ブラシレスモータスタータキット(RX71M)
[ソフトウェア編]
取扱説明書

ルネサス エレクトロニクス社 RX71M(QFP-100 ピン)搭載
ブラシレスモータスタータキット

-本書を必ずよく読み、ご理解された上でご利用ください

株式会社北斗電子

REV.1.0.0.0

ー目　次ー

注意事項 **1**

安全上のご注意 **2**

CD 内容 **4**

注意事項 **4**

1.　チュートリアル **5**

1.1.　はじめに 5
1.2.　マイコンボード初期設定 6
1.3.　モータに電流を流す 19
1.4.　A/D 変換と PWM を試す 22
1.5.　ホールセンサの値をみる 30
1.6.　モータを回してみる 34
1.7.　過電流・過熱・短絡保護の動作 40
1.8.　相電圧・相電流の観測 45

2.　サンプルプログラム **48**

2.1.　プログラム仕様 48
2.2.　関数仕様 49
2.2.1.　全体及び main 関数 49
2.2.2.　モータ制御関数 50
2.2.3.　割り込み関数 52
2.2.4.　その他内部で呼び出される関数 54
2.2.5.　グローバル変数 55
2.2.6.　プログラムの動作を制御する定義値 57
2.2.1.　プログラムで使用しているマイコン機能 59
2.3.　モータ制御アルゴリズム 60
2.3.1.　始動制御 60
2.3.2.　回転数制御 61
2.4.　安全機構 61
2.4.1.　過電流保護 61
2.4.2.　過熱保護 61

取扱説明書改定記録 62
お問合せ窓口 62

注意事項

本書を必ずよく読み、ご理解された上でご利用ください

【ご利用にあたって】

1. 本製品をご利用になる前には必ず取扱説明書をよく読んで下さい。また、本書は必ず保管し、使用上不明な点がある場合は再読し、よく理解して使用して下さい。
2. 本書は株式会社北斗電子製マイコンボードの使用方法について説明するものであり、ユーザシステムは対象ではありません。
3. 本書及び製品は著作権及び工業所有権によって保護されており、全ての権利は弊社に帰属します。本書の無断複写・複製・転載はできません。
4. 弊社のマイコンボードの仕様は全て使用しているマイコンの仕様に準じております。マイコンの仕様に関しましては製造元にお問い合わせ下さい。弊社製品のデザイン・機能・仕様は性能や安全性の向上を目的に、予告無しに変更することがあります。また価格を変更する場合や本書の図は実物と異なる場合もありますので、御了承下さい。
5. 本製品のご使用にあたっては、十分に評価の上ご使用下さい。
6. 未実装の部品に関してはサポート対象外です。お客様の責任においてご使用下さい。

【限定保証】

1. 弊社は本製品が頒布されているご利用条件に従って製造されたもので、本書に記載された動作を保証致します。
2. 本製品の保証期間は購入戴いた日から1年間です。

【保証規定】

保証期間内でも次のような場合は保証対象外となり有料修理となります

1. 火災・地震・第三者による行為その他の事故により本製品に不具合が生じた場合
2. お客様の故意・過失・誤用・異常な条件でのご利用で本製品に不具合が生じた場合
3. 本製品及び付属品のご利用方法に起因した損害が発生した場合
4. お客様によって本製品及び付属品へ改造・修理がなされた場合

【免責事項】

弊社は特定の目的･用途に関する保証や特許権侵害に対する保証等、本保証条件以外のものは明示・黙示に拘わらず一切の保証は致し兼ねます。また、直接的・間接的損害金もしくは欠陥製品や製品の使用方法に起因する損失金・費用には一切責任を負いません。損害の発生についてあらかじめ知らされていた場合でも保証は致し兼ねます。
ただし、明示的に保証責任または担保責任を負う場合でも、その理由のいかんを問わず、累積的な損害賠償責任は、弊社が受領した対価を上限とします。本製品は「現状」で販売されているものであり、使用に際してはお客様がその結果に一切の責任を負うものとします。弊社は使用または使用不能から生ずる損害に関して一切責任を負いません。
保証は最初の購入者であるお客様ご本人にのみ適用され、お客様が転売された第三者には適用されません。よって転売による第三者またはその為になすお客様からのいかなる請求についても責任を負いません。
本製品を使った二次製品の保証は致し兼ねます。

安全上のご注意

製品を安全にお使いいただくための項目を次のように記載しています。絵表示の意味をよく理解した上でお読み下さい。

表記の意味

⚠警告	取扱を誤った場合、人が死亡または重傷を負う危険が切迫して生じる可能性がある事が想定される
⚠注意	取扱を誤った場合、人が軽傷を負う可能性又は、物的損害のみを引き起こすが可能性がある事が想定される

絵記号の意味

	一般指示 使用者に対して指示に基づく行為を強制するものを示します		**一般禁止** 一般的な禁止事項を示します
	電源プラグを抜く 使用者に対して電源プラグをコンセントから抜くように指示します		**一般注意** 一般的な注意を示しています

⚠警告

以下の警告に反する操作をされた場合、本製品及びユーザシステムの破壊・発煙・発火の危険があります。マイコン内蔵プログラムを破壊する場合もあります。

1. 本製品及びユーザシステムに電源が入ったままケーブルの抜き差しを行わないでください。
2. 本製品及びユーザシステムに電源が入ったままで、ユーザシステム上に実装されたマイコンまたはIC等の抜き差しを行わないでください。
3. 本製品及びユーザシステムは規定の電圧範囲でご利用ください。
4. 本製品及びユーザシステムは、コネクタのピン番号及びユーザシステム上のマイコンとの接続を確認の上正しく扱ってください。

発煙・異音・異臭にお気付きの際はすぐに使用を中止してください。

電源がある場合は電源を切って、コンセントから電源プラグを抜いてください。そのままご使用すると火災や感電の原因になります。

⚠注意

以下のことをされると故障の原因となる場合があります。

1. 静電気が流れ、部品が破壊される恐れがありますので、ボード製品のコネクタ部分や部品面には直接手を触れないでください。
2. 次の様な場所での使用、保管をしないでください。

   ホコリが多い場所、長時間直射日光があたる場所、不安定な場所、衝撃や振動が加わる場所、落下の可能性がある場所、水分や湿気の多い場所、磁気を発するものの近く
3. 落としたり、衝撃を与えたり、重いものを乗せないでください。
4. 製品の上に水などの液体や、クリップなどの金属を置かないでください。
5. 製品の傍で飲食や喫煙をしないでください。

ボード製品では、裏面にハンダ付けの跡があり、尖っている場合があります。

取り付け、取り外しの際は製品の両端を持ってください。裏面のハンダ付け跡で、誤って手など怪我をする場合があります。

CD メディア、フロッピーディスク付属の製品では、故障に備えてバックアップ(複製)をお取りください。

製品をご使用中にデータなどが消失した場合、データなどの保証は一切致しかねます。

アクセスランプがある製品では、アクセスランプが点灯中に電源を切ったり、パソコンをリセットをしないでください。

製品の故障の原因となったり、データが消失する恐れがあります。

本製品は、医療、航空宇宙、原子力、輸送などの人命に関わる機器やシステム及び高度な信頼性を必要とする設備や機器などに用いられる事を目的として、設計及び製造されておりません。

医療、航空宇宙、原子力、輸送などの設備や機器、システムなどに本製品を使用され、本製品の故障により、人身や火災事故、社会的な損害などが生じても、弊社では責任を負いかねます。お客様ご自身にて対策を期されるようご注意ください。

CD 内容

添付「ソフトウェア CD」には、以下の内容が収録されています。

・チュートリアル
　TUTORIAL フォルダ以下
・サンプルプログラム
　SAMPLE フォルダ以下
・マニュアル
　manual フォルダ以下

注意事項

本マニュアルに記載されている情報は、ブラシレスモータスタータキットの動作例・応用例を説明したものです。

お客様の装置・システムの設計を行う際に、本マニュアルに記載されている情報を使用する場合は、お客様の責任において行ってください。本マニュアルに記載されている情報に起因して、お客様または第三者に損害が生じた場合でも、当社は一切その責任を負いません。

本マニュアルに、記載されている事項に誤りがあり、その誤りに起因してお客様に損害が生じた場合でも、当社は一切その責任を負いません。

本マニュアルの情報を使用した事に起因して発生した、第三者の著作権、特許権、知的財産権侵害に関して、当社は一切その責任を負いません。当社は、本マニュアルに基づき、当社または第三者の著作権、特許権、知的財産権の使用を許諾する事はありません。

1. チュートリアル

1.1. はじめに

本チュートリアルでは、マイコンの基本的なプログラムを説明致しますが、ルネサスエレクトロニクスの Web から「RX71M グループ　ユーザーズマニュアル　ハードウェア編」を予めダウンロードして、手元に用意してください。マイコンの詳細な設定は、このマニュアル（以下「ハードウェアマニュアル」と記載する）に記載されています。

また、本マニュアルで説明するプログラムは、ルネサスエレクトロニクス CS+の環境向けに作成されていますので、CS+の環境を PC にインストールしておいてください。なお、CS+のインストール等は、ルネサスエレクトロニクス社のマニュアルを参照してください。

マイコンの基本的なプログラムは行えるが、モータ制御は初めてという方を想定した構成となっており、ホールセンサを使用してブラシレスモータを回転させる方法。モータが動作しているときの、電圧や電流を観測する方法。また、モータドライバボードの保護回路の使用方法の習得を目的にしています。

以下が、本チュートリアルで学べる事柄となります。

・マイコンボードを動作させるための初期設定（クロック設定やモジュールスタンバイの解除）
・モータのコイルに流す電流を制御する方法
・A/D 変換
・PWM（パルス幅変調）
・ホールセンサの値を取得する方法
・温度値の取得
・過電流検出の方法
・モータが動作しているときの相電圧・相電流の取得

1.2. マイコンボード初期設定

参照プロジェクト：TUTORIAL1

本キットに付属のマイコンボード(HSBRX71M100)は、RX71M グループのマイコンを搭載しており、クロック周波数 240MHz で動作させることが出来ます。マイコンの動作モードやクロック周波数は、プログラムで設定する必要があります。

本プロジェクトでは、

・マイコンボードの初期設定
・基本的な動作

を確認します。（※本プロジェクトは、モータドライバボード・接続ボードをつないだ状態で実行してください）

プロジェクト一式をローカルのディスクにコピーし、

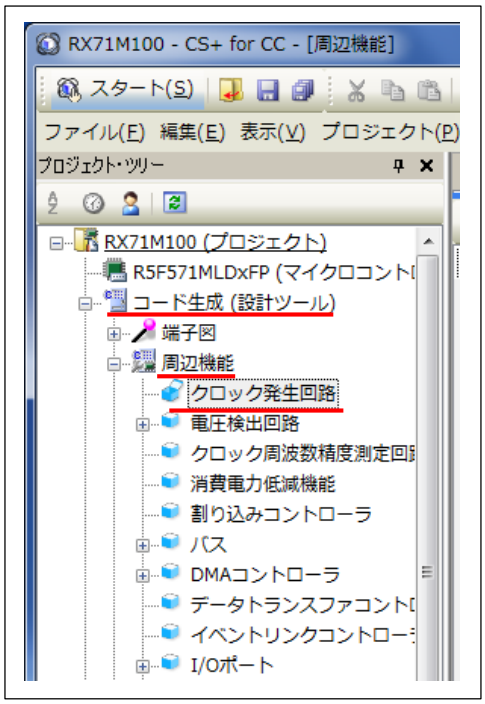
BLS_MOTOR1_RX71M100¥RX71M100.mtpj

を、ダブルクリック（または、CS+から「プロジェクト」－「プロジェクトを開く」で）して、プロジェクトを開きます。

・E1 を使用している場合
マイコンボードと E1 を接続してください。
モータドライバボードに電源を投入してください。

マイコンで動作するプログラムを作成する場合、クロック等の初期設定が必要です。CS+では、コード生成機能があり、面倒なレジスタ設定のプログラムコードをメニューで選択して生成する事ができます。

プロジェクトを開き、メニューを展開し

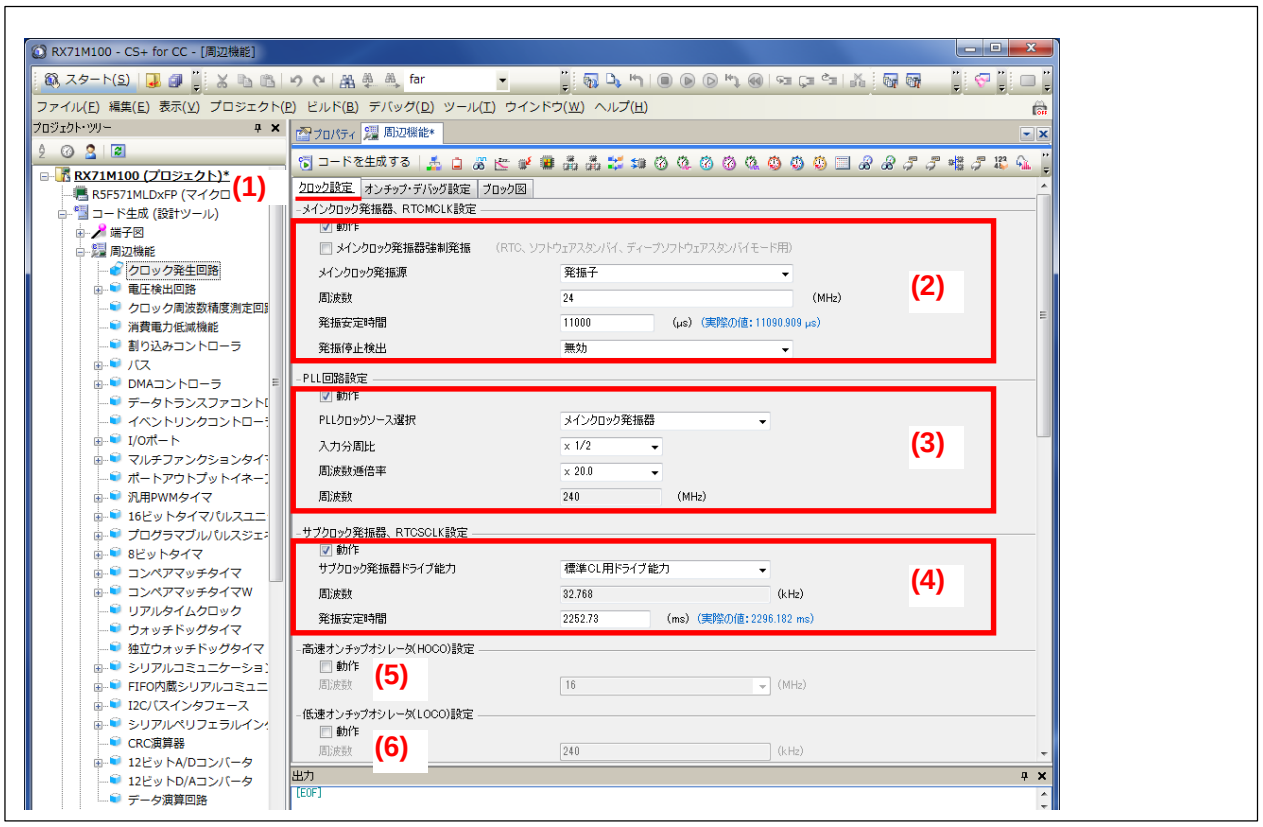


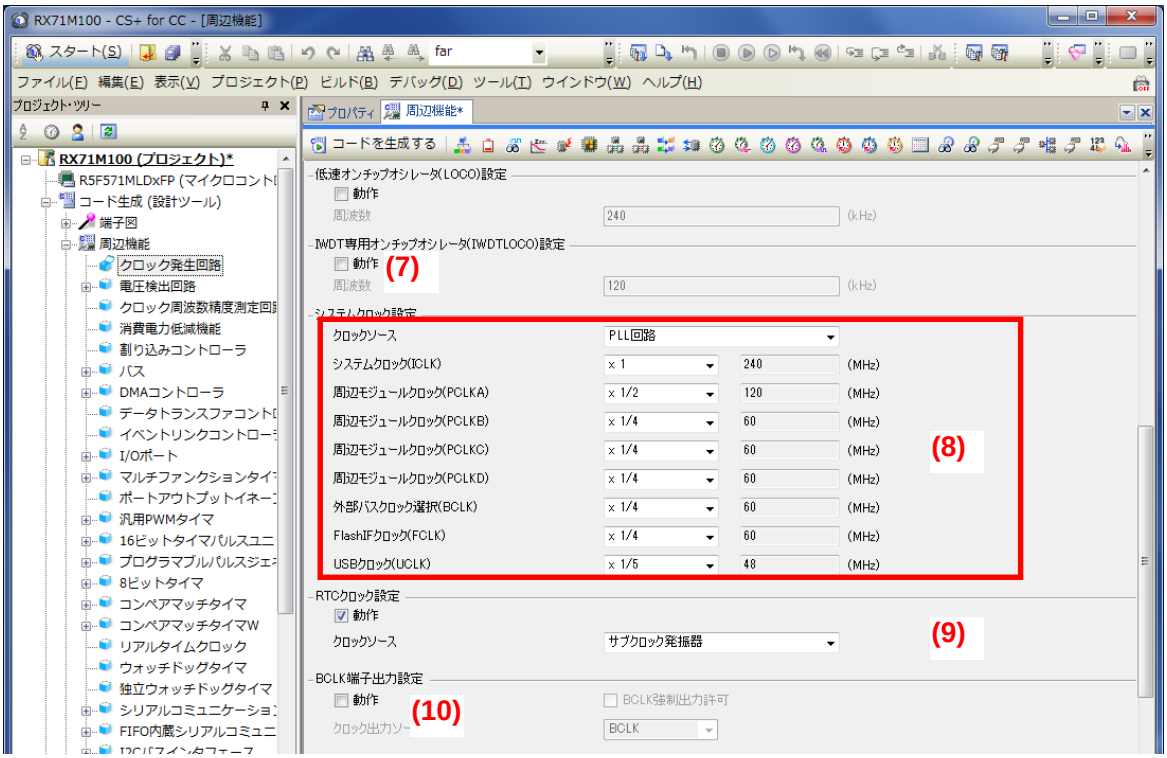
コード生成

周辺機能

クロック発生回路

をダブルクリックしてください。（メニューが展開していないときは、＋を押して展開してください）





選択項目は

(1)クロック設定タブ

(2)メインクロック発振器、RTCMCLK 設定

動作　「チェック」

メインクロック発振器強制発振　「ー（チェックを外す）」

メインクロック発振源　「発振子」

周波数　「24」

発振安定時間　「11000」（デフォルト値のまま）

発振停止検出　「無効」

(3)PLL 回路設定

動作　「チェック」

PLL クロックソース選択　「メインクロック発振器」

入力分周比　「x 1/2」

逓倍比　「x 20.0」

(4)サブクロック発振器、RTCSCLK設定

動作　「チェック」

サブクロック発振器ドライブ能力　「標準 CL ドライブ入力」

発振安定時間　「2252.73」（デフォルト値のまま）

(5)高速オンチップオシレータ（HOCO)設定

動作　「ー（チェックを外す）」　(*1)

(*1)本チュートリアルでは未使用です、設定はどちらでも構いません

(6)低速オンチップオシレータ(LOCO)設定

動作　「ー(チェックを外す)」　(*1)

(7)IWDT 専用オンチップオシレータ(IWDTLOCO)設定

動作　「ー(チェックを外す)」　(*1)

