RX71M 24V モータキット
ソフトウェア資料

HSBRX71M100, HSBRX71M100-MIF, BLM_EV2 モータ制御ソフトウェア

-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.1.1.0.0

ー目　次ー

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、アクセスランプの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。

製品の故障や、データ消失の原因となります。

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。

概要

本書は、「RX71M 24V モータキット」付属 CD に含まれる、サンプルプログラムの解説を行う資料となります。

注意事項

本マニュアルに記載されている情報は、RX71M 24V モ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。

サンプルプログラム CD

製品に付属しているサンプルプログラム CD の内容を下記に示します。

フォルダ			内容
SOURCE¥	RX71M_24V_MOTOR_REV1¥		モータ制御サンプルプログラムソース（CS+プロジェクト）
BINARY¥			コンパイル済み mot ファイル格納フォルダ
DOCUMENT¥			
	RX71M_24V_MOTOR_Software_REV_X_s.pdf		本資料

サンプルプログラムは、ルネサスエレクトロニクス CS+(CS+forCC)向けのプロジェクトで作成されています。
CS+ for CC Ver7.0 以降を予めインストール願います。

マイコンボードへのプログラムの書き込みには、ルネサスエレクトロニクス RenesasFlashProgrammer が使用できますので、予めインストール願います。（CS+と E1 を使用してプログラムを書き込む際は、RenesasFlashProgrammer のインストールは必須ではありません）

1. サンプルプログラムの構成

1.1. CS+プロジェクトフォルダ

サンプルプログラムは、ルネサスエレクトロニクス CS+向けのプロジェクトとして作成されています。

CS+のプロジェクトフォルダは

SOURCE¥RX71M_24V_MOTOR_REV1

となっています。

1.2. プログラムで実装されている機能

本サンプルプログラムでは、以下の機能が実装されています。

・モータの正転・逆転
・回転数可変
・UVW 相電圧、UW 相電流、電源電圧、電源電流、ボリューム、温度センサの A/D 値取得、D/A コンバータ経由でのモニタ出力
・異常検出停止（過熱、過電流）

1.3. ソースファイルフォルダ構成

RX71M_24V_MOTOR_REV1¥src

以下には、SmartConfigurator で生成されたコードが格納されています。src 以下のコードを修正する場合は、基本的には SmartConfigurator の設定を変更し、コード生成を行ってください。

RX71M_24V_MOTOR_REV1¥ RX71M_24V_MOTOR.c

main()関数

RX71M_24V_MOTOR_REV1¥user_src

以下に、モータ制御の関数等が含まれます。

フォルダ	説明
user_src¥blsm	モータ制御関数
user_src¥sci	SCI(UART)ドライバ

2. モータ制御方式

2.1. 相補 PWM 制御

モータドライバボード(BLM_EV2)のモータ駆動回路（モータ駆動ロジック＋モータ駆動 FET）を示します。

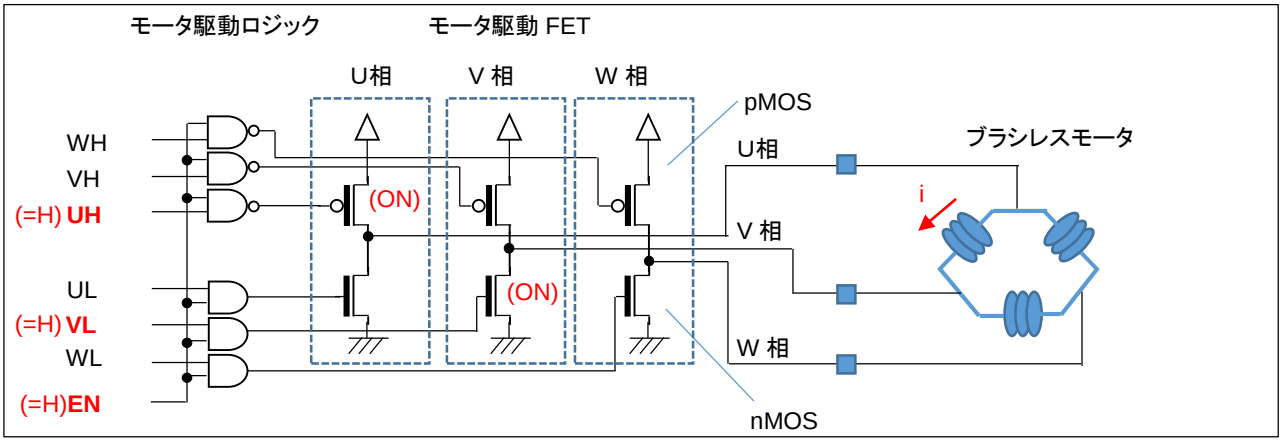


図 2-1　モータ駆動回路

各信号の接続先を表に示します。

表 2-1　信号接続

信号名	マイコン接続ポート	用途	備考
UH	PE5/GTIOC0A	U 相 H 側出力	
UL	PE2/GTIOC0B	U 相 L 側出力	
VH	PE4/GTIOC1A	V 相 H 側出力	
VL	PE1/GTIOC1B	V 相 L 側出力	
WU	PE3/GTIOC2A	W 相 H 側出力	
WL	PE0/GTIOC2B	W 相 L 側出力	
EN	PE6	Hi-Z 制御	モータドライバボード J4 1-2 ショートで EN 信号有効

各信号は正論理となっており、H で ON する動作となります。EN=UH=VL=H, その他=L の設定で、モータの U 相端子から V 相端子へ電流が流れる制御となります。

本サンプルプログラムは、3 相の相補 PWM 波形でモータの回転に必要な磁界を生成しています。

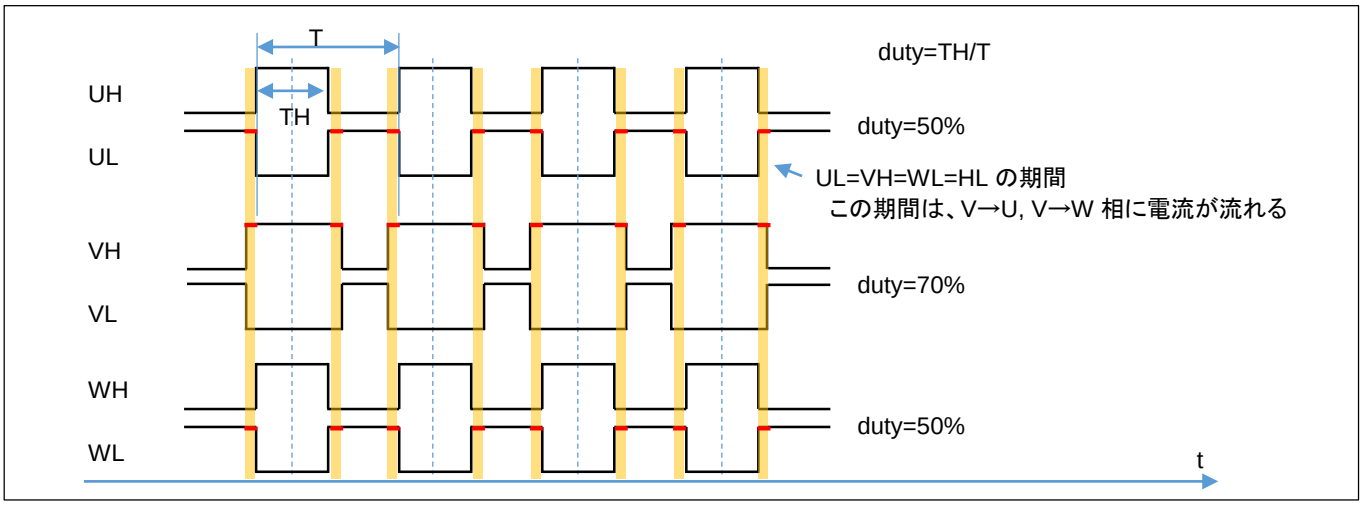


図 2-2　相補 PWM 波形

図 2-2 の相補 PWM では、duty の差分（70-50=20%）の時間だけ、V→U, V→W に電流が流れる事となります。3 相の duty を任意の値に設定する事により、「電流の大きさ」と「どの向きに電流を流すか」の制御が行えます。

