DIY で三輪車製作
TB67H450FNG を使用した工作ガイド

リファレンスガイド

RD204-RGUIDE-01-J

東芝デバイス&ストレージ株式会社

2020-09-14
Rev. 1.0

目次

1. はじめに 3

2. 略語 3

3. DIY 三輪車について 3

4. ハードウェア構成 4

4.1. BOM 4

4.2. ブロック図 5

4.3. ハードウェア結線 5

5. TB67H450FNG 基本動作 6

6. ソフトウェア構成 7

6.1. ソフトウェア作成環境 7

6.2. モーター動作 7

6.3. 関数階層 8
6.3.1. PWM 信号制御関数（Level 1） 8
6.3.2. 単体モーター制御関数（Level 2） 9
6.3.3. DIY 三輪車制御関数（Level 3） 9

7. DIY 三輪車動作 10

8. TB67H450FNG 使用時の特長 11

9. APPENDIX 12

9.1. ベース三輪車の説明 12

9.2. ソフトウェアサンプルコード 13
9.2.1. 初期定義 13
9.2.2. Arduino Setup 関数 13
9.2.3. Arduino Loop 関数 14
9.2.4. DIY 三輪車制御関数（Level 3） 15
9.2.5. モーター制御関数（Level 2） 16
9.2.6. PWM 信号制御関数（Level 1） 17

1. はじめに

本リファレンスガイドは、東芝製モータードライバーIC TB67H450FNG と制御用に Arduino Nano を使用して、「DIY 三輪車」を動作させるリファレンスデザインについて記載しています。TB67H450FNG は、動作電圧範囲（4.5V～44 V）が広く、低消費電力で、汎用性の高いピン配置になっており、より評価、開発が容易なICです。

本ガイドでの応用例は、TB67H450FNG をそれぞれ左右のタイヤを駆動するモーター用に使用しており、各モーターの制御を Arduino Nano で行う仕様としています。本ガイドでは、ハードウェアとソフトウェアの概要について記載しています。

2. 略語

MCD	:	Motor Control Driver IC	（モータードライバーIC）
マイコン	:	Microcontroller Unit	（マイクロコントローラー）
BOM	:	Bill of Materials	（部品表）
DIY	:	Do It Yourself	（ドゥ イット ユアセルフ）
EVB	:	Evaluation Board	（評価基板）
RM	:	Reference model	（リファレンスモデル）

3. DIY 三輪車について

DIY 三輪車は、東芝製の汎用モータードライバーIC TB67H450FNG を搭載したリファレンスモデルです。TB67H450FNG をマイコン制御することにより、様々なパターンの動作を実現することができます。

図 3-1 に TB67H450FNG を使用した DIY 三輪車を示します。

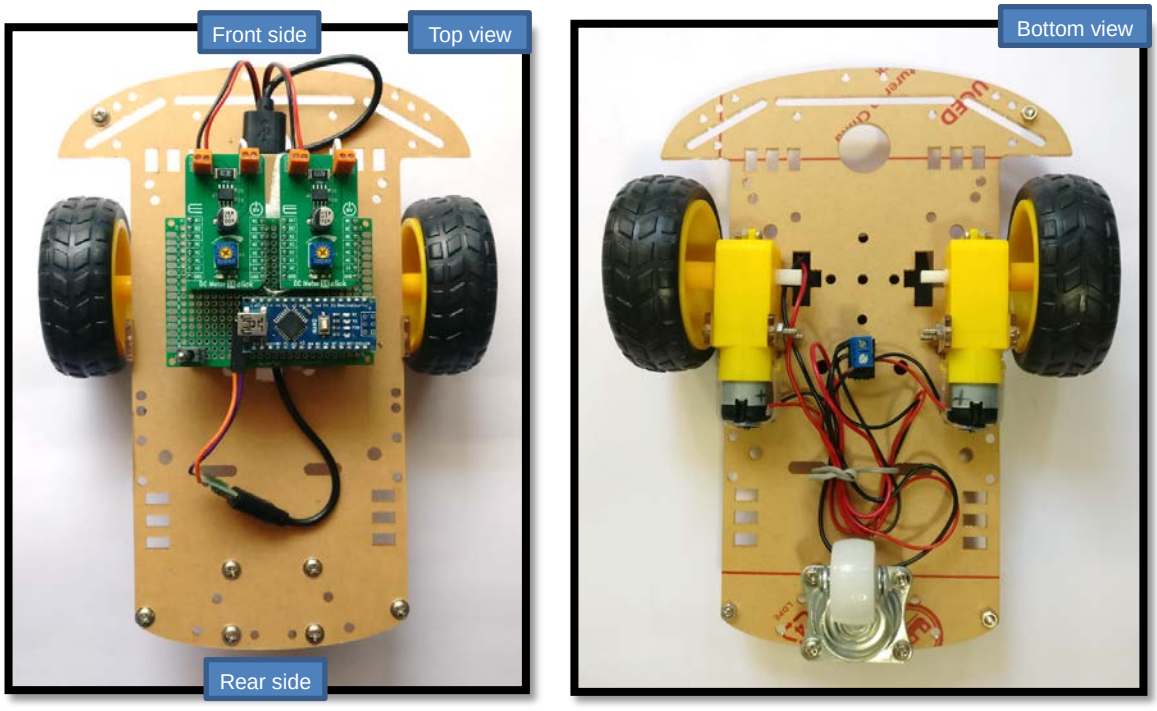

図 3-1　DIY 三輪車（MCD 付）（左画：上面図、右画：底面図）

4. ハードウェア構成

4.1. BOM

DIY 三輪車を作製するために以下のコンポーネントを使用しています。

- ベース三輪車（シャーシ ＋ モーター ＋ タイヤ）（セクション 9.1 を参照）
- Arduino Nano（マイコン基板）
- TB67H450FNG EVB（MIKROE-3982）（セクション 5 を参照）
- リチウムイオン電池（5V 出力）
- 接続ケーブル
- スイッチ

