングルを接続）
- Bluetooth モジュールRN−42(*)
- おもちゃの電車（DCブラシモータ駆動）
- 電池（充電式ニッケル水素電池(Ni-MH) 1.2V x4）

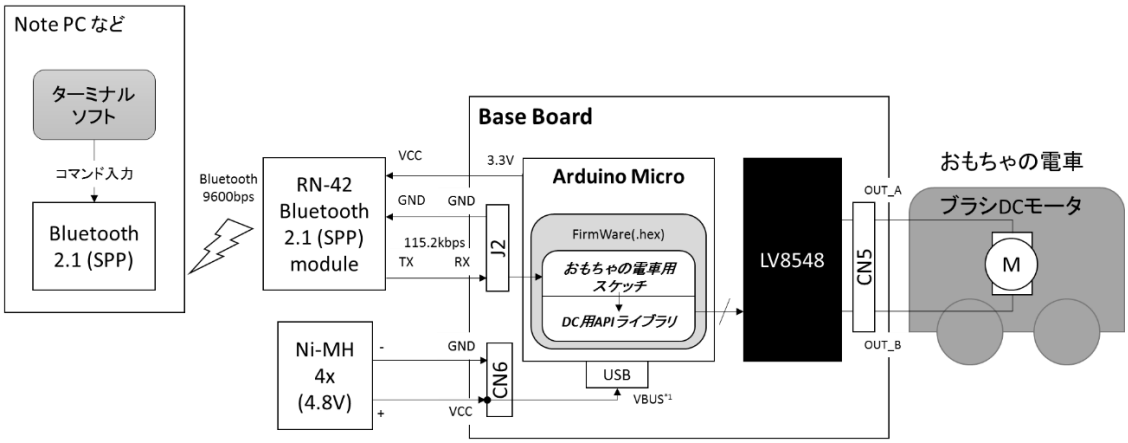
*Microchip 社製のBluetooth 2.1 + EDR モジュール。Digikeyで取り扱われています。
Microchip 社の製品紹介ページ
http://www.microchip.com/wwwproducts/en/RN42

仕様

- PC とおもちゃの電車は Bluetooth 2.1 (SPP ※2) で通信する
- 電源は充電式ニッケル水素電池（4本直列で約5 V）から供給
- PC からコマンド（キー操作）でおもちゃの電車を操作できる
- 操作コマンドは r: 発車、s: 停車、+: 加速、−: 減速、b: ブレーキ
- 加速、減速は PWM の duty （ステップ 5%）を変更することにより実現する
- PWM は Forward<−>Open 駆動にて実現する

*SPP は Bluetooth モジュールをシリアルポートと同様に扱えるプロファイルです。RN-42 をペアリングすると RN-42 の UART_TX/RX 端子は PC からシリアルポートとして認識されます。

ブロック図



*1 VBUS は USB の +5V に接続することを意味します

図 6. ブロック図

接続について

- ベースボードのOUT_A/OUT_B端子をおもちゃの電車のブラシDCの端子に接続します。
- Bluetooth モジュール RN−42 の 電源はVDD をベースボードの 3.3 V およびGNDから供給します。TX(送信) をベースボードの J2 の RX(受信) に接続します。またRN−42は3−6V入出力トレラントなので J2 の TX を RN−42 の RX に繋いでも問題ありません。
- ベースボードの電源は充電式ニッケル水素電池（4本直列で約5V）から供給します。CN6のGNDおよびUSBのGNDに電池のマイナスを、CN6 の B VCCおよびUSBのVBUSに電池のプラスを接続してください。
- PC が Bluetooth 2.1 に対応していない場合はドングルをつけて対応してください。

RN−42 Bluetooth モジュールについて

RN−42 は工場出荷状態で、デフォルトで SPP スレーブになっています。通信速度はデフォルトでPC 側は 9600bps, Arduino に接続するUART_TX/RX 端子は 115.2 Kbps に設定されています。ここでは工場出荷状態で使うことを想定しています。

コード

```
#include <LV8548_DC_Lib.h>
Lib_LV8548DC Lib;

int duty = 0; // duty のグローバル変数。初期値0。

void setup()
{
 // J2 端子の TX/RX は Serial1 で制御できるので Serial1 を初期化する
 Serial1.begin(115200);
Lib.initLib();
 delay(5000);
 // ドライバの初期化は適当なコードを GUI から生成して使ってください
 Lib.setPWMFrequency(2);
 Lib.setRotation(0, 0);
 Lib.setCtlVoltage(0, duty);
 Lib.setStartFlag(0, 1);
}

void loop()
{
 // J2 端子の TX/RX 操作にはライブラリ関数 guiSerialRead() は使えないので
 // ループで Serial1 の入力を直接監視します（ポーリングといいます）

 if (Serial1.available() > 0) { // シリアルバッファにデータがあった場合
  int rd = Serial1.read(); // シリアルデータを読みます
  switch (rd) { // 送られてきた文字（キー）を判定します
   case 'r': // 発車
    Lib.setStartFlag(0, 1); // モータの駆動を開始します
    break;
   case 's': // 停車 (Open で停車します)
    Lib.setStartFlag(0, 0);
    break;
   case 'b': // ブレーキ
    Lib.setStartFlag(0, 2); // ブレーキで停車します
    break;
   case '+': // 加速
    if (duty < 100) {
duty += 5; // duty を 5%ずつ増やします
     Lib.setCtlVoltage(0, duty); // duty の変更
    }
    break;
   case '−': // 減速
    if (duty > 0) {
duty −= 5; // duty を 5%ずつ減らします
     Lib.setCtlVoltage(0, duty); // duty の変更
    }
    break;
  }
 }
}
```

設定および操作について

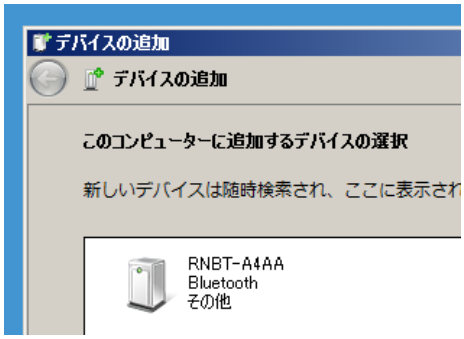
ターミナルソフトのインストール

Arduino IDE のシリアルモニタはシリアル送信毎に送信ボタンを押す必要があり、本アプリケーションには適しません。

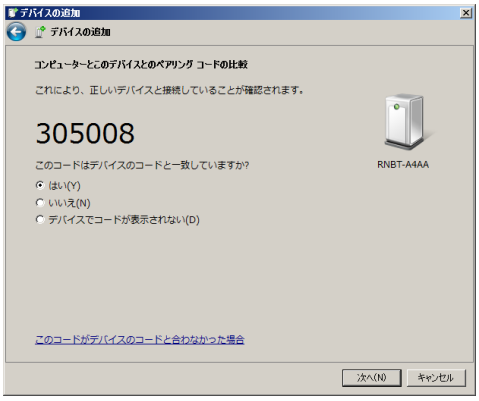
別途ターミナルソフト (TeraTerm https://ja.osdn.net/projects/ttssh2/) 等をインストールしてください。上記のTeraTermであれば、キーを押すとすぐにシリアルが送信され、操作が効率良く行えます。

Bluetoothのペアリング

Bluetooth 2.1 が使える PC を用意してRN−42とペアリングをします。スタートメニューから、デバイスとプリンターを選び、デバイスの追加をクリックすると、以下のようなデバイスとして RN−42 が見つかります。



このデバイスをダブルクリックすると



ペアリングの確認が行われます。「このコードはデバイスのコードと一致していますか？」で

「はい」を選択し、「次へ」をクリックするとデバイスが追加されます。

ターミナルソフトの設定

ペアリング後に、ターミナルソフトを立ち上げて、ペアリングしたデバイスに接続します。工場出荷状態だと 9600bps, パリティなし, スタートビット1, ストップビット1になります。エコーバックは必須ではありませんが、ローカルエコーを有効にすることを推奨します。

操作

コード例ではキーボードからの一文字コマンドによって各種動作を行います。

r: 発車 s: 停車 b: ブレーキ +: 速度アップ -: 速度ダウン

次のステップ

本アプリケーション適用例を参考に、以下のような、さらに高度なアプリケーション開発にチャレンジしてみてください。

1. 発車時は徐々に速度を上げ、停車時は徐々に速度を落とす制御を適用してみてください。
2. Bluetooth を介したターミナルソフトからの入力の代わりに、ベースボードCN8 の A2−5 の端子に、ボリュームなどの可変抵抗を利用したアナログ入力や、タクトスイッチなどを利用したデジタル信号入力を行い、PWMの duty をコントロールできるようにしてみてください。

API関数仕様

DCモータ制御API 概要

本章では、LV8548モータドライバ向けArduino Micro DCモータ制御APIの概要について記述します。

概要

本APIは、Arduino Microからオン・セミコンダクター社製 モータドライバLV8548を使用して、DCモータを制御するためのライブラリを提供します。ユーザーは、本APIライブラリをArduino IDEからインクルードし、目的に沿ったAPI関数をスケッチに記述することで容易にLV8548を使用したDCモータ制御が可能となります。

ただし、本APIではTCCR0を使用しており、制限事項があるため注意が必要です。(特記事項参照)

