PILS(PSIM + PE-Expert4)対応
アプリケーションパッケージ

PMモータベクトル制御パッケージ

(PE-APP-PMVEC / PIL)

取　扱　説　明　書

Ver.1.2

2017/05/31

Mywayプラス株式会社

目次

1. はじめに

1.1. 本アプリケーションパッケージの概要

本アプリケーションパッケージは、パワエレに特化した回路シミュレータ PSIM に搭載された PILS(Processor In the Loop Simulation)機能を用いて、実機であるコントローラ部と PSIM 上でモデル化されたプラント部で構成されます。

コントローラ部には PE-Expert4 で使用している DSP ボード(C6657)を使用し、プラント部には永久磁石同期モータ(PMSM)をモデル化した PSIM 上の回路を使用します。

PILS 機能を用いることにより、プラント部のハードウェアを用意することなくコントローラ部の実機デバッグ環境を構築できます。

本アプリケーションパッケージは、"PM モータベクトル制御パッケージ(PE-APP-PMVEC / C6657)"を PILS に移植したものとなっています。

ハードウェアシステムや、制御ブロック等の詳細は、PE-APP-PMVEC / C6657(PJ151241)の関連ドキュメントを参照願います。

1) 本アプリケーションパッケージは、PSIMとPE-Expert4を利用しPMSMのセンサ付きベクトル制御を模擬します。
2) 本モデルを構成する各ハードウェアモジュールは、PJ151241で使用されているものをPSIM向けに最適化されたPSIMのサブサーキットとして実装されています。今回作成した各サブサーキットのほとんどは当社製の標準製品であるため、当社製標準製品を利用するシステムのモデル化を行う場合、容易にシステム構成を行うことができます。
3) 利用しているPEOS関数の詳細は、ファンクションリファレンスマニュアルを参照してください。
4) 制御プログラムは、PIL_ENABLEマクロ定義により、PILS用と、実プラント用に簡単に切り替えることができます。

1.2. 動作環境

本アプリケーションパッケージは以下の弊社製品と組み合わせて使用することを前提に作成しております。

これ以外の組み合わせでの動作は保証できませんのでご注意ください。

[本アプリケーションパッケージに含まれるもの]

- 本説明書
- ソフトウェア CD-ROM

[本アプリケーションパッケージ以外に必要になるもの]

- PE-Expert4 DSP ボード　　MWPE4-C6657
- PE-Expert4 シリーズ専用ラック　MWPE4-RACK12
- 統合開発環境　PE-ViewX　　MWPE-VIEW-X
- PE-Expert4 専用ライブラリ　　MWPE-EXP4-LIB
- メディアコンバーター式
- PSIM Ver11 CD-ROM
- JTAG I/F　　TMDSEMU200-U

1.3. 関連マニュアル

開発環境及び制御ボード・各関数等の機能につきましては以下の個別マニュアルを参照願います。

1) PE-Expert4システムハードウェア　各マニュアル
2) PE-ViewX　各マニュアル
3) PE-Expert4専用ライブラリ　ファンクションリファレンスマニュアル
4) PSIMユーザーズガイド
5) PE-APP-PMVEC / C6657　151241-A2-001

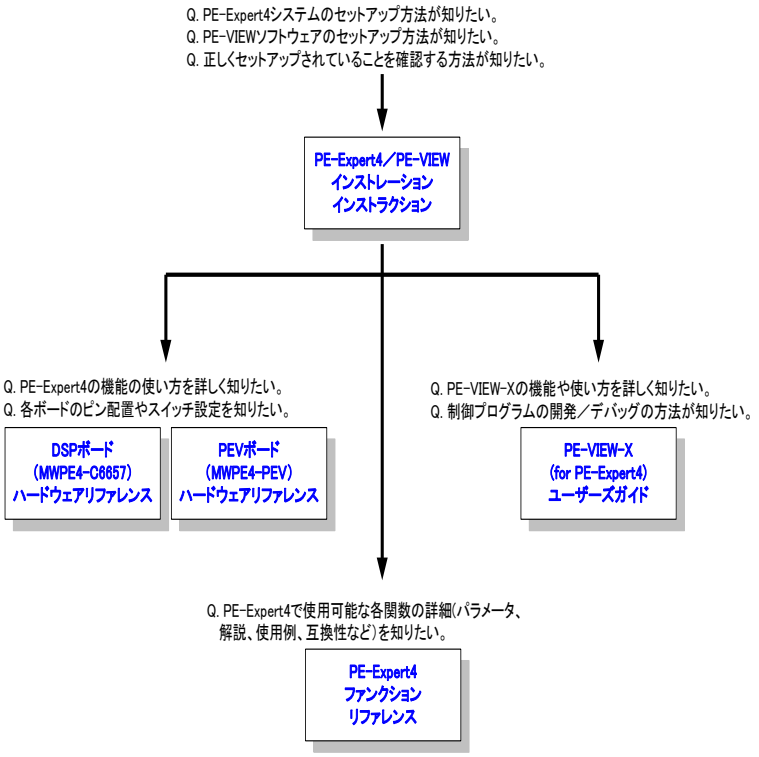


図 1　PE-Expert4 システムのマニュアル構成

2. 準備

2.1. ハードウェアの準備

2.1.1. 接続作業

本アプリケーションパッケージのハードウェア構成は以下のようになります。

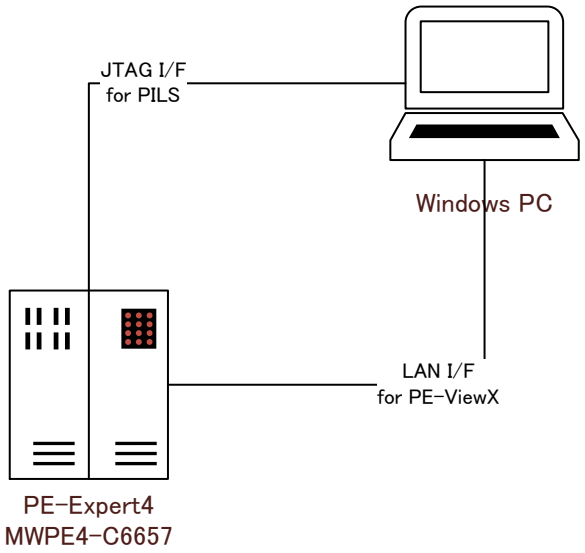


図 2.1　ハードウェアの配線概略図

本アプリケーションパッケージを使用する場合、JTAG I/F は必須となります。

2.2. パソコンと制御ボードの接続

パソコンで制御ソフトウェアのダウンロード作業を行うためには、パソコンと制御ボードを接続する必要があります。「PE-VIEW-X インストレーションインストラクション」に従って接続作業を行ってください。

2.3. ソフトウェアのインストール

本アプリケーションパッケージを使用するためにはあらかじめ PSIM、C コンパイラ、PE-ViewX、および PE-Expert4 専用ライブラリがセットアップされている必要があります。

「PSIM インストールマニュアル」、「PE-ViewX インストレーションインストラクション」に従ってインストールしてください。

2.4. USB JTAG ケーブルのインストール

必要に応じて USB JTAG ケーブルのアップグレードを行う必要があります。下記手順を参照の上、アップグレード作業を行ってください。

http://processors.wiki.ti.com/index.php/XDS200

2.5. PILS 対応制御ソフトの構成方法

PIL モジュールを使用するためには以下の作業を行います。

2.5.1. C ソースコードへの追記

作業内容	目的	備考
先頭行での PIL_ENABLE マクロ定義	実機環境と、PILS 環境の切替のため。	PMVEC_EXP4C6657_1.c(L1)参照
PEV ボードのラップ関数導入	PEV ボード用 I/F ラップ関数の取り込み。	PMVEC_EXP4C6657_1.c(L61-62)参照
PILS 同期周期定義	PSIM と DSP の同期周期を定義します。	PMVEC_EXP4C6657_1.c(L64)参照 本同期周期は実機環境で使用する最短周期割り込みの半分を目安に設定してください。
PILS 同期ハンドラの実装	PSIM と DSP 間で I/F 変数の更新を行うポイントを設けます。また、I/F 変数の更新中にタイマが進むのを抑制するためタイマ停止･再開の記述を行います。	PMVEC_EXP4C6657_1.c(L314-339,741-747)参照
PEV 割込み関数実装	PILS 環境における通常関数呼び出し対応	PMVEC_EXP4C6657_1.c(L482-484,575-577)参照
PILS 同期割り込みの多重化	割込み処理中でも同期割り込みが受理されるようにします。	PMVEC_EXP4C6657_1.c(L596-598)参照

2.5.2. サポートライブラリの追加

下図に挙げた例のようにオブジェクトファイル/ライブラリファイルを構成してください。

$(PEOS_PATH)¥lib¥Main.obj が構成されている場合は削除してください。

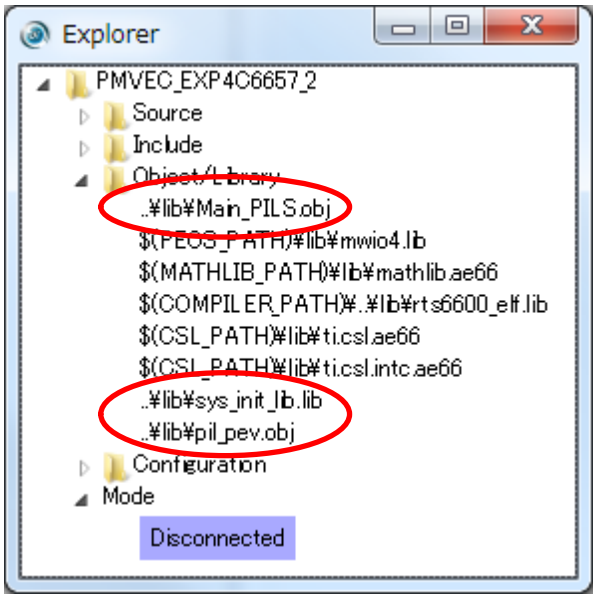


図 3.2　プロジェクトエクスプローラによる表示

2.5.3. Code Composer Studio(CCS)への参照パスの追加

PIL モジュールは、CCS の JTAG デバッグ機能を利用しています。この機能を参照するために[オプション]-[パス設定]を選択し、PSIM 検索パスに CCS への参照パスを追加して下さい。