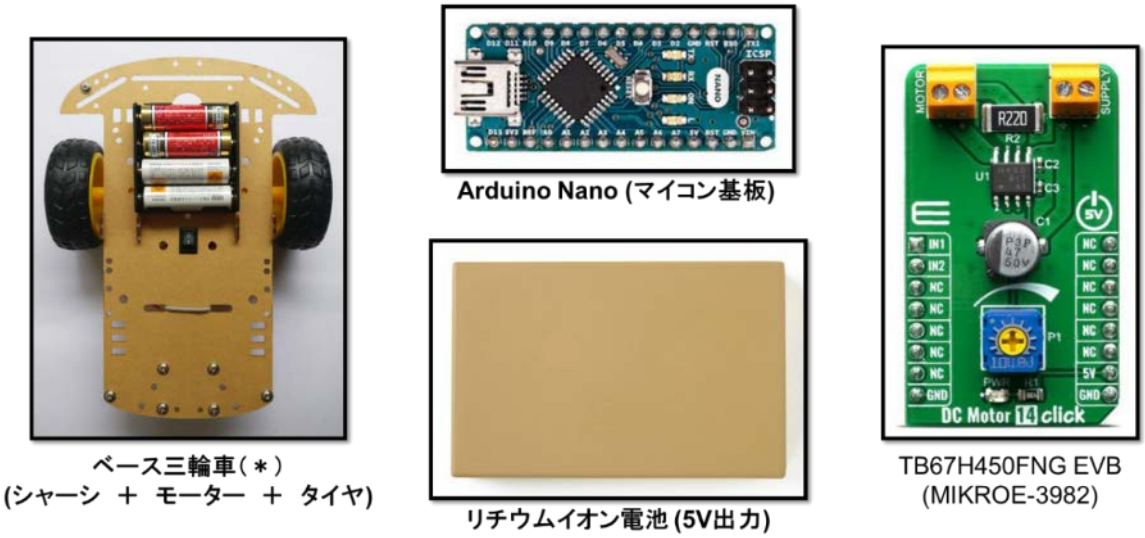

図 4-1 メインコンポーネント

（※）今回の DIY では 4×単三電池の代わりにリチウムイオン電池（5V 出力）を使用します。

これらのコンポーネントを組み合わせた DIY 三輪車を図 4-2 に示します。

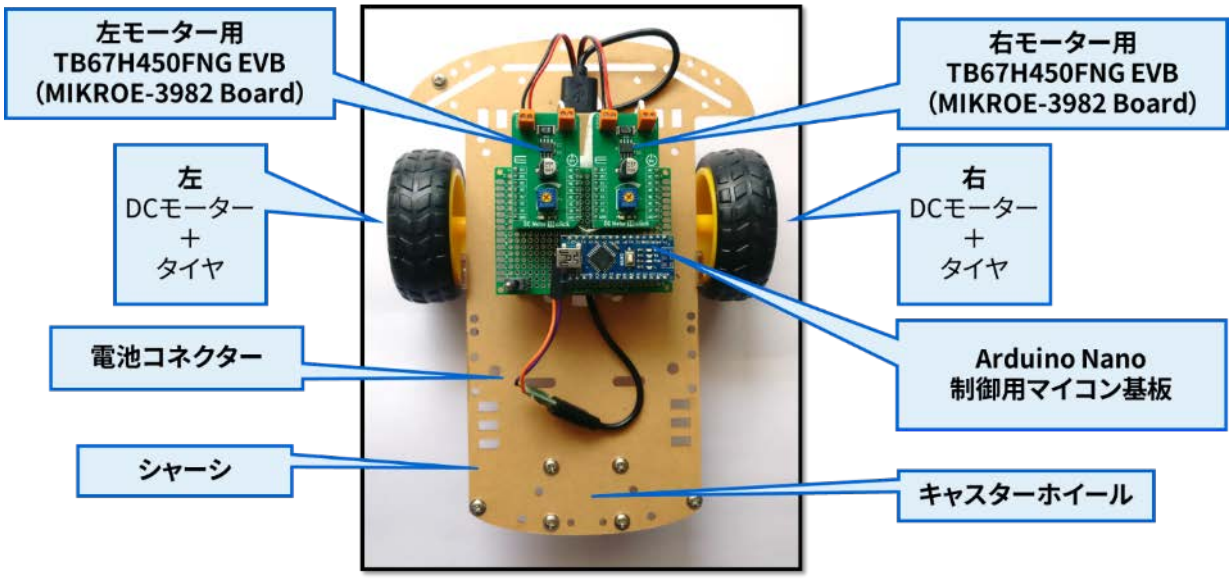

図 4-2 DIY 三輪車構成

4.2. ブロック図

DIY 三輪車のブロック図を図 4-3 に示します。全システムが 5V の電源で動作します。DIY 三輪車の 2 つのタイヤをそれぞれ別の DC モーターで駆動します。これらの DC モーターを独立して制御するために、2つの TB67H450FNG EVB を使用しています。Arduino Nano では、TB67H450FNG EVB 1 台毎に2つの制御 PWM 信号を生成し、DIY 三輪車として合計 4 つの PWM 信号を生成しています。

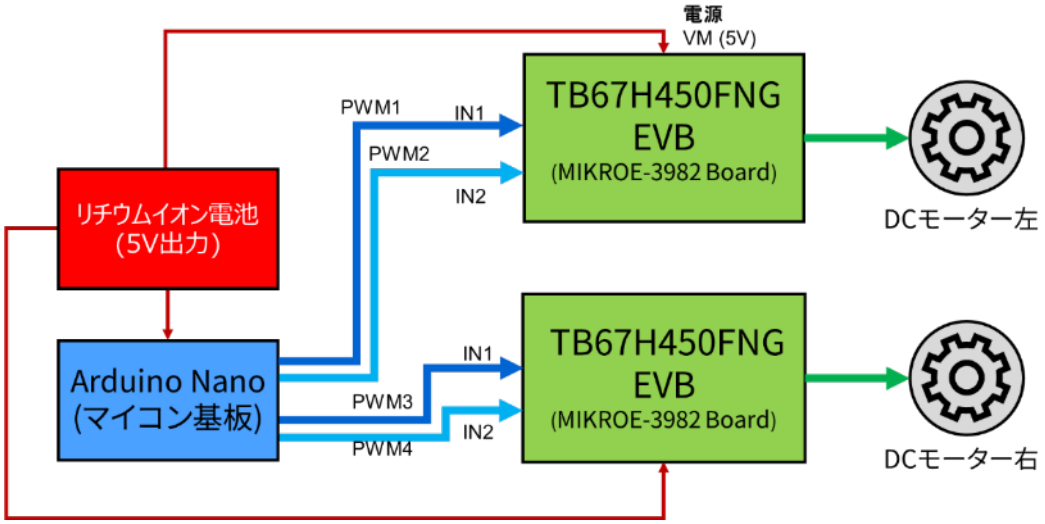

図 4-3 DIY 三輪車ブロック図

4.3. ハードウェア結線

DIY 三輪車に使用されているコンポーネントの結線を図 4-4 に示します。リチウムイオン電池（5V 出力）と GND は Arduino Nano と 2 つの TB67H450FNG EVB の電源（VM）と GND へ接続します。

各モーターの入力端子を TB67H450FNG EVB のモーター用信号出力へ接続します。左右のモーターの接続を逆にしていますので、例えば前進に設定すると両方のタイヤの回転方向が同じになるため、前進します。