ライブラリファイル構成

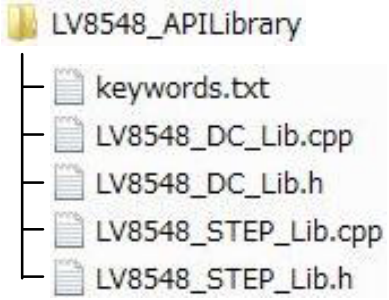


表 2. ライブラリファイル一覧

#	ファイル名	内容
1	keywords.txt	キーワードファイル(スケッチ内の表示色変更対象ワードの設定)
2	LV8548_DC_Lib.cpp	ソースファイル
3	LV8548_DC_Lib.h	ヘッダファイル

Arduino Micro/LV8548 Base Board ピンアサイン

Arduino Micro/LV8548 Base Board ピンアサインを以下に示します。

Arduino Micro入出力ピンの白背景色は、ユーザーが利用可能なリソースを表します。

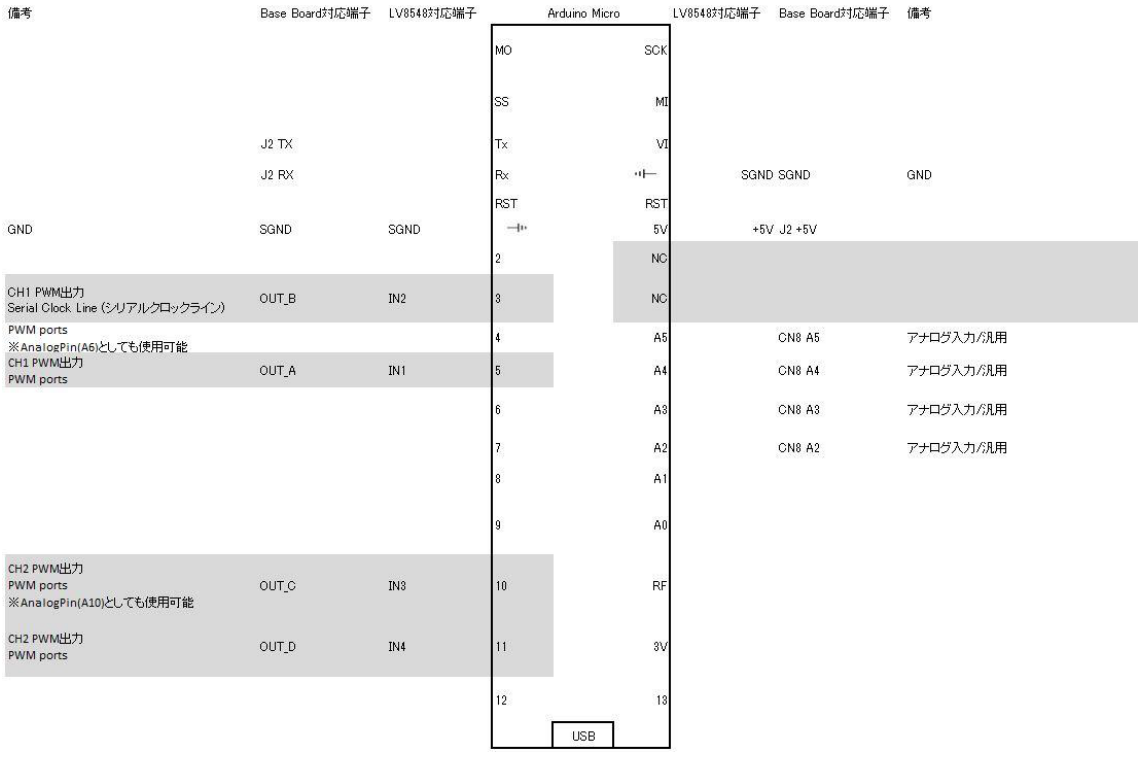


図 7. Arduino Micro/LV8548 Base Board ピンアサイン

DCモータ制御API 初期設定

本章では、DCモータ制御APIで行う初期設定について記述します。

API で使用するリソース

本APIでは表2-3-3に示すArduino ピンおよび対応するATmega32U4のタイマレジスタを使用しており、ユーザーはこれを利用することができません。レジスタ詳細についてはATmega16U4-32U4_Datasheet.pdfを参照してください。

表 3. Arduino Micro ピンと対応するATmega32U4タイマレジスタ

#	Arduinoピン	タイマレジスタ	概要	章番号
1	D3, IO11	TCCR0B	タイマ/カウンタ0制御レジスタB	TCCR0B
2	IO9, IO10	TCCR1B	タイマ/カウンタ1制御レジスタB	TCCR1B
3	D5	TCCR3B	タイマ/カウンタ3制御レジスタB	TCCR3B
4	D6, IO13	TCCR4B(*)	タイマ/カウンタ4制御レジスタB	TCCR4B

*initLib関数によって初期化を行う場合、TCCR0B, TCCR1B, TCCR3Bのみが使用され、TCCR4Bは使用されません。

初期設定

各パラメータ・各タイマレジスタ・各入出力ピンの初期化は、initLib関数(initLib参照)もしくはinitLibPortSet関数(initLibPortSet参照)によって行われます。本APIライブラリを使用する際は、必ずinitLib関数、もしくはinitLibPortSet関数のいずれかをスケッチのsetup()内で呼び出し、各パラメータ・各タイマレジスタ・各入出力ピンの初期化を行ってください。

表 4. 初期化内容一覧

#	初期化対象	初期設定値
1	各タイマレジスタ設定	タイマレジスタ設定詳細を参照
2	各入出力ピン	initLib initLibPortSetを参照
3	各パラメータ設定	内部パラメータ一覧を参照

API関数仕様

API関数概要

LV8548モータドライバ ステッパモータ制御API関数を以下に示します。

表 5.

#	関数名	概要	章番号
1	initLib	− レジスタ設定(PWM(*1)出力ピン固定) − 各パラメータデフォルト設定	initLib
2	initLibPortSet	− レジスタ設定(PWM(6)出力ピン選択式) − 各パラメータデフォルト設定	initLibPortSet
3	setCtlVoltage	− PWM Dutyの設定	setCtlVoltage
4	setPWMFrequency	− PWM周波数の設定	setPWMFrequency
5	setRotation	− 回転方向とPWM駆動方法の設定	setRotation
6	setStartFlag	− モータ ON/OFFの設定 (Start, Brake, Openのいずれかのステートを設定)	setStartFlag
7	readAdc	− アナログ電圧値の測定	readAdc
8	readModule	− A0ピンアナログ電圧値の取得(機種判別用)	readModule
9	setDelay	− Timer0分周比に合わせた専用delay関数	setDelay
10	guiSerialRead	− Serial受信データのバッファ管理	guiSerialRead
11	guiSerialParse	− GUIからのメッセージの解析と実行	guiSerialParse

6. 第5章ハードウエア仕様「PWMとは」を参照してください。

API関数詳細

initLib

表 6. initLib

API関数名	initLib()		
Class	Lib_LV8548_DC		
属性	Public		
引数	型	変数	内容
	void	なし	なし
戻り値	型		内容
	int		0: “成功” 1: “失敗”(初期化失敗時)
処理概要	1. Arduino(ATmega32u4)タイマレジスタの設定 – TCCR0初期設定(TCCR0B参照) – TCCR1初期設定(TCCR1B参照) – TCCR3初期設定(TCCR3B参照) 2. PWM出力ピンの設定 – IN1をD5に設定 – IN2をD3に設定 – IN3をIO10に設定 – IN4をIO11に設定 3. 各パラメータデフォルト設定。 – 各パラメータ初期値設定失敗時は失敗(1)を返します。(各関数の戻り値が成功(0)でない場合)		
使用例 (Sketch)	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { **Lib.initLib();** // 初期設定 }		

initLibPortSet

表 7. initLibPortSet

API関数名	initLibPortSet(*byte* in1, *byte* in2, *byte* in3, *byte* in4)		
Class	Lib_LV8548_DC		
属性	Public		
引数	**型**	**変数**	**内容**
	byte	in1	IN1に設定するポート
	byte	in2	IN2に設定するポート
	byte	in3	IN3に設定するポート
	byte	in4	IN4に設定するポート
戻り値	**型**		**内容**
	Int		0: “成功” 1: “失敗”
処理概要	1. Arduino(ATmega32u4)タイマレジスタの設定 入力されたピン番号に応じて、タイマレジスタを初期化します。 2. PWM出力ピンの設定 入力されたピン番号に応じて、PWM出力ピンを設定します。 3. 各パラメータデフォルト設定 – 存在しないピン番号、PWM制御対象外のピン番号を入力された場合は失敗(1)を返します。 – 各パラメータ初期値設定失敗時は失敗(1)を返します。(各関数の戻り値が成功(0)でない場合)		
使用例 (sketch)	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { **Lib.initLibPortSet(3,5,10,11);** // 初期設定3ピンをIN1,5ピンをIN2,10ピンをIN3,11ピンをIN4にセット }		

