PLECS

Tutorial

Hands on Introduction to PLECS TI C2000 Code Generation

PLECSによるTI C2000コード生成の紹介

Tutorial Version 1.0

KESCO KEISOKU ENGINEERING SYSTEM
計測エンジニアリングシステム株式会社
https://kesco.co.jp

1 はじめに

このチュートリアルでは、Texas Instruments(TI) C2000プロセッサファミリ向けのPLECS組み込みコード生成ツールの主要な関数ブロックについて学びます。TI C2000マイクロコントローラ(MCU)は、リアルタイムのパワーエレクトロニクス制御用を目的としています。

始める前に このチュートリアルを開始する前に、以下を利用できるよう、インストールしておいてください:

- 1つのPLECS Coderのライセンス。
- TI C2000 Target Support Library: TI C2000 Target Support User Manualのクイックスタートで TI C2000のコード生成の構成に関するステップバイステップの指示に従ってください[1]。
- 1つのTI C2000 Piccolo MCU F28069 LaunchPad [2]開発キット。以後"LaunchPad"と呼びます。
- RT Box 1 Target Support Library: RT Box User Manual [3]のクイックスタート、PLECSとRT Boxの構成に関するステップバイステップの指示に従ってください。

オプションとしてセクション4、5、および6では、コントローラをリアルタイムで検証するために、次の項目を使用します。これらは、オフライン(PLECS)で閉ループシミュレーションを実行するときは必要ないことに注意してください。

- 1台のPLECS RT Box 1
- 1つのRT Box LaunchPadインタフェースボード[4]。以後“Interface Board”と呼びます。

RT Box(オプション)を使用したチュートリアルのハードウェアの配置は、図1に示しています。以下の構成も確認してください:

- Interface Board(緑)のジャンパ**RST**はオープンのままです。
- LaunchPad(赤)のジャンパ**JP6**はオープンのままです。

図1: RT Boxを使用したこのチュートリアルのハードウェア設定(オプション)

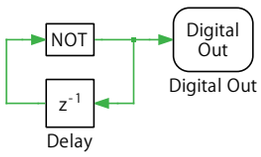
2 LEDの点滅

最初の演習では、MCUのGPIO 34に接続されたLaunchPad上の赤色LED "D9"を点滅させる簡単なプログラムを作成します。**ウィンドウ**のドロップダウンメニューから**ライブラリブラウザ**を開きます。メインの回路図に"サブシステム"ブロックを配置し、“Controller”というラベルを付けます。

“Controller”サブシステム内で、デューティサイクル50%、0.5HzでLEDを点滅させるモデルを作成します。LEDは1秒間点灯し、その後1秒間消灯します。簡単な方法は、図2に示す概略図を参照してください。Digital Outブロックは**TI C2000 Target**ライブラリから取得する必要があることに注意してください。Digital Outブロックの**Digital output GPIO number(s)**パラメータを`34`に設定します。

回路図を完成したら、**シミュレーション** -> **シミュレーションパラメータ...**ウィンドウの**初期化**タブにある**モデル初期化コマンド**ウィンドウに次のコード行を入力して、実行ステップ サイズを定義します: `Ts.Controller = 100e-6;`

図2: LED点滅コントローラの回路図



"Controller"サブシステムは、構成に応じて、TI 28069 LaunchPadのターゲット固有のコードに直接変換できます。以下の手順に従って、"Controller"サブシステムを LaunchPadにアップロードします。

あなたのタスク:

1 最上位レベルの回路図から“Controller”サブシステムを右クリックし、**サブシステム** -> **実行の設定...**を選択します。構成ウィンドウで、**コード生成機能の有効化**チェックボックスを選択します。

2 **Coder** -> **Coderオプション...**ウィンドウで、システムリストから"Controller"を選択します。

3 **タスク**タブから、"離散化ステップ サイズ"フィールドに`Ts.Controller`と入力します。

4 次に、**ターゲット**タブで、ドロップダウンメニューから`TI2806x`デバイスを選択します。

5 **Build type**に`Build and program`を選択し、**Board**に`LaunchPad`を選択します。

6 LaunchPadがUSBケーブル経由でホストコンピュータに接続されていることを確認してから、**ビルド**をクリックします。

ビルドプロセスが完了するまでお待ちください。正しくプログラムされていれば、LaunchPadの赤色LED "D9"は0.5Hz、50%のオン時間で点滅します。

この段階では、モデルは参照モデル`c2000_tutorial_1.plecs`と同じになるはずです。

3 PWM出力

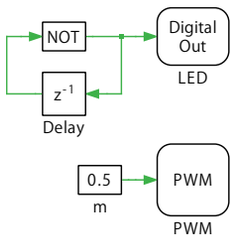
TI C2000 TargetライブラリのPWMコンポーネントは、パワーエレクトロニクスコンバータのスイッチングデバイスを制御するための PWM信号を生成するために使用されます。

あなたのタスク:

1 “Controller”サブシステムに**TI C2000 Target**ライブラリからPWMブロックを追加し、**PWM generator(s)**パラメータが`1`に、**Carrier frequency**パラメータが`10e3`に設定されていることを確認します。

2 定数ブロックを使用して、PWMブロックにデューティサイクルを供給します。50%のデューティー比を持つPWM信号を生成するには、定数の値を0.5に設定する必要があります。この手順については、図3に示す回路図を参照してください。

図3: PWMブロックを追加したコントローラの回路図



3 MCUの動作中にPWM信号のデューティ サイクルを変更するには、**Coder** -> **Coderオプション...**ウィンドウから**パラメータのインライン化**タブの**例外**リストに定数ブロックをドラッグします。

4 次に、前の演習の最後と同じ手順に従って、"Controller"サブシステムをMCUにアップロードします。

5 アップロードが完了したら、**Coderオプション**ウィンドウの**外部モード**タブからMCUをホストPCに接続します。まず、**ターゲットデバイス**フィールドの横にあるボタンをクリックします。適切な**デバイス種類**を選択し、**検索**ボタンをクリックして利用可能なデバイスを表示します。最後に、**デバイス名**ドロップダウンメニューを使用して、使用可能なデバイスのリストから適切なデバイスを選択します。**OK**をクリックして選択したデバイスを確認し、**接続**ボタンをクリックして外部モードを有効にします。

この時点で、定数ブロックのパラメータウィンドウを通じてPWMのデューティ サイクルを変更できるはずです。

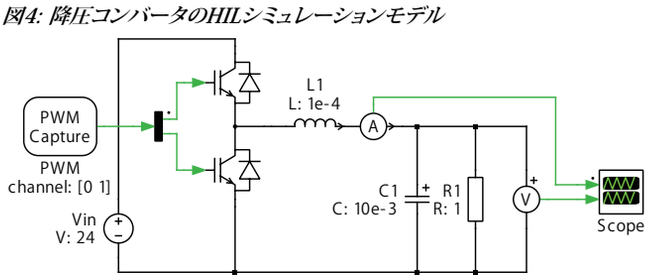
オプション: アナログオシロスコープを使用して、LaunchPadの**J4**コネクタのピン**39**と**40**に2つのプローブを接続してみます。反対極性の2つのPWM信号を測定します。定数ブロックの値を変更すると(外部モードでMCUに接続されている場合)、デューティサイクルの変化をオシロスコープで観察できます。

4 降圧コンバータのHILシミュレーション

ハードウェアインザループ(HIL)シミュレーションは、組み込みコントローラの動作を検証する一般的な手法です。ここで、プログラムしたLaunchPadをテストするための簡単なHILモデルを作成します。モデルに変更を加える前に、**外部モード**タブの**切断**ボタンをクリックして外部モードを非アクティブ化します。

次に、最上位レベルの回路図に新しいサブシステムを作成し、"Plant"という名前を付けます。"Plant"サブシステムで降圧コンバータのシミュレーション モデルを構築するには、図4を参照してください。プラントモデルの実行ステップ サイズを定義するには、**シミュレーション** -> **シミュレーションパラメータ...**ウィンドウの **初期化**タブにある**モデル初期化コマンド**ウィンドウに次のコード行を入力します。`Ts.Plant = 1e-6;`