・XY-3 相変換

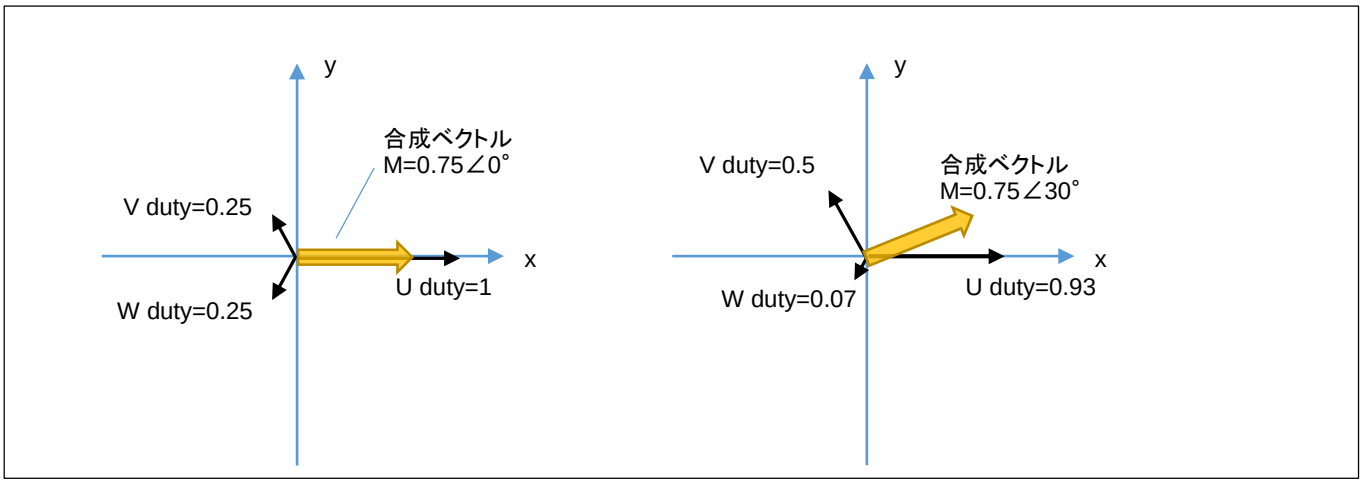


図 2-3　XY-3 相変換

サンプルプログラムでは、モータに回転磁界を与える際、UVW の合成ベクトルの大きさ(duty)と角度を与える様になっています。プログラムでは、与えられた合成ベクトルの大きさ(r)と角度(θ)を UVW の各相の PWM duty に変換しています。

dutyU = (r × 4/3 × cos(θ)) / 2 + 0.5
dutyV = (r × 4/3 × cos(θ–120°)) / 2 + 0.5
dutyW = (r × 4/3 × cos(θ–240°)) / 2 + 0.5

(*1) r=0~0.75 の範囲を取る場合、4/3 を乗ずる
　　r=0~1 の範囲とする場合（1 に正規化した duty を与える場合）は、4/3 は不要

※dutyU=1, dutyV=0, dutyW=0 とした場合は、1∠0° の出力が得られるが、角度によっては UVW の合成で 100% の duty を得られない

※全角度で、86.7%（√3 /2）の duty を得られる変換式は存在するが、その場合 UVW の変化が正弦波とならない

UVW を正弦波で変化させる場合、合成ベクトルの duty の最大値は 75%となると考えます。

※120 度制御では、最大 100%の duty を与えることができるが、正弦波相補 PWM では加える事ができる最大 duty は 75%に制限されるものと考えます。

・デッドタイム制御

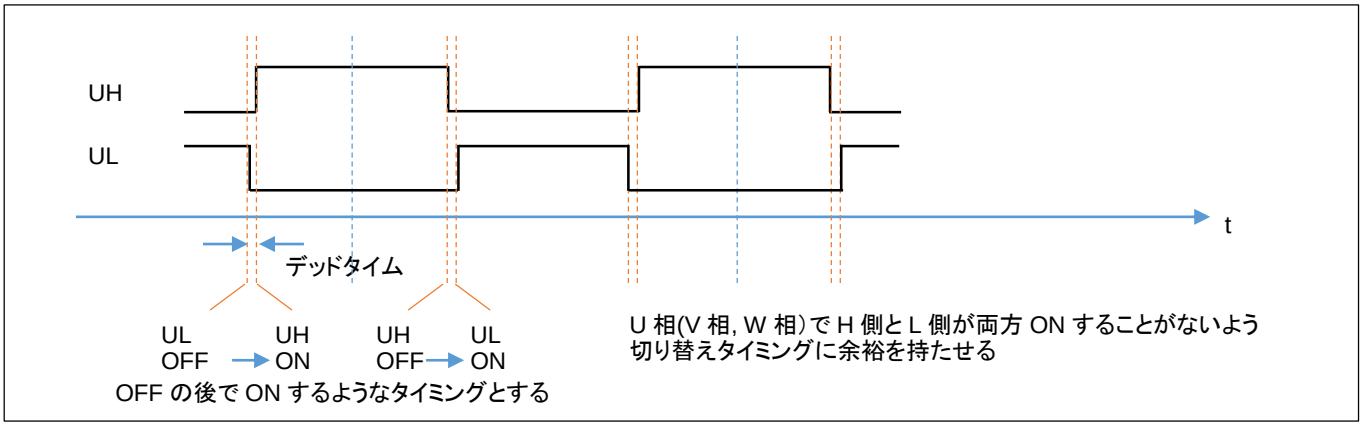


図 2-4　デッドタイム制御

モータ駆動 FET の駆動波形で、H 側と L 側は常に反転（相補）する波形としているが、単に極性を逆にするわけではなく、OFF, ON のタイミングに時間差を設けている。同じ相の H 側(pMOS)と L 側(nMOS)が、両方 ON した場合、電源間のショートとなり、FET の焼損や、電源ダウンの原因となるため、確実にペアとなる相手側が OFF した後で、ON 制御を行う。

2.2. 回転制御

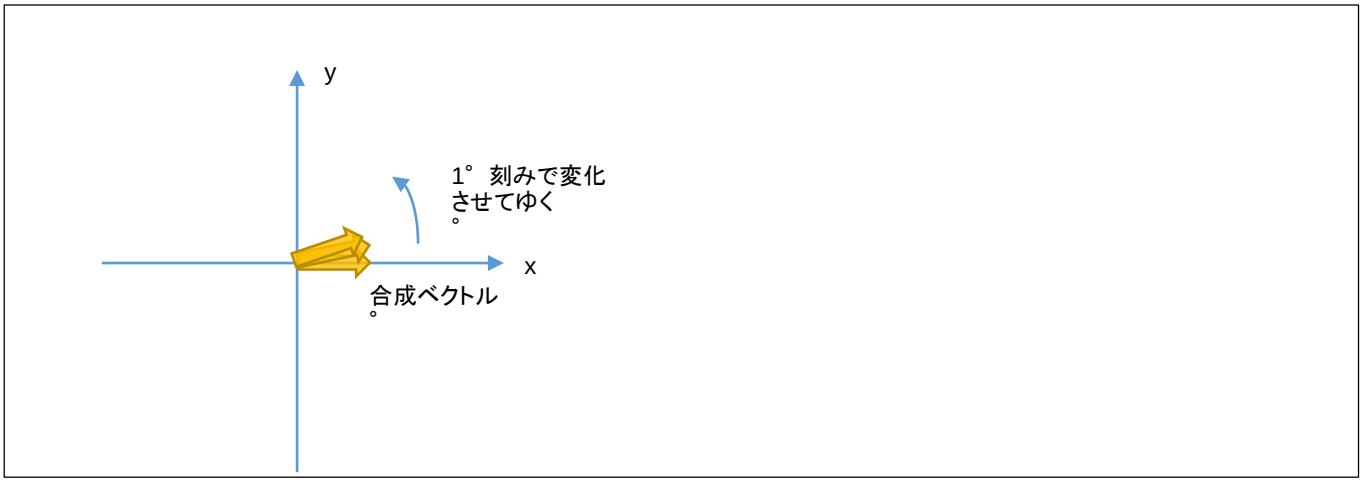


図 2-5　回転制御

本サンプルプログラムでは、モータを回転させる際、UVW の合成ベクトルを回転方向に応じ一定時間毎に変化させる方式としています。

今回のターゲットのモータは、磁界を 360° 回転させた場合、モータが 1/4 回転する動作となりましたので、1 回転で 360 x 4 = 1440° 合成ベクトルの角度を変化させています。

2.3. 回転数制御

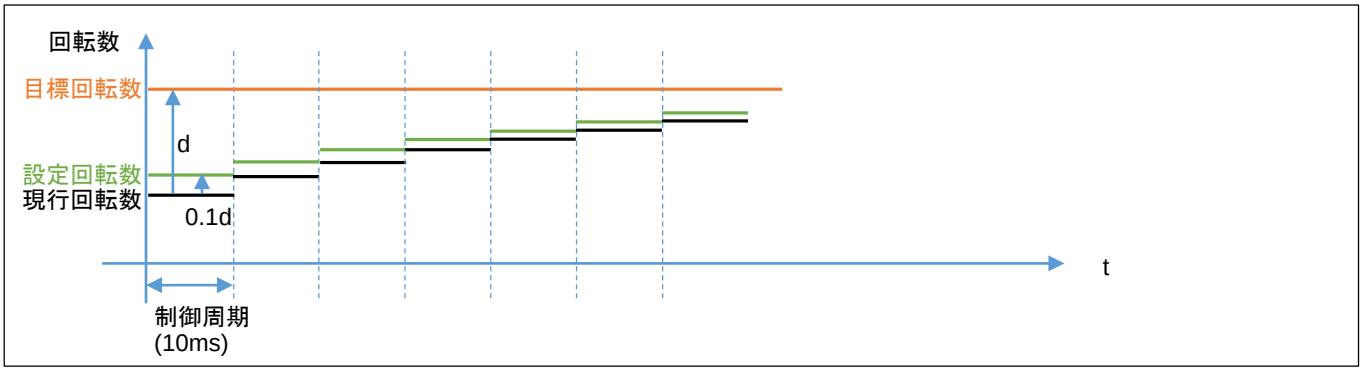


図 2-6　回転数制御

本サンプルプログラムでは、現行の回転数を取得し、目標とする回転数との差分(d)を算出します。差分(d)の 10%だけ目標回転数に近づけた値を「設定回転数」とます。「設定回転数」に応じた回転制御（設定回転数に対応する周期で、UVW の合成ベクトルを変化させてゆく）を行います。なお、目標回転数と現行回転数の差が 10%以内の場合は、設定回転数を目標回転数に設定します。

2.4. 始動制御

モータ始動時は、設定回転数を 600rpm に設定し、600rpm 固定で合成ベクトルを回していきます。

始動制御の終了条件である、
・エンコーダセンサが回転ターゲットと同じ方向の回転を一定回数（4000 回）以上検出
が満たされた場合は、始動制御を終了し、通常の回転数制御に移行します。

始動制御中は、回転数固定で以下の duty 制御を行います。

(1)回転数が始動時の設定回転数(600rpm)の 80%未満
(2)エンコーダセンサが回転ターゲットと同じ方向の回転を検出した回数が一定回数（100 回）未満
(1) OR (2)の場合は、duty を増加させる。

(3)回転数が始動時の設定回転数(600rpm)の 150%超過
(4)エンコーダセンサが回転ターゲットと同じ方向の回転を検出した回数が一定回数（100 回）以上
(3) AND (4)の場合は、duty を減少させる。

duty の増分（減少）値は、収束を速めるため、

10% → 5% → 2.5% ...
の様に徐々に変化量を小さくしていきます。

2.5. duty 制御

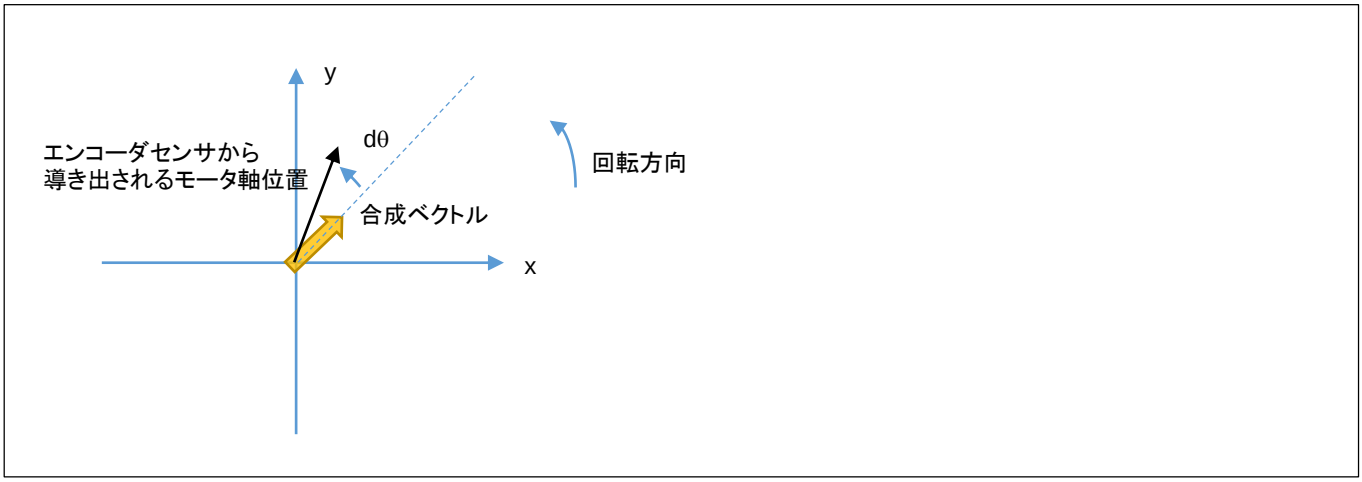


図 2-7　duty 制御

一定周期(50us)毎に、エンコーダセンサ位置をモニタし、前回から 30°　変化したタイミングで、エンコーダセンサの軸位置と現在の印加磁界（合成ベクトル）の差分(dθ)をモニタする。

差分が 3°　以上の場合、遅れが出ている場合は、duty を 0.05%増加させ、進みが出ている場合は duty を 0.05%減少させます。（図 2-7 の場合は、遅れ）

duty の変化値は、制御周期(10ms)間、変数に累積値を格納し、制御周期(10ms)毎に、制御に反映させます。

2.6. 脱調防止制御

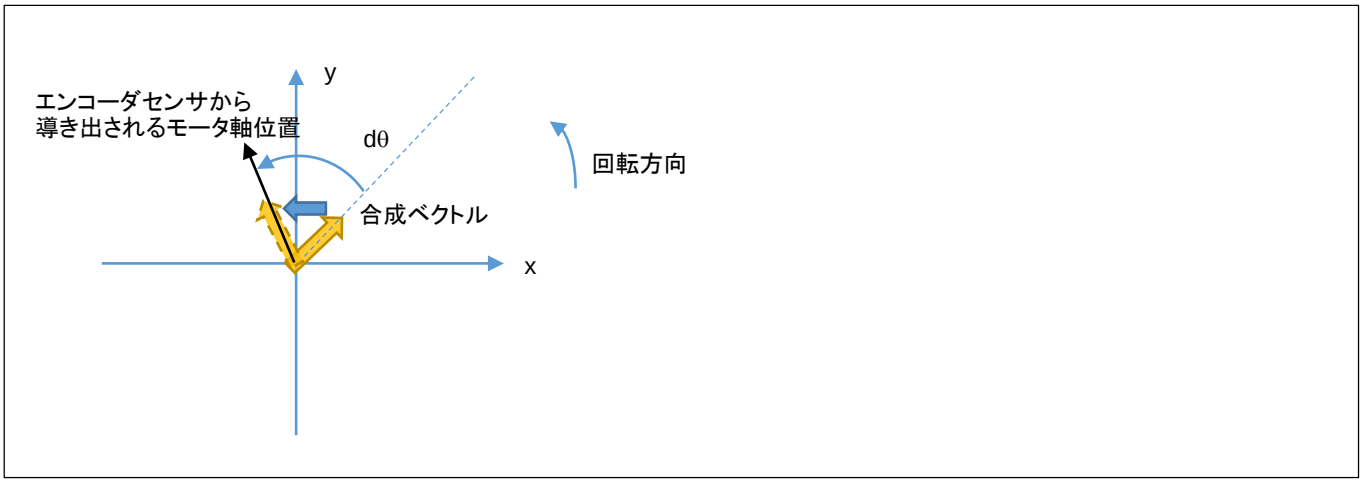


図 2-8　脱調防止制御

2.5 の duty 制御で、エンコーダセンサと現在の印加磁界（合成ベクトル）の差分(dθ)が、20°　以上ある場合は、

・duty を大きく(0.2%)変化させる

・現在の印加磁界の方向をエンコーダセンサに合わせる

という制御を行います。印加磁界の角度とモータの軸位置が大きくずれると、脱調を起こすため、強制的に印加磁界を軸位置にセットします。

3. 関数、使用変数

3.1. マイコン使用機能

表 3-1 プログラムで使用しているマイコン機能

使用コンポーネント	用途	備考
S12AD0/S12AD1	Analog 電圧値取得	
RSPI	D/A コンバータへのデータ送信	
SCI1/SCI2	デバッグメッセージの表示	
CMT0	回転磁界の生成	回転数に応じて周期を設定
CMT1	モータ制御	50us 毎
CMT2	ADC 定期呼び出し	20us 毎
CMT3	D/A 送信, SCI 表示	40us
CMTW0	モータ制御	10ms 毎
GPT0/GPT1/GPT2	3 相相補 PWM 波形生成	
TMR0_TMR1	回転数取得	
MTU0/MTU1/MTU8	エンコーダセンサ値計測	

プログラムでは、マイコンの上記機能を使用しています。

表 3-2 プログラムで使用している割り込み

割り込みレベル	呼び出し元	用途
14	RSPI0	SPI データ送信割り込み
13	S12AD1	ADC 変換終了
12	S12AD0	ADC 変換終了
12	MTU1	エンコーダ Z 相
11	CMT2	ADC 変換開始
9	CMT0	回転磁界生成
8	SCI1/SCI2	SCI（読み出し）
7	CMT1	50us 毎モータ制御
5	CMTW0	10ms 毎モータ制御
3	CMT3	SCI バッファ表示処理、D/A 送信

プログラムでは、マイコンの上記割り込みを使用しています。

3.2. ユーザ関数仕様(blsm.c)

blsm_init

概要:初期化関数
宣言:
　　void blsm_init(void)
説明:
　・変数の初期化を行います

引数:
　　なし
戻り値:
　　なし

　モータを一旦停止させた後、再度始動する場合は、本関数を呼び出してください。

blsm_adc_matrix_init

概要:初期化関数
宣言:
　　void blsm_adc_matrix_init(void)
説明:
　・ADC 値保持変数の初期化を行います

引数:
　　なし
戻り値:
　　なし

blsm_cos_table

概要:cos 値算出
宣言:
　　float blsm_cos_table(short angle)
説明:
　・cos 値を返します

引数:

short angle　角度(1°単位)

戻り値:

与えた角度の cos 値

blsm_uvw_duty

概要:XY 平面の r, θから UVW 相の duty を算出する

宣言:

void blsm_uvw_duty(short angle, float duty, float *uvw)

説明:

・XY 平面の、r(ベクトルの大きさ), θ(角度)を、モータに印加する UVW 相の duty 比に変換します

引数:

short angle　角度(1°単位)

float duty　ベクトルの大きさ:0-0.75

float *uvw　計算結果 UVW3 相分 float[3]

戻り値:

なし

blsm_pwm_duty_set

概要:UVWduty 値変更

宣言:

void blsm_pwm_duty_set(float *uvw)

説明:

・UVW 相の duty 値を GPT0/1/2 のレジスタ値に設定します

引数:

float *uvw　UVW duty 値 UVW3 相分 float[3]

戻り値:

なし

blsm_ltc2620_reg_write

概要:D/A コンバータデータ送信

宣言:

unsigned short blsm_ltc2620_reg_write(unsigned char port_flag, unsigned short *data)

説明:

・D/A コンバータにデータを送信します

引数:

unsigned char port_flag　値を設定したい D/A コンバータのポート指定

0x03: ポート A,B（2 ポートの送信）, 0xff: ポート A-H（8 ポートの送信）

unsigned short *data　送信データポインタ(*1)

戻り値:

0x0000　正常終了

0x0001　引数エラー

0x0002　データ送信中

(*1)（変数寿命が切れない）グローバル変数のアドレスを与えてください

blsm_encorder_init

概要:A, B, Z 相を使用するエンコーダ初期化

宣言:

void blsm_encorder_init(void)

説明:

・MTU の位相計数モード（A,B,Z 相）を設定、初期化

引数:

なし

戻り値:

なし

blsm_encorder_start

概要:A, B, Z 相を使用するエンコーダスタート

宣言:

void blsm_encorder_start(void)

説明:

・MTU の位相計数モード（A,B,Z 相）の計数開始

引数:

なし

戻り値:

なし

blsm_encorder_stop

概要:A, B, Z 相を使用するエンコーダストップ
宣言:
　　void blsm_encorder_start(void)
説明:
　・MTU の位相計数モード（A,B,Z 相）の計数停止

引数:
　　なし
戻り値:
　　なし

blsm_enc2angle

概要:エンコーダセンサ位置から理想的な印加磁界位置を算出
宣言:
　　unsigned short blsm_enc2angle(void)
説明:
　・エンコーダセンサ位置に応じた理想的な印加磁界の角度を算出します

引数:
　　なし
戻り値:
　　角度(0-360)

blsm_rpm2cmcor

概要:RPM 値からタイマ設定値を算出
宣言:
　　unsigned short blsm_rpm2cmcor(unsigned short rpm)
説明:
　・回転数[rpm]を、CMT0 のタイマ設定値に変換します

引数:
　　unsigned short rpm　　回転数[rpm]
戻り値:

CMCOR レジスタ設定値

blsm_enc_pos

概要:エンコーダ値取得

宣言:

unsigned short blsm_enc_pos(void)

説明:

・エンコーダ値を 0-8191 の絶対値に変換します

・回転方向に拘わらず位置と値を 1:1 で対応させた値を返します

引数:

なし

戻り値:

エンコーダカウンタ位置(0-8191)

0xffff　異常値

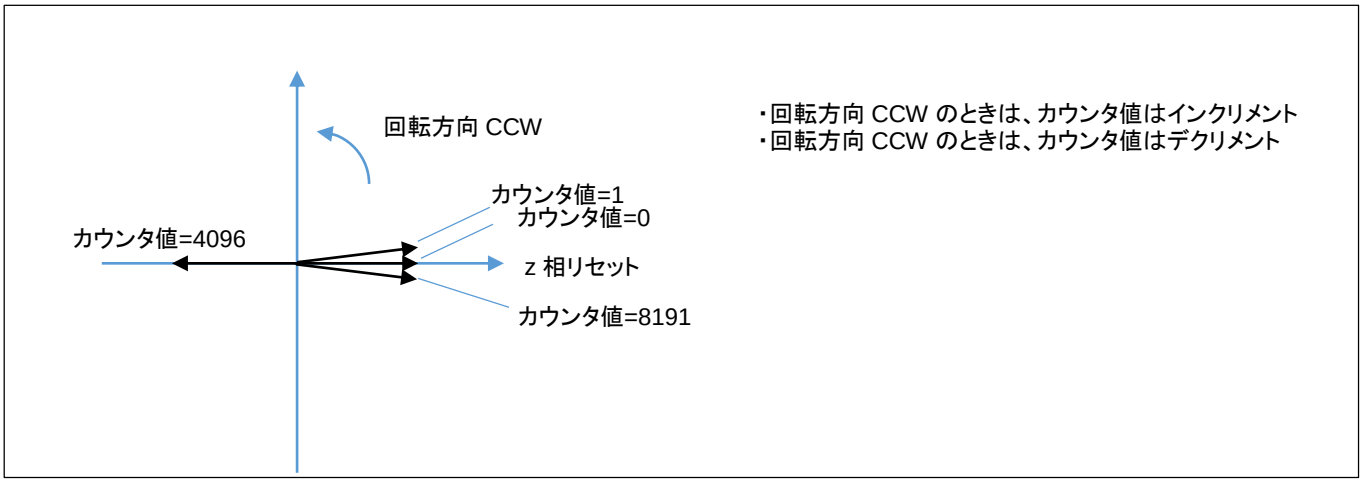


図 3-1　エンコーダ値

blsm_enc_direction

概要:回転方向検出

宣言:

short blsm_enc_direction(unsigned short enc_pos, unsigned short prev_enc_pos)

説明:

・現在の回転方向を求めます

・回転方向は与えられた 2 点が円周の近いほうを通って遷移したと考えます

引数:

unsigned short enc_pos　現在のエンコーダ位置

unsigned short prev_enc_pos　1 周期前のエンコーダ位置

戻り値:

1　回転方向は CCW

-1　回転方向は CW

0　停止（1 周期前のエンコーダ位置から動いていない）

127　異常値

blsm_enc_diff

概要:エンコーダ移動量算出

宣言:

unsigned short blsm_enc_diff(unsigned short enc_pos, unsigned short prev_enc_pos)

説明:

・2 点間のエンコーダ値の差分を計算します

・与えられた 2 点が円周の近いほうを通って遷移したと考えます

引数:

unsigned short enc_pos　現在のエンコーダ位置

unsigned short prev_enc_pos　1 周期前のエンコーダ位置

戻り値:

エンコーダ差分値

blsm_adc_average_val

概要:平均値の算出

宣言:

void blsm_adc_average_val(void)

説明:

・ADC の初期値の平均値を求めます

・200 点のデータの内、最小 10 点、最大 10 点を除いた 180 点のデータを相加平均します

引数:

なし（グローバル変数から値を読み出し）

戻り値:

なし（グローバル変数に格納）

blsm_temp_val

概要:温度値の算出
宣言:
　short blsm_temp_val(unsigned short ad_val)
説明:
・ADC の値から温度センサの温度値を計算
・予め計算されたテーブルを参照します

引数:
　unsigned short ad_val　A/D 変換値
戻り値:
　温度[℃]

blsm_target_rpm_val

概要:目標回転数の算出
宣言:
　unsigned short blsm_target_rpm_val(unsigned short ad_val)
説明:
・ボリュームの設定値を回転数に変換

引数:
　unsigned short ad_val　A/D 変換値
戻り値:
　回転数[rpm]

3.3. ユーザ関数仕様(sci.c)

sciInit

概要:初期化
宣言:
　void sciInit(unsigned char ch);
説明:
・SCI チャネルの初期化

引数:
　unsigned char ch　初期化チャネル(1-2)
　　1:マイコンボード J5(20P), 2:インタフェースボード J2(USB-miniB)

戻り値:

なし

sciRead

概要:キーボード読み取り

宣言:

char sciRead(unsigned char ch);

説明:

・キーボードから 1 文字の読み取りを行う

引数:

unsigned char ch　チャネル(1-2)

戻り値:

キー読み出し値(0xff: 読み出しデータなし)

sciWrite, sciWriteBuf

概要:1 文字出力

宣言:

void sciWrite(unsigned char ch, unsigned char Buffer);

void sciWriteBuf(unsigned char ch, unsigned char Buffer);

説明:

・端末に 1 文字出力を行う

引数:

unsigned char ch　チャネル(1-2)

unsigned char Buffer　出力する文字

戻り値:

なし

※Buf 付き関数は、本関数ではバッファリングのみ行い、一定期間毎の割り込みで文字出力
Buf なし関数は、SCI のバッファレジスタへの書き込みが完了するまで関数の実行が終了しない
モータ回転中等、ウェイトが入ると困る場合は、Buf 付きを用いる

sciPrint, sciPrintBuf

概要:文字列出力

宣言:

void sciPrint(unsigned char ch, char *Buffer);

void sciPrintBuf(unsigned char ch, char *Buffer);

説明:

・端末に文字列の出力を行う

引数:

unsigned char ch　チャネル(1-2)

char *Buffer　出力する文字列（¥0 終端, ¥r は¥r¥n に変換）

戻り値:

なし

sciPrintByte, sciPrintByteBuf

概要:HEX 出力

宣言:

sciPrintByte(unsigned char ch, unsigned char dmy);

sciPrintByteBuf(unsigned char ch, unsigned char dmy);

説明:

・端末に 1 バイトを hex 表記で表示する

引数:

unsigned char ch　チャネル(1-2)

unsigned char dmy　出力する値

戻り値:

なし

sciPrintWord, sciPrintWordBuf

概要:HEX 出力

宣言:

void sciPrintWord(unsigned char ch, unsigned short dmy);

void sciPrintWordBuf(unsigned char ch, unsigned short dmy);

説明:

・端末に 2 バイトを hex 表記で表示する

引数:

unsigned char ch　チャネル(1-2)

unsigned short dmy　出力する値

戻り値:

なし

sciPrintDword, sciPrintDwordBuf

概要:HEX 出力

宣言:

void sciPrintDword(unsigned char ch, unsigned long dmy);

void sciPrintDwordBuf(unsigned char ch, unsigned long dmy);

説明:

・端末に 4 バイトを hex 表記で表示する

引数:

unsigned char ch　チャネル(1-2)

unsigned long dmy　出力する値

戻り値:

なし

sciPrintByteInt, sciPrintByteIntBuf

概要:10 進出力

宣言:

void sciPrintByteInt(unsigned char ch, unsigned char dmy);

void sciPrintByteIntBuf(unsigned char ch, unsigned char dmy);

説明:

・端末に 1 バイトを 10 進表記で表示する

引数:

unsigned char ch　チャネル(1-2)

unsigned char dmy　出力する値

戻り値:

なし

sciPrintShortInt, sciPrintShortIntBuf

概要:10 進出力

宣言:

void sciPrintShortInt(unsigned char ch, unsigned short dmy);

void sciPrintShortIntBuf(unsigned char ch, unsigned short dmy);

説明:

・端末に 2 バイトを 10 進表記で表示する

引数:

unsigned char ch　チャネル(1-2)

unsigned short dmy　出力する値

戻り値:

なし

sciPrintShortSignedInt, sciPrintShortSignedIntBuf

概要:符号付き 10 進出力

宣言:

void sciPrintShortSignedInt(unsigned char ch, short dmy);

void sciPrintShortSignedIntBuf(unsigned char ch, short dmy);

説明:

・端末に 2 バイトを 10 進表記(符号付き)で表示する

引数:

unsigned char ch　チャネル(1-2)

short dmy　出力する値

戻り値:

なし

sciPrintLongInt, sciPrintLongIntBuf

概要:10 進出力

宣言:

void sciPrintLongInt(unsigned char ch, unsigned long dmy);

void sciPrintLongIntBuf(unsigned char ch, unsigned long dmy);

説明:

・端末に 4 バイトを 10 進表記で表示する

引数:

unsigned char ch　チャネル(1-2)

unsigned long dmy　出力する値

戻り値:

なし

3.4. 定義定数

・blsm.c

```
#define BLSM_BOARD_IDENTIFY_MESSAGE"[DRIVE SIDE]"
```

起動時に、端末（インタフェースボード J2, USB-miniB）に表示される追加メッセージを指定。2 枚のボードの識別のため。

```
#define BLSM_PWM_CYCLE   1500     //25us-> 40kHz
```

3 相相補 PWM 波形のサイクルを指定。40kHz, 25us 周期の場合は、1500

```
#define BLSM_DEAD_TIME   50
```

デットタイム時間を指定。

```
#define BLSM_BREAK_DUTY   0.5
```

ブレーキ時は、UVW 各相の duty を 50%に設定

```
#define BLSM_LTC2620_CMD_REG_WRITE   0x0
#define BLSM_LTC2620_CMD_REG_WRITE_AND_ALL_OUTPUT_EN   0x2
```

D/A コンバータチップ(LTC2620)に送信するコマンドを定義

```
#define BLSM_AD_VAL_NUM   3000     //20us x 3000point
#define BLSM_AD_VAL_INDEX   10
#define BLSM_INFO_VAL_INDEX   4
```

A/D 変換値の保存を 3000 ポイント（20us 刻み）、A/D 変換値 10 種，その他情報 4 種に設定

```
#define BLSM_LED_OFF   0
#define BLSM_LED_ON   1
#define BLSM_LED_BLINK   2
#define BLSM_LED_BLINK_FAST   3
#define BLSM_LED_BLINK2   4
#define BLSM_LED_BLINK3   5
```

```
#define BLSM_LED_BLINK4  6
```

LED の点滅パターン定義

```
#define BLSM_RPM_MAX  5000    //VR での設定回転数最大 5000rpm
#define BLSM_RPM_MIN  300    //VR での設定回転数最小 300rpm
```

設定可能な最大、最小の回転数

```
#define BLSM_DIRECTION_CCW  1
#define BLSM_DIRECTION_CW  -1
#define BLSM_DIRECTION_STOP  0
```

回転方向の定義

```
#define BLSM_DIRECTION_STABLE_THRESHOLD1  100
#define BLSM_DIRECTION_STABLE_THRESHOLD2  4000
//50us x 4000 = 0.2s これ以上安定した場合 phase2 に移行
#define BLSM_DIRECTION_STABLE_THRESHOLD3  4500
//BLSM_DIRECTION_STABLE_THRESHOLD2 と 3 の差分 (4500-4000)=500 50us x 500 = 25ms
これ以上安定しない場合は phase1 に戻る
```

回転方向の安定閾値の設定
100 回同じ方向に回転しない場合は、duty を増加させる
4000 回同じ方向に回転した場合は、始動制御(phase1)から通常制御(phase2)に移行
4500 回以上同じ方向に回っている場合は、回転安定を示す変数値を 4500 にセットする
※回転安定（4500 回以上同じ方向）時、500 回回転が安定しない状態となると、始動制御(phase1)に戻る

```
#define BLSM_ENCORDER_OFFSET_CCW  300    //300 度
#define BLSM_ENCORDER_OFFSET_CW  60    //60 度
```

脱調防止制御の際、回転方向が CCW の時は 300° のオフセット、CW のときは 60° オフセットを掛ける。

```
#define BLSM_ANGLE_THRESHOLD1  3    //duty 増減の閾値[°]
#define BLSM_ANGLE_THRESHOLD2  20    //印加磁界を強制的にシフトさせる閾値[°]
```

通常制御(phase2)の時、3° 以上のずれを検出すると、duty の増減
20° 以上のずれを検出すると、脱調防止制御を行う

```
#define BLSM_DUTY_DIFF1  0.0005
```

```
//0.05%, BLSM_ANGLE_THRESHOLD1  のずれを検出した際の増分
#define BLSM_DUTY_DIFF2  0.002
//0.2%,   BLSM_ANGLE_THRESHOLD2  のずれを検出した際の増分
#define BLSM_DUTY_MAX  0.75    //duty 設定最大値
#define BLSM_DUTY_MIN  0.0    //duty 設定最小値
```

通常制御(phase2)の時、3°以上のずれを検出すると、duty を 0.005%増減させる
脱調防止制御の際、0.2% duty を増減させる
duty の設定範囲は、0-75%とする

```
#define BLSM_ADC_INIT_AVERAGE_NUM  200
//200(-20)点のデータを平均化  BLSM_AD_VAL_NUM(3000)未満、21 以上
```

電源投入後、ADC 値の平均値を算出するデータ数
200 を指定した場合、最大、最小の各 10 点を除いた 180 点のデータを平均化する

```
#define BLSM_OVER_TEMP_STOP  1  //0:無効  1:有効
#defineBLSM_OVER_TEMP_THRESHOLD  70  //70℃
```

過熱停止機能有効、70℃で停止

```
#define BLSM_OVER_CURRENT_STOP  1    //0:無効  1:有効
#defineBLSM_OVER_CURRENT_THRESHOLD  3.0    //3A
#defineBLSM_OVER_CURRENT_COUNT_THRESHOLD  100
//10ms 期間中  50us 刻みで 100 回
```

過電流停止機能有効、3A、50us 毎に電流値をモニタし、10ms の期間中に 100 回閾値を超えた場合停止

```
#define BLSM_ERROR_OVER_CURRENT  0x1    //過電流停止
#define BLSM_ERROR_OVER_TEMP  0x2    //過熱停止
#define BLSM_ERROR_DA  0x4    //D/A 変換データ送信エラー
#define BLSM_ERROR_AD0  0x8    //A/D 変換エラー（AD0 側）
#define BLSM_ERROR_AD1  0x10    //A/D 変換エラー（AD1 側）
```

エラーコード定義

```
#define BLSM_DISPLAY_INFO_INTERVAL  300    //10ms x 300 = 3 秒間隔
```