設定可能ピンと対応するタイマレジスタ

設定可能ピン	対応するタイマレジスタ
D3	TCCR0
D5	TCCR3
D6	TCCR4
IO9	TCCR1
IO10	TCCR1
IO11	TCCR0
IO13	TCCR4

setCtlVoltage

表 8. setCtlVoltage

API関数名	setCtlVoltage(*byte* motorNo, *byte* duty)		
Class	Lib_LV8548_DC		
属性	Public		
引数	**型**	**変数**	**内容**
	byte	motorNo	モータの番号 0: Ch1(OUT_A, OUT_B) 1: Ch2(OUT_C, OUT_D) ピンアサインは図1参照
	byte	Duty	出力比率(Duty) [値域 0 - 100] 0:出力0%, 100:出力100%
戻り値	**型**		**内容**
	Int		0: “成功”, 1: “失敗”
処理概要	- 引数motorNoで指定したモータに引数dutyのPWM Duty をセットし、モータの回転制御を実行します。(回転中のPWM Duty変更可) - 引数motorNoが値域外の場合は失敗(1)を返します。 - 引数dutyが値域外の場合は失敗(1)を返します。		
使用例 (Sketch)	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Lib.initLib(); // 初期設定 } void loop() { **Lib.setCtlVoltage(0,50);** // Motor1 duty 50% 設定 }		

setPWMFrequency

表 9. setPWMFrequency

API関数名	setPWMFrequency(*byte* Hz)		
Class	Lib_LV8548_DC		
属性	Public		
引数	型	変数	内容
	byte	Hz	PWM周波数 [値域 0 – 3] 0: 7.813 kHz 1: 0.977 kHz 2: 0.244 kHz 3: 0.061 kHz
戻り値	型		内容
	Int		0: “成功”, 1: “失敗”
処理概要	− 引数hzに応じてタイマレジスタ(TCCRxB) の分周比を変更することで、指定のPWM周波数に変更します。(回転中のPWM周波数変更可) − 引数hzが値域外の場合は失敗(1)を返します。		
使用例 (Sketch)	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Lib.initLib(); // 初期設定 } void loop() { **Lib.setPWMFrequency(1);** // 周波数0.977 kHzをセット }		

setRotation

表 10. setRotation

API関数名	setRotation(*byte* motorNo, *byte* select)		
Class	Lib_LV8548_DC		
属性	Public		
引数	**型**	**変数**	**内容**
	byte	motorNo	モータの番号 0: Ch1(OUT_A, OUT_B) 1: Ch2(OUT_C, OUT_D) ピンアサインは図1-1参照
	byte	select	PWMモード [値域 0 - 5] 出力パターン一覧参照
戻り値	**型**		**内容**
	int		0: “成功”, 1: “失敗”
処理概要	- 引数motorNoで指定したモータに引数selectで指定したPWMモードをセットし、 モータの回転制御を実行します。(回転中のPWMモード変更可) - 引数motorNoが値域外の場合は失敗(1)を返します。 - 引数selectが値域外の場合は失敗(1)を返します。		
使用例 (Sketch)	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Lib.initLib(); // 初期設定 } void loop() { **Lib.setRotation(1,0);** // モータ2を正転(Forward&Brake) 設定 }		

出力パターン一覧

PWMモード	**引数**	**IN1(IN3)**	**IN2(IN4)**
正転: Forward↔Brake	0	High	PWM(逆相)
正転: Forward↔Open(Standby)	1	PWM	Low
逆転: Reverse↔Brake	2	PWM(逆相)	High
逆転: Reverse↔Open(Standby)	3	Low	PWM
StandBy	4	Low	Low
Brake	5	High	High

*Forward Reverse切り替え時は、急激な回転方向切り替えによる故障を回避するため数m～数百msのStandby区間もしくはBrake区間を挟んでください。

setStartFlag

表 11. **setStartFlag**

API関数名	setStartFlag(*byte* motorNo, *byte* select)		
Class	Lib_LV8548_DC		
属性	Public		
引数	型	変数	内容
	byte	motorNo	モータの番号(Ch1:0,Ch2:1) 0: Ch1(OUT_A, OUT_B) 1: Ch2(OUT_C, OUT_D) ピンアサインは図1参照
	byte	select	0: Open(待機で停止) 1: Start(回転開始) 2: Brake(ブレーキで停止)
戻り値	型		内容
	int		0: “成功” 1: “失敗”
処理概要	− 引数motorNoで指定したモータの回転状態を引数select で指定した状態にセットします。(回転中の変更可) − 引数motorNoが値域外の場合は失敗(1)を返します。 − 引数selectが値域外の場合は失敗(1)を返します。		
使用例 **(Sketch)**	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Lib.initLib(); // 初期設定 } void loop() { **Lib. setStartFlag (0,1);** //モータ1を回転状態にセット }		

readAdc

表 12. **readAdc**

API関数名	readAdc(*byte pin*)		
Class	Lib_LV8548_DC		
属性	public		
引数	型	変数	内容
	byte	pin	ピン番号 1: A1 2: A2 3: A3 4: A4 5: A5
戻り値	型		内容
	int		成功…指定したピンのアナログ電圧値 失敗…地域外の入力をした場合
処理概要	① 引数値域チェック ② 引数のピンに対してanalogRead()を実行し、取得した値を返す		
使用例 **(sketch抜粋)**	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Lib.initLib(); // 各パラメータ初期化 } void loop(){ int value; value = Lib.readAdc (1); // 1ピンを読み込む }		

readModule

表 13. **readModule**

API関数名	readModule()		
Class	Lib_LV8548_DC		
属性	public		
引数	型	変数	内容
	void	なし	なし
戻り値	型		内容
	int		AnalogRead()
処理概要	① A0ピンにanalogRead()を実行し、取得した値を返す		
使用例 **(sketch抜粋)**	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Lib.initLib(); // 各パラメータ初期化 } void loop(){ int value; value = Lib.readModule (); // A0ピンのアナログ電圧値を読み込む }		

setDelay

表 14. **setDelay**

API関数名	setDelay(*uint32_t* msec)		
Class	Lib_LV8548_DC		
属性	public		
引数	型	変数	内容
	uint32_t	msec	待機時間
戻り値	型		内容
	int		0: 成功
処理概要	① 引数(msec)の時間、待機する。		
使用例 (sketch抜粋)	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Lib.initLib(); // 各パラメータ初期化 Lib.setDelay(1000); // 1秒待機 } void loop() { }		

guiSerialRead

表 15. **guiSerialRead**

API関数名	guiSerialRead()		
Class	Lib_LV8548_DC		
属性	Public		
引数	型	変数	内容
	void	なし	なし
戻り値	型		内容
	void		なし
処理概要	− シリアル受信データのバッファリング管理を行います。 − guiSerialParse をコールして受信データの解析と実行を行います。 − 3秒間シリアル受信がない場合はモータを停止します。(全出力OFFでトルクを失う)		
使用例 (Sketch)	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Serial.begin(19200); // Baud rate 19200でポートを開く Lib.initLib(); // 初期設定 } void loop() { **Lib. guiSerialRead** (); //シリアルメッセージの受信 }		

guiSerialParse

表 16. guiSerialParse

API関数名	guiSerialParse(*char* *type)		
Class	Lib_LV8548_DC		
属性	Public		
引数	**型**	**変数**	**内容**
	char*	*type	GUIとのSerial通信の解析 [値域 0x03: getId識別子 0x04: timeoutPol識別子 0x41: setCtlVoltage 識別子 [0x00−0x01] (Ch1,Ch2) [0x00−0x64] (0%~100%(Duty)) 0x44: setRotation 識別子 [0x00−0x01] (Ch1,Ch2) [0x00−0x05] (PWMモード0−5) 0x67: setPWMFrequency識別子 [0x00−0x03] (PWM周波数の設定) 0x68: setStartFlag識別子 [0x00−0x01] (Ch1,Ch2) [0x00−0x02] (Open, Start, Brake)
戻り値	**型**	**内容**	
	int	0: “成功”, 1: “失敗”	
処理概要	− GUI からのメッセージの解析、実行を行います。 − 引数が値域外の場合は失敗(1)を返します。		
使用例1 (Sketch)	Lib_LV8548_DC Lib; // Lib_LV8548_DC class定義 void setup() { Lib.initLib(); // 初期設定 } void loop() { **Lib.guiSerialRead();** //guiSerialRead関数処理でguiSerialParse処理呼ぶ。 }		

表 16. **guiSerialParse**

使用例2 (Sketch)	// ユーザーは、guiSerialParse() をオーバーライドすることによって、 シリアルインターフェースをカスタマイズして利用できます。// //Lib_LV8548DCクラスを継承して派生クラスの作成// class Lib_LV8548DC_Ex : public Lib_LV8548DC { public: virtual ~Lib_LV8548DC_Ex() {} int guiSerialParse(char *type) override; }; Lib_LV8548DC_Ex Ex; //継承クラスのインスタンス化 //guiSerialParse関数をオーバーライド// int Lib_LV8548DC_Ex::guiSerialParse(char *serialRecvStr) { switch (serialRecvStr[0]) { case 'a': { Serial.println("Command"); return SUCCESS; //成功(0)を返します。 } default: { Serial.print("unknown command"); } } return FAILURE; //失敗(1)を返します。 } void setup() { Ex.initLib();// 初期設定 Serial.begin(19200); // Baud rate 19200でポートを開く } void loop() { Ex.guiSerialRead();//guiSerialRead関数処理でguiSerialParse処理を呼ぶ。 }

