ブラシレスモータスタータキット(RL78G24)
[ソフトウェア　サンプルプログラム編]
取扱説明書

ルネサス エレクトロニクス社　RL78/G24(QFP-64 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Hokuto Electronic

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、アクセスランプ点灯中に電源の切断を行わないでください。

製品の故障や、データの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。

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 [本マニュアルで説明する内容]
・プログラムのバイナリ（MOT ファイル）
　BIN 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. サンプルプログラム

1.1. プログラム仕様・動作

サンプルプログラムは、TUTORIAL_BC（相補 PWM 駆動+センサレス）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。

参照プロジェクト：RL78G24_BLMKIT_SAMPLE
参照プロジェクト：RL78G24_COM_BLMKIT_SAMPLE （COM ポートデバッグ版）

本プログラムは、以下の仕様を実装しています。

・ホールセンサを使用/センサレス切り替え
（センサレスの場合、センサパターンを生成する電圧値の平均化・ヒステリシスの有効・無効をコマンドで切り換え）
・VR にて回転数を変更
・CH-1 側は相補 PWM で駆動（PWM は 6 相）
・CH-2 側は 120 度制御で駆動（PWM は 1 相）
・SW1 回転・停止を制御(CH-1)
・SW2 回転・停止を制御(CH-2)
・SW3 ホールセンサを使用/センサレス切り替え
・過電流保護停止機能(*1)
・過熱停止機能（デフォルトでは、50℃で停止）
・進角調整機能（シリアル端末を接続し、キーボードの qwe で変更）（CH-1 側のみ）

(*1)
・1 回でも 8A(peak)を超えると停止（デフォルト無効化） ※コマンドで有効化切り替え
・10ms 間に 360 回以上過電流が検出された場合停止（デフォルト有効化）
・1s 間に 1000 回以上過電流が検出された場合停止（デフォルト有効化）

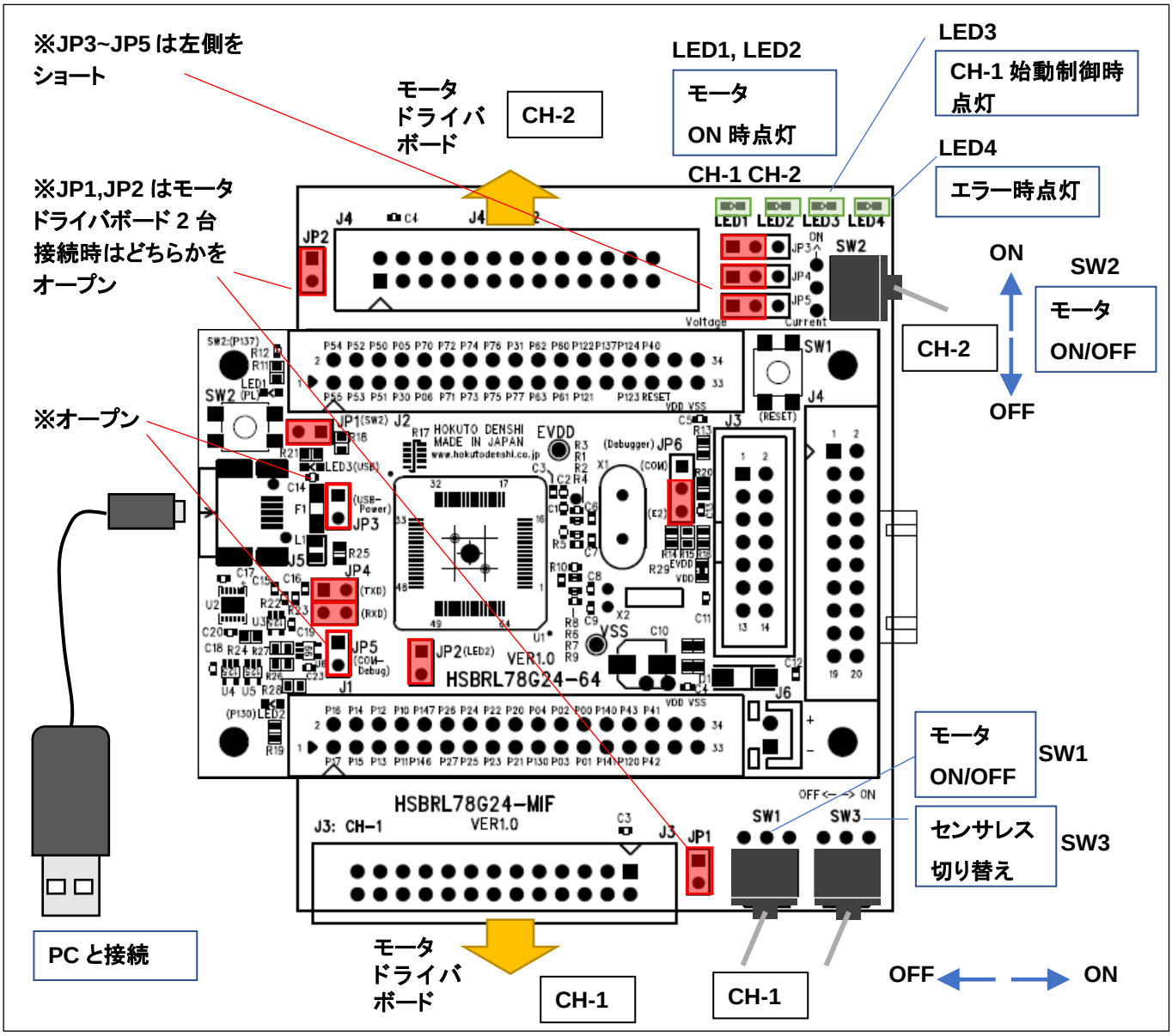
	OFF	ON
SW1	モータ停止(CH-1)	モータ運転(CH-1)
SW2	モータ停止(CH-2)	モータ運転(CH-2)
SW3	ホールセンサ使用	ホールセンサ未使用

	最低	最大
VR	1,200[rpm](*1) 1,800[rpm]	12,000[rpm]

(*1)CH-1 は、相補 PWM で駆動され、最低回転数は 1200[rpm]
CH-2 は、120 度制御で駆動され、最低回転数は 1800[rpm]
チュートリアルでは VR で duty を直接制御していましたが、本サンプルプログラムでは VR は回転数を制御します。

－モータドライバボードの設定等－

・LED と SW の役割，ジャンパ設定（COM ポートデバッグを使用しない場合）

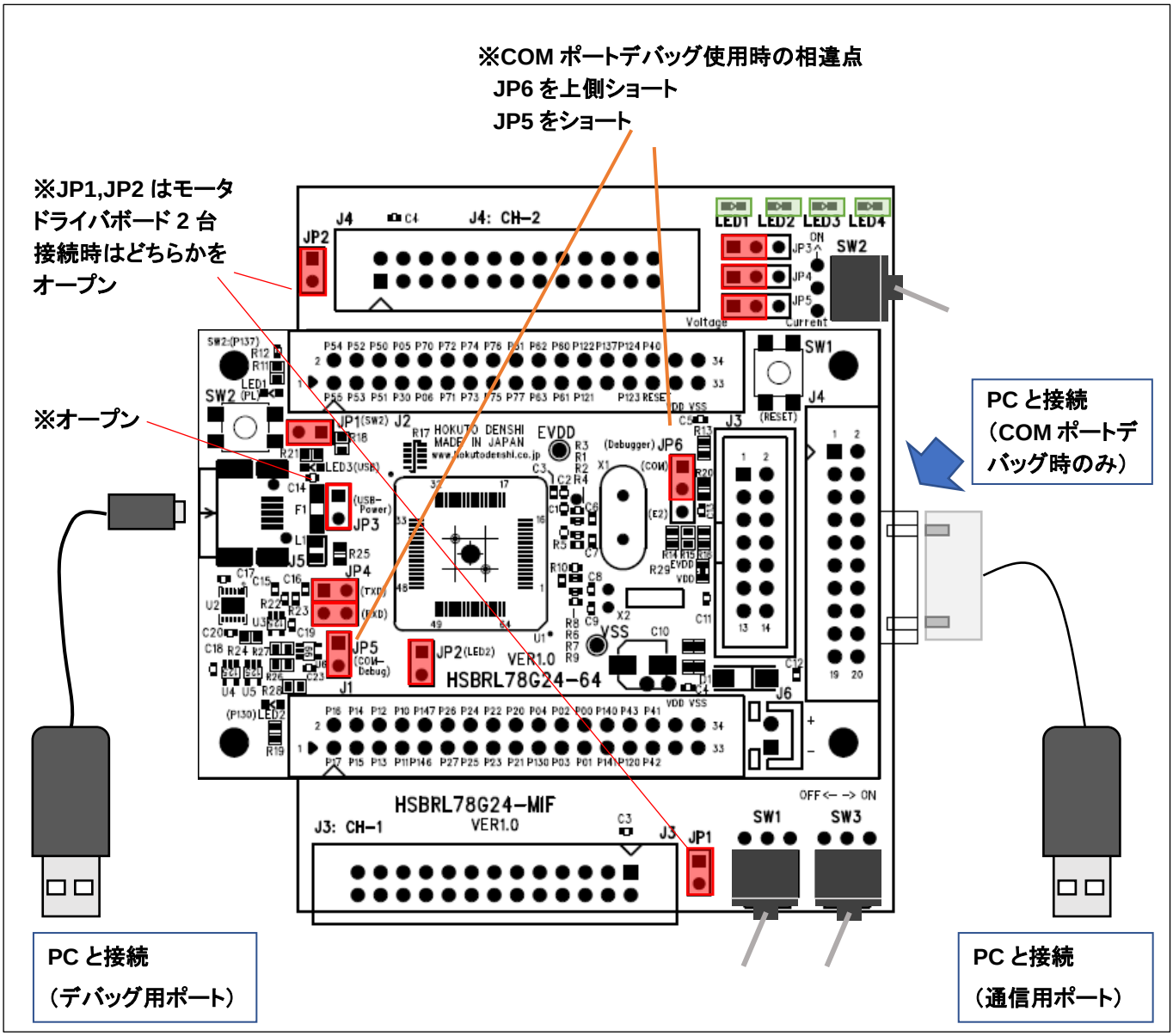


※シリアルポートの接続は、モータを動かす上で必須ではありませんが、回転数や温度のモニタ、外部からコマンド入力を行う場合に必要です。

・LED の情報

	消灯	点灯
LED1	CH-1 モータ停止	CH-1 モータ運転
LED2	CH-2 モータ停止	CH-2 モータ運転
LED3	始動制御時以外(CH-1)	始動制御中(CH-1)
LED4	エラーなし	エラーあり

・COM ポートデバッグ時のジャンパ設定



COM ポートデバッグ使用時はマイコンボード上の JP5 と JP6 の変更が必要です。マイコンボード上の USB ポートはデバッグに使用されますので、通信を行いたい場合は接続ボード上の J5（5P コネクタ）に、USB-Serial 変換機器（当社製品の場合は、USB-1S(JST)を接続してください。）

ー起動時のシリアル端末の表示ー

```
Copyright (C) 2025 HokutoDenshi. All Rights Reserved.

RL78/G24 / BLUSHLESS MOTOR STARTERKIT SAMPLE


EXPLANATION:
SW1 -> CH-1 motor ON/OFF
SW2 -> CH-2 motor ON/OFF
SW3 -> OFF:Hall sensor use, ON:Pseudo hall sensor pattern use
LED1 : CH-1 active ON/OFF
LED2 : CH-2 active ON/OFF
LED3 : Start up ON, Normal OFF
LED4 : Error status
VR -> duty(0-100%)

COMMAND:
s : stop <-> start display information(toggle)
A : A/D convert data display
D : rotation direction->CCW <-> roration direction->CW (toggle)
a : pseudu hall sensor pattern <-> average volatage(toggle)
h : pseudu hall sensor pattern <-> hysterisis volatage(toggle)
B : BRAKE
q : forward angle -1    [CH-1]
w : forward angle +1    [CH-1]
e : forward angle =0    [CH-1]
f : FAA function <-> CPU function (toggle)
1 : UVW calc -> sine
2 : UVW calc -> sine(2)
3 : UVW calc -> sine + 3harmonic
4 : UVW calc -> sine + 3harmonic(2)
5 : UVW calc -> another version
6 : UVW calc -> another version(100% power)
7 : UVW calc -> another version(100% power)(2)
C : Over current interrupt(ONCE) stop VAILD<->INVALID(toggle)
z : debug display(LEVEL1)(toggle)
x : debug display(LEVEL2)(toggle)
c : debug display(LEVEL3)(toggle)
v : debug display(LEVEL4)(toggle)
b : debug display(LEVEL5)(toggle)

>
Motor driver board connection check...
   CH-1 Connected.
   CH-2 NOT connected?(Motor hall sensor not detected.)
```

115,200bps
の設定で端末を開いてください

・キーボードから入力可能なコマンド

コマンド	設定内容	備考
s	5 秒毎の画面表示の停止	トグル動作（停止←→再開）
A	A/D 変換値の表示（直近の 100us 毎のサンプリング値 10 ポイント）	モータが動作している場合
D	回転方向の切り替え	トグル動作（CCW←→CW）
a	疑似ホールセンサパターン使用時、電圧の平均値を用いる	トグル動作(ON/OFF)
h	疑似ホールセンサパターン使用時、ヒステリシスを有効化	トグル動作(ON/OFF)
B	ブレーキ	モータが動作している場合
q	進角調整-1°[CH-1]	調整可能範囲は-45°まで
w	進角調整+1°[CH-1]	調整可能範囲は+45°まで
e	進角調整値を 0°にリセット[CH-1]	
f	相補 PWM の UVW 分解に FAA を使用・未使用の切り替え	トグル動作（使用←→未使用）
1	UVW の分解アルゴリズムを正弦波変換に設定	
2	UVW の分解アルゴリズムを正弦波変換(2)に設定	正規化後に duty を乗ずる
3	UVW の分解アルゴリズムを正弦波 3 倍高調波重畳に設定	
4	UVW の分解アルゴリズムを正弦波 3 倍高調波重畳(2)に設定	正規化後に duty を乗ずる
5	UVW の分解アルゴリズムを別バージョン 1 に設定	全角度 86.6%のパワー
6	UVW の分解アルゴリズムを別バージョン 2 に設定	30°単位の角度に 100%のパワー
7	UVW の分解アルゴリズムを別バージョン 2'に設定	6 の直線近似版
C	1 回の過電流検出でモータを停止	トグル動作（有効←→無効）
z	デバッグ表示(LEVEL1)ON	トグル動作(ON/OFF)
x	デバッグ表示(LEVEL2)ON	トグル動作(ON/OFF)
c	デバッグ表示(LEVEL3)ON	トグル動作(ON/OFF)
v	デバッグ表示(LEVEL4)ON	トグル動作(ON/OFF)
b	デバッグ表示(LEVEL5)ON	トグル動作(ON/OFF)

・デバッグ表示内容

	表示内容	備考
LEVEL1	duty の増加・減少	
LEVEL2	ホールセンサが切り替わった際の磁界印加角度	CH-1 のみ
LEVEL3	回転数と 10ms 毎の duty 増分値	
LEVEL4	2ms 毎のホールセンサ,疑似ホールセンサ値	
LEVEL5	UVW の相電圧と相電圧の平均値	A/D 変換値

・情報表示（5 秒に 1 回） SW1 を ON にしモータを回転させた状態

```
                                CH-1          CH-2
Motor Driver Board             : Connect      NoConnect
Active                         : o            x
hall sensor                    :   Pseudu hall sensor pattern, phase voltage
   average      ->    ON
   hysteresis ->    ON
rotation control(PHASE)        : Normal(2)    Stop(0)
UVW calculation method         : (1) [FAA]
target speed([rpm])            :   4959       12000
current target speed([rpm]) :   4959          0
forward angle([deg])           :       0          -
target direction               :    CCW       STOP
rotation speed([rpm])          :   4545          0
rotation speed[ave]([rpm])   :   4780          0
Temperature(A/D value)         :   2155       3697
Temperature(degree)            :      28        101
VR(A/D value)                  :   1285       3784
duty[%]                        :   41.9       0.0
Debug print level              : 1(x) 2(x) 3(x) 4(x) 5(x)
```

各種情報を表示

・画面表示内容

項目	表示例	内容	備考
Mortor Driver Board	Connect NoConnect	モータドライバボード接続有無	起動時にモータドライバボードが接続されているかをチェックしてその結果を表示
Active	o x	SW が ON で回転制御対象の時は o	
hall sensor	Motor hall sensor Pseudu hall sensor pattern, phase voltage	モータ内蔵のホールセンサを使用するか、相電圧で回転を検出するか（センサレス駆動）	SW3 で切り換え
average	ON OFF	相電圧の平均値を使用するか否か	センサレス駆動時のみ表示 a コマンドで ON/OFF
hysteresis	ON OFF	相電圧のヒステリシスを有効化するか否か	センサレス駆動時のみ表示 h コマンドで ON/OFF
rotation control(PHASE)	Stop(0) Startup(1) Normal(2) Brake(3)	モータの制御フェーズ 停止(0) 始動制御(1) 通常制御(2) ブレーキ(3)	ブレーキは相補 PWM 駆動の CH-1 側のみ有効 （CH-2 は停止とブレーキは同じ動作）
UVW calculation method	(1) ~ (7)	UVW 分解アルゴリズム (1)~(7)の 7 通り	コマンド 1~7 で切り換え （CH-1 のみ有効）
target speed([rpm])	4959	VR ツマミで設定した目標回転数([rpm])	
current target speed([rpm])	4959	現在の制御回転数[rpm]	
forward angle([deg])	0	進角調整値[°]	（CH-1 のみ）
target direction	CCW CW BRAKE STOP	回転方向	
rotation speed([rpm])	4545	現在の回転数[rpm]	
rotation speed(ave)([rpm])	4780	現在の平均回転数	直近の 10ms 毎 16 値の平均
Temperature(A/D value)	2155	温度センサーA/D 変換値	0~4095
Temperature(degree)	28	温度センサー摂氏温度[℃]	
VR(A/D value)	1285	VR の A/D 変換値	0~4095
duty[%]	41.9	現在の duty 比[%]	
Debug print level	1(o) 2(x) 3(x) 4(x) 5(x)	LEVEL1 が有効 LEVEL2~LEVEL5 が無効の場合	コマンド zxcvb で有効・無効切り替え

・進角調整の例（CH-1 側のみ）

シリアル端末にキーボードから進角調整を行う事が可能です。（キーボードの qwe を使用）

	進角調整 （遅れ方向） -1°	進角調整 （進み方向） +1°	リセット(=0)
CH-1	q	w	e

キーボードの w を 5 回押すと、

```
                                    CH-1          CH-2
Motor Driver Board                : Connect       NoConnect
Active                            : o             x
hall sensor                       :   Pseudu hall sensor pattern, phase voltage
   average      ->   ON
   hysteresis ->   ON
rotation control(PHASE)         : Normal(2)   Stop(0)
UVW calculation method            : (1) [FAA]
target speed([rpm])               :   4572          11833
current target speed([rpm]) :   4569              0
forward angle([deg])              :        0              -
target direction                  :     CCW          STOP
rotation speed([rpm])             :   4761              0
rotation speed[ave]([rpm])    :   4696              0
Temperature(A/D value)            :   2184          3691
Temperature(degree)               :       29            113
VR(A/D value)                     :   1154          3807
duty[%]                           :   41.0          0.0
Debug print level                 : 1(x) 2(x) 3(x) 4(x) 5(x)
```

```
[CH-1] forward angle -> 1

[CH-1] forward angle -> 2

[CH-1] forward angle -> 3

[CH-1] forward angle -> 4

[CH-1] forward angle -> 5

                                    CH-1          CH-2
Motor Driver Board                : Connect       NoConnect
Active                            : o             x
hall sensor                       :   Pseudu hall sensor pattern, phase voltage
   average      ->   ON
   hysteresis ->   ON
rotation control(PHASE)         : Normal(2)   Stop(0)
UVW calculation method            : (1) [FAA]
target speed([rpm])               :   4506          12000
current target speed([rpm]) :   4596              0
forward angle([deg])              :        5              -
target direction                  :     CCW          STOP
rotation speed([rpm])             :   4761              0
rotation speed[ave]([rpm])    :   4694              0
Temperature(A/D value)            :   2187          3677
Temperature(degree)               :       29            101
VR(A/D value)                     :   1153          3747
```

進角調整値(forward angle)の値が変化し、進角調整が掛かった状態となります。
※高速回転時は、進角を利かせる（+の値，速めに電流方向を切り替える）と有利です

1.2. モータ制御アルゴリズム

TUTORIAL_BC「相補 PWM 信号での駆動＋センサレス駆動」をベースとしています。

※「チュートリアル編」のマニュアルも併せて参照ください

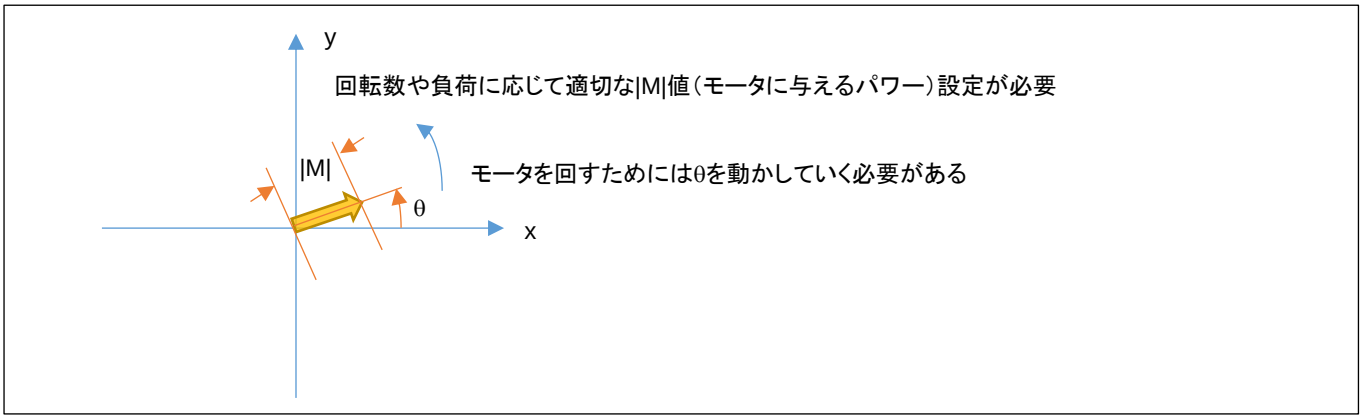
TUTORIAL_BC のプログラムに、
・回転数(duty)制御
・始動制御
を追加したものが、本サンプルプログラムとなります。

CH-1 側の基本的な制御は相補 PWM なので、プログラムとしてはその時々の

・ベクトルの大きさ|M| （duty 比…g_duty 変数値）
・ベクトルの角度θ （磁界印加角度…g_angle 変数値）

を決めて設定するという動作です。

（|M|とθの値は、実際には UVW 3 値の PWM 値に変換されて、タイマが駆動されます。しかし、その部分は機械的な処理を行うだけですので、プログラムの肝は「いま、このタイミングでの|M|,θ値を決定」する事となります。）



CH-2 側は、電流の印加方向はホールセンサパターンで決定します。制御対象となるのは duty 値のみです。

1.2.1. UVW 変換アルゴリズム[CH-1 側]

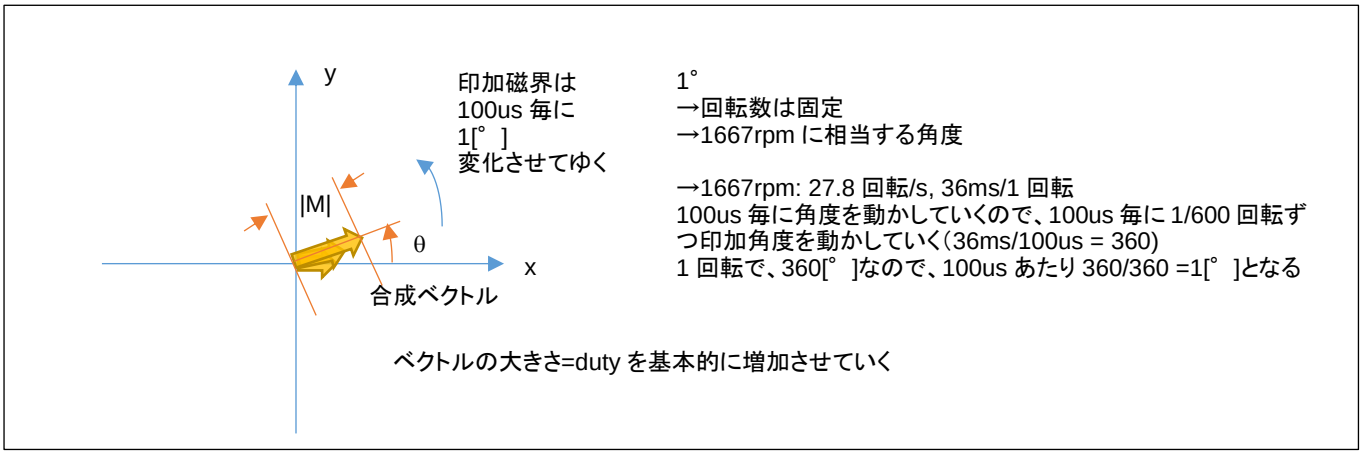
|M|,θ値を U, V, W の各相の duty 値に変換するアルゴリズムのデフォルトは正弦波駆動です
（この変換の話は「ソフトウェア　チュートリアル編」のマニュアル 2.2 節を参照ください。）
本サンプルプログラムでは、7 種類の変換アルゴリズムを用意しています。

```
blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin;                  //(1)正弦波駆動（デフォルト）
//blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin_post;           //(2)正弦波駆動，duty を後から乗算
//blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin_3harmonic;      //(3)正弦波 3 倍高調波重畳
//blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin_3harmonic_post; //(4)正弦波 3 倍高調波重畳，duty を後から乗算
//blm_angle_to_uvw_duty = blm_angle_to_uvw_duty1;                   //(5)別バージョン 1，全方向に 86.6%のパワー
//blm_angle_to_uvw_duty = blm_angle_to_uvw_duty2;                   //(6)別バージョン 2，60°で割り切れる角度では 100%のパワー
//blm_angle_to_uvw_duty = blm_angle_to_uvw_duty2x;                  //(7)別バージョン 2' 別バージョン 2 を直線近似したもの
```

キーボードからのコマンド 1~7 で(1)~(7)の変換アルゴリズムの切り替えが可能です。

1.2.2. 始動制御

(1)CH-1 側の始動制御
フェーズ 1（g_phase[] = BLM_PHASE_1(=1)　のとき）の制御、SW を ON にして、モータを停止状態から立ち上げる際の動作です。



・印加磁界の角度(θ)は 1667rpm に相当する角度（1[°]）を 100us 毎に）動かしていく
・duty(|M|)は 10ms 毎に調整（条件により増加または減少）

となるような動作です。

始動制御の方式は、TUTORIAL_B で行っている事と同様のものとなります。

TUTORIAL_B では、
・回転数（＝印加磁界の切り替わりタイミング）は一定
・duty は VR に連動して変わる
という方式でした。

サンプルプログラムの、始動制御は
・回転数（＝印加磁界の切り替わりタイミング）は一定
・duty は回転が安定するまで duty 値を増やしていく（場合によっては減らす）
という方式です。TUTORIAL_B で VR ツマミを回して手動で変えていた duty をプログラムで増加させているだけで、行っている事は同一です。

duty は最初は 0 ですが、徐々に増やしていきます。

このとき、
・回転安定回数
→ホールセンサ切り替わり時、
目標の回転方向とあっていればカウンタをインクリメント
目標の回転方向と逆であればカウンタをデクリメント
・回転数（速度）
の 2 項目をモニタリングします。

始動制御中に、

回転数が BLM_START_RPM_LOWER(=800rpm)未満
または、回転安定回数が BLM_DIRECTION_STABLE_THRESHOLD1(=6)以下であれば duty 増加

回転数が、BLM_START_RPM_UPPER(=1500rpm)以上
かつ、回転安定回数が BLM_DIRECTION_STABLE_THRESHOLD1(=6)以上であれば duty 減少
（手動で調整する際の「ツマミを回しすぎたので戻す作業」のイメージ）

の様に duty を調整します。

また、

回転数が BLM_START_RPM_LOWER(=800rpm)以上
かつ、回転安定回数が BLM_DIRECTION_STABLE_THRESHOLD2(=24)以上

であれば、始動制御の状態を抜けて通常制御に移行します。

回転安定回数は、最低回転数(BLM_START_RPM_LOWER=800[rpm])に達した後、

・目標回転方向と実際の回転方向が一致

の際に、カウンタ値が加算され、カウンタ値が BLM_DIRECTION_STABLE_THRESHOLD2(=24)（1200rpm で 4 回転、0.2 秒に相当）以上であれば、phase=2（通常制御）に移行させます。

duty を 0 から増加させていくと、duty の小さな領域では軸が振動するだけで、モータは回転しない。duty が増えてくると振動の振幅が大きくなり正回転（目標とする回転方向）と逆回転を短時間で繰り返す。duty をさらに大きくすると、安定して回転するようになる。duty を大きくしすぎると回転数が上がりすぎるので、duty は戻す（減少させる）という動作です。

blm.h 内の

```
//始動回転数
#define BLM_START_RPM 1200
#define BLM_START_RPM_LOWER 800  //始動時の下限
#define BLM_START_RPM_UPPER 1500  //始動時の上限
#define BLM_START_RPM_OVER 4000  //始動時の回転数異常と判断

//回転安定
#define BLM_DIRECTION_STABLE_THRESHOLD_1 6  //1回転分
#define BLM_DIRECTION_STABLE_THRESHOLD_2 24  //1200rpmの場合0.2s以上回転安定時PHASE2に移行
#define BLM_DIRECTION_STABLE_THRESHOLD_3 30  //回転安定検出の最大値
```

上記で、始動時の回転数や回転安定とみなす回数等を定義しています。

始動時の duty ですが、最初は大きく値を変更、徐々に変更する値を小さくしていく様にしています。

blm.c 内

```
//始動時のPHASE1でのduty変化量
//                                          0.2%  0.2%  0.25% 0.4% 0.8% 1.6%  3.2%  6.4%
const unsigned short g_phase1_diff_array[8] = {131,  131,  163,  262, 524, 1048, 2097, 4194};
//%値にdutyの取り扱い倍率である65536を掛けた値
//                                           0     1     2     3    4    5     6     7
volatile unsigned short g_phase1_diff_index[BLM_CH_NUM] = {7};//6.4%から変化量を調整
```

最初は、0%→7.5%に増加させます。次は、6.4%→9.6%→11.2%の様に徐々に増分値を減らしていきます。変化量は、blm.c 内に定義しています。この値を小さくすると始動に時間が掛かる様になり、大きくし過ぎると始動時に発振の様な挙動（duty を増やし過ぎて戻る）を示します。

(2)CH-2 側の始動制御

電流の印加方向は、6ms のインターバルタイマを使用して、6ms 毎に変化させます。6ms→1776rpm です。

duty は(1)の CH-1 同じ方法で制御します。徐々に増やしていき、回転安定が得られた時点で始動制御を抜ける動作となります。

1.2.3. 通常制御

(1)CH-1 側の通常制御

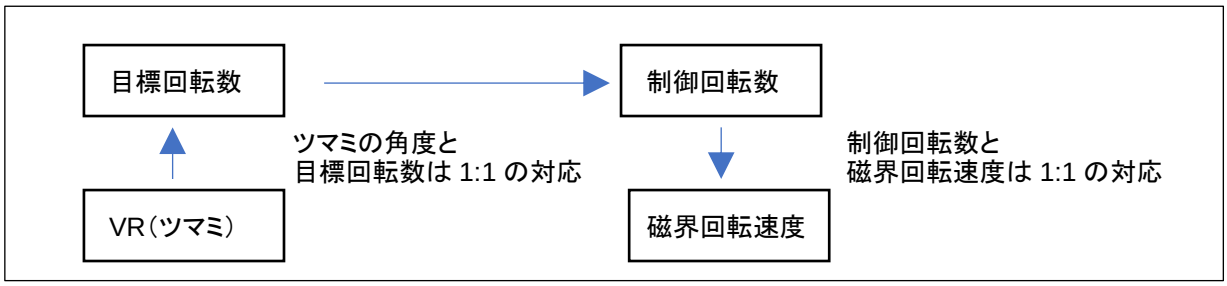
フェーズ 2（g_phase[] = BLM_PHASE_2(=2)　のとき）の制御です。

通常制御においても、印加磁界角度(θ)と duty(|M|)をどういう値とするのか、という話となります。

TUTORIAL_B2 では、VR ツマミで duty を変化させていたので、duty はツマミの角度によって一意に決まるという方式です。本サンプルプログラムでは、VR ツマミと対応するのは、目標回転数です。

目標回転数と印加磁界角度を 1:1 で対応させた場合、目標回転数を短時間で変更した場合、回転と印加磁界角度の整合が取れなくなります。そのため、本サンプルプログラムでは、制御回転数という概念（変数）を用意しています。

目標回転数：ユーザが VR ツマミで設定した回転数
制御回転数：プログラムで決める「今現在の設定」の回転数



最終的には、制御回転数＝目標回転数としますが、目標回転数は急激に変化することもあるので、プログラム上の設定回転数（＝制御回転数）は別に値を設けています。

モータ動作は制御回転数に制御され、制御回転数と目標回転数のすり合わせは別に行うという方式です。

1.2.3.1. 印加磁界角度の設定[CH-1 側]

・制御回転数(g_current_target_rpm)から、100us 毎の角度増分値(g_angle_diff)を算出（10ms 毎）

・角度増分値を印加磁界角度(g_angle)に加算（100us 毎）

・ホールセンサから導出される、理想印加角度に補正（ホールセンサ切り替わり時）

印加磁界角度に関しては、100us 毎に制御回転数に対応する分だけ動かしていき、ホールセンサ切り替わりのタイミングでリセット（ホールセンサ位置に応じた角度に設定）しています。

実際の回転数と制御回転数が合っていない場合でも、ホールセンサ切り替わりのタイミングで印加磁界角度はリセットされます。（ずれが累積する事がない様にしています）

g_angle 他角度は、1°単位の取り扱いだと分解能が荒すぎるので、1°を 64 で表す事としています。
（1°単位とすると、100us 毎に 1°加算すると、1667[rpm]。2°加算すると、3334[rpm]となり、その間の速度が表現できない。角度は、×64 の数値として取り扱い、100us 毎に 64 加算すると、1667[rpm]。128 加算すると、3334[rpm]。）

blm.h 内

```
//角度を取り扱う倍率
#define BLM_ANGLE_MULTIPLIER 64
#define BLM_ANGLE_MULTIPLIER_SHIFT 6  //2^6=64
```

cos/sin 等は°単位で扱っているので、cos(angle)を求める場合は、6bit シフトさせて°単位にしてから計算しています。

1.2.3.2. duty の設定[CH-1 側]

・目標回転数と制御回転数のずれが 5%以内

→PI 制御を行う（duty の増分積算値を一定時間(10ms)毎に duty に反映させる）

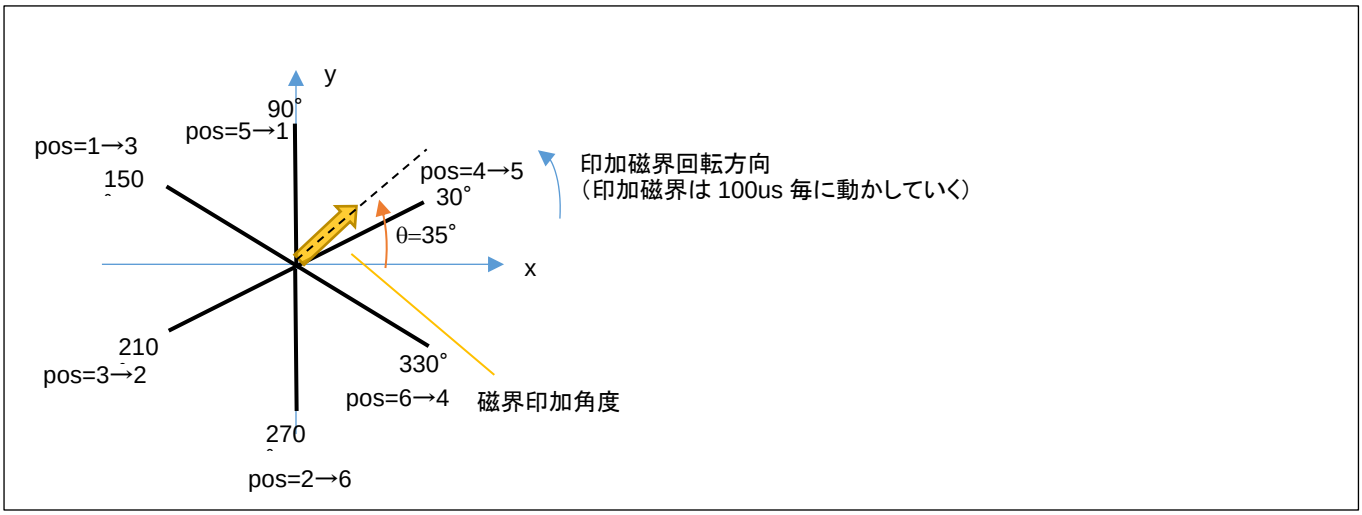
ホールセンサが切り替わったタイミングで、理想的な印加磁界角度を算出。

理想的な印加磁界角度に対し現在の印加磁界角度が BLM_ANGLE_DIFF_THRESHOLD2(=30)度以上大きい
→duty 増分積算値から BLM_DUTY_DIFF2(0.2%)を減算（ホールセンサ切り替わり毎）

理想的な印加磁界角度に対し現在の印加磁界角度が BLM_ANGLE_DIFF_THRESHOLD2(=30)度以上小さい
→duty 増分積算値に BLM_DUTY_DIFF2(0.2%)を加算（ホールセンサ切り替わり毎）

理想的な印加磁界角度に対し現在の印加磁界角度が BLM_ANGLE_DIFF_THRESHOLD1(=15)度以上大きい
→duty 増分積算値から BLM_DUTY_DIFF1(0.05%)を減算（ホールセンサ切り替わり毎）

理想的な印加磁界角度に対し現在の印加磁界角度が BLM_ANGLE_DIFF_THRESHOLD1(=15)度以上小さい
→duty 増分積算値に BLM_DUTY_DIFF1(0.05%)を加算（ホールセンサ切り替わり毎）



例えば、ホールセンサの読み取り値が 4→5 に変わった際、印加磁界角度が 35° だとすると、ホールセンサの切り替わりタイミングが遅い（実際のモータの回転数が印加磁界角度より遅れている）事となります。

ここで、印加磁界角度に対し、ホールセンサ切り替わりが BLM_ANGLE_DIFF_THRESHOLD1(=15)度以上遅れている場合は、モータの回転数を上げたいので duty 増分積算値の値に BLM_DUTY_DIFF1(=0.05%)を加算します。

逆に、印加磁界角度に対し、ホールセンサ切り替わりが BLM_ANGLE_DIFF_THRESHOLD1(=15)度以上進んでいる場合は、duty 増分積算値の値から BLM_DUTY_DIFF1(=0.05%)を減算します（回転数を落とす方向）。

印加磁界角度とホールセンサのずれがない場合は、duty が適切だとして、duty を変更しません。

印加磁界角度に対し、ホールセンサ切り替わりタイミングがもっと大きくずれている（BLM_ANGLE_DIFF_THRESHOLD2(=30)度以上）の場合は、duty 増分積算値をより大きく変化（BLM_DUTY_DIFF2(=0.2%)）させます。

その後、印加磁界角度（図の黄色矢印）を、図では 30° （理想値）に変更します。ホールセンサ切り替わりのタイミングで、常にホールセンサ（＝モータの軸位置）に応じた印加磁界角度に調整します。（この動作は duty 値の設定ではなく、印加磁界角度の設定の話です。）

上記で求めた duty 増分積算値(g_duty_diff_integral)を duty 値(g_duty)に加算（10ms 毎）。

blm.h 内の

```
#define BLM_DUTY_DIFF_1 32 //0.05%(x65536), ANGLE_DIFF_THRESHOLD_1 のずれを検出した際の増分
#define BLM_DUTY_DIFF_2 131 //0.2%(x65536), ANGLE_DIFF_THRESHOLD_2 のずれを検出した際の増分
```

```
//角度のずれ
#define BLM_ANGLE_DIFF_THRESHOLD_1 DEGREE_15 //15度以上ずれている場合は、dutyの微調整を行う
#define BLM_ANGLE_DIFF_THRESHOLD_2 DEGREE_30 //30 度以上ずれている場合は、duty の調整を行う
```

上記で、角度のずれや角度ずれがある場合の duty 比の増減値を決めています。

・目標回転数と制御回転数のずれが 5%以上

→ずれが大きいので、duty を大きく調整する

・目標回転数と現在の回転数の割合×フィードバック係数(BLM_DUTY_FEEDBACK_RATE)を duty に乗算(10ms 毎)（比例制御）

blm.h 内

```
#define BLM_DUTY_FEEDBACK_RATE 0.01f //1%
```

※duty 値の取り扱いに関して

チュートリアルでは duty 値は、0-100%を 0-256 の数値で取り扱っていました。この場合の分解能は、0.4%程度となります。

それに対し、サンプルプログラムでは、duty を+0.05%するなどもっと細かい分解能で取り扱いたいので、0-100%を 0-65536 の値で取り扱っています。

duty を 0-65536 の数値で取り扱う事で、duty を伴う計算は long 型(32bit)での演算が主となります。（一部の計算は、16bit の範囲で収まる様に、ビットシフトした後で乗算するケースもあります。）

サンプルプログラム内では、duty を+0.05%したい場合は、

g_duty[i] += BLM_DUTY_DIFF_1;　　// BLM_DUTY_DIFF_1=32

の様になります。

1.2.3.3. 制御回転数の設定[CH-1 側]

・目標回転数と制御回転数のずれが 5%以内

→制御回転数 = 目標回転数に設定

※ほぼずれがないので、制御回転数を最終的な回転数の目標である目標回転数に設定してしまう

・目標回転数と制御回転数のずれが 5%以上

・目標回転数と現在の回転数の割合×フィードバック係数(BLM_RPM_FEEDBACK_RATE)を制御回転数に乗算(10ms 毎)

blm.h 内

```
//回転数フィードバック
#define BLM_RPM_FEEDBACK_RATE 0.20f //20%
```

※徐々に制御回転数を目標回転数に近づけていく動作

制御回転数（＝印加磁界を進める速度）は、

目標回転数（例えば、10,000rpm）、回転数（平均）（5,000rpm）の場合、（差が 5%以上のケース）

目標に対しての現状値の割合： rate = 1-(1000/5000) = 0.5
5000 × (rate × 0.2 +1.0) = 5500rpm
（0.2 はフィードバック係数, BLM_RPM_FEEDBACK_RATE）

が、次の制御回転数として設定されます。

目標回転数と制御回転数に大きく乖離がある場合は、一気に制御回転数を変更せず少しずつ目標に近づけていくイメージとなります。

なお、前節で説明している duty は、

目標回転数（例えば、10,000rpm）、回転数（平均）（5,000rpm）の場合、現在の duty=0.3 の場合、（差が 5%以上のケース）

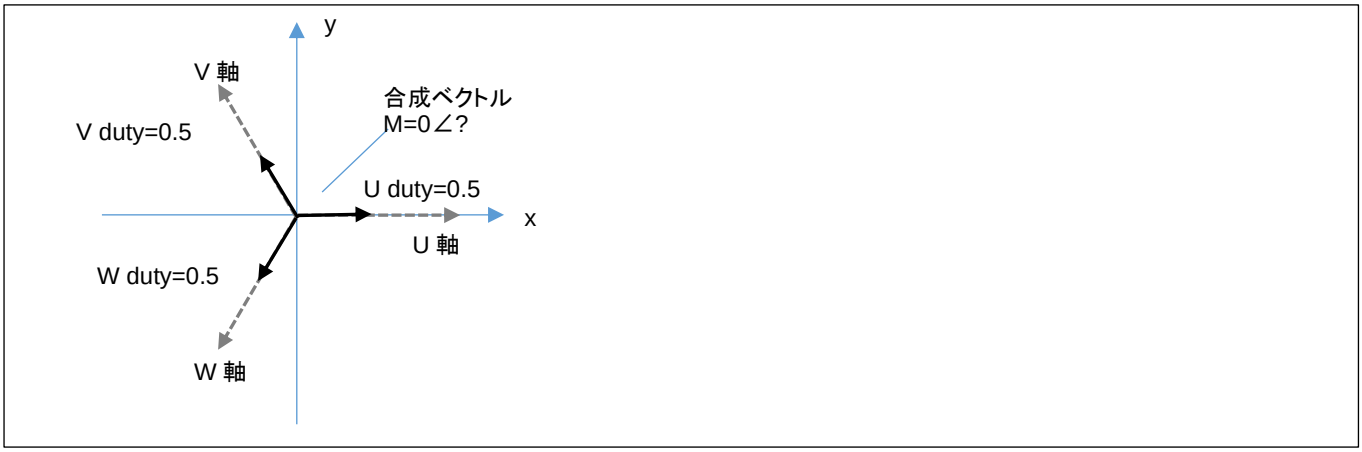
目標に対しての現状値の割合： rate = 1-(1000/5000) = 0.5
0.3 × (rate × 0.01 + 1.0) = 0.3015(30.15%)
（0.01 はフィードバック係数, BLM_DUTY_FEEDBACK_RATE）

が、新しい duty として設定されます。

制御回転数, duty とも、10ms 毎に新しい値が算出され、目標に近づけていく事となります。

1.2.3.4. ブレーキの制御[CH-1 側]

モータ回転時に、B コマンドでブレーキが掛かる様になります。



U,V,W に同じ duty を設定する事でブレーキを掛ける事が出来ます。サンプルプログラムでは、(U, V, W) = (0.5, 0.5, 0.5)としています。

0.5=50%でなくても、UVW 相に同じ duty を設定すればブレーキは掛かりますが、duty が大きいほど強いブレーキが掛かります。（サンプルプログラムでは、各相 50%としています。）

モータ回転時に SW1=OFF にしてモータを停止する場合と、B コマンドでモータを停止する場合で違いが出るはずです。また、SW=OFF 時（モータに印加する電圧 OFF）と、ブレーキ時（モータに電圧を印加）で、軸の空転に掛かる力が異なっています。

モータ回転時に、ブレーキを掛けた場合、瞬間的に電流が流れるため、ブレーキ=ON の場合は、10ms, 1s 毎の過電流検出を一時的に無効化する様にしています。

(2)CH-2 側の通常制御

1.2.3.5. duty の設定[CH-2 側]

・目標回転数と制御回転数のずれが 5%以上

→ずれが大きいので、duty を大きく調整する

・目標回転数と現在の回転数の割合×フィードバック係数(BLM_DUTY_FEEDBACK_RATE)を duty に乗算(10ms 毎)

・目標回転数と制御回転数のずれが 5%未満

→duty の変更は行わない

1.2.3.6. 制御回転数の設定[CH-2 側]

制御回転数の設定は CH-1 と同じ。

1.2.3.7. ブレーキの制御[CH-2 側]

120 度制御の場合、ブレーキに相当する印加パターンが存在しないので、CH-2 側は B コマンド入力時に単に duty を 0 に設定する操作を行います。(CH-2 側は、ブレーキと停止の動作は等価)

1.3. ホールセンサパターン取得アルゴリズム

モータ内蔵のホールセンサを使用する場合は、TUTORIAL5 で行っている方式、ホールセンサを汎用 I/O ポートに接続し、I/O ポートの 0/1 で読み取ります。

相電圧を使用してホールセンサパターンを生成する（疑似ホールセンサパターンの使用、センサレス駆動の）場合は、TUTORIAL_C, TUTORIAL_BC で説明している方式です。

SW3 により、どちらのセンサを使用するかを選択可能です。

SW3=OFF では、モータ内蔵ホールセンサ。疑似ホールセンサパターンを使用する場合は、SW2=ON です。

blm.h 内で

```
//疑似ホールセンサパターン
#define BLM_HALL_PSEUDO_SENSOR_AVERAGE 0x1//b0=1:電圧の移動平均をホールセンサパターンとする，b0=0:その時の電圧(A/D値4点の平均）をホールセンサパターンとする
#define BLM_HALL_PSEUDO_SENSOR_HYS 0x2//b1=1:ヒステリシスを有効にする，b1=0:ヒステリシス無効
#define BLM_HALL_PSEUDO_SENSOR_HYS_VAL 4//4 = 20mV/5000mV*1024，20mV程度ヒステリシスを付ける
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_0 0//疑似ホールセンサパターン使用時は速度に応じて0-40°のオフセットを付ける
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_1 10
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_2 20
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_3 30
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_4 40
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_RPM_1 2000//2000[rpm]までは0°
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_RPM_2 6000//6000[rpm]までは10°
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_RPM_3 8000//8000[rpm]までは20°
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_RPM_4 10000//10000[rpm]までは30°，それ以上は40°
```

設定があり、サンプルプログラムではデフォルトで

・相電圧は移動平均（BLM_HALL_PSEUDO_SENSOR_AVERAGE を定義）

・ヒステリシス有効化（BLM_HALL_PSEUDO_SENSOR_HYS を定義）

としています。

※移動平均を使用するか使用しないか選択可能としています（a コマンドで切り換え）

※ヒステリシスを使用するか使用しないか選択可能としています（h コマンドで切り換え）

疑似ホールセンサパターンを使用した際は、センサの切り替えタイミングが遅れる傾向があるので、本サンプルプログラムでは、磁界印加角度に回転数に応じたオフセットを付けています。(CH-1 側)

回転数	オフセット量
~2000[rpm]	0°
~6000[rpm]	10°
~8000[rpm]	20°
~10000[rpm]	30°
10000[rpm]~	40°

UVW の相電圧の平均化に関しては、blm.h 内で

(1)相電圧の長期的な平均値の算出

```
//ADC長期間の移動平均
#define BLM_ADC_LONG_AVERAGE 512 //512点の平均を求める（256, 512, 1024, 2048, 4096の値が有効）

#define BLM_ADC_LONG_AVERAGE_HIST 8 //512点の平均値の8点の移動平均を取り最終的な平均値を求める
（2,4,8,16,32の値が有効）
```

(2)相電圧の短期的な移動平均の算出

```
//ADC短期間の移動平均
#define BLM_ADC_SHORT_AVERAGE_HIST 4 //移動平均のポイント数（2,4,8,16,32の値が有効）
```

※a コマンドで電圧の平均値を使用する様にした場合に有効

(1)で求めた長期的（DC 的）な平均と、(2)で求めた短期的な移動平均（ノイズ除去が目的）の大小比較で、疑似的なホールセンサパターンを生成しています。この辺りは、TUTORIAL_C, TUTORIAL_BC で説明している話ですので、チュートリアル編のマニュアルを参照してください。

TUTORIAL_C, TUTORIAL_BC では、回転方向 CCW に対応した疑似ホールセンサパターン生成としていましたが、本サンプルプログラムでは、CCW, CW の両方の回転方向に対応したパタン生成としています。

CCW : V 相値×4 + U 相値×2 + W 相値×1
CW: $\overline{\text{W 相値}}$×4 + $\overline{\text{V 相値}}$×2 + $\overline{\text{U 相値}}$×1

（U 相値は、U 相の電圧が U 相の平均電圧より高い場合は 1，低い場合は 0）

1.4. 安全機構

過電流状態と(モータドライバボードの)過熱状態を検知してモータを停止する機能を持たせています。
g_error_check_flag 変数に、有効化したい定数の値を設定する事で、どの安全機構を有効化するか決められます。

1.4.1. 過電流保護

(1)1 回の過電流検出停止を有効化

g_error_check_flag |= BLM_ERROR_OVER_CURRENT_STOP_1;

この場合は割り込みでモータを停止させます。過電流検出端子(*INT)が 1 回でも L(過電流状態)に切り替わった際に、モータは停止となります。(C コマンドで有効、無効を切り替え。起動時は無効。)

(2)10ms 間に一定回数の過電流検出での停止を有効化

g_error_check_flag |= BLM_ERROR_OVER_CURRENT_STOP_2;

100us の割り込みルーチン内で、過電流検出端子(*INT)が L(過電流状態)であった場合、変数(g_over_current_counter_1)をインクリメントします。
10ms 毎に、変数の値を定数(BLM_OVER_CURRENT_THRESHOLD_10MS)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、モータを停止させます。
g_over_current_counter_1 変数は、10ms 毎にクリアされます。
BLM_OVER_CURRENT_THRESHOLD_10MS の初期値は、90 です。100us 毎に 10ms なので、100回チェックした中の 90 回以上、10ms 間に 90%以上過電流状態となった場合過電流保護で停止となります。

(3)1s 間に一定回数の過電流検出での停止を有効化

g_error_check_flag |= BLM_ERROR_OVER_CURRENT_STOP_3;

100us の割り込みルーチン内で、過電流検出端子(*INT)が L(過電流状態)であった場合、変数(g_over_current_counter_2)をインクリメントします。
1s 毎に、変数の値を定数(BLM_OVER_CURRENT_THRESHOLD_1S)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、モータを停止させます。
g_over_current_counter2 変数は、1s 毎にクリアされます。
BLM_OVER_CURRENT_THRESHOLD_1S の初期値は、1000 です。100us 毎に 1s なので、10,000回チェックした中で、1,000 回以上なので、1s 間に 10%以上過電流状態となった場合過電流保護で停止となります。

過電流検出は、(1)を有効化した場合、(2)(3)で停止する前に(1)で止まります。(2)(3)の設定は、(1)を無効化した場合有効です。(1)の有効・無効は C コマンドで切り替わります。

blm.h 内

```
//過電流停止カウント回数
#define BLM_OVER_CURRENT_COUNT_10MS 90 //100us毎にチェックを行い10msあたり90回以上過電流検出で停止（最大100）
#define BLM_OVER_CURRENT_COUNT_1S 1000 //100us毎にチェックを行い1sあたり1000回以上過電流検出で停止（最大10,000）
```

1.4.2. 過熱保護

g_error_check_flag |= BLM_ERROR_OVER_TEMP;

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(BLM_UPPER_TEMP)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ます。

BLM_UPPER_TEMP の初期値は 50 です（摂氏 50 度）。

```
//過熱停止[℃]
#define BLM_OVER_TEMP   50
```

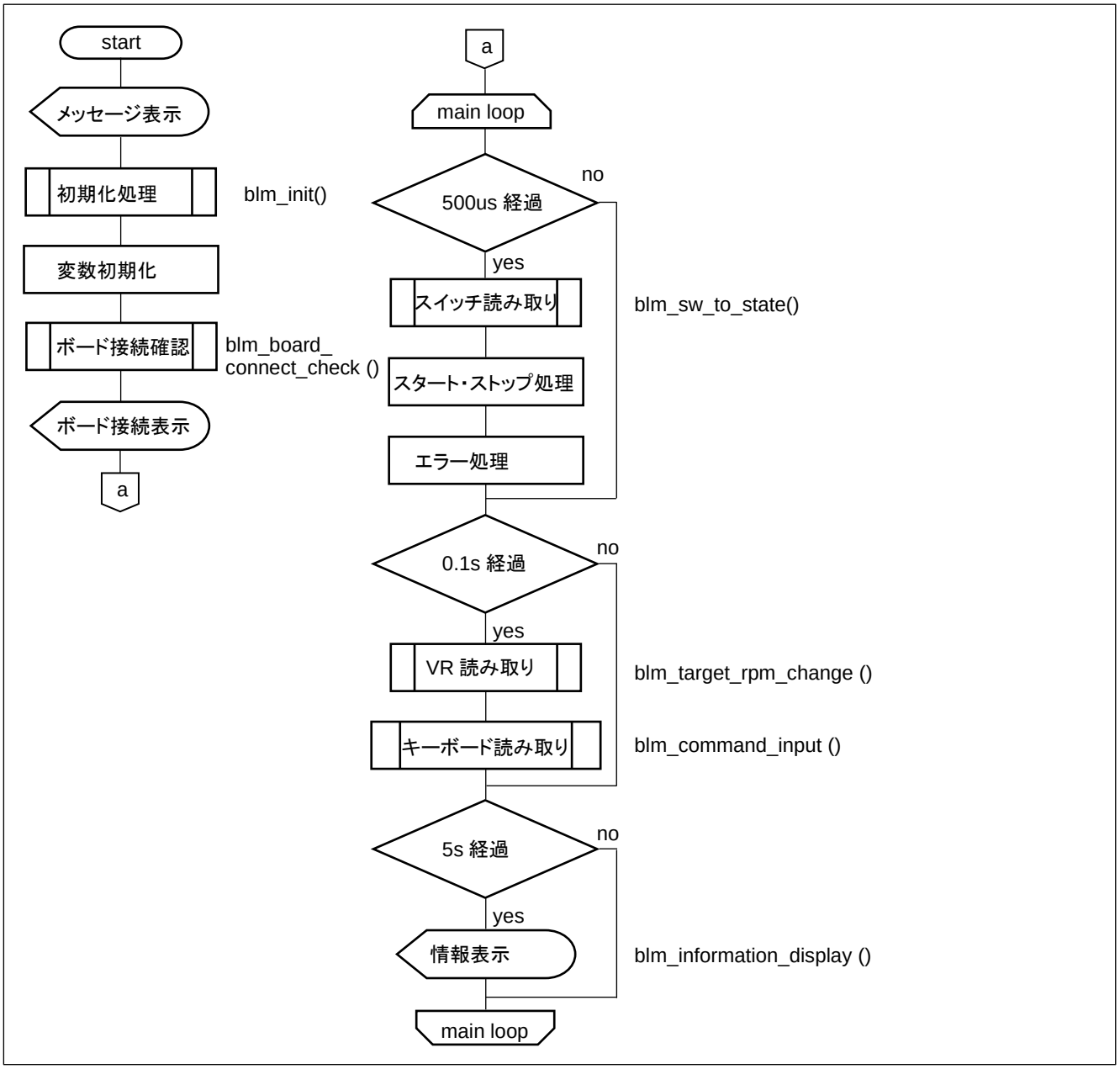
過電流停止と過熱保護のやり方は、TUTORIAL6 で行っている方式と同じです。

1.5. 関数仕様

1.5.1. 全体及び main 関数

main 関数[blm_main.c 内 blm_main()]

ーフローチャートー



メイン関数では、

・初期化
・モータドライバボードの接続確認
・スイッチの読み取り
・モータのスタート・ストップ処理
・エラー表示
・VR（ターゲット回転数）の読み取り
・キーボード読み取り
・画面表示

を行います

メッセージ表示は、今までのチュートリアル同様、シリアル(UART0)に、115,200bps で出力されます。
（※COM ポートデバッグ時は、UART2 を使用。）

モータドライバボードが接続されていないと判断された場合でも、SW を ON 側に倒すと制御信号を送ります。
（※チュートリアルでは、モータドライバボードが接続されていない場合は動作しない設定です。本サンプルプログラムでは、完全にセンサレスで使用する事を想定して、モータドライバボードが接続されていないと判断された場合でも、信号は送られます。）

1.5.2. main 関数内で実行される関数（blm_main.c に含まれる関数）

blm_sw_to_state

概要：スイッチ読み取り関数
宣言：static void blm_sw_to_state(void)
説明：
・スイッチの状態を反映しているグローバル変数読み取り
・スイッチの状態をグローバル変数に格納
・CH-1 動作中である場合 LED1 の点灯、動作でない場合 LED1 の消灯
・CH-2 動作中である場合 LED2 の点灯、動作でない場合 LED2 の消灯
・エラーが無い場合は LED4 の消灯（LED4 の点灯はエラー発生時に行う）
を行います
引数：なし
戻り値：なし
補足：
SW1/SW2 が ON 時：　g_state[ ] = BLM_CH_STATE_ACTIVE;　　//モータは動作状態
SW1/SW2 が OFF 時：　g_state[ ] = BLM_CH_STATE_INACTIVE;　//モータは停止状態
SW3 が ON 時：　g_hall_sensor = BLM_HALL_PSEUDO;　　//疑似ホールセンサパターンを使用
SW3 が OFF 時：　g_hall_sensor = BLM_HALL_MOTOR;　　//モータのホールセンサを使用
とします。（g_state がモータの動作状態を決める変数、g_hall_sensor がホールセンサの使用を決める変数）
また, g_sw_flag が ON に切り替わった際に、
現在の回転方向の設定（D コマンドで設定）により、
　回転方向の設定=CCW：　g_target_direction[ ] = BLM_CCW;　//回転方向は反時計回り(CCW)
　回転方向の設定=CW：　g_target_direction[ ] = BLM_CW;　　//回転方向は時計回り(CW)
の設定を行います。

blm_target_rpm_change

概要：目標回転数設定関数
宣言：static void blm_target_rpm_change(void)
説明：
・VR の読み取り値を目標回転数に変換
を行います。
引数：なし
戻り値：なし
補足：
　VR の角度に応じて、g_target_rpm[ ]　を設定します

blm_information_display

概要:情報表示関数
宣言:static void blm_information_display(void)
説明:
・シリアル端末への情報表示
を行います。
引数:なし
戻り値:なし
補足:
・モータドライバボードの接続有無　Motor Driver Board
・回転中かどうか　Active
・使用するホールセンサ　hall sensor
・現在のフェーズ　ratation control(PHASE)
・UVW 分解方法　UVW calculation method [CH-1 のみ]
・目標回転数　target speed([rpm])
・現在のターゲット回転数　current target speed([rpm])
・進角　forward angle([deg]) [CH-1 のみ]
・ターゲット回転方向　target direction
・回転数　rotation speed([rpm])
・回転数（平均）　rotation speed[ave]([rpm])
・温度センサー値（A/D 変換データ値）　Temperature(A/D value)
・温度　Temperature(degree)
・VR の A/D 変換データ値　VR(A/D value)
・duty 値　duty[%]
・デバッグ表示レベル　Debug print level
を表示します。

blm_command_input

概要:コマンド入力関数
宣言:static void blm_command_input(void)
説明:
・キーボードからのコマンドの処理
を行います。
引数:なし
戻り値:なし

補足:

s : information_display_flag 変数の TRUE/FALSE のトグル（5 秒毎の画面表示 ON/OFF）

A: blm_adc_hisory_display()を実行（A/D 変換結果の表示）

D: 回転方向(CCW/CW)の設定

a: g_hall_pseudo_sensor_flag の変更（センサレス駆動時平均電圧の使用）

h: g_hall_pseudo_sensor_flag の変更（センサレス駆動時ヒステリシスの使用）

B: ブレーキを ON にする（モータ回転時のみ）

q: g_angle_forward[BLM_CH_1] -= 1; 進角を 1° 遅らせる [CH-1 のみ]

w: g_angle_forward[BLM_CH_1] += 1; 進角を 1° 進める [CH-1 のみ]

e: g_angle_forward[BLM_CH_1] = 0; 進角をリセット [CH-1 のみ]

f: UVW 分解の演算で FAA を使用するか使用しないかのトグル [CH-1 のみ]

1: blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin; UVW 分解法を正弦波駆動とする [CH-1 のみ]

2: blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin_post; UVW 分解法を正弦波駆動（正規化後の duty 乗算）とする [CH-1 のみ]

3: blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin_3harmonic; UVW 分解法を正弦波+3 倍高調波駆動とする [CH-1 のみ]

4: blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin_3harmonic_post; UVW 分解法を正弦波+3 倍高調波駆動（正規化後の duty 乗算）とする [CH-1 のみ]

5: blm_angle_to_uvw_duty = blm_angle_to_uvw_duty1; UVW 分解法を別バージョン(1)とする [CH-1 のみ]

6: blm_angle_to_uvw_duty = blm_angle_to_uvw_duty2; UVW 分解法を別バージョン(2)とする [CH-1 のみ]

7: blm_angle_to_uvw_duty = blm_angle_to_uvw_duty2x; UVW 分解法を別バージョン(2)の直線近似版とする [CH-1 のみ]

C: g_error_check_flag の変更（1 回の過電流停止の有効・無効化）

z: g_debug_print_flag の変更（LEVEL1 のデバッグ用時の有効・無効化）

x: g_debug_print_flag の変更（LEVEL2 のデバッグ用時の有効・無効化）

c: g_debug_print_flag の変更（LEVEL3 のデバッグ用時の有効・無効化）

v: g_debug_print_flag の変更（LEVEL4 のデバッグ用時の有効・無効化）

b: g_debug_print_flag の変更（LEVEL5 のデバッグ用時の有効・無効化）

blm_error_display

概要:情報表示関数
宣言:static void blm_error_display(unsigned short ch)
説明:
・シリアル端末へのエラー情報表示
を行います。
引数:
　unsigned short ch : エラー表示を行う ch
戻り値:なし
補足:
　停止理由と過電流の場合は過電流のカウント数、過熱の場合は温度を表示します

blm_adc_history_display

概要:情報表示関数
宣言:static void blm_adc_history_display(void)

説明:
・シリアル端末への A/D 変換結果の表示
を行います。

引数:なし
戻り値:なし
補足:
　過去 10 点の A/D 変換の値を端末に表示します。(現在動作中の場合のみ)
※FAA 使用時は、空きメモリがないため、A/D 変換履歴は 10 点のみ保持

1.5.3. モータ制御関数（blm.c に含まれる関数）

blm_init

概要：モータ初期化関数
宣言：void blm_init(void)
説明：
・変数の初期化
・タイマ、割り込み等のマイコン周辺機能の動作開始
を行います
引数：なし
戻り値：なし

blm_start[ ]

概要：モータ動作開始関数
宣言：以下の関数への関数ポインタ
　　void blm_start_ch1(void)
　　void blm_start_ch2(void)
説明：
・モータの動作開始
を行います
引数：なし
戻り値：なし
補足：
　blm_start[BLM_CH_1]() -> blm_start_ch1()
　blm_start[BLM_CH_2]() -> blm_start_ch2()

blm_stop[ ]

概要：モータ停止関数
宣言：以下の関数への関数ポインタ
　　void blm_stop_ch1(void)
　　void blm_stop_ch2(void)

説明：
・モータの動作停止
を行います
引数：なし
戻り値：なし

blm_dutyset[ ]

概要: duty 設定関数
宣言:以下の関数への関数ポインタ
　　void blm_dutyset_ch1(short angle_mul, unsigned short duty)
　　void blm_dutyset_ch2x(short dummy, unsigned short duty)

説明:
・合成ベクトルの duty と印加角度を指定することで、相補 PWM の UVW 各相の duty に変換して設定
を行います
引数:
　short angle_mul: 印加角度（×64[°]の値で指定）
　unsigned short duty: 合成ベクトルの duty(0-65535)
戻り値: なし
補足:
　　印加角度は引数に 64 を指定した場合 1°、128 で 2°の様に°単位の角度×64 を引数に取ります
　　duty は、0-1（0-100%)を、0-65535 で指定します 引数に 32768 を指定した場合、duty=50%, 0.5 となります
　　CH-2 側は 120 度制御なので、角度の引数(angle_mul)は存在しません
　　blm_dutyset[BLM_CH_2](64, 128); の様に呼び出した場合は、磁界印加角度を示す第一引数は無視され、
　　duty の設定(128/256=50%)が有効となります。

blm_pseudo_hall_sensor_pos[ ]

概要: 疑似ホールセンサパターン取得関数
宣言:以下の関数への関数ポインタ
　　unsigned short blm_pseudo_hall_sensor_pos_ch1(short target_direction)
　　unsigned short blm_pseudo_hall_sensor_pos_ch2(short target_direction)

説明:
・UVW の相電圧の値から、現在のホールセンサ位置の推定
を行います
引数:
　short target_direction: 回転方向(CCW=1, CW=-1)
戻り値:
　1~6 のホールセンサ位置
　0, 7: ホールセンサパターンの取得に失敗

blm_angle_to_uvw_duty

概要:UVW 相の duty を求める関数
宣言:以下の関数への関数ポインタ

void blm_angle_to_uvw_duty_sin_faa(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty_sin_post_faa(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty_sin_3harmonic_faa(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty_sin_3harmonic_post_faa(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty1_faa(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty2_faa(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty2x_faa(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty_sin(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty_sin_post(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty_sin_3harmonic(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty_sin_3harmonic_post(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty1(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty2(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

void blm_angle_to_uvw_duty2x(short angle, unsigned short duty, unsigned short half_cycle, blm_uvw *phase_duty)

説明:
・合成ベクトルの duty と印加角度から UVW 相の duty に分解した値の算出
を行います
引数:
short angle: 印加角度(° 単位値で指定)
funsigned short duty: 合成ベクトルの印加 duty(0-255)
unsignd short half_cycle: タイマの半サイクル値(三角波の谷から山までのタイマサイクル値)

blm_uvw *phase_duty　各相のタイマサイクル値（計算結果,戻り値）（blm_uvw 構造体）
戻り値：なし
補足：
(1)blm_angle_to_uvw_duty_sin_faa,　デフォルト、正弦波変換, FAA 版
(2)blm_angle_to_uvw_duty_sin_post_faa,　正弦波変換の duty を後で乗算するパターン, FAA 版
(3)blm_angle_to_uvw_duty_sin_3harmonic_faa,　3 倍高調波を重畳した正弦波変換, FAA 版
(4)blm_angle_to_uvw_duty_sin_3harmonic_post_faa,　3 倍高調波を重畳した正弦波変換の duty を後で乗算, FAA 版
(5)blm_angle_to_uvw_duty1_faa,　どの角度にも 86.6%のパワーとする, FAA 版
(6)blm_angle_to_uvw_duty2_faa,　角度により最大 100%のパワーとする, FAA 版
(7)blm_angle_to_uvw_duty2x_faa,　blm_angle_to_uvw_duty2 の直線近似版, FAA 版
(8)blm_angle_to_uvw_duty_sin,　正弦波変換
(9)blm_angle_to_uvw_duty_sin_post,　正弦波変換の duty を後で乗算するパターン
(10)blm_angle_to_uvw_duty_sin_3harmonic,　3 倍高調波を重畳した正弦波変換
(11)blm_angle_to_uvw_duty_sin_3harmonic_post,　3 倍高調波を重畳した正弦波変換の duty を後で乗算
(12)blm_angle_to_uvw_duty1,　どの角度にも 86.6%のパワーとする
(13)blm_angle_to_uvw_duty2,　角度により最大 100%のパワーとする
(14)blm_angle_to_uvw_duty2x,　blm_angle_to_uvw_duty2 の直線近似版

のバージョンを用意（UVW 変換のアルゴリズムの違いを見ることが出来ます）

blm_angle_to_uvw_duty = blm_angle_to_uvw_duty_sin_3harmonic_faa;
（関数ポインタ）=（関数の実体）
の様に、14 種類の関数（実体）のどれを使用するか代入してください。
(1)と(8)は等価な動作（同様に(2)と(9)、…(7)と(14)も等価）となります。
(1)~(7)は FAA 使用版、(8)~(14)は CPU コアでの演算となります。

本関数の angle 引数は° 単位なので、g_angle[0] >> 6（g_angle を 1/64 した値）を angle 引数としてください
本関数の duty 引数は 0-255 の範囲なので、g_duty[0] >> 8（g_duty の 0-65535 を 0-255 に変換した値）を duty 引数としてください

blm_ideal_angle

概要:ホールセンサ切り替わり時の理想的な印加磁界角度を求める関数
宣言: float blm_ideal_angle(unsigned short pos, short direction, short angle_adjust_mul, unsigned short *err_flag)

説明:
・ホールセンサが切り替わったタイミングでの理想的な印加磁界角度の算出
を行います
引数:
unsigned short pos: ホールセンサ位置(1~6)
short direction: 回転方向(CCW=1, CW=-1)
short angle_adjust_mul: 印加角度補正値(° ×64 値で指定)
unsigned short *err_flag: エラーフラグ戻り値
戻り値:
理想的な角度([°]×64 値)
補足:
*err_flag ホールセンサ位置が 1~6 の値の場合は FALSE(エラーなし)、それ以外 TRUE(エラーあり)を返す

blm_angle_diff_calc

概要:制御周期毎の角度増分を求める関数
宣言: float blm_angle_diff_calc(short diff_angle_mul, short ideal_angle_mul, short angle_mul, short target_direction)
説明:
・制御周期毎に増分とする印加磁界角度の算出
を行います
引数:
short diff_angle: 現時点の制御周期毎の角度増分(×64[°]の値で指定)
short ideal_angle: ホールセンサ切り替わり時の理想的な印加角度(×64[°]の値で指定)
short angle: 現在の角度(×64[°]の値で指定)
short direction: 回転方向(CCW=1, CW=-1)
戻り値:
制御周期毎の角度増分値([°]×64 値)
ideal_angle = angle となる様に角度増分値を補正する

blm_rpm_to_diff_angle

概要:回転数から制御周期毎の角度増分を求める関数
宣言:short blm_rpm_to_diff_angle(unsigned short rpm)
説明:
・回転数から制御周期毎に増分とする印加磁界角度の算出
を行います
引数:
　unsigned short rpm:　回転数[rpm]
戻り値:
　回転数に応じた制御周期毎の角度増分値([°]×64 値)

blm_direction

概要:モータの回転方向を求める関数
宣言:short blm_direction(unsigned char pos, unsigned char prev_pos)
説明:
・ホールセンサ位置(現位置と、1 つ前の位置)からモータ回転方向の算出
を行います
引数:
　unsigned char pos:　現在のホールセンサ位置
　unsigned char prev_pos:　1 つ前のホールセンサ位置
戻り値:
　1:　回転方向は CCW(反時計回り)
　-1:　回転方向は CW(時計回り)
　-127:　回転方向不明(異常)

1.5.4. その他の関数（blm_common.c に含まれる関数）

blm_board_connect_check

概要：モータドライバボード接続確認関数
宣言：unsigned short blm_connect_check(unsigned short ch)
説明：
・モータドライバボードの接続状態の確認
を行います
引数：
　unsigned short ch: チェックを行う ch（BLM_CH_1(0)）
戻り値：
　0 :モータドライバボード非接続,
　1 :モータドライバボード接続

blm_hall_sensor_pos[ ]

概要：ホールセンサ位置関数
宣言：以下の関数への関数ポインタ
　unsigned short blm_hall_sensor_pos_ch1(void)
　unsigned short blm_hall_sensor_pos_ch2(void)

説明：
・ホールセンサ位置の算出
を行います
引数：なし
戻り値：
　HS3×4+HS2×2+HS1 で算出される 1~6 のホールセンサ値（7 の場合はモータドライバボード未接続）

blm_current_monitor[ ]

概要：過電流検出関数
宣言：以下の関数への関数ポインタ
　unsigned short blm_current_monitor_ch1(void)
　unsigned short blm_current_monitor_ch2(void)

説明：
・過電流検出
を行います
引数：なし

戻り値:

1: 過電流検出なし

0: 過電流状態

blm_board_temp

概要:温度算出関数

宣言:short blm_board_temp(unsigned short adc_val)

説明:

・温度センサ A/D 変換値から温度値の算出

を行います

引数:

unsigned short adc_val: 温度センサ A/D 変換値

戻り値:

温度値(摂氏温度、1 度単位)

blm_sw_read

概要:スイッチ読み取り関数

宣言:unsigned short blm_sw_read(void)

説明:

・スイッチの状態の読み取り

を行います

引数:なし

戻り値:

bit0: SW1=ON の時 1, OFF の時 0

bit1: SW2=ON の時 1, OFF の時 0

bit2: SW3=ON の時 1, OFF の時 0

bit3: SW4(マイコンボード上の SW2)=ON の時 1, OFF の時 0

blsm_led_out

概要:LED 設定関数

宣言:void blm_led_out(unsigned char pattern)

説明:

・変換ボード上, マイコンボード上の LED の点灯、消灯制御

を行います

引数:

unsigned char pattern:

bit0　1:LED1 点灯，0:LED1 消灯
bit1　1:LED2 点灯，0:LED2 消灯
bit2　1:LED3 点灯，0:LED3 消灯
bit3　1:LED4 点灯，0:LED4 消灯
bit4　1:LED5（マイコンボード上の LED2）点灯，0:LED5 消灯

戻り値：なし

blm_led_change_on

概要：LED 設定関数
宣言：void blm_led_change_on(unsigned char pattern)
説明：
・LED の点灯への切り替え
を行います

引数：
unsigned char pattern:
bit0　1:LED1 点灯，0:LED1 現在の状態から変更しない
bit1　1:LED2 点灯，0:LED2 現在の状態から変更しない
bit2　1:LED3 点灯，0:LED3 現在の状態から変更しない
bit3　1:LED4 点灯，0:LED4 現在の状態から変更しない
bit4　1:LED5 点灯，0:LED5 現在の状態から変更しない

戻り値：なし

blm_led_change_off

概要：LED 設定関数
宣言：void blm_led_change_off(unsigned char pattern)
説明：
・LED の消灯への切り替え
を行います

引数：
unsigned char pattern:
bit0　1:LED1 消灯，0:LED1 現在の状態から変更しない
bit1　1:LED2 消灯，0:LED2 現在の状態から変更しない
bit2　1:LED3 消灯，0:LED3 現在の状態から変更しない
bit3　1:LED4 消灯，0:LED4 現在の状態から変更しない
bit4　1:LED5 消灯，0:LED5 現在の状態から変更しない

戻り値：なし

1.5.5. 割り込み関数

割り込み関数は、blm_intr.c 内で定義され、タイマや ADC 処理等を行っています。

blm_interrupt_tau00

概要:TAU0_0 タイマ割り込み（100us 周期）
宣言:void blm_interrupt_tau00(void)
説明:
・モータの回転数の取得
・A/D 変換の起動
を行います
補足:
blm_interrupt_tau00 から、モータが ON の場合 blm_interrupt_ch?関数を呼び出し blm_interrupt_ch?関数内で以下の処理を行っています
・ホールセンサ位置の取得
・磁界印加角度の設定（CH-1 側のみ）
・過電流モニタ
・回転方向の算出
・磁界印加角度の補正（CH-1 側のみ）
・duty 補正値の算出（CH-1 側のみ）

blm_interrupt_itl0001

概要:ITL000_001 タイマ割り込み（10ms 周期）
宣言:void blm_interrupt_itl0001(void)
説明:
・過電流停止処理
・過熱停止処理
・動作フェースの遷移
・duty 補正（フィードバック）
・回転数の計算
を行います
補足:
モータ制御の定期処理の周期割り込みです。本サンプルプログラムでは、モータ制御に関わる定期的に実行したい処理を本関数内で行っています。

blm_interrupt_itl1213

概要:ITL012_ITL013 タイマ割り込み(500ms 周期)
宣言:void blm_interrupt_itl1213(void)
説明:
・変数のインクリメント
を行います
補足:
　メイン関数内で 5 秒毎のメッセージ表示用のタイマ

blm_interrupt_adc

概要:A/D 変換完了割り込み
宣言:void blm_interrupt_adc(void)
説明:
・A/D 変換結果の回収(グローバル変数にコピー)
・次の端子の A/D 変換の実行
を行います
補足:
　　最後の A/D 変換対象端子の A/D 変換結果の回収を行った後、blm_post_adc_ch?関数を呼び出し、
　　blm_post_adc_ch?関数内で以下の処理を行っています
　・UVW 相の平均電圧(長周期と短周期の 2 種類)を算出
　・A/D 変換結果の履歴(10 回分)をグローバル変数に保存

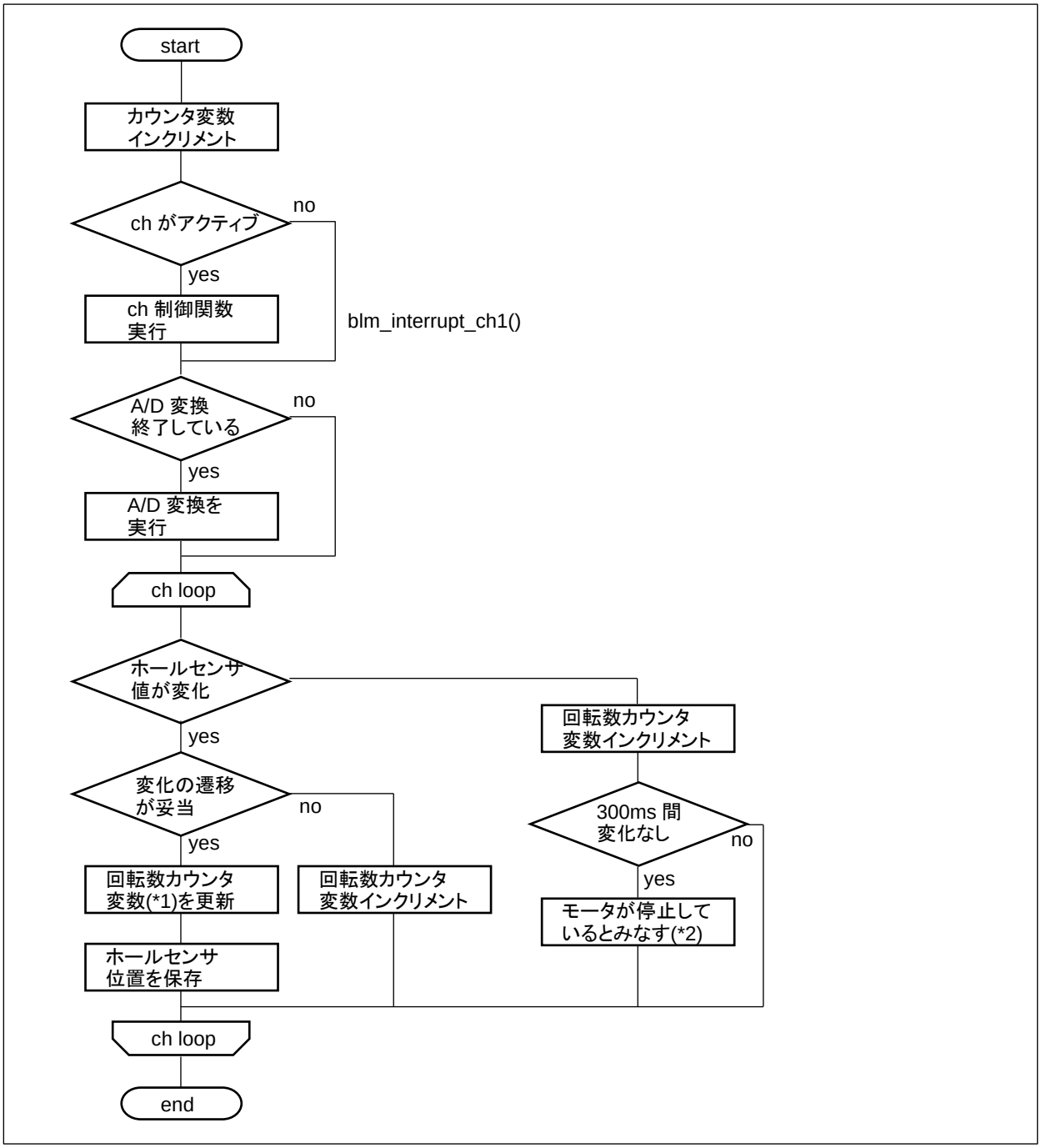
blm_interrupt_intc7
blm_interrupt_intc4

概要:端子割り込み
宣言:void blm_interrupt_intc7(void)　…CH-1 側
宣言:void blm_interrupt_intc4(void)　…CH-2 側

説明:
・*INT(過電流信号)の検出
を行います
補足:
　1 回の過電流信号でモータを停止させる設定の場合有効な割り込み
(10ms 毎, 1 秒毎の過電流停止の場合は、本関数は使用されない)

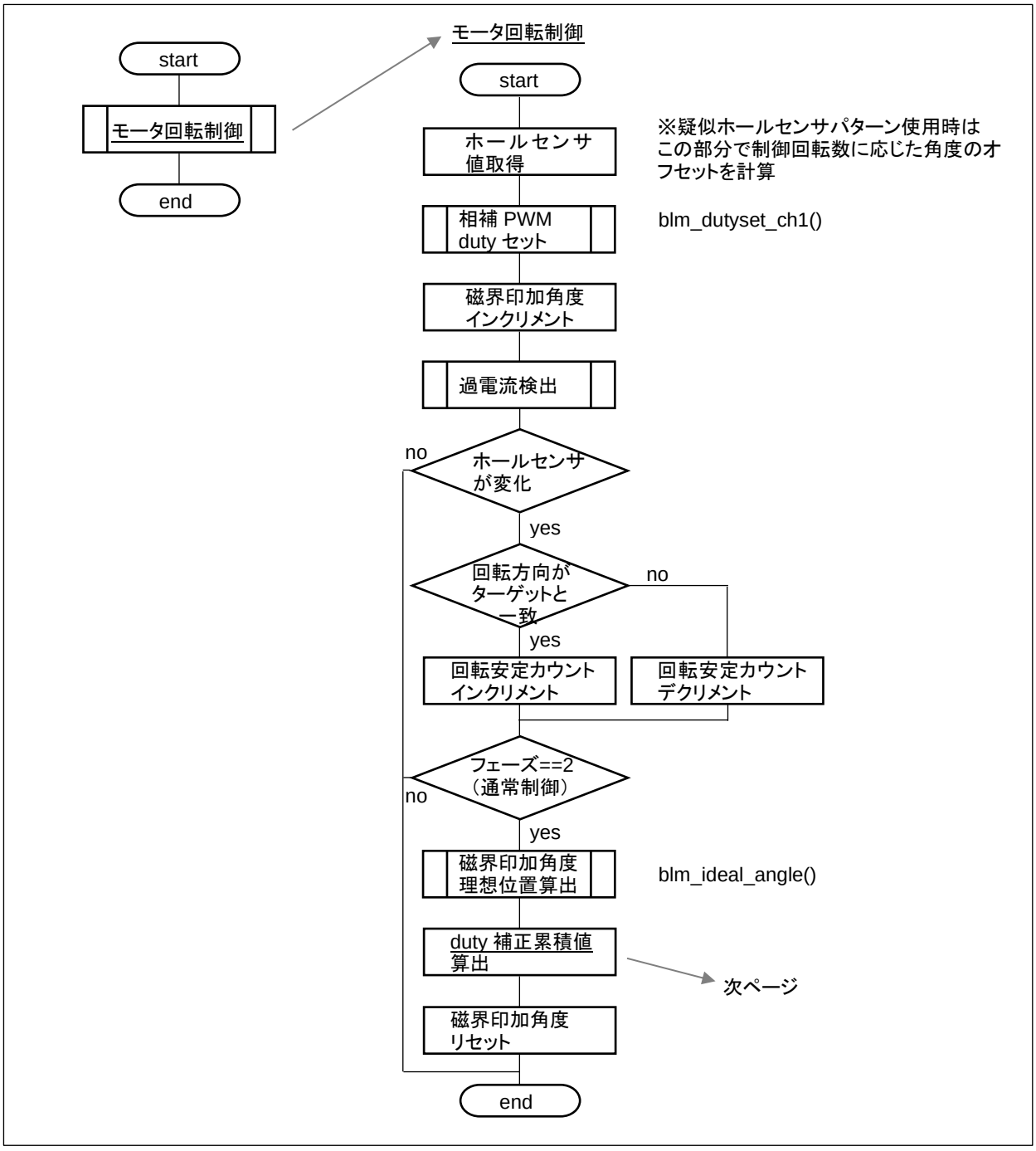
1.6. フローチャート

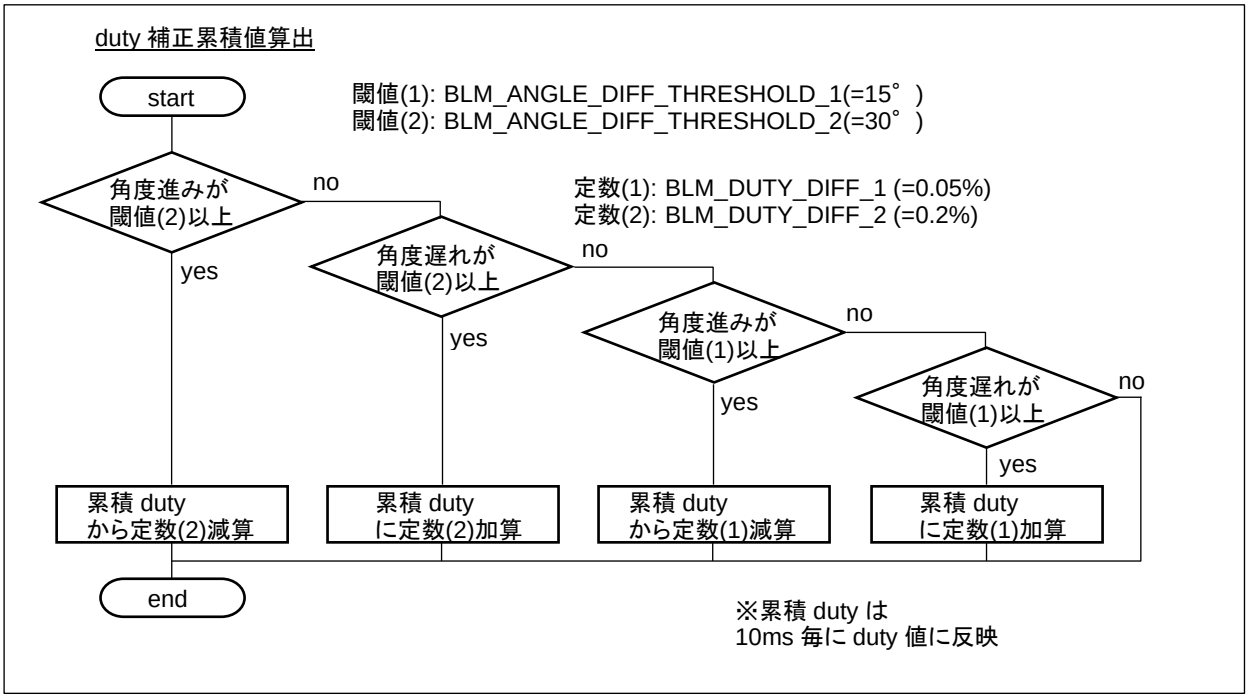
－blm_interrupt_tau00（100us 毎の割り込み）フローチャート－



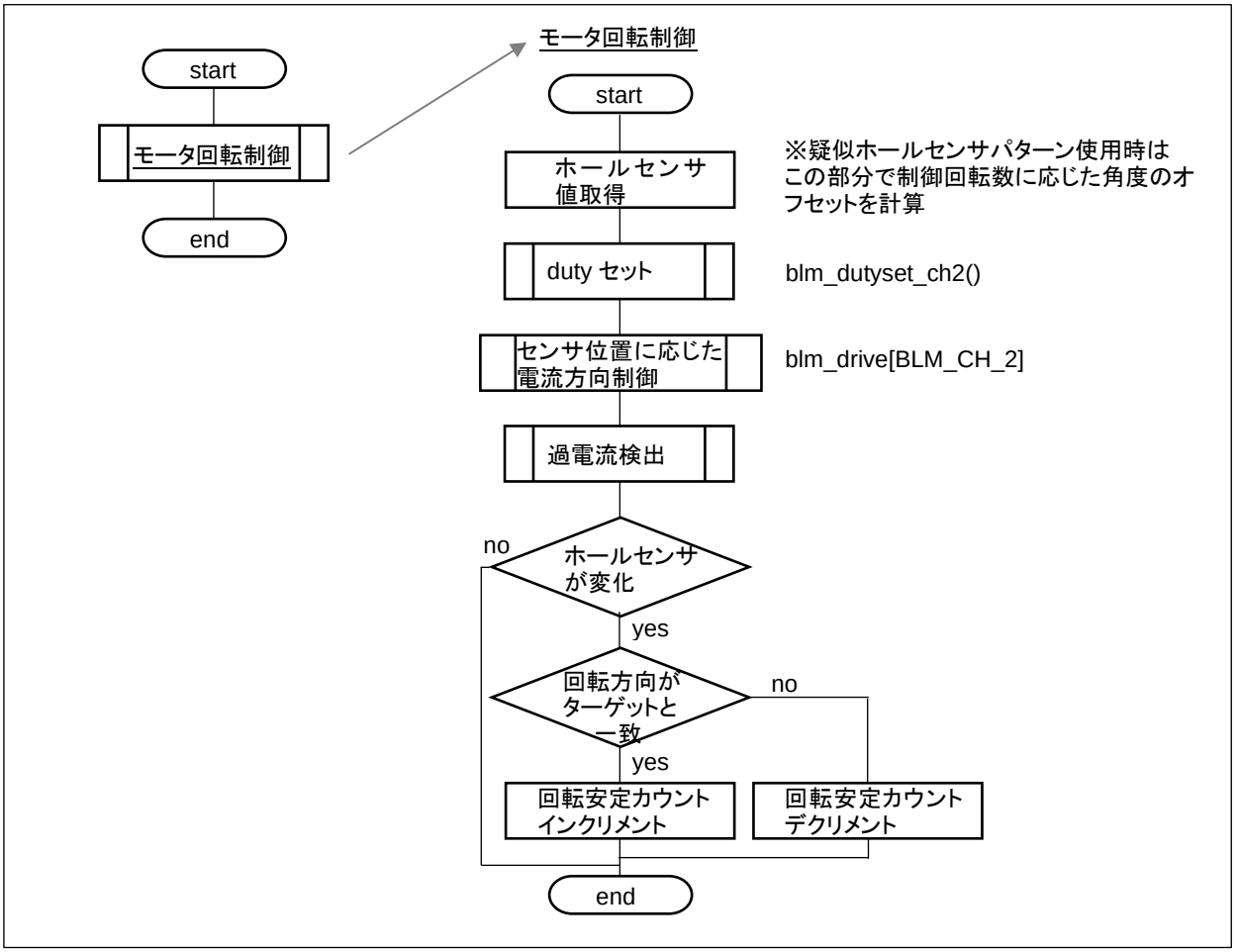
(*1)10ms の周期処理で回転数を算出するのに使用
(*2)始動制御からやり直し

ーblm_intrrupt_ch1（100us 毎の割り込みから呼ばれる関数）全体フローチャートー

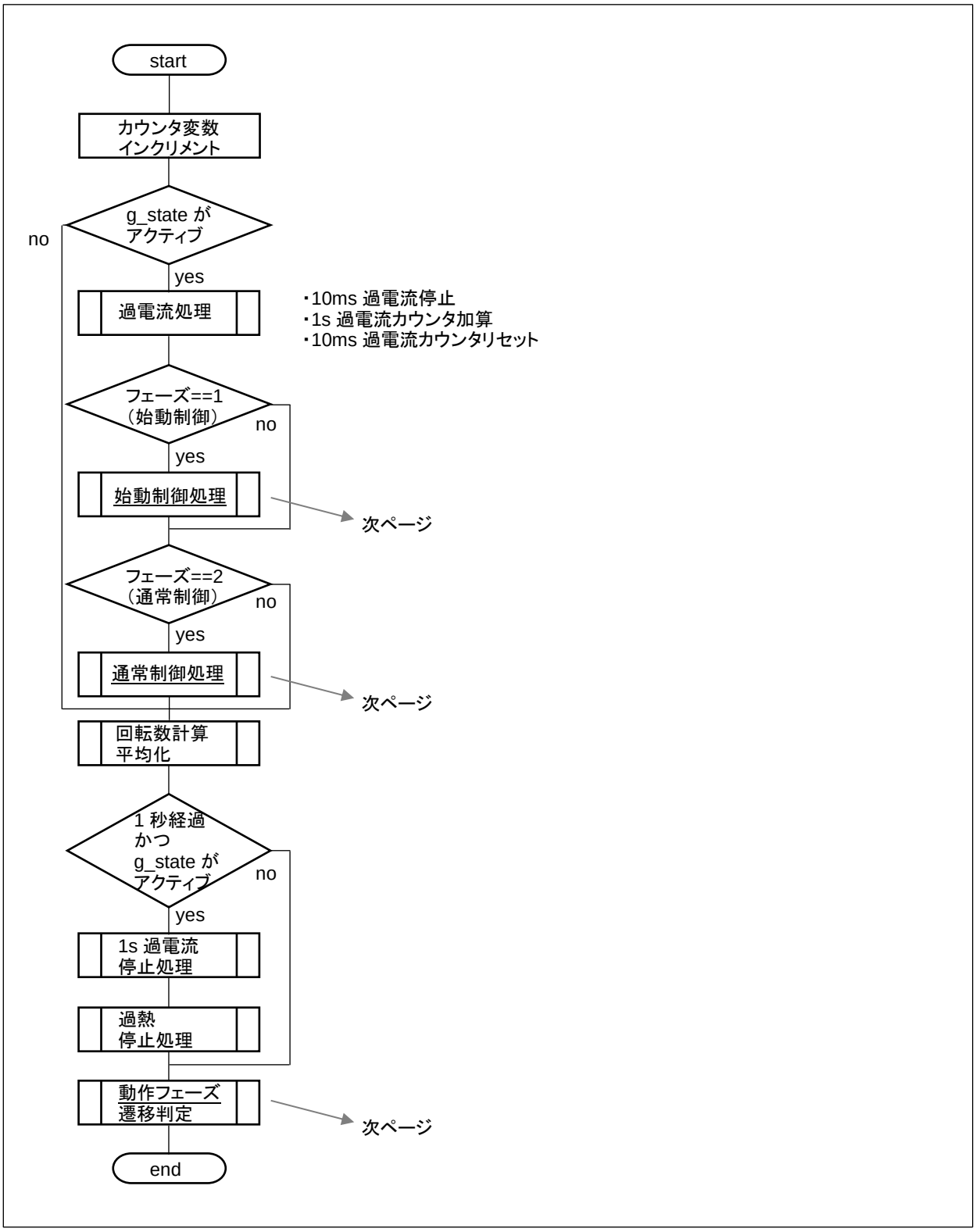


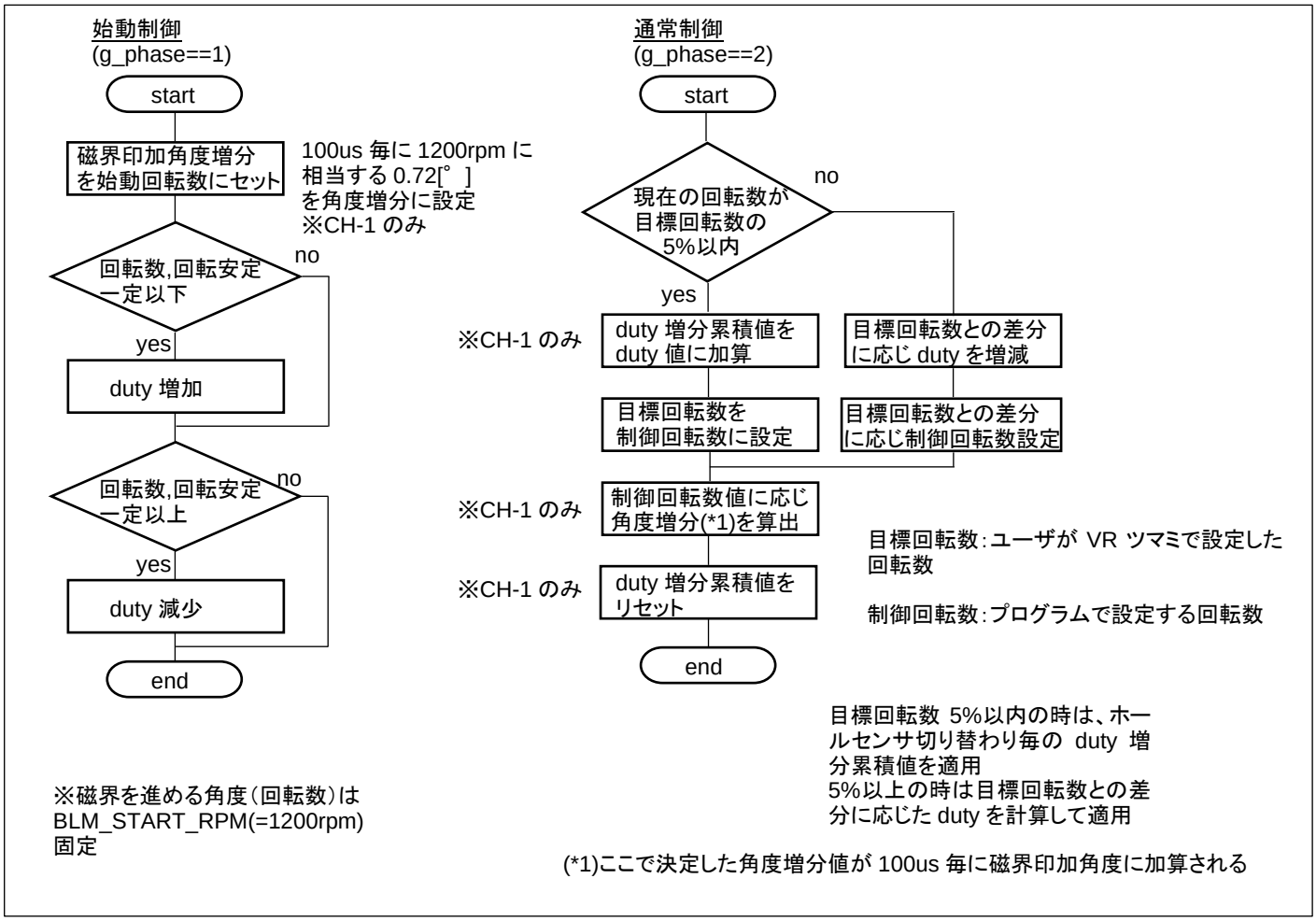

duty 補正累積値算出
start
閾値(1): BLM_ANGLE_DIFF_THRESHOLD_1(=15°)
閾値(2): BLM_ANGLE_DIFF_THRESHOLD_2(=30°)
角度進みが閾値(2)以上
no
yes
定数(1): BLM_DUTY_DIFF_1 (=0.05%)
定数(2): BLM_DUTY_DIFF_2 (=0.2%)
角度遅れが閾値(2)以上
no
yes
角度進みが閾値(1)以上
no
yes
角度遅れが閾値(1)以上
no
yes
累積 duty から定数(2)減算
累積 duty に定数(2)加算
累積 duty から定数(1)減算
累積 duty に定数(1)加算
end
※累積 duty は
10ms 毎に duty 値に反映

ーblm_intrrupt_ch2（100us 毎の割り込みから呼ばれる関数）全体フローチャートー

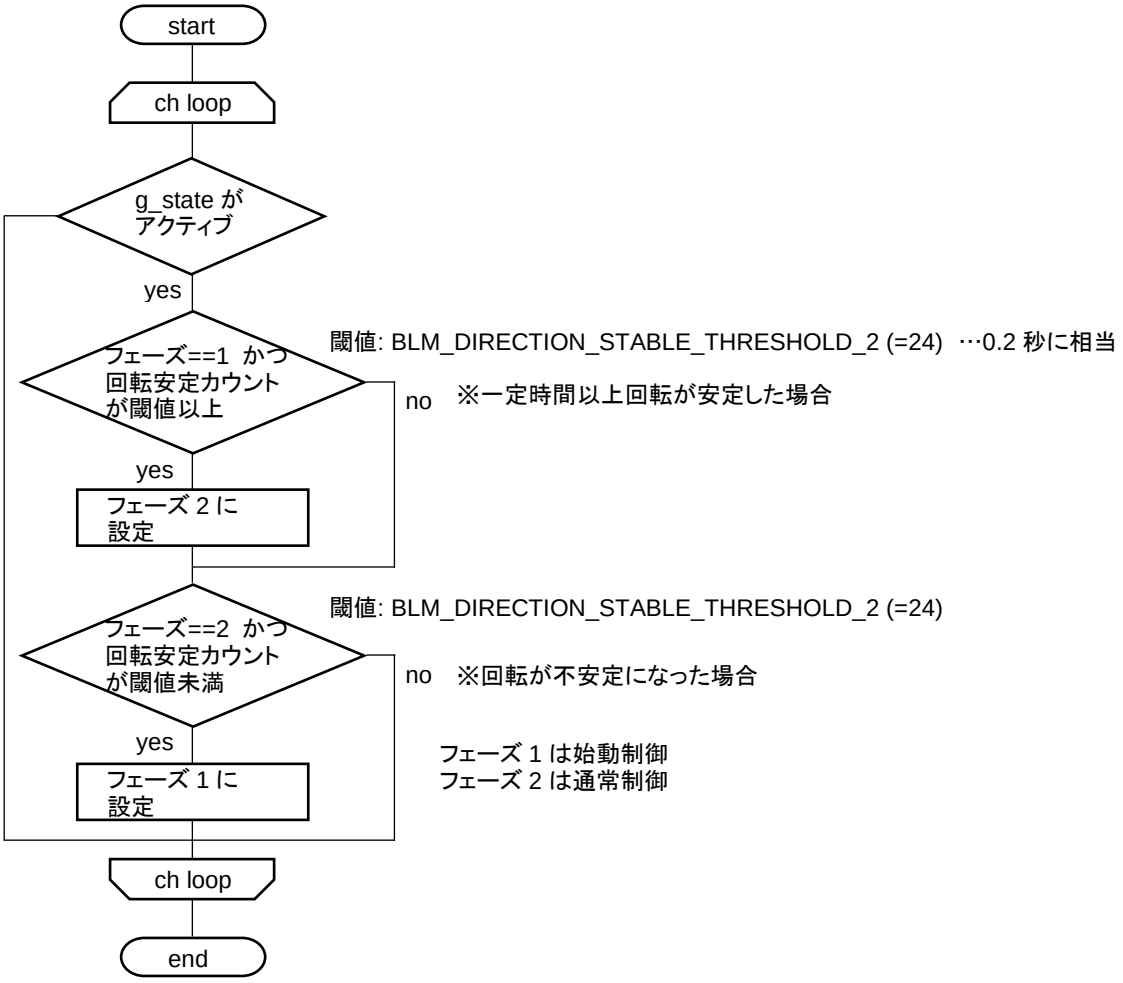


ーblm_interrupt_itl0001（10ms 毎の割り込み）フローチャートー




始動制御
(g_phase==1)
start
磁界印加角度増分
を始動回転数にセット
100us 毎に 1200rpm に
相当する 0.72[°]
を角度増分に設定
※CH-1 のみ
回転数,回転安定
一定以下
no
yes
duty 増加
回転数,回転安定
一定以上
no
yes
duty 減少
end
※磁界を進める角度(回転数)は
BLM_START_RPM(=1200rpm)
固定
通常制御
(g_phase==2)
start
現在の回転数が
目標回転数の
5%以内
no
yes
※CH-1 のみ
duty 増分累積値を
duty 値に加算
目標回転数との差分
に応じ duty を増減
目標回転数を
制御回転数に設定
目標回転数との差分
に応じ制御回転数設定
※CH-1 のみ
制御回転数値に応じ
角度増分(*1)を算出
※CH-1 のみ
duty 増分累積値を
リセット
end
目標回転数:ユーザが VR ツマミで設定した
回転数
制御回転数:プログラムで設定する回転数
目標回転数 5%以内の時は、ホー
ルセンサ切り替わり毎の duty 増
分累積値を適用
5%以上の時は目標回転数との差
分に応じた duty を計算して適用
(*1)ここで決定した角度増分値が 100us 毎に磁界印加角度に加算される

動作フェーズ遷移判定


start
ch loop
g_state が
アクティブ
yes
フェーズ==1 かつ
回転安定カウント
が閾値以上
閾値: BLM_DIRECTION_STABLE_THRESHOLD_2 (=24) …0.2 秒に相当
no ※一定時間以上回転が安定した場合
yes
フェーズ 2 に
設定
フェーズ==2 かつ
回転安定カウント
が閾値未満
閾値: BLM_DIRECTION_STABLE_THRESHOLD_2 (=24)
no ※回転が不安定になった場合
yes
フェーズ 1 に
設定
フェーズ 1 は始動制御
フェーズ 2 は通常制御
ch loop
end

1.7. グローバル変数

g_blm_drive_method

概要:駆動方式を格納する変数
宣言:const unsigned short g_blm_drive_method[BLM_CH_NUM]
説明:
BLM_DRIVE_6_PHASE_PWM(0)　相補 PWM
BLM_DRIVE_120_DEG(1)　120 度制御
のいずれかが設定される
補足:
g_blm_drive_method[BLM_CH_1] = BLM_DRIVE_6_PHASE_PWM;
g_blm_drive_method[BLM_CH_2] = BLM_DRIVE_120_DEG;

g_adc_scan_flag

概要:A/D のスキャン状態を示すフラグ変数
宣言:volatile unsigned short g_adc_scan_flag
説明:
0~BLM_ADC_TERMINALS(14)の値を取る
次の A/D 変換対象端子の番号を保持する変数

g_adc_scan_sequence

概要:A/D のスキャン対象端子を指定する変数
宣言:const unsigned char g_adc_scan_sequence[BLM_ADC_TERMINALS]
説明:
g_adc_scan_sequence[BLM_ADC_TERMINALS] = {0, 1, 2, 3, 4, 5, 6, 7, 16, 17, 18, 19, 28, 30};
1 回目の A/D 変換は ANI0, 2 回目は ANI1, …14 回目は ANI30 端子を A/D 変換対象として定義

g_adc_result_pointer

概要:A/D のスキャン結果を特定の変数に紐づけするポインタ
宣言:unsigned short *g_adc_result_pointer[BLM_ADC_TERMINALS]
説明:
g_adc_result_pointer[0] = &g_adc_result[BLM_CH_1].volume;
1 回目の A/D 変換(ANI0)の変換結果を、CH-1 の volume 変数に格納する様に設定

blm_adc g_adc_result

概要:A/D 変換結果を格納する変数
宣言:blm_adc g_adc_result[BLM_CH_NUM]
説明:
g_adc_result には、A/D 変換結果が格納される(100us 毎に更新)
補足:
blm_adc は、構造体で、以下のメンバを含む。

```
typedef struct{
    unsigned short volume;          //VRの値
    unsigned short v_u_phase;    //U相の電圧
    unsigned short v_v_phase;    //V相の電圧
    unsigned short v_w_phase;    //W相の電圧
    unsigned short i_u_phase;    //U相の電流
    unsigned short i_v_phase;    //V相の電圧
    unsigned short i_w_phase;    //W相の電圧
    unsigned short temp;            //温度センサの値
    unsigned short v_power;      //電源電圧
} blm_adc;
```

CH-1 の U 相の電圧の A/D 変換値は、g_adc_result.v_u_phase[BLM_CH_1]

g_adc_result_buf_ch1
g_adc_result_buf_ch2

概要:A/D 変換結果の履歴を格納する変数
宣言:blm_adc g_adc_result_buf_ch1[BLM_ADC_HIST] // BLM_ADC_HIST = 10
blm_adc g_adc_result_buf_ch2[BLM_ADC_HIST]
説明:
g_adc_result_buf_ch?[ ]には、A/D 変換結果が(直前の 10 ポイントが)格納される(100us 毎に更新)
リングバッファ状にデータが格納。最新のデータは、インデックスが g_adc_result_buf_index-1 のデータ。

g_adc_result_buf_index

概要:A/D 変換結果の履歴を格納する変数のインデックス
宣言:unsigned short g_adc_result_buf_index[BLM_CH_NUM] //BLM_CH_NUM = 2
説明:
adc_result_buf_ch?[ ]のインデックスを保持する変数

g_adc_result_buf_flag

概要:A/D 変換結果の履歴の格納を抑止する変数
宣言:volatile unsigned short g_adc_result_buf_flag
説明:
フラグがセット(0 以外)されている場合、g_adc_result_buf_ch?を更新しない
A/D 変換の結果の表示中にフラグをセットする

g_adc_v_phase_short_term_average
g_adc_v_phase_long_term_average

概要:相電圧の平均値を格納する変数
宣言:blm_uvw_uint16 g_adc_v_phase_short_term_average[BLM_CH_NUM]
blm_uvw_uint16 g_adc_v_phase_long_term_average[BLM_CH_NUM]
説明:
g_adc_v_phase_short_term_average は、相電圧の短期間の平均、直近 4 点の移動平均。
g_adc_v_phase_long_term_average は、相電圧の長期間の平均(512 点の平均値の、8 点の移動平均。約 400ms の平均)。
※下線部の値は定数値(変更可能)
補足:

```
typedef struct{
    unsigned short u;        //U 相
    unsigned short v;        //V 相
    unsigned short w;        //W 相
} blm_uvw_uint16;
```

g_adc_v_phase_long_term_average[BLM_CH_1].u が、CH-1 の U 相の平均電圧

g_state

概要:チャネル状態変数
宣言:volatile unsigned short g_state[BLM_CH_NUM]
説明:
現在の状態を示す変数
BLM_CH_STARE_INACTIVE(0): 停止
BLM_CH_STARE_ACTIVE(1): 回転

g_board_connect_info

概要:モータドライバボード接続状態変数
宣言:volatile unsigned short g_board_connect_info[BLM_CH_NUM]
説明:
モータドライバボード接続状態を示す変数
BLM_NO_CONNECT(0): モータドライバボード未接続
BLM_CONNECT(1): モータドライバボード接続

g_error

概要:エラー状態変数
宣言:volatile blm_error g_error[BLM_CH_NUM]
説明:
g_error.status
BLM_NO_ERROR(0): エラーなし
BLM_ERROR_OVER_TEMP_STOP(0x1): 過熱停止
BLM_ERROR_OVER_CURRENT_STOP1(0x2): 過電流停止 1(1 回の割り込みで停止)
BLM_ERROR_OVER_CURRENT_STOP2(0x4): 過電流停止 2(10ms 間に規定回数オーバ)
BLM_ERROR_OVER_CURRENT_STOP3(0x8): 過電流停止 3(1s 間に規定回数オーバ)
g_error.temp 過熱停止時の温度
g_error.over_current_count_1 10ms 間の過電流カウント数
g_error.over_current_count_1 1s 間の過電流カウント数
補足:
blm_error は、構造体で、以下のメンバを含む。

```
typedef struct{
    unsigned short status;
    short temp;                          //温度
    unsigned short over_current_count_1;  //10msの過電流検出回数
    unsigned short over_current_count_2;  //1sの過電流検出回数
} blm_error;
```

g_error_check_flag

概要:エラーチェック項目設定変数
宣言:volatile unsigned short g_error_check_flag
説明:
エラーチェック対象を指定
補足:
g_error_check_flag = BLM_ERROR_OVER_TEMP_STOP | BLM_ERROR_OVER_CURRENT_STOP3;
の場合、過熱停止と、1 秒毎の過電流停止を有効化。
BLM_ERROR_OVER_TEMP_STOP(0x1) 過熱停止
BLM_ERROR_OVER_CURRENT_STOP1(0x2) 1 回の過電流検出で停止
BLM_ERROR_OVER_CURRENT_STOP2(0x4) 10ms 毎一定回数の過電流検出で停止
BLM_ERROR_OVER_CURRENT_STOP3(0x8) 1s 毎一定回数の過電流検出で停止
を OR で指定。

g_duty

概要:duty 比設定変数
宣言:volatile long g_duty[BLM_CH_NUM]
説明:
duty 比を設定する変数。0-100%を 0-65535 で取り扱う。
本変数値を適切な値（目標回転数に合った値）となるように制御を行う。

g_sensor_pos

概要:ホールセンサ位置変数
宣言:volatile unsigned short g_sensor_pos[BLM_CH_NUM]
説明:
100us 毎に読み取ったホールセンサの情報を 1~6 の数値に変換した値を格納

g_rotation_counter

概要:回転数計カウンタ変数
宣言:volatile unsigned short g_rotation_counter[BLM_CH_NUM]
説明:
100us 毎にインクリメント、ホールセンサの値が変化した際にリセットされる変数

g_rpm

概要:回転数を示す変数
宣言:volatile unsigned short g_rpm[BLM_CH_NUM]
説明:
現在の回転数(ホールセンサが変化した際に計算され更新)

g_rpm_ave

概要:平均の回転数を示す変数
宣言:volatile unsigned short g_rpm_ave[BLM_CH_NUM]
説明:
現在の平均回転数(10ms 毎の回転数の過去 16 値の平均値)
過去の値を保存するのに、g_rpm_hist[ ][16], g_rpm_hist_index[ ]を使用。

g_rpm_hist

概要:回転数計算変数
宣言:unsigned short g_rpm_hist[BLM_CH_NUM][16]
説明:
10ms 毎の回転数の過去 16 値(リングバッファ状に保存)

g_rpm_hist_index

概要:回転数計算変数
宣言:unsigned short g_rpm_hist_index[BLM_CH_NUM]
説明:
10ms 毎の回転数の過去の値のリングバッファのインデックス

g_over_current_counter_1
g_over_current_counter_2

概要:過電流検出回数を示す変数
宣言:volatile unsigned short g_over_current_counter_1[BLM_CH_NUM]
volatile unsigned short g_over_current_counter_2[BLM_CH_NUM]
説明:
g_over_current_counter_1: 10ms 間に 100us 毎に過電流チェックを行い過電流検出された回数(10ms でリセット)
g_over_current_counter_2: 1s 間に 100us 毎に過電流チェックを行い過電流検出された回数(1s でリセット)

g_phase

概要:現在の動作フェーズを示す変数
宣言:volatile unsigned short g_phase[BLM_CH_NUM]
説明:
BLM_PHASE_0(0): 停止
BLM_PHASE_1(1): 始動制御
BLM_PHASE_2(2): 通常制御
BLM_PHASE_3(3): ブレーキ

g_hall_sensor

概要:ホールセンサ区分変数
宣言:volatile unsigned short g_hall_sensor
説明:
モータ内蔵のホールセンサを使用するか、相電圧から計算される疑似ホールセンサパターンを使うかを選択する変数。
補足:
BLM_HALL_MOTOR(1): モータ内蔵のホールセンサを使用
BLM_HALL_PSEUDO(2): 疑似ホールセンサパターンを使用

g_hall_pseudo_sensor_flag

概要:疑似ホールセンサフラグ変数
宣言:volatile unsigned short g_hall_pseudo_sensor_flag
説明:
相電圧から計算される疑似ホールセンサパターンを使う場合、「平均電圧」「ヒステリシス」の使用を選択するフラグ変数。
補足:
BLM_HALL_PSEUDO_SENSOR_AVERAGE(0x1): 平均電圧を使用
BLM_HALL_PSEUDO_SENSOR_HYS(0x2): ヒステリシスを使用
平均電圧とヒステリシスの両方を有効にする場合は、
g_hall_pseudo_sensor_flag =
BLM_HALL_PSEUDO_SENSOR_AVERAGE | BLM_HALL_PSEUDO_SENSOR_HYS;

g_target_direction

概要:回転方向の設定変数
宣言:volatile short g_target_direction[BLM_CH_NUM]

説明:

BLM_CCW(1): 反時計回り(Counter Clock Wise)

BLM_CW(-1): 時計回り(Clock Wise)

BLM_BRAKE(2): ブレーキ

BLM_STOP(0): 停止

g_direction

概要:現在の回転方向を示す変数

宣言:volatile short g_direction[BLM_CH_NUM]

説明:

BLM_CCW(1): 反時計回り(Counter Clock Wise)

BLM_CW(-1): 時計回り(Clock Wise)

BLM_BRAKE(2): ブレーキ

BLM_STOP(0): 停止

BLM_UNKNOWN(-127): センサの切り替わりが異常

g_stable

概要:回転方向の安定を示す変数

宣言:volatile unsigned short g_stable[BLM_CH_NUM]

説明:

ホールセンサ切り替わり時、切り替わり前の値との比較で回転方向(g_direction)を判断します。回転方向が、設定した回転方向(g_target_direction)と一致しているとき、回転は安定していると判断して、g_stable をインクリメントします。不一致の時は、g_stable をデクリメントします。

g_target_rpm

概要:設定回転数を示す変数

宣言:volatile unsigned short g_target_rpm[BLM_CH_NUM]

説明:

ユーザが VR ツマミで設定した回転数を格納する変数。単位は、rpm 値で格納されます。この回転数に近づくように回転数を制御します。

g_current_target_rpm

概要:現在の制御回転数を示す変数

宣言:volatile unsigned short g_current_target_rpm[BLM_CH_NUM]

説明:
現在の制御回転数を格納する変数で、100us 毎に磁界印加角度を進める際の角度は、この変数から算出されます。g_current_target_rpm（制御回転数）と g_target_rpm（目標回転数）の差分が小さいときには、g_current_target_rpm = g_target_rpm となる様に設定し、差分が大きいときは一定のレート(BLM_RPM_FEEDBACK_RATE)で、g_current_target_rpm を g_target_rpm に近づけていきます。

g_phase1_diff_array

概要:始動時の duty を変化させるテーブル変数
宣言:const unsigned short g_phase1_diff_array[8]
説明:
{131, 131, 163, 262, 524, 1048, 2097, 4194};
0.2% 0.2% 0.25% 0.4% 0.8% 1.6% 3.2% 6.4%
始動制御時の立ち上がりを速くするためのテーブル。始動制御時は、本テーブルに書かれている値で duty の立ち上がりのカーブを決定する。
g_phase1_diff_index は、本変数のインデックスで使用される。

g_phase1_diff_index

概要:始動時の duty を変化させるテーブルのインデックス
宣言:volatile unsigned short g_phase1_diff_index[BLM_CH_NUM]
説明:
duty の立ち上げ時には、g_phase1_diff_index[ ] が 7→6→5→...→0 に変化する。duty の増やし方は、+6.4%, +3.2%, ... +0.2%となる。

g_duty_diff_integral

概要:duty 比設定変数
宣言:volatile short g_duty_diff_integral[BLM_CH_NUM]
説明:
通常制御時、duty の累積差分値を格納する変数。ホールセンサ切り替わりのポイントで、回転数が速い（duty を減少させる）、回転数が遅い（duty を増加させる）に応じた duty の変更値を本変数に格納し、10ms のタイミングで g_duty に本変数の値を加算する。（相補 PWM 制御時のみ使用）

g_angle

概要:印加角度設定変数
宣言:volatile short g_angle[BLM_CH_NUM]

説明:
印加角度を設定する変数。度×64 単位。100us 毎に、本変数の値と duty 比の値(g_duty)を UVW 分解して、実際の PWM 波形に反映させる。(相補 PWM 制御時のみ使用)

g_angle_diff

概要:印加角度差分設定変数
宣言:volatile short g_angle_diff[BLM_CH_NUM]
説明:
100us 毎に本変数を、g_angle に加算(CCW 時)、減算(CW 時)する。100us 毎の磁界印加角度の増分値。g_current_target_rpm から算出される。(相補 PWM 制御時のみ使用)

g_angle_forward

概要:進角調整設定変数
宣言:volatile short g_angle_forward[BLM_CH_NUM]
説明:
ホールセンサ切り替わり時の理想印加角度の調整を行う変数です。初期値 0。ラジアン単位。
正の値の時は、進み方向(速めに磁界印加角度をスイッチング)、負の値は遅れ方向です。
(キーボードから進角調整を行うと、度単位での調整となりますが、プログラム内ではラジアンに変換されて、進角値が調整されます。)(相補 PWM 制御時のみ使用)

g_timer_count

概要:タイマ周期のを示す変数
宣言:unsigned short g_timer_count[BLM_CH_NUM]
説明:
相補 PWM を構成するタイマーの谷から山まで(山から谷まで)のカウント値を保持する変数。1 相 PWM の場合は、タイマ周期。PWM キャリア周波数を途中で変更しない限りは、決め打ちの値でも良いが、変数としている。

g_blm_angle_to_uvw_method

概要:UVW 変換アルゴリズムを示す変数
宣言:unsigned short g_blm_angle_to_uvw_method
説明:
7 種類用意している UVW 変換アルゴリズムのどのアルゴリズムを使用しているかを示す変数。1~7
(どの UVW 変換アルゴリズムを使用しているか、画面表示のための変数)

g_timer1_counter
g_timer2_counter
g_timer3_counter

概要:カウンタ変数
宣言:volatile unsigned short g_timer1_counter
　　volatile unsigned short g_timer2_counter
　　volatile unsigned short g_timer3_counter
説明:
　TAU0_0(100us), ITL000_ITL001(10ms), ITL012_ITL013(500ms)毎にインリメントされる変数。メイン関数内で、g_timer1_counter は 5 カウント(500us)毎にスイッチの読み取りに使用。g_timer2_counter は、10 カウント(0.1s)毎に目標回転数の更新に使用。g_timer3_counter は、10 カウント(5s)毎に、画面表示に使用。

g_debug_print_flag

概要:デバッグ表示フラグ変数
宣言:volatile unsigned short g_debug_print_flag
説明:
　BLM_DEBUG_LEVEL_1(0x1)　簡易デバッグ表示（duty の増加、減少）
　BLM_DEBUG_LEVEL_2(0x2)　ホールセンサ切り替わり時の角度表示
　BLM_DEBUG_LEVEL_3(0x4) duty 値の表示
　BLM_DEBUG_LEVEL_4(0x8)　ホールセンサ値の表示
　BLM_DEBUG_LEVEL_5(0x10)　相電圧と相電圧の平均値の表示
g_debug_print_flag = BLM_DEBUG_LEVEL_1 | BLM_DEBUG_LEVEL_2;
で、LEVEL1 と LEVEL2 のデバッグ表示を有効化する。

1.8. プログラムの動作を制御する定義値

```
#define PI 3.14159265358979f
```

円周率(π)の値

```
#define SQRT3_DIV2 221          //√3/2×256
#define SQRT3 443               //√3×256
#define MSQRT3 -443             //-√3×256
#define N2_DIV_SQRT3 295        //2/√3×256
```

√3 /2, √3, -√3, 2/√3 の値。プログラム内では、整数演算で計算するため、256 倍した値を使用。

```
#define OFF_DIRECTION 0
#define U_V_DIRECTION 1
#define U_W_DIRECTION 2
#define V_W_DIRECTION 3
#define V_U_DIRECTION 4
#define W_U_DIRECTION 5
#define W_V_DIRECTION 6
```

モータ制御の電流方向を定義する定数。120 度制御時に使用。

```
#define BLM_DRIVE_6_PHASE_PWM  0        //相補PWM
#define BLM_DRIVE_120_DEG  1            //120 度制御
```

モータ制御方式の定数。相補 PWM（6 相 PWM)と 120 度制御。

```
#define BLM_CH_1        0
#define BLM_CH_2        1
#define BLM_CH_NUM    2
```

ch 番号と、ch 数の定義。本キットでは、2ch のサポート。

```
#define BLM_CH_STATE_INACTIVE     0     //モータは非動作状態
#define BLM_CH_STATE_ACTIVE       1     //モータは動作状態
```

スイッチ(SW1/SW2)の ON/OFF で決まる、モータが回転制御状態かどうかを示す定数。

```
#define BLM_NO_ERROR                            0     //エラーなし
#define BLM_ERROR_OVER_TEMP_STOP                0x0001    //過熱停止
#define BLM_ERROR_OVER_CURRENT_STOP_1   0x0002    (*1)
```

```
#define BLM_ERROR_OVER_CURRENT_STOP_2    0x0004     (*2)
#define BLM_ERROR_OVER_CURRENT_STOP_3    0x0008     (*3)
#define BLM_ERROR_DISPLAY_STOP                  0x0010    //画面表示ため停止
```

(*1)　//割り込みを使い 1 回でも過電流が観測されると停止
(*2)　//10ms の間 BLM_OVER_CURRENT_COUNT_10MS 回数を超えたら停止
(*3)　//1s の間 BLM_OVER_CURRENT_COUNT_1S 回数を超えたら停止

エラーステータスを示す定数。

```
#define BLM_NO_CONNECT 0     //モータドライバボード未接続
#define BLM_CONNECT      1     //モータドライバボード接続
```

モータドライバボードの接続状況を示す定数。

```
#define BLM_OVER_CURRENT_COUNT_10MS        90
      //100us 毎にチェックを行い 10ms あたり 90 回以上過電流検出で停止（最大 100）
#define BLM_OVER_CURRENT_COUNT_1S          1000
      //100us 毎にチェックを行い 1s あたり 1000 回以上過電流検出で停止（最大 10,000）
```

過電流検出の閾値。

BLM_ERROR_OVER_CURRENT_STOP2 有効時 BLM_OVER_CURRENT_COUNT_10MS の値が使用され、BLM_ERROR_OVER_CURRENT_STOP3 有効時 BLM_OVER_CURRENT_COUNT_1S の値が使用されます。

100us 間隔で 10ms 間に 100 回の過電流チェックが行われるが、BLM_OVER_CURRENT_COUNT_10MS を 90 にした場合、10ms 間に 90%(90/100)以上の過電流が検出された場合に停止となります。

```
#define BLM_OVER_TEMP     50
```

過熱停止の閾値。定義値 50 の場合は、モータドライバボード上の温度センサ（サーミスタ）が 50℃を超えた時に、モータは停止となります。

```
#define BLM_ADC_TERMINALS     14
```

A/D 変換端子は、14 端子。

```
#define BLM_ADC_BIT     12
```

A/D 変換のビット数は、12bit とする。

A/D 変換対象の端子数の設定

```
#define BLM_ADC_HIST       10        //10 ポイント分を保存
```

(2bytes x 9 x 2ch x 10=0.36kB)..RL78G24 は RAM12kB（※RL78G24 の RAM がパンクしない範囲で設定してください, FAA を有効にすると FAA で 6kB のメモリを消費しますので、残りは 6kB です）
※cos, sin, tan, √3/tan のテーブルデータを RAM 上に配置（サンプルプログラムでのデフォルト）している場合は、使用可能な RAM がほぼ残りません

A/D 変換の履歴を保存する数。

```
#define BLM_ADC_FLAG_COPY_STOP     0x0001  //表示用履歴変数の更新を止めるフラグ
```

A/D 変換を一時停止するフラグ変数

```
#define BLM_ADC_LONG_AVERAGE    512
```

相電圧の長周期の平均の計算を 512 点で行う。（256, 512, 1024, 2048, 4096 の値が設定可能）

```
#define BLM_ADC_LONG_AVERAGE_HIST    8
```

相電圧の長周期の平均の計算を 1024 点の値の 8 点の移動平均を取り、最終的な平均値を求める。（2, 4, 8, 16, 32 の値が設定可能）

```
#define BLM_ADC_SHORT_AVERAGE_HIST    4
```

相電圧の短周期の平均を 4 点の移動平均で求める。（2, 4, 8, 16, 32 の値が設定可能）

```
#define BLM_DUTY_MAX                  100
#define BLM_DUTY_MIN                  0
```

duty 比の設定値の最大、最小。%単位、整数での指定。

```
#define BLM_DUTY_DIFF_1               32
      //0.05%, ANGLE_DIFF_THRESHOLD1  のずれを検出した際の duty 増分値   0.05%×65536
#define BLM_DUTY_DIFF_2               131
      //0.2%, ANGLE_DIFF_THRESHOLD2  のずれを検出した際の duty 増分値   0.2%×65536
```

通常制御において、ホールセンサ切り替わり時、15°(ANGLE_DIFF_THRESHOLD_1)以上のずれがあった場合は、duty に 32(BLM_DUTY_DIFF_1)を加減算。30°(ANGLE_DIFF_THRESHOLD_2)以上のずれがあった場合は、duty に 131(BLM_DUTY_DIFF_2)を加減算。

```
#define BLM_DUTY_FEEDBACK_RATE    0.01f      //1%
```

通常制御において、回転数が目標回転数から大きく離れている(5%以上)場合、duty を 1%(BLM_DUTY_FEEDBACK_RATE)ずつ近づけていく。

```
#define BLM_PHASE_0    0     //停止
#define BLM_PHASE_1    1     //始動制御
#define BLM_PHASE_2    2     //通常制御
#define BLM_PHASE_3    3     //ブレーキ
```

動作フェーズを示す定数。

```
#define BLM_HALL_MOTOR    1
#define BLM_HALL_PSEUDO 2
```

ホールセンサとして、モータ内蔵のホールセンサを使用する(1)か、疑似ホールセンサパターンを使用する(2)かを定義する定数。

```
#define BLM_HALL_PSEUDO_SENSOR_AVERAGE    0x1
```

疑似ホールセンサパターンを使用する場合に、短期間の平均値を使用するフラグ。

```
#define BLM_HALL_PSEUDO_SENSOR_HYS    0x2
```

疑似ホールセンサパターンを使用する場合に、ヒステリシスを有効にするフラグ。

```
#define BLM_HALL_PSEUDO_SENSOR_HYS_VAL    16
      //16 = 20mV/5000mV*4096, 20mV 程度ヒステリシスを付ける
```

ヒステリシスを付ける場合の、ヒステリシス値。上記の場合は、平均値+20mV で、0→1 に変化し、平均値-20mV で、1→0 に変化する。

```
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_0     0
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_1     10
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_2     20
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_3     30
```

```
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_ANGLE_4     40
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_RPM_1       2000      //2000[rpm]までは 0°
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_RPM_2       6000      //6000[rpm]までは 10°
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_RPM_3       8000      //8000[rpm]までは 20°
#define BLM_HALL_PSEUDO_SENSOR_OFFSET_RPM_4       10000     //1000[rpm]までは 30°, それ
以上は 40°
```

疑似ホールセンサパターンを使用する場合、回転速度に応じて磁界印加角度に定義値のオフセットを付ける設定。

```
#define BLM_CCW        1    //反時計回り
#define BLM_CW         -1   //時計回り
#define BLM_STOP       0    //停止
#define BLM_BRAKE      2    //ブレーキ
#define BLM_UNKNOWN -127    //不明（異常）
```

モータの回転方向を定義する定数。

```
#define BLM_MAX_RPM        12000    //上限
#define BLM_MIN_RPM        1200     //下限
#define BLM_MIN_RPM2       1800     //下限
```

最高、最低回転数。VR ツマミを回した際の目標回転数の上限と下限。相補 PWM 時は、1200 を下限として、120 度制御時は 1800 を下限とする。

```
#define BLM_START_RPM                 1200    //始動回転数
#define BLM_START_RPM_LOWER           800     //始動制御時の最低回転数
#define BLM_START_RPM_UPPER           1500    //始動制御時の最高回転数
#define BLM_START_RPM_OVER            4000    //始動制御時の異常と判断する回転数
```

始動制御（フェーズ=1）時の固定回転数 1200rpm(BLM_START_RPM)。及び、始動制御における、この値未満は回転が安定していないとみなす最低回転数 800rpm(BLM_START_RPM_LOWER)。始動制御で、この回転数を越えた場合 duty を減らす最高回転数 1500rpm(BLM_START_RPM_UPPER) の設定値。また、4000rpm(BLM_START_RPM_OVER)を超えた場合は、ホールセンサの切り替わりが上手く検出できていないと判断する。

```
#define BLM_DIRECTION_STABLE_THRESHOLD1   6          //1 回転分
#define BLM_DIRECTION_STABLE_THRESHOLD2   24
      //1200rpm の場合 0.2s 以上回転安定時 PHASE2 に移行
#define BLM_DIRECTION_STABLE_THRESHOLD3   30
      //回転安定検出の最大値
```

始動制御時、回転安定が 6(BLM_DIRECTION_STABLE_THRESHOLD1)に達していない場合、duty を増やします。同じく、回転安定が 24(BLM_DIRECTION_STABLE_THRESHOLD2)を超えた場合、通常制御（フェーズ 2）に移行します。回転安定の最大値が、BLM_DIRECTION_STABLE_THRESHOLD3=30 です。回転安定が 30 を超えている場合、回転安定カウントのインクリメントは行いません。

```
#define BLM_RPM_FEEDBACK_RATE     0.20f                  //20%
```

回転数のフィードバック係数。目標回転数と制御回転数が乖離している場合、この定数の割合で制御回転数を目標回転数に近づけていきます。

```
#define BLM_ANGLE_MULTIPLIER     64
#define BLM_ANGLE_MULTIPLIER_SHIFT     6          //2^6=64
```

角度の取り扱い倍率。g_angle は、64 で 1°を表す。

```
#define BLM_FORWARD_ANGLE_MAX     45
#define BLM_FORWARD_ANGLE_MIN     -45
```

進角調整範囲、単位°-45~45°の範囲で進角調整が有効です。

```
#define DEGREE_0 (0)
#define DEGREE_15 (15 * BLM_ANGLE_MULTIPLIER)
…
#define DEGREE_360 (360 * BLM_ANGLE_MULTIPLIER)
```

角度に、角度取扱係数（BLM_ANGLE_MULTIPLIER(=64)）を掛けた定数です。

```
#define BLM_ANGLE_DIFF_THRESHOLD_1     DEGREE_15
    //15 度以上ずれている場合は、duty の微調整を行う
#define BLM_ANGLE_DIFF_THRESHOLD_2     DEGREE_30
    //30 度以上ずれている場合は、duty の調整を行う
```

通常制御時、15°(BLM_ANGLE_DIFF_THRESHOLD_1)以上理想角度と現在の印加角度がずれた場合、duty の微調整を行い、30°(BLM_ANGLE_DIFF_THRESHOLD_2)以上理想角度と現在の印加角度がずれた場合、duty の調整を行う。

```
#define BLM_CONTROL_PERIOD    100.0e-6f
```

基本制御周期、100us。

#define BLM_USE_FAA　　0
#define BLM_NOT_USE_FAA　　1

FAA を使用するか否かを設定する定数。

#define BLM_PORT_DEBUG

定義時、ポートデバッグを有効にする。

#define BLM_DEBUG_PRINT_1　//LEVEL1 デバッグ表示
#define BLM_DEBUG_PRINT_2　// LEVEL2 デバッグ表示
#define BLM_DEBUG_PRINT_3　// LEVEL3 デバッグ表示
#define BLM_DEBUG_PRINT_4　// LEVEL4 デバッグ表示
#define BLM_DEBUG_PRINT_5　// LEVEL5 デバッグ表示

定義時デバッグ表示をフラグ変数によって制御。未定義時、条件コンパイルでデバッグ表示を削除。

#define BLM_DEBUG_LEVEL_1　0x1　//LEVEL1 デバッグ表示フラグ
#define BLM_DEBUG_LEVEL _2　0x2　// LEVEL2 デバッグ表示フラグ
#define BLM_DEBUG_LEVEL _3　0x4　// LEVEL3 デバッグ表示フラグ
#define BLM_DEBUG_LEVEL _4　0x8　// LEVEL4 デバッグ表示フラグ
#define BLM_DEBUG_LEVEL _5　0x10　// LEVEL5 デバッグ表示フラグ

BLM_DEBUG_PRINT_1, BLM_DEBUG_PRINT_2 が定義されており、
フラグ変数、g_debug_print_flag = BLM_DEBUG_LEVEL_1 | BLM_DEBUG_LEVEL_2;
の場合、LEVEL1, LEVEL2 のデバッグ表示が有効となる

//#define BLM_DEBUG_PRINT_2
g_debug_print_flag = BLM_DEBUG_LEVEL_1 | BLM_DEBUG_LEVEL_2;
の場合（BLM_DEBUG_PRINT_2 が未定義）、LEVEL2 のデバッグ表示は無効となる。

フラグ変数(g_debug_print_flag)は、任意のタイミングで変更可能であるが、BLM_DEBUG_PRINT_2 を未定義とした場合、フラグ変数をチェックするプログラムコードが無効化される。デバッグ出力という余計な処理にリソースを食われたくない場合に、BLM_DEBUG_PRINT_?を未定義としてください。

1.9. プログラムで使用している機能と割り込み

1.9.1. プログラムで使用しているマイコン機能

コンポーネント名	機能名	用途・備考
r_bsp	共通/クロック発生回路	初期状態で追加済み
Config_INTC	割り込み	過電流停止で使用、立下りエッジ(*1)
Config_ITL000_ITL001	タイマ（インターバル）	10ms タイマ
Config_ITL012_ITL013	タイマ（インターバル）	500ms タイマ
Config_TAU0_0	タイマ（インターバル）	100us タイマ
Config_TAU0_2	タイマ(PWM)	120 度制御の PWM 波形出力に使用(CH-2)
Config_TDR0_TDR1	タイマ RD(PWM)	相補 PWM 波形出力(CH-1)
Config_TRJ0	タイマ RJ（インターバル）	120 度制御の始動制御 6ms タイマ(CH-2)
Config_ADC	A/D 変換	
Config_PORT	ポート機能	SW, LED の動作，モータ制御端子の制御
Config_UART0	シリアル・アレイ・ユニット	UART 通信
(Config_UART2)	シリアル・アレイ・ユニット	UART 通信（COM ポートデバッグ時に使用）
Config_FAA	フレキシブル・アプリケーション・アクセラレータ	相補 PWM の UVW 分解の計算に使用(*2)

(*1)設定に応じて使用、C コマンドで使用・未使用を切り替え

(*2)f コマンドで使用・未使用を切り替え

1.9.2. 使用端子

端子名	役割	割り当て	備考
P00	QU(CH-2)	出力	
P01-P03	A/D 変換	ANI30,ANI16,ANI17	
P05	HS3(CH-1)	入力（プルアップ）	ホールセンサ入力端子として使用
P06	HS2(CH-1)	入力（プルアップ）	ホールセンサ入力端子として使用
P10-P15	Q1U~Q3L(CH-1)	周辺機能(タイマ RD)	相補 PWM 出力端子に設定
P16	QU(CH-1)	出力	
P17	QL(CH-1)	出力	
P20-P27	A/D 変換	ANI0-ANI7	
P31	QL(CH-2)	周辺機能(TAU0)	PWM 出力端子に設定
P40	デバッグ	TOOL0	E2Lite, E2 接続時に使用
P41	SW1	入力	SW を ON 側に倒した際 L，外部でプルアップ
P42	SW2	入力	SW を ON 側に倒した際 L，外部でプルアップ
P43	SW3	入力	SW を ON 側に倒した際 L，外部でプルアップ
P50	UART 通信（受信）	周辺機能(RXD0)	COM ポートデバッグ時は TOOLRXD に割り当て
P51	UART 通信（送信）	周辺機能(TXD0)	COM ポートデバッグ時は TOOLTXD に割り当て
P52	HS1(CH-2)	入力（プルアップ）	ホールセンサ入力端子として使用
P53	HS2(CH-2)	入力（プルアップ）	ホールセンサ入力端子として使用
P54	HS3(CH-2)	入力（プルアップ）	ホールセンサ入力端子として使用
P55	*INT(CH-2)	割り込み(INTP4)	立下りエッジ
P60	LED1	出力（初期値 H）	初期状態で LED は消灯
P61	LED2	出力（初期値 H）	初期状態で LED は消灯
P62	LED3	出力（初期値 H）	初期状態で LED は消灯
P63	LED4	出力（初期値 H）	初期状態で LED は消灯
P70-P75	Q1U~Q3L(CH-2)	出力	
(P76)	UART 通信（受信）	周辺機能(RXD2)	COM ポートデバッグ時に使用
(P77)	UART 通信（送信）	周辺機能(TXD2)	COM ポートデバッグ時に使用
P120	A/D 変換	ANI19	

P130	LED	出力	マイコンボード上の LED2
P137	プッシュ SW	入力	マイコンボード上の SW2
P140	HS1(CH-1)	入力（プルアップ）	ホールセンサ入力端子として使用
P141	*INT(CH-1)	割り込み(INTP7)	立下りエッジ
P146,P147	A/D 変換	ANI28,ANI18	

1.9.3. プログラムで使用している割り込み

機能名	分類	割り込みレベル	備考
INTP4	端子割り込み	0	過電流停止(CH-2)
INTP7	端子割り込み	0	過電流停止(CH-1)
INTAD0	A/D 変換	1	A/D 変換結果の処理
INTTM00	タイマ	2	100us の定期処理
INTITL	タイマ	3	10ms の定期処理 500ms の定期処理
INTST0 INTSR0 INTSRE0	通信	3	画面表示、コマンド入力

※割り込みレベル

0:最優度高い

3:優先度低い

基本的に多重割り込みを有効化しているので、優先度が n の割り込み処理実行中に、優先度 m, (m>n)の割り込みが掛ると、m の割り込みが先に処理されます。

1.10.デバッグ補助機能

モータの制御においては、デバッガを使用したデバッグが難しい場合があります。ブレークポイントを張って、プログラムを停止した場合、「磁界印加角度のインクリメント動作が止まりモータが回らない」等の状態となってしまうためです。

また、出力端子がアクティブな状態（U 相から V 相に電流を流している状態等）でプログラムが停止すると、過大な電流が流れて、モータドライバボードが破損します。（モータは PWM 駆動等、適切なタイミングで電流を ON/OFF しながら駆動する必要があります。モータ内部はコイル（インダクタンス）で構成されていますので、電流が流れる状態を維持すると（＝DC 的に電流を印加すると）、過大な電流が流れます。…インダクタンスは周波数に比例したインピーダンスを持つため、DC 的には理想的なインダクタンスのインピーダンスは 0 となります。…実際はコイルの配線抵抗もありますので、0 ではないですが。）
（マイコンのタイマ機能を使用して PWM 波形を生成している場合等、デバッガでプログラムの動作を止めても波形出力が継続されるケースもあります。）

モータのデバッグに関しては、デバッガで安易にブレークできない（ケースが多い）事に留意ください。

本サンプルプログラムでは、デバッグの補助機能として、
・UART による情報表示
・端子を使ったデバッグ
を用意しています。

1.10.1.UART を使用した情報表示

UART では、5 秒に 1 回、回転数や duty、温度等の情報を表示しています。

blm¥blm_main.c 内で

```
const unsigned long information_display_interval = (unsigned long)(5.0 / 500.0e-3);//5 秒毎に画面に情報を表示（500ms:ITL012_ITL013 で何カウントか）
```

表示更新タイミングを設定していますので、5.0 の値を変更すれば、表示タイミングの変更が可能です。キーボードから s コマンドを入力すると、表示を止める事が可能です。（もう一度 s を入力すると表示は再開される。）

blm¥blm.h 内で

```
//デバッグ表示
#define BLM_DEBUG_PRINT_1 //定義時デバッグ情報(LEVEL1)を出力を可能とする
#define BLM_DEBUG_PRINT_2 //定義時デバッグ情報(LEVEL2)を出力を可能とする
#define BLM_DEBUG_PRINT_3 //定義時デバッグ情報(LEVEL3)を出力を可能とする
#define BLM_DEBUG_PRINT_4 //定義時デバッグ情報(LEVEL4)を出力を可能とする
#define BLM_DEBUG_PRINT_5 //定義時デバッグ情報(LEVEL5)を出力を可能とする
```

上記を有効化（デフォルト）しておけば、キーボードからのコマンドにより追加のデバッグ出力が可能になります。

・LEVEL1 のデバッグ出力

キーボードから z コマンドを入力すると、duty の増加・減少の表示。

```
   CH-1 START
d+d+d+d+d+d+d+d+d+d+d+d+d+d+d+d+d+d+d+d+d+d+<<
```

黒字の部分が追加される表示

d+ ：始動制御(BLM_PHASE_1)で duty を増加
d- ：始動制御(BLM_PHASE_1)で duty を減少

```
> > > > > > > > > > > > > > >> > > > >> > > > >> > > > > > > > > > > > > > > > > > > < < < < < < <
< < < < < < < < < < < < < < < < < < < < < < < < < <
```

<< ：通常制御(BLM_PHASE_2)で回転が速すぎるので duty 値を大きく減少
< ：通常制御(BLM_PHASE_2)で回転が速いので duty 値を減少

>> ：通常制御(BLM_PHASE_2)で回転が遅すぎるので duty 値を大きく増加
> ：通常制御(BLM_PHASE_2)で回転が遅いので duty 値を増加

といった情報が表示に追加されます。

・LEVEL2 のデバッグ出力（角度表示は CH-1 のみ、phase 遷移は CH-2 側も表示）

キーボードから x コマンドを入力すると、100us の割り込みルーチンでのホールセンサが切り替わった際の角度の表示。

```
c1,H:6->4@0
c1,H:4->5@75
c1,H:5->1@144
c1,H:1->3@201

**********************************************

c1->phase2
c1,H:6->4@281>330
c1,H:4->5@26>30
c1,H:5->1@88>90
```

c1:*REV*c1,H:0->6@0
設定回転方向と逆回転した場合
（始動時はモータ軸が振動するので逆回転も起こります）

始動制御から通常制御
切り替わり

c1　CH-1 側である事を示す
H:6->4　ホールセンサの値が 6 から 4 に切り替わった
@0　その時の磁界印加角度(0-359°)
@281>330　磁界印加角度をセンサー切り替わり時の理想角度である（330°）に補正（通常制御時に行われる）

・LEVEL3 のデバッグ出力

キーボードから c コマンドを入力すると、回転数と duty の値の表示。

```
  CH-1 START
c1,phase=1,rpm=0/0,duty=4194,stable=0
c1,phase=1,rpm=0/0,duty=6291,stable=0
c1,phase=1,rpm=0/0,duty=7339,stable=0
（中略）
c1,phase=1,rpm=800/213,duty=11956,stable=9
c1,phase=1,rpm=1449/304,duty=12087,stable=10
c1,phase=1,rpm=1428/393,duty=12218,stable=12
c1,phase=1,rpm=1369/479,duty=12349,stable=13
（中略）
c1,phase=1,rpm=1538/1116,duty=12873,stable=22
c1,phase=1,rpm=1587/1215,duty=12873,stable=23
c1,phase=1,rpm=1098/1284,duty=12873,stable=24
c1,phase=2,rpm=1351/1279/1431/3172,duty=12948,stable=26
c1,phase=2,rpm=1298/1286/1438/3172,duty=13023,stable=28
c1,phase=2,rpm=1724/1343/1497/3172,duty=13096,stable=30
（中略）
c1,phase=2,rpm=2941/3245/3235/3235,di=0,duty=17418,stable=30
c1,phase=2,rpm=3030/3219/3235/3235,di=0,duty=17418,stable=30
c1,phase=2,rpm=2941/3180/3235/3235,di=0,duty=17418,stable=30
```

duty を増加させていくが
回転は始まらない

徐々に回転が速くなる
（回転安定判定はされていない）

duty を上げなくても安定して回転
（始動制御を抜ける直前）

通常制御に移行後
目標回転数に近づけていく段階

目標回転数に達し PI 制御移行後

c1　CH-1 側である事を示す
phase=1　始動制御(BLM_PHASE_1)である事を示す
phase=2　通常制御(BLM_PHASE_2)である事を示す
rpm=　:回転数/回転数（平均）　[phase=1 の時]
rpm=　:回転数/回転数（平均）/制御回転数/目標回転数　[phase=2 の時]
duty=　:設定 duty 値×1,000（処理速度向上のため整数値での取り扱い）
stable=　:回転安定数（24 を超えると通常制御に移行）
di=　:duty の累積加算値×10,000（処理速度向上のため整数値での取り扱い）　[phase=2 で PI 制御移行後]
（di=0 の時は duty を加算も減算もしない状態，ホールセンサ切り替わりのタイミングで累積値を加算して、10ms のタイミングで duty 値に反映）（※di 表示は CH-1 側のみ）

・LEVEL4 のデバッグ表示

キーボードから v コマンドを入力すると、ホールセンサの情報の表示。

```
c1,pos:5,4
c1,pos:5,5
c1,pos:1,1
c1,pos:3,3
c1,pos:2,3
c1,pos:2,2
c1,pos:6,6
c1,pos:4,6
c1,pos:4,4
c1,pos:5,5
c1,pos:1,5
c1,pos:1,1
```

n,n　ホールセンサ，疑似ホールセンサ

c1　CH-1 側である事を示す
pos:5,4
　モータ内蔵のホールセンサの値が 5
　UVW の相電圧から算出した疑似ホールセンサパターンの値が 4

デフォルトでは、2ms 毎のホールセンサ値を表示します。

・LEVEL5 のデバッグ表示

キーボードから b コマンドを入力すると、UVW の相電圧の情報の表示。

```
c1:v(ave),1930,1919,1918,1706,1892,2165
c1:v(ave),1930,1919,1918,1785,1789,2195
c1:v(ave),1930,1919,1918,1903,1703,2163
c1:v(ave),1930,1919,1918,2021,1646,2104
```

c1　CH-1 側である事を示す

v(ave) 1930,1919,1918,1706,1892,2165
　U 相電圧の長周期の平均, V 相電圧の長周期の平均, W 相電圧の長周期の平均、U 相電圧の短周期の平均, V 相電圧の短周期の平均, W 相電圧の短周期の平均

※デフォルト設定では、長周期は 1024 点の平均の 8 点の移動平均、短周期は 8 点の移動平均

※b コマンドは CPU 負荷が大きく回転制御に影響を及ぼします

※UART の出力のバッファに関して

　UART 向けに（デフォルトで）512 バイト（文字）のバッファを 2 つ設けています（出力用と格納用）。出力が終わった段階で、出力用と格納用のバッファを差し替えます。（交互に使用します）
　文字出力は、115,200bps なので、1 文字の出力に約 87us 程度掛かります。バッファが溢れた場合、データを捨てる設定としています。

blm¥blm_main.c

```
g_sci_send_nowait_flag = FLAG_SET;//UART の表示が間に合わない場合はデータを捨てる
```

　上記フラグを設定しない場合、バッファが溢れた場合、UART の文字出力が終わりバッファに空きができるまで、プログラムの処理が止まりますので、モータ駆動の場合はフラグを設定する事が推奨です。

※バッファが溢れた場合、データを捨てる設定ですので、出力する情報量が多い場合、途中で表示が切れたようになります。

※バッファの 512 バイトを超える情報を表示させる場合、
g_sci_send_nowait_flag = FLAG_CLEAR;
とすると、バッファに空きが生じるまで、文字出力を待つようになりますので、全ての情報が出力可能ですが、SCI の処理内で空ループが回る動作となります。モータ制御中は基本的には、FLAG_SET とするのが推奨です。
（メモリ容量に空きがあれば、sci.h 内で定義されているバッファ容量(#define SCI_SEND_BUF_SIZE 512)の値を増やすという対応もできます。）
（blm_main.c 内の A/D 変換結果の表示を行う際は、一時的に FLAG_CLEAR を設定しています。）

※RL78G24 は RAM が 12kB ですが FAA 使用時の空きは 6kB なので、あまりバッファ容量を増やす余地はありません

UART の文字表示は、sci¥sci.c に定義されている関数で行っています。

・UART 関連関数

sci_start

SCI の初期化を行います。最初に実行してください。

使用例:
sci_start();

sci_write_str

文字列の表示を行います。

使用例:
sci_write_str("display string¥n");
→表示文字:display string[改行]

sci_write_uint16

数値表示を行います。

使用例:
unsigned short a = 12345;
sci_write_uint16(a);
→表示文字:12345

バリエーション:

関数名	引数	説明
sci_write_uint8	unsigned char	符号なし 8bit 数値表示
sci_write_uint16	unsigned short	符号なし 16bit 数値表示
sci_write_uint32	unsigned long	符号なし 32bit 数値表示
sci_write_int8	char	符号付き 8bit 数値表示（負数の場合のみ-を表示します）
sci_write_int16	short	符号付き 16bit 数値表示（負数の場合のみ-を表示します）
sci_write_int32	long	符号付き 32bit 数値表示（負数の場合のみ-を表示します）

sci_write_uint16_hex

16 進数で数値表示を行います。("0x"等の表示は行いません)

使用例:
unsigned short a = 0x1234;
sci_write_uint16_hex(a);
→表示文字:1234

バリエーション:

関数名	引数	説明
sci_write_uint8_hex	unsigned char	符号なし 8bit hex 表示
sci_write_uint16_hex	unsigned short	符号なし 16bit hex 表示
sci_write_uint32_hex	unsigned long	符号なし 32bit hex 表示

sci_write_flush

出力バッファに溜まっているデータを吐き出させます。(出力バッファが空になるまで、プログラムの実行が止まります。)

使用例:
sci_write_flush();

補足:
長いデータ(バッファ容量の 512 バイト)を超えるデータを出力させる場合、バッファ溢れによりデータが捨てられますので、定期的に本関数を実行するか、
g_sci_send_nowait_flag = FLAG_CLEAR;
として、バッファ溢れ時に出力を待つようにしてください。

また、デバッガでプログラムを停止させた場合、文字出力の処理も停止しますので、プログラムのブレーク前に画面表示を終わらせたい場合は、本関数実行後にブレークを掛けてください。

sci_read_char

キーボードから入力した文字の読み出しを行います。

使用例:

```
unsigned short ret;
unsigned char c;
ret = sci_read_char(&c);
if (ret != SCI_RECEIVE_DATA_EMPTY)
{
   if (c == 's') blm_stop(BLM_CH1);       //キーボードから s が入力された際、CH1 のモータを停止させる
}
```

補足:

入力バッファは(初期設定値で)16 バイト(文字)用意しています。sci_read_char ではバッファに格納されている一番古いデータが取り出せますので、複数の文字(例えば 123 等の数値入力させた場合)を読み出す場合は、関数の戻り値が SCI_RECEIVE_DATA_EMPTY になるまで複数回本関数を呼び出してください。

float2str

float 型の変数から文字列への変換を行います。

使用例:

```
float a = 1.2345;
char buf[20];
float2str(a, 2, buf);
sci_write_str(buf);
```

→表示文字:1.23

引数:

第 1 引数　表示させる数値
第 2 引数　表示させる小数点以下の桁数
第 3 引数　文字列格納バッファ

バリエーション:

関数名	第 1 引数	説明
float2str	float	浮動小数点数(float)の文字列への変換
double2str	double	浮動小数点数(double)の文字列への変換
float2str_eformat	float	浮動小数点数(float)の文字列への変換(e 形式) ※1.23e-3 等
double2str_eformat	double	浮動小数点数(double)の文字列への変換(e 形式) ※1.23e-3 等

1.10.2.端子を使ったデバッグ

モータ制御で特定のアクションが起こった際に UART 表示よりもリアルタイムで情報を出力する用途で使用頂ける機能です。

本機能を有効化する場合、

¥blm¥blm.h

```
//デバッグ用ポート
#define BLM_PORT_DEBUG      //定義時ポートによるデバッグを有効化する
```

上記定義を有効化してください。(デフォルトで有効)

ポート名(基板端子)	当該ポート:L 出力	当該ポート:H 出力	当該ポート:トグル出力
P04(J1-20)	BLM_DEBUG_PORT_1_L	BLM_DEBUG_PORT_1_H	BLM_DEBUG_PORT_1_T
P30(J2-7)	BLM_DEBUG_PORT_2_L	BLM_DEBUG_PORT_2_H	BLM_DEBUG_PORT_2_T
P76(J2-16) (*1)	BLM_DEBUG_PORT_3_L	BLM_DEBUG_PORT_3_H	BLM_DEBUG_PORT_3_T
P77(J1-17) (*1)	BLM_DEBUG_PORT_4_L	BLM_DEBUG_PORT_4_H	BLM_DEBUG_PORT_4_T

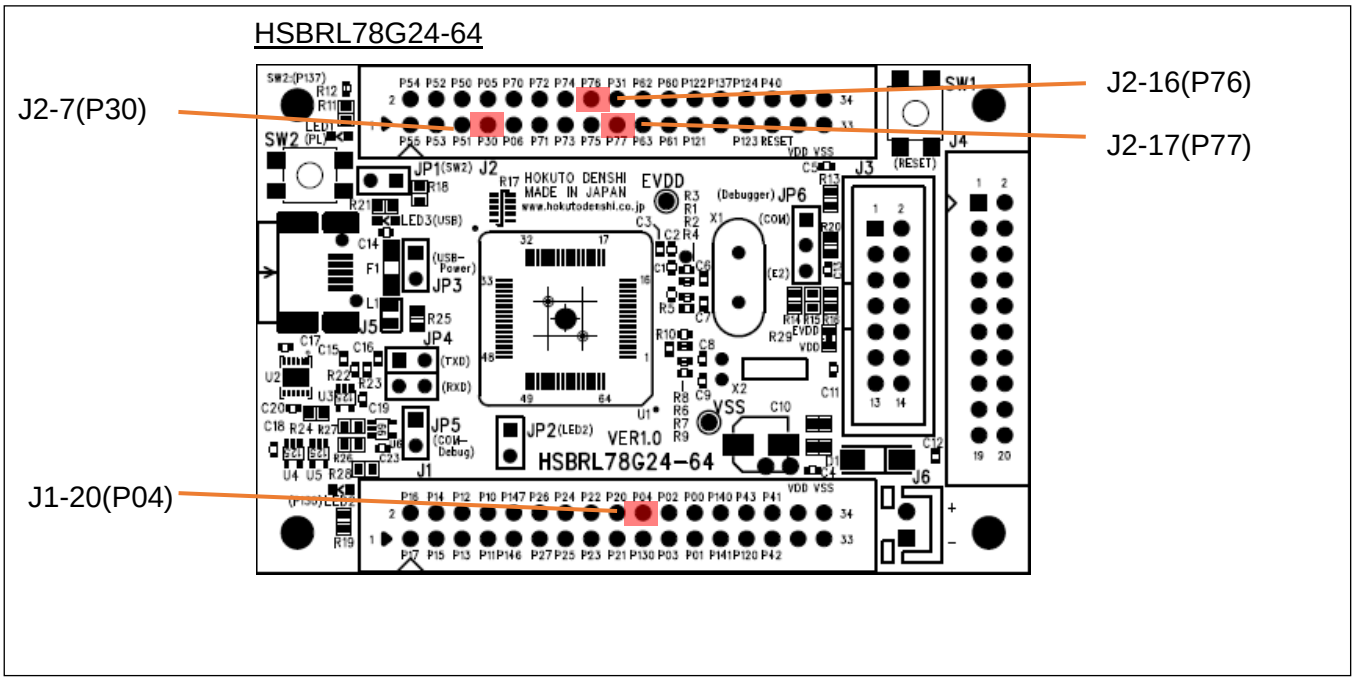
BLM_DEBUG_PORT_1_H
特定の処理
BLM_DEBUG_PORT_1_L

とすると、P04 端子をオシロスコープ等でモニタする事により、特定の処理に掛かる時間を計測可能です。

(*1)P76, P77 は COM ポートデバッグ使用時に UART2 の通信端子に割り当てられますので、COM ポートデバッグ有効時は使用不可です。

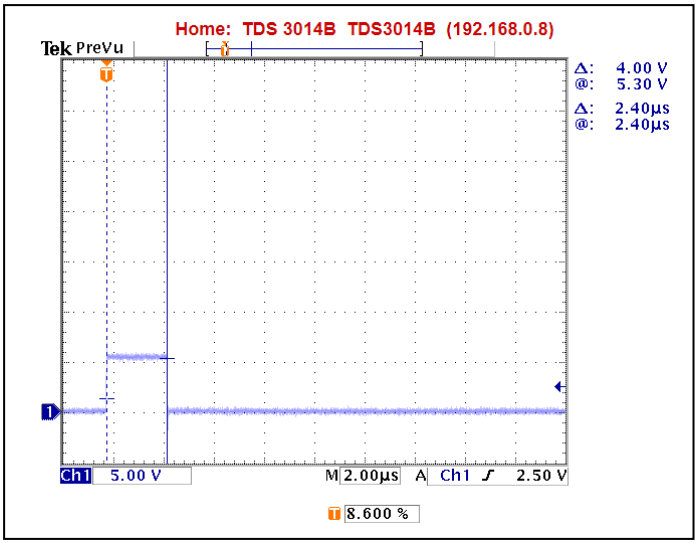
サンプルプログラムでは、以下の設定としています。

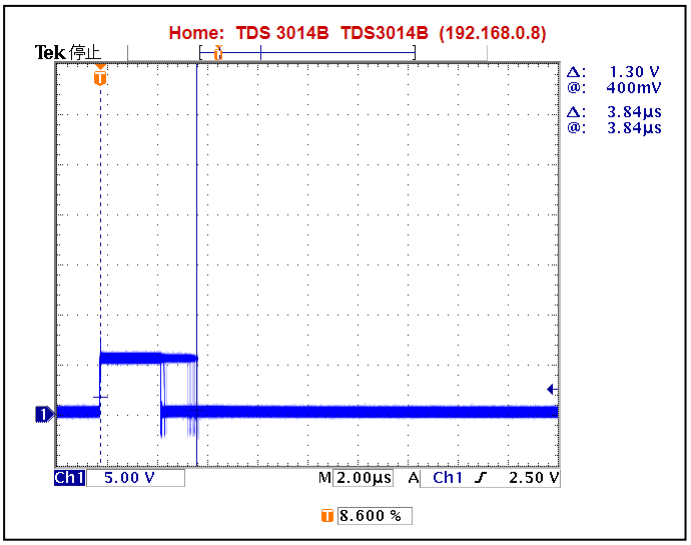
ポート名(基板端子)	観測内容
P04(J1-20)	100us 毎の割り込み処理(TAU0_0)の blm_interrupt_ch1 終了後 H, 終了時に L
P30(J2-7)	10ms 毎の割り込み処理(ITL000_ITL001)の先頭で H, 終了時に L
P76(J2-16)	blm_interrupt_ch1 の先頭で H, 終了時に L
P77(J2-17)	blm_interrupt_ch2 の先頭で H, 終了時に L



端子デバッグ機能を使用する際には、プログラム内に端子を変化させる命令（L 出力、H 出力、トグル出力のいずれか）を埋め込んでおき、オシロスコープ等を使って端子の変化を観測してください。

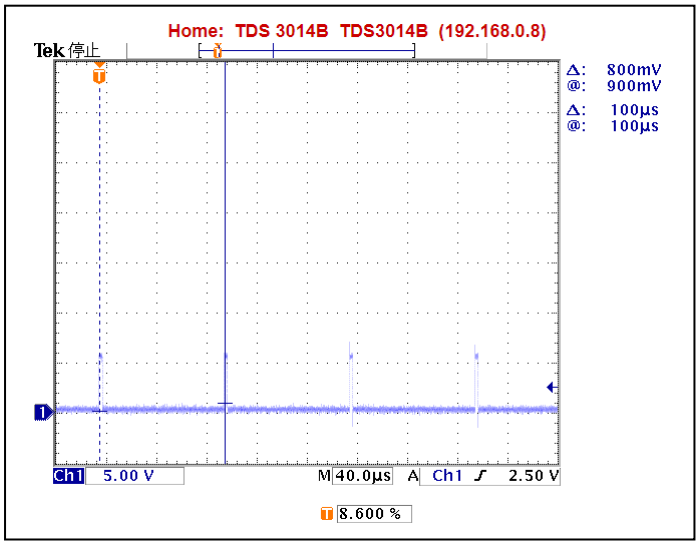
・100us の割り込みモニタ例　　重ね書き





上記は、100us の割り込み処理（ch 毎の処理終了後 H、割り込みの終了で L）をモニタした例です。処理時間は、2.4us 程度である事が判ります。右が波形を重ね書きした場合です。処理時間のばらつきがあり、最大 4us 程度掛かっている事が見て取れます。

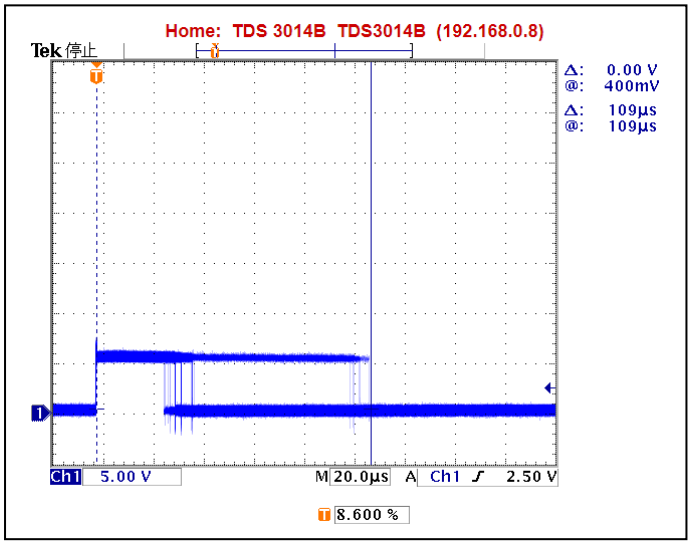
・100us の割り込みを長周期で観測



処理が歯抜け（100us 毎に実行されないタイミングがある）になっていないか、1 回の処理時間に大きなばらつきがないか等の情報が得られます。もし、100us の割り込み処理において、実行時間が長い処理があれば、10ms の定期処理に追い出す等の対応が考えられます。）

（モータが上手く回らない場合は、定期的に割り込みが実行されていないという事も考えられます。端子モニタがデバッグで役に立つ事もあります。）

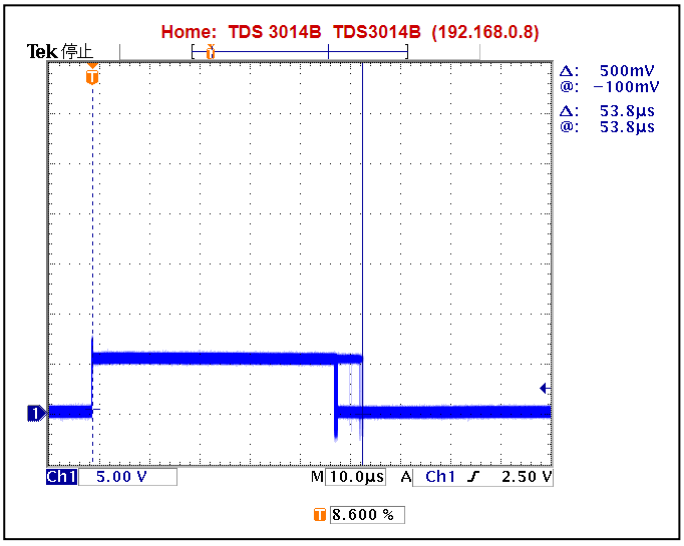
・10ms の割り込みモニタ例　※重ね書き



割り込み処理に掛かる時間は、1 秒毎の処理等もあるので毎回同じではなく、ばらつきは出ます。この波形例では、25us~110us 程度の処理時間となっています。

・CH 毎の処理(blm_interrupt_ch1)（100us 毎）　※重ね書き

(



CH 毎の定期制御（100us 毎）の割り込みを重ね書きした波形例です。1 回の割り込み処理で~54us 程度の処理時間が掛かっています。

相補 PWM の各相の duty 計算（三角関数を使用した計算）は、この関数内で実行されますので、ある程度時間が掛かります。（A/D 変換の割り込みは、100us の割り込みより優先度を高く設定しているので、この時間には A/D 変換の割り込みも含んだ時間です。）

定期的な処理が時間内に終了しているかを確認する手段として、（とても原始的なデバッグ手法となりますが）ポートデバッグは有効です。

1.11.メモリ使用量に関して

DefaultBuild 内に出力されるマップファイル(RL78G24_BLMKIT_SAMPLE.map)を見ると、

*** Total Section Size ***

RAMDATA SECTION:　00001900 Byte(s)
ROMDATA SECTION:　00001e9f Byte(s)
PROGRAM SECTION:　00009076 Byte(s)

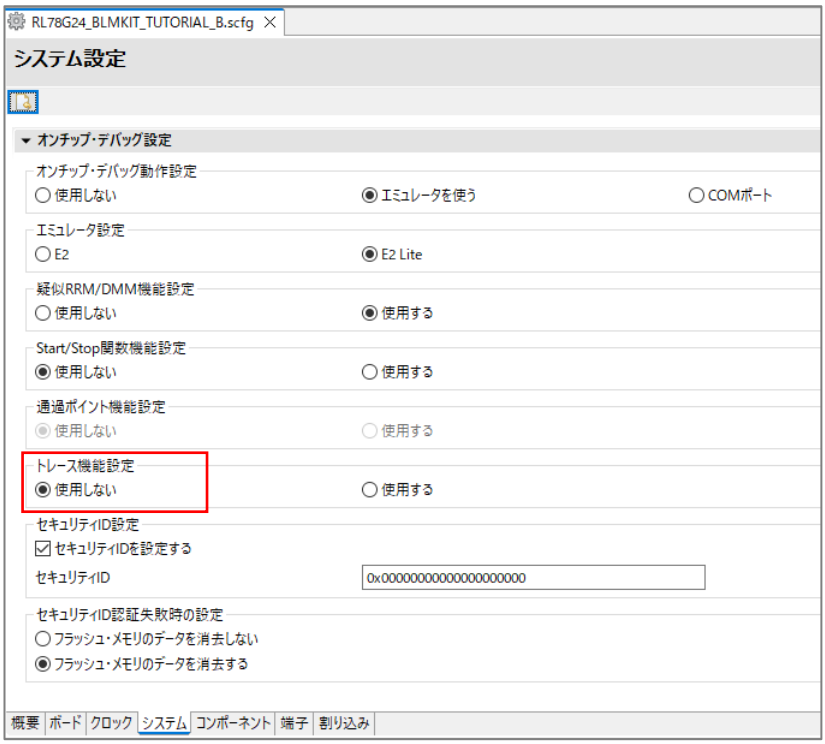
となっています。
ROM データは、合計 AF15H(=44,821bytes)で、マイコンの ROM 容量 128kB に対して、大幅に余裕はあります。
RAM は、1900H(=6,400bytes)で、マイコンの RAM 容量 12kB から見ると、まだ余裕がある様にも見えます。

－RAM の使用量－

セクション名	開始アドレス	終了アドレス	使用サイズ 16 進数（カッコ内 10 進数）	備考
.dataR	FCF00	FD2BD	3BE(958)	初期値付きの変数
.bss	FD2BE	FD7A3	4E6(1254)	初期値なしの変数（UART のバッファ 512bytes　x2 など）
	FD7A4	FD7FF	5C(92)	空き
FAACODER	FD800	FDC9B	49C(1180)	FAA コード(4k)
FAADATAR	FE800	FE87F	80(128)	FAA データ(2k)
RAMUR_n	FF000	FFB3F	b40(2880)	cos や sin のテーブル(2.8kB)
	FFB40	FFE1F	2E0(736)	スタック（自動変数）
	FFE20	FFEDF	C0(192)	ショートアドレッシング領域（未使用）

FAA を使用した場合、FD800H からの 4kB と FE800H からの 2kB の合計 6kB は、FAA で占有する形となります。
RAM の前半（FCF00H~FD7FFH）には、92 バイトの空きしかありません。（A/D 変換の履歴変数もこの RAM 前半に配置されています。）RAM の後半（FF000H~）に、cos や sin のテーブルデータ（720×4、合計 2.8kB）を配置しており、FFE1FH より上の領域はスタック（自動変数）で使用されます。
ショートアドレッシングに対応した、EEF20H~FFEDFH の 192 バイトの領域は未使用なので、何かに使用する事は可能です。
（6kB の FAA 領域の内、使用しているのは 1180+128+スタック数バイトの 1300 バイト強です。この領域に無駄があります。FAA 使用時に、FAA で使用するメモリを自由に調整できれば、もう少し RAM には余裕が生まれるのですが、現状そのような機能はありません。）

スマート・コンフィグレータのシステムの設定では、



オンチップ・デバッグ動作設定で「エミュレータを使う」「COM ポート」を選択して、エミュレータ使用する設定とする場合は、「トレース機能設定」は、「使用しない」を選んで下さい。

トレース機能を使う場合、FD300H~FD6FFH の 1kB がトレース機能で使用されます。（この領域は、ユーザ側で使用しない様、調整する必要があります。）

RAM の使用容量の主たるものに関しては、

用途	変数名	使用容量[bytes]	備考
UART 送信バッファ	g_sci_send_buf	1024	
UART 受信バッファ	g_sci_recv_buf	16	
ADC 履歴	g_adc_result_buf_ch1	180	10 データ分を確保
ADC 履歴	g_adc_result_buf_ch2	180	10 データ分を確保
cos テーブル	g_cos_table	720	ROM に配置可能
sin テーブル	g_sin_table	720	ROM に配置可能(*1)
tan テーブル	g_tan_table	720	ROM に配置可能(*1)
√3/tan テーブル	g_sqrt3_div_tan_table	720	ROM に配置可能(*1)

上記の様なものがあります。

上記の内、三角関数のテーブルデータは参照のみのデータとなりますので、予め計算して、ROM 上に配置する事も可能です。
(*1)UVW 分解法として、別バージョン(1)，別バージョン(2)を使わない場合は不要。
（正弦波変換や 3 倍高調波による UVW 分解の場合は、cos のテーブルデータのみ必要。）
※UVW 分解法に関しては、ソフトウェア編(1)のマニュアルを参照してください。

サンプルプログラムでは、RAM 容量をほぼ限界まで使用していますので、RAM を使う処理を追加する場合は、RAM 使用量の削減が必要になってきます。

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	2025.7.23	—	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:https://www.hokutodenshi.co.jp

ルネサス エレクトロニクス RL78/G24(QFP-64 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 (RL78G24) [ソフトウェア サンプルプログラム編] 取扱説明書

株式会社北斗電子

 2025 年 7 月 23 日改訂 REV.1.0.0.0 (250723)