回転数等の情報の表示頻度、3 秒 1 回

#define BLSM_DEBUG_PRINT1 1   //0:非表示, 1:表示

通常制御(phase2)の時、3°以上のずれや 20°以上のずれがあった場合画面表示を行う

#define BLSM_DEBUG_PRINT2 0   //0:非表示, 1:表示

BLSM_DEBUG_PRINT1より詳細な情報を表示する

#define BLSM_DEBUG_PRINT3 0   //0:非表示, 1:表示

モータを停止させた際、直前の A/D 変換値（3000 ポイント）を画面表示する

#define BLSM_DEF_DEBUG_COUNTER   //定義，非定義

この変数を定義した場合、デバッグ向けに、各コンペアマッチタイマが呼び出された回数を変数に格納する

//#defineBLSM_TARGET_RPM_FIX   //定義，非定義

この変数を定義した場合、回転数を固定とする
未定義の場合、ボリュームで回転数を制御する

#define BLSM_DEF_DEBUG_PORT   //定義，非定義

この変数を定義した場合、コンペアマッチタイマの割り込みルーチンに入った際、PC4-PC7（マイコンボード J3-23 - J3-26）を H とし、割り込みルーチンを抜ける際に L とする

・sci.c

#define SCI_BUF_SIZE 2048

Buf 付き関数で使用する出力のバッファサイズ（バイト）

3.5. グローバル変数

・blsm.c

```
unsigned short g_blsm_error_flag=0;
```

エラーフラグ
（b0: 過電流停止, b1: 過熱停止, b2: D/A データ送信エラー（40us 周期に間に合っていない場合は除く）
b3: A/D ch0 変換エラー（20us 周期に間に合っていない）, b4: A/D ch1 変換エラー（20us 周期に間に合っていない））

```
unsigned short g_blsm_angle=0;
```

磁界印加角度(0-359), ° 単位

```
float g_blsm_duty = 0.0;
```

モータ磁界印加 duty（0-0.75)

```
unsigned char g_blsm_ad0_flag=0;
```

AD0 変換中を示すフラグ

```
unsigned short g_blsm_voltage_adval[3]={0,0,0};     //UVW 電圧値
unsigned short g_blsm_current_adval[3]={0,0,0};     //UVW 電流値（V の電流はセンサー未接続）
unsigned short g_blsm_power_v_adval=0;     //電源電圧値
unsigned short g_blsm_power_i_adval=0;     //電流値
```

A/D 変換値格納変数

```
unsigned char g_blsm_ad1_flag=0;
```

AD1 変換中を示すフラグ

```
unsigned short g_blsm_vr_adval=0;     //モータドライバボード VR 値
unsigned short g_blsm_temp_adval=0;     //温度（サーミスタ）電圧値
```

A/D 変換値格納変数

unsigned short g_blsm_ad_val[BLSM_AD_VAL_NUM][BLSM_AD_VAL_INDEX];

A/D 変換値の履歴を保存する変数（デフォルト 3000 ポイント）（リングバッファ）

unsigned short g_blsm_ad_val_index=0;

A/D 変換値の履歴のインデックス（リングバッファのインデックス）

unsigned short g_blsm_ad_val_average[BLSM_AD_VAL_INDEX];

A/D 変換値の初期値の平均値を格納する変数

unsigned short g_blsm_info_val[BLSM_AD_VAL_NUM][BLSM_INFO_VAL_INDEX];

A/D 変換のタイミングで保存される情報を格納する変数

unsigned short g_blsm_array_work[BLSM_ADC_INIT_AVERAGE_NUM];

ソート用のワーク変数

unsigned short g_blsm_enc_pos=0;

現在のエンコーダ位置を示す変数

unsigned short g_blsm_prev_enc_pos=0;

1 周期(50us)前のエンコーダ位置を示す変数

short g_blsm_enc_direction;

現在の回転方向を示す変数（0:停止, 1:CCW, -1:CW）

unsigned short g_blsm_enc_rpm;

現在の回転数[rpm]を示す変数

unsigned short g_blsm_rpm_ave=0;

現在の平均の回転数[rpm]を示す変数（10ms 毎 16 ポイントの平均化）

unsigned short g_blsm_rpm_hist[16];

回転数の履歴を保存する変数

```
unsigned short g_blsm_temp_ave=0;
```

現在の平均化された温度を示す変数（10ms 毎 16 ポイントの平均化）

```
unsigned short g_blsm_temp_hist[16];
```

温度の履歴を保存する変数

```
unsigned short g_blsm_current_ave;
```

現在の平均化された電源電流値を示す変数（10ms 毎 16 ポイントの平均化）

```
unsigned short g_blsm_current_hist[16];
```

電流の履歴を保存する変数

```
unsigned short g_blsm_hist_index=0;
```

回転数、温度、電流の履歴向けのインデックス

```
unsigned long g_blsm_current_work=0;
unsigned long g_blsm_current_sum=0;
unsigned short g_blsm_current_index=0;
```

50us 毎の電流観測値を、平均化するワーク変数

```
unsigned short g_blsm_current_over_count=0;
unsigned short g_blsm_current_over_count_max=0;
```

過電流検出された回数とその最大値

```
unsigned short g_blsm_current_over_threshold=4095;
```

過電流検出の閾値（A/D 変換値に変換したものを格納）

```
unsigned char g_blsm_phase=0;
```

動作モードを示す変数（0:停止, 1:始動制御, 2:通常制御, 3:ブレーキ, 4:異常停止）

```
unsigned short g_blsm_start_rpm=600;
```

始動時の設定回転数

```
unsigned short g_blsm_start_rpm_upper, g_blsm_start_rpm_lower;
```

始動時に duty を減少させる回転数上限（回転数が g_blsm_start_rpm_upper 以上の時は duty を減らす）
始動時に duty を増加させる回転数下限（回転数が g_blsm_start_rpm_lower 以下の時は duty を増やす）

```
//ターゲット回転数
unsigned short g_blsm_target_rpm=2000;
//BLSM_TARGET_RPM_FIX を定義しているときはこの値が有効
```

目標回転数（ボリュームの角度によって設定）

```
unsigned short g_blsm_current_target_rpm;
```

現在の設定回転数（この値を、CMT0 のタイマ値に反映）

```
short g_blsm_target_direction=BLSM_DIRECTION_CCW;     //1:CCW, -1:CW
```

回転方向のターゲットを示す変数（値を変更する場合はモータ停止時に行う）

```
float g_blsm_duty_diff_integral=0;
```

duty の増減値（増分値、減少値の累積を保存し、10ms 毎に duty 値に反映させる）

```
unsigned short g_blsm_stable_threshold1=BLSM_DIRECTION_STABLE_THRESHOLD1;
unsigned short g_blsm_stable_threshold2=BLSM_DIRECTION_STABLE_THRESHOLD2;
unsigned short g_blsm_stable_threshold3=BLSM_DIRECTION_STABLE_THRESHOLD3;
```

回転方向安定を判断する閾値

```
unsigned short g_blsm_stable=0;
```

回転方向がターゲットと一致している際に、インクリメントされる変数

```
const float g_blsm_phase1_diff_array[8] = {0.001, 0.0015625, 0.003125, 0.00625, 0.0125, 0.025,
0.05, 0.1};
unsigned short g_blsm_phase1_diff_index=7;
```

始動制御時の duty の変化量を決める変数
最初は 10%変化させ、2 回目は 5%、3 回目は 2.5%と変化量を減らしていく事により、初期の収束を速める

```
unsigned char g_blsm_cmt1_init=1;
```

50us 毎の制御で初期化を行う事を示すフラグ変数
1 にセットしている場合、CMT1（50us 毎）割り込み時、各種変数を初期化する
モータを一旦停止させた場合再始動前に 1 を設定

```
unsigned char g_blsm_spi_flag=0;     //b0:send, b1:recv
```

SPI 送信時に、送信中であることを示すフラグ
（本プログラムでは、SPI の受信は使用していないので、送信時に 0x1 をセット、送信が完了すると、0x1 がクリアされる）

```
unsigned long g_blsm_da_send_data[8];
```

SPI で送信される、D/A コンバータに与える送信値

```
unsigned char g_blsm_led_state=BLSM_LED_OFF;
```

LED（マイコンボード D9）の消灯、点灯、点滅状態を指定する変数

```
const unsigned char g_blsm_led_blink2[20]={1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
const unsigned char g_blsm_led_blink3[20]={1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
const unsigned char g_blsm_led_blink4[20]={1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0};
```

LED の点滅のパターン（2 回点滅、3 回点滅、4 回点滅）を定義している変数

```
unsigned char g_blsm_z_phase_flag=0;
```

モータエンコーダの Z 相の信号入力時クリアされるフラグ変数

```
#ifdef BLSM_DEF_DEBUG_COUNTER
unsigned long g_blsm_cmt0_count=0;
unsigned long g_blsm_cmt1_count=0;
unsigned long g_blsm_cmt2_count=0;
unsigned long g_blsm_cmt3_count=0;
unsigned long g_blsm_cmtw0_count=0;
unsigned long g_blsm_spi_error_count=0;
unsigned long g_blsm_ad0_error_count=0;
```

```
unsigned long g_blsm_ad1_error_count=0;
#endif
```

各コンペアマッチタイマの呼び出し回数、エラー数をカウントする変数

・sci.c

```
unsigned char G_RxiFlag_1;
unsigned char G_TxiFlag_1;
unsigned char G_ErrFlag_1;
unsigned char G_RxiFlag_2;
unsigned char G_TxiFlag_2;
unsigned char G_ErrFlag_2;
```

割り込みフラグ（受信、送信完了、エラー）

```
unsigned char sci_buf_1[SCI_BUF_SIZE];
unsigned short sci_buf_write_p_1=0;
unsigned short sci_buf_read_p_1=0;
```

SCI1 側のバッファ、バッファの書き込みポインタ、読み出しポインタ

```
unsigned char sci_buf_2[SCI_BUF_SIZE];
unsigned short sci_buf_write_p_2=0;
unsigned short sci_buf_read_p_2=0;
```

SCI2 側のバッファ、バッファの書き込みポインタ、読み出しポインタ

4. プログラムの動作説明

4.1. メイン関数

初期化

・各種変数の初期化
・GPT, CMT 等の各種タイマの開始

を行った後、マイコンボードの SW(SW2)の押下により、

モータ回転→停止→ブレーキ→停止
START(1)→STOP(2)→BREAK(3)→STOP(0)

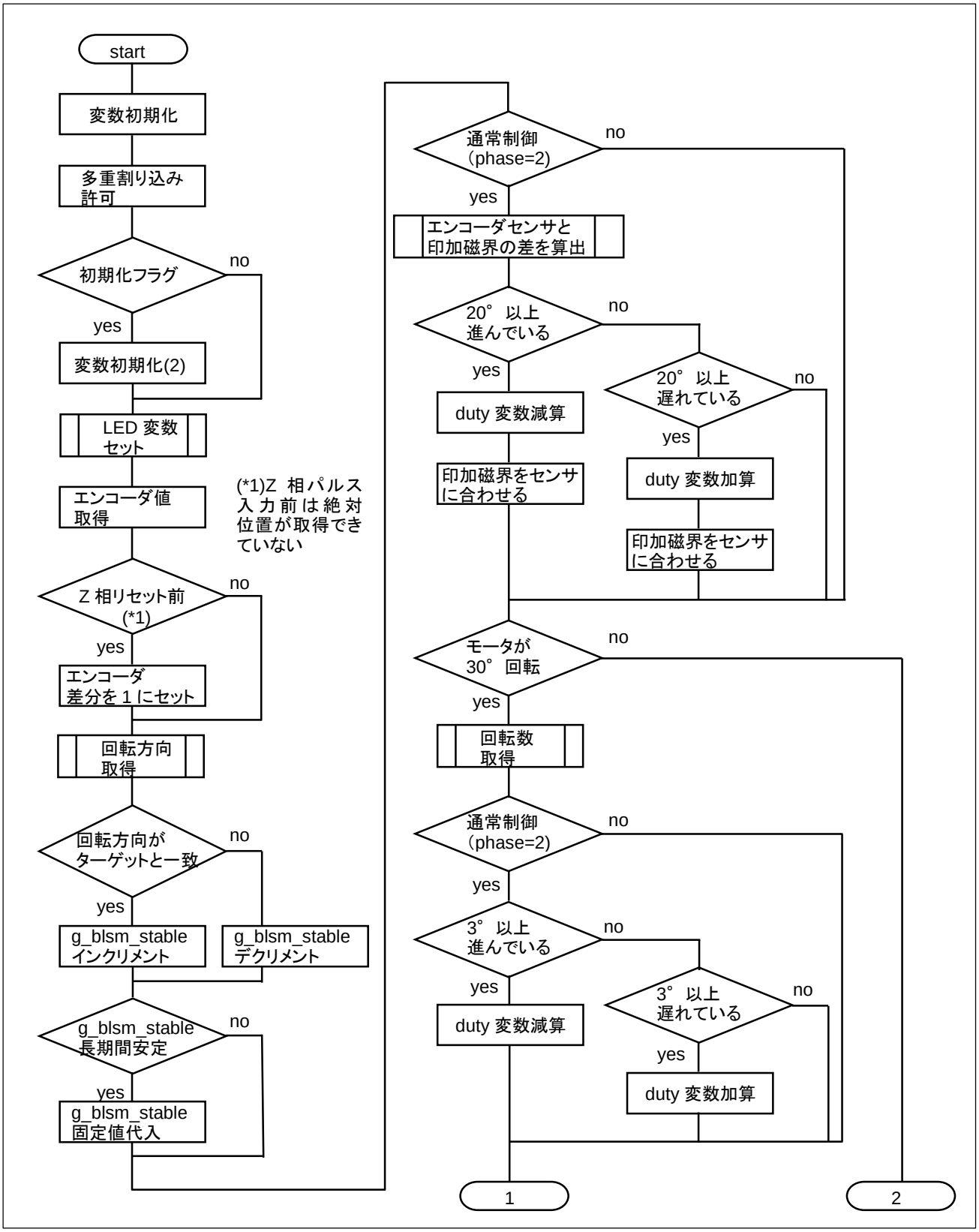
をトグルで切り替える動作を行います。

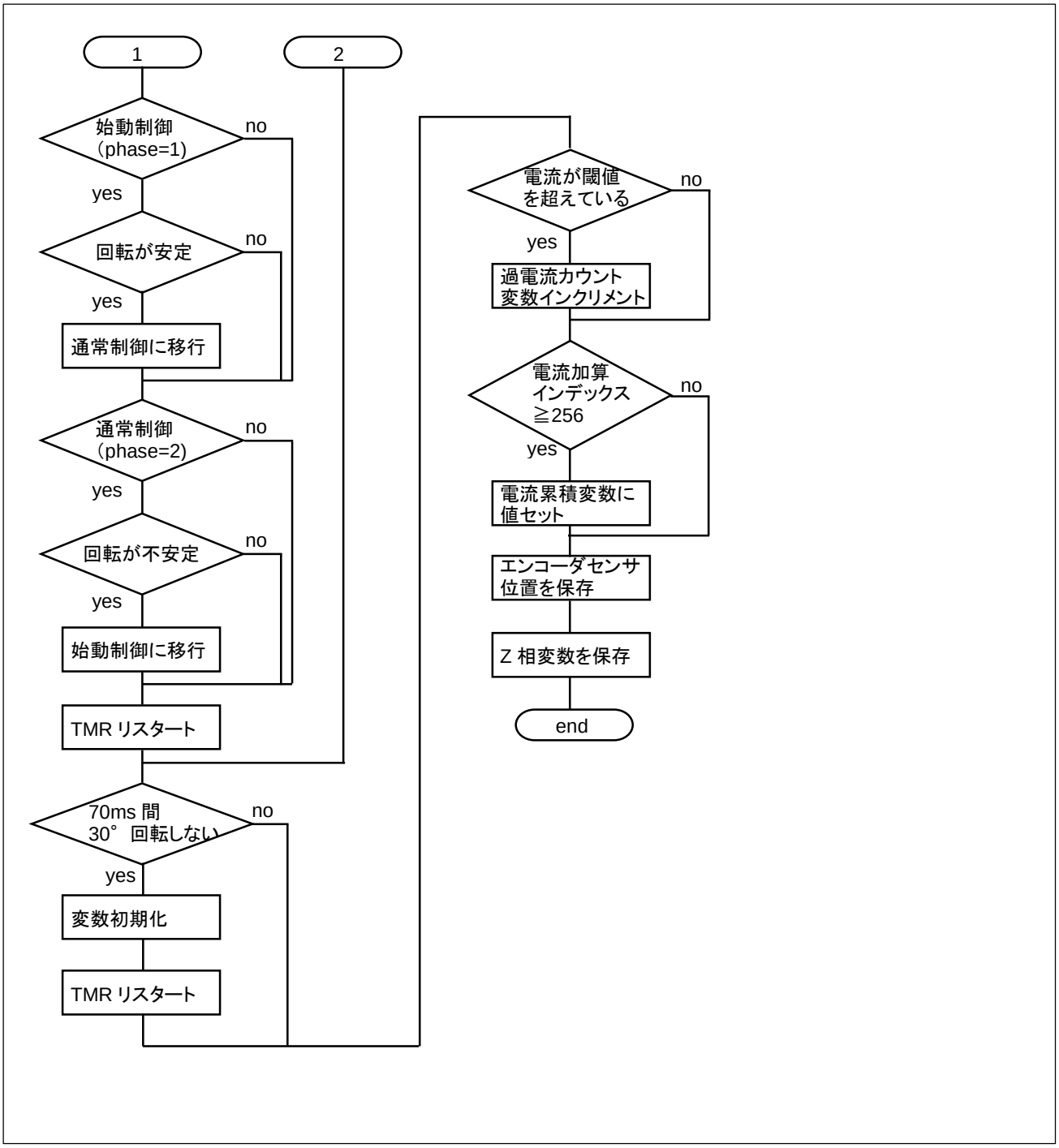
```
Copyright (C) 2018 HokutoDenshi. All Rights Reserved.
BLM_EV2, 24V Motor Test Program for RX71M(HSBRX71M100) [REV1].
[DRIVE SIDE]

START(1)   SW2 押下
STOP(2)    SW2 押下

BREAK(3)   SW2 押下
STOP(0)    SW2 押下
```

プログラム動作上の各種情報は、インタフェースボード（HSBRX71M100-MIF, J2 USB-miniB）から、シリアル(115,200bps)で出力されます。

4.2. モータ制御

(1)50us 毎の制御(Config_CMT1_user.c)