シリアルインターフェース仕様

本章では、PCとArduino MicroをUSB接続した際のシリアルインターフェースについて記述します。

Arduino側プログラム（スケッチ）に、シリアル通信設定、並びにguiSerialRead関数を実装し、PCからシリアル通信を介して各APIに対応したメッセージ送信することで、LV8548を使用したDCモータ制御が可能です。

guiSerialRead関数実装方法は、guiSerialRead を参照してください。

表 17. メッセージ一覧

#	コマンド	対応API	コマンド概要	Length	章番号
1	0x03	None	APIライブラリ識別用IDを取得します。	1 byte	getId
2	0x04	timeoutPole	一定間隔で送信されるシリアル接続監視用のコマンドです。	1 byte	timeoutPole
3	0x05	initLib	対応APIをCallし，APIの各設定を初期化します	1 byte	initLib
4	0x41	setCtlVoltage	setCtlVoltageをCallし、 モータ1,2に対して出力比率(duty)[%]を設定します。	3 byte	setCtlVoltage
5	0x67	setPWMFrequency	setPWMFrequencyをCallし、 PWMの周波数を設定します。	2 byte	setPWMFrequency
6	0x44	setRotation	setRotationをCallし、 モータ1,2に対して回転方向とPWM駆動方法を設定します。	3 byte	setRotation
7	0x68	setStartFlag	setStartFlagをCallし、 モータ1, 2に対してStart, Brake, Openのいずれかの状態を通知します。	3 byte	setStartFlag
8	0x64	readAdc	対応APIをCall し、指定したアナログ入力ピンの電圧を読み出します	3 byte	readAdc
9	0x48	readModule	対応APIをCallし、Moduleを識別します	3 byte	readModule

メッセージ構成詳細

GUIからArduino Microに送信されるメッセージ構成の詳細を記述します。

getId

- コマンド概要
  APIライブラリ識別用IDを取得するためのコマンドです。
  APIはこのコマンドを受信すると対応モータドライバ名、APIバージョンなどが識別できるIDを返します。

Command from GUI to Motor Driver Kit

Byte	0
Field	CMD
Value	0x03

timeoutPole

- コマンド概要
  Arduino Microのシリアル接続状態を監視するためのコマンドです。
  GUIはこのコマンドを1秒ごとにArduino Microに送信します。APIはこのコマンドを含めて3秒間のシリアル受信データがない場合、フェイルセーフのためtimeoutPole関数がCallされ、自動的にモータを停止します。

Command from GUI to Motor Driver Kit

Byte	0
Field	CMD
Value	0x04

initLib

- コマンド概要
  APIの各設定を初期化するためのコマンドです。
GUIはGUIからシリアル切断を行う際にこのコマンドをArduino Microに送信し、initLib関数をCallします。
initLib関数の詳細はinitLibを参照してください。

Command from GUI to Motor Driver Kit

Byte	0
Field	CMD
Value	0x05

setCtlVoltage

- コマンド概要
  setCtlVoltage関数を呼び出し、モータ1(OUT_A, OUT_B), モータ2(OUT_C, OUT_D)に対して出力比率(duty)[%]を設定するためのコマンドです。
  setCtlVoltage関数の詳細はsetCtlVoltageを参照してください。

Command from GUI to Motor Driver Kit

Byte	0	1	2
Field	CMD	CH	STEP ANGLE
Value	0x41	0x00−0x01	0x00−0x64

Field CH: モータ番号

Field DUTY: PWM Duty

setPWMFrequency

- コマンド概要
  setPWMFrequency関数を呼び出し、PWM周波数を設定するためのコマンドです。
  setPWMFrequency関数の詳細はsetPWMFrequencyを参照してください。

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	FREQ
Value	0x67	0x00−0x03

Field FREQ: PWM周波数

setRotation

- コマンド概要
  モータ1(OUT_A, OUT_B), モータ2(OUT_C, OUT_D)に対して回転方向とPWM駆動方法を設定するためのコマンドです。
  setRotation関数の詳細はsetRotationを参照してください。

Command from GUI to Motor Driver Kit

Byte	0	1	2
内容	CMD	CH	MODE
Value	0x44	0x00−0x01	0x00−0x05

Field CH: モータ番号

Field MODE: 回転方向とPWM駆動方法

setStartFlag

- コマンド概要
  モータ1(OUT_A, OUT_B), モータ2(OUT_C, OUT_D)に対してStart, Brake, Openのいずれかの状態を通知するためのコマンドです。
  setStartFlag関数の詳細はsetStartFlagを参照してください。

Command from GUI to Motor Driver Kit

Byte	0	1	2
内容	CMD	CH	FLAG
Value	0x68	0x00−0x01	0x00−0x02

Field CH: モータ番号

Field FLAG: モータ ON/OFFの設定

readAdc

- コマンド概要
  readAdc関数を呼び出し、指定したアナログピンの電圧測定を行うためのコマンドです。
  readAdc関数をCallし、読み出したアナログ電圧値をCMD 0x62と共に送信側に返します。
  readAdc関数の詳細はreadAdcを参照してください。

Command from GUI to Motor Driver Kit

Byte	0	1
Field	CMD	CH
Value	0x64	0x01−0x05

Field CH: アナログピン(A1~A5)

Command from Motor Driver Kit to GUI

Byte	0	1 （下位）	2 （上位）
Field	CMD	RECVADC	
Value	0x62	0x0000−0x03FF	

Field RECVADC: アナログ電圧値(0~1023)

readModule

- コマンド概要
  readModule関数を呼び出し、Arduinoと接続されているモジュールを識別するためのコマンドです。
  guiSerialParse関数は、readModule関数をCallし、読み出したアナログ電圧値をCMD 0x49と共に送信側に返します。
  readModule関数の詳細は readModule を参照してください。

Command from GUI to Motor Driver Kit

Byte	0
Field	CMD
Value	0x48

Command from Motor Driver Kit to GUI

Byte	0	1 （下位）	2 （上位）
Field	CMD	RECVMODULE	
Value	0x49	0x0000−0x03FF	

Field RECVMODULE: 機種別アナログ電圧値(0~1023)

タイマレジスタ設定詳細

TCCR0B 設定

TCCR0Bは8bit カウンタ/タイマ0用レジスタです。
各bit以下の表の内容を示します。

表 18.

Bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W
初期値 Arduino Micro	0	0	0	0	0	0	1	1

LV8548モータドライバ対応では IO11 ピン、D3ピンで用いるPWM周波数を初期立ち上げ時に7812.5 Hzとしています。(APIにより途中変更可能)

[Fast PWM周波数 算出計算式]
(クロック周波数 /分周比)×(1/カウンタ数）[Hz]
= {(16.0×10^6 / 8)×(1/256) } / 10^3 = 7812.5 [Hz]

分周比は下記表のようにTCCR0BのCS（Clock Selector）3ビット組み合わせで決定されるため、(CS02,CS01,CS00) = (0, 1, 0) を設定します。

表 19.

CS02	CS01	CS00	分周（比）
0	0	0	タイマ/カウンタ 動作停止
0	0	1	1/1 (前置分周なし)
0	1	0	1/8
0	1	1	1/64
1	0	0	1/256
1	0	1	1/1024
1	1	0	外部クロック
1	1	1	外部クロック

当該レジスタの設定は、initLib関数(initLib参照)、もしくはinitLibPortSet関数(initLibPortSet参照)を、当該レジスタ対応ピン(表3参照)であるD3ピンまたはIO11ピンを指定して呼び出す場合に、初期設定の一環として実施されます。なお、initLibPortSet関数を呼び出す場合に、IO11 ピン、D3ピンが指定されなければ実施されません。

TCCR1B 設定

TCCR1Bは8bit カウンタ/タイマ1用レジスタです。
各bit以下の表の内容を示します。

表 20.

Bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初期値 Arduino Micro	0	0	0	0	0	0	0	0

LV8548モータドライバ対応ではIO10 ピン、IO9ピンで用いるPWM周波数を初期立ち上げ時に7812.5 Hzとしています。(APIにより途中変更可能)

なお、8bit位相基準PWMがデフォルトであるため、TCCR1A,TCCR1Bのビットを操作する必要があります。(後述)

[Fast PWM周波数 算出計算式]

(クロック周波数 /分周比)×(1/カウンタ数）[Hz]

= {(16.0×10^6 / 8)×(1/256) } / 10^3 = 7812.5 [Hz]

分周比は下記表のようにTCCR1BのCS（Clock Selector）3ビット組み合わせで決定されるため、(CS12,CS11,CS10) = (0, 1, 0) を設定します。

表 21.

CS12	CS11	CS10	分周（比）
0	0	0	タイマ/カウンタ 動作停止
0	0	1	1/1
0	1	0	1/8
0	1	1	1/64
1	0	0	1/256
1	0	1	1/1024
1	1	0	外部クロック使用
1	1	1	外部クロック使用

当該レジスタの設定は、initLib関数(initLib参照)、もしくはinitLibPortSet関数(initLibPortSet参照)を当該レジスタ対応ピン(表3参照)であるIO9ピンまたはIO10ピンを指定して呼び出す場合に、初期設定の一環として実施されます。なお、initLibPortSet関数を呼び出す場合に、IO9ピン、IO10ピンが指定されなければ実施されません。

TCCR3B 設定

TCCR3Bは8bit カウンタ/タイマ3用レジスタです。
各bit以下の表の内容を示します。

表 22.

Bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初期値 Arduino Micro	0	0	0	0	0	0	0	0

LV8548モータドライバ対応ではD5 ピンで用いるPWM周波数を初期立ち上げ時に7812.5 Hzとしています。(APIにより途中変更可能)

なお、8bit位相基準PWMがデフォルトであるため、TCCR1A,TCCR1Bのビットを操作する必要があります。(後述)

[Fast PWM周波数 算出計算式]

(クロック周波数 /分周比)×(1/カウンタ数）[Hz]

= {(16.0×10^6 / 8)×(1/256) } / 10^3 = 7812.5 [Hz]

分周比は下記表のようにTCCR3BのCS（Clock Selector）3ビット組み合わせで決定されるため、(CS32,CS31,CS30) = (0, 1, 0) を設定します。

表 23.