CCSv5 を標準インストールしている場合は"C:¥ti¥ccsv5"を追加します。

2.5.4. PIL モジュールの設定

PIL モジュールを使用する場合、対象制御プログラムや入出力データ、同期周波数、同期ポイント(ブレークポイント)等を指定する必要があります。

以下に本サンプルで使用されている主な入出力変数の一覧を示します。

IN/OUT	変数名	型	範囲	用途
IN	_IF_PIL_PEV_abz	INT32	0～4095	エンコーダカウンタ
	_IF_PIL_PEV_pio_in	INT32	b0:RUN/STOP b1:Reset	16bit DI スイッチ入力
	_IF_PIL_PEV_ad_raw_0	INT32	-8192～8191	14bitAD
	_IF_PIL_PEV_ad_raw_1			
	...			
	_IF_PIL_PEV_ad_raw_7			
	_IF_PIL_PEV_inverter_int_evnt	INT32	0/1	キャリア割込み
OUT	_IF_PIL_PEV_abz_reset	INT32	0/1	エンコーダリセット
	_IF_PIL_PEV_abz_maxcount	INT32	4096	エンコーダ最大値
	_IF_PIL_PEV_inverter_enable	INT32	0/1	ゲート出力可
	_IF_PIL_PEV_ad_mode	INT32	0～3	AD 変換モード
	_IF_PIL_PEV_inverter_u	INT32	0000H～FFFFH 変調率 0 のとき、8000H	U 相変調率
	_IF_PIL_PEV_inverter_v			V 相変調率
	_IF_PIL_PEV_inverter_w			W 相変調率
	_IF_PIL_PEV_pio_out	INT32	b0:RUN/IDLE b3:過電流 b4:過電圧	16bit DO 状態出力

3. PMSM のセンサ付きベクトル制御

3.1. PMSM のベクトル制御の基礎

ここでは PMSM の一般的なベクトル制御の方法について簡単に紹介します。

詳細な説明は本アプリケーションパッケージに付属する書籍などの参考文献をご覧ください。

PMSM が発生するトルク T は以下の式で表すことができます。

$$T = P\{\phi_m i_q + (L_d - L_q) i_d i_q\} \qquad (3.1)$$

i_d, i_q ：電機子電流の dq 軸成分
L_d, L_q ：電機子巻線インダクタンスの dq 軸成分
ϕ_m ：永久磁石による電機子鎖交磁束
P ：極対数

上式の右辺第一項は q 軸電流に比例する項で、マグネットトルクと呼ばれます。

また右辺第二項は IPMSM のインダクタンスの位置依存性に起因する項で、リラクタンストルクと呼ばれます。

SPMSM の場合、L_d=L_q なのでトルクは q 軸電流に比例します。

また、IPMSM の場合でも L_d=0 に制御した場合、トルクは q 軸電流に比例します。

また、PMSM の dq 回転座標での状態方程式は以下のように表すことができます。

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} r + pL_d & -\omega_e L_q \\ \omega_e L_d & r + pL_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_e \phi_m \end{bmatrix} \qquad (3.2)$$

v_d, v_q ：電機子電圧の dq 軸成分
i_d, i_q ：電機子電流の dq 軸成分
r ：電機子抵抗
L_d, L_q ：電機子巻線インダクタンスの dq 軸成分
p ：微分演算子
ω_e ：回転角速度
ϕ_m ：永久磁石による電機子鎖交磁束

これより、d, q 軸電流は d, q 軸電圧により制御可能なことがわかります。

PMSM のベクトル制御では、これらの関係をもとに、電流ベクトルを調整しトルクの制御を行います。下図はベクトル制御による PMSM の速度制御系の構成例です。

一般に PMSM のベクトル制御では d 軸電流を 0 に制御し、q 軸電流でトルクを制御します。

d 軸電流・q 軸電流はそれぞれ d 軸電圧・q 軸電圧で制御します。

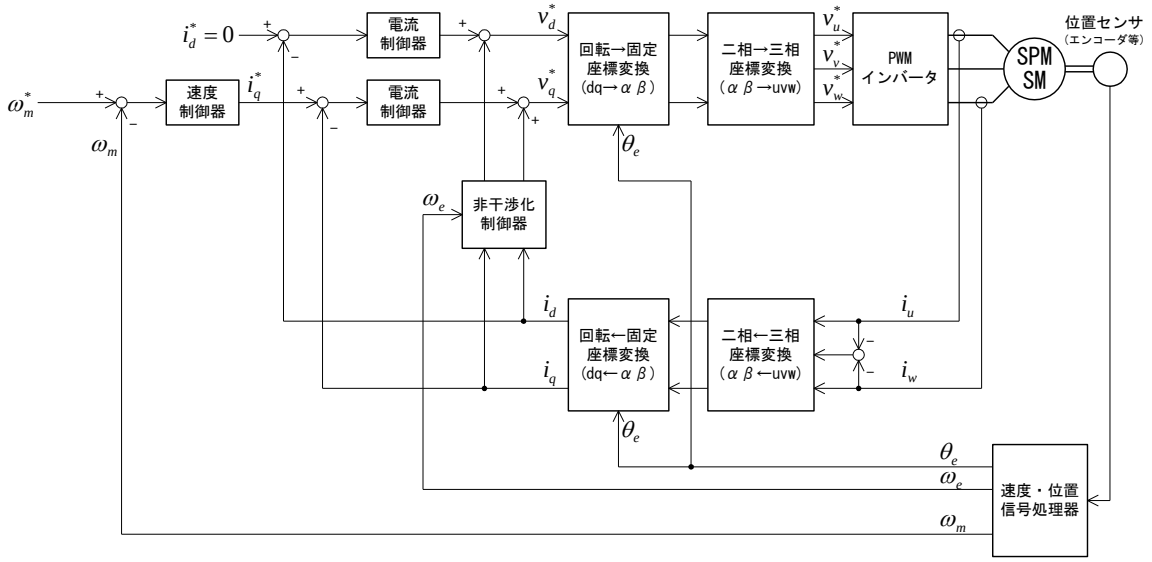


図 4　PMSM のベクトル制御の構成

さらに制御の高性能化のために非干渉制御と呼ばれる制御を使用することがあります。

上で述べたように d・q 軸電流はそれぞれ d・q 軸電圧で制御しますが、実際には (3.2) 式でわかるように d・q 軸間で干渉し合う速度起電力があります。

この影響を排除し、d 軸電流と q 軸電流を独立に制御するための制御方法が非干渉制御です。

非干渉制御はフィードフォワード制御の構成となり、設計にはモータパラメータが必要となります。

4. 基本的な制御プログラムと動作確認

本章では基本的な制御プログラムを用いて実際に付属のモータの速度制御を行います。

PE-Expert4 システムでは座標変換や三相 PWM 出力、ABZ カウンタなどのモータ制御に用いられる機能が関数として提供されているため、モータ制御プログラムを容易に作成することができます。

本章で用いるソースファイルは ”PMVEC_EXP4C6657_1.c” です。

4.1. 制御ブロック図

本章で用いる制御プログラム(PMVEC_EXP4C6657_1.c)では、簡略化のために非干渉制御を省略した以下の制御ブロックを用います。

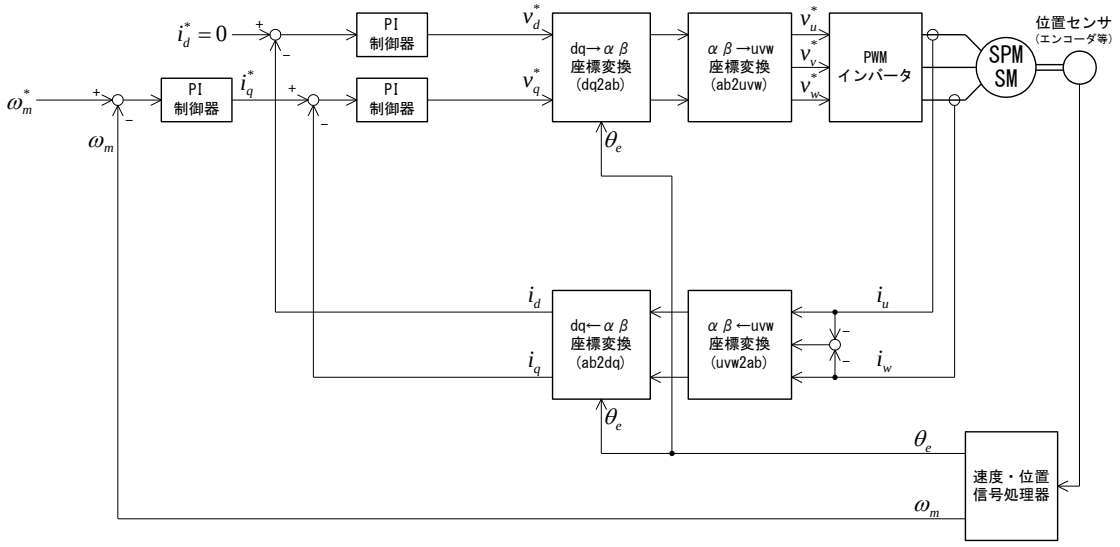


図 5 “PMVEC_EXP4C6657_1.c”の制御ブロック図

4.2. 制御プログラムのフローチャートとプログラムの概要

4.2.1. 関数一覧

“PMVEC_EXP4C6657_1.c”は複数の関数で構成されています。以下に各関数の機能の概略を示します(ソースファイル“PMVEC_EXP4C6657_1.c”では以下の順序と同じ順序で各関数が記述されています。)。