50us 毎に

・エンコーダ値の取得

・回転方向取得

・回転方向の安定を判断

・通常制御時センサーと印加磁界に 20° 以上の差がある場合印加磁界を補正（＋duty を補正）

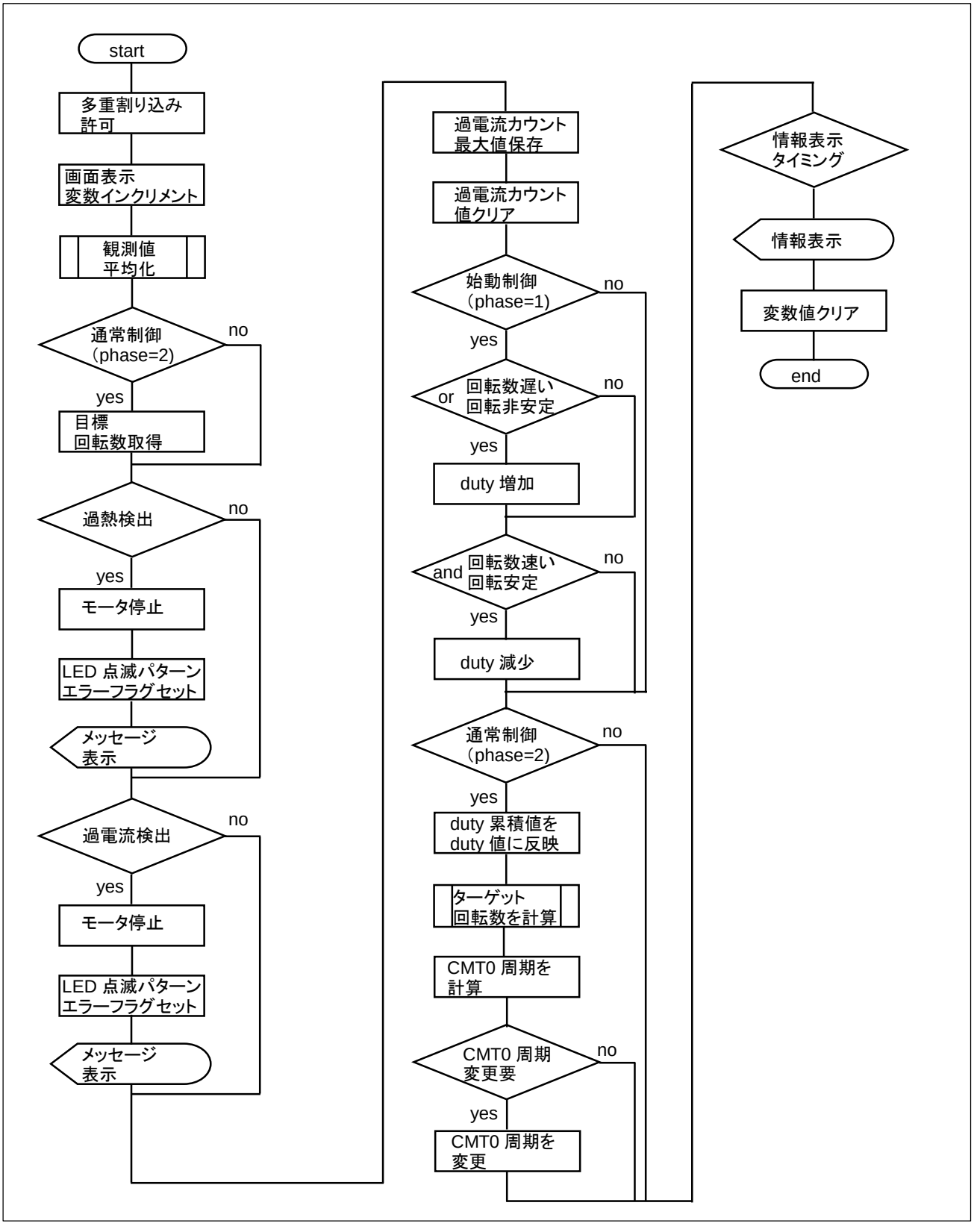
・30° の回転毎に

　・回転数取得

　・センサーと印加磁界に 3° 以上のずれがある場合 duty を補正

を行います。

(2)10ms 毎の制御(Config_CMTW0_user.c)



10ms 毎に
・回転数、温度、電流値の平均化（16 点）
・過熱停止の判断
・過電流停止の判断
・始動制御の場合
　・回転数・回転安定に応じた duty の増減
・通常制御の場合
　・50us 毎の制御で累積された duty 値の反映
　・印加磁界のターゲット回転数変更
・情報の画面表示
を行います。

(3)20us 毎の処理(Config_CMT2_user.c)

・A/D 変換値の履歴保存
・A/D 変換スタート
　を行います。A/D 変換値としては、20us 刻み(50kHz)となります。

(3)40us 毎の処理(Config_CMT2_user.c)

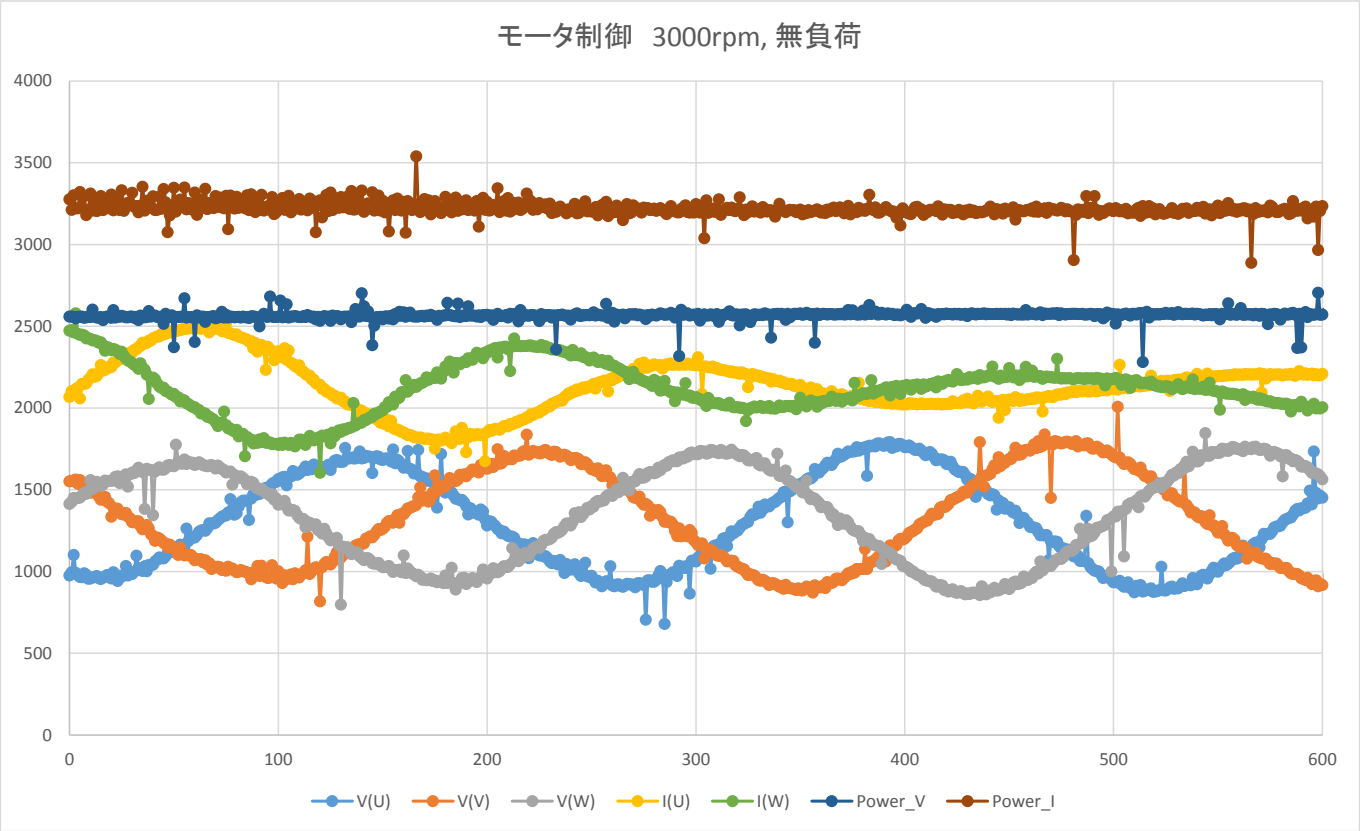
・D/A 出力向け電流値の計算
・D/A コンバータデータ送信
・SCI バッファ画面出力
・LED の点灯・点滅制御
を行います。
※CMT2 は割り込み優先度が低いため、他の処理が実行中は後回しにされる場合があります
（処理のリアルタイム性は確保されていません）

(4)磁界回転制御(Config_CMT0_user.c)

・印加磁界を 1° 進める処理
を行います。
※CMT0 はターゲット回転数に応じたタイミングに設定されます

5. A/D 変換値出力例

A/D 変換の出力値をプロットした例を示します。



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	2019.1.21	—	初版発行
REV.1.1.0.0	2019.1.28	—	ドキュメントタイトル変更

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

HSBRX71M100, HSBRX71M100-MIF, BLM_EV2 モータ制御ソフトウェア

RX71M 24V MOTOR
ソフトウェア資料

株式会社 北斗電子

 2019 年 1 月 21 日改訂 REV.1.0.0.0 (190121)