CS32	CS31	CS30	分周（比）
0	0	0	タイマ/カウンタ 動作停止
0	0	1	1/1
0	1	0	1/8
0	1	1	1/64
1	0	0	1/256
1	0	1	1/1024
1	1	0	外部クロック使用
1	1	1	外部クロック使用

当該レジスタの設定は、initLib関数(initLib参照)、もしくはinitLibPortSet関数(initLibPortSet参照)を当該レジスタ対応ピン(表3参照)であるD5ピンを指定して呼び出す場合に、初期設定の一環として実施されます。なお、initLibPortSet関数呼び出す場合に、D5ピンが指定されなければ実施されません。

TCCR4B 設定

TCCR4Bは10bit カウンタ/タイマ4用レジスタです。各bit以下の表の内容を示します。

表 24.

Bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	PWM4X	PSR4	DTPS41–	DTPS40	CS43	CS42	CS41	CS40
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
初期値 Arduino Micro	0	0	0	0	0	0	0	0

LV8548ドライバ対応ではD6 ピン,IO13ピンで用いるPWM周波数を初期立ち上げ時に7812.5 Hzとしています。(APIにより途中変更可能)

[Fast PWM周波数 算出計算式]

(クロック周波数 /分周比)×(1/カウンタ数）[Hz]

= {(16.0×10^6 / 8)×(1/1024) } / 10^3 = 7812.5 [Hz]

分周比は下記表のようにTCCR4BのCS（Clock Selector）4ビット組み合わせで決定されるため、(CS43,CS42,CS41,CS40) = (0,1,0,0) を設定します。

表 25.

CS43	CS42	CS41	CS40	分周（比）
0	0	0	0	タイマ/カウンタ 動作停止
0	0	0	1	1/1
0	0	1	0	1/2
0	0	1	1	1/4
0	1	0	0	1/8
0	1	0	1	1/16
0	1	1	0	1/32
0	1	1	1	1/64
1	0	0	0	1/128
1	0	0	1	1/256
1	0	1	0	1/512
1	0	1	1	1/1024
1	1	0	0	1/2048
1	1	0	1	1/4096
1	1	1	0	1/8192
1	1	1	1	1/16384

当該レジスタの設定は、initLibPortSet関数(initLibPortSet参照)を当該レジスタ対応ピン(表3参照)であるD6ピン、IO13ピンを指定して呼び出す場合に、初期設定の一環として実施されます。なお、initLibPortSet関数を呼び出す場合に、D6ピン、IO13ピンが指定されなければ実施されません。

IO9ピン IO10ピンを8bit高速PWM(※)に設定する

*:ATmega32u4に搭載される高周波数PWM波形生成モードIO9ピン IO10ピンを8bit高速PWMに設定するために、下記表のように設定を行います。

表 26. TCCR1B

bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
設定値 Arduino Micro	0	0	0	0	1	0	0	0

表 27. **TCCR1A**

bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	COM1A1	COM1A0	COM1B1–	COM1B0	COM1C1	COM1C0	WGM11	WGM10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
設定値 Arduino Micro	0	0	0	0	0	0	0	1

D5ピンを8bit高速PWM(※)に設定する

*:ATmega32u4に搭載される高周波数PWM波形生成モード

D5ピンを8bit高速PWMに設定するために、下記表のように設定を行います。

表 28. **TCCR3B**

bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
設定値 Arduino Micro	0	0	0	0	1	0	0	0

表 29. **TCCR3A**

bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	COM3A1	COM3A0	COM3B1–	COM3B0	COM3C1	COM3C0	WGM31	WGM30
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
設定値 Arduino Micro	0	0	0	0	0	0	0	1

D6ピン、IO13ピンを8bit高速PWM(※)に設定する

*:ATmega32u4に搭載される高周波数PWM波形生成モード

D6ピン、IO13ピンを8bit高速PWMに設定するために、下記表のように設定を行います。

表 30. **TCCR4B**

bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	PWM4X	PSR4	DTPS41	DTPS42	CS43	CS42	CS41	CS40
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
設定値 Arduino Micro	0	0	0	0	0	0	0	1

表 31. **TCCR4A**

bit	7(MSB)	6	5	4	3	2	1	0(LSB)
	COM4A1	COM4A0	COM4B1–	COM4B0	FOC4A	FOC4B	PWM4A	PWM4B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
設定値 Arduino Micro	0	0	0	0	0	0	1	1

特記事項

TCCR0変更による影響

当該LV8548モータドライバ制御APIでは、TCCR0の分周比を変更しており、Arduino Microのデフォルトライブラリ関数に影響があります。下記関数が正常動作しないため、必要に応じて自作する必要があります。tone()、analogRead()、 micros()、millis()、delayMicroseconds()、delay()

内部パラメータ/テーブル/関数 概要

内部パラメーター覧

内部パラメーター一覧を下記に示します。

表 32. 内部パラメーター覧

#	パラメータ名	初期値	内容	更新契機
PROTECTED				
1	motor1RotateMode	ROTATION_FW_BK(0)	モータ1側 PWMモード	setRotationで変更
2	motor2RotateMode	ROTATION_FW_BK(0)	モータ2側 PWMモード	setRotationで変更
3	motor1Duty	DUTY_MIN(0)	モータ1側Duty	setControlVoltageで変更
4	motor2Duty	DUTY_MIN(0)	モータ2側Duty	setControlVoltageで変更
5	motor1StartFlag	FLAG_OFF	モータ1側Start状態	setStartFlagで変更
6	motor2StartFlag	FLAG_OFF	モータ2側Start 状態	setStartFlagで変更
7	inPin[4]	−	PWM出力ピン	
PRIVATE				
8	DUTY_MIN	0	Duty[%] 最小値	固定値
9	DUTY_MAX	100	Duty[%] 最大値	固定値
10	FLAG_OFF	0	モータストップ	固定値
11	FLAG_ON	1	モータスタート	固定値
12	FLAG_BRAKE	2	モータブレーキ	固定値
13	ROTATION_FW_BK	1	Forward&Brake	固定値
14	ROTATION_FW_SB	0	Forward&Standby	固定値
15	ROTATION_RV_BK	3	Reverse&Brake	固定値
16	ROTATION_RV_SB	2	Reverse&Standby	固定値
17	ROTATION_SB_SB	4	Standby	固定値
18	ROTATION_BK_BK	5	Brake	固定値
19	MOTOR_NO1	0	モータ1	固定値
20	MOTOR_NO2	1	モータ2	固定値
21	PIN_NUMBER_MIN	1	ピン番号最小値(A0)	固定値
22	PIN_NUMBER_MAX	5	ピン番号最大値(A5)	固定値
23	SRMES_GET_ID	0x03	getId シリアルメッセージ 識別子	固定値
24	SRMES_POLLING_ID	0x04	timeoutPol シリアルメッセージ 識別子	固定値
25	SRMES_SET_INITIAL	0x05	initLib API対応シリアルメッセージ識別子	固定値
26	SRMES_CTL_VOLTAGE	0x41	setCtlVoltage API対応 シリアルメッセージ 識別子	固定値
27	SRMES_ROTATION	0x44	setRotation API対応 シリアルメッセージ 識別子	固定値

表 32. 内部パラメーター一覧 (continued)

#	パラメータ名	初期値	内容	更新契機
PRIVATE				
28	SRMES_PWM_FREQSET	0x67	setPWMFrequency API対応 シリアルメッセージ 識別子	固定値
29	SRMES_START_FLAG	0x68	setStartFlag API対応 シリアルメッセージ 識別子	固定値
30	SRMES_READ_MODULE	0x48	readModule API対応シリアルメッセージ識別子	固定値
31	SRMES_RES_READ_MODULE	0x49	readModule API 応答用対応シリアルメッセージ識別子	固定値
32	SRMES_RES_READ_ADC	0x62	readAdc API応答用対応シリアルメッセージ識別子	固定値
33	SRMES_READ_ADC	0x64	readAdc API対応シリアルメッセージ識別子	固定値
34	PWM_FREQ_DividingRatio_8	0	分周比1/8 PWM周波数7.813 kHz	固定値
35	PWM_FREQ_DividingRatio_64	1	分周比1/64 PWM周波数 0.977 kHz	固定値
36	PWM_FREQ_DividingRatio_256	2	分周比1/256 PWM周波数 0.244 kHz	固定値
37	PWM_FREQ_DividingRatio_1024	3	分周比1/1024 PWM周波数 0.061 kHz	固定値

内部構造体一覧

内部構造体一覧を下記に示します。

表 33. 内部構造体一覧

#	構造体名	内容
1	SrMesDivCtlVoltage	setCtlVoltageシリアル通信パラメータ格納構造体
2	SrMesDivPwmFreqset	setPWMFrequencyシリアル通信パラメータ格納構造体
3	SrMesDivRotation	setRotationEnableシリアル通信パラメータ格納構造体
4	SrMesDivStartFlag	setStartFlagシリアル通信パラメータ格納構造体
5	SrMesDivReadAdc	readAdcシリアル通信パラメータ格納用構造体
6	SrMesDivRecvReadAdc	readAdc応答用シリアル通信パラメータ格納用構造体
7	SrMesDivRecvReadModule	readModuleシリアル通信パラメータ格納用構造体