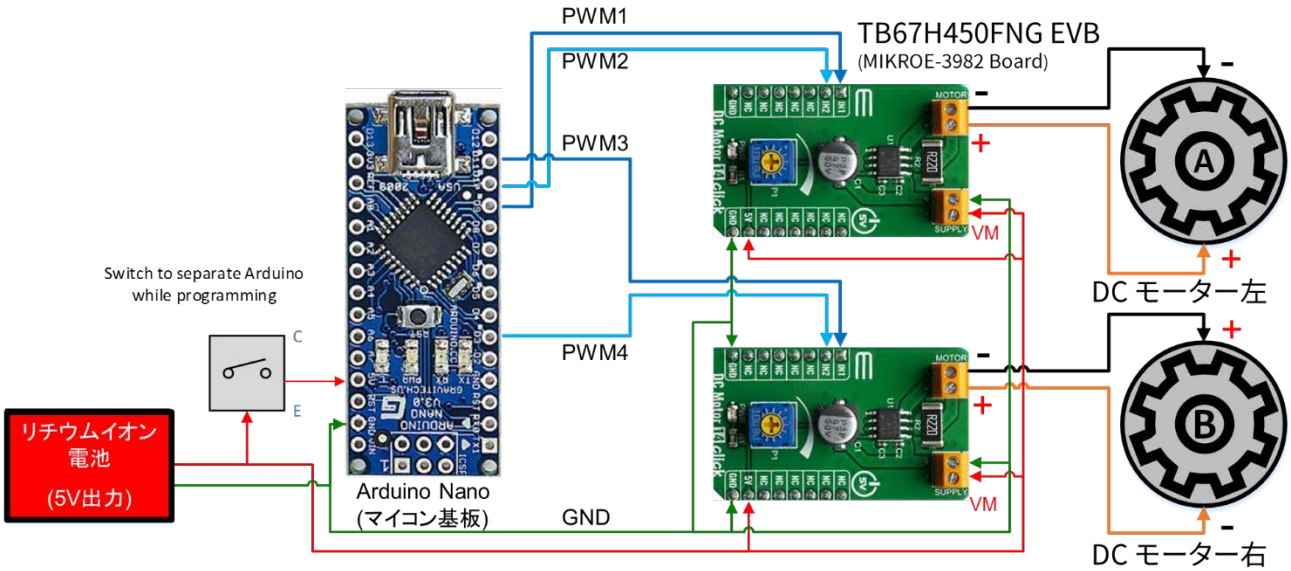

図 4-4 ハードウェア結線

TB67H450FNG EVB と Arduino Nano 基板の回路図はそれぞれのウェブサイトを参照してください。
TB67H450FNG EVB (Mikroe-3982): https://www.mikroe.com/dc-motor-14-click
Arduino Nano 基板説明: https://store.arduino.cc/usa/arduino-nano

5. TB67H450FNG 基本動作

MIKROE-3982 は TB67H450FNG の EVB です。

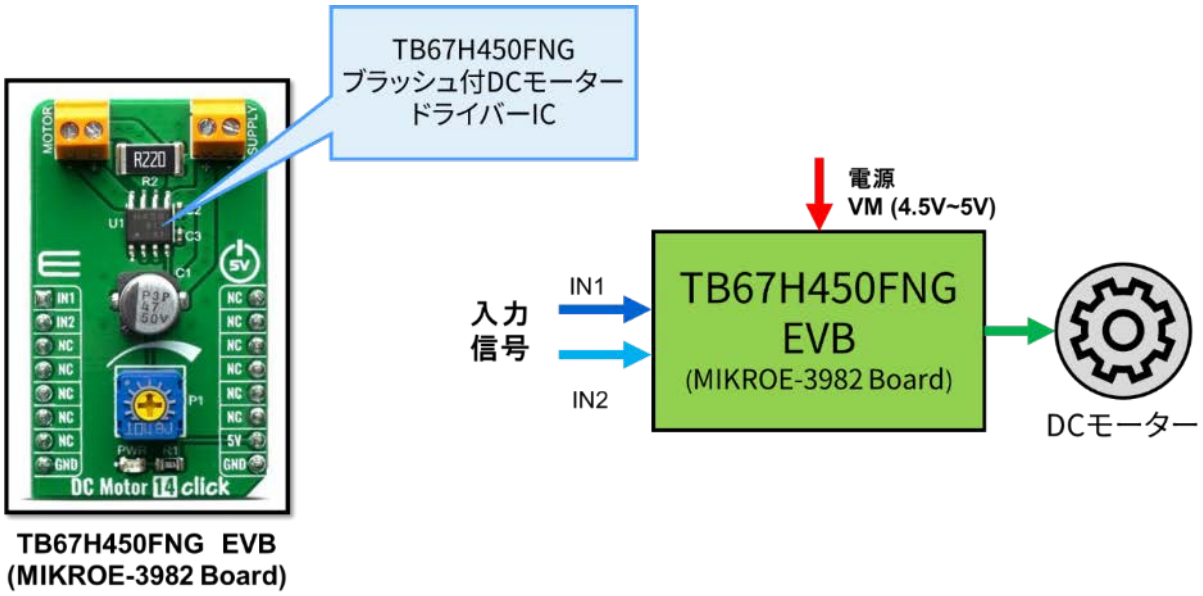

図 5-1 TB67H450FNG EVB

TB67H450FNG は表 1 の通り、IN1/IN2 の入力信号により 4 種類の DC モーター制御モードに対応しています。
また、IN1/IN2 信号に PWM を入力することにより、モーターの回転速度制御ができます。

IN1	IN2	OUT1	OUT2	モード
L	L	OFF (Hi-Z)	OFF (Hi-Z)	ストップ 1ms 経過でスタンバイモード
H	L	H	L	正転
L	H	L	H	逆転
H	H	L	L	ブレーキ

表 1 TB67H450FNG の DC モーター制御モード

TB67H450FNG の詳細は下記のウェブサイトを参照してください。

TB67H450FNG EVB (MIKROE-3982): https://www.mikroe.com/dc-motor-14-click
TB67H450FNG データシート: https://toshiba.semicon-storage.com/jp/semiconductor/product/motor-driver-ics/brushed-dc-motor-driver-ics/detail.TB67H450FNG.html

6. ソフトウェア構成

6.1. ソフトウェア作成環境

- Arduino IDE (1.8.10)
- Windows 10 32bit/64bit PC

Arduino ウェブサイト: https://www.arduino.cc

6.2. モーター動作

2つのモーターの組み合わせによる8種類の動作モードを以下に示します。

モード	左モーター制御	右モーター制御
前進	前方向に回転	前方向に回転
後退	後方向に回転	後方向に回転
ブレーキ	ブレーキ	ブレーキ
スタンドバイ	スタンドバイ	スタンドバイ
右方向転換	前方向に回転	後方向に回転
左方向転換	後方向に回転	前方向に回転
右方向転回	前方向に回転(高速)	前方向に回転(低速)
左方向転回	前方向に回転(低速)	前方向に回転(高速)

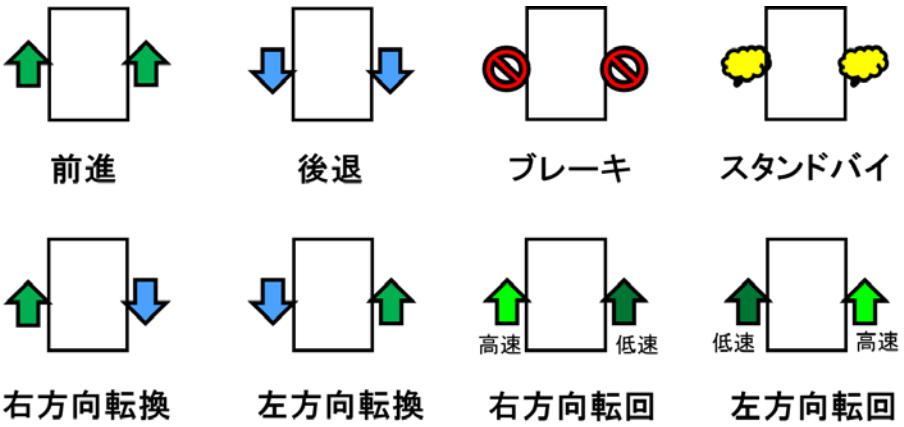

図 6-1 DIY 三輪車動作モード

6.3. 関数階層

セクション 6.2 に示した動作モードを作るために以下のようなソフトウェア関数階層を準備します。

- **Level 1**: PWM 信号を生成するために、ハードウェア（Timer）を設定する関数
- **Level 2**: Level 1 のハードウェア設定関数を使用して、モーターを制御する関数
- **Level 3**: Level 2 のモーター制御の関数を使用して、車体を制御する関数

Level 3 の車体制御の関数を使用して、簡単に様々な動作をさせることができます。

図 6-2　ソフトウェア関数階層（ブロック図）

```
//Function Definitions
//Car movement functions (Level 3)
void car_forward(int spd); // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]    spd(%):0-100
void car_reverse(int spd); // Move Car Reverse[Left_Motor-Bw, R_Motor-Bw]    spd(%):0-100
void car_brake();          // BRAKE Car movement[Left_Motor-BRAKE, R_Motor-BRAKE]
void car_standby();        // STANDBY Car motors [Left_Motor-Standby, R_Motor-Standby]
void car_turn_left(int deg);      // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]     deg: rotaion angle
void car_turn_right(int deg);     // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]    deg: rotaion angle
void car_turn_circle(int L_spd, int R_spd); //Turn car in circle L_spd & R_spd are speed of left & R wheels
//Motor control functions (Level 2)
void L_forward(int spd); // Move Left motor forward[LIN1-H, LIN2-L]    spd(%):0-100
void L_reverse(int spd); // Move Left motor reverse[LIN1-L, LIN2-H]    spd(%):0-100
void L_brake();          // Brake Left motor [LIN1-H, LIN2-H]
void L_standby();        // Standby Left motor [LIN1-L, LIN2-L]
void R_forward(int spd); // Move Right motor forward[RIN1-H, RIN2-L]    spd(%):0-100
void R_reverse(int spd); // Move Right motor reverse[RIN1-L, RIN2-H]    spd(%):0-100
void R_brake();          // Brake Right motor [RIN1-H, RIN2-H]
void R_standby();        // Standby Right motor [RIN1-L, RIN2-L]
//Timer, pin functions (Level 1)
void timer_out_enDi(int pin,int enable); //pin: 9,10,11,3      enable: en:Output_Enable, di:Output_Disable
void pin_duty_set(int pin, int duty); //pin: 9,10,11,3       duty(%):0-100
```

車体 速度設定

車体 回転角度設定

モーター 速度設定

図 6-3　全関数階層

サンプルコードはセクション 9.2 を参照してください。

6.3.1. PWM 信号制御関数（Level 1）

図 4-3 に示した TB67H450FNG 制御用 PWM[1 - 4]信号は、Arduino Nano に搭載されている ATmega328 マイコン（※）の 2 つのハードウェアタイマー（タイマー1、2）を使用して生成します。

（※）Arduino Nano と ATmega328 マイコンの詳細については下記のウェブサイトを参照してください。

Arduino Nano: https://store.arduino.cc/usa/arduino-nano
ATmega328 マイコン: https://www.microchip.com/wwwproducts/en/ATmega328

PWM 信号制御のために以下の関数を作ります。

- timer_out_enDi(pin, enable) ： 端子ごとに PWM を Enable/Disable することができます。
- pin_duty_set(pin, duty) ： 端子ごとに PWM の Duty を設定することができます。

6.3.2. 単体モーター制御関数（Level 2）

TB67H450FNG は以下の 4 つの動作モードに対応しています。（セクション 5 を参照）

- 正転（回転速度調整が可能）
- 逆転（回転速度調整が可能）
- ブレーキ
- スタンバイ

ソフトウェアで単体モーターを制御するために 4 つの Level 2 関数を準備します。

- **モーター正転**: **forward(x)** ： モーター正転用信号生成
  「x」入力により回転速度を設定します。
- **モーター逆回転**: **reverse(x)** ： モーター逆回転用信号生成
  「x」入力により回転速度を設定します。
- **ブレーキ**: **brake()** ： モーターブレーキ用入力信号生成
- **スタンバイ**: **standby()** ： モーター制御を停止するためにモーター入力を Hi-Z にする。

図 6-4 単体モーター単体制御向け Level 2 関数

6.3.3. DIY 三輪車制御関数（Level 3）

セクション 6.3.2 で説明したモーター単体動作用の Level 2 関数を使用して、左右のモーターを同時に制御する 7 種類の動作モードの Level 3 関数を準備します。
これらの関数により、DIY 三輪車としての基本動作制御が可能になります。

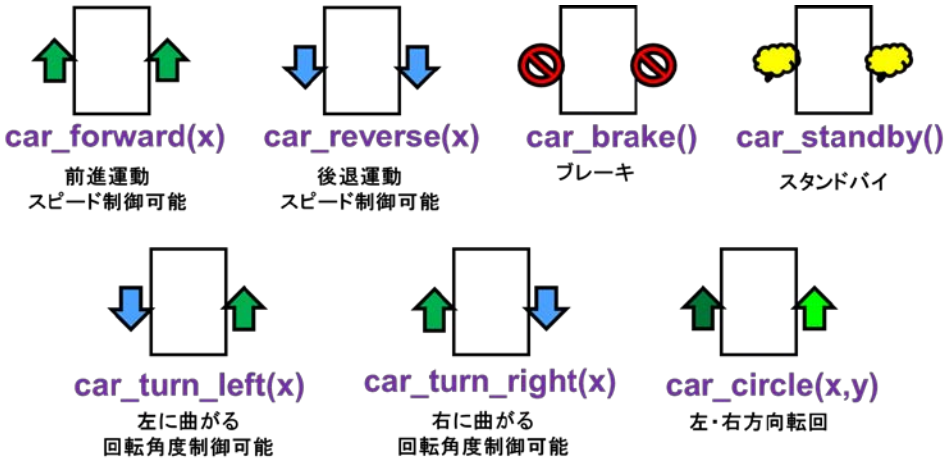

図 6-5 DIY 三輪車制御向け Level 3 関数

7. DIY 三輪車動作

セクション 6.3.3 の Level 3 関数を組み合わせることにより、図 7-1 に示すように、DIY 三輪車の様々な動作を実現することができます。

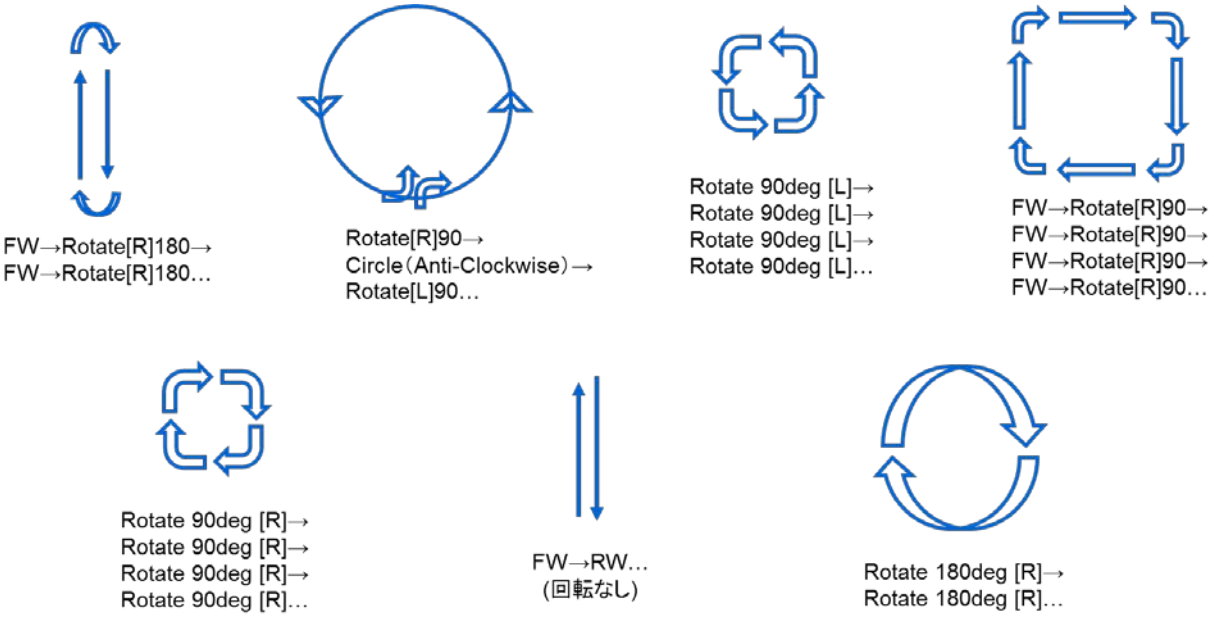

図 7-1 TB67H450FNG による DIY 三輪車動作パターン例

8. TB67H450FNG 使用時の特長

TB67H450FNG の 4 つの動作モードを使用することにより、DC モーターに様々な制御を行うことができます。
TB67H450FNG 使用時の特長を以下の表に示します。

TB67H450FNG 使用時の特長	ベース三輪車（※）
● 単一電源において、正回転・逆回転動作 ● ブレーキ制御 ● 速度調整が可能 ● 様々な動作が可能 ➢ 左右方向転換、前進、後退など	● 単一電源において、一方向の回転動作 ● 電源供給停止により、惰性回転・停止 ● 速度調整が不可 ● 複雑な動作が不可

※ ベース三輪車については、セクション 9.1 を参照してください。

9. Appendix

9.1. ベース三輪車の説明

ベース三輪車は、以下の部品から構成されています。

- ゴム付きタイヤ　2 個
- キャスターホイール　1 個
- DC モーター　2 個
- シャーシ　1 台
- スイッチ　1 個
- 単三用電池ボックス　1 個