より詳細な処理については次項のフローチャートをご覧ください。

※関数の機能一覧

void MW_main(void) : メインルーチン
・起動 → 初期設定 → 割り込み待ち無限ループ
（起動・停止・エラー処理／LED表示処理）
※起動・停止はMWPE-STK-IO2に実装されているトグルスイッチで
切り替えます。

void error_check(void) : エラーチェックルーチン
・出力過電流／DCリンク過電圧エラー検出
・エラー停止処理

void pwm_int(void):キャリア同期割り込み
・モーター角度検出
・dq軸電流制御
※dq軸電流制御は高速な制御が必要になるため、PWMキャリアに
同期した割り込みルーチンで実行します
・PWM生成
・サーボオン＆サーボオフ処理

void timer0_int(void):タイマ割り込み
・モータ速度検出
・速度制御
※速度制御は電流制御ほど高速に制御する必要はないので、
より低速なタイマ割り込みルーチンで実行します。
※速度指令値はMWPE-STK-IO2に実装された可変抵抗で設定
した値をAD変換機能を用いて入力します。

void initialize(void):システム 初期化ルーチン
・変数初期化
・PWM初期化
・AD初期化
・タイマ初期化
・ABZカウンタ初期化

void pil_intr(void):PILS用インタフェイスポイント
・本関数の入り口で処理停止
・PILSブロックで定義されたインタフェイス変数の交換
・処理再開
・ブレークポイントで停止不可のリソース(タイマ等)の動作再開
・外部割込み処理の起動

void tm2_intr(void):PILS用同期ハンドラ
・割り込みフラグクリア
・ブレークポイントで停止不可のリソース(タイマ等)の動作中断
・PILS用インタフェイスポイント(上記)の起動

4.2.2. 各関数のフローチャート

〈メインルーチン：MW_main()〉

メインルーチンではまず初期化を行い、その後割り込み待ちのメインループに入ります。

メインループ中でディジタル入力の状態を監視し、起動／停止処理とエラー復帰処理を行っています。

また、状態に応じて LED の点灯・消灯を行うためにディジタル出力を制御しています。

DI0：運転／停止切替
DI0=1 で運転・DI0=0 で停止
(MWPE4-PEV ボードの DI0 に MWPE-STK-IO2 の DI3 が接続されています。)
(MWPE-STK-IO2 の DI3 スイッチが on のとき運転・off のとき停止)
DI1：エラーリセット
DI1=0 のときに DI1=1 にするとエラーリセット
(MWPE4-PEV ボードの DI1 に MWPE-STK-IO2 の DI4 が接続されています。)
(停止モードで MWPE-STK-IO2 の DI4 スイッチを on にするとエラーリセット)