内部関数一覧

内部関数一覧を下記に示します。

表 34. 内部構造体一覧

#	関数名	内容	Call元関数
1	motorRotation	ArduinoのanalogWrite 関数により各出力PinにPWM出力設定を行うことで、モータの回転制御を実行する	setCtlVoltage setRotation setStartFlag
2	dutyRateToValue	Duty[%]をduty値(0～255)に変換する	motorRotation
3	setTCCR0	TCCR0レジスタの設定を行う	initLibPortSet
4	setTCCR1	TCCR1レジスタの設定を行う	initLibPortSet
5	setTCCR3	TCCR3レジスタの設定を行う	initLibPortSet
6	setTCCR4	TCCR4レジスタの設定を行う	initLibPortSet
7	setPinMode	ピンをOutputに指定	initLib initLibPortSet
8	timeoutPol	ポーリングタイムアウト時にモータを停止	guiSerialRead
9	dutyRateToReverseValue	Duty[%]をduty (0～255)に変換する	motorRotation

Arduino PWM出力とDutyの対応

- motorRotation
  setCtlVoltage関数、setRotation関数、setStartFlag関数でそれぞれ設定されるPWM Duty、回転方向とPWM駆動方法、モータのStart, Open, Brake状態をArduinoのPWM出力関数analogWrite()によってArduinoの各出力ピンにPWM出力を行います。各ピンへの出力を行う際に、dutyRateToValue関数を呼び出し、setCtlVoltage関数で設定されたPWM Duty[%]をduty値に変換した値をanalogWrite()に設定します。
- dutyRateToValue
  setCtlVoltage関数で設定されたPWM Duty (0%~100%)を0~255のduty値に変換します。

変換式は下記

duty = Round ((PWM Duty[%] × 255) / 100)

*Round = 四捨五入

ハードウエア仕様

動作条件

電源電圧について

LV8548のデータシートには推奨動作条件として電源電圧が4.0～16 Vと規定されています。

この範囲の電圧がICに印加されているとき、ICは安定して動作することを示しています。

ただし、その電圧でモータを駆動することになるため、この範囲の電圧であってもモータにとっては低すぎて駆動できなかったり、高すぎてモータを故障させる場合があります。

また、巻き線抵抗が低いモータを使用した場合、過電流が生じ、ICが破壊する可能性があります。使用するモータのスペックを確認のうえ、電圧を決定してください。

電源に電池を使用する場合、電圧が低下して4 V以下になることが想定されます。このとき、IC内部の回路動作が不安定となる場合があるため、ICが自動的に動作を停止します。（減電圧保護機能）

また、モータを駆動すると電源電圧が持ち上がる現象が起こることがあります。LV8548は最大定格として20Vを規定していますので、瞬時であってもこの電圧を超えることはIC破壊の原因となります。

（P44 「PWMモード=Forward/Reverse↔Openの場合」参照）

最大電流について

モータドライバICが内蔵する出力トランジスタや、ICチップとICピンを接続する金属線が流せる電流は限られています。これ以上の電流を流すとIC破壊の原因となります。これにより、モータドライバICは機種ごとに出力最大電流（以下Iomax）が規定されています。

LV8548のIomaxは1Aと規定されており、これを超えないように注意が必要です。

また、Iomaxは常に流せる電流を意味するものではありません。Iomax以下の電流値であっても、温度上昇により動作が停止することがあります。（次項参照）

動作温度について

半導体製品は最大定格ジャンクション温度（150°C前後）を超えるとIC破壊の原因となります。

逆の言い方をすると、最大定格ジャンクション温度までは正常動作が期待されます。

IC、特にモータドライバICは電力を消費し、それ自体が発熱します。使用環境の周囲温度とICの自己発熱により、IC内部の温度が最大定格ジャンクション温度を超えるとICが自動的に動作を停止します。（サーマルシャットダウン機能）また、ICの最大定格、推奨動作条件、電気的特性は周囲温度（Ta）が25℃で規定されているということをご理解ください。（詳しくはLV8548 ICのデータシートをご参照ください。）

関連用語集

逆起電圧とは

モータは電気エネルギーを機械エネルギーに変換すると同時に、その逆となる発電もしています。このときに発生する電圧を逆起電圧、逆起電力、誘起電圧とも呼びます。

モータの起動時は逆起電圧が発生しておらず、モータへの印加電圧とモータの巻き線抵抗による電流が流れます（突入電流）。

突入電流はモータ起動時の短時間ではありますが、モータ駆動電流のピークとなります。

逆起電圧はモータの回転スピードがあがるにつれて大きくなり、印加電圧を打ち消す働きをするため、モータ駆動電流は減少します。

Hブリッジとは

図8のようにBaseboardのモータ接続端子（OUT_A,OUT_B,OUT_C,OUT_D）につながるICの端子内部にはトランジスタがつながっています。このトランジスタとモータコイルによって『H』の文字を構成することから、この回路をHブリッジと呼びます。

LV8548は2組のHブリッジを内蔵しています。

これにより、DCモータを2個ないしステッパモータ（バイポーラタイプ）を1個駆動することができます。

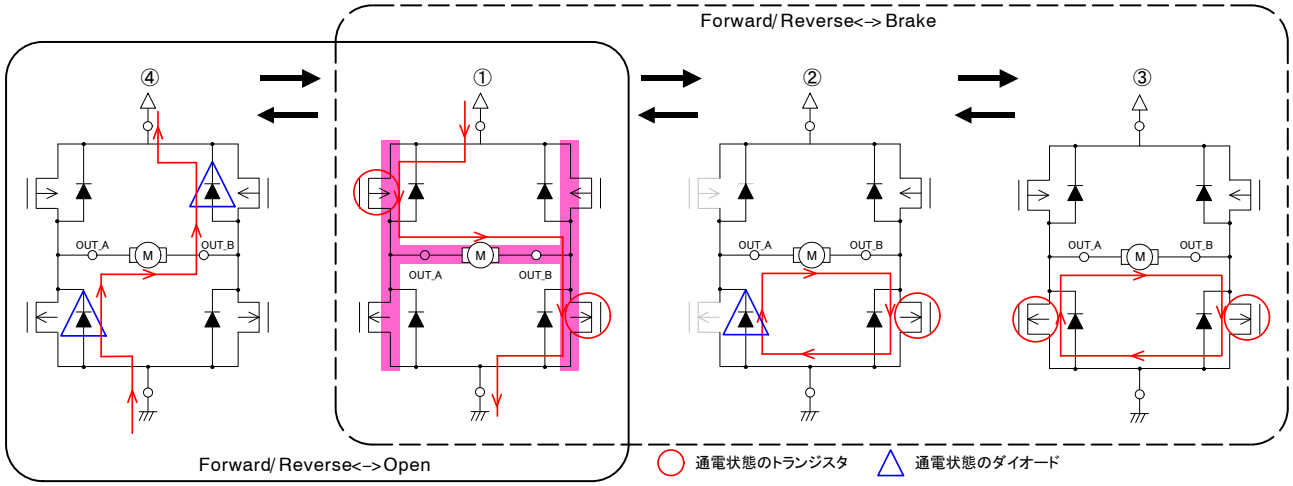


図 8. LV8548 の H ブリッジ状態遷移による電流経路

PWMとは

LV8548のDCモータ制御にも用いているPWMとは、Pulse Width Modulation（パルス幅**変**調）の略です。モータに電池を直接つないだときのように一定の電**圧**を印加するのではなく、パルス状の電**圧**を印加し、平均印加電**圧**を**変**化させることでモータの回転スピード、トルクをコントロールし、モータを効率よく駆動するために広く用いられている制御方式です。

ただし、あるパルス幅におけるモータの回転スピード、トルクは温度や負荷など、一定の条件下において保たれるものであり、これらの条件が**変**化した場合、回転スピード、トルクにも**変**化が生じます。

例えば、動作条件が**変**化しても回転スピードを保ちたい場合はフォトインタラプタのようなセンサーをシステムに組み込み、回転スピードを監視してパルス幅を調節するなどの手法が考えられます。

LV8548では2つのパターンのPWM制御により、HブリッジトランジスタのON/OFFを切り替えてDCモータを駆動します。

PWMモード＝Forward/Reverse<->Brakeの場合（図8、9参照）

3. 丸印の2つのトランジスタがONしてOUT_Aから OUT_Bの方向に電流が流れます。
（逆回転の場合はOUT_A側は下のトランジスタ、OUT_B側は上のトランジスタがONします。）この状態がPWM周期に占める割合をON Duty（オン デューティ）と呼び、この大小がモータスピードの大小に影響します。
4. OUT_Aの上トランジスタがOFFして、電源からの電流供給が途**絶**えます。
コイルは①と同じ方向に電流を流そうとする（コイルの慣性電流）ため、IC内**蔵**の三角印のダイオードを**経**由して、矢印の**経**路で電流が流れます。

②の状態がなく、①から③に遷移した場合、仮にOUT_Aの上トランジスタのOFFが**遅**れ、下トランジスタがONすると、電源とGNDがショート状態となり、大電流が流れる恐れがあります（貫通電流）。これを防ぐためにLV8548では②が設けられており、この状態はごくわずかな時間に限られます。
また、③から次の①に切り替わる際も同**様**に②を**経**由して貫通電流を防止しています。

5. OUT_Aの下トランジスタがONして②の電流ループが維持されます。
②③は電流の供給源が**絶**たれているため、電流は徐々に減少していきます。

このとき、2つの下トランジスタがONしてOUT_A、OUT_Bの電位がほぼGNDレベルの同電位になります。またモータ回転中にこの状態を保つと回生電流がなくなった後、逆起電**圧**による逆向きの電流がコイルに流れるため、モータは急減速します。
これらのことから、③をショートブレーキとよびます。
ただし、PWM駆動中の③の時間は非常に短いため、モータにブレーキがかかることはありません。