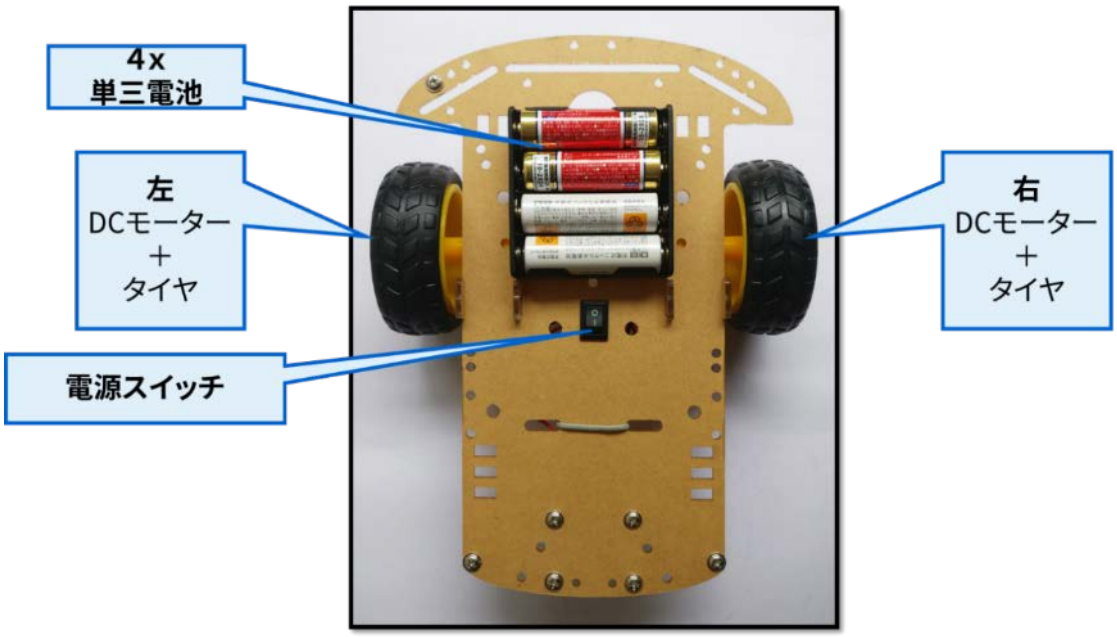

図 9-1　ベース三輪車

ベース三輪車のブロック図を図 9-2 に示します。

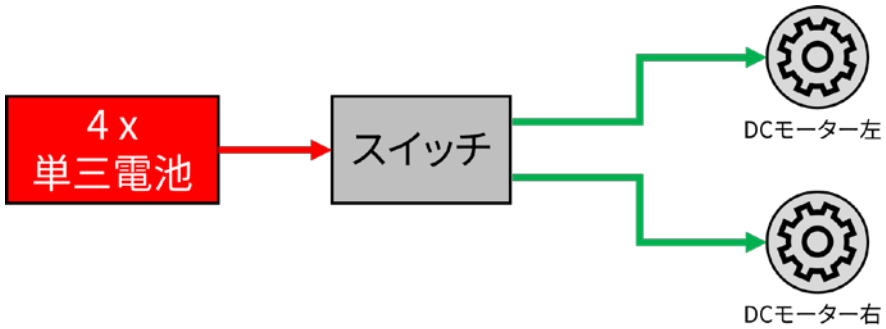

図 9-2　ベース三輪車ブロック図

ベース三輪車のスイッチは、モーターに対する電源供給 ON/OFF を行うため、

スイッチ ON:　前進
スイッチ OFF:　惰性回転 → 停止

の動作のみに対応しています。

9.2. ソフトウェアサンプルコード

Arduino IDE で作られた DIY 三輪車制御ソフトウェアサンプルコードを以下に示します。

9.2.1. 初期定義

```
#define pinLin1 9  // Left motor MCD IN1
#define pinLin2 10 // Left motor MCD IN2
#define pinRin1 11 // Right motor MCD IN1
#define pinRin2 3  // Right motor MCD IN2
#define en 1
#define di 0

//Initial third wheel status (during power up)
char third_wheel='B'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.

//Function Definitions
//Car movement functions (Level 3)
void car_forward(int spd); // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
void car_reverse(int spd); // Move Car Reverse[Left_Motor-Bw, R_Motor-Bw]   spd(%):0-100
void car_brake();         // BRAKE Car movement[Left_Motor-BRAKE, R_Motor-BRAKE]
void car_standby();       // STANDBY Car motors [Left_Motor-Standby, R_Motor-Standby]
void car_turn_left(int deg);     // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]    deg: rotaion angle
void car_turn_right(int deg);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotaion angle
void car_turn_circle(int L_spd, int R_spd); //Turn car in circle L_spd & R_spd are speed of left & R wheels
//Motor control functions (Level 2)
void L_forward(int spd); // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
void L_reverse(int spd); // Move Left motor reverse[LIN1-L, LIN2-H]   spd(%):0-100
void L_brake();         // Brake Left motor [LIN1-H, LIN2-H]
void L_standby();       // Standby Left motor [LIN1-L, LIN2-L]
void R_forward(int spd); // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
void R_reverse(int spd); // Move Right motor reverse[RIN1-L, RIN2-H]   spd(%):0-100
void R_brake();         // Brake Right motor [RIN1-H, RIN2-H]
void R_standby();       // Standby Right motor [RIN1-L, RIN2-L]
//Timer, pin functions (Level 1)
void timer_out_enDi(int pin,int enable); //pin: 9,10,11,3    enable: en:Output_Enable, di:Output_Disable
void pin_duty_set(int pin, int duty); //pin: 9,10,11,3     duty(%):0-100
```

9.2.2. Arduino Setup 関数

```
////////////////////////////////////////////////////////////////////////////////////////////////
// the setup function runs once when you press reset or power the board
////////////////////////////////////////////////////////////////////////////////////////////////
void setup() {
  Serial.begin(9600);

  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(pinLin1, OUTPUT);
  pinMode(pinLin2, OUTPUT);
  pinMode(pinRin1, OUTPUT);
  pinMode(pinRin2, OUTPUT);

  digitalWrite(pinLin1, LOW);
  digitalWrite(pinLin2, LOW);
  digitalWrite(pinRin1, LOW);
  digitalWrite(pinRin2, LOW);


  //TIMER 1 (8bit, fixed top value-255)
  TCCR1A = _BV(WGM10) ; //Fast PWM Mode
  TCCR1B = _BV(WGM12)  | _BV(CS12);          //Prescaler 1/256 (For changing frequency) // 244Hz
  //TCCR1B = _BV(WGM12)  | _BV(CS11) | _BV(CS10); //Prescaler 1/64 (For changing frequency) // 1kHz
  //TCCR1B = _BV(WGM12)  | _BV(CS11);           //Prescaler 1/8 (For changing frequency) // 7.8kHz
  //TCCR1B = _BV(WGM12)  | _BV(CS10);           //Prescaler 1/1 (For changing frequency) // 62kHz
  OCR1A = 64; //Counter for pin9
  OCR1B = 128;  //Counter for pin10
  //TIMER 2
  TCCR2A = _BV(WGM21) | _BV(WGM20); //Fast PWM mode
  TCCR2B =  _BV(CS22) | _BV(CS21); //Prescaler 1/256 (For changing frequency) 244// Hz   40%
  //TCCR2B =  _BV(CS22); //Prescaler 1/64 (For changing frequency) // 1kHz   50%
  //TCCR2B =  _BV(CS21); //Prescaler 1/8 (For changing frequency) // 7.8kHz  70%
  //TCCR2B = _BV(CS20);  //Prescaler 1/1 (For changing frequency) // 62kHz
  OCR2A = 140;  //PWM on Pin 11 (Duty OCR2A/255)
  OCR2B = 200;  //PWM on pin 3   (Duty OCR2B/255)

  car_standby();  // STANDBY Car motors [Left_Motor-Standby, R_Motor-Standby]
  if(third_wheel=='B' or third_wheel=='F'){
    digitalWrite(LED_BUILTIN, HIGH);
  }else{
    digitalWrite(LED_BUILTIN, LOW);
  }
  delay(4000);
}
```

9.2.3. Arduino Loop 関数

```
//////*************************************************************************************///////
//                the loop function runs over and over again forever
//////*************************************************************************************///////
void loop() {
  ////////////////////////////////////  FW-Rotate[R]180-FW-Rotate[R]180
  delay(2000);
  for(int i=0;i<2;i++){
    //FW
    car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
    delay(500);
    car_brake();
    delay(200);
    car_turn_right(180);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(500);
    //Return
    car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
    delay(500);
    car_brake();
    delay(200);
    car_turn_right(180);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(500);
  }
  ////////////////////////////////////  Rotate 90deg [L]x4
  delay(2000);
  for(int i=0;i<4;i++){
    car_turn_left(90);     // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]    deg: rotaion angle
    delay(500);
  }
  ////////////////////////////////////  Rotate 90deg [R]x4
  delay(2000);
  for(int i=0;i<4;i++){
    car_turn_right(90);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(500);
  }
  ////////////////////////////////////  FW & RW (without rotatio & without acceleration)
  delay(2000);
  for(int i=0;i<4;i++){
    car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
    delay(500);
    car_brake();
    delay(200);
    car_reverse(50); // Move Car Reverse[Left_Motor-Bw, R_Motor-Bw]   spd(%):0-100
    delay(480);
    car_brake();
    delay(200);
  }
  ////////////////////////////////////  Rotate 180deg [R]x2
  delay(2000);
  for(int i=0;i<2;i++){
    car_turn_right(180);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(500);
  }
  ////////////////////////////////////  normal Square
  delay(2000);
  for(int i=0;i<1;i++){
    //FW (1st side of square)
    car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
    delay(700);
    car_brake();
    delay(200);
    for(int i=0;i<3;i++){  //(2nd, 3rd, 4th side of square)
      //Rotate_Right
      car_turn_right(90);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
      delay(200);
      //FW
      car_forward(50);  // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
      delay(700);
      car_brake();
      delay(200);
    }
    //Rotate_Right
    car_turn_right(90);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
    delay(200);
  }
  ////////////////////////////////////  Circle
  delay(2000);
  car_turn_right(90);    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
  delay(200);
  car_turn_circle(40,100); //Turn car in circle L_spd & R_spd are speed of left & R wheels
  delay(2900);
  car_brake();
  delay(200);
  car_turn_left(90);     // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]    deg: rotaion angle
  delay(200);
}
```

9.2.4. DIY 三輪車制御関数（Level 3）

```
////////////////////////////////////////////////////////////////////////////////////////
//////////            Car Movement control functions            ////////////////////
////////////////////////////////////////////////////////////////////////////////////////
void car_forward(int spd){ // Move Car Forward[Left_Motor-Fw, R_Motor-Fw]   spd(%):0-100
  L_forward(spd); // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
  R_forward(spd); // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
  third_wheel='B'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.
  digitalWrite(LED_BUILTIN, HIGH);
}
void car_reverse(int spd){ // Move Car Reverse[Left_Motor-Bw, R_Motor-Bw]   spd(%):0-100
  L_reverse(spd); // Move Left motor reverse[LIN1-L, LIN2-H]   spd(%):0-100
  R_reverse(spd); // Move Right motor reverse[RIN1-L, RIN2-H]   spd(%):0-100
  third_wheel='F'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.
  digitalWrite(LED_BUILTIN, HIGH);
}
void car_brake(){        // BRAKE Car movement[Left_Motor-BRAKE, R_Motor-BRAKE]
  L_brake();         // Brake Left motor [LIN1-H, LIN2-H]
  R_brake();         // Brake Right motor [RIN1-H, RIN2-H]
}
void car_standby(){       // STANDBY Car motors [Left_Motor-Standby, R_Motor-Standby]
  L_standby();       // Standby Left motor [LIN1-L, LIN2-L]
  R_standby();       // Standby Right motor [RIN1-L, RIN2-L]
}
void car_turn_left(int deg){     // Turn Car Left[Left_Motor-Rw, R_Motor-Fw]    deg: rotation angle
  int extra=0;
  //Turn Left
  R_forward(60); // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
  L_reverse(60); // Move Left motor reverse[LIN1-L, LIN2-H]   spd(%):0-100    //motor speed cant be super slow

    if(third_wheel == 'F' || third_wheel == 'B'){ //90deg different from desired direction (left)
    extra=0;
  } else if (third_wheel == 'R'){  //180deg different from desired direction (left)
    extra=14;
  }

  if(deg<120){
    delay(277*(deg/90.0)+extra);
  }
  else{
    delay(495*(deg/180.0)+extra);
  }

  car_brake();
  third_wheel='L'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.
  digitalWrite(LED_BUILTIN, LOW);
}
void car_turn_right(int deg){    // Turn Car Right[Left_Motor-Fw, R_Motor-Rw]   deg: rotation angle
  int extra=0;
  //Turn Right
  L_forward(60); // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
  R_reverse(60); // Move Right motor reverse[RIN1-L, RIN2-H]   spd(%):0-100

  if(third_wheel == 'F' || third_wheel == 'B'){ //90deg different from desired direction (right)
    extra=0;
  } else if (third_wheel == 'L'){  //180deg different from desired direction (right)
    extra=14;
  }

  if(deg<120){
    delay(281*(deg/90.0)+extra);
  }
  else{
    delay(500*(deg/180.0)+extra);
  }
  car_brake();
  third_wheel='R'; //Back 'B', Front 'F', Left 'L', Right 'R'  //Status of which side the third wheel is
protruding.
  digitalWrite(LED_BUILTIN, LOW);
}
void car_turn_circle(int L_spd, int R_spd){ //Turn car in circle L_spd & R_spd are speed of left & R wheels
  L_forward(L_spd); // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
  R_forward(R_spd); // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
}
```