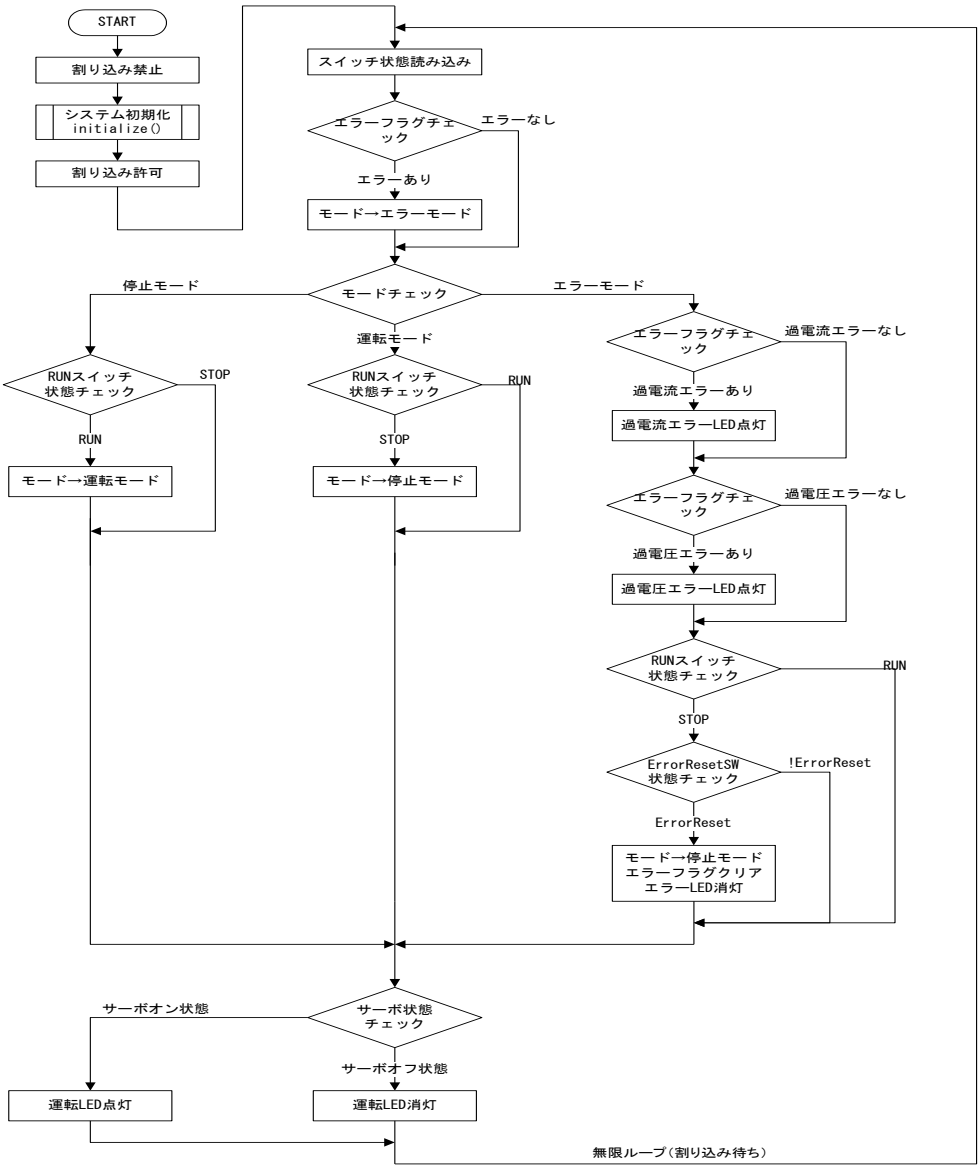


図 6　MW_main()関数のフローチャート

〈システムエラーチェック：error_check()〉

モータ過電流および直流リンク過電圧の検出を行い、エラー時には PWM 出力を停止し、エラーモードに移行します。

本関数は pwm_int()関数から呼び出されます。

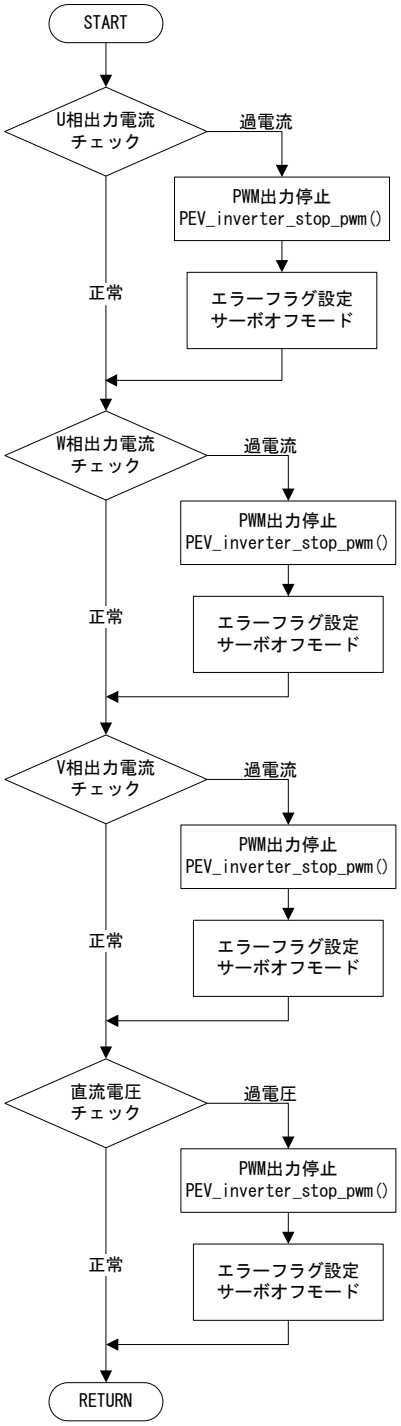


図 7　error_check()関数のフローチャート

〈PWM キャリア同期割り込み：pwm_int()〉

PWM キャリア三角波の「山」に同期して呼び出される割り込みルーチンです。

本ルーチンでは AD 変換機能によるセンサデータ等の入力と電流制御ループの演算、および PWM 指令値の更新を行っています。

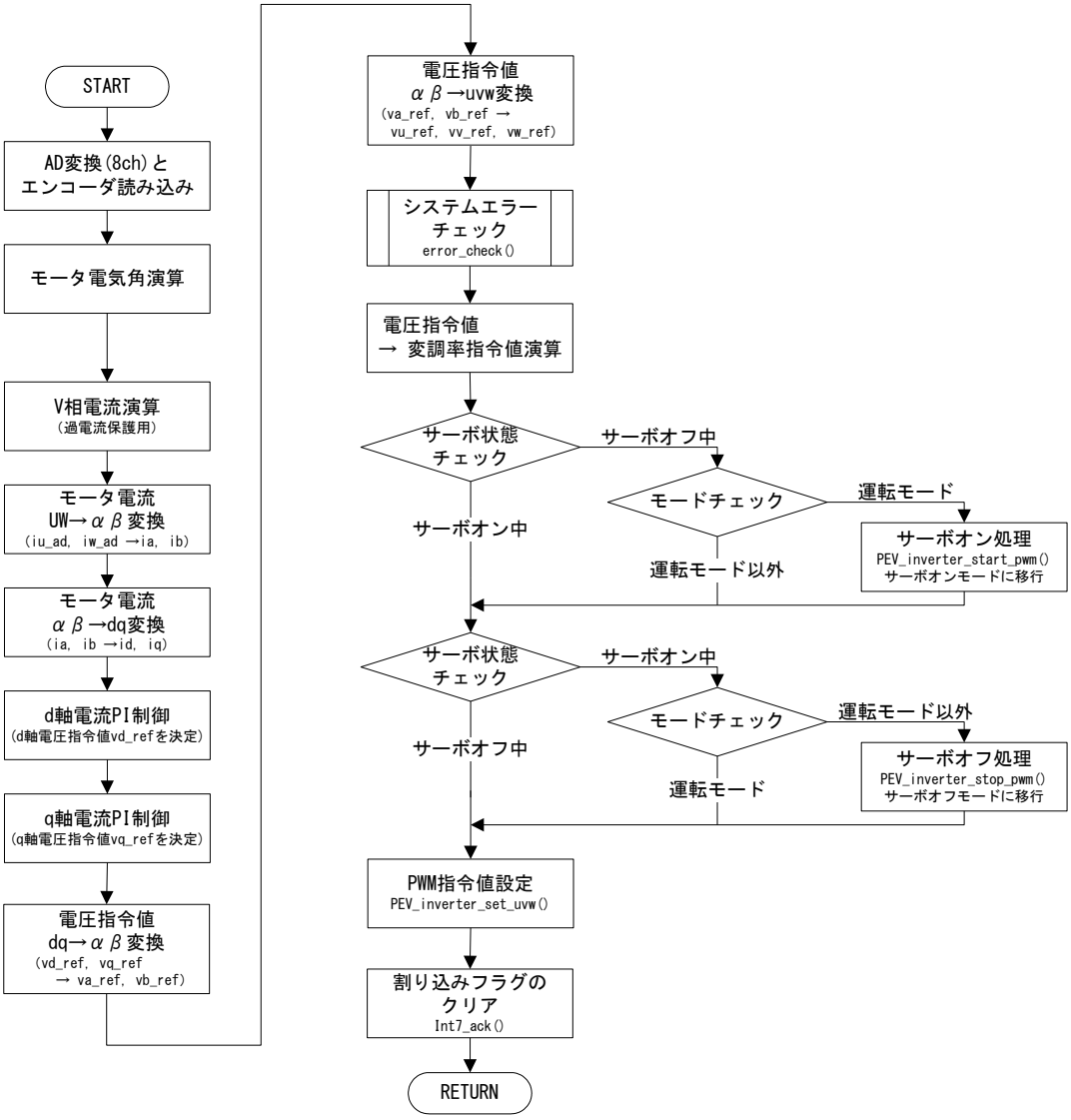


図 8　pwm_int()関数のフローチャート

〈タイマ割り込み：timer0_int ()〉

インターバルタイマにより一定周期で呼び出される割り込みルーチンです。

本プログラムでは 2ms ごとに本ルーチンが呼び出されます。

本ルーチンでは速度制御ループの演算を行い、電流指令値を設定しています。

本プログラムではタイマ割り込みの中で多重割り込みを許可しているため、タイマ割り込み中に PWM キャリア同期割り込みが発生した場合、キャリア同期割り込み処理が実行されます。

これに対して、PWM キャリア同期割り込み中では多重割り込みを許可していないため、PWM キャリア同期割り込み処理実行中にタイマ割り込みは発生せず、pwm_int()終了まで割り込みは遅延されます。

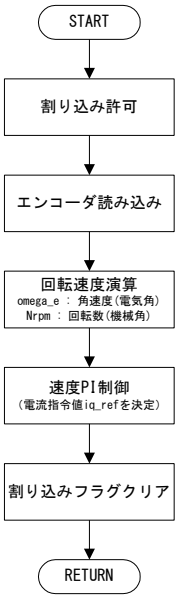


図 9　timer0_int()関数のフローチャート

〈システム初期化：initialize()〉

変数の初期値設定と各機能の初期化を行います。

本関数は MW_main()関数から 1 回だけ呼び出されます。

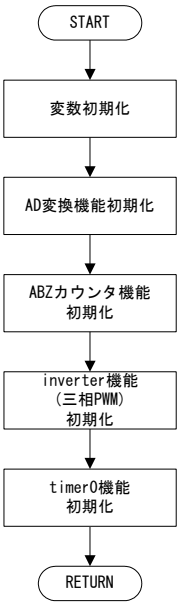


図 10　initialize()関数のフローチャート

<PILS 用インタフェイスポイント：pil_intr()>

PILS 用のインタフェイス変数に対して値の交換を行います。

PSIM のシミュレーションと実プロセッサの時間同期を取るために、この関数の入り口にブレークポイントが置かれます。

プロセッサコアの命令ポインタがブレークポイントに達すると、プロセッサコアは命令の実行を停止しますが、今回のターゲットである C6657 では共通ペリフェラルであるタイマは動作を継続するので、本関数と、後述の tm2_intr()の二段階構造を用いて動作が中断されないペリフェラルの動作をケアする必要があります。

また、PILS においては、PE-Expert4 の拡張ボードからの外部割込みも発生しないので、PSIM 上でモデル化された拡張ボードからの割り込み信号を外部割込みとして処理します。

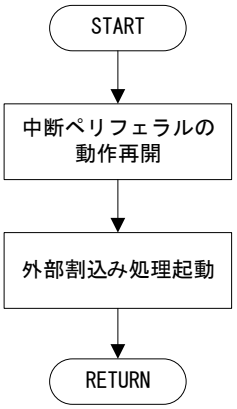


図 11　pil_intr()関数のフローチャート

<PILS 用同期ハンドラ：tm2_intr()>

PILS 用インタフェイスポイントを起動する同期ハンドラです。

PSIM のシミュレーションと実プロセッサの時間同期を取るために、PSIM が想定する定周期で本関数を実行します。

pil_intr()の記述にあるとおり、ブレークポイントで動作を継続するペリフェラルの動作を一時停止する必要があります。

一時停止後、PILS 用インタフェイスポイントを起動します。

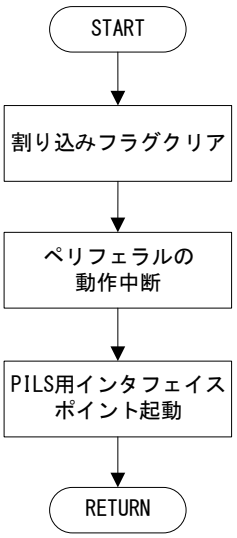


図 12　tm2_intr()関数のフローチャート

4.2.3. 制御プログラム作成時の留意点

① 制御周期について

一般にサーボプログラムは下図のように複数の制御ループを持ち、内側のループほど高速な制御が要求されます。

このため、多くの場合制御周期の異なる複数の割り込みルーチンを用いて制御プログラムを作成します。

本プログラムでは電流制御ループを 100μs ごとに呼び出される PWM キャリア同期割り込みで、速度制御ループを 2ms ごとに呼び出されるタイマ割り込みで実現しています。

サーボモータによる
位置制御系の構成例

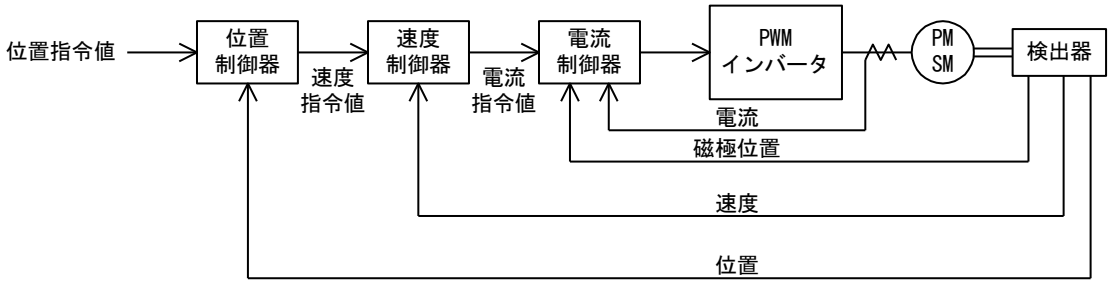


図 13　サーボモータによる位置制御系の構成例

② 保護について

制御ソフトを作成する場合、主回路電源を投入する前に必ず行うべきこととして保護プログラムの実装が挙げられます。

保護プログラムは万一制御が正常に動作しなかった場合に過電流や過電圧などによってインバータ・モータ等が破壊されるのを防ぐものです。

本プログラムでは代表的な保護として直流リンク過電圧および出力電流過電流の検出を行い、検出された場合にはインバータを停止する処理を行っています。

また、これと合わせて過大な指令値が発生しないように電流指令値および電圧指令値にリミッタを設けてあります。

速度制御器および電流制御器は PI コントローラを用いているため、指令値リミッタと合わせて積分リミッタも実装されています。

これらの保護ルーチンは本格的に制御プログラムを動作させる前に、保護レベルを通常より小さくするなどして保護が正常に働くことを確認しておくことをお勧めします。

これにより、安心して制御系のデバッグ・チューニングを行うことができるようになります。

本ソフトウェアに実装されている保護ルーチンは必要最低限のものです。必要に応じて追加することをお勧めします。
（例えば手動スイッチによる緊急停止や、速度が上限値を超えた場合の保護など）

③ PWM 変調方式について

本プログラムでは PWM 変調方式として三角波キャリア比較変調方式を用いていますが、モータ制御の用途では電圧利用率を高めるために指令値に三次高調波を重畳させたり、変調方式として空間ベクトル変調方式を採用することもあります。

PE-Expert4 専用ライブラリには標準で空間ベクトル変調の関数が備えられており、空間ベクトル変調を容易に実現することが可能です。
詳細は「ファンクションリファレンスマニュアル」の PEV_inverter_set_ra()の項をご覧ください。