①→②→③→②→①→②→③→・・・という状態遷移によって、貫通電流が発生することのないモータ駆動が可能です。

PWMモード＝Forward/Reverse↔Openの場合（図8, 10参照）

① 丸印の2つのトランジスタがONしてOUT_AからOUT_Bの方向に電流が流れます。
（逆回転の場合はOUT_A側は下のトランジスタ、OUT_B側は上のトランジスタがONします。）この状態がPWM周期に占める割合をDuty（デューティ）と呼び、この大小がモータスピードの大小に影響します。

④ OUT_Aの上トランジスタ、OUT_Bの下トランジスタがOFFして、電源からの電流供給が途**絶**えます。

コイルは①と同じ方向に電流を流そうとするため、IC内蔵の三角印のダイオードを経由して、矢印のループで電流が流れます（回生電流）。
モータ回転中にこの状態を保つと回生電流がなくなった後は逆起電圧による電流を流す経路がないため、モータは惰性で回りながら、減速して止まります。

④のとき、回生電流は電源に戻っていきます。
安定化電源の場合、この電流を吸収して、印加電圧を保つことができますが、ACアダプタや電池の場合、電流を吸収することができず、大きな電圧上昇を引き起こします。
これにより、ICに印加される電圧が急上昇し、ICの破壊に至る場合があります。

LV8548の場合、電源電圧の最大定格が20 Vと規定されており、これを超えないように注意が必要です。
Baseboardに設置されている100 μFの電解コンデンサはこのときの電圧上昇を抑える働きをします。（図11参照）
この電圧上昇を抑える方法として、図11に示すような電圧クランプ回路を導入する方法があります。
Tr：FQP20N06, Di：1N5240B（ともにオンセミコンダクター），R1：150 Ωとした場合、R2を100～330 Ωの範囲で調整することでVCC電圧を20 V以下に制限することができます。VCC電圧が設定値を超えるとTrがONして、電圧上昇の原因となる電流をGNDに抜くことで、それ以上の電圧上昇を抑えます。

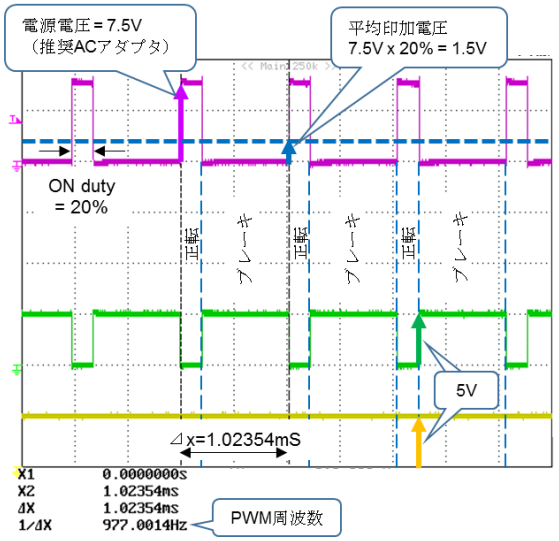


図 9. PWM波形例（Forward/Reverse↔Brake）

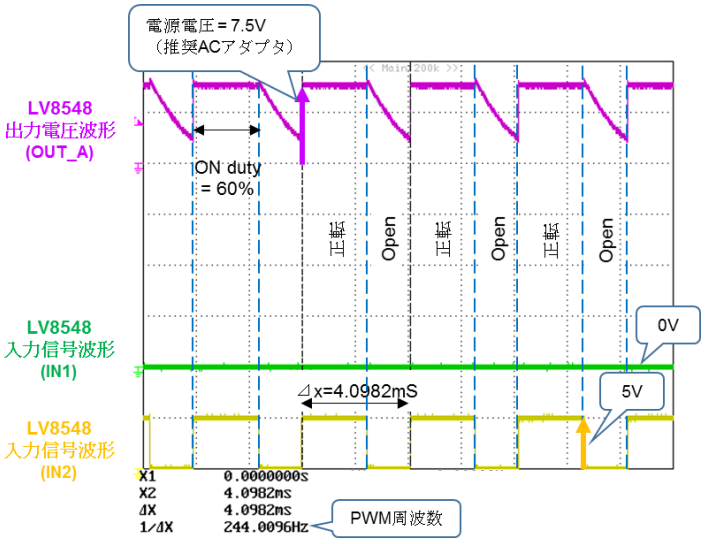


図 10. PWM波形例（Forward/Reverse↔Open）

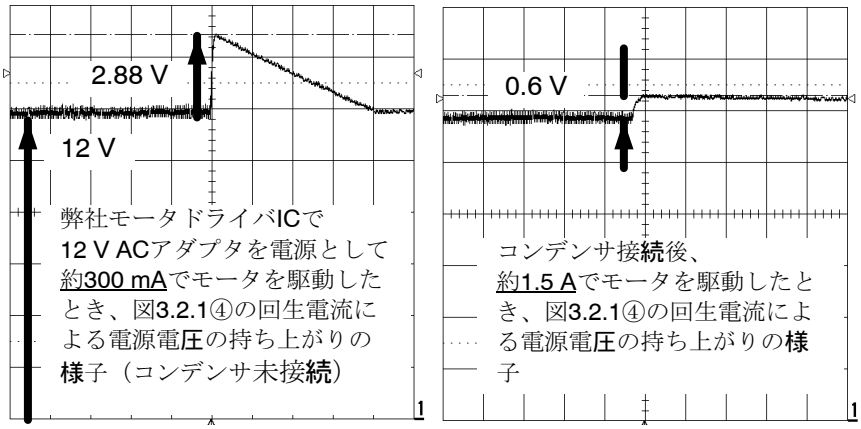


図 11. 回生電流による電源電圧の持ち上がりと電解コンデンサの効果

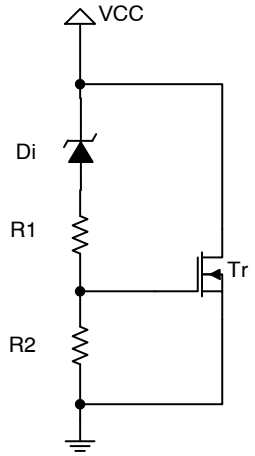


図 12. 電源電圧クランプ回路例

基板レイアウトの留意点

モータドライバICを実装する基板は、電流、電力消費による発熱、モータ駆動によるノイズなどを考慮したレイアウトを施さないと、期待されるパフォーマンスが得られない場合があります。

LV8548は単純な機能の製品であるため、比較的設計の余裕度はありますが、ポイントを紹介します。

VCC（電源）− GND間には必ずコンデンサを設置する

モータを回すために電源からモータに電流が供給されます。その瞬間、電源電圧が降下します。PWM制御の場合、この現象が頻発するため、ICの内部回路の動作が不安定になることがあります。この電圧降下を抑えるため、コンデンサを設置します。
このコンデンサはICのできるだけ近くに設置することが重要です。

VCC、GND、OUTのラインは太く、短く

この3種類のラインは大きな電流が流れます。細く、長いラインは抵抗を持つことになり、発熱や電圧降下の原因になります。
多層基板で設計する場合は、層間の同じライン同士をスルーホールで接続し、抵抗分をさげる方法もあります。

制御回路系のGNDとパワー系のGNDは分けて配線する

ICによっては2種類のGNDピンを持っているものがあります。IC内部の制御回路系のGNDピンをSGND、モータ駆動電流が流れるパワー系のGNDをPGNDなどとしています。1で説明した電源電圧の降下とは逆に、モータ駆動電流がGNDに流れることで、GNDの配線抵抗と電流の関係で、GNDのレベルが持ち上がります。この影響を制御回路が受けると、制御の基準となるGNDレベルが変動するため、誤動作の原因となります。これを避けるため、SGNDとPGNDは極力分けて配線し、VCC–GND間コンデンサが設置されているGNDの位置で1点あわせするのが理想です。
LV8548の場合はSGND、PGNDの区別はありませんが、ArduinoのGNDとの接続ラインは基板裏面の半分に大きな面積をとり、コンデンサの真裏でLV8548のGNDに接続しています。

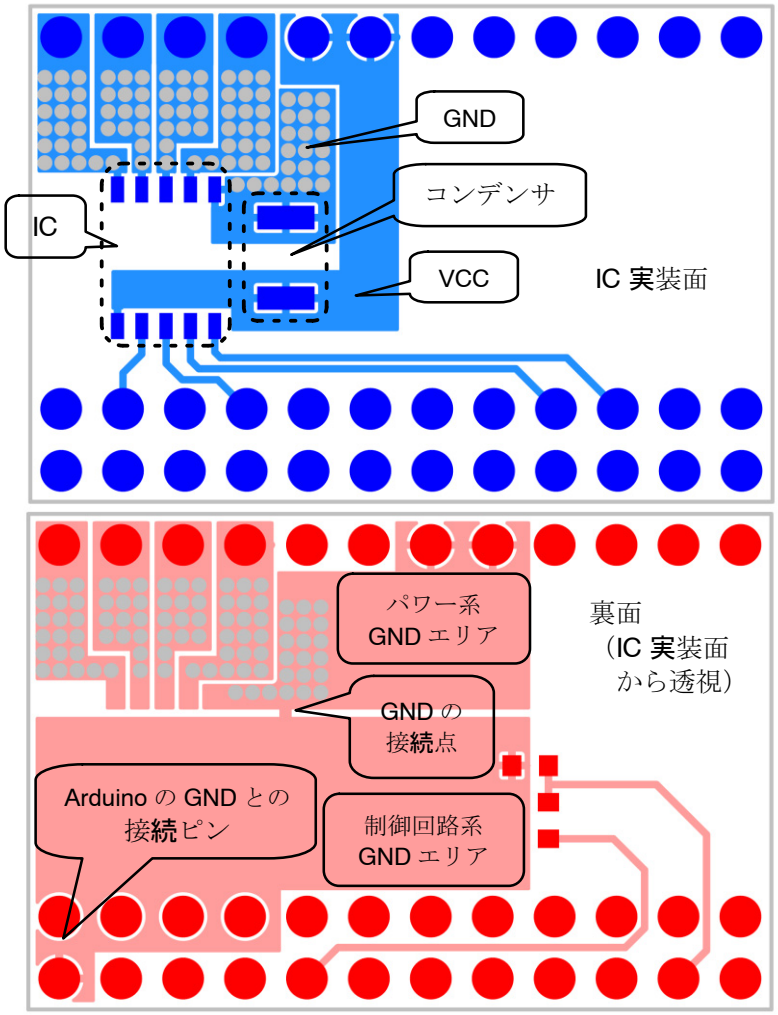


図 13. 基板レイアウトの留意点

ボード回路図

モータ ドライバモジュール (LV8548MCSLDGEVB)