図4: 降圧コンバータのHILシミュレーションモデル



あなたのタスク:

1 スイッチングセルには、**電気回路/パワー素子モジュール**ライブラリのハーフ ブリッジブロックを使用します。

2 回路図に示されている値を使用して、インダクタ、キャパシタ、抵抗器を構成します。

3 デマルチプレクサブロック経由でPWM Captureブロックを2つのIGBTに接続し、**Digital Input Channel(s)**(デジタル入力チャネル)をベクトル`[0 1]`に設定します。**Averaging interval (offline only)**(平均間隔)を`Ts.Plant`に設定します。

4 最上位レベルの回路図から、"Controller"サブシステムのPWM出力を、"Plant"サブシステムの PWM Capture入力に接続します。

この段階で、PLECSで電力コンバータと開ループコントローラをシミュレートできるようになりました。モデルを実行し、"Plant"サブシステムのPLECSスコープでインダクタ電流と出力電圧を観察します。

以下の手順に従って、"Plant"サブシステムを RT Boxにアップロードします:

あなたのタスク:

1 最上位レベルの回路図から"Plant"サブシステムを右クリックし、**サブシステム** -> **実行の設定**を選択します。構成ウィンドウで、**コード生成機能の有効化**チェックボックスを選択します。

2 **Coderオプション**ウィンドウの**システム**リストから、"Plant"を選択します。

3 **タスク**タブから、**離散ステップサイズ**フィールドに`Ts.Plant`と入力します。

4 **ターゲット**タブのドロップダウンメニューから`PLECS RT Box 1`を選択します。

5 利用可能な RT Boxを選択するには、**ターゲットデバイス**フィールドの横にある双眼鏡ボタンをクリックします。

6 **ビルド**をクリックします。

アップロードが成功したら、**Coderオプション**ウィンドウの**システム**リストから"Plant"を選択したままにします。次に、**外部モード**タブから**接続**ボタンを押して"Plant"サブシステムの外部モードをアクティブにし、下の**自動トリガを有効化**ボタンをクリックします。"Plant"サブシステム内でPLECSスコープを開きます。スイッチングのリップル周波数を含むインダクタ電流が約12Aを中心とし、キャパシタ電圧が12Vであることを確認します。前述のように、"Controller"サブシステムの外部モードを有効にして、PWM出力のデューティサイクルを変更することもできます。そうすることで、降圧コンバータの平均インダクタ電流とキャパシタ電圧の変化が観察できます。

この段階では、モデルは参照モデル`c2000_tutorial_2.plecs`と同じになるはずです。

5 ADCサンプリング

閉ループ制御では、MCUのアナログ/デジタルコンバータ(ADC)を使用して、アナログ信号の測定値を、組み込み制御コードで使用できる、信号を表す変数に変換します。この演習では、前回の演習で紹介したPWM出力ブロックと組み合わせて、さまざまなADCサンプリング方法を練習します。

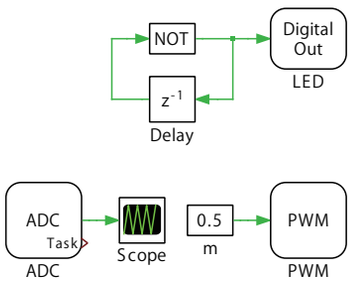
あなたのタスク:

1 **TI C2000 Target**ライブラリから ADCブロックを"Controller"サブシステムに配置し、**Analog input channel(s)**パラメータを`7`に設定します。

2 この構成は、LaunchPadのジャンパ**J1**のピン**23**から供給される ADCチャネル7の入力が制御環境に読み込まれることを示しています。

3 次に、**Scale(s)**パラメータを`10`に設定します。図5に示すように、PLECSスコープコンポーネントをADCブロックの出力ポートに接続します。

図5: ADCブロックを追加したControllerモデル

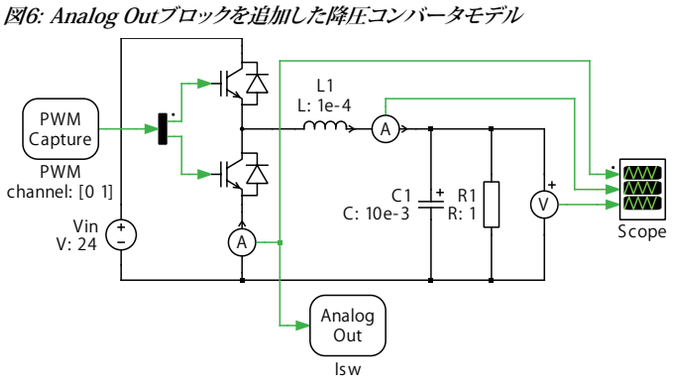


ここで、"Plant"サブシステムを変更して、LaunchPadの電流検知信号を作成します。低電圧アプリケーションでは、回路電流は通常、ローサイドスイッチから取得され、低コストのシャントベースの電流センサを導入できます。

あなたのタスク:

1 降圧コンバータのローサイドスイッチと直列に電流計を配置します。図6に示す電流計の方向に注意してください。

図6: Analog Outブロックを追加した降圧コンバータモデル



2 **PLECS RT Box**ライブラリから**Analog Out**ブロックを"Plant"サブシステムにドラッグし、名前を"Isw"に変更して、その入力ポートを電流計に接続します。

3 Analog Outブロックの**Scale**(スケール)パラメータを`0.1`に指定します。次に、**Minimum output voltage**(最小出力電圧)を`0`に、**Maximum output voltage**(最大出力電圧)を`3.3`に設定します。

これらの構成では、RT Boxのアナログ出力チャネル0は、$V = I_{sw} \cdot 0.1$に等しく、0V ～ 3.3Vの範囲に制限されたアナログ電圧信号を生成します。ここで、I_{sw}は電流計からの電流測定値です。インタフェースボードでは、RT Boxのこのアナログ出力チャネル0は、LaunchPad上のMCUのADC input 7に接続されています。

変更した"Controller"サブシステムをLaunchPadにアップロードし、"Plant"サブシステムを RT Box にアップロードします。"Controller"サブシステムの外部モードを有効にし、PLECSスコープで出力を観察します。

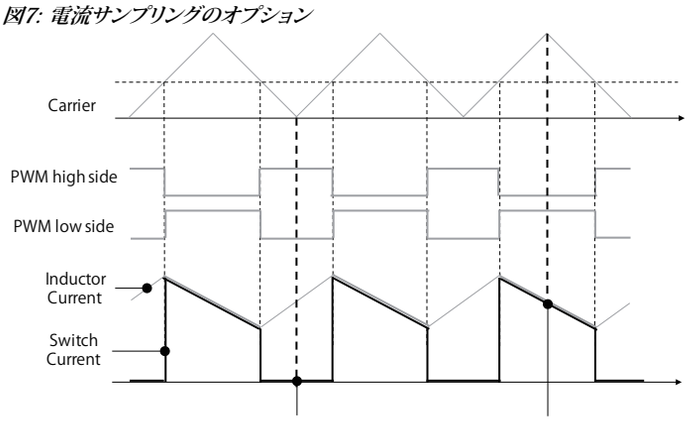
この段階では、モデルは参照モデル`c2000_tutorial_3_1.plecs`と同じになるはずです。

スコープ内の ADC信号が常にゼロなのはなぜですか(小さな振幅の不規則なノイズを除く)?

実際の電流はパルス状の性質を持っていますが、コントローラは電流がゼロの期間のみそれを測定します。

注意: MCUでは、通常、デューティサイクル(変調指数)を搬送波と比較することによってPWM信号が生成されます。変調指数が搬送波より大きい場合、MCUはハイサイドスイッチをオンにし、ローサイドスイッチをオフにします。これにより、降圧コンバータのインダクタ電流が増加します。そうしないと、ローサイドスイッチに電流が流れ、インダクタ電流が減少します。デフォルトでは、キャリアが"0"に達し、インダクタ電流がハイサイドスイッチを流れ始めると、ADC入力チャネルの検知信号が読み取られます。したがって、図7に示すように、ゼロ電流が測定されます。

図7: 電流サンプリングのオプション



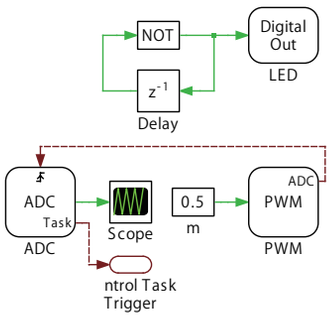
平均インダクタ電流を制御する場合、最小電流リップルと最大電流リップルの中間の電流をサンプリングすることが望ましいです。この目的のために、ローサイドスイッチの電流検知では、キャリアが最大値に達したときに電流検知信号を読み取るようにADCを再構成する必要があります。図8を参照して、次の変更を行ってください:

あなたのタスク:

1. PWMブロックのパラメータウィンドウを開き、**イベント**タブに切り替えます。**ADC trigger**パラメータを`Overflow`に変更します。
2. 次に、ADCブロックのパラメータウィンドウで、**Trigger source**パラメータを`Show trigger port`に変更します。
3. 前の手順を完了すると、PWMブロックに追加のトリガ出力ポートが表示され、ADCブロックにトリガ入力ポートが表示されます。これら2つのポートを接続します。
4. 最後に、**TI C2000 Target**から制御タスク トリガコンポーネントを取得し、ADCブロックの"Task"ポートに接続します。

要約すると、これらの構成により、PWMブロックはキャリア最大値でADCサンプリングをトリガし、ADCがサンプリングを終了すると、制御タスク(LEDの点滅など)が実行されます。
変更した"Controller"サブシステムをアップロードし、外部モードを有効化します。これで、PLECSスコープでゼロ以外の電流を確認できるはずです。

図8: ADCサンプリングと制御タスクの実行を構成

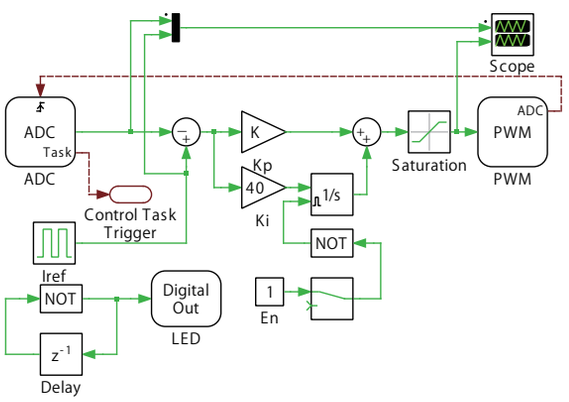


この段階では、モデルは参照モデル`c2000_tutorial_3_2.plecs`と同じになるはずです。

6 閉ループ制御

この演習では、降圧コンバータの閉ループ電流コントローラを構成します。図9を参照して"Controller"サブシステムを変更し、次の操作を実行してください:

図9: 電流制御用閉ループPIレギュレータを備えたControllerサブシステム



あなたのタスク:

1 基準電流入力として機能するパルス発生器ブロック"Iref"を含めます。**周波数**を`20`Hz、**High-state出力**を`7`、**Low-state出力**を`5`に設定します。

2 2つのゲイン(利得)ブロックの"Kp"ゲインを`0.01`に、"Ki"ゲインを`40`に設定します。

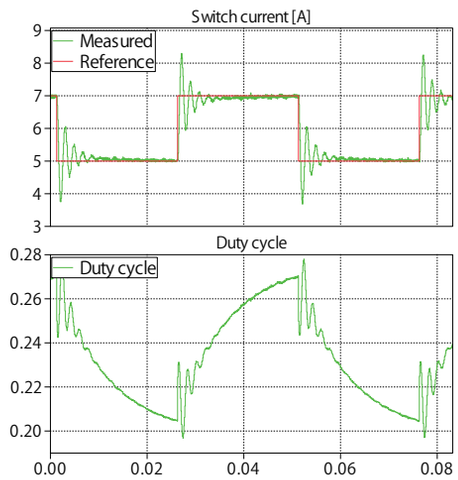
3 飽和ブロックの**上限**パラメータを`0.9` に、**下限**パラメータを`0.05`に設定します。