④ 速度検出について

本プログラムでは速度検出のためのエンコーダ値読み込みを timer0_int()ルーチン中で行っています。
具体的には 2ms 間隔で呼び出される timer0_int()ルーチンの先頭でエンコーダ値のサンプリングを行い、前回のエンコーダ値との差分をとって速度を計算しています。
この方法で速度サンプリングを行うと、pwm_int()処理中に timer0 割り込みが発生した場合に timer0_int()実行開始のタイミングが遅延するため、まれに不正確な速度が検出されることがあります。

これを避けるためには、PWM キャリア同期割り込みルーチン中で速度サンプリングを行う必要がありますが、PWM キャリア同期割り込みルーチンは呼び出し間隔が 100μs と短いため、前回のエンコーダカウンタサンプリング値との差分から速度を求める手法では十分に精度を確保できない問題があります。
例えば本アプリケーションパッケージと同様の 1024 パルスのエンコーダが付属しているモータが 500rpm で回転している場合、100μs でのエンコーダカウンタの差分値はわずかに 3～4 カウントとなってしまいます。

PWM キャリア同期割り込みルーチン中で速度サンプリングを行い、有効な速度検出精度を確保するためには以下のような手法が考えられます。

1. 速度を計算する際に 1 サンプル前のエンコーダ値との差分を求めるのでなく、n サンプル前(例えば 20 サンプル前)のエンコーダ値との差分を求める。
2. エンコーダ値の差分から速度を求めたあとにローパスフィルタを追加する。

4.4章の“PMVEC_EXP4C6657_1.c”は上記の 1.の手法で速度検出を行うように変更してありますので、参考にしてください。

4.3. 制御プログラムの変数・定数一覧

〈#define で定義している値〉

一部の定数は#define で定義しています。付属のモータと弊社製インバータ MWINV-1R022 を組み合わせて使用する場合は変更する必要はありません。

名称	意味	設定値	備考
BDN	PEV ボード番号	0	PEV ボードのデフォルト番号です
PF	モータ極対数	4	付属のモータは 8 極で極対数は 4 です
CNT_RESOLUTION	エンコーダカウント数	4096	4 逓倍後の値を設定してください。 初期状態で付属のモータの設定となっています。 (1024 パルス／4096 カウント)
ENC_DIRECTION	エンコーダカウント方向	1	モータの回転順方向とエンコーダの回転順方向が一致している場合は 1 を、逆になっている場合は-1 を設定してください。
FS	PWM キャリア周波数(Hz)	10000	スイッチング周波数・AD サンプリング周波数・電流制御周波数も FS に一致します。
DT	デッドタイム	4000	弊社製インバータを使用する場合、左記以下の値を設定しないでください。
TM	タイマ割込周期(μs)	2000	
f_timer	タイマ割込周波数(Hz)	1e6/TM	変更不可
TWO_PI	演算用定数	2π	
DTWOPI		1/2π	
ZERO		0.0	
ONE		1.0	
gain_cnt2rad_e		→	変更不可 エンコーダカウント値を電気角[rad]に変換する際の変換ゲイン (2π＊ PF) / CNT_RESOLUTION
gain_rpm2radps_e		→	変更不可 モータ回転数[rpm]を電気角速度[rad/s]に変換する際の変換ゲイン (2π ＊ PF) / 60
gain_radps_e2rpm		→	変更不可 電気角速度[rad/s]をモータ回転数[rpm]に変換する際の変換ゲイン 1 / gain_rpm2radps_e
tmpgain		→	変更不可 エンコーダカウント値を電気角[rad]に変換する際の変換ゲイン gain_cnt2rad_e ＊ f_timer
IAC_ERRLEVEL	出力過電流エラー 保護レベル	4	適切な値に設定してください。 設定を誤るとインバータやモータを破損する危険があります。
VDC_ERRLEVEL	直流過電圧エラー 保護レベル	350	
R_VU_AD	AD レンジ設定	500.0	MWINV-1R022 のセンサに合わせた設定となっています。異なるインバータを使用する際は設定を変更してください。
R_VW_AD	AD レンジ設定	500.0	
R_IU_AD	AD レンジ設定	6.25	
R_IW_AD	AD レンジ設定	6.25	
R_VDC_AD	AD レンジ設定	500.0	
R_IDC_AD	AD レンジ設定	6.25	
R_NRPM_REF_AD	AD レンジ設定	1500.0	回転数指令値[rpm](MWPE-STK-IO2 の可変抵抗で設定可能)の上限値です。変更可能。
R_TEMP_AD	AD レンジ設定	5.0	使用していない AD 入力チャンネル(ch7)のレンジ設定です。
SW_RUN	DI ビット設定	0x01	DI0 を運転／停止切替に使用
SW_RESET	DI ビット設定	0x02	DI1 をエラーリセットに使用

名称	意味	設定値	備考
DO_RUN	DO ビット設定	0	運転表示 LED 用
DO_ERR_IAC	DO ビット設定	3	出力過電流エラーLED 用
DO_ERR_VDC	DO ビット設定	4	直流過電圧エラーLED 用
ST_IAC_ERR	エラーフラグビット設定	0x01	出力過電流エラー用
ST_VDC_ERR	エラーフラグビット設定	0x02	直流過電圧エラー用
MODE_STOP	モード設定用	0	停止モード
MODE_RUN	モード設定用	1	運転モード
MODE_ERROR	モード設定用	3	エラーモード
SERVO_OFF	サーボ状態設定用	0	サーボオフ中
SERVO_ON	サーボ状態設定用	1	サーボオン中
PIL_ENABLE	PILS を有効にします。	-	この定義はコンパイルオプションとして使用されています。 実機環境用のプログラムを生成する場合はこの定義を無効化してください。
__USE_BSD	数学定数を有効にします。	-	この定義は変更しないでください。
PIL_PERIOD	PILS での同期演算周期	50	単位はμs です。設定を変更する場合、PIL ブロックの値も同じように変更してください。

〈変数〉

変数名	型	意味	備考
enc_count_dq	int	エンコーダカウント値	ロータ電気角演算用。pwm_int()で使用
enc_count	int	エンコーダカウント値	回転速度演算用
enc_count_old	int	エンコーダカウント値 (1 サンプル前の値)	
enc_count_diff	int	エンコーダカウント値 の差分	
theta_e	float	ロータ電気角[rad]	enc_count_dq から演算されます
theta_e_offset	float	ロータ電気角の オフセット[rad]	エンコーダの Z パルス発生位置がずれている場合に設定します。初期値では 0.0 が設定されています。
vu_ad	float	U 相電圧(AD 変換した値)	インバータ各部のセンサ信号を AD 変換した値
vw_ad	float	W 相電圧(AD 変換した値)	
iu_ad	float	U 相電流(AD 変換した値)	
iw_ad	float	W 相電流(AD 変換した値)	
Vdc_ad	float	直流電圧(AD 変換した値)	
Idc_ad	float	直流電流(AD 変換した値)	
iv	float	V 相電流計算値	過電流保護用
R_Vdc	float	直流電圧の 1/2 の逆数	変調率計算用。 本プログラムでは initialize()関数内で 2.0/144.0 を設定し定数として扱っています。
ia	float	出力電流のα軸成分	三相二相変換後の値
ib	float	出力電流のβ軸成分	
id	float	出力電流の d 軸成分	dq 変換後の値
iq	float	出力電流の q 軸成分	
kp_omega	float	速度制御用 PI の P ゲイン	速度制御
ki_omega	float	速度制御用 PI の I ゲイン	
iq_ref	float	q 軸電流指令値	
iq_refp	float	速度制御 PI の 比例制御器出力	
iq_refi	float	速度制御 PI の 積分制御器出力	
iq_limit	float	q 軸電流のリミット値	
d_omega_e	float	速度偏差	
kp_id	float	d 軸電流制御用 PI の P ゲイン	d 軸電流制御
ki_id	float	d 軸電流制御用 PI の I ゲイン	
vdq_limit	float	電流制御 PI のリミッタ	
vd_refp	float	d 軸電流制御 PI の 比例制御器出力	
vd_refi	float	d 軸電流制御 PI の 積分制御器出力	
did	float	d 軸電流偏差	
id_ref	float	d 軸電流指令値 = 0.0	
kp_iq	float	q 軸電流制御用 PI の P ゲイン	q 軸電流制御
ki_iq	float	q 軸電流制御用 PI の I ゲイン	
vq_refp	float	q 軸電流制御 PI の 比例制御器出力	
vq_refi	float	q 軸電流制御 PI の 積分制御器出力	
diq	float	q 軸電流偏差	
vd_ref	float	d 軸電圧指令値	dq 軸電圧指令値 d 軸／q 軸電流制御 PI の出力
vq_ref	float	q 軸電圧指令値	
va_ref	float	α軸電圧指令値	dq 逆変換後の電圧指令値
vb_ref	float	β軸電圧指令値	

変数名	型	意味	備考
vu_ref	float	u 相電圧指令値	二相三相変換後の電圧指令値
vv_ref	float	v 相電圧指令値	
vw_ref	float	w 相電圧指令値	
omega_e	float	モータ角速度(電気角)	rad/sec
omega_e_ref	float	モータ角速度指令値 (電気角)	rad/sec。Nrpm_ref から演算
Nrpm	float	モータ回転数[rpm]	観測用。制御には使用していない
Nrpm_ref	float	モータ回転数指令値[rpm]	回転数指令値。AD の ch6 より入力
sw_buffer	int	ディジタル入力バッファ	SW 状態読み込み用
flag_error	int	エラーフラグ	エラー発生時にエラー要因を書き込む変数
Mode_System	int	システム状態フラグ	現在のモードを記憶するための変数。 運転(MODE_RUN)(1)/停止(MODE_STOP)(0)/ エラー(MODE_ERROR)(3)の各モード。
Mode_Servo	int	サーボ状態フラグ	現在のサーボ状態を記憶するための変数。 サーボオン状態(SERVO_ON)(1)/サーボオフ (SERVO_OFF)(0)状態のいずれか。
temp	float	AD の ch7 入力用	AD 予備入力。未使用
enc_count_dq_real	int	エンコーダのカウンタ値	AD(8ch)と同時に読みこととができます
ad_data[8]	float	AD(8ch)同時読み込み用 バッファ	AD 同時読み込み時使用するバッファです