9.2.5. モーター制御関数（Level 2）

```
////////////////////////////////////////////////////////////////////////////////////////
//////////                Motor control functions                 ////////////////////
////////////////////////////////////////////////////////////////////////////////////////
void L_forward(int spd){  // Move Left motor forward[LIN1-H, LIN2-L]   spd(%):0-100
  //pin L in1
  timer_out_enDi(pinLin1, en); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  pin_duty_set(pinLin1, spd);    //pin:9,10,11,3     duty(%):0-100
  //pin L in2
  timer_out_enDi(pinLin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin2, LOW);
}
void L_reverse(int spd){ // Move Left motor reverse[LIN1-L, LIN2-H]   spd(%):0-100
  //pin L in1
  timer_out_enDi(pinLin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin1, LOW);
  //pin L in2
  timer_out_enDi(pinLin2, en); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  pin_duty_set(pinLin2, spd);    //pin:9,10,11,3     duty(%):0-100
}
void L_brake(){          // Brake Left motor [LIN1-H, LIN2-H]
  //pin L in1
  timer_out_enDi(pinLin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin1, HIGH);
  //pin L in2
  timer_out_enDi(pinLin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin2, HIGH);
}
void L_standby(){       // Standby Left motor [LIN1-L, LIN2-L]
  //pin L in1
  timer_out_enDi(pinLin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin1, LOW);
  //pin L in2
  timer_out_enDi(pinLin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinLin2, LOW);
}

void R_forward(int spd){  // Move Right motor forward[RIN1-H, RIN2-L]   spd(%):0-100
  //pin R in1
  timer_out_enDi(pinRin1, en); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  pin_duty_set(pinRin1, spd);    //pin:9,10,11,3     duty(%):0-100
  //pin R in2
  timer_out_enDi(pinRin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin2, LOW);
}
void R_reverse(int spd){ // Move Right motor reverse[RIN1-L, RIN2-H]   spd(%):0-100
  //pin R in1
  timer_out_enDi(pinRin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin1, LOW);
  //pin R in2
  timer_out_enDi(pinRin2, en); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  pin_duty_set(pinRin2, spd);    //pin:9,10,11,3     duty(%):0-100
}
void R_brake(){          // Brake Right motor [RIN1-H, RIN2-H]
  //pin R in1
  timer_out_enDi(pinRin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin1, HIGH);
  //pin R in2
  timer_out_enDi(pinRin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin2, HIGH);
}
void R_standby(){       // Standby Right motor [RIN1-L, RIN2-L]
  //pin R in1
  timer_out_enDi(pinRin1, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin1, LOW);
  //pin R in2
  timer_out_enDi(pinRin2, di); //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  digitalWrite(pinRin2, LOW);

}
```

9.2.6. PWM 信号制御関数（Level 1）

```
////////////////////////////////////////////////////////////////////////////////////////////
//////////                  Timer, pin functions                  ////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
void pin_duty_set(int pin, int duty){ //pin:9,10,11,3     duty(%):0-100
  switch(pin){
    case 9: //Pin9  - OC1A - Timer1
      OCR1A= (duty*255)/100;
      break;
    case 10://Pin10 - OC1B - Timer1
      OCR1B= (duty*255)/100;
      break;
    case 11://Pin11 - OC2A - Timer2
      OCR2A= (duty*255)/100;
      break;
    case 3: //Pin3  - OC2B - Timer2
      OCR2B= (duty*255)/100;
      break;
  }
}

void timer_out_enDi(int pin,int enable){ //pin: 9,10,11,3    enable: en:Output_En, di:Output_Di
  //Serial.print(pin);
  //Serial.println(enable);
  switch (pin){
    case 9: //Pin9 - OC1A - Timer1
      if(enable==1){
        TCCR1A |=_BV(COM1A1); //Timer output on pin 9 ON
      } else if(enable==0){
        TCCR1A &=~_BV(COM1A1); //Timer output on pin 9 OFF
      }
      break;
    case 10: //Pin10 - OC1B - Timer1
      if(enable==1){
        TCCR1A |=_BV(COM1B1); //Timer output on pin 10 ON
      } else if(enable==0){
        TCCR1A &=~_BV(COM1B1); //Timer output on pin 10 OFF
      }
      break;
    case 11: //Pin11 - OC2A - Timer2
      if(enable==1){
        TCCR2A |=_BV(COM2A1); //Timer output on pin 11 ON
      } else if(enable==0){
        TCCR2A &=~_BV(COM2A1); //Timer output on pin 11 OFF
      }
      break;
    case 3: //Pin3 - OC2B - Timer2
      if(enable==1){
        TCCR2A |=_BV(COM2B1); //Timer output on pin 3 ON
      } else if(enable==0){
        TCCR2A &=~_BV(COM2B1); //Timer output on pin 3 OFF
      }
      break;
    default:
      break;
  }
}
```

ご利用規約

本規約は、お客様と東芝デバイス＆ストレージ株式会社（以下「当社」といいます）との間で、当社半導体製品を搭載した機器を設計する際に参考となるドキュメント及びデータ（以下「本リファレンスデザイン」といいます）の使用に関する条件を定めるものです。お客様は本規約を遵守しなければなりません。本リファレンスデザインをダウンロードすることをもって、お客様は本規約に同意したものとみなされます。なお、本規約は変更される場合があります。当社は、理由の如何を問わずいつでも本規約を解除することができます。本規約が解除された場合は、お客様は、本リファレンスデザインを破棄しなければなりません。またお客様が本規約に違反した場合は、お客様は、本リファレンスデザインを破棄し、その破棄したことを証する書面を当社に提出しなければなりません。

第１条 禁止事項

お客様の禁止事項は、以下の通りです。

1. 本リファレンスデザインは、機器設計の参考データとして使用されることを意図しています。信頼性検証など、それ以外の目的には使用しないでください。
2. 本リファレンスデザインを販売、譲渡、貸与等しないでください。
3. 本リファレンスデザインは、高低温・多湿・強電磁界などの対環境評価には使用できません。
4. 本リファレンスデザインを、国内外の法令、規則及び命令により、製造、使用、販売を禁止されている製品に使用しないでください。

第２条 保証制限等

1. 本リファレンスデザインは、技術の進歩などにより予告なしに変更されることがあります。
2. 本リファレンスデザインは参考用のデータです。当社は、データおよび情報の正確性、完全性に関して一切の保証をいたしません。
3. 半導体素子は誤作動したり故障したりすることがあります。本リファレンスデザインを参考に機器設計を行う場合は、誤作動や故障により生命・身体・財産が侵害されることのないように、お客様の責任において、お客様のハードウェア・ソフトウェア・システムに必要な安全設計を行うことをお願いします。また、使用されている半導体素子に関する最新の情報（半導体信頼性ハンドブック、仕様書、データシート、アプリケーションノートなど）をご確認の上、これに従ってください。
4. 本リファレンスデザインを参考に機器設計を行う場合は、システム全体で十分に評価し、お客様の責任において適用可否を判断して下さい。当社は、適用可否に対する責任は負いません。
5. 本リファレンスデザインは、その使用に際して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または実施権の許諾を行うものではありません。
6. 当社は、本リファレンスデザインに関して、明示的にも黙示的にも一切の保証（機能動作の保証、商品性の保証、特定目的への合致の保証、情報の正確性の保証、第三者の権利の非侵害保証を含むがこれに限らない。）をせず、また当社は、本リファレンスデザインに関する一切の損害（間接損害、結果的損害、特別損害、付随的損害、逸失利益、機会損失、休業損、データ喪失等を含むがこれに限らない。）につき一切の責任を負いません。

第３条 輸出管理

お客様は本リファレンスデザインを、大量破壊兵器の開発等の目的、軍事利用の目的、あるいはその他軍事用途の目的で使用してはなりません。また、お客様は「外国為替及び外国貿易法」、「米国輸出管理規則」等、適用ある輸出関連法令を遵守しなければなりません。

第４条 準拠法

本規約の準拠法は日本法とします。