(8)システムクロック設定

クロックソース　PLL 回路

システムクロック(ICLK)　「x 1」

周辺モジュールクロック(PCLKA)　「x 1/2」

周辺モジュールクロック(PCLKB)　「x 1/4」

周辺モジュールクロック(PCLKC)　「x 1/4」

周辺モジュールクロック(PCLKD)　「x 1/4」

外部バスクロック(BCLK)　「x 1/4」

FlashIF クロック(FCLK)　「x 1/4」

USB クロック(UCLK)　「x 1/5」

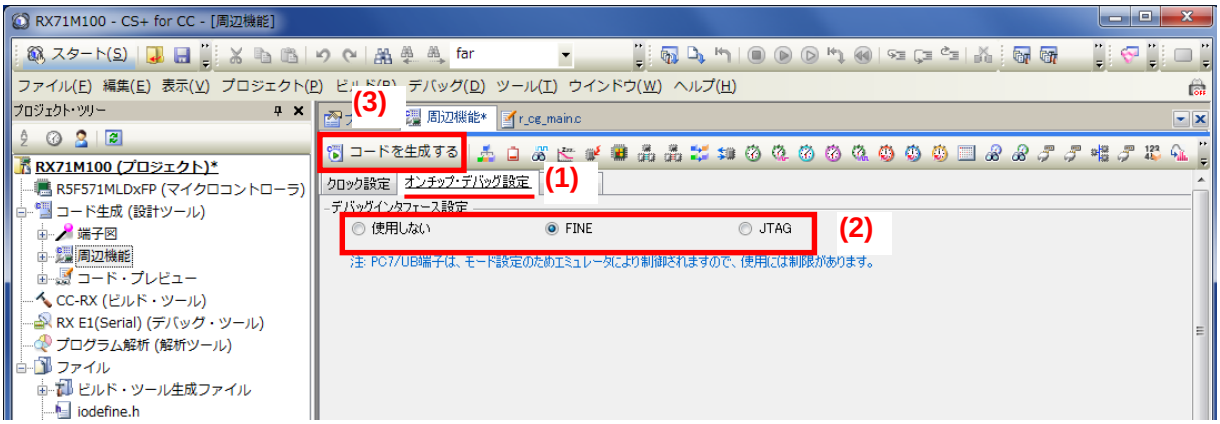
(9)RTCクロック設定

動作　「チェック」

クロックソース　「サブクロック発振器」

(10)BCLK 端子出力設定

動作　「ー(チェックを外す)」

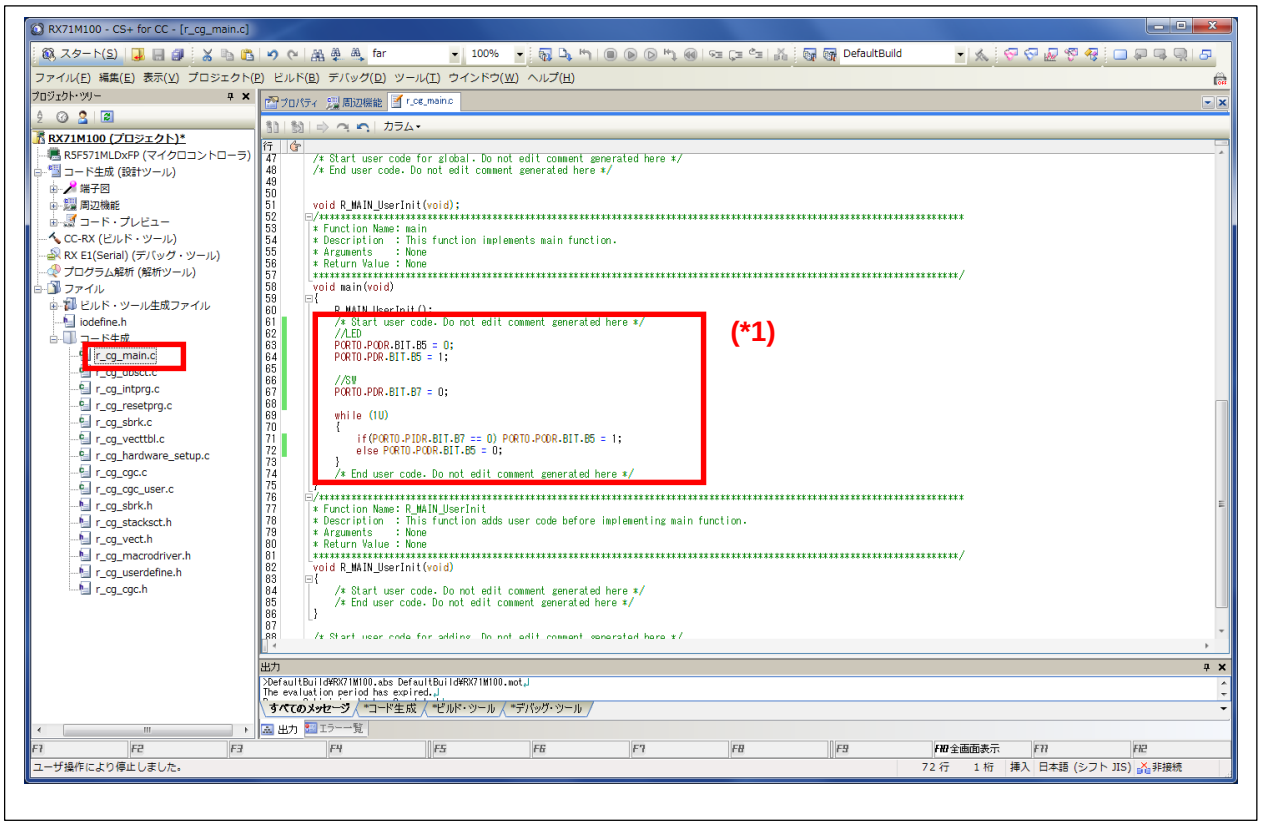


(1)オンチップ・デバッグ設定タブ

(2)デバッグインタフェース設定　「FINE」

(3)コード生成　ボタンを押す　**[コード生成機能で設定項目を変更した場合必ずコード生成ボタンを押してください]**

上記設定で、CS+がクロック設定のプログラムコードを自動で生成します。(コード生成プラグインをインストール及び有効にする必要があります。詳細は CS+のマニュアルを参照ください。)



ファイル

コード生成　以下の

r_cg_main.c

をダブルクリックします。

CS+のコード生成機能により、自動生成されたソースコードにユーザ側でソースを追加する場合は、(*1)の部分

```
    /* Start user code. Do not edit comment generated here */
    while (1U)
    {
        ;
    }
    /* End user code. Do not edit comment generated here */
```

Start user code. から End user code. の中に入れる必要があります。(この領域から外れた場所に追加すると、コード生成ボタンを押したときのコード再生成で、ユーザ側で追加したものが消えてしまいます。

上記、while の部分に、本チュートリアルの目的である、スイッチの読み取り及び LED の点灯制御のソースをいれてみます。

```
/* Start user code. Do not edit comment generated here */

//LED
PORT0.PODR.BIT.B5 = 0;
PORT0.PDR.BIT.B5 = 1;

//SW
PORT0.PDR.BIT.B7 = 0;

while (1U)
{
    if(PORT0.PIDR.BIT.B7 == 0) PORT0.PODR.BIT.B5 = 1;
    else PORT0.PODR.BIT.B5 = 0;
}
/* End user code. Do not edit comment generated here */
```

LED が接続されているポート
(P05)を出力ポートに設定，初期出力は 0

SW が接続されているポート
(P07)を入力ポートに設定

while ループ（本プログラムでは永遠にこの中をループ）

P07(SW2)が押されていれば(L)なら
P05(LED)を ON(H)
（押されていなけば OFF）

※I/O ポートの初期設定は、コード生成機能を使用して設定する事も可能ですが、記述は単純ですので、ここではプログラムコードを書き下しています

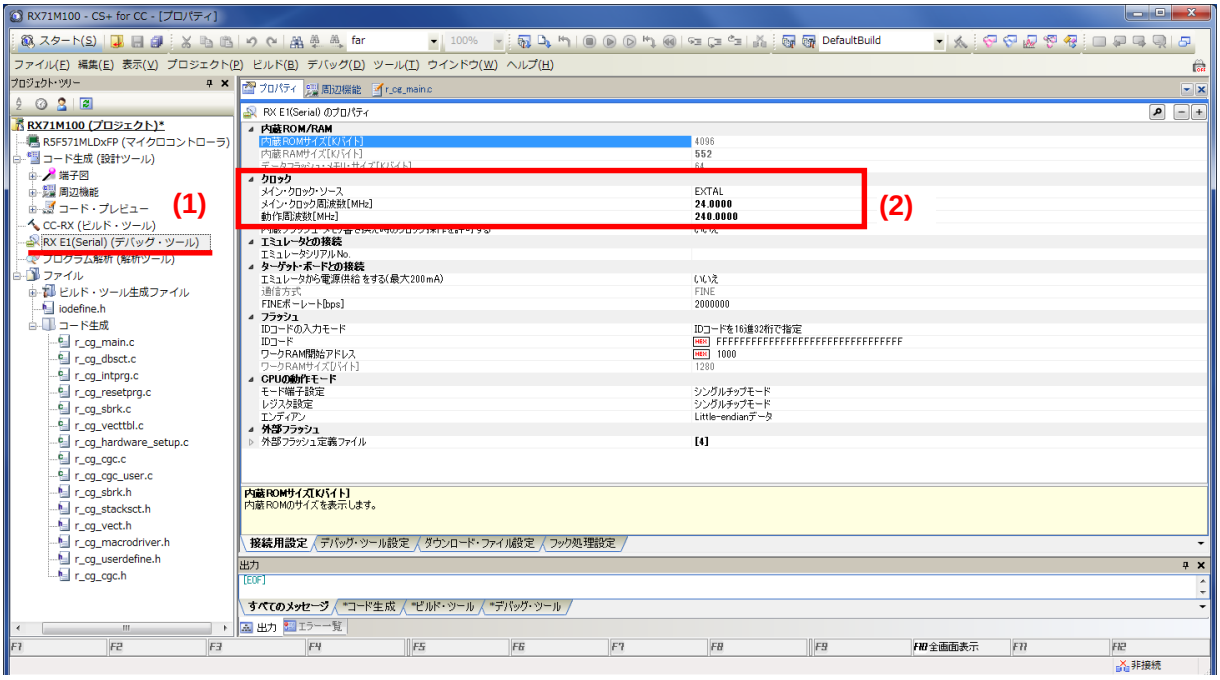
本チュートリアルを実行して確認する際は

J11：ショート

SW3-4：ON 側

としてください。

デバッガ(E1)を接続する際は以下の設定を行ってください。



(1)RX E1(Serial)（デバッグ・ツール）を選択

(2)クロック

メインクロックソース　「EXTAL」を選択

メインクロック周波数　「24.0」

動作周波数　「240.0」

「デバッグ」－「ビルド&デバッグ・ツールへダウンロード」（もしくは、F6 キーを押す）
「デバッグ」－「実行」（もしくは、F5 キーを押す）

上記操作で、プログラムがマイコンボード上で実行されています。

SW2 を ON/OFF させてみてください。SW2 に連動して、LED(D9)が、点灯・消灯した場合プログラムは、マイコンボードにダウンロードされ、実行されていることを示します。

・E1 をお持ちでない場合（USB-ADAPTER または、市販の USB-Serial 変換機器を使って書き込みを行う場合）
CS+のメニューから
「ビルド」－「ビルド・プロジェクト」（もしくは、F7 キーを押す）

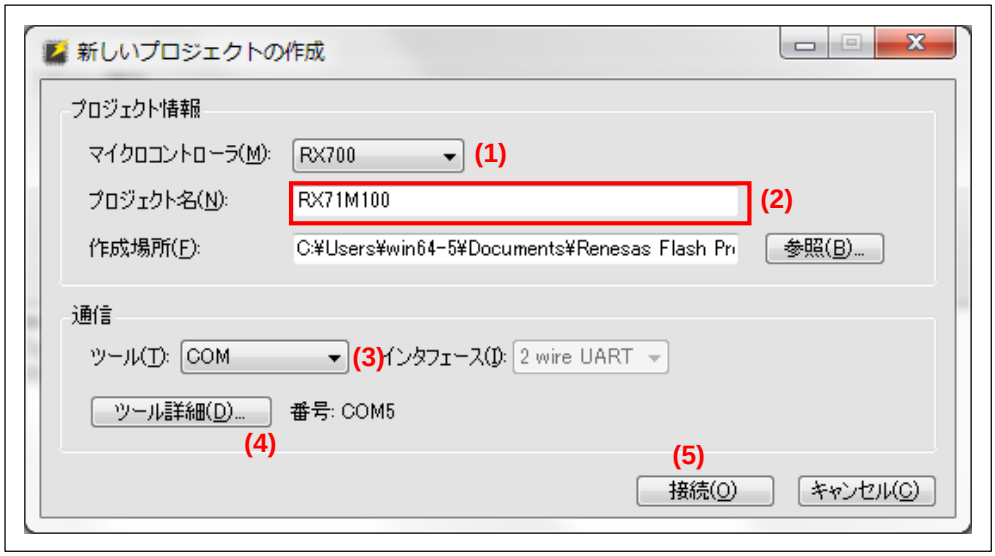
BLS_MOTOR1_RX71M100¥DefaultBuild¥RX71M100.mot
ファイルが出来上がっていることを確認してください。

マイコンボード上の SW3(DIP-SW)の 1,2 を ON 側としてください。
マイコンボード上の J4(FLASH IF)に USB-ADAPTER を挿してください。(*1)
モータドライバボードに電源を投入してください。

(*1)
市販の USB-Serial 変換機器（0-5V 系の信号の送受信が出来るタイプのもの）を使用する場合は
GND は、J4-20P コネクタの 2,4,6,8,10,12,14,16 のいずれかに接続。
USB-Serial 変換の TX は、J4-20P コネクタの 17 ピンに接続。
USB-Serial 変換の RX は、J4-20P コネクタの 15 ピンに接続。
※TX, RX の表記は、USB-Serial 変換機器基準の表記です（マイコンでは逆となります）

RenesasFlashProgrammer(RFP)を起動し、

「ファイル」ー「新しいプロジェクトを作成」



(1)RX700　を選択

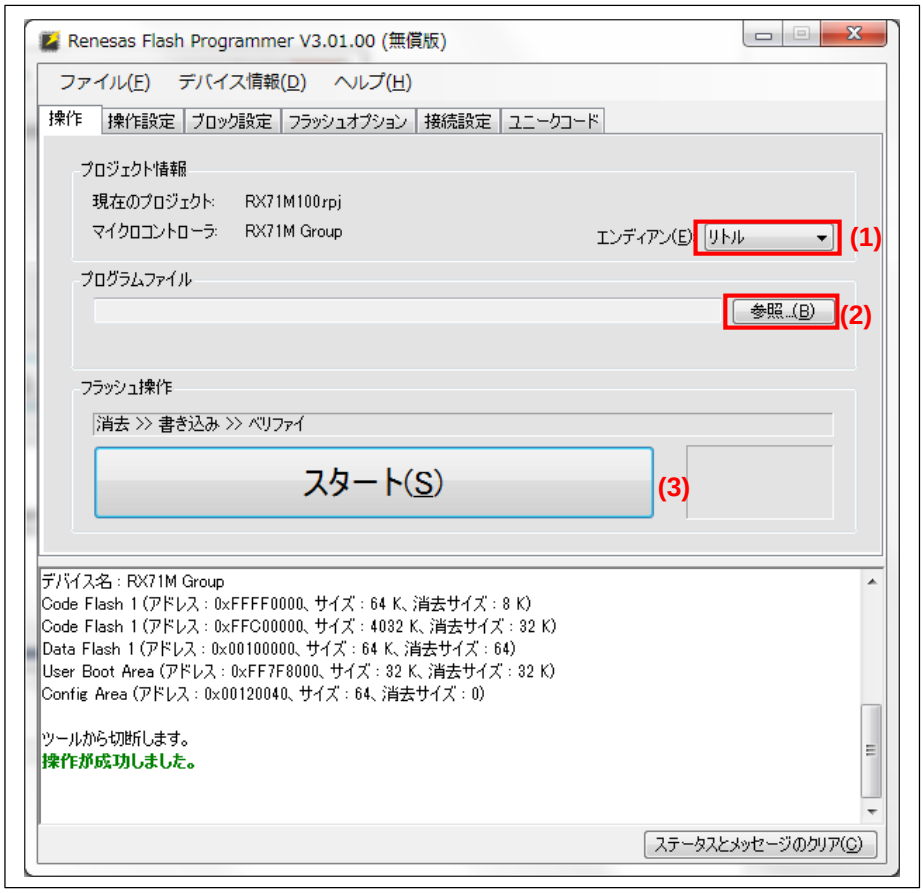
(2)適当な名称を入力

(3)COM　を選択

(4)ツール詳細のボタンを押し、USB-Serial 変換機器が接続されている COM ポート番号を選択

（番号はデバイスマネージャ等で予め確認してください）

(5)接続ボタンを押す



(1)リトル　を選択

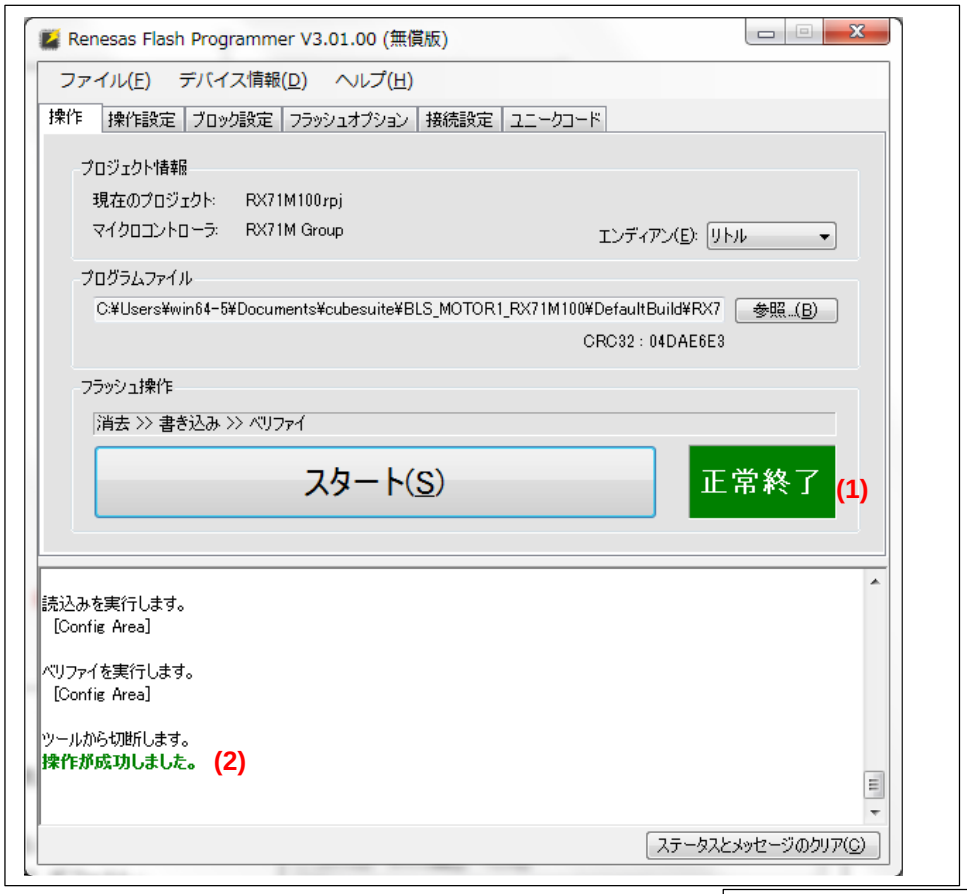
(2)TUTORIAL1 で生成した mot ファイルを指定

（BLS_MOTOR1_RX71M100¥DefaultBuild　の下に、RX24T100.mot　というファイル名でファイルがあるはずです）

ーーボード上のリセットスイッチを押す(*1)ーー（または電源再投入）

(3)スタートボタンを押す

(*1)リセットスイッチ(SW1)はコネクタに挟まれた配置となっていますが、先の細い棒等で押してください



(1)正常終了

(2)操作が成功しました

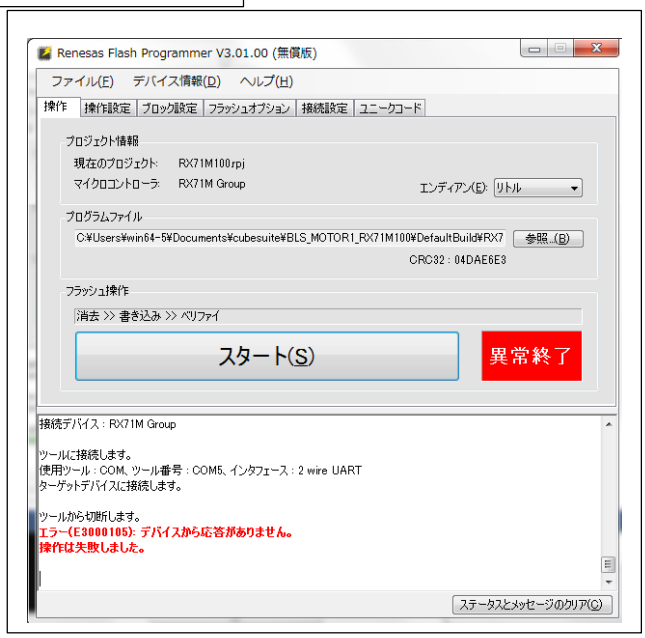
の表示となれば良い

※フラッシュ操作は、デフォルト

（消去>>書き込み>>ベリファイ）の

ままで構いません

スタートボタンを押す前にマイコンボードの
リセットスイッチを押さないと
右記のエラーとなります
（この場合マイコンボードのリセットを
押して再度スタートボタンを押してください）

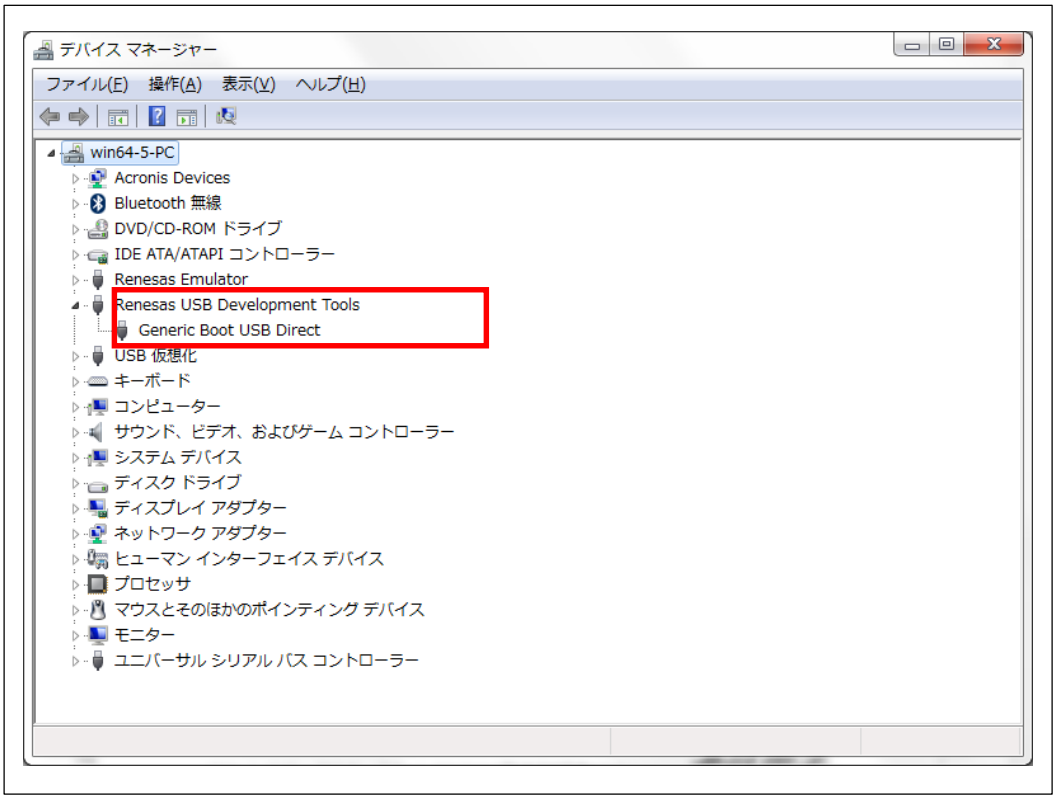


マイコンボード上の SW3(DIP-SW)の 1,2 を数字側(OFF)としてください。
モータドライバボードに電源を再投入してください。

・E1 をお持ちでない場合（USB-miniB ケーブルを使って書き込みを行う場合）
CS+のメニューから
「ビルド」ー「ビルド・プロジェクト」（もしくは、F7 キーを押す）

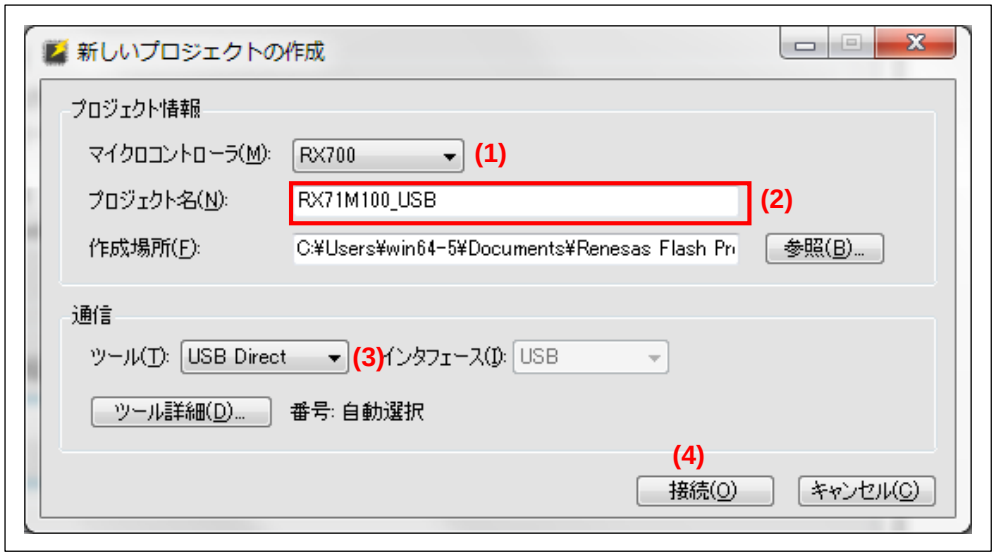
BLS_MOTOR1_RX71M100¥DefaultBuild¥RX71M100.mot
ファイルが出来上がっていることを確認してください。

モータドライバボードの電源を切ってください。
マイコンボードの電源コネクタ(J7)を抜いてください。
マイコンボード上の J14 をショートとしてください。
マイコンボード上の SW3(DIP-SW)の 1 を ON 側としてください。
マイコンボード上の SW3(DIP-SW)の 2 を数字側(OFF)としてください。
マイコンボード上の J9（USB-miniB コネクタ，基板裏面）に USB-miniB ケーブル（市販品）挿して PC と接続してください。



Renesas USB Development Tools
Generic Boot USB Direct
のデバイスがデバイスマネージャ上で見えれば問題ありません。
（見えない場合は、別途ドライバのインストールが必要です。RenesasFlashProgrammer を再インストールしてみてください。）

「ファイル」ー「新しいプロジェクトを作成」

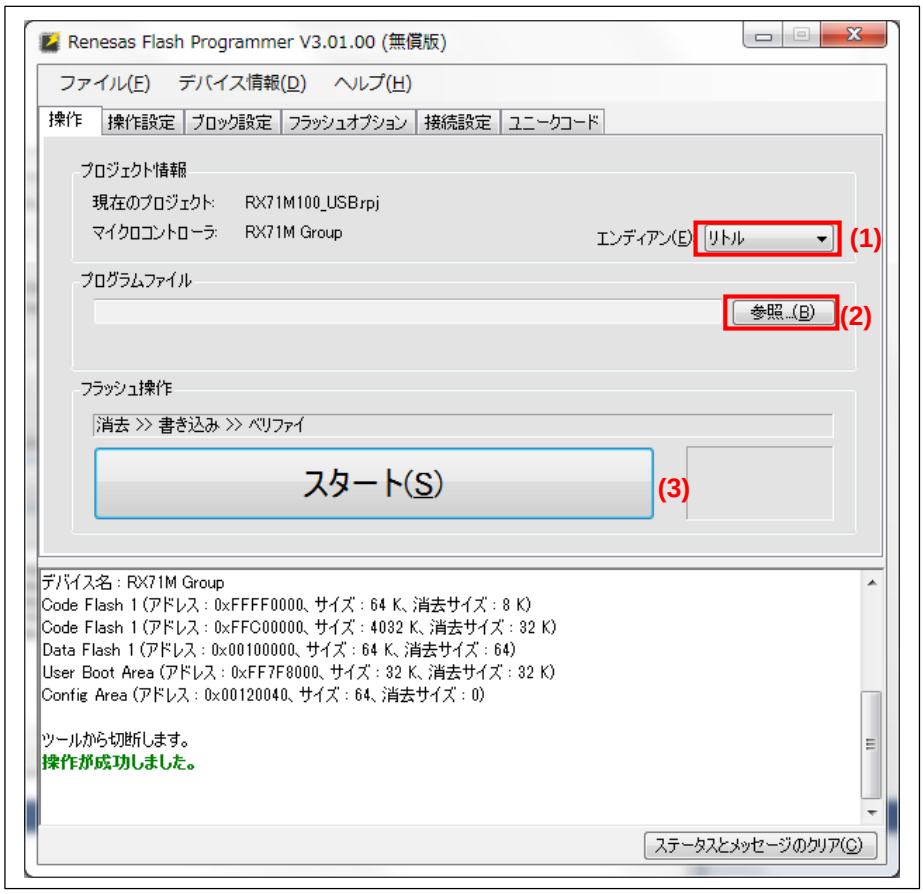


(1)RX700　を選択

(2)適当な名称を入力

(3)COM　を選択

(4)接続ボタンを押す



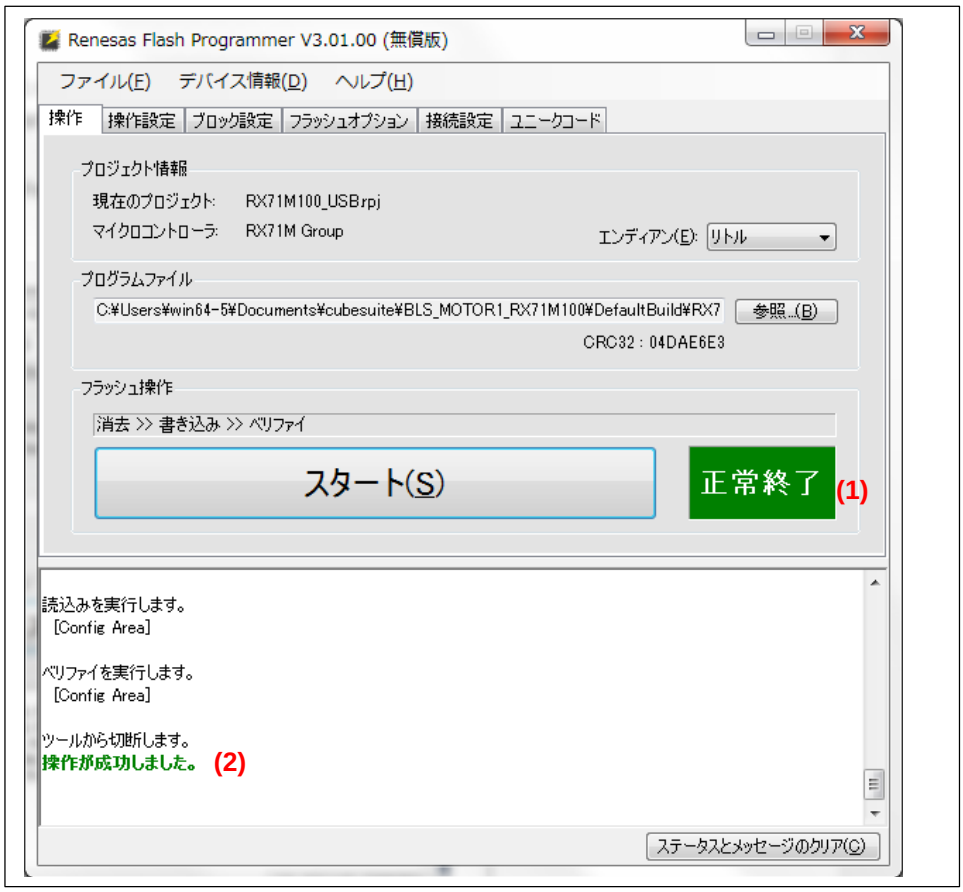
(1)リトル　を選択

(2)TUTORIAL1 で生成した mot ファイルを指定

（BLS_MOTOR1_RX71M100¥DefaultBuild　の下に、RX24T100.mot　というファイル名でファイルがあるはずです）

ーーボード上のリセットスイッチを押すーー

(3)スタートボタンを押す



USB-mini ケーブルを外してください。

マイコンボード上の SW3(DIP-SW)の 1 を数字側(OFF)としてください。

J7 に電源ケーブルを挿してください。

モータドライバボードに電源を投入してください。

プログラムの動作としては、E1 を使った場合と同じです。

いかがでしょう、スイッチと LED が連動するという簡単なプログラムですが、動作したでしょうか。

動作しない場合は、
・電源が入っているか（モータドライバボード上の PL(D5)が点灯している、マイコンボード上の PL が点灯しているか）
・マイコンボードが動作モードになっているか（SW3 の 1,2 が数字側となってるか）
・SW3 の 3 が ON になっているか
・J11 がショート（ショートジャンパを挿しているか）
等を確認してください。

今回は、マイコンのクロックの設定は、CS+のコード生成機能が行いましたので、ハードウェアマニュアルを見てレジスタの値を設定する必要がありませんでした。

細かな設定をするのであれば、コード生成ではカバーが出来ないケースもありますが、メニューで選ぶだけでレジスタの設定が完了するコード生成を活用してみてください。

1.3. モータに電流を流す

参照プロジェクト：TUTORIAL2

プロジェクト一式をローカルのディスクにコピーし、

BLS_MOTOR2_RX71M100¥RX71M100.mtpj

を開いてください。プログラムのビルドと、マイコンへの書き込みは、1.2 と同様に行ってください。

プログラムを実行し、SW2 を押すと、LED(D9)が点灯（押すたびに点灯←→消灯が切り替わる）し、モータから「カチッ」という音が聞こえてくると思います。（モータの軸は一瞬動きます。指で軸に触れている状態ですと、明らかに指に振動が伝わると思います。）

BLS_MOTOR2_RX71M100¥cg_src¥r_cg_main.c

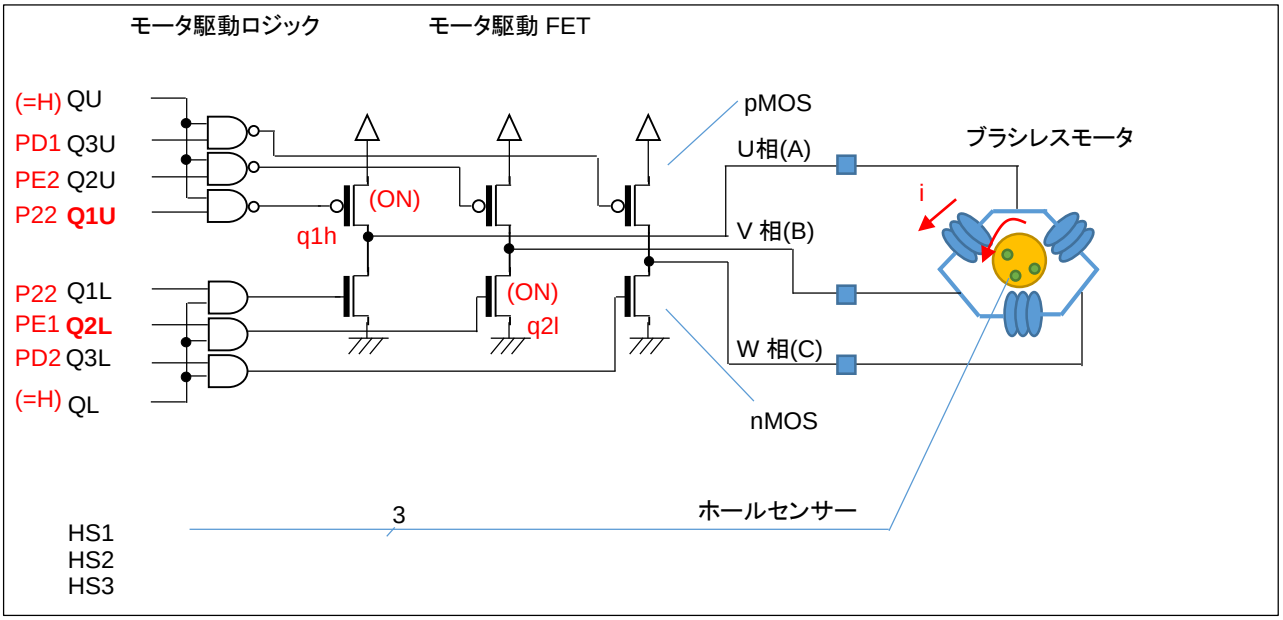
```
if(mortor_current_flag == 1)
{
    switch(state)
    {
        case 0:
            //W -> V
            PORTD.PODR.BIT.B1 = 1;
            PORTE.PODR.BIT.B1 = 1;
            for(x=0; x<width; x++) nop();
            PORTD.PODR.BIT.B1 = 0;
            PORTE.PODR.BIT.B1 = 0;
            break;
        case 1:
            //W -> U
            PORTD.PODR.BIT.B1 = 1;
            PORT2.PODR.BIT.B3 = 1;
            for(x=0; x<width; x++) nop();
            PORTD.PODR.BIT.B1 = 0;
            PORT2.PODR.BIT.B3 = 0;
            break;
        case 2:
            //V -> U
            PORTE.PODR.BIT.B2 = 1;
            PORT2.PODR.BIT.B3 = 1;
            for(x=0; x<width; x++) nop();
            PORTE.PODR.BIT.B2 = 0;
            PORT2.PODR.BIT.B3 = 0;
            break;
        case 3:
            //V -> W
            PORTE.PODR.BIT.B2 = 1;
            PORTD.PODR.BIT.B2 = 1;
            for(x=0; x<width; x++) nop();
            PORTE.PODR.BIT.B2 = 0;
            PORTD.PODR.BIT.B2 = 0;
            break;
        case 4:
            //U -> W
            PORT2.PODR.BIT.B2 = 1;
            PORTD.PODR.BIT.B2 = 1;
            for(x=0; x<width; x++) nop();
            PORT2.PODR.BIT.B2 = 0;
            PORTD.PODR.BIT.B2 = 0;
            break;
        case 5:
            //U -> V
            PORT2.PODR.BIT.B2 = 1;
            PORTE.PODR.BIT.B1 = 1;
            for(x=0; x<width; x++) nop();
            PORT2.PODR.BIT.B2 = 0;
            PORTE.PODR.BIT.B1 = 0;
            break;
        default:
            break;
    }
}
```

SW2 が押されたときにフラグ変数をセット

state は実行の度にインクリメントされて 5 までいくと 0 に戻る

モータに電流を流す（W 相から V 相に対して）
width で定義した時間空ループ（ウェイト）
電流を切る

ここで、モータに電流を U→V に流すとき（付属ケーブルでは、青→黄の方向の電流）
P22=H, PE1=H
の様に設定しています。
これは、



P22 は、U の H 側を制御しており（Q1U につながっている）、PE1 は V の L 側を制御しています（Q2L につながっている）。P22 と PE1 を 1(H)とすると、q1h, q2l の 2 つの FET が ON し、モータのコイルを経由して図の i の電流が流れます。U→V 以外の定義も同様に、3 相ある電極の 2 相間に電流を流す設定となります。

本プログラムでは、モータの軸が一瞬振れる感覚がありますが、回転には程遠いと思います。モータを回転させる制御まで、あといくつかのステップがあります。ここでは、モータ内の任意のコイルの任意の方向に電流を流す事が、プログラムで行える事を理解してください。

1.4. A/D 変換と PWM を試す

参照プロジェクト：TUTORIAL3

プロジェクト一式をローカルのディスクにコピーし、

BLS_MOTOR4_RX71M100¥RX71M100.mtpj

を開いてください。

次のチュートリアルでは、マイコンの A/D 変換（Analog to Digital 変換）機能と、PWM（Pulse Width Modulation: パルス幅変調）を試してみます。モータの制御から一旦離れますが、どちらもモータを動かすのに必要な機能となります。

本プログラムでは、

・VR の回転角に応じたパルス幅の信号が、P21/QL から出力される。

・モータドライバボード上の温度センサ(R54)の値を拾う

という動作を行います。本プログラムでは、情報を SCI1(TXD1:P26, RXD1:P30)に、シリアル通信で出力します。USB-ADAPTER（別売オプション）を、J4 に挿す、または、市販の USB-Serial モジュールを P26, P30 に接続してください（シリアル通信のモニタは必須ではありませんが、接続した場合、プログラムの動作が判り易くなると思います）。

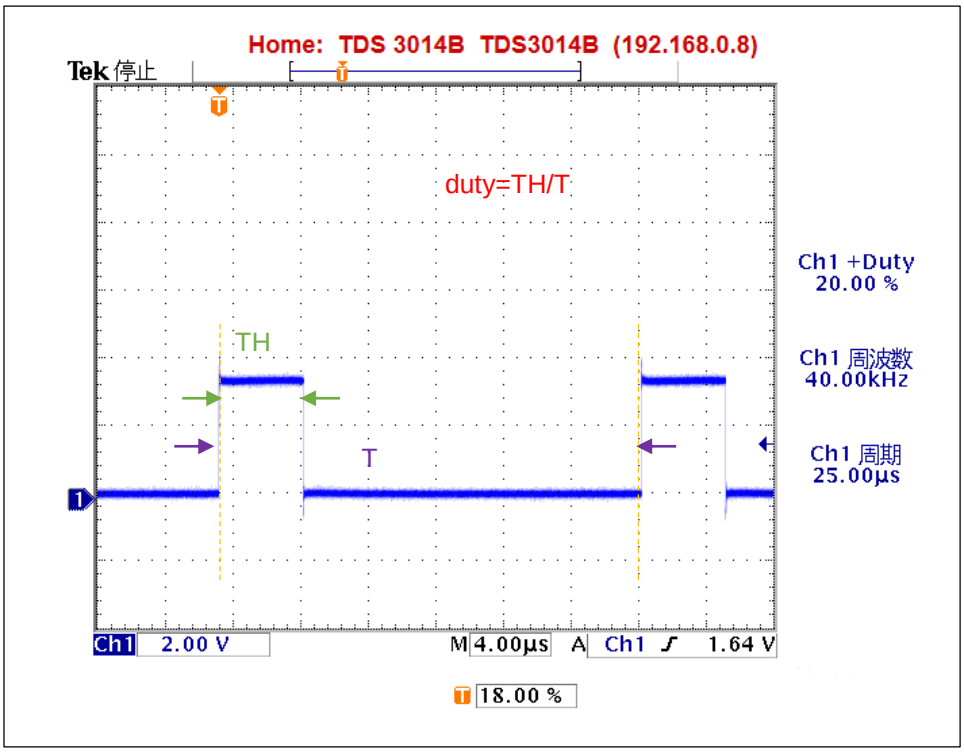
・シリアル端末から出力される情報（5 秒に 1 回更新）

```
Tempatature(A/D value) : 2111
Tempatature(degree)    : 27
VR(A/D value)          : 818
P21 duty               : 599 / 3000
```

PC 上では、teraterm 等のシリアル端末ソフトで表示してください

115,200bps, 8bit, none, 1bit の設定で表示できます

・オシロスコープで QL 端子を観測した波形



599/3000 = 20%
VR で設定した値と
波形は連動しています

Tempatature(A/D value)は、温度センサーの出力です。温度センサーの出力は、マイコンの P46/AN006 に接続されており、ここでは当該端子を AN006 の機能(A/D 変換入力)として使っています。RX71M の A/D 変換機能は、12bit となっているので、0~4095 までの値を取ります。(この値は、約 25℃のときに、2048 になります。温度が高いほど数値は大きくなります)

Tempatature(degree)は、温度センサーの A/D 変換値を摂氏温度に変換したものです。温度の変換は、ブラシレスモータスタータキットの取扱説明書に計算式を示してあります。(計算式は、exp の計算を含む手間のかかるものとなっているので、本プログラムでは予め A/D 変換値と温度の 80 点のテーブルを作成しておき、テーブルから温度を換算しています)ここでは、27℃とっていますが、ドライヤー等でモータドライバボードの温度センサ部(R54:黒いヒートシンクの下側付近にある)を暖めると、画面表示上の温度も上昇するかと思います。

VR(A/D value)は、モータドライバボード上の VR(ボリューム)を回すと、値が変わるはずです。時計回りに目一杯回すと 0。反時計回りに目一杯回すと、3722(4095×10/11)近傍になるはずです。VR は、プログラム上で値を拾いアナログ入力デバイスとして、任意の用途で使用する事ができます。

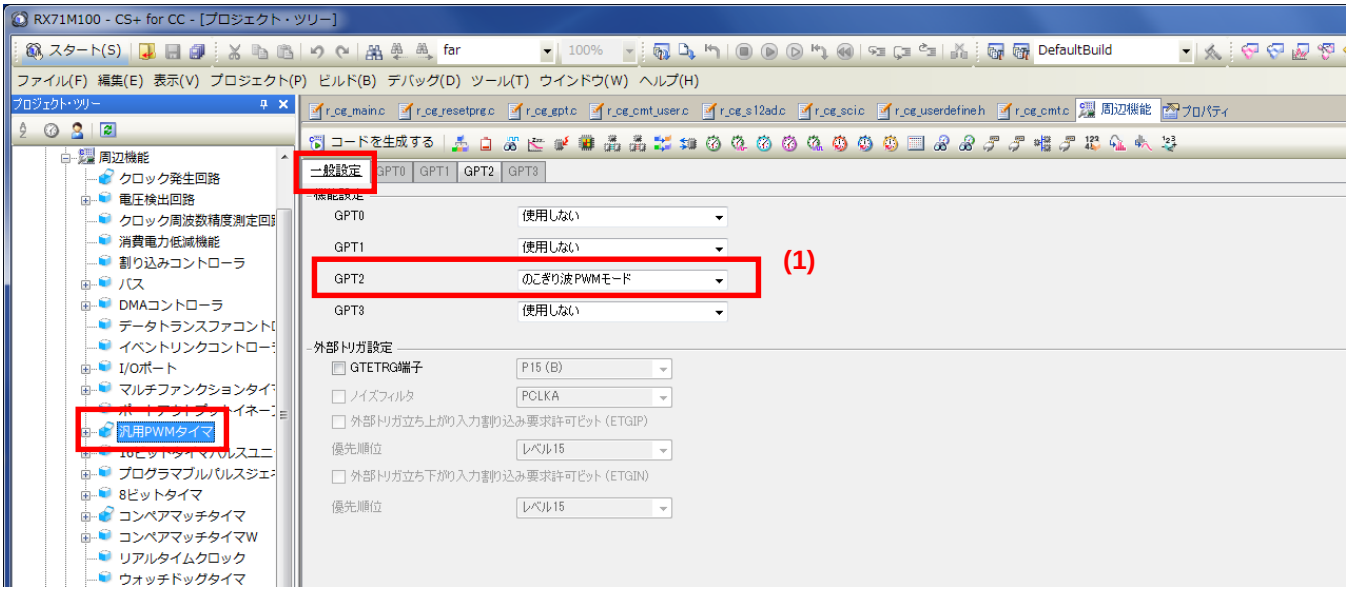
P21 duty は、P21/QL 端子の duty 値となります。この値は、VR の読み取り値に連動して変更を行っています。(VR の読み取り値/4095×100)0~90%程度まで設定可能です(本プログラムでは)。この値と、実際に P21/QL 端子から出力されるパルス波形は連動しています。

ここで、VR の読み取り値をプログラムで処理して、P21/QL 端子の duty 比を決めている事に注意願います。(プログラムの介在なしに、VR と duty が連動しているわけではありません)

GPT2.GTCCRA のレジスタ値を変えると、P21 に出力される、H パルスの幅が変わります。この様に、パルス幅を変える制御を、PWM(パルス幅変調)といい、モータの制御ではよく使用される方式です。

なお、VR→duty の変化はリアルタイムに追随(10ms に 1 回)し、画面表示は 5 秒に 1 回行っています。

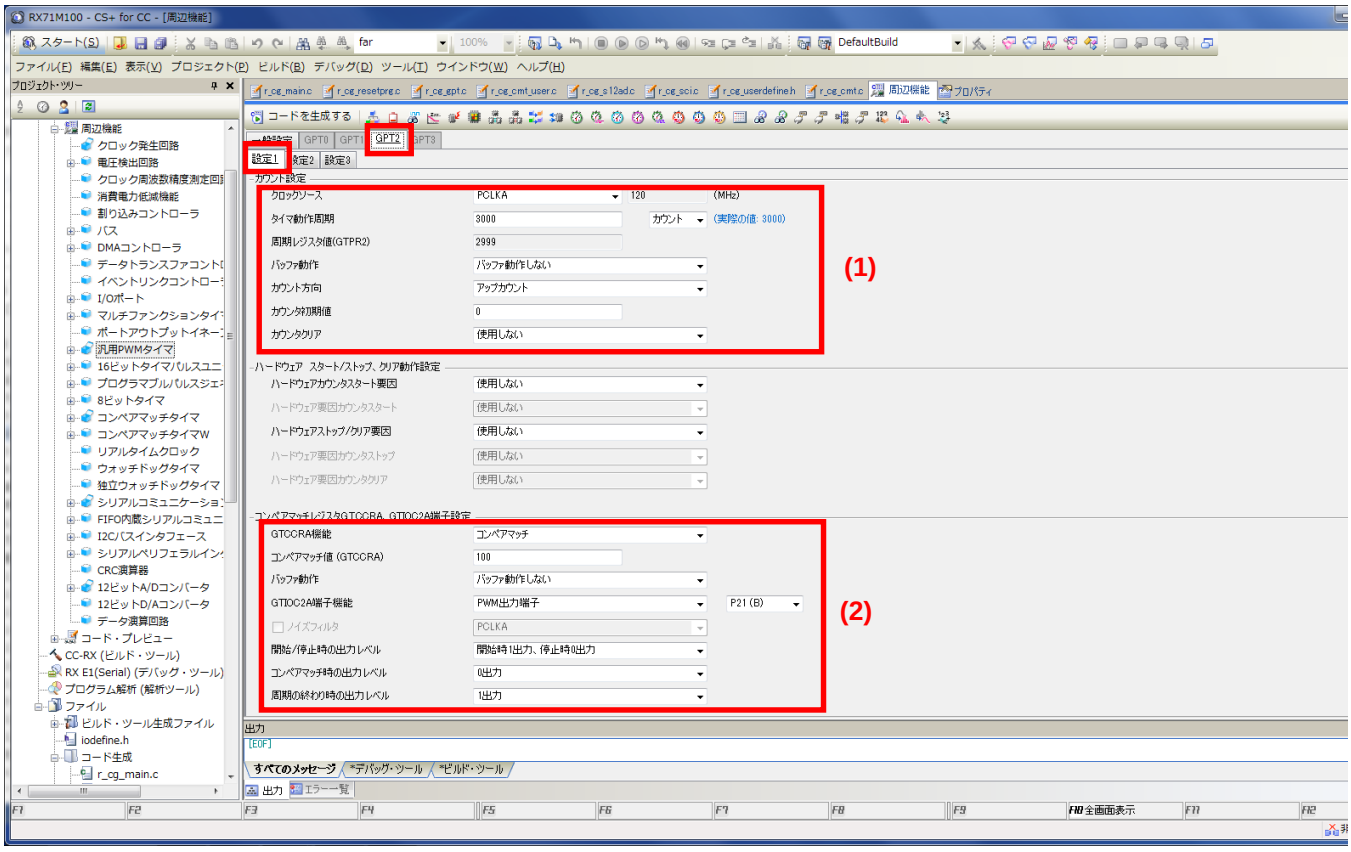
○PWM 波形出力設定（汎用 PWM タイマ）



汎用 PWM タイマ

一般設定タブ

(1)GPT2　「のこぎり波 PWM モード」



GPT2 タブ

設定 1 タブ

(1)カウント設定

クロックソース　「PCLKA」　(120MHz)

タイマ動作周期　「3000 カウント」

カウント方向　「アップカウント」

カウンタ初期値　「0」

(2)コンペアマッチレジスタ GTCCRA、GTIOC2A 端子設定

GTCCRA機能　「コンペアマッチ」

コンペアマッチ値(GTCCRA)　「100」

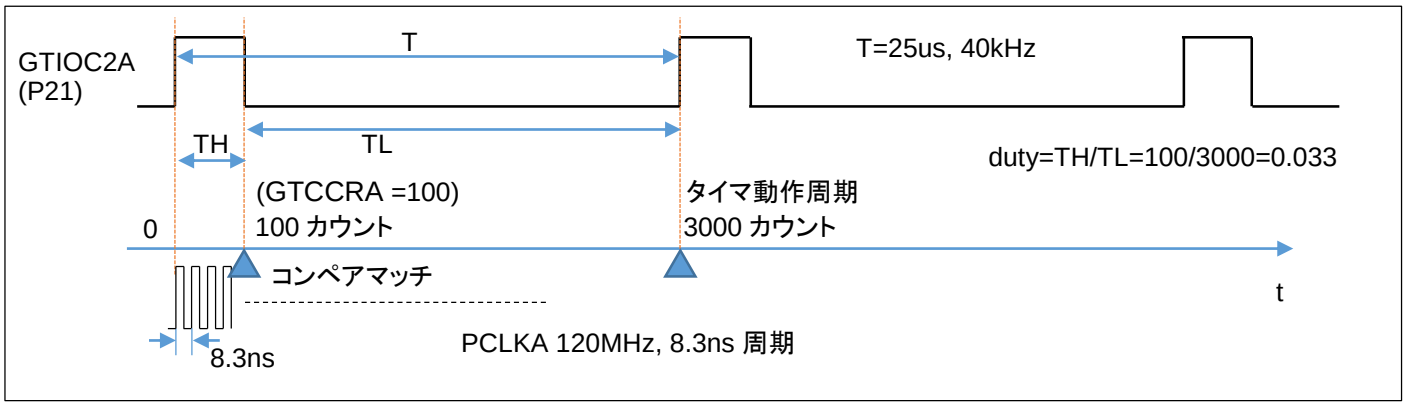
GTIOC2A 端子機能　「PWM 出力端子」

開始/停止時の出力レベル　「開始時1 出力、停止時 0 出力」

コンペアマッチ時の出力レベル　「0 出力」

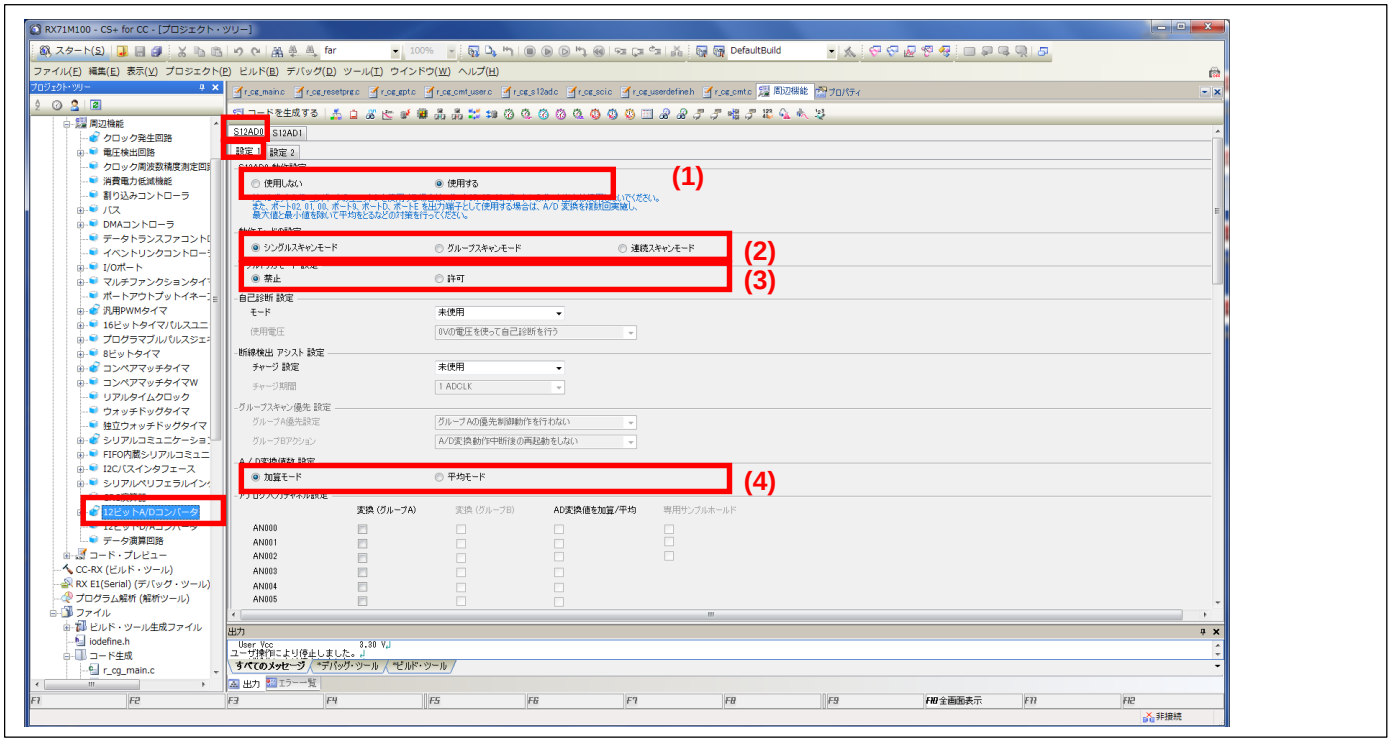
周期終わり時の出力レベル　「1 出力」

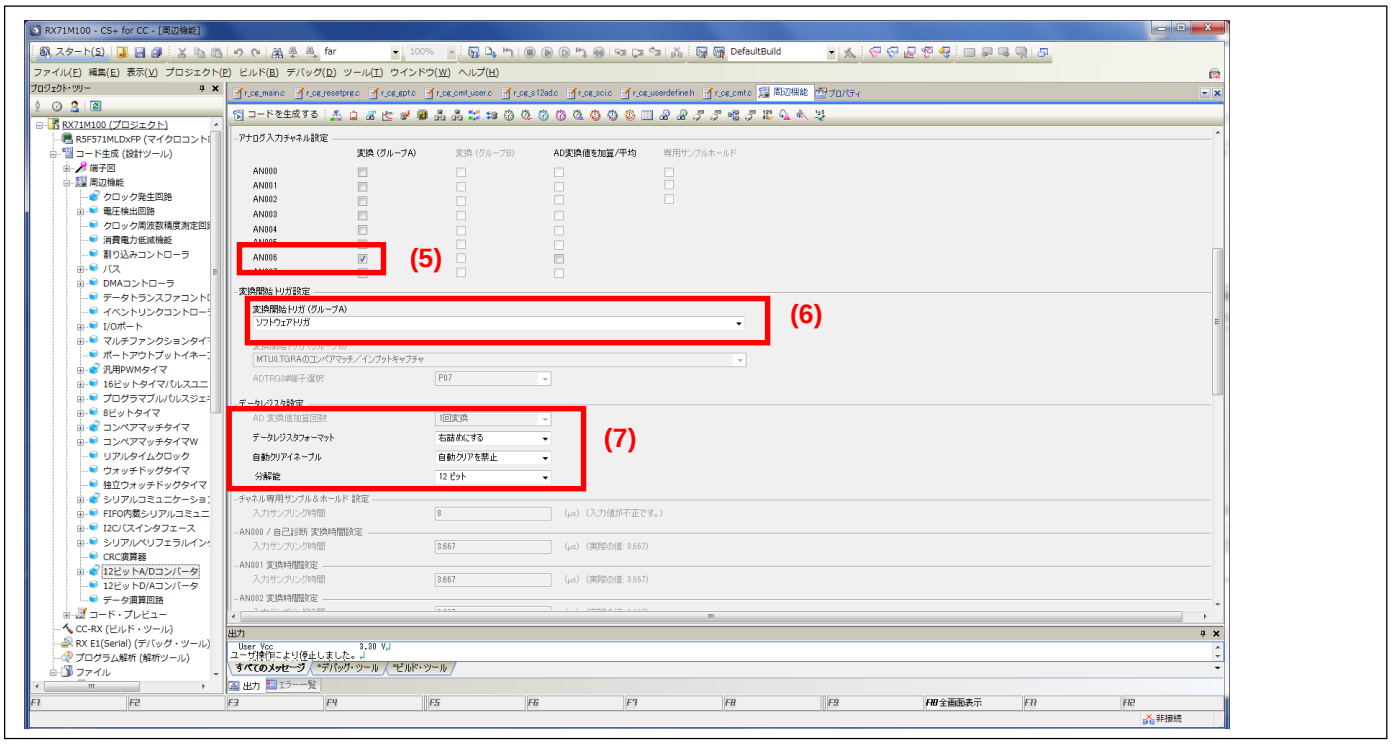
※設定項目として記載していないものはデフォルト値のままです、動作上必要な項目を抜き出して上記に記載しています

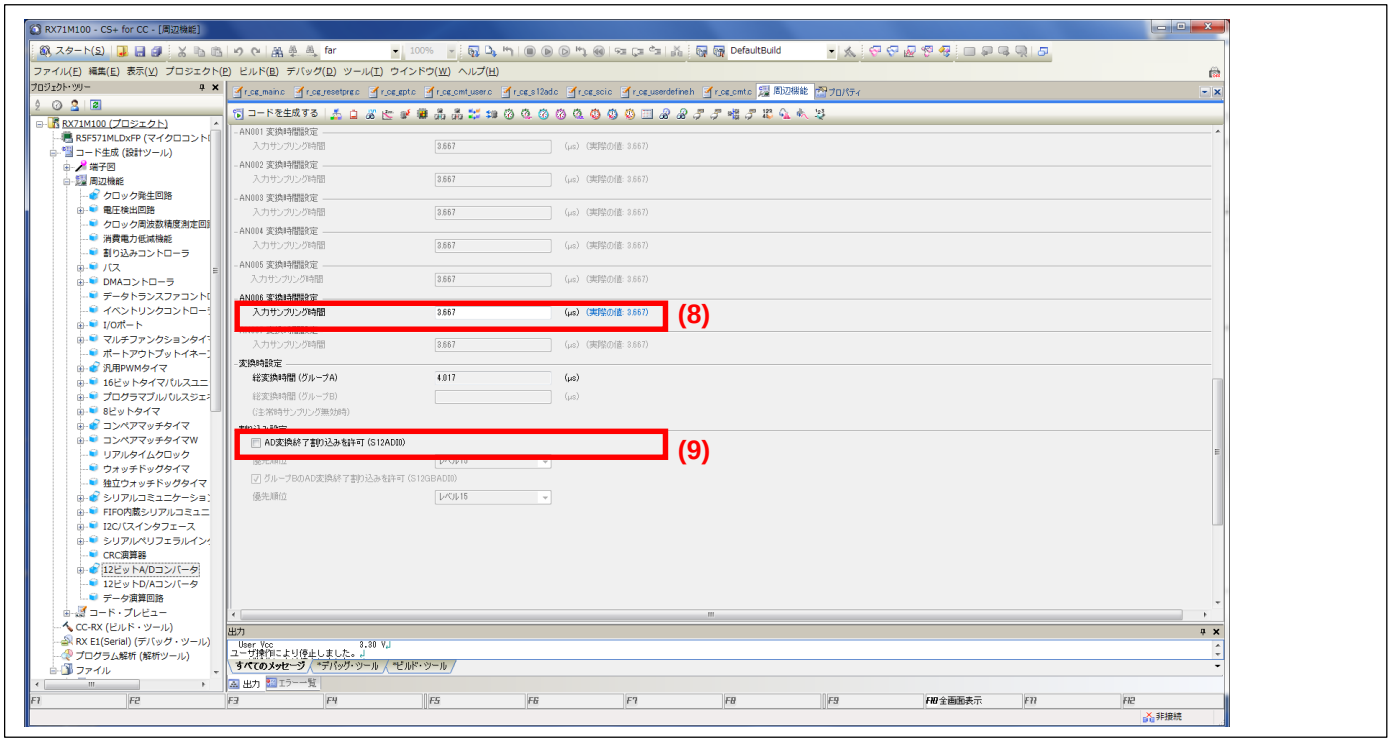


汎用 GPT タイマの機能は、設定により種々のパルス波形をつくる事ができますが、最も単純な duty を変更する PWM（Pulse Width Moduration, パルス幅変調）の波形も簡単に出力することができます。

○A/D 変換（12 ビット A/D コンバータ）







12 ビット A/D コンバータ

S12AD0 タブ

(1)S12AD0 動作設定　「使用する」

(2)動作モードの設定　「シングルスキャンモード」

(3)ダブルトリガーモード設定　「禁止」

(4)A/D 変換値数　設定　「加算モード」

(5)アナログ入力チャネル設定

AN006　「チェック」　（モータドライバボードの温度センサ）

(6)変換開始トリガ設定

変換開始トリガ（グループ A）　「ソフトウェアトリガ」

(7)データレジスタ設定

データレジスタフォーマット　「右詰めにする」

自動クリアイネーブル　「自動クリアを禁止」

分解能　「12 ビット」

(8)AN006 変換時間設定

入力サンプリング時間　「3.667」　（デフォルト値）

(9)割り込み設定

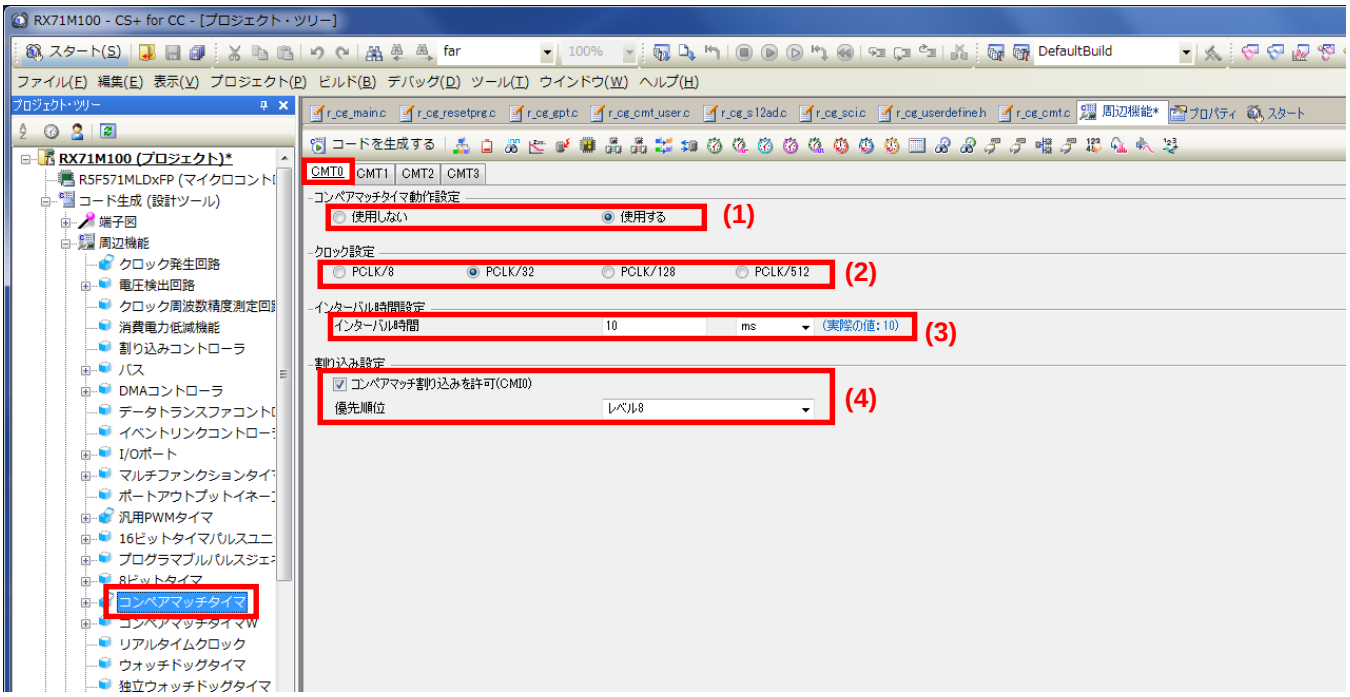
AD 変換終了割り込みを許可(S12ADI0)　「ー（チェックを外す）」(*1)

(*1)本チュートリアルでは、プログラムを単純にするため A/D 変換終了の割り込みを使用しておりません

S12AD0 側の設定は、モータドライバボードの温度センサを読み取る設定となります。

本マニュアルでは記載を省略しますが、同様の設定を、S12AD1 側でも行い、モータドライバボードの VR（ボリューム）の値を読み取れるようにしています。

○コンペアマッチタイマ



コンペアマッチタイマ

CMT0 タブ

(1)コンペアマッチタイマ動作許可　「使用する」

(2)クロック設定　「PCLK/32」

(3)インターバル時間設定

　インターバル時間　「10」「ms」

(4)割り込み設定

　コンペアマッチ割り込み許可(CMT0)　「チェック」

　優先順位　「レベル8」

上記設定後、「コード生成」すると、

BLS_MOTOR3_RX71M100¥cg_src¥r_cg_cmt_user.c が生成されます。

```
static void r_cmt_cmi0_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */

    /* End user code. Do not edit comment generated here */
}
```

コード生成ではプログラムの枠組みのみ生成されますので、実体(中身)を上記、Start - End の中に記述します。
この仕組みは、TUTORIAL1 の r_cg_main.c でも同じで、コード生成機能を使用した場合で共通です。

コンペアマッチタイマは、一定時間ごとに特定の処理を行う場合に便利な機能です。

本チュートリアルでは、CMT0 を 10ms に設定し、10ms 毎に VR の値を読み取り、duty に反映させる。10ms の割り込みルーチンが 500 回呼び出される毎に(5s に 1 回)シリアル端末に情報を表示するという動作となります。

BLS_MOTOR3_RX71M100¥cg_src¥r_cg_cmt_user.c

```
static void r_cmt_cmi0_interrupt(void){          10ms に 1 回呼び出される
    /* Start user code. Do not edit comment generated here */
    unsigned short temparature_ad_val, vr_ad_val;
    unsigned short temparature;
    unsigned char i;
    static unsigned long timer_counter=0;

    timer_counter++;

    R_S12AD0_Start();//A/D conv start
    while(S12AD.ADCSR.BIT.ADST == 1);//wait finished
    R_S12AD1_Start();//A/D conv start                    A/D 変換指示
    while(S12AD1.ADCSR.BIT.ADST == 1);//wait finished
    R_S12AD0_Get_ValueResult(6, &temparature_ad_val);    値の読み出し
    R_S12AD1_Get_ValueResult(8, &vr_ad_val);

    //tempatature calc      温度値への変換
    for(i=0; i<80; i++)
    {
        if(temparature_ad_val < g_blsm_ad_val_table[i]) break;
    }
    temparature = g_blsm_temp_table[i];
                                                         VR の A/D 変換値より duty 値を決める
    GPT2.GTCCRA = (unsigned short)((float)(vr_ad_val)/4095.0*3000.0);

    if((timer_counter % 500) == 0)          シリアル端末への表示は 5 秒に 1 回行う
    {                                       (A/D 変換及び duty の更新は 10ms 毎)
        sciPrintBuf("¥n");
        sciPrintBuf("Tempatature(A/D value) : ");
        sciPrintShortIntBuf(temparature_ad_val);
        sciPrintBuf("¥n");

        sciPrintBuf("Tempatature(degree)    : ");
        sciPrintShortIntBuf(temparature);
        sciPrintBuf("¥n");

        sciPrintBuf("VR(A/D value)          : ");
        sciPrintShortIntBuf(vr_ad_val);
        sciPrintBuf("¥n");

        sciPrintBuf("P21 duty               : ");
        sciPrintShortIntBuf(GPT2.GTCCRA);
        sciPrintBuf(" / 3000 ¥n");
    }
    /* End user code. Do not edit comment generated here */
}
```

1.5. ホールセンサの値をみる

参照プロジェクト：TUTORIAL4

プロジェクト一式をローカルのディスクにコピーし、

BLS_MOTOR4_RX71M100¥RX71M100.mtpj

を開いてください。

本チュートリアルでは、ホールセンサの値を読み取っています。
TUTORIAL3 同様、シリアル端末を接続してください。

・シリアル端末から出力される情報

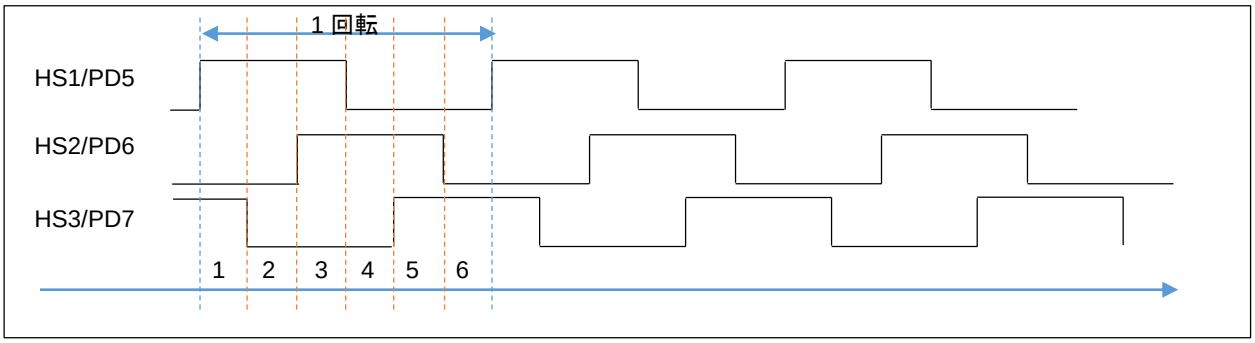
```
BLUSHLESS MOTOR STARTER KIT TUTORIAL3
pos = 01 (HS2 changed)
pos = 05 (HS3 changed)
pos = 04 (HS1 changed)
pos = 06 (HS2 changed)   モータ側から見て反時計回りに軸を回した場合
pos = 02 (HS3 changed)
pos = 03 (HS1 changed)
pos = 01 (HS2 changed)
pos = 03 (HS2 changed)
pos = 02 (HS1 changed)
pos = 06 (HS3 changed)   モータ側から見て時計回りに軸を回した場合
pos = 04 (HS2 changed)
pos = 05 (HS1 changed)
pos = 01 (HS3 changed)
```

PC 上では、teraterm 等のシリアル端末ソフトで表示してください

115,200bps, 8bit, none, 1bit の設定で表示できます

ここで、モータの軸を手でまわしてみてください。pos ＝ の数字が 01 から 06 まで変化したと思います（数値の変化の仕方は、一見順不同に見えると思います）。

この数値は、ホールセンサの位置を示しています。モータの軸は 1 回転あたり、6 回クリック（止まるところ）があると思います。モータの軸を
01 [←方向回転] → 05 [→方向回転] → 01
上記の様に左右に動かすと、数値は 01→05→01 の様に元に戻ると思います。この数値は、軸の絶対位置を表しています。



本プログラムで表示される数値は、

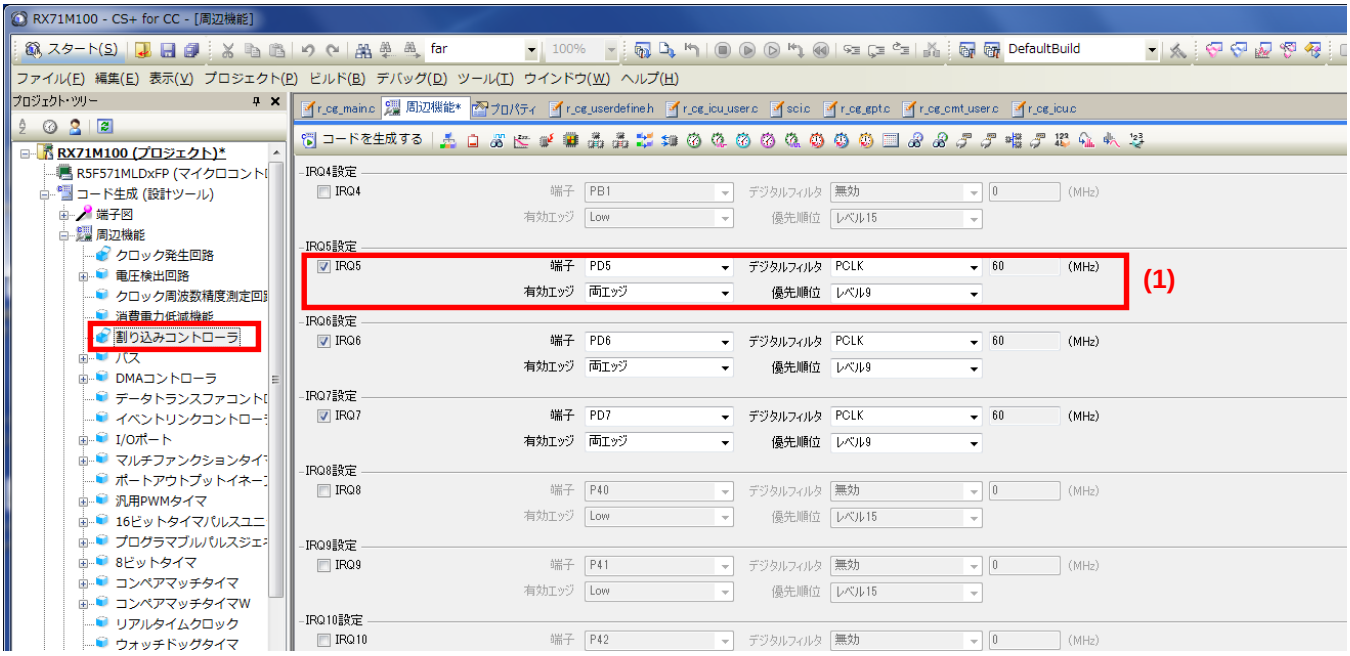
数値(pos)	上図の 1~6	PD7 (b2)	PD6 (b1)	PD5 (b0)
05	1	1	0	1
01	2	0	0	1
03	3	0	1	1
02	4	0	1	0
06	5	1	1	0
04	6	1	0	0

pos = PD7×4 + PD6×2 + PD5
となります。(プログラムの処理を容易にするため、PD7, PD6, PD5 に重み付けを行い加算した数値)
ホールセンサの出力は、接続されているマイコンのピンを入力回路に設定して、デジタル的に読み取っています。
モータの軸を手で回すと、上記表の数値に従い変化 03→02 または 03→01(回転方向による)すると思います。これは、プログラムで「いまモータの回転軸がどの位置にあるのか」を把握できていることを示します。

本プログラムでは、
PD5 → IRQ5
PD6 → IRQ6
PD7 → IRQ7
の PD5~PD7 を割り込み端子に設定し、信号が変化したタイミングで画面表示されるようになっています。

この様にすると、ホールセンサの出力(PD5~PD7)が変化したタイミングで、プログラムに特定のアクションを行わせる事ができます。

○割り込み設定



割り込みコントローラ

(1)IRQ5 設定

IRQ5　「チェック」

端子　「PD5」

デジタルフィルタ「PCLK」

有効エッジ　「両エッジ」

優先順位　「レベル 9」

※IRQ6, IRQ7 も同じ設定

BLS_MOTOR4_RX71M100¥cg_src¥r_cg_icu_user.c

```
static void r_icu_irq5_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    unsigned char pos;

    pos = hs_read();

    sciPrintBuf("pos = ");
    sciPrintByteBuf(pos);
    sciPrintBuf(" (HS1 changed)¥n");
    /* End user code. Do not edit comment generated here */
}

static void r_icu_irq6_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    unsigned char pos;

    pos = hs_read();

    sciPrintBuf("pos = ");
    sciPrintByteBuf(pos);
    sciPrintBuf(" (HS2 changed)¥n");
    /* End user code. Do not edit comment generated here */
}

static void r_icu_irq7_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    unsigned char pos;

    pos = hs_read();

    sciPrintBuf("pos = ");
    sciPrintByteBuf(pos);
    sciPrintBuf(" (HS3 changed)¥n");
    /* End user code. Do not edit comment generated here */
}
```

PD5 が変化(L→H または H→L)したときに呼び出される

ホールセンサ値の読み取り

結果の表示

PD6 が変化(L→H または H→L)したときに呼び出される

PD7 が変化(L→H または H→L)したときに呼び出される

BLS_MOTOR4_RX71M100¥cg_src¥r_cg_main.c

```
unsigned char hs_read(void)
{
    unsigned char pos;

    pos = (PORTD.PIDR.BIT.B7 << 2) + (PORTD.PIDR.BIT.B6 << 1) + PORTD.PIDR.BIT.B5;

    return pos;
}
```

1.6. モータを回してみる

参照プロジェクト：TUTORIAL5

プロジェクト一式をローカルのディスクにコピーし、

BLS_MOTOR5_RX71M100¥RX71M100.mtpj

を開いてください。

プログラム実行前に、VR を目一杯時計回り（軸を正面からみて）に回してください。
※ブルーの矢印はボード下方向から見た場合の向きです

徐々に VR を反時計回りに回していきます。

最初は変化がないと思いますが（ここで電源から流れ出る電流がモニタ出来る場合は、徐々に電流値が増しているはずです）、消費電流は増加し（大体~0.7A 程度）、いずれスムーズに回転を始めます。一度回転が始まると、消費電流はぐっと下がります（0.2A 程度）。このとき、回転数を絞っても回転が続きます。
（モータが回転せず、消費電流が~0.3A 程度ある状態のときに、軸を手で回すと回転が始まると思います）

VR を目一杯回すと、回転数は最大となり、消費電流は 2.5A 程度となります。（接続する電源の電流容量に注意願います。電源の容量が足りない場合は、電圧が下がりモータの回転数が落ちたり、電圧の下がり幅が大きいとマイコンにリセットが掛かることもあります）

・シリアル端末から出力される情報

```
BLUSHLESS MOTOR STARTER KIT TUTORIAL5
Tempatature(A/D value) : 2176
Tempatature(degree)    : 29
VR(A/D value)          : 1152
PWM duty               : 210 / 750
rotation speed         : 4936 [rpm]
```

115,200bps, 8bit, none, 1bit の設定

温度センサの A/D 変換値
温度
VR の回転角度に応じた A/D 変換値
PWM の duty 設定値（VR に連動）
回転数[rpm 値]
が 5 秒毎に表示されます。

本チュートリアルでは、マイコンのマルチファンクションタイマパルスユニット 3(MTU3)を使用しています。RX71M の MTU3 にはブラスレスモータを駆動する便利な機能があり、UVW の各層に流す電流をマイコンが切り替えてくれます。

VR は、duty を可変する目的で使用しています。VR の回転角度に応じて 0~90%程度の範囲で duty を変えています。

モータドライバボードと RX71M は、

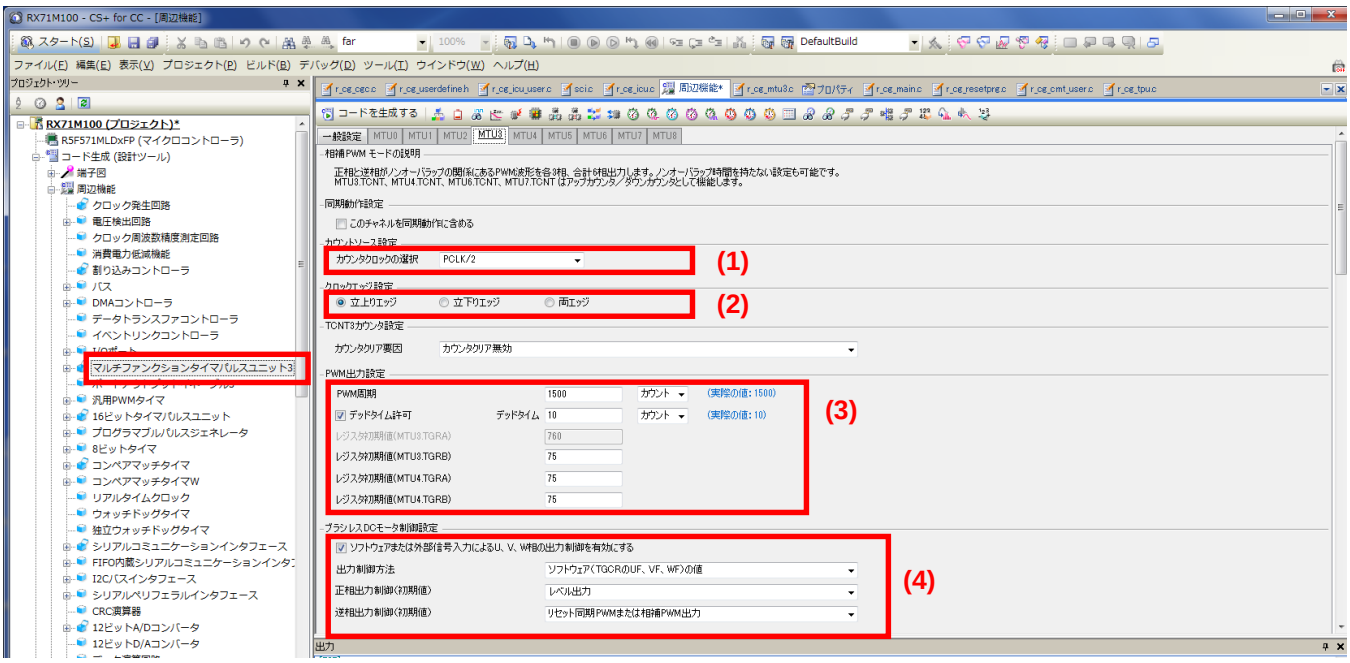
P22/MTIOC3B　Q1U　H 側 PWM 制御なし
PE2/MTIOC4A　Q2U　H 側 PWM 制御なし
PD1/MTIOC4B　Q3U　H 側 PWM 制御なし
P23/MTIOC3D　Q1L　L 側 PWM 制御
PE1/MTIOC4C　Q2L　L 側 PWM 制御
PD2/MTIOC4D　Q3L　L 側 PWM 制御

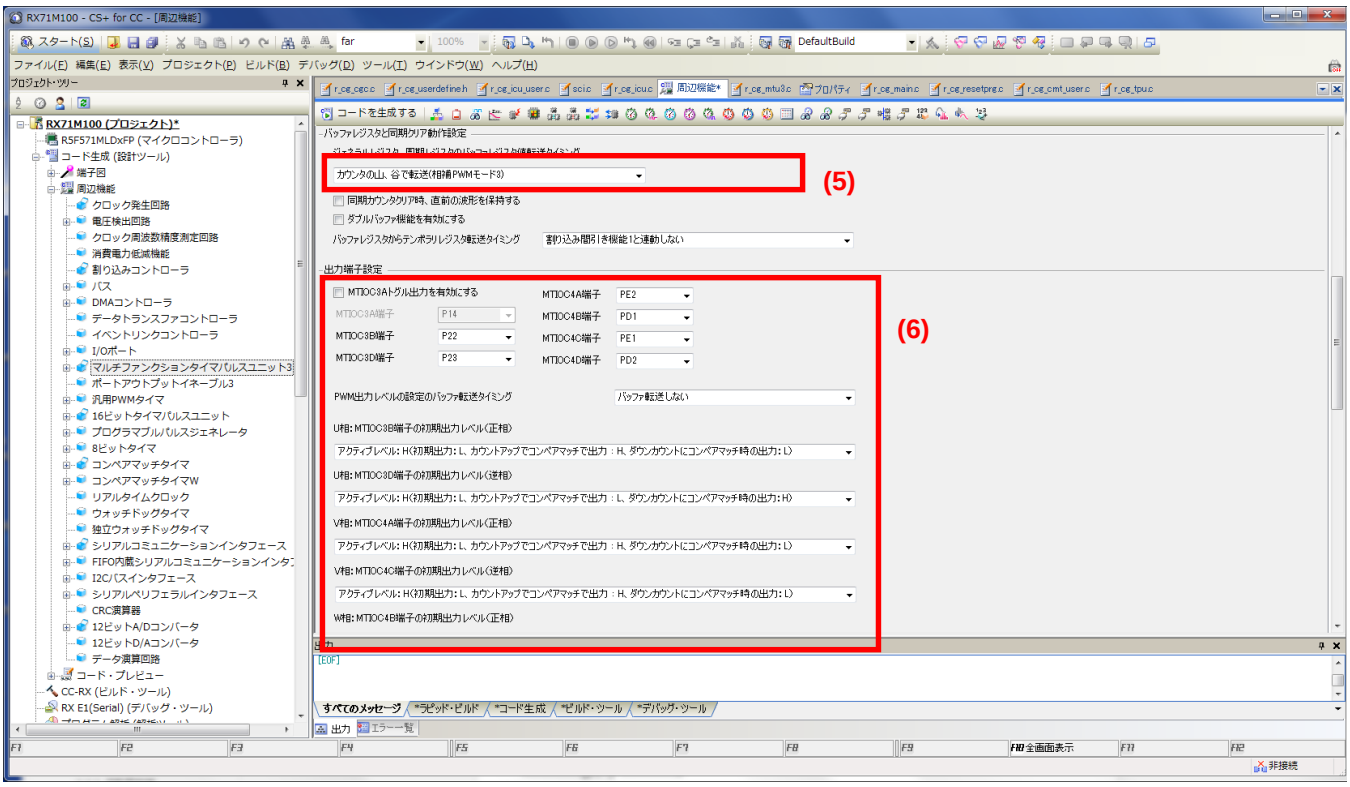
が接続される様になっています。

本チュートリアルの設定では、H 側の 3 相は PWM 制御なし（ON するタイミングで H キープ）。L 側の 3 相は、VR の回転角度に応じた PWM 制御としています。

回転数は、16 ビットタイマパルスユニットをカウンタとして使用し、1/6 回転（ホールセンサが切り替わるタイミング）に掛かる時間から計算を行っています。

○マルチファンクションタイマパルスユニット 3 設定





マルチファンクションタイマパルスユニット 3

(1)カウントソース設定

カウンタクロックの設定　「PCLK/2」

(2)クロックエッジ設定　「立上りエッジ」

(3)PWM 出力設定

PWM 周期　「1500」「カウント」

デッドタイム許可　「チェック」　デッドタイム　「10」「カウント」

レジスタ初期値(MTU3.TGRB)　「75」

レジスタ初期値(MTU4.TGRA)　「75」

レジスタ初期値(MTU4.TGRB)　「75」

(4)ブラシレス DC モータ制御設定

ソフトウェアまたは外部信号入力による U、v、w 相の出力制御を有効にする　「チェック」

出力制御方法　「ソフトウェア」

正相出力制御（初期値）　「レベル出力」

正相出力制御（初期値）　「リセット同期 PWM または相補 PWM 出力」

(5)バッファレジスタと同期クリア動作設定

ジェネラルレジスタ、周期レジスタのバッファレジスタ値転送タイミング　「カウンタの山、谷で転送（相補 PWM モード 3）」

(6)出力端子設定

MTIOC3A トグル出力を有効にする　「ー（チェックしない）」

MTIOC3B 端子　「P22」

MTIOC3D 端子　「P23」

MTIOC4A 端子　「PE2」

MTIOC4B 端子　「PD1」

MTIOC4C 端子　「PE1」

MTIOC4D 端子　「PD2」

U 相 MTIOC3B 端子の初期出力レベル（正相）　「アクティブレベル H」

U 相 MTIOC3D 端子の初期出力レベル（逆相）　「アクティブレベル H」

V 相 MTIOC4A 端子の初期出力レベル（正相）　「アクティブレベル H」

V 相 MTIOC4C 端子の初期出力レベル（逆相）　「アクティブレベル H」

W 相 MTIOC4B 端子の初期出力レベル（正相）　「アクティブレベル H」

W 相 MTIOC4D 端子の初期出力レベル（逆相）　「アクティブレベル H」

BLS_MOTOR4_RX71M100¥cg_src¥r_cg_icu_user.c

```
static void r_icu_irq5_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    if(PORTD.PIDR.BIT.B5 == 0) MTU.TGCRA.BIT.WF = 0;
    else MTU.TGCRA.BIT.WF = 1;

    rpm_calc();
    /* End user code. Do not edit comment generated here */
}

static void r_icu_irq6_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    if(PORTD.PIDR.BIT.B6 == 0) MTU.TGCRA.BIT.UF = 0;
    else MTU.TGCRA.BIT.UF = 1;

    rpm_calc();
    /* End user code. Do not edit comment generated here */
}

static void r_icu_irq7_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    if(PORTD.PIDR.BIT.B7 == 0) MTU.TGCRA.BIT.VF = 0;
    else MTU.TGCRA.BIT.VF = 1;

    rpm_calc();
    /* End user code. Do not edit comment generated here */
}
```

ホールセンサの端子割り込み
ホールセンサの出力が変わった時
WF レジスタの値を変更

回転数の計算

モータ制御の核心部は、上記のコードで完結しています。ホールセンサの値と、UF, VF, WF レジスタの値を同期させるだけです。実際の波形生成・切り替えは、マイコンが行います。

なお、このコードはモータを時計回り(モータ側から見て)させます。

※モータを逆回転させるときは、ホールセンサとレジスタの対応をずらす事により対応可能です

後述のサンプルプログラムでは、時計回り・反時計回りの両方に対応しています

・回転数の計算

BLS_MOTOR4_RX71M100¥cg_src¥r_cg_main.c

```
void rpm_calc(void)
{
    unsigned long count;
    float rpm_tmp;

    count = ((unsigned long)TPU1.TCNT << 16) + TPU2.TCNT;

    rpm_tmp = 600.0e6 / (float)count;//rpm = 1/(T*16.66n)*60/6 = 600e6/T

    rpm_val[rpm_val_index++] = (unsigned short)rpm_tmp;

    if(rpm_val_index == RPM_HIST_SIZE) rpm_val_index=0;

    TPU2.TCNT=0;
    TPU1.TCNT=0;

}
```

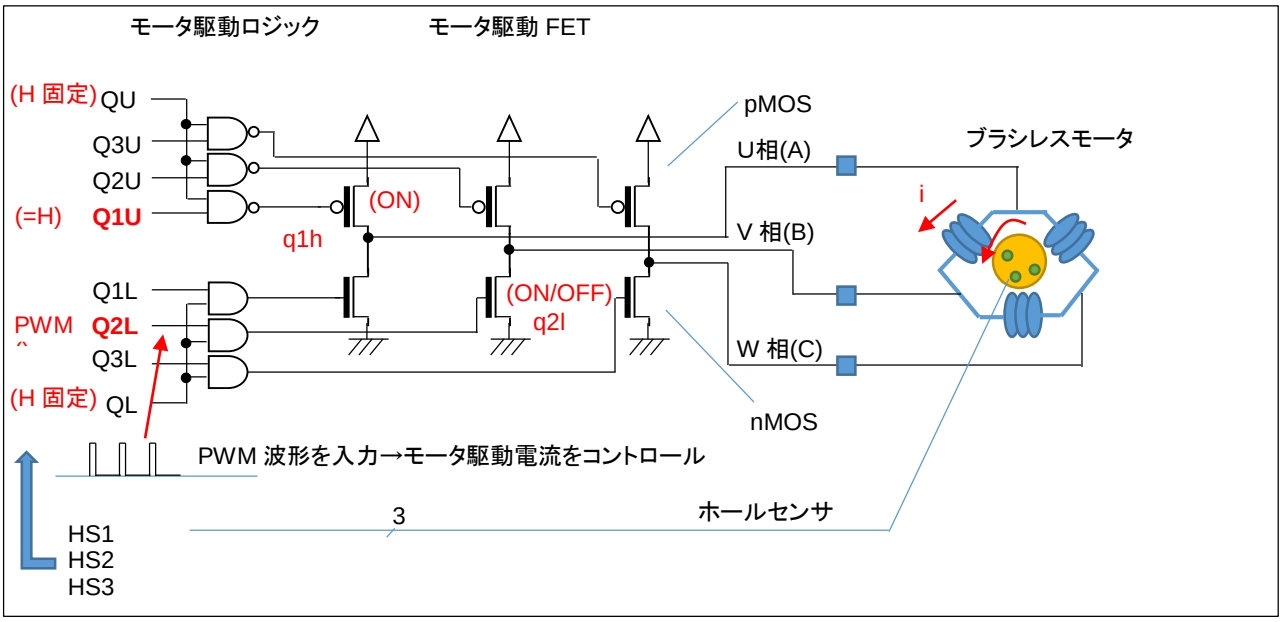
16 ビットカウンタをカスケード接続し 32 ビットカウンタとして使用

1 カウント 16.66ns, 1/6 回転で何カウントとなるか

回転数は平均化するため配列変数に格納

カウンタをクリアする

・マルチファンクションタイマパルスユニット 3(MTU3)を使用した駆動



本チュートリアルでは、
QU=QL=H 固定（QU, QL の機構は未使用で、動作する方向に固定）
としています。
※QU, QL は 1 相の PWM 波形で UVW の L 側もしくは H 側 3 相をパルス駆動する機構ですが、MTU3 は 6 相 PWM 波形を出力できる能力を備えていますので、本チュートリアルでは未使用としています

Q1U~Q3U は、ホールセンサの値により、1 つだけ ON(H)となります。
Q1L~Q3L は、ホールセンサの値により、1 つだけ PWM 制御されます。

PWM 周期は、40kHz(25us)固定で、duty は VR により可変です。

10ms 毎のコンペアマッチタイマ割り込みを使用して VR→duty の反映を行っています。

duty を変えると、モータに入力される電力が変わり、回転数に反映されます。本チュートリアルでは、回転数の検出も行っていますので、duty と回転数の関係を確認する事ができます。

1.7. 過電流・過熱・短絡保護の動作

参照プロジェクト：TUTORIAL6

プロジェクト一式をローカルのディスクにコピーし、

BLS_MOTOR6_RX71M100¥RX71M100.mtpj

を開いてください。

本チュートリアルのプログラムは、「過電流」「過熱」「短絡」のいずれかを検出した場合モータを停止する様になっています。

基本的な動作は、TUTORIAL5 と同じです。VR を絞った状態でプログラムを実行し、徐々に VR を回して回転数を上げてみてください。モータドライバボード上の D3(CURRENT MONITOR)が、非点灯から暗く点灯。そして徐々に明るくなっていく様子が観測できるかと思います。

いずれ、モータが停止し、端末には以下のメッセージが表示されます。

```
Tempatature(A/D value) : 2339
Tempatature(degree)      : 34
VR(A/D value)            : 2418
PWM duty                 : 442 / 750
rotation speed           : 10966 [rpm]

OVER CURRENT detected.
***MOTOR STOP***
```

過電流検出停止の場合

リセット後、過電流で停止しない程度で長時間回転を続けると以下のメッセージが表示されます。

```
Tempatature(A/D value) : 2533
Tempatature(degree)      : 38
VR(A/D value)            : 1651
PWM duty                 : 302 / 750
rotation speed           : 7507 [rpm]

OVER HEAT detected.
***MOTOR STOP***
```

過熱停止の場合

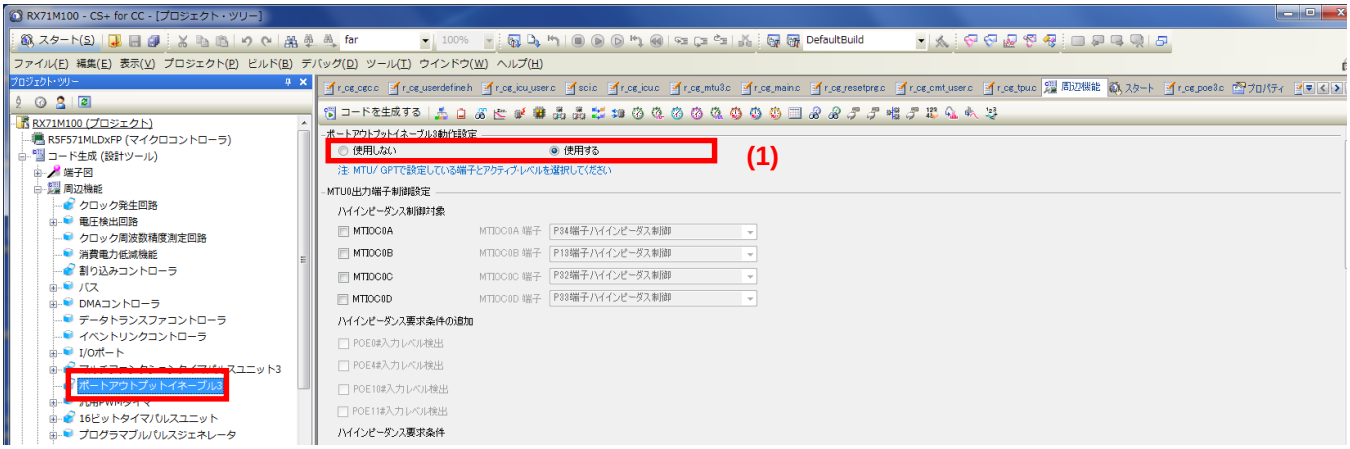
再びリセット後、モータが回転している状態で、マイコンボードの SW2 を押してください。

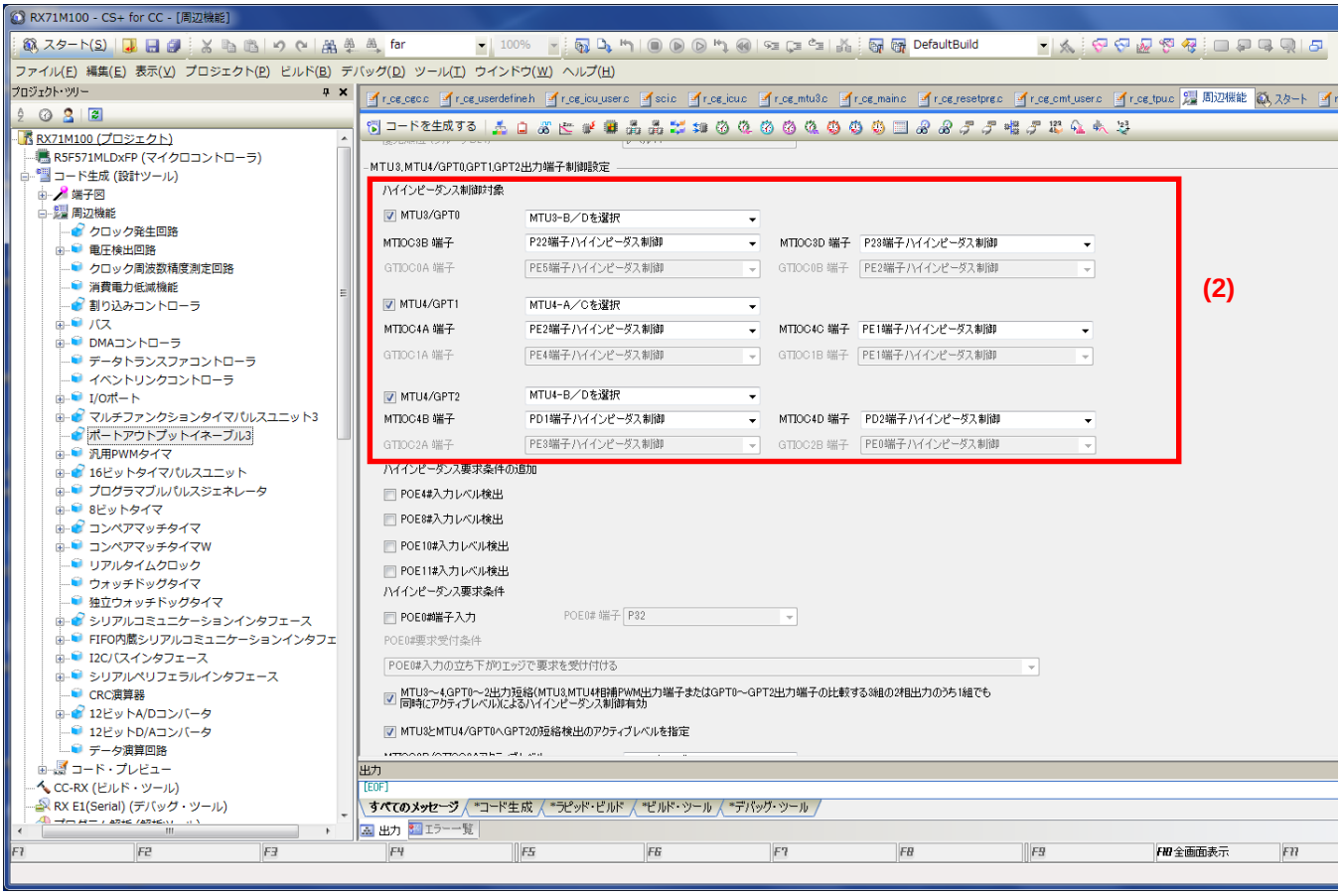
```
Tempatature(A/D value) : 2216
Tempatature(degree)    : 30
VR(A/D value)          : 856
PWM duty               : 156 / 750
rotation speed         : 3520 [rpm]

POE3 detected.
***MOTOR STOP***
```

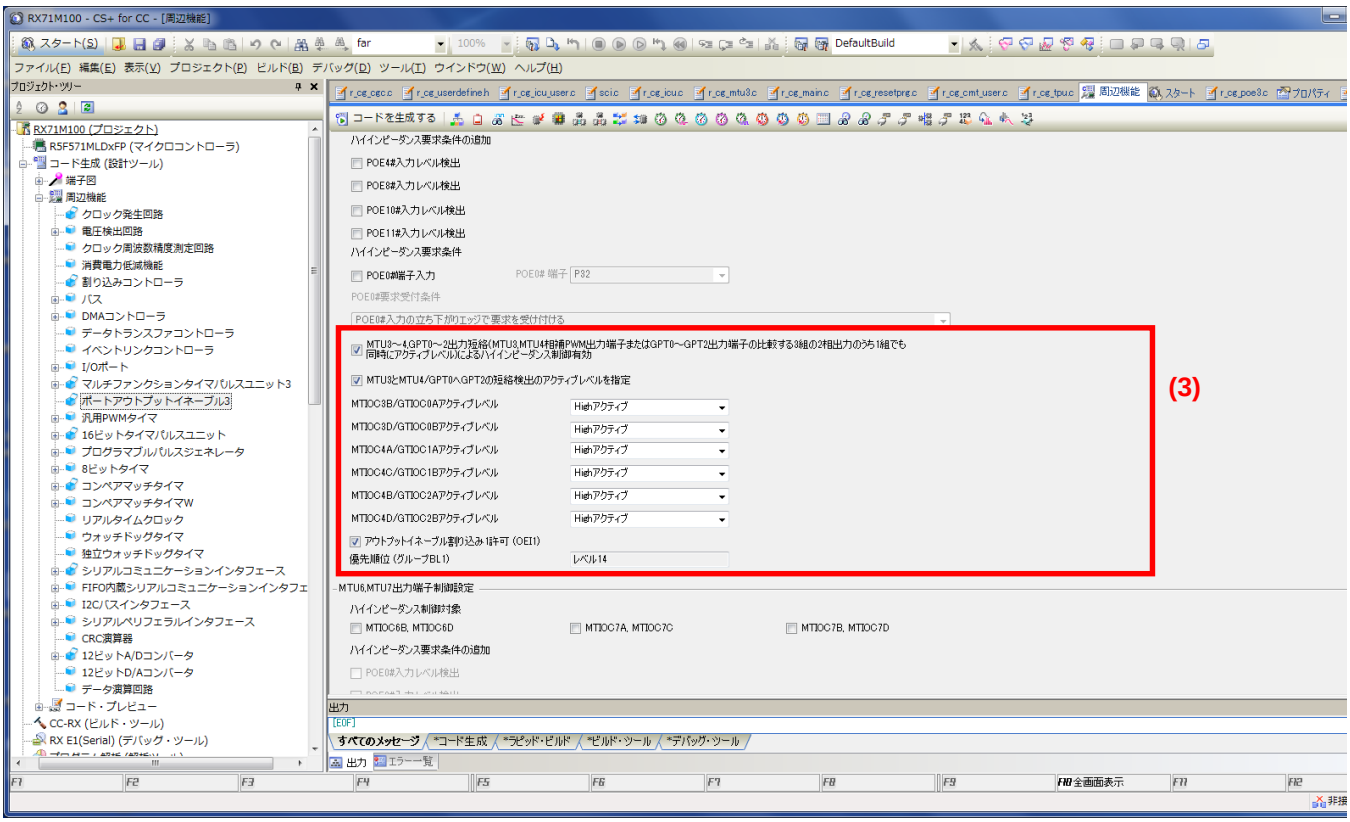
短絡停止の場合

SW2 を押すと、プログラムで
Q1U, Q2U, Q3U を全て H 出力設定
となります。Q1L, Q2L, Q3L のいずれかが H 出力となったタイミングで、短絡検出され、モータは停止します。
※本チュートリアルでは、モータ駆動端子を強引に汎用ポート設定し H 出力としていますが、プログラムのミスや PWM 波形のレジスタ設定の設定値の関係で、L 側と H 側の端子ペアが両方 H（アクティブ）となったときに、POE3 は発動します。

○ポートアウトプットイネーブル 3(POE3)設定




RX71M100 - CS+ for CC - [周辺機能]
MTU3,MTU4/GPT0,GPT1,GPT2出力端子制御設定
ハイインピーダンス制御対象
MTU3/GPT0
MTU3-B／Dを選択
MTIOC3B 端子
P22端子ハイインピーダス制御
MTIOC3D 端子
P23端子ハイインピーダス制御
GTIOC0A 端子
PE5端子ハイインピーダス制御
GTIOC0B 端子
PE2端子ハイインピーダス制御
MTU4/GPT1
MTU4-A／Cを選択
MTIOC4A 端子
PE2端子ハイインピーダス制御
MTIOC4C 端子
PE1端子ハイインピーダス制御
GTIOC1A 端子
PE4端子ハイインピーダス制御
GTIOC1B 端子
PE1端子ハイインピーダス制御
MTU4/GPT2
MTU4-B／Dを選択
MTIOC4B 端子
PD1端子ハイインピーダス制御
MTIOC4D 端子
PD2端子ハイインピーダス制御
GTIOC2A 端子
PE3端子ハイインピーダス制御
GTIOC2B 端子
PE0端子ハイインピーダス制御
ハイインピーダンス要求条件の追加
POE4#入力レベル検出
POE8#入力レベル検出
POE10#入力レベル検出
POE11#入力レベル検出
ハイインピーダンス要求条件
POE0#端子入力
POE0# 端子 P32
POE0#要求受付条件
POE0#入力の立ち下がりエッジで要求を受け付ける
MTU3～4,GPT0～2出力短絡(MTU3,MTU4相補PWM出力端子またはGPT0～GPT2出力端子の比較する3組の2相出力のうち1組でも同時にアクティブレベルによるハイインピーダンス制御有効
MTU3とMTU4/GPT0へGPT2の短絡検出のアクティブレベルを指定
(2)


RX71M100 - CS+ for CC - [周辺機能]
ハイインピーダンス要求条件の追加
POE4#入力レベル検出
POE8#入力レベル検出
POE10#入力レベル検出
POE11#入力レベル検出
ハイインピーダンス要求条件
POE0#端子入力
POE0# 端子 P32
POE0#要求受付条件
POE0#入力の立ち下がりエッジで要求を受け付ける
MTU3～4,GPT0～2出力短絡(MTU3,MTU4相補PWM出力端子またはGPT0～GPT2出力端子の比較する3組の2相出力のうち1組でも同時にアクティブレベルによるハイインピーダンス制御有効
MTU3とMTU4/GPT0へGPT2の短絡検出のアクティブレベルを指定
MTIOC3B/GTIOC0Aアクティブレベル
High アクティブ
MTIOC3D/GTIOC0Bアクティブレベル
High アクティブ
MTIOC4A/GTIOC1Aアクティブレベル
High アクティブ
MTIOC4C/GTIOC1Bアクティブレベル
High アクティブ
MTIOC4B/GTIOC2Aアクティブレベル
High アクティブ
MTIOC4D/GTIOC2Bアクティブレベル
High アクティブ
アウトプットイネーブル割り込み1許可 (OEI1)
優先順位 (グループBL1)
レベル14
MTU6,MTU7出力端子制御設定
ハイインピーダンス制御対象
MTIOC6B, MTIOC6D
MTIOC7A, MTIOC7C
MTIOC7B, MTIOC7D
ハイインピーダンス要求条件の追加
POE0#入力レベル検出
(3)

ポートアウトプットイネーブル 3

(1)ポートアウトプットイネーブル 3 動作設定　「使用する」

(2)MTU3, MTU4/GPT0, GPT1, GPT2 出力端子制御設定

ハイインピーダンス制御対象

MTU3/GPT0　「チェック」　「MTU3-B/D を選択」

MTIOC3B 端子　「P22 端子ハイインピーダンス制御」

MTIOC3D 端子　「P23 端子ハイインピーダンス制御」

MTU4/GPT1　「チェック」　「MTU4-A/C を選択」

MTIOC4A 端子　「PE2 端子ハイインピーダンス制御」

MTIOC4C 端子　「PE1 端子ハイインピーダンス制御」

MTU4/GPT2　「チェック」　「MTU4-B/D を選択」

MTIOC4B 端子　「PD1 端子ハイインピーダンス制御」

MTIOC4D 端子　「PD2 端子ハイインピーダンス制御」

(3)

MTU3~4 出力短絡ハイインピーダンス制御有効　「チェック」

MTIOC3B アクティブレベル　「High アクティブ」

MTIOC3D アクティブレベル　「High アクティブ」

MTIOC4A アクティブレベル　「High アクティブ」

MTIOC4C アクティブレベル　「High アクティブ」

MTIOC4B アクティブレベル　「High アクティブ」

MTIOC4D アクティブレベル　「High アクティブ」

アウトプットイネーブル割り込み許可(OEI1)　「チェック」　「レベル 14」

・過熱・過電流保護設定部

BLS_MOTOR6_RX71M100¥cg_src¥r_cg_cmt_user.c

```
//過熱検出                    10ms 割り込み毎に実行される
if(temparature >= BLSM_TEMPARATURE_THRESHOLD)
{
    PORTD.PODR.BIT.B3 = 0;//QU
    PORT2.PODR.BIT.B1 = 0;//QL
    POE3.SPOER.BIT.MTUCH34HIZ = 1;

    R_CMT0_Stop();//CMT0停止

    sciPrint("¥nOVER HEAT detected.¥n***MOTOR STOP***¥n");
}

//過電流検出                  10ms 割り込みの 100 回に 1 回(1 秒に 1 回)実行される
if((g_blsm_over_current_check_start == 1) && (g_blsm_over_current_count >=
BLSM_OVER_CURRENT_COUNT_THRESHOLD))
{
    PORTD.PODR.BIT.B3 = 0;//QU
    PORT2.PODR.BIT.B1 = 0;//QL
    POE3.SPOER.BIT.MTUCH34HIZ = 1;

    R_CMT0_Stop();//CMT0停止

    sciPrint("¥nOVER CURRENT detected.¥n***MOTOR STOP***¥n");

}
```

```
#define BLSM_UPPER_TEMP 40
#define BLSM_OVER_CURRENT_COUNT_THRESHOLD 1000
```

モータに流れる電流を止める処理

モータ動作で使用しているタイマを止める処理

監視している変数は 100us 毎に呼び出される CMT1 でセットしている

モータに流れる電流を止める処理

モータ動作で使用しているタイマを止める処理

・短絡保護設定部

BLS_MOTOR6_RX71M100¥cg_src¥r_cg_poe3_user.c

```
void r_poe3_oei1_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    PORTD.PODR.BIT.B3 = 0;//QU
    PORT2.PODR.BIT.B1 = 0;//QL

    ICU.GENBL1.BIT.EN9 = 0;//OEN1割り込みディスエーブル

    R_CMT0_Stop();//CMT0停止

    sciPrint("¥nPOE3 detected.¥n***MOTOR STOP***¥n");
    /* End user code. Do not edit comment generated here */
}
```

短絡検出時実行される割り込みルーチン

QU, QL は手動で停止させる

1.8. 相電圧・相電流の観測

参照プロジェクト：TUTORIAL7

プロジェクト一式をローカルのディスクにコピーし、

BLS_MOTOR7_RX71M100¥RX71M100.mtpj

を開いてください。

本プログラムでは、A/D 変換の機能を使い、U, V, W の各相電圧および U, V, W の L 側の電流観測用抵抗に流れている電流を取得します。プログラムの動作としては、基本的には TUTORIAL6 と同じです。
但し、TUTORIAL6 と異なる点があり、（モータが回っているときに）SW2 を押すと、押した瞬間から 400 ポイント（50us×400=20ms 間）分の相電圧、相電流のデータを取得して表示するという動作となります。

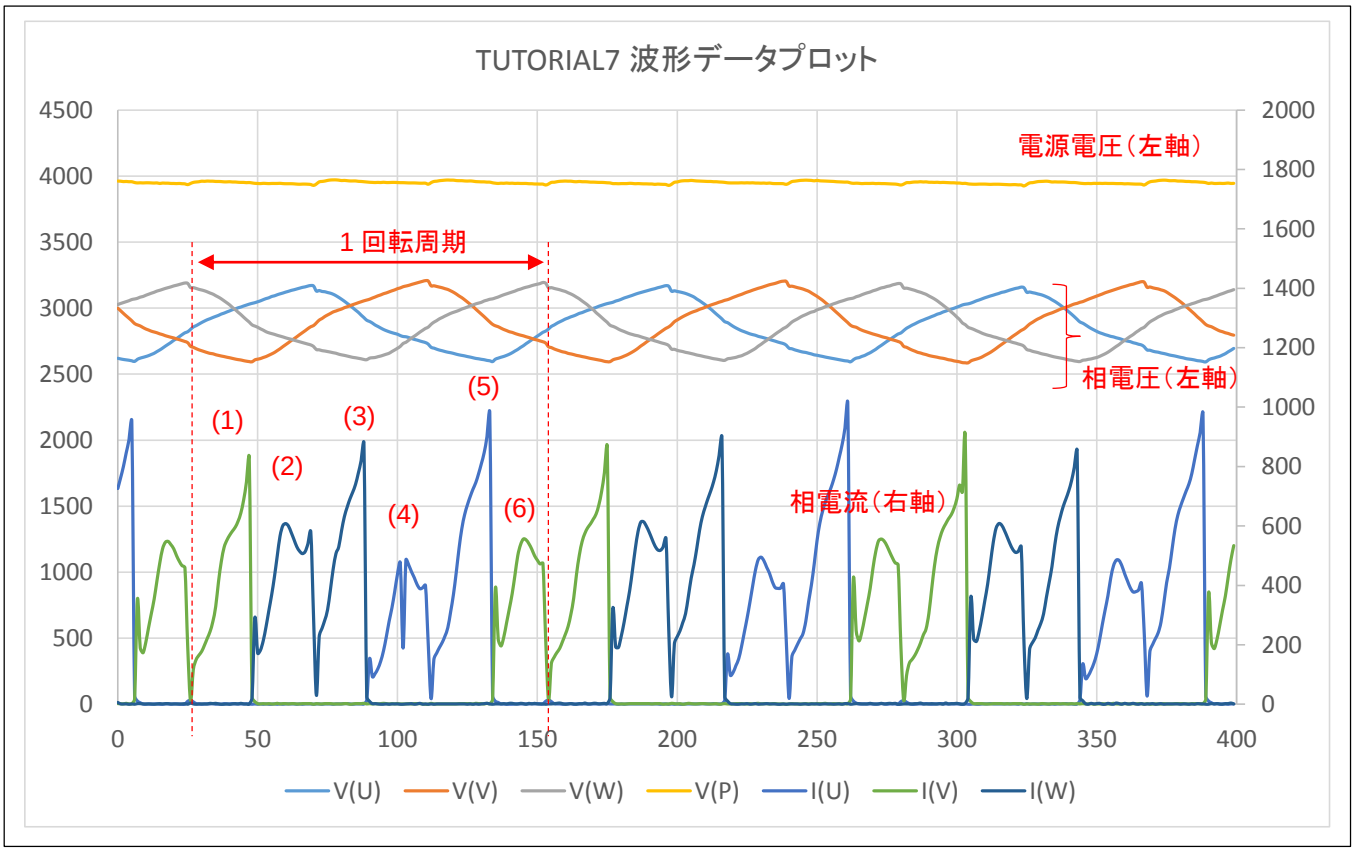
・シリアル端末から出力される情報

```
BLUSHLESS MOTOR STARTER KIT TUTORIAL7

Tempatature(A/D value) : 2148
Tempatature(degree)    : 28
VR(A/D value)          : 2116
PWM duty               : 387 / 750
rotation speed         : 9469 [rpm]
                                          ここで SW2 を押す

index,V(U),V(V),V(W),V(P),I(U),I(V),I(W)
0,2307,2364,2529,3451,0,694,0
1,2313,2360,2533,3449,1,695,0
2,2323,2357,2537,3449,0,679,0
3,2330,2353,2541,3448,0,656,0
4,2338,2350,2546,3448,1,626,0
…
```

duty，温度，回転数を表示します
（TUTORIAL6 と同じ動作）

一瞬（20ms のデータ取得）後
400 ポイントのデータが画面に
表示されます
（データは、CSV 形式となっているので
Excel 等に貼り付けてください）

・シリアル端末から出力される情報を Excel でプロットした波形



※縦軸は、A/D 変換値(0~4095)

BLS_MOTOR7_RX24T100¥cg_src¥r_cg_cmt_user.c

回転制御のプログラム

```
switch(pos)
{
    case 1:
        PORT7.PODR.BYTE = W_V_DIRECTION;   (6)
    break;
    case 5:
        PORT7.PODR.BYTE = W_U_DIRECTION;   (5)
    break;
    case 4:
        PORT7.PODR.BYTE = V_U_DIRECTION;   (4)
    break;
    case 6:
        PORT7.PODR.BYTE = V_W_DIRECTION;   (3)
    break;
    case 2:
        PORT7.PODR.BYTE = U_W_DIRECTION;   (2)
    break;
    case 3:
        PORT7.PODR.BYTE = U_V_DIRECTION;   (1)
    break;
    default:
    break;
}
```

W→V に電流を流す：上のグラフの(6)に対応

時系列

波形は、端末に表示された数値をプロットしたものである。数値は 12bit A/D 変換の生データ（0-4095）となっています。相電圧（モータドライバボード上でフィルタリングされている）は、sin 波に近い波形。相電流は 1/3 周期に 2 回山が出来る（U の L 側に着目すると、V→U, W→U と、続けて電流を流す処理を行うので）形となります。

※制御プログラム次第で、波形は変わります

本プログラムでは、A/D 変換の生データを一旦メモリに格納し、それをシリアル端末経由で出力する処理を行っていますが、実際のモータ制御プログラムでは、得られたデータをリアルタイムに処理して、モータの制御に活用する事が出来ます。

モータ制御のチュートリアル編はここまでとなります。モータドライバボードに備わっている機能は一通り試しましたので、この後はマイコンの機能を活用し、様々なアルゴリズムでブラシレスモータを制御してみてください。

なお、RX71M のチュートリアルでは、可能な限り CS+の「コード生成」の機能を使用しています。「コード生成」機能は、プログラムの枠組みやレジスタ初期設定を自動で生成してくれますので、活用すれば便利な機能だと思います。もちろん、ある程度 RX のプログラムに慣れてきたら、「コード生成」機能を使わずにレジスタ設定等を書き下す事もできますので、必要に応じて（初めて使用する機能はとりあえず「コード生成」に任せる等）使用してみて下さい。

2. サンプルプログラム

2.1. プログラム仕様

サンプルプログラムは、チュートリアルの内容を踏まえ、回転数や回転方向を制御できるものとなっています。

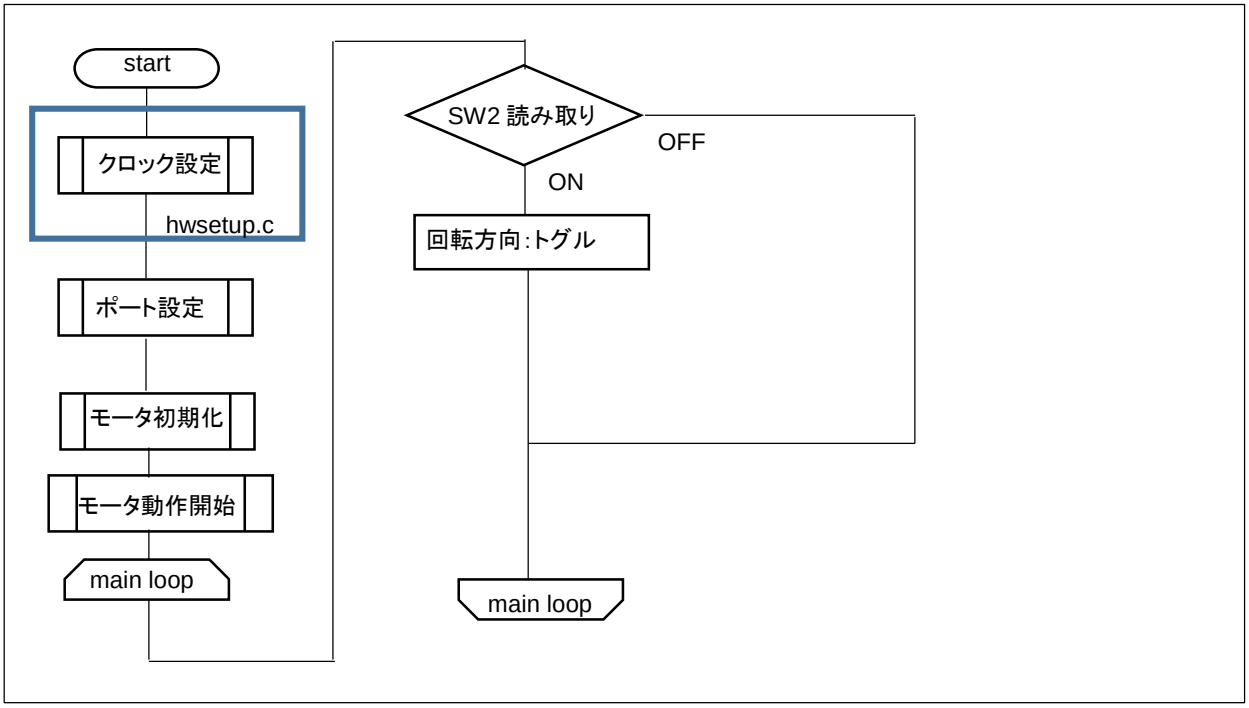
本プログラム(BLSM_SAMPLE1_RX71M100)は、以下の仕様を実装しています。

・ホールセンサを使用
・VR にて回転数を変更
・SW2 で回転（正転・逆転）・停止を制御
・過電流保護停止機能
・過熱停止機能

SW2　反時計回り→時計回り→停止をトグルで切り替え
モータドライバボード上 VR　回転数制御

2.2. 関数仕様

2.2.1. 全体及び main 関数



SW2 を押す度に

・反時計回り

・時計回り

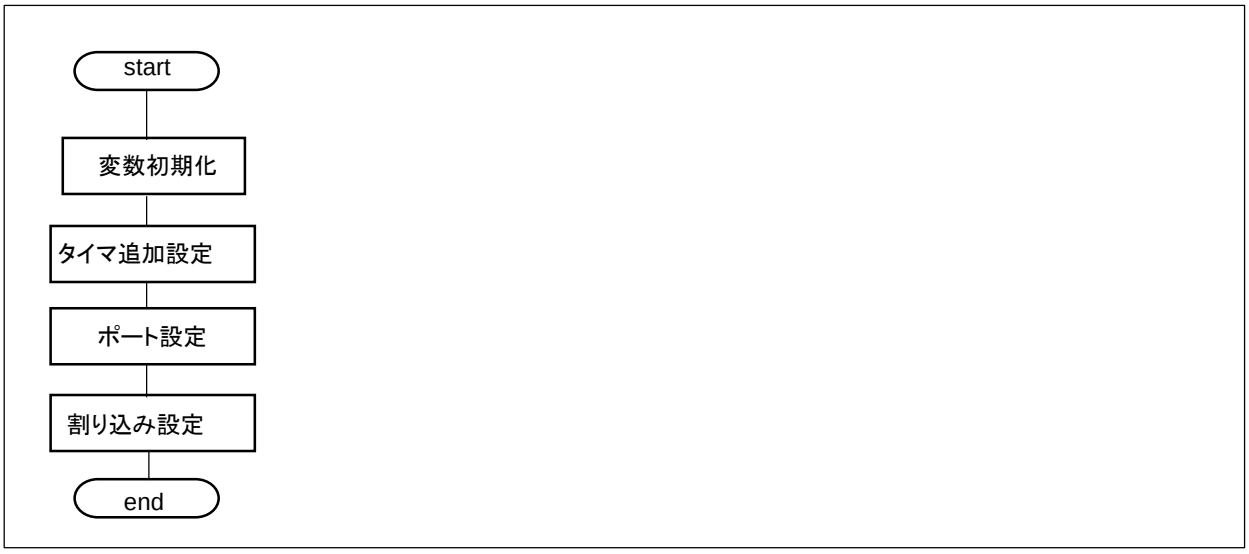
・停止

がトグルで切り替わります。

2.2.2. モータ制御関数

blsm_motor_init

概要:初期化関数
宣言:void blsm_motor_init(void)
説明:
・変数の定義・初期化
・端子の設定
・タイマーの設定
を行います
引数:なし
戻り値:なし



blsm_motor_start

概要:モータ制御開始関数
宣言:void blsm_motor_start (void)
説明:
・変数の初期化
・タイマーのカウント開始
を行います
引数:なし
戻り値:なし

blsm_motor_stop

概要:モータ制御停止関数
宣言:void blsm_motor_stop (void)
説明:
・タイマー停止
・変数の初期化
を行います
引数:なし
戻り値:なし

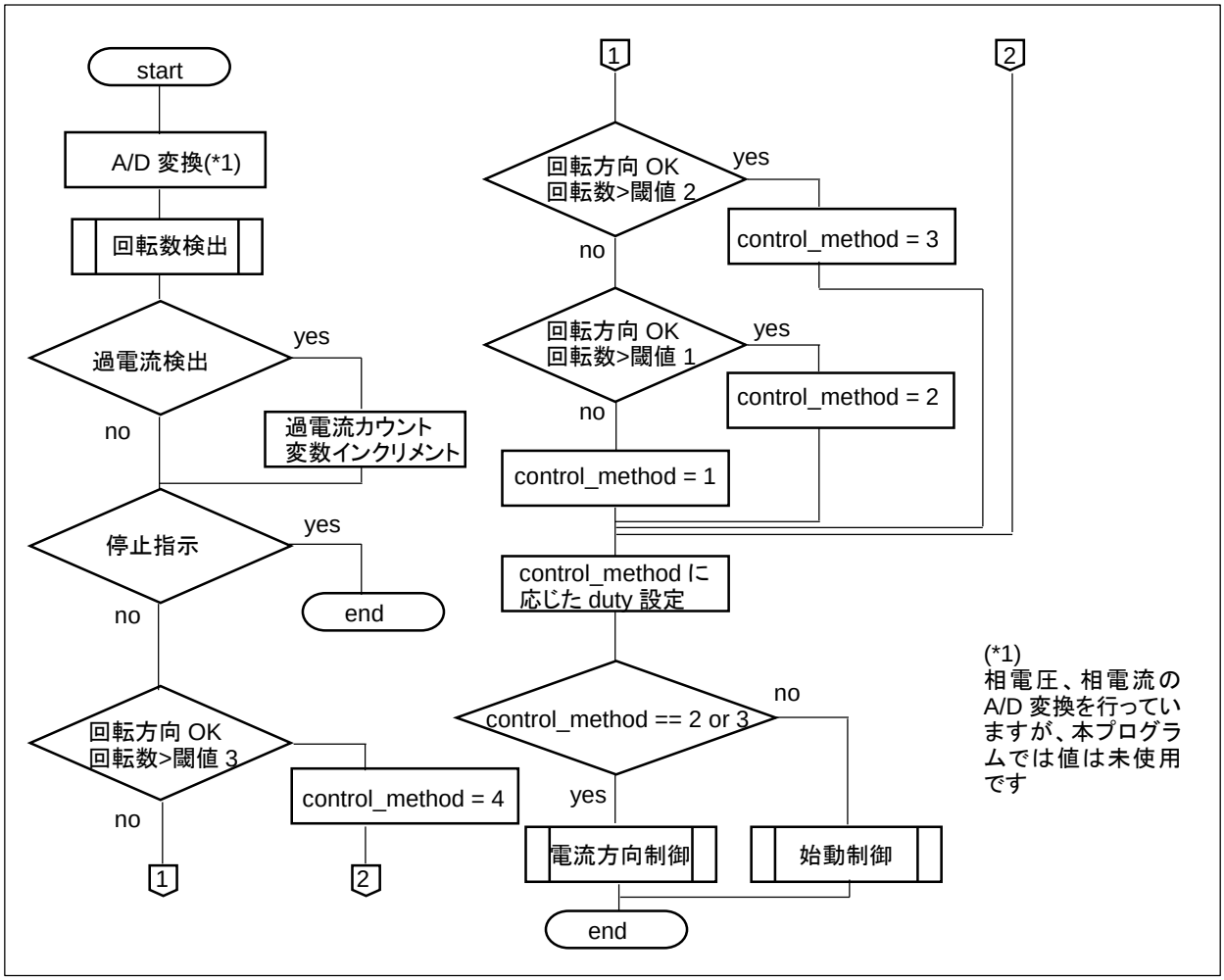
2.2.3. 割り込み関数

モータ制御プログラム本体の処理は、割り込みを使用して行います。

r_cmt_cmi1_interrupt

概要：タイマー割り込み（50us 周期）
宣言：static void r_cmt_cmi1_interrupt(void)
説明：
・回転数・回転方向検出
・始動制御（始動時のみ）
を行います
引数：なし
戻り値：なし

－フローチャート－



r_cmt_cmi0_interrupt

概要：タイマー割り込み（10ms 周期）

宣言：static void r_cmt_cmi0_interrupt(void)

説明：

・過電流検出停止処理

・過熱停止処理・始動制御（始動時のみ）

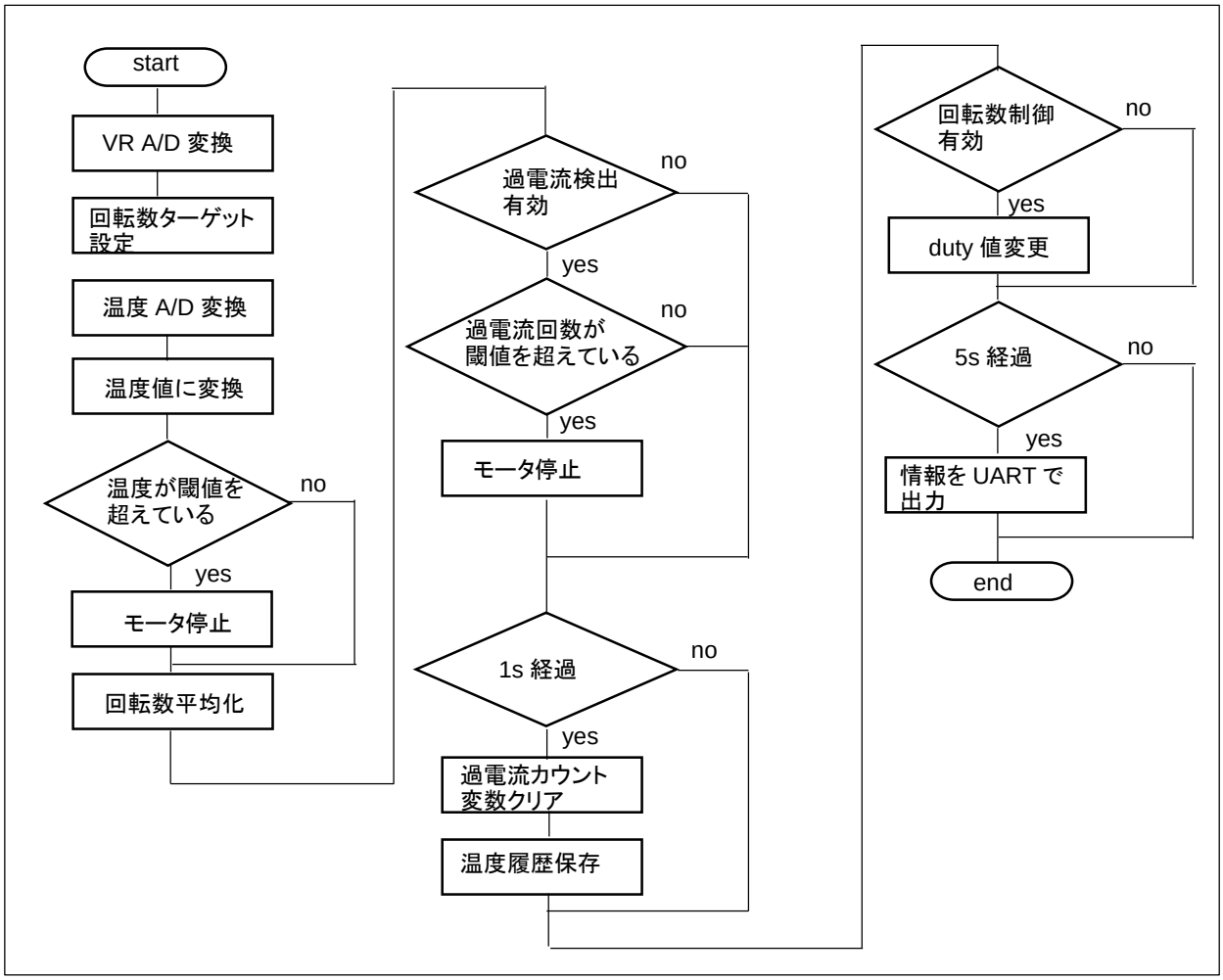
・回転数制御（目標回転数と現時点の回転数を比較して、モータ電流を増減させる）

を行います

引数：なし

戻り値：なし

ーフローチャートー



2.2.4. その他内部で呼び出される関数

blsm_hs_control

概要：モータ制御関数
宣言：void blsm_hs_control (void)
説明：
・センサー位置と回転方向に応じた磁界をかける処理
を行います
引数：なし
戻り値：なし

blsm_hs_pos_phase

概要：モータ始動制御関数
宣言：void blsm_hs_pos_phase (void)
説明：
・センサー位置と回転方向に応じた、1ステップ先の磁界をかける処理
を行います
始動時により大きなテンションをかけるために使用
引数：なし
戻り値：なし

blsm_hs_neg_phase

概要：モータ制御関数
宣言：void blsm_hs_neg_phase (void)
説明：
・センサー位置と回転方向に応じた、逆方向の磁界をかける処理
を行います
始動時に逆方向のテンションをかけるために使用
引数：なし
戻り値：なし

blsm_rpm_calc

概要：モータ制御関数
宣言：void blsm_rpm_calc (void)

説明:
・現時点の回転数
・回転方向
を取得します
引数:なし
戻り値:なし(グローバル変数に格納)

2.2.5. グローバル変数

(1)本プログラムの入力となる変数

g_blsm_target_direction

概要:モータの回転方向を決める変数
宣言:short g_blsm_target_direction
説明:
0:モータは停止します
1:モータは反時計回りします
-1:モータは時計回りします

g_blsm_target_rpm

概要:モータの回転数を決める変数
宣言:unsigned short g_blsm_target_rpm
説明:モータの目標回転数(rpm)を指定します
2000~20000 の値が有効です。

上記変数に値をセットすると、セットされた値に応じてモータの回転が制御されます。

(2)本プログラムで書き換えられる変数(抜粋)

以下の変数は任意のタイミングで参照可能です。

g_blsm_direction

概要:モータの回転方向を示す変数
宣言:short g_blsm_direction
説明:
0: モータは停止しています
1: モータは反時計回りで回転しています
-1: モータは時計回りで回転しています

127: 回転方向が検出できない

g_blsm_rpm

概要:モータの回転数(rpm)を示す変数
宣言:unsigned short g_blsm_rpm
説明:モータの回転数を rpm で示します

g_blsm_rpm_ave

概要:モータの回転数(rpm)の平均値示す変数
宣言:unsigned short g_blsm_rpm_ave
説明:モータの回転数を rpm で示します

g_blsm_temparature

概要:モータドライバボードの温度を示す変数
宣言:short g_blsm_temparature
説明:基板の温度[℃]を示します

g_blsm_control_method

概要:制御のステージを示す変数
宣言:unsigned char g_blsm_control_method
説明:
0: 停止状態
1: 始動制御(<100rpm)
2: 始動開始(1)(<500rpm)
3: 始動開始(2)(<1000rpm)
4: 通常制御(>1000rpm)

g_blsm_error_code

概要:エラー種別を示す変数
宣言:unsigned char g_blsm_error_code
説明:
0: エラーが発生していない
1: 過電流検出
2: 過熱検出
4:短絡(POE3)が検出された

※エラーが発生した場合は、本変数にエラー種別を設定してモータは停止します（g_blsm_target_direction=0 とります）
※エラーフラグのクリアは本変数に 0 を書き込むか、blsm_motor_stop()で行ってください
※エラーフラグが立った状態でも、g_blsm_target_direction=1, -1 をセットする事によりモータは再び回転します

2.2.6. プログラムの動作を制御する定義値

#define BLSM_OVER_CURRENT_CONTROL 1

過電流検出を行いモータを停止させる場合"1"を定義。

#define BLSM_OVER_CURRENT_COUNT_THRESHOLD 16000

過電流停止の閾値を定義する。50us 毎に過電流検出の端子レベルをチェックし、1s 間にこの回数以上過電流となっていた場合、モータを停止させる。（大きい数値を設定した方が、過電流によりモータが止まりにくくなる、最大値は 20000）
※20000 以上の値を設定すると、過電流でモータが止まることはない事に注意

#define BLSM_TEMPARATURE_CONTROL 1

過熱検出を行いモータを停止させる場合"1"を定義。

#define BLSM_TEMPARATURE_THRESHOLD 50

過熱停止を行う場合の閾値。[℃]で定義。

#define BLSM_ROTATION_SPEED_CONTROL 1

回転数制御を行う場合"1"を定義

#define BLSM_DUTY_MAX 735

設定される duty の最大値を定義（735/750=98%）

#define BLSM_DUTY_MIN 40

設定される duty の最小値を定義（40/750=5.3%）

#define BLSM_RPM_MAX 20000

設定可能な回転数の最大を定義[rpm]

#define BLSM_RPM_MIN 2000

設定可能な回転数の最小を定義[rpm]

#define BLSM_RPM_FEEDBACK_CONST 0.1

回転数制御のフィードバック係数を定義
0.1 の場合は、目標に対し 10%ずつ近づけていく

#define BLSM_RPM_FEEDBACK_THRESHOLD 0.1

回転制御の閾値を定義
0.1 の場合は、目標回転数との誤差 10%以内であれば、回転数（duty）を変更しない

#define BLSM_CONTROL_DUTY0 0

ステージ 0（停止時）の duty

#define BLSM_CONTROL_THRESHOLD1 100

ステージ 1（初期始動）を抜ける回転数[rpm]

#define BLSM_CONTROL_DUTY1_START 120

ステージ 1 の開始時の duty 120(/2000)

#define BLSM_CONTROL_DUTY1_MAX 300

ステージ 1 の duty の最大値(/2000)

#define BLSM_CONTROL_DUTY1_DIFF 5

ステージ 1 の duty の増分値
モータを始動させる際、この値を加算して duty を増やしていく

#define BLSM_CONTROL_CYCLE1 400

始動時 1 サイクル持続時間 50us×400 = 20ms

#define BLSM_CONTROL_CYCLE1_PRE 160

始動時に逆向きのテンションを加える時間 50us×160 = 8ms

#define BLSM_CONTROL_THRESHOLD2 500

ステージ 2 を抜ける回転数[rpm]

#define BLSM_CONTROL_DUTY2 150

ステージ 2 の duty(/2000)

#define BLSM_CONTROL_THRESHOLD3 1000

ステージ 3 を抜ける回転数[rpm]

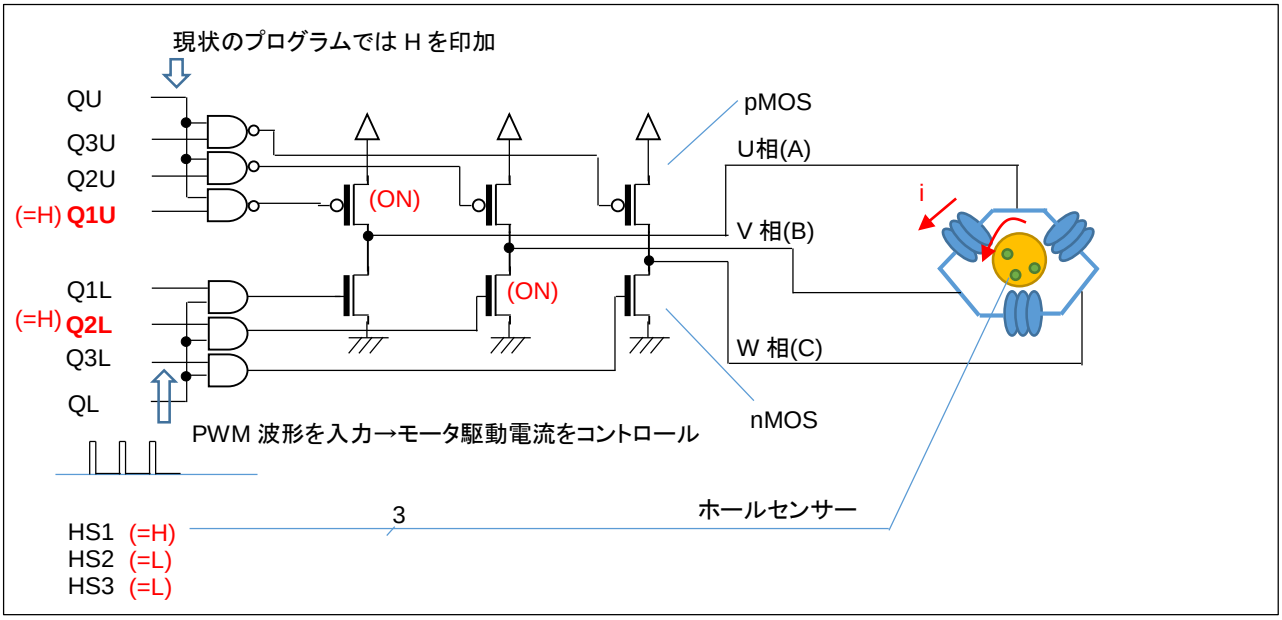
#define D_BLSM_CONTROL_DUTY3 200

ステージ 3 の duty(/2000)

2.2.1. プログラムで使用しているマイコン機能

機能名	分類	用途
CMT0	タイマー	モータ制御タイマー割り込み(10ms 周期)
CMT1	タイマー	モータ制御タイマー割り込み(50us 周期)
MTU3	タイマー	モータ制御
TPU1,2	タイマー	モータ回転数カウント
IRQ5,6,7	端子割り込み	ホールセンサ変化検出
POE3	アウトプットイネーブル	短絡検出
S12AD	A/D 変換	温度検出, VR 読み取り, 相電圧, 相電流

2.3. モータ制御アルゴリズム



ホールセンサーの出力に応じて、pMOS, nMOS を 1 つずつ ON させることにより、特定の相電流（上図では、U→V 相に電流を流している）を発生させ、モータのロータにトルクを発生させます。ホールセンサーの出力が切り替わるタイミングで、次の相電流に切り替える事により、回転を維持させます。

回転数に関しては、Q1L~Q3L に印加する波形の duty 比を変えてモータに流す電流を変化させる事により制御しています。QU と QL は H 固定で使用しています。

－ホールセンサ位置と電流の方向の関係－

HS2/HS1/HS0	正転（前進）	逆転（後進）
1 0 1	W→U	V→W
0 0 1	W→V	V→U
0 1 1	U→V	W→U
0 1 0	U→W	W→V
1 1 0	V→W	U→V
1 0 0	V→U	U→W

2.3.1. 始動制御

－始動制御－

ステージ	状態	回転数[rpm]	制御
ステージ 0	停止	0	
ステージ 1	始動制御	<100	(1)8ms 正方向のトルクをかける (2)通常の回転制御をかける -----20ms 以内に回転数が 100rpm に達すればステージ 1 に移行 (3)8ms 逆方向のトルクをかける (4)通常の回転制御をかける -----20ms 以内に回転数が 100rpm に達すればステージ 1 に移行
ステージ 2	始動開始(1)	<500	duty 制限（電流制限）がある通常の回転制御
ステージ 3	始動開始(2)	<1000	duty 制限（電流制限）がある通常の回転制御
ステージ 4	通常制御	>1000	目標の回転数に対し duty（電流）制御

始動制御は、上記の方式で行う。一度回転を始めると、ホールセンサーの値に応じた磁界をかければ良いが、始動時はモータの軸に揺さぶりをかける方向の磁界をかける事により初期の回転を促す。

2.3.2. 回転数制御

回転数制御は単純なアルゴリズムで行っています。
10ms 周期で、
目標の回転数 / 現在の回転数
を算出し、0.9~1.1（目標に対し、10%以内の差異）であれば、duty（電流）の変更は行わず、目標との差分が 10%以上であれば、差分の 10%値を増減させます。（例えば、目標 10,000rpm で、現在 5,000rpm の場合、電流値を 2 倍(10,000/5,000)の 10%である 20%増加させるよう制御を行います。）

2.4. 安全機構

2.4.1. 過電流保護

50us の割り込みルーチン内で、過電流フラグ（*INT=L）が立っていた場合、変数(g_blsm_over_current_count)をインクリメントする。
1 秒毎に、変数の値を定数(BLSM_OVER_CURRENT_COUNT_THRESHOLD)値と比較し、値が定数を超えていた場合、過電流状態と判断して、モータを停止させる。
g_blsm_over_current_count 変数は、1 秒毎にクリアされる。
1 秒間の平均で過電流を判断している理由は、1 回の過電流フラグで過電流の判断を行った場合、モータの突入電流で保護機構が働いてしまうケースがあるためです。

2.4.2. 過熱保護

10ms の割り込みルーチン内で、温度をモニタ（A/D 変換及びテーブル比較）し、温度が定数(BLSM_TEMPARATURE_THRESHOLD)を超えていた場合、過熱状態と判断してモータを停止させる。

取扱説明書改定記録

バージョン	発行日	ページ	改定内容
REV.1.0.0.0	2016.10.5	—	初版発行

お問合せ窓口

最新情報については弊社ホームページをご活用ください。

ご不明点は弊社サポート窓口までお問合せください。

株式会社 北斗電子

〒060-0042 札幌市中央区大通西 16 丁目 3 番地 7

TEL 011-640-8800　FAX 011-640-8801

e-mail：support@hokutodenshi.co.jp (サポート用)、order@hokutodenshi.co.jp (ご注文用)

URL:http://www.hokutodenshi.co.jp

商標等の表記について

- 全ての商標及び登録商標はそれぞれの所有者に帰属します。
- パーソナルコンピュータを PC と称します。

ルネサス エレクトロニクス RX71M(QFP-100 ピン)搭載
ブラシレスモータスタータキット

ブラシレスモータスタータキット(RX71M)
[ソフトウェア編]
取扱説明書

株式会社北斗電子

©2016 北斗電子 Printed in Japan 2016 年 10 月 5 日改訂 REV.1.0.0.0 (161005)