〈列挙型〉

変数名	設定値	意味	備考
VU_AD	0	AD(8ch)同時読み込み用 バッファのインデックス	AD(8ch)同時読み込み用バッファからそれぞれの AD 値の読み出し用です
VW_AD	1		
IU_AD	2		
IW_AD	3		
VDC_AD	4		
IDC_AD	5		
NRPM_REF	6		
TEMP_AD	7		

4.4. 制御プログラムのソースファイル

ファイル名：PMVEC_EXP4C6657_1.c

バージョン：1.0

制御プログラムのソースファイル本体は、本パッケージに同梱されています。

詳細な内容はそちらを参照願います。

4.5. 操作手順と動作確認

この節では本アプリケーションパッケージの制御プログラムを用いてプラントモデルのモータを動作させるまでの手順を解説します。

ここでは、7ページの「2. 準備」に従ってシステムのセットアップが終了し、PE-ViewXシステムの基本的な操作方法を習得していることを前提とします。

PE-ViewXのセットアップ方法及び操作方法に関しましては、別冊の「PE-ViewXインストレーション インストラクション」をご覧ください。

4.5.1. 操作部について

本モデルは回転数指令値の入力、状態表示をトップモデルで行うことができます。

下図(トップモデルの一部)に回転数指令値電圧と状態表示出力用デジタル出力の割り当てを示します。

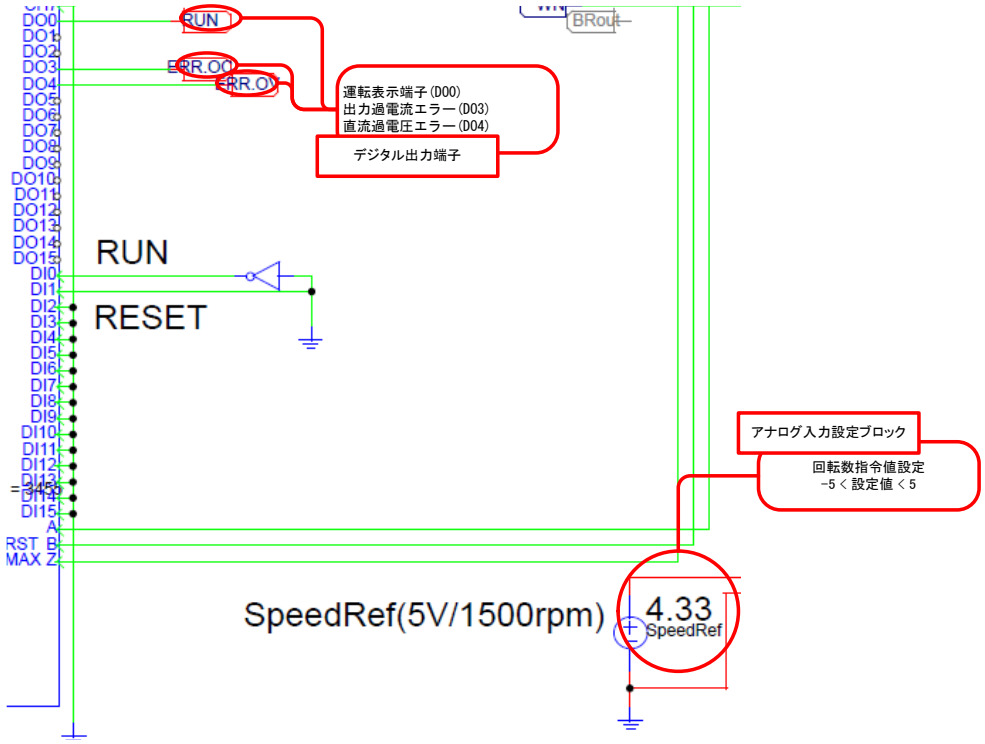


図 14　トップモデルの機能割り当て(PMVEC_EXP4C6657_1.c)

4.5.2. PE-ViewX による制御プログラムのビルド

①PE-ViewX をインストールしたコンピュータで[スタート]→[プログラム]→[PE- ViewX]→[PE-Viewx]と操作し、PE-ViewX を起動します。

・ 下図のメインウィンドウが表示されます

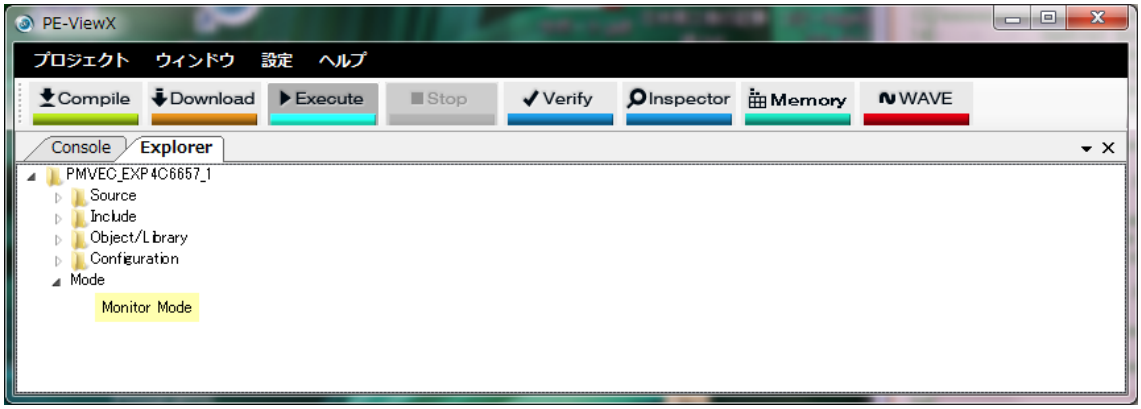


図 15　PE-ViewX の起動

②[プロジェクト]→[開く]ボタンを押し、“PMVEC_EXP4C6657_1.prj”ファイルを選択します。本アプリケーションパッケージを標準インストールした場合、PMVEC_EXP4C6657_1.prj ファイルは以下のフォルダに配置されます。

C:¥Myway¥APPWork¥PMVEC_C6657¥PMVEC_EXP4C6657_1.prj

③[Compile] ボタンを押すと、ロードモジュールの生成を行います。

・ “Compile OK.　“と表示されれば、ビルド処理が正常に完了したことになります。

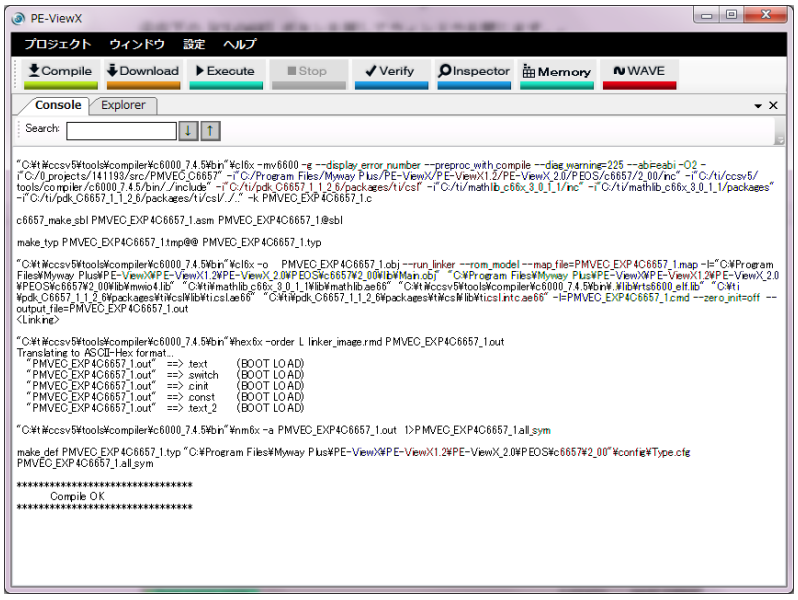


図 16　Compile の実行

④ メインウィンドウの［Download］ボタンを押してください。以下のダイアログが表示されます。ダウンロードが終了すると自動的にダイアログが閉じます。

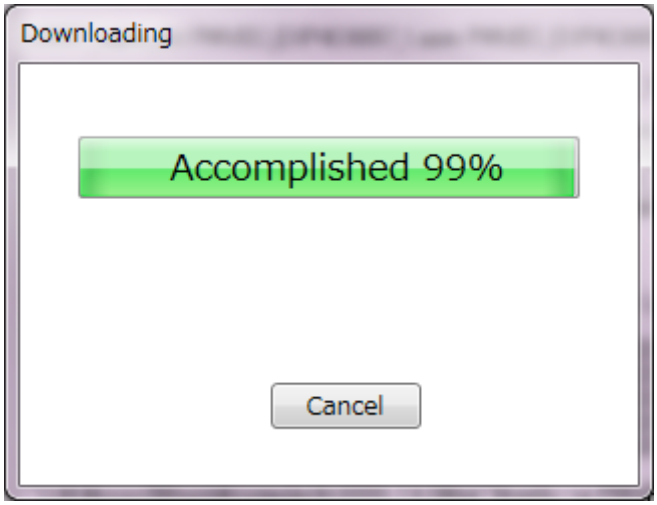


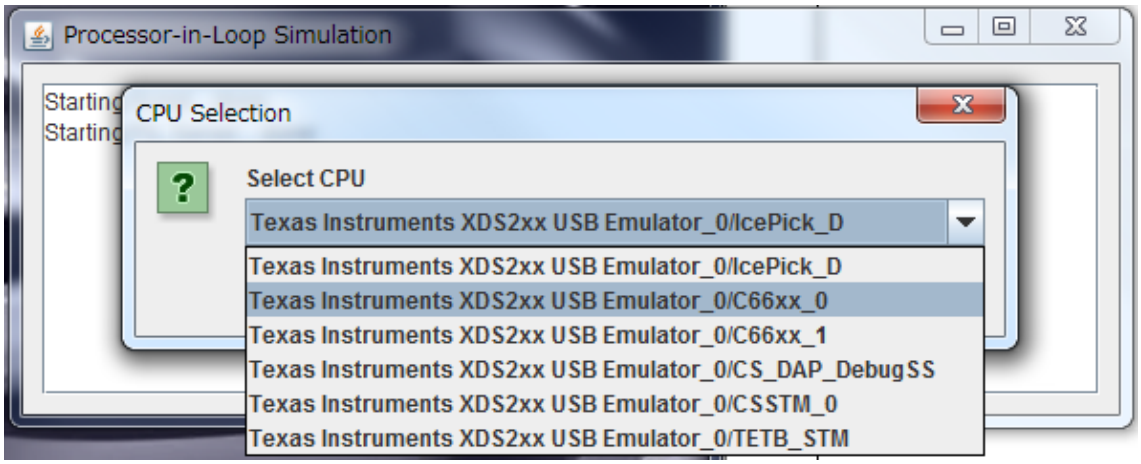
図 17　Download の実行

4.5.3. シミュレーションの始動と停止

① トップレベルの SpeedRef 直流電圧設定を 0 に設定します(回転数指令値の初期値を 0 とするため)。

② シミュレーションを開始します。

③ シミュレーションを開始すると、JTAG インタフェイスの選択画面が表示されるので、"Texas Instruments XDS2xx USB Emulator_0/C66xx_0"を選択してください。



④ トップレベルの SpeedRef 直流電圧設定を-5 < 設定値 < 5 の範囲で設定してください。

・ 回転数指令値設定用ブロックの設定に応じてモータが回転します(SpeedMon の値が変化します)。正転・逆転可能です。

※条件によってはプラントが過電流・過電圧・低電圧等を検出して停止することがあります。プラントで異常を検出して停止した場合、エラー出力が印可されます。その場合、デジタル出力の RUN が OFF し、エラー種別が ERR.OC/ERR.OV に出力されます。

4.5.4. PIL ブロックによる内部波形の観測

PIL ブロックに出力端子を追加することにより制御プログラムの内部変数を同期間隔でモニタすることが可能になります。

ここでは PIL ブロックに出力端子を追加する方法を簡単に紹介します。

①PIL ブロックのパラメータ設定画面を開く

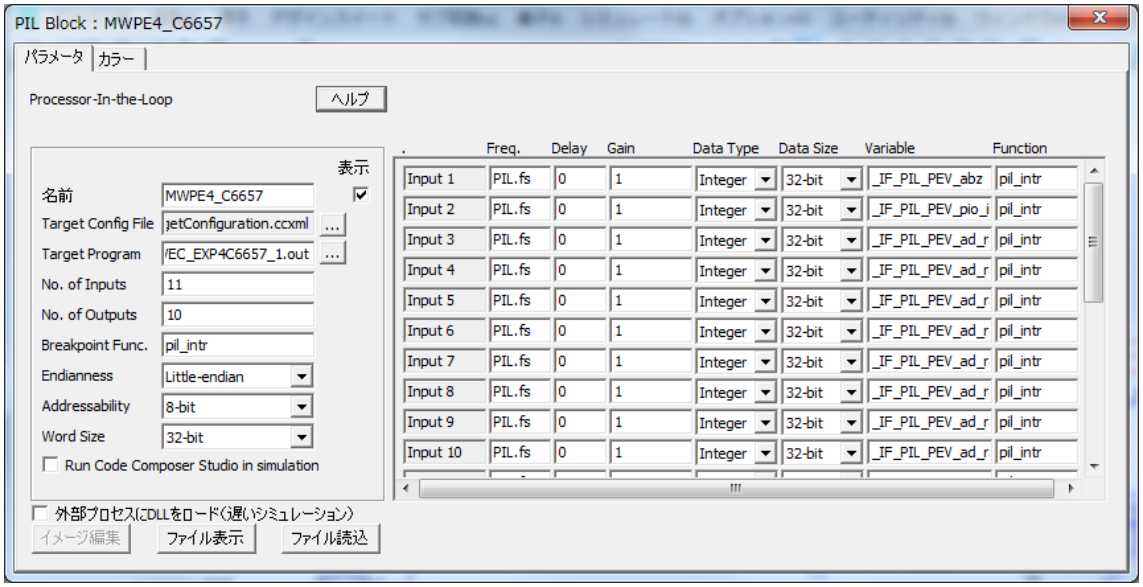


図 18　PIL ブロックのパラメータ設定画面(PSIM)

・モニタしたい変数分の出力チャンネル数を"No. of Outputs"に追加します。
・右側の設定欄に、変数の型/ビット幅/変数名/同期ハンドラ関数名を設定します。
・設定画面を閉じると、ブロックの右側に出力端子が追加されます。
・各端子に電圧スコープや、電圧プローブを接続して値をモニタしてください。

内部変数を観測することで、制御プログラムの動作を理解・確認することができます。

5. パラメータチューニングと高機能化

本章では前章で解説した基本的な制御プログラムをもとに機能の追加およびチューニングを行います。

5.1. 直流電圧変動対策

インバータの直流入力電圧は負荷の大きさによって変化します。また、減速時には直流側にエネルギーが回生されるため、直流電圧が増加します。

先のプログラムでは直流電圧として 141V を想定し、電圧指令値に 2.0/141.0 を乗じて変調率指令値を演算していました。

これでは直流電圧の変化に対応することができません。

そこで、以下のように直流電圧検出値の 1/2 の逆数を使用することで、直流電圧が変化しても電圧指令値通りの電圧を出力することができるようになります。

〈プログラム改造例〉

pwm_int()関数の変調率指令値算出部分(PMVEC_EXP4C6657_1.c:L549)を以下のように書き換えます。

```
if ( Vdc_ad < VDC_MIN ) Vdc_ad = VDC_MIN;
R_Vdc = TWO / Vdc_ad;

mu_ref = vu_ref * R_Vdc;                                /*** U相変調率の算出      ***/
mv_ref = vv_ref * R_Vdc;                                /*** V相変調率の算出      ***/
mw_ref = vw_ref * R_Vdc;                                /*** W相変調率の算出      ***/
```

Vdc_ad に下限値を設定しているのは、逆数演算時に除算オーバフローが発生するのを防ぐためです。

これに必要な VDC_MIN および逆数演算に使用している TWO を共に float で定義し、initialize()ルーチン中で以下のように初期化します。

```
float       TWO;                            /*** 2.0                                 ***/
float       VDC_MIN;                        /*** 直流電圧の逆数計算時のVdcの下限値  ***/
```

```
TWO         = 2.0;
VDC_MIN     = 10.0;
```

5.2. 動作確認

この状態でプログラムを動作させます。

手順に従ってプログラムを起動します。MG-SET につながれた負荷抵抗を無効にした状態で速度指令値設定ブロックを 4.33[V]に設定します。回転数がだんだん大きくなるのが確認できます。

この状態で MG-SET の rpm 端子から出力される電圧を確認します。rpm は指令値とほぼ等しく、1300rpm 付近になっていることが確認できます。

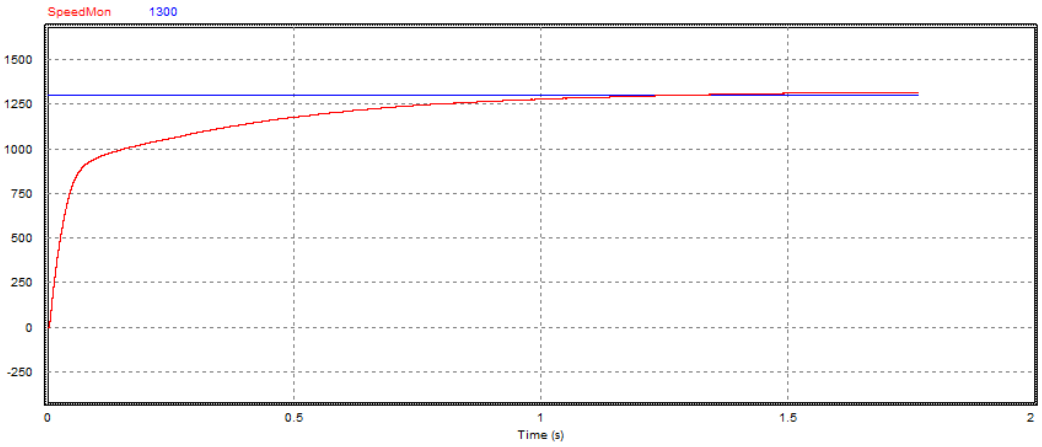


図 19　シミュレーションによる過渡特性(速度が追従している場合)

次に負荷抵抗の設定を 50[Ω]に設定します。

回転数が下がることが確認できます。

約 1300rpm の回転数指令値に対して回転数は約 800rpm まで下がります。

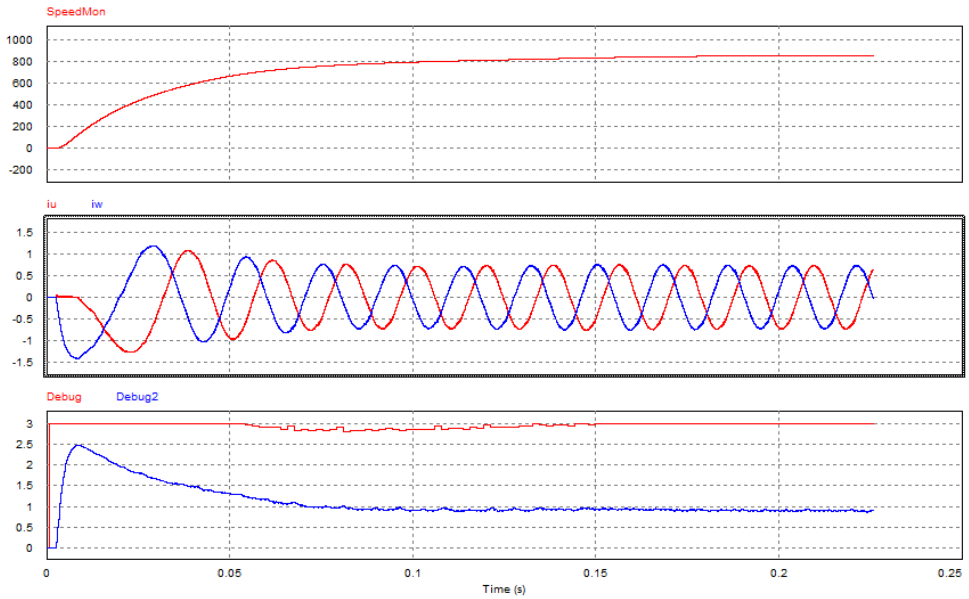


図 20　PSIM による過渡特性(速度が追従していない場合)

この状態で q 軸電流指令値 iq_ref(Debug)と q 軸電流 iq(Debug2)を上図で比較してみます。

iq は iq_ref に追従できず、iq_ref は+3A のリミット値に達してしまっていることが確認できます(上段の赤色が iq_ref、青色が iq です)。

実際の q 軸電流は約 0.2A しか流れていませんので、リミット値までは余裕があるのですが、指令値がリミット値に達してしまったためにこれ以上電流を流せなくなってしまっていると考えられます。

これは電流制御系のパラメータ設定に問題があると考えられます。

電流制御系は PI 制御系ですが、現時点では I ゲインが 0 となっているため(下図参照)、実質的には比例制御と考えられます。

このため定常偏差が残ってしまい、電流指令値に電流値が一致しない状態となっています。

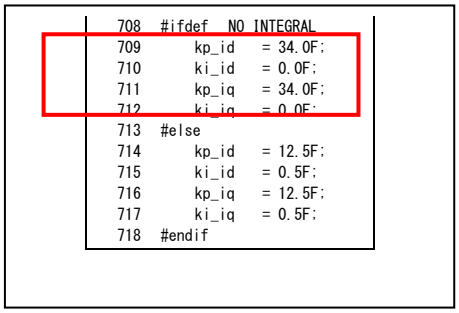


図 21　ソースコード上のパラメータ設定確認(制御ゲインの確認)

そこで、次節でパラメータの調整を行います。

5.3. 制御パラメータの調整

本制御ソフトのような複数の制御ループが存在する制御系では、内側のループから設計を行います。

PI 制御器の設計方法は様々な方法が知られていますが、ここでは設計手順については割愛します。添付の書籍等の参考文献をご覧ください。

ここでは調整後のパラメータと動作特性を紹介します。

kp_id = 12.5
ki_id = 0.5
kp_iq = 12.5
ki_iq = 0.5
kp_omega = 0.03
ki_omega = 0.0001

先程の条件と同じ条件での動作波形を以下に示します。

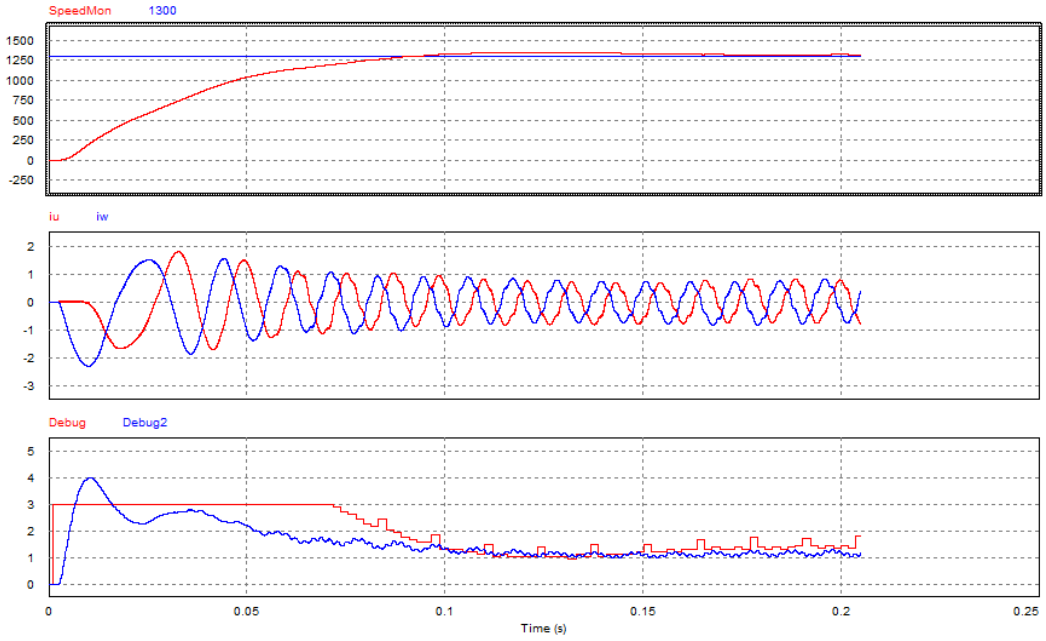


図 22　PSIM による過渡特性(q 軸電流が追従している場合)

q 軸電流・回転数共に指令値にほぼ追従していることが確認できます。

5.4. 非干渉制御の追加

ここでは前章では簡略化のために省略していた非干渉制御を追加します。

PMSM の回転座標での回路方程式は以下のように表されます。

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} r + pL_d & -\omega L_q \\ \omega L_d & r + pL_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega\phi_m \end{bmatrix} \qquad (5.1)$$

(L_d, L_q はそれぞれ d 軸方向、q 軸方向のインダクタンスであり、SPMSM の場合一般に L_d=L_q=L_a となります。)

前章のプログラムでは d 軸電圧で d 軸電流を、q 軸電圧で q 軸電流を制御していましたが、実際には dq 軸間に干渉項が存在することがわかります。

この干渉項を計算してあらかじめ除去することで、dq 軸間の干渉がなくなり、互いに独立して制御を行うことができるようになります。

これが非干渉制御です。具体的には以下の式で表すように dq 軸電圧指令値に速度に比例する干渉項を加えることで実現できます。

$$\begin{aligned} v_d^* &= v_d^{*\prime} - \omega_e L_q i_q \\ v_q^* &= v_q^{*\prime} + \omega_e \left(\phi_m + L_d i_d\right) \end{aligned} \qquad (5.2)$$

v_d^*, v_q^* : PWM制御に用いるdq軸電圧指令値

v_d^*', v_q^*' : 電流制御器の出力のdq軸電圧指令値

〈プログラム改造例〉

以下の 4 行を vd_ref, vq_ref 演算後に追加することで実現できます。

（合わせて変数定義を追加する必要があります。）

```
dvd_ref = La * iq * omega_e;
dvq_ref =  ( Ke + La * id ) * omega_e;
vd_ref -= dvd_ref;
vq_ref += dvq_ref;
```

非干渉制御を行うためにはモータのパラメータが必要になります。

5.5. その他のよく用いられる処理

5.5.1. フィルタ処理

電流センサ・電圧センサなどからの入力信号はノイズ成分を含んでいることが多いため、これを除去するために測定値や内部変数に対してフィルタ処理を行うことがあります。

特に本アプリケーションパッケージで紹介した方法でベクトル制御を行う場合、dq 回転座標で表された量(例えば id, iq)は理論的には直流となるため、ローパスフィルタ処理を行うことによって容易に有効な成分を取り出すことができます。

5.5.2. ソフトスタート処理

起動時にはじめから大きな速度指令値を入力すると、速度制御器に対して大きな値が入力され、制御系の大きな過渡変化を引き起こします。

また大きな電圧指令値が発生するために過電流保護などがかかりやすくなってしまいます。

これを防ぐために、起動時に指令値を 0 から徐々に大きくする「ソフトスタート」と呼ばれる制御が一般に行われています。

これによりなめらかな起動が可能になります。

制御方式によっては同様の目的で内部の制御ゲインを 0 から徐々に大きくする方式をとるものもあります。

5.5.3. 加速度リミッタ

モータの加速度は慣性モーメントとモータの発生トルク・負荷トルクによって決定されます。

また、モータが発生可能なトルクの最大値はモータ固有に決まっています。

従って、特定の条件を考えたときのモータの加速度はモータの最大トルクで制限されることになり、それ以上の加速は不可能なことがわかります。

このため、加速度に相当する速度指令値の傾きにリミッタをかけ、速度指令値が一定以上の傾きで急変するのを防ぐような制御系を構築することがあります。

ご注意

1. 本資料に記載された製品の仕様は、予告なく変更することがあります。
2. 本資料の内容については、万全を期しておりますが、万一ご不明な点などがありましたら、弊社までお申しつけください。
3. 本資料に記載された情報に起因する損害または特許権その他権利の侵害に関しては、弊社は一切の責任を負いません。
4. 本資料によって第三者または弊社の特許権その他権利の実施権を許諾するものではありません。
5. 弊社の書面許諾なく、本資料の一部または全部を無断で複製することを固くお断りします。
6. 本資料に記載された製品を改造したものに対しては、弊社は一切の責任を負いません。
7. 本資料に記載された製品をユーザ装置に組み込む際には、バックアップやフェイルセーフ機能をシステム的に設置してください。
8. 弊社は、人命に関わる装置として特別に開発したものは用意しておりません。
9. 本資料に記載されている会社名、商品名は、各社の商標または登録商標です。

Copyright 2002-2017 by Myway Plus Corporation
All rights reserved. No part of this manual may be photocopied or reproduced in any form or by any means without the written permission of Myway Corporation.