4 PWMが非アクティブになったときに積分のオーバーフローを回避するには、積分器ブロックの**飽和上限**を`1`に、**飽和下限**を`0`に設定します。次に、**外部リセット**パラメータでレベルを選択して、積分器ブロックのリセットポートを有効にします。図9に示すように、手動切替スイッチ、定数ブロック、および**NOT**を選択した論理演算子ブロックを接続します。論理演算子の出力を積分器のリセットポートに接続します。

5 ゲイン"Kp"と"Ki"、手動切替スイッチブロック、定数ブロック"Iref"を**Coderオプション**ウィンドウの**パラメータのインライン化**タブにドラッグします。

変更したコントローラモデルを MCUにアップロードする前に、オフラインシミュレーションでコントローラのパフォーマンスを事前に検証できます。シミュレーションを開始し、回路の応答を観察します。"Controller"サブシステムのPLECSスコープ内の波形は、図10のようになるはずです。"Plant"サブシステムの電流波形にはスイッチング周波数リップルが含まれますが、"Controller"サブシステムでは基本成分のみが表示されることに注意してください。

図10: "Kp"ゲインが0.01、"Ki"ゲインが40の場合の回路応答



これらの変更が完了したら、新しい"Controller"サブシステムを LaunchPadにアップロードし、**外部モード**と**自動トリガ**を有効にします。"Iref"ブロックからの電流参照の変化とトリガを同期することができます。ここで、PIレギュレータのパラメータをオンラインで調整します。

あなたのタスク:

1 手動切替スイッチを切り替えて、MCUのPWM出力をアクティブにします。積分器がリセットされると、非常に小さなKpゲインのみが有効になり、その結果デューティサイクルが低くなることに注意してください。飽和ブロックは、PWMのデューティサイクルを0.05に固定します。したがって、PLECSスコープ上では少量の電流が測定されるはずです。

2 "Kp"と"Ki"のゲインを他の値に設定し、結果の測定値をPLECSスコープで観察します。実行時にPIパラメータへのすべての変更の結果を確認できるように、"Controller"サブシステムの**外部モード**がアクティブになっていることを確認してください。

3 振動とオーバーシュートを減らすことを目的として、PIパラメータの調整を続けます。

この段階では、モデルは参照モデル`c2000_tutorial_4.plecs`と同じになるはずです。

7 まとめ

これで、PLECSでコントローラモデルを構築し、制御ロジックを実際の組み込みプロセッサに展開できました。将来、パワーエレクトロニクスコンバータのプロトタイプの制御設計を行う際には、PLECSツールチェーン環境を活用して、その柔軟性と可観測性を活用することを検討してください。

8 参考文献

[1] *TI C2000 Target Support User Manual*, Plexim GmbH, Online:
https://www.plexim.com/sites/default/files/c2000manual.pdf
日本語版は以下から:
https://adv-auto.co.jp/products/plexim/manual.html

[2] Texas Instruments, *"LAUNCHXL-F28069M Overview" User's Guide, SPRUI11B, 2019*.

[3] *RT Box User Manual*, Plexim GmbH, Online: https://www.plexim.com/sites/default/files/rtboxmanual.pdf
日本語版は以下から:
https://adv-auto.co.jp/products/plexim/manual.html

[4] *RT Box LaunchPad Interface Board*, Plexim GmbH, Online:
https://www.plexim.com/sites/default/files/launchpadinterfacemanual.pdf
日本語版は以下から:
https://adv-auto.co.jp/products/plexim/manual.html

改訂履歴:

Tutorial Version 1.0　　初版

Plexim

Pleximへの連絡方法:

☎ +41 44 533 51 00　Phone

+41 44 533 51 01　Fax

✉ Plexim GmbH　Mail
Technoparkstrasse 1
8005 Zurich
Switzerland

@ info@plexim.com　Email

http://www.plexim.com　Web

KESCO KEISOKU ENGINEERING SYSTEM
計測エンジニアリングシステム株式会社
https://kesco.co.jp

Embedded Code Generation Tutorial