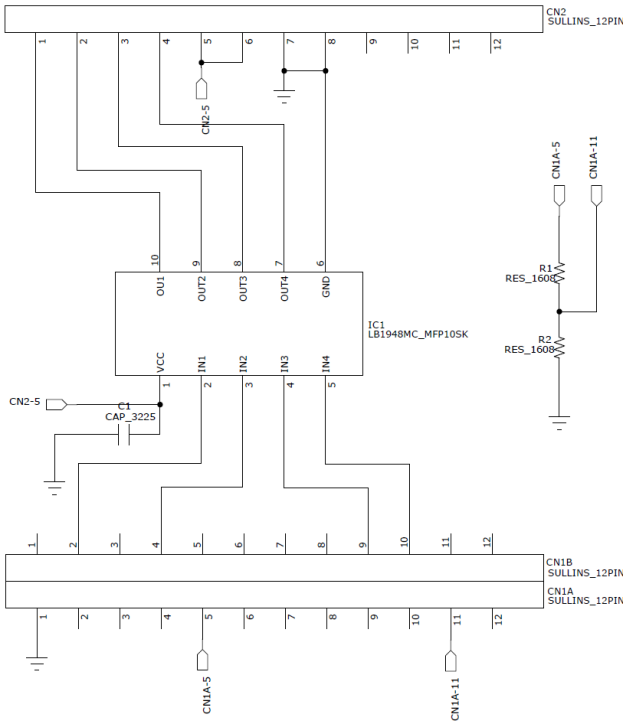


図 14. LV8548MCSLDGEVB モジュール回路図

表 35. LV8548MCモータ ドライバモジュール 部品表

部品記号	数量	部品名	値	許容差	サイズ	メーカ	製品名
IC1	1	モータドライバ	−	−	MFP10SK	オン セミコンダクター	LV8548MC
(R1)	1				1608(0603Inch)		
R2	1	チップ抵抗	100 kΩ, 0.1 W	±5%	1608(0603Inch)	KOA	RK73B1J**104J
C1	1	チップコンデンサ	10 μF, 50 V	±20%	3225(1210Inch)	村田製作所	GRM32ER71H106KA12L
CN1A, 1B	1	ピンヘッダ	12 pins x 2	−	30.48 x 5.08	Wurth Elektronik	61302421121
CN2	1	ピンヘッダ	12 pins	−	30.48 x 2.54	Wurth Elektronik	61301211121
PCB	1	基板	−		30.48 x 20.32		

R1にはなにも実装しないでください。

ベースボード (ONBB4AMGEVB)

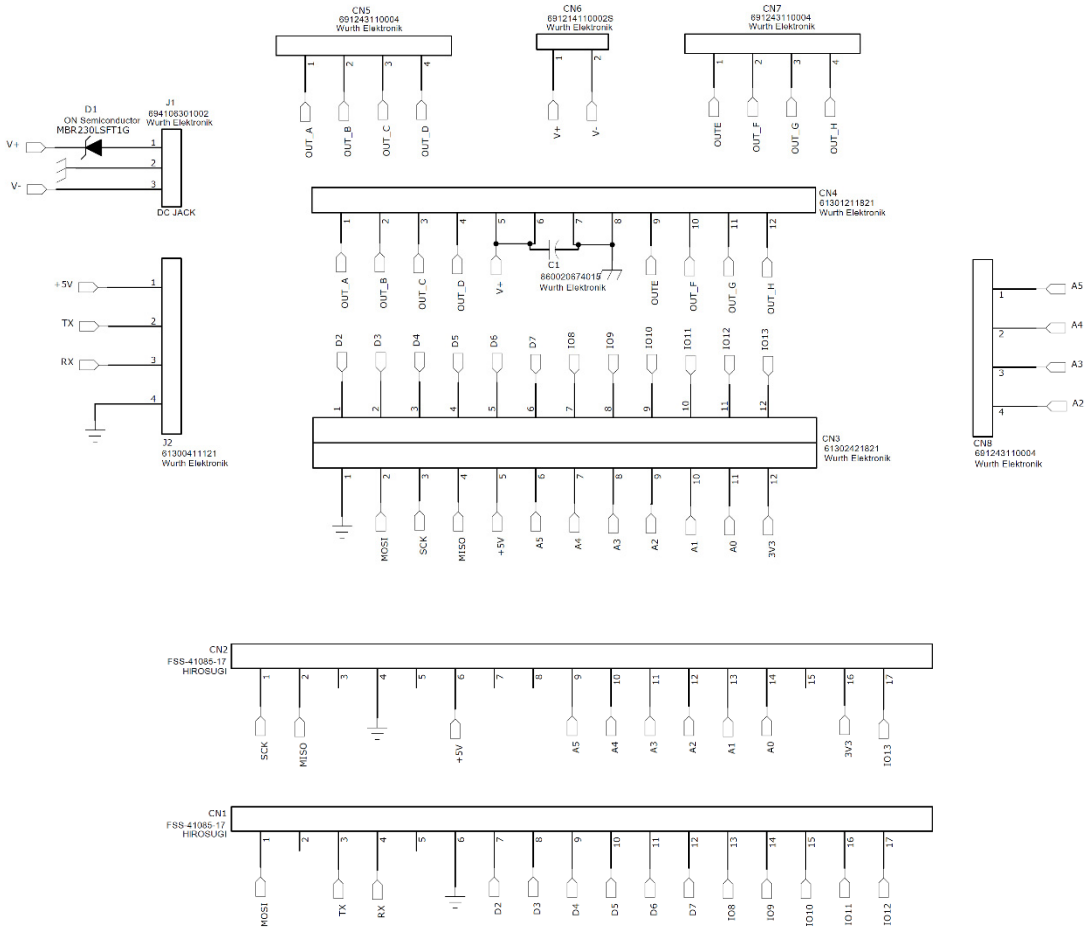


図 15. ONBB4AMGEVB ベースボード回路図

- 回路図内の同じピン名同士は配線で接続されています。
- DCジャックにコネクタを挿入すると、CN6のV−はオープンとなります。
  これにより、CN6に別の電源を接続しても故障の原因とはなりません。
  同時にCN6からの電圧入力は無効となります。またV−からGNDレベルにアクセスすることはできません。
  詳しくはメーカサイトをご参照ください。
  https://katalog.we−online.de/en/em/DC_RIGHT_ANGLED_6_4_69410X301002?sid=c33324d235
- Arduinoの電源はUSBコネクタから供給されます。
  スタンドアローン動作時はスマートフォン充電器からの USB給電も可能です。
- ArduinoのGND, J2のGNDとDCジャックのGNDまたはCN6のV−は、モータドライバモジュールが挿入されることによって接続します。

表 36. ベースボード 部品表

部品記号	数量	部品名	値	許容差	サイズ	メーカ	製品名
D1	1	ダイオード	−	−	SOD123	オン セミコンダクター	MBR230LSFT1G
CN1,2	2	Arduino Micro用 コネクタ	−	−	Φ1.02 x17 −2.54 pitch	廣杉計器	FSS−41085−17
CN3	1	モジュール用 コネクタ	−	−	Φ1.02 x12 x2lines −2.54 pitch	Wurth Elektronik	61302421821
CN4	1	モジュール用 コネクタ	−	−	Φ1.02 x12 −2.54 pitch	Wurth Elektronik	61301211821
CN5,7, 8	3	モータ接続用 コネクタ	−	−	Φ1.1 x4 −3.5 pitch	Wurth Elektronik	691243110004
CN6	1	電源接続用 コネクタ	−	−	Φ1.1 x2 −3.5 pitch	Wurth Elektronik	691214110002S
J1	1	DCジャック	−	−	9.0 x 14.5	Wurth Elektronik	694106301002
J2	1	UART用 ピンヘッダ	−	−	Φ1.1 x4 −2.54 pitch	Wurth Elektronik	61300411121
C1	1	電解コンデンサ	100 μF, 50 V	±10%	−	Wurth Elektronik	860020674015
PCB	1	PCB	−	−	80 x 60		

ベースボードの代わりに自作基板などを使用する場合は**VCC–GND**端子間に必ず**C1**相当の電解コンデンサを設置してください。未設置はモジュールの破壊、故障の原因となります。

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
19521 E. 32nd Pkwy, Aurora, Colorado 80011 USA
Phone: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
Fax: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800−282−9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910

ON Semiconductor Website: **www.onsemi